PROBLEM SET 2.1

Problem 1

(a)

$$\begin{aligned}
|\mathbf{A}| &= \begin{vmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{vmatrix} = 1 \begin{vmatrix} 3 & 4 \\ 4 & 5 \end{vmatrix} - 2 \begin{vmatrix} 2 & 4 \\ 3 & 5 \end{vmatrix} + 3 \begin{vmatrix} 2 & 3 \\ 3 & 4 \end{vmatrix} \\
&= 1(-1) - 2(-2) + 3(-1) = 0 \quad \text{Singular} \ \blacktriangleleft
\end{aligned}$$

(b)

$$\begin{aligned}
|\mathbf{A}| &= \begin{vmatrix} 2.11 & -0.80 & 1.72 \\ -1.84 & 3.03 & 1.29 \\ -1.57 & 5.25 & 4.30 \end{vmatrix} \\
&= 2.11 \begin{vmatrix} 3.03 & 1.29 \\ 5.25 & 4.30 \end{vmatrix} + 0.80 \begin{vmatrix} -1.84 & 1.29 \\ -1.57 & 4.30 \end{vmatrix} + 1.72 \begin{vmatrix} -1.84 & 3.03 \\ -1.57 & 5.25 \end{vmatrix} \\
&= 2.11(6.2565) + 0.80(-5.8867) + 1.72(-4.9029) \\
&= 0.058\,867 \quad \text{Ill conditioned} \ \blacktriangleleft
\end{aligned}$$

(c)

$$\begin{aligned}
|\mathbf{A}| &= \begin{vmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{vmatrix} = 2 \begin{vmatrix} 2 & -1 \\ -1 & 2 \end{vmatrix} + 1 \begin{vmatrix} -1 & -1 \\ 0 & 2 \end{vmatrix} \\
&= 2(3) + 1(-2) = 4 \quad \text{Well-conditioned} \ \blacktriangleleft
\end{aligned}$$

(d)

$$\begin{aligned}
|\mathbf{A}| &= \begin{vmatrix} 4 & 3 & -1 \\ 7 & -2 & 3 \\ 5 & -18 & 13 \end{vmatrix} = 4 \begin{vmatrix} -2 & 3 \\ -18 & 13 \end{vmatrix} - 3 \begin{vmatrix} 7 & 3 \\ 5 & 13 \end{vmatrix} - 1 \begin{vmatrix} 7 & -2 \\ 5 & -18 \end{vmatrix} \\
&= 4(28) - 3(76) - 1(-116) = 0 \quad \text{Singular} \ \blacktriangleleft
\end{aligned}$$

Problem 2

(a)

$$\mathbf{A} = \mathbf{LU} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 5/3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 4 \\ 0 & 3 & 21 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 4 \\ 1 & 5 & 25 \\ 1 & 7 & 39 \end{bmatrix} \quad \blacktriangleleft$$

$$|\mathbf{A}| = |\mathbf{L}|\,|\mathbf{U}| = (1 \times 1 \times 1)(1 \times 3 \times 0) = 0 \quad \blacktriangleleft$$

(b)

$$\mathbf{A} = \mathbf{LU} = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & -3 & 1 \end{bmatrix} \begin{bmatrix} 2 & -1 & 1 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 4 & -2 & 2 \\ -2 & 2 & -4 \\ 2 & -4 & 11 \end{bmatrix} \quad \blacktriangleleft$$

$$|\mathbf{A}| = |\mathbf{L}|\,|\mathbf{U}| = (2 \times 1 \times 1)(2 \times 1 \times 1) = 4 \quad \blacktriangleleft$$

Problem 3

First solve $\mathbf{Ly} = \mathbf{b}$:

$$\begin{bmatrix} 1 & 0 & 0 \\ 3/2 & 1 & 0 \\ 1/2 & 11/13 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix}$$

$$\begin{aligned} y_1 &= 1 \\ \frac{3}{2}(1) + y_2 &= -1 \qquad y_2 = -\frac{5}{2} \\ \frac{1}{2}(1) + \frac{11}{13}\left(\frac{-5}{2}\right) + y_3 &= 2 \qquad y_3 = \frac{47}{13} \end{aligned}$$

Then solve $\mathbf{Ux} = \mathbf{y}$:

$$\begin{bmatrix} 2 & -3 & -1 \\ 0 & 13/2 & -7/2 \\ 0 & 0 & 32/13 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -5/2 \\ 47/13 \end{bmatrix}$$

$$\begin{aligned} \frac{32}{13}x_3 &= \frac{47}{13} \qquad x_3 = \frac{47}{32} \quad \blacktriangleleft \\ \frac{13}{2}x_2 - \frac{7}{2}\left(\frac{47}{32}\right) &= -\frac{5}{2} \qquad x_2 = \frac{13}{32} \quad \blacktriangleleft \\ 2x_1 - 3\left(\frac{13}{32}\right) - \frac{47}{32} &= 1 \qquad x_1 = \frac{59}{32} \quad \blacktriangleleft \end{aligned}$$

Problem 4

The augmented coefficient matrix is

$$[\mathbf{A}|\mathbf{b}] = \begin{bmatrix} 2 & -3 & -1 & 3 \\ 3 & 2 & -5 & -9 \\ 2 & 4 & -1 & -5 \end{bmatrix}$$

Elimination phase:

$$\begin{aligned} \text{row 2} &\leftarrow \text{row 2} - \frac{3}{2} \times \text{row 1} \\ \text{row 3} &\leftarrow \text{row 3} - \text{row 1} \end{aligned}$$

$$\begin{bmatrix} 2 & -3 & -1 & 3 \\ 0 & 13/2 & -7/2 & -27/2 \\ 0 & 7 & 0 & -8 \end{bmatrix}$$

$$\text{row 3} \leftarrow \text{row 3} - \frac{14}{13} \times \text{row 2}$$

$$\begin{bmatrix} 2 & -3 & -1 & 3 \\ 0 & 13/2 & -7/2 & -27/2 \\ 0 & 0 & 49/13 & 85/13 \end{bmatrix}$$

Solution by back substitution:

$$\begin{aligned} \frac{49}{13}x_3 &= \frac{85}{13} & x_3 &= \frac{85}{49} = 1.7347 \quad \blacktriangleleft \\ \frac{13}{2}x_2 - \frac{7}{2}\left(\frac{85}{49}\right) &= -\frac{27}{2} & x_2 &= -\frac{8}{7} = -1.1429 \quad \blacktriangleleft \\ 2x_1 - 3\left(-\frac{8}{7}\right) - \frac{85}{49} &= 3 & x_1 &= \frac{32}{49} = 0.6531 \quad \blacktriangleleft \end{aligned}$$

Problem 5

The augmented coefficient matrix is

$$[\mathbf{A}|\mathbf{B}] = \begin{bmatrix} 2 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 \\ -1 & 2 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -2 & 0 & 0 \end{bmatrix}$$

Before elimination, we exchange rows 2 and 3 in order to reduce the amount of algebra:

$$\left[\begin{array}{rrrrrr} 2 & 0 & -1 & 0 & 1 & 0 \\ -1 & 2 & 0 & 1 & 0 & 1 \\ 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 0 & 0 \end{array}\right]$$

Elimination phase:

$$\text{row } 2 \leftarrow \text{row } 2 + \frac{1}{2} \times \text{row } 1$$

$$\left[\begin{array}{rrrrrr} 2 & 0 & -1 & 0 & 1 & 0 \\ 0 & 2 & -1/2 & 1 & 1/2 & 1 \\ 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 0 & 0 \end{array}\right]$$

$$\text{row } 3 \leftarrow \text{row } 3 - \frac{1}{2} \times \text{row } 2$$

$$\left[\begin{array}{rrrrrr} 2 & 0 & -1 & 0 & 1 & 0 \\ 0 & 2 & -1/2 & 1 & 1/2 & 1 \\ 0 & 0 & 9/4 & -1/2 & -1/4 & -1/2 \\ 0 & 0 & 1 & -2 & 0 & 0 \end{array}\right]$$

$$\text{row } 4 \leftarrow \text{row } 4 - \frac{4}{9} \times \text{row } 3$$

$$\left[\begin{array}{rrrrrr} 2 & 0 & -1 & 0 & 1 & 0 \\ 0 & 2 & -1/2 & 1 & 1/2 & 1 \\ 0 & 0 & 9/4 & -1/2 & -1/4 & -1/2 \\ 0 & 0 & 0 & -16/9 & 1/9 & 2/9 \end{array}\right]$$

First solution vector by back substitution:

$$\begin{aligned} -\frac{16}{9}x_4 &= \frac{1}{9} & x_4 &= -\frac{1}{16} \\ \frac{9}{4}x_3 - \frac{1}{2}\left(-\frac{1}{16}\right) &= -\frac{1}{4} & x_3 &= -\frac{1}{8} \\ 2x_2 - \frac{1}{2}\left(-\frac{1}{8}\right) + \left(-\frac{1}{16}\right) &= \frac{1}{2} & x_2 &= \frac{1}{4} \\ 2x_1 - \left(-\frac{1}{8}\right) &= 1 & x_1 &= \frac{7}{16} \end{aligned}$$

Second solution vector:

$$\begin{aligned}
-\frac{16}{9}x_4 &= \frac{2}{9} & x_4 = -\frac{1}{8} \\
\frac{9}{4}x_3 - \frac{1}{2}\left(-\frac{1}{8}\right) &= -\frac{1}{2} & x_3 = -\frac{1}{4} \\
2x_2 - \frac{1}{2}\left(-\frac{1}{4}\right) + \left(-\frac{1}{8}\right) &= 1 & x_2 = \frac{1}{2} \\
2x_1 - \left(-\frac{1}{4}\right) &= 0 & x_1 = -\frac{1}{8}
\end{aligned}$$

Therefore,

$$\mathbf{X} = \begin{bmatrix} 7/16 & -1/8 \\ 1/4 & 1/2 \\ -1/8 & -1/4 \\ -1/16 & -1/8 \end{bmatrix} \quad \blacktriangleleft$$

Problem 6

After reordering rows, the augmented coefficient matrix is

$$\begin{bmatrix} 1 & 2 & 0 & -2 & 0 & -4 \\ 0 & 1 & -1 & 1 & -1 & -1 \\ 0 & 1 & 0 & 2 & -1 & 1 \\ 0 & 0 & 2 & 1 & 2 & 1 \\ 0 & 0 & 0 & -1 & 1 & -2 \end{bmatrix}$$

Elimination phase:

$$\text{row 3} \leftarrow \text{row 3} - \text{row 2}$$

$$\begin{bmatrix} 1 & 2 & 0 & -2 & 0 & -4 \\ 0 & 1 & -1 & 1 & -1 & -1 \\ 0 & 0 & 1 & 1 & 0 & 2 \\ 0 & 0 & 2 & 1 & 2 & 1 \\ 0 & 0 & 0 & -1 & 1 & -2 \end{bmatrix}$$

$$\text{row 4} \leftarrow \text{row 4} - 2 \times \text{row 3}$$

$$\begin{bmatrix} 1 & 2 & 0 & -2 & 0 & -4 \\ 0 & 1 & -1 & 1 & -1 & -1 \\ 0 & 0 & 1 & 1 & 0 & 2 \\ 0 & 0 & 0 & -1 & 2 & -3 \\ 0 & 0 & 0 & -1 & 1 & -2 \end{bmatrix}$$

$$\text{row 5} \leftarrow \text{row 5} - \text{row 4}$$

$$\left[\begin{array}{rrrrrr} 1 & 2 & 0 & -2 & 0 & -4 \\ 0 & 1 & -1 & 1 & -1 & -1 \\ 0 & 0 & 1 & 1 & 0 & 2 \\ 0 & 0 & 0 & -1 & 2 & -3 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{array}\right]$$

Back substitution:

$$\begin{array}{rcll} -x_5 & = & 1 & x_5 = -1 \blacktriangleleft \\ -x_4 + 2(-1) & = & -3 & x_4 = 1 \blacktriangleleft \\ x_3 + 1 & = & 2 & x_3 = 1 \blacktriangleleft \\ x_2 - 1 + 1 - (-1) & = & -1 & x_2 = -2 \blacktriangleleft \\ x_1 + 2(-2) - 2(1) & = & -4 & x_1 = 2 \blacktriangleleft \end{array}$$

Problem 7

(a)

$$\mathbf{A} = \left[\begin{array}{rrr} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{array}\right]$$

Use Gauss elimination storing each multiplier in the location occupied by the element that was eliminated. The multipliers are enclosed in boxes $\boxed{\text{thus}}$.

$$\text{row 2} \leftarrow \text{row 2} - \left(-\frac{1}{4}\right) \times \text{row 2}$$

$$\left[\begin{array}{rrr} 4 & -1 & 0 \\ \boxed{-1/4} & 15/4 & -1 \\ 0 & -1 & 4 \end{array}\right]$$

$$\text{row 3} \leftarrow \text{row 3} - \left(-\frac{4}{15}\right) \times \text{row 2}$$

$$\left[\begin{array}{rrr} 4 & -1 & 0 \\ \boxed{-1/4} & 15/4 & -1 \\ 0 & \boxed{-4/15} & 56/15 \end{array}\right]$$

Thus

$$\mathbf{U} = \left[\begin{array}{rrr} 4 & -1 & 0 \\ 0 & 15/4 & -1 \\ 0 & 0 & 56/15 \end{array}\right] \blacktriangleleft \qquad \mathbf{L} = \left[\begin{array}{rrr} 1 & 0 & 0 \\ -1/4 & 1 & 0 \\ 0 & -4/15 & 1 \end{array}\right] \blacktriangleleft$$

(b)

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T$$

Substituting for $\mathbf{L}\mathbf{L}^T$ from Eq. (2.16), we get

$$\begin{bmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix} = \begin{bmatrix} L_{11}^2 & L_{11}L_{21} & L_{11}L_{31} \\ L_{11}L_{21} & L_{21}^2 + L_{22}^2 & L_{21}L_{31} + L_{22}L_{32} \\ L_{11}L_{31} & L_{21}L_{31} + L_{22}L_{32} & L_{31}^2 + L_{32}^2 + L_{33}^2 \end{bmatrix}$$

Equating matrices term-by term:

$$\begin{aligned} L_{11}^2 &= 4 & L_{11} &= 2 \\ 2L_{21} &= -1 & L_{21} &= -\frac{1}{2} \\ 2L_{31} &= 0 & L_{31} &= 0 \\ \left(-\frac{1}{2}\right)^2 + L_{22}^2 &= 4 & L_{22} &= \frac{\sqrt{15}}{2} \\ -\frac{1}{2}(0) + \frac{\sqrt{15}}{2}L_{32} &= -1 & L_{32} &= -\frac{2}{\sqrt{15}} \\ 0^2 + \left(-\frac{2}{\sqrt{15}}\right)^2 + L_{33}^2 &= 4 & L_{32} &= 2\sqrt{\frac{14}{15}} \end{aligned}$$

Therefore,

$$\mathbf{L} = \begin{bmatrix} 2 & 0 & 0 \\ -1/2 & \sqrt{15}/2 & 0 \\ 0 & -2/\sqrt{15} & 2\sqrt{14/15} \end{bmatrix} \blacktriangleleft$$

Problem 8

$$\mathbf{A} = \begin{bmatrix} -3 & 6 & -4 \\ 9 & -8 & 24 \\ -12 & 24 & -26 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} -3 \\ 65 \\ -42 \end{bmatrix}$$

Decomposition of $\mathbf{A}$ (multipliers are enclosed in boxes):

$$\begin{aligned} \text{row 2} &\leftarrow \text{row 2} - (-3) \times \text{row 1} \\ \text{row 3} &\leftarrow \text{row 3} - 4 \times \text{row 1} \end{aligned}$$

$$\begin{bmatrix} -3 & 6 & -4 \\ \boxed{-3} & 10 & 12 \\ \boxed{4} & 0 & -10 \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} -3 & 6 & -4 \\ 0 & 10 & 12 \\ 0 & 0 & -10 \end{bmatrix} \qquad \mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 4 & 0 & 1 \end{bmatrix}$$

Solution of $\mathbf{Ly} = \mathbf{b}$:

$$\begin{aligned} y_1 &= -3 \\ -3(-3) + y_2 &= 65 \qquad y_2 = 56 \\ 4(-3) + y_3 &= -42 \qquad y_3 = -30 \end{aligned}$$

Solution of $\mathbf{Ux} = \mathbf{y}$:

$$\begin{aligned} -10x_3 &= -30 \qquad x_3 = 3 \ \blacktriangleleft \\ 10x_2 + 12(3) &= 56 \qquad x_2 = 2 \ \blacktriangleleft \\ -3x_1 + 6(2) - 4(3) &= -3 \qquad x_1 = 1 \ \blacktriangleleft \end{aligned}$$

Problem 9

$$\mathbf{A} = \begin{bmatrix} 2.34 & -4.10 & 1.78 \\ -1.98 & 3.47 & -2.22 \\ 2.36 & -15.17 & 6.18 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 0.02 \\ -0.73 \\ -6.63 \end{bmatrix}$$

Decomposition of $\mathbf{A}$ (multipliers are enclosed in boxes):

$$\begin{aligned} \text{row 2} &\leftarrow \text{row 2} - (-0.846\,154) \times \text{row 1} \\ \text{row 3} &\leftarrow \text{row 3} - 1.008\,547 \times \text{row 1} \end{aligned}$$

$$\begin{bmatrix} 2.34 & -4.10 & 1.78 \\ \boxed{-0.846\,154} & 0.000\ 769 & -0.713\,846 \\ \boxed{1.008\,547} & -11.034\,96 & 4.384\,786 \end{bmatrix}$$

$$\text{row 3} \leftarrow \text{row 3} - (-14349.75) \times \text{row 2}$$

$$\begin{bmatrix} 2.34 & -4.10 & 1.78 \\ \boxed{-0.846\,154} & 0.000\ 769 & -0.713\,846 \\ \boxed{1.008\,547} & \boxed{-14349.75} & -10239.13 \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} 2.34 & -4.10 & 1.78 \\ 0 & 0.000\ 769 & -0.713\,846 \\ 0 & 0 & -10\,239.1 \end{bmatrix} \qquad \mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ -0.846\,154 & 1 & 0 \\ 1.008\,547 & -14349.7 & 1 \end{bmatrix}$$

Solution of $\mathbf{Ly} = \mathbf{b}$:

$$\begin{aligned} y_1 &= 0.02 \\ -0.846\,154(0.02) + y_2 &= -0.73 \qquad y_2 = -0.713\,07\,7 \\ 1.008\,547(0.02) - 14349.7(-0.713\,07\,7) + y_3 &= -6.63 \qquad y_3 = -10\,239.1 \end{aligned}$$

Solution of $\mathbf{Ux} = \mathbf{y}$:

$$\begin{aligned} -10\,239.1x_3 &= -10\,239.1 \qquad x_3 = 1.0 \ \blacktriangleleft \\ 0.000\ 769x_2 - 0.713\,846 &= -0.713\,07\,7 \qquad x_2 = 1.0 \ \blacktriangleleft \\ 2.34x_1 - 4.10 + 1.78 &= 0.02 \qquad x_1 = 1.0 \ \blacktriangleleft \end{aligned}$$

Problem 10

$$\mathbf{A} = \begin{bmatrix} 4 & -3 & 6 \\ 8 & -3 & 10 \\ -4 & 12 & -10 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Decomposition of $\mathbf{A}$ (multipliers are enclosed in boxes):

$$\begin{aligned} \text{row } 2 &\leftarrow \text{row } 2 - 2 \times \text{row } 1 \\ \text{row } 3 &\leftarrow \text{row } 3 - (-1) \times \text{row } 1 \end{aligned}$$

$$\begin{bmatrix} 4 & -3 & 6 \\ \boxed{2} & 3 & -2 \\ \boxed{-1} & 9 & -4 \end{bmatrix}$$

$$\text{row } 3 \leftarrow \text{row } 3 - 3 \times \text{row } 2$$

$$\begin{bmatrix} 4 & -3 & 6 \\ \boxed{2} & 3 & -2 \\ \boxed{-1} & \boxed{3} & 2 \end{bmatrix}$$

$$U = \begin{bmatrix} 4 & -3 & 6 \\ 0 & 3 & -2 \\ 0 & 0 & 2 \end{bmatrix} \qquad L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 3 & 1 \end{bmatrix}$$

<u>First solution vector</u>

Solution of $\mathbf{Ly} = \mathbf{b}$:

$$\begin{aligned} y_1 &= 1 \\ 2(1) + y_2 &= 0 \qquad y_2 = -2 \\ -1 + 3(-2) + y_3 &= 0 \qquad y_3 = 7 \end{aligned}$$

Solution of $\mathbf{Uy} = \mathbf{x}$:

$$\begin{aligned} 2x_3 &= 7 \qquad x_3 = \frac{7}{2} \\ 3x_2 - 2\left(\frac{7}{2}\right) &= -2 \qquad x_2 = \frac{5}{3} \\ 4x_1 - 3\left(\frac{5}{3}\right) + 6\left(\frac{7}{2}\right) &= 1 \qquad x_1 = -\frac{15}{4} \end{aligned}$$

<u>Second solution vector</u>

Solution of $\mathbf{Ly} = \mathbf{b}$:

$$\begin{aligned} y_1 &= 0 \\ 2(0) + y_2 &= 1 \qquad y_2 = 1 \\ -1(0) + 3(1) + y_3 &= 0 \qquad y_3 = -3 \end{aligned}$$

Solution of $\mathbf{Ux} = \mathbf{y}$:

$$\begin{aligned}
2x_3 &= -3 \qquad x_3 = -\frac{3}{2} \\
3x_2 - 2\left(-\frac{3}{2}\right) &= 1 \qquad x_3 = -\frac{2}{3} \\
4x_1 - 3\left(-\frac{2}{3}\right) + 6\left(-\frac{3}{2}\right) &= 0 \qquad x_1 = \frac{7}{4}
\end{aligned}$$

Therefore,

$$\mathbf{X} = \begin{bmatrix} 7/2 & -3/2 \\ 5/3 & -2/3 \\ -15/4 & 7/4 \end{bmatrix} \blacktriangleleft$$

Problem 11

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 1 \\ 3/2 \\ 3 \end{bmatrix}$$

Substituting for $\mathbf{LL}^T$ from Eq. (2.16), we get

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} L_{11}^2 & L_{11}L_{21} & L_{11}L_{31} \\ L_{11}L_{21} & L_{21}^2 + L_{22}^2 & L_{21}L_{31} + L_{22}L_{32} \\ L_{11}L_{31} & L_{21}L_{31} + L_{22}L_{32} & L_{31}^2 + L_{32}^2 + L_{33}^2 \end{bmatrix}$$

Equating matrices term-by term:

$$\begin{aligned}
L_{11} &= 1 \qquad L_{21} = 1 \qquad L_{31} = 1 \\
1^2 + L_{22}^2 &= 2 \qquad L_{22} = 1 \\
(1)(1) + (1)L_{32} &= 2 \qquad L_{32} = 1 \\
1^2 + 1^2 + L_{33}^2 &= 3 \qquad L_{33} = 1
\end{aligned}$$

Thus

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \mathbf{L}^T = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Solution of $\mathbf{Ly} = \mathbf{b}$:

$$\begin{aligned}
y_1 &= 1 \\
1 + y_2 &= \frac{3}{2} \qquad y_2 = \frac{1}{2} \\
1 + \frac{1}{2} + y_3 &= 3 \qquad y_3 = \frac{3}{2}
\end{aligned}$$

Solution of $\mathbf{L}^T\mathbf{x} = \mathbf{y}$:

$$\begin{aligned}
x_3 &= \frac{3}{2} \quad \blacktriangleleft \\
x_2 + \frac{3}{2} &= \frac{1}{2} \qquad x_2 = -1 \quad \blacktriangleleft \\
x_1 - 1 + \frac{3}{2} &= 1 \qquad x_1 = \frac{1}{2} \quad \blacktriangleleft
\end{aligned}$$

Problem 12

$$A = \begin{bmatrix} 4 & -2 & -3 \\ 12 & 4 & -10 \\ -16 & 28 & 18 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 1.1 \\ 0 \\ -2.3 \end{bmatrix}$$

Decomposition of $\mathbf{A}$ (multipliers are enclosed in boxes):

$$\begin{aligned}
\text{row } 2 &\leftarrow \text{row } 2 - 3 \times \text{row } 1 \\
\text{row } 3 &\leftarrow \text{row } 3 - (-4) \times \text{row } 1
\end{aligned}$$

$$\begin{bmatrix} 4 & -2 & -3 \\ \boxed{3} & 10 & -1 \\ \boxed{-4} & 20 & 6 \end{bmatrix}$$

$$\text{row } 3 \leftarrow \text{row } 3 - 2 \times \text{row } 2$$

$$\begin{bmatrix} 4 & -2 & -3 \\ \boxed{3} & 10 & -1 \\ \boxed{-4} & \boxed{2} & 8 \end{bmatrix}$$

Therefore

$$\mathbf{U} = \begin{bmatrix} 4 & -2 & -3 \\ 0 & 10 & -1 \\ 0 & 0 & 8 \end{bmatrix} \qquad \mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ -4 & 2 & 1 \end{bmatrix}$$

Solution of $\mathbf{Ly} = \mathbf{b}$:

$$\begin{aligned}
y_1 &= 1.1 \\
3(1.1) + y_2 &= 0 \qquad y_2 = -3.3 \\
-4(1.1) + 2(-3.3) + y_3 &= -2.3 \qquad y_3 = 8.7
\end{aligned}$$

Solution of $\mathbf{Ux} = \mathbf{y}$:

$$\begin{aligned}
8x_3 &= 8.7 \qquad x_3 = 1.0875 \quad \blacktriangleleft \\
10x_2 - 1.0875 &= -3.3 \qquad x_2 = -0.221\,25 \quad \blacktriangleleft \\
4x_1 - 2(-0.221\,25) - 3(1.0875) &= 1.1 \qquad x_1 = 0.98 \quad \blacktriangleleft
\end{aligned}$$

Problem 13

$$\mathbf{A} = \begin{bmatrix} \alpha_1 & 0 & 0 & \cdots \\ 0 & \alpha_2 & 0 & \cdots \\ 0 & 0 & \alpha_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Since the banded structure of a matrix is preserved during decomposition, **L** must be a diagonal matrix. Therefore,

$$\mathbf{LL}^T = \begin{bmatrix} L_{11}^2 & 0 & 0 & \cdots \\ 0 & L_{22}^2 & 0 & \cdots \\ 0 & 0 & L_{33}^2 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

It follows from $\mathbf{A} = \mathbf{LL}^T$ that

$$\mathbf{L} = \begin{bmatrix} \sqrt{\alpha_1} & 0 & 0 & \cdots \\ 0 & \sqrt{\alpha_2} & 0 & \cdots \\ 0 & 0 & \sqrt{\alpha_3} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \quad \blacktriangleleft$$

Problem 14

```
## problem2_1_14
from numpy import dot,array

def gaussElimin(a,b):
    n,m = b.shape     # Replaces n = len(b)
  # Elimination Phase
    for k in range(0,n-1):
        for i in range(k+1,n):
           if a[i,k] != 0.0:
               lam = a [i,k]/a[k,k]
               a[i,k+1:n] = a[i,k+1:n] - lam*a[k,k+1:n]
               b[i] = b[i] - lam*b[k]
  # Back substitution
    for k in range(n-1,-1,-1):
        for i in range(m): # New loop (over all constant vectors)
            b[k,i] = (b[k,i] - dot(a[k,k+1:n],b[k+1:n,i]))/a[k,k]
    return b
```

```
a = array([[2,-1,0],[-1,2,-1],[0,-1,1]])*1.0
b = array([[1,0,0],[0,1,0],[0,0,1]])*1.0
print(''The solution is:\n'',gaussElimin(a,b))
input(''\nPress return to exit'')

The solution is:
[[ 1.  1.  1.]
 [ 1.  2.  2.]
 [ 1.  2.  3.]]
```

Problem 15

This program prompts for the number of equations.

```
## problem2_1_15
from numpy import zeros,array
from LUdecomp import *

n = eval(raw_input(''Number of equations ==> ''))
a = zeros((n,n))
b = zeros(n)
for i in range(n):
   for j in range(n):
      a[i,j] =1.0/(i+j+1)
      b[i] = b[i] + a[i,j]
LUdecomp(a)
LUsolve(a,b)
print(''The solution is:\n'',b)
input(''\nPress return to exit'')
```

The largest n for which 6-figure accuracy is achieved seems to be 8:

```
Number of equations ==> 8
The solution is:
[ 1.         1.         0.99999997  1.00000017  0.99999955  1.00000064
  0.99999955  1.00000013]
```

Problem 16

Forward sustitution The kth equation of $\mathbf{Ly} = \mathbf{b}$ is

$$L_{k1}y_1 + L_{k2}y_2 + \cdots + L_{kk}y_k = b_k$$

Solving for y_k yields

$$\begin{aligned} y_k &= \frac{b_k - (L_{k,1}y_1 + L_{k,2}y_2 + \cdots + L_{k,k-1}y_{k-1})}{L_{kk}} \\ &= b_k - \frac{\begin{bmatrix} L_{k,1} & L_{k,2} & \cdots & L_{k,k-1} \end{bmatrix} \cdot \begin{bmatrix} y_1 & y_2 & \cdots & y_{k-1} \end{bmatrix}}{L_{k,k}} \end{aligned}$$

This expression, evaluated with $k = 1, 2, \ldots, n$ (in that order), constitutes the forward substitution phase In `choleskiSol` the b's are overwritten with y's during the computations.

Back substitution A typical (kth) equation of $\mathbf{L}^T\mathbf{x} = \mathbf{y}$ is

$$L_{k,k}x_k + L_{k+1,k}x_{k+1} + L_{k+2,k}x_{k+2} + \cdots + L_{n,k}x_n = y_k$$

The solution for x_k is

$$\begin{aligned} x_k &= \frac{y_k - (L_{k+1,k}x_{k+1} + L_{k+2,k}x_{k+2} + \ldots L_{n,k}x_n)}{L_{k,k}} \\ &= \frac{y_k - \begin{bmatrix} L_{k+1,k} & L_{k+2,k} & \cdots & L_{n,k} \end{bmatrix} \cdot \begin{bmatrix} x_{k+1} & x_{k+2} & \cdots & x_n \end{bmatrix}}{L_{kk}} \end{aligned}$$

In back substitution we evaluate this expression in the order $k = n, n-1, \ldots, 1$. Note that in `choleskiSol` the vector $\mathbf{x}$ overwrites the vector $\mathbf{y}$.

Problem 17

```
## problem2_1_17
from numpy import zeros,array
from LUdecomp import *

x = array([0,1,3,4],float)
y = array([10,35,31,2],float)
n = len(x)
a = zeros((n,n))
for i in range(n):
   a[0:n,i] = x[0:n]**i
LUdecomp(a)
LUsolve(a,y)
print(''The coefficients are:\n'',y)
input(''\nPress return to exit'')

The coefficients are:
[ 10.  34.  -9.   0.]
```

Problem 18

```
## problem2_1_18
from numpy import zeros,array
from LUdecomp import *

x = array([0,1,3,5,6],float)
y = array([-1,1,3,2,-2],float)
n = len(x)
a = zeros((n,n))
for i in range(n):
   a[0:n,i] = x[0:n]**i
LUdecomp(a)
LUsolve(a,y)
print(''The coefficients are:\n'',y)
input(''\nPress return to exit''

The coefficients are:
 [-1.  2.68333333 -0.875  0.21666667 -0.025]
```

Problem 19

$$
\begin{aligned}
f(x) &= c_0 + c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4 \\
f''(x) &= 2c_2 + 6c_3 x + 12c_4 x^2
\end{aligned}
$$

The specified conditions result in the equations

$$\mathbf{Ac} = \mathbf{y}$$

```
## problem2_1_19
from numpy import zeros,array
from LUdecomp import *
a = zeros((5,5))
a[0] = [1,0,0,0,0]                          # f(0)
a[1] = [1,0.75,0.75**2,0.75**3,0.75**4]   # f(0.75)
a[2] = [1,1,1,1,1]                          # f(1)
a[3] = [0,0,2,0,0]                          # f''(0)
a[4] = [0,0,2,6,12]                         # f''(1)
y = array([1,-0.25,1,0,0])
LUdecomp(a)
LUsolve(a,y)
```

```
print(''The coefficients are:\n'',y)
input(''\nPress return to exit'')

The coefficients are:
[  1.00000000e+00  -5.61403509e+00   1.77635684e-15   1.12280702e+01
  -5.61403509e+00]
```

Therefore, the polynomial is

$$f(x) = 1 - 5.6140x + 11.2281x^3 - 5.6140x^4$$

Problem 20

$$\mathbf{A} = \begin{bmatrix} 3.50 & 2.77 & -0.76 & 1.80 \\ -1.80 & 2.68 & 3.44 & -0.09 \\ 0.27 & 5.07 & 6.90 & 1.61 \\ 1.71 & 5.45 & 2.68 & 1.71 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 7.31 \\ 4.23 \\ 13.85 \\ 11.55 \end{bmatrix}$$

```
## problem2_1_20
from numpy import zeros,array,dot
from LUdecomp import *

b = array([7.31,4.23,13.85,11.55])
n = len(b)
a = zeros((n,n))
a[0] = [ 3.50, 2.77,-0.76, 1.80]
a[1] = [-1.80, 2.68, 3.44,-0.09]
a[2] = [ 0.27, 5.07, 6.90, 1.61]
a[3] = [ 1.71, 5.45, 2.68, 1.71]
a_orig = a.copy()
LUdecomp(a)
LUsolve(a,b)
print("The solution is:\n",b)
x = b.copy()
det = 1.0
for i in range(n): det = det*a[i,i]
print("\nDeterminant =",det)
b = dot(a_orig,x)
print("\nThe product Ax is:\n",b)
input("\nPress return to exit")

The solution is:
[ 1.  1.  1.  1.]
```

```
Determinant = -0.22579734

The product Ax is:
[  7.31   4.23  13.85  11.55]
```

The determinant is a little smaller than the elements of $\mathbf{A}$, indicating a mild case of ill-conditioning. Noting that the normal printout of Python is rounded off to 8 significant figures after the decimal point, we conclude that the solution is at least 8-figure accurate. The discrepancy between the computed and the true solution would appear only if more figures are printed out.

Problem 21

Find inverse of $\mathbf{A}$ first.

```
>>> from numpy import array,float
>>> from numpy.linalg import inv
>>> A = array([[1,-1,-1],[0,1,-2],[0,0,1]],float)
>>> print(inv(A))
[[ 1.  1.  3.]
 [ 0.  1.  2.]
 [ 0.  0.  1.]
```

(a)

$$\begin{aligned}
\|A\|_e &= \sqrt{1^2+1^2+1^2+1^2+2^2+1^2} = 3 \\
\left\|A^{-1}\right\|_e &= \sqrt{1^2+1^2+3^2+1^2+2^2+1^2} = \sqrt{17} \\
\mathrm{cond}_e &= \|A\|_e \left\|A^{-1}\right\|_e = 3\sqrt{17} = 12.37 \quad \blacktriangleleft
\end{aligned}$$

(b)

$$\begin{aligned}
\|A\|_\infty &= 3 \text{ (determined by row 1 or row 2)} \\
\left\|A^{-1}\right\|_\infty &= 5 \text{ (determined by row 1)} \\
\mathrm{cond}_\infty &= \|A\|_\infty \left\|A^{-1}\right\|_\infty = 3(5) = 15 \quad \blacktriangleleft
\end{aligned}$$

Problem 22

```
## problem 2_1_22
```

```
from numpy import array,float,sqrt
from numpy.linalg import inv

def eCond(A):
    eNorm = lambda A : sqrt(sum(sum(A**2)))
    return eNorm(A)*eNorm(inv(A))

A = array([[1,4,9,16],[4,9,16,25],[9,16,25,36],  \
           [16,25,36,49]],dtype=float)
print(eCond(A))
input("Press return to exit")
```

The output of this program is

```
  2.85629312616e+017
```

Problem 23

```
## problem2_1_23
from gaussElimin import *
from numpy import sum,zeros
from numpy.random import rand

n = 500
a = rand(n,n)
b = zeros(n)
for i in range(n):
    b[i] = sum(a[i])
print(gaussElimin(a,b))
input("Press return to exit"))

[ 1.  1.  1. ... 1.]
```

Gauss elimination is remarkably stable—there is no significant roundoff error in the solution of 500 equations. We also solved 1000 equations with the same result.

Problem 24

```
## problem2_1_24
```

```
from numpy import array
from gaussElimin import *

a = array([[ 5+1j,  5+2j, -5+3j,  6-3j],  \
           [ 5+2j,  7-2j,  8-1j, -1+3j],  \
           [-5+3j,  8-1j, -3-3j,  2+2j],  \
           [ 6-3j, -1+3j,  2+2j,  0+8j]])*1.0
b = array([15-35j, 2+10j, -2-34j, 8+14j])*1.0
x = gaussElimin(a,b)
print(x)
input("Press return to exit")
```

```
[ 2.00000000e+00 -1.77635684e-15j  -4.70677991e-16 -4.00000000e+00j
 -1.55169468e-15 +4.00000000e+00j   1.00000000e+00 +1.00000000e+00j]
```

After dropping the miniscule terms, the solution is

$$\mathbf{x} = \begin{bmatrix} 2 & -4i & 4i & 1+i \end{bmatrix}^T$$

Problem 25

```
##problem2_1_25
from numpy import array,zeros
from math import sin,cos,pi
from gaussElimin import *

theta = pi/4.0
g = 9.81
m = array([ 10.0, 4.0,  5.0, 6.0])
mu = array([0.25, 0.3,  0.2])
a = array([[ 1.0,  0.0,  0.0, m[0]],
           [-1.0,  1.0,  0.0, m[1]],
           [ 0.0, -1.0,  1.0, m[2]],
           [ 0.0,  0.0, -1.0, m[3]]])
b = zeros(4)
for i in range(3):
    b[i] = m[i]*g*(sin(theta) - mu[i]*cos(theta))
b[3] = -m[3]*g
print(gaussElimin(a,b))
input ("Press return to exit")
```

```
 35.89135719  48.86055656  68.54041454   1.61340242]
```

Hence $T_1 = 35.89$ N, $T_2 = 48.86$ N, $T_3 = 68.54$ N and $a = 1.6134$ m/s^2 ◀

PROBLEM SET 2.2

Problem 1

$$\mathbf{A} = \begin{bmatrix} 3 & -3 & 3 \\ -3 & 5 & 1 \\ 3 & 1 & 5 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 9 \\ -7 \\ 12 \end{bmatrix}$$

Noting that $\mathbf{A}$ is symmetric, we have two choices: (1) the Gauss elimination scheme that stores the multipliers in the *upper* portion of the matrix and results in $\mathbf{A} \to \left[\mathbf{0}\backslash\mathbf{D}\backslash\mathbf{L}^T\right]$ (see Example 2.10); or (2) the regular Gauss elimination that produces an upper triangular matrix $\mathbf{A} \to \mathbf{U}$. We choose the latter, which is somewhat simpler to implement in hand computation.

$$\begin{aligned} \text{row 2} &\leftarrow \text{row 2} + \text{row 1} \\ \text{row 3} &\leftarrow \text{row 3} - \text{row 1} \end{aligned}$$

$$\begin{bmatrix} 3 & -3 & 3 \\ 0 & 2 & 4 \\ 0 & 4 & 2 \end{bmatrix}$$

$$\text{row 3} \leftarrow \text{row 3} - 2 \times \text{row 2}$$

$$\mathbf{U} = \begin{bmatrix} 3 & -3 & 3 \\ 0 & 2 & 4 \\ 0 & 0 & -6 \end{bmatrix}$$

We obtain L^T by dividing each row of U by its diagonal element. Thus

$$\mathbf{L}^T = \begin{bmatrix} 1 & -1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Solution of $\mathbf{Ly} = \mathbf{b}$:

$$\begin{aligned} y_1 &= 9 \\ -9 + y_2 &= -7 \qquad y_2 = 2 \\ 9 + 2(2) + y_3 &= 12 \qquad y_3 = -1 \end{aligned}$$

Solution of $\mathbf{Ux} = \mathbf{y}$:

$$\begin{aligned} -6x_3 &= -1 \qquad x_3 = \frac{1}{6} \;\blacktriangleleft \\ 2x_2 + 4\left(\frac{1}{6}\right) &= 2 \qquad x_2 = \frac{2}{3} \;\blacktriangleleft \\ 3x_1 - 3\left(\frac{2}{3}\right) + 3\left[\frac{1}{6}\right] &= 9 \qquad x_1 = \frac{7}{2} \;\blacktriangleleft \end{aligned}$$

Problem 2

$$\mathbf{A} = \begin{bmatrix} 4 & 8 & 20 \\ 8 & 13 & 16 \\ 20 & 16 & -91 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 24 \\ 18 \\ -119 \end{bmatrix}$$

Since $\mathbf{A}$ is symmetric, we could employ the same Gauss elimination $\mathbf{A} \to \mathbf{U}$ that was used in Problem 1. However, we chose the form $\mathbf{A} \to \left[\mathbf{0}\backslash\mathbf{D}\backslash\mathbf{L}^T\right]$ obtained by storing the multipliers (shown enclosed in boxes) in the *upper* half of the matrix.

$$\begin{aligned} \text{row 2} &\leftarrow \text{row 2} - 2 \times \text{row 1} \\ \text{row 3} &\leftarrow \text{row 3} - 5 \times \text{row 1} \end{aligned}$$

$$\begin{bmatrix} 4 & \boxed{2} & \boxed{5} \\ 0 & -3 & -24 \\ 0 & -24 & -191 \end{bmatrix}$$

$$\text{row 3} \leftarrow \text{row 3} - 8 \times \text{row 2}$$

$$\left[\mathbf{0}\backslash\mathbf{D}\backslash\mathbf{L}^T\right] = \begin{bmatrix} 4 & \boxed{2} & \boxed{5} \\ 0 & -3 & \boxed{8} \\ 0 & 0 & 1 \end{bmatrix}$$

Therefore,

$$\mathbf{L}^T = \begin{bmatrix} 1 & 2 & 5 \\ 0 & 1 & 8 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 5 & 8 & 1 \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{U} = \mathbf{D}\mathbf{L}^T = \begin{bmatrix} 4 & 8 & 20 \\ 0 & -3 & -24 \\ 0 & 0 & 1 \end{bmatrix}$$

Solution of $\mathbf{Ly} = \mathbf{b}$:

$$\begin{aligned} y_1 &= 24 \\ 2(24) + y_2 &= 18 \qquad y_2 = -30 \\ 5(24) + 8(-30) + y_3 &= -119 \qquad y_3 = 1 \end{aligned}$$

Solution of $\mathbf{Ux} = \mathbf{y}$:

$$\begin{aligned} x_3 &= 1 \blacktriangleleft \\ -3x_2 - 24(1) &= -30 \qquad x_2 = 2 \blacktriangleleft \\ 4x_1 + 8(2) + 20(1) &= 24 \qquad x_1 = -3 \blacktriangleleft \end{aligned}$$

Problem 3

$$\mathbf{A} = \begin{bmatrix} 2 & -2 & 0 & 0 & 0 \\ -2 & 5 & -6 & 0 & 0 \\ 0 & -6 & 16 & 12 & 0 \\ 0 & 0 & 12 & 39 & -6 \\ 0 & 0 & 0 & -6 & 14 \end{bmatrix}$$

Noting that $\mathbf{A}$ is symmetric, we use the reduction $\mathbf{A} \to \left[\mathbf{0}\backslash\mathbf{D}\backslash\mathbf{L}^T\right]$ obtained by storing the multipliers (shown enclosed in boxes) in the *upper* half of the matrix during Gauss elimination.

$$\text{row 2} \leftarrow \text{row 2} - (-1) \times \text{row 1}$$

$$\begin{bmatrix} 2 & \boxed{-1} & 0 & 0 & 0 \\ 0 & 3 & -6 & 0 & 0 \\ 0 & -6 & 16 & 12 & 0 \\ 0 & 0 & 12 & 39 & -6 \\ 0 & 0 & 0 & -6 & 14 \end{bmatrix}$$

$$\text{row 3} \leftarrow \text{row 3} - (-2) \times \text{row 2}$$

$$\begin{bmatrix} 2 & \boxed{-1} & 0 & 0 & 0 \\ 0 & 3 & \boxed{-2} & 0 & 0 \\ 0 & 0 & 4 & 12 & 0 \\ 0 & 0 & 12 & 39 & -6 \\ 0 & 0 & 0 & -6 & 14 \end{bmatrix}$$

$$\text{row 4} \leftarrow \text{row 4} - 3 \times \text{row 3}$$

$$\begin{bmatrix} 2 & \boxed{-1} & 0 & 0 & 0 \\ 0 & 3 & \boxed{-2} & 0 & 0 \\ 0 & 0 & 4 & \boxed{3} & 0 \\ 0 & 0 & 0 & 3 & -6 \\ 0 & 0 & 0 & -6 & 14 \end{bmatrix}$$

$$\text{row 5} \leftarrow \text{row 5} - (-2) \times \text{row 4}$$

$$\left[\mathbf{0}\backslash\mathbf{D}\backslash\mathbf{L}^T\right] = \begin{bmatrix} 2 & \boxed{-1} & 0 & 0 & 0 \\ 0 & 3 & \boxed{-2} & 0 & 0 \\ 0 & 0 & 4 & \boxed{3} & 0 \\ 0 & 0 & 0 & 3 & \boxed{-2} \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Thus

$$\mathbf{D} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} \quad \blacktriangleleft \quad \mathbf{L}^T = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 \\ 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -2 & 1 & 0 & 0 \\ 0 & 0 & 3 & 1 & 0 \\ 0 & 0 & 0 & -2 & 1 \end{bmatrix} \blacktriangleleft$$

Problem 4

$$\mathbf{A} = \begin{bmatrix} 6 & 2 & 0 & 0 & 0 \\ -1 & 7 & 2 & 0 & 0 \\ 0 & -2 & 8 & 2 & 0 \\ 0 & 0 & 3 & 7 & -2 \\ 0 & 0 & 0 & 3 & 5 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 2 \\ -3 \\ 4 \\ -3 \\ 1 \end{bmatrix}$$

LU decomposition of $\mathbf{A}$:

$$\text{row } 2 \leftarrow \text{row } 2 - (-0.1667) \times \text{row } 1$$

$$\begin{bmatrix} 6 & 2 & 0 & 0 & 0 \\ \boxed{-0.1667} & 7.3333 & 2 & 0 & 0 \\ 0 & -2 & 8 & 2 & 0 \\ 0 & 0 & 3 & 7 & -2 \\ 0 & 0 & 0 & 3 & 5 \end{bmatrix}$$

$$\text{row } 3 \leftarrow \text{row } 3 - (-0.2727) \times \text{row } 2$$

$$\begin{bmatrix} 6 & 2 & 0 & 0 & 0 \\ \boxed{-0.1667} & 7.3333 & 2 & 0 & 0 \\ 0 & \boxed{-0.2727} & 8.5454 & 2 & 0 \\ 0 & 0 & 3 & 7 & -2 \\ 0 & 0 & 0 & 3 & 5 \end{bmatrix}$$

$$\text{row } 4 \leftarrow \text{row } 4 - 0.3511 \times \text{row } 3$$

$$\begin{bmatrix} 6 & 2 & 0 & 0 & 0 \\ \boxed{-0.1667} & 7.3333 & 2 & 0 & 0 \\ 0 & \boxed{-0.2727} & 8.5454 & 2 & 0 \\ 0 & 0 & \boxed{0.3511} & 6.2978 & -2 \\ 0 & 0 & 0 & 3 & 5 \end{bmatrix}$$

$$\text{row } 5 \leftarrow \text{row } 5 - 0.4764 \times \text{row } 4$$

$$[\mathbf{L}\backslash\mathbf{U}] = \begin{bmatrix} 6 & 2 & 0 & 0 & 0 \\ \boxed{-0.1667} & 7.3333 & 2 & 0 & 0 \\ 0 & \boxed{-0.2727} & 8.5454 & 2 & 0 \\ 0 & 0 & \boxed{0.3511} & 6.2978 & -2 \\ 0 & 0 & 0 & \boxed{0.4764} & 5.9528 \end{bmatrix}$$

Solution of $\mathbf{Ly} = \mathbf{b}$:

$$\begin{aligned}
y_1 &= 2 \\
-0.1667(2) + y_2 &= -3 \qquad y_2 = -2.6667 \\
-0.2727(-2.6667) + y_3 &= 4 \qquad y_3 = 3.2728 \\
0.3511(3.2728) + y_4 &= -3 \qquad y_4 = -4.1491 \\
0.4764(-4.1491) + y_5 &= 1 \qquad y_5 = 2.9766
\end{aligned}$$

Solution of $\mathbf{Ux} = \mathbf{y}$:

$$\begin{aligned}
5.9528x_5 &= 2.9766 \qquad x_5 = 0.5000 \blacktriangleleft \\
6.2978x_4 - 2(0.5000) &= -4.1491 \qquad x_4 = -0.5000 \blacktriangleleft \\
8.5454x_3 + 2(-0.5000) &= 3.2728 \qquad x_3 = 0.5000 \blacktriangleleft \\
7.3333x_2 + 2(0.5000) &= -2.6667 \qquad x_2 = -0.5000 \blacktriangleleft \\
6x_1 + 2(-0.5000) &= 2 \qquad x_1 = 0.5000 \blacktriangleleft
\end{aligned}$$

