UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DEL DISEÑO

MÁSTER UNIVERSITARIO EN INGENIERÍA AERONÁUTICA

TRABAJO FIN DE MÁSTER

ANÁLISIS DE MÉTODOS DE INTEGRACIÓN NUMÉRICA PARA PROBLEMAS DINÁMICOS. IMPLEMENTACIÓN Y VALIDACIÓN MEDIANTE UNA APLICACIÓN DE MATLAB

AUTOR: Iván Navarro Crespo

TUTOR: Mario Lázaro Navarro

Valencia, Diciembre 2020

A mi familia
A María
Por su apoyo incondicional
Y a ti, abuelo

Resumen

Los métodos de integración numérica son una herramienta potente para la resolución de problemas que involucren ecuaciones diferenciales, en el caso de este trabajo la ecuación de la dinámica estructural. Se va a hacer una revisión y análisis de algunos de los métodos más conocidos, como el método de Newmark-β, y algún método más reciente como es el método de Bathe modificado. Mediante el análisis de un caso práctico, aplicando los distintos métodos, se verán algunas de las ventajas e inconvenientes asociados a los mismos. Se diseña un ejemplo de aplicación en el ala de un avión con múltiples grados de libertad la cual se modelará mediante una viga siguiendo la teoría de vigas de Euler-Bernoulli. Se observará el comportamiento que tiene en función de cual sea el método utilizado y las fuerzas empleadas.

Para recopilar todos los métodos usados, se ha creado una apliación en el entorno de Matlab mediante la interfaz de App Designer en donde se podrán resolver problemas dinámicos genéricos con cada uno de los métodos utilizados y distintos tipos de fuerzas y validar métodos nuevos de forma simple. Los datos generados en desplazamientos, velocidades y aceleraciones podrán conservarse para su postproceso. Se concluye que el uso de un método u otro depende de las características del problema dinámico a resolver y los recursos disponibles por el usuario. Respecto a la aplicación se proponen mejoras que ayuden al usuario a un mejor entendimiento de la misma y la inclusión de nuevas herramientas que permitan contrastar resultados de forma más efectiva.

Palabras clave: estructura, ecuación dinámica, metodo de integración, integración numérica, Newmark-β, Bathe, App Designer, Matlab, ala.

Resum

Els mètodes d'integració numèrica són una ferramenta potent per a la resolució de problemes que involucren equacions diferencials, en el cas d'aquest treball l'equació de la dinàmica estructural. Es farà una revisió i anàlisi d'alguns dels mètodes més coneguts, com el mètode de Newmark-β, i algun mètode més recent com és el mètode de Bathe modificat. Mitjançant l'anàlisi d'un cas pràctic, aplicant els diferents mètodes, es veuran alguns dels avantatges e inconvenients associats a aquests. Es dissenya un exemple d'aplicació en l'ala d'un avió amb múltiples graus de llibertat la qual es modelarà mitjançant una biga seguint la teoria de bigues de Euler-Bernoulli. S'observarà el comportament que té en funció de qual siga el mètode utilitzat i les forces emprades.

Per a recopilar tots els mètodes usats, s'ha creat una apliació a l'entorn de Matlab mitjançant la interfície d'App Designer on es podran resoldre problemes dinàmics genèrics amb cadascun dels mètodes utilitzats i diferents tipus de forces i validar mètodes nous de manera simple. Les dades generades en desplaçaments, velocitats i acceleracions podran conservar-se per a la seua postprocés. Es conclou que l'ús d'un mètode o un altre depén de les característiques del problema dinàmic a resoldre i els recursos disponibles per l'usuari. Amb respecte a l'aplicació es proposen millores que ajuden a l'usuari a un millor enteniment de la mateixa i la inclusió de noves ferramentas que permeten contrastar resultats de forma més efectiva.

Paraules clau: estructura, equació dinàmica, mètode d'integració, integració numèrica, Newmark-β, Bathe, App Designer, Matlab, Ala.

Abstract

Numerical integration methods are a powerful tool for solving problems involving differential equations, in the case of this master thesis the equation of structural dynamics. It will be done a review and analysis of some of the best known methods, such as the Newmark-β method, and some more recent methods such as the modified Bathe method. By means of the analysis of a practical case, applying the different methods, some of the advantages and disadvantages associated to them will be seen. An example of application is designed in the wing of an aircraft with multiple degrees of freedom which will be modeled by means of a beam following Euler-Bernoulli's beam theory. The behavior of the beam will be observed as a function of the method used and the forces employed.

To collect all the methods used, an application has been created in the Matlab environment through the App Designer interface where you can solve generic dynamic problems with each of the methods used and different types of forces and validate new methods in a simple way. The data generated in displacements, velocities and accelerations can be kept for post-processing. It is concluded that the use of one method or another depends on the characteristics of the dynamic problem to be solved and the resources available to the user. With respect to the application, improvements are proposed that will help the user to better understand it and the inclusion of new tools that will allow to contrast results in a more effective way.

Key words: structure, dynamic equation, integration method, numerical integration, Newmark-β, Bathe, App Designer, Matlab, wing.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Desde que las capacidades de computación han ido aumentando a lo largo de los años, distintos métodos se han desarrollado con el fin de resolver problemas que carecían de solución analítica y gracias a su desarrollo, se ha conseguido aproximar su solución. El objetivo de estos métodos no es sólamente resolver las ecuaciones, si no también proveer eficiencia computacional. Esto es así ya que la capacidad computacional de un ordenador, a día de hoy, no es infinita y la memoria que puede requerir un método de este tipo tiene que adecuarse a la potencia computacional para que pueda resolverse. Es por eso por lo que se buscan métodos programables cuyo uso de la memoria sea lo más reducido posible, así como el tiempo de computación requerido para obtener la solución.

Los métodos de integración numérica en el ámbito de la ingeniería tienen multitud de aplicaciones, desde el estudio del efecto de los vientos alrededor de un edificio, el cálculo del área de la sección transversal de un río hasta problemas de transmisión de calor. En el caso de este trabajo, se van a utilizar métodos de integración numérica en el tiempo para la resolución de problemas transitorios asociados a la dinámica estructural.

1.1. Método de elementos finitos

El análisis mediante elementos finitos para la resolución de problemas dinámicos con cargas variables (y constantes) en el tiempo es en el que se va a centrar este trabajo. El método de elementos finitos es uno de los procedimientos más usados en análisis estructural. Esto es así ya que su uso se puede extender a multitud de aplicaciones en el campo ingenieril. El método de los elementos finitos se implementa en programas para realizar el análisis del problema. Requieren de una determinada capacidad computacional, como ya se ha hablado anteriormente.

El objetivo de estos programas es ser capaz de resolver problemas cuya solución análitica es inexistente o muy complicada usando modelos que pueden ser o no complejos. Un aspecto muy importante de este método es que los resultados que se pueden obtener sólamente son los que aparecen en su propio desarrollo, no puede predecir variables que no estén involucradas en su procedimiento (al menos con precisión).

El método de elementos finitos en un primer momento se diseñó para resolver problemas de la mecánica estructural, pero se dieron cuenta que la misma filosofía podía aplicarse en otros campos. El método de elementos finitos al ser un procedimiento de resolución numérica no exacta, requiere que sus resultados sean posteriormente analizados con el fin de alcanzar unos requerimientos de precisión. Si estos requerimientos no son obtenidos entonces habrá que modificar los distintos parámetros del método con el fin de ajustar la precisión. Los métodos de integración numérica temporal también requieren estas comprobaciones con la diferencia de que, en este caso, el parámetro de control será el paso temporal utilizado como se verá a lo largo de este trabajo.

La base del método de elementos finitos es muy similar, como se verá, a la de la integración numérica temporal. Se trata de discretizar espacialmente una estructura en un conjunto de elementos donde se promedia la solución

mediante los cálculos que proporciona el método. La discretización espacial se consigue mediante la técnica de mallado, la cual no es trivial y en muchas ocasiones tiene que ejecutarse conforme al modelo que se quiere simular. Las técnicas de mallado no serán objetivo de este trabajo. Mediante el método de los elementos finitos [3], aplicado a la mecánica estructural, se llega a una solución del tipo:

$$\mathbf{KU} = \mathbf{f} \tag{1.1}$$

Donde **K** es la matriz de rigidez de la estructura, **U** es el vector que contiene la solución en desplazamientos y **f** es el vector que contiene las fuerzas aplicadas a la estructura . Esta ecuación se la conoce como la ecuación de equilibrio. Dentro del vector de fuerzas se pueden encontrar fuerzas dependientes del tiempo, que ocasiona que los desplazamientos también sean variables con el tiempo y por tanto la ecuación (1.1) proporciona los valores de equilibrio para un instante del tiempo determinado. Sin embargo, en algunas situaciones las cargas se pueden aplicar de forma rápida o pueden ser variables en el tiempo (**f(t)**). Por ello, las fuerzas de inercia han de ser consideradas y aparece el problema dinámico el cual va a ser protagonista en este trabajo. En la ecuación de equilibrio estático las fuerzas de inercia estaban incluidas en el vector de fuerzas **f**, ahora se incluyen como parte de las fuerzas sobre el cuerpo obteniendo la ecuación de equilibrio dinámico:

$$\mathbf{M\ddot{U}} + \mathbf{KU} = \mathbf{f} \tag{1.2}$$

Esta ecuación se obtiene fácilmente de la segunda ley de Newton aplicado a un sistema masa - muelle:

$$\frac{d}{dt}(m\frac{du}{dt}) = \sum p(t) \tag{1.3}$$

Donde el sumatorio de fuerzas *p(t)* engloba la carga aplicada y la fuerza resistente que ejerce el muelle por la tendencia natural del sistema a recuperar la posición de equilibrio.

$$\mathbf{M\ddot{U}} = \mathbf{f(t)} - \mathbf{KU} \tag{1.4}$$

Donde la ecuación (1.4) es la ecuación de equilibrio dinámico y **M** es la matriz de masas del sistema. Además, de la experiencia, se suele incluir un término disipativo debido a la evidencia de la dispación de energía durante la vibración. Este se incluye en la ecuación teniendo en cuenta que es dependiente de las velocidades de amortiguamiento. Con ello se obtiene la ecuación dinámica en equilibrio completa:

$$\mathbf{M\ddot{U}} + \mathbf{C\dot{U}} + \mathbf{KU} = \mathbf{f} \tag{1.5}$$

Donde la matriz **C** representa la matriz de amortiguamiento. Esta matriz no se suele construir a partir del modelo estructural usado, si no que se suele representar en función de las matrices de rigidez y de masas. En este trabajo se verá una de las formas de modelizar la matriz de amortiguamiento.

Una vez ha sido planteado el sistema de ecuaciones que ha de ser resuelto, se necesitará un método de integración temporal que cumpla los requisitos antes mecionados para poder predecir los desplazamientos asociados a los grados de libertad del modelo estructural.

1.2. Concepto de integración temporal

La integración temporal es una herramienta matemática utilizada para resolver problemas en el dominio del tiempo cuya resolución analítica no es posible y esto hace necesario su aproximación numérica para obtener la solución. El funcionamiento de la discretización espacial se puede observar en la figura 1.1.

Se trata de dividir el espacio temporal en pasos temporales más pequeños y dentro de cada paso temporal se resuelven las ecuaciones con el método de integración temporal correspondiente.

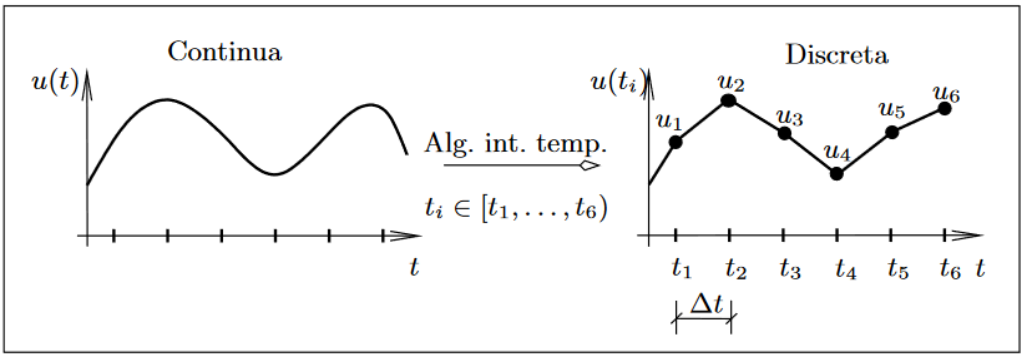

Figura 1.1: Discretización temporal.

1.3. Historia de los métodos de integración temporal en problemas estructurales

Los métodos de integración numérica existen desde hace muchos años. Los algoritmos numéricos son, al menos, tan antiguos como el papiro del Rhind egipcio (c. 1650 a.c.), que describe un método de búsqueda de raíces para resolver una ecuación simple. Además, los antiguos matemáticos griegos hicieron muchos más avances en los métodos numéricos. De cualquier modo, los algoritmos basados en la resolución de ecuaciones diferenciales no aparecieron hasta más tarde, en los principios de la Edad Moderna.

El método de las diferencias finitas fueron introducidas por Brook Taylor en 1715 y también han sido estudiadas como objetos matemáticos abstractos y autónomos en los trabajos de George Boole (1860), L. M. Milne-Thomson (1933) y Károly Jordan (1939). Las diferencias finitas se remontan a uno de los algoritmos de Jost Bürgi (c. 1592) y a trabajos de otros, como Isaac Newton. El cálculo formal de las diferencias finitas puede ser visto como una alternativa al cálculo de los infinitesimales. Constituyó uno de los primeros avances en la consecución de métodos de integración precisos y eficientes.

En el año 1768, Leonard Euler, publicó en su libro *Institutionum calculi integralis*, el famoso método de Euler para resolución de ecuaciones diferenciales ordinarias y que sentaría un precedente para los futuros métodos de

integración numérica. Muchos de los métodos que han sido creados posteriormente tienen su base teórica en el método que ideó Euler. De hecho, el método de Runge-Kutta de primer orden que llegó más tarde es el mismo que el método de Euler.

Entre finales del siglo XIX y principios del siglo XX, en 1895, Runge propuso una generalización del esquema de Euler de manera que en un mismo paso temporal se permitían varias evaluaciones de la derivada obteniendo una mayor precisión del resultado.

Posteriormente, Heun (1900) y Kutta (1901) continuaron el trabajo de Runge con algunas aportaciones como asentar una base para estos métodos y desarrollar métodos prácticos para incrementar la eficiencia y la precisión. Fue remarcable el trabajo de Nyström (1925), el cual corrigió los métodos de Kutta de quinto orden que contenían errores y estaban incompletos. Con el trabajo de Nyström terminaba la primera fase de los métodos de Runge-Kutta y además, se extendió el uso de de los métodos de Runge-Kutta para la resolución de sistemas de ecuaciones diferenciales de segundo orden. Estos sistemas solían pertenecer a los problemas dinámicos como los que se van a resolver en este trabajo.

Los algoritmos enfocados a la resolución de problemas mediante un método de integración numérico en el tiempo para estructuras, surgieron en los años 50, años en los que la capacidad computacional comenzó a emerger con el nacimiento de la informática. Esto permitió el desarrollo de métodos más complejos que antes no tenían oportunidad de aparecer sin ayuda de la tecnología.

Concretamente en 1959, apareció el primer método orientado a resolución de problemas dinámicos estructurales con el desarrollo de los algoritmos de Newmark [4]. Este algoritmo era capaz de resolver problemas estructurales de cualquier grado de complicación con cualquier relación entre la fuerza y el desplazamiento, desde un problema lineal hasta uno no lineal. Además,

permitía la introducción de cualquier fuerza dinámica. Cuando la carga aplicada es adecuada, con una introducción de amortiguamiento suficiente para prevenir el movimiento oscilatorio indefinido, el procedimiento podía ser usado para determinar el comportamiento estático de una estructura hasta el colapso. También permitía la resolución del problema dinámico cuando las condiciones de contorno son dependientes del tiempo.

Los métodos de Houbolt surgieron en los años 50, como el método de Newmark-β, y fueron los primeros en introducir control de la disipación numérica para evitar las oscilaciones espurias creadas al no resolver los modos asociados a altas frecuencias. Este algotritmo se introdujo en su momento en muchos de los programas de elementos finitos debido a sus propiedades de supresión asintótica de la solución en altas frecuencias. Este método está basado principalmente en el método de las diferencias finitas presentado anteriormente.

En el año 1968 Wilson estableció el método *factor θ de Wilson*, parecido a la familia de métodos Newmark pero con la diferencia de la manera de aproximación de la solución y de los parámetros utilizados en el método.

En el año 1977, Hughes, Hilber y Tayler crean el método HHT-α [5] cuyas inciales le dan nombre y se presenta como un método incondicionalmente estable que presenta mejores características de amortiguamiento de la respuesta que los anteriores.

En 1993 se crea el método α-generalizado por J.Chung y Hulbert [6]. Este método surge como una corrección de los métodos que había con la misma estructura en cuanto al control disipación numérica, por la cual eran criticados.

Más adelante, han ido apareciendo nuevos métodos para resolver problemas estructurales Dinámicos como el método de Bathe (2005) y el método de Bathe modificado.

1.4. Objetivos y motivación

La ejecución de este trabajo de fin de máster nace de la necesidad de automatizar un proceso, en este caso la resolución del problema dinámico, y ser capaz de mostrar los resultados asociados a las distintas formas de resolución del problema. De la experiencia como alumno, la teoría de vibraciones siempre ha resultado difícil de comprender y con este trabajo también se quiere exponer de una forma sencilla la resolución numérica de estos problemas.

Se plantea la pregunta de qué metodos de todos los expuestos son los más adecuados para analizar la respuesta del problema. Ante esta pregunta, el trabajo tiene como objetivo, en primer lugar, realizar un análisis de los métodos. Este análisis se va a extender tanto a la precisión del método respecto de la solución exacta como a la estabilidad del mismo en función de los parámetros que puede ser variados. Además, una vez hayan sido analizados y se hayan visto sus ventajas e inconvenientes, se establecerán cuales serían sus campos de aplicación óptimos.

Después, se van a emplear en un caso de aplicación aeronáutico como es el caso de un ala que actúa como una viga empotrada. Se generará el modelo y, posteriormente, se calcularán los desplazamientos de la misma. Es un caso de aplicación recurrente puesto que en el proceso de certificación de un avión se deben de certificar muchas partes y, entre ellas, el ala del avión se somete a distintas pruebas. Mediante este programa se puede aproximar la respuesta del ala ante distintas cargas de forma que el proceso de cerificación se pueda realizar de forma segura. Estos programas se diseñarán y ejecutarán mediante *Matlab*. Además se desarrollará la teoría del modelo de viga utilizado que en este trabajo será el modelo *Euler-Bernoulli*.

Por último, habiendo estudiado los distintos métodos, sus apliaciones y un ejemplo práctico, se ha decidido, haciendo uso de la funcionalidad *Appdesigner* de *Matlab*, crear una aplicación cuyo objetivo será ser capaz de mostrar las siguientes características:

- Poder introducir un modelo estructural cualquiera (matrices $\mathbf{M}$, $\mathbf{K}$ y el vector $\mathbf{f}$).
- Seleccionar el grado de libertad del cual se quiere obtener la respuesta transitoria.
- Poder seleccionar el método de integración temporal para la resolución del problema.
- En cuanto al vector de fuerzas $\mathbf{f}$, que sea capaz de simular cargas constantes, armónicas y puntuales.
- Que sea escalable, es decir, que sea fácil de modificar y de añadir nuevas funcionalidades para que esté más completo.
- Que permita modificar los parámetros de los métodos.
- Por último, que sea capaz de comparar el uso de distintos métodos de integración para un problema concreto con el objetivo ver sus comportamientos.

Esta aplicación podría tener un uso en el ámbito académico en relación al estudio del comportamiento de un sistema que esté afectado por las distintas cargas principales que se pueden encontrar (carga rápida tras suelta estática, carga escalón, carga de percusión y carga armónica).

Capítulo 2

Métodos de integración numérica temporal

Como se ha visto en el Capítulo 1, los métodos de integración numérica temporal se han ido a desarrollando a lo largo del tiempo y además se han realizado modificaciones de los métodos clásicos de forma que se han corregido alguna de las debilidades que presentaban.

El objetivo de los métodos de integración numérica es la aproximación de la solución real a base de interpolar la misma en un paso temporal que se denominará Δt. Un método numérico es un algoritmo que se utiliza para obtener valores numéricos como solución a un problema matemático.

Se llama modelo matemático a un conjunto de ecuaciones con datos e incógnitas que describen algún problema real como puede ser el movimiento de un edificio ante el viento o el movimiento de un péndulo. El análisis numérico es la disciplina que se ocupa de la descripción y análisis de los métodos numéricos. En el análisis numérico se analizan características como la precisión del método (error de la solución) y la estabilidad del método (convergente o divergente). En la figura 2.1 se observa un esquema de integración numérica mediante los conceptos explicados.

Los esquemas de integración puede clasificarse en:

- Métodos explícitos
- Métodos implícitos.

Los métodos explícitos de integración tienen la característica principal de que son condicionalmente estables, es decir, que existe un paso temporal que se denominará como paso temporal crítico (Δt_{crit}) por encima del cual el esquema será inestable. Los métodos explícitos se diferencian de los implícitos porque no necesitan información del paso temporal $\Delta t + t$ para predecir la respuesta. En los métodos explícitos, la presencia en un sistema de constantes de tiempo pequeñas obliga a emplear pasos de integración pequeños para preservar la estabilidad de la integración numérica, mientras que la presencia de constantes de tiempo grandes obliga a simular periodos de tiempo largos para observar la respuesta del sistema. La presencia simultánea de constantes de tiempo pequeñas y grandes conduce a sistemas matemáticamente rígidos, que consumen grandes recursos de computación. Por lo que hay una limitación a la hora de representar simultáneamente fenómenos rápidos y lentos, aunque computacionalmente son métodos más sencillos.

Los métodos implícitos sí que utilizan la información del paso temporal $\Delta t + t$ para precedir la respuesta pero en este caso pueden ser incondicionalmente o condicionalmente estables. Surgen para resolver el problema que surgía, en los esquemas explícitos, de rigidez.

Los métodos que se van a estudiar en este trabajo son la familia de métodos Newmark-β, el método de Bathe, el método de Runge-Kutta aplicado a problemas estructurales, el conocido como *Precise Integration Method* (PIM) aplicado a problemas estructurales y por último una modificación reciente del método de Bathe que permite modificar más parámetros que en el método clásico con las consecuencias que se verán.

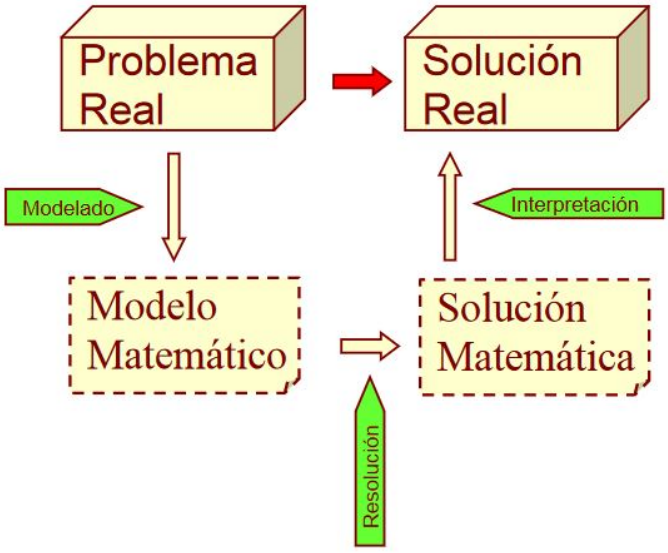

Figura 2.1: Modelo de integración numérica

2.1. Método de superposición modal

Los métodos de integración directa que se van a ver en los siguientes apartados involucran la resolución de numerosas ecuaciones que, además, tienen una dependecia directa con el tamaño de la matriz de rigidez que está asociada al problema.

Esto implica que los métodos de integración directa son muy útiles cuando la cantidad de pasos temporales a resolver es pequeña y, por lo tanto, son más apropiados de utilizar cuando se quiere analizar un instante pequeño de tiempo. Cuando ocurre que el tiempo que se quiere resolver es mayor, es conveniente transformar las ecuaciones de manera que la solución sea menos costosa operacionalmente. Una de las maneras para de alcanzar este objetivo es plantear que, si el coste de la solución es directamente proporcional al tamaño de la matriz de rigidez, si reducimos su tamaño, entonces el coste operacional decrecerá proporcionalmente.

Lo que se propone para lograr este objetivo es la transformación del vector de desplazamientos que contiene los desplazamientos en los nodos de la

siguiente forma:

$$\mathbf{U}(t) = \Phi\ \mathbf{X}(t) \tag{2.1}$$

Donde Φ representa la matriz modal que es de tamaño $n\ x\ n$ donde n es el número de grados de libertad del problema a resolver. Los términos del vector $\mathbf{X}(t)$ se denominarán deplazamientos generalizados y es de tamaño n *x 1*. Introduciendo la ecuación 2.1 en la ecuación de equilibrio dinámico con amortiguamiento se obtiene:

$$\tilde{\mathbf{M}}\ddot{\mathbf{X}}(t) + \tilde{\mathbf{C}}\dot{\mathbf{X}}(t) + \tilde{\mathbf{K}}\mathbf{X}(t) = \tilde{\mathbf{f}}(t) \tag{2.2}$$

Y las nuevas matrices $\tilde{\mathbf{M}}$, $\tilde{\mathbf{C}}$ y $\tilde{\mathbf{K}}$, vienen dadas por las expresiones:

$$\tilde{\mathbf{M}} = \mathbf{\Phi}^T\mathbf{M}\mathbf{\Phi};\quad \tilde{\mathbf{C}} = \mathbf{\Phi}^T\mathbf{C}\mathbf{\Phi};\quad \tilde{\mathbf{K}} = \mathbf{\Phi}^T\mathbf{K}\mathbf{\Phi};\quad \tilde{\mathbf{f}} = \mathbf{\Phi}^T\mathbf{f} \tag{2.3}$$

Se va a continuar el método asumiendo que el amortiguamiento es despreciable. Se supone la ecuación de equilibrio dinámico sin amortiguamiento con una solución de U de la forma $\mathbf{U}(\mathbf{t}) = \mathbf{U}e^{i\omega t}$. Si se sustituye esta expresión en la ecuación de equilibrio en condición de vibraciones libres se obtiene:

$$\left(\mathbf{K} - \omega^2\mathbf{M}\right)\mathbf{U}\ e^{i\omega t} = 0 \tag{2.4}$$

Cuyas soluciones satisfacen,

$$det|\mathbf{K} - \omega^2\mathbf{M}| = 0 \tag{2.5}$$

de donde se obtienen los valores de frecuencias naturales no amortiguadas ω_n^2 que son los autovalores del sistema.

Asociada a cada autovalor, hay un autovector que se va a expresar como $(\boldsymbol{\phi})_n$, que queda totalmente determinado a falta de una constante y que cumple,

$$\mathbf{K}(\boldsymbol{\phi})_n = \omega_n^2\mathbf{M}(\boldsymbol{\phi})_n \tag{2.6}$$

El siguiente paso es normalizar el modo. Hay muchas maneras pero la más típica es hacer la normalización de los modos respecto de la matriz de masas, de manera que se cumpla:

$$(\boldsymbol{\phi})_n^T \mathbf{M} (\boldsymbol{\phi})_n = 1 \tag{2.7}$$

A estos modos se les llama modos normales y se puede demostrar que también cumplen,

$$(\boldsymbol{\phi})_r^T \mathbf{K} (\boldsymbol{\phi})_n = \omega_n^2 \tag{2.8}$$

Una de las propiedades más importantes de los modos naturales es la ortogonalidad. Esto es, premultiplicando por el modo $(\boldsymbol{\phi})_s^T$ en la ecuación 2.6,

$$(\boldsymbol{\phi})_s^T \mathbf{K} (\boldsymbol{\phi})_r - \omega_r^2 (\boldsymbol{\phi})_s^T \mathbf{M} (\boldsymbol{\phi})_r = 0 \tag{2.9}$$

y si se premultiplica por $(\phi)_s^T$,

$$(\boldsymbol{\phi})_r^T \mathbf{K} (\boldsymbol{\phi})_s - \omega_s^2 (\boldsymbol{\phi})_r^T \mathbf{M} (\boldsymbol{\phi})_s = 0 \tag{2.10}$$

Al ser las matrices $\mathbf{K}$ y $\mathbf{M}$ simétricas, la anterior ecuación se puede escribir como:

$$(\boldsymbol{\phi})_s^T \mathbf{K} (\boldsymbol{\phi})_r - \omega_s^2 (\boldsymbol{\phi})_s^T \mathbf{M} (\boldsymbol{\phi})_r = 0 \tag{2.11}$$

Y restando esta ecuación a la ecuación (2.9) se obtiene:

$$(\omega_s^2 - \omega_r^2)(\boldsymbol{\phi})_s^T \mathbf{M} (\boldsymbol{\phi})_r = 0 \tag{2.12}$$

En donde si se tienen distintas frecuencias se ha de cumplir,

$$(\boldsymbol{\phi})_s^T \mathbf{M} (\boldsymbol{\phi})_r = 0 \tag{2.13}$$

Si esta ecuación se cumple, se dice que los modos r y s son ortogonales respecto a la matriz de masas y tomando las ecuación 2.9 se obtiene que los modos r y s son ortogonales respecto a la matriz de rigidez.

Suponiendo que todos los modos son ortogonales, la matriz modal del sistema $\mathbf{\Phi}$ es una matriz de dimensión $n\ x\ n$ cuyas columnas son los modos normales:

$$\mathbf{\Phi} = [(\boldsymbol{\phi})_1, (\boldsymbol{\phi})_2, ..., (\boldsymbol{\phi})_n] \tag{2.14}$$

La matriz modal del sistema cumple las siguientes igualdades:

$$\Phi^T \mathbf{K} \Phi = \mathbf{\Omega}^2 \tag{2.15}$$

$$\Phi^T \mathbf{M} \Phi = \mathbf{I} \tag{2.16}$$

Donde $\mathbf{I}$ es la matriz identidad y $\mathbf{\Omega^2}$ es una matriz diagonal que contiene las frecuencias naturales del sistema.

En el anterior caso se ha dicho que se iba a suponer que la matriz de amortiguamiento era nula. En el caso en el que se tuviera en cuenta una matriz de amortiguamiento proporcional, quedaría definida la matriz como:

$$\mathbf{C} = a_0 \mathbf{M} + a_1 \mathbf{K} \tag{2.17}$$

Los modos obtenidos en este caso son idénticos al caso anterior sin amortiguar. La diferencia es que en este caso las frecuencias naturales tienen una parte real y otra imaginaria como:

$$\zeta_r = \frac{\omega_r a_0}{2} + \frac{a_1}{2\omega_r} \tag{2.18}$$

$$\bar{\omega}_r = \omega_r \sqrt{1 - \zeta_r^2} \tag{2.19}$$

Donde ζ_r respresenta el factor de amortiguamiento y $\bar{\omega}_r$ representa la frecuencia natural amortiguada.

Los parámetros modales son muy útiles puesto que se puede discretizar

la respuesta por modos, de manera que principalemente se puedan analizar las respuestas estructurales por separado. Esto conlleva una reducción en el número de operaciones ya que el tamaño de las matrices se reduce. Además, el amortiguamiento modal permite predecir la vida a fatiga y reducir las respuestas en resonancia.

Los picos de la respuesta de una estructura están diréctamente relacionados con la cantidad de modos del sistema, cada pico respresenta un modo. Las frecuencias modales se pueden determinar observando los picos de la FRF.

La función de respuesta en frecuencia (FRF) puede obtenerse incluyendo en la fuerza la ecuación como $\mathbf{f} = \mathbf{f}(t)e^{i\omega t}$ e introduciéndolo en la ecuación de equilibrio de vibraciones forzadas se obtiene:

$$H(\omega) = [\big(K - \omega^2 M\big) + i(\omega C)]^{-1} \tag{2.20}$$

Como esta función es simétrica, con una de las filas pueden obtenerse los modos de manera que solo hay que desplazar la posición donde se aplica la fuerza. Los modos se obtiene con la parte imaginaria del espectro de frecuencia [7].

Algunas de las ventajas asociadas al modo superposición son:

- Comprender mejor el comportamiento de las estructuras.
- Controlar la integridad de la estructura para evitar problemas identificándolos.
- Simular cambios en las especificaciones como cambios de la fuerza aplicada. Permite analizar variedad de casos.
- Reducción del coste de las simulaciones siendo capaz de discretizar los modos.

2.2. Método de Newmark-β

El método lleva el nombre de Nathan M. Newmark, antiguo profesor de ingeniería civil de la Universidad de Illinois en Urbana-Champaign, que lo desarrolló en 1959 como un novedoso método de computación para resolución de problemas dinámicos estructurales. Para ello se centra en la resolución de la ecuación dinámica aproximando la solución en velocidades y desplazamientos [4].

Se parte de la ecuación de la dinámica en equilibrio:

$$\mathbf{M}\ddot{\mathbf{U}}^{\Delta t+t} + \mathbf{C}\dot{\mathbf{U}}^{\Delta t+t} + \mathbf{K}\mathbf{U}^{\Delta t+t} = \mathbf{f}^{\Delta t+t} \tag{2.21}$$

Para obtener la solución de los desplazamientos y velocidades $\mathbf{U}^{\Delta t+t}$ y $\dot{\mathbf{U}}^{\Delta t+t}$ se aproxima mediante una serie de Taylor:

$$\mathbf{U}^{\Delta t+t} = \mathbf{U}^t + \Delta t\dot{\mathbf{U}}^t + \frac{\Delta t^2}{2!}\ddot{\mathbf{U}}^t + \frac{\Delta t^3}{3!}\dddot{\mathbf{U}}^t + ... \tag{2.22}$$

$$\dot{\mathbf{U}}^{\Delta t+t} = \dot{\mathbf{U}}^t + \Delta t\ddot{\mathbf{U}}^t + \frac{\Delta t^2}{2!}\dddot{\mathbf{U}}^t + ... \tag{2.23}$$

Donde $\dddot{\mathbf{U}}^t$ representa la derivada tercera del campo de desplazamientos. Newmark truncó ambas series de forma que el último término fuera el que incluía la derivada tercera e incluyo algunos parámetros.

$$\mathbf{U}^{\Delta t+t} \simeq \mathbf{U}^t + \Delta t\dot{\mathbf{U}}^t + \frac{\Delta t^2}{2!}\ddot{\mathbf{U}}^t + \beta\Delta t^3\dddot{\mathbf{U}}^t \tag{2.24}$$

$$\dot{\mathbf{U}}^{\Delta t+t} \simeq \dot{\mathbf{U}}^t + \Delta t\ddot{\mathbf{U}}^t + \gamma\Delta t^2\dddot{\mathbf{U}}^t \tag{2.25}$$

Para calcular la derivada tercera del campo de desplazamientos (campo de aceleraciones), se supone que la aceleración varía de forma lineal en un intervalo de tiempo [t,t+Δt] y aplicando diferencias finitas se obtiene el resultando.

$$\dddot{\mathbf{U}}^t = \frac{\ddot{\mathbf{U}}^{\Delta t+t} - \ddot{\mathbf{U}}^t}{\Delta t} \tag{2.26}$$

Introduciendo la ecuación (2.26) en (2.24) se obtiene finalmente:

$$\mathbf{U}^{\Delta t+t} = \mathbf{U}^t + \Delta t\dot{\mathbf{U}}^t + \frac{\Delta t^2}{2!}\ddot{\mathbf{U}}^t + \beta\Delta t^3\frac{\ddot{\mathbf{U}}^{\Delta t+t} - \ddot{\mathbf{U}}^t}{\Delta t} \tag{2.27}$$

$$= \mathbf{U}^t + \Delta t\dot{\mathbf{U}}^t + (\frac{1}{2} - \beta)\Delta t^2\ddot{\mathbf{U}}^t + \beta\Delta t^2\ddot{\mathbf{U}}^{\Delta t+t} \tag{2.28}$$

$$\dot{\mathbf{U}}^{\Delta t+t} = \dot{\mathbf{U}}^t + \Delta t\ddot{\mathbf{U}}^t + \gamma\Delta t^2\frac{\ddot{\mathbf{U}}^{\Delta t+t} - \ddot{\mathbf{U}}^t}{\Delta t} \tag{2.29}$$

$$= \dot{\mathbf{U}}^t + (1-\gamma)\Delta t\ddot{\mathbf{U}}^t + \gamma\Delta t\ddot{\mathbf{U}}^{\Delta t+t} \tag{2.30}$$

Para obtener la ecuación que resuelve el campo de aceleraciones se despeja este de la ecuación (2.27) obteniendo:

$$\ddot{\mathbf{U}}^{\Delta t+t} = \frac{1}{\beta\Delta t^2}(\mathbf{U}^{\Delta t+t} - \mathbf{U}^t) - \frac{1}{\beta\Delta t}\dot{\mathbf{U}}^t + (1 - \frac{1}{2\beta})\ddot{\mathbf{U}}^t \tag{2.31}$$

Si la expresión del campo de aceleraciones de la ecuación (2.31) se introduce en la ecuación (2.29) se obtiene:

$$\dot{\mathbf{U}}^{\Delta t+t} = \frac{\gamma}{\beta\Delta t}(\mathbf{U}^{\Delta t+t} - \mathbf{U}^t) + (1 - \frac{\gamma}{\beta})\dot{\mathbf{U}}^t + (1 - \frac{\gamma}{2\beta})\Delta t\ddot{\mathbf{U}}^t \tag{2.32}$$

Con ello se obtiene el campo de velocidades y aceleraciones en función de los siguientes parámetros:

$$\dot{\mathbf{U}}^{\Delta t+t} = f(\mathbf{U}^{\Delta t+t}, \mathbf{U}^t, \Delta t, \beta, \gamma, \dot{\mathbf{U}}^t, \ddot{\mathbf{U}}^t) \tag{2.33}$$

$$\ddot{\mathbf{U}}^{\Delta t+t} = f(\mathbf{U}^{\Delta t+t}, \mathbf{U}^t, \Delta t, \beta, \dot{\mathbf{U}}^t, \ddot{\mathbf{U}}^t) \tag{2.34}$$

Los valores de los que dependen estas funciones son conocidos, por lo que solo quedaría definir el campo de desplazamientos en el instante [t+Δt]. Para ello simplemente se debe sustituir las ecuaciones (2.33) y (2.34) en la

ecuación de equilibrio dinámico (2.21) obteniendo la siguiente expresión,

$$\mathbf{K}_{\text{eff}}\ \mathbf{U}^{\Delta t+t} = \mathbf{F}_{\text{eff}} \tag{2.35}$$

donde:

$$\mathbf{K}_{\text{eff}} = \frac{1}{\beta\Delta t^2}\mathbf{M} + \frac{\gamma}{\beta\Delta t}\mathbf{C} + \mathbf{K} \tag{2.36}$$

$$\mathbf{F}_{\text{eff}} = \mathbf{f}^{\Delta t+t} + \left(\frac{1}{\beta\Delta t^2}\mathbf{M} + \frac{\gamma}{\beta\Delta t}\mathbf{C}\right)\mathbf{U}^t + \left(\frac{1}{\beta\Delta t}\mathbf{M} + (1-\frac{\gamma}{\beta})\mathbf{C}\right)\dot{\mathbf{U}}^t \tag{2.37}$$

$$- \left((1-\frac{1}{2\beta})\mathbf{M} + (1-\frac{\gamma}{2\beta}\right)\Delta t\mathbf{C})\ddot{\mathbf{U}}^t \tag{2.38}$$

Es común utilizar una serie de coeficientes que ayudan a simplificar las expresiones anteriores y que ayudaran a la hora de programar el esquema de integración. Los campos de velocidad y desplazamiento también quedan simplificados con estos parámetros. Los valores de los coeficientes se muestran a continuación:

$$a_1 = \frac{1}{\beta\Delta t^2} \qquad a_2 = -\frac{1}{\beta\Delta t} \qquad a_3 = 1 - \frac{1}{2\beta}$$

$$a_4 = \gamma\Delta t a_1 \qquad a_5 = 1 + \gamma\Delta t a_2 \qquad a_6 = \Delta t(1-\gamma+\gamma a_3)$$

Con ello el procedimiento de computación queda del siguiente modo:

1. Construción de las matrices $\mathbf{M}$, $\mathbf{C}$ y $\mathbf{K}$ y obtención de los coeficientes.
2. Construcción de la matriz $\mathbf{K}_{\text{eff}}$:

$$\mathbf{K}_{\text{eff}} = (a_1\mathbf{M} + a_4\mathbf{C} + \mathbf{K}) \tag{2.39}$$

3. Se establen las condiciones iniciales para incializar el cáculo de $\mathbf{U}^0$ y

$\dot{\mathbf{U}}^0$ de las cuales se obtiene:

$$\ddot{\mathbf{U}}^0 = \mathbf{M}^{-1}(\mathbf{f}^0 - \mathbf{C}\dot{\mathbf{U}}^0 - \mathbf{K}\mathbf{U}^0) \tag{2.40}$$

4. Se actualizan las variables inicializadas como obtenidas.

5. A partir de ahora para cada paso de tiempo $t + \Delta t$.

6. Se obtiene en primer lugar $\mathbf{F}_{\text{eff}}$:

$$\mathbf{F}_{\text{eff}} = \mathbf{f}^{\Delta t+t} + \mathbf{M}(a_1\mathbf{U}^t - a_2\dot{\mathbf{U}}^t - b_3\ddot{\mathbf{U}}^t) + \mathbf{C}(a_4\mathbf{U}^t - a_5\dot{\mathbf{U}}^t - b_6\ddot{\mathbf{U}}^t) \tag{2.41}$$

7. Se resuelve el sistema de la ecuación (2.35) para obtener el campo de desplazamientos.

8. Se calcula el campo de velocidades y aceleraciones mediante:

$$\left.\begin{aligned} \ddot{\mathbf{U}}^{\Delta t+t} &= a_1(\mathbf{U}^{\Delta t+t} - \mathbf{U}^t) + a_2\dot{\mathbf{U}}^t + b_3\ddot{\mathbf{U}}^t \\ \dot{\mathbf{U}}^{\Delta t+t} &= a_4(\mathbf{U}^{\Delta t+t} - \mathbf{U}^t) + a_5\dot{\mathbf{U}}^t + b_6\ddot{\mathbf{U}}^t \end{aligned}\right\}$$

9. Se actualizan las variables de forma que:

$$\mathbf{U}^t \Leftarrow \mathbf{U}^{\Delta t+t} \;;\; \dot{\mathbf{U}}^t \Leftarrow \dot{\mathbf{U}}^{\Delta t+t} \;;\; \ddot{\mathbf{U}}^t \Leftarrow \ddot{\mathbf{U}}^{\Delta t+t}$$

10. Por último se vuelve al comienzo y se resuelve para $t + \Delta t$.

Los parámetros γ y β del esquema son parámetros que sirven para alcanzar la precisión deseada del método. En función del valor de estos parámetros, se estará aproximando el campo de aceleraciones de una manera u otra. Cuando se tiene $\gamma = 0.5$ y $\beta = 1/6$, entonces se está aplicando el método donde se aproxima la aceleración de forma lineal como se ve en la figura 2.2. Newmark propuso unos valores de estas constantes de $\gamma = 0.5$ y $\beta = 0.25$ con el fin de que el esquema fuera incondicionalmente estable, como se verá en la sección de estabilidad. El método con estas constantes se conoce como

la regla del trapecio (figura 2.3).

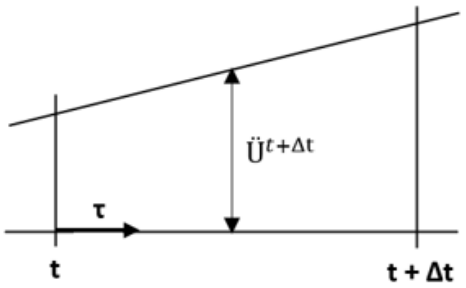

Figura 2.2: Aceleración lineal.

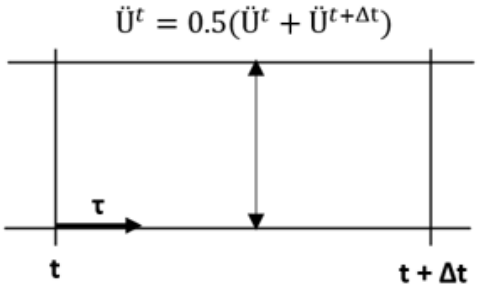

Figura 2.3: Regla del trapecio.

El método de Newmark-β es un método con uso extendido en aplicaciones de dinámica estructural. Proporciona una solución con un coste computacional muy bajo y con una precisión adecuada. El hecho de poder controlar la respuesta mediante los parámetros β y γ y el paso temporal Δt, permite crear distintos esquemas que permitan resolver problemas que requieren distintos tipos de necesidades computacionales.

Una de las principales desventajas del método es su fuerte dependencia de precisión con el paso temporal utilizado, por lo que es un parámetro crítico. Además como se verá en los resultados, el método tiene dificultad para amortiguar los modos de alta frecuencia cuando la diferencia de orden entre las frecuencias naturales del sistema son elevadas; lo que genera oscilaciones espurias en la solución.

2.3. Método de Bathe

El método de integración numérica propuesto por Bathe [8], a diferencia del método de Newmark, emplea dos pasos temporales para analizar la respuesta en un paso Δt (dos subpasos).

Para el primer subpaso temporal, el método de Bathe usa la regla del trapecio del método de Newmark y en el segundo subpaso temporal utiliza

el método de Euler de tres puntos hacia atrás.

En este esquema se utiliza para la interpolación de la solución en el primer subpaso temporal la regla del trapecio de Newmark que es incondicionalmente estable y tiene una precisión de segundo orden en cuanto al análisis lineal. Cuando se realizan análisis no lineales, este método puede volverse inestable y por lo tanto obtener una respuesta dinámica incorrecta, ya que se dejaría de conservar la energía y el momento.

Una manera de aliviar este efecto podría ser mediante la introducción de un término de amortiguamiento, modificando distintos parámetros, lo cual es la base de los métodos α. Estos no estarán recogidos en este TFM. Otra forma para resolver este problema, es mediante un esquema de integración temporal que no dependiera del ajuste de distintos parámetros para llegar a la solución, sino que simplemente hubiera que ajustar el paso temporal Δt. [8]

De esta necesidad surge el método de Bathe. Antes de este método, este tipo de estructura se probó y demostró su funcionalidad en sistemas de primer orden con resultados satisfactorios.

Para este método, que denominará a lo largo del trabajo como 'el método clásico de Bathe', hay un parámetro importante que define el comportamiento del esquema de integración que es el valor de cada subpaso temporal. Para este método, se va a definir un subpaso temporal igual para cada subpaso de modo que el tiempo de cada uno de ellos será la mitad del paso temporal Δt. De esta forma se va a explicar cómo quedaría definido matemáticamente el método.

