

DOT/FAA/TC-21/24

Federal Aviation Administration
William J. Hughes Technical Center
Aviation Research Division
Atlantic City International Airport
New Jersey 08405

Certification Analysis Using the Uncontained Engine Debris Damage Assessment Model (UEDDAM)



U.S. Department of Transportation
Federal Aviation Administration

July 2021

Final report

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof. The U.S. Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report. The findings and conclusions in this report are those of the author(s) and do not necessarily represent the views of the funding agency. This document does not constitute FAA policy. Consult the FAA sponsoring organization listed on the Technical Documentation page as to its use.

This report is available at the Federal Aviation Administration William J. Hughes Technical Center's Full-Text Technical Reports page: actlibrary.tc.faa.gov in Adobe Acrobat portable document format (PDF).

Form DOT F 1700.7 (8-72)

Reproduction of completed page authorized

1. Report No. DOT/FAA/TC-21/24		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle Certification Analysis Using the Uncontained Engine Debris Damage Assessment Model (UEDDAM)				5. Report Date February 2021	
				6. Performing Organization Code	
7. Author(s) Ellen Phifer				8. Performing Organization Report No. SURVICE-TR-0896	
9. Performing Organization Name and Address SURVICE Engineering 4695 Millennium Drive Belcamp, MD 21017				10. Work Unit No. (TRAIS)	
				11. Contract or Grant No. N68936-17-D-0028	
12. Sponsoring Agency Name and Address Naval Air Warfare Center Weapons Division 1900 N. Knox Road, Mail Stop 6210 China Lake, CA 93555-6106				13. Type of Report and Period Covered March 2019 – February 2021	
				14. Sponsoring Agency Code	
15. Supplementary Notes					
16. Abstract <p>This report documents how to use the Uncontained Engine Debris Damage Assessment Model (UEDDAM) to perform rotor burst hazard analyses in support of aircraft design and certification activities. Using a generic twin-engine aircraft as a case study, general guidance and recommended processes are provided on the use and function of UEDDAM in aircraft design optimization and hazard mitigation. Additional uses of UEDDAM within the overarching domain of aircraft safety, to include hole size calculations in support of decompression analyses, are also discussed.</p>					
17. Key Words Uncontained, engine, UEDDAM, AC 20-128, hazard, debris, fragment, model, catastrophic, failure, prevention, certification, analysis, risk, damage			18. Distribution Statement This document is available to the U.S. public through the National Technical Information Service (NTIS), Springfield, Virginia 22161. This document is also available from the Federal Aviation Administration William J. Hughes Technical Center at actlibrary.tc.faa.gov .		
19. Security Classif. (of this report) Unclassified		20. Security Classif. (of this page) Unclassified		21. No. of Pages 59	22. Price

Contents

1	Introduction.....	1
1.1	Purpose.....	1
1.2	Background	1
1.3	Related documents	2
1.4	Integration of hazard analysis within the aircraft development cycle.....	3
2	Implementation of a multi-fragment analysis for rotor burst assessments.....	4
3	UEDDAM analysis approach.....	5
3.1	Establish	7
3.1.1	Flight phases	7
3.1.2	Functional hazard.....	7
3.1.3	Debris fragment definition.....	8
3.2	Assess.....	10
3.2.1	Damage modes and effects analysis	10
3.2.2	Hazard tree analysis	11
3.3	Define.....	13
3.4	Build.....	14
3.4.1	TARGET FILE	15
3.4.2	JTYPE FILE.....	16
3.4.3	PK FILE	17
3.4.4	MV FILE.....	17
3.4.5	Control File	18
3.4.6	DEBRIS FILE.....	19
3.4.7	NEARFIELD FILE.....	19
3.4.8	AUTOFAIL FILE.....	20
3.4.9	MATERIAL FILE	20
3.4.10	HOLEGRP FILE.....	20
3.5	Verify	20

3.6	Execute	22
3.7	Post-process.....	23
3.8	Analyze.....	26
3.9	Design optimization	30
4	Excursions.....	31
4.1	Penetration adjustments.....	31
4.2	Hole size calculations.....	33
4.3	Small component contributions.....	36
4.4	Debris Characterizations	41
5	Conclusion	44
6	References.....	45
A	UEDDAM version 6.0 user and analyst manual	A-1
B	Fault tree development tool (FTDT) version 1.0 user manual.....	B-1
C	UEDDAM visualizer version 3 user and analyst manual.....	C-1

Figures

Figure 1. UEDDAM analysis process.....	6
Figure 2. Portion of GT DMEA.....	11
Figure 3. GT fuel system hazard tree for flight phases 2, 3, 6, and 7.....	12
Figure 4. GT fuel system hazard tree developed using FTDT.....	12
Figure 5. Early design fuel system hazard trees.....	13
Figure 6. Aircraft geometry definition transition from early design to final concept.....	14
Figure 7. GT Aircraft geometry.....	16
Figure 8. FTDT visualization of the MV FILE for the GT case study.....	18
Figure 9. Variance and the relationship to exposure.....	19
Figure 10. Use of the UEDDAM Visualizer in verification.....	21
Figure 11. Small component hypersampling.....	22
Figure 12. Probability of hazard versus number of iterations for six debris types.....	22
Figure 13. Calculation of C_R , C_E , and C_A for the GT (MFRAG debris category).....	25
Figure 14. GT case study normalized average system contributions to mean aircraft risk (MFRAG debris category).....	26
Figure 15. GT case study overall rotor stage risk (DISK debris category).....	27
Figure 16. GT case study system contributions (DISK debris category).....	28
Figure 17. GT case study translational risk angle plot (DISK debris category).....	29
Figure 18. GT mean aircraft risk (MULTI debris category) as a function of engine nacelle thickness.....	31
Figure 19. Increase in mean aircraft risk across debris types for $\gamma=0.707$	33
Figure 20. Holesize input file for decompression analysis of the GT.....	34
Figure 21. Maximum and average decompression cumulative hole size data.....	35
Figure 22. Illustration of number of fragment penetrations.....	36
Figure 23. Computational run time comparisons.....	37
Figure 24. Increase in mean aircraft risk across debris types using hypersampling run control option.....	38
Figure 25. Example of potentially missed components.....	39
Figure 26. System contributions to DISK and MULTI debris probability of catastrophic hazard, with and without hypersampling run control option.....	40
Figure 27. Change in mean aircraft risk across debris types using different debris characterization parameters.....	43
Figure 28. MFRAG system contribution comparison using different debris characterization parameters.....	44

Tables

Table 1. Flight phases identified in AC 20-128A	7
Table 2. Debris model characteristics comparison	9
Table 3. Additional debris types considered.....	10
Table 4. GT mean engine and aircraft risks	24
Table 5. Debris types considered in DOT/FAA/AR-04/16	41
Table 6. MFRAG characterization data comparison	42

Acronyms

Acronym	Definition
AC	Advisory Circular
ACO	Aircraft Certification Office
APU	Auxiliary Power Unit
ARAC	Aviation Rulemaking Advisory Committee
BHN	Brinell Hardness Number
CAD	Computer-Aided Design
CFR	Code of Federal Regulations
DFM	Debris Fragment Model
DISK	Single One-Third Disk Fragment Debris Category
DMEA	Damage Modes and Effects Analysis
DoD	Department of Defense
DOT	Department of Transportation
Dp	Flight Phase Failure Distribution
ETOPS	Extended-range Twin Engine Operational Performance Standards
FAA	Federal Aviation Administration
FAR	Federal Aviation Regulation
FASTGEN	Fast Shotline Generator
FMEA	Failure Modes and Effects Analysis
FTDT	Fault Tree Development Tool
GT	Generic Twin-Engine Aircraft
HPC	High-Performance Computer
MFRAG	Intermediate Fragment Debris Category
MULTI	Disk Failure Debris Category
MULTI2	Blade Failure Debris Category
NAWCWD	Naval Air Warfare Center Weapons Division
NR	Non-Redundant
NTSB	National Transportation Safety Board
R	Redundant
SINGL	Small Fragment Debris Category
STL	Stereolithography
STNDRD	3,000 ft.-lb Kinetic Energy Fragment Debris Category
UEDDAM	Uncontained Engine Debris Damage Assessment Model

Executive summary

As part of the Federal Aviation Administration's (FAA) Aircraft Catastrophic Failure Prevention Program, the Uncontained Engine Debris Damage Assessment Model (UEDDAM) was developed to provide a standardized tool for uncontained engine rotor failure analysis.

UEDDAM is both a design tool capable of conducting aircraft configuration trade studies as well as a certification tool which can aid in showing compliance with Title 14 Code of Federal Regulations (CFR) 25.903(d)(1). The focus of this report is on the recommended process to be employed to assess an aircraft's risk to uncontained engine debris during the design and certification phases of its development life cycle. The process is illustrated by a case study, the subject of which is a generic twin-engine aircraft.

1 Introduction

1.1 Purpose

This report demonstrates the capabilities of the software tool UEDDAM, which can be used to perform a rotor burst analysis following the methodology defined in AC20-128A to satisfy the requirements of 23.901(f), 23.903(b)(1), 25.901(d) and 25.903(d)(1) of the Federal Aviation Regulations (FAR) pertaining to design precautions taken to minimize the hazards to an airplane in the event of uncontained engine or auxiliary power unit (APU) rotor failure. However, as stated in FAA Order 8110.4C, Type Certification (U.S. Department of Transportation Federal Aviation Administration, 2017), the FAA approves the data, not the analytical technique, so the FAA holds no list of acceptable analyses, approved computer codes, or standard formulas. Use of a well-established analysis technique is not enough to guarantee the validity of the result. The applicant must show the data are valid. Consequently, the Aircraft Certification Office (ACO) and its representatives are responsible for finding the data accurate, and applicable, and that the analysis does not violate the assumptions of the problem.

This effort was conducted under the Federal Aviation Administration (FAA) Aircraft Catastrophic Failure Prevention Program. Under the same program, the Naval Air Warfare Center Weapons Division (NAWCWD) at China Lake, CA was tasked with developing an analysis tool to evaluate the probability of hazard to an aircraft given engine rotor failure. To address this tasking, NAWCWD teamed with SURVICE Engineering to develop a computer program called the Uncontained Engine Debris Damage Assessment Model (UEDDAM). The model is intended to aid the design and certification of aircraft, providing a standardized tool (methodology) to conduct rotor burst assessments. As part of that effort, a generic twin-engine aircraft (GT) has been evaluated for engine rotor burst using UEDDAM. This case study, as presented herein, clearly illustrates the potential uses and benefits of UEDDAM in assessing rotor burst events and it is intended to provide a step-by-step process to be followed during the design and certification phases of an aircraft's development life cycle.

1.2 Background

The UEDDAM code was developed to address an industry/FAA need for an analytical tool capable of conducting rotor burst assessment that incorporates single and multiple fragment penetration to assess system-level hazards. UEDDAM was developed as a design tool capable of conducting aircraft configuration trade studies and as a certification tool to aid in showing compliance with the aforementioned Title 14 Code of Federal Regulations (CFR) requirements.

UEDDAM is based on vulnerability assessment codes used in industry during aircraft design and development to minimize the vulnerability of military aircraft to ballistic threats.

It is well understood that a rotor burst analysis is a complex one. UEDDAM was developed to provide useful tools to aide in conducting the analysis and presenting the results. A visualizer was developed, which provides visualization of the complex data and information generated from a UEDDAM run. It allows visualization of the aircraft geometry, debris hazard zones, debris trajectories, probability plots of the hazard levels, and translation risk angles.

Initial development of UEDDAM was performed in conjunction with the Aviation Rulemaking Advisory Committee (ARAC), Transport Airplane Engine Installation Group, Power Plant Installation Harmonization Working Group. This group initially revised Advisory Circular (AC) 20-128 to AC 20-128A in 1997. A follow-on effort to develop an analysis method to account for multiple small fragments was under way until it was placed in moratorium in 2003. The recent National Transportation Safety Board (NTSB) hearing on the 2016 Chicago O'Hare 767 uncontained engine failure and subsequent fire hull loss included criticism of the FAA for not completing that work, along with a recommendation that the effort be finalized. Further, the Australian Transportation Safety Board investigating the 2010 A380 uncontained engine failure in Singapore found that multiple small fragments disabled redundant safety systems for the outboard engine. The engine thrust setting could no longer be controlled after the inboard engine failed. FAA efforts in collaboration with Department of Defense (DoD) have continued to improve UEDDAM during the moratorium and this updated report highlights those improvements.

1.3 Related documents

The following documents (all downloadable from the FAA's Technical Library, accessed from https://www.faa.gov/about/office_org/headquarters_offices/ang/library/) relate to this report:

- "Design Considerations for Minimizing Hazards Caused by Uncontained Turbine Engine and Auxiliary Power Unit Rotor Failure," AC 20-128A
- "Small Engine Uncontained Debris Analysis," DOT/FAA/AR-99/7 (Frankenberger, Small Engine Uncontained Debris Analysis, 1999)
- "Large Engine Uncontained Debris Analysis," DOT/FAA/AR-99/11 (Frankenberger, Large Engine Uncontained Debris Analysis, 1999)

- “Large Engine Uncontained Debris Analysis – High-Bypass Ratio Engine Update,” DOT/FAA/TC-19/10 (Frankenberger, Large Engine Uncontained Debris Analysis - High-Bypass Ratio Engine Update, 2019)
- “Engine Debris Penetration Testing,” DOT/FAA/AR-99/19 (Manchor & Frankenberger, 1999)
- “Engine Debris Fuselage Penetration Testing, Phase I,” DOT/FAA/AR-01/27 (Lundin, Engine Debris Fuselage Penetration Testing, Phase I, 2001)
- “Engine Debris Fuselage Penetration Testing, Phase II,” DOT/FAA/AR-01/27, II (Lundin, Engine Debris Fuselage Penetration Testing, Phase II, 2002)
- “Advanced Aircraft Materials, Engine Debris Penetration Testing,” DOT/FAA/AR-03/37 (Lundin & Mueller, Advanced Aircraft Materials, Engine Debris Penetration Testing, 2005)
- “Uncontained Engine Debris Analysis Using the Uncontained Engine Debris Damage Assessment Model,” DOT/FAA/AR-04/16 (Seng, Manion, & Frankenberger, 2004)

1.4 Integration of hazard analysis within the aircraft development cycle

As a design tool, UEDDAM can provide early insight to the rotor burst hazard for a given aircraft configuration. Early in the aircraft development cycle a simple aircraft geometry can be developed. Aircraft skin, major structural elements, and flight critical components are easily modeled to provide a reasonable representation of the aircraft configuration. During this phase of an aircraft development program, the design is very fluid. A simple approach should be taken to model the configuration so that design trade studies can be conducted quickly. The use of very detailed models during the early development phase will significantly hamper the analyst’s ability to respond to configuration changes and provide meaningful recommendations to improve the design. Keeping the aircraft model simple, the analyst can conduct design trade studies to address rotor burst hazard minimization. Additional detail should be added to components showing high hazard contribution to refine the analysis. As the aircraft design matures, the model fidelity can be increased as detailed component information becomes available and as further analysis warrants.

As a certification tool, UEDDAM provides a standardized approach to conduct rotor burst hazard assessment. UEDDAM output provides insight to the rotor burst hazard in several ways. UEDDAM output can be used to develop a top-level 1-in-20 analysis to address compliance to

CFRs in accordance with the recommended methodology defined in AC20-128A. It also provides specific details at the system and component level. The output can be categorized by rotor or debris category, providing a high level of flexibility in viewing analysis results.

The benefits of performing a rotor burst hazard analysis at different times within the aircraft development cycle are significant. A rotor burst hazard analysis conducted during preliminary design can provide early insight into problem areas, or critical system placement concerns that can best be addressed early in the design process through the relocation of individual components. As the design matures, additional detail is added to the hazard analysis to refine the result and continue to monitor the status of the design. A final analysis is used to establish the aircraft performance to FAR requirements. Documentation of this process and the ability to show changes made to the aircraft design to improve the hazard probability can be used to illustrate means of minimization.

2 Implementation of a multi-fragment analysis for rotor burst assessments

The 2010 and 2016 incidents described briefly in section 1 are only two recent examples of why the FAA is highly concerned with the protection of aircraft from multiple fragment releases. To better understand the characteristics of the debris associated with uncontained failures, the FAA has collaborated with NAWCWD to develop a database containing detailed damage information of ~75 uncontained events obtained from individual contractors and agencies, as well as from incident sites (through coordination with the NTSB). From the ~700 data points collected, the average number of damages¹ observed per event is approximately 12, which speaks to the multi-fragment nature of uncontained events.

In addition to data collection and database development, NAWCWD conducted analysis of the data to define uncontained debris characteristics such as size, weight, velocity (energy), trajectory, and quantity of fragments for a given event. From this analysis arose the Debris Fragment Model (DFM). The DFM defines generic fragment characterizations for different types of events (e.g. compressor disk event, high-pressure turbine spacer – rim event, and fan blade event) and for different types of engines (i.e., high bypass ratio, low bypass ratio, and turboprop). For example, a fan blade event that emanates from a high bypass ratio engine is described in the DFM by the release and impact of not one single fragment measuring 1/3 blade length, but approximately 7 individual fragments varying in size from 10% to 50% of a blade length.

¹ A damage in this case refers to a hole in the aircraft structure or other subsystem injury.

Additional information regarding the DFM can be found in section 5.3.2.5 of Appendix A as well as in the three debris analysis FAA documents referenced in section 1.3 (Frankenberger, Small Engine Uncontained Debris Analysis, 1999) (Frankenberger, Large Engine Uncontained Debris Analysis, 1999) (Frankenberger, Large Engine Uncontained Debris Analysis - High-Bypass Ratio Engine Update, 2019).

The initial development of UEDDAM stemmed from the need for a tool that could more easily assess the multi-fragment failure scenarios defined in the DFM. Not only did the tool need to be able to handle the simultaneous release of multiple individually sized fragments with varying trajectories, but it also needed to perform numerous Monte Carlo iterations of each release. At the same time the DFM was being developed, a series of tests was conducted to develop a set of penetration equations and constants to model the impacts of engine debris to aircraft skin and components. These test series have been documented in the four engine debris penetration testing FAA documents referenced in section 1.3 (Manchor & Frankenberger, 1999) (Lundin, Engine Debris Fuselage Penetration Testing, Phase I, 2001) (Lundin, Engine Debris Fuselage Penetration Testing, Phase II, 2002) (Lundin & Mueller, Advanced Aircraft Materials, Engine Debris Penetration Testing, 2005). The result of these test series are the ballistic limit and residual velocity equations coded in UEDDAM along with empirically derived values for the shear constant (C_s) for five materials (Aluminum 2024, steel [Brinell Hardness Number (BHN) = 300], Titanium [BHN = 285], Inconel 625 LCF, and general composite).

3 UEDDAM analysis approach

As laid out in the AC 20-128A User's Manual, there are five steps in performing a complete safety and risk analysis of an aircraft. The steps are summarized as follows:

1. Establish (at the design definition) the functional hazards that can arise.
2. Establish a functional hazard tree.
3. Establish the aircraft geometry and fragment model.
4. Apply risk factors, such as phase of flight, and calculate the risk for each threat (i.e., hazardous combination as per the functional hazard tree) for each rotor stage.
5. Tabulate, summarize, and average all cases.

UEDDAM, in conjunction with its post-processors, helps an analyst perform steps 3 thru 5 of this process, thereby reducing his or her workload. The particular process used to perform a UEDDAM safety and risk assessment of an aircraft can be seen in Figure 1. Note that the first

two steps in the UEDDAM process (i.e., establish and assess) and the last two steps (i.e., post-process and analyze) correlate directly to steps 1 & 2 and 4 & 5, respectively, as laid out in the AC 20-128A User’s Manual. Step 3 in the AC 20-128A has been expanded into four steps in the UEDDAM process (i.e., define, build, verify, and execute). It is this central step where the bulk of the analysis occurs and that UEDDAM was designed to help simplify and standardize.

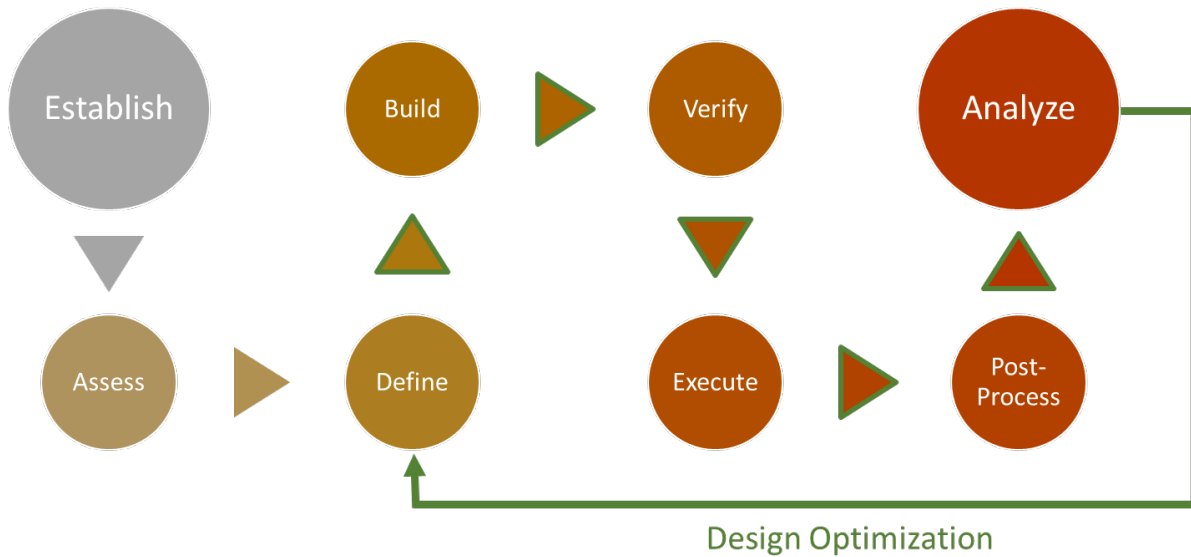


Figure 1. UEDDAM analysis process

In addition to the direct correlation between the UEDDAM analysis process and the steps identified in the appendix to the AC 20-128A, the UEDDAM analysis process highlights design optimization as a key element. Because 14 CFR 25.903(d) requires that the hazard from an uncontained engine debris release be minimized, it is important that this process be followed early in the aircraft development cycle and iterated upon during the various phases of the design. For example, when this process is introduced early on, different configurations of critical systems can be assessed and compared so that those configurations that perform best when faced with an uncontained debris release can be identified for the final design.

It is important to note that at each step in the UEDDAM analysis process fundamental assumptions will have to be made by the analyst. These assumptions should be documented clearly and accompanied by adequate rationale. By employing general design guidelines (i.e., critical system separation, redundancy, and shielding) up front, iterating the uncontained rotor analysis throughout the evolution of the design, and documenting all assumptions and rationale, a manufacturer can clearly illustrate their understanding of and adherence to the minimization requirement in the CFR.

The following sections will provide general guidance for how to step through the analysis process laid out in Figure 1 using the GT case study as an example. The intent of approaching the document in this way is to focus on the analysis process rather than getting into the specifics of how to build individual input files or the use of explicit keywords, thereby limiting the discussion to top-level concepts, approaches, and practices, keeping the document more succinct and manageable. For detailed or more in-depth discussions, the analyst will be pointed to the applicable section(s) in the UEDDAM User’s Manual (included as Appendix A to this document) for reference.

3.1 Establish

The first step of the UEDDAM analysis process is to establish the analysis objective (i.e., the particular flight phase(s) of interest, the functional hazard(s), and the debris category(ies) to be assessed).

3.1.1 Flight phases

The flight phases identified in the AC 20-128A (along with their associated failure distributions [Dp]) are provided in Table 1. Based on an average of all non-containments occurring in the US and UK, spanning the years 1966-1976, the Dp values in are the industry-accepted standards for the percentage of engine failures occurring within each phase of flight.

Table 1. Flight phases identified in AC 20-128A

Flight Phase	Phase Identification	Phase Failure Distribution (Dp)
1	Take-off before V1	35%
2	V1 to first power reduction (+30s)	20%
3	Climb	22%
4	Cruise	14%
5	Descent	3%
6	Approach thru landing	2%
7	Landing/Reverse	4%

These seven flight phases and associated failure distributions were assessed in the GT case study.

3.1.2 Functional hazard

In the context of performing an uncontained rotor burst analysis as part of design and certification of an aircraft, more often than not this means evaluating an aircraft for a single

functional hazard across multiple flight phases. The functional hazard considered in the GT case study was catastrophic failure, defined as the inability of the aircraft to continue safe flight and land. According to the AC 20-128A, “continued safe flight and landing means the airplane is capable of continued controlled flight and landing, possibly using emergency procedures and without exceptional pilot skill or strength, with conditions of considerably increased flightcrew workload and degraded flight characteristics of the airplane.” By focusing hazard minimization efforts on those components/systems whose loss would result in a catastrophic failure as opposed to attempting to minimize all functional hazards, resources can be more judiciously expended and the intent of the CFR is met.

3.1.3 Debris fragment definition

The final phase of establishing the analysis objective is to define the debris types that will be assessed. Again, the AC can (and should) be referenced. To satisfy the safety analysis recommended, the AC 20-128A requires use of the following engine failure model (“unless for the particular engine type concerned, relevant service experience, design data, test results or other evidence justify the use of a different model”):

- Single one-third disk fragment with a maximum dimension corresponding to one-third of the disk with one-third blade height and a fragment spread angle of $\pm 3^\circ$;
- Intermediate fragment with a maximum dimension corresponding to one-third of the bladed disk radius and a fragment spread angle of $\pm 5^\circ$;
- Small fragments with a maximum dimension corresponding to the tip half of the blade airfoil and a fragment spread angle of $\pm 15^\circ$; and
- Fan blade fragment with a maximum dimension corresponding to the blade tip with one-third the blade airfoil height and a fragment spread angle of $\pm 15^\circ$.

Because an airframe manufacturer will have ready access to specific engine element dimensions, these debris models can be developed in a relatively straightforward manner. For the GT case study, a generic multi-debris model was developed based on historical data collected from actual uncontained events and populated in a database (referred to as the DFM). As discussed in section 2, this model provides a multiple fragment definition by engine component and failure mode (e.g., fan disk failure). For the GT case study, the three debris models are referred to as DISK, MFRAG, and SINGL (where SINGL includes small blade fragments from a partial turbine blade fragment up to a 1/3 fan blade fragment). Table 2 provides a generalized comparison of AC 20-

128A requirements and the GT case study DFM-based characteristics (in terms of fragment spread angles).

Table 2. Debris model characteristics comparison

Debris Model		Fragment Spread Angle	
AC 20-128A	GT	AC 20-128A	DFM
Single One-Third Disk Fragment	DISK	$\pm 3^\circ$ for all engine stages	variable within $+5^\circ$ to -11° *
Intermediate Fragment	MFRAG	$\pm 5^\circ$ for all engine stages	variable within $+15^\circ$ to -25° *
Small Fragment	SINGL	$\pm 15^\circ$ for all engine stages	variable within $+20^\circ$ to -60° *
Fan Blade Fragment		$\pm 15^\circ$	$+20^\circ$ to -30°

* Depending on engine stage. The DFM defines the number, size, velocity, and trajectory of fragments per stage by failure mode.

In addition to the standard set of debris characteristics normally considered as part of an uncontained engine debris analysis, for the GT case study, an additional three debris models were selected. The first two (debris types MULTI and MULTI2) were developed from the DFM's characterization to represent a disk failure and blade failure, respectively. Rather than assessing a single one-third disk fragment, intermediate fragment, and small fragment independently of one another, the MULTI debris type combines all three fragment types into a single analysis and the MULTI2 debris type combines intermediate and small fragments into a single analysis. The third type of fragment that was examined is one that, no matter which rotor stage it originates from, impacts with a standard amount of kinetic energy. For the GT case study, a kinetic energy of 3,000 ft.-lb., which correlates to a 0.25-lb. fragment traveling at 900 fps, was selected as representative. This type of fragment, given the STNDRD nomenclature for the GT case study, can be used as a benchmark to assess the level of protection provided by the aircraft skin and structure or as a basis of comparison to assess one aircraft with another or to compare against other debris types. The characteristics for the three non-standard debris types are detailed in Table 3.

Table 3. Additional debris types considered

Debris Type	# of Fragments	Fragment Spread Angle
MULTI	1–2 disk segments & 5–21 fragments*	variable within +15° to -60°*
MULTI2	7–10 fragments*	variable within +35° to -50°*
STNDRD	1 fragment	+15° to -15°

* Depending on engine stage. The DFM defines the number, size, velocity, and trajectory of fragments per stage by failure mode.

3.2 Assess

The second step in the UEDDAM analysis process is to perform a flight phase criticality analysis of the total aircraft system, to include developing a damage modes and effects analysis (DMEA) and hazard trees.

3.2.1 Damage modes and effects analysis

A DMEA consists of identifying critical components, their function, and the impact of their damage-caused failures on the total aircraft system at each of the assessed flight phases. Critical components are those components that, when defeated either individually or jointly with other components, result in the loss of an essential aircraft function which would prevent continuous safe flight and landing. Both redundant and non-redundant critical components should be considered in a DMEA. Generally speaking, the following damage modes are typically considered: penetration, severance, jamming, cracking, deformation, etc. Obviously, the identification of critical components, their failures, and their impacts on the total aircraft system will depend on the objective established in the first step of the UEDDAM analysis process. Typical sources of information an analyst can consult to develop a DMEA include structural safety analyses, failure modes and effects analyses (FMEA), and system breakdown structures.

For the GT case study, a DMEA was developed using input from industry and general knowledge of the components, operations, and functions required of typical fixed-wing aircraft platforms. Figure 2 illustrates a portion of the flight phase criticality analysis developed in Microsoft Excel for the GT case study, where the redundancy columns indicate whether a component is redundant (R), non-redundant (NR), or not critical (blank) with respect to maintaining continued safe flight and/or landing during that particular phase of flight.

Component	Function	Damage Mode	Failure Mode	Next Higher Effects	End Effects	Redundancy						
						Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6	Phase 7
Hydraulic system 1 accumulator	Maintains consistent pressure and fluid for hydraulic system 1 pumps	Penetration	Leakage	Loss of hydraulic fluid	Loss of hydraulic system 1	R	R	R	R	R	R	R
Right wing fuel tank	Contains fuel	Penetration into lower 1/3 of fuel	Loss of fuel supply to primary & secondary pumps for engine	Loss of fuel supply to engine 2	No. 2 engine flameout		R	R		NR	R	R

Figure 2. Portion of GT DMEA

3.2.2 Hazard tree analysis

The second step in the UEDDAM analysis process also suggests the development of hazard trees, which are visual representations of the critical components and their redundancies. An aircraft system in this representation is defeated when the tree is completely “cut.” An analyst can construct these trees directly from a completed DMEA, but might also be able to reference fault tree analysis diagrams developed by safety and/or reliability analysts within their organization.

For the GT case study, hazard trees were developed directly from the DMEA and verified through industry review. An example hazard tree from the GT analysis is illustrated in Figure 3. In this example, it is clear that defeat or loss of fuel from a single wing tank is insufficient to result in uncontrolled flight or inability to land during flight phases 2, 3, 6, and 7; however, couple that loss with the defeat or loss of both primary and secondary pumps from the opposite tank, and the aircraft would suffer a catastrophic hazard.

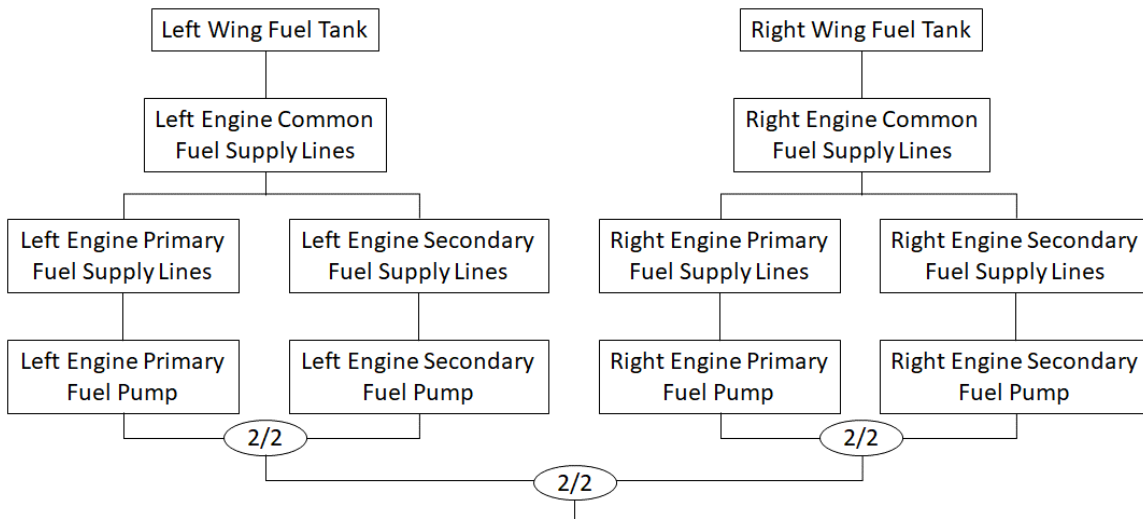


Figure 3. GT fuel system hazard tree for flight phases 2, 3, 6, and 7

Viewed another way, the same hazard tree was generated using a fault tree development and visualization utility developed for use with UEDDAM called the Fault Tree Development Tool (FTDT). This hazard tree, shown in Figure 4, conveys the same information as that illustrated in Figure 3, but follows a format that was devised to be more familiar to people in the aircraft design and manufacturing industry. In addition, the FTDT can be used to graphically develop hazard trees that can then be exported to a file format compatible with the UEDDAM code. Additional information regarding the FTDT utility can be found later in section 3.4 as well as in Appendix B.

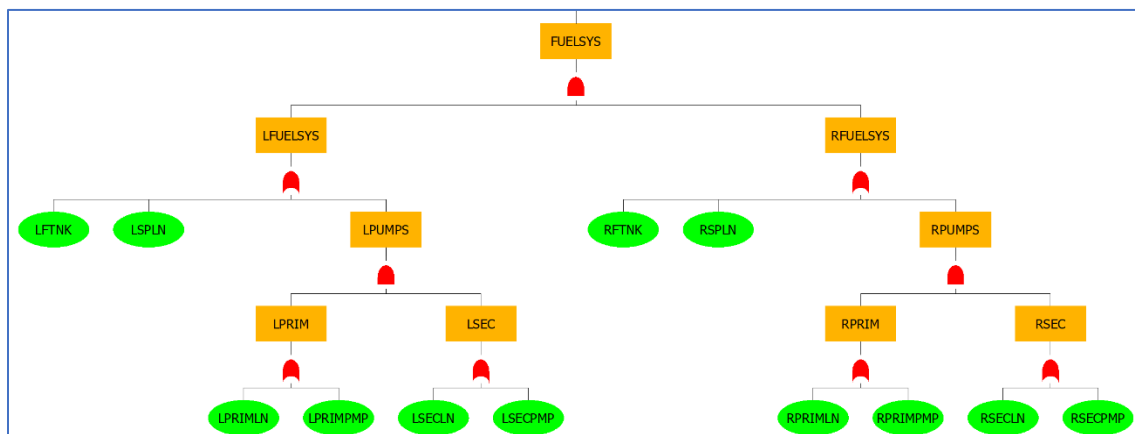


Figure 4. GT fuel system hazard tree developed using FTDT

It is important to note that as the design of the aircraft grows in complexity, so too will the DMEA and hazard trees. For example, early on in the design, there may only be a sense that there will be two redundant systems providing fuel to a twin-engine aircraft, but the intricacies of

fuel routing, crossfeed, pumping, etc. have yet to be defined. A UEDDAM rotor burst analysis can still be performed at this early stage, however. The DMEA and hazard trees will merely be simpler in nature. In this case, the fuel system hazard tree might look something akin to what is illustrated in Figure 5 as opposed to what was shown in Figures 3 and 4.

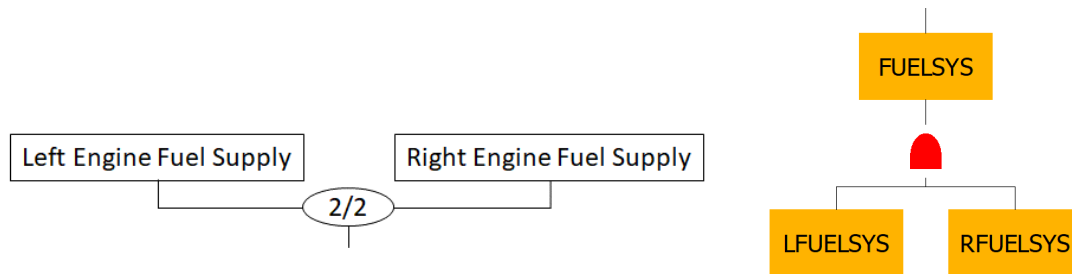


Figure 5. Early design fuel system hazard trees

3.3 Define

The third step in the UEDDAM analysis process is to define the aircraft geometry in terms of critical systems and major structure. Critical systems were defined in the previous step through development of the DMEA and hazard trees. However, in this step, the aircraft geometry definition should begin to consider not only all critical components within the rotor disk debris zone, but all shielding and air vehicle surface components as well. In the case of the GT, for example, this means that in addition to all flight control, propulsion, fuel, and structural elements identified in the DMEA, the fuselage and wing skin, engine nacelles and pylons, and floor structure need to be defined.

It is important to note that as the complexity of the design increases over time, the complexity of the aircraft geometry will also increase (see Figure 6). As was discussed in the previous section, the lack of complexity does not preclude execution of a UEDDAM rotor burst analysis early in the design process. On the contrary, establishing early definition of critical systems and shielding/air vehicle surface components from the point of view of an uncontained rotor burst event can only help to minimize the overall hazard.

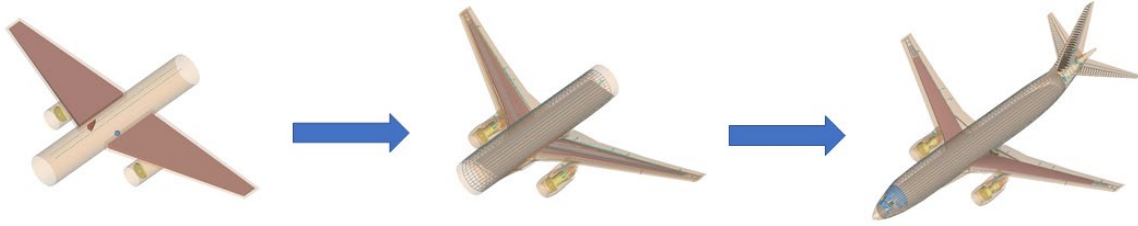


Figure 6. Aircraft geometry definition transition from early design to final concept

3.4 Build

The fourth step in the UEDDAM analysis process is to build the necessary input files for a UEDDAM assessment. The format and content of each is covered in Appendix A (beginning in section 5), but for completeness, a brief description of the various input files has been included here:

- TARGET FILE – A three-dimensional representation of the air vehicle in the geometry format known as FASTGEN (Fast Shotline Generator). The data used to develop this input file is gathered in step three of the UEDDAM analysis process.
- JTYPE FILE – An input file assigning properties (i.e., name, material type, density, and criticality) to individual components. Note that portions of this input file are derived from information contained in the DMEA, which is developed in step two of the UEDDAM analysis process.
- PK FILE – An input file containing probability of aircraft hazard, given component hit tables for all critical components.
- MV FILE – A textual representation of the hazard trees developed in step two of the UEDDAM analysis process.
- Control File – An input file containing run specifications for the main UEDDAM code.
- DEBRIS FILE – An input file containing the information characterizing the specific debris for use in a UEDDAM assessment. The data used to populate this input file is established in step one of the UEDDAM analysis process.
- NEARFIELD FILE – A listing of geometric components that are treated as so-called “near-field” components (i.e., the small, non-shielding components internal to an engine nacelle).
- AUTOFAIL FILE – A listing of geometric components and systems that are driven by an engine and that should therefore automatically fail given an uncontained event.

- MATERIAL FILE – An input file identifying material names and associated properties required for penetration calculations.
- HOLEGRP FILE – A listing of geometric components for which UEDDAM will generate hole size statistics.

3.4.1 TARGET FILE

The aircraft geometry illustrated in Figure 7 was developed for the GT case study. The TARGET FILE consists of individual components grouped by system into nine standard assemblies: skin, propulsion, personnel, flight controls, fuel, landing gear, structure, electrical, and miscellaneous. While the majority of the components in the TARGET FILE were modeled using FASTGEN primitives, some of the more intricate components in terms of contour (e.g., skin) were modeled with a computer-aided design (CAD) package and then converted. To aid aircraft manufacturers in development of the TARGET FILE, two utilities are included with UEDDAM – a stereolithography (STL) converter and a NASTRAN converter (specifically Patran). Both utilities convert geometric entities existing in another format to FASTGEN (see section 5.1.2 of Appendix A for more information).

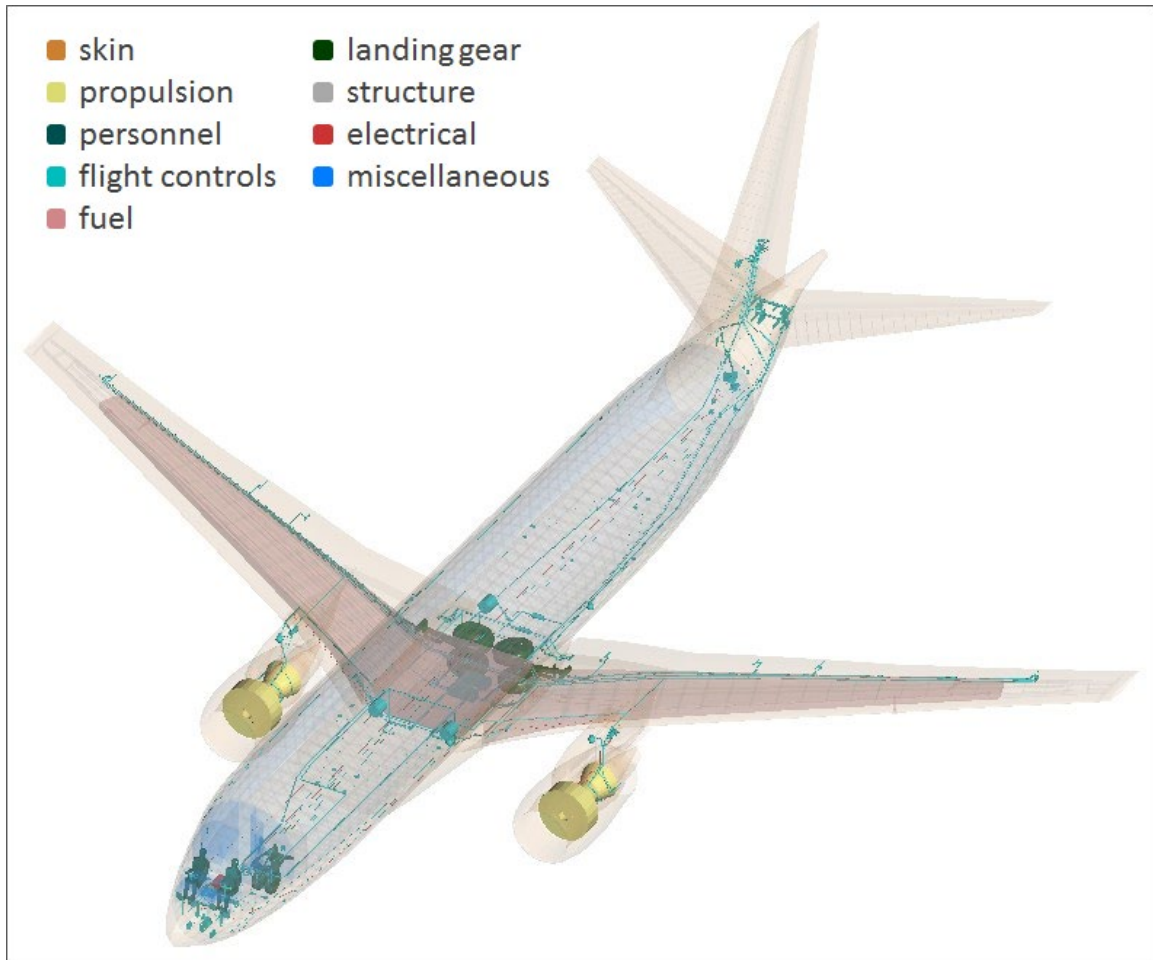


Figure 7. GT Aircraft geometry

3.4.2 JTYPE FILE

The component properties file, otherwise known as the JTYPE FILE, describes component information such as material type, density (or normal thickness), and criticality. For the GT case study, most of the components were modeled as either aluminum or steel, with the occasional component denoted as either titanium, composite, or some sort of fluid (e.g., water). Similarly, most of the components were modeled true to their form and thus have an assigned density of 100% (indicating that UEDDAM should perform penetration calculations through the component without adjusting the observed thickness). The final section of the JTYPE FILE for the GT case study attributes a marker to each phase of flight in which a component is considered critical. The marker points to a table in the next input file to be discussed, the PK FILE.

3.4.3 PK FILE

For the GT case study, only two tables were used to quantify probability of aircraft hazard given component hit in the PK FILE. The first table was used for critical components that were considered “soft” (i.e., if struck by any debris larger than 10 grains [where 7,000 grains is equivalent to 1 pound], the component would be considered to have been defeated). The second table was used for critical components that were considered “hard” (i.e., failure of the component would only occur if the mass of the striking debris were greater than 7,000 grains). Using generic inputs such as these allow an analyst to look at worst-case scenarios and limit the amount of detailed modeling and/or testing necessary.

3.4.4 MV FILE

The MV FILE contains the hazard trees and is likely the most complicated file for an analyst to build due to its textual nature and formatting restrictions. Thus, to help with the development of this file, two tools are included with the distribution of UEDDAM: the MAC tool and a tool called FTDT. Usage of the MAC tool is described in section 5.2.3 of Appendix A, while FTDT is covered separately in Appendix B and was discussed briefly in section 3.2. Figure 8 shows an example of what the MV FILE for the GT case study looks like when viewed in FTDT, where the black outlined systems have been collapsed for ease of visualization.

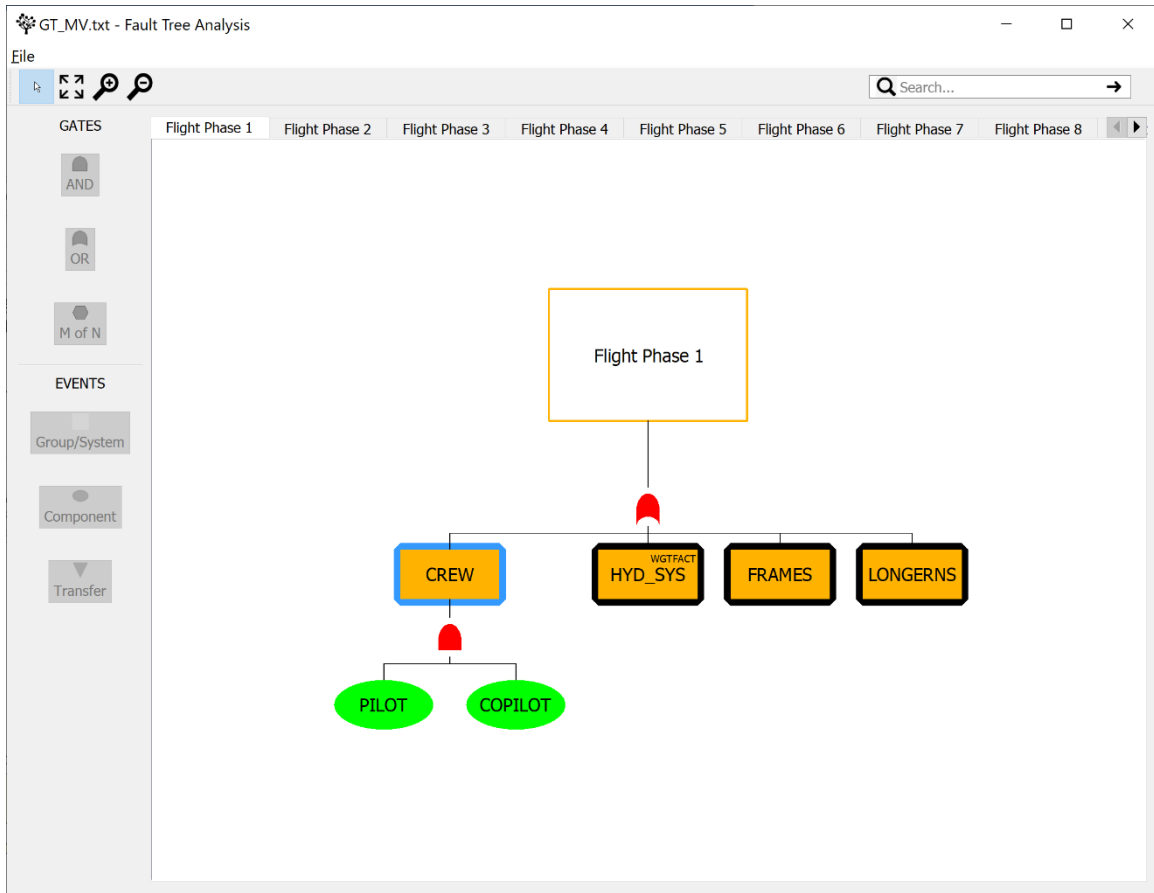


Figure 8. FTDT visualization of the MV FILE for the GT case study

3.4.5 Control File

As stated previously, the Control File provides the run specifications necessary for execution of the main UEDDAM code. It includes entries to define the name and location of the input files to use, the specific parameters of an event (e.g., geometric location of the undamaged disk origin, number of and angular spacing between adjacent release points, and number of Monte Carlo iterations to perform at each release point), the name of the output files to generate, and a host of optional run-time controls. Only two of the optional run-time controls were specified for use in the GT case study: the multiple processor flag and the resource estimation flag. The first of these two controls specifies among how many processors a UEDDAM analysis can be split, allowing the user to take advantage of a multi-processor computer and thereby decreasing overall run times. The second control indicates whether UEDDAM should provide a rough estimate of the anticipated run time and approximate disk space usage prior to beginning a run. Other run-time control options of note include the target reduction flag, small component hypersampling, high-performance computer (HPC) cluster flag, and gamma flag. Each of these controls is described in detail in section 5.3.1 of Appendix A.

3.4.6 DEBRIS FILE

The DEBRIS FILE contains information describing the specific debris (the characteristics of which were defined in the first step of the process) for use in a UEDDAM assessment. One of the inputs in this file is the debris characterization record, which, among other things, specifies a variance angle to be used for the debris. The variance is a delta angle in the plane of rotation around the nominal release angle in which a random number draw will be used to determine the actual release angle used for analysis. The diagram on the left of Figure 9 illustrates what is meant by variance in the context of a UEDDAM analysis, where the view is down the centerline of an engine. The illustration on the right of Figure 9 demonstrates how the variance angle input relates to fuselage exposure in a nominal, wing-mounted twin-engine aircraft. Fan and compressor events for these types of aircraft have historically demonstrated fuselage exposure angles of $\sim 40^\circ$. Thus, for the GT case study, a variance angle of 20° was used for all fan/compressor debris characterizations. However, because turbine events have exhibited exposure angles of over 70° (i.e., debris from a single event has impacted the fuselage as well as the wing), for the GT case study, a variance angle of 35° was used for all turbine events.

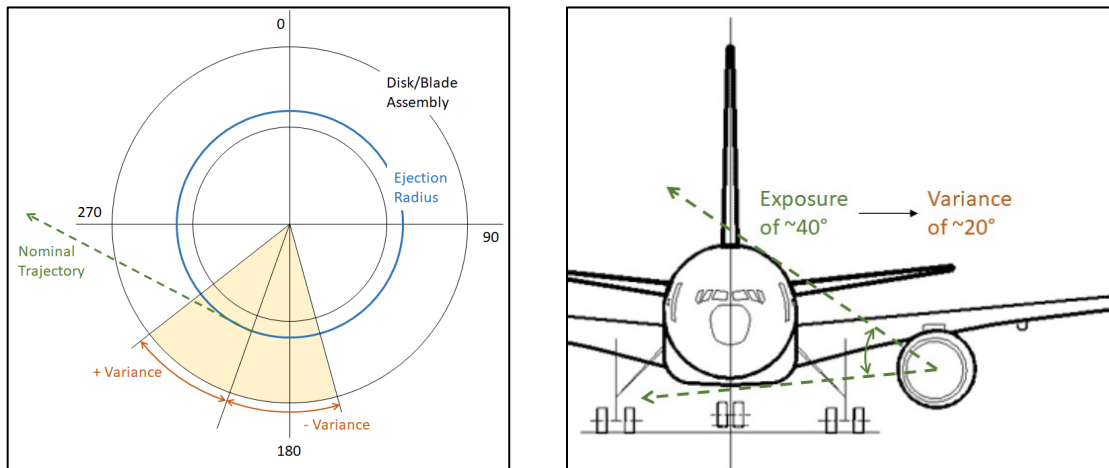


Figure 9. Variance and the relationship to exposure

3.4.7 NEARFIELD FILE

The NEARFIELD FILE for the GT case study consisted of those components (both critical and non-critical alike) internal to the engine nacelles. The following types of components were included as nearfield for each engine: engine disks and blades, digital electronic control, accessory gearbox, engine case, thrust reverser actuator, hydraulic pump, fuel pump, and electrical generator.

3.4.8 AUTOFAIL FILE

The AUTOFAIL FILE for the GT case study included those critical components and systems that would cease to function given an uncontained release. For each engine, these components included modeled internal components (i.e., disks and blades), engine casing, hydraulic systems, digital electronic control, accessory gearbox, fuel pump, and electrical cabling.

3.4.9 MATERIAL FILE

For the GT case study, the standard MATERIAL FILE distributed with UEDDAM and consisting of the five basic material types discussed in section 2 (i.e., steel, titanium, aluminum, composite, and Inconel) was used without modification.

3.4.10 HOLEGRP FILE

As stated previously, the HOLEGRP FILE contains a list of geometric components for which UEDDAM will generate hole size statistics (e.g., maximum hole area in square inches). For the baseline GT case study described herein, this input file was not used. An excursion using this input file is presented in Section 4.

3.5 Verify

The fifth step of the UEDDAM analysis process is to perform a verification of the various input files created. Setting up a simplified UEDDAM execution and troubleshooting the resulting output until the run is successful can accomplish the first half of this step in the process (troubleshooting is discussed in section 6.3 of Appendix A). The second half is accomplished by analyzing the results from this execution.

It is during this step in the analysis process that the UEDDAM Visualizer (a utility developed to allow visualization of the complex data and information generated from a UEDDAM run) can be most useful for an analyst. The utility (the user's manual for which is included as Appendix C) allows visualization of the aircraft geometry, debris hazard zones, debris trajectories, probability plots of the hazard levels, and translational risk angles, all of which can help detect logic and/or typing errors. For example, Figure 10 illustrates the display of the debris trajectories associated with a single release of the DISK debris type from the GT's left engine fan section. A single element of the bundle that makes up the debris is highlighted in blue and the specific components that the ray passes through are listed in the box labeled Shotline Data. By cycling through the individual elements of the debris bundle, an analyst would be able to determine that while multiple structural and hydraulic system components were damaged by this event, the logic

dictated by the hazard trees is coded correctly such that this event results in a probability of hazard of 0.0.

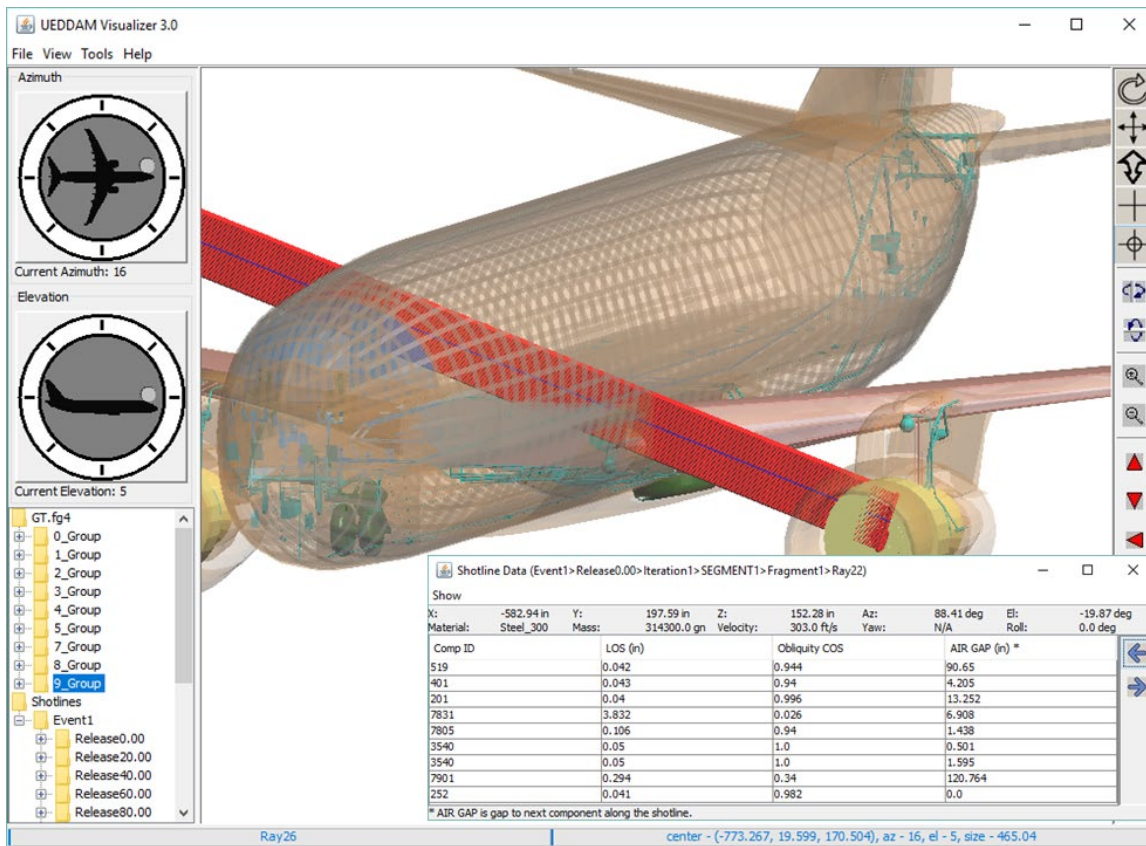


Figure 10. Use of the UEDDAM Visualizer in verification

Similarly, an analyst might discover during this investigation that a component that should have been impacted by this event was being missed due to its size relative to the debris bundle element spacing. In this case, the analyst can add a run-time option to the Control File that instructs UEDDAM to generate one or more additional debris bundle elements to ensure all components that should be impacted by a debris event actually are. Figure 11 illustrates this process, referred to as hypersampling. For additional information regarding hypersampling, see sections 5.3.1.5 and 9.10 in Appendix A. Note that the baseline GT case study did not implement this run-time control option as it was deemed unnecessary due to the debris bundle element spacing used. An excursion that examines the effect of its use (on both run times and results) is presented in Section 4.

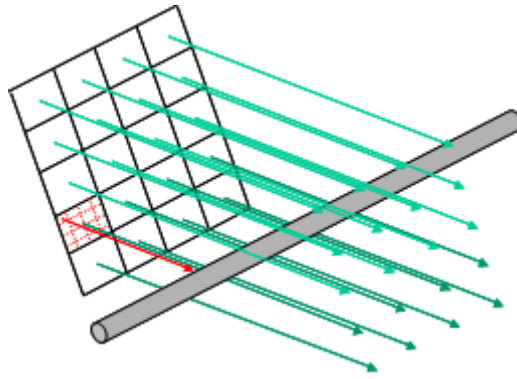


Figure 11. Small component hypersampling

In addition to verifying the correctness of the input files and because UEDDAM uses a Monte Carlo approach to debris releases, this step in the UEDDAM analysis process is also when an analyst should determine the appropriate number of UEDDAM iterations of each release point necessary for a stable solution. For the GT case study, this determination was made by examining the resultant total aircraft-level probability of hazard due to debris being released from one of the engines' fans across two flight phases. The resulting analysis (see Figure 12) determined that for large debris types (e.g., DISK), the necessary number of iterations was ~ 100 whereas for smaller debris types (e.g., MFRAG and SINGL), the required number of iterations was larger (i.e., ~ 170).

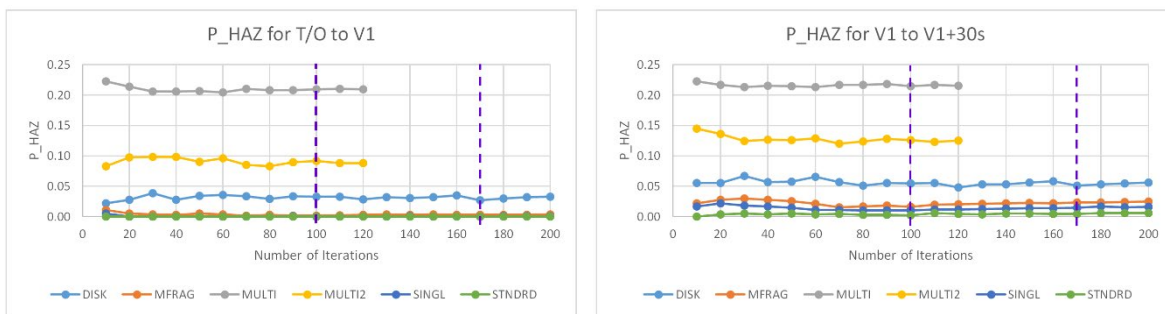


Figure 12. Probability of hazard versus number of iterations for six debris types

3.6 Execute

The sixth step of the UEDDAM analysis process is to set up and execute a complete set of production runs for post-processing and analysis. Recall that step 4 of the AC 20-128A safety and risk analysis process, outlined at the beginning of section 2, indicates that a complete analysis must consist of an assessment of the risk for each debris type and for each rotor stage across all phases of flight. Therefore, once step five (verify) is complete, a rotor burst analyst must assemble all the necessary inputs and execute the analysis for a minimum of # debris types

\times # engines \times # rotor stages separate instantiations. For the GT case study, a total of 108 separate executions of UEDDAM were required ($3 \times 2 \times 18$). This process has been made simpler for a UEDDAM analyst through the development of a Microsoft Excel utility called Run_Setup.xls. The spreadsheet with embedded macro assists an analyst in setting up a file structure and series of UEDDAM Control Files with user-defined options and output file specifications. Additional information regarding the run setup utility can be found in sections 6.4 and 9.11 of Appendix A.

One other important UEDDAM feature to note as part of this step in the analysis process is the ability of UEDDAM to take advantage of multi-processor and high-performance computers (HPC). Coupled with the simple advance and availability of computing technology over the past decade, these multi-threading capabilities have reduced UEDDAM run times on standard PCs significantly (i.e., from days/weeks to minutes/hours). For example, the entire set of DISK debris runs performed for the GT case study was executed on a Dell Precision 7720 laptop with eight 2.90 GHz processors (6 of which were used for execution) and 64.0 GB RAM in a matter of 12.5 hours. Using the HPC option (4 nodes, 112 cores), this same analysis was executed in just under 6 hours. Additional information regarding the multi-threading and HPC options available to the UEDDAM analyst can be found in sections 5.3.1.6 and 5.3.1.7 of Appendix A.

3.7 Post-process

Step seven in the UEDDAM analysis process is post-processing the results obtained from step six. UEDDAM, upon successful completion, supplies the user with information that can be used to aid in showing compliance with the AC 20-128A requirements (and more). Recall that UEDDAM and its post-processors assist a rotor burst analyst with accomplishing steps 3 through 5 of the AC 20-128A process, as outlined at the beginning of section 2. Steps 3 through 5 can be more explicitly written as:

- a) For each rotor stage, establish the threat windows (i.e., the translational and spread risk angles) where, due to a combination of individual damages, a catastrophic risk exists.
- b) For each rotor stage, calculate the risk factor for all critical hazards.
- c) For each rotor stage, calculate the combined risk factor (C_{rf}) for the phase of flight and average over 360° .
- d) Calculate the overall rotor stage risk (C_R) for each stage over all n flight phases (incorporating Dp).

$$C_R = (Dp_1 \times C_{rf_1}) + (Dp_2 \times C_{rf_2}) + \dots + (Dp_n \times C_{rf_n})$$

e) Average all m rotor stages to obtain the mean engine risk (C_E).

$$C_E = \frac{C_{R1} + C_{R2} + \dots + C_{Rm}}{m}$$

f) Average all k engines to obtain the mean aircraft risk (C_A).

$$C_A = \frac{C_{E1} + C_{E2} + \dots + C_{Ek}}{k}$$

UEDDAM internally performs the calculations in steps a) through c); in particular, C_{rf} can be taken directly from the SUMMARY FILE (one of the UEDDAM output files² available to an analyst). Thru simple post-processing (i.e., applying the phase failure distributions [Dp] identified in Table 1 and then performing a series of averages), steps d) thru f) can be calculated, resulting in a mean aircraft risk for the particular debris category of interest.

For the GT case study, steps d) thru f) were calculated for each of the six debris categories of interest (DISK, MFRAG, SINGL, MULTI, MULTI2, and STNDRD) as illustrated in Figure 13. Mean engine and aircraft risks for all six debris categories assessed in the GT case study are provided in Table 4.

Table 4. GT mean engine and aircraft risks

Risk	Debris Category					
	SINGL	MFRAG	DISK	MULTI	MULTI2	STNDRD
C_{E1}	0.001	0.018	0.049	0.068	0.019	0.004
C_{E2}	0.003	0.021	0.050	0.066	0.022	0.005
C_A	0.002	0.020	0.050	0.067	0.021	0.004

² UEDDAM output files are discussed in more detail in section 7 of Appendix A.

		Flight Mode							
		1	2	3	4	5	6	7	
	Dp	0.35	0.20	0.22	0.14	0.03	0.02	0.04	
Engine	Rotor	Combined Risk Factor (C_{rf})							C_R
1	Fan	0.0033	0.0232	0.0212	0.0137	0.0180	0.0180	0.0180	0.0140
	LPC-1	0.0072	0.0261	0.0183	0.0085	0.0085	0.0173	0.0173	0.0143
	LPC-2	0.0069	0.0275	0.0199	0.0098	0.0098	0.0180	0.0180	0.0150
	LPC-3	0.0075	0.0284	0.0222	0.0108	0.0108	0.0193	0.0193	0.0162
	HPC-1	0.0085	0.0337	0.0291	0.0147	0.0163	0.0252	0.0252	0.0202
	HPC-2	0.0085	0.0320	0.0288	0.0154	0.0176	0.0248	0.0248	0.0199
	HPC-3	0.0082	0.0304	0.0268	0.0121	0.0167	0.0252	0.0252	0.0185
	HPC-4	0.0085	0.0330	0.0291	0.0131	0.0193	0.0268	0.0268	0.0200
	HPC-5	0.0085	0.0343	0.0301	0.0137	0.0212	0.0268	0.0268	0.0206
	HPC-6	0.0085	0.0343	0.0307	0.0147	0.0232	0.0252	0.0252	0.0209
	HPC-7	0.0088	0.0363	0.0327	0.0167	0.0284	0.0261	0.0261	0.0223
	HPC-8	0.0069	0.0373	0.0340	0.0176	0.0301	0.0248	0.0248	0.0222
	HPC-9	0.0069	0.0363	0.0324	0.0163	0.0304	0.0235	0.0235	0.0214
	HPT	0.0088	0.0475	0.0402	0.0299	0.0642	0.0328	0.0328	0.0295
	LPT-1	0.0000	0.0265	0.0265	0.0049	0.0451	0.0240	0.0240	0.0146
	LPT-2	0.0000	0.0235	0.0235	0.0049	0.0451	0.0211	0.0211	0.0132
LPT-3	0.0000	0.0260	0.0260	0.0054	0.0456	0.0235	0.0235	0.0144	
LPT-4	0.0000	0.0250	0.0250	0.0049	0.0466	0.0225	0.0225	0.0139	
	Average	Mean Engine Risk (C_{E1})							0.0184
2	Fan	0.0042	0.0248	0.0216	0.0108	0.0206	0.0193	0.0193	0.0145
	LPC-1	0.0134	0.0304	0.0265	0.0144	0.0144	0.0265	0.0265	0.0206
	LPC-2	0.0131	0.0307	0.0271	0.0147	0.0147	0.0268	0.0268	0.0208
	LPC-3	0.0124	0.0307	0.0278	0.0144	0.0144	0.0271	0.0271	0.0207
	HPC-1	0.0121	0.0350	0.0320	0.0147	0.0154	0.0307	0.0307	0.0226
	HPC-2	0.0121	0.0379	0.0346	0.0170	0.0193	0.0327	0.0327	0.0244
	HPC-3	0.0118	0.0389	0.0359	0.0186	0.0229	0.0333	0.0333	0.0251
	HPC-4	0.0124	0.0405	0.0376	0.0206	0.0255	0.0350	0.0350	0.0265
	HPC-5	0.0118	0.0415	0.0386	0.0216	0.0278	0.0356	0.0356	0.0269
	HPC-6	0.0121	0.0412	0.0382	0.0203	0.0307	0.0350	0.0350	0.0267
	HPC-7	0.0118	0.0389	0.0373	0.0196	0.0330	0.0340	0.0340	0.0259
	HPC-8	0.0121	0.0405	0.0402	0.0209	0.0376	0.0373	0.0373	0.0275
	HPC-9	0.0124	0.0412	0.0409	0.0206	0.0395	0.0376	0.0376	0.0279
	HPT	0.0108	0.0338	0.0309	0.0201	0.0926	0.0309	0.0309	0.0248
	LPT-1	0.0000	0.0211	0.0196	0.0029	0.0647	0.0196	0.0196	0.0121
	LPT-2	0.0000	0.0206	0.0191	0.0025	0.0662	0.0191	0.0191	0.0118
LPT-3	0.0000	0.0181	0.0172	0.0015	0.0676	0.0172	0.0172	0.0107	
LPT-4	0.0000	0.0191	0.0172	0.0020	0.0696	0.0172	0.0172	0.0110	
	Average	Mean Engine Risk (C_{E2})							0.0211
								Mean Aircraft Risk (C_A)	0.0198

Figure 13. Calculation of C_R , C_E , and C_A for the GT (MFRAG debris category)

Because it is equally important to know the various system and component contributions to aircraft risk, a Java-based post-processing tool called Analysis_Tool has been developed for use with UEDDAM. This utility summarizes and provides averages for multiple UEDDAM output files using the same d) thru f) calculations described previously, thereby further simplifying an analyst's workload. An example of the data obtained using this utility is shown in Figure 14, which illustrates the various system contributions to mean aircraft risk from an MFRAG event,

for which the largest contributors can be seen to be structural and propulsion related. Additional details regarding Analysis_Tool can be found in section 8 of Appendix A.

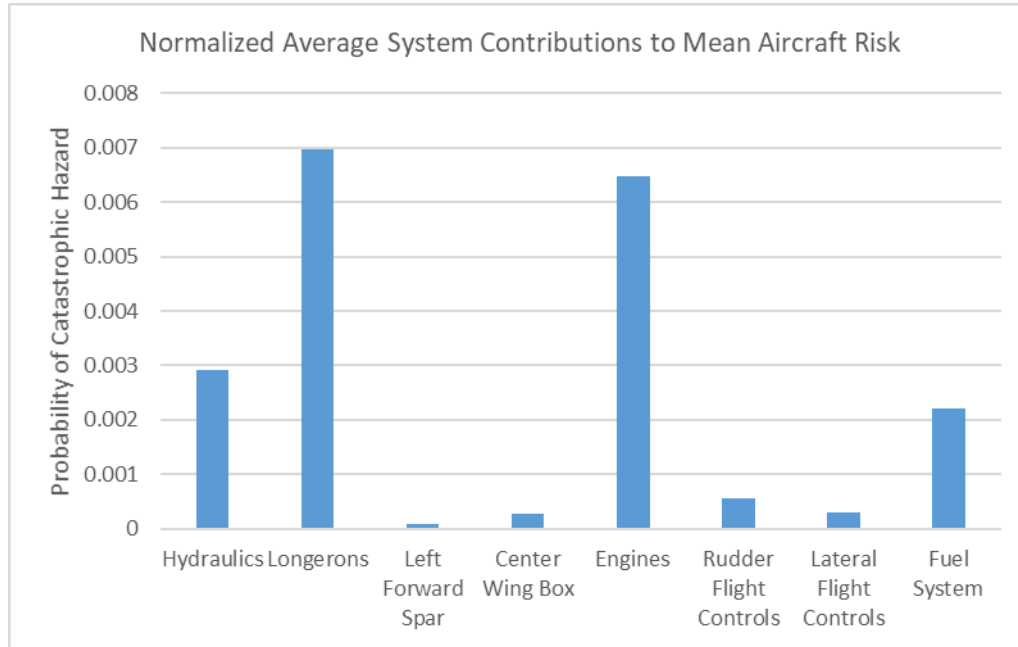


Figure 14. GT case study normalized average system contributions to mean aircraft risk (MFRAG debris category)

3.8 Analyze

Step eight of the UEDDAM analysis process is detailed examination of the post-processed results. One metric that can be investigated is the comparison of C_A values with the certification requirements in the AC 20-128A. For the GT case study, the acceptable risk level criteria from the AC 20-128A can be summarized as $C_{A_DISK} \leq 0.05$ (1-in-20), $C_{A_MFRAG} \leq 0.025$ (1-in-40), and $C_{A_SINGL} \leq 0.05$ ($2 \times$ corresponding average criterion). Comparing these requirements to the results in Table 4 illustrate that the GT aircraft, as modeled and analyzed, is just capable of meeting the requirements. However, rather than rest on these numbers, it would also be prudent to investigate the major contributors to the mean aircraft risk and establish whether or not anything can be done in terms of design or mitigation to minimize the risk even further. An example of this type of analysis is illustrated in Figure 15. The graph in this figure is of the overall stage risk (C_R) determined for each rotor stage and each engine for the DISK debris type. From this graph, it is clear that the largest contributors to mean aircraft risk are the number one engine low-pressure turbine stages, at least for the release of a single, one-third disk fragment.

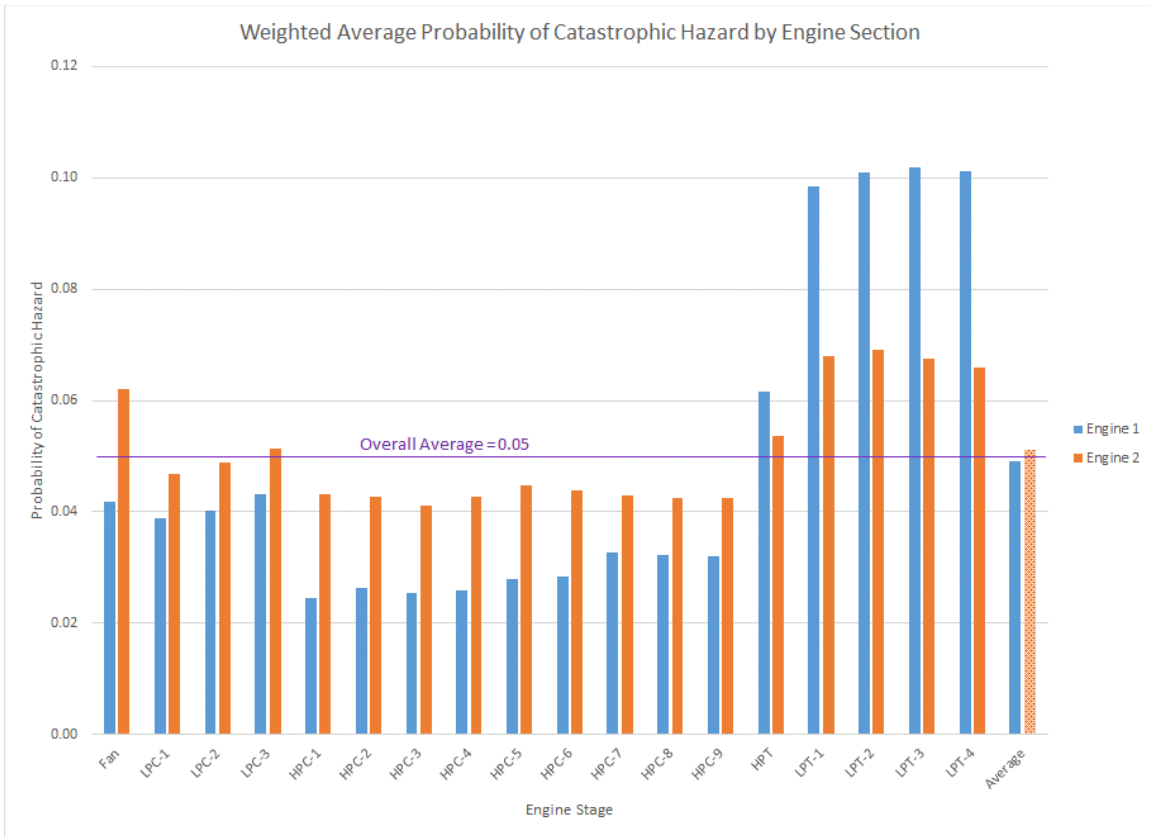


Figure 15. GT case study overall rotor stage risk (DISK debris category)

Delving even further into the results from the GT case study, a step that is most important early on in the design cycle, Figure 16 highlights the individual system contributions that are leading to catastrophic losses for these rotor stages. From these graphs, as well as examination of the geometry, DMEA, and hazard trees associated with the GT case study, it can be determined that the largest contributor to failure is loss of consecutive load-bearing structural members in the fuselage. Armed with this knowledge, a design team can investigate potential mitigation techniques to further reduce the risks associated with an uncontained release.

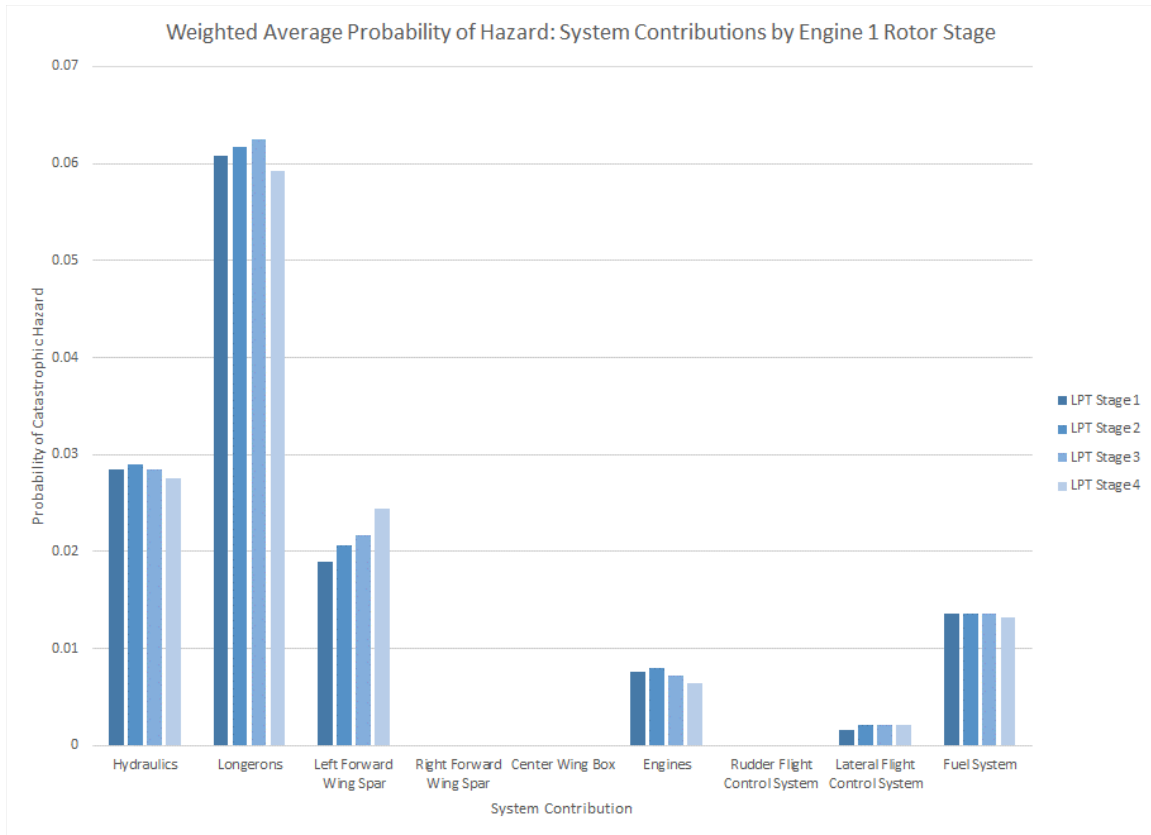


Figure 16. GT case study system contributions (DISK debris category)

The results of the more realistic assessments of a disk and a blade failure (i.e., the MULTI and MULTI2 debris categories, respectively) were shown side-by-side with the AC 20-128A baseline debris type results for the GT case study in Table 4. As evidenced by the C_A values observed, the MULTI debris type is definitely more taxing on the GT aircraft than any of the three individual categories are alone, but the MULTI2 debris type appears to be on par with the MFRAG category.

To further help an analyst pinpoint system- and component-level contributors, UEDDAM has been developed with a feature that captures the specific iterations that meet a user-defined criterion. For example, in the GT case study, the next step of analysis that might be undertaken is to determine which release points (and associated iterations) result in high probabilities of catastrophic hazard, so that the specific release trajectories can be investigated and failures better understood. Additional information regarding significant iteration capture and usage can be found in sections 5.3.1.10, 7.7, and 9.4 of Appendix A.

Finally, another type of analysis that can be performed with the output obtained from a UEDDAM assessment is an examination of critical component translational and fore/aft risk

angles, as defined in the AC 20-128A. The UEDDAM Visualizer has a built-in utility to display these angles graphically for ease of review and presentation. Figure 17 illustrates a sample of a translational risk angle plot from the GT case study (DISK debris having been released from the number one engine fan). This graphic highlights that the majority of the critical hydraulic system components can only be impacted by an uncontained fan disk release over a very narrow range of trajectories, nominally between 330° and 10°. Section 3.2 of Appendix C provides additional information regarding how these plots can be obtained.

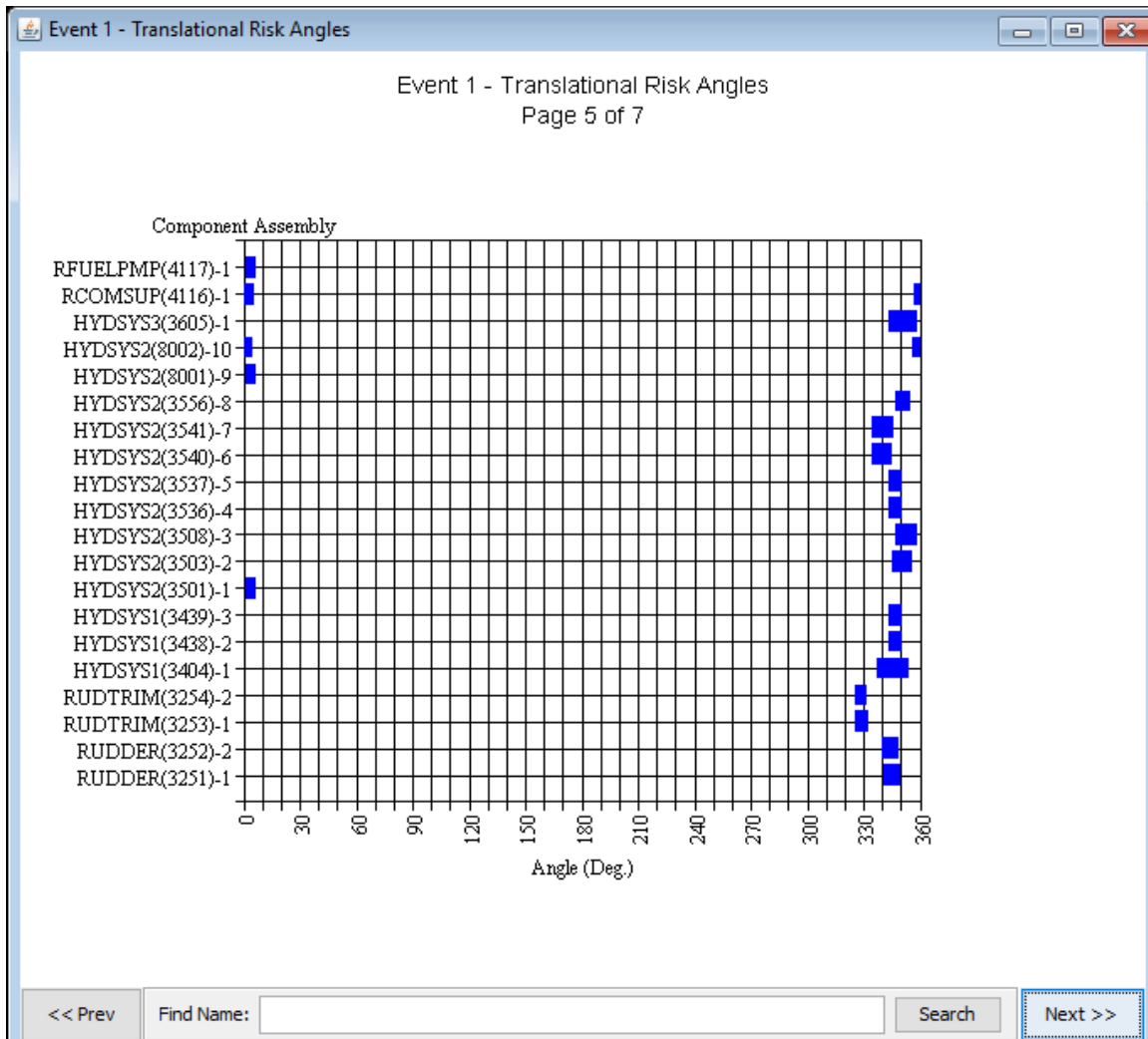


Figure 17. GT case study translational risk angle plot (DISK debris category)

3.9 Design optimization

The final step in the UEDDAM analysis process is optimization. As touched on throughout this discussion, the ultimate goal of performing an uncontained rotor debris assessment is to minimize/demonstrate minimization of the risk in such an occurrence. The only way to truly minimize the risk is to iterate the design of the aircraft through a series of refinements. For the GT case study, one type of design optimization that might be considered is one that works to enhance the structural integrity of the longerons, such that even more need to be damaged before a catastrophic loss is realized. Another option might be to consider the addition of an electrical backup to the hydraulically actuated flight control systems thereby incorporating an additional redundancy. One final example would be to incorporate additional material into the engine nacelles themselves, which would act to shield flight critical components from the effects of debris. Assessing an improved design, such as any one of these, using the UEDDAM analysis process would require circling back to step 3 (if the modification is geometric) or step 4 to update the appropriate inputs and then repeating steps 5–8.

Consider the incorporation of additional material to each GT engine nacelle, for example. In the analysis documented in DOT/FAA/AR-04/16 (Seng, Manion, & Frankenberger, 2004), the nacelle thickness was increased from a nominal 0.04 in. to 1.1 in. of aluminum in three increments (0.12 in., 0.48 in., and 1.1 in.). Analysis results from these trade studies indicated that the GT could meet a 1-in-20 requirement to a more damaging debris field by increasing the aluminum thickness to a value somewhere between 0.12 in. and 0.48 in. Performing a similar analysis on the current model results in a parallel trend. For this design optimization, rather than altering the geometry, the JTYPE FILE was modified to virtually increase the thickness of the engine nacelle components by changing their thickness descriptor from a density factory to a normal thickness. With that simple change made, steps 5-8 were repeated twice for the MULTI debris type; the first execution considered an increase in GT engine nacelle thickness to 0.15 in. and the second to 0.25 in. In both instances, a decrease of mean aircraft risk was observed (18% and 39% decrease for the 0.15-in. and 0.25-in. thick skins, respectively) (see Figure 18).

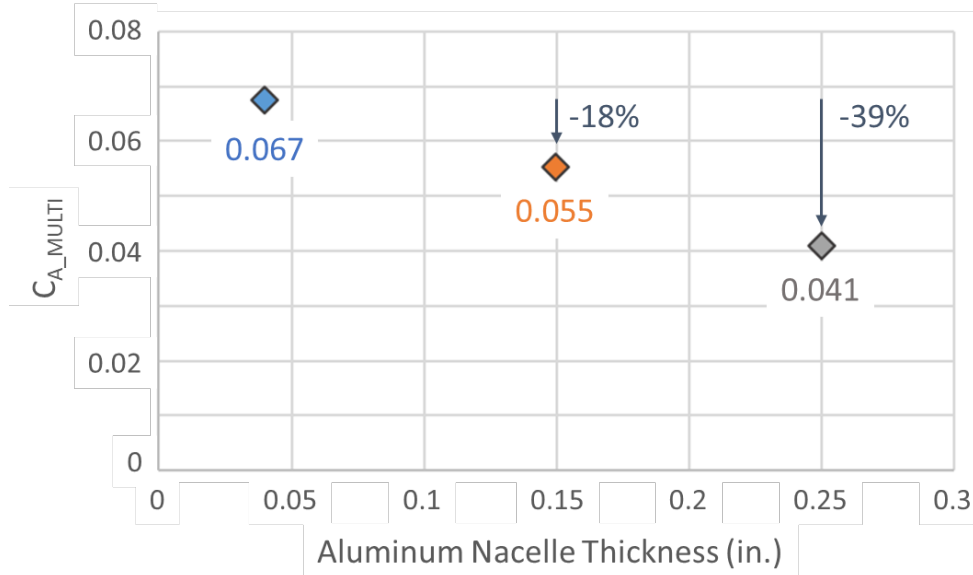


Figure 18. GT mean aircraft risk (MULTI debris category) as a function of engine nacelle thickness

While a simple example, the aforementioned assessment illustrates the UEDDAM analysis process' ability to concretely and quantitatively demonstrate minimization of risk. Upon further analysis of results, additional recommendations can be made, and the steps repeated until the design has truly been optimized to minimize the risk associated with an uncontained engine failure.

4 Excursions

Whereas the prior section of this document focused on the use of UEDDAM to perform a safety and risk analysis of an aircraft as defined in the AC 20-128A, the remainder will explore additional uses of the model within the overarching domain of aircraft safety.

4.1 Penetration adjustments

The penetration equations embedded within UEDDAM are based on those used by the military and civilian vulnerability communities, the goal being to rapidly estimate the residual velocity of various engine blade and disk debris into aircraft structures. There are two equations. The first, referred to as the V_{50} equation, is used to determine whether or not the debris penetrates a given material, where V_{50} is the limit velocity and is defined as the velocity at which a fragment has zero energy after penetrating a target. Given penetration, the second equation determines how much residual velocity (V_r) the debris carries forward.

The basis for the development of the baseline V_{50} equation is the assumption that the normal component of the impact velocity is the only contributor to the kinetic energy required for penetration of a material. In light of more recent studies into the penetration of launch and reentry debris by the FAA (Wilde & Draper, 2010), however, this assumption has been shown to be non-conservative at high obliquities, allowing for predictions of no penetration under conditions for which the data indicate penetration. Thus, the V_{50} equation has been amended since initial development of UEDDAM to allow the user more flexibility to better correlate results with existing test data. One such modification is the introduction of an obliquity adjustment factor, gamma ($0 \leq \gamma \leq 1$), which indicates that more than just the normal component of the impact velocity contributes to the kinetic energy required to penetrate the impacted material. The adjustment factor is introduced to the denominator of the V_{50} equation such that a value of $\gamma=1$ returns the updated V_{50} equation to its original form and a value of $\gamma=0$ treats every impact as perpendicular. Because matching penetration test data from different obliquity angles could potentially require different values of gamma, the user is also permitted the flexibility of entering a value of gamma for each of five obliquity angle bins (i.e., $0^\circ-30^\circ$, $30^\circ-45^\circ$, $45^\circ-60^\circ$, $60^\circ-75^\circ$, and $75^\circ-90^\circ$).

As an excursion to the baseline GT analysis described in Section 3, the same analysis was repeated keeping all inputs the same but modifying the value of gamma. In the case of launch and reentry debris, analysis indicates that the optimized gamma value of 0.707 (Wilde & Draper, 2010) is the least conservative value required to ensure the modified V_{50} equation remains conservative when compared to a limited data set. With the aforementioned research as a data point, gamma was set equal to 0.707 across all obliquity angle bins for this excursion. Upon execution and analysis of the excursion results, the mean aircraft risk (C_A) across the various debris types increased by between 4 and 18%, with the largest nominal increase of 0.005 observed for the MULTI debris type (see Figure 19). Because this excursion decreased the value of gamma assumed in the GT case study from 1.0 to 0.707, an across-the-board increase in mean aircraft risk was anticipated. However, the magnitude of the increase was unknown and it appears to have had a greater impact on the penetration capability of a disk segment over that of small and/or large fragments.

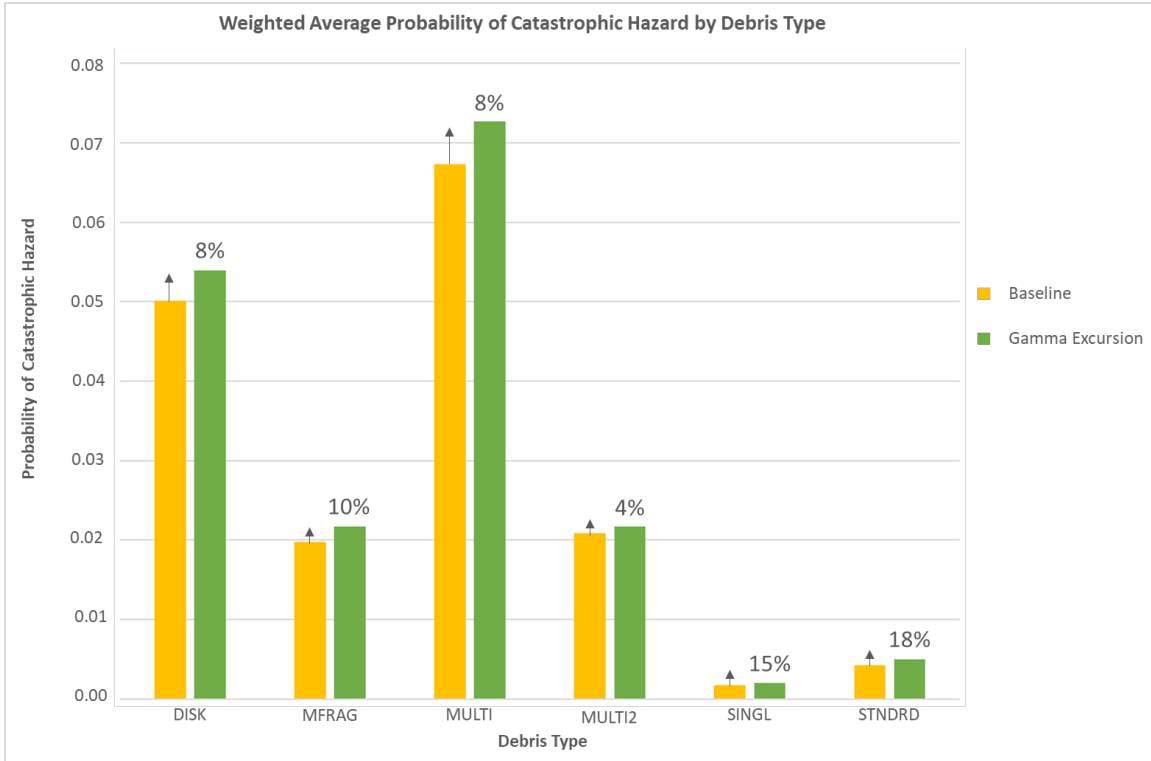


Figure 19. Increase in mean aircraft risk across debris types for $\gamma=0.707$

Note that as with any deviation from the standard AC 20-128A process, sufficient justification and evidence must accompany any UEDDAM certification analysis results using a modified version of the V_{50} equation. Additional information regarding the UEDDAM penetration equations can be found in sections 3.7 and 5.3.5 of Appendix A.

4.2 Hole size calculations

Given an uncontained engine failure, another metric UEDDAM generates that will likely be of interest to users is the cumulative hole size made in a group of components by a particular debris release³. Currently, it is envisioned that this metric could be used in support of decompression analyses (in accordance with 14 CFR 25.841) as well as Extended-range Twin-engine Operational Performance Standards (ETOPS) assessments. Because UEDDAM determines both the locations and associated areas of the holes formed by impacting rotor debris, a manufacturer can use this information to anticipate the potential for a decompression scenario and develop mitigation tactics (either structural or through modification of the flight envelope) as required for occupant safety. Additionally, this same damage information could be used by a manufacturer to

³ Additional information on hole size calculations in UEDDAM can be found in sections 3.8, 5.3.6, and 7.6 of Appendix A.

ensure extended operations diversion is available after an uncontained event. For example, an uncontained rotor burst assessment might point to the need for the installation of one or more fuel isolation systems if it is determined that holes of sufficient area to completely drain a wing tank of all fuel are highly probable in this occurrence. To meet the intent of the ETOPS regulations, a manufacturer would need to provide assurances that the remaining fuel (given isolation systems are in place) is sufficient to allow the aircraft to continue to operate (albeit at a reduced capacity) and land at an adequate airport.

To illustrate use of the hole size metrics obtained from a UEDDAM analysis, an excursion to the baseline GT analysis described in Section 3 was performed, keeping all inputs the same except for the inclusion of the optional holesize input file. Contained within the file is a listing of 17 skin and structural components that make up the pressure vessel of the GT, to include the forward and aft pressure bulkheads. The optional input file and associated components (highlighted in purple) are illustrated in Figure 20.

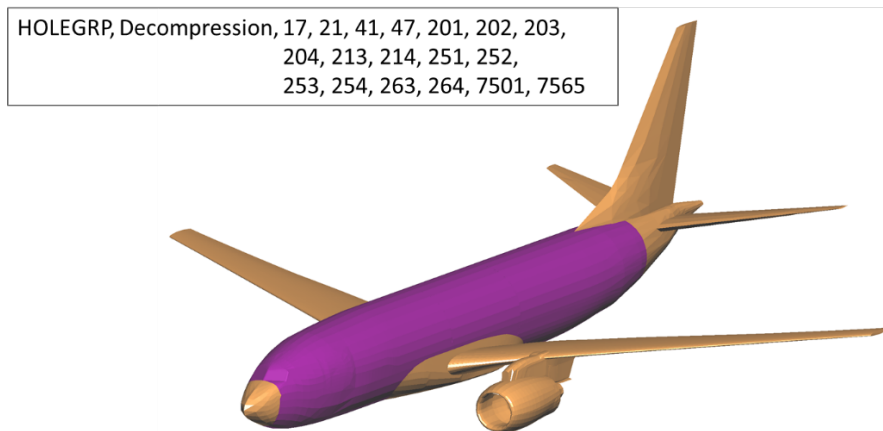


Figure 20. Holesize input file for decompression analysis of the GT

Upon execution of UEDDAM and analysis of the outputs generated for each debris type, it can be observed that the STNDRD debris type resulted in the smallest cumulative hole sizes in the pressure vessel, with maximum cumulative hole areas nearing 25 in² and average cumulative hole areas staying below 5 in². At the opposite end of the spectrum, the MULTI debris type resulted in the largest cumulative hole sizes, with maximum cumulative hole areas reaching 4300 in² (29.9 ft²). The graphic depicted in Figure 21 illustrates the maximum and average cumulative hole areas per engine stage and rotational release angle for the STNDRD and MULTI debris types.

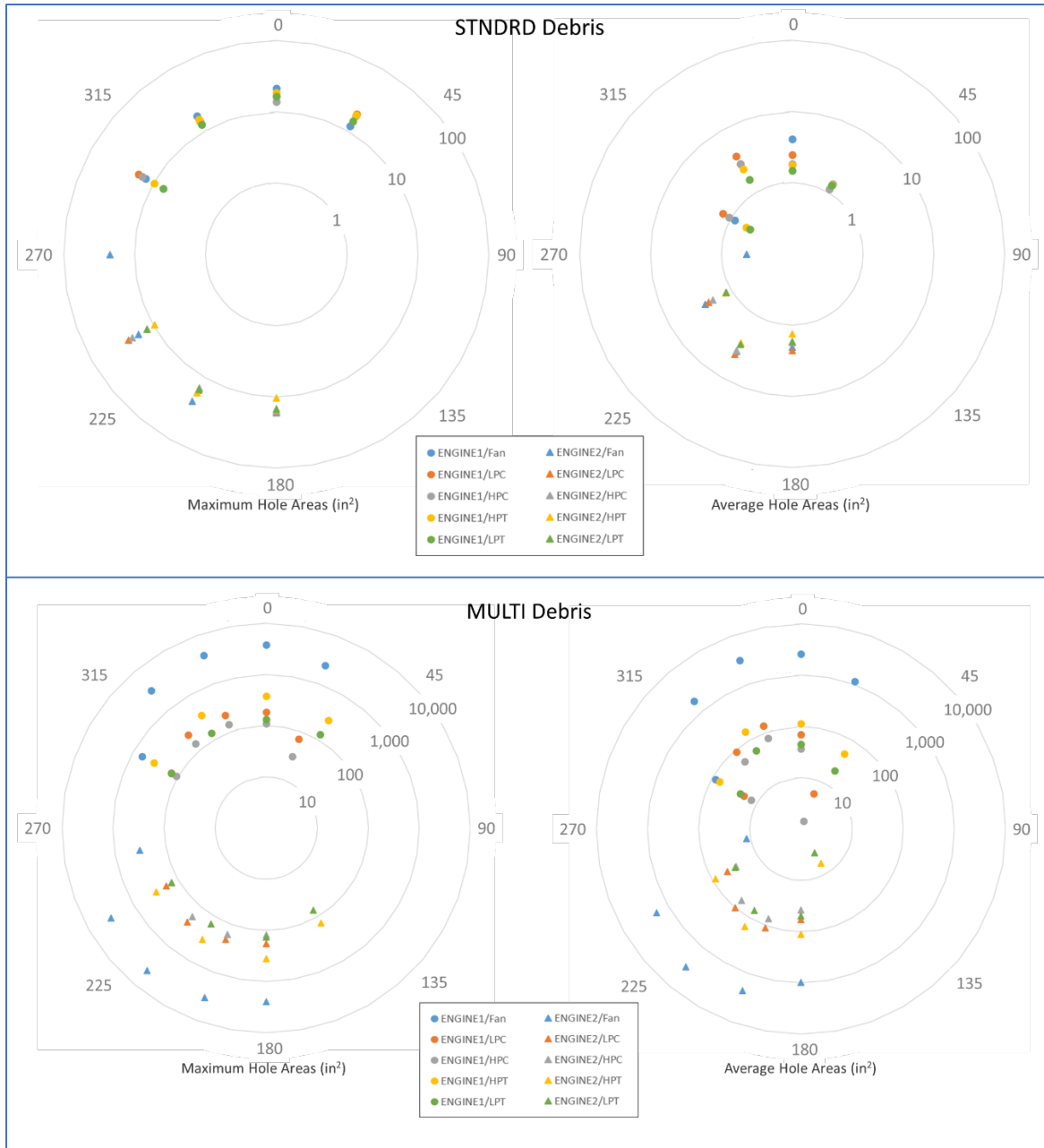


Figure 21. Maximum and average decompression cumulative hole size data

To fully characterize the cumulative hole size data, UEDDAM also outputs the average number of fragment impacts on and penetrations to the components of interest. In the case of the pressure vessel of the GT, this data was examined to determine the average number of fragment intersections contributing to the large cumulative hole areas. Examining the MULTI debris penetration data indicates that the largest cumulative hole sizes (from the Engine 1 and Engine 2 Fan sections) are realized due to approximately 10–40 individual debris impacts across the pressure vessel, depending on release angle. The image in Figure 22 illustrates a specific

iteration (no. 4) of the 220° release of debris from the Engine 2 Fan section. In this iteration, a cumulative hole size of 3874.12 in² is calculated from 40 individual penetrations of 14 large fragments (pale blue bundles), 7 small fragments (dark blue lines), and 1 one-third disk segment (yellow bundle) across the pressure vessel. With a better understanding of both the type and number of impacts leading to a decompression event, different mitigation techniques can be investigated.

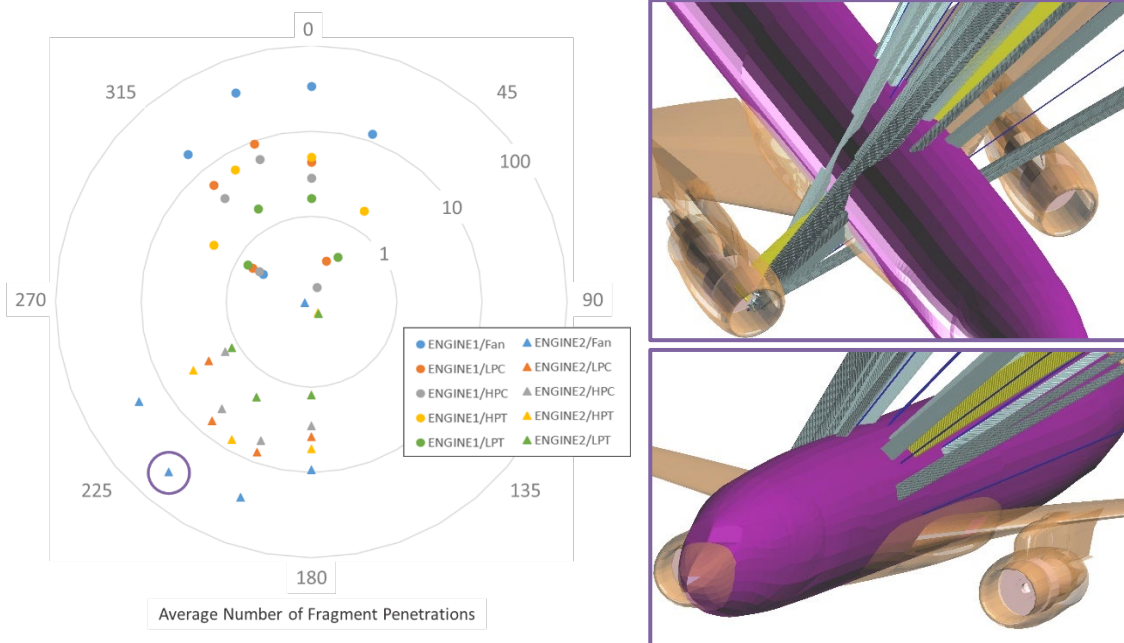


Figure 22. Illustration of number of fragment penetrations

4.3 Small component contributions

As discussed in Section 3.5, because of the way debris is modeled in UEDDAM, it is possible for components that should be impacted by an event to be missed entirely due to their size relative to the debris bundle element spacing. In this case, the analyst can add a run-time option to the Control File that instructs UEDDAM to generate one or more additional debris bundle elements to ensure all components that should be impacted by a debris event actually are (this process is referred to as hypersampling). While the baseline GT case study did not implement this run-time control option (it was deemed unnecessary due to the debris bundle element spacing used), an excursion was undertaken to determine the effects of implementing the hypersampling option on both run times and results.

Implementation of the hypersampling run control option is done through three UEDDAM control file keywords (one for each fragment type) that specify the subgrid that should be used to generate additional debris bundle elements. For this excursion, the baseline GT analysis described in Section 3 was repeated keeping all inputs the same except for the inclusion of the hypersampling run control option. All debris types were modified similarly, such that the subgrid selected for each was one-third the size of the original debris grid (e.g., the baseline disk segment released from the fan section of each engine had a specified grid size of 0.75 in.; thus, the hypersampling run control option indicated the use of a 0.25-in. subgrid for the analysis). The first observation from this excursion was the large increase in computational run times, most notably for the MULTI debris type, which saw a 16+-fold increase over that of the baseline analysis (Figure 23). While the increase in run times was anticipated (although, perhaps not this great a degree), the more important observations can be garnered from the numerical results themselves.

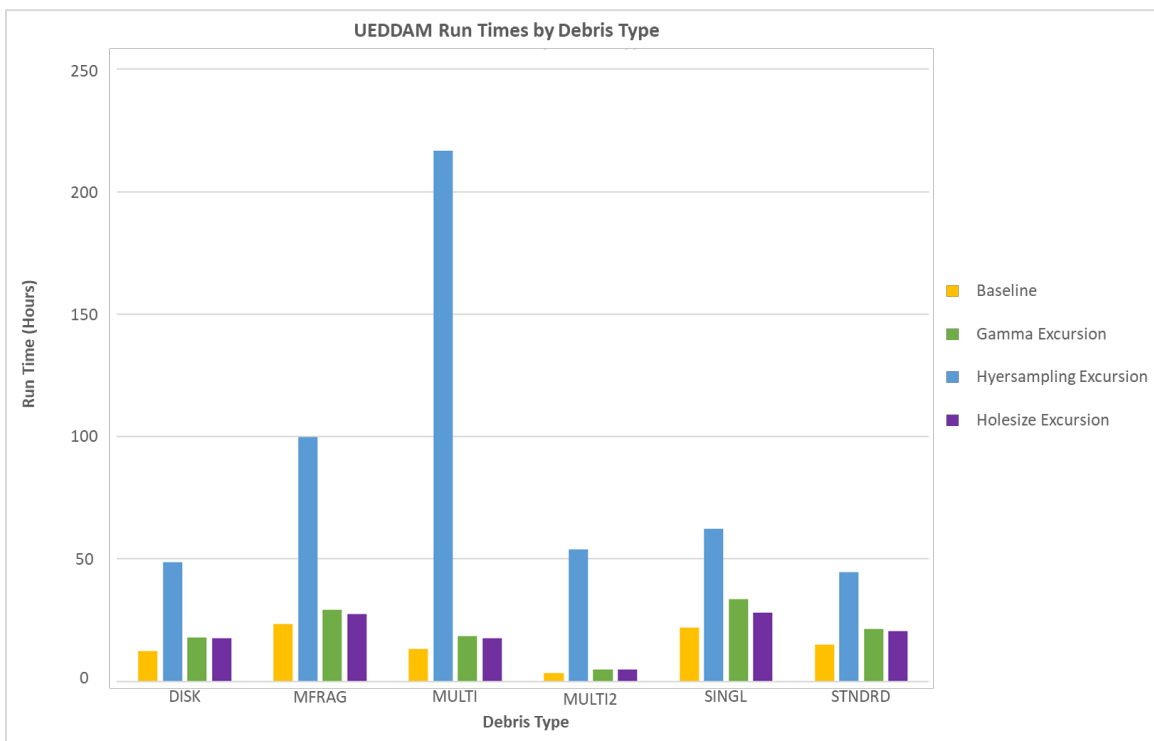


Figure 23. Computational run time comparisons

Upon execution and analysis of the excursion results, the mean aircraft risk (C_A) across the various debris types increased by between 4 and 39%, with the largest nominal increase of 0.008 observed for the MULTI debris type (Figure 24).

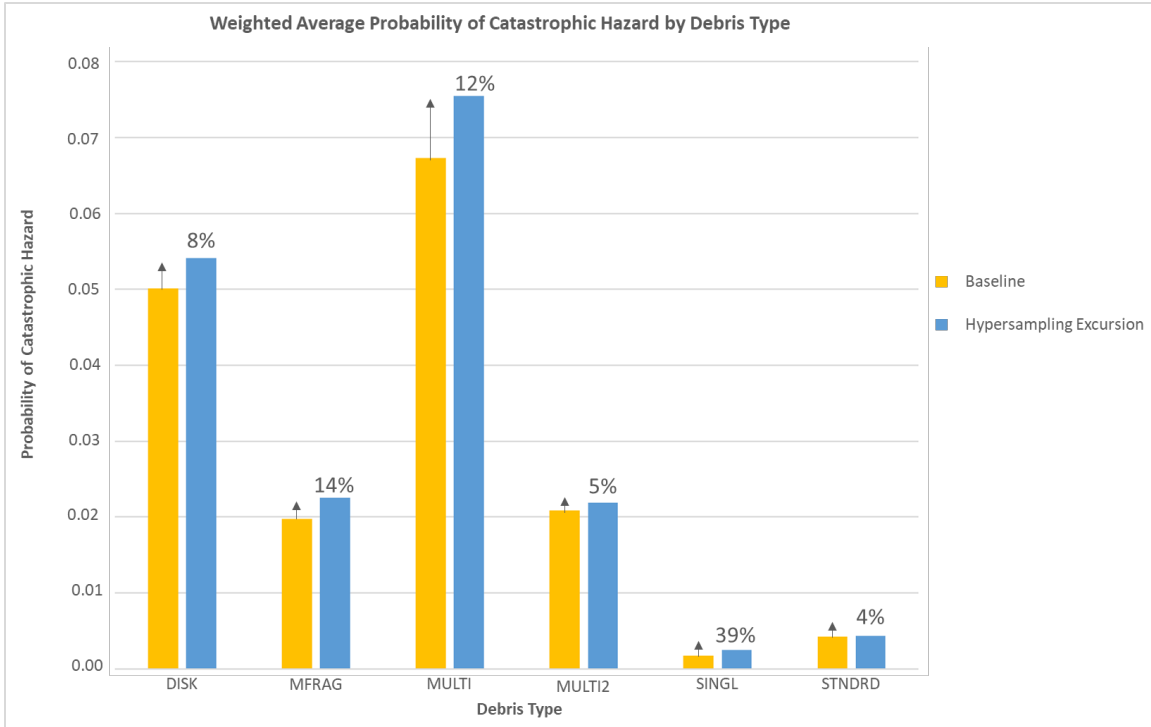


Figure 24. Increase in mean aircraft risk across debris types using hypersampling run control option

As an example of a scenario in which components could potentially be missed by a debris bundle, Figure 25 illustrates several small control cables and a frame in the path of a disk segment released from the fan section of the port engine. Due to the modeling and positioning of the components in relation to the debris bundle spacing (in addition to the fact that each ray in the bundle is treated as infinitely thin), it is easy to see how even a relatively small debris grid size could result in a potentially misinformed evaluation.

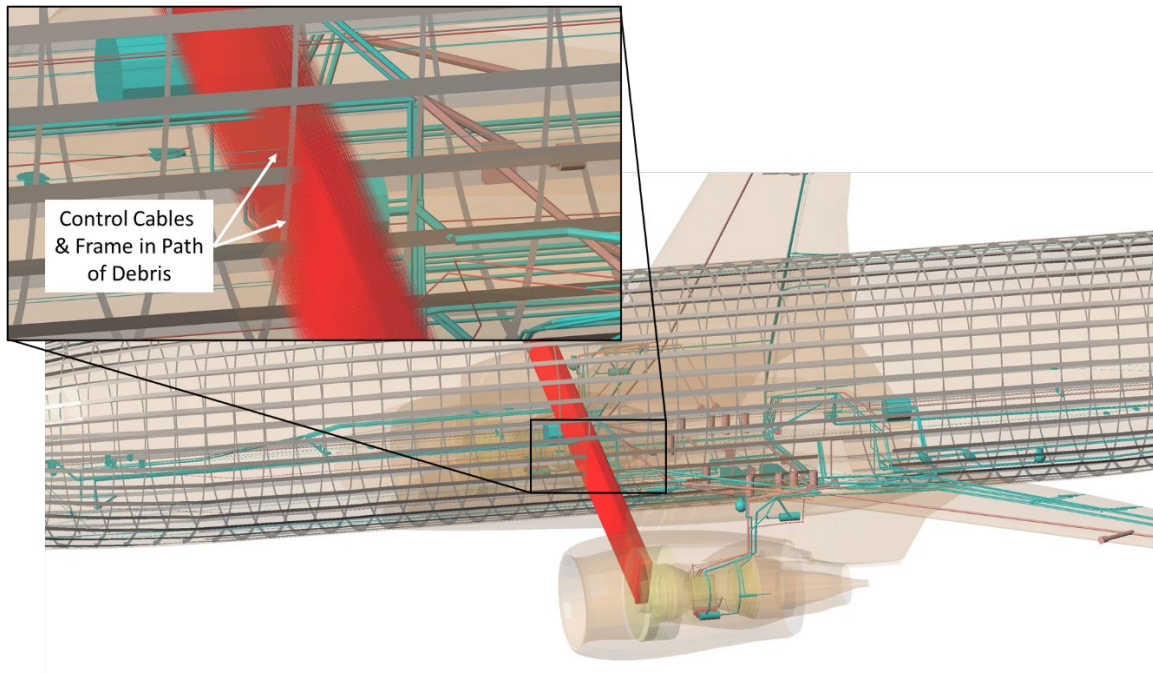


Figure 25. Example of potentially missed components

Further investigations into the specific system failures contributing to the C_A for the DISK, MFRAG, MULTI, and MULTI2 debris types show the largest nominal increases in flight controls (lateral, rudder, and elevator), hydraulics, structural members (longerons and frames), and engines (namely, fuel feed lines), all of which consist of components that are smaller in one or more dimension when looked at from the view of the rotor burst (see Figure 26). By incorporating the hypersampling run control option, for example, both the DISK and MULTI analysis results point to an increased importance of the lateral flight control system in terms of a focal point for potential design changes.

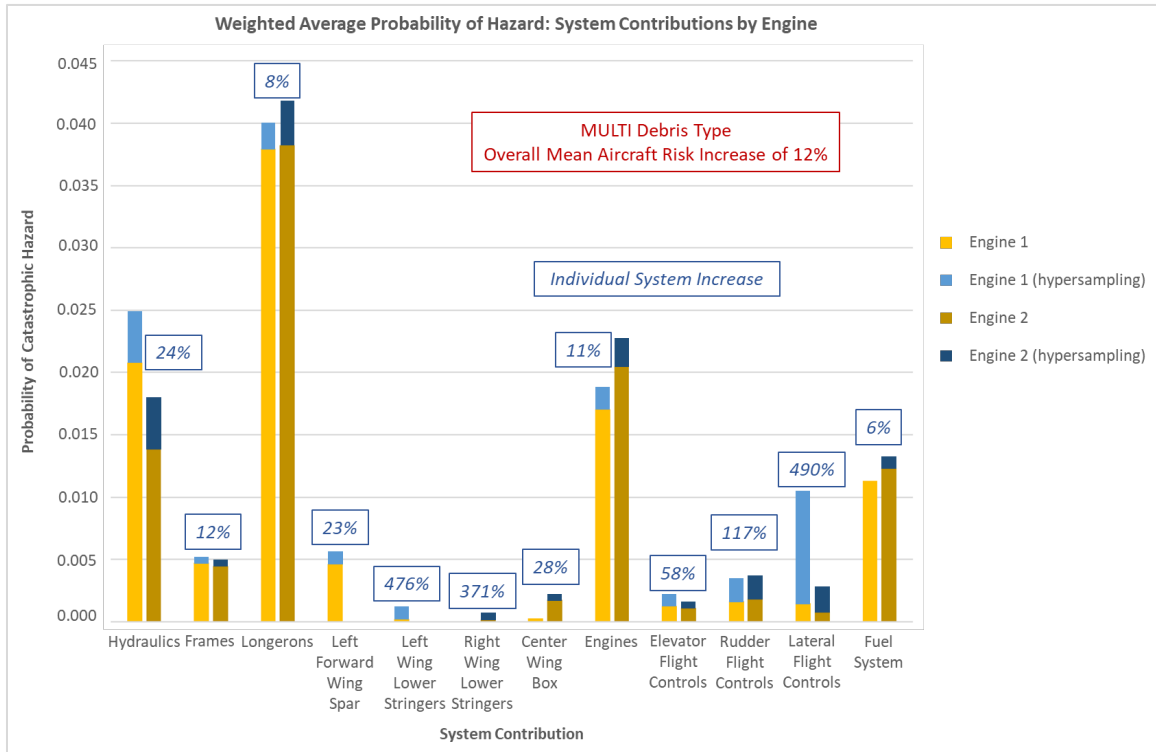
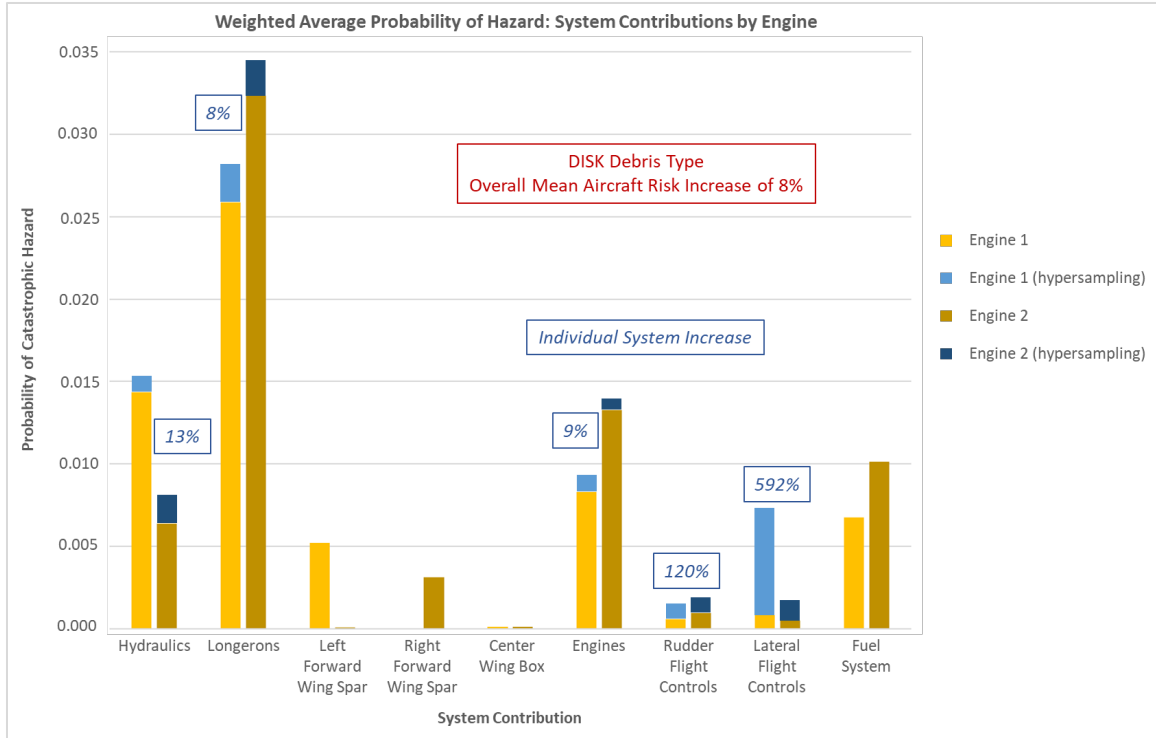


Figure 26. System contributions to DISK and MULTI debris probability of catastrophic hazard, with and without hypersampling run control option

4.4 Debris Characterizations

As a final excursion to the baseline GT analysis, we revisit the debris characterization data. Recall that for the GT case study, a generic multi-debris model was developed based on historical data collected from actual uncontained events and populated in a database. Because the DFM considers historical data, the model evolves as additional data are incorporated. Thus, if we take a snapshot of the DFM prior to 2020, the characterizations appear different. When a similar generic analysis was conducted in the early 2000's (Seng, Manion, & Frankenberger, 2004), for example, the DISK, MFRAG, MULTI, and SINGL characterizations were different from those used in the current GT Baseline case study and the STNDRD and MULTI2 debris types were not studied. One difference is the change in variance angles considered (2.5° or 10°, depending on debris type, versus 20° or 35°, depending on engine stage). This change in variance results in a change in number of release angles as well as a change in number of iterations required for a stable result (see Table 5).

Table 5. Debris types considered in DOT/FAA/AR-04/16

Debris Type	Description	Fan	Compressor	Turbine	Iterations
DISK	Single 1/3 rd disk segment	2.5° variance, 0.5-in. grid	2.5° variance, 0.5-in. grid	2.5° variance, 0.5-in. grid	30
MFRAG	Single large fragment	2.5° variance, 0.5-in. grid	2.5° variance, 0.5-in. grid	2.5° variance, 0.5-in. grid	30
MULTI	Single 1/3 rd disk segment with multiple small and/or large fragments	Disk: 2.5° variance, 0.85-in. grid; Fragments: 10° variance, 0.5-in. grid	Disk: 2.5° variance, 0.5-in. grid; Fragments: 10° variance, 0.5-in. grid	Disk: 2.5° variance, 0.5-in. grid; Fragments: 10° variance, 0.5-in. grid	30
SINGL	Single small fragment	10° variance	10° variance	10° variance	100

Another difference is observed in the characterizations themselves. The masses, velocities, and dimensions of individual fragments are not consistent between the two sets of debris. As an example, consider the MFRAG debris type. For the baseline GT analysis, the fan fragment weighs 3.75 lbs., travels at a velocity of 894 fps, and has length, width, and thickness dimensions of 9.6 in., 8 in., and 0.3 in., respectively. In the early 2000's assessment, however, the fan fragment weighed only 1.8 lbs., traveled at a velocity of 813 fps, and had length, width, and thickness dimensions of 8 in., 8 in., and 0.2 in, respectively. MFRAG debris type characterization data comparisons across all engine sections are provided in Table 6. The data in

the table indicates that for all engine sections except for the fan, the mass of the considered debris for this debris type decreased from that assessed in the early 2000's.

Table 6. MFRAG characterization data comparison

Engine Section	DFM	Mass (lbs.)	Velocity (fps)	Length (in.)	Width (in.)	Thickness (in.)
Fan	Excursion	1.8	813	8	8	0.2
	GT Baseline	3.75	894	9.6	8	0.3
Low Pressure Compressor	Excursion	6	523	13.5	7	1
	GT Baseline	4.4	523	16	4	1
High Pressure Compressor	Excursion	6	523	13.5	7	1
	GT Baseline	4.4	523	16	4	1
Low Pressure Turbine	Excursion	11.3	505	18.3	1.7	1.3
	GT Baseline	3.8	505	6	4	1
High Pressure Turbine	Excursion	10	967	22	1.5	1.5
	GT Baseline	8.3	967	13.5	8	2

For this final excursion, the baseline GT analysis was reassessed keeping all inputs the same except for the debris characterization data, number of release angles, and number of iterations. The change in mean aircraft risk for the four debris types under consideration is illustrated in Figure 27, with the greatest percent change observed for the MFRAG debris type (an increase of 17%) and the greatest nominal change observed for the MULTI debris type (a decrease of 0.0045 in probability of catastrophic hazard). Note that the results are neither consistently lower nor higher than those of the baseline GT analysis, as the effects of debris modification are not as straightforward as the other excursions discussed in the previous three sections. For example, referring back to Table 6, the fragment emanating from the Fan section in the baseline GT analysis has a higher mass than that in the excursion, but the opposite is true for all other engine sections.

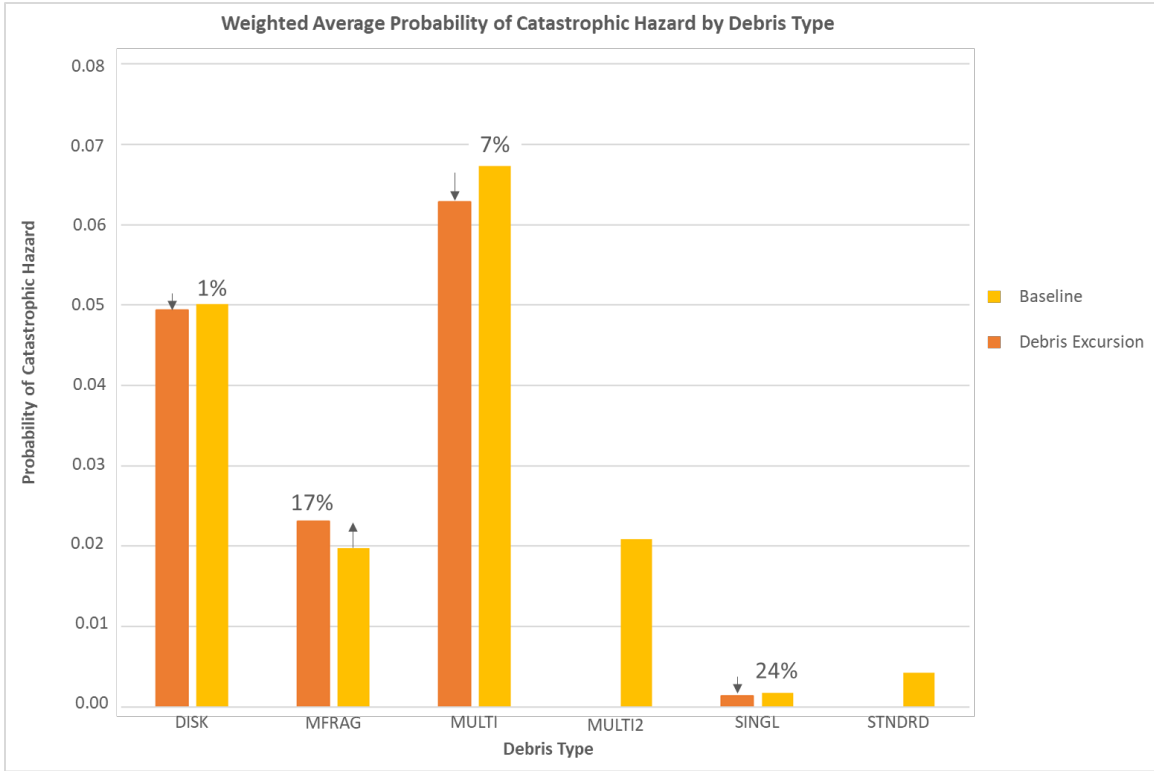


Figure 27. Change in mean aircraft risk across debris types using different debris characterization parameters

A further examination of the MFRAG results indicates the system contributions that are driving the 17% increase. Significant increases are observed in structure (specifically, longerons) and hydraulics components and are mainly due to debris emanating from the high- and low-pressure turbine stages (see Figure 28).

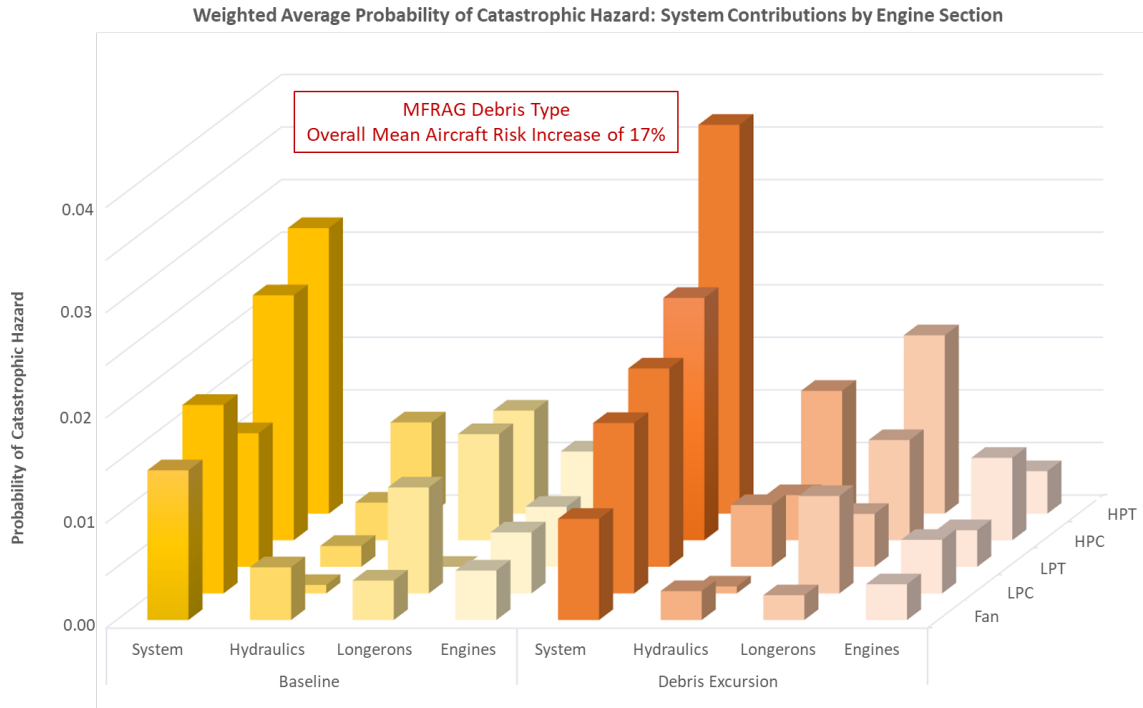


Figure 28. MFRAG system contribution comparison using different debris characterization parameters

5 Conclusion

Uncontained rotor failures are a risk that must be addressed by airframe manufacturers worldwide. While manual methods to assess this risk exist, the analysis process and automation afforded by UEDDAM reduces errors, permits standardization, allows for ease of trade studies, and introduces a manageable means of evaluating uncontained engine events stochastically.

UEDDAM provides a standardized approach to conduct single or multi fragment rotor burst hazard analysis. Multi fragment debris models have been developed to characterize in-service aircraft rotor burst events. The UEDDAM process provides a better analytical representation of these in-service events and a better representation of the true aircraft hazard.

UEDDAM as a design tool can provide early insight to the rotor burst hazard for a given aircraft configuration. Additionally, trade studies can be performed to conduct cost/benefit analyses of component relocation or shielding showing minimization of the rotor burst hazard.

As a certification tool, UEDDAM provides a standardized approach to conducting rotor burst safety and risk assessments as defined in the AC 20-128A. UEDDAM output provides insight to the rotor burst hazard and can be used to develop a top-level 1-in-20 analysis to address compliance to CFR in accordance with the recommended methodology defined in AC20-128A.

As evidenced by the GT case study, UEDDAM outputs data that allows determination of what engine sections are the major contributors to damage and, in addition, allows determination of what systems and components are responsible for the hazard.

6 References

- Frankenberger, C. E. (1999). *Large Engine Uncontained Debris Analysis*. Final Report, U.S. Department of Transportation, Federal Aviation Administration. Retrieved from <http://www.tc.faa.gov>
- Frankenberger, C. E. (1999). *Small Engine Uncontained Debris Analysis*. Final Report, U.S. Department of Transportation, Federal Aviation Administration. Retrieved from <http://www.tc.faa.gov>
- Frankenberger, C. E. (2019). *Large Engine Uncontained Debris Analysis - High-Bypass Ratio Engine Update*. Final Report, U.S. Department of Transportation, Federal Aviation Administration. Retrieved from <http://www.tc.faa.gov>
- Lundin, S. J. (2001). *Engine Debris Fuselage Penetration Testing, Phase I*. Final Report, U.S. Department of Transportation, Federal Aviation Administration. Retrieved from <http://www.tc.faa.gov>
- Lundin, S. J. (2002). *Engine Debris Fuselage Penetration Testing, Phase II*. Final Report, U.S. Department of Transportation, Federal Aviation Administration. Retrieved from <http://www.tc.faa.gov>
- Lundin, S. J., & Mueller, R. B. (2005). *Advanced Aircraft Materials, Engine Debris Penetration Testing*. Final Report, U.S. Department of Transportation, Federal Aviation Administration. Retrieved from <http://www.tc.faa.gov>
- Manchor, J., & Frankenberger, C. (1999). *Engine Debris Penetration Testing*. Final Report, U.S. Department of Transportation, Federal Aviation Administration. Retrieved from <http://www.tc.faa.gov>
- Seng, S., Manion, J., & Frankenberger, C. (2004). *Uncontained Engine Debris Analysis Using the Uncontained Engine Debris Damage Assessment Model*. Final Report, U.S. Department of Transportation, Federal Aviation Administration. Retrieved from <http://www.tc.faa.gov>

U.S. Department of Transportation Federal Aviation Administration. (2017). *Order 8110.4C Chg 6 Type Certification*. Washington D.C.: Retrieved from www.faa.gov/regulations_policies/orders_notices.

Wilde, P. D., & Draper, C. (2010). Aircraft Protection Standards and Implementation Guidelines for Range Safety. *48th Annual AIAA Aerospace Sciences Meeting* (pp. AIAA 2010-1542). Orlando: AIAA.

A UEDDAM version 6.0 user and analyst manual



UEDDAM User
Manual

Uncontained Engine Debris Damage Assessment Model (UEDDAM) Version 6.0 User and Analyst Manual

Prepared by:

**Ellen M. Phifer
Doug R. Howard**

September 2020

Further dissemination only as directed by NAWCWD, Code 418300D, China Lake, CA 93555, September 2020 or higher DOD authority.

INTENTIONALLY LEFT BLANK

ABSTRACT

The Uncontained Engine Debris Damage Assessment Model (UEDDAM) performs disk burst assessments using a modified, unlimited distribution of COVART 6.9 (FAA). COVART (distributed through the Defense Systems Information Analysis Center [DSIAC]) has long been used in the evaluation of aircraft vulnerability to kinetic energy threats (fragments and projectiles). For engine debris containment studies, COVART is used to generate line-of-sight data (representing the trajectories of uncontained engine debris originating from within vehicle engines through aircraft components) and estimate fragment penetration characteristics, component damage, and resultant hazard level evaluations for each trajectory. UEDDAM provides a reduced user workload by automatically formatting the data required to run COVART, executing it over many iterations, and accumulating/formatting the output to generate average hazard levels over multiple debris origins for the entire aircraft under evaluation.

UEDDAM v6.0 expands on the capabilities of previous versions to include the following updates: 1) enhancements to the parallel computing performance of UEDDAM on computing clusters with distributed memory architecture and network file systems and 2) incorporation of a standalone fault tree tool capable of development, analysis, and file conversion.

INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

1	INTRODUCTION	1
1.1	Background	1
1.2	Purpose	2
2	MODEL OVERVIEW	3
2.1	UEDDAM Analysis Process	3
2.2	General Description	5
2.3	UEDDAM Metrics	5
3	UEDDAM FEATURES AND METHODOLOGY	8
3.1	UEDDAM Program Flow	8
3.1.1	Load UEDDAM Control Data	8
3.1.2	Remove Engine Near-Field Components from Target Geometry File	9
3.1.3	Generate Engine/Event-Dependent COVART Support Files	9
3.1.4	Determine Translational and Spread Intercept Angles	10
3.1.5	Perform Geometric Reduction	10
3.1.6	Generate CBULK and MASTER Files and Run COVART6	10
3.1.7	Perform Monte-Carlo Analysis	11
3.1.8	Intermediate-Process Hazard Level Results	11
3.1.9	Repeat Process for Each Nominal Release Point of a Disk Stage	12
3.1.10	Final-Process Hazard Level Results	12
3.1.11	Repeat Process for Each Disk Event of an Engine	12
3.1.12	Repeat Process for Each Engine	12
3.2	Debris Type Analysis Methods	17
3.2.1	Disk Segment Debris Type	18
3.2.2	Large Fragment Debris Type	18
3.2.3	Small Fragment Debris Type	19
3.3	Near-Field Feature	19
3.4	Geometry Reduction and Fault Tree Interrogator	19
3.5	Monte-Carlo Debris Characteristics	21
3.6	Auto-Fail Feature	22
3.7	Penetration Equations	22
3.8	Hole Size Calculations	25
4	ANALYSIS OBJECTIVE AND SYSTEM FLIGHT PHASE CRITICALITY ANALYSIS	26
4.1	Establish Objective	26
4.2	Perform Criticality Analysis	27
4.2.1	Damage Modes and Effects Analysis (DMEA)	27
4.2.2	Hazard Trees	29
5	AIRCRAFT GEOMETRY DEFINITION AND UEDDAM INPUT GENERATION	31
5.1	Geometric Description (TARGET FILE)	31
5.1.1	FASTGEN Overview	32
5.1.2	Conversion to FASTGEN from Other Geometry Formats	35
5.1.3	Guidelines/Techniques and Code Limitations (Lessons Learned)	39
5.2	Describe Component and Component Redundancies	40
5.2.1	Component Properties File (JTYPE FILE)	40

5.2.2	Probability of Hazard Given Hit File (PK FILE)	42
5.2.3	Multiply Vulnerable File (MV FILE).....	43
5.2.4	Guidelines/Techniques and Code Limitations.....	57
5.3	Define Analysis Parameters	58
5.3.1	Control File.....	58
5.3.2	Debris Characterization File (DEBRIS FILE)	68
5.3.3	Near-Field Components File (NEARFIELD FILE).....	79
5.3.4	Auto-Fail Components File (AUTOFAIL FILE)	80
5.3.5	Material Properties File (MATERIAL FILE)	81
5.3.6	Hole Size Group File (HOLEGRP FILE)	82
5.3.7	Guidelines/Techniques and Code Limitations.....	83
6	INPUT FILE VERIFICATION AND UEDDAM EXECUTION	84
6.1	Operating System Requirements.....	84
6.2	Running UEDDAM	85
6.3	Troubleshooting	86
6.4	Production Runs.....	87
6.4.1	Input Data Tables	88
6.4.2	Run Setup Panel	89
7	UEDDAM OUTPUT	91
7.1	Event Level Summary File (SUMMARY FILE).....	91
7.2	Component Level Summary File (HITSUM FILE).....	92
7.3	Detailed Hazard Information File (DETAIL FILE).....	93
7.4	Translational Risk Angle File (INTERCEPT FILE)	94
7.5	TRAJECTORY FILE.....	95
7.6	Hole Size File (HOLESIZE FILE)	95
7.7	Significant Iteration Summary Files	96
7.7.1	Summary File (SIGITSYS FILE).....	97
7.7.2	Critical Component Hit Summary File (SIGITCRIT FILE)	97
7.7.3	Cumulative Hole Area Summary File (SIGITCHOLE FILE)	98
7.7.4	Hole Size Summary File (SIGITSHOLE FILE)	99
7.8	VISUALIZER FILE.....	99
7.9	ERROR FILE.....	100
8	POST-PROCESSING.....	102
9	SAMPLE CASES	105
9.1	Sample Case One – S-FRAGMENT.....	105
9.2	Sample Case Two – L-FRAGMENT.....	108
9.3	Sample Case Three – SEGMENT.....	109
9.4	Sample Case Four – Significant Iterations.....	111
9.5	Sample Case Five – Debris Distributions	114
9.5.1	Uniform Distribution	114
9.5.2	Normal Distribution.....	115
9.5.3	Skewed Distribution	115
9.6	Sample Case Six – Generic Twin-Engine Aircraft.....	116
9.7	Sample Case Seven – Modification of Shear Constant	117
9.8	Sample Case Eight – UEDDAM Hole Size Estimation.....	118
9.9	Sample Case Nine – Logic Statement Inputs.....	119

9.10	Sample Case Ten – Hypersampling	120
9.11	Sample Case Eleven – Setup Macro and Post-Processing.....	122
9.12	Sample Case Twelve – Infinite Energy Disk	126
9.13	Sample Case Thirteen – Penetration Equations	127
9.14	Sample Case Fourteen – Multi-Segment Analyses.....	128
10	REFERENCES	130
11	GLOSSARY	132
APPENDIX A:	UEDDAM INTEGRATION WITH COVART 6.X.....	A-1
APPENDIX B:	AIRCRAFT GEOMETRY DEVELOPMENT REFERENCE: EXCERPTS FROM FASTGEN 6.1 TARGET DESCRIPTION DOCUMENT	B-1
APPENDIX C:	CRITICAL INPUT DEVELOPMENT REFERENCE: EXCERPTS FROM COVART 6.7 USER’S MANUAL	C-1
APPENDIX D:	EXCERPTS FROM THE REPORT “UNCONTAINED ENGINE DEBRIS ANALYSIS USING THE UNCONTAINED ENGINE DEBRIS DAMAGE ASSESSMENT MODEL”	D-1
APPENDIX E:	THE GENERIC ENGINE DEBRIS FRAGMENT MODEL TABLES	E-1
APPENDIX F:	DYNAMIC SHEAR CONSTANT (Cs) DEVELOPMENT	F-1

LIST OF FIGURES

Figure 2-1. UEDDAM Analysis Process.....	4
Figure 2-2. Typical Trajectory Plotting [8].....	6
Figure 2-3. Definition – Threat Window [8].	7
Figure 3-1. Generalized UEDDAM Program Flow (Page 1 of 5).	13
Figure 3-2. Debris Segment Swath Gridding.....	18
Figure 3-3. Generic Engine Near-Field Example.	19
Figure 3-4. UEDDAM Geometry Reduction Example.....	20
Figure 3-5. Generic Twin Engine Aircraft Monte-Carlo Example.....	21
Figure 3-6. Generic Engine Auto-Fail Example.	22
Figure 3-7. Description of Fragment Parameters.....	23
Figure 4-1. Current Location in the UEDDAM Analysis Process: Analysis Objective and Criticality Analysis.....	26
Figure 4-2. Hydraulic Pressure Line in a Generic Twin-Engine Aircraft.....	28
Figure 4-3. Generic Twin-Engine Aircraft Hydraulic System Hazard Tree.....	30
Figure 5-1. Current Location in the UEDDAM Analysis Process: Aircraft Geometry Definition and UEDDAM Input Generation.....	31
Figure 5-2. FASTGEN Geometric Description Structure.....	32
Figure 5-3. FASTGEN Coordinate System.	33
Figure 5-4. FASTGEN CBULK File of a Generic Hydraulic Pump.	35
Figure 5-5. Solid Tetrahedron Described in Stereolithography Format.	36
Figure 5-6. Triangles and Grid Points Used to Describe a Tetrahedron.....	37
Figure 5-7. Solid Tetrahedron Described in FASTGEN Format.	37
Figure 5-8. Sample $P_{\text{haz/h}}$ Tables and Graphs.....	43
Figure 5-9. Partial Circumferential Frames Hazard Tree.	44
Figure 5-10. Circumferential Frames in a Generic Twin-Engine Aircraft.	45
Figure 5-11. Partial Simplified Circumferential Frames Hazard Tree.	45
Figure 5-12. Complete Further Simplified Circumferential Frames Hazard Tree.....	46
Figure 5-13. Dependent vs. Independent Fault Trees.	47
Figure 5-14. Multiple Adjacent Component Utility.	48
Figure 5-15. Example MAC Tool Usage.....	49
Figure 5-16. Completion Window Generated by Successful Execution of MAC Tool.	50
Figure 5-17. MV FILE Generated by MAC Tool.....	50
Figure 5-18. Comparison of Original JTYPE FILE (Left) to that Generated by the MAC Tool (Right).	51
Figure 5-19. Generic Three-Engine Aircraft.	51
Figure 5-20. MV Entry for Weighting Factors – Engine Failure Example.	52
Figure 5-21. MV Entry for Weighting Factors – Hydraulic System Failure Example.....	53
Figure 5-22. Small Component Hypersampling (Subgrid and Saved Ray Highlighted in Red).....	61
Figure 5-23. Disk Centroid and Orientation Angles.....	66
Figure 5-24. Disk Assessment Angles and Step Increments.	67
Figure 5-25. Example of the UEDDAM Control Input File.....	68
Figure 5-26. Debris Release Radii.	70

Figure 5-27. Parallel Ray Grid for Generic Fragment Dimensions.	71
Figure 5-28. Debris Characterization – Fore/Aft Spread Angles.	73
Figure 5-29. Debris Characterization – Variance.	73
Figure 5-30. Debris Characterization – Uniform Distribution.....	74
Figure 5-31. Debris Characterization – Normal Distribution.	74
Figure 5-32. Debris Characterization – Skewed Distribution.....	75
Figure 5-33. Sample Fragment Dimensions.	76
Figure 5-34. Fragment Orientation.	76
Figure 5-35. Mass Distribution Fraction.....	78
Figure 5-36. Generic Engine Debris Model Example Debris File Entry.....	79
Figure 5-37. Example of NEARFIELD FILE.....	80
Figure 5-38. AUTOFAIL FILE Example.	81
Figure 5-39. Standard UEDDAM MATERIAL FILE.....	82
Figure 5-40. HOLEGRP FILE Example.....	82
Figure 5-41. User-Defined Identifications and Their Use in Multiple Records.	83
Figure 6-1. Current Location in the UEDDAM Analysis Process: Input File Verification and UEDDAM Execution.....	84
Figure 6-2. Required Directory Structure for Production Runs and Post-Processing.	88
Figure 7-1. Sample SUMMARY FILE.....	92
Figure 7-2. Sample HITSUM FILE.....	93
Figure 7-3. Sample DETAIL FILE.....	93
Figure 7-4. Sample INTERCEPT FILE.....	94
Figure 7-5. Sample HOLESIZE FILE.	96
Figure 7-6. Sample SIGITSYS FILE.....	97
Figure 7-7. Sample SIGITCRIT FILE.	98
Figure 7-8. Sample SIGITCHOLE FILE.....	99
Figure 8-1. Current Location in the UEDDAM Analysis Process: Post-Processing.....	102
Figure 8-2. Analysis_Tool_Input.txt Example.	103
Figure 9-1. Simple Geometry for Sample Cases One Thru Four.	106
Figure 9-2. Front View for Sample Case One.	106
Figure 9-3. Top View for Sample Case One.....	107
Figure 9-4. UEDDAM Spread and Translational Risk Angles for Sample Case One.	108
Figure 9-5. Front View for Sample Case Two.....	108
Figure 9-6. UEDDAM Spread and Translational Risk Angles for Sample Case Two.....	109
Figure 9-7. Front View for Sample Case Three.....	110
Figure 9-8. UEDDAM Spread and Translational Risk Angles for Sample Case Three.....	111
Figure 9-9. Sample Case Four – SIGITCHOLE Output.....	112
Figure 9-10. Sample Case Four – Significant Iteration Trajectories.	112
Figure 9-11. Sample Case Four – SIGITSYS Output.....	113
Figure 9-12. Sample Case Five – Uniform Distribution for Debris Characterization.	114
Figure 9-13. Sample Case Five – Normal Distribution for Debris Characterization.....	115
Figure 9-14. Sample Case Five - Skewed Distribution for Debris Characterization.....	116
Figure 9-15. Generic Twin-Engine Aircraft for Sample Case Six.....	116
Figure 9-16. Hazard Probability Summary for Sample Case Six.	117
Figure 9-17. Sample Case Seven Geometry.	117
Figure 9-18. Debris Release Angles for Sample Case Eight.	118

Figure 9-19. Sample Case Eight HOLESIZE FILE.....	119
Figure 9-20. Model Setup for Sample Case Nine.....	119
Figure 9-21. UEDDAM Approximation of Large Fragment Trajectories for Sample Case Ten.....	121
Figure 9-22. UEDDAM Small Component Fix for Sample Case Ten.....	121
Figure 9-23. Lower Half of Setup Macro Spreadsheet.....	122
Figure 9-24. Upper Half of Setup Macro Spreadsheet.....	123
Figure 9-25. Files Created by Setup Macro.....	124
Figure 9-26. Sample Case Eleven: PH_sum.txt and Combined_Debris.csv.....	125
Figure 9-27. JTYPE FILE Changes for Sample Case Twelve.....	126
Figure 9-28. Sample Case Thirteen Geometry.....	127
Figure 9-29. Sample Case Fourteen Debris.....	129

LIST OF TABLES

Table 3-1. List of Parameters Used in V_{50} Equation.....	23
Table 3-2. List of Parameters Used in Residual Velocity Equation.....	25
Table 4-1. Flight Phases Identified in AC 20-128A.....	27
Table 4-2. DMEA Entry for Generic Twin-Engine Aircraft Hydraulic Pressure Line.....	29
Table 5-1. Available FASTGEN Primitives.....	32
Table 5-2. Common Group Identification Numbers.....	34
Table 5-3. FASTGEN to NASTRAN Element Cross-Reference.....	38
Table 5-4. Predefined Material Types.....	41
Table 5-5. Engine Failure Matrix.....	52
Table 5-6. Entry of P_{HAZ} for Failure Combinations in COVART.....	53
Table 5-7. Hydraulic System Failure Matrix.....	53
Table 6-1. Sample UEDDAM Run Times.....	85
Table 7-1. UEDDAM Warnings that Can Safely Be Ignored.....	100
Table 9-1. Manual Analysis Results for Sample Case One.....	107
Table 9-2. Manual Analysis Results for Sample Case Two.....	109
Table 9-3. Manual Analysis Results for Sample Case Three.....	110
Table 9-4. Sample Case Nine Failure Combinations.....	120

PREFACE

This manual was prepared for the Naval Air Warfare Center Weapons Division under contract N68936-17-D-0028, Task Order 0007. All questions and comments regarding the content contained herein may be directed as follows:

SURVICE Engineering Company
Applied Technology Operation
ATTN: Mr. Doug Howard
3538 Aldino Road
Hangar #6
Churchville, MD 21028
410.273.7722
doug.howard@survice.com

Note: Throughout the UEDDAM User's Manual, there are terms used that may be unfamiliar to the reader. In order to provide additional insight and/or background information, some of the more important terms are defined in the glossary. These terms are formatted in ***bold, italicized font*** in this manual.