Problem 5

$$[\mathbf{A}|\mathbf{b}] = \begin{bmatrix} 4 & -2 & 1 & 2 \\ -2 & 1 & -1 & -1 \\ -2 & 3 & 6 & 0 \end{bmatrix} \qquad \mathbf{s} = \begin{bmatrix} 4 \\ 2 \\ 6 \end{bmatrix} \qquad \mathbf{r} = \begin{bmatrix} 1 \\ 1 \\ 1/3 \end{bmatrix}$$

No need to pivot here.

$$\begin{aligned}
\text{row 2} &\leftarrow \text{row 2} + \frac{1}{2} \times \text{row 1} \\
\text{row 3} &\leftarrow \text{row 3} + \frac{1}{2} \times \text{row 1}
\end{aligned}$$

$$\begin{bmatrix} 4 & -2 & 1 & 2 \\ 0 & 0 & -1/2 & 0 \\ 0 & 2 & 13/2 & 1 \end{bmatrix} \qquad \mathbf{s} = \begin{bmatrix} * \\ 1/2 \\ 13/2 \end{bmatrix} \qquad \mathbf{r} = \begin{bmatrix} * \\ 0 \\ 4/13 \end{bmatrix}$$

Exchanging rows 2 and 3 triangulizes the coefficient matrix:

$$\begin{bmatrix} 4 & -2 & 1 & 2 \\ 0 & 2 & 13/2 & 1 \\ 0 & 0 & -1/2 & 0 \end{bmatrix}$$

Back substitution:

$$\begin{aligned}
x_3 &= 0 \blacktriangleleft \\
2x_2 + \frac{13}{2}(0) &= 1 \qquad x_2 = \frac{1}{2} \blacktriangleleft \\
4x_1 - 2\left(\frac{1}{2}\right) + 0 &= 2 \qquad x_1 = \frac{3}{4} \blacktriangleleft
\end{aligned}$$

Problem 6

$$[\mathbf{A}|\mathbf{b}] = \begin{bmatrix} 2.34 & -4.10 & 1.78 & 0.02 \\ -1.98 & 3.47 & -2.22 & -0.73 \\ 2.36 & -15.17 & 6.81 & -6.63 \end{bmatrix} \qquad \mathbf{s} = \begin{bmatrix} 4.10 \\ 3.47 \\ 15.17 \end{bmatrix}$$

$$\mathbf{r} = \begin{bmatrix} 2.34/4.10 \\ 1.98/3.47 \\ 2.36/15.17 \end{bmatrix} = \begin{bmatrix} 0.5707 \\ 0.5706 \\ 0.1556 \end{bmatrix}$$

No need to pivot here.

$$\begin{aligned} \text{row 2} &\leftarrow \text{row 2} + (1.98/2.34) \times \text{row 1} \\ \text{row 3} &\leftarrow \text{row 3} - (2.36/2.34) \times \text{row 1} \end{aligned}$$

$$\begin{bmatrix} 2.34 & -4.10 & 1.78 & 0.02 \\ 0 & 0.0008 & -0.7138 & -0.7131 \\ 0 & -11.0350 & 5.0148 & -6.6502 \end{bmatrix}$$

Without computing $\mathbf{r}$, it is clear that row 3 must be the next pivot row. We do not physically interchange rows 2 and 3, but carry out the elimination "in place":

$$\text{row 2} \leftarrow \text{row 2} + (0.0008/11.0350) \times \text{row 3}$$

$$\begin{bmatrix} 2.34 & -4.10 & 1.78 & 0.02 \\ 0 & 0 & -0.7134 & -0.7136 \\ 0 & -11.0350 & 5.0148 & -6.6502 \end{bmatrix}$$

Back substitution:

$$\begin{aligned} -0.7134x_3 &= -0.7136 \qquad x_3 = 1.0003 \blacktriangleleft \\ -11.0350x_2 + 5.0148(1.0003) &= -6.6502 \qquad x_2 = 1.0572 \blacktriangleleft \\ 2.34x_1 - 4.10(1.0572) + 1.78(1.0003) &= 0.02 \qquad x_1 = 1.1000 \blacktriangleleft \end{aligned}$$

Problem 7

We do not physically interchange rows, but eliminate "in place".

$$[\mathbf{A}|\mathbf{b}] = \begin{bmatrix} 2 & -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 \end{bmatrix} \qquad \mathbf{r} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1/2 \end{bmatrix}$$

$$\text{row 4} \leftarrow \text{row 4} + \frac{1}{2} \times \text{row 1}$$

$$\begin{bmatrix} 2 & -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 3/2 & -1 & 0 & 1/2 \end{bmatrix} \qquad \mathbf{r} = \begin{bmatrix} * \\ 0 \\ 1/2 \\ 1 \end{bmatrix}$$

$$\text{row 3} \leftarrow \text{row 3} + \frac{2}{3} \times \text{row 4}$$

$$\begin{bmatrix} 2 & -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 4/3 & -1 & 1/3 \\ 0 & 3/2 & -1 & 0 & 1/2 \end{bmatrix} \qquad \mathbf{r} = \begin{bmatrix} * \\ 1 \\ 1 \\ * \end{bmatrix}$$

$$\text{row 2} \leftarrow \text{row 2} + \frac{3}{4} \times \text{row 3}$$

$$\begin{bmatrix} 2 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 4/3 & -1 & 1/3 \\ 0 & 3/2 & -1 & 0 & 1/2 \end{bmatrix}$$

Note that by rearranging rows, the coefficient matrix could be given an upper triangular form. There is no need for this rearrangement, since back substitution can be carried out just as easily on the matrix as it is:

$$\begin{aligned} \frac{1}{4}x_4 &= \frac{1}{4} & x_4 &= 1 \blacktriangleleft \\ \frac{4}{3}x_3 - 1 &= \frac{1}{3} & x_3 &= 1 \blacktriangleleft \\ \frac{3}{2}x_2 - 1 &= \frac{1}{2} & x_2 &= 1 \blacktriangleleft \\ 2x_1 - 1 &= 1 & x_1 &= 1 \blacktriangleleft \end{aligned}$$

Problem 8

We chose Gauss elimination with pivoting (pivoting is essential here due to the zero element in the top left corner of the coefficient matrix).

```
## problem2_2_8
from numpy import array
from gaussPivot import *

a = array([[ 0, 2, 5,-1], \
           [ 2, 1, 3, 0], \
           [-2,-1, 3, 1], \
           [ 3, 3,-1, 2]])*1.0
b = array([-3, 3,-2, 5])*1.0
```

```
print(''The solution is:\n'',gaussPivot(a,b))
input(''\nPress return to exit'')

The solution is:
[2.00000000e+00 -1.00000000e+00  3.60072332e-17  1.00000000e+00]
```

Thus $\mathbf{x} = \begin{bmatrix} 2 & -1 & 0 & 1 \end{bmatrix}^T$ ◄

Problem 9

As the coefficient matrix is clearly diagonally dominant, it would not benefit from pivoting. Hence we use the non-pivoting LU decomposition functions for tridiagonal matrices.

```
## problem2_2_9
from numpy import ones
from LUdecomp3 import *

n = 10
b = ones((n))*5.0
b[0] = 9.0
c = ones((n-1))*(-1.0)
d = ones((n))*(4.0)
e = ones((n-1))*(-1.0)
LUdecomp3(c,d,e)
print(''\nThe solution is:\n'',LUsolve3(c,d,e,b))
input(''\nPress return to exit'')

The solution is:
[ 2.90191936  2.60767745  2.52879042  2.50748425  2.50114659
  2.4971021   2.48726181  2.45194513  2.3205187   1.83012968]
```

Problem 10

Unless there are obvious reasons to do otherwise, play it safe by using pivoting. Here we chose LU decomposition with pivoting.

```
## problem2_2_10
from LUpivot import *
from numpy import array
```

```
a = array([[1.3174, 2.7250, 2.7250, 1.7181],  \
           [0.4002, 0.8278, 1.2272, 2.5322],  \
           [0.8218, 1.5608, 0.3629, 2.9210],  \
           [1.9664, 2.0011, 0.6532, 1.9945]])
b = array([8.4855, 4.9874, 5.6665, 6.6152])
a,seq = LUdecomp(a)
print(''The solution is:\n'',LUsolve(a,b,seq))
input(''Press return to exit'')

The solution is:
[ 1.  1.  1.  1.]
```

Problem 11

We use LU decomposition with pivoting:

```
## problem2_2_11
from LUpivot import *
from numpy import array

a = array([[10, -2,-1, 2, 3,  1,-4, 7],  \
           [ 5, 11, 3,10,-3,  3, 3,-4],  \
           [ 7, 12, 1, 5, 3,-12, 2, 3],  \
           [ 8,  7,-2, 1, 3,  2, 2, 4],  \
           [ 2,-15,-1, 1, 4, -1, 8, 3],  \
           [ 4,  2, 9, 1,12, -1, 4, 1],  \
           [-1,  4,-7,-1, 1,  1,-1,-3],  \
           [-1,  3, 4, 1, 3, -4, 7, 6]],float)
b = array([0,12,-5,3,-25,-26,9,-7],float)
a,seq = LUdecomp(a)
print("The solution is:\n",LUsolve(a,b,seq))
input("Press return to exit")

The solution is:
[-1.  1. -1.  1. -1.  1. -1.  1.]
```

Problem 12

As the coefficient matrix is symmetric and diagonally dominant, Choleski's decomposition is the most efficient method of solution.

```
## problem2_2_12
from numpy import array,zeros
from choleski import *

k = array([1,2,1,1,2],float)
W = array([2,1,2],float)
a = zeros((3,3))
a[0,0] = k[0] + k[1] + k[2] + k[4]
a[0,1] = a[1,0] = -k[2]
a[0,2] = a[2,0] = -k[4]
a[1,1] = k[2] + k[3]
a[1,2] = a[2,1] = -k[3]
a[2,2] = k[3] + k[4]
L = choleski(a)
x = choleskiSol(L,W)
print("Displacements are (in units of W/k):\n",x)
input("Press return to exit")
```

```
Displacements are (in units of W/k):
[ 1.66666667  2.66666667  2.66666667]
```

Problem 13

Since the coefficient matrix is symmetric and diagonally dominant, we use Choleski's decomposition.

```
## problem2_2_13
from numpy import array,ones
from choleski import *

a = array([[ 2,-1, 0, 0, 0],  \
           [-1, 4,-1, 0, 0],  \
           [ 0,-1, 4,-1,-2],  \
           [ 0, 0,-1, 2,-1],\
           [ 0, 0,-2,-1, 3]],float)
W = ones(5)
L = choleski(a)
```

```
x = choleskiSol(L,W)
print("Displacements are (in units of W/k):\n",x)
input("Press return to exit")
```

```
Displacements are (in units of W/k):
[ 1.4  1.8  4.8  5.6  5.4]
```

Problem 14

Here the coefficient matrix is diagonally dominant, so that pivoting is not needed. We chose LU decomposition as the method of solution.

```
## problem2_2_14
from numpy import array
from LUdecomp import *
k = array([[27.58, 7.004, -7.004, 0.0, 0.0 ], \
 [ 7.004, 29.57, -5.253, 0.0, -24.32 ], \
 [-7.004, -5.253, 29.57, 0.0, 0.0 ], \
 [ 0.0, 0.0, 0.0, 27.58, -7.004], \
 [ 0.0, -24.32, 0.0, -7.004, 29.57]])
p = array([0.0, 0.0, 0.0, 0.0, -45.0])/1000.0 # Convert to MN
LUdecomp(k)
print(''The displacements are (in meters):\n'',LUsolve(k,p))
input(''Press return to exit'')
```

```
The displacements are (in meters):
[ 0.00144044 -0.00648249 -0.00081041 -0.00185182 -0.00729199]
```

Problem 15

(a)

We use LU decomposition with pivoting (pivoting is a must here):

```
## problem2_2_15
from numpy import array
from LUpivot import *
from math import sqrt
c = 1.0/sqrt(2.0)
a = array([[-1.0,  1.0, -c,   0.0, 0.0, 0.0],  \
```

```
          [ 0.0,  0.0,  c,   1.0, 0.0, 0.0],  \
          [ 0.0, -1.0, 0.0,  0.0, -c,  0.0],  \
          [ 0.0,  0.0, 0.0,  0.0,  c,  0.0],  \
          [ 0.0,  0.0, 0.0,  0.0,  c,  1.0],  \
          [ 0.0,  0.0, 0.0, -1.0, -c,  0.0]])
b = array([0.0,  18.0, 0.0, 12.0, 0.0, 0.0])
a,seq = LUdecomp(a)
print(''The forces are (in kN):\n'',LUsolve(a,b,seq))
input(''Press return to exit'')

The forces are (in kN):
[-42.   -12.   42.42640687 -12.   16.97056275 -12.   ]
```

(b)

After rearranging rows, we get

$$\begin{bmatrix} -1 & 1 & -1/\sqrt{2} & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1/\sqrt{2} & 0 \\ 0 & 0 & 1/\sqrt{2} & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1/\sqrt{2} & 0 \\ 0 & 0 & 0 & 0 & 1/\sqrt{2} & 1 \\ 0 & 0 & 0 & 0 & 1/\sqrt{2} & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 18 \\ 0 \\ 0 \\ 12 \end{bmatrix}$$

Interchanging columns 5 and 6 yields

$$\begin{bmatrix} -1 & 1 & -1/\sqrt{2} & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & -1/\sqrt{2} \\ 0 & 0 & 1/\sqrt{2} & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1/\sqrt{2} \\ 0 & 0 & 0 & 0 & 1 & 1/\sqrt{2} \\ 0 & 0 & 0 & 0 & 0 & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_6 \\ P_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 18 \\ 0 \\ 0 \\ 12 \end{bmatrix}$$

Back substitution:

$$\begin{aligned} \frac{1}{\sqrt{2}}P_5 &= 12 & P_5 &= 16.971 \text{ kN} \blacktriangleleft \\ P_6 + \frac{1}{\sqrt{2}}(16.971) &= 0 & P_6 &= -12.000 \text{ kN} \blacktriangleleft \\ -P_4 - \frac{1}{\sqrt{2}}(16.971) &= 0 & P_4 &= -12.000 \text{ kN} \blacktriangleleft \\ \frac{1}{\sqrt{2}}P_3 + (-12.000) &= 18 & P_3 &= 42.426 \text{ kN} \blacktriangleleft \\ -P_2 - \frac{1}{\sqrt{2}}(16.971) &= 0 & P_2 &= -12.000 \text{ kN} \blacktriangleleft \\ -P_1 + (-12.000) - \frac{1}{\sqrt{2}}(42.426) &= 0 & P_1 &= -42.000 \text{ kN} \blacktriangleleft \end{aligned}$$

Problem 16

We could rearrange the rows and columns of the coefficient matrix so as to arrive at an upper triangular matrix, as was done in Problem 15. This would definitely facilitate hand computations, but is hardly worth the effort when a computer is used. Therefore, we solve the equations as they are, using Gauss elimination with pivoting (pivoting is essential here). The following program prompts for θ:

```
## problem2_2_16
from numpy import array
from gaussPivot import *
from math import sin,cos,pi

theta = eval(input(''theta in degrees ==> ''))
theta = theta*pi/180.0
c = cos(theta)
s = sin(theta)
a = array([[ c,  1.0,  0.0,   0.0, 0.0],  \
           [0.0,   s,  0.0,   0.0, 1.0],  \
           [0.0,  0.0, 2.0*s, 0.0, 0.0],  \
           [0.0, - c,   c,    1.0, 0.0],  \
           [0.0,   s,   s,    0.0, 0.0]])
b = array([0.0, 0.0, 1.0, 0.0, 0.0])
print(''The forces are:\n'',gaussPivot(a,b))
input(''Press return to exit'')
```

```
theta in degrees ==> 53.0
The forces are:
[ 1.04029944 -0.62606783  0.62606783 -0.75355405  0.5        ]
```

Problem 17

The equations are

$$\begin{bmatrix} 50+R & -R & -30 \\ -R & 65+R & -15 \\ -30 & -15 & 45 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 120 \end{bmatrix}$$

The coefficient matrix is diagonally dominant, so that pivoting is unnecessary. Gauss elimination was chosen for the method of solution.

```
## problem2_2_17
from numpy import array
from gaussElimin import *
R = [5.0, 10.0, 20.0]
for r in R:
    a = array([[ 50.0 + r,       -r, -30.0],  \
               [       -r, 65.0 + r, -15.0],  \
               [    -30.0,    -15.0,  45.0]])
    b = array([0.0, 0.0, 120.0])
    print("\nR =",r,"ohms")
    print("The currents are (in amps):\n",gaussElimin(a,b))
input("Press return to exit")
```

```
R = 5.0 ohms
The currents are (in amps):
[ 2.82926829  1.26829268  4.97560976]

R = 10.0 ohms
The currents are (in amps):
[ 2.66666667  1.33333333  4.88888889]

R = 20.0 ohms
The currents are (in amps):
[ 2.4516129   1.41935484  4.77419355]
```

Problem18

Kirchoff's equations for the 4 loops are

$$\begin{aligned}
50(i_1 - i_2) + 30(i_1 - i_3) &= -120 \\
50(i_2 - i_1) + 15i_2 + 25(i_2 - i_4) + 10(i_2 - i_3) &= 0 \\
30(i_3 - i_1) + 10(i_3 - i_2) + 20(i_3 - i_4) + 5i_3 &= 0 \\
20(i_4 - i_3) + 25(i_4 - i_2) + (10 + 30 + 15)\, i_4 &= 0
\end{aligned}$$

or

$$\begin{bmatrix} 80 & -50 & -30 & 0 \\ -50 & 100 & -10 & -25 \\ -30 & -10 & 65 & -20 \\ 0 & -25 & -20 & 100 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix} = \begin{bmatrix} -120 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Since the coefficient matrix is diagonally dominant, Gauss elimination without pivoting can be used safely:

```
## problem2_2_18
from numpy import array
from gaussElimin import *

a = array([[ 80, -50, -30,   0],  \
           [-50, 100, -10, -25],  \
           [-30, -10,  65, -20],  \
           [  0, -25, -20, 100]])*1.0
b = array([-120, 0, 0, 0])*1.0
print(''The currents are (in amps):\n'',gaussElimin(a,b))
input(''Press return to exit'')
```

```
The loop currents are (in amps):
[-4.18239492 -2.66455194 -2.71213323 -1.20856463]
```

Problem 19

Pivoting is strongly advised here. The following program, which prompts for n, uses Gauss elimination with pivoting.

```
## problem2_2_19
from numpy import zeros
from gaussPivot import *

n = eval(input(''\nnumber of equations ==> ''))
a = zeros((n,n))
b = zeros(n)
for i in range(n):
    for j in range(n):
        a[i,j] = a[j,i] =(i + j)**2
        b[i] = b[i] + a[i,j]
print(''The solution is:\n'',gaussPivot(a,b))
input(''Press return to exit'')
```

```
number of equations ==> 2
The solution is:
[ 1.  1.]

number of equations ==> 3
The solution is:
[ 1.  1.  1.]
```

```
number of equations ==> 4
The solution is:
Matrix is singular
```

The determinant of the coefficient becomes zero for $n \geq 4$. Although a solution of the equations is $x = \begin{bmatrix} 1 & 1 & \cdots \end{bmatrix}^T$ for all n, this solution is not unique if $n \geq 4$.

Problem 20

We apply the conservation equation

$$\Sigma\,(Qc)_{\text{in}} + \Sigma\,(Qc)_{\text{out}} = 0$$

to each vessel, where Q is the flow rate of water, and c is the concentration. The results are

$$\begin{array}{ll} 1 & -8c_1 + 4c_2 + 4(20) = 0 \\ 2 & 8c_1 - 10c_2 + 2c_3 = 0 \\ 3 & 6c_2 - 11c_3 + 5c_4 = 0 \\ 4 & 3c_3 - 7c_4 + 4c_5 = 0 \\ 5 & 2c_4 - 4c_5 + 2(15) = 0 \end{array}$$

Since these equations are tridiagonal, we solve them with `LUdecomp3` and `LUsolve3`:

```
## problem2_2_20
import numpy as np
from LUdecomp3 import *
c = np.array([8,6,3,2],float)
d = np.array([-8,-10,-11,-7,-4],float)
e = np.array([4,2,5,4],float)
b = np.array([-80,0,0,0,-30],float)
c,d,e = LUdecomp3(c,d,e)
print('conc =',LUsolve3(c,d,e,b))
```

The solution for the concentrations is (units are mg/m^3):

```
conc = [19.72222222 19.44444444 18.33333333  17.    16.   ]
```

Problem 21

The conservation equations

$$\Sigma\,(Qc)_{\text{in}} + \Sigma\,(Qc)_{\text{out}} = 0$$

for the four tanks are

$$
\begin{array}{lr}
1 & -6c_1 + 4c_2 + 2(25) = 0 \\
2 & -7c_2 + 3c_3 + 4c_4 = 0 \\
3 & 4c_1 - 4c_3 = 0 \\
4 & 2c_1 + c_3 - 4c_4 + 1(50) = 0
\end{array}
$$

The coefficient matrix is diagonally dominant, so that pivoting is not needed. The following program uses Gauss eliminetion.

```
## problem2_2_21
import numpy as np
from gaussElimin import *
A = np.array([[-6, 4, 0, 0],  \
              [ 0,-7, 3, 4],  \
              [ 4, 0,-4, 0],  \
              [ 2, 0, 1,-4]],float)
b = np.array([-50,0,0,-50],float)
print('conc =',gaussElimin(A,b))
input("\nPress return to exit")
```

The concentrations are (in mg/m^3)

```
conc = [30.55555556 33.33333333 30.55555556 35.41666667]
```

Problem 22

The coefficient matrix is symmetric and pentadiagonal. Therefore, the functions in the module `LUdecomp5` will be used in the solution.

```
## problem2_2_22
from LUdecomp5 import *
from numpy import ones

n = 10                     # Number of equations
d = ones(n)*6.0            # Principal diagonal
d[0] = 7.0; d[n-1] = 7.0
e = ones(n-1)*(-4.0)       # Second diagonal
f = ones(n-2)              # Third diagonal
b = ones(n)                # Right-hand side
d,e,f = LUdecomp5(d,e,f)   # Decompose
x = LUsolve5(d,e,f,b)      # Solve
print('x =\n',x)
input('\nPress return to exit')
```

```
x =
[  5.  15.  26.  35.  40.  40.  35.  26.  15.   5.]
```

Problem 23

```
## problem2_2_23
from numpy import array,zeros,ones
from LUdecomp3 import *

n = 10
d = ones((n),complex)*2.0
d[n-1] = 1.0
c = -ones((n-1),complex)*1.0j
e = c.copy()
b = zeros((n),complex)
b[0] = 100.0 + 100.0j

c,d,e = LUdecomp3(c,d,e)
x = LUsolve3(c,d,e,b)
print('     real         imag')
for i in range(n):
    print('{0:13.9f} {1:13.9f}'.format(x[i].real,x[i].imag))
input("Press return to exit")
```

```
    real         imag
 41.421349985  41.421349985
-17.157300030  17.157300030
 -7.106749926  -7.106749926
  2.943800178  -2.943800178
  1.219149569   1.219149569
 -0.505501041   0.505501041
 -0.208147487  -0.208147487
  0.089206066  -0.089206066
  0.029735355   0.029735355
 -0.029735355   0.029735355
```

PROBLEM SET 2.3

Problem 1

The inverse of $\mathbf{B}$ is obtained by interchanging the first two *columns* of $\mathbf{A}^{-1}$:

$$\mathbf{B}^{-1} = \begin{bmatrix} 0 & 0.5 & 0.25 \\ 0.4 & 0.3 & 0.45 \\ 0.2 & -0.1 & -9.16 \end{bmatrix} \quad \blacktriangleleft$$

Problem 2

$$\mathbf{A} = \begin{bmatrix} 2 & 4 & 3 \\ 0 & 6 & 5 \\ 0 & 0 & 2 \end{bmatrix}$$

Solve $\mathbf{AX} = \mathbf{I}$ by back substitution, one column of $\mathbf{X}$ at a time.
Solution of $\mathbf{Ax} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$(column 1 of $\mathbf{X}$):

$$\begin{aligned} 2x_3 &= 0 & x_3 &= 0 \\ 6x_2 + 5(0) &= 0 & x_2 &= 0 \\ 2x_1 + 4(0) + 3(0) &= 1 & x_1 &= \frac{1}{2} \end{aligned}$$

Solution of $\mathbf{Ax} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$(column 2 of $\mathbf{X}$):

$$\begin{aligned} 2x_3 &= 0 & x_3 &= 0 \\ 6x_2 + 5(0) &= 1 & x_2 &= \frac{1}{6} \\ 2x_1 + 4\left(\frac{1}{6}\right) + 3(0) &= 0 & x_1 &= -\frac{1}{3} \end{aligned}$$

Solution of $\mathbf{Ax} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$(column 3 of $\mathbf{X}$):

$$\begin{aligned} 2x_3 &= 1 & x_3 &= \frac{1}{2} \\ 6x_2 + 5\left(\frac{1}{2}\right) &= 0 & x_2 &= -\frac{5}{12} \\ 2x_1 + 4\left(-\frac{5}{12}\right) + 3\left(\frac{1}{2}\right) &= 0 & x_1 &= \frac{1}{12} \end{aligned}$$

$$\mathbf{A}^{-1} = \mathbf{X} = \begin{bmatrix} 1/2 & -1/3 & 1/12 \\ 0 & 1/6 & -5/12 \\ 0 & 0 & 1/2 \end{bmatrix} \blacktriangleleft$$

$$\mathbf{B} = \begin{bmatrix} 2 & 0 & 0 \\ 3 & 4 & 0 \\ 4 & 5 & 6 \end{bmatrix}$$

Solve $\mathbf{BX} = \mathbf{I}$ by forward substitution, one column of $\mathbf{X}$ at a time. Solution of $\mathbf{Bx} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$(column 1 of $\mathbf{X}$):

$$\begin{aligned} 2x_1 &= 1 \qquad x_1 = \frac{1}{2} \\ 3\left(\frac{1}{2}\right) + 4x_2 &= 0 \qquad x_2 = -\frac{3}{8} \\ 4\left(\frac{1}{2}\right) + 5\left(-\frac{3}{8}\right) + 6x_3 &= 0 \qquad x_3 = -\frac{1}{48} \end{aligned}$$

Solution of $\mathbf{Bx} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$(column 2 of $\mathbf{X}$):

$$\begin{aligned} 2x_1 &= 0 \qquad x_1 = 0 \\ 3(0) + 4x_2 &= 1 \qquad x_2 = \frac{1}{4} \\ 4(0) + 5\left(\frac{1}{4}\right) + 6x_3 &= 0 \qquad x_3 = -\frac{5}{24} \end{aligned}$$

Solution of $\mathbf{Bx} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$(column 3 of $\mathbf{X}$):

$$\begin{aligned} 2x_1 &= 0 \qquad x_1 = 0 \\ 3(0) + 3x_2 &= 0 \qquad x_2 = 0 \\ 4(0) + 5(0) + 6x_3 &= 1 \qquad x_3 = \frac{1}{6} \end{aligned}$$

$$\mathbf{B}^{-1} = \mathbf{X} = \begin{bmatrix} 1/2 & 0 & 0 \\ -3/8 & 1/4 & 0 \\ -1/48 & -5/24 & 1/6 \end{bmatrix} \blacktriangleleft$$

Problem 3

$$\mathbf{A} = \begin{bmatrix} 1 & 1/2 & 1/4 & 1/8 \\ 0 & 1 & 1/3 & 1/9 \\ 0 & 0 & 1 & 1/4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solve $\mathbf{AX} = \mathbf{I}$ by back substitution, one column of $\mathbf{X}$ at a time.
Solution of $\mathbf{Ax} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T$(column 1 of $\mathbf{X}$):

$$\begin{aligned}
x_4 &= 0 \\
x_3 + \frac{1}{4}(0) &= 0 \qquad x_3 = 0 \\
x_2 + \frac{1}{3}(0) + \frac{1}{9}(0) &= 0 \qquad x_2 = 0 \\
x_1 + \frac{1}{2}(0) + \frac{1}{4}(0) + \frac{1}{8}(0) &= 1 \qquad x_1 = 1
\end{aligned}$$

Solution of $\mathbf{Ax} = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}^T$(column 2 of $\mathbf{X}$):

$$\begin{aligned}
x_4 &= 0 \\
x_3 + \frac{1}{4}(0) &= 0 \qquad x_3 = 0 \\
x_2 + \frac{1}{3}(0) + \frac{1}{9}(0) &= 1 \qquad x_2 = 1 \\
x_1 + \frac{1}{2}(1) + \frac{1}{4}(0) + \frac{1}{8}(0) &= 0 \qquad x_1 = -\frac{1}{2}
\end{aligned}$$

Solution of $\mathbf{Ax} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^T$(column 3 of $\mathbf{X}$):

$$\begin{aligned}
x_4 &= 0 \\
x_3 + \frac{1}{4}(0) &= 1 \qquad x_3 = 1 \\
x_2 + \frac{1}{3}(1) + \frac{1}{9}(0) &= 0 \qquad x_2 = -\frac{1}{3} \\
x_1 + \frac{1}{2}\left(-\frac{1}{3}\right) + \frac{1}{4}(1) + \frac{1}{8}(0) &= 0 \qquad x_1 = -\frac{1}{12}
\end{aligned}$$

Solution of $\mathbf{Ax} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$(column 4 of $\mathbf{X}$):

$$\begin{aligned}
x_4 &= 1 \\
x_3 + \frac{1}{4}(1) &= 1 \qquad x_3 = -\frac{1}{4} \\
x_2 + \frac{1}{3}\left(-\frac{1}{4}\right) + \frac{1}{9}(1) &= 0 \qquad x_2 = -\frac{1}{36} \\
x_1 + \frac{1}{2}\left(-\frac{1}{36}\right) + \frac{1}{4}\left(-\frac{1}{4}\right) + \frac{1}{8}(1) &= 0 \qquad x_1 = -\frac{7}{144}
\end{aligned}$$

$$\mathbf{A}^{-1} = \mathbf{X} = \begin{bmatrix} 1 & -1/2 & -1/12 & -7/144 \\ 0 & 1 & -1/3 & -1/36 \\ 0 & 0 & 1 & -1/4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \blacktriangleleft$$

Problem 4

(a) We solve $\mathbf{AX} = \mathbf{I}$ by Gauss elimination (LU decomposition could also be used, but it takes more space in hand computation).
The augmented coefficient matrix is

$$(\mathbf{A}|\mathbf{I}) = \begin{bmatrix} 1 & 2 & 4 & 1 & 0 & 0 \\ 1 & 3 & 9 & 0 & 1 & 0 \\ 1 & 4 & 16 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} \text{row 2} &\leftarrow \text{row 2} - \text{row 1} \\ \text{row 3} &\leftarrow \text{row 3} - \text{row 1} \end{aligned}$$

$$\begin{bmatrix} 1 & 2 & 4 & 1 & 0 & 0 \\ 0 & 1 & 5 & -1 & 1 & 0 \\ 0 & 2 & 12 & -1 & 0 & 1 \end{bmatrix}$$

$$\text{row 3} \leftarrow \text{row 3} - 2 \times \text{row 2}$$

$$[\mathbf{U}|\mathbf{Y}] = \begin{bmatrix} 1 & 2 & 4 & 1 & 0 & 0 \\ 0 & 1 & 5 & -1 & 1 & 0 \\ 0 & 0 & 2 & 1 & -2 & 1 \end{bmatrix}$$

Solution of $\mathbf{Ux} = \begin{bmatrix} 1 & -1 & -1 \end{bmatrix}^T$(column 1 of $\mathbf{X}$):

$$\begin{aligned} 2x_3 &= 1 & x_3 &= 0.5 \\ x_2 + 5(0.5) &= -1 & x_2 &= -3.5 \\ x_1 + 2(-3.5) + 4(0.5) &= 1 & x_1 &= 6 \end{aligned}$$

Solution of $\mathbf{Ux} = \begin{bmatrix} 0 & 1 & -2 \end{bmatrix}^T$(column 2 of $\mathbf{X}$):

$$\begin{aligned} 2x_3 &= -2 & x_3 &= -1 \\ x_2 + 5(-1) &= 1 & x_2 &= 6 \\ x_1 + 2(6) + 4(-1) &= 0 & x_1 &= -8 \end{aligned}$$

Solution of $\mathbf{Ux} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$(column 1 of $\mathbf{X}$):

$$\begin{aligned} 2x_3 &= 1 & x_3 &= 0.5 \\ x_2 + 5(0.5) &= 0 & x_2 &= -2.5 \\ x_1 + 2(-2.5) + 4(0.5) &= 0 & x_1 &= 3 \end{aligned}$$

$$\mathbf{A}^{-1} = \mathbf{X} = \begin{bmatrix} 6.0 & -8.0 & 3.0 \\ -3.5 & 6.0 & -2.5 \\ 0.5 & -1.0 & 0.5 \end{bmatrix} \blacktriangleleft$$

(b) Use Gauss elimination. The augmented coefficient matrix is

$$[\mathbf{B}|\mathbf{I}] = \left[\begin{array}{rrrrrr} 4 & -1 & 0 & 1 & 0 & 0 \\ -1 & 4 & -1 & 0 & 1 & 0 \\ 0 & -1 & 4 & 0 & 0 & 1 \end{array}\right]$$

$$\text{row } 2 \leftarrow \text{row } 2 + 0.25 \times \text{row } 1$$

$$\left[\begin{array}{rrrrrr} 4 & -1 & 0 & 1 & 0 & 0 \\ 0 & 3.75 & -1 & 0.25 & 1 & 0 \\ 0 & -1 & 4 & 0 & 0 & 1 \end{array}\right]$$

$$\text{row } 3 \leftarrow \text{row } 3 + \frac{1}{3.75} \times \text{row } 2$$

$$[\mathbf{U}|\mathbf{Y}] = \left[\begin{array}{rrrrrr} 4 & -1 & 0 & 1 & 0 & 0 \\ 0 & 3.75 & -1 & 0.25 & 1 & 0 \\ 0 & 0 & 3.7333 & 0.06667 & 0.2667 & 1 \end{array}\right]$$

Solution of $\mathbf{Ux} = \left[\begin{array}{ccc} 0 & 0 & 1 \end{array}\right]^T$(column 1 of $\mathbf{X}$):

$$\begin{aligned} 3.7333x_3 &= 1 & x_3 &= 0.267\,9 \\ 3.75x_2 - 0.2679 &= 0 & x_2 &= 0.071\,43 \\ 4x_1 - 0.0714 &= 0 & x_1 &= 0.017\,86 \end{aligned}$$

Solution of $\mathbf{Ux} = \left[\begin{array}{ccc} 0 & 1 & 0.2667 \end{array}\right]^T$ (column 2 of $\mathbf{X}$):

$$\begin{aligned} 3.7333x_3 &= 0.2667 & x_3 &= 0.071\,43 \\ 3.75x_2 - 0.07134 &= 1 & x_2 &= 0.2857 \\ 4x_1 - 0.2857 &= 0 & x_1 &= 0.071\,43 \end{aligned}$$

Solution of $\mathbf{Ux} = \left[\begin{array}{ccc} 1 & 0.25 & 0.06667 \end{array}\right]^T$ (gives column 1 of $\mathbf{X}$):

$$\begin{aligned} 3.7333x_3 &= 0.06667 & x_3 &= 0.017\,86 \\ 3.75x_2 - 0.01786 &= 0.25 & x_2 &= 0.07143 \\ 4x_1 - 0.07143 &= 1 & x_1 &= 0.2679 \end{aligned}$$

$$\mathbf{B}^{-1} = \mathbf{X} = \left[\begin{array}{ccc} 0.2679 & 0.071\,4 & 0.017\,9 \\ 0.0714 & 0.2857 & 0.071\,4 \\ 0.017\,9 & 0.071\,4 & 0.267\,9 \end{array}\right] \quad \blacktriangleleft$$

Problem 5

Solve $\mathbf{AX} = \mathbf{I}$ by Gauss elimination. The augmented coefficient matrix is

$$[\mathbf{A}|\mathbf{I}] = \begin{bmatrix} 4 & -2 & 1 & 1 & 0 & 0 \\ -2 & 1 & -1 & 0 & 1 & 0 \\ 1 & -2 & 4 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} \text{row 2} &\leftarrow \text{row 2} + 0.5 \times \text{row 1} \\ \text{row 3} &\leftarrow \text{row 3} - 0.25 \times \text{row 1} \end{aligned}$$

$$\begin{bmatrix} 4 & -2 & 1 & 1 & 0 & 0 \\ 0 & 0 & -0.5 & 0.5 & 1 & 0 \\ 0 & -1.5 & 3.75 & -0.25 & 0 & 1 \end{bmatrix}$$

This completes the elimination stage (by switching rows 2 and 3, the coefficient matrix would have upper triangular form).
Solving $\mathbf{Ux} = \begin{bmatrix} 1 & 0.5 & -0.25 \end{bmatrix}^T$ (column 1 of $\mathbf{X}$):

$$\begin{aligned} -0.5x_3 &= 0.5 \qquad x_3 = -1 \\ -1.5x_2 + 3.75(-1) &= -0.25 \qquad x_2 = -2.3333 \\ 4x_1 - 2(-2.3333) + (-1) &= 1 \qquad x_1 = -0.6667 \end{aligned}$$

Solving $\mathbf{Ux} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$ (column 2 of $\mathbf{X}$):

$$\begin{aligned} -0.5x_3 &= 1 \qquad x_3 = -2 \\ -1.5x_2 + 3.75(-2) &= 0 \qquad x_2 = -5 \\ 4x_1 - 2(-5) + (-2) &= 0 \qquad x_1 = -2 \end{aligned}$$

Solving $\mathbf{Ux} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ (column 3 of $\mathbf{X}$):

$$\begin{aligned} -0.5x_3 &= 0 \qquad x_3 = 0 \\ -1.5x_2 + 3.75(0) &= 1 \qquad x_2 = -0.6667 \\ 4x_1 - 2(-0.6667) + 0 &= 0 \qquad x_1 = -0.3333 \end{aligned}$$

$$\mathbf{A}^{-1} = \mathbf{X} = \begin{bmatrix} -0.6667 & -2 & -0.3333 \\ -2.3333 & -5 & -0.6667 \\ -1 & -2 & 0 \end{bmatrix} \quad \blacktriangleleft$$

Problem 6

The following program is a modification of the program in Example 2.13. It uses LU decomposition with pivoting and computes $|\mathbf{A}|$ in addition to $\mathbf{A}^{-1}$(the determinant is helpful in gaging the conditioning of the matrix).

```
## problem2_3_6a
from numpy import array,identity,prod,diagonal
from LUpivot import *

def matInv(a):
    n = len(a[0])
    aInv = identity(n)
    a,seq = LUdecomp(a)
    det = abs(prod(diagonal(a)))
    for i in range(n):
        aInv[:,i] = LUsolve(a,aInv[:,i],seq)
    return aInv,det

a = array([[ 5, -3, -1,  0], \
           [-2,  1,  1,  1], \
           [ 3, -5,  1,  2], \
           [ 0,  8, -4, -3]],float)
aInv,det = matInv(a)
print("Inverse:\n",aInv)
print("Absolute value of determinant =",det)
input("\nPress return to exit")
```

```
Inverse:
[[ 1.125   1.     -0.875  -0.25  ]
 [ 0.8125  1.     -0.6875 -0.125 ]
 [ 2.1875  2.     -2.3125 -0.875 ]
 [-0.75    0.      1.25    0.5   ]]
Absolute value of determinant = 16.0
```

The program to invert $\mathbf{B}$ uses LU decomposition for tridiagonal matrices. Since $\mathbf{B}$ is diagonally dominant, there is no need to compute its determinant.

```
## problem2_3_6b
from numpy import identity,ones
from LUdecomp3 import *

def matInv(c,d,e):
    n = len(d)
    bInv = identity(n)
```

```
    c,d,e = LUdecomp3(c,d,e)
    for i in range(n):
        bInv[:,i] = LUsolve3(c,d,e,bInv[:,i])
    return bInv

c = ones((3))*(-1.0)
d = ones((4))*4.0
e = c.copy()
bInv = matInv(c,d,e)
print(''Inverse:\n'',bInv)
input(''\nPress return to exit'')
```

```
Inverse:
[[ 0.26794258  0.07177033  0.01913876  0.00478469]
 [ 0.07177033  0.28708134  0.07655502  0.01913876]
 [ 0.01913876  0.07655502  0.28708134  0.07177033]
 [ 0.00478469  0.01913876  0.07177033  0.26794258]]
```

Problem 7

We used `program2_3_6a` in Problem 6 (only the matrix was retyped) with the following result:

```
Inverse:
[[-2.22649869e+15 -2.22649869e+15 -2.22649869e+15 -4.45299738e+15
    4.45299738e+15]
 [-1.05758688e+16 -1.05758688e+16 -1.05758688e+16 -2.11517376e+16
   2.11517376e+16]
 [-1.77107851e+15 -1.77107851e+15 -1.77107851e+15 -3.54215701e+15
   3.54215701e+15]
 [ 0.00000000e+00  0.00000000e+00  1.00000000e+00  1.00000000e+00
  -1.00000000e+00]
 [ 4.50359963e+15  4.50359963e+15  4.50359963e+15  9.00719925e+15
  -9.00719925e+15]]
Absolute value of determinant = 1.97619698383e-014
```

Because the determinant is very small, the matrix is ill-conditioned, so that computed value of the inverse is completely unreliable.

Problem 8

As **K** exhibits diagonal dominance, there is no need for pivoting. Here we used LU decomposition to invert **K**.

```
## problem2_3_8
from numpy import array,identity,dot
from LUdecomp import *

def matInv(k):
    n = len(k[0])
    kInv = identity(n)
    k = LUdecomp(k)
    for i in range(n):
        kInv[:,i] = LUsolve(k,kInv[:,i])
    return kInv

k = array([[ 27.58,  7.004, -7.004, 0.0,    0.0  ], \
           [ 7.004, 29.57,  -5.253, 0.0,  -24.32 ], \
           [-7.004, -5.253, 29.57,  0.0,    0.0  ], \
           [ 0.0,    0.0,    0.0,  27.58,  -7.004], \
           [ 0.0,  -24.32,   0.0,  -7.004, 29.57 ]])
p = array([0, 0, 0, 0, -45])/1000.0
kInv = matInv(k)
print(''Flexibility matrix (in m/MN):\n'',kInv)
print(''\nDisplacements (in meters):\n'',dot(kInv,p))
input(''\nPress return to exit'')

Flexibility matrix (in m/MN):
[[ 0.04670666 -0.03657862  0.00456496 -0.00812893 -0.03200971]
 [-0.03657862  0.16461698  0.02057952  0.03658314  0.14405524]
 [ 0.00456496  0.02057952  0.0385552   0.00457342  0.018009  ]
 [-0.00812893  0.03658314  0.00457342  0.04670867  0.04115148]
 [-0.03200971  0.14405524  0.018009    0.04115148  0.16204425]]

Displacements (in meters):
[ 0.00144044 -0.00648249 -0.00081041 -0.00185182 -0.00729199]
```

Problem 9

We used the program `problem2_3_6a` in Problem 6 for the inversion of **A** with the following results:

```
Inverse:
[[-2.73050828  1.48528059 -1.34659614 -0.0324862 ]
 [ 0.4824057  -0.27874885  0.23321067  0.02932383]
 [-0.65403634  0.32428703 -0.34061638  0.00672723]
 [ 2.         -1.          1.          0.        ]]
Absolute value of determinant = 17392.0
```

The large determinant is indicative of a well conditioned matrix.

Again we employed the program `problem2_3_6a` in Problem 6. to invert **B**. The results were

```
Inverse:
[[ 2. -1.  0.  0.]
 [ 0.  2. -1.  0.]
 [ 1. -1.  2. -1.]
 [-2.  0. -1.  1.]]
Absolute value of determinant = 1.0
```

As the determinant is not small, the results are reliable.

