5.1 Der eingespannte Flügel

Lösungen

Aufgabe 1

a) Berechnungsmodell der Struktur

Abbildung 1.1 zeigt das Berechnungsmodell der Struktur. Es besteht aus Balken für den vorderen Holm, den Hauptholm, den hinteren Holm sowie die Rippen.

Das folgende GNU Octave-Skript erstellt das Berechnungsmodell und berechnet die ersten fünf Eigenschwingungen. Die Komponente und die Definition der Knotenpunktsets werden in der Datei `solid.bin` gespeichert. Die Knotenpunktsets werden später für die Definition der Splines benötigt.

```
# Übungsblatt 5.1, Aufgabe 1: Knickflügel
#
# a) Strukturmodell
#    Erzeugte Dateien:
#      solid.bin  enthält Strukturmodell mit Eigenschwingungen
#                 und Knotenpunkt-Sets für die Splines
```

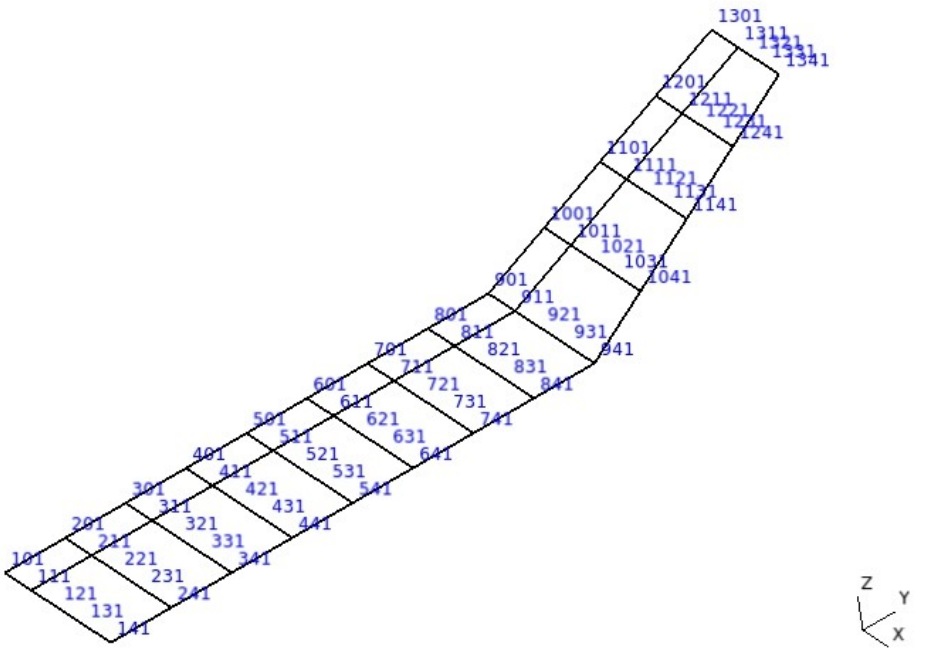


Abbildung 1.1: Balkenmodell der Struktur

```
#
# ------------------------------------------------------------

  fid = fopen("solid.res", "wt");

# Strukturmodell
# --------------

  x1 =  100;  % Vorderer Holm
  x2 =  375;  % Hauptholm
  x3 = 1200;  % Hinterer Holm
  x4 =  800;  % Hinterer Holm an der Flügelspitze

  ya = 4000;  % y-Position des Knicks
  yt = 6000;  % y-Position der Flügelspitze
  zt = 1000;  % z-Position der Flügelspitze

# Modelltyp

  solid.type    = "solid";
  solid.subtype = "3d";

# Material

  mat.type =  "iso";
  mat.E    =  70000;
  mat.ny   =   0.34;
  mat.rho  = 2.7E-9;

# Vorderer Holm

  geom   = mfs_beamsection("I", 30, 100, 2, 2);
  geom.v = [0, 0, 1];

  nodes(1).id =  101; nodes(1).coor = [ 100,    0,    0];
  nodes(2).id =  901; nodes(2).coor = [ 100, 4000,    0];
  nodes(3).id = 1301; nodes(3).coor = [ 100, 6000, 1000];

  idnew = 201 : 100 : 801;
  idelt = 1 : 8;

  [nodes, elem1] = mfs_line(nodes, 101, 901, idnew, idelt,
                            "b2", geom, mat);

  idnew = 1001 : 100 : 1201;
  idelt = 9 : 12;

  [nodes,elem2] = mfs_line(nodes, 901, 1301, idnew, idelt,
                           "b2", geom, mat);

# Hauptholm

  geom   = mfs_beamsection("box", "thin", 150, 180, 2);
  geom.v = [0, 0, 1];
```

```
  n = length(nodes);

  nodes(++n).id =  111; nodes(n).coor = [ 375,    0,    0];
  nodes(++n).id =  911; nodes(n).coor = [ 375, 4000,    0];
  nodes(++n).id = 1311; nodes(n).coor = [ 375, 6000, 1000];

  idnew = 211 : 100 : 811;
  idelt = 13 : 20;

  [nodes, elem3] = mfs_line(nodes, 111, 911, idnew, idelt,
                            "b2", geom, mat);

  idnew = 1011 : 100 : 1211;
  idelt = 21 : 24;

  [nodes,elem4] = mfs_line(nodes, 911, 1311, idnew, idelt,
                           "b2", geom, mat);

# Hinterer Holm

  geom   = mfs_beamsection("I", 30, 100, 2, 2);
  geom.v = [0, 0, 1];

  n = length(nodes);

  nodes(++n).id =  141; nodes(n).coor = [ 1200,    0,    0];
  nodes(++n).id =  941; nodes(n).coor = [ 1200, 4000,    0];
  nodes(++n).id = 1341; nodes(n).coor = [  800, 6000, 1000];

  idnew = 241 : 100 : 841;
  idelt = 25 : 32;

  [nodes, elem5] = mfs_line(nodes, 141, 941, idnew, idelt,
                            "b2", geom, mat);

  idnew = 1041 : 100 : 1241;
  idelt = 33 : 36;

  [nodes,elem6] = mfs_line(nodes, 941, 1341, idnew, idelt,
                           "b2", geom, mat);


  elemh = [elem1, elem2, elem3, elem4, elem5, elem6];

# Vordere Rippen

  geomi   = mfs_beamsection("I", 40, 120, 1, 1);
  geomi.v = [0, 0, 1];

  geoma   = mfs_beamsection("I", 40, 120, 1, 1);
  geoma.v = [0, -1 , 2];

  ide = length(elemh) + 1;
  id1 = 101; id2 = 111;
```

```
  for n = 1 : 9
      elemr(n).id = ide++;  elemr(n).nodes = [id1, id2];
      elemr(n).type = "b2"; elemr(n).geom = geomi;
      elemr(n).mat  = mat;
      id1 += 100; id2 += 100;
  end

  for n = 10 : 13
      elemr(n).id = ide++;  elemr(n).nodes = [id1, id2];
      elemr(n).type = "b2"; elemr(n).geom = geoma;
      elemr(n).mat  = mat;
      id1 += 100; id2 += 100;
  end

# Hintere Rippen

  id1 = 111; id2 = 141; idnew = [121, 131];

  for n = 1 : 9
      idelt = ide : ide + 2;
      [nodes, elemx] = mfs_line(nodes, id1, id2, idnew, idelt,
                                "b2", geomi, mat);
      id1 += 100; id2 += 100; idnew += 100;
      ide += 3;
      elemr = [elemr, elemx];
  end

  for n = 1 : 4
      idelt = ide : ide + 2;
      [nodes, elemx] = mfs_line(nodes, id1, id2, idnew, idelt,
                                "b2", geoma, mat);
      id1 += 100; id2 += 100; idnew += 100;
      ide += 3;
      elemr = [elemr, elemx];
  end

  solid.nodes     = nodes;
  solid.elements  = [elemh, elemr];

# Knotenpunkt-Sets (für Splines)

  nset.inner_wing = [ 101 : 100 : 901, 111 : 100 : 911, ...
                      121 : 100 : 911, 131 : 100 : 931, ...
                      141 : 100 : 941 ];
  nset.outer_wing = [ 901 : 100 : 1301, 911 : 100 : 1311, ...
                      921 : 100 : 1321, 931 : 100 : 1331, ...
                      941 : 100 : 1341 ];

# Einspannung

  prescribed(1).id = 101; prescribed(1).dofs = 1 : 3;
  prescribed(2).id = 111; prescribed(2).dofs = 1 : 6;
  prescribed(3).id = 141; prescribed(3).dofs = 1 : 3;

  solid.constraints.prescribed = prescribed;
```

```
# Analyse
# -------

  wings = mfs_new(fid, solid);
  mfs_export("solid.msh", "msh", wings, "mesh", "mesh", "axes");

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

  wings = mfs_freevib(wings, 5);
  mfs_print(fid, wings, "modes", "freq");
  mfs_export("modes.dsp", "msh", wings, "modes", "disp");

  save -binary solid.bin wings nset

  fclose(fid);
```

Die ersten vier Eigenschwingungen sind in Abbildung 1.2 dargestellt. Es handelt sich um die erste vertikale und die erste horizontale Biegeschwingung, die erste Torsionsschwingung und die zweite vertikale Biegeschwingung.

Die Ausgabedatei enthält folgende Ergebnisse:

```
Mefisto 2.4: Building new component from input "solid"

  Model Type = solid, Model Subtype = 3d

  Number of nodes      =    65,  Number of elements =    88
  Number of element types =  1
  Number of global     degrees of freedom =   390
  Number of local      degrees of freedom =   378
  Number of prescribed degrees of freedom =    12


Mass properties of component "wings"

  Coordinates of reference point:      0.0000,     0.0000,      0.0000

  Rigid body mass matrix:

     3.9872e-02 -4.6374e-20 -1.3781e-20 -4.5716e-17  6.9525e+00 -1.2082e+02
     3.7329e-20  3.9872e-02  6.4080e-19 -6.9525e+00 -1.7724e-16  1.9225e+01
     1.8827e-20  5.9280e-19  3.9872e-02  1.2082e+02 -1.9225e+01  5.2001e-17
     7.5471e-17 -6.9525e+00  1.2082e+02  4.9551e+05 -5.6595e+04 -3.0150e+03
     6.9525e+00 -1.6932e-16 -1.9225e+01 -5.6595e+04  1.8475e+04 -3.7405e+04
    -1.2082e+02  1.9225e+01  2.6481e-16 -3.0150e+03 -3.7405e+04  5.0416e+05

  Mass =  3.9872e-02

  Inertia tensor with respect to reference point:

     4.9551e+05 -5.6595e+04 -3.0150e+03
    -5.6595e+04  1.8475e+04 -3.7405e+04
    -3.0150e+03 -3.7405e+04  5.0416e+05

  Coordinates of center of mass:   482.1608,  3030.1327,   174.3704

  Inertia tensor with respect to center of mass:

     1.2820e+05  1.6580e+03  3.3726e+02
     1.6580e+03  7.9932e+03 -1.6338e+04
     3.3726e+02 -1.6338e+04  1.2879e+05

--------------------------------------------------------------------------------
```

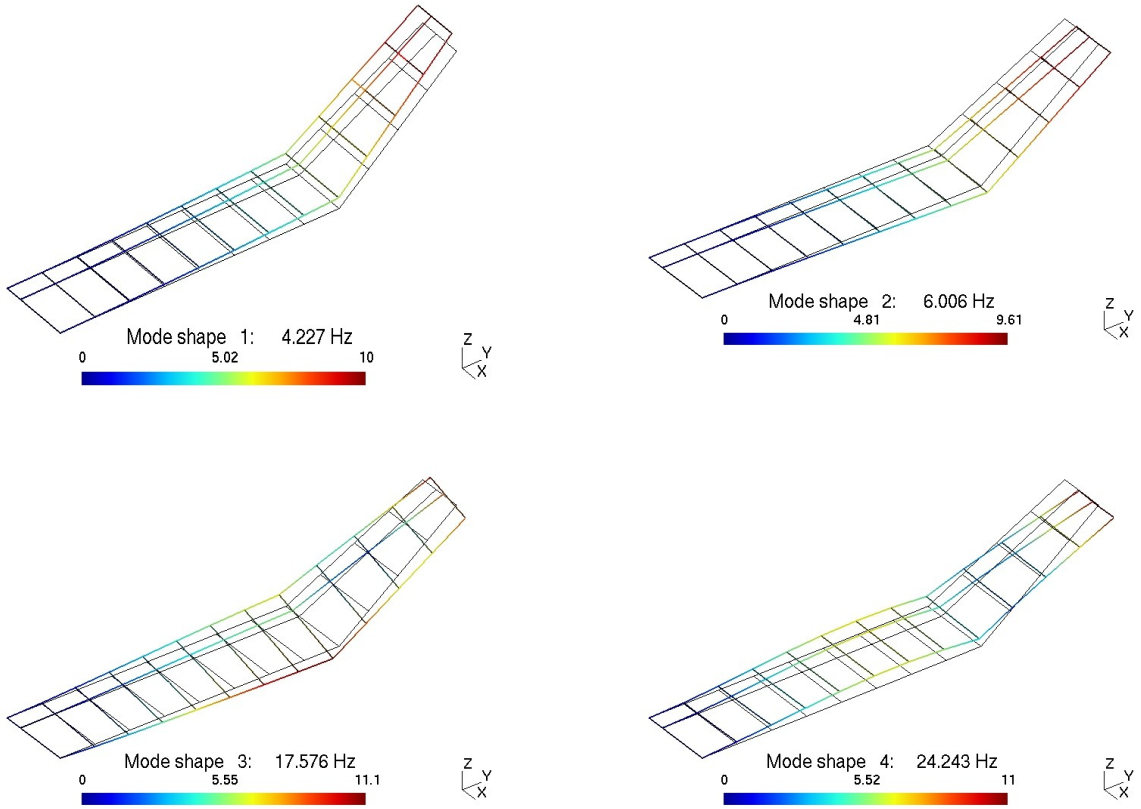


Abbildung 1.2: Einige Eigenschwingungen

```
Component "wings"

Natural frequencies:

  Mode    Circ. Frequency     Frequency
  ------------------------------------
     1         26.56003        4.22716 Hz
     2         37.73625        6.00591 Hz
     3        110.43352       17.57604 Hz
     4        152.32502       24.24328 Hz
     5        159.60354       25.40169 Hz
```

Die Masse des Flügels beträgt $0{,}03987$ t, entsprechend $39{,}87$ kg.

b) Aerodynamik-Modell

Das Aerodynamik-Modell besteht aus je einer Auftriebsfläche für den Innenflügel, den Außenflügel und das Querruder. Die Auftriebsflächen und ihre Diskretisierung sind in Abbildung 1.3 dargestellt.

Um eine über den Querruderbeginn gleichmäßige Diskretisierung zu erreichen, wird eine in Flügeltiefenrichtung gleichmäßige Unterteilung gewählt. Dabei wird die Anzahl der Panels so angepasst, dass die Panels des Querruders die gleiche Länge haben wie die des Flügels. In Spannweitenrichtung wird eine Verfeinerung in Richtung des Knicks und der Flügelspitze gewählt.

