5.4 Transiente Analyse

Lösungen

Aufgabe 1

a) Frequenzgehalt der Anregung

Der Frequenzgehalt der Anregung wird durch eine Fourier-Transformation bestimmt. Die Anregung wird mit der Funktion **pulse.m** berechnet, die später auch für die transiente Analyse verwendet wird.

```
function y = pulse(t, t0)

# Übungsblatt 5.4: Kraftstoss
#
# ------------------------------------------------------------

  nt = length(t);
  y  = zeros(1, nt);

  ix = find(t <= t0);
  ts = t(ix) / t0;

  y(ix) = sin(pi * ts).^2;

end
```

Das folgende GNU Octave-Skript berechnet die Fourier-Transformation, stellt sie graphisch dar und vergleicht die ursprüngliche Zeitreihe mit der durch Rücktransformation der bei $f_C = 3/t_0 = 120\,\mathrm{Hz}$ abgeschnittenen Fourier-Transformation. Diese Zeitreihe wird mithilfe der Funktion **resample** berechnet, indem die Abtastrate entsprechend verkleinert wird. Zur besseren Darstellung wird der Zeitschritt anschließend wieder um den Faktor 20 verkleinert.

```
# Übungsblatt 5.4, Aufgabe 1: Elliptische Platte
#                             Untersuchung der Belastung
#
# ------------------------------------------------------------

  pkg load signal       % Enthält Funktion resample
  file = mfilename();

# Daten

  t0   = 0.025;         % Impulsdauer
  fc   =     3;         % Abschneidefrequenz multipliziert mit t0
```

```
  tmax = 5 * t0;        % Länge der Zeitreihe
  dt   = 0.02 * t0;     % Zeitschritt
  t    = 0 : dt : tmax;
  nt   = length(t);
  fs   = t0 / dt;       % Abtastrate multipliziert mit t0

# Funktionswerte

  y = pulse(t, t0);

# Fourier-Transformation divididert durch t0

  Y   = dt * fft(y) / t0;
  f   = (0 : nt - 1) * fs / nt;   % f * t0
  ixp = floor(0.5 * nt);

# Abschneiden
```