Tras lo explicado, los campos de velocidad y aceleraciones para el primer subpaso temporal, habiendo definido en el esquema de Newmark el método del trapecio, quedan como:

$$\mathbf{U}^{\Delta t/2+t} = \mathbf{U}^t + \frac{\Delta t}{4}(\dot{\mathbf{U}}^{\Delta t/2+t} + \dot{\mathbf{U}}^t) \tag{2.42}$$

$$\dot{\mathbf{U}}^{\Delta t/2+t} = \mathbf{U}^t + \frac{\Delta t}{4}(\ddot{\mathbf{U}}^{\Delta t/2+t} + \ddot{\mathbf{U}}^t) \tag{2.43}$$

Introduciendo las ecuaciones (2.43) (2.42) y despejando el campo de aceleraciones se obtiene:

$$\ddot{\mathbf{U}}^{\Delta t/2+t} = \mathbf{U}^t + \frac{\Delta t}{2}\dot{\mathbf{U}}^t + \frac{\Delta t^2}{16}(\dot{\mathbf{U}}^{\Delta t/2+t} + \dot{\mathbf{U}}^t) \tag{2.44}$$

Haciendo uso de las ecuaciones (2.42), (2.43) y (2.44) en la ecuación (2.21) para un paso de tiempo $t+\Delta t/2$ se obtiene una ecuación similar a la ecuación (2.35) donde sus componentes toman los siguientes valores:

$$\mathbf{K}_{\text{eff}}^{(t+\Delta t/2)} = \frac{16}{\Delta t^2}\mathbf{M} + \frac{4}{\Delta t}\mathbf{C} + K \tag{2.45}$$

$$\mathbf{F}_{\text{eff}}^{(t+\Delta t/2)} = \mathbf{f}^{\Delta t/2+t} + \mathbf{M}(\frac{16}{\Delta t^2}\mathbf{U}^t + \frac{8}{\Delta t}\dot{\mathbf{U}}^t + \ddot{\mathbf{U}}^t) + \mathbf{C}(\frac{4}{\Delta t}\mathbf{U}^t + \dot{\mathbf{U}}^t) \tag{2.46}$$

Se resuelve el segundo paso temporal $t+\Delta t$ aplicando el método de euler hacia atrás de 3 puntos obteniendo los siguientes campos de velocidades y aceleraciones:

$$\dot{\mathbf{U}}^{\Delta t+t} = \frac{1}{\Delta t}\mathbf{U}^t - \frac{4}{\Delta t}\mathbf{U}^{\Delta t/2+t} + \frac{3}{\Delta t}\mathbf{U}^{\Delta t+t} \tag{2.47}$$

$$\ddot{\mathbf{U}}^{\Delta t+t} = \frac{1}{\Delta t}\dot{\mathbf{U}}^t - \frac{4}{\Delta t}\dot{\mathbf{U}}^{\Delta t/2+t} + \frac{3}{\Delta t}\dot{\mathbf{U}}^{\Delta t+t} \tag{2.48}$$

Combinando las ecuaciones (2.47) y (2.48) e introduciendo en la ecuación (2.21) se obtiene como para el anterior subpaso una ecuación de la forma (2.35) con los siguientes coeficientes:

$$\mathbf{K}_{\text{eff}}^{(t+\Delta t)} = \frac{9}{\Delta t^2}\mathbf{M} + \frac{3}{\Delta t}\mathbf{C} + K \tag{2.49}$$

$$\mathbf{F}_{\text{eff}}^{(t+\Delta t)} = \mathbf{f}^{\Delta t+t} + \mathbf{M}(\frac{12}{\Delta t^2}\mathbf{U}^{\Delta t/2+t} - \frac{3}{\Delta t^2}\mathbf{U}^{t} + \frac{4}{\Delta t}\dot{\mathbf{U}}^{\Delta t/2+t} \tag{2.50}$$

$$- \frac{1}{\Delta t}\dot{\mathbf{U}}^{t}) + \mathbf{C}(\frac{4}{\Delta t}\mathbf{U}^{\Delta t/2+t} - \frac{1}{\Delta t}\mathbf{U}^{t})$$

De la misma manera que en el método de Newmark, se van a utilizar unos coeficientes para simplificar las ecuaciones. Los coeficientes tienen los siguientes valores:

$$b_0 = \frac{16}{\Delta t^2} \qquad b_1 = \frac{4}{\Delta t} \qquad b_2 = \frac{9}{\Delta t^2}$$

$$b_3 = \frac{3}{\Delta t} \qquad b_4 = 2b_1 \qquad b_5 = \frac{12}{\Delta t^2}$$

$$b_6 = -\frac{3}{\Delta t^2} \qquad b_7 = -\frac{1}{\Delta t}$$

Con ello el procedimiento de computación queda del siguiente modo:

1. Construción de las matrices $\mathbf{M}$, $\mathbf{C}$ y $\mathbf{K}$ y obtención de los coeficientes.
2. Construcción de la matrices $K^{eff(t+\Delta t/2)}$ y $K^{eff(t+\Delta t)}$:

$$\mathbf{K}_{\text{eff}}^{(t+\Delta t/2)} = (b_0\mathbf{M} + b_1\mathbf{C} + \mathbf{K})$$
$$\mathbf{K}_{\text{eff}}^{(t+\Delta t)} = (b_2\mathbf{M} + b_3\mathbf{C} + \mathbf{K})$$

3. Se establen las condiciones iniciales para inicializar el cáculo de $\mathbf{U}^0$ y $\dot{\mathbf{U}}^0$, de las cuales se obtiene:

$$\ddot{\mathbf{U}}^0 = \mathbf{M}^{-1}(\mathbf{f}^0 - \mathbf{C}\dot{\mathbf{U}}^0 - \mathbf{K}\mathbf{U}^0) \tag{2.51}$$

4. Se actualizan las variables inicializadas como obtenidas.

Para cada paso temporal se realiza el siguiente procedimiento iterativo:

1. Se resuelve el primer subpaso $t + \Delta t/2$.
2. Se obtiene en primer lugar $\mathbf{F}_{\text{eff}}^{(t+\Delta t/2)}$:

$$\mathbf{F}_{\text{eff}}^{(t+\Delta t/2)} = \mathbf{f}^{\Delta t+t} + \mathbf{M}(b_0\mathbf{U}^t + b_4\dot{\mathbf{U}}^t + \ddot{\mathbf{U}}^t) + \mathbf{C}(b_1\mathbf{U}^t + \dot{\mathbf{U}}^t) \qquad (2.52)$$

3. Se resuelve el sistema de la ecuación (2.35) para obtener el campo de desplazamientos.
4. Se calcula el campo de velocidades y aceleraciones mediante:

$$\left.\begin{aligned} \ddot{\mathbf{U}}^{\Delta t/2+t} &= b_1(\dot{\mathbf{U}}^{\Delta t/2+t} - \dot{\mathbf{U}}^t) - \ddot{\mathbf{U}}^t \\ \dot{\mathbf{U}}^{\Delta t/2+t} &= b_1(\mathbf{U}^{\Delta t/2+t} - \mathbf{U}^t) - \dot{\mathbf{U}}^t \end{aligned}\right\}$$

5. Se resuelve el segundo subpaso $t + \Delta t$.
6. Se obtiene en primer lugar $\mathbf{F}_{\text{eff}}^{(t+\Delta t)}$:

$$\begin{aligned} \mathbf{F}_{\text{eff}}^{(t+\Delta t)} = \mathbf{f}^{\Delta t+t} &+ \mathbf{M}(b_5\mathbf{U}^{\Delta t/2+t} + b_6\mathbf{U}^t + b_1\dot{\mathbf{U}}^{\Delta t/2+t} + b_7\dot{\mathbf{U}}^t) \\ &+ \mathbf{C}(b_1\mathbf{U}^{\Delta t/2+t} + b_7\mathbf{U}^t) \end{aligned} \qquad (2.53)$$

7. Se resuelve el sistema de la ecuación (2.49) para obtener el campo de desplazamientos.
8. Se calcula el campo de velocidades y aceleraciones mediante:

$$\left.\begin{aligned} \dot{\mathbf{U}}^{\Delta t+t} &= -b_7\mathbf{U}^t - b_1\mathbf{U}^{\Delta t/2+t} + b_3\mathbf{U}^{\Delta t+t} \\ \ddot{\mathbf{U}}^{\Delta t+t} &= -b_7\dot{\mathbf{U}}^t - b_1\dot{\mathbf{U}}^{\Delta t/2+t} + b_3\dot{\mathbf{U}}^{\Delta t+t} \end{aligned}\right\}$$

9. Se actualizan las variables de forma que:

$$\mathbf{U}^t \Leftarrow \mathbf{U}^{\Delta t+t} \;;\; \dot{\mathbf{U}}^t \Leftarrow \dot{\mathbf{U}}^{\Delta t+t} \;;\; \ddot{\mathbf{U}}^t \Leftarrow \ddot{\mathbf{U}}^{\Delta t+t}$$

10. Por último se vuelve al comienzo (paso 1 del proceso iterativo) y se resuelven otra vez los dos subpasos hasta alcanzar el tiempo deseado.

Este esquema tiene como particularidad el uso de varios pasos temporales para calcular la respuesta en un paso de tiempo Δt. Se podría imaginar que esto requeriría un mayor esfuerzo computacional para obtener el resultado, pero el potencial de este esquema reside en que usando pasos temporales más largos se pueden obtener resultados en cuanto a precisión y estabilidad muy buenos. Además, es potente en la resolución de sistemas no lineales.

En cuanto a los sistemas lineales, el método de Bathe clásico tiene un buen comportamiento en su aplicación a la dinámica estrucural porque elimina las oscilaciones espurias, que si ocurrían mediante el método de Newmark-β cuando hay mucha diferencia entre los modos asociados a altas frecuencias y los modos asociados a bajas frecuencias. Es capaz de captar bien la respuesta de todos los modos.

2.4. Método de Bathe Modificado

En el anterior método se proponía una solución para sistemas dinámicos tanto lineales como no lineales basado en dos subpasos temporales para resolver la respuesta en un paso de tiempo Δt. Se proponía como parámetro para controlar la estabilidad y precisión de la solución el paso temporal. En este método se propone un esquema similar al método de Bathe Clásico, pero con la introducción de distintos parámetros que permitan controlar la disipación numérica como se hace en los métodos α, *HHT*, *Wilson*... [5] [6].

Tanto en el método de Newmark-β como en el método de Bathe clásico hay unos parámetros clásicos de utilización para los esquemas, como son el subpaso temporal de valor la mitad del paso temporal o los parámetros γ y β del método de Newmark-*beta* con valores 0.5 y 0.25 respectivamente.

El uso de estos parámetros en general conduce a una solución de los esquemas robusta en términos de precisión, pero es posible en función del caso que se quiera analizar, una modificación de estos parámetros para el control de la disipación numérica.

Para ello, en este esquema se mantienen los dos subpasos temporales. En el primero, se utiliza la regla del trapecio de Newmark, como ocurría en el método de Bathe clásico y en el segundo subpaso, en vez de utilizar el método de Euler de tres puntos hacia atrás, utiliza la regla del trapecio con 3 puntos con los parámetros β_1 y β_2 que son los correspondientes a los usados en el método de Newmark. Estos parámetros fundamentalmente actúan como control de la caída de amplitud del método y de la elongación del periodo. Además, se ofrece la posibilidad de variar el parámetro del valor temporal del subpaso 'θ' que en el caso del método de Bathe clásico estaba fijado como 0.5. A este esquema se le conoce como el esquema β_1-β_2 *Bathe* [1], en este trabajo de fin de máster se le denominará como esquema de Bathe modificado.

Se presentan las ecuaciones que definen el método en el que los cambios se observarán en el segundo subpaso que es donde se introduce el cambio real del método. Para el primer subpaso de tiempo $t+\theta\Delta t$ las ecuaciones quedan como:

$$\mathbf{U}^{\theta\Delta t+t} = \mathbf{U}^t + \frac{\theta\Delta t}{2}(\dot{\mathbf{U}}^{\theta\Delta t+t} + \dot{\mathbf{U}}^t) \tag{2.54}$$

$$\dot{\mathbf{U}}^{\theta\Delta t+t} = \mathbf{U}^{\Delta t+t} + \frac{\theta\Delta t}{2}(\ddot{\mathbf{U}}^{\theta\Delta t+t} + \ddot{\mathbf{U}}^t) \tag{2.55}$$

De donde se obtiene:

$$\mathbf{U}^{\theta\Delta t+t} = \mathbf{U}^t + (\theta\Delta t)\dot{\mathbf{U}}^t + (\frac{\theta\Delta t}{2})^2(\ddot{\mathbf{U}}^{\theta\Delta t+t} + \ddot{\mathbf{U}}^t) \tag{2.56}$$

Siendo la ecuación del movimiento para este método,

$$\mathbf{M}\ddot{\mathbf{U}}^{\theta\Delta t+t} + \mathbf{C}\dot{\mathbf{U}}^{\theta\Delta t+t} + \mathbf{K}\mathbf{U}^{\Delta t+t} = \mathbf{f}^{\theta\Delta t+t} \tag{2.57}$$

Combinando las ecuaciones (2.54)-(2.56) e introduciendolas en la ecuación (2.57) se llega a una expresión del tipo (2.35) cuyas incógnitas valen:

$$\mathbf{K}_{\text{eff}}^{(t+\theta\Delta t)} = \frac{4}{(\theta\Delta t)^2}\mathbf{M} + \frac{2}{\theta\Delta t}\mathbf{C} + K \tag{2.58}$$

$$\mathbf{F}_{\text{eff}}^{(t+\theta\Delta t)} = \mathbf{f}^{\theta\Delta t+t} + \mathbf{M}(\frac{4}{(\theta\Delta t)^2}\mathbf{U}^t + \frac{4}{\theta\Delta t}\dot{\mathbf{U}}^t + \ddot{\mathbf{U}}^t) + \mathbf{C}(\frac{2}{\theta\Delta t}\mathbf{U}^t + \dot{\mathbf{U}}^t) \tag{2.59}$$

Se resuelve el segundo paso temporal $t + \Delta t$ aplicando el método de Newmark en 3 puntos obteniendo los siguientes campos de desplazamiento y velocidad:

$$\mathbf{U}^{\Delta t+t} = \mathbf{U}^t + (\theta\Delta t)(\beta_1\dot{\mathbf{U}}^{\theta\Delta t+t} + (1-\beta_1)\dot{\mathbf{U}}^t) \tag{2.60}$$

$$+ ((1-\theta)\Delta t)(\beta_2\dot{\mathbf{U}}^{\Delta t+t} + (1-\beta_2)\dot{\mathbf{U}}^{\theta\Delta t+t})$$

$$\dot{\mathbf{U}}^{\Delta t+t} = \dot{\mathbf{U}}^{\Delta t+t} + (\theta\Delta t)(\beta_1\ddot{\mathbf{U}}^{\theta\Delta t+t} + (1-\beta_1)\ddot{\mathbf{U}}^t) \tag{2.61}$$

$$+ ((1-\theta)\Delta t)(\beta_2\ddot{\mathbf{U}}^{\Delta t+t} + (1-\beta_2)\ddot{\mathbf{U}}^{\theta\Delta t+t})$$

Combinando las ecuaciones (2.60) y (2.61) como en el anterior subpaso del método se llega a un campo de velocidades y de aceleraciones de la siguiente forma:

$$\dot{\mathbf{U}}^{\Delta t+t} = \frac{1}{\beta_2(1-\theta)\Delta t}[\mathbf{U}^{\Delta t+t} - \mathbf{U}^t \tag{2.62}$$

$$- (\theta\Delta t)(1-\beta_1)\Delta t\dot{\mathbf{U}}^t - (\beta_1\theta\Delta t + (1-\theta)(1-\beta_2)\Delta t)\dot{\mathbf{U}}^{\theta\Delta t+t}]$$

$$\ddot{\mathbf{U}}^{\Delta t+t} = \frac{1}{\beta_2(1-\theta)\Delta t}[\dot{\mathbf{U}}^{\Delta t+t} - \dot{\mathbf{U}}^t \tag{2.63}$$

$$- (\theta\Delta t)(1-\beta_1)\Delta t\ddot{\mathbf{U}}^t - (\beta_1\theta\Delta t + (1-\theta)(1-\beta_2)\Delta t)\ddot{\mathbf{U}}^{\theta\Delta t+t}]$$

Antes de completar el esquema se van a definir un conjunto de coeficientes asociados al método para simplificar las ecuaciones en este paso temporal:

$$c_0 = \frac{2}{\theta\Delta t} \qquad c_1 = \frac{4}{(\theta\Delta t)^2} \qquad c_2 = \frac{4}{\theta\Delta t}$$

$$c_3 = \frac{1}{\beta_2(1-\theta)\Delta t} \quad c_4 = \frac{\theta(1-\beta_1)+\beta_2(1-\theta)}{(\beta_2(1-\theta))^2\Delta t} \quad c_5 = \frac{\theta\beta_1+(1-\beta_2)(1-\theta)}{(\beta_2(1-\theta))^2\Delta t}$$

$$c_6 = \frac{\theta(1-\beta_1)}{\beta_2(1-\theta)} \qquad c_7 = \frac{\theta\beta_1+(1-\beta_2)(1-\theta)}{\beta_2(1-\theta)} \quad c_8 = \frac{1}{(\beta_2(1-\theta))^2\Delta t}$$

Haciendo uso de los coeficientes, introduciendo la ecuación (2.62) en (2.63) y reescribiendo el campo de velocidades quedan definidos ambos campos por:

$$\dot{\mathbf{U}}^{\Delta t+t} = c_3(\mathbf{U}^{\Delta t+t} - \mathbf{U}^t) - c_6\dot{\mathbf{U}}^t - c_7\dot{\mathbf{U}}^{\theta\Delta t+t} \tag{2.64}$$

$$\ddot{\mathbf{U}}^{\Delta t+t} = c_8(\mathbf{U}^{\Delta t+t} - \mathbf{U}^t) - c_4\dot{\mathbf{U}}^t - c_5\dot{\mathbf{U}}^{\theta\Delta t+t} - c_6\ddot{\mathbf{U}}^t - c_7\ddot{\mathbf{U}}^{\theta\Delta t+t} \tag{2.65}$$

Introduciendo estas ecuaciones en la ecuación dinámica en equilibrio, se

obtiene la expresión de los desplazamientos en función de las matrices efectivas (2.35) cuyos coeficientes valen para el intervalo de tiempo $[t, t + \Delta t]$:

$$\mathbf{K}_{\text{eff}}^{(t+\Delta t)} = c_8\mathbf{M} + c_3\mathbf{C} + K \tag{2.66}$$

$$\mathbf{F}_{\text{eff}}^{(t+\theta\Delta t)} = \mathbf{f}^{\theta\Delta t+t} + \mathbf{M}(c_8\mathbf{U}^t + c_4\dot{\mathbf{U}}^t + c_5\dot{\mathbf{U}}^{\Delta t+t} + c_6\ddot{\mathbf{U}}^t \tag{2.67}$$

$$+ c_7\ddot{\mathbf{U}}^{\theta\Delta t+t}) + \mathbf{C}(c_3\mathbf{U}^t + c_6\dot{\mathbf{U}}^t + c_7\dot{\mathbf{U}}^{\theta\Delta t+t})$$

El procedimiento de computación para este esquema se enumera a continuación:

1. Construción de las matrices $\mathbf{M}$, $\mathbf{C}$ y $\mathbf{K}$ y obtención de los coeficientes.
2. Construcción de la matrices $\mathbf{K}_{\text{eff}}^{(t+\theta\Delta t)}$. $\mathbf{K}_{\text{eff}}^{(t+\Delta t)}$ queda determinada por la ecuación (2.66):

$$\mathbf{K}_{\text{eff}}^{(t+\theta\Delta t)} = (c_1\mathbf{M} + c_0\mathbf{C} + \mathbf{K})$$

3. Se establen las condiciones iniciales para incializar el cáculo de $\mathbf{U}^0$ y $\dot{\mathbf{U}}^0$, de las cuales se obtiene:

$$\ddot{\mathbf{U}}^0 = \mathbf{M}^{-1}(\mathbf{f}^0 - \mathbf{C}\dot{\mathbf{U}}^0 - \mathbf{K}\mathbf{U}^0) \tag{2.68}$$

4. Se actualizan las variables inicializadas como obtenidas.

Para cada paso temporal se realiza el siguiente procedimiento iterativo:

1. Se resuelve en primer lugar el primer subpaso $t + \theta\Delta t$.
2. Se obtiene $\mathbf{F}_{\text{eff}}^{(t+\theta\Delta t)}$:

$$\mathbf{F}_{\text{eff}}^{(t+\theta\Delta t)} = \mathbf{f}^{\Delta t+t} + \mathbf{M}(c_1\mathbf{U}^t + c_2\dot{\mathbf{U}}^t + \ddot{\mathbf{U}}^t) + \mathbf{C}(c_0\mathbf{U}^t + \dot{\mathbf{U}}^t)$$

3. Se resuelve el sistema de la ecuación (2.35) para obtener el campo de desplazamientos.

4. Se calcula el campo de velocidades y aceleraciones mediante:

$$\left.\begin{aligned}\ddot{\mathbf{U}}^{\theta\Delta t+t} &= c_0(\dot{\mathbf{U}}^{\theta\Delta t+t} - \dot{\mathbf{U}}^{t}) - \ddot{\mathbf{U}}^{t}\\ \dot{\mathbf{U}}^{\theta\Delta t+t} &= c_0(\mathbf{U}^{\theta\Delta t+t} - \mathbf{U}^{t}) - \dot{\mathbf{U}}^{t}\end{aligned}\right\}$$

5. Se resuelve el segundo subpaso $t + \Delta t$.

6. Se obtiene en primer lugar $\mathbf{F}_{\text{eff}}^{(t+\Delta t)}$ que viene dado por la ecuación (2.67).

7. Se resuelve el sistema de la ecuación (2.35) para obtener el campo de desplazamientos.

8. Se calcula el campo de velocidades y aceleraciones a través de las ecuaciones y (2.64) y (2.65).

9. Se actualizan las variables de forma que:

$$\mathbf{U}^{t} \Leftarrow \mathbf{U}^{\Delta t+t} \;;\; \dot{\mathbf{U}}^{t} \Leftarrow \dot{\mathbf{U}}^{\Delta t+t} \qquad ;\; \ddot{\mathbf{U}}^{t} \Leftarrow \ddot{\mathbf{U}}^{\Delta t+t}$$

10. Por último se vuelve al comienzo (paso 1 del proceso iterativo) y se resuelven otra vez los dos subpasos hasta alcanzar el tiempo deseado.

Se puede demostrar que usando $\beta_1 = \beta_2 = 0{,}5$ se llega al esquema de la regla trapezoidal en ambos subpasos temporales y con los valores de $\beta_1 = 1/3$, $\beta_2 = 2/3$ y $\theta = 0{,}5$ se obtiene el método de Bathe.

La ventaja principal de la inclusión de estos parámetros, es el control de la disipación numérica para cualquiera sea el caso que se esté analizando. Además, en función de los parámetros usados, se puede conseguir reducir la elongación del periodo de la respuesta del sistema en comparación con la que se tiene con el método de Bathe como se ve en la figura 2.4. El problema de este esquema es que no hay ninguna combinación de parámetros que se

considere perfecta, por lo tanto para cada problema deberá realizarse un estudio numérico para considerar cual es la mejor combinación.

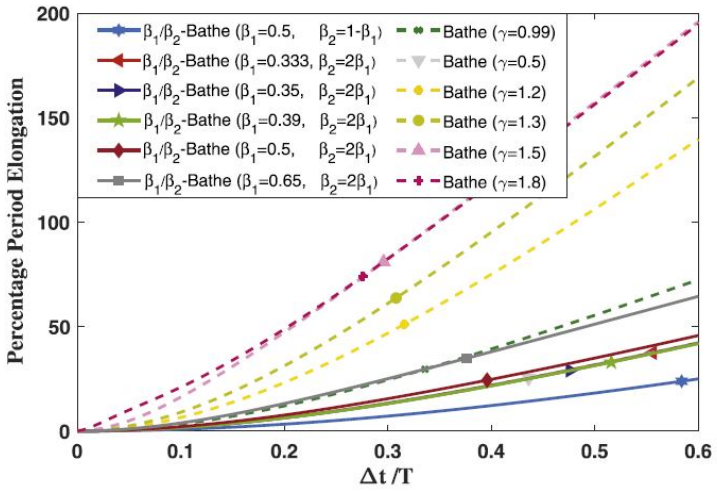

Figura 2.4: Porcentaje de elongación del periodo del método Bathe modificado en función de los parámetros β_1 y β_2 y el método de Bathe clásico [1]

2.5. Precise Integration Method (PIM)

Se trata de un método de integración numérica de alta precisión que puede resolver sistemas dinámicos estructurales lineales e invariantes en el tiempo.

Este esquema temporal resuelve los problemas de estabilidad que aparecen en el Método de Newmark-β o *Wilson*, los cuales no son capaces de representar los modos de alta frecuencia con precisión con un paso temporal Δ_t relativamente grande. Además, como su propio nombre indica, presenta un método preciso con un coste computacional pequeño

Para este esquema se parte de la ecuación de movimiento,

$$\mathbf{M\ddot{U}} + \mathbf{C\dot{U}} + \mathbf{KU} = \mathbf{f} \tag{2.69}$$

con sus respectivas condiciones iniciales de desplazamiento y velocidad.

Después se realiza una transformación Hamiltoniana [9]- [10], aunque la ecuación a transformar como tal no sea de ese tipo. La transformación mencionada se desarrolla a continuación. Se define primero una transformación del vector solución en desplazamientos:

$$\mathbf{p} = \mathbf{M}\dot{\mathbf{U}} + \frac{\mathbf{C}\mathbf{U}}{2} \tag{2.70}$$

$$\dot{\mathbf{U}} = \mathbf{M}^{-1}\mathbf{p} - \frac{\mathbf{M}^{-1}\mathbf{C}\mathbf{U}}{2} \tag{2.71}$$

Que si se introduce en la ecuación (2.69) se reescribe como:

$$\dot{\mathbf{p}} = -\left(\mathbf{K} - \frac{\mathbf{C}\mathbf{M}^{-1}\mathbf{C}}{4}\right)\mathbf{U} - \frac{\mathbf{C}\mathbf{M}^{-1}\mathbf{p}}{2} + \mathbf{f} \tag{2.72}$$

De estas ecuaciones se puede obtener un sistema de ecuaciones lineales duales,

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{A}\mathbf{q} + \mathbf{D}\mathbf{p} + \mathbf{f}_q \\ \dot{\mathbf{p}} &= \mathbf{B}\mathbf{q} + \mathbf{C}\mathbf{p} + \mathbf{f}_p \end{aligned} \tag{2.73}$$

donde,

$$\mathbf{q} = \mathbf{U} \quad ; \quad \mathbf{A} = -\frac{\mathbf{M}^{-1}\mathbf{C}}{2}$$

$$\mathbf{B} = -\left(\mathbf{K} - \frac{\mathbf{C}\mathbf{M}^{-1}\mathbf{C}}{4}\right) \quad ; \quad \mathbf{C} = \frac{\mathbf{C}\mathbf{M}^{-1}}{2}$$

$$\mathbf{D} = \mathbf{M}^{-1}; \qquad \mathbf{f}_p = \mathbf{f}; \qquad \mathbf{f}_q = 0$$

Para la resolución de este sistema, la teoría Hamiltoniana sería válida

si el sistema fuera conservativo, pero en el momento que se introduce una disipación en forma de matriz de amortiguamiento **C**, este sistema ya no es válido y hay que recurrir a un método temporal de integración directa.

Después de aplicar la transformación, el sistema resultante, que puede ser mediante vectores o mediante matrices, se puede expresar mediante,

$$\dot{\mathbf{v}} = \mathbf{H}\mathbf{v} + \mathbf{g} \tag{2.74}$$

con unas condiciones iniciales de $\mathbf{v(0)}$ dadas donde,

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{D} \\ \mathbf{B} & \mathbf{C} \end{bmatrix} \; ; \mathbf{v} = \begin{bmatrix} \mathbf{q} \\ \mathbf{p} \end{bmatrix} \tag{2.75}$$

Esta ecuación diferencial ordinaria, se puede resolver mediante el análisis de su solución homogénea y su solución particular.

2.5.1. Solución homogénea

Debido a que la matriz **H** es una matriz que no es dependiente del tiempo, la solución general a la ecuación viene dada por:

$$\mathbf{v} = \exp(\mathbf{H}t) \cdot \mathbf{v}_0 \tag{2.76}$$

Se introduce el concepto de matriz exponencial **T**, que va a ser la base de este método. Esta puede escribirse como:

$$\mathbf{T} = \exp(\mathbf{H}\Delta t) \tag{2.77}$$

2.5.1.1. Computación de la matriz exponencial T

Para la computación de la matriz exponencial, se va a recurrir a la división del paso temporal Δt en muchos pasos temporales (τ). Para ello se define el parámetro $m = 2^N$, donde N siempre será un número entero. Se reescribe la matriz exponencial haciendo uso de este parámetro.

$$\exp(\mathbf{H}\tau) = \left[\exp\left(\frac{\mathbf{H}\Delta t}{m}\right)\right]^m \tag{2.78}$$

Se puede expandir esta matriz según el desarrollo de Taylor.

$$\exp(\mathbf{H}\tau) = \mathbf{I}_n + \mathbf{H}\tau + \frac{(\mathbf{H}\tau)^2}{2} + \frac{(\mathbf{H}\tau)^3}{3!} + \frac{(\mathbf{H}\tau)^4}{4!} + ... \tag{2.79}$$

Definida esta matriz el problema se reduce a resolver,

$$\mathbf{v}_{k+1} = \mathbf{T}\mathbf{v}_k \tag{2.80}$$

con $k = 1, 2, 3..., n$. Donde n sería el número de pasos temporales a resolver del problema.

Al realizar el cambio del paso temporal, el nuevo paso temporal en caso de tener una N de valor elevado sería muy pequeño y ello nos permite truncar la expresión de (2.79).

$$\exp(\mathbf{H}\tau) = \mathbf{I}_n + \mathbf{H}\tau + \frac{(\mathbf{H}\tau)^2}{2} + \frac{(\mathbf{H}\tau)^3}{3!} + \frac{(\mathbf{H}\tau)^4}{4!} \tag{2.81}$$

La expresión de la matriz exponencial tiene en su primer término la matriz unidad y va seguida de términos mucho más pequeños debido al paso temporal definido. Esto puede escribirse,

$$\exp(\mathbf{H}\tau) \approx \mathbf{I}_n + \mathbf{T}_a \tag{2.82}$$

$$\mathbf{T}_a = (\mathbf{H}\tau) + (\mathbf{H}\tau)^2\left[\mathbf{I}_n + \frac{(\mathbf{H}\tau)}{3} + \frac{(\mathbf{H}\tau)^2}{12}\right]/2 \tag{2.83}$$

La matriz $\mathbf{T}_a$ puede demostrarse que es muy pequeña. Es por ello que en todas las iteraciones, de la matriz $\mathbf{T}$ sólo se guarda el término $\mathbf{T}_a$ puesto que si no la precisión del método se vería afectada. Siguiendo el desarrollo de Zhong [10] la matriz completa $\mathbf{T}_a$ se obtendría a partir de la iteración de la siguiente ecuación desde 1 hasta N:

$$\mathbf{T}_a = 2\mathbf{T}_a + \mathbf{T}_a \cdot \mathbf{T}_a \tag{2.84}$$

Una vez esté la matriz $\mathbf{T}_a$ computada sí que se puede añadir la matriz unidad para completar el valor de $\mathbf{T}$ de forma que,

$$\mathbf{T} = \mathbf{I} + \mathbf{T}_a \tag{2.85}$$

Mediante la computación de la matriz exponencial $\mathbf{T}$ queda definida la solución del problema homogéneo y faltaría definir la resolución del problema particular.

2.5.2. Solución particular

Para la solución particular, se tiene en cuenta el vector de fuerzas externas $\mathbf{f}$, lo cual hace que la respuesta debida a esa fuerza se pueda computar mediante la integral de Duhamel, donde t es un instante arbitrario en el tiempo.

$$\mathbf{v}\left(t_{k+1}\right) = \mathbf{T}(\Delta t)\mathbf{v}\left(t_k\right) + \int_{t_k}^{t_{k+1}} \mathbf{T}\left(t_{k+1} - \tau\right)\mathbf{g}(\tau)d\tau \tag{2.86}$$

Donde en la ecuación (2.86) t_k es el instante de tiempo k-ésimo y la matriz exponencial se computa mediante la siguiente expresión, como se ha visto:

$$\mathbf{T} = \exp[t\mathbf{H}] \tag{2.87}$$

El segundo término de la ecuación (2.86) es la integral particular donde se recoge el vector de fuerzas $\mathbf{g}(\tau)$, que recoge el valor de la fuerza a lo largo del paso temporal. Operando la integral y suponiendo que:

$$\mathbf{g}(t) = \mathbf{r}_0 + \mathbf{r}_1(t - t_k) \tag{2.88}$$

entonces,

$$\mathbf{v}_{k+1} = \mathbf{T}[\mathbf{v}_k + \mathbf{H}^{-1}(\mathbf{r}_0 + \mathbf{H}^{-1}\mathbf{r}_1)] - \mathbf{H}^{-1}[\mathbf{r}_0 + \mathbf{H}^{-1}\mathbf{r}_1 + \Delta t\mathbf{r}_1] \tag{2.89}$$

Para obtener la derivada de $\mathbf{v}$, simplemente se resolvería la ecuación (2.74) y con ello quedaría cerrado el problema.

En este método el paso crítico es la evaluación de la matriz exponencial $\mathbf{T}$. El error puede venir inducido de la truncación de la serie de Taylor y en errores de redondeos del programa de cálculo usado. El método PIM es muy útil puesto que computacionalmente es muy eficiente. Se puede ver que la resolución del problema depende directamente de la computación de la matriz exponencial cuya resolución no conlleva muchas operaciones. Una vez esta está evaluada, resolver la ecuación con la que se obtiene el resultado en el paso $t + \Delta t$ es muy sencillo dado que la matriz exponencial $\mathbf{T}$ se queda almacenada. Además, otra de las ventajas sobre el resto de métodos expuestos, es que este método tiene la capacidad de resolver problemas que varían en el tiempo. Esto es que las matrices del problema varian en función del tiempo.

Este método, a diferencia del método de Runge-Kutta, divide el paso temporal en 2^N subpasos y evalúa los incrementos del subpaso temporal (matriz exponencial). En el caso de Runge-Kutta, se evalua el valor completo del vector y no los incrementos, lo cual hace que no sea capaz de evaluar la solución con tanta precisión [11].

2.6. Método de Runge-Kutta

La resolución de ecuaciones diferenciales mediante un esquema de integración de Runge-Kutta de cualquier orden, sólo es válida para sistemas de ecuaciones diferenciales o ecuaciones diferenciales de primer orden.

Para el caso que se quiere analizar, que es el de la resolución de la ecuación de la dinámica, este método de integración directa no podría utilizarse puesto que la ecuación en cuestión es de segundo orden. Con el fin de poder utilizar este esquema de integración en esta ecuación, se tiene que hacer uso del modelo de espacio de los estados.

Este modelo se suele utilizar cuando queremos realizar un control sobre un proceso. El modelo de espacio de los estados debe proporcionar un modelo lo más similar posible al modelo que está siendo modificado. El modelo parte de la ecuación que se quiere resolver, en este caso la ecuación dinámica:

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{C}\dot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{f} \tag{2.90}$$

El objetivo del modelo de espacio de los estados va a ser generar una ecuación o sistema de ecuaciones de primer orden. La ecuación de equilibrio dinámico también se cumple cuando la analizamos en un instante de tiempo $t + \Delta t$, como se ha visto en los anteriores métodos. Se propone el siguiente cambio de coordenadas de modo que el vector $\mathbf{U}$ que contiene los desplazamientos ahora se puede escribir como:

$$\begin{cases} \mathbf{x}_1 = \mathbf{U} \\ \mathbf{x}_2 = \dot{\mathbf{U}} \end{cases} \xrightarrow[Diferenciando]{\frac{d}{dt}} \begin{cases} \dot{\mathbf{x}}_1 = \dot{\mathbf{U}} \\ \dot{\mathbf{x}}_2 = \ddot{\mathbf{U}} \end{cases} \xrightarrow{\dot{\mathbf{x}}_1 = \mathbf{x}_2} \begin{cases} \dot{\mathbf{x}}_1 = \mathbf{x}_2 \\ \dot{\mathbf{x}}_2 = \ddot{\mathbf{U}} \end{cases} \tag{2.91}$$

Con ello se obtiene un sistema de ecuaciones formado por:

$$\begin{cases} \dot{\mathbf{x}}_1 = \mathbf{x}_2 \\ \\ \mathbf{M}\dot{\mathbf{x}}_2 + \mathbf{C}\mathbf{x}_2 + \mathbf{K}\mathbf{x}_1 = \mathbf{f} \end{cases} \tag{2.92}$$

Diviendo la ecuación (2.92) entre la matriz de masas $\mathbf{M}$ se obtiene:

$$\begin{cases} \dot{\mathbf{x}}_1 = \mathbf{x}_2 \\ \\ \dot{\mathbf{x}}_2 = -\mathbf{M}^{-1}\mathbf{C}\mathbf{x}_2 - \mathbf{M}^{-1}\mathbf{K}\mathbf{x}_1 + \mathbf{M}^{-1}\mathbf{f} \end{cases} \tag{2.93}$$

La ecuación del modelo de espacio de los estados esta formado por 2 ecuaciones. En la primera ecuación, se encuentran los vectores de estado que definen la situación del problema como se indica en la ecuación (2.94), y en

la segunda ecuación los vectores de salida aunque esta última se utiliza en aplicaciones de control.

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{2.94}$$

Para la resolución de la ecuación dinámica, solo se va a hacer uso de la ecuación de estado y no de la ecuación de salida, puesto que simplemente se quiere resolver la ecuación de primer orden mediante el esquema de Runge-Kutta.

En vista de la ecuación (2.94), se necesitan definir la matriz **A** y la Matriz **B**, ambas denominadas las matrices de dinámica de tiempo discreto (matriz de estado) y la matriz de entrada respectivamente. El vector de estado, denominado como $\mathbf{x}$(t), es el llamado vector de estados y contiene las soluciones de la ecuación para cada instante de tiempo. El vector u es el vector que contiene la excitación o las fuerzas asociadas al sistema dinámico. El valor de las matrices **A** y **B** en función del sistema de ecuaciones (2.93) es:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix} \; ; \mathbf{B} = \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1} \end{bmatrix} \tag{2.95}$$

Por lo que el sistema queda como:

$$\begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix} \tag{2.96}$$

Los autovalores de la matriz de estados **A** contienen los polos del sistema que son los análogos a los obtenidos al resolver el problema de autovalores de la ecuación dinámica. La parte real es la resta de la frecuencia natural por el ratio de amortiguamiento y la parte compleja es la frecuencia natural amortiguada. En un caso sin amortiguamiento se obtienen directamente los valores de las frecuencias naturales. Con ello, se tiene una ecuación que sí puede ser resuelta por cualquiera de los esquemas de integración de Runge-Kutta.

2.6.1. Esquema de Runge-Kutta

Una vez se ha conseguido reducir el orden de las ecuaciones a solucionar se va a explicar el esquema con el que se va a resolver el sistema de la ecuación (2.96). La historia de este método comienza cuando se intentan generalizar las expresiones del esquema de Euler con varias evaluaciones de la solución en un mismo paso. Esta idea fue llevada a cabo por Runge. Más tarde, fue Kutta quien realizó más aportaciones en este campo obteniendo finalmente la resolución de problemas con esquemas de cuarto y hasta quinto orden.

El método de Euler trata básicamente de analizar la tangente de una curva que es muy cercana a la curva de la solución exacta. La idea de los métodos de Runge-Kutta es analizar la tangente de la curva en un paso temporal, pero no solamente una vez como hace Euler, si no varias evaluaciones de la tangente para aproximar la curva. Estos métodos se han seguido trabajando durante años debido a sus buenas prestaciones computacionales tanto en los métodos implícitos como en los explícitos, en donde los métodos implícitos actualmente tienen una mayor potencia debido a la capacidad de resolver ecuaciones diferenciales rígidas. Son muy utilizados para resolver problemas de CFD (Computational Fluid Dynamics).

El esquema que se va a ampliar para el análisis dinámico es el que proporciona Runge-Kutta de cuarto orden, puesto que es uno de los más famosos y de los que proporciona mejores resultados. Los esquemas de Runge-Kutta son válidos para la resolución de cualquier ecuación deferencial de la forma: [12]

$$\dot{x} = f(t, x) \tag{2.97}$$

En los métodos de Runge-Kutta de alto orden, las condiciones algebraicas de los coeficientes asociadas al método se vuelven un poco más complicadas. Para explicar el método de cuarto orden de Runge-Kutta usado, antes se va a comentar la tabla de coeficientes que representa los métodos de Runge Kutta, también conocida como *Matriz de Butcher*. El tablón tiene la siguiente forma:

c	**A**
	b^{T}

Tabla 2.1: *Matriz de Butcher*

En donde el vector c es el que indica las posiciones de los valores de la etapa dentro del paso temporal, **A** es una matriz que indica cuál es la dependencia de las derivadas que se encuentran en otras etapas y el vector b es un vector que contiene el peso de la cuadratura, es decir, muestra la dependencia del resultado de las derivadas evaluadas en distintas etapas del paso temporal. Para el caso de los métodos explícitos se remarca que la matriz **A** será una matriz con ceros en la diagonal y en la parte triangular superior (matriz triangular inferior).

Una vez explicado cómo se representan los métodos de Runge-Kutta, el tablón asociado al **método de cuarto orden**, que es el que se va a utilizar en el trabajo, tendría el siguiente aspecto:

0				
c_2	a_{21}			
c_3	a_{31}	a_{32}		
c_4	a_{41}	a_{42}	a_{43}	
	b_1	b_2	b_3	b_4

Tabla 2.2: *Matriz de Butcher* para esquema de Runge-Kutta de cuarto orden

Los coeficientes del tablón, según las condiciones que deben cumplir [12], son para el esquema de cuarto orden:

$$b_1 + b_2 + b_3 + b_4 = 1 \tag{2.98}$$

$$b_2 c_2 + b_3 c_3 + b_4 c_4 = \frac{1}{2} \tag{2.99}$$

$$b_2 c_2^2 + b_3 c_3^2 + b_4 c_4^2 = \frac{1}{3} \tag{2.100}$$

$$b_3 a_{32} c_2 + b_4 a_{42} c_2 + b_4 a_{43} c_3 = \frac{1}{6} \tag{2.101}$$

$$b_2 c_2^3 + b_3 c_3^3 + b_4 c_4^3 = \frac{1}{4} \tag{2.102}$$

$$b_3 c_3 a_{32} c_2 + b_4 c_4 a_{42} c_2 + b_4 c_4 a_{43} c_3 = \frac{1}{8} \tag{2.103}$$

$$b_2 a_{32} c_2^2 + b_3 a_{42} c_3^2 + b_4 a_{43} c_4^2 = \frac{1}{12} \tag{2.104}$$

$$b_4 a_{43} a_{32} c_2 = \frac{1}{24} \tag{2.105}$$

De la combinación de estas ecuaciones se demuestra que $c_4 = 1$. Por lo tanto, la Matriz de Butcher que va a ser utilizada queda con los siguientes valores:

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$

Tabla 2.3: Coeficientes de la *Matriz de Butcher* para el esquema RK41

El esquema general de un método de Runge-Kutta de R-evaluaciones matemáticamente se expresa como [13]:

$$\begin{cases} x_{n+1} = x_n + \Delta t \Phi(t_n, x_n; \Delta t) \\ \\ \Phi(t, x; \Delta t) = \sum_{r=1}^{R} b_r k_r \\ \\ k_1 = f(t, x) \\ \\ k_r = f\Big(t + \Delta t c_r, x + \Delta t \sum_{s=1}^{r-1} a_{rs} k_s\Big), \ r = 2, 3, \ldots, R. \\ \\ c_r = \sum_{s=1}^{r-1} a_{rs}, \ r = 2, 3, \ldots, R. \end{cases} \tag{2.106}$$

Se utilizan 'R' evaluaciones de la función f, análoga a la ecuación (2.97), y después se hace una media ponderada en Φ que requiere que se cumpla $\sum_{r=1}^{R} b_r = 1$. La función f representa la ecuación que se quiere resolver. Se va a utilizar la siguiente notación para llegar al esquema final de cuarto orden,

$$f_t = \frac{\partial f}{\partial t} \qquad f_x = \frac{\partial f}{\partial x}$$

$$f_{tt} = \frac{\partial^2 f}{\partial t^2} \qquad f_{tx} = \frac{\partial^2 f}{\partial t \partial x} \qquad f_{xx} = \frac{\partial^2 f}{\partial x^2}$$

y,

$$F = f_t + f f_y, \qquad G = f_{tt} + 2 f f_{tx} + f^2 f_{xx}$$

Mediante esta notación se obtiene la función Φ usando un desarrollo de Taylor;

$$\Phi_T(t, x, \Delta t) = f + \frac{1}{2}\Delta t F + \frac{1}{6}\Delta t^2(Ff_x + G) + \mathbf{O}(\Delta t^3) \tag{2.107}$$

Haciendo uso de estas ecuaciones y realizando cálculos y operaciones matemáticas, los cuales no son el objetivo de este trabajo y por ello no se muestran, se llega a la expresión final del esquema de Runge-Kutta de cuarto orden:

$$x_{n+1} = x_n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{2.108}$$

$$k_1 = f(t_n, x_n) \tag{2.109}$$

$$k_2 = f(y_n + \frac{1}{2}\Delta t, x_n + \frac{1}{2}\Delta t k_1) \tag{2.110}$$

$$k_3 = f(y_n + \frac{1}{2}\Delta t, x_n + \frac{1}{2}\Delta t k_2) \tag{2.111}$$

$$k_4 = f(y_n + \Delta t, x_n + \Delta t k_3) \tag{2.112}$$

Las ecuaciones (2.108)-(2.112) se resuelven de forma iterativa para cada paso temporal hasta alcanzar el instante final de tiempo t y con ello se alcanza la solución de $\mathbf{x}$. Para un mejor entendimiento del esquema de Runge-Kutta de cuarto orden, se ilustra en la figura 2.5 como funciona la evaluación de un paso temporal.

Una de las ventajas del método de Runge Kutta es que la acción a realizar en cada paso es el análisis de la función $f(t_n, x_n)$, y que al realizar en un mismo subpaso varias evaluaciones, la precisión del método es mayor manteniendo un mismo paso temporal Δt.

Por otro lado, el método de Runge-Kutta requiere una evaluación del lado derecho de la ecuación diferencial en repetidas ocasiones, lo que hace que el tiempo de simulación y el coste del método sea mayor. Por tanto, el parámetro de control de este método simplemente sería el control sobre el paso temporal Δt.