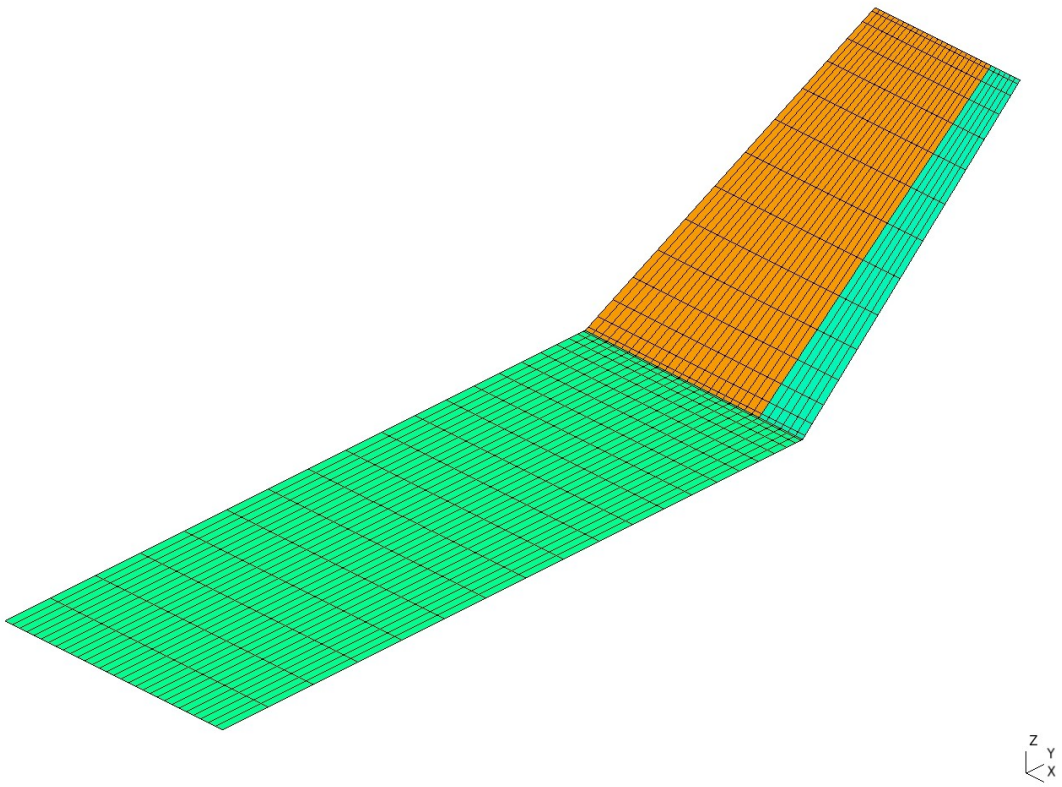


Abbildung 1.3: Aerodynamik-Modell

Das folgende GNU Octave-Skript erstellt das Berechnungsmodell und berechnet die Druckverteilung für die beiden Konfigurationen

1. $\alpha = 2°$ und $\eta = 0°$ sowie
2. $\alpha = 2°$ und $\eta = 2°$.

Die Komponente und die Dichte der Luft werden in der Datei `aero.bin` gespeichert. Die Daten für die graphische Darstellung werden in der Datei `rigid.plt` gespeichert. Diese Datei wird später benötigt, um die Druckverteilung des flexiblen Flügels mit der des starren Flügels zu vergleichen.

```
# Übungsblatt 5.1, Aufgabe 1: Knickflügel
#
# b) Aerodynamik-Modell
#
#    Erzeugte Dateien:
#      aero.bin   enthält aerodynamisches Modell und
#                 Druckverteilung
#      rigid.plt  enthält Daten für xy-Plot
#
# ------------------------------------------------------------

  warning("off", "Octave:missing-glyph");

  colors = [1, 0, 0; 0, 1, 0; 0, 0, 1; 1, 0, 0.5];
```

```
  set(0, "defaultlinelinewidth", 2);
  set(0, "defaultaxesfontsize", 10);
  set(0, "defaultaxescolororder", colors);

  fid = fopen("aero.res", "wt");

# Aerodynamik-Modell
# ------------------

  ya = 4000;  % y-Position des Knicks
  yt = 6000;  % y-Position der Flügelspitze
  zt = 1000;  % z-Position der Flügelspitze

  cr = 1500;  % Flügeltiefe an der Flügelwurzel
  ct = 1000;  % Flügeltiefe an der Flügelspitze
  k  =  0.2;  % Klappentiefenverhätlnis des Querruders

  nxi =  30;  % Panels in x-Richtung für den Innenflügel
  nxo =  24;  % Panels in x-Richtung für den Außenflügel
  nxa =   6;  % Panels in x-Richtung für das Querruder

  nyi =  20;  % Panels in y-Richtung für den Innenflügel
  nyo =  15;  % Panels in y-Richtung für den Außenflügel

  alpha =        2;   % Anstellwinkel in Grad
  eta   =   [0, 2];   % Querruderwinkel in Grad
  v     =     40E3;   % Anströmgeschwindigkeit in mm/s
  rho   = 1.21E-12;   % Luftdichte in t/mm^3

# Profildaten

  cmbi = mfs_airfoil("NACA", 5, 30);
  cmbo = mfs_airfoil("NACA", 5, 30, 0, 1 - k);
  cmba = mfs_airfoil("NACA", 5, 30, 1 - k, 1);

# Modelltyp

  aero.type    = "aero";
  aero.subtype =  "vlm";
  aero.symy    =      0;

# Punkte an der Flügelvorderkante

  points(1).id = 1; points(1).coor = [0,  0,  0];
  points(2).id = 2; points(2).coor = [0, ya,  0];
  points(3).id = 3; points(3).coor = [0, yt, zt];

# Punkte an der Querrudervorderkante

  points(4).id = 12; points(4).coor = [(1 - k) * cr, ya,  0];
  points(5).id = 13; points(5).coor = [(1 - k) * ct, yt, zt];

# Auftriebsfläche für den Innenflügel

  ls(1).id = 1; ls(1).points = [1, 2];
```

```
  ls(1).chord = cr; ls(1).camber = cmbi;
  ls(1).nx = nxi; ls(1).typex = "linear";
  ls(1).ny = nyi; ls(1).typey = "cos>";

# Auftriebsfläche für den Außenflügel

  ls(2).id = 2; ls(2).points = [2, 3];
  ls(2).chord = (1 - k) * [cr, ct]; ls(2).camber = cmbo;
  ls(2).nx = nxo; ls(2).typex = "linear";
  ls(2).ny = nyo; ls(2).typey = "cos";

# Auftriebsfläche für das Querruder

  ls(3).id = 3; ls(3).points = [12, 13];
  ls(3).chord = k * [cr, ct]; ls(3).camber = cmba;
  ls(3).nx = nxa; ls(3).typex = "linear";
  ls(3).ny = nyo; ls(3).typey = "cos";

# Querruder als Kontrollfläche

  controls(1).name = "aileron";
  controls(1).ls   = 3;

# Konfigurationen

  qdyn = 0.5 * rho * v^2;

  for n = 1 : 2
      config(n).name = ...
         sprintf("Conf. 1: alpha = %5.2f, eta = %5.2f",
                 alpha, eta(n));
      config(n).qdyn = qdyn; config(n).alpha = alpha;
      config(n).aileron = eta(n);
  end

# Definitionen zum Modell hinzufügen

  aero.points   = points;
  aero.ls       = ls;
  aero.controls = controls;
  aero.config   = config;

# Analyse
# -------

  winga = mfs_new(fid, aero);
  mfs_export("aero.msh", "msh", winga, "mesh");

  winga = mfs_statresp(winga);
  winga = mfs_results(winga, "statresp", "panel");

  mfs_export("aero.pos", "msh", winga, "statresp", "pressure");

  [xi(:, 1), pri(:, :, 1), y(1)] = ...
     mfs_xydata(winga, "statresp", "pressure", 1, 1);
```

```
[xi(:, 2), pri(:, :, 2), y(2)] = ...
   mfs_xydata(winga, "statresp", "pressure", 1,
              floor(0.8 * nyi));
pri = 1000 * pri;
xi  = xi / 1000;

[xo(:, 1), pro(:, :, 1), y(3)] = ...
   mfs_xydata(winga, "statresp", "pressure", [2, 3], 3);
[xo(:, 2), pro(:, :, 2), y(4)] = ...
   mfs_xydata(winga, "statresp", "pressure",
              [2, 3], floor(0.8 * nyo));
pro = 1000 * pro;
xo  = xo / 1000;

y = y / 1000;

for k = 1 : length(y)
    leg{k} = sprintf("y = %4.2f m", y(k));
end
for k = 1 : length(eta)
    header{k} = sprintf("\\eta = %2.0f\\circ", eta(k));
end

crm = cr / 1000;

figure(1, "position", [100, 500, 750, 500],
          "paperposition", [0, 0, 14, 10]);

subplot(1, 2, 1)
   plot(xi(:, 1), squeeze(pri(:, 1, 1)),
        xi(:, 2), squeeze(pri(:, 1, 2)),
        xo(:, 1), squeeze(pro(:, 1, 1)),
        xo(:, 2), squeeze(pro(:, 1, 2)));
   legend(leg, "location", "northeast");
   legend("boxoff"); legend("left");
   title(header{1}, "fontweight", "normal");
   axis([0, crm, -0.1, 0.9]);
   set(gca(), "xtick", 0 : 0.25 : crm);
   grid;
   xlabel("x [m]");
   ylabel("p [kPa]");

subplot(1, 2, 2)
   plot(xi(:, 1), squeeze(pri(:, 2, 1)),
        xi(:, 2), squeeze(pri(:, 2, 2)),
        xo(:, 1), squeeze(pro(:, 2, 1)),
        xo(:, 2), squeeze(pro(:, 2, 2)));
   title(header{2}, "fontweight", "normal");
   axis([0, crm, -0.1, 0.9]);
   set(gca(), "xtick", 0 : 0.25 : crm);
   grid;
   xlabel("x [m]");

saveas(1, "aero.jpg");
```

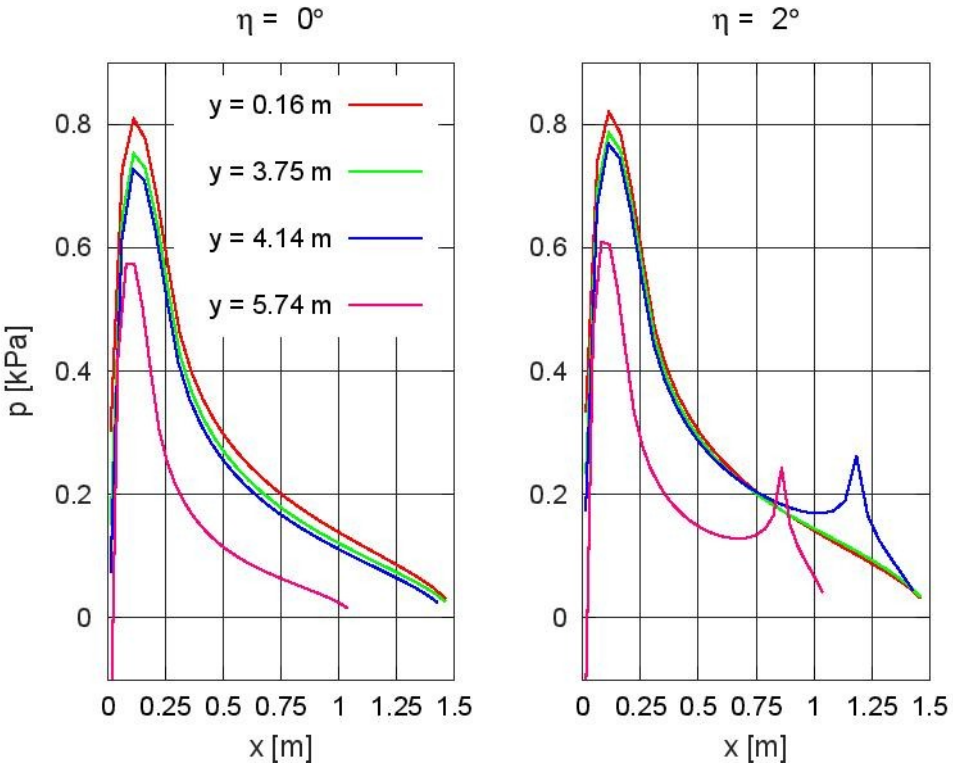


Abbildung 1.4: Druckverteilungen für den starren Flügel

```
save -binary aero.bin winga rho
save -binary rigid.plt crm nyi nyo eta xi xo pri pro

fclose(fid);
```

Abbildung 1.4 zeigt die Druckverläufe an vier typischen Flügelschnitten ohne und mit Querruderausschlag. Wie zu erwarten nimmt der Druck zur Flügelspitze hin ab. Deutlich zu sehen ist der Druckanstieg an der Vorderkante des Querruders, wenn das Querruder ausgeschlagen ist.

c) Aeroelastik-Modell

Zur Kopplung des Strukturmodells mit dem Aerodynamik-Modell müssen die Splines definiert werden. Für die drei Auftriebsflächen werden drei Splines benötigt. Wegen des Knicks des Flügels wird die Auftriebsfläche des Innenflügels an die Strukturknoten des Innenflügels angeschlossen, während die Auftriebsflächen des Außenflügels und des Querruders an die Strukturknoten des Außenflügels angeschlossen werden. Die Knotenpunkt-Sets des Innen- und Außenflügels wurden in Teilaufgabe a) definiert. Sie sind in der Datei `solid.bin` gespeichert.

Das Strukturmodell hat neun Knotenpunktsreihen (Punkte mit gleicher *y*-Koordinate) für den Innenflügel und fünf Knotenpunktsreihen für den Außenflügel. Daher werden für den Spline des Innenflügels acht Spline-Segmente und für die Splines des Außenflügels fünf Spline-Segmente gewählt. Die Anzahl

der Spline-Segmente darf nicht größer sein als die Anzahl der Punkte mit unterschiedlicher *y*-Koordinate.

Der folgende Abschnitt des GNU Octave-Skripts definiert das Aeroelastik-Modell:

```
# Übungsblatt 5.1, Aufgabe 1: Knickflügel
#
# c) Aeroelastik-Modell
#
#    Benötigte Dateien: solid.bin, aero.bin
#
#    Erzeugte Dateien:
#      aeroelastic.bin  enthält das aeroelastische Modell
#
# ------------------------------------------------------------

  fid = fopen("aeroelastic.res", "wt");

# Aeroelastik-Modell
# ------------------

  model.type = "aeroelastic";

  load solid.bin
  load aero.bin

  model.solid = wings;
  model.aero  = winga;

# Splines

  spline(1).id = 1; spline(1).type = "tb"; spline(1).lsid = 1;
  spline(1).nodes = nset.inner_wing;
  spline(1).data.nbreaks = 8;

  spline(2).id = 2; spline(2).type = "tb"; spline(2).lsid = 2;
  spline(2).nodes = nset.outer_wing;
  spline(2).data.nbreaks = 4;

  spline(3).id = 3; spline(3).type = "tb"; spline(3).lsid = 3;
  spline(3).nodes = nset.outer_wing;
  spline(3).data.nbreaks = 4;

  model.splines = spline;
```

Zur Kontrolle der Splines werden die in Teilaufgabe b) berechneten aerodynamischen Lasten auf das Strukturmodell übertragen und die resultierenden Lasten verglichen:

```
# Analyse
# -------

# Komponente und Splines erzeugen
```

```
  wing = mfs_new(fid, model);
  wing = mfs_splines(wing);

  save -binary aeroelastic.bin wing rho

# Aerodynamische Lasten aus u5_1_1b auf Struktur übertragen

  wings = mfs_transfer(wing, winga, "statresp", "loads");
  [FA, MA] = mfs_getresp(winga, "statresp", "aeload");
  [FS, MS] = mfs_getresp(wings, "load", "resultant");

  nc = columns(FA);

  fprintf(fid, "Load resultants of aerodynamic component: \n\n");
  for k = 1 : nc
      fprintf(fid, "  Configuration %2d:\n", k)
      fprintf(fid, "     F = [%10.3e, %10.3e, %10.3e] kN\n",
              FA(:, k) / 1000);
      fprintf(fid, "     M = [%10.3e, %10.3e, %10.3e] kNm\n",
              MA(:, k) * 1e-6);
  end

  fprintf(fid, "\nLoad resultants of solid component: \n\n");
  for k = 1 : nc
      fprintf(fid, "  Configuration %2d:\n", k)
      fprintf(fid, "     F = [%10.3e, %10.3e, %10.3e] kN\n",
              FS(:, k) / 1000);
      fprintf(fid, "     M = [%10.3e, %10.3e, %10.3e] kNm\n",
              MS(:, k) * 1e-6);
  end
```

Anschließend werden die Verschiebungen der Strukturknoten infolge der aerodynamischen Lasten berechnet und auf die Knoten des Aerodynamik-Modells übertragen:

```
# Verschiebungen

  wings = mfs_statresp(wings);
  winga = mfs_transfer(wing, wings, "statresp", "disp");

  mfs_export("solid.dsp", "msh", wings, "statresp", "disp");
  mfs_export("aero.dsp", "msh", winga, "statresp", "disp");

# Modelle und Ergebnisse kombinieren

  mfs_merge("solid.msh", "aero.msh", "wing.msh", "msh");
  mfs_merge("solid.dsp", "aero.dsp", "wing.dsp", "msh");

  fclose(fid);
```

Die resultierenden Kräfte und Momente sind in der Ausgabedatei enthalten.

```
Mefisto 2.4: Building new component from input "model"

  Model Type = aeroelastic

  Number of splines      =           3
```

```
Load resultants of aerodynamic component:

  Configuration  1:
     F = [ 0.000e+00, -2.616e-01,  2.141e+00] kN
     M = [ 5.817e+00, -9.063e-01, -9.915e-02] kNm
  Configuration  2:
     F = [ 0.000e+00, -3.342e-01,  2.351e+00] kN
     M = [ 6.735e+00, -1.041e+00, -1.504e-01] kNm

Load resultants of solid component:

  Configuration  1:
     F = [ 0.000e+00, -2.616e-01,  2.141e+00] kN
     M = [ 5.817e+00, -9.063e-01, -9.915e-02] kNm
  Configuration  2:
     F = [ 0.000e+00, -3.342e-01,  2.351e+00] kN
     M = [ 6.735e+00, -1.041e+00, -1.504e-01] kNm
```

Die Resultierenden der Kräfte auf der Struktur stimmen mit den Resultierenden der aerodynamischen Kräfte überein.