1 INTRODUCTION

1.1 BACKGROUND

An *event* resulting in the release of uncontained debris from an aircraft engine can have devastating effects on and can potentially result in a catastrophic loss of the aircraft. The Federal Aviation Administration (FAA), under the Aircraft Catastrophic Failure Prevention Program, has initiated several activities to address the uncontained engine debris problem. For one of these activities, the Naval Air Warfare Center Weapons Division (NAWC-WD) Survivability Division, Code 418000D, was tasked by the FAA to characterize the uncontained engine debris and develop an uncontained engine debris analysis tool. Accordingly, the SURVICE Engineering Company, under the direction of NAWC-WD, was tasked to develop and code this tool, named the Uncontained Engine Debris Damage Assessment Model (UEDDAM). This tool assesses the effects of uncontained engine debris on the aircraft and is useful for aircraft design, assessment, and certification. It models multiple disk segment, large fragment, and small fragment debris trajectories and their end effects on the aircraft system.

The process of the UEDDAM development was divided into three phases. Phase I emphasized the collection of data, the definition of the problems, and the selection of existing computer codes to solve the problems [1]. Under Phase I, the selected computer codes (FASTGEN [FAST hotline GENerator] and COVART [Computation Of Vulnerable ARea Tool]) were modified to address issues specific to uncontained engine debris damage assessments. The modified tools were named FASTGEN 3.2 DB (Disk Burst) and COVART 4.1 DB [2, 3]. Phase I was completed in FY97. Phase II's emphasis was on applying the data gathered and tools developed during Phase I to representative airframes, assessing the tools' adequacy in addressing the problem, and determining any changes necessary to improve the tools [4]. Phase II, which included McDonnell Douglas and Boeing – Seattle testing of UEDDAM and feedback for UEDDAM improvements, was completed in FY98. Phase III's goal was to integrate the feedback from Phase II into tool improvements and to produce a releasable version of the tools. Under Phase III, the newest version of FASTGEN (Version 5 Alpha) without modifications was used in UEDDAM. Additionally, COVART's external bursting algorithm was used to minimize modifications required to COVART. Phase III was initiated in FY99 and completed at the end of CY99 resulting in UEDDAM v1.0.

Once developed, the SURVICE Engineering Company and NAWC-WD undertook a study using UEDDAM to assess the probability of catastrophic *hazard* for generic aircraft for various uncontained engine debris events. During the course of this study, improvements to UEDDAM were made and the sample cases were updated resulting in UEDDAM v1.1 and, subsequently v1.2. Based on the lessons learned from two generic aircraft studies, which included a generic twin-engine commercial aircraft and a generic business jet [5], additional updates were made to the UEDDAM code resulting in UEDDAM v2.0. As Aviation Rulemaking Advisory Committee (ARAC) participants use the tool in their analyses, feedback regarding the intended function and requests for changes are constantly reviewed for use in the next phases of development. For example, industry input indicated a need for an intermediate step between the failure analyses

commercial manufacturers do as part of their reliability and system safety assessments and the failure analysis trees required as input into UEDDAM. Accordingly, as part of UEDDAM v3.0 development efforts, an optional input file, LOGIC FILE, and the code required to convert it to a format suitable for COVART was implemented. Additionally, the ability to obtain hole size statistics from a UEDDAM analysis was implemented in UEDDAM v3.0 based on an industry-identified need for data that could support decompression analyses. Another example of industry involvement in UEDDAM development is the 2008 effort to include a FASTGEN conversion tool that could be used from within Patran® (a CAE modeling and pre-/post-processing software). As it is common for airframe manufacturers to have a finite element model (FEM) of their aircraft to support structural analyses, a conversion utility for Patran models makes building the geometric model for a UEDDAM analysis simpler (at least for structural components). Finally, several other pre/post processing tools (e.g., *Analysis_Tool*, *FTDT*, *Run_Setup*, *mac*, and *stl2fast4*) have been developed over the years as the need arose.

This user's guide documents the input requirements, usage, and outputs of UEDDAM v6.0 and its associated tools and utilities.

1.2 PURPOSE

This guide not only defines the file requirements and outlines UEDDAM methodology, but also acts as a tutorial, illustrating step-by-step how to perform an aircraft engine rotor burst analysis with UEDDAM – from establishing the analysis objectives and performing a criticality analysis to executing the program and post processing the results. Excerpts from the documentation for COVART6 [6, 7] are provided in the appendices to support construction of the geometric target description and other required input files that are not generated by UEDDAM. Additionally, sample cases are provided for the user to verify that the program is working correctly and serve as examples for developing new assessments.

2 MODEL OVERVIEW

2.1 UEDDAM ANALYSIS PROCESS

As laid out in the Advisory Circular (AC) 20-128A User's Manual [8], there are five steps in performing a complete safety and risk analysis of an aircraft. These steps are summarized as follows:

1. Establish (at the design definition) the functional hazards that can arise.
2. Establish a functional hazard tree.
3. Establish the aircraft geometry, fragment trajectories, and trajectory ranges both for *translational* and *spread risk angles* (see Figures 2-2 and 2-3) for each damage.
4. Apply risk factors, such as phase of flight, and calculate the risk for each threat (i.e., hazardous combination as per the functional hazard tree) for each rotor stage.
5. Tabulate, summarize, and average all cases.

UEDDAM, in conjunction with its post processors, helps perform steps 3 thru 5 listed above, thereby reducing the workload of the analyst. Note also that because 14 CFR (Code of Federal Regulations) 25.903(d) requires that the hazard from an uncontained engine debris release be minimized, this process (specifically steps 3 thru 5) can and should be iterated upon during the design phase (e.g., different configurations of critical systems can be assessed and compared so that those configurations that perform best when faced with an uncontained debris release can be identified for the final design).

The particular process used to perform a UEDDAM safety and risk assessment of an aircraft can be seen in Figure 2-1. Note the correspondence to the AC 20-128A process steps, 1 through 5. One of the purposes of this manual is to guide a user through this process. Thus, periodically throughout this user's manual, Figure 2-1 will be referenced in order to highlight the current location in the UEDDAM analysis process.

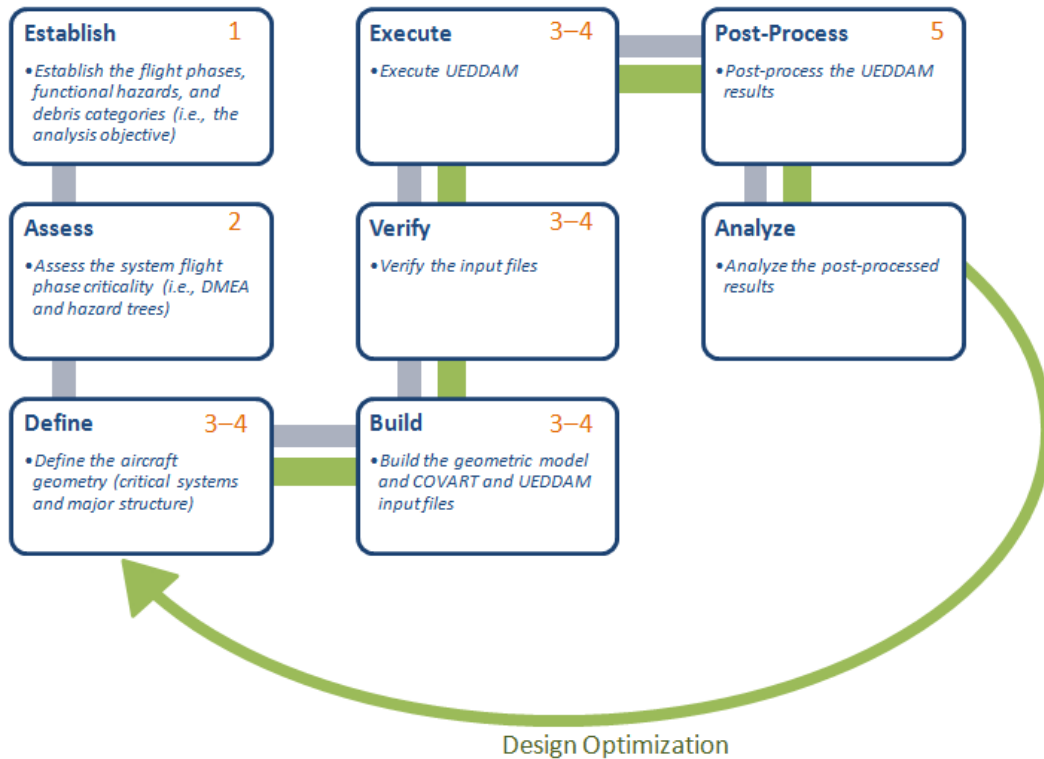


Figure 2-1. UEDDAM Analysis Process.

The first step in performing a UEDDAM analysis is to establish the analysis objectives (i.e., the particular flight phase(s) of interest, the functional hazard(s) , and the *debris category(ies)* to be assessed). Second, and most important, is to perform a system flight phase criticality analysis thru the use of a DMEA (Damage Modes and Effects Analysis) and associated hazard trees. The purpose of this step is to identify critical systems and components as well as their redundancies and damage modes. Next, using the information developed in the second step, define the aircraft geometry, making sure not only to include components of critical systems, but major structural components as well as those components that provide significant shielding. The next step involves input file generation – specifically, construction of a geometric model and other necessary UEDDAM input files based on the specifications laid out in the first three steps. The fifth step is to verify the input files by performing a “simple” UEDDAM run and observing the results. (Selecting a narrow release angle where the tangent impacts the fuselage can provide the most insight.) Once the input files have been verified, all that remains is to execute UEDDAM, post process the output data, and analyze the results.

Each of the aforementioned steps will be discussed, in detail, in the context of this user’s guide.

2.2 GENERAL DESCRIPTION

The main UEDDAM program is written in the C++ programming language and performs file input and output control, data checking, run control of the COVART6 program, and hazard level result summation and averaging. Inputs define the aircraft geometry, debris characterizations, Monte-Carlo iteration control, and hazard level(s). Outputs may be generated for each engine disk in summary format, for each critical component contribution per Monte-Carlo iteration, or as a tabulation of risk angles for each critical component per disk.

The purpose of running FASTGEN* is to develop line-of-sight (**LOS**) data for use in the subsequent analysis model, COVART. The LOS data consists of the following specific vehicle component data: 1) identification number, 2) impact coordinate location, 3) thickness, and 4) impact obliquity angle. These data are calculated from a three-dimensional geometric description along one or more straight-line paths (shotlines) defined by a set of input controls. These shotlines are generated for each release origin defined by the user in the UEDDAM input file.

UEDDAM models multiple disk segment, large fragment, and small fragment debris effects on an aircraft system. **Debris types** may be assessed independently or together as a single evaluation of the hazard for specified debris release events (called debris categories). COVART performs penetration assessment for the disk debris characterizations, considering component criticality and redundancy, and summarizes the component contribution to aircraft hazard level for a given release origin. The results from COVART are accumulated by UEDDAM for multiple iterations of fragment trajectories from a release origin, multiple release origins about the circumference of the rotor disk, multiple rotor assemblies, and multiple engines.

2.3 UEDDAM METRICS

UEDDAM provides the analyst with a probability of hazard (P_{HAZ}), which is defined as the probability of an event causing sufficient damage to result in a predefined hazard level (e.g., catastrophic damage) to the aircraft. Not only is the user provided with a total system level P_{HAZ} , but additionally, UEDDAM calculates P_{haz} for the various components to illustrate their contributions to P_{HAZ} .

In addition to the shotline-based hazard level assessment, the existing manual method defined in AC 20-128A is automated within UEDDAM to provide translational risk

* With the release of COVART Version 6.0, FASTGEN and COVART were integrated into a single model, known simply as COVART6. In order to remain backwards compatible, however, COVART6 allows the user to execute the model in one of three different modes: FASTGEN5 Legacy Mode, COVART5 Legacy Mode, and COVART6 Integrated Mode. Whereas UEDDAM Version 3.1 made use of the FASTGEN and COVART legacy mode options, UEDDAM Version 4.0 and later versions make use of the Integrated Mode option. However, to distinguish the shotlining portion of COVART6 from the penetration and hazard level computation portions, throughout this manual, the statements “running FASTGEN” and “running COVART”, respectively, are still used.

angles, spread risk angles, and **probability of hit ratios**. In the manual assessment, the large disk segment (typically 1/3 disk including 1/3 blade height) cuts a rectangular swath along its release trajectory tangent to the engine rotation at the point of separation. Since the segment is also rotating about its own center of mass after separation, the height of the swath is thus equal to twice the greatest radius measured from the segment center of mass (see Figure 2-2). (Note that UEDDAM approximates the swath by doubling the greatest radius measured from the segment *centroid*.) For each aircraft critical component, the initial and final engine rotation angles that cause the swath to intercept the component are determined (see Figure 2-3). The ratio of the angle between the release trajectories bounding an aircraft component (called the translational risk angle) to 360 degrees subsequently represents the probability of that component being struck by the separating disk segment ($P_{h(t)}$). Detailed presentation of the manual method is provided in the AC 20-128A.

In addition to the translational risk angles (TRA), the UEDDAM output table includes spread risk angles (SRA), illustrated in Figure 2-3, as well as the resulting probability of hit (P_h) values. The final P_h value for each component is computed as the product of the $P_{h(t)}$ for translational angles multiplied by the $P_{h(s)}$ for the spread angles:

$$P_h = P_{h(t)} \times P_{h(s)},$$

where $P_{h(s)}$ is defined similarly to $P_{h(t)}$ (i.e., the ratio of the SRA to the difference between the fore and aft spread limits).

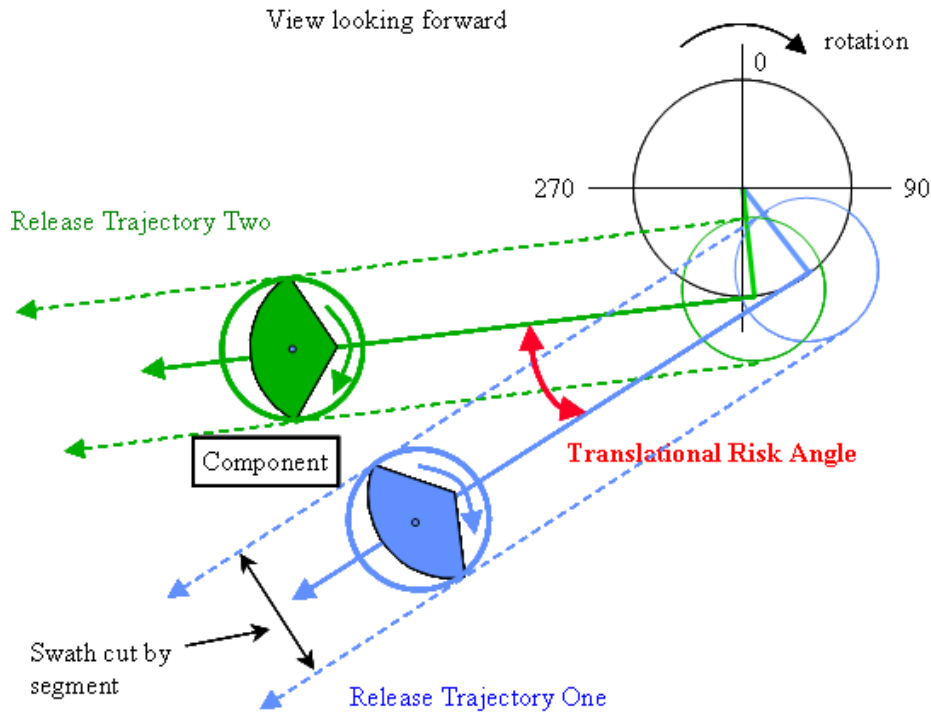


Figure 2-2. Typical Trajectory Plotting [8].

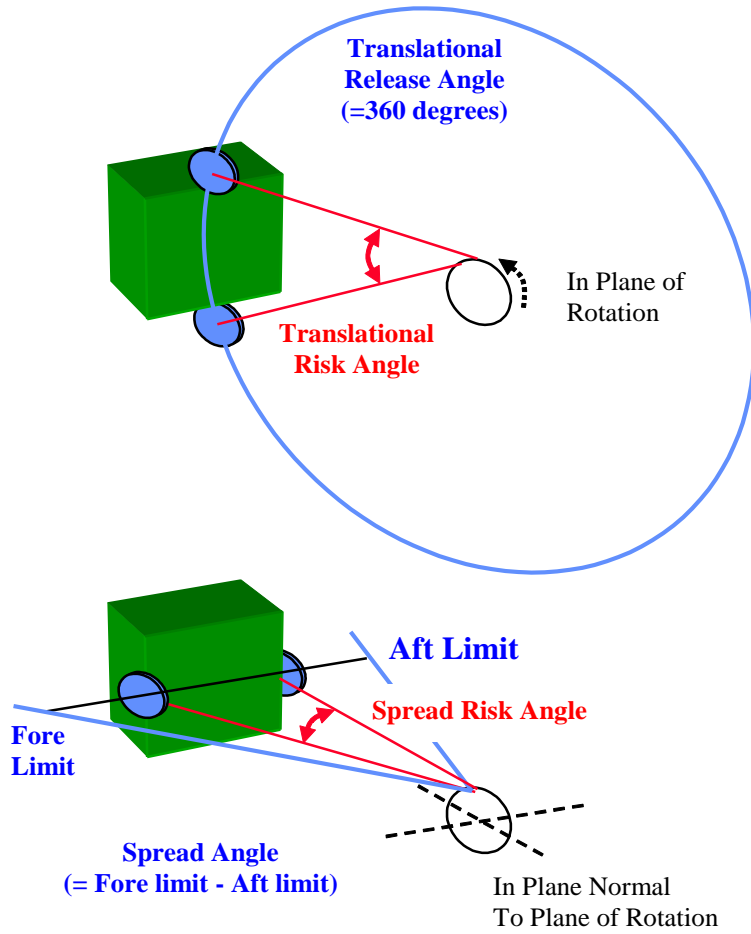


Figure 2-3. Definition – Threat Window [8].

3 UEDDAM FEATURES AND METHODOLOGY

3.1 UEDDAM PROGRAM FLOW

The following paragraphs outline the general process followed by UEDDAM during an assessment run. Note that it is the user-created control file (see Section 5.3.1) that guides the UEDDAM code through the following tasks. Program data structures within UEDDAM are dynamically allocated wherever possible. Intermediate files are used as necessary during execution and deleted when no longer required. Figure 3-1 illustrates the generalized UEDDAM program flow.

3.1.1 Load UEDDAM Control Data

A UEDDAM run session is initiated by executing the “ueddam.exe” code located in the “bin” directory. A control file is required for the execution of UEDDAM. The name of the control file may be specified on the command line or omitted. If the name of the control file is omitted on the command line, then UEDDAM will look for the control data file named “UEDDAM_Control” in the working directory. The control file defines parameters and identifies external support files containing the required data for the session.

3.1.1.1 Initialize Structures and Variables

UEDDAM sets the default parameters, clears error flags, and creates data structure templates. This prepares UEDDAM for reading the input files and loading data into dynamically allocated memory for processing.

3.1.1.2 Read Input Controls and Data

Most input controls and data may be entered in an order convenient to the analyst. UEDDAM reads all input data and loads its internal structures prior to processing any rotor burst events. In addition to reading in the input controls and data, it is during this step that a fault tree file containing logic statements (if specified in the UEDDAM control file) is converted to a COVART6 *MV* file by a call to the executable *mkwgtf.exe*. Refer to the input formats (Section 5) for requirements and limitations specific to each input record.

3.1.1.3 Verify Control and Input Data Completeness

At completion of input loading, UEDDAM verifies that required inputs for a UEDDAM run have been entered. It also performs basic checks that valid inputs have been entered within limits specified for the given record. If errors are found in the input deck, they are reported to the error log file specified in the control file, or, if none is specified, to the

command window (albeit in less detail). If fatal errors are found, UEDDAM will terminate the run.

3.1.1.4 Estimate Resource Usage

Upon successful input verification (and if not disabled by the user), UEDDAM will provide a very rough estimate of the run time and disk space needed to perform the run as defined by the input files. The user will then be given the option to terminate or continue with the run based on these estimates.

3.1.1.5 Generate Static COVART Support Files

The COVART6 *HEAD* and *THREAT* files are constant data sets based upon the current session parameters. To avoid looping delays caused by rewriting constant data, these files are created prior to entering the main event loop.

3.1.2 Remove Engine Near-Field Components from Target Geometry File

Near-Field components are considered to be non-shielding components adjacent and/or internal to the engine. Debris characterizations currently consider these components to be part of the debris being released from a rotor burst event and therefore must not be considered as part of the remaining aircraft. UEDDAM removes these components from the analysis to eliminate errors in debris penetration predictions. However, it is important to note that if a *critical* component is included in the near-field list for a target geometry, it must also be included in the auto-fail list (see Section 3.1.3) so that its failure is appropriately accounted for in the overall hazard of the aircraft.

3.1.3 Generate Engine/Event-Dependent COVART Support Files

One role of the COVART6 *MASTER* file is to reference data source files for a COVART run. For a given event, the records associated with the COVART run within this file are constant and are created outside the release point and iteration loops to improve loop timing efficiency.

Engine-mounted components fail when the respective engine is lost. Even though these components are not necessarily damaged, they must still be counted as no longer functioning when evaluating component redundancy in the aircraft fault tree. To accomplish this, UEDDAM modifies the master COVART6 *JTYPE* file to include automatic failure components by engine reference.

The JTYPE is additionally modified to include a list of components for which a hole size output is requested as well as entries for material types to be considered in the analysis. These records are added to the end of the new JTYPE file for passing to COVART6.

3.1.4 Determine Translational and Spread Intercept Angles

These boundaries provide spread and translational angular limits for the target geometry components relative to each burst disk center and orientation. Intercept angle limits provide the ability to reduce target geometry file size and compute probabilities of hit for comparison with the manual procedure discussed in Reference 8.

3.1.5 Perform Geometric Reduction

If specified by the user in the UEDDAM control file, a reduced **CBULK** target geometry file is created for COVART6 processing. (Note that the original geometry file remains unchanged.) UEDDAM uses the disk debris characterizations, component intercept angle limits, and event release point controls to determine which components lie within the possible debris trajectories. The reduced geometry file contains only those components that could possibly be hit by debris. This minimizes the number of components that COVART6 must process within each iteration. Additionally, UEDDAM interrogates the fault tree to determine if any combination of failures of the remaining components could result in any of the defined hazard levels. If none of the remaining components meet this criterion then no FASTGEN or COVART runs are required and the execution skips to Section 3.1.8, reporting a P_{HAZ} of 0.0 for that release point (see Section 5.3.1.4 for additional information).

3.1.6 Generate CBULK and MASTER Files and Run COVART6

Outside of the iteration loop, the release-dependent CBULK target geometry file is converted to binary format in order to decrease disk space requirements and reduce processing time.

All of the fragments and disk segments for a given event iteration are represented as divergent rays (or shotlines) in UEDDAM. Shotlines are specified in the UEDDAM-generated MASTER file as a combination of **UDASPECT** and **UDAIM** records. It is during generation of the iteration-dependent MASTER file that UEDDAM performs **hypersampling** of the target geometry, if specified in the UEDDAM control file. Hypersampling allows for the generation of specific shotlines (in addition to those comprising the debris swath bundle) used to make sure that the swath does not inadvertently miss “small” components in the geometry (see Section 5.3.1.5 for additional information).

The COVART6 program executes externally to UEDDAM and processes the iteration-dependent MASTER and release-dependent CBULK files. If the user specifies in the UEDDAM control file that he or she is executing on a multi-processor machine and wishes to take advantage of this capability, UEDDAM will divide the COVART6 runs among the number of processors specified (see Section 5.3.1.6 for additional information).

Output files are read by UEDDAM for summation and averaging by release point and rotor burst event.

3.1.7 Perform Monte-Carlo Analysis

For each release point specified by the user, random number draws are made within the user-defined rotational release point variance, fore/aft spread angle, and fragment orientation limits. When the number of iterations in the corresponding EVENT record is greater than 1, each release point is repeated with new random number draws for each iteration prior to processing the next release point. The steps discussed in Section 3.1.6 are repeated within the multiple-iteration loop. When the detailed output file is specified, component probability-of-hazard, P_{haz} , values are printed for each iteration.

3.1.8 Intermediate-Process Hazard Level Results

Component P_{haz} is averaged over all iterations for each release point. When the detailed output file (DETAIL FILE) is specified, these average component P_{haz} values for each release point are appended to the data discussed in Section 3.1.7. When the hazard summary output (SUMMARY FILE) is specified, the system (total aircraft, not component) average P_{HAZ} is printed to the main summary output file for each release point.

In addition to hazard probabilities, when the hole size output file (HOLESIZE FILE) is specified, UEDDAM calculates hole size statistics for each group defined in the hole size group input file. For each release point, minimum, maximum, and average cumulative hole sizes are recorded for all groups. Additionally, for each release point, UEDDAM prints the average number of fragment impacts and the average number of fragment penetrations for each group to the HOLESIZE FILE.

Finally, when any one of the significant iteration output files (SIGITSYS FILE, SIGITCRIT FILE, SIGITCHOLE FILE, or SIGITSHOLE FILE) is specified, UEDDAM records the iteration number and value (aircraft P_{HAZ} , cumulative or single hole size, or number of critical components hit) that meets/exceeds the criterion specified in the UEDDAM control data file.

3.1.9 Repeat Process for Each Nominal Release Point of a Disk Stage

For each nominal release point, the steps defined in Sections 3.1.4 through 3.1.8 will be repeated.

3.1.10 Final-Process Hazard Level Results

Component P_{haz} is averaged over all release points for each event. When the hits summary output (HITSUM FILE) is specified, these average component P_{haz} values are printed for each release point to the HITSUM FILE. For the hazard summary output, system (total aircraft, not component) average P_{HAZ} is appended to the SUMMARY FILE for each event.

3.1.11 Repeat Process for Each Disk Event of an Engine

For each disk stage within an engine (event), the steps discussed in Sections 3.1.3 through 3.1.10 will be repeated.

3.1.12 Repeat Process for Each Engine

For each engine, the steps described in Sections 3.1.2 through 3.1.11 will be repeated.

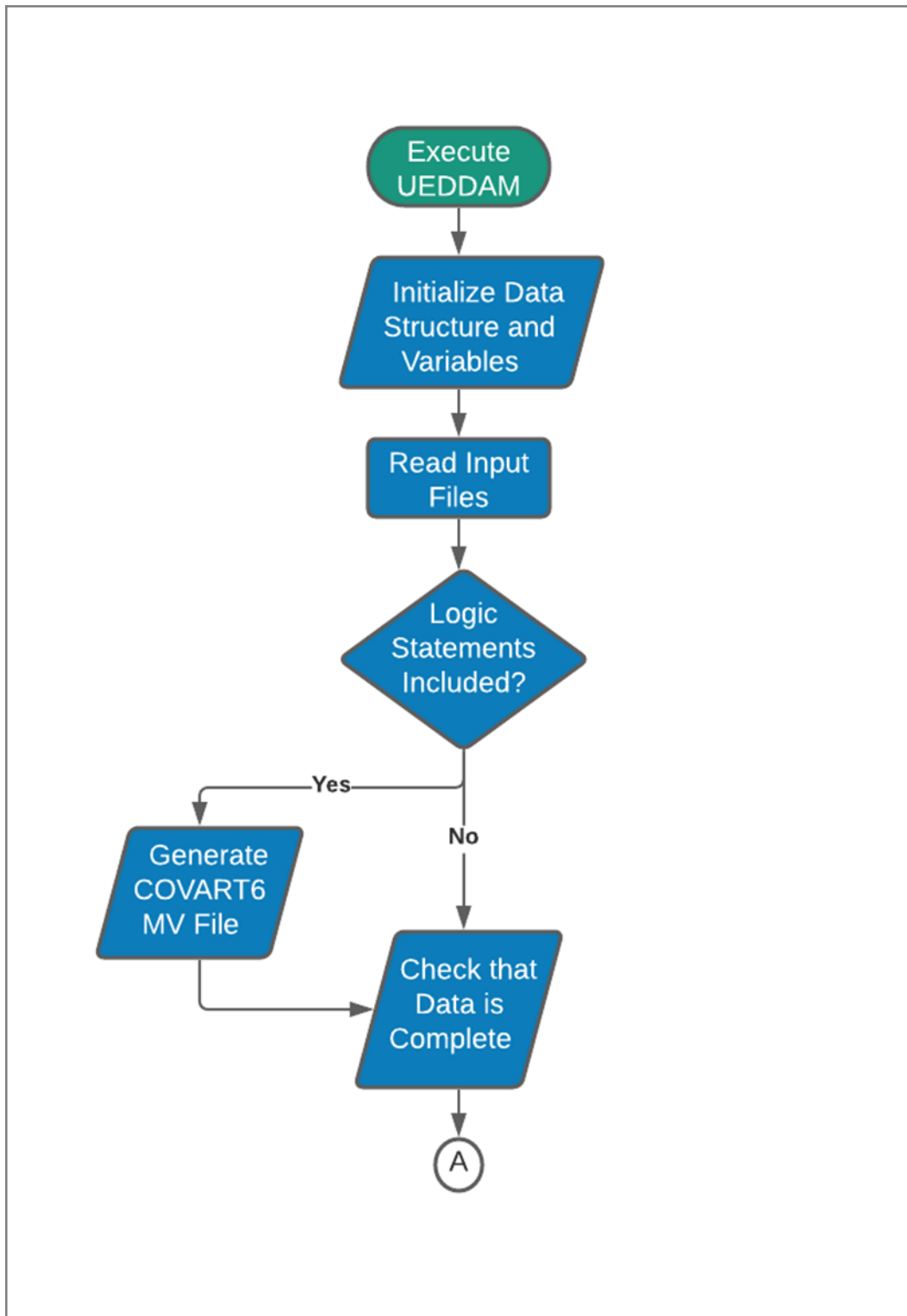


Figure 3-1. Generalized UEDDAM Program Flow (Page 1 of 5).

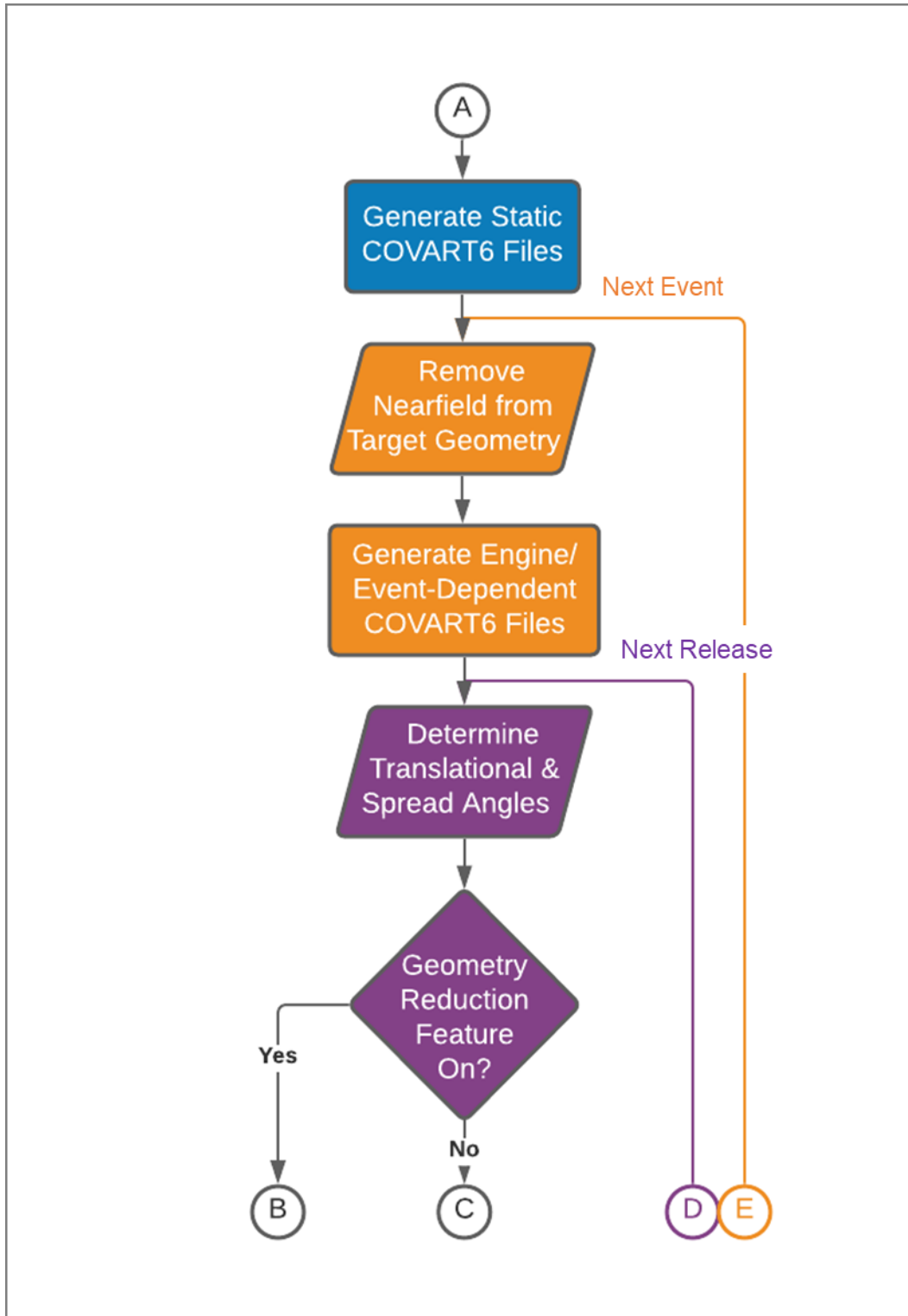


Figure 3-1. Generalized UEDDAM Program Flow (Page 2 of 5).

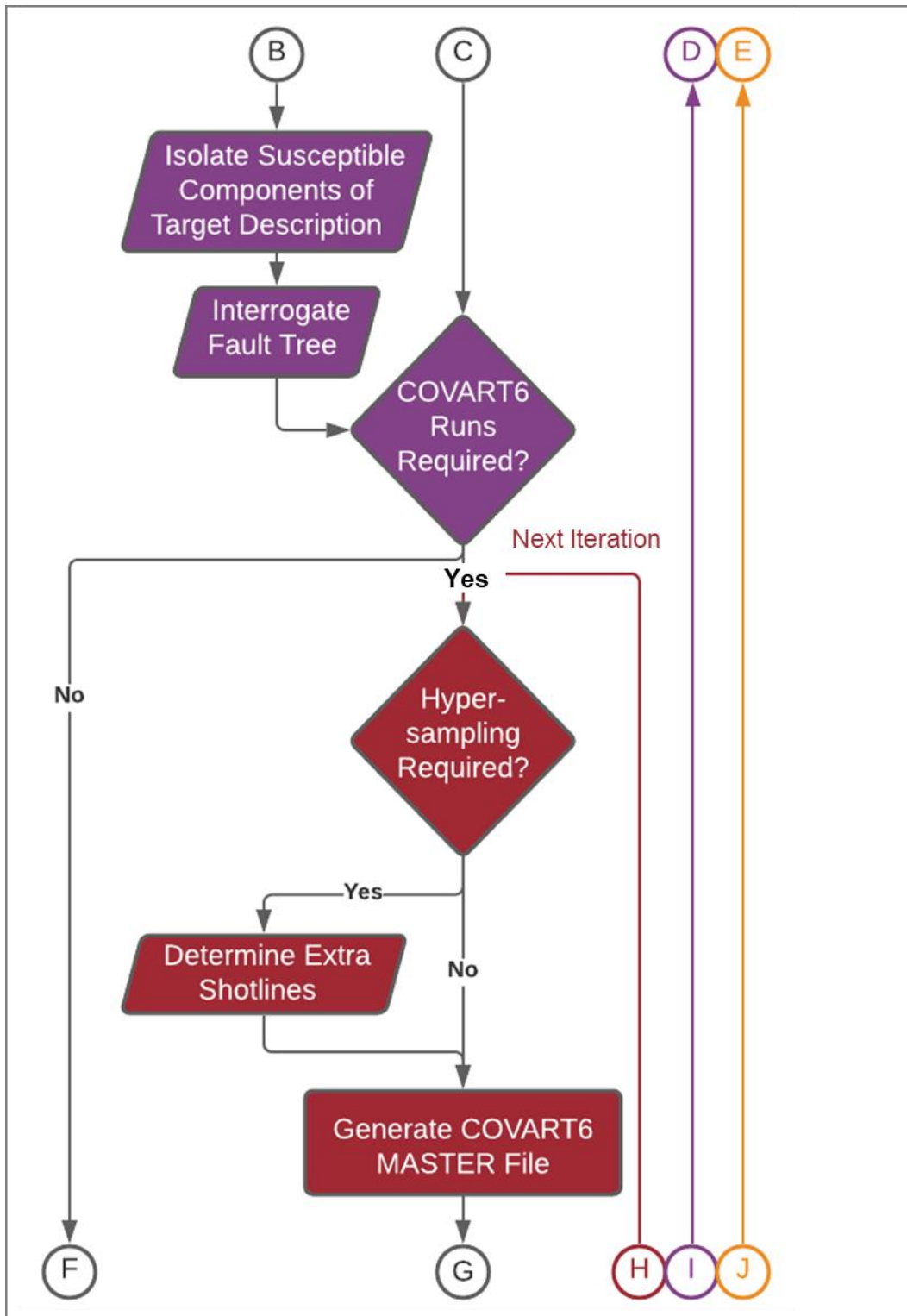


Figure 3-1. Generalized UEDDAM Program Flow (Page 3 of 5).

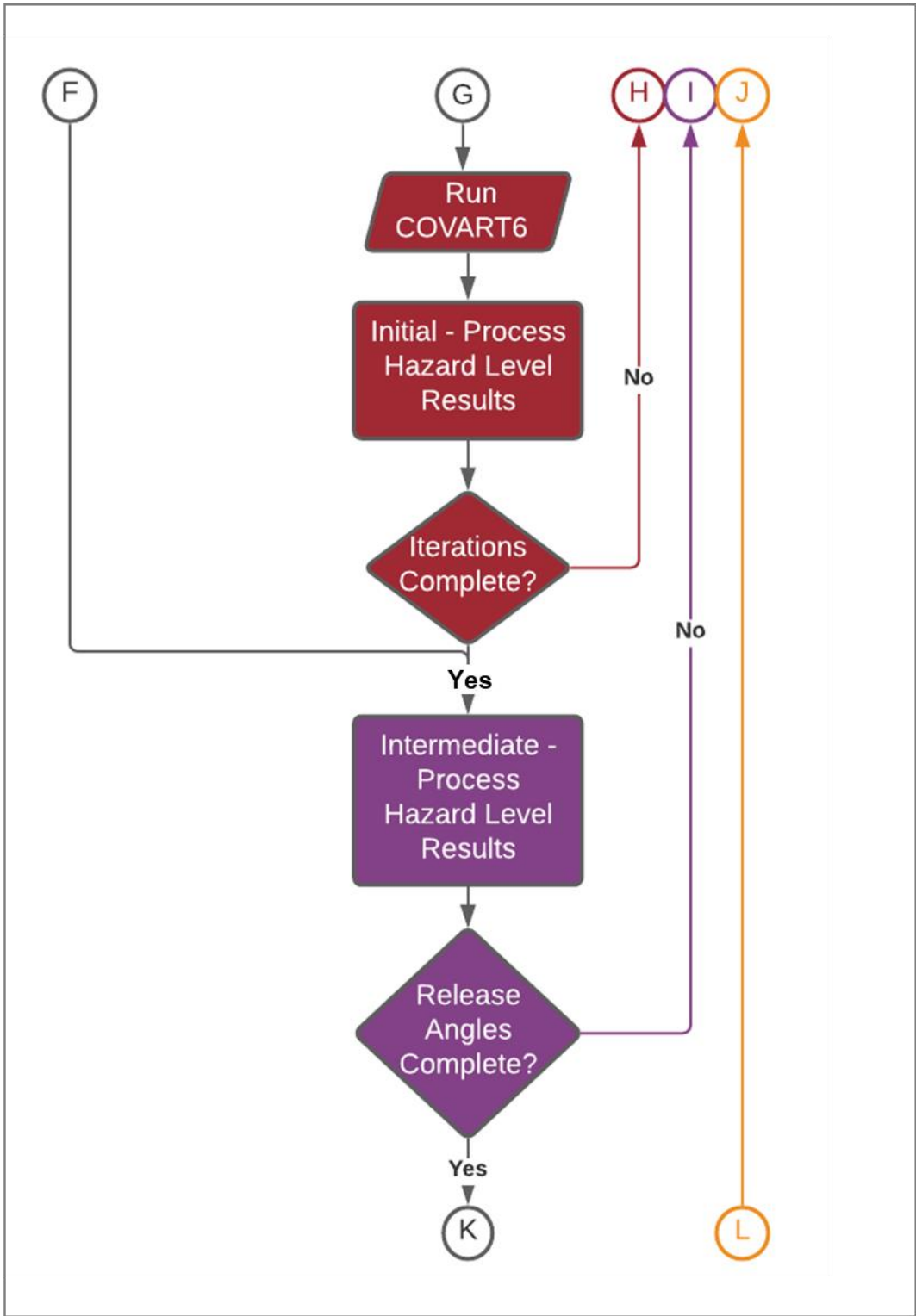


Figure 3-1. Generalized UEDDAM Program Flow (Page 4 of 5).

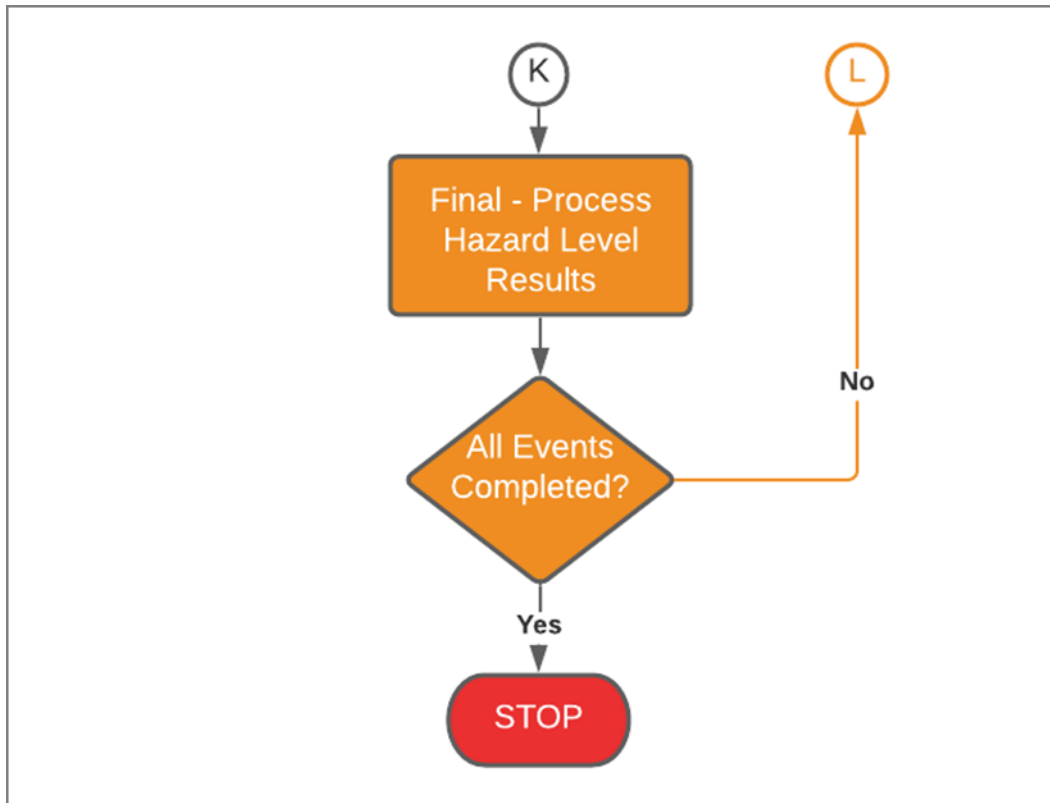


Figure 3-1. Generalized UEDDAM Program Flow (Page 5 of 5).

3.2 DEBRIS TYPE ANALYSIS METHODS

In accordance with the AC20-128A, three types of rotor burst debris are considered within UEDDAM:

1. Disk segments
2. Large fragments
3. Small fragments

Any of these fragments can be generated when an uncontained engine failure occurs and it is understood that not all fragments are liberated in every service failure. However, as the regulations require minimization, the worst case event must be considered for each engine and rotor stage on the aircraft. For some stages, this may require a multiple fragment analysis, in which more than one fragment (potentially of type 1, 2, and/or 3) is released simultaneously. It is this specific type of analysis (one that was of importance to the ARAC PowerPlant Installation Harmonization Working Group [PPIHWG]) that drove the development of UEDDAM.

3.2.1 Disk Segment Debris Type

The disk segment debris type represents the separation of a large section of disk as a single intact segment from the rotor assembly. The trajectory is primarily in the rotor's plane of revolution but UEDDAM allows for fore or aft deviations from this plane (i.e., spread angle). UEDDAM can also consider multiple segment releases (i.e., tri-hub rotor burst) through user-defined offsets defining the lead/lag angle of subsequent segments from the first segment release trajectory. For each segment released, a parallel grid of shotlines is generated to represent the swath cut by the respective segment (see Figure 3-2). Since the disk segment is rotating about its center of mass along the swath, the shotlines that represent the outer limits of the swath potentially have different penetration characteristics than the shotlines located near the center. For this reason, UEDDAM permits the user to apply a mass weighting factor to shotlines within each grid as a function of the distance from the segment center of mass (see Section 5.3.2.4).

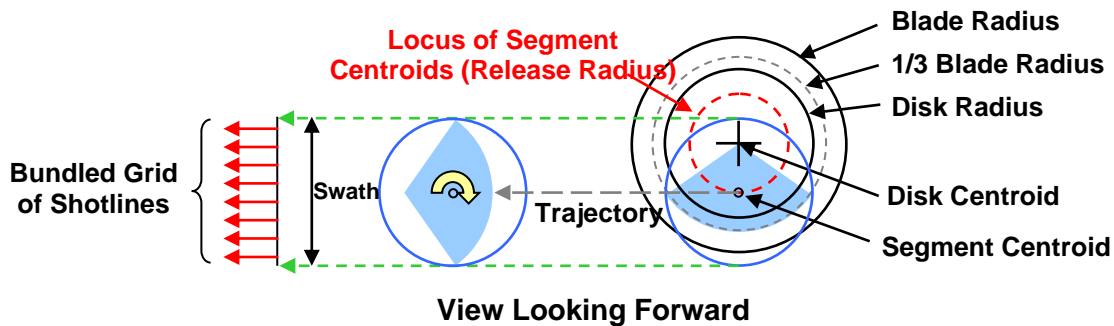


Figure 3-2. Debris Segment Swath Gridding.

UEDDAM processes multiple iterations and release points with random number draws from the fore/aft spread angle and rotational release point variance. Output options permit the printing of individual iteration results and/or an average over all iterations for each hazard level for each disk event.

3.2.2 Large Fragment Debris Type

Fragment debris is characterized as either large fragments or small fragments (fragment size is relative to the size of the engine and aircraft components – see Section 5.3.2.5). For the large fragments (i.e., fan blades, first stage fan blade fragments, etc.), a single shotline trajectory is inadequate to represent the impact presented area of the fragment. In this mode, a grid of parallel shotlines is generated to portray the impacting area of each large fragment on the aircraft. User-defined mass distribution factors may be applied to the shotline grid as discussed for the segment debris. Dimensions of the shotline grid are based upon the orientation data of each fragment. For Monte-Carlo runs, release trajectories from multiple large fragments vary in both spread angle and translational release angle about a release point. Fragment orientation is also variable in the Monte-Carlo runs.

3.2.3 Small Fragment Debris Type

Small fragments represent the remaining disk and blade fragments and other miscellaneous debris exiting the engine case. For this debris (i.e., compressor and turbine blade fragments), individual shotlines are adequate to represent each fragment presented area impact on the aircraft. Each release trajectory is randomly drawn from user-defined spread angle limits and translational angle variances about a release point.

3.3 NEAR-FIELD FEATURE

The geometric model of an aircraft typically contains detailed internal and external engine components. Since debris trajectories for a UEDDAM analysis initiate inside the engine case but engine debris characterization data is often measured external to the engine (i.e., the debris slowdown in penetrating the engine has already been taken into account), the near-field feature in UEDDAM eliminates these internal components (identified in the near-field input file [NEARFIELD FILE]) from the geometric model so that the trajectory data remains valid. For example, all components within and including the engine casings (those components shown in red in the simplified engine model in Figure 3-3) might be listed in the NEARFIELD FILE for a UEDDAM run on this aircraft. However, the components external to the engine (those components shown in blue in Figure 3-3) might be excluded from the NEARFIELD FILE. Note that depending on the engine debris characterization data, all components included within the engine nacelle itself may need to be included in the NEARFIELD FILE.

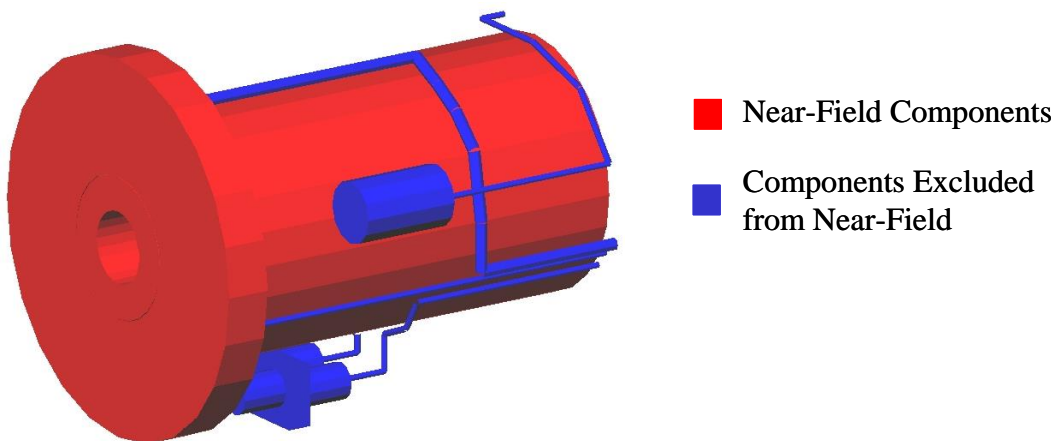


Figure 3-3. Generic Engine Near-Field Example.

3.4 GEOMETRY REDUCTION AND FAULT TREE INTERROGATOR

Due to the Monte-Carlo analysis that is performed for every release angle, FASTGEN run times can become excessive. In order to minimize the number of components that COVART6 must process within each iteration, UEDDAM, as a first cut, searches

through the geometric model and selects only those components that lie within the rotational release and fore/aft spread variances. (Note that UEDDAM not only considers critical components, but all components within the **hazard zone** so that shielding is unaffected.) UEDDAM then checks to see if failure of any of the remaining components can possibly lead to an aircraft hazard. If no combination of the selected component failures can result in a hazardous event, UEDDAM skips the execution of COVART6 and assigns a probability of hazard of 0.00 for that release. As an example, Figure 3-4 shows the geometric reduction process and resulting simplified geometry for a generic business jet. All components highlighted in red in the second picture lie within the hazard zone (visually represented as a transparent gray pyramid) for that particular release point. Note that for this particular engine event and release, COVART6 would have to be run in order to generate a probability of hazard, but the number of components it would have to process would be greatly reduced. For example, the pressure bulkhead (shown as a gray disk in the first two images of Figure 3-4) is removed from the final geometry due to its location forward of the debris zone.

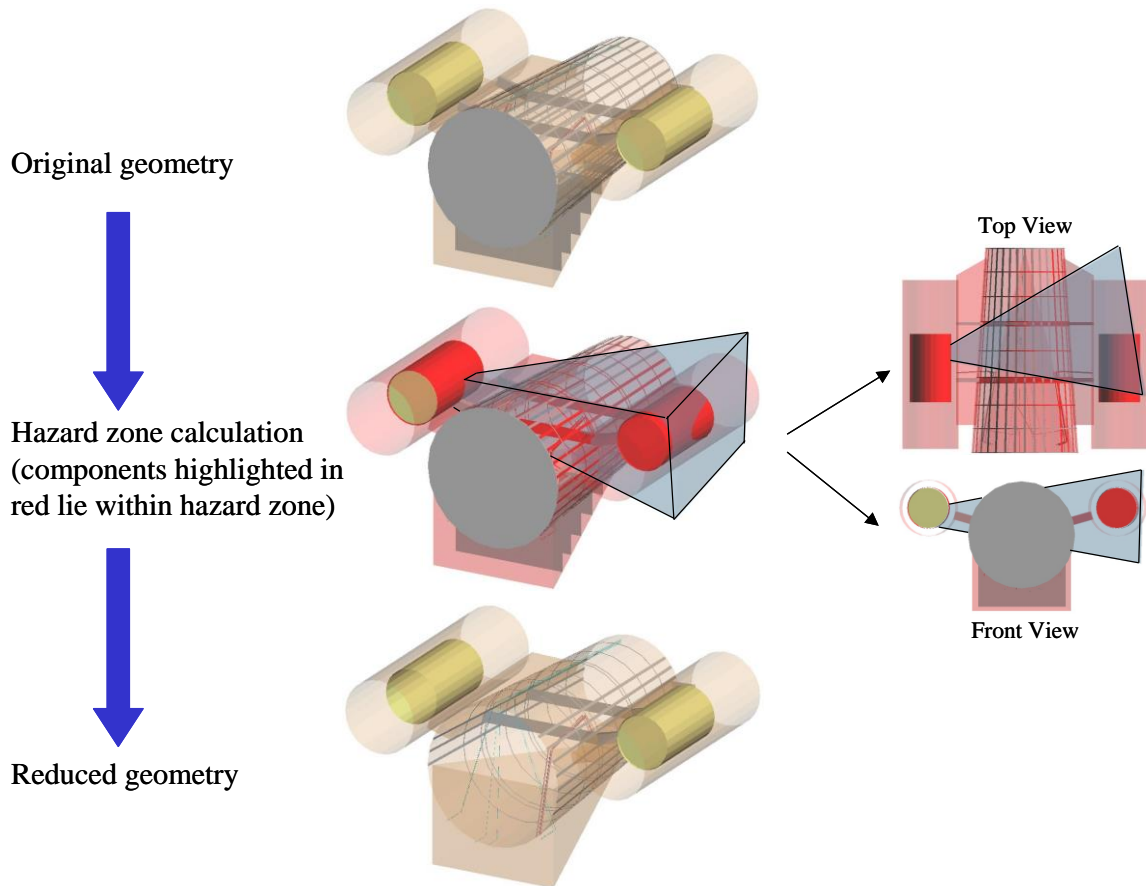


Figure 3-4. UEDDAM Geometry Reduction Example.

3.5 MONTE-CARLO DEBRIS CHARACTERISTICS

To obtain more informative results, UEDDAM has the capability to average P_{haz} over multiple iterations in which debris trajectories are randomly released within user input fore/aft and rotation spread angle limits. The value for the rotational release angle is drawn from a uniform distribution (as prescribed by the AC 20-128A), whereas the distribution from which to draw the fore/aft release angle is an option to the user (uniform, normal, or skewed[†], with uniform being the default) and can be used to more accurately represent the likelihood of debris trajectories within the fore/aft angle limits. Figure 3-5 shows an example of the three available distributions applied to a generic twin-engine aircraft. The red lines in the pictures on the left indicate the small fragment trajectories (released from the High Bypass Ratio Engine number 1 fan at an angle of 0°) for five iterations. The fore and aft angle limits used were 15° and -30° , respectively.

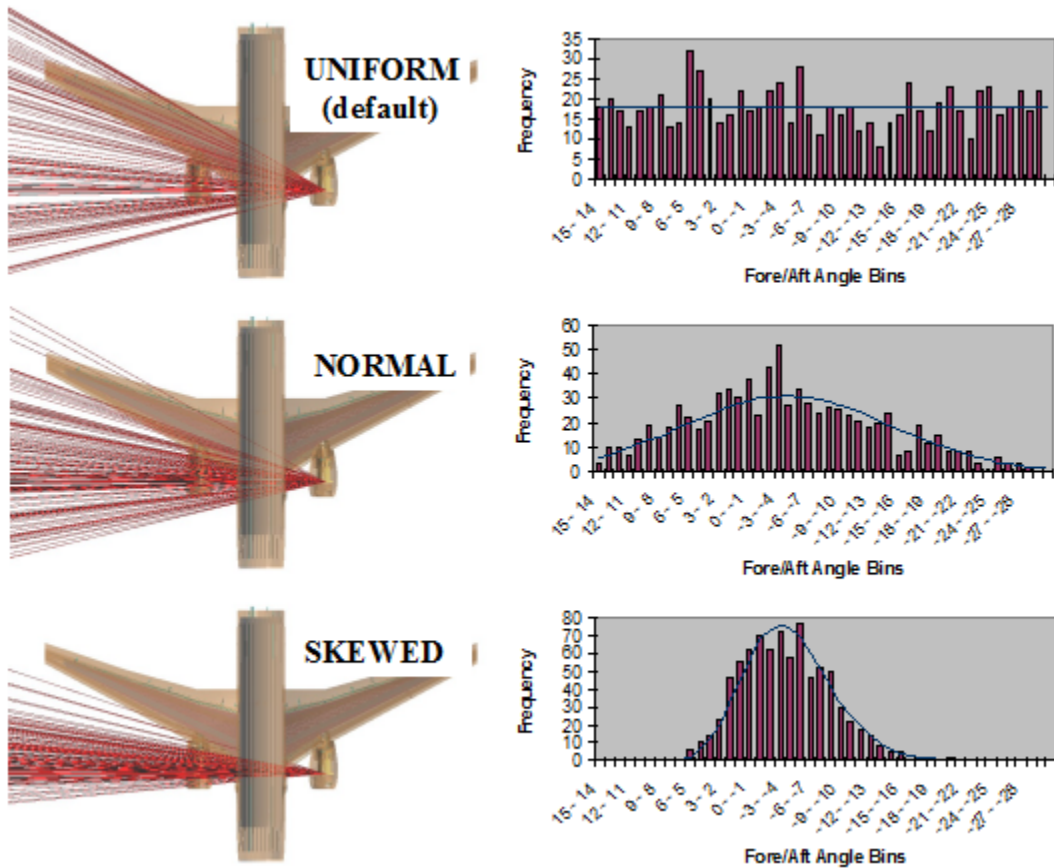


Figure 3-5. Generic Twin Engine Aircraft Monte-Carlo Example.

[†] Note that the skewed distribution can be skewed either fore or aft. The particular distribution depicted in Figure 3-5 is skewed aft.

3.6 AUTO-FAIL FEATURE

Given an uncontained engine failure, the engine from which the debris was released as well as all components and systems driven by that engine will no longer be operational. In order to correctly interpret the fault tree within the model, then, it is necessary to make sure that these components are failed in conjunction with the engine itself even if they are not physically damaged by the debris release. UEDDAM accomplishes this through the use of the auto-fail input file (AUTOFAIL FILE). UEDDAM reads the input file, determines which components are associated with the particular engine undergoing the uncontained failure, and adds the appropriate AUTOFAIL records to the end of the COVART6 JTYPE file. These records indicate to COVART that the individual components should be assigned a P_{haz} of 1.0 at all applicable hazard levels whether or not they appear on any debris shotlines. As an example, consider the engine-driven hydraulic pump (highlighted red in Figure 3-6) along with the associated portion of the fault tree (set to a damaged state). For an uncontained failure of this engine, the engine-driven hydraulic pump (Hyd. Pump #1), if included in the AUTOFAIL FILE for this engine, would be automatically failed by UEDDAM; thus, if the uncontained engine debris managed to defeat Hyd. Pump #2, this branch of the fault tree would be defeated.

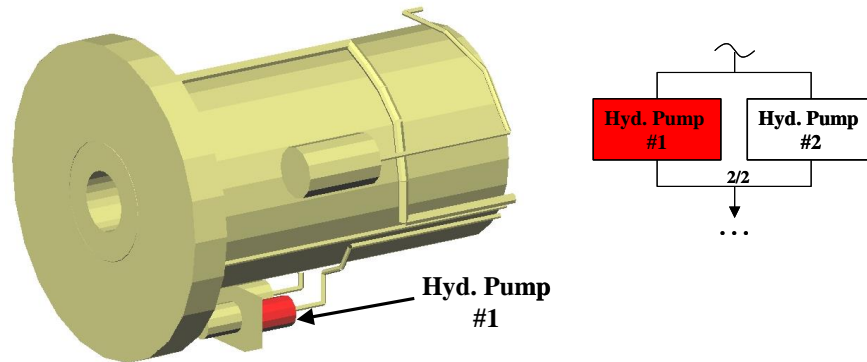


Figure 3-6. Generic Engine Auto-Fail Example.

3.7 PENETRATION EQUATIONS

The current UEDDAM penetration equations are based on those used by the military and civilian vulnerability communities, the goal being to rapidly estimate the residual velocity of various engine blade and disk debris into aircraft structures. The first is the V_{50} equation, which is originally derived from an energy absorption equation taken from the 1977 FAA document RD-77-44 [9]. The FAA V_{50} equation, coded for UEDDAM versions 4.0 and earlier, is given by

$$V_{50} = \sqrt{\frac{2LC_s t^2}{m \cos^2 \theta}}$$

Table 3-1. List of Parameters Used in V_{50} Equation

Symbol	Definition	Units
L	Debris presented area perimeter	meters
C_s	Plate shear constant [10]	Pascals
t	Plate thickness	meters
m	Debris mass	kilograms
θ	Obliquity angle	degrees

Note that the debris presented area perimeter, L, can be calculated as

$$L = 2[\lambda\cos(\delta) + \tau\sin(\delta)] + 2\omega, \text{ (see Figure 3-7).}$$

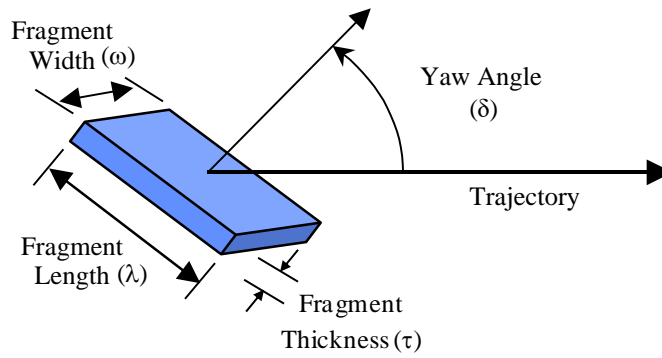


Figure 3-7. Description of Fragment Parameters.

V_{50} , defined as the velocity at which half the fragments will perforate the target, is treated as the ballistic limit velocity (i.e., the velocity at which a fragment has zero energy after penetrating a target). This equation supports the conservative assumption that the normal component of the impact velocity is the only contributor to the kinetic energy required for penetration of the material.

With the development of UEDDAM v4.1, however, and in light of more recent studies into the penetration of launch and reentry debris by the FAA [11], the overly conservative normal component assumption in the FAA V_{50} equation has been removed in favor of the inclusion of an adjustment factor, gamma (γ). This new equation,

$$V_{50} = \sqrt{\frac{2LC_s t^2}{m \cos^2 \gamma \theta}},$$

indicates that more than just the normal component of the impact velocity contributes to the kinetic energy required to penetrate the impacted material (note that a value of $\gamma=1$ returns the updated V_{50} equation to its original form). In the case of launch and reentry debris, analysis indicates that the optimized γ value of 0.707 is the least conservative value required to ensure the modified V_{50} equation is conservative when compared to a limited test data set. However, in order to provide flexibility, the user is requested to

supply values of γ ($0 \leq \gamma \leq 1$) for five obliquity angle bins, empirically derived based on examination of non-ideal projectile and compact fragment impact data provided in References 12 and 13: 0° – 30° , 30° – 45° , 45° – 60° , 60° – 75° , and 75° – 90° (see Section 5.3.1.8 for the control file inputs).

With the development of UEDDAM v4.3, an additional modification has been made to the FAA V_{50} equation. In addition to the adjustment factor gamma (γ), the equation now also considers two values for the shear constant C_s in order to better represent the shielding capabilities of materials. The two values of C_s are referred to as C_{s_50} and C_{s_0} . The C_{s_50} value is the standard, conservative C_s that has been used in all previous versions of UEDDAM and that is provided in the standard material file. The accepted C_{s_50} values have shown good correlation in testing for those cases where fragments demonstrate moderate to high residual velocities. For those cases where fragments impact at or below the ballistic limit velocity of the material, however, a second, higher coefficient is warranted. The C_{s_0} value is a larger C_s calculated by extrapolating penetration test data in a different manner.

UEDDAM uses the two values of C_s to calculate the V_{50} by following the subsequent steps:

1. Calculate V_{50} using the C_{s_0} value

$$V_{50} = \sqrt{\frac{2LC_{s_0}t^2}{m\cos^2\gamma\theta}}$$

2. Compare the calculated V_{50} to the impact velocity of the debris.
 - a. If the impact velocity is less than the calculated V_{50} , the debris is assumed to have been stopped (i.e., residual velocity equals 0).
 - b. If the impact velocity is greater than the calculated V_{50} , an updated V_{50} is computed using C_{s_50}

$$V_{50} = \sqrt{\frac{2LC_{s_50}t^2}{m\cos^2\gamma\theta}}$$

The updated FAA V_{50} equation is treated as the ballistic limit velocity in the second equation, the residual velocity (V_r) penetration equation:

$$V_r = \frac{\sqrt{V^2 - V_{50}^2}}{1.0 + \frac{\rho A_p t}{W\cos\theta}}$$

This physics-based equation with empirical constants originates from a declassified program, called Project Thor run at the Johns Hopkins Ballistic Analysis Laboratory in 1961. The empirical constants allow for the equation's use with non-traditional impactors (e.g., fan blade fragments, disk segments, etc.).

Table 3-2. List of Parameters Used in Residual Velocity Equation

Symbol	Definition	Units
V	Debris impacting velocity	meters per second
ρ	Target density	grains per cubic meter
A_p	Debris presented area	square meters
t	Plate thickness	meters
W	Debris weight	grains
θ	Obliquity angle	degrees

Note that the presented area of the fragment, A_p , can be calculated as

$$A_p = \varpi[\lambda \cos(\delta) + \tau \sin(\delta)], \text{ (see Figure 3-7).}$$

The methodology used in UEDDAM is supported by four impact test series (encompassing 227 test shots) conducted at the Naval Air Warfare Center, China Lake [10, 14, 15, 16].

3.8 HOLE SIZE CALCULATIONS

Given an uncontained engine failure, another metric that could potentially be of interest to the user is the cumulative hole size made in a group of components by a particular debris release. This feature was incorporated to support decompression analysis in accordance with 14 CFR 25.841 with the area and locations of holes predicted by the rotor burst analysis. If a manufacturer can anticipate that a rotor disk burst event could potentially result in a decompression scenario, mitigation steps can be taken early on in the design process to strengthen or protect those areas of the fuselage subject to concern or the flight envelope can be modified accordingly as required for occupant safety. Note that for hole size calculations during decompression analysis, the historical debris model (see section 5.3.2.5) is the preferred defined threat; all fragments in a rotor stage failure case should be included in the analysis to get a realistic hole size.

To this end, UEDDAM can generate cumulative hole size statistics for specific component groupings if requested. UEDDAM accomplishes this task by modifying the COVART6 JTYPE input file to include a list of components for which hole size statistics are desired. These components are passed to COVART6, where hole size calculations are performed based on penetration calculations. After exiting COVART6, UEDDAM accumulates the output hole sizes based on the component groupings originally specified and writes the appropriate values to the HOLESIZE FILE.

4 ANALYSIS OBJECTIVE AND SYSTEM FLIGHT PHASE CRITICALITY ANALYSIS

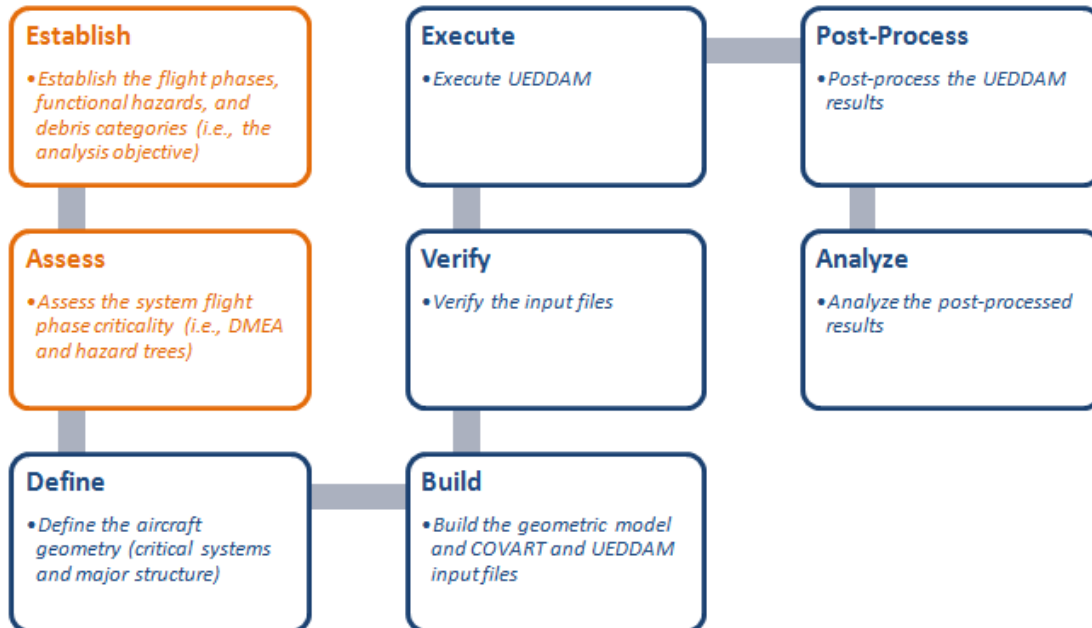


Figure 4-1. Current Location in the UEDDAM Analysis Process: Analysis Objective and Criticality Analysis.

4.1 ESTABLISH OBJECTIVE

The first step in performing a UEDDAM analysis of a particular aircraft is to establish the analysis objective, which will vary depending on the goal of the assessment. This consists not only of determining the particular flight phase(s) and functional hazard(s) of interest, but also establishing the necessary debris characteristics under consideration.

UEDDAM inputs allow for a wide variety of possible assessments. The one most commonly employed is evaluating an aircraft for a single functional hazard over multiple flight phases. As an example of this type of assessment, consider analyzing an aircraft's risk for catastrophic failure (i.e., damage and failures resulting in major structural failure of the aircraft, loss of control, or the potential for major loss of life). In this case, the AC 20-128A identifies seven[‡] flight phases (see Table 4-1), each assigned a specific phase failure distribution (Dp). Based on an average of all non-containments occurring in the US and UK, spanning the years 1966–1976, the Dp values in Table 4-1 are the industry-accepted standards for the percentage of engine failures occurring within each flight phase.

[‡] With the update to COVART6, UEDDAM is now able to accept up to 15 different flight phases in a single analysis.

Table 4-1. Flight Phases Identified in AC 20-128A

Flight Phase	Phase Identification	Phase Failure Distribution (Dp)
1	Take-off before V1	35%
2	V1 to first power reduction (+30s)	20%
3	Climb	22%
4	Cruise	14%
5	Descent	3%
6	Approach thru landing	2%
7	Landing/Reverse	4%

As stated previously, another aspect of establishing the objective of a UEDDAM analysis is to determine the types and characteristics of the debris to assess. At this phase in the analysis process, it is not necessary to completely describe all characteristics of the debris (see Section 5.3.2), but rather to generalize the debris of interest. For example, consider the goal described previously (i.e., analyzing an aircraft’s risk for catastrophic failure). With this goal in mind, it might be pertinent to assess the aircraft’s risk against an average 1/3 disk segment. Taking into account that there are multiple engine stages and each disk stage has slightly different characteristics (such as material, thickness, diameter, rotation speed, etc.), the final debris file for this assessment would consist of no more than N different debris descriptions (where N is the number of engine stages). For each engine and each engine section, and thereby each debris description, UEDDAM would average the resulting P_{HAZ} values to provide the aircraft’s risk against an average 1/3 disk segment.

4.2 PERFORM CRITICALITY ANALYSIS

The second step in performing a UEDDAM assessment of a particular aircraft is to perform a criticality analysis. This step includes developing a DMEA as well as hazard trees to describe the total aircraft system.

4.2.1 Damage Modes and Effects Analysis (DMEA)

A DMEA consists of identifying critical components, their function, and the impact of their damage-caused failures on the total aircraft system. Critical components are those components that, when defeated either individually or jointly with other components, result in the loss of an essential aircraft function. Both redundant (or multiply vulnerable [MV]) and non-redundant (or singly vulnerable [SV]) critical components should be considered in a DMEA. Generally speaking, the following damage modes are typically considered: penetration, severance, jamming, cracking, deforming, etc. Critical

components, their failures, and their impacts on the total aircraft system depend on the objective established in the first step of the UEDDAM analysis process.

As an example, consider a hydraulic pressure line in a generic twin-engine aircraft, as seen in Figure 4-2, highlighted in red. Its function is to transfer pressurized hydraulic fluid from the pumps to hydraulic actuators in the tail. If uncontained engine debris penetrates this line, there is now a leak in hydraulic system one. Assuming no isolation capability along this particular hydraulic line, a leak results in an immediate loss of pressure in and eventual loss of hydraulic system one. Suppose the objective of this assessment (as discussed in step one above) is to analyze a generic twin-engine aircraft for catastrophic damage from an average 1/3 disk segment during cruise. Since an essential function of the aircraft in cruise is flight control and the loss of hydraulics equates to the loss of flight control, the loss of hydraulic system one results in a partial loss of hydraulic capability (i.e., a loss of redundancy) and thus a diminished flight control capacity.

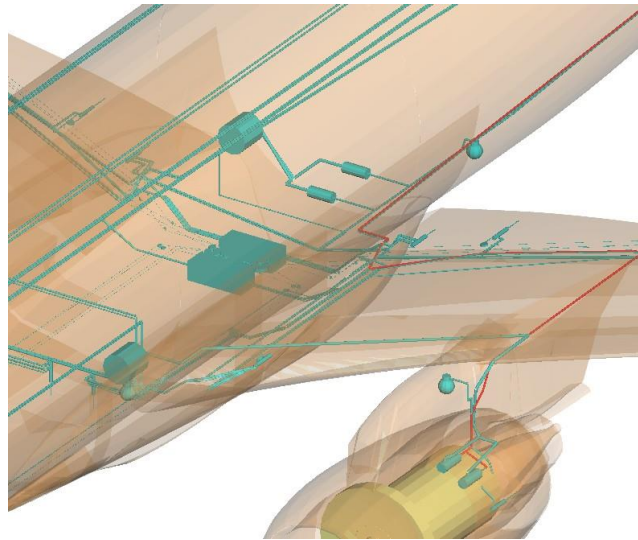


Figure 4-2. Hydraulic Pressure Line in a Generic Twin-Engine Aircraft.

Generally, the format for a DMEA is tabular; listing, component by component, the component's function, damage mode, and transient and end effects of the component's damage. For the particular hydraulic line discussed above, the DMEA entry would be as seen in Table 4-2.

Table 4-2. DMEA Entry for Generic Twin-Engine Aircraft Hydraulic Pressure Line

Component	Function	Damage Mode	Local Effects	Next Higher Effects	End Effects
Hydraulic pressure line	Delivers pressurized hydraulic fluid from the pumps to actuators in the tail; part of hydraulic system one	Penetration	Leak causes loss of pressure in hydraulic system one	Reduced authority in associated hydraulic actuators	Loss of aircraft hydraulic system redundancy and reduction in control

4.2.2 Hazard Trees

Hazard trees are visual representations of the critical components and their redundancies. An aircraft system is defeated when the tree is completely “cut.” Since critical components and their redundancies can vary between hazard levels and flight phases, there can be many different hazard trees depending on the complexity of the analysis.

For example, consider the hydraulic system of a generic twin-engine aircraft, part of which was discussed in the preceding section. The hazard tree for this system is shown (along with the geometric representation of the critical components) in Figure 4-3. By defeating the hydraulic pressure line supplying fluid to the tail of the aircraft (part of HYDSYS1 in Figure 4-3), only the left-most branch of the tree would be cut. Hydraulic systems two and three would continue to provide enough hydraulic power to support flight control of the aircraft. If, however, the uncontained engine debris not only penetrated a pressure line from hydraulic system one, but in addition one from each of the remaining systems, the entire tree would be cut, resulting in defeat of the hydraulic system and therefore catastrophic damage to the aircraft.

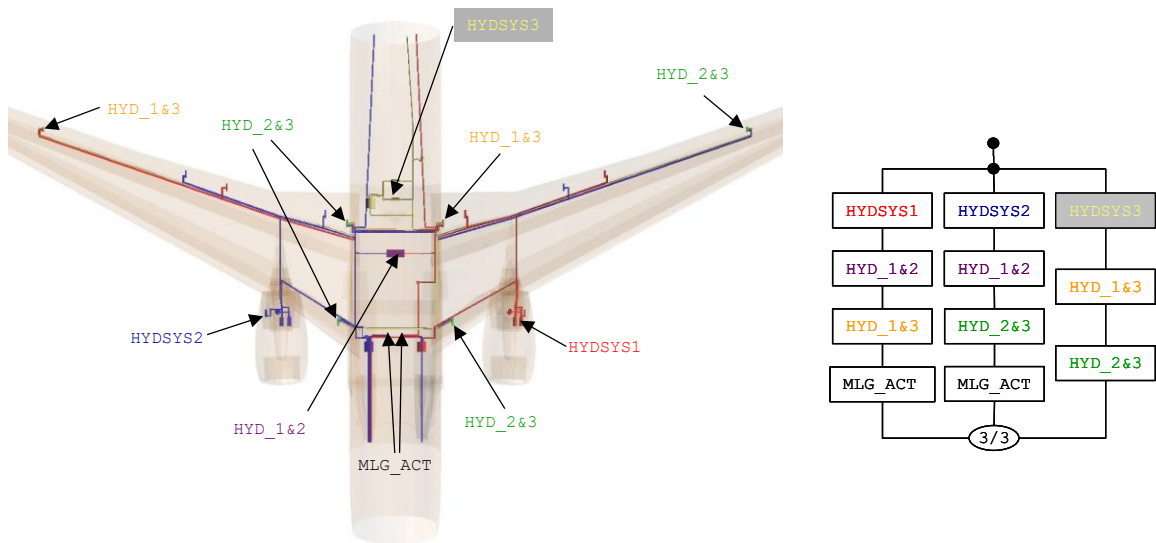


Figure 4-3. Generic Twin-Engine Aircraft Hydraulic System Hazard Tree.

5 AIRCRAFT GEOMETRY DEFINITION AND UEDDAM INPUT GENERATION

With the analysis objective established and the criticality analysis complete, the next step in the UEDDAM analysis process is to develop the various input files for UEDDAM execution. These consist of the following files (with their associated UEDDAM control file keyword designators included in parentheses): a geometric model (TARGET FILE), component properties (JTYPE FILE), probabilities of hazard given a hit (PK FILE), multiply vulnerable groupings (MV FILE or LOGIC FILE), debris characterizations (DEBRIS FILE), near-field components (NEARFIELD FILE), auto-fail components (AUTOFAIL FILE), material type definitions (MATERIAL FILE), and hole size groupings (HOLEGRP FILE).

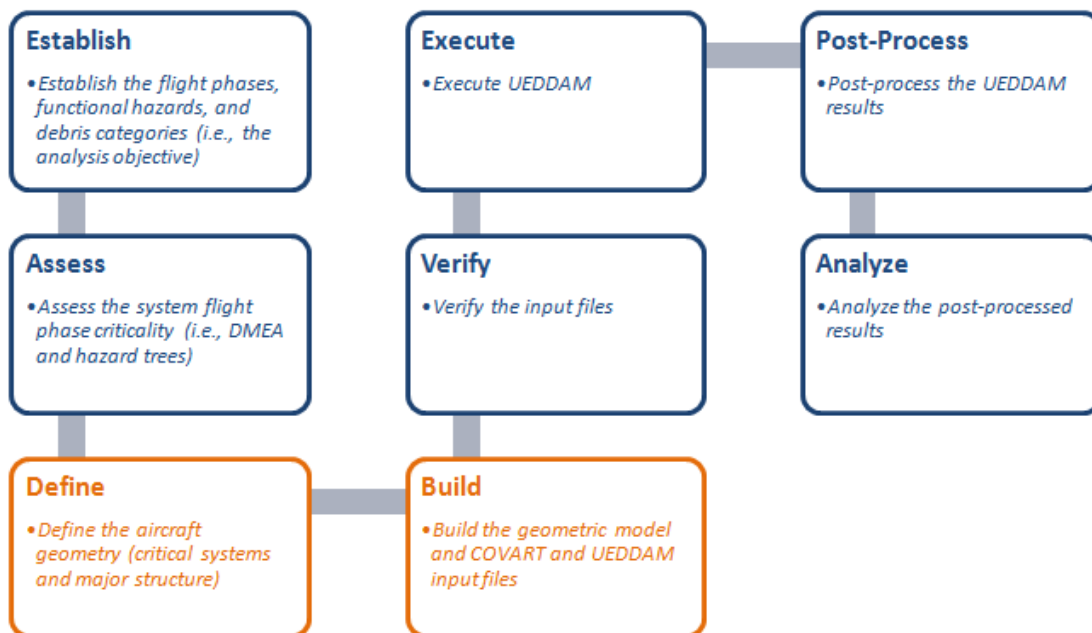


Figure 5-1. Current Location in the UEDDAM Analysis Process: Aircraft Geometry Definition and UEDDAM Input Generation.

5.1 GEOMETRIC DESCRIPTION (TARGET FILE)

The geometric model created for a UEDDAM analysis is a three-dimensional representation of the air vehicle. The aircraft geometry definition should include not only all critical components within the rotor disk debris zone (as identified in the DMEA described in Section 4.2.1), but all shielding and air vehicle surface components as well. In the case of the generic twin-engine aircraft, for example, this means in addition to all flight control, propulsion, fuel, and structural elements identified in the DMEA, the fuselage and wing skin, engine nacelles and pylons, and floor structure have to be modeled as well.

5.1.1 FASTGEN Overview

The particular geometry format used by UEDDAM is FASTGEN, a format structured around NASTRAN. Detailed descriptions of all the necessary inputs for FASTGEN can be found in Appendix B. The general structure of the geometric description is shown in Figure 5-2.

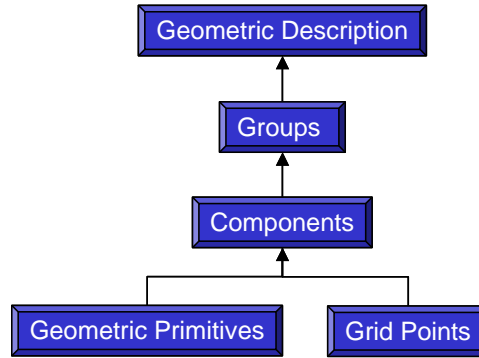


Figure 5-2. FASTGEN Geometric Description Structure.

Building a model begins by defining components (e.g., hydraulic pump, engine fan blade, trailing edge spar, etc.). Components are given an identification number between 1 and 999 and a mode (either plate mode, represented by a 1, or volume mode, represented by a 2). They are composed of numbered geometric primitives (or elements), which define the surfaces of complex geometric shapes (see Table 5-1). The primitives, in turn, are defined by numbered grid points, which locate points in three-dimensional space. The FASTGEN coordinate system is illustrated in Figure 5-3.

Table 5-1. Available FASTGEN Primitives

FASTGEN Keyword	Description of Primitive
CTRI	Triangular facet
CQUAD	Quadrilateral facet
CLINE	Line (rod)
<i>CCONE1</i>	<i>Thin-wall cone/cylinder</i>
CCONE2	Thick-wall cone/cylinder
<i>CCONE3</i>	<i>Compound thick-wall cone/cylinder</i>
CELBOW	Bent cylinder/cone
CHEX1	Hollow hexahedron (box)
CHEX2	Solid hexahedron (box)
<i>CBAR</i>	<i>Reinforcing member approximation</i>
CSPHERE	Sphere

Italicized blue text indicates a primitive that users are discouraged from employing as it is likely that it will be phased out of future versions of FASTGEN.

As an example of the construction of a component, consider a generic hydraulic pump. When simplified, this pump can be represented by a single solid cylinder (or cone), assigned to be in volume mode and given the ID number 402. The pump is then described by a CCONE2, which requires the definition of two grid points (keyword GRID) and inner and outer radii at each point. Since the pump is being treated as a solid, the inner radii at each point would be zero.

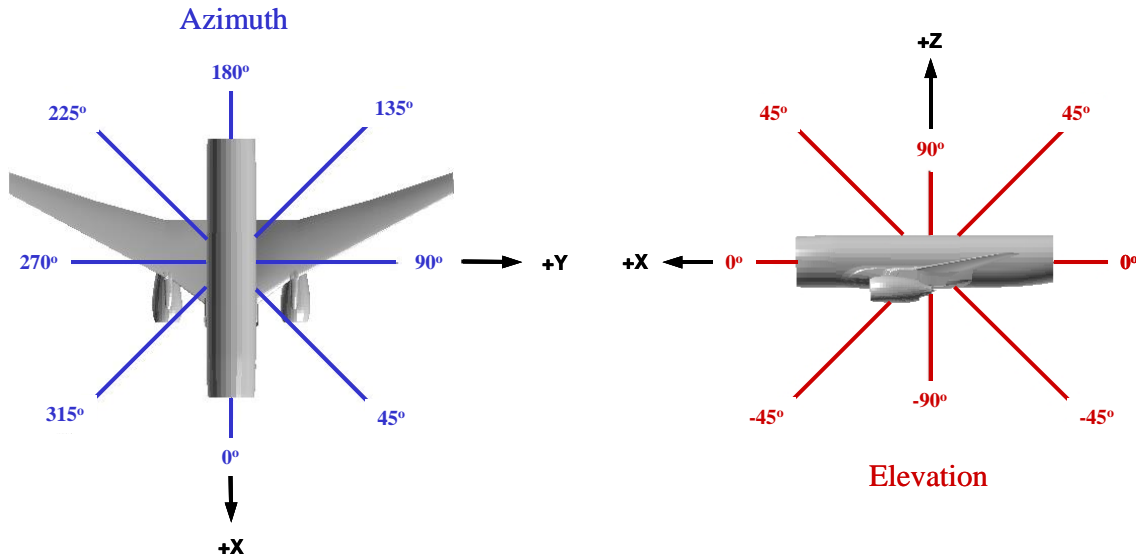


Figure 5-3. FASTGEN Coordinate System.

Components are then combined into global regions, called groups, and are assigned a group identification number. Common group identification numbers (see Table 5-2) are based on aircraft subsystems and are restricted by FASTGEN to be between 0 and 49. Based on these assignments, the aforementioned hydraulic pump would be given a group number of 3. (Note that the component is then typically referred to as component number 3402). Figure 5-4 illustrates what the FASTGEN CBULK file for a generic hydraulic pump would look like (note that the ‘.’ represents a space).

Table 5-2. Common Group Identification Numbers

Legacy Groupings	
Group ID	Description
0	Skin
1	Power Plant
2	Crew
3	Flight Control System
4	Fuel System
5	Ammunition
6	Armament

Table 5-2. Common Group Identification Numbers (cont.)

7	Structural Members		
8	Electrical System/Avionics		
9	Miscellaneous		
Detailed Groupings			
Group ID	Description	Group ID	Description
0	Skin - Forward Fuselage	27	Fuel - Center Fuselage Tank
1	Skin - Forward Intermediate Fuselage	28	Fuel - Aft Intermediate Fuselage Tank
2	Skin - Center Fuselage	29	Fuel - Aft Fuselage Tank
3	Skin - Aft Intermediate Fuselage	30	Fuel - Left Wing Tank
4	Skin - Aft Fuselage	31	Fuel - Right Wing Tank
5	Skin - Left Wing	32	Fuel Lines
6	Skin - Right Wing	33	Ammunition
7	Skin - Horizontal Tail	34	Armament
8	Skin - Vertical Tail	35	Structure - Forward Fuselage
9-10	Engine #1	36	Structure - Forward Intermediate Fuselage
11-12	Engine #2	37	Structure - Center Fuselage
13-14	Engine #3	38	Structure - Aft Intermediate Fuselage
15-16	Engine #4	39	Structure - Aft Fuselage
17	Crew	40	Structure - Left Wing
18	Flight Control - Pitch	41	Structure - Right Wing
19	Flight Control - Yaw	42	Structure - Horizontal Tail
20	Flight Control - Roll	43	Structure Vertical Tail
21	Hydraulic System #1	44	Electrical Boxes
22	Hydraulic System #2	45	Electrical Lines
23	Hydraulic System #3	46	Electrical PAO Lines
24	Hydraulic System #4	47	Electrical Antenna/Radar/ECM
25	Fuel - Forward Tank	48-49	Miscellaneous
26	Fuel - Forward Intermediate Fuselage Tank		

```

$NAME .....3 .....402 .....Gen. Hyd. Pump
SECTION .....3 .....402 .....2 .....1
GRID .....1 .....0.0 .....0.0 .....0.0
GRID .....2 .....6.0 .....0.0 .....0.0
CCONE2 .....1 .....1 .....1 .....2 .....2.50C1
C1 .....2.50 .....0.0 .....0.0

```

Figure 5-4. FASTGEN CBULK File of a Generic Hydraulic Pump.

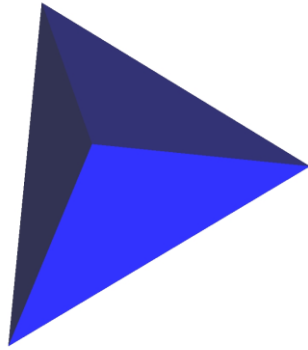
The specific format for each entry in the geometry file can be found in Appendix B, which contains excerpts from the FASTGEN 6.1 Target Description Document [6].

5.1.2 Conversion to FASTGEN from Other Geometry Formats

Typically, airframe manufacturers have a geometric model either currently in development or already developed in a particular geometric modeling package. Most companies performing work for DoD have proprietary codes for generating FASTGEN geometric files from these other packages. At the request of non-DoD aircraft companies, two options are included with UEDDAM for the generation of FASTGEN from CAD: a stereolithography (STL) converter and a NASTRAN (Patran specifically) converter.

5.1.2.1 Stereolithography Conversion

It is possible to convert a file in Stereolithography (STL) format (which can be created by most CAD packages) into FASTGEN format in a fairly straightforward manner. Stereolithography format is an ASCII or binary listing of triangular surfaces describing a computer generated solid model. Since one of the available primitives in FASTGEN is a triangle (denoted by the keyword CTRI), a component exported in STL format can be directly converted into FASTGEN format. As an example, consider the tetrahedron seen and described (in STL format) in Figure 5-5.



```

solid
facet normal 0.00 0.00 -1.00
  outer loop
    vertex 0.000 0.000 0.000
    vertex -10.000 5.000 0.000
    vertex -10.000 -5.000 0.000
  endloop
endfacet
facet normal 1.00 1.00 1.00
  outer loop
    vertex 0.000 0.000 0.000
    vertex -10.000 5.000 0.000
    vertex -5.000 0.000 10.000
  endloop
endfacet
facet normal -1.00 0.00 1.00
  outer loop
    vertex -10.000 5.000 0.000
    vertex -10.000 -5.000 0.000
    vertex -5.000 0.000 10.000
  endloop
endfacet
facet normal 1.00 -1.00 1.00
  outer loop
    vertex 0.000 0.000 0.000
    vertex -10.000 -5.000 0.000
    vertex -5.000 0.000 10.000
  endloop
endfacet

```

Figure 5-5. Solid Tetrahedron Described in Stereolithography Format.

First, it is important to note that no component information (e.g., component identification number, group number, etc.) appears in the above STL description. In order to generate the FASTGEN version of this tetrahedron, the component information must be assigned (e.g., component number 9612 in volume mode), the four grid points need to be extracted from the twelve vertices listed in the STL description, and these grid points should then be used to create the four triangles identified in Figure 5-6. The completed FASTGEN version of the tetrahedron is illustrated in Figure 5-7.

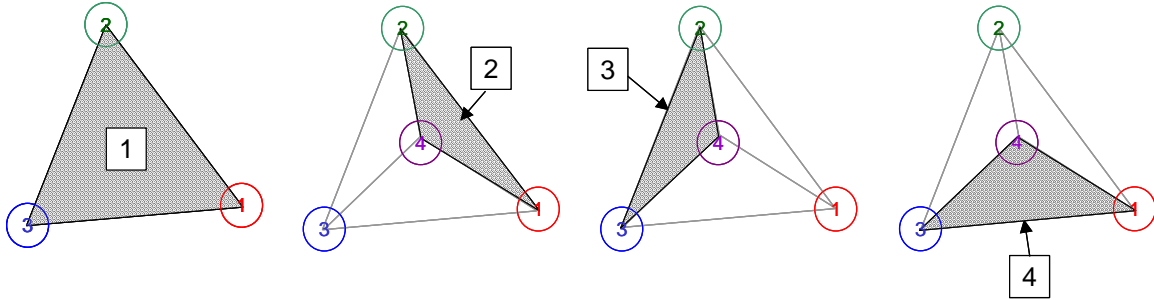
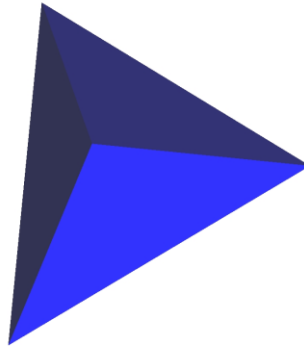


Figure 5-6. Triangles and Grid Points Used to Describe a Tetrahedron.



SECTION	9	612	2				
GRID	1		0.000	0.000	0.000		
GRID	2		-10.000	5.000	0.000		
GRID	3		-10.000	-5.000	0.000		
GRID	4		-5.000	0.000	10.000		
CTRI	1	1	1	2	3	0.0	2
CTRI	2	1	1	2	4	0.0	2
CTRI	3	1	2	3	4	0.0	2
CTRI	4	1	1	3	4	0.0	2
ENDDATA							

Figure 5-7. Solid Tetrahedron Described in FASTGEN Format.

Included in the “tools” directory of the PC UEDDAM installation is an executable (*stl2fast4.exe*) that was written to perform the above described process. The converter accepts an STL file in either binary or ASCII format (containing one or more than one component) and outputs a FASTGEN geometry file according to user-specified component properties (e.g., component number, material type, normal thickness, etc.). In those cases where an ASCII-formatted STL input file contains more than one component, the converter will apply the user-specified component number to the first component in the file and number each subsequent component sequentially. For specific usage instructions for the STL-to-FASTGEN converter, see the README file in the “tools” directory.

5.1.2.2 Patran Conversion

FASTGEN is structured around the NASTRAN format. The main difference between the two stems from the fact that the NASTRAN input format is both highly structured and flexible, a difficult feature to support. For example, NASTRAN requires each structural element to be connected to another structural element, while FASTGEN does not permit connectivity between groups and/or components. Because of the similarity in element definitions between the two formats (see Table 5-3), however, and the fact that most commercial airframers have an FEM of their aircraft to support structural analyses, the decision to develop a utility to convert NASTRAN-formatted components to FASTGEN-formatted components was an obvious one.

Table 5-3. FASTGEN to NASTRAN Element Cross-Reference.

FASTGEN Records	NASTRAN Records
GRID	GRID
<i>CBAR</i>	CBAR
CLINE	CROD, CBAR, CBEAM
CTRI	CTRIA1, CTRIA2, CTRIA4
CQUAD	CQUAD1, CQUAD2, CQUAD4
<i>CCONE1</i>	None
CCONE2	None
<i>CCONE3</i>	None
CSPHERE	None
CHEX1	CHEXA1
CHEX2	CHEXA1

Such a converter was developed by MSC Software® for the FAA based on Patran release version 2007r1a (Build 15.0.036) in 2008 and is included with the UEDDAM installation. In the “tools” directory of the PC installation is a directory called “Patran_Preference”. Within this folder are five files: *fastgen.plb*, *fastgen_utils.plb*, *fastgen_template.db*, *p3epilog.pcl*, and *Patran-to-FASTGEN_Converter.pdf*. The first four files are the actual utility, as delivered by MSC Software, and contain the FASTGEN analysis and utilities codes, a template including the FASTGEN analysis preference, and an example set of start-up commands. The final file is a user’s manual (to include installation instructions) for the tool.

The Patran conversion tool is actually a set of analysis libraries that can be accessed directly within Patran. Once installed, the user can load an existing model, map material and property set data to FASTGEN-recognizable data, manage FASTGEN SECTION definitions, and export SECTION data in a complete FASTGEN geometric model. One of the advantages of the Patran converter over the STL converter is that material data is kept with the component, reducing the manual data entry required to complete the model. For specific usage instructions for the Patran converter, see the user’s manual, *Patran-to-FASTGEN_Converter.pdf*, in the “tools/Patran_Preference” directory.

5.1.2.3 Model Validation and Completion

To use either converter, each component that is either a critical, shielding, or air vehicle surface component needs to be identified and manually edited. In the case of the STL converter, for example, the level of detail of components should be reduced as much as possible within the original CAD package. With the Patran converter, specific element groupings into individual components must be made within Patran prior to export.

The FASTGEN files generated using either conversion process must be assembled with other components/models, verified (in terms of grouping and properties), and visually inspected for errors. Specifically, as the STL converter results in a separate FASTGEN geometry file for each STL file input, unless all components have been saved out into a single ASCII STL file, the first step that must typically be performed after conversion is to concatenate all the individual files together. This step can be skipped when using the Patran converter if the user exports multiple components simultaneously or selects to append components to an existing FASTGEN geometric model. Once a single model has been developed, there are several options available to verify its “correctness.” One is to run COVART6 in legacy FASTGEN mode outside of UEDDAM. By shotlining the geometry over a fine grid and through multiple azimuth/elevation angle combinations, the model can be checked for errors (e.g., interferences, warped quads, degenerate triangles, etc.). The FASTGEN model can also be visually inspected, either through direct inspection of the text file or graphically through the use of a FASTGEN geometry viewer. The UEDDAM Visualizer (the executable, *Uvis.bat*, is located in the “bin” directory of the PC UEDDAM installation) can be used to assist in this process as the visualizer was developed not only as a tool to assist in analyzing UEDDAM outputs, but also as a FASTGEN aircraft geometric model development utility. For more information on how to use the UEDDAM Visualizer to inspect a geometric model, see the visualizer user’s manual, located in the “doc” directory of the installation.

5.1.3 Guidelines/Techniques and Code Limitations (Lessons Learned)

When developing a geometric model, either from scratch or through conversion, it is important to observe the following guidelines. Not doing so could result in errors in the analysis and problems in code operation.

- Do not use blank lines or tabs in the text file, as they are not recognized by FASTGEN; however, spaces are permitted and should be used to align the data to the appropriate column in the file (see Appendix B for specific requirements).
- Do not repeat grid point or primitive identification numbers within a single component.
- Although available in FASTGEN, it is recommended that CCONE1s and CCONE3s not be used in geometric model development as they are being phased out in subsequent versions.
- Do not repeat grid point identification numbers within a single primitive.
- Avoid component interferences, unless necessary.

- Assign logical names to components and use comments (denoted by the keywords \$NAME and \$COMMENT, respectively) to provide internal documentation.
- A component in volume mode should not contain plate mode primitives (e.g., CHEX1, CLINE, etc.) and vice versa.
- Limit the level of component detail to reduce the introduction of errors.
- A single component should consist of a single SECTION record.
- Group identification numbers should be kept to between 0 and 49.
- Keep the number of grid points used in a single component to below 50,000.
- When converting a geometry, eliminate small subcomponents (e.g., nuts, bolts, wire connectors, etc.) within a CAD tool before exporting to STL.
- When converting a geometry, reduce the level of detail of components (e.g., simplify curvatures, eliminate internals, generalize shapes, etc.) within a CAD tool before exporting to STL (or use 3rd party software after exporting to STL).
- Grid point coordinates must be real numbers (i.e., they must contain decimal points).
- The first grid point identification number defined in a component should be the smallest identification number used in that component.

5.2 DESCRIBE COMPONENT AND COMPONENT REDUNDANCIES

The three files discussed in this section include the following: component properties (JTYPE FILE), probability of hazards given a hit (PK FILE), and multiply vulnerable groupings (MV FILE). Most of the essential information needed to create these files has already been gathered in previous steps of the analysis process. However, it is necessary to assemble additional descriptive information for each component (e.g., the specific materials—and percentage of each—within each component, how “easy” each component is to defeat, etc.) in order to complete these files.

The specific format for each possible entry in the JTYPE, PK, and MV files can be found in Appendix C, which contains excerpts from the COVART 6.7 User’s Manual [7].

5.2.1 Component Properties File (JTYPE FILE)

The component properties file, otherwise known as the JTYPE FILE, describes component information such as material type, density (or normal thickness), and criticality. One property that the JTYPE FILE assigns to a component number is an identification name. The name consists of no more than eight alphanumeric characters and is also used in both the MV FILE and AUTOFAIL FILE to reference the component, as necessary. As an example, consider the hydraulic pressure line of the generic twin-engine aircraft previously discussed in Sections 4.2.1 and 4.2.2. Since the pressure line is part of hydraulic system one, a natural name to associate with the line (and, in fact, all other components strictly belonging to hydraulic system one) would be HYDSYS1. Note

that it is not necessary for each component to be assigned a unique name, as the previous example illustrates. However, the component name is the lowest level that COVART6, and therefore UEDDAM, can output. In other words, if multiple components are grouped into a single name then the contribution of the individual components under that name cannot be determined.

In addition to assigning a name to each component, the JTYPE FILE also designates a density factor (or normal thickness) and a material type. A list of available predefined material types and their associated identification numbers is provided in Table 5-4. Additionally, since there are a large variety of materials in use on aircraft today that are not listed in Table 5-4, UEDDAM will accept up to 44 new material types. For more information regarding the definition of additional material types, see Section 5.3.5. The density factor/normal thickness is available to the user as a way to manipulate the thickness (for penetration purposes) of a component without modifying that component's description in the geometry file. This value can be entered either as a percentage (e.g., 65 would represent 65%) or as a nominal thickness in hundreds of an inch (e.g., -15 would represent a normal thickness of 0.15 inches, where the negative sign indicates that this value is to be treated as a thickness, rather than a percentage). The complete description of the hydraulic line discussed previously, as it would appear in the JTYPE FILE, is

COMPPHY.....3105HYDSYS1.....1.....100.....8

indicating that component 3105 is to be considered 100% Aluminum 2024 for penetration purposes (note that a ‘.’ represents a space in this illustration to delineate the format clearly).

Table 5-4. Predefined Material Types

Material Description	JTYPE FILE ID	UEDDAM Material Name
Steel (BHN = 300)	5	Steel_300
Titanium (BHN = 285)	7	Ti_285
Aluminum 2024	8	Al_2024
Composite (General)	46	Graphite
Inconel [®] 625 LCF	47	Inconel
Water	51	---
Gasoline	52	---
Lubrication	53	---
JP1 Fuel	54	---
JP4 Fuel	55	---
JP5 Fuel	56	---
JP8 Fuel	57	---

The density factor/normal thickness of a component can be manipulated in order to mimic a component's actual physical characteristics. As an example, consider an engine

control electronics box, represented in a geometric description by a single solid box (i.e., the control boards contained within are not modeled as separate components). In order to better approximate the control box's penetration characteristics without changing its geometric description, the density entry in the JTYPE FILE for this component should be modified accordingly (most likely reduced to a value less than 50%).

There are certain instances when it becomes necessary to consider multiple failure modes of a single component with different redundancy levels. This is accomplished with an alias (denoted in the JTYPE FILE by the keyword COMALIAS). The alias component is assigned a component number, name, material type, and density to distinguish it from the aliased component, but does not affect penetration. As an example of how an alias might be used, consider a component that is both redundant and non-redundant during a particular flight phase (as determined in the DMEA). The 'real' (i.e., modeled) component can be used to represent the non-redundant aspect of the component while the alias can be used to represent the redundant aspect of the same component. Another example of the use of aliases is given in Section 5.2.3.

Finally, within the JTYPE FILE, the criticality of each component is considered. This is achieved by assigning probability-of-hazard-given-hit ($P_{\text{haz/h}}$) tables to each critical component (and any aliases used) for the various flight phases being considered (as determined in the analysis objective phase of the UEDDAM analysis process). The entry is denoted by the keyword COMPPK and can be repeated up to six times for each component to represent different damage mechanisms. The $P_{\text{haz/h}}$ tables are discussed further in the next section.

5.2.2 Probability of Hazard Given Hit File (PK FILE)

The probability of hazard given hit file, referred to simply as the PK FILE, contains all probability of aircraft hazard given component hit ($P_{\text{haz/h}}$) tables for each and every critical component (and alias) referenced in the JTYPE FILE. $P_{\text{haz/h}}$ is defined as the product of the probability of component damage given a hit on that component ($P_{\text{cd/h}}$) and the probability of the aircraft reaching a certain hazard level given component damage ($P_{\text{haz/cd}}$). The PK FILE has the capability of assessing many different types of damage (including penetration, fire, holes, blast, etc.). However, for the purposes of a UEDDAM assessment, typically only one type of table is used: the impactor weight table (denoted by the keyword WGT).

The impactor weight table relates the probability of defeating the aircraft (by means of defeating a component) to certain fragment properties (i.e., mass [measured in grains] and velocity [measured in fps]). Typically, for a UEDDAM assessment, only a few $P_{\text{haz/h}}$ tables are necessary. For example, it may be possible to split all critical components into one of two categories: soft or hard, where soft means "easy" to defeat and hard means "difficult" to defeat. A soft component $P_{\text{haz/h}}$ table would say that any projectile weighing between 15 and 90000 grains (7000 grains \equiv 1 pound) and striking the component with velocities between 2 and 10000 fps (5000 fps for the larger projectile) would defeat the

component, resulting in a probability of hazard to the aircraft of 1.0. On the other extreme, a “hard” component $P_{\text{haz/h}}$ table would say that only projectiles weighing a minimum of 7050 grains would result in a probability of hazard to the aircraft of 1.0. These two $P_{\text{haz/h}}$ tables (as well as their graphical representation) are shown in Figure 5-8.

```

PKTABLE .....1WGT .....10 .....2 .....0 "Soft" .....W1-1
W1-1 .....15 .....1 .....0.0 .....2 .....1.0 .....10000 .....1.0 .....1W1-2
W1-2 .....90000 .....1 .....0.0 .....2 .....1.0 .....5000 .....1.0 .....1
PKTABLE .....2WGT .....7000 .....2 .....0 "Hard" .....W2-1
W2-1 .....7050 .....1 .....0.0 .....2 .....1.0 .....10000 .....1.0 .....1W2-2
W2-2 .....90000 .....1 .....0.0 .....2 .....1.0 .....5000 .....1.0 .....1

```

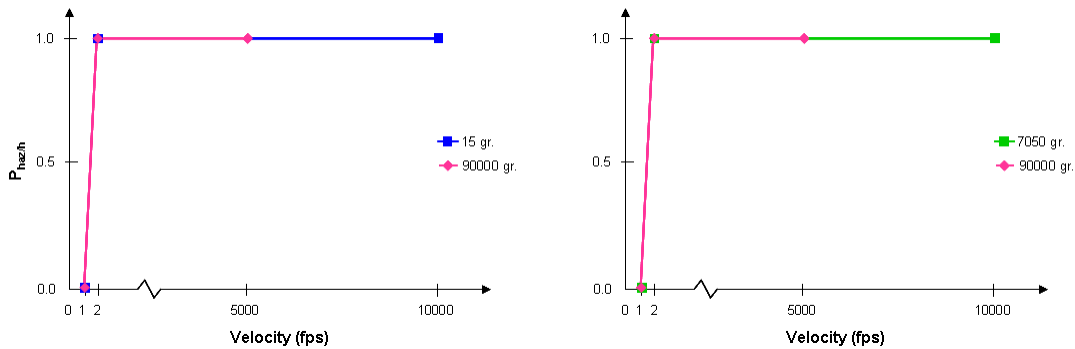


Figure 5-8. Sample $P_{\text{haz/h}}$ Tables and Graphs.

Intuitively, this appears overly simplistic. However, as the goal of the analysis is to show minimization, a conservative $P_{\text{haz/h}}$ function reduces the justification needed to show a component can survive an impact. The current certification requirements (as detailed in the AC 20-128A) depend on redundancy and therefore support this approach.

5.2.3 Multiply Vulnerable File (MV FILE)

The multiply vulnerable file (MV FILE) provides a means of input for the hazard trees created in step two of the UEDDAM analysis process. Hazard trees for various critical aircraft functions (e.g., propulsion, roll control, etc.) are entered as groups (denoted by the keyword GRPMULT) by flight phase (denoted by the keyword MVKILL). Each group, in turn, is composed of systems and potentially subsystems (denoted by the keyword SYSMULT), representing specific subsets of the hazard tree. A system may be composed of a single component (specified by its identification name from the JTYPE FILE) or a sequence of components and/or subsystems either in series or parallel. For each parallel combination, an input in the MV FILE allows the user to indicate how many of the parallel items must be defeated in order to defeat the system.

For example, consider the hazard tree for the hydraulic system of a generic twin-engine aircraft as seen in Figure 4-3 (Section 4.2.2) and suppose the particular UEDDAM

assessment objective under consideration is to determine the probability of catastrophic failure during the first three phases of flight (see Table 4-1 [Section 4.1]). The group HYD_SYS would be constructed of three parallel systems, identified as HYD_1, HYD_2, and HYD_3 (all three of which would have to be defeated in order to defeat the group), with each system being comprised of a sequence of components in series. In the MV FILE, the first few lines of the definition of the HYD_SYS group would look like the following:

```
GRPMULT ·HYD_SYS ······3 ······3 ······c1 ······
c1 ······HYD_1 ···HYD_2 ···HYD_3 ······
```

For this analysis, since loss of the entire hydraulic system is considered catastrophic during take-off, V1 to V1+30 seconds, and climb, the group HYD_SYS would have to be repeated three times within the MV FILE, once for each phase of flight.

As a more complicated example, consider the generic twin-engine aircraft discussed previously (and in more detail in Appendix D). For this analysis, suppose that the loss of 4 adjacent circumferential frames (of which there are a total of 36) is considered a catastrophic event (note that this is different from what is considered catastrophic in the analysis presented in Appendix D). The hazard tree developed in step two of the UEDDAM analysis process for the frames can be seen in Figure 5-9. If the hazard tree were input as seen, however, UEDDAM would fail to execute due to limitations within COVART. (Note that frames 4 through 33 each appear 4 separate times within this tree, and thus a simplification is required.)

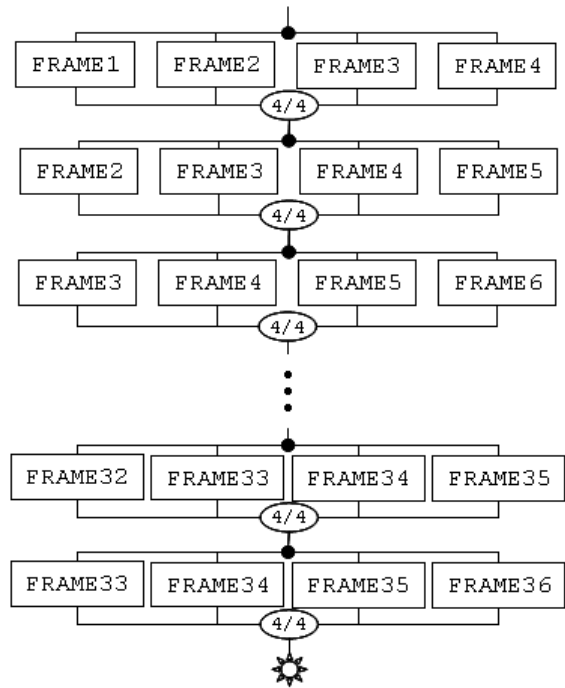


Figure 5-9. Partial Circumferential Frames Hazard Tree.

As an attempt at simplifying the above hazard tree, consider dividing the frames into sets of six. The first set would contain frames 1 thru 6, the second, 7 thru 12, etc. And, instead of assessing strictly the loss of four adjacent circumferential frames, consider the loss of any four of the six frames in a set (since the most likely occurrence will be the loss of four adjacent frames). Upon consideration, notice that failures could also occur between sets (i.e., the loss of frames 4 thru 7 would also result in catastrophic loss of the aircraft). Thus, the hazard tree needs to take these cases into account by including overlapping sets (see Figure 5-10). Therefore, the first 6 sets would include frames 1 thru 36 in sets of 6 whereas the next 5 sets would include frames 4 thru 33. A portion of this simplified hazard tree can be seen in Figure 5-11.

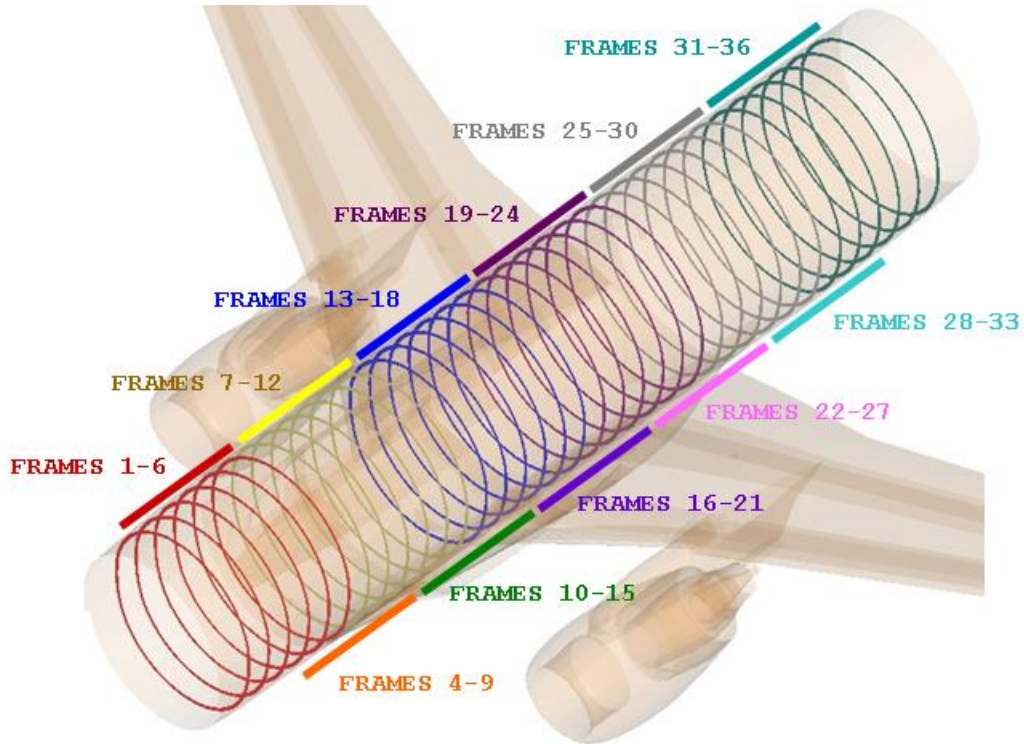


Figure 5-10. Circumferential Frames in a Generic Twin-Engine Aircraft.

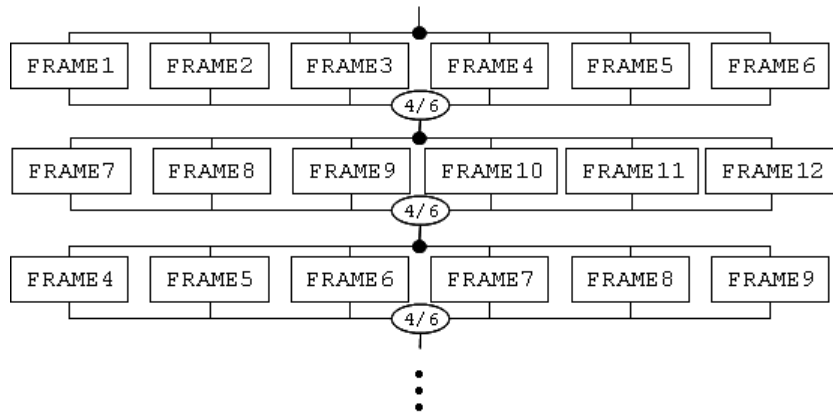


Figure 5-11. Partial Simplified Circumferential Frames Hazard Tree.

Note that frames 4 thru 33 now only appear twice in the fault tree. In order to simplify this hazard tree even further, the introduction of aliases is required. By assigning an alias in the JTYPE FILE to each of the multiply occurring frames (indicated by the name AFRAME#), each frame then only appears once in the hazard tree, as seen in the complete hazard tree in Figure 5-12.

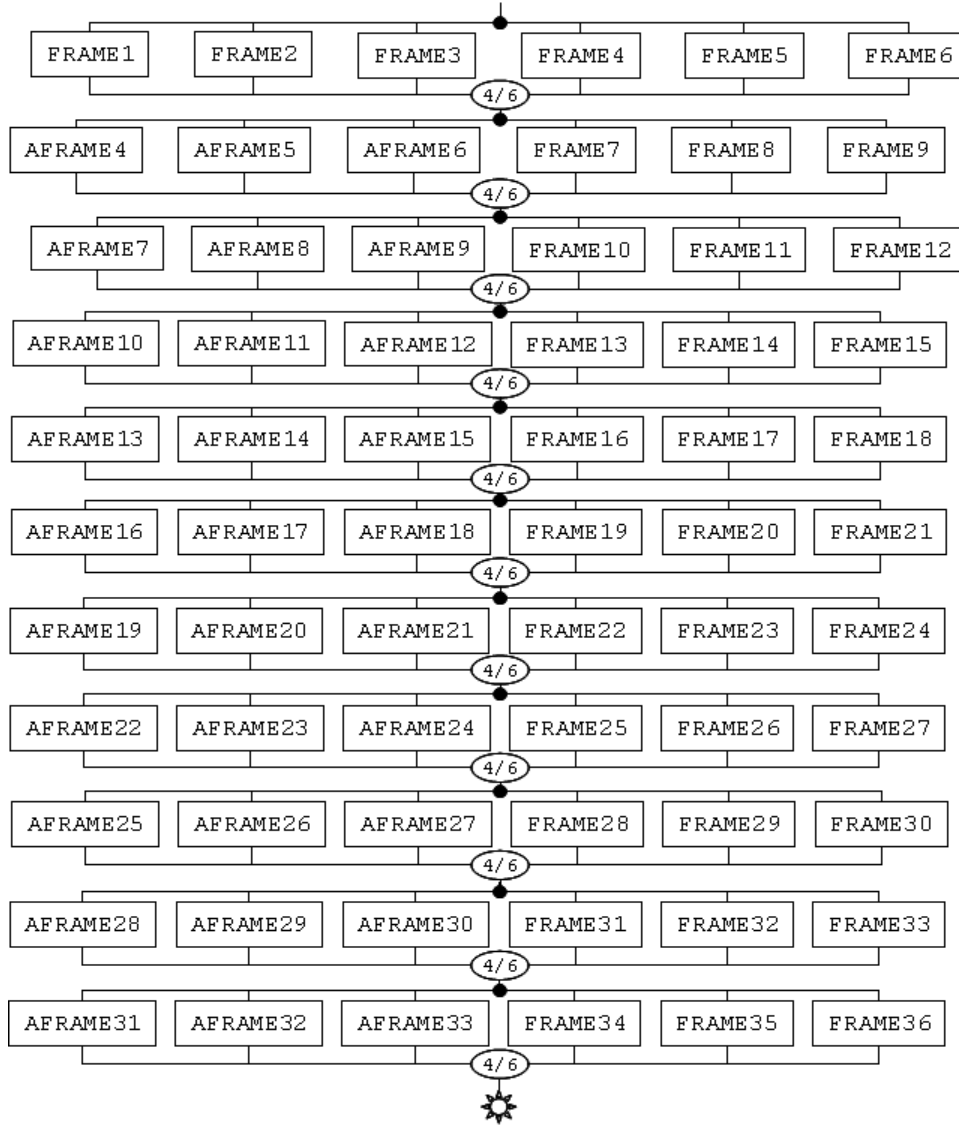


Figure 5-12. Complete Further Simplified Circumferential Frames Hazard Tree.

Note that the last step in the simplification process described above treats the components in the fault tree as if they were independent, as opposed to dependent. This assumption only results in an accurate assessment of the fault tree if the $P_{\text{haz/h}}$ values in the tables assigned to both the frames and the aliases of the frames are 0.0 or 1.0; as these values approach 0.5, the error increases. As a simple example of why this occurs, consider the more basic fault trees displayed in Figure 5-13. The tree on the left considers component

“A” a dependent component, whereas the tree on the right considers “A” an independent component (i.e., makes A’ an alias of A).

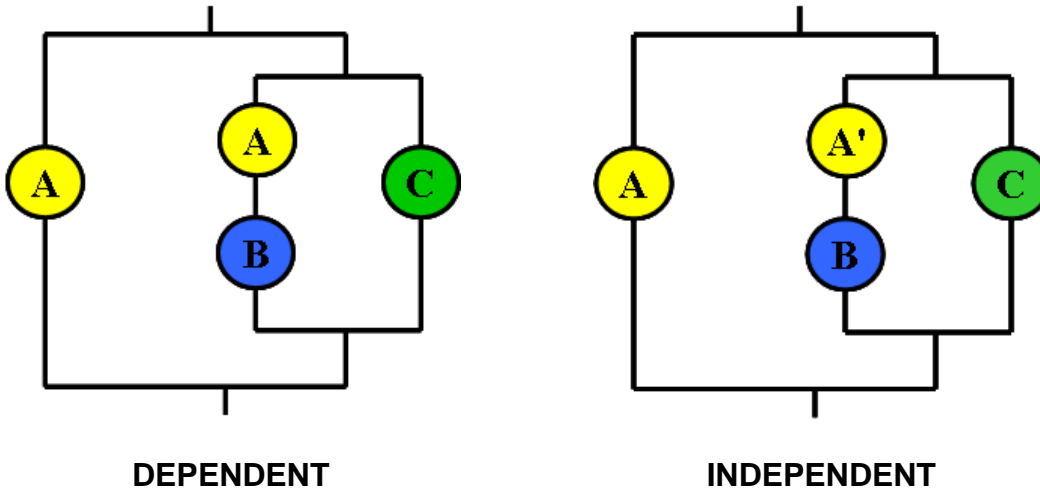


Figure 5-13. Dependent vs. Independent Fault Trees.

From basic probability theory, the analysis of the tree on the left can be written as:

$$\begin{aligned}
 P_{\text{haz_dep}} &= [P(\text{A damaged}) \text{ AND } P(\text{B damaged}) \text{ AND } P(\text{C damaged})] \\
 &\quad \text{OR } [P(\text{A damaged}) \text{ AND } P(\text{B not damaged}) \text{ AND } P(\text{C damaged})] \\
 P_{\text{haz_dep}} &= [P_{\text{haz/h_A}} \times P_{\text{haz/h_B}} \times P_{\text{haz/h_C}}] + [P_{\text{haz/h_A}} \times (1 - P_{\text{haz/h_B}}) \times P_{\text{haz/h_C}}] \\
 P_{\text{haz_dep}} &= [a \times b \times c] + [a \times (1 - b) \times c] \\
 P_{\text{haz_dep}} &= abc + ac - abc \\
 P_{\text{haz_dep}} &= ac
 \end{aligned}$$

whereas the tree on the right is analyzed as:

$$\begin{aligned}
 P_{\text{haz_ind}} &= [P(\text{A damaged}) \text{ AND } P(\text{A' damaged}) \text{ AND } P(\text{B not damaged}) \text{ AND} \\
 &\quad P(\text{C damaged})] \\
 &\quad \text{OR } [P(\text{A damaged}) \text{ AND } P(\text{A' damaged}) \text{ AND } P(\text{B damaged}) \text{ AND} \\
 &\quad P(\text{C damaged})] \\
 &\quad \text{OR } [P(\text{A damaged}) \text{ AND } P(\text{A' not damaged}) \text{ AND } P(\text{B damaged}) \\
 &\quad \text{AND } P(\text{C damaged})] \\
 P_{\text{haz_ind}} &= [P_{\text{haz/h_A}} \times P_{\text{haz/h_A'}} \times (1 - P_{\text{haz/h_B}}) \times P_{\text{haz/h_C}}] \\
 &\quad + [P_{\text{haz/h_A}} \times P_{\text{haz/h_A'}} \times P_{\text{haz/h_B}} \times P_{\text{haz/h_C}}] \\
 &\quad + [P_{\text{haz/h_A}} \times (1 - P_{\text{haz/h_A'}}) \times P_{\text{haz/h_B}} \times P_{\text{haz/h_C}}] \\
 P_{\text{haz_ind}} &= [a \times a' \times (1 - b) \times c] + [a \times a' \times b \times c] + [a \times (1 - a') \times b \times c] \\
 P_{\text{haz_ind}} &= aa'c - aa'bc + aa'bc + abc - aa'bc \\
 P_{\text{haz_ind}} &= aa'c + abc - aa'bc
 \end{aligned}$$

Note that $P_{\text{haz_dep}} = P_{\text{haz_ind}}$ if $a (= a') = b = c = 0$ or 1 , but that $P_{\text{haz_dep}} = 0.25$ and $P_{\text{haz_ind}} = 0.1875$ if $a (= a') = b = c = 0.5$.

5.2.3.1 Multiple Adjacent Component Tool

As the process described in the previous section can be somewhat cumbersome, a utility has been generated to automate the methodology used to model multiple adjacent aircraft structural components and generate an MV FILE. The utility, *mac.exe*, is provided with the UEDDAM installation and is located in the “tools” directory. To run the utility, double-click on the executable. Figure 5-14 illustrates the multiple adjacent component (MAC) utility as it initially appears.

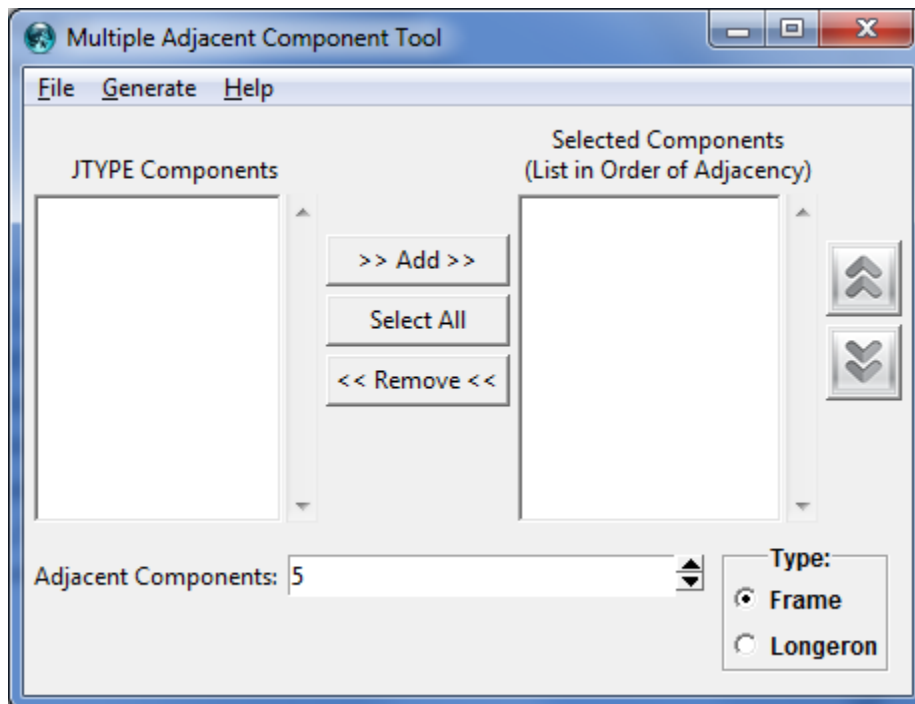


Figure 5-14. Multiple Adjacent Component Utility.

To use the MAC tool, open an existing JTYPE FILE using the File→Open selection on the menu bar. The tool loads the component identification numbers included in the JTYPE FILE to the leftmost selection box. Select the component identification numbers of the components of interest from this box and click on the >>Add>> button (in the generic twin example discussed previously, 36 component identification numbers would be selected). These components then appear in the rightmost selection box. One thing of importance to note is that the selected components must be listed in order of adjacency for the tool to work properly. The up and down arrows on the rightmost side of the tool display can be used to reorder components once they have been selected. Next, enter the following information: the number of adjacent components required for failure (e.g., 4) and the type of component redundancies under consideration (i.e., frame or longeron,

where type longeron implies that the components wrap around, such that the last component and the first component are considered adjacent components).

To execute the tool, choose the Generate→Run selection on the menu bar. Once complete, the MAC tool will open a dialog box notifying the user that it has created two new files: an updated JTYPE FILE containing the COMPPHY, COMPPK, and COMALIAS records for the new alias components and an MV FILE containing the GRPMULT generated according to the user inputs. Note that the MAC tool automatically assigns alias components to unused component identification numbers in the JTYPE FILE and prefixes the beginning of an alias component name with an 'A' to distinguish it from the real component.

As an example of the usability of the MAC tool, let us return to considering the loss of 4 adjacent circumferential frames from a generic twin-engine aircraft. (Recall that there were a total of 36 frames modeled in the aircraft and that the loss of 4 adjacent frames was considered to be a catastrophic event.) Rather than walk through the steps identified in Figures 5-9, 5-11, and 5-12 at the beginning of this section to create a fault tree and subsequently, an MV FILE, we can simply open the MAC tool, load the JTYPE FILE for the generic twin-engine aircraft (see sample case 6 distributed with UEDDAM), select the component ID numbers of the 36 circumferential frames (ID numbers 7501-7536), change the number of adjacent components to 4, and select the Frame type (see Figure 5-15).

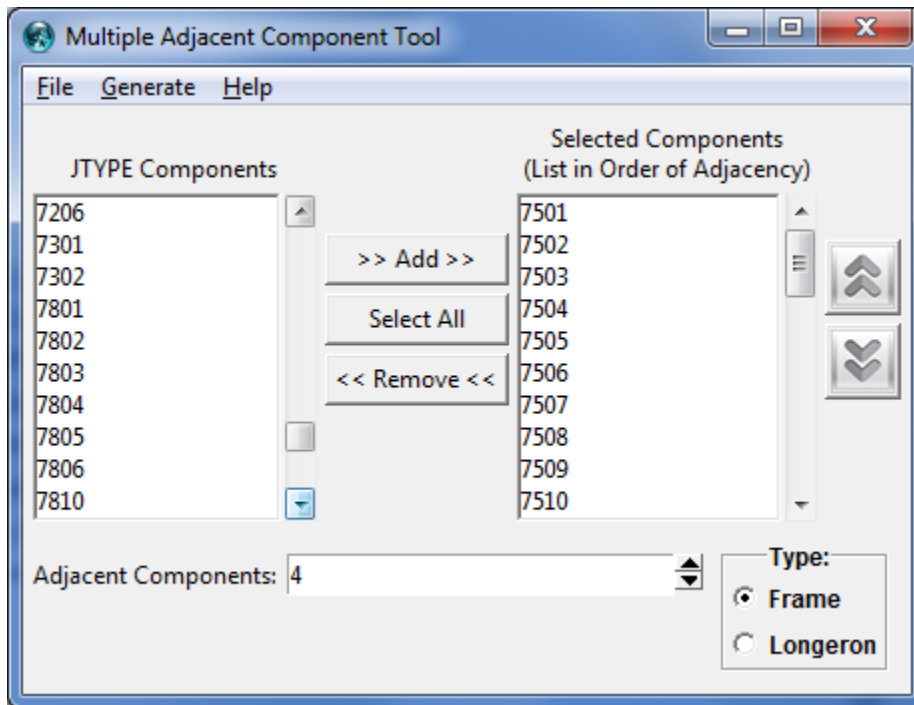


Figure 5-15. Example MAC Tool Usage.

To execute the tool, choose Generate→Run from the menu bar. Upon completion, the MAC tool generates a window similar to that displayed in Figure 5-16, letting the user know that it has successfully completed and created two new files (in this case, mv1 and jtype1) in the same directory as the original JTYPE FILE.

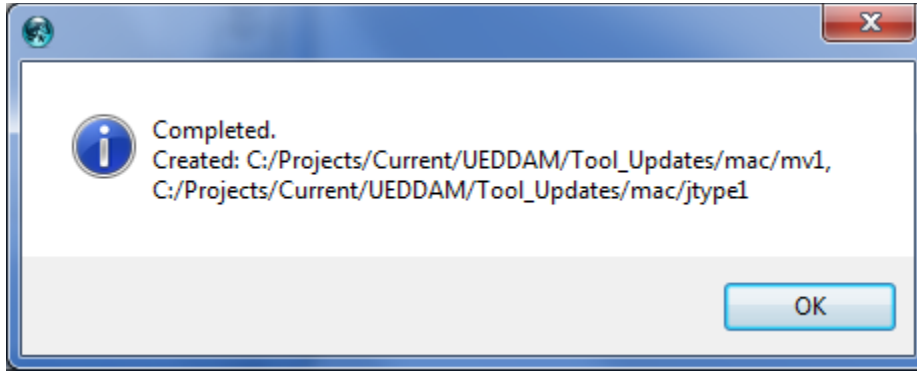


Figure 5-16. Completion Window Generated by Successful Execution of MAC Tool.

Figure 5-17 contains an image of the MV FILE generated by the MAC tool. The user could take the fault tree from this file and integrate it into an existing MV FILE or use this file as a basis from which to generate a new MV FILE. (Note that the contents of this file are exactly what is represented pictorially in Figure 5-12.)

```

MVKILL          1
GRPMULT FRAMES          1          11
FRAMES FRM_1  FRM_2  FRM_3  FRM_4  FRM_5  FRM_6  FRM_7  FRM_8  FRAMES
FRAMES FRM_9  FRM_10 FRM_11
SYSMULT FRM_1          4          6          FRM_1
FRM_1 FRAME1 FRAME2 FRAME3 FRAME4 FRAME5 FRAME6
SYSMULT FRM_2          4          6          FRM_2
FRM_2 AFRAME4 AFRAME5 AFRAME6 FRAME7 FRAME8 FRAME9
SYSMULT FRM_3          4          6          FRM_3
FRM_3 AFRAME7 AFRAME8 AFRAME9 FRAME10 FRAME11 FRAME12
SYSMULT FRM_4          4          6          FRM_4
FRM_4 AFRAME10AFRAME11AFRAME12FRAME13 FRAME14 FRAME15
SYSMULT FRM_5          4          6          FRM_5
FRM_5 AFRAME13AFRAME14AFRAME15FRAME16 FRAME17 FRAME18
SYSMULT FRM_6          4          6          FRM_6
FRM_6 AFRAME16AFRAME17AFRAME18FRAME19 FRAME20 FRAME21
SYSMULT FRM_7          4          6          FRM_7
FRM_7 AFRAME19AFRAME20AFRAME21FRAME22 FRAME23 FRAME24
SYSMULT FRM_8          4          6          FRM_8
FRM_8 AFRAME22AFRAME23AFRAME24FRAME25 FRAME26 FRAME27
SYSMULT FRM_9          4          6          FRM_9
FRM_9 AFRAME25AFRAME26AFRAME27FRAME28 FRAME29 FRAME30
SYSMULT FRM_10         4          6          FRM_10
FRM_10 AFRAME28AFRAME29AFRAME30FRAME31 FRAME32 FRAME33
SYSMULT FRM_11         4          6          FRM_11
FRM_11 AFRAME31AFRAME32AFRAME33FRAME34 FRAME35 FRAME36
ENDNAME
ENDGROUP

```

Figure 5-17. MV FILE Generated by MAC Tool.

Figure 5-18 compares the original JTYPE FILE (left) with the new JTYPE FILE (right) created by the MAC tool. As is evident by comparing these two files, the MAC tool has inserted COMPPHY and COMALIAS records into the original JTYPE FILE for the

additional components needed. Note that as the component ID numbers 1–30 were not used in the original JTYPE FILE, the MAC tool assigned these numbers to the alias components AFRAME4, AFRAME5, ..., AFRAME33.

COMPPHY	7836NONCRIT	1	100	8	0
COMPPHY	7837NONCRIT	1	100	8	0
COMPPHY	7838NONCRIT	1	100	8	0
COMPPHY	7839NONCRIT	1	100	8	0
COMPPHY	7840NONCRIT	1	100	8	0
COMPPHY	7841NONCRIT	1	100	8	0
COMPPHY	7842NONCRIT	1	100	8	0
COMPPHY	7843NONCRIT	1	100	8	0
COMPPHY	7901NONCRIT	1	100	8	0
COMPPHY	7501FRAME1	1	100	8	0
COMPPHY	7502FRAME2	1	100	8	0
COMPPHY	7503FRAME3	1	100	8	0
COMPPHY	7504FRAME4	1	100	8	0
COMPPHY	7505FRAME5	1	100	8	0
COMPPHY	7506FRAME6	1	100	8	0
COMPPHY	7507FRAME7	1	100	8	0
COMPPHY	7508FRAME8	1	100	8	0
COMPPHY	7509FRAME9	1	100	8	0
COMPPHY	7510FRAME10	1	100	8	0
COMPPHY	7511FRAME11	1	100	8	0
COMPPHY	7512FRAME12	1	100	8	0
COMPPHY	7513FRAME13	1	100	8	0
COMPPHY	7514FRAME14	1	100	8	0
COMPPHY	7515FRAME15	1	100	8	0
COMPPHY	7516FRAME16	1	100	8	0
COMPPHY	7517FRAME17	1	100	8	0
COMPPHY	7518FRAME18	1	100	8	0
COMPPHY	7519FRAME19	1	100	8	0
COMPPHY	7520FRAME20	1	100	8	0
COMPPHY	7521FRAME21	1	100	8	0

COMPPHY	7836NONCRIT	1	100	8	0
COMPPHY	7837NONCRIT	1	100	8	0
COMPPHY	7838NONCRIT	1	100	8	0
COMPPHY	7839NONCRIT	1	100	8	0
COMPPHY	7840NONCRIT	1	100	8	0
COMPPHY	7841NONCRIT	1	100	8	0
COMPPHY	7842NONCRIT	1	100	8	0
COMPPHY	7843NONCRIT	1	100	8	0
COMPPHY	7901NONCRIT	1	100	8	0
COMPPHY	7501FRAME1	1	100	8	0
COMPPHY	7502FRAME2	1	100	8	0
COMPPHY	7503FRAME3	1	100	8	0
COMPPHY	7504FRAME4	1	100	8	0
COMPPHY	1AFRAME4	1	-1	51	0
COMALIAS	7504	1			
COMPPHY	7505FRAME5	1	100	8	0
COMPPHY	2AFRAME5	1	-1	51	0
COMALIAS	7505	2			
COMPPHY	7506FRAME6	1	100	8	0
COMPPHY	3AFRAME6	1	-1	51	0
COMALIAS	7506	3			
COMPPHY	7507FRAME7	1	100	8	0
COMPPHY	4AFRAME7	1	-1	51	0
COMALIAS	7507	4			
COMPPHY	7508FRAME8	1	100	8	0
COMPPHY	5AFRAME8	1	-1	51	0
COMALIAS	7508	5			
COMPPHY	7509FRAME9	1	100	8	0
COMPPHY	6AFRAME9	1	-1	51	0
COMALIAS	7509	6			

Figure 5-18. Comparison of Original JTYPE FILE (Left) to that Generated by the MAC Tool (Right).

5.2.3.2 Weighting Factors

A valuable feature in the MV FILE is the ability to apply weighting factors to sets of system failures within an MV group. This is useful if different combinations of failures have different probabilities of outcome. For example, consider the situation of a three engine aircraft illustrated in Figure 5-19.

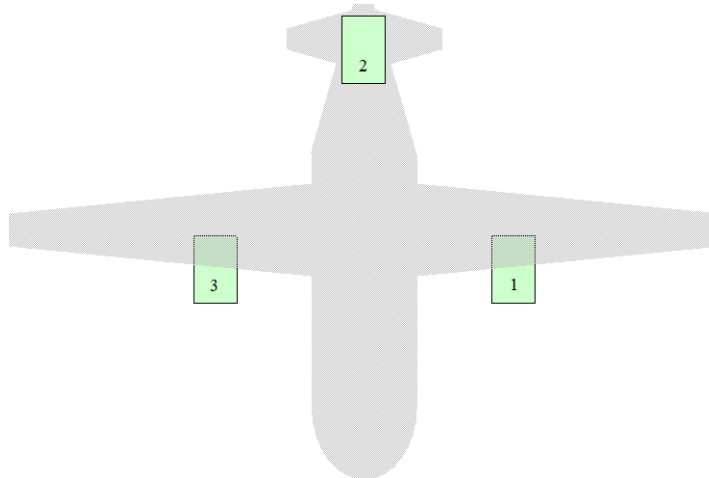


Figure 5-19. Generic Three-Engine Aircraft.

For the flight condition of interest, a loss of the number two engine is recoverable, a loss of both engines one and three is recoverable 95% of the time (controlled descent for

landing), a loss of all three engines is recoverable 60% of the time (engine out landing), and the loss of either engine one or engine three alone, or in combination with engine two, is recoverable only 20% of the time due to the yaw induced by the differential thrust. Table 5-5 summarizes the combinations of failures and their resultant probability of catastrophic outcome (= 1 - probability of recovery).

Table 5-5. Engine Failure Matrix

Engine #1 State	Engine #2 State	Engine #3 State	Probability of Catastrophic Outcome (P _{HAZ})
Functional	Functional	Functional	0.00
Failed	Functional	Functional	0.80
Functional	Failed	Functional	0.00
Failed	Failed	Functional	0.80
Functional	Functional	Failed	0.80
Failed	Functional	Failed	0.05
Functional	Failed	Failed	0.80
Failed	Failed	Failed	0.40

COVART6 allows for consideration of these different combinations of failures through the weighting factor input fields in the MV FILE. A single value of P_{HAZ} is entered for each combination of possible failures. For example, the above matrix would be entered in the MV FILE as shown in Figure 5-20 below, given that the MV group has been previously defined in the file.

WGTFACT	1	3							WGTFACT1
WGTFACT1	0	0.8	0	0.8	0.8	0.05	0.8	0.4	

Figure 5-20. MV Entry for Weighting Factors – Engine Failure Example.

Note that the P_{HAZ} for each of the combinations of failures must be entered in a particular way. Table 5-5 and Figure 5-20 use this pattern. To explain this pattern, Table 5-6 was constructed by denoting a failure in Table 5-5 as a “1” and a non-failure as a “0”. Also included in Table 5-6 is the binary sequence from 0 to 2³.

Table 5-6. Entry of P_{HAZ} for Failure Combinations in COVART

Pattern in Table 5-5	Binary Count from 0 to 2 ³
000	000
100	001
010	010
110	011
001	100
101	101
011	110
111	111

The entry of P_{HAZ} follows the exact mirror of a binary sequence from 0 to 2^N, where N is the number of systems in the MV group (3 in this example). More detail on the weighting factor entry fields is discussed in Appendix C of this manual.

We end this discussion with another example. Consider a UEDDAM analysis for the landing roll flight mode where the failure of all three hydraulic systems will result in a catastrophic event 5% of the time due to the inability to use the brakes. Table 5-7 shows the combinations of failures and P_{HAZ} for each combination.

Table 5-7. Hydraulic System Failure Matrix

Hydraulic System #1 State	Hydraulic System #2 State	Hydraulic System #3 State	Probability of Catastrophic Outcome (P _{HAZ})
Functional	Functional	Functional	0.00
Failed	Functional	Functional	0.00
Functional	Failed	Functional	0.00
Failed	Failed	Functional	0.00
Functional	Functional	Failed	0.00
Failed	Functional	Failed	0.00
Functional	Failed	Failed	0.00
Failed	Failed	Failed	0.05

This can be represented in the MV FILE as seen in Figure 5-21.

WGTFACT	1	3							WGTFACT1
WGTFACT1	0	0	0	0	0	0	0	0	0.05

Figure 5-21. MV Entry for Weighting Factors – Hydraulic System Failure Example.

5.2.3.3 Logic Statements

When it comes time to perform an uncontained engine debris analysis, airframe manufacturers in the commercial industry have fault trees either currently in development or already developed for use in performing risk assessments. Some manufacturers develop fault trees using logic statements. It is often the case that translating these logic statements directly into an MV FILE for a UEDDAM analysis is both difficult and time consuming, especially as the use of weighting factors requires specification of 2^N values (not to mention, in a specific order). In order to facilitate the process, UEDDAM has the capability of reading in AND, OR, and NOT logic statements within an MV FILE. The executable *mkwgtf.exe* (which is called by UEDDAM upon encounter of the appropriate keyword in the UEDDAM input control file) converts these logic statements into weighting factors for use by COVART6. Some time must still be spent in order to translate the preexisting logic statements into the proper format for UEDDAM, but much of the more tedious work is performed by the code.

As an example, consider the lateral flight control system of a generic commercial aircraft. This can be a complicated system, consisting of many different components (e.g., pulleys, brackets, control rods, control cables, hydraulic fluid lines, hydraulic actuators, etc.). Suppose that we are provided with a set of 14 logic statements representing the specific combinations of component failures required to defeat the lateral flight control system. For each logic statement, we are also provided with the associated probability of catastrophic outcome during three different flight phases (MVKILL 1, 2, and 3) given that specific combination occurs. The logic statements (in terms of component identification numbers) and associated probabilities are as follows:

```
LFC1 = [(3364 OR 3365) NOT (3651 OR 3652 OR 3653 OR 3654 OR 3740 OR 3741 OR
3742 OR 3743 OR 3744 OR 3745 OR 3100 OR 3101)]
      PHAZ(1) = 0.85, PHAZ(2) = 0.1, PHAZ(3) = 0.4

LFC2 = [3101 NOT (3651 OR 3652 OR 3653 OR 3654 OR 3740 OR 3741 OR 3742 OR 3743
OR 3744 OR 3745 OR 3364 OR 3365 OR 3100)]
      PHAZ(1) = 0.0, PHAZ(2) = 0.2, PHAZ(3) = 0.0

LFC3 = [3100 NOT (3651 OR 3652 OR 3653 OR 3654 OR 3740 OR 3741 OR 3742 OR 3743
OR 3744 OR 3745 OR 3364 OR 3365 OR 3101)]
      PHAZ(1) = 0.1, PHAZ(2) = 0.0, PHAZ(3) = 0.55

LFC4 = [(3100 AND 3101) NOT (3651 OR 3652 OR 3653 OR 3654 OR 3740 OR 3741 OR
3742 OR 3743 OR 3744 OR 3745 OR 3364 OR 3365)]
      PHAZ(1) = 0.8, PHAZ(2) = 1.0, PHAZ(3) = 0.6

LFC5 = [{(3364 OR 3365) AND 3101} NOT (3651 OR 3652 OR 3653 OR 3654 OR 3740 OR
3741 OR 3742 OR 3743 OR 3744 OR 3745 OR 3100)]
      PHAZ(1) = 1.0, PHAZ(2) = 0.0, PHAZ(3) = 0.3

LFC6 = [{(3364 OR 3365) AND 3100} NOT (3651 OR 3652 OR 3653 OR 3654 OR 3740 OR
3741 OR 3742 OR 3743 OR 3744 OR 3745 OR 3101)]
      PHAZ(1) = 0.6, PHAZ(2) = 0.25, PHAZ(3) = 0.1

LFC7 = [{(3364 OR 3365) AND (3100 AND 3101)} NOT (3651 OR 3652 OR 3653 OR 3654
OR 3740 OR 3741 OR 3742 OR 3743 OR 3744 OR 3745)]
      PHAZ(1) = 0.2, PHAZ(2) = 0.0, PHAZ(3) = 0.0
```

LFC8 = [(3651 OR 3652 OR 3653 OR 3654) NOT {(3740 OR 3741 OR 3742 OR 3743 OR 3744 OR 3745) OR (3100 OR 3101)}]
 $P_{HAZ}(1) = 0.15, P_{HAZ}(2) = 0.9, P_{HAZ}(3) = 0.05$

LFC9 = [{(3651 OR 3652 OR 3653 OR 3654) AND (3364 OR 3365)} NOT {(3740 OR 3741 OR 3742 OR 3743 OR 3744 OR 3745) OR 3101}]
 $P_{HAZ}(1) = 0.6, P_{HAZ}(2) = 0.2, P_{HAZ}(3) = 0.3$

LFC10 = [{(3651 OR 3652 OR 3653 OR 3654) AND 3101} NOT {(3740 OR 3741 OR 3742 OR 3743 OR 3744 OR 3745) OR (3364 OR 3365 OR 3100)}]
 $P_{HAZ}(1) = 0.05, P_{HAZ}(2) = 0.9, P_{HAZ}(3) = 0.9$

LFC11 = [{(3651 OR 3652 OR 3653 OR 3654) AND 3100} NOT {(3740 OR 3741 OR 3742 OR 3743 OR 3744 OR 3745) OR (3364 OR 3365 OR 3101)}]
 $P_{HAZ}(1) = 0.75, P_{HAZ}(2) = 0.1, P_{HAZ}(3) = 0.5$

LFC12 = [{(3651 OR 3652 OR 3653 OR 3654) AND (3100 AND 3101)} NOT {(3740 OR 3741 OR 3742 OR 3743 OR 3744 OR 3745) OR (3364 OR 3365)}]
 $P_{HAZ}(1) = 0.0, P_{HAZ}(2) = 0.25, P_{HAZ}(3) = 0.8$

LFC13 = [{(3740 OR 3741 OR 3742 OR 3743 OR 3744 OR 3745)} NOT {(3651 OR 3652 OR 3653 OR 3654) OR (3364 OR 3365 OR 3100 OR 3101)}]
 $P_{HAZ}(1) = 0.1, P_{HAZ}(2) = 0.1, P_{HAZ}(3) = 0.1$

LFC14 = [(3151 AND 3369 AND 3370) NOT (3651 OR 3652 OR 3653 OR 3654 OR 3740 OR 3741 OR 3742 OR 3743 OR 3744 OR 3745 OR 3364 OR 3365 OR 3100 OR 3101)]
 $P_{HAZ}(1) = 0.0, P_{HAZ}(2) = 0.3, P_{HAZ}(3) = 0.65$

The following process details, using the aforementioned logic statements as an example, the steps required in order to build a properly formatted fault tree input file:

1. Define the group of interest.

Example: GRPMULT → LAT_FC

2. Define the systems of interest by examining the preexisting logic statements and looking for reappearing component groupings.