Problem 10

This program computes $\mathbf{L}^{-1}$ and checks the result by calculating $\mathbf{L}\mathbf{L}^{-1}$.

```
## problem2_3_10
from numpy import array,identity,dot

def solve(L,b):
    # Solves [L]{x} = {b} by forward substitution
    n = len(L[0])
    b[0] = b[0]/L[0,0]
    for k in range(1,n):
        b[k] = (b[k] - dot(L[k,0:k],b[0:k]))/L[k,k]
    return b

def matInv(L):
    # Solves [L][X] = [I] one column of [X] at a time
    n = len(L[0])
    Linv = identity(n)
    for i in range(n):
        Linv[:,i] = solve(L,Linv[:,i])
    return Linv
```

```
L = array([[30,  0,  0,  0], \
           [18, 36,  0,  0], \
           [ 9, 12, 36,  0], \
           [ 5,  4,  9, 36]])*1.0
Linv = matInv(L)
print(''Inverse:\n'',Linv)
print(''Check L*Linv:\n'',dot(L,Linv))
input(''\nPress return to exit'')
```

```
Inverse:
[[ 0.03333333  0.          0.          0.        ]
 [-0.01666667  0.02777778  0.          0.        ]
 [-0.00277778 -0.00925926  0.02777778  0.        ]
 [-0.00208333 -0.0007716  -0.00694444  0.02777778]]
Check L*Linv:
[[ 1.  0.  0.  0.]
 [ 0.  1.  0.  0.]
 [ 0.  0.  1.  0.]
 [ 0.  0.  0.  1.]]
```

Problem 11

We first rearrange the equations so that the diagonal terms dominate:

$$\begin{bmatrix} 7 & 1 & 1 \\ -3 & 7 & -1 \\ -2 & 5 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ -26 \\ 1 \end{bmatrix}$$

The iterative equations are

$$\begin{aligned} x_1 &= \frac{6 - x_2 - x_3}{7} \\ x_2 &= \frac{-26 + 3x_1 + x_3}{7} \\ x_3 &= \frac{1 + 2x_1 - 5x_2}{9} \end{aligned}$$

Starting with $x = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$, successive iterations yield

$$\begin{aligned} x_1 &= \frac{6 - 1 - 1}{7} = 0.571 \\ x_2 &= \frac{-26 + 3(0.571) + 1}{7} = -3.327 \\ x_3 &= \frac{1 + 2(0.571) - 5(-3.327)}{9} = 2.086 \end{aligned}$$

$$
\begin{aligned}
x_1 &= \frac{6-(-3.327)-2.086}{7} = 1.034 \\
x_2 &= \frac{-26+3(1.034)+2.086}{7} = -2.973 \\
x_3 &= \frac{1+2(1.034)-5(-2.973)}{9} = 1.993
\end{aligned}
$$

$$
\begin{aligned}
x_1 &= \frac{6-(-2.973)-1.993}{7} = 0.997 \\
x_2 &= \frac{-26+3(0.997)+1.993}{7} = -3.002 \\
x_3 &= \frac{1+2(0.997)-5(-3.002)}{9} = 2.000
\end{aligned}
$$

$$
\begin{aligned}
x_1 &= \frac{6-(-3.002)-2.000}{7} = 1.000 \blacktriangleleft \\
x_2 &= \frac{-26+3(1.000)+2.000}{7} = -3.000 \blacktriangleleft \\
x_3 &= \frac{1+2(1.000)-5(-3.000)}{9} = 2.000 \blacktriangleleft
\end{aligned}
$$

Problem 12

The equations are already in optimal order. The formulas for the iterations are

$$
\begin{aligned}
x_1 &= \frac{2x_2-3x_3-x_4}{12} \\
x_2 &= \frac{2x_1-6x_3+3x_4}{15} \\
x_3 &= \frac{20-x_1-6x_2+4x_4}{20} \\
x_4 &= \frac{3x_2-2x_3}{9}
\end{aligned}
$$

Starting with $x_1 = x_2 = x_3 = x_4 = 1$, we get

$$
\begin{aligned}
x_1 &= \frac{2(1)-3(1)-1}{12} = -0.167 \\
x_2 &= \frac{2(-0.167)-6(1)+3(1)}{15} = -0.222 \\
x_3 &= \frac{20-(-0.167)-6(-0.222)+4(1)}{20} = 1.275 \\
x_4 &= \frac{3(-0.222)-2(1.275)}{9} = -0.357
\end{aligned}
$$

$$\begin{aligned}
x_1 &= \frac{2(-0.222) - 3(1.275) - (-0.357)}{12} = -0.326 \\
x_2 &= \frac{2(-0.326) - 6(1.275) + 3(-0.357)}{15} = -0.625 \\
x_3 &= \frac{20 - (-0.326) - 6(-0.625) + 4(-0.357)}{20} = 1.132 \\
x_4 &= \frac{3(-0.625) - 2(1.132)}{9} = -0.460
\end{aligned}$$

Subsequent iterations yield

Iteration	x_1	x_2	x_3	x_3
3	-0.349	-0.591	1.103	-0.442
4	-0.337	-0.575	1.101	-0.436
5	-0.335	-0.572	1.101	-0.435
6	-0.334	-0.572	1.101	-0.435

Thus $\mathbf{x} = \begin{bmatrix} -0.334 & -0.572 & 1.101 & -0.435 \end{bmatrix}^T$ ◀

Problem 13

With $\omega = 1.1$, the iterative equations become

$$\begin{aligned}
x_1 &= 1.1\frac{15 + x_2}{4} - 0.1x_1 \\
x_2 &= 1.1\frac{10 + x_1 + x_3}{4} - 0.1x_2 \\
x_3 &= 1.1\frac{10 + x_2 + x_4}{4} - 0.1x_3 \\
x_4 &= 1.1\frac{10 + x_3}{3} - 0.1x_3
\end{aligned}$$

The starting values are

$$\begin{aligned}
+x_1 &= \frac{15}{4} = 3.75 \qquad x_2 = \frac{10}{4} = 2.5 \\
x_3 &= \frac{10}{4} = 2.5 \qquad x_4 = \frac{10}{3} = 3.33
\end{aligned}$$

Iterations yield

$$\begin{aligned}
x_1 &= 1.1\frac{15 + 2.5}{4} - 0.1(3.75) = 4.44 \\
x_2 &= 1.1\frac{10 + 4.44 + 2.5}{4} - 0.1(2.5) = 4.41 \\
x_3 &= 1.1\frac{10 + 4.41 + 3.33}{4} - 0.1(2.5) = 4.63 \\
x_4 &= 1.1\frac{10 + 4.63}{3} - 0.1(3.33) = 5.03
\end{aligned}$$

$$
\begin{aligned}
x_1 &= 1.1\frac{15+4.41}{4} - 0.1(4.44) = 4.89 \\
x_2 &= 1.1\frac{10+4.89+4.63}{4} - 0.1(4.41) = 4.93 \\
x_3 &= 1.1\frac{10+4.93+5.03}{4} - 0.1(4.63) = 5.03 \\
x_4 &= 1.1\frac{10+5.03}{3} - 0.1(5.03) = 5.01
\end{aligned}
$$

The next two iterations yield

Iteration	x_1	x_2	x_3	x_3
3	4.99	5.01	5.00	5.00
4	5.00	5.00	5.00	5.00

Thus the solution is $x_1 = x_2 = x_3 = x_4 = 5$ ◄

Problem 14

Starting with $\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$, the first iteration is

$$
\begin{aligned}
\mathbf{r}_0 &= \mathbf{b} - \mathbf{A}\mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \\
\mathbf{s}_0 &= \mathbf{r}_0 \\
\mathbf{A}\mathbf{s}_0 &= \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \\
\alpha_0 &= \frac{\mathbf{s}_0^T \mathbf{r}_0}{\mathbf{s}_0^T \mathbf{A}\mathbf{s}_0} = \frac{3}{1} = 3 \\
\mathbf{x}_1 &= \mathbf{x}_0 + \alpha_0 \mathbf{s}_0 = \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix}
\end{aligned}
$$

Second iteration:

$$\begin{aligned}
\mathbf{r}_1 &= \mathbf{b} - \mathbf{A}\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix} = \begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix} \\
\beta_0 &= -\frac{\mathbf{r}_1^T \mathbf{A}\mathbf{s}_0}{\mathbf{s}_0^T \mathbf{A}\mathbf{s}_0} = -\frac{-2}{1} = 2 \\
\mathbf{s}_1 &= \mathbf{r}_1 + \beta_0 \mathbf{s}_0 = \begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 3 \end{bmatrix} \\
\mathbf{A}\mathbf{s}_1 &= \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 3 \\ 3 \end{bmatrix} = \begin{bmatrix} -3 \\ 3 \\ 0 \end{bmatrix} \\
\alpha_1 &= \frac{\mathbf{s}_1^T \mathbf{r}_1}{\mathbf{s}_1^T \mathbf{A}\mathbf{s}_1} = \frac{0+3+3}{0+9+0} = \frac{2}{3} \\
\mathbf{x}_2 &= \mathbf{x}_1 + \alpha_1 \mathbf{s}_1 = \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix} + \frac{2}{3} \begin{bmatrix} 0 \\ 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 5 \end{bmatrix}
\end{aligned}$$

Third and final iteration:

$$\begin{aligned}
\mathbf{r}_2 &= \mathbf{b} - \mathbf{A}\mathbf{x}_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 5 \\ 5 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \\
\beta_1 &= -\frac{\mathbf{r}_2^T \mathbf{A}\mathbf{s}_1}{\mathbf{s}_1^T \mathbf{A}\mathbf{s}_1} = -\frac{-3}{9} = \frac{1}{3} \\
\mathbf{s}_2 &= \mathbf{r}_2 + \beta_1 \mathbf{s}_1 = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} + \frac{1}{3} \begin{bmatrix} 0 \\ 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} \\
\mathbf{A}\mathbf{s}_2 &= \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ -2 \\ 2 \end{bmatrix} \\
\alpha_2 &= \frac{\mathbf{r}_2^T \mathbf{s}_2}{\mathbf{s}_2^T \mathbf{A}\mathbf{s}_2} = \frac{2}{4} = \frac{1}{2} \\
\mathbf{x} &= \mathbf{x}_2 + \alpha_2 \mathbf{s}_2 = \begin{bmatrix} 3 \\ 5 \\ 5 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 6 \end{bmatrix} \quad \blacktriangleleft
\end{aligned}$$

Problem 15

Starting with $\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$, the first iteration is

$$
\begin{aligned}
\mathbf{r}_0 &= \mathbf{b} - \mathbf{A}\mathbf{x}_0 = \begin{bmatrix} 4 \\ 10 \\ -10 \end{bmatrix} - \begin{bmatrix} 3 & 0 & -1 \\ 0 & 4 & -2 \\ -1 & -2 & 5 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 4 \\ 10 \\ -10 \end{bmatrix} \\
\mathbf{s}_0 &= \mathbf{r}_0 \\
\mathbf{A}\mathbf{s}_0 &= \begin{bmatrix} 3 & 0 & -1 \\ 0 & 4 & -2 \\ -1 & -2 & 5 \end{bmatrix} \begin{bmatrix} 4 \\ 10 \\ -10 \end{bmatrix} = \begin{bmatrix} 22 \\ 60 \\ -74 \end{bmatrix} \\
\alpha_0 &= \frac{\mathbf{s}_0^T \mathbf{r}_0}{\mathbf{s}_0^T \mathbf{A}\mathbf{s}_0} = \frac{4^2 + 10^2 + (-10)^2}{4(22) + 10(60) + (-10)(-74)} = 0.15126 \\
\mathbf{x}_1 &= \mathbf{x}_0 + \alpha_0 \mathbf{s}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + 0.15126 \begin{bmatrix} 4 \\ 10 \\ -10 \end{bmatrix} = \begin{bmatrix} 0.60504 \\ 1.51261 \\ -1.51261 \end{bmatrix}
\end{aligned}
$$

Second iteration:

$$
\begin{aligned}
\mathbf{r}_1 &= \mathbf{b} - \mathbf{A}\mathbf{x}_1 = \begin{bmatrix} 4 \\ 10 \\ -10 \end{bmatrix} - \begin{bmatrix} 3 & 0 & -1 \\ 0 & 4 & -2 \\ -1 & -2 & 5 \end{bmatrix} \begin{bmatrix} 0.60504 \\ 1.51261 \\ -1.51261 \end{bmatrix} \\
&= \begin{bmatrix} 0.67227 \\ 0.92434 \\ 1.19331 \end{bmatrix} \\
\beta_0 &= -\frac{\mathbf{r}_1^T \mathbf{A}\mathbf{s}_0}{\mathbf{s}_0^T \mathbf{A}\mathbf{s}_0} = -\frac{0.67227(22) + 0.92434(60) + 1.19331)(-74)}{4(22) + 10(60) + (-10)(-74)} \\
&= 0.012643 \\
\mathbf{s}_1 &= \mathbf{r}_1 + \beta_0 \mathbf{s}_0 = \begin{bmatrix} 0.67227 \\ 0.92434 \\ 1.19331 \end{bmatrix} + 0.012643 \begin{bmatrix} 4 \\ 10 \\ -10 \end{bmatrix} = \begin{bmatrix} 0.72284 \\ 1.05077 \\ 1.06688 \end{bmatrix} \\
\mathbf{A}\mathbf{s}_1 &= \begin{bmatrix} 3 & 0 & -1 \\ 0 & 4 & -2 \\ -1 & -2 & 5 \end{bmatrix} \begin{bmatrix} 0.72284 \\ 1.05077 \\ 1.06688 \end{bmatrix} = \begin{bmatrix} 1.10164 \\ 2.06932 \\ 2.51002 \end{bmatrix} \\
\alpha_1 &= \frac{\mathbf{s}_1^T \mathbf{r}_1}{\mathbf{s}_1^T \mathbf{A}\mathbf{s}_1} = \frac{0.72284(0.67227) + 1.05077(0.92434) + 1.06688(1.19331)}{0.72284(1.10164) + 1.05077(2.06932) + 1.06688(2.51002)} \\
&= 0.48337 \\
\mathbf{x}_2 &= \mathbf{x}_1 + \alpha_1 \mathbf{s}_1 = \begin{bmatrix} 0.60504 \\ 1.51261 \\ -1.51261 \end{bmatrix} + 0.48337 \begin{bmatrix} 0.72284 \\ 1.05077 \\ 1.06688 \end{bmatrix} = \begin{bmatrix} 0.95444 \\ 2.02052 \\ -0.99691 \end{bmatrix}
\end{aligned}
$$

Third and final iteration:

$$\begin{aligned}
\mathbf{r}_2 &= \mathbf{b} - \mathbf{A}\mathbf{x}_2 = \begin{bmatrix} 4 \\ 10 \\ -10 \end{bmatrix} - \begin{bmatrix} 3 & 0 & -1 \\ 0 & 4 & -2 \\ -1 & -2 & 5 \end{bmatrix} \begin{bmatrix} 0.95444 \\ 2.02052 \\ -0.99691 \end{bmatrix} \\
&= \begin{bmatrix} 0.13977 \\ -0.07590 \\ -0.01997 \end{bmatrix} \\
\beta_1 &= -\frac{\mathbf{r}_2^T \mathbf{A}\mathbf{s}_1}{\mathbf{s}_1^T \mathbf{A}\mathbf{s}_1} \\
&= -\frac{0.13977(1.10164) + (-0.07590)(2.06932) + (-0.01997)(2.51002)}{0.72284(1.10164) + 1.05077(2.06932) + 1.06688(2.51002)} \\
&= 9.4201 \times 10^{-3} \\
\mathbf{s}_2 &= \mathbf{r}_2 + \beta_1 \mathbf{s}_1 = \begin{bmatrix} 0.13977 \\ -0.07590 \\ -0.01997 \end{bmatrix} + (9.4201 \times 10^{-3}) \begin{bmatrix} 0.72284 \\ 1.05077 \\ 1.06688 \end{bmatrix} \\
&= \begin{bmatrix} 0.14658 \\ -0.06600 \\ -0.00992 \end{bmatrix} \\
\mathbf{A}\mathbf{s}_2 &= \begin{bmatrix} 3 & 0 & -1 \\ 0 & 4 & -2 \\ -1 & -2 & 5 \end{bmatrix} \begin{bmatrix} 0.14658 \\ -0.06600 \\ -0.00992 \end{bmatrix} = \begin{bmatrix} 0.44966 \\ -0.24416 \\ -0.06418 \end{bmatrix} \\
\alpha_2 &= \frac{\mathbf{r}_2^T \mathbf{s}_2}{\mathbf{s}_2^T \mathbf{A}\mathbf{s}_2} \\
&= \frac{0.13977(0.14658) + (-0.07590)(-0.06600) + (-0.01997)(-0.00992)}{0.14658(0.44966) + (-0.06600)(-0.24416) + (-0.00992)(-0.06418)} \\
&= 0.31084 \\
\mathbf{x} &= \mathbf{x}_2 + \boldsymbol{\alpha}_2 \mathbf{s}_2 = \begin{bmatrix} 0.95444 \\ 2.02052 \\ -0.99691 \end{bmatrix} + 0.31084 \begin{bmatrix} 0.14658 \\ -0.06600 \\ -0.00992 \end{bmatrix} \\
&= \begin{bmatrix} 1.0000 \\ 2.0000 \\ -1.0000 \end{bmatrix} \blacktriangleleft
\end{aligned}$$

Problem 16

```
## problem2_3_16
from numpy import array,dot,zeros
from gaussSeidel import *
```

```
def iterEqs(x,omega):
    for i in range(len(x)):
        sum = dot(a[i],x) - a[i,i]*x[i]
        x[i] = omega*(b[i] - sum)/a[i,i] + (1.0 - omega)*x[i]
    return x

a = array([[ 3,-2, 1, 0, 0, 1],
           [-2, 4,-2, 1, 0, 0],
           [ 1,-2, 4,-2, 1, 0],
           [ 0, 1,-2, 4,-2, 1],
           [ 0, 0, 1,-2, 4,-2],
           [ 1, 0, 0, 1,-2, 3]],float)
b = array([10,-8,10,10,-8,10],float)

x = zeros(len(b))
x,numIter,omega = gaussSeidel(iterEqs,x)
print('x =',x)
print('Number of iterations =',numIter)
print('Relaxation factor =',omega)
input("\nPress return to exit")
```

```
x = [ 1.3 -0.3  4.2  4.2 -0.3  1.3]
Number of iterations = 34
Relaxation factor = 1.3034931233067257
```

Problem 17

```
## problem2_3_17
from numpy import zeros
from gaussSeidel import *

def iterEqs(x,omega):
    n = len(x)
    x[0] =omega*(x[1] - x[n-1])/4.0 + (1.0 - omega)*x[0]
    for i in range(1,n-1):
        x[i] = omega*(x[i-1] + x[i+1])/4.0 + (1.0 - omega)*x[i]
    x[n-1] = omega*(100.0 - x[0] + x[n-2])/4.0 \
           + (1.0 - omega)*x[n-1]
    return x

n = eval(input("Number of equations ==> "))
x = zeros(n)
```

```
x,numIter,omega = gaussSeidel(iterEqs,x)
print("\nNumber of iterations =",numIter)
print("\nRelaxation factor =",omega)
print("\nThe solution is:\n",x)
input("\nPress return to exit")

Number of equations ==> 20

Number of iterations = 21

Relaxation factor = 1.09767975583

The solution is:
[-7.73502692e+00 -2.07259421e+00 -5.55349941e-01 -1.48805549e-01
 -3.98722562e-02 -1.06834753e-02 -2.86164518e-03 -7.63105381e-04
 -1.90776345e-04  8.65000954e-14  1.90776346e-04  7.63105381e-04
  2.86164518e-03  1.06834753e-02  3.98722562e-02  1.48805549e-01
  5.55349941e-01  2.07259421e+00  7.73502692e+00  2.88675135e+01]
```

It took 259 iterations in Example 2.17. This illustrates the profound effect that diagonal dominance has on the rate of convergence in the Gauss-Seidel method.

Problem 18

```
#!/usr/bin/python
## problem2_3_18
from numpy import zeros,sqrt
from conjGrad import *

def Ax(v):
    n = len(v)
    Ax = zeros(n)
    Ax[0] = 4.0*v[0] - v[1] + v[n-1]
    Ax[1:n-1] = -v[0:n-2] + 4.0*v[1:n-1] - v [2:n]
    Ax[n-1] = -v[n-2] + 4.0*v[n-1] + v[0]
    return Ax

n = eval(input("Number of equations ==> "))
b = zeros(n)
b[n-1] = 100.0
x = zeros(n)
```

```
x,numIter = conjGrad(Ax,x,b)
print("\nThe solution is:\n",x)
print("\nNumber of iterations =",numIter)
input("\nPress return to exit")

Number of equations ==> 20

The solution is:
[-7.73502692e+00 -2.07259421e+00 -5.55349941e-01 -1.48805549e-01
 -3.98722562e-02 -1.06834753e-02 -2.86164518e-03 -7.63105381e-04
 -1.90776345e-04  0.00000000e+00  1.90776345e-04  7.63105381e-04
  2.86164518e-03  1.06834753e-02  3.98722562e-02  1.48805549e-01
  5.55349941e-01  2.07259421e+00  7.73502692e+00  2.88675135e+01]
Number of iterations = 9
```

Problem 19

```
## problem2_3_19
from numpy import zeros,array
from conjGrad import *

def Ax(v):
    Ax = zeros(9)
    Ax[0] =               - 4.0*v[0] + v[1] + v[3]
    Ax[1] =          v[0] - 4.0*v[1] + v[2] + v[4]
    Ax[2] =          v[1] - 4.0*v[2]        + v[5]
    Ax[3] = v[0]          - 4.0*v[3] + v[4] + v[6]
    Ax[4] = v[1] + v[3] - 4.0*v[4] + v[5] + v[7]
    Ax[5] = v[2] + v[4] - 4.0*v[5]        + v[8]
    Ax[6] = v[3]          - 4.0*v[6] + v[7]
    Ax[7] = v[4] + v[6] - 4.0*v[7] + v[8]
    Ax[8] = v[5] + v[7] - 4.0*v[8]
    return Ax

b = array([0,0,100,0,0,100,200,200,300])*(-1.0)
x = zeros(9)
x,numIter = conjGrad(Ax,x,b)
print(''\nThe solution is:\n'',x)
print(''\nNumber of iterations ='',numIter)
input(''\nPress return to exit'')

The solution is:
```

```
[  21.42857143   38.39285714   57.14285714   47.32142857   75.
   90.17857143   92.85714286  124.10714286  128.57142857]

Number of iterations = 4
```

Problem 20

(a) The equations can be written as

$$\begin{aligned}
x_1 &= 0.6x_2 + 16 \\
x_2 &= 0.5x_1 + 0.5x_3 \\
x_3 &= 0.5x_2 + 0.5x_4 \\
x_4 &= 0.5x_3 + 0.5x_5 - 10 \\
x_5 &= 0.6x_4
\end{aligned}$$

```
## problem2_3_20a
from numpy import dot, zeros
from math import sqrt

tol = 0.0001
x = zeros(5)
for i in range(100):
    xOld = x.copy()
    x[0] = 0.6*x[1] + 16.0
    x[1] = 0.5*x[0] + 0.5*x[2]
    x[2] = 0.5*x[1] + 0.5*x[3]
    x[3] = 0.5*x[2] + 0.5*x[4] - 10.0
    x[4] = 0.6*x[3]
    dx = sqrt(dot((x-xOld),(x-xOld)))
    if dx < tol:
        print("Number of iterations =",i)
        print("Displacements =\n", x)
        break
input("\nPress return to exit")

Number of iterations = 48
Displacements =
[ 20.71443624   7.85734356  -4.99979929 -17.85698229 -10.71418938]
```

(b)

```
## problem2_3_20b
```

```
from numpy import zeros
from gaussSeidel import *

def iterEqs(x,w):
    v = 1.0 - w
    x[0] = w*(0.6*x[1] + 16.0) + v*x[0]
    x[1] = w*(0.5*x[0] + 0.5*x[2]) + v*x[1]
    x[2] = w*(0.5*x[1] + 0.5*x[3]) + v*x[2]
    x[3] = w*(0.5*x[2] + 0.5*x[4] - 10.0) + v*x[3]
    x[4] = w*(0.6*x[3]) + v*x[4]
    return x

x = zeros(5)
x,numIter,omega = gaussSeidel(iterEqs,x,0.0001)
print("Number of iterations =",numIter)
print("Displacements =\n",x)
print("Relaxation factor =",omega)
input("\nPress return to exit")
```

```
Number of iterations = 25
Displacements =
[ 20.71430925   7.85716468  -4.99998455 -17.85713402 -10.71428189]
Relaxation factor = 1.38242134295
```

We see that relaxation almost halves the number of iterations.

Problem 21

The equations can be written as

$$\begin{aligned}
-5x_1 + 3x_2 &= -80 \\
3x_1 - 6x_2 + 3x_3 &= 0 \\
3x_2 - 6x_3 + 3x_4 &= 0 \\
3x_3 - 6x_4 + 3x_5 &= 60 \\
3x_4 - 5x_5 &= 0
\end{aligned}$$

```
## problem2_3_21
from numpy import array,zeros
from conjGrad import *

def Ax(x):
    Ax = zeros(5)
```

```
    Ax[0] = -5.0*x[0] + 3.0*x[1]
    Ax[1] = 3.0*x[0] - 6.0*x[1] + 3.0*x[2]
    Ax[2] = 3.0*x[1] - 6.0*x[2] + 3.0*x[3]
    Ax[3] = 3.0*x[2] - 6.0*x[3] + 3.0*x[4]
    Ax[4] = 3.0*x[3] - 5.0*x[4]
    return Ax
b = array([-80.0, 0.0, 0.0, 90.0, 0.0])
x = zeros(5)
x,numIter = conjGrad(Ax,x,b,0.0001)
print("x =\n",x)
print("Number of iterations =",numIter)
input("\nPress return to exit")
```

```
x =
[ 20.71428571   7.85714286  -5.         -17.85714286 -10.71428571]
Number of iterations = 4
```

PROBLEM SET 3.1

Problem 1

(a)

i	0	1	2
x_i	-1.2	0.3	1.1
$y_i = P_0[x_i]$	-5.76	-5.61	-3.69

At $x = 0$:

$$\begin{aligned} P_1[x_0, x_1] &= \frac{(0 - x_1)P_0[x_0] + (x_0 - 0)P[x_1]}{x_0 - x_1} \\ &= \frac{(-0.3)(-5.76) + (-1.2)(-5.61)}{-1.2 - 0.3} = -5.64 \end{aligned}$$

$$\begin{aligned} P_1[x_1, x_2] &= \frac{(0 - x_2)P_0[x_1] + (x_1 - 0)P_0[x_2]}{x_1 - x_2} \\ &= \frac{(-1.1)(-5.61) + 0.3(-3.69)}{0.3 - 1.1} = -6.33 \end{aligned}$$

$$\begin{aligned} P_2[x_0, x_1, x_2] &= \frac{(0 - x_2)P_1[x_0, x_1] + (x_0 - 0)P_1[x_1, x_2]}{x_0 - x_2} \\ &= \frac{(-1.1)(-5.64) + (-1.2)(-6.33)}{-1.2 - 1.1} = -6.0 \quad \blacktriangleleft \end{aligned}$$

(b)

$$\begin{aligned} \ell_0(0) &= \frac{(0 - x_1)(0 - x_2)}{(x_0 - x_1)(x_0 - x_2)} = \frac{(-0.3)(-1.1)}{(-1.2 - 0.3)(-1.2 - 1.1)} = 0.0957 \\ \ell_1(0) &= \frac{(0 - x_0)(0 - x_2)}{(x_1 - x_0)(x_1 - x_2)} = \frac{1.2(-1.1)}{[0.3 - (-1.2)]\,(0.3 - 1.1)} = 1.1000 \\ \ell_2(0) &= \frac{(0 - x_0)(0 - x_1)}{(x_2 - x_0)(x_2 - x_1)} = \frac{-1.2(-0.3)}{[1.1 - (-1.2)]\,(1.1 - 0.3)} = -0.1957 \end{aligned}$$

$$\begin{aligned} P_2(0) &= \sum_{i=0}^{2} y_i \ell_i(0) \\ &= -5.76(0.0957) + (-5.61)(1.1000) + (-3.69)(-0.1957) = 6.0 \quad \blacktriangleleft \end{aligned}$$

Problem 2

i	0	1	2	3	4	5	6
x_i	0	0.5	1	1.5	2	2.5	3
y_i	1.8421	2.4694	2.4921	1.9047	0.8509	-0.4112	-1.5727

(a)

This is inverse interpolation: find x where $y = 0$. Using points 4-6:

$$\begin{aligned}
\ell_4(0) &= \frac{(0-y_5)(0-y_6)}{(y_4-y_5)(y_4-y_6)} \\
&= \frac{0.4112(1.5727)}{(0.8509-(-0.4112))(0.8509-(-1.5727))} = 0.2114 \\
\ell_5(0) &= \frac{(0-y_4)(0-y_6)}{(y_5-y_4)(y_5-y_6)} \\
&= \frac{-0.8509(1.5727)}{(-0.4112-0.8509)\,(-0.4112-(-1.5727))} = 0.9129 \\
\ell_6(0) &= \frac{(0-y_4)(0-y_5)}{(y_6-y_4)(y_6-y_5)} \\
&= \frac{-0.8509(0.4112)}{(-1.5727-0.8509)\,(-1.5727-(-0.4112))} = -0.1243 \\
x|_{y=0} &= \sum_{i=4}^{6} x_i \ell_i(0) \\
&= 2(0.2114) + 2.5(0.9129) + 3(-0.1243) = 2.332 \quad \blacktriangleleft
\end{aligned}$$

(b)

Use points 3-6:

$$
\begin{aligned}
\ell_3(0) &= \frac{(0-y_4)(0-y_5)(0-y_6)}{(y_3-y_4)(y_3-y_5)(y_3-y_6)} \\
&= \frac{-0.8509(0.4112((1.5727)}{(1.9047-0.8509)\,(1.9047-(-0.4112))\,(1.9047-(-1.5727))} \\
&= -0.0648 \\
\ell_4(0) &= \frac{0-y_3}{y_4-y_3}\,[\ell_4(0)]_{\text{3-point}} = \frac{-1.9047}{0.8509-1.9047}(0.2114) = 0.3821 \\
\ell_5(0) &= \frac{0-y_3}{y_5-y_3}\,[\ell_5(0)]_{\text{3-point}} = \frac{-1.9047}{-0.4112-1.9047}(0.9129) = 0.7508 \\
\ell_6(0) &= \frac{0-y_3}{y_6-y_3}\,[\ell_6(0)]_{\text{3-point}} = \frac{-1.9129}{-1.5727-1.9129}(-0.1243) = -0.0682 \\
x\big|_{y=0} &= \sum_{i=3}^{6} x_i \ell_i(0) \\
&= 1.5(-0.0648) + 2(0.3821) + 2.5(0.7508) + 3(-0.0682) = 2.339 \quad \blacktriangleleft
\end{aligned}
$$

Problem 3

Interpolationg at $x = 0.7679$:

$$
\begin{aligned}
P_1[x_0, x_1] &= \frac{(x-x_1)P_0[x_0] + (x_0-x)P_0[x_1]}{x_0-x_1} \\
&= \frac{(0.7679-0.5)(1.8421) + (0-0.7679)(2.4694)}{0-0.5} = 2.8055 \\
P_1[x_1, x_2] &= \frac{(x-x_2)P_0[x_1] + (x_1-x)P_0[x_2]}{x_1-x_2} \\
&= \frac{(0.7679-1.0)(2.4694) + (0.5-0.7679)(2.4921)}{0.5-1.0} = 2.4816 \\
P_1[x_2, x_3] &= \frac{(x-x_3)P_0[x_2] + (x_2-x)P_0[x_3]}{x_2-x_3} \\
&= \frac{(0.7679-1.5)(2.4921) + (1.0-0.7679)(1.9047)}{1.0-1.5} = 2.7648
\end{aligned}
$$

$$
\begin{aligned}
P_2[x_0,x_1,x_2] &= \frac{(x-x_2)P_1[x_0,x_1]+(x_0-x)P_1[x_1,x_2]}{x_0-x_2} \\
&= \frac{(0.7679-1.0)(2.8055)+(0-0.7679)(2.4816)}{0-1.0} = 2.5568 \\
P_2[x_1,x_2,x_3] &= \frac{(x-x_3)P_1[x_1,x_2]+(x_1-x)P_1[x_2,x_3]}{x_1-x_3} \\
&= \frac{(0.7679-1.5)(2.4816)+(0.5-0.7679)(2.7648)}{0.5-1.5} = 2.5575
\end{aligned}
$$

$$
\begin{aligned}
y_{\max} &= P_3[x_0,x_1,x_2,x_3] = \frac{(x-x_3)P_2[x_0,x_1,x_2]+(x_0-x)P_2[x_1,x_2,x_3]}{x_0-x_3} \\
&= \frac{(0.7679-1.5)(2.5568)+(0-0.7679)(2.5575)}{0-1.5} = 2.5572 \quad \blacktriangleleft
\end{aligned}
$$

Problem 4

Interpolating at $x = 0.25\pi$:

$$\begin{aligned}
P_1[x_0, x_1] &= \frac{(x - x_1)P_0[x_0] + (x_0 - x)P_0[x_1]}{x_0 - x_1} \\
&= \frac{(0.25\pi - 0.5))(-1.0) + (0 - 0.25\pi)(1.75)}{0 - 0.5} = 3.3197 \\
P_1[x_1, x_2] &= \frac{(x - x_2)P_0[x_1] + (x_1 - x)P_0[x_2]}{x_1 - x_2} \\
&= \frac{(0.25\pi - 1.0)(1.75) + (0.5 - 0.25\pi)(4.0)}{0.5 - 1.0} = 3.0343 \\
P_1[x_2, x_3] &= \frac{(x - x_3)P_0[x_2] + (x_2 - x)P_0[x_3]}{x_2 - x_3} \\
&= \frac{(0.25\pi - 1.5)4.0) + (1.0 - 0.25\pi)(5.75)}{1.0 - 1.5} = 3.2489 \\
P_1[x_3, x_4] &= \frac{(x - x_4)P_0[x_3] + (x_3 - x)P_0[x_4]}{x_3 - x_4} \\
&= \frac{(0.25\pi - 2.0)(5.75) + (1.5 - 0.25\pi)(7.0)}{1.5 - 2.0} 3.9635
\end{aligned}$$

$$\begin{aligned}
P_2[x_0, x_1, x_2] &= \frac{(x - x_2)P_1[x_0, x_1] + (x_0 - x)P_1[x_1, x_2]}{x_0 - x_2} \\
&= \frac{(0.25\pi - 1.0)(3.3197) + (0 - 0.25\pi)(3.0343)}{0 - 1.0} = 3.0955 \\
P_2[x_1, x_2, x_3] &= \frac{(x - x_3)P_1[x_1, x_2] + (x_1 - x)P_1[x_2, x_3]}{x_1 - x_3} \\
&= \frac{(0.25\pi - 1.5)(3.0343) + (0.5 - 0.25\pi)(3.2489)}{0.5 - 1.5} = 3.0955 \\
P_2[x_2, x_3, x_4] &= \frac{(x - x_4)P_1[x_2, x_3] + (x_2 - x)P_1[x_3, x_4]}{x_2 - x_4} \\
&= \frac{(0.25\pi - 2.0)(3.2489) + (1.0 - 0.25\pi)(3.9635)}{1.0 - 2.0} = 3.0955
\end{aligned}$$

There is no need to go futher. The tabulated function is clearly a quadratic (interpolating over any three points gives the same result). Hence $y(0.25\pi) = 3.0955$ ◀

Problem 5

Use Newton's method. The formulas

$$\begin{aligned}\nabla y_i &= \frac{y_i - y_0}{x_i - x_0} \qquad \nabla^2 y_i = \frac{\nabla y_i - \nabla y_1}{x_i - x_1} \\ \nabla^3 y_i &= \frac{\nabla^2 y_i - \nabla^2 y_2}{x_i - x_2} \qquad \nabla^4 y_i = \frac{\nabla^3 y_i - \nabla^3 y_3}{x_i - x_3}\end{aligned}$$

yield the following tableau:

i	x_i	y_i	∇y_i	$\nabla^2 y_i$	$\nabla^3 y_i$	$\nabla^4 y_i$
0	0	−0.7854				
1	0.5	0.6529	2.8766			
2	1.0	1.7390	2.5244	−0.7043		
3	1.5	2.2071	1.9950	−0.8816	−0.3546	
4	2.0	1.9425	1.3640	−1.0084	−0.3041	0.1009

The diagonal terms in the tableau are the coefficients of Newton's polynomial. We evaluate this polynomial at $x = 0.25\pi$ with the recurrence relations

$$\begin{aligned}P_0(0.25\pi) &= 0.1009 \\ P_1(0.25\pi) &= -0.3546 + (0.25\pi - 1.5)(0.1009) = -0.4267 \\ P_2(0.25\pi) &= -0.7043 + (0.25\pi - 1.0)(-0.4267) = -0.6127 \\ P_3(0.25\pi) &= 2.8766 + (0.25\pi - 0.5)(-0.6127) = 2.7017 \\ P_4(0.25\pi) &= y|_{0.25\pi} = -0.7854 + (0.25\pi - 0)(2.7017) = 1.3365 \blacktriangleleft\end{aligned}$$

At $x = 0.5\pi$ the recurrence relations are

$$\begin{aligned}P_0(0.5\pi) &= 0.1009 \\ P_1(0.5\pi) &= -0.3546 + (0.5\pi - 1.5)(0.1009) = -0.3475 \\ P_2(0.5\pi) &= -0.7043 + (0.5\pi - 1.0)(-0.3475) = -0.9027 \\ P_3(0.5\pi) &= 2.8766 + (0.5\pi - 0.5)(-0.9027) = 1.9100 \\ P_4(0.5\pi) &= y|_{0.5\pi} = -0.7854 + (0.5\pi - 0)(1.9100) = 2.2148 \blacktriangleleft\end{aligned}$$

Problem 6

The divided difference table is

i	x_i	y_i	∇y_i	$\nabla^2 y_i$	$\nabla^3 y_i$	$\nabla^4 y_i$	$\nabla^5 y_i$
0	-2	-1					
1	1	2	1				
2	4	59	10	3			
3	-1	4	5	-2	1		
4	3	24	5	2	1	0	
5	4	-53	26	-5	1	0	0

The last nonzero diagonal term $\nabla^3 y_3$ is the coefficient of the cubic term in Newton's polynomial. Therefore, the data points lie on a *cubic* ◀.

Problem 7

Constructing the divided difference table:

i	x_i	y_i	∇y_i	$\nabla^2 y_i$	$\nabla^3 y_i$	$\nabla^4 y_i$
0	-3	0				
1	2	5	1			
2	-1	-4	-2	1		
3	3	12	2	1	0	
4	1	0	0	1	0	0

Hence the polynomial is

$$\begin{aligned} P_2(x) &= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) \\ &= 0 + [x - (-3)] + [x - (-3)]\,(x - 2) \\ &= (x + 3)(1 + x - 2) = (x + 3)(x - 1) \quad \blacktriangleleft \end{aligned}$$

Problem 8

i	0	1	2
x_i	-1	1	3
$y_i = P_0[x_i]$	17	-7	-15

$$\begin{aligned} P_1[x_0, x_1] &= \frac{(x - x_1)P_0[x_0] + (x_0 - x)P_0[x_1]}{x_0 - x_1} \\ &= \frac{(x-1)(17) + (-1-x)(-7)}{-1-1} = -12x + 5 \end{aligned}$$

$$\begin{aligned} P_1[x_1, x_2] &= \frac{(x - x_2)P_0[x_1] + (x_1 - x)P_0[x_2]}{x_1 - x_2} \\ &= \frac{(x-3)(-7) + (1-x)(-15)}{1-3} = -4x - 3 \end{aligned}$$

$$\begin{aligned} P_2[x_0, x_1, x_2] &= \frac{(x - x_2)P_1[x_0, x_1] + (x_0 - x)P_1[x_1, x_2]}{x_0 - x_2} \\ &= \frac{(x-3)(-12x+5) + (-1-x)(-4x-3)}{-1-3} \\ &= 2x^2 - 12x + 3 \quad \blacktriangleleft \end{aligned}$$

Problem 9

i	0	1	2
h_i (km)	0	3	6
ρ_i (kg/m^3)	1.225	0.905	0.652

$$\begin{aligned} \ell_0 &= \frac{(h - h_1)(h - h_2)}{(h_0 - h_1)(h_0 - h_2)} = \frac{(h-3)(h-6)}{(0-3)(0-6)} = \frac{(h-3)(h-6)}{18} \\ \ell_1 &= \frac{(h - h_0)(h - h_2)}{(h_1 - h_0)(h_1 - h_2)} = \frac{(h-0)(h-6)}{(3-0)(3-6)} = -\frac{h(h-6)}{9} \\ \ell_2 &= \frac{(h - h_0)(h - h_1)}{(h_2 - h_0)(h_2 - h_1)} = \frac{(h-0)(h-3)}{(6-0)(6-3)} = \frac{h(h-3)}{18} \end{aligned}$$

$$\begin{aligned} \rho(h) &= \sum_{i=0}^{2} \rho_i \ell_i \\ &= 1.225\frac{(h-3)(h-6)}{18} - 0.905\frac{h(h-6)}{9} + 0.652\frac{h(h-3)}{18} \\ &= 0.003722h^2 - 0.1178h + 1.225 \quad \blacktriangleleft \end{aligned}$$

Problem 10

i	0	1	2
x_i	0	1	2
y_i	0	2	1

For natural spline we have $k_0 = k_2 = 0$. The equation for k_1 is

$$\begin{aligned} k_0 + 4k_1 + k_2 &= \frac{6}{h^2}(y_0 - 2y_1 + y_2) \\ 0 + 4k_1 + 0 &= \frac{6}{1^2}\left[0 - 2(2) + 1\right] \qquad k_1 = -4.5 \end{aligned}$$

The interpolant in $0 \le x \le 1$ is

$$\begin{aligned} f_{0,1}(x) &= -\frac{k_1}{6}\left[\frac{(x-x_0)^3}{x_0-x_1} - (x-x_0)(x_0-x_1)\right] \\ &\quad + \frac{y_0(x-x_1) - y_1(x-x_0)}{x_0-x_1} \\ &= \frac{4.5}{6}\left(\frac{(x-0)^3}{0-1} - (x-0)(0-1)\right) + \frac{0-2(x-0)}{0-1} \\ &= -0.75x^3 + 2.75x \quad \blacktriangleleft \end{aligned}$$

The interpolant in $1 \le x \le 2$ is

$$\begin{aligned} f_{1,2}(x) &= \frac{k_1}{6}\left[\frac{(x-x_2)^3}{x_1-x_2} - (x-x_2)(x_1-x_2)\right] \\ &\quad + \frac{y_1(x-x_2) - y_2(x-x_1)}{x_1-x_2} \\ &= -\frac{4.5}{6}\left(\frac{(x-2)^3}{1-2} - (x-2)(1-2)\right) + \frac{2(x-2)-(x-1)}{1-2} \\ &= 0.75(x-2)^3 - 1.75x + 4.5 \quad \blacktriangleleft \end{aligned}$$

Check:

$$\begin{aligned} f'_{0,1}(x) &= -3(0.75)x^2 + 2.75 = -2.25x^2 + 2.75 \\ f'_{1,2}(x) &= 3(0.75)(x-2)^2 - 1.75 = 2.25(x-2)^2 - 1.75 \end{aligned}$$

$$\begin{aligned} f'_{0,1}(1) &= -2.25(1)^2 + 2.75 = 0.5 \\ f'_{1,2}(1) &= 2.25(1-2)^2 - 1.75 = 0.5 \qquad \text{O.K.} \end{aligned}$$

$$\begin{aligned} f''_{0,1}(1) &= -2.25(2) = -4.5 \\ f''_{1,2}(1) &= 2.25(2)(1-2) = -4.5 \qquad \text{O.K.} \end{aligned}$$

Problem 11

i	0	1	2	3	4
x_i	1	2	3	4	5
y_i	13	15	12	9	13

For equally spaced knots, the equations for the curvatures are

$$k_{i-1} + 4k_i + k_{i+1} = \frac{6}{h^2}(y_{i-1} - 2y_i + y_{i+1}), \quad i = 1, 2, 3$$

Noting that $k_0 = k_4$ and $h = 1$, we get

$$\begin{aligned} 4k_1 + k_2 &= 6\,[13 - 2(15) + 12] = -30 \\ k_1 + 4k_2 + k_3 &= 6\,[15 - 2(12) + 9] = 0 \\ k_2 + 4k_3 &= 6\,[12 - 2(9) + 13] = 42 \end{aligned}$$

Solution of these equations is

$$k_1 = -7.286 \qquad k_2 = -0.857 \qquad k_3 = 10.714$$

The interpolant between knots 2 and 3 is

$$\begin{aligned} f_{2,3}(x) &= \frac{k_2}{6}\left[\frac{(x-x_3)^3}{x_2 - x_3} - (x - x_3)(x_2 - x_3)\right] \\ &\quad - \frac{k_3}{6}\left[\frac{(x-x_2)^3}{x_2 - x_3} - (x - x_2)(x_2 - x_3)\right] \\ &\quad + \frac{y_2(x - x_3) - y_3(x - x_2)}{x_2 - x_3} \end{aligned}$$

Hence

$$\begin{aligned} f_{2,3}(3.4) &= \frac{-0.857}{6}\left[\frac{(3.4-4)^3}{3-4} - (3.4-4)(3-4)\right] \\ &\quad - \frac{10.714}{6}\left[\frac{(3.4-3)^3}{3-4} - (3.4-3)(3-4)\right] \\ &\quad + \frac{12(3.4-4) - 9(3.4-3)}{3-4} \\ &= 0.0548\,48 - 0.599\,98 + 10.8 = 10.255 \quad \blacktriangleleft \end{aligned}$$

Problem 12

After reordering, the data are

i	0	1	2	3	4
x_i	1.0	0.8	0.6	0.4	0.2
y_i	−1.049	−0.266	0.377	0.855	1.150

The equations for the curvatures at the interior knots are (note that the roles of x and y are interchanged and $k_0 = k_4 = 0$):

$$\begin{aligned} & k_{i-1}(y_{i-1} - y_i) + 2k_i(y_{i-1} - y_{i+1}) + k_{i+1}(y_i - y_{i+1}) \\ = \;& 6\left(\frac{x_{i-1} - x_i}{y_{i-1} - y_i} - \frac{x_i - x_{i+1}}{y_i - y_{i+1}}\right), \quad i = 1, 2, 3 \end{aligned}$$

These are simultaneous equations $\mathbf{A}\mathbf{k} = \mathbf{b}$, where $\mathbf{k} = \begin{bmatrix} k_1 & k_2 & k_3 \end{bmatrix}^T$ and

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} 2\,(-1.049 - 0.377) & -0.266 - 0.377 & 0 \\ -0.266 - 0.377 & 2\,(-0.266 - 0.855) & 0.377 - 0.855 \\ 0 & 0.377 - 0.855 & 2(0.377 - 1.150) \end{bmatrix} \\ &= \begin{bmatrix} -2.852 & -0.643 & 0 \\ -0.643 & -2.242 & -0.478 \\ 0 & -0.478 & -1.546 \end{bmatrix} \end{aligned}$$

+

$$\mathbf{b} = 6\begin{bmatrix} \dfrac{1.0 - 0.8}{-1.049 - (-0.266)} - \dfrac{0.8 - 0.6}{-0.266 - 0.377} \\ \dfrac{0.8 - 0.6}{-0.266 - 0.377} - \dfrac{0.6 - 0.4}{0.377 - 0.855} \\ \dfrac{0.6 - 0.4}{0.377 - 0.855} - \dfrac{0.4 - 0.2}{0.855 - 1.150} \end{bmatrix} = \begin{bmatrix} 0.3337 \\ 0.6442 \\ 1.5573 \end{bmatrix}$$

The solution is

$$\begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} = \begin{bmatrix} -0.1069 \\ -0.0449 \\ -0.9934 \end{bmatrix}$$

The interpolant between knots 1 and 2 is

$$\begin{aligned} f_{1,2}(y) &= \frac{k_1}{6}\left[\frac{(y - y_2)^3}{(y_1 - y_2)} - (y - y_2)(y_1 - y_2)\right] \\ &\quad - \frac{k_2}{6}\left[\frac{(y - y_1)^3}{(y_1 - y_2)} - (y - y_1)(y_1 - y_2)\right] \\ &\quad + \frac{x_1(y - y_2) - x_2(y - y_1)}{y_1 - y_2} \end{aligned}$$

Evaluating at $y = 0$:

$$\begin{aligned} f_{1,2}(0) &= \frac{-0.1069}{6}\left(\frac{(-0.377)^3}{-0.266 - 0.377} + 0.377(-0.266 - 0.377)\right) \\ &\quad + \frac{0.0449}{6}\left(\frac{(0.266)^3}{-0.266 - 0.377} - 0.266(-0.266 - 0.377)\right) \\ &\quad + \frac{0.8(-0.377) - 0.6(0.266)}{-0.266 - 0.377} \\ &= 0.0028 + 0.0011 + 0.7173 = 0.7212 \blacktriangleleft \end{aligned}$$

Problem 13

i	0	1	2	3
x	0	1	2	3
y	1	1	0.5	0

With evenly spaced knots, the equations for the curvatures are

$$k_{i-1} + 4k_i + k_{i+1} = \frac{6}{h^2}(y_{i-1} - 2y_i + y_{i+1}), \quad i = 1, 2$$

With $k_0 = k_1$, $k_3 = k_2$ and $h = 1$ these equations are

$$\begin{aligned} 5k_1 + k_2 &= 6(1 - 2(1) + 0.5) = -3 \\ k_1 + 5k_2 &= 6\,[1 - 2(0.5) + 0] = 0 \end{aligned}$$

The solution is $k_1 = -5/8$, $k_2 = 1/8$. The interpolant can now be evaluated from

$$\begin{aligned} f_{i,i+1}(x) &= \frac{k_i}{6}\left[\frac{(x - x_{i+1})^3}{x_i - x_{i+1}} - (x - x_{i+1})(x_i - x_{i+1})\right] \\ &\quad - \frac{k_{i+1}}{6}\left[\frac{(x - x_i)^3}{x_i - x_{i+1}} - (x - x_i)(x_i - x_{i+1})\right] \\ &\quad + \frac{y_i(x - x_{i+1}) - y_{i+1}(x - x_i)}{x_i - x_{i+1}} \end{aligned}$$

Substituting $x_i - x_{i+1} = -1$ and $i = 2$, this reduces to

$$\begin{aligned} f_{2,3}(x) &= \frac{k_2}{6}\left[-(x - x_3)^3 + (x - x_3)\right] - \frac{k_3}{6}\left[-(x - x_2)^3 + (x - x_2)\right] \\ &\quad -y_2(x - x_3) + y_3(x - x_2) \end{aligned}$$

Therefore,

$$\begin{aligned} f_{2,3}(2.6) &= \frac{1/8}{6}\left[-(2.6 - 3)^3 + (2.6 - 3)\right] - \frac{1/8}{6}\left[-(2.6 - 2)^3 + (2.6 - 2)\right] \\ &\quad -0.5(2.6 - 3) + 0 \\ &= 0.185 \blacktriangleleft \end{aligned}$$

Problem 14

This program prompts for x:

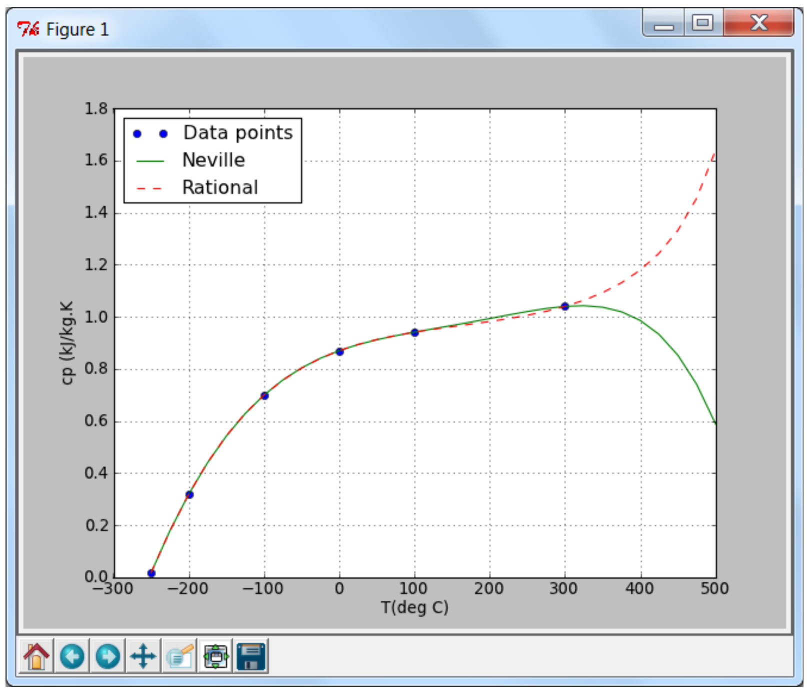
```
## problem3_1_14
from neville import *
from numpy import array

xData = array([-2.0, -0.1, -1.5, 0.5, -0.6, 2.2, 1.0, 1.8])
yData = array([2.2796, 1.0025, 1.6467, 1.0635,  \
               1.0920, 2.6291, 1.2661, 1.9896])
while True:
    try: x = eval(input("\nx ==> "))
    except SyntaxError: break
    print ("y = ",neville(xData,yData,x))
input("Done. Press return to exit")

x ==> 1.1
y =  1.32619402777

x ==> 1.2
y =  1.39375781058

x ==> 1.3
y =  1.46930770699
```

Problem 15

We chose Neville's method for the polynomial interpolation (Newton's method would produce identical results).

```
## problem3_1_15
from neville import *
from rational import *
from numpy import array,arange,zeros
import matplotlib.pyplot as plt

xData = array([-250, -200, -100, 0, 100, 300])*1.0
yData = array([0.0163, 0.318, 0.699, 0.870, 0.941, 1.04])
T = arange(-250.0,525.0,25.0)
n = len(T)
cp_nev = zeros(n)
```

```
cp_rat = zeros(n)
for i in range(n):
    cp_nev[i] = neville(xData,yData,T[i])
    cp_rat[i] = rational(xData,yData,T[i])
plt.plot(xData,yData,'o',T,cp_nev,'-',T,cp_rat,'--')
plt.xlabel('T(deg C'); plt.ylabel('cp (kJ/kg.K')
plt.legend(('Data points','Neville','Rational'),loc=0)
plt.grid(True)
plt.show()
input("Press return to exit")
```

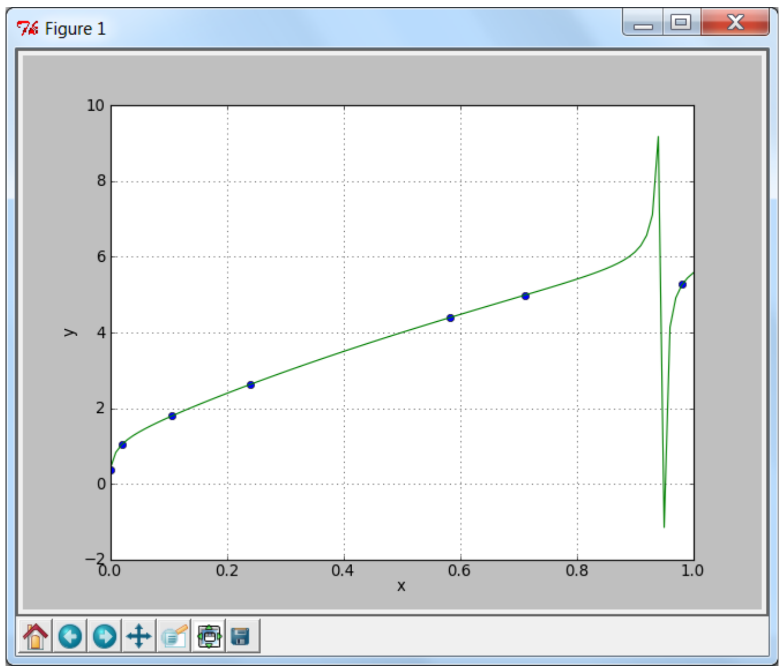


The plot shows that polynomial interpolation is slightly superior to the rational function interpolation in the range of the data points. Both are unacceptable for extrapolation.