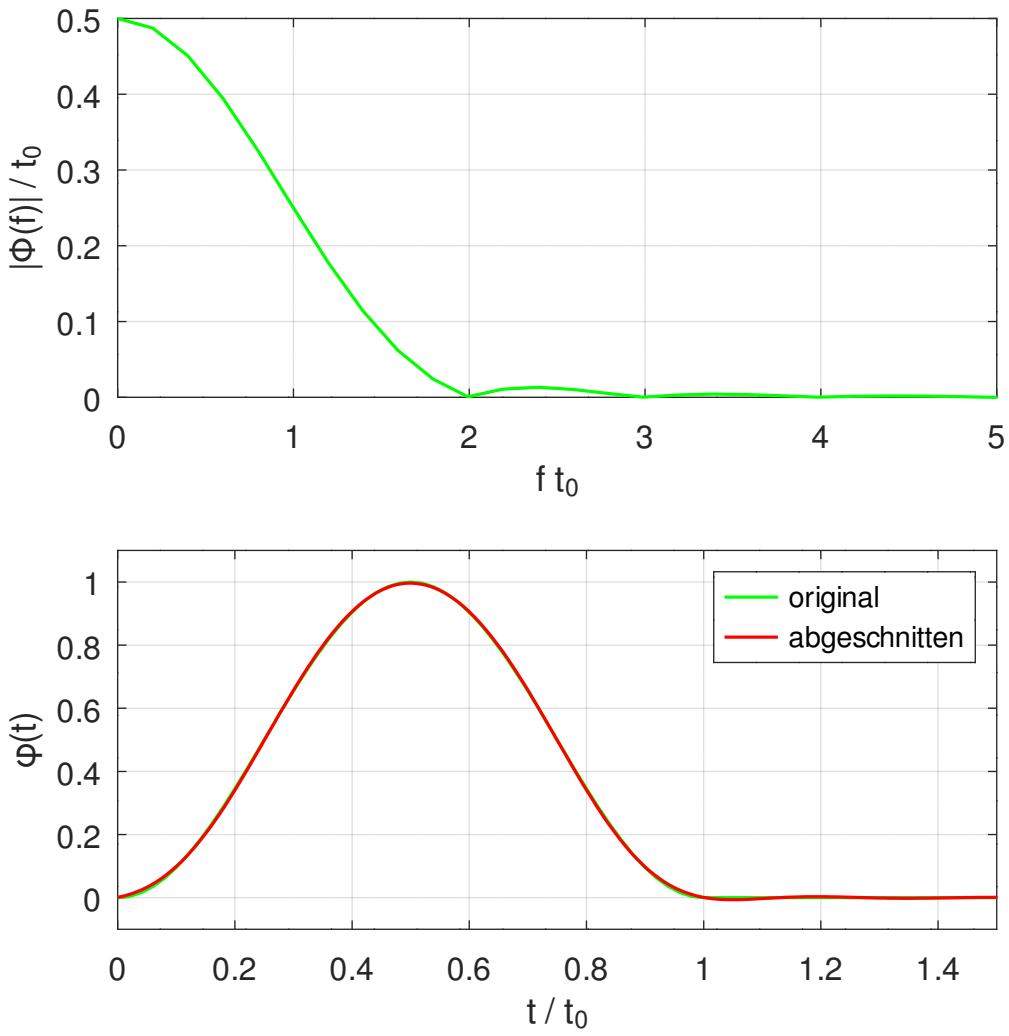


Abbildung 1.1: Anregung und Fourier-Transformation

```
  fa  = round(0.5 * fs);      % Nyquist-Frequenz mult. mit t0
  yc  = resample(y, fc, fa);  % Downsampling
  dtc = dt * fa / fc;
  yr  = resample(yc, 20, 1);  % Upsampling
  tr  = (0 : length(yr) - 1) * dtc / 20;

# Ausgabe

  figure(1, "position", [100, 100, 700, 700],
            "paperposition", [0, 0, 15, 15]);
  subplot(2, 1, 1);
     plot(f(1:ixp), abs(Y(1:ixp)), "color", "green");
     xlim([0, 5]);
     grid;
     ylim([0, 0.5]);
     xlabel('f t_0');
     ylabel('|\Phi(f)| / t_0');
  subplot(2, 1, 2);
     plot(t/t0, y, "color", "green",
          tr/t0, yr, "color", "red");
     legend('original', 'abgeschnitten');
     grid;
     xlim([0, 1.5]);
     ylim([-0.1, 1.1]);
     xlabel('t / t_0');
     ylabel('\phi(t)');
  print([file, ".svg"], "-dsvg");
```

Abbildung 1.1 zeigt, dass der Betrag der Fourier-Transformierten ab einer Frequenz von $2/t_0$ sehr klein und oberhalb von einer Frequenz von $3/t_0$ praktisch null ist. Die zur bei $3/t_0$ abgeschnittenen Fourier-Transformierten gehörende Zeitreihe weicht nur wenig von der gegebenen Zeitreihe ab.

b) Modalanalyse

Das folgende Gmsh-Skript definiert die Geometrie und die Netzfeinheit. Die Ellipse wird aus vier Ellipsensegmenten zusammengesetzt.

```
/* ------------------------------------------------------------
   Übungsblatt 5.4, Aufgabe 1: Elliptische Platte

   Physical Groups:  Platte       Surface   Elemente
                     Einspannung  Curve     Lagerung
                     Last         Point     Last
                     A            Point     Auswertepunkt
                     B            Point     Auswertepunkt
                     C            Point     Auswertepunkt
                     D            Point     Auswertepunkt
                     E            Point     Auswertepunkt
------------------------------------------------------------ */

// Abmessungen
```

```
   DefineConstant [a = 500,    // große Halbachse
                   b = 375];   // kleine Halbachse

// Netzfeinheit

nofel = GetValue("Anzahl Elemente entlang großer Halbachse = ?",
                 25);
elen  = a / nofel;

// Ellipsenpunkte

Point(1) = { 0,  0, 0, elen};  // Mittelpunkt
Point(2) = {-a,  0, 0, elen};
Point(3) = { 0,  b, 0, elen};
Point(4) = { a,  0, 0, elen};
Point(5) = { 0, -b, 0, elen};

// Brennpunkte

f = Sqrt(a^2 - b^2);

Point(6) = {-f, 0, 0, elen};
Point(7) = { f, 0, 0, elen};

// Weitere Punkte für Auswertung

Point(8) = {-0.5 * f, 0, 0, elen};
Point(9) = { 0.5 * f, 0, 0, elen};

// Ellipsenbögen

Ellipse(1) = {2, 1, 3};
Ellipse(2) = {3, 1, 4};
Ellipse(3) = {4, 1, 5};
Ellipse(4) = {5, 1, 2};

// Fläche

Curve Loop(1) = {1, 2, 3, 4};
Plane Surface(1) = {1};

// Elemente

Physical Surface("Platte") = {1};

// Einspannung

Physical Curve("Einspannung") = {1 : 4};

// Last (linker Brennpunkt)

Physical Point("Last") = {6};

// Auswertepunkte
```

