1.3 Biege-Torsionsflattern

Lösungen

Aufgabe 1

Die Theodorsen-Funktion ist definiert durch

$$C(k)=F(k)+iG(k)=\frac{H_1^{(2)}(k)}{H_1^{(2)}(k)+iH_0^{(2)}(k)}$$

mit den Hankelfunktionen 2. Art der Ordnungen 0 und 1. Für $k=0$ sind Zähler und Nenner null. Der Wert eins ergibt sich als Grenzwert. Dieser Fall muss daher gesondert betrachtet werden.

GNU Octave Code

```
function [C, F, G] = theodorsen(k)

# usage: [C, F, G] = theodorsen(k)
#
# Input  k(:)  Reduced frequencies
#
# Output C(:)  Theodorsen function
#        F(:)  Real part of Theodorsen function
#              (optional)
#        G(:)  Imaginary part of Theodorsen function
#              (optional)
#
# The function computes the Theodorsen function
#
#                   H_1^2(k)
#    C(k) = ---------------------
#           H_1^2(k) + i H_0^2(k)
#
# ------------------------------------------------

# Check arguments

  if (nargin != 1 || nargout < 1 || nargout > 3)
     print_usage();
  end

  if (! isreal(k))
     error("Reduced frequencies must be real\n");
  end

  if (! isempty(find(k < 0)))
```

```
      error("Reduced frequencies must be positive\n");
   end

# Compute function

   C = ones(1, length(k));

   ix = find(k > 0);

   H12 = besselh(1, 2, k(ix));
   H02 = besselh(0, 2, k(ix));

   C(ix) = H12 ./ (H12 + i * H02);

   if (nargout > 1)
      F = real(C);
      G = imag(C);
   end

end
```

Mit diesem Code ergeben sich folgende Werte der Theodorsen-Funktion für die in Fung, An Introduction to the Theory of Aeroelasticity auf Seite 214 angegebenen reduzierten Frequenzen:

```
    k       F       G
 ----------------------
 10.00  0.5006  -0.0124
  6.00  0.5017  -0.0206
  4.00  0.5037  -0.0305
  3.00  0.5063  -0.0400
  2.00  0.5130  -0.0577
  1.50  0.5210  -0.0736
  1.20  0.5300  -0.0877
  1.00  0.5394  -0.1003
  0.80  0.5541  -0.1165
  0.66  0.5699  -0.1308
  0.60  0.5788  -0.1378
  0.56  0.5857  -0.1428
  0.50  0.5979  -0.1507
  0.44  0.6130  -0.1592
  0.40  0.6250  -0.1650
  0.34  0.6469  -0.1738
  0.30  0.6650  -0.1793
  0.24  0.6989  -0.1862
  0.20  0.7276  -0.1886
  0.16  0.7628  -0.1876
  0.12  0.8063  -0.1801
  0.10  0.8319  -0.1723
  0.08  0.8604  -0.1604
  0.06  0.8920  -0.1426
  0.05  0.9090  -0.1306
  0.04  0.9267  -0.1160
  0.25  0.6926  -0.1852
  0.01  0.9824  -0.0457
```

```
  0.00  1.0000   0.0000
```

Die Werte der Theodorsen-Funktion stimmen bis auf die Werte für die reduzierten Frequenzen 0,01 und 0,025 mit den von Fung angegebenen Werten überein.

Aufgabe 2

Für die aerodynamische Matrix gilt

$$\left[A(k)\right]_N = \pi S k^2 \begin{bmatrix} -2\,k_a/c & k_b \\ -m_a & m_b\,c/2 \end{bmatrix}$$

mit den instationären aerodynamischen Beiwerten

$$k_a(k) = -1 + \frac{2i}{k} C(k) \quad , \quad m_a = \frac{1}{2}$$

$$k_b(k) = -\frac{1}{2} + \frac{i}{k}\left(1 + 2\,C(k)\right) + \frac{2\,C(k)}{k^2} \quad , \quad m_b(k) = \frac{3}{8} - \frac{i}{k} \quad .$$

Einsetzen der aerodynamischen Beiwerte ergibt:

$$\left[A(k)\right]_N = \pi S \begin{bmatrix} -2\left(2\,i\,C(k)\,k - k^2\right)/c & 2\,C(k) + i\,k\left(1 + 2\,C(k)\right) - k^2/2 \\ -k^2/2 & \left(3\,k^2/8 - i\,k\right)c/2 \end{bmatrix}$$