Problem 16

```
## problem3_1_16
from rational import *
from numpy import array,zeros,arange
import matplotlib.pyplot as plt

xData = array([0.0, 0.0204, 0.1055, 0.241, 0.582, 0.712, 0.981])
```

```
yData = array([0.385, 1.04, 1.79, 2.63, 4.39, 4.99, 5.27])
x = arange(0.0, 1.01, 0.01)
n = len(x)
y_nev = zeros(n)
y_rat = zeros(n)
for i in range(n):
    y_rat[i] = rational(xData,yData,x[i])
plt.plot(xData,yData,'o',x,y_rat,'-')
plt.xlabel('x'); plt.ylabel('y')
plt.grid(True); plt.show()
input("Press return to exit")
```



The interpolant looks good in the range $0 < x < 0.7$, but is unacceptable in $x > 8$. The cause of the problem is a pole at about $x = 0.96$.

Problem 17

Since cubic spline resists spurious wiggles, it can be used with relative safety over numerous data points. This program does cubic spline interpolation on the logarithms of the data. It prompts for user input of x. In this case, $x = \text{Re}$ and $y = c_D$.

```
## problem3_1_17
from cubicSpline import *
```

```
from numpy import array,log10

xData = array([0.2, 2.0, 20.0, 200.0, 2000.0, 20000.0])
yData = array([103.0, 13.9, 2.72, 0.8, 0.401, 0.433])
logxData = log10(xData)
logyData = log10(yData)
k = curvatures(logxData,logyData)
while True:
    try: x = eval(input("\nx ==> "))
    except SyntaxError: break
    logx = log10(x)
    logy = evalSpline(logxData,logyData,k,logx)
    print("y =",10.0**logy)
input("Done. Press return to exit")
```

```
x ==> 5.0
y = 6.90159241175

x ==> 50.0
y = 1.59083540468

x ==> 500.0
y = 0.557433520352

x ==> 5000.0
y = 0.38682177204
```

Problem 18

The following program uses Neville's method. It prompts for x and the range of the data points (first point number, last point number) over which the interpolation is to be performed. Note that $x = \text{Re}$ and $y = c_D$.

```
## problem3_1_18
from neville import *
from numpy import array,log10

xData = array([0.2, 2.0, 20.0, 200.0, 2000.0, 20000.0])
yData = array([103.0, 13.9, 2.72, 0.8, 0.401, 0.433])
logxData = log10(xData)
logyData = log10(yData)
while True:
```

```
    try: x = eval(input("\nx ==> "))
    except SyntaxError: break
    logx = log10(x)
    n1,n2 = eval(input("Data point range (1st,last) ==> "))
    logy = neville(logxData[n1:n2+1],logyData[n1:n2+1],logx)
    print("y =",10.0**logy)
input("Done. Press return to exit")
```

```
x ==> 5.0
Data point range (1st,last) ==> 0,3
y = 6.93256528318

x ==> 50.0
Data point range (1st,last) ==> 1,4
y = 1.58061844705

x ==> 500.0
Data point range (1st,last) ==> 2,5
y = 0.562747719041

x ==> 5000.0
Data point range (1st,last) ==> 2,5
y = 0.372226842883
```

Problem 19

We could use global cubic spline interpolation or polynomial interpolation over nearest-neighour data points. The following program, which prompts for the viscosity, uses the cubic spline.

```
## problem3_1_19
from cubicSpline import *
from numpy import array

tData = array([0.0, 21.1, 37.8, 54.4, 71.1, 87.8, 100.0])
mData = array([1.79, 1.13, 0.696, 0.519, 0.338, 0.321, 0.296])
k = curvatures(tData,mData)
while True:
    try: mu = eval(input("\nTemperature ==> "))
    except SyntaxError: break
    print("Viscocity =",evalSpline(tData,mData,k,mu))
input("Done. Press return to exit")
```

```
Temperature ==> 10.0
Viscocity = 1.47498133187

Temperature ==> 30.0
Viscocity = 0.870147884458

Temperature ==> 60.0
Viscocity = 0.45408107032

Temperature ==> 90.0
Viscocity = 0.319452861245
```

Problem 20

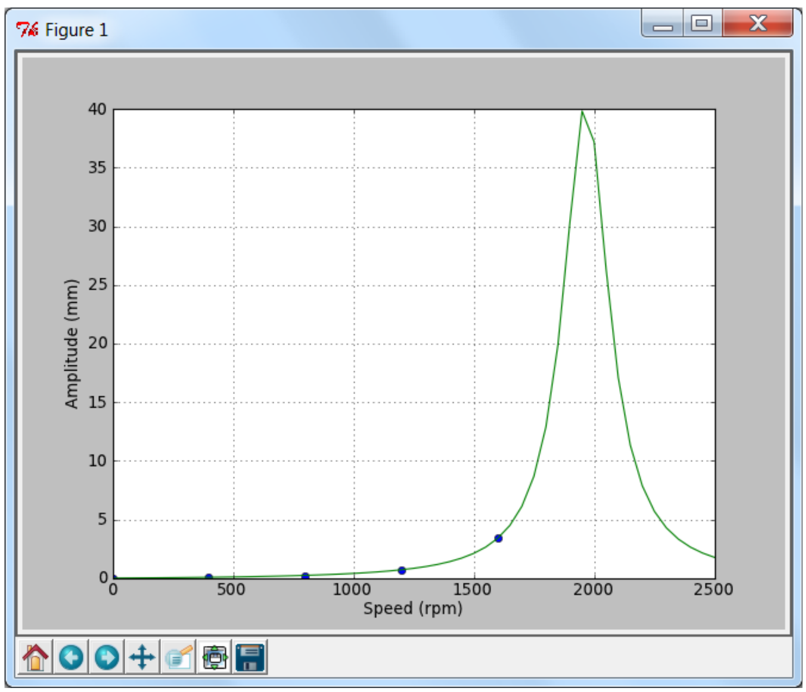
As we have extrapolation, polynomial interpolation over all the data points is dangerous. The following is essentially the program used in Problem 14; it interpolates over three points with Neville's method:

```
## problem3_1_20
from neville import *
from numpy import array

hData = array([6.10, 7.625, 9.150])
rData = array([0.5328, 0.4481, 0.3741])
while True:
    try: h = eval(input("\nh ==> "))
    except SyntaxError: break
    print("rho = ",neville(hData,rData,h))
input("Done. Press return to exit")

h ==> 10.5
rho =  0.317520451492
```

Problem 21

```
## problem3_1_21
from numpy import array,arange,zeros
from rational import *
import matplotlib.pyplot as plt
```

```
xData = array([0.0, 400.0, 800.0, 1200.0, 1600.0])
yData = array([0.0, 0.072, 0.233, 0.712, 3.400])
x = arange(0.0,2550.0,50.0)
n = len(x)
y = zeros(n)
for i in range(n):
    y[i] = rational(xData,yData,x[i])
plt.plot(xData,yData,'o',x,y,'-')
plt.xlabel('Speed (rpm)'); plt.ylabel('Amplitude (mm)')
plt.grid(True); plt.show()
input("Press return to exit")
```



Inspection of the plot reveals resonance at about 1950 rpm.

PROBLEM SET 3.2

Problem 1

The equation of the regression line is

$$f(x) = a + bx = (\bar{y} - b\bar{x}) + bx$$

Thus

$$f(\bar{x}) = \bar{y} \quad \text{Q.E.D.}$$

Problem 2

	x	y	$x - \bar{x}$	$x(x - \bar{x})$	$y(x - \bar{x})$	$f(x)$	$y - f(x)$
	-1.0	-1.00	-1.0	1.00	1.000	-1.02	0.02
	-0.5	-0.55	-0.5	0.25	0.275	-0.52	-0.03
	0.0	0.00	0.0	0.00	0.000	-0.02	0.02
	0.5	0.45	0.5	0.25	0.225	0.48	-0.03
	1.0	1.00	1.0	1.00	1.000	0.98	0.02
$\sum$	0.0	-0.100		2.50	2.500		

$$\bar{x} = \frac{1}{5}\sum x = 0 \qquad \bar{y} = \frac{1}{5}\sum y = \frac{-0.100}{5} = -0.02$$

$$\begin{aligned} b &= \frac{\sum y(x - \bar{x})}{\sum x(x - \bar{x})} = \frac{2.500}{2.50} = 1.0 \\ a &= \bar{y} - \bar{x}b = -0.02 - 0(1.0) = -0.02 \end{aligned}$$

The regression line is

$$f(x) = -0.02 + x \blacktriangleleft$$

$$S = \sum [y - f(x)]^2 = 3(0.02)^2 + 2(-0.03)^2 = 0.003$$

The standard deviation is ($n + 1 =$ the number of data points; $m =$ degree of interpolating polynomial)

$$\sigma = \sqrt{\frac{S}{n - m}} = \sqrt{\frac{0.003}{4 - 1}} = 0.0316 \blacktriangleleft$$

Problem 3

	x (Stress)	y (Strain)	$x - \bar{x}$	$x(x - \bar{x})$	$y(x - \bar{x})$
	34.5	0.46	−51.75	−1785	−23.81
	69.0	0.95	−17.25	−1190	−16.39
	103.5	1.48	17.25	1785	25.53
	138.0	1.93	51.75	7142	99.88
	34.5	0.34	−51.75	−1785	−17.60
	69.0	1.02	−17.25	−1190	−17.60
	103.5	1.51	17.25	1785	26.05
	138.0	2.09	51.75	7142	108.16
	34.5	0.73	−51.75	−1785	−37.78
	69.0	1.10	−17.25	−1190	−18.98
	103.5	1.62	17.25	1785	27.95
	138.0	2.12	51.75	7142	109.71
$\sum$	1 035.0	15.35		17 854	265.12

$$\bar{x} = \frac{\sum x}{12} = \frac{1035.0}{12} = 86.25 \qquad \bar{y} = \frac{\sum y}{12} = \frac{15.35}{12} = 1.2792$$

Converting the strain y from mm/m to m/m, we have

$$b = \frac{\sum y(x - \bar{x})}{\sum x(x - \bar{x})} = \frac{265.12 \times 10^{-3}}{17\,854}(\text{MPa})^{-1} = 1.4849 \times 10^{-5}(\text{MPa})^{-1}$$

The modulus of elasticity is

$$E = \frac{1}{b} = \frac{1}{1.4849 \times 10^{-5}}\text{MPa} = 67350 \text{ MPa} = 67.34 \text{ GPa} \blacktriangleleft$$

Problem 4

Let x = stress and y = strain.

	x	y	W	W^2x	W^2y	$x-\bar{x}$	$W^2x(x-\bar{x})$	$W^2y(x-\bar{x})$
	34.5	0.46	1.0	34.5	0.460	−51.75	−1785	−23.81
	69.0	0.95	1.0	69.0	0.950	−17.25	−1190	−16.39
	103.5	1.48	1.0	103.5	1.480	17.25	1785	25.53
	138.0	1.93	1.0	138.0	1.930	51.75	7142	99.88
	34.5	0.34	1.0	34.5	0.340	−51.75	−1785	−17.60
	69.0	1.02	1.0	69.0	1.020	−17.25	−1190	−17.60
	103.5	1.51	1.0	103.5	1.510	17.25	1785	26.05
	138.0	2.09	1.0	138.0	2.090	51.75	7142	108.16
	34.5	0.73	0.5	8.6	0.183	−51.75	−446	−9.44
	69.0	1.10	0.5	17.3	0.275	−17.25	−298	−4.74
	103.5	1.62	0.5	25.9	0.405	17.25	445	6.99
	138.0	2.12	0.5	34.5	0.530	51.75	1785	27.43
$\sum$				776.3	11.173		13 390	204.46

$$\sum W^2 = 8 + 4(0.25) = 9$$

$$\hat{x} = \frac{\sum W^2 x}{\sum W^2} = \frac{776.3}{9} = 86.25 \qquad \hat{y} = \frac{\sum W^2 y}{\sum W} = \frac{11.173}{9} = 1.2414$$

Converting the strain y from mm/m to m/m, we have

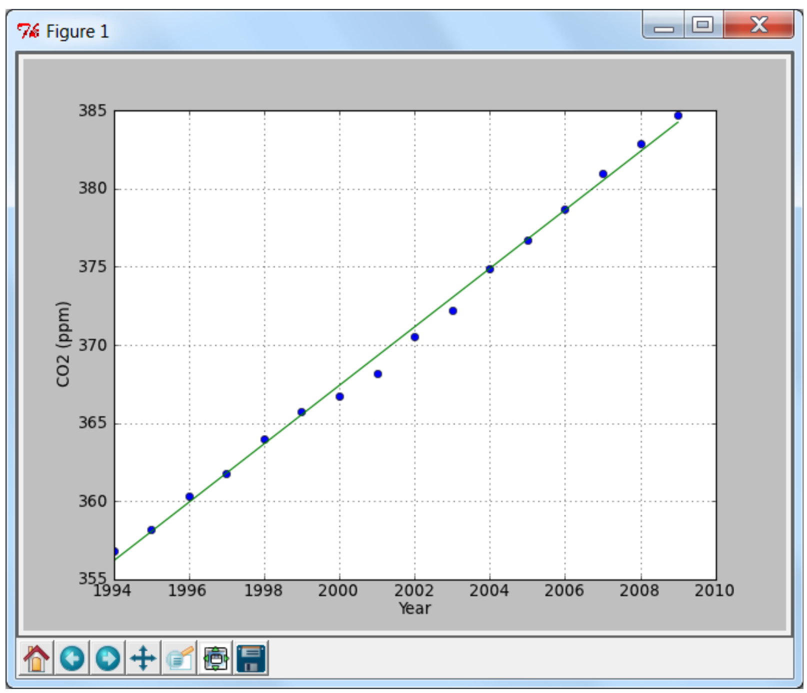
$$b = \frac{\sum W^2 y(x-\hat{x})}{\sum W^2 x(x-\hat{x})} = \frac{204.46 \times 10^{-3}}{13\,390}(\text{MPa})^{-1} = 1.5270 \times 10^{-5}(\text{MPa})^{-1}$$

The modulus of elasticity is

$$E = \frac{1}{b} = \frac{1}{1.5270 \times 10^{-5}}\text{MPa} = 65.49 \text{ MPa} = 65.49 \text{ GPa} \blacktriangleleft$$

Problem 5

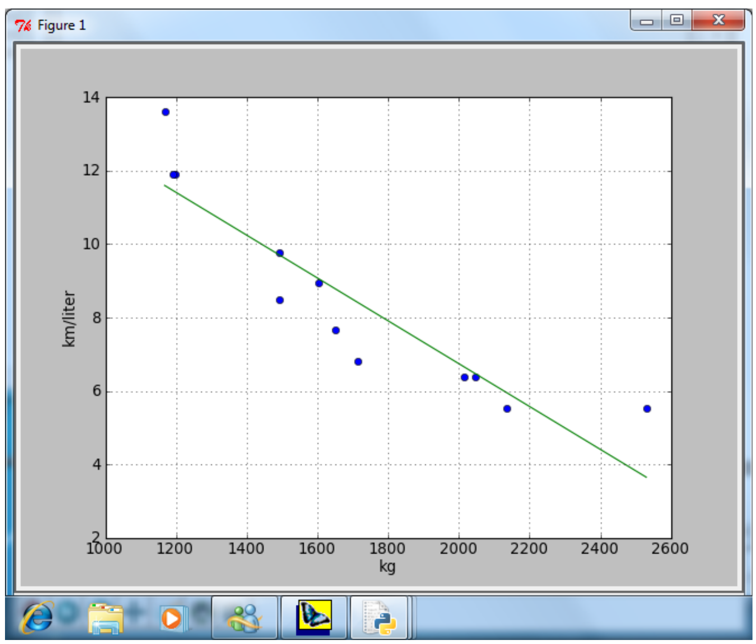
```
# problem3_2_5
import numpy as np
from polyFit import *
from plotPoly import *

ppm = np.array([356.8, 358.2, 360.3, 361.8, 364.0, 365.7, \
                366.7, 368.2, 370.5, 372.2, 374.9, 376.7, \
                378.7, 381.0, 382.9, 384.7])
year = np.arange(1994,2010,1)
c = polyFit(year,ppm,1)
print('Increase in ppm/year =',c[1])
plotPoly(year,ppm,c,'Year','CO2 (ppm)')
input("\nPress return to exit")

Increase in ppm/year = 1.87220588236
```



Problem 6

```
## problem3_2_6
```

```
from numpy import array
from polyFit import *
from plotPoly import *

xData = array([1198, 1715, 2530, 2014, 2136, 1492, 1652, \
               1168, 1492, 1602, 1192, 2045])*1.0
yData = array([11.9, 6.8, 5.53, 6.38, 5.53, 8.50, 7.65, \
               13.60, 9.78, 8.93, 11.90, 6.38])
coeff = polyFit(xData,yData,1)
print("Coefficients are:\n",coeff)
print("Std. deviation =",stdDev(coeff,xData,yData))
plotPoly(xData,yData,coeff,'kg','km/liter')
input("Finished. Press return to exit")

Coefficients are:
[  1.84099405e+01  -5.83313334e-03]
Std. deviation = 1.1641716342135628
```



Problem 7

```
## problem3_2_7
from numpy import array
from polyFit import *
from plotPoly import *
```

```
xData = array([0.0, 1.525, 3.050, 4.575, 6.1, \
               7.625, 9.15])
yData = array([1.0, 0.8617, 0.7385, 0.6292, 0.5328, \
               0.4481, 0.3741])
coeff = polyFit(xData,yData,2)
print("Coefficients are:\n",coeff)
print("Std. deviation =",stdDev(coeff,xData,yData))
h = 10.5
den = coeff[0] + coeff[1]*h + coeff[2]*h**2
print("Rel. density at h = 10.5 km =",den)
plotPoly(xData,yData,coeff,'Elevation (km)','Rel. density')
input("Finished. Press return to exit")
```

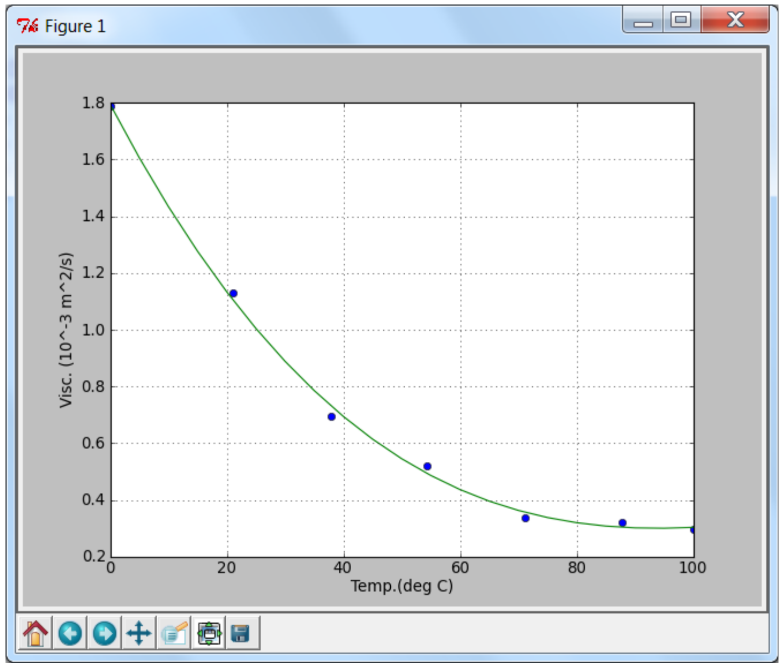
```
Coefficients are:
[ 0.99889524 -0.09344731  0.00276321]
Std. deviation = 0.0013473077210637862
Rel. density at h = 10.5 km = 0.32234242971
```

Problem 8

```
## problem3_2_8
from numpy import array
from polyFit import *
from evalPoly import *
from plotPoly import *

tData = array([0.0, 21.1, 37.8, 54.4, 71.1, 87.8, 100.0])
mData = array([1.79, 1.13, 0.696, 0.519, 0.338, 0.321, 0.296])
t = array([10.0,30.0,60.0,90.0])
coeff = polyFit(tData,mData,3)
print("Coefficients are:\n",coeff)
print("Std. deviation =",stdDev(coeff,tData,mData))
for i in range(4):
    mu,dmu,ddmu = evalPoly(coeff,t[i])
    print(" Temp. =",t[i],"  Visc. =",mu)
plotPoly(tData,mData,coeff,'Temp.(deg C)','Visc. (10^-3 m^2/s)')
input("Finished. Press return to exit")
```

```
Coefficients are:
[1.79570895e+00 -3.93212797e-02  3.28569164e-04 -8.45886166e-07]
Std. deviation = 0.03400520427645581
```

```
Temp. = 10.0   Visc. = 1.43450717837
Temp. = 30.0   Visc. = 0.888943874512
Temp. = 60.0   Visc. = 0.436569740092
Temp. = 90.0   Visc. = 0.30155298339
```

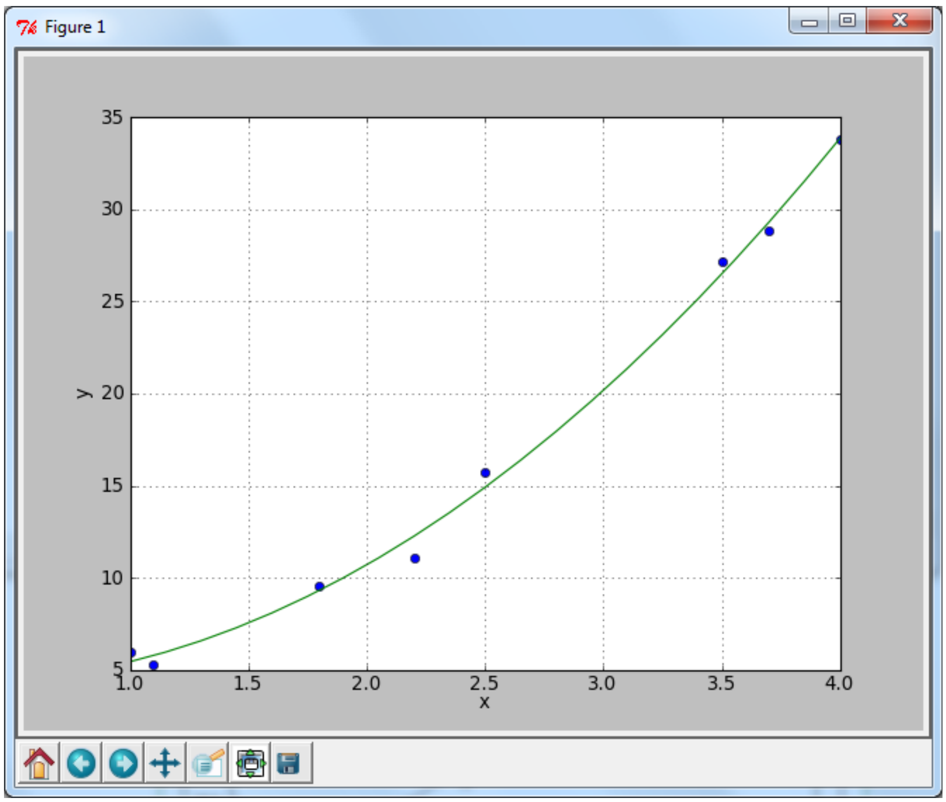


Problem 9

```
## problem3_2_9
from numpy import array
from polyFit import *
from plotPoly import *

xData = array([1.0, 2.5, 3.5, 4.0, 1.1, 1.8, 2.2, 3.7])
yData = array([6.008, 15.722, 27.13, 33.772, 5.257, 9.549, \
               11.098, 28.828])
for m in range(1,3):
    coeff = polyFit(xData,yData,m)
    print("Degree of polynomial =",m)
    print("Coefficients are:\n",coeff)
    print("Std. deviation =",stdDev(coeff,xData,yData),"\n")
plotPoly(xData,yData,coeff)
input("Finished. Press return to exit")
```

```
Degree of polynomial ==> 1
Coefficients are:
[-6.18989525  9.43854354]
Std. deviation = 2.2435638279603114

Degree of polynomial ==> 2
Coefficients are:
[ 4.40567377 -1.06889613  2.10811822]
Std. deviation = 0.8129279610540698
```

The *quadratic* (plotted below) is a much better fit.

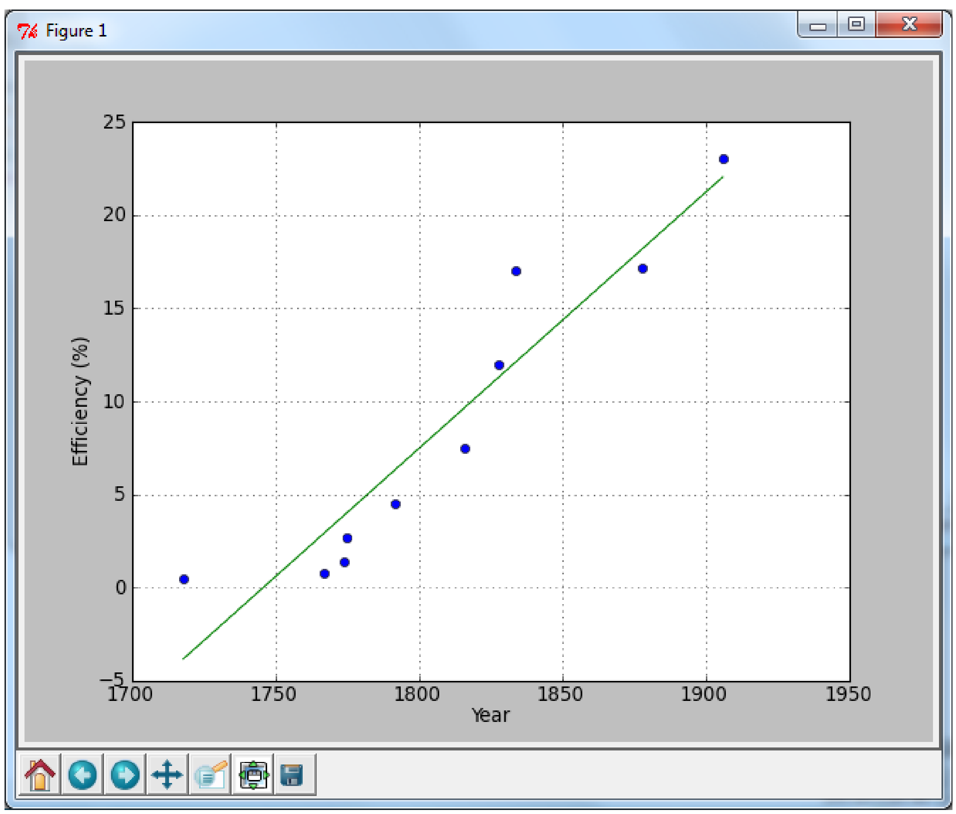


Problem 10

```
## problem3_2_10
from numpy import array
from polyFit import *
from plotPoly import *

xData = array([1718, 1767, 1774, 1775, 1792, 1816, 1828, \
```

```
                1834, 1878,1906])*1.0
yData = array([0.5, 0.8, 1.4, 2.7, 4.5, 7.5, 12.0, \
                17.0, 17.2, 23.0])
coeff = polyFit(xData,yData,1)
print("Coefficients are:\n",coeff)
print("Std. deviation =",stdDev(coeff,xData,yData))
plotPoly(xData,yData,coeff,'Year','Efficiency (%)')
input("Finished. Press return to exit")
```

```
Coefficients are:
[ -2.40391746e+02   1.37688935e-01]
Std. deviation = 2.8552022884971544
```



The predicted efficiency in year 2000 is $-240.39 + 0.13769(2000) = 35.0\%$ ◀

Problem 11

	T	$T^2 \times 10^{-3}$	$T^3 \times 10^{-6}$	$T^4 \times 10^{-9}$	k	kT	$kT^2 \times 10^{-3}$
	79	6.24	0.49	0.04	1.000	79.00	6.24
	190	36.10	6.86	1.30	0.932	177.08	33.65
	357	127.45	45.50	16.24	0.839	299.52	106.93
	524	274.58	143.88	75.39	0.759	397.72	208.40
	690	476.10	328.51	226.67	0.693	478.17	329.94
$\sum$	1840	920.47	525.24	319.64	4.223	1431.49	685.16

The coefficients **a** of the quadratic are given by the solution of the equation

$$\begin{bmatrix} 5 & \sum T & \sum T^2 \\ \sum T & \sum T^2 & \sum T^3 \\ \sum T^2 & \sum T^3 & \sum T^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum k \\ \sum kT \\ \sum kT^2 \end{bmatrix}$$

$$\begin{bmatrix} 5 & 1840 & 920.47 \times 10^3 \\ 1840 & 920.47 \times 10^3 & 525.24 \times 10^6 \\ 920.47 \times 10^3 & 525.24 \times 10^6 & 319.64 \times 10^9 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 4.223 \\ 1431.49 \\ 685.16 \times 10^3 \end{bmatrix}$$

The solution is

$$a_0 = 1.0526 \qquad a_1 = -0.6807 \times 10^{-3} \qquad a_2 = 0.2310 \times 10^{-6}$$

Thus the quadratic approximation is

$$k = 1.0526 - 0.6807 \times 10^{-3} T + 0.2310 \times 10^{-6} T^2 \quad \blacktriangleleft$$

Problem 12

$$\begin{aligned} f(x) &= ax^b \\ F(x) &= \ln f(x) = \ln a + b \ln x \end{aligned}$$

The residuals are

$$\begin{aligned} r_i &= y_i - f(x_i) = y_i - ax_i^b && \text{(a)} \\ R_i &= \ln y_i - F(x_i) = \ln y_i - \ln a - b \ln x_i = \ln \frac{y_i}{a} - b \ln x_i && \text{(b)} \end{aligned}$$

But from Eq. (a)

$$y_i - r_i = ax_i^b$$

so that

$$\begin{aligned} \ln(y_i - r_i) &= \ln a + b \ln x_i \\ b \ln x_i &= \ln \frac{y_i - r_i}{a} \end{aligned}$$

Substitution into Eq. (b) yields

$$R_i = \ln\frac{y_i}{a} - \ln\frac{y_i - r_i}{a} = \ln\frac{y_i}{y_i - r_i} = \ln\frac{1}{1 - r_i/y_i}$$

For small r_i/y_i we can approximate

$$R_i \approx \ln\left(1 + \frac{r_i}{y_i}\right) \approx \frac{r_i}{y_i} \quad \text{Q.E.D.}$$

Problem 13

i	0	1	2	3	4	5
x_i	-0.5	-0.19	0.02	0.20	0.35	0.50
y_i	-3.558	-2.874	-1.995	-1.040	-0.068	0.677

The fitting function is a linear form with

$$f_0(x) = \sin\frac{\pi t}{2} \qquad f_1(x) = \cos\frac{\pi t}{2}$$

The coefficients a and b are given by the solution of the equations

$$\begin{bmatrix} \sum_i f_0^2(x_i) & \sum_i f_0(x_i)f_1(x_i) \\ \sum_i f_0(x_i)f_1(x_i) & \sum_i f_1^2(x_i) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_i f_0(x_i)y_i \\ \sum_i f_1(x_i)y_i \end{bmatrix} \tag{a}$$

i	$f_0(x_i)$	$f_1(x_i)$	$f_0^2(x_i)$	$f_0(x_i)f_1(x_i)$	$f_1^2(x_i)$	$f_0(x_i)y_i$	$f_1(x_i)y_i$
0	-0.7071	0.7071	0.5000	-0.5000	0.5000	2.5159	-2.5159
1	-0.2940	0.9558	0.0865	-0.2810	0.9135	0.8451	-2.7469
2	0.0314	0.9995	0.0010	0.0314	0.9990	-0.0627	-1.9940
3	0.3090	0.9511	0.0955	0.2939	0.9045	-0.3214	-0.9891
4	0.5225	0.8526	0.2730	0.4455	0.7270	-0.0355	-0.0580
5	0.7071	0.7071	0.5000	0.5000	0.5000	0.4716	0.4716
$\sum$			1.4560	0.4898	4.5440	3.4130	-7.8323

Equations (a) are

$$\begin{bmatrix} 1.4560 & 0.4898 \\ 0.4898 & 4.5440 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 3.4130 \\ -7.8323 \end{bmatrix}$$

The solution is

$$a = 3.034 \qquad b = -2.051 \blacktriangleleft$$

Problem 14

i	0	1	2	3	4
x_i	0.5	1.0	1.5	2.0	2.5
y_i	0.49	1.60	3.36	6.44	10.16

Rather than fitting $y = ae^{bx}$, we use linear regression to fit $\ln y = \ln a + bx$ with the weights $W_i = y_i$.

i	$z_i = \ln y_i$	$y_i^2 x_i$	$y_i^2 z_i$	$x_i - \hat{x}$	$y_i^2 x_i(x_i - x)$	$y_i^2 z_i(x_i - x)$
0	-0.7133	0.12	-0.17	-1.7711	-0.213	0.303
1	0.4700	2.56	1.20	-1.2711	-3.254	-1.529
2	1.2119	16.93	13.68	-0.7711	-13.058	-10.551
3	1.8625	82.95	77.25	-0.2711	-22.487	-20.941
4	2.3185	258.06	239.32	0.2289	59.071	54.781
$\sum$		360.63	331.28		20.059	22.063

$$\sum_i y_i^2 = 0.49^2 + 1.60^2 + 3.36^2 + 6.44^2 + 10.16^2 = 158.79$$

$$\begin{aligned}
\hat{x} &= \frac{\sum_i y_i^2 x_i}{\sum_i y_i^2} = \frac{360.63}{158.79} = 2.2711 \\
\hat{z} &= \frac{\sum_i y_i^2 z_i}{\sum_i y_i^2} = \frac{331.28}{158.79} = 2.0863
\end{aligned}$$

$$b = \frac{\sum_i y_i^2 z_i(x_i - x)}{\sum_i y_i^2 x_i(x_i - x)} = \frac{22.063}{20.059} = 1.0999 \quad \blacktriangleleft$$

$$\begin{aligned}
\ln a &= \hat{z} - b\hat{x} = 2.0863 - 1.0999(2.2711) = -0.41168 \\
a &= e^{-0.41168} = 0.6625 \quad \blacktriangleleft
\end{aligned}$$

Problem 15

i	0	1	2	3	4
x	0.5	1.0	1.5	2.0	2.5
y	0.541	0.398	0.232	0.106	0.052

The fitting function is $y = axe^{bx}$, or $y/x = ae^{bx}$. We linearize the problem by fitting $\ln(y/x) = \ln a + bx$ with the weights $W_i = y_i$.

i	$z_i = \ln y_i/x_i$	$y_i^2 x_i$	$y_i^2 z_i$	$x_i - \hat{x}$	$y_i^2 x_i(x_i - x)$	$y_i^2 z_i(x_i - x)$
0	0.0788	0.1463	0.0231	−0.2993	−0.04380	−0.00690
1	−0.9213	0.1584	−0.1459	0.2007	0.03179	−0.02929
2	−1.8665	0.0807	−0.1005	0.7007	0.05657	−0.07039
3	−2.9375	0.0225	−0.0330	1.2007	0.02698	−0.03963
4	−3.8728	0.0068	−0.0105	1.7007	0.01150	−0.01781
$\sum$		0.4147	−0.2668		0.08304	−0.16402

$$\sum_i y_i^2 = 0.541^2 + 0.398^2 + 0.232^2 + 0.106^2 + 0.052^2 = 0.5188$$

$$\begin{aligned}
\hat{x} &= \frac{\sum_i y_i^2 x_i}{\sum_i y_i^2} = \frac{0.4147}{0.5188} = 0.7993 \\
\hat{z} &= \frac{\sum_i y_i^2 z_i}{\sum_i y_i^2} = \frac{-0.2668}{0.5188} = -0.5143
\end{aligned}$$

$$b = \frac{\sum_i y_i^2 z_i(x_i - x)}{\sum_i y_i^2 x_i(x_i - x)} = \frac{-0.16402}{0.08304} = -1.9752 \blacktriangleleft$$

$$\begin{aligned}
\ln a &= \hat{z} - b\hat{x} = -0.5143 - (-1.9752)(0.7993) = 1.0645 \\
a &= e^{\ln a} = e^{1.0645} = 2.899 \blacktriangleleft
\end{aligned}$$

Computation of standard deviation:

$$f(x) = axe^{bx} = 2.899xe^{-1.9752x}$$

i	y_i	$f(x_i)$	$[y_i - f(x_i)]^2 \times 10^6$
0	0.541	0.53998	1.047
1	0.398	0.40226	18.122
2	0.232	0.22475	52.609
3	0.106	0.11162	21.552
4	0.052	0.05197	0.000
$\sum$			103.330

$$\sigma = \sqrt{\frac{S}{5-2}} = \sqrt{\frac{103.330 \times 10^{-6}}{3}} = 5.87 \times 10^3 \blacktriangleleft$$

Problem 16

The fitting function is $\gamma = ae^{bt}$ (the value of b should come out to be negative). We linearize the problem by fitting

$$\ln \gamma = \ln a + bt \tag{a}$$

with the weights $W_i = \gamma_i$. In the program shown we use the notation $x = t$ and $y = \gamma$.
The half-life $t_{1/2}$ is obtained by substituting $\gamma = 0.5$ into Eq. (a) and solving for t, obtaining

$$t_{1/2} = \frac{\ln 0.5 - \ln a}{b}$$

```
## problem3_2_16
from numpy import array,arange,dot,log
from polyFit import *
from plotPoly import *

x = arange(0.0,5.51,0.5)
y = array([1.000, 0.994, 0.990, 0.985, 0.979, 0.977, \
               0.972, 0.969, 0.967, 0.960, 0.956, 0.952])
z = log(y)
xBar = dot(x,y**2)/dot(y,y)
zBar = dot(z,y**2)/dot(y,y)
b = dot(y**2,z*(x-xBar))/dot(y**2,x*(x-xBar))
ln_a = zBar - b*xBar
print("half_life =", (log(0.5)-ln_a)/b, "years")
input("Press return to exit")
```

```
half_life = 79.9962166337 years
```

Problem 17

The function to be minimized is

$$S(a,b,c) = \sum_{i=1}^{n} (z_i - a - bx_i - cy_i)^2$$

which yields

$$\begin{aligned}
\frac{\partial S}{\partial a} &= -2\sum_{i=1}^{n}(z_i - a - bx_i - cy_i) = 0 \\
\frac{\partial S}{\partial b} &= -2\sum_{i=1}^{n} x_i(z_i - a - bx_i - cy_i) = 0 \\
\frac{\partial S}{\partial c} &= -2\sum_{i=1}^{n} y_i(z_i - a - bx_i - cy_i) = 0
\end{aligned}$$

This can be written as

$$\begin{aligned}
na + b\sum_{i=1}^{n} x_i + c\sum_{i=1}^{n} y_i &= \sum_{i=1}^{n} z_i \\
a\sum_{i=1}^{n} x_i + b\sum_{i=1}^{n} x_i^2 + c\sum_{i=1}^{n} x_i y_i &= \sum_{i=1}^{n} x_i z_i \\
a\sum_{i=1}^{n} y_i + b\sum_{i=1}^{n} x_i y_i + c\sum_{i=1}^{n} y_i^2 &= \sum_{i=1}^{n} y_i z_i
\end{aligned}$$

Q.E.D.

Problem 18

The normal equations to be solved are

$$\begin{bmatrix} n & \Sigma x_i & \Sigma y_i \\ \Sigma x_i & \Sigma x_i^2 & \Sigma x_i y_i \\ \Sigma y_i & \Sigma x_i y_i & \Sigma y_i^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \Sigma z_i \\ \Sigma x_i z_i \\ \Sigma y_i z_i \end{bmatrix}$$

From the given data we have

$$\begin{aligned}
n &= 6 \qquad \Sigma x_i = 7 \qquad \Sigma y_i = 4 \qquad \Sigma z_i = 5.88 \\
\Sigma x_i^2 &= 13 \qquad \Sigma y_i^2 = 6 \qquad \Sigma x_i y_i = 6 \\
\Sigma x_i z_i &= 4.44 \qquad \Sigma y_i z_i = 4.55
\end{aligned}$$

Thus the normal equations are

$$\begin{bmatrix} 6 & 7 & 4 \\ 7 & 13 & 6 \\ 4 & 6 & 6 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 5.88 \\ 4.44 \\ 4.55 \end{bmatrix}$$

The solution is

$$a = 1.413 \qquad b = -0.621 \qquad c = 0.438$$

so that the fitting function becomes

$$f(x, y) = 1.413 - 0.621x + 0.438y \;\blacktriangleleft$$

PROBLEM SET 4.1

Problem 1

The problem is to find the zero of $f(x) = x^3 - 75$. With $f'(x) = 3x^2$, Newton's formula is

$$x \leftarrow x - \frac{x^3 - 75}{3x^2}$$

Starting with $x = 4$, succesive iterations yield

$$\begin{aligned}
x &\leftarrow 4 - \frac{4^3 - 75}{3(4)^2} = 4.229 \\
x &\leftarrow 4.229 - \frac{4.229^3 - 75}{3(4.229)^2} = 4.217 \\
x &\leftarrow 4.217 - \frac{4.217^3 - 75}{3(4.217)^2} = 4.217 \quad \blacktriangleleft
\end{aligned}$$

Problem 2

$$f(x) = x^3 - 3.23x^2 - 5.54x + 9.84$$

We begin with a root search starting at $x = 1$ and launch bisection once the root is bracketed.

x	$f(x)$	Interval
1.0	2.070	
1.2	0.269	
1.4	-1.503	$(1.2, 1.4)$
$(1.2 + 1.4)/2 = 1.3$	-0.624	$(1.2, 1.3)$
$(1.2 + 1.3)/2 = 1.25$	-0.179	$(1.2, 1.25)$
$(1.2 + 1.25)/2 = 1.225$	0.045	$(1.225, 1.25$
$(1.225 + 1.25)/2 = 1.2375$	-0.067	$(1.225, 1.2375)$
$(1.225 + 1.2375)/2 = 1.2313$	-0.012	$(1.225, 1.2313)$
$(1.225 + 1.2313)/2 = 1.2282$	0.017	$(1.2282, 1.2313)$
$(1.2282 + 1.2313)/2 = 1.2298$	0.002	$(1.2298, 1.2313)$
$(1.2298 + 1.2313)/2 = 1.2306$	-0.005	$(1.2298, 1.2306$
$(1.2298 + 1.2306)/2 = 1.2302$	-0.002	$(1.2298, 1.2302)$

The root is $x = 1.230$ ◄

Problem 3

$$f(x) = \cosh x \cos x - 1$$

The starting points are

$$x_1 = 4 \qquad x_2 = 5$$

The first step is bisection, giving us the point

$$x_3 = 4.5$$

Each subsequent step uses the quadratic interpolation formula

$$x = -\frac{f_2 f_3 x_1(f_2 - f_3) + f_3 f_1 x_2(f_3 - f_1) + f_1 f_2 x_3(f_1 - f_2)}{(f_1 - f_2)(f_2 - f_3)(f_3 - f_1)}$$

to compute the improved value of x, followed by reordering of data points using the following scheme:

$$\begin{aligned} \text{if } x &< x_3 : x_2 \leftarrow x_3 \\ \text{if } x &> x_3 : x_1 \leftarrow x_3 \\ x_3 &\leftarrow x \end{aligned}$$

Here are the results of the computations:

x_1	x_2	x_3	f_1	f_2	f_3	x	$f(x)$
4.000	5.000	4.500	−18.850	20.051	−10.489	4.907	12.038
4.500	5.000	4.907	−10.489	20.051	12.038	4.716	−0.818
4.500	4.907	4.716	−10.489	12.038	−0.818	4.731	0.0582
4.716	4.907	4.731	−0.818	12.038	0.0582	4.730	

Hence the root is $x = 4.730$ ◄

Problem 4

Newton's formula is

$$x \leftarrow x - \frac{f(x)}{f'(x)}$$

where

$$\begin{aligned} f(x) &= \cosh x \cos x - 1 \\ f'(x) &= \sinh x \cos x - \cosh x \sin x \end{aligned}$$

Starting with $x = 4.5$, successive applications of the formula yield

$$\begin{aligned}
x &\leftarrow 4.5 - \frac{-10.489}{34.52} = 4.804 \\
x &\leftarrow 4.804 - \frac{4.573}{66.31} = 4.735 \\
x &\leftarrow 4.735 - \frac{0.283}{58.20} = 4.730 \\
x &\leftarrow 4.730 - \frac{0.001}{57.65} = 4.730 \quad \blacktriangleleft
\end{aligned}$$

Problem 5

$$f(x) = \tan x - \tanh x$$

x	$f(x)$	Interval
7.0	-0.129	
7.4	1.049	$(7.0, 7.4)$
$(7.0 + 7.4)/2 = 7.2$	0.305	$(7.0, 7.2)$
$(7.0 + 7.2)/2 = 7.1$	0.065	$(7.0, 7.1)$
$(7.0 + 7.1)/2 = 7.05$	-0.036	$(7.05, 7.1)$
$(7.05 + 7.1)/2 = 7.075$	0.013	$(7.05, 7.075)$
$(7.05 + 7.075)/2 = 7.063$	-0.011	$(7.063, 7.075)$
$(7.063 + 7.075)/2 = 7.069$	0.000	

$$x = 7.069 \quad \blacktriangleleft$$

Problem 6

$$\begin{aligned}
f(x) &= \sin x + 3\cos x - 2 \\
f'(x) &= \cos x - 3\sin x
\end{aligned}$$

$$x \leftarrow x - \frac{f(x)}{f'(x)}$$

Starting with $x = -2$, successive applications of Newton's iterative formula yield

$$\begin{aligned}
x &\leftarrow -2 - \frac{-4.1577}{2.3117} = -0.2015 \\
x &\leftarrow -0.2015 - \frac{0.7392}{1.5801} = -0.6693 \\
x &\leftarrow -0.6693 - \frac{-0.2676}{2.6456} = -0.5682 \\
x &\leftarrow -0.5682 - \frac{-0.0093}{2.4571} = -0.5644 \\
x &\leftarrow -0.5644 - \frac{0.0000}{2.445} = -0.5644 \blacktriangleleft
\end{aligned}$$

Starting with $x = 2$, we get

$$\begin{aligned}
x &\leftarrow 2 - \frac{-2.3391}{-3.1440} = 1.2560 \\
x &\leftarrow 1.2560 - \frac{-0.1203}{-2.5430} = 1.2087 \\
x &\leftarrow 1.2087 - \frac{-0.0021}{-2.4512} = 1.2078 \\
x &\leftarrow 1.2078 - \frac{0.0000}{-2.4495} = 1.2078 \blacktriangleleft
\end{aligned}$$

Problem 7

$$f(x) = \sin x + 3\cos x - 2$$

$$x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i)$$

Start with $x_0 = -2$, $x_1 = -1.5$:

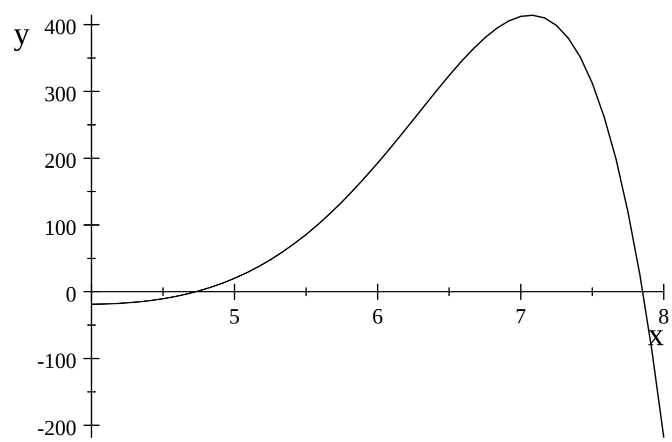
$$\begin{aligned}
x_2 &= -1.5 - \frac{-1.5 - (-2)}{-2.7853 - (-4.1577)}(-2.7853) = -0.4852 \\
x_3 &= -0.4852 - \frac{-0.4852 - (-1.5)}{0.1872 - (-2.7853)}(0.1872) = -0.5491 \\
x_4 &= -0.5491 - \frac{-0.5491 - (-0.4852)}{0.0369 - 0.1872}(0.0369) = -0.5648 \\
x_5 &= -0.5648 - \frac{-0.5648 - (-0.5491)}{-0.0013 - 0.0369}(-0.0013) = -0.5643 \\
x_6 &= -0.5643 - \frac{-0.5643 - (-0.5648)}{0.0000 - (-0.0013)}(0.0000) = -0.5643 \blacktriangleleft
\end{aligned}$$

Start with $x_0 = 2$, $x_1 = 1.5$:

$$\begin{aligned}
x_2 &= 1.5 - \frac{1.5 - 2}{-0.7903 - (-2.3391)}(-0.7903) = 1.2449 \\
x_3 &= 1.2449 - \frac{1.2449 - 1.5)}{-0.0921 - (-0.7903)}(-0.0921) = 1.2112 \\
x_4 &= 1.2112 - \frac{1.2112 - 1.2449}{-0.0083 - (-0.0921)}(-0.0083) = 1.2079 \\
x_5 &= 1.2079 - \frac{1.2079 - 1.2112}{-0.0001 - (-0.0083)}(-0.0001) = 1.2079 \blacktriangleleft
\end{aligned}$$

Problem 8

$f(x) = \cosh x \cos x - 1$



(a)

We see from the plot that a root of $f(x) = 0$ is at approximately $x = 4.75$.