Abbildung 1.5 zeigt die Verschiebungen der Knoten des Aerodynamik-Modells und des Strukturmodells für beide Konfigurationen. Die Verschiebungskomponente senkrecht zu den Auftriebsflächen ist für beide Modelle identisch. Verschiebungskomponenten tangential zu den Auftriebsflächen werden nicht auf das Aerodynamik-Modell übertragen, da sie keinen Einfluss auf die Aerodynamik haben.

Die Übereinstimmung der Resultierenden und der Verschiebungskomponenten senkrecht zu den Auftriebflächen zeigt, dass die Splines korrekt definiert sind.

d) Statische Divergenz

Das folgende GNU Octave-Skript berechnet den Staudruck, bei dem statische Divergenz auftritt, und daraus die zugehörige Geschwindigkeit. Die Datei `aeroelastic.bin` enthält das Aerodynamik-Modell und die Dichte der

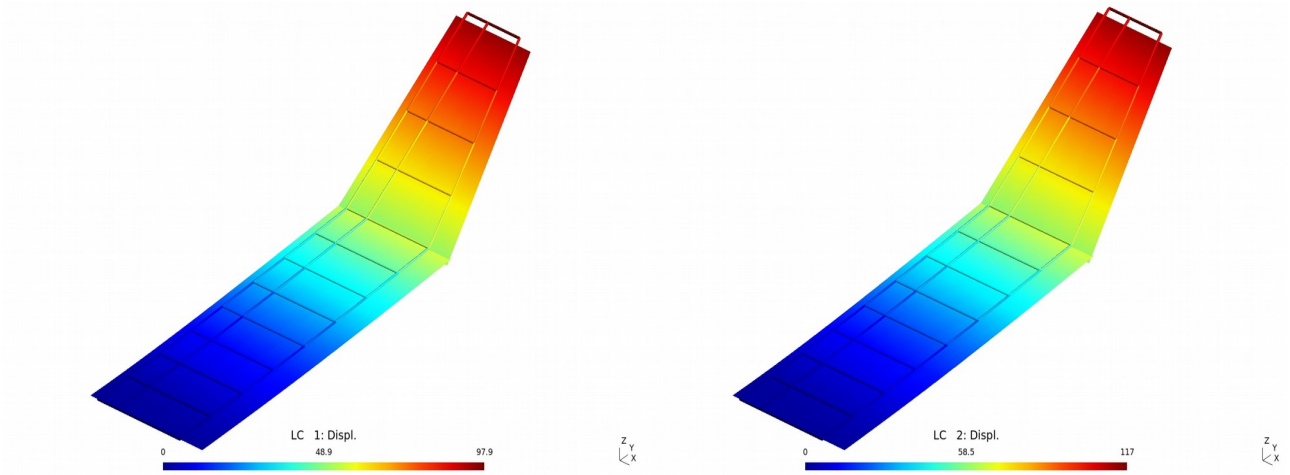


Abbildung 1.5: Vergleich der Verschiebungen

Luft. Die Ergebnisse der Divergenzanalyse werden in den Komponenten **wings** (Strukturmodell) und **winga** (Aerodynamik-Modell) gespeichtert.

```
# Übungsblatt 5.1, Aufgabe 1: Knickflügel
#
# d) Statische Divergenz
#
#    Benötigte Dateien: aeroelastic.bin
#
# ------------------------------------------------------------------

  fid = fopen("diverg.res", "wt");

  load aeroelastic.bin

# Analyse
# -------

  [wings, winga, nfound] = mfs_diverg(wing, 2);
  if (nfound)
     mfs_print(fid, winga, "diverg", "qdyn");
     qd = mfs_getresp(winga, "diverg", "qdyn", 1);
     vd = sqrt(2 * qd / rho) / 1000;
     fprintf(fid, "\n  qd = %12.4e MPa, vd = %7.2f m/s\n",
             qd, vd);
  end

  fclose(fid);
```

Die Ausgabedatei hat den folgenden Inhalt:

```
------------------------------------------------------------------------------

Component "winga"

Dynamic pressure at divergence:

  No.    Dynamic pressure
  ----------------------
   1        6.6019e-02


  qd =   6.6019e-02 MPa, vd =  330.34 m/s
```

Die Geschwindigkeit, bei der statische Divergenz auftritt, entspricht fast der Schallgeschwindigkeit. In diesem Geschwindigkeitsbereich kann die Strömung nicht mehr als inkompressibel betrachtet werden. In dem Geschwindigkeitsbereich, in dem die Strömung als inkompressibel betrachtet werden kann, d. h. für Strömungsgeschwindigkeiten bis etwa $100\ \mathrm{m/s}$, tritt keine statische Divergenz auf.

e) Flexibler Flügel

Die Berechnungen für den flexiblen Flügel werden mit der aeroelastischen Komponente durchgeführt. Die primären Ergebnisse, das heißt die Verschie-

bungen und die Wirbelstärken, werden in der Struktur-Komponente und der Aerodynamik-Komponente gespeichert. Daraus lassen sich dann alle weiteren Ergebnisse berechnen.

Der folgende Abschnitt des GNU Octave-Skripts berechnet die Ergebnisse:

```
# Übungsblatt 5.1, Aufgabe 1: Knickflügel
#
# e) Aerodynamische Analyse
#
#    Benötigte Dateien: aeroelastic.bin, rigid.plt
#
# ------------------------------------------------------------

  warning("off", "Octave:missing-glyph");

  set(0, "defaultlinelinewidth", 2);
  set(0, "defaultaxesfontsize", 10);

  fid = fopen("flexible.res", "wt");

  load aeroelastic.bin

# Analyse
# -------

# Statische Antwort

  [wings, winga] = mfs_statresp(wing);

  winga = mfs_results(winga, "statresp", "panel");
  mfs_export("solid.dsp", "msh", wings, "statresp", "disp");
  mfs_export("aero.pos", "msh", winga, "statresp",
             "pressure", "disp");

# Resultierende Kräfte

  [FA, MA] = mfs_getresp(winga, "statresp", "aeload");
  nc       = columns(FA);

  fprintf(fid, "Load resultants of aerodynamic component: \n\n");
  for k = 1 : nc
      fprintf(fid, "  Configuration %2d:\n", k)
      fprintf(fid, "     F = [%10.3e, %10.3e, %10.3e] kN\n",
              FA(:, k) / 1000);
      fprintf(fid, "     M = [%10.3e, %10.3e, %10.3e] kNm\n",
              MA(:, k) * 1e-6);
  end
```

Anschließend wird der Verlauf des Drucks in ausgewählten Flügelschnitten aus der Komponente extrahiert und zusammen mit den Ergebnisse des starren Flügels graphisch dargestellt:

```
# Vergleich der Druckverteilungen
```

```
load rigid.plt

[xi(:, 1), pfi(:, :, 1), y(1)] = ...
   mfs_xydata(winga, "statresp", "pressure", 1, 1);
[xi(:, 2), pfi(:, :, 2), y(2)] = ...
   mfs_xydata(winga, "statresp", "pressure", 1,
              floor(0.8 * nyi));
pfi = 1000 * pfi;
xi  = xi / 1000;

[xo(:, 1), pfo(:, :, 1), y(3)] = ...
   mfs_xydata(winga, "statresp", "pressure", [2, 3], 3);
[xo(:, 2), pfo(:, :, 2), y(4)] = ...
   mfs_xydata(winga, "statresp", "pressure",
              [2, 3], floor(0.8 * nyo));
pfo = 1000 * pfo;
xo  = xo / 1000;

y = y / 1000;

for k = 1 : length(y)
    header{k} = sprintf("\\eta = %1d\\circ, y = %4.2f m",
                        eta(1), y(k));
end

figure(1, "position", [100, 500, 750, 500],
          "paperposition", [0, 0, 14, 10]);

subplot(2, 2, 1)
   plot(xi(:, 1), squeeze(pri(:, 1, 1)), "color", "red",
        xi(:, 1), squeeze(pfi(:, 1, 1)), "color", "green");
   legend("rigid", "flexible", "location", "northeast");
   legend("boxoff"); legend("left");
   title(header{1}, "fontweight", "normal");
   axis([0, crm, -0.1, 0.8]);
   set(gca(), "xtick", 0 : 0.25 : crm);
   grid;
   ylabel("p [kPa]");

subplot(2, 2, 2)
   plot(xi(:, 2), squeeze(pri(:, 1, 2)), "color", "red",
        xi(:, 2), squeeze(pfi(:, 1, 2)), "color", "green");
   legend("rigid", "flexible", "location", "northeast");
   legend("boxoff"); legend("left");
   title(header{2}, "fontweight", "normal");
   axis([0, crm, -0.1, 0.8]);
   set(gca(), "xtick", 0 : 0.25 : crm);
   grid;

subplot(2, 2, 3)
   plot(xo(:, 1), squeeze(pro(:, 1, 1)), "color", "red",
        xo(:, 1), squeeze(pfo(:, 1, 1)), "color", "green");
   legend("rigid", "flexible", "location", "northeast");
   legend("boxoff"); legend("left");
   title(header{3}, "fontweight", "normal");
```

```
    axis([0, crm, -0.1, 0.8]);
    set(gca(), "xtick", 0 : 0.25 : crm);
    grid;
    xlabel("x [m]");
    ylabel("p [kPa]");

subplot(2, 2, 4)
    plot(xo(:, 2), squeeze(pro(:, 1, 2)), "color", "red",
         xo(:, 2), squeeze(pfo(:, 1, 2)), "color", "green");
    legend("rigid", "flexible", "location", "northeast");
    legend("boxoff"); legend("left");
    title(header{4}, "fontweight", "normal");
    axis([0, crm, -0.1, 0.8]);
    set(gca(), "xtick", 0 : 0.25 : crm);
    grid;
    xlabel("x [m]");

print("flexible1.jpg", "-djpg");

for k = 1 : length(y)
    header{k} = sprintf("\\eta = %1d\\circ, y = %4.2f m",
                        eta(2), y(k));
end

figure(2, "position", [250, 250, 750, 500],
          "paperposition", [0, 0, 14, 10]);

subplot(2, 2, 1)
    plot(xi(:, 1), squeeze(pri(:, 2, 1)), "color", "red",
         xi(:, 1), squeeze(pfi(:, 2, 1)), "color", "green");
    legend("rigid", "flexible", "location", "northeast");
```

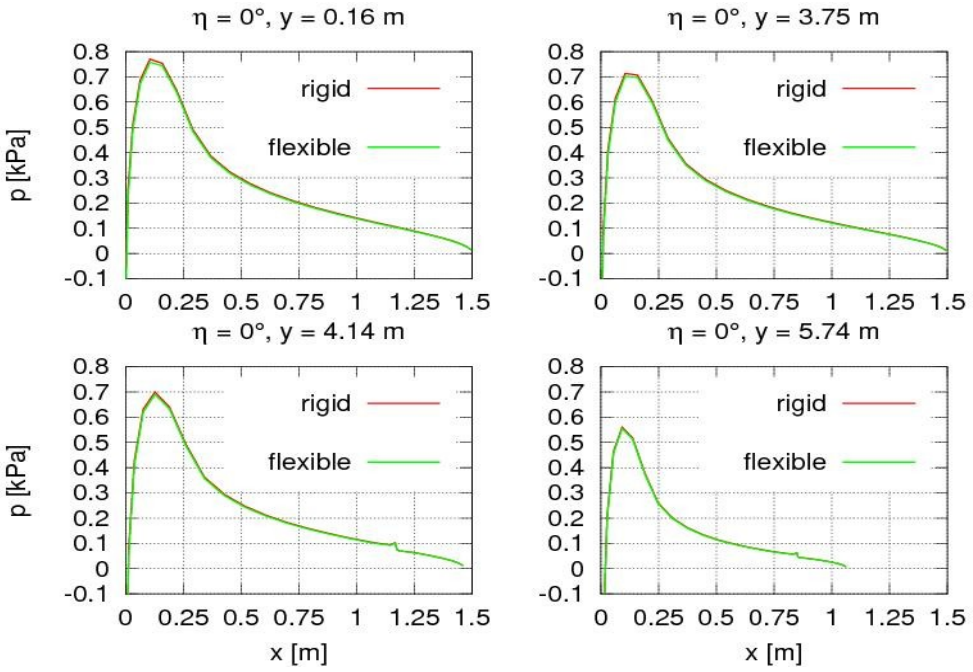


Abbildung 1.6: Druckverläufe ohne Querruderausschlag

```
    legend("boxoff"); legend("left");
    title(header{1}, "fontweight", "normal");
    axis([0, crm, -0.1, 0.9]);
    set(gca(), "xtick", 0 : 0.25 : crm);
    grid;
    ylabel("p [kPa]");

subplot(2, 2, 2)
    plot(xi(:, 2), squeeze(pri(:, 2, 2)), "color", "red",
         xi(:, 2), squeeze(pfi(:, 2, 2)), "color", "green");
    legend("rigid", "flexible", "location", "northeast");
    legend("boxoff"); legend("left");
    title(header{2}, "fontweight", "normal");
    axis([0, crm, -0.1, 0.9]);
    set(gca(), "xtick", 0 : 0.25 : crm);
    grid;

subplot(2, 2, 3)
    plot(xo(:, 1), squeeze(pro(:, 2, 1)), "color", "red",
         xo(:, 1), squeeze(pfo(:, 2, 1)), "color", "green");
    legend("rigid", "flexible", "location", "northeast");
    legend("boxoff"); legend("left");
    title(header{3}, "fontweight", "normal");
    axis([0, crm, -0.1, 0.9]);
    set(gca(), "xtick", 0 : 0.25 : crm);
    grid;
    xlabel("x [m]");
    ylabel("p [kPa]");

subplot(2, 2, 4)
    plot(xo(:, 2), squeeze(pro(:, 2, 2)), "color", "red",
```

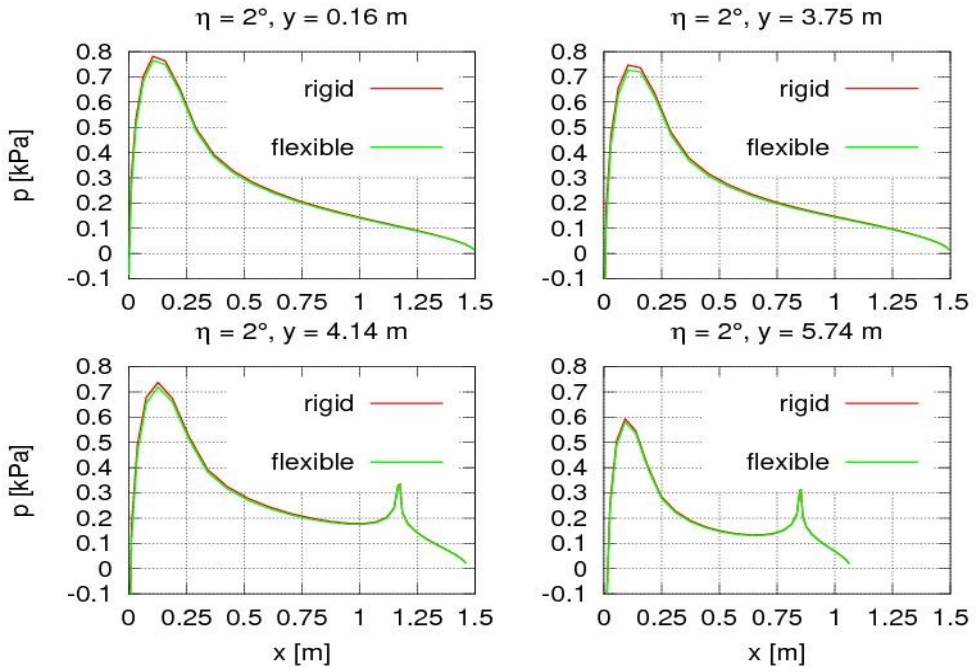


Abbildung 1.7: Druckverläufe mit Querruderausschlag

```
          xo(:, 2), squeeze(pfo(:, 2, 2)), "color", "green");
    legend("rigid", "flexible", "location", "northeast");
    legend("boxoff"); legend("left");
    title(header{4}, "fontweight", "normal");
    axis([0, crm, -0.1, 0.9]);
    set(gca(), "xtick", 0 : 0.25 : crm);
    grid;
    xlabel("x [m]");

  print("flexible2.jpg", "-djpg");

  fclose(fid);
```

Die Abbildungen 1.6 und 1.7 zeigen die Druckverläufe ohne und mit Querruderausschlag für den starren und den flexiblen Flügel. Da die Geschwindigkeit weit unterhalb der Geschwindigkeit für statische Divergenz liegt, unterscheiden sich die Druckverläufe zwischen starrem und flexiblem Flügel kaum.