GNU Octave Code

```
function A = aero2dus(k, c, S)

# usage: A = aero2dus(k, c, S))
#
# Input  k(:)         Reduced frequencies
#        c            Chord
#        S            Surface (optional, default: 1)
#
# Output A(2, 2, :)  Aerodynamic matrices
#
# The function computes the unsteady aerodynamic matrix.
# This matrix relates the motion of the aerodynamic
# centre to the aerodynamic loads at the aerodynamic
# centre.
#
# ------------------------------------------------------

# Check arguments

  if (nargin < 2 || nargin > 3 || nargout != 1)
     print_usage();
  end
```

```
  if (! isempty(find(k < 0)))
     error("Reduced frequencies must be positive\n");
  end

  if (! c > 0)
     error("Chord must be positive\n");
  end

  if (nargin == 3)
     if (! S > 0)
        error("Surface must be positive\n");
     end
  else
     S = 1;
  end

# Theodorsen function

  C  = theodorsen(k);
  k2 = k.^2;

# Matrix elements

  A(1, 1, :) = -2 * (2 * i * C .* k - k2) / c;
  A(1, 2, :) = 2 * C + i * k .* (1 + 2 * C) - 0.5 * k2;
  A(2, 1, :) = -0.5 * k2;
  A(2, 2, :) = (0.375 * k2 - i * k) * c / 2;

  A = pi * S * A;

end
```

Aufgabe 3

Die zu lösende Gleichung lautet

$$\left([K] - \mu \left(4 \frac{k^2}{c^2} [M] + \frac{\rho}{2} [A(k)] \right) \right) [\hat{u}] = [0] \quad \text{mit} \quad \mu = \frac{v_\infty^2}{1 + i\gamma} \quad .$$

Dabei beschreibt der Vektor $[\hat{u}]$ die Bewegung des Schwerpunkts. Für den Strukturdämpfungskoeffizienten γ und die Geschwindigkeit v_∞ gilt:

$$\gamma = -\frac{\Im(\mu)}{\Re(\mu)}, \quad v_\infty = \sqrt{\frac{\Re^2(\mu) + \Im^2(\mu)}{\Re(\mu)}}$$

GNU Octave Code der Funktion **flutter_k**

```
function [v, g, w, u] = flutter_k(K, M, A, c, rho, k)

# usage: [v, g, w, u] = flutter_k(K, M, A, c, rho, k)
#
```

```
# Input  K           Stiffness matrix
#        M           Mass matrix
#        A(2, 2, :)  List of areodynamic matrices
#        c           Chord
#        rho         Mass density of air
#        k(:)        Reduced frequency
#
# Output v(2, :)     Velocity
#        g(2, :)     Structural damping coefficients
#        w(2, :)     Circular frequencies
#        u(2, 2, :)  Eigenvectors (optional)
#
# The function solves the flutter equation using the k-method.
#
# ------------------------------------------------------------

# Check arguments

  if (nargin != 6 || nargout < 3 || nargout > 4)
     print_usage();
  end
  return_u = nargout == 4;

  if (! isempty(find(k < 0)))
     error("Reduced frequencies must be positive\n");
  end

  if (! c > 0)
     error("Semi-chord must be positive\n");
  end

  if (! rho > 0)
     error("Mass density must be positive\n");
  end

# Scale aerodynamic matrix

  A = 0.5 * rho * A;

# Solution at first reduced frequency

  MA = (2 * k(1) / c)^2 * M + A(:, :, 1);

  [x, myinv] = eig(MA, K);
  my = 1 ./ diag(myinv);

  g1 = -imag(my) ./ real(my);
  v1 = sqrt((real(my).^2 + imag(my).^2) ./ real(my));
  w1 = 2 * v1 * k(1) / c;

  [w(:, 1), ix] = sort(w1);
  v(:, 1)       = v1(ix);
  g(:, 1)       = g1(ix);
  x1            = x(:, ix);
  nx1           = norm(x1, "cols");
```

```
  if (return_u) u(:, :, 1) = x1; end

# Solution at remaining reduced frequencies

  nk = length(k);
  if (nk < 2) return; end

  for n = 2 : nk

      MA = (2 * k(n) / c)^2 * M + A(:, :, n);

      [x, myinv] = eig(MA, K);
      my = 1 ./ diag(myinv);

      g1 = -imag(my) ./ real(my);
      v1 = sqrt((real(my).^2 + imag(my).^2) ./ real(my));
      w1 = 2 * v1 * k(n) / c;
      nx = norm(x, "cols");

      % Check for mode crossing

      ix = [1, 2];

      c1 = x1(:, 1)' * x(:, 1) / (nx1(1) * nx(1));
      c2 = x1(:, 1)' * x(:, 2) / (nx1(1) * nx(2));

      if (abs(c2) > abs(c1)) ix = [2, 1]; end

      % Store results

      v(:, n) = v1(ix);
      g(:, n) = g1(ix);
      w(:, n) = w1(ix);

      x1 = x(:, ix); nx1 = nx(:, ix);

      if (return_u) u(:, :, n) = x1; end

  end

end
```