(b)

The first "improved" value of x predicted by the Newton-Raphson formula is at the intersection of the tangent at $x = 4$ and the x-axis. Since the tangent is almost horizontal, the intersection point is off the right end of the plot (in $x > 8$). It is clear that subsequent iterations would keep x away from the root near 4.75.
We can confirm our findings from the Newton-Raphson formula:

$$x \leftarrow x - \frac{f(x)}{f'(x)} = 4 - \frac{f(4)}{f'(4)} = 4 - \frac{-18.85}{2.829} = 10.66$$

which is indeed "off the chart".

Problem 9

$$\begin{aligned} f(x) &= x^3 - 1.2x^2 - 8.19x + 13.23 \\ f'(x) &= 3x^2 - 2.4x - 8.19 \end{aligned}$$

In Example 4.7 it was suggested that if m is the multiplicity of the root, convergence can be improved by using the modified version

$$x \leftarrow x - m\frac{f(x)}{f'(x)}$$

of the Newton-Raphson formula (in our case $m = 2$).
Starting with $x = 2$, we get

$$\begin{aligned} x &\leftarrow 2 - 2\frac{0.05}{-0.99} = 2.1010 \\ x &\leftarrow 2.1010 - 2\frac{5.205 \times 10^{-6}}{0.0103} = 2.1000 \\ x &\leftarrow 2.1000 - 2\frac{0.000}{1.02 \times 10^{-6}} = 2.1000 \quad \blacktriangleleft \end{aligned}$$

Problem 10

```
## problem4_1_10
from math import sin,cos
from rootsearch import *
from ridder import *

def f(x):
    return x*sin(x) + 3.0*cos(x) - x

a,b,dx = (-6.0, 6.0, 0.5)
print("The roots are:")
while True:
    x1,x2 = rootsearch(f,a,b,dx)
    if x1 != None:
        a = x2
        root = ridder(f,x1,x2)
        if root != None: print(root)
    else:
        print("\nDone")
        break
```

```
input("Press return to exit")

The roots are:
-4.71238898038469
-3.2088387319804816
1.570796326794896
```

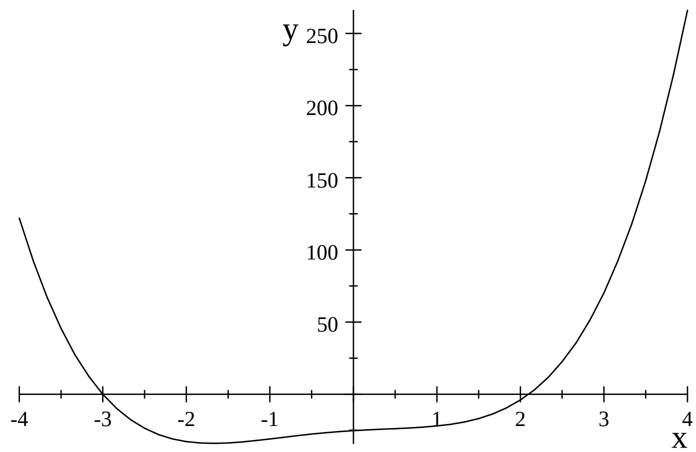
Problem 11

```
## problem4_1_11
from math import sin,cos
from rootsearch import *
from newtonRaphson import *

def f(x):
    return x*sin(x) + 3.0*cos(x) - x

def df(x):
    return x*cos(x) - 2.0*sin(x) -  1.0

a,b,dx = (-6.0, 6.0, 0.5)
print("The roots are:")
while True:
    x1,x2 = rootsearch(f,a,b,dx)
    if x1 != None:
        a = x2
        root = newtonRaphson(f,df,x1,x2)
        if root != None: print(root)
    else:
        print("\nDone")
        break
input("Press return to exit"

The roots are:
-4.712388980385274
-3.2088387319804794
1.570796326855054
```

Problem 12

$$
\begin{aligned}
f(x) &= x^4 + 0.9x^3 - 2.3x^2 + 3.6x - 25.2 \\
f'(x) &= 4x^3 + 2.7x^2 - 4.6x + 3.6
\end{aligned}
$$

Whenever possible, the function should plotted in order to gain information about its behaviour and locate its zeros.



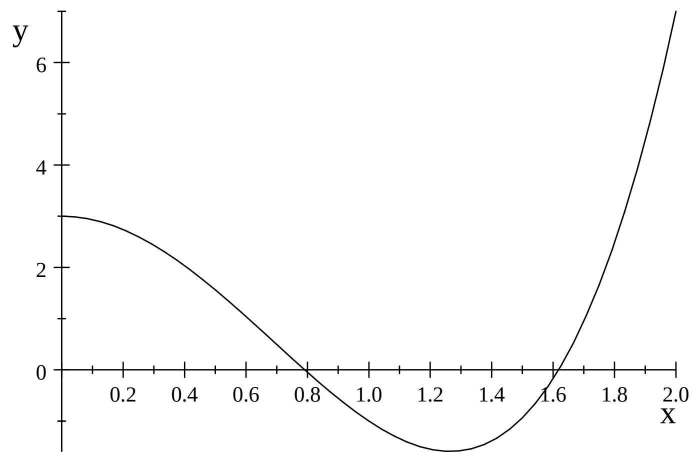
From the plot of $f(x)$ we see that there are two roots, located in $(-3.2, -2.8)$ and $(2.0, 2.4)$—note that the ticks on the x-axis are 0.4 units apart. As the derivative of the function is easily obtained, we use the Newton-Raphson method due to its superior covergence. The program below prompts for the brackets (a, b) of each root.

```
## problem4_1_12
from newtonRaphson import *

def f(x):
    return x**4 + 0.9*x**3 - 2.3*x**2 + 3.6*x - 25.2

def df(x):
    return 4.0*x**3 + 2.7*x**2 - 4.6*x + 3.6

while True:
    try: brackets = eval(input("\n(a,b) ==> "))
    except SyntaxError: break
    a,b = brackets
    print("Root =",newtonRaphson(f,df,a,b))
input("Press return to exit")

(a,b) ==> -3.2,-2.8
```

```
Root = -3.0

(a,b) ==> 2.0,2.4
Root = 2.1
```

Problem 13

$$\begin{aligned} f(x) &= x^4 + 2x^3 - 7x^2 + 3 \\ f'(x) &= 4x^3 + 6x^2 - 14x \end{aligned}$$

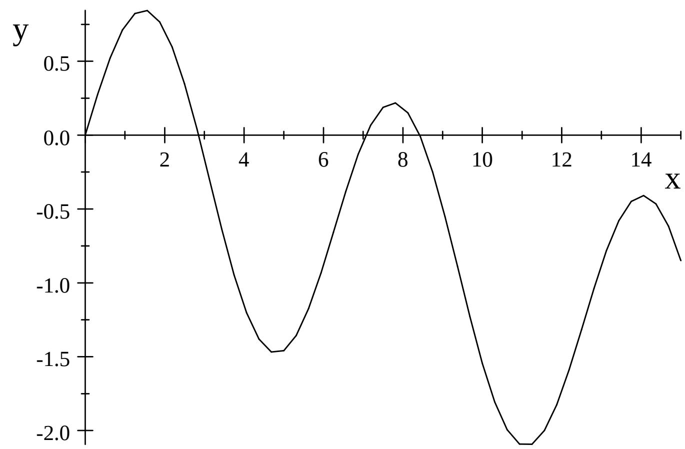


By inspection of the plot of $f(x)$ we see that there are two positive roots, located in $(0.7, 0.9)$ and $(1.5, 1.7)$. By changing the functions `f(x)` and `df(x)`, we compute the roots with the program in Problem 12. The results are

```
(a,b) ==> 0.7,0.9
Root = 0.7912878473316272

(a,b) ==> 1.5,1.7
Root = 1.6180339887508777
```

Problem 14

$$f(x) = \sin x - 0.1x$$



The plot shows that all the positive nonzero roots are in the interval $(2, 9)$. Here we take the easy way out and let the program (based on the program in Problem 10) do the bracketing as well as the computation of roots:

```
## problem4_1_14
from math import sin
from rootsearch import *
from ridder import *

def f(x):
    return sin(x) - 0.1*x

a,b,dx = (2.0, 9.0, 0.5)
print("The roots are:")
while 1:
    x1,x2 = rootsearch(f,a,b,dx)
    if x1 != None:
        a = x2
        root = ridder(f,x1,x2)
        if root != None: print (root)
    else:
        print("\nDone")
        break
input("Press return to exit")
```

```
The roots are:
2.852341894450099
7.068174358095818
8.423203932360494
```

Problem 15

$$\begin{aligned} f(\beta) &= \cosh\beta\cos\beta + 1 \\ f'(\beta) &= \sinh x\cos x - \cosh x\sin x \end{aligned}$$

This is a modification of the program in Problem 11 (Newton-Raphson method). Rather than finding all the roots in (a, b), it computes only the first n roots.

```
## problem4_1_15
from math import cosh,sinh,sin,cos
from rootsearch import *
from newtonRaphson import *

def f(x):
    return cosh(x)*cos(x) + 1.0

def df(x):
    return sinh(x)*cos(x)-cosh(x)*sin(x)

a,b,dx = (0.0, 10.0, 0.5)
n = 2
print("The roots are:")
for i in range(n):
    x1,x2 = rootsearch(f,a,b,dx)
    if x1 != None:
        a = x2
        root = newtonRaphson(f,df,x1,x2)
        if root != None: print(root)
    else:
        print("\nDone")
        break
input("Press return to exit")
```

```
The roots are:
1.8751040687119613
4.694091132974175
```

$$\begin{aligned} I &= \frac{bh^3}{12} = \frac{(25\times10^{-3})\,(2.5\times10^{-3})^3}{12} = 32.55\times10^{-12}\ \mathrm{m}^4 \\ EI &= (200\times10^9)(32.55\times10^{-12}) = 6.51\ \mathrm{N}\cdot\mathrm{m}^2 \\ m &= \rho bhL = 7850(\big(25\times10^{-3}\big)\,(2.5\times10^{-3})(0.9) = 0.4416\ \mathrm{kg} \end{aligned}$$

The natural frequencies f_i are given by

$$\beta_i^4 = (2\pi f_i)^2 \frac{mL^3}{EI}$$

$$f_i = \frac{1}{2\pi}\sqrt{\frac{EI}{mL^3}}\beta_i^2 = \frac{1}{2\pi}\sqrt{\frac{6.51}{0.4416(0.9)^3}}\beta_i^2 = 0.7157\beta_i^2$$

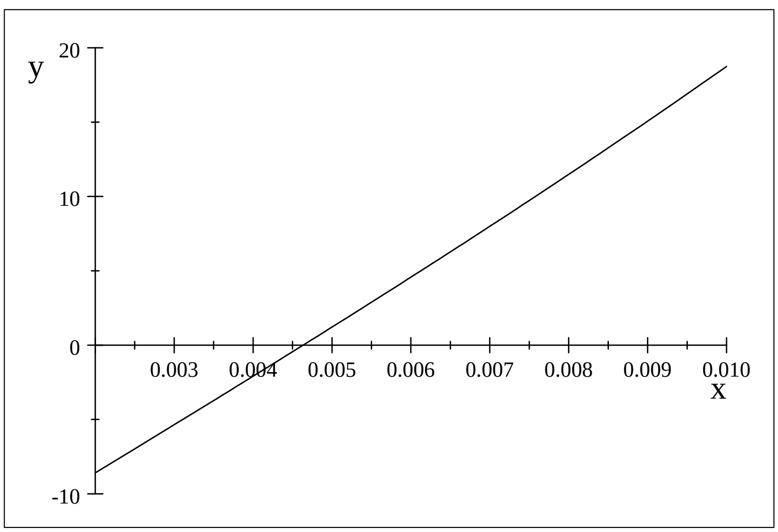
$$\begin{aligned} f_1 &= 0.7157(1.8751)^2 = 2.52 \text{ cps} \blacktriangleleft \\ f_2 &= 0.7157(4.694)^2 = 15.77 \text{ cps} \blacktriangleleft \end{aligned}$$

Problem 16

With $L = 160$ m and $h = 15$ m, the given equations become

$$s = \frac{2}{\lambda}\sinh 80\lambda \qquad f(\lambda) = \frac{1}{\lambda}(\cosh 80\lambda - 1) - 15 = 0$$

In the program shown, we first solve $f(\lambda)$ for λ, and then substitute the result into the expression for s. By plotting $f(\lambda)$, we see that there is a root is in the interval $(0.0045, 0.005)$.




```
## problem4_1_16
from ridder import *
from math import cosh,sinh

def f(lam):
    return 1.0/lam*(cosh(80*lam) - 1.0) - 15.0
```

```
lam = ridder(f,0.0045,0.005)
print("lambda =",lam)
s = 2.0/lam*sinh(80.0*lam)
print("length of cable =",s,"m")
input("Press return to exit")

lambda = 0.004634177945497287
length of cable = 163.6904403009579 m
```

Problem 17

We non-dimensionalize the secant formula by dividing both sides by E:

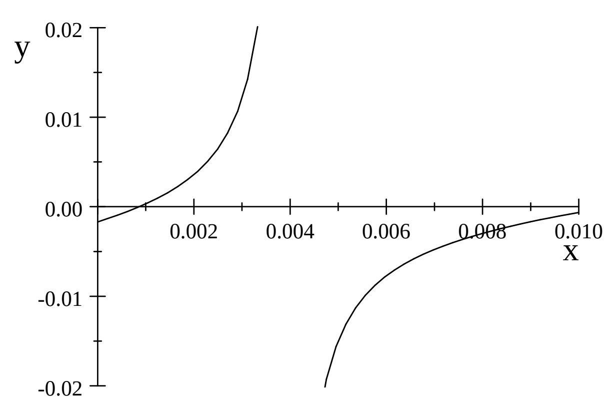
$$f\left(\frac{\bar{\sigma}}{E}\right) = \frac{\bar{\sigma}}{E}\left[1 + \frac{ec}{r^2}\sec\left(\frac{L}{2r}\sqrt{\frac{\bar{\sigma}}{E}}\right)\right] - \frac{\sigma_{\max}}{E}$$

Substituting

$$\begin{aligned} \frac{ec}{r^2} &= \frac{85(170)}{(142)^2} = 0.7166 \qquad \frac{L}{2r} = \frac{7100}{2(142)} = 25.0 \\ \frac{\sigma_{\max}}{E} &= \frac{120 \times 10^6}{71 \times 10^9} = 1.6901 \times 10^{-3} \end{aligned}$$

and using the notation $u = \bar{\sigma}/E$, the secant formula is

$$f(u) = u\left(1 + 0.7166\sec\left(25\sqrt{u}\right)\right) - 1.6901 \times 10^{-3}$$



The plot of $f(u)$ shows that the smallest root is in the interval $(0.0004, 0.0012)$. We used Brent's method to compute this root:

```
## problem4_1_17
from brent import *
from math import cos,sqrt
def f(u):
    c1 = 0.7166; c2 = 25.0; c3 = 1.6901e-3
    return u*(1.0 + c1/cos(c2*sqrt(u))) - c3

print ''u ='', brent(f,0.0004,0.0012)
raw_input(''Press return to exit'')

u = 0.0008603241318958689
```

Problem 18

Dividing both sides of the Bernoulli equation

$$\frac{Q^2}{2gb^2h_0^2} + h_0 = \frac{Q^2}{2gb^2h^2} + h + H$$

by h_0, we get

$$\frac{Q^2}{2gb^2h_0^3} + 1 = \frac{Q^2}{2gb^2h_0^3}\left(\frac{h_0}{h}\right)^2 + \frac{h}{h_0} + \frac{H}{h_0}$$

Introducing $u = h/h_0$ and rearranging, this becomes

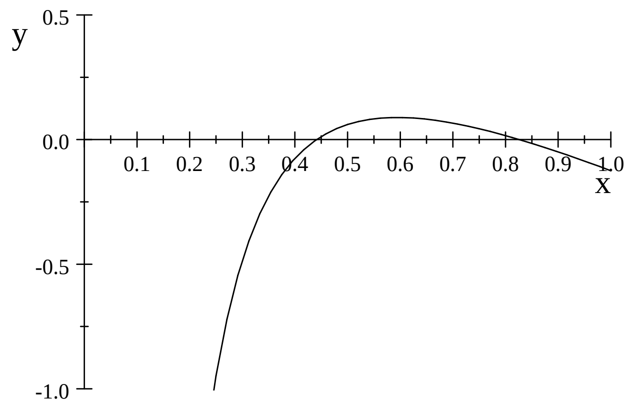
$$f(u) = \frac{Q^2}{2gb^2h_0^3}\left(1 - \frac{1}{u^2}\right) - u + \left(1 - \frac{H}{h_0}\right) = 0$$

Substituting

$$\begin{aligned} \frac{Q^2}{2gb^2h_0^3} &= \frac{(1.2)^2}{2(9.81)(1.8)^2(0.6)^3} = 0.10487 \\ 1 - \frac{H}{h_0} &= 1 - \frac{0.075}{0.6} = 0.875 \end{aligned}$$

we obtain

$$f(u) = 0.10487\left(1 - \frac{1}{u^2}\right) - u + 0.875$$



The plot of $f(u)$ indicates that there are two roots. The smaller root, which is in $(0.4, 0.48)$, can be determined with the following program:

```
## problem4_1_18
from ridder import *
def f(u):
    return 0.10487*(1.0-1.0/u**2) - u + 0.875

print("u =", ridder(f,0.4,0.48))
input("Press return to exit")
```

```
u = 0.4412494455570732
```

Problem 19

$$v = u \ln \frac{M_0}{M_0 - \dot{m}t} - gt$$

We want the root of

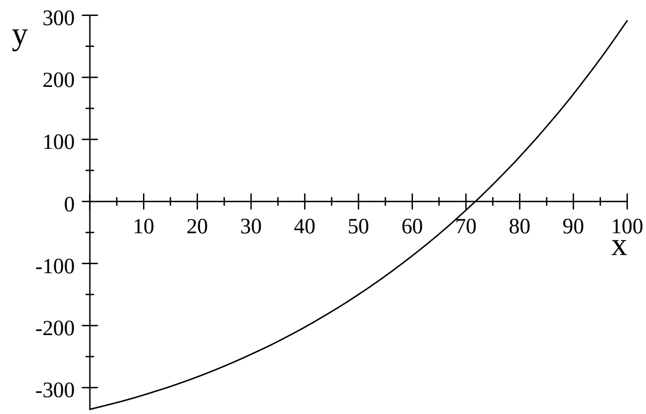
$$f(t) = u \ln \left[\frac{1}{1 - (\dot{m}/M_0)t} \right] - gt - v_{\text{sound}} = 0$$

where

$$\frac{\dot{m}}{M_0} = \frac{13.3 \times 10^3}{2.8 \times 10^6} = 0.004\,75 \text{ s}^{-1}$$

Thus

$$f(t) = 2510 \ln \left(\frac{1}{1 - 0.004\,75t} \right) - 9.81t - 335$$



The plot of $f(t)$ locates the root in $(68\text{ s}, 76\text{ s})$. The following program, based on the one in Prob. 18, was used for the computation of the root:

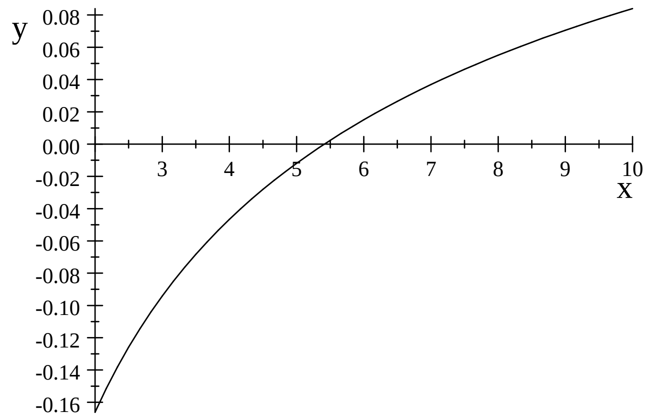
```
## problem4_1_19
from ridder import *
from math import log
def f(t):
    return 2510.0*log(1.0/(1.0-0.00475*t)) \
           - 9.81*t - 335.0

print("t =", ridder(f,68.0,76.0),"sec")
input("Press return to exit")

t = 70.87797226808104 sec
```

Problem 20

$$\eta = \frac{\ln(T_2/T_1) - (1 - T_1/T_2)}{\ln(T_2/T_1) + (1 - T_1/T_2)/(\gamma - 1)}$$

With $\eta = 0.3$, $\gamma - 1 = 2/3$ and the notation $u = T_2/T_1$, the equation becomes

$$f(u) = \frac{\ln u - (1 - 1/u)}{\ln u + 1.5(1 - 1/u)} - 0.3 = 0$$



From the plot of $f(u)$ we see that the root is in $(5.2, 5.6)$. We found this root with the following program:

```
## problem4_1_20
from ridder import *
from math import log
def f(u):
    numerator = log(u) - (1.0 -1.0/u)
    denominator = log(u) + 1.5*(1.0 - 1.0/u)
    return numerator/denominator - 0.3

print("T2/T1 =", ridder(f,5.2,5.6))
input("Press return to exit")

T2/T1 = 5.412548241399093
```

Problem 21

```
## problem4_1_21
from math import sin,cos,atan,sqrt
from ridder import *
from rootsearch import *

def f(zeta):
    Z  = a**2/sqrt((1.0 - a**2)**2 + (2.0*zeta*a)**2)
    phi = atan(2.0*zeta*a/(1.0 - a**2))
    X = sqrt((1.0 + Z*cos(phi))**2 + (Z*sin(phi))**2)
    return X  - 1.5

m = 0.2; k = 2880.0; omega = 96.0
```

```
p = sqrt(k/m)
a = omega/p

zeta1,zeta2 = rootsearch(f, 0.0, 50.0, 1.0)
c = ridder(f,zeta1,zeta2)*2.0*m*p
print('c =',c,'N.s/m')
input("Press return to exit")

c = 22.584242294130664 N.s/m
```

Problem 22

The volume of the tank is $V_0 = \pi r^2 L$. When the tank is 3/4 full, we have $V = 0.75V_0$, or

$$\phi - \left(1 - \frac{h}{r}\right)\sin\phi = 0.75\pi$$

Letting $\alpha = 1 - h/r$, the equation to be solved is (the unknown is α)

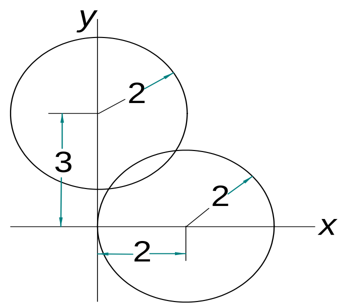
$$0.75\pi - (\phi - \alpha\sin\phi) = 0$$

where $\phi = \cos^{-1}\alpha$.

```
## problem4_1_22
from math import sin,acos,pi
from ridder import *

def f(alpha):  # Notation: alpha = 1 - h/r
    phi = acos(alpha)
    return 0.75*pi - (phi -  alpha*sin(phi))

alpha = ridder(f, -1.0, 1.0)
print('h/r =',1.0 - alpha)
input("Press return to exit")

h/r = 1.4039727532995172
```

Problem 23

$$\begin{aligned} f_1(x,y) &= (x-2)^2 + y^2 - 4 \\ f_2(x,y) &= x^2 + (y-3)^2 - 4 \end{aligned}$$



The rough locations of the intersection points are $(2, 2)$ and $(0, 1)$. Letting $x = x_0$ and $y = x_1$, the following program computes the coordinates of the first point:

```
## problem4_1_23
from numpy import zeros,array
from newtonRaphson2 import *

def f(x):
    f = zeros(len(x))
    f[0] = (x[0] - 2.0)**2 + x[1]**2 - 4.0
    f[1] = x[0]**2 + (x[1] - 3.0)**2 - 4.0
    return f

x = array([2.0, 2.0])
print("(x,y) =",newtonRaphson2(f,x))
input ("\nPress return to exit")

x = array([2.0, 2.0])
print ''(x,y) ='',newtonRaphson2(f,x)
raw_input (''\nPress return to exit'')
```

```
(x,y) = [ 1.72057669  1.98038446]
```

Changing the starting point to $(0.0, 1.0)$, the same program yields for the second point

```
(x,y) = [ 0.27942331  1.01961554]
```

Problem 24

$$\begin{aligned} f_1(x,y) &= \sin x + 3\cos x - 2 \\ f_2(x,y) &= \cos x - \sin y + 0.2 \end{aligned}$$

The following program uses the notation $x = x_0$ and $y = x_1$:

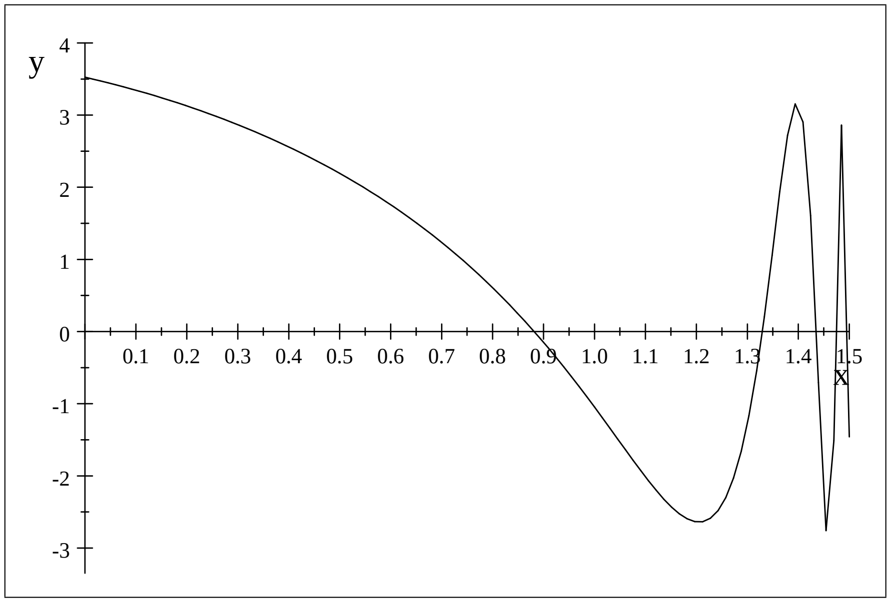
```
## problem4_1_24
from numpy import zeros,array
from math import sin,cos
from newtonRaphson2 import *

def f(x):
    f = zeros(len(x))
    f[0] = sin(x[0]) + 3.0*cos(x[1]) - 2.0
    f[1] = cos(x[0]) - sin(x[1]) + 0.2
    return f

x = array([1.0, 1.0])
print("(x,y) =",newtonRaphson2(f,x))
input ("\nPress return to exit")

(x,y) = [ 0.7911678   1.12673723]
```

Problem 25

$$\begin{aligned} \tan x - y &= 1 \\ \cos x - 3\sin y &= 0 \end{aligned}$$

It is very difficult to search for the roots of simultaneous equations. Here we can overcome the difficulty by solving the first equation for y and substituting the result into the second equation. This gives us the single transcendental equation

$$f(x) = \cos x - 3\sin(\tan x - 1) = 0$$



From the plot of $f(x)$ we see that there are 5 roots in the interval $(0, 1.5)$. The first root is about $x = 0.88$; the other roots are closely spaced near the end of the interval (the spacing of roots becomes infinitesimal at $x = \pi/2$). The program listed below searches for the roots from $x = 0.8$ to 1.5 in increments of 0.025 (the increment has to be small in order to catch all the roots). Once a root is bracketed, it is computed with Ridder's method.

```
## problem4_1_25
from math import tan,cos,sin
from rootsearch import *
from ridder import *

def f(x):
    return cos(x)-3.0*sin(tan(x) - 1.0)

a,b,dx = (0.8, 1.5, 0.025)
print("The roots are:")
while True:
    x1,x2 = rootsearch(f,a,b,dx)
    if x1 != None:
        a = x2
        x = ridder(f,x1,x2)
        if x != None:
            y = tan(x) - 1.0
            print("(x,y) =",x,y)
    else:
        print("\nDone")
        break
input("Press return to exit"))
```

```
The roots are:
(x,y) = 0.8815925944959485 0.21359471457166057
(x,y) = 1.3294021265325462 3.061822535805115
(x,y) = 1.435176095358017 6.328268868969814
(x,y) = 1.4748716040430452 9.392846641421915
(x,y) = 1.4973496737104217 12.590833266567566
```

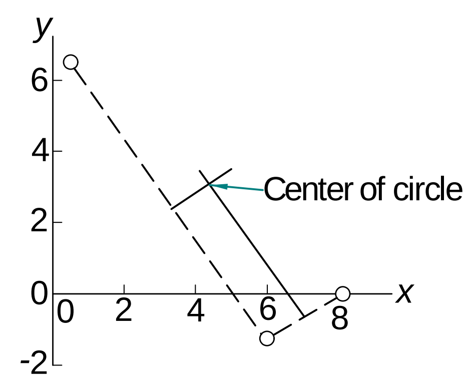
Problem 26

$$(x - a)^2 + (y - b)^2 - R^2 = 0$$

Substituting the coordinates of the given points into the above equation, we get

$$\begin{aligned}
(8.21 - a)^2 + b^2 - R^2 &= 0 \\
(0.34 - a)^2 + (6.62 - b)^2 - R^2 &= 0 \\
(5.96 - a)^2 + (-1.12 - b)^2 - R^2 &= 0
\end{aligned}$$

By plotting the points, we can estimate the parameters of the circle. It appears that reasonable starting values are $a = 5$, $b = 3$ and $R = 5$.



Here is the program that refines the above values (the notation $a = x_0$, $b = x_1$, $R = x_2$ was used):

```
## problem4_1_26
from numpy import zeros,array
from newtonRaphson2 import *

def f(x):
    f = zeros(len(x))
    f[0] = (8.21 - x[0])**2 + x[1]**2 - x[2]**2
```

```
    f[1] = (0.34 - x[0])**2 + (6.62 - x[1])**2 - x[2]**2
    f[2] = (5.96 - x[0])**2 + (-1.12 - x[1])**2 - x[2]**2
    return f

x = array([5.0, 3.0, 5.0])
print("(a,b,R) =",newtonRaphson2(f,x))
input ("\nPress return to exit")

(a,b,R) = [ 4.83010565  3.96992168  5.21382431]
```

Problem 27

$$\frac{C}{1+e\sin(\theta+\alpha)}-R=0$$

After substituting the three sets of given data, we obtain the simulataneous equations

$$\begin{aligned}\frac{C}{1+e\sin(-\pi/6+\alpha)}-6870 &= 0\\ \frac{C}{1+e\sin(\alpha)}-6728 &= 0\\ \frac{C}{1+e\sin(\pi/6+\alpha)}-6615 &= 0\end{aligned}$$

The starting value $C = 6800$ seems reasonable, but e and α are not easy to guess. The orbit has some eccentricity, so that $e = 0.5$ should not be out of line ($e = 0$ will not work because it results in a singular Jacobian matrix). We also used $\alpha = 0$, which was a pure guess.
The minimum value of R is

$$R_{\min}=\frac{C}{1+e}$$

occuring at

$$\sin(\theta+\alpha)=1 \qquad \theta=\frac{\pi}{2}-\alpha$$

With the notation $C = x_0$, $e = x_1$ and $\alpha = x_2$, we arrive at the following program:

```
## problem4_1_27
from numpy import zeros,array
from math import sin,pi
from newtonRaphson2 import *

def f(x):
```

```
    f = zeros(len(x))
    f[0] = x[0]/(1.0 + x[1]*sin(-pi/6.0 + x[2])) - 6870.0
    f[1] = x[0]/(1.0 + x[1]*sin(x[2])) - 6728.0
    f[2] = x[0]/(1.0 + x[1]*sin(pi/6.0 + x[2])) - 6615.0
    return f

x = array([6800, 0.5, 0.0])
x= newtonRaphson2(f,x)
print("(C,e,alpha) =",x)
print("Rmin =",x[0]/(1.0 + x[1]),"km")
print("Theta =",(pi/2.0 - x[2])*180.0/pi,"deg")
input("\nPress return to exit")

(C,e,alpha) = [ 6.81929379e+03  4.05989591e-02  3.40783998e-01]
Rmin = 6553.23910701 km
Theta = 70.4745151918 deg
```

Problem 28

$$\begin{aligned} x &= (v\cos\theta)t \\ y &= -\frac{1}{2}gt^2 + (v\sin\theta)t \end{aligned}$$

We also need the expression for dy/dx:

$$\begin{aligned} \frac{dx}{dt} &= v\cos\theta \qquad \frac{dy}{dt} = -gt + v\sin\theta \\ \frac{dy}{dx} &= \frac{-gt + v\sin\theta}{v\cos\theta} \end{aligned}$$

Letting τ denote the time of flight, the specified end conditions are

$$x(\tau) = 300 \text{ m} \qquad y(\tau) = 61 \text{ m} \qquad \left.\frac{dy}{dx}\right|_{\tau} = -1$$

which yield the equations

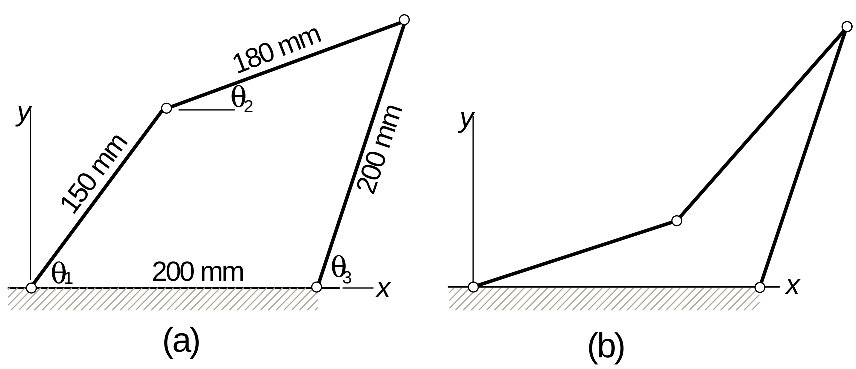
$$\begin{aligned} (v\cos\theta)\,\tau - 300 &= 0 \\ -\frac{1}{2}g\tau^2 + (v\sin\theta)\,\tau - 61 &= 0 \\ \frac{-g\tau + v\sin\theta}{v\cos\theta} + 1 &= 0 \end{aligned}$$

To estimate the initial values of the unknowns, we guess $\tau = 10$ s and $\theta = 45°$. Then the first of the above equations yields $v = 300/(\tau\cos\theta) \approx 57$ m/s. The following program uses the notation $\tau = x_0$, $v = x_1$ and $\theta = x_2$:

```
## problem4_1_28
from numpy import zeros,array
from math import sin,cos,pi
from newtonRaphson2 import *

def f(x):
    f = zeros(len(x))
    g = 9.81
    vc = x[1]*cos(x[2]); vs = x[1]*sin(x[2])
    f[0] = vc*x[0] - 300.0
    f[1] = -0.5*g*x[0]**2 + vs*x[0] - 61.0
    f[2] = (-g*x[0] + vs)/vc + 1.0
    return f

x = array([10.0, 57.0, pi/4.0])
t,v,theta = newtonRaphson2(f,x)
print("Time of flight =",t,"s")
print("Launch speed =",v,"m/s")
print("Angle of elevation =",theta*180.0/pi,"deg")
input("\nPress return to exit")

Time of flight = 8.57894917873 s
Launch speed = 60.3533459817 m/s
Angle of elevation = 54.5909609221 deg
```

Problem 29

$$
\begin{aligned}
150\cos\theta_1 + 180\cos\theta_2 - 200\cos\theta_3 &= 200 \\
150\sin\theta_1 + 180\sin\theta_2 - 200\sin\theta_3 &= 0
\end{aligned}
$$



We estimate from the figure that $\theta_1 = 60^\circ$ and $\theta_2 = 20^\circ$ in configuration (a). Here is the program that refines these values:

```
## problem4_1_29
from numpy import zeros,array
from math import sin,cos,pi
from newtonRaphson2 import *

def f(x):
    f = zeros(len(x))
    theta3 = 75.0*pi/180.0
    f[0] = 150.0*cos(x[0]) + 180.0*cos(x[1]) \
           - 200.0*(cos(theta3) + 1.0)
    f[1] = 150.0*sin(x[0]) + 180.0*sin(x[1]) \
           - 200.0*sin(theta3)
    return f

x = array([20.0, 50.0])*pi/180.0
theta1,theta2 = newtonRaphson2(f,x)
print("Theta1 =",theta1*180.0/pi,"deg")
print("Theta2 =",theta2*180.0/pi,"deg")
input("\nPress return to exit")
```

```
Theta1 = 54.9812167844 deg
Theta2 = 23.0031009327 deg
```

For configuration (b) we estimate $\theta_1 = 20^\circ$ and $\theta_2 = 50^\circ$. With these starting values the above program yields

```
Theta1 = 20.0187832156 deg
Theta2 = 51.9968990673 deg
```

Problem 30

Letting $\mathbf{x} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & T \end{bmatrix}^T$ results in the program shown below. Rough estimates (starting values) of the variables are

$$\theta_1 = 0.8 \text{ rad} \qquad \theta_2 = 0.3 \text{ rad} \qquad \theta_3 = 0.4 \text{ rad} \qquad T = 20 \text{ kN}$$

```
## problem4_1_30
from math import sin,cos,tan
from newtonRaphson2 import *
from numpy import zeros,array

def f(x):
    y = zeros(4)
```

```
    y[0] = x[3]*(-tan(x[1]) + tan(x[0])) - 16.0
    y[1] = x[3]*( tan(x[2]) + tan(x[1])) - 20.0
    y[2] = -4.0*sin(x[0]) - 6.0*sin(x[1]) + 5.0*sin(x[2]) + 3.0
    y[3] = 4.0*cos(x[0]) + 6.0*cos(x[1]) + 5.0*cos(x[2]) - 12.0
    return y

xStart = array([0.8, 0.3, 0.4, 20.0])
print("Solution is ",newtonRaphson2(f,xStart))
input("\nPress return to exit")
```

```
Solution is  [  0.93580275   0.43344988   0.58004954  17.88840896]
```

Therefore, the results are

$$
\begin{aligned}
\theta_1 &= 0.9358 \text{ rad } = 53.62^\circ \ \blacktriangleleft \\
\theta_2 &= 0.4334 \text{ rad } = 24.83^\circ \ \blacktriangleleft \\
\theta_3 &= 0.5800 \text{ rad } = 33.23^\circ \ \blacktriangleleft \\
T &= 17.89 \text{ kN}
\end{aligned}
$$

Problem 31

```
##problem4_1_31
from rational import *
from ridder import *
from numpy import array

xData = array([0.0,  0.25,   0.50,   0.75,   1.0,   1.25,   1.5])
yData = array([0.0, -1.223, -2.269, -2.842, -2.213, 2.548, 55.507])
def f(x): return rational(xData,yData,x)
print('Root =',ridder(f,1.0,1.25))
input("Press return to exit")
```

```
Root = 1.16558232236
```

PROBLEM SET 4.2

Problem 1

$$P_3(x) = 3x^3 + 7x^2 - 36x + 20 \qquad r = -5$$

$$\begin{aligned} b_2 &= a_3 = 3 \\ b_1 &= a_2 + rb_2 = 7 + (-5)(3) = -8 \\ b_0 &= a_1 + rb_1 = -36 + (-5)(-8) = 4 \end{aligned}$$

$$\therefore P_2 = 3x^2 - 8x + 4 \;\blacktriangleleft$$

Problem 2

$$P_4(x) = x^4 - 3x^2 + 3x - 1 \qquad r = 1$$

$$\begin{aligned} b_3 &= a_4 = 1 \\ b_2 &= a_3 + rb_3 = 0 + 1(1) = 1 \\ b_1 &= a_2 + rb_2 = -3 + 1(1) = -2 \\ b_0 &= a_1 + rb_1 = 3 + 1(-2) = 1 \end{aligned}$$

$$\therefore P_3 = x^3 + x^2 - 2x + 1 \;\blacktriangleleft$$

Problem 3

$$P_5(x) = x^5 - 30x^4 + 361x^3 - 2178x^2 + 6588x - 7992 \qquad r = 6$$

$$\begin{aligned} b_4 &= a_5 = 1 \\ b_3 &= a_4 + rb_4 = -30 + 6(1) = -24 \\ b_2 &= a_3 + rb_3 = 361 + 6(-24) = 217 \\ b_1 &= a_2 + rb_2 = -2178 + 6(217) = -876 \\ b_0 &= a_1 + rb_1 = 6588 + 6(-876) = 1332 \end{aligned}$$

$$P_4(x) = x^4 - 24x^3 + 217x^2 - 876x + 1332$$

Problem 4

$$P_4(x) = x^4 - 5x^3 - 2x^2 - 20x - 24 \qquad r = 2i$$

$$\begin{aligned}
b_3 &= a_4 = 1 \\
b_2 &= a_3 + rb_3 = -5 + (2i)(1) = -5 + 2i \\
b_1 &= a_2 + rb_2 = -2 + (2i)(-5 + 2i) = -6 - 10i \\
b_0 &= a_1 + rb_1 = -20 + (2i)(-6 - 10i) = -12i
\end{aligned}$$

$$P_3(x) = x^3 - (5 - 2i)x^2 - (6 + 10i)x - 12i \quad \blacktriangleleft$$

Problem 5

$$P_3(x) = 3x^3 - 19x^2 + 45x - 13 \qquad r = 3 - 2i$$

$$\begin{aligned}
b_2 &= a_3 = 3 \\
b_1 &= a_2 + rb_2 = -19 + (3 - 2i)(3) = -10 - 6i \\
b_0 &= a_1 + rb_1 = 45 + (3 - 2i)(-10 - 6i) = 3 + 2i
\end{aligned}$$

$$P_2(x) = 3x^2 - (10 + 6i)\,x + (3 + 2i) \quad \blacktriangleleft$$

Problem 6

$$P_3(x) = x^3 + 1.8x^2 - 9.01x - 13.398 \qquad r = -3.3$$

$$\begin{aligned}
b_2 &= a_3 = 1 \\
b_1 &= a_2 + rb_2 = 1.8 + (-3.3)(1) = -1.5 \\
b_0 &= a_1 + rb_1 = -9.01 + (-3.3)(-1.5) = -4.06
\end{aligned}$$

$$P_2(x) = x^2 - 1.5x - 4.06$$

The roots are

$$r = \frac{1.5 \pm \sqrt{1.5^2 + 4(4.06)}}{2} = \frac{1.5 \pm 4.3}{2} = \begin{cases} 2.9 \\ -1.4 \end{cases} \quad \blacktriangleleft$$

Problem 7

$$P_3(x) = x^3 - 6.64x^2 + 16.84x - 8.32 \qquad r = 0.64$$

$$\begin{aligned}
b_2 &= a_3 = 1 \\
b_1 &= a_2 + rb_2 = -6.64 + 0.64(1) = -6 \\
b_0 &= a_1 + rb_1 = 16.84 + 0.64(-6.0) = 13
\end{aligned}$$

$$P_2(x) = x^2 - 6x + 13$$

The roots are

$$r = \frac{6 \pm \sqrt{6^2 - 4(13)}}{2} = \frac{6 \pm 4i}{2} = \begin{cases} 3 + 2i \\ 3 - 2i \end{cases} \quad \blacktriangleleft$$

Problem 8

$$P_3(x) = 2x^3 - 13x^2 + 32x - 13 \qquad r = 3 - 2i$$

$$\begin{aligned}
b_2 &= a_3 = 2 \\
b_1 &= a_2 + rb_2 = -13 + (3 - 2i)(2) = -7 - 4i \\
b_0 &= a_1 + rb_1 = 32 + (3 - 2i)(-7 - 4i) = 3 + 2i
\end{aligned}$$

$$P_2(x) = 2x^2 - (7 + 4i)x + (3 + 2i)$$

Since complex roots come in conjugate pairs, we know that a zero of $P_2(x)$ is

$$r = 3 + 2i \quad \blacktriangleleft$$

$$\begin{aligned}
b_1 &= a_2 = 2 \\
b_0 &= a_1 + rb_1 = -(7 + 4i) + (3 + 2i)(2) = -1
\end{aligned}$$

$$P_1(x) = -1 + 2x$$

The zero of $P_1(x)$ is

$$r = 0.5 \quad \blacktriangleleft$$

Problem 9

$$P_4(x) = x^4 - 3x^3 + 10x^2 - 6x - 20 \qquad r = 1 + 3i$$

$$\begin{aligned}
b_3 &= a_4 = 1 \\
b_2 &= a_3 + rb_3 = -3 + (1+3i)(1) = -2 + 3i \\
b_1 &= a_2 + rb_2 = 10 + (1+3i)(-2+3i) = -1 - 3i \\
b_0 &= a_1 + rb_1 = -6 + (1+3i)(-1-3i) = 2 - 6i
\end{aligned}$$

$$P_3(x) = x^3 + (-2+3i)x^2 + (-1-3i)x + (2-6i)$$

Another zero of $P_4(x)$ is the conjugate of $1 + 3i$, namely

$$r = 1 - 3i$$

$$\begin{aligned}
b_2 &= a_3 = 1 \\
b_1 &= a_2 + rb_2 = (-2+3i) + (1-3i)(1) = -1 \\
b_0 &= a_1 + rb_1 = (-1-3i) + (1-3i)(-1) = -2
\end{aligned}$$

$$P_2(x) = x^2 - x + -2$$

The roots of the quadratic are

$$r = \frac{1 \pm \sqrt{1^2 + 4(2)}}{2} = \begin{cases} 2 \\ -1 \end{cases}$$

Thus the roots of $P_4(x)$ are $1 \pm 3i$, 2 and -1. ◄

Problem 10

```
## problem4_2_10
from polyRoots import *
from numpy import array

a = array([-3.52, 2.1, -2.52, 2.1, 1.0])
print(" Real part    Imaginary part")
r = polyRoots(a)
for i in range(len(a)-1):
    rReal = r[i].real; rImag = r[i].imag
    if abs(rReal) < 1.0e-12: rReal = 0.0
    print("{:12.4e} {:12.4e}".format(rReal,rImag))
```

```
input("Press return to exit")
```

```
 Real part    Imaginary part
 1.1000e+000  0.0000e+000
 0.0000e+000 -1.0000e+000
 0.0000e+000  1.0000e+000
-3.2000e+000  0.0000e+000
```

Problem 11

We used the program in Problem 10 with

```
a = array([-624, 4, 780, -5, -156, 1],float)
```

```
 Real part    Imaginary part
 1.0000e+000  0.0000e+000
-1.0000e+000  0.0000e+000
 2.0000e+000  0.0000e+000
-2.0000e+000  0.0000e+000
 1.5600e+002  0.0000e+000
```

Problem 12

We used the program in Problem 10 with

```
a = array([-150, 130, 57, -34, -8, 4, 1],float)
```

```
 Real part    Imaginary part
 1.0000e+000  0.0000e+000
 2.0000e+000 -1.0000e+000
-3.0000e+000 -1.0000e+000
 2.0000e+000  1.0000e+000
-3.0000e+000  0.0000e+000
-3.0000e+000  1.0000e+000
```

Problem 13

We used the program in Problem 10 with

```
a = array([-100, 220, 19, -124, -13, 34, 28, 8],float)

Real part    Imaginary part
  5.0000e-01   0.0000e+00
  1.0000e+00  -5.0000e-01
  1.0000e+00   5.0000e-01
 -1.0000e+00  -2.0000e+00
 -2.0000e+00  -2.1386e-08
 -1.0000e+00   2.0000e+00
 -2.0000e+00   2.1386e-08
```

The double root ($x = 2$) contains an erroneous imaginary part that is small, but nevertheless considerably larger than the cutoff criterion of 10^{-12} used in `polyRoots`. Double roots often cause difficulties in numerical algorithms.

Problem 14

We used the program in Problem 10 with

```
a = array([-6.0*(1.0-1.0j),1.0,-6.0*(1.0+1.0j),2.0])

Real part    Imaginary part
  8.9745e-01   7.5566e-01
 -6.0964e-01  -6.7275e-01
  2.7122e+00   2.9171e+00
```

Problem 15

We used the program in Problem 10 with

```
a = array([-84, 30-14j, -8+5j, 5+1j, 1],float)
```

```
 Real part    Imaginary part
 2.0000e+000  0.0000e+000
 0.0000e+000  2.0000e+000
 0.0000e+000 -3.0000e+000
-7.0000e+000  0.0000e+000
```

Note that the complex roots do not appear as conjugate pairs if the coefficients of the polynomial are not real.