Aufgabe 2

Um spätere Parameterstudien zu erleichtern, werden alle Parameter, die die Geometrie, die Querschnittsabmessungen, das Material und die Diskretisierung beschreiben, in einer eigenen Datei `params.m` definiert. Die Datei hat folgenden Inhalt:

```
# Übungsblatt 5.1, Aufgabe 2: Flügel
#
#     Definition aller Parameter
#
#     Einheiten: N, mm, t, Grad
#
# ------------------------------------------------------------

# Geometrie
# ---------

  sweep =   -5;   % Pfeilungswinkel
  ya    = 5000;   % y-Position des Querruders
  yt    = 8000;   % y-Position der Flügelspitze
  cr    = 1200;   % Flügeltiefe an der Flügelwurzel
  ct    =  800;   % Flügeltiefe an der Flügelspitze

# Strukturmodell
# --------------

# Material: Aluminium

  E   =  70000;
  ny  =   0.34;
  rho = 2.7E-9;

# Querschnitte
```

```
  b_Rippen = 20;
  h_Rippen = 20;
  s_Rippen =  2;

  b_Stringer_1 = 30;
  h_Stringer_1 = 30;
  t_Stringer_1 =  2;

  b_Stringer_2 = 50;
  h_Stringer_2 = 40;
  t_Stringer_2 =  4;

  b_Stringer_3 = 40;
  h_Stringer_3 = 30;
  t_Stringer_3 =  4;

  b_Stringer_4 = 20;
  h_Stringer_4 = 20;
  t_Stringer_4 =  2;

# Blechdicken

  t_Holm_1 = 1;
  t_Holm_2 = 2;
  t_Holm_3 = 2;
  t_Holm_4 = 1;

  t_Rippen = 1;

  t_Haut = 1;

# Aerodynamik-Modell
# ------------------

# Geometrie

  kp   = 0.2;   % Klappentiefenverhältnis
  atip =  -3;   % Einstellwinkel an der Flügelspitze

# Konfiguration

  alpha =      2; % Anstellwinkel in Grad
  eta   = [0, 2]; % Querruderausschlag in Grad

# Auswertung

  ycoli = [1, 10]; % Panelreihen für Druckausgabe, Innenflügel
  ycolo = [3, 10]; % Panelreihen für Druckausgabe, Außenflügel

# Diskretisierung

  nxi   =   40;   % Panels in x-Richtung für den Innenflügel
  nxo   =   32;   % Panels in x-Richtung für den Außenflügel
  nxa   =    8;   % Panels in x-Richtung für das Querruder
```

```
  nyi   =   15;   % Panels in y-Richtung für den Innenflügel
  nyo   =   15;   % Panels in y-Richtung für den Außenflügel

# Aeroelastik-Modell
# ------------------

  nseg  = 10;     % Spline-Segmente
```

Die Diskretisierungsparameter für die Auftriebsflächen sind so gewählt, dass sich eine gleichmäßige Unterteilung in *x*-Richtung ergibt.

a) Berechnungsmodell der Struktur

Das folgende GNU Octave-Skript erstellt das Strukturmodell und berechnet die ersten zehn Eigenschwingungen. Zuerst werden die Parameter eingelesen und Gmsh gestartet, um die Vernetzung zu erstellen. Dem Aufruf von Gmsh werden die aktuellen Werte der Parameter als Argumente übergeben (Schlüsselworte **`-setnumber`**). Damit werden die in der Datei `solid.geo` definierten Konstanten überschrieben.

```
# Übungsblatt 5.1, Aufgabe 2: Flügel
#
# a) Strukturmodell: Definition und Berechnung der ersten 10
#                    Eigenschwingungen
#
#    Erzeugte Dateien: solid.bin  Komponente mit Struktur
#                      solid.set  Elementsets
#
# ------------------------------------------------------------

  fid  = fopen("modes.res", "wt");

  nofmod = 10;      % Anzahl Eigenschwingungen
  debug  =  0;      % Geometrie-Informationen ausgeben:
                    %    0 = nein, 1 = ja

# Modell definieren
# -----------------

# Daten definieren

  params

# Vernetzung erstellen

  keywords = "-2 -o solid.msh ";
  keywords = [keywords, sprintf("-setnumber sweep %f ", sweep)];
  keywords = [keywords, sprintf("-setnumber b %f ", yt)];
  keywords = [keywords, sprintf("-setnumber c_root %f ", cr)];
  keywords = [keywords, sprintf("-setnumber c_tip %f ", ct)];

  if (debug)
```

```
    keywords = [keywords,  ...
                sprintf("-setnumber debug %d ", debug)];
end

gmshcmd = ["gmsh ", "solid.geo ", keywords];

status = system(gmshcmd);
if (status)
   error("gmsh: Vernetzung gescheitert\n");
end
```

Die erzeugte Vernetzung ist in Abbildung 2.1 dargestellt.

Anschließend werden die Übersetzungsdaten definiert:

```
# Übersetzungsdaten

  mat.type = "iso";
  mat.E    = E;
  mat.ny   = ny;
  mat.rho  = rho;

  data.type    = "solid";
  data.subtype = "3d";

# Rippen

  [geom, e] = mfs_beamsection("T", b_Rippen, h_Rippen, s_Rippen);
  geom.v    = [0, 0, -1];
  geom.P    = [0, -e(1)];
```

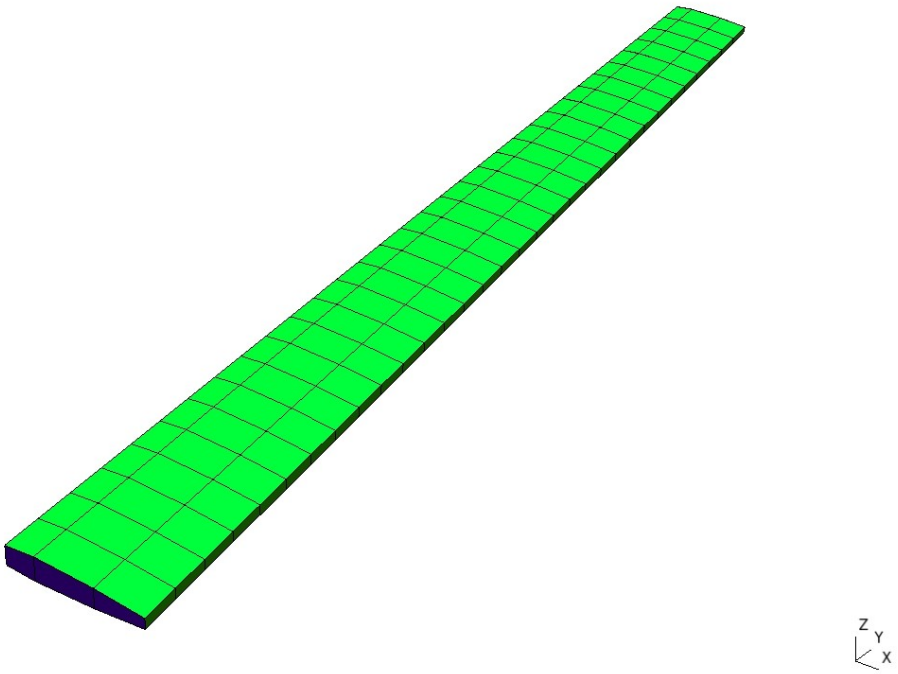


Abbildung 2.1: Berechnungsmodell der Struktur

```
  Ribs_Upper_Part.type = "elements";
  Ribs_Upper_Part.name = "b2";
  Ribs_Upper_Part.geom = geom;
  Ribs_Upper_Part.mat  = mat;

  Ribs_Lower_Part        = Ribs_Upper_Part;
  Ribs_Lower_Part.geom.v = [0, 0, 1];

  data.Ribs_Upper_Part = Ribs_Upper_Part;
  data.Ribs_Lower_Part = Ribs_Lower_Part;

  [geom, e] = mfs_beamsection("T", b_Rippen, h_Rippen, s_Rippen);
  geom.v    = [1, 0, 0];
  geom.P    = [0, -e(1)];
  Ribs_Front_Part.type = "elements";
  Ribs_Front_Part.name = "b2";
  Ribs_Front_Part.geom = geom;
  Ribs_Front_Part.mat  = mat;

  Ribs_End_Part = Ribs_Front_Part;

  Ribs_End_Part.geom.v = [-1, 0, 0];

  data.Ribs_Front_Part = Ribs_Front_Part;
  data.Ribs_End_Part   = Ribs_End_Part;

  geoms.t = t_Rippen;

  Ribs.type = "elements";
  Ribs.name = "q4";
  Ribs.geom = geoms;
  Ribs.mat  = mat;

  data.Ribs = Ribs;

# Stringer

  [geom, e] = mfs_beamsection("T", b_Stringer_1,
                                   h_Stringer_1, t_Stringer_1);
  geom.v    = [0, 0, -1];
  geom.P    = [0, -e(1)];
  Stringer_1_Upper.type = "elements";
  Stringer_1_Upper.name = "b2";
  Stringer_1_Upper.geom = geom;
  Stringer_1_Upper.mat  = mat;

  Stringer_1_Lower        = Stringer_1_Upper;
  Stringer_1_Lower.geom.v = [0, 0, 1];

  data.Stringer_1_Upper = Stringer_1_Upper;
  data.Stringer_1_Lower = Stringer_1_Lower;

  [geom, e] = mfs_beamsection("T", b_Stringer_2,
                                   h_Stringer_2, t_Stringer_2);
  geom.v    = [0, 0, -1];
```

```
  geom.P    = [0, -e(1)];
  Stringer_2_Upper.type = "elements";
  Stringer_2_Upper.name = "b2";
  Stringer_2_Upper.geom = geom;
  Stringer_2_Upper.mat  = mat;

  Stringer_2_Lower        = Stringer_2_Upper;
  Stringer_2_Lower.geom.v = [0, 0, 1];

  data.Stringer_2_Upper = Stringer_2_Upper;
  data.Stringer_2_Lower = Stringer_2_Lower;

  [geom, e] = mfs_beamsection("T", b_Stringer_3,
                                   h_Stringer_3, t_Stringer_3);
  geom.v    = [0, 0, -1];
  geom.P    = [0, -e(1)];
  Stringer_3_Upper.type = "elements";
  Stringer_3_Upper.name = "b2";
  Stringer_3_Upper.geom = geom;
  Stringer_3_Upper.mat  = mat;

  Stringer_3_Lower        = Stringer_3_Upper;
  Stringer_3_Lower.geom.v = [0, 0, 1];

  data.Stringer_3_Upper = Stringer_3_Upper;
  data.Stringer_3_Lower = Stringer_3_Lower;

  [geom, e] = mfs_beamsection("T", b_Stringer_4,
                                   h_Stringer_4, t_Stringer_4);
  geom.v    = [0, 0, -1];
  geom.P    = [0, -e(1)];
  Stringer_4_Upper.type = "elements";
  Stringer_4_Upper.name = "b2";
  Stringer_4_Upper.geom = geom;
  Stringer_4_Upper.mat  = mat;

  Stringer_4_Lower        = Stringer_4_Upper;
  Stringer_4_Lower.geom.v = [0, 0, 1];

  data.Stringer_4_Upper = Stringer_4_Upper;
  data.Stringer_4_Lower = Stringer_4_Lower;

# Schubfeld des vorderen Holms

  geoms.t = t_Holm_1;

  Front_Spar.type = "elements";
  Front_Spar.name = "q4";
  Front_Spar.geom = geoms;
  Front_Spar.mat  = mat;

  data.Front_Spar = Front_Spar;

# Schubfeld des vorderen Hauptholms
```

```
  geoms.t = t_Holm_2;

  Main_Spar.type = "elements";
  Main_Spar.name = "q4";
  Main_Spar.geom = geoms;
  Main_Spar.mat  = mat;

  data.Main_Spar = Main_Spar;

# Schubfeld des hinteren Hauptholms

  geoms.t = t_Holm_3;

  Back_Spar.type = "elements";
  Back_Spar.name = "q4";
  Back_Spar.geom = geoms;
  Back_Spar.mat  = mat;

  data.Back_Spar = Back_Spar;

# Schubfeld des Endholms

  geoms.t = t_Holm_4;

  Rear_Spar.type = "elements";
  Rear_Spar.name = "q4";
  Rear_Spar.geom = geoms;
  Rear_Spar.mat  = mat;

  data.Rear_Spar = Rear_Spar;

# Außenhaut

  geoms.t = t_Haut;

  Skin.type = "elements";
  Skin.name = "q4";
  Skin.geom = geoms;
  Skin.mat  = mat;

  data.Skin = Skin;

# Einspannung

  Clamped.type = "constraints";
  Clamped.name = "prescribed";
  Clamped.dofs = 1 : 6;

  data.Clamped = Clamped;
```

Danach wird die Vernetzung eingelesen, die Komponente erstellt und die Berechnung durchgeführt. Die Struktur **`eset`** enthält die Elementsets, die aus den physikalischen Gruppen erzeugt wurden. Sie werden später in Teilaufgabe e) benötigt und deshalb in der Datei `solid.set` gespeichert. Die Kompo-

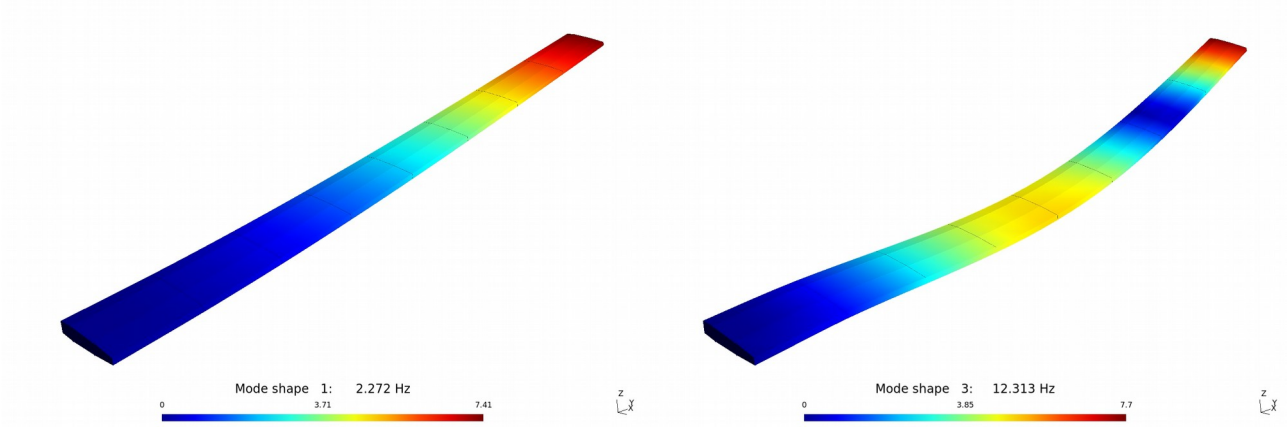
nente wird für die folgenden Teilaufgaben in der Datei `solid.bin` gespeichert.

```
# Rechnung
# --------

  [model, ~, eset] = mfs_import(fid, "solid.msh", "msh", data);
  wings = mfs_new(fid, model);
  mfs_export("solid.axes", "msh", wings, "mesh", "axesonly");

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

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

  save -binary solid.bin wings
  save -binary solid.set eset

  fclose(fid);
```

Einige Biege- und Torsionsschwingungen sind in Abbildung 2.2 dargestellt.