Example: SYSMULT → LAT_1 = 3364 OR 3365
 SYSMULT → LAT_2 = 3651 OR 3652 OR 3653 OR 3654
 SYSMULT → LAT_3 = 3740 OR 3741 OR 3742 OR 3743 OR 3744 OR 3745
 SYSMULT → LAT_4 = 3100
 SYSMULT → LAT_5 = 3101
 SYSMULT → LAT_6 = 3151 AND 3369 AND 3370

3. Redefine the logic statements in terms of the newly defined systems.

Example: LFC1 = [LAT_1 NOT (LAT_2 OR LAT_3 OR LAT_4 OR LAT_5)]
 LFC2 = [LAT_5 NOT (LAT_2 OR LAT_3 OR LAT_1 OR LAT_4)]
 LFC3 = [LAT_4 NOT (LAT_2 OR LAT_3 OR LAT_1 OR LAT_5)]
 LFC4 = [(LAT_4 AND LAT_5) NOT (LAT_2 OR LAT_3 OR LAT_1)]
 LFC5 = [(LAT_1 AND LAT_5) NOT (LAT_2 OR LAT_3 OR LAT_4)]
 LFC6 = [(LAT_1 AND LAT_4) NOT (LAT_2 OR LAT_3 OR LAT_5)]
 LFC7 = [{LAT_1 AND (LAT_4 AND LAT_5)} NOT (LAT_2 OR LAT_3)]
 LFC8 = [LAT_2 NOT {LAT_3 OR (LAT_4 OR LAT_5)}]
 LFC9 = [(LAT_2 AND LAT_1) NOT (LAT_3 OR LAT_5)]
 LFC10 = [(LAT_2 AND LAT_5) NOT {LAT_3 OR (LAT_1 OR LAT_4)}]
 LFC11 = [(LAT_2 AND LAT_4) NOT {LAT_3 OR (LAT_1 OR LAT_5)}]
 LFC12 = [{LAT_2 AND (LAT_4 AND LAT_5)} NOT (LAT_3 OR LAT_1)]
 LFC13 = [LAT_3 NOT {LAT_2 OR (LAT_1 OR LAT_4 OR LAT_5)}]
 LFC14 = [LAT_6 NOT (LAT_2 OR LAT_3 OR LAT_1 OR LAT_4 OR LAT_5)]

4. Develop the MV FILE (as modified for use with logic statements) for the groups, systems, and logic statements developed in steps 1 thru 3.

Example:

```

MVKILL          1
GRPMULT  LAT_FC      6      6      C1
C1      LAT_1  LAT_2  LAT_3  LAT_4  LAT_5  LAT_6
SYSMULT  LAT_1      1      2      C2
C2      3364      3365
SYSMULT  LAT_2      1      4      C3
C3      3651      3652      3653      3654
SYSMULT  LAT_3      1      6      C4
C4      3740      3741      3742      3743      3744      3745
SYSMULT  LAT_4      1      1      C5
C5      3100
SYSMULT  LAT_5      1      1      C6
C6      3101
SYSMULT  LAT_6      3      3      C7
C7      3151      3369      3370
ENDNAME
LOGICST      0.85 LFC1 = LAT_1 .NOT. (LAT_2 .OR. LAT_3 .OR. LAT_4      C8
C8      .OR. LAT_5)
LOGICST      0.00 LFC2 = LAT_5 .NOT. (LAT_2 .OR. LAT_3 .OR. LAT_1      C9
C9      .OR. LAT_4)
LOGICST      0.10 LFC3 = LAT_4 .NOT. (LAT_2 .OR. LAT_3 .OR. LAT_1      C10
C10      .OR. LAT_5)
LOGICST      0.80 LFC4 = (LAT_4 .AND. LAT_5) .NOT. (LAT_2 .OR. LAT_3      C11
C11      .OR. LAT_1)
LOGICST      1.00 LFC5 = (LAT_1 .AND. LAT_5) .NOT. (LAT_2 .OR. LAT_3      C12
C12      .OR. LAT_4)
LOGICST      0.60 LFC6 = (LAT_1 .AND. LAT_4) .NOT. (LAT_2 .OR. LAT_3      C13
C13      .OR. LAT_5)
LOGICST      0.20 LFC7 = (LAT_1 .AND. LAT_4 .AND. LAT_5) .NOT. (LAT_2      C14
C14      .OR. LAT_3)
LOGICST      0.15 LFC8 = LAT_2 .NOT. (LAT_3 .OR. LAT_4 .OR. LAT_5)
LOGICST      0.60 LFC9 = (LAT_2 .AND. LAT_1) .NOT. (LAT_3 .OR. LAT_5)
LOGICST      0.05 LFC10 = (LAT_2 .AND. LAT_5) .NOT. (LAT_3 .OR. LAT_1      C15
C15      .OR. LAT_4)
LOGICST      0.75 LFC11 = (LAT_2 .AND. LAT_4) .NOT. (LAT_3 .OR. LAT_1      C16
C16      .OR. LAT_5)
LOGICST      0.00 LFC12 = (LAT_2 .AND. LAT_4 .AND. LAT_5) .NOT.      C17
C17      (LAT_3 .OR. LAT_1)
LOGICST      0.10 LFC13 = LAT_3 .NOT. (LAT_2 .OR. LAT_1 .OR. LAT_4      C18
C18      .OR. LAT_5)
LOGICST      0.00 LFC14 = LAT_6 .NOT. (LAT_2 .OR. LAT_3 .OR. LAT_1      C19
C19      .OR. LAT_4 .OR. LAT_5)
ENDGROUP
ENDDATA

```

The format used for the logic statements within the MV FILE is as follows.

Keyword Columns 1-8	Probability Columns 9-16	Logic Statement Columns 17-72
LOGICST	0.4	LS1 = (SYS1 .AND. SYS4) .NOT. (SYS2

Logic statements for a particular group should be placed between the ENDNAME and ENDGROUP entries of the corresponding GRPMULT. The following connectives can be used to describe a logic statement: *.AND.*, *.OR.*, and *.NOT.* . The equals sign and

parentheses are recognized symbols; spaces are used as separators. If a logic statement is longer than 54 characters (including spaces, parentheses, etc.), columns 73-80 can be used for a continuation record and the format for the second line is as follows. Note that columns 73-80 on the first line must exactly match columns 1-8 on the second line.

Continuation Record Columns 1-8	Logic Statement (cont.) Columns 9-72
C1	.AND. SYS3 .OR. SYS5)

Upon proper execution of UEDDAM, a file named “generated_mv.txt” will be output; it will contain, in place of the user-defined logic statements, the utility-computed weighting factors as passed to COVART6.

5.2.4 Guidelines/Techniques and Code Limitations

When developing the inputs describing components and their redundancies, the following guidelines and techniques should be followed:

- All COMPPHY entries for components that are part of a COMALIAS record must come before that COMALIAS record in the JTYPE FILE.
- All COMPPHY entries must come before COMPPK entries in the JTYPE FILE.
- Assign a material code of water (51) and a density factor of 1 (or a nominal thickness of -1) to an alias component in the JTYPE FILE.
- A component cannot be both aliased and an alias.
- Component names in the JTYPE FILE should not contain any of the following words or symbols: “ERROR”, “OR”, “AND”, “.”, “/”, “+”, or “-”.
- Do not use blank lines or tabs in any of the COVART6 input files, as they are not recognized; however, spaces are permitted and should be used to align the data to the appropriate column in the file (see Appendix C for specific requirements).
- Each line of internal file documentation should be preceded by \$COMMENT; this keyword tells COVART6 to ignore all other data on that line.
- There is a limit to the “size” of a fault tree in the MV FILE depending on the redundancy level, the number of groups/systems/components, and the number of branches in which a single component exists; fault tree simplification is extremely important.
- The MV FILE limits the fault tree to 24 systems per group, 24 subsystems per system, etc.
- COVART6 has limited the number of weighting factors that can be entered to 2^{20} . Thus, a valid MV FILE can contain weighting factors for a single group of 20 systems or a combination of groups with less than 20 systems as long as the number of factors entered does not exceed 2^{20} .
- A component can only appear in one group within a single flight phase in the MV FILE, although it can appear multiple times in that group.

- Split a single UEDDAM run into multiple runs (based on flight phase) to reduce the total size of the MV FILE.
- The weights and velocities considered in the $P_{\text{haz/h}}$ tables should encompass the entire range of uncontained engine debris being assessed.

5.3 DEFINE ANALYSIS PARAMETERS

The UEDDAM input files described in this section include the following: UEDDAM control file, debris characterizations (DEBRIS FILE), near-field components (NEARFIELD FILE), auto-fail components (AUTOFAIL FILE), material type definitions (MATERIAL FILE), and hole size groupings (HOLEGRP FILE) . These files define the specific analysis parameters (e.g., the location and name of the necessary input files and executables, the desired output files, the specific debris to be analyzed, etc.) as determined in step one of the UEDDAM analysis process.

5.3.1 Control File

The control file provides run specifications for the main UEDDAM code. The format for the input file uses keywords to identify each type of input (note that all inputs are case-sensitive). This allows the sequence of inputs to be re-ordered to suit the analyst. A comma separates each element of input; therefore, a comma cannot be used as part of any text string entries unless the entire string is enclosed within double quotation marks. Whitespace characters (spaces, tabs, and blank lines) may be used to indent data to improve human readability of the file without affecting program operation. The “\$” character, when used as the first character of a line, indicates a comment line and is ignored by the program. The following paragraphs describe the requirements for each input record type and include an accompanying figure to illustrate each record. Comments are included within parentheses in the figures and should not be present in the actual input record.

5.3.1.1 Hazard Level Specification (HAZARD)

The user may enter up to 15 HAZARD records (i.e., flight phases or hazard levels) for an assessment. At least one record must be specified. See Table 4-1 (Section 4.1) for definition of the flight phases.

Keyword	Level ID	Description
HAZARD,	1,	Take-off to V1
HAZARD,	3,	Initial Climb

The “Level ID” corresponds to the flight phase or hazard level and is the same as that used in the JTYPE and MV files. The “Description” is user defined.

5.3.1.2 COVART6 Executable File Name

UEDDAM allows the user to provide a path (either relative or absolute) and name to the COVART6 executable that will be invoked during an analysis. If this keyword is not provided in the control file, UEDDAM will check the computer's environment variable settings for the location of the required executable (these variables are set during installation). If UEDDAM is unable to locate the COVART6 executable through one of these two methods, the user will be prompted to enter the executable's location.

Keyword	File Name
COVART EXEC,	..\bin\covart_ueddams.exe (COVART6 executable path/name)

5.3.1.3 Input File Names

Several files must be accessed by UEDDAM during a run. Files in the following list must already exist prior to beginning a UEDDAM analysis and be similarly listed in the control file[§]. The file names in the following list are for example only; any name can be given to these files. Note that if any or all of these files lie in a different directory from the control file, the path, in addition to the name, must be specified.

Keyword	File Name
TARGET FILE,	geometry.dat (FASTGEN geometric description)
JTYPE FILE,	comp_crit.dat (component criticality reference)
PK FILE,	comp_pk.dat (component kill probability tables)
MV FILE, [§]	comp_redund.dat (hazard tree – MV data) <i>Cannot be used with LOGIC FILE</i>
LOGIC FILE, [§]	comp_logic.dat (hazard tree – MV data with logic statements) <i>Cannot be used with MV FILE</i>
DEBRIS FILE,	debris.dat (Optional. Engine debris characterization data)
NEARFIELD FILE,	comp_near.dat (list by engine for automatic deletion)
AUTOFAIL FILE,	comp_fail.dat (list by engine for automatic failure)
MATERIAL FILE,	mat.dat (material type descriptions)
HOLEGRP FILE,	comp_holesz.dat (component groupings for hole size calculations)

5.3.1.4 Target Reduction Flag (TARGET REDUCTION)

This flag tells UEDDAM whether or not to reduce the target geometry file so that only components within the maximum fore/aft and rotational variance limits are kept for further analysis. Additionally, UEDDAM will then interrogate the fault tree to determine whether or not COVART6 runs are even required with the remaining components. This feature reduces the run time requirements for COVART6 but takes some additional time, up front, to process. Thus, for small target geometries with limited debris characteristics the preprocessing times may exceed actual run times. It is recommended that the user

[§] Note that either of the two hazard tree data files (i.e., MV FILE and LOGIC FILE) can be present in the list of input files, but not both.

test the effectiveness of this feature prior to UEDDAM production runs. If no flag is specified, the default is “0,” corresponding to the target reduction feature being turned off. Note that if the target reduction flag is set to “1” and COVART6 runs are skipped, random number draws will be different which could potentially result in different results than if the target reduction feature were not used.

Keyword	Numerical Values
TARGET REDUCTION,	0 (off) or 1 (on)

5.3.1.5 *Small Component Hypersampling (SMCOMP FIX)*

Because of the way that UEDDAM simulates the penetration of large debris such as disk segments and large fragments (i.e., dividing the debris up into a grid of shotlines), it is possible for relatively small components in the geometry to be missed even though they are directly in the path of the large debris. In order to ensure that the components that should be hit do indeed get hit, only two options were available to the user prior to the release of UEDDAM v3.0: 1) decrease the size of the grid used to define the debris shotlines or 2) increase the size of any small components (and adjust the density/thickness values accordingly) such that they are at least as big as the grid size used to define the debris shotlines. Note that the first option results in exponentially greater run times and the second option results in a geometry that must be rebuilt for each debris type assessed. As neither of these options is optimal, UEDDAM now includes a feature that allows for the generation of additional shotlines within the swath of the debris using a technique referred to as hypersampling.

For the hypersampling routine, UEDDAM employs FastRT, a program developed by SURVICE that performs similar tasks to COVART6 but is optimized for speed. FastRT first raytraces the geometry with the original set of gridded shotlines determined by the grid size specified in the debris file (see Section 5.3.2). UEDDAM then divides the swath even further into hypersampling subgrids of user-specified size. The geometry will then be interrogated again by FastRT to determine whether the hypersampling rays hit any additional components. If so, the first hypersampling ray that hits an additional component is saved and added to the list of shotlines to be fed to COVART6. Figure 5-22 illustrates the concept, with the red arrow representing the additional shotline that would be added to the gridded shotlines (green arrows) before being fed to COVART6.

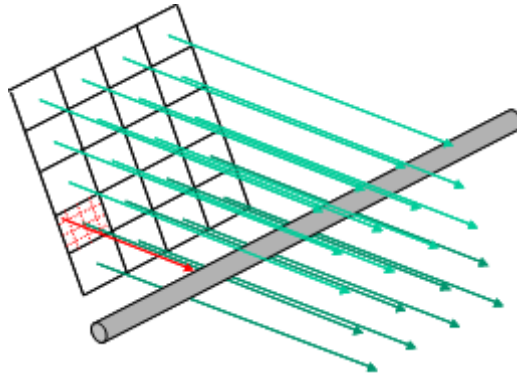


Figure 5-22. Small Component Hypersampling (Subgrid and Saved Ray Highlighted in Red).

In order to avoid a similar problem with small components potentially being missed by small fragments, beginning with the release of UEDDAM v4.3, a hypersampling option has been added for this debris type as well. Because the small fragment debris type is by default represented by a single shotline, it is not possible to simply subdivide an existing grid as is the case for the large fragment and disk debris types. Instead, UEDDAM implements small fragment hypersampling as follows: 1) calculate the four corners of the small fragment area using the length and width dimensions in the appropriate debris characterization record; 2) add four rays to the existing single shotline based on the calculated corners; 3) create a grid of shotlines based on the user-input subgrid size.

The hypersampling routine will only be employed if the appropriate record (denoted by the keywords `SMCOMP FIX SGMT`, `SMCOMP FIX LFRG`, and `SMCOMP FIX SFRG`) appears in the UEDDAM control file. The other inputs include the subgrid size (in inches) UEDDAM should use for the hypersampling routine as well as the position of the shotlines (i.e., either centered or random) within the subgrid.

Keyword	Subgrid	Position
<code>SMCOMP FIX SGMT</code> ,	0.28,	0 (center) or 1 (random)
<code>SMCOMP FIX LFRG</code> ,	0.15,	0 (center) or 1 (random)
<code>SMCOMP FIX SFRG</code> ,	0.04,	0 (center) or 1 (random)

It is recommended that the “position” of the shotline be centered and that $\sim 1/3$ the width of the debris be used for a subgrid size.

5.3.1.6 Multiple Processor Flag (*THREADS*)

This record is optional and allows the user to take advantage of a multi-processor computer to improve run times of UEDDAM. When the keyword `THREADS` is specified in the UEDDAM control file followed by a number greater than 1, UEDDAM will divide COVART6 runs and/or component intercept angle computations among that number of processors. It is recommended that the user enter a number of threads equal to

one less than the total number of processors available as one of the processors will be tied up with the management of the overarching UEDDAM process. When the user enters a -1 after the keyword THREADS, UEDDAM will query the computer to determine the number of available processors.

Keyword	Number
THREADS,	4

5.3.1.7 High-Performance Computer Cluster Flag (*CLUSTER*)

Similar to the keyword THREADS, this record is optional and allows the user to take advantage of the processing power of a high-performance computer (HPC) cluster. (Note that the assumed cluster design is a network of machines that use some form of shared storage system.) When the keyword CLUSTER is specified in the control file, UEDDAM will parse COVART6 runs out to individual nodes of the cluster. The way in which the COVART input and output files are handled depends on the value of the parameter entered after the CLUSTER keyword. If the user enters a value of 0, UEDDAM will have the nodes execute any COVART6 runs on the shared storage system. This option could cause run time performance issues, but might be necessary if the computational nodes have limited local disk space. If the user enters a value of 1 or 2, UEDDAM will have the nodes execute any COVART6 runs in their own local file system and copy the outputs back to the shared storage when the run is complete. The difference between these two parameter values is how the COVART input files are referenced. In the case of a 1, the COVART input files are links to their corresponding file on the shared storage system. If the user enters a value of 2, however, the COVART input files are copied to the local node. The default value of the parameter, should one not be entered, is 1. Also, note that the multiple processor flag (keyword THREADS) and the high-performance computer cluster flag (keyword CLUSTER) are mutually exclusive (i.e., when running UEDDAM on a single computer, a value assigned to the keyword CLUSTER will have no effect on execution; likewise, when running UEDDAM on an HPC cluster, a value assigned to the keyword THREADS will have no effect on execution).

Keyword	Number
CLUSTER,	0 (none), 1 (partial), or 2 (full)

5.3.1.8 Resource Estimation Flag (*RESOURCE ESTIMATOR*)

This record is optional and provides the user with a very rough estimate of the anticipated run time and approximate disk space usage prior to beginning a run. Based on this information, the user is then given the option to continue with the execution or cancel the run altogether. If not included in the control file, UEDDAM defaults to a value of “1”

(on) for the resource estimation flag. Note that for batch processing, the flag should be set to “0” (off).

Keyword	Number
RESOURCE ESTIMATOR,	0 (off) or 1 (on)

5.3.1.9 Gamma Flag (PENETRATION)

This flag represents the inclusion of the obliquity angle, gamma (γ), to the modified FAA V_{50} penetration equation available beginning with the release of UEDDAM v4.1. With this keyword, the user is given the option of specifying 5 potentially different values of γ corresponding to the following obliquity angle bins: 0° – 30° , 30° – 45° , 45° – 60° , 60° – 75° , and 75° – 90° . Note that if this keyword is left out of the UEDDAM control file, UEDDAM will default to using the original FAA V_{50} equation (a similar result is achieved by entering $\gamma=1.0$ for each obliquity angle bin). See Section 3.7 for further definition of gamma and the penetration equations.

Keyword	Value, 0° – 30° bin	Value, 30° – 45° bin	Value, 45° – 60° bin	Value, 60° – 75° bin	Value, 75° – 90° bin
PENETRATION,	0.6,	0.7,	0.8,	0.9,	0.95

5.3.1.10 Output File Names

These records are optional and, when listed in the control file, will result in specific output files being created during a UEDDAM analysis. If none of these records is given, UEDDAM will create the SUMMARY FILE only and name it “HazSummary”. Note that the file names provided are for example only; any name can be given to these files.

Keyword	File Name	Options
SUMMARY FILE,	hazard_sum.txt (release point summaries per event)	—
DETAIL FILE,	hazard_detail.txt (component iteration details)	—
INTERCEPT FILE,	comp_intrcpt.txt (component angle intercept tables),	0.5
HITSUM FILE,	hits_sum.txt (component release point summaries)	—
TRAJECTORY FILE,	trajectory.txt (individual shotline trajectories)	—
VISUALIZER FILE,	visfile.txt (visualizer support file),	0
ERROR FILE,	error.txt (errors and warnings),	1
HOLESIZE FILE,	holesize.txt (component grouping hole sizes)	—
SIGITSYS FILE,	sigitsys.txt (PHAZ significant iteration summaries),	0.99
SIGITCRIT FILE,	sigitcrit.txt (critical component hit summaries)	—
SIGITCHOLE FILE,	sigitchole.txt (cumulative hole area significant iteration summaries),	30
SIGITSHOLE FILE,	sigitshole.txt (hole size significant iteration summaries),	5

Note that several of these files either require or allow for additional options. The INTERCEPT FILE is one; the number that is placed after the file name defines the tolerance for subassembly combining in UEDDAM. In order to perform some of the

angle intercept computations, UEDDAM breaks up components into pieces (or subassemblies). The optional tolerance value determines whether or not the user wants to see component angle intercept tables for entire components or for subassemblies of these components. If the value after the INTERCEPT FILE is zero or omitted, no combining of subassemblies will occur. However, if a tolerance value is included, subassemblies that have bounding box-to-subassembly area ratios greater than or equal to this value will be merged into a single assembly for output purposes.

Three of the four significant iteration summary files also require additional input. The SIGITSYS FILE will provide information detailing iterations whose total system-level PHAZ exceeds the value entered after the filename. In a similar fashion, the SIGITCHOLE FILE and SIGITSHOLE FILE will provide information detailing iterations whose cumulative hole and single hole sizes (in square inches), respectively, exceed the values entered after the filenames.

The final two files with additional optional inputs are the ERROR FILE and the VISUALIZER FILE. For the ERROR FILE, after the user-defined output file name, the user may enter a value defining the level of detail to include in the output: 1 = only fatal errors (this is the default level if none is entered), 2 = standard level of error reporting, and 3 = all warning and error messages (note that this file is most useful for debugging purposes and should not be generated during production-level runs). For the VISUALIZER FILE, the optional input (either a 0 or a 1) tells UEDDAM whether or not to write out information that will be used by the Visualizer to draw critical component risk zones. The default value, if none is entered, is 0 (i.e., UEDDAM will not write out this extra information). Note also that if the user inputs the value 1 for this option, the INTERCEPT FILE must also be selected as an output file (UEDDAM will report an error if this file is not present in the list of output files in the control file). The details and content of all output files are discussed in Section 7 of this manual.

5.3.1.11 Event Identification (EVENT)

This record is used by UEDDAM to define the parameters of an event. Note that each EVENT record must have a corresponding LOCATION record. Descriptions of the non-keyword data elements are provided following the sample record.

Keyword	Event Type	Engine ID	Engine Type	Engine Section	Rotor Stage	# of Iterations	Random # Seed	Debug Flag
EVENT,	DISK,	Engine 1,	High Bypass Turbofan,	HPC,	Stage 5,	100,	9679,	0

Event Type – This is the user-defined identification for the specific event being analyzed. It identifies the appropriate data from the DEBRIS records in the DEBRIS FILE (i.e., the text must appear the same in both records).

Engine ID – This is the user-defined identification for the engine. It identifies the appropriate data from the LOCATION and LIMITS records (i.e., the text must appear the same in all records).

Engine Type – This is the user-defined identification for the type of engine under investigation. It identifies the appropriate data from the DEBRIS records in the DEBRIS FILE (i.e., the text must appear the same in both records).

Engine Section – This is the user-defined identification for the engine section under analysis. It specifies the appropriate data in the LOCATION, LIMITS, and DEBRIS records (i.e., the text must appear the same in all records).

Rotor Stage – This is the user-defined identification for the specific stage of the engine section under analysis. It specifies the appropriate data in the LOCATION and LIMITS records (i.e., the text must appear the same in all records).

Number of Iterations – This specifies the number of iterations in the Monte-Carlo analysis for each nominal release point.

Random Number Seed – This is the random number seed used to initiate the pseudo-random number draws used during a UEDDAM analysis. Note that different compilers use different random number generators and thus, analyses performed with the same random number seed on different computers could potentially produce different results.

Debug Flag – This field indicates if intermediate output files are kept from the COVART6 runs for later use. Its default setting is 0, or no intermediate output files. A setting of 1 indicates that all intermediate output files produced during a COVART run will be saved, whereas a setting of 2 will result in the same output files with the addition of those intermediate output files produced during a FASTGEN run. See Section 6 for additional notes regarding the debug flag.

5.3.1.12 Disk Location and Orientation (LOCATION)

This record identifies the x,y,z coordinates of the undamaged disk origin (or centroid) as well as the orientation (in azimuth and elevation) of its axis (see Figure 5-23). Coordinate values are in units of inches per the geometry file data coordinates. The azimuth and elevation of the axis are in degrees. Alternatively, the disk axis may be oriented using dx/dy/dz direction vectors. Note that the “up” orientation (which corresponds to the 0° debris release angle) is always assumed to be perpendicular to the disk axis (rotated toward the +Z-axis). As an example, consider a tilt-rotor nacelle assembly rotated for hover mode. Since the engine axis is oriented along Az = 0° and El = 90°, the “up” orientation will be considered to be along Az = 0° and El = 0°. If multiple “up” orientations are valid, the azimuth aspect of the disk axis will determine the “up” orientation used by UEDDAM. When only two of the last three data fields contain data, UEDDAM will assume Az/El orientation data in degrees. With data in all three of the last data fields (i.e., dz not blank), UEDDAM will assume direction vector displacement data in inches.

Keyword	Engine ID	Engine Section	Rotor Stage	Rotation Direction	Centroid x-value	Centroid y-value	Centroid z-value	Az (dx)	El (dy)	(dz)
LOCATION,	Engine 1,	HPT,	Stage 3,	CCW,	-330.00,	200.00,	125.00,	0.0,	0.0	

Engine ID – This is the user-defined identification for the engine. It assigns the LOCATION data to the appropriate EVENT record (i.e., the text must appear the same in both records).

Engine Section – This is the user-defined identification for the engine section under analysis. It assigns the LOCATION data to the appropriate EVENT record (i.e., the text must appear the same in both records).

Rotor Stage – This is the user-defined identification for the stage of the engine section under analysis. It assigns the LOCATION data to the appropriate EVENT record (i.e., the text must appear the same in both records).

Rotation Direction – This identifies the direction of engine rotation when standing on the aircraft nose and looking aft. Valid values are “CW” for clockwise and “CCW” for counter-clockwise.

Centroid x, y, and z – The disk origin centroid coordinates (in inches) in the aircraft model.

Az and El or dx, dy, and dz – Engine/disk axis orientation (in degrees or inches).

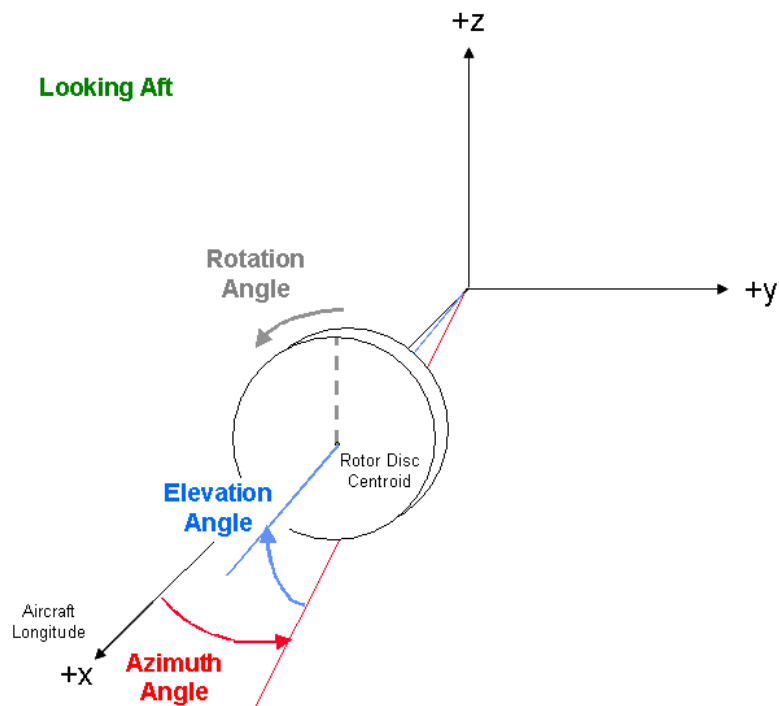


Figure 5-23. Disk Centroid and Orientation Angles.

5.3.1.13 Disk Release Limits (LIMITS)

The LIMITS record is optional. This record specifies the rotational starting position (ASTART), extent of rotation (ASIZE), and step increment (AINCR) for which UEDDAM will generate shotlines and assessments. Nominal release points will be computed for an integer number of steps within the specified rotation, beginning at ASTART (see Figure 5-24). For example, if the user specifies ASTART = 0°, ASIZE = 360°, and AINCR = 15, UEDDAM will assess the entire circumference (with nominal release angles every $360/15 = 24^\circ$). If this record is omitted, UEDDAM will compute results for the entire circumference of the disk using 72 increments (i.e.,

nominal release angles every 5°). If only one release point is desired for a UEDDAM run, the value of **ASTART** must be set to the desired rotational release angle and the values of **ASIZE** and **AINCR** must be set to 360 and 1, respectively (or 0 and 0, respectively).

Keyword	Engine ID	Engine Section	Rotor Stage	Start Value (ASTART)	Size Value (ASIZE)	No. of Increments (AINCR)
LIMITS,	Engine 1,	HPT,	Stage 3,	75.0,	115.0,	4

Engine ID – This is the user-defined identification for the engine. It assigns the rotational LIMITS data to the appropriate EVENT record (i.e., the text must appear the same in both records).

Engine Section – This is the user-defined identification for the engine section under analysis. It assigns the rotational LIMITS data to the appropriate EVENT record (i.e., the text must appear the same in both records).

Rotor Stage – This is the user-defined identification for the stage of the engine section under analysis. It assigns the LIMITS data to the appropriate EVENT record (i.e., the text must appear the same in both records).

Start Value – Specifies the number of degrees of rotation from top-dead-center to the start of the assessment rotational zone. See Figure 5-24, variable **ASTART**.

Size Value – Specifies the number of degrees of arc through which the assessment is processed. A value of 360 (or greater) will always perform a single complete rotation regardless of the starting rotation. See Figure 5-24, variable **ASIZE**. (The stop value for a partial analysis is obtained by sweeping through **ASIZE** degrees from **ASTART**.)

Number of Increments – Specifies the number of step increments to evaluate within the assessment arc. The first assessment will use the starting rotation (**ASTART**) for its nominal release point and assess **AINCR** additional release points subsequently. If the last release point matches the first (i.e., **ASIZE** = 360), that point will not be processed. See Figure 5-24, variable **AINCR**.

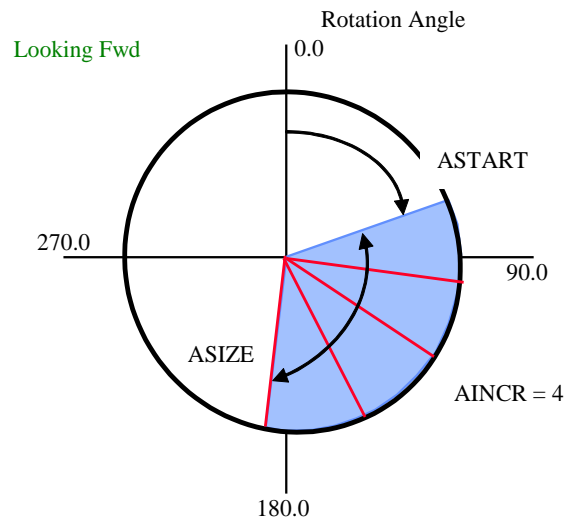


Figure 5-24. Disk Assessment Angles and Step Increments.

5.3.1.14 Sample Control File

A sample control file is illustrated in Figure 5-25 for assessing a single event from one engine section. Probability of hazard results are generated for three defined hazard levels, and all output files are created.

```
$-----Hazard Levels-----
HAZARD, 1, First Level
HAZARD, 2, Second Level
HAZARD, 3, Third Level
$
$-----Executable-----
COVART EXEC, ..\..\bin\covert_ueddams.exe
$
$-----Input Files-----
TARGET FILE, ..\Geometry\Basic_Aircraft.fg4
JTYPE FILE, jtype.txt
PK FILE, pkh.txt
MV FILE, mv.txt
NEARFIELD FILE, UEDDAM_Nearfield.txt
AUTOFIELD FILE, UEDDAM_Autofield.txt
DEBRIS FILE, ..\Debris\disk.txt
MATERIAL FILE, material.txt
HOLEGROUP FILE, hole_groups.txt
$
$-----Output Files-----
SUMMARY FILE, Outputs\summary.txt
DETAIL FILE, Outputs\detail.txt
INTERCEPT FILE, Outputs\comp_intercept.txt
HITSUM FILE, Outputs\hitsum.txt
TRAJECTORY FILE, Outputs\traj.txt
VISUALIZER FILE, Outputs\visual.txt, 1
ERROR FILE, Outputs\error.txt, 1
HOLESIZE FILE, Outputs\hole_sizes.txt
SIGITSYS FILE, Outputs\system.txt, 0.97
SIGITCRIT FILE, Outputs\critical.txt
SIGITSHOLE FILE, Outputs\cumulative.txt, 5.3
SIGITSHOLE FILE, Outputs\single.txt, 1.2
$
$-----Option Flags-----
TARGET REDUCTION, 0
SMCOMP FIX SGMT, 0.33, 1
THREADS, 3
RESOURCE ESTIMATOR, 0
PENETRATION, 0.6, 0.65, 0.7, 0.75, 0.8
$
$-----Event Definition-----
EVENT, DISK, Left Engine, High Bypass Turbofan, FAN, Stage 1, 1, 2405, 1
LOCATION, Left Engine, FAN, Stage 1, CCW, 76.5, 200.0, -57.0, 0, 0
LIMITS, Left Engine, FAN, Stage 1, 335.0, 12.0, 3
```

Figure 5-25. Example of the UEDDAM Control Input File.

5.3.2 Debris Characterization File (DEBRIS FILE)

The DEBRIS FILE contains information characterizing the specific debris for use in a UEDDAM assessment. As with the UEDDAM control file, lines beginning with the “\$”

symbol are comments and are ignored by the program. Note that the same records described below in sections 5.3.2.1 thru 5.3.2.4 can be included in the UEDDAM control file instead of (or in addition to) the debris file. If these records are in both the DEBRIS FILE and in the UEDDAM control file, then the control file data will supersede the DEBRIS FILE data.

5.3.2.1 Disk Debris Mode Record (DEBRIS)

The DEBRIS record is the first of a series of records, all of which must occur at least once for each EVENT record. The DEBRIS record relates the subsequent debris characterization records to a given event. This record is associated with one or more debris group identifier records (Section 5.3.2.2), one or more debris characterization records (Section 5.3.2.3), and zero or more debris mass distribution records (Section 5.3.2.4).

Keyword	Engine Type	Event Type	Engine Section	Debris Type I	Debris Type II
DEBRIS,	High Bypass Turbofan,	DISK,	LPT,	“All” or “SEGMENT” or “L-FRAGMENT” or “S-FRAGMENT”,	Blank or “SEGMENT” or “L-FRAGMENT” or “S-FRAGMENT”,

Engine Type – This is the user-defined identification for the engine. It assigns the DEBRIS data to the appropriate EVENT record in the control file (i.e., the text must appear the same in both records/files).

Event Type – This is the user-defined identification for the type of event that the debris represents. It assigns the DEBRIS data to the appropriate EVENT record in the control file (i.e., the text must appear the same in both records/files).

Engine Section – This is the user-defined identification for the engine section under analysis. It assigns the DEBRIS data to the appropriate EVENT record in the control file (i.e., the text must appear the same in both records/files).

Debris Type I/II – These fields identify the types of fragments produced in the event to be assessed. The first debris type field can contain the word “All” to indicate that all debris types for this event are to be assessed. When the first debris type field is specified “All,” the second debris type field is blank. Individual debris group identifiers (SEGMENT, L-FRAGMENT, and S-FRAGMENT) indicate that only one or two debris types will be assessed (rather than all characterized). The second debris type field is used to specify the second type of debris to be assessed.

5.3.2.2 Debris Group Identifiers

The debris group identifier record follows a DEBRIS record and may be one of three types identifying whether the debris group is a disk segment (SEGMENT), a large fragment (L-FRAGMENT), or a small fragment (S-FRAGMENT). One or more of these records are required based on the entries in the debris type fields in the DEBRIS record

(Section 5.3.2.1). Figure 5-26 and Figure 5-27 illustrate the parameters defined for the debris group identifiers.

Keyword	Centroid Radius	Number of Debris Groups	Grid Size	Multi-Segment Debris
SEGMENT,	10.0,	3 (must be ≤ 10),	1,	1

(further description for this debris type is described in Sections 5.3.2.3 and 5.3.2.4)

Keyword	Centroid Radius	Number of Debris Groups	Grid Size
L-FRAGMENT,	12.5,	5 (must be ≤ 10),	1

(further description for this debris type is described in Sections 5.3.2.3 and 5.3.2.4)

Keyword	Centroid Radius	Number of Debris Groups
S-FRAGMENT,	11.0,	2 (must be ≤ 10)

(further description for this debris type is described in Section 5.3.2.3)

Centroid Radius – This field contains the release radius from the disk origin centroid to the debris center of mass being thrown from the engine (see Figure 5-26). The centroid radius is entered in inches to be consistent with geometry coordinates. In the case of a SEGMENT, the centroid radius (R_c) is calculated as follows:

$$R_c = \frac{4R \sin(\theta/2)}{3\theta}$$

where R is the disk radius (measured in inches) and θ is the degrees of arc (measured in radians) that the segment subtends.

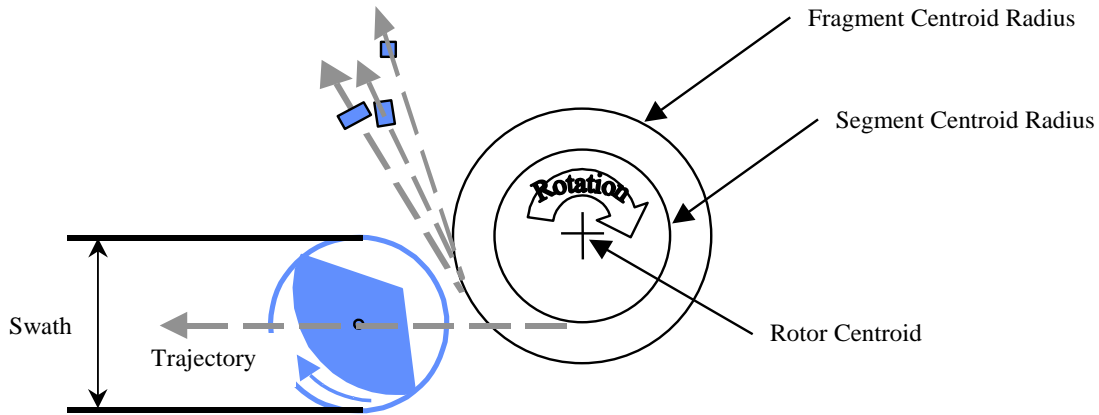


Figure 5-26. Debris Release Radii.

Number of Debris Groups – This field specifies the number of debris characterization records (Section 5.3.2.3) that must accompany this type of debris for this event. Note that in the case of S-FRAGMENTS and L-FRAGMENTS, each debris group represents a different average mass and thus does not have to be equal to the total number of fragments.

Grid Size – This field specifies the distance in inches between adjacent shotlines generated for the parallel array of grids when assessing segment and large fragment debris types. Fragment swath dimensions are computed from debris characterization dimensions and combined with the grid size to determine shotline origins for each grid cell within the swath. Figure 5-27 illustrates the parallel ray grid relationship to disk segment and large fragment dimensions.

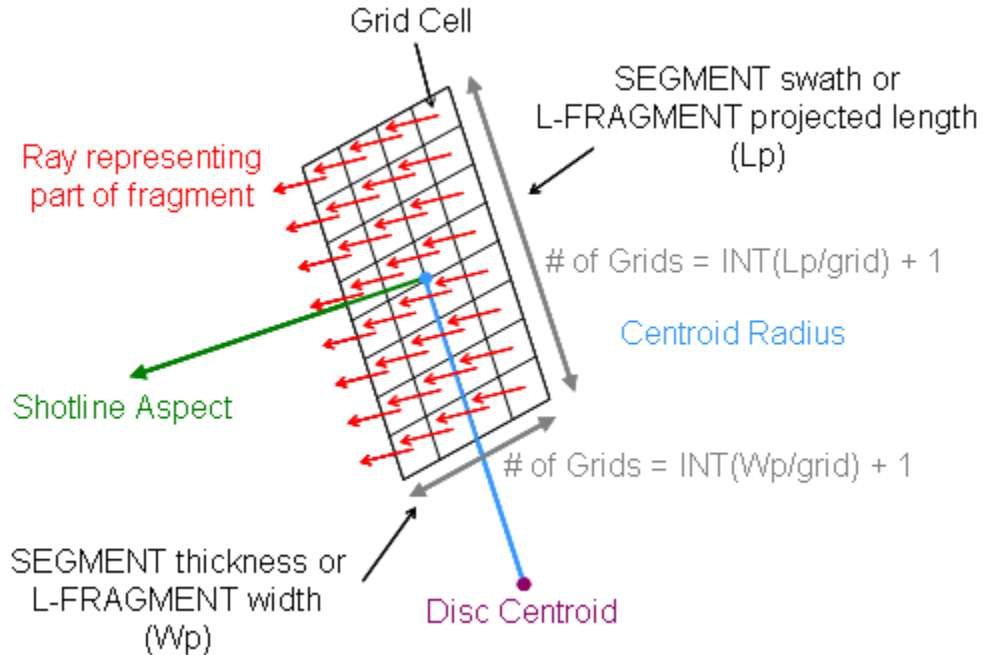


Figure 5-27. Parallel Ray Grid for Generic Fragment Dimensions.

Multi-Segment Debris – This optional field identifies how the user wants to simulate the release of small and/or large fragments with each segment being released in a multi-segment analysis. A value of “0” indicates that small and/or large fragments will be released with the first segment only; this is the default behavior (which will result if this field is left blank). A value of “1” indicates that small and/or large fragments will be released with all segments. Note that the small and/or large fragment characterization(s) will be the same for each segment release, the offset for small and/or large fragments is controlled by the segment offset, and the variance and fore/aft trajectories will be redrawn for each release (see Section 5.3.2.3).

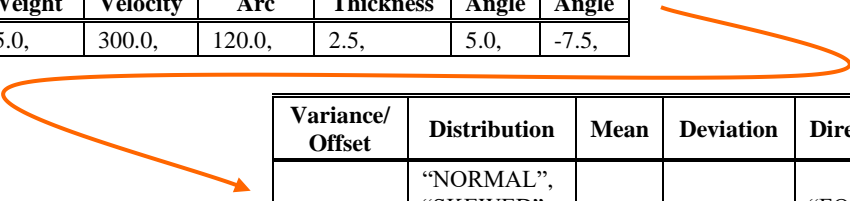
5.3.2.3 Debris Characterizations

This record is repeated “Number of Debris Groups” times (identified on the preceding record described in Section 5.3.2.2). Unlike other records described thus far, this record lacks a keyword and *must immediately follow* its respective debris group identifier record (Section 5.3.2.2) or mass distribution record (Section 5.3.2.4) for the same debris type. A maximum of 10 fragment characterization groups may be used within each debris type for multiple fragment assessments. Due to the large masses involved with uncontained engine debris, input units for fragment weight are in pounds rather than grains, as in the PK FILE. The remaining inputs are in standard English units (i.e., feet per second,

inches, and degrees, as appropriate). The format for this record varies depending on the debris type that it follows.

Following a SEGMENT record:

Group ID	Segment Material	Segment Weight	Release Velocity	Segment Arc	Rotor Thickness	Fwd Angle	Aft Angle
1,	Steel,	45.0,	300.0,	120.0,	2.5,	5.0,	-7.5,



Variance/Offset	Distribution	Mean	Deviation	Direction
10.0,	“NORMAL”, “SKEWED”, “UNIFORM”, or blank	-1.0,	0.5,	“FORE” or “AFT”

Group ID – Identifies the i^{th} group number from within “Number of Debris Groups” for the debris type.

Segment Material – This UEDDAM material name is either selected from the predefined material types listed in Table 5-4 or is defined in the MATERIAL FILE (see Section 5.3.5).

Segment Weight – Weight of disk segment in pounds.

Release Velocity – Specifies the velocity of the segment at the exterior surface of the engine nacelle (in fps). Note that if the cowl structure is modeled, then the exit velocity should be the fragment initial velocity based on the fragment centroid radius and rotor speed minus 25% for frictional effects. The formula to use in this case is:

$$\text{Exit Velocity} = 0.75 \times (\pi/360) \times \text{RPM} \times R_c,$$

where:

R_c = centroid radius in inches, and
RPM = engine revolutions per minute.

Segment Arc – The number of degrees of arc that the segment contains. For example, 1/3 of a disk equals an arc segment of 120 degrees.

Rotor Thickness – Maximum thickness of the rotor disk in inches (see Figure 5-28). Segment material density is treated uniformly across the thickness.

Forward Angle – This field specifies the forward angle (see Figure 5-28), which represents the maximum forward deviation from the disk’s plane of rotation from which the debris could potentially be released. A negative forward angle indicates that all debris is released aft of the plane of rotation.

Aft Angle – This field specifies the aft angle (see Figure 5-28), which represents the maximum aft deviation from the disk’s plane of rotation from which the debris could potentially be released. A positive aft angle indicates that all debris is released forward of the plane of rotation.

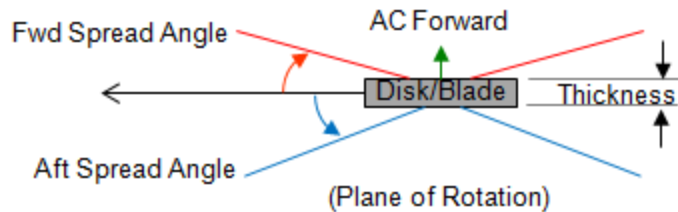


Figure 5-28. Debris Characterization – Fore/Aft Spread Angles.

Variance/Offset – As shown in Figure 5-29, the variance is a delta angle in the plane of rotation around the nominal release angle in which a random number draw will be used to determine the actual release angle used for shotline generation. When multiple segments are released from a single disk stage, the variance/offset field for subsequent segments specifies an angular offset from the first segment’s nominal release angle. For example, suppose three equally divided segments (120 degrees each) are thrown from an engine. The first segment has a variance of 5 degrees, the second segment has an offset of 90 degrees, and the third segment has an offset of 200 degrees. Given a release point of 50 degrees, the first segment trajectory is drawn randomly within the variance for the release point (i.e., $50^{\circ} \pm 5^{\circ}$); the second segment trajectory is drawn randomly within the same variance for the nominal release point plus offset1 (i.e., $140^{\circ} \pm 5^{\circ}$); and the third segment trajectory, like the second, is drawn randomly within the same variance for the nominal release point plus offset2 (i.e., $250^{\circ} \pm 5^{\circ}$). Note that if the user wants to simulate the release of more than one segment at a single release point, all but the first segment must be converted to large fragments (see L-FRAGMENT debris characterization record).

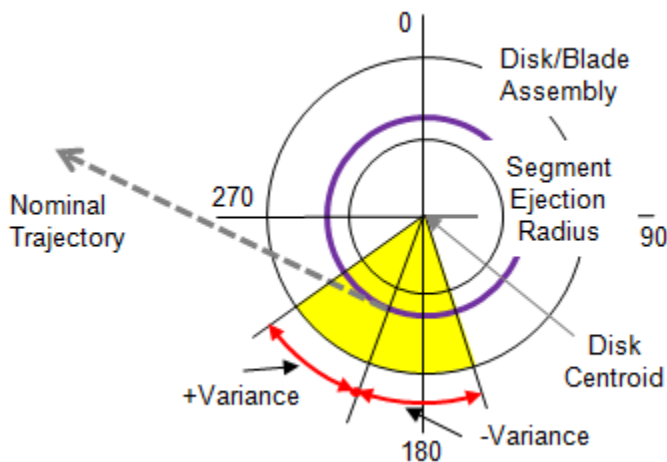


Figure 5-29. Debris Characterization – Variance.

Distribution – During a UEDDAM analysis, a random draw is performed to select the specific angle of deviation from the disk’s plane of rotation from which debris will be released. If the keyword UNIFORM appears in the Distribution field, each angle within the fore/aft spread is equally likely to be chosen in the random draw (see Figure 5-30). If the NORMAL distribution is chosen, one particular angle μ is more likely to be drawn than any other and symmetric angles fore and aft of μ are equally likely to be drawn (see Figure 5-31). If the distribution is SKEWED, one particular angle γ is again more likely to be drawn than any other but symmetric angles fore and aft of γ are no longer equally likely to be drawn (see Figure 5-32). (Note that the SKEWED distribution is a modification of the statistical Gamma distribution.) If this field remains blank, the default distribution is UNIFORM.

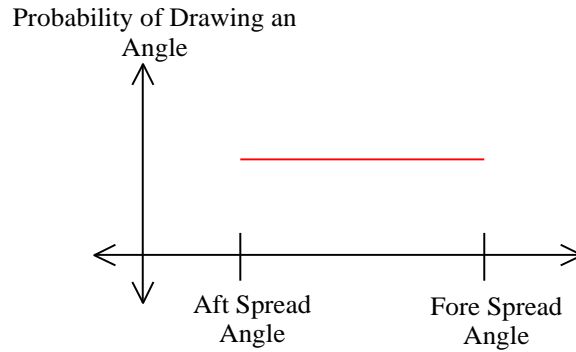


Figure 5-30. Debris Characterization – Uniform Distribution.

Mean – The angle that is most likely to be chosen in a non-uniform random draw (i.e., μ or γ , depending on the selected distribution). (See Figure 5-31 and Figure 5-32.) The mean must lie in the interval between the aft spread angle and the fore spread angle. Note that if the UNIFORM distribution is selected, this entry is not required.

Deviation – The rate at which the selected non-uniform distribution falls off on either side of the mean (measured in degrees). For the NORMAL distribution, the deviation is also referred to as the standard deviation (σ). For the SKEWED distribution, the deviation is also referred to as a scale or stretch parameter (λ). (See Figure 5-31 and Figure 5-32.) In either case, the deviation must be greater than zero. Note that if the UNIFORM distribution is selected, this entry is not required.

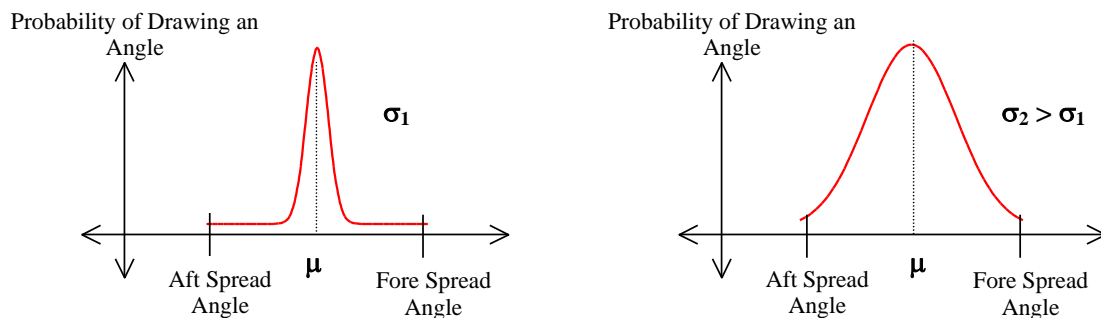


Figure 5-31. Debris Characterization – Normal Distribution.

Direction – This field specifies in which direction (either FORE or AFT of the mean) the angles are more likely to be drawn from a SKEWED distribution (see Figure 5-32). Note that if the UNIFORM or NORMAL distribution is selected, this entry is not required.

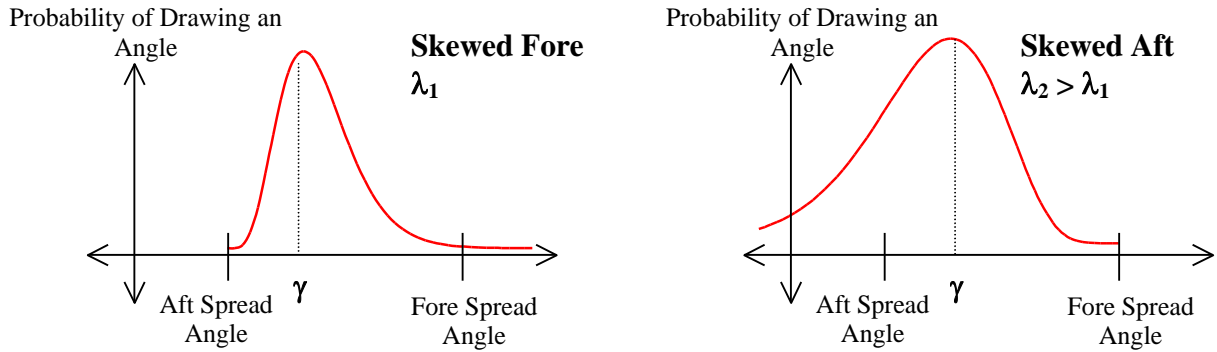


Figure 5-32. Debris Characterization – Skewed Distribution.

Following an L-FRAGMENT or S-FRAGMENT record:

Group ID	Count	Frag. Material	Weight	Velocity	Length	Width	Thickness	Orientation
1,	10,	Steel,	0.25,	500.0,	4.0,	2.0,	0.3,	45.0,

Sigma	Fwd. Angle	Aft Angle	Variance	Distribution	Mean	Dev.	Dir.
45.0,	30.0,	-70.0,	30.0,	“NORMAL”, “SKEWED”, “UNIFORM”, or blank	-25.0,	5.0,	“FORE” or “AFT”

Group ID – Identifies the i^{th} group number from within “Number of Debris Groups” for the debris type.

Count – Specifies the number of fragments within the current mass group.

Fragment Material – This UEDDAM material name is either selected from the predefined material types listed in Table 5-4 or is defined in the MATERIAL FILE (see Section 5.3.5).

Weight – The average weight of the individual fragments in pounds.

Velocity – Specifies the average exit velocity of the fragments at the exterior surface of the engine nacelle in feet per second. Note that if the cowl structure is modeled, then the exit velocity should be calculated using the equations shown in the preceding SEGMENT format discussion.

Length – Specifies the length of the fragment in inches (see Figure 5-33). If converting a SEGMENT to an L-FRAGMENT, length is calculated as follows:

$$\text{length} = 2R\sin(\theta/2),$$

where R is the disk radius and θ is the segment arc.

Width – Specifies the width of the fragment in inches (see Figure 5-33). If converting a SEGMENT to an L-FRAGMENT, width is set equal to the grid size.

Thickness – Specifies the thickness of the fragment in inches (see Figure 5-33). If converting a SEGMENT to an L-FRAGMENT, fragment thickness is the same dimension as rotor thickness.

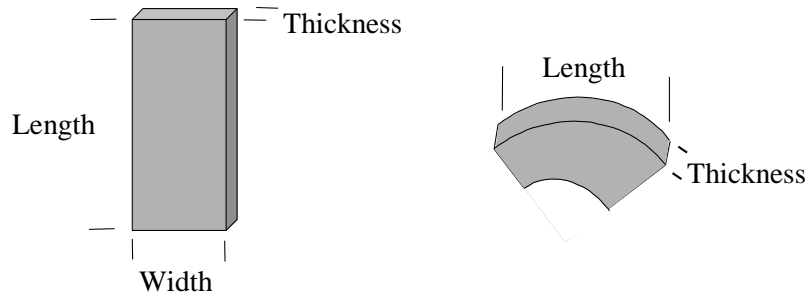


Figure 5-33. Sample Fragment Dimensions.

Orientation – The nominal orientation (tilt) angle between the fragment’s trajectory direction and its primary surface normal direction (see Figure 5-34). If converting a SEGMENT to an L-FRAGMENT, orientation should be set to 0.

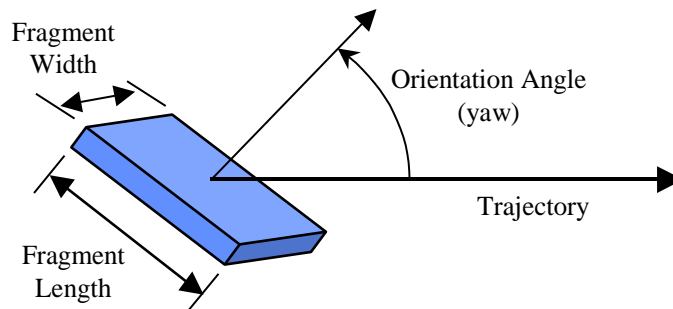


Figure 5-34. Fragment Orientation.

Sigma – The maximum delta in fragment orientation angle from the nominal value. Used in Monte-Carlo analysis. If converting a SEGMENT to an L-FRAGMENT, sigma should be set to 0.

Forward Angle – Forward spread angle as shown in Figure 5-28 and discussed in the preceding SEGMENT format discussion.

Aft Angle – Aft spread angle as shown in Figure 5-28 and discussed in the preceding SEGMENT format discussion.

Variance – As shown in Figure 5-29, the variance angle is an angle range in the plane of rotation around the nominal release angle in which a random number draw determines the actual release angle used for shotline generation.

Distribution – As discussed in the preceding SEGMENT format discussion and seen in Figures 5-30, 5-31, and 5-32, the distribution from which a random draw is performed to select the specific angle of deviation from the disk’s plane of rotation from which debris will be released. The optional inputs for this field are NORMAL, UNIFORM, or SKEWED. If left blank, the default distribution is UNIFORM.

Mean – As shown in Figure 5-31 and Figure 5-32, the angle that is most likely to be chosen in a non-uniform random draw (i.e., μ or γ , depending on the selected distribution). The mean must lie in the interval between the aft spread angle and the forward spread angle. Note that if the UNIFORM distribution is selected, this entry is not required.

Deviation – As discussed in the preceding SEGMENT format discussion and seen in Figure 5-31 and Figure 5-32, the rate at which the selected non-uniform distribution falls off on either side of the mean (i.e., σ or λ , depending on the selected distribution). The deviation, in both cases, must be greater than zero. Note that if the UNIFORM distribution is selected, this entry is not required.

Direction – This field specifies in which direction (either FORE or AFT of the mean) the angles are more likely to be drawn from a SKEWED distribution (see Figure 5-32). Note that if the UNIFORM or NORMAL distribution is selected, this entry is not required.

5.3.2.4 Debris Mass Distribution

In an attempt to better represent the performance of the disk segments and large fragments during penetration, UEDDAM includes an option to vary the mass distribution of these debris types. For example, near the center of mass, the segment/fragment realistically retains its full penetration capability, while near its edges, it loses some penetration capability (due to rotation, tapered shape, etc.). It is not intended for the sum of the mass distribution to equal the total mass of the fragment. The intention of this feature is to permit the user to adjust the mass profile to better simulate the penetration effects of a large fragment on nonhomogeneous structure. This record must directly follow its respective SEGMENT or L-FRAGMENT record.

No. of Mass Divisions (N)	Dist. Fraction for Mass 1	Mass Fraction for Mass 1	Dist. Fraction for Mass 2	Mass Fraction for Mass 2	...	Dist. Fraction for Mass N	Mass Fraction for Mass N
10,	0.25,	1.0,	0.50,	1.0,	...	1.0,	0.8,

Number of Mass Divisions – This field specifies the number of distance/mass pairings to be processed for the associated debris characterization (Section 5.3.2.3). The number of divisions counts from the center of the SEGMENT/L-FRAGMENT to one edge of its length. Each division will be mirrored for the length from the center to the opposite edge. Figure 5-35 illustrates a mass distribution for four divisions.

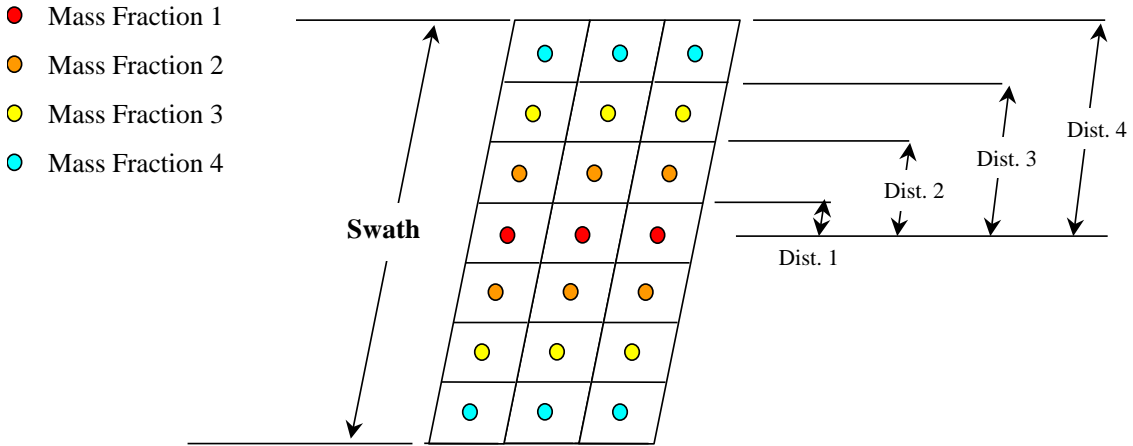


Figure 5-35. Mass Distribution Fraction.

Distance Fraction – This field specifies the fractional displacement from the debris center to the outer boundary for a given mass fraction. All shotlines in the parallel grid that are located within this distance fraction but beyond the previous distance fraction are assigned the paired mass fraction value. Note that the shading of the ranges and the distance parameters in Figure 5-35 are actually the distance fraction multiplied by one-half the swath length.

Mass Fraction – This field specifies the percent of the SEGMENT/L-FRAGMENT mass to apply to penetration equations within COVART. For example, components near the center of the swath are impacted with the full penetration force of the debris. As the distance between the component and swath centerline increases, the penetration force reduces possibly deflecting the debris rather than penetrating the component (i.e., a glancing blow). Note that the trajectory itself is not altered, this only accounts for reduced damage and failure due to a glancing blow. Distance fraction and mass fraction must be entered as pairs. Be careful not to split the pair across record boundaries for readability (distance on one line and mass on the next is an error); continuation lines must start with a distance value and all lines must end with a mass value.

5.3.2.5 Generic Engine Debris Model

The generic debris model [5] (reproduced in Appendix E) was compiled from multiple sources of uncontained engine debris damage data by the NAWCWD, China Lake and contains fragment characterizations for turboprop, low bypass ratio, and high bypass ratio engine categories.

It is often of interest to develop an engine debris model specifically for an assessment. In order to generate this data, the generic debris fragment model (a database of analyzed and compiled historical event data) must be applied to the specific engine disk and blade physical characteristics to determine the physical size and weight of uncontained debris fragments. The fragment release velocity used can be either the standard fragment velocity (as presented in the generic debris fragment model) or calculated based on engine physical speeds, nacelle description, and engine rotor speeds. As an example, consider a steel high pressure turbine (HPT) blade of a high bypass ratio (HBR) engine with physical dimensions blade length = 2.25 in., blade width = 1.5 in., blade thick-

ness = 0.1 in., and disk radius = 12 in. According to the generic debris fragment model (see Appendix E), for an HPT blade event, the average number of fragments released per event is 10; the normalized size of a blade fragment is 80% (where “normalized size” is the fragment length divided by the blade length, in this case); the debris weight is 0.25 lbs.; the standard velocity of a blade fragment is 337 feet per second; and the spread angle is -50° to 20°. Thus, applying this model to the physical HPT blade, the resulting debris has dimension length = $0.80 \times 2.25 = 1.8$ in., with other dimensions remaining the same. Assuming that the blade debris is released from the disk rim, the centroid radius for the S-FRAGMENT is $12 + (1.8/2) = 12.9$ in. In order to complete the debris record for this HPT blade event, there are three remaining inputs that cannot be determined from the generic engine debris model or the engine physical characteristics: the fragments’ orientation angle, sigma, and release variance. For these parameters, the values 45°, 0°, and 20° are chosen as representative. The completed debris file entry for this HBR engine HPT blade event can be seen in Figure 5-36 in the form of the records described in Sections 5.3.2.1 thru 5.3.2.4.

```

$High Bypass Ratio Engine Fragment Characterization
$ HPT Section
DEBRIS,High Bypass Ratio,BLADE,HPT,S-FRAGMENT
S-FRAGMENT,12.9,1
1,10,Steel_300,0.25,337,1.8,1.5,0.1,45,0,20,-50,20,UNIFORM

```

Figure 5-36. Generic Engine Debris Model Example Debris File Entry.

5.3.3 Near-Field Components File (NEARFIELD FILE)

The NEARFIELD FILE contains a list of geometric components that are treated as so-called “near-field” components. Near-field components are the small, non-shielding components internal to each engine nacelle. This input is required in order to exclude these components from the shotline output as they have typically already been considered in the debris mass and velocity characterization data. This list should not, however, contain any nearby components that may act as significant shielding to the engine debris. There is only one record type for this file (keyword NEARFIELD). It identifies the engine for which the near-field components are assigned and lists all of the associated near-field components.

Keyword	Engine ID	Count (N)	Comp. ID 1	Comp. ID 2	Comp. ID 3	...	Comp. ID N
NEARFIELD,	Engine 1,	10,	1001,	1002,	1003,	...	1010

Engine ID – This is the user-defined identification for the engine. It assigns the NEARFIELD data to the appropriate EVENT record in the control file (i.e., the text must appear the same in both records/files).

Count – The number of near-field components associated with this engine.

Component ID # – The component IDs are the component identification numbers (from the TARGET FILE) for each near-field component. The component IDs can wrap over successive lines, and there are no restrictions on how many components may be listed per line or on how many lines. The only requirement is that all of the components associated with the engine are entered in succession (i.e., numerical order) and separated by a comma space.

Figure 5-37 shows an example of a NEARFIELD FILE for a twin-engine aircraft.

```
NEARFIELD, Engine 1, 10, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010
NEARFIELD, Engine 2, 15, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061,
1062, 1063, 1064, 1065
```

Figure 5-37. Example of NEARFIELD FILE.

5.3.4 Auto-Fail Components File (AUTOFAIL FILE)

The AUTOFAIL FILE is required in order to automatically fail components and systems that are driven by the engine experiencing the uncontained debris event. The so-called “auto-fail” components are used during evaluation of component and system redundancy for each uncontained engine debris event. There is only one record type for this file (keyword AUTOFAIL). It identifies the engine for which the automatically failed components are to be assigned and lists all of the flight phases or hazard levels to which the failure is critical.

Keyword	Engine ID	Comp. ID Name	Hazard Levels of Criticality					
AUTOFAIL,	Engine 1,	HYD_P1,	1,	2,	3,	4,	5,	6

Engine ID – This is the user-defined identification for the engine. It assigns the AUTOFAIL data to the appropriate EVENT record in the control file (i.e., the text must appear the same in both records/files).

Component ID Name – The component names (as designated in the JTYPE FILE) for each automatically failed component.

Hazard Levels of Criticality – The hazard levels of criticality are those flight phases or hazard levels for which a failure of the component has a consequence. Up to 15 hazard levels of criticality can be identified per UEDDAM run session.

One record is to be entered for each component related to an engine. There are no limits on the number of components that can be assigned to an engine. Figure 5-38 shows an example of the AUTOFAIL FILE for a twin-engine aircraft.


```

AUTOFAIL, Engine 1, FUEL_P1, 1, 2, 3, 4, 5
AUTOFAIL, Engine 1, BYPASS1, 2, 3, 4, 5
AUTOFAIL, Engine 2, HYD_P2, 1, 2, 3, 4, 5
AUTOFAIL, Engine 2, GEN_2, 2, 3, 4, 5
AUTOFAIL, Engine 2, FUEL_P2, 4, 5
AUTOFAIL, Engine 2, BYPASS2, 4, 5

```

Figure 5-38. AUTOFAIL FILE Example.

5.3.5 Material Properties File (MATERIAL FILE)

Beginning with UEDDAM v4.3, the MATERIAL FILE is now a required input for UEDDAM analyses. This file identifies material names, assigns COVART6 identification numbers, and defines the material properties required for penetration calculations (see Section 3.7). Due to the large variety of materials in use on aircraft today, in addition to the predefined material types available to the user (see Table 5-4), the MATERIAL FILE also allows the user to define up to 44 additional material types for his or her particular study (for a total of 50 non-liquid materials). There is only one record type for this file (keyword MATPROP).

Keyword	UEDDAM Material Name	COVART Material ID	Density	Shear Constant 1 (C _{s_50})	Shear Constant 2 (C _{s_0}) (optional)
MATPROP,	Aluminum_Other,	33,	372.0,	20029.0,	42000.0

UEDDAM Material Name – User-defined name to be associated with the material type. This name is used in the debris characterization records (see Section 5.3.2.3) associated with L-FRAGMENT, S-FRAGMENT, and SEGMENT debris group identifiers.

COVART Material ID – User-selected identification number to be associated with the material. This material ID number is used in the JTYPE FILE to assign a material type to a component. The selected material ID number must be in the range [1,50].

Density – Density (in grains per cubic inch) of the material type.

Shear Constant 1 (C_{s_50}) – Standard shear constant (in psi) associated with the material type (see Reference 10 for more information regarding this parameter). (Note: the value entered for shear constant is converted from psi to Pascals within the FAA penetration equation module of COVART.)

Shear Constant 2 (C_{s_0}) – Secondary shear constant (in psi) associated with the material type (see Section 3.7 for more information regarding this parameter). (Note: the value entered for shear constant is converted from psi to Pascals within the FAA penetration equation module of COVART.) This is an optional record intended to provide additional flexibility to the user to better illustrate the shielding characteristics of materials.

The standard MATERIAL FILE, included in each of the sample case directories as part of the UEDDAM installation package, is additionally provided in Figure 5-39 for reference.

```
MATPROP, Steel_300, 5, 1980.0, 188500.0
MATPROP, Ti_285, 7, 1130.0, 130529.4
MATPROP, Al_2024, 8, 701.0, 40029.0
MATPROP, Graphite, 46, 390.6, 36403.2
MATPROP, Inconel, 47, 2135.0, 174039.2
```

Figure 5-39. Standard UEDDAM MATERIAL FILE.

5.3.6 Hole Size Group File (HOLEGRP FILE)

The HOLEGRP FILE contains a list of geometric components for which UEDDAM will generate hole size statistics. There is only one record type for this file (keyword HOLEGRP). It defines the name of a group for which hole size statistics are desired and lists all of the components that are to be considered part of this group. As this feature was developed for decompression analysis, a typical list of components would include the pressure vessel structure of the skin, windows, doors, floor, and pressure bulkhead. Note that this feature would not be used for rear-mounted engines, where the burst zone is behind the pressure bulkhead.

Keyword	Hole Size Group Name	Count (N)	Comp. ID 1	Comp. ID 2	Comp. ID 3	...	Comp. ID N
HOLEGRP,	Skin,	10,	351,	352,	353,	...,	360

Hole Size Group Name – User-defined name to be associated with the hole size group. This name will be used in the hole size file output.

Count – The number of components associated with the hole size group name.

Component ID # – The component IDs are the TARGET FILE component identification numbers for each HOLEGRP component. The component IDs can wrap over successive lines, and there are no restrictions on how many components may be listed per line or on how many lines. The only requirement is that all of the components associated with the group are entered in succession (i.e., numerical order).

Figure 5-40 shows an example of the HOLEGRP FILE for a twin-engine aircraft.

```
HOLEGRP, Port Wing Skin, 16, 900, 902, 904, 906, 908, 910, 912, 914, 916, 918, 920, 922, 924,
926, 928, 930
HOLEGRP, Fuselage Skin, 100, 351-400, 404,408,450-459, 603, 607, 700, 716-750
HOLEGRP, Starboard Wing Skin, 16, 901, 903, 905, 907, 909, 911, 913, 915, 917, 919, 921,
923, 925, 927, 929, 931
```

Figure 5-40. HOLEGRP FILE Example.

5.3.7 Guidelines/Techniques and Code Limitations

The following are guidelines and known limitations inherent to the UEDDAM code:

- Take care that case, spelling, and spacing is consistent between inputs in the various UEDDAM input files. This is especially true in the case of user-defined identifications (see
- Figure 5-41, which identifies the user-defined engine descriptors associated with an individual event along with all the other records/keywords that use these same descriptors).

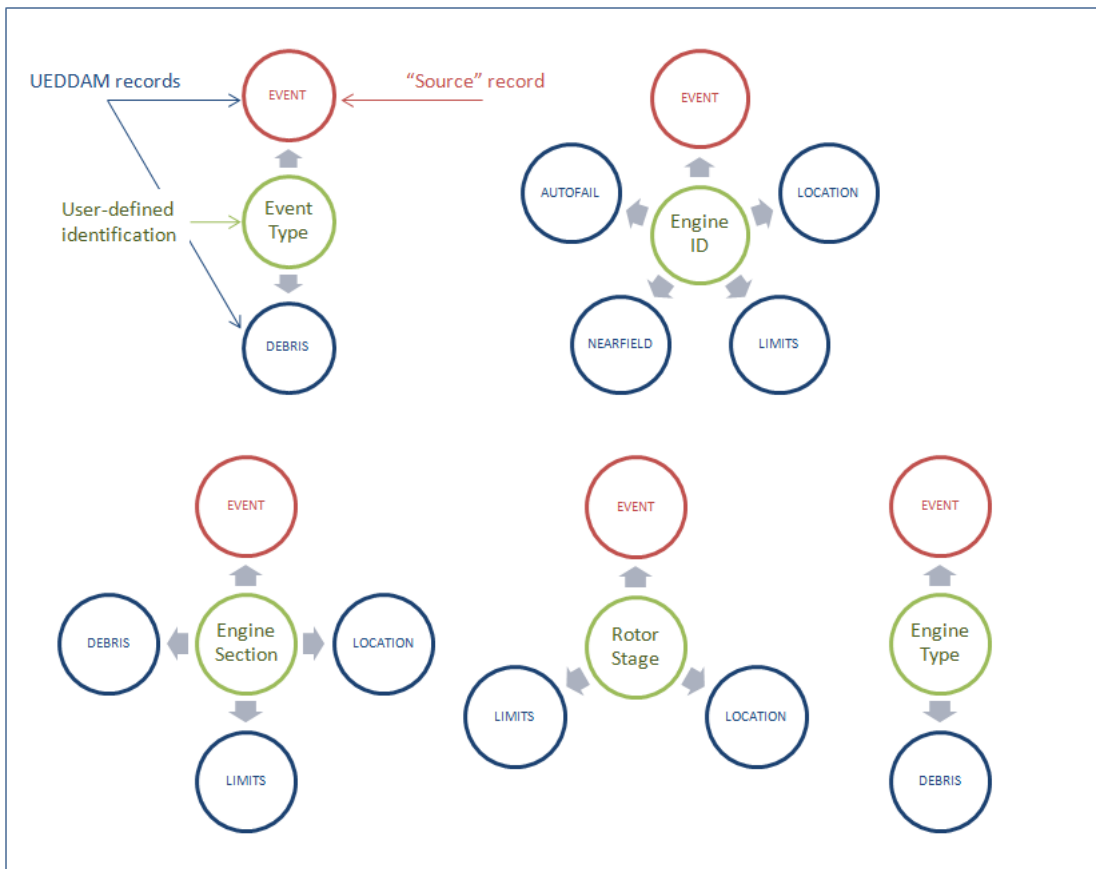


Figure 5-41. User-Defined Identifications and Their Use in Multiple Records.

- When performing batch UEDDAM runs, the full executable pathname in the control file - including executable - is limited by the command window to 260 characters. Additionally, if any of the directory names contain spaces, double quotes must be used around the entire pathname in the control file.
- Do not put blank lines or comment lines between records defining a debris category (i.e., once a record of the type described in Section 5.3.2.1 is entered, no blank lines or comments can be entered until after a record of the type described in Section 5.3.2.4 has been entered). Blank lines and comments are acceptable elsewhere within the debris file.

6 INPUT FILE VERIFICATION AND UEDDAM EXECUTION

After input generation, the next step of the UEDDAM analysis process is to perform a verification of the various input files created. Setting up a “simple” UEDDAM execution and troubleshooting the resulting errors until the run is successful can accomplish the first half of this step in the process. The second half of this step is accomplished by analyzing the results (as defined in the various UEDDAM output files discussed in Section 7) from this “simple” execution. Upon completion of input file verification, production runs can be set up and executed in order to produce results for post processing and analysis.

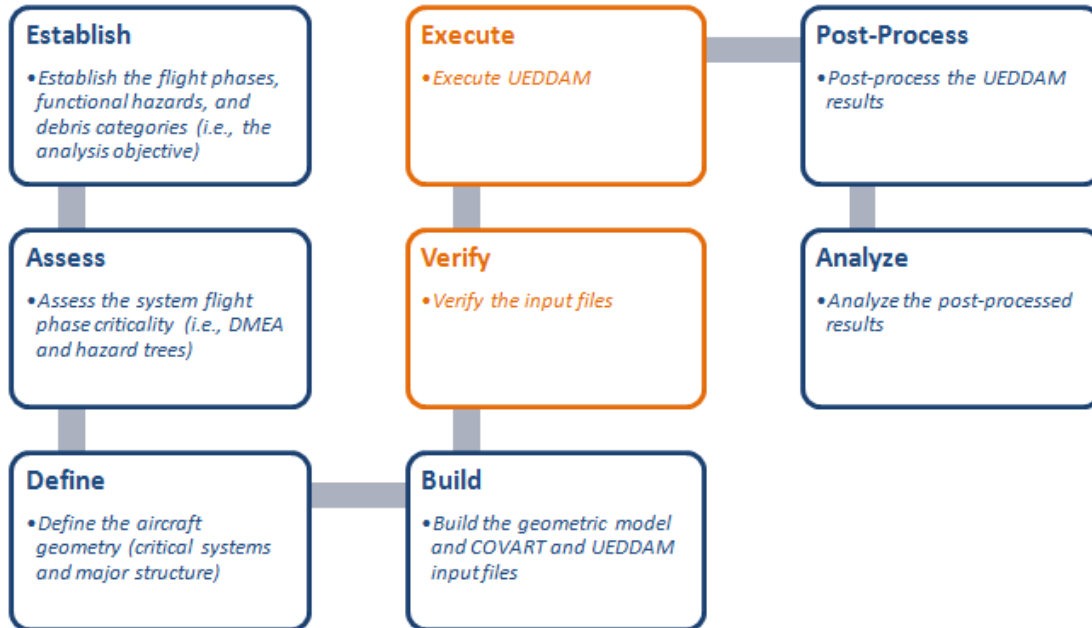


Figure 6-1. Current Location in the UEDDAM Analysis Process: Input File Verification and UEDDAM Execution.

6.1 OPERATING SYSTEM REQUIREMENTS

UEDDAM has been successfully tested on PCs** running the Microsoft Windows 10 and CentOS v6.8 and v7.5 operating systems. It is recommended that UEDDAM production runs be performed on PCs with no less than 512 MB RAM and with processor speeds no less than 1 GHz.

In order to provide the user with example run times, a complete set of events for the generic twin-engine aircraft discussed in Appendix D (a single debris type released every 5° from each of the 18 stages of each engine) was run using UEDDAM v3.1, v5.1, and v6.0 on a variety of computers available to SURVICE and the FAA. Because the three PCs had multiple processors, the THREADS keyword was utilized in all cases. For the

** Note that the hard drive must be NTFS formatted.

HPC executions, the “0” option for the CLUSTER keyword was selected. Run times are illustrated in Table 6-1.

Table 6-1. Sample UEDDAM Run Times

Computer Type		Dell Precision T3400	Dell Precision 7720	Dell Precision 7720	Node Cluster of Dell R630 & R730 Servers with 28 CPUs per Node and a 56 Gb/s Infiniband Network
Computer Description	Processor Type	Intel Core 2 Duo	Intel Xeon E3	Intel Xeon E3	Intel Xeon E5
	Processor Speed	2.66GHz	2.9GHz	2.9GHz	2.3/2.6 Ghz
	RAM	7.93GB	64GB	64GB	256 GB per node
	Hard Drive	750GB	1TB	1TB	N/A
	UEDDAM Version	v3.1	v5.1	v6.0	v6.0
Appendix D – SINGL (5 Iterations)	Run Times	6 hr 18 min	3 hr 36 min	3 hr 46 min	4 hr 1 min*
Appendix D – MULTI (100 Iterations)		6 day 8 hr 5 min	1 day 7 hr 55 min	1 day 4 hr 25 min	0 day 10 hr 40 min**

* 1 Node (28 Cores)

** 4 Nodes (112 Cores)

Of importance to note is that use of the HPC option does not result in runtime improvements over standalone systems for the SINGL analysis due to its low number of iterations. Not until the number of iterations is greater than the number of available cores does use of the HPC option become a valuable alternative.

6.2 RUNNING UEDDAM

UEDDAM execution takes place from a command line. Therefore, it is necessary to open a command prompt or terminal window and change directories to the one containing the UEDDAM control file. Then, from the control window, type “ueddam control.ucf”, where control.ucf is the name of the UEDDAM control file to be executed. Note that if the UEDDAM executable does not reside in the same directory as the control file, in order to execute UEDDAM without typing in the absolute (or relative) path to ueddams.exe, it must be properly set in the path environment variable (note that if the UEDDAM installation successfully completed, the path variable should already have been modified appropriately).

During a UEDDAM execution, the program reports its current status (i.e., the specific event and/or release angle being processed) in the control window. Below/adjacent to the status descriptor, a progress bar indicates the percent completion for that specific

event/release angle. Upon successful completion, the program will report the total UEDDAM run time.

6.3 TROUBLESHOOTING

In order to determine why an execution of UEDDAM might be failing, it is first important to know the names and contents of some of the more useful files generated by UEDDAM (when the debugging flag is set to 1 or 2). The three most important of these files include:

- O_COVART – This is the COVART printout file. It includes an echo of all COVART input data as well as any COVART error messages.
- UEDDAM Error File – See Section 5.3.1.5 for defining the UEDDAM error file.
- LOGIC.ERR – This is the mkwgtf.exe error file. It includes messages generated to provide the user assistance in tracking down input errors in the logic file.

The following are the most common error messages output by UEDDAM in the error file (see Section 7.7 for a description of the error file). For each error message listed, techniques to track down the error are given.

- **FATAL ERROR: Cannot open JTYPE (MV, PK) file for reading**
 - Solution: In the UEDDAM control file, verify that the path specified for the JTYPE (MV, PK) FILE is correct.
- **FATAL ERROR: Could not find one or more of the required files**
 - Solution: In the UEDDAM control file, verify that the paths specified for the various input files are correct.
- **FATAL ERROR: One or more events without debris**
 - Solution: In the UEDDAM control and debris files, verify that the event type and engine type are specified correctly.
- **FATAL ERROR: One or more events without locations**
 - Solution: In the UEDDAM control and debris files, verify that the engine section, engine ID, and rotor stage are specified correctly.
- **FATAL ERROR: No Component for COMPPK record**
 - Solution: In the JTYPE file, verify that any component with a COMPPK record also has a corresponding geometric representation in the model or COMALIAS record.
- **FATAL ERROR: Error in MV file**
 - Solution: The problem involves the MV FILE. If an O_COVART file was generated, check it for error messages.

For those times when UEDDAM terminates “ungracefully” (i.e., when a UEDDAM.exe – Application Error window appears), the problem most likely involves one of the COVART input files, as opposed to one of the UEDDAM-specific inputs. If an O_COVART file was generated, check it for error messages. Types of problems that may exist include, but are not limited, to the following: mistyped word; incorrect

spacing; a PK table ID used in the JTYPE file, but not defined in the PK file; use of an undefined material type in the JTYPE file; etc.

Note that when a failed run does occur, it is important that all output files (including any debugging folders and files) are deleted before a UEDDAM run is attempted again.

The UEDDAM visualizer can be used to check for non-fatal errors in various UEDDAM input files. The UEDDAM visualizer can display hazard zones, shotlines, probability plots, and risk angle plots all of which can help detect logic and/or typing errors. However, in order to access the full capability of these features, certain output files from a successful UEDDAM execution are required. To display hazard zones, for example, the VISUALIZER file is needed. To display shotlines, the debugging flag should be set to "1" in the control file and the VISUALIZER file is necessary. To display probability plots, the SUMMARY and DETAIL files are needed. Finally, to view risk angle plots, the INTERCEPT file is required. For more information on any of the visualizer's specific capabilities, see the UEDDAM Visualizer User Manual, included with the UEDDAM installation.

6.4 PRODUCTION RUNS

Upon completion of input file verification, production runs can begin. A Microsoft Excel spreadsheet (with embedded macro) and Java-based post-processing tool (discussed in Section 8) have been developed for the PC to aid in performing the production runs and post-processing for a specific type of UEDDAM assessment. This specific assessment consists of analyzing an aircraft's risk for catastrophic failure against one or more debris categories over multiple flight phases. The particular directory structure that the spreadsheet creates can be seen in Figure 6-2.

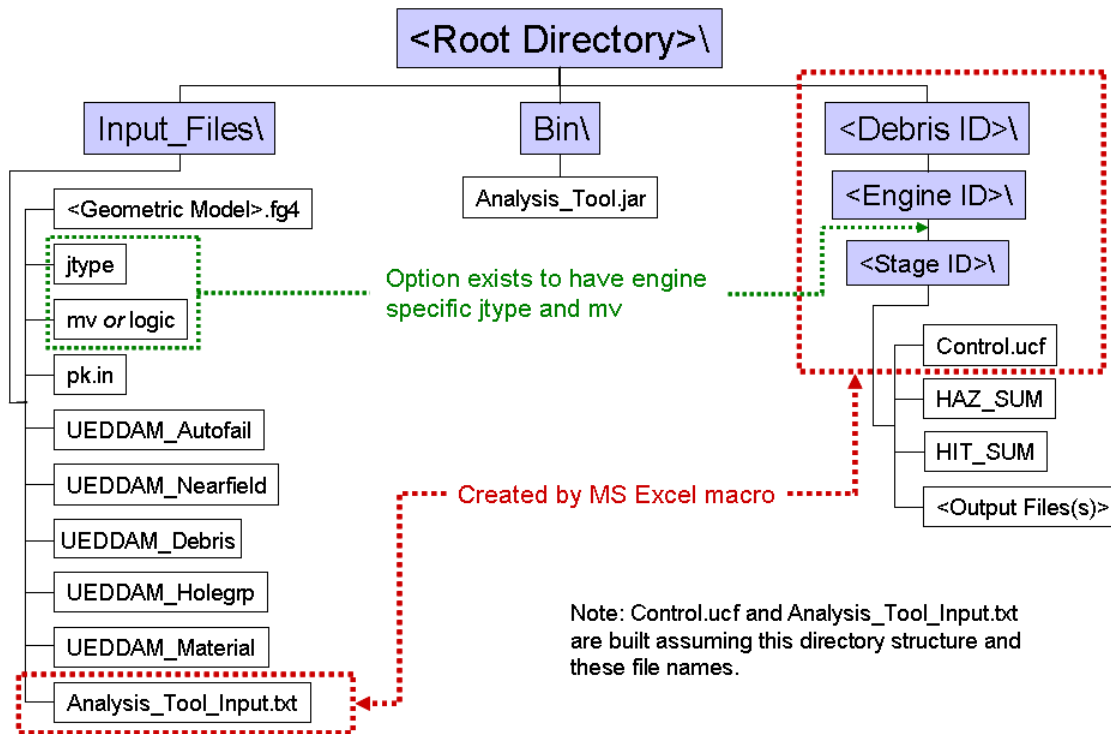


Figure 6-2. Required Directory Structure for Production Runs and Post-Processing.

The spreadsheet, titled Run_Setup.xls, is provided in the “tools” directory of the PC UEDDAM installation package. It assumes the utilization of the file structure detailed in Figure 6-2. The files and folders within the dashed red boxes are created by the Run_Setup.xls macro. Note that this file structure creates a unique control file for each EVENT record. If multiple EVENTS are specified in a single UEDDAM control file, each EVENT output is appended to the output from the first EVENT run. Thus, a single EVENT per control file was used to simplify the search algorithms used in the post-processing program (see Section 8). Upon execution, each control file specifies that the output files requested be written to the same folder as contains the control file.

The spreadsheet is composed of two parts: a run setup panel and input data tables. It is recommended that the user open the spreadsheet Run_Setup.xls for reference in the following sections. (For a sample illustrating use of the setup macro, see Section 9.11.)

6.4.1 Input Data Tables

Up to six debris types and associated descriptions can be input in the Debris Category Data Table. The entry beneath the last debris type entry should be the word “ALL”, which indicates to the embedded macro the end of the data list as well as allows the user to select all the debris types in the run setup panel. The macro assumes that a debris file named “UEDDAM_Debris” exists in the “Input_Files” directory and contains all debris types listed in the Debris Category Data Table.

Similar to the debris types, up to six engine IDs may be entered in the Engine Type Data Table with the word “ALL” in the row beneath the last engine ID entry. As with the debris categories, the “ALL” statement indicates to the macro the end of the data list as well as allows the user the option (in the run setup panel) to choose that all engines be evaluated in UEDDAM.

Up to 15 flight phases may be entered in the Flight Phase Data Table. The flight phase descriptions will be used as titles for the output data and should therefore not contain any spaces. The flight phase failure distribution values (Dp) assign a probability of a rotor burst event occurring in each phase of flight (see Table 4-1 for the failure distributions defined in the AC 20-128A). The sum of these probabilities should always equal 1.0. Note that although UEDDAM does not utilize the Dp values, they will be used by the post-processing program to calculate the full flight profile P_{HAZ} .

The executable location cells allow the user to change the path for the UEDDAM and UEDDAM-COVART executables. The location of the UEDDAM executable file is used in the batch file that the macro can create if specified. The entered UEDDAM-COVART executable file path will be specified within each of the control files.

The embedded macro assumes that all stages entered in the Engine Disk Centroid Location Data Table are rotating in the same direction and that each engine has the same stages. The centroid of each disk should be entered using the FASTGEN geometry coordinates.

6.4.2 Run Setup Panel

Once all the control file input data has been entered, the user can use the form at the top of the spreadsheet to specify options for control file generation. The debris type, engine name, and component name menus in the set-up panel are populated by the input data tables previously discussed. The user can select a single debris type, engine, and component combination, or if “ALL” is selected from the drop-down menu, a control file will be generated for each of the items in that menu.

The user can select the desired output files by clicking on the appropriate check boxes on the lower central portion of the panel. Note that if the check box for the HOLESIZE output file is selected, the macro assumes that a hole group input file named “UEDDAM_Holegrp” exists in the “Input_Files” directory. For the four output files that require additional information (i.e., ERROR, SIGITSYS, SIGITSHOLE, and SIGITCHOLE FILES), the macro will use the value entered in the adjacent input box to complete the control file entry. Due to the size of the output files, it is recommended that the DETAIL, INTERCEPT, VISUALIZER, TRAJECTORY, and ERROR files be deselected during production runs since the aforementioned files are primarily used for debugging and/or investigation of smaller data sets. The SUMMARY and HITSUM files do not appear in the list of available output files, as the embedded macro automatically includes them for output in the control files.

Both the JTYPE and MV/LOGIC files should be located in the “Input_Files” directory if they are the same for all engines being assessed. If, however, the engines have different JTYPE and MV/LOGIC files, copies of the appropriate files should be included in each “Engine ID” directory, rather than the “Input_Files” directory. Selecting the appropriate option (i.e., “UNIQUE FOR EACH ENGINE” or “SAME FOR EACH ENGINE”) indicates the location of the JTYPE and MV/LOGIC files.

Seven run options are available to the user: “DEBUGGING,” “TARGET REDUCTION,” “HYPER SAMPLING,” “PENETRATION,” “GENERATE BATCH FILE,” “EXECUTE ANALYSIS_TOOL,” and “THREADS.” If the “DEBUGGING” option is selected, a value of 1 will be entered for the debugging flag at the end of the EVENT record; otherwise, a value of 0 is entered. It is recommended that debugging not be used for production runs as the size of the output files created can be quite large. Selecting any of the “TARGET REDUCTION,” “HYPER SAMPLING,” “PENETRATION,” or “THREADS” options results in the corresponding entry to be written in the control file. Note that if the “HYPER SAMPLING,” “PENETRATION,” and/or “THREADS” options are selected, the embedded macro will use the values entered/selected for “Hypersampling Options,” “Gamma Values,” and/or “Multiple Processor Flag,” respectively, to complete the control file entry. The “GENERATE BATCH FILE” option writes a Microsoft Windows batch file called “UEDDAM_Runs.bat” to the root directory. This batch file contains an execution statement for each of the control files generated by the macro. Selecting the “EXECUTE ANALYSIS_TOOL” option adds the command to execute the post-processing program at the end of the batch file.

A file whose content is specified by the “Summary of Output” options on the lower right-hand side of the panel controls the data scanned by the post-processor, Analysis_Tool.jar. The “ALL OUTPUT FILES” option results in an input file that instructs the post-processing program to scan the folders for all possible combinations of debris type, engine ID, and component. The “NEW OUTPUT FILES” option will direct the post-processing program to search only those UEDDAM runs executed by the batch file. Note that if “ALL” is selected for the debris, engine, and component, the “ALL OUTPUT FILES” and “NEW OUTPUT FILES” options would result in the same summary tables. Finally, if “NONE” is selected, the post-processing program input file will not be created.

The macro is executed by clicking on the button labeled “Set Up UEDDAM Input Control Files.” Note that the macro does not actually execute the batch file if the “GENERATE BATCH FILE” option is selected. This file can be executed either by double clicking on it or calling it from a command prompt. The latter option is preferred since if an error occurs during a batch file executed by double clicking, the program window will close automatically and the user will be unable to determine the cause of failure.

7 UEDDAM OUTPUT

All but one of the output files from UEDDAM are in comma-separated text (CSV) format. Up to 12 output files may be created based upon the presence of the respective filename record in the UEDDAM control file. If no output files are specified in the control file, UEDDAM will by default output only event level summary information in a file named “HazSummary.”

7.1 EVENT LEVEL SUMMARY FILE (SUMMARY FILE)

The SUMMARY FILE is designed to give the user a quick and easy reference of event impact on the total system, based on release point. Each event is written as a separate summary within the SUMMARY FILE. The event summary sections are further subdivided into two parts.

The first part is the input review (header) section. This echoes event input data, including the EVENT, LOCATION, and LIMITS records used for the event. The second part is a descriptive table (data section) that shows P_{HAZ} of the total system for the event at specific release angles. The release angles will encompass the angles noted as “Start” and “Stop” in the header section. At each release angle, there is a P_{HAZ} given for each of the hazard levels the user specified in the UEDDAM control file. Additionally, for each event, UEDDAM prints out an average P_{HAZ} over all release points and an average P_{HAZ} over 360° (assuming the release angles not assessed have zero probability of hazard). Figure 7-1 provides an example of a SUMMARY FILE, for a single release point of 0°.

```

Event Identification:
, Event Type      , BLADE,
, Engine ID       , Engine1,
, Engine Type     , HighBypassRatio,
, Engine Section  , HPT,
, Rotor Stage    , Stagel,
, Engine Parameters:
,   , Rotation Direction , CCW
,   , x-value of centroid , -1367.70, in,
,   , y-value of centroid , 743.00, in,
,   , z-value of centroid , 183.80, in,
,   , Direction Vectors of Engine Axis:
,   ,   , Dx, 1.00
,   ,   , Dy, 0.00
,   ,   , Dz, 0.00

, Run Parameters:
,   , Start                , 0.00, deg,
,   , Size                 , 0.00, deg,
,   , Stop                 , 0.00, deg,
,   , # of Release Points  , 1,,

, Statistical Parameters for each release point:
,   , # of Iterations      , 1,,
,   , Random Number Seed   , 9679,,

,   , Debris Release Angle, Serious Damage, Significant Damage, Minor Damage,
,   ,                   0.00, 0.0000e+000, 1.0000e+000, 1.0000e+000,
,   ,                   Average, 0.0000e+000, 1.0000e+000, 1.0000e+000,
,   ,                   Average over 360 deg, 0.0000e+000, 2.7778e-003, 2.7778e-003,

```

Figure 7-1. Sample SUMMARY FILE.

7.2 COMPONENT LEVEL SUMMARY FILE (HITSUM FILE)

The HITSUM FILE gives the total average P_{haz} values of all iterations for each release point for each component as well as the P_{HAZ} for the overall system. The header echoes the respective EVENT record from the UEDDAM control input file and provides column headers for the P_{haz} levels. The subsequent lines identify the component and its P_{haz} contribution for each hazard level. The first component listed is for the entire aircraft (labeled “SYSTEM”) and is also printed in the SUMMARY FILE as the average over all release points for each event. Additionally, when more than one release point is considered, UEDDAM provides the total average P_{haz} values over 360° for each component (again, UEDDAM assumes that release angles not assessed result in a zero P_{haz}) as well as the average over all release angles assessed. Figure 7-2 provides an example of a HITSUM FILE for a single release point.

```

EVENT:, DISK, Engine1, HighBypassTurbofan, HPT, Stage1

Average ALL Iterations, 1 Release Points
,NAME           ,           1,           2,           3,           4,           5,
,SYSTEM          ,0.0000e+000,0.0000e+000,0.0000e+000,1.0000e+000,1.0000e+000,
,LeftP1         ,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,
,RightP1        ,0.0000e+000,0.0000e+000,0.0000e+000,8.7330e-001,8.7330e-001,
,RightP2        ,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,
,RightP3        ,0.0000e+000,0.0000e+000,0.0000e+000,1.0000e+000,1.0000e+000,
,RightP4        ,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,
,RightP5        ,0.0000e+000,0.0000e+000,0.0000e+000,9.9600e-001,9.9600e-001,
,RightP6        ,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,
,Engine         ,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,

```

Figure 7-2. Sample HITSUM FILE.

7.3 DETAILED HAZARD INFORMATION FILE (DETAIL FILE)

The **DETAIL FILE** produced by UEDDAM is similar to the **HITSUM FILE** except for the fact that it breaks out the component level P_{haz} results for the individual iterations for each release point. Following the last iteration for a release point, the average (over all iterations) system and component P_{haz} are printed for the completed release point. Note that UEDDAM eliminates spurious output data due to component redundancies by utilizing the built-in fault tree interrogator; UEDDAM only allows for the output of sub-systems and components with direct impact to the hazard. For example, consider the fault tree for the hydraulic system of a generic twin-engine aircraft considered in Figure 4-3 and in Section 5.2.3. UEDDAM, instead of printing results for each individual component in the fault tree, would only print results for **HYD_SYS**. Figure 7-3 provides an example of a **DETAIL FILE** for a single-iteration assessment. It is recommended that this file only be generated for small runs.

```

EVENT:, DISK, Engine1, HighBypassTurbofan, HPT, Stage1

Iteration 1, Release Point 1:
,NAME           ,           1,           2,           3,           4,           5,
,SYSTEM          ,0.0000e+000,0.0000e+000,0.0000e+000,1.0000e+000,1.0000e+000,
,LeftP1         ,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,
,RightP1        ,0.0000e+000,0.0000e+000,0.0000e+000,8.7330e-001,8.7330e-001,
,RightP2        ,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,
,RightP3        ,0.0000e+000,0.0000e+000,0.0000e+000,1.0000e+000,1.0000e+000,
,RightP4        ,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,
,RightP5        ,0.0000e+000,0.0000e+000,0.0000e+000,9.9600e-001,9.9600e-001,
,RightP6        ,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,
,Engine         ,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,

Average Release Point 1:
,NAME           ,           1,           2,           3,           4,           5,
,SYSTEM          ,0.0000e+000,0.0000e+000,0.0000e+000,1.0000e+000,1.0000e+000,
,LeftP1         ,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,
,RightP1        ,0.0000e+000,0.0000e+000,0.0000e+000,8.7330e-001,8.7330e-001,
,RightP2        ,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,
,RightP3        ,0.0000e+000,0.0000e+000,0.0000e+000,1.0000e+000,1.0000e+000,
,RightP4        ,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,
,RightP5        ,0.0000e+000,0.0000e+000,0.0000e+000,9.9600e-001,9.9600e-001,
,RightP6        ,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,
,Engine         ,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,0.0000e+000,

```

Figure 7-3. Sample DETAIL FILE.

7.4 TRANSLATIONAL RISK ANGLE FILE (INTERCEPT FILE)

The output in the INTERCEPT FILE duplicates the results of the manual assessment process discussed in the AC 20-128A [8]. It provides the translational rotation angles, the fore and aft spread angles, and the probability of hit P_h for each individual component. The header records of the output file echo the respective EVENT entered and provide column headers for the data.

The column headers are defined according to the following information:

- Minimum rotational angle (MIN(t)).
- Maximum rotational angle (MAX(t)).
- The percent of the rotational angle window that the component encompasses ($P_h(t)$). It is computed as: $P_h(t) = (\text{Max}(t) - \text{Min}(t)) / 360.0$
- The maximum fore angle of the component (FORE(s)).
- The maximum aft angle of the component (AFT(s)).
- The percent of the fore/aft spread angle window that the component encompasses ($P_h(s)$). It is computed as: $P_h(s) = (\text{FORE}(s) - \text{AFT}(s)) / (\text{Fwd Angle} - \text{Aft Angle})$
- The subcomponent P_h is computed as $P_h(t)$ times $P_h(s)$.

Finally, the name of critical components and assemblies (as defined in the JTYPE FILE), the respective component numbers that comprise the JTYPE name, and the intercept angles for each component are printed. Figure 7-4 illustrates a simple INTERCEPT FILE where each JTYPE name represents a single component.

EVENT:	DISK, Engine1, HighBypassTurbofan, HPT, Stage1						
	MIN(t),	MAX(t),	Ph(t),	FORE(s),	AFT(s),	Ph(s),	Ph,
LeftP1,	Component Number: 1,						
	131.051,	265.486,	0.373,	5.000,	-5.000,	1.000,	0.373,
RightP1,	Component Number: 2,						
	357.725,	2.306,	0.013,	-2.231,	-5.000,	0.277,	0.004,
RightP2,	Component Number: 3,						
	,	,	,	,	,	,	,
RightP3,	Component Number: 4,						
	357.167,	2.893,	0.016,	2.928,	-2.928,	0.586,	0.009,
RightP4,	Component Number: 5,						
	,	,	,	,	,	,	,
RightP5,	Component Number: 6,						
	357.725,	2.306,	0.013,	5.000,	2.231,	0.277,	0.004,
RightP6,	Component Number: 7,						
	8.165,	12.887,	0.013,	5.000,	-5.000,	1.000,	0.013,
Engine,	Component Number: 1001,						

Figure 7-4. Sample INTERCEPT FILE.

7.5 TRAJECTORY FILE

The output in the TRAJECTORY FILE relays information about individual shotline data to the user. The output clearly displays the different release trajectories with respect to nominal and delta release angles, fragment weight and yaw, fore/aft trajectories, etc. The trajectory output is broken into three parts.

The first part is similar to the first part of the SUMMARY FILE in that it echoes the input event data for the assessment. The second part is a reformatted display of the entered debris characterization for the given event. The third is a detailed table of information created from the individual shotlines used to simulate the fragment types of the event. The text version of this file can be difficult to interpret, and it is recommended that this file be viewed using any available spreadsheet software. Once in a spreadsheet, the data contained is self-explanatory. Also, because this file outputs detailed data for each and every ray generated during a UEDDAM run, it can become extremely large. It is recommended that this file only be generated for small runs.

7.6 HOLE SIZE FILE (HOLESIZE FILE)

The HOLESIZE FILE contains hole size statistics for the components and groups as defined in the HOLEGRP FILE. For each group of components, UEDDAM will output the minimum, maximum, and average hole areas (in square inches) averaged over iteration for every release angle. Additionally, this output file includes the average number of fragments that hit and the average number of fragments that penetrate each group of components (averaged over iteration) for every release angle. The header records of the HOLESIZE FILE echo the component groupings as entered in the HOLEGRP FILE, echo the respective EVENT entered, and provide column headers for the data. Figure 7-5 illustrates a simple HOLESIZE FILE for three release angles of an LFRAG event.

```

Group Information:
, Port_Skin, 201, 202, 203, 204, 213, 214,
, Stbd_Skin, 251, 252, 253, 254, 263, 264,
, Port_Wing, 101, 102, 103, 104, 105, 106,
, , 107, 108, 109, 110,
, Stbd_Wing, 111, 112, 113, 114, 115, 116,
, , 117, 118, 119, 120,

EVENT:, LFRAG, Engine1, HighBypassRatio, Fan, Stage1

Release Point 1:, Angle 350.00,
,Group Name, Maximum Hole Area, Minimum Hole Area, Average Hole Area, Fragment Hits,
Fragment Penetrations,
,Port_Skin, 23.74, 0.00, 10.01, 1.40,
1.40,
,Stbd_Skin, 24.37, 0.00, 10.26, 1.60,
1.60,
,Port_Wing, 114.68, 9.84, 52.13, 2.80,
2.80,
,Stbd_Wing, 0.00, 0.00, 0.00, 0.00,
0.00,

Release Point 2:, Angle 0.00,
,Group Name, Maximum Hole Area, Minimum Hole Area, Average Hole Area, Fragment Hits,
Fragment Penetrations,
,Port_Skin, 27.75, 9.41, 17.06, 2.00,
2.00,
,Stbd_Skin, 18.52, 0.00, 10.75, 1.20,
1.20,
,Port_Wing, 73.30, 41.06, 53.58, 3.80,
3.80,
,Stbd_Wing, 33.36, 0.00, 20.29, 1.60,
1.60,

Release Point 3:, Angle 10.00,
,Group Name, Maximum Hole Area, Minimum Hole Area, Average Hole Area, Fragment Hits,
Fragment Penetrations,
,Port_Skin, 99.21, 75.34, 87.50, 6.40,
6.40,
,Stbd_Skin, 0.00, 0.00, 0.00, 0.00,
0.00,
,Port_Wing, 0.00, 0.00, 0.00, 0.00,
0.00,
,Stbd_Wing, 0.00, 0.00, 0.00, 0.00,
0.00,

```

Figure 7-5. Sample HOLESIZE FILE.

7.7 SIGNIFICANT ITERATION SUMMARY FILES

Because for large runs, the entire output can be too large to be retained (or simply, too overwhelming to sift through), the user has the option to specify that significant iteration summary files be generated. These files give the user the ability to focus on those particular release points and iterations that result in damage above a certain threshold. The output can be summarized according to four different criteria: total system-level P_{HAZ} , critical components, single impact hole size, or cumulative hole size. The header portion of each output file is an echo of the EVENT record, details of the specific release points evaluated for that event, and an echo of the user-input evaluation criteria. What follows are the release point significant iterations.

7.7.1 Summary File (SIGITSYS FILE)

The SIGITSYS FILE contains information detailing iterations whose total system-level P_{HAZ} exceeds the user-input value in the UEDDAM control file. For each release angle, those iterations that have at least one hazard level meeting the criterion are enumerated. Following the iteration identification number is the hazard level ID and associated P_{HAZ} value for each of the hazard levels with a total system-level P_{HAZ} exceeding the user-input value. Figure 7-6 illustrates a simple SIGITSYS FILE for three release points (five iterations each) of a multiple fragment event.

```
EVENT:, MULTI, Engine1, HighBypassRatio, Fan, Stage1
, , Start , 350.00, deg,
, , Size , 20.00, deg,
, , Stop , 10.00, deg,
, , # of Release Points , 3,,
, , Random Number Seed , 9627,,

Criteria:, 1.00, P_HAZ
Debris Release Angle:, 350.00
Number of Iterations:, 5

Iteration #, Hazard Level,P_HAZ, Hazard Level,P_HAZ, Hazard Level,P_HAZ,
1, 1,1.00 ,
2, 2,1.00 ,
3, 2,1.00 ,
4, 1,1.00 ,
5, 3,1.00 ,
Number of Significant Iterations:, 5

Criteria:, 1.00, P_HAZ
Debris Release Angle:, 0.00
Number of Iterations:, 5

Iteration #, Hazard Level,P_HAZ, Hazard Level,P_HAZ, Hazard Level,P_HAZ,
1, 1,1.00 , 2,1.00 , 3,1.00 ,
3, 1,1.00 , 2,1.00 , 3,1.00 ,
4, 1,1.00 ,
5, 1,1.00 ,
Number of Significant Iterations:, 4

Criteria:, 1.00, P_HAZ
Debris Release Angle:, 10.00
Number of Iterations:, 5

Iteration #, Hazard Level,P_HAZ, Hazard Level,P_HAZ, Hazard Level,P_HAZ,
1, 1,1.00 , 3,1.00 ,
2, 2,1.00 ,
5, 1,1.00 ,
Number of Significant Iterations:, 3
```

Figure 7-6. Sample SIGITSYS FILE.

7.7.2 Critical Component Hit Summary File (SIGITCRIT FILE)

The SIGITCRIT FILE contains information detailing iterations where the uncontained engine debris impacts one or more critical components. For each release angle, those iterations that have at least one hazard level meeting the criterion are enumerated. Following the iteration identification number is the hazard level ID and associated

number of critical components hit for each of the hazard levels with one or more hit on a critical component. Figure 7-7 illustrates a simple SIGITCRIT FILE for three release points (five iterations each) of a multiple fragment event.

```

EVENT:, MULTI, Engine1, HighBypassRatio, Fan, Stage1

, , Start , 350.00, deg,
, , Size , 20.00, deg,
, , Stop , 10.00, deg,
, , # of Release Points , 3,,
, , Random Number Seed , 9627,,

Debris Release Angle:, 350.00
Number of Iterations:, 5

Iteration #, Hazard Level,# of Crits Hit, Hazard Level,# of Crits Hit, Hazard Level,#
of Crits Hit,
, 1, 1,12 , 2,2 , 3,5
,
, 2, 1,12 , 2,2 , 3,2
,
, 3, 1,15 , 2,4 , 3,5
,
, 4, 1,11 , 2,1 , 3,4
,
, 5, 1,2 , 2,1 , 3,1
,
Number of Significant Iterations:, 5

Debris Release Angle:, 0.00
Number of Iterations:, 5

Iteration #, Hazard Level,# of Crits Hit, Hazard Level,# of Crits Hit, Hazard Level,#
of Crits Hit,
, 1, 1,13 , 2,4 ,
, 3, 1,19 , 2,8 , 3,12
,
, 4, 1,12 , 3,5 ,
Number of Significant Iterations:, 3

Debris Release Angle:, 10.00
Number of Iterations:, 5

Iteration #, Hazard Level,# of Crits Hit, Hazard Level,# of Crits Hit, Hazard Level,#
of Crits Hit,
, 1, 1,11 , 2,3 , 3,7
,
Number of Significant Iterations:, 1

```

Figure 7-7. Sample SIGITCRIT FILE.

7.7.3 Cumulative Hole Area Summary File (SIGITCHOLE FILE)

The SIGITCHOLE FILE contains information detailing iterations where fragment penetrations result in a cumulative hole size exceeding the user-input value in the UEDDAM control file. For each release angle, those iterations that meet the criterion are enumerated. Following the iteration identification number is the cumulative hole size in square inches. Figure 7-8 illustrates a simple SIGITCHOLE FILE for three release points (five iterations each) of a multiple fragment event.

```

EVENT:, MULTI, Engine1, HighBypassRatio, Fan, Stage1
, , Start , 350.00, deg,
, , Size , 20.00, deg,
, , Stop , 10.00, deg,
, , # of Release Points , 3,,
, , Random Number Seed , 9627,,

Criteria:, 43.00, in^2
Debris Release Angle:, 350.00
Number of Iterations:, 5

Iteration #, Cumulative Holedsize
1, 43.11
2, 162.79
4, 82.24
Number of Significant Iterations:, 3

Criteria:, 43.00, in^2
Debris Release Angle:, 0.00
Number of Iterations:, 5

Iteration #, Cumulative Holedsize
1, 82.23
2, 115.58
3, 119.72
4, 106.65
5, 84.20
Number of Significant Iterations:, 5

Criteria:, 43.00, in^2
Debris Release Angle:, 10.00
Number of Iterations:, 5

Iteration #, Cumulative Holedsize
3, 98.84
5, 75.34
Number of Significant Iterations:, 2

```

Figure 7-8. Sample SIGITCHOLE FILE.

7.7.4 Hole Size Summary File (SIGITSHOLE FILE)

The SIGITSHOLE FILE contains information detailing iterations where fragment penetrations result in a single hole size exceeding the user-input value in the UEDDAM control file. The output contained in this file is identical to that in the SIGITCHOLE FILE, but the data represents the single largest hole size generated by fragment penetrations, as opposed to the cumulative.

7.8 VISUALIZER FILE

The VISUALIZER FILE is a non-human-readable binary output file used solely by the UEDDAM visualizer (provided as part of the UEDDAM installation). The output file provides the visualizer with the necessary information to render the trajectories and trajectory limit information. Because this file contains information only useful to the visualizer, no discussion of its contents is included here (for additional information,

please see the UEDDAM Visualizer User Manual, provided as part of the UEDDAM installation). Note that it is recommended that this file only be generated for small runs.

7.9 ERROR FILE

The ERROR FILE contains the warnings and error messages as output by UEDDAM to the level of detail requested by the user (e.g., level 1 = fatal errors only). Generation of this file is recommended when in the “Verify” phase of the UEDDAM process as it can help the user identify where problems are occurring. It is important to note that a level 3 ERROR FILE contains a copious amount of benign warnings that can safely be ignored by the user. As an example, Table 7-1 gives warnings that are typically reported for an assessment and that can be safely ignored.

Table 7-1. UEDDAM Warnings that Can Safely Be Ignored

UEDDAM WARNING	NOTES
WARNING: NEGATIVE AIRGAP, RESET TO 0.0, FOR THESE TWO LOS':	
WARNING: Unused Debris, will be removed:	UEDDAM does not remove data from the debris file created by the user, but rather from an intermediate file created during execution if it is not needed for the particular event being assessed.
WARNING: No Component (L_ENG) to add data to.	
WARNING: Component number 7918: Not in CBULK file.	If component 7918 is an alias component, this warning can safely be ignored. If not, there might be an error in either the JTYPE or geometry input files. Note that UEDDAM will run to completion in either case.
WARNING: ..\..\input_files\DEBRIS1(6) -- S-FRAGMENT record SIGMA value breaks practical limits of 0 to 90.	
WARNING: ASIZE for EVENT greater than or equal to 360 deg. Changed to 337.5. Decreased number of Increments to 15.	

UEDDAM errors signal the user that a problem with one or several of the inputs exists. The errors reported by UEDDAM are not necessarily fatal. (Fatal errors are discussed in

Section 6.3). For example, the following is a non-fatal error that may arise out of an assessment and would be reported in a level 2 or level 3 ERROR FILE:

- UEDDAM ERROR: There are duplicate element id numbers(1) within a component.
 - Note that UEDDAM, in this case, will internally renumber the elements and therefore will execute correctly, but the user should correct the error for future assessments.

8 POST-PROCESSING

At this point in the UEDDAM analysis process, all input files have been created and verified and all runs have been successfully completed. All that remains is to post process and analyze the results. This section will focus on post-processing the results from the particular production runs discussed in Section 6.4 (i.e., analyzing an aircraft's risk for catastrophic failure against one or more debris categories over multiple phases of flight).

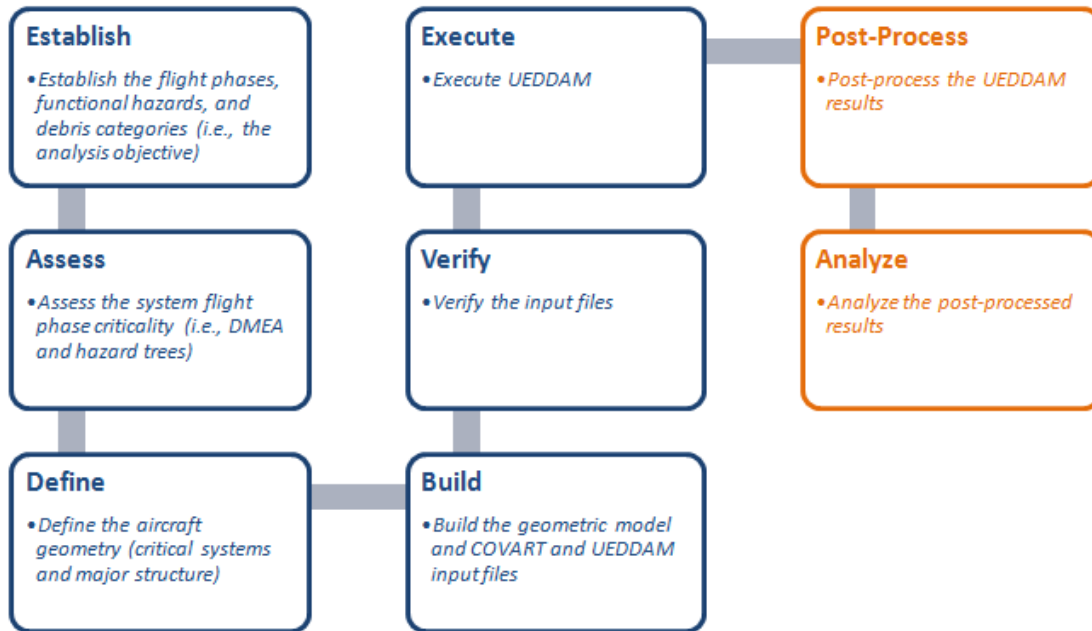


Figure 8-1. Current Location in the UEDDAM Analysis Process: Post-Processing.

UEDDAM, upon successful completion, supplies the user with information that can be used to show compliance with the AC 20-128A requirements (and more). Recall that UEDDAM and its post processors accomplish steps 3) through 5) of the AC 20-128A process, as outlined in Section 2.1. Steps 3) through 5) can be more explicitly written as:

- a) For each rotor stage, establish the threat windows (i.e., the translational and spread risk angles) where, due to a combination of individual damages, a catastrophic risk exists.
- b) For each rotor stage, calculate the risk factor for all critical hazards.
- c) For each rotor stage, calculate the combined risk factor (C_{sp}) for the phase of flight and average over 360° .
- d) Calculate the overall rotor stage risk (C_s) for each stage over all n flight phases (incorporating D_p).
 - $C_s = (D_{p1} \times C_{sp,1}) + (D_{p2} \times C_{sp,2}) + \dots + (D_{pn} \times C_{sp,n})$
- e) Average all m rotor stages to obtain the mean engine risk (C_E).
 - $C_E = (C_{s1} + C_{s2} + \dots + C_{sm}) / m$
- f) Average all k engines to obtain the mean aircraft risk (C_A).
 - $C_A = (C_{E1} + C_{E2} + \dots + C_{Ek}) / k$

UEDDAM internally performs the calculations in steps a) thru c); in particular, C_{sp} can be taken directly from the SUMMARY FILE (see Section 7.1). Thru simple post-processing (i.e., applying the phase failure distributions [Dp] identified in Table 4-1 and then performing a series of averages), steps d) thru f) can be calculated, resulting in a mean aircraft risk for the particular debris category of interest.

As an example, consider the analysis of a generic twin-engine aircraft's risk for catastrophic failure against a single one-third disk segment over all 7 phases of flight. Each engine of the generic twin has 18 rotor stages (i.e., fan, low pressure compressor stages 1 thru 3, high pressure compressor stages 1 thru 9, high pressure turbine, and low pressure turbine stages 1 thru 4). For each of the 36 engine rotor stages (18 rotor stages per engine and two engines), a separate UEDDAM execution is performed using the debris category DISK (a single one-third disk segment). These runs result in $7 \times 18 \times 2 = 252$ C_{sp} values in 36 separate UEDDAM SUMMARY FILES. For a particular stage and engine (i.e., engine 1, fan stage), $C_s = (C_{sp1} \times Dp_1) + (C_{sp2} \times Dp_2) + \dots + (C_{sp7} \times Dp_7)$. This calculation directly leads to the mean engine risk for a single engine (i.e., engine 1) $C_E = (C_{s1} + C_{s2} + \dots + C_{s18}) / 18$ and therefore the mean aircraft risk $C_A = (C_{E1} + C_{E2}) / 2$.

Since it is equally important to know the various system and component contributions to aircraft risk, a Java-based post-processing tool called Analysis_Tool.jar has been developed for the PC by NAWCWD that summarizes and provides averages for multiple UEDDAM output files. As in the case of the UEDDAM control file, the input to Analysis_Tool.jar (which is called Analysis_Tool_Input.txt) follows a {Keyword}, {Data} format, as follows.

Keyword	Entries
DEBRIS,	Debris category(ies) being assessed; names correspond to Event Type in the disk debris mode record in the DEBRIS FILE and in the EVENT record in the control file
ENGINES,	Engine(s) being assessed; names correspond to Engine ID in the EVENT, LOCATION, and LIMITS records in the control file
ROTORS,	Rotor stage(s) being assessed
FLIGHT_PHASES,	Flight phase(s) being assessed; names correspond to hazard level specification description in the control file
FLIGHT_PHASE_WEIGHTS,	Corresponding flight phase failure distribution value(s)

For the example discussed previously, Analysis_Tool_Input.txt would be as shown in Figure 8-2.

```

DEBRIS, DISK
ENGINES, ENGINE1, ENGINE2
ROTORS, Fan, LPC-1, LPC-2, LPC-3, HPC-1, HPC-2, HPC-3, HPC-4, HPC-5, HPC-6, HPC-7, HPC-8, HPC-9, HPT, LPT-1, LPT-2, LPT-3, LPT-4
FLIGHT_PHASES, T/O_to_V1, V1_to_V1+30, Initial_Climb, Cruise, Descent, Approach, Landing
FLIGHT_PHASE_WEIGHTS, 0.35, 0.2, 0.22, 0.14, 0.3, 0.2, 0.4

```

Figure 8-2. Analysis_Tool_Input.txt Example.

Note that the input file for the post-processing tool is automatically generated when the “summary of output” option is selected in the Microsoft Excel spreadsheet, used to create the directory structure and necessary UEDDAM control files for production runs (see Section 6.4). Thus, the same directory structure, executable locations, and file names (as illustrated in Figure 6-2) are expected in order for the post processor to execute successfully.

The executable for the post processor is provided in the “tools” directory of the PC UEDDAM installation package. Upon successful execution, Analysis_Tool.jar provides the user with five summary output files:

- *PH_Sum.txt* (located in the <Root Directory>) – provides an overall summary of the results to include the **combined risk factor** by category of debris, engine, rotor section, and flight phase; **overall stage risk** by category of debris, engine, and rotor section; **mean engine risk** by category of debris and engine; and **mean aircraft risk** by category of debris.
- *Combined_Debris.csv* (located in the <Root Directory>) – provides an average P_{haz} value of each contributing component for the various debris categories.
- *Combined_Engine.csv* (located in every <Debris ID> directory) – provides an average P_{haz} value of each contributing component for the various engines.
- *Combined_Rotor.csv* (located in every <Engine ID> directory) – provides an average P_{haz} value of each contributing component for the various rotor sections.
- *Combined_Flight_Phase.csv* (located in every <Stage ID> directory) – provides an average P_{haz} value of each contributing component for the various flight phases.

The last four of these files are in comma separated text format that can be read into a spreadsheet program for manipulation and plotting purposes. (For a sample illustrating use of the post-processor, see Section 9.11.)

9 SAMPLE CASES

Fourteen sample cases have been developed to allow the user to become familiar with UEDDAM and its capabilities. The first three samples involve small fragments, large fragments, and segments applied to a simple geometry. The fourth, fifth, and sixth samples each exercise UEDDAM's treatment of more complicated geometrical descriptions (i.e., a partial section of a representative business jet and a generic twin-engine aircraft) as well as more complicated debris characterizations. The next four sample cases were developed as part of the UEDDAM v3.0 development effort; they exercise each of the additional features added as part of the software upgrade. The final four sample cases were developed to illustrate use of the setup macro and post-processing tool; how to perform a UEDDAM assessment using the conservative assumptions described in the appendix to the AC 20-128A (i.e., infinite energy disk); use of the PENETRATION keyword added as part of UEDDAM v4.1 development; and multi-segment analysis options added as part of UEDDAM v5.0 development. Actual inputs and results for these sample cases are provided in the distribution. Descriptions of the individual samples are given in the following text.

To execute UEDDAM with the sample cases:

- The "bin" directory must be set in the path system variable (if the UEDDAM installation was completed successfully, this will have already been done).
- Change directories to the directory where the UEDDAM control file is for the respective sample case.
- Type the command "ueddam control.ucf", where control.ucf is the name of the UEDDAM control file for the particular sample case to be run.

9.1 SAMPLE CASE ONE – S-FRAGMENT

Sample case one demonstrates UEDDAM's ability to discern which components are susceptible to hits. Only small fragment debris is considered for this case. A dummy engine (cylinder) was constructed and then surrounded by plates at various spread angle locations as shown in Figure 9-1. The cylinder representing the engine geometry was included in the NEARFIELD FILE to allow the fragments to exit the dummy engine unimpeded.

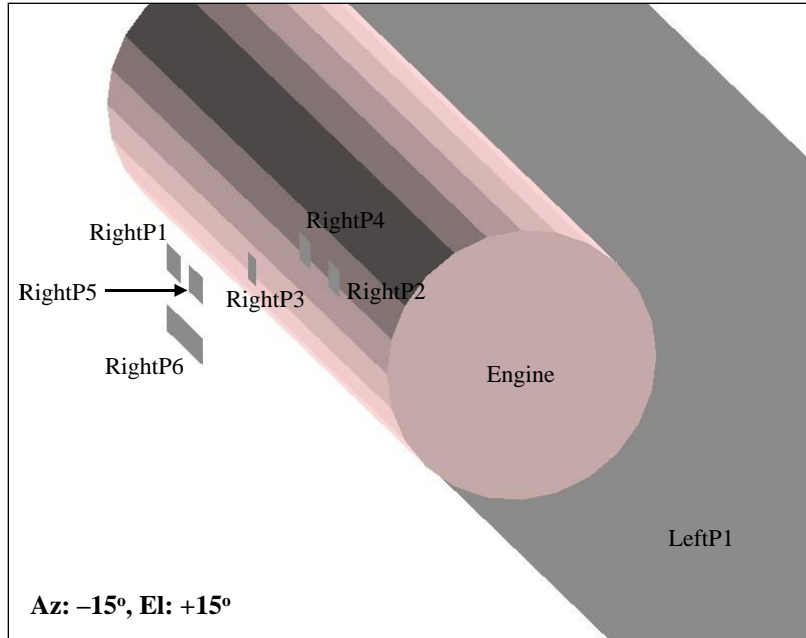


Figure 9-1. Simple Geometry for Sample Cases One Thru Four.

The debris for this sample case consists of small fragments and assesses a single release point at 0° rotation (a single iteration is performed about that point). The rotational release variance is 0° (illustrated in Figure 9-2) and the fragment fore/aft spread angle limits are $\pm 5^\circ$ (illustrated in Figure 9-3).

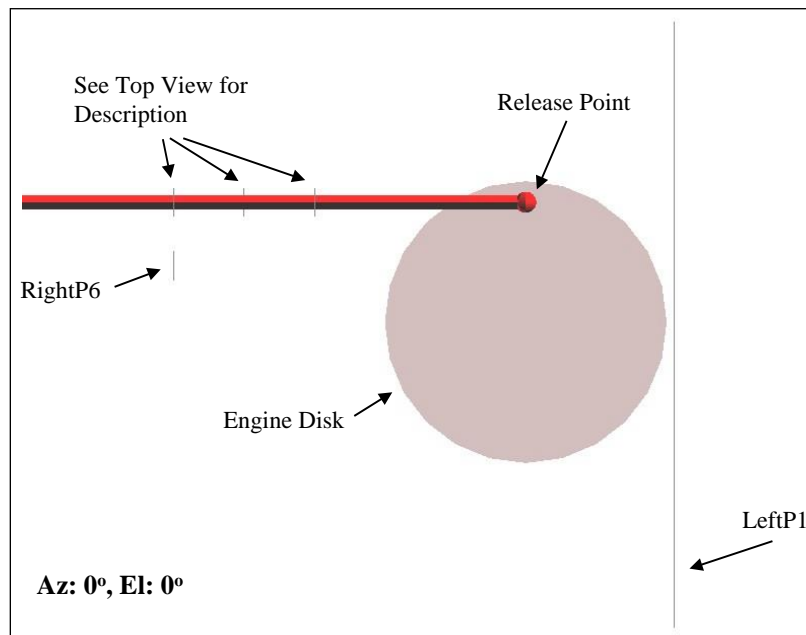


Figure 9-2. Front View for Sample Case One.

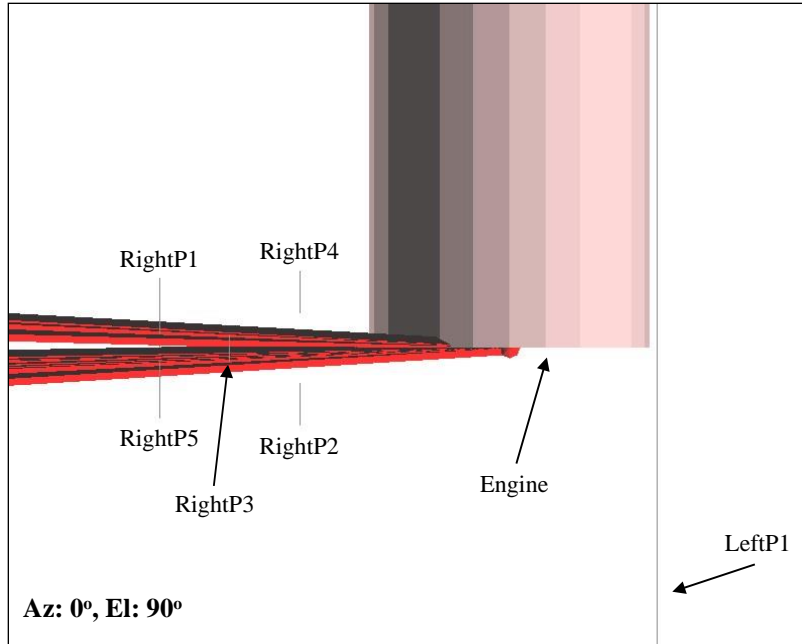


Figure 9-3. Top View for Sample Case One.

In order to verify that UEDDAM was correctly assessing viability, a manual assessment was performed to determine the susceptibility to hits (based on Figure 9-2 and Figure 9-3). A component is within the spread angle of the small fragments if at any rotational angle about the engine, the component lies no more than 5° either fore or aft of the stage releasing the debris. A component is within the translational angle of the small fragments if the component's extent encompasses 0° (the rotational release of the small fragments). Note that if a component does not lie within the fore/aft spread angle of the small fragments, it is not necessary to determine whether it lies within the debris' translational angle. Figure 9-4 and Table 9-1 show the results from the UEDDAM execution and the manual analysis, respectively. Note that the fore/aft spread and translational risk angle results from the UEDDAM execution of sample one provide more detail than those seen in Table 9-1, but are in agreement with the manually assessed results.

Table 9-1. Manual Analysis Results for Sample Case One

Component	Within Spread Angle?	Within Translational Angle?
Plate 1 (LeftP1)	Yes	No
Plate 2 (RightP1)	Yes	Yes
Plate 3 (RightP2)	No	N/A
Plate 4 (RightP3)	Yes	Yes
Plate 5 (RightP4)	No	N/A
Plate 6 (RightP5)	Yes	Yes
Plate 7 (RightP6)	Yes	No

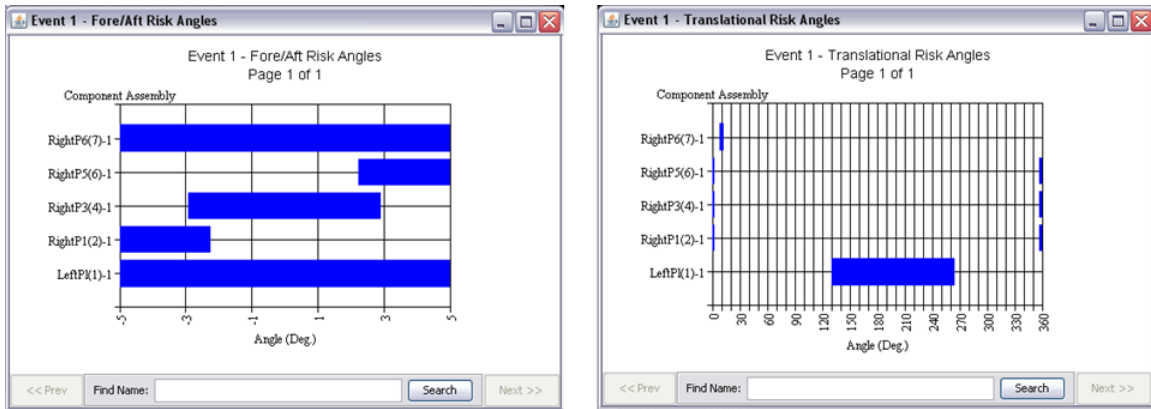


Figure 9-4. UEDDAM Spread and Translational Risk Angles for Sample Case One.

9.2 SAMPLE CASE TWO – L-FRAGMENT

Sample case two repeats sample case one using the large fragment debris type. This sample only assesses one release point at 0° rotation and only one iteration about that point. The spread and variance limits are the same as in the first sample. The primary difference in sample two is the addition of the potentially susceptible area cut by the large fragment swath. Figure 9-5 illustrates the swath that is associated with the large fragments. Note that Figure 9-1 and Figure 9-3 also apply.

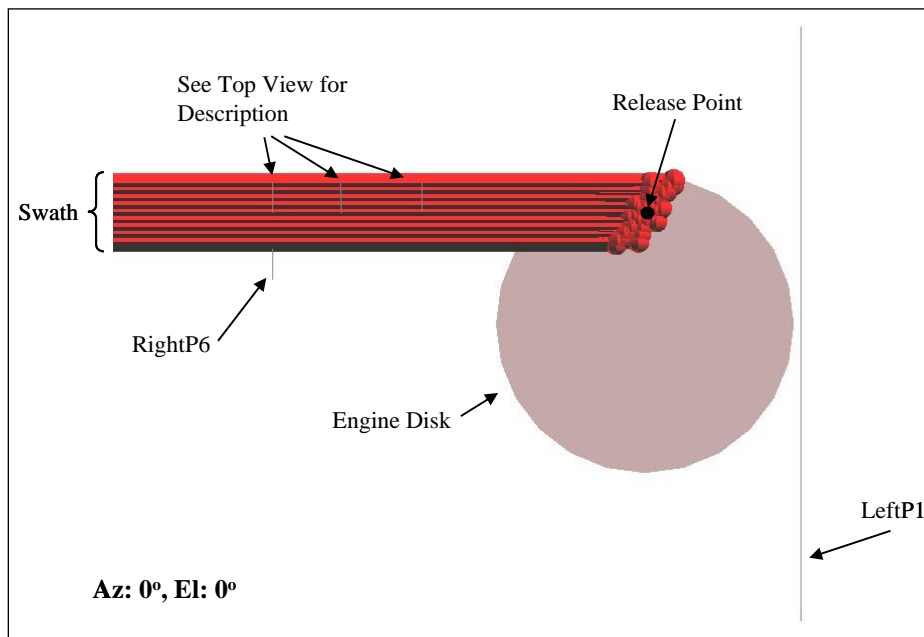


Figure 9-5. Front View for Sample Case Two.

Manually assessed and UEDDAM execution hit susceptibility results are shown in Table 9-2 and Figure 9-6, respectively. It is interesting to note that the HITSUM FILE shows variances from sample case one in component contributions to P_{HAZ} due to

changes in debris type. However, the overall system level UEDDAM results are identical to those in the first case.

Table 9-2. Manual Analysis Results for Sample Case Two

Component	Within Spread Angle?	Within Translational Angle?
Plate 1 (LeftP1)	Yes	No
Plate 2 (RightP1)	Yes	Yes
Plate 3 (RightP2)	No	N/A
Plate 4 (RightP3)	Yes	Yes
Plate 5 (RightP4)	No	N/A
Plate 6 (RightP5)	Yes	Yes
Plate 7 (RightP6)	Yes	Yes

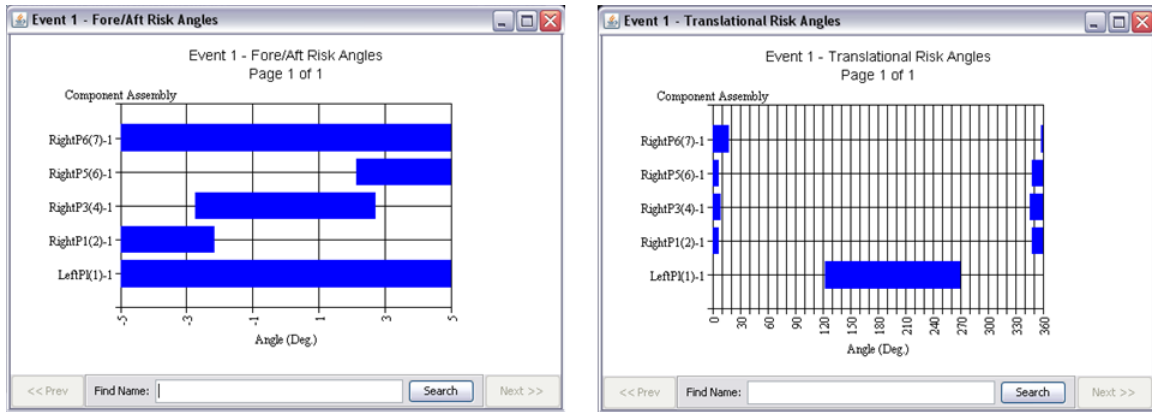


Figure 9-6. UEDDAM Spread and Translational Risk Angles for Sample Case Two.

9.3 SAMPLE CASE THREE – SEGMENT

Sample case three duplicates cases one and two with the only change being the use of the segment debris type rather than the small or large fragment. The segments were released at three rotational angles within 0° rotational release variance and with a spread angle variance of 5° fore and aft. The three rotational release angles were 0°, 120°, and 240° as shown in Figure 9-7. A single iteration was performed.

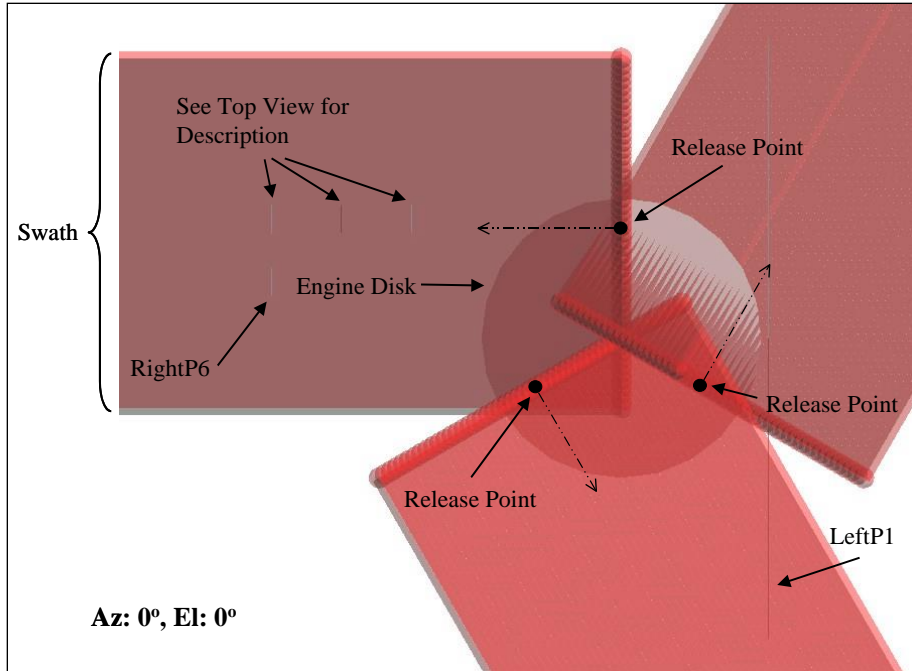


Figure 9-7. Front View for Sample Case Three.

Manually assessed and UEDDAM execution hit susceptibility results for each plate are shown in Table 9-3 and Figure 9-8, respectively.

Table 9-3. Manual Analysis Results for Sample Case Three

Component	Within Spread Angle?	Within Translational Angle?
Plate 1 (LeftP1)	Yes	Yes
Plate 2 (RightP1)	Yes	Yes
Plate 3 (RightP2)	No	N/A
Plate 4 (RightP3)	Yes	Yes
Plate 5 (RightP4)	No	N/A
Plate 6 (RightP5)	Yes	Yes
Plate 7 (RightP6)	Yes	Yes

In this case, five of the seven plates are hit. Two plates on the right (namely RightP2 and RightP4) are not hit due to the fact that they lie outside the spread limits of the segment (see Figure 9-3).

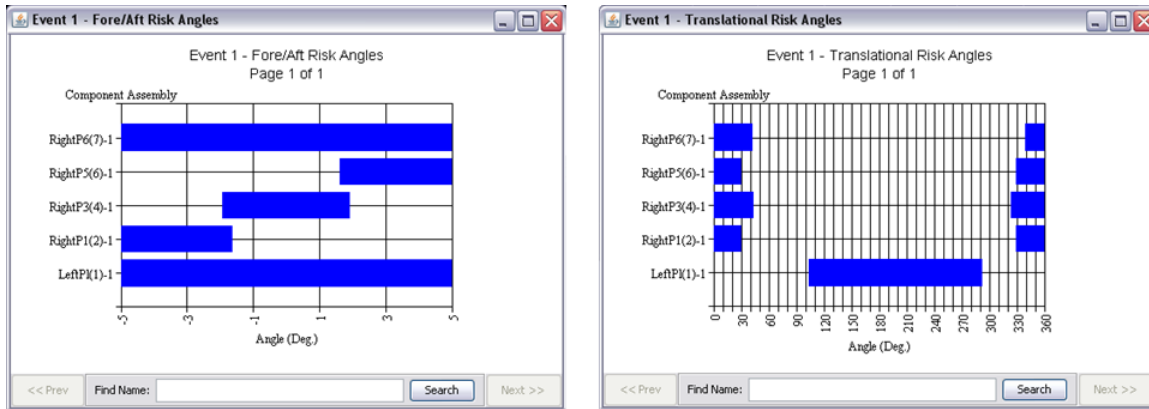


Figure 9-8. UEDDAM Spread and Translational Risk Angles for Sample Case Three.

Notice that just as for the large fragment in sample two, the segment debris in sample three hits plate RightP6, whereas the small fragment debris in sample one does not. This illustrates that while RightP6 is outside the rotational variance, it is within the swath dimensions of the disk segment and large fragment.

9.4 SAMPLE CASE FOUR – SIGNIFICANT ITERATIONS

This sample shows a variety of functions implemented in UEDDAM. The geometry considered is a partial section of a representative business jet. The sample control file implements a single event (MULTI) consisting of both large and small fragment releases from the number 2 HBR engine’s HPC, stage 3. The run was set up with 100 iterations of a single release point (180° with 0° variance) in order to better illustrate the advantages of the significant iteration feature. In addition, the aft and fore spread angles for the small and large fragments are set to $-30^{\circ}/-45^{\circ}$ and $10^{\circ}/20^{\circ}$, respectively, and the debugging feature is turned on for all of these runs so that the shotlines can be viewed in the UEDDAM visualizer.

Figure 9-9 shows one of the output files created for this sample case; namely, the SIGITCHOLE FILE. As indicated in the file, only 22 of the 100 iterations met the criteria specified in the UEDDAM control file (i.e., iterations resulting in cumulative hole sizes greater than 12 in^2). To investigate these significant iterations further, the UEDDAM visualizer can be used to examine only those trajectories that resulted in the critical cumulative hole size being exceeded (see Figure 9-10).

```

UEDDAM v5.0
EVENT:, MULTI, ENGINE2, HighBypassRatio, HPC, Stage3
, , Start , 180.00, deg,
, , Size , 0.00, deg,
, , Stop , 180.00, deg,
, , # of Release Points , 1,,
, , Random Number Seed , 2201,,

Criteria:, 12.00, in^2
Debris Release Angle:, 180.00
Number of Iterations:, 100

Iteration #, Cumulative Holesize
9, 14.98
11, 16.02
13, 14.38
16, 14.43
17, 12.35
18, 12.44
22, 14.55
25, 12.70
30, 14.94
37, 13.99
40, 15.00
41, 15.90
47, 12.41
57, 14.40
61, 13.28
65, 13.88
67, 12.11
68, 15.94
69, 14.21
76, 15.14
79, 12.80
90, 12.57
Number of significant Iterations:, 22

```

Figure 9-9. Sample Case Four – SIGITCHOLE Output.

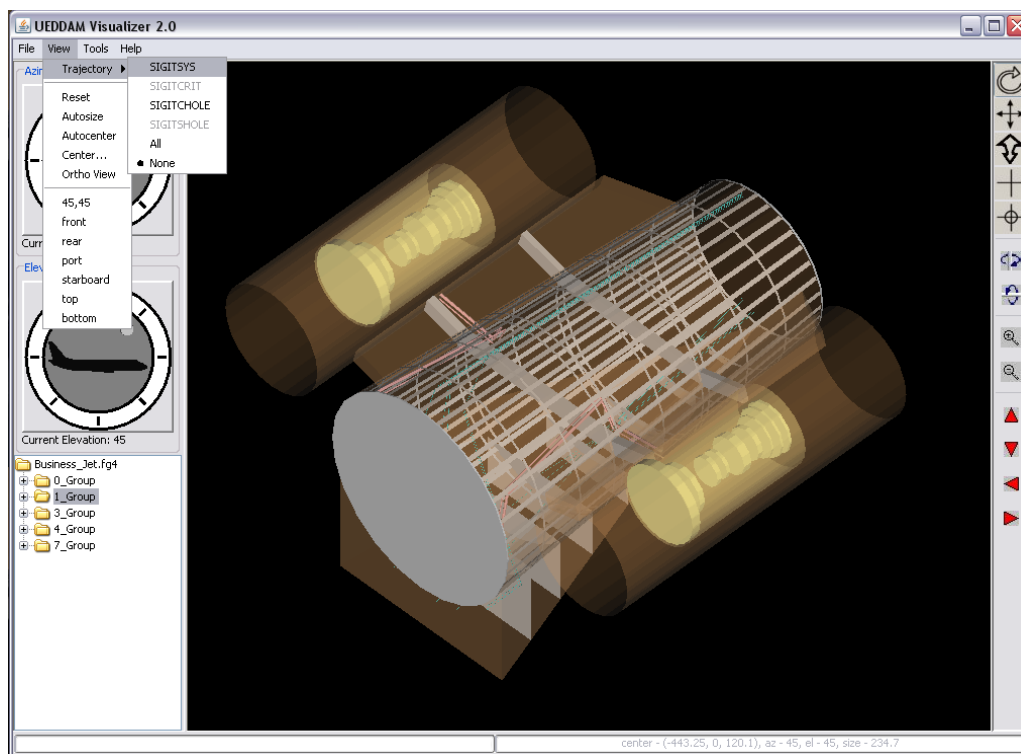


Figure 9-10. Sample Case Four – Significant Iteration Trajectories.

Figure 9-11 shows another of the output files created for this sample case; namely, the SIGITSYS FILE. As indicated in this file, 73 of the 100 iterations met the criteria specified in the UEDDAM control file (i.e., iterations resulting in total aircraft P_{HAZ} greater than 0.90). Note that most iterations result in a total aircraft P_{HAZ} greater than the critical value for all three hazard levels under consideration. Iterations 3 and 13, on the other hand, do not meet the criteria for hazard level 2. This difference can be examined in more detail by looking at the other UEDDAM output files created and/or by using the UEDDAM visualizer to load those specific trajectories.

```

UEDDAM v5.0
EVENT:, MULTI, ENGINE2, HighBypassRatio, HPC, Stage3
, , Start , 180.00, deg,
, , Size , 0.00, deg,
, , Stop , 180.00, deg,
, , # of Release Points , 1,,
, , Random Number Seed , 2201,,

Criteria:, 0.90, P_HAZ
Debris Release Angle:, 180.00
Number of Iterations:, 100

Iteration #, Hazard Level,P_HAZ, Hazard Level,P_HAZ, Hazard Level,P_HAZ,
1, 1,0.99 , 2,0.99 , 3,0.99 ,
2, 1,1.00 , 2,1.00 , 3,1.00 ,
3, 1,0.99 , 3,0.99 ,
4, 1,1.00 , 2,1.00 , 3,1.00 ,
5, 1,1.00 , 2,1.00 , 3,1.00 ,
6, 1,1.00 , 2,1.00 , 3,1.00 ,
8, 1,1.00 , 2,1.00 , 3,1.00 ,
9, 1,0.99 , 2,0.99 , 3,0.99 ,
10, 1,1.00 , 2,1.00 , 3,1.00 ,
11, 1,1.00 , 2,1.00 , 3,1.00 ,
12, 1,1.00 , 2,1.00 , 3,1.00 ,
13, 1,0.99 , 3,0.99 ,
14, 1,1.00 , 2,1.00 , 3,1.00 ,
15, 1,1.00 , 2,1.00 , 3,1.00 ,
16, 1,1.00 , 2,1.00 , 3,1.00 ,

89, 1,1.00 , 2,1.00 , 3,1.00 ,
90, 1,0.99 , 2,0.99 , 3,0.99 ,
91, 1,1.00 , 2,1.00 , 3,1.00 ,
93, 1,1.00 , 2,1.00 , 3,1.00 ,
94, 1,1.00 , 2,1.00 , 3,1.00 ,
95, 1,1.00 , 2,1.00 , 3,1.00 ,
96, 1,1.00 , 2,1.00 , 3,1.00 ,
97, 1,1.00 , 2,1.00 , 3,1.00 ,
98, 1,1.00 , 2,1.00 , 3,1.00 ,
99, 1,1.00 , 2,1.00 , 3,1.00 ,
100, 1,1.00 , 2,1.00 , 3,1.00 ,
Number of Significant Iterations:, 73

```

Figure 9-11. Sample Case Four – SIGITSYS Output.

Additional significant iteration output files generated for this sample case (but not reproduced here) include the SIGITCRIT and SIGITSHOLE FILES. Both can be viewed by browsing to the samples directory of the UEDDAM installation directory.

9.5 SAMPLE CASE FIVE – DEBRIS DISTRIBUTIONS

Sample case five consists of three parts, which compare the three available distributions (uniform, normal, and skewed) in the debris characterization record. The geometry considered is a partial section of a representative business jet. For all three parts of this sample, the debris assessed consists of a single small fragment, which is released from the number 2 HBR engine’s HPC, stage 3. In order to better see the nature of the distributions, one hundred iterations of a single rotational release angle of 180° (with 0° variance) are performed. In addition, the aft and fore spread angles are set to -30° and 10° , respectively, and the debugging feature is turned on for all of these runs so that the shotlines can be viewed using the UEDDAM visualizer. Note that this sample case also makes use of the multi-threaded capability available in UEDDAM. To turn this capability off for use on a computer without multiple processors, simply place a \$ before the THREADS keyword in each of the three UEDDAM control files.

9.5.1 Uniform Distribution

The following debris record was extracted from the UEDDAM control file Control_Uniform.ucf, which is included in the directory “sample5:”

```
DEBRIS, High Bypass Ratio, SINGLE, HPC, S-FRAGMENT  
S-FRAGMENT, 12, 1  
1, 1, Steel_300, 0.19, 609, 4, 2, 0.15, 45, 0, 10, -30, 0, UNIFORM
```

The results of this run can be seen in Figure 9-12. The image on the left was generated using the UEDDAM visualizer and shows the generated trajectories along with the geometry of the representative business jet. Accompanying the image is a histogram of the fore/aft release angles overlaid with a plot of the governing uniform distribution.

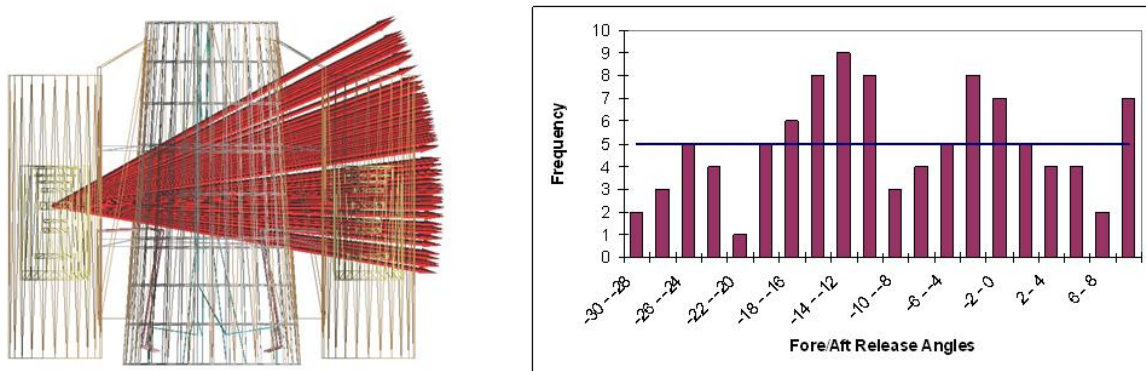


Figure 9-12. Sample Case Five – Uniform Distribution for Debris Characterization.