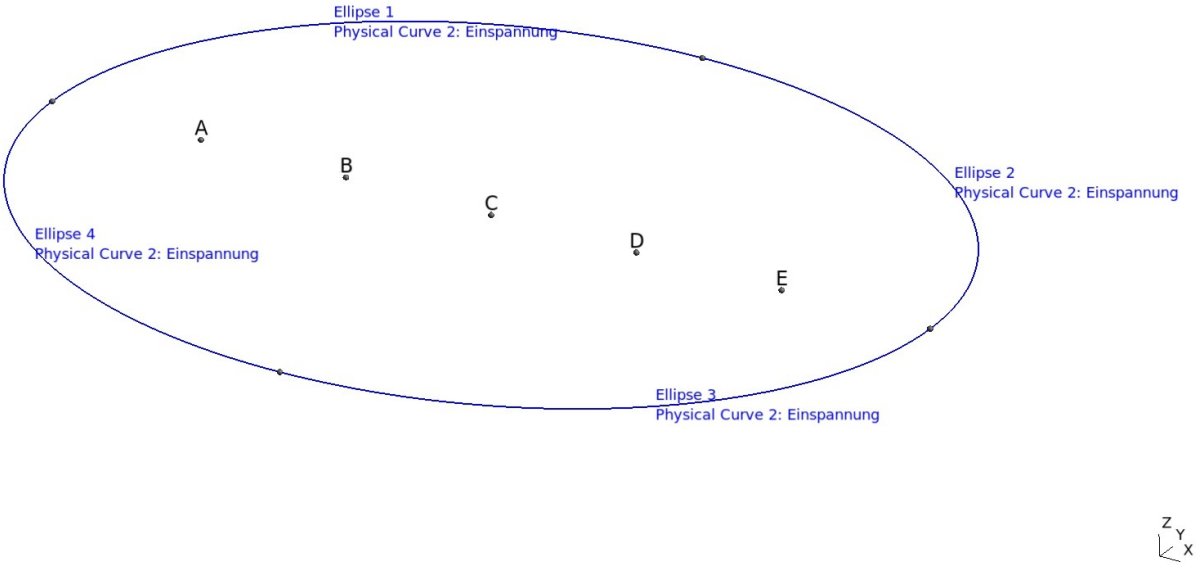


Abbildung 1.2: Geometrie der Platte und Auswertepunkte

```
Physical Point("A") = {6};
Physical Point("B") = {8};
Physical Point("C") = {1};
Physical Point("D") = {9};
Physical Point("E") = {7};

// Vernetzung

Point{1, 6 : 9} In Surface {1};

Mesh.RecombineAll = 1;

// Darstellung der Auswertepunkte

fontsize = 24;
fonttype =  4;
textpos  =  1;
font = fontsize + 2^8 * fonttype + 2^16 * textpos;

coorA = Point{6};
coorB = Point{8};
coorC = Point{1};
coorD = Point{9};
coorE = Point{7};

View "Auswertepunkte" {
   T3(coorA[0], coorA[1], coorA[2], font){ "A" };
   T3(coorB[0], coorB[1], coorB[2], font){ "B" };
   T3(coorC[0], coorC[1], coorC[2], font){ "C" };
```