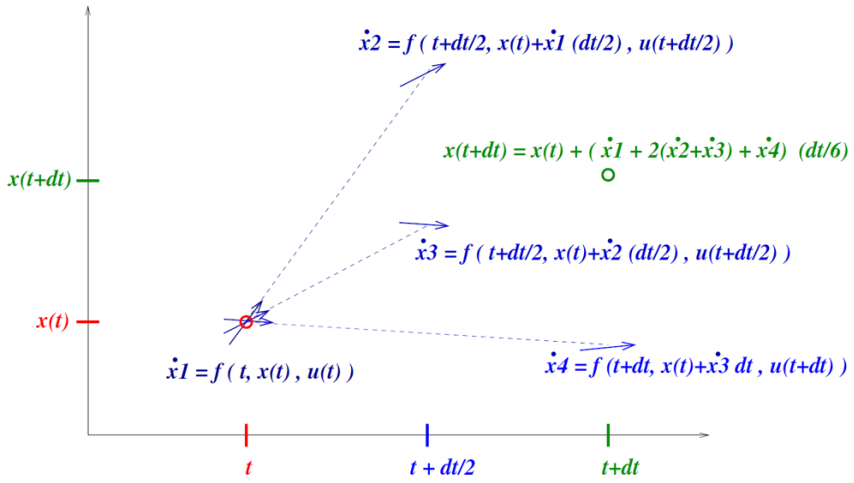

Figura 2.5: Evaluación del paso temporal del método RK4

Mediante este método se podría resolver el sistema de ecuaciones propuesto en (2.94) mediante el modelo de espacio de los estados. Se estaría resolviendo una ODE de primer orden y para ello se va a utilizar, para implementar el método, un programa de cálculo matématico (*Matlab*).

2.7. Estabilidad y precisión de los métodos de integración numérica

El objetivo del esquema de integración es obtener la respuesta del sistema dinámico de forma precisa computacionalmente sin necesidad de resolver analíticamente la EDO de segundo orden. Bien es sabido que la ecuación dinámica (1.2) en muchos casos donde se tienen pocos grados de libertad podría resolverse de forma sencilla a mano, pero en el momento que se incluyen muchos grados de libertad la solución de esta ecuación se complica y se hace necesario el uso de métodos directos de integración numérica. El uso de estos métodos requiere de gran precisión. Se puede entender como **precisión** de un método a la capacidad del mismo para que la solución que proporciona sea lo más parecida posible a la solución exacta del problema.

La manera de controlar la precisión de los métodos de integración es controlando el paso temporal Δt usado, ya que estos métodos lo que hacen es interpolar la función de los campos de desplazamientos, velocidad y aceleración en un intervalo de tiempo $[t,t+\Delta t]$. Este paso temporal deberá ser elegido de forma que se corresponda con el periodo más pequeño del sistema, lo cual puede significar que el Δt podría ser muy pequeño para cumplir los requerimientos de precisión.

Con lo anteriormente expuesto, resultaría razonable decir que, si el periodo más pequeño del sistema es T_{short}, el paso temporal requerido debería ser alrededor de 10 veces más pequeño ($T_{short}/10$). Sin embargo, en elementos finitos se busca que la respuesta asociada a los modos con las frecuencias más bajas sean predichas de forma precisa. Esto cambia el criterio de selección de paso temporal, de forma que se selecciona el periodo hasta el cual se querría obtener la respuesta con precisión y el paso temporal se seleccionaría de modo que sea unas 10 veces menor que ese periodo. Al hacer este cambio, se puede ver que se integra también de manera efectiva. Con este nuevo Δt, los modos cuyo paso temporal es más grande que la mitad del periodo natural T, tienen una respuesta indefinida que no se sabe si va a ser precisa o estable en el esquema utilizado. En esto que se acaba de explicar se basa la estabilidad de un esquema de integración. La **estabilidad** implica que para cualquier condición de un problema, usando un $\Delta t/T$ grande, la solución no deben amplificarse y por lo tanto no debe arrojar resultados imprecisos en la integración de las respuestas en los modos bajos. La estabilidad también implica que los errores que se van introduciendo en cada paso de la integración no tiendan a infinito y que se asegure la convergencia del esquema. La convergencia de un esquema de integración queda asegurada cuando al reducir el paso temporal Δt, la solución numérica tiende a la solución analítica. Por último, la integración será estable cuando se asegure un paso de tiempo lo suficientemente pequeño para integrar la respuesta del elemento con la frecuencia más alta del sistema (periodo más pequeño), sin divergencias en la respuesta.

Si se supone que se han obtenido las soluciones para todos los pasos

temporales (Δt, $2\Delta t$, ...) y qué solución va a obtenerse en el paso temporal siguiente, entonces, para cualquiera sea el método utilizado, se tiene una fórmula que se puede usar recursivamente y que toma la forma de:

$$\hat{\mathbf{U}}^{\Delta t+t} = \mathbf{S} \cdot \hat{\mathbf{U}}^{t} + \mathbf{L}(\mathbf{r}^{t+v}) \tag{2.113}$$

En esta fórmula se almacenan los resultados en los vectores $\mathbf{U}$ de los distintos campos en los pasos temporales y $\mathbf{r}$ es un vector donde se encuentran las fuerzas aplicadas. Con ello aparecen las matrices $\mathbf{S}$ y $\mathbf{L}$ denominadas respectivamente operadores de aproximación y de cargas. La matriz que va a resultar interesante a la hora de analizar la estabilidad de los esquemas es la $\mathbf{S}$, la cual se forma de un modo distinto para cada sistema de integración que se esté analizando [8]. Además se va a tener en cuenta que el amortiguamiento del problema es nulo asique $\xi = 0$.

Como la estabilidad debe examinarse cualquiera sean las condiciones iniciales del problema, en la ecuación (2.113) se va a considerar que el vector de cargas es $\mathbf{r}$=0, y por lo tanto, la ecuación que queda es:

$$\hat{\mathbf{U}}^{\Delta t+t} = \mathbf{S} \cdot \hat{\mathbf{U}}^{t} \tag{2.114}$$

La matriz $\mathbf{S}$ puede descomponerse espectralmente como $\mathbf{S} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1}$, donde $\mathbf{V}$ es una matriz que contiene los autovectores de $\mathbf{S}$ y $\mathbf{D}$ es una matriz diagonal con los autovalores de $\mathbf{S}$. Una vez definida para cada esquema de integración, se puede determinar la estabilidad. Para ello se define el radio espectral como $\rho(\mathbf{S})$ donde:

$$\rho(\mathbf{S}) = max|\lambda_i| \qquad i = 1, 2, ... \tag{2.115}$$

Es decir, se obtiene con el máximo valor absoluto de los autovalores de la matriz $\mathbf{S}$ que estos, además, estarán situados en el plano complejo. Habiendo obtenido el valor del radio espectral la condición de estabilidad se define como:

- $\rho(\mathbf{S})$ debe ser menor o igual que 1.

- Que todos los módulos de los autovalores de **S** sean menores que 1.

Es decir, en el momento que haya un autovalor mayor que 1, la condición de estabilidad no se cumple y el esquema pasa a ser inestable. El objetivo es plantear esquemas de integración para sea cual sea el paso temporal utilizado, sea estable. Si esto no es posible, deberá ser un esquema de integración que tenga una amplitud de estabilidad en términos de paso temporal muy elevada.

Otro aspecto para evaluar la precisión del método es el error asociado al mismo. Los errores pueden introducirse debido al redondeo y al truncamiento.

- Los errores de redondeo vienen dados por el número de decimales que guarde el programa de cálculo con el que se esté implementando el método (punto flotante).
- Los errores de truncamiento son los debidos a la eliminación de términos de las series con los que se realizan las aproximaciones de términos en la discretización de los esquemas.

Habiendo analizado la estabilidad del método de integración directa utilizado, se podría decir que los sistemas incondicionalmente estables son los mejores. Esto se debe a que sea cual sea el paso temporal usado la solución va a converger al resultado. Esto es verdad, pero se debe tener en cuenta otro aspecto importante en estos sistemas, la precisión del método utilizado. Por mucha libertad que exista al elegir el paso temporal, este debe ser elegido de forma que se aproxime a la solución de forma precisa con el menor error posible para obtener unos resultados realistas. Por ello, se va a realiza el estudio de la estabilidad, precisión/error de los esquemas expuestos anteriormente.

Primero se ha de hallar la matriz **S** de cada método. En primer lugar, se va a analizar la estabilidad del método Newmark-β. Tras analizar la matriz $\mathbf{S}_{Newmark}$ se puede establecer una condición de estabilidad incondicional para este método, que es:

$$2\beta \geq \gamma \geq \frac{1}{2} \tag{2.116}$$

Mientras que los valores de β y γ cumplan el requisito de la ecuación (2.116), la respuesta del sistema será estable. En la figura 2.6 se muestran distintos radios espectrales para distintos valores de β y γ:

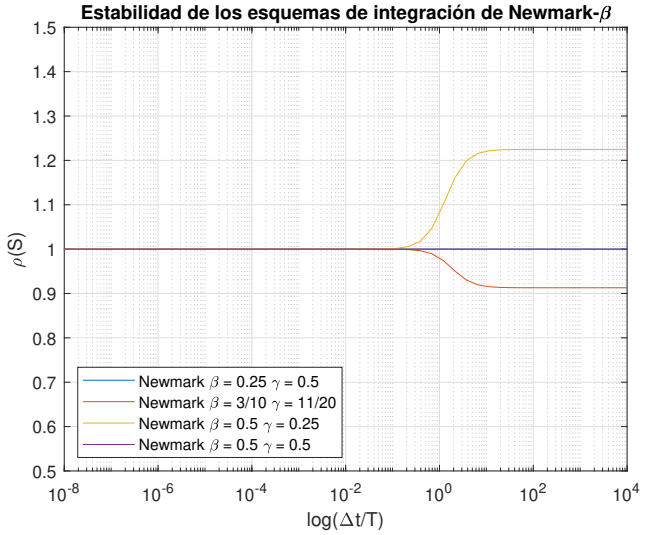

Figura 2.6: Estabilidad del esquema Newmark-β

En la figura 2.6 se puede observar como todos los esquemas propuestos, menos el que está definido por $\beta = 0.5$ y $\gamma = 0.25$, cumplen el requisito de estabilidad incondicional y son estables. Sin embargo, el que se ha mencionado no los cumple. Sólamente cumple con el requisito de estabilidad para un cierto rango de valores del paso temporal Δt.

En segundo lugar, se va a estudiar la estabilidad del esquema de Bathe junto con la estabilidad del método de Bathe modificado, modificando los valores de β_1 y β_2 y manteniendo constante el parámetro del subpaso temporal θ en 0.5. El resultado se puede observar en las figuras 2.7 y 2.8.

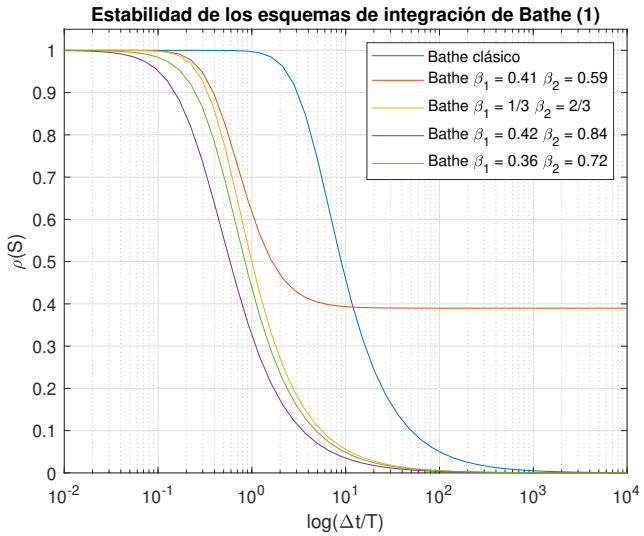

Figura 2.7: Estabilidad de los esquemas Bathe y Bathe modificado

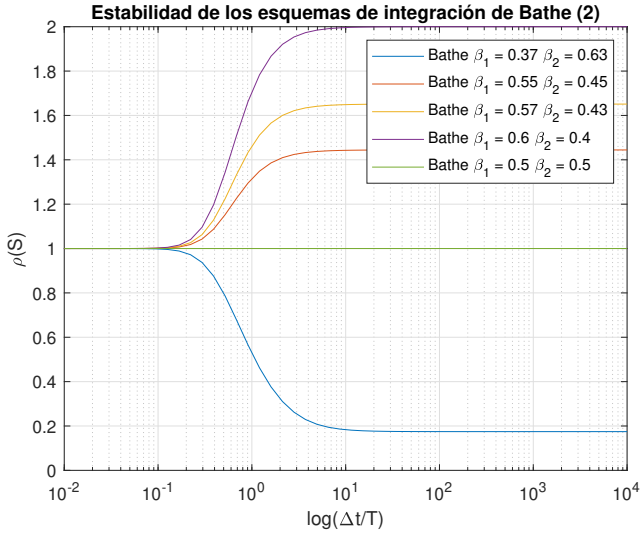

Figura 2.8: Estabilidad del esquema Bathe modificado

Se puede ver como el esquema de Bathe es un esquema condicionalmente estable puesto que para valores de $\beta_1 \geq \beta_2$, el sistema a partir de un valor de paso temporal se vuelve inestable. Sin embargo, para todo valor correspondiente a $\beta_1 = 1/3$ hasta $\beta_1 = 0{,}5$ con $\beta_2 = 1 - \beta_1$, el sistema es incondicionalmente estable. De la misma manera que para todo valor corres-

pondiente a $\beta_1 = 1/3$ hasta infinito con $\beta_2 = 2\beta_1$.

En el esquema de integración PIM, el paso crítico es el cálculo de la matriz exponencial **T**. El error de precisión en este método se produce principalmente por la truncación de la serie de Taylor en el término de orden 5. Teniendo en cuenta que en el caso de tener N=20 se obtendría m=1048576, y por lo tanto, significaría que el paso temporal inicial quedaría dividido m veces. Esto hace que la resolución del método sea muy potente y, sumado a la truncación realizada, permite que el método sea estable y preciso para la mayoría de pasos temporales. Esto se puede ver en la figura 2.9.

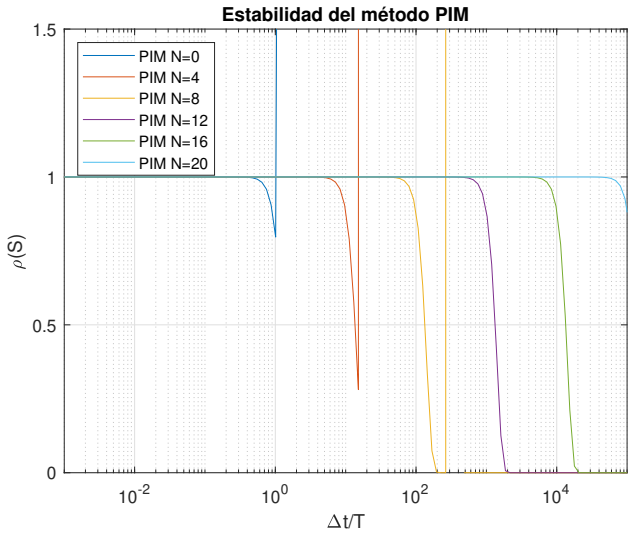

Figura 2.9: Región de estabilidad del esquema *PIM*.

En la figura 2.9 se observa como el rango de pasos temporales donde la solución es estable, aumenta con N como era de esperar. Se plantea un esquema condicionalmente estable pero que a efectos prácticos se puede considerar como incondicionalmente estable y de alta precisión comparado con el de Newmark-β, ya que este último trunca las aproximaciones en el tercer término. Por último, otra razón por la que el método es áltamente preciso es debido a que el sistema siempre tiene en cuenta la parte incremental y con ello evita los problemas que sí que ocurren en otros métodos de Runge-

Kutta, donde la solución tiene una fuerte dependencia de la precisión del vector solución en cada paso temporal, es decir, cambios pequeños en la solución pueden generar resultados erróneos en los siguientes pasos temporales.

Para el esquema de integración de Runge-Kutta de cuarto orden, su región de estabilidad se define por:

$$|R(\mu)| \leq 1 \tag{2.117}$$

Donde R es una función de estabilidad que depende del orden del esquema usado. Para RK4 toma el valor de:

$$R(\mu) = 1 + \mu + \frac{1}{2}\mu^2 + \frac{1}{6}\mu^3 + \frac{1}{24}\mu^4 \tag{2.118}$$

Si se grafica, se obtiene la región de estabilidad que se observa en la figura 2.10.

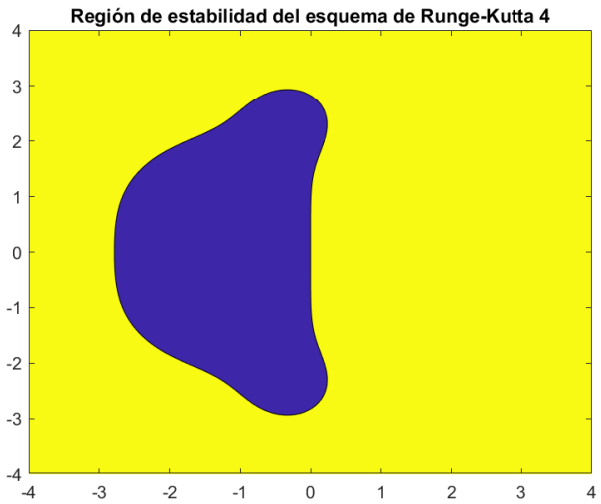

Figura 2.10: Región de estabilidad del esquema RK4

La región compleja es la representada por todos los valores de semiplano vertical menores que 0. La primera conclusión que se puede sacar del gráfico es que el esquema va a ser estable si $\lambda\Delta t$, donde λ representa los autovalores

del problema, cae en la región de estabilidad que se distingue por el color morado. Por ello, habrá que analizar la estabilidad para cada caso y con ello, se podrá obtener el paso temporal óptimo para cada problema.

Aunque podría pensarse que la estabilidad depende únicamente de los autovalores del problema λ, la realidad es que tiene una fuerte dependencia con Δt. De hecho, en los esquemas de Runge-Kutta de cuarto orden, la estabilidad está fuertemente reestringida por el paso temporal.

Capítulo 3

Resultados y comparación de los métodos de integración numérica temporal en un caso genérico

Una vez se han expuesto los distintos métodos de integración que se van a utilizar en el trabajo, en este capítulo se van a mostrar los resultados que arrojan los mismos aplicados a un sistema dinámico bajo la acción de distintos tipos de carga. Se van a utilizar para cada esquema los conceptos de precisión y estabilidad ya explicados en el Capítulo 2 para la elección de los esquemas con los parámetros asociados a los mismos.

El objetivo principal será ver como se comportan los distintos esquemas ante la resolución del sistema y para ello se compararán entre sí. Además, se va a hacer uso de la resolución exacta del problema para ver, no sólo como es el comportamiento de cada esquema ante el sistema, si no también para ver como se comportan respecto a la solución exacta del mismo. Por ello, se comparará el error de ambos sistemas y se tratará de obtener y demostrar los puntos fuertes y débiles de cada esquema, algunos de los cuales ya han sido mencionados en el Capítulo 2.

Para ello se va a definir un sistema dinámico sencillo de dos grados de

libertad con el cual se pueden obtener casi todos los parámetros necesarios para comparar los esquemas de integración numérica utilizados. El diagrama de bloques del sistema puede observarse en la figura 3.1, y el valor de las constantes asocidadas al mismo para la construcción de la ecuación dinámica se pueden ver en la Tabla 3.1. En este caso no se va a introducir disipación en el sistema en forma de amortiguamiento ya que complica el cálculo computacional y las ecuaciones, y las conclusiones a las que se van a llegar son idénticas. De cualquier forma, los *scripts* creados tienen la capacidad de resolver problemas con disipación energética y prueba de ello será el caso práctico a resolver en el siguiente capítulo.

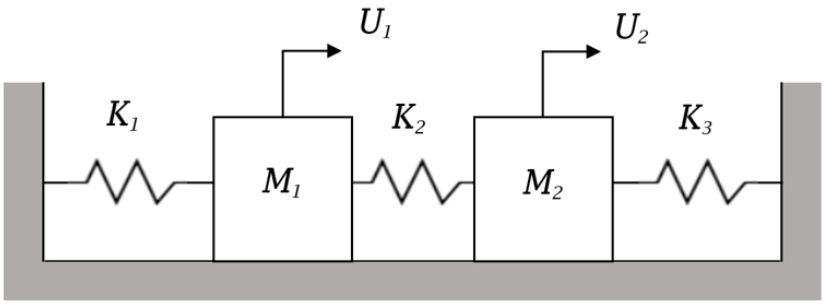

Figura 3.1: Diagrama de bloques del sistema dinámico utilizado.

Rigideces			**Masa**	
$\mathbf{K}_1$	$\mathbf{K}_2$	$\mathbf{K}_3$	$\mathbf{M}_1$	$\mathbf{M}_2$
6	4	2	1	2

Tabla 3.1: Datos del sistema dinámico a estudiar.

Se va a considerar que las vibraciones del sistema son forzadas aplicando un tipo de caso de carga para ver como se comportan los sistemas de integración. La carga aplicada será variable con el tiempo y de carácter aleatoria. La excitación dependiente del tiempo se calculará por medio de una fuerza aleatoria donde para su definición se divide el paso temporal para representarla con mayor resolución. Se seleccionan distintas frecuencias para su definición y, por último, se afecta con una gaussiana para obtener la fuerza que se representa en la figura 3.2.

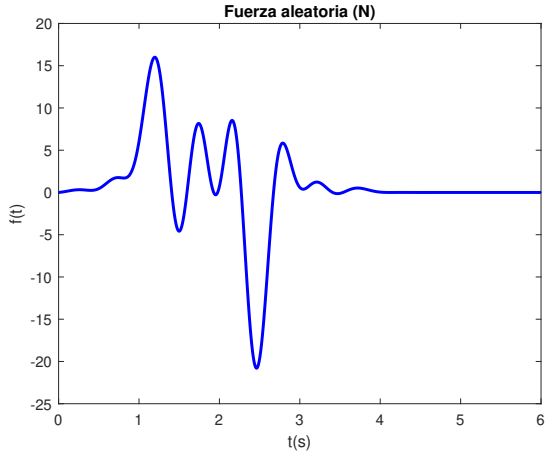

Figura 3.2: Fuerza aleatoria gaussiana aplicada en el primer nodo.

El sistema dado por los datos de la Tabla 3.1 se puede representar matricialmente como:

$$\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{U}}_1 \\ \ddot{\mathbf{U}}_2 \end{bmatrix} + \begin{bmatrix} 10 & -4 \\ -4 & 6 \end{bmatrix} \begin{bmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \end{bmatrix} = \begin{bmatrix} f_1(t) \\ f_2(t) \end{bmatrix} \tag{3.1}$$

Habiendo definido el problema se va a resolver el mismo mediante el primer esquema de integración propuesto, el esquema de Newmark-β. La respuesta de este sistema ante una fuerza de carácter aleatorio y dependiente del tiempo y los errores asociados a la respuesta exacta del sistema se puede ver en las figuras 3.3-3.13.

Para mejor entendimiento del lector, cuando se utiliza la notación 'Newmark-0.25-0.5' se refiere a que se está utilizando el esquema Newmark con $\beta = 0{,}25$ y $\gamma = 0{,}5$. Se ve claramente como la respuesta cuando se utiliza un Δt elevado difiere sustancialmente de la respuesta en el caso exacto. La condición del esquema de Newmark-β de estabilidad incondicional con los parámetros utilizados nos asegura que la respuesta converge. En la figuras asociadas a los errores, se ve como estos decrecen con el paso temporal, teniendo errores muy cercanos al 0 % para el esquema utilizado con $\Delta t = 0.01$s. Se observan algunos errores muy elevados debidos a que el orden de magnitud del valor

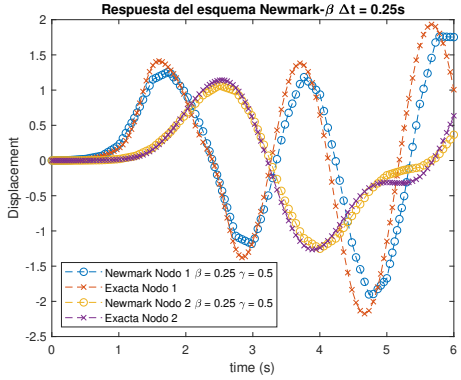

Figura 3.3: Newmark-0.25-0.5 $\Delta t = 0.25$s

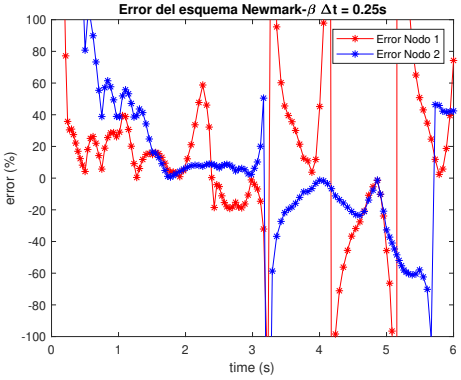

Figura 3.4: Error Newmark-0.25-0.5 $\Delta t = 0.25$s

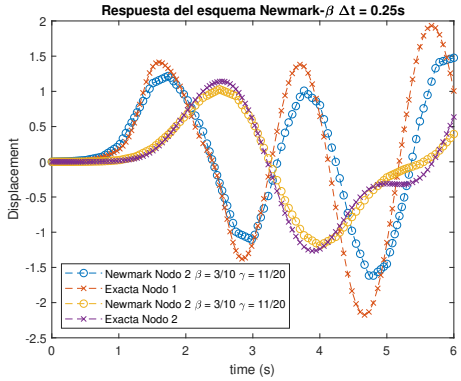

Figura 3.5: Newmark-3/10-11/20 $\Delta t = 0.25$s

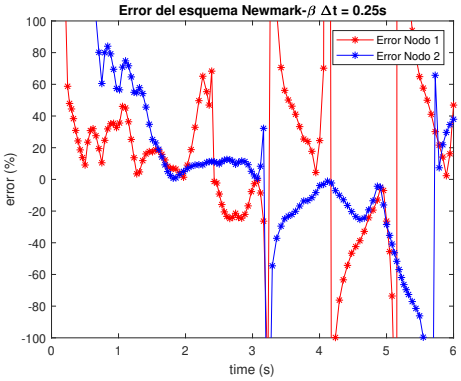

Figura 3.6: Error Newmark-3/10-11/20 $\Delta t = 0.25$s

real es muy pequeño y pequeñas variaciones en la solución del esquema generan esos errores.

En cuanto a la diferencia del uso del esquema con $\beta = 0{,}25$ y $\gamma = 0{,}5$ con $\beta = 3/10$ y $\gamma = 11/20$ se observa como para un mismo paso temporal, la respuesta obtenida con el método 'Newmark-0.25-0.5' para este sistema, es ligeramente más precisa teniendo el mismo coste computacional.

Uno de los incovenientes de los esquema de Newmark-β es que en sistemas dinámicos en los cuales las diferencias entre las frecuencias más elevadas

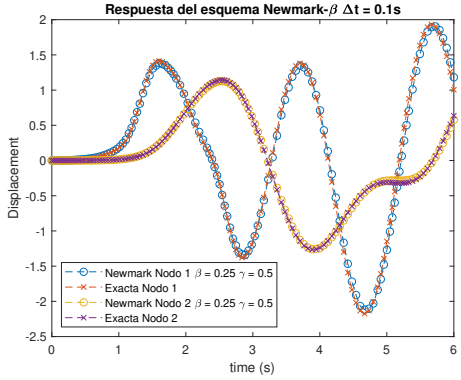

Figura 3.7: Newmark-0.25-0.5 $\Delta t = 0.1$s

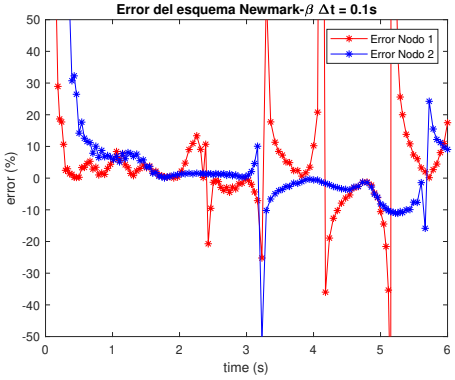

Figura 3.8: Error Newmark-0.25-0.5 $\Delta t = 0.1$s

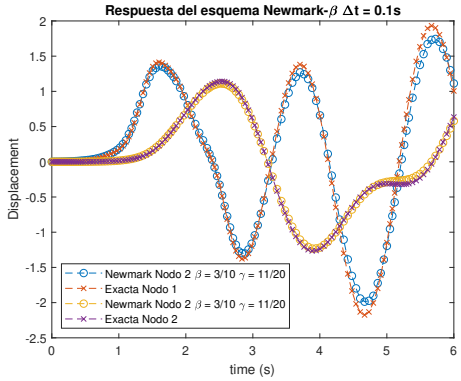

Figura 3.9: Newmark-3/10-11/20 $\Delta t = 0.1$s

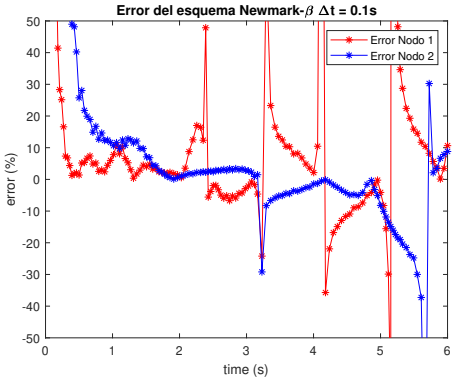

Figura 3.10: Error Newmark-3/10-11/20 $\Delta t = 0.1$s

y más pequeñas son muy altas, el esquema proporciona mucha disipación numérica y por tanto no es capaz de converger a la respuesta real. Esto ocurre sobre todo usando la regla del Trapecio y si se usan otros parámetros en el esquema se puede llegar a disminuir. A pesar de ello, el uso de este esquema no es recomendable en aquellos sistemas que cumplan los requisitos de frecuencias (periodos) antes expuestos.

Los siguientes esquemas a analizar van a ser el esquema de Bathe y el esquema de Bathe modificado usando distintos parámetros. Los resultados de los esquemas se pueden ver en las figuras 3.15 - 3.26.

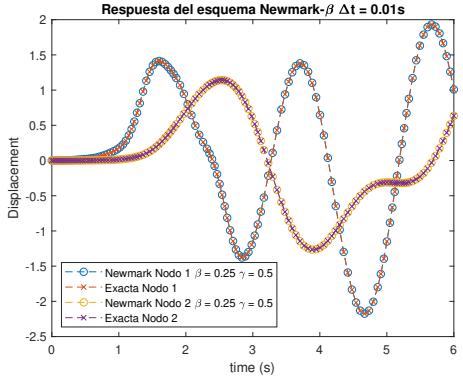

Figura 3.11: Newmark-0.25-0.5 $\Delta t = 0.01$s

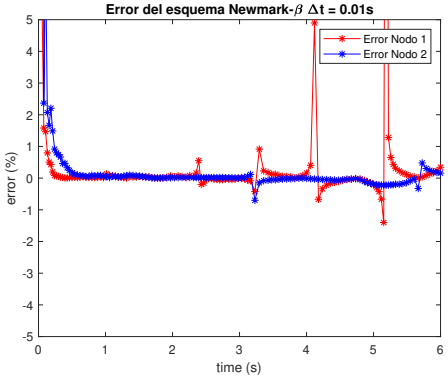

Figura 3.12: Error Newmark-0.25-0.5 $\Delta t = 0.01$s

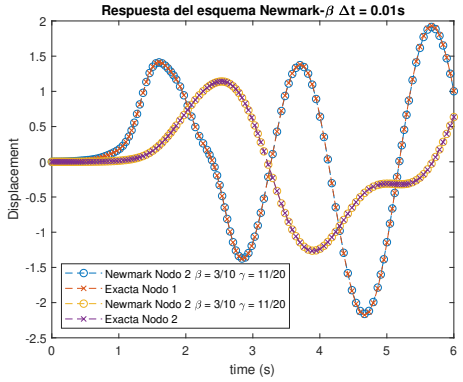

Figura 3.13: Newmark-3/10-11/20 $\Delta t = 0.01$s

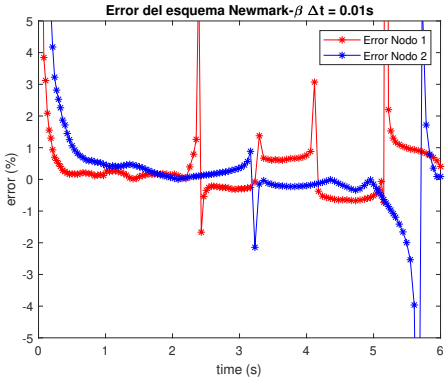

Figura 3.14: Error Newmark-3/10-11/20 $\Delta t = 0.01$s

De los esquemas de Bathe se pueden extraer conclusiones interesantes. Para mejor entendimiento del lector se va a denominar como Bathe-0.5-0.5-0.5 al esquema de Bathe modificado con θ=0.5, β_1=0.5 y β_2=0.5.

En primer lugar, se observa como al igual que en el método de Newmark-β, la respuesta converge a la solución exacta a medida que se disminuye el paso temporal. Cuando Δt=0.25s se obtienen unos errores en la solución que, aunque son bajos, en algunos casos se elevan demasiado (50 %). Se ve también una pérdida de amplitud de la respuesta y de fase propia de estos esquemas cuando el paso temporal, como es el caso, es elevado. Para ese paso

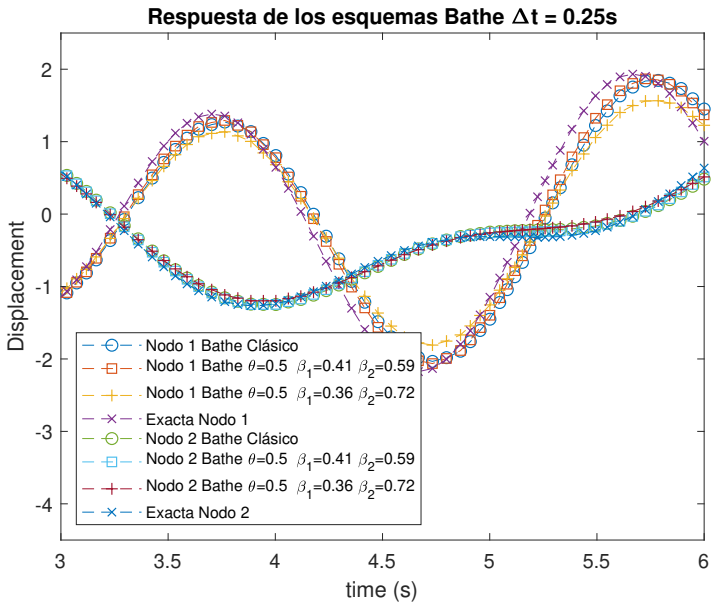

Figura 3.15: Respuesta mediante Bathe (1) $\Delta t = 0.25$s

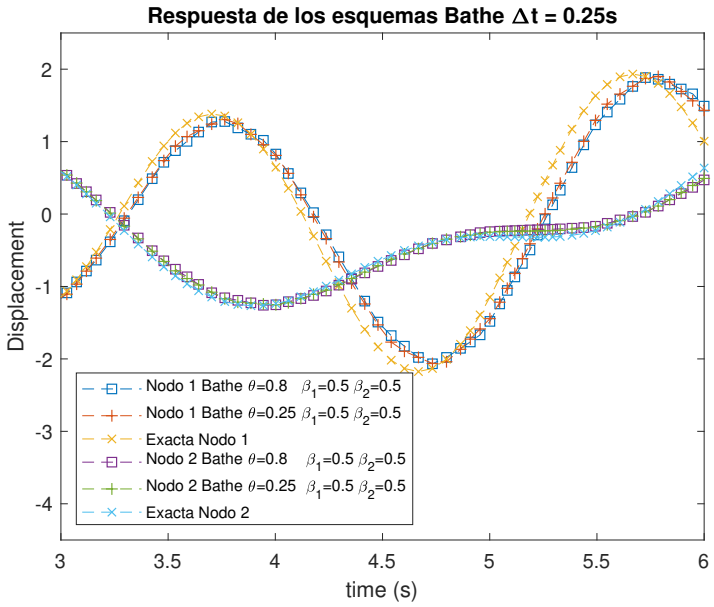

Figura 3.16: Respuesta mediante Bathe (2) $\Delta t = 0.25$s

temporal se observa como en general todos los esquemas presentan un error similar, siendo el caso de los esquemas Bathe0.5-0.41-0.59 y Bathe0.25-0.5-

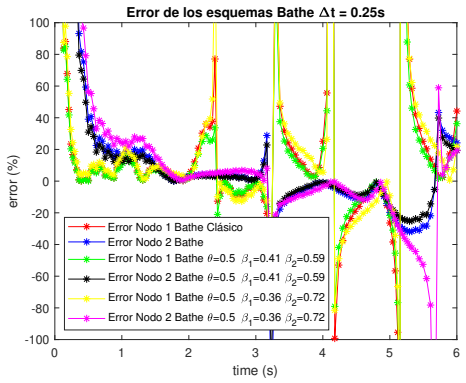

Figura 3.17: Error Bathe (1) $\Delta t = 0.25$s

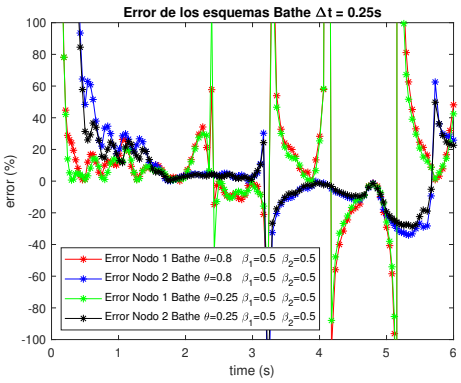

Figura 3.18: Error Bathe (2) $\Delta t = 0.25$s

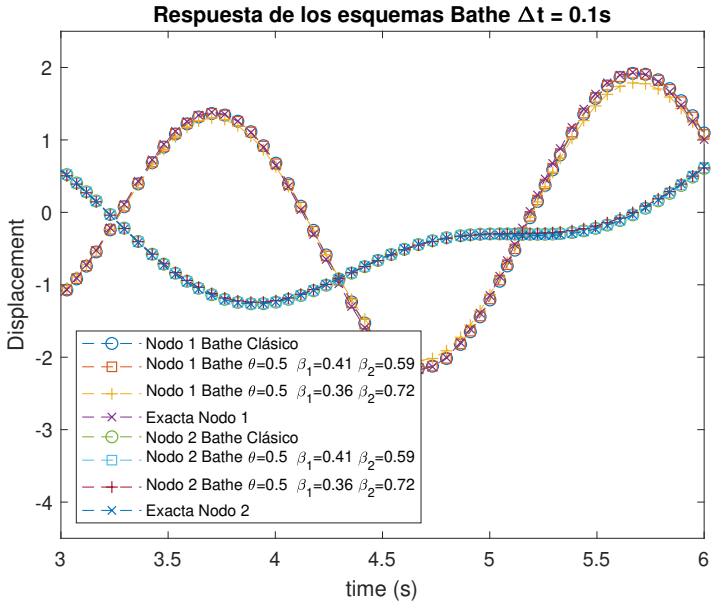

Figura 3.19: Respuesta mediante Bathe (1) $\Delta t = 0.1$s

0.5 los que tienen menor error asociado. De cualquier modo estos errores son muy elevados por lo que hay que recurrir a disminuir el paso temporal.

Para un Δt=0.1s, se ve como los errores en la solución disminuyen de manera considerable, estando los más altos sobre el 30 % sin contar algunos puntos concretos del inicio como se vió en la resolución mediante Newmark.

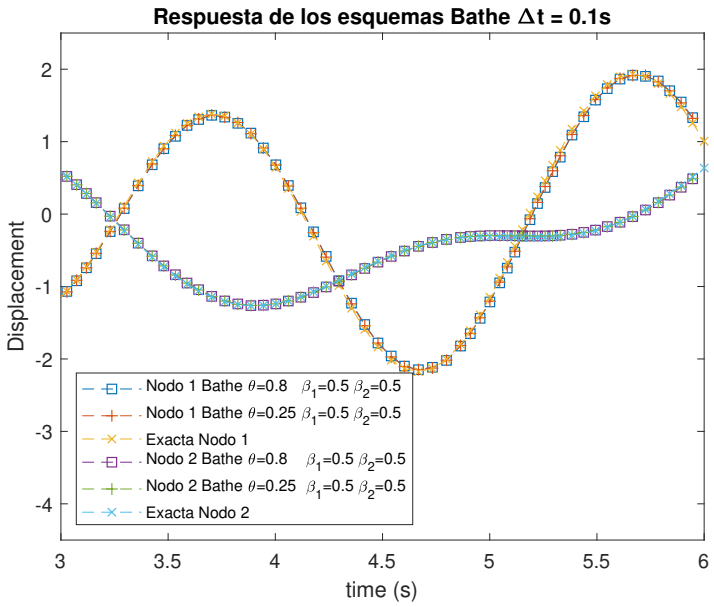

Figura 3.20: Respuesta mediante Bathe (2) $\Delta t = 0.1$s

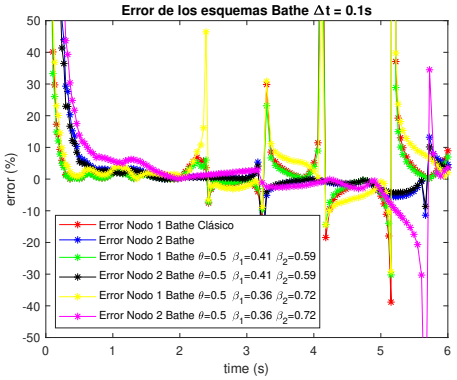

Figura 3.21: Error Bathe (1) $\Delta t = 0.1$s

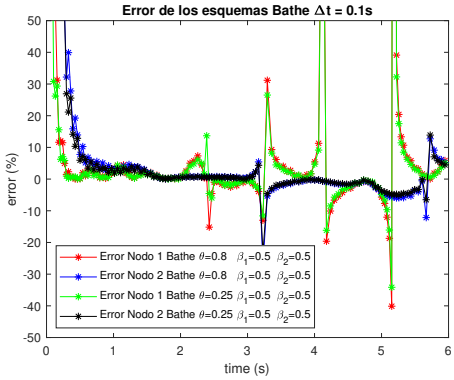

Figura 3.22: Error Bathe (2) $\Delta t = 0.1$s

Además, comparando con Newmark se ha conseguido reducir para el mismo paso temporal el error de manera considerable, a costa de un mayor coste computacional. Para este paso temporal se puede concluir una cosa, y es-que el esquema de Bathe-0.25-0.5-0.5 devuelve una mejor respuesta que el esquema de Bathe-0.8-0.25-0.25. En pincipio no debería ocurrir, porque en el esquema en el que se utiliza θ=0.25 con el método de Bathe modificado,

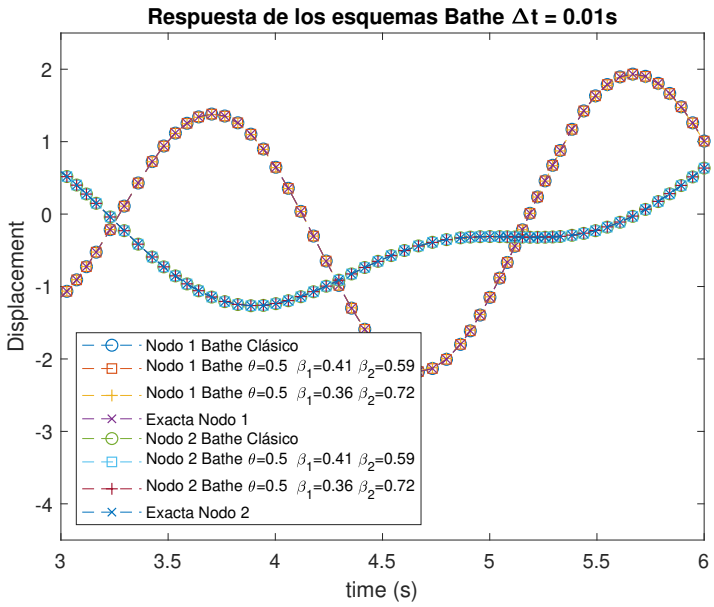

Figura 3.23: Respuesta mediante Bathe (1) $\Delta t = 0.01$s

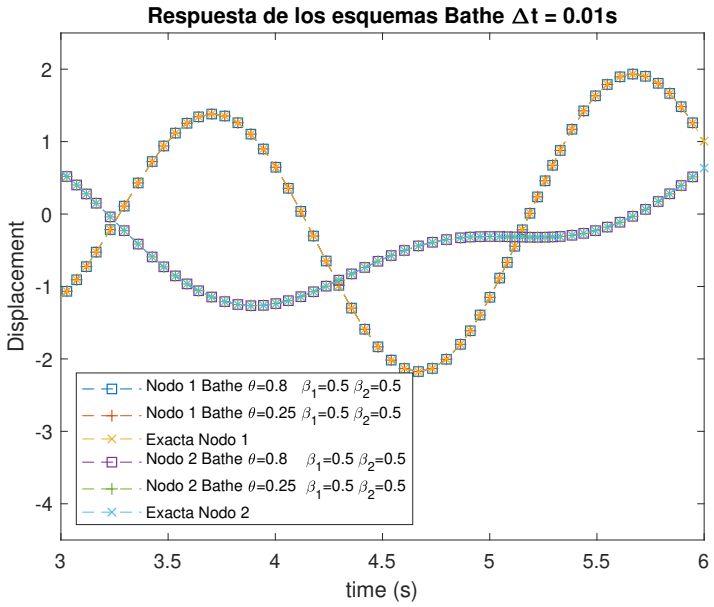

Figura 3.24: Respuesta mediante Bathe (2) $\Delta t = 0.01$s

se está resolviendo durante un 75 % del paso temporal el esquema mediante un esquema Newmark con la regla del trapecio. El 25 % del paso temporal

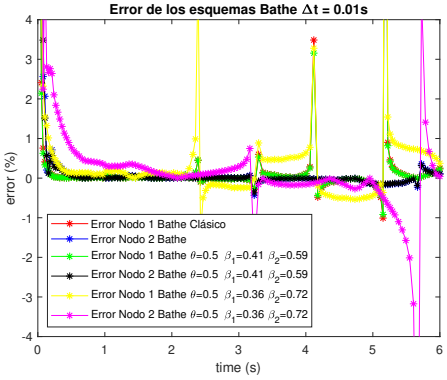

Figura 3.25: Error Bathe (1)
$\Delta t = 0.01$s

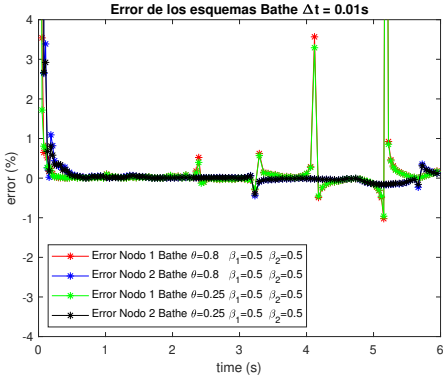

Figura 3.26: Error Bathe (2)
$\Delta t = 0.01$s

restante se resuelve mediante la regla del trapecio como se ha visto. En el esquema con θ=0.8, se resuelven el paso temporal entero con la regla del trapecio de Newmark por lo que debería obtenerse la misma precisión aunque por el caso utilizado es más preciso usando un valor de θ=0.25. Esto se observa en la figura 3.22.