9.5.2 Normal Distribution

The debris characterization record for part two of this sample case was modified to make the distribution normal with a mean of -10° and a deviation of 5° . The following debris record was extracted from the UEDDAM control file Control_Normal.ucf:

```
DEBRIS, High Bypass Ratio, SINGLE, HPC, S-FRAGMENT  
S-FRAGMENT, 12, 1  
1, 1, Steel_300, 0.19, 609, 4, 2, 0.15, 45, 0, 10, -30, 0, NORMAL, -10, 5
```

The results of this run can be seen in Figure 9-13. As in part one of sample case five, a histogram of the fore/aft release angles overlaid with a plot of the governing normal distribution accompanies a UEDDAM visualizer image of the representative business jet with generated trajectories. The distribution of the fore/aft release angles clearly differs from that seen in Figure 9-12.

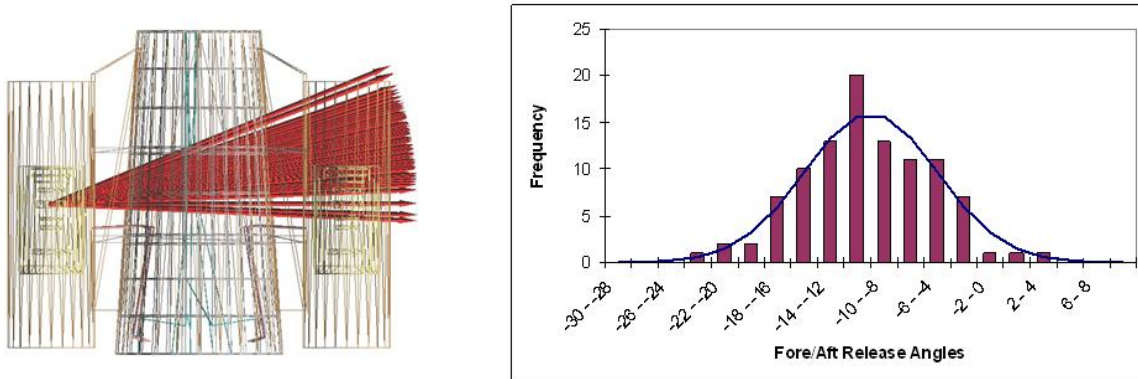


Figure 9-13. Sample Case Five – Normal Distribution for Debris Characterization.

9.5.3 Skewed Distribution

For part three of sample case five, the debris characterization record was again modified to make the distribution skewed with a mean of -10° (the same as in part two), a deviation of 2° , and a direction of AFT. The following debris record was extracted from the UEDDAM control file Control_Skew.ucf:

```
DEBRIS, High Bypass Ratio, SINGLE, HPC, S-FRAGMENT  
S-FRAGMENT, 12, 1  
1, 1, Steel_300, 0.19, 609, 4, 2, 0.15, 45, 0, 10, -30, 0, SKEWED, -10, 2, AFT
```

The results of this run can be seen in Figure 9-14. Again, a histogram of the fore/aft release angles overlaid with a plot of the governing “skewed” distribution show how the trajectories generated in part three of sample case five differ greatly from those in parts one and two.

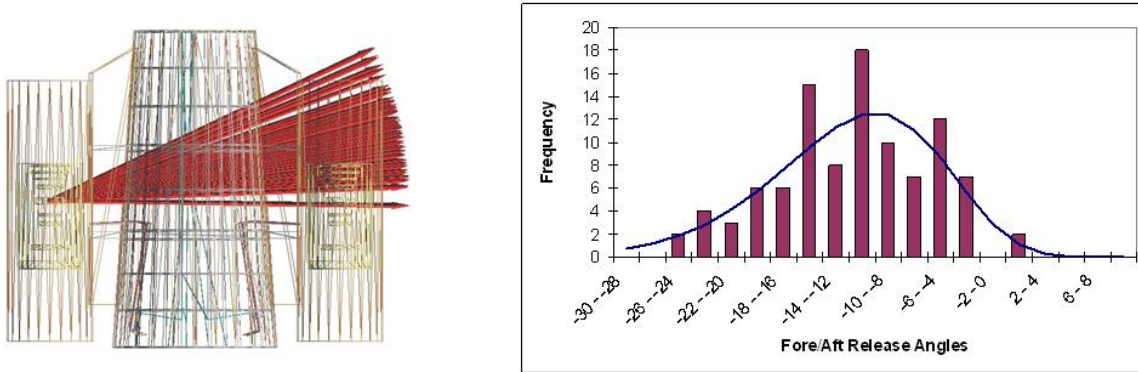


Figure 9-14. Sample Case Five - Skewed Distribution for Debris Characterization.

9.6 SAMPLE CASE SIX – GENERIC TWIN-ENGINE AIRCRAFT

Sample case six exercises UEDDAM’s ability to operate on a complex TARGET FILE (that of a generic twin-engine aircraft, as seen in Figure 9-15) along with realistic debris characteristics. In this sample, the debris assessed consists of a single 1/3-disk segment plus multiple small fragments. The event MULTI occurs in the number 1 HBR engine’s fan, stage 1. For comparison purposes, five rotational release angles in the interval from 330° to 60° were considered (with five iterations apiece). Due to the complexity of the target, the geometry reduction flag was turned on to reduce overall run time and a tolerance was specified for the component intercept angle output file to combine some of the subassemblies, thereby reducing the size of the INTERCEPT FILE. In addition, the debugging feature was turned on.

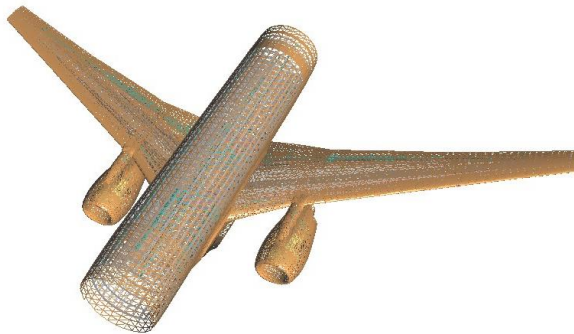


Figure 9-15. Generic Twin-Engine Aircraft for Sample Case Six.

For this run, three catastrophic hazard levels (take-off to V1, V1 to V1+30, and initial climb) were specified. One of the many results gathered from this execution can be seen by examining a plot (generated using the UEDDAM visualizer) of the contents of the SUMMARY FILE as shown in Figure 9-16. The hazard probability summary shows that, if the debris is released from 15°, the probability that catastrophic damage will be incurred during the take-off to V1 phase is 0.0, whereas during the remaining two flight phases, catastrophic damage occurs with a probability of 0.2.

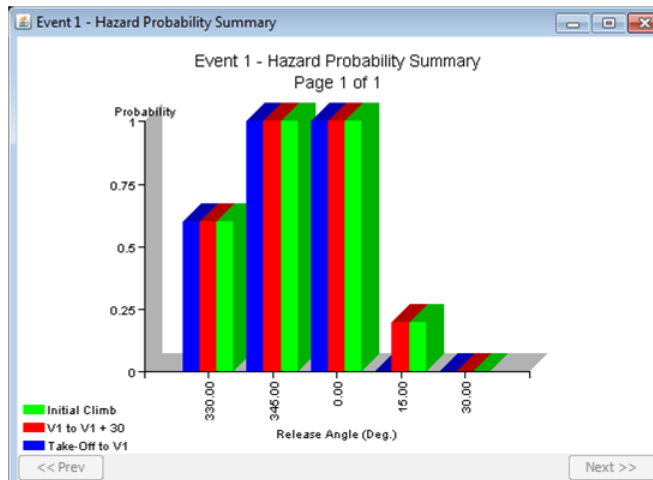


Figure 9-16. Hazard Probability Summary for Sample Case Six.

9.7 SAMPLE CASE SEVEN – MODIFICATION OF SHEAR CONSTANT

Sample case seven demonstrates a feature new to UEDDAM v4.3: the capability for a user to define/modify two shear constants for a material type for use with the FAA-based penetration equations through the MATERIAL FILE. The geometry used for this sample case is shown in Figure 9-17 and consists of a dummy engine surrounded by several small plates (highlighted green in the figure) at various angles. A large Aluminum plate (highlighted blue in the figure) is placed between the engine and the smaller plates to represent a shielding component. The engine debris for this sample case is released from the top of the representative engine and must penetrate the shielding plate prior to impacting any of the smaller critical plates.

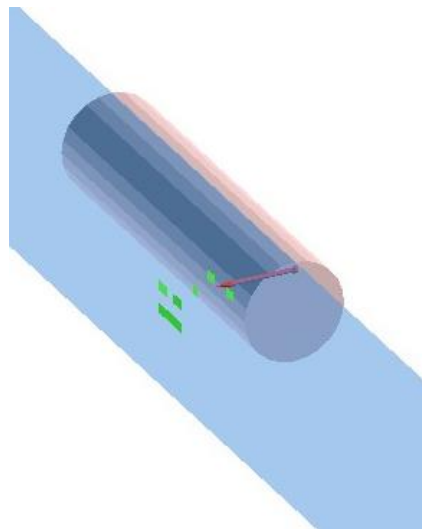


Figure 9-17. Sample Case Seven Geometry.

There are two separate UEDDAM runs for this sample: “Cs_50” and “Cs_0.” The “Cs_50” portion of this sample case is the baseline scenario, which executes UEDDAM for the event shown in Figure 9-17 using the material properties found in the standard MATERIAL FILE. The “Cs_0” portion of this sample case, on the other hand, executes UEDDAM for the exact same scenario, except that a second shear constant has been added to the material properties for Al_2024 in the MATERIAL FILE. Upon comparison of the SUMMARY FILEs from the two UEDDAM runs, the “Cs_0” scenario results in the probabilities of significant and/or minor damage occurring having been completely eliminated. This, of course, is due to the change in penetration assumptions regarding the material when fragments impact at or below the ballistic limit velocity (i.e., the inclusion of a second, higher C_{s_0} value to characterize the shielding plate).

9.8 SAMPLE CASE EIGHT – UEDDAM HOLE SIZE ESTIMATION

Sample case eight illustrates the UEDDAM hole size estimation methodology added for UEDDAM v3.0. The data set used is the same as that for sample case six, except that for this scenario, the debris consists of 3 large fragments, each released from the number 1 engine at the same point, but at 3 different release angles (i.e., 340°, 350°, and 0°), as illustrated in Figure 9-18.

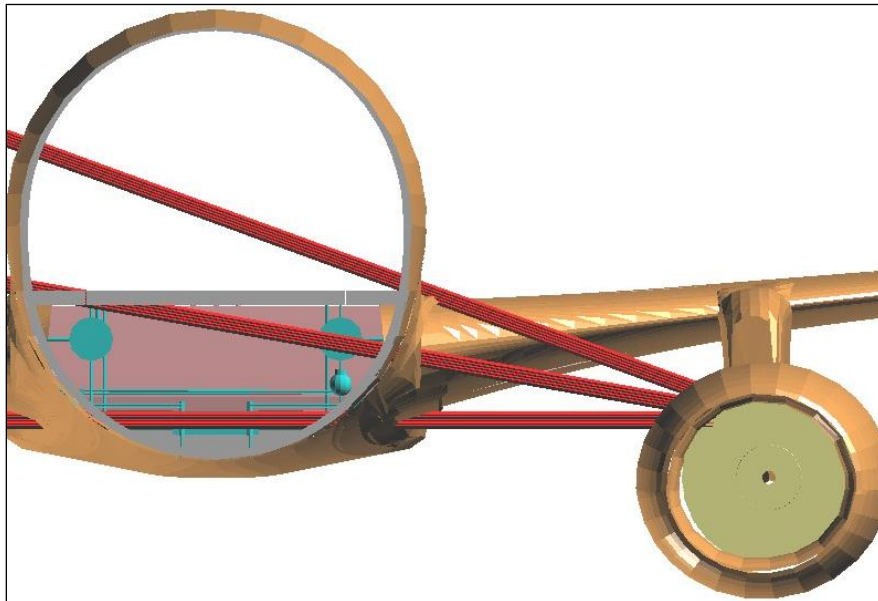


Figure 9-18. Debris Release Angles for Sample Case Eight.

For sample case eight, the hole size component groupings are defined in the HOLEGRP FILE and consist of four distinct groups (i.e., Port_Skin, Stbd_Skin, Port_Wing, and Stbd_Wing), each of which is made up of either 6 or 10 individual components. UEDDAM calculates damage size statistics for each HOLEGRP identified and outputs this information to the HOLESIZE FILE. A portion of the HOLESIZE FILE for sample case eight is provided in Figure 9-19.


```

Group Information:
, Port_skin, 201, 202, 203, 204, 213, 214,
, Stbd_skin, 251, 252, 253, 254, 263, 264,
, Port_wing, 101, 102, 103, 104, 105, 106,
, , 107, 108, 109, 110,
, Stbd_wing, 111, 112, 113, 114, 115, 116,
, , 117, 118, 119, 120,

EVENT:, MULTI, ENGINE1, HighBypassRatio, Fan, Stage1

Release Point 1:, Angle 340.00,
, Group Name , Maximum Hole Area, Minimum Hole Area, Average Hole Area, Fragment Hits, Fragment Penetrations,
, Port_skin , , 38.86, 28.54, 33.86, 8.00, 8.00,
, Stbd_skin , , 24.05, 24.02, 24.04, 3.60, 3.60,
, Port_wing , , 0.00, 0.00, 0.00, 0.00, 0.00,
, Stbd_wing , , 0.00, 0.00, 0.00, 0.00, 0.00,

Release Point 2:, Angle 350.00,
, Group Name , Maximum Hole Area, Minimum Hole Area, Average Hole Area, Fragment Hits, Fragment Penetrations,
, Port_skin , , 25.40, 25.37, 25.39, 3.00, 3.00,
, Stbd_skin , , 39.62, 14.37, 26.45, 6.60, 6.60,
, Port_wing , , 39.61, 13.75, 31.53, 2.60, 2.60,
, Stbd_wing , , 0.00, 0.00, 0.00, 0.00, 0.00,

```

Figure 9-19. Sample Case Eight HOLESIZE FILE.

9.9 SAMPLE CASE NINE – LOGIC STATEMENT INPUTS

Sample case nine demonstrates UEDDAM’s logic statement processing utility, which takes a series of logic statements and converts them to MV FILE inputs. For this sample case, the simple geometric setup shown in Figure 9-20 is used. Additionally, the failure combinations of interest (and their equivalent logic statements) are provided in Table 9-4, where C# is used in place of CLINE# for simplification. Note that in Table 9-4, “0” represents “not failed” and “1” represents “failed.” The series of logic statements shown in the last column of Table 9-4 are entered in the LOGIC FILE for this sample case.

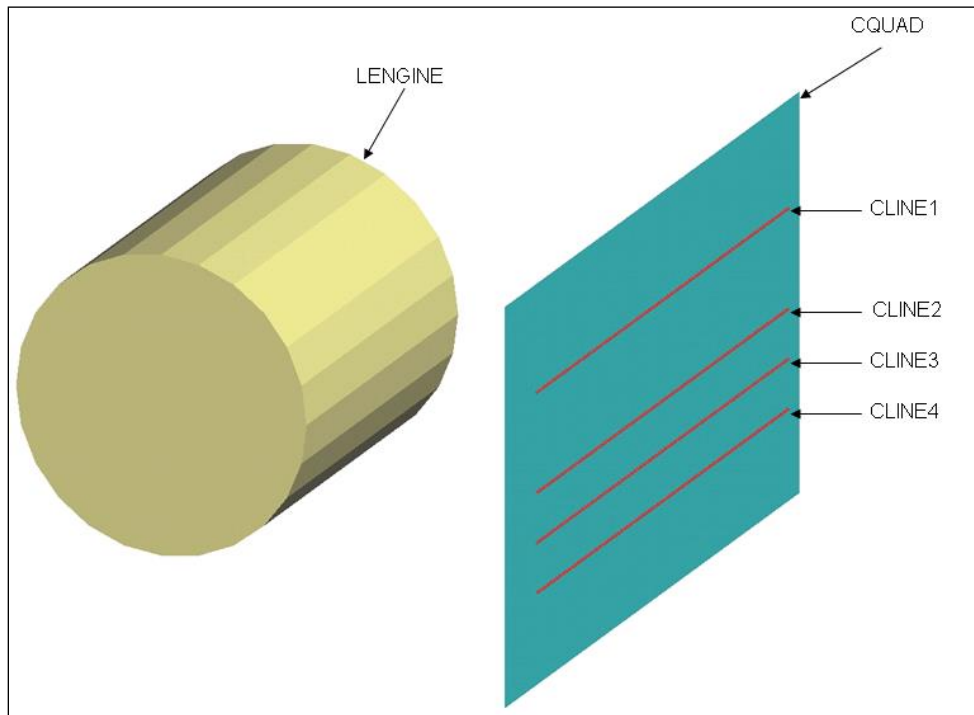


Figure 9-20. Model Setup for Sample Case Nine.

Table 9-4. Sample Case Nine Failure Combinations

Probability of System Failure	Component Failures				Equivalent Logic Statement
	C1	C2	C3	C4	
0.00	0	0	0	0	.NOT. (C1 .OR. C2 .OR. C3 .OR. C4)
0.06	0	0	0	1	C4 .NOT. (C1 .OR. C2 .OR. C3)
0.12	0	0	1	0	C3 .NOT. (C1 .OR. C2 .OR. C4)
0.18	0	0	1	1	(C3 .AND. C4) .NOT. (C1 .OR. C2)
0.24	0	1	0	0	C2 .NOT. (C1 .OR. C3 .OR. C4)
0.30	0	1	0	1	(C2 .AND. C4) .NOT. (C1 .OR. C3)
0.36	0	1	1	0	(C2 .AND. C3) .NOT. (C1 .OR. C4)
0.42	0	1	1	1	(C2 .AND. C3 .AND. C4) .NOT. C1
0.48	1	0	0	0	C1 .NOT. (C2 .OR. C3 .OR. C4)
0.54	1	0	0	1	(C1 .AND. C4) .NOT. (C2 .OR. C3)
0.60	1	0	1	0	(C1 .AND. C3) .NOT. (C2 .OR. C4)
0.66	1	0	1	1	(C1 .AND. C3 .AND. C4) .NOT. C2
0.72	1	1	0	0	(C1 .AND. C2) .NOT. (C3 .OR. C4)
0.78	1	1	0	1	(C1 .AND. C2 .AND. C4) .NOT. C3
0.84	1	1	1	0	(C1 .AND. C2 .AND. C3) .NOT. C4
1.00	1	1	1	1	(C1 .AND. C2 .AND. C3 .AND. C4)

For this sample, a single release point at 20° was selected to hit and fail three of the four components (specifically, CLINE2, CLINE3, and CLINE4, but not CLINE1). Upon successful execution, UEDDAM outputs a probability of hazard value of 0.42 in the SUMMARY FILE for this particular release angle. This result is verified by examining the highlighted row in Table 9-4.

9.10 SAMPLE CASE TEN – HYPERSAMPLING

Sample case ten illustrates the hypersampling capability added for UEDDAM 3.0. As discussed in Section 5.3.1.5, very small components can potentially be missed by the non-dimensional shotlines generated to approximate large fragment and/or segment presented areas, as illustrated in Figure 9-21.

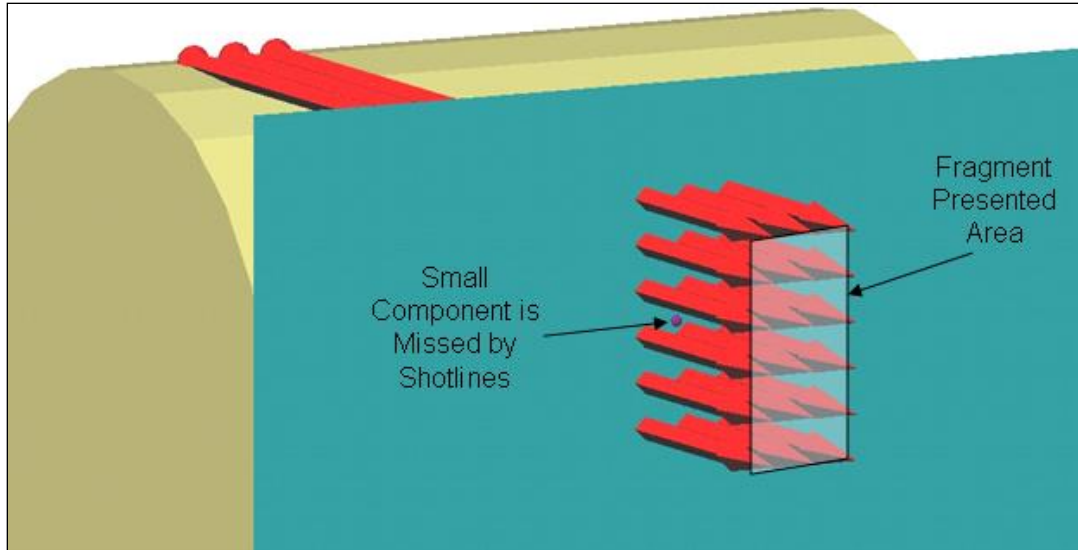


Figure 9-21. UEDDAM Approximation of Large Fragment Trajectories for Sample Case Ten.

There are two separate UEDDAM runs included for this sample case. The first scenario (“No SCF”) does not activate the hypersampling option in UEDDAM and results in the shotlines representing the trajectory of the large fragment seen in Figure 9-21. The second scenario (“Uniform SCF”), which activates UEDDAM’s hypersampling option, results in an extra shotline being generated to hit the small component (see Figure 9-22). The “No SCF” UEDDAM execution results in a hazard probability of 0.00. This result can be potentially misleading, however, if the small component is considered critical (which, for this sample case, it is). The “Uniform SCF” run, due to the additional shotline it generates, results in the correct hazard probability of 1.00.

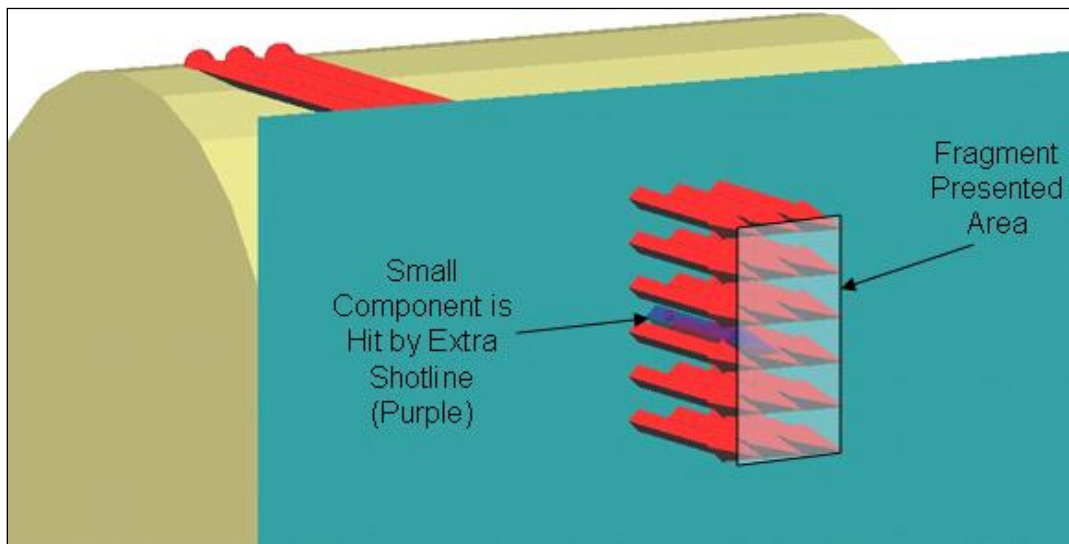


Figure 9-22. UEDDAM Small Component Fix for Sample Case Ten.

The top half of the setup macro spreadsheet (shown in Figure 9-24) is partially populated by the information entered in the lower half; the remaining areas are for user input and options (e.g., path to root directory, random number seed, number of iterations, run options, output file selection, etc.).

Populated from User Input

Create Files in: (Root Directory) C:\Program Files\SURVICE\UEDDAM\samples\sample11
 (do not put \ at the end)

Target File (in Root Dir) Generic_Twin.fig4

Select Debris Type: MULTI, SINGL, ALL
 Select Engine: ENGINE1, ENGINE2, ALL
 Select Component: Fan, LPT, ALL

Number of Iterations: 2
 Random Seeds: 250
 Number of Iterations (SINGL): 30
 Start Release Angle: 0
 Sweep: 360 Degrees from Start Angle
 Number of Release Angles: 12

User Input

JTYPE and MV/LOGIC Files
 UNIQUE FOR EACH ENGINE
 SAME FOR EACH ENGINE

Select Run Options
 DEBUGGING
 TARGET REDUCTION
 HYPERSAMPLING
 PENETRATION
 EXECUTE ANALYSIS_TOOL
 GENERATE BATCH FILE
 THREADS

Fault Tree Input File
 MV FILE
 LOGIC FILE

Select Output Files
 DETAIL FILE
 TRAJECTORY FILE
 INTERCEPT FILE
 HOLESIZE FILE
 VISUALIZER FILE
 SIGITCRIT FILE
 ERROR FILE
 SIGITSYS FILE
 SIGITSHOLE FILE
 SIGITCHOLE FILE

Summary of Output
 ALL OUTPUT FILES
 NEW OUTPUT FILES
 NONE (DEFAULT)

Hypersampling Options
 RANDOM
 CENTERED
 SGMT: 0.15 Inches
 LFRG: 0.15 Inches
 SFRG: [Yellow Box] Inches

Multiple Processor Flag
 # of proc: [Grey Box]

0°-30° bin: [Grey Box] **30°-45° bin:** [Grey Box]
45°-60° bin: [Grey Box] **60°-75° bin:** [Grey Box]
75°-90° bin: [Grey Box] **Gamma Values**

User Options

Set Up UEDDAM Input Control Files

Figure 9-24. Upper Half of Setup Macro Spreadsheet.

For sample case 11, debris is released every 30° between 0° and 360° and multiple iterations are performed at each release angle (the exact number of iterations is dependent upon the debris type being assessed—2 iterations for debris type MULTI, 30 for debris type SINGL). Four of the available ten optional output files are selected for generation, the target reduction flag has been toggled on, and segment and large fragment hypersampling (with a subgrid of 0.15 inches) has been selected. When the “Set Up UEDDAM Input Control Files” button is pushed, the built-in macro creates the directory structure and UEDDAM control files for the desired analysis. Figure 9-25 shows the directory structure and one of the eight control files created.

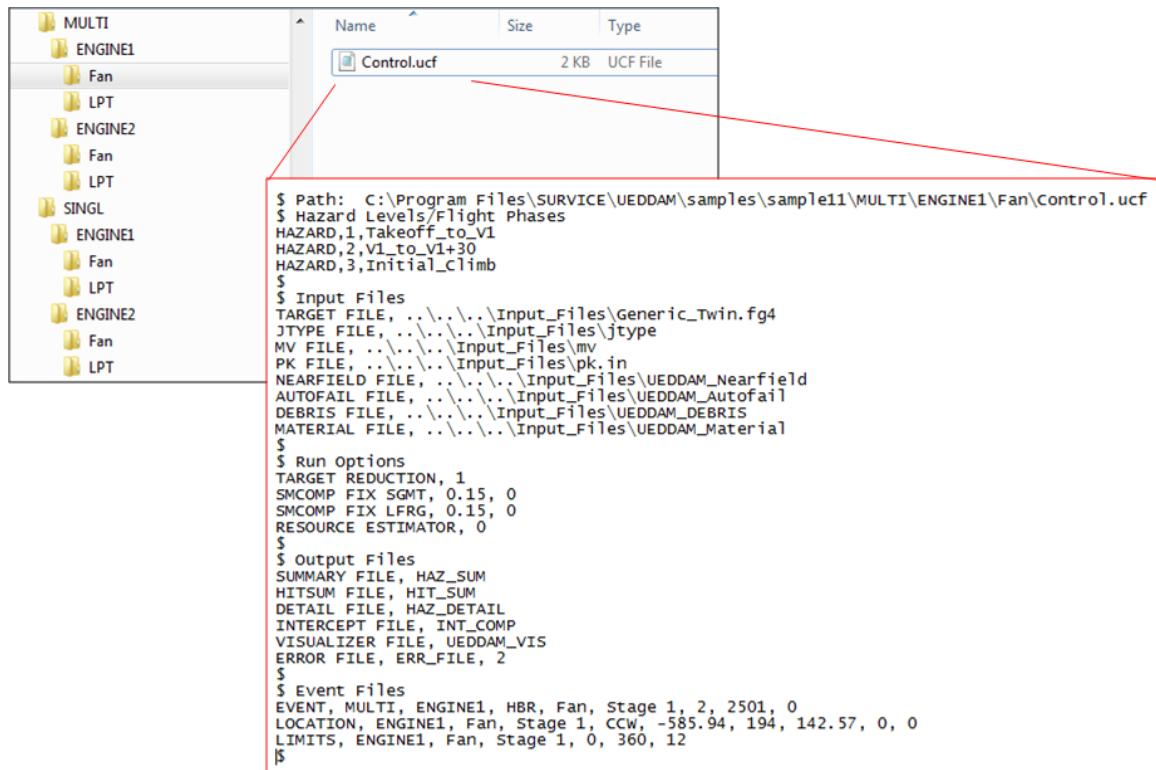


Figure 9-25. Files Created by Setup Macro.

In addition, because the “Generate Batch File” option was selected, an executable file called UEDDAM_Runs.bat is created in the root directory. Finally, the input file required to run Analysis_Tool.jar is created in the Input_Files directory (it is called Analysis_Tool_Input.txt).

After UEDDAM has been successfully executed for each of the eight debris-engine-stage combinations, the user is left with 48 separate output files from which he or she must draw conclusions about the aircraft. The Analysis_Tool.jar post-processing program was developed to make this process a little simpler. To execute this tool, the user must create a Bin directory in the sample11 directory and copy the Analysis_Tool.jar executable from the tools directory of the UEDDAM installation into the Bin directory of sample11 (recall that the tool expects the directory structure illustrated in Figure 6-2). Once this step is complete, simply double-clicking on the Analysis_Tool.jar executable will run the tool, which generates 2 top-level summary files (Combined_Debris.csv and PH_sum.txt) and 14 lower-level summary files (Combined_Engine.csv [2], Combined_Rotor.csv [4], and Combined_Flight_Phase.csv [8]). The two top-level summary files are reproduced in Figure 9-26.

```

*****ENGINE1*****
-----Takeoff_to_v1-----
          Fan      LPT
MULTI      0.16667  0.04167
SINGL      0.00000  0.00000

-----v1_to_v1+30-----
          Fan      LPT
MULTI      0.16667  0.04167
SINGL      0.00000  0.00000

-----Initial_Climb-----
          Fan      LPT
MULTI      0.16667  0.04167
SINGL      0.00000  0.00000

COMPONENT PK (Dp ADJUSTED AVERAGES)
          Fan      LPT
MULTI      0.12834  0.03208
SINGL      0.00000  0.00000

ENGINE PK (Average of Components)
MULTI      0.08021
SINGL      0.00000

*****ENGINE2*****
-----Takeoff_to_v1-----
          Fan      LPT
MULTI      0.12500  0.00000
SINGL      0.00000  0.00000

-----v1_to_v1+30-----
          Fan      LPT
MULTI      0.16667  0.08333
SINGL      0.00278  0.00000

-----Initial_Climb-----
          Fan      LPT
MULTI      0.16667  0.08333
SINGL      0.00000  0.00000

COMPONENT PK (Dp ADJUSTED AVERAGES)
          Fan      LPT
MULTI      0.11375  0.03500
SINGL      0.00056  0.00000

ENGINE PK (Average of Components)
MULTI      0.07438
SINGL      0.00028

***** AIRCRAFT Pk/h *****
MULTI      0.07729
SINGL      0.00014

```

Combined Debris - Average		
Average over all release angles:		
	MULTI	SINGL
SYSTEM	0.07729	0.00014
HYD_SYS	0.00438	0
FRAMES	0.01604	0
FRM_2	0.01604	0
FRM_7	0.00802	0
LONGERNS	0.06417	0
LNGRN_1	0.01604	0
LNGRN_2	0.01604	0
LNGRN_3	0.03208	0
LNGRN_4	0.02406	0
LNGRN_5	0.02406	0
LNGRN_6	0.00802	0
LNGRN_7	0.0401	0
LNGRN_8	0.00802	0
WINGX	0.00438	0
FWDX	0.00438	0
ENGINES	0.03062	0
ELEVATOR	0.00875	0
LAT_CONT	0.00875	0
RUDDER	0.00417	0.00014
RUDR_FCS	0.00229	0
Average over 360-degrees:		
	MULTI	SINGL
SYSTEM	0.07729	0.00014
HYD_SYS	0.00438	0
FRAMES	0.01604	0
FRM_2	0.01604	0
FRM_7	0.00802	0
LONGERNS	0.06417	0
LNGRN_1	0.01604	0
LNGRN_2	0.01604	0
LNGRN_3	0.03208	0
LNGRN_4	0.02406	0
LNGRN_5	0.02406	0
LNGRN_6	0.00802	0
LNGRN_7	0.0401	0
LNGRN_8	0.00802	0
WINGX	0.00438	0
FWDX	0.00438	0
ENGINES	0.03062	0
ELEVATOR	0.00875	0
LAT_CONT	0.00875	0
RUDDER	0.00417	0.00014
RUDR_FCS	0.00229	0

Figure 9-26. Sample Case Eleven: PH_sum.txt and Combined_Debris.csv.

From these two summary files, a few observations can be made: (1) the aircraft experiences a higher risk when debris is released from ENGINE1 than if it were released from ENGINE2; 2) the aircraft experiences a higher risk from the MULTI debris type than from the SINGL debris type; and 3) the largest contributing systems to aircraft risk from the MULTI debris type are LONGERNS and ENGINES. By analyzing the other summary files created in a similar manner, additional insights can be discovered.

9.12 SAMPLE CASE TWELVE – INFINITE ENERGY DISK

Sample case twelve illustrates how to perform a UEDDAM assessment using the conservative assumptions detailed in the appendix to the AC 20-128A. According to the risk analysis methodology described, the 1/3 disk fragment (as modeled in paragraph 9(a) of AC 20-128A) “is considered to possess infinite energy.” This statement is further explained as follows: the fragment is “capable of severing lines, wiring, cables, and unprotected structure in its path, and to be undeflected from its original trajectory unless deflection shields are fitted; however, protective shielding or an engine being impacted may be assumed to have sufficient mass to stop even the most energetic fragment.” [8]

UEDDAM as it is coded takes into account the specific penetration capabilities of the debris as it interacts with the surrounding aircraft components. In order to force the model to stop calculating penetration of all components in the path of the debris (minus any deflection shields and/or a second engine), changes must be made to the material properties assigned to these components in the JTYPE FILE. To build this sample case, input files from sample case five (the generic business jet) were copied and modified as follows: all components (minus the components of the left engine since the disk debris is being released from the right engine in this sample case) were changed to material type 51 (i.e., water) with a 1% density factor (see Figure 9-27).

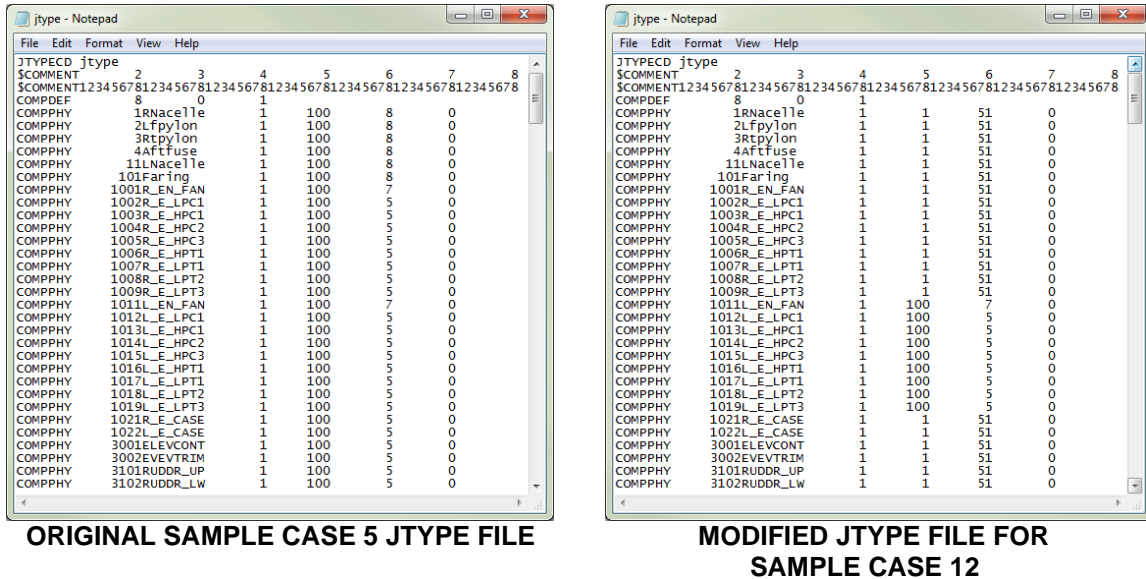


Figure 9-27. JTYPE FILE Changes for Sample Case Twelve.

Because the penetration equations embedded in UEDDAM are only used to calculate impacts to solid materials, by making this simple change to the material characteristics of the aircraft components, UEDDAM no longer calculates the penetration capabilities of the disk debris unless it encounters the opposite engine. Thus, the disk fragment is now considered to have infinite energy.

9.13 SAMPLE CASE THIRTEEN – PENETRATION EQUATIONS

Sample case thirteen illustrates use of the PENETRATION control file keyword added for UEDDAM v4.1. This sample case uses a modified version of the simplistic geometry and inputs seen in sample cases one through three so that the differences introduced by the change in penetration equations can be more readily observed (see Figure 9-28).

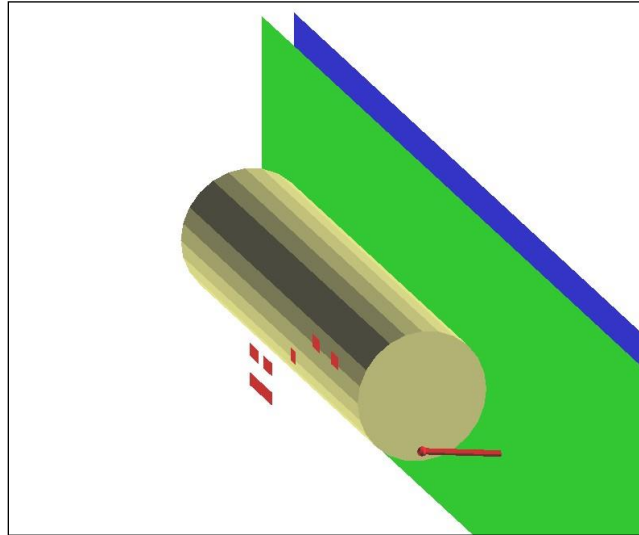


Figure 9-28. Sample Case Thirteen Geometry.

Two different scenarios are presented in sample case thirteen. The first does not include the PENETRATION keyword in the control file so that the values of gamma for all obliquity angle bins default to 1.0. The second scenario introduces the PENETRATION keyword to the control file and defines all values of gamma as 0.1. In both scenarios, a single small fragment is released from the first stage HPT at 180°; the fragment's trajectory is constrained between forward and aft release angles of 25° and 24°, respectively; and the fragment is travelling at 1890fps.

Upon successful execution of UEDDAM, a difference in P_{HAZ} can be seen in the resulting SUMMARY FILES. However, it is not until investigating the resulting DETAIL FILES that the difference between the original and modified penetration equations is better understood. In the first scenario, only the first plate along the fragment's trajectory (i.e., LeftP1 – the green plate in Figure 9-28) is attributed a P_{haz} at each of the assessed hazard levels. In the second scenario, on the other hand, both the first and second plates along the fragment's trajectory (i.e., LeftP1 and LeftP2 – the green and blue plates in Figure 9-28, respectively) receive a P_{haz} . This difference is explained by the calculation of V_{50} . Recall that the V_{50} is inversely proportional to $\cos^{\gamma}\theta$, where θ is the impact obliquity angle of the debris; thus, all else remaining equal, a larger value of gamma corresponds to a larger V_{50} (since $\cos\theta$ is constrained to be between 0 and 1) and vice versa. In the first scenario, the V_{50} calculated for penetration of LeftP1 is approximately 2040fps, which is greater than the impact velocity of the small fragment and thus the debris is stopped by that first plate. In the second scenario, however, the V_{50} calculated

for penetration of LeftP1 is approximately 1870fps, which is less than the impact velocity of the small fragment and thus the debris is allowed to continue on its trajectory (with a reduced velocity) to impact LeftP2.

9.14 SAMPLE CASE FOURTEEN – MULTI-SEGMENT ANALYSES

Sample case fourteen illustrates the use of multi-segment analysis options added as part of UEDDAM v5.0 development. The sample case is subdivided into two parts. In the first part, UEDDAM is instructed to allow the release of small and large fragments only with the release of the first segment (as indicated by a “0” in the debris file on the debris group identifier SEGMENT definition record). The second scenario instructs UEDDAM to allow the release of small and large fragments with the release of all three segments (as indicated by a “1” in the debris file on the debris group SEGMENT definition record). The sample case reuses the input files from sample cases 1, 2, and 3 with the only modification being the compilation of the three types of debris into a single debris category (MULTI). The first debris file is shown below.

```

$ UEDDAM Sample #14 - Tri-hub release; small and large fragments
$ released with the first segment
$ Debris Data
$
DEBRIS, High Bypass Turbofan, MULTI, HPT, All
  SEGMENT, 15.0, 3, 1, 0
    1, Steel_300, 12, 300, 120, 1, 5.0, -5.0, 0.0
      3, 0.33, 1.0, 0.66, 1.0, 1.0, 0.80
    2, Steel_300, 11, 400, 120, 1, 5.0, -5.0, 120.0
      3, 0.33, 1.0, 0.66, 1.0, 1.0, 0.80
    3, Steel_300, 10, 500, 120, 1, 5.0, -5.0, 240.0
      3, 0.33, 1.0, 0.66, 1.0, 1.0, 0.80
  L-FRAGMENT, 15.0, 4, 1
    1, 3, Steel_300, 2, 500, 4, 2, 0.50, 0, 45., 5.0, -5.0, 0.0
      3, 0.33, 1.0, 0.66, 1.0, 1.0, 0.80
    2, 3, Steel_300, 3, 800, 8, 2, 0.50, 0, 45., 5.0, -5.0, 0.0
      3, 0.33, 1.0, 0.66, 1.0, 1.0, 0.80
    3, 2, Steel_300, 5, 300, 10, 2, 0.50, 0, 45., 5.0, -5.0, 0.0
      3, 0.33, 1.0, 0.66, 1.0, 1.0, 0.80
    4, 2, Steel_300, 7, 250, 12, 3, 0.50, 0, 45., 5.0, -5.0, 0.0
      3, 0.33, 1.0, 0.66, 1.0, 1.0, 0.80
  S-FRAGMENT, 17.0, 4
    1, 3, Steel_300, 0.25, 500, 1, 1, 0, 45., 45., 5.0, -5.0, 0.0
    2, 3, Steel_300, 0.50, 800, 2, 1, 0, 45., 45., 5.0, -5.0, 0.0
    3, 2, Steel_300, 0.75, 300, 4, 1, 0, 45., 45., 5.0, -5.0, 0.0
    4, 2, Steel_300, 1.00, 250, 5, 2, 0, 45., 45., 5.0, -5.0, 0.0

```

Multi-segment
debris flag

The results of sample case fourteen are best observed by examining the debris generated in each of the two cases (see Figure 9-29). In the image on the left, the small (green) and large (purple) fragments are released only with the first of the three segments (blue). In the image on the right, the same small and large fragments are released with all three of the segment releases.

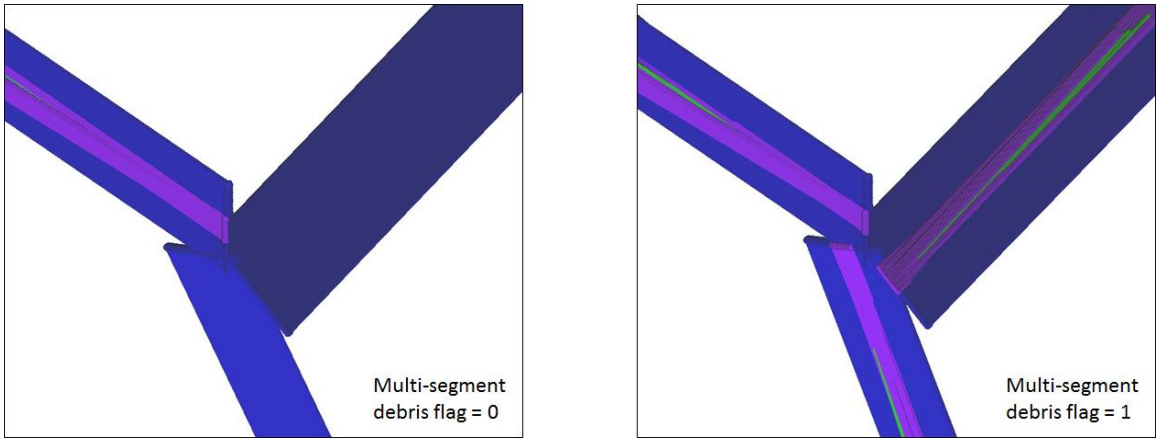


Figure 9-29. Sample Case Fourteen Debris.

10 REFERENCES

1. Rhoads, James E., *Uncontained Engine Debris Analysis Summary Report*, SURVICE-TR-96-040, SURVICE Engineering Company, Aberdeen, Maryland, October 1996.
2. Strausser, Robert L., *FASTGEN 3.2 DB User's Guide*, SURVICE-TR-96-038, SURVICE Engineering Company, Aberdeen, Maryland, October 1996.
3. Butkiewicz, Mark T., *COVART 4.1 DB User's Guide*, SURVICE-TR-96-039, SURVICE Engineering Company, Aberdeen, Maryland, October 1996.
4. Manion, John K., *Uncontained Engine Debris Damage Assessment Model Development - Phase II Audit Trail Report*, SURVICE-TR-98-028, SURVICE Engineering Company, Aberdeen, Maryland, December 1998.
5. Seng, Silvia, John Manion, and Charles Frankenberger, *Uncontained Engine Debris Analysis Using the Uncontained Engine Debris Damage Assessment Model*, DOT/FAA/AR-04/16, Naval Air Warfare Center, Weapons Division, China Lake, California, September 2004.
6. *FASTGEN 6.1: Target Description Document*, JASPO-M-07-03-027, ITT Corporation, Beavercreek, Ohio, 31 March 2010.
7. *COVART 6.7: User's Manual*, JASP-M-14-03-003, Booz Allen Hamilton/Harris, Dayton, Ohio, 31 March 2016.
8. *Design Considerations for Minimizing Hazards Caused by Uncontained Turbine Engine and Auxiliary Power Unit Rotor*, AC 20-128A, U.S. Department of Transportation, Federal Aviation Administration, 25 March 1997.
9. Gunderson, C.O., *Study to Improve Airframe Turbine Engine Rotor Blade Containment*, FAA-RD-77-44, McDonnell Douglas Corporation, July 1977.
10. Lundin, Steven J. and Richard B. Mueller, *Advanced Aircraft Materials, Engine Debris Penetration Testing*, DOT/FAA/AR-03/37, Naval Air Warfare Center, Weapons Division, China Lake, December 2005.
11. Wilde, Paul D. and Chris Draper, *Aircraft Protection Standards and Implementation Guidelines for Range Safety*, AIAA Aerospace Sciences Meeting, 4-7 January 2010, Orlando, Florida, AIAA # 2010-1542.
12. Finnegan, S.A., et al, *A Study of Obliquity Effects on Perforation and Ricochet Process in Thin Plates Impacted by Compact Fragments*, 14th International Symposium on Ballistics, Quebec, Canada, 26-29 September, 1993.

13. Goldsmith, W., *Review: Non-Ideal Projectile Impacts on Targets*, International Journal of Impact Engineering, Vol. 22, 1999, pp. 95-39.
14. Manchor, Joseph and Charles Frankenberger, *Engine Debris Penetration Testing*, DOT/FAA/AR-99/19, Naval Air Warfare Center, Weapons Division, China Lake, California, November 1999.
15. Lundin, Steven J., *Engine Debris Fuselage Penetration Testing, Phase I*, DOT/FAA/AR-01/27, Naval Air Warfare Center, Weapons Division, China Lake, California, August 2001.
16. Lundin, Steven J., *Engine Debris Fuselage Penetration Testing, Phase II*, DOT/FAA/AR-01/27, II, Naval Air Warfare Center, Weapons Division, China Lake, California, September 2002.

11 GLOSSARY

CBULK: a COVART6 input file that defines a target. A target is composed of complex geometric shapes (objects), modeled by using a series of points and simple geometric shapes called elements.

debris category: a combination of disk segments, large fragments, and/or small fragments released during an uncontained failure that represent a type of event; for example, a debris category called “blade” may consist of multiple large and small fragments that represent the debris released from an uncontained failure of a rotor blade (the event type).

debris type: a single rotor disk segment, large fragment, or small fragment released during an uncontained engine failure.

event: an uncontained failure of a turbine engine or any failure which results in the escape of rotor fragments from the engine or APU that could result in a hazard.

event type: see **debris category**.

hazard: a level of harm resulting in the following consequences to the aircraft or its passengers/crew: (a) *Catastrophic* – an occurrence resulting in multiple fatalities, usually with the loss of the airplane; (b) *Severe* – an occurrence resulting in a forced landing, actual loss of aircraft while occupants were on board, or serious injuries or fatalities; (c) *Serious* – an occurrence resulting in substantial damage to the aircraft or second unrelated system, uncontrolled fires, rapid depressurization of the cabin, permanent loss of thrust or power greater than one propulsion system, temporary or permanent inability to climb and fly 1000 feet above terrain along the intended route that results in restricted capability, malfunctions or failures that cause smoke or other fumes that result in a serious impairment, or any temporary or permanent impairment of aircraft controllability.

hazard zone: a zone encompassing all potential debris trajectories from a single release of an event; rotational and fore/aft release variances about the release point are considered

HEAD: a COVART6 input file that contains titles to be used in various output tables.

hypersampling: a rapid method of locating potentially missed components due to the discrete shotlines representing the debris swath

JTYPE: a COVART6 input file that describes component information (e.g., component number, material code, density ratio, Av storage location, PK table numbers, etc.). The components listed must match the target description components defined in the CBULK file.

LOS: a sequenced list of components along a shotline. The COVART6 LOS output file contains specific component data for use in vulnerable area models: 1) group and component identification number, 2) location, 3) thickness, and 4) shotline obliquity angle.

MASTER: the primary simulation control file for COVART6 in integrated mode; for UEDDAM v4.0 and later versions, the MASTER file takes the place of the BASIC and CONTROL files in describing the major input files, program options, shotline analysis parameters, and data file specifications for program execution

MV: when two or more critical components must sustain damage before the aircraft is damaged, these components are referred to as multiply vulnerable (MV), or redundant. The COVART6 MV input file defines how the multiply vulnerable components are combined in a hazard tree, in a form that permits COVART6 to compute P_{haz} of the multiply vulnerable group as a whole.

PK: a COVART6 input file containing the probabilities that define P_{haz} given a hit

probability of hazard (P_{haz}): the probability of an event causing sufficient damage to result in a predefined hazard level; capital letters (HAZ) indicate the damage to the whole system, whereas lowercase letters (haz) indicate the damage is done on a component level.

probability of hit ratio (P_{h}): the probability of a component being hit by a rotor disk segment during an uncontained failure; $P_{\text{h}} = (\text{TRA}/360) \times (\text{SRA}/\text{spread angle})$

spread risk angle (SRA) : the number of degrees of included arc in the plane normal to the plane of rotation that intersects a component being analyzed.

THREAT: a COVART6 input file containing records used to describe the threat; in the case of UEDDAM, the threat is uncontained engine debris in the form of individual fragments.

translational risk angle (TRA) : the number of degrees of included arc in the plane of rotation that intersects a component being analyzed.

UDAIM: a COVART6 analysis option defining a shotline aim point location (in target x, y, z coordinates); multiple UDAIM records are grouped together by a single UDASPECT record

UDASPECT: a COVART6 analysis option specifying a specific target orientation (in terms of azimuth and elevation angles)

APPENDIX A

UEDDAM INTEGRATION WITH COVART 6.X

A-1. Final Changes for COVART 6.2.x Integration of UEDDAM

Integration was conducted in several phases:

- 1) As part of the UEDDAM v4.1 development effort, the FAA penetration equation module in COVART 6.1.1 was updated with a new V_{50} calculation. Code modifications were made to the following files: *error_handler.f90*, *read_master.f90*, *faafrag.f*, *faafragctl.f*, *faafrinit.f*, *faasl.f*, and *initlib.f*.
- 2) As part of the UEDDAM v4.3 development effort, the FAA penetration equation module in COVART 6.1.1 was again updated with a new V_{50} calculation methodology. Code modifications were made to the following files: *faafrag.f*, *faafragctl.f*, *faafrinit.f*, *init_EACA_stub.f*, *initlib.f*, and *inputi.f*.

File: Covart6\error_handler.f90

-modified definition of the local variable parameter iNumMessages1

```
! emp 22 Aug 2012 Added a new error message, requiring a change to the parameter
iNumMessages1
! INTEGER, PARAMETER :: iNumMessages1 = 514 - 6*iNumMessages
INTEGER, PARAMETER :: iNumMessages1 = 515 - 6*iNumMessages
```

-added new error message for UEDDAM gamma value check

```
, ' ERROR: LBP AND UBP CAN NOT BOTH = 0 IF NUMSTP IS > 0 ' & !511
, ' ERROR: NVIEW ON ENCOUNTER REC MUST = 1 FOR RADIAL ZONES ' & !512
, ' ERROR: RADIAL ZONES ONLY COMPATIBLE W SINGLE FRAG/RAY APPROACH ' & !513
, ' ERROR: INPUT FOR UDSHOT FIELD ON UEDDAM RECORD NOT RECOGNIZED ' & !514
! emp 22 Aug 2012 New message added for UEDDAM
, ' ERROR: GAMMA VALUES MUST BE BETWEEN 0.0 AND 1.0 ' & !515
```

File: Covart6\read_master.f90

- added new common block UDFAAPEN for access to UDGAM
- added UDGAM to UEDDAM record read statement
- added checks for UDGAM

```

! emp 6 Aug 2012 Created new common block for UDGAM (penetration
! equation variable input)
COMMON /UDFAAPEN/ UDGAM
REAL :: UDGAM(5)

integer :: SLT_Level

CHARACTER * 80 :: CF971, CF975

! emp 6 Aug 2012 Added UDGAM to UEDDAM record
! read(RECORD, '(2A8)') CDRAG, UDSHOT
read(RECORD, '(2A8,5F8.2)') CDRAG, UDSHOT, &
UDGAM(1), UDGAM(2), &
UDGAM(3), UDGAM(4), &
UDGAM(5)

call fieldconvert(CDRAG)

! emp 22 Aug 2012 Added checks for UDGAM
IF ((UDGAM(1) .LT. 0.0) .OR. (UDGAM(1) .GT. 1.0)) THEN
iErrorLevel = eLError
WRITE(cMessage, '(A)') RECDTA
CALL ErrorWrite(515)
ELSE IF ((UDGAM(2) .LT. 0.0) .OR. (UDGAM(2) .GT. 1.0)) THEN
iErrorLevel = eLError
WRITE(cMessage, '(A)') RECDTA
CALL ErrorWrite(515)
ELSE IF ((UDGAM(3) .LT. 0.0) .OR. (UDGAM(3) .GT. 1.0)) THEN
iErrorLevel = eLError
WRITE(cMessage, '(A)') RECDTA
CALL ErrorWrite(515)
ELSE IF ((UDGAM(4) .LT. 0.0) .OR. (UDGAM(4) .GT. 1.0)) THEN
iErrorLevel = eLError
WRITE(cMessage, '(A)') RECDTA
CALL ErrorWrite(515)
ELSE IF ((UDGAM(5) .LT. 0.0) .OR. (UDGAM(5) .GT. 1.0)) THEN
iErrorLevel = eLError
WRITE(cMessage, '(A)') RECDTA
CALL ErrorWrite(515)
END IF

```

File: libraries\FAAPEN\faafrag.f

- added second array, DYNshr0, to MTL1UDFAA common block
- added new common block for UDFAAPEN for access to UDGAM
- added DYNshr0, UDGAM, CosObl, and GamExp variables


```

c emp 2015 Added second array, DYNSHR0, to MTL1UDFAA common block
COMMON /MTL1UDFAA/ DYNSHR(MXMT), DYNSHR0(MXMT)
clag
COMMON /THRT2FAA/ DIA (MAXWGT,4), LSHAPE (MAXWGT), FL (MAXWGT),
&
LF (MAXWGT)
C emp 22-Aug-2012 Added COMMON UDFAAPEN
COMMON /UDFAAPEN/ UDGAM(5)

c emp 6 Aug 2012 Added CosObl to declarations
c REAL RHOt, TlEq, WT, V, AreaP, WR, V50, Vr
REAL RHOt, TlEq, CosObl, WT, V, AreaP, WR, V50, Vr

CLAG TYPE DECL FOR LABELED COMMON VARS
INTEGER IEXTBP, MTFRAG
INTEGER LF, LSHAPE
REAL WFRAG, VFRAG, RLDFRG, APFRAG, RLFRAG
c emp 2015 Added second array, DYNSHR0
REAL BHN, RHO, SIGY, SIGU, TAUS, DYNSHR, DYNSHR0
REAL DIA, FL
C emp 22-Aug-2012 Added UDGAM
REAL UDGAM

CLAG LOCAL VAR DECLARATIONS
INTEGER KF, NRIC, N
REAL VV, RHOF, SIGF, AP
c emp 2015 Modified local variable rmGd
REAL rmkg, Tm, rmGd50, rmL, V50m, Q4
c emp 22 Aug 2012 Added local variable GamExp
c emp 2015 Added new local variable rmGd0
REAL GamExp, rmGd0

CLAG SUBROUTINE ARGUMENT DECLARATIONS
INTEGER FrID, TgtMat
c emp 6 Aug 2012 Added CosObl to declarations
c REAL RHOt, TlEq, WT, V, AreaP, WR, V50, Vr
REAL RHOt, TlEq, CosObl, WT, V, AreaP, WR, V50, Vr

```

- updated penetration equation calculation based on Reference AIAA 2010-1542
- updated penetration equation calculations to use 2 C_s values

```

c emp 6 Aug 2012
c emp Updated in 2012 based on Reference AIAA 2010-1542
c emp
c emp gam = obliquity modification (unitless)
c emp
c emp V50 = sqrt( (2. * L * Cs * t**2) / (m * (cos(theta))**(2. * gam)) )
CLAG
CLAG **** NOTE FOR UEDDAM USE ONLY, NOT ADDRESSING COMBINED PLATES ****
CLAG
CLAG Factors used in unit conversion:
CLAG 7000 grains/pound, 2.205 pounds/kilogram, 39.37 inches/meter,
CLAG 6894.757 Pascals/psi, 3.281 feet/meter

rmkg = WT / 7000. / 2.205
c emp 6 Aug 2012 Changed definition of Tm for updated penetration equations
c Tm = TlEq / 39.37
Tm = (TlEq * CosObl) / 39.37
rmGd50 = DYNshr0(TgtMat) * 6894.757
rmGd0 = DYNshr0(TgtMat) * 6894.757
rmL = RLFRAG / 39.37
c emp 6 Aug 2012 Added check for obliquity angle and assignment of Gamma
IF ( CosObl .GE. 0.866 ) THEN
    GamExp = 2. * UDgAM(1)
ELSEIF ( CosObl .GE. 0.707 ) THEN
    GamExp = 2. * UDgAM(2)
ELSEIF ( CosObl .GE. 0.5 ) THEN
    GamExp = 2. * UDgAM(3)
ELSEIF ( CosObl .GE. 0.259 ) THEN
    GamExp = 2. * UDgAM(4)
ELSE
    GamExp = 2. * UDgAM(5)
ENDIF

c emp 22 Aug 2012 Updated calculation of V50
c V50m = SQRT(2. * rmL * rmGd * Tm**2. / rmkg)
c emp 2015 Added IF statements for V50 calculations; the IF statements
c check first to see if the two Cs values entered by the user
c are the same - if they are, V50 is calculated as before. If
c not, V50 is calculated using Cs_0. If this new V50 value is
c greater than the impact velocity, the debris is stopped (i.e.,
c Vr = 0); otherwise, V50 is recalculated using Cs_50 and this
c V50 is then used in the Vr calculations.
IF ( ABS(rmGd50 - rmGd0) .LE. 0.01 ) THEN
    V50m = SQRT((2. * rmL * rmGd50 * Tm**2.)/(rmkg * CosObl**GamExp))
    V50 = V50m * 3.281
ELSE
    V50m = SQRT((2. * rmL * rmGd0 * Tm**2.)/(rmkg * CosObl**GamExp))
    V50 = V50m * 3.281
    IF (VV .GT. V50) THEN
        V50m = SQRT((2. * rmL * rmGd50 * Tm**2.)/(rmkg * CosObl**GamExp))
        V50 = V50m * 3.281
    END IF
ENDIF

```

File: libraries\FAAPEN\faafragctl.f

-added DYNshr0 to MTL1UDFAA common block and variable declarations
 -added CosObl to subroutine parameters and call

```

c emp 2015 Added second array, DYNSHR0, to MTL1UDFAA common block
COMMON /MTL1UDFAA/ DYNSHR(MXMT), DYNSHR0(MXMT)
CLAG END CHANGE
COMMON /THRT2FAA/ DIA (MAXWGT,4), LSHAPE (MAXWGT), FL (MAXWGT),
& LF (MAXWGT)
C
C Global Variables
C
INTEGER JRES
REAL CD, FLUI, VMAX
CLAG ADDED DYNSHR FOR FAA PENETRATION EQNS
c emp 2015 Added second array, DYNSHR0
REAL BHN, RHO, SIGY, SIGU, TAUS, DYNSHR, DYNSHR0
CLAG END CHANGE

!

c emp 6 Aug 2012 Added CosObl
c REAL TlEq, WT, V, AreaP, WR, V50, Vr
REAL TlEq, CosObl, WT, V, AreaP, WR, V50, Vr

!

c emp 6 Aug 2012 Added CosObl to call
c CALL FAAfrag ( FrID, TgtMat, RHOt, TlEq, WT, V, AreaP,
CALL FAAfrag ( FrID, TgtMat, RHOt, TlEq, CosObl, WT, V, AreaP,
& WR, V50, Vr)

```

File: libraries\FAAPEN\faafrinit.f

- added DYNSHR0 to MTL1UDFAA common block and variable declarations
- added common block for new penetration equations
- added UDGAM and UDGAMP as global and local variables, respectively

```

c emp 2015 Added second array, DYNSHR0, to MTL1UDFAA common block
COMMON /MTL1UDFAA/ DYNshr(MXMT), DYNSHR0(MXMT)

!

c emp 22 Aug 2012 Added UDFAAPEN COMMON block for new penetration equations
COMMON /UDFAAPEN/ UDgAM(5)

C
C Global Variables
C
CLAG ADDED DYNshr FOR FAA PENETRATION EQNS
C REAL BHN, RHO, SIGY, SIGU, TAUS
c emp 2015 Added second array, DYNSHR0
REAL BHN, RHO, SIGY, SIGU, TAUS, DYNshr, DYNSHR0
CLAG END CHANGE
REAL E, BULK, DENAL, RA, U
REAL CCBF, CBF, CH, CCF, FCM
REAL FG, FV, FT, FS, FLUI
REAL PROL, PRON, PROTD, DENRAT
REAL DIA, FL, DIA2, FIRE
c emp 22 Aug 2012 Added UDgAM for new penetration equations
c emp REAL CD, VMAX, AM, B
REAL CD, VMAX, AM, B, UDgAM
INTEGER LSHAPE, LF

C
C Local Variables
C

CLAG ADDED DYNshRP FOR FAA PENETRATION EQNS
c emp 2015 Added second array, DYNshR0P
REAL BHNP(MXMT), RHOP(MXMT), SIGYP(MXMT), SIGUP(MXMT),
& TAUSP(MXMT), DYNshRP(MXMT), DYNshR0P(MXMT)
CLAG END CHANGE

!

c emp 22 Aug 2012 Added UDgAMP for new penetration equations
c emp REAL CDP(MXSHAP)
REAL CDP(MXSHAP), UDgAMP(5)

```

-added initialization for DYNSHR0 and UDgAM variables

```

DYNshr(I) = DYNshRP(I)
c emp 2015 Added initialization for DYNshR0
DYNshR0(I) = DYNshR0P(I)

!

c emp 22 Aug 2012 Added initialization for UDgAM values
DO 400 I = 1, 5
UDgAM(I) = UDgAMP(I)
400 CONTINUE

```

File: libraries\FAAPEN\faasl.f

-added FAA penetration equation variables COBL, WR, V50, and VR to FAAFragCtl call

```

c emp 6 Aug 2012 New call for FAA Fragments
c CALL FAAFragCtl ( FrID, TgtMat, LOS, WPs, VPs, AreaP,
CALL FAAFragCtl ( FrID, TgtMat, LOS, COBL, WPs, VPs, AreaP,
& WR, V50, VR)

```

File: Covart6\init_EACA_stub.f

- modified MTL1UDFAA common block definition
- added TDYNSHR0 to variable declarations

```

C emp 2015 Changed common block name from MATA1UD to MTL1UDFAA
C Added second array, DYNSHR0, to common block
COMMON /MTL1UDFAA/ DYNSHR(MXMT), DYNSHR0(MXMT)

!

c emp 2015 Added second array, TDYNSHR0
DIMENSION TDYNSHR(MXMT), TDYNSHR0(MXMT)

```

- added definition of data for TDYNSHR0
- modified TRHO data definition

```

C emp 2015 Added definition of data for new TDYNSHR0 array
DATA TDYNSHR0 / 6* 188500.0 , 130529.4 , 8* 40029.0 ,
+ 5* 188500.0 , 9* 40029.0 , 4* 188500.0 , 3* 40029.0 , 36403.2 ,
+ 7* 188500.0 ,130529.4 , 36403.2 , 174039.2 , 3* 188500.0 /

!

C emp 2015 Modified TRHO(46) and TRHO(47) for composite and
C inconel, respectively; they were previously set
C equal to 0.0
DATA TRHO / 1980.0 , 1980.0 , 1980.0 , 1980.0 , 1980.0 ,
5 1980.0 , 1130.0 , 701.0 , 672.0 , 672.0 , 672.0 ,
6 686.0 , 708.0 , 693.0 , 430.0 , 1980.0 , 1800.0 ,
7 2240.0 , 2760.0 , 4727.0 , 188.0 , 235.0 , 325.0 ,
8 295.0 , 309.0 , 485.0 , 620.0 , 240.0 , 190.0 ,
8 4669.0 , 1980.0 , 1980.0 , 4256.0 , 329.0 , 175.0 ,
8 91.0 , 392.0 , 380.0 , 392.0 , 392.0 , 392.0 ,
8 3*4214.0 , 1141.0 , 390.6 , 2135.0 , 3* 0.0/

```

- added initialization of DYNSHR0

```

DYNSHR(I) = TDYNSHR(I)
C emp 2015 Added second array, DYNSHR0, initialization
DYNSHR0(I) = TDYNSHR0(I)

CLAG
601 CONTINUE

```

File: Covart6\initlib.f

- modified definition of MTL1UDFAA common block
- added common block UDFAAPEN for access to UDGAM

```

c emp 2015 Changed common block name from MATAL1UD to MTL1UDFAA
c      Added second array, DYNshr0, to common block
COMMON /MTL1UDFAA/ DYNshr(MXMT), DYNshr0(MXMT)

!

c emp 22 Aug 2012 Added UDFAAPEN COMMON block for new penetration equations
COMMON /UDFAAPEN/ UDgAM(5)

```

-added UDgAM and DYNshr0 to FAAFrInit call

```

      CALL FAAFrInit ( BHN, RHO, SIGY, SIGU, TAUS,
&                   DYNshr, E, BULK, DENAL, RA, U, CF,
&                   CCBF, CBF, CH, CCF, FCM, FG, FV,
&                   FT, FS, JRES, PROL, PRON, PROTD,
&                   DENRAT, DIA, FLUI, CD,
c emp 22 Aug 2012 Added UDgAM
c emp 2015 Added DYNshr0
c emp      &                   LSHAPE, FL, LF, DIA2, AM )
&                   LSHAPE, FL, LF, DIA2, AM, UDgAM, DYNshr0 )

```

File: Covart6\input.f

-modified definition of MATAL1 and MTL1UDFAA common blocks

```

c emp 2015 Removed DYNshr from MATAL1 common block
c      COMMON /MATAl1/ RHO(MXMT), DYNshr(MXMT)
COMMON /MATAl1/ RHO(MXMT)
c emp 2015 Added MTL1UDFAA common block; added second array,
c      DYNshr0, to common block
COMMON /MTL1UDFAA/ DYNshr(MXMT), DYNshr0(MXMT)
CHARACTER * 8 HER, DER

```

-modified MCHECK logic

```

cflag need to check against IFAAF array for UEDDAM materials
      IF (ITHSP .NE. 2) THEN
        JCHECK = JTCGF(JM)
      ELSE
c emp 2015 Added MCHECK logic for UEDDAM
        JCHECK = IFAAF(JM)
        MCHECK = IFAAF(JM)
        MCHECK = MCHECK/100
      END IF
cflag end change

```

-modified read statement for MATPROP record

```

C **** MATPROP input record for UEDDAM ****
C **** ****
C **** ****
      ELSE IF ( DATTyp .EQ. 'MATPROP' ) THEN
        READ( RECORD, '(I8)' ) IMAT
        IF ( ( IMAT .GT. 0 ) .AND. ( IMAT .LT. 50 ) ) THEN
c emp 2015 Added DYNShR0 to READ
          READ( RECORD, '(8X,2A8,3F8.1)' ) HER(IMAT), DER(IMAT),
            & RHO(IMAT), DYNShR(IMAT), DYNShR0(IMAT)
          JTCGF( IMAT ) = 100 * IMAT
        END IF

```

After modifications were made to COVART, SURVICE submitted SCRs to the JASPO SCR site (<https://www.bahdayton.com/scr/login.aspx>): COVART 6.2.1_11 and COVART 6.2.1_28. These SCRs are detailed below.

SCR COVART 6.2.1_11

Subject:

FAA Penetration Equation Library Update

Deficiency Description:

As part of the most recent UEDDAM update, a change was made to the FAA penetration equations that requires a new set of variables be read in through the MASTER file. Additionally, the FAA penetration equation library requires updates.

Suggested Solution:

The UEDDAM entry in the MASTER file will need to be modified to include five additional parameters:

UEDDAM DRAGOFF SHOT G1 G2 G3 G4 G5

where G1 through G5 are real number constrained to be between 0.0 and 1.0. Additionally, the FAA penetration equation library will need to be modified to use these new parameters. All required changes have been made in the attached source code files (X). Each change has been prefixed with emp and an August 2012 date for ease of location.

SCR COVART 6.2.1_28

Subject:

FAA Penetration Equation Module Update

Deficiency Description:

An update has been made to the FAA penetration equation module for use by UEDDAM. In addition to the changes made within the module, in order to implement the change, it was necessary to make modifications to files outside of the FAA penetration equation module itself. Namely, a change needed to be made to the routine that reads in the JTYPE file so that an additional variable could be read in from the MATPROP record. In addition, a few common blocks were found to either be named incorrectly or contain incorrect variables, which was preventing the proper passage of values between subroutines.

Suggested Solution:

All source code files requiring modifications have been attached to this SCR. All changes are prefixed in the code with "c emp 2015". Note that some of the requested changes are duplicated in SCR COVART 6.2.1_27, but have been provided for completeness.

A-2. Final Changes for COVART 6.1 Integration of UEDDAM

Integration was conducted in several phases:

- 1) Integration into COVART 6.1: ITT integrated the UEDDAM modifications to COVART 6.0.1 into COVART 6.1 with documentation of changes (see Section A-2, *Final Changes for COVART 6.0.1 Integration of UEDDAM*). Additionally, ITT developed new UEDDAM-specific modules (*UEDDAMSINGL.f90*, *Process_UEDDAM.f90*, and *SetAutoFailPk.f90*), JTYPE records, and MASTER file keywords to allow UEDDAM to work with COVART6 in integrated mode (off the shelf).
- 2) Verification of COVART 6.1 (Integrated Mode) for use with UEDDAM: SURVICE Engineering, as part of the UEDDAM 4.0 development effort, discovered several problems with the COVART 6.1 code that would not allow it to work properly with UEDDAM. Code modifications were made to the following files: *UEDDAMSINGL.f90* and *Process_UEDDAM.f90*. Details follow.

File: Covart6\UEDDAMSINGL.f90

All code modifications (minus those implemented solely for debugging purposes) are detailed below.

- added access to global variables RMAX and RMIN
- added local variables necessary for removing components from the shotline that are “behind” the release point

```
! emp Added RMAX and RMIN for UEDDAM v4.0
      use global_data, ONLY : DSAS, DCAS, DSES, DCES, RSA, RCA, RSE, &
                           RCE, IF43, RMAX, RMIN
! emp End of addition
:
! emp Added following variables for UEDDAM v4.0
!
      REAL(Kind=RK2) :: dx1, dx2, entsec, newlos, OrigAimX, OrigAimY
      REAL(Kind=RK2) :: OrigAimZ, TransAimX, dirvecx, dirvecy, vsinel
      REAL(Kind=RK2) :: dirvecz, transdirvecx, vcosaz, vsinaz, vcosel
      integer :: intremove, J, K, empdebug
      TYPE(ProcShotline_Type) :: temp_sl_data
!
! emp End of addition
```

- added call to TRANSANG function to make the cosine and sine of the view azimuth and elevation angles available
- added initialization of Dir and RTGDEF variables

```

! emp Added debug flag definition for UEDDAM v4.0 (set empdebug = 0 to turn
! off all the extra write statements); also added a call to the TRANSANG
! function to have the view cos/sin az/el values available; finally, added
! initialization of Dir and RTGDEF as in covart6.f90
!
empdebug = 0
IF ((empdebug == 1) .AND. (IVCUR == 1)) OPEN (45, FILE='debug_trace.txt')

CALL TRANSANG(RAZ(IVCUR), REL(IVCUR), vsinaz, vcosaz, vsinel, vcosel)
dirvecx = -1.0 * vcosaz * vcosel
dirvecy = -1.0 * vcosel * vsinaz
dirvecz = -1.0 * vsinel
transdirvecx = (((dirvecx * vcosaz) + (dirvecy * vsinaz)) * vcosel) +
(dirvecz * vsinel)

Dir%RSA = RSA
Dir%RCA = RCA
Dir%RSE = RSE
Dir%RCE = RCE

RTGDEF%Xmax = RMIN
RTGDEF%Xmin = RMAX
RTGDEF%Ymax = RMIN
RTGDEF%Ymin = RMAX
RTGDEF%Zmax = RMIN
RTGDEF%Zmin = RMAX

! emp End of addition

```

-modified storage allocation and calls to shotline routines to store shotline data in a temporary storage variable, as opposed to Shotline_Data

```

! emp Modified to allocate storage to temp_sl_data rather than Shotline_Data
!
IF (ALLOCATED(temp_sl_data%LOS)) &
deallocate (temp_sl_data%LOS)
! intersection storage
allocate (temp_sl_data%LOS(Hits(I)), STAT=iStatus)
! emp End of modification
:
! emp Modified to put data in temp_sl_data rather than Shotline_Data
!
CALL FGRT_SLReturn(I,temp_sl_data, Hits(I), Errors)
ELSE
CALL BRLCAD_RT_SLReturn(I,temp_sl_data, Hits(I), Errors)
!
! emp End of modification

```

-added an algorithm that removes components from the temporary shotline that are “behind” the release point and then stores the resulting shotline information in Shotline_Data

```

!temp Added following section of code to remove components on the shotline
! that are "behind" the release point - needed for UEDDAM v4.0

      intremove = 0
      J = 1
      OrigAimX = RXSHT4(IVCUR, I)
      OrigAimY = RYSHT4(IVCUR, I)
      OrigAimZ = RZSHT4(IVCUR, I)
      TransAimX = (((OrigAimX * vcosaz) + (OrigAimY * vsinaz)) * vcosel) +
(OrigAimZ * vsinel)
      DO
        IF (transdirvecx < 0) THEN
          dx1 = TransAimX - temp_sl_data%LOS(J)%RX
          dx2 = TransAimX - temp_sl_data%LOS(J)%RX +
temp_sl_data%LOS(J)%LOS
        ELSE
          dx1 = temp_sl_data%LOS(J)%RX - TransAimX
          dx2 = temp_sl_data%LOS(J)%RX - temp_sl_data%LOS(J)%LOS -
TransAimX
        END IF

! If the difference between the x-value of the entrance and the x-value of the
! aim point (i.e., dx1) is less than zero, or if the difference between the x-
! value of the aim point and the x-value of the exit (i.e., dx2) is less than
! zero, the intersection must either be removed or its LOS must be reduced.

        IF (dx1 <= 0 .OR. dx2 <= 0) THEN
          IF (dx1 < 0 .AND. dx2 >= 0) THEN

! Shrink the line-of-sight and normal thickness for this component as the
! release point is inside the component

            newlos = temp_sl_data%LOS(J)%LOS + dx1
            entsec = temp_sl_data%LOS(J)%ISEC1
            temp_sl_data%LOS(J)%LOS = newlos
            temp_sl_data%LOS(J)%RTH = (1000.0 / entsec) * newlos
            J = J + 1

          ELSE

! Remove the intersection....it's behind the release!

            intremove = intremove + 1
            IF (Hits(I) - intremove < J) EXIT
            DO K = J, Hits(I) - intremove
              temp_sl_data%LOS(K) = temp_sl_data%LOS(K+1)
            END DO
          END IF
        ELSE
          J = J+1
        END IF
      IF (J - 1 >= Hits(I) - intremove) EXIT

    ENDDO

! Reduce the number of total intersections for the shotline by the
! number of components removed.

      Hits(I) = Hits(I) - intremove

! Now allocate the actual shotline array to be used throughout the
! remaining code

      IF (ALLOCATED(Shotline_Data%LOS)) &
        deallocate (Shotline_Data%LOS)
! intersection storage
      allocate (Shotline_Data%LOS(Hits(I)), STAT=iStatus)
      IF (iStatus /= 0) THEN
        iErrorLevel = e1Error
        CALL ErrorWrite(124) ! memory allocation error
      END IF

```

```

DO J = 1, Hits(I)
  Shotline_Data%LOS(J) = temp_sl_data%LOS(J)
END DO
:
! emp End of addition

```

-added a deallocation statement for the temporary hotline data storage

```

! emp Deallocate temporary storage
!
! IF (ALLOCATED(temp_sl_data%LOS)) deallocate (temp_sl_data%LOS)
!
! emp End of addition

```

File: Covart6\Process_UEDDAM.f90

All code modifications (minus those implemented solely for debugging purposes) are detailed below.

-modified definition of AFACT

```

SLFACT = 1.0
!
! need to look at 0.25 - may be different for UEDDAM
! emp It is different for UEDDAM...changed from 0.25 * 144
! to 1.0 * 144 (or simply, 144)
!
! AFACT = 144.0 ! Area factor
! emp End of change

```

-added ISRAY definition for HOLESIZE file

```

! Process this hotline
!
WFRAG = RMAS(CurView, J)
MTFRAG = IMATTP(CurView, J)
VFRAG = RVEL(CurView, J)
APFRAG = APFRAGUD(CurView, J)
RLFRAG = RLFRAGUD(CurView, J)
RLDFRG = FL(1)
ISHFRG = LSHAPE(1)
! emp Added ISRAY definition for HOLESIZE file
ISRAY = ISRAY + 1
! emp End of addition

```

- 3) Submittal of SCRs to repair COVART 6.1: SURVICE submitted three SCRs to the JASPO SCR site (<https://www.bahdayton.com/scr/login.aspx>): COVART 6.0.2_30, COVART 6.0.2_35, and COVART 6.1.1_4. The SCRs are detailed below.

SCR COVART 6.0.2_30

Subject:

COVART6 Integrated Mode for UEDDAM v4.0 (Note – this applies to COVART v6.1)

Deficiency Description:

1. Missing variable initializations
2. Missing variable declarations
3. Algorithm needed to remove components on the shotline that are behind the release point
4. Target PK not being accumulated over all UDASPECT/UDAIM records in a single MASTER file

Suggested Solution:

Attached code (*NewUEDDAMSNGL.f90* and *NewProcess_UEDDAM.f90*) addresses the problems identified in 1, 2, and 3 of the deficiency description. More time would be needed to address the problem identified in 4.

SCR COVART 6.0.2_35

Subject:

AVDATOUT Record Not Working Properly (COVART 6.1)

Deficiency Description:

COVART does not generate a file with the name specified on the AVDATOUT record in the MASTER file.

Suggested Solution:

Ron White located the problem: “it’s the result of a variable in read_master that is not getting incremented but that is used as a test to open the AVDAT file.”

SCR COVART 6.1.1_4

Subject:

Shotline File Generation for UEDDAM v4.0

Deficiency Description:

The UEDDAM Visualizer requires a shotline file for some of its built-in features to work properly. With the upgrade to COVART integrated mode, we no longer have the option to output a shotline file if needed (as the RESTART capability does not work with UEDDAM mode). Note that this SCR was communicated to Tim Staley in October 2010 as one of the upgrades that needed to take place for UEDDAM v4.0.

Suggested Solution:

We would like to add a new variable to the UEDDAM line in the MASTER file. The additional variable should be added to the end of the line and its options should be SHOT or NOSHOT. For example, the following line could appear in the MASTER file:

UEDDAM DRAGOFF SHOT

The variable needs to be passed to the UEDDAMSNGL.f90 module and will tell the code whether or not to generate a Shotline file. I’ve attached a sample file of what the Shotline file could look like – note that the only restrictions on the file are that it must contain the following information (i.e., format is not important, as long as we know what the format is):

Per Fragment (Shotline) ray:

- * Mass
- * Velocity
- * Fragment Material
- * Impact Count

Per Impact:

- * Component ID
- * Air Gap
- * LOS
- * Obliquity

- 4) Integration of Repairs into COVART 6.2: ITT implemented the code changes documented in #2 above, addressed the remaining SCR issues, and debugged a couple of newly discovered problems with the Autofail routines.

Development and testing by SURVICE during this iteration were performed on 2.00GHz Dell Latitude D620 Microsoft Windows XP Professional platform. The Windows version was built and tested using Microsoft Visual Studio 2005 Professional Edition with the integrated Intel Fortran compiler.

A-3. Final Changes for COVART 6.0.1 Integration of UEDDAM

Integration was conducted in several phases:

- 1) Integration into COVART 5: Integrated the UEDDAM modifications to COVART 4.2 into COVART 5 with documentation of changes (see Section A-4, *Correspondence: UEDDAM Appendix A with Annotated COVART 5 Modifications - 3 October 2008*). ITT reviewed changes, incorporated some into updated source code, and provided notes on suggested alternative changes where conflicts existed or coding standards needed to be applied (see Section A-3, *Correspondence: Summary of COVART 6 Changes for UEDDAM Based on COVART 5 Modifications - 12 January 2009*).
- 2) Verification of documented change requests: Comparison of the notes provided from ITT and the Appendix A documentation of COVART 5 changes showed that the two new materials were not added; two dimensioning parameters in the parm.inc file were not updated, and the initialization of many parameters had been relocated and/or zeroed.
- 3) Implementation of changes per ITT notes: Two primary changes were requested. First, additional common blocks were utilized rather than altering existing blocks; common blocks utilized within the library routines were given modified names. Second, storage of the fragment presented area (APFRAG) and fragment mass (WFRAG) values were altered in order to avoid a precision change to the EXOB variable. This required changing how the OUNPKA and related ASCII and binary shotline files (SHOTLINE and SHOTBIN) were written, as well as how the binary SHOTBIN file was interpreted.
- 4) Updates to flag parameters and variable access: In order to get the existing code to run with the above changes, common blocks had to be added/modified in several routines, and flag parameters utilized by the previous version had to be updated to reflect the new ITHSP=2 value now used to indicate a UEDDAM run in COVART legacy mode.
- 5) Input reformatting: Input deck changes for COVART 6 were identified and applied to the UEDDAM test cases.
- 6) Naming convention changes for code builds: There were naming conflicts that had to be resolved in order to satisfy the constructs of the multiple build environments being used, particularly to support the Intel FORTRAN compiler for the Windows build without altering the Absoft and Linux build capabilities.

- 7) Updates to FASTGEN routines for new SHOT4/VIEW4 processing: The routines for processing these records did not function correctly. They were iteratively modified with ITT to operate correctly and produce datasets identical to the VIEW3 processing they were intended to replace.
- 8) Updates for HOLESIZE I/O and processing: Common blocks and parameter declarations were added to upper-level routines to make this data accessible.

Development and testing during this iteration were performed on 2.54GHz Dell Precision M6400 Microsoft Windows Vista Business platforms. The Windows version was built and tested using Microsoft Visual Studio 2005 Professional Edition with the integrated Intel Fortran compiler. The Linux version was built and tested using a Fedora 10 32-bit Linux virtual machine (using VMware) with the GNU Fortran compiler. Sets of makefiles were created for production and debug builds of both versions.

Listed below on a file-by-file basis are the modifications implemented for UEDDAM execution within COVART 6.0.1.

File: Covart6\areach.f

Now storing the number of components on the ray and the fragment weight in grains (WFRAG) in the IECD variable in the shotline header. In the shotline file, IECD will have updated WFRAG values in the “high end” of the integer (7 digits), and the number of components on the ray value in the “low end” of the integer (last 3 digits). Assumes MAXINT value of 2,147,483,647, which will allow the range of 0 to 2,147,482 for WFRAG and 0 to 999 for number of components on the ray. Fragment presented area (APFRAG) value is now stored directly in the single-precision EXOB variable.

-added access to new common blocks EXTBPDU and END2 for access to APFRAG, RLFrag, and ITHSP variables

```

cflag added access to APFRAG, RLFrag
COMMON /EXTBPDU/ APFRAG, RLFrag
cflag end change
:
cflag added access to ITHSP
COMMON /END2 / ITHSP, IUNGRD
cflag end change

```

-added local temp variable IWFTMP for use when converting frag weight from shotline file variable IECD

```

cflag added local temp variable
INTEGER IWFTMP
cflag

```


-added comment info for interpreting UEDDAM ray headers

```
CLAG
CLAG NOTE: FOR UEDDAM RAY HEADER:
CLAG IECO = FRAG WEIGHT IN GRAINS X 1000 +
CLAG NUMBER OF COMPONENTS ON RAY
CLAG EXOB = FRAGMENT PRESENTED AREA, SQ. INCHES
CLAG
```

-added conversion code for new store of fragment weight with number of components on ray in IECO variable, store of fragment presented area directly in EXOB variable, and conditional for extracting RLFrag value from CTH shotline variable

```
clag for ueddram
clag ITN = IECO(LL)
      IF ( ITHSP .NE. 2 ) THEN
          ITN = IECO(LL)
      ELSE
          IWFTMP = INT(IECO(LL)/1000)
          ITN = IECO(LL) - (IWFTMP * 1000)
          APFRAG = EXOB(LL)
      END IF
clag end change
:
clag added for ueddram
clag DIRX = CTH(LL)
      IF ( ITHSP .NE. 2 ) THEN
          DIRX = CTH(LL)
      ELSE
          DIRX = 0.0
          RLFrag = CTH(LL)
      END IF
clag end change
```

-updated code for using extracted fragment weight from IECO variable

```
clag WFRAG = SOLANG
clag for ueddram
      IF ( ITHSP .NE. 2 ) THEN
          WFRAG = SOLANG
      ELSE
          WFRAG = FLOAT(IWFTMP)
      END IF
clag end change
```

File: Covart6\areasb.f

-added access to new common block UEDDAMIO for access to IOHOLE variable

```
cdrh for UEDDAM
COMMON /UEDDAMIO/ IOHOLE
cdrh
```

-added file open statement for holesize output

```
cdrh for UEDDAM -- HOLESIZE
      IF ( NUMVW .EQ. 1 ) THEN
C      OPEN HOLE SIZE OUTPUT FILE
      OPEN ( IOHOLE, FILE='HOLESIZES' )
      ENDIF
cdrh
```

File: Covart6\areasd.f

-added access to new common block END2 for access to ITHSP variable

```
clag added access to ITHSP
      COMMON /END2 / ITHSP, IUNGRD
clag end change
```

File: Covart6\bploc.f

-added check for valid ITHSP=2 UEDDAM legacy mode processing

```
clag need to check for ITHSP=2 valid UEDDAM legacy mode processing
c      IF ( ITHSP .EQ. 0 ) THEN
      IF ( ITHSP .EQ. 0 .OR. ITHSP .EQ. 2 ) THEN
clag end change
:
clag need to check for ITHSP=2 valid UEDDAM legacy mode processing
clag ?? or will ICOVBX value stop entry ??
c      IF ( (ICOVBX .EQ. 1) .AND. (ITHSP .EQ. 0) ) THEN
      IF ( (ICOVBX .EQ. 1) .AND. (ITHSP .EQ. 0 .OR. ITHSP .EQ. 2) ) THEN
clag end change
```

File: Covart6\covart5.f

-added access to new common blocks MATAL1UD, MATUSEUD for access to DYNshr, IFAAF variables

```
clag added for UEDDAM
      COMMON /MATAL1UD/ DYNshr(MXMT)
clag end change
:
clag added for UEDDAM
      COMMON /MATUSEUD/ IFAAF(MXMTCD)
clag end change
```

File: Covart6\covart6.f90

-added access to new common block UEDDAMIO for access to IOHOLE variable

```
! drh for UEDDAM
COMMON /UEDDAMIO/ IOHOLE
! drh
```

-added type declaration for IOHOLE variable

```
! drh for UEDDAM
integer :: IOHOLE
! drh
```

-set value for holesize output file

```
! drh for UEDDAM
IOHOLE = 12
! drh
```

File: Covart6\faasl.f

-added access to new common block END2 for access to ITHSP variable

```
clag added for UEDDAM
COMMON /END2 / ITHSP, IUNGRD
clag end change
```

-added loop case for UEDDAM last shotline intersection processing

```
clag Not sure if this mimics old behavior or not,
clag but would think you should still process last shotline
clag intersection for UEDDAM run as well
ELSE IF((N .EQ. NN) .AND. (ITHSP .EQ. 2)) THEN
    ICALLV = 1
clag end change
```

-corrected calls to FAAFragCtl routine

```
C          CALL AREASV
CLAG NEW CALL FOR FAA FRAGS

clag          CALL FAAFragCtl ( FrID, TgtMat, LOS, WPs, VPs, AreaP
CALL FAAFragCtl ( FrID, TgtMat, LOS, WPs, VPs, AreaP,
&                WR, V50, Vr)

CLAG END CHANGE
:
clag removed RHOT
clag          CALL FAAFragCtl (FrID, TgtMat, RHot, LOS, WPs, VPs, AreaP,
CALL FAAFragCtl (FrID, TgtMat, LOS, WPs, VPs, AreaP,
&                WR, V50, Vr)
CLAG END CHANGE
```

File: Covart6\init_EACA_stub.f

-added alternate code for setting new material string descriptors if changes to init.f are not acceptable; requires access to MATHER common block string descriptors HER, DER for new materials in positions 46, 47.

```
cflag for ueddams if changes to init.f are not accepted
c      COMMON /MATHER/ HER(MXMTCD), DER(MXMTCD)
cflag not sure I need to redeclare type here?
c      CHARACTER * 8 HER, DER
cflag end change
:
cflag if material names init here v. in init.f
c      CHARACTER * 8 THDRUD
c      DIMENSION THDRUD(4)
c      DATA THDRUD / 'COMPOSIT' , 'E      ' , 'INCONEL@' , ' 625 LCF' /
cflag end change

cflag for ueddams if changes to init.f are not accepted
c      HER(46) = THDRUD(1)
c      DER(46) = THDRUD(2)
c      HER(47) = THDRUD(3)
c      DER(47) = THDRUD(4)
cflag end change
```

-utilized recommended common block naming convention for UEDDAM-specific parameters in separate common blocks

-restored material constant initialization values from earlier "approved for release" version (using a modified COVART 4.2.3 build) for use in UEDDAM compiled versions of COVART 6.x.

{see routine init_EACA_stub.f}

File: Covart6\initlib_u.f

-added access to new common block MATAL1UD for access to DYNshr variable

```
cflag added for UEDDAM
c      COMMON /MATAL1UD/ DYNshr(MXMT)
cflag end change
```

-corrected call to FAAFrInit to include DYNshr variable

```
cflag added DYNshr
cflag      CALL FAAFrInit ( BHN, RHO, SIGY, SIGU,
cflag      &              TAUS, E, BULK, DENAL, RA, U, CF,
cflag      &              CALL FAAFrInit ( BHN, RHO, SIGY, SIGU, TAUS,
cflag      &              DYNshr, E, BULK, DENAL, RA, U, CF,
```

File: Covart6\init.f

-added string descriptors to THER, TDER for new materials in positions 46, 47

```
crag for ueddam
crag + 'TITANIUM', 5*'          ', 'WATER  ',
      + 'TITANIUM', 'COMPOSIT', 'INCONEL@', 3*'          ', 'WATER  ',
crag end change
:
crag for ueddam
crag + 5*'          ',
      + 'E          ', ' 625 LCF', 3*'          ',
crag end change
```

-may need to revisit variables NPCT, NPKCT settings if used elsewhere for UEDDAM

```
crag for ueddam, vals were NPCT 25000, NPKCT 6000
crag but if you're no longer using these params because you're
crag init'ing to values from parm.inc file instead, then why
crag keep them here? If they're used elsewhere, then I need them
crag reset per above for UEDDAM.
crag
crag      DATA NPCT / 10000/
crag      DATA NPKCT / 2999/
DATA NPCT / 25000/
DATA NPKCT / 6000/
crag
```

-may need to add conditional for using alternate init_EACA routine

```
crag Either compile in alternate file (current) for this routine when
crag making a UEDDAM build, or will need to add conditional here later
crag
CALL init_EACA
```

File: Covart6\inputd.f

-added check for valid ITHSP=2 UEDDAM legacy mode processing

```
crag need to check for ITHSP=2 valid UEDDAM legacy mode processing
c      IF ( ITHSP .EQ. 0 ) THEN
      IF ( ITHSP .EQ. 0 .OR. ITHSP .EQ. 2 ) THEN
crag end change
```

File: Covart6\inputi.f

-added access to new common blocks END2, HEPEN, JTYP2H, MATUSEUD for access to ITHSP, IHPROJ, JTYPH, and IFAAF variables

```
clag for UEDDAM
      COMMON /END2 / ITHSP, IUNGRD
clag
:
clag need this for new MATCODE checks and UEDDAM
      COMMON /HEPEN / IHPROJ
clag
:
cdrh for UEDDAM
      COMMON /JTYP2H/ JTYPH(MKJTYP)
cdrh
:
clag for UEDDAM
      COMMON /MATUSEUD/ IFAAF(MXMTCD)
clag
```

-added dimension statement for ICHO variable

```
cdrh for UEDDAM
      DIMENSION ICHO(8)
cdrh
```

-added parens for clarity and check versus IFAAF array for UEDDAM materials

```
clag added some parens in this IF block for clarity
      ELSE IF ( (IPROJ .GE. 2) .AND. (IPROJ .LE. 4) ) THEN
      MCHECK = IPROJ(JM)
      MCHECK = MCHECK/100
      IF (IHPROJ .EQ. 0) THEN
c      JCHECK = JTCGF(JM)
clag need to check against IFAAF array for UEDDAM materials
      IF (ITHSP .NE. 2) THEN
      JCHECK = JTCGF(JM)
      ELSE
      JCHECK = IFAAF(JM)
      END IF
clag end change
```

File: Covart6\parm.inc

-increased baseline values for MXPCTB (per NPKCT) and MXPCLC (per NPCT)

```
! MXPCTB - Maximum Number of PK Tables
!
!lag for ueddams
!lag PARAMETER (MXPCTB = 2999)
PARAMETER (MXPCTB = 6000)
!lag end change
:
! MXPCLC - Maximum Number of locations for PK table information in the VEL and
PRK
! Arrays. These arrays contain the the number of steps for each
curve
! and the data for each step for each PK table. The average number
of
! curves for all the PK table types is 11 curves. If an assumption
of an
! average number of steps for each curve is 5, and the average
number
! of curves for a table is 11, then the size of these arrays need to
be
!  $5 * 11 * 2999 = 164,945$ . Currently 10,000 seems to be doing well
since
! most of the components refer to common PK tables.
!
! ADJUSTED for JSF
!
! PARAMETER (MXPCLC = 10000)
!lag for ueddams
!lag PARAMETER (MXPCLC = 20000)
PARAMETER (MXPCLC = 25000)
!lag end change
```

File: Covart6\read_threat_u.f90

-added check for valid ITHSP=2 UEDDAM legacy mode processing

```
      CASE(cThrParm)
      :
      ! lg need to recognize ITHSP=2 as valid for legacy mode processing
      !   IF ( ITHSP .EQ. 0 ) THEN
      !     IF ( ITHSP .EQ. 0 .OR. ITHSP .EQ. 2 ) THEN
      :
      CASE(cHeFrag)
      :
      ! lg need to recognize ITHSP=2 as valid for legacy mode processing
      !   IF ( ITHSP .EQ. 0 ) THEN
      !     IF ( ITHSP .EQ. 0 .OR. ITHSP .EQ. 2 ) THEN
      :
      !   POST-READ OPERATIONS FOR HE THREATS
      :
      ! lg need to recognize ITHSP=2 as valid for legacy mode processing
      !   IF ( ITHSP .EQ. 0 ) THEN
      !     IF ( ITHSP .EQ. 0 .OR. ITHSP .EQ. 2 ) THEN
      !       IF ( .NOT. HEFRGFlg ) THEN
      :
      ! lg need to recognize ITHSP=2 as valid for legacy mode processing
      !   IF ( ITHSP .EQ. 0 ) THEN
      !     IF ( ITHSP .EQ. 0 .OR. ITHSP .EQ. 2 ) THEN
      !       DO JJ = 1, NOS
      :
      ! lg need to recognize ITHSP=2 as valid for legacy mode processing
      !   IF ( ITHSP .EQ. 0 ) THEN
      !     IF ( ITHSP .EQ. 0 .OR. ITHSP .EQ. 2 ) THEN
      !       DO J = 2, NOS
      :
      ! lg need to recognize ITHSP=2 as valid for legacy mode processing
      !   IF ( ITHSP .EQ. 0 ) THEN
      !     IF ( ITHSP .EQ. 0 .OR. ITHSP .EQ. 2 ) THEN
      :
      !   CONVERT ETA TO NUMBER OF FRAGMENTS IF NECESSARY
```

File: Covart6\runshot4.f90

-changed bounds for allocating BurstPoints array

```
!LAG      allocate (BurstPoints(INSHT), STAT=iStatus)
          allocate (BurstPoints(GroupSize), STAT=iStatus)
```

-eliminated extra WriteEndOfViewMessages

```
!LAG per RodS 17Jul09
!LAG      call WriteEndOfViewMessages
          if (allocated(Shotline_Data%LOS)) deallocate(Shotline_Data%LOS)
```


-changed locations for heading writes & shotline data flush

```
!7/23/09 DRH: modified as per Rod Stewart's instructions
      call WriteShotlineViewHeaders(I)
      if (Hits(I) .NE. 0) then
:
:
!      Store the shotline data
!
      Flush = 0
      call Output_Shotline (Shotline_Data, Flush)
!
!      Output the shotline data
!
!7/20/09 DRH: modified as per Rod Stewart's instructions
      Flush = 0
      call Output_Shotline (Shotline_Data, Flush)

      if (ALLOCATED(Shotline_Data%LOS)) deallocate(Shotline_Data%LOS)

      end if

!7/23/09 DRH: modified as per Rod Stewart's instructions
      Flush = 1
      call Output_Shotline (Shotline_Data, Flush)

      END DO
```

-moved loop end past cleanup calls

```
!LAG Moved loop end past cleanup calls
      END DO
!
! Retain the shotline data storage memory if needed for next view
:
!LAG relocated loop end here
      END DO

      END
```

File: Covart6\sol.f90

-added initialization for IGRIDS

```
!LAG added IGRIDS
      IGRIDS = 0
```

-added MAX function

```
!LAG added per RodS 16Jul09
      IGRIDS = MAX(IGRIDS, INSHT4)
```

File: Covart6\tables.f

-split CKILL into single- and double-character CKILL1, CKILL2 to eliminate blanks in concatenated filenames

```
clag To eliminate 'blanks' in concatenated file names, need two types of
clag character reps for kill numbers
clag CKILL1 - single char for kill numbers 1-9
clag CKILL2 - two chars for kill numbers 10-16
clag CHARACTER * 2 CKILL
CHARACTER * 1 CKILL1
CHARACTER * 2 CKILL2
clag end change
:
clag Updated to eliminate 'blank' chars in filenames
c WRITE ( CKILL, 6 ) NKILL(KLV)
c JSFILE(KLV) = 'jsem.'//CKILL//'.'//CAZI//'.'//CELE
IF ( NKILL(KLV) .LT. 10 ) THEN
WRITE ( CKILL1, 5 ) NKILL(KLV)
JSFILE(KLV) = 'jsem.'//CKILL1//'.'//CAZI//'.'//CELE
ELSE
WRITE ( CKILL2, 6 ) NKILL(KLV)
JSFILE(KLV) = 'jsem.'//CKILL2//'.'//CAZI//'.'//CELE
END IF
clag end change
:
clag Updated to eliminate 'blank' chars in filenames
c WRITE ( CKILL, 6 ) NKILL(KLV)
C *** FIXED FILE OPENING BUG FOR SHAZAM FILES --- RKS 1/25/08
C CKSZFILE(KLV) = 'shazam.'//CKILL//'.'//CAZI//'.'//CELE
c SZFILE(KLV) = 'shazam.'//CKILL//'.'//CAZI//'.'//CELE
IF ( NKILL(KLV) .LT. 10 ) THEN
WRITE ( CKILL1, 5 ) NKILL(KLV)
SZFILE(KLV) = 'shazam.'//CKILL1//'.'//CAZI//'.'//CELE
ELSE
WRITE ( CKILL2, 6 ) NKILL(KLV)
SZFILE(KLV) = 'shazam.'//CKILL2//'.'//CAZI//'.'//CELE
END IF
clag end change
```

File: Covart6\zondat.f

-added check for valid ITHSP=2 UEDDAM legacy mode processing

```
cclag need to check for ITHSP=2 valid UEDDAM legacy mode processing
c      IF ( ITHSP .EQ. 0 ) THEN
          IF ( ITHSP .EQ. 0 .OR. ITHSP .EQ. 2 ) THEN
              CALL FRAGCK( LSHAPE(1), MATFHE )
          :
cclag need to check for ITHSP=2 valid UEDDAM legacy mode processing
c      IF ( ITHSP .EQ. 0 ) THEN
          IF ( ITHSP .EQ. 0 .OR. ITHSP .EQ. 2 ) THEN
              IF ( .NOT. HFRAG ) THEN
          :
cclag need to check for ITHSP=2 valid UEDDAM legacy mode processing
c      IF ( ITHSP .EQ. 0 ) THEN
          IF ( ITHSP .EQ. 0 .OR. ITHSP .EQ. 2 ) THEN
              DO 5025 JJ = 1, NOS
          :
cclag need to check for ITHSP=2 valid UEDDAM legacy mode processing
c      IF ( ITHSP .EQ. 0 ) THEN
          IF ( ITHSP .EQ. 0 .OR. ITHSP .EQ. 2 ) THEN
              DO 5027 J = 2, NOS
          :
cclag need to check for ITHSP=2 valid UEDDAM legacy mode processing
c      IF ( ITHSP .EQ. 0 ) THEN
          IF ( ITHSP .EQ. 0 .OR. ITHSP .EQ. 2 ) THEN
C
C      W W W W W W W W W W W W W W W W W W W W W W W W W W W W W
C
C      CONVERT ETA TO NUMBER OF FRAGMENTS IF NECESSARY
C
```

File: libraries\FAAPEN\faafrag.f

-separated DYNshr into labeled common MTL1UDFAA

```
cclag separated DYNshr
COMMON /MATL1FAA/ BHN(MXMT), RHO(MXMT), SIGY(MXMT),
&
SIGU(MXMT), TAUS(MXMT)
c      &
COMMON /MTL1UDFAA/ DYNshr(MXMT)
cclag end change
```

-changed case match for consistency in Vr references

```
cclag      VR = 0.0
          Vr = 0.0

cclag      VR = SQRT (VV ** 2 - V50 ** 2) / (1. + Q4)
          Vr = SQRT (VV ** 2 - V50 ** 2) / (1. + Q4)
```

File: libraries\FAAPEN\faafragctl.f

-restored MATHE1FAA common block

```
clag restored MATHE1FAA w/JRES value
COMMON /MATHE1FAA/ JRES
```

-separated DYNshr into labeled common MTL1UDFAA

```
CLAG ADDED DYNshr FOR FAA PENETRATION EQNS
COMMON /MATL1FAA/ BHN (MXMT) , RHO (MXMT) , SIGY (MXMT) , SIGU (MXMT) ,
&
    TAUS (MXMT)
clag separated DYNshr
c
&
    TAUS (MXMT) , DYNshr (MXMT)
COMMON /MTL1UDFAA/ DYNshr (MXMT)
CLAG END CHANGE
```

-removed variable declarations for LL, RIFrag

```
clag removed LL
clag INTEGER KKPEN, LL
INTEGER KKPEN

clag removed RIFrag
clag REAL TlEq, WT, RIFrag, V, AreaP, WR, V50, Vr
REAL TlEq, WT, V, AreaP, WR, V50, Vr
```

-corrected FAAfrag call by removing RIFrag parameter

```
clag removed RIFrag parameter from call
clag CALL FAAfrag ( FrID, TgtMat, RHOt, TlEq, WT, RIFrag, V, AreaP,
CALL FAAfrag ( FrID, TgtMat, RHOt, TlEq, WT, V, AreaP,
```

-changed case match for consistency in Vr references

```
clag VR = 0.0
Vr = 0.0

clag VR = V * EXP (-EXPOW)
Vr = V * EXP (-EXPOW)
```

File: libraries\FAAPEN\faafrinit.f

-separated DYNshr into labeled common MTL1UDFAA

```
CLAG ADDED DYNshr FOR FAA PENETRATION EQNS
COMMON /MATL1FAA/ BHN (MXMT) , RHO (MXMT) , SIGY (MXMT) , SIGU (MXMT) ,
&
    TAUS (MXMT)
clag separated DYNshr
clag &
    TAUS (MXMT) , DYNshr (MXMT)
COMMON /MTL1UDFAA/ DYNshr (MXMT)
CLAG END CHANGE
```

Updates Required for Pre-Existing UEDDAM Input Datasets

MASTER File - ****New File****

- Required to set run mode to legacy COVART5 operation and provide name of BASIC control file

MODE	COVART5	BASIC
ENDDATA		

BASIC File

- Add 3 parameters to TABFLAG record

\$COMMENT	IOPKH	IOJSEM	NVAAZI	NVAELE	TGTVA	IOSHZM	IBIGLO	IRBURST
TABFLAG	0	1	1	1	0	0	0	0

THREAT File

- Add 1 parameter to THRPARM record, THSPEC=2 indicates UEDDAM run

\$COMMENT	IPROJ	NVEL	NWT	YAW	AMU1	AMU2	THSPEC
THRPARM	2	1	1				2

- Add 2-3 parameters to HECHAR record (if DSTMAX already present, insert WARHDX and WARHDL parameters preceding DSTMAX; otherwise, add all 3 parameters)

\$COMMENT	NOS	IX	IHALF	CHARGN	FUZTIM	FUZDIS	WARHDX	WARHDL	DSTMAX
HECHAR	2	1	0	200000.	0.0	6.0	0.	0.	0

JTYPE File

- All COMPPHY records must occur before any COMALIAS records
- Change material code index (JM) values of '37' to '46' for Composite
- Change material code index (JM) values of '38' to '47' for Inconel® 625 LCF

PKH File

- Add 1 parameter to FIREDATA record (EXCNT=1 default setting for number of exit fires that can occur from a single flash/function)

\$COMMENT	FIRE	KTANK	EXCNT
FIREDATA	0.5	2	1

A-4. Correspondence: Summary of COVART6 Changes for UEDDAM Based on COVART 5 Modifications - 12 January 2009

Summary of changes made to COVART6 to accommodate the FAA Penetration Library (for UEDDAM) based on changes made by SURVICE Engineering to COVART5 as documented in e-mail dated 10/03/2008.

Change 1:

All material data constants have been removed from the **init** subroutine contained in the *init.f* file in COVART6 main. These material data constants have been placed in the **init_EACA** subroutine in the *init_EACA.f* file. The **init** subroutine calls **init_EACA**.

A separate file *init_EACA_stub.f* contains a stubbed copy of **init_EACA** for use in the UEDDAM version of COVART6. All material values in the stubbed version of **init-EACA** are initialized to zero. Makefiles have been updated with two separate build paths: one for UEDDAM and one for a regular COVART6 release. The makefile path for the UEDDAM version of COVART6 uses the stubbed version of **init_EACA**.

Any material constants required by the UEDDAM version of COVART6 will have to be added back into the stubbed version of **init_EACA**. For example, the DYNshr constants added to the **init** subroutine by SURVICE Engineering for UEDDAM could be added and initialized in the stubbed version of the **init_EACA** subroutine.

Change 2:

The MXHOUT parameter for UEDDAM output has been added to the parm.inc file (MXHOUT = 9999) and this parameter file has been updated for all the COVART libraries.

Change 3:

The FAA penetration library subroutines routines **faafragctl**, **faafrag**, **faafrinit** and **faaheinit** have been used to create the FAA Penetration library [a shared-object (.so) file for Linux systems and a dynamically loadable library (.dll) file for Windows]. The subroutines were included as provided by SURVICE Engineering with only one significant change: a Windows DLLEXPORT command was placed at the beginning of the **faaheinit**, **faafrinit**, and **faafragctl** subroutines to flag these routines as library entry points for the Windows build process. This command is preceded by an exclamation mark (!) so that it appears to be a FORTRAN 90 comment to the Linux build process.

Added an additional call to **faafragctl** in **faasl** after the unpenetrated combined plates have been uncombined for individual plate penetration. The SURVICE Engineering version of **faafragctl** only contained one call to **faasl**.

faaheinit is called by the COVART6 **herayctl** subroutine and **faafrinit** is called by the COVART6 **initlib** subroutine when the variable ITHSP is set to 2.

An additional subroutine, **FAAGetVersion** was added to the penetration library to return a FORTRAN Character*80 argument, FAAVER, that identifies the library version number. The FAAVER variable was added to MODVER, a labeled COMMON block used in the **covart5**, **inputc**, **read_master**, and **getver** subroutines.

Change 4:

Added **faasl** subroutine to COVART6 makefiles so it can be called by **herayctl** instead of the **fragsl** subroutine.

Change 5:

Changed COVART6 **read_threat** subroutine to add the ITHSP = 2 condition on the THRPRM record and modified if-then-else tests to include ITHSP=2 in the range of possibilities to indicate a UEDDAM run.

Change 6:

Added named COMMON block END2 containing ITHSP to the **herayctl** subroutine to make ITHSP available for if-then-else test.

Created named COMMON block EXTBDPUD containing APFRAG and RLFRAG variables rather than expanding named COMMON block EXTBDP with those variables. This method is usually preferred since the COMMON block EXTBDP is used in several other places in COVART6. Included the new COMMON block, EXTBDPUD, in the **herayctl** and **faasl** subroutines.

Created named COMMON block UEDDAMIO containing the IOHOLE variable for the **faasl** subroutine rather than placing IOHOLE into the INOUT7 COMMON that is used in several different places in COVART6.

Change 7:

Since initial projectile yaw calculations are ITAR restricted, this calculation that was performed in several places in COVART5 has been moved into the **get_projyaw** subroutine in the **get_projyaw.f** file. An ITAR compliant stub that returns a value of 0.0 for PHI (yaw angle) was written and can be found in the file **get_projyaw_stub.f**. Makefiles were updated to use the stubbed version when a UEDDAM version of COVART6 is being compiled.

Change 8:

All FRAGPEN, PROJPEN, and FATEPEN library routines have been stubbed. The stubbed versions are contained in files with subroutine names suffixed with the word stub.

The following FRAGPEN library routines have been stubbed:

frag-ctl, fragheinit, fragversion, and jtcgfrinit.

The following PROJPEN library routines have been stubbed:

COVART_DATA, COVART_PROJ, jtcgapiinit, proj-ctl, and projversion.

The following FATEPEN3 library routines have been stubbed:

cloudhdlr, fatehdlrvers, fatephdlr, heinithdlr, inihdlr, and pkshathdlr.

General comment about labeled COMMON:

COVART6 developer's have adopted the practice of not using the same names for COMMON blocks in the libraries and in the COVART6 main program. For example if the variables A, B, C, and D in the labeled COMMON block ABCD in COVART6 main are required for the PROJPEN library, they will be passed to the library through the interface to local variables AP, BP, CP, and DP then stored in a common block labeled ABCDP with the global variables (to the library) A, B, C, and D.

If these variables are required in the JTJG FRAGPEN library they will be passed through the library interface to local variables AJ, BJ, CJ, and DJ then stored in a COMMON block labeled ABCDJ and with global variables (to the library) A, B, C, and D.

Comment about SURVICE change #3 (packing APFRAG and WFRAG into EXOB and doubling precision of EXOB in the ray-header record):

We would recommend not changing the binary file format of the OUNPKA file by increasing the precision of EXOB in the ray-header as this makes the binary file incompatible with current pending and future COVART releases. Rather we would recommend packing one of the fragment variables in the single-precision EXOB variable and packing the other into the top half of another variable in the ray-header. One possibility is to pack the second fragment variable into the top half of the variable for number of intersections on a ray.

A-5. Correspondence: UEDDAM Appendix A with Annotated COVART 5 Modifications - 3 October 2008

Background

A stripped down and modified version of COVART 4.2 was created for use with UEDDAM 3.0, as noted:

Significant modifications were made to COVART 4.2 in order to permit implementation into UEDDAM. The most significant modification was performed by KETRON and consisted of removing all export-controlled features of COVART 4.2 to permit free distribution of UEDDAM throughout the world. Generally, all armor-piercing incendiary (API) threat functioning and penetration algorithms as well as most of the high explosive incendiary (HEI) modules were removed.

Following the KETRON modifications, other smaller changes were made to the code by SURVICE Engineering to permit accurate data transfer between UEDDAM and COVART and to implement the new penetration equations developed by NAWC-WD Code 418000D.

The most significant change made by SURVICE Engineering was to implement the new penetration equations. The subroutine "FRAG" was rewritten based on the China Lake equations.

It is now desirable to integrate the UEDDAM modifications to COVART back into the current, configuration-controlled COVART 5 (and developmental COVART 6) models. However, two factors make reintegration of the UEDDAM modifications more difficult than simply "plugging in the updates." First, COVART 5 was a major upgrade that included restructured, modularized code and separate penetration library modules, significant changes to the code layout. Elements changed for UEDDAM are not necessarily located within the same files, or are echoed redundantly among separated initialization routines within the penetration libraries. Second, since UEDDAM was using a separate, standalone version of COVART 4.2, testing was limited to the HE portions used by UEDDAM, without concern for any potential introduction of errors to unused fragment and projectile routines. Care and adequate testing must now be performed to make sure that the UEDDAM modifications operate as intended without breaking or modifying the other operating modes in COVART 5.

This document reiterates the change information provided in Appendix A of the UEDDAM 3.0 User Manual to categorize required changes, and provides suggested changes to COVART 5 to implement the desired capabilities for UEDDAM use. When applicable, additional discussion is included to address any caveats or concerns with the specific implementation recommended. A modified version of COVART 5 was developed to test the suggested updates; testing is still ongoing.

Development and testing were performed on a 1.5GHz Dell Inspiron 8600 Microsoft Windows XP Professional platform using Microsoft Visual Studio 2005 Professional Edition with the integrated Intel Fortran compiler. A set of makefiles were created for production and debug builds of both the existing and updated COVART 5 versions.

General Discussion

{Expand on following: About built as separate library, dupe of jtcgfrag library; list of routines requiring changes; concern w/releasability of a hobbled version if source not separated correctly; correlation with other changes since this version5 (more materials); might be better implementation to use IHPROJ toggle, add another LOSTAP=20?? type for UEDDAM-formatted hotline info v. reusing type 14, and making a unique labeled common to use for this versus packing into existing hotline structure with subsequent change in EXOB precision; impact of plate combining, and also if all materials adequately defined for use across models (diffs w/generalized composite properties, other props needed for Inconel), numbered v. embedded format statements; scope mirror of labeled common blocks; etc.}

Change 1: Modifications to the (Former) FRAG Routine

In COVART 5, the FRAG routine (and related AREASN routine) was replicated across the separate penetration libraries. In keeping with the JTTCG fragment penetration library layout, a similar UEDDAM/FAA fragment penetration library was established. A new directory, 'ueddam-frag', with subdirectory 'FAAPEN' was created in the 'source code' directory. The following source files (and related makefiles) were created/modified based on those in the JTTCG fragment penetration library (in an attempt to ensure that the same initializations/processes remained intact):

main\COVART5\faasl.f: Shotline processing routine, calls FAAFragCtl()

ueddam-frag\FAAPEN\faafninit.f: Initialization routine for libraries

ueddam-frag\FAAPEN\faaheinit.f: Initialization routine for HE generated fragments

ueddam-frag\FAAPEN\faafragctl.f: Fragment penetration routine, calls FAAfrag()

ueddam-frag\FAAPEN\faafrag.f: Reworked FRAG routine with FAA V50

Source code for these routines is included at the end of this document.

Control for this type of fragment when used by UEDDAM is handled via the herayctl.f file:

```
CLAG ADD /HEPEN / FOR IHPROJ TOGGLE
COMMON /HEPEN / IHPROJ
CLAG END CHANGE
:
C
C Call the proper fragment shotline handling routine
C
CLAG SHOULD TRUE TOGGLE BE IHPROJ? (0=JTTCG, -2|-3|-4=FATE, 1=UEDDAM FAA?)
CLAG IPROJ HERE, BUT INITIALLY IPROJ=2 AND GETS CHANGED TO 0 AFTER BURST
C IF (IPROJ .EQ. 0) THEN
C CALL FRAGHEInit (DIA, LSHAPE, FL, LF, DIA2, PROL, DENRAT)
C CALL FRAGSL (POWER)
CLAG
IF ( IHPROJ .EQ. 0 ) THEN
CALL FRAGHEInit ( DIA, LSHAPE, FL, LF, DIA2, PROL, DENRAT )
CALL FRAGSL ( POWER )
ELSE IF ( IHPROJ .EQ. 1 ) THEN
CALL FAAHEInit ( DIA, LSHAPE, FL, LF, DIA2, PROL, DENRAT )
CALL FAASL ( POWER )
CLAG END CHANGE
ELSE
CALL FATEHEInit ( DIA, LSHAPE, FL, LF, DIA2, RHOUSE, PROL,
& DENRAT, IALY )
CALL FATESL ( POWER )
END IF
```

Note that, since the UEDDAM/FAA fragment penetration is only used with HE type threats, no faactl.f control file for this type of fragment *as an independent threat type* is necessary.

Change 2: Labeled Common /EXTBPD/

The common block "EXTBPD" was modified in all relevant subroutines to include the presented area of the fragment to be used in the new penetration algorithm. Figure A-1 shows the actual change as coded in COVART.

```
C JKM 9/14/99 UEDDAM
C      COMMON /EXTBPD/ IEXTBP, WFRAG, VFRAG, MTFRAG           04582000   980915
      COMMON /EXTBPD/ IEXTBP, WFRAG, VFRAG, MTFRAG, APFRAG, RLFRAG 04582000   980915
C JKM 9/14/99 UEDDAM
```

Figure A-1. Changes to EXTBPD

The new variables "MTFRAG", "APFRAG", and "RLFRAG" represent the fragment material, presented area, and presented area perimeter length, respectively. The presented area and the presented area perimeter of the fragment are calculated in UEDDAM and passed to COVART through the shotline file.

In COVART 5, this block already contained an additional variable, RLDFRG. Variable APFRAG, fragment presented area, and variable RLFRAG, fragment presented area perimeter length, were added as below:

```
CLAG  ADDED FRAG PRESENTED AREA PERIMETER LENGTH AND
CLAG  PRESENTED AREA TO EXTBPD
C      COMMON /EXTBPD/ IEXTBP, WFRAG, VFRAG, MTFRAG, RLDFRG
      COMMON /EXTBPD/ IEXTBP, WFRAG, VFRAG, MTFRAG, RLDFRG,
      & APFRAG, RLFRAG
CLAG  END CHANGE
```

Labeled common /EXTBPD/ was updated in the following existing files:

main\COVART5: areach.f, areas.f, covart.f, fatefgctl.f, fragctl.f, heivam.f, heivsl.f, herayctl.f, init.f

Labeled common /EXTBPD/ was also used in the new UEDDAM modules:

ueddam-frag\FAAPEN: faafrag.f

Labeled common /EXTBPD/ had a commented-out reference that was NOT updated in the following existing files:

main\COVART5: initlib.f, pkasgn.f

Change 3: Labeled Common /SLDAT2/

The value of fragment presented area is packed in to the variable "EXOB" that is used in the ray header record using the formula:

$$EXOB = WFRAG + \frac{APFRAG}{1000}.$$

In order to give EXOB the precision necessary to store the information, it has been changed to a double-precision variable in COVART. Figure A-2 shows the implementation as coded in COVART.

```
COMMON /SLDAT2/ EXOB (170), ENOB (170), MTCODE (170)          05625000    960315
C JKM UEDDAM
      DOUBLE PRECISION EXOB
C JKM UEDDAM
```

Figure A-2. Changes to EXOB

The value of fragment presented area perimeter is stored in the shotline file in the variable "CTH" that is used in the ray header record. "RLFRAG" is set equal to CTH in subroutine AREACH. {See section A-5, [Change 4: Subroutine AREACH.](#)}

In COVART 5, the array dimension had been changed to a parameter reference, NLSFRC = 170, from the parm.inc file. Variable EXOB was changed from a single precision (REAL*4) to a DOUBLE PRECISION (REAL*8) variable:

```
COMMON /SLDAT2/ EXOB (NLSFRC), ENOB (NLSFRC), MTCODE (NLSFRC)
CLAG EXOB TYPE CHANGED FROM SINGLE (R*4) TO DOUBLE (R*8) PRECISION TO
CLAG STORE PACKED PRESENTED AREA DATA FOR UEDDAM APPLICATION
      DOUBLE PRECISION EXOB
CLAG END CHANGE
```

Variable EXOB in labeled common /SLDAT2/ was updated in the following existing files:

```
main\COVART5:  areach.f, areasc.f, areasd.f, arease.f, burstb.f, burstc.f,
               burstd.f, bursts.f, covart.f, fasgn2.f, fastgn.f, gift.f,
               heivsl.f, pgen.f, shotgn.f, skipbp.f, slchk.f, wcout.f,
               wcout2.f, weob.f, weov.f, weov2.f, wray.f, wshotl.f
```

Variable EXOB in labeled common /SLDAT2/ was used in the new UEDDAM modules:

```
ueddam-frag\FAAPEN:  NONE
```

Change 4: Subroutine MVCHK

The UEDDAM code will always create an MV file, even if there are no redundancies. This resulted in a problem with the MVCHK subroutine in COVART when it looks in arrays related to the MV input. Since there are no MV groups in the MV file if there are no redundancies, the MVCHK routine would attempt to access the 0th element on the array JENDS. To fix this problem, the change shown in Figure A-3 was made to subroutine MVCHK.

```

C JKM For UEDDAM
C      J = ABS( JENDS( NGROUP ) )           30246000   960315
C      DO 8000 I = 1, J                     30247000   960315
C          IF ( MUL(I) .NE. 0 ) THEN        30248000   960315
C              L = L + 1                    30249000   960315
C              MUL(L) = MUL(I)              30250000   960315
C              IF ( JENDS(K) .EQ. I ) THEN  30251000   960315
C                  JENDS(K) = L            30252000   960315
C                  K = K + 1                30253000   960315
C                  DO 8010 KK = K, NGROUP   30254000   960315
C                      IF ( JENDS(K) .LT. 0 ) THEN 30255000   960315
C                          JENDS(K) = -L      30256000   960315
C                          K = K + 1          30257000   960315
C                      ELSE                  30258000   960315
C                          GO TO 8020         30259000   960315
C                      END IF                30260000   960315
C 8010 CONTINUE                            30261000   960315
C 8020 CONTINUE                            30262000   960315
C          END IF                           30263000   960315
C      END IF                               30264000   960315
C 8000 CONTINUE                            30265000   960315
C
      if(ngroup.gt.0) then
      J = ABS( JENDS( NGROUP ) )           30246000   960315
      DO 8000 I = 1, J                     30247000   960315
          IF ( MUL(I) .NE. 0 ) THEN        30248000   960315
              L = L + 1                    30249000   960315
              MUL(L) = MUL(I)              30250000   960315
              IF ( JENDS(K) .EQ. I ) THEN  30251000   960315
                  JENDS(K) = L            30252000   960315
                  K = K + 1                30253000   960315
                  DO 8010 KK = K, NGROUP   30254000   960315
                      IF ( JENDS(K) .LT. 0 ) THEN 30255000   960315
                          JENDS(K) = -L      30256000   960315
                          K = K + 1          30257000   960315
                      ELSE                  30258000   960315
                          GO TO 8020         30259000   960315
                      END IF                30260000   960315
 8010 CONTINUE                            30261000   960315
 8020 CONTINUE                            30262000   960315
          END IF                           30263000   960315
      END IF                               30264000   960315
 8000 CONTINUE                            30265000   960315
      else
      J = 0
      end if
C JKM For UEDDAM

```

Figure A-3. Changes to Subroutine MVCHK

In COVART 5, the source code was identical, so the same wrapping IF-ELSE-ENDIF statement was inserted:

```
CLAG  UEDDAM FIX FOR USE WHEN NGROUP=0
      IF ( NGROUP.GT.0 ) THEN
CLAG
      J = ABS( JENDS( NGROUP ) )
      DO 8000 I = 1, J
          IF ( MUL(I) .NE. 0 ) THEN
              L = L + 1
              MUL(L) = MUL(I)
              IF ( JENDS(K) .EQ. I ) THEN
                  JENDS(K) = L
                  K = K + 1
                  DO 8010 KK = K, NGROUP
                      IF ( JENDS(KK) .LT. 0 ) THEN
                          JENDS(KK) = -L
                          K = K + 1
                      ELSE
                          GO TO 8020
                      END IF
                  CONTINUE
              CONTINUE
          END IF
      END IF
      8010 CONTINUE
      8020 CONTINUE
      8000 CONTINUE
CLAG  ELSE
      J = 0
      END IF
CLAG  END UEDDAM FIX
```

Change 5: Subroutine AREACH

Disable velocity reduction from release to first component due to air drag

The COVART code also degrades the fragment velocity from the release point to the first component hit along the ray. Apparently this is done to account for air drag. While this is desirable, the COVART code does not account for the forward velocity of the aircraft in computing fragment velocity. Also, the rays generated from the burst point through the aircraft assume straight-line trajectories. If the air drag is to be considered, then it should be done completely. That is, consideration for the forward velocity of the aircraft and the effects of the wind on the trajectory of the fragments should be included. Since this would be beyond the scope of this phase of the UEDDAM development, the air drag effects were eliminated in COVART. This was done by simply setting the distance from the release point to the first component hit to zero in subroutine AREACH. Figure A-4 shows the implementation.

```
C JKM UEDDAM
C
C This mod makes the airgap from the release point to the first
C component on the ray zero inches. This prevents COVART
C from reducing the velocity between the release point to the first
C component on the shotline.
C
C DIRX = CTH(LL) 37420000 960315
C IF (ITHRT .EQ. 3 .AND. IEXTBP .EQ. 1) THEN
C     DIRX = 0.00 37420000 960315
C     RLFRAG = CTH(LL)
C ELSE
C     DIRX = CTH(LL)
C END IF
C JKM UEDDAM
```

Figure A-4. Change to DIRX {in AREACH}

In COVART 5, the source code was identical. However, the conditional used to evoke the change for UEDDAM, (ITHRT .EQ. 3 .AND. IEXTBP .EQ. 1), is not sufficiently limiting when operating outside UEDDAM. Since UEDDAM utilizes the HE threat type with the new fragment penetration model, the limiting condition was changed to use the IHPROJ variable in labeled common /HEPEN /. COVART 5 currently indicates the type of fragment product threats using values of IHPROJ = 0 for JTCG fragments, and values of IHPROJ = {-2 | -3 |-4 } for FATEPEN fragments. A value of IHPROJ = 1 was used to indicate UEDDAM/FAA fragments.


```

CLAG THIS MOD MAKES THE AIRGAP FROM THE RELEASE POINT TO THE FIRST
CLAG COMPONENT ON THE RAY ZERO INCHES. THIS PREVENTS COVART FROM
CLAG REDUCING THE VELOCITY BETWEEN THE RELEASE POINT TO THE FIRST
CLAG COMPONENT ON THE SHOTLINE. ALSO SETS THE FRAGMENT PRESENTED
CLAG AREA PERIMETER VALUE THAT WAS PACKED INTO CTH BY UEDDAM
CLAG *** COULD ACHIEVE DIRX=0 BY ADDING DRAGMODL INPUT TO BASIC
CLAG *** 'DRAGMODL NONE'
C DIRX = CTH(LL)
  IF ( IHPROJ .EQ. 1 ) THEN
    DIRX = 0.0
    RLFrag = CTH(LL)
  ELSE
    DIRX = CTH(LL)
  END IF
CLAG END CHANGE

```

It was noted that it may be possible to achieve the same effect by utilizing the DRAGMODL option in the BASIC file. Per the documentation, adding the line:

```

CLAG *** COULD ACHIEVE DIRX=0 BY ADDING DRAGMODL INPUT TO BASIC
DRAGMODL NONE

```

to the BASIC file should disable air drag reduction for HEI fragments (between the burst point and the first component only) for distances greater than 24 inches. The original change was implemented until it can be ascertained that using the DRAGMODL option achieves the same effect.

Extract fragment weight, presented area data from shotline file for UEDDAM/FAA fragments

In order to extract the fragment weight and presented area, the change shown in Figure A-5 was made to subroutine AREACH.

```

C JKM 9/14/99 for UEDDAM
C WFRAG = SOLANG 37447000 980915
  WFRAG = FLOAT (INT (EXOB (LL))) 37447000 980915
  APFRAG = (EXOB (LL) - WFRAG) * 1000.
C JKM 9/14/99 for UEDDAM

```

Figure A-5. Change to AREACH

In COVART 5, the source code was identical. However, the change that was implemented needs to be conditional for operation outside UEDDAM. Again, since UEDDAM utilizes the HE threat type with the new fragment penetration model, the limiting condition was changed to use the IHPROJ variable in labeled common /HEPEN /. The conditional (IHPROJ .NE. 1) was used to indicate non-UEDDAM/FAA fragments.

```
CLAG  ADDED ACCESS TO IHPROJ
      COMMON /HEPEN / IHPROJ
CLAG  END CHANGE
:
CLAG  FOR UEDDAM
      IF ( IHPROJ .NE. 1 ) THEN
          WFRAG = SOLANG
      ELSE
          WFRAG = FLOAT (INT (EXOB (LL) ) )
          APPFRAG = (EXOB (LL) -WFRAG) * 1000.
      END IF
CLAG  END CHANGE
```

Change 6: TDYNSHR Array Added to Subroutine INIT

A new parameter, shear constant, had to be included to utilize the FAA penetration equations from which China Lake derived its new equations. The values of the shear constant for the COVART material types were stored in the array “DYNSHR”. This array contains 50 elements, and the values are hard-coded via a FORTRAN “DATA” statement in subroutine INIT. Figure A-6 shows the DATA statement for DYNSHR.

```
C JKM 11/14/00 UEDDAM
C JKM Updates 10/21/02
C   The units of DYNSHR are psi
C JKM  DATA DYNSHR / 7*188500.0, 7*30450.0, 30450.0, 35*188500.0 /
C JKM           Steel , Titanium, Al,
C JKM  DATA DYNSHR / 6*188500.0, 130529.4, 7*40029.0,40029.0,5*188500.0,
C JKM           Composite, Inconel
C JKM           +9*40029.0, 4*188500.0, 3*40029.0, 36403.2, 174039.2, 3*40029.0,
C JKM           +9*188500.0 /
C JKM 11/14/00 UEDDAM
```

Figure A-6. Changes to Subroutine INIT

Note that “DYNSHR” (as hard-coded in Figure A-6) only contains valid values for steel, titanium, aluminum, generalized composite, and Inconel. If different material types are desired and the Cs values for these materials are entered in the MATERIAL file (along with all other necessary information), the entered values will overwrite the hard-coded values for the corresponding material identification number. The user-defined values of Cs are read into the appropriate array locations in DYNSHR in the subroutine INPUTI. {See section A-5, [Change 10: Subroutine INPUTI](#).}

Material shear constants are utilized by the new UEDDAM/FAA penetration model. Since most of the material properties are initialized in the INIT subroutine, the shear constants were added to the INIT subroutine, in array TDYNSHR, rather than burying the initialization of these constants in the penetration model. Note that, in COVART 5, a distinction was made between the initialized data definitions and the variable arrays used in the computation. Whereas the UEDDAM version used the DYNSHR array to serve both purposes, COVART 5 stores the initialized data in the TDYNSHR array, and loads those values into the DYNSHR array for use in subsequent computations.

Additionally, in the UEDDAM implementation, two new material types, generalized composite and Inconel, were added to the material type arrays at material code index (JM) positions 37 and 38, respectively. However, in COVART 5, two material types had already been added in material code index positions 37 and 38: generalized composite and BMI composite. This precluded using those material code index positions for the new UEDDAM materials. Since there might be differences in material property characterization for the generalized composite properties defined in COVART 5 and the UEDDAM version, the easiest solution was to insert the new UEDDAM materials at unused material code index positions 46 and 47. {See section A-5, [Change 14: Update Material Properties for Generalized Composite and Inconel](#).}

Assuming that the two generalized composite characterizations are similar, the material shear constant for the UEDDAM generalized composite was also used for the COVART 5 generalized composite at MATCODE position 37:

```
CLAG
CLAG MATERIAL SHEAR CONSTANTS ADDED FOR FAA PENETRATION EQNS
CLAG UNITS OF TDYNshr ARE PSI
CLAG           Steel      , Titanium ,      Al      ,
+ DATA TDYNshr / 6* 188500.0 , 130529.4 , 8* 40029.0 , 5* 188500.0 ,
+           9* 40029.0 , 4* 188500.0 , 3* 40029.0 , 36403.2 , 7* 188500.0 ,
+           130529.4 , 36403.2 , 174039.2 , 3* 188500.0 /
CLAG           Composite , Inconel
CLAG END CHANGE
```

Change 7: DYNshr Added to Labeled Common /MATAL1/

To pass the values of “DYNshr” between subroutines, “DYNshr” was added to the common block “MATAL1”. Figure A-8 shows the changes to common block “MATAL1”.

```
C JKM 11/14/00 UEDDAM      COMMON /MATAL1/ BHN(50), RHO(50), SIGY(50), SIGU(50), TAUS(50)
      COMMON /MATAL1/ BHN(50), RHO(50), SIGY(50), SIGU(50), TAUS(50), 06099000 960315
      &
      DYNshr(50)
C JKM 11/14/00 UEDDAM
```

Figure A-8. Changes to Common Block MATAL1

In COVART 5, the array dimension had been changed to a parameter reference, MXMT = 50, from the parm.inc file. Array DYNshr, containing material shear constants (in psi), was added to labeled common /MATAL1/:

```
CLAG  ADDED SHEAR CONSTANT, DYNshr, FOR FAA PENETRATION EQNS
C      COMMON /MATAL1/ BHN(MXMT), RHO(MXMT), SIGY(MXMT), SIGU(MXMT),
C      &
C      TAUS(MXMT)
C      COMMON /MATAL1/ BHN(MXMT), RHO(MXMT), SIGY(MXMT), SIGU(MXMT),
C      &
C      TAUS(MXMT), DYNshr(MXMT)
CLAG  END CHANGE
```

Labeled common /MATAL1/ was updated in the following existing files:

```
main\COVART5:  areasa.f, areasd.f, arease.f, areasg.f, covart.f, fatefgctl.f,
               fatesl.f, fragctl.f, herayctl.f, init.f, initlib.f, inpute.f,
               inputf.f, inputz.f, thcoeff.f, zondat.f
jtcg-proj\PROJPEN:  projhe.f
```

Labeled common /MATAL1/ has a commented-out reference that was NOT updated in the following existing files:

```
Fatepen2.5\FATEPEN:  fluid.f
```

There are additional name-variant labeled common blocks that track the same material parameters but have different scope within the program. The following lists these blocks and the directories and files in which they were updated:

/MTAL1D/	damage\DAMAGE:	gtspft.f, pif2.f, pif3.f, pkinit.f
/MATL1F/	fatepen2.5\FATEPEN2:	(unchanged but commented in fateheinit.f)
	jtcg-frag\FRAGPEN:	frag-ctl.f, frag.f, jtcgfrinit.f
		(unchanged but commented in fragheinit.f)
	ueddam-frag\FAAPEN:	(new files) faafrag.f, faafragctl.f, faafrinit.f
/MTAL1J/	jtcg-proj\PROJPEN:	fluid.f, hlfspace.f, incdfu.f, jtcgapiinit.f,
pltplg.f,		proj-ctl.f, projdu.f, project.f, pstrai.f,
		ricoch.f
/MTAL1P/	fatepen2.5\FATEPEN2:	debris.f, fatepeninit.f, frag2.f, plate.f,

punch.f (unchanged but commented in
cloud.f)

Change 8: Subroutine INPUTZ

The materials “generalized composite” and Inconel are new materials for COVART. These materials were placed in the material array locations 37 and 38. These array locations are empty in COVART. COVART actually does a check to see if a material “37” has been specified in the inputs. If so, COVART flags this as an error and terminates the run. To avoid this, Subroutine INPUTZ was modified to remove this check. The lines that perform this check were simply commented out in subroutine INPUTZ.

This change was not seen in the inputz.f source file from the UEDDAM version, so no changes were made in the COVART5 version. However, any conditionals referencing materials “37” and “38” were reviewed (routines areach.f, areash.f, bploc.f, heivam.f) to see if they were relevant for this effort.

Conditionals in routines areach.f and areash.f are relevant for FATEPEN fragment penetration only. However, the following code, from both bploc.f and heivam.f, may need to be revisited relative to its use with the UEDDAM “generalized component” material:

```
C The original COVART calls cmpjec for handling Graphite Epoxy
C penetration. Projhe uses CTHK for the LOS, but cmpjec uses
C TLEQ for the LOS. This creates problems whenever there are
C combined plates on the shotline since TLEQ is the equivalent
C LOS for the combined plate not the LOS for this specific plate.
C This has been implemented for compatibility purposes but should
C be changed once testing is finished.
C
      IF (KK .EQ. 37) THEN
        CLOS = TLEQ(I)
        CBL = COSEQ(I)
      ELSE
        CLOS = CTHK
        CBL = COBL
      END IF
```

Change 9: New Labeled Common /JTYP2H/

To incorporate the changes required to output hole size statistics for components, several COVART subroutines were modified and two new common blocks were added. To pass the list of components requiring hole size calculations between subroutines, the common block JTYP2H was added:

```
COMMON /JTYP2H/ JTYPH(9999) .
```

New labeled common /JTYP2H/ was declared as below:

```
CLAG  ADDED BLOCK TO PASS LIST OF COMPONENTS REQUIRING HOLE SIZE CALCULATIONS
      COMMON /JTYP2H/ JTYPH(MXHOUT)
CLAG
```

The array dimension parameter, MXHOUT = 9999 array, was added to the parm.inc file:

```
CLAG  NEW PARAM MXHOUT FOR UEDDAM HOLESIZE ARRAYS
C     INTEGER MXDPAL, MXSYGP, PBBLCK, MXMTCD, MXPNCT
      INTEGER MXDPAL, MXSYGP, PBBLCK, MXMTCD, MXPNCT, MXHOUT
CLAG  END CHANGE
:
CLAG  NEW PARAM MXHOUT FOR UEDDAM HOLESIZE ARRAYS
C     MXHOUT - Maximum Number Components to Output Holesize Statistics
C
      PARAMETER (MXHOUT = 9999)
C
CLAG  END CHANGE
```

Labeled common /JTYP2H/ was added in the following existing files:

main\COVART5: covart.f, inputi.f

Labeled common /JTYP2H/ is used in the new UEDDAM module:

main\COVART5: faasl.f

Change 10: New Labeled Common /INOUT7/

Additionally, the common block INOUT7 was added:

```
COMMON /INOUT7/ IOHOLE,
```

where IOHOLE = 12 defines the unit number of the COVART output file to be passed to UEDDAM for analysis.

In COVART5, labeled common /INOUT7/ had already been declared. The IOHOLE parameter was added to the existing common block:

```
CLAG  ADDED HOLE SIZE OUTPUT FILE (IOHOLE) FOR UEDDAM
C     COMMON /INOUT7/ ICOVBX, IOCOVX
      COMMON /INOUT7/ ICOVBX, IOCOVX, IOHOLE
CLAG  END CHANGE
```

Labeled common /INOUT7/ was modified in the following existing files:

main\COVART5: areasb.f, bploc.f, covart.f, init.f, inputb.f, inputc.f

Labeled common /INOUT7/ was used in the new UEDDAM modules:

main\COVART5: faasl.f

Labeled common /INOUT7/ had a commented-out reference that was NOT updated in the following existing files:

main\COVART5: initlib.f

Change 11: Subroutine INPUTI

Hole Size Computation

The particular component identification numbers for which to compute hole sizes are read into COVART in the subroutine INPUTI (see Figure A-9).

```
ELSE IF ( DATTYP .EQ. 'HOLEOUT ' ) THEN 15406320CFAA060509
  IF ( TPHY ) THEN 15406340CFAA060509
    READ( RECORD, '(I8)' ) ( ICHO(ICH), ICH = 1, 8 ) 15406360CFAA060509
    DO 520 ICH = 1, 8 15406380CFAA060509
      IF ( ( ICHO(ICH) .GT. 0 ) .AND.
&          ( ICHO(ICH) .LE. JTYPI(I) ) ) THEN
        ITOP = I
        IBOT = 1
530      CONTINUE
        IF ( ITOP .GE. IBOT ) THEN
          ITEST = ( ITOP + IBOT ) / 2
          IF ( ICHO(ICH) .GT. JTYPI(ITEST) ) THEN
            IBOT = ITEST + 1
          ELSE IF ( ICHO(ICH) .LT. JTYPI(ITEST) ) THEN
            ITOP = ITEST - 1
          ELSE
            JTYPH(ITEST) = 1
            GO TO 520
          END IF
          GO TO 530
        END IF
      END IF
      WRITE( IOP, '(5X,A,I8,A)' ) 15406500CFAA060509
&      '***** WARNING: UNKNOWN COMPONENT', 15406520CFAA060509
&      ICHO(ICH), ' ON HOLEOUT RECORD.' 15406540CFAA060509
      WRITE( IOP, '(A8,A72)' ) DATTYP, RECORD 15406560CFAA060509
520    CONTINUE 15406580CFAA060509
  ELSE 15406600CFAA060509
    WRITE( IOP, '(5X,A)' ) 15406620CFAA060509
&    '***** WARNING: HOLEOUT RECORD OUT OF PLACE. THIS ONE:' 15406640CFAA060509
    WRITE( IOP, '(A8,A72)' ) DATTYP, RECORD 15406660CFAA060509
  END IF 15406680CFAA060509
  WRITE( IOP, '(A8,A72)' ) DATTYP, RECORD 15406700CFAA060509
  WRITE( IOP, '(A8,A72)' ) DATTYP, RECORD 15406720CFAA060509
  WRITE( IOP, '(A8,A72)' ) DATTYP, RECORD 15406740CFAA060509
  END IF 15406760CFAA060509
```

Figure A-9. Changes to Subroutine INPUTI for Hole Size Calculations

In COVART 5, the conditional statements above were added into the existing source. Numbered FORMAT statements were used in place of the embedded formats for the READ and WRITE statements. The /JTYP2H/ labeled common block was also added to the routine.

```

CLAG  ADDED BLOCK TO PASS LIST OF COMPONENTS REQUIRING HOLE SIZE CALCULATIONS
COMMON /JTYP2H/ JTYPH(MXHOUT)
CLAG
:
CLAG  ADDED UEDDAM MODS FOR READING IN COMPONENT IDS TO COMPUTE HOLE SIZE FOR
ELSE IF ( DATTYP .EQ. 'HOLEOUT ' ) THEN
  IF ( TPHY ) THEN
    READ( RECORD, 1970 ) ( ICHO(ICH), ICH = 1, 8 )
    DO 520 ICH = 1, 8
      IF ( ( ICHO(ICH) .GT. 0 ) .AND.
        & ( ICHO(ICH) .LE. JTYP1(I) ) ) THEN
        ITOP = I
        IBOT = 1
530      CONTINUE
        IF ( ITOP .GE. IBOT ) THEN
          ITEST = ( ITOP + IBOT ) / 2
          IF ( ICHO(ICH) .GT. JTYP1(ITEST) ) THEN
            IBOT = ITEST + 1
          ELSE IF ( ICHO(ICH) .LT. JTYP1(ITEST) ) THEN
            ITOP = ITEST - 1
          ELSE
            JTYPH(ITEST) = 1
            GO TO 520
          END IF
          GO TO 530
        END IF
        WRITE( IOP, 1700 ) ICHO(ICH)
        WRITE( IOP, 1710 ) DATTYP, RECORD
      ELSE
        IF ( ICHO(ICH) .NE. 0 ) THEN
          WRITE( IOP, 1720 ) ICHO(ICH)
        END IF
520      CONTINUE
    ELSE
      WRITE( IOP, 1730 )
      WRITE( IOP, 1710 ) DATTYP, RECORD
    END IF
CLAG  END CHANGE #1
:
CLAG  ADDED FOR UEDDAM HOLESIZE OUTPUT
1700 FORMAT (5X, '***** WARNING: UNKNOWN COMPONENT', I8,
  & ' ON HOLEOUT RECORD (IGNORED).')
1710 FORMAT (A8, A72)
1720 FORMAT (5X, '***** WARNING: OUT-OF-RANGE COMPONENT NUMBER', I8,
  & ' ON HOLEOUT RECORD (IGNORED).')
1730 FORMAT (5X, '***** WARNING: HOLEOUT RECORD OUT OF PLACE. THIS ONE:')
CLAG  END CHANGE
:
CLAG  ADDED FOR UEDDAM HOLESIZE ARRAY INPUT
1970 FORMAT( 8I8 )

```

Material Property Definitions

Note that “DYNshr” (as hard-coded in Figure A-6 {see section, [Change 5: DYNshr Array Added to Subroutine INIT](#)}) only contains valid values for steel, titanium, aluminum, generalized composite, and Inconel. If different material types are desired and the Cs values for these materials are entered in the MATERIAL file (along with all other necessary information), the entered values will overwrite the hard-coded values for the corresponding material identification number. The user-defined values of Cs are read into the appropriate array locations in DYNshr in the subroutine INPUTI (see Figure A-7).

```
ELSE IF ( DATTYP .EQ. 'MATPROP' ) THEN
  READ( RECORD, '(I8)' ) IMAT
  IF ( ( IMAT .GT. 0 ) .AND. ( IMAT .LT. 50 ) ) THEN
    READ( RECORD, '(8X,2A8,F8.0,F16.0)' ) HER(IMAT),
    &
    DER(IMAT), RHO(IMAT), DYNshr(IMAT)
    JTCGF( IMAT ) = 100 * IMAT
    END IF
```

Figure A-7. Changes to Subroutine INPUTI for Material Properties

In COVART 5, the source code was identical. However, the change that was implemented needs to reflect both the JTCG and UEDDAM/FAA fragment methodology settings. Since UEDDAM utilizes the HE threat type with the new fragment penetration model, a conditional was inserted using the IHPROJ variable in labeled common /HEPEN /. The conditional ‘IHPROJ .NE. 1’ was used to indicate the JTCG fragment case. Additionally, the embedded format for the READ statement was replaced with a numbered FORMAT statement:

```
CLAG  ADDED FOR FAA PENETRATION EQNS
COMMON /HEPEN / IHPROJ
CLAG  END CHANGE
:
CLAG  BEGIN CHANGE #2 FOR USER-DEF'D MATPROP TO OVERRIDE DEFAULTS FOR UEDDAM
ELSE IF ( DATTYP .EQ. 'MATPROP' ) THEN
  READ( RECORD, '(I8)' ) IMAT
  IF ( ( IMAT .GT. 0 ) .AND. ( IMAT .LT. 50 ) ) THEN
    READ( RECORD, 1980 ) HER(IMAT),
    &
    DER(IMAT), RHO(IMAT), DYNshr(IMAT)
    IF ( IHPROJ .NE. 1 ) THEN
      JTCGF( IMAT ) = 100 * IMAT
    ELSE
      IFAAF( IMAT ) = 100 * IMAT
    END IF
  END IF
CLAG  END CHANGE #2
:
CLAG  ADDED FOR MATERIAL PROPERTY OVERRIDE DATA INPUT
1980  FORMAT( 8X, 2A8, F8.0, F16.0 )
```

Change 12: Hole Size Computation and Output in (Former) Subroutine AREASN

In the subroutine AREASN, the resulting hole size and penetration information is written to a COVART output file for passing to UEDDAM (see Figure A-10).

```
IF ( JTYPH( ICNO(N) ) .EQ. 1 ) THEN
  IF ( VR .GT. 0.0 ) THEN
    IPENET = 1
    AHOLE = APFRAG / COSOBL( N )
    AHOLE = MIN( AHOLE, 9999999999.0 )
  ELSE
    IPENET = 0
    AHOLE = 0.0
  END IF
  WRITE( IOHOLE, '(2I8,F16.3,F8.4,I8)' ) ISRAY, JCOMP(N), AHOLE,
&                                     COSOBL(N), IPENET
END IF
```

Figure A-10. Changes to Subroutine AREASN for Hole Size Calculations

In COVART 5, the old AREASN routine was replicated across the different penetration libraries with modifications relevant to each library. Since the hole size computation is specific to the UEDDAM/FAA penetration library, the changes above were implemented there in file faasl.f. The embedded format for the WRITE statement was replaced with a numbered FORMAT statement:

```
CLAG  ADDED HOLE SIZE COMPUTATION AND PEN INFO OUTPUT
IF ( JTYPH( ICNO(N) ) .EQ. 1 ) THEN
  IF ( VR .GT. 0.0 ) THEN
    IPENET = 1
    AHOLE = APFRAG / COSOBL( N )
    AHOLE = MIN( AHOLE, 9999999999.0 )
  ELSE
    IPENET = 0
    AHOLE = 0.0
  END IF
  WRITE( IOHOLE, 2180 ) ISRAY, JCOMP(N), AHOLE,
&                                     COSOBL(N), IPENET
END IF
CLAG  END CHANGE
```

Change 13: Increased Array Sizes for JTYPE Elements and PK Tables in File parm.inc

Other, minor modifications were made to COVART to accommodate larger data sizes, handle binary reads with different compilers, and identify new material types (i.e., generalized composite and Inconel).