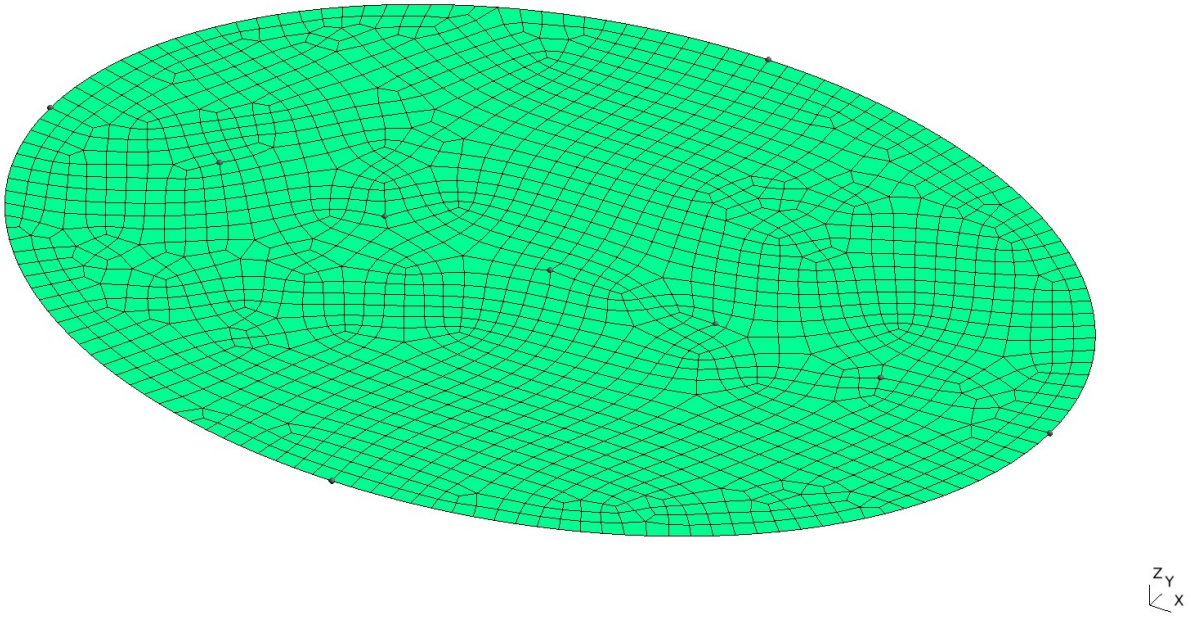


Abbildung 1.3: Vernetzung der Platte

```
   T3(coorD[0], coorD[1], coorD[2], font){ "D" };
   T3(coorE[0], coorD[1], coorE[2], font){ "E" };
};
```

Die Geometrie mit den Auswertepunkten ist in Abbildung 1.2 dargestellt, während Abbildung 1.3 die Vernetzung zeigt. Für die Vernetzung wurden 25 Elemente entlang der großen Halbachse gewählt.

Das folgende GNU Octave-Skript definiert das Berechnungsmodell, berechnet die ersten 20 Eigenschwingungen und schätzt den Fehler in der Formänderungsenergie ab. Die Komponente sowie die Knotenpunkt-Sets werden für die anschließende transiente Analyse in einer Binärdatei gespeichert,

```
# Übungsblatt 5.4, Aufgabe 1: Elliptische Platte
#                             Modell und Modalanalyse
#
# ------------------------------------------------------------

  file = mfilename();
  filb = file(1 : end-1);

  fid = fopen([file, ".res"], "wt");

# Daten (N, mm, s)

  mat = struct("type", "iso",
               "E", 210000, "ny", 0.3, "rho", 7.85E-9);
```

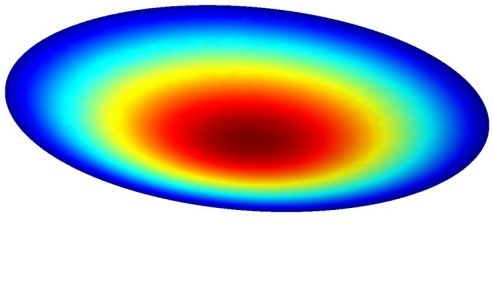
```
  geom = struct("t", 2);

  nofmod  =  20;
  fmax    = 120;
  F       =  10;
  damping = struct("type", "Rayleigh", "data", [2e-5, 4]);

# Übersetzungsdaten

  PLATE = struct("type", "solid", "subtype", "3d");

  PLATE.Platte = struct("type", "elements", "name", "s4",
                        "geom", geom, "mat", mat);

  PLATE.Last   = struct("type", "loads", "name", "point",
                        "data", [0, 0, -F]);

  PLATE.Einspannung = struct("type", "constraints",
                             "name", "prescribed",
                             "dofs", [1 : 3, 6]);

  PLATE.A = struct("type", "nodeset");
  PLATE.B = struct("type", "nodeset");
  PLATE.C = struct("type", "nodeset");
  PLATE.D = struct("type", "nodeset");
  PLATE.E = struct("type", "nodeset");

  PLATE.damping = damping;

# Analyse

  [model, nset] = mfs_import(fid, [filb, ".msh"], "msh", PLATE);

  plate = mfs_new(fid, model);

  plate = mfs_stiff(plate);
  plate = mfs_mass(plate);
  mfs_massproperties(fid, plate);

  plate = mfs_freevib(plate, nofmod);
  mfs_print(fid, plate, "modes", "freq");
  mfs_export("modes.dsp", "msh", plate, "modes", "disp");

  mfs_reductionerror(fid, plate, fmax);

  save("-binary", [filb, ".bin"], "plate", "nset");

  fclose(fid);
```