Die Ausgabedatei enthält folgende Ergebnisse:

```
Reading model from file "solid.msh", MSH file version 4.1

  Physical Group                    Type
  ------------------------------------------------
  Ribs_Upper_Part                   elements
  Ribs_Lower_Part                   elements
  Ribs_Front_Part                   elements
  Ribs_End_Part                     elements
  Stringer_1_Upper                  elements
  Stringer_2_Upper                  elements
  Stringer_3_Upper                  elements
  Stringer_4_Upper                  elements
  Stringer_1_Lower                  elements
  Stringer_2_Lower                  elements
  Stringer_3_Lower                  elements
  Stringer_4_Lower                  elements
  Front_Spar                        elements
  Main_Spar                         elements
  Back_Spar                         elements
  Rear_Spar                         elements
  Ribs                              elements
  Skin                              elements
  Clamped                           constraints

Mefisto 2.4: Building new component from input "model"

  Model Type = solid, Model Subtype = 3d

  Number of nodes      =   264,  Number of elements =   675
  Number of element types =  2
  Number of global     degrees of freedom =  1584
  Number of local      degrees of freedom =  1560
  Number of prescribed degrees of freedom =    24


Mass properties of component "wings"
```



1. vertikale Biegeschwingung | 2. vertikale Biegeschwingung

1. Torsionsschwingung | 2. Torsionsschwingung

Abbildung 2.2: Einige Eigenschwingungen

```
 Coordinates of reference point:       0.0000,      0.0000,       0.0000

 Rigid body mass matrix:

    8.2165e-02  5.2095e-20  1.6777e-20  8.1391e-17  1.0635e+00 -3.1517e+02
    3.9222e-22  8.2165e-02 -4.2913e-21 -1.0635e+00 -7.3817e-18 -3.5166e+01
    1.2775e-20 -4.4401e-21  8.2165e-02  3.1517e+02  3.5166e+01 -5.8717e-17
    6.6281e-17 -1.0635e+00  3.1517e+02  1.6521e+06  1.7190e+05  5.1642e+02
    1.0635e+00 -9.2954e-18  3.5166e+01  1.7190e+05  2.2530e+04 -3.8009e+03
   -3.1517e+02 -3.5166e+01 -7.2533e-17  5.1642e+02 -3.8009e+03  1.6742e+06

 Mass =  8.2165e-02

 Inertia tensor with respect to reference point:

    1.6521e+06  1.7190e+05  5.1642e+02
    1.7190e+05  2.2530e+04 -3.8009e+03
    5.1642e+02 -3.8009e+03  1.6742e+06

 Coordinates of center of mass:  -427.9940,  3835.7737,    12.9440

 Inertia tensor with respect to center of mass:

    4.4319e+05  3.7007e+04  6.1229e+01
    3.7007e+04  7.4657e+03  2.7856e+02
    6.1229e+01  2.7856e+02  4.5026e+05

--------------------------------------------------------------------------------

Component "wings"
```

```
Natural frequencies:

  Mode    Circ. Frequency     Frequency
  ----------------------------------------
     1         14.27354        2.27170 Hz
     2         58.31315        9.28083 Hz
     3         77.36441       12.31293 Hz
     4        188.59327       30.01555 Hz
     5        204.98570       32.62449 Hz
     6        305.88484       48.68308 Hz
     7        388.71297       61.86559 Hz
     8        462.03858       73.53572 Hz
     9        621.61906       98.93375 Hz
    10        726.15189      115.57066 Hz
```

Die Masse des Flügels beträgt $0{,}08217$ t, entsprechend $82{,}17$ kg.

b) Aerodynamik-Modell

Das Aerodynamik-Modell besteht aus je einer Auftriebsfläche für den Innenflügel, den Außenflügel und das Querruder. Die Auftriebsflächen und ihre Diskretisierung sind in Abbildung 2.3 dargestellt.

Das folgende GNU Octave-Skript erstellt das Berechnungsmodell und berechnet die Druckverteilung für die beiden Konfigurationen

1. $\alpha = 2°$ und $\eta = 0°$ sowie
2. $\alpha = 2°$ und $\eta = 2°$.

Die Modellbeschreibung, die Komponente, der Auftriebsbeiwert und die auf

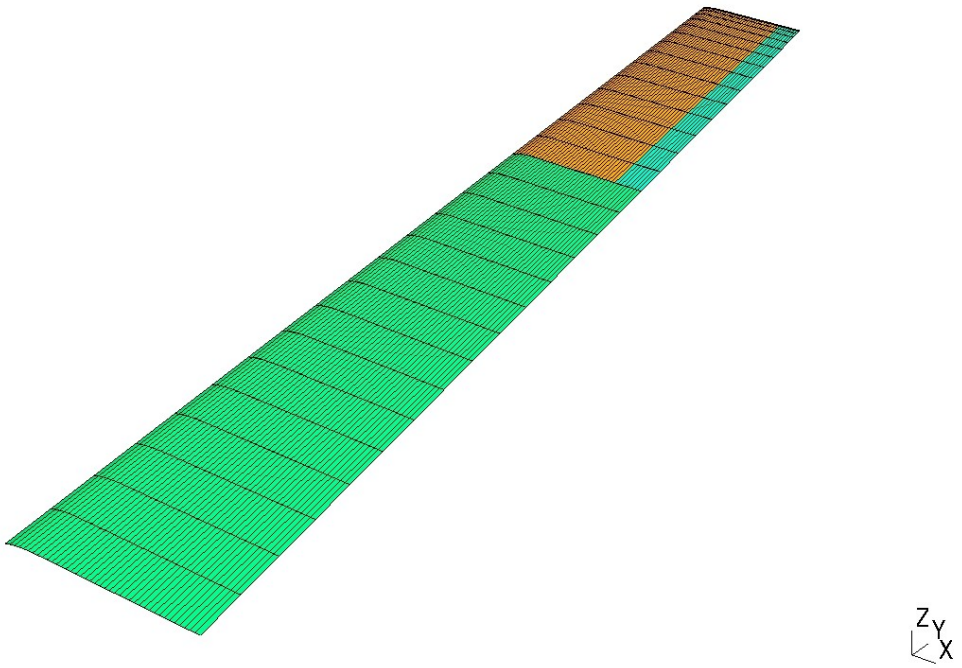


Abbildung 2.3: Aerodynamik-Modell

den Staudruck bezogene Änderung des Rollmoments infolge des Querruderausschlags werden in der Datei `aero.bin` gespeichert. Diese Datei wird für alle folgenden Teilaufgaben benötigt. Die Verläufe des Druckbeiwerts in den ausgewählten Flügelschnitten werden in der Datei `rigid.plt` gespeichert, um sie in Teilaufgabe e) mit den Verläufen für den elastischen Flügel vergleichen zu können.

```
# Übungsblatt 5.1, Aufgabe 2: Flügel
#
# b) Aerodynamik-Modell: Berechnung von Druckbeiwert,
#                        Auftriebsbeiwert und Änderung des
#                        Rollmoments
#
#    Erzeugte Dateien: aero.bin   Komponente mit Aerodynamikmodell
#                      rigid.bin  Daten für xy-Plot
#
# ------------------------------------------------------------------

  warning("off", "Octave:missing-glyph");

  colors = [1, 0, 0; 0, 1, 0; 0, 0, 1; 1, 0, 0.5];

  set(0, "defaultlinelinewidth", 2);
  set(0, "defaultaxesfontsize", 10);
  set(0, "defaultaxescolororder", colors);

  fid  = fopen("aero.res", "wt");

# Aerodynamik-Modell
# ------------------

# Daten definieren

  params

# Profildaten

  cmbi = mfs_airfoil("NACA", 5, 30);
  cmbo = mfs_airfoil("NACA", 5, 30, 0, 1 - kp);
  cmba = mfs_airfoil("NACA", 5, 30, 1 - kp, 1);

# Modelltyp

  aero.type    = "aero";
  aero.subtype =  "vlm";
  aero.symy    =      0;

# Punkte an der Flügelvorderkante

  sa = ya / yt;
  ca = (1 - sa) * cr + sa * ct;
  sw = tand(sweep);

  xr = -0.5 * cr;
```

```
  xa = ya * sw - 0.5 * ca;
  xt = yt * sw - 0.5 * ct;

  points(1).id = 1; points(1).coor = [xr,  0, 0];
  points(2).id = 2; points(2).coor = [xa, ya, 0];
  points(3).id = 3; points(3).coor = [xt, yt, 0];

# Punkte an der Querrudervorderkante

  xa += (1 - kp) * ca; xt += (1 - kp) * ct;

  points(4).id = 4; points(4).coor = [xa, ya, 0];
  points(5).id = 5; points(5).coor = [xt, yt, 0];

# Auftriebsfläche für den Innenflügel

  ls(1).id = 1; ls(1).points = [1, 2];
  ls(1).chord = [cr, ca]; ls(1).camber = cmbi;
  ls(1).nx = nxi; ls(1).typex = "linear";
  ls(1).ny = nyi;

# Auftriebsfläche für den Außenflügel

  ls(2).id = 2; ls(2).points = [2, 3];
  ls(2).chord = (1 - kp) * [ca, ct]; ls(2).camber = cmbo;
  ls(2).nx = nxo; ls(2).typex = "linear";
  ls(2).ny = nyo; ls(2).typey = "cos>";
  ls(2).alpha = [0, atip];

# Auftriebsfläche für das Querruder

  ls(3).id = 3; ls(3).points = [4, 5];
  ls(3).chord = kp * [ca, ct]; ls(3).camber = cmba;
  ls(3).nx = nxa; ls(3).typex = "linear";
  ls(3).ny = nyo; ls(3).typey = "cos>";
  ls(3).alpha = [0, atip];

# Querruder als Kontrollfläche

  controls(1).name = "aileron";
  controls(1).ls   = 3;

# Konfigurationen

  for k = 1 : 2
      config(k).name = ...
                sprintf("Conf. %d: alpha = %5.2f, eta = %5.2f",
                        k, alpha, eta(k));
      config(k).alpha = alpha; config(k).aileron = eta(k);
  end

# Definitionen zum Modell hinzufügen

  aero.points   = points;
  aero.ls       = ls;
```

```
  aero.controls = controls;
  aero.config   = config;

# Analyse
# -------

  winga = mfs_new(fid, aero);
  mfs_export("aero.msh", "msh", winga, "mesh", "camber");

  winga = mfs_statresp(winga);
  winga = mfs_results(winga, "statresp", "panel");

  mfs_export("aero.pos", "msh", winga, "statresp", "pressure");

# Auftriebsbeiwert

  [F, M] = mfs_getresp(winga, "statresp", "aeload");
  A      = mfs_getresp(winga, "mesh", "area");
  cLR    = F(3, 1) / A;
  dMxR   = M(1, 2) - M(1, 1);

  fprintf(fid, "Area = %11.4e\n", A);
  fprintf(fid, "cLR  = %11.4e, dMxR = %11.4e\n", cLR, dMxR);

  save -binary aero.bin aero winga cLR dMxR

# Druckbeiwert

  [xi(:, 1), pri(:, :, 1), y(1)] = ...
     mfs_xydata(winga, "statresp", "pressure", 1, ycoli(1));
  [xi(:, 2), pri(:, :, 2), y(2)] = ...
     mfs_xydata(winga, "statresp", "pressure", 1, ycoli(2));
  [xo(:, 1), pro(:, :, 1), y(3)] = ...
     mfs_xydata(winga, "statresp", "pressure", [2, 3], ycolo(1));
  [xo(:, 2), pro(:, :, 2), y(4)] = ...
     mfs_xydata(winga, "statresp", "pressure", [2, 3], ycolo(2));

  xi = xi / cr;
  xo = xo / cr;
  y  = y / 1000;

  save -binary rigid.plt xi xo y pri pro

  for k = 1 : length(y)
      leg{k} = sprintf("y = %4.2f m", y(k));
  end
  for k = 1 : length(eta)
      header{k} = sprintf("\\eta = %2.0f\\circ", eta(k));
  end
  figure(1, "position", [100, 500, 1000, 500],
            "paperposition", [0, 0, 16, 12]);

  subplot(1, 2, 1)
     plot(xi(:, 1), squeeze(pri(:, 1, 1)),
          xi(:, 2), squeeze(pri(:, 1, 2)),
```

```
          xo(:, 1), squeeze(pro(:, 1, 1)),
          xo(:, 2), squeeze(pro(:, 1, 2)));
    title(header{1}, "fontweight", "normal");
    axis([-1, 0.5, -0.5, 1]);
    grid;
    xlabel('x/c'); ylabel('\Delta c_P');

  subplot(1, 2, 2)
    plot(xi(:, 1), squeeze(pri(:, 2, 1)),
         xi(:, 2), squeeze(pri(:, 2, 2)),
         xo(:, 1), squeeze(pro(:, 2, 1)),
         xo(:, 2), squeeze(pro(:, 2, 2)));
    legend(leg, "location", "southeast");
    legend("boxoff"); legend("left");
    title(header{2}, "fontweight", "normal");
    axis([-1, 0.5, -0.5, 1]);
    grid;
    xlabel('x/c');

  print("rigid.jpg", "-djpg");

  fclose(fid);
```

Die Ausgabedatei enthält folgende Ergebnisse:

```
Mefisto 2.4: Building new component from input "aero"
```