Por último, usando un paso temporal muy pequeño como es Δt=0.01s, se observa como la solución exacta y la estimada prácticamente son las mismas. En las figuras 3.25 3.26 se puede ver el orden de magnitud de los errores. Se confirma como el esquema Bathe-0.25-0.5-0.5 tiene una mayor precisión, confirmando el buen comportamiento del método de Newmark-β y errores que apenas superan el 2 %. También se puede ver como el esquema de Bathe clásico tiene una buena respuesta comparado con los métodos de Bathe modificado con control de disipación numérica, donde con el método de Bathe-0.5-0.41-0.59 se consigue un error que apenas supera el 1 %, teniendo el mismo coste computacional que el resto de los métodos.

En siguiente lugar, se va a analizar el esquema de *Precise Integration Method* (PIM) para distintos valores del parámetro *N*. Los resultados de los esquemas se pueden ver en las figuras 3.27 - 3.35.

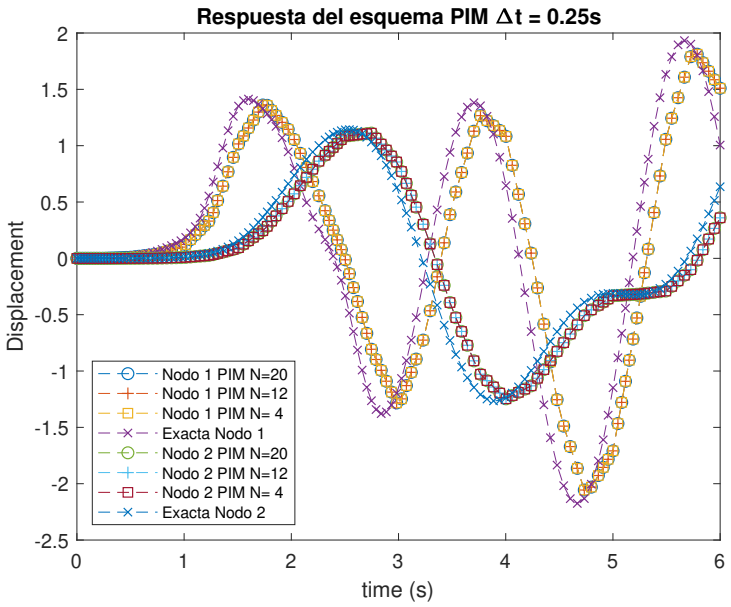

Figura 3.27: Respuesta mediante PIM $\Delta t = 0.25$s

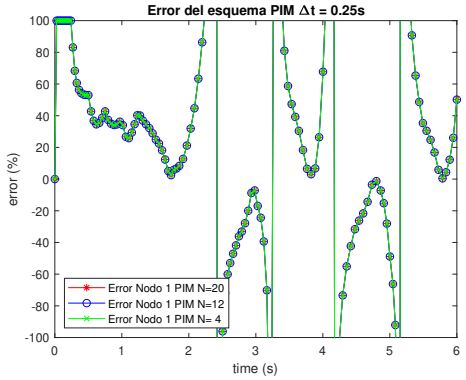

Figura 3.28: Error PIM nodo 1 $\Delta t = 0.25$s

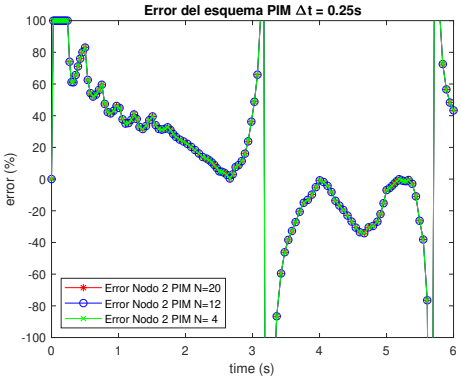

Figura 3.29: Error PIM nodo 2 $\Delta t = 0.25$s

De las figuras asociadas al PIM, se pueden extraer las siguientes conclusiones. El parámetro N utilizado no influye en la precisión de la respuesta del sistema, como puede verse, si no simplemente en la estabilidad del método. Como se vió en la sección de estabilidad, al aumentar el parámetro N, se ampliaba el rango de pasos temporales que hacían que la solución fuese estable.

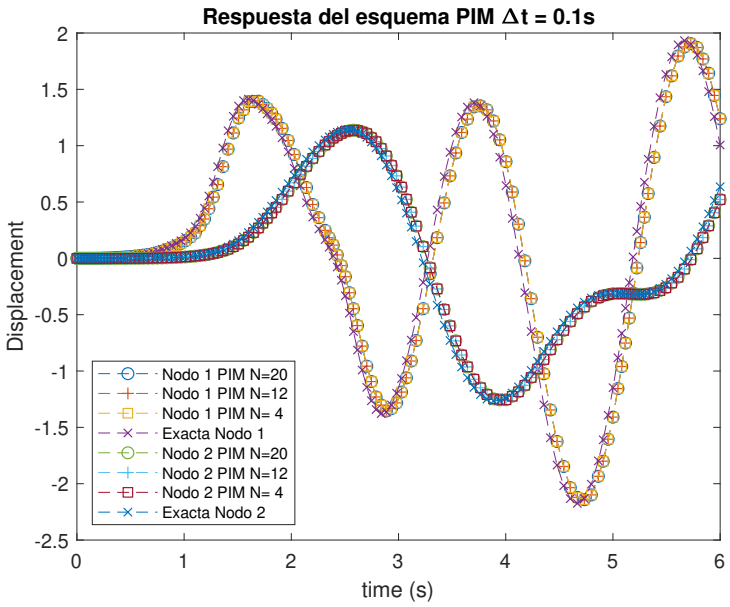

Figura 3.30: Respuesta mediante PIM $\Delta t = 0.1$s

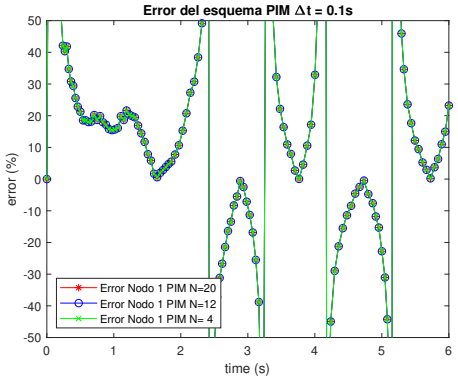

Figura 3.31: Error PIM nodo 1 $\Delta t = 0.1$s

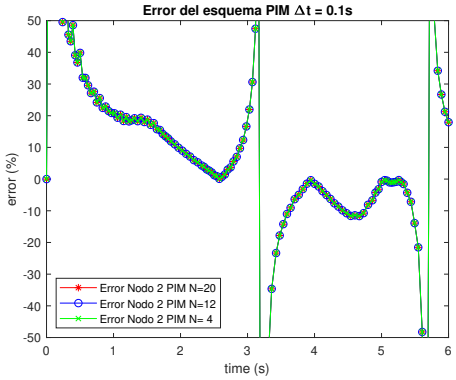

Figura 3.32: Error PIM nodo 2 $\Delta t = 0.1$s

Respecto de la respuesta, como en otros métodos, el método se va aproximando a la solución real mientras se reduce el valor del paso temporal. Cuando se usa el paso temporal de Δt=0.25s, se ve como la solución, al igual que en el método de Bathe, está algo desfasada respecto a la solución real y se tienen errores elevados con algo menos de disipación numérica que en los

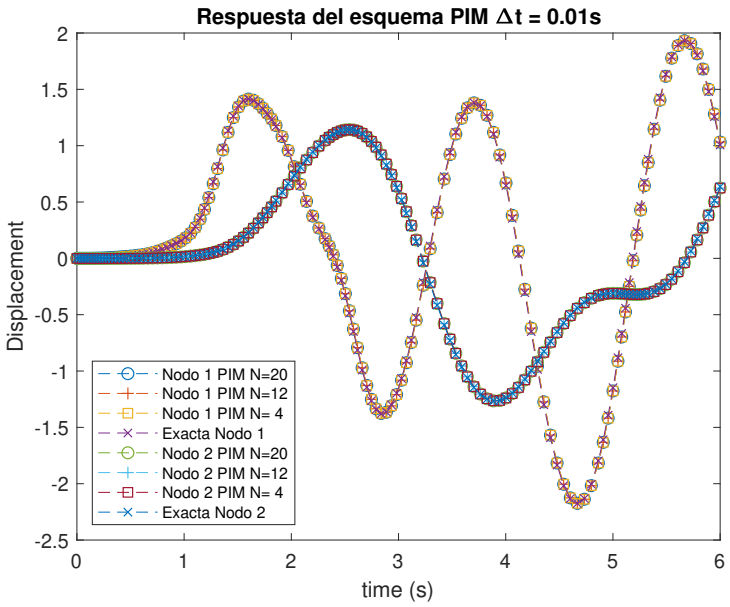

Figura 3.33: Respuesta mediante PIM $\Delta t = 0.01$s

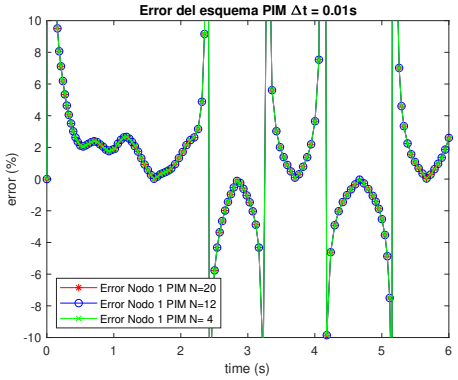

Figura 3.34: Error PIM nodo 1 $\Delta t = 0.01$s

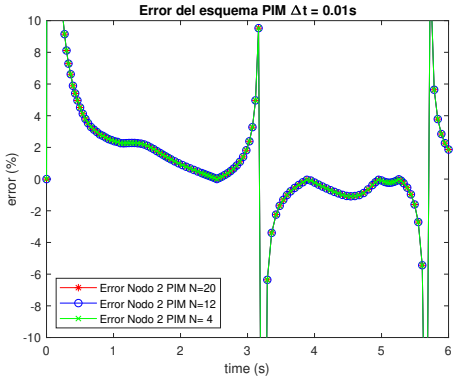

Figura 3.35: Error PIM nodo 2 $\Delta t = 0.01$s

métodos de Newmark-β y Bathe.

Para un paso temporal Δt=0.1s, la respuesta mejora sustancialmente y los errores se reducen al margen de 15 % - 25 % con un coste computacional razonable. Esto es debido a que la matriz exponencial solo hay que computarla una vez y a que el esquema se basa en la iteración de la solución para

cada paso temporal.

Para el paso temporal de Δt=0.01s, se tiene una respuesta semejante a la obtenida exacta. Los errores se encuentran entorno al 5 %, aunque en este caso para el nodo 1 se ve como hay bastantes picos de error que están directamente asociados a la elevada pendiente que tiene la respuesta y además los valores pequeños que se manejan cuando la solución se acerca al valor 0, generando que pequeñas variaciones en la solución produzcan esos errores tan elevados.

Por último, se va a analizar el esquema de Runge-Kutta de cuarto orden. Los resultados de los esquemas se pueden ver en las figuras 3.36 - 3.41.

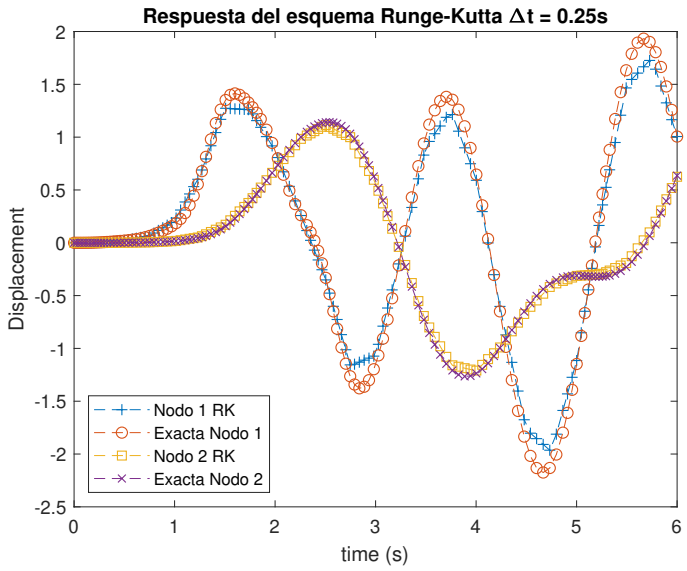

Figura 3.36: Respuesta mediante Runge-Kutta cuarto orden $\Delta t = 0.25$s

Respecto a los resultados que arroja el esquema de Runge-Kutta de cuarto orden, se puede ver como para todos los pasos temporales usados es el esquema que mejores resultados presenta. Esto es debido como se ha comentado en la definición del esquema, a las múltiples evaluaciones de la solución que hace dentro de un mismo paso temporal. El inconveniente es que, aunque la resolución del problema sea mejor (más preciso), computacionalmente es

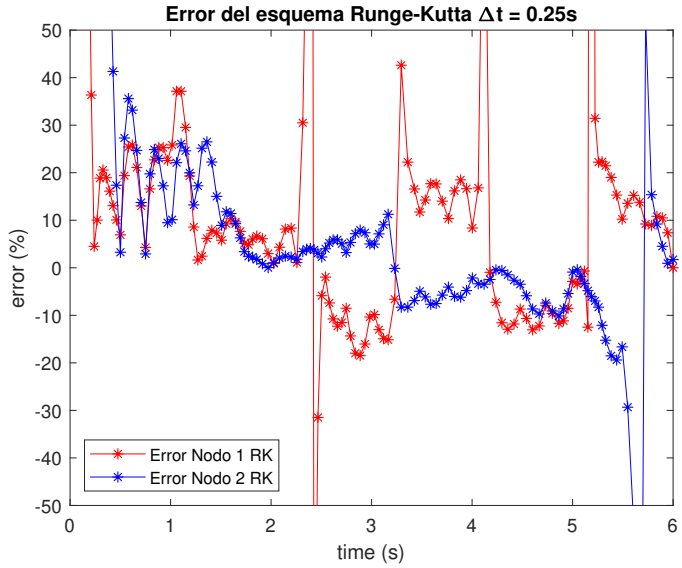

Figura 3.37: Error del esquema de Runge-Kutta cuarto orden $\Delta t = 0.25$s

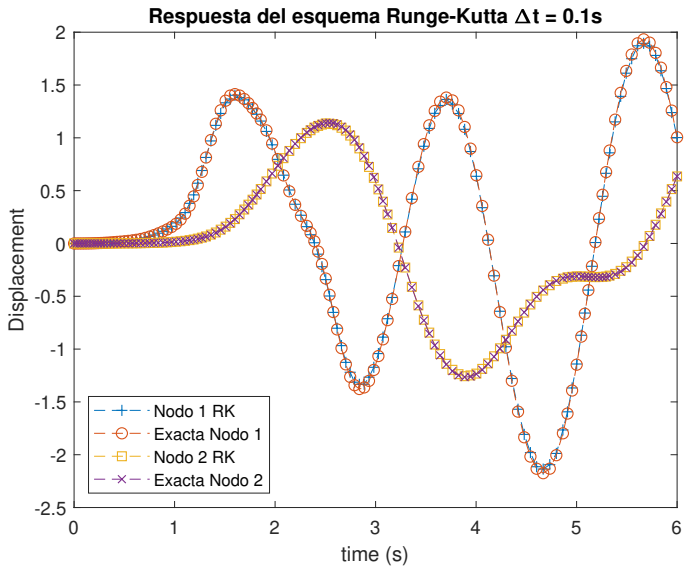

Figura 3.38: Respuesta mediante Runge-Kutta cuarto orden $\Delta t = 0.1$s

un método caro.

Los errores asociados a la respuesta mediante Runge-Kutta, para un paso temporal de Δt=0.1s, se encuentran alrededor del 5 % en la mayoría de los

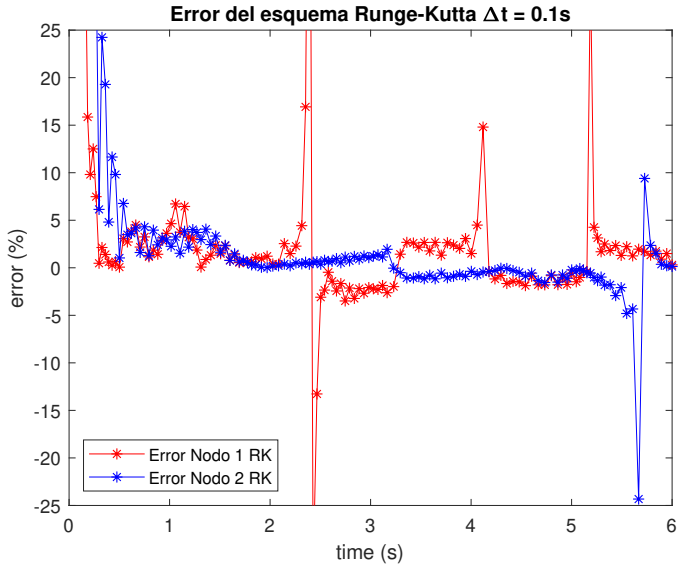

Figura 3.39: Error del esquema de Runge-Kutta cuarto orden $\Delta t = 0.1$s

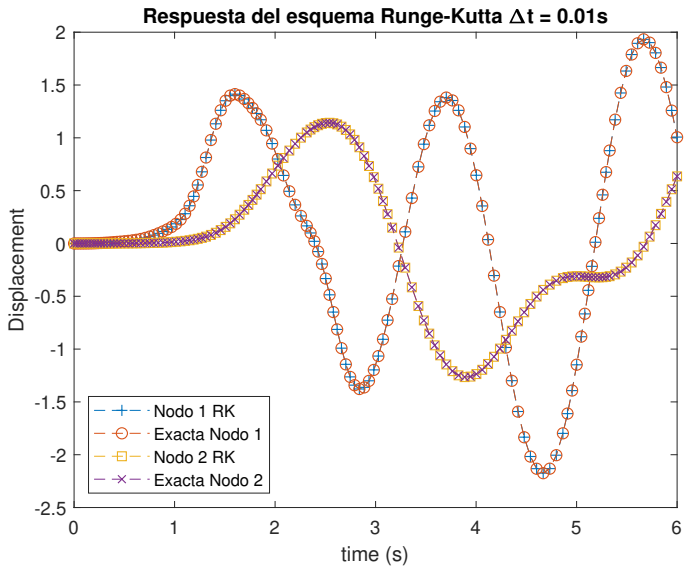

Figura 3.40: Respuesta mediante Runge-Kutta cuarto orden $\Delta t = 0.01$s

puntos. Si estos datos se comparan con a los esquemas de Newmark-β, Bathe, Bathe modificado y PIM, hay una diferencia en el error pequeña (los picos de error que sí que se producián en estos esquemas son menores cuantitativamente).

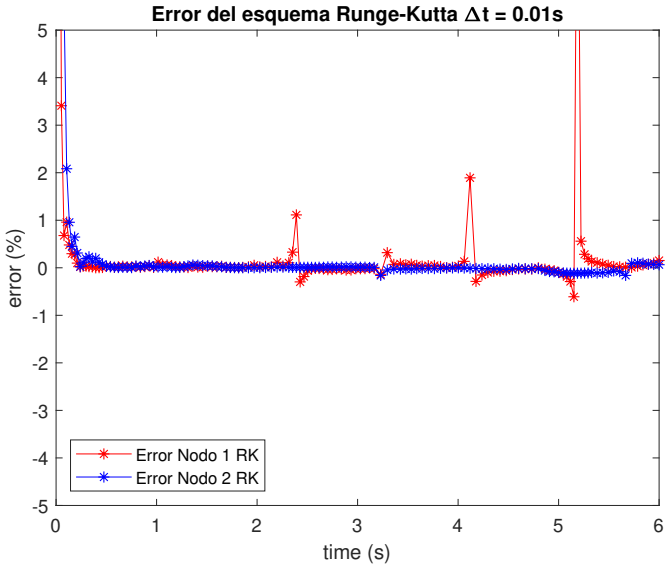

Figura 3.41: Error del esquema de Runge-Kutta cuarto orden $\Delta t = 0.01$s

En cuanto a la respuesta con un paso temporal de Δt=0.01s, se puede ver como el error de la respuesta es prácticamente nulo a excepción de algunos picos. Se puede decir que para este valor de paso temporal la respuesta mediante este esquema es prácticamente identica a la exacta. Por ello, en términos de precisión, el esquema de Runge-Kutta es el más potente de todos los vistos en términos de exactitud.

Capítulo 4

Aplicación a un caso práctico. Ala del avión

Las alas de las aeronaves están diseñadas para trabajar a flexión y torsión principalmente, por lo que los materiales con los que estén construidas deben proporcionarles resistencia suficiente frente a esfuerzos de flexión para que no se partan. Además, deben de poseer una rigidez torsional que haga que no sufran deformaciones elevadas debidas a la torsión, lo cual puede generar cambios en el ángulo de ataque en algunas zonas del ala. Esto podría conllevar la entrada en pérdida del perfil.

La importancia de que un ala trabaje correctamente a flexión queda demostrada cuando el ala puede tener varios metros de diferencia en el plano vertical cuando está cargada de combustible respecto a cuando está sin él. Además en los test de certificación siempre hay pruebas de flexión en las que el ala se flexiona al máximo quedando prácticamente en posición vertical.

La estructura del ala ha ido cambiando a lo largo del tiempo y actualmente se pueden encontrar varios tipos de estructura en función de si el avión es comercial o de combate. En el caso de los aviones comerciales, se pueden encontrar estructuras conocidas como *'Thick box beam structure'* las cuales están formadas por varios largueros a lo largo del perfil (entre dos y tres) . Para el caso de las aeronaves de combate, cuyos perfiles tienen baja relación de aspecto, se pueden encontrar las estructuras de tipo *'Multi-spar box'* y las

estructuras para aviones con ala en delta, *'Delta wing box'*.

El objetivo de la estructura del ala es, aparte de generar la sustentación para que el avión pueda mantenerse en el aire y ascender, ser capaz de transmitir al fuselaje y soportar las cargas que se generan debido a la interacción fluido-estructura en vuelo, o las debidas a su propio peso, el motor, etc... El cálculo de los esfuerzos que debe soportar el ala debe ir acompañado siempre de un factor de seguridad establecido por la normativa FAR/CS-25 de **1.5**.

El objetivo en este apartado será simular la respuesta de un ala ante distintas cargas que se van a generar sobre ella y para ello se necesita un modelo de ala del cual se pueda generar una estructura sin necesidad de recurrir al modelado 3D. Se va a realizar una simplificación que consiste en considerar el ala como una viga empotrada de sección rectangular.

Se puede decir que es una buena aproximación, ya que el rectángulo que se está considerando podría asemejarse al cajón de torsión de un ala formado por los largueros. En este caso se desestimaría la parte de 'slats' y 'flaps' (figura 4.1). Por lo tanto, haciendo uso de unos datos que se asemejen a un perfil de aeronave comercial, el comportamiento de esta viga sería similar al que tendría el ala de la aeronave. Para ello se va a desarrollar la teoría de vigas de Euler-Bernoulli con la que modelar el comportamiento estructural del ala.

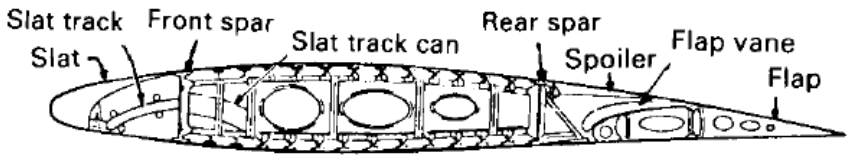

Figura 4.1: Sección transversal de un ala genérica de un avión comercial

4.1. Viga de Euler-Bernoulli

La teoría de Euler-Bernoulli de vigas es muy utilizada en estructuras aeronáuticas de pared finas como el ala o el fuselaje. Tiene un papel importante, ya que mediante esta teoría se diseñan estructuras matemáticamente para que después puedan ser analizadas por la mecánica de sólidos.

La base de la teoría es que la viga en su plano se comporta como un elemento infinitamente rígido y no hay deformaciones en el plano de la sección transversal. Esto no es lo que ocurre en la realidad pero se va a ver que es una buena aproximación. Además, se asume que durante la deformación en el plano, la sección transversal se mantiene plana y normal al eje de deformación de la viga [14].

La teoría tiene buenos resultados cuando se aplica a estructuras con secciones transversales sólidas y cuando los materiales con los que está formada la estructura tienen un comportamiento isotrópico. Se va a suponer para explicar la teoría, un estado tridimensional con los ejes *(x,y,z)* y con las direcciones **1**,**2** y **3** respectivamente.

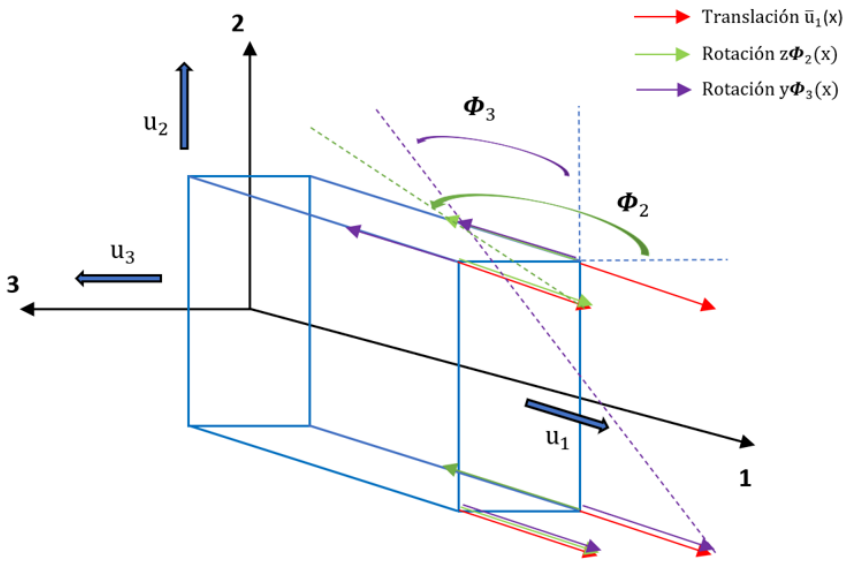

Figura 4.2: Campo de desplazamiento axial

Las condiciones impuestas por la teoría de Euler-Bernouilli modelan la viga de forma que, para la condición de indeformabilidad en el plano se obtiene que el vector de campo de desplazamientos (**U**) en la sección transversal consta de dos translaciones de cuerpo rígido:

$$u_2(x, y, x) = \bar{u}_2(x) \qquad u_3(x, y, x) = \bar{u}_3(x) \tag{4.1}$$

De la condición de la sección transversal plana tras la deformación, se obtiene un campo de desplazamientos axial que consiste en una translación en el eje **1** y dos rotaciones Φ_2 y Φ_3, las cuales se ilustran en la figura 4.2.

$$u_1(x, y, z) = \bar{u}_1(x) + z\Phi_2(x) - y\Phi_3(x) \tag{4.2}$$

La tercera condición del modelo Euler-Bernoulli implica la igualdad de la inclinación de la deformada del eje y de la rotación de la sección. Teniendo en cuenta que el criterio de signos de translaciones y momentos es tal que estos son positivos en la dirección de los ejes y que:

$$\Phi_3 = \frac{d\bar{u}_2}{dx} \qquad \Phi_2 = -\frac{d\bar{u}_3}{dx} \tag{4.3}$$

Sustituyendo (4.3) en la ecuación (4.2), el campo de desplazamiento a partir de la viga de Euler-Bernouilli queda definido por las ecuaciones (4.1) y (4.2). Los valores de $\bar{u}_1$, $\bar{u}_2$ y $\bar{u}_3$ representan las translaciones de cuerpo rígido que satisfacen la teoría de Euler-Bernoulli y las derivadas representan las rotaciones de la sección transversal.

Las deformaciones pueden expresarse en función del campo de desplazamientos como:

$$\epsilon_1 = \frac{\partial u_1}{\partial x} = \frac{d\bar{u}_1(x)}{dx} - z\frac{d^2\bar{u}_3(x)}{dx^2} - y\frac{d^2\bar{u}_2(x)}{dx^2}$$

$$\epsilon_2 = \frac{\partial u_2}{\partial y} = 0 \qquad \epsilon_3 = \frac{\partial u_3}{\partial z} = 0$$

$$\gamma_{12} = \frac{\partial u_1}{\partial y} + \frac{\partial u_2}{\partial x} = 0 \quad \gamma_{13} = \frac{\partial u_1}{\partial z} + \frac{\partial u_3}{\partial x} = 0 \quad \gamma_{23} = \frac{\partial u_2}{\partial z} + \frac{\partial u_3}{\partial y} = 0$$

El campo de deformaciones en el plano es cero debido a las implicaciones de la teoría sobre la sección transversal (infinitanmente rígida en su plano). Que las deformaciones transversales sean nulas, es debido a la condición de que la sección transversal permanece normal respecto al eje de la deformada de la viga.

Para el campo de tensiones, se define un estado tridimensional de tensiones en donde la variable R_1 representa la fuerza axial de la viga a lo largo del eje **1**. Las fuerzas de cortadura se representan como $V_2(x)$ actuando sobre el eje **2** y $V_3(x)$ actuando sobre el eje **3**. Quedan definidas por las siguientes ecuaciones:

$$R_1(x) = \int_A \sigma_1(x, y, z) dA \tag{4.4}$$

$$V_2(x) = \int_A \tau_{12}(x, y, z) dA \tag{4.5}$$

$$V_3(x) = \int_A \tau_{13}(x, y, z) dA \tag{4.6}$$

También aparecen los momentos flectores $M_2(x)$ actuando sobre el eje **2** y $M_3(x)$ actuando sobre el eje **3**.

$$M_2(x) = \int_A z\sigma_1(x, y, z) dA \tag{4.7}$$

$$M_3(x) = -\int_A y\tau_{12}(x, y, z) dA \tag{4.8}$$

El valor del esfuerzo $\sigma(x, y, z)$ depende del tipo de carga que se esté aplicando y se obtiene mediante la Ley de Hooke.

4.1.1. Elemento hermítico de 2 nodos

Una vez se han definido las condiciones de la teoría de Euler-Bernoulli para vigas generalizado, sin entrar a los casos de carga en concreto, se va a definir el elemento viga hermítico de 2 nodos.

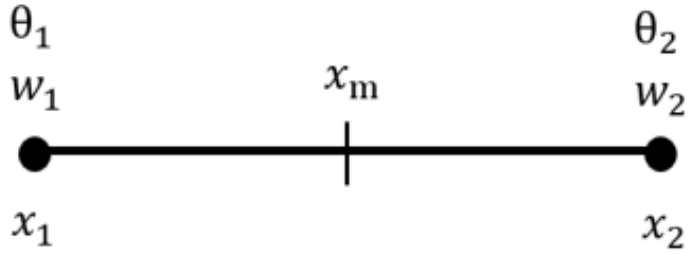

Figura 4.3: Elemento viga de dos nodos

Se representa la felcha ($w(x)$) en un punto de la viga como una función de interpolación de tercer orden con cuatro constantes. Esta interpolación viene exigida por la condición de deformación transversal nula y la condición de continuidad de flecha y giro ($\theta(x)$) entre elemento y elemento.

$$w(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3 \tag{4.9}$$

$$\theta(x) = \frac{\partial w}{\partial x} \tag{4.10}$$

Para obtener los coeficientes de la ecuación (4.9) se debe resolver el siguiente sistema de ecuaciones:

$$w_1 = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_1^2 + \alpha_3 x_1^3 \tag{4.11}$$

$$\theta_1 = \alpha_1 + 2\alpha_2 x_1 + 3\alpha_3 x_1^2 \tag{4.12}$$

$$w_2 = \alpha_0 + \alpha_1 x_2 + \alpha_2 x_2^2 + \alpha_3 x_2^3 \tag{4.13}$$

$$\theta_2 = \alpha_1 + 2\alpha_2 x_2 + 3\alpha_3 x_2^2 \tag{4.14}$$

Se efectúa el siguiente cambio de variable:

$$x_m = \frac{x_1 + x_2}{2} \quad ; \quad \xi = \frac{2}{l^e}(x - x_m) \tag{4.15}$$

Mediante el cambio de variable se puede expresar la ecuación (4.9) como,

$$w(\xi) = N_1(\xi) w_1 + N_{1\theta} \frac{dw(\xi)}{d\xi} + N_2(\xi) w_2 + N_{2\theta} \frac{dw(\xi)}{d\xi} \tag{4.16}$$

y aplicando,

$$\frac{dw}{d\xi} \cdot \frac{d\xi}{dx} = \frac{dw}{dx} \frac{2}{l^e} = \theta \frac{2}{l^e} \tag{4.17}$$

$$w(\xi) = N_1(\xi) w_1 + N_{1\theta} \theta_1(\xi) \frac{l^e}{2} + N_2(\xi) w_2 + N_{2\theta} \theta_2(\xi) \frac{l^e}{2} \tag{4.18}$$

Donde $(N_1, N_{1\theta}, N_2, N_\theta)$ representan las funciones de forma del elemento viga asociados a los nodos 1 y 2 respectivamente. Las funciones de forma del elemento viga, son funciones que pertenecen a la familia de los *polinomios de Hermite*. Una propiedad que deben de cumplir estas funciones de forma es:

$$N_i(\xi_j) = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases} \tag{4.19}$$

Las funciones de forma para el elemento viga hermítico de 2 nodos toman los siguientes valores y están representadas en la figura 4.4:

$$N_1(\xi) = \frac{\xi^3}{4} - \frac{3\xi}{4} + \frac{1}{2} \tag{4.20}$$

$$N_{1\theta}(\xi) = \frac{\xi^3}{4} - \frac{\xi^2}{4} - \frac{\xi}{4} + \frac{1}{4} \tag{4.21}$$

$$N_2(\xi) = -\frac{\xi^3}{4} + \frac{3\xi}{4} + \frac{1}{2} \tag{4.22}$$

$$N_{2\theta}(\xi) = \frac{\xi^3}{4} + \frac{\xi^2}{4} - \frac{\xi}{4} - \frac{1}{4} \tag{4.23}$$

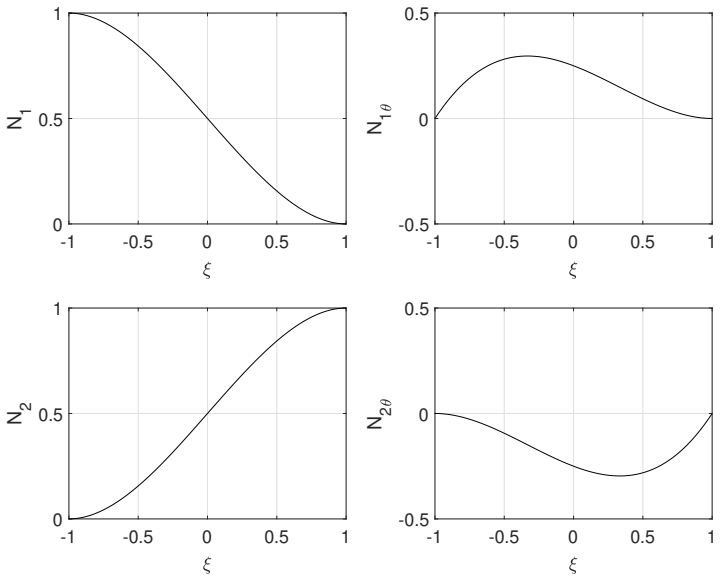

Figura 4.4: Funciones de forma del elemento viga hermítico de 2 nodos

La ecuación (4.18) se puede escribir como:

$$w(\xi) = \mathbf{N}\mathbf{a}^e \tag{4.24}$$

donde $\mathbf{N}$ es un vector con las funciones de forma,

$$\mathbf{N}(\xi) = [N_1(\xi)\ N_{1\theta}\frac{l^e}{2}\ N_2(\xi)\ N_{2\theta}\frac{l^e}{2}] \tag{4.25}$$

y $\mathbf{a}^e$ es el vector de desplazamientos modales,

$$\mathbf{a}^e = [w_1 \ \theta_1 \ w_2 \ \theta_1]^T \tag{4.26}$$

Se obtiene mediante la siguiente ecuación la curvatura en un punto de coordenada ξ:

$$\chi(\xi) = \frac{d^2w(\xi)}{dx^2} = \frac{d}{d\xi}\Big(\frac{dw(\xi)}{dx}\Big)\frac{d\xi}{dx} = \underbrace{\frac{d^2\mathbf{N}(\xi)}{d\xi^2}\frac{4}{(l^e)^2}}_{\mathbf{B}_f}\mathbf{a}^e \tag{4.27}$$

con,

$$\frac{d^2\mathbf{N}(\xi)}{d\xi^2} = \Big[\frac{d^2N_1(\xi)}{d\xi^2} \ \frac{d^2N_{1\theta}(\xi)}{d\xi^2}\frac{l^e}{2} \ \frac{d^2N_2(\xi)}{d\xi^2} \ \frac{d^2N_{2\theta}(\xi)}{d\xi^2}\frac{l^e}{2}\Big] \tag{4.28}$$

La curvatura queda finalmente expresada como:

$$\chi(\xi) = \mathbf{B}_f\mathbf{a}^e \tag{4.29}$$

La matriz $\mathbf{B}_f$ se conoce como la matriz de deformación a flexión o de curvatura del elemento. Esta matriz toma el siguiente valor.

$$\mathbf{B}_f = \begin{bmatrix} \frac{6\xi}{2} & \frac{3\xi-1}{l^e} & -\frac{6\xi}{2} & \frac{3\xi+1}{l^e} \\ l^e & 0 & l^e & 0 \end{bmatrix} \tag{4.30}$$

La expresión general del principio de trabajos virtuales (PTV) en notación matricial se puede expresar:

$$\int\int\int_V \delta\epsilon\sigma dV = \int_0^l \delta u^T b dV + \sum_{i=1}^{p}\delta u_i X_i \tag{4.31}$$

Donde b representa el vector de fuerzas internas, el último término representa las cargas en los nodos, δu y $\delta\epsilon$ son el movimiento y deformación virtual de un punto de la línea media de la viga y δu_i es el movimiento virtual del punto de actuación dela carga puntual X_i. Aplicado al elemento viga, se puede escribir como:

$$\int_{l^e} \delta\chi EI\chi dx = \left(\int_{-1}^{+1} \underbrace{[\delta\mathbf{a}^e]^T \mathbf{B}_f^T}_{\delta\chi} (EI) \mathbf{B}_f \underbrace{\frac{l^e}{2} d\xi}_{dx} \right) \mathbf{a}^e = \tag{4.32}$$

$$= \int_{-1}^{+1} \underbrace{[\delta\mathbf{a}^e]^T \mathbf{N}^T}_{\delta w} \frac{ql^e}{2} d\xi + \sum_{i=1}^{2} \delta w_i Z_i + \sum_{j=1}^{2} \delta\theta_i M_j \tag{4.33}$$

Donde q representa la carga repartida. Operando, se llega a la famosa expresión de la ecuación de equilibrio estático:

$$\mathbf{K}^e \cdot \mathbf{a}^e - \mathbf{q}^e = \mathbf{f}^e \tag{4.34}$$

La matriz $\mathbf{K}^e$ es la matriz de rigidez, ya mecionada en anteriores capítulos del trabajo, y esta matriz tiene el valor para cada elemento de:

$$\mathbf{K}^e = \int_{-1}^{+1} \mathbf{B}_f^T \mathbf{B}_f \frac{EIl^e}{2} d\xi = \left(\frac{EI}{l^3} \right)^e \begin{bmatrix} 12 & 6l^e & -12 & 6l^e \\ \ddots & 4(l^e)^2 & -6l^e & 2(l^e)^2 \\ & \ddots & 12 & -6l^e \\ sym & & \ddots & 4(l^e)^2 \end{bmatrix} \tag{4.35}$$

El vector de fuerzas nodales $\mathbf{q}^e$ debido a una carga q debidamente distribuida se expresa como:

$$\mathbf{q}^e = \int_{-1}^{+1} \mathbf{N}^T \frac{ql^e}{2} d\xi = ql^e \left[\frac{1}{2} \ \frac{l^e}{12} \ \frac{1}{2} \ -\frac{l^e}{12} \right]^T \tag{4.36}$$

Y por último el vector $\mathbf{f}^e$ de fuerzas nodales en el equilibrio donde se incluyen las cargas/momentos puntuales en los nodos.

En la introducción se presentó la ecuación de equilibrio estático (1.1). Esta ecuación es válida para obtener la solución estática de la estructura, pero cuando se quiere realizar un análisis dinámico de vibraciones, las fuerzas de inercia deben de ser tenidas en cuenta. Esto introduce un término nuevo que

es la matriz de masas consistente $\mathbf{M}$, la cual lleva asociada un término de segundo orden que es el campo de aceleraciones, como expresa la ecuación (1.2).

Para calcular la matriz de masas consistente de un elemento se usa la siguiente expresión dependiente de las funciones de forma, del área de la sección A y de la densidad ρ del material utilizado en la viga.

$$\mathbf{M}^e = \int_{l^e} \mathbf{N}\rho A\mathbf{N}^T dx = \frac{1}{2}\rho A l^e \int_{-1}^{+1} \mathbf{N}\mathbf{N}^T d\xi \tag{4.37}$$

$$\mathbf{M}^e = \frac{\rho A}{420}\begin{bmatrix} 156 & 22l^e & 54 & -13l^e \\ \ddots & 4(l^e)^2 & 13l^e & -3(l^e)^2 \\ & \ddots & 156 & -22l^e \\ sym & & \ddots & 4(l^e)^2 \end{bmatrix} \tag{4.38}$$

La ecuación (1.2) queda definida ensamblando ($\mathfrak{E}$) las matrices de rigidez de cada elemento y las matrices de masa. Así se podría resolver para relizar el análisis dinámico de la estructura mediante los métodos explicados en el Capítulo 2:

$$\mathbf{M} = \mathfrak{E}M^e \tag{4.39}$$

$$\mathbf{K} = \mathfrak{E}K^e \tag{4.40}$$

4.2. Diseño del ala

Una vez expuesta la introducción del procedimiento analítico, se va a proponer el diseño del ala para este trabajo. El avión escogido ha sido el Airbus A320 ya que es uno de los aviones más famosos de toda la historia, tiene un diseño alar muy simple y sus datos son accesibles. Para ello se necesitarán algunos datos los cuales se pueden encontrar en su manual como son la envergadura o la cuerda del ala a lo largo de la semi-envergadura [15].

Algunos de los datos que se van a utilizar de la aeronave se muestran en la Tabla 4.1.

Envergadura [m]	34
Superficie alar (S_w) [m^2]	122,4
Superficie estabilizador (S_t) [m^2]	30,7
Cuerda en la raíz (c_r) [m]	6,07
Cuerda en la punta (c_t) [m]	1,64
MTOW [kg]	78017,89
MLW [kg]	66224,49
MZFW [kg]	67131,67
Masa ala [kg]	6000
$C_{L\alpha}$	5,21
Máxima altitud operativa [m]	12500

Tabla 4.1: Datos necesarios del avión Airbus 320

El siguiente paso es seleccionar el perfil alar con el que luego se diseñará la viga que se va a simular. Para ello se va a seleccionar un perfil NACA asociado al avión A320 que en este caso es el NACA 23015 que se muestra en la figura 4.5. Para diseñar el perfil, se ha recurrido a la página web de creación de perfiles *airfoilstools.com* la cual tiene una base amplia de datos de perfiles aerodinámicos y permite modificarlos al gusto del usuario.

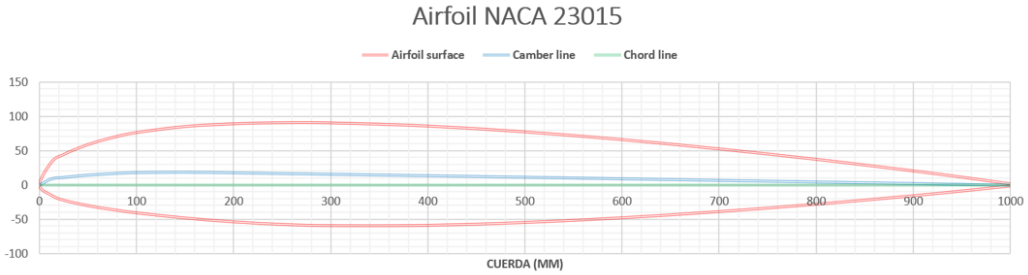

Figura 4.5: Perfil alar NACA 23015 estandar de 1 metro de cuerda.

Para el diseño de la caja de torsión mencionada, se va a hacer uso de las consideraciones expuestas en [16]. Además, la viga que va a simular el ala en este caso se va a dividir en **3 secciones**, cada una con un área distinta que reproduzca la tendencia decreciente de la cuerda en un ala. Cuanto mayor

fuera el número de secciones usadas, más se acercaría el diseño del trabajo al diseño de un ala real.

El ala se va a dividir en **10 elementos** con **11 nodos**, cada elemento con la misma longitud. Se van a crear las **3 secciones proporcionales a 3 tramos distintos del ala del Airbus A320** como se ve en la figura 4.6.

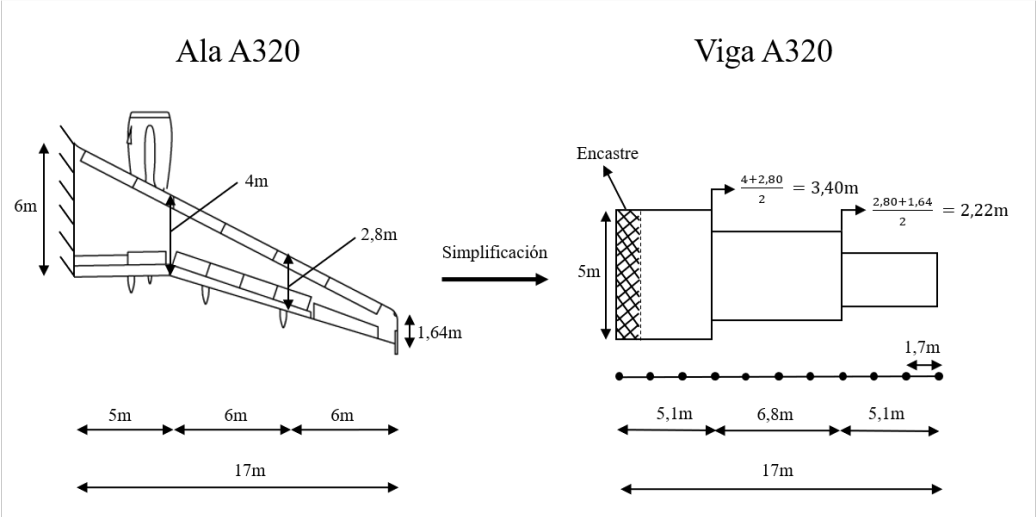

Figura 4.6: Secciones de la viga simplificada del ala del A320.