Die folgende Liste zeigt die Formänderungsenergien und die Fehlerabschätzung:

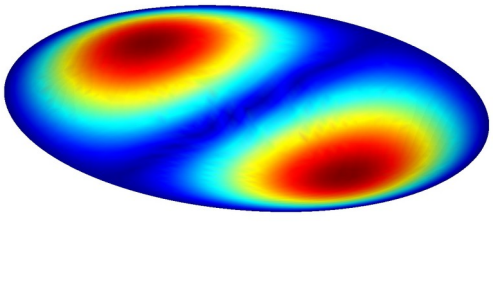
Modal strain energies of component "plate"

Loadcase 1:

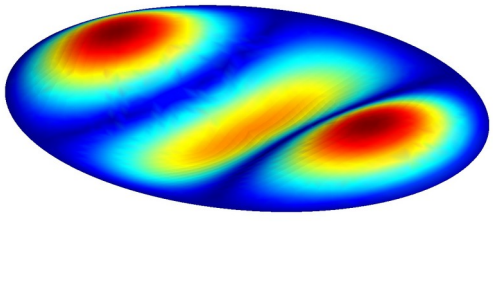
mode	frequency	En/ES	Sum	1 - Sum
1	13.90 Hz	4.36040e-01	0.436040	5.63960e-01
2	32.52 Hz	2.85686e-01	0.721726	2.78274e-01
3	44.55 Hz	1.05681e-10	0.721726	2.78274e-01
4	59.80 Hz	1.43686e-01	0.865411	1.34589e-01
5	70.72 Hz	6.33377e-10	0.865411	1.34589e-01
6	93.20 Hz	1.61173e-04	0.865573	1.34427e-01
7	95.46 Hz	6.05647e-02	0.926137	7.38626e-02
8	104.34 Hz	4.35758e-07	0.926138	7.38622e-02
9	129.13 Hz	6.35671e-04	0.926774	7.32265e-02
10	138.97 Hz	2.09140e-02	0.947687	5.23125e-02
11	145.30 Hz	1.36136e-08	0.947688	5.23125e-02
12	159.26 Hz	8.02222e-09	0.947688	5.23125e-02
13	173.49 Hz	1.77677e-03	0.949464	5.05357e-02
14	189.58 Hz	5.93559e-03	0.955400	4.46001e-02
15	193.42 Hz	4.64881e-07	0.955400	4.45997e-02
16	204.29 Hz	5.50380e-07	0.955401	4.45991e-02
17	227.20 Hz	3.85151e-03	0.959252	4.07476e-02
18	243.21 Hz	5.33733e-07	0.959253	4.07471e-02
19	246.24 Hz	1.42851e-03	0.960681	3.93186e-02
20	248.58 Hz	1.31444e-06	0.960683	3.93172e-02