Problem 16

$$\omega^4 + 2\frac{c}{m}\omega^3 + 3\frac{k}{m}\omega^2 + \frac{c}{m}\frac{k}{m}\omega + \left(\frac{k}{m}\right)^2 = 0$$

With $c/m = 12\ \mathrm{s}^{-1}$ and $k/m = 1500\ \mathrm{s}^{-2}$, we get

$$\omega^4 + 24\omega^3 + 4500\omega^2 + 18 \times 10^3\omega + 2.25 \times 10^6 = 0$$

We used the program in Problem 10 with

```
a = array([2.25e6, 18.0e3, 4500.0, 24.0, 1.0])
```

```
Real part    Imaginary part
-6.2302e-001 -2.4030e+001
-6.2302e-001  2.4030e+001
-1.1377e+001  6.1354e+001
-1.1377e+001 -6.1354e+001
```

Therefore, the two combinations of (ω_r, ω_i) are

$$\left(-0.0623\ \mathrm{s}^{-1}, 24.03\ \mathrm{s}^{-1}\right) \text{ and } \left(-11.38\ \mathrm{s}^{-1}, 61.35\ \mathrm{s}^{-1}\right) \blacktriangleleft$$

Problem 17

The slope of the beam is

$$\begin{aligned} y' &= \frac{w_0}{120EI}(5x^4 - 9L^2x^2 + 6L^3x) \\ &= \frac{w_0L^4}{120EI}(5\xi^4 - 9\xi^2 + 6\xi) \end{aligned}$$

where $\xi = x/L$. Since $y' = 0$ at the point of maximum displacement, the value of ξ that we are looking for is a root of

$$P_4(\xi) = 5\xi^4 - 9\xi^2 + 6\xi$$

We could find the our roots of this equation with the function `polyroots`, but this is unnecessary. Because the slope of the beam is zero at supports, we know that two of the roots are $\xi = 0$ and $\xi = 1$. Factoring out these roots, we have

$$P_4(\xi) = \xi(\xi - 1)(b_1\xi^2 + b_2\xi + b_3)$$

The b's are obtained by Horner's algorithm:

$$\begin{aligned} b_1 &= a_1 = 5 \\ b_2 &= a_2 + rb_1 = 0 + 1(5) = 5 \\ b_3 &= a_3 + rb_2 = -9 + 1(5) = -4 \end{aligned}$$

We have now reduced the problem to finding the roots of the quadratic equation

$$5\xi^2 + 5\xi - 4 = 0$$

The positive root is

$$\xi = \frac{-5 + \sqrt{5^2 - 4(5)(-4)}}{10} = 0.5247 \ \blacktriangleleft$$

PROBLEM SET 5.1

Problem 1

$$\begin{aligned} f(x-h_1) &= f(x) - f'(x)h_1 + \frac{1}{2}f''(x)h_1^2 - \frac{1}{6}f'''(x)h_1^3 + \cdots \\ f(x+h_2) &= f(x) + f'(x)h_2 + \frac{1}{2}f''(x)h_2^2 + \frac{1}{6}f'''(x)h_2^3 + \cdots \end{aligned}$$

Multiplying the first expression by h_2/h_1 and adding it to the second expression yields

$$\begin{aligned} &\frac{h_2}{h_1}f(x-h_1) + f(x+h_2) \\ = &\left(\frac{h_2}{h_1}+1\right)f(x) + \frac{1}{2}f''(x)\left(\frac{h_2}{h_1}h_1^2 + h_2^2\right) + \frac{1}{6}f'''(x)\left(\frac{h_2}{h_1}h_2^3 - h_1^3\right) + \cdots \end{aligned}$$

$$f''(x) = \frac{\dfrac{h_2}{h_1}f(x-h_1) - \left(\dfrac{h_2}{h_1}+1\right)f(x) + f(x+h_2)}{\dfrac{h_2}{h_1}\left(1+\dfrac{h_2}{h_1}\right)\dfrac{h_1^2}{2}} + \mathcal{O}(h) \quad \blacktriangleleft$$

Problem 2

$$\begin{aligned} f'''(x) &= [f''(x)]' = \left[\frac{f(x-2h) - 2f(x-h) + f(x)}{h^2}\right]' \\ &= \frac{1}{h^2}\left[f'(x-2h) - 2f'(x-h) + f'(x)\right] \\ &= \frac{1}{h^2}\left[\frac{f(x-2h)-f(x-3h)}{h} - 2\frac{f(x-h)-f(x-2h)}{h} + \frac{f(x)-f(x-h)}{h}\right] \\ &= \frac{-f(x-3h) + 3f(x-2h) - 3f(x-h) + f(x)}{h^3} \quad \blacktriangleleft \end{aligned}$$

Problem 3

Central difference approximations for $f''(x)$ of $\mathcal{O}(h^2)$ are

$$\begin{aligned} g(h) &= \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \\ g(2h) &= \frac{f(x+2h) - 2f(x) + f(x-2h)}{(2h)^2} \end{aligned}$$

Richardson's extrapolation gives us an approximation of $\mathcal{O}(h^4)$:

$$\begin{aligned} f''(x) &\approx \frac{4g(h) - g(2h)}{4-1} \\ &= \frac{4}{2}\frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{1}{12}\frac{f(x+2h) - 2f(x) + f(x-2h)}{h^2} \\ &= \frac{-f(x+2h) + 16f(x+h) - 30f(x) + 16f(x-h) - f(x-2h)}{12h^2} \quad \blacktriangleleft \end{aligned}$$

Problem 4

Taylor series:

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \cdots && \text{(a)} \\ f(x+2h) &= f(x) + 2hf'(x) + 2h^2 f''(x) + \frac{4h^3}{3}f'''(x) + \cdots) && \text{(b)} \\ f(x+3x) &= f(x) + 3hf'(x) + \frac{9h^2}{2}f''(x) + \frac{9h^3}{2}f'''(x) + \cdots && \text{(c)} \end{aligned}$$

Eq. (b) $-2\times$ Eq. (a):

$$f(x+2h) - 2f(x+h) = -f(x) + h^2 f''(x) + h^3 f'''(x) + \cdots \qquad \text{(d)}$$

Eq. (c) $-3\times$ Eq. (a):

$$f(x+3h) - 3f(x+h) = -2f(x) + 3h^2 f''(x) + 4h^3 f'''(x) + \cdots \qquad \text{(e)}$$

Eq. (e) $-3\times$ Eq. (d):

$$f(x+3h) - 3(x+2h) + 3f(x+h) = f(x) + h^3 f'''(x) + \cdots$$

$$f'''(x) \approx \frac{-f(x) + 3f(x+h) - 3f(x+2h) + f(x+3h)}{h^3} \quad \blacktriangleleft$$

Problem 5

Taylor series:

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \frac{h^4}{24}f^{(4)}(x) + \cdots \\ f(x-h) &= f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(x) + \frac{h^4}{24}f^{(4)}(x) - \cdots \end{aligned}$$

Adding the expressions yields

$$f(x+h) + f(x-h) = 2f(x) + h^2 f''(x) + \frac{h^4}{12}f^{(4)}(x) + \cdots \tag{a}$$

Taylor series:

$$\begin{aligned} f(x+2h) &= f(x) + 2hf'(x) + 2h^2 f''(x) + \frac{4h^3}{3}f'''(x) + \frac{2h^4}{3}f^{(4)}(x) + \cdots \\ f(x-2h) &= f(x) - 2hf'(x) + 2h^2 f''(x) - \frac{4h^3}{3}f'''(x) + \frac{2h^4}{3}f^{(4)}(x) - \cdots \end{aligned}$$

Adding the expressions gives us

$$f(x+2h) + f(x-2h) = 2f(x) + 4h^2 f''(x) + \frac{4h^4}{3}f^{(4)}(x) + \cdots \tag{b}$$

$4\times$ Eq. (a) $-$ Eq. (b):

$$-f(x+2h) + 4f(x+h) + 4f(x-h) - f(x-2h) = 6f(x) - h^4 f^{(4)}(x) + \cdots$$

$$f^{(4)}(x) \approx \frac{f(x+2h) - 4f(x+h) + 6f(x) - 4f(x-h) + f(x-2h)}{h^4} \quad \blacktriangleleft$$

Problem 6

x	2.36	2.37	2.38	2.39
$f(x)$	0.85866	0.86289	0.86710	0.87129

Use forward differences of $\mathcal{O}(h^2)$:

$$\begin{aligned} f'(x) &\approx \frac{-f(x+2h) + 4f(x+h) - 3f(x)}{2h} \\ f'(2.36) &\approx \frac{-0.86710 + 4(0.86289) - 3(0.85866)}{0.02} = 0.424 \quad \blacktriangleleft \end{aligned}$$

$$\begin{aligned} f''(x) &\approx \frac{-f(x+3h) + 4f(x+2h) - 5f(x+h) + 2f(x)}{h^2} \\ f''(2.36) &\approx \frac{-0.87129 + 4(0.86710) - 5(0.86289) + 2(0.85866)}{0.01^2} \\ &= -0.200 \quad \blacktriangleleft \end{aligned}$$

Problem 7

x	0.97	1.00	1.05
$y = f(x)$	0.85040	0.84147	0.82612

As the spacing of data points is uneven, use a polynomial interpolant:

$$P_2(x) = a_0 + a_1 x + a_2 x^2$$

where the coefficients are given by the equations

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum y_i x_i \\ \sum y_i x_i^2 \end{bmatrix}$$

$$\begin{bmatrix} 3.0000 & 3.0200 & 3.0434 \\ 3.0200 & 3.0434 & 3.0703 \\ 3.0434 & 0.3070 & 3.1008 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 2.5180 \\ 2.5338 \\ 2.5524 \end{bmatrix}$$

The solution is

$$a_0 = 1.0260 \qquad a_1 = -0.0678 \qquad a_2 = -0.1167$$

$$\begin{aligned} f'(1) &\approx P_2'(1) = a_1 + 2a_2 = -0.0678 - 2(0.1167) = -0.301 \blacktriangleleft \\ f''(1) &\approx P_2''(1) = 2a_2 = 2(-0.1167) = -0.233 \blacktriangleleft \end{aligned}$$

Probem 8

x	0.84	0.92	1.00	1.08	1.16
$f(x)$	0.431 711	0.398 519	0.367 879	0.339 596	0.313 486

Central difference approximations for $f''(x)$ of $\mathcal{O}(h^2)$ are

$$\begin{aligned} g(h) &= \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \\ g(2h) &= \frac{f(x+2h) - 2f(x) + f(x-2h)}{(2h)^2} \end{aligned}$$

At $x = 1.0$ we get

$$\begin{aligned} g(0.8) &= \frac{0.339\,596 - 2(0.367\,879) + 0.398\,519}{0.8^2} = 3.682\,81 \times 10^{-3} \\ g(1.6) &= \frac{0.313\,486 - 2(0.367\,879) + 0.431\,711}{1.6^2} = 3.687\,11 \times 10^{-3} \end{aligned}$$

Richardson's extrapolation gives us an approximation of $\mathcal{O}(h^4)$:

$$f''(1) \approx \frac{4g(0.8) - g(1.6)}{4-1} = \frac{4(3.682\,81) - 3.687\,11}{3} \times 10^{-3} = 3.6814 \times 10^{-3} \blacktriangleleft$$

Problem 9

x	0	0.1	0.2	0.3	0.4
$y = f(x)$	0.000 000	0.078 348	0.138 910	0.192 916	0.244 981

Central difference approximations for $f'(x)$ of $\mathcal{O}(h^2)$ are

$$\begin{aligned} g(h) &= \frac{f(x+h) - f(x-h)}{2h} \\ g(2h) &= \frac{f(x+2h) - f(x-2h)}{4h} \end{aligned}$$

At $x = 0.2$:

$$\begin{aligned} g(0.1) &= \frac{0.192\,916 - 0.078\,348}{0.2} = 0.57\,284 \\ g(0.2) &= \frac{0.244\,981 - 0}{0.4} = 0.612\,45\,3 \end{aligned}$$

We obtain an approximation of $\mathcal{O}(h^3)$ by Richardson's extrapolation:

$$f'(0.2) \approx \frac{4g(0.1) - g(0.2)}{4-1} = \frac{4(0.57\,284) - 0.612\,45\,3}{3} = 0.\,559\,64 \quad \blacktriangleleft$$

Problem 10

The true result is

$$f'(0.8) = \cos(0.8) = 0.696\,707$$

(a)

First forward difference approximation:

$$f'(0.8) \approx \frac{\sin(0.8+h) - \sin(0.8)}{h} = \frac{\sin(0.8+h) - 0.71736}{h}$$

h	$\sin(0.8+h)$	$f'(0.8)$
0.001	0.71805	0.69
0.0025	0.71910	0.696 ◄
0.005	0.72083	0.694

Note that two fignificant figures were lost in the computations.

(b)

First central difference approximation:

$$f'(0.8) \approx \frac{\sin(0.8+h) - \sin(0.8-h)}{2h}$$

h	$\sin(0.8+h)$	$\sin(0.8-h)$	$f'(0.8)$
0.01	0.72429	0.71035	0.697
0.025	0.73455	0.69972	0.6966 ◀
0.05	0.75128	0.68164	0.6964

Here one significant figure was lost in the computation.

Problem 11

x	-2.2	-0.3	0.8	1.9
$f(x)$	15.180	10.962	1.920	-2.040

Since there are four data points, the interpolant intersecting all these points is a cubic:

$$f(x) \approx a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

so that

$$f'(0) \approx a_1 \qquad f''(0) \approx 2a_2 \quad ◀$$

The following program computes the coefficients of the polynomial with the function `polyFit` descibed in Art. 3.4.

```
## problem5_1_11
from polyFit import *
from numpy import array

x = array([-2.2, -0.3, 0.8, 1.9])
y = array([15.18, 10.962, 1.92, -2.04])
a = polyFit(x,y,3)
print("1st derivative at x = 0: ",a[1])
print("2nd derivative at x = 0: ",a[2]*2.0)
input("Press return to exit")

1st derivative at x = 0:  -8.56
2nd derivative at x = 0:  -0.6
```

Problem 12

$$x = R\left(\cos\theta + \sqrt{2.5^2 - \sin^2\theta}\right)$$

Letting $\omega = d\theta/dt$ (constant), we have

$$\dot{x} = \frac{dx}{dt} = \frac{dx}{d\theta}\omega \qquad \ddot{x} = \frac{d^2x}{dt^2} = \frac{d^2x}{d\theta^2}\omega^2$$

Using central differences of $O(h^2)$, the finite difference approximation for the acceleration is

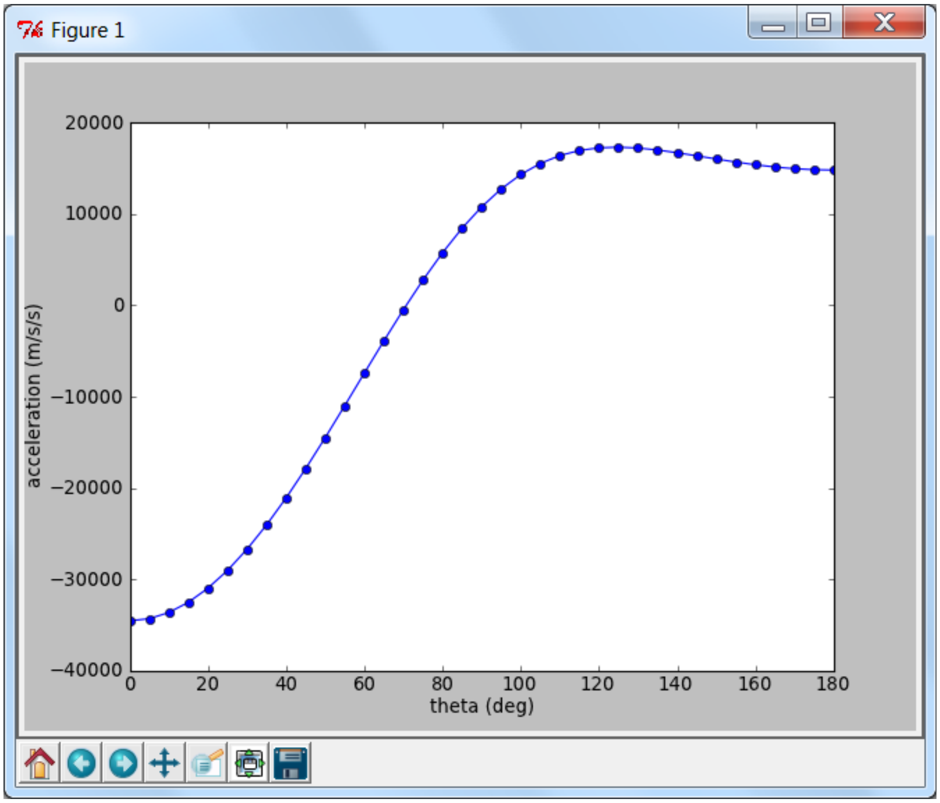
$$\ddot{x} \approx \frac{x(\theta+h) = 2x(\theta) + f(\theta-h)}{h^2}\omega^2 = \frac{f(\theta+h) - 2f(\theta) + f(\theta-h)}{h^2}R\omega^2$$

where

$$f(\theta) = \cos\theta + \sqrt{2.5^2 - \sin^2\theta}$$

We chose $h = 0.1^\circ$ in the computations.

```
## problem5_1_12
from numpy import sin,cos,sqrt,arange
from math import pi
import matplotlib.pyplot as plt

def f(theta):
    return cos(theta) + sqrt(6.25 - sin(theta)**2)

h = 0.1*pi/180.0              # units: rad
R = 0.09                      # units: m
omega = 5000.0*(2*pi)/60.0    # units: rad/s
const = R*omega**2/h**2

# Following equations are vectorized
angle = arange(0.0,185.0,5.0) # theta in degrees
theta = angle*pi/180.0        # theta in radians
accel = const*(f(theta-h) - 2.0*f(theta) + f(theta+h))

plt.plot(angle,accel,'-o')
plt.xlabel('theta (deg)')
plt.ylabel('acceleration (m/s/s)')
plt.show()

input("Press return to exit")
```



Problem 13

t (s)	9	10	11
α	$54.80°$	$54.06°$	$53.34°$
β	$65.59°$	$64.59°$	$63.62°$

$$x = a\frac{\tan\beta}{\tan\beta - \tan\alpha} \qquad y = a\frac{\tan\alpha\tan\beta}{\tan\beta - \tan\alpha}$$

Using central differences of $O(h^2)$, we have for the velocities

$$v_x = \dot{x} \approx \frac{x(11\text{ s}) - x(9\text{ s})}{2} \qquad v_y = \dot{y} \approx \frac{y(11\text{ s}) - y(9\text{ s})}{2}$$

The speed and the climb angle are

$$v = \sqrt{v_x^2 + v_y^2} \qquad \gamma = \tan^{-1}(v_y/v_x)$$

```
## problem5_1_13
from math import tan,atan,pi,sqrt
from numpy import array
```

```
def f(alpha,beta):
    ta = tan(alpha); tb = tan(beta)
    x = tb/(tb - ta); y = x*ta
    return x,y

a =500.0
alpha = array([54.80, 53.34])*pi/180.0
beta  = array([65.59, 63.62])*pi/180.0
x0,y0 = f(alpha[0],beta[0])
x1,y1 = f(alpha[1],beta[1])
vx = a*(x1 - x0)/2.0
vy = a*(y1 - y0)/2.0
print("Speed =",sqrt(vx**2 + vy**2),"m/s")
print("Gamma =",atan(vy/vx)*180.0/pi,"deg")
input("Press return to exit")

Speed = 50.099441629653164 m/s
Gamma = 15.137987979364299 deg
```

Problem 14

θ (deg)	0	30	60	90	120	150
β (deg)	59.96	56.42	44.10	25.72	-0.27	-34.29

$$\dot{\beta} = \frac{d\beta}{d\theta}\frac{d\theta}{dt} = \frac{d\beta}{d\theta}(1 \text{ rad/s})$$

We compute $d\beta/d\theta$ using Eq. (5.10):

$$\begin{aligned} f'_{i,i+1}(x) &= \frac{k_i}{6}\left[\frac{3(x-x_{i+1})^2}{x_i - x_{i+1}} - (x_i - x_{i+1})\right] \\ &\quad -\frac{k_{i+1}}{6}\left[\frac{3(x-x_i)^2}{x_i - x_{i+1}} - (x_i - x_{i+1})\right] + \frac{y_i - y_{i+1}}{x_i - x_{i+1}} \end{aligned} \tag{a}$$

Evaluating at $x = x_i$ yields

$$\begin{aligned} f'_{i,i+1}(x_i) &= \frac{k_i}{6}\left[\frac{3(x_i-x_{i+1})^2}{x_i - x_{i+1}} - (x_i - x_{i+1})\right] \\ &\quad -\frac{k_{i+1}}{6}\left[\frac{3(x_i-x_i)^2}{x_i - x_{i+1}} - (x_i - x_{i+1})\right] + \frac{y_i - y_{i+1}}{x_i - x_{i+1}} \end{aligned}$$

Letting $h = x_{i+1} - x_i$, the expression reduces to

$$f'_{i,i+1}(x_i) = -\left(\frac{k_i}{3} + \frac{k_{i+1}}{6}\right)h - \frac{y_i - y_{i+1}}{h}$$

At the last knot we need to evaluate Eq. (a) at $x = x_{i+1}$:

$$\begin{aligned} f'_{i,i+1}(x_{i+1}) &= \frac{k_i}{6}\left[\frac{3(x_{i+1}-x_{i+1})^2}{x_i - x_{i+1}} - (x_i - x_{i+1})\right] \\ &\quad -\frac{k_{i+1}}{6}\left[\frac{3(x_{i+1}-x_i)^2}{x_i - x_{i+1}} - (x_i - x_{i+1})\right] + \frac{y_i - y_{i+1}}{x_i - x_{i+1}} \\ &= \left(\frac{k_i}{6} + \frac{k_{i+1}}{3}\right)h - \frac{y_i - y_{i+1}}{h} \end{aligned}$$

There is no need to convert angles into radians, since $d\beta/d\theta$ is dimensionless.

```
## problem5_1_14
from cubicSpline import curvatures
from numpy import array

x = array([0.0, 30.0, 60.0, 90.0, 120.0, 150.0])
y = array([59.96, 56.42, 44.10, 25.72, -0.27, -34.29])
h = 30.0
k = curvatures(x,y)
n = len(x)-1
print("Theta (deg)  BetaDot (rad/s)")
for i in range(n):
    betaDot = -(k[i]/3.0 + k[i+1]/6.0)*h - (y[i] - y[i+1])/h
    print("{:7.1f} {:15.4f}".format(x[i],betaDot))
betaDot = (k[n-1]/6.0 + k[n]/3.0)*h - (y[n-1] - y[n])/h
print("{:7.1f} {:15.4f}".format(x[n],betaDot))
input("Press return to exit")
```

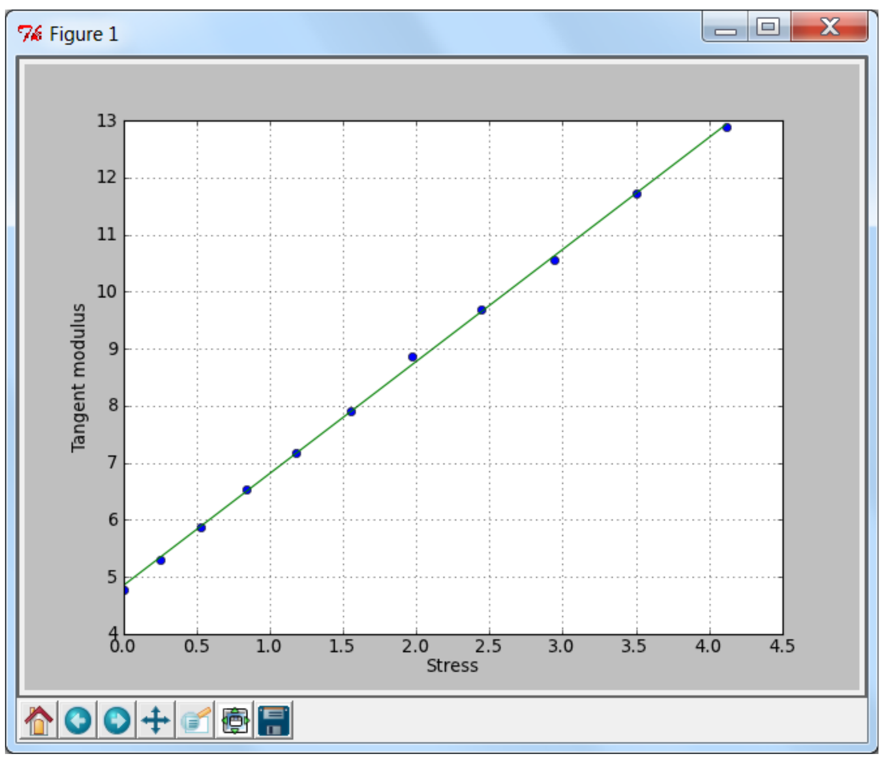
```
Theta (deg)  BetaDot (rad/s)
    0.0         -0.0505
   30.0         -0.2530
   60.0         -0.5235
   90.0         -0.7229
  120.0         -1.0220
  150.0         -1.1900
```

Problem 15

```
## problem5_1_15
from numpy import array,zeros
from polyFit import *
from plotPoly import *
```

```
s = array([0.000, 0.252, 0.531, 0.840, 1.184, 1.558,   \
                1.975, 2.444, 2.943, 3.500, 4.115])
h = 0.05    # Increment of strain
n = len(s)

# Finite differences of O(h^2)
tan_mod = zeros(n)
tan_mod[0] =(-3.0*s[0] + 4.0*s[1] - s[2])/(2.0*h)
tan_mod[n-1] = (3.0*s[n-1] - 4.0*s[n-2] + s[n-3])/(2.0*h)
for i in range(1,n-1):
    tan_mod[i] = (-s[i-1] + s[i+1])/(2.0*h)

c = polyFit(s,tan_mod,1)
print('[a,b] =',c)
plotPoly(s,tan_mod,c,'Stress','Tangent modulus')
input('Press return to exit')

[a,b] = [ 4.84329478  1.9648308 ]
```

The following plot verifies the results:



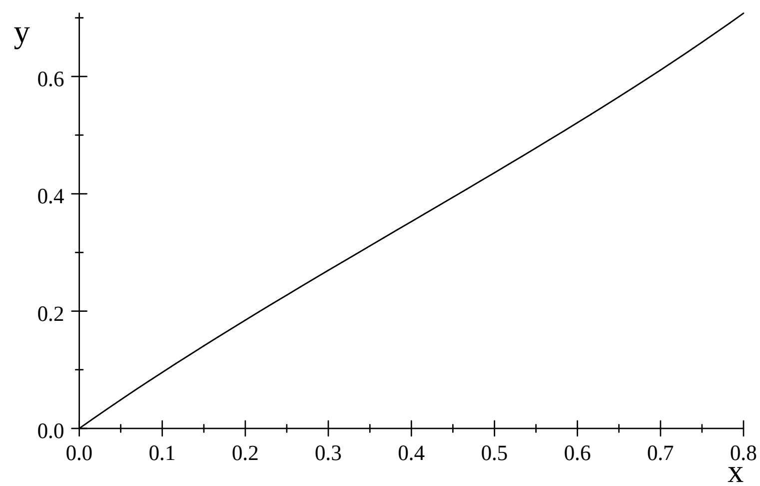
PROBLEM SET 6.1

Problem 1

$$I = \int_0^{\pi/4} \ln(1 + \tan x) dx$$

$$\begin{aligned}
I_1 &= \{\ln[1 + \tan(0)] + \ln[1 + \tan(\pi/4)]\} \frac{\pi}{8} = 0.272\,19\,8 \\
I_2 &= \frac{1}{2}(0.272\,19\,8) + \frac{\pi}{8} \ln[1 + \tan(\pi/8)] = 0.272\,19\,8 \\
I_3 &= \frac{1}{2}(0.272\,19\,8) + \frac{\pi}{16} \{\ln[1 + \tan(\pi/16)] + \ln[1 + \tan(3\pi/16)]\} \\
&= 0.272\,19\,8
\end{aligned}$$

It seems that a single panel (I_1) yields 6-figure accuracy. This fortuitous cicumstance can be explained by the plot the function, which is practically a straight line in the region of integration.



Problem 2

i	1	2	3	4	5	6	7
v_i (m/s)	1.0	1.8	2.4	3.5	4.4	5.1	6.0
P_i (kW)	4.7	12.2	19.0	31.8	40.1	43.8	43.2
$(v/P)_i$ (kN^{-1})	0.2128	0.1475	0.1263	0.1101	0.1097	0.1164	0.1389

$$\Delta t = m\int_{1s}^{6s}(v/P)\,dv$$

Uneven spacing of data points on the v-axis precludes the use of Simpson's rule or Romberg integration. The best we can do is to apply the trapezoidal rule to each panel and sum the results:

$$I = \int_1^6 (v/P)\,dv \approx \frac{1}{2}\sum_{i=1}^{6}\left[(v/P)_i + (v/P)_{i+1}\right](v_{i+1}-v_i)$$

i	$(v/P)_i + (v/P)_{i+1}$	$(v_{i+1}-v_i)$	$\left[(v/P)_i + (v/P)_{i+1}\right](v_{i+1}-v_i)$
1	0.3603	0.8	0.2882
2	0.2738	0.6	0.1643
3	0.2364	1.1	0.2600
4	0.2198	0.9	0.1978
5	0.2261	0.7	0.1583
6	0.2553	0.9	0.2298
$\sum$			1.2984

$$\begin{aligned} I &= \frac{1.2984}{2} = 0.6492\ \text{m/(kN}\cdot\text{s)} = 0.6492\times10^{-3}\ \text{m/(N}\cdot\text{s)} \\ \Delta t &= mI = 2000(0.6492\times10^{-3}) = 1.2984\ \text{s} \ \blacktriangleleft \end{aligned}$$

Problem 3

$$I = \int_{-1}^{1} f(x)\,dx \qquad f(x) = \cos(2\cos^{-1}x)$$

Two panels ($h = 1$):

x	-1	0	1
$f(x)$	1.0	-1.0	1.0

$$I = [2\,(1.0) + 4(-1.0)]\,\frac{1}{3} = -0.6667 \ \blacktriangleleft$$

Four panels ($h = 1/2$):

x	-1	$-1/2$	0	$1/2$	1
$f(x)$	1.0	-0.5	-1.0	-0.5	1.0

$$I = [2\,(1.0) + 8(-0.5) + 2(-1.0)]\,\frac{1}{6} = -0.6667 \ \blacktriangleleft$$

Six panels ($h = 1/3$):

x	-1	$-2/3$	$-1/3$	0	$1/3$	$2/3$	1
$f(x)$	1.0	-0.1111	-0.7778	-1.0	-0.7778	-0.1111	1.0

$$I=[2(1.0)+8(-0.1111)+4\,(-0.7778)+4(-1.0)]\,\frac{1}{9}=-0.6667 \blacktriangleleft$$

The function $f(x)$ appears to be a quadratic, which can be integrated exactly with Simpson's rule. Indeed, it can be shown that $\cos(2\cos^{-1}x)=-1+2x^2$.

Problem 4

$$I=\int_1^\infty (1+x^4)^{-1}dx$$

$$\begin{aligned} x^3 &= \frac{1}{t} \qquad 3x^2dx=-\frac{dt}{t^2} \\ dx &= -\frac{dt}{3x^2t^2}=-\frac{dt}{3(1/t)^{2/3}t^2}=-\frac{dt}{3t^{4/3}} \end{aligned}$$

$$I=\int_1^0\left(1+\frac{1}{t^{4/3}}\right)^{-1}\left(-\frac{1}{3t^{4/3}}\right)dt=\int_0^1\frac{dt}{3(t^{4/3}+1)}$$

t	0	0.2	0.4	0.6	0.8	1.0
$\left[3(t^{4/3}+1)\right]^{-1}$	0.3333	0.2984	0.2575	0.2214	0.1913	0.1667

$$\begin{aligned} I &\approx [0.3333+2(0.2984+0.2575+0.2214+0.1913)+0.1667]\,0.1 \\ &= 0.2437 \blacktriangleleft \end{aligned}$$

Problem 5

x (m)	0.00	0.05	0.10	0.15	0.20	0.25
F (N)	0	37	71	104	134	161
x (m)	0.30	0.35	0.40	0.45	0.50	
F (N)	185	207	225	239	250	

$$I=\frac{1}{2}mv^2=\int_0^{0.5\text{ m}}F\,dx$$

Using Simpson's rule:

$$\begin{aligned} I &\approx \left[\begin{array}{c} 0+4(37+104+161+207+239) \\ +2(71+134+185+225)+250 \end{array}\right]\frac{0.05}{3} \\ &= 74.53\text{ N}\cdot\text{m} \end{aligned}$$

$$v=\sqrt{\frac{2I}{m}}=\sqrt{\frac{2(74.53)}{0.075}}=44.58\text{ m/s} \blacktriangleleft$$

Problem 6

$$f(x) = x^5 + 3x^3 - 2 \qquad I = \int_0^2 f(x)\,dx$$

Recursive trapezoidal rule:

$$\begin{aligned}
R_{1,1} &= [f(0) + f(2)]\,\frac{H}{2} = (-2 + 54)\frac{2}{2} = 52 \\
R_{2,1} &= \frac{1}{2}R_{1,1} + \frac{H}{2}f(1) = \frac{52}{2} + \frac{2}{2}(2) = 28 \\
R_{3,1} &= \frac{1}{2}R_{2,1} + \frac{H}{4}\,[f(0.5) + f(1.5)] \\
&= \frac{1}{2}(28) + \frac{2}{4}(-1.593\,75 + 15.718\,75) = 21.0625
\end{aligned}$$

Romberg extrapolation:

$$\mathbf{R} = \begin{bmatrix} 52 & & \\ 28 & 20 & \\ 21.0625 & 18.75 & 18.6667 \end{bmatrix}$$

Because the error in $R_{3.3}$ is $\mathcal{O}(h^6)$, the result is exact for a polynomial of degree 5. Therefore,

$$I = 18.6667 \blacktriangleleft$$

is the "exact" integral.

Problem 7

x	0	$\pi/4$	$\pi/2$	$3\pi/4$	π
$f(x)$	1.0000	0.3431	0.2500	0.3431	1.0000

Romberg integration:

$$\begin{aligned}
R_{1,1} &= [f(0) + f(\pi)]\,\frac{H}{2} = (1 + 1)\frac{\pi}{2} = 3.1416 \\
R_{2,1} &= \frac{1}{2}R_{1,1} + \frac{H}{2}f(\pi/2) = \frac{\pi}{2} + \frac{\pi}{2}(0.25) = 1.9635 \\
R_{3,1} &= \frac{1}{2}R_{2,1} + \frac{H}{4}\,[f(\pi/4) + f(3\pi/4)] \\
&= \frac{1}{2}(1.9635) + \frac{\pi}{4}(0.3431 + 0.3431) = 1.5207
\end{aligned}$$

$$\mathbf{R} = \begin{bmatrix} 3.1416 & & \\ 1.9635 & 1.5708 & \\ 1.5207 & 1.3732 & 1.3600 \end{bmatrix}$$

$$I = \int_0^{\pi} f(x)\,dx \approx 1.3600 \blacktriangleleft$$

This result has an error $\mathcal{O}(h^6)$. Note that trapezoidal rule would result in $I = 1.5207$ with an error $\mathcal{O}(h^2)$, and Simpson's rule would yield $I = 1.3732$ with an error $\mathcal{O}(h^4)$.

Problem 8

$$I = \int_0^1 \frac{\sin x}{\sqrt{x}} dx$$

$$x = t^2 \qquad dx = 2t\,dt$$

$$I = \int_0^1 \frac{\sin(t^2)}{t} 2t\,dt = \int_0^1 2\sin(t^2)\,dt = \int_0^1 f(t)\,dt$$

Romberg integration:

$$\begin{aligned}
R_{1,1} &= [f(0) + f(1)]\frac{H}{2} = (0 + 1.6829)\frac{1}{2} = 0.8415 \\
R_{2,1} &= \frac{1}{2}R_{1,1} + \frac{H}{2}f(0.5) = \frac{0.8415}{2} + \frac{1}{2}(0.4948) = 0.6682 \\
R_{3,1} &= \frac{1}{2}R_{2,1} + \frac{H}{4}[f(0.25) + f(0.75)] \\
&= \frac{0.6682}{2} + \frac{1}{4}(0.1249 + 1.0667) = 0.6320 \\
R_{4,1} &= \frac{1}{2}R_{3,1} + \frac{H}{8}[f(0.125) + f(0.375) + f(0.625) + f(0.875)] \\
&= \frac{0.6320}{2} + \frac{1}{8}(0.0312 + 0.2803 + 0.7615 + 1.3860) = 0.6234
\end{aligned}$$

$$\mathbf{R} = \begin{bmatrix} 0.8415 & & & \\ 0.6682 & 0.6104 & & \\ 0.6320 & 0.6199 & 0.6205 & \\ 0.6234 & 0.6205 & 0.6205 & 0.6205 \blacktriangleleft \end{bmatrix}$$

Problem 9

According to Eq. (3.10) the cubic spline interpolant is

$$\begin{aligned}
f_{i,i+1}(x) &= \frac{k_i}{6}\left[\frac{(x-x_{i+1})^3}{x_i-x_{i+1}} - (x-x_{i+1})(x_i-x_{i+1})\right] \\
&\quad -\frac{k_{i+1}}{6}\left[\frac{(x-x_i)^3}{x_i-x_{i+1}} - (x-x_i)(x_i-x_{i+1})\right] \\
&\quad +\frac{y_i(x-x_{i+1}) - y_{i+1}(x-x_i)}{x_i-x_{i+1}}
\end{aligned}$$

Integrating over the panel yields

$$\begin{aligned}
\int_{x_i}^{x_{i+1}} f_{i,i+1}(x)\,dx &= -\frac{k_i}{24}\,(x_i-x_{i+1})^3 + \frac{k_i}{12}(x_i-x_{i+1})^3 \\
&\quad -\frac{k_{i+1}}{24}\,(x_i-x_{i+1})^3 + \frac{k_{i+1}}{12}(x_i-x_{i+1})^3 \\
&\quad -\frac{1}{2}(x_i-x_{i+1})(y_i+y_{i+1})
\end{aligned}$$

Substituting $x_i - x_{i+1} = -h$, this becomes

$$\int_{x_i}^{x_{i+1}} f_{i,i+1}(x)\,dx = -\frac{h^3}{24}\,(k_i+k_{i+1}) + \frac{h}{2}(y_i+y_{i+1})$$

Therefore,

$$\begin{aligned}
I &= \int_{x_0}^{x_n} y(x)\,dx = \sum_{i=0}^{n-1}\left[\int_{x_i}^{x_{i+1}} f_{i,i+1}(x)\,dx\right] \\
&= -\frac{h^3}{24}(k_0+2k_1+2k_2+\cdots+2k_{n-1}+k_n) \\
&\quad +\frac{h}{2}(y_0+2y_1+2y_2+\cdots+2y_{n-1}+y_n) \text{ Q.E.D.}
\end{aligned}$$

Problem 10

$$\begin{aligned}
\sin x &= t^2 \qquad \cos x\,dx = 2t\,dt \\
\sqrt{1-\sin^2 x}dx &= 2tdt \qquad dx = \frac{2t}{\sqrt{1-t^4}}dt
\end{aligned}$$

$$\int_0^{\pi/4} \frac{dx}{\sqrt{\sin x}} = \int_0^{2^{-1/4}} \frac{2t}{\sqrt{1-t^4}}dt$$

```
### problem6_1_10
from romberg import *
from math import sqrt

def f(x): return 2.0*x/sqrt(1.0-x**4)

a = 0.0; b = 1.0/sqrt(sqrt(2.0))
I,nPanels = romberg(f,a,b)
print("Intergral =",I)
print("Number of panels =",nPanels)
input("Press return to exit")
```

```
Intergral = 0.78539816425
Number of panels = 64
```

Problem 11

$$h(\theta_0) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - \sin^2(\theta_0/2)\sin^2\theta}}$$

```
## problem6_1_11
from romberg import *
from math import sqrt,sin,pi
from numpy import array

def f(x): return 1.0/sqrt(1.0 - (sin(ampl/2.0)*sin(x))**2)

theta0 = array([15.0, 30.0, 45.0])*pi/180.0
for i in range(3):
    ampl = theta0[i]
    I,nPanels = romberg(f,0.0,pi/2.0)
    print("Amplitude =",ampl*180.0/pi,"deg")
    print("h =",I,"\n")
input("Press return to exit")
```

```
Amplitude = 15.0 deg
h = 1.57755165307

Amplitude = 30.0 deg
h = 1.59814200257

Amplitude = 45.0 deg
h = 1.63358630909
```

Problem 12

$$w(r) = w_0 \int_0^{\pi/2} \frac{\cos^2\theta}{\sqrt{(r/a)^2 - \sin^2\theta}} d\theta$$

When $r = 2a$, we have

$$\frac{w}{w_0} = \int_0^{\pi/2} \frac{\cos^2\theta}{\sqrt{4 - \sin^2\theta}} d\theta$$

```
## problem6_1_12
from romberg import *
from math import sqrt,sin,cos,pi

def f(x): return cos(x)**2/sqrt(4.0 - sin(x)**2)

I,nPanels = romberg(f,0.0,pi/2.0)
print("w/wo =",I)
input("Press return to exit")

w/wo = 0.4062988864
```

Problem 13

$$f(x) = \mu g + \frac{k}{m}(\mu b + x)\left(1 - \frac{b}{\sqrt{b^2 + x^2}}\right)$$

$$\begin{aligned} \mu g &= 0.3(9.81) = 2.943 \text{ m/s}^2 \\ \frac{k}{m} &= \frac{80}{0.8} = 100 \text{ s}^{-2} \\ \mu b &= 0.3(0.4) = 0.12 \text{ m} \end{aligned}$$

$$\begin{aligned} f(x) &= 2.943 + 100(0.12 + x)\left(1 - \frac{0.4}{\sqrt{0.16 + x^2}}\right) \\ I &= \int_0^{0.4} f(x)\,dx \qquad v_0 = \sqrt{2I} \end{aligned}$$

```
## problem6_1_13
from romberg import *
from math import sqrt
```

```
def f(x):
    return 2.943 + 100.0*(0.12 + x)  \
           *(1.0 - 0.4/sqrt(0.16 + x**2))

I,nPanels = romberg(f,0.0,0.4)
print("v0 =",sqrt(2.0*I),"m/s")
input("Press return to exit")

v0 = 2.49767483249 m/s
```

Problem 14

$$g(u) = u^3 \int_0^{1/u} \frac{x^4 e^x}{(e^x - 1)^2} dx$$

Note that

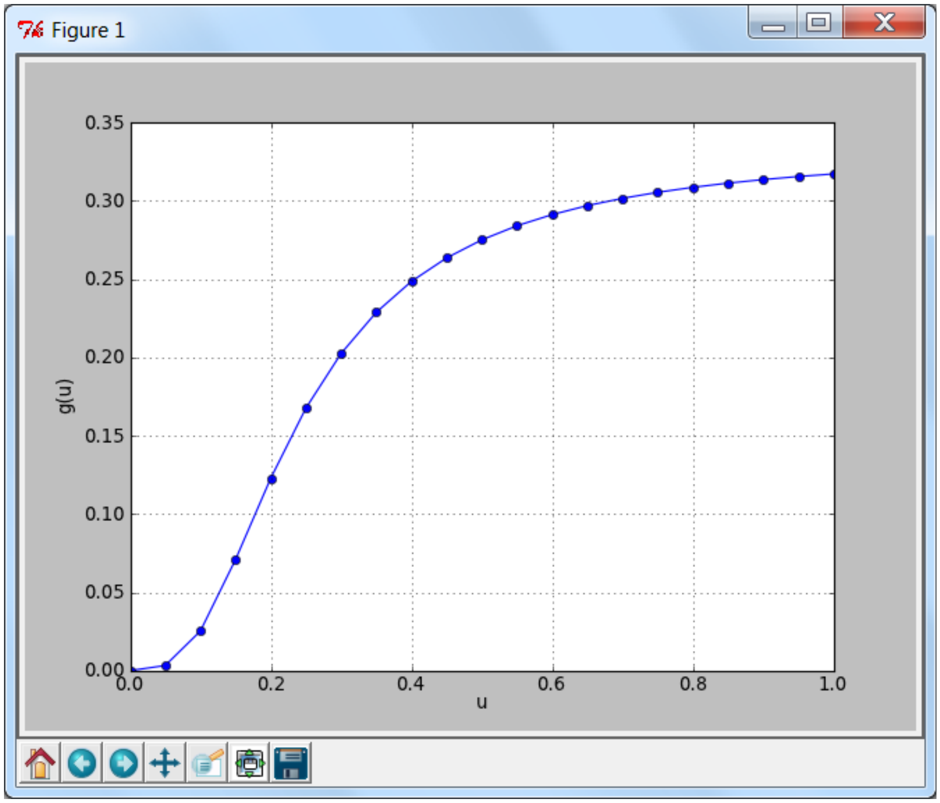
$$\int_0^{\infty} \frac{x^4 e^x}{(e^x - 1)^2} dx$$

is finite, so that $g(0) = 0$. Also

$$\frac{x^4 e^x}{(e^x - 1)^2} \to 0 \text{ as } x \to 0$$

```
## problem6_1_14
from romberg import *
from numpy import arange,zeros
from math import exp
import matplotlib.pyplot as plt

def f(x):
    if x == 0: return 0.0
    else: return (x**4)*exp(x)/(exp(x) - 1.0)**2

u = arange(0.0,1.01,0.05)
g = zeros(len(u))
for i in range(len(u)):
    if i != 0:
        I,nPanels = romberg(f,0.0,1.0/u[i])
        g[i]= (u[i]**3)*I

plt.plot(u,g,'-o')
plt.xlabel('u');plt.ylabel('g(u)')
plt.grid(True)
```

```
plt.show()
input("Press return to exit")
```



Problem 15

$$i(t) = i_0 e^{-t/t_0} \sin(2t/t_0) \qquad E = \int_0^\infty R\,[i(t)]^2\,dt$$

$$\begin{aligned} i_0 &= 100\ \text{A} \qquad R = 0.5\ \Omega \qquad t_0 = 0.01\ \text{s} \\ Ri_0^2 &= 0.5(100)^2 = 5000 \end{aligned}$$

Since we cannot deal with infinite integration limits, we must change the upper limit from ∞ to τ, where τ is a time during which the current *just* reaches negligible magnitude. If τ is too large, Romberg integration will not work—it converges prematurely to $E = 0$. In the program below we tried $\tau = 0.1$ s:

```
## problem6_1_15
from romberg import *
from math import exp,sin
```

```
def f(t):
    R = 0.5
    i0 = 100.0
    t0 = 0.01
    return R*(i0*exp(-t/t0)*sin(2.0*t/t0))**2

energy,numPanels = romberg(f,0.0,0.1)
print("Energy =", energy,"W.s")
input("Press return to exit")
```

```
Energy = 9.99999996113 W.s
```

Problem 16

The following program uses Romberg integration:

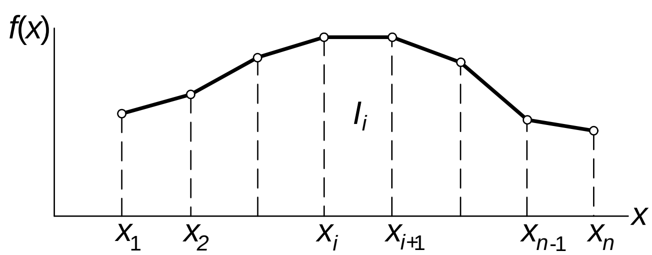
```
## problem6_1_16
from math import sin,pi,sqrt
from romberg import *

def f(t):
    return (sin(pi*t/0.05) - 0.2*sin(2.0*pi*t/0.05))**2

Integral,nPanels = romberg(f,0.0,0.05)
print('iRMS in amps =',sqrt(Integral/0.05))
input("Press return to exit")
```

The result is:

```
iRMS in amps = 0.7211102300036695
```

Problem 17

(a)



The trapezoidal rule for a single panel is

$$I_i = \frac{1}{2}(f_i + f_{i+1})(x_{i+1} - x_i)$$

so that the composite trapezoidal rule becomes

$$\begin{aligned}
I &= \sum_{i=1}^{n-1} I_i \\
&= \frac{1}{2}[(f_1 + f_2)(x_2 - x_1) + (f_2 + f_3)(x_3 - x_2) + (f_3 + f_4)(x_4 - x_3) \\
&\quad + \cdots + (f_{n-1} + f_n)(x_n - x_{n-1})] \\
&= \frac{1}{2}[f_1(x_2 - x_1) + f_2(x_3 - x_1) + f_3(x_4 - x_2) + \cdots + f_i(x_{i+1} - x_{i-1}) \\
&\quad + \cdots + f_{n-1}(x_n - x_{n-2}) + f_n(x_n - x_{n-1})] \quad \blacktriangleleft
\end{aligned}$$

(b)

```
## problem6_1_17
from numpy import array

stress = array([586,662,765,841,814,689,600])*10.0e6
strain = array([1,25,45,68,89,122,150])*0.001
n = len(stress)

## Trapezoidal rule:
modulus = stress[0]*(strain[1] - strain[0])
for i in range(1,n-1):
    modulus = modulus + stress[i]*(strain[i+1] - strain[i-1])
modulus = modulus + stress[n-1]*(strain[n-1] - strain[n-2])
print('Modulus in Pa =',modulus/2.0)
input("Press return to exit")
```

```
Modulus in Pa = 1079380000.0
```