Las medidas que se obtienen de aproximar la vista en planta del ala se observan en la tabla 4.2. Las medidas de longitud se han aproximado teniendo en cuenta que cada elemento de la viga mide 1,7m de longitud.

Dimensión de la viga longitudinal		
Tramo	Longitud [m]	Cuerda [m]
1	5,1	5,0
2	6,8	3,4
3	5,1	2,22

Tabla 4.2: Dimensiones de la vista en planta de la viga.

Una de las consideraciones geométricas que no se van a tener en cuenta es la torsión geométrica del ala. Esto es que el ala se diseña con una determinada torsión, la cual produce distintos ángulos de ataque en cada sección del ala.

En este trabajo el ala tendrá el mismo ángulo de ataque para todas las secciones. Al tratarse de un análisis dinámico y no de un análisis aerodinámico, este aspecto no es muy relevante, pero se remarca para mayor entendimiento del proceso de diseño.

Tampoco se va a tener en cuenta el ángulo diedro del ala que para este avión es positivo. Este ángulo afecta a la estabilidad de la aeronave de forma que, cuando se realiza un alabeo, el ala del lado hacia donde se ha alabeado produce una mayor sustentación por lo que el momento generado respecto al centro de masas de la aeronave aumenta también y el avión tiene a estabilizarse. El fenómeno de aumento de sustentación no es competencia de este trabajo, pero en definitiva es debido a un cambio de ángulo de ataque en cada ala. Por lo explicado se puede decir que no considerar este efecto tampoco va a variar de forma significativa los resultados obtenidos, aunque en un análisis aerodinámico sí que sería un factor importante a considerar.

En los aviones comerciales puede observarse en la parte del ala que está cercana al encastre, que la cuerda usada es mayor. Esto es debido a que esa zona suele estar reservada para guardar el tren de aterrizaje y, por consiguiente, es un espacio hueco donde no se tendrían largueros ni costillas. Para el caso de estudio y la redimensión del ala en viga, ese espacio se va a considerar como útil para posicionar elementos estructurales del ala como largueros, costillas y larguerillos. Por lo tanto, esta modificación sí que podría generar diferencias respecto al caso real.

La última consideración que no se va a tomar en cuenta es la flecha (positiva) que tiene el ala en todos los aviones comerciales. En este caso se tomará el ala (viga) como un elemento que se extiende de forma perpendicular al fuselaje.

La posición de los elementos de la estructura interna del ala se determina siguiendo los siguientes pasos, que son los mismo que se usarán en el diseño que se va a hacer del ala: [16]

- Localizar el larguero del borde de ataque a un porcentaje constante de la cuerda, desde la raíz hasta la punta del ala. El larguero de borde de ataque debe situarse entre el 12 % o 17 % de la cuerda.
- Localizar el larguero de borde de salida de forma similar. En este caso deberá estar situado entre el 55 % y el 60 % de la cuerda. Normalmente, se suele seleccionar el 60 % para que el alerón se sitúe en un 30 % de la cuerda.

En el ala se pueden encontrar otros elementos como los alerones, los flaps, los slats, costillas, etc. Los cuales tienen su procedimiento de colocación propio [16], pero no se va a entrar en ello en detalle puesto que no influye en el estudio.

Atendiendo a las especificaciones explicadas anteriormente, se puede comenzar a diseñar el ala del estudio a partir de las distintas secciones. Se van a seleccionar los valores de 12 % de la cuerda para el larguero del borde de ataque y 60 % para el larguerillo de borde de salida, puesto que el resto de las partes del ala se van a despreciar. Como explican los criterios, este posicionamiento de los largueros respecto al porcentaje de la cuerda será constante con toda el ala.

Tras generar los perfiles y aplicar los requisitos de situación de la caja de torsión el resultado se puede ver en las figuras 4.7, 4.8 y 4.9.

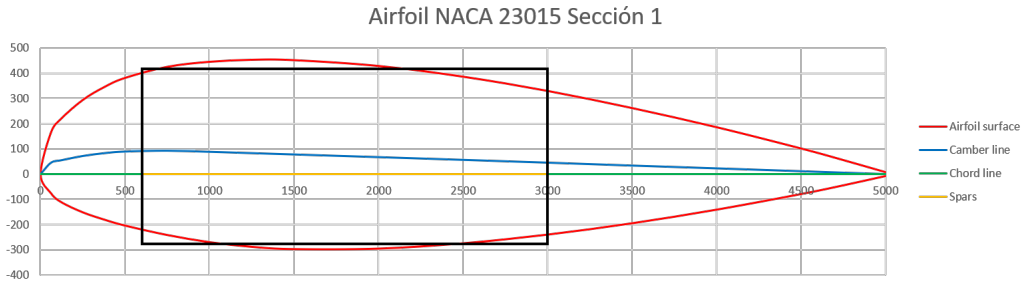

Figura 4.7: Primera sección transversal de la viga (ala) creada.

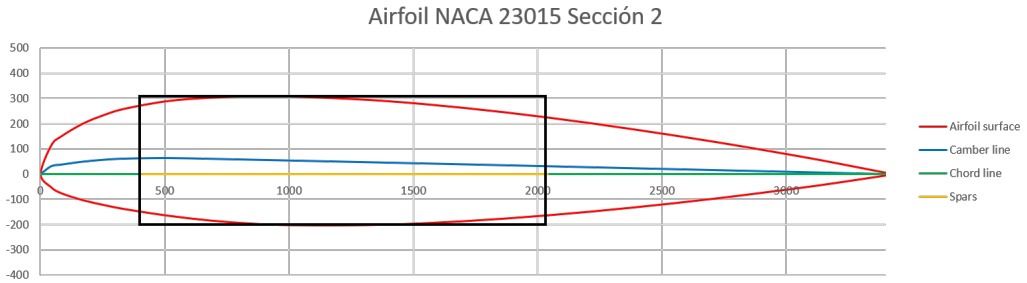

Figura 4.8: Segunda sección transversal de la viga (ala) creada.

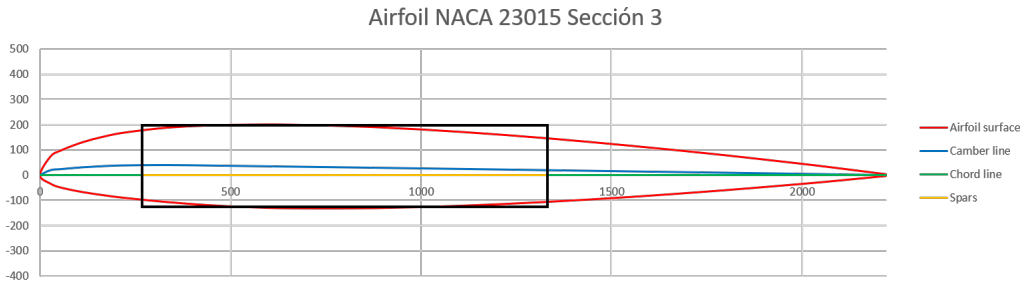

Figura 4.9: Tercera sección transversal de la viga (ala) creada.

De este modo, mediante las figuras 4.7, 4.8 y 4.9 queda dimensionada la viga completamente a falta de concretar el espesor que tendría su sección (caja de torsión). Para poder dimensionar este dato, hay que establecer una analogía entre el espesor del revestimiento del ala de la aeronave y el espesor de la sección de la caja de torsión. Por ello, una buena aproximación sería considerar que el espesor de la caja de torsión diseñada sería similar al revestimiento del ala de una aeronave.

Este tipo de valores suele tener el orden de milímetros, pero los fabricantes de aviones no suelen publicar este tipo de datos asociados al espesor de los revestimientos de las distintas secciones del avión. Los revestimientos del ala de los aviones suelen estar fabricados con aluminio y en algunas ocasiones con material compuesto. Además, en la mayoría de las ocasiones suelen llevar recubrimientos como es el caso del recubrimiento en fibra de vidrio para aviones con revestimiento de aluminio. Este tipo de estructuras se usa para que aguante los momentos generados por la flexión (*bending moments*). Para

el Airbus 320, el valor del revestimiento del ala se sitúa en las 1/8 pulgadas. La viga en este momento se encuentra completamente dimensionada. Las dimensiones de las secciones transversales del ala se muestran en la Tabla 4.3, además de algunos parámetros necesarios para el análisis dinámico como es el momento de inercia (I_z).

Datos de las secciones transversales de la viga					
Sección	Ancho [m]	Alto [m]	Espesor [m]	Área [m^2]	I_z [m^4]
1	2,4	0,695	0,003175	0,019612927	0,001996380
2	1,632	0,5	0,003175	0,013497878	0,000703167
3	1,0656	0,325	0,003175	0,008789988	0,000192332

Tabla 4.3: Datos de las secciones transversales de la viga.

4.3. Amortiguamiento proporcional (Rayleigh)

Todo sistema con respuestas no lineales o los sistemas lineales que no tienen un amortiguamiento proporcional, requieren de la evaluación de matrices proporcionales de amortiguamiento con el fin de obtener el amortiguamiento requerido por la estructura, ya que esta matriz no se puede crear a raíz de los ratios de amortiguamiento. Además, al tener una estructura constituida entera por el mismo material es razonable que se contruya una matriz única de amortiguamiento. En caso de tener varios materiales en la estructura, se podría recurrir a la construcción de una matriz no proporcional en donde se ensamblaran 2 matrices de amortiguamiento distintas cada una asociada a cada material [17].

La forma más sencilla de generar la matriz de amortiguamiento proporcional es haciendo que sea proporcional a las matrices de masa (**M**) y de rigidez (**K**) de la estructura. Por ello la matriz de amortiguamiento podría definirse como:

$$\mathbf{C} = a_0\mathbf{M} \tag{4.41}$$

$$\mathbf{C} = a_1\mathbf{K} \tag{4.42}$$

El comportamiento de estas matrices puede conocerse a través de la evaluación del valor de amortiguamiento modal para cada una.

$$2\omega_n M_n \xi_n = a_0 M_n \;\; ó \;\; a_1 K_n \;\; (Donde\ K_n = \omega_n^2) \tag{4.43}$$

De la ecuación (4.43) se puede obtener,

$$\xi_n = \frac{a_0}{2\omega_n} \;\; ó \;\; \xi_n = \frac{a_1\omega_n}{2} \tag{4.44}$$

Se puede combrobar la dependencia de ambas ecuaciones con la frecuencia natural. Esto es una limitación para el método puesto que la respuesta dinámica depende de la contribución a la misma de todos los modos de la estructura, aunque estas ecuaciones hayan sido desacopladas usando el modo superposición. Por tanto, si se tiene un rango de frecuencias muy elevado, las amplitudes de los modos se podrían ver distorsionados.

Rayleigh, para solucionar este problema, tuvo la idea de asumir que la matriz de amortiguamiento ($\mathbf{C}$) es proporcional a la combinación de la masa con la rigidez obteniendo.

$$\mathbf{C} = a_0\mathbf{M} + a_1\mathbf{K} \tag{4.45}$$

A la matriz de la ecuación (4.45) se la conoce como matriz de amortiguamiento de Rayleigh cuyo nuevo valor del ratio de amortiguamiento es:

$$\xi_n = \frac{a_0}{2\omega_n} + \frac{a_1\omega_n}{2} \tag{4.46}$$

Se pueden obtener las curvas que relacionan las frecuencias naturales con los ratios de amortiguamiento para los tres casos de amortiguamiento como se observa en la figura 4.10.

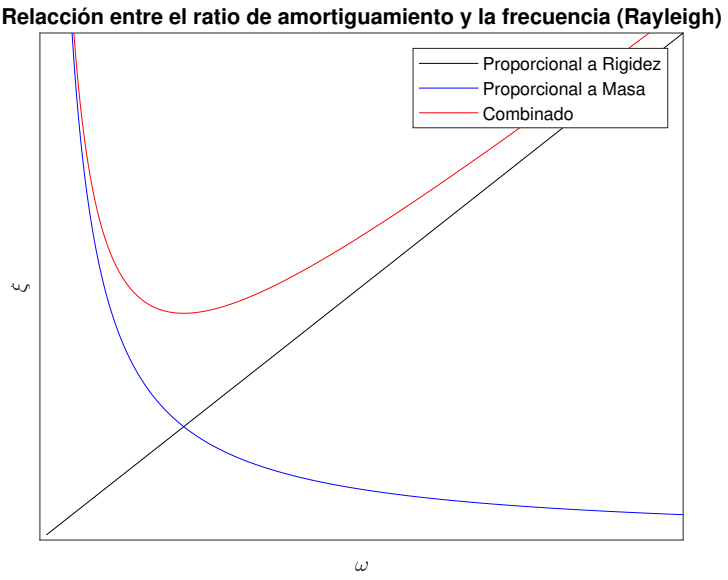

Figura 4.10: Relacción entre frecuencias y ratio de amortiguamiento.

La ecuación (4.46) tiene tres incógnitas en la misma ecuación, pero en el caso de que se conocieran dos frecuencias (ω_n y ω_m) asociadas a dos modos de vibración distintos con sus respectivos valores de amortiguamiento (ξ_n y ξ_m), se podrían obtener los valores de a_0 y a_1 sin más que resolver el sistema de ecuaciones.

En la mayoría de los casos, se asume que el amortiguamiento de ambas frecuencias es igual, de modo que $\xi_n = \xi_m$. Por tanto el sistema a resolver queda como:

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = 2\frac{\omega_n\omega_m}{\omega_n^2 - \omega_m^2} \begin{bmatrix} \omega_n & -\omega_m \\ -\frac{1}{\omega_n} & \frac{1}{\omega_n} \end{bmatrix} \begin{bmatrix} \xi_n \\ \xi_m \end{bmatrix} \tag{4.47}$$

Aplicando $\xi_n = \xi_m = \xi$:

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \frac{2\xi}{\omega_n + \omega_m} \begin{bmatrix} \omega_n\omega_m \\ 1 \end{bmatrix} \tag{4.48}$$

Determinada la ecuación (4.48), sólo faltaría determinar cuales son las frecuencias que deberían de utilizarse. Una de las frecuencias debería ser aquella asociada a la frecuencia fundamental del problema y la otra debería seleccionarse de modo que sea la mayor de las frecuencias asociadas a los modos que contribuyen de forma significativa a la respuesta dinámica.

Si se observa la figura 4.10, se ve claramente como todas las frecuencias comprendidas en el rango de las frecuencias seleccionadas tendrán asociados coeficientes de amortiguamientos menores. Además, todas las frecuencias fuera de este rango tendrían ratios de amortiguamiento significativamente mayores por lo que los modos asociados a altas frecuencias del sistema se eliminarán de forma efectiva debido a los elevados ratios de amortiguamiento.

El objetivo será seleccionar un ratio de amortiguamiento real que pueda alcanzarse por la estructura e intentar que mediante una selección adecuada de las frecuencias ω_n y ω_m los modos asociados a las altas frecuencias queden debidamente amortiguados.

4.4. Análisis de la respuesta del Ala

Habiendo dimensionado el ala, se calculan las matrices de masa y de rigidez de la estructura mediante las ecuaciones (4.38) y (4.35). Mediante el cálculo de estas matrices se pueden obtener las frecuencias asociadas a los modos de la estructura que se muestran en la siguiente tabla:

Se puede observar como los modos tienen un rango de frencuencias variado. Se tienen desde modos asociados a bajas frecuencias como modos asociados a altas frecuencias. Esta diferencia de frecuencias va a condicionar fuertemente tanto la estabilidad del método como el paso temporal que se deba utilizar.

Modo	Frecuenia (Hz)	Modo	Frecuencia (Hz)
1	3,71	16	749
2	15,6	17	797,7
3	36,1	18	904,7
4	74,3	19	978
5	90,7	20	1123,8
6	121,1	21	1128,5
7	175,2	22	1306,2
8	232,5	23	1367,2
9	247,4	24	1518,8
10	324,5	25	1598,4
11	363,7	26	1629,3
12	421,9	27	1840,5
13	543	28	1965,4
14	557,8	29	2226,1
15	638,3	30	2554,8

Tabla 4.4: Tabla de modos de la estructura del ala del avión

Como se vió en la sección de estabilidad, el paso temporal debe ajustarse de forma que su valor esté relacionado con la frecuencia más alta del problema. En este caso, la frecuencia más alta tiene como valor 2554,8Hz que en periodo es aproximadamente 0,0004s. Este paso temporal es muy pequeño y el utilizarlo al analizar el problema causaría un tiempo de simulación demasiado alto. Por lo tanto, se seleccionará un paso temporal suficiente como para que se representen una cantidad de modos adecuados en la respuesta y no afecte de forma importante al resultado, es decir, que no se desvíe del resultado de la solución exacta.

En el primer caso que se va a analizar, se va a utilizar una fuerza que está afectada por una Gaussiana y las frecuencias que están asociadas a la fuerza se pueden observar en la tabla 4.5.

La fuerza utilizada en este primer caso con las frecuencias utilizadas se representa en la figura 4.11. Esta fuerza se aplicará en el extremo libre del ala, en sentido vertical. Además, se usa un factor de amortiguamiento para

Frecuencia	Valor (Hz)
1	0,08
2	0,2
3	0,8
4	1,6

Tabla 4.5: Frecuencias de la fuerza utilizada en el primer caso.

la estructura de:

$$\xi = 0,05 \tag{4.49}$$

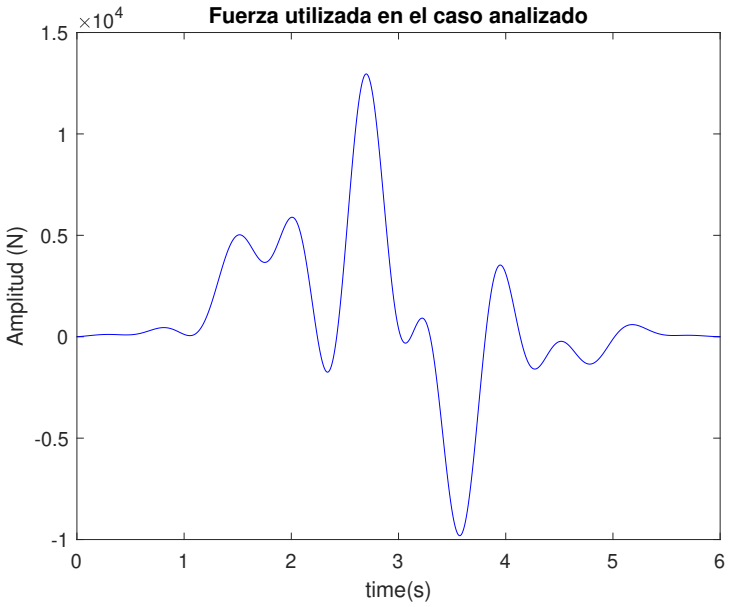

Figura 4.11: Fuerza aplicada en el primer caso.

Es una fuerza perturbativa que como puede observarse no tiene frecuencias altas asociadas. Mediante *Matlab*, al igual que en el anterior caso práctico, se simula la respuesta del sistema mediante los distintos métodos estudiados. En primer lugar, se representa la respuesta con un paso temporal $\Delta t = 0,1$s junto con la solución exacta. Los resultados se muestran en las figuras 4.12 y 4.13 y están asociados al campo de desplazamientos de la punta del ala, aunque los desplazamientos en otras secciones serían muy similares pero con

distintas amplitudes

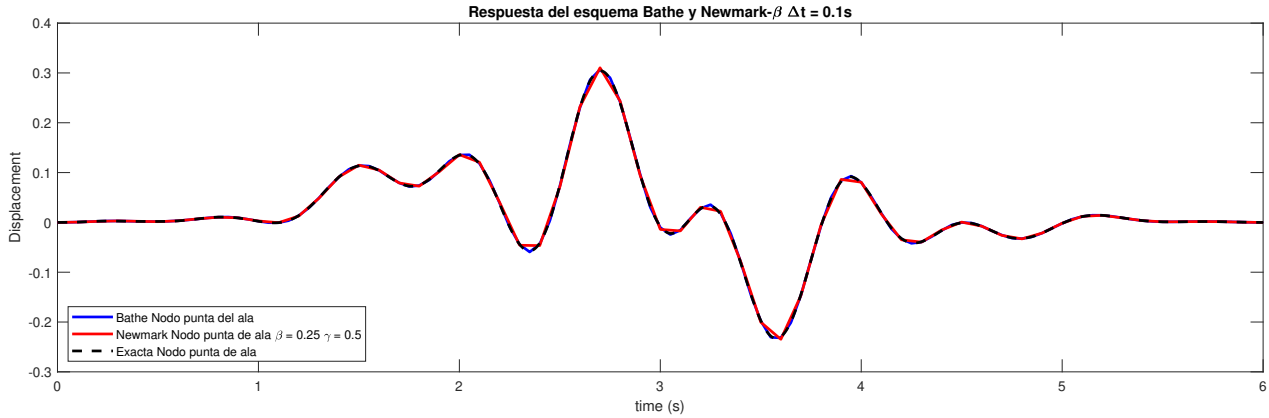

Figura 4.12: Campo de desplazamientos de la punta del ala mediante el método de Newmark-β y Bathe con $\Delta t = 0{,}1$s en el primer caso de fuerza aleatoria.

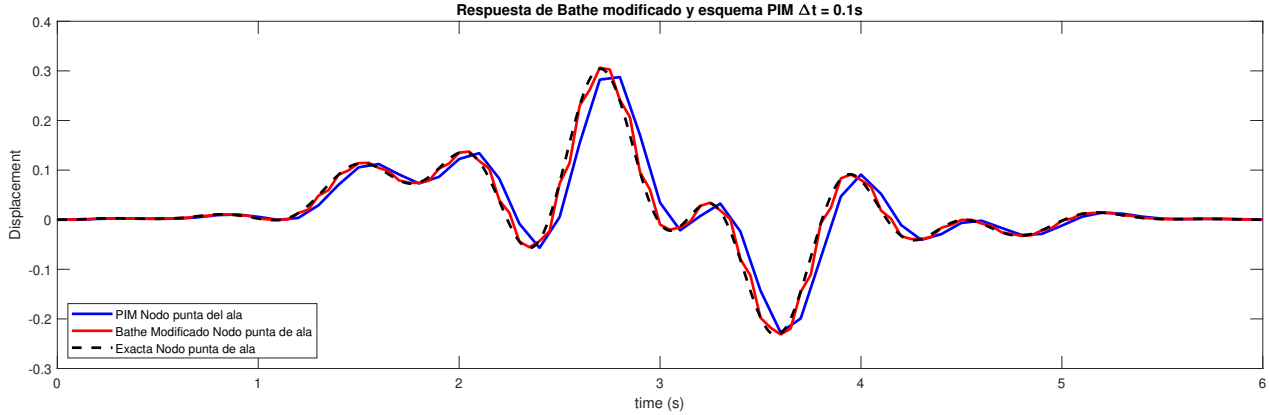

Figura 4.13: Campo de desplazamientos de la punta del ala mediante el PIM y Bathe modificado con $\Delta t = 0{,}1$s en el primer caso de fuerza aleatoria.

En vista de las figuras, se puede ver como utilizando el paso temporal de 0,1s la respuesta del sistema mediante el método de Bathe se aproxima bastante a la solución exacta debido a el subpaso temporal que analiza por cada paso temporal, de forma que contiene más puntos de solución del problema. En el método de Newmark-β, se aproxima la solución bien aunque no es capaz de reproducir la resolución exactamente debido al paso temporal utilizado. En el caso del PIM, el método no es capaz de representar la

solución con mucha precisión, a pesar de la precisión que el método da por definición, debido al paso temporal utilizado. El método de Bathe modificado, se utiliza con los parámetros que mejor solución ofrecieron en el caso práctico. Esto es, $\theta = 0,25$, $\beta_1 = 0{,}5$ y $\beta_2 = 0{,}5$ (en los siguientes casos se utilizarán estos valores para el método). Mediante este método, la solución se aproxima bastante a la solución real pero en algunos puntos se desvía no llegando a seguir la línea de solución exacta.

Se observa como la respuesta que se obtiene tiene la misma forma que la fuerza. La explicación de ello reside, no en el hecho de que la estructura sea muy rígida, si no en el hecho de que en la fuerza que se está aplicando no se consigue activar ningún modo de excitación de la estructura. El valor del modo con la frecuencia más baja de la estructura es de 3,71Hz y el valor de la frecuencia en la fuerza más alta es de 1,6Hz. Es por ello por lo que la fuerza no consigue excitar ninguno de los modos de la estructura y el ala se mueve de la misma forma que varía la fuerza.

No se ha representado la respuesta del método Runge-Kutta. Ha sido imposible puesto que la respuesta que ofrece este método con el paso temporal usado es divergente. Esto es debido, a que los autovalores asociados al problema no caen en la zona de estabilidad que se vió en la figura 2.10 con el paso temporal usado de 0,1s. Los autovalores (λ_i) del problema de Runge-Kutta son asociados a la matriz $\mathbf{A}$. Los autovalores de esta matriz tendrán una parte imaginaria y una parte real. La condición que ha de cumplirse es que:

$$\hat{\lambda}_{RK} = \lambda_i \Delta t \rightarrow \textit{Region de estabilidad} \tag{4.50}$$

En el último caso de estudio, ningún autovalor cae en la zona de estabilidad por lo tanto no puede representarse la respuesta.

Se va a analizar el mismo caso para un paso temporal de $\Delta t = 0{,}01$s. Se representa la solución en las figuras 4.14 y 4.15.

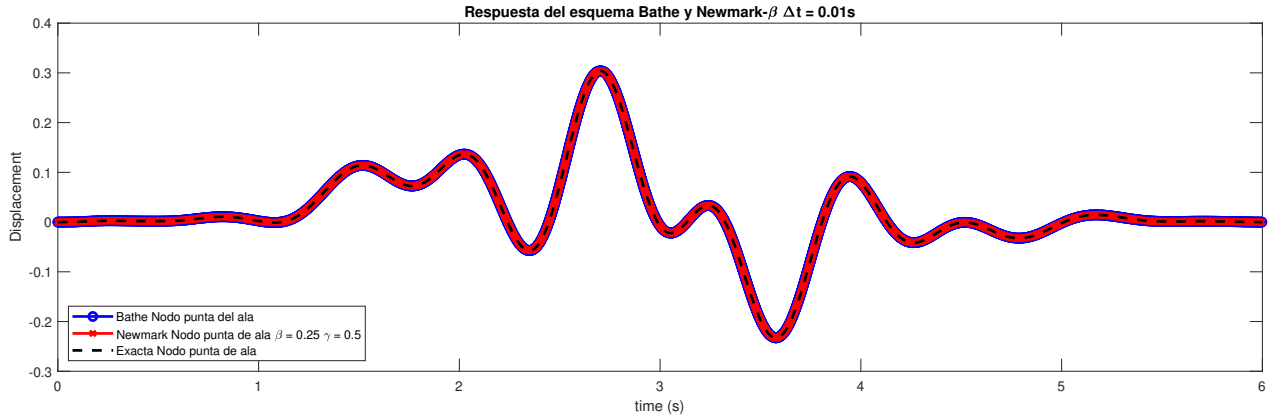

Figura 4.14: Campo de desplazamientos de la punta del ala mediante el método de Newmark-β y Bathe con Δt = 0,01s en el primer caso de fuerza aleatoria.

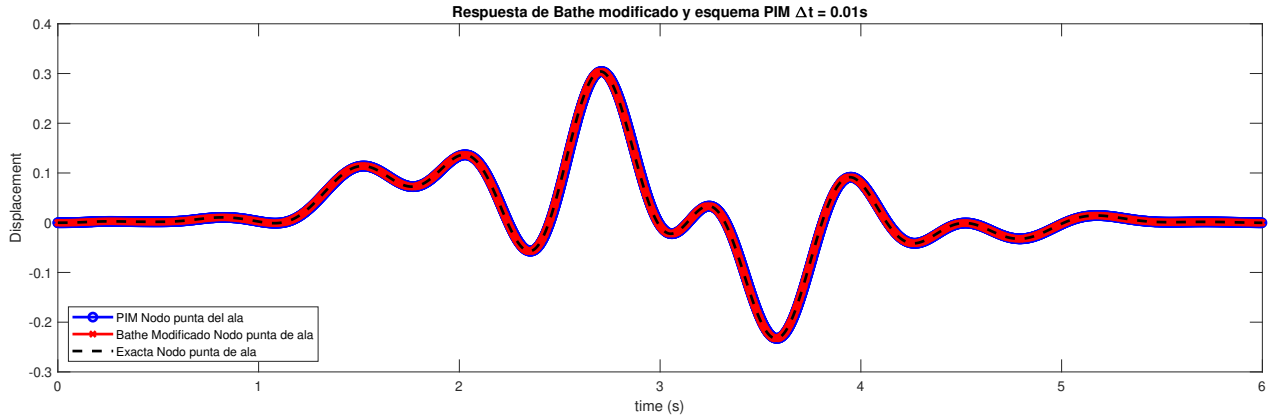

Figura 4.15: Campo de desplazamientos de la punta del ala mediante el PIM y Bathe modificado con Δt = 0,01s en el primer caso de fuerza aleatoria.

Para la solución de este caso, todos los métodos ofrecen una situación muy precisa en comparación con la solución real, con un error muy pequeño. De hecho, ampliando la imagen en el postprocesado, se ha podido ver como el método de Bathe es el esquema que mejor consigue aproximar la solución, al igual que en el primer caso, pero la diferencia es tan pequeña que a efectos prácticos no se puede considerar una diferencia sustancial.

En el caso de Runge-Kutta, la respuesta usando este paso temporal sigue siendo divergente y no se puede resolver. Se ha comprobado en este caso, como caen los autovalores en la gráfica de estabilidad 2.10 y el resultado se observa en la figura 4.16.

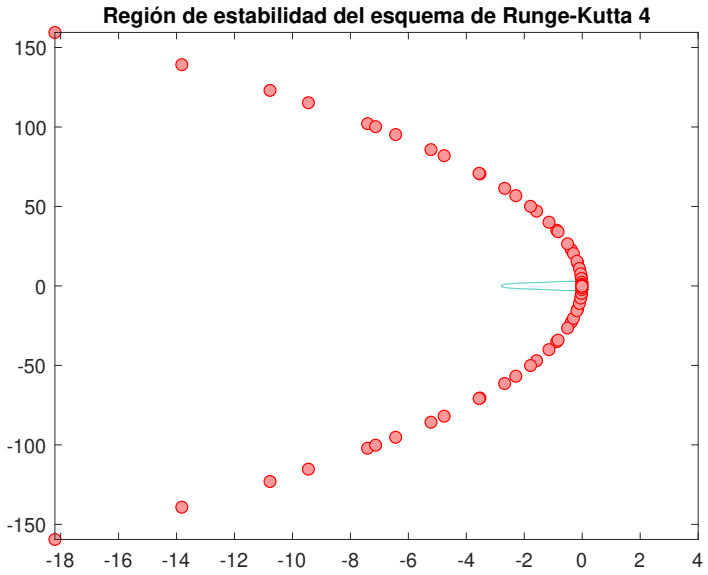

Figura 4.16: Autovalores del método de Runge-Kutta con Δt=0,01s.

Surge la pregunta de cuál sería el paso temporal que se tendría que utilizar para poder resolver el método en este problema. Mediante una iteración, se observa que con un paso temporal de Δt=0,000184s los autovalores caen en la región de estabilidad como se ve en la figura 4.17.

Este valor es el mismo para cualquier caso que se analice puesto que depende de las matrices del problema, que al estar resolviendo un problema lineal, son constantes, y de el paso temporal, cuyo valor depende de las condiciones del problema. Por lo tanto, la respuesta de este esquema no va a ser representada ya que al tener un paso temporal tan pequeño, no se disponen de medios para resolver el problema eficientemente.

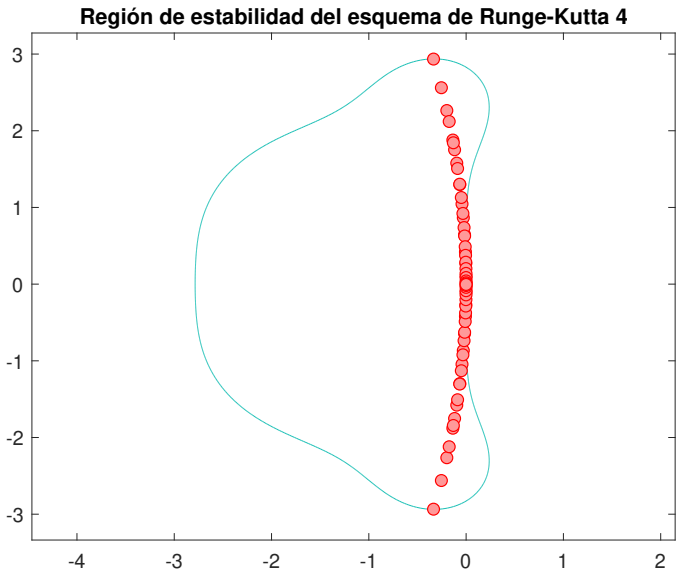

Figura 4.17: Autovalores del método de Runge-Kutta para con Δt=0,000184s.

Para el siguiente caso que se va a analizar, se va a cambiar la fuerza aplicada en el caso. La posición donde se va a aplicar se mantendrá y se van a cambiar las frecuencias que afectan a la fuerza. Las nuevas frecuencias asociadas a la fuerza se muestran en la tabla 4.6.

Frecuencia	**Valor (Hz)**
1	0,8
2	3,98
3	7,96
4	11,4

Tabla 4.6: Frecuencias de la fuerza utilizada en el segundo caso.

La forma que toma la nueva fuerza se muestra en la figura 4.18.

En este caso, las frecuencias asociadas a la fuerza sí que afecta a la estructura puesto que excita al menos un modo (representado en la figura 4.19), el que lleva asociado una frecuencia de 3,71Hz. Por lo tanto, se debería de ver

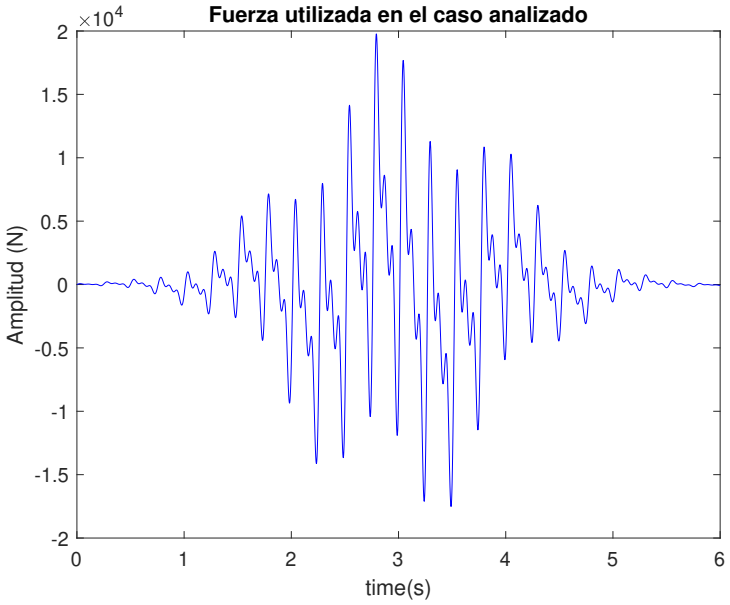

Figura 4.18: Fuerza aplicada en el segundo caso.

un cambio en la respuesta obtenida. Se van a representar a continuación las respuestas, mediante el uso de un paso temporal de Δt=0,1s , de los esquemas de integración. Los resultados se muestran en las figuras 4.20 y 4.21.

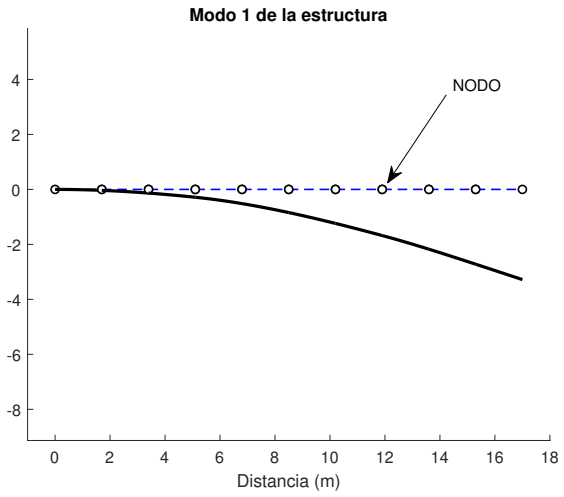

Figura 4.19: Modo 1 de la estructura alar.

Lo que se puede extraer de estas figuras, es que para ese paso temporal

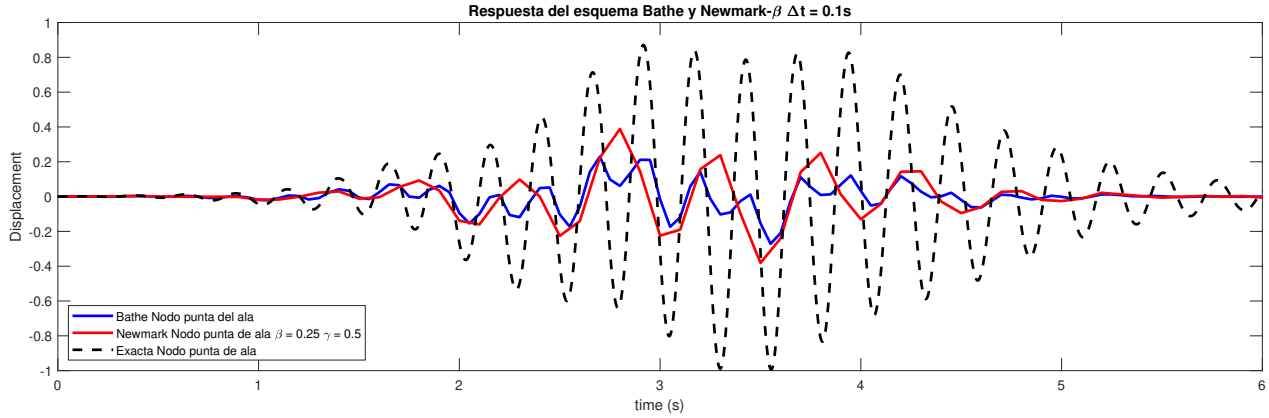

Figura 4.20: Campo de desplazamientos de la punta del ala mediante el método de Newmark-β y Bathe con $\Delta t = 0{,}1$s en el segundo caso de fuerza aleatoria.

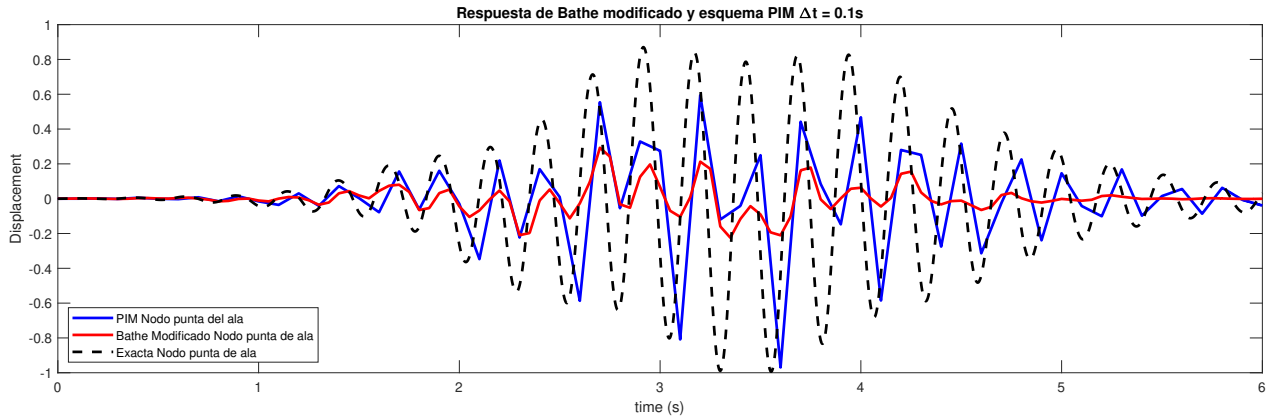

Figura 4.21: Campo de desplazamientos de la punta del ala mediante el PIM y Bathe modificado con $\Delta t = 0{,}1$s en el segundo caso de fuerza aleatoria.

usado, ninguno de los esquemas es capaz de seguir la evolución de la solución real. En comparación con el anterior caso, se ve como la solución tiene unas frecuencias asociadas mayores y esto es lo que impide a estos esquemas resolver correctamente el campo de desplazamientos. La frecuencia afectada es la asociada a 3,71Hz, tiene un periodo de 0,27s y según lo comentado relacionado con la precisión de la solución en el Capítulo 2, se indica que le paso temporal debería ser 10 veces menor que el periodo que quiere resolverse. Con ello se obtendría que el paso temporal para poder resolver este esquema co-

rrectamente y representar bien las frecuencias sería de 0,027s. La respuesta de Runge-Kutta no se muestra por la misma razón que ocurría en el primer caso.

Se va a utilizar otro paso temporal para representar la respuesta de Δt=0.1s, de modo que la respuesta debería verse representada correctamente. En las figuras 4.22 y 4.23 se observa el resultado.

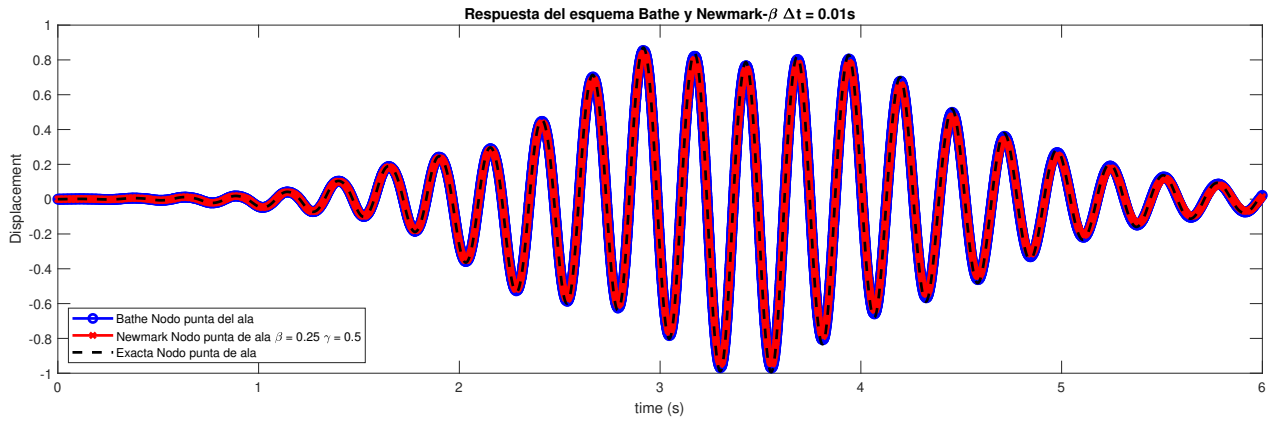

Figura 4.22: Campo de desplazamientos de la punta del ala mediante el método de Newmark-β y Bathe con Δt = 0,01s en el segundo caso de fuerza aleatoria.

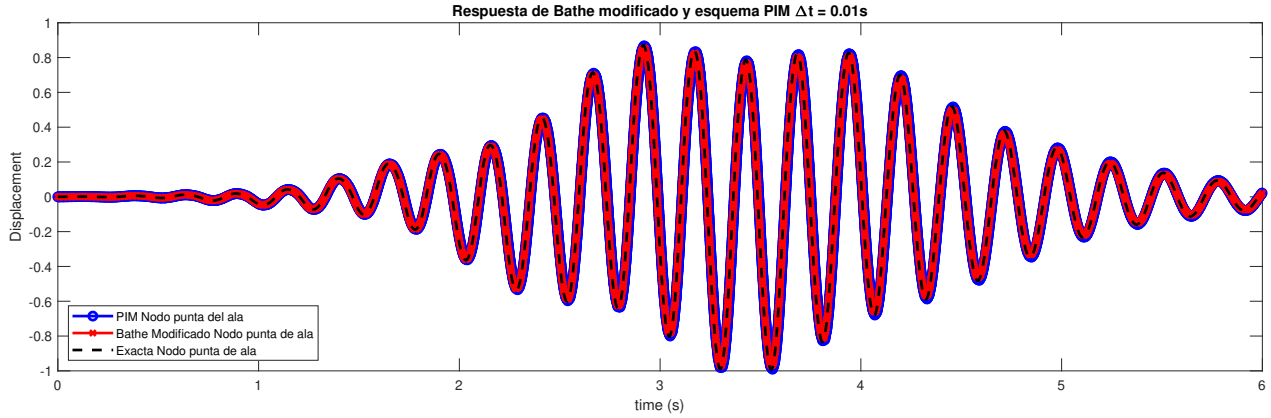

Figura 4.23: Campo de desplazamientos de la punta del ala mediante el PIM y Bathe modificado con Δt = 0,01s en el segundo caso de fuerza aleatoria.

Se ve como al utilizar este paso temporal, todos los esquemas representan

la respuesta con muy poco error respecta a la respuesta exacta. Esto confirma la necesidad de reducir el paso temporal para adecuarlo a las necesidades del problema. A diferencia del anterior problema y como se preeveía, la respuesta que se produce no tiene la misma forma que la fuerza utilizada, ya que se está mostrando la respuesta a al menos uno de los modos de la estructura que es afectado.

Por último, se va a analizar el caso de una carga constante situada en la punta del ala. Esta carga va a simular la posición de un motor de A320 en el extremo del ala. Sabiendo que el peso del motor aproximadamente es de 2300kg, la fuerza a aplicar constante tendrá un valor de -23,345kN (negativa ya que el peso ejerce una fuerza hacia abajo y la parte positiva en el grado de libertad vertical está orientada hacia arriba).

Se va a utilizar un paso temporal en primer lugar de Δt=0,1s y las respuestas de los distintos esquemas se muestran en las figuras 4.24 y 4.25.

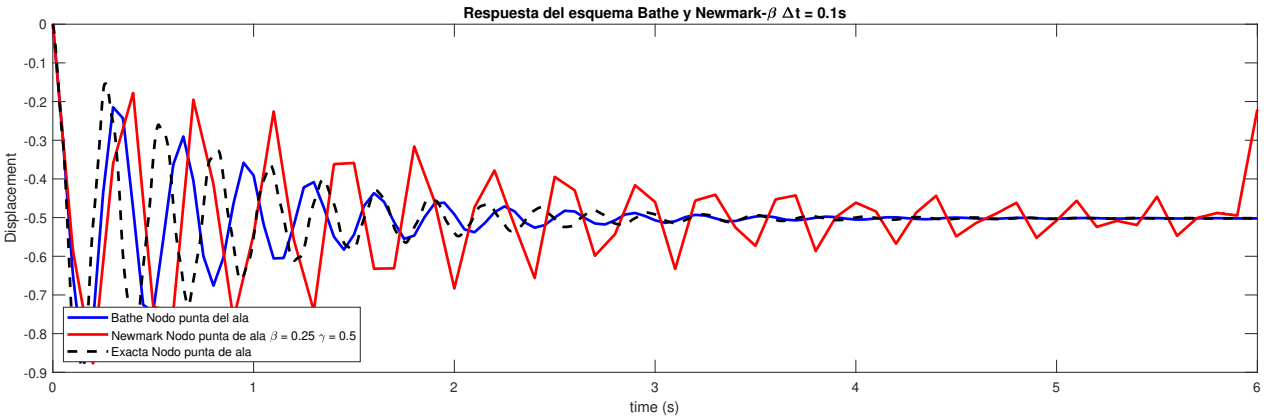

Figura 4.24: Campo de desplazamientos de la punta del ala mediante el método de Newmark-β y Bathe con Δt = 0,1s con fuerza constante.