In COVART 5, the maximum number of JTYPE elements was set using parameter MXJTYP = 9999 in file parm.inc. This value was increased to 40000 to accommodate the larger number of unique critical components modeled in current target descriptions.

```
CLAG ADJUSTED FOR UEDDAM/FAA FRAG PENETRATION
C     PARAMETER (MXJTYP = 9999)
      PARAMETER (MXJTYP = 40000)
CLAG END CHANGE
```

In COVART 5, the maximum number of locations for PK table information for the VEL and PRK arrays was set using parameter MXPCLC = 20000 in file parm.inc. This value was increased to 25000 to accommodate the larger number of tables used with current analyses.

```
CLAG ADJUSTED for UEDDAM
C     PARAMETER (MXPCLC = 20000)
      PARAMETER (MXPCLC = 25000)
CLAG END CHANGE
```

Change 14: Update Material Properties for Generalized Composite and Inconel

Other, minor modifications were made to COVART to accommodate larger data sizes, handle binary reads with different compilers, and identify new material types (i.e., generalized composite and Inconel).

In COVART 5, labeled common /MATUSE/ contains arrays of flags indicating which penetrator and target material types are valid for use with specific penetration methodologies. Array IFAAF was added for use with the UEDDAM/FAA fragments:

```
CLAG  ADDED IFAAF(MXMTCD) ARRAY FOR FAA PENETRATION EQN. IN UEDDAM (SEE JTCGF)
C      COMMON /MATUSE/ JTCGF(MXMTCD), IFATEF(MXMTCD), IRODF(MXMTCD),
C      & IPROJF(MXMTCD), IALY(MAXWGT)
      COMMON /MATUSE/ JTCGF(MXMTCD), IFATEF(MXMTCD), IRODF(MXMTCD),
      & IPROJF(MXMTCD), IFAAF(MXMTCD), IALY(MAXWGT)
CLAG  END CHANGE
```

Labeled common /MATUSE/ was updated in the following existing files:

main\COVART5: covart.f, fasgn2.f, fatefgctl.f, fragck.f, fragctl.f, herayctl.f,
init.f, initlib.f, inpute.f, inputf.f, inputh.f, inputi.f, inputz.f,
slchk.f

There was an additional name-variant labeled common block, /MTUSEP/, that tracks the same material parameters but has different scope within the program. /MTUSEP/ was updated in the following routines:

fatepen2.5\FATEPEN2: debris.f, fateheinit.f, fatepeninit.f

Note that, in COVART 5, a distinction was made between the initialized data definitions and the variable arrays used in the computation. So values for the IFAAF array are initialized in the INIT routine from data contained in new array IFAAT. Settings in the JTCGT array (for use with JTCG fragments) were the basis for the new IFAAT array for use with the UEDDAM/FAA fragments. However, entries were modified to reflect the acceptable penetrator materials used in UEDDAM and the relocated entries for the new generalized composite and Inconel materials.

Subroutine INIT now contains the following data for IFAAT:

```
CLAG  ADDED VALID MATERIAL ARRAY INFO FOR FAA PENETRATION EQNS
CLAG  (JTCGT BASIS, WITH UEDDAM COMPOSITE, INCONEL AT MATCODES 46, 47)
      DATA IFAAT / 0101, 0202, 0303, 0404, 0505, 0606, 0700, 0800,
      + 0900, 1000, 1100, 1200, 1300, 1400, 1500, 1600,
      + 1700, 1800, 1900, 2000, 2100, 2200, 2300, 2400,
      + 2500, 2600, 2700, 2800, 2900, 3000, 3100, 3200,
      + 0, 0, 0, 0, 3700, 3800, 0, 0,
      + 0, 0, 0, 0, 0, 4600, 4700, 0,
      + 0, 0, 5100, 5200, 5300, 5400, 5500, 5600,
      + 5700 /
CLAG  END CHANGE
```

Subroutine INIT loads the initialized IFAAT data into the IFAAF array for use in subsequent computations:

```

C --- Initialize Target Material Codes, Text Descriptions
C --- And Default MATPKTAB Table Values
C
C      DO 1620 I = 1, JREST
C      DO 1620 I = 1, MXMTCB
C          JTCGF(I) = JTCGT(I)
C          IFATEF(I) = IFATET(I)
C          IRODF(I) = IRODT(I)
C          IPROJF(I) = IPROJT(I)
C          HER(I) = THER(I)
C          DER(I) = TDER(I)
C          MTPKFC(I) = ITMTPK(I)
CLAG  ADDED FOR FAA PENETRATION EQN
C          IFAAF(I) = IFAAT(I)
1620  CONTINUE

```

Subroutine INPUTI was updated to use the IFAAF value in lieu of the JTCGF value when UEDDAM/FAA penetration (as indicated by the 'IHPROJ .EQ. 1' conditional) is used:

```

C      ENSURE THAT THE MATERIAL CODE SPECIFIED IS A VALID
C      MATERIAL CODE FOR THE SPECIFIC TYPE OF PENETRATION
C      EQUATION THAT WILL BE USED
C
C          IF ( IPROJ .EQ. 1 ) THEN
C              MCHECK = IPROJF(JM)
C          ELSE IF ( IPROJ .EQ. 0 ) THEN
C              MCHECK = JTCGF(JM)
C          ELSE IF ( (IPROJ .LE. -2) .AND. (IPROJ .GE. -4) ) THEN
C              MCHECK = IFATEF(JM)
C          ELSE IF ( IPROJ .EQ. 2 ) THEN
C              MCHECK = IPROJF(JM)
C              MCHECK = MCHECK / 100
CLAG  ADDED CONDITIONAL FOR FAA PENETRATION EQN
CLAG  USING FLAG IHPROJ = 1
C              IF ( IHPROJ .EQ. 1 ) THEN
C                  JCHECK = IFAAF(JM)
C              ELSE
C                  JCHECK = JTCGF(JM)
C              END IF
CLAG  END CHANGE
C              JCHECK = JCHECK / 100
C              IF ( MCHECK .EQ. 0 .OR. JCHECK .EQ. 0 ) THEN
C                  GO TO 1822
C              ELSE
C                  GO TO 871
C              END IF

```

Updates Required for Pre-Existing UEDDAM Input Data

BASIC File

- Change IHPROJ value from '0' to '1'

JTYPE File

- Change material code index (JM) values of '37' to '46'
- Change material code index (JM) values of '38' to '47'

File: main\COVART5\faasl.f

```

SUBROUTINE FAASL(POWER)
CLAG SHELL OF FRAGSL.F USED TO IMPLEMENT FAA
C
C =====
C SUBROUTINE FAASL
C =====
C THIS ROUTINE CONTROLS THE COMPONENT CALCULATION LOOP
C
C =====
C INPUTS:
C
C
C =====
C OUTPUTS:
C
C
C =====
C Revision History:
C
C Rev.      Date      By      Description
C
C 0.0       22 Feb 2006 RRC    Initial implementation replacing areasn. This
C                               routine implements the jtcg fragment portion of
C                               areasn.
C
C =====

INCLUDE 'parm.inc'

COMMON /ALIAS / JTYP6(MXJTYP), JALIAS(MXALIS), LALIAS
COMMON /AMUS / AMU, AMU1, AMU2, IAMU1, IAMU2, YAW
COMMON /ANGS2 / IS, SINAMU
COMMON /AREANQ/ VSL(MXSLCP), WSL(MXSLCP)
COMMON /AREAPK/ PK(MAXKL)
COMMON /AREBC3/ IPH, IPL, IPHS, IPLS
COMMON /AREFHZ/ ILO(MAXLO), IPRINT, JLO(MXSLCP)
COMMON /ARERIC/ IRIC, NRIC, RPHE
COMMON /ARMANY/ NN
COMMON /ARMSUN/ JV

CFIR FIRE ROUTINE
CFIR
COMMON /ARNSTO/ FFUNCT(6)

CFIR
CFIR FIRE ROUTINE
CVCUT VELOCITY CUTOFF
CVCUT
COMMON /CUTOFF/ KUTOFF, KJA(MAXKL), KLCUT,
& PRA(MAXVEL,MAXKL), VC(MAXWGT,MAXKL), VCDELTA

CVCUT
CVCUT VELOCITY CUTOFF
COMMON /DIAM / DS, IAMU
COMMON /DRAG2 / ADRAG(MXSHAP+1), RHOAIR, IDRAG
COMMON /END / ITRGET, IPROJ, JKILL, ITHRT
CLAG EXTERNAL BURST COMMON (FOR UEDDAM)
COMMON /EXTBPD/ IEXTBP, WFRAG, VFRAG, MTFRAG, RLDFRG,
& APFRAG, RLFRAG

CLAG END CHANGE
C
LOGICAL LHEFIR
COMMON /FIRE / NAL, KTANK, AIRG(MXAGV), YAWAG(MXYAWV),
& FIREDT(MXFIRV), LHEFIR
C
COMMON /FPPKTV/ VPEN
COMMON /FUELXI/ IXTANK, IFUEL, IOFUNC
COMMON /FUELXR/ DFUEL, DFUNCT, FXTHK, FUNXTK, POFUNC(6)
COMMON /HEVRAY/ SOLANG, ISZONE, COSN, DIRX, DIRY, DIRZ, ISRAY
COMMON /IAT / ICOMP, ISW1, LDSKIP, LQS, LSKIP, MAT, NKILL(7)
COMMON /INITIA/ ACVEL, FIRE, FUNCT(MXSLCP), XFUNCT(MXSLCP)

```

```

COMMON /INITI2/ SFUNCT(4,MXSLCP)
COMMON /INOUT / IOC, IOP, IOP1, IOP2, IOP3, NLINES
CLAG BLOCK FOR ADDED HOLE SIZE OUTPUT FILE (IOHOLE) FOR UEDDAM
COMMON /INOUT7/ ICOVBX, IOCOVX, IOHOLE
CLAG END CHANGE
COMMON /JTYPE / NTYPE, JTYPI (MXJTYPI), JTYPI3 (MXJTYPI,MAXKL),
JTSUB (MAXCMP)
CLAG COMPONENT LIST FOR HOLESIZE CALCULATIONS
COMMON /JTYPI2H/ JTYPIH (MXHOUT)
CLAG
COMMON /JTYPI2I/ LJTYPI2 (MXJTYPI), MJTYPI2 (MXJTYPI)
COMMON /JTYPI2R/ DJTYPI2 (MXJTYPI)
COMMON /LOCCMP/ IFRSTC, NPREVC, NLASTC
COMMON /MAR40 / SLPK (MAXWGT*MAXVEL,MAXKL), PKILO (MAXLO,MAXKL,4)
COMMON /MATPKT/ MTPKFC (MXMTCD), NMTPKT, MTPKTN
COMMON /MATS / ICODES (MXSLMT), SHOTDN (MXSLCP)
COMMON /MPKILS/ NMPK, MPKILL (MXCPPK,MXDMG+1), NMPKDM, NMPKCX
COMMON /MUANG / PSH (90), PHISH (20),RMUMIN, RMUMAX, YAWS (20), IS2S
COMMON /PKGRP / IPKTYP (MXPKTB)
COMMON /PROJ1 / PROL (MAXWGT,4), PRON (MAXWGT,4), PROTD (MAXWGT,4),
DENRAT (MAXWGT)
C
REAL NSP
COMMON /SHATRC/ VVCI, WRSHWR, NSP, DFCI, PHIF, PHIT
C
COMMON /SHOTL1/ JM, KK, N, VR, V, WR, WT, NCOR, Q2, QRSUM
COMMON /SHOT40/ ICNO (MXSLCP), TL (MXSLCP), XAIR (MXSLCP),
COSOBL (MXSLCP), RFLUE (MXSLCP)
COMMON /SHOT41/ NOPK (MXSLCP), FRGMIN (MXSLCP,MAXKL),
JCOMP (MXSLCP), KKS (MXSLCP)
COMMON /SHOTQ1/ COSEQ (MXSLCP), TLEQ (MXSLCP), KKSEQ (MXSLCP),
TLGEQ (MXSLCP)
COMMON /SHOTQ2/ KKPREQ, TOLAG, PPCT (MXSLCP), PPCTP (MXSLCP),
INCMB (MXSLCP)
COMMON /SHOTQ3/ TLG (MXSLCP)
COMMON /THRET1/ PHI, V50, KF, AREAP
COMMON /THRET2/ DIA (MAXWGT,4), LSHAPE (MAXWGT), FL (MAXWGT),
LF (MAXWGT)
COMMON /DUCOMM/ V50N,T,AP,COSPHI,PLEN,SOBL,ABTP,SINPHI,NSHAPE
COMMON /PFRAG2/ PHIR
C
DIMENSION PFUNCT(6)

LOGICAL LCOMBN, LVULLG
LOGICAL LUNCOM, LCOMB

REAL Vs, WTs, PHIs
REAL Vr, WTr, PHIr
REAL LOS, COSBOL

INTEGER TgtMat, FrID, NCOREs, NCOREr, INCMB

FrID = JM
Vs = V
Vr = Vs
WTs = WT
WTr = WTs
IStateS = NCOR
IStateR = IStateS
N = 0
VPEN = V
WTPEN = WT
PHIPEN = PHI
NCORPN = NCOR
ISHAPP = ISHAPE
LCOMBN = .FALSE.
LCOMB = .FALSE.
DO 200 IFT = 1, 6
FFUNCT(IFT) = 0.0
PFUNCT(IFT) = 0.0
200 CONTINUE

```



```

C          MATERIAL TYPE FLAGGED AS CAUSING SPARK ON FRAGMENT IMPACT
C
      FUNCT(N) = 1.0
      SFUNCT(1,N) = 1.0
      XFUNCT(N) = 1.0
ELSE
C
C          NON-METAL PLATE
C
      VSL(N) = 0.0
      WSL(N) = 0.0
END IF
CFIR
CFIR  FIRE ROUTINES

IF(N .NE. NN) THEN
      ICALLV = 1
ELSE IF(IRIC .GT. 0) THEN
      ICALLV = 1
ELSE
      ICALLV = 0
      DO 180 LOOP = KLCUT, JKILL
        LP = JTYP3( ICNO(N), NKILL(LOOP) )
        IF ( LP .GT. 0 ) THEN
          IF ( IPKTYP(LP) .EQ. 4 .OR. IPKTYP(LP) .EQ. 5 ) THEN
            ICALLV = 1
            GO TO 1600
          END IF
          ELSE IF ( LP .LT. 0 ) THEN
            DO 190 JLP = 2, MPKILL( -LP, 1 )
              LLP = MPKILL( -LP, JLP )
              IF ( IPKTYP( LLP ) .EQ. 4 .OR.
                & IPKTYP( LLP ) .EQ. 5 ) THEN
                ICALLV = 1
                GO TO 1600
              END IF
            CONTINUE
          END IF
        END IF
      CONTINUE
END IF
180 CONTINUE
END IF
C
1600 CONTINUE
C
C Based on the plate combining flag, and the LUNCOM flag
C either use the combined plate data, or the individual
C plate data for penetration
C
IF (INCOMB(N) .NE. 0 .AND. LUNCOM) THEN
  TgtMat = KKS(N)
  KK = TgtMat
  LOS = TL(N)
  COBL = COSOBL(N)
  XAIR1 = XAIR(N)
  VPs = Vs
  WPs = WTs
  INCMB = 0
ELSE
  TgtMat = KKSEQ(N)
  KK = TgtMat
  LOS = TLEQ(N)
  COBL = COSEQ(N)
  XAIR1 = XAIR(N)
  INCMB = INCOME(N)
  IF (.NOT. LCOMB) THEN
    VPs = Vs
    WPs = WTs
  ENDIF
ENDIF
IF (N .EQ. 1 .AND. INCOMB(N) .NE. 0 ) INCMB = 0

```

```

      IF(ICALLV .EQ. 1 .AND. JTYP6(ICNO(N)) .GE. 0) THEN
        DF1HSV = DF1HH
        CALL AREASV
CLAG  NEW CALL FOR FAA FRAGS
      CALL FAAFragCtl ( FrID, TgtMat, LOS, WPs, RLFrag, VPs, AreaP,
&          WR, V50, Vr )
CLAG  END CHANGE

      IF (JTYP6(ICNO(N)) .GE. 0) THEN
&      IF ( (INCOMB(N) .NE. 0) .AND. (.NOT. LUNCOM) .AND.
        (.NOT. LCOMB)) THEN
&      IF ((VR .LT. 1.0) .OR. (WTr .LT. 0.0001)) THEN

C
C  IF PENETRATION OF A COMBINED PLATE DOES NOT OCCUR,
C  UNCOMBINE THE PLATE, ROLL BACK PENETRATION COMPUTATION
C  AND PRETEND THE COMBINATION NEVER OCCURRED.
C  SEE MDR NUMBER ...
C  THIS MAY, UNDER CERTAIN CIRCUMSTANCES PERMIT PENETRATION TO OCCUR
C  THAT OTHERWISE WOULDN'T, BUT USERS OBJECT MORE TO VISIBLE PK
C  ANOMALIES THAN TO INVISIBLE PENETRATION ANOMALIES
C
        NRIC = 0
        TgtMat = KKS(N)
        KK = TgtMat
        LOS = TL(N)
        COBL = COSOBL(N)
        INCMB = 0
        LUNCOM = .TRUE.

C
        V = VPEN
C
        WT = WTPEN
C
        PHI = PHIPEN
C
        NCOR = NCORPN
C
        QRSUM = QRSUMP
C
        DIA(JM,NCOR) = DIAPEN
C
        PROL(JM,NCOR) = PROLPN
C
        ISHAPE = ISHAPP
C
        CURDIA = CURDPN
C
        RLF1 = RLF1PN
C
        IAMU = IAMUPN
C
        AMU = AMUPEN
C
        RNF2 = RNF2PN
C
        RMF2 = RMF2PN
C
        DF2 = DF2PEN
C
        RNF3 = RNF3PN
C
        RMF3 = RMF3PN
C
        DF3 = DF3PEN
C
        RNT = RNTPEN
C
        RMT = RMTPEN
C
        DT = DTPEN
C
        PHIF = PHIFPN
C
        PHIT = PHITPN
C
        DF1H = DF1HPN
C
        STOT = STOTPN
C
        Q = QPEN
C
        Q2 = Q2PEN
C
        KK = KKS(N)
C
        T = TL(N) * COSOBL(N)
C
        DF1HH = DF1HSV
C
        LUNCOM = .TRUE.
C
        CEQSV = COSEQ(N)
C
        TLEQSV = TLEQ(N)
C
        KKSESV = KKSEQ(N)
C
        TLGESV = TLGEQ(N)
C
        INCM SV = INCOMB(N)
C
        PPCTSV = PPCT(N)
C
        PPCPSV = PPCTP(N)
C
        COSEQ(N) = COSOBL(N)
C
        TLEQ(N) = TL(N)

```

```

C          KKSEQ(N) = KKS(N)
C          TLGEQ(N) = TLG(N)
C          INCOMB(N) = 0
C          PPCT(N) = 1.0
C          PPCTP(N) = 1.0
C          CALL AREASV

CLAG CALL IS FOR UEDDAM/FAA PENETRATION
C          CALL JTCGFragCtl (FrID, IStateS, IStateR, COBL, TgtMat,
C          &                Los, WTs, Vs, Vr, WTr, V50, AreaP, INCMB,
C          &                COSEQ(N), LUNCOM, KKSEQ(N))
C          CALL FAAFragCtl ( FrID, TgtMat, LOS, WPs, RLFrag, VPs,
C          &                AreaP, WR, V50, Vr )
CLAG END CHANGE
      Wr = WTr
      NCOR = IStateS
      ELSE
      VPs = Vs
      VPr = Vr
      WPs = WTs
      WPr = WTr
      Wr = WTr
      LCOMB = .TRUE.
      NCOR = IStateS
      END IF
      ELSE IF (LCOMB) THEN
      Wr = WTr
      NCOR = IStateS
      ELSE
      Wr = WTr
      NCOR = IStateS
      END IF
      END IF
      ELSE
      V50 = 0.0
      END IF
C
C          IF ( INCOMB(N) .NE. 0 .AND. LCOMB) THEN
C          IF ( ( IFRSTC .GT. 0 ) .OR. ( JTYP6(ICNO(N)) .GE. 0 ) ) THEN
C
C              USE INTERPOLATED WEIGHTS, SPEEDS AND MOMENTA FOR PK
C
C              WRPEN = WR
C              VRPEN = VR
C              WR = WPs + PPCT(N) * ( WPr - WPs )
C              VR = VPs + PPCT(N) * ( VPr - VPs )
C              WR = WT + PPCT(N) * ( WR - WT )
C              VR = V + PPCT(N) * ( VR - V )
C              PHIRPN = PHI
C              PHIR = PHI
C          END IF
C          END IF
C
C          NCORR = NCOR
C          NCOR = NCORSV
C          DIAR = DIA(JM, NCORR)
C          DIA(JM, NCOR) = DIASAV
C          PROLR = PROL(JM, NCORR)
C          PROL(JM, NCOR) = PROLSV
C
C          RESTORE IMPACTOR CONFIGURATION FOR PK COMPUTATION
C
C          IF ( INCOMB(N) .NE. 0 ) THEN
C          IF ( ( IFRSTC .GT. 0 ) .OR. ( JTYP6(ICNO(N)) .GE. 0 ) ) THEN
C          V = VPK
C          WT = WTPK
C          PHI = PHIPK
C          IF ( JTYP6(ICNO(N)) .GE. 0 ) THEN
C          WTPK = WR
C          VPK = VR

```

```

C          PHIPK = PHIR
C          END IF
C          END IF
C          END IF
CLAG  ADDED HOLE SIZE COMPUTATION AND PEN INFO OUTPUT
      IF ( JTYPH( ICNO(N) ) .EQ. 1 ) THEN
          IF ( VR .GT. 0.0 ) THEN
              IPENET = 1
              AHOLE = APPFRAG / COSOBL( N )
              AHOLE = MIN( AHOLE, 9999999999.0 )
          ELSE
              IPENET = 0
              AHOLE = 0.0
          END IF
          WRITE( IOHOLE, 2180 ) ISRAY, JCOMP(N), AHOLE,
      S          COSOBL(N), IPENET
          END IF
CLAG  END CHANGE

      IF (LUNCOM) THEN
          TLSAV = TLEQ(N)
          TLEQ(N) = LOS
          KKSAV = KKSEQ(N)
          KKSEQ(N) = TgtMat
          COBSAV = COSEQ(N)
          COSEQ(N) = COBL
      ENDIF
C *****
C          DAMAGE CALCULATIONS
C *****

      CALL PKASGN (2, POWER)

CC
CC  SUBROUTINE FUELXO HANDLES EXIT FIRE AND FUNCTION THROUGH FLUID
CC  PKS
C  IF ( (IFUEL .GT. 0) .AND. (JTYP6(ICNO(N)) .GE. 0) ) THEN
C      CALL FUELXO( POWER )
C  END IF
CC  SUBROUTINE AREASO CONTROLS THE KILL LEVEL LOOP AND COMPUTES
CC  THE PROBABILITY OF KILL FOR EACH COMPONENT
CC
C  LO = LJTYP2( ICNO(N) )
C  DO 150 KL = KLCUT, JKILL
C      PK(KL) = 0.0
C      PKILO( N, KL, 4 ) = 0.0
CC      IF ( JTYP3( ICNO(N), NKILL(KL) ) .NE. 0 ) THEN
CC
CC          NOPK ACCOUNTS FOR DOUBLE-HIT EXCLUSION
CC          AND NON-CRITICAL COMPONENTS ON HEVART RAYS
CC          FRGMIN IS THE MINIMUM FRAGMENT SIZE TO EVALUATE
CC          (>0.0 FOR CONICAL BLAST VULNERABLE COMPONENTS
CC          IN HEIVAM)
CC
C          IF ( NOPK(N) .NE. 1 .AND. WT .GE. FRGMIN(N,KL) ) THEN
C              CALL AREASO( KL, LO, POWER )
C          ELSE
C              IF ( KL .EQ. 1 .AND. IPRINT .EQ. 1 ) THEN
C                  WRITE(IOP1, 2120) 0.0, WT, V, TLEQ(N), COSEQ(N),
C              +                JCOMP(N), JV, JM, KL, NCOR, IPHS, IPLS,
C              +                FFUNCT(6), PHI, V50
C              END IF
C          END IF
CC
C 150 CONTINUE

C
C          IF PLATES ARE COMBINED, RESET IMPACTOR CONFIGURATION

```

```

C          FOR PENETRATION
C
C      IF ( ( ( IFRSTC .GT. 0 ) .OR. ( JTYP6( ICNO(N) ) .GE. 0 ) ) .AND.
C      &      ( INCOMB(N) .NE. 0 ) ) THEN
C          V = VPEN
C          WT = WTPEN
C          PHI = PHIPEN
C          NCOR = NCORPN
C          QRSUM = QRSUMP
C          DIA(JM,NCOR) = DIAPEN
C          PROL(JM,NCOR) = PROLPN
C          ISHAPE = ISHAPP
C          CURDIA = CURDPN
C          RLF1 = RLF1PN
C          IAMU = IAMUPN
C          AMU = AMUPEN
C          RNF2 = RNF2PN
C          RMF2 = RMF2PN
C          DF2 = DF2PEN
C          RNF3 = RNF3PN
C          RMF3 = RMF3PN
C          DF3 = DF3PEN
C          RNT = RNTPEN
C          RMT = RMTPEN
C          DT = DTPEN
C          PHIF = PHIFPN
C          PHIT = PHITPN
C          DF1H = DF1HPN
C          STOT = STOTPN
C          Q = QPEN
C          Q2 = Q2PEN
C
C          VR = VRPEN
C          WR = WRPEN
C          PHIR = PHIRPN
C      END IF
CC
C      NCOR = NCORR
C      DIA(JM, NCOR) = DIAR
C      PROL(JM, NCOR) = PROLR
C
C      IF (LUNCOM) THEN
C          TLEQ(N) = TLSAV
C          KKSEQ(N) = KKSAV
C          COSEQ(N) = COBSAV
C      ENDIF
C
C -----
C Not at the last component or we want ricochet
C -----
C
C      IF ( ( N .LT. NN ) .OR. ( IRIC .GT. 0 ) ) THEN
C
C      No ricochet occurred, so continue processing the shotline
C
C          IF ( NRIC .LE. 0 ) THEN
C
C          The fragment has stopped. If this is not a combined plate, exit
C          Otherwise replace striking with residual and continue working
C
C              IF((WR .LE. 0.0) .OR. (VR .LE. 1.0)) THEN
C                  IF ( INCOMB(N) .LE. 0 .OR. LUNCOM) THEN
C                      GO TO 1776
C                  END IF
C              ELSE
C                  V = VR
C                  WT = WR
C              END IF
C
CC      START NEXT COMPONENT ON SHOTLINE
CC
CC

```



```

C          IF ( JTYPE6(ICNO(N)) .GE. 0 ) THEN
C              XTL = TL(N)
C          IF ( SHOTDN(N) .GT. 0.0 ) THEN
C              XTL = XTL / SHOTDN(N)
C          END IF
C          IF ( IOFUNC .GT. 0 ) THEN
CC
CC          IF AIR SPACE OR ULLAGE AFTER PREVIOUS COMPONENT,
CC          FUNCTION THROUGH FUEL NO LONGER APPLIES
CC
C              IF ( ( (KKS(N) .GT. 50) .AND.
C                  & (SHOTDN(N) .GT. 0.0) .AND.
C                  & (SHOTDN(N) .LE. 0.10) ) .OR.
C                  & (XAIR(NPREVC)*COSOBL(N) .GT. 0.1) ) THEN
C                  IOFUNC = 0
C                  DFUNCT = 0.0
C                  FUNXTK = 0.0
C                  DO 120 II = 1, 6
C                      POFUNC(II) = 0.0
C 120          CONTINUE
C              ELSE
C                  DFUNCT = DFUNCT + XTL + XAIR(N)
C                  IF ( KKS(N) .LT. 50 ) THEN
C                      FUNXTK = FUNXTK + XTL
C                  END IF
C              END IF
C          END IF
C          IF ( ( IFUEL .GT. 0 ) .AND.
C              & ( IFUEL .LT. N ) ) THEN
C              IF ( XFUNCT(NPREVC) .LE. 0.0 ) THEN
C                  DFUEL = DFUEL + XTL + XAIR(N)
C                  IF ( (XAIR(N) .GT. 0.10) .OR.
C                      & (N .EQ. NN) ) THEN
C                      IXTANK = IXTANK + 1
C                  END IF
C                  FXTHK = FXTHK + XTL
C              ELSE
CC
CC          ALREADY HAVE HAD A FUNCTION AFTER FUEL
CC
C              IFUEL = 0
C              IXTANK = 0
C              FXTHK = 0.0
C              DFUEL = 0.0
C          END IF
C          END IF
C          END IF

C -----
C      Check for the end of the shotline
C      Go back to 100 and loop again
C      Otherwise go to 300 and exit
C -----

          IF ( N .LT. NN ) THEN
          IF ( JTYPE6(ICNO(N)) .GE. 0 ) THEN
C
C      If a combined plate and this is the last contributor, reset flags
C      otherwise update residual and striking and continue
C
          IF (LCOMB) THEN
          IF (INCOMB(N) .LT. 0) THEN
              LCOMB = .FALSE.
          ELSE
              Vs = Vr
              V = Vr
              WTs = Wr
              WT = Wr
          ENDIF

```

```

C          ENDIF
C          IF ( INCOMB(N) .GT. 0 ) THEN
C              LCOMBN = .TRUE.
C          ELSE
C              VPEN = V
C              WTPEN = WT
C              PHIPEN = PHI
C              NCORPN = NCOR
C              QRSUMP = QRSUM
C              DIAPEN = DIA(JM,NCOR)
C              PROLPN = PROL(JM,NCOR)
C              ISHAPP = ISHAPE
C              CURDPN = CURDIA
C              RLF1PN = RLF1
C              IAMUPN = IAMU
C              AMUPEN = AMU
C              RNF2PN = RNF2
C              RMF2PN = RMF2
C              DF2PEN = DF2
C              RNF3PN = RNF3
C              RMF3PN = RMF3
C              DF3PEN = DF3
C              RNTPEN = RNT
C              RMTPEN = RMT
C              DTPEN = DT
C              PHIFPN = PHIF
C              PHITPN = PHIT
C              DF1HPN = DF1H
C              STOTPN = STOT
C              QPEN = Q
C              Q2PEN = Q2
C              LCOMBN = .FALSE.
C              KKPREQ = KKSEQ(N)
CC
C              WTPK = WT
C              VPK = V
C              PHIPK = PHI
C          END IF
C      END IF

C
C      If this is the last plate of combined plates OR
C      If this combined plate has be UnCombined OR
C      IF this is a single plate
C
C          IF (INCOMB(N) .LE.0 .OR. LUNCOM) THEN
C              Vs = Vr
C              V = Vr
C              WT = WTr
C              WTs = WTr
C              IStateS = IStateR
C              NCOR = IStateR
C          ENDIF
C          ENDIF
C          GO TO 100
C      ELSE
C
C      IF PLATE COMBINING HAD TO BE UNWOUND, RESTORE EQ VALUES
C
C          IF ( LUNCOM ) THEN
C              COSEQ(N) = CEQSV
C              TLEQ(N) = TLEQSV
C              KKSEQ(N) = KKSESV
C              TLGEQ(N) = TLGESV
C              INCOMB(N) = INCMSV
C              PPCT(N) = PPCTSV
C              PPCTP(N) = PPCPSV
C          END IF
C          GO TO 300
C      END IF

```

1776

CONTINUE

```

      END IF
    END IF
C
C   IF PLATE COMBINING HAD TO BE UNWOUND, RESTORE EQ VALUES
C
C     IF ( LUNCOM ) THEN
C       COSEQ(N) = CEQSV
C       TLEQ(N) = TLEQSV
C       KKSEQ(N) = KKSESV
C       TLGEQ(N) = TLGESV
C       INCOMB(N) = INCMSV
C       PPCT(N) = PPCTSV
C       PPCTP(N) = PPCPSV
C     END IF
C
C   300 CONTINUE
C
CVCUT VELOCITY CUTOFF
CVCUT
  IF(KUTOFF .EQ. -1) THEN
    RETURN
  END IF
CVCUT
CVCUT VELOCITY CUTOFF
C
  IF ( IPRINT .GT. 0 ) THEN
    WRITE ( IO1, * ) ' '
CRIC RICOCHET STATISTICS
CRIC
  IF ( IRIC .GT. 0 ) THEN
    IF ( NRIC .GT. 0 ) THEN
      WRITE ( IO1, 2160 ) RPHE, VR, WR
    END IF
  END IF
CRIC
CRIC RICOCHET STATISTICS
  END IF

  RETURN

2120 FORMAT( F10.4, 2F10.2, 2F10.4, 5I8, 2I5, F9.4, F10.4, F10.2 )
2130 FORMAT(1X, F5.3, F9.2, F9.1, 3(F9.3, F8.0, 2X), I6, 2I4, I3, I5,
+ I2, F9.4, F8.3, F10.1)
2160 FORMAT ( 5X, 'RICOCHET - RPHE =', F8.2, ' VR =', F8.2,
+ ' WR =', F8.2 )
2170 FORMAT(1X, 'PASS ', I1, 4X, 'MU ANGLE IN DEGREES =',
+ F5.1, 4X, 'PROBABILITY =', F6.3)
CLAG ADDED FOR UEDDAM HOLESIZE OUTPUT
2180 FORMAT( 2I8, F16.3, F8.4, I8 )
  END

```

```

SUBROUTINE FAAFrInit ( BHNP, RHOP, SIGYP, SIGUP, TAUSP,
DYNSHRP, EP, BULKP, DENALP, RAP, UP, CFP,
CCBFP, CBFP, CHP, CCFP, FCMF, FGP, FVP,
FTP, FSP, JRESP, PROLP, PRONP, PROTDP,
DENRATP, DIAP, FLUIP, CDP,
LSHAPEP, FLP, LFP, DIA2P, AMP )

CLAG SHELL OF JTCGFRINIT.F USED TO IMPLEMENT FAA
C
C =====
C      SUBROUTINE faaFrInit
C =====
C
C *****
C *** EXPORT RESTRICTED PER ARMS EXPORT CONTROL ACT. ***
C *** DISSEMINATE IN ACCORDANCE WITH DOD DIR 5230.25. ***
C *****
C      THIS SUBROUTINE INITIALIZES ARRAYS AND CONSTANT VALUES USED
C      THROUGHOUT THE PROGRAM
C
C
C
C =====
C INPUTS:
C
C
C =====
C OUTPUTS:
C
C
C =====
C Revision History:
C
C Rev.      Date      By      Description
C
C 4.5.001   13 Dec 2005 RRC     Initial implementation. Adapted from the original
C                                     COVART init subroutine.
C
C =====

      IMPLICIT NONE

      INCLUDE 'parm.inc'

      COMMON /DRAGF / CD(MXSHAP), FLUI(MXFLUT), VMAX
CLAG ADDED DYNshr FOR FAA PENETRATION EQNS
      COMMON /MATL1F/ BHN(MXMT), RHO(MXMT), SIGY(MXMT), SIGU(MXMT),
& TAUS(MXMT), DYNshr(MXMT)
CLAG END CHANGE
      COMMON /MATL2F/ E(MXMT), BULK(MXMT), DENAL(MXMT), RA(MXMT),
& U(MXMT), CF
&
      COMMON /MTFR1F/ CCBF(MXMT), CBF(MXMT), CH(MXMT), CCF(MXMT),
& FCM(MXMT)
&
      COMMON /MTFR2F/ FG(MXMT), FV(MXMT), FT(MXMT), FS(MXMT)
      COMMON /MTHE1F/ JRES, FIRE
      COMMON /PRJ1F / PROL(MAXWGT,4), PRON(MAXWGT,4), PROTDX(MAXWGT,4),
& DENRAT(MAXWGT)
&
      COMMON /RICDTF/ AM(MXMT), B(MXMT)
      COMMON /THRT2F/ DIA(MAXWGT,4), LSHAPE(MAXWGT), FL(MAXWGT),
& LF(MAXWGT)
&
      COMMON /THRT3F/ DIA2(MAXWGT)

C
C Global Variables
C
CLAG ADDED DYNshr FOR FAA PENETRATION EQNS

```

```

C      REAL      BHN, RHO, SIGY, SIGU, TAUS
REAL      BHN, RHO, SIGY, SIGU, TAUS, DYNshr
CLAG END CHANGE
REAL      E, BULK, DENAL, RA, U
REAL      CCBF, CBF, CH, CCF, FCM
REAL      FG, FV, FT, FS, FLUI
REAL      PROL, PRON, PROTD, DENRAT
REAL      DIA, FL, DIA2, FIRE
REAL      CD, VMAX, AM, B
INTEGER   LSHAPE, LF

C
C Local Variables
C
CLAG ADDED DYNshRP FOR FAA PENETRATION EQNS
REAL      BHNP (MXMT), RHOP (MXMT), SIGYP (MXMT), SIGUP (MXMT),
&          TAUSP (MXMT), DYNshRP (MXMT)
CLAG END CHANGE
REAL      EP (MXMT), BULKP (MXMT), DENALP (MXMT), RAP (MXMT),
&          UP (MXMT)
REAL      AMP (MXMT), BP (MXMT)
REAL      CCBFP (MXMT), CBFP (MXMT), CHP (MXMT), CCFP (MXMT),
&          FCM (MXMT)
REAL      FGP (MXMT), FVP (MXMT), FTP (MXMT), FSP (MXMT)
REAL      PROLP (MAXWGT,4), PRONP (MAXWGT,4), PROTD (MAXWGT,4),
&          DENRATP (MAXWGT)
REAL      DIA2P (MAXWGT), DIAP (MAXWGT,4), FLP (MAXWGT)
INTEGER   LSHAPEP (MAXWGT), LFP (MAXWGT)
REAL      CF, CFP, FIREP, FLUIP (MXFLUT)
INTEGER   JRES, JRESP, NMTPKT, NMTPKTP, MTPKTN, MTPKTNP
INTEGER   I, J
REAL      CDP (MXSHAP)

C
CF = CFP
JRES = JRESP

DO 800 I=1, MXFLUT
FLUI (I) = FLUIP (I)
800 CONTINUE

DO 810 I=1, MXSHAP
CD (I) = CDP (I)
810 CONTINUE

DO 100 I = 1, MXMT

BHN (I) = BHNP (I)
RHO (I) = RHOP (I)
SIGY (I) = SIGYP (I)
SIGU (I) = SIGUP (I)
TAUS (I) = TAUSP (I)
E (I) = EP (I)
BULK (I) = BULKP (I)
DENAL (I) = DENALP (I)
RA (I) = RAP (I)
U (I) = UP (I)
CCBF (I) = CCBFP (I)
CBF (I) = CBFP (I)
CH (I) = CHP (I)
CCF (I) = CCFP (I)
FCM (I) = FCM (I)
FG (I) = FGP (I)
FV (I) = FVP (I)
FT (I) = FTP (I)
FS (I) = FSP (I)
AM (I) = AMP (I)
CLAG ADDED INIT FOR DYNshRP VALUES
DYNshRP (I) = DYNshRP (I)
CLAG END CHANGE

```

```
100 CONTINUE

DO 200 I = 1, MAXWGT
DO 300 J = 1, 4
    PROL(I,J) = PROLP(I,J)
    PRON(I,J) = PRONP(I,J)
    PROTD(I,J) = PROTDP(I,J)
    DIA(I,J) = DIAP(I,J)
300 CONTINUE

DIA2(I) = DIA2P(I)
DENRAT(I) = DENRATP(I)
LSHAPE(I) = LSHAPEP(I)
FL(I) = FLP(I)
LF(I) = LFP(I)

200 CONTINUE

RETURN

C
END
```

File: ueddam-frag\FAAPEN\faaheinit.f

```

SUBROUTINE FAAHeInit (DIAP, LSHAPEP, FLP, LFP, DIA2P, PROLP,
DENRATP)
C
CLAG SHELL OF FRAGHEINIT.F USED TO IMPLEMENT FAA
CLAG NO CHANGES, SAME INIT OF FRAGMENT PARAMS
C
C =====|
C SUBROUTINE FAAHeInit |
C =====|
C
C *****|
C *** EXPORT RESTRICTED PER ARMS EXPORT CONTROL ACT. ***|
C *** DISSEMINATE IN ACCORDANCE WITH DOD DIR 5230.25. ***|
C *****|
C THIS SUBROUTINE INITIALIZES ARRAYS AND CONSTANT VALUES USED|
C FOR HE Fragments|
C
C
C
C =====|
C INPUTS: |
C |
C |
C =====|
C OUTPUTS: |
C |
C |
C =====|
C Revision History: |
C |
C Rev. Date By Description |
C 4.5.001 13 Dec 2005 RRC Initial implementation. Adapted from the original |
C COVART init subroutine. |
C |
C =====|

IMPLICIT NONE

INCLUDE 'parm.inc'

COMMON /PRJ1F / PROL(MAXWGT,4), PRON(MAXWGT,4), PROTD(MAXWGT,4),
DENRAT(MAXWGT)
COMMON /THRT2F/ DIA(MAXWGT,4), LSHAPE(MAXWGT), FL(MAXWGT),
LF(MAXWGT)
COMMON /THRT3F/ DIA2(MAXWGT)

C
C Global Variables
C
REAL PROL, PRON, PROTD, DENRAT
REAL DIA, FL, DIA2, FIRE
INTEGER LSHAPE, LF

C
C Local Variables
C
REAL PROLP(MAXWGT,4), PRONP(MAXWGT,4), PROTDP(MAXWGT,4),
DENRATP(MAXWGT)
REAL DIA2P(MAXWGT), DIAP(MAXWGT,4), FLP(MAXWGT)
INTEGER LSHAPEP(MAXWGT), LFP(MAXWGT)
INTEGER JRES, JRESP, NMPKTP, NMPKTN, MTPKTN, MTPKTNP
INTEGER I, J

C
C
C
DO 200 I = 1, MAXWGT
DO 300 J = 1, 4
DIA(I,J) = DIAP(I,J)
PROL(I,J) = PROLP(I,J)

```

300 CONTINUE

DENRAT(I) = DENRATP(I)
DIA2(I) = DIA2P(I)
LSHAPE(I) = LSHAPEP(I)
FL(I) = FLP(I)
LF(I) = LFP(I)

200 CONTINUE

RETURN
END

File: ueddam-frag\FAAPEN\faafragctl.f

```

SUBROUTINE FAAFragCtl ( FrID, TgtMat, TlEq, WT, RlFrag, V, AreaP,
&
WR, V50, Vr )

CLAG SHELL OF FRAG-CTL.F USED TO IMPLEMENT FAA
C
C =====
C SUBROUTINE AREASV
C =====
C
C
C =====
C INPUTS:
C
C
C =====
C OUTPUTS:
C
C
C =====
C Revision History:
C
C Rev.      Date      By      Description
C
C 0.0       22 Feb 2006 RRC    Initial implementation replacing areasv. This
C                               routine implements the JTGC fragment portion of
C                               areasv.
C
C =====
C
C
C IMPLICIT NONE
C
C INCLUDE 'parm.inc'
C
C COMMON /MTH1F/ JRES
C COMMON /DRAGF / CD(MXSHAP), FLUI(MXFLUT), VMAX
CLAG ADDED DYNshr FOR FAA PENETRATION EQNS
C COMMON /MATL1F/ BHN(MXMT), RHO(MXMT), SIGY(MXMT), SIGU(MXMT),
&
TAUS(MXMT), DYNshr(MXMT)
CLAG END CHANGE
C COMMON /THRT2F/ DIA(MAXWGT,4), LSHAPE(MAXWGT), FL(MAXWGT),
&
LF(MAXWGT)
C
C Global Variables
C
C INTEGER JRES
C REAL CD, FLUI, VMAX
CLAG ADDED DYNshr FOR FAA PENETRATION EQNS
C REAL BHN, RHO, SIGY, SIGU, TAUS, DYNshr
CLAG END CHANGE
C REAL DIA, FL
C INTEGER LSHAPE, LF
C
C Local Variables
C
C INTEGER KKPEN, LL
C REAL FLU, EXPOW, RHOT
C
C Subroutine parameters
C
C INTEGER FrID, TgtMat
C REAL TlEq, WT, RlFrag, V, AreaP, WR, V50, Vr
C
C -----
C Fragments through non-fluid materials
C -----
CLAG ADDED CODES FOR THE TWO NEW MATERIALS AT 46, 47
C IF ( TgtMat .LE. 38 ) THEN
```

```

      IF ( ( TgtMat .LE. 38 ) .OR.
+      ( TgtMat .EQ. 46 ) .OR. ( TgtMat .EQ. 47 ) ) THEN
CLAG END CHANGE
      KKPEN = TgtMat
      IF ( ( TgtMat .GE. 9 ) .AND. ( TgtMat .LE. 14 ) ) THEN
      KKPEN = 8
      END IF
      RHOt = RHO(KKPEN)

      CALL FAAfrag ( FrID, TgtMat, RHOt, TlEq, WT, RlFrag, V, AreaP,
      WR, V50, Vr )
      C -----
      C           Fragments through Fluid materials
      C -----
      ELSE IF ((TgtMat .GE. 51) .AND. (TgtMat .LE. JRES)) THEN
      V50 = 0.0
      VR = 0.0
      WR = WT
      FLU = FLUI(TgtMat - 50)
      C
      C PENETRATION HANDBOOK - EQUATION NUMBER 71
      CLAG DRAG COEF. = 1.0, SO REMOVED FROM EXPOW EXPRESSION
      EXPOW = AreaP * FLU * TlEq / ( WT + WT )
      IF (EXPOW .LT. 11.5) THEN
      VR = V * EXP(-EXPOW)
      END IF
      END IF
      C
      RETURN
      END

```

File: ueddams-frag\FAAPEN\faafrag.f

```

SUBROUTINE FAAfrag ( FrID, TgtMat, RHOt, TlEq, WT, V, AreaP,
WR, V50, Vr )

CLAG SHELL OF FRAG.F USED TO IMPLEMENT FAA
C
C =====
C SUBROUTINE FAAfrag
C =====
C
C MODIFIED VERSION OF FRAG SUBROUTINE FOR UEDDAM - EXPORT RESTRICTED DATA
C REMOVED
C
CFRG LARGE FRAGMENT ROUTINES
CFRG
C -----
C UPDATE 11-12-02 TO ADD TITANIUM, COMPOSITE, AND INCONEL
C UPDATE 11-12-02 TO IMPROVE AL DYNAMIC SHEAR (SHEAR CONSTANT)
C -----
C PENETRATION EQUATIONS FOR LARGE FRAGMENTS CODED FROM NAWCWD
C TESTING AND ANALYSIS. THESE EQUATIONS ARE BASED ON THE
C JTCG/ME PENETRATION EQUATIONS HANDBOOK. MODIFICATIONS HAVE BEEN
C MADE FOR THE CONSIDERATION OF LARGE SLOW FRAGMENTS. NO WEIGHT
C LOSS OCCURS AND THE FRAGMENT AP IS CALCULATED OFF LINE IN THE
C UEDDAM CODE.
C
C THIS ROUTINE UTILIZES THE PENETRATION EQUATIONS FOR FRAGMENTS
C TO APPROXIMATE CHANGES IN THE MOTION OF A FRAGMENT IMPACTOR
C WHICH OCCUR AS A RESULT OF PENETRATION THROUGH SHOTLINE COMPONENT
C PLATES. THIS ROUTINE IS USED ONLY WHEN FRAGMENTS ARE UNDER
C CONSIDERATION IN THE MODEL.
C
C NO FRAGMENT SHATTER OR MASS LOSS
C
C NO FRAGMENT RICOCHET
C
C RHOt , R*4, - A VALUE WHICH IS THE SPECIFIC WEIGHT OF A
C FRAGMENT.
C
C SIGF , R*4, - A VALUE WHICH IS THE ULTIMATE STRENGTH OF A
C FRAGMENT. (TENSILE STRENGTH)
C
C VV , R*4, - A VALUE WHICH IS THE VELOCITY OF AN IMPACTOR
C VV = V

CLAG
C Subroutine arguments:
C INTEGER FrID input: fragment index
C INTEGER TgtMat input: impacted component material index
C REAL RHOt input: target component specific weight (grains/in**2)
**check units**
C REAL TlEq input: equivalent LOS of component (TlEq solids; Tl
fluids)
C REAL WT input: fragment weight at impact (assumes no mass loss)
C REAL RlFrag input: fragment presented area perimeter length
C REAL V input: fragment velocity at impact
C REAL AreaP input: fragment presented area
C REAL WR output: fragment residual weight (= WT, assumes no mass
loss)
C REAL V50 output: fragment V50 (penetration cutoff limiting
velocity)
C REAL Vr output: fragment residual velocity
C
C Local variables:
C INTEGER KF local: material type index for fragment material
C REAL RHOt local: fragment specific weight (grains/in**2)
C REAL SIGF local: fragment ultimate strength aka tensile strength
(grains/in**2)
C REAL VV local: fragment velocity at impact (ft/sec)

```

```

C REAL    rmkg      local: fragment mass (converted from grains/in**2 to
kg/cc)
C REAL    Tm        local: equivalent LOS thru component (converted from
inches to meters)
C REAL    rmGd      local: shear constant for component material (converted
from psi to kPa)
C REAL    rmL       local: fragment presented area perimeter length
(converted from inches to meters)
C REAL    V50m     local: FAA computed V50 value (in meters per second)
C REAL    Q4        local: Computed factor for residual velocity (English
units)

      IMPLICIT NONE

      INCLUDE 'parm.inc'

      COMMON /EXTBPF/ IEXTBP, WFRAG, VFRAG, MTFRAG, RLDFRG,
APFRAG, RLFRAG
      COMMON /MATL1F/ BHN(MXMT), RHO(MXMT), SIGY(MXMT),
SIGU(MXMT), TAUS(MXMT), DYNshr(MXMT)
      COMMON /THRT2F/ DIA(MAXWGT,4), LSHAPE(MAXWGT), FL(MAXWGT),
LF(MAXWGT)

CLAG  TYPE DECL FOR LABELED COMMON VARS
      INTEGER IEXTBP, MTFRAG
      INTEGER LF, LSHAPE
      REAL    WFRAG, VFRAG, RLDFRG, APFRAG, RLFRAG
      REAL    BHN, RHO, SIGY, SIGU, TAUS, DYNshr
      REAL    DIA, FL

CLAG  LOCAL VAR DECLARATIONS
      INTEGER KF, NRIC, N
      REAL    VV, RHOF, SIGF, AP
      REAL    rmkg, Tm, rmGd, RmL, V50m, Q4

CLAG  SUBROUTINE ARGUMENT DECLARATIONS
      INTEGER FrID, TgtMat
      REAL    RHOt, TLEq, WT, V, AreaP, WR, V50, Vr

C *****
C Initialize variables used for penetration
C *****
C
      VV = V
      WR = WT
      KF = LF(FrID)
      RHOF = RHO(KF)
      SIGF = SIGU(KF)
CLAG ADDED FROM UEDDAM
      AP = APFRAG
      AREAP= AP
CLAG  END CHANGE

C *****
C Calculate the Ballistic Limit (V50)
CLAG USING FAA EQUATION
C *****
CLAG
CLAG Reference 1977 FAA RD-77-44
CLAG
CLAG L = Fragment presented area perimeter (meters)
CLAG Cs = Target shear constant (Pascals)
CLAG t = Target thickness (meters)
CLAG m = Fragment mass (kilograms)
CLAG theta = Angle of obliquity of the fragment impact (degrees)
CLAG
CLAG V50 = sqrt( (2. * L * Cs * t**2) / (m * (cos(theta))**2) )
CLAG
CLAG **** NOTE FOR UEDDAM USE ONLY, NOT ADDRESSING COMBINED PLATES ****
CLAG
CLAG Factors used in unit conversion:

```

```

CLAG      7000 grains/pound, 2.205 pounds/kilogram, 39.37 inches/meter,
CLAG      6894.757 Pascals/psi, 3.281 feet/meter

      rmkg = WT / 7000. / 2.205
      Tm = TlEq / 39.37
      rmGd = DYNshr(TgtMat) * 6894.757
      rmL = RLFrag / 39.37
      V50m = SQRT(2. * rmL * rmGd * Tm**2. / rmkg)
      V50 = V50m * 3.281

C
C *****
C ricochet handling - NO RICOCHET
C *****
C
CLAG Access to these vars commented out
CLAG Assume not really used since just stats to echo
      IF ( VV .LE. V50 ) THEN
CLAG Assume NRIC, N not really used since no ricochet & only stats echo
CLAG but vars/common commented out so might have garbage values...
      NRIC = N
CLAG
      VR = 0.0
      RETURN
      END IF

C *****
C Penetration handling - calc residual vel
C *****
C
C PENETRATION HANDBOOK - EQUATION NUMBER 57
C
C Q4 = p * Ap * T / W * (cosine theta)
C
C Q4 = RHot * AP * TlEq / WT
C
C PENETRATION HANDBOOK - EQUATION NUMBER 56
C
C Vr = SQRT((V **2) - (V50 **2)) / (1 + Q4)
C
C VR = SQRT (VV ** 2 - V50 ** 2) / (1. + Q4)
C
      RETURN

CFRG
CFRG ALL FRAGMENT ROUTINES
C
      END

```

APPENDIX B

**AIRCRAFT GEOMETRY DEVELOPMENT REFERENCE: EXCERPTS
FROM FASTGEN 6.1 TARGET DESCRIPTION DOCUMENT**

This appendix contains excerpts from *FASTGEN 6.1: Target Description Document* and is intended to aid the user in developing the aircraft geometric target description. These excerpts are only for the data that is necessary to run UEDDAM.

GENERAL DESCRIPTION

INTRODUCTION

Prediction of damage to a target caused by ballistic impact of projectiles has been an important long time goal of military analysts. A number of analytical procedures and target description techniques have evolved. One widely accepted approach to vulnerability analysis is the shotline method. This method involves projecting a number of parallel rays through the target from a specified direction and describing the encounters along each ray. The result is a sequential list of components, subsets of the target, which are encountered by a shotline.

Fast Shotline Generator (FASTGEN) traces the path of a projectile's shotline through a target. The target is composed of objects called components. These components are modeled by generating a three-dimensional target database. The set of components encountered along a shotline is arranged in the order of encounter. This sequenced list of components along a shotline is called a line of sight (LOS).

With the release of FASTGEN 6, the target handling, ray generation and ray tracing features of the code were modularized in order to integrate with COVART6. The ray tracing functionality of FASTGEN 6 is a library (*.dll or *.so) called by COVART, with the other functions being integrated into COVART6. In spite of this significant change in the structure of the code, the FASTGEN target format remains unchanged from FASTGEN 5, and with the exception of some deprecated elements, FASTGEN 4 as well.

This document describes the data and format for creating a target file for use by FASTGEN.

Items in *italicized blue text* refer to deprecated features, that is, features supported in the current version of the software, but will be removed in future versions of the software and as such their use is discouraged. Items in *italicized bold text* are important FASTGEN requirements that the user must heed in order to produce correct and reliable results using FASTGEN and/or COVART.

TARGET CONSTRUCTION OVERVIEW

INTRODUCTION

System-level ballistic vulnerability analyses require a user-defined representation of the target model. All software in this category uses the concept of ray tracing to model the

interaction between the threat and the target. The purpose of this manual is to provide an easy-to-use resource for the FASTGEN target model builder.

COVART6 TERMINOLOGY

With the release of COVART6, there is new terminology that is important for the FASTGEN model builder to understand. With integration of ray tracing into COVART6, it is no longer necessary to run FASTGEN separately before running COVART since COVART6 performs ray tracing as shotline or ray intersection data is required. When running COVART6 in this integrated mode, a MASTER file is created and used in place of the COVART BASIC file that describes the type of analysis to perform (e.g., gridded HE threat, single HE threat, gridded KE threat, etc.). When running COVART6 in this integrated mode, the FASTGEN target file simply becomes another input file to COVART specified in the MASTER file. The FASTGEN output files that were formerly generated as input to COVART are no longer required and are not generated when running in integrated mode.

To maintain backward compatibility with existing FASTGEN and COVART 5 files, COVART6 can also be run in a legacy FASTGEN or legacy COVART mode. In legacy FASTGEN mode, the MASTER file simply identifies the mode (FASTGEN) and indicates the name of the FASTGEN target file. The FASTGEN portion of the code is run to ray trace the target and FASTGEN output files are generated for use in a subsequent COVART legacy run.

In legacy COVART mode, the MASTER file identifies the mode (COVART5) and points to the COVART BASIC file. All other FASTGEN output and COVART input files are used as needed in COVART6 legacy mode just as they were used in the FASTGEN/COVART5 data processing flow.

TARGET CONSTRUCTION PROCESS

A failure modes, effects, and criticality analysis (FMECA) should be accomplished prior to preparing a FASTGEN target description. A FMECA is a multidisciplinary (reliability, maintainability, safety, survivability, etc.) system design evaluation procedure. The FMECA in itself does not determine the vulnerability characteristics of a target. The FMECA should be used as source data to determine which components are included in a FASTGEN target description.

Given a list of components determined by the FMECA, the geometric shape of each component is digitized by developing a three-dimensional target database. The target database can be at any level of resolution consistent with FASTGEN constraints. Components of the target database should be larger than the several sub-grids, and elements of components should be larger than the sub-grid. The target database includes all flight and mission critical components of the operationally configured target. It also

includes all components which effectively degrade the ability of a threat effect to cause damage/failure (i.e. providing shielding). All target surfaces (skin and transparencies) are also modeled in detail because they provide some degree of shielding or masking of other components.

A FASTGEN target database is based on the paradigm that target surfaces can be approximated by a series of adjacent lines, triangles, quadrilaterals, cones, cylinders, spheres, and hexahedrons. This database preparation process is intricate and must be accomplished according to inherent FASTGEN logical requirements and limitations.

The source data for the geometric database may be obtained in several forms: engineering drawings, CAD/CAM database, NASTRAN internal loads model for structural data, and other digitized data. Using digitized source data greatly reduces the database generation task, but not the debugging and error correction. The use of non-FASTGEN based data increases the time required to finalize the target description.

Target Coordinate System

The FASTGEN model builder constructs a target in the target spatial coordinate system. The origin of this Cartesian coordinate system is typically defined to be the target's center of gravity. However, the origin may be placed at the most forward point (i.e. nose) of the target or the target's geometric center. Referring to Figure B-1, the positive x-axis is oriented forward along the aircraft's centerline. The positive y-axis is oriented along the aircraft's left wing (as viewed by the pilot). The positive z-axis is oriented out the top of the aircraft, completing the right-handed coordinate system.

This target orientation is required for any ballistic vulnerability analysis performed using FASTGEN and COVART. Shot line processing algorithms and other computations inside FASTGEN and COVART expect this orientation. Any other orientation adopted by the user will produce incorrect analysis results.

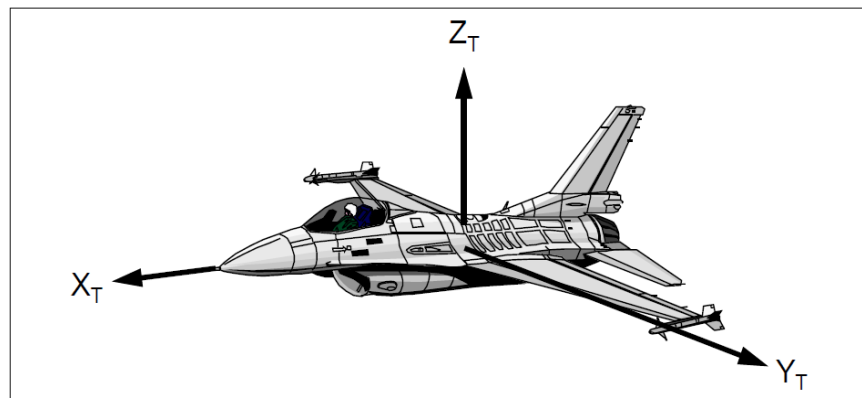


Figure B-1. FASTGEN's Target Spatial Coordinate System.

Plate Mode and Volume Mode Components

FASTGEN accommodates two distinct types of geometric components, plate mode and volume. Plate mode components have extents specified by coordinates in two dimensions and a constant thickness in the third dimension. Volume mode components, on the other hand, are three dimensional in that they are defined in space by a collection of primitives, each having extents defined by coordinates in three dimensions. However, volume mode components have no explicitly defined thicknesses. The volume mode component thickness is defined by virtue of the volume enclosed by the elements comprising the component.

Geometric Overlaps or Interferences

As stated above, plate mode components are two dimensional with normal thicknesses and implied Line-of-Sight (LOS) thicknesses that are dependent upon shotline angles. Furthermore, the thicknesses of volume mode components are not explicitly defined so the FASTGEN model building task can sometimes be overwhelming to the inexperienced user. The various plate and volume mode components can overlap and interfere, often with unanticipated and unintended consequences to the vulnerability analysis.

TARGET DATABASE ORGANIZATION

FASTGEN assumes that components are spatially compact (i.e. the absolute differences between the coordinate minimums and maximums are small in size) and contiguous. Keeping components small in size decreases the number of times the elements in a component must be compared to shot line locations and a component that is non-contiguous or physically separated may produce unpredictable or invalid results within COVART. Versions of FASTGEN prior to FASTGEN4 enforced this approach by requiring order dependent data. The FASTGEN4 format assumes the user will build the component in a compact form. Non-spatially compact components will increase the execution time of FASTGEN.

The CBULK file records are specified using fixed format lines containing ten fields with eight columns per field. Each field contains an input variable that is a real (R), integer (I), or character (C) data type. The first field in the first line of a record contains a character variable that identifies the record, and must be left justified. Some records require two lines to hold all the information. These records use the tenth column of the first line and the first column of the second line as continuation variables. The continuation variables in field ten and one must match in value and digit location. All other input variables will be either real, character, or integer variables, and must fall within the required field.

Input records that may appear in the target description file are VEHICLE, HOLE, WALL, SECTION, \$COMMENT, \$NAME, GRID, CBACKING, *CBAR*, CLINE,

CHEX1, CHEX2, CTRI, CQUAD, *CCONE1*, CCONE2, *CCONE3*, CSPHERE, COMPSPLT, CHGCOMP, SHIELD, and ENDDATA records.

All COMPSPLT, CBACKING, CHGCOMP, SHIELD, HOLE and WALL records should precede the first SECTION record. Each component starts with a SECTION record. GRID records must follow the SECTION record. The element records CBAR, CLINE, CHEX1, CHEX2, CTRI, CQUAD, CCONE1, CCONE2, CCONE3, and CSPHERE must follow GRID records.

Most graphics packages require unique grid point and element identification numbers throughout the bulk data file. FASTGEN only requires unique grid point and element identification numbers within a component. It is recommended that unique identification numbers be used throughout the bulk data file. \$COMMENT and \$NAME records may be placed anywhere in the bulk data file. *The ENDDATA record may only be used once. Any data following the ENDDATA record will be ignored.*

FASTGEN and NASTRAN

The FASTGEN bulk data file is derived from FASTGEN and NASTRAN input formatting requirements. Structural analysts have used NASTRAN since the early 1970s. NASTRAN input format is highly structured and flexible. This flexibility is difficult to support. FASTGEN is structured around the NASTRAN format, but several differences exist. A cross-reference of FASTGEN and NASTRAN elements is shown in Table B-1.

Table B-1. FASTGEN vs. NASTRAN Format

FASTGEN Record	NASTRAN Record
GRID	GRID
CBAR	CBAR
CLINE	CROD, CBAR, CBEAM
CTRI	CTRIA1, CTRIA2, CTRIA4
CQUAD	CQUAD1, CQUAD2, CQUAD4
CCONE1	None
CCONE2	None
CCONE3	None
CSPHERE	None
CHEX1	CHEXA1
CHEX2	CHEXA1

The basic difference between FASTGEN and finite element models (FEM), such as NASTRAN, is that FEM requires each structural element to be connected to another structural element, while FASTGEN does not permit connectivity between groups and/or components.

TARGET DATABASE KEYWORDS

This section provides a brief overview of the keywords available to the FASTGEN database developer during the target construction phase.

The FASTGEN database defines a target composed of complex geometric objects. These objects are modeled with a series of grid points and simple primitives called elements in three-dimensional space. Objects are defined by the user within global regions called groups and localized regions called components. *Each component must be initialized by the user with a SECTION record.*

SECTION Defines the beginning of a new component

A FASTGEN database has hierarchical, relational, and object-oriented features. Objects used to construct a FASTGEN target are classified as a group, a component, or a primitive. A group identification number defines a general class of components. A component identification number defines a specific object. Element identification numbers are used to define portions of a specific component. Grid point primitives are used to establish the location of each element in three-dimensional space. An example of this classification is an engine control unit panel. The engine is in the power plant group, the control unit is a component in the power plant group, and the panel is an element in the control unit component.

Grid point primitives are the basis of constructing elements; a Cartesian coordinate system is used.

GRID Defines the X, Y, and Z coordinates of a geometric point in Cartesian space

Elements define the surfaces of an object. Elements are defined by connecting a series of grid points. Supported elements in FASTGEN6 are:

CCONE2	Defines a thick wall cone/cylinder shaped element
CELBOW	Defines a torus segment used to join two CLINE or CCONE elements
CHEX1	Defines a thin wall hexahedron shaped element
CHEX2	Defines a solid hexahedron shaped element
CLINE	Defines a line-shaped element
CQUAD	Defines a quadrilateral-shaped element
CSPHERE	Defines a sphere-shaped object
CTRI	Defines a triangular-shaped object

As FASTGEN has evolved, it is natural that primitives that receive minimal use or are difficult to use will be phased out over time. The following primitives are still supported in FASTGEN6, but the user is strongly discouraged from using these elements to construct new target database files. The continued existence of these primitives supports

assessments of legacy targets that would require significant resources to update the target model.

- CBAR* Defines a reinforcing member typically associated with a structural component. Use of this element is discouraged.
- CCONE1* Defines a thin wall cone/cylinder shaped element. Use of this element is discouraged.
- CCONE3* Defines a compound thick wall cone/cylinder shaped element. Use of this element is discouraged.

As discussed previously, components may be modeled in plate mode or volume mode. The concept of an interference, or geometric overlap, was also discussed previously. A user-defined interference is an intentional intersection of a volume mode component with another component to introduce target model simplifications, such as a hydraulic line running through a fluid volume. Typically, FASTGEN will identify all geometric overlaps and output this information to its output files. The WALL record is a keyword at the disposal of the user to suppress generation of an intentional interference. **All WALL records must precede the first SECTION record in the target database. WALL records do not alter the LOS generated by the FASTGEN ray tracer, they merely suppress the generation of warnings regarding the interference of two components.**

WALL Defines the intersection of two or more components.

It is important for the user to distinguish between intentional and unintentional interferences, which are often produced by the CAD to FASTGEN geometry conversion process, or from other user modeling errors.

Volume subtraction is accomplished by defining a HOLE record associated with a volume mode component. All HOLE records must precede the first SECTION record in the target database.

HOLE Defines a hole within a surrounding volume. Subtracts a volume or plate from the outer volume.

For HOLE records, the surrounding component must be in volume mode, but the interfering components may be in either plate or volume mode. A typical example of a surrounding volume is a fuel tank. This tank could have bulkheads modeled as plates. The bulkhead component identification number would be identified on a WALL record. A large channel in this tank could be modeled using volume subtraction. The channel component identification number would be identified on a HOLE record.

In many cases, fuel tanks will be partially full at fuel levels specified for vulnerability assessments. Therefore, a fuel tank contains both fuel and ullage. A convenient method to model the ullage in the fuel tank is available through the use of the COMPSPLT record, which defines the fuel/ullage boundary inside the fuel tank.

COMPSPLT Defines the fuel level within a fuel tank.

All COMPSPLT records must precede the first SECTION record in the target database.

The CHGCOMP feature allows the user to alter the shot line trace history (LOS file) without explicitly changing the target description. It is up to the user to decide if the use of CHGCOMP is appropriate for their specific analysis.

CHGCOMP Modify the trace history of a shot line without changing the target description.

Internal documentation within the FASTGEN database is strongly encouraged. Documenting the database as it is being constructed greatly improves its maintainability. One or more \$COMMENT records should be used by the user to describe each component being modeled. The use of \$COMMENT records in the target file is optional.

\$COMMENT User-defined comments

The \$NAME record is a mechanism that allows the user to associate information with specific components. The \$NAME record is used by some target geometry viewers to allow display of the associated component information within the viewer. \$NAME records should contain a group and component identification number. The contents of a \$NAME record are usually application and target dependent. The use of \$NAME records in the target file is optional.

\$NAME Defines analysis specific information for a modeled aircraft component

The name of the target description is defined with the VEHICLE record, but only plays a role if COVART6 is executed in FASTGEN legacy mode. The use of the VEHICLE record in the target file is optional.

VEHICLE Defines the vehicle code

The SHIELD record can be placed in the target file by the user to define a component or range of components that act as a blast shield. Since air blast can be a significant damage mechanism, extreme caution should be exercised if this record is placed in the target database file.

SHIELD Defines a component or range of components that act as a blast shield.

The table below is provided as an example, showing the general structure and order dependencies of the records in a FASTGEN target file.

VEHICLE		
\$COMMENT		Optional. May appear anywhere in the target file.
HOLE		Optional. Must appear before first SECTION record.
...		
WALL		Optional. Must appear before first SECTION record.
...		
CHGCOMP		Optional. May appear anywhere in the target file.
...		
COMPSPLT		Optional. Must appear before first SECTION record.
...		
\$NAME		Optional.
\$COMMENT		Optional. May appear anywhere in the target file.
SECTION		Required. Must appear before GRID and element records for a given component.
	GRID	Required. Must occur after SECTION record and before element records for a given component.
	...	
	Elements	Required. Examples include CTRI, CQUAD, CLINE, etc. Must occur after GRID records for a given component.
	...	
\$NAME		Optional.
\$COMMENT		Optional. May appear anywhere in the target file.
SECTION		Required. Must appear before GRID and element records for a given component.
	GRID	Required. Must occur after SECTION record and before element records for a given component.
	...	
	Elements	Required. Examples include CTRI, CQUAD, CLINE, etc. Must occur after GRID records for a given component.
	...	
ENDDATA		Required. Must be the last record in the target file.

TARGET MODELING LIMITATIONS

Internal FASTGEN array sizes limit a single target description to 50 groups per target description and 999 components per group. In contrast to past versions of FASTGEN, there are no explicit limits on the number of grid points or elements per component. FASTGEN uses dynamic array allocation to reserve required space as needed. There are implicit limits that exist because of the eight-character field width characteristic of FASTGEN and COVART database files. It is possible to alter FASTGEN parameter statements in the source code to increase these static array sizes, but the new user is strongly discouraged from doing so. Unguided changes to these parameters may cause FASTGEN to produce inconsistent results.

TARGET INPUT RECORDS

INTRODUCTION

This section provides a detailed description of each target input file record available to the FASTGEN target model builder. Input record names are listed in alphabetical order to make it easier for the model builder to find record names in an efficient manner.

In FASTGEN prior to version 6.0, the target file was required to be named CBULK. In FASTGEN6 the names of the ASCII and binary (voxelized) target are user defined and input on the COVART6 MASTER file.

INPUT DATA RECORD: *CBACKING*

Description: *Defines backing component occurrence.*

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
<i>CBACKING</i>	<i>igrp1</i>	<i>icmp1</i>	<i>igrp2</i>	<i>icmp2</i>	<i>rbptn</i>	<i>rbppo</i>	<i>ibmpt</i>		
<i>CBACKING</i>	<i>0</i>	<i>123</i>	<i>7</i>	<i>123</i>	<i>0.25</i>	<i>0.3</i>	<i>8</i>		
C	I	I	I	I	R	R	I		

<u>Parameters:</u>	<u>Units:</u>	<u>Description:</u>
<i>igroup1</i>	-----	<i>Group number of a component in the bulk file.</i>
<i>icmp1</i>	-----	<i>Component number of a component in the bulk file.</i>
<i>igroup2</i>	-----	<i>Group number of the backing component (not in the bulk file).</i>
<i>icmp2</i>	-----	<i>Component number of the backing component (not in the bulk file).</i>
<i>rbptn</i>	<i>Inches</i>	<i>Normal thickness of the backing component.</i>
<i>rbppo</i>	-----	<i>Probability of occurrence of the backing component.</i>
<i>ibmpt</i>	-----	<i>Material type of the backing component.</i>

REMARKS:

- 1. Used under special circumstances where the effects of increasing modeling fidelity is desired without spending the resources in explicitly modeling the additional components.*
- 2. Use only in conjunction with plate mode components.*
- 3. Randomly inserts a component behind a known component to capture the effect of a non-uniformly placed component.*
- 4. Intended to be used with older target descriptions.*

INPUT DATA RECORD: *CBAR*

Description: *Defines a reinforcing member element.*

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
<i>CBAR</i>	<i>eid</i>	<i>mid</i>	<i>g1</i>	<i>g2</i>	<i>igra</i>	<i>icoa</i>	<i>th</i>	<i>r1</i>	
<i>CBAR</i>	<i>55</i>	<i>14</i>	<i>991</i>	<i>992</i>	<i>1</i>	<i>099</i>	<i>0.04</i>	<i>3.0</i>	
C	I	I	I	I	I	I	R	R	

Parameters:

Units:

Description:

eid

Element identification number (eid>0).

mid

Material identification number (mid>0).

g1, g2

Grid point identification numbers (g1 and g2>0, no repeated grid identification numbers).

igra

Optional. Group number for component that element is associated with. See Remark 4. This is not the same as the component this element is part of.

icoa

Optional. Component number for component that element is associated with. See Remark 4. This is not the same as the component this element is part of.

th

Inches

Normal thickness (th>0.0, and th<=r1). Bar elements should be used in plate mode only.

r1

Inches

Half-width (r1>0.0).

REMARKS:

- 1. Element identification numbers must be unique with respect to all other element identification numbers within a component.*
- 2. The maximum difference between the highest and lowest element identification number is parameter ILOS within a component. ILOS elements are allowed within a component. The FORTRAN parameter, ILOS, is further explained in the Software Size Limits section of this manual. This element counts as one element.*
- 3. The material identification numbers for several materials are defined in the FASTGEN Materials Chart.*

4. *This element is intended to provide an approximate model of reinforcing members that may require much effort to treat exactly. Common cross sectional shapes for reinforcing members include L, U, T, and hat.*
5. *The input width determines the intersection, using the influence mode computation in the same way the CLINE intersections are modeled. LOS thickness is determined using the normal thickness and the obliquity of the adjacent component. The optional input of the associated component affects the exact modeling of the intersection with the bar. If the associated component is not specified, and the computed intersection is within one grid spacing of one of the adjacent components on the shotline, that intersection is moved to be contiguous with the adjacent component and given the same obliquity as the adjacent component. If the associated component is specified and one of the adjacent components on the shotline is the associated component, then the same procedure is used, with the bar placed contiguous to the associated component. If no associated component is specified and the bar is not near any component, or if the associated component is specified, but neither of the adjacent components on the shotline are the associated component, then the intersection computed using the CLINE technique is used. In any case, if the computed intersection places the bar component within a volume mode component, the bar will be moved to a location outside the volume mode component. Also, if possible, a bar component will not be placed as the first or last component on a shotline. If there is a plate mode component adjacent to a bar at the beginning or end of a shotline, the components will be moved so the plate mode component (typically aircraft skin) is on the outside.*

INPUT DATA RECORD: CCONE1

Description: *Defines a thin wall cone/cylinder shaped element.*

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
CCONE1	eid	mid	g1	g2			th	r1	c1
CCONE1	55	14	981	982			3.0	8.0	55
C	I	I	I	I			R	R	C

1	2	3	4	5	6	7	8	9	10
c2	r2	end1	end2						
55	6.5	2	1						
C	R	I	I						

<u>Parameters:</u>	<u>Units:</u>	<u>Description:</u>
<i>eid</i>		<i>Element identification number (eid>0).</i>
<i>mid</i>		<i>Material identification number (mid>0).</i>
<i>g1, g2</i>		<i>Grid point identification numbers (g1 and g2>0, no repeated grid identification numbers).</i>
<i>th</i>	<i>Inches</i>	<i>Normal wall thickness (th>0.0).</i>
<i>r1, r2</i>	<i>Inches</i>	<i>Radius at g1 and g2 (r1 and r2>=0.).</i>
<i>c1, c2</i>		<i>Continuation record flag c1 must equal the value and position (left justified) of c2. This flag allows the user to locate misplaced continuation records. Normally, c1 and c2 equal eid.</i>
<i>end1, end2</i>		<i>End plate closure condition at g1 and g2; 1 = open (if radius=0., end is open) [DEFAULT] 2 = closed.</i>

REMARKS:

1. Element identification numbers must be unique with respect to all other element identification numbers within a component.

2. The maximum difference between the highest and lowest element identification number is parameter ILOS within a component. ILOS elements are allowed within a component. The FORTRAN parameter, ILOS, is further explained in the Software Size Limits section of this manual. This element counts as one element.
3. The thickness is assumed to be small, i.e., less than 2 inches. An element thicker than 2 inches should be modeled with a CCONE2 element.
4. Continuation record must follow parent.
5. The material identification numbers for several materials are defined in the FASTGEN Materials Chart.
6. The normal wall thickness is measured perpendicular from the cone wall, which in general, is not parallel to the cone's radial direction. For a cylinder, the normal wall thickness and the wall height are equal. For a cone, the normal wall thickness and the wall height are different. See Figure B-2 for a more descriptive illustration.

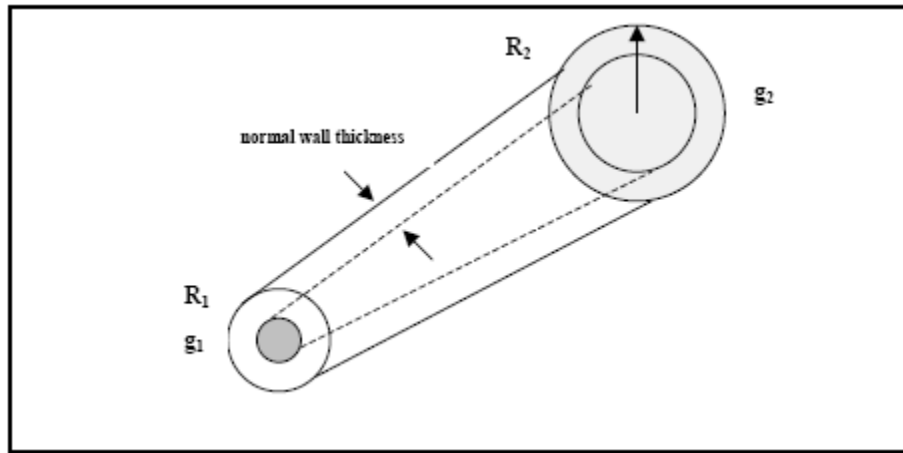


Figure B-2. *CCONE1*

7. End plate thickness is equal to the normal wall thickness. For very small cone lengths, the end plate thickness is set equal to the cone length if one end plate is specified, or set equal to the half the cone length if both end plates are active.
8. Special care should be taken when multiple CCONE1 elements are used in sequence to construct a complex object to insure that the cone ends match up correctly and that interferences are not generated.
9. When building complex objects without end plates, use of the CCONE2 element is recommended.
10. Figure B-3 is a schematic of a complex object composed of 3 CCONE1 elements. The first two elements (A and C) are identical with specified outer radii and thickness t_a . Element B is a transition CCONE1 element between A and C. The normal wall

thickness for element B (t_b) is different from the thickness of elements A and C. Each of these thicknesses is designated with red arrows.

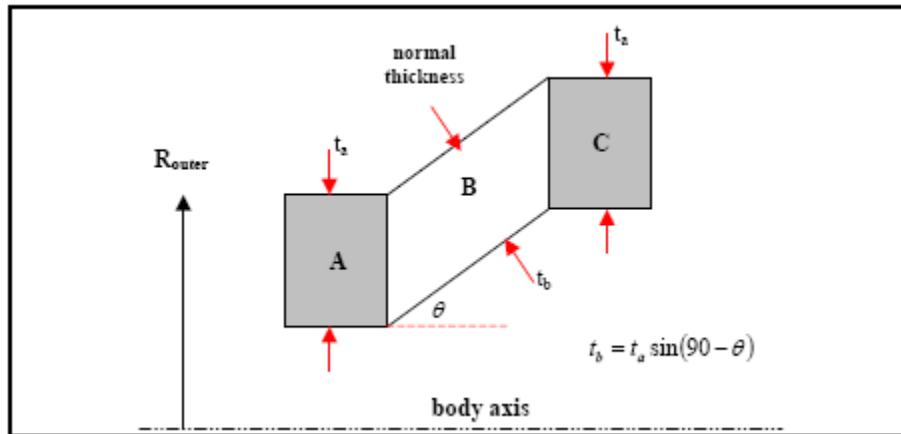


Figure B-3. *CCONE1 Normal Wall Thicknesses*

INPUT DATA RECORD: CCONE2

Description: Defines a thick wall cone/cylinder shaped element.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
CCONE2	eid	mid	g1	g2				ro1	c1
CCONE2	55	14	981	982				5.0	55
C	I	I	I	I				R	C

1	2	3	4	5	6	7	8	9	10
c2	ro2	ri1	r12						
55	10.0	3.0	7.0						
C	R	R	R						

<u>Parameters:</u>	<u>Units:</u>	<u>Description:</u>
eid		Element identification number (eid>0).
mid		Material identification number (mid>0).
g1,g2		Grid point identification numbers (g1 and g2>0, no repeated grid identification numbers).
ro1,ro2	Inches	Outer normal radius at g1 and g2 (ro1 and ro2>=0.0).
ri1,ri2	Inches	Inner normal radius at g1 and g2 (ri1 and ri2>=0.0).
c1,c2		Continuation record flag c1 must equal the value and position (left justified) of c2. This flag allows the user to locate misplaced continuation records. Normally, c1 and c2 equal eid.

REMARKS:

1. Element identification numbers must be unique with respect to all other element identification numbers within a component.
2. The maximum difference between the highest and lowest element identification number is parameter ILOS within a component. ILOS elements are allowed within a component. The FORTRAN parameter, ILOS, is further explained in the Software Size Limits section of this manual. This element counts as one element.

3. Continuation record must follow parent.
4. The CCONE2 elements may only be used in volume mode components.
5. The material identification numbers for several materials are defined in the FASTGEN Materials Chart.

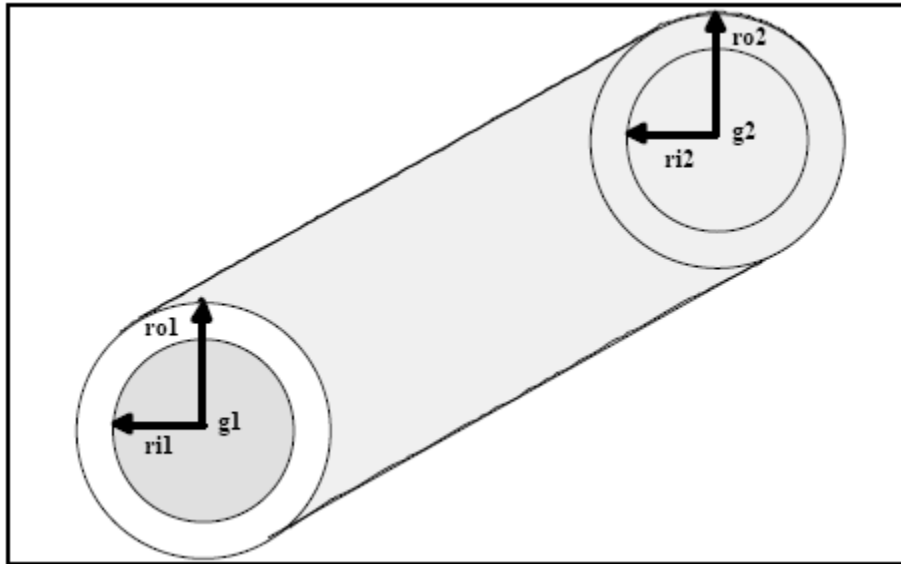


Figure B-4. CCONE2

INPUT DATA RECORD: CCONE3

Description: *Defines a compound thick wall cone/cylinder shaped element.*

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
<i>CCONE3</i>	<i>eid</i>	<i>mid</i>	<i>g1</i>	<i>g2</i>	<i>g3</i>	<i>g4</i>			<i>c1</i>
<i>CCONE3</i>	<i>55</i>	<i>14</i>	<i>981</i>	<i>982</i>	<i>983</i>	<i>984</i>			<i>55</i>
C	I	I	I	I	I	I			C

1	2	3	4	5	6	7	8	9	10
<i>c2</i>	<i>ro1</i>	<i>ro2</i>	<i>ro3</i>	<i>ro4</i>	<i>ri1</i>	<i>ri2</i>	<i>ri3</i>	<i>ri4</i>	
<i>55</i>	<i>10.0</i>	<i>3.0</i>	<i>7.0</i>	<i>8.0</i>	<i>5.0</i>	<i>2.0</i>	<i>3.0</i>	<i>4.0</i>	
C	R	R	R	R	R	R	R	R	

Parameters:

Units:

Description:

eid

Element identification number (eid>0).

mid

Material identification number (mid>0).

g1, g2, g3, g4

Grid point identification numbers (g1, g2, g3, and g4>0, no repeated grid identification numbers).

ro1, ro2, ro3, ro4

Inches

Outer normal radius at g1, g2, g3, g4 (ro1, ro2, ro3, and ro4>=0.0)

ri1, ri2, ri3, ri4

Inches

Inner normal radius at g1, g2, g3, g4 (ri1, ri2, ri3, and ri4>=0.0)

c1, c2

Continuation record flag c1 must equal the value and position (left justified) of c2. This flag allows the user to locate misplaced continuation records. Normally, c1 and c2 equal eid.

REMARKS:

- 1. Element identification numbers must be unique with respect to all other element identification numbers within a component.*
- 2. The maximum difference between the highest and lowest element identification number is parameter ILOS within a component. ILOS elements are allowed within a component. The FORTRAN parameter, ILOS, is further explained in the Software Size Limits section of this manual. This element counts as one element.*

3. Continuation record must follow parent.
4. The *CCONE3* element may only be used in volume mode components.
5. The material identification numbers for several materials are defined in the *FASTGEN Materials Chart*.
6. The *CCONE3* element is treated internally as 3 *CCONE2* type elements.
7. It is recommended that the grids be input so that points 1 and 4 represent the ends and points 2 and 3 are sequential between them. If this is not done, *FASTGEN* will reorder the points and associated radii so that this is the case, and it will be more difficult for the user to understand the target description. The discussion in the following Remarks assumes that the grid points are ordered in this way.
8. Points 1 and 2 may coincide and points 3 and 4 may coincide. Points 2 and 3 may coincide as long as points 1 and 4 are distinct. A warning message is output if 3 of the 4 points coincide.
9. It is not necessary to define all of *ro1*, *ro2*, *ro3*, *ro4*, *ri1*, *ri2*, *ri3*, and *ri4*. At least one radius must be defined at points 1 and 4; if only one radius is defined, that radius is used for both the inner and outer radius at that end. If *ro2* is not defined, its value is interpolated between *ro1* and *ro3* or *ro4* (*ro4* is used if *ro3* is not defined). If *ro3* is not defined, its value is interpolated between *ro2* or *ro1* (*ro1* is used if *ro2* is not defined) and *ro4*. An analogous process is used if *ri2* or *ri3* is not defined. If two points coincide, it doesn't matter which point a radius is assigned to. For example, if points 2 and 3 coincide, *ri2* and *ro3* could be defined, or *ro2* and *ri3* could be defined; the result would be the same. If different values are read in for the inner or outer radii at coincident grids, the result is undefined and no error is reported.

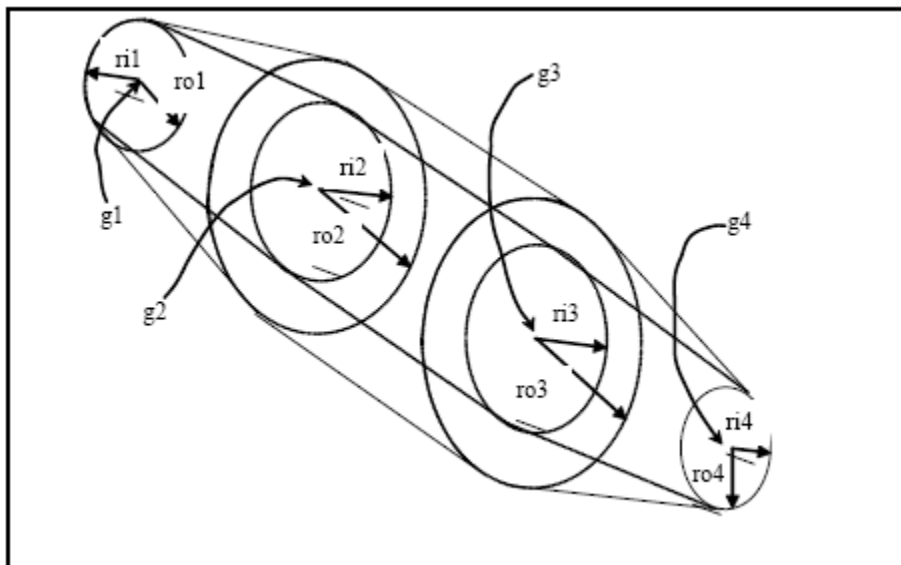


Figure B-5. *CCONE3* Nomenclature

INPUT DATA RECORD: CELBOW

Description: Defines a torus segment used to join two CLINE or CCONE elements.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
CELBOW	eid	mid	g1	g2	g3	g4	Th	R1	
CELBOW	55	14	991	992	993	994	0.125	1.0	
C	I	I	I	I	I	I	R	R	

Parameters:

Units:

Description:

eid

Element identification number (eid>0).

mid

Material identification number (mid>0).

g1,g2, g3, g4

Grid point identification numbers (g1 through g4>0, no repeated identification numbers).

Th

Inches

Torus tube thickness (th≥0.0)

R1

Inches

Torus tube radius

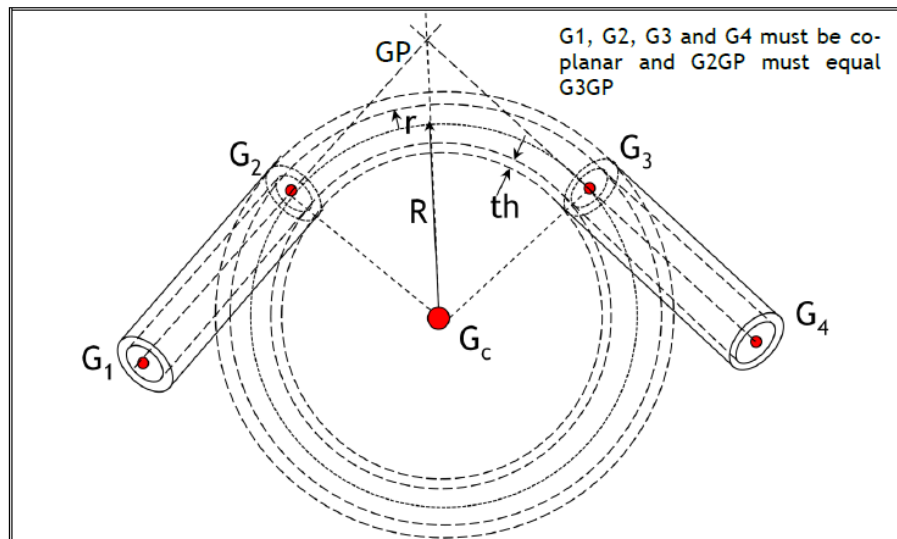


Figure B-6. CELBOW.

REMARKS:

1. CELBOW elements are volume mode elements.

2. Element identification numbers must be unique with respect to all other element identification numbers within a component.
3. The maximum difference between the highest and lowest element identification number is parameter ILOS within a component. ILOW elements are allowed within a component. The FORTRAN parameter, ILOS, is further explained in the Software Size Limits section of this manual.
4. Grid points g1 through g4 must be ordered such that g1 and g2 are the defining grid points of one of the CLINE/CCONE elements to be connected with a CELBOW and g3 and g4 are the defining grid points for the opposite CLINE/CCONE. The torus segment will be constructed to connect g2 and g3.
5. Grid points g1, g2, g3, and g4 must be co-planar or a gap will exist between the CELBOW and one of the CLINE/CCONE elements to be joined. Furthermore, the space between the two CLINE/CCONE elements to be joined by the CELBOW must be symmetric (i.e., the intersection point of the CLINE/CCONE axes must be equidistant from each of the CELBOW endpoints, g2 and g3).
6. If the CELBOW thickness, Th, is zero, the CELBOW is assumed to be a solid and to have closed ends. If Th is greater than zero, the CELBOW is assumed to be open inside the wall thickness and have open ends. For open CELBOW elements (Th > 0.0), the tube radius, R1 is the inside tube radius and R1 + Th is the outside tube radius.
7. If the CLINE/CCONE elements to be connected are parallel, the CELBOW element will be a half-torus in which case G₂, G_c, and G₃ will be co-linear with the torus in the plane defined by G₂, G_c, and G₃.

INPUT DATA RECORD: CHEX 1

Description: Defines a thin wall hexahedron shaped element.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
CHEX1	eid	mid	g1	g2	g3	g4	g5	g6	c1
CHEX1	55	14	991	992	993	994	996	997	6
C	I	I	I	I	I	I	I	I	C

1	2	3	4	5	6	7	8	9	10
c2	g7	g8					th	pos	
6	998	999					0.05	1	
C	I	I					R	I	

<u>Parameters:</u>	<u>Units:</u>	<u>Description:</u>
eid		Element identification number (eid>0).
mid		Material identification number (mid>0).
g1,g2, g3, g4 g5, g6, g7, g8		Grid point identification numbers (g1 through g8>0, no repeated identification numbers).
th	Inches	Normal thickness (th>0.0)
pos		Grid point position relative to normal thickness. 1 = center 2 = front face [DEFAULT]
c1,c2		Continuation record flag c1 must equal the value and position (left justified) of c2. This flag allows the user to locate misplaced continuation records. Normally, c1 and c2 equal eid.

REMARKS:

1. Element identification numbers must be unique with respect to all other element identification numbers within a component.
2. The maximum difference between the highest and lowest element identification number is parameter ILOS within a component. ILOS elements are allowed within a

component. The FORTRAN parameter, ILOS, is further explained in the Software Size Limits section of this manual. This element counts as four elements.

3. Grid points g1 through g8 must be ordered around the perimeter of the element. The direction of order (clockwise or counterclockwise) is not specified. Grid points g5 through g8 must be ordered in the same direction as grid points g1 through g4.
4. Continuation record must follow parent.
5. The software divides a CHEX element into twelve triangles. Grid points g1, g2, and g5 are used to define the first triangle, while g2, g6, and g5 are used for the second triangle and so on until each triangle is defined.

Table B-2. CHEX1 Triangles and Associated Grid Points

Triangles	Grid Identification Numbers		
1	g1	g2	g5
2	g2	g6	g5
3	g2	g3	g6
4	g3	g7	g6
5	g3	g4	g7
6	g4	g8	g7
7	g4	g1	g8
8	g1	g5	g8
9	g5	g7	g8
10	g5	g6	g7
11	g1	g2	g3
12	g1	g3	g4

6. When a shotline enters and exits the CHEX element no additional triangles are analyzed.
7. If two or more points are positioned at the same location, the triangles will degenerate into a line or a point. The CHEX element is sufficiently general to properly account for degenerated triangles.
8. The center of an element is well defined. The front face of an element is orientation dependent. The front face option is supported to simplify the translation of FASTGEN 3 target descriptions into FASTGEN 4 target descriptions.
9. The material identification numbers for several materials are defined in the FASTGEN Materials Chart.

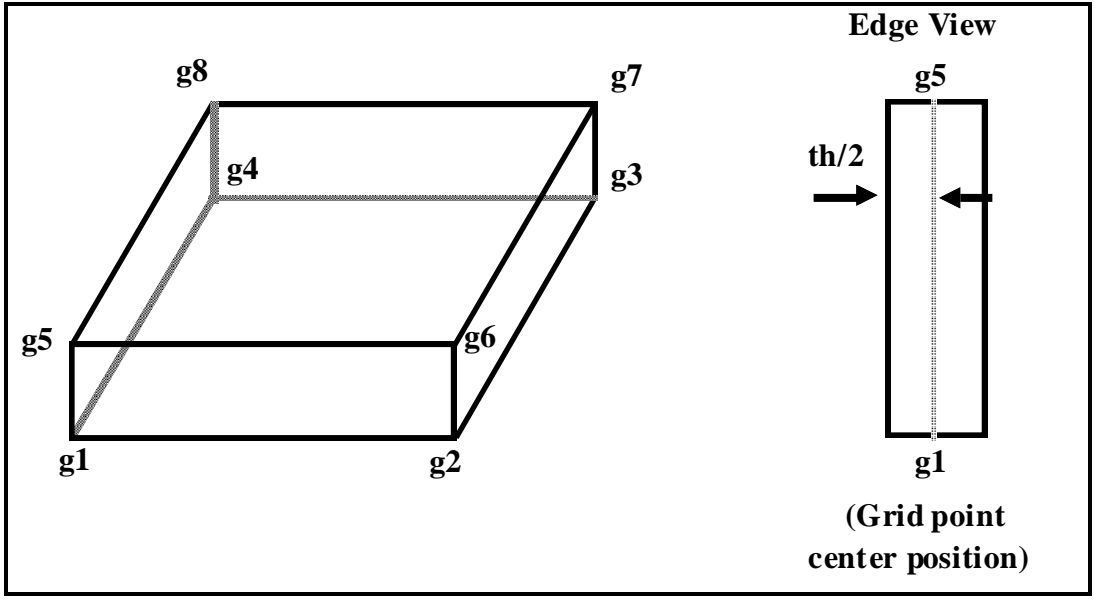


Figure B-7. CHEX1

INPUT DATA RECORD: CHEX2

Description: Defines a solid hexahedron shaped element.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
CHEX2	eid	mid	g1	g2	g3	g4	g5	g6	c1
CHEX2	55	14	991	992	993	994	996	997	6
C	I	I	I	I	I	I	I	I	C

1	2	3	4	5	6	7	8	9	10
c2	g7	g8							
6	998	999							
C	I	I							

Parameters:

Units:

Description:

eid

Element identification number (eid>0).

mid

Material identification number (mid>0).

g1,g2, g3, g4
g5, g6, g7, g8

Grid point identification numbers (g1 through g8>0, no repeated identification numbers).

c1,c2

Continuation record flag c1 must equal the value and position (left justified) of c2. This flag allows the user to locate misplaced continuation records. Normally, c1 and c2 equal eid.

REMARKS:

1. Element identification numbers must be unique with respect to all other element identification numbers within a component.
2. The maximum difference between the highest and lowest element identification number is parameter ILOS within a component. ILOS elements are allowed within a component. The FORTRAN parameter, ILOS, is further explained in the Software Size Limits section of this manual. This element counts as four elements.
3. Grid points g1 through g8 must be ordered around the perimeter of the element. The direction of order (clockwise or counterclockwise) is not specified. Grid points g5 through g8 must be ordered in the same direction as grid points g1 through g4.

4. Continuation record must follow parent.
5. The software divides a CHEX element into twelve triangles. Grid points g1, g2, and g5 are used to define the first triangle, while g2, g6, and g5 are used for the second triangle and so on until each triangle is defined.

Table B-3. CHEX2 Triangles and Associated Grid Points

Triangles	Grid Identification Numbers		
1	g1	g2	g5
2	g2	g6	g5
3	g2	g3	g6
4	g3	g7	g6
5	g3	g4	g7
6	g4	g8	g7
7	g4	g1	g8
8	g1	g5	g8
9	g5	g7	g8
10	g5	g6	g7
11	g1	g2	g3
12	g1	g3	g4

6. The CHEX2 element may only be used in volume mode components.
7. The material identification numbers for several materials are defined in the FASTGEN Materials Chart.

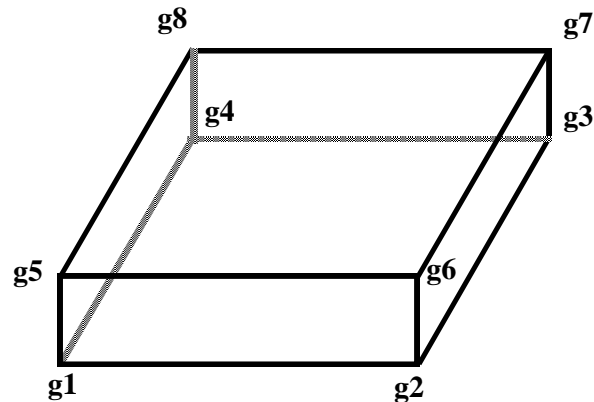


Figure B-8. CHEX2

INPUT DATA RECORD: CLINE

Description: Defines a line (rod) shaped element.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
CLINE	eid	mid	g1	g2			th	r1	
CLINE	55	14	991	992			0.04	3.0	
C	I	I	I	I			R	R	

<u>Parameters:</u>	<u>Units:</u>	<u>Description:</u>
eid		Element identification number (eid>0).
mid		Material identification number (mid>0).
g1, g2		Grid point identification numbers (g1 and g2>0, no repeated grid identification numbers).
th	Inches	Normal thickness (th>=0.0 and th<=r1). If th equals 0.0, then the element must be in a volume mode component.s
r1	Inches	Normal radius (r1>0.0)

REMARKS:

1. Element identification numbers must be unique with respect to all other element identification numbers within a component.
2. The maximum difference between the highest and lowest element identification number is parameter ILOS within a component. ILOS elements are allowed within a component. The FORTRAN parameter, ILOS, is further explained in the Software Size Limits section of this manual. This element counts as one element.
3. The radius is assumed to be small, i.e., less than 2 inches.
4. The zero thickness volume mode option is supported to simplify the translation of FASTGEN 3 target descriptions into FASTGEN 4 target descriptions. New models should use volume mode components when modeling solid CLINE elements.
5. The material identification numbers for several materials are defined in the FASTGEN Materials Chart.

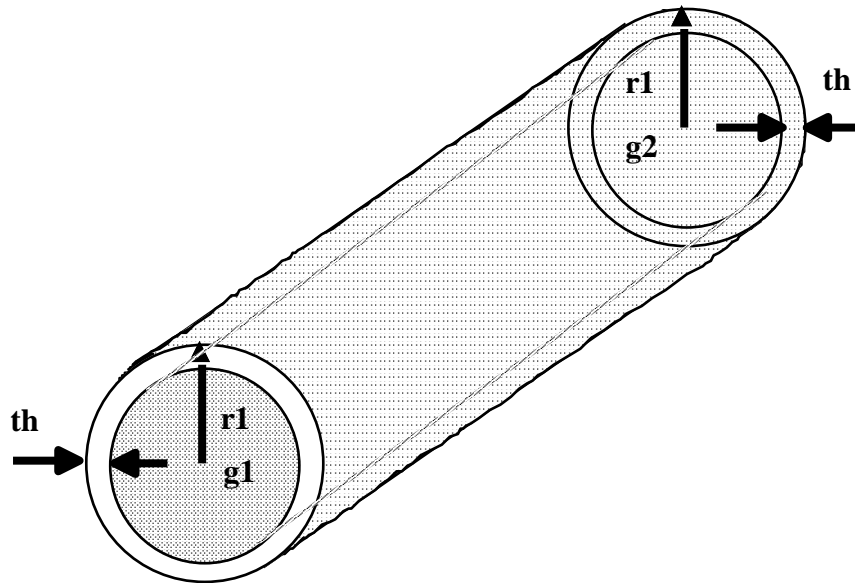


Figure B-9. CLINE Geometry Definitions

INPUT DATA RECORD: COMPSPLT

Description: Defines a component to be split by a plane perpendicular to the z-axis.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
COMPSPLT	gr	co	gr1	col	z				
COMPSPLT	4	105	4	205	7.5				
C	I	I	I	I	R				

Parameters:

Units:

Description:

gr		Group number of the component in the target description that is to be split.
co		Component identification number of the component in the target description that is to be split.
gr1		Group number of the new component (upper part).
col		Component number of the new component (upper part).
z	Inches	The z-coordinate of the plane which splits the component into two parts. This plane is parallel to the x-y plane in the target coordinate system.

REMARKS:

1. This record is read in with HOLE and WALL records in the CBULK file.
2. This input is intended to allow easy separation of an entire fuel tank into a fuel region and an ullage region with separate component numbers.

INPUT DATA RECORD: CQUAD

Description: Defines a quadrilateral shaped element.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
CQUAD	eid	mid	g1	g2	g3	g4	th	pos	
CQUAD	55	14	991	992	993	994	0.04	1	
C	I	I	I	I	I	I	R	I	

Parameters:

Units:

Description:

eid

Element identification number (eid>0).

mid

Material identification number (mid>0).

g1, g2, g3, g4

Grid point identification numbers (g1 through g4>0, no repeated grid identification numbers).

th

Inches

Normal thickness (th>0.0).

pos

Grid point position relative to normal thickness.

1 = center

2 = front face [DEFAULT]

REMARKS:

1. Element identification numbers must be unique with respect to all other element identification numbers within a component.
2. The maximum difference between the highest and lowest element identification number is the parameter ILOS within a component. ILOS elements are allowed within a component. The FORTRAN parameter, ILOS, is further explained in the Software Size Limits section of this manual. This element counts as one element.
3. Grid points g1 through g4 must be ordered around the perimeter of the element. The direction of order (clockwise or counterclockwise) is not specified.
4. FASTGEN divides a CQUAD element into two triangle elements. Grid points g1, g2, and g3 are used for the first triangle element, while g1, g3, and g4 are used for the second triangle element.

Table B-4. CQUAD Triangles and Associated Grid Points

Triangles	Grid Identification Numbers		
1	g1	g2	g5
2	g1	g3	g4

5. If two or more points are positioned at the same location the triangles will degenerate to a line or a point. The CQUAD element is omitted when the element degenerates to a line or a point. Any CQUAD element with a thickness greater than 2 inches should be modeled with a CHEX2 element.
6. The CQUAD is assumed to be nearly planar. For any given shotline, only one triangle (of the two triangles that defines the quadrilateral) will be hit.
7. The center of an element is well defined. The front face of an element is orientation dependent. The front face option is supported to simplify the translation of FASTGEN 3 target descriptions into FASTGEN 4 target descriptions. New models should use the center option.
8. The material identification numbers for several materials are defined in the FASTGEN Materials Chart.

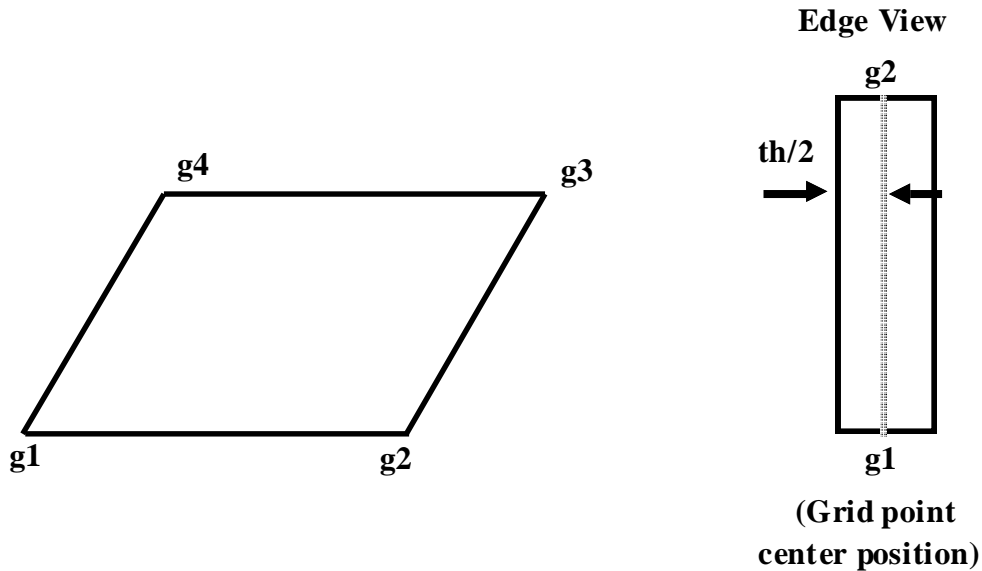


Figure B-10. CQUAD Geometry Definitions

INPUT DATA RECORD: CSPHERE

Description: Defines a sphere shaped element.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
CSPHERE	eid	mid	g1				th	r1	
CSPHERE	55	14	981				0.125	8.2	
C	I	I	I				R	R	

<u>Parameters:</u>	<u>Units:</u>	<u>Description:</u>
eid		Element identification number (eid>0).
mid		Material identification number (mid>0).
g1		Grid point identification number (g1>0).
th	Inches	Normal thickness (th>0.0).
r1		Normal radius (r1>0.0).

REMARKS:

1. Element identification numbers must be unique with respect to all other element identification numbers within a component.
2. The maximum difference between the highest and lowest element identification number is the parameter ILOS within a component. ILOS elements are allowed within a component. The FORTRAN parameter, ILOS, is further explained in the Software Size Limits section of this manual. This element counts as one element.
3. The material identification numbers for several materials are defined in the FASTGEN Materials Chart.

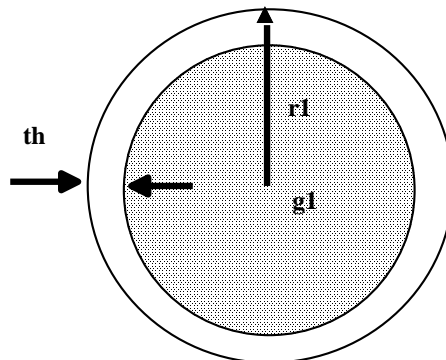


Figure B-11. CSPHERE

INPUT DATA RECORD: CTRI

Description: Defines a triangular shaped element.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
CTRI	eid	mid	g1	g2	g3		th	pos	
CTRI	55	14	991	992	993		0.03	1	
C	I	I	I	I	I		R	I	

<u>Parameters:</u>	<u>Units:</u>	<u>Description:</u>
eid		Element identification number (eid>0).
mid		Material identification number (mid>0).
g1, g2, g3		Grid point identification numbers (g1 through g3>0, no repeated grid identification numbers).
th	Inches	Normal thickness (th>0.0).
pos		Grid point position relative to normal thickness. 1 = center 2 = front face [DEFAULT]

REMARKS:

1. Element identification numbers must be unique with respect to all other element identification numbers within a component.
2. The maximum difference between the highest and lowest element identification number is the parameter ILOS within a component. ILOS elements are allowed within a component. The FORTRAN parameter, ILOS, is further explained in the Software Size Limits section of this manual. This element counts as one element.
3. Grid points g1 through g3 must be ordered around the perimeter of the element. The direction of order (clockwise or counterclockwise) is not specified.
4. The thickness is assumed to be small, i.e., less than 2 inches. An element thicker than 2 inches should be modeled with a CHEX2 element.
5. The center of an element is well defined. The front face of an element is orientation dependent. The front face option is supported to simplify the translation of FASTGEN 3 target descriptions into FASTGEN 4 target descriptions. New models should use the center option.
6. The material identification numbers for several materials are defined in the FASTGEN Materials Chart.

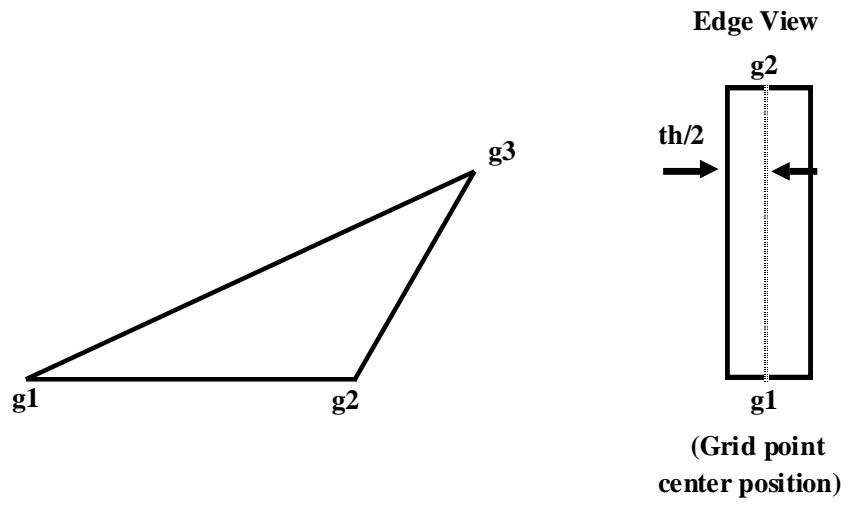


Figure B-12. CTRI Geometry Definitions

INPUT DATA RECORD: ENDDATA

Description: Defines the end of the Bulk Data File.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
ENDDATA									
ENDDATA									
C									

REMARKS:

1. Data following the ENDDATA record will not be used.
2. The ENDDATA record may be used one time.

INPUT DATA RECORD: GRID

Description: Defines the location of a 3-D geometric point.

Format, Example and Data Type:

1	2	3	4	5	6	7	8	9	10
GRID	g1		x	y	z				
GRID	6		10.25	10.25	-10.25				
C	I		R	R	R				

Parameters:

Units:

Description:

g1

Grid point identification number (g1>0).

x, y, z

Inches

Location of the grid point:

x increasing is toward the front

x decreasing is toward the aft/back

y increasing is toward the left from pilot's view point

y decreasing is toward the right from pilot's view point

z increasing is toward the top

z decreasing is toward the bottom

REMARKS:

1. All grid point identification numbers must be unique with respect to all other grid point identification numbers within a component.
2. The maximum difference between the highest and lowest grid identification number is parameter ILOS within a component. ILOS GRID records are allowed within a component. The FORTRAN parameter, ILOS, is further explained in the Software Size Limits section of this manual.
3. GRID records must follow each SECTION record. The GRID record directly after the SECTION record must have the smallest grid identification number of all GRID records within a component.
4. The value of the grid identification number is limited by the field size.
5. x, y, and z values must include a decimal point.

INPUT DATA RECORD: HOLE

Description: Defines a hole within a surrounding volume. This procedure subtracts a volume or a plate from the outer volume.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
HOLE	gr	co	gr1	co1	gr2	co2	gr3	co3	
HOLE	3	12	9	111	10	107	14	263	
C	I	I	I	I	I	I	I	I	

Parameters:

Units:

Description:

gr, co

Group and component identification number of the surrounding volume ($0 \leq gr \leq 9$, $1 \leq co \leq 999$).

gr1, co1,
gr2, co2,
gr3, co3

Group and component identification number of the subtracted components ($0 \leq gr \leq 9$, $1 \leq co \leq 999$).

REMARKS:

1. Surrounding components entered on the HOLE record must be in volume mode.
2. A surrounding component must totally enclose the interfering components.
3. An interfering component entered on the HOLE record may be either in a volume or a plate mode.
4. All HOLE records should precede the first SECTION record.
5. IHW records are allowed within the bulk data file. The FORTRAN parameter IHW is further explained in the Software Size Limits section of this manual.

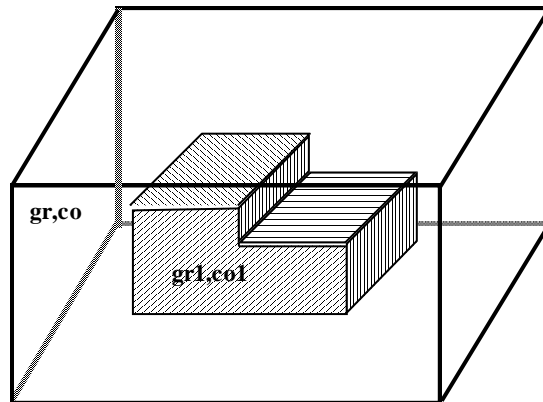


Figure B-13. HOLE Geometry Definitions

INPUT DATA RECORD: SECTION

Description: Defines the beginning of a new component.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
SECTION	gr	co	mo	sp					
SECTION	2	173	1	4					
C	I	I	I	I					

Parameters:

Units:

Description:

gr

Group identification number (0<=gr<=49). Table B-5 is a suggestion for component groupings for aircraft targets. (See remark 5 for older definitions)

TABLE B-5. Common Aircraft Organization by Group Number

Group No.	Components	Group No.	Components
0	skin forward fuselage	27	fuel center fuselage tank
1	skin forward intermediate fuselage	28	fuel aft intermediate fuselage tank
2	skin center fuselage	29	fuel aft fuselage tank
3	skin aft intermediate fuselage	30	fuel left wing tank
4	skin aft fuselage	31	fuel right wing tank
5	skin left wing	32	fuel lines
6	skin right wing	33	ammunition
7	skin horizontal tail	34	armament
8	skin vertical tail	35	structure forward fuselage
9-10	engine #1	36	structure forward intermediate fuselage
11-12	engine #2	37	structure center fuselage
13-14	engine #3	38	structure aft intermediate fuselage
15-16	engine #4	39	structure aft fuselage
17	crew	40	structure left wing
18	flight control pitch	41	structure right wing
19	flight control yaw	42	structure horizontal tail
20	flight control roll	43	structure vertical tail
21	hydraulic system #1	44	electrical boxes
22	hydraulic system #2	45	electrical lines
23	hydraulic system #3	46	electrical PAO lines
24	hydraulic system #4	47	electrical antenna/radar/ECM
25	fuel forward tank	48-49	miscellaneous
26	fuel forward intermediate fuselage tank		

co

Component identification number (1<=co<=999). Components are subsets of groups, i.e., a radar power

supply line replaceable unit (LRU) on the right wing is a component within the electrical group.

mo

Plate/volume mode:

1 = plate

2 = volume

sp

Space code (0<=sp<=5) [not required]

Aircraft

0 = redefinition component

1 = fuselage/engine pods

2 = cockpit

3 = interior of wings

4 = vertical fins/elevators

5 = exterior

REMARKS:

1. \$COMMENT and \$NAME records should follow each SECTION record. Descriptive keywords such as line replaceable unit (LRU) should be used throughout the database.
2. The SECTION record may be used one time per component.
3. Elements defined in volume mode components should have element thickness set equal to 0.0.
4. Space codes are not used within FASTGEN. Space codes entered into FASTGEN on the SECTION record are passed through the software without error checking or modification, except to denote the end of a shotline.
5. Group identification number (0<=gr<=9) for legacy FASTGEN target descriptions are as follows:

Aircraft

0 = skin

1 = power plant

2 = crew

3 = flight control system

4 = fuel system

5 = ammunition including bombs

6 = armament

7 = structural members

8 = electrical system/avionics

9 = miscellaneous

INPUT DATA RECORD: SHIELD

Description: Defines a component or range of components as a blast shield.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
SHIELD	icomp1	icomp2	rpress						
SHIELD	1005	1250	5000.						
C	I	I	R						

Parameters:

Units:

Description:

icomp1

First component number in range.

icomp2

Second component number in range.

rpress

Pressure limit for shield effectiveness. These components will completely shield the blast pressures below this limit.

REMARKS:

- Multiple SHIELD records can be used. The component number ranges cannot overlap the range in other SHIELD records.
- Setting the value of icomp2 is optional. If this record is omitted, the shield will only be for one component as defined in icomp1.
- Setting the value of rpress is optional. If the pressure is not defined, then the shield is considered to be 100% effective for any pressure.
- No blast rays will be generated through an intact shield.
- Therefore, the four following options are allowed:

```

SHIELD      1001      1006
SHIELD      1001
SHIELD      1001      1006      5000.
SHIELD      1001                          5000.
    
```

INPUT DATA RECORD: VEHICLE

Description: Defines the vehicle code.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
VEHICLE	code	date	iveh						
VEHICLE	B-2	9/2/92	65535						
C	C	C	C						

Parameters:

Units:

Description:

code

User defined vehicle code.

date

Date database was last updated.

iveh

Vehicle code number.

REMARKS:

1. The VEHICLE record may be used one time.
2. The vehicle code number is only used for output to a COVART-format burst point file. If no vehicle code number is entered, FASTGEN will calculate one.

INPUT DATA RECORD: WALL

Description: Defines the intentional intersection of two or more components.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
WALL	gr	co	gr1	co1	gr2	co2	gr3	co3	
WALL	9	123	9	124	3	311	8	097	
C	I	I	I	I	I	I	I	I	

Parameters:

Units:

Description:

gr, co

Group and component identification number of the surrounding volume ($0 \leq \text{gr} \leq 9$, $1 \leq \text{co} \leq 999$).

gr1, co1

gr2, co2

gr3, co3

Group and component identification number of the interfering component ($0 \leq \text{gr} \leq 9$, $1 \leq \text{co} \leq 999$).

REMARKS:

1. Surrounding components entered on the WALL record must be in the volume mode.
2. A surrounding component may have several interfering components.
3. An interfering component entered on the WALL record may be either in a volume or a plate mode.
4. All WALL records should precede the first SECTION record.
5. IHW records are allowed within the bulk data file. The FORTRAN parameter IHW is further explained in the Software Size Limits section of this manual.

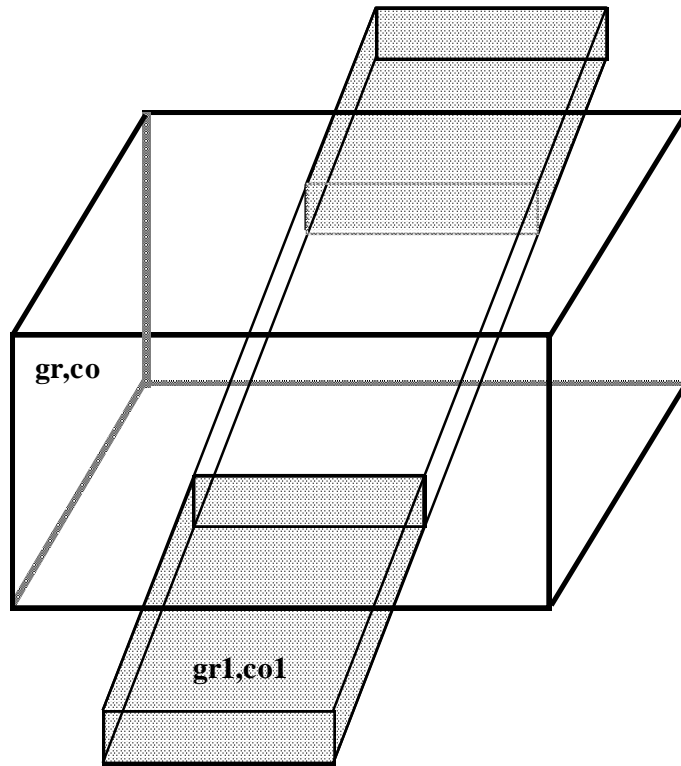


Figure B-14. WALL Geometry Definitions

INPUT DATA RECORD: \$COMMENT

Description: Defines user comments.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
\$COMMENT	text								
\$COMMENT	B-2 target model								
C	C								

Parameters: Units: Description:
text User defined text.

REMARKS:

1. See SECTION record for group names.
2. The \$COMMENT record may be used any number of times.

INPUT DATA RECORD: \$NAME

Description: Defines names and components.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
\$NAME	gr	co		wuc	lcn				text
\$NAME	0	172		11					LRU, flight control
C	I	I		I	C				C

Parameters:

Units:

Description:

gr

Group identification number (0<=gr<=9).

co

Component identification number (0<=co<=999).

wuc

Work Unit Code (WUC) number (11<=wuc<=99).

lcn

Logistics Control Number (LCN).

text

Descriptive name of components.

REMARKS:

1. Group and component identification numbers are defined on the SECTION record.
2. The \$NAME record should be used one time for each component.

SOFTWARE SIZE LIMITS

Table B-6 lists some of the current FORTRAN parameters for static array sizes.

TABLE B-6. FASTGEN Array Names, Sizes, and Descriptions

Array	Size	Description
ILOS	50000	Number of GRID records within a component.
IHW	40000	Number of HOLE and WALL records, each.

APPENDIX C

**CRITICAL INPUT DEVELOPMENT REFERENCE: EXCERPTS FROM
COVART 6.7 USER'S MANUAL**

This appendix contains excerpts from *COVART 6.7: User's Manual* and is intended to aid the user in developing the component properties file (JTYPE FILE), probability of hazard given hit file (PK FILE), and multiply vulnerable file (MV FILE). These excerpts are only for the data that is necessary to run UEDDAM.

COVART6 INPUT FILES

JTYPE FILE

INTRODUCTION

The purpose of the JTYPE file is to organize the components/regions of the target in a manner that facilitates the assignment of material properties and the assessment of P_K . To this end, the JTYPE file allows the user to do the following:

- Create NAMs – collections of components/regions for which the user would like to control material assignments or see in the output.
 - Assign physical properties (material code, density ratio, fluid properties for ignition) to components/regions in the target file
 - Define alias components (copies of the component/region that allow for singly and multiply vulnerable assignments)
 - Assign $P_{cd/h}$ tables to the component/region for each kill level

In COVART, the user assigns $P_{cd/h}$ relationships to individual components/regions, but COVART reports results in terms of NAMs the user creates in the JTYPE file. A NAM is a collection of components composing a function, a group of parts with the same material type, or kill mechanism for a target. The creation of NAM's helps the user to organize and simplify the output generated by COVART.

The JTYPE file provides information required for other features found in COVART. The MV file references the NAMs defined in the JTYPE file to create multiply-vulnerable group and system definitions. Furthermore, the NAM multiplier feature can also reference NAMs from the JTYPE file in applying $P_{K/d}$ corrections during an analysis. The JTYPE file, in turn, references the $P_{cd/h}$ tables the user defines in the $P_{cd/h}$ input file. The JTYPE file is required for every analysis.

DATA TYPES

The JTYPECD data type must be the first input line to ensure that the correct JTYPE file is processed. The program compares the TCODE parameter to CODEJ parameter on the JTYPEFILE data type in the MASTER file. If these character strings do not match, processing of the JTYPE file is terminated, but the program will continue input validation for the remaining files.

Data type COMPDEF defines constants for the JTYPE file: MAT, the default material code for components not listed in the JTYPE array; and LOCHK, an option flag for

indicating how vulnerable area storage locations, LO, values are assigned. LO values are either assigned by the program or are user supplied. IDOUBF is a flag which specifies whether to calculate P_{KS} for every hit on a component or just the first hit on the component along the shotline. This must be the second line of data.

If LOCHK is equal to zero the storage location value, LO, is assigned by the program.

The next five data types describe the components. This file does not have to necessarily contain all the components in the target description. Components not in the JTYPE file will be assumed to have the default material type, a relative density of 1.0, and be non-critical. The default choice for the default material type is usually Aluminum 2024 (Material Code 8).

Data type COMPPHY describes component characteristics, including component code number, output nomenclature, density, material code, and a flag for determining if the presented area is desired. One line for each component is required, and all COMPPHY data types must be input before any other data types are input.

The component number is stored in the JTYP1 array. The variables LO and NAM are used for storage of component vulnerable areas and redundant component calculations. NAM is an output label array and there is a one-to-one correspondence between it and the LO array. The output order of the component vulnerable area and $P_{K/H}$ results is determined by the order of components in the LO array.

The component density ratio/normal thickness (JD) parameter enables the user to change, for penetration purposes, the thickness input from the shotline file without changing the actual target description. This value is either the density times 100 or the plate thickness in hundredths of an inch entered as a negative value. For example, if the target skin was described as 0.05 inches thick, then the target skin could be changed to 0.10 inches thick by inputting -10 as the value for JD. For correctly described components that are solids, such as a structural beam, a value of 100 is input for JD. However, for correctly described components that are not homogeneous in composition, such as a gear box, a scaling factor with a decimal point implied between columns 30 and 31 can be input. For example, 50 could be input as a value for JD. When wanting to remove a NAM from an analysis, such as a piece of armor, the user can accomplish this by inputting a zero for JD. These components are sometimes referred to as “phantom” components. The material code must be input and must be a valid code. (Material codes are listed in Table C-1.)

TABLE C-1. Component Material Code Numbers

1	Steel (BHN = 100)	26	Doron
2	Steel (BHN = 150)	27	Bullet resistant glass
3	Steel (BHN = 200)	28	Hard Rubber
4	Steel (BHN = 250)	29	Soft Rubber
5	Steel (BHN = 300)*	30	Depleted Uranium
6	Steel (BHN = 350)	31	Steel (BHN = 550)
7	Titanium (BHN = 285)*	32	Steel (BHN = 600)

8	Aluminum 2024*	34	Phenolic
9	Aluminum 5083	35	Oak
10	Aluminum 5154	36	Pine
11	Aluminum 5356	37	Graphite/Epoxy
12	Aluminum 6061	38	BMI Composite
13	Aluminum 7075	45	Titanium (BHN = 180)
14	Aluminum 7039	[46]	[Composite (General)*]
15	Magnesium	[47]	[Inconel 625 LCF*]
16	Face Hardened Steel	51	Water
17	Cast Iron	52	Gasoline
18	Copper	53	Lubrication
19	Lead	54	JP1 Fuel
20	Tuballoy	55	JP4 Fuel
21	Unbonded Nylon	56	JP5 Fuel
22	Bonded Nylon	57	JP8 Fuel
23	Lexan	58	DF-2 (Diesel)
24	Cast Plexiglass	59	MIL-H-56056 (Hydraulic)
25	Stretched Plexiglass	60	MIL-H-83282 (Hydraulic)

* Predefined material types within the default UEDDAM MATERIAL FILE.

Any components stored in a single LO location are assumed to be singly vulnerable with respect to any other component stored in that location. For components that are part of a redundant set (i.e., are multiply vulnerable with respect to each other) the various parts of the redundant set must not be stored in the same LO location. For example, the pilot and copilot, if redundant, must be stored in different LO locations and assigned different NAM labels.

P_{cd/h}

Data type COMPPK allows the user to assign P_{cd/h} tables for up to 15 kill levels with up to six tables allowed per kill level. Tables capturing component damage at the most severe kill level (e.g., Attrition or KK-kill) are input in the KL1 field. Tables capturing component damage at the second kill level are input in the KL2 field. When inclusive kill levels are computed, a component vulnerable at A-Kill level is also vulnerable at the B-Kill level, and a P_{cd/h} table number should be recorded in the fields related to both kill levels (even if it is the same P_{cd/h} table).

For a prevent take-off/prevent mission (PTO/PM) study, tables related to catastrophic on ground (COG) kill should be input in the KL1 field. The KL2 field should hold the P_{cd/h} table assignment related to a PTO24 or PM24. The KL3 field should hold a table assignment that defines damage between PTO24/PM24 and PTO0/PM0. Finally the KL4 field should hold a table assignment defining damage for PTO0/PM0. It should be noted that the tables assigned for kill levels 2, 3, and 4 in this example do not describe the

probability of killing the target given a hit but do define the probability that the target requires at least a given time to repair the target (this includes COG damage).

JD ADJUSTMENTS IN COVART

In COVART, the line of sight calculated for any target intersection can be modified through the use of the density ratio. This parameter, JD, can be input by the user for every target component/region using the COMPPHY record in the JTYPE file. JD has different meanings depending on the sign of its value. When positive, this parameter is either the ratio of material densities or a scaling factor to account for non-homogeneous solids. When negative, the absolute value of this parameter is the normal thickness of the entity in hundredths of an inch.

Input Data Type: COMPDEF

Description: Defines constants for JTYPE file and default material for all components not defined in the JTYPE array.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
COMPDEF	MAT	LOCHK	IDOUBF	RAD					
COMPDEF	2	0	1	1.0					
C	I	I	I	R					

<u>Parameters</u>	<u>Units</u>	<u>Description</u>
MAT	-----	Material code for components not listed in JTYPE array. If zero is input, default value is 8 for Aluminum 2024.
LOCHK	-----	Option flag which indicates if vulnerable area storage locations (LO) values are computed or input 0 assigned by COVART (Set to 0 for UEDDAM runs) 1 user input
IDOUBF	-----	Option flag which selects whether to calculate P _K S for every hit on a component or just the first hit 0 use the variable IDOUB on data record COMPPHY to determine how double hits are treated on a component-by-component basis 1 calculate P _K S for every hit on a component
RAD	-----	Influence mode radius (Not relevant for UEDDAM).

REMARKS:

1. The COMPDEF data type is required to appear in the JTYPE input database. If this record is omitted, COVART will report an error and stop.
2. The COMPDEF data type must be the second record that appears in the JTYPE file, following the JTYPECD input record.
3. Any data that follows the ENDDATA data type in the JTYPE file will not be processed by COVART.
4. IDOUBF equal one indicates that every impact on a component with Pcd/h table assigned will result in COVART calculating a Pcd/h for the intersection. This mimics the behavior found previously in COVART II and COVART 3.0. Setting this value

equal to zero allows the user to control, at the component level, whether COVART calculates a $P_{cd/h}$ for every intersection or just the first intersection on a particular component.

Input Data Type: COMPPHY

Description: Defines JTYPE physical component input data.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
COMPPHY	KTYP1	NAM	JA	JD	JM	IDOUB	AZAXIS	ELAXIS	
COMPPHY	2005	CABLE	1	100	5	1	90.	0.	
C	I	C	I	I	I	I	R	R	

<u>Parameters</u>	<u>Units</u>	<u>Description</u>
KTYP1	-----	Component code number. KTYP1 is stored in the JTYP1 array. Valid values for KTYP1 are: $1 \leq \text{KTYP1} \leq 99999$.
NAM	-----	Eight character name used in printed output. (cannot include the characters “.”, “/”, “-“, or “+”). Names for components included in multiply vulnerable groups may not include asterisks “*” or spaces “ ” either.
JA	-----	Not used in UEDDAM. Set to 1 for UEDDAM runs.
JD	-----	Component density or normal thickness. If the value is negative, then JD contains the normal plate thickness. Density is a percentage; normal thickness is in hundredths of an inch.
JM	-----	Material code; if the value is zero, the default material code specified by MAT on the COMPDEF data type is used. Refer to Table C-1 for list of valid plate material codes.
IDOUB	-----	Multi-hit flag; if IDOUB > 0, P_k will be calculated for every hit of this component. If blank or zero, a P_k will be calculated for only the first hit. IDOUB is used only if IDOUBF on data type COMPDEF is zero.
AZAXIS	degrees	Azimuth angle for the component orientation. AZAXIS is only needed for components which are assigned an axial P_k table type. This input is only relevant if axial P_k tables are being used. <u>Note:</u> An entry of 0.0 is interpreted differently than a blank entry. See Remark 7 below.
ELAXIS	degrees	Elevation angle for the component orientation. ELAXIS is

only needed for components which are assigned an axial P_K table type. This input is only relevant if axial P_K tables are being used. Note: An entry of 0.0 is interpreted differently than a blank entry. See Remark 7 below.

REMARKS:

1. The COMPPHY data type is required to appear in the JTYPE input database. If this record is omitted, COVART will report an error and stop.
2. COMPPHY data types must appear in the JTYPE file after the JTYPECD and COMPDEF data types.
3. In order to run the same type of methodology as COVART II and COVART 3.0, set IDOUBF on input record COMPDEF to one.
4. The azimuth and elevation angle for the component defined zero degree angle used in the AXIAL2 P_K tables.
5. “Phantom” components, if required, should be assigned zero density (JD).
6. It is recommended that values for AZAXIS and ELAXIS only be entered for components which are assigned AXIAL2 P_K tables, and that AZAXIS and ELAXIS be left blank for all other components. This practice will conserve space in the internal array used to store data for use with the axial P_K tables, and will also reduce the number of computations required for each ray in an HE or HEI evaluation.
7. There is a difference between an explicit entry of 0.0 and a blank entry for AXAXIS and ELAXIS. If both AZAXIS and ELAXIS are left blank, no azimuth and elevation are stored for the component. This is an error if the component is assigned an AXIAL2 P_K table type. If an explicit value (even 0.) is entered for AZAXIS and ELAXIS, then an azimuth and elevation will be stored for the component.

Input Data Type: COMPPK

Description: Defines JTYPE P_{cd/h} table references.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
COMPPK	KTYP1	KL1	KL2	KL3	KL4	KL5	KL6	KL7	C1
COMPPK	2005	0	0	12	5	6	7	12	A1
C	I	I	I	I	I	I	I	I	C

C2	KL8	KL9	KL10	KL11	KL12	KL13	KL14	KL15	
A2	12	5	0	0	12	6	7	5	
C	I	I	I	I	I	I	I	I	

<u>Parameters</u>	<u>Units</u>	<u>Description</u>
KTYP1	-----	Component code number KTYP1 is stored in the JTYP1 array. Valid values for KTYP1 are: 1≤KTYP1≤99999.
KL _i	-----	P _{cd/h} table number for kill level i
C1, C2	-----	Continuation line flag, C1, must be identical to the character string and position (left-justified) of C2

REMARKS:

- The term “kill level” represents flight phases or hazard levels for a UEDDAM analysis.**
- The COMPPK data type is required to appear in the JTYPE input database. If this record is omitted, COVAT will report an error and stop.
- COMPPK data types must appear in the JTYPE file after the JTYPECD, COMPDEF, and COMPPHY data types.
- The COMPPK data type is only required for vulnerable components. A component that is not a vulnerable component functions as shielding.
- For the exclusive kill methodology, COVART requires that the first and most severe damage level is entered as KL1. The second damage level is entered as KL2, etc. There is no such requirement for the inclusive kill methodology.

6. The COMPPK data type may be repeated for the same component number for up to six damage mechanisms per kill level.
7. Continuation records are not necessary when entering table assignment for less than seven $P_{cd/h}$ tables.

Input Data Type: ENDDATA

Description: Defines end of file.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
ENDDATA									
ENDDATA									
C									

REMARKS:

1. The ENDDATA data type is required to appear in the JTYPE input database. If this record is omitted, COVART will report an error and stop.
2. The ENDDATA data type must be the last record that appears in the JTYPE file.
3. Any data following the ENDDATA data type in the JTYPE file will not be processed by COVART.

Input Data Type: JTYPECD

Description: Defines JTYPE file header code.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
JTYPECD	TCODE								
JTYPECD	JTYPE								
C	C								

<u>Parameters</u>	<u>Units</u>	<u>Description</u>
TCODE	-----	File code compared to parameter CODEJ on JTYPEFILE data type in MASTER file to ensure correct JTYPE file is processed.

REMARKS:

1. The JTYPECD data type is required to appear in the JTYPE input database. If this record is omitted, COVART will report an error and stop.
2. The JTYPECD data type must be the first record that appears in the JTYPE file.

Optional Records

Input Data Type: COMALIAS

Description: Used to define alias components.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
COMALIAS	ITDCOM	KAL(1)	KAL(2)	KAL(3)	KAL(4)	KAL(5)	KAL(6)	KAL(7)	C1
COMALIAS	8000	8001	8002	8003	8004	8005	8006	8007	MORE
C	I	I	I	I	I	I	I	I	C

C2		KAL(8)	KAL(9)	KAL(10)	KAL(11)	KAL(12)	KAL(13)	KAL(14)	C3
MORE		8001	8002	8003	8004	8005	8006	8007	
C		I	I	I	I	I	I	I	C

<u>Parameters</u>	<u>Units</u>	<u>Description</u>
ITDCOM	-----	Component number of a component in the target database file. This component is referred to as the “aliased” component.
KAL(I)	-----	Component number of the i^{th} component to be inserted ahead of each occurrence of component ITDCOM in a shotline or HEIVAM ray. This component is referred to as the “alias” component.
C1,C2...	-----	Continuation line flag, C1 must be identical to the character string and position (left-justified) of C2.

REMARKS:

1. The COMALIAS data type is an optional record that can appear in the JTYPE input database.
2. Alias components are used to model components that are both singly and multiply vulnerable for the same kill level.
3. It is critical that the user realizes that alias components are only a means to assign a component to more than one NAM per kill level and should not be used to replace

geometry within a target database.

4. Both the aliased component (ITDCOM) and the alias component(s) (KAL) must have COMPPHY and COMPPK records defined in the JTYPE file. The COMPPHY records for both the aliased component and the alias component(s) have to be defined before the COMALIAS records.
5. A component cannot be both aliased, and an alias.
6. Assign a material code of 51 (water), a density of -1 (0.01 inch normal thickness), and an LO number when defining an alias component with record type COMPPHY.
7. There can be at most 99 alias components assigned to the “real” aliased component.
8. If an alias is used because a component can be both singly and multiply vulnerable, then it is recommended that the singly vulnerable P_K tables be assigned to the “real” component, and the multiply vulnerable P_K tables be assigned to the alias.
9. Alias information is also output with the JTYPE echo. For a “real” component, at the end of the line, a “<” is followed by the numbers of the component’s aliases. At the end of the lines for an alias component, a “>” is followed by the component number of the “real” component it is associated with. This allows the user to add a component in front of a modeled component with no effect on penetration.
10. Alias components can only be assigned $P_{cd/h}$ tables of the weight-speed, aspect-dependent, and yaw angle-dependent types. This is a departure from the implementation found in the code prior to COVART 5.0, where it was possible to erroneously assign any $P_{cd/h}$ table to alias components. In older versions of COVART, the code would attempt to calculate a $P_{cd/h}$ for alias components under these circumstances with unpredictable results. Starting with COVART 5.0, if any $P_{cd/h}$ table type other than the three listed above is assigned to an alias component, the resultant $P_{cd/h}$ for that component is set equal to zero. As a result, the vulnerable areas resulting from target databases with incorrect alias components can be dramatically different between older versions of COVART and COVART 5.0.

MULTIPLY VULNERABLE FILE

INTRODUCTION

The components of a target can be divided (for a given threat) into two classes: critical and non-critical. Critical components are those parts of a platform that, when hit, can lead to its defeat at a particular damage level. Non-critical components are those elements whose defeat does not lead to the defeat of the platform but whose presence shields critical components. Critical components of a target can, in turn, be divided into two classes: singly vulnerable (SV) and multiply vulnerable (MV). A component is singly vulnerable if the defeat of that component alone results in the defeat of the platform. When two or more components must be killed simultaneously to defeat the platform, analysts refer to these components as multiply vulnerable components. Multiply vulnerable components can be grouped into sets of two or more systems. A kill results only if at least a specified number, $N > 1$, of the systems in the group are defeated (for a given kill category).

COVART supports the following features for fault trees: single-shot defeat assessment of a multiply vulnerable group, dependent components (i.e., a component appearing more than once in the fault tree for a particular multiply vulnerable group), weighting factors, and M of N kills.

Utilizing this feature requires the use of NAMs (i.e., names of singly vulnerable components or groups of components). The user creates and defines the composition of NAMs in the JTYPE file. For more information on the setup of the JTYPE file, please refer to the previous section.

This section provides an example of a platform with multiply vulnerable components to illustrate how one defines groups, systems, subsystems, and components within COVART. Some examples of groups include crew, propulsion, and flight controls. Table C-2 provides the breakdown for a potential aircraft propulsion system.

TABLE C-2. Multiply Vulnerable System Example

Group	- Propulsion	
System	- Rt Engine	Left Engine
Subsystem	- Compressor	Fuel Manifold and Jets
Components	- Stator Blades	Lines
	- Rotor Blades	Jets
	- Cans	

MULTIPLY VULNERABLE THEORY

COVART's features for processing multiply vulnerable components or subsystems allow users to model the mitigating effects of redundant components and/or subsystems on system/platform defeat. In processing, COVART calculates the probability of component dysfunction given a hit (Pcd/h) for components at the NAM level and then considers user-defined multiply vulnerable relationships in determining the probability of kill given a hit (Pk/h) for functions featuring redundancy. Helpful in the assessment of multiply vulnerable system performance are fault trees (i.e., deactivation diagrams), as they provide the means to visualize the redundancy found in a particular function. For example, for aircraft, it is common to have fault trees to represent such functions as the flight controls, propulsion, electrical systems, and the equipment necessary for mission completion. Figure C-1 provides an example of a flight control fault tree. COVART's multiply vulnerable component processing facilitate the input of fault trees to support vulnerability/lethality analyses.

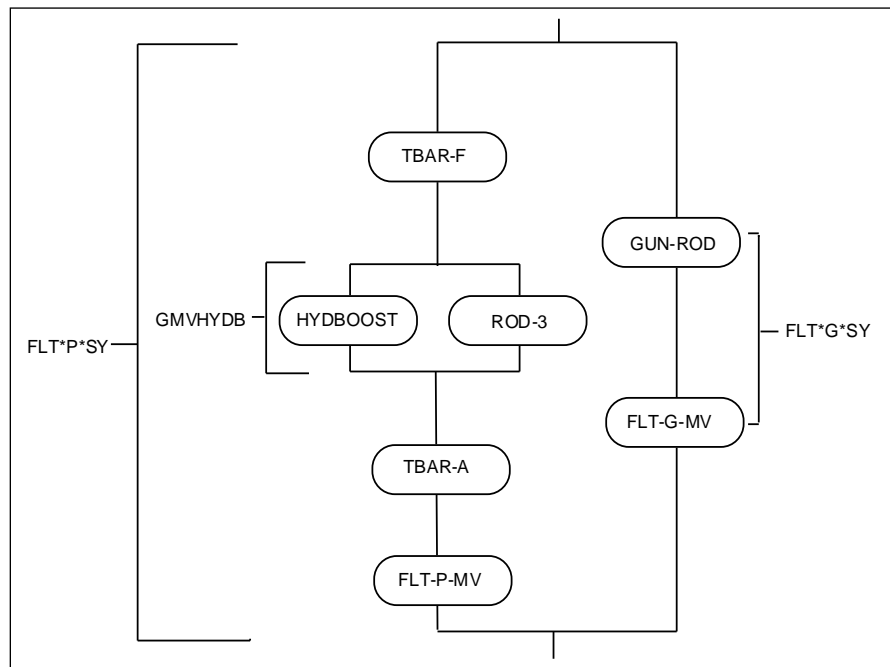


Figure C-1. FLT-C-MV Group.

Due to processing concerns, analysts should avoid including singly vulnerable components in multiply vulnerable fault trees where possible. The goal in COVART's handling of multiply vulnerable components and subsystems is to develop a group P_k that COVART can then combine with other singly vulnerable elements (via the survival rule) to calculate platform P_k . Singly vulnerable components/subsystems already meet the criteria for the survival rule, so no additional processing is necessary for these items. In fact, adding singly vulnerable branches to multiply vulnerable groups increases the complexity of the math supporting the assessment of group P_k values, which may have a negative impact on very complex fault trees. It helps to think of the processing for a

particular kill category/level as one big kill tree. Singly vulnerable components already lead to the kill of the platform at that level while multiply vulnerable components/subsystems need a little processing before one can apply their damage to the rest of the platform.

Fault tree descriptions in COVART employ a leveled approach. The highest level in multiply vulnerable processing in the model is the group. COVART considers the P_k for the group as singly vulnerable with respect to the platform although it may include redundant systems. A group includes one or more (up to 24) systems and each system may include up to 24 subsystems or components. Components comprise the lowest level of the fault tree hierarchy within COVART, and the user defines these via NAMs in the JTYPE file.

The elements within each group, system, or subsystem can have a series or parallel relationship with each other. A series relationship indicates that only one in a chain of elements must be defeated in order to defeat the larger subsystem, system, or group. On the other hand, a parallel relationship indicates that more than one element must be simultaneously defeated in order to defeat the larger subsystem, system, or group. Parallel relationships can require the defeat of all or some subset of its elements. When only a subset needs to be defeated, this case is referred to as an “M of N” relationship, where M is the number of elements that must be defeated in the system, and N is the total number of elements in the system. Additionally, the user has the ability to weight the defeat of certain combinations of elements when they are more or less lethal than other combinations. This is the weighting factor feature.

COVART’s multiply vulnerable system processing features allow for the definition of calculations that do not follow normal fault tree processing. These features do not influence the calculation of aircraft vulnerable areas but provide users the ability to combine the outputs of several components, subsystems, or systems into quantities useful for reporting results. The first is the COMBMV feature. With this feature, the user can add or subtract the resultant vulnerable areas of systems, subsystems, or components to determine other vulnerable areas of interest (e.g., a user could sum individual crewmember vulnerable areas to determine a crew vulnerable area). The second feature of this type is the SYSSING feature. Here, the user can combine groups of singly vulnerable components together under the same name for output purposes. Between the features here and in the JTYPE file, COVART provides the means for users to create meaningful results supporting their analyses.

DATA TYPES

The MVCD data type must be the first record in the Multiply Vulnerable (MV) file to ensure COVART processes the file the user truly intends to use. The program compares the TCODE parameter to the CODEMV parameter on the MVFILE data type in the MASTER file. If these character strings do not match, processing of the MV file is terminated, but the program will continue input validation of the remaining files.

An MV group definition starts with the MVKILL data type and ends with the ENDGROUP data type. COVART supports up to fifteen possible kill categories or levels, and each kill category can have a variable number of MV groups. The user specifies the applicable kill category using the MVKILL data type.

Following the MVKILL data type, the user provides the name of the group, NAM; the number of systems required for failure, M; and the number of systems, N, in the group. The user employs the GRPMULT data type to provide this information.

After the GRPMULT data type, the user can define the remainder of the group. This includes systems defined using the SYSMULT data type and subsystems defined using the NAMPRT data type. For subsystem inputs, the user can control the output of results for a particular subsystem with the PRT field of the NAMPRT record. At the lowest level of the MV group must be components (i.e., NAMs) that the user has defined in the JTYPE file. The user notes the end of inputs for the systems, subsystems, and components in the group using the ENDNAME data type.

The COMBMV data type defines mathematical equations involving the results calculated for systems and subsystems defined in a group. These equations have no impact on the total platform vulnerable area and are optional. COMBMV includes the name of the desired result, MNAM, followed by inputs for the subsystems and systems involved in the equation as well as the signs between quantities. Following any COMBMV inputs, the user finally ends a group definition with the ENDGROUP data type.

After the user inputs all groups for all kill levels, he or she can then input optional weighting factors for groups using the WGTFACT data type. The user specifies the group number (IGROUP, which is an index based on the order of the group definitions in the MV file), the number of systems in the group (LSYS), and the weighting factors (WGT) associated with each system combination. The user must input weighting factors in a particular order, so please check the remarks on the WGTFACT data type for more details.

Special singly vulnerable combinations of components are defined with the SYSSING data type. The user creates a name for the SYSSING definition (ANAM) along with the number of components in the system (NUM). The user identifies members of the SYSSING using component numbers (COMP). This feature differentiates the SYSSING data type from the other data types in the MV file. Most of the other data types in this file will reference components by their NAM, but the SYSSING references these components by their packed component number (same format as component numbers in the JTYPE file). The user should note that COVART does not store output information by component number but rather by NAM. As a result, assigning a component number to a SYSSING definition equates to assigning the corresponding NAM to the SYSSING. Also, care must be taken to not assign components to both MV and SYSSING definitions. COVART will check the NAM associated with each component number in a

SYSSING and determine if the NAM is part of an MV group as well. If it is, COVART will throw an error and drop the SYSSING from the analysis.

An illustration of how the multiply vulnerable input data is developed for a section of flight controls follows. Figure C-2 shows the section of flight control with the pilot's and gunner's system identified.

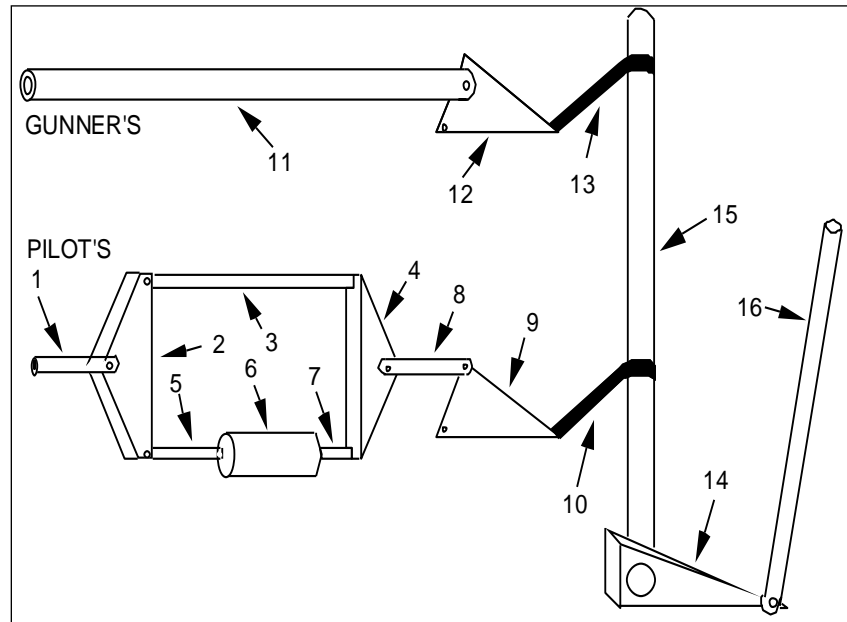


Figure C-2. Section of Flight Control System.