Thus the modulus of toughness is 108 MPa ◄

Problem 18

We chose Romberg integration. Note that $\lim_{t\to 0} t^{-1}\sin t = 1$.

```
## problem6_1_18
from romberg import *
```

```python
from math import sin

def Si(x):

    def  f(t):
        if t == 0.0: return 1.0
        else: return sin(t)/t

    I,nPanels = romberg(f,0.0,x)
    return I

# Test the program by computing Si(1.0):
x = 1.0
print('Si(',x,') =',Si(x))
input('Press return to exit')
```

```
Si( 1.0 ) = 0.946083070387
```

PROBLEM SET 6.2

Problem 1

$$f(x) = \frac{\ln x}{x^2 - 2x + 2} \qquad I = \int_1^{\pi} f(x)dx$$

$$x_i = \frac{b+a}{2} + \frac{b-a}{2}\xi_i \qquad I \approx \frac{b-a}{2}\sum_{i=0}^{n} A_i f(x_i)$$

(a) 2-node quadrature:

$$\begin{aligned}
x_0 &= \frac{\pi+1}{2} + \frac{\pi-1}{2}(-0.577\,350) = 1.452\,572 \\
x_1 &= \frac{\pi+1}{2} + \frac{\pi-1}{2}(0.577\,350) = 2.689\,021 \\
A_0 &= A_1 = 1
\end{aligned}$$

$$I \approx \frac{\pi-1}{2}(0.256\,743 + 0.309\,868) = 0.6067 \quad \blacktriangleleft$$

(b) 4-node quadrature:

$$\begin{aligned}
x_0 &= \frac{\pi+1}{2} + \frac{\pi-1}{2}(-0.861\,136) = 1.148\,69\,5 \\
x_1 &= \frac{\pi+1}{2} + \frac{\pi-1}{2}(-0.339\,981) = 1.706\,74\,6 \\
x_2 &= \frac{\pi+1}{2} + \frac{\pi-1}{2}(0.339\,981) = 2.434\,84\,7 \\
x_3 &= \frac{\pi+1}{2} + \frac{\pi-1}{2}(0.861\,136) = 2.992\,89\,8
\end{aligned}$$

i	x_i	$f(x_i)$	A_i	$A_i f(x_{i($
0	1.148 69 5	0.135 628	0.347 855	0.047 179
1	1.706 74 6	0.356 514	0.652 145	0.232 499
2	2.434 84 7	0.290 927	0.652 145	0.189 727
3	2.992 89 8	0.220 499	0.347 855	0.076 702
$\sum$				0.546 107

$$I \approx \frac{\pi-1}{2}(0.546\,107) = 0.5848 \quad \blacktriangleleft$$

Problem 2

$$f(x) = (1 - x^2)^3 \qquad I = \int_0^\infty e^{-x} f(x)\, dx \approx \sum_{i=0}^{n} A_i f(x_i)$$

Since $f(x)$ is a polynomial of degree 6, we use 4-node quadrature ($n = 3$) for an exact result:

i	x_i	$f(x_i)$	A_i	$A_i f(x_i)$
0	0.322 548	0.719 234	0.603 154	0.434
1	1.745 761	−8.586 927	0.357 418	−3.069
2	4.536 620	$-7.507\,569 \times 10^3$	$0.388\,791 \times 10^{-1}$	−291.954
3	9.395 071	$-6.645\,926 \times 10^5$	$0.539\,295 \times 10^{-3}$	−358.411
$\sum$				−653.000

$$I = -653.0 \quad ◀$$

Problem 3

$$I = \int_0^{\pi/2} \frac{dx}{\sqrt{\sin x}}$$

$$\begin{aligned} \sin x &= t^2 \qquad \cos x\, dx = 2t\, dt \qquad \sqrt{1 - t^4} dx = 2t\, dt \\ dx &= \frac{2t}{\sqrt{(1 - t^2)(1 + t^2)}} dt \end{aligned}$$

$$\begin{aligned} I &= 2\int_0^1 \frac{dt}{\sqrt{(1 - t^2)(1 + t^2)}} = \int_{-1}^1 \frac{dt}{\sqrt{(1 - t^2)(1 + t^2)}} \\ I &= \int_{-1}^1 \frac{g(t)}{\sqrt{1 - t^2}} dt \approx \frac{\pi}{n+1} \sum_{i=0}^{n} g(t_i) \qquad g(t) = \frac{1}{\sqrt{1 + t^2}} \qquad n = 5 \end{aligned}$$

i	$t_i = \cos \dfrac{(2i+1)\pi}{2n+2}$	$g(t_i)$
0	0.965 926	0.719 255
1	0.707 107	0.816 497
2	0.258 819	0.968 100
3	−0.258 819	0.968 100
4	−0.707 107	0.816 497
5	−0.965 926	0.719 255
$\sum$		5.007 703

$$I \approx \frac{\pi}{6}(5.007\,703) = 2.622\,03 \quad ◀$$

Problem 4

$$I = \int_0^{\pi} f(x)\,dx \qquad f(x) = \sin x$$

The truncation error is

$$E = \frac{(b-a)^{2n+3}\left[(n+1)!\right]^4}{(2n+3)\left[(2n+2)!\right]^3} f^{(2n+2)}(c), \quad a < c < b$$

where

$$a = 0 \qquad b = \pi \qquad n = 3$$
$$f^{(2n+2)}(x) = f^{(8)}(x) = \sin x$$

Thus

$$\begin{aligned} E_{\min} &= \frac{(\pi-0)^9(4!)^4}{9(8!)^3}\sin 0 = 0 \ \blacktriangleleft \\ E_{\max} &= \frac{(\pi-0)^9(4!)^4}{9(8!)^3}\sin\frac{\pi}{2} = 1.6764\times 10^{-5} \ \blacktriangleleft \end{aligned}$$

Problem 5

$$I = \int_0^{\infty} e^{-x} f(x)\,dx \qquad f(x) = \sin x$$

The truncation error is

$$E = \frac{\left[(n+1)!\right]^2}{(2n+2)!} f^{(2n+2)}(c), \quad 0 < c < \infty$$

Noting that

$$f_{\min}^{(2n+2)} = -1 \qquad f_{\max}^{(2n+2)} = 1$$

we have

$$E_{\min,\max} = \pm\frac{\left[(n+1)!\right]^2}{(2n+2)!}$$

n	$\lvert E_{\min,\max}\rvert$
6	2.914×10^{-4}
7	7.7704×10^{-5}
8	2.057×10^{-5}
9	5.413×10^{-6}
10	1.418×10^{-6}
11	3.698×10^{-7}

To be sure of 6 decimal place accuracy, one should use $n = 11$; that is 12 nodes $\blacktriangleleft$

Problem 6

$$I = \int_0^1 \frac{2x+1}{\sqrt{x(1-x)}} dx$$

$$x = \frac{1}{2}(1+t) \qquad dx = \frac{1}{2}dt$$

$$\begin{aligned} I &= \int_{-1}^{1} \frac{2+t}{\sqrt{(1-t^2)}} dt \\ I &= \int_{-1}^{1} \frac{f(t)}{\sqrt{(1-t^2)}} dt \qquad f(t) = 2+t \end{aligned}$$

Since $f(t)$ is linear in t, Gauss-Chebyshev quadrature will give the exact integral with a single node. Substituting $n = 0$ into the quadrature formulas

$$I = \frac{\pi}{n+1} \sum_{i=0}^{n} f(t_i) \qquad t_i = \cos \frac{(2i+1)\pi}{2n+2}$$

we get

$$I = \pi f\left(\cos \frac{\pi}{2}\right) = \pi\,(2+0) = 2\pi \quad \blacktriangleleft$$

Problem 7

$$I = \int_0^{\pi} \sin x \ln x \, dx$$

Let

$$x = \pi z \qquad dx = \pi \, dz$$

$$\begin{aligned} I &= \pi \int_0^1 \sin(\pi z) \ln(\pi z)\, dz \\ &= \pi \ln \pi \int_0^1 \sin(\pi z)\, dz + \pi \int_0^1 \sin(\pi z) \ln z \, dz \end{aligned}$$

The first term is

$$I_1 = \pi \ln \pi \int_0^1 \sin(\pi z)\, dz = \pi \ln \pi \left(\frac{2}{\pi}\right) = 2 \ln \pi = 2.289\,46$$

The second term

$$I_2 = \pi \int_0^1 f(z) \ln z \, dz \qquad f(z) = \sin(\pi z)$$

can be evaluated with Gauss quadrature with logarithmic singularity. Using $n = 3$ we get

i	z_i	$f(z_i)$	A_i	$A_i f(z_i)$
0	0.041 449	0.129 848	0.383 464	0.049 792
1	0.245 275	0.696 533	0.386 875	0.269 471
2	0.556 165	0.984 473	0.190 435	0.187 478
3	0.848 982	0.456 838	0.039 226	0.017 920
$\sum$				0.524 661

$$I_2 = -\pi(0.524661) = -1.648\,27$$

$$I = I_1 + I_1 = 2.289\,46 - 1.648\,27 = 0.6412 \quad \blacktriangleleft$$

The true value of the integral is 0.641 182.

Problem 8

$$I = \int_0^\pi f(x)\,dx \qquad f(x) = x \sin x$$

$$E = \frac{(b-a)^{2n+3}\left[(n+1)!\right]^4}{(2n+3)\left[(2n+2)!\right]^3} f^{(2n+2)}(c), \quad a < c < b$$

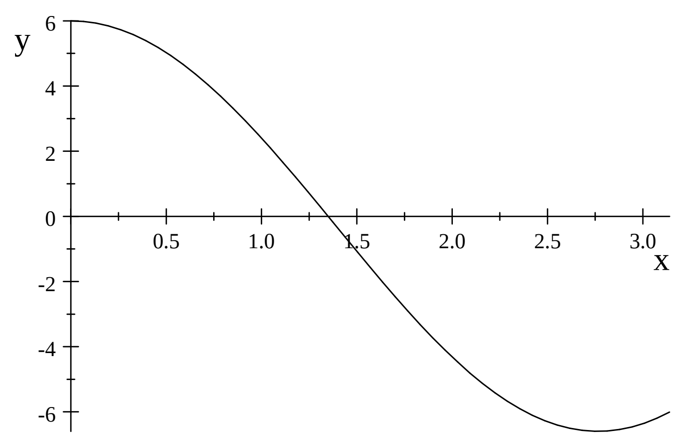
With $n = 2$ (3 nodes) the error becomes

$$E = \frac{(\pi - 0)^7 (3!)^4}{7(6!)^3} f^{(6)}(c) = 1.498 \times 10^{-3} f^{(6)}(c)$$

where

$$f^{(6)}(x) = \frac{d^6}{dx^6}(x \sin x) = 6\cos x - x \sin x$$

is plotted below.



The plot shows that $f^{(6)}_{\max} = 6$ and $f^{(6)}_{\min}$ occurs at $x \approx 2.75$, so that

$$f^{(6)}_{\min} \approx 6\cos 2.75 - 2.75\sin 2.75 = -6.60$$

Therefore,

$$\begin{aligned} E_{\min} &= 1.498 \times 10^{-3}(-6.60) = -9.89 \times 10^{-3} \quad \blacktriangleleft \\ E_{\max} &= 1.498 \times 10^{-3}(6) = 8.99 \times 10^{-3} \quad \blacktriangleleft \end{aligned}$$

To find the actual error, we must evaluate the integral with Gauss-Legendre quadrature:

$$\begin{aligned} x_0 &= \frac{\pi}{2}(1 - 0.774\,597) = 0.354\,06\,2 \\ x_1 &= \frac{\pi}{2}(1 - 0) = 1.570\,79\,6 \\ x_2 &= \frac{\pi}{2}(1 + 0.774\,597) = 2.787\,530 \end{aligned}$$

i	x_i	$f(x_i)$	A_i	$A_i f(x_i)$
0	0.354 06 2	0.068 199	0.555 556	0.068 199
1	1.570 79 6	1.570 79 6	0.888 889	1.396 264
2	2.787 530	0.536 927	0.555 556	0.536 927
$\sum$				2.001 390

$$I \approx \frac{\pi}{2}(2.001\,390) = 3.143\,77\,6$$

The exact integral is $I = \pi = 3.141\,593$, so that the actual error is

$$E = 3.143\,77\,6 - 3.141\,593 = 2.18 \times 10^{-3} \quad \blacktriangleleft$$

Problem 9

$$I = \int_0^2 f(x)\,dx \qquad f(x) = \frac{\sinh x}{x}$$

Try Gauss-Legendre quadrature with $n = 2$ (3 nodes):

$$\begin{aligned} x_0 &= 1 - 0.774\,597 = 0.225\,40\,3 \\ x_1 &= 1 \\ x_2 &= 1 + 0.774\,597 = 1.774\,597 \end{aligned}$$

i	x_i	$f(x_i)$	A_i	$A_i f(x_i)$
0	0.225 40 3	1.008 489	0.555 556	0.560 272
1	1	1.175 201	0.888 889	1.044 623
2	1.774 597	1.613 987	0.555 556	0.896 660
$\sum$				2.501 555

$$I \approx 2.502 \quad \blacktriangleleft$$

The true value of the integral is $I = 2.501\,567$.

Problem 10

$$I = \int_0^\infty \frac{x\,dx}{e^x + 1}$$

$$e^x = \frac{1}{t} \qquad e^x dx = -\frac{1}{t^2}dt \qquad dx = -\frac{1}{t}dt \qquad x = -\ln t$$

$$I = \int_1^0 \frac{-\ln t}{(1/t + 1)(-t)}dt = -\int_0^1 \frac{\ln t}{1+t}dt$$

Use Gauss 4-node ($n = 3$) quadrature with logarithmic singularity.

$$I = \int_0^1 f(t) \ln t\,dt \qquad f(x) = \frac{1}{1+t}$$

i	t_i	$f(t_i)$	A_i	$A_i f(x_i)$
0	0.041 449	−0.960 201	0.383 464	−0.368 203
1	0.254 275	−0.797 273	0.386 875	−0.310674
2	0.556 165	−0.642 60 5	0.190 435	−0.122 375
3	0.848 982	−0.540 838	0.039 226	−0.021214
$\sum$				−0.822 466

$$I \approx 0.822\,466 \;\blacktriangleleft$$

The true value of the integral is $I = 0.822\,467$. The discrepancy is due to unavoidable roundoff errors.

Problem 11

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \qquad y = \frac{b}{a}\sqrt{a^2 - x^2}$$

$$\frac{2x}{a^2}dx + \frac{2y}{b^2}dy = 0 \qquad \frac{dy}{dx} = -\frac{b^2}{a^2}\frac{x}{y} = -\frac{b}{a}\frac{x}{\sqrt{a^2 - x^2}}$$

$$S = 2\int_{-a}^{a} \sqrt{1 + (dy/dx)^2}dx = 2\int_{-a}^{a} \sqrt{1 + \frac{b^2}{a^2}\frac{x^2}{a^2 - x^2}}dx$$

Because the integrand is singular at $x = a$, the integral in its present form is not well-suited for quadrature. But with a change of variable

$$x = a\xi \qquad dx = a\,d\xi$$

$$
\begin{aligned}
S &= 2a\int_{-1}^{1}\sqrt{1+\frac{b^2}{a^2}\frac{\xi^2}{1-\xi^2}}d\xi = 2\int_{-1}^{1}\frac{\sqrt{(1-\xi^2)a^2+b^2\xi^2}}{\sqrt{1-\xi^2}}d\xi \\
S &= 2\int_{-1}^{1}\frac{f(\xi)}{\sqrt{1-\xi^2}}d\xi \qquad f(\xi)=\sqrt{(1-\xi^2)a^2+b^2\xi^2}
\end{aligned}
$$

the integral can be evaluated with Gauss-Chebyshev quadrature.
We found by experimentation that the number of nodes required to achieve the specified accuracy increases with the eccentricity of the ellipse. Consequently, we chose $n = 5\max(a/b, b/a)$ which appears to give 5 decimal point accuracy over a wide range of eccentricities.

```
## problem6_2_11
from math import sqrt,cos,pi

def f(x):
    return sqrt((1.0 - x**2)*a**2 + (b*x)**2)

def S(a,b,n):
    S = 0.0
    for i in range(n+1):
        x = cos((i + 0.5)*pi/(n + 1)); S = S + f(x)
    return 2.0*S*pi/(n + 1)

a = eval(input("Length of first semiaxis  ==> "))
b = eval(input("Length of second semiaxis ==> "))
n = int(5*max(a/b,b/a))
print("S =",S(a,b,n))
input("Press return to exit")
```

```
Length of first semiaxis  ==> 2.0
Length of second semiaxis ==> 1.0
S = 9.68844903025
```

The true value of circumference is $S = 9.688\,448$

Problem 12

$$
\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt
$$

Here the required number of nodes is dependent on the value of x (larger x requires more nodes). In the following program we overcome this problem by brute force: we apply Gauss-Legendre quadrature with $n = 3, 4, \ldots$ until successive results are in agreement within 10^{-6}.

```
## problem 6_2_12
from math import pi,sqrt,exp
from gaussQuad import *

def f(t): return exp(-t**2)

def erf(x):
    if x > 5.0: return 0,1.0
    Iold = gaussQuad(f,0.0,x,3)
    for n in range(4,10):
        Inew = gaussQuad(f,0.0,x,n)
        if abs(Inew-Iold) < 1.0e-6: break
        Iold = Inew
    return n,2.0/sqrt(pi)*Inew

x = eval(input("x ==> "))
n,I = erf(x)
print("erf(x) =",I)
print("Number of nodes =",n + 1)
input("\nPress return to exit")
```

```
x ==> 1.0
erf(x) = 0.842700786116
Number of nodes = 6
```

Problem 13

$$C = \int_0^1 \left(\left(\sqrt{2}-1\right)^2 - \left(\sqrt{1+z^2}-1\right)^2 \right)^{-1/2} dz$$

This can be written in the form

$$\begin{aligned} C &= \frac{1}{2}\int_{-1}^{1} \frac{f(z)}{\sqrt{1-z^2}} dz \\ f(z) &= \sqrt{\frac{1-z^2}{\left(\sqrt{2}-1\right)^2 - \left(\sqrt{1+z^2}-1\right)^2}} \end{aligned}$$

where $f(z)$ is free of singularities. The integral can be now evaluated with Gauss-Chebyshev quadrature ($n = 10$ is required for 6 decimal place accuracy).

```
## problem6_2_13
from numpy import array
from math import sqrt,cos,pi
```

```
def f(z):
    b = (sqrt(2.0) - 1.0)**2
    return sqrt((1.0 - z**2) / (b -(sqrt(1.0 + z**2) - 1.0)**2))

n = 10
C = 0.0
for i in range(n+1):
    z = cos((i + 0.5)*pi/(n + 1))
    C = C + f(z)
C = 0.5*pi/(n + 1)*C
print("C =",C)
input("Press return to exit")

C = 3.266702315927328
```

Problem 14

$$C\left(\frac{h}{b}\right) = \int_0^1 z^2\sqrt{1+\left(\frac{2h}{b}z\right)^2}dz$$

We use Gauss-Legendre quadrature with $n = 5$ (6 nodes), which was found to be sufficient for 4 decimal point accuracy.

```
## problem 6_2_14
from math import sqrt
from gaussQuad import *

def f(z): return (z**2)*sqrt(1.0 + (2.0*r*z)**2)

while 1:
    try: r = eval(input("\nh/b ==> "))
    except SyntaxError: break
    print("C =",gaussQuad(f,0.0,1.0,5))
input("\nPress return to exit")

h/b ==> 0.5
C = 0.420158376422

h/b ==> 1.0
C = 0.606337803619

h/b ==> 2.0
```

```
C = 1.05889534599
```

Problem 15

$$I = \int_0^{\pi/2} \ln(\sin x)dx = I_1 + I_2 + I_3$$

$$\begin{aligned} I_1 &= \int_0^{0.01} \ln(\sin x)dx \approx \int_0^{0.01} \ln x\,dx = [x \ln x - x]_0^{0.01} \\ &= 0.01(\ln 0.01 - 1) \end{aligned}$$

$$I_2 = \int_{0.01}^{0.2} \ln(\sin x)dx \qquad I_3 = \int_{0.2}^{\pi/2} \ln(\sin x)dx$$

To guarantee 6-decimal point accuracy, we compute both I_2 and I_3 with $n = 3, 4, \ldots$ until successive results are in agreement within 10^{-6}.

```
## problem6_2_15
from math import sin,log,pi
from gaussQuad import *

def f(x): return log(sin(x))

def I(a,b):
    Iold = gaussQuad(f,a,b,3)
    for n in range(4,30):
        Inew = gaussQuad(f,a,b,n)
        if abs(Inew - Iold) < 1.0e-6: break
        Iold = Inew
    return Inew

integral = I(0.01,0.2) + I(0.2,pi/2)  \
           + 0.01*log(0.01) - 0.01
print("Integral =",integral)
input("\nPress return to exit")

Integral = -1.08879242746
```

Problem 16

h (m)	0	15	35	52	80	112
p (Pa)	310	425	530	575	612	620

```
## problem6_2_16
from polyFit import polyFit
from numpy import array
from gaussQuad import *

def p(h):   # Interpolant for p(h)
    return c[0] + c[1]*h + c[2]*h**2 + c[3]*h**3

def px(h):  # Interpolant for h*p(h)
    return h*p(h)

hData = array([0, 15, 35, 52, 80, 112])*1.0
pData = array([310, 425, 530, 575, 612, 620])*1.0
c = polyFit(hData,pData,3)
resultant = gaussQuad(p,0.0,112.0,2) # 2 nodes reqd.
moment = gaussQuad(px,0.0,112.0,3)   # 3 nodes reqd.
print("h of pressure center =", moment/resultant,"m")
input("Press return to exit")

h of pressure center = 60.5730320569 m
```

Problem 17

Since the spline in each segment is cubic, integration order of 2 is sufficient in the Gauss-Legendre quadrature (recall that quadrature with 2 integration points is exact for a cubic).

```
## problem6_2_17
from cubicSpline import *
from numpy import array
from math import sqrt

def integral(xData,yData):
    m = len(xData)
    k = curvatures(xData,yData)
    integral = 0.0
    for i in range(m-1):                    # Loop over segments
        c1 = (xData[i+1] + xData[i])/2.0
        c2 = (xData[i+1] - xData[i])/2.0
        x1 = c1 - c2/sqrt(3.0)              # x-coord. of node 1
        x2 = c1 + c2/sqrt(3.0)              # x-coord. of node 2
        y1 = evalSpline(xData,yData,k,x1)  # Interpolant at node 1
```

```
        y2 = evalSpline(xData,yData,k,x2)  # Interpolant at node 2
        integral = integral + c2*(y1 + y2) # Eq. (6.29)
    return integral

stress = array([586,662,765,841,814,689,600])*10.0e6
strain = array([1,25,45,68,89,122,150])*0.001
print('Modulus in Pa =', integral(strain,stress))
input("Press return to exit")
```

```
Modulus in Pa = 1081678339.52
```

This is approximately the same value as calculated in Problem 17, Problem Set 6.1.

PROBLEM SET 6.3

Problem 1

$$I = \int_{-1}^{1}\int_{-1}^{1}(1-x^2)(1-y^2)\,dx\,dy$$

As the integral is biquadratic, second-order quadrature is exact. The region of integration is a "standard" rectangle. All 4 integration points contribute the same amount to the integral:

$$I = 4(1 - 0.577350^2)^2 = 1.777\,8 \quad \blacktriangleleft$$

Problem 2

$$I = \int_{y=0}^{2}\int_{x=0}^{3} f(x,y)\,dx\,dy \qquad f(x,y) = x^2y^2$$

Since the integrand is biquadratic, second-order quadrature is exact. The coordinates of the integration points are

$$\begin{aligned}
x_{0,1} &= \frac{3+0}{2} \pm \frac{3-0}{2}(0.577350) = \begin{cases} 2.366\,025 \\ 0.633\,975 \end{cases} \\
y_{0,1} &= \frac{2+0}{2} \pm \frac{2-0}{2}(0.577350) = \begin{cases} 1.577350 \\ 0.422\,650 \end{cases}
\end{aligned}$$

The area scale factor (constant in this case) is

$$|\mathbf{J}| = \frac{\text{area of rectangle}}{\text{area of “std”. rectangle}} = \frac{3\times 2}{2\times 2} = 1.5$$

$$\begin{aligned}
I &= \sum_{i=0}^{1}\sum_{j=0}^{1} A_iA_j f(x_i,y_j)\,|\mathbf{J}| \\
&= 1.5\left[\begin{array}{c} (2.366\,025)^2\,(1.577350)^2 + (2.366\,025)^2(0.422\,650)^2 \\ +(0.633\,975)^2(1.577350)^2 + (0.633\,975)^2(0.422\,650)^2 \end{array}\right] \\
&= 24.0000 \quad \blacktriangleleft
\end{aligned}$$

Problem 3

$$\int_{-1}^{1}\int_{-1}^{1} f(x,y)\,dx\,dy \qquad f(x,y) = e^{-(x^2+y^2)}$$

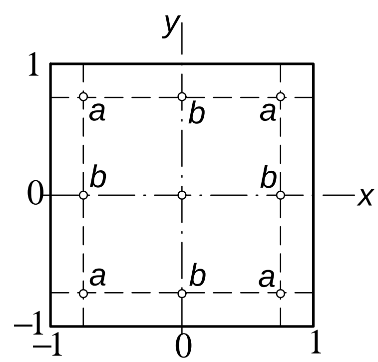
The integration is over a "standard" rectangle.

(a)

All four integation points contribute the same amount. Therefore,

$$I = \sum_{i=0}^{1}\sum_{j=0}^{1} A_i A_j f(x_i, y_j) = 4\exp\left[-2(0.577350)^2\right] = 2.0537$$

(b)



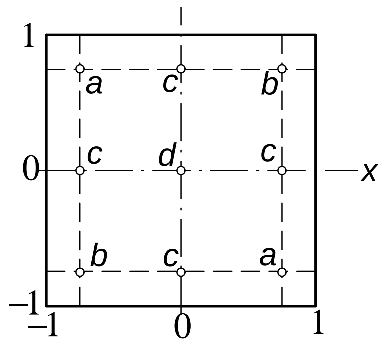
Values of $f(x,y)$ at the integration points are

$$\begin{aligned} f_a &= \exp\left[-2(0.774\,597)^2\right] = 0.301\,194 \\ f_b &= \exp\left[-(0.774\,597)^2\right] = 0.548\,811 \\ f_{\text{center}} &= 1 \end{aligned}$$

$$\begin{aligned} I &\approx 4(0.555\,556)^2(0.301\,194) \\ &\quad +4(0.555\,556)(0.888\,889)(0.548\,811) + (0.888\,889)^2 \\ &= 2.2460 \ \blacktriangleleft \end{aligned}$$

Problem 4

$$\int_{-1}^{1}\int_{-1}^{1} f(x,y)dx\,dy \qquad f(x,y) = \cos\frac{\pi(x-y)}{2}$$



Values of $f(x,y)$ at the integration points are

$$\begin{aligned}
f_a &= \cos\frac{2\pi(-0.774\,597)}{2} = -0.759\,583 \\
f_b &= \cos(0) = 1 \\
f_c &= \cos\frac{\pi(0.774\,597)}{2} = 0.346\,711 \\
f_d &= \cos(0) = 1
\end{aligned}$$

$$\begin{aligned}
I &\approx 2(0.555\,556)^2(-0.759\,583) + 2(0.555\,556)^2(1) \\
&\quad +4(0.555\,556)(0.888\,889)(0.346\,711) + (0.888\,889)^2(1) \\
&= 1.6234 \blacktriangleleft
\end{aligned}$$

Problem 5

$$I = \int\int_A xy\,dx\,dy$$

$$\mathbf{x} = \begin{bmatrix} 0 & 2 & 4 & 0 \end{bmatrix}^T \qquad \mathbf{y} = \begin{bmatrix} 0 & 0 & 4 & 4 \end{bmatrix}^T$$

$$\begin{aligned}
x(\xi,\eta) &= \sum_{k=1}^{4} N_k(\xi,\eta)x_k \\
&= \frac{1}{4}(1+\xi)(1-\eta)(2) + \frac{1}{4}(1+\xi)(1+\eta)(4) \\
&= \frac{1}{2}(1+\xi)(3+\eta)
\end{aligned}$$

$$
\begin{aligned}
y(\xi,\eta) &= \sum_{k=1}^{4} N_k(\xi,\eta) y_k \\
&= \frac{1}{4}(1+\xi)(1+\eta)(4) + \frac{1}{4}(1-\xi)(1+\eta)(4) \\
&= 2(1+\eta)
\end{aligned}
$$

$$
\mathbf{J}(\xi,\eta) = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} (3+\eta)/2 & 0 \\ (1+\xi)/2 & 2 \end{bmatrix} \qquad |\mathbf{J}(\xi,\eta)| = 3+\eta
$$

$$
\begin{aligned}
I &= \int_{-1}^{1}\int_{-1}^{1} x(\xi,\eta) y(\xi,\eta)\, |\mathbf{J}(\xi,\eta)|\; d\eta\, d\xi \\
&= \int_{-1}^{1}\int_{-1}^{1} \left[\frac{1}{2}(1+\xi)(3+\eta)\right] [2(1+\eta)]\,(3+\eta)\, d\eta\, d\xi \\
&= \int_{-1}^{1}\int_{-1}^{1} (9 + 15\eta + 7\eta^2 + \eta^3 + 9\xi + 15\xi\eta + 7\xi\eta^2 + \xi\eta^3)\, d\eta\, d\xi \\
&= 4\int_{0}^{1}\int_{0}^{1} (9 + 7\eta^2)\, d\eta\, d\xi = 4\left(9 + \frac{7}{3}\right) = \frac{136}{3} \quad \blacktriangleleft
\end{aligned}
$$

Problem 6

$$
I = \int\int_A x\, dx\, dy
$$

$$
\mathbf{x} = \begin{bmatrix} -1 & 1 & 4 & 0 \end{bmatrix}^T \qquad \mathbf{y} = \begin{bmatrix} 0 & 0 & 3 & 3 \end{bmatrix}^T
$$

$$
\begin{aligned}
x(\xi,\eta) &= \sum_{k=1}^{4} N_k(\xi,\eta) x_k \\
&= \frac{1}{4}(1-\xi)(1-\eta)(-1) + \frac{1}{4}(1+\xi)(1-\eta)(1) + \frac{1}{4}(1+\xi)(1+\eta)(4) \\
&= \frac{1}{2}(2 + 3\xi + 2\eta + \xi\eta) \\
y(\xi,\eta) &= \sum_{k=1}^{4} N_k(\xi,\eta) y_k \\
&= \frac{1}{4}(1+\xi)(1+\eta)(3) + \frac{1}{4}(1-\xi)(1+\eta)(3) = \frac{3}{2}(1+\eta)
\end{aligned}
$$

$$
\mathbf{J}(\xi,\eta) = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} (3+\eta)/2 & 0 \\ (2+\xi)/2 & 3/2 \end{bmatrix} \qquad |\mathbf{J}(\xi,\eta)| = \frac{3}{4}(3+\eta)
$$

$$
\begin{aligned}
I &= \int_{-1}^{1}\int_{-1}^{1} x(\xi,\eta)\,|\mathbf{J}(\xi,\eta)|\;d\eta\,d\xi \\
&= \int_{-1}^{1}\int_{-1}^{1}\left[\frac{1}{2}(2+3\xi+2\eta+\xi\eta)\right]\left[\frac{3}{4}(3+\eta)\right]\;dx\,dy \\
&= \int_{-1}^{1}\int_{-1}^{1}\left(\frac{9}{4}+3\eta+\frac{3}{4}\eta^2+\frac{27}{8}\xi+\frac{9}{4}\xi\eta+\frac{3}{8}\xi\eta^2\right)dx\,dy \\
&= 4\int_{0}^{1}\int_{0}^{1}\left(\frac{9}{4}+\frac{3}{4}\eta^2\right)dy\,dx = 4\left(\frac{9}{4}+\frac{1}{4}\right) = 10 \;\blacktriangleleft
\end{aligned}
$$

Problem 7

$$\int\int_A x^2 dx\,dy$$

The quadratic triangle formula (3 integration points) is exact for this integral. Referring to Fig. 6.10 in the text, the coordinates of the integration points are

$$x_a = 0 \qquad x_b = x_c = 1.5$$

$$I = A\sum_{k=a}^{c} W_k f(x_k, y_k) = 9\left(\frac{1}{3}\right)(0^2+1.5^2+1.5^2) = 13.5 \;\blacktriangleleft$$

Problem 8

$$\int\int_A x^3 dx\,dy$$

We must use the cubic triangle formula for exact result. The corner x-coordinates are

$$\mathbf{x} = \begin{bmatrix} 0 & 3 & 0 \end{bmatrix}^T$$

and the x-coordinates of the integration points become

$$
\begin{aligned}
x_a &= \frac{1}{3}(0+3+0) = 1 \\
x_b &= \frac{1}{5}(0+3)+\frac{3}{5}(0) = 0.6 \\
x_c &= \frac{3}{5}(0)+\frac{1}{5}(3+0) = 0.6 \\
x_d &= \frac{1}{5}(0+0)+\frac{3}{5}(3) = 1.8
\end{aligned}
$$

$$
\begin{aligned}
I &= A\sum_{k=a}^{d} W_k f(x_k, y_k) \\
&= 9\left[-\frac{27}{48}(1)^3 + \frac{25}{48}\left(0.6^3 + 0.6^3 + 1.8^3\right)\right] = 24.3 \blacktriangleleft
\end{aligned}
$$

Problem 9

$$
\int\int_A (3-x)y\,dx\,dy
$$

Quadratic triangle formula is exact in this case. The integration points are located at

$$
\begin{aligned}
x_a &= 0 \qquad x_b = x_c = 1.5 \\
y_a &= y_c = 2 \qquad y_b = 0
\end{aligned}
$$

$$
\begin{aligned}
I &= A\sum_{k=a}^{c} W_k f(x_k, y_k) \\
&= 6\left(\frac{1}{3}\right)[(3-0)(2) + (3-1.5)(0) + (3-1.5)(2)] = 18 \blacktriangleleft
\end{aligned}
$$

Problem 10

$$
I = \int\int_A x^2 y\,dx\,dy
$$

$$
\mathbf{x} = \begin{bmatrix} 0 & 3 & 0 \end{bmatrix}^T \qquad \mathbf{y} = \begin{bmatrix} 0 & 0 & 4 \end{bmatrix}^T
$$

The integrand is cubic, requiring 4 integration points, which are located at

$$
\begin{aligned}
x_a &= \frac{1}{3}(0+3+0) = 1 \\
x_b &= \frac{1}{5}(0+3) + \frac{3}{5}(0) = \frac{3}{5} \\
x_c &= \frac{3}{5}(0) + \frac{1}{5}(3+0) = \frac{3}{5} \\
x_d &= \frac{1}{5}(0+0) + \frac{3}{5}(3) = \frac{9}{5}
\end{aligned}
$$

$$
\begin{aligned}
y_a &= \frac{1}{3}(0+0+4) = \frac{4}{3} \\
y_b &= \frac{1}{5}(0+0) + \frac{3}{5}(4) = \frac{12}{5} \\
y_c &= \frac{3}{5}(0) + \frac{1}{5}(0+4) = \frac{4}{5} \\
y_d &= \frac{1}{5}(0+4) + \frac{3}{5}(0) = \frac{4}{5}
\end{aligned}
$$

$$
\begin{aligned}
I &= A\sum_{k=a}^{d} W_k f(x_k, y_k) \\
&= 6\left\{-\frac{27}{48}(1)^2\frac{4}{3} + \frac{25}{48}\left[\left(\frac{3}{5}\right)^2\frac{12}{5} + \left(\frac{3}{5}\right)^2\frac{4}{5} + \left(\frac{9}{5}\right)^2\frac{4}{5}\right]\right\} \\
&= 7.2 \blacktriangleleft
\end{aligned}
$$

Problem 11

$$
I = \int\int_A f(x,y)\,dx\,dy \qquad f(x,y) = xy(2-x^2)(2-xy)
$$

The integrand $f(x,y)$ is a 4th degree polynomial in x. In addition, $|\mathbf{J}(\xi,\eta)|$ is generally a quadratic, so that the integrand of $\int\int_A f(\xi,\eta)\,|\mathbf{J}(\xi,\eta)|\,d\xi\,d\eta$ is a polynomial of degree 6, requiring quadrature of order $m = 4$. The following program prompts for m:

```
#!/usr/bin/python
## example6_16b
import numpy as np
import math
from triangleQuad import *

def f(x,y):
    return (x**2 + y**2)/2.0        \
           -(x**3 - 3.0*x*y**2)/6.0 \
           -2.0/3.0

xCorner = np.array([-1.0, -1.0, 2.0])
yCorner = np.array([math.sqrt(3.0), -math.sqrt(3.0), 0.0])
print("Integral =",triangleQuad(f,xCorner,yCorner))
input("Press return to  exit")

Integration order ==> 4
Integral = 41.853968254
```

Problem 12

$$I = \int\int_A f(x,y)\,dx\,dy \qquad f(x,y) = xy\exp(-x^2)$$

As $f(x,y)$ is not a polynomial, quadrature is not exact. Of course, the accuracy increases with the order m of integration, but it is difficult to determine beforehand the relationship between m and the error. The following program prompts for m, which makes it easy to determine its proper value by experimentation:

```
## problem6_3_12
from gaussQuad2 import *
from numpy import array
from math import exp

def f(x,y):
    return x*y*exp(-x**2)

x = array([-3.0,  1.0, 3.0, -1.0])
y = array([-2.0, -2.0, 2.0,  2.0])
while True:
    try: m = eval(input("\nIntegration order ==> "))
    except SyntaxError: break
    print("Integral =", gaussQuad2(f,x,y,m))
input("Press return to exit")
```

```
Integration order ==> 6
Integral = 0.378837537786

Integration order ==> 8
Integral = 0.379595281052

Integration order ==> 10
Integral = 0.37955440857
```

It seems that $I = 0.3796$ ◀ is achievable with 8th order quadrature.

Problem 13

$$I = \int\int_A f(x,y)\,dx\,dy \qquad f(x,y) = (1-x)(y-x)y$$

The program below uses `triangleQuad` (the cubic integration formulas for a triangle). Because the integrand is a cubic, the result is exact.

```
## problem6_3_13
from numpy import array
from triangleQuad import *

def f(x,y): return (1.0 - x)*(y - x)*y

xCorner = array([0.0, 1.0, 1.0])
yCorner = array([0.0, 0.0, 1.0])
print("Integral =",triangleQuad(f,xCorner,yCorner))
input("Press return to  exit")

Integral = -0.00833333333333
```

Problem 14

$$I = \int\int_A f(x,y)\,dx\,dy \qquad f(x,y) = \sin \pi x$$

The quadrature will not be exact because $f(x, y)$ is not a polynomial. The following is essentially the program Problem 13:

```
## problem6_3_14
from numpy import array
from math import sin,pi
from triangleQuad import *

def f(x,y): return sin(pi*x)

xCorner = array([0.0, 1.0, 1.0])
yCorner = array([0.0, 0.0, 1.0])
print("Integral =",triangleQuad(f,xCorner,yCorner))
input("Press return to  exit")

Integral = 0.310239475206
```

In comparison, the true value of the integral is $I = 0.318\,310$.

Problem 15

$$I = \int\int_A f(x,y)\,dx\,dy \qquad f(x,y) = \sin\pi x \sin\pi(y-x)$$

We used the program below, which prompts for the integration order m, to evaluate the integral with increasing m until the desired 6-digit accuracy was reached).

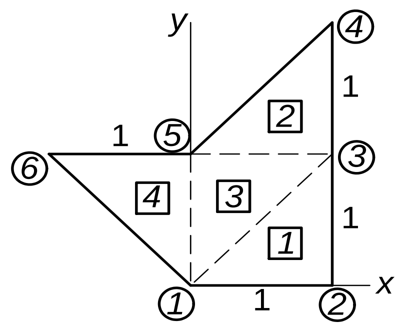
```
## problem6_3_15
from gaussQuad2 import *
from numpy import array
from math import sin,pi

def f(x,y): return sin(pi*x)*sin(pi*(y - x))

x = array([0.0, 1.0, 1.0, 1.0])
y = array([0.0, 0.0, 1.0, 1.0])
while True:
    try: m = eval(input("\nIntegration order ==> "))
    except SyntaxError: break
    print("Integral =", gaussQuad2(f,x,y,m))
input("Press return to exit")
```

```
Integration order ==> 4
Integral = -0.202592182933

Integration order ==> 5
Integral = -0.202643639957

Integration order ==> 6
Integral = -0.202642345757
```

The last result concides with the true value of the integral $I = -2/\pi^2 = -0.202\,642$ ◀

Problem 16

The figure shows the numbering of the corner points and the elements. The data used by the program (the arrays `x`, `y` and `cornerID`) are derived from this figure.




```
## problem6_3_16
from numpy import array,zeros
from triangleQuad import *

def f(x,y): return x*y*(y - x)
# Coordinates of corners
x = array([0.0, 1.0, 1.0, 1.0, 0.0, -1.0])
y = array([0.0, 0.0, 1.0, 2.0, 1.0,  1.0])
# Corner numbers of elements (counter-clockwise)
cornerID = array([[1, 2, 3],  \
                  [3, 4, 5],  \
                  [1, 3, 5],  \
                  [5, 6, 1]])
# Corner coordinates of an element
xCorner = zeros((3))
yCorner = zeros((3))
integral = 0.0
# Loop over elements
for i in range(len(cornerID)):
    # Assemble x and y coordinate arrays of the element
    for j in range(3):
        xCorner[j] = x[cornerID[i,j] - 1]
        yCorner[j] = y[cornerID[i,j] - 1]
    integral = integral + triangleQuad(f,xCorner,yCorner)
print(''Integral ='',integral)
input(''Press return to exit'')

Integral = 0.133333333333
```

PROBLEM SET 7.1

Problem 1

The integration formulas are

$$\begin{aligned} K_0 &= hF(x, y) \\ K_1 &= hF\left(x + \frac{h}{2}, y + \frac{1}{2}K_0\right) \\ y(x+h) &= y(x) + K_1 \end{aligned}$$

where

$$F(x, y) = -4y + x^2$$

Step 1 ($x = 0$ to 0.015):

$$\begin{aligned} y(0) &= 1 \qquad K_0 = 0.015\left[-4(1) + 0^2\right] = -0.06 \\ K_1 &= 0.015\left[-4\left(1 + \frac{-0.06}{2}\right) + \left(0 + \frac{0.015}{2}\right)^2\right] = -0.05820 \end{aligned}$$

$$y(0.015) = 1 + (-0.05820) = 0.9418$$

Step 2 ($x = 0.015$ to 0.03):

$$\begin{aligned} K_0 &= 0.015\left[-4\,(0.9418) + 0.15^2\right] = -0.05617 \\ K_1 &= 0.015\left[-4\left(0.9418 + \frac{-0.05617}{2}\right) + \left(0.015 + \frac{0.015}{2}\right)^2\right] = -0.05482 \end{aligned}$$

$$y(0.03) = 0.9418 + (-0.05482) = 0.8870 \blacktriangleleft$$

The exact solution given in Example 7.1 is $y(0.03) = 0.8869$. The discrepancy is within the roundoff error.

Problem 2

$$F(x, y) = -4y + x^2$$

$$
\begin{aligned}
K_0 &= hF(x,y) = 0.03\left(-4(1)+0^2\right) = -0.12 \\
K_1 &= hF\left(x+\frac{h}{2}, y+\frac{K_0}{2}\right) \\
&= 0.03\left(-4\left(1+\frac{-0.12}{2}\right)+\left(0+\frac{0.03}{2}\right)^2\right) = -0.112\ 79 \\
K_2 &= hF\left(x+\frac{h}{2}, y+\frac{K_1}{2}\right) \\
&= 0.03\left(-4\left(1+\frac{-0.112\ 79}{2}\right)+\left(0+\frac{0.03}{2}\right)^2\right) = -0.113\ 23 \\
K_3 &= hF\left(x+h, y+K_2\right) \\
&= 0.03\left(-4(1-0.113\ 23)+(0+0.03)^2\right) = -0.106\ 39 \\
y(0.1) &= y(0)+\frac{1}{6}(K_0+2K_1+2K_2+K_3) \\
&= 1+\frac{1}{6}\left(-0.12+2(-0.112\ 79)+2(-0.113\ 23)+(-0.106\ 39)\right) \\
&= 0.8869 \ \blacktriangleleft
\end{aligned}
$$

The result agrees with the analytical solution.

Problem 3

The integration formula is

$$y(x+h) = y(x) + y'(x)h = y(x) + 0.1\sin y$$

$$
\begin{aligned}
y(0) &= 1.0 \\
y(0.1) &= 1.0 + 0.1\sin 1.0 = 1.0841 \\
y(0.2) &= 1.0841 + 0.1\sin 1.0841 = 1.1725 \\
y(0.3) &= 1.1725 + 0.1\sin 1.1725 = 1.2647 \\
y(0.4) &= 1.2647 + 0.1\sin 1.2647 = 1.3601 \\
y(0.5) &= 1.3601 + 0.1\sin 1.3601 = 1.4579 \ \blacktriangleleft
\end{aligned}
$$

In Example 7.3 we used the 2nd-order Runge-Kutta method, which yielded the true result $y(0.5) = 1.4664$. Hence Euler's method is in error by

$$\frac{1.4579 - 1.4664}{1.4664} \times 100 = -0.60\%$$

Problem 4

$$y' = y^{1/3} \qquad y(0) = 0$$

One solution is clearly $y = 0$. To prove that $y = (2x/3)^{3/2}$ is also a solution, we compute

$$y' = \frac{d}{dx}\left(\frac{2x}{3}\right)^{3/2} = \frac{3}{2}\left(\frac{2x}{3}\right)^{1/2}\frac{2}{3} = \left(\frac{2x}{3}\right)^{1/2} = y^{1/3} \text{ Q.E.D.}$$

(a)

If $y(0) = 0$, the solution $y = 0$ would be produced. Let us try intergating with the 4th-order Runge-Kutta method from $x = 0$ to 1 (only the initial and final values are printed):

```
## problem7_1_4
from numarray import zeros,array
from run_kut4 import *
from printSoln import *

def F(x,y):
 F = zeros(1)
 F[0] = y[0]**(1.0/3.0)
 return F

x = 0.0          # Start of integration
xStop = 1.0      # End of integration
y = array([0.0]) # Initial values of {y}
h = 0.01         # Step size
freq = 0         # Printout frequency
X,Y = integrate(F,x,y,xStop,h)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

```
        x          y[ 0 ]
  0.0000e+000   0.0000e+000
  1.0000e+000   0.0000e+000
```

(b)

If $y(0)$ is any non-zero number, the solution $y' = y^{1/3}$ would be produced. With the intial condition $y(0) = 10^{-16}$ the above program results in

```
        x          y[ 0 ]
  0.0000e+000   1.0000e-016
  1.0000e+000   5.4025e-001
```

The analytical solution is $y(1) = (2/3)^{3/2} = 0.5443$, so the the numerical solution is not very accurate. The disceptancy is caused by singularity of y'' and higher derivatives at $x = 0$, which in results in a large truncation error in the first integration step.

Problem 5

We use the notation $y = y_0$, $y' = y_1$, $y'' = y_2$ etc.

(a)

$$\ln y' + y = \sin x \qquad y' = \exp(\sin x - y)$$
$$y_0' = \exp(\sin x - y_0) \quad \blacktriangleleft$$

(b)

$$y''y - xy' - 2y^2 = 0 \qquad y'' = \frac{xy'}{y} + 2y$$
$$\begin{bmatrix} y_0' \\ y_1' \end{bmatrix} = \begin{bmatrix} y_1 \\ xy_1/y_0 + 2y_0 \end{bmatrix} \quad \blacktriangleleft$$

(c)

$$y^{(4)} - 4y''\left(1 - y^2\right)^{1/2} = 0 \qquad y^{(4)} = 4y''\left(1 - y^2\right)^{1/2}$$
$$\begin{bmatrix} y_0' \\ y_1' \\ y_2' \\ y_3' \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ 4y_2\left(1 - y_0^2\right)^{1/2} \end{bmatrix} \quad \blacktriangleleft$$

(d)

$$(y'')^2 = \left|32y'x - y^2\right| \qquad y'' = \left|32y'x - y^2\right|^{1/2}$$
$$\begin{bmatrix} y_0' \\ y_1' \end{bmatrix} = \begin{bmatrix} y_1 \\ \left|32y_1x - y_0^2\right|^{1/2} \end{bmatrix} \quad \blacktriangleleft$$

Problem 6

We use the notation $x = y_0$, $y = y_1$, $\dot{x} = y_2$ and $\dot{y} = y_3$

(a)

$$\ddot{y} = x - 2y \qquad \ddot{x} = y - x$$

$$\begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \end{bmatrix} = \begin{bmatrix} y_2 \\ y_3 \\ y_1 - y_0 \\ y_0 - 2y_1 \end{bmatrix} \blacktriangleleft$$

(b)

$$\ddot{y} = -y(\dot{y}^2 + \dot{x}^2)^{1/4} \qquad \ddot{x} = -x(\dot{y}^2 + \dot{x}^2)^{1/4} - 32$$

$$\begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \end{bmatrix} = \begin{bmatrix} y_2 \\ y_3 \\ -y_0(y_3^2 + y_2^2)^{1/4} - 32 \\ -y_1(y_3^2 + y_2^2)^{1/4} \end{bmatrix} \blacktriangleleft$$

(c)

$$\ddot{y} = (4\dot{x} - t\sin y)^{1/2} \qquad \ddot{x} = (4\dot{y} - t\cos y)/x$$

$$\begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \end{bmatrix} = \begin{bmatrix} y_2 \\ y_3 \\ (4y_3 - t\cos y_1)/y_0 \\ (4y_2 - t\sin y_1)^{1/2} \end{bmatrix} \blacktriangleleft$$

Problem 7

$$\frac{d^2\theta}{d\tau^2} = -\sin\theta$$

With the notation $\theta = y_0$, $\dot{\theta} = y_1$ the equivalent first-order differential equations are

$$\mathbf{F} = \begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ -\sin y_0 \end{bmatrix}$$

We release the pendulum from rest at $\theta = 1$, $\tau = 0$ and determine the time it takes for it to return to the starting point for the first time. Hence the initial conditions are

$$\begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

To assure that the integration covers one period, we stop at $\tau = 2.2\pi$ (this is 10% larger than the period for small amplitudes). We use the 4th-order Runge-Kutta method with $h = 0.25$.