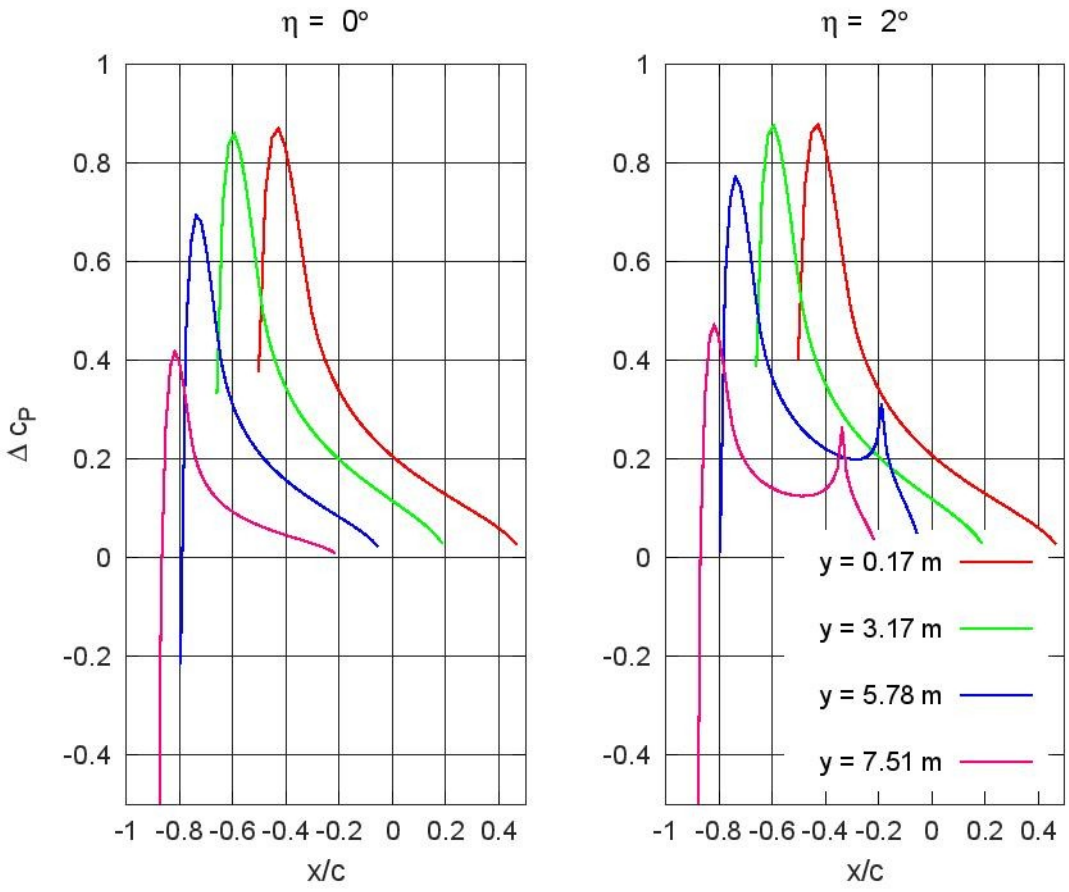


Abbildung 2.4: Druckbeiwerte für den starren Flügel

```
  Model Type = aero, Model Subtype = vlm

  Number of nodes  =  1328,  Number of panels =  1200
  Number of lifting surfaces =            3
  Number of control surfaces =            1
  Number of configurations   =            2
  Reference chord length     =  0.00000e+00
  Symmetry plane: y          =  0.00000e+00

Area =  8.0000e+06
cLR  =  2.5400e-01, dMxR =  1.4550e+09
```

Abbildung 2.4 zeigt die Druckbeiwerte an vier typischen Flügelschnitten ohne und mit Querruderausschlag. Wie zu erwarten nimmt der Druck zur Flügelspitze hin ab. Deutlich zu sehen ist der Druckanstieg an der Vorderkante des Querruders, wenn das Querruder ausgeschlagen ist.

c) Aeroelastik-Modell

Für die Kopplung der drei Auftriebsflächen mit dem Strukturmodell müssen drei Splines definiert werden. Alle drei Splines werden an alle Knoten des Strukturmodells angeschlossen, um eine stetige Verformung des Aerodynamik-Modells zu erreichen. Es werden zehn Spline-Segmente verwendet.

Der folgende Abschnitt des GNU Octave-Skripts definiert das Aeroelastik-Modell:

```
# Übungsblatt 5.1, Aufgabe 2: Flügel
#
# c) Aeroelastik-Modell: Definition der Splines, Übertragung der
#    aerodynamischen Lasten, Vergleich der Resultierenden und
#    Berechnung der Verschiebungen
#
#    Benötigte Dateien: solid.bin, aero.bin
#
#    Erzeugte Dateien:  aeroelastic.bin  Aeroelastisches Modell
#
# ------------------------------------------------------------

  fid  = fopen("aeroelastic.res", "wt");

# Aeroelastik-Modell
# ------------------

  params

  load solid.bin
  load aero.bin

  model.type  = "aeroelastic";

  model.solid = wings;
  model.aero  = winga;

# Spline für Innenflügel
```

```
  spline(1).id = 1; spline(1).type = "tb"; spline(1).lsid = 1;
  spline(1).data.nbreaks = nseg;

# Spline für Außenflügel

  spline(2).id = 2; spline(2).type = "tb"; spline(2).lsid = 2;
  spline(2).data.nbreaks = nseg;

# Spline für Querruder

  spline(3).id = 3; spline(3).type = "tb"; spline(3).lsid = 3;
  spline(3).data.nbreaks = nseg;

  model.splines = spline;

# Komponente und Splines erzeugen

  wing = mfs_new(fid, model);
  wing = mfs_splines(wing);

  save -binary aeroelastic.bin model wing
```

Modell und Komponente werden für die weiteren Teilaufgaben in der Datei `aeroelastic.bin` gespeichert.

Zur Kontrolle der Splines werden die aus den in Teilaufgabe b) berechneten Druckverteilungen ermittelten Lasten auf das Strukturmodell übertragen und die resultierenden Lasten verglichen.

```
# Aerodynamische Lasten aus u5_1_2b auf Struktur übertragen

  wings = mfs_transfer(wing, winga, "statresp", "loads");
  [FA, MA] = mfs_getresp(winga, "statresp", "aeload");
  [FS, MS] = mfs_getresp(wings, "load", "resultant");

  nc = columns(FA);

  fprintf(fid, "Load resultants of aerodynamic component: \n\n");
  for k = 1 : nc
      fprintf(fid, "  Configuration %2d:\n", k)
      fprintf(fid, "     F = [%10.3e, %10.3e, %10.3e] kN\n",
              FA(:, k) / 1000);
      fprintf(fid, "     M = [%10.3e, %10.3e, %10.3e] kNm\n",
              MA(:, k) * 1e-6);
  end

  fprintf(fid, "\nLoad resultants of solid component: \n\n");
  for k = 1 : nc
      fprintf(fid, "  Configuration %2d:\n", k)
      fprintf(fid, "     F = [%10.3e, %10.3e, %10.3e] kN\n",
              FS(:, k) / 1000);
      fprintf(fid, "     M = [%10.3e, %10.3e, %10.3e] kNm\n",
              MS(:, k) * 1e-6);
```

```
  end
```

Anschließend werden die Verschiebungen der Strukturknoten infolge der aerodynamischen Lasten berechnet und auf die Knoten des Aerodynamik-Modells übertragen. Zum besseren Vergleich werden Strukturmodell und Aerodynamik-Modell zu einem Gesamtmodell kombiniert.

```
# Verschiebungen

  wings = mfs_statresp(wings);
  winga = mfs_transfer(wing, wings, "statresp", "disp");

  mfs_print(fid, wings, "statresp", "reac");

  mfs_export("solid.lds", "msh", wings, "load", "point");
  mfs_export("solid.dsp", "msh", wings, "statresp", "disp");
  mfs_export("aero.dsp", "msh", winga, "statresp", "disp");

# Modelle und Ergebnisse kombinieren

  mfs_merge("solid.msh", "aero.msh", "wing.msh", "msh");
  mfs_merge("solid.dsp", "aero.dsp", "wing.dsp", "msh");

  fclose(fid);
```

Die Ausgabedatei enthält folgende Ergebnisse:

```
Mefisto 2.4: Building new component from input "model"

  Model Type = aeroelastic

  Number of splines      =            3

Load resultants of aerodynamic component:

  Configuration  1:
     F = [ 0.000e+00,  0.000e+00,  2.032e+03] kN
     M = [ 6.442e+03,  9.944e+02,  0.000e+00] kNm
  Configuration  2:
     F = [ 0.000e+00,  0.000e+00,  2.283e+03] kN
     M = [ 7.897e+03,  1.130e+03,  0.000e+00] kNm

Load resultants of solid component:

  Configuration  1:
     F = [ 0.000e+00,  0.000e+00,  2.032e+03] kN
     M = [ 6.442e+03,  9.944e+02,  0.000e+00] kNm
  Configuration  2:
     F = [ 0.000e+00,  0.000e+00,  2.283e+03] kN
     M = [ 7.897e+03,  1.130e+03,  0.000e+00] kNm

--------------------------------------------------------------------------------

Component "wings"

Reaction loads of loadcase  1:

   node          Fx          Fy          Fz          Mx          My          Mz

      3   -2.157e+06   2.998e+07  -7.281e+05   1.070e+08   9.265e+06  -1.242e+07
      4    2.252e+06  -2.998e+07  -6.722e+05   1.065e+08   9.419e+06   1.445e+07
      5   -2.836e+06   2.102e+07  -3.243e+05   4.915e+07   5.640e+06   5.527e+06
      6    2.741e+06  -2.103e+07  -3.074e+05   4.920e+07   5.549e+06  -4.744e+06
```

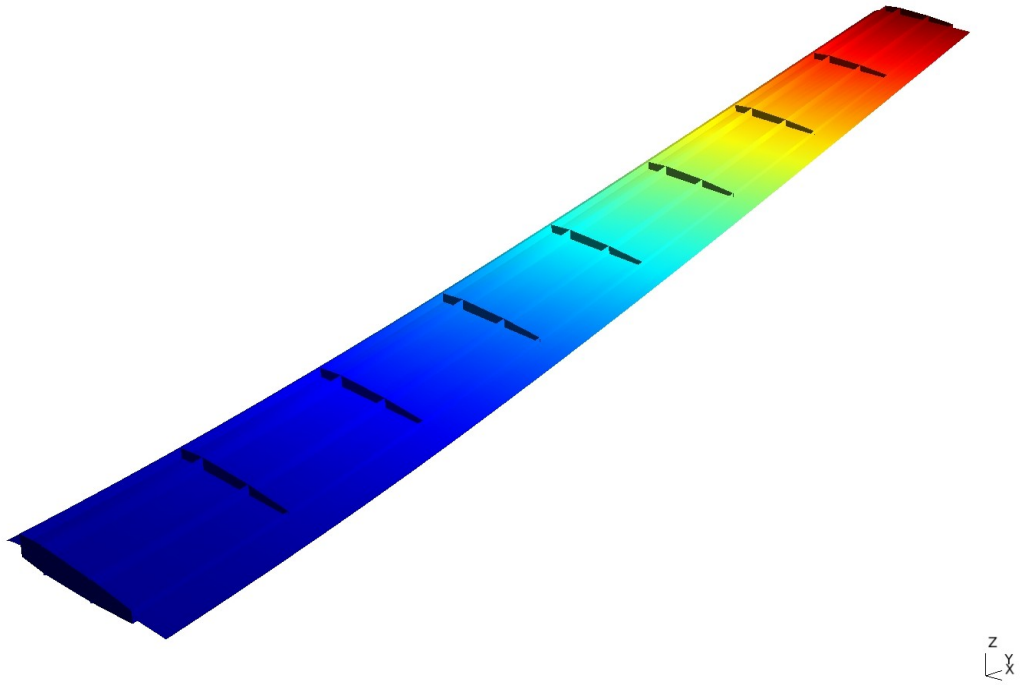


Abbildung 2.5: Vergleich der Verschiebungen

```
Reaction loads of loadcase  2:

   node          Fx          Fy          Fz          Mx          My          Mz

      3   -2.442e+06   3.669e+07  -8.041e+05   1.318e+08   1.132e+07  -1.560e+07
      4    2.559e+06  -3.668e+07  -7.330e+05   1.312e+08   1.150e+07   1.805e+07
      5   -3.409e+06   2.587e+07  -3.846e+05   6.077e+07   6.929e+06   6.657e+06
      6    3.292e+06  -2.588e+07  -3.611e+05   6.082e+07   6.817e+06  -5.698e+06
```

Die resultierenden Lasten stimmen exakt überein.

Die Verschiebungen sind in Abbildung 2.5 dargestellt. Um die Verschiebungen des Aerodynamik-Modells besser sichtbar zu machen, ist die Außenhaut des Strukturmodells ausgeblendet. Es kann festgestellt werden, dass die Verschiebungen sehr gut zusammen passen.

Da diese Verschiebungen zu einem Staudruck von $1\ \mathrm{MPa}$ gehören, sind ihre Absolutwerte sehr groß.

d) Aeroelastische Analysen

Zunächst wird der Staudruck berechnet, bei dem statische Divergenz auftritt. Diese Berechnung kann direkt mit dem in Teilaufgabe c) erstellten Aeroelastik-Modell durchgeführt werden.

```
# Übungsblatt 5.1, Aufgabe 2: Flügel
#
# d) Aeroelastische Analysen:
#    Statische Divergenz, Auftriebsbeiwert und Ruderwirkungsfaktor
```

```
#    in Abhängigkeit vom Staudruck
#
#    Benötigte Dateien: aero.bin, aeroelastic.bin
#
# ------------------------------------------------------------------

  warning("off", "Octave:missing-glyph");

  set(0, "defaultlinelinewidth", 2);
  set(0, "defaultaxesfontsize", 10);

  fid  = fopen("flexible1.res", "wt");

  params

# Statische Divergenz
# -------------------

  load aeroelastic.bin

  [wings, winga, nfound] = mfs_diverg(wing, 5);

  if (nfound)
     mfs_print(fid, winga, "diverg", "qdyn");
     qd = mfs_getresp(winga, "diverg", "qdyn", 1);
  else
     fclose(fid);
     return;
  end
```

Die Ausgabedatei enthält folgende Werte für den Staudruck bei Divergenz:

```
Component "winga"

Dynamic pressure at divergence:

  No.    Dynamic pressure
  ----------------------
   1        1.2865e-02
   2        1.6691e-01
   3        5.7601e-01
   4        1.1889e+00
   5        1.9068e+00
```

Der erste Wert in der Liste ist der Staudruck, bei dem Divergenz auftritt, d. h. $q_D = 1{,}29 \cdot 10^{-2}$ MPa. Bei einer Luftdichte von $1{,}21$ kg/m^3 entspricht dieser Staudruck einer Geschwindigkeit von etwa 146 m/s. Diese Geschwindigkeit beträgt etwa 43 % der Schallgeschwindigkeit, womit der Gültigkeitsbereich der inkompressiblen Theorie bereits überschritten ist.

Anschließend werden neue Konfigurationen definiert, um Auftriebsbeiwert und Ruderwirkungsfaktor in Abhängigkeit vom Staudruck zu berechnen:

```
# Auftriebsbeiwert und Ruderwirkungsfaktor
# ----------------------------------------

  load aero.bin
```

```
# Neue Konfigurationen definieren

  qr   = [0.05 : 0.05 : 0.5];
  nq   = length(qr);
  qdyn = qr * qd;

  nc = 1;

  for k = 1 : 2
      for l = 1 : nq
          name = sprintf("Conf. %d: qdyn = %10.4e, eta = %5.2f",
                         nc, qdyn(l), eta(k));
          config(nc).name = name;
          config(nc).qdyn = qdyn(l);
          config(nc).alpha = alpha;
          config(nc).aileron = eta(k);
          nc++;
      end
  end

  aero.config = config;
```

Die Berechnung von Auftriebsbeiwert und Ruderwirkungsfaktor erfolgt genauso wie im Falle einer ungekoppelten aerodynamischen Berechnung. Der Auftriebsbeiwert wird auf den Auftriebsbeiwert des starren Flügels bezogen.

```
# Neue Aerodynamik-Komponente

  winga = mfs_new(fid, aero);

# Neue Aeroelastik-Komponente

  model.aero = winga;

  wing = mfs_new(fid, model);

# Rechnung

  wing = mfs_splines(wing);

  [wings, winga] = mfs_statresp(wing);
  winga = mfs_results(winga, "statresp", "panel");

# Auftriebsbeiwert

  A      = mfs_getresp(winga, "mesh", "area");
  [F, M] = mfs_getresp(winga, "statresp", "aeload");

  fprintf(fid,
          "     qdyn          Fx           Fy           Fz      ");
  fprintf(fid, "    Mx           My           Mz\n");
  for k = 1 : length(qdyn)
      fprintf(fid, "  %10.4e  %10.4e  %10.4e  %10.4e",
              qdyn(k), F(:, k));
```

```
      fprintf(fid, "  %10.4e  %10.4e  %10.4e\n", M(:, k));
  end

  cLF = F(3, 1 : nq) ./ (A * qdyn);
  cLr = cLF / cLR;

# Ruderwirkungsfaktor

  dMxF = M(1, nq + 1 : end) - M(1, 1 : nq);
  dMxF = dMxF ./ qdyn;
  etaR = dMxF / dMxR;
```

In der Ausgabedatei finden sich dazu die folgenden Ergebnisse für die Lasten ohne Querruderausschlag:

qdyn	Fx	Fy	Fz	Mx	My	Mz
6.4327e-04	0.0000e+00	0.0000e+00	1.3436e+03	4.2982e+06	6.6227e+05	0.0000e+00
1.2865e-03	0.0000e+00	0.0000e+00	2.7678e+03	8.9392e+06	1.3747e+06	0.0000e+00
1.9298e-03	0.0000e+00	0.0000e+00	4.2865e+03	1.3983e+07	2.1459e+06	0.0000e+00
2.5731e-03	0.0000e+00	0.0000e+00	5.9167e+03	1.9505e+07	2.9868e+06	0.0000e+00
3.2164e-03	0.0000e+00	0.0000e+00	7.6801e+03	2.5599e+07	3.9110e+06	0.0000e+00
3.8596e-03	0.0000e+00	0.0000e+00	9.6048e+03	3.2389e+07	4.9362e+06	0.0000e+00
4.5029e-03	0.0000e+00	0.0000e+00	1.1727e+04	4.0033e+07	6.0854e+06	0.0000e+00
5.1462e-03	0.0000e+00	0.0000e+00	1.4096e+04	4.8744e+07	7.3893e+06	0.0000e+00
5.7894e-03	0.0000e+00	0.0000e+00	1.6777e+04	5.8813e+07	8.8898e+06	0.0000e+00
6.4327e-03	0.0000e+00	0.0000e+00	1.9864e+04	7.0646e+07	1.0645e+07	0.0000e+00

Da der Flügel exakt in der *xy*-Ebene liegt, sind die Kräfte in *x*- und *y*-Richtung sowie das Moment um die z-Achse exakt null.