Se observa en las gráficas, que la respuesta dinámica se ve amortiguada aproximadamente a los 4 segundos por lo que el factor de amortiguamiento utilizado es quizás algo alto, ya que en condiciones normales se sabe que la estructura alcanzaría el equilibrio en un periodo de tiempo más largo. La solución obtenida con el esquema de Newmark-β tiene una gran dispersión

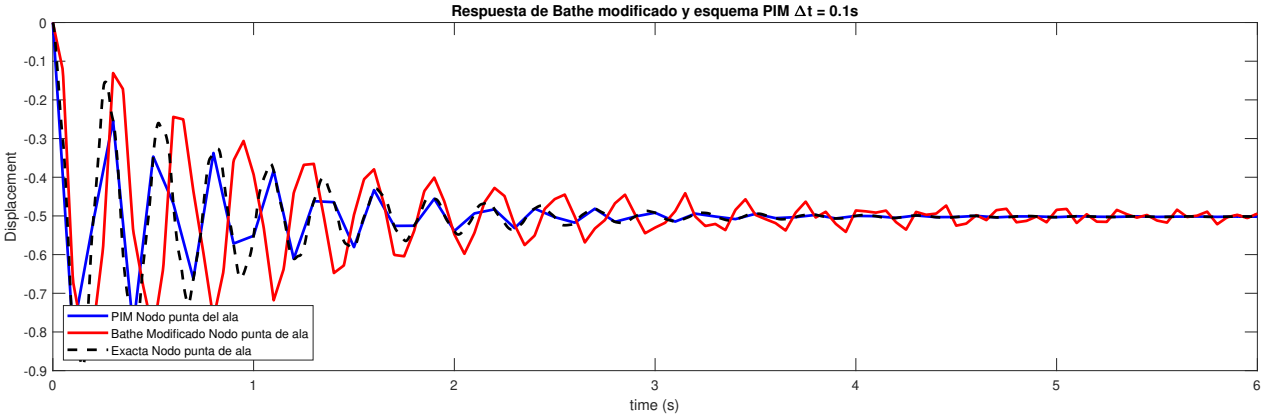

Figura 4.25: Campo de desplazamientos de la punta del ala mediante el PIM y Bathe modificado con $\Delta t = 0{,}1$s con fuerza constante.

numérica y no es capaz se seguir el valor de la solución exacta. En los esquemas de Bathe y Bathe modificado se elimina la dispersión numérica que se introducía con el método de Newmark-β pero tiene un desfase respecto de la solución real que impide representar el campo de desplazamientos correctamente, debido al paso temporal utilizado. El PIM en este caso consigue no desfasarse de la solución exacta pero se observa una pérdida de amplitud respecto a la solución real. Al no cambiar las matrices y no alcanzar el paso temporal mínimo, la solución por Runge-Kutta no puede ser representada.

Se va a representar la solución del campo de velocidades puesto que su solución se considera de interés de estudio. Se representa en las figuras 4.26 y 4.27.

En ellas se ve como el esquema de Newmark-β mantiene la dispersión numérica en el campo de velocidades al igual que en el campo de desplazamientos, que es lo que hace que esta no se represente correctamente. En el resto de esquemas se observa como tienen un comportamiento similar que el que tenían en la representación del campo de desplazamientos. Para ver si estos problemas se solucionan reduciendo el paso temporal, se va a representar la respuesta de la estructura alar con un paso temporal de Δt=0,01s.

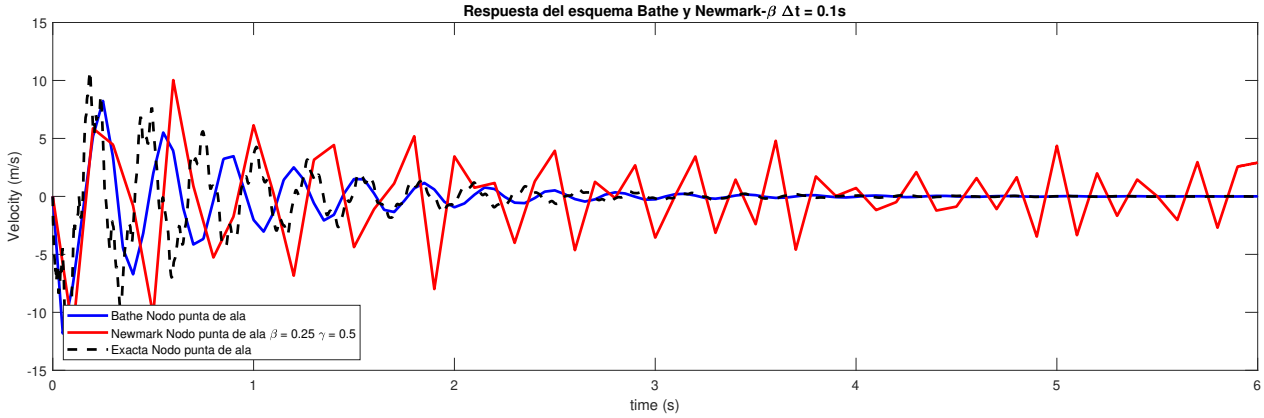

Figura 4.26: Campo de velocidades de la punta del ala mediante el método de Newmark-β y Bathe con $\Delta t = 0{,}1$s con fuerza constante.

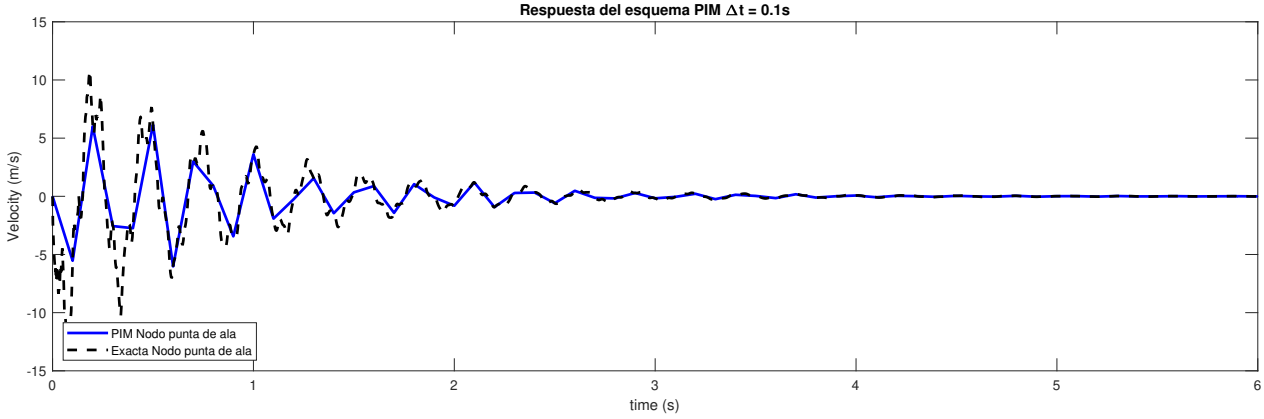

Figura 4.27: Campo de velocidades de la punta del ala mediante el PIM con $\Delta t = 0{,}1$s con fuerza constante.

En la figuras 4.28 y 4.29 se representa el campo de desplazamientos.

En la respuesta en desplazamientos con el nuevo paso temporal, se aprecia como todos los esquemas representan con poco error el campo de desplazamientos respecto a la solución exacta. Sólo se ven algunas diferencias en el método de Bathe modificado pero son muy pequeñas. Se observa en las figuras 4.30 y 4.31 el campo de velocidades.

En vista del campo de velocidades con Δt=0,01s, al reducir el paso temporal se ha conseguido aproximar bien el campo de velocidades en todos los

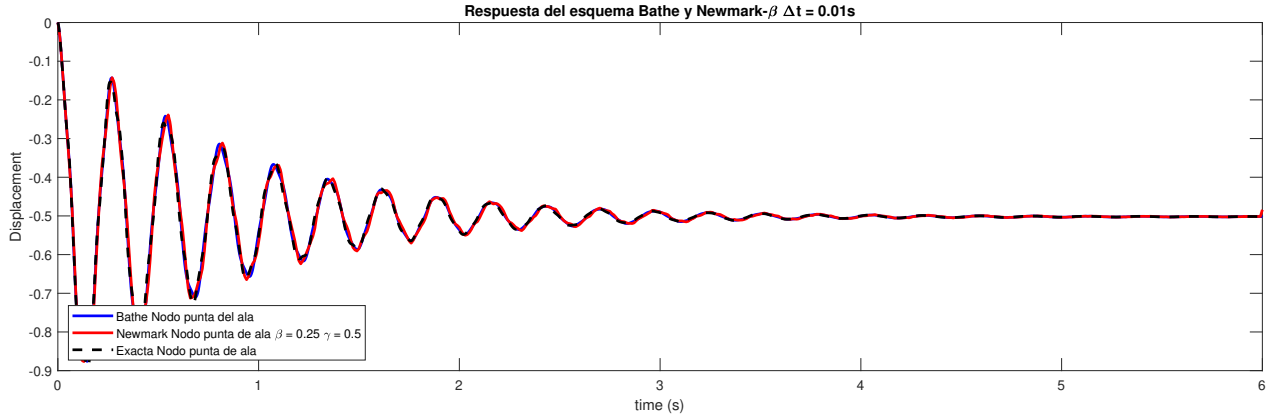

Figura 4.28: Campo de desplazamientos de la punta del ala mediante el método de Newmark-β y Bathe con $\Delta t = 0{,}01$s con fuerza constante.

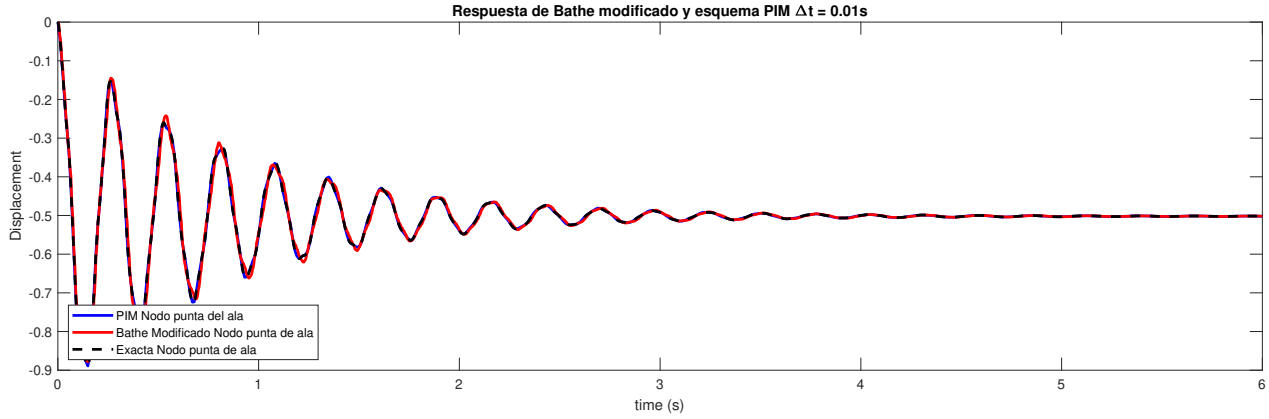

Figura 4.29: Campo de desplazamientos de la punta del ala mediante el PIM y Bathe modificado con $\Delta t = 0{,}01$s con fuerza constante.

métodos excepto en el caso del método de Newmark-β. Se observa que aunque el campo de desplazamientos quede bien aproximado, el campo de velocidades mantiene la disipación numérica en forma de oscilaciones espurias de la solución como se expuso en las desventajas del método de mismo. Esto es debido a que el método tiene dificultades en muchas ocasiones de representar los modos asociados a las altas frecuencias. En este caso, la fuerza constante excita modos de altas frecuencias que imposibilitan que el método de Newmark represente estos modos y genere las oscilaciones espurias en la solución. En la figura 4.32 se puede ver detallada la respuesta en un instante de tiempo.

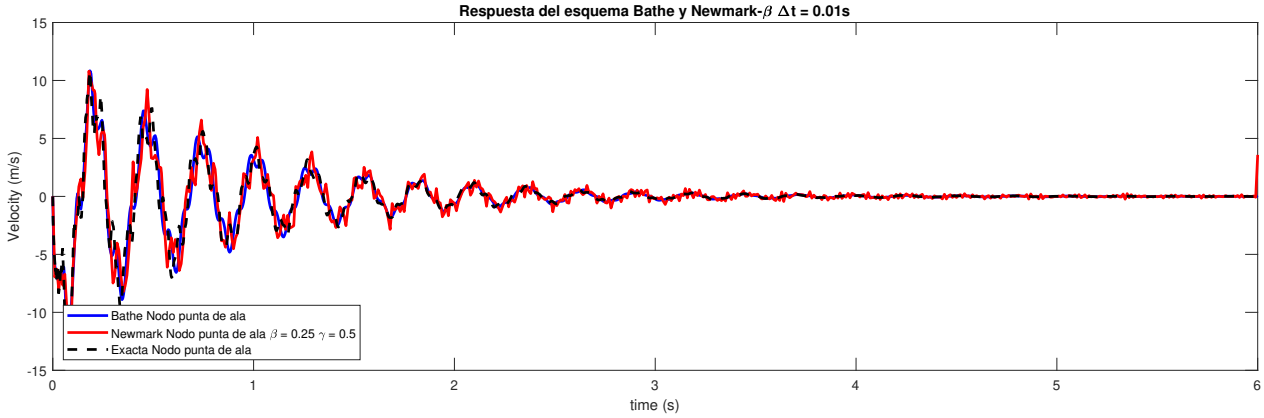

Figura 4.30: Campo de velocidades de la punta del ala mediante el método de Newmark-β y Bathe con $\Delta t = 0{,}01$s con fuerza constante.

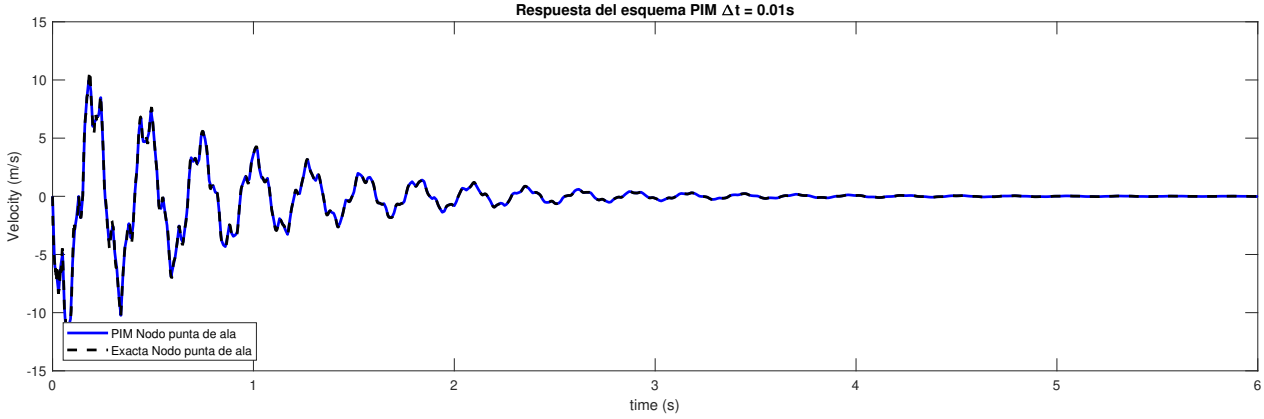

Figura 4.31: Campo de velocidades de la punta del ala mediante el PIM con $\Delta t = 0{,}01$s con fuerza constante.

Se observa como se producen las oscilaciones espurias alrededor de la solución exacta y además, el esquema de Bathe presenta imprecisiones debidas al paso temporal usado. Como se ha comentado anteriormente, el paso temporal utilizado debería ajustarse al periodo del modo excitado más alto (mayor frecuencia).

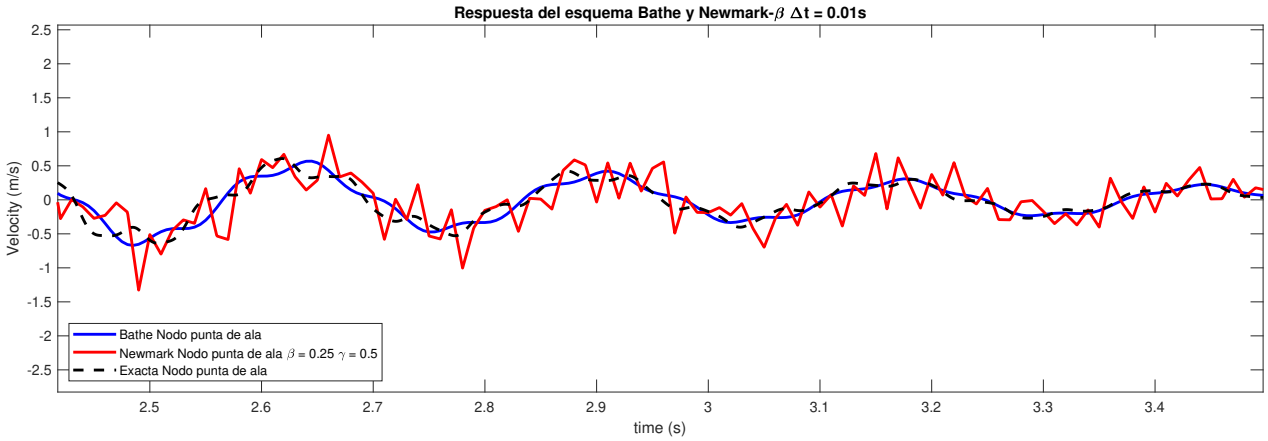

Figura 4.32: Campo de velocidades detallado de la punta del ala mediante el método de Newmark-β y Bathe con $\Delta t = 0{,}01$s con fuerza constante.

Capítulo 5

Creación de una herramienta visual para análisis de problemas dinámicos mediante *App Designer* de *Matlab*

En este capítulo se va a desarrollar la creación de una herramienta de análisis dinámico mediante una interfaz gráficada programa por medio de *App Designer*. *App Designer* es un software desarrollado por *Matlab* el cual ofrece la posibilidad de crear una interfaz gráfica con la posibilidad de que el propio usuario pueda programarla a su manera.

Para su funcionamiento, se presenta en el display un canvas en donde se pueden posicionar distintos elementos proporcionados por el software para montar la aplicación. *App Designer* es el que se encarga de crear el código base para estos elementos colocados en el *Lay-out* y el usuario es el que se encarga de añadir el código en función del uso que se les quiera dar.

Además, el software comprueba automáticamente la existencia de errores en el mismo y ofrece soluciones. En caso de que sea un error que el programa no detecta porque no afecta a la estructura del código, el compilador avisará cuando se lance el programa.

En cuanto a los componentes que se pueden añadir en el programa se pueden encontrar (se enumeraran aquellos utilizados en el programa a desarrollar):

- Componentes comunes: Son aquellos que responden a interacciones.
- Ejes: Permiten la visualización de resultados.
- Herramientas de contenedores y figuras: Son aquellas para agrupar componentes o generar pestañas.
- Componentes de instrumentación: Permiten visualizar los estados del programa y generar datos de entrada.

A continuación se van a enumerar los componentes que engloba cada sección explicada anteriormente.

5.1. Componentes comunes

Button

Este botón se puede programar para realizar una acción siempre que sea pulsado.

Checkbox

Genera una lista en la que permite al usuario marcar una o varias opciones mediante un *tick*.

Dropdown

Genera un desplegable con distintas opciones y sólo permite seleccionar una de ellas.

NumericEditField

Campo que permite introducir caracteres de tipo numérico.

EditField

Campo que permite introducir caracteres de tipo *string*.

Image

Permite introducir una imagen guardada en el ordenador y modificar sus propiedades.

ListBox

Permite generar una lista de elementos y marcar o desmarcar los que se deseen.

Button Group

Permite seleccionar mediante un círculo pequeño sólamente una de los opciones de una lista.

Slider

Genera una barra con un rango de valores numéricos y ofrece al usuario la posibilidad de poder desplazarse a lo largo de ella para seleccionar un valor concreto.

Spinner

Permite mediante unas flechas de 'Arriba' y 'Abajo' cambiar el valor numérico asociado al complemento.

StateButton

Funciona como un *Button* con la diferencia de que al ser pulsado, el botón permanece pulsado y realiza la acción programada y al volver a pulsarlo vuelve a su posición inicial con la posibilidad de realizar otra acción distinta.

Label

Crea un espacio para escribir lo que el usuario quiera.

5.2. Ejes

UIAxes

Genera unos ejes que sirven como salida a los distintos datos y pueden ser modificados mediante botones comunes o incluso modificar sus propiedades desde las propias opciones del botón.

5.3. Herramientas de contenedor y figura

Panel

Genera un cuadro con un titulo donde agrupar componentes.

TabGroup

Genera en el *Lay-out* distintas pestañas a las cuales se les puede modificar el nombre y orden.

5.4. Componentes de instrumentación

Lamp

Indicador que puede iluminarse en distintos colores en función de la programación de la App

Switch

Actúa como un interruptor. Tiene dos posiciones, 'On' y 'Off'.

5.5. Lay-out de la aplicación de análisis dinámico

Se va a explicar la interfaz diseñada para la resolución de problemas dinámicos. Se explicará elemento por elemento la funcionalidad que tiene cada elemento del programa y su funcionamiento estará detallado en uno de los apéndices.

La pantalla principal del programa se muestra en la figura 5.1.

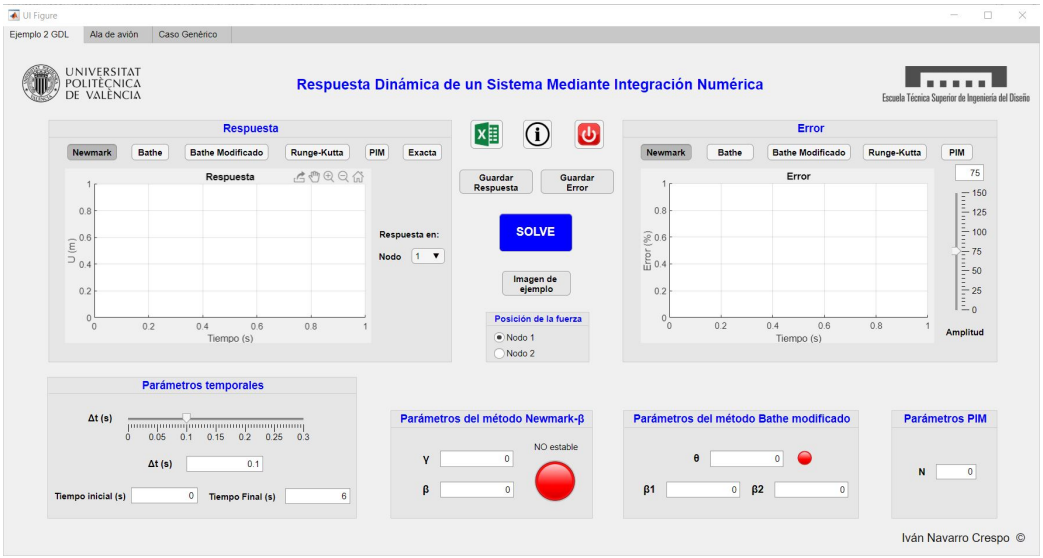

Figura 5.1: Layout del programa.

El programa tiene tres pestañas de trabajo en función del problema que se quiera resolver. La primera pestaña que tiene como nombre 'Ejemplo 2 GDL' y en ella se resuelve el problema propuesto en el capítulo 3 del trabajo. La segunda pestaña llamada 'Ala de avión' resuelve el problema propuesto del capítulo 4 del trabajo y por último la pestaña llamada 'Caso genérico' resuelve un caso cualquiera que el usuario quiera resolver. Para la explicación del programa la explicación se va a centrar en esta última pestaña puesto que contiene más funcionalidades y de su funcionamiento se puede extrapolar el funcionamiento de las 2 primeras pestañas.

En primer lugar se va a explicar el panel que contiene los parámetros temporales. En este panel se encuentra el *slider* del paso temporal Δt que permite selecionar el paso temporal de un rango de 0 a 0.3 segundos cuya arquitectura se muestra a continuación. En el cuadro numérico debajo del slider, se muestra el valor de paso temporal elegido.

```
function tsSliderValueChanged(app, event)

            value = app.tsSlider.Value;
            app.PasoTempField.Value = value;

            if value ==0
            opts = struct('WindowStyle','modal',...
                'Interpreter','tex');
            errordlg('El paso temporal tiene que tomar un valor
    distinto de 0',...
                'Parameters Error', opts);
            else

            end
        end
```

Se puede ver como hay un protector de código en forma de mensaje de error, de forma que si se escoge un paso temporal de valor 0 segundos, aparece un error como el que se ve en la figura 5.2.

Figura 5.2: Cuadro de error en el paso temporal.

Debajo, se puede seleccionar el tiempo inicial de la simulación que por defecto será 0 y el tiempo final de la simulación que por defecto será de 6 segundos. En la figura 5.3 se puede ver como queda conformado este panel.

Figura 5.3: Panel de parámetros temporales.

El siguiente panel a analizar será el panel de 'Parámetros de Newmark-β'. Se puede ver en la figura 5.4. En el panel podemos distinguir dos cuadros de entrada numérica para dar un valor a los dos parámetros del método Newmark-β definidos en el capítulo 2 del trabajo.

Figura 5.4: Panel de parámetros del método Newmark.

Además, se ha incluido un elemento tipo *lamp* que sirve para controlar la estabilidad del esquema seleccionado en función de los valores de β y γ seleccionados. El código asociado al color que adquiere la lámpara está está asociado a los cuadros numéricos y es el mismo en ambos.

```
function gammavalue_3ValueChanged(app, event)

    valuegamma = app.gammavalue_3.Value;
    valuebeta = app.betavalue_3.Value;
```

```
        if valuegamma >= 0.5 && valuebeta >= valuegamma/2

            app.LampNewmarkstab_3.Color = 'g';
            app.EstadoLabel_3.Text = 'Estable';

        else

            app.LampNewmarkstab_3.Color = 'r';
            app.EstadoLabel_3.Text = 'NO Estable';

        end
    end
```

Cuando se cumple la condición de estabilidad la lámpara se pone en color verde y en el texto que aparece arriba aparece la palabra 'Estable' y si la condición de estabilidad la lámpara tiene color rojo y el texto de arriba aparece como 'No Estable'.

Se analiza ahora el panel de 'Parámetros del método Bathe modificado' que engloba los parámetros tanto del método de Bathe como del método de Bathe modificado. Se muestra su estructura en la figura (5.5).

Figura 5.5: Panel de parámetros del método Bathe y Bathe modificado.

En este caso se tienen tres cuadros numéricos donde se insertan los valores de los parámetros asociados al método de Bathe definidos en el capítulo 2 del trabajo, y además hay otro elemento tipo *lamp* al igual que en el método de Newmark-β que en este caso tiene como objetivo proteger el parámetro

θ introducido. La estructura del código está envuelta en el cuadro numérico del parámetro θ y tiene el siguiente código asociado.

```
function thetavalue_3ValueChanged(app, event)

            valuetheta = app.thetavalue_3.Value;

            if valuetheta == 0

                app.LampBathemod_3.Color = 'r';

            else

                app.LampBathemod_3.Color = 'g';

            end
        end
```

En donde la lámpara permanece roja mientras el valor del parámetro θ sea 0 y se vuelve del color verde cuando el valor del parámetro θ sea distinto de cero.

El siguiente panel a analizar es el panel de 'Parámetros PIM' en donde se define simplemente el parámetro N del método en un cuadro numérico como se ve en la figura 5.6.

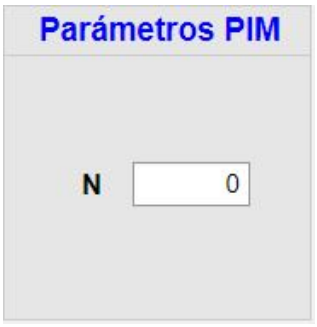

Figura 5.6: Panel de parámetros del PIM.

El siguiente elemento a analizar es el panel de fuerzas en donde se in-

troducen los valores de los distintos tipos de fuerza que pueden aplicarse y la posición donde esta fuerza está aplicada (nodo). El panel se ilustra en la figura 5.7.

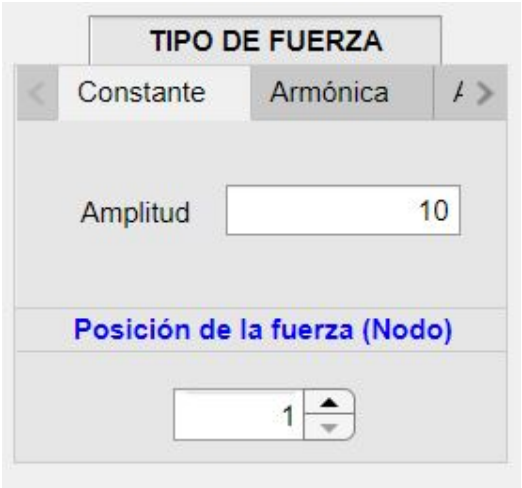

Figura 5.7: Panel de tipo y posición de la fuerza.

Las opciones de fuerza que pueden elegirse para el caso general son:

- Constante
- Armónica
- Aleatoria

Para la fuerza constante, la aplicación permite introducir un valor de la amplitud de la fuerza. Para el caso de la fuerza armónica, la aplicación permite introducir una amplitud de la excitación y una frecuencia. Por último, para el caso de la fuerza aleatoria simplemente permite el ingreso de la amplitud de la fuerza, que es del mismo tipo que la utilizada en el capítulo 3.

En el caso de la posición de la fuerza, mediante un spinner se puede introducir el valor donde se va a querer introducir la fuerza, de modo que si la estructura tiene varios grados de libertad, la posición de la fuerza indica en cuales de los grados de libertad del nodo se va a colocar en función del

ensamblado de las matrices estructurales.

El panel a explicar ahora va a ser en el que se representa la respuesta y viene ilustrado en la figura 5.8.

Figura 5.8: Panel de respuesta temporal.

En este panel abajo a la derecha, puede seleccionarse el tipo de fuerza que se va a aplicar en el problema. Con el panel explicado antes se definen los valores de esa fuerza. Más arriba se puede seleccionar mediante un *spinner* el nodo donde queremos mostrar los desplazamientos que al igual que en el panel anterior, si hay más de un grado de libertad, la respuesta que se muestra es en el grado de libertad específico de ese nodo.

Los botones que hay arriba, son botones que al pulsarse realizan distintas acciones. En este caso, Del botón que esté pulsado se mostrará la respuesta. En el caso que haya varios botones pulsados, el programa mostrará la respuesta del método pulsado en último lugar. Cuando se 'despulsa' un botón, la grafica se borra y para mostrar la respuesta de otro método habrá que pulsar otro botón. El código utilizado en estos botones es:

```
function NewmarkButton_3ValueChanged(app, event)
            value = app.NewmarkButton_3.Value;
            nodo  = app.SpinnerRespuestaNodo.Value;
```

```
            if value==1

                app.UIAxes_3.Title.String = 'Desplazamiento
Nodal Newmark';
                    plot(app.UIAxes_3,app.Param.t,...
                        app.Resultado.Newmark.d(nodo,:));

            else
                app.UIAxes_3.cla

            end
        end
```

También se muestra el error de la respuesta comparado con el resultado exacto, este panel está ilustrado en la figura 5.9.

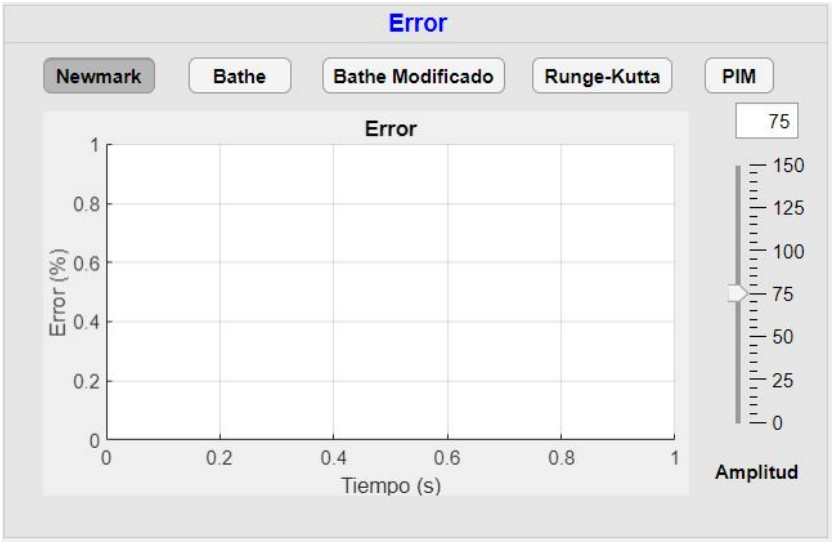

Figura 5.9: Panel de error de la respuesta.

En este panel al igual que en el otro, se utilizan los mismos botones para representar el error del método que se desee. El código asociado a ello es el siguiente:

```
function NewmarkButtonerror_3ValueChanged(app, event)
```

```
            value = app.NewmarkButtonerror_3.Value;

            if value==1

                app.UIAxes2_3.Title.String = 'Error Newmark';
                plot(app.UIAxes2_3,app.Resultado.Exacto.t_exacta
,...
                    app.error.errorN,'-*');

            else
                app.UIAxes2_3.cla
            end
        end
```

En este panel, se ha añadido un slider que sirve para controlar el valor del eje vertical de la gráfica del error. Se ha incluido este slider de control puesto que para grandes pasos temporales el error puede ser grande, pero para pasos temporales pequeños el error puede ser de valor muy pequeño (1-2 %) y el tener esta herramienta mejora la visibilidad de los resultados. Además el valor utilizado en la escala aparece en un cuadro de texto como se ve. El código utilizado para esta herramienta es el siguiente:

```
function AmplitudSlider_4ValueChanging(app, event)
            changingValue = event.Value;
            app.AmplitudField_3.Value=round(changingValue);
            if changingValue==0
            opts = struct('WindowStyle','modal',...
                'Interpreter','tex');
            errordlg('No puede ser 0 el valor del eje Y,
    ajustelo de nuevo',...
                'Limits Error', opts);
            else
                ylim(app.UIAxes2_3,[-app.AmplitudField_3.Value
    app.AmplitudField_3.Value])
            end
        end
```

Además este slider está protegido como se ve en el código, de forma que si en el valor de amplitud se elige el valor de 0, salta un error como el que se observa en la figura 5.10.

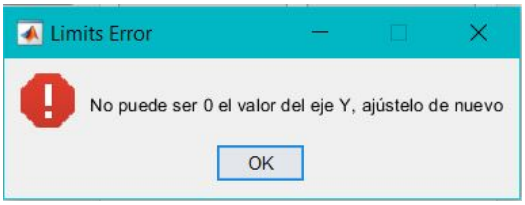

Figura 5.10: Panel de error de la respuesta.

La siguiente parte de la aplicación a explicar, son los botones de 'guardar respuesta' y 'guardar error'. Estos son los que se ilustran en la figura 5.11.

Figura 5.11: Botones de guardado de respuesta y error.

Mediante estos botones, se guarda en la misma carpeta que esté seleccionado en Matlab un archivo de tipo imagen de la respuesta o del error. Si se quieren guardar varias imágenes, habra que cambiar el nombre de la imagen guardada y al darle a guardar se creará otra con el mismo nombre pero distinto contenido. El código asociado a estos botones es el siguiente:

```
function GuardarRespuestaButton_3Pushed(app, event)
            SaveFigures(app,app.UIAxes_3,'RespuestaGenerica','
    Respuesta')
        end

        function GuardarErrorButton_3Pushed(app, event)
            SaveFigures(app,app.UIAxes2_3,'ErrorGenerico','
    Respuesta')
        end
```

La función *SaveFigures* tiene asociado el siguiente algoritmo:

```
function SaveFigures(app,UIAxes,SaveName,caso)

            %Funcion para guardar imagenes
            %Inputs
            %------
            %
            %app = Propiedades (Parametros y Matrices)
            %UIAxes = Grafico del que se quiere guardar la
    imagen
            %SaveName = Nombre del archivo
            %caso = Si lo que se guarda es la imagen de la
    respuesta o
            %error

            h = figure;
            h.Visible = 'off';
            x = UIAxes.XAxis.Parent.Children.XData;
            y = UIAxes.XAxis.Parent.Children.YData;

            switch caso

                case 'Respuesta'

                    plot(x,y,'b')

                case 'Error'

                    plot(x,y,'-b*')
                    h.CurrentAxes.YLim = [-app.AmplitudSlider_2.
    Value...
                        app.AmplitudSlider_2.Value];
            end


            h.CurrentAxes.YLabel.String = UIAxes.YLabel.String;
            h.CurrentAxes.YLabel.FontSize = UIAxes.YLabel.
    FontSize;
            h.CurrentAxes.XLabel.String = UIAxes.XLabel.String;
            h.CurrentAxes.XLabel.FontSize = UIAxes.XLabel.
```

```
FontSize;
            h.CurrentAxes.Title.String = UIAxes.Title.String;
            h.CurrentAxes.Title.FontSize = UIAxes.Title.FontSize
;
            h.CurrentAxes.XLim = [0 max(x)];
            saveas(h,SaveName,'jpg')
            savefig(h,SaveName)
            delete(h)

        end

    end
```

Bajo estos botones, se encuentran los botones de 'SOLVE' y 'CARGAR MATRICES', ilustrados en la figura 5.12.

Figura 5.12: Botones solución y carga de matrices.

El primero de ellos es el que se encarga de mostrar la respuesta y el error en función de los parámetros que se han introducido en el problema. El segundo, es un botón que se encarga de leer de un archivo *.xls* las matrices asociadas al problema y las guarda en una variable que luego es usada directamente por el botón de 'SOLVE'. El código asociado al botón que resuelve el problema se incluirá en los anexos y el código asociado a la carga de matrices se expone a continuación.

```
function CargarMATRICESButtonPushed(app, event)
            app.Matrices.K = readmatrix('Matrices.xlsx','Sheet',
    'Rigidez');
            app.Matrices.M = readmatrix('Matrices.xlsx','Sheet',
    'Masas');
            app.Matrices.C = readmatrix('Matrices.xlsx','Sheet',
    'Amortiguamiento');
```

```
    end
```

Por último, se va a explicar la fila de tres botones de arriba que se ilustran en la figura 5.13. Estos actúan como herramientas de la aplicación al igual que los botones de guardado de imágenes.

Figura 5.13: Botones de herramientas.

El botón de la derecha, se encarga de guardar los datos del campo de desplazamientos, velocidades y aceleraciones. El código usado es el siguiente:

```
function ExcelGenericoButtonPushed(app, event)
            nodo = app.SpinnerRespuestaNodo.Value;
        %RESPUESTA Y ERROR
        %Newmark
        xlswrite('DesplazamientosGenerico.xlsx',app.Resultado.
    Newmark.d(nodo,:),'Newmark','A1');
        xlswrite('DesplazamientosGenerico.xlsx',app.Resultado.
    Newmark.dp(nodo,:),'Newmark','A2');
        xlswrite('DesplazamientosGenerico.xlsx',app.Resultado.
    Newmark.dpp(nodo,:),'Newmark','A3');
        xlswrite('DesplazamientosGenerico.xlsx',app.Param.t,'
    Newmark','A5');
        xlswrite('ErrorGenerico.xlsx',app.error.errorN,'Newmark'
    ,'A1');
        xlswrite('ErrorGenerico.xlsx',app.Param.t,'Newmark','A2'
    );
        %Runge-Kutta
        xlswrite('DesplazamientosGenerico.xlsx',app.Resultado.RK
    .x_sol(nodo,:),'Runge-Kutta','A1');
        xlswrite('DesplazamientosGenerico.xlsx',app.Resultado.RK
    .x_sol(nodo+app.Param.gdl,:),'Runge-Kutta','A2');
        xlswrite('DesplazamientosGenerico.xlsx',app.Resultado.RK
    .x_drv(nodo,:),'Runge-Kutta','A3');
```

```
        xlswrite('DesplazamientosGenerico.xlsx',app.Resultado.RK
    .t,'Runge-Kutta','A5');
        xlswrite('ErrorGenerico.xlsx',app.error.errorRK,'Runge-
    Kutta','A1');
        xlswrite('ErrorGenerico.xlsx',app.Resultado.RK.t,'Runge-
    Kutta','A2');
        % Exacto
        xlswrite('DesplazamientosGenerico.xlsx',app.Resultado.
    Exacto.u_exacta(nodo,:),'Solucion Exacta','A1');
        xlswrite('DesplazamientosGenerico.xlsx',app.Resultado.
    Exacto.u_exacta(nodo+app.Param.gdl,:),'Solucion Exacta','A2')
    ;
        xlswrite('DesplazamientosGenerico.xlsx',app.Resultado.
    Exacto.t_exacta','Solucion Exacta','A4');
        end
```

El botón central, es un botón que abre un archivo *.pdf* en el que se indican las instrucciones de uso de la aplicación. Este archivo será adjuntado en los anexos y el código asociado a él es:

```
function InfoButtonEjemploPushed(app, event)
            winopen('GUIA PARA EL USO DE LA APLICACION.pdf')
        end
```

Por último, el botón de la derecha es un botón que sirve para cerrar la aplicación y el código utilizado es el siguiente:

```
function SalidaButtonGenericoPushed(app, event)
            opc=questdlg('Desea salir del programa','Salir','Si'
    ,'No','No');
            if strcmp(opc,'Si')
                delete(app)
            end
        end
```

El cual al pulsarlo hace emerger una vemtana que contiene lo ilustrado en la figura 5.14.

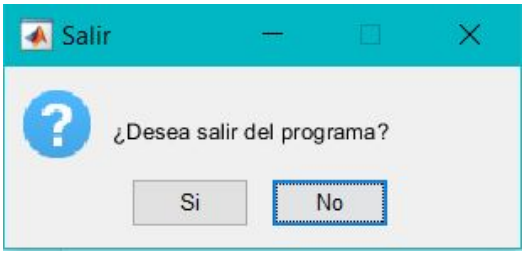

Figura 5.14: Ventana emergente del botón de cerrar aplicación.

5.5.1. Empaquetar aplicaciones en *App Designer*

Con la aplicación terminada, *Matlab* permite a los usuarios empaquetar la aplicación para que sea utilizada por otros usuarios de *Matlab*. De esta forma, se puede compartir de manera sencilla para que sea utilizada por otros usuarios.

Para encontrar la opción de empaquetar, hay que abrir la pestaña 'Designer' y en esa pestaña seleccionar el botón de 'Share' como se observa en la figura 5.15.

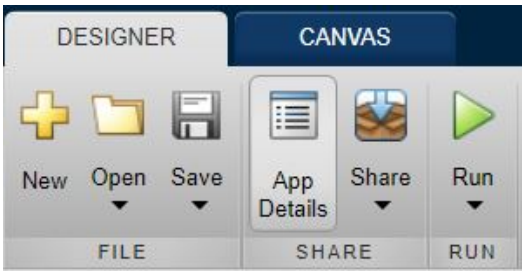

Figura 5.15: Pestaña de 'Designer'.

Dentro del botón 'Share', se selcciona la opción de 'MATLAB App' y se abre un cuadro emergente como el que se observa en la figura 5.16.

En el cuadro emergente permite que el usuario edite:

- El nombre de la aplicación.
- La versión de la aplicación.
- El creador de la aplicación.

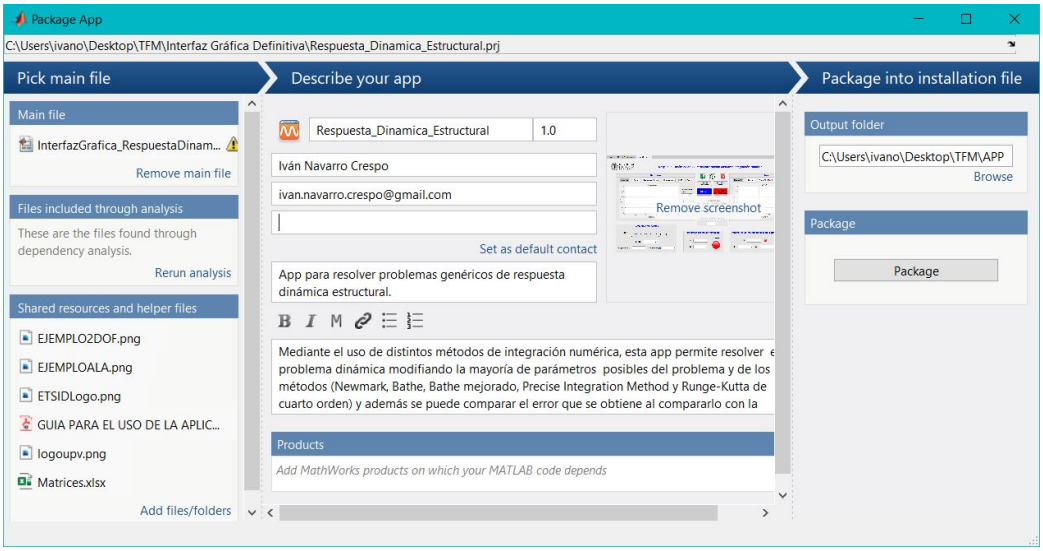

Figura 5.16: Ventana emergente de la empaquetación del App.

- La comapañía, si la hubiera, responsable de la aplicación.
- Un pequeño resumen de la aplicación
- Los productos (*software*) necesarios para correr la aplicación. En caso de que los usuarios no tengan ese *software* instalado, no funcionaría la aplicación.

En el cuadro de la izquierda de 'Main File', es el archivo *.MLAPP* que se ha seleccionado para empaquetar. Bajo él, se encuentra 'Files included through analysis' en donde se incluyen todas las carpetas de *Matlab* detectadas como archivos dependientes. Además, debajo en el cuadro de 'Shared resources and helper files' se pueden añadir más archivos que sean dependientes de la aplicación si no se detectan automáticamente.

A la derecha, en la sección de 'Output Folder', se selecciona la ubicación del ordenador en la que se quiere empaquetar la aplicación. Por último, pulsando en el botón 'Package', se crea el archivo *.mlappinstall*. Además, una vez se ha empaquetado, en la sección de 'apps' de la ventana principal de matlab, se puede instalar la aplicación mediante el botón ilustrado en la figura 5.17 para poder usar la aplicación desde ese panel de forma rápida. Una vez la

aplicación se ha distribuido a otros usuarios, este paso es el que deberían de seguir para tener instalada la aplicación en *Matlab* y poder utilizarla.