GNU Octave Skript zur Lösung des Testproblems

```
# Übungsblatt 1.3, Aufgabe 3: Test der Funktion flutter_k
#
# ------------------------------------------------------------

  wd = pwd; ix = strfind(wd, "Aeroelastik")-1;
  addpath([wd(1 : ix), "Aeroelastik/Tools"]);

  set(0, "defaultlinelinewidth", 2);
  set(0, "defaultaxesfontname", "Arial");
  set(0, "defaultaxesfontsize", 12);
```

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

# Daten (kg, m)

  c   =   0.4;   % Halbe Flügeltiefe
  S   =   0.4;   % Flügelfläche
  xN  =   0.1;   % Neutralpunkt
  xE  =  0.15;   % Elastisches Zentrum
  xS  =   0.2;   % Schwerpunkt
  m   =    25;   % Masse
  Jy  =   0.4;   % Massenträgheitsmoment bzgl. Schwerpunkt
  kH  =  5000;   % Hubsteifigkeit
  kT  =  1000;   % Nicksteifigkeit
  rho =  1.21;   % Massendichte der Luft

# Festlegung der reduzierten Frequenzen

  kmin = 0.025;  % kleinste reduzierte Frequenz
  kmax =   0.8;  % größte reduzierte Frequenz
  kinc = 0.005;  % Inkrement

# Matrizen

  M = diag([m, Jy]);
  K = [kH, kH * (xS - xE);
       kH * (xS - xE), kT + kH * (xS - xE)^2];

  TSN = [ 1, 0; xS - xN, 1];

  k  = kmin : kinc : kmax;
  nk = length(k);

  A = aero2dus(k, c, S);

  for n = 1 : nk
      A(:, :, n) = TSN * A(:, :, n) * TSN';
  end

# Schwingungen ohne Strömung

  [x, w0] = eig(M, K);
  w0 = 1 ./ sqrt(diag(w0));
  xm = max(abs(x));
  x  = x / diag(xm);

  fprintf(fid, "1. Schwingung: w = %7.2f 1/s, ", w0(1));
  fprintf(fid, "zS = %7.4f, alpha = %7.4f\n", x(:, 1));
  fprintf(fid, "2. Schwingung: w = %7.2f 1/s, ", w0(2));
  fprintf(fid, "zS = %7.4f, alpha = %7.4f\n", x(:, 2));

# Lösung der Fluttergleichung

  [v, g, w, u] = flutter_k(K, M, A, c, rho, k);
```

```
# Flattergeschwindigkeit

  izg = find(g(2, :) > 0);
  [vf, ixf] = min(v(2, izg));
  wf = w(2, ixf);
  kf = k(ixf);

  fprintf(fid, "\nFlattern:\n");
  fprintf(fid, "  v = %6.2f m/s, w = %6.2f 1/s, k = %6.3f\n",
          vf, wf, kf);