In Kapitel 1.2 wurde anhand eines einfachen Modells für den Ruderwirkungsfaktor die analytische Beziehung

$$\eta_R = \frac{1 - q_\infty / q_R}{1 - q_\infty / q_D}$$

gefunden. Diese Gleichung lässt sich umformen zu

$$\eta_R = \frac{1 - (q_D / q_R) q_\infty / q_D}{1 - q_\infty / q_D} .$$

In dieser Form wird deutlich, dass die Beziehung nur vom Verhältnis q_D/q_R abhängt. Für $q_R > q_D$ strebt der Ruderwirkungsfaktor bei Annäherung an q_D gegen unendlich. Es tritt keine Ruderumkehr auf. Für $q_R < q_D$ strebt der Ruderwirkungsfaktor gegen minus unendlich. Er wird null für $q_\infty = q_R$.

Das Verhältnis q_D/q_R lässt sich durch Anpassung der Gleichung an die berechneten Werte ermitteln. Die Gleichung hat die Form

$$y = \frac{1 - m x}{1 - x}$$

mit $y = \eta_R$, $x = q_\infty / q_D$ und $m = q_D / q_R$. Der Parameter m wird so gewählt, dass die Summe der Quadrate der Abweichung der berechneten Werte von der gegebenen Gleichung minimal wird:

$$\frac{d}{dm}\sum_n \left(\frac{1-m\,x_n}{1-x_n} - y_n \right)^2 = 0$$

Ausführen der Ableitung ergibt

$$\sum_n \left(\frac{1-m\,x_n}{1-x_n} - y_n \right) \frac{x_n}{1-x_n} = 0\,.$$

Daraus folgt

$$\sum_n \left(\frac{1}{1-x_n} - y_n \right) \frac{x_n}{1-x_n} = m \sum_n \left(\frac{x_n}{1-x_n} \right)^2,$$

woraus m bestimmt werden kann. Das Skript berechnet den Kehrwert

$$\frac{1}{m} = \frac{q_R}{q_D},$$

der das Verhältnis des Staudrucks bei Ruderumkehr zum Staudruck bei Divergenz angibt.

```
# Staudruck bei Ruderumkehr

  a = 1 ./ (1 - qr);
  b = qr .* a;
  a = (a - etaR) .* b;
  c = b.^2;

  qRqD = sum(c) / sum(a);

  fprintf(fid, "\n  qR/qD = %6.4f\n", qRqD);

# Graphische Darstellung von Auftriebsbeiwert und
# Ruderwirkungsfaktor

  q  = 0 : 0.01 : qr(end);
  eR = (1 - q / qRqD) ./ (1 - q);

  qt = 0 : 0.1 : qr(end);

  figure(1, "position", [100, 500, 1000, 500],
            "paperposition", [0, 0, 14, 10]);

  subplot(1, 2, 1);
     plot(qr, cLr, "color", "red");
     grid;
     set(gca(), "xtick", qt);
     xlabel('q_{/Symbol \245} / q_D');
     ylabel("c_L/c_{LR}");

  subplot(1, 2, 2);
     plot(qr, etaR, "color", "red", "marker", "x",
```

```
                  "linestyle", "none",
        q, eR, "color", "green");
    if (qRqD > 1)
       legend("num.", "anal.", "location", "northwest");
       else
    legend("num.", "anal.", "location", "southwest");
    end
    legend("boxoff"); legend("left");
    grid;
    set(gca(), "xtick", qt);
    xlabel('q_{/Symbol \245} / q_D');
    ylabel('\eta_R');

print("Beiwerte.jpg", "-djpg");

fclose(fid);
```

Beim vorwärts gepfeilten Flügel gilt $q_R > q_D$. Die Rechnung liefert einen Wert von $q_R/q_D = 1{,}36$. Abbildung 2.6 zeigt eine sehr gute Übereinstimmung zwischen dem analytisch und dem numerisch berechneten Ruderwirkungsfaktor. In der Abbildung ebenfalls dargestellt ist der Verlauf des auf den Auftriebsbeiwert des starren Flügels bezogenen Auftriebsbeiwerts.

e) Detaillierte Auswertung bei 250 km/h

Das folgende GNU Octave-Skript berechnet die Druckverteilung, die Verschiebungen und die Spannungen bei einer Anströmgeschwindigkeit von 250 km/h für die folgenden Konfigurationen:

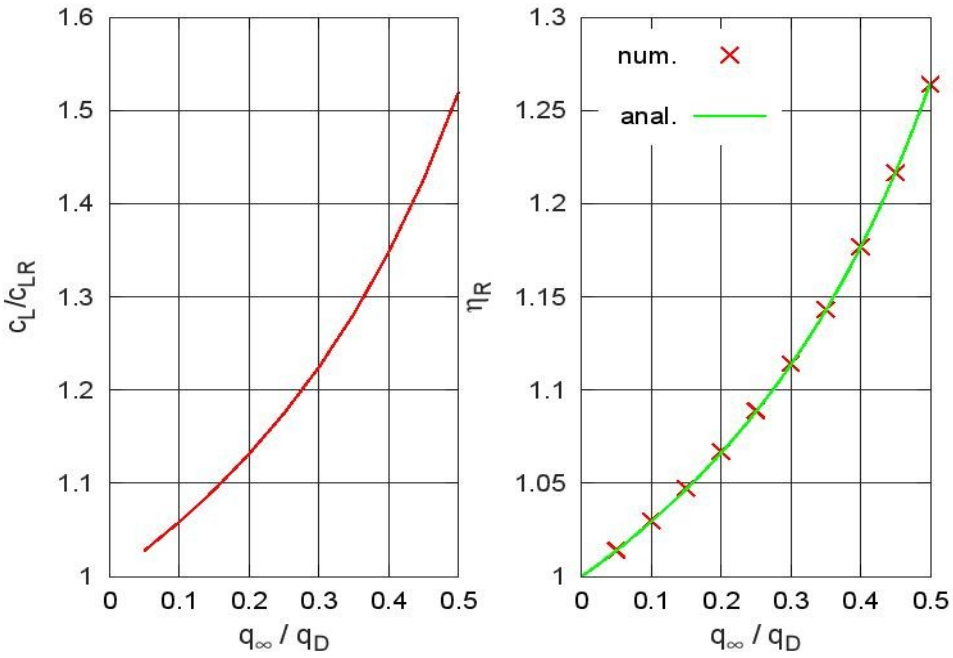


Abbildung 2.6: Auftriebsbeiwert und Ruderwirkungsfaktor

1. Anstellwinkel 2°, kein Querruderausschalg
2. Anstellwinkel 2°, Querruderausschlag 2°

Die Druckverteilung wird als Druckbeiwert an vier ausgewählten Flügelschnitten dargestellt und mit den Ergebnissen für den starren Flügel verglichen.

```
# Übungsblatt 5.1, Aufgabe 2: Flügel
#
# e) Detaillierte Auswertung für 250 km/h: Druckverteilung,
#    Verschiebung, Spannungen
#
#    Benötigte Dateien: aero.bin, aeroelastic.bin, solid.set,
#                       rigid.plt
#
# ------------------------------------------------------------

  warning("off", "Octave:missing-glyph");

  set(0, "defaultlinelinewidth", 2);
  set(0, "defaultaxesfontsize", 10);

  fid  = fopen("flexible2.res", "wt");

# Daten

  params

  v   =      250;    % Geschwindigkeit in km/h
  rho = 1.21e-12;    % Luftdichte in t/mm^3

# Neue Konfigurationen definieren

  load aero.bin

  vmms = 1000 * v / 3.6;
  qdyn = 0.5 * rho * vmms^2;

  for k = 1 : 2
      name = sprintf("Conf. %d: qdyn = %10.4e, eta = %5.2f",
                     k, qdyn, eta(k));
      config(k).name = name;
      config(k).qdyn = qdyn;
      config(k).alpha = alpha;
      config(k).aileron = eta(k);
  end

  aero.config = config;

# Neue Aerodynamik-Komponente

  winga = mfs_new(fid, aero);

# Neue Aeroelastik-Komponente

  load aeroelastic.bin
```

```
  model.aero = winga;

  wing = mfs_new(fid, model);

# Rechnung

  wing = mfs_splines(wing);

  [wings, winga] = mfs_statresp(wing);
  winga = mfs_results(winga, "statresp", "panel");
  wings = mfs_results(wings, "statresp", "element");

  mfs_export("aero.pos", "msh", winga, "statresp",
             "pressure", "disp");
  mfs_export("solid.pos", "msh", wings, "statresp",
             "disp", "stress", "resultant");

  mfs_merge("solid.pos", "aero.pos", "flexible.dsp", "msh");

# Druckverläufe

  load rigid.plt

  [xif(:, 1), pfi(:, :, 1)] = ...
     mfs_xydata(winga, "statresp", "pressure", 1, ycoli(1));
  [xif(:, 2), pfi(:, :, 2)] = ...
     mfs_xydata(winga, "statresp", "pressure", 1, ycoli(2));
  [xof(:, 1), pfo(:, :, 1)] = ...
     mfs_xydata(winga, "statresp", "pressure", [2, 3], ycolo(1));
  [xof(:, 2), pfo(:, :, 2)] = ...
     mfs_xydata(winga, "statresp", "pressure", [2, 3], ycolo(2));

  pfi = pfi / qdyn;
  pfo = pfo / qdyn;

  for k = 1 : length(y)
      header{k} = sprintf("\\eta = %1d\\circ, y = %4.2f m",
                          eta(1), y(k));
  end

  figure(1, "position", [100, 500, 750, 500],
            "paperposition", [0, 0, 14, 10]);

  subplot(2, 2, 1)
     plot(xi(:, 1), squeeze(pri(:, 1, 1)), "color", "red",
          xi(:, 1), squeeze(pfi(:, 1, 1)), "color", "green");
     legend("rigid", "flexible", "location", "southeast");
     legend("boxoff"); legend("left");
     title(header{1});
     grid;
     ylim([-1, 1]);
     ylabel('\Delta c_P');

  subplot(2, 2, 2)
```

```
    plot(xi(:, 2), squeeze(pri(:, 1, 2)), "color", "red",
         xi(:, 2), squeeze(pfi(:, 1, 2)), "color", "green");
    legend("rigid", "flexible", "location", "southeast");
    legend("boxoff"); legend("left");
    title(header{2});
    grid;
    ylim([-1, 1]);

  subplot(2, 2, 3)
    plot(xo(:, 1), squeeze(pro(:, 1, 1)), "color", "red",
         xo(:, 1), squeeze(pfo(:, 1, 1)), "color", "green");
    legend("rigid", "flexible", "location", "southeast");
    legend("boxoff"); legend("left");
    title(header{3});
    grid;
    ylim([-1, 1]);
    ylabel('\Delta c_P'); xlabel('x/c');

  subplot(2, 2, 4)
    plot(xo(:, 2), squeeze(pro(:, 1, 2)), "color", "red",
         xo(:, 2), squeeze(pfo(:, 1, 2)), "color", "green");
    legend("rigid", "flexible", "location", "southeast");
    legend("boxoff"); legend("left");
    title(header{4});
    grid;
    ylim([-1, 1]);
    xlabel('x/c');

  print("druck1.jpg", "-djpg");

  for k = 1 : length(y)
      header{k} = sprintf("\\eta = %1d\\circ, y = %4.2f m",
                          eta(2), y(k));
  end

  figure(2, "position", [200, 600, 750, 500],
            "paperposition", [0, 0, 14, 10]);

  subplot(2, 2, 1)
    plot(xi(:, 1), squeeze(pri(:, 2, 1)), "color", "red",
         xi(:, 1), squeeze(pfi(:, 2, 1)), "color", "green");
    legend("rigid", "flexible", "location", "southeast");
    legend("boxoff"); legend("left");
    title(header{1});
    grid;
    ylim([-1, 1]);
    ylabel('\Delta c_P');

  subplot(2, 2, 2)
    plot(xi(:, 2), squeeze(pri(:, 2, 2)), "color", "red",
         xi(:, 2), squeeze(pfi(:, 2, 2)), "color", "green");
    legend("rigid", "flexible", "location", "southeast");
    legend("boxoff"); legend("left");
    title(header{2});
    grid;
```

```
  ylim([-1, 1]);

     subplot(2, 2, 3)
     plot(xo(:, 1), squeeze(pro(:, 2, 1)), "color", "red",
          xo(:, 1), squeeze(pfo(:, 2, 1)), "color", "green");
     legend("rigid", "flexible", "location", "southeast");
     legend("boxoff"); legend("left");
     title(header{3});
     grid;
     ylim([-1, 1]);
     ylabel('\Delta c_P'); xlabel('x/c');

  subplot(2, 2, 4)
     plot(xo(:, 2), squeeze(pro(:, 2, 2)), "color", "red",
          xo(:, 2), squeeze(pfo(:, 2, 2)), "color", "green");
     legend("rigid", "flexible", "location", "southeast");
     legend("boxoff"); legend("left");
     title(header{4});
     grid;
     ylim([-1, 1]);
     xlabel('x/c');

  print("druck2.jpg", "-djpg");

# Spannungen in den Balken

  load solid.set

  x1 = 0.5 * b_Stringer_2;
```