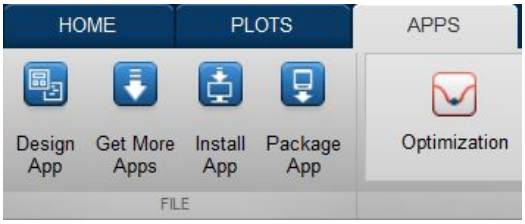

Figura 5.17: Botón de instalación de la aplicación

Capítulo 6

Presupuesto

Una vez se ha desarrollado el trabajo y todos los estudios involucrados están terminados se puede establecer un coste real del proyecto.

En primer lugar, sólo una persona se ha necesitado para llevar a cabo este proyecto. Teniendo en cuenta la experiencia, se va a considerar que las horas trabajadas han sido trabajadas por un ingeniero junior al cual se le atribuirá un sueldo de acuerdo a su puesto. Como materiales de trabajo, se ha utilizado únicamente un ordeandor de alta gama con gran capacidad computacional para efectuar los cálculos. También se ha utilizado el software de *Matlab* provisto por la UPV y el paquete de *Microsoft Office* proporcionado también por la UPV.

Muchos de los documentos bibliográficos se han obtenido a través de Internet de forma gratuita y otros han sido proporcionados por el tutor del trabajo por lo que se va a considerar que el dinero invertido en ellos va a ser nulo.

Se van a incluir también las horas dedicadas por el tutor del trabajo en reuniones con el alumno. En este caso se va a considerar el precio por hora de un catedrático.

Teniendo en cuenta estos aspectos, el presupuesto total se puede ver des-

glosado en la tabla 6

Elemento	**Coste Unitario**	**Cantidad**	**Coste Total (€)**
Ingeniero Aeroespacial Junior	11,8€/hora	338	3988,4
Catedrático de Universidad	19,6€/hora	15	294
Licencia de Matlab	250€	1	250
Licencia de Office	69€	1	69
Bibliografía utilizada	0€	22	0
Ordenador alta gama	1500€	1	1500
		TOTAL	6101,4€

Tabla 6.1: Tabla de presupuesto del proyecto.

El coste total del proyecto, como se ve en la tabla, asciende a los 6101,4 euros.

Capítulo 7

Conclusiones y líneas de trabajo futuro

El aumento de capacidad computacional se ha ido incrementando con los años y ha ayudado de forma muy notable a la aparición de nuevos métodos de resolución de ecuaciones diferenciales. El problema que tenían asociado anteriormente residía en en límite que los métodos tenían para resolver un número indeterminado de operaciones. Con la aparición de la informática, empezaron a emerger nuevos métodos que generaron una revolución en el análisis dinámico estructural (también en otros ámbitos) y mejoraron el conocimiento que se tenían de algunos sucesos como los terremotos o las grandes rachas de vientos.

Los aspectos más importantes que dominan los métodos de integración en el tiempo son el paso temporal elegido y los parámetros asociados a los métodos y el tipo de problema a resolver (variables asociadas al problema). En cuanto al paso temporal (Δt) es el factor que va a determinar cuantas operaciones se van a resolver para un tiempo dado y por lo tanto, cual va a ser el coste computacional que se va a tener al resolver el problema. También es el factor que junto con la frecuencias que se quieran resolver del problema va a determinar la estabilidad de un esquema determinado. Se ha visto, que el factor que más influye en el cálculo del paso temporal es el periodo más pequeño del sistema, ya que el paso temporal como mínimo debería de ser tan pequeño como ese periodo aunque se ha visto que debería ser menor pa-

ra ser conservartivos. Este aspecto se ha corroborado en el ejemplo del ala, donde con un paso temporal grande la respuesta no era capaz de resolver las altas frecuencias de la solución. Los parámetros de los métodos son importantes ya que actúan como parámetros de control de los mismos, y son los que proporcionan determinadas cualidades y diferencian a unos métodos de otros en algunos casos.

Algunos de esos métodos se han visto en este trabajo. El método de Newmark-β se basa en una aproximación muy sencilla, inspirada en antiguos métodos existentes en aquel momento. Como se ha visto, suponía una variación lineal de la aceleración calculado mediante el método de diferencias finitas. Las ventajas que se pueden ver del método de Newmark-β están asociadas a los sitemas que tienen asociados modos con bajas frecuencias, se ha demostrado que es capaz de resolver todos estos modos sin introducir disipación numérica en el problema. Además, el método no genera una pérdida de amplitud de la respuesta. Al ser uno de los primeros métodos en aparecer, se ha visto que tiene varias desventajas. En primer lugar, no hay posibilidad de controlar la disipación numérica. Esto es, que el método no es capaz de resolver bien los modos asociados a altas frecuencias como se vió en el ejemplo del ala con la fuerza constante. En la resolución del campo de velocidades, las respuesta oscilaba entre la solución exacta. En cuanto a la estabilidad del esquema, se ha visto que es un esquema condicionalmente estable y solo cumpliendo una relación entre los parámetros del problema se llega a la estabilidad incondicional. En cuanto a la elección a los parámetros se ha visto que el paso temporal afecta fuertemente a la precisión del esquema, pasando de errores con pasos temporales altos del 40 % en promedio a errores de menos de un 1 % con pasos temporales pequeños y en el caso del ala directamente no conseguía aproximar la solución exacta. También se ha visto que la modificación de los parámetros β y γ lleva a soluciones que aparentemente son similares pero en términos de error se pueden apreciar las diferencias, esto dependerá del problema. En la figura 7.1, se puede observar como se producen las oscilaciones en un problema simulado mediante la aplicación creada de dos nodos cuya diferencia de frecuencias naturales es de

tres órdenes ($\sim$ 1 rad/s y $\sim$ 100rad/s), al igual que ocurría en la figura 4.32.

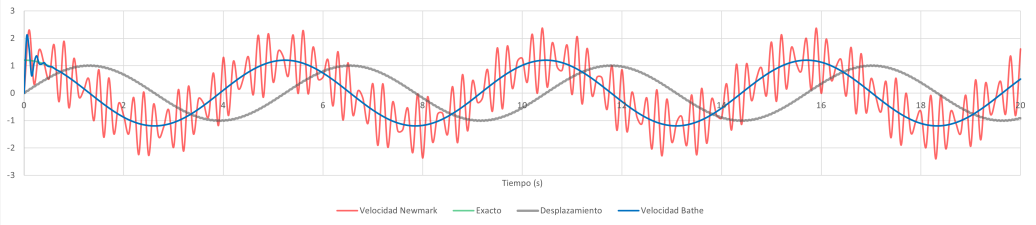

Figura 7.1: Ejemplo de dispersión numérica mediante el método de Newmark y Bathe.

Los métodos de Bathe y Bathe modificado, se han analizado a la vez puesto que la filosofía que siguen es la misma. En estos, al igual que en el método de Newmark-β, se ha observado mayor precisión en la solución al disminuir el paso temporal como era de esperar. Al usar grandes pasos temporales, en ambas se ve como se produce una elongación del periodo de la solución, generando errores en las soluciones como se ha podido ver, pero en ningun caso superior a la que se produce mediante el método de Newmark-β. En la figura 7.2 se observa la comparativa entre el porcentaje de elongación del periodo para Newmark y Bathe [1]. En el caso de Bathe modificado, se han visto para el caso práctico de estudio como algunas de las combinaciones utilizadas mejoran la respuesta del sistema en términos de error y gracias a los nuevos parámetros se tiene un control personalizado de la disipación numérica aunque esta va asociada al tipo de problema que esté siendo analizado. Más concretamente, se han visto beneficios en el caso práctico genérico cuando el subpaso temporal no se dividía de manera equitativa, si no cuando el primer subpaso es más pequeño que el segundo, por lo tanto se resolvía en mayor porcentaje el problema haciendo uso del método de Newmark-β. Sin embargo, esos mismos parámetros utilizados en el caso de la estructura ala no han llevado a resultados del todo satisfactorios a pesar de la complejidad del problema. Sería necesario realizar un estudio por problema de cuál sería el método que mejor se adeucua a las características del problema que quiere resolverse. En cuanto a la estabilidad, se ha visto que el método de Bathe es incondicionalmente estable a diferencia del método de Bathe modificado

que es condicionalmente estable debido a la introducción de los parámetros en el segundo subpaso temporal aunque la estabilidad no sigue las mismas reglas que en el método de Newmark-β. Además, con estos métodos se pude eliminar la disipación numérica que aparecía en Newmark como se ve en la figura 7.1 y como se ha demostrado en el caso de la estructura alar.

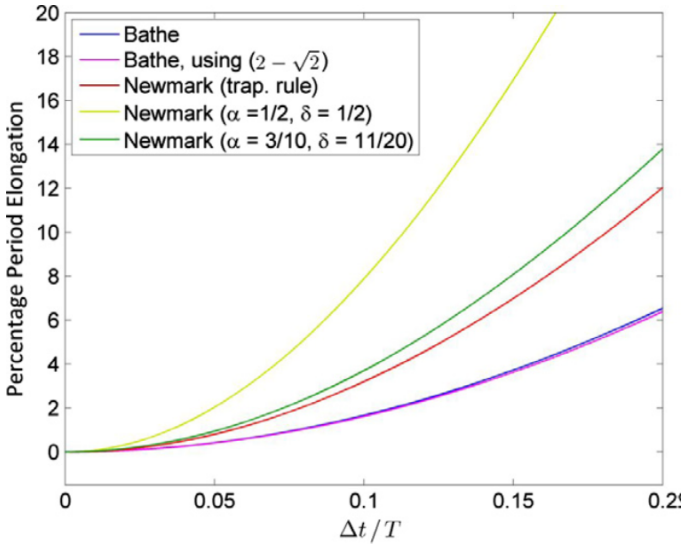

Figura 7.2: Diferencias de porcentaje de elongación del periodo entre el método de Newmark-β y Bathe [2]

Se ha presentado también el *Precise Integration Method* (PIM) que surgió en los años 90 y en este trabajo se ha tratado el modelo clásico, aunque posteriormente han salido versiones distintas basadas en el mismo principio. Se ha visto como el método se basa en la computación de una matriz que va evaluando los incrementos de los subpasos temporales que se generan en el método intencionadamente y que por lo tanto presume de ser un método eficiente. En los resultados vistos del caso práctico genérico, se suponía que la precisión que se alcanzaba con este método debería ser mayor a la alcanzada con Runge-Kutta, sin embargo se ve como no se cumple. Aunque el error es parecido, el del PIM es algo mayor. Además, se observa como el resultado es el mismo para el caso práctico genérico sea cual sea el Valor de N puesto que el caso para no necesitaba del nivel de estabilidad como para que se necesi-

tara un valor de estabilidad y precisión mayor que ofrecen valores de N más altos. En el caso práctico de la estructura alar, se representa la respuesta del método con N=20 únicamente. En la figura 7.3 se puede ver como en este caso el valor de N si tenía importancia, ya que utilizando N=4 la estabilidad del método se sitúa en la zona de $\rho(\mathbf{S}) > 1$ y la solución diverge. Otra de las ventajas asociadas al método, es que el paso temporal necesitado para la estabilidad de este método es mayor que en el caso de Runge-Kutta. Esto es muy útil para problemas donde el coste computacional sea un factor importante, ya que con este método se consigue una solución precisa con un coste computacional bajo. En el caso de la estructura alar, utilizando pasos temporales grandes no se ha conseguido aproximar la respuesta adecuadamente a pesar de las buenas propiedades de aproximación de la solución del método, pero al reducir el paso temporal la solución se ajusta en todos casos a la solución exacta, situación que no ocurría con los métodos de Bathe, Bathe modificado y de Newmark-β.

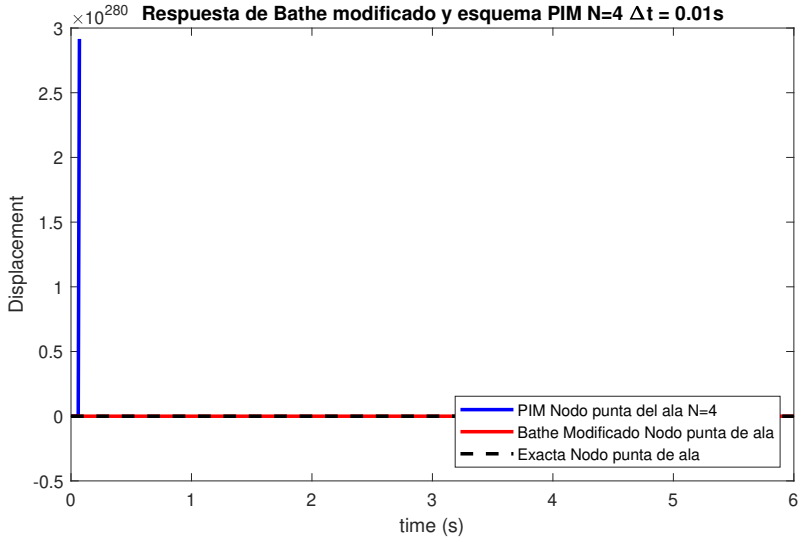

Figura 7.3: Campo de desplazamientos de la punta del ala mediante el PIM con N=4 con $\Delta t = 0.01$s con fuerza constante.

El método de Runge-Kutta se ha visto como el método más preciso de todos en la resolución del caso práctico genérico. Esto es posible gracias a

las múltiples evaluaciones de la solución a lo largo del paso temporal. El único punto negativo que ofrece el método a diferencia del resto, esque su zona de estabilidad es muy pequeña y en función del problema que se esté resolviendo y lleva al requerimientos de pasos temporales muy pequeños que influyen directamente en el coste computacional de la solución al tener que resolver pasos temporales más pequeño. Esto último se ha demostrado en el caso práctico del ala, en donde para poder resolver el problema se necesitaba un paso temporal de Δt=0.000184s el cual era demasiado pequeño para poder resolverlo eficientemente con los medios disponibles y no se ha podido mostrar su respuesta.

En cuanto a la aplicación creada en *Matlab* con *App designer*, se ha demostrado como se puede convertir en una herramienta muy potente para el análisis de problemas dinámicos estructurales. Muchas de las conclusiones incluidas en el trabajo se han logrado y entendido gracias al procesamiento de datos obtenidos mediante la resolución de problemas en esta herramienta. Se observa como la codificación del programa es robusta y se ha provisto de métodos para reducir el error que pudiera introducirse debido al factor humano. Otro aspecto a destacar de la aplicación es que el código base, mostrado en los anexos, es muy sencillo de modificar y se pueden añadir nuevos métodos de forma sencilla para su comparativa. El programa también tiene alguna desventaja o incompletitud. Aunque se ha incluido una barra de estado para ver en qué punto de resolución se encuentra el programa, puede que según el tipo de problema que se esté analizando y del paso temporal elegido, la solución no converja y el problema se quede parado sin avanzar en la solución y este no avise.

7.1. Líneas de trabajo futuro

Respecto a posibles mejoras en el trabajo realizado, se van a explicar pequeñas actualizaciones que se pueden añadir a la aplicación diseñada en *Matlab* con el objetivo de hacerla más manejable, útil y con más funcionalidades para el usuario que la utiliza.

Posibles cambios menores que se pueden añadir son:

- Introducción de las matrices del problema dentro de la ventana del programa.
- Obtención de la respuesta de la velocidad y de la aceleración en la propia ventana.
- Procesado previo a la solución de la estabilidad de los esquemas que se están utilizando con los datos introducidos.
- Nuevos botones que permitan las superposición de soluciones en una misma gráfica para poder comparar directamente en la aplicación sin necesidad de postprocesado en Excel.

Como proyecto más ambicioso, se plantea la creación de un programa de elementos finitos para resolver problemas en 2D mediante el mismo programa que se ha utilizado para generar la estructura alar del Capítulo 4 combinado con la aplicación de Matlab. Con ello se podría resolver tanto casos estáticos, que son sencillos de obtener como se vió en la introducción del trabajo, como casos dinámicos y ver la evolución de las tensiones y los desplazamientos a lo largo de la estructura utilizada y del tiempo.

Como se ha comentado, una de las utilidades del estudio de los métodos y posterior desarrollo de la aplicación es la comparación de estos con nuevos métodos para ver las ventajas e inconvenientes que aportan respecto de los métodos existentes. Es por ello que se va a presentar un método que está aún en fase de diseño avanzada, desarrollado por el profesor de la *Universitat Politècnica de València* (UPV) Mario Lázaro Navarro [18] que tras confirmarse su validez podría implementarse para su comparación.

Se considera el sistema dinámico:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f}(t), \quad \mathbf{u}(0) = \mathbf{u}_0, \quad \dot{\mathbf{u}}(0) = \dot{\mathbf{u}}_0 \tag{7.1}$$

Donde los parámetros considerados son conocidos del resto del trabajo. Se considera la hipótesis de amortiguamiento ligero para interpretar la ecuación anterior desde el punto de vista perturbativo. Para ello, se aplica un parámetro de perturbación ϵ cuyo valor se acota entre el 0 y el 1 y multiplica a la matriz de amortiguamiento. La ecuación queda escrita como:

$$\mathbf{M}\ddot{\mathfrak{u}} + \epsilon\mathbf{C}\dot{\mathfrak{u}} + \mathbf{K}\mathfrak{u} = \mathfrak{f}(t) \tag{7.2}$$

La respuesta se puede escribir como función del parámetro ϵ introducido y el tiempo.

$$\mathfrak{u}(t,\epsilon) = \sum_{n=0}^{\infty} \mathbf{x}^{(n)}(t)\epsilon^n = \mathbf{x}^{(0)}(t) + \mathbf{x}^{(1)}(t)\epsilon + \cdots \tag{7.3}$$

Con ello, se tiene la solución para la ecuación 7.1 ($\epsilon = 1$) como,

$$\mathbf{u}(t) = \mathfrak{u}(t,1) = \sum_{n=0}^{\infty} \mathbf{x}^{(n)}(t) = \mathbf{x}^{(0)}(t) + \mathbf{x}^{(1)}(t) + \mathbf{x}^{(2)}(t) + \cdots \tag{7.4}$$

Se tiene también el caso en el que ϵ es 0 y se da el problema no amortiguado que queda,

$$\mathfrak{u}(t,0) = \mathbf{x}^{(0)}(t) \tag{7.5}$$

La función $\mathbf{x}^{(0)}(t)$ es la solución del problema no amortiguado con las condiciones inciales $\mathbf{u}_0$.

$$\begin{cases} \mathbf{M}\ddot{\mathbf{x}}^{(0)} + \ \mathbf{K}\mathbf{x}^{(0)} = \mathfrak{f}(t) \\ \mathbf{x}^{(0)}(0) = \mathbf{u}_0, \quad \dot{\mathbf{x}}^{(0)}(0) = \mathbf{u}_0 \end{cases} \tag{7.6}$$

El resto de términos (términos *n-ésimos)* se obtienen derivando la ecuación dinámica perturbada. Resolviendo $\mathbf{x}^{(n)}$ se obtiene una ecuación que depende del término anterior. Por ello para $n \geq 1$ se imponen condiciones homogéneas a todos los términos y evaluando en ϵ=1:

$$\begin{cases} \mathbf{M}\ddot{\mathbf{x}}^{(n)} + \mathbf{K}\mathbf{x}^{(n)} = -\mathbf{C}\mathbf{x}^{(n-1)} \\ \mathbf{x}^{(n)}(0) = \mathbf{0}, \quad \dot{\mathbf{x}}^{(0)}(0) = \mathbf{0} \end{cases} \tag{7.7}$$

Se resuelve en primer lugar **la solución a la iteración n = 0** en el problema no amortiguado llamando $\mathbf{x}_\mathbf{k}^{(\mathbf{0})} = \mathbf{x}^{(\mathbf{0})}(\mathbf{t_k})$ con $t_{k+1} = t_k + \Delta t$ con $t_0 = 0$ obteniendo:

$$\begin{array}{l} \mathbf{x}_{k+1}^{(0)} = \mathbf{G}(\Delta t)\mathbf{x}_k^{(0)} + \mathbf{H}(\Delta t)\dot{\mathbf{x}}_k^{(0)} + \int_{t=t_k}^{t_{k+1}} \mathbf{H}\left(t_{k+1} - t\right)\mathbf{M}^{-1}\mathfrak{f}(t)dt \\ \dot{\mathbf{x}}_{k+1}^{(0)} = \dot{\mathbf{G}}(\Delta t)\mathbf{x}_k^{(0)} + \dot{\mathbf{H}}(\Delta t)\dot{\mathbf{x}}_k^{(0)} + \int_{t=t_k}^{t_{k+1}} \mathbf{G}\left(t_{k+1} - t\right)\mathbf{M}^{-1}\mathfrak{f}(t)dt \end{array} \tag{7.8}$$

Las matrices $\mathbf{G}$(t) y $\mathbf{H}$(t) son las matrices de respuesta escalón y respuesta impulso respectivamente y se obtienen de la combinación modal de las respuestas y su valores son los siguientes:

$$\begin{array}{l} [\mathbf{G}(t)] = [\mathbf{\Phi}]\,\mathrm{diag}\left(g_i(t)\right)[\mathbf{\Phi}]^{\mathbf{T}}[\mathbf{M}] \\ [\mathbf{H}(t)] = [\mathbf{\Phi}]\,\mathrm{diag}\left(h_i(t)\right)[\mathbf{\Phi}]^{\mathbf{T}}[\mathbf{M}] \end{array} \tag{7.9}$$

Donde $[\mathbf{\Phi}]$ es la matriz que contiene los autovectores del problema y $g_i(t)$ y $h_i(t)$ toman el siguiente valor:

$$\begin{array}{l} g_i(t) = \exp\left(-\xi_i\omega_i t\right)\left(\cos\left(\omega_{\mathrm{di}}t\right) + \frac{\xi_i\omega_i}{\omega_{\mathrm{di}}}\sin\left(\omega_{\mathrm{di}}t\right)\right) \\ h_i(t) = \exp\left(-\xi_i\omega_i t\right)\left(\frac{1}{\omega_{\mathrm{di}}}\sin\left(\omega_{\mathrm{di}}t\right)\right) \quad \text{y} \quad \omega_{\mathrm{di}} = \omega_i\sqrt{1-\xi_i^2} \end{array} \tag{7.10}$$

Las matrices de respuesta cumplen las siguientes identidades,

$$\begin{array}{l} \dot{\mathbf{G}}(t) = -\mathbf{A}\mathbf{H}(t), \quad \mathbf{H}(0) = \mathbf{0} \\ \dot{\mathbf{H}}(t) = \mathbf{G}(t), \quad \mathbf{G}(0) = \mathbf{I}_N \end{array} \tag{7.11}$$

donde $\mathbf{A} = \mathbf{M}^{-1}\mathbf{K}$. Ambas matrices cumplen la ecuación diferencial $\mathbf{M}\ddot{\mathbf{Z}} + \mathbf{K}\mathbf{Z} = \mathbf{0}$, donde $\mathbf{Z}(t) \in \mathbb{R}^{N\times N}$. Ambas funciones se pueden expresar como función de matrices

$$\mathbf{G}(t) = \cos(\sqrt{\mathbf{A}}t), \quad \mathbf{H}(t) = \mathbf{A}^{-1/2}\sin(\sqrt{\mathbf{A}}t) \tag{7.12}$$

Se interpola la función de la fuerza $\mathfrak{f}$ en p puntos del intervalo $[t_k, t_{k+1}]$ de la siguiente forma:

$$\mathfrak{f}(t) \approx \sum_{i=1}^{p} \mathcal{L}_i(t) \mathfrak{f}\left(t_k + \frac{(i-1)\Delta t}{p-1}\right) \tag{7.13}$$

Donde $\mathcal{L}_i(t)$ son polinomios de interpolación de Lagrange. Introduciendo esta expresión en las integrales de la ecuación 7.8 y operando se obtiene:

$$\int_{t=t_k}^{t_{k+1}} \mathbf{H}\left(t_{k+1}-t\right) \mathbf{M}^{-1} \mathfrak{f}(t) dt = \sum_{i=1}^{p} \mathbf{L}_{ui} \mathbf{M}^{-1} \mathfrak{f}\left(t_k + \frac{(i-1)\Delta t}{p-1}\right) \tag{7.14}$$

$$\int_{t=t_k}^{t_{k+1}} \mathbf{G}\left(t_{k+1}-t\right) \mathbf{M}^{-1} \mathfrak{f}(t) dt = \sum_{i=1}^{p} \mathbf{L}_{vi} \mathbf{M}^{-1} \mathfrak{f}\left(t_k + \frac{(i-1)\Delta t}{p-1}\right) \tag{7.15}$$

donde,

$$\mathbf{L}_{ui} = \int_{t=t_k}^{t_{k+1}} \mathbf{H}\left(t_{k+1}-t\right) \mathcal{L}_i(t) dt \quad , \quad \mathbf{L}_{vi} = \int_{t=t_k}^{t_{k+1}} \mathbf{G}\left(t_{k+1}-t\right) \mathcal{L}_i(t) dt, \quad 1 \leq i \leq p \tag{7.16}$$

Por tanto se puede obtener un sistema iterativo que se escribe como:

$$\mathbf{X}_{k+1}^{(0)} = \mathbf{T}\mathbf{X}_k^{(0)} + \mathbf{L}\mathbf{g}_k \tag{7.17}$$

de donde,

$$\begin{gathered} \mathbf{X}_k^{(0)} = \left\{ \begin{array}{c} \mathbf{x}_k^{(0)} \\ \dot{\mathbf{x}}_k^{(0)} \end{array} \right\}, \quad \mathbf{T} = \left[\begin{array}{cc} \mathbf{G}(\Delta t) & \mathbf{H}(\Delta t) \\ \dot{\mathbf{G}}(\Delta t) & \dot{\mathbf{H}}(\Delta t) \end{array} \right] \\ \mathbf{L} = \left[\begin{array}{ccc} \mathbf{L}_{u1} & \cdots & \mathbf{L}_{up} \\ \mathbf{L}_{v1} & \cdots & \mathbf{L}_{vp} \end{array} \right], \quad \mathbf{g}_k = \left\{ \begin{array}{c} \mathbf{M}^{-1}\mathfrak{f}\left(t_k\right) \\ \vdots \\ \mathbf{M}^{-1}\mathfrak{f}\left(t_{k+1}\right) \end{array} \right\} \end{gathered} \tag{7.18}$$

Se resuelve en los siguientes pasos **la iteración con n≥1**. Se reescribe $x_k^{(n)} = x^{(n)}, \dot{x}_k^{(n)} = \dot{x}^{(n)}$ y la solución a la ecuación 7.7 en cada paso temporal es:

$$\begin{aligned}\mathbf{x}_{k+1}^{(n)} &= \mathbf{G}(\Delta t)\mathbf{x}_k^{(n)} + \mathbf{H}(\Delta t)\dot{\mathbf{x}}_k^{(n)} - \int_{t=t_k}^{t_{k+1}} \mathbf{H}\left(t_{k+1}-t\right)\mathbf{M}^{-1}\mathbf{C}\dot{\mathbf{x}}^{(n-1)}(t)dt \\ \dot{\mathbf{x}}_{k+1}^{(n)} &= \dot{\mathbf{G}}(\Delta t)\mathbf{x}_k^{(n)} + \dot{\mathbf{H}}(\Delta t)\dot{\mathbf{x}}_k^{(n)} - \int_{t=t_k}^{t_{k+1}} \mathbf{G}\left(t_{k+1}-t\right)\mathbf{M}^{-1}\mathbf{C}\dot{\mathbf{x}}^{(n-1)}(t)dt\end{aligned} \tag{7.19}$$

Integrando por partes para expresar las integrales sin incluir la derivada y haciendo el cambio de variable de $t = t_k + \Delta t\xi$ con $0 \leq \xi \leq 1$:

$$\begin{aligned}-\int_{t=t_k}^{t_{k+1}} \mathbf{H}\left(t_{k+1}-t\right)\mathbf{M}^{-1}\mathbf{C}\dot{\mathbf{x}}^{(n-1)}(t)dt &= \mathbf{H}(\Delta t)\mathbf{M}^{-1}\mathbf{C}\mathbf{x}_k^{(n-1)} \\ &\quad - \Delta t\int_{\xi=0}^{1} \mathbf{G}[\Delta t(1-\xi)]\mathbf{M}^{-1}\mathbf{C}\mathbf{x}^{(n-1)}\left(t_k+\xi\Delta t\right)d\xi \\ -\int_{t=t_k}^{t_{k+1}} \mathbf{G}\left(t_{k+1}-t\right)\mathbf{M}^{-1}\mathbf{C}\dot{\mathbf{x}}^{(n-1)}(t)dt &= -\mathbf{G}(0)\mathbf{M}^{-1}\mathbf{C}\mathbf{x}_{k+1}^{(n-1)} + \mathbf{G}(\Delta t)\mathbf{M}^{-1}\mathbf{C}\mathbf{x}_k^{(n-1)} \\ &\quad + \Delta t\mathbf{A}\int_{\xi=0}^{1} \mathbf{H}[\Delta t(1-\xi)]\mathbf{M}^{-1}\mathbf{C}\mathbf{x}^{(n-1)}\left(t_k+\xi\Delta t\right)d\xi\end{aligned} \tag{7.20}$$

En estas ecuaciones, $\mathbf{x}^{(n-1)}(t)$ se asume conocida puesto que viene el resultado de una iteración anterior a la cual se está resolviendo en este paso. Para obtener la solución analítica a las integrales, se aproxima la solución mediante polinomios de interpolación con la forma:

$$\begin{aligned}\mathcal{N}_1(\xi) &= (1-\xi)^2(1+2\xi) \\ \partial\mathcal{N}_1(\xi) &= (1-\xi)^2\xi \\ \mathcal{N}_2(\xi) &= (3-2\xi)\xi^2 \\ \partial\mathcal{N}_2(\xi) &= -(1-\xi)\xi^2\end{aligned} \tag{7.21}$$

y la aproximación de $x^{(n-1)}(t)$ quedaría,

$$\mathbf{x}^{(n-1)}\left(t_k+\xi\Delta t\right) \approx \mathcal{N}_1(\xi)\mathbf{x}_k^{(n-1)} + \partial\mathcal{N}_1(\xi)\Delta t\dot{\mathbf{x}}_k^{(n-1)} + \mathcal{N}_2(\xi)\mathbf{x}_{k+1}^{(n-1)} + \partial\mathcal{N}_2(\xi)\Delta t\dot{\mathbf{x}}_{k+1}^{(n-1)} \tag{7.22}$$

Introduciendo 7.22 en 7.20 se obtiene:

$$-\int_{t=t_k}^{t_{k+1}} \mathbf{H}\,(t_{k+1}-t)\mathbf{M}^{-1}\mathbf{C}\dot{\mathbf{x}}^{(n-1)}(t)dt = \tag{7.23}$$

$$\alpha_{uu}\mathbf{x}_k^{(n-1)} + \alpha_{uv}\dot{\mathbf{x}}_k^{(n-1)} + \beta_{uu}\mathbf{x}_{k+1}^{(n-1)} + \beta_{uv}\dot{\mathbf{x}}_{k+1}^{(n-1)} \tag{7.24}$$

$$\tag{7.25}$$

$$-\int_{t=t_k}^{t_{k+1}} \mathbf{G}\,(t_{k+1}-t)\mathbf{M}^{-1}\mathbf{C}\dot{\mathbf{x}}^{(n-1)}(t)dt = \tag{7.26}$$

$$\alpha_{vu}\mathbf{x}_k^{(n-1)} + \alpha_{vv}\dot{\mathbf{x}}_k^{(n-1)} + \beta_{vu}\mathbf{x}_{k+1}^{(n-1)} + \beta_{vv}\dot{\mathbf{x}}_{k+1}^{(n-1)} \tag{7.27}$$

donde,

$$\begin{aligned}
\alpha_{uu} &= \mathbf{H}(\Delta t)\mathbf{M}^{-1}\mathbf{C} - \Delta t\int_{\xi=0}^{1}\mathbf{G}[\Delta t(1-\xi)]\mathbf{M}^{-1}\mathbf{C}\mathcal{N}_1(\xi)d\xi \\
\alpha_{uv} &= -\Delta t^2\int_{\xi=0}^{1}\mathbf{G}[\Delta t(1-\xi)]\mathbf{M}^{-1}\mathbf{C}\partial\mathcal{N}_1(\xi)d\xi \\
\alpha_{vu} &= \mathbf{G}(\Delta t)\mathbf{M}^{-1}\mathbf{C} + \Delta t\ \mathbf{A}\int_{\xi=0}^{1}\mathbf{H}[\Delta t(1-\xi)]\mathbf{M}^{-1}\mathbf{C}\mathcal{N}_1(\xi)d\xi \\
\alpha_{vv} &= \Delta t^2\ \mathbf{A}\int_{\xi=0}^{1}\mathbf{H}[\Delta t(1-\xi)]\mathbf{M}^{-1}\mathbf{C}\partial\mathcal{N}_1(\xi)d\xi \\
\beta_{uu} &= -\Delta t\int_{\xi=0}^{1}\mathbf{G}[\Delta t(1-\xi)]\mathbf{M}^{-1}\mathbf{C}\mathcal{N}_2(\xi)d\xi \\
\beta_{uv} &= -\Delta t^2\int_{\xi=0}^{1}\mathbf{G}[\Delta t(1-\xi)]\mathbf{M}^{-1}\mathbf{C}\partial\mathcal{N}_2(\xi)d\xi \\
\beta_{vu} &= -\mathbf{M}^{-1}\mathbf{C} + \Delta t\ \mathbf{A}\int_{\xi=0}^{1}\mathbf{H}[\Delta t(1-\xi)]\mathbf{M}^{-1}\mathbf{C}\mathcal{N}_2(\xi)d\xi \\
\beta_{vv} &= \Delta t^2\ \mathbf{A}\int_{\xi=0}^{1}\mathbf{H}[\Delta t(1-\xi)]\mathbf{M}^{-1}\mathbf{C}\partial\mathcal{N}_2(\xi)d\xi
\end{aligned} \tag{7.28}$$

Con esto puede escribirse un esquema recursivo para la iteración n de forma compactada como:

$$\mathbf{X}_{k+1}^{(n)} = \mathbf{T}\mathbf{X}_k^{(n)} + \boldsymbol{\alpha}\mathbf{X}_k^{(n-1)} + \boldsymbol{\beta}\mathbf{X}_{k+1}^{(n-1)}, \quad n \geq 1 \tag{7.29}$$

donde,

$$\mathbf{X}_k^{(n)} = \left\{ \begin{array}{c} \mathbf{x}_k^{(n)} \\ \dot{\mathbf{x}}_k^{(n)} \end{array} \right\}, \quad \mathbf{T}(\Delta t) = \left[\begin{array}{cc} \mathbf{G}(\Delta t) & \mathbf{H}(\Delta t) \\ \dot{\mathbf{G}}(\Delta t) & \dot{\mathbf{H}}(\Delta t) \end{array} \right] \tag{7.30}$$

$$\boldsymbol{\alpha} = \left[\begin{array}{cc} \boldsymbol{\alpha}_{uu} & \boldsymbol{\alpha}_{uv} \\ \boldsymbol{\alpha}_{vu} & \boldsymbol{\alpha}_{vv} \end{array} \right], \quad \boldsymbol{\beta} = \left[\begin{array}{cc} \boldsymbol{\beta}_{uu} & \boldsymbol{\beta}_{uv} \\ \boldsymbol{\beta}_{vu} & \boldsymbol{\beta}_{vv} \end{array} \right] \tag{7.31}$$

Las condiciones iniciales ya están aplicadas en la iteración en $n = 0$ y por tanto se tiene para $n \geq 1$, $\mathbf{X^{(n)}}$=$\mathbf{0}$ y si se expresan las ecuaciones 7.29 representando todos los instantes de tiempo se obtiene:

$$\left[\begin{array}{ccccc} \mathbf{I}_{2N} & \mathbf{0}_{2N} & \cdots & \mathbf{0}_{2N} & \mathbf{0}_{2N} \\ -\mathbf{T} & \mathbf{I}_{2N} & \cdots & \mathbf{0}_{2N} & \mathbf{0}_{2N} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0}_{2N} & \mathbf{0}_{2N} & \cdots & -\mathbf{T} & \mathbf{I}_{2N} \end{array} \right] \left\{ \begin{array}{c} \mathbf{X}_1^{(n)} \\ \mathbf{X}_2^{(n)} \\ \vdots \\ \mathbf{X}_k^{(n)} \end{array} \right\} = \left[\begin{array}{ccccc} \boldsymbol{\beta} & \mathbf{0}_{2N} & \cdots & \mathbf{0}_{2N} & \mathbf{0}_{2N} \\ \boldsymbol{\alpha} & \boldsymbol{\beta} & \cdots & \mathbf{0}_{2N} & \mathbf{0}_{2N} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0}_{2N} & \mathbf{0}_{2N} & \cdots & \alpha & \boldsymbol{\beta} \end{array} \right] \left\{ \begin{array}{c} \mathbf{X}_1^{(n-1)} \\ \mathbf{X}_2^{(n-1)} \\ \vdots \\ \mathbf{X}_k^{(n-1)} \end{array} \right\} \tag{7.32}$$

Y se puede escribir un esquema recursivo como:

$$\mathbf{X}^{(n)} = \mathbf{S}\mathbf{X}^{(n-1)} \tag{7.33}$$

Este tipo de ecuación ya se mencionó en el Capítulo 2 cuando se habló de la estabilidad de los esquemas de integración. En este caso, de la misma manera, si el radio espectral de la matriz $\rho(\mathbf{S}) < 1$, entonces los términos $\mathbf{X}_k^{(n)}$ están en progresión geométrica descendente y el esquema es estable y convergente para todo k = 1,2,3,... . Los autovalores del esquema recursivo son los mismos que los asociados a la matriz $\boldsymbol{\beta}$ por lo que la condición necesaria para que la serie generada por la perturbación del amortiguamiento sea convergente es,

$$\rho(\boldsymbol{\beta}) < 1 \tag{7.34}$$

Considerando la solución a la ecuación diferencial cogiendo hasta el término n de la serie,

$$\mathbf{U}_k^{(n)} = \left\{ \begin{array}{c} \mathbf{u}_k^{(n)} \\ \dot{\mathbf{u}}_k^{(n)} \end{array} \right\} = \mathbf{X}_k^{(0)} + \mathbf{X}_k^{(1)} + \cdots + \mathbf{X}_k^{(n)} = \sum_{l=0}^{n} \mathbf{X}_k^{(l)} \tag{7.35}$$

Si ahora se coge lo obtenido en la solución a las iteraciones y se expresan las

mismas desde $0 \leq l \leq n$,

$$\begin{aligned} \mathbf{X}_{k+1}^{(0)} =& \mathbf{T}\mathbf{X}_{k}^{(0)} + \mathbf{L}\mathbf{g}_k \\ -\boldsymbol{\beta}\mathbf{X}_{k+1}^{(0)} + \mathbf{X}_{k+1}^{(1)} =& \mathbf{T}\mathbf{X}_{k}^{(1)} + \boldsymbol{\alpha}\mathbf{X}_{k}^{(0)} \\ & \vdots \\ -\boldsymbol{\beta}\mathbf{X}_{k+1}^{(n-1)} + \mathbf{X}_{k+1}^{(n)} =& \mathbf{T}\mathbf{X}_{k}^{(n)} + \boldsymbol{\alpha}\mathbf{X}_{k}^{(n-1)} \end{aligned} \tag{7.36}$$

Se obtiene si $n \geq 1$:

$$-\beta\mathbf{U}_{k+1}^{(n-1)} + \mathbf{U}_{k+1}^{(n)} = \mathbf{T}\mathbf{U}_{k}^{(n)} + \alpha\mathbf{U}_{k}^{(n-1)} + \mathbf{L}\mathbf{g}_k \tag{7.37}$$

Teniendo en cuenta que $\mathbf{U}_k = \lim_{n\to\infty} \mathbf{U}_k^{(n)}$, se obtiene tomando límites en la ecuación anterior:

$$(\mathbf{I}_{2N} - \boldsymbol{\beta})\,\mathbf{U}_{k+1} = (\mathbf{T} + \boldsymbol{\alpha})\mathbf{U}_k + \mathbf{L}\mathbf{g}_k \tag{7.38}$$

Finalmente, el esquema recursivo para cada instante de tiempo puede escribirse como:

$$\mathbf{U}_{k+1} = \mathbf{a}\mathbf{U}_k + \mathbf{b}\mathbf{g}_k \tag{7.39}$$

con las matrices $\mathbf{a}$ y $\mathbf{b}$ que toman los siguientes valores:

$$\mathbf{a} = (\mathbf{I}_{2N} - \boldsymbol{\beta})^{-1}\,(\mathbf{T} + \boldsymbol{\alpha}), \quad \mathbf{b} = (\mathbf{I}_{2N} - \boldsymbol{\beta})^{-1}\,\mathbf{L} \tag{7.40}$$

Bibliografía

[1] Malakiyeh M.M., Shojaee S, and Bathe K.J. The Bathe time integration method revisited for prescribing desired numerical dissipation. *Computers and Structures*, 212:289–298, 2019.

[2] Bathe K. J. and Noh G. Insight into an implicit time integration scheme for structural dynamics. *Computers and Structures*, 98-99:1–6, 2012.

[3] Bathe K.J. and Wilson E.L. Numerical methods in finite element, 1976.

[4] Newmark N.M. A Method of Computation for Structural Dynamics. *Engineering Mechanics Division*, 1959.

[5] Hilber H.M., Hughes Thomas J.R., and Taylor R.L. Improved numerical dissipation for time integration algorithms in structural dynamics. *Earthquake Engineering and Structural Dynamics*, 5:283–292, 1977.

[6] Kadapa C., Dettmer W.G., and Perić D. On the advantages of using the first-order generalised-alpha scheme for structural dynamic problems. *Computers and Structures*, 193:226–238, 2017.

[7] Avitable P. Experimental modal analysis. *Sound and Vibration*, 2001.

[8] Bathe K.J. *Finite Element Procedures*. 2006.

[9] Zhong W.X. On precise integration method. *Journal of Computational and Applied Mathematics*, 163(1):59–78, 2004.

[10] Zhong W.X. and Williams F.W. A precise time step integration method. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 208(6):427–430, 1994.

[11] Chuang C.C. and Wu C. Application of an Innovative Precise Integration Method in Solving Equilibrium Equations of Motion for Structural Dynamic Problems. *15th World Conference on Earthquake Engineering (15WCEE)*, 2012.

[12] Butcher J.C. *Numerical Methods for Ordinary Differential Equations*. 2016.

[13] Vadillo F. Métodos de un paso, Métodos de Runge-Kutta, 2019.

[14] Craig J.I. and Bachau O.A. *Structural Analysis with applications to Aerospace Structures*. 2009.

[15] Airbus. *Aircraft Characteristics Airport and Maintenance Planning (A320)*. 2005.

[16] Niu Michael C.Y. *Airframe Structural Design*. 1995.

[17] Clough R.W. and Penzien J. *Dynamics of structures*. 2003.

[18] Lazaro M. Homotopy analysis of transient problems using artificial perturbation of damping. 2020.

[19] Chaves Eduardo W.V. *Integración numérica en el tiempo*. 2010.

[20] Cid G. *Programación de Interfaz gráfica en App Designer para el control vectorial de imanes permanentes*. 2018.

[21] Gavin H.P. Numerical Integration in Structural Dynamics. *CEE 541. Structural Dynamics*, 2018.

[22] Garcia-Fogeda P. and Sanz Andres A. *Introducción a las vibraciones*. Garceta, 1 edition, 2014.

Anexo

Documento de instrucciones del funcionamiento de la aplicación en 'App Designer'

GUIA PARA EL USO DE LA APLICACIÓN 'RESPUESTA DINÁMICA ESTRUCTURAL'

En la presente guía se quiere dar unas instrucciones de uso para la aplicación diseñada mediante el entorno de MATLAB de *'Respuesta dinámica estructural'*. La aplicación surge como una idea para complementar un Trabajo de Fin de Máster (TFM) y está orientada a ser una herramienta didáctica en un principio, pero tiene una gran escalabilidad al tener métodos de elementos finitos implementados en su '*core*'.

Al abrir la aplicación, en la parte superior se pueden distinguir 3 pestañas. Las pestañas 'Ejemplo 2 GDL' y 'Ala avión', corresponden a la resolución de dos ejemplos que se han utilizado en el TFM y que se han diseñado para obtener resultados de estos haciendo uso de la aplicación.

La pestaña 'Caso genérico' es en la que se va a centrar la guía y las pestañas de ejemplos seguirán los mismos pasos y los pasos indicados que no se encuentren en estas pestañas simplemente se omitirán. La pantalla que nos encontraremos en la pestaña del caso general tendrá el siguiente aspecto.

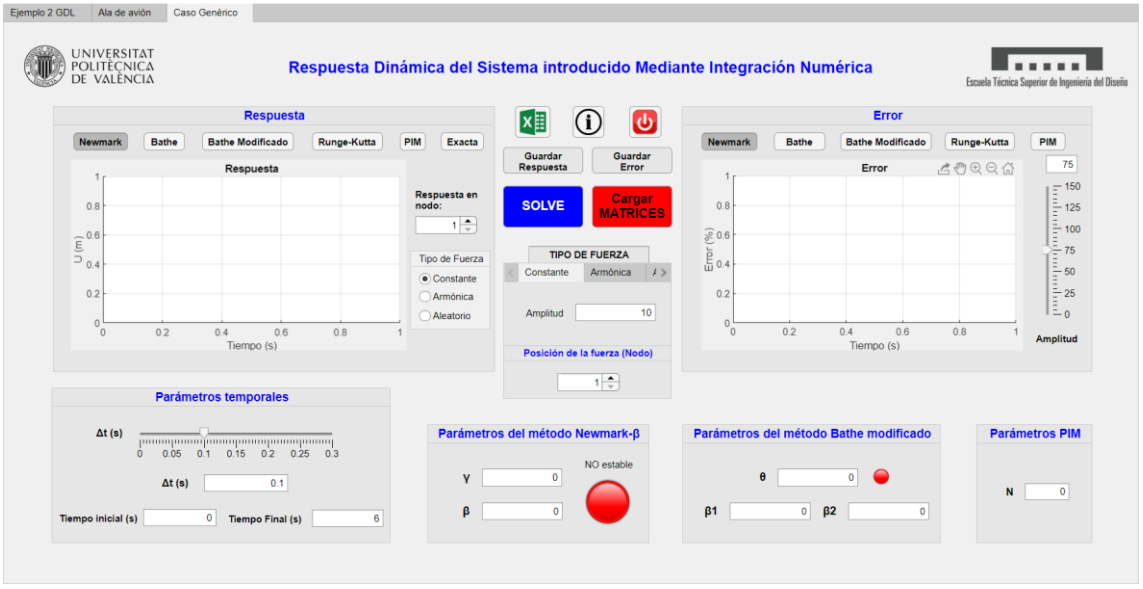

Ilustración 1: Pestaña del caso genérico

Los pasos por realizar para su correcta ejecución se enumeran a continuación:

- En primer lugar, al abrir Matlab se debe seleccionar el '*Path*' donde se van a querer dejar guardados todos los documentos. En ese mismo '*Path*' será necesario un archivo XLS que se deberá llamar ***Matrices.xlsx*** (comprobar que la versión de Excel utilizada es compatible) y deberá rellenarse con las matrices del sistema que se quiera resolver con el siguiente formato, en donde la posición (1,1) de la matriz se corresponde con la celda A1 en todas las pestañas.