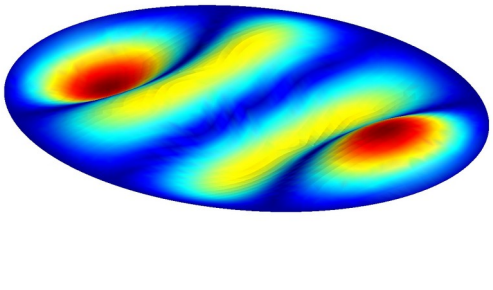
1. Eigenschwingung: 13,90 Hz



2. Eigenschwingung: 32,52 Hz



4. Eigenschwingung: 59,80 Hz



7. Eigenschwingung: 95,46 Hz

Abbildung 1.4: Einige Eigenschwingungen

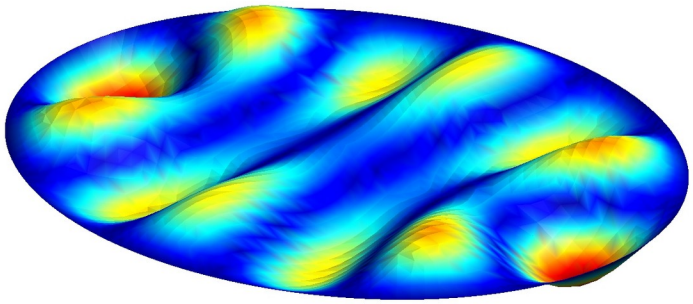


Abbildung 1.5: 19. Eigenschwingung: 246,2 Hz

```
Upper bound on relative strain energy error (fmax =  120.00 Hz)
   with static correction:    1.3761e-02
   without static correction: 3.3420e-02
```

Der Fehler bei Rechnung mit Restmodekorrektur ist kleiner als 1,4 %. Da die Abschneidefrequenz von 120 Hz bereits großzügig gewählt ist, ist diese Genauigkeit ausreichend.

Die wesentlichen Eigenschwingungen im Frequenzbereich bis 120 Hz sind die Eigenschwingungen 1, 2, 4 und 7. Sie sind in Abbildung 1.4 dargestellt. Die höchste berechnete Eigenschwingung, die einen Beitrag zur Antwort liefert, ist die 19. Eigenschwingung. Abbildung 1.5 zeigt, dass die Vernetzung fein genug ist.

c) Transiente Analyse

Mit einer niedrigsten Frequenz von $f_1 = 13{,}90\ \text{Hz}$ und einer höchsten berechneten Frequenz von $f_{max} = 248{,}6\ \text{Hz}$ ergibt sich für die längste Periode ein Wert von ca. $0{,}07\ \text{s}$ und für die kürzeste Periode von ca. $4\ \text{ms}$. Basierend auf diesen Werten wird ein Zeitschritt von $0{,}5\ \text{ms}$ und eine Simulationsdauer von $0{,}4\ \text{s}$ gewählt. Damit ergibt sich folgendes GNU Octave-Skript für die Durchführung der transienten Analyse:

```
# Kapitel 5.4, Aufgabe 1: Elliptische Platte
#                         Modale transiente Analyse
#
# ------------------------------------------------------------

  file = mfilename();
  filb = file(1 : end-1);
  fid = fopen([file, ".res"], "wt");
```

```
  colors = [1, 0, 0; 0, 1, 0; 0, 0, 1; 1, 0, 1; 0, 1, 1];
  set(0, "defaultaxescolororder", colors);
  set(0, "defaultaxesfontsize", 12);

# Daten

  dt =  5e-4;  % Zeitschritt
  t0 = 0.025;  % Impulsdauer
  ts =   0.4;  % Simulationsdauer

# Komponenten einlesen

  load([filb, ".bin"]);

# Last

  exci = struct("lc", 1, "func", "pulse", "params", t0);

# Transiente Analyse

  plate = mfs_transresp(plate, [dt, ts], "load", exci);

# Auswertung

  rid = [nset.A, 3; nset.B, 3; nset.C, 3; nset.D, 3; nset.E, 3];

  t = mfs_getresp(plate, "transresp", "time");
  u = mfs_getresp(plate, "transresp", "disp", rid);
  v = mfs_getresp(plate, "transresp", "velo", rid) * 1e-3;
  a = mfs_getresp(plate, "transresp", "acce", rid) * 1e-3;

  figure(1, "position", [50, 500, 750, 750],
            "paperposition", [0, 0, 16, 18]);
  subplot(3, 1, 1);
     plot(t, u);
     legend("A", "B", "C", "D", "E", "location", "bestoutside");
     set(gca(), "xtick", [0, 0.025, 0.1 : 0.1 : ts]);
     grid;
     axis("labely");
     ylabel('u_z [mm]');
  subplot(3, 1, 2);
     plot(t, v);
     legend("A", "B", "C", "D", "E", "location", "bestoutside");
     set(gca(), "xtick", [0, 0.025, 0.1 : 0.1 : ts]);
     grid;
     axis("labely");
     ylabel('v_z [m/s]');
  subplot(3, 1, 3);
     plot(t, a);
     legend("A", "B", "C", "D", "E", "location", "bestoutside");
     set(gca(), "xtick", [0, 0.025, 0.1 : 0.1 : ts]);
     grid;
     xlabel('t [s]');
     ylabel('a_z [m/s^2]');
  print([file, ".svg"], "-dsvg");
```