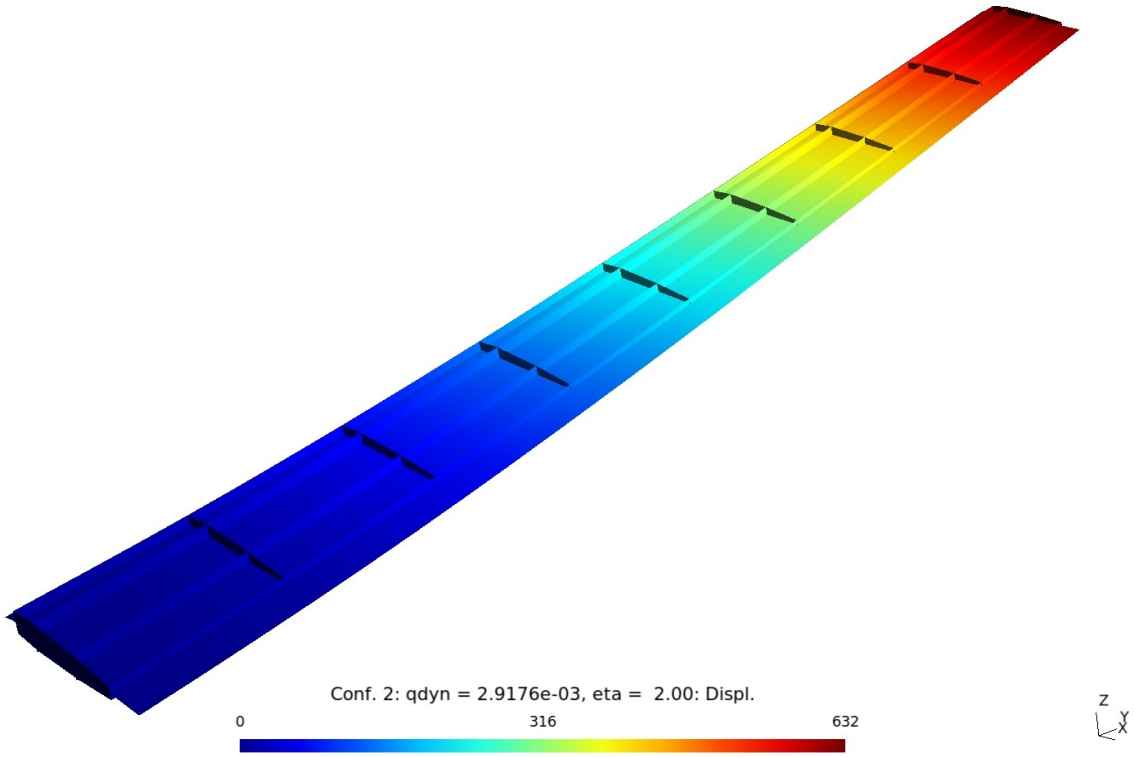


Abbildung 2.7: Verformung bei Querruderausschlag

```
zo = -0.5 * h_Stringer_2^2 / (b_Stringer_2 + h_Stringer_2);
zo -= 0.5 * t_Stringer_2;
zu = h_Stringer_2 + zo;
coor = [x1, zo; -x1, zo; 0, zu];
mfs_beamstress(fid, wings, "statresp",
           [eset.Stringer_2_Upper(1), eset.Stringer_2_Lower(1)],
           coor, 2);

x1 = 0.5 * b_Stringer_3;
zo = -0.5 * h_Stringer_3^2 / (b_Stringer_3 + h_Stringer_3);
zo -= 0.5 * t_Stringer_3;
zu = h_Stringer_3 + zo;
coor = [x1, zo; -x1, zo; 0, zu];
mfs_beamstress(fid, wings, "statresp",
           [eset.Stringer_3_Upper(1), eset.Stringer_3_Lower(1)],
           coor, 2);

fclose(fid);
```

Der verformte Flügel bei Querruderausschlag ist in Abbildung 2.7 dargestellt. An der Flügelspitze tritt eine maximale Auslenkung von $0{,}63\ \mathrm{m}$ auf, was bei einem Flügel mit einer Länge von $8\ \mathrm{m}$ realistisch ist.

Die größte Vergleichsspannung in den Schubfeldern tritt bei Querruderausschlag auf. Sie ist in Abbildung 2.8 dargestellt. Die größte Spannung hat einen Wert von $175\ \mathrm{MPa}$ und tritt an der Einspannung auf.

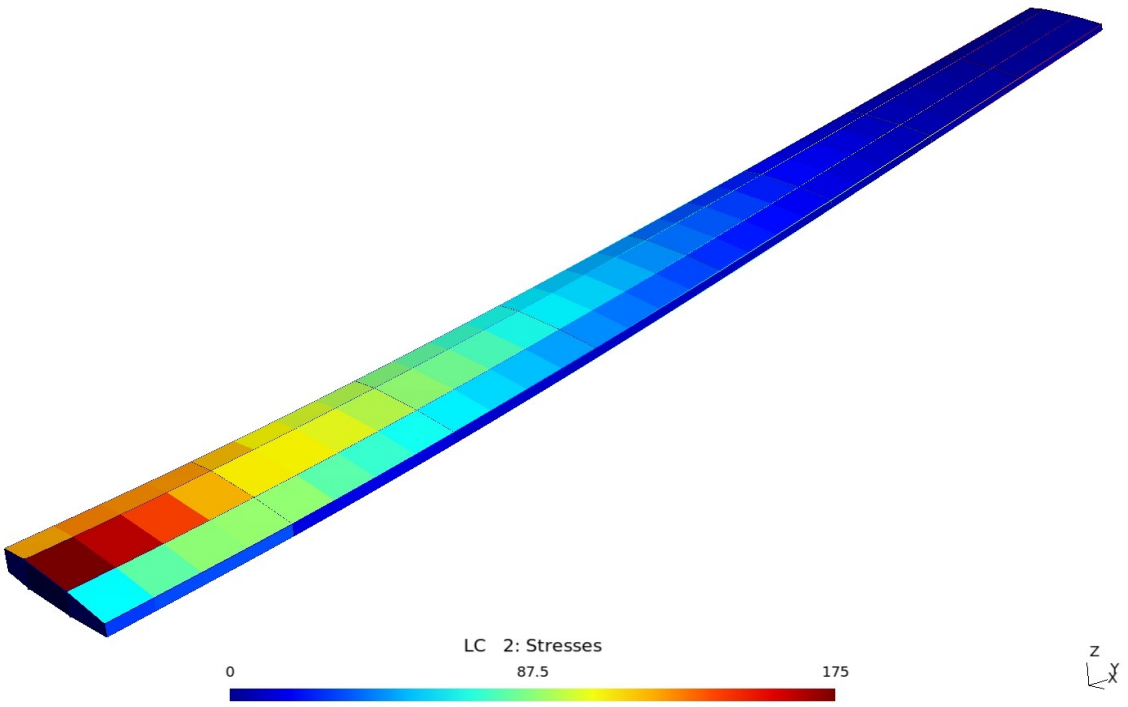


Abbildung 2.8: Vergleichsspannung bei Querruderausschlag

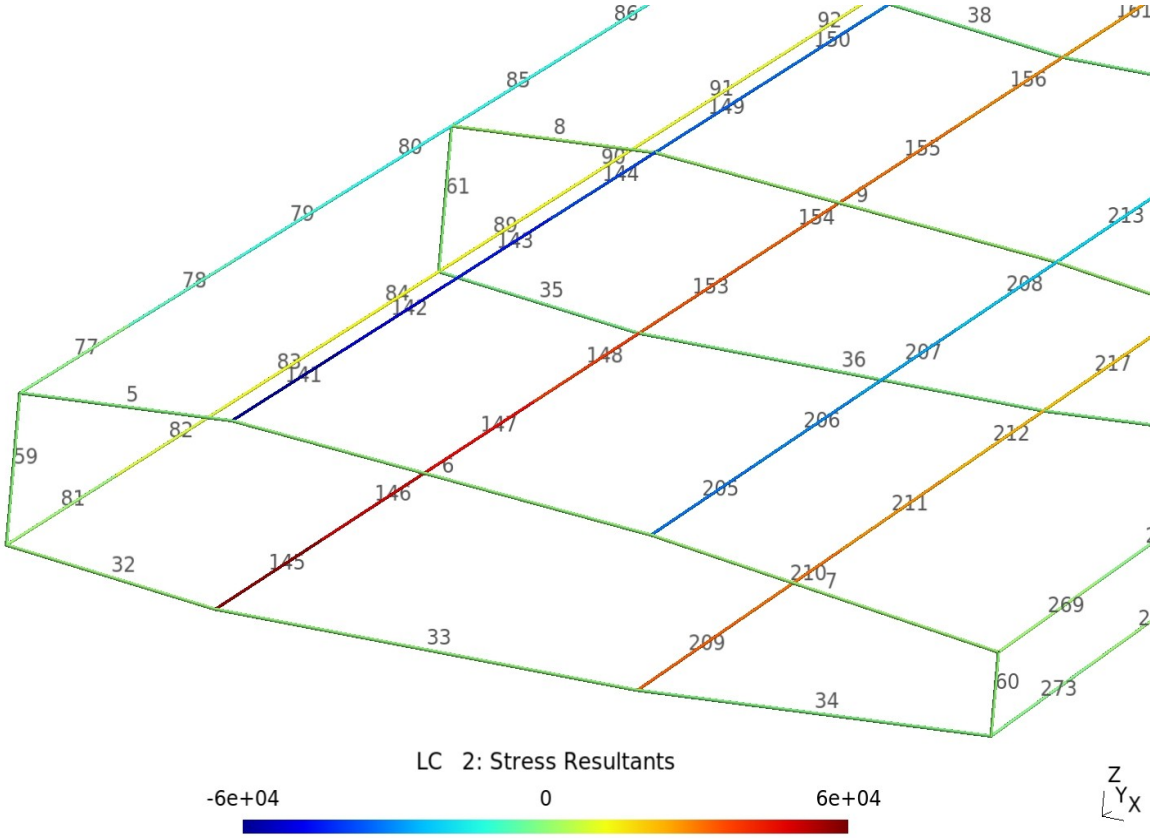


Abbildung 2.9: Normalkräfte in den Balken bei Querruderausschlag

Abbildung 2.9 zeigt die Normalkräfte in den Balkenelementen sowie die Nummern der Balkenelemente. Die größten Kräfte treten in den Stringern der beiden Hauptholme auf. Für die Elemente mit den größten Normalkräften, d. h. für die Elemente 141 und 145 sowie 205 und 209, werden die Spannungen an den drei Eckpunkten des T-Profils berechnet. Die Ausgabedatei enthält die folgenden Werte:

```
Component "wings"

Beam stresses of loadcase 2

  element  point      y           z          sig
  ------------------------------------------------
      141      1   2.500e+01  -1.089e+01  -2.028e+02
               2  -2.500e+01  -1.089e+01  -2.086e+02
               3   0.000e+00   2.911e+01  -7.809e+01
      145      1   2.500e+01  -1.089e+01   2.114e+02
               2  -2.500e+01  -1.089e+01   1.987e+02
               3   0.000e+00   2.911e+01   7.702e+01


--------------------------------------------------------------------------------

Component "wings"

Beam stresses of loadcase 2

  element  point      y           z          sig
  ------------------------------------------------
      205      1   2.000e+01  -8.429e+00  -1.563e+02
               2  -2.000e+01  -8.429e+00  -1.489e+02
```

```
          3    0.000e+00    2.157e+01   -6.902e+01
  209     1    2.000e+01   -8.429e+00    1.515e+02
          2   -2.000e+01   -8.429e+00    1.540e+02
          3    0.000e+00    2.157e+01    6.920e+01
```

Die Spannungen werden für beide Knotenpunkte der Balkenelemente berechnet. Die größte Spannung beträgt 212 MPa. Die Tatsache, dass sich für die einzelnen Auswertepunkte unterschiedliche Spannungen ergeben, zeigt, dass die Balken auch auf Biegung beansprucht werden.

Die Abbildungen 2.10 und 2.11 vergleichen die Verläufe der Druckbeiwerte in Flügeltiefenrichtung in ausgewählten Flügelschnitten beim flexiblen Flügel mit den entsprechenden Verläufen beim starren Flügel. Deutlich zu erkennen ist, dass der Druckbeiwert beim flexiblen Flügel größer ist als beim starren Flügel, wobei die Abweichung mit zunehmendem Abstand von der Flügelwurzel zunimmt. Ursache dafür ist der infolge der Torsion erhöhte Anstellwinkel.

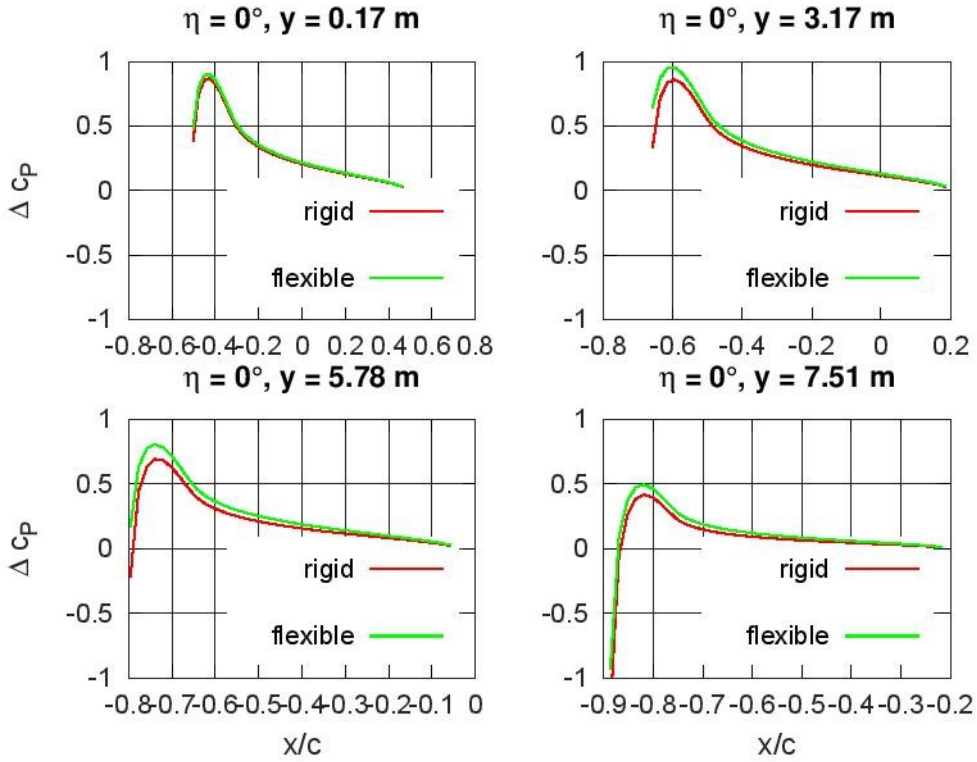


Abbildung 2.10: Druckbeiwerte für den flexiblen Flügel ohne Querruderausschlag

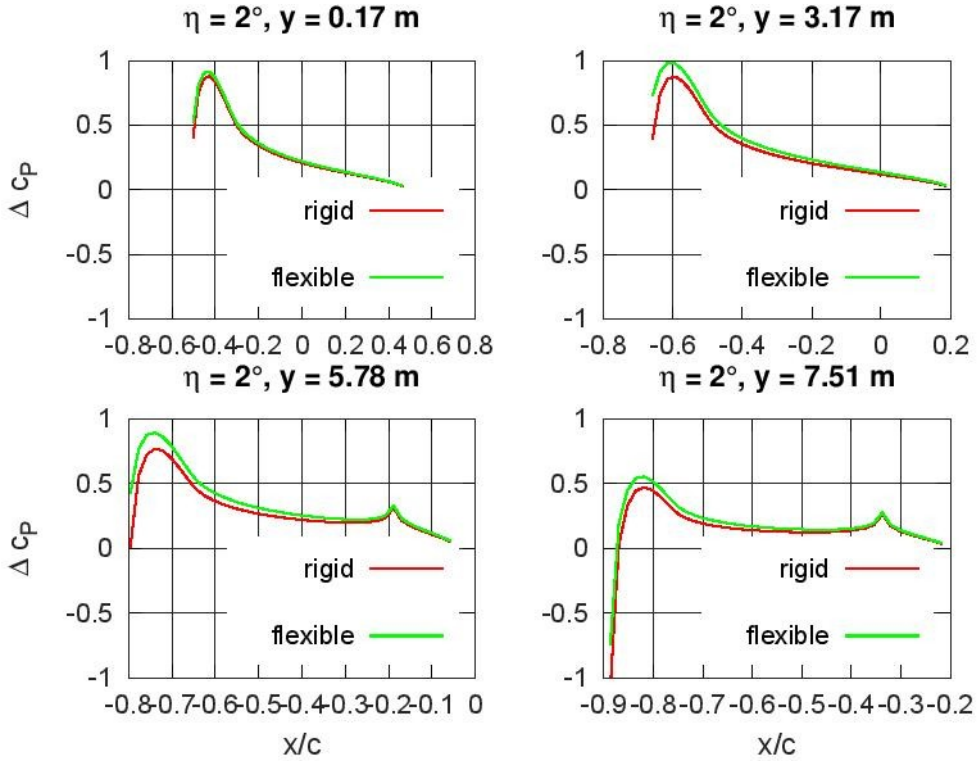


Abbildung 2.11: Druckbeiwerte für den flexiblen Flügel mit Querruderausschlag