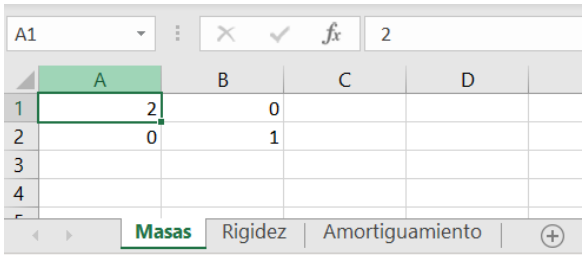

Ilustración 2: Formato de la hoja de Excel para introducir las matrices

- Con ese archivo en el '*Path*' de MATLAB, para cargar las matrices simplemente hay que pulsar el botón de ***Cargar MATRICES***.
- El siguiente paso es modificar en la zona inferior los parámetros de los métodos de integración y del paso temporal al gusto del usuario. Nótese que dependiendo del problema utilizar un paso temporal muy pequeño puede ralentizar la solución de forma considerable. Hay unos indicadores rojos los cuales deben de estar en verde cuando modifiquemos las variables para que no haya problemas en la resolución, pero aun así se debe de rellenar todos los cuadros con los parámetros.
- En la zona central se encuentra un panel con el **tipo de fuerza** que se le puede aplicar y que se puede seleccionar en un panel que está dentro del panel de RESPUESTA con la gráfica. Una vez seleccionada la fuerza, en el panel central se busca la pestaña asociada a la fuerza y se le modifica en el/los cuadro/s el parámetro.
- Se selecciona el nodo en el que se quiere que se aplique la fuerza y el nodo del cual queremos obtener la respuesta en los paneles de **POSICION DE LA FUERZA** y **RESPUESTA EN NODO** respectivamente.
- Por último, para resolver se clica al botón ***SOLVER*** de donde se obtienen los resultados.
- Los resultados se obtienen en las gráficas. Para ver los resultados de forma correcta, se puede ver como por defecto el **Botón de Newmark** está pulsado y por ello nos muestra su resultado, se deberá de **despulsar** el Botón de Newmark lo que hará que se borre la gráfica y pulsar el botón del método que se quiere resolver para que aparezca su gráfica asociada. Este proceso puede aplicarse tanto a la gráfica de respuesta como a la gráfica de error.
- En la gráfica de error hay un '*slider*' que se puede ir modificando e indica la amplitud del Eje del error de la gráfica para obtener mayor o menor resolución en error.

Para finalizar, se han incluido un botón con un icono de Excel, el cual al pulsarlo exporta los resultados obtenidos a un archivo de Excel que se guardará en el 'Path' de Matlab seleccionado. Además, hay dos botones identificados como **GUARDAR ERROR** Y **GUARDAR RESPUESTA**, con los cuales se guarda un archivo .fig y .jpg con las gráficas mostradas en ese momento. Para finalizar, se encuentran los botones de información, que genera este PDF de guía del APP y un botón de salida para salir de la aplicación.

Funciones asociadas a la aplicación en 'App Designer'. Código del Botón 'SOLVE' de la aplicación

Algoritmos de la definición de las fuerzas involucradas en el problema

```
function h = unitstep(app,t)

            %Funcion escalon unitario
            size_t = size(t);
            h = zeros(size_t);
            id = (t >= 0);
            h(id) = 1;
        end



function [fe,t_f] = fuerzaRandom(app,caso)

            %Funcion para la definicion de una fuerza aleatoria
            %f(t)=e^{-(t-tm)^2/(2sm^2) * fa * sin(wa*t)*U(t-Te)
            tm = 2;
            sm = 0.3*tm;

            %Intervalo de tiempo (cada dt lo subdividimos en
    partes:
            %para captar con precision los valores (inicio, a
    dt/3, a 2dt/3,
            %a final=dt)
            %k, k+1/3, k+2/3, k+1
            dt_f = app.Param.dt/40;

            %Serie de datos temporales para la fuerza
            t_f = (app.Param.ti:dt_f:app.Param.tf)';

            %Inicializacion
            fe = zeros(size(t_f));
```

```
            % Frecuencias, amplitudes e intervalos de aplicacion

            switch caso

                case 'Ejemplo'

                    wa = [0.8, 2, 7, 12];

                case 'Ala'

                    wa = [0.5, 1.2, 5, 10];

            end

            fa = [0.5, 0.8, 0.6, 0.5];
            Te = [16*pi, 16*pi, 16*pi, 16*pi];
            NumArmonicos = length(wa);

            for a = 1:NumArmonicos
                fe = fe + fa(a) * sin(wa(a)*t_f).*(1 - unitstep(
    app,t_f-Te(a)));
            end

            % Lo afectamos por una gaussiana
            fe = exp(-(t_f - tm).^2 / (2*sm^2)) .* fe;
        end
```

Algoritmo del método de Newmark-β

```
function [d,dp,dpp] = NewmarkMethod_ejemplo(app,beta,gamma,caso,
    metodo)

            % Inputs
            %------
            %
            % app: propierdades (Param y matrices)
            % beta: Constante del metodo
```

```
        %gamma: Constante del metodo
        %caso: Selecciona que tipo de excitacion se tiene
en el problema
        %metodo: tipo de fuerza utilizada
        %
        %Outputs
        %-------
        %d, dp, dpp: Campos de desplazamientos, velocidades
 y aceleracion
        %respectivamente

        %Parametros del metodo

        b1 = 1/(beta*app.Param.dt.^2);
        b2 = -1/(beta*app.Param.dt);
        b3 = 1-1/(2*beta);
        b4 = gamma*app.Param.dt*b1;
        b5 = 1+gamma*app.Param.dt*b2;
        b6 = app.Param.dt*(1-gamma+gamma*b3);

        %Condiciones iniciales

        d   = zeros(app.Param.gdl,app.Param.nt);
        dp  = zeros(app.Param.gdl,app.Param.nt);
        dpp = zeros(app.Param.gdl,app.Param.nt);
        F   = zeros(app.Param.gdl,app.Param.nt+1);

        d(:,1)  = zeros(app.Param.gdl,1); %Desplazamiento
inicial
        dp(:,1) = zeros(app.Param.gdl,1); %Velocidad incial

        %Matriz de Rigidez efectiva [Keff]

        Keff = app.Matrices.K+b1*app.Matrices.M+b4*app.
Matrices.C;

        %Se obtiene la aceleracion inicial en funcion de la
 fuerza
```

```
            switch caso
                case 'Ejemplo'
                    k=20;
                    if app.Nodo1Button.Value == 1
                        pos = 1;
                    else
                        pos = 2;
                    end
                    [fe,tf] = fuerzaRandom(app,'Ejemplo');
                    fe = k*interp1(tf,fe,app.Param.t,'spline');
                case 'Ala'
                    k=10000;
                    pos = app.SpinnerFuerzaNodo.Value;
                    [fe,tf] = fuerzaRandom(app,'Ala');
                    fe = k*interp1(tf,fe,app.Param.t,'spline');

                case 'Generico'
                    k=app.AmplitudGenericoEditField.Value;
                    pos = app.SpinnerFuerzaNodoGenerico.Value;
                    [fe,tf] = fuerzaRandom(app,'Ejemplo');
                    fe = k*interp1(tf,fe,app.Param.t,'spline');
            end


            %Resuelve

            switch metodo

                case 'Constante'

                    for i = 1:app.Param.nt
                        F(pos,i) = app.AmplitudFuerzaCteGenerico
.Value;
                    end
                    F0 = F(:,1);
                    dpp(:,1) = app.Matrices.M\(F0-app.Matrices.K
*d(:,1)-app.Matrices.C*dp(:,1));

                case 'Sinusoidal'
```

```
                F0    = zeros(app.Param.gdl,1);
                dpp(:,1) = app.Matrices.M\(F0-app.Matrices.K
*d(:,1)-app.Matrices.C*dp(:,1));
                F(pos,:)    = app.AmplitudSinusoidalGenerico.
Value*...
                    sin(app.Param.t*app.
FrecuenciaSinusoidalGenerico.Value);

            case 'Aleatorio'

                F0        = zeros(app.Param.gdl,1);
                dpp(:,1) = app.Matrices.M\(F0-app.Matrices.K
*d(:,1)-app.Matrices.C*dp(:,1));
                F(pos,:) = fe;
        end

        %Se crea el bucle temporal para resolver el paso [t
+t*dt]

        Feff = zeros(app.Param.gdl,app.Param.nt);

        for i = 1:app.Param.nt

            Feff(:,i)    = F(:,i+1)+app.Matrices.M*(b1*d(:,i)
-b2*dp(:,i)-b3*dpp(:,i))+...
                app.Matrices.C*(b4*d(:,i)-b5*dp(:,i)-b6*dpp
(:,i));
            d(:,i+1)     = Keff\Feff(:,i);
            dpp(:,i+1)   = b1*(d(:,i+1)-d(:,i))+b2*dp(:,i)+b3
*dpp(:,i);
            dp(:,i+1)    = b4*(d(:,i+1)-d(:,i))+b5*dp(:,i)+b6
*dpp(:,i);

        end

    end
```

Algoritmo del método de Bathe

```
function [d,dp,dpp,Bathet] = BatheMethod(app,caso,metodo)

            % Inputs
            %------
            %
            % app: propiedades (Param y matrices)
            % caso: Selecciona que tipo de excitacion se tiene
    en el problema
            % metodo: tipo de fuerza utilizada
            %
            % Outputs
            %-------
            % d, dp, dpp: Campos de desplazamientos, velocidades
     y aceleracion
            % respectivamente
            % Bathet: Vector de tiempos del metodo de Bathe

            % Parametros del metodo para simplificar ecuaciones

            b0 = 16/(app.Param.dt.^2);
            b1 = 4/app.Param.dt;
            b2 = 9/(app.Param.dt.^2);
            b3 = 3/app.Param.dt;
            b4 = 2*b1;
            b5 = 12/app.Param.dt.^2;
            b6 = -3/app.Param.dt.^2;
            b7 = -1/app.Param.dt;

            % Preparamos los subpasos para este metodo

            Bathedt = app.Param.dt/2;
    % Aqui se define el dt/2
            Bathet  = 0:Bathedt:app.Param.tf+0.2;
    % Vector de tiempos
            nt      = fix((app.Param.tf-app.Param.ti)/Bathedt);
    % Numero de pasos
```

```
        % Condiciones iniciales

        d   = zeros(app.Param.gdl,nt);
        dp  = zeros(app.Param.gdl,nt);
        dpp = zeros(app.Param.gdl,nt);
        F   = zeros(app.Param.gdl,nt+5);

        d(:,1)  = zeros(app.Param.gdl,1); % Desplazamiento
inicial
        dp(:,1) = zeros(app.Param.gdl,1); % Velocidad incial


        % Matriz de Rigidez efectiva [Keff]

        Keff1 = app.Matrices.K+b0*app.Matrices.M+b1*app.
Matrices.C;
        Keff2 = app.Matrices.K+b2*app.Matrices.M+b3*app.
Matrices.C;

        % Se obtiene la aceleracion inicial en funcion de la
 fuerza

        switch caso
            case 'Ejemplo'
                k=20;
                if app.Nodo1Button.Value == 1
                    pos = 1;
                else
                    pos = 2;
                end
                [fe,tf] = fuerzaRandom(app,'Ejemplo');
                fe = k*interp1(tf,fe,Bathet,'spline');
            case 'Ala'
                k=10000;
                pos = app.SpinnerFuerzaNodo.Value;
                [fe,tf] = fuerzaRandom(app,'Ala');
                fe = k*interp1(tf,fe,Bathet,'spline');

            case 'Generico'
```

```
                k=app.AmplitudGenericoEditField.Value;
                pos = app.SpinnerFuerzaNodoGenerico.Value;
                [fe,tf] = fuerzaRandom(app,'Ejemplo');
                fe = k*interp1(tf,fe,Bathet,'spline');
            end

            switch metodo

                case 'Constante'

                    for k = 1:nt+2
                        F(pos,k) = app.AmplitudFuerzaCteGenerico
.Value;
                    end
                    F0 = F(:,1);
                    dpp(:,1) = app.Matrices.M\(F0-app.Matrices.K
*d(:,1)-app.Matrices.C*dp(:,1));

                case 'Sinusoidal'

                    F0   = zeros(app.Param.gdl,1);
                    dpp(:,1) = app.Matrices.M\(F0-app.Matrices.K
*d(:,1)-app.Matrices.C*dp(:,1));
                    F(pos,:)   = app.AmplitudSinusoidalGenerico.
Value*...
                        sin(Bathet*app.
FrecuenciaSinusoidalGenerico.Value);

                case 'Aleatorio'

                    F0       = zeros(app.Param.gdl,1);
                    dpp(:,1) = app.Matrices.M\(F0-app.Matrices.K
*d(:,1)-app.Matrices.C*dp(:,1));
                    F(pos,1:length(fe)) = fe;
            end

            %Se crea el bucle temporal

            Feff = zeros(app.Param.gdl,nt+4);
```

```
        for i = 1:(nt/2)

            % Primer subpaso para t + dt/2

            Feff(:,2*i)    = F(:,2*i)+app.Matrices.M*(b0*d
(:,2*i-1)+b4*dp(:,2*i-1)+dpp(:,2*i-1))+...
                app.Matrices.C*(b1*d(:,2*i-1)+dp(:,2*i-1));
            d(:,2*i)       = Keff1\Feff(:,2*i);
            dp(:,2*i)      = b1*(d(:,2*i)-d(:,2*i-1))-dp(:,2*i
-1);
            dpp(:,2*i)     = b1*(dp(:,2*i)-dp(:,2*i-1))-dpp(:,2
*i-1);

            % Segundo subpaso para t + dt

            Feff(:,2*i+1)    = F(:,2*i+1)+app.Matrices.M*(b5*
d(:,2*i)+b6*d(:,2*i-1)+...
                b1*dp(:,2*i)+b7*dp(:,2*i-1))+app.Matrices.C*
(b1*d(:,2*i)+b7*d(:,2*i-1));
            d(:,2*i+1)       = Keff2\Feff(:,2*i+1);
            dp(:,2*i+1)      = -b7*d(:,2*i-1)-b1*d(:,2*i)+b3*d
(:,2*i+1);
            dpp(:,2*i+1)     = -b7*dp(:,2*i-1)-b1*dp(:,2*i)+b3*
dp(:,2*i+1);

        end

    end
```

Algoritmo del método de Bathe modificado

```
function [d,dp,dpp,Bathet] = BatheMethodModif(app,gamma,beta1,
    beta2,caso,metodo)

        % Inputs
        %------
        %
```

```
        %app: propierdades (Param y matrices)
        %gamma: Porcentaje de division del paso temporal (
dt)
        %beta1 y beta2: Parametros para el segundo subpaso
del esquema
        %caso: Selecciona que tipo de excitacion se tiene
en el problema
        %metodo: Tipo de fuerza utilizada
        %
        %Outputs
        %-------
        %d, dp, dpp: Campos de desplazamientos, velocidades
 y aceleracion
        %respectivamente
        %Bathet: Vector de tiempos del metodo Bathe
Modificado

        %Parametros del metodo Bathe para simplificar
ecuaciones

        b0 = 2/(app.Param.dt*gamma);
        b1 = 4/(app.Param.dt*gamma)^2;
        b2 = 4/(app.Param.dt*gamma);
        %b3 = 1/(beta2*(1-gamma)*app.Param.dt);
        b4 = (gamma*(1-beta1)+beta2*(1-gamma))/((beta2*(1-
gamma))^2*app.Param.dt);
        b5 = (gamma*beta1+(1-beta2)*(1-gamma))/((beta2*(1-
gamma))^2*app.Param.dt);
        b6 = gamma*(1-beta1)/(beta2*(1-gamma));
        b7 = (gamma*beta1+(1-beta2)*(1-gamma))/(beta2*(1-
gamma));
        b8 = 1/(beta2*(1-gamma)*app.Param.dt)^2;
        b9 = 1/(beta2*(1-gamma)*app.Param.dt);

        %Preparamos los subpasos para este metodo

        Bathedt = app.Param.dt*gamma;
        nt      = fix(2*(app.Param.tf-app.Param.ti)/app.
Param.dt);
```

```
        % Se crea el vector de tiempo de modo que en las
posiciones impares este el
        % dt y en las pares el gamma*dt

        Bathet    = zeros(1,nt+2);
        Bathet(1) = 0;
        Bathet(2) = gamma*app.Param.dt;

        for i = 2:nt/2+2

            Bathet(2*i-1) = Bathet(2*i-3)+app.Param.dt;

            if  2*i-1 > nt+1
                break
            end

            Bathet(2*i) = Bathet(2*i-1)+Bathedt;

        end

        % Condiciones iniciales

        d   = zeros(app.Param.gdl,nt);
        dp  = zeros(app.Param.gdl,nt);
        dpp = zeros(app.Param.gdl,nt);
        F   = zeros(app.Param.gdl,nt+3);

        d(:,1)  = zeros(app.Param.gdl,1); % Desplazamiento
inicial
        dp(:,1) = zeros(app.Param.gdl,1); % Velocidad incial

        % Matriz de Rigidez efectiva [Keff]

        Keff1 = app.Matrices.K+b1*app.Matrices.M+b0*app.
Matrices.C;
        Keff2 = app.Matrices.K+b8*app.Matrices.M+b9*app.
Matrices.C;
```

```
            %Se obtiene la aceleracion inicial

            switch caso
                case 'Ejemplo'
                    k=20;
                    if app.Nodo1Button.Value == 1
                        pos = 1;
                    else
                        pos = 2;
                    end
                    [fe,tf] = fuerzaRandom(app,'Ejemplo');
                    fe = k*interp1(tf,fe,Bathet,'spline');
                case 'Ala'
                    k=10000;
                    pos = app.SpinnerFuerzaNodo.Value;
                    [fe,tf] = fuerzaRandom(app,'Ala');
                    fe = k*interp1(tf,fe,Bathet,'spline');

                case 'Generico'
                    k=app.AmplitudGenericoEditField.Value;
                    pos = app.SpinnerFuerzaNodoGenerico.Value;
                    [fe,tf] = fuerzaRandom(app,'Ejemplo');
                    fe = k*interp1(tf,fe,Bathet,'spline');
            end

            switch metodo

                case 'Constante'

                    for k = 1:nt+2
                        F(pos,k) = app.AmplitudFuerzaCteGenerico
.Value;
                    end
                    F0 = F(:,1);
                    dpp(:,1) = app.Matrices.M\(F0-app.Matrices.K
*d(:,1)-app.Matrices.C*dp(:,1));

                case 'Sinusoidal'
```

```
                F0    = zeros(app.Param.gdl,1);
                dpp(:,1) = app.Matrices.M\(F0-app.Matrices.K
*d(:,1)-app.Matrices.C*dp(:,1));
                F(pos,:)    = app.AmplitudSinusoidalGenerico.
Value*...
                    sin(Bathet*app.
FrecuenciaSinusoidalGenerico.Value);

            case 'Aleatorio'

                F0        = zeros(app.Param.gdl,1);
                dpp(:,1) = app.Matrices.M\(F0-app.Matrices.K
*d(:,1)-app.Matrices.C*dp(:,1));
                F(pos,1:length(fe)) = fe;
        end

        %Se crea el bucle temporal

        Feff = zeros(app.Param.gdl,nt+4);

        for j = 1:(nt/2)

            %Primer subpaso para t + dt*gamma

            Feff(:,2*j)    = F(:,2*j)+app.Matrices.M*(b1*d
(:,2*j-1)+b2*dp(:,2*j-1)+dpp(:,2*j-1))+...
                app.Matrices.C*(b0*d(:,2*j-1)+dp(:,2*j-1));
            d(:,2*j)     = Keff1\Feff(:,2*j);
            dp(:,2*j)    = b0*(d(:,2*j)-d(:,2*j-1))-dp(:,2*j
-1);
            dpp(:,2*j)   = b0*(dp(:,2*j)-dp(:,2*j-1))-dpp(:,2
*j-1);

            %Segundo subpaso para t + dt

            Feff(:,2*j+1)    = F(:,2*j+1)+app.Matrices.M*(b8*
d(:,2*j-1)+b4*dp(:,2*j-1)+...
                b5*dp(:,2*j)+b6*dpp(:,2*j-1)+b7*dpp(:,2*j))+
app.Matrices.C*(b9*d(:,2*j-1)+...
```

```
                b6*dp(:,2*j-1)+b7*dp(:,2*j));
            d(:,2*j+1)     = Keff2\Feff(:,2*j+1);
            dp(:,2*j+1)    = b9*(d(:,2*j+1)-d(:,2*j-1))-b6*dp
(:,2*j-1)-b7*dp(:,2*j);
            dpp(:,2*j+1)   = b8*(d(:,2*j+1)-d(:,2*j-1))-b4*dp
(:,2*j-1)-b5*dp(:,2*j)-...
                b6*dpp(:,2*j-1)-b7*dpp(:,2*j);

        end
    end
```

Algoritmo del PIM

```
function [v,vp] = PreciseIntegrationMethod(app,N,caso,metodo)

        %Inputs
        %------
        %
        %app: Propiedades (Param y Matrices)
        %N: Constante del PIM
        %metodo: Tipo de fuerza utilizada
        %caso: Tipo de fuerza aplicada. Armonica o
constante
        %
        %Outputs
        %-------
        %v, vp: Campos de desplazamientos, velocidades y
aceleracion
        %eA_absmax: Es el radio espectral
        %Tperiod: Valores de paso temporales / Periodo
donde se representa el
        %radio espectral

        %El sistema a resolver es el siguiente:
        %
        %vp(t) = A*v+r

        %Se definen los valores necesarios para la
```

```
computacion de la matriz
        % exponencial T

        app.Param.N    = N;                                    %
Recomendacion del metodo
        app.Param.tau = app.Param.dt/2^app.Param.N;
    % Subpaso temporal

        % Se genera la Matriz A [2gdl x 2gdl]

        A = [(-app.Matrices.M\app.Matrices.C)/2 inv(app.
Matrices.M); ...
            -(app.Matrices.K-(app.Matrices.C*inv(app.
Matrices.M)*app.Matrices.C)/4) ...
            -(app.Matrices.C/app.Matrices.M)/2];

        % Se evalua la Matiz Ta(dtau) para inicializarla

        Ta = A*app.Param.tau+...
            (A*app.Param.tau)^2*(eye(app.Param.gdl*2)+(A*app
.Param.tau)/3+(A*app.Param.tau)^2/12)/2;

        % Se ejecuta el bucle que calcula Ta

        for i = 1:app.Param.N

            Ta = 2*Ta + Ta*Ta;

        end

        % Suma de la matriz de identidad tras computar Ta
para evitar errores de
        % precision

        T = eye(app.Param.gdl*2) + Ta;

        % Las condiciones iniciales son

        v  = zeros(2*app.Param.gdl,app.Param.nt);
```

```
vp = zeros(2*app.Param.gdl,app.Param.nt);

v(:,1)  = zeros(2*app.Param.gdl,1);

%Definicion de la fuerza

switch caso
    case 'Ejemplo'
        k=20;
        if app.Nodo1Button.Value == 1
            pos = 1;
        else
            pos = 2;
        end
        [fe,tf] = fuerzaRandom(app,'Ejemplo');
        fe = k*interp1(tf,fe,app.Param.t,'spline');
    case 'Ala'
        k=10000;
        pos = app.SpinnerFuerzaNodo.Value;
        [fe,tf] = fuerzaRandom(app,'Ala');
        fe = k*interp1(tf,fe,app.Param.t,'spline');
        case 'Generico'
        k=app.AmplitudGenericoEditField.Value;
        pos = app.SpinnerFuerzaNodoGenerico.Value;
        [fe,tf] = fuerzaRandom(app,'Ejemplo');
        fe = k*interp1(tf,fe,app.Param.t,'spline');
end

%Resuelve

switch metodo

    case 'Constante'

        r1 = zeros(app.Param.gdl,app.Param.nt);
        r2 = zeros(app.Param.gdl,app.Param.nt);

        for j = 1:app.Param.nt
```

```
                    r2(pos,j) = app.
AmplitudFuerzaCteGenerico.Value;

                end
                r  = [r1;r2];

                vp(:,1) = r(:,1);

            case 'Sinusoidal'

                r1 = zeros(app.Param.gdl,app.Param.nt+1);
                r2 = zeros(app.Param.gdl,app.Param.nt+1);

                r2(pos,:) = app.AmplitudSinusoidalGenerico.
Value*...
                    sin(app.Param.t*app.
FrecuenciaSinusoidalGenerico.Value);

                r  = [r1;r2];

                vp(:,1) = r(:,1);

            case 'Aleatorio'

                F0      = zeros(app.Param.gdl,1);
                r1      = zeros(app.Param.gdl,1);
                r2      = F0;
                r       = [r1;r2];
                vp(:,1) = r;
                r       = zeros(2*app.Param.gdl,app.Param.
nt+1);

                r(app.Param.gdl+pos,:) = fe;

        end

        %Se soluciona, la primera mitad de filas del
resultado representan el
        %desplazamiento o giros en los nodos
```

```
respectivamente

            for j = 1:app.Param.nt

                v(:,j+1) = T*(v(:,j)+A\r(:,j))-A\r(:,j);

            end

            %Para calcular las aceleraciones se hace uso de la
relacion vp = A*v+r

            for j = 1:app.Param.nt

                vp(:,j+1) = A*v(:,j+1)+r(:,j);

            end

        end
```

Algotirmo de Runge Kutta. Definición de espacio de los estados

```
function [x_dot] = dxdt(~,x,u,A,B)

            %Inputs
            %------
            %
            %x: vector de estados [gdl*2 x 1]
            %u: vector de fuerza  [gdl x 1]
            %A: Matriz de ES
            %B: Matriz de ES
            %
            %Outputs
            %-------
            %xdot: Solucion de la ecuacion de espacio de los
estados

            %Modelo espacio de estados
```

```
            x_dot = A*x+B*u;
        end


function [t,x_sol,x_drv] = RK4(app,x0,u,t,A,B)

            %Inputs
            %------
            %
            %app: Propiedades (Parametros y Matrices)
            %x0 : Vector con las condiciones iniciales de
    tama o [2*gdl x 1]
            %t  : Vector con los tiempos
            %u  : Matriz con las fuerzas en todos los tiempos
    de tama o [gdl x 1]
            %A: Matriz de estado
            %B: Matriz de entrada
            %
            %Outputs
            %-------
            %t: Vector de tiempos
            %x_sol: Solucion de la ecuacion diferencial
            %x_drv: La derivada de la solucion

            points = length(t);       %Calcula los puntos del
    intervalo

            %dxdt1 = dxdt(x0,u(:,1)); %Computa la primera
    salida

            n = length(x0(:));

            x_sol = nan(n,points); %Inicializacion
            x_drv = nan(n,points); %Inicializacion

            [~,cu] = size(u);

            %Llena u de ceros si no est  bien dimensionado
```

```
        if (cu < points), u(:,cu+1:points) = 0; end

        for p = 1:points            %Se van recorriendo
pasos temporales

            dxdt1 = dxdt(app,x0,u(:,p),A,B);    %k1
            x_sol(:,p) = x0(:);          %Estado actual
            x_drv(:,p) = dxdt1(:);       %Estado actual


            if p == points, break; end
            time = t(p);
            dt   = t(p+1)-time;          %El paso de tiempo
para este intervalo
            dt2  = dt/2;                 %Mitad del paso
temporal

            dxdt2 = dxdt(app,x0+dxdt1*dt2 , (u(:,p)+u(:,p+1)
)/2.0,A,B); %k2
            dxdt3 = dxdt(app,x0+dxdt2*dt2 , (u(:,p)+u(:,p+1)
)/2.0,A,B); %k3
            dxdt4 = dxdt(app,x0+dxdt3*dt  ,  u(:,p+1)
     ,A,B); %k4

            x0 = x0 + (dxdt1+2*(dxdt2+dxdt3)+dxdt4)*dt/6.0;

        end

    end
```

Algritmo de la solución Exacta

```
function [t,u] = TimeSol_Exacta(app,x0,A,b,fe,Datos,caso,metodo)

        %%FUNCION PARA OBTENER LA SOLUCIoN DEL SISTEMA DE
ECUACIONES DIFERENCIALES
        %
```

```
            %Sistema din mico
            %Forma canonica (espacio-estado): dz/dt = A.z + b*g
(t)
            %Forma est ndar (cl sica):        M.dz/dt = R.z +
 r.g(t)
            %
            %app = Propiedades (Parametros y Matrices)
            %Datos = Vector con los datos de la fuerza
            %x0 = Condiciones iniciales
            %metodo = Tipo de fuerza que se va a aplicar
            %caso = Tipo de problema al que se va a aplicar
            %z = [u; u'] = Vector columna con variables estado
            %z0 = u(t), z1 = u'(t)
            %A = Matriz del sistema
            %b = Vector de terminos independientes
            %fe = Funcion de entrada de excitacion forzada
            %Relaciones entre las matrices
            %A = M^(-1) * R
            %b = M^(-1) * r


            %Condiciones iniciales

            z_inicial = x0;

            %%SOLUCION DEL PROBLEMA

            %dz/dt = A * z + b*g(t)
            %z(0) = z_inicial

            %A: Matriz 2Nx2N (conocido)
            %z: Vector 2Nx1  (incognitga)
            %b: Vector 2Nx1  (conocido)
            %z_ini: Condicion inicial, 2Nx1

            switch caso
                case 'Ejemplo'
                    k=20;
                case 'Ala'
```

```
                k=10000;
            case 'Generico'

                if app.AleatorioButton.Value == 1

                    k = app.AmplitudGenericoEditField.Value;

                elseif app.ArmnicaButton.Value == 1

                    k = app.AmplitudSinusoidalGenerico.Value
;

                else %Fuerza Constante
                    k = app.AmplitudFuerzaCteGenerico.Value;
                end
        end

        %DEFINICION DE FUERZAS
-------------------------------

        switch metodo

            case 'Aleatorio'

                f = @(t,z) A * z  + k...
                    * b * interp1(Datos.tf,Datos.fe,t,'
spline');

            case 'Constante'

                f = @(t,z) A * z  + b * interp1(app.Param.t,
fe,t,'spline');

            case 'Sinusoidal'

                f = @(t,z) A * z  + b * interp1(app.Param.t,
fe,t,'spline');

        end
```

```
            % Solucion de la Ecuacion diferencial.
            [t,u] = ode45(f,[app.Param.ti app.Param.tf],
    z_inicial);

            % t: Vector columna mx1 con el intervalo de tiempos
    [tmin, tmax]. N=n mero
            % de puntos en la solucion
            % u: Vector con la solucion: Matriz de tama o mxN
            % u(:,1) = Solucion de la primera funcion
            % u(:,2) = Solucion de la segunda funcion
            % ..................................
            % xa(:,N) = Solucion de la funcion N

            % Solucion de autovalores (estabilidad)
            %[V,D] = eig(A)

            u = u';

        end
```

Algoritmo del botón 'SOLVE'

```
function SOLVEButton_3Pushed(app, event)
            wb = waitbar(0,'CARGANDO');
            [Msize,~] = size(app.Matrices.M);

            % Numero de Grados de Libertad del problema

            app.Param.gdl = Msize;

            % Bloque de definicion del Paso Temporal y Tiempos

            app.Param.dt = app.tsSlider_3.Value;
        % Paso Temporal
            app.Param.ti = app.TiempoinicialsEditField_3.Value;
        % Tiempo inicial
            app.Param.tf = app.TiempoFinalsEditField_3.Value;
```

```
        %Tiempo final
            app.Param.t  = app.Param.ti:app.Param.dt:app.Param.
tf;    %Vector de tiempos
            app.Param.nt = fix((app.Param.tf-app.Param.ti)/app.
Param.dt);

            nodo    = app.SpinnerRespuestaNodo.Value;

            close(wb)
            wb = waitbar(0.2,'CARGANDO');

            if app.AleatorioButton.Value == 1

                u  = zeros(app.Param.gdl,length(app.Param.t));
%Vector de fuerzas SS

                t = app.Param.t;                                %
Vector de tiempos para RK4

                A = [zeros(app.Param.gdl,app.Param.gdl) eye(app.
Param.gdl);...
                    -app.Matrices.M\app.Matrices.K -app.Matrices
.M\app.Matrices.C];  %Matriz de estado
                B = [zeros(app.Param.gdl,app.Param.gdl); inv(app
.Matrices.M)];   %Matriz de entrada

                p = zeros(app.Param.gdl,1); %Vector de posicion
 de la fuerza

                Datos                   = [];
                nodoFuerza              = app.
SpinnerFuerzaNodoGenerico.Value;


                %Se selecciona la posicion de la fuerza

                p(nodoFuerza) = 1;

                %Se crea el vector donde se resuelve B*u
```

```
            b = [zeros(app.Param.gdl,1) ; inv(app.Matrices.M
)*p];

            [Datos.fe,Datos.tf] = fuerzaRandom(app,'Ejemplo'
);
            fe = app.AmplitudGenericoEditField.Value*interp1
(Datos.tf,Datos.fe,app.Param.t,'spline');
            u(nodoFuerza,:) = fe;

            x0      = zeros(app.Param.gdl*2,1);
% Condiciones iniciales

            % Resultado Exacto

            [app.Resultado.Exacto.t_exacta,app.Resultado.
Exacto.u_exacta] = ...
                TimeSol_Exacta(app,x0,A,b,fe,Datos,'Generico
','Aleatorio');

            % Solucion Newmark

            beta  = app.betavalue_3.Value;
            gamma_n = app.gammavalue_3.Value;

            [app.Resultado.Newmark.d,app.Resultado.Newmark.
dp,...
                app.Resultado.Newmark.dpp] =
NewmarkMethod_ejemplo(app,beta,gamma_n,'Generico','Aleatorio'
);

            % Solucion Bathe

            [app.Resultado.Bathe.d,app.Resultado.Bathe.dp,
...
                app.Resultado.Bathe.dpp,app.Resultado.Bathe.
Bathet] = BatheMethod(app,'Generico','Aleatorio');

            % Solucion Bathe Modificado
```

```
                gamma = app.thetavalue_3.Value;
                beta1 = app.beta1value_3.Value;
                beta2 = app.beta2value_3.Value;

                [app.Resultado.Bathemod.d,app.Resultado.Bathemod
.dp,...
                    app.Resultado.Bathemod.dpp,app.Resultado.
Bathemod.Bathet] =...
                    BatheMethodModif(app,gamma,beta1,beta2,'
Generico','Aleatorio');

                % Solucion PIM

                N = app.NPIMEditField_3.Value;

                [app.Resultado.PIM.v,app.Resultado.PIM.vp] =...
                    PreciseIntegrationMethod(app,N,'Generico','
Aleatorio');

                % Solucion Runge-Kutta

                [app.Resultado.RK.t,app.Resultado.RK.x_sol,...
                    app.Resultado.RK.x_drv] = RK4(app,x0,u,t,A,B
);

            elseif app.ArmnicaButton.Value == 1

                u  = zeros(app.Param.gdl,length(app.Param.t));
% Vector de fuerzas SS

                t = app.Param.t;                                    %
Vector de tiempos para RK4

                A = [zeros(app.Param.gdl,app.Param.gdl) eye(app.
Param.gdl);...
                    -app.Matrices.M\app.Matrices.K -app.Matrices
.M\app.Matrices.C]; % Matriz de estado
                B = [zeros(app.Param.gdl,app.Param.gdl); inv(app
```

```
.Matrices.M)];   %Matriz de entrada

            p = zeros(app.Param.gdl,1); %Vector de posicion
 de la fuerza

            Datos                   = [];
            nodoFuerza              = app.
SpinnerFuerzaNodoGenerico.Value;

            %Se selecciona la posicion de la fuerza

            p(nodoFuerza) = 1;

            %Se crea el vector donde se resuelve B*u

            b = [zeros(app.Param.gdl,1) ; inv(app.Matrices.M
)*p];

            fe = app.AmplitudSinusoidalGenerico.Value*...
                sin(app.Param.t*app.
FrecuenciaSinusoidalGenerico.Value);
            u(nodoFuerza,:) = fe;

            x0 = zeros(app.Param.gdl*2,1);              %
Condiciones iniciales

            %Resultado Exacto

            [app.Resultado.Exacto.t_exacta,app.Resultado.
Exacto.u_exacta] = ...
                TimeSol_Exacta(app,x0,A,b,fe,Datos,'Generico
','Sinusoidal');

            %Solucion Newmark

            beta  = app.betavalue_3.Value;
            gamma_n = app.gammavalue_3.Value;

            [app.Resultado.Newmark.d,app.Resultado.Newmark.
```

```
dp, ...
                    app.Resultado.Newmark.dpp] =
NewmarkMethod_ejemplo(app, beta, gamma_n, 'Generico', 'Sinusoidal
');

                % Solucion Bathe

                [app.Resultado.Bathe.d, app.Resultado.Bathe.dp,
...
                    app.Resultado.Bathe.dpp, app.Resultado.Bathe.
Bathet] = BatheMethod(app, 'Generico', 'Sinusoidal');

                % Solucion Bathe Modificado

                gamma = app.thetavalue_3.Value;
                beta1 = app.beta1value_3.Value;
                beta2 = app.beta2value_3.Value;

                [app.Resultado.Bathemod.d, app.Resultado.Bathemod
.dp, ...
                    app.Resultado.Bathemod.dpp, app.Resultado.
Bathemod.Bathet] =...
                    BatheMethodModif(app, gamma, beta1, beta2, '
Generico', 'Sinusoidal');

                % Solucion PIM

                N = app.NPIMEditField_3.Value;

                [app.Resultado.PIM.v, app.Resultado.PIM.vp] =...
                    PreciseIntegrationMethod(app, N, 'Generico', '
Sinusoidal');

                % Solucion Runge-Kutta

                [app.Resultado.RK.t, app.Resultado.RK.x_sol, ...
                    app.Resultado.RK.x_drv] = RK4(app, x0, u, t, A, B
);
```

```
        else %Fuerza Constante

            u  = zeros(app.Param.gdl,length(app.Param.t));
%Vector de fuerzas SS

            t = app.Param.t;                                    %
Vector de tiempos para RK4

            A = [zeros(app.Param.gdl,app.Param.gdl) eye(app.
Param.gdl);...
                -app.Matrices.M\app.Matrices.K -app.Matrices
.M\app.Matrices.C]; %Matriz de estado
            B = [zeros(app.Param.gdl,app.Param.gdl); inv(app
.Matrices.M)];  %Matriz de entrada

            p = zeros(app.Param.gdl,1); %Vector de posicion
 de la fuerza

            Datos                   = [];
            nodoFuerza              = app.
SpinnerFuerzaNodoGenerico.Value;

            %Se selecciona la posicion de la fuerza

            p(nodoFuerza) = 1;

            %Se crea el vector donde se resuelve B*u

            b = [zeros(app.Param.gdl,1) ; inv(app.Matrices.M
)*p];

            fe = zeros(1,length(app.Param.t));
            for l = 1:length(app.Param.t)

                fe(1,l) = app.AmplitudFuerzaCteGenerico.
Value;

            end
            u(nodoFuerza,:) = fe;
```

```
            x0 = zeros(app.Param.gdl*2,1);                % Condiciones iniciales

            % Resultado Exacto

            [app.Resultado.Exacto.t_exacta,app.Resultado.Exacto.u_exacta] = ...
                TimeSol_Exacta(app,x0,A,b,fe,Datos,'Generico','Constante');

            % Solucion Newmark

            beta  = app.betavalue_3.Value;
            gamma_n = app.gammavalue_3.Value;

            [app.Resultado.Newmark.d,app.Resultado.Newmark.dp,...
                app.Resultado.Newmark.dpp] = NewmarkMethod_ejemplo(app,beta,gamma_n,'Generico','Constante');

            % Solucion Bathe

            [app.Resultado.Bathe.d,app.Resultado.Bathe.dp,...
                app.Resultado.Bathe.dpp,app.Resultado.Bathe.Bathet] = BatheMethod(app,'Generico','Constante');

            % Solucion Bathe Modificado

            gamma = app.thetavalue_3.Value;
            beta1 = app.beta1value_3.Value;
            beta2 = app.beta2value_3.Value;

            [app.Resultado.Bathemod.d,app.Resultado.Bathemod.dp,...
                app.Resultado.Bathemod.dpp,app.Resultado.Bathemod.Bathet] =...
```

```
                BatheMethodModif(app,gamma,beta1,beta2,'
Generico','Constante');

            %Solucion PIM

            N = app.NPIMEditField_3.Value;

            [app.Resultado.PIM.v,app.Resultado.PIM.vp] =...
                PreciseIntegrationMethod(app,N,'Generico','
Constante');

            %Solucion Runge-Kutta

            [app.Resultado.RK.t,app.Resultado.RK.x_sol,...
                app.Resultado.RK.x_drv] = RK4(app,x0,u,t,A,B
);

        end

            close(wb)
            wb = waitbar(0.7,'CARGANDO');
            valueNewmark   = app.NewmarkButton_3.Value;
            valueBathe     = app.BatheButton_3.Value;
            valueBatheMod  = app.BatheModificadoButton_3.
Value;
            valuePIM       = app.PIMButton_3.Value;
            valueRK        = app.RungeKuttaButton_3.Value;
            valueExacto    = app.ExactaButton_3.Value;

            if valueNewmark == 1

                app.UIAxes_3.Title.String = 'Desplazamiento
Nodal Newmark';
                plot(app.UIAxes_3,app.Param.t,...
                    app.Resultado.Newmark.d(nodo,:));

            elseif valueBathe == 1

                app.UIAxes_3.Title.String = 'Desplazamiento
```

```
Nodal Bathe';
                plot(app.UIAxes_3,app.Resultado.Bathe.Bathet
...
                    (1:length(app.Resultado.Bathe.d)),...
                    app.Resultado.Bathe.d(nodo,:));

            elseif valueBatheMod == 1

                app.UIAxes_3.Title.String = 'Desplazamiento
Nodal Bathe Modificado';
                plot(app.UIAxes_3,app.Resultado.Bathemod.
Bathet...
                    (1:length(app.Resultado.Bathemod.d)),...
                    app.Resultado.Bathemod.d(nodo,:));

            elseif valuePIM == 1

                app.UIAxes_3.Title.String = 'Desplazamiento
Nodal PIM';
                plot(app.UIAxes_3,app.Param.t,...
                    app.Resultado.PIM.v(nodo,:));

            elseif valueRK == 1

                app.UIAxes_3.Title.String = 'Desplazamiento
Nodal Runge-Kutta';
                plot(app.UIAxes_3,app.Resultado.RK.t,...
                    app.Resultado.RK.x_sol(nodo,:));

            elseif valueExacto == 1

                app.UIAxes_3.Title.String = 'Desplazamiento
Nodal Exacto';
                plot(app.UIAxes_3,app.Resultado.Exacto.
t_exacta,...
                    app.Resultado.Exacto.u_exacta(nodo,:));

            else
```

```
            end

            % Se programa la gr fica que contiene los
errores


            app.UIAxes2_3.YLim = ([-app.AmplitudSlider_3.
Value app.AmplitudSlider_3.Value]);

            valueNewmarkerror  = app.NewmarkButtonerror_3.
Value;
            valueBatheerror    = app.BatheButtonerror_3.
Value;
            valueBatheModerror = app.BatheModButtonerror_2.
Value;
            valuePIMerror      = app.PIMButtonerror_3.Value;
            valueRKerror       = app.RKButtonerror_3.Value;

            % Error Newmark

            Nd_interp = interp1(app.Param.t,app.Resultado.
Newmark.d(nodo,:),...
                app.Resultado.Exacto.t_exacta,'lineal');
            app.error.errorN = abs((app.Resultado.Exacto.
u_exacta(nodo,:)...
                -Nd_interp'))./app.Resultado.Exacto.u_exacta
(nodo,:)*100;

            % Error Bathe

            Bd_interp = interp1(app.Resultado.Bathe.Bathet
...
                (1:length(app.Resultado.Bathe.d)),...
                app.Resultado.Bathe.d(nodo,:),...
                app.Resultado.Exacto.t_exacta,'lineal');
            app.error.errorB = abs((app.Resultado.Exacto.
u_exacta(nodo,:)...
            -Bd_interp'))./app.Resultado.Exacto.u_exacta(
nodo,:)*100;
```

```
            % Error Bathe Modificado

            Bmd_interp = interp1(app.Resultado.Bathemod.Bathet
...
                (1:length(app.Resultado.Bathe.d)),...
                app.Resultado.Bathemod.d(nodo,:),...
                app.Resultado.Exacto.t_exacta,'lineal');
            app.error.errorBm = abs((app.Resultado.Exacto.
u_exacta(nodo,:)...
                -Bmd_interp'))./app.Resultado.Exacto.u_exacta(
nodo,:)*100;

            % Error PIM

            PIM_interp = interp1(app.Param.t,app.Resultado.PIM.v
(nodo,:),...
                app.Resultado.Exacto.t_exacta,'lineal');
            app.error.errorPIM = abs((app.Resultado.Exacto.
u_exacta(nodo,:)...
                -PIM_interp'))./app.Resultado.Exacto.u_exacta(
nodo,:)*100;

            % Error Runge-Kutta

            rk_interp = interp1(app.Param.t,...
                app.Resultado.RK.x_sol(nodo,:),...
                app.Resultado.Exacto.t_exacta,'lineal');
            app.error.errorRK = abs((app.Resultado.Exacto.
u_exacta(nodo,:)...
                -rk_interp'))./app.Resultado.Exacto.u_exacta(
nodo,:)*100;


            if valueNewmarkerror == 1

                app.UIAxes2_3.Title.String = 'Error Newmark';
                plot(app.UIAxes2_3,app.Resultado.Exacto.t_exacta
,...
```

```
                app.error.errorN,'-*');

        elseif valueBatheerror == 1

            app.UIAxes2_3.Title.String = 'Error Bathe';
            plot(app.UIAxes2_3,app.Resultado.Exacto.t_exacta
,...
                app.error.errorB,'-*');

        elseif valueBatheModerror == 1

            app.UIAxes2_3.Title.String = 'Error Bathe
Modificado';
            plot(app.UIAxes2_3,app.Resultado.Exacto.t_exacta
,...
                app.error.errorBm,'-*');

        elseif valuePIMerror == 1

            app.UIAxes2_3.Title.String = 'Error PIM';
            plot(app.UIAxes2_3,app.Resultado.Exacto.t_exacta
,...
                app.error.errorPIM,'-*');

        elseif valueRKerror == 1

            app.UIAxes2_3.Title.String = 'Error Runge-Kutta'
;
            plot(app.UIAxes2_3,app.Resultado.Exacto.t_exacta
,...
                app.error.errorRK,'-*');

        else

        end
        close(wb)
        wb = waitbar(1,'COMPLETADO');
    end
```