Table C-3 identifies the sixteen components in this flight control section. This table lists the vulnerable components, their JTYPE numbers, LO numbers, and the associated names (NAMs). The first thirteen form the multiply vulnerable group. Components 1 through 10 describe the pilot's system, and components 11, 12, and 13 describe the gunner's system. To defeat the flight control, both systems must be defeated. On the other hand, components 14, 15, and 16 are considered singly vulnerable. Since the NAM names for singly vulnerable components are not used in the definition of a multiply vulnerable group, their component numbers are listed instead.

Table C-3. Flight Control Vulnerable Components

MULTIPLY VULNERABLE COMPONENTS			
JTYP1	Component Name	LO Number	NAM Name
1	T-Bar Tube Member	28	TBAR-F
2	T-Bar Solid Member	28	TBAR-F
3	Control Rod	27	ROD-3
4	T-Bar Solid Member	25	TBAR-A
5	Control Rod	26	HYDBOOST
6	Hydraulic Boost Cylinder	26	HYDBOOST
7	Hydraulic Piston	26	HYDBOOST
8	T-Bar Tube Member	25	TBAR-A
9	Bellcrank	24	FLT-P-MV
10	Control Rod	24	FLT-P-MV
11	Control Rod	22	GUN-ROD
12	Bellcrank	23	FLT-G-MV
13	Control Rod	23	FLT-G-MV
SINGLY VULNERABLE COMPONENTS			
JTYP1	Component Name	LO Number	Component Number
14	Control Rod	22	1001
15	Pivot	21	1002
16	Control Rod	20	1003

A diagram considering only the multiply vulnerable components from Table C-3 is in Figure C-3. A smaller diagram is then created by combining JTYP1 numbers with the same LO numbers (Figure C-4).

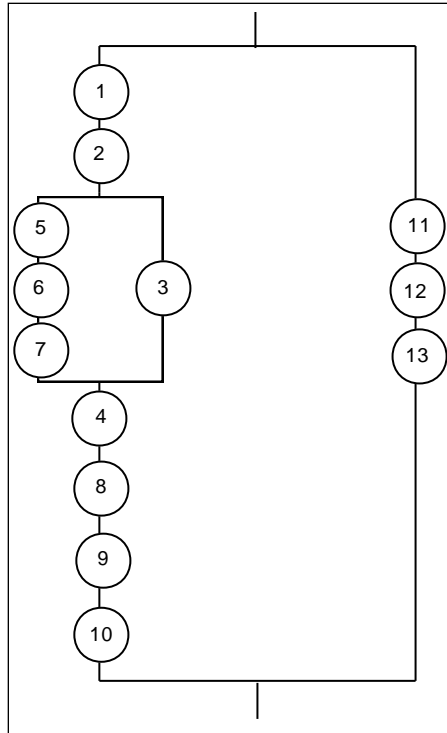


Figure C-3. Flight Control System with JTYPE Numbers.

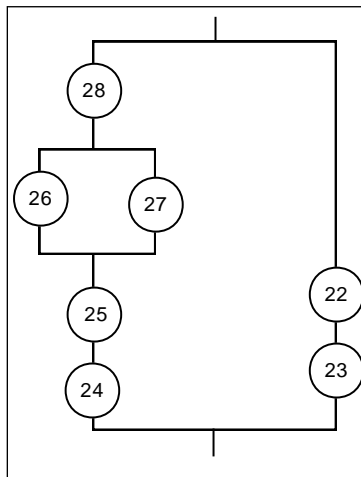


Figure C-4. Flight Control System with LO Numbers.

The LO numbers are now replaced with the corresponding NAM name. Moreover, systems and subsystems are also assigned names as shown in Figure C-1.

The fault tree pictured in Figure C-1 represents a portion of the flight control system of an aircraft. The whole fault tree corresponds to a multiply vulnerable group named FLT-C-MV. (This is just an arbitrary eight character name.) This group is composed of two

systems in parallel. The system on the left (FLT*P*SY) represents the pilot's flight controls. The system on the right (FLT*G*SY) represents the gunner's flight controls. FLT*P*SY is composed of 3 components, TBAR-F, TBAR-A, and FLT-P-MV, which are defined in the JTYPE input file, and one subsystem, GMVHYDB. TBAR-F, TBAR-A, FLT-P-MV, and GMVHYDB are combined in series to form system FLT*P*SY. FLT*G*SY contains components GUN-ROD and FLT-G-MV in series. GMVHYDB is a parallel combination of components HYDBOOST and ROD-3, hence both components must be defeated for subsystem GMVHYDB to be defeated.

A detailed description of the input data preparation process along with the required values is provided to assist in the generation of the multiply vulnerable database that will correspond to the above sample case.

1. The group designation is defined with the GRPMULT data type. The group consists of the pilot and gunner systems; both of which must be defeated to defeat the group.

Parameter	Value	Description
NAM	FLT-C-MV	Group name
M	2	Systems required for failure
N	2	Systems in the group
C1	FLT-C-MV	Continuation line flag

2. The elements within the group are defined on the continuation of the data type. The two systems are identified by name, NAM.

Parameter	Value	Description
C2	FLT-C-MV	Continuation line flag
N1	FLT*P*SY	System 1 NAM
N2	FLT*G*SY	System 2 NAM

3. The system designations are defined with the SYSMULT data type. The user must describe the composition of each system with a hierarchy of systems that ultimately end with components. First is the description of the pilot flight control system.

Parameter	Value	Description
NAM	FLT*P*SY	System name
M	1	Systems required for failure
N	4	Subsystems in the system
C1	FLT*P*SY	Continuation line flag

4. The elements within the system are defined on the continuation of the data type. The four subsystems are identified by name, NAM.

Parameter	Value	Description
C2	FLT*P*SY	Continuation line flag
N1	TBAR-F	Subsystem 1 NAM
N2	GMVHYDB	Subsystem 2 NAM
N3	TBAR-A	Subsystem 3 NAM
N4	FLT-P-MV	Subsystem 4 NAM

5. Next, the composition of gunner's systems is defined using the SYSMULT data type.

Parameter	Value	Description
NAM	FLT*G*SY	System name
M	1	Systems required for failure
N	2	Subsystems in the system
C1	FLT*G*SY	Continuation line flag

6. The elements within the system are defined on the continuation of the data type. The two subsystems are identified by name, NAM.

Parameter	Value	Description
C2	FLT*G*SY	Continuation line flag
N1	GUN-ROD	Subsystem 1 NAM
N2	FLT-G-MV	Subsystem 2 NAM

7. The composition of subsystem GMVHYDB is defined on the NAMPRT data type.

Parameter	Value	Description
NAM	GMVHYDB	Subsystem name
M	2	Components required for failure
N	2	Components in the system
PRT	0	Printer flag
C1	GMVHYDB	Continuation line flag

8. The elements within the subsystem are defined with the continuation of the data type. The two components are identified by name, NAM.

Parameter	Value	Description
C2	GMVHYDB	Continuation line flag
N1	HYDBOOST	Component 1 NAM
N2	ROD-3	Component 2 NAM

9. Lastly, the user completes the definition of group elements by including the ENDNAME data type.

If other vulnerable areas are desired, they can be specified using the COMBMV data type to add or subtract previously defined areas using Boolean algebra. For example, consider the vulnerable area of the pilot's system without the GMVHYDB subsystem which has two parallel elements. This vulnerable area, P*SY*SV, depends on the probability of defeating FLT*P*SY but not defeating GMVHYDB. This result is obtained by using the COMBMV data line with the following input parameters.

Parameter	Value	Description
MNAM	P*SY*SV	Multiply vulnerable subdivision name
NC1	FLT*P*SY	Element 1 NAM
S2	-	Sign used to combine elements
NC2	GMVHYDB	Element 2 NAM

This same vulnerable area could have been also defined by adding all the other elements in the system except GMVHYDB. This input on the COMBMV data line is as follows:

Parameter	Value	Description
MNAM	P*SY*SV	Multiply vulnerable subdivision name
NC1	FLT-P-MV	Element 1 NAM
S2	+	Sign used to combine elements
NC2	TBAR-F	Element 2 NAM
S3	+	Sign used to combine elements
NC3	TBAR-A	Element 3 NAM

Lastly, the entire multiply vulnerable group is completed by including an ENDGROUP data type line.

As an example of a singly vulnerable system, consider the three singly vulnerable components (JTYPE numbers 14, 15, and 16) of the flight control section pictured in Figure C-2. These components with component numbers 1001, 1002, and 1003 (listed in Table C-3) can be grouped as a singly vulnerable system. These data are input using the following parameters.

1. The singly vulnerable combination is defined with the SYSSING data type.

Parameter	Value	Description
ANAM	FLT-CTRL	Singly vulnerable system name
NUM	3	Number of components
C1	CONT	Continuation line flag

2. The composition of the singly vulnerable system is defined on the continuation of the data type. The user inputs the component number of each part forming the singly vulnerable system. Entering a component number here effectively assigns the associated NAM (e.g., NAM, "100", in this case) to the SYSSING.

Parameter	Value	Description
C2	CONT	Continuation line flag
COMP(1)	1001	Component number of 1 st component
COMP(2)	1002	Component number of 2 nd component
COMP(3)	1003	Component number of 3 rd component

Input Data Type: COMBMV

Description: Defines equations using results found for MV systems.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
COMBMV	MNAM	NC1	S2	NC2	S3	NC3	S4	NC4	C1
COMBMV	HYDR	HYD1	-	HYD2					
C	C	C	C	C	C	C	C	C	C

C2	S5	NC5	S6	NC6	S7	NC7	S8	NC8	
C	C	C	C	C	C	C	C	C	

<u>Parameters</u>	<u>Units</u>	<u>Description</u>
MNAM	-----	Multiply vulnerable subdivision name
NC1, ..., NC8	-----	NAM names that constitute a multiply vulnerable system
S2, S3, ..., S8	-----	Signs used to combine NAM names defined by NC1, NC2, etc. into the multiply vulnerable equation. Their values may be either a "+" (plus) or "-" (minus).
C1, C2	-----	Continuation line flag C1 must be identical to the character string and position (left-justified) of C2. It is recommended that C1 and C2 equal the character string of NAM.

REMARKS:

1. The COMBMV data type is optional. It must come after the ENDNAME data type and before the ENDGROUP data type.
2. If an equation includes less than four elements, a continuation record should not be used. However, if more than four items are included in an equation, the user must add a continuation record.
3. No equation can use more than eight items.

Input Data Type: ENDDATA

Description: Defines end of file.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
ENDDATA									
ENDDATA									
C									

REMARKS:

1. The ENDDATA data type is required any time the user inputs an MV database. If this record is omitted, COVART will report an error and stop.
2. The ENDDATA data type is required to be the last record in the optional MV database.

Input Data Type: ENDGROUP

Description: Specifies the end of a multiply vulnerable group.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
ENDGROUP									
ENDGROUP									
C									

REMARKS:

1. The ENDGROUP data type is required any time the user inputs an MV database. If this record is omitted, COVART will report an error and stop.
2. The ENDGROUP data type is required to be the last line in the specification of a multiply vulnerable group.
3. The user will next input another MVKILL record or continue to the end of file data types.

Input Data Type: ENDNAME

Description: Specifies the end of the formal MV group definition.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
ENDNAME									
ENDNAME									
C									

REMARKS:

1. The ENDNAME data type is required any time the user inputs an MV database. If this record is omitted, COVART will report an error and stop.
2. The ENDNAME data type is one of the last lines input for a particular multiply vulnerable group. Only the COMBMV or ENDGROUP data types can follow this data type.

Input Data Type: GRPMULT

Description: Defines composition of an MV group.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
GRPMULT	NAM	M	N						C1
GRPMULT	G	16	20						G1
C	C	I	I						C

C2	N(1)	N(2)	N(3)	N(4)	N(5)	N(6)	N(7)	N(8)	C3
G1	SYS1	SYS2	SYS3	SYS4	SYS5	SYS6	SYS7	SYS8	G2
C	C	C	C	C	C	C	C	C	C

C4	N(9)	N(10)	N(11)	N(12)	N(13)	N(14)	N(15)	N(16)	C5
G2	SYS9	SYS10	SYS11	SYS12	SYS13	SYS14	SYS15	SYS16	G3
C	C	C	C	C	C	C	C	C	C

C6	N(17)	N(18)	N(19)	N(20)	N(21)	N(22)	N(23)	N(24)	
G3	SYS17	SYS18	SYS19	SYS20					
C	C	C	C	C	C	C	C	C	

<u>Parameters</u>	<u>Units</u>	<u>Description</u>
NAM	-----	Group name.
M	-----	Number of systems required for failure.
N	-----	Number of systems in the group (maximum of 24).
C1, C2, C3, C4, C5, C6	-----	Continuation records. It is recommended that they equal the character string of NAM.
N(i)	-----	System or component NAM names on the right-hand side of the MV equation. These elements are combined to form

the group.

REMARKS:

1. The GRPMULT data type is required any time the user inputs an MV database. If this record is omitted, COVART will report an error and stop.
2. To illustrate, consider a group, G, comprised of three systems, S1, S2 and S3, where all three systems must be defeated to defeat the group. The logical “equation” to define this group is $G=S1.AND.S2.AND.S3$ (3/3), and the inputs associated with a definition like this would look like:

1	2	3	4	5	6	7	8	9	10
GRPMULT	NAM	M	N						C1
GRPMULT	G	3	3						GCONT
C	C	I	I						

C2	N1	N2	N3	N4	N5	N6	N7	N8	
GCONT	S1	S2	S3						
C	C	C	C						

If the same group had a serial relationship, only one system in the group needs to be defeated to defeat the vehicle. The logical equation in this case would be: $G=S1 .OR. S2 .OR. S3$. The inputs necessary to model this would look like:

1	2	3	4	5	6	7	8	9	10
GRPMULT	NAM	M	N						C1
GRPMULT	G	1	3						GCONT
C	C	I	I						

C2	N1	N2	N3	N4	N5	N6	N7	N8	
GCONT	S1	S2	S3						
C	C	C	C						

3. System or component names should not contain the characters ".", "/", "OR",

"AND", "+", or "-".

4. COVART will always print results for the group under the given name in the output section labeled, "Single Shot Vulnerable Areas of Multiply Vulnerable Groups".

Input Data Type: MVCD

Description: Specifies MV file code for comparison to MASTER file value.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
MVCD	TCODE								
MVCD	MV								
C	C								

<u>Parameters</u>	<u>Units</u>	<u>Description</u>
TCODE	-----	File code compared to CODEMV from MVFILE data type in the MASTER file to ensure that the correct Multiply Vulnerable (MV) file is processed.

REMARKS:

1. The MVCD data type is required any time the user inputs an MV database. If this record is omitted, COVART will report an error and stop.
2. The MVCD data type must be the first record that appears in the MV file.

Input Data Type: MVKILL

Description: Specifies beginning of new MV group and defines the applicable kill category (**Flight Phase for UEDDAM**).

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
MVKILL	KILLG								
MVKILL	3								
C	I								

<u>Parameters</u>	<u>Units</u>	<u>Description</u>
KILLG	-----	Kill category (flight phase for UEDDAM) code for multiply vulnerable groups. Since COVART6 allows the definition of up to 15 kill levels, any integer value from 1 to 15 is permissible.

REMARKS:

1. The MVKILL data type is required any time the user inputs an MV database. It is the starting record for every MV group definition.

INPUT DATA TYPE: NAMPRT

Description: Defines composition of a subsystem. The user can control whether COVART prints the results for this subsystem in A_V tables.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
NAMPRT	NAM	M	N	PRT					C1
NAMPRT	CNTRL	2	2	1					V1
C	C	I	I	I					C

C2	N1	N2	N3	N4	N5	N6	N7	N8	
V1	STICK	PEDAL							
C	C	C	C	C	C	C	C	C	

<u>Parameters</u>	<u>Units</u>	<u>Description</u>
NAM	-----	Component or subsystem name
M	-----	Number of subsystems (or components) required for failure. A value of “1” implies that the components are combined in series.
N	-----	Number of subsystems (or components) in the NAM
PRT	-----	Print vulnerable area table flag: 0 Do not print vulnerable area table 1 Print vulnerable area table
C1, C2	-----	Continuation line flag C1 must be identical to the character string and position (left-justified) of C2. It is recommended that C1 and C2 equal the character string of NAM.
N1, N2, ..., N8	-----	Subsystem of component NAM names on the right-hand side of the MV equation. These elements are combined to form the system.

REMARKS:

1. The NAMPRT data type is an optional record in the MV database file.
2. The NAMPRT record may include up to 24 components or subsystems. Component names should come from the JTYPE while the user should use NAMPRT records to define the composition of other subsystems.
3. Component and subsystem names should not contain the strings “.”, “/”, “OR”, “AND”, “+”, or “-“.

INPUT DATA TYPE: SYSMULT

Description: Defines composition of a system.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
SYSMULT	NAM	M	N						C1
SYSMULT	BLADE	1	20						S1
C	C	I	I						C

C2	N(1)	N(2)	N(3)	N(4)	N(5)	N(6)	N(7)	N(8)	C3
S1	SYS1	SYS2	SYS3	SYS4	SYS5	SYS6	SYS7	SYS8	S2
C	C	C	C	C	C	C	C	C	C

C4	N(9)	N(10)	N(11)	N(12)	N(13)	N(14)	N(15)	N(16)	C5
S2	SYS9	SYS10	SYS11	SYS12	SYS13	SYS14	SYS15	SYS16	S3
C	C	C	C	C	C	C	C	C	C

C6	N(17)	N(18)	N(19)	N(20)	N(21)	N(22)	N(23)	N(24)	
S3	SYS17	SYS18	SYS19	SYS20					
C	C	C	C	C	C	C	C	C	

<u>Parameters</u>	<u>Units</u>	<u>Description</u>
NAM	-----	System name
M	-----	Number of subsystems (or components) required for failure. A value of "1" implies that the components are combined in series.
N	-----	Number of subsystems (or components) in the system (1≤N≤24)
C1, C2, C3 C4, C5, C6 N(i)	----- -----	Continuation records. It is recommended that they equal the character string of NAM. System or component NAM names on the right-hand side

of the MV equation. These elements are combined to form the group.

REMARKS:

1. The SYSMULT data type is a required record any time the user inputs an MV database.
2. System, subsystem, or component names should not contain the characters “.”, “/”, “OR”, “AND”, “+”, or “-”.
3. COVART will always print results for the group under the given name in the output section labeled, "Single Shot Vulnerable Areas of Multiply Vulnerable Groups".

INPUT DATA TYPE: SYSSING

Description: Defines singly vulnerable components to be combined into a singly vulnerable system NAM name.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
SYSSING	ANAM	NUM							C1
SYSSING	ENGINE	10							V1
C	C	I							C

C2	CMP(1)	CMP(2)	CMP(3)	CMP(4)	CMP(5)	CMP(6)	CMP(7)	CMP(8)	C3
V1	1001	1002	1003	1004	1005	1006	1007	1008	V2
C	I	I	I	I	I	I	I	I	C

C4	CMP(9)	CMP(10)	CMP(11)	CMP(12)	CMP(13)	CMP(14)	CMP(15)	CMP(16)	
V2	1009	1010							
C	I	I	I	I	I	I	I	I	

<u>Parameters</u>	<u>Units</u>	<u>Description</u>
ANAM	-----	Singly vulnerable system name.
NUM	-----	Number of components to form the singly vulnerable system ANAM
C1-C4	-----	Continuation line flag C1 must be identical to the character string and position (left-justified) of C2; C3 must also equal C4.
CMP(i)	-----	ith component number that forms the singly vulnerable system ANAM. Only one component for each NAM name is required.

REMARKS:

1. The SYSSING data type is optional.

2. SYSSING records must appear outside the definition of a group (i.e., after a record of type ENDGROUP). It is preferable that the SYSSING records appear at the end of the MV file to make it more understandable.
3. The total number of all components combined to form all ANAMs and twice the number of ANAMs must be less than 1000.
4. If there are more than eight components specified on the SYSSING record, input continues on a continuation line, and C1 and C2 must be specified and agree in character string and position.
5. It is important to note that adding a component number to a SYSSING effectively assigns the NAM in the JTYPE file via COMPPHY records. The presented and vulnerable areas reported by COVART for the SYSSING will reflect the results calculated for the corresponding NAM's and not the individual components entered here.
6. Components cannot be assigned to both MV and SYSSING groups. When COVART detects that a SYSSING component is part of an MV definition, the program will throw an error and drop the SYSSING associated with the component.

INPUT DATA TYPE: WGTFACT

Description: Defines weighting factors.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
WGTFACT	IGROUP	LSYS							C1
WGTFACT	1	2							V1
C	I	I							C

C2	WGT(1)	WGT(2)	WGT(3)	WGT(4)	WGT(5)	WGT(6)	WGT(7)	WGT(8)	C4
V1	.2	.3	.05	.45					
C	R	R	R	R	R	R	R	R	C

<u>Parameters</u>	<u>Units</u>	<u>Description</u>
IGROUP	-----	Group number for which multiply vulnerable weighting factors must be read. Groups are numbered in the order of the GRPMULT data types that define the groups.
LSYS	-----	Number of systems in the group IGROUP (maximum of eight systems in a group)
C1, C2	-----	Continuation line flag C1 must be identical to the character string and position (left-justified) of C2.
WGT(1)	-----	Weighting factor for the 1st term ($Q_1 Q_2 Q_3 \dots Q_{LSYS}$)
WGT(2)	-----	Weighting factor for the 2nd term ($P_1 Q_2 Q_3 \dots Q_{LSYS}$), where P_i is the probability of kill given a hit on the i th component and $Q_i = 1 - P_i$.
WGT(3)	-----	Weighting factor for the 3rd term ($Q_1 P_2 Q_3 \dots Q_{LSYS}$)
WGT(4)	-----	Weighting factor for the 4th term ($P_1 P_2 Q_3 \dots Q_{LSYS}$)
WGT(5)	-----	Weighting factor for the 5th term ($Q_1 Q_2 P_3 \dots Q_{LSYS}$)
WGT(6)	-----	Weighting factor for the 6th term
WGT(7)	-----	Weighting factor for the 7th term

WGT(8) ----- Weighting factor for the 8th term ($P_1 P_2 P_3 \dots Q_{LSYS}$)

REMARKS:

1. The WGTFACT data type is an optional record.
2. The WGTFACT data type should come after the last ENDGROUP record in the MV file.
3. IGROUP is an index of the group within the context of the MV file. The group associated with the first GRPMULT record in the file is IGROUP = 1. It may help the user to track IGROUP in a comment before the MVKILL record for a particular group when they desire to use weighting factors.
4. P_i implies that the i^{th} component is killed, and Q_i implies that the i^{th} component survives.
5. 2^{LSYS} = number of weighting factors to be read in. If the value of LSYS is three or less, only one continuation line with the eight weighting factors is required. Otherwise, additional continuation lines are required according to the list below:

LSYS	Continuation Lines
3	1
4	2
5	4
6	8
7	16
8	32

6. Weighting factors must be input in a specific order. The first weighting factor input applies to the state of the multiply vulnerable group in which none of its systems have been killed ($Q_1, Q_2, Q_3 \dots Q_{LSYS}$). The second weighting factor input applies to the state of the group in which only system number 1 has been killed ($P_1, Q_2, Q_3 \dots Q_{LSYS}$). The third weighting factor input applies to the state of the group in which only system number 2 has been killed ($Q_1, P_2, Q_3 \dots Q_{LSYS}$). The systems are numbered in the order in which their names are entered on the GRPMULT data type. That is, system number 1 is the first system named on the second record of the GRPMULT data type, system number 2 is the second system named, and so on. The value of LSYS entered on the WGTFACT data type must match the number of systems actually entered on the GRPMULT data type for the group.

PCD/H FILE

INTRODUCTION

One required input file for every COVART analysis is a probability of component dysfunction given a hit ($P_{cd/h}$) database file. The contents of this file are a series of tables and curves that estimate component “damage” resulting from the damage modes of kinetic energy or high-explosive weapons. Historically, COVART interpreted the probabilities defined in this database as probabilities of kill given a hit ($P_{k/h}$). $P_{k/h}$ is the product of $P_{cd/h}$ and the probability of system kill given component damage ($P_{k/d}$) as shown in the equation below.

$$P_{k/h} = P_{cd/h} \times P_{k/d}$$

In prior versions of COVART, the user created a $P_{k/h}$ database file. In this file, the user had to combine $P_{cd/h}$ and $P_{k/d}$ into a single table that only applied to a particular modeling situation. In COVART6, $P_{cd/h}$ and $P_{k/d}$ are now separate with the user defining $P_{cd/h}$ through a series of curves and $P_{k/d}$ with a constant value. The $P_{cd/h}$ tables reflect the $P_{k/h}$ tables historically input into COVART while a new file supports the input of $P_{k/d}$ values. This new paradigm allows the user to create curves based solely on component dysfunction and modify the effects of placing the component in different contexts.

DATA TYPES

In the Pcd/h file, the PKCD data type must come first. The program compares the TCODE parameter to CODEP parameter on the PKFILE data type in the MASTER file. If these character strings do not match, processing of the $P_{cd/h}$ file is terminated, but the program will continue input validation of remaining files.

The next three record types (FIREDATA, AIRGAP, and YAW) are required for COVART but not used in UEDDAM. For UEDDAM runs, use the following default settings (where “.” indicates a space):

```
FIREDATA .....0.5 .....2  
AIRGAP .....1.0 .....3.0 .....6.0 .....12.0 .....18.0 .....24.0  
FIRETHK .....100.0  
YAW .....10.0 .....20.0 .....30.0 .....40.0 .....50.0 .....60.0
```

The most numerous record in the $P_{cd/h}$ file is the PKTABLE record. The first line of this data type defines the kind of $P_{cd/h}$ table, the number of curves composing the table, and parameters defining the applicability of the table. After this, the remaining lines of the data type define the associated curves. The first input on a PKTABLE record is the $P_{cd/h}$ table number, K, followed by the identification label, PKTYPE. Following the label, the user provides one or more of the following pieces of information depending on the type of table under consideration:

- WCUT – the smallest mass that will cause damage

- DCUT – the smallest gap in front of a component that will result in a front-face fire
- NM – the number of mass curves
- INFL – a flag for influence mode
- MFLG – a flag to indicate whether a CMHITS table is in penetration or impact mode
- IANYDB – a flag for determining the applicability of the table
- LAB1, LAB2 – eight-character labels for the table when it is echoed to the output

The PKTABLE data type is used to input all the $P_{cd/h}$ tables. The format of each type of kill table is similar; however, the data used to define the actual curves may differ.

$P_{CD/H}$ TABLES

The $P_{cd/h}$ table format used for UEDDAM is the piecewise linear impactor weight table format (WGT).

The two basic forms for $P_{cd/h}$ input are a piecewise linear table or an exponential curve. Piecewise linear tables can be input as a single table or as multiple sets of tables. The exponential function is included but is rarely used. Each PKTABLE type requires different numbers of tables and curves within each table.

A piecewise linear $P_{cd/h}$ table is described by a series of up to seven straight lines (i.e., 8 velocity versus $P_{cd/h}$ points) per mass curve. Several mass curves can be entered for one $P_{cd/h}$ table. An example of piecewise linear $P_{cd/h}$ curves is shown in Figure C-5.

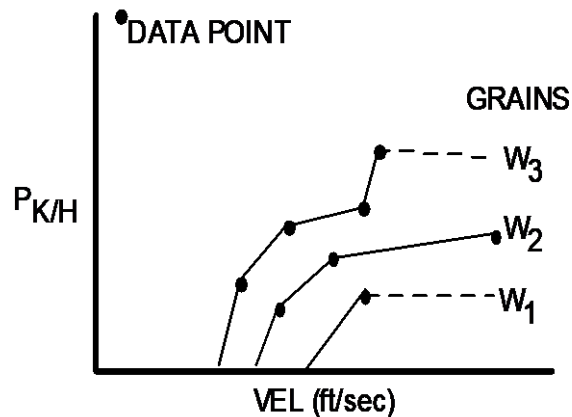


Figure C-5. $P_{CD/H}$ Curves.

Input Data Type: ENDDATA

Description: Defines end of file.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
ENDDATA									
ENDDATA									
C									

REMARKS:

1. The ENDDATA data type is required in the P_{cd/h} database. If this record is omitted, COVART will report an error and stop.
2. The ENDDATA data type must be the last record that appears in the P_{cd/h} file.
3. Any data that follows the ENDDATA data type in the P_{cd/h} file will not be processed by COVART.

Input Data Type: PKCD

Description: Defines P_{cd/h} file code.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
PKCD	TCODE								
PKCD	PK								
C	C								

<u>Parameters</u>	<u>Units</u>	<u>Description</u>
TCODE	-----	File code compared to CODEP from PKFILE data type in MASTER file to ensure correct P _{cd/h} file is processed

REMARKS:

1. The PKCD data type is required to appear in the P_{cd/h} database. If this record is omitted, COVART will report an error and stop.
2. The PKCD data type must be the first record that appears in the P_{cd/h} file.

Input Data Type: PKTABLE – WGT

Description: Defines piecewise linear $P_{cd/h}$ table that depends on impactor mass.

Format, Example, and Data Type:

1	2	3	4	5	6	7	8	9	10
PKTABLE	K	PKTYPE(K)	WCUT(K)	NM		IANYDB(K)	LAB1	LAB2	C1
PKTABLE	12	WGT	10.	10		0	TEST	WGT	P1
C	I	C	R	I		I	C	C	C

C2	RWT(J)	VEL(1)	PRK(1)	VEL(2)	PRK(2)	VEL(3)	PRK(3)	IFLPK	C3
P1	720.	1800.	.05	2000.	.07	3000.	0.1	0	P2
C	R	R	R	R	R	R	R	I	C

C4		VEL(4)	PRK(4)	VEL(5)	PRK(5)	VEL(6)	PRK(6)	IFLPK	C5
P2		4050.	0.25	6500.	0.3	7000.	0.48	0	P3
C		R	R	R	R	R	R	I	C

C6		VEL(7)	PRK(7)	VEL(8)	PRK(8)			IFLPK	C7
P3		7500.	0.9	9000.	1.0			1	
C		R	R	R	R			I	C

<u>Parameters</u>	<u>Units</u>	<u>Description</u>
K	-----	$P_{cd/h}$ table number
PKTYPE	-----	WGT label identifying the $P_{cd/h}$ curve
WCUT	grains	Smallest mass that will cause damage in the kth $P_{cd/h}$ table. This mass should be as large as possible to reduce the computer processing time and to produce $P_{cd/h}$ values when interpolation occurs.
NM	-----	Number of mass curves (maximum of 15 curves)

IANYDB	-----	Flag for plate debris (Set to 0 for UEDDAM)
LAB1	-----	Label printed on the first line at the right-hand side of the $P_{cd/h}$ table in the echo file
LAB2	-----	Label printed on the second line at the right-hand side of the $P_{cd/h}$ table in the echo file
RWT(J)	grains	Mass associated with the j^{th} $P_{cd/h}$ curve
VEL(I)	ft/sec	i^{th} impact speed for the j^{th} $P_{cd/h}$ curve
PRK(I)	-----	i^{th} $P_{cd/h}$ value for the j^{th} $P_{cd/h}$ curve
IFLPK	-----	Flag that indicates whether this line continues or ends a $P_{cd/h}$ curve 0 continues the $P_{cd/h}$ curve 1 ends the $P_{cd/h}$ curve
C1 - C6	-----	Continuation line flag C1 must be identical to the character string and position (left-justified) of C2, C3 should be identical to C4, and C5 should be identical to C6.

REMARKS:

1. The impact speeds must be input as increasing values for proper interpolation between points in the curve, and no two impact speeds may be the same for any one curve.
2. The impact masses must be input as increasing values for proper interpolation between points.
3. Each new mass index starts on a new data line. This represents the beginning of a new curve.
4. There must be at least two, but not more than eight, pairs of values for VEL and PRK for each mass curve.
5. If more than three pairs of speed and $P_{cd/h}$ combinations are input, additional lines are required.
6. Parameters PKTYPE, WCUT, INFL, and IANYDB are each stored in an array subscripted by the $P_{cd/h}$ table number K.

APPENDIX D

**EXCERPTS FROM THE REPORT “UNCONTAINED ENGINE DEBRIS
ANALYSIS USING THE UNCONTAINED ENGINE DEBRIS DAMAGE
ASSESSMENT MODEL”**

The following is an excerpt from the following report: Seng, Silvia, John Manion, and Chuck Frankenberger, *Uncontained Engine Debris Analysis Using the Uncontained Engine Debris Damage Assessment Model*, DOT/FAA/AR-04/16, NAWC-WD, China Lake, California, September 2004. Included are the results from two generalized UEDDAM analyses: one for a generic business jet and one for a generic twin-engine aircraft.

DISCUSSION

OBJECTIVE

The objective for this study was to demonstrate the capabilities of the UEDDAM code. Through this process, an analysis was conducted that illustrates how UEDDAM can be used to address current Advisory Circular (AC) 20-128A methodologies, in addition to providing a design tool to conduct trade studies aimed at mitigating the uncontained engine debris hazard. Two generic aircraft models were defined: a generic twin-engine aircraft and a generic business jet. To ensure that the generic aircraft were representative, industry input was used in the construction of the FASTGEN geometric target models, fault trees, and multiply vulnerable component list.

DEFINITIONS AND NOMENCLATURE

Several terms are used in this report to describe the analysis that are not self-explanatory. This section will discuss what is meant by these terms.

A rotor burst analysis is conducted using several debris scenarios: a single disk segment, single blade fragment, or multiple fragments. The debris category is a name used to define the type of analysis being conducted (i.e., for a single disk analysis the debris category used is DISK1, for multiple fragments, the debris category is MULTI). The debris types are a description used by UEDDAM to distinguish how the fragment will be defined. There are three debris types: small fragments, large fragments, and sections. So, a single small fragment debris category (SINGL) may consist of a single small fragment, and a multiple fragment debris category (MULTI) may consist of a 1/3 disk section and 12 small fragments.

The small fragment debris type refers to fragments that are small enough to be sufficiently defined by a single shotline. The size of these fragments are described by height, width, and thickness. The large fragments are also described by height, width, and thickness, but are too large to for a single shotline to represent the path of the debris. Therefore, a grid is constructed within the fragment presented area. The grid spacing is defined by the user in the debris record. The center of the grid is the centroid of the presented area. The grid lines are parallel lines mapped relative to the fragment centroid. A shotline will originate at each point on the resulting grid. The shape of the section debris type is defined by the arc of the segment (in degrees) and the inner and outer radii.

The grid spacing is defined in the debris record and are mapped as parallel gridlines and drawn outward from the centroid (Figure D-1).

UEDDAM was run for each combination of flight phase, aircraft engine, rotor stage, and uncontained debris category. Each of the individual runs will be referred to as a case. A run will refer to a single execution of UEDDAM whereas a production run will indicate all the runs required to produce the output for all combinations of flight phase, rotor, engine, and debris category for a given aircraft configuration.

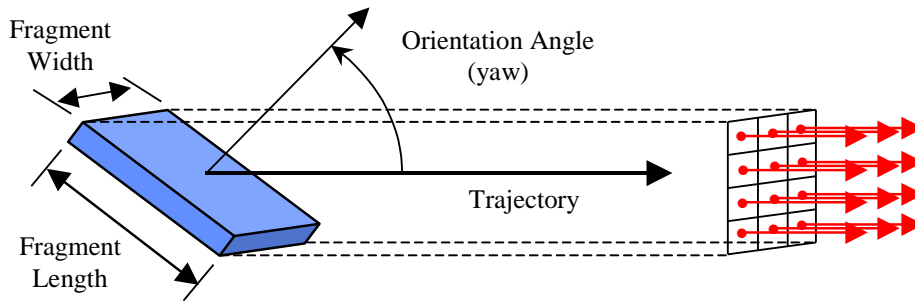


Figure D-1. Shotline Grid Definition.

The probability of catastrophic hazard to the aircraft will be denoted as P_{HAZ} . P_{HAZ} is the 1-in-20 value specified in the AC 20-128A and is specific to a particular debris category. The rotor, flight phase, or engine average probability of hazard will be identified by describing the level of averaging and P_{haz} (lowercase subscript), for example Engine 1 average P_{haz} .

ANALYSIS APPROACH

ANALYSIS OBJECTIVES

The objectives for the analysis for the two generic aircraft were to determine the probability of catastrophic hazard given an uncontained engine event (P_{HAZ}) for each aircraft, determine the major contributors to the P_{HAZ} , and perform trade studies to minimize the contributions of those components to the P_{HAZ} .

TARGET DESCRIPTIONS

The aircraft geometric representations are called the geometric target models and were created using FASTGEN4. Some of the components were excluded from the geometric target models because the component was noncritical and did not significantly reduce the fragment velocity, such as the interior wall panels.

Generic Business Jet

The geometric target for the GBJ comprises 11.5 ft behind the aft pressure bulkhead (Figure D-2) of the aircraft. The engines are small, high-bypass ratio turbofan engines mounted on the aft fuselage. Each engine consists of nine rotors: the fan and low-pressure compressor (LPC1 and LPC2); three-stage, high-pressure compressor (HPC1, HPC2, and HPC3); one-stage, high-pressure turbine (HPT1); and a three-stage, low-pressure turbine (TLP1, TLP2, and TLP3—the nomenclature was changed because LPT is a computer-reserved word). Some of the components included in the target model are the fuselage frames, longerons, skin, engine beams, fuel supply lines, fuel motive lines, control cables, nacelles, engine case, and the aft pressure bulkhead. The forward fuselage was not included because the forward-most spread angle did not intersect any components forward of the aft pressure bulkhead.

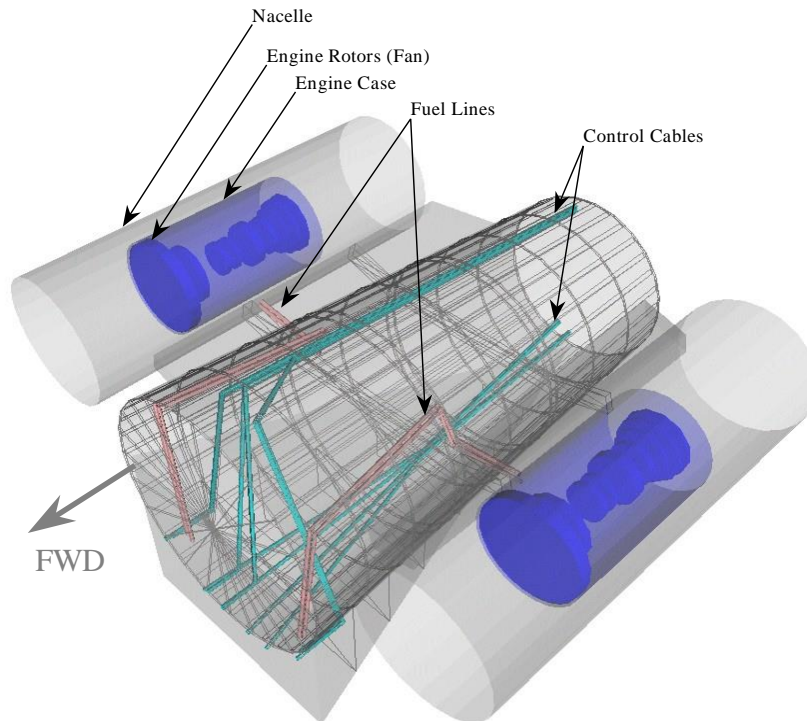


Figure D-2. Generic Business Jet Target Geometry.

Generic Twin-Engine Aircraft

The GT geometric target consisted of the fuselage of a wing-mounted twin-engine aircraft (Figure D-3). The engines are large, high-bypass ratio turbofans. Each engine consisted of 18 rotors. The systems included in the geometric target model are the fuel, hydraulic, and flight control systems, including pumps, actuators, mixers, cables, and supply and return lines. Fuselage and wing fuel tanks were included as well as the fuel motive lines. The aircraft structure included frames and longerons in the fuselage and a

more complex wing structure. Because damage to a spar web did not result in failure of the spar unless several other components were damaged as well, the spars were split into the upper spar cap, web, and lower spar cap. Although seating and main cabin furniture and fixtures were not included, the cabin floor was included.

The fuselage skin was created in a three-dimensional computer-aided design (CAD) program. Small triangles were used to provide the contours of the skin when converting the skin from the CAD program to FASTGEN4 geometry. The number of elements (triangles) generated by the conversion often result in large file sizes of as much as two orders of magnitude greater than those constructed using the simple geometric shapes available in FASTGEN such as cylinders, cones, and rectangular boxes. Therefore, many of the components for the GT were created using the FASTGEN4 primitives.

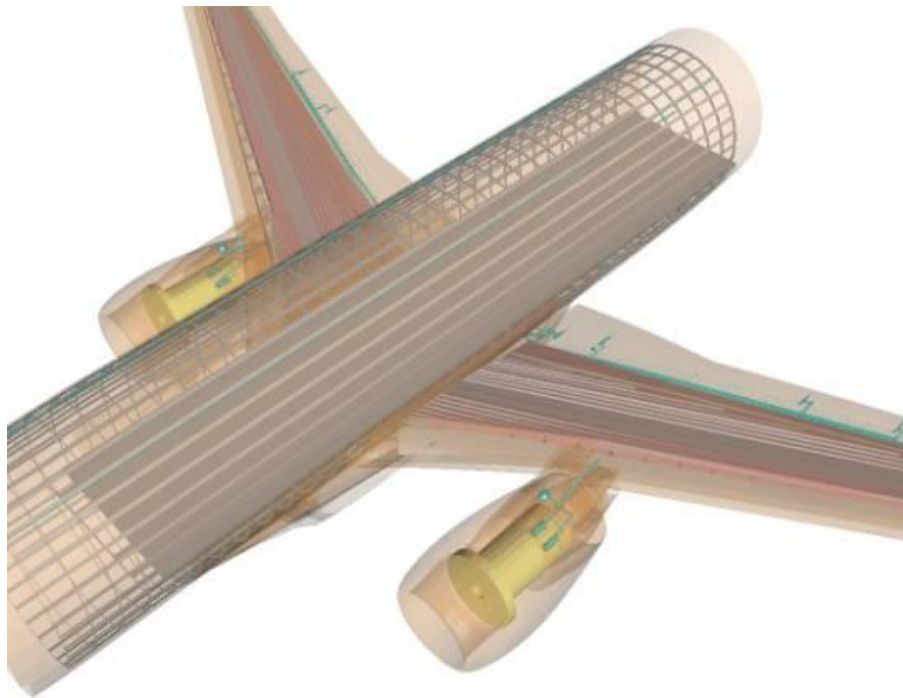


Figure D-3. Generic Twin Target Geometry.

ANALYSIS PARAMETERS

The following input parameters and assumptions were used for both the GBJ and GT aircraft UEDDAM analyses.

Flight Phase Probability of Failure

Rotor burst analyses were conducted for catastrophic hazards to the aircraft. Because component criticalities can vary with flight phase, the criticality of each component was defined for each flight phase. The breakdown of flight phases was based on the accepted

TABLE D-2. High-Bypass Ratio Engine Fragment Characterization

Component	Number of Fragments (Average/Event)	Normalized Size	Weight lbs (% of total)	Velocity (ft/sec)at 0° plane	Spread Angle Degrees
Fan					
Blade Event (Helical)	5.25	25%	1.8 (20%)	713	+21 to -35
Disk Event					
Blades	27.7	25%	1.8 (20%)	813	+15 to -30
Disks	3	100%	45 (38%)	303	+2 to -3
Compressor					
Blade Event					
Spacer – Rim					
Blades	6.5	100%	0.25 (100%)	642	+15 to -3
Rim	2	80%	6.0 (30%)	523	+15 to 0
Disk Event					
Blades	4.5	50%	0.19 (76%)	609	+5 to -25
Disk (Large Fragment)	1	85%	9 (45%)	380	+5 to -5
Disk (Intermediate Fragment)	1	30%	4 (20%)	385	+10 to -5
HP Turbine					
Blade Event	10	80%	0.25 (100%)	337	+20 to -50
Spacer- Rim					
Blades	11	85%	.25 (100%)	871	+15 to -40
Rim	1	50%	10 (83%)	967	0 to -12
Spacer	1	50%	2 (16%)	781	+20 to -37
Disk Event					
Blades	12	70%	.25 (100%)	871	+15 to -60
Disk	1	30%	10 (8%)	967	+3 to -11
LP Turbine					
Blade Event	6.5	50%	.25 (37%)	212	+20 to -45
Blade Event Last Stage	6.5	50%	.25 (37%)	200	0 to -75
Spacer- Rim					
Blades	5	15%	0.05 (7%)	326	+6 to -20
Rim	1	20%	11.3 (9%)	505	+5 to -5
Disk Event					
Blades	5	28%	0.25 (20%)	313	+5 to -40
Disk	1	20%	18 (15%)	535	+3 to -5

+ angles are forward, - angles are aft of the rotor plane of rotation

Note: See Appendix F for the latest fragment model, this table has been superseded.

The DFM describes the debris fragments in nondimensional terms for each engine section, such as fan blade size as a percentage of the full blade length. These nondimensional values were applied to each rotor to determine the size and mass of the fragments used in this analysis.

The following debris categories were used in the analysis of the GBJ and GT.

- DISK1—a single 1/3 disk section of the rotor representing a disk event
- MFRAG—an intermediate sized fragment representing a rim event
- MULTI—a single 1/3 disk section and multiple small fragments representing a disk event
- SINGL—a single small fragment, representing a blade event
- DSKFR—a single small fragment and a single 1/3 disk section
- MULTI3—three 1/3 disk fragments and associated small fragments for 1 of the 1/3 sections

At the recommendation of the 28th ARAC, Power Plant Installation Harmonization Working Group (PPIHWG) meeting participants, the last two debris categories were not used for the trade studies. This reduction in debris categories was made to reduce the number of analysis cases.

The debris characterization for the AC 20-128A suggested debris categories uses the same spread angles for all engine sections. (Figures D-4 and D-5 show the spread and release angle definitions.) The AC 20-128A recommends that spread angles of $\pm 3^\circ$ be used for 1/3 disk cases, $\pm 5^\circ$ for intermediate sized fragments (rim events) and $\pm 15^\circ$ for small fragments (blade events). The DFM spread angles are defined by engine section for each debris category. Table D-3 shows a comparison of the DFM-recommended debris characterization and the AC 20-128A recommended values. In most cases, the DFM spread angles are larger than the FAA recommendations; however, the three boxes highlighted in table D-3 (1/3 disk and intermediate fragment events for the fan and the small fragments from the compressors spread angles) indicate spread angles lower than those suggested by the AC 20-128A. Also, in some cases, the DFM suggested angles are shifted forward or aft of the existing AC 20-128A values. However, because of the overlap in the hazard zones of multiple rotors, the overall coverage for the DFM spread angles is greater than that of the AC.

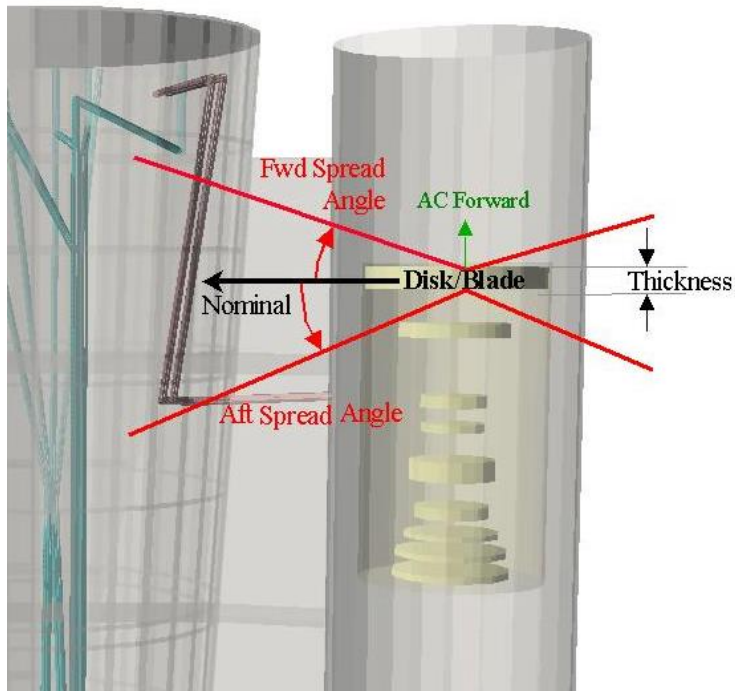


Figure D-4. Forward and Aft Spread Angle Definition.

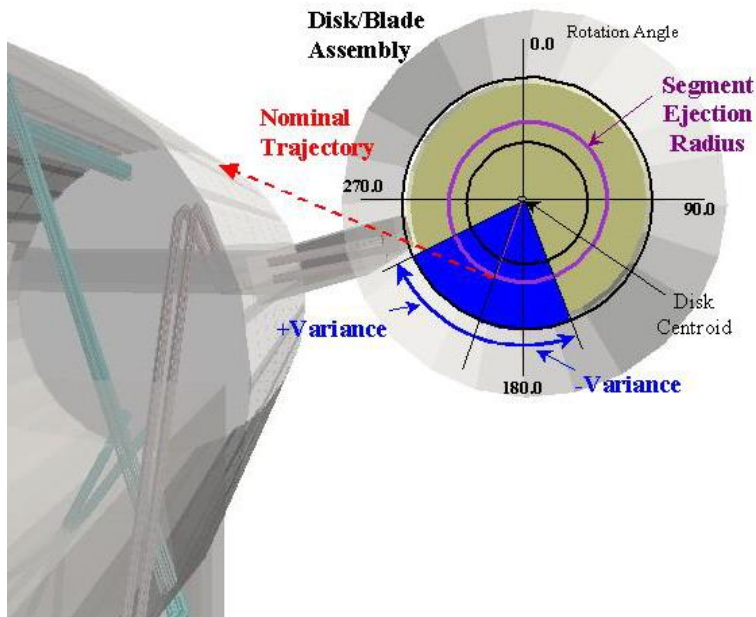


Figure D-5. Translational Angle Limits Definition.

TABLE D-3. Comparison of DFM and AC 20-128A Spread Angles

	Disk Event (1/3 Disk)		Rim Event (Intermediate Fragment)		Blade Event (Small Fragment)	
	DFM	AC20-128	DFM	AC20-128	DFM	AC20-128
	FAN	+2°/-3°		+2°/-3°		+21°/-35°
LPC	+5°/-5°		+10°/-0°		+15°/-3°	
HPC		+3°/-3°		+5°/-5°		+15°/-15°
HPT	+3°/-11°		+0°/-12°		+20°/-50°	
LPT	+3°/-5°		+5°/-30°		+20°/-45°	

Near-Field Components

UEDDAM uses the near-field file to list event-engine specific components that are close to the engine (within the nacelle). The DFM defines fragment velocities after exiting the nacelle; therefore, the nacelles, engine case, and engine-mounted accessories were placed in the near-field file. Thus, those components defined in the near-field file do not reduce the energy of the debris fragments.

Auto-Fail Components

UEDDAM uses the autofail file to define an event-engine specific list of components that would be nonfunctional as a result of an uncontained engine event. For example, generators that are run by the engine or other accessories dependent on bleed air may not function following a rotor burst, even if they were not hit by debris. The rotors and critical engine-mounted accessories for each aircraft were listed in the autofail file.

Probability of Catastrophic Hazard Given a Hit ($P_{\text{haz/h}}$)

A Failure Modes and Effects Analysis was used to identify critical components and aid in a fault tree analysis. The fault tree analysis was then used to aid the definition of multiply vulnerable groups. The fault tree in Figure D-6 reflects a single phase of flight of the GBJ. This type of analysis was repeated for each phase of flight for both aircraft examined in this analysis.

Critical components were listed in the JTYPE file and assigned to PK tables. The PK tables, specified in the PK file, define the probabilities of catastrophic hazard, given a hit ($P_{\text{haz/h}}$). The PK tables consist of several curves that define the component $P_{\text{haz/h}}$, given velocity for fragments of different sizes. Each critical component is assigned a PK table; however, each PK table can be used to describe multiple components. Using these tables, COVART interpolates the $P_{\text{haz/h}}$ of a component for fragments of any size and velocity. The $P_{\text{haz/h}}$ combines the probability of component dysfunction, given a hit ($P_{\text{d/h}}$), and the probability of catastrophic hazard, given component dysfunction ($P_{\text{haz/d}}$).

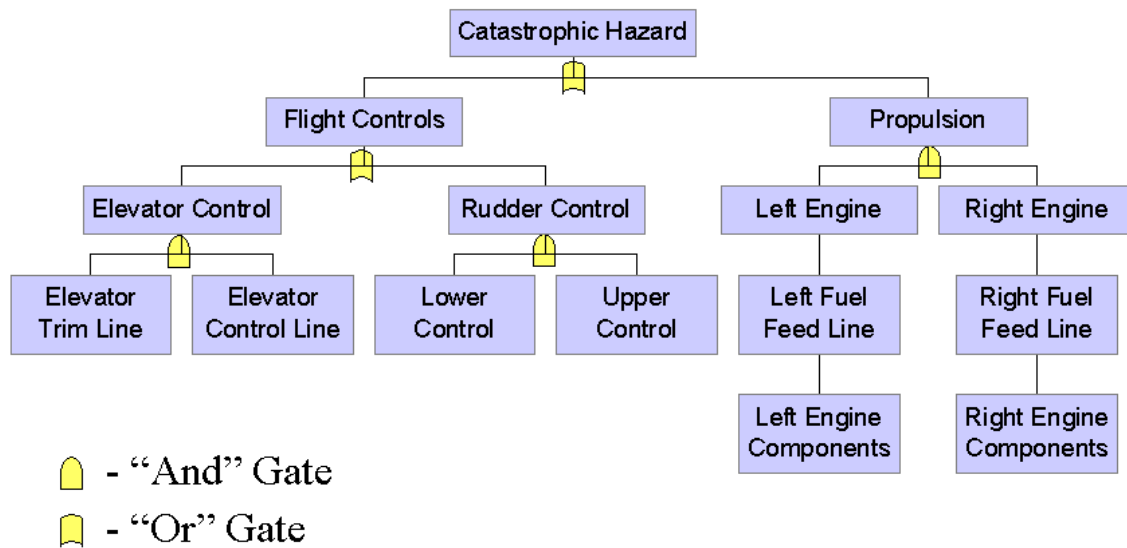


Figure D-6. Sample Fault Tree (GBJ for V1 to V1 + 30 seconds).

A conservative approach was taken for these analyses, and most critical components hit by fragments with a velocity greater than 2 ft/s were considered dysfunctional ($P_{\text{haz/h}} = 1.0$). Larger structural components that were unlikely to be damaged by small fragments were defined in a second PK table that specified that only debris fragments of 7000 grains (1 lb) or larger would result in component dysfunction ($P_{\text{haz/h}} = 1.0$).

OUTPUT

UEDDAM allows the user to select up to eight output files, as listed below.

1. Hazard Summary—The hazard summary file contains the average P_{haz} of all the iterations for each release angle and hazard level (or flight phase), the average P_{haz} for all the release angles swept by UEDDAM, and the average P_{haz} over 360° of sweep of the release angles. Any angle not swept by UEDDAM is considered to have a P_{haz} of 0. The two averages will be the same if all 360° of possible release angles are swept.
2. Hit Summary—The hit summary file contains the component contributions to P_{HAZ} . The summary contains two averages. The first set of component contribution to P_{haz} values are for the release angles swept by UEDDAM. The second set of values assume that any angles not swept by UEDDAM have a P_{HAZ} of zero and calculate the component contributions for a 360° release angle sweep.

3. Detail File—The detail file is similar to the hit summary file in that it contains the component contributions to P_{HAZ} . However, the component contributions to P_{HAZ} for each iteration are written to this file.
4. Visualizer File—This file is used internally by the UEDDAM Visualizer V1.0 and contains shotline information.
5. Intercept File—This file contains component translational and forward and aft spread risk angles for each component. The file contains comma-separated values and is easiest to understand if opened in Excel. These intercept angles are calculated independently of any of the shotlines.
6. Trajectory File—This file contains fragment release and shotline orientation data. The file also lists the release point (x,y, and z-location), and ray orientation (yaw and roll) for each shotline.
7. Error File—The error file contains warnings and errors output by UEDDAM.
8. Debugging File—The debugging file contains all the temporary files generated by UEDDAM during a run. This file was intended to be used only for troubleshooting.

Because of the number of individual runs required to complete a performance run, only the hazard summary and hit summary files are output in these cases. The other output files are useful in troubleshooting and for closer inspection of specific cases.

Each individual run will result in a hazard and hit summary file. Each of the hazard summary files contains a P_{haz} for that combination of rotor, flight phase, engine, and debris type. At the end of each hazard summary is the average P_{haz} over a 360° sweep of release angles for each flight phase. A flight phase failure distribution weighted average of the flight phase P_{haz} was calculated for each run resulting in the average P_{haz} for a given rotor and debris type over a full flight envelope. This was then averaged for each engine, and finally, the two engine averages were combined to produce the aircraft 1-in-20 values.

The hit summary contains a list of components and multiply vulnerable groups (systems of redundant components), the average P_{haz} for each of those components or groups, and the SYSTEM P_{haz} for each run. The P_{haz} for each component or group is referred to as the component contribution to the SYSTEM P_{haz} . The SYSTEM P_{haz} is the probability of catastrophic hazard for the run, the same as the P_{haz} over 360° release that is reported in the hazard summary file. The sum of all of the component contributions to the P_{haz} can sometimes be greater than the SYSTEM P_{haz} . This difference occurs because UEDDAM uses the incremental method for determining P_{haz} along a shotline. This means that the P_{haz} along a shotline will not exceed 1.0 even if more than one critical component is intersected by that shotline. However, each of the critical components intersected by the

shotline will be assigned a P_{haz} . So, if two or more critical components lie on the shotline, the sum component contributions may exceed the calculated SYSTEM P_{haz} .

For the purposes of this analysis (and to be consistent with AC 20-128A guidance), the component dysfunction, given a hit, for all components was 1.0. Therefore, the SYSTEM P_{haz} for that iteration will only be affected if that component was the only other critical component on that shotline. Therefore, it is important to look at both the system (overall) P_{HAZ} and component P_{haz} for each configuration.

INPUT AND OUTPUT FORMATTING

In addition to UEDDAM, several other programs were used to reduce the user workload. Input and output formatting programs were created to automate the process of creating the UEDDAM control files and to collect the data from the various output files. These input and output programs were required due to the number of cases that needed to be executed for a production run. For the GBJ, three programs were used. The first program was an input generation program that created all the file structure and control files required for a production run as well as input decks for the output program. The second program was a batch file that was created by the input generation program to execute UEDDAM for each of the control files it created. The third program used an input deck created by the file generation program to open each output file and copy the probability of hazard for each combination of flight phase, debris category, engine, and rotor, and write it to a table. The values from the completed tables were then used to perform the 1-in-20 analysis, as specified in the AC 20-128A, appendix 1. A similar process was also used for the GT analysis.

ANALYSIS RESULTS

The following section discusses the results of the UEDDAM analysis. Due to the increased size and complexity, fewer trade studies and investigations were performed on the GT than the GBJ.

GENERIC TWIN

Analysis Results

The results of the first analysis (called original) were presented at the 28th PPIHWG meeting. The participants recommended a series of modifications to the GT to highlight specific failure characteristics and better represent in-service aircraft. The primary changes in the model geometry were to lower the engines relative to the fuselage to address cross engine debris, and to modify the wing spar criticality to include the damage tolerance built into the wing spar and stringer structure. In addition to aircraft modifications, the PPIHWG participants recommended a change in the debris categories assessed and the resulting analysis was referred to as the updated baseline analysis. A comparison by debris type of the 1-in-20 results for the original and baseline analyses can be seen in Figure D-7.

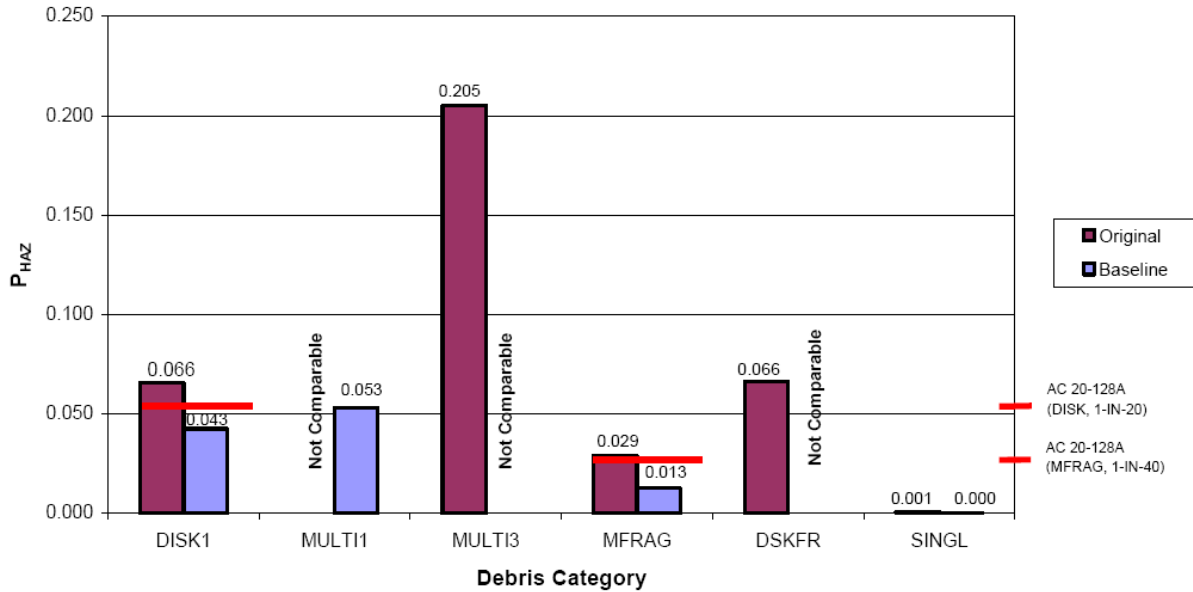


Figure D-7. Comparison of Overall Averages between Original and Baseline Results.

Note that for both the DISK1 and MFRAG debris categories, the original overall averages exceeded the AC 20-128A criteria for acceptable risk levels for similar debris types (see Table D-4), shown as red lines in Figure D-7. After modifications were made to the generic twin engine aircraft, the baseline overall averages of the aforementioned debris categories decreased to below the acceptable risk level. The specific reason for this reduction is discussed further on.

TABLE D-4. Summary of Acceptable Risk Level Criteria
(Extraction of table 1, AC 20-128A)

Requirement	Criteria
Average 1/3 Disk Fragment	1 in 20 (0.050)
Average Intermediate Fragment	1 in 40 (0.025)
Average Alternate Model	1 in 20 @ $\pm 5^\circ$ Spread Angle (0.050)
Multiple Disk Fragments	1 in 10 (0.100)
Any Single Fragment (Except for Structural Damage)	2 \times corresponding <u>average</u> criterion

Figures D-8 through D-19 provide additional details of the results organized by debris category—the odd numbered figures show the results by engine section (averaged over both engines), whereas the even numbered figures show the main contributors to catastrophic damage (averaged over all engine sections and both engines). It is important to note that for the debris categories DISK1, MFRAG, DSKFR, and MULTI3 (see Figures D-8, D-10, D-14, and D-16), the turbine section of the engine contributes the most to the overall probability of catastrophic damage in the original analysis due to the location of the turbine section with respect to the leading-edge spar. The debris zone of the turbine section intersects the wing leading-edge spar (the original analysis had the wing leading-edge spar as critical and singularly vulnerable). This also explains the dramatic drop in the turbine section contribution between the original and baseline results for the DISK1 and MFRAG debris categories. The increased redundancy of the wing structure greatly reduces the impact of the turbine section debris. For the debris categories SINGL and MULTI1, it is the fan section that makes the largest contribution (see Figures D-11 and D-17). In fact, for both the original and the baseline analyses, the only significant contributor to SINGL debris damage is from the fan section. This is because the fan section is the only section that can produce a large enough fragment to defeat critical components.

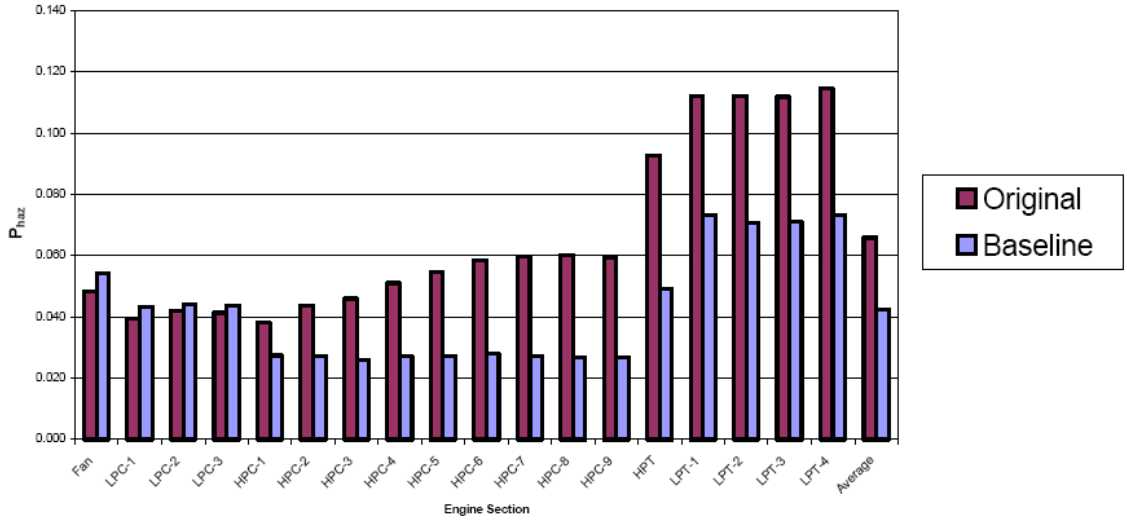


Figure D-8. Comparison between Original and Baseline Results by Engine Section for DISK1.

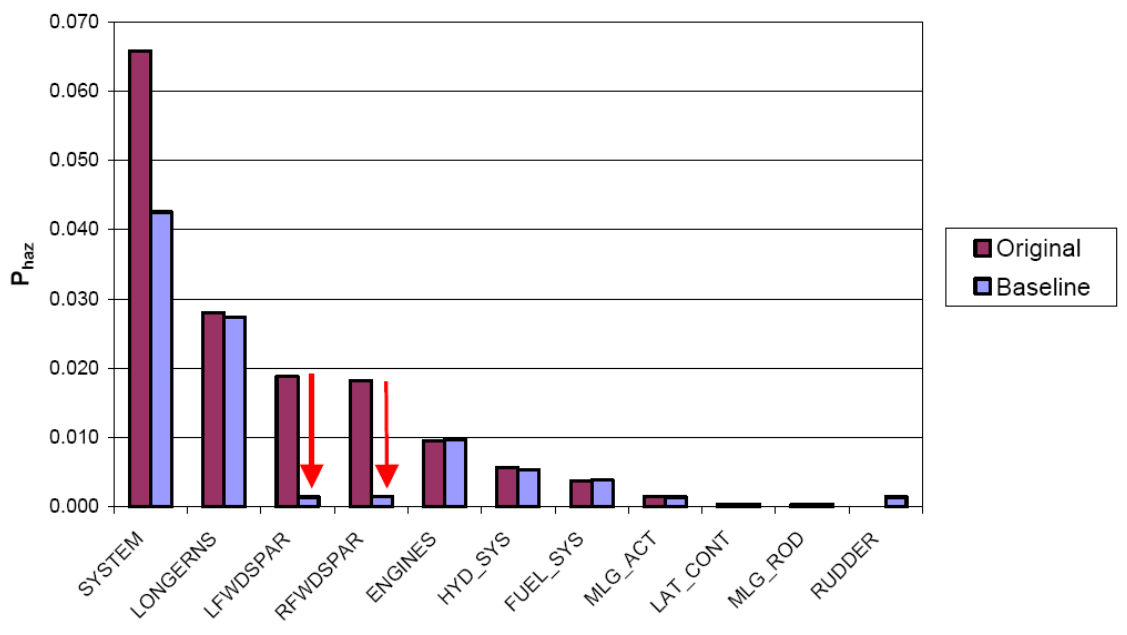


Figure D-9. System Contribution Comparison between Original and Baseline Results for DISK1.

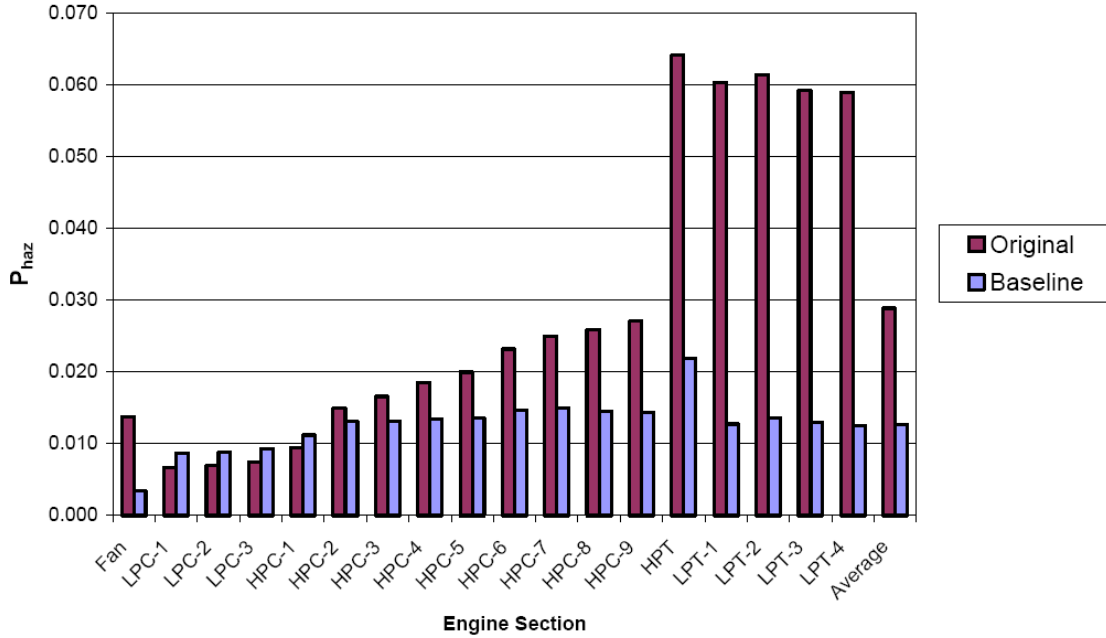


Figure D-10. Comparison between Original and Baseline Results by Engine Section for MFRAG.

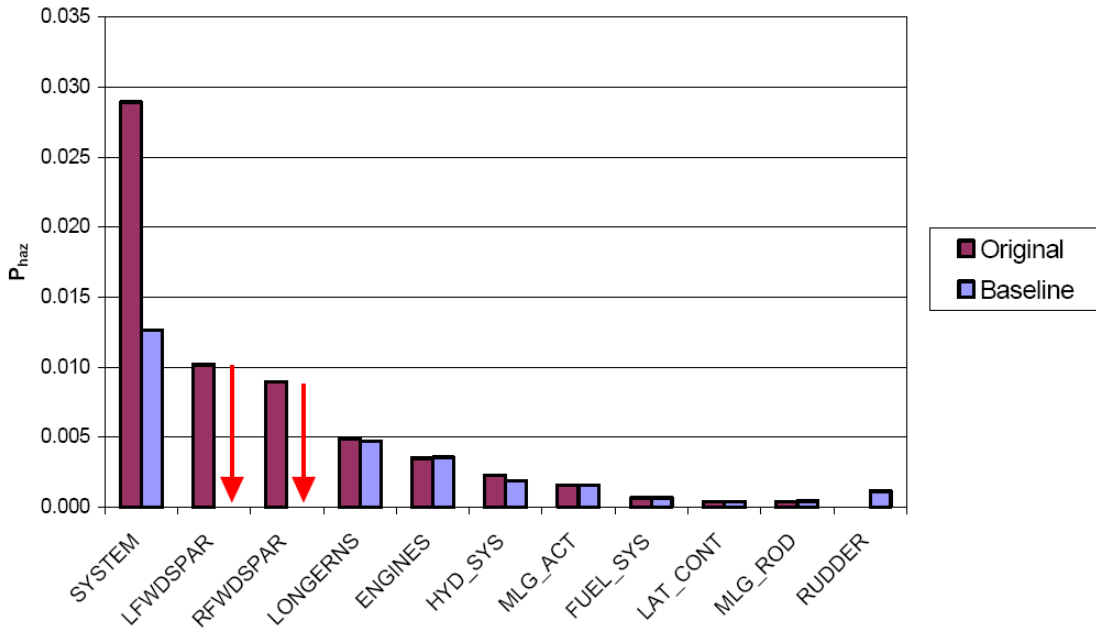


Figure D-11. System Contribution Comparison between Original and Baseline Results for MFRAG.

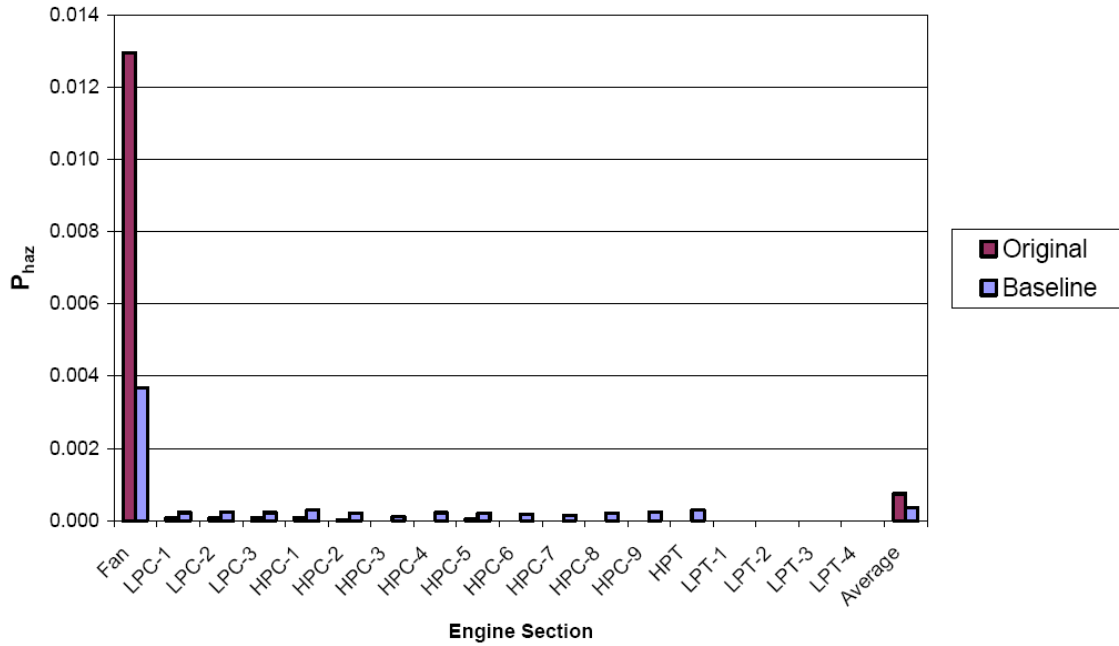


Figure D-12. Comparison between Original and Baseline Results by Engine Section for SINGL.

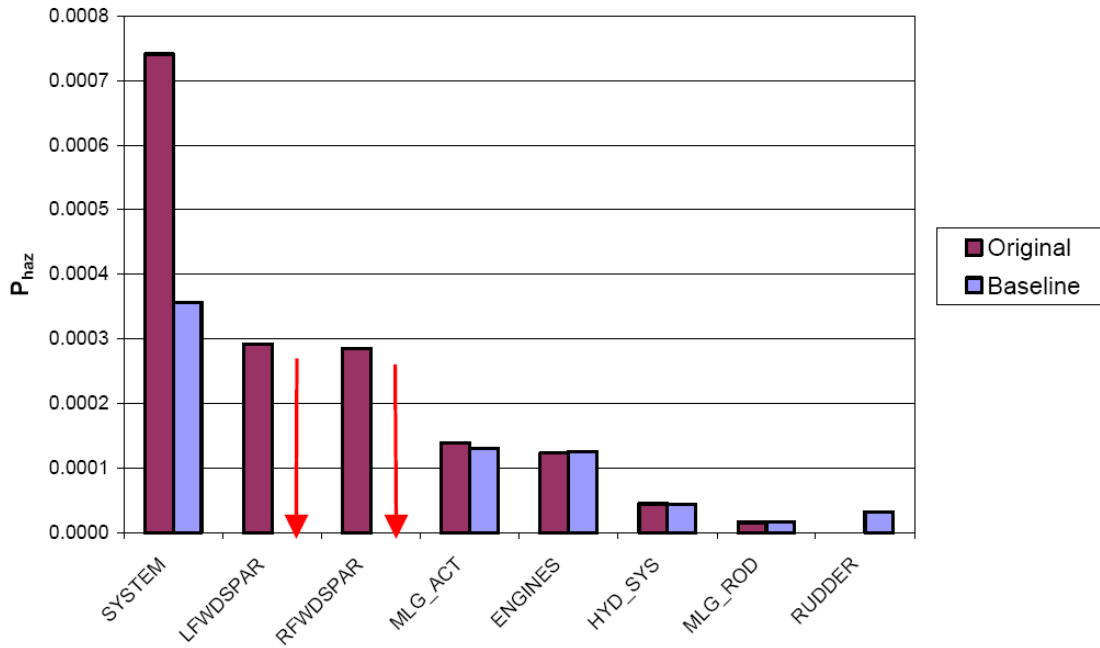


Figure D-13. System Contribution Comparison between Original and Baseline Results for SINGL.

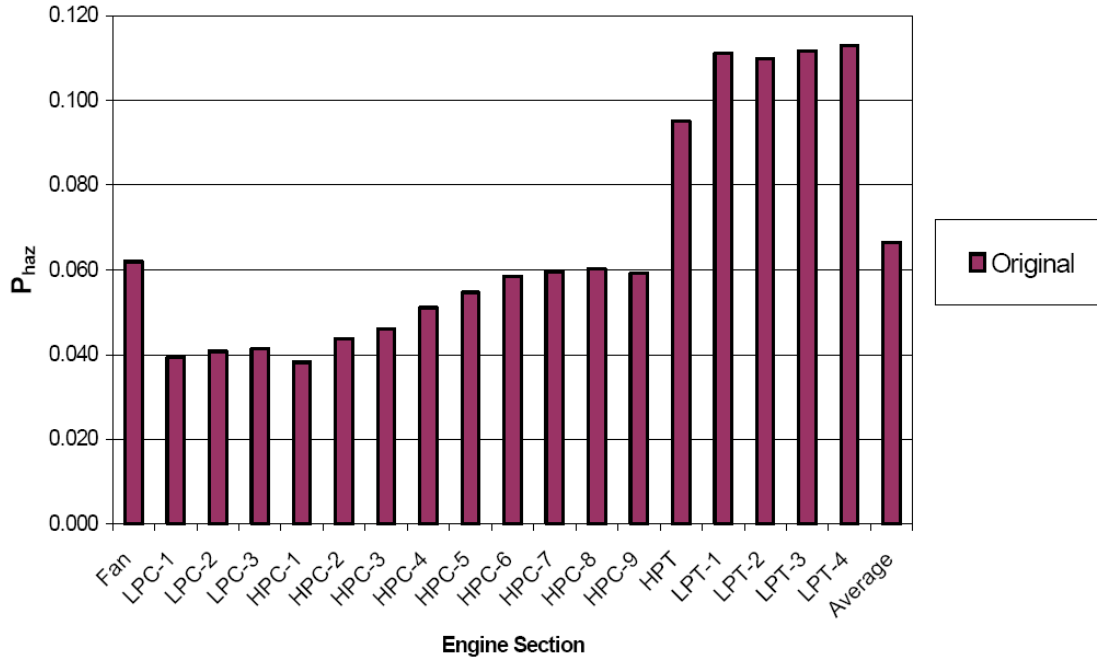


Figure D-14. Original Results by Engine Section for DSKFR.

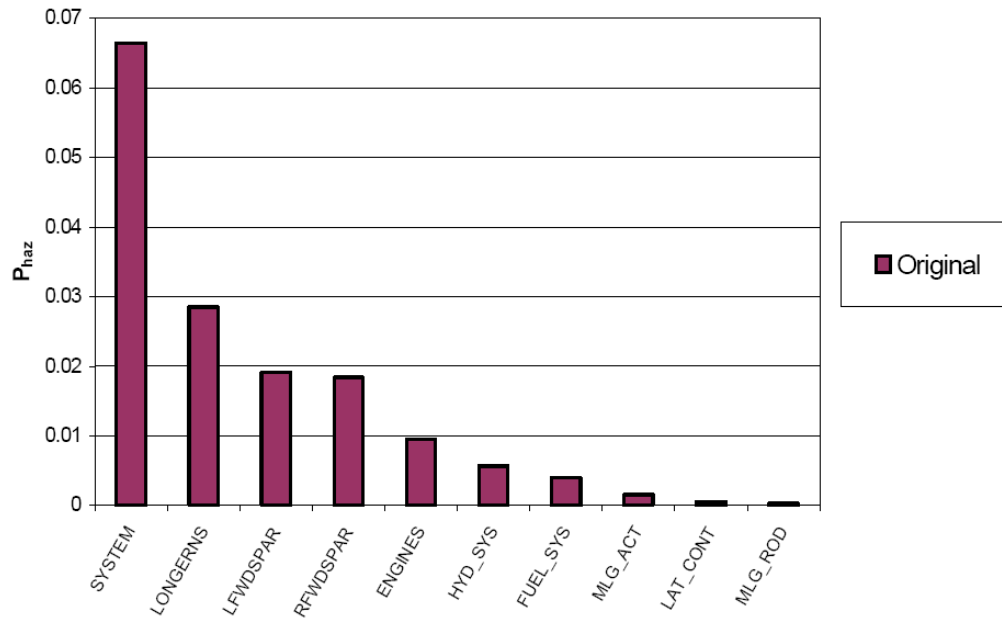


Figure D-15. Original System Contribution Results for DSKFR.

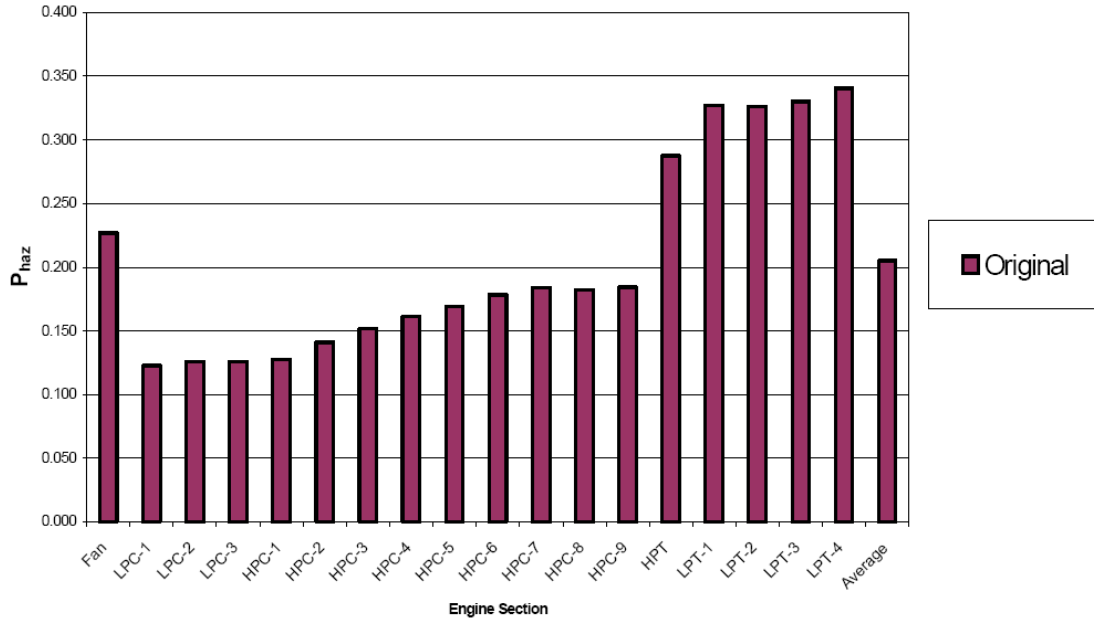


Figure D-16. Original Results by Engine Section for MULTI3.

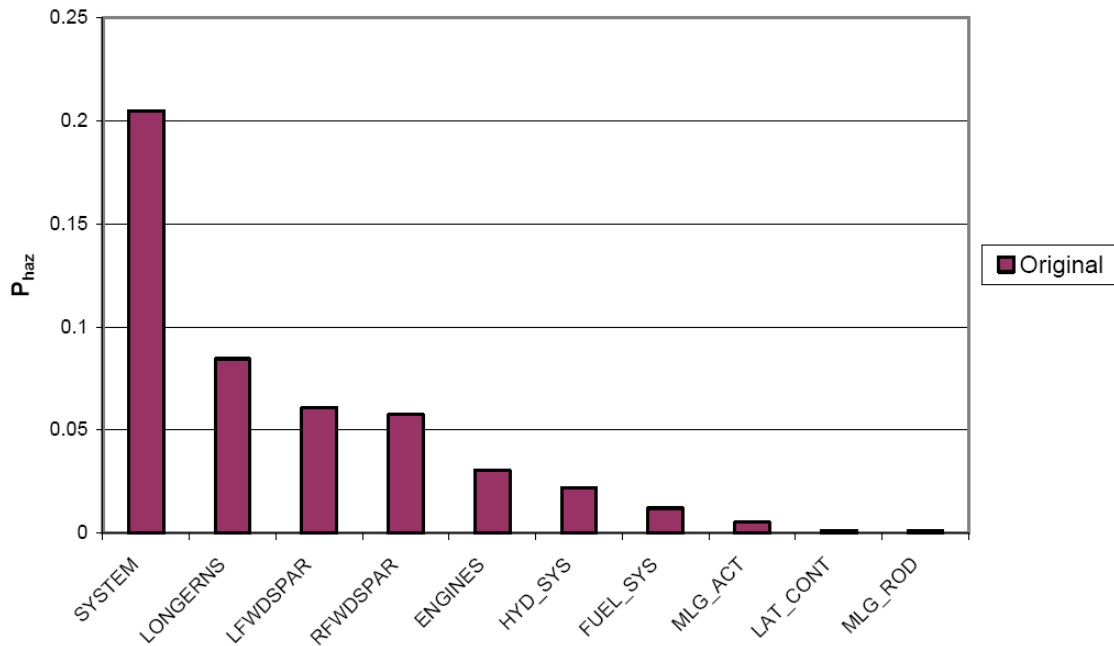


Figure D-17. Original System Contribution Results for MULTI3.

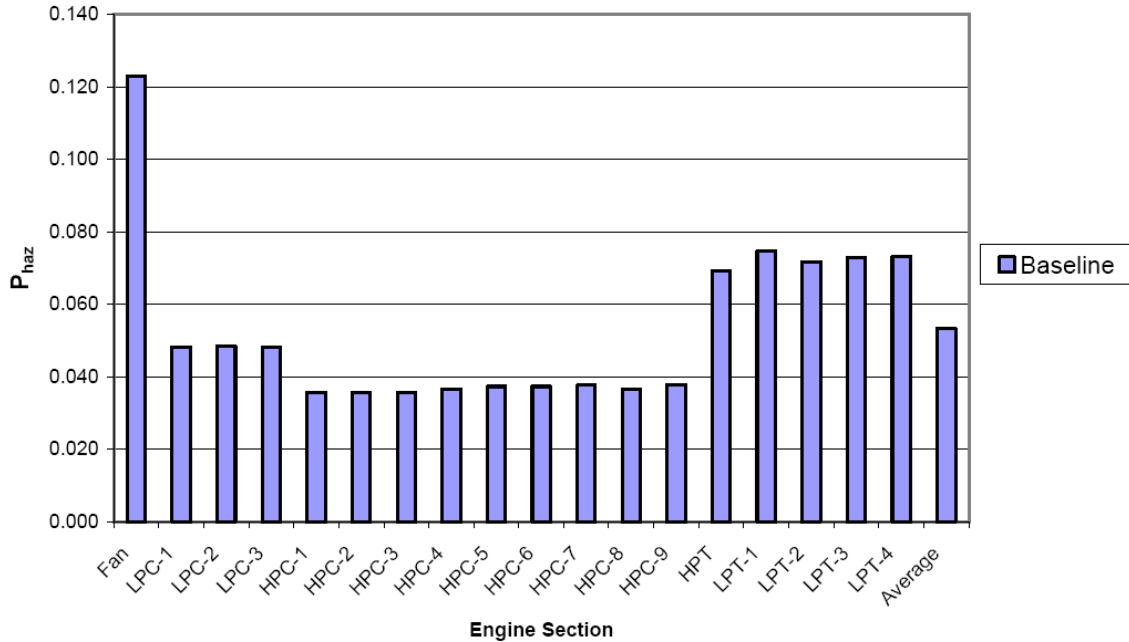


Figure D-18. Baseline Results by Engine Section for MULTI1.

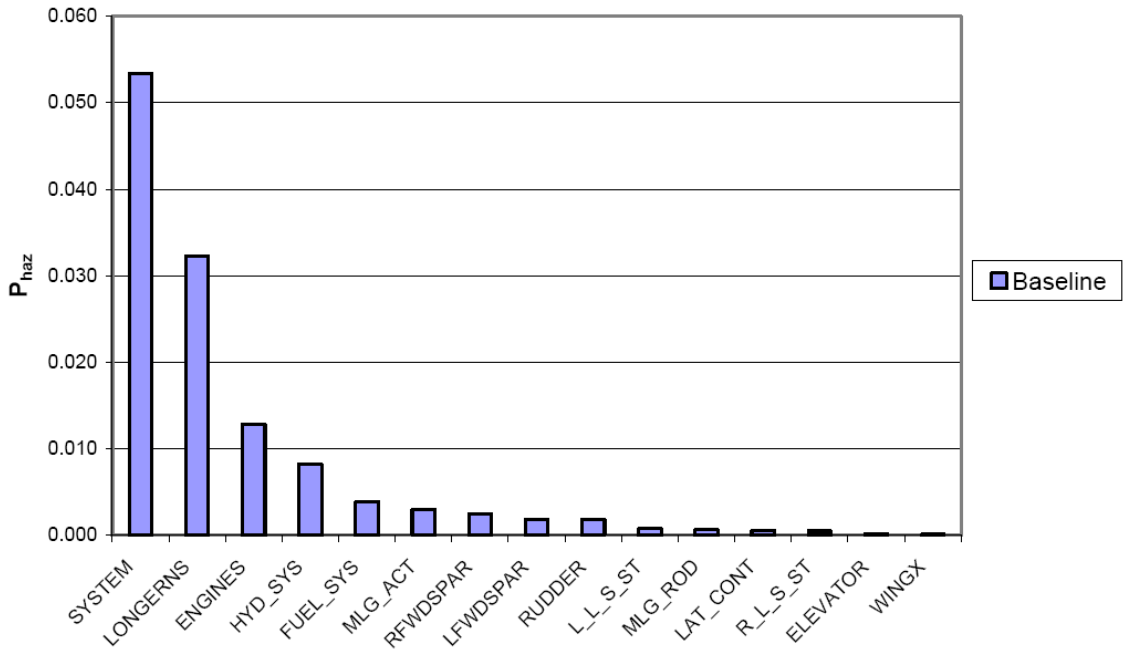


Figure D-19. Baseline System Contribution Results for MULTI1.

For the three comparable debris categories (DISK1, MFRAG, and SINGL), the decline from the original to the baseline results can be explained by the change in wing spar criticality (see Figures D-8, D-10, and D-12). In the original analysis, the left and right forward spars were defeated when a large fragment penetrated either the leading-edge or trailing-edge spar web. In the baseline analysis, however, for the left and right forward

spars to be defeated, not only would the spar web have to be penetrated, but the top and bottom spar caps as well. In addition, certain combinations of failures of upper and lower stringers could cause catastrophic damage. Thus, by increasing the redundancy of the wing spar (and thereby changing its criticality), the overall probability of catastrophic damage was reduced.

Note also from Figures D-8, D-10, and D-12 that the rudder shows an increase in contribution to catastrophic damage. This is due to the fact that for the baseline condition, the loss of primary rudder control alone on takeoff would be sufficient to result in catastrophic damage whereas in the original analysis, it was assumed that, on takeoff, a loss of both primary and trim rudder control would be required for a catastrophic event. Also note that even though the wing fuel tanks became critical in the baseline analysis, no change resulted in their contribution. This results from the engines being far enough forward of the wing fuel tanks that the debris misses the tanks in almost all cases. Also, the impact of any potential hits on the wing fuel tanks are further diminished by the phase failure distribution of the descent phase (3%).

In addition to the modifications from the original configuration to the baseline configuration, a parametric study (referred to as the alternate analysis) in which the engines were lowered from the baseline configuration such that the engine centerline was level with the bottom of the aircraft skin was performed to address cross engine debris under the fuselage. This adjustment was made to achieve a GT that represents a broader group of in-service aircraft. A comparison by debris category of the results from the baseline and alternate analyses can be seen in Figure D-20.

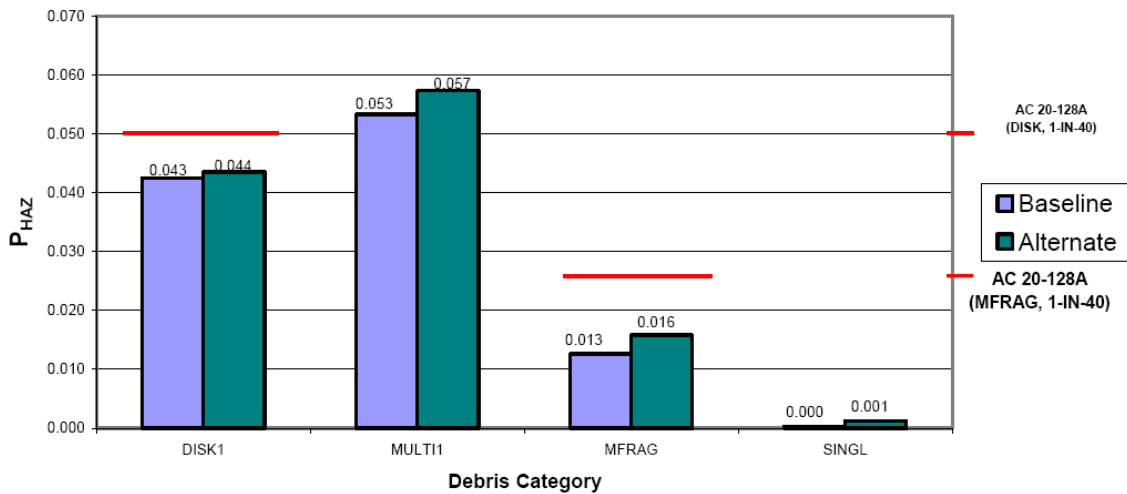


Figure D-20. Comparison of Overall Averages between Baseline and Alternate Results

The red lines in Figure D-20 again represent the AC 20-128A levels of maximum allowable probability of catastrophic hazard (see Table D-4) for the debris categories DISK1 and MFRAG. Note that lowering the engines increases the overall probability of

catastrophic damage for each debris category assessed, but not enough to raise the probabilities to an unacceptable risk level.

As before, Figures D-21 through D-28 provide further details of the results organized by debris category; the even figures show the results by engine section (averaged over both engines), whereas the odd figures show the main contributors to catastrophic damage (averaged over all engine sections and both engines). Since all debris categories assessed were identical, the results for the baseline and alternate analyses are always displayed on the same graph. It is important to note that for the debris categories DISK1 and MFRAG, the turbine section of the engine (specifically, the low-pressure turbine for DISK1 and the high-pressure turbine for MFRAG) contributes the most to the overall probability of catastrophic damage. For the DISK1 debris categories, the low-pressure turbine section debris is large enough to defeat the leading-edge spar in some cases. For the MFRAG debris category (and the DISK1), the high-pressure turbine section debris has enough size and energy to defeat redundant structure. For the debris categories SINGL and MULTI1, it is the fan section that makes the largest contribution (see Figures D-22, D-24, D-26, and D-28). However, contrary to the original and baseline analyses, in the alternate analysis, the fan section is not the only significant contributor to SINGL debris damage—the compressor section (including both the high- and low-pressure sections) and the high-pressure turbine play a nontrivial part as well due to the cross-engine debris effects.

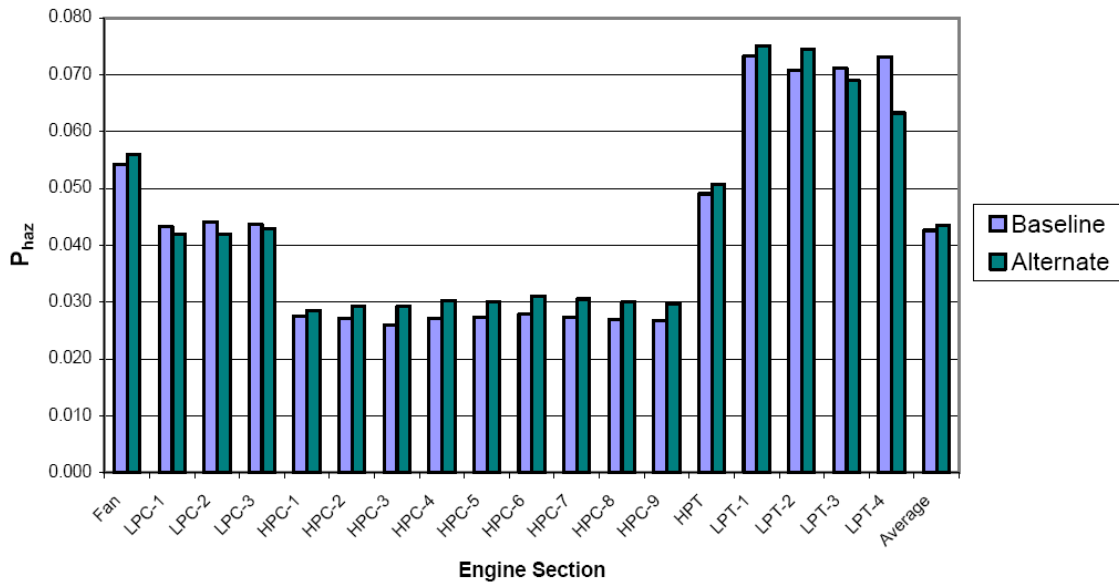


Figure D-21. Comparison between Baseline and Alternate Results by Engine Section for DISK1.

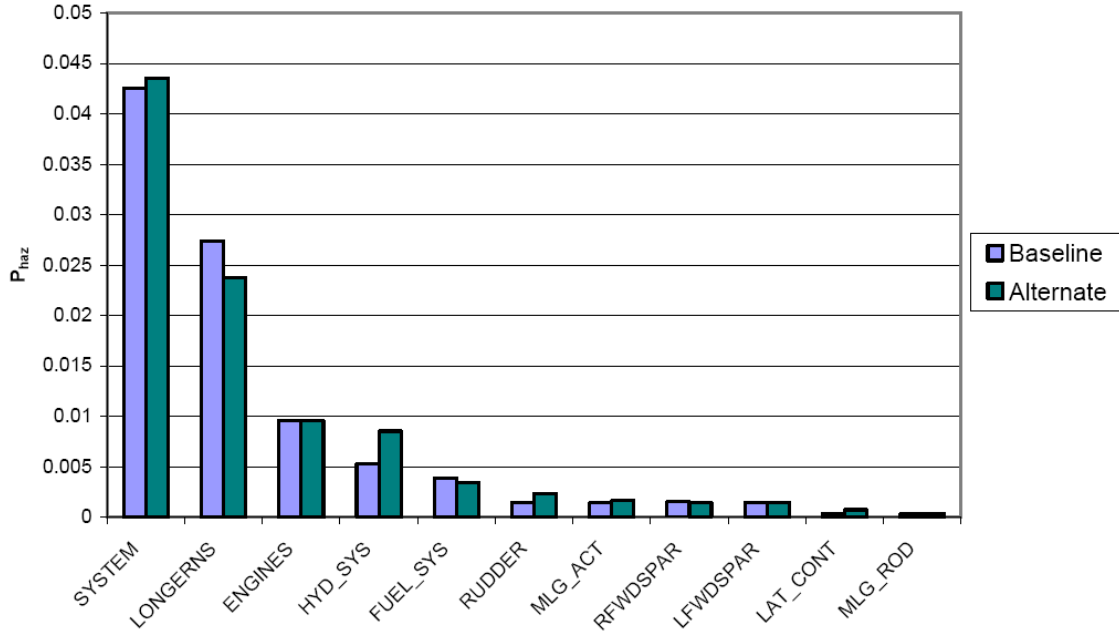


Figure D-22. System Contribution Comparison between Baseline and Alternate Results for DISK1.

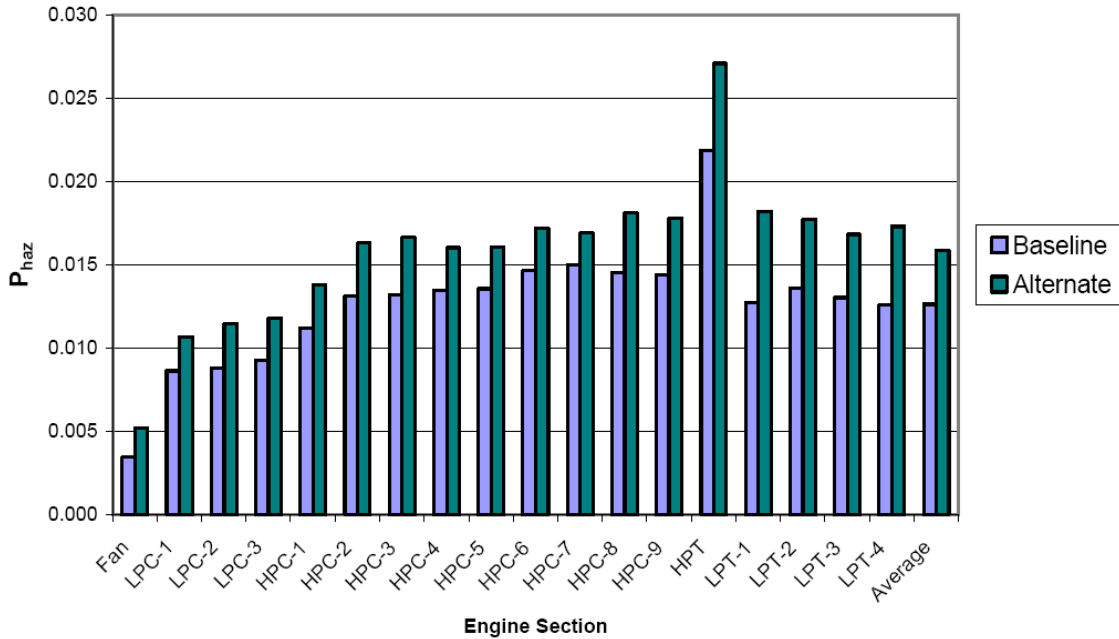


Figure D-23. Comparison between Baseline and Alternate Results by Engine Section for MFRAG.

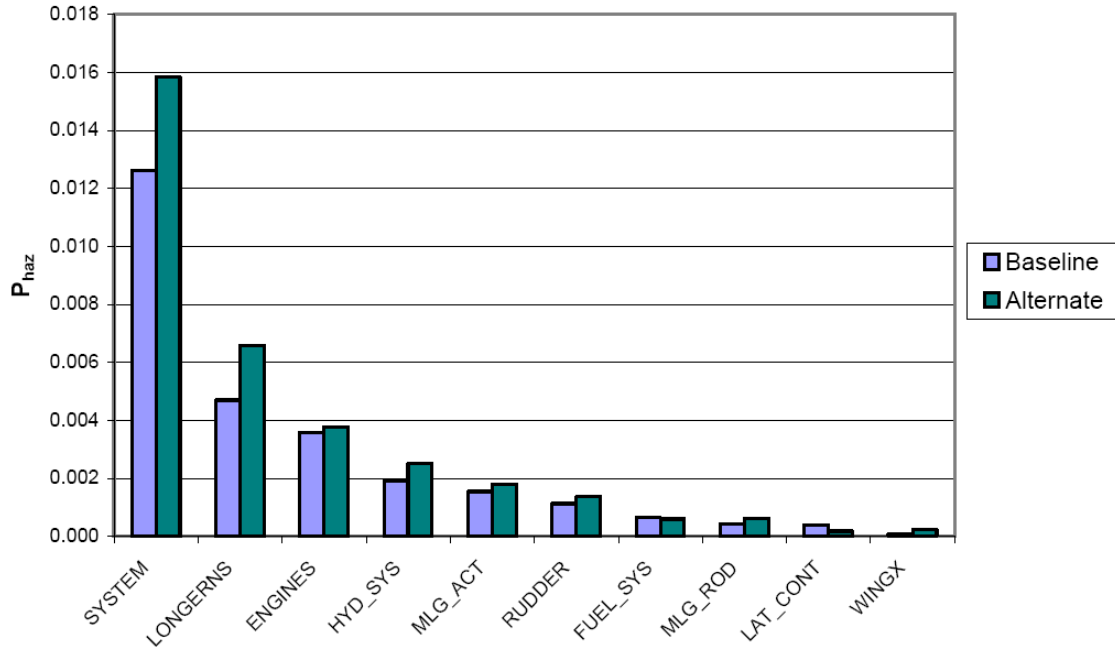


Figure D-24. System Contribution Comparison between Baseline and Alternate Results for MFRAG.

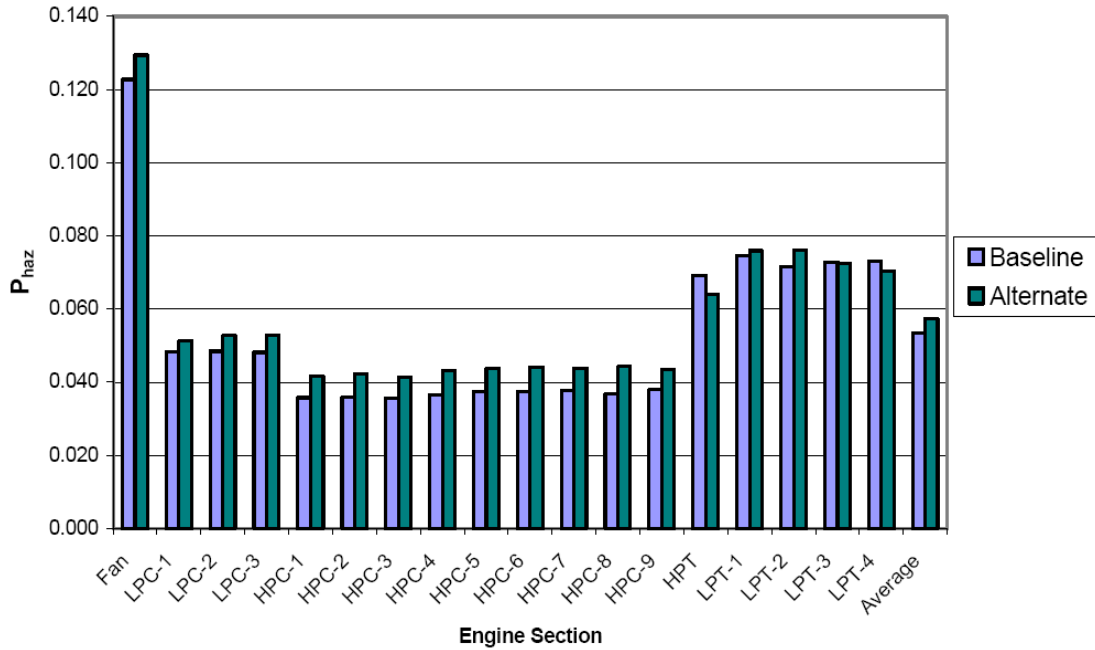


Figure D-25. Comparison between Baseline and Alternate Results by Engine Section for MULTI1.

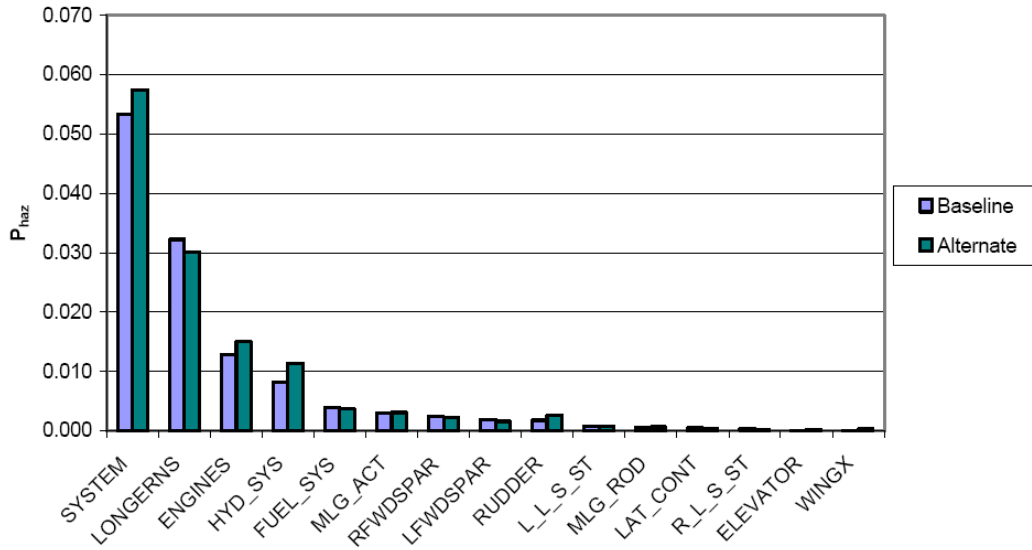


Figure D-26. System Contribution Comparison between Baseline and Alternate Results for MULTI1.

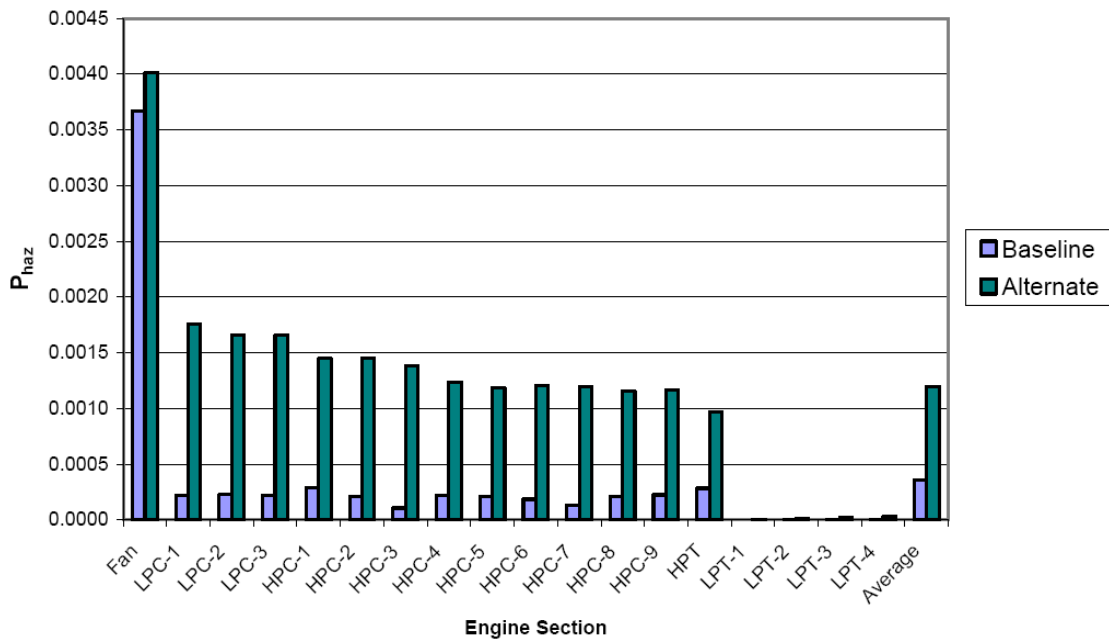


Figure D-27. Comparison between Baseline and Alternate Results by Engine Section for SINGL.

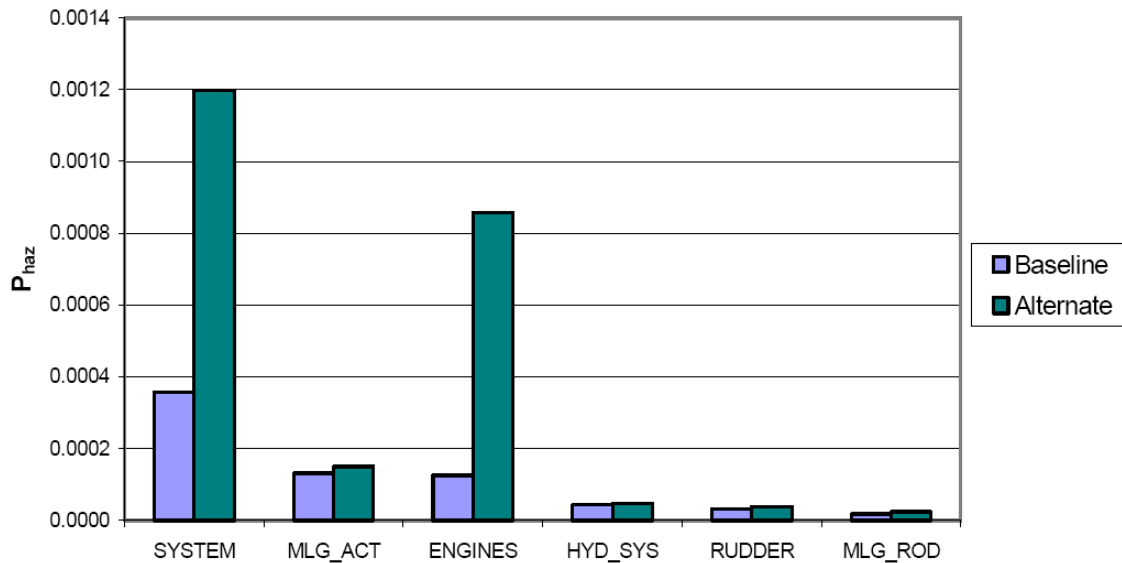


Figure D-28. System Contribution Comparison between Baseline and Alternate Results for SINGL.

Figures D-22, D-24, D-26, and D-28 show the baseline and alternate system contributions to catastrophic damage for the DISK1, MFRAG, MULTI, and SINGL debris categories. The DISK1, MFRAG, and MULTI system contributions are very similar. Specifically, the top three contributors for all three debris types are the LONGERNS (defeat of five adjacent fuselage structural longerons), ENGINES (defeat of opposite engine via cross engine debris), and HYD_SYS (defeat of redundant components in the hydraulic systems). With LONGERNS being the leading contributor to catastrophic damage, a thorough investigation of this assumption is warranted. The results indicate that for the larger debris fragments, the aircraft fuselage structural defeat is a major issue. Note that this contribution is based on the assumption that defeat of five adjacent fuselage structural longerons would result in catastrophic damage in all flight modes. The second leading contributor for the DISK1, MFRAG, and MULTI debris categories is the defeat of the opposite engine via cross engine debris. The contribution of ENGINES changes little from the baseline to the alternate aircraft configuration because the larger fragments have sufficient energy to penetrate through fuselage and damage opposite engine. The hydraulics system (HYD_SYS) was the third highest contributor to P_{HAZ} . This indicates that the arrangements of the hydraulic system components align with respect to some of the debris trajectories. For the SINGL debris category (see Figure D-28) the leading contributors are the MLG_ACT (critical elements of the main landing gear actuator), ENGINES, and HYD_SYS. For the alternate configuration, the ENGINES become the leading contributor revealing the sensitivity of cross engine defeat based on engine location for the small fragments.

The increase from the baseline to the alternate results is manifested as increases in different system contributions, depending on the debris category considered (see Figures D-22, D-24, D-26, and D-28). For instance, when considering the debris category

MFRAG, increases in every system (excluding the fuel and lateral flight control systems) contribute to the overall increase in P_{HAZ} from baseline to alternate results. The debris category DISK1, on the other hand, shows a primary increase in contribution from the hydraulic system, but is offset by a decrease in the fuselage structure contribution from baseline to alternate results. Similarly, the MULTI1 debris category follows the DISK1 trend with the exception that the small fragments released in the MULTI1 case additionally increase the cross engine defeat probability (ENGINES) for the alternate configuration. Lastly, for the debris category SINGL, the overall probability of catastrophic damage increase is concentrated in a nearly eight-fold increase in the engine system's contribution, which is a direct result of exposing the engines to cross-engine debris. The reason that there is not a dramatic increase for the other debris types is that the other debris types have enough energy to penetrate through the aircraft body and defeat the other engine in the baseline configuration, whereas the SINGL debris does not. Exposing the opposite engine has little impact for the other debris types, but allows the smaller and lower energy fragments of the SINGL debris category to reach and damage the opposite engine. The turbine blades (in the SINGL debris category) do not contribute much to the overall P_{HAZ} of the system for several reasons (see Figure D-27). This is primarily because the relatively small size and energy of the turbine blade fragments. Also, the turbines are located in the aft section of the engine and the aft spread angle tends to be further aft than other sections increasing the likelihood of the fragments missing the opposite engine.

Sensitivity Study

The purpose of the sensitivity study was to assess the responsiveness of the P_{HAZ} to an increase in engine nacelle thickness using UEDDAM. The study was performed with the alternate geometric configuration and took into consideration three debris categories (MULTI1, DISK1, and SINGL). The nacelle thickness considered in this study were selected by estimating the thickness of aluminum required to stop all the large and small fragments making up the debris categories MULTI1 and SINGL and applying these skin thicknesses to the areas indicated in red in Figure D-29. The equation used in estimating these thicknesses was the COVART fragment penetration equation, which is founded on FAA equations that have been modified by China Lake (based on engine debris penetration testing). For example, using the fragment penetration model, the residual velocity of the large fan fragment in the debris category MULTI1 can be found to be equal to zero after it penetrates a 0.48" aluminum plate. Thus, in addition to the original 0.04" thickness, the sensitivity study included both the upper and lower limits of 1.1" (corresponding to the large HPT fragment) and 0.12" (corresponding to the small LPT fragment) as well as a midpoint thickness of 0.48".

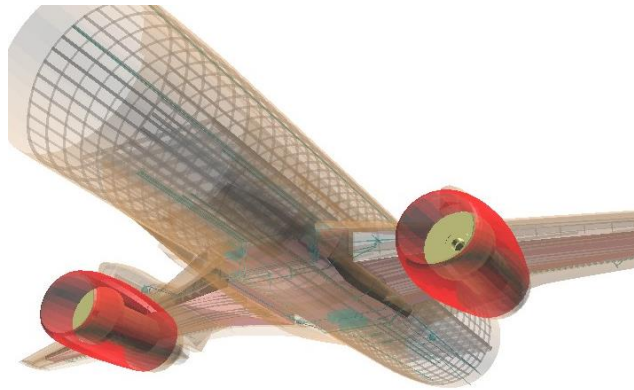


Figure D-29. Alternate Geometric Configuration with Engine Nacelles Highlighted in Red.

Analysis

A summary of the results of the sensitivity study can be seen in Figure D-30. As expected, by increasing the engine nacelle thickness, the probability of catastrophic damage is reduced, but the decrease does not occur in a linear fashion. An increase 0.09" from the original 0.04" thick aluminum was seen to have a limited effect in reducing the probability of catastrophic damage due to MULTI1, however, the increase in nacelle thickness to 0.48" reduces the P_{HAZ} to 0.6%, well below the AC 20-128A 1-in-20 requirement for a single 1/3 disk segment (see Table D-4). Although the MULTI1 debris category contains multiple fragments in addition to a disk segment, this analysis shows that the GT can meet a 1-in-20 requirement to a more damaging debris field. Because the effects of increasing the skin thickness are nonlinear, more data points between these two nacelle thickness should be performed for a better evaluation of the optimum nacelle thickness to reduce damage from larger fragments.

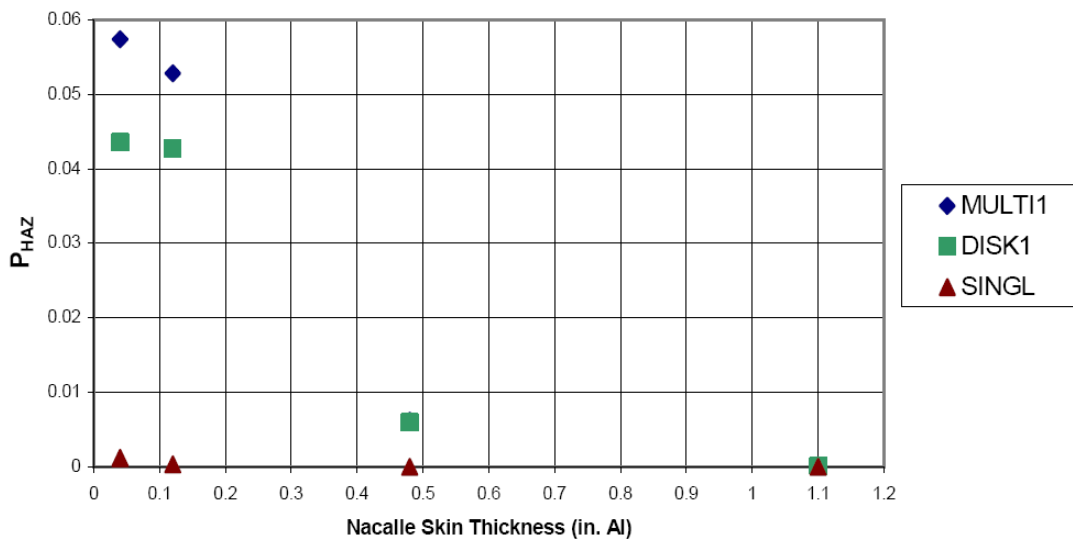


Figure D-30. Probability of Catastrophic Damage (P_{HAZ}) Vs. Nacelle Skin Thickness for Three Debris Types.

One notable result was seen for the HPT section of the 1.1" nacelle thickness case. The DISK1 probabilities for catastrophic damage for the HPTs of engines 1 and 2 were equal to zero, but the same probabilities for the debris category MULTI1 were slightly greater than zero. At first this appears counterintuitive because the only difference between the two debris types is the addition of small fragments, which should not be able to perforate the 1.1" engine nacelle. However, upon closer inspection, it was determined that the HPT hazard zone for the debris category MULTI1 (highlighted in blue in Figure D-31) extends beyond the aft end of the engine nacelle and therefore, the shotlines for some of the small fragments were not shielded by the thickened nacelle skin. Note that had the SINGL UEDDAM runs been performed for the 0.48" and 1.1" thicknesses, the same phenomenon would have been noticed since the fore and aft spread angles for the HPT small fragments are the same (+15° and -60°) in both the SINGL and MULTI1 debris categories; however, because the values for the SINGL debris case are almost an order of magnitude smaller than those of the other debris categories, this phenomenon would be insignificant at the scale shown in Figure D-30.

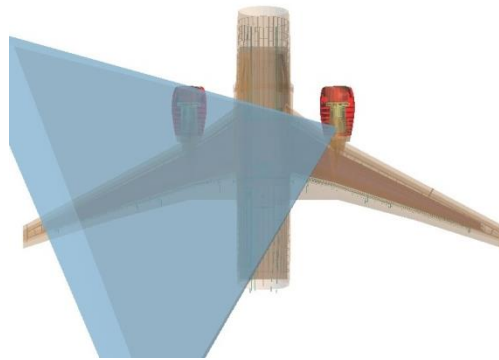


Figure D-31. Engine 1 HPT Hazard Zone for the Debris Type MULTI1

As part of the analysis, a rough estimate of the weight of shielding (i.e. the portion of the engine nacelles that would be thickened) required to protect the critical components of the aircraft from all of the debris categories was calculated. To do this, it was assumed that the engine nacelle could be modeled as a cylinder with a radius of approximately 38 inches and a uniform thickness.

These rough estimates assumed a single thickness of aluminum along the entire length of the engine nacelle. However, tailoring the nacelle thickness to each engine section and/or using more advanced materials (i.e., Titanium, Kevlar, Spectra, S-glass, etc.) could dramatically reduce the weights listed in Table D-5. For example, Table D-6 shows the thickness and weights of titanium (density = 0.163 lb./in³) that would give the same level of protection as those thicknesses of aluminum in Table D-5 against a 1.8 lb fan blade fragment of the MULTI1 debris category. Note that to estimate the net weight increase of each thickness of titanium, you must subtract the weight of the corresponding thickness of aluminum and then multiply by 2. For example, the net weight increase for the 0.064" thickness of titanium is $2 \times (52 - 20) = 64$ lbs.

TABLE D-5. Volumes and Weights for Each Engine of Required Shielding for Different Engine Nacelle Thicknesses

Thickness of Al	0.04 in.	0.12 in.	0.14 in.	0.48 in.	1.1 in.
Volume (in ³)	200	600	700	2400	5500
Weight (lb.)	20	59	69	235	538

TABLE D-6. Equivalent Volumes and Weights of Titanium for each Engine

Thickness Al	0.04 in.	0.12 in.	0.14 in.	0.48 in.	1.1 in.
Weight Al (lb.)	20	59	69	235	538
Thickness Ti	0.02 in.	0.055 in.	0.064 in.	0.22 in.	0.51 in.
Weight Ti (lb.)	15	45	52	180	412

Figure D-32 shows how the system contributions change based on engine nacelle thickness. The analysis results indicate that the contribution to P_{HAZ} from cross-engine debris damage (ENGINES) is slightly minimized by a nacelle thickness of 0.12" but is almost eliminated at 0.48". Somewhere between a nacelle thickness of 0.12" to 0.48" the ENGINES contribution is significantly impacted. This equates to a weight of aluminum of somewhere between 78 and 430 lbs (50 and 320 lbs of titanium, respectively).

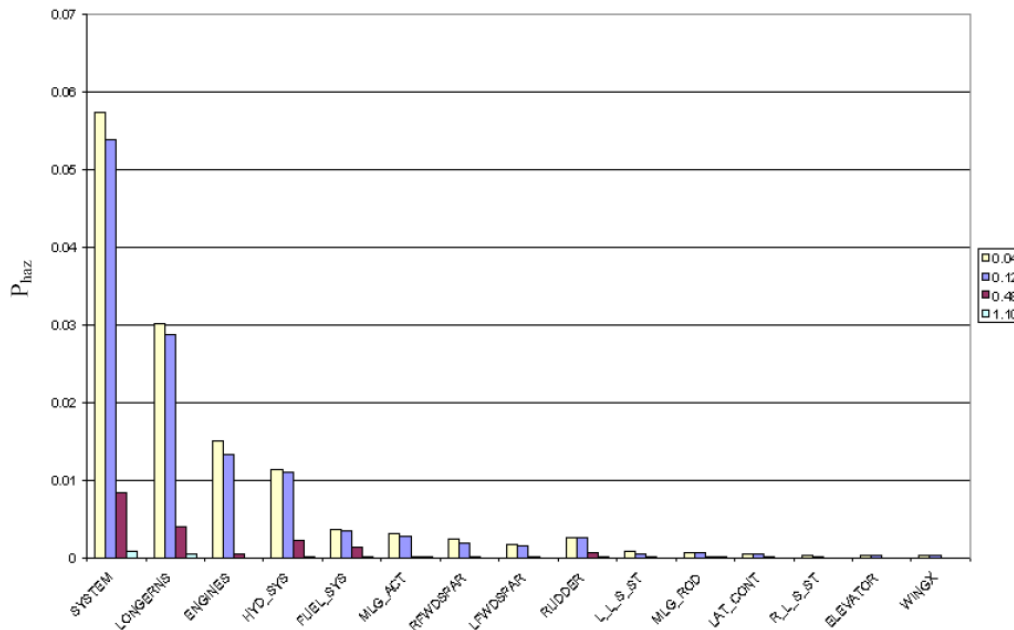


Figure D-32. System Contribution Vs. Engine Nacelle Thickness.

Generic Twin Conclusions

The UEDDAM analysis of the GT shows that the aircraft has an overall average P_{HAZ} due to the single disk segment debris category (DISK1) of 0.043 for the baseline configuration and 0.044 for the alternate (engines lowered) configuration. This is below the AC 20-128A acceptable value of 0.050 (1-in-20) for the single 1/3 disk fragment and for the average alternate model. Note that the DISK1 debris model (see Table D-2) varies the fore/aft spread angle based on the engine section of the release, which is more representative of actual debris distributions than the AC 20-128A models. Also note that the DISK1 debris model covers a larger fore/aft spread than either the single 1/3 segment or the alternate models of AC 20-128A. While the UEDDAM DISK1 model differs slightly from the AC 20-128A models, it is more representative of a real 1/3 disk segment event and addresses the intent of the AC 20-128A models. Therefore, based on the assumptions of this analysis, the GT meets the FAA requirements for the uncontained engine debris risk for the single 1/3 disk fragment and the alternate debris model.

The UEDDAM analysis of the intermediate fragment (MFRAG) shows an overall P_{HAZ} of 0.013 and 0.016 for the baseline and alternate aircraft configurations, respectively. This is well below the AC 20-128A acceptable value of 0.025 (1-in-40) for the intermediate fragment. Note that the MFRAG debris model (see Table D-2) also varies the fore/aft spread angle based on the engine section of the release, which is more representative of actual debris distributions and covers a larger fore/aft spread than the intermediate fragment model of AC 20-128A. In addition, the MFRAG masses exceed 1/30 of the corresponding bladed disk masses, which is the AC 20-128A required size for intermediate fragments. While the UEDDAM MFRAG model differs slightly from the AC 20-128A model, it is more representative of a real intermediate fragment event and addresses the intent of the AC 20-128A model. Therefore, based on the assumptions of this analysis, the GT meets the FAA uncontained engine debris risk requirements for the intermediate fragment.

This study did not investigate the multiple disk fragment model discussed in AC 20-128A; therefore, conclusions as to the ability of the GT to meet the AC 20-128A requirements cannot be drawn without further analysis.

The alternate aircraft configuration (engines lowered) increased the catastrophic risk probabilities for all four debris categories assessed and is, therefore, a less preferred option, although this increase did not exceed any AC 20-128A requirements.

Although the design meets the AC 20-128A requirements, there are some trends in the data that identify areas for improvement to reduce the risk of uncontained engine events. For the DISK1, MULTI, and the MFRAG debris categories, the three leading contributors to catastrophic damage are, the fuselage longerons, engines, and hydraulic system. By toughening the fuselage structure, a great reduction in risk can be achieved. While the engines are the second leading contributor, other than moving the engines farther apart or adding shielding, not much else can be done to reduce their contributions. Rerouting the hydraulic system and adding additional redundancies can achieve a

moderate reduction in risk. The SINGL debris category risk values are extremely low and can be further reduced through shielding. In fact, shielding can potentially lower the risk values for all debris categories, but imposes a cost in terms of weight.

The nacelle skin thickness sensitivity study showed a significant reduction in risk to uncontained engine debris could be achieved through shielding alone. For example, by increasing the engine nacelle thickness from 0.04 to 0.14 inch, even the MULTI1 debris type catastrophic risk value is expected to decrease to the 1-in-20 value. A rough estimate shows that the increase in weight for this reduction would be approximately 100 lbs using aluminum to provide the shielding. By optimizing the nacelle thickness based on engine section, and using advanced armor materials, the weight penalty could be reduced.

GENERIC BUSINESS JET

Convergence

Two approaches were taken in examining the convergence for the GBJ runs. The first examined only the 1-in-20 analysis values. However, as stated previously, the individual results for each case are also significant. Therefore, the fan P_{HAZ} convergence was also examined.

A random seed is specified by the user in the control file. This seed is used by UEDDAM to generate the shotline trajectories and to allow for repeatability by using the same random numbers to generate the shotlines for various runs. The number of shotlines required to conduct an analysis will vary, depending on the size of the hazard zone and the target size. The variance of the P_{HAZ} for a given number of iterations selected was examined based on three production runs, all of which utilized the same configuration and input files, but different random seeds. The random seeds (A) 2501, (B) 7406, and (C) 0003 were used for this comparison. The random seed 2501 was used for all other cases in this analysis and trade study.

The three runs with all parameters except for the random seeds were compared before and after the 1-in-20 averaging was performed. The majority of the cases were within 20%–30% of an equivalent case. A few cases resulted in percent differences as high as 130%–200%. However, the 1-in-20 average (absolute value) percent difference for the aircraft P_{HAZ} was 2.2%, with the largest percent difference in the aircraft P_{HAZ} values for the three cases run being 5.6% (see Table D-7). The percent difference equation used for these comparisons was the difference between the two values divided by the average of the two values. Therefore, the large percent difference in the small fragment cases may be caused by small changes to the P_{HAZ} being more amplified by the percent difference equation than those debris categories with higher P_{HAZ} .

TABLE D-7. Percent Difference for in Aircraft P_H for Runs with Three Different Random Seeds (A, B, and C)

	Seeds A-B	Seeds B-C	Seeds A-C
DISK1	-4.58%	-0.70%	-3.88%
MULTI	-1.14%	-2.20%	1.06%
MFRAG	3.97%	5.57%	-1.60%
SINGL	1.30%	-2.20%	3.50%

Due to the large differences in the rotor level P_{haz} for runs where different random seeds were used, a closer examination was done for a single case. The fan was examined at the V1 to V1+30 phase of flight over 360° (72 release angles), specifying only five iterations per release angle for each DISK1, MULTI, and MFRAG debris category. The results for the 25 iterations (per release angle) runs for the three random seeds used above were also incorporated into the convergence data. The results for the smaller test case to examine convergence matched those of the production run comparison. Twenty-five iterations resulted in a variance of approximately ±20%. The results of the fan-only comparisons indicate that 50 iterations or more result in a variation of less than ±2% for the 1/3-fan disk section and intermediate fragment cases (see Figure D-33).

Because the UEDDAM small fragment debris type was used for the SINGL debris category, the results were examined separately. The runs were performed using the same configuration as the other debris categories, but 20 iterations were specified since the small fragment debris type only uses a single shotline to represent the fragment. The results show that for 120 or more iterations, the variation is less than ±2% (Figure D-34).

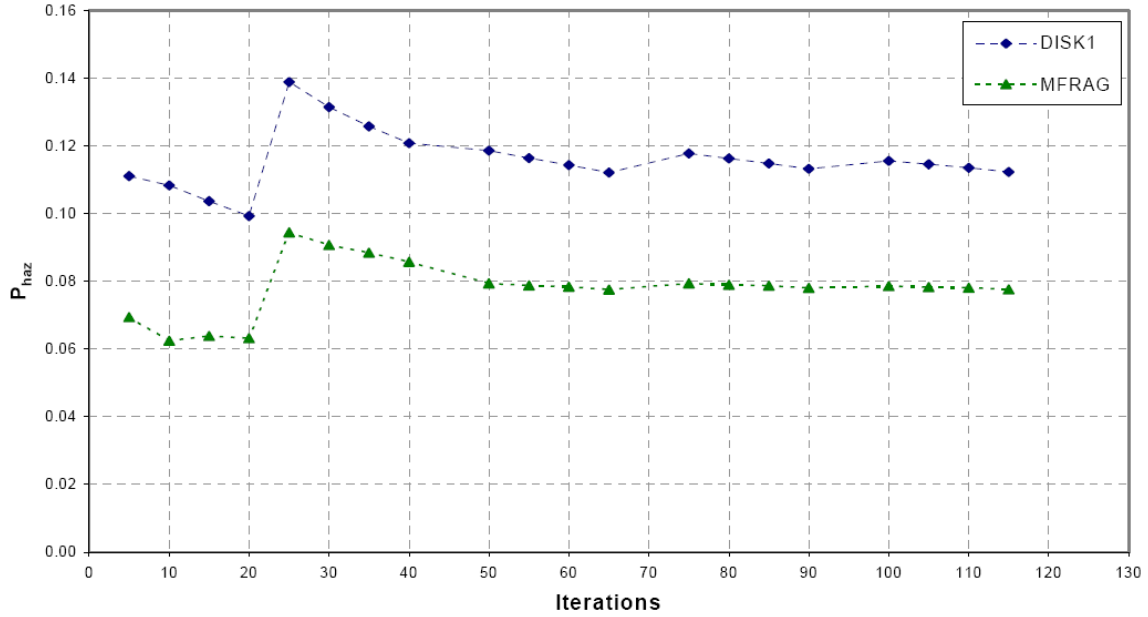


Figure D-33. Convergence for Rotor P_{haz} (Left Engine Fan, V1 to V1+30, 72 Release Angles)

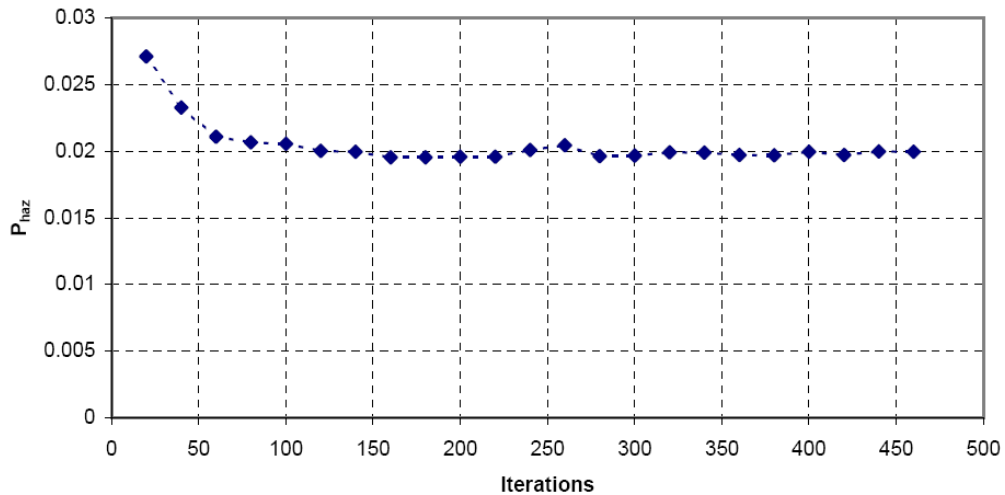


Figure D-34. Convergence for Rotor P_{haz} (Left Engine Fan, V1 to V1+30, SINGL Debris Case, 72 Release Angles)

The test for convergence should be run for each rotor of new target model because the size of the hazard zones will change, thus changing the number of iterations required to fully represent the possible shotlines.

Though the 1-in-20 values were within $\pm 6\%$ percent difference, the analysis of the fan P_{haz} convergence indicated that the number of iterations used in the analyses were insufficient. Despite these results, only 25 iterations (and 100 for the single small

fragment) were used to reduce the run time. This results in 194,400 iterations for a full run on a single configuration. The number of shotlines for these cases will depend on the number of nodes on the shotline grid specified by the debris characterization record for segment and large fragment debris types. Because the purpose of these analyses was aimed more towards exercising the code and examining the capabilities, representative, but not necessarily certification level, values were needed. Also the same random seed was used for each of the cases for all of the production runs, so the shotlines for each case were identical.

Baselines

The relative contribution of each component to the overall P_{HAZ} of the system was consistent for all of the baseline runs (Figures D-35 through D-37). Damage to the opposite (nonevent) engine was the largest contributor to the aircraft P_{HAZ} for both 1/3 disk section and single small fragment debris categories. The engines and the structure were the next most vulnerable to catastrophic damage from 1/3 disk sections from the first five rotors (Fan, LPC2, HPC1, HPC2, and HPC3). Debris from the four aft rotors (HPT and LPT sections) primarily affect the P_{HAZ} contribution of the opposite engine, aft engine beam, and structure. Catastrophic damage from severed fuel supply and fuel motive lines (resulting in fire) was also a major contributor to the fan P_{HAZ} (rotor) for the 1/3 disk section debris category.

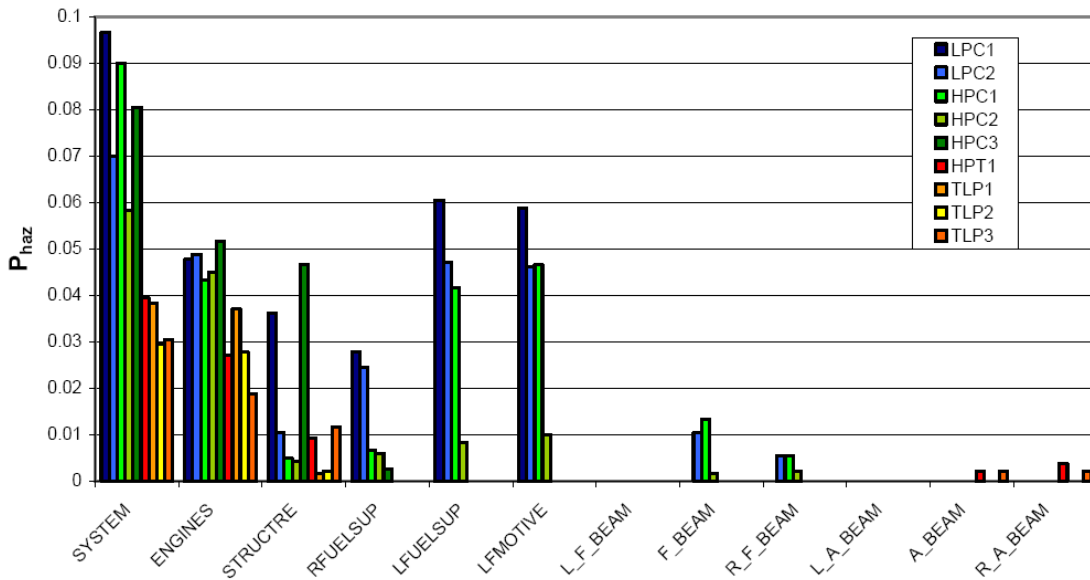


Figure D-35. One-Third Disk Section Component Contribution to P_{HAZ} (Left Engine Baseline, V1 to V1+30 Seconds).

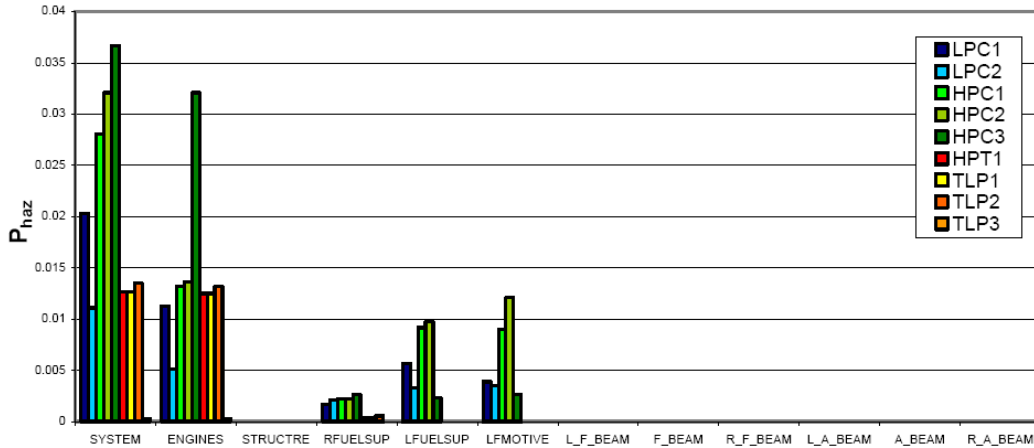


Figure D-36. Single Small Fragment Component Contribution to P_{haz} (Left Engine Baseline, V1 to V1+30 Seconds).

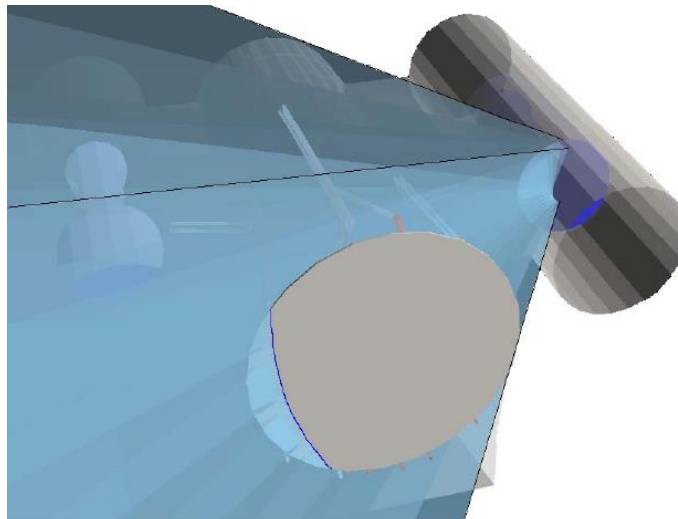


Figure D-37. Hazard Zone for Fan (Single Small Fragment Debris Category).

The turbine section is aft of the fuel supply and motive lines, thus the small fragments from the turbine section do not endanger the fuel lines as much as the forward (fan/compressor) rotors. The major contributor to P_{HAZ} for the single small fragment cases was damage to the opposite engine.

The fan (LPC1) and HPC3 were two of the largest and heaviest of the engine rotors. The size and positioning of these two rotors make them the large contributors to the overall P_{HAZ} . Although the HPT1 is heavier than the HPC3, it is located aft of the fuel lines and does not contribute as much to the engine P_{haz} .

DFM vs. FAA AC 20-128 Spread Angle Specifications

A comparison was conducted between the spread angles for uncontained engine debris specified by the AC 20-128 and the DFM. Because it was used as a baseline for the analysis, the DFM was also used as the baseline for this comparison.

In some instances the hazard zone became smaller when the AC 20-128A values were used (Table D-8). For example, the small fragment spread angles for the HTP1 changed from 20° forward and -50° aft to ±15° from the plane of rotation (specified by the DFM and FAA AC 20-128A, respectively). The major contributors to the single small fragment P_{HAZ} were the fuel lines and the opposite engine. Reducing the spread angles by 40° drastically changed the ratio of critical components to hazard zone area by focusing the hazard zone on the two most vulnerable components in the aft fuselage, the opposite engine and the fuel lines. The AC 20-128A suggested spread angles for turbine the resulted in a P_{HAZ} of almost double that of the DFM for the both the left and right engines.

TABLE D-8. Debris Spread Angles Comparison

COMP	SINGL				MFRAG				DISK1			
	Generic Uncontained Engine Debris Fragment Model		AC 20-128		Generic Uncontained Engine Debris Fragment Model		AC 20-128		Generic Uncontained Engine Debris Fragment Model		AC 20-128	
	Fwd Spread	Aft Spread	Fwd Spread	Aft Spread	Fwd Spread	Aft Spread	Fwd Spread	Aft Spread	Fwd Spread	Aft Spread	Fwd Spread	Aft Spread
LPC1	21	-35	15	-15	2	-3	5	-5	2	-3	3	-3
LPC2	15	-3	15	-15	10	0	5	-5	5	-5	3	-3
HPC1	15	-3	15	-15	10	0	5	-5	5	-5	3	-3
HPC2	15	-3	15	-15	10	0	5	-5	5	-5	3	-3
HPC3	15	-3	15	-15	10	0	5	-5	5	-5	3	-3
HPT1	20	-50	15	-15	0	-12	5	-5	3	-11	3	-3
LPT1	20	-45	15	-15	5	-30	5	-5	3	-5	3	-3
LPT2	20	-45	15	-15	5	-30	5	-5	3	-5	3	-3
LPT3	0	-75	15	-15	5	-30	5	-5	3	-5	3	-3

The AC 20-128A debris characterization resulted in a larger hazard zone and higher P_{HAZ} from damage to the opposite engines and adjacent fuel lines for the LPC section. The fan disk spread angles increased from 2° forward and -3° aft to ±3° and resulted in an increase of approximately 15% in the left engine fan P_{HAZ} and 5% for the right engine fan

P_{haz} . Most of the intermediate fragment hazard zones shifted aft or increased in size. This shift or increase in hazard zone reduced ratio of presented area of critical components to noncritical area resulting in lower P_{HAZ} .

The shotline density and probabilistic nature of the analysis might also have contributed to the differences in both the individual run P_{haz} and aircraft P_{HAZ} variations. Because the spread angle changed, the trajectories generated by UEDDAM are different despite the same random seed having been used. The overall changes for the aircraft P_{HAZ} ranged from a 3.1% increase for the 1/3 disk sections to a 25% decrease for the intermediate sized fragments.

Changes to the Debris Characterization

A second change in the debris file was made to the MULTI debris category. The original debris file using the DFM specified a single small fragment (25% of the blade) associated with the disk burst. Based on the events used to produce the DFM, the events were re-examined to produce a modified debris model. The small fragment size was changed to two sizes of fragments and fore/aft spread angles (Table D-9). The number of fragments remained the same, but the sizes better represented the historical data.

The larger sized small fragments associated with the compressor disk burst in the modified debris file had a smaller spread angle than those in the original debris file. The new hazard zone focused more of the fragments into the area where many critical components are located, resulting in a higher P_{HAZ} . The fan fragments changed in size, but not spread angles, and resulted in lower component P_{haz} .

The two sizes of low-pressure turbine small fragments also resulted in lower component P_{haz} . The overall result was a 0.0021 (or 3.6%) increase in the overall aircraft P_{HAZ} for the new MULTI debris category compared to the run where only one size small fragment was used. Because only the MULTI debris category was modified, all other debris category results were unchanged.

TABLE D-9. Changes to MULTI Debris Characterization

Debris Category	Comp	DFM				Modified DFM			
		No. of Fragments	Weight (lb)	Forward Spread Angle	Aft Spread Angle	No. of Fragments	Weight (lb)	Forward Spread Angle	Aft Spread Angle
MULTI	LPC1	1	9.850	2	-3	1	9.850	2	-3
		27.7	0.123	15	-30	17.7	0.079	15	-30
	LPC2	1	3.130	5	-5	1	3.130	5	-5
		5	0.044	5	-25	3	0.040	5	-25
	HPC1	1	1.970	5	-5	1	1.970	5	-5
		5	0.025	5	-25	3	0.023	5	-25
	HPC2	1	1.060	5	-5	1	1.060	5	-5
		5	0.017	5	-25	3	0.015	5	-25
	HPC3	1	8.420	5	-5	1	8.420	5	-5
		5	0.204	5	-25	3	0.367	5	-25
	HPT1	1	10.470	3	-11	1	10.470	3	-11
		12	0.074	15	-60	7	0.033	15	-60
	LPT1	1	2.680	3	-5	1	2.680	3	-5
		5	0.051	5	-40	3	0.020	5	-40
	LPT2	1	3.220	3	-5	1	3.220	3	-5
		5	0.071	5	-40	3	0.028	5	-40
	LPT3	1	6.710	3	-5	1	6.710	3	-5
		5	0.091	5	-40	3	0.036	5	-40
						2	0.063	5	-40

Pressure Bulkhead Criticality

An investigation was conducted to examine the consequences of making the aft pressure bulkhead critical during cruise. This run emphasized the importance of the debris characterization. The aft pressure bulkhead was critical only during the cruise portion of the flight.

The Final Climb phase includes final climb, cruise, and initial descent. The debris file used the spread angles from the DFM report. The DFM spread angles for the fan are 21° forward and -35° aft of the plane of rotation. The FAA AC20-128 only specifies a ±15° fore/aft spread angle criteria for uncontained blade (small fragment) events.

The intercept file indicates that the bulkhead would be in the hazard zone for release angles of 355° to 50° (±5°) for the left engine and 140° to 210° (±5°). The pressure bulkhead was outside the hazard zone for the remaining rotors.

Because the single small fragment debris case for the fan were the only category and engine rotor to effect the pressure bulkhead, the change in P_{HAZ} was very small (1.83e-5) and resulted in a change to the baseline P_{HAZ} due to a single small fragment of less than 0.3%.

Skin Thickness Increase

The skin thickness of the aft section of the GBJ was increased to protect the structure and opposite engine from uncontained engine debris. For this trade study, the skin thickness was varied from 0.032" (baseline) to 0.080" thick aluminum in increments of 0.016". The three configurations of 0.048", 0.064", and 0.080" thick skin resulted in an increase of 44 lb, 88 lb, and 133 lb, respectively.