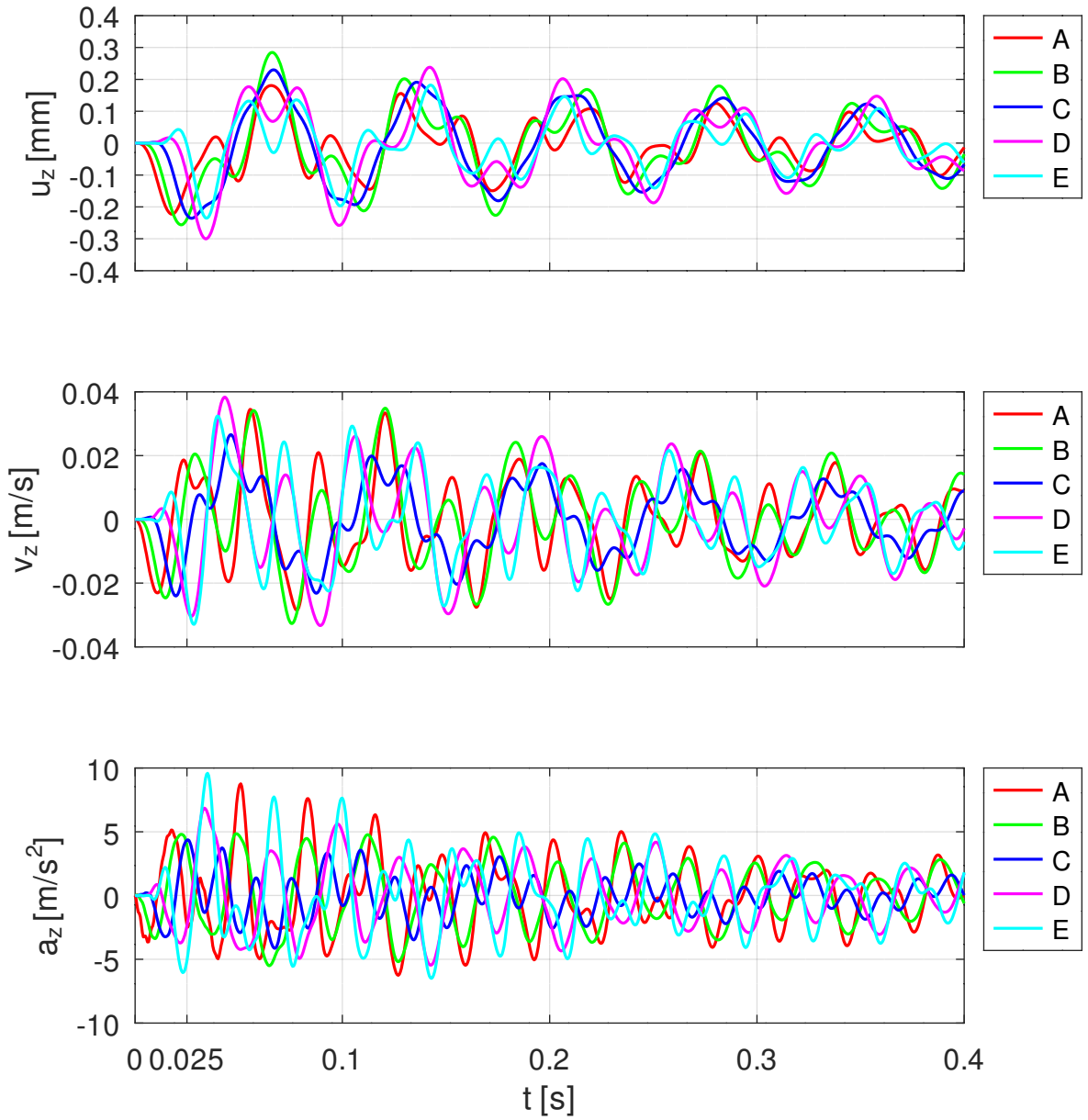


Abbildung 1.6: Ergebnisse

```
fclose(fid);
```

Die Ergebnisse sind in Abbildung 1.6 dargestellt.