# V-g und V-w plots

  figure(1, "position", [100, 500, 750, 500],
            "paperposition", [0, 0, 15, 12]);
  subplot(2, 1, 1);
  plot(v(1, :), g(1, :), "color", "green",
       v(2, :), g(2, :), "color", "red");
  legend("Hubschwingung", "Nickschwingung",
         "location", "southwest");
  legend("boxoff"); legend("left");
  grid;
  set(gca(), "xlim", [0, 100]);
  ylim = get(gca(), "ylim");
  line([vf, vf], ylim);
  ylabel('\gamma');
  subplot(2, 1, 2);
  plot(v(1, :), w(1, :), "color", "green",
       v(2, :), w(2, :), "color", "red");
  legend("Hubschwingung", "Nickschwingung", "location", "west");
  legend("boxoff"); legend("left");
  grid;
  set(gca(), "xlim", [0, 100]);
  ylim = get(gca(), "ylim");
  line([vf, vf], ylim);
  ylabel('\omega [1/s]');
  xlabel("v [m/s]");

  print([file, ".jpg"], "-djpg");

  fclose(fid);
```

Die Datei **u1_3_3.res** enthält die folgenden Ergebnisse:

```
1. Schwingung: w =   50.34 1/s, zS = -0.0043, alpha = -1.0000
2. Schwingung: w =   14.05 1/s, zS = -1.0000, alpha =  0.2678

Flattern:
  v =  67.45 m/s, w =  26.98 1/s, k =  0.080
```

Die Verläufe von γ und ω sind in Abbildung 3.1 dargestellt.

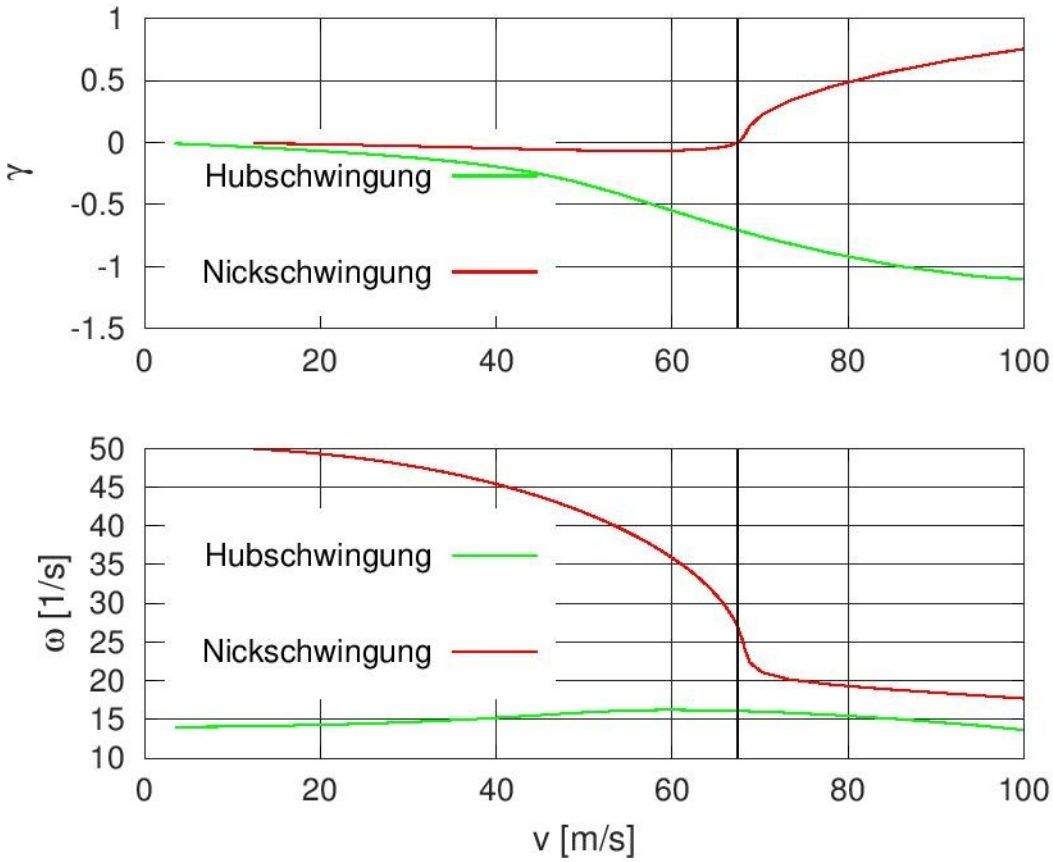

Abbildung 3.1: Flutterkurven