Table D-10 presents a summary of the UEDDAM results for increasing skin thickness. Increasing the skin thickness from 0.032" to 0.08" thick aluminum reduces the P_{HAZ} by 37% for the single small fragment debris category. The increased thickness provides a small reduction to the P_{HAZ} for the 1/3 disk case and intermediate fragment debris categories. The percent differences calculated to illustrate the reduction are important figures, but can be deceiving. It is important to realize that though the increased skin thickness reduced the P_{HAZ} for the small fragment by 37%, the actual reduction in P_{HAZ} (0.00352) is relatively small because the baseline P_{HAZ} was small. The reduction in P_{HAZ} for the 1/3 disk section was almost 0.001 but only reduces the P_{HAZ} by 2.09% of the baseline hazard level.

TABLE D-10. Skin Thickness Study Aircraft P_{HAZ} Results*

Skin Thickness	Baseline (0.032")	0.048"		0.064"		0.08"		Hybrid	
		P _{HAZ}	ΔP _{HAZ} %	P _{HAZ}	ΔP _{HAZ} %	P _{HAZ}	ΔP _{HAZ} %	P _{HAZ}	ΔP _{HAZ} %
1/3 Disk	0.04588	0.04587	-0.03%	0.04581	-0.17%	0.04492	-2.09%	0.04547	-0.91%
Multiple	0.06853	0.06612	-3.52%	0.06452	-5.85%	0.06148	-10.29%	0.06485	-5.38%
Intermediate Fragment	0.03745	0.03723	-0.58%	0.037	-1.20%	0.03568	-4.74%	0.03623	-3.25%
Single Fragment	0.00962	0.00838	-12.84%	0.00735	-23.62%	0.00606	-37.03%	0.00614	-36.19%

* ΔP_{HAZ} indicates change from baseline values

The small changes in P_{HAZ} to the 1/3 disk section and intermediate fragment debris categories are mostly due to reductions in the contribution to P_{HAZ} from the fan (Figures D-38 and D-39). This slight decrease in P_{HAZ} for the first stage fan was largely due to reductions in the contribution to P_{HAZ} from structural damage. Because the engine debris must penetrate five out of eight consecutive stringers, the increases in skin thickness combined with the larger area of the blade and multiple penetrations, reduces the energy of the disk sections. The increased skin thickness effectively stops many of the small fragments from penetrating the fuselage, resulting in a reduction in the contribution to P_{HAZ} from damage to the opposite engine for the single small fragment category (Figure D-40). However, the skin thickness modifications have a very limited effect on the contribution of the fuel lines to system P_{HAZ} .

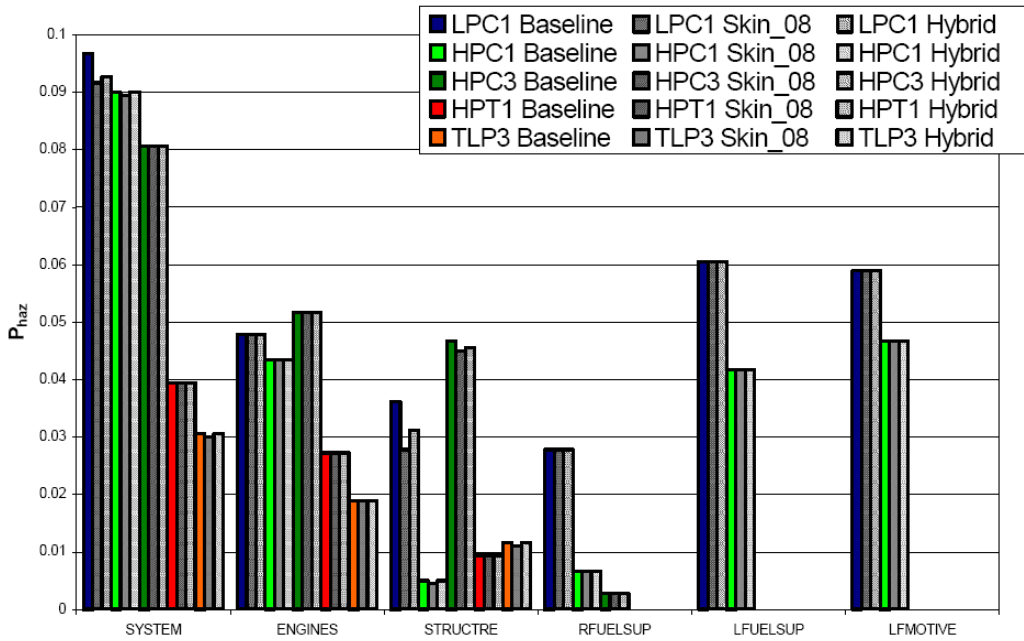


Figure D-38. Component Contribution for Increased Skin Thickness (1/3 Disk Case, V1 to V1+30 Seconds)

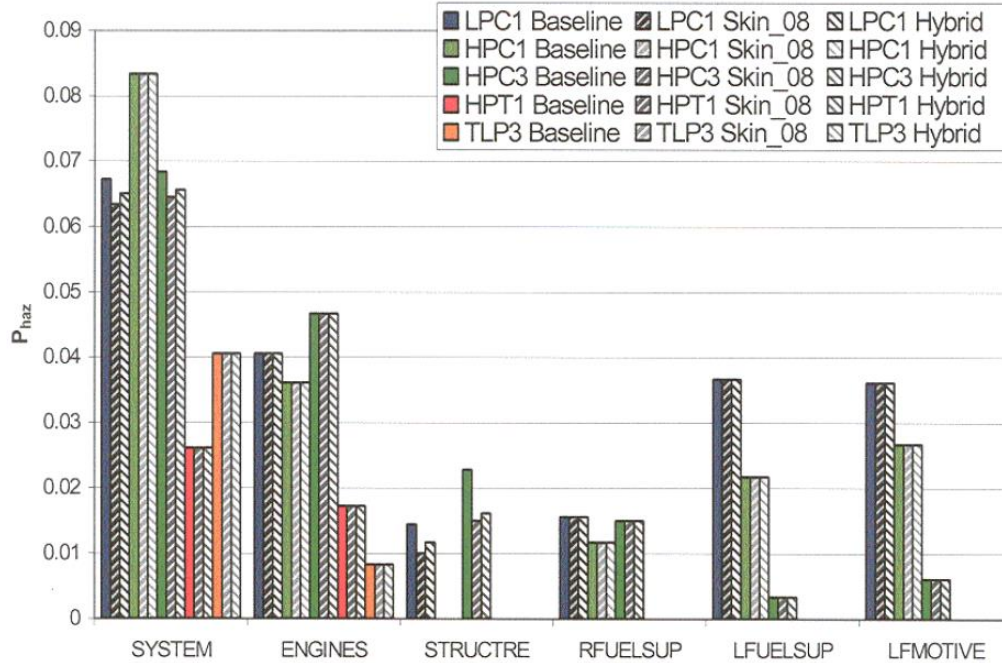


Figure D-39. Component Contribution for Increased Skin Thickness (Left Engine, V1 to V1+30 Seconds, Intermediate Fragment Case)

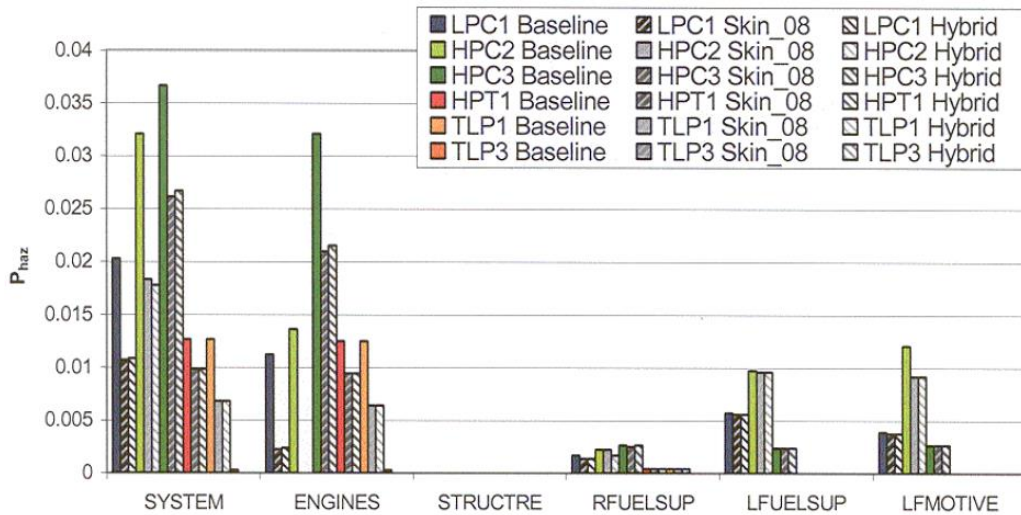


Figure D-40. Component Contribution for Increased Skin Thickness (Left Engine, V1 to V1+30 Seconds, Single Small Fragment Case)

The sensitivity of P_{HAZ} to weight was calculated for each of the cases. This was done by calculating the ratio of the change in P_{HAZ} to weight added to the aircraft by the additional shielding. Based on these three configurations, the sensitivity of P_{HAZ} reduction per lb of shielding weight added is highest for the 0.08" in thick skin. However, the addition of 133 lb of shielding to the aircraft due to the increased skin

thickness is a source of concern. Because any one of these modifications to the fuselage skin would add considerable weight to the aircraft (44 lb to 133 lb), these results emphasized the need to examine more localized armor for critical components.

To reduce the amount of weight added to the aircraft, an alternate configuration, called hybrid, was modeled. The modification to the skin was limited to increasing the skin adjacent to the engines to 0.080" thick aluminum. The engines are mounted in such a way that the tops of the engines are very near the same height as the top of the fuselage. Therefore, the area directly between the two engines encompasses almost the entire top half of the fuselage (indicated in orange in Figure D-41). The length of the shielded area was 60" with two sections extended to 70" to shield the section of the fuel lines inside the hazard zone. Because of the location of the engines, most of the top half of the GBJ fuselage was within the cross-engine damage area. Increasing the skin thickness in these areas would provide additional protection to the opposite engine and fuel lines as well as the upper fuselage structure. The results of this modification were within 1% of the P_{HAZ} values for the 0.080" skin for the small fragment and intermediate fragment debris categories. The change in the P_{HAZ} went from -2.09% to -0.91% for the 1/3 disk section and from -4.74% to -3.25% for the hybrid skin (a change in aircraft P_{HAZ} of 0.0013 and 0.0055, respectively). The additional weight of 16 lb is significantly lower than the 0.080" thick skin, which weighed 133 lb, while maintaining a similar reduction in P_{HAZ} from small fragments.

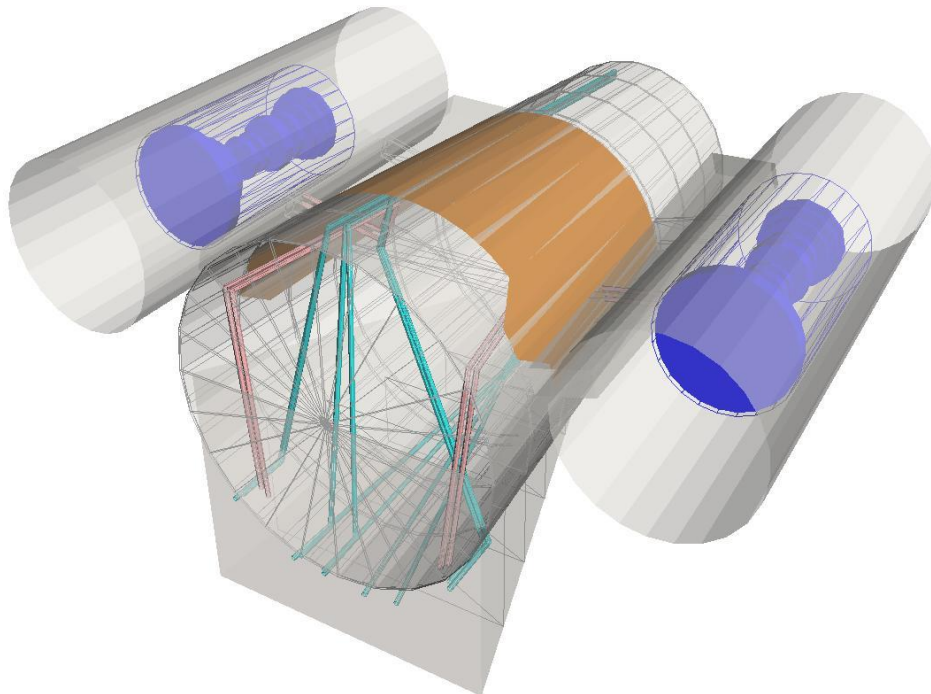


Figure D-41. GBJ Model with Localized Skin Thickness Increase (Hybrid)

Fuel Line Repositioning

Of the three major contributors to the P_{HAZ} (the engines, structure, and fuel lines), the fuel lines are the only components that can be effectively repositioned. Therefore, rerouting of the fuel lines to take advantage of natural protection afforded by the aircraft structure was examined. Because the fuel lines were modeled as long thin cylinders along a simple path, the changes were easily made. The break point locations were moved so that the placement of the fuel lines took advantage of the existing structure. The debris had to pass through the pylons, a stringer, and in some cases, the engine beam before striking the fuel lines. The left supply and motive fuel lines were moved down several inches. The right fuel and motive lines were also moved down slightly so that they lay along a stringer, but still above the pylon (Figure D-42(a) shows the original configuration; Figure D-42(b) indicates the modified fuel line routing).

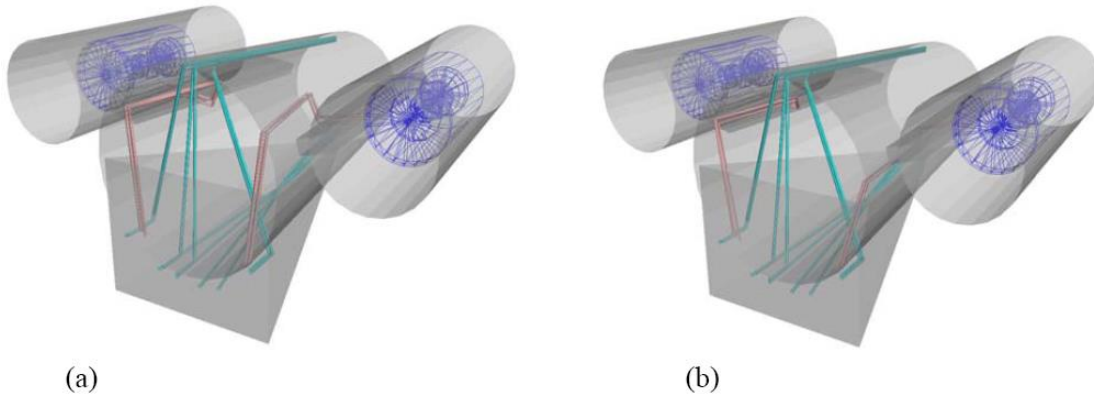


Figure D-42. (a) Baseline 5 and (b) Rerouted Fuel Lines

This configuration was only partially successful. The P_{HAZ} for the single small fragment was reduced by 30.59%. However, the P_{HAZ} for the remaining debris categories, almost all of which are several times larger than the single small fragment case, increased (see Table D-11).

TABLE D-11. Comparison of Fuel Line Repositioning Results

	Baseline	Rerouted Fl	ΔP_{HAZ}	$\Delta P_{HAZ}\%$
DISK1	0.0459	0.0527	0.0051	13.31%
MFRAG	0.0375	0.0394	0.0009	2.87%
SINGL	0.0096	0.0083	-0.0021	-30.59%

Because the incremental method for P_{haz} calculation was used, multiple critical components along the same shotline do not increase the overall P_{HAZ} . Therefore, if damage to only one component along the shotline was prevented, the P_{haz} may not change because the other critical components along that shotline would still be defeated. Such was the case for this configuration. Because the engines and possibly the opposite engine

fuel supply and motive lines lay on the same shotline, the repositioning of the fuel lines created a larger vulnerable area. Moving the fuel lines increased the translational risk angles by almost 10° (Figure D-43). In the original position, the fuel lines were placed such that a shotline passing through the fuel lines would likely strike the opposite fuel supply lines or the opposite engine. The larger fragments had sufficient energy to penetrate the pylon, skin, and stringers and continued through to cause damage to the fuel lines resulting in a fire hazard. However, only one attempt was made in repositioning of the fuel lines. Because fuel lines cannot entirely be removed from the near-field area, industry standard practices suggest reducing the exposure by routing the fuel lines in the most direct route possible.

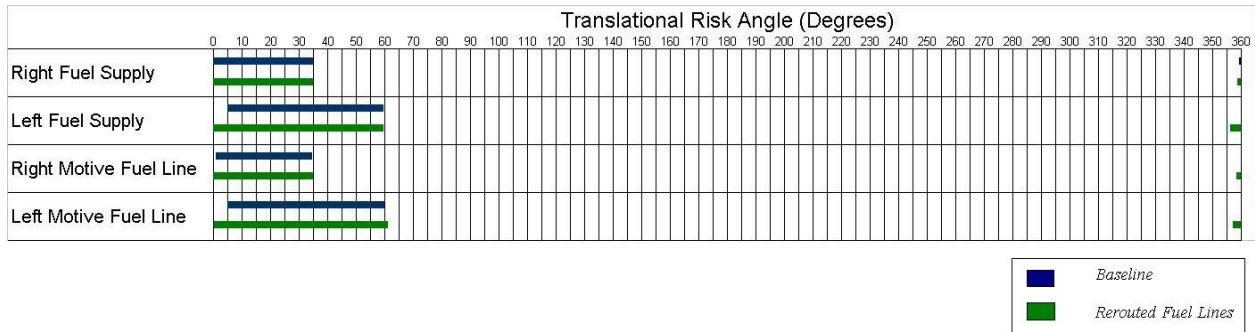


Figure D-43. Translational Risk Angles for Fuel Lines.

Fuel Line Shielding

The critical component shielding portion of this study focused on the fuel lines. The results from baseline runs revealed a high contribution to the P_{HAZ} by damage to the opposite engine, the structure, and the fuel lines adjacent to the damaged engine, resulting in fire. The localized shielding is intended to reduce the P_{HAZ} and weigh less than increasing skin thickness of the fuselage. This approach might also be easier to retrofit onto existing aircraft.

Baseline runs indicated that damage to the fuel lines from uncontained engine debris from the first four rotors were a significant contributor to the overall P_{HAZ} (up to 48%). Damage to the fuel lines adjacent to the event engine can result in a fire. Damage to the nonevent engine fuel lines would result fuel starvation and loss of thrust from the functioning engine and possibly catastrophic hazard. However, the adjacent fuel lines are larger contributors to the overall P_{HAZ} .

Two plates were used to shield the fuel lines. One plate was used to shield the horizontal section of the fuel lines; and another, smaller, plate was used for the vertical section (Figure D-44). The horizontal plates were approximately 7.5" wide, tapering to 6.5" at the aft end and approximately 42" long. The vertical plate measured 9" by 4". The plate thickness and material was varied. Cases with only one of the two shielding sections (horizontal or vertical sections) in place were run to identify which part of the fuel line was the greater contributor to the P_{HAZ} . This was necessary because the fuel and motive

lines were all modeled as a single component. The results indicated that approximately 13% of the contribution to the total P_{HAZ} comes from the shielding of the vertical section of the fuel lines and 29% from the horizontal section, the remaining 58% of the overall P_{HAZ} is due to structural or opposite engine damage.

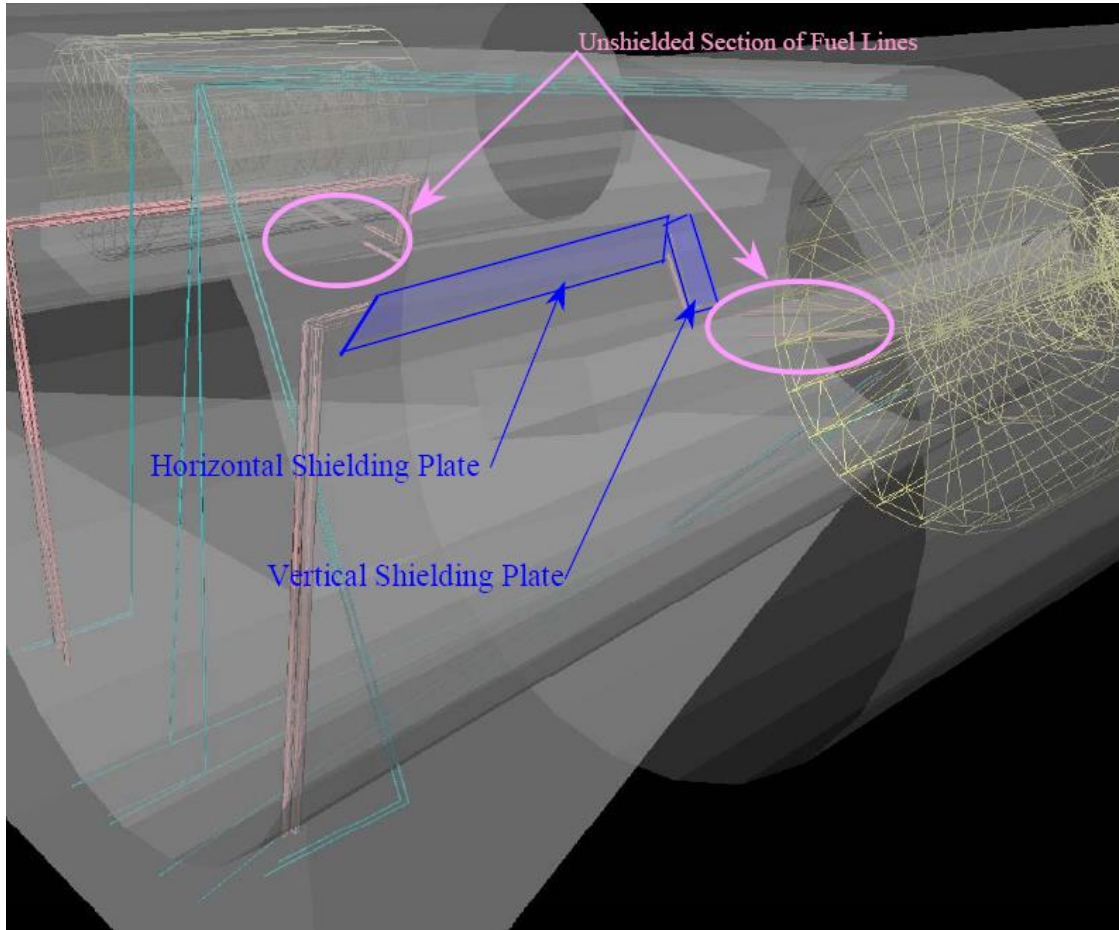


Figure D-44. Horizontal and Vertical Shielding Plates

The penetration equations were used to estimate the plate thickness required to stop the uncontained engine debris. The size, weight, and velocity of the fragments from the third stage HPC3 and steel, aluminum, and titanium plates were used for this estimation. This rotor was selected as the test case because it was one of the larger fragments and had a high P_{HAZ} relative to the other rotors for the single small fragment case. The small, intermediate, and 1/3 disk fragment characterizations were taken from the debris file.

Aluminum, titanium, and steel plates were considered for the component shielding. According to the penetration equation results, the aluminum shields had to be more than twice the thickness of either titanium or steel shielding plates to stop the HPC fragments. The titanium plates required to stop the various HPC fragments were only slightly thicker than the steel plate but much lighter (Figure D-45).

According to the penetration equation, a 0.08" thick steel plate will result in almost the same residual velocity for a 0.11" thick titanium plate (517 ft/s and 519 ft/s, respectively) for a given projectile size and orientation. Runs were conducted for shielding plates adjacent to the fuel lines of 0.08" thick steel and 0.11" thick titanium fuel. The resulting P_{HAZ} for these shielding plates were within 5% of each other. Additional runs were conducted for 0.20" thick steel plates and a 0.24" thick titanium plates adjacent to the fuel lines. For this case, the percent difference was less than 0.1%. From the results of this comparison, one can infer that the results from plates of one material can be applied to equivalent plates of a different material.

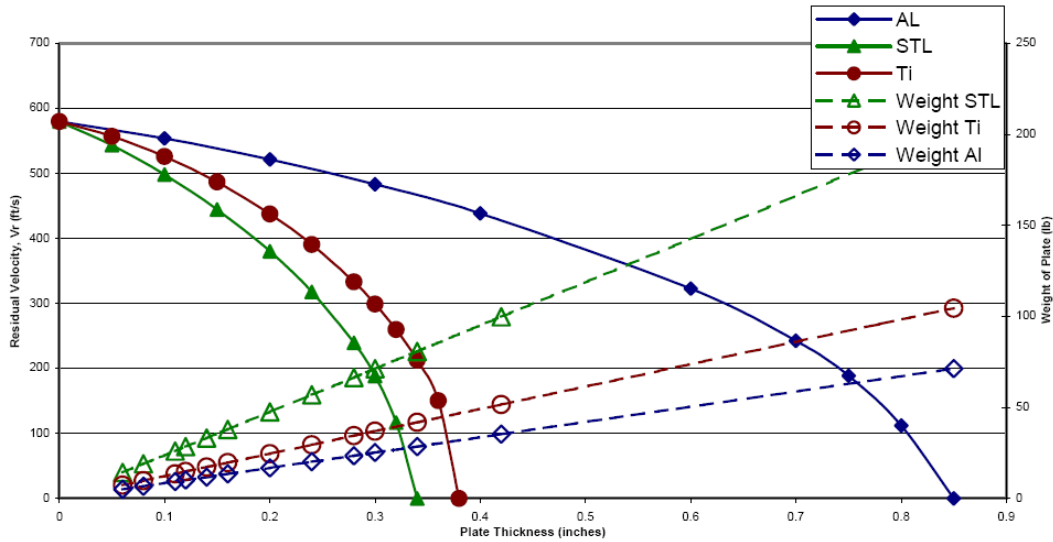


Figure D-45. Plate Residual Velocity and Weight Comparison (1/3 Disk Section).

To evaluate the potential P_{HAZ} reduction from the protecting of adjacent fuel lines from fire, a run was conducted with the fuel lines adjacent to the event engine categorized as noncritical to represent maximum shielding of the fuel lines. The probability of catastrophic hazard (fire) in the event of a hit (P_{haz}) for the fuel lines were set to 0 for the case labeled Fuel Lines NC (indicating fuel lines Not Critical with no shielding included in the target model). The addition shielding resulted in P_{HAZ} lower than that of the Fuel Lines NC configuration in some cases, and in others, the P_{HAZ} was higher than identified by the noncritical fuel line case. The shielding was meant to protect the fuel lines, but being located between two engines, it also offered limited protection to the opposite fuel lines and nonevent engine. This placement of the shielding may have resulted in the lower than expected P_{HAZ} in some cases. The cases where the P_{HAZ} was greater the noncritical fuel line case may be attributed to the small section of fuel lines between the fuselage and engine that was not shielded (Figure D-44).

The aircraft P_{HAZ} for the most effective thicknesses of titanium and steel plates are listed below in Table D-12 and plotted in Figure D-46. The highest ratio of reduction in P_{HAZ} per lb of shielding added was for the 0.18" titanium shielding plates.

TABLE D-12. Aircraft P_{HAZ} from Uncontained Engine Debris for GBJ with Fuel Line Shielding Plates*

	Baseline P _{HAZ}	0.08" Ti	0.11" Ti	0.18" Ti	0.22" It	0.24" Ti
1/3 Disk	0.04588	-0.17%	-0.68%	-13.67%	-19.07%	-19.07%
Intermediate Fragments	0.03745	-3.59%	-7.47%	-26.65%	-26.65%	-26.65%
Single Fragments	0.00962	-37.77%	-39.14%	-39.14%	-39.14%	-39.14%

* Percent difference from baseline values

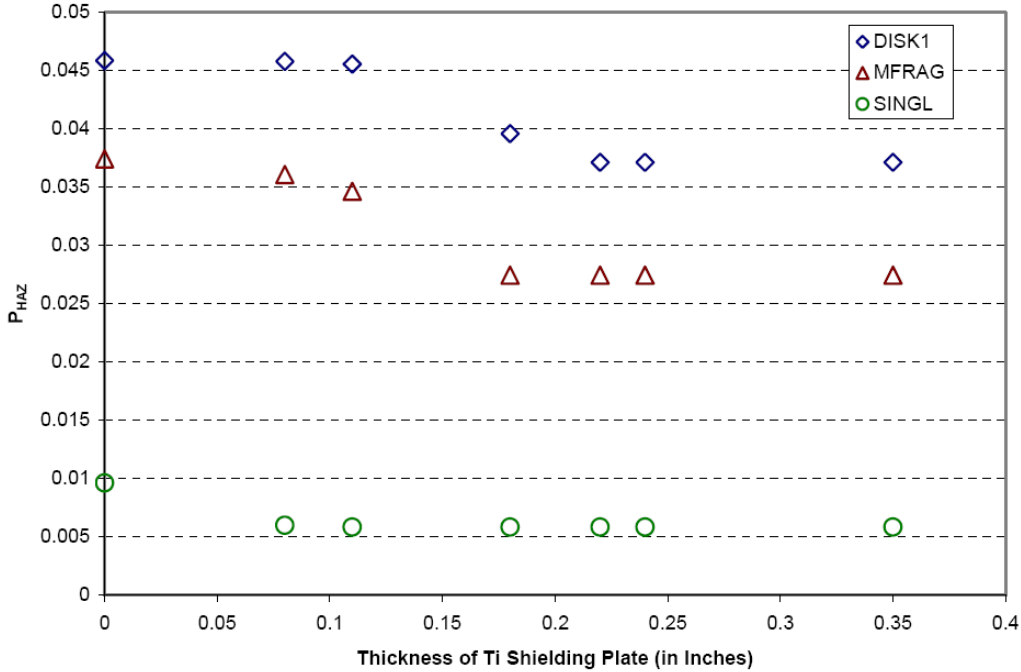


Figure D-46. Overall P_{HAZ} for GBJ with Titanium Fuel Line Shielding Plates.

The shielding plates reduced the P_{HAZ} for the aircraft by almost eliminating the contribution of fuel lines adjacent to the event engine. The results for a left engine disk event with the 0.22" thick titanium shield indicate no contribution to P_{HAZ} from damage to the left fuel lines, a reduction in P_{HAZ} of approximately 0.06. However, because the incremental method for P_{haz} calculation was used, the effects of this reduction in component P_{haz} are reduced in the system (rotor) and overall P_{HAZ} (Figure D-47).

A significant contribution to the reduction of P_{HAZ} is due to decrease in the component P_{haz} resulting from damage from the fan. Though the shielding is intended to provide protection to the fuel lines, the plates also offer limited protection to opposite engine and structure (Figures D-47 through D-49).

The shield did not stop all of the small fragments from damaging the fuel lines. This may be due to the small, unshielded portion of the fuel lines from the fuselage to the engine.

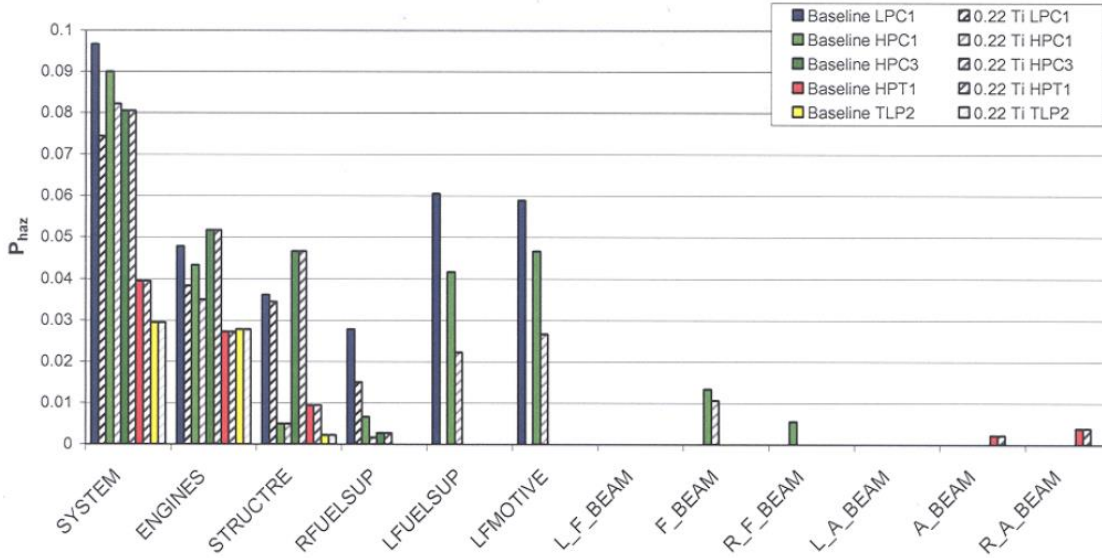


Figure D-47. Component Contribution with Fuel Line Shielding for GBJ (Left Engine, V1 to V1+30, 1/3 Disk)

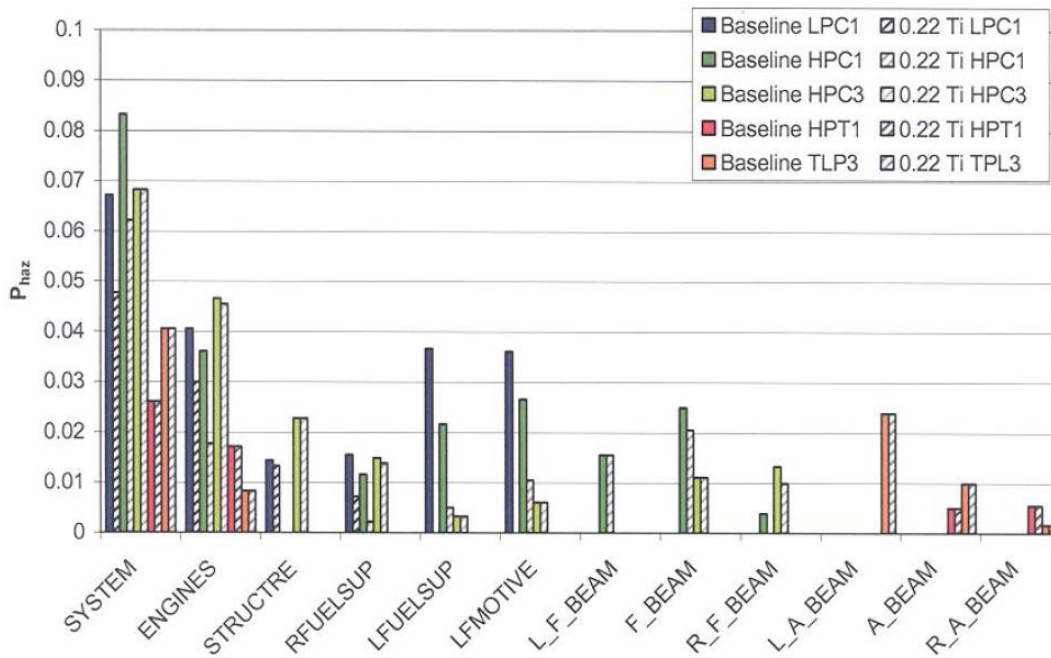


Figure D-48. Component Contribution with Fuel Line Shielding for GBJ (Left Engine, V1 to V1+30, Intermediate Fragments)

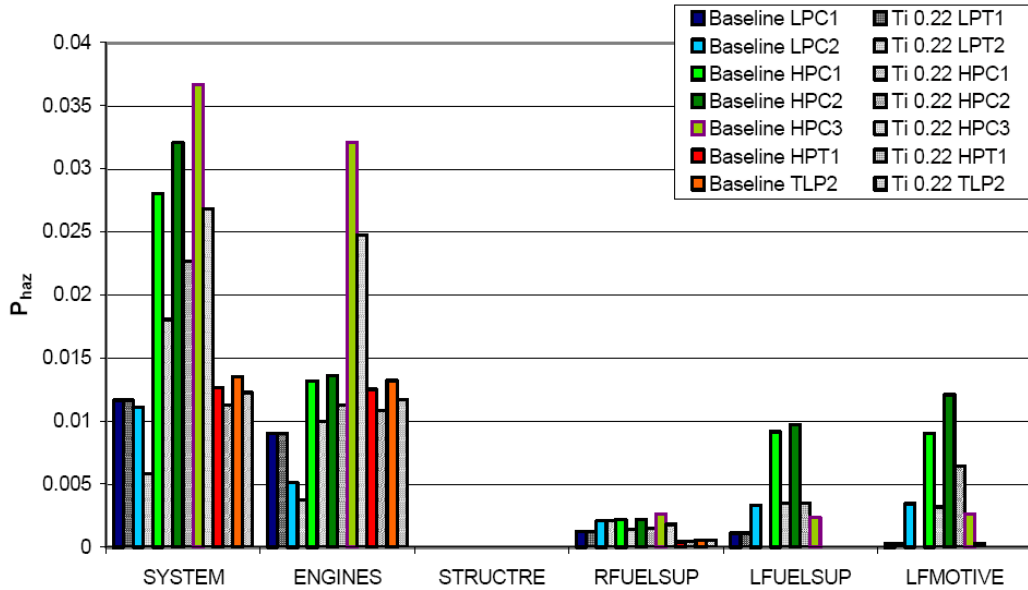


Figure D-49. Component Contribution with Fuel Line Shielding for GBJ (Left Engine, V1 to V1+30, Single Small Fragment)

General Business Jet Analysis Summary

Although the increased skin thickness was effective in reducing the P_{HAZ} for all the debris cases, the increased weight may be prohibitive. Even a 50% increase in skin thickness required the addition of 44 lbs. in the aft section of the aircraft. The localized shielding resulted in a similar reduction in P_{HAZ} but weighed only 16 lb. However, increasing the skin thickness did not reduce the P_{HAZ} for the intermediate fragment to below the 1-in-40 (0.025) AC 20-128A maximum allowable P_{HAZ} for the GBJ (Table D-10).

The plates were effective in preventing the fuel lines from being severed and, hence, possible fires. Shielding of the fuel lines reduced the P_{HAZ} by up to 39% for the small fragment case, a change of 0.0038 to the P_{HAZ} . The P_{HAZ} for the intermediate fragment debris category was reduced by as much as 0.010; this represents a reduction of almost 27% of the aircraft (baseline) P_{HAZ} debris category (for shielding plates 0.18" Ti or thicker). This shielding was slightly less effective for the 1/3 disk sections, resulting in a reduction of P_{HAZ} of 0.0088 or 19% for the 0.22" Ti plates.

Although the component repositioning has the potential for the least weight penalty of the mitigation strategies attempted, the increase in P_{HAZ} for the larger fragments outweighed its success in stopping the small fragments.

For the sensitivity analysis, the results of each type of modification were compared based on the reduction in P_{HAZ} per pound of weight added to the aircraft. The results indicate that the 0.22" Ti plate resulted in the highest reduction in P_{HAZ} per pound of weight added to the aircraft for the large and intermediate debris categories (Table D-13). The localized skin thickness increase (hybrid) resulted in the greatest reduction in P_{HAZ} due to

the single small fragment debris category, but did not reduce the P_{HAZ} for the intermediate fragment to below the 1-in-40 maximum allowable P_{HAZ} . Also, the baseline P_{HAZ} for the single small fragment debris category was several times smaller than those of any other debris category examined.

TABLE D-13. Change in P_{HAZ} per Pound Shielding
 ΔP_{HAZ} per lb. Shielding

	DISK	MFRAG	SINGL	Weight (lb.)
0.18" Ti Plate	-2.83E-04	-4.51E-04	-1.70E-04	22
0.22" Ti Plate	-3.23E-04	-4.70E-04	-1.39E-04	27
0.24" Ti Plate	-2.96E-04	-4.31E-04	-1.27E-04	28
0.080" Skin Thickness	-3.12E-06	-9.19E-06	-2.61E-05	133
Hybrid (Localized 0.080" Skin)	-2.60E-05	-7.66E-05	-4.68E-04	16
Hybrid (Localized 0.160" Skin)	-7.62E-05	-1.65E-04	-2.34E-04	32

The 0.22" titanium fuel line shielding plate would constitute 27 lb of additional weight to the aircraft, including 3 mounting flanges on each side, and would result in as much as a 39% reduction in the P_{HAZ} from a single small fragment (Table D-14). The mounts were made of the same material and thickness as the shield and located at the fuselage frame locations.

TABLE D-14. Results for 0.22" Thick Titanium Fuel Line Shielding Plate

	Baseline	0.24" Ti	% Difference
DISK1	0.04588	0.03713	-19.07%
MULTI	0.06853	0.05817	-15.12%
MFRAG	0.03745	0.02473	-33.97%
SINGL	0.00962	0.00585	-39.14%

CONCLUSIONS AND RECOMMENDATIONS

CONCLUSIONS

The results of the generic aircraft analysis show that Uncontained Engine Debris Damage Assessment Model (UEDDAM) is a viable tool to assess rotor burst hazard analysis. Study results indicate that common aircraft design practices result in acceptable risk levels to realistic engine debris. Even when considering 1/3 disk segment debris trajectories out of plane of the disk rotation, values of risk probability were below the 1-in-20 requirement. This supports validation of UEDDAM in that the results show

common design practices that have been proven in aircraft certification programs that meet Advisory Circular (AC) 20-128A minimization intent are also predicted by UEDDAM to be within those limits. While this is not a conclusive validation, it does shed a favorable light on UEDDAM.

One of the benefits of UEDDAM over existing manual methods is that UEDDAM automates a critical part of the AC 20-128A analysis process. Automation reduces errors, permits standardization, allows for ease of trade studies, and introduces a manageable means of evaluating uncontained engine events stochastically.

By using UEDDAM as part of the uncontained engine debris damage assessment process, a more accurate and useful analysis can be achieved. UEDDAM outputs data that allows determination of what engine sections are the major contributors to damage and, in addition, allows determination of what systems and components are responsible for the hazard. Based on this information, design improvements can be identified and prioritized. Since aircraft system and component contributions can be quantified, design improvements can be focused on specific aircraft systems. This allows for a more tailored design process that can consider more than just shielding components. For example, the analysis of the generic twin-engine aircraft (GT) showed that the hydraulic system contributed significantly to the hazard. Simply rerouting hydraulic lines and moving components can achieve a measurable reduction in risk.

UEDDAM allows for consideration of segment, intermediate, and small fragment trajectories that are out of plane of the rotor rotation. This goes beyond the common manual, infinite energy 1/3 disk segment analysis that only considers segment releases in plane with the rotor rotation. By looking out of the plane, additional vulnerabilities associated with a realistic uncontained engine event can be identified and remedied. UEDDAM also considers energy reduction of all debris types. This allows for consideration of the inherent shielding effects of major structure, such as landing gear. The common manual method of assuming that the debris energy never decreases (infinite energy) does not allow for consideration of shielding effects.

UEDDAM allows for parametric studies. Once the data set has been developed, parametric studies can be performed easily by making small changes to the input and rerunning UEDDAM. This automation saves time over manual methods where parametric studies must manually repeat the entire analysis process.

Uncontained engine events are highly stochastic in nature. UEDDAM is designed to consider the stochastic nature of uncontained engine events by varying release point, debris trajectory, and debris orientation over multiple iterations to give a more complete assessment of the threat. The fore/aft trajectory can be varied uniformly, normally, or skewed to allow better representation of real debris.

The UEDDAM visualizer aides in input development and output data analysis. The Visualizer was used in support of this analysis by plotting the components in three-dimensional space to allow analysis and discussion of the input and output data. The

hazard zone plots allowed easy discovery of the limitation in shielding provided by the engine nacelle (trajectories of the turbine section debris escaped aft of the engine nacelle). While not shown in this analysis, the UEDDAM Visualizer creates plots of the translational risk angles described in AC 20-128A.

This effort has demonstrated that UEDDAM addresses the industry/Federal Aviation Administration need for an analytical tool to conduct rotor burst assessment that includes fragment penetration, system level hazard assessment, and multiple debris fragments. UEDDAM as a design tool can provide early insight to the rotor burst hazard for a given aircraft configuration. Additionally, trade studies can be performed to conduct cost/benefit analyses and minimize the rotor burst hazard. As a certification tool, UEDDAM provides a standardized approach to conduct rotor burst hazard assessment. UEDDAM output provides insight to the rotor burst hazard and can be used to develop a top level 1 in 20 analysis to address compliance to Code of Federal Regulations.

It is well understood that a rotor burst analysis is a complex analysis. UEDDAM was developed to provide useful tools to aide in conducting the analysis and presenting the results. A UEDDAM Visualizer was developed to allow visualization of the complex data and information generated from a UEDDAM run. It allows visualization of the aircraft geometry, debris hazard zones, debris trajectories, probability plots of the hazard levels, and translational risk angles.

RECOMMENDATIONS

The lessons learned from this study show that UEDDAM has great potential to provide uncontained engine debris damage assessment in support of design and certification. This study shows promise, but it was performed by the developers who are intimately familiar with FASTGEN and COVART. To get a better gauge of the usefulness of UEDDAM, it is recommended that an aircraft manufacturer exercise UEDDAM using the personnel that would normally do such analyses.

Use of the UEDDAM visualizer greatly enhanced data preparation and analysis but there are areas for improvement within the visualizer. The user interface is adequate but there is room for improvement. The data displayed in the visualizer was selected by the UEDDAM developers in anticipation of what the commercial aircraft manufacturers might be interested in. As part of the UEDDAM exercise described above, it is recommended that the aircraft manufacturer attempt to exercise the visualizer and document any improvement suggestions.

The rotor burst analysis is a complex process. UEDDAM provides a means to standardize this process. UEDDAM and its supporting tools are powerful, allowing for a wide variety of approaches to support uncontained engine debris damage assessment. To achieve the full power and flexibility of UEDDAM as an analysis tool, many variables and inputs must be accurately defined and generated. It is recommended that, following the aircraft manufacturer analysis, analysis guidelines be developed for using UEDDAM.

APPENDIX E

THE GENERIC ENGINE DEBRIS FRAGMENT MODEL TABLES

The following is an excerpt from the following report: Seng, Silvia, John Manion, and Chuck Frankenberger, *Uncontained Engine Debris Analysis Using the Uncontained Engine Debris Damage Assessment Model*, DOT/FAA/AR-04/16, NAWC-WD, China Lake, California, September 2004. Included are the debris fragment model tables that should be used to aid the user in developing debris characterizations for a UEDDAM analysis.

Debris Fragment Model Tables

An uncontained turbine engine debris fragment model was defined using the database and fragment penetration tools developed under the FAA sponsored NAWCWPNS Uncontained Engine Debris Damage Mitigation Program. The debris fragment model is sub-divided into turboshaft, low bypass ratio (LBPR) and high bypass ratio (HBPR) engine categories. Within each category component failure types are defined and the uncontained debris characteristics provided. The model defines the average number of fragments per event, fragment size, velocity and trajectory.

The basis for this model is the *Large Engine Uncontained Debris Analysis* report and *Small Engine Uncontained Debris Analysis* report. These reports are based on historical event data to define the debris size and trajectories, specific engine characteristics and debris penetration analysis to define the fragment velocities. The event data has been analyzed and compiled into a database that can be used to define debris trends for engine categories and failure modes.

Fragment size is primarily based on damage (hole size) that has been done on aircraft. The damage dimensions were normalized to the engine component dimensions. For a fan blade event, aircraft damage was normalized by dividing the damage length by the fan blade length. Aircraft damage done by disk fragments was normalized by dividing the damage length by the disk diameter. This normalization process provides a realistic estimate of the fragment size that caused the damage, and provided a means to scale damage from one engine type to another.

The fragment model was developed to encompass a “significant majority” (approximately 85 % of the data) of the damages to the aircraft in terms of fragment size and trajectory angle. Data excluded were generally small damages at extreme trajectory angles which penetrated aircraft secondary structure only. To develop the fragment tables for each engine category and failure mode (e.g., HBPR fan blade failure) the data was sorted and plotted in a histogram based on debris size and debris trajectory. It should be noted that the particle sizes represent only those particles that actually struck the aircraft, and that the actual event may have produced other fragment sizes and initial velocities. The tables below represent only those particles that were documented as a threat to the aircraft.

Fragment velocity is the velocity of the fragment after it exits the cowl/nacelle structure. The analytical process defined in the *Large Engine Uncontained Debris Analysis* report was used to calculate the fragment velocity. The velocity values were based on the initial

fragment-centroid velocity value with some resulting attenuation after penetrating the surrounding structure. In the cases of blade failures, a review of numerous fan blade out test results provided additional insight to the fragment velocity. High speed movie documentation confirmed that sliding frictional effects of released blade fragments against engine case structure as the fragment traveled in a forward helical trajectory would reduce their velocities to approximately 75% of the original blade tip velocity. This 75% rule was also applied to blade particles in events initiated by disk failure as well, to account for blade breakage prior to exiting the case as well as blade particle friction. This analysis was supported by one “data rich” field event where initial velocities were known and fragment size and energy were confirmed by aircraft damage analysis. For blade initiated events, blade fragment velocities are based on penetration of the engine case, and surrounding cowl structure. For disk initiated events, blade fragment velocities are based on penetration of the engine cowl structure only (assumes the disk has compromised the engine containment case). Large disk fragment velocities (fragments weighing more than 10 lbs. are considered large) are based on initial velocity only. Small disk fragment velocities are based on the initial velocity and penetration of the engine case and cowling.

Where event data was unavailable for a failure mode for an engine type, the normalized debris size and trajectory angles from the other engine type were used. This was done in three cases:

- LBPR compressor spacer-rim was scaled for HBPR compressor data
- LBPR compressor disk data was scaled for HBPR compressor data
- HBPR high-pressure turbine data was scaled for LBPR HPT data

Scaling of the data included accounting for different component dimensions (disk diameter and blade length), mass, and rotational speeds.

Normalized Size is the fragment size divided by the blade length or disk diameter respectively.

Weight is the debris weight in pounds. The weight percentage is based on the general rotor characteristics defined in the *Large Engine Uncontained Debris Analysis* report Figure 3-5. Blade weight was divided by the blade weight and disk fragments are divided by disk weight. Disk rim fragments are divided by the disk weight.

Note: The engine fragment characterization tables are based upon engines installed in typical nacelle designs of the type certified in commercial transport designs of the 1970's and 1980's time period. Nacelle skin thicknesses used for the three engine types to determine residual fragment velocities are the following:

TABLE E-1. Nacelle Skin Thickness

	HBPR	LBPR	Turboprop & Turboshaft
Fan ³	0.025 Al 0.025 Al 0.04 Al	0.04 Al	Not Applicable
Compressor	0.04 Al	0.04 Al	Disk 0.1 Steel 0.08 Steel 0.06 Al
Turbine	0.04 Al	0.04 Al	Disks 0.1 Steel 0.08 Steel 0.06 Al Blades 0.06 Al

1. Skin Thicknesses are in inches
2. Turboprop and turboshaft steel skins represent additional ducting and casing around the rotating components.
3. For HBPR fan disk events, blade fragments penetrated only the 0.04" Al.
4. For all blade containment case penetration events, the initial velocity is not reduced by 25%, and the fragments penetrate a 0.25" Steel case then the cowl.

Note: The fan blade fragment model was updated to better represent realistic debris damage characteristics. Updates were made to the number of debris for a given debris size, weight, velocity, and spread angle.

TABLE E-2. Turboprop Engine Fragment Characterization

Component	Number of Events	Number of Fragments (Average/ Event)	Normalized Size	Weight (lbs)	Velocity (ft/sec) at 0° plane	Spread Angle degrees
Fan	N/A					
Compressor						
Blade Event	0	--				
Spacer - Rim	3	1				
Blades						
Rim			89%	2.6	731	±5
Disk Event	1	1				
Blades						
Disk			67%	10.2	580	±5
HP Turbine						
Blade Event	0					
Spacer- Rim	0					
Blades						
Rim	5	1	50%	0.72	541	+5 to -11
Disk Event						
Blades						
Disk	3	1	100%	8.38	533	+5 to -15
LP/PT Turbine						
Blade Event	10		63%	0.1	609	+15 to -15
Spacer- Rim	10	1				
Blades						
Rim			20%	1.05	662	+5 to -11
Disk Event	4	1				
Blades						
Disk			68%	4	572	+5 to -15

+ angles are forward, - angles are aft of the rotor plane of rotation

TABLE E-3. Low Bypass Ratio Engine Fragment Characterization

Component	Number of Events	Number of Fragments (Average/Event)	Normalized Size	Weight Lbs (% of total)	Velocity ft/sec at 0° plane	Spread Angle degrees
Fan						
Blade Event	2	17				
		0	10%	0.07 (2%)		+20 to -10
		3	20%	0.3 (7%)	907	+20 to -10
		6	30%	0.8 (19%)	976	+20 to -10
		3	50%	2.0 (48%)	939	+20 to -10
		5	100%	4.2 (100%)	748	+20 to -10
Disk Event	5					
Blades		17				
		6	10%	0.07 (2%)	1102	+25 to -40
		7	20%	0.3 (7%)	1041	+25 to -40
		1	30%	0.8 (19%)	4021	+25 to -40
		2	50%	2.0 (48%)	955	+25 to -40
		1	100%	4.2 (100%)	762	+10 to 0
Disks		1	100%	41 (41%)	317	+5 to -4
Compressor						
Blade Event	2					
Spacer – Rim	2					
Blades		6.5	100%	0.25 (100%)	642	+15 to -3
Rim		2	80%	6 (30%)	565	+15 to 0
Disk Event	6					
Blades		7.3	100%	0.25 (100%)	642	+15 to -30
Disk (Large Fragment)		1	85%	9 (45%)	334	+5 to -5
Disk (Intermediate Fragment)		1	30%	4 (20%)	460	+10 to -5
HP Turbine						
Blade Event	0	10	80%	0.25 (100%)	336	+20 to -50
Spacer – Rim	0					
Blades		11	85%	0.25 (100%)	871	+15 to -40
Rim		1	50%	8.5 (7%)	1000	0 to -12
Disk Event	0					
Blades		12	70%	0.25 (100%)	871	+15 to -60
Disk		1	30%	10 (8%)	743	+3 to -11

TABLE E-3. Low Bypass Ratio Engine Fragment Characterization (Continued)

Component	Number of Events	Number of Fragments (Average/Event)	Normalized Size	Weight Lbs (% of total)	Velocity ft/sec at 0° plane	Spread Angle degrees
LP Turbine						
Blade Event	5	8.6	50%	0.25 (37%)	378	+15 to -35
Spacer – Rim	1					
Blades		10	50%	0.25 (37%)	889	+20 to -15
Rim		1	56%	6.3 (5%)	918	+5 to -5
Disk Event	5					
Blades		23.4	50%	0.25 (37%)	889	+10 to -70
Disk		2	100%	70 (58%)	571	+3 to -5

+ angles are forward, - angles are aft of the rotor plane of rotation

TABLE E-4. High Bypass Ratio Engine Fragment Characterization

Component	Number of Events	Number of Fragments (Average/Event)	Normalized Size	Weight Lbs (% of total)	Velocity ft/sec at 0° plane	Spread Angle degrees
Fan						
Blade Event (Helical)	11	7				
		3	10%	0.33 (3%)	904	+35 to -35
		2	20%	2.0 (16%)	895	+20 to -45
		1	30%	3.75 (30%)	877	+20 to -30
		1	50%	6.24 (50%)	808	+15 to -30
		0	70%	8.61 (70%)		
		0	100%	12.5 (100%)		
Disk Event	3					
Blades		27				
		10	10%	0.33 (3%)	935	+10 to -30
		9	20%	2.0 (16%)	928	+15 to -25
		2	30%	3.75 (30%)	894	+10 to -25
		4	50%	6.24 (50%)	822	+10 to -20
		1	70%	8.61 (70%)	796	+10 to -20
		1	100%	12.5 (100%)	644	+15 to +5
Disks		3	100%	45 (38%)	303	+2 to -3
Compressor						
Blade Event	0					
Spacer – Rim	1					
Blades		6.5	100%	0.25 (100%)	642	+15 to -3
Rim		2	80%	6.0 (30%)	523	+15 to 0
Disk Event	2					
Blades		4.5	50%	0.19 (76%)	609	+5 to -25
Disk (Large Fragment)		1	85%	9 (45%)	380	+5 to -5
Disk (Intermediate Fragment)		1	30%	4 (20%)	385	+10 to -5

TABLE E-4. High Bypass Ratio Engine Fragment Characterization (Continued)

Component	Number of Events	Number of Fragments (Average/Event)	Normalized Size	Weight Lbs (% of total)	Velocity ft/sec at 0° plane	Spread Angle degrees
HP Turbine						
Blade Event	1	10	80%	0.25 (100%)	337	+20 to -50
Spacer – Rim	3					
Blades		11	85%	0.25 (100%)	871	+15 to -40
Rim		1	50%	10 (83%)	967	0 to -12
Spacer		1	50%	2 (16%)	781	+20 to -37
Disk Event	5					
Blades		12	70%	0.25 (100%)	871	+15 to -60
Disk		1	30%	10 (8%)	967	+3 to -11
LP Turbine						
Blade Event	8	6.5	50%	0.25 (37%)	212	+20 to -45
Blade Event Last Stage	4	6.5	50%	0.25 (37%)	200	0 to -75
Spacer – Rim	2					
Blades		5	15%	0.05 (7%)	326	+6 to -20
Rim		1	20%	11.3 (9%)	505	+5 to -5
Disk Event						
Blades		5	28%	0.25 (20%)	313	+5 to -40
Disk		1	20%	18 (15%)	535	+3 to -5

+ angles are forward, - angles are aft of the rotor plane of rotation

The uncontained failure event rates by component set forth in the Table below are taken from the AIA PC 342-1, Committee on Continued Airworthiness Assessment Methodology (CAAM) Supplemental Report on Turbine Engine Uncontained Events dated 5 February 1997. The event rates cover the period 01/01/1982 through 11/30/96 and are the most current rates available. The data used to characterize fragment hazards to the aircraft were obtained from all uncontained engine - aircraft events where fragment information was available, including those from earlier time periods. There is no agreement in the event counts between these two sets of data and such an agreement should not be assumed to exist. For the Table below the uncontained events for high and low pressure compressors were combined under the heading - Compressor. Similarly, the uncontained events for high and intermediate pressure turbines were combined under the heading - HP Turbine.

TABLE E-5. Uncontained Event Rate Table

Component	Turboprop		Low Bypass Ratio		High Bypass Ratio	
	Number of Events	Rate *	Number of Events	Rate *	Number of Events	Rate *
Fan						
Blade Event			26	2.2E-7	90	10.1E-7
Disk Event			8	0.68E-7	4	0.45E-7
Compressor						
Blade Event	1	0.08E-7	13	1.1E-7	2	0.22E-7
Spacer – Rim	0		6	0.51E-7	1	0.11E-7
Disk Event	10	0.84E-7	10	0.84E-7	10	1.1E-7
HP Turbine						
Blade Event	7	0.59E-7	4	0.34E-7	2	0.22E-7
Spacer – Rim	2	0.17E-7	0		4	0.45E-7
Disk Event	18	1.5E-7	0		6	0.67E-7
LP Turbine						
Blade Event	10	0.84E-7	48	4.1E-7	64	7.2E-7
Spacer – Rim	0		0			
Disk Event	5	0.42E-7	7	0.59E-7	5	0.56E-7

* Per Aircraft Flight

Based on 118.381E+6 Turboprop Aircraft Flights, 118.346E+6 Low Bypass Ratio Turbofan Aircraft Flights and 89.269E+6 High Bypass Ratio Turbofan Aircraft Flights in this calendar period.

APPENDIX F

DYNAMIC SHEAR CONSTANT (C_s) DEVELOPMENT

This appendix contains an excerpt from the following report: Manion, John, *Analysis of the Shear Constant (Cs) Used in FAA Ballistic Limit Velocity Equation (DRAFT)*, SURVICE-TR-06-034, SURVICE Engineering, Valparaiso, Florida, September 2006. Included is a discussion of the development of the original C_s value used for Aluminum in the penetration equations employed by UEDDAM as well as a proposed change to the C_s value and what effect this would have on a typical UEDDAM analysis.

BACKGROUND

The Federal Aviation Administration (FAA) initiated a research program, the Uncontained Engine Debris Mitigation Program (UEDMP), to investigate and determine methods to mitigate the damage caused by uncontained engine debris. Damage from such an event is often considerable, but not necessarily catastrophic as long as structural integrity is maintained, fires are not sustained, and critical systems do not become inoperable. The UEDMP, managed by the Airport and Aircraft Safety Research and Development Division located at the FAA William J. Hughes Technical Center, Aircraft Catastrophic Failure Prevention Program (ACFPP), works with industry and government to determine possible engineering solutions to reduce injuries and critical damage resulting from uncontained engine events. As part of this program, the Naval Air Warfare Center-Weapons Division (NAWC-WD) was tasked to evaluate ballistic damage analysis tools and vulnerability (damage) reduction techniques currently in use within the Department of Defense (DoD). From these tools, the Uncontained Engine Debris Damage Assessment Model (UEDDAM) was built.

The UEDDAM tool was developed based on DoD vulnerability analysis tools. NAWC-WD tasked the SURVICE Engineering Company to design and code the UEDDAM application as an aircraft design and certification tool intended to address the uncontained engine debris hazard. The UEDDAM application was built upon the existing DoD tools (Computation of Vulnerable Area and Repair Time (COVART) and Fast Shotline Generator (FASTGEN)). These codes were modified to better describe an uncontained engine event and to permit unlimited distribution of the tools. Within the UEDDAM application, penetration equations are used to determine the change in state of the fragment as a result of impacts to aircraft skin and components along the fragment trajectory. These penetration equations require test data to derive empirical constants.

Testing has provided the fragment penetration characteristics in terms of fragment orientation at impact, impact velocity, and fragment residual velocity. Testing at NAWC-WD has been conducted in support of the UEDDAM code, specifically in developing the penetration equations and constants to model the impacts of engine fan blades. Three series of tests have been completed to date.

The first test series investigated small (less than 2-in. square) to medium-sized (3-in. by 8-in.) fragments impacted into aluminum plates and engine cowlings (Ref. F-1^{††}).

^{††} Ref. F-1. Manchor, J. and C. Frankenberger, "Engine Debris Penetration Testing," FAA report, DOT/FAA/AR-99/19, June 1999.

Performed in 1998, this early testing also investigated the accuracy of several ballistic impact prediction methods, accepting both the Joint Technical Coordinating Group on Munitions Effectiveness (JTTCG/ME), residual velocity (V_r), and ballistic limit (V_{50}) equations as reasonable prediction tools for fan blade impacts.

The second series, the following year, investigated small- to medium-sized fragments impacted into an actual narrow-body commercial aircraft fuselage, denoted as Fuselage Test Phase I (Ref. F-2^{‡‡}).

The third series, in 2000, impacted medium- to large-sized (8-in. by 8-in.) fragments into the same fuselage (Ref. F-3^{§§}). During the fuselage testing, the interaction of various aircraft structural elements apparently created some disparity in the accuracy of the predictions of the penetration equations. Analysis of this phenomena determined that the V_{50} equation developed from the FAA Energy Equation was a more effective prediction tool for both single skin and complex structural impacts. As a result of this test series, the FAA V_{50} equation replaced the JTTCG/ME V_{50} equation in UEDDAM.

The FAA Ballistic Limit Equation used in UEDDAM is:

$$V_{50} = \sqrt{\frac{2 \cdot L \cdot C_s \cdot t^2}{m \cdot \cos^2(\theta)}}, \quad (1)$$

where:

- L = Presented area perimeter (m),
- C_s = Shear constant (Pa),
- t = Target thickness (m),
- m = Fragment mass (kg),
- θ = Obliquity angle (degrees),
- V_{50} = Ballistic limit velocity (m/s).

Following an impact, the residual velocity of the fragment is predicted within UEDDAM using the following formula:

$$V_{r, \text{Predicted}} = \frac{\sqrt{V_i^2 - V_{50}^2}}{1 + \frac{\rho \cdot A_p \cdot t}{m \cdot \cos(\theta)}}, \quad (2)$$

where:

- $V_{r, \text{Predicted}}$ = Penetration equation prediction for residual velocity (m/s),
- θ = Obliquity angle (degrees),
- m = Fragment mass (kg),
- t = Target thickness (m),
- A_p = Fragment presented area (m^2),

^{‡‡} Ref. F-2. Lundin, S., “Engine Debris Fuselage Penetration Testing Phase I,” FAA report, DOT/FAA/AR-01/27, September 1999.

^{§§} Ref. F-3. Lundin, S., “Engine Debris Fuselage Penetration Testing Phase II,” FAA report, DOT/FAA/AR-01/27, II, August 2002.

- ρ = Target material density (kg/m³),
- V_{50} = Ballistic limit velocity (m/s),
- V_i = Fragment initial velocity (m/s).

The test series data was used to empirically derive the shear constant (C_s) in the FAA ballistic limit equation. The method of C_s derivation will be discussed in detail later in this report.

Recent analysis of the test series data has brought into question the ability of these equations to accurately predict penetration ($V_{r, Predicted}$) of the fragments following impacts. Specifically, the methods used to derive the value of C_s and the resultant errors in $V_{r, Predicted}$ have been questioned.

PURPOSE

The purpose of this report is to independently review the C_s derivation method and investigate alternate methods of deriving C_s from the test data that may reduce the overall error and/or increase the confidence in penetration prediction. Another important issue investigated in this report is the sensitivity of UEDDAM results to small changes in C_s that may occur by empirically deriving C_s using different methods.

The scope of this analysis is limited to the material 2024 aluminum.

DERIVATION OF SHEAR CONSTANT

CURRENT METHOD

The currently accepted value of C_s for aluminum of 276 MPa that is used in UEDDAM for penetration calculations was derived using the following methodology.

The penetration test data were used to determine the C_s that would produce the best agreement between the actual and predicted residual velocities. The process used was multifold and required a computation of the root mean squared (rms) error (E) and the correlation factor (Q), the simple percent error in residual velocities, and a measure of conservatism. An Excel spreadsheet was created that solved the penetration equations for all the impact data and then determined an rms error for the Q factors, where Q is defined as

$$Q = 1 - \frac{V_{r,Test} - V_{r,Predicted}}{V_i} \tag{3}$$

where:

- Q = Correlation factor
- V_i = Test impact velocity (m/s),

$$\begin{aligned} V_{r, \text{Predicted}} &= \text{Penetration equation prediction for residual velocity (m/s),} \\ V_{r, \text{Test}} &= \text{Test residual velocity (m/s).} \end{aligned}$$

This correlation factor describes the difference between measured test residual velocities and predicted residual velocities, normalized by the initial velocity, and offset to a scale of 1. Equation 3 produces values between 0 and 2, where a value less than 1 represents an under-prediction and greater than 1 represents an over-prediction. Consequently, a value of 1 describes a perfect prediction. For each material, Q is calculated with respect to shot number. Since Q is a function of residual velocity, only test shots where a penetration was both predicted to occur and actually occurred are considered (*this turns out to be an important distinction, as will be discussed later*).

The rms error (E) is defined as:

$$E = \sqrt{\frac{1}{n} \cdot \sum_{k=1}^n \left(\frac{V_{r, \text{Test}_k} - V_{r, \text{Predicted}_k}}{V_{i_k}} \right)^2} \quad (4)$$

or

$$E = \sqrt{\frac{1}{n} \cdot \sum_{k=1}^n (1 - Q_k)^2} \quad (5)$$

where:

$$\begin{aligned} E &= \text{rms error in predictions,} \\ n &= \text{Number of test shots,} \\ V_{i_k} &= \text{Impact velocity of test k (m/s),} \\ V_{r, \text{Predicted}_k} &= \text{Penetration equation prediction for residual velocity of test k (m/s),} \\ V_{r, \text{Test}_k} &= \text{Residual velocity of test k (m/s).} \end{aligned}$$

To derive the appropriate C_s value, a typical iterative technique to search for the minimum value of E is used. This technique starts by setting C_s to an assumed value and calculating $V_{r, \text{Predicted}}$ for each test shot using equations 1 and 2 in this report. Second, E is calculated over all of the test shots using equation 4. Next, a new value of C_s is chosen and E is determined. This is continued until E is minimized. This process was automated with the *Solver* function in Excel. Any optimizer solution must be checked for local minima conditions since they can lead to false solutions. After a solution is found, the C_s is varied manually to explore the region for false minima.

When the test V_i is lower than the V_{50} , the value of E is excluded from the averaging process. This is done for purely mathematical reasons since the result for $V_{r, \text{Predicted}}$ becomes imaginary. In reality, if V_i is less than V_{50} , the value of $V_{r, \text{Predicted}}$ should be zero.

The test data used to estimate the currently accepted value of C_s (276 MPa) covers the first test series performed in 1998 (Ref. F-1), the Fuselage Phase I testing (Ref. F-2), and the Fuselage Phase II testing (Ref. F-3). A summary of the shot identifications used to calculate C_s is shown in Table F-1.

Table F-1. Summary of Test Shots Used for C_s Derivation

Test Series	Shot I.D. Numbers
Aluminum Plate Tests (Ref. F-1)	3, 4, 5, 7, 8, 9, 10, 23, 24, 64, 65, 66, 67, & 68
B-727 Phase I Tests(Ref. F-2)	5, 6, 8, 9, 12, 13, 14, 15, 16, 17, 17.5, 19, 20, 22, 23, 24, 25, 26.5, 33, 35, 37, 54, 55, 56, 57, 58, & 59
B727 Phase II Tests (Ref. F-3)	1, 2, 3.1, 4, 5, 8, 9, 10, 11.1, 12, 14, 15, 16, 23, 24, 25, 26, 27, 28, 37, 38, 39, 40, 41, & 42

The actual test data for the shots listed in Table F-1 are summarized at the end of this report (see Tables F-7 thru F-9).

Using the currently accepted value of C_s (276 MPa) and the data from the tests indicated in Table F-1, E is calculated to be 0.098 and there exists an average percent error of -7.11% over all of the test data.

ISSUES WITH CURRENT METHOD

Since some of the Phase I and Phase II test shots hit secondary structure (stringers and spars), it was postulated that the value of C_s may have been skewed. The reason for this postulation is that the value of $V_{r, \text{Predicted}}$ for the tests that hit secondary structure assumed a material thickness equal to the skin thickness when in reality, the thickness was the skin plus the secondary structure.

Previous conclusions in Reference F-3, however, state that the secondary structure was not a significant energy absorber and therefore should not skew the data. If the assertion in Reference F-3 is true, then the value of C_s should not change significantly when looking only at the data that did not hit the secondary structure.

To address this issue, values of C_s were derived using the existing methodology from the aluminum plate tests only (Table F-9) and from all of the test data that did not perforate secondary structure. The test shots that did not hit secondary structure are denoted as “skin only” tests in this report. A summary of the “skin only” tests used to derive C_s are shown in Table F-2.

Table F-2. Summary of “Skin Only” C_s Test Shots

Test Series	Shot I.D. Numbers
Aluminum Plate Tests (Ref. F-1)	3, 4, 5, 7, 8, 9, 10, 23, 24, 64, 65, 66, 67, & 68
B-727 Phase I Tests(Ref. F-2)	5, 6, 12, 16, 17, 17.5, 19, 20, 23, 24, 25, 35, 37, 57, 58, & 59
B727 Phase II Tests (Ref. F-3)	1, 2, 23, 27, & 28

The values of C_s , E, and percent error generated from the Aluminum plate tests and the “skin only” shots are listed in Table F-3. For comparison purposes, the currently accepted values of C_s , E, and percent error are also shown in Table F-3.

Table F-3. Values of C_s Derived from Selected Test Shots

Test Series	C_s (MPa)	E Within Data Set	% Error Within Data Set
Aluminum Plate Tests (Ref. F-1)	255	0.028	-1.19
“Skin Only” Shots (Test Shots listed in Table F-2)	265	0.066	-3.16
Currently Accepted Value (Test Shots listed in Table F-1)	276	0.098	-7.11

From Table F-3 it can be seen that, depending on which subset of data is used, the value of C_s changes. This could imply that the secondary structure is a significant energy absorber and should be considered in the C_s derivation. This implication, however, contradicts the conclusions of Reference F-3 which stated that “these data show that the secondary structures are not significant energy absorbers when a small to large fragment impacts an aircraft structure.”

To address this issue, a closer examination of the C_s minimization methodology is warranted.

From Reference F-3, the solution of determining an appropriate value of C_s uses the Excel worksheet *Solver* function that minimizes the value of E. This is an automated process to search for local minima. Once the solver finds the minima, the value of C_s was varied manually about that point to explore the region for false minima. It is not clear as to how encompassing that search was, so a plot of E versus C_s was performed over a wide range of C_s values and is illustrated in Figure F-1. Note that the plot in Figure F-1 is for the entire data set list in Table F-1.

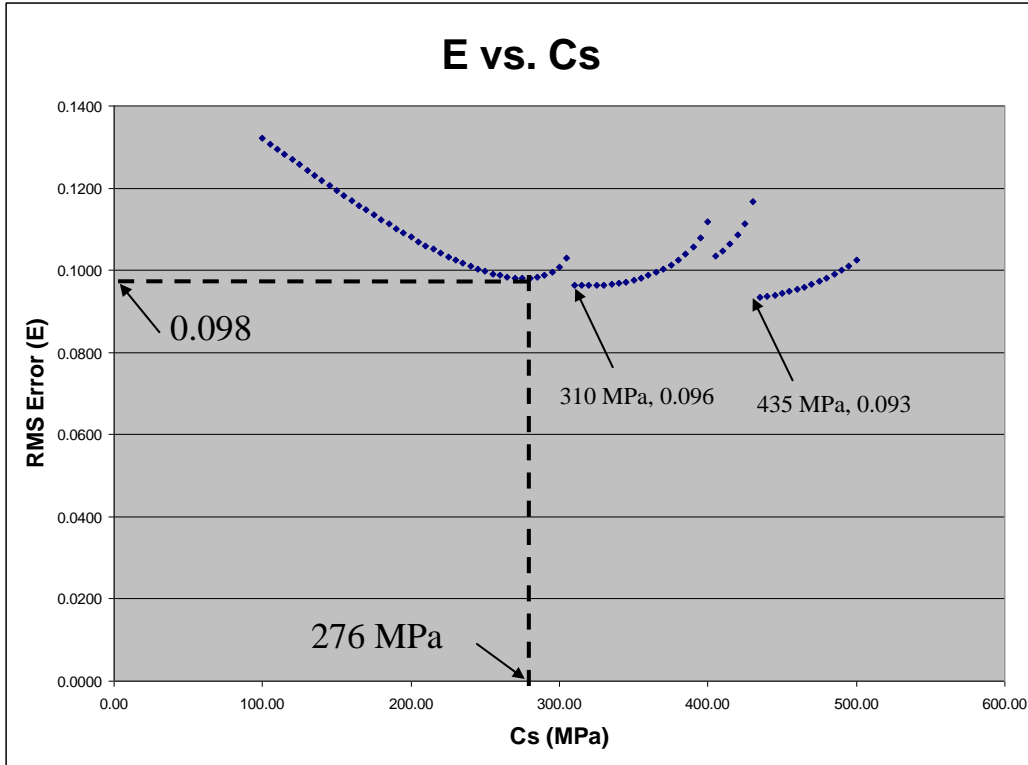


Figure F-1. Graph of E vs. C_s for Test Shots Listed in Table F-1.

From Figure F-1, there are clearly other minimums at 310 MPa and 435 MPa that result in an RMS error (E) less than the currently accepted value of 276 MPa. Other interesting anomalies from Figure F-1 are the three large discontinuities in the function at C_s values of 310, 405, and 435 MPa.

The same plots were made for the aluminum plate tests and the “skin only” tests listed in Table F-2 and are shown in Figures F-2 and F-3, respectively.

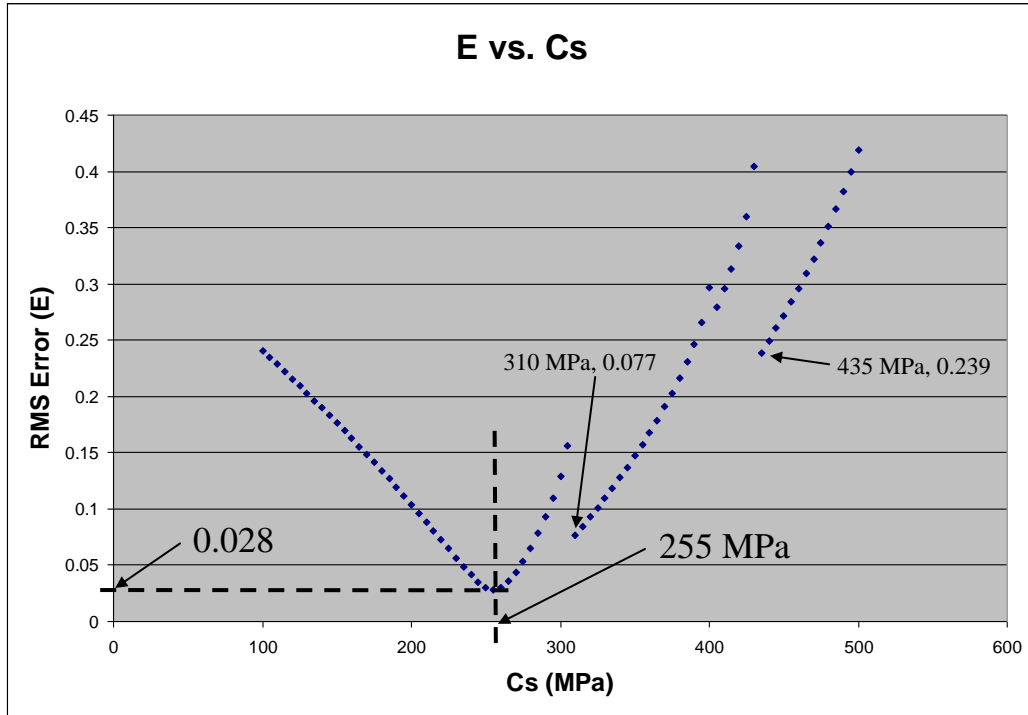


Figure F-2. Graph of E vs. C_s for Aluminum Plate Test Shots Only.

From Figure F-2, there are other minimums but the one that results in the lowest value of E is the C_s value of 255 MPa. Again, note the large discontinuities in the function at 310, 405, and 435 MPa.

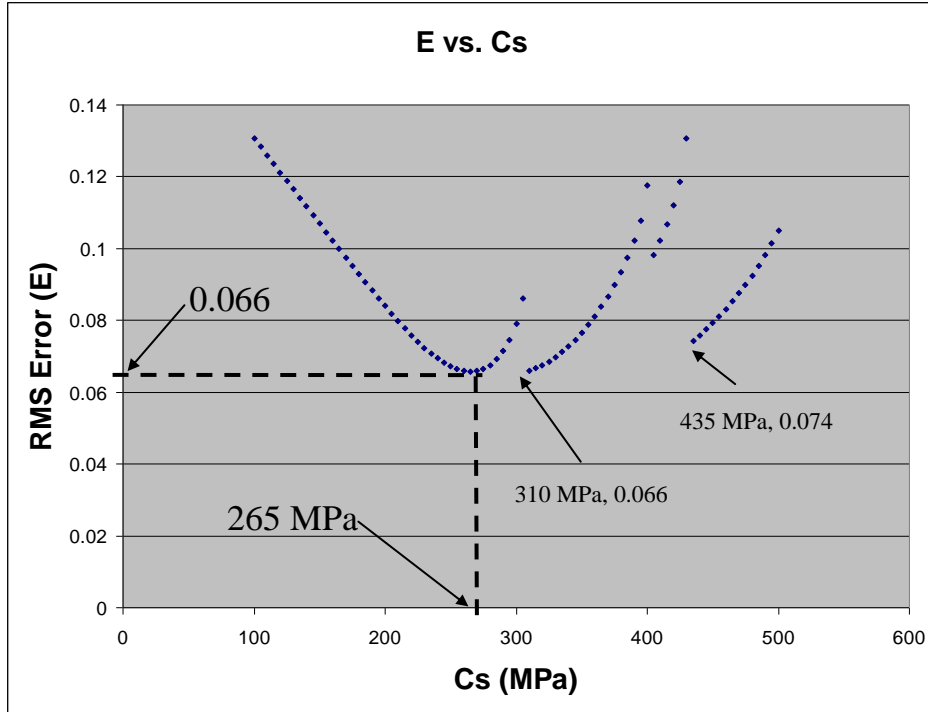


Figure F-3. Graph of E vs. C_s for “Skin Only” Test Shots.

From Figure F-3 there are other minimums but the one that results in the lowest value of E is the C_s value of 265 MPa. Again, note the large discontinuities in the function at 310, 405, and 435 MPa.

The results in Figures F-1 thru F-3 imply that another value of C_s (e.g., 310 MPa) may be better suited to representing the test data. The value of C_s that results in the lowest minima and is consistent throughout the data sets is 310 MPa. This consistency aligns with the assertion that C_s should not vary significantly when comparing shots that hit secondary aircraft structure and shots that did not. The large discontinuity in the E vs. C_s functions at $C_s = 310$ MPa, however, brings into question the authenticity of the minima at that point.

A closer look at the data and methodology reveals that the reason for the discontinuity at $C_s = 310$ MPa is that at that point, the value of $V_{r, \text{Predicted}}$ for shot #23 of the aluminum plate tests is equal to zero. Since the predicted residual velocity is zero, the Q value associated with $V_{r, \text{Predicted}}$ and $V_{r, \text{Test}}$ is not included in calculating E even though the value of $V_{r, \text{Test}}$ is non-zero. This is as a direct result of the RMS error calculation methodology. Looking at $C_s = 405$ MPa, a similar observation is made: the discontinuity is due to $V_{r, \text{Predicted}} = 0$ for shot #66 of the aluminum plate test and is thus excluded from the calculation of E. Finally, the discontinuity at $C_s = 435$ MPa is due to $V_{r, \text{Predicted}} = 0$ for shot #68 of the aluminum plate test data. Note that because the aluminum plate test data is included in each of the data sets in Figures F-1 thru F-3, the discontinuities occur at the same C_s values in all three.

So, excluding the error associated with a test where $V_{r, \text{Predicted}} = 0$ but $V_{r, \text{Test}} \neq 0$ is not valid since there is a real error associated with that prediction that must be accounted for. Therefore, we cannot conclude that the minimum at $C_s = 310$ MPa is any more valid than the accepted value of $C_s = 265$ MPa as both values are based on minimizing error.

PROPOSED NEW METHOD

The proposed new method for computing C_s follows the same methodology to calculate E and minimizes E with respect to C_s as previously discussed. The difference is that instead of excluding Q values from the E calculation where either $V_{r, \text{Predicted}} = 0$ or $V_{r, \text{Test}} = 0$, all Q values when either of the two residual velocities are greater than zero are included in the E calculation. Note that if $V_{r, \text{Predicted}}$ and $V_{r, \text{Test}}$ are both equal to zero, the respective Q is still excluded from the calculation of E .

Using this new methodology, Figures F-4 thru F-6 were generated for all of the shots in Table F-1, the aluminum plate test only, and the “skin only” shots, respectively.

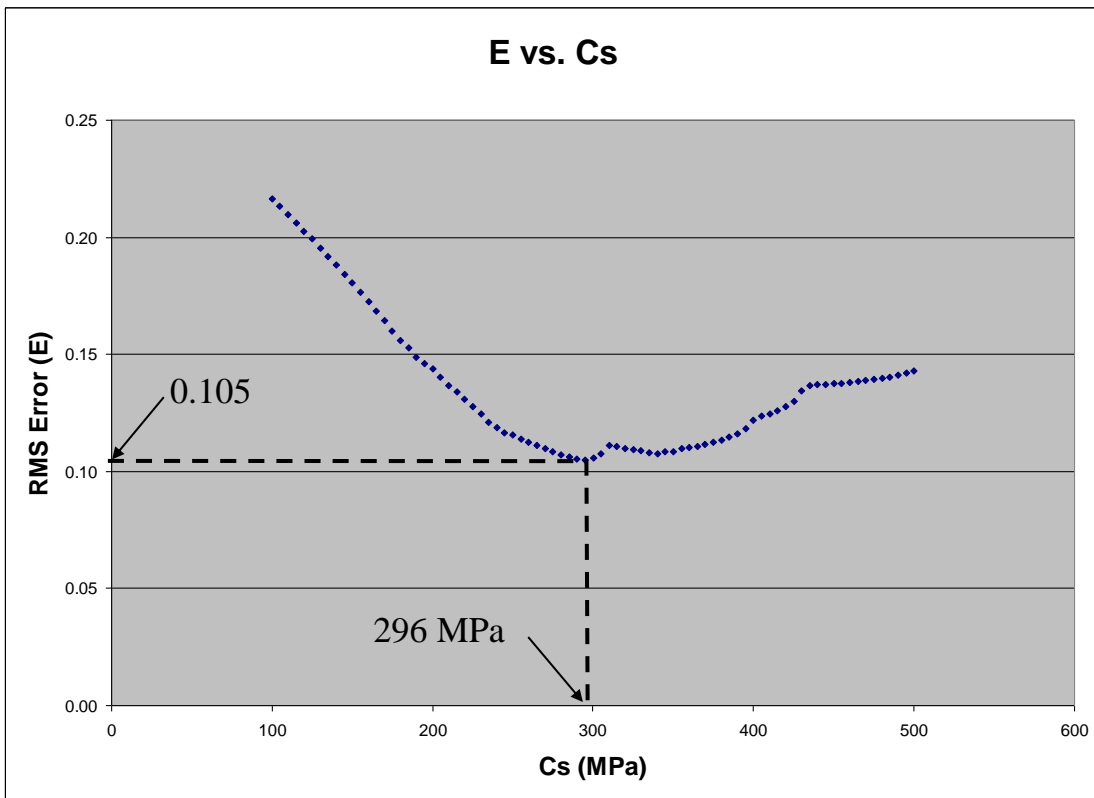


Figure F-4. Graph of E vs. C_s for Test Shots Listed in Table F-1 Using the Proposed New Method.

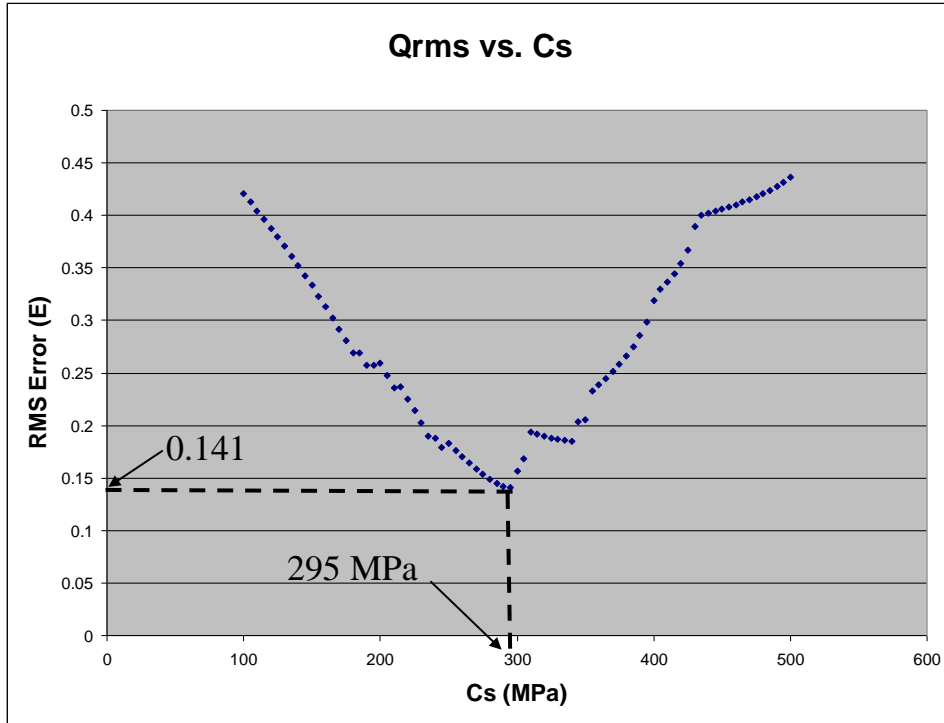


Figure F-5. Graph of E vs. C_s for Aluminum Plate Test Shots Only Using the Proposed New Method.

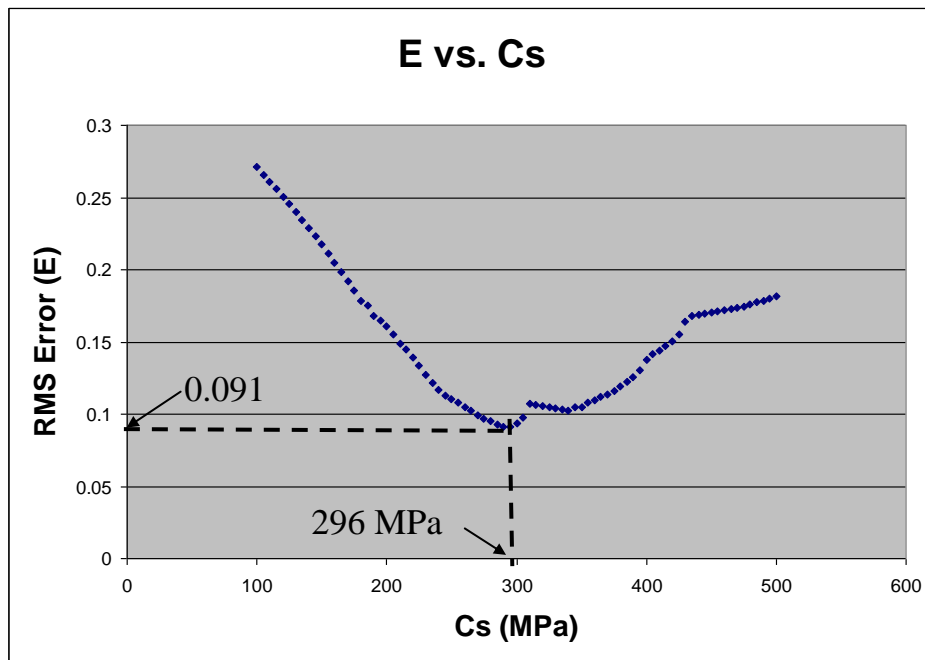


Figure F-6. Graph of E vs. C_s for “Skin Only” Test Shots Using the Proposed New Method.

From Figures F-4 thru F-6, it can be seen that although there are discontinuities in the slope of the data, there are not discreet jumps in the graph of E vs. C_s as was the case using the original methodology. Note also from Figures F-4 thru F-6 that the minimum E occurs at a consistent value of approximately 296 MPa regardless of the subset of data used. This is consistent with the assertion that C_s should not vary significantly when comparing shots that hit aircraft structure and shots that do not.

Table F-4 summarizes the values of C_s , E, and average percent error for the three data sets used to produce Figures F-4 thru F-6.

Table F-4. Values of C_s Derived from Selected Test Shots Using the Proposed New Method

Test Series	C_s (MPa)	E Within Data Set	% Error Within Data Set
Aluminum Plate Tests	295	0.141	19.50
Test Shots listed in Table F-2 (Skin Only)	296	0.091	0.22
Test Shots listed in Table F-1	296	0.105	-5.91

Note from Table F-4 that overall, the value of E has an apparent increase over using the old methodology. This is due to the fact that all errors in velocity estimation are included, even the errors based on predicting a zero residual velocity.

RECOMMENDATIONS

The proposed new method described in the previous section of this report minimizes the overall error while allowing for some error in each of the individual predictions. This method also considers all errors where the predicted or actual residual velocity is greater than zero and is therefore more complete than the original method. It is recommended that the proposed new method be used to update the value of C_s for all of the material types tested to date. The C_s value for aluminum based on this new method is 296 MPa. It is recommended that this new value for aluminum be used in UEDDAM for penetration calculations through aluminum.

SENSITIVITY OF C_s TO MEASURED DATA

Other factors, besides the methodology, that can affect the value of C_s are errors in the measured test data. These factors are investigated in this section by assessing the sensitivity of C_s to the test parameters used to derive the value of C_s . To do this, equations (1) and (2) were simultaneously solved for C_s to obtain the following relation:

$$C_s = \frac{m \cdot \cos^2(\theta)}{2 \cdot L \cdot t^2} \left(V_i^2 - V_r^2 \left[1 + \frac{\rho \cdot A_p \cdot t}{m \cdot \cos(\theta)} \right]^2 \right) \quad (6)$$

where:

- C_s = Shear constant (Pa),
- L = Presented area perimeter (m),
- t = Target thickness (m),
- m = Fragment mass (kg),
- θ = Obliquity angle (degrees),
- V_r = Residual velocity (m/s),
- A_p = Fragment presented area (m²),
- ρ = Target material density (kg/m³),
- V_i = Fragment initial velocity (m/s).

From the test data set, a representative test shot that resulted in a value of C_s close to the proposed value of 296 MPa was selected for this analysis. The test shot selected is #67 of the aluminum panel testing (see Table F-5).

Table F-5. Test Parameters for Shot #67 of the Aluminum Panel Test Series

Shot #	m (kg)	Avg. Frag Length (in.)	Avg. Frag Width (in.)	V_i (ft/s)	V_r (ft/s)	L (in.)	A_p (in ²)	θ (deg)	t (in)	C_s (MPa)
67	0.679	7.75	7.38	536	339	16.79	3.66	0	0.25	289.7

From this data set, the values of A_p , L , t , V_i , and V_r were each varied by $\pm 20\%$ of their original values in 1% increments and C_s was calculated at each point. Figure F-7 shows a plot of the percent change in C_s as a function of the percent change of the selected test parameters.

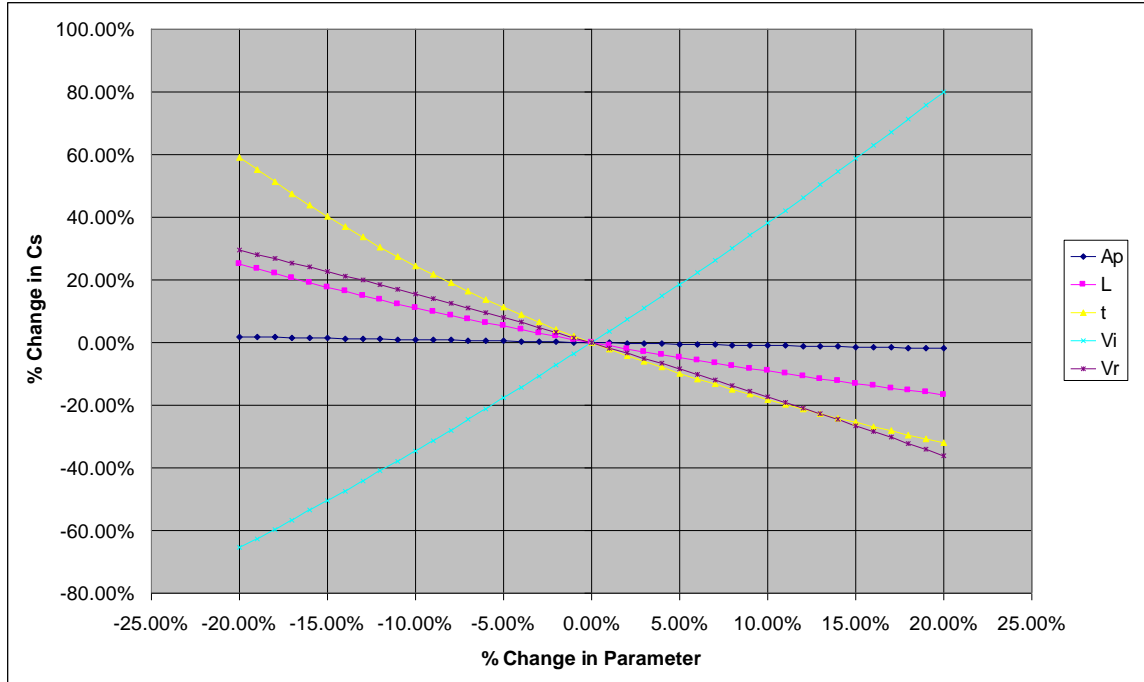


Figure F-7. Sensitivity of C_s to Selected Test Parameters.

From Figure F-7, C_s can be seen to be quite sensitive to changes (i.e., errors) in V_i throughout the range of values analyzed. A 20% change in V_i can affect the calculated C_s by as much as 80%. It is therefore of the highest importance to ensure the velocity measurements are accurate in the tests. The material thickness, t , is the second highest driver in C_s sensitivity. As t decreases from the nominal value, C_s becomes more sensitive to changes in t . Therefore, for thin skin, it is very important to accurately measure the thickness at the perforation location. C_s is also quite sensitive to changes in V_r . Like V_i , then, it is important to accurately measure V_r . A 20% change in presented area perimeter, L , can have an equal affect on the value calculated for C_s . Surprisingly, C_s is not very sensitive to A_p in the region analyzed.

Based on this analysis, the most sensitive parameters affecting C_s are V_i , V_r , and t . It is therefore paramount that these parameters be accurately measured from the test data to ensure that the value of C_s is accurate.

The value of C_s is least sensitive to the parameters L and A_p . This turns out to be beneficial since it is L and A_p that are most likely to have significant errors. The values of L and A_p are actually derived from measurements taken from high speed film of the orientation of the fragment just prior to impact. These measurements compute the projected area and projected perimeter of the fragment onto the material being hit. In actuality, the parameters A_p and L physically represent the area and perimeter of the hole but it is assumed that the projected area and projected perimeter of the fragment is suitable to represent this area. The FAA reviewed selected test photographs and determined that the actual hole and perimeter produced from the impact did differ from

the presented area and perimeter of the fragment in some cases. The calculated perimeter length, L , of the fragment varied on average from the actual perimeter length of the hole by about 11%. This could have an impact on the computed value of C_s of about 10%.

SENSITIVITY OF UEDDAM RESULTS TO C_s

With the potential changes in the value of C_s and the uncertainty in the final value, the question arises as to the sensitivity of UEDDAM results to changes in C_s . This section describes a preliminary sensitivity study designed to address this question.

APPROACH

Due to time and funding constraints, this was a very limited study. The aircraft used for this study was the generic twin-engine aircraft provided in the UEDDAM v3.0 sample case data. A single flight mode and only the first stage of the fan section of one engine were assessed.

Three debris sets were investigated; each set assessing eight values of C_s . This resulted in a run matrix of 24 UEDDAM runs. The value of C_s was varied using the material modification input feature made available in UEDDAM v3.0. Table F-6 summarizes the parameters used in this analysis.

Table F-6. C_s Sensitivity Study Parameters

Parameter	Values
Aircraft	Generic twin engine aircraft
Flight mode	Single flight mode = V_1 to V_{1+30s}
Uncontained debris location	Stage one of engine one fan section
Debris	1) one-third disk segment 2) One-third disk segment and 28 small fragments 3) 28 small fragments
Release parameters	Releases from 0° - 360° in 5° increments 50 iterations per release point
Segment grid parameters	2" main grid 0.25" hyper-sample grid
C_s values	276 MPa 211 MPa 247 MPa 255 Mpa 265 Mpa 295 Mpa 350 Mpa 476 MPa

The three debris sets were intended to cover the spectrum of fragment penetration capability. The one-third disk segment represents the toughest fragment; the 28 small

fragments represent the low-end fragments; and the one-third disk segment and 28 small fragments represent a mix of fragments.

The C_s values were selected cover the range of values that have been derived from various sets of data using various methodologies.

It is expected that the one-third disk segment case should be relatively insensitive to C_s due to the ability of the large fragment to perforate easily through aircraft structure. The 28 small fragments case should be the most sensitive to C_s changes since the fragment energies are more likely to be degraded by aircraft structure. Finally, it is anticipated that the one-third disk segment and 28 small fragments case should be somewhere in between the other two cases in terms of C_s sensitivity.

RESULTS

UEDDAM was run for each of the three debris types for each of the eight C_s values shown in Table F-6. The results of each of the 24 UEDDAM runs are plotted in Figure F-8.

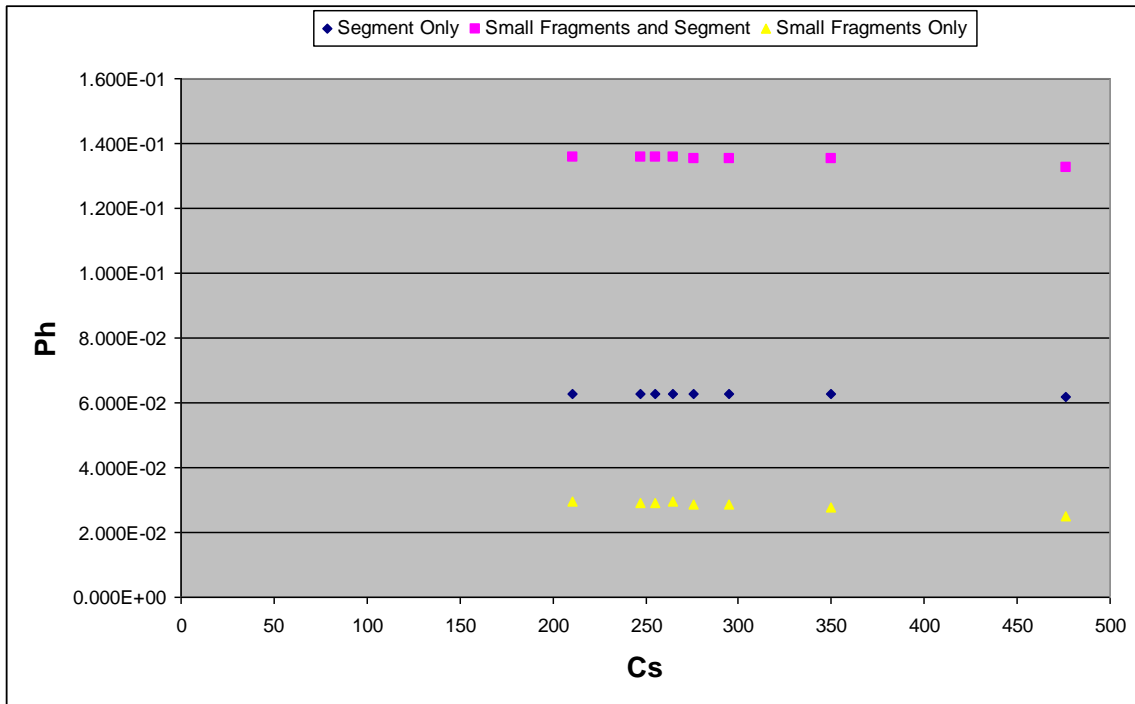


Figure F-8. UEDDAM Sensitivity to C_s Results.

From Figure F-8, the effect of varying C_s on UEDDAM results over the values assessed is clearly very small. To amplify the differences, the results were also plotted in Figure F-9 as a percent change in P_{HAZ} vs. the percent change in C_s .

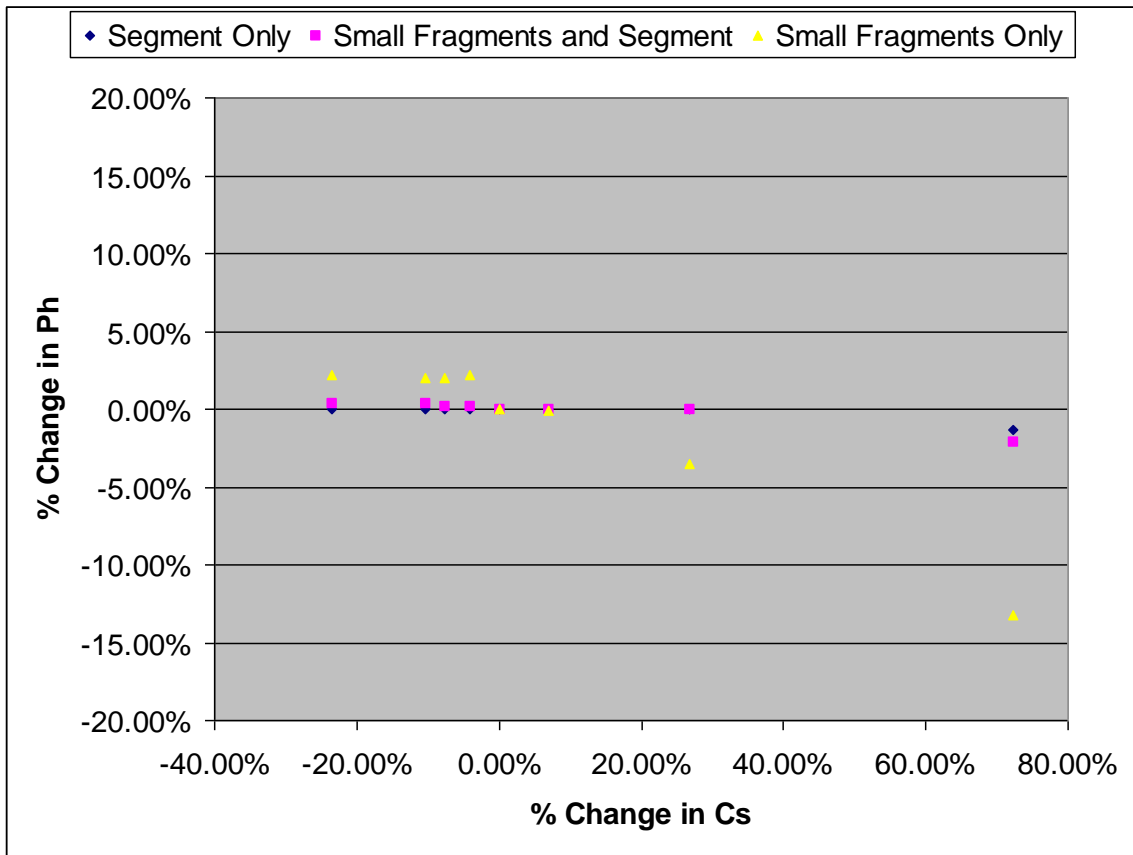


Figure F-9. Percent Change in P_{HAZ} vs. Percent Change in C_s .

From Figures F-8 and F-9, then, the results are as expected. The only significant impact of varying C_s values on UEDDAM results is for the debris set that only includes small fragments. Note, however, that even for the 28 small fragments case, a change in C_s on the order of 55%–60% is required for a change in P_{HAZ} of 10%. The proposed new value of C_s (296 MPa) is only 7% different than the current value (276 MPa).

CONCLUSIONS

This sensitivity study was extremely limited. It only considered a single stage of one engine and one flight mode. Therefore, the results can not be treated as conclusive. A much more rigorous study is required to achieve conclusive results.

However, the fact that this limited study achieved results that were expected gives a good indication that UEDDAM results should be insensitive to variations in C_s on the order of $\pm 30\%$ about 276 MPa. Beyond that value, the UEDDAM results should only be slightly sensitive for the smaller fragments.

TEST SHOT SUMMARIES

Table F-7. B727 Phase I Test Shot Summaries (Ref. F-2)

Shot #	m (g)	Frag Len avg (x-axis) (in)	Frag Width avg (z-axis) (in)	V _i (ft/s)	V _r (ft/s)	L (in)	A _p (in ²)	θ (deg)	t (in)	FAA V ₅₀ (ft/s) (C _s =276 MPa)	FAA JTCC plg V _r (ft/s)	(Q-1) ²
5	299.1	7.63	3.00	572	421	12.51	9.74	7.0	0.114	231.89	446.99	0.0021
6	333.5	9.44	3.00	515	462	6.64	0.77	3.0	0.142	197.37	468.77	0.0002
8	295.8	7.63	3.00	813	641	6.94	1.41	8.0	0.184	280.96	733.45	0.0129
9	180.9	4.70	3.00	892	735	7.17	1.74	8.0	0.184	365.18	752.77	0.0004
12	127.2	4.75	2.75	658	613	5.88	0.53	8.0	0.085	181.11	622.56	0.0002
13	319.0	7.25	3.00	655	522	7.80	2.68	8.0	0.094	145.75	616.40	0.0208
14	314.8	8.75	3.00	802	620	9.83	5.72	8.0	0.101	177.92	721.34	0.0160
15	306.0	7.44	3.00	867	541	9.10	4.55	8.0	0.121	208.02	777.56	0.0744
16	335.2	8.00	3.00	367	263	9.30	2.10	8.0	0.114	188.47	304.99	0.0131
17	366.0	7.19	3.00	505	441	7.70	2.52	8.0	0.088	126.52	475.77	0.0047
17.5	184.6	4.75	3.00	577	505	7.10	1.65	8.0	0.094	182.80	527.10	0.0015
19	144.9	4.40	3.20	530	497	7.01	0.64	10.0	0.068	149.93	501.39	0.0001
20	328.2	8.00	3.00	840	833	7.96	2.48	10.0	0.062	96.01	817.03	0.0004
22	302.5	7.88	2.88	795	725	6.66	1.16	14.0	0.100	150.96	766.81	0.0028
23	327.5	7.31	3.00	385	327	8.00	2.95	8.0	0.104	161.26	335.28	0.0005
24	335.2	8.00	3.00	340	243	12.23	7.62	8.0	0.084	159.95	275.88	0.0094
25	398.1	8.25	3.00	519	484	9.17	4.30	8.0	0.097	146.76	475.02	0.0003
26.5	334.6	8.88	3.00	574	494	8.13	1.16	8.0	0.060	92.46	561.22	0.0137
33	327.8	9.50	3.00	687	373	8.84	4.25	8.0	0.200	327.42	539.80	0.0590
35	327.5	7.31	3.00	811	544	7.51	2.66	8.0	0.184	277.01	713.50	0.0437
37	340.5	8.88	3.00	799	698	6.69	0.78	8.0	0.197	274.58	735.20	0.0022
54	317.3	7.70	2.90	470	437	6.51	1.03	8.0	0.086	122.80	447.96	0.0005
55	317.3	7.70	2.90	472	430	7.15	1.73	8.0	0.070	104.00	452.56	0.0023
56	328.0	8.00	3.00	373	352	7.08	1.58	8.0	0.054	79.09	360.24	0.0005
57	343.7	8.88	3.00	399	386	8.06	2.01	8.0	0.060	91.60	382.21	0.0001
58	343.7	8.88	3.00	495	490	7.47	0.73	8.0	0.079	116.10	477.54	0.0006
59	328.2	8.00	3.00	608	590	6.84	1.26	8.0	0.137	196.45	561.92	0.0021

Table F-8. B727 Phase II Test Shot Summaries (Ref. F-3)

Shot #	m (g)	Frag Len avg (x-axis) (in)	Frag Width avg (z-axis) (in)	V _i (ft/s)	V _r (ft/s)	L (in)	A _p (in ²)	θ (deg)	t (in)	FAA V ₅₀ (ft/s) (C _s =276 MPa)	FAA JTCG plg V _r (ft/s)	(Q-1) ²
1	509.0	7.00	4.00	478	441	15.18	12.96	0.0	0.074	126.16	424.73	0.0012
2	419.0	7.00	4.00	503	454	14.11	11.77	0.0	0.079	142.21	438.58	0.0009
3.1	817.1	8.00	8.00	416	397	17.64	6.52	0.0	0.078	112.41	389.60	0.0003
4	884.5	8.00	8.00	560	407	25.48	36.16	0.0	0.105	175.93	444.93	0.0046
5	884.5	8.00	8.00	480	440	23.01	27.83	0.0	0.073	115.44	422.20	0.0014
8	1400.6	7.00	2.00	519	435	19.25	14.53	0.0	0.083	95.48	491.08	0.0117
9	1440.0	7.00	2.00	493	405	20.75	13.97	0.0	0.082	97.17	466.50	0.0156
10	1376.1	7.00	2.00	490	356		6.40	0.0	0.184	0.00	471.69	0.0557
11.1	873.7	8.00	8.00	445	267	22.76	25.34	0.0	0.115	183.23	352.20	0.0367
12	708.6	8.00	8.00	662	647	18.07	3.97	0.0	0.075	118.23	639.19	0.0001
14	871.8	8.00	8.00	869	725	29.59	54.33	0.0	0.045	81.84	767.44	0.0024
15	821.8	8.00	8.00	872	841	22.62	3.02	0.0	0.045	73.70	862.45	0.0006
16	868.3	8.00	8.00	986	864	29.86	55.36	0.0	0.048	86.96	863.45	0.0000
23	596.6	7.00	5.50	780	775	14.58	7.80	0.0	0.097	149.70	723.85	0.0043
24	585.2	8.00	8.00	721	571	29.42	53.59	14.9	0.080	183.24	518.76	0.0053
25	851.5	8.00	8.00	692	450	27.31	43.56	14.9	0.080	146.36	567.27	0.0287
26	802.6	8.00	8.00	741	649	21.19	5.76	14.6	0.075	124.32	712.53	0.0074
27	480.1	7.00	5.00	879	681	17.56	11.63	14.9	0.119	232.49	746.60	0.0056
28	483.6	7.00	5.00	873	526	21.15	27.32	14.9	0.145	308.70	590.21	0.0054
37	823.4	8.00	8.00	894	814	24.52	33.53	14.9	0.065	114.58	788.58	0.0008
38	802.1	8.00	8.00	666	592	21.59	22.07	14.9	0.065	108.94	606.12	0.0004
39	802.1	8.00	8.00	615	459	25.64	34.02	14.9	0.065	118.72	534.24	0.0150
40	800.2	8.00	8.00	845	741	22.57	2.54	14.9	0.083	142.40	822.77	0.0094
41	777.2	8.00	8.00	736	621	23.97	31.72	14.9	0.085	152.49	619.13	0.0000
42	777.2	8.00	8.00	663	525	18.32	4.61	14.9	0.083	130.18	635.43	0.0277

Table F-9. Aluminum Plate Test Shot Summaries (Ref. F-1)

Shot #	m (g)	Frag Len avg (x-axis) (in)	Frag Width avg (z-axis) (in)	V _i (ft/s)	V _r (ft/s)	L (in)	A _p (in ²)	θ (deg)	t (in)	FAA V ₅₀ (ft/s) (C _s =276 MPa)	FAA JTCG plg V _r ⁻ (ft/s)	(Q-1) ²
3	772.0	7.63	7.56	553	0	30.37	57.65	0.0	0.250	489.52	139.46	
4	772.0	7.63	7.56	541	0	29.96	56.09	0.0	0.250	486.22	130.63	
5	772.0	7.63	7.56	368	0	26.32	42.34	0.0	0.250	455.72	≈0*	
7	770.0	7.63	7.56	402	0	29.34	53.73	0.0	0.250	481.72	≈0*	
8	770.0	7.63	7.56	499	0	16.13	3.78	0.0	0.375	535.74	≈0*	
9	748.0	7.13	7.94	588	0	56.55	30.13	0.0	0.375	1017.89	≈0*	
10	748.0	7.13	7.94	608	0	28.42	49.81	0.0	0.375	721.66	≈0*	
23	715.0	7.63	7.75	603	229	17.02	5.69	0.0	0.375	571.17	171.34	0.0090
24	748.0	7.13	7.94	554	0	21.61	21.95	0.0	0.375	629.26	≈0*	
64	741.0	7.75	7.38	445	0	26.68	43.31	0.0	0.250	468.33	≈0*	
65	741.0	7.75	7.38	469	0	24.92	37.19	0.0	0.250	452.59	78.33	
66	650.0	7.75	7.38	563	207	23.36	31.62	0.0	0.250	467.87	201.90	0.0001
67	679.1	7.75	7.38	536	339	16.79	3.66	0.0	0.250	388.06	348.47	0.0003
68	679.1	7.75	7.38	511	283	18.69	12.29	0.0	0.250	409.48	253.87	0.0033

Index

- \$, 65, 78, 134
- \$COMMENT, 44, 64
- \$NAME, 44
- .AND., 60, 63
- .NOT., 60, 63
- .OR., 60, 63
- AC 20-128A, 3, 6, 22, 29, 48, 106, 112, 121, 125, 147
- AFT, 84, 87, 88, 91, 135
- All, 79, 80
- analysis objective, 4, 29
- Analysis_Tool.jar, 107, 122, 123, 142, 145
- Analysis_Tool_Input.txt, 122, 123, 145
- arc, 81, 83, 85
- AUTOFAIL, 94, 95
- auto-fail component, 10, 23, 45, 67, 94, 95
- AUTOFAIL FILE, 23, 24, 65, 67, 94
- Aviation Rulemaking Advisory Committee (ARAC), 2, 18
- azimuth, 74, 75, 76
- bin directory, 9, 43, 125
- binary sequence, 59
- business jet, generic/representative, 1, 21, 125, 131, 134, 135, 147
- CBAR, 37, 42
- CBEAM, 42
- CBULK, 11, 12, 37, 39, 119, 153
- CCONE(1/2/3), 36, 37, 42, 44
- CCW, 75
- CELBOW, 36
- centroid, 6, 74, 75, 76, 81, 106
- centroid radius (R_c), 81, 82, 83, 85
- certification, 1, 48
- CHEX(1/2), 37, 42, 44
- CHEXA1, 42
- CLINE, 36, 42, 44, 140, 141
- CLUSTER, 69, 70
- COMALIAS, 47, 55, 57, 64, 103
- combined risk factor. *See* risk factor: combined
- Combined_Debris.csv, 123, 145
- Combined_Engine.csv, 124, 145
- Combined_Flight_Phase.csv, 124, 145
- Combined_Rotor.csv, 124, 145
- component
 - identification name, 45, 49, 55, 64, 95, 112
 - identification number, 5, 36, 54, 55, 94, 97, 112
 - intercept angle, 6, 11, 72, 112, 127, 128, 130, 136
 - property, 45
- COMPPHY, 46, 55, 57, 64
- COMPPK, 47, 55, 64, 103
- control file, 9, 10, 11, 12, 26, 35, 60, 65, 99, 102, 103, 104, 105, 106, 107, 108, 109, 125
- control.ucf, 102, 125
- conversion, 2, 39, 42, 43, 44
- COVART EXEC, 66
- CQUAD, 36, 42
- CQUAD(1/2/4), 42
- critical component, 43
 - as input, 10, 21, 33, 35, 47, 48
 - as output, 5, 6, 13, 72, 112, 115, 116
 - criticality analysis, 4, 31, 33
- criticality analysis, 4, 30
- CROD, 42
- CSPHERE, 37, 42
- CTRI, 36, 39, 42
- CTRIA(1/2/4), 42
- CW, 75
- damage modes effects analysis (DMEA), 4, 31, 35
- debris
 - category, 4, 5, 99, 104, 105, 121, 122, 123, 124, 153
 - characterization, 5, 10, 11, 20, 66, 78, 79, 82, 83, 91, 92, 93, 96, 125, 134, 135
 - mass distribution, 20, 79, 83, 91
 - model, generic, 92
 - multi-segment, 81, 83, 149, 150
 - type, 5, 18, 68, 79, 80, 81, 82, 83, 84, 89, 91, 105, 106, 153
 - zone. *See* hazard:zone
- DEBRIS, 74, 79, 80, 119
- DEBRIS FILE, 65, 66, 74, 78, 122
- debris mass distribution, 19
- debug flag, 73, 74, 102, 104, 107
- decompression analysis, 2, 27, 97
- degenerate triangle, 43
- density
 - component, 45, 46, 47, 64, 68
 - debris, 85
 - material type, 96
 - target, 27
- dependent fault tree, 52, 53
- design, 1, 3, 27
- DETAIL FILE, 13, 71, 111
- distribution
 - deviation, 83, 87, 88, 91
 - direction, 83, 87, 88, 91
 - mean, 83, 87, 88, 91
 - normal, 22, 84, 86, 87, 88, 90, 134, 135
 - phase failure (D_p), 29, 30, 106, 122, 123
 - skewed, 22, 84, 86, 87, 88, 91, 134, 135
 - uniform, 22, 84, 86, 87, 88, 91, 134
- elevation, 74, 75, 76
- ENDGROUP, 63
- ENDNAME, 63
- engine

- identification, 73, 74, 75, 76, 77, 94, 95, 103, 106, 122
- section, 73, 74, 75, 76, 77, 79, 103, 106
- type, 73, 74, 79, 103, 106
- ENGINES, 122
- environment variable, 66, 102
- error, 9, 43, 44, 72, 73, 92, 100, 102, 103, 104, 119, 120
- ERROR FILE, 10, 72, 73, 103, 119
- event, 1, 6, 10, 13, 73, 74, 79, 82, 92, 153
- EVENT, 12, 73, 75, 77, 79, 94, 95, 105, 107, 109, 122
- event type, 73, 74, 79, 122
- failure analysis, 2, 60
- failure combinations/matrix, 11, 21, 57, 58, 59, 60, 139, 140, 141
- FASTGEN
 - coordinate system, 36, 37
 - group, 37, 39, 40, 42
 - primitive, 36, 37, 39, 42, 44
- fastgen.plb, 42
- fastgen_template.db, 42
- fastgen_utils.plb, 42
- FastRT, 68
- fault tree. *See* hazard:tree
- finite element, 2
- flight phase, 4, 29, 30, 33, 47, 49, 64, 65, 94, 95, 104, 106, 121, 122, 123, 124
- FLIGHT_PHASE_WEIGHTS, 123
- FLIGHT_PHASES, 123
- FORE, 84, 87, 88, 91
- fragment
 - large, 1, 5, 18, 19, 67, 80, 82, 91, 125, 128, 153
 - orientation, 12, 20, 88, 90
 - small, 1, 5, 18, 20, 80, 125, 153
- gamma
 - distribution, 86
 - penetration equation variable, 25, 26, 71, 107, 148, 149
- generated_mv.txt, 63
- geometric model, 2, 4, 10, 11, 12, 20, 21, 35, 36, 39, 43, 44, 67, 68, 81, 106
- geometry, 3, 4, 5, 35
- geometry reduction, 11, 21, 136
- grain, 48
- grid
 - point, 36, 37, 39, 40, 41, 42, 44, 45
 - size, 68, 69, 81, 82, 83
- GRPMULT, 49, 55, 63
- hazard
 - functional, 3, 4, 29
 - level, 5, 6, 11, 24, 33, 47, 65, 94, 95, 123, 153
 - tree, 3, 4, 11, 23, 24, 31, 33, 48, 49, 50, 51, 52, 53, 64, 67, 111
 - zone, 21, 35, 104, 153
- HAZARD, 65
- HazSummary, 71, 109
- HEAD file, 10, 153
- HITSUM FILE, 13, 72, 107, 110
- hole size, 2, 11, 13, 27, 67, 72, 73, 97, 114, 115, 117, 118, 138
- HOLEGRP, 97, 138
- HOLEGRP FILE, 65, 67, 97, 114, 138
- HOLESIZE FILE, 13, 28, 72, 113, 138
- hypersampling, 12, 67, 107, 141, 153
- impactor weight table, 47, 48
- independent fault tree, 52, 53
- infinite energy disk, 125, 147
- INTERCEPT FILE, 71, 72, 104, 112, 136
- interference, 43, 44
- iteration
 - random, 5, 10, 11, 12, 19, 21, 22, 23, 71, 73, 74, 110, 111, 134
 - significant, 13, 72, 73, 114, 131
- JTYPE FILE, 11, 23, 45, 47, 49, 51, 54, 55, 56, 57, 64, 66, 95, 96, 103, 112, 147, 154
- L-FRAGMENT, 79, 80, 81, 88, 90, 91, 92, 96, 128
- LIMITS, 74, 76, 109, 123
- line-of-sight (LOS), 5, 119, 154
- LOCATION, 73, 74, 75, 109, 123
- LOGIC FILE, 2, 66, 139
- logic statement, 9, 60, 66, 139
- LOGICST, 63
- mac.exe, 54
- mass weighting factor. *See* debris mass distribution
- MASTER file, 10, 12, 154
- MATERIAL FILE, 65, 67, 84, 89, 137
- material type
 - component, 11, 25, 41, 43, 45, 46, 47, 64, 67, 96, 103
 - debris, 11, 30, 46, 67, 83, 84, 88, 89, 96
- MATPROP, 96
- mean. *See also* distribution:mean
 - aircraft risk (C_A), 122, 123
 - engine risk (C_E), 122, 123
- mkwgtf.exe, 9, 60, 103
- mode
 - plate, 36, 44
 - volume, 36, 37, 44
- Monte Carlo. *See* iteration:random
- multiple adjacent component (MAC) tool/utility, 54
- multiple processors, 69, 134
- multiply vulnerable (MV), 31, 45, 48, 154
- MV FILE, 45, 48, 64, 66, 103, 139
- MVKILL, 49, 61
- NASTRAN, 36, 39, 42
- NAWCWD, 1, 27, 92, 122
- NEARFIELD, 93, 94

near-field component, 10, 20, 93, 94
 NEARFIELD FILE, 20, 65, 67, 93
 NORMAL, 84, 86, 87, 88, 90, 91
 O_COVART, 102, 103
 obliquity angle, 5, 25, 26, 27, 71, 148, 149, 154
 operating system, 100
 orientation

- engine axis, 74, 75
- fragment. *See* fragment orientation

 overall stage risk. *See* risk: rotor stage
 p3epilog.pcl, 42
 Patran, 2, 39, 42, 43
 Patran_Preference, 42, 43
 Patran-to-FASTGEN_Converter.pdf, 42, 43
 penetration, 5, 10, 13, 19, 25, 28, 31, 46, 47, 67, 91, 92, 117, 118, 147, 148
 PENETRATION, 71, 107, 125, 148
 penetration equations, 24, 71, 92, 96, 137, 138, 147, 148
 PH_Sum.txt, 123, 145, 146
 phase failure distribution. *See* distribution: phase failure
 PK FILE, 45, 47, 66, 103, 154
 plate mode. *See* mode: plate
 post-processing, 104, 105, 106, 107, 108, 121, 125, 142
 PowerPlant Installation Harmonization Working Group (PPIHWG), 18
 presented area, 19, 20, 27
 presented area perimeter, 24, 25
 probability of

- aircraft hazard given component damage ($P_{\text{haz/cd}}$), 47
- component damage given a hit ($P_{\text{cd/h}}$), 47
- hazard (P_{HAZ}), 6, 13, 30, 58, 59, 60, 61, 63, 72, 73, 109, 110, 115, 154
- hazard given a hit ($P_{\text{haz/h}}$), 47, 48, 52, 65
- hazard, component (P_{haz}), 6, 12, 13, 22, 24, 110, 111, 123, 124, 154
- hit ratio (P_{h}), 6, 112, 154

 production runs, 67, 73, 100, 104, 107, 121, 123
 project Thor, 26
 random number, 12, 19, 20, 22, 67, 69, 73, 74, 85, 86, 87, 90, 91
 redundancy, 4, 5, 10, 31, 33, 47, 48, 54, 64, 94, 111, 154
 regulation, 3, 18, 27
 RESOURCE ESTIMATOR, 70, 71
 risk

- analysis (assessment), 3, 60
- angle, 5, 104, 112, *See also* spread:risk angle (SRA) *or* translational:risk angle (TRA)
- factor, 3, 121
- factor, combined (C_{sp}), 121, 123
- rotor stage (C_{s}), 122, 123

 rotation direction, 75
 rotational release, 22, 76, 77
 rotational release point variance, 12, 19, 20, 21, 22, 67, 83, 85, 88, 90, 153
 rotor

- burst, 2, 9, 10, 12, 18, 19, 27, 106, 153
- stage, 3, 18, 73, 74, 75, 76, 77, 103, 121, 122, 123, 124

 ROTORS, 123
 run setup panel, 105, 106
 run time, 10, 21, 67, 68, 69, 70, 100, 101, 102, 136
 Run_Setup.xls, 105, 142
 safety analysis/assessment, 2, 3
 SECTION, 39, 43, 44
 SEGMENT, 79, 80, 81, 83, 84, 91, 92, 96, 129
 segment, disk, 1, 5, 6, 18, 67, 80, 82, 91, 125, 129, 131, 153
 S-FRAGMENT, 79, 80, 81, 88, 90, 96, 125
 shear constant, 24, 96, 137
 shielding, 4, 10, 21, 35, 43, 93, 137, 138, 147
 shotline, 1, 5, 12, 19, 20, 24, 67, 68, 69, 72, 82, 85, 90, 92, 104, 113, 153, 154, 155
 SIGITCHOLE FILE, 13, 72, 73, 117, 131, 132
 SIGITCRIT FILE, 13, 72, 116, 133
 SIGITSHOLE FILE, 13, 72, 73, 118, 133
 SIGITSYS FILE, 13, 72, 73, 115, 133
 sigma, 88, 90
 singly vulnerable (SV), 31
 SKEWED, 84, 86, 87, 88, 91
 SMCOMP FIX, 67, 69, 141
 spread

- intercept angle, 11
- release angle, fore/aft, 12, 19, 20, 21, 22, 83, 85, 86, 87, 88, 90, 91
- risk angle (SRA), 3, 6, 112, 121, 154, *See also* risk:angle

 stereolithography (STL), 39, 43, 44
 stl2fast4.exe, 41
 subassembly, 72
 SUMMARY FILE, 13, 71, 109, 122
 SYSMULT, 49
 SYSTEM, 110
 TARGET FILE, 35, 66, 94, 97, 136
 TARGET REDUCTION, 67, 107
 thickness

- component, 5, 24, 27, 41, 45, 46, 64, 68, 154
- debris, 30, 83, 85, 88, 89, 90

 Thor. *See* project Thor
 THREADS, 69, 107, 134
 THREAT, 10, 154
 threat window, 8, 121, *See also* spread:risk angle (SRA) *or* translational:risk angle (TRA)
 tolerance, 72, 136
 trajectory, 3, 6, 7, 19, 20, 72, 86, 90, 92, 113, 118
 TRAJECTORY FILE, 72, 113

translational
 intercept angle, 11
 release angle, 19, 20, *See also* rotational
 release
 risk angle (TRA), 3, 6, 112, 121, 154, 155,
 See also risk:angle
tri-hub rotor burst. *See* rotor:burst
twin-engine aircraft, generic, 1, 23, 31, 32, 33,
 34, 35, 45, 49, 51, 55, 94, 95, 97, 100, 111,
 122, 125, 136
UDAIM, 12, 155
UDASPECT, 12, 155
UEDDAM analysis process, 3
UEDDAM Visualizer, 43, 44, 72, 104, 118, 119
UEDDAM_Debris, 106
UEDDAM_Holegrp, 106

UNIFORM, 84, 86, 87, 88, 91
Uvis.bat, 43
V₅₀ equation, 24, 25, 26, 71, 148, 149
variance/offset, 19, 83, 85, 86, *See also*
 rotational release point variance
velocity
 ballistic limit, 25, 26, *See also* V₅₀ equation
 release, 83, 85, 88, 89, 93
 residual (V_r), 24, 26, 27
VISUALIZER FILE, 72, 118
volume mode. *See* mode: volume
V_r equation. *See* velocity:residual (V_r)
warning, 72, 73, 119
warped quad, 43
weighting factor, 57, 60, 64
WGT, 47

B Fault tree development tool (FTDT) version 1.0 user manual



FTDT User Manual

Fault Tree Development Tool (FTDT) Version 1.0 User Manual

Prepared by:

**Jennifer Barker
Devin Smaldore
Ellen Phifer**

September 2020

*Further dissemination only as directed by NAWCWD, Code 418300D, China Lake,
CA 93555, September 2020 or higher DOD authority.*

INTENTIONALLY LEFT BLANK

Table of Contents

1. Introduction	1
1.1. Background	1
1.2. Overview	1
1.3. Compatibility	2
2. External Data.....	3
2.1. Importing MV Files.....	3
2.2. Exporting MV Files	3
2.3. MVKILL/Flight Phase	4
3. Fault Tree Elements	5
3.1. Event Types	5
3.2. Transfer Event.....	5
3.3. Gates	6
4. Manipulating the Canvas	6
4.1. Expand and Contract.....	6
4.2. Zoom In and Zoom Out	7
4.3. Search Feature	8
5. Editing Events.....	9
5.1. Item Dialog.....	9
5.2. Subsystems Limit.....	11
5.3. Copy/Paste Subtrees.....	11
5.4. Deleting Elements.....	12
5.5. Logic Statements and Weighting Factors.....	12
6. References	18

List of Figures

Figure 1. Illustration of the Ability of FTDT to Read In and Write Out COVART MV Files.	1
Figure 2. Example of an MV Input File.	2
Figure 3. File Selection Dialog.	3
Figure 4. Illustration of the Export to MV File Option.	4
Figure 5. Exiting with Unsaved Changes.	4
Figure 6. Event Types.	5
Figure 7. Gate Types and Associated M of N Dialog.	6
Figure 8. Changing Gate Types.	6
Figure 9. Usage of Collapse and Expand Subtrees.	7
Figure 10. Illustration of Canvas with Zoom to Fit Usage.	8
Figure 11. Illustration of the Search Feature.	9
Figure 12. Item Dialogs for a Group, System, and Component Event.	10
Figure 13. Duplicate Name Warnings.	11
Figure 14. Subsystem Limit Error.	11
Figure 15. Copy/Paste Examples.	11
Figure 16. Delete Item Warning.	12
Figure 17. Logic Statement Dialog.	13
Figure 18. Assignment of Weighting Factors in the Logic Statement Builder.	14
Figure 19. Illustration of a Group/System Event with Assigned Logic Statements.	15
Figure 20. Item Dialog with Logic Statements Applied.	15
Figure 21. Removing Logic Statements.	16
Figure 22. Weighting Factors Dialog.	17
Figure 23. Illustration of a Group/System Event with Assigned Weighting Factors.	18

1. INTRODUCTION

1.1. Background

The Federal Aviation Administration’s (FAA) Aircraft Catastrophic Failure Prevention Program is a congressionally funded budget line item that was established to look into improving aircraft safety as a result of the 1989 Sioux City accident. A significant portion of the research performed has been directed to support the Aviation Rulemaking Advisory Committee (ARAC), Power Plant Installation Harmonization Working Group (PPIHWG), including the development of an industry standard for evaluation of the threat to an airplane from an uncontained engine event. In support of this goal, the FAA has collaborated with ARAC and funded Lawrence Livermore National Laboratory, SRI International, and the Naval Air Warfare Center – Weapons Division (NAWC-WD) for extensive research into material properties and analytical capabilities to improve the knowledge base and tool suite for rotor-burst analysis. A prototype tool, the Uncontained Engine Debris Damage Assessment Model (UEDDAM), was developed by the SURVICE Engineering Company for NAWC-WD and is under evaluation with the FAA and industry to determine its suitability to conduct rotor-burst safety assessments. One of the recommendations that resulted from the FAA’s evaluation was to simplify the UEDDAM input file development process through the creation of utilities that would unburden the analyst. One such utility developed is referred to as the Fault Tree Development Tool (FTDT), which both imports and exports appropriately formatted fault trees for use in UEDDAM. Because UEDDAM was built on Joint Aircraft Survivability Program Office (JASPO) funded codes (i.e., Fast Shotline Generator [FASTGEN] and Computation of Vulnerable Area Tool [COVART]), FTDT can also be used to assist vulnerability analysts in development of MV files for COVART assessments; thus, the entire JASP community (users and evaluators alike) is provided with a tool to build and validate vulnerability fault trees.

1.2. Overview

FTDT is a fault tree analysis software specifically designed to read in existing MV files or generate new fault trees within the graphical user interface (GUI), edit the trees, and then export to the COVART MV file format (see Figure 1). Additionally, because it was originally developed for use with UEDDAM, FTDT contains support for logic statement and weighting factor development and visualization.

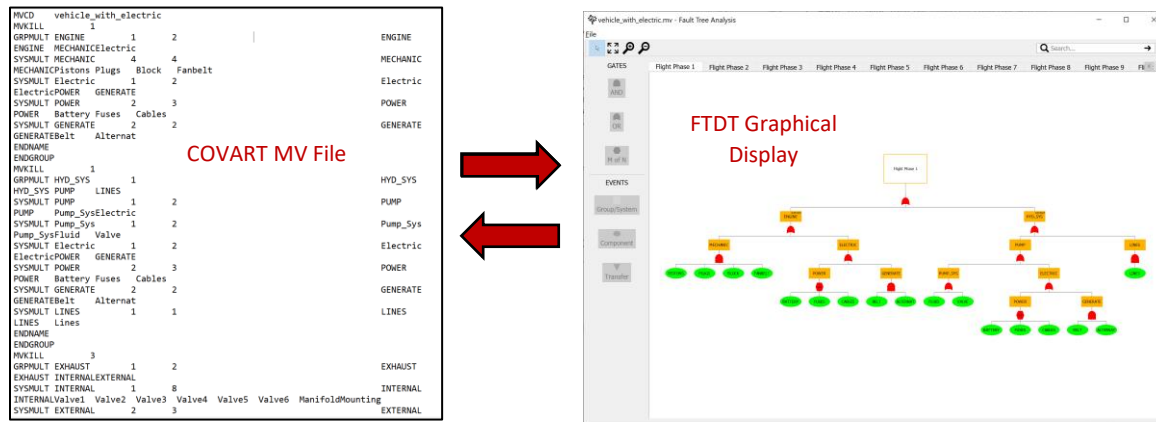


Figure 1. Illustration of the Ability of FTDT to Read In and Write Out COVART MV Files.

It is the intent of this user's guide to provide documentation of the aforementioned features, to include examples of their usage and known limitations. Note that this document is not intended to be a user manual for COVART or fault tree analysis in general; nor is it to be used as a standalone document. Rather, it is the hope of the authors that this supplement will be used in conjunction with the COVART 6.9 User Manual (distributed by the Defense Systems Information Analysis Center [DSIAC]), and/or the UEDDAM 6.0 User Manual (distributed by NAWC-WD for the FAA).

1.3. Compatibility

A standard input for COVART and UEDDAM, the MV file defines how multiply vulnerable (or redundant) components are combined into a fault tree and the supporting logic needed to determine the probability of damage incurred by the system. The input file consists of groups of critical functions, along with subsystems of those groups, defined for an associated flight phase or hazard level (Figure 2). For more information regarding the format and function of MV files, reference section 7 of the COVART User Manual (Ref. 1) and/or section 5.2.3 of the UEDDAM User Manual (Ref. 2).

MVCD	mv				
MVKILL		1			
GRPMULT	ENGINE	1	2		ENGINE
ENGINE	MECHANICELECTRIC				
SYSTEMULT	MECHANIC	4	4		MECHANIC
MECHANIC	PISTONS PLUGS BLOCK FANBELT				
SYSTEMULT	ELECTRIC	1	2		ELECTRIC
ELECTRIC	POWER GENERATE				
SYSTEMULT	POWER	2	3		POWER
POWER	BATTERY FUSES CABLES				
SYSTEMULT	GENERATE	2	2		GENERATE
GENERATE	BELT ALTERNAT				
ENDNAME					
ENDGROUP					
MVKILL		1			
GRPMULT	HYD_SYS	1	2		HYD_SYS
HYD_SYS	PUMP LINES				
SYSTEMULT	PUMP	1	2		PUMP
PUMP	PUMP_SYSELECTRIC				
SYSTEMULT	PUMP_SYS	1	2		PUMP_SYS
PUMP_SYS	FLUID VALVE				
SYSTEMULT	ELECTRIC	1	2		ELECTRIC
ELECTRIC	POWER GENERATE				
SYSTEMULT	POWER	2	3		POWER
POWER	BATTERY FUSES CABLES				
SYSTEMULT	GENERATE	2	2		GENERATE
GENERATE	BELT ALTERNAT				
SYSTEMULT	LINES	1	1		LINES
LINES	LINES				
ENDNAME					
ENDGROUP					
MVKILL		3			
GRPMULT	EXHAUST	1	2		EXHAUST
EXHAUST	INTERNALEXTERNAL				
SYSTEMULT	INTERNAL	1	8		INTERNAL
INTERNAL	VALVE1 VALVE2 VALVE3 VALVE4 VALVE5 VALVE6 MANIFOLDMOUNTING				
SYSTEMULT	EXTERNAL	2	3		EXTERNAL

Figure 2. Example of an MV Input File.

While structurally simple, an MV file can be somewhat difficult to understand just by examining the text, especially for those unfamiliar with its format. Fault tree diagrams, on the other hand, are a more intuitive interface for understanding system failures. Enter FTDT. With the ability to import and export MV files, FTDT affords the user the ability to display, develop, and evaluate MV files graphically using fault tree diagrams.

2. EXTERNAL DATA

2.1. Importing MV Files

A valuable feature of the FTDT is the ability to import existing MV files and visualize them as fault trees. To import an MV file, simply select **File** → **Import From MV File** from the FTDT menu bar (see Figure 3 for the file selection dialog). Files can have either a .mv or .txt file extension. Once an MV file is selected, FTDT converts and graphically displays it as a fault tree, as previously illustrated in Figure 1. The name of the file will appear in the title bar of the GUI. Note, however, that the file will open with all trees collapsed (Figure 9). Additional information about expanding trees is provided in Section 4.1.

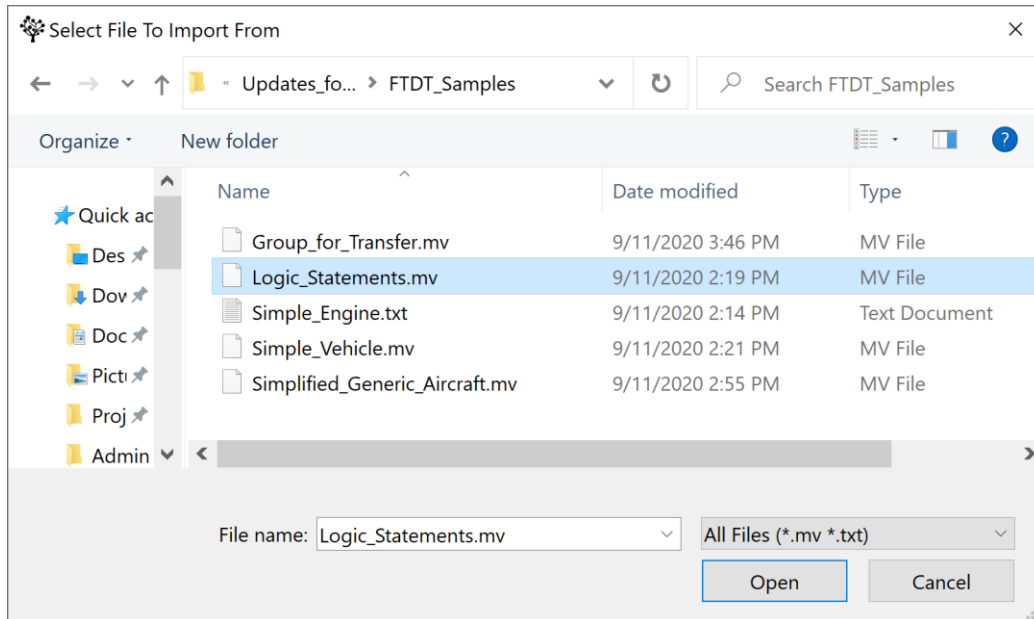


Figure 3. File Selection Dialog.

2.2. Exporting MV Files

As fault trees are in the process of being built, they can be saved to an intermediate file format with the default extension .xml (accomplished by selecting **File** → **Save as XML** from the FTDT menu bar) or .json (accomplished by selecting **File** → **Save as JSON**). Once a fault tree is completed, exporting it to an MV file is accomplished by selecting **File** → **Export to MV file** from the FTDT menu bar (see Figure 4). The file saved will be given the default extension .mv. Note that when working with partially completed fault trees, they should be saved in XML or JSON format even if an extended period of time will occur before continued modification.

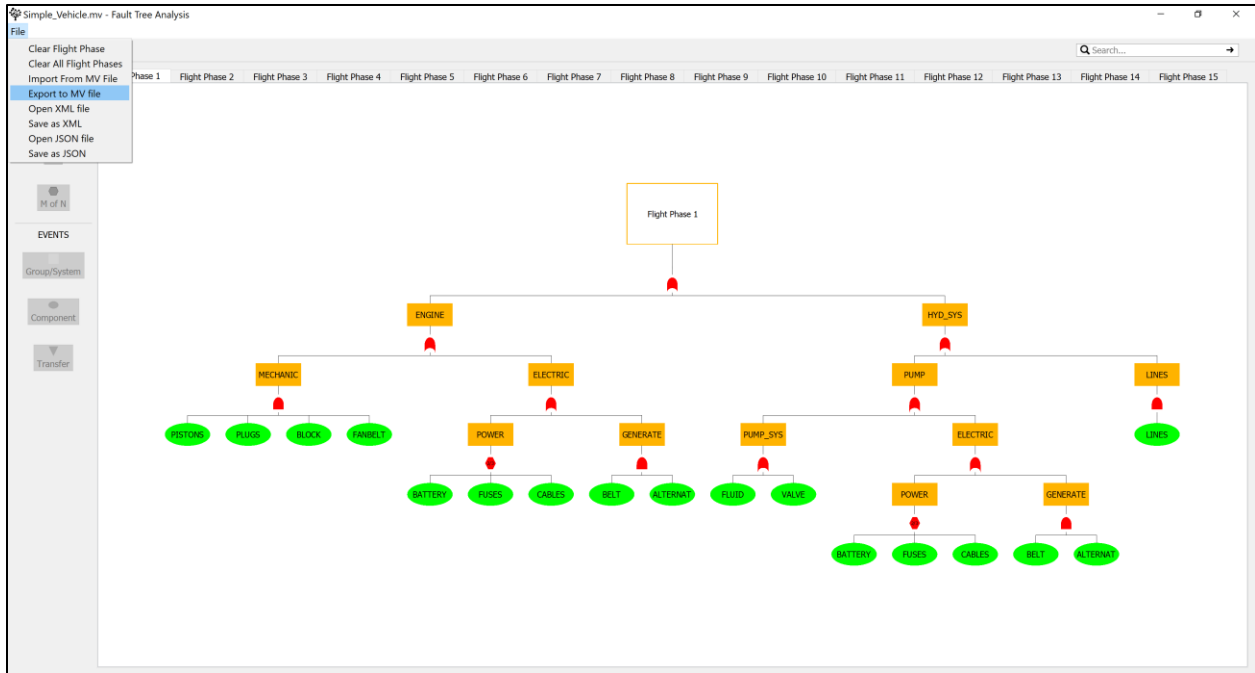


Figure 4. Illustration of the Export to MV File Option.

When changes are made to the file, an asterisk will appear in the title bar next to the file name to indicate that it has changed. The file will need to be saved or exported to preserve the changes. Additionally, when exiting the program with unsaved changes, the user will be prompted to save the changes before exiting (Figure 5).

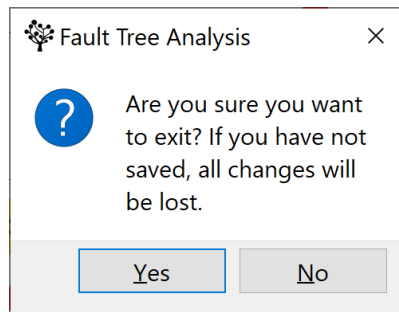


Figure 5. Exiting with Unsaved Changes.

2.3. MVKILL/Flight Phase

In an MV file, the MVKILL function defines the *Hazard Level* or *Flight Phase* for multiply vulnerable groups. Within FTDT, the user can define up to fifteen hazard levels/flight phases (the maximum allowed by COVART) per tree which are applied to second level events. Each of the MVKILL categories may have multiple groups and a component may appear multiple times within that group. However, each instance of a single component/group may only be associated with one hazard level/flight phase.

To make navigating the fault tree more intuitive, tabs have been implemented for each of the 15 possible MVKILLS (identified as *Flight Phases*) and are placed above the main drawing canvas. Clicking on a specific **Flight Phase** tab will change the canvas to show the tree for only those groups that are assigned to that MVKILL level.

3. FAULT TREE ELEMENTS

3.1. Event Types

Event types consist of the *Group/System Event* (depicted in FTDT as a dark yellow box) and the *Component Event* (depicted in FTDT as a green oval) (Figure 6). A Group/System event correlates to groups and systems in the MV file. Top-level Group/System events are directly under the Flight Phase element and gate and are identified as GRUPMULTs in the exported MV file. All lower Group/System events are SYSMULTs in the exported MV file. A Component event correlates to single components or NAMs in the MV file.

All events must be placed beneath a gate (described below in Section 3.3).

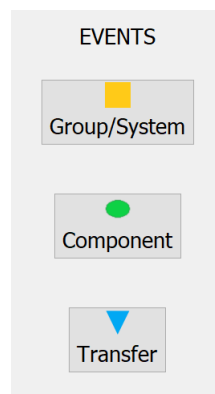


Figure 6. Event Types.

3.2. Transfer Event

The **Transfer** symbol (depicted in FTDT as a blue triangle) allows for modularization of the fault tree through the incorporation of external sub-trees. To incorporate an existing tree into another tree, select the Transfer function as the next element. Once the Transfer element has been applied to the tree, double click the icon to browse to the tree of interest and select it for opening. Files can be in either MV or .xml format. All events (and only events) from the single corresponding Flight Phase tab will be incorporated and displayed in the FTDT canvas once the tree is exported to an MV file. If it is saved as an XML file, the Transfer element will remain and retain the file path to the linked tree, but the transferred elements will not be incorporated into the tree itself.

3.3. Gates

The MV file only supports **AND** and **OR** logic. As a subset of the AND logic, COVART allows the user to specify a certain number **M** of the **N** elements that must be failed for the system to fail – not strictly an AND gate logic. AND gates are represented by a red arch; OR gates are represented by a red irregular arch; and M of N gates are represented by a red hexagon with the M/N value displayed on it. Double click on the M of N gate to edit the value of M.

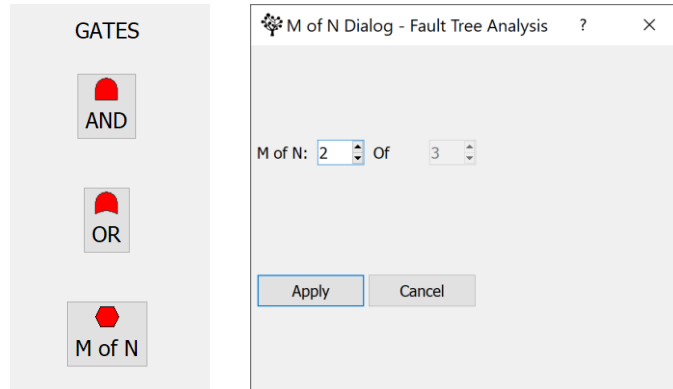


Figure 7. Gate Types and Associated M of N Dialog.

Gates can be added by selecting the desired event (which will then be outlined in blue) and selecting the desired gate. Immediately after adding a gate, events can be added to that gate without re-selecting it. If, however, additional events are to be placed under other gates, the gate must be selected first. Note that gates will also have a highlight to show that they are selected.

Users can change gates from one type to another by right clicking and selecting the desired gate (Figure 8).

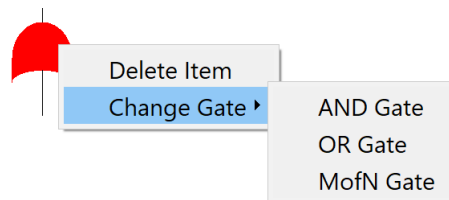


Figure 8. Changing Gate Types.

4. MANIPULATING THE CANVAS

4.1. Expand and Contract

Due to the extensive nature of most vehicle- and system-level fault trees, without the ability to expand and contract different sections of the tree, it can quickly become difficult to navigate on the canvas screen. Thus, for ease of handling, when a gate and component are added to a tree, the ability to collapse a subtree is added. Right clicking on a Group/System event opens a menu displaying **Collapse**

Subtree. When a tree is collapsed, a thick black border is drawn around it (see Figure 9). Right clicking on a collapsed subtree gives the option to **Expand a Subtree**. Note that it is not possible to **Paste a Subtree** or add to it when it is collapsed, but the user can still **Copy** or **Delete** it.

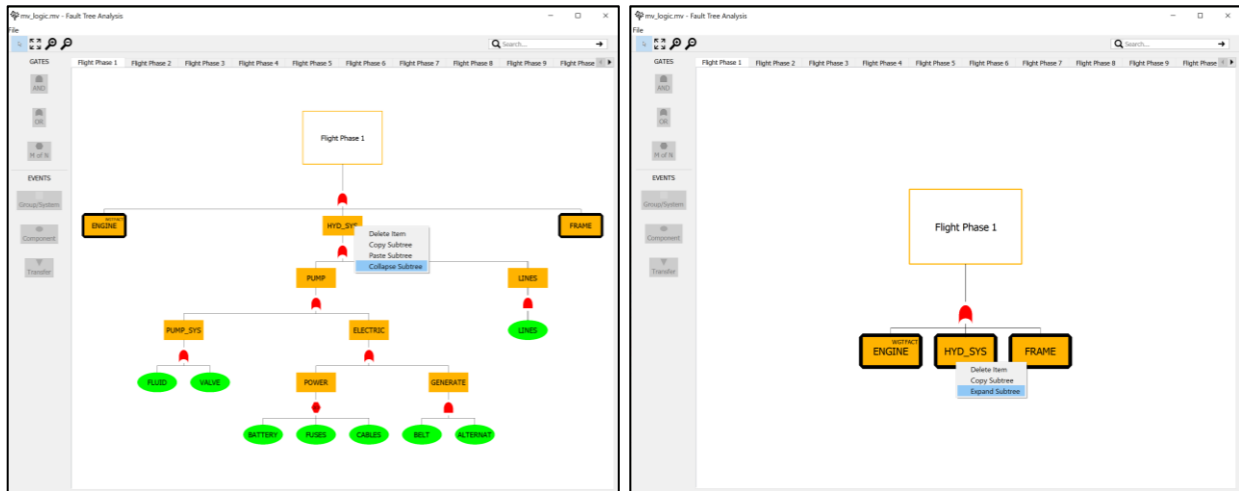



Figure 9. Usage of Collapse and Expand Subtrees.

The entire tree can be collapsed to the top-level groups and expanded by right clicking on the Flight Phase element and selecting either **Collapse All Subtrees** or **Expand All Subtrees**, respectively. Note that when MV files are imported, they will be initially displayed as collapsed at the top level.

4.2. Zoom In and Zoom Out

The canvas automatically adjusts to display all items when the file is opened and when components are added/deleted. To zoom in and out, use the scroll wheel on the mouse. If the tree in the view has become too large to fit on the screen, scroll bars will appear on the right and below the canvas to adjust the position.

The buttons under the menu bar are another method for zooming. The magnifying glasses with “+” and “-” will zoom in and out respectively. The **Zoom to Fit** button () will zoom in or out as needed to display the entire tree on the canvas (Figure 10).

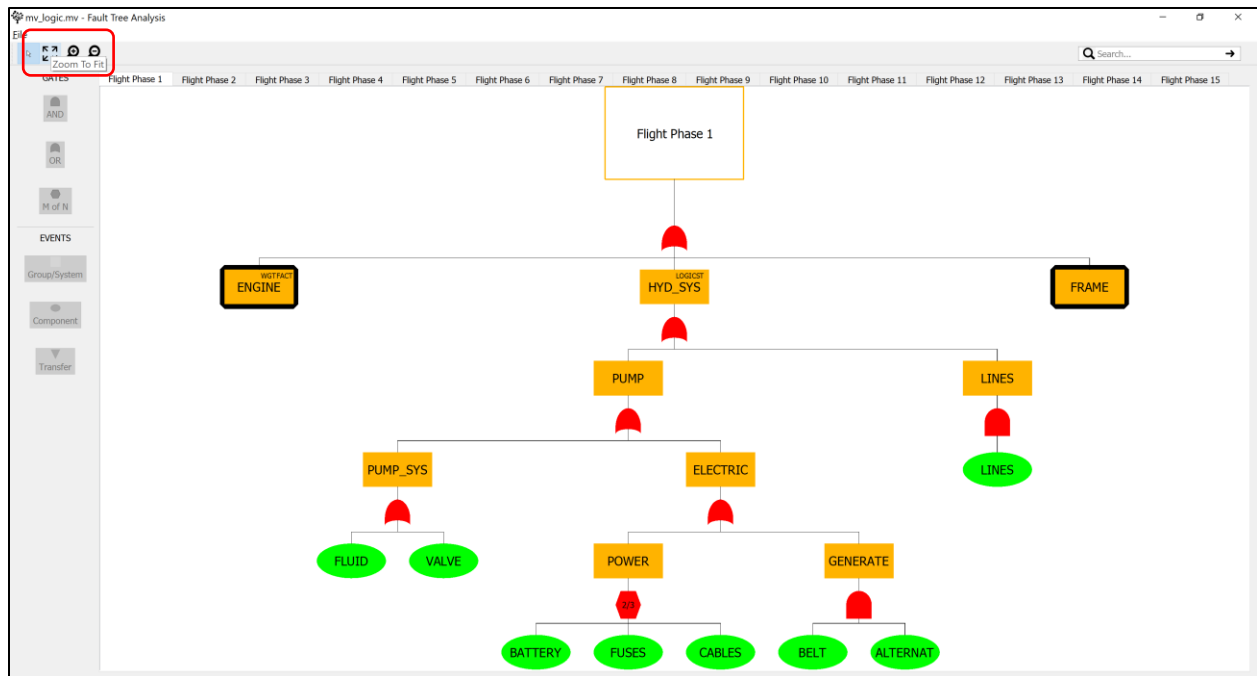


Figure 10. Illustration of Canvas with Zoom to Fit Usage.

4.3. Search Feature

Beneath the menu bar and to the far right of the canvas is a search box. The user can type in a search string and if it is found on the current Flight Phase tab, the canvas will be centered over and zoomed in on the element(s) and a yellow highlight will appear around the matching element(s) (Figure 11). Note that the search feature is best used on a fully expanded fault tree. If no matching element(s) is(are) found, a dialog box will appear indicating “No matches were found.”

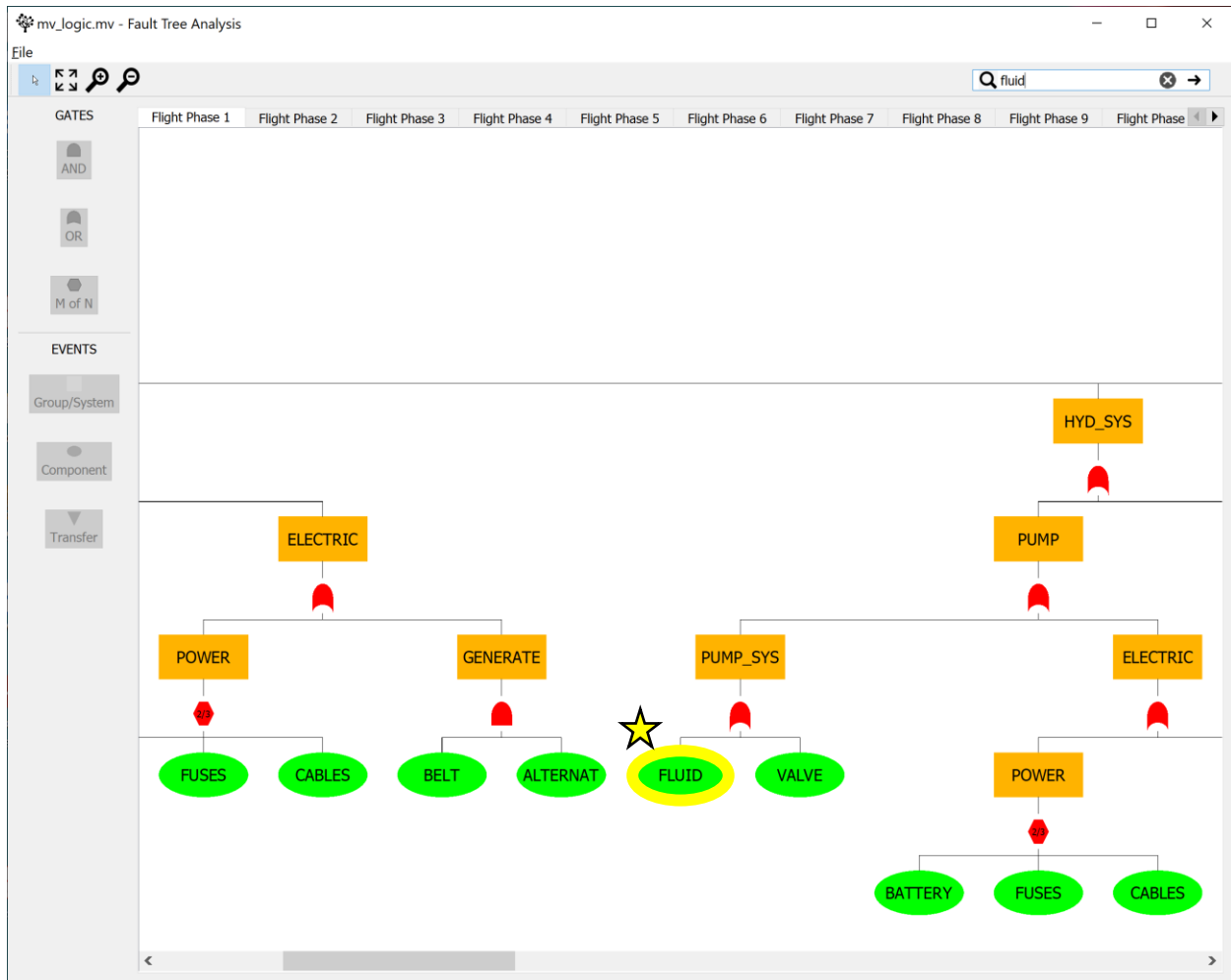


Figure 11. Illustration of the Search Feature.

5. EDITING EVENTS

5.1. Item Dialog

Group/System and Component events in FTDT can be given an identification label (or key). The identification label of Group/System and Component events is used in the COVART MV file as the GRPMULT/SYSMULT and NAM names, respectively. Due to COVART convention of only accepting up to eight characters per system name, FTDT limits identification labels to 8 alphanumeric characters to maintain COVART (and therefore UEDDAM) compatibility. Additionally, Flight Phase, Group/System, and Component events can be given descriptive information in the form of comments. Text in the descriptive field of Flight Phase and Group/System events can be used both for reference in FTDT as well as when exported to MV format and read in on subsequent imports. The descriptive text associated with Component events is only used within FTDT. All descriptive text associated with Group/System events is exported with the preface \$COMMENT whereas that associated with Flight Phase is placed after the MVKILL # entry.

To change the ID label for a Component or Group/System event, the user must access the associated dialog, achieved by double-clicking on the symbol of interest on the FTDT canvas (Figure 12). When creating names for components and systems it should be kept in mind that the names should be not only descriptive but also allow for differentiation from other similar components and systems.

Note that names cannot include the space (), period (.), forward slash (/), plus (+) or minus (-) symbols. Spaces will be automatically replaced by an underscore (_) and the other prohibited characters will not be able to be entered.

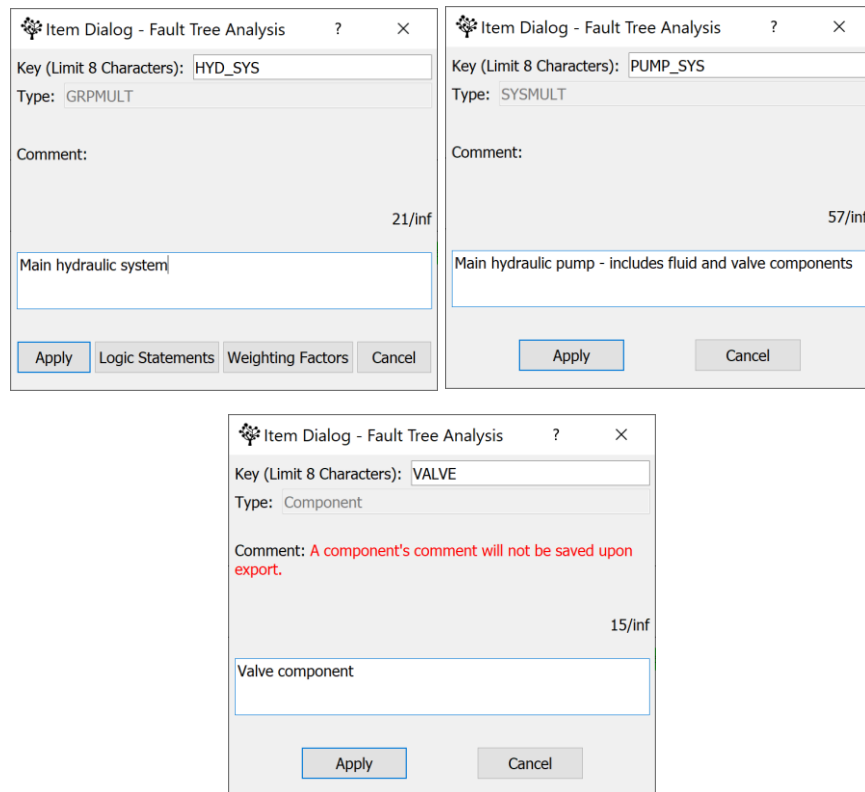


Figure 12. Item Dialogs for a Group, System, and Component Event.

Also note that each event is not required to have a unique name and an event can occur multiple times in a fault tree. However, two dissimilar types of events (i.e., Group/System vs. Component) may not have the same name. If a user attempts to use the same name with similar events, a warning will appear but still allow the user to use the duplicate name. If the name is used for dissimilar events, the user will get a warning and will not be able to use the duplicate name (Figure 13).

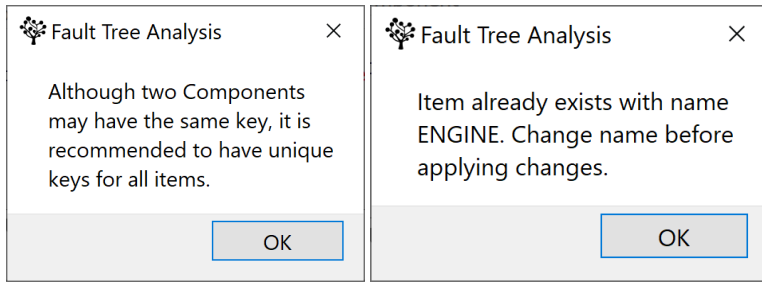


Figure 13. Duplicate Name Warnings.

5.2. Subsystems Limit

Users operating COVART version 6 are restricted to 24 sub-elements for each group or system in an MV file. Thus, once 24 sub-components have been reached, FTDT will restrict any additional events from being added with an Error Dialog stating compatibility limitations (Figure 14).

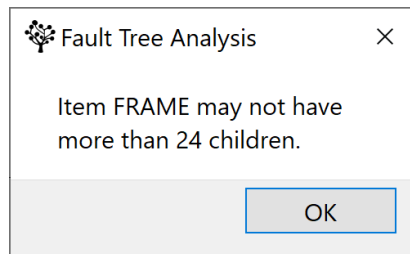


Figure 14. Subsystem Limit Error.

5.3. Copy/Paste Subtrees

There may be times where it is more convenient to copy and paste a portion of a tree than build it from scratch. To copy a subtree, right click on the topmost element in the subtree and select **Copy Subtree**. To paste it, simply right click on the event (not the gate) that it will be placed under and select **Paste Subtree** (Figure 15).

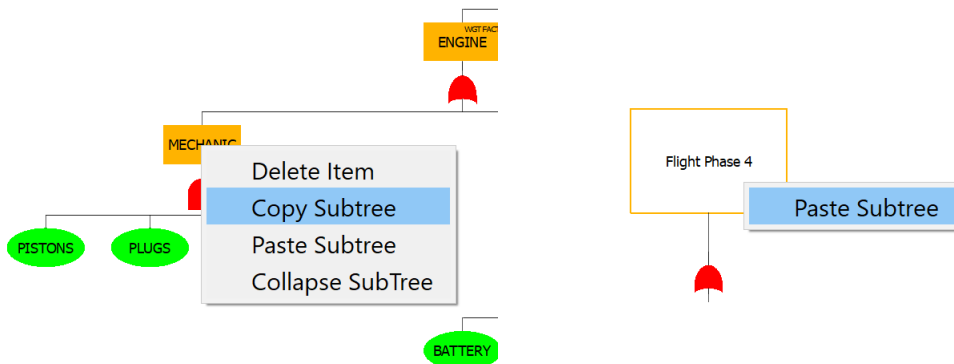


Figure 15. Copy/Paste Examples.

5.4. Deleting Elements

Items can be deleted from the canvas by right clicking the element and selecting **Delete Item**. Note that deleting an item will also delete any subordinate elements, should the item have any. Thus, when deleting any item, a warning will pop up (Figure 16), reminding users that deleting an item will also delete any children it has. There is no undo option, so use caution when deleting.

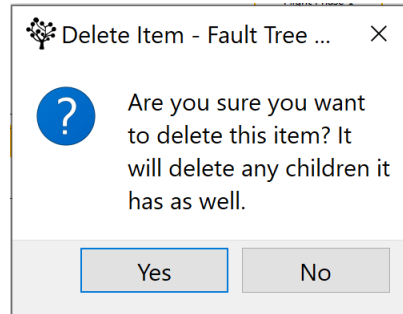


Figure 16. Delete Item Warning.

Another way to delete elements from the canvas is to select either the **File → Clear Flight Phase** or the **File → Clear All Flight Phases** option from the FTDT menu bar. As the menu text would indicate, the first option will clear all elements from the canvas of the currently selected Flight Phase tab whereas the second option will clear all elements from all tabs (thereby deleting the entire tree). A similar warning to that illustrated in Figure 16 appears upon selection of either of these options to ensure the user is aware of the specific ramifications of their selection.

5.5. Logic Statements and Weighting Factors

The MV input file allows for the application of weighting factors to sets of system failures within a multiply vulnerable group. These factors differentiate probabilities of outcomes for different combinations of failure. In FTDT, the individual combinations of critical systems can be entered in one of two ways (Logic Statements or Weighting Factors) and are assigned through the associated dialog. Since weighting factors are applied at the GRPMULT level in a COVART MV file, Group/System Events on the second level (i.e., directly beneath the MVKILL/Flight Phase event) are the only events where the user can access either dialog.

To open the logic statement dialog, open a Group/System event item dialog (see Figure 12) and select the **Logic Statements** button. The logic statement dialog (see Figure 17) is divided into three sections: Statements, Elements, and Statement Builder.

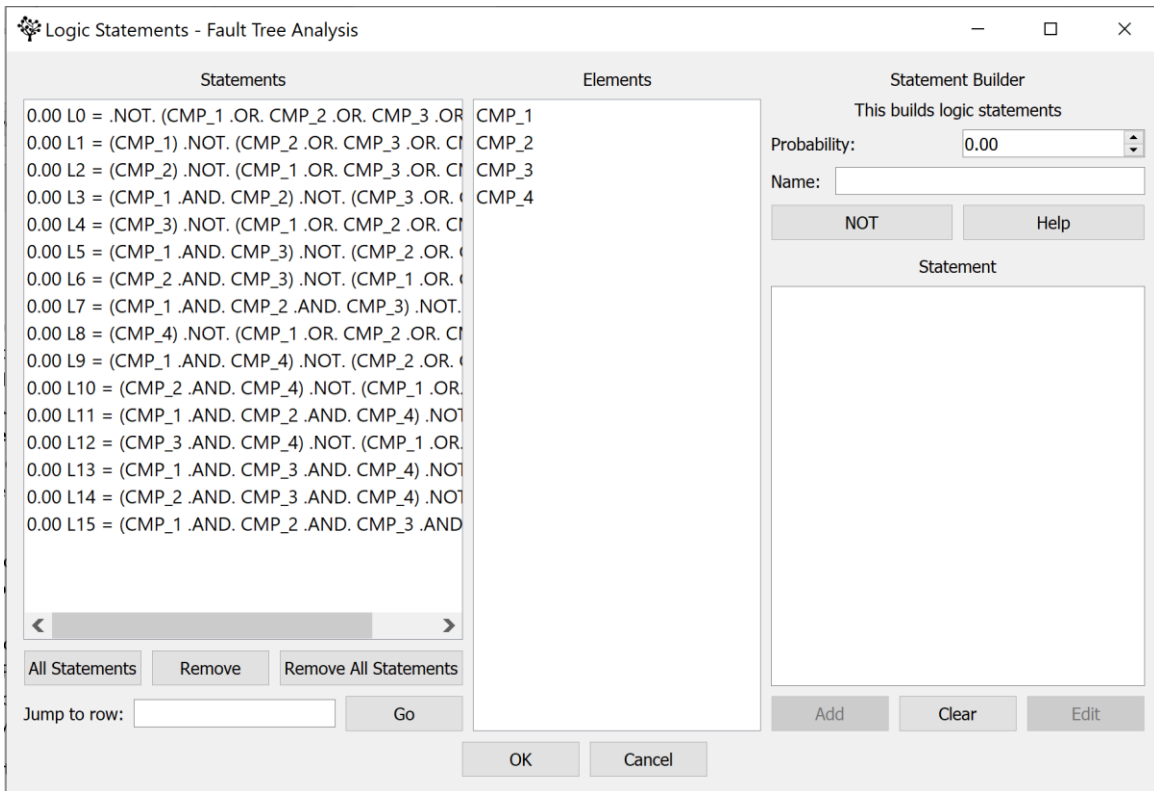


Figure 17. Logic Statement Dialog.

The Statements panel contains a list of all possible statements created using the elements of that event, as listed in the Elements panel. By default, FTDT generates all possible logic statements when the logic dialog is first opened. Individual statements can be removed from the list by selecting the **Remove** button. Similarly, all logic statements can be removed by selecting the **Remove All Statements** button. Once removed, the full list of possible statements can be regenerated by selecting the **All Statements** button. Useful for extensive fault trees, the **Jump to row** tool allows the user to search for a specific statement.

To develop a new logic statement, it is recommended that the user first select the **Remove All** button to clear the automatically generated statements. Begin by selecting an element in the Elements panel and double click to send it to the Statement Builder panel. Selecting more elements from the panel will create an AND statement (i.e. CMP_1 .AND. CMP_2). To create an OR statement select the **NOT** button, located above the statement box. All following elements will be added as OR statements (i.e. (CMP_1 .AND. CMP_2) .NOT. (CMP_3 .OR. CMP_4.)). Alternatively, the user can simply type in the Statement panel to change the syntax. Once complete, select **Add** to finalize the statement. If no name is entered, the Statement Builder will default to L1 (for the first statement). It is important to remember that user-built statements should only be used to create or modify UEDDAM MV files.

To change a probability associated with a particular logic statement, select the statement in the Statements panel and double click to send it to the Statement Builder panel. The desired factor is then

entered into the Probability field. Press the **Edit** button to apply the probability to the statement (see Figure 18).

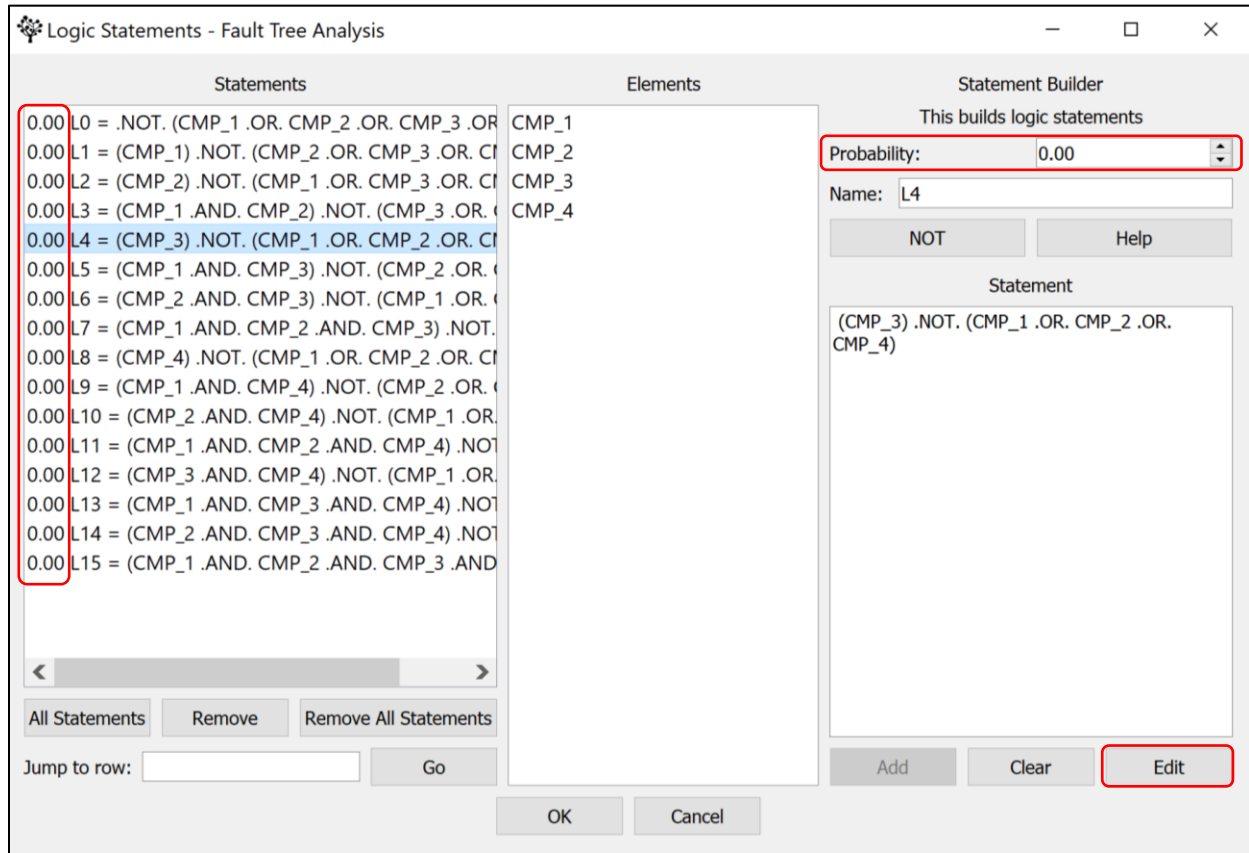


Figure 18. Assignment of Weighting Factors in the Logic Statement Builder.

To make those groups with associated logic statements easily identifiable when viewing the fault tree diagram within FTDT, the label “LOGICST” is added to the Group/System event element (see Figure 19).

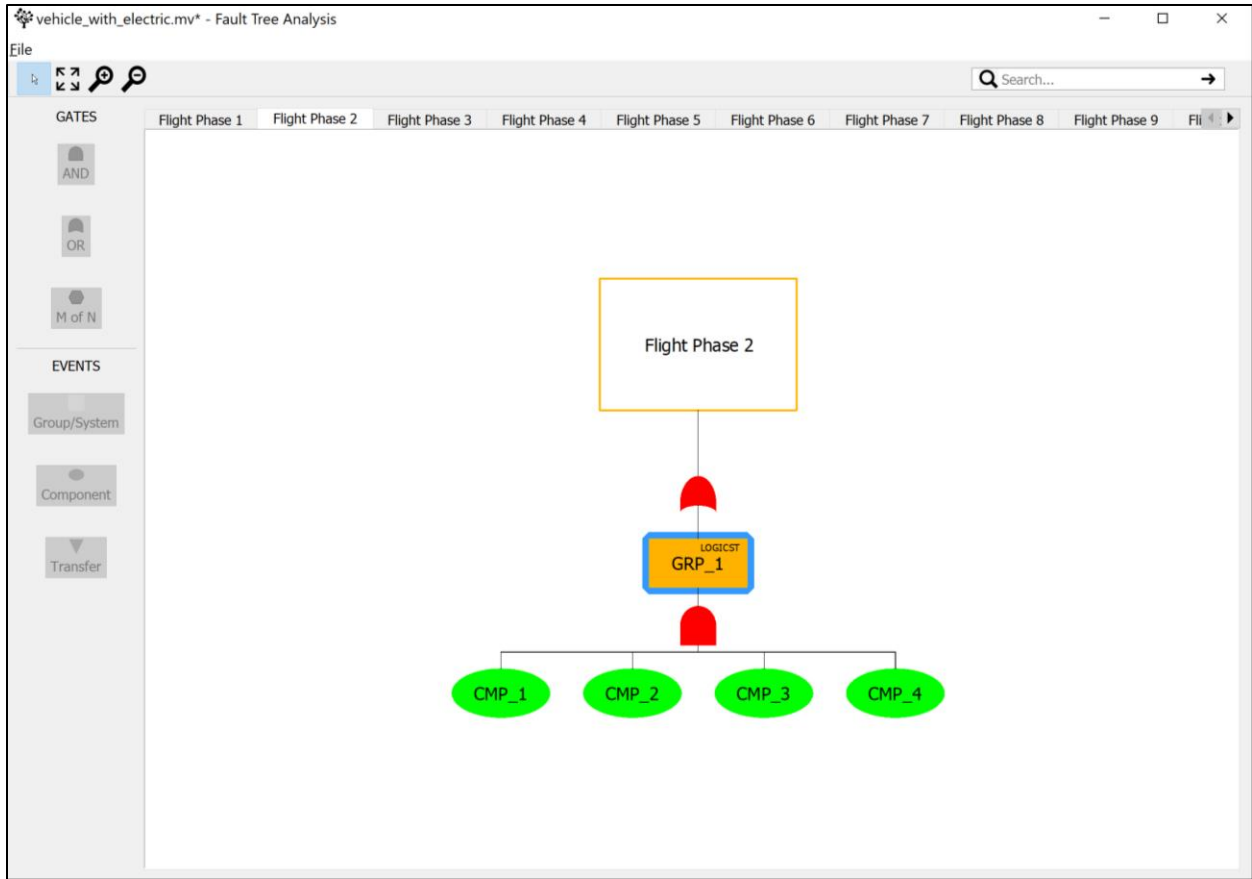


Figure 19. Illustration of a Group/System Event with Assigned Logic Statements.

Once logic statements are applied, the user is restricted from also applying weighting factors until all logic statements have been deleted (Figure 20).

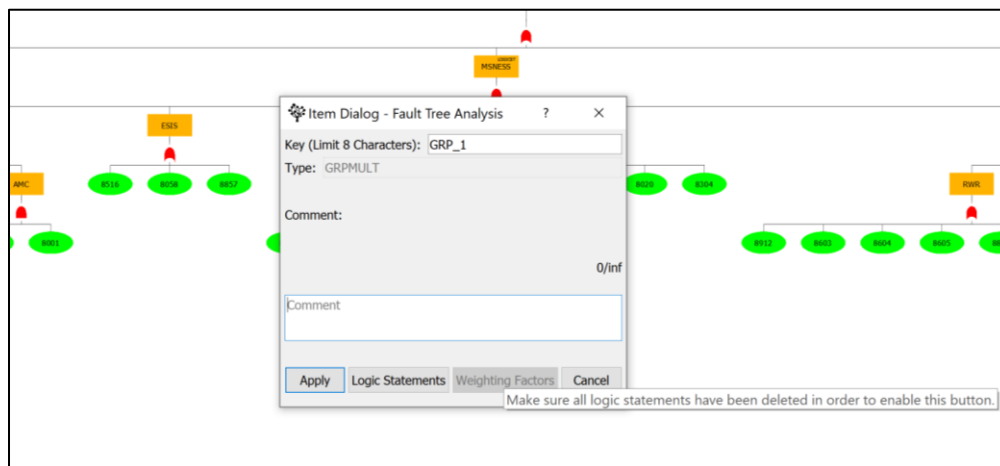


Figure 20. Item Dialog with Logic Statements Applied.

To remove logic statements, open the Logic Statements Dialog and select **Remove All Statements** and click **OK** (see Figure 21). This will clear the items in the Statements panel and allow for edits to the weighting factors in the item dialog.

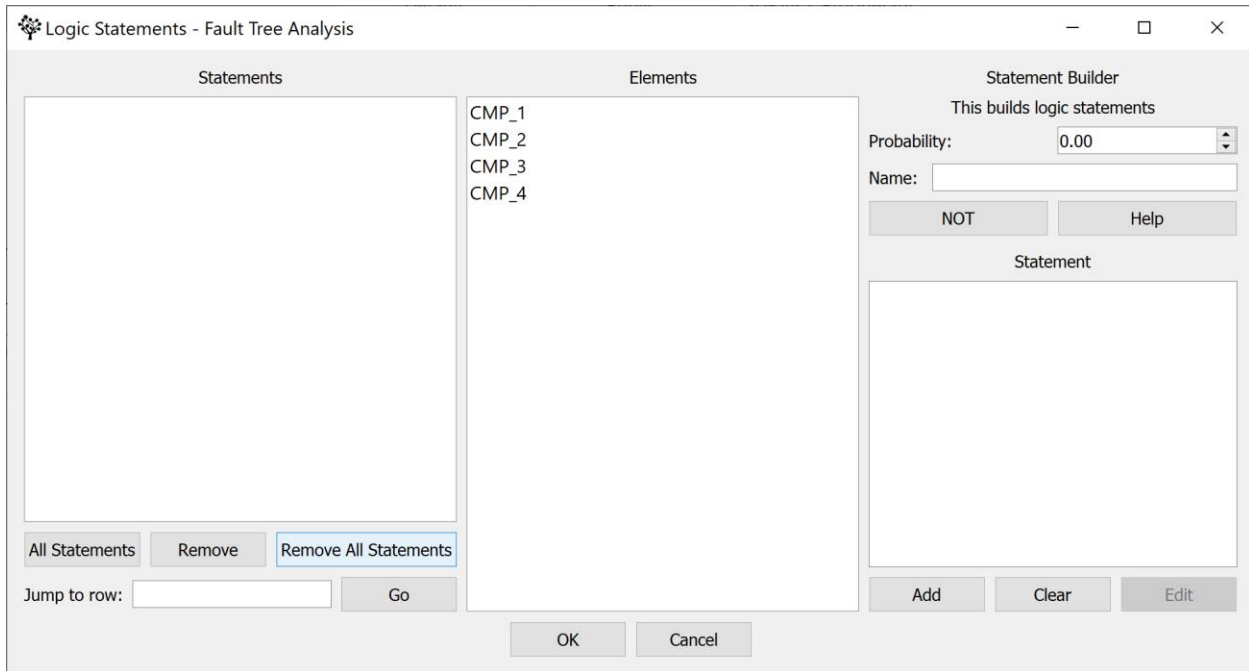


Figure 21. Removing Logic Statements.

To assign weighting factors for a Group/System event, select **Weighting Factors** from the item dialog (Figure 20). This will bring up the Weight Factors dialog which consists of an ordered list of all potential Functional/Failed combinations of the elements along with their associated weighting factors (see Figure 22). To assign weighting factors, the user simply either uses the up/down arrow buttons to change the probability in 0.01 increments or types the desired value directly into the corresponding Weight Factors box. To save the assignments, click the **OK** button.

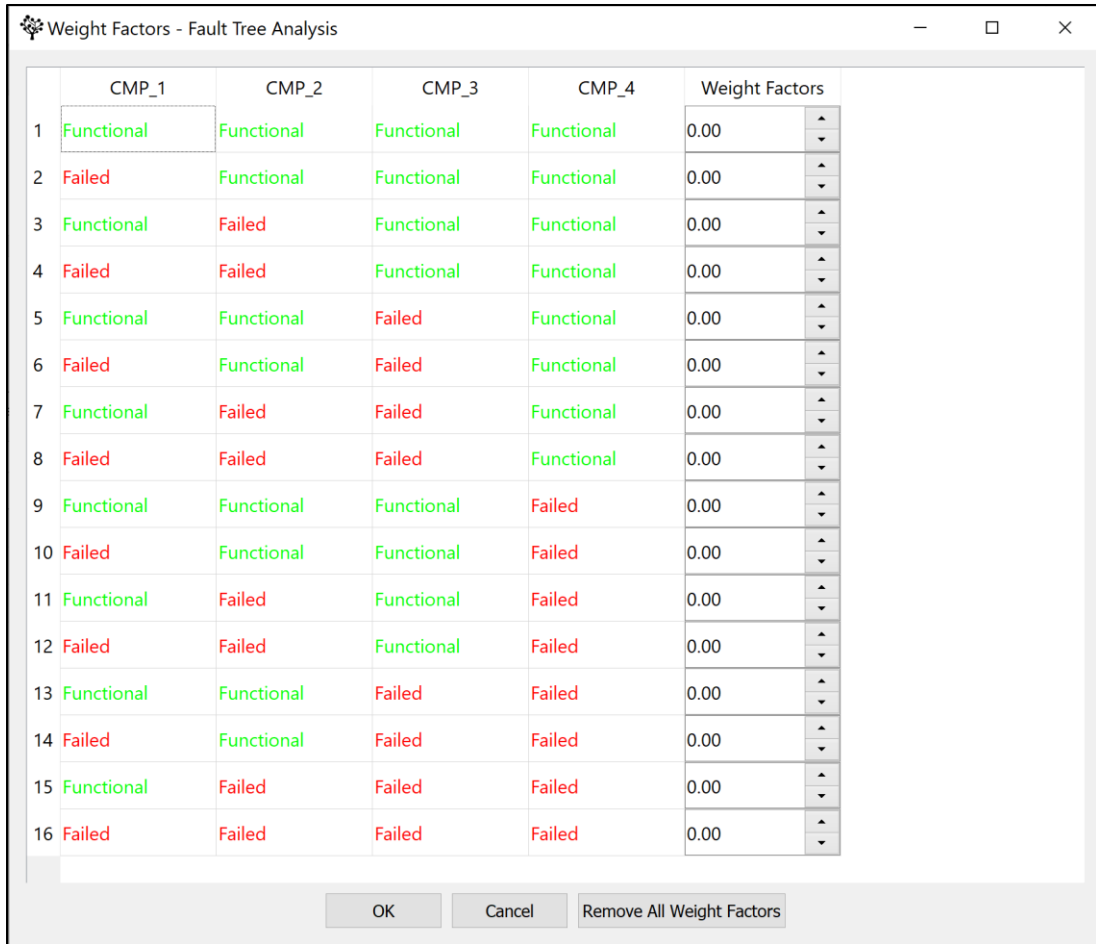


Figure 22. Weighting Factors Dialog.

Note that unlike the logic statement dialog, the user is unable to remove individual weighting factor combinations due to the requirement that weighting factors be entered in a particular order within the MV file. As was done for logic statements, to make those groups with associated weighting factors easily identifiable when viewing the fault tree diagram within FTDT, the label “WGTFACT” is added to the Group/System event element (see Figure 23).

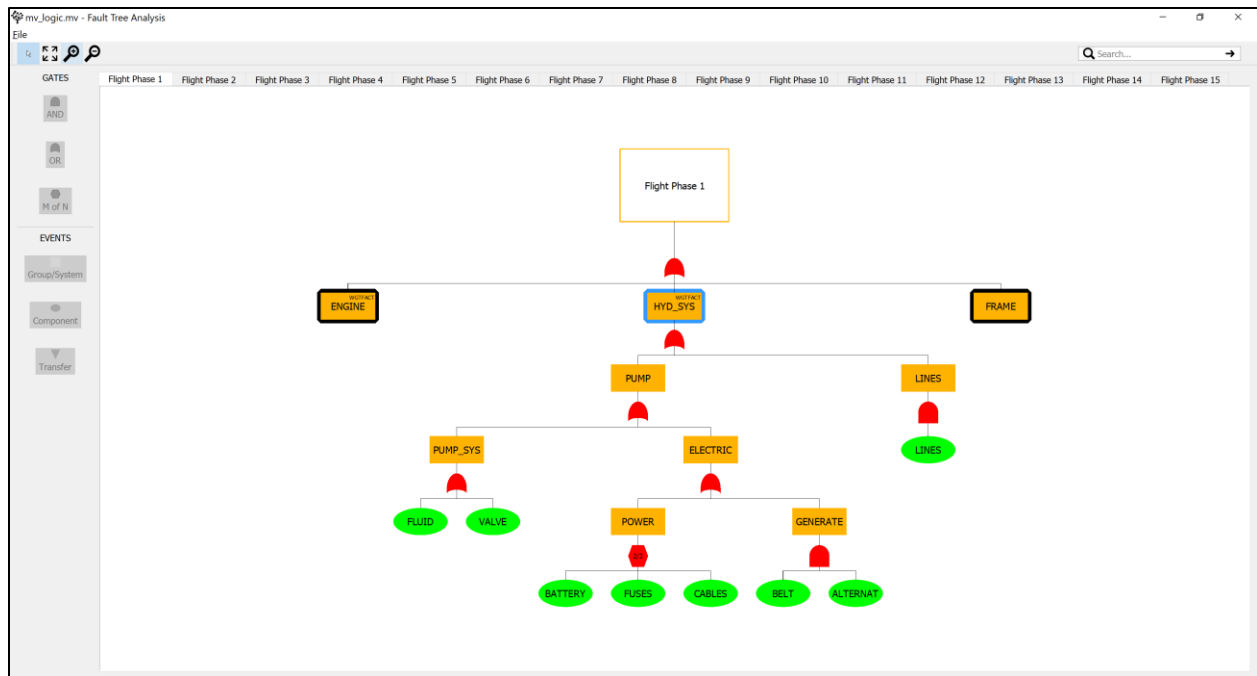


Figure 23. Illustration of a Group/System Event with Assigned Weighting Factors.

In a similar manner as was done with logic statements, to remove all weighting factors (and thereby re-enable the logic statements dialog), open the weight factors dialog, select **Remove all Weight Factors**, and select **OK**. This will reset all values in the Weight Factors column to 0.00 and allow for edits to the logic statements in the item dialog.

6. REFERENCES

1. *COVART 6.9: User's Manual*, JASP-M-16-01, Booz Allen Hamilton/Harris, Dayton, Ohio, 15 September 2017.
2. *Uncontained Engine Debris Damage Assessment Model (UEDDAM) Version 6.0 User and Analyst Manual*, SURVICE-TR-0890, SURVICE Engineering, Belcamp, Maryland, September 2020.

C UEDDAM visualizer version 3 user and analyst manual



Visualizer User
Manual

Uncontained Engine Debris Damage Assessment Model (UEDDAM) Visualizer Version 3.0 User and Analyst Manual

Prepared by:

Ellen M. Phifer

May 2018

Further dissemination only as directed by NAWCWD, Code 418300D, China Lake, CA 93555, May 2018 or higher DOD authority.

This page intentionally left blank.

ABSTRACT

The Uncontained Engine Debris Damage Assessment Model (UEDDAM) Visualizer was developed to allow visualization of the complex data and information generated from a UEDDAM run. It allows visualization of the aircraft geometry, debris hazard zones, debris trajectories, probability plots of the hazard levels, and translational risk angles. The UEDDAM Visualizer was written in the Java programming language, so it is transportable to any computer platform that has Java and Java 3D runtime environments. It has been successfully tested on personal computers running the Windows 7 and 10 operating systems. The PC installation CD contains the Java 6 and Java 3D 1.5.1 runtime environments.

TABLE OF CONTENTS

1.	INTRODUCTION	1
2.	USER INTERFACE AND DEVELOPMENT MODE OPERATION.....	1
2.1	Menu Bar	2
2.1.1	File Menu	2
2.1.2	View Menu.....	7
2.1.3	Tools Menu	10
2.1.4	Help Menu	13
2.2	View Control.....	13
2.3	Display Window	14
2.4	Toolbar	14
2.5	Status Bar	17
3.	ANALYSIS MODE OPERATION	17
3.1	Trajectory Option.....	18
3.2	2D Plots.....	22
3.2.1	Probability Plots.....	23
3.2.2	Risk Angles.....	28
3.3	Significant Iterations.....	31
3.4	Risk Zone	32
3.5	Find	33
4.	REFERENCES	34

LIST OF FIGURES

Figure 1. UEDDAM Visualizer Display.....	2
Figure 2. Open Display Window.	3
Figure 3. Information Window Displaying Numerical Groupings.....	4
Figure 4. Drop-Down Menu Displaying Object Options.....	4
Figure 5. Color Selections.....	6
Figure 6. Transparency Selector Window.	7
Figure 7. Center Option.	9
Figure 8. Projection Types.	9
Figure 9. Tools Menu.....	10
Figure 10. Cross Section Dialog Box.	11
Figure 11. Cutaway Example.....	12
Figure 12. Slice Example.....	13
Figure 13. View Control.	14
Figure 14. Toolbar.	14
Figure 15. Component Data Window.	16
Figure 16. Trajectory►All Option.....	18
Figure 17. Fragments of Displayed Iteration.	19
Figure 18. Hazard Zones Displayed.....	20
Figure 19. Shotline Data Window.	21
Figure 20. Shotline Data Window Details.	22
Figure 21. 2D Plots Window.	23
Figure 22. Probability Plots Options.....	24
Figure 23. Hazard Summary Option.....	25
Figure 24. Averages Option.....	26
Figure 25. Flight Phase by Release Angle Display.....	27
Figure 26. Damage Release Contribution Display.	28
Figure 27. Risk Angles Option.	29
Figure 28. Translational Option.	29
Figure 29. Fore/Aft Option.	30
Figure 30. Component Intercept Window.	31
Figure 31. Significant Iterations Option.	32
Figure 32. Risk Zone Tool.....	33
Figure 33. Find Tool.	34

PREFACE

This manual was prepared for the Naval Air Warfare Center Weapons Division under contract N68936-17-D-0028, Task Order 0001. All questions and comments regarding the content contained herein may be directed as follows:

SURVICE Engineering Company
Applied Technology Operation
ATTN: Mr. Doug Howard
4603A Compass Point Rd.
Belcamp, MD 21017
443.360.4605
doug.howard@survice.com

1. INTRODUCTION

The Uncontained Engine Debris Damage Assessment Model (UEDDAM) Visualizer was developed to allow visualization of the complex data and information generated from a UEDDAM run. It allows visualization of the aircraft geometry, debris hazard zones, and debris trajectories as well as probability plots of the hazard levels and translational risk angles.

The UEDDAM Visualizer was written in the Java programming language, so it is transportable to any computer platform that has Java and Java 3D runtime environments (the installation CD contains the Java 6 and Java 3D 1.5.1 runtime environments). It has been successfully tested on personal computers (PC) running the Microsoft Windows 7 and 10 operating systems.

The UEDDAM Visualizer can be run in two basic modes: development and analysis. First, it can be used to simply view the geometry of a FASTGEN (version 4 or later) file. In development mode, the aircraft geometric model with the extension “.fg4” is read directly by the UEDDAM Visualizer and displayed. In the second basic operating mode, any or all of the data generated from a successful UEDDAM run can be viewed in concert with the geometry. In analysis mode, the UEDDAM control file with the extension “.ucf” is read in by the UEDDAM Visualizer and used to access all of the appropriate data from the UEDDAM run. For more information on the data generated from a UEDDAM run, see the UEDDAM Version 5.1 User’s Manual (Reference 1).

The purpose of this user’s manual is to familiarize the user with the UEDDAM Visualizer’s two basic modes of operation. Section 2 focuses on manipulation of an aircraft geometric model through the Visualizer’s user interface (i.e., development mode). The focus of Section 3 is to explore the UEDDAM analysis features available to the user within the Visualizer (i.e., analysis mode). In order to facilitate location of a specific topic of interest, keywords have been highlighted using bold and colored text throughout this manual.

2. USER INTERFACE AND DEVELOPMENT MODE OPERATION

To launch the UEDDAM Visualizer from a PC running Microsoft Windows*, simply double-click on the *UVis* desktop icon or browse to the directory where UEDDAM is installed and double-click on the file named *Uvis.bat* (located in the *bin* subdirectory). After the UEDDAM splash screen disappears, the Visualizer Display (see Figure 1) will load.

* To launch the UEDDAM Visualizer from a UNIX workstation or PC running Linux, open a terminal window and change directories to the *bin* subdirectory of the UEDDAM installation. At the command prompt, type *uvis*.



Figure 1. UEDDAM Visualizer Display.

The Visualizer Display consists of six elements—**Menu Bar**, **View Control**, **Information Window**, **Display Window**, **Toolbar**, and **Status Bar**—each of which is discussed in more detail in the sections that follow.

2.1 MENU BAR

In Figure 1, the **Menu Bar** can be seen at the top left of the Visualizer Display. The **Menu Bar** consists of the following selections: **File**, **View**, **Tools**, and **Help**.

2.1.1 File Menu

Selecting the **File** menu with the left mouse button will open a drop-down menu consisting of **Open**, **Load View**, **Save View**, and **Exit** options as well as a list of recently opened files. To open a file not in the recently opened list, select the **Open** option with the left mouse button. A window will appear, allowing you to browse to directories where *.ucf or *.fg4 files are stored, as illustrated in Figure 2.

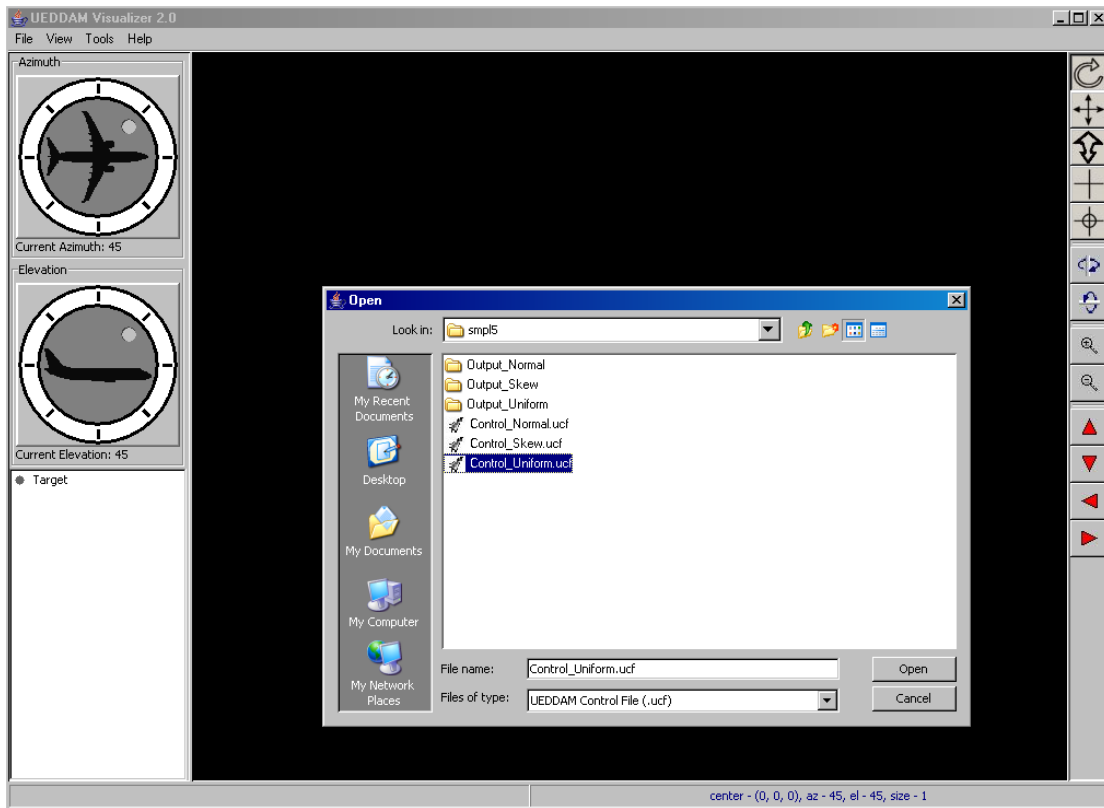


Figure 2. **Open** Display Window.

Select the desired file with the left mouse button and choose the Open option button to load the file. The left-most portion of the **Status Bar** (see Figure 1) will show the status of the data loading process. When data are successfully read into the Visualizer, the geometry information is loaded into memory and the FASTGEN file name will appear in the **Information Window**. Double clicking the left mouse button on the file folder that appears next to the FASTGEN file name will then expose the numerical groupings found within the FASTGEN file. Figure 3 illustrates the **Information Window** display of the numerical groupings in sample case five's FASTGEN file, *Business_Jet.fg4*.

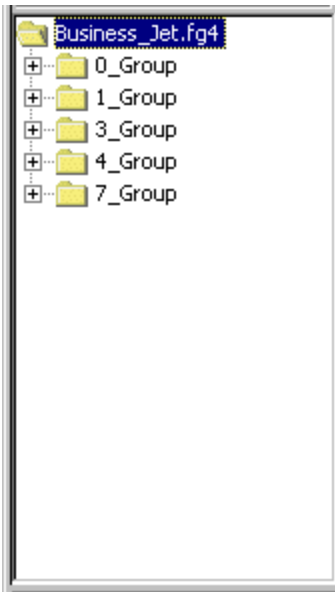


Figure 3. **Information Window** Displaying Numerical Groupings.

Double clicking the left mouse button on any of the groups will then drop down a listing of all the component identification numbers found in that specific grouping. A single click of the right mouse button on the file folder, numerical group, or any component ID number in the **Information Window** will give a menu of display options for the selected object. Figure 4 illustrates the drop-down listing and the available options.

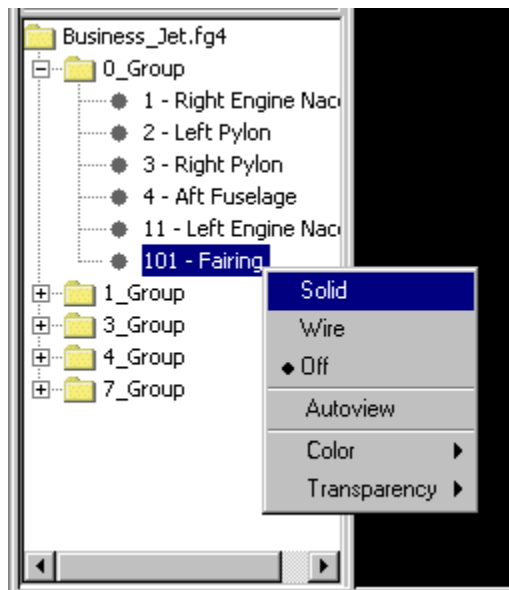


Figure 4. Drop-Down Menu Displaying Object Options.

The display options include **Solid**, **Wire**, **Off**, **Autoview**, **Color**, and **Transparency**.

- **Solid**, **Wire**, and **Off** – Clicking either the right or left mouse button on any one of these three options will take the selected object (file, folder, number grouping, or component) and display it as either a solid (**Solid**) or wireframe (**Wire**) three-dimensional (3-D) representation in or remove (**Off**) it from the **Display Window**.
- **Autoview** – Clicking on the **Autoview** option for any object currently displayed (in either solid or wireframe mode) will re-center the view such that the center of the object selected coincides with the center of the **Display Window**. Once centered, the Visualizer will zoom in or out on the object such that the extents of the object selected are just within the bounds of the **Display Window**. (Note that the **Autoview** option does not change the azimuth or elevation angle of the view.)
- **Color** – The **Color** option contains the following basic color choices, which, if selected, is applied to the specified object: **Red**, **Orange**, **Yellow**, **Green**, **Blue**, **Indigo**, and **Violet**. Two additional options, **Default** and **Select**, are available under the **Color** menu as well (note that the **Default** option is not available at the file level). Choosing the **Default** option will return the selected object(s) to the default color scheme. Choosing the **Select** option will open a separate window displaying various color swatches. There are also options to enter **HSB** or **RGB** values. Figure 5 illustrates the **Color** option as well as the color swatch window opened when the **Select** option is chosen.

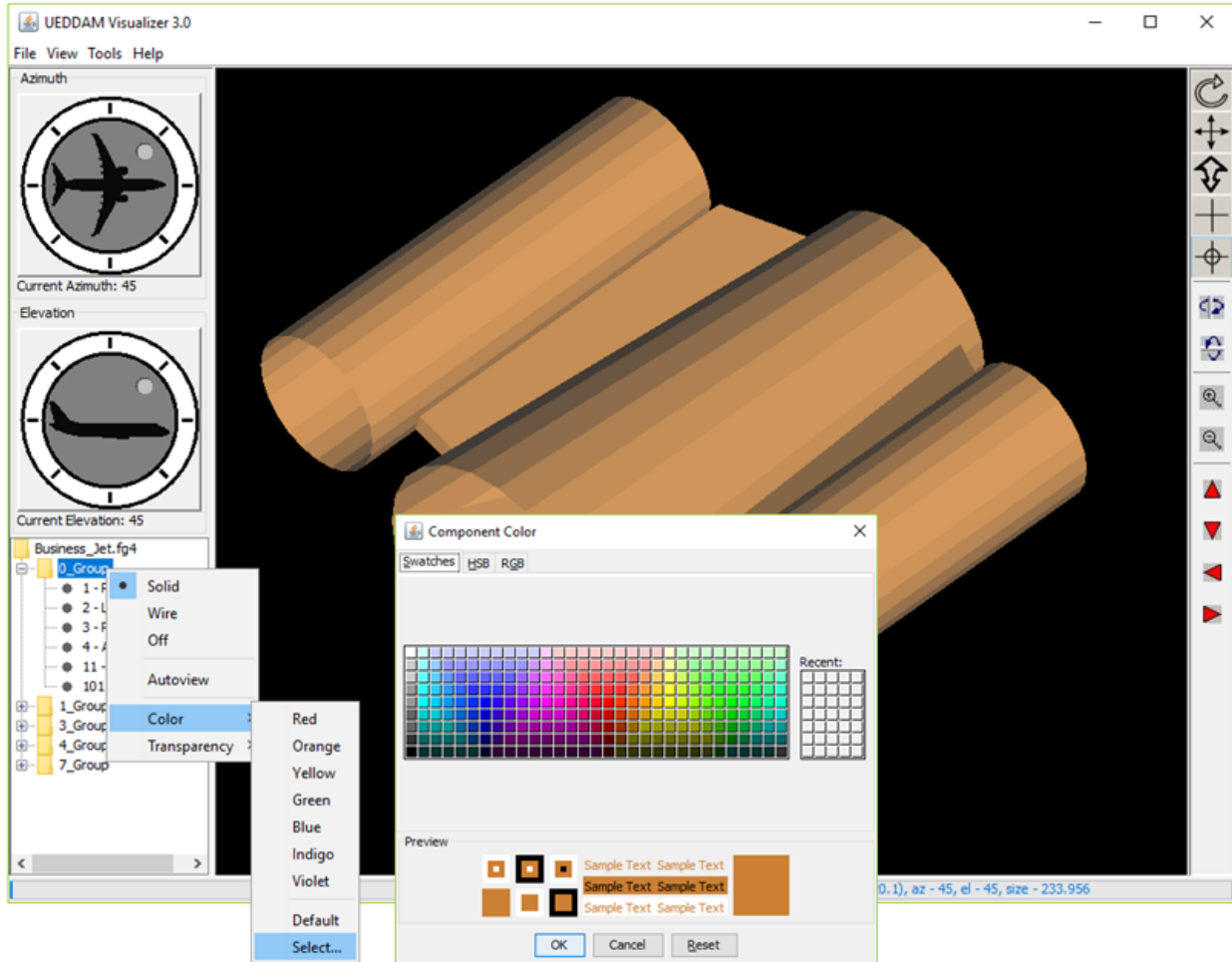


Figure 5. **Color** Selections.

- **Transparency** – Selecting the **Transparency** option will drop down a menu with the following preset values: 0% (default), 25%, 50%, and 75%. A **Select** option is also available to obtain additional values for transparency to the preset ones. The **Select** option brings up a transparency selector window with a slider (see Figure 6).

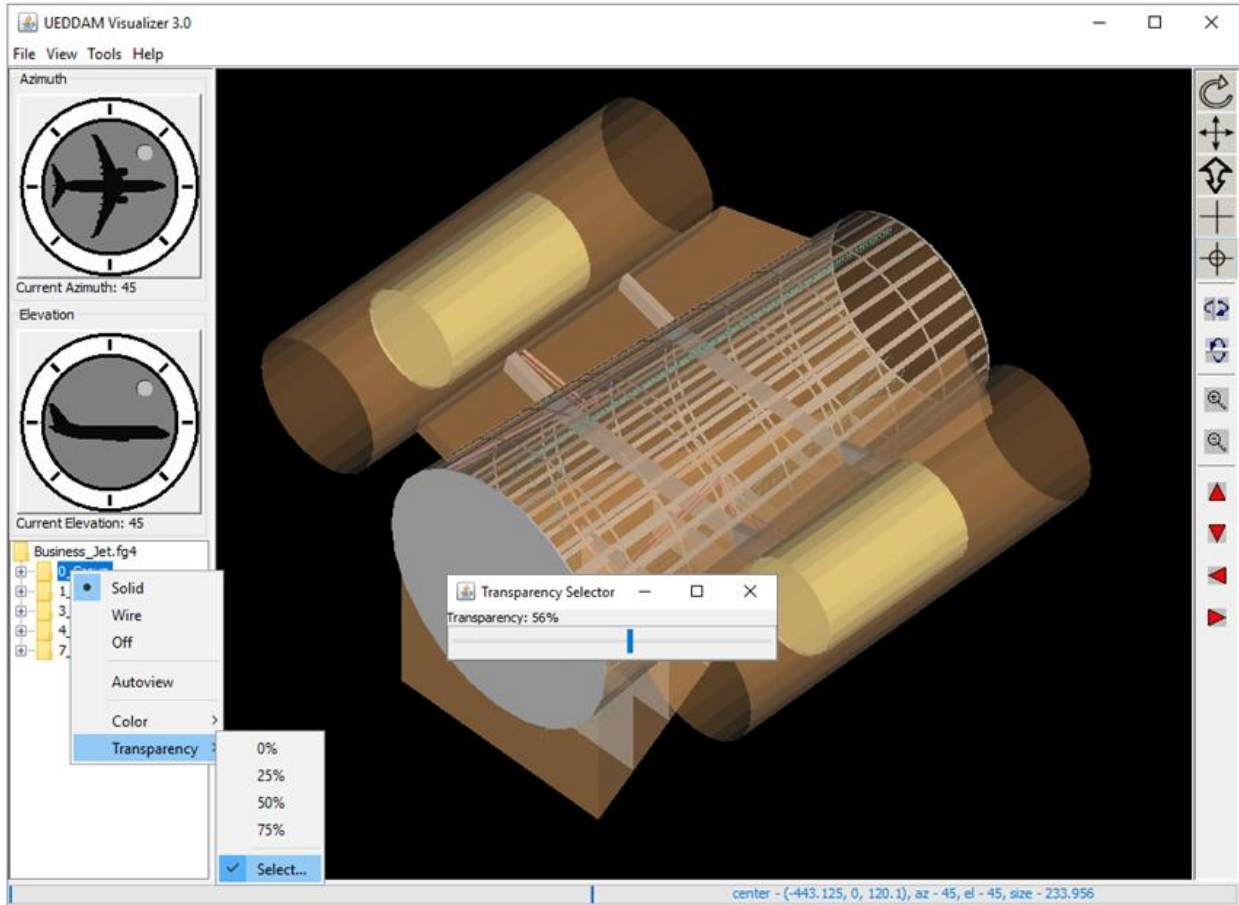


Figure 6. **Transparency** Selector Window.

The next two options under the **File** menu (i.e., **Save View** and **Load View**) allow the user to save and reload all of their display settings for a FASTGEN file (e.g., color, transparency, etc.). Selecting the **Save View** option opens a window which allows the user to save the display settings to a file; the information is saved to a binary file with the automatic extension *.vif (UEDDAM View Information File). To open a previously saved view file, select the **Load View** option and browse to and select the *.vif file of interest.

Selecting **Exit** with the left mouse button will exit the UEDDAM Visualizer program.

2.1.2 View Menu

Selecting the **View** menu from the **Menu Bar** with the left mouse button will open a drop-down menu consisting of the following options: **Trajectory**, **Reset**, **Autosize**, **Autocenter**, **Center**, and **Ortho View (Perspective View)** along with a set of standard view angles. The **Trajectory** feature is discussed in Section 3 of this manual; the remaining **View** menu options assist the user in viewing their FASTGEN geometry file:

- **Reset** – Selecting the **Reset** option performs a similar set of operations as the **Autoview** option discussed in the previous section. For any object(s) currently displayed in the **Display Window**, selecting the **Reset** option will reset the view azimuth/elevation angles to $45^{\circ}/45^{\circ}$, re-center the view such that the center of the object(s) displayed coincides with the center of the **Display Window**, and zoom in or out on the object(s) such that the extents of the object(s) displayed are just within the bounds of the **Display Window**. (Note that the **Reset** option does change the azimuth and elevation angles of the view.)
- **Autosize** and **Autocenter** – The **Autosize** and **Autocenter** options perform two of the three **Reset** operations; namely, zooming and re-centering, respectively. Upon selecting the **Autosize** option, the Visualizer zooms in or out on the object(s) currently displayed in the **Display Window** such that the extents of the object(s) displayed would be just within the bounds of the **Display Window** if the object(s) was(were) centered. The **Autocenter** option re-centers the view such that the center of the object(s) displayed coincides with the center of the **Display Window**.
- **Center** – The **Center** option allows the user to specify the location of the center of the **Display Window**. When the **Center** option is selected, a window appears for the user to enter the x, y, z location (in target coordinates) to be used for the display center. The title of this new window is the current x, y, z location of the display center (which can also be found on the right side of the **Status Bar** [see Section 2.5]). The new center location must be entered using commas to separate the individual numbers. Figure 7 illustrates the **Center** option.

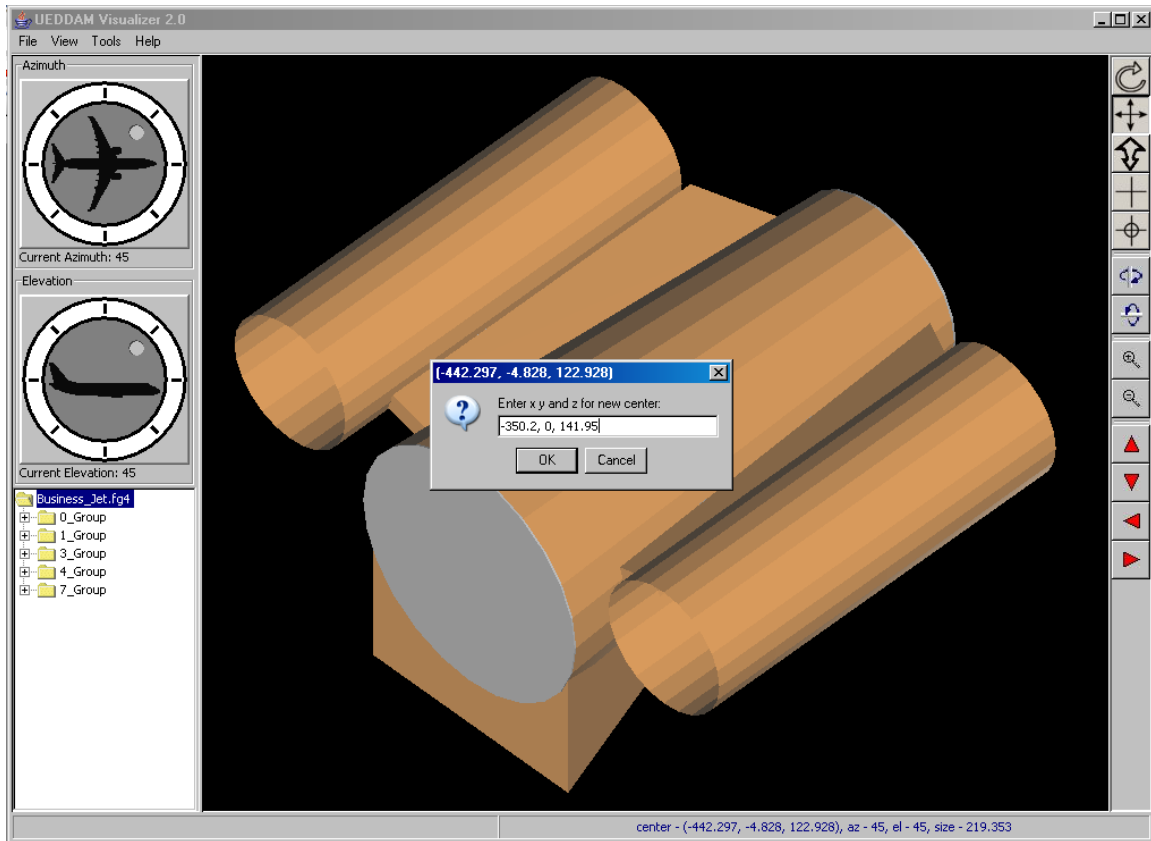


Figure 7. **Center** Option.

- **Ortho View (Perspective View)** – Selecting the **Ortho View** option changes the view type from perspective (default) to orthographic. The user returns the view to a perspective one by selecting the **Perspective View** option from the **View** menu. Figure 8 illustrates the difference between the two projection types.

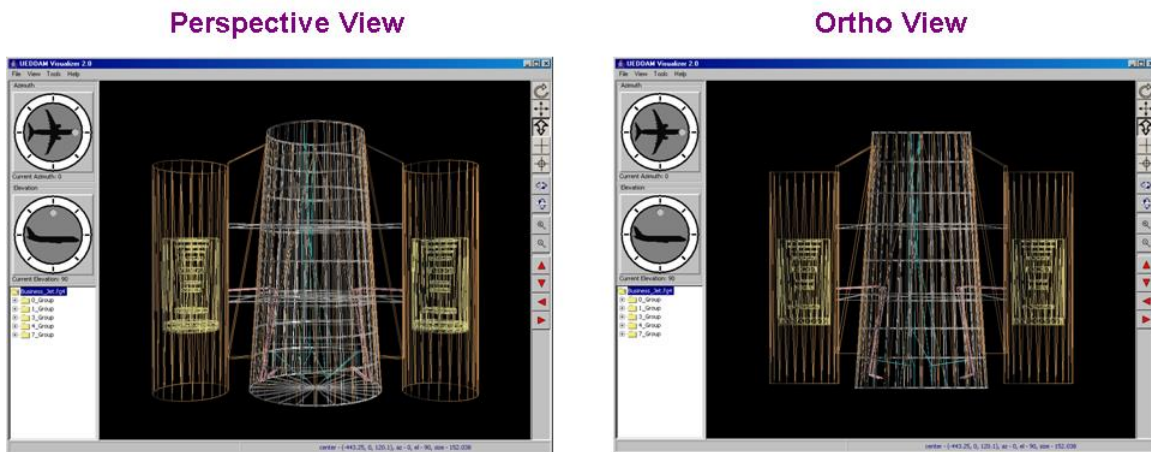


Figure 8. Projection Types.

- Standard View Angles – When the user selects one of the available standard view angles (i.e., **45,45**, **front**, **rear**, **port**, **starboard**, **top**, or **bottom**) from the **View** menu, the Visualizer modifies the azimuth (az) and elevation (el) angles of the **Display Window** accordingly (i.e., **45,45**→45°az, 45°el; **front**→0°az, 0°el; **rear**→180°az, 0°el; **port**→90°az, 0°el; **starboard**→270°az, 0°el; **top**→0°az, 90°el; or **bottom**→0°az, -90°el).

2.1.3 Tools Menu

Selecting the **Tools** menu with the left mouse button will drop down a menu consisting of the following options: **Background Color**, **Capture Image**, **2D Plots**, **Significant Iterations**, **Risk Zone**, **Cross Section**, and **Find**. Figure 9 shows the UEDDAM Visualizer **Tools** menu. The **2D Plots**, **Significant Iterations**, **Risk Zone**, and **Find** options are discussed in Section 3.

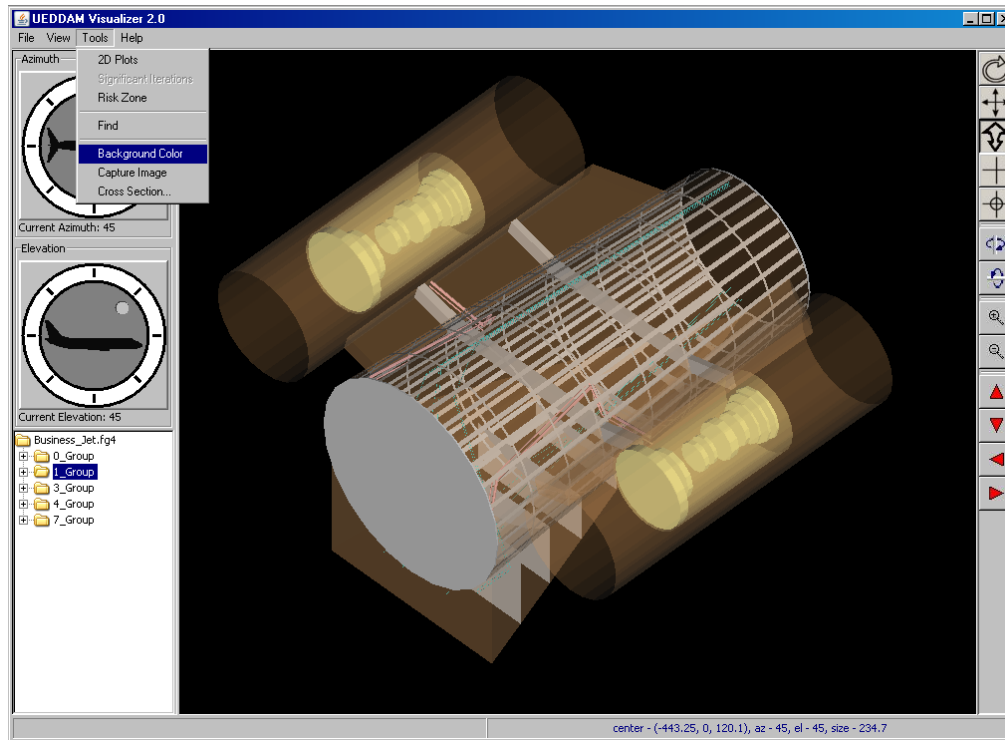


Figure 9. **Tools** Menu.

2.1.3.1 Background Color

The **Background Color** tool allows the user to change the color of the **Display Window** background (default = black). Selecting this option with the left mouse button will open an identical window to the color swatch display shown in Figure 5. The window displays predetermined color swatches which can be selected by simply clicking on the desired color. There are also options to input **HSB** and **RGB** values, if desired.

2.1.3.2 Capture Image

The **Capture Image** tool allows the user to save an image of the current **Display Window** for later reference. Selecting this option will open a file save window where the user can specify a name and location to save the JPEG image. The name of the file defaults to the azimuth, elevation of the display, but it can be changed to any desired naming convention.

2.1.3.3 Cross Section

The **Cross Section** tool allows the user to visualize a cross-section of the FASTGEN geometry. When the **Cross Section** option is selected from the **Tools** menu, a dialog box is opened that allows the user to choose between two types of cross-sections (**Slice** or **Cutaway**) as well as specify the **Azimuth**, **Elevation**, and **Origin** (x,y,z-coordinates) of the cross-section (see Figure 10).

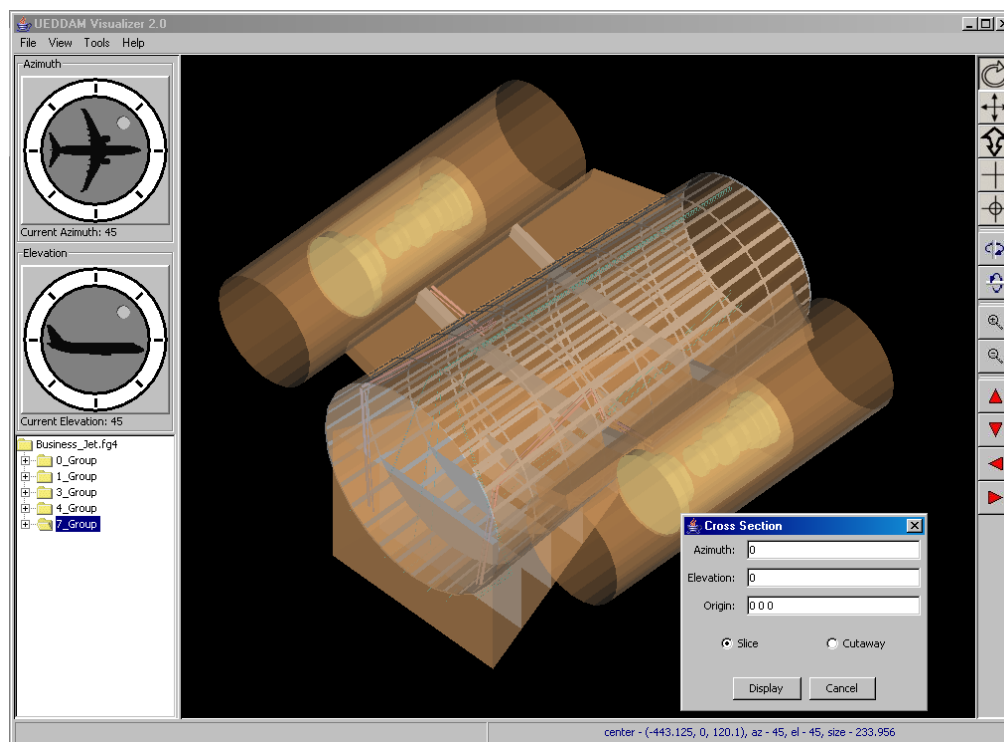


Figure 10. **Cross Section** Dialog Box.

The two types of cross-sections available to the user are described as follows:

- **Cutaway** – When the **Cutaway** option is selected and the user presses the Display button, the Visualizer internally draws a plane through the FASTGEN geometry currently displayed in the **Display Window**. This virtual plane passes through the user-supplied **Origin** and is parallel to the **Azimuth** and **Elevation** angles entered in the **Cross Section** dialog box. The Visualizer then opens a secondary display

window. Presented in this display window is the geometry that remains once any geometry located above/in front of the virtual plane is removed. Note that the geometry displayed in the secondary display window retains some of the settings from the main [Display Window](#) (e.g., color and transparency), but not all settings (e.g., view angle, display center, and zoom level). An example of this type of cross-section is shown in Figure 11. Another way to think about the **Cutaway** cross-section is that it allows the user to see *inside* the geometry by *cutting away* those elements that are *blocking the view*.

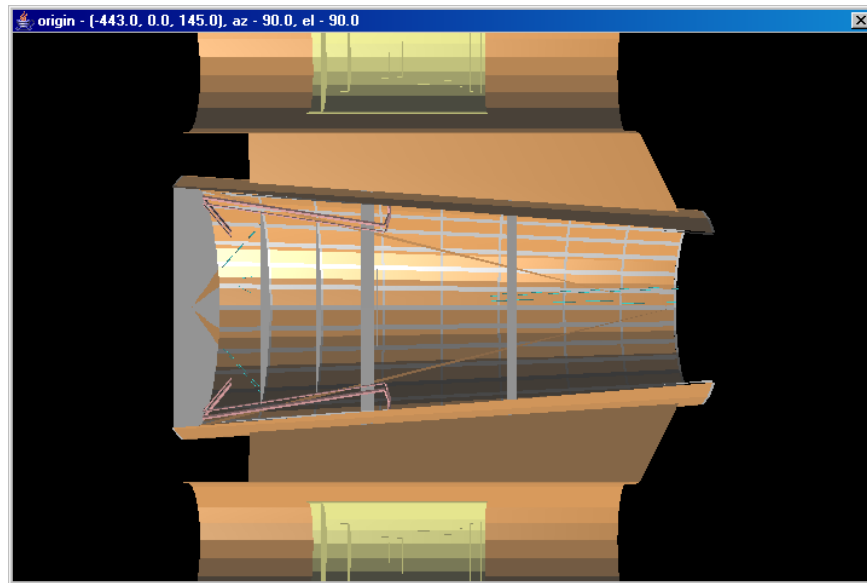


Figure 11. **Cutaway** Example.

- **Slice** – The **Slice** option allows the user to display a single *slice* of the geometry. As in the **Cutaway** option, upon selecting the **Slice** option and pressing the Display button, the Visualizer internally draws a plane parallel to the user-selected **Azimuth** and **Elevation** angles through the geometry. However, rather than the virtual plane being made to pass through the **Origin**, the Visualizer draws two planes, one slightly above/in front of the specified **Origin** and one slightly below/behind it. The Visualizer then opens a secondary display window in which only the elements of the geometry that lie within the bounds of the two virtual planes are displayed. An example of this type of cross-section is shown in Figure 12.

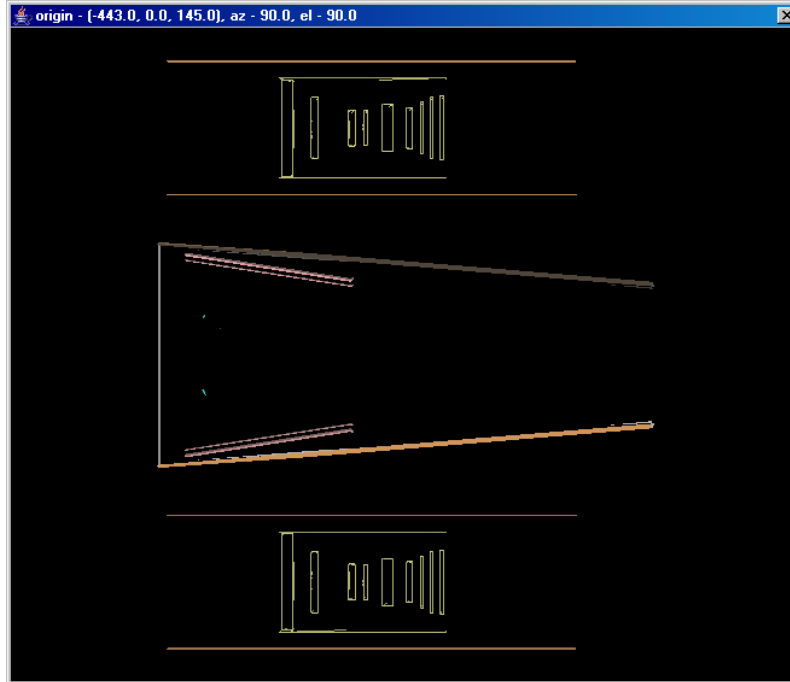


Figure 12. **Slice** Example.

2.1.4 Help Menu

Currently, the only option under the **Help** menu selection is **About UEDDAM Visualizer**. Selecting this option will bring up an information window containing the UEDDAM Visualizer version number.

2.2 VIEW CONTROL

Located directly under the **Menu Bar** is the **View Control**. As shown in Figure 13, the **View Control** consists of two dials labeled **Azimuth** and **Elevation**. The current azimuth and elevation values are displayed under each dial. You will notice that the default view when the *.ucf or *.fg4 file is opened is 45° azimuth, 45° elevation. Inside of the shaded area of the dial is a circle above and to the right of the aircraft symbol. The circle represents your viewpoint. By placing the cursor on the viewpoint, holding down the left or right mouse button and dragging it around the shaded area, you are able to change the azimuth and elevation of the geometry in the **Display Window**. Outside of the shaded area, selection ticks are given at 45° increments. You may also click the left or right mouse button on any of the predetermined azimuth or elevation ticks to *snap* to that aspect. The current azimuth and elevation are always updated under each view control dial.

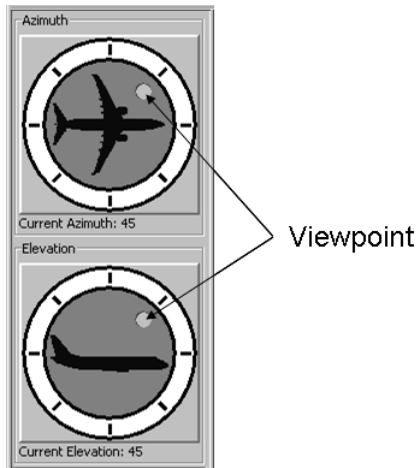


Figure 13. **View Control.**

2.3 DISPLAY WINDOW

The **Display Window** is the area of the screen where the geometry is displayed. The UEDDAM Visualizer enables the user to view detailed 3-D information directly through the **Display Window**.

2.4 TOOLBAR

The **Toolbar** consists of 13 buttons and is located on the right side of the Visualizer display. Figure 14 shows the UEDDAM Visualizer **Toolbar**.

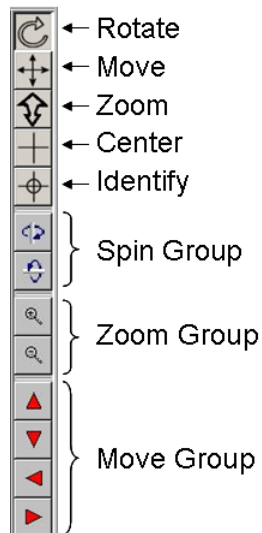


Figure 14. **Toolbar.**

 **Rotate**

The **Rotate** button is an additional way, beyond the **View Control** dials (see Section 2.2), to control the viewing angle of the display. Using the left mouse button, click on the **Rotate** button to activate this feature. Then, position the mouse so that it is in the **Display Window**. Finally, while holding down the left mouse button, move the mouse in any direction to rotate the viewing angle.

 **Move**

To move the geometry around the **Display Window**, select the **Move** button. Once the **Move** button has been depressed, position the mouse so that it is in the **Display Window**, hold down the left mouse button, and move the mouse in any direction to move the display center and reposition the view.

 **Zoom**

The **Zoom** button is used to move the display closer to or farther from the geometry. Zooming is accomplished by clicking the left mouse button over the **Zoom** button to turn this feature *on*. Then, move the mouse over to the **Display Window** and, while holding the left mouse button down, move the mouse up to zoom in on the geometry or down to zoom out.

 **Center**

The **Center** button allows the user to re-center the **Display Window** using the mouse. To use, first click the **Center** button to activate this feature. Once activated, position the mouse so that it is in the **Display Window** and click the left mouse button wherever the new view center should be located.

 **Identify**

The **Identify** button provides the user the ability to determine the number and name of a particular component of the geometry currently being displayed in the **Display Window**. Pressing the **Identify** button turns this feature *on*. To identify a component, move the mouse so that it is in the **Display Window** and, with the left mouse button, click on a component. The component identification number and name (if one has been provided in the FASTGEN geometry file) are displayed on the left side of the **Status Bar** (see Section 2.5). Note that information is displayed only for the outermost component (i.e., if the component of interest is beneath another component in the **Display Window**, information for the blocked component can be obtained only after the top component is turned off).

If a *.ucf file has been opened (for a successfully completed UEDDAM run) in addition/as opposed to a *.fg4 file, the **Identify** button provides more information than just the component identification number and name. When the user clicks on a critical component in the **Display Window**, a secondary display window is opened that provides the following additional

component data: **COVART NAME**, **Material**, **% Density**, **Hazard Levels of Criticality**, **NearField Association**, and **COMALIAS** (see Figure 15). Consult Reference 1 for more information on any of these data fields.

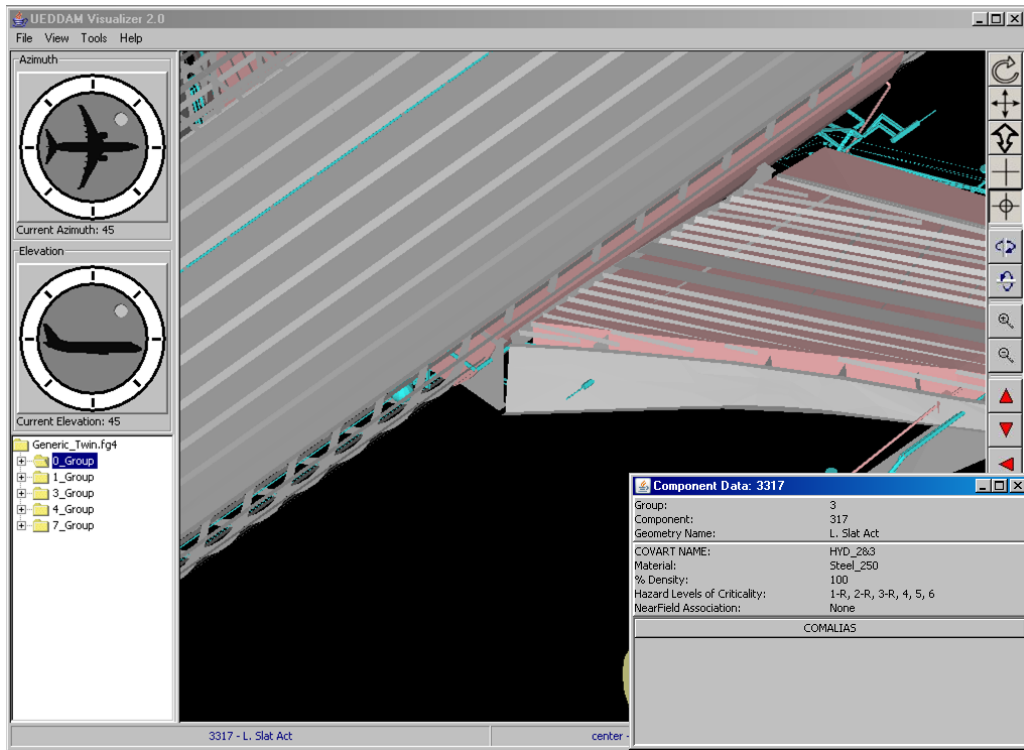


Figure 15. Component Data Window.



Spin Group

The **Spin Group** of buttons is used to automatically rotate the displayed geometry in real time. To rotate about the Z axis, click the left mouse button over the top button, turning it *on*. To stop rotation, click on the same button again, turning it *off*. To rotate about the Y axis, select the bottom button. To stop rotation, click again to turn *off*. You may have both buttons turned *on* at the same time to rotate about both axes simultaneously.



Zoom Group

The **Zoom Group** of buttons is used to move the display closer to or farther from the geometry. Pressing the top (or bottom) button of the **Zoom Group** once moves the display one increment closer to (or further away from) the geometry. Holding down either of the two **Zoom Group** buttons results in continuous zooming in the specified direction.



Move Group

The **Move Group** of buttons is used to move the geometry about the screen in the direction of the selected arrow. These buttons work in the same way that the **Zoom Group** of buttons do. A single click will move the view of the screen an increment. Holding the left mouse button down over the button will result in continuous movement until the button is released.

2.5 STATUS BAR

The **Status Bar** is located along the bottom of the UEDDAM Visualizer Display, under the **Information** and **Display Windows**, and can present three types of information: **Progress**, **Settings**, and **Identification**. While a file is loading, **Progress** (i.e., the percentage of information that has been read in) is tracked on the left side of the **Status Bar**. Once a file has been loaded and the geometry is being displayed in the **Display Window**, the left side of the **Status Bar** is used to present **Identification** information (i.e., component identification information that results from pressing the **Identify** button on the **Toolbar** [see Section 2.4]). The right side of the **Status Bar** displays the current **Settings** (i.e., center, viewing angles, and size) of the **Display Window**.

3. ANALYSIS MODE OPERATION

In order to use the analysis tools discussed in this section, the user must first open a *.ucf file from a successfully completed UEDDAM run (see Section 2.1.1 for instructions on how to open a file). This UEDDAM run must have included the following output files and options (see Sections 5.3.1.10 and 5.3.1.11 of Reference 1) to access certain features (listed after each file/option) in the Visualizer:

- VISUALIZER FILE – display **Hazard Zones**, **Comp Risk Angs**, and *Shotlines*
- SUMMARY FILE and DETAIL FILE – display **Probability Plots**
- INTERCEPT FILE – display **Risk Angles** and **Comp Risk Angs**
- Debugging flag set to “1” – display *Shotlines* and **Shotline Information**
- TRAJECTORY FILE – display **Risk Zone**
- One or all of the following: SIGITSYS FILE, SIGITCRIT FILE, SIGITCHOLE FILE, SIGITSHOLE FILE – display **SIGITSYS**, **SIGITCRIT**, **SIGITCHOLE**, and/or **SIGITSHOLE** *Shotlines*

3.1 TRAJECTORY OPTION

The **Trajectory** option under the **View** menu is a cascade-type menu (as indicated by the ► symbol) with additional options displayed to the right: **SIGITSYS**, **SIGITCRIT**, **SIGITCHOLE**, **SIGITSHOLE**, **All**, and **None**. When a “.ucf” file is initially loaded, the default **Trajectory View** is **None**. Selecting any option other than **None** with the left mouse button will display a *Shotlines* file folder under the FASTGEN file name in the **Information Window**, as shown in Figure 16. The **All** option enables the trajectories for all of the events assessed in the “.ucf” file to be displayed, whereas the **SIGITSYS**, **SIGITCRIT**, **SIGITCHOLE**, and **SIGITSHOLE** options limit the visualization to only those trajectories that are associated with iterations that meet a specified criterion (see sections 5.3.1.7 and 7.7 of Reference 1 for more information regarding the significant iterations files). For any of the options, selecting the *Shotlines* folder with a double click of the left mouse button will drop down a listing of the *Events* of the “.ucf” file. Selecting any of the *Event* folders displayed with the left mouse button will show a listing of the *Release Points* for that specific event. Similarly, selecting any of the *Release Point* folders will show a listing of the *Iterations* for that release point, et cetera until the individual *Rays* used to evaluate the engine debris are displayed (see Figure 16).

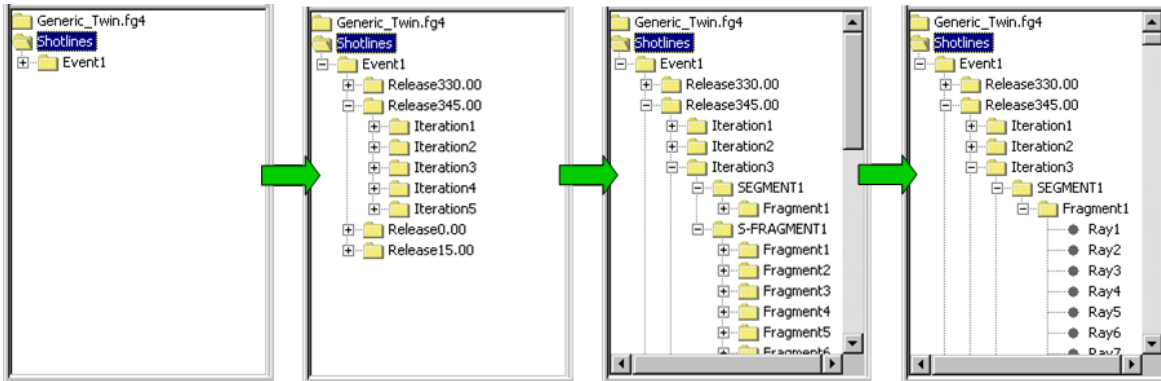


Figure 16. **Trajectory ► All** Option.

On any of the levels of the *Shotlines* tree, clicking the right mouse button will display the **Solid**, **Wire**, **Off**, **Autoview**, **Color**, and **Transparency** menu options previously discussed in section 2.1.1. These selections allow the user to view the individual trajectories that make up the engine debris in the **Display Window**. Figure 17 illustrates the display of all of the fragment trajectories (represented as rays) of a single *Iteration* (Iteration 3) of a single *Release Point* (Release0.00) of the *Event* (Event1) assessed in Sample Case 6.

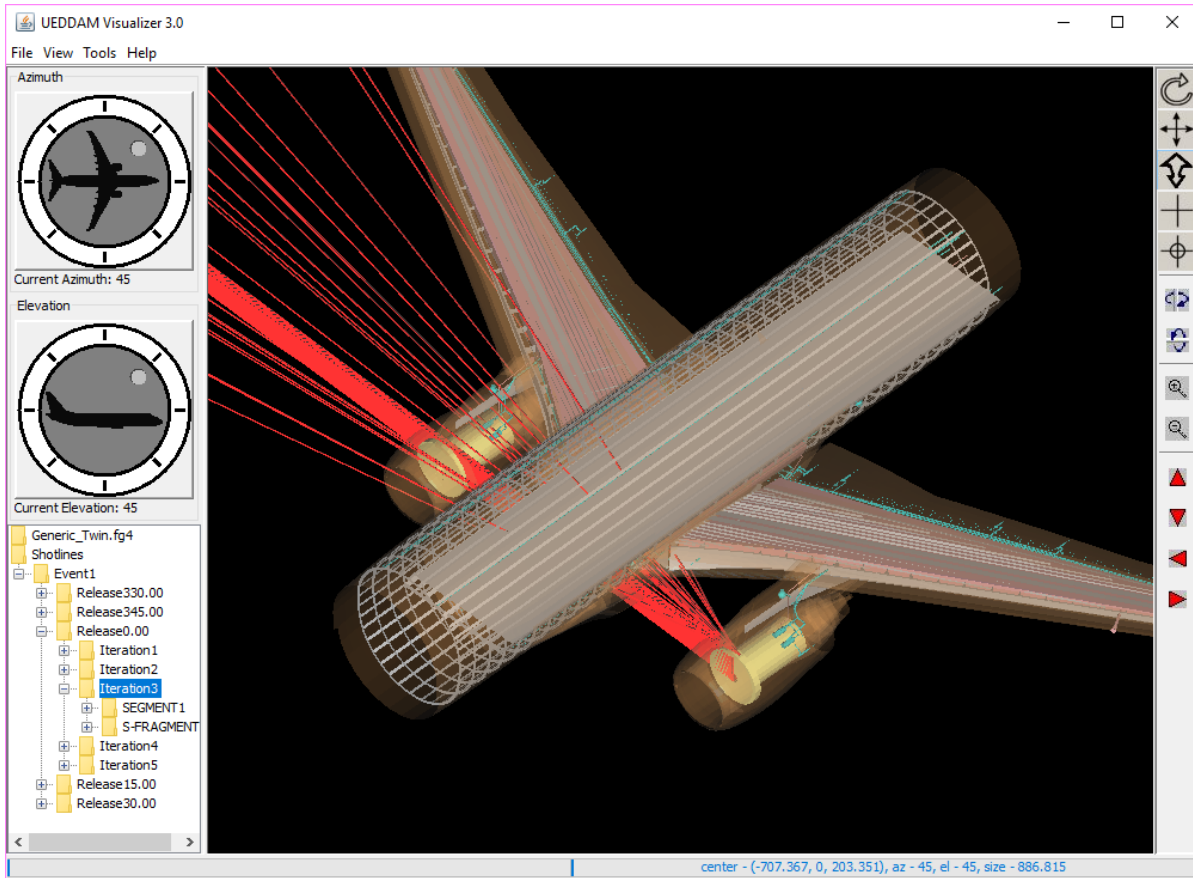


Figure 17. Fragments of Displayed Iteration.

In addition to the viewing options previously discussed, at both the *Event* and *Release Point* levels of the *Shotlines* tree, a further option is available: **Hazard Zones**. The **Hazard Zones** option is a toggle switch option. When the **Hazard Zones** toggle is *on* the entire range of possible trajectories for the selected *Event* or *Release Point* is shaded in a transparent blue (for additional information regarding the definition of a hazard zone, see Reference 1, Section 3.4). Figure 18 illustrates the **Hazard Zones** menu toggled *on* and the display of the hazard zone for the same *Event* (Event1) and *Release Point* (Release0.00) as in Figure 17.

Finally, if the appropriate outputs have been specified in the Control file, an additional option will appear at the *Event* level of the *Shotlines* tree: **Comp Risk Angs**. When the **Comp Risk Angs** menu option is toggled *on*, a secondary window is opened containing a list of all components in the target description that could potentially be hit by the debris generated for the selected *Event*. Next to each component is a check box. When a component is selected from this secondary window, the Visualizer will attempt to draw a wedge representing the component's translational and fore/aft risk angles in the **Display Window** (for a different method of displaying this same information, see section 3.2.2).

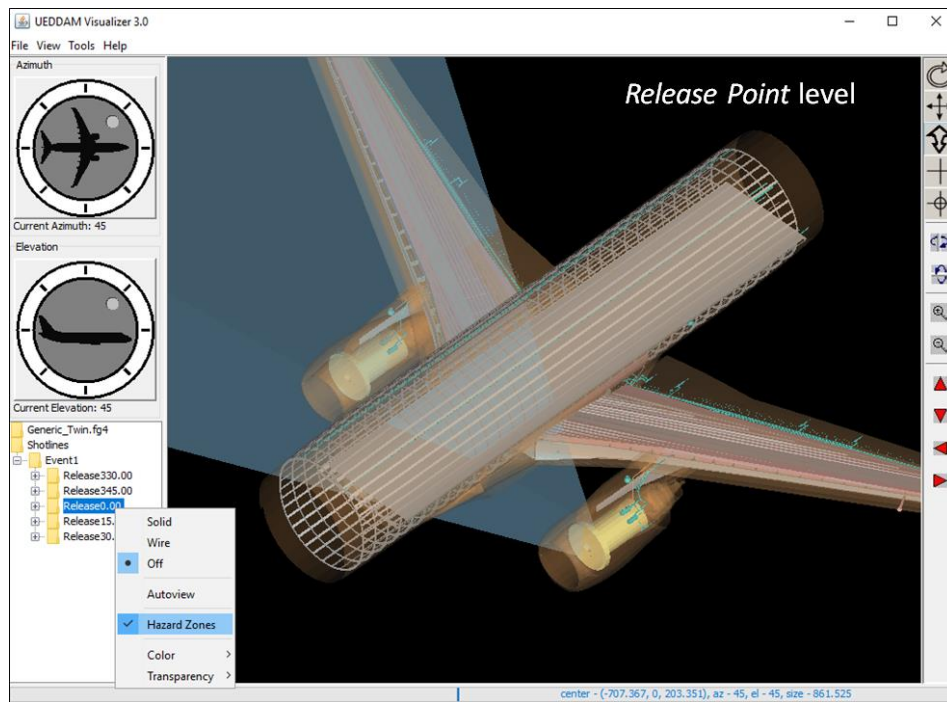
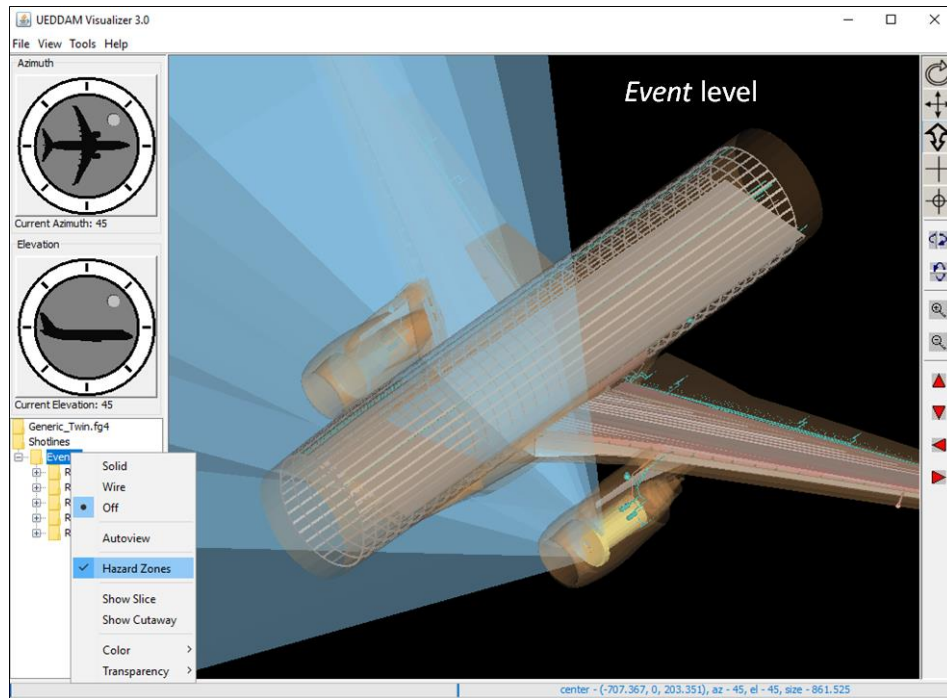


Figure 18. Hazard Zones Displayed.

When the individual *Rays* from any level of the *Shotlines* tree are displayed in the **Display Window** and the **Identify** button has been depressed, clicking on any single *Ray* in the **Display Window** with the left mouse button will color that ray blue and open a supplementary data window (as shown in Figure 19). The shotline data window is divided into four sections: the

Menu Bar, the **Fragment Information** section, the **Shotline Information** section, and a **Shotline Selection** section, as illustrated in Figure 20.

- The **Menu Bar** section of the shotline data window currently only contains a single menu (**Show**) with a single option (**Comps Hit**). Selecting the **Comps Hit** option with the left mouse button will shade all of the components hit by the particular *Ray* blue. Toggling this option *off* returns the components to their default color.
- The **Fragment Information** section of the shotline data window displays specific information about the particular *Ray* selected. The information provided includes the X, Y, Z coordinates and azimuth and elevation angles of the *Release Point* of this *Ray* (in the target coordinate system). Additionally provided is the material type, mass, ejection velocity, yaw angle, and roll angle of the debris associated with this *Ray*.

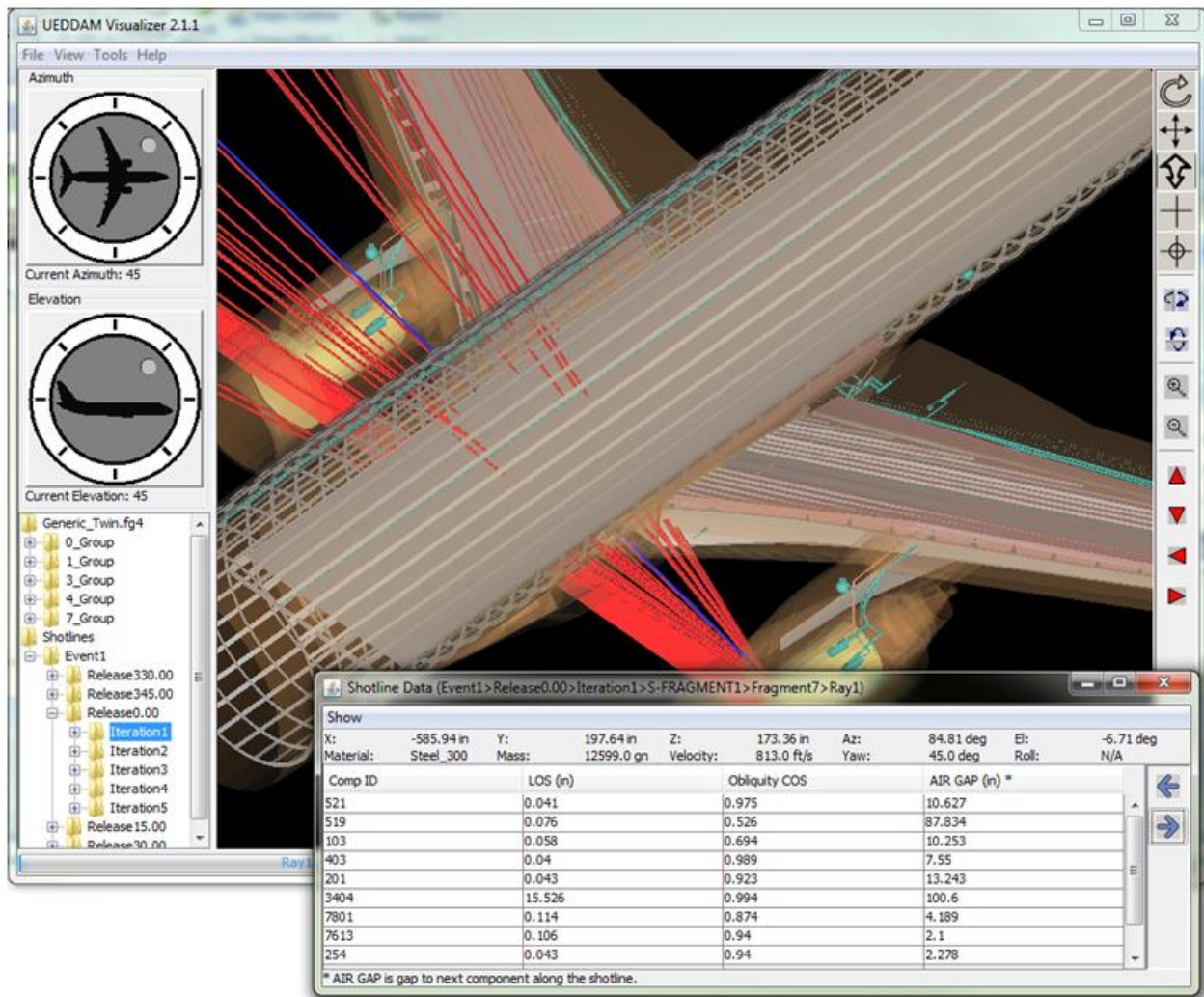


Figure 19. Shotline Data Window.

- The **Shotline Information** section contains a listing of all of the components (in order of impact) hit by the particular *Ray*, along with the line-of-sight thickness through the component, the cosine of the obliquity angle of impact, and the air gap to the next component along the shotline.

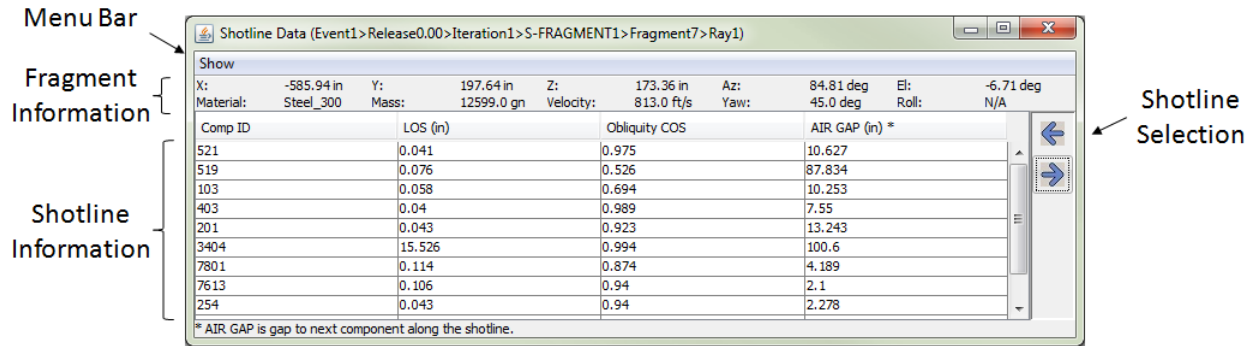


Figure 20. Shotline Data Window Details.

- The **Shotline Selection** section contains two buttons (left and right arrows) that allow the user to scroll through individual *Rays* making up a SEGMENT or L-FRAGMENT bundle from within the shotline data window without having to physically select those *Rays* with the mouse in the **Display Window**. The Visualizer colors the selected ray blue and updates the **Fragment Information** and **Shotline Information** sections with each click of an arrow.

3.2 2D PLOTS

Selecting the **2D Plots** option under the **Tools** menu with the left mouse button will open a separate window. The **2D Plots** window is composed of three sections: the **Menu Bar**, the **Information Window**, and the **Plot Display**. Figure 21 illustrates the **2D Plots** window. The only option currently available in the **File** menu is **Close**. Selecting this option simply closes the **2D Plots** window. Likewise, the **View** menu only contains a single option: **Non-Crits**. Toggling this option *on* (default) or *off* determines whether or not information for non-critical components is displayed in the risk angle plots. A non-critical component is defined as a component that is not assigned a PK table in the JTYPE input file (for additional information regarding critical and non-critical components, see Reference 1, Section 5.2). The **Information Window** has two folders displayed: **Probability Plots** and **Risk Angles**, each of which is discussed separately in the sections that follow.

Menu Bar

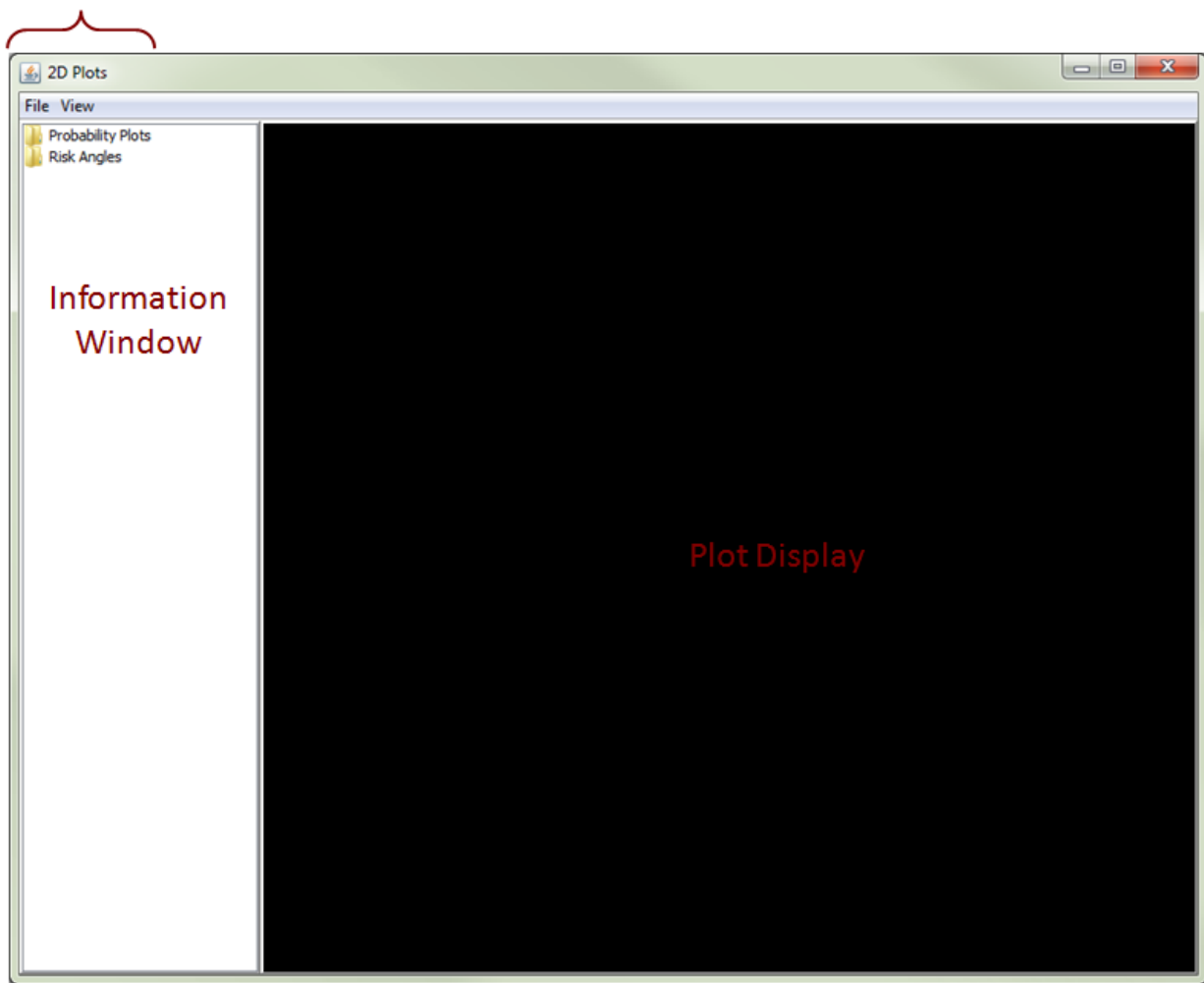


Figure 21. **2D Plots** Window.

3.2.1 Probability Plots

Double-clicking on the **Probability Plots** folder with the left mouse button will drop down a menu listing the *Events* in the “.ucf” file. Selecting any of the *Events* listed with the left mouse button will drop down another menu with the **Hazard Summary** and **Averages** options. Figure 22 illustrates these options.

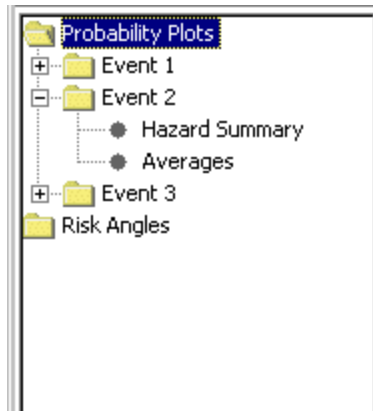


Figure 22. **Probability Plots** Options.

Double-clicking on the **Hazard Summary** option with the left mouse button will open a Hazard Probability Summary window within the **Plot Display** (see Figure 23). This graphic display shows the aircraft hazard probabilities for all flight phases assessed at each release angle of the *Event* selected. If the number of release angles for any given *Event* is larger than can be displayed on a single page, the Next and Previous buttons located in the bottom corners of the display window can be used to scroll through the pages. The page number will always appear under the title at the top of the display.

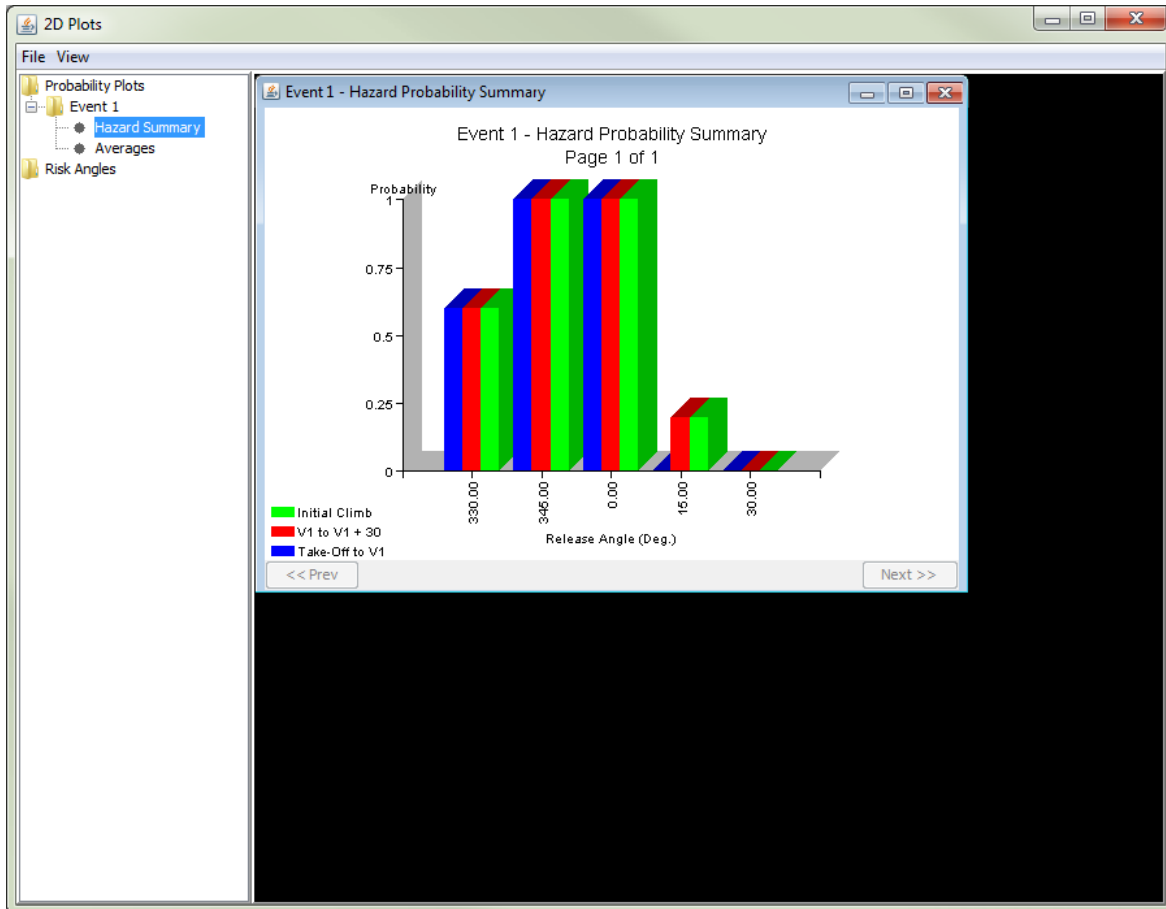


Figure 23. Hazard Summary Option.

Double-clicking on the **Averages** option with the left mouse button opens an Average Hazard Summary window within the **Plot Display** (see Figure 24). The data presented includes two sets of averages computed by UEDDAM: the blue columns represent the aircraft hazard probabilities at each flight phase assessed averaged over all *Release Points* analyzed; the red columns represent the aircraft hazard probabilities at each flight phase assessed averaged over the entire realm of possible release points (i.e., 360°). Note that the average over 360° assumes that the aircraft hazard probability for any *Release Point* not assessed is equal to 0.

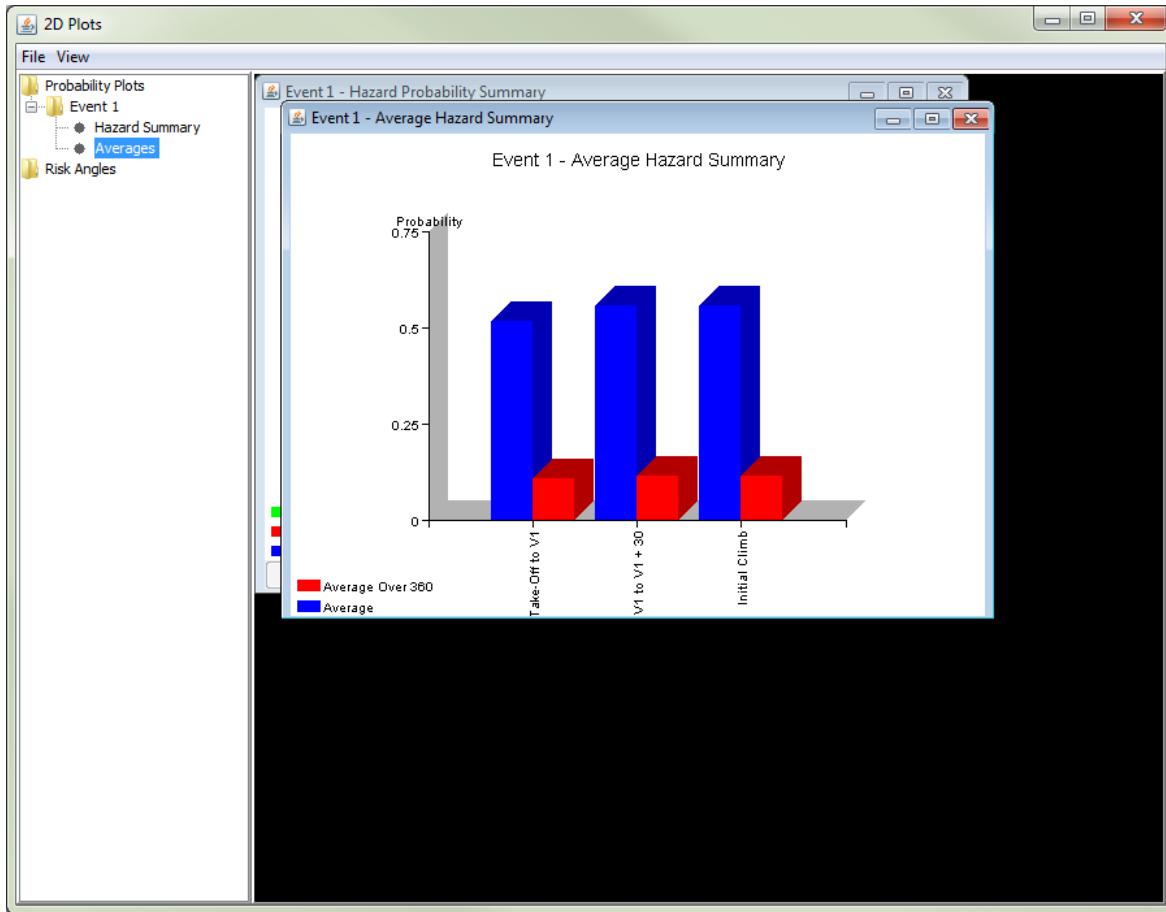


Figure 24. Averages Option.

Selecting any of the blue columns in the Average Hazard Summary window with the right mouse button will open yet another graphic display. This plot represents the probability of hazard at the selected flight phase for each *Release Angle* assessed and the average over all release angles assessed. Figure 25 illustrates the Flight Phase by Release Angle window.

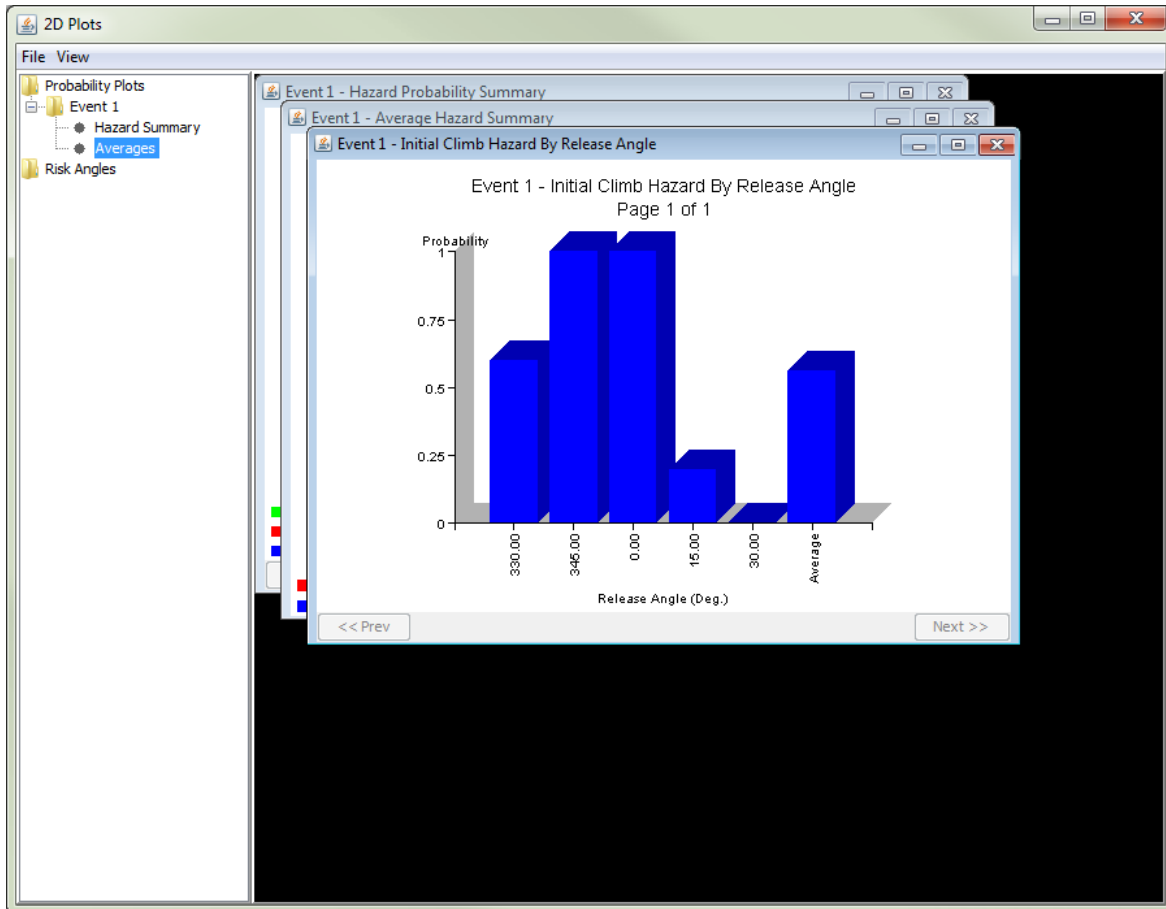


Figure 25. Flight Phase by Release Angle Display.

If the number of release angles for any given *Event* is larger than can be displayed on a single page, the Next and Previous buttons located in the bottom corners of the display window can be used to scroll through the pages.

Selecting any of the blue columns on the Flight Phase by Release Angle plot with the right mouse button will open a Damage Release Contribution graphic display for the particular *Release Angle* selected. This new plot breaks down the data one step further by displaying the probability of hazard of those components and/or systems that contribute to the overall aircraft probability of hazard (also displayed on the graph). This plot is for a specific flight phase and *Release Angle* and interpretation of the data requires knowledge of both COVART and probability theory. Figure 26 illustrates the Damage Release Contribution display.

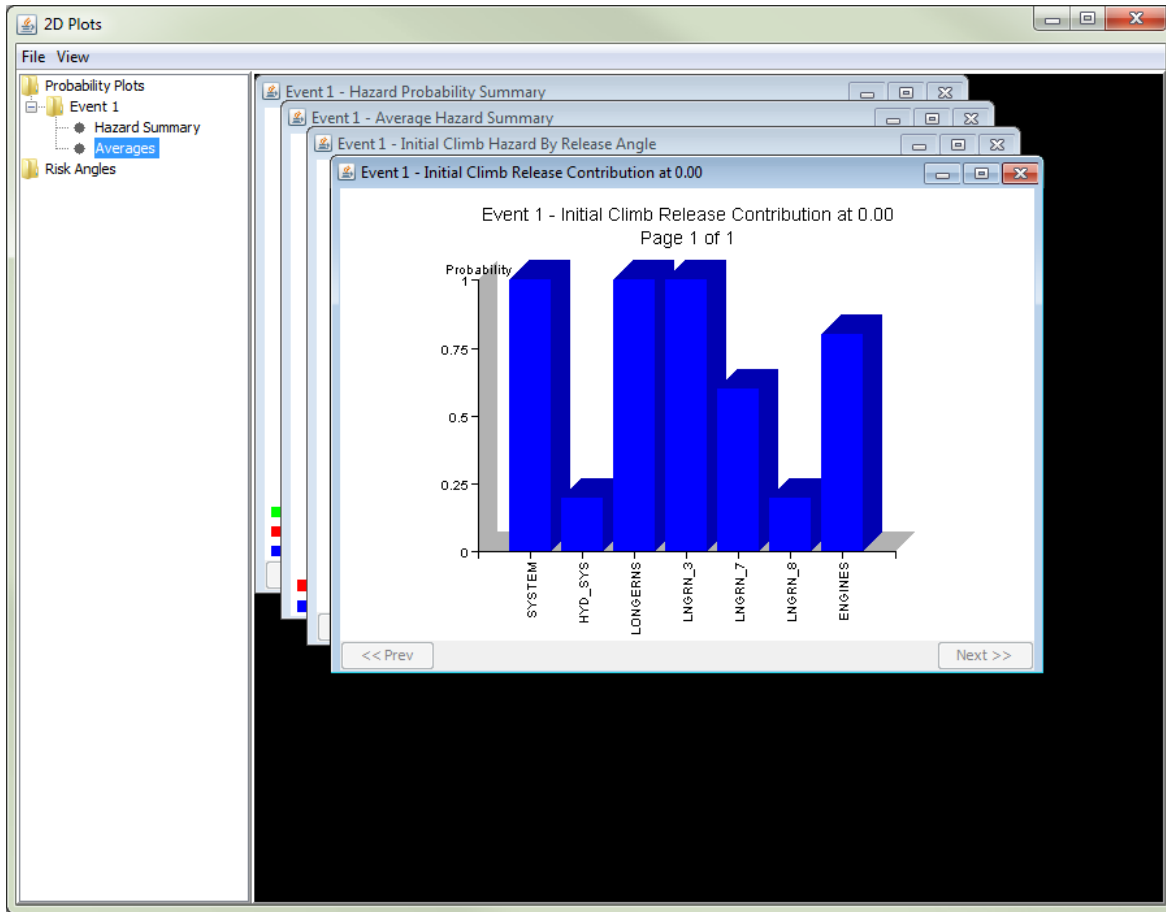


Figure 26. Damage Release Contribution Display.

3.2.2 Risk Angles

Similar to the **Probability Plots** option, double-clicking on the **Risk Angles** folder in the **Information Window** of the **2D Plots** window with the left mouse button will drop down a menu listing the *Events* in the “.ucf” file. Selecting any of the *Events* listed with the left mouse button will drop another menu with the **Translational** and **Fore/Aft** options. Figure 27 illustrates these options.

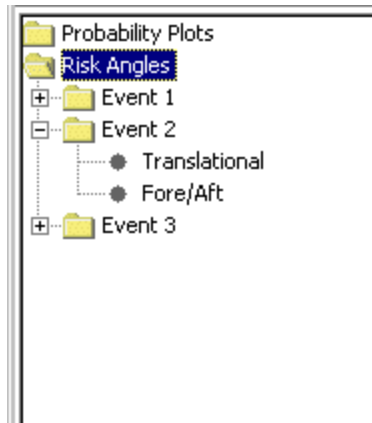


Figure 27. Risk Angles Option.

Double-clicking the **Translational** option with the left mouse button will bring up a plot in the **Plot Display** equivalent to those described in the FAA AC 20-128A (Reference 2). Figure 28 illustrates the resulting Translational Risk Angle bar graph.

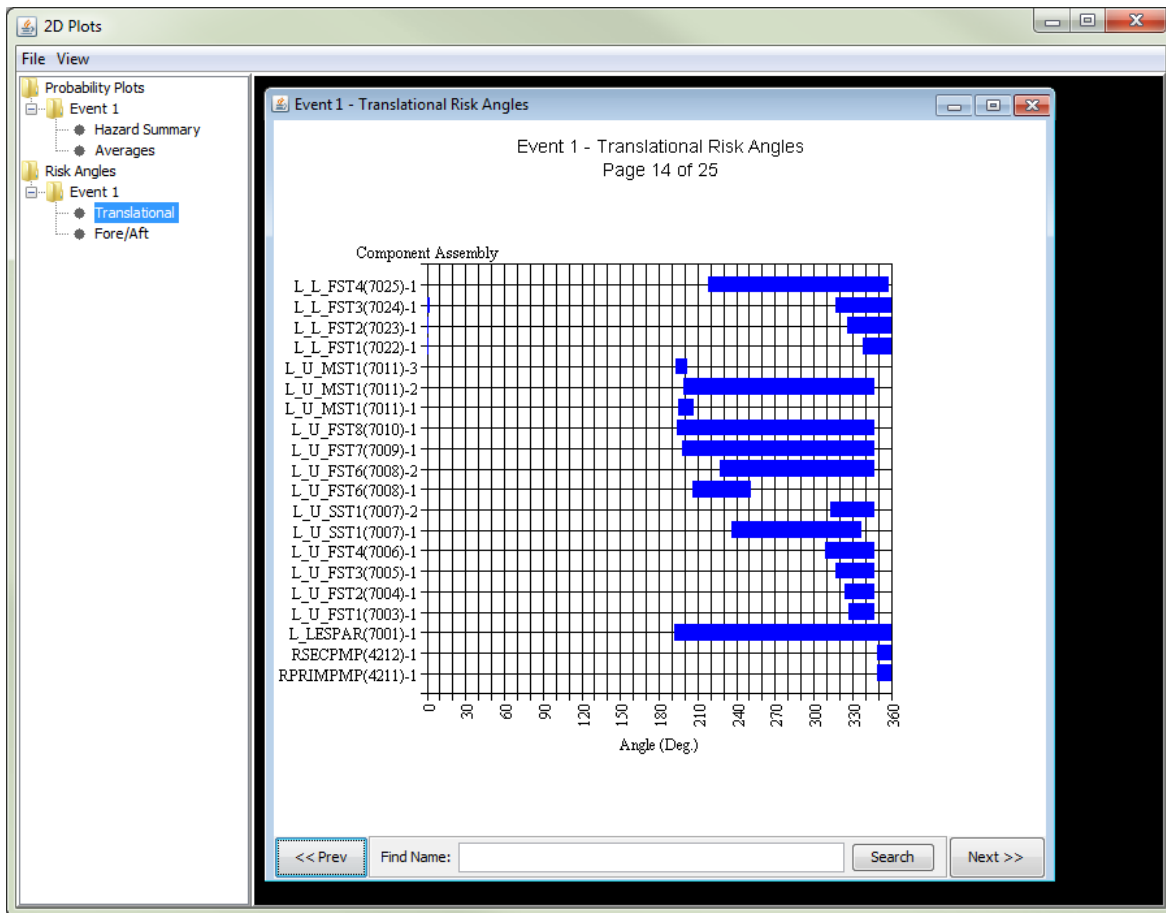


Figure 28. Translational Option.

Along the x-axis are the translational release angles. The y-axis contains a list of components identified by their COVART name followed by the associated component identification number in parentheses. The dash number is the assembly ID for that component (see Reference 1 for additional information regarding component assemblies). The bar for each component runs from the minimum translational release angle that will intersect that component to the maximum; collectively, these angles are referred to as the component's translational risk angles.

Because the number of components will be larger than can be displayed on a single page, the Next and Previous buttons are used to navigate between pages. At the bottom of the Translational Risk Angle window, there is also a Search field and button. By entering text in the Search field, the user can skip directly to a component that matches the search text. Clicking on the Search button again will find the next component matching the search criteria and so on. Upon reaching the end of the list during a search, a window stating that the search has gone to the end of the list will appear. Note that the Search field is case sensitive.

Similarly, double-clicking the **Fore/Aft** option with the left mouse button will bring up a Fore/Aft Risk Angle plot in the **Plot Display** (see Figure 29). The only difference between the Fore/Aft Risk Angle plot and the Translational Risk Angle plot are the angles represented along the x-axis.

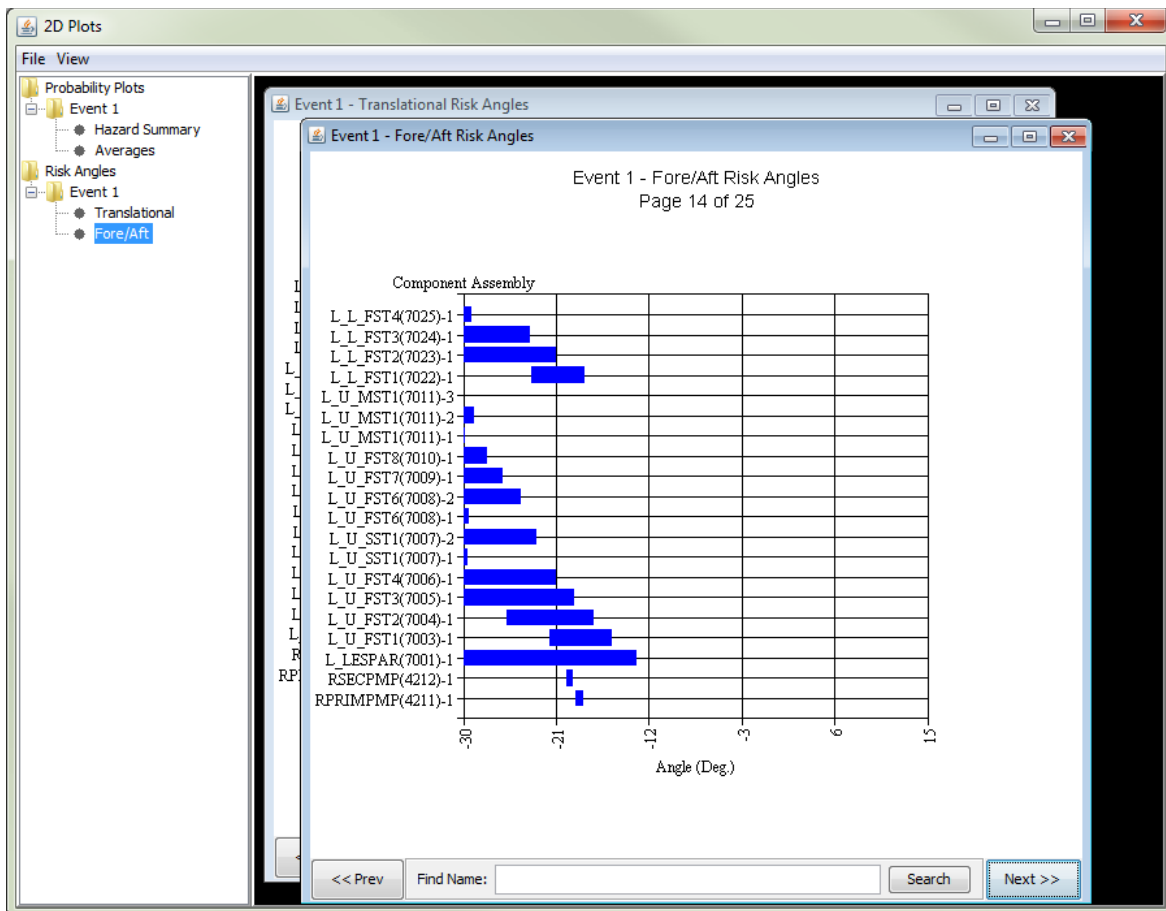
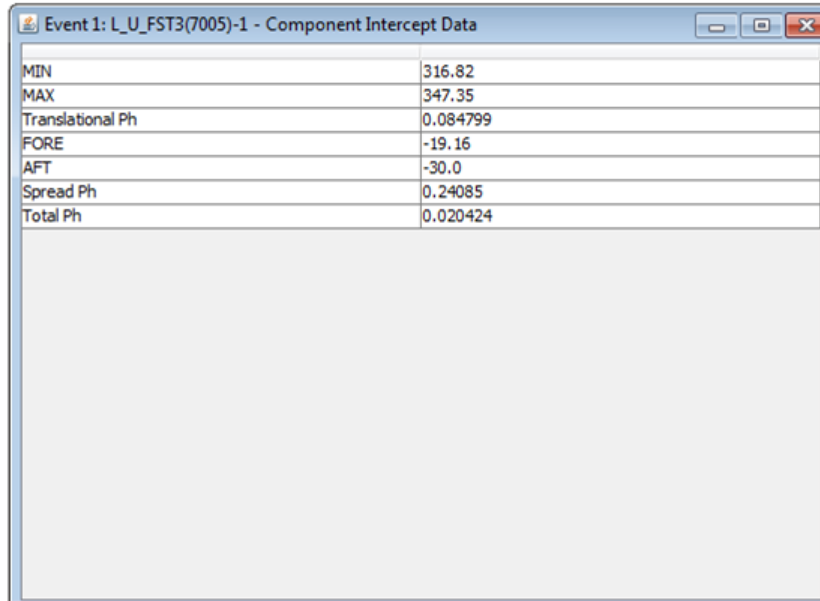


Figure 29. Fore/Aft Option.

Selecting any one of the blue bars on either the Fore/Aft Risk Angle plot or the Translational Risk Angle plot with the right mouse button will open a Component Intercept Data window (see Figure 30). The window contains a summary of the translational and spread risk angles and associated probability of hits for the particular component selected.



Event 1: L_U_FST3(7005)-1 - Component Intercept Data	
MIN	316.82
MAX	347.35
Translational Ph	0.084799
FORE	-19.16
AFT	-30.0
Spread Ph	0.24085
Total Ph	0.020424

Figure 30. Component Intercept Window.

3.3 SIGNIFICANT ITERATIONS

Selecting the **Significant Iterations** option under the **Tools** menu with the left mouse button will bring up a secondary window in which the SIGIT* file(s) output from a successful UEDDAM execution are displayed. If more than one SIGIT* file was requested in the UEDDAM control file, each output file will be displayed under a separate tab within the **Significant Iterations Window** (see Figure 31). To close the secondary window and return to the main UEDDAM Visualizer Display, the user should select the **File – Close** option from the menu bar of the **Significant Iterations Window**.

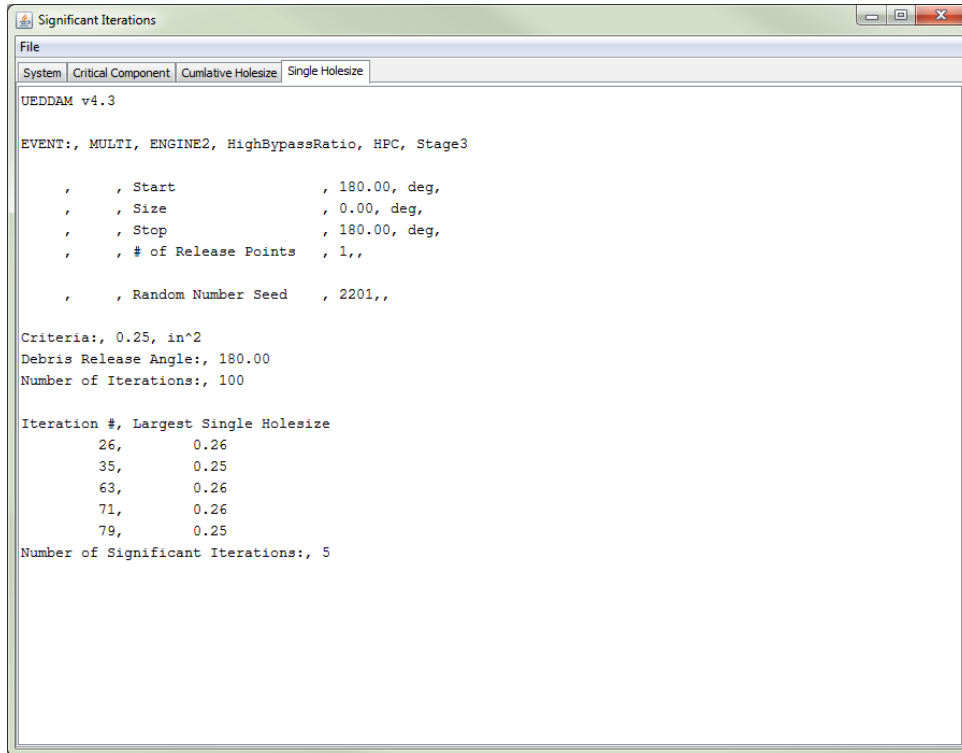


Figure 31. **Significant Iterations** Option.

3.4 RISK ZONE

Another option under the **Tools** menu is **Risk Zone**. When this option is selected, the Visualizer opens a secondary window displaying the geometry as is currently displayed in the **Display Window**, except with the following modifications: the view is set to 90° azimuth, 90° elevation and the geometry is Autosized and Autocentered in the **Risk Zone Window**. Note that the geometry displayed in the **Risk Zone Window** cannot be manipulated. In addition to the geometry, a series of **Event Buttons**, each representing a single *Event* specified in the UEDDAM control file, appear along the left-hand side of the **Risk Zone Window**. When one of the **Event Buttons** is pressed, the Visualizer draws four lines through the geometry (two port and two starboard) indicating the maximum forward and aft spread angles for that specific *Event*. Components of the geometry that lie between these lines are at risk of being hit by the debris associated with this *Event*. Figure 32 illustrates the **Risk Zone** option, with 4 of the possible 18 *Events* selected for display. To close the secondary window and return to the main UEDDAM Visualizer Display, the user should select the **File – Close** option from the menu bar of the **Risk Zone Window**.

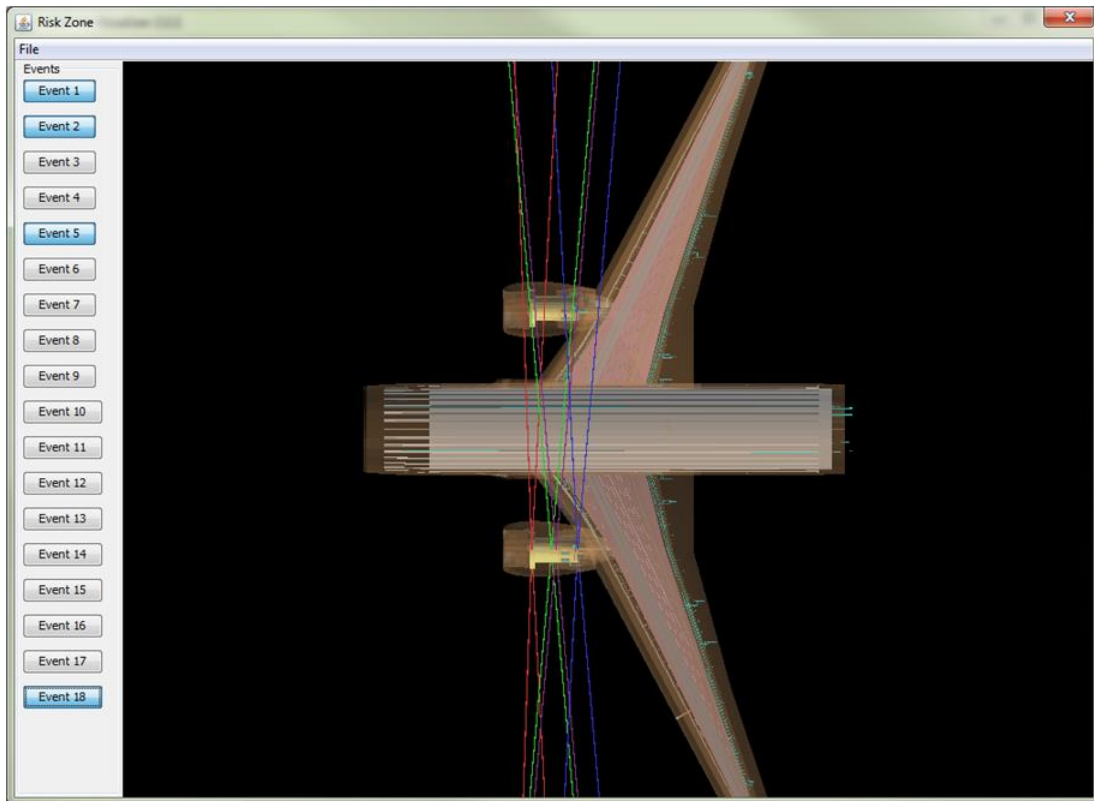


Figure 32. Risk Zone Tool.

3.5 FIND

The **Find** option (located under the **Tools** menu) allows the user to graphically locate a component (or group of components) of the geometry through a COVART name. When the **Find** tool is selected, the Visualizer opens a **Search** window. In the search field, the user can type either the complete or partial text of the COVART name of interest (the wild card character, used in the case of a partial name, is the *). Note that the user can perform a case-sensitive search using the toggle switch located in the **Search** window. The Visualizer then attempts to locate a component (or components) that match the text entered. In the case of a partial field search resulting in a match with more than one COVART name, a **List Box** will appear allowing the user to select the particular name of interest. Once a selection has been made, the component (or group of components) corresponding to the selected COVART name will be changed to solid, non-transparent, and highlighted in red in the **Display Window**; all other components of the geometry currently displayed will be made transparent. This process is illustrated in Figure 33.

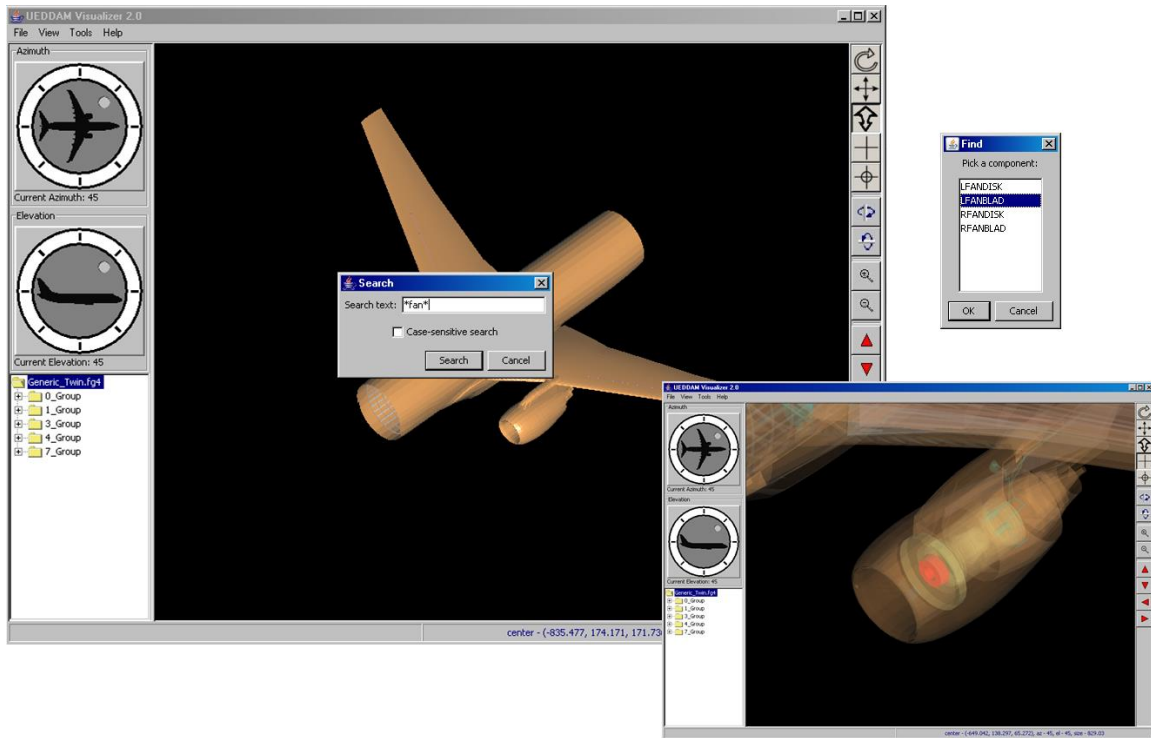


Figure 33. Find Tool.

4. REFERENCES

- 1) Phifer, Ellen M. and Doug R. Howard, *Uncontained Engine Debris Damage Assessment Model (UEDDAM) Version 5.1 User and Analyst Manual*, SURVICE-TR-0872, SURVICE Engineering Company, Belcamp, Maryland, May 2018.
- 2) *Design Considerations for Minimizing Hazards Caused by Uncontained Turbine Engine and Auxiliary Power Unit Rotor*, AC 20-128A, U.S. Department of Transportation, Federal Aviation Administration, 25 March 1997.