```
## problem7_1_7
from numpy import zeros,array
from run_kut4 import *
from printSoln import *
from math import sin,pi

def F(x,y):
    F = zeros(2)
    F[0] = y[1]; F[1] = -sin(y[0])
    return F

x = 0.0                 # Start of integration
xStop = 2.2*pi          # End of integration
y = array([1.0,0.0])    # Initial values of {y}
h = 0.25                # Step size
freq = 1                # Printout frequency

X,Y = integrate(F,x,y,xStop,h)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

The part of the printout that spans the return of the pendulum to the release position is (note the change in the sign of the velocity y_1):

```
        x           y[ 0 ]         y[ 1 ]
   6.5000e+00    9.8315e-01     1.6776e-01
   6.7500e+00    9.9892e-01    -4.1972e-02
```

The value of τ at the instant when $d\theta/d\tau = 0$ can be estimated from two-term Taylor series expansion

$$\begin{aligned}
\left.\frac{d\theta}{d\tau}\right|_{6.75+\Delta\tau} &= \left.\frac{d\theta}{d\tau}\right|_{6.75} + \left.\frac{d^2\theta}{d\tau^2}\right|_{6.75} \Delta\tau \\
0 &= -0.041\,972 + (-\sin 0.99892)\,\Delta\tau \\
\Delta\tau &= -0.04991 \\
\tau &= 6.75 - 0.04991 = 6.700 \quad \blacktriangleleft
\end{aligned}$$

Thus the period is $6.700\sqrt{L/g}$ ◀

Problem 8

$$\ddot{y} = g - \frac{c_D}{m}\dot{y}^2$$

With the notation $\theta = y_0$, $\dot{\theta} = y_1$ the equivalent first-order differential equations are

$$\mathbf{F} = \begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ g - (c_D/m)y_1^2 \end{bmatrix}$$

with the initial conditions

$$\begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Without air resistance it takes approximately 10 s for a 500 m fall (obtained from $t = \sqrt{2g/h}$). With air resistance the time should be considerably longer; we estimate 15 s. The program below uses the 4th-order Runge-Kutta method with $h = 0.5$ s.

```
## problem7_1_8
from numpy import zeros,array
from run_kut4 import *
from printSoln import *

def F(x,y):
    g = 9.80665; c = 0.2028; m = 80.0
    F = zeros(2)
    F[0] = y[1]; F[1] = g - c/m*y[1]**2
    return F

x = 0.0                 # Start of integration
xStop = 15.0            # End of integration
y = array([0.0,0.0])  # Initial values of {y}
h = 0.5                 # Step size
freq = 1                # Printout frequency

X,Y = integrate(F,x,y,xStop,h)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

Here is a portion of the output:

```
       x          y[ 0 ]        y[ 1 ]
   1.2000e+01    4.8180e+02    5.9433e+01
   1.2500e+01    5.1162e+02    5.9828e+01
```

The time t of the 500 m fall is estimated Taylor series:

$$\begin{aligned}
y(12.5+\Delta t) &= y(12.5)+y'(12.5)\,\Delta t \\
500 &= 511.62+59.828\,\Delta t \\
\Delta t &= -0.194 \\
t &= 12.5-0.194 = 12.306 \text{ s} \quad \blacktriangleleft
\end{aligned}$$

Problem 9

$$\ddot{y} = \frac{P(t)}{m} - \frac{k}{m}y \qquad y(0) = \dot{y}(0) = 0$$

$$P(t) = \begin{cases} 10t \text{ N} & \text{when } t < 2 \text{ s} \\ 20 \text{ N} & \text{when } t \geq 2 \text{ s} \end{cases}$$

The equivalent first-order differential equations are

$$\mathbf{F} = \begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ (P - ky_0)/m \end{bmatrix}$$

The maximum displacement should occur soon after P reaches its full value of 20 N at $t = 2$ s. We quessed this time to be about 2.4 s.

```
## problem7_1_9
from numpy import zeros,array
from run_kut4 import *
from printSoln import *

def F(x,y):
    m = 2.5; k = 75.0
    if x < 2.0: P = 10.0*x
    else: P = 20.0
    F = zeros(2)
    F[0] = y[1]; F[1] = (P - k*y[0])/m
    return F

x = 0.0                 # Start of integration
xStop = 2.4             # End of integration
y = array([0.0,0.0])    # Initial values of {y}
h = 0.1                 # Step size
freq = 1                # Printout frequency

X,Y = integrate(F,x,y,xStop,h)
printSoln(X,Y,freq)
raw_input(''\nPress return to exit'')
```

Here is the printout of the two points that span the maximum displacement (the point where $\dot{y}$ changes its sign):

```
      x           y[ 0 ]          y[ 1 ]
  2.1000e+00    3.0063e-01     5.0283e-02
  2.2000e+00    3.0044e-01    -5.3863e-02
```

We find the time of maximum displacement from the Taylor series:

$$\begin{aligned}
\dot{y}(2.1+\Delta t) &= \dot{y}(2.1)+\ddot{y}(2.1)\,\Delta t \\
0 &= 0.050\,283+\frac{20-75(0.3006\,3)}{2.5}\Delta t \\
\Delta t &= 0.049\,35 \\
t &= 2.1+0.049\,35 = 2.149 \text{ s}
\end{aligned}$$

The maximum displacement is

$$\begin{aligned}
y_{\text{max}} &= y(2.149) = y(2.1)+\frac{1}{2}\dot{y}(2.1)\Delta t \\
&= 0.300\,63+\frac{1}{2}(0.050\,283)(0.049\,35) = 0.3019 \text{ m} \quad \blacktriangleleft
\end{aligned}$$

Note that $\dot{y}(2.1)$ was multiplied by $1/2$. This factor takes into account the fact that $\dot{y}$ cannot be considered as constant during the time interval Δt, since it varies from $\dot{y}(2.1)$ to zero. Therefore, we must use the *average velocity* during this period, which is $\dot{y}(2.1)/2$.
The computed displacement somewhat bigger than the static displacement (which assumes that the load is applied very slowly) $y_{\text{static}} = P_{\text{max}}/k = 20/75 = 0.2667$ m.

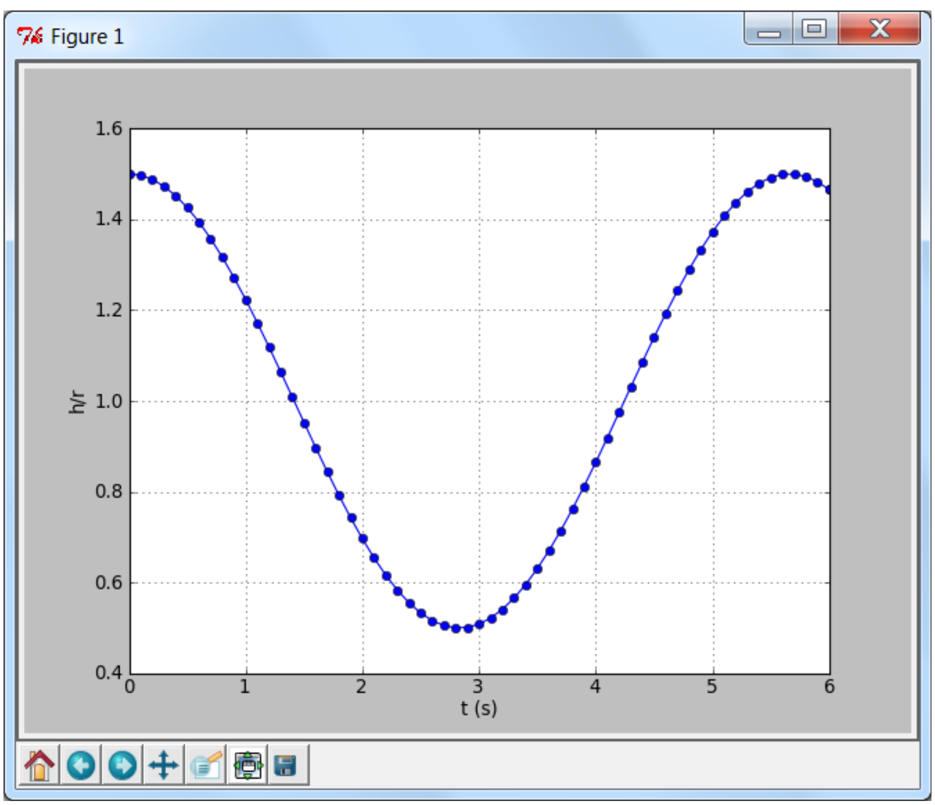
Problem 10

The equivalent first-order differential equations are

$$\mathbf{F} = \begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ \dfrac{2}{\pi}\left[\tan^{-1}\dfrac{1-y_0}{\sqrt{2y_0-y_0^2}} + (1-y_0)\sqrt{2y_0-y_)^2}\right] \end{bmatrix}$$

```
## problem7_1_10
from numpy import zeros,array
from run_kut4 import *
from math import atan,sqrt,pi
import matplotlib.pyplot as plt
```

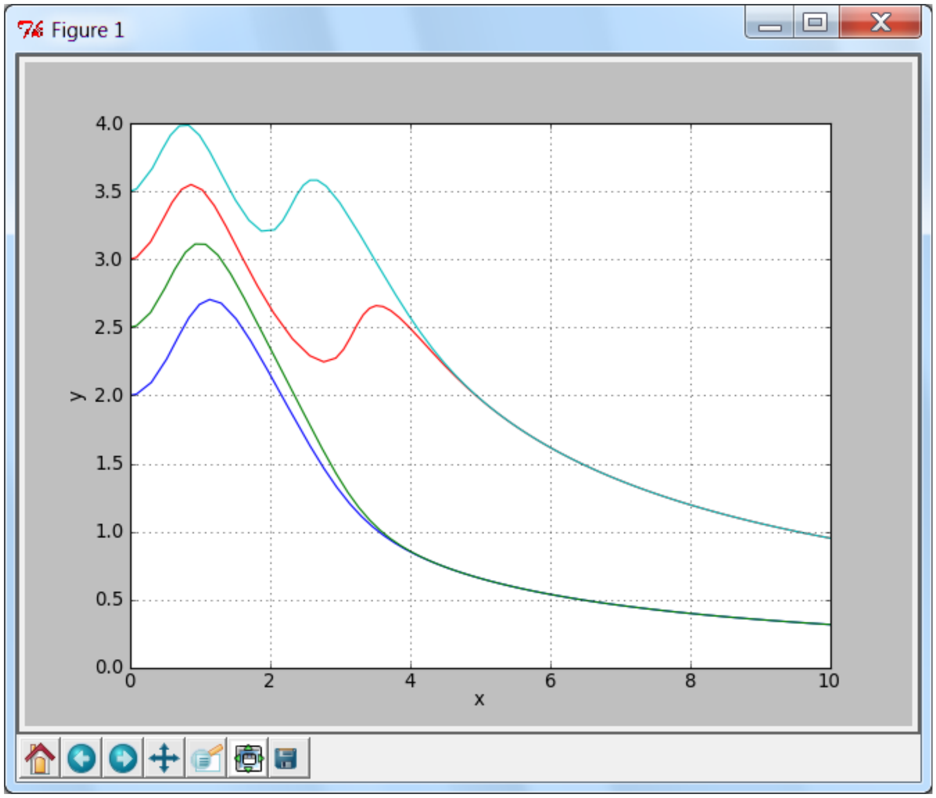
```
def F(t,y):
    p = sqrt(2.0*y[0]-y[0]**2)
    q =1.0 - y[0]
    F = zeros(2)
    F[0] = y[1]
    F[1] = 2.0/pi*(atan(q/p) + q*p)
    return F

t = 0.0               # Start of integration
tStop = 6.0           # End of integration
y = array([1.5,0.0])  # Initial values of {y}
h = 0.1               # Step size

T,Y = integrate(F,t,y,tStop,h)
plt.plot(T,Y[:,0],'o-')
plt.xlabel('t (s)'); plt.ylabel('h (m)')
plt.grid(True)
plt.show()
input("\nPress return to exit")
```



Period is approximately 5.65 s. ◀

Problem 11

```
## problem7_1_11
from numpy import array,zeros
from math import sin
import matplotlib.pyplot as plt
from run_kut4 import *

def F(x,y):
    F = zeros(1)
    F[0] = sin(x*y[0])
    return F

x = 0.0
y = array([[2.0],[2.5],[3.0],[3.5]])
xStop = 10.0
h = 0.1

X1,Y1 = integrate(F,x,y[0],xStop,h)
X2,Y2 = integrate(F,x,y[1],xStop,h)
X3,Y3 = integrate(F,x,y[2],xStop,h)
X4,Y4 = integrate(F,x,y[3],xStop,h)

plt.plot(X1,Y1[:,0],'-',X2,Y2[:,0],'-',  \
         X3,Y3[:,0],'-',X4,Y4[:,0],'-')
plt.xlabel('x'); plt.ylabel('y')
plt.grid(True)
plt.show()
input("Press return to exit")
```



Problem 12

$$\ddot{r} = \left(\frac{\pi^2}{12}\right)^2 r \sin^2 \pi t - g \sin\left(\frac{\pi}{12}\cos \pi t\right) \qquad r(0) = \dot{r}(0) = 0$$

With the notation $y_0 = r$, $y_1 = \dot{r}$ the equivalent first-order differential equations are

$$\mathbf{F} = \begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ (\pi^2/12)^2\, y_0 \sin^2 \pi t - g \sin\left[(\pi/12)\cos \pi t\right] \end{bmatrix}$$

The period of integration (`xStop = 4.0`) and step size (`h = 0.2`) were arrived at by trial-and error. Here a plotting routine proved to be helpful.

```
## problem7_1_12
from numpy import zeros,array
from run_kut4 import *
from printSoln import *
from math import sin,cos,pi

def F(x,y):
    g = 9.80665; c = pi/12.0
```

```
    F = zeros(2)
    F[0] = y[1]
    F[1] = (c*pi)**2*y[0]*sin(pi*x)**2 - g*sin(c*cos(pi*x))
    return F

x = 0.0                # Start of integration
xStop = 4.0            # End of integration
y = array([0.75,0.0]) # Initial values of {y}
h = 0.2                # Step size
freq = 1               # Printout frequency

X,Y = integrate(F,x,y,xStop,h)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

The slider reaches the end of the rod when $r = 2$ m. The two points spanning this event are shown below.

```
     x          y[ 0 ]         y[ 1 ]
 3.4000e+000   1.7123e+000   1.7134e+000
 3.6000e+000   2.0843e+000   1.9615e+000
```

Two-term Taylor series expansion about $t = 3.6$ s yields

$$\begin{aligned} r(3.6 + \Delta t) &= r(3.6) + \dot{r}(3.6)\,\Delta t \\ 2 &= 2.0843 + 1.9615\,\Delta t \\ \Delta t &= -0.04298 \text{ s} \end{aligned}$$

Therefore, the time when the slider leaves the rod is

$$t = 3.6 - 0.04298 = 3.557 \text{ s} \quad \blacktriangleleft$$

Problem 13

$$\begin{aligned} \ddot{x} &= -\frac{C_D}{m}\dot{x}v^{1/2} \qquad \ddot{y} = -\frac{C_D}{m}\dot{y}v^{1/2} - g \\ x(0) &= y(0) = 0 \qquad \dot{x}(0) = 50\cos 30^\circ \qquad \dot{y}(0) = 50\sin 30^\circ \end{aligned}$$

Letting

$$\mathbf{y} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}$$

the first-order differential equations are

$$\mathbf{F} = \begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ -(C_D/m)y_1 v^{1/2} \\ y_3 \\ -(C_D/m)y_3 v^{1/2} - g \end{bmatrix}$$

Without air resistance, the time of flight is $2(v_0 \sin 30°)/g = 2(25)/9.8 \approx 5$ s. Since air resistance reduces the flight time, we quessed `xStop = 4.0`. The time increment h can be quite large here because the trajectory is a smooth curve; `h = 0.2` was considered sufficient.

```
## problem7_1_13
from numpy import zeros,array
from run_kut4 import *
from printSoln import *
from math import sin,cos,pi,sqrt

def F(x,y):
    g = 9.80665; C = 0.03; m = 0.25
    sqrtv = sqrt(sqrt(y[1]**2 + y[3]**2))
    F = zeros(4)
    F[0] = y[1]
    F[1] = -C/m*y[1]*sqrtv
    F[2] = y[3]
    F[3] = -C/m*y[3]*sqrtv - g
    return F

x = 0.0               # Start of integration
xStop = 4.0           # End of integration
y = array([0.0,50.0*cos(pi/6.0),0.0,50.0*sin(pi/6.0)])
                      # Init. values
h = 0.2               # Step size
freq = 1              # Printout frequency

X,Y = integrate(F,x,y,xStop,h)
printSoln(X,Y,freq)
input("\nPress return to exit")
```

Here is the printout of the two points spanning the instant when $y = 0$:

```
     x         y[ 0 ]       y[ 1 ]       y[ 2 ]       y[ 3 ]
3.4000e+00   6.1289e+01   7.0962e+00   9.3838e-01  -1.2671e+01
3.6000e+00   6.2645e+01   6.4720e+00  -1.6732e+00  -1.3430e+01
```

The time of flight can be estimated from the two-term Taylor series expansion of y about $t = 3.4$ s:

$$\begin{aligned} y(3.4 + \Delta t) &= y(3.4) + \dot{y}(3.4)\,\Delta t \\ 0 &= 0.93838 + (-12.671)\,\Delta t \\ \Delta t &= 0.07406 \text{ s} \\ t &= 3.4 + 0.07406 = 3.474 \text{ s} \quad \blacktriangleleft \end{aligned}$$

The range is obtained from the Taylor series expansion of x:

$$\begin{aligned} R &= x(3.4 + \Delta t) = x(3.4) + \dot{x}(3.4)\,\Delta t \\ &= 61.289 + 7.0962(0.07406) = 61.81 \text{ m} \quad \blacktriangleleft \end{aligned}$$

Problem 14

$$\ddot{\theta} = \frac{a(b-\theta) - \theta\dot{\theta}^2}{1+\theta^2} \qquad \theta(0) = 2\pi \qquad \dot{\theta}(0) = 0$$

With the notation $\theta = y_0$, $\dot{\theta} = y_1$, the equivalent first-order differential equations are

$$F = \begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ \left[a(b - y_0) - y_0 y_1^2\right]/(1 + y_0^2) \end{bmatrix}$$

As the time increment h is hard to predict, we let the program find it. We start with $h = 0.1$ and halve the interval in each subsequent integration until successive results differ by less than a prescribed tolerance.

```
## problem7_1_14
from numpy import zeros,array
from run_kut4 import *
from printSoln import *
from math import sin,cos,pi

def F(x,y):
    a = 100.0; b = 15.0
    F = zeros(2)
    F[0] = y[1]
    F[1] = (a*(b - y[0]) - y[0]*y[1]**2)/(1.0 + y[0]**2)
    return F

x = 0.0                  # Start of integration
xStop = 0.5              # End of integration
y = array([2.0*pi,0.0]) # Init. values
h = 0.1                  # Initial step size
```

```
freq = 0                  # Printout frequency
tol = 1.0e-4              # Error tolerance in yMax
yEndOld = 0.0
while 1:
    X,Y = integrate(F,x,y,xStop,h)
    yEnd = Y[len(Y)-1,0]
    if abs(yEnd - yEndOld) < tol: break
    h = h/2.0
    yEndOld = yEnd
print("h =", h)
printSoln(X,Y,freq)
input("\nPress return to exit")
```

By specifying `freq = 0` only the first and last points are printed:

```
h = 0.05
        x          y[ 0 ]        y[ 1 ]
   0.0000e+00    6.2832e+00    0.0000e+00
   5.0000e-01    8.3768e+00    6.7175e+00
```

$$\theta(0.5) = 8.377 \text{ rad} \blacktriangleleft \qquad \dot{\theta}(0.5) = 6.718 \text{ rad/s} \blacktriangleleft$$

Problem 15

$$\ddot{r} = r\dot{\theta}^2 + g\cos\theta - \frac{k}{m}(r - L) \qquad \ddot{\theta} = \frac{-2\dot{r}\dot{\theta} - g\sin\theta}{r}$$

$$r(0) = 0.5 \text{ m} \qquad \dot{r}(0) = 0 \qquad \theta(0) = \frac{\pi}{3} \qquad \dot{\theta}(0) = 0$$

Using the notation

$$\mathbf{y} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} r \\ \dot{r} \\ \theta \\ \dot{\theta} \end{bmatrix}$$

the differential equations are

$$\mathbf{F} = \begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_0 y_3^2 + g\cos y_2 - (k/m)(y_0 - L) \\ y_3 \\ -(2y_1 y_3 + g\sin y_2)/y_0 \end{bmatrix}$$

A pendulum with a stiff arm has a period $\tau = 2\pi\sqrt{L/g}$ for small amplitudes. Although our problem is far removed from a simple pendulum, this formula can still give us a very rough estimate of the time of integration (a quarter of the period):

$$t = \frac{\pi}{2}\sqrt{\frac{0.5}{9.8}} = 0.35 \text{ s}$$

To be on the safe side, the period of integration should be somewhat longer, say, 0.5 s.

```
## problem7_1_15
from numpy import zeros,array
from run_kut4 import *
from printSoln import *
from math import sin,cos,pi

def F(x,y):
    g = 9.80665; k = 40.0; L = 0.5; m = 0.25
    F = zeros(4)
    F[0] = y[1]
    F[1] = y[0]*y[3]**2 + g*cos(y[2]) - k/m*(y[0] - L)
    F[2] = y[3]
    F[3] = -(2.0*y[1]*y[3] + g*sin(y[2]))/y[0]
    return F

x = 0.0                            # Start of integration
xStop = 0.5                        # End of integration
y = array([0.5,0.0,pi/3.0,0.0]) # Init. values
h = 0.025                          # Step size
freq = 1                           # Printout frequency

X,Y = integrate(F,x,y,xStop,h)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

These two points span the time when $\theta = 0$:

```
      x         y[ 0 ]        y[ 1 ]        y[ 2 ]        y[ 3 ]
4.2500e-01   6.1870e-01  -1.3669e-01   5.4800e-02  -3.5924e+00
4.5000e-01   6.1499e-01  -1.5723e-01  -3.5664e-02  -3.6398e+00
```

We first estimate the time t when $\theta = 0$, where $t = 4.5 + \Delta t$ s. We obtain Δt from the Taylor series

$$\begin{aligned}
\theta(4.5 + \Delta t) &= \theta(4.5) + \dot{\theta}(4.5)\,\Delta t \\
0 &= -0.035\,664 + (-3.6398)\,\Delta t \\
\Delta t &= -0.009\,798
\end{aligned}$$

The length of the cord at $\theta = 0$ is given by

$$\begin{aligned} r(4.5+\Delta t) &= r(4.5) + \dot{r}(4.5)\,\Delta t \\ &= 0.614\,99 + (-0.157\,23)(-0.009\,798) \\ &= 0.6165 \text{ m} \quad \blacktriangleleft \end{aligned}$$

Problem 16

Changing the initial condition to $r(0) = 0.575$ m in Problem 15, the points spanning $\theta = 0$ are

x	y[0]	y[1]	y[2]	y[3]
4.0000e-01	6.6414e-01	5.8568e-01	6.0208e-02	-2.8607e+00
4.2500e-01	6.7509e-01	2.8260e-01	-1.0184e-02	-2.7776e+00

The computations now yield

$$\begin{aligned} \theta(4.25+\Delta t) &= \theta(4.25) + \dot{\theta}(4.25)\,\Delta t \\ 0 &= -0.010\,184 + (-2.7776)\,\Delta t \\ \Delta t &= -0.003\,666 \end{aligned}$$

$$\begin{aligned} r(4.25+\Delta t) &= r(4.25) + \dot{r}(4.25)\,\Delta t \\ &= 0.675\,09 + 0.028\,260(-0.003\,666) \\ &= 0.6750 \text{ m} \quad \blacktriangleleft \end{aligned}$$

Problem 17

$$\ddot{y} = -\frac{k}{m}y - \mu g\frac{\dot{y}}{|\dot{y}|} \qquad y(0) = 0.1 \text{ m} \qquad \dot{y}(0) = 0$$

Using the notation $y = y_0$, $\dot{y} = y_1$, the first-order differential equations are

$$\mathbf{F} = \begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ -(k/m)y_0 - \mu g y_1/|y_1| \end{bmatrix}$$

A rough idea of the period of the motion can obtained by removing the friction term from the differential equation. Without friction, the period is $\tau = 2\pi/\sqrt{k/m} = 2\pi/\sqrt{3000/6} \approx 0.28$ s, so that 0.3 s seems to be a reasonable period of integration. We chose for the time increment $h = 0.025$ s, printing out and plotting every 4th point.

```
## problem7_1_17
from numpy import zeros,array
from run_kut4 import *
from printSoln import *

def F(x,y):
    g = 9.80665; k = 3000.0; mu = 0.5; m = 6.0
    F = zeros(2)
    F[0] = y[1]
    if y[1] > 0.0: F[1] = -k/m*y[0] - mu*g
    else: F[1] = -k/m*y[0] + mu*g
    return F

x = 0.0                          # Start of integration
xStop = 0.3                      # End of integration
y = array([0.1, 0.0])            # Init. values
h = 0.0025                       # Step size
freq = 4                         # Printout frequency

X,Y = integrate(F,x,y,xStop,h)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

Here is a printout of the two points spanning the peak displacement:

```
      x            y[ 0 ]         y[ 1 ]
  2.8000e-01    6.0754e-02    3.5807e-02
  2.9000e-01    5.9730e-02   -2.2971e-01
```

The peak displacement occurs at time $t = 0.28 + \Delta t$ when the velociy vanishes. We can compute Δt from the Taylor series

$$\dot{y}(0.28 + \Delta t) = \dot{y}(0.28) + \ddot{y}(0.28)\,\Delta t \tag{a}$$

where

$$\begin{aligned}\ddot{y}(0.28) &= -\frac{k}{m}y(0.28) - \mu g \\ &= -\frac{3000}{6}(0.060\,754) - 0.5(9.806\,65) = -35.280 \text{ m/s}^2\end{aligned}$$

Substitution into Eq. (a) yields

$$\begin{aligned}0 &= 0.035\,807 + (-35.280)\,\Delta t \\ \Delta t &= 0.001\,014\,9 \text{ s}\end{aligned}$$

The peak displacement is given by

$$\begin{aligned}
y(0.28+\Delta t) &= y(0.28)+\frac{1}{2}\dot{y}(0.28)\,\Delta t \\
&= 0.060\,754+\frac{1}{2}0.035807(0.001\,014\,9) \\
&= 0.06077 \text{ m}
\end{aligned}$$

The analytical formula gives for the peak displacement

$$y(0)-4\frac{\mu mg}{k}=0.1-4\frac{0.5(6)(9.80665)}{3000}=0.06077 \text{ m} \quad \text{Checks}$$

Problem 18

We use the notation $y=y_0$, $y'=y_1$ in both problems. Being unable to determine a suitable time increment h beforehand, we let the program do it for us. Starting with an initial guess for h, the program integrates the differential equations with h, $h/2$, $h/4$, etc. until the results of two successive integrations agree within a prescribed tolerance.

(a)

$$y''+0.5(y^2-1)y'+y=0 \qquad y(0)=1 \qquad y'(0)=0$$

This is a version of the well-known Van der Pol equation. The equivalent first-order differential equations are

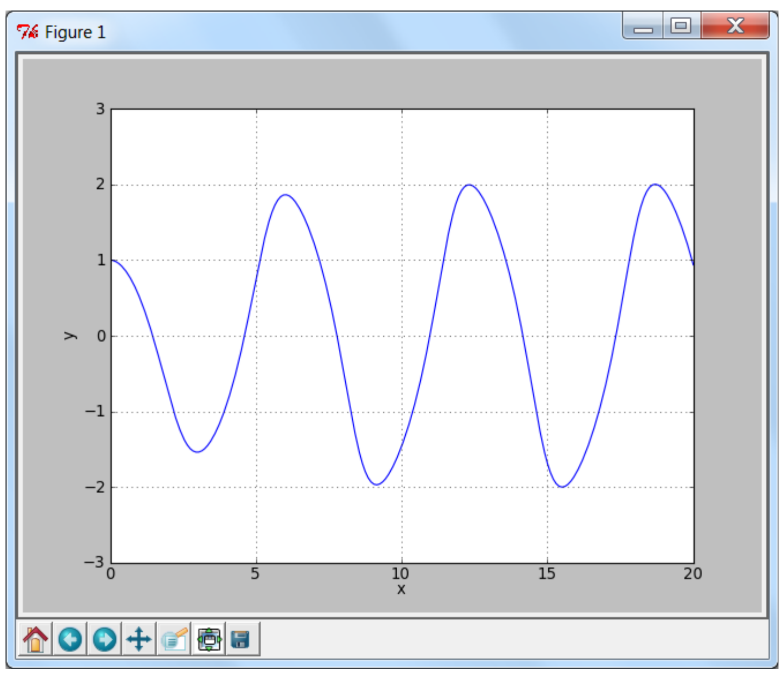
$$\mathbf{F}=\begin{bmatrix} y_0' \\ y_1' \end{bmatrix}-\begin{bmatrix} y_1 \\ -0.5(y_0^2-1)y_1-y_0 \end{bmatrix}$$

```
## problem7_1_18a
from numpy import zeros,array
from run_kut4 import *
import matplotlib.pyplot as plt

def F(x,y):
    F = zeros(2)
    F[0] = y[1]
    F[1] = -0.5*(y[0]**2 - 1.0)*y[1] - y[0]
    return F

x = 0.0                         # Start of integration
xStop = 20.0                    # End of integration
y = array([1.0, 0.0])           # Init. values
```

```
h = 0.2                         # Init. step size
tol = 1.0e-4                    # Error tolerance in y

yEndOld = 0.0
while 1:
    X,Y = integrate(F,x,y,xStop,h)
    yEnd = Y[len(Y)-1,0]
    if abs(yEnd - yEndOld) < tol: break
    h = h/2
    yEndOld = yEnd
print("h =",h)
plt.plot(X,Y[:,0],'-')
plt.xlabel('x'); plt.ylabel('y')
plt.grid(True)
plt.show()
input("\nPress return to exit")
```

The output is:

```
h = 0.05
```



Note that the initial increment $h = 0.2$ was reduced to 0.05 in the last run of the program.

(b)

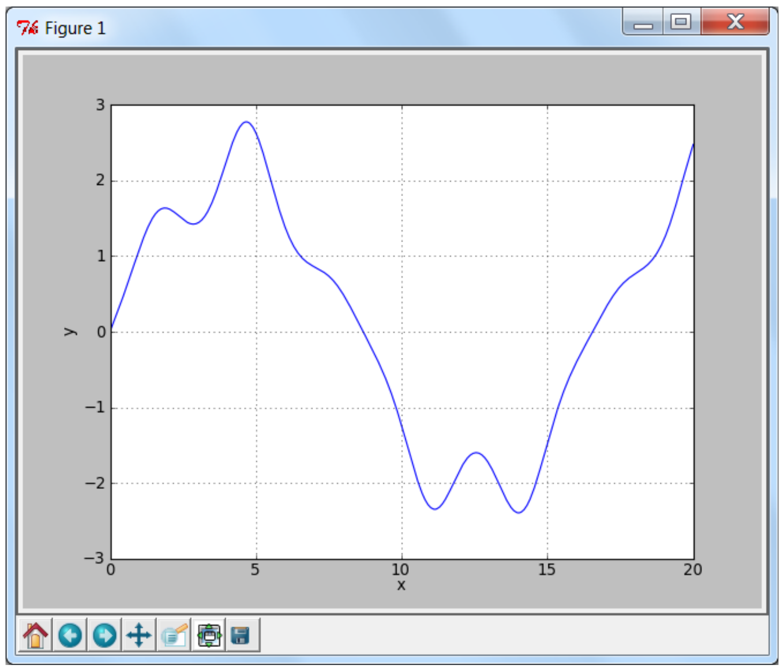
$$y'' = y\cos 2x \qquad y(0) = 0 \qquad y'(0) = 1$$

This differential equation is called Mathieu's equation. The equivalent first-order equations are

$$\mathbf{F} = \begin{bmatrix} y_0' \\ y_1' \end{bmatrix} - \begin{bmatrix} y_1 \\ y_0 \cos 2x \end{bmatrix}$$

We used the program listed in Part (a); only `F(x,y)` and the initial conditions were changed. The ouput is:

```
h = 0.05
```



Problem 19

$$y'' + \frac{1}{x}y' + y = 0 \qquad y(0) = 1 \qquad y'(0) = 0$$

With the notation $y = y_0$, $y' = y_1$, the first-order equations become

$$\mathbf{F} = \begin{bmatrix} y_0' \\ y_1' \end{bmatrix} - \begin{bmatrix} y_1 \\ -y_1/x - y_0 \end{bmatrix}$$

We used the program in Problem 18; only `F(x,y)` and the data was changed:

```
## problem7_1_19
from numpy import zeros,array
from run_kut4 import *
from printSoln import *
```

```
def F(x,y):
    F = zeros(2)
    F[0] = y[1]
    F[1] = -y[1]/x - y[0]
    return F

x = 1.0e-12                       # Start of integration
xStop = 5.0                       # End of integration
y = array([1.0, 0.0])             # Init. values
h = 0.2                           # Init. step size
freq = 0                          # Printout frequency
tol = 1.0e-4                      # Error tolerance in y

yEndOld = 0.0
while 1:
    X,Y = integrate(F,x,y,xStop,h)
    yEnd = Y[len(Y)-1,0]
    if abs(yEnd - yEndOld) < tol: break
    h = h/2
    yEndOld = yEnd
print(''h ='',h)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

```
h = 0.025
        x          y[ 0 ]         y[ 1 ]
  1.0000e-12    1.0000e+00    0.0000e+00
  5.0000e+00   -1.7759e-01    3.2756e-01
```

The relatively small increment $h = 0.025$ was needed to attain satisfactory argeement with the tabulated value.

Problem 20

(a)

$$y'' = 16.81y \qquad y(0) = 1.0 \qquad y'(0) = -4.1$$

Solution of the differential equation is (note that $\sqrt{16.81} = 4.1$)

$$y = Ae^{4.1x} + Be^{-4.1x}$$

The initial conditions yield

$$\begin{aligned} y(0) &= A+B=1 \qquad y'(0)=4.1(A-B)=-4.1 \\ A &= 0 \qquad B=1 \end{aligned}$$

Therefore,

$$y = e^{-4.1x} \quad \blacktriangleleft$$

(b)

As numerical integration proceeds, the dormant term $Ae^{4.1x}$ will become alive and eventually dominates the solution. This is a case numerical instability caused by sensitivity of the solution to initial conditions. Numerical integration will not work here.

(c)

Using the 4th-order Runge-Kutta method with $h = 0.1$, the initial and final points of the solution are

```
       x          y[ 0 ]        y[ 1 ]
  0.0000e+00   1.0000e+00   0.0000e+00
  8.0000e+00   8.7386e+13   3.5828e+14
```

Clearly the numerical solution is unstable.

Problem 21

$$\begin{aligned} \frac{di_1}{dt} &= \frac{-3Ri_1 - 2Ri_2 + E}{L} \\ \frac{di_2}{dt} &= -\frac{2}{3}\frac{di_1}{dt} - \frac{i_2}{3RC} + \frac{\dot{E}}{3R} \\ i_1(0) &= i_2(0) = 0 \end{aligned}$$

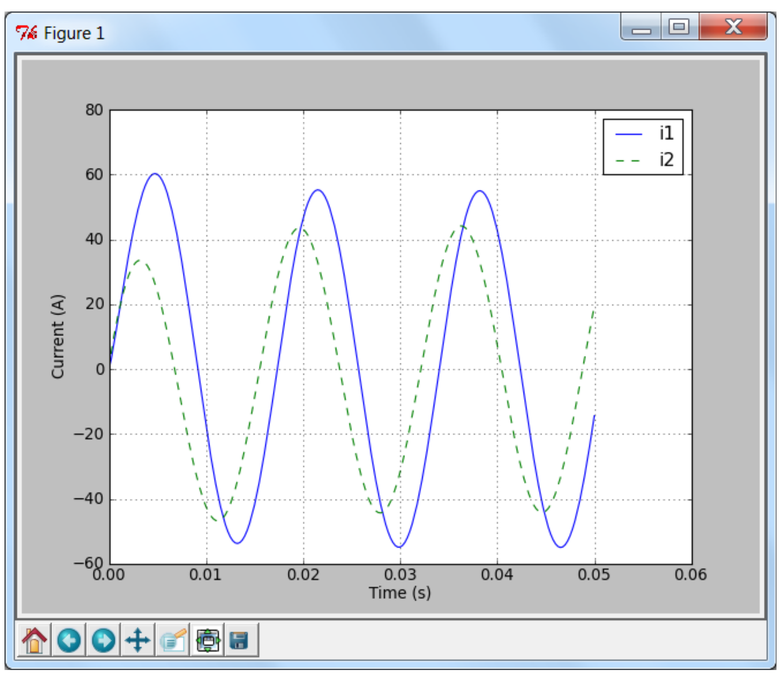
Using the notation $i_1 = y_0$, $i_2 = y_1$, the differential equations are

$$F = \begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} (-3Ry_0 - 2Ry_1 + E)/L \\ \left[-2F_1 - y_1/(RC) + \dot{E}/R\right]/3 \end{bmatrix}$$

```
## problem7_1_21
from numpy import zeros,array
from run_kut4 import *
import matplotlib.pyplot as plt
```

```
from math import sin,cos,pi

def F(x,y):
    R = 1.0; L = 0.2e-3; C = 3.5e-3;
    E = 240.0*sin(120.0*pi*x)
    dE = 240.0*120.0*pi*cos(120.0*pi*x)
    F = zeros(2)
    F[0] = (-3.0*R*y[0] - 2.0*R*y[1] + E)/L
    F[1] = (-2.0*F[0] - y[1]/R/C + dE/R)/3.0
    return F

x = 0.0
y = array([0.0, 0.0])
xStop = 0.05
h = 0.00025
X,Y = integrate(F,x,y,xStop,h)
plt.plot(X,Y[:,0],'-',X,Y[:,1],'--')
plt.xlabel('Time (s)'); plt.ylabel('Current (A)')
plt.grid(True)
plt.legend(('i1','i2'),loc=0)
plt.show()
input("\nPress return to exit")
```

Here are the plots of the currents:



Problem 22

$$
\begin{aligned}
L\frac{di_1}{dt} + Ri_1 + \frac{q_1 - q_2}{C} &= E \\
L\frac{di_2}{dt} + Ri_2 + \frac{q_2 - q_1}{C} + \frac{q_2}{C} &= 0
\end{aligned}
$$

$$
\begin{aligned}
\frac{dq_1}{dt} &= i_1 \qquad \frac{dq_2}{dt} = i_2 \\
q_1(0) &= q_2(0) = i_1(0) = i_2(0) = 0
\end{aligned}
$$

With the notation

$$
\mathbf{y} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ i_1 \\ i_2 \end{bmatrix}
$$

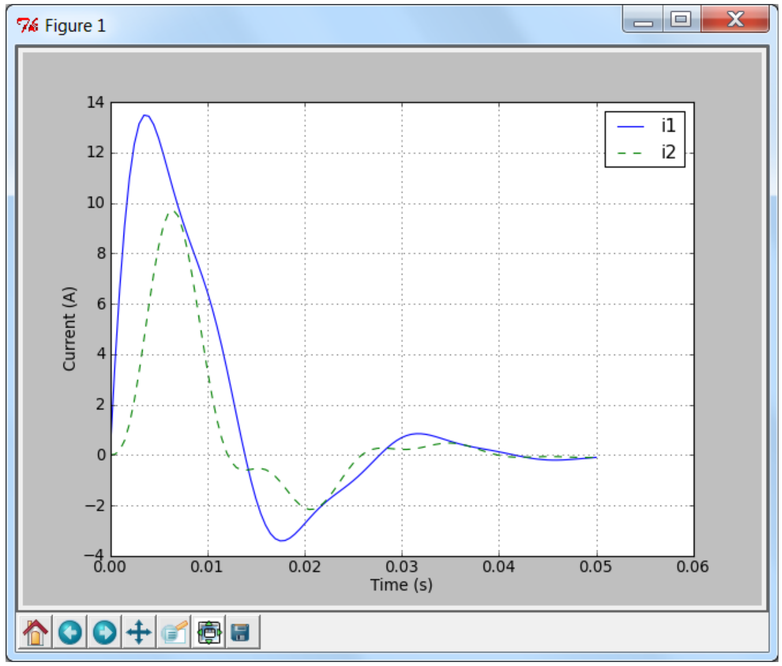
the first-order differential equations are

$$
\mathbf{F} = \begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \end{bmatrix} = \begin{bmatrix} y_2 \\ y_3 \\ \left[E - Ry_2 - (y_0 - y_1)/C\right]/L \\ \left[-Ry_3 - (y_1 - y_0)/C - y_1/C\right]/L \end{bmatrix}
$$

```
## problem7_1_22
from numpy import zeros,array
from run_kut4 import *
import matplotlib.pyplot as plt
def F(x,y):
    R = 0.25; L = 1.2e-3; C = 5.0e-3; E = 9.0
    F = zeros(4)
    F[0] = y[2]
    F[1] = y[3]
    F[2] = (E - R*y[2] - (y[0] - y[1])/C )/L
    F[3] = (-R*y[3] - (y[1] - y[0])/C - y[1]/C)/L
    return F

x = 0.0
y = array([0.0, 0.0, 0.0, 0.0])
xStop = 0.05
h = 0.0005
X,Y = integrate(F,x,y,xStop,h)
plt.plot(X,Y[:,2],'-',X,Y[:,3],'--')
plt.xlabel('Time (s)'); plt.ylabel('Current (A)')
plt.grid(True)
```

```
plt.legend(('i1','i2'),loc=0)
plt.show()
input("\nPress return to exit")
```

Here are the plots of the currents:



Problem 23

The integral

$$y(x) = \int_0^x \frac{\sin t}{t} dt$$

is the solution of the initial value problem

$$\dot{y} = \frac{\sin t}{t} \qquad y(0) = 0$$

at $t = x$. We chose the 4th-order Runge-Kutta method with $h = 0.05$, printing every 5th integration step.

```
## problem7_1_23
from numpy import zeros,array
from run_kut4 import *
from printSoln import *
from math import sin
```

```
def F(x,y):
    F = zeros(1)
    if x ==0: F[0] = 1.0
    else: F[0] = sin(x)/x
    return F

x = 0.0                # Start of integration
xStop = 3.6            # End of integration
y = array([0.0])       # Initial values of {y}
h = 0.04               # Step size
freq = 5               # Printout frequency
X,Y = integrate(F,x,y,xStop,h)
printSoln(X,Y,freq)
input("\nPress return to exit")
```

```
        x           y[ 0 ]
   0.0000e+00    0.0000e+00
   2.0000e-01    1.9956e-01
   4.0000e-01    3.9646e-01
   6.0000e-01    5.8813e-01
   8.0000e-01    7.7210e-01
   1.0000e+00    9.4608e-01
   1.2000e+00    1.1080e+00
   1.4000e+00    1.2562e+00
   1.6000e+00    1.3892e+00
   1.8000e+00    1.5058e+00
   2.0000e+00    1.6054e+00
   2.2000e+00    1.6876e+00
   2.4000e+00    1.7525e+00
   2.6000e+00    1.8004e+00
   2.8000e+00    1.8321e+00
   3.0000e+00    1.8487e+00
   3.2000e+00    1.8514e+00
   3.4000e+00    1.8419e+00
   3.6000e+00    1.8219e+00
```

These values are in agreement with published tables of the sine integral.

PROBLEM SET 7.2

Problem 1

$$y'' = 380y - y' \qquad y(0) = 1 \qquad y'(0) = -20$$

With $y = y_0$, $y' = y_1$ the equivalent first-order differential equations are

$$\begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 380 & -1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}$$

These equations are of the form $\dot{\mathbf{y}} = -\boldsymbol{\Lambda}\mathbf{y}$, where

$$\boldsymbol{\Lambda} = \begin{bmatrix} 0 & -1 \\ -380 & 1 \end{bmatrix}$$

The eigenvalues of $\boldsymbol{\Lambda}$ are the roots of

$$\begin{vmatrix} 0 - \lambda & -1 \\ -380 & 1 - \lambda \end{vmatrix} = 0 \qquad \lambda^2 - \lambda - 380 = 0$$

which yields $\lambda_1 = -19$, $\lambda_2 = 20$. Therefore,

$$y = C_1 e^{-\lambda_1 x} + C_2 e^{-\lambda_2 x} = C_1 e^{19x} + C_2 e^{-20x}$$

From the initial conditions we get

$$\begin{aligned} y(0) &= 1: \qquad C_1 + C_2 = 0 \\ y'(0) &= -20: \qquad 19C_1 - 20C_2 = -20 \\ C_1 &= 0 \qquad C_2 = 1 \end{aligned}$$

so that $y = e^{-20x}$ ◀

It would be difficult to obtain the solution numerically due to the dormant term $C_1 e^{19x}$.

Problem 2

$$y' = x - 10y \qquad y(0) = 10$$

(a)

$$\begin{aligned}
y &= 0.1x - 0.01 + 10.01e^{-10x} \\
y' &= 0.1 - 100.1e^{-10x} \\
x - 10y &= x - 10(0.1x - 0.01 + 10.01e^{-10x}) \\
&= 0.1 - 100.1e^{-10x} = y' \text{ Q.E.D.}
\end{aligned}$$

$$y(0) = 0 - 0.01 + 10.01 = 10 \text{ Checks}$$

(b)

$$h < \frac{2}{\lambda} \qquad h < \frac{2}{10} = 0.2 \ \blacktriangleleft$$

Problem 3

The analytical solution is

$$y(5) = 0.1(5) - 0.01 + 10.01e^{-10(5)} = 0.4900$$

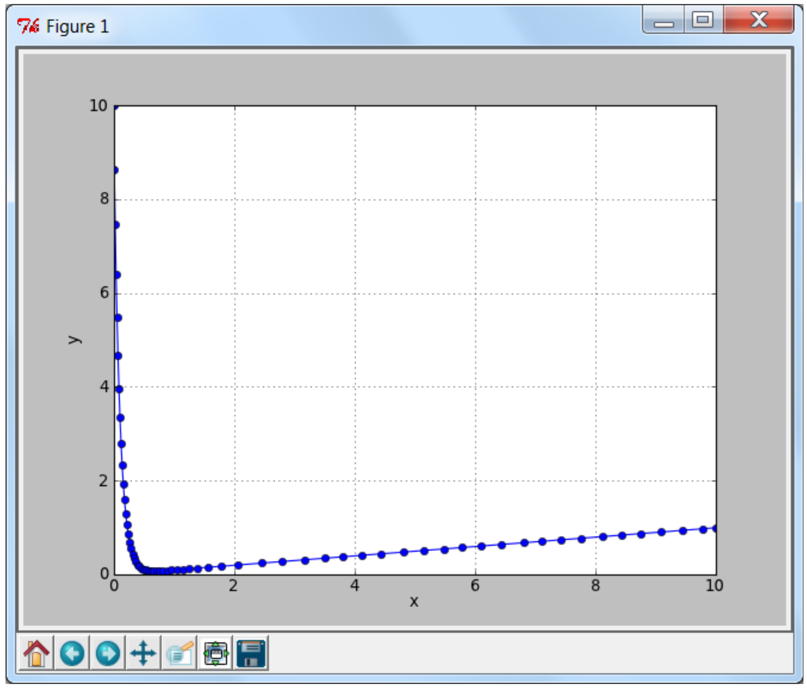
```
## problem7_2_3
from numpy import zeros,array
from run_kut4 import *
from printSoln import *

def F(x,y):
    F = zeros(1)
    F[0] = x - 10.0*y[0]
    return F

x = 0.0                         # Start of integration
xStop = 5.0                     # End of integration
y = array([5.0])                # Init. values
h = array([0.1, 0.25, 0.5])     # Step size
freq = 0                        # Printout frequency

for i in range(len(h)):
    X,Y = integrate(F,x,y,xStop,h[i])
    print("\nh =",h[i])
    printSoln(X,Y,freq)
input("\nPress return to exit")

h = 0.1
```

```
        x          y[ 0 ]
   0.0000e+00    5.0000e+00
   5.0000e+00    4.9000e-01

h = 0.25
        x          y[ 0 ]
   0.0000e+00    5.0000e+00
   5.0000e+00    4.9087e-01
$\partial $
h = 0.5
        x          y[ 0 ]
   0.0000e+00    5.0000e+00
   5.0000e+00    1.1740e+12
```

In Problem 2 the stable range of h was estimated as $h < 0.2$. Thus $h = 0.1$ is stable and $h = 0.5$ is unstable, as verified by the numerical results. On the other hand. $h = 0.25$ is close to the borderline—it is stable in the specified range of integration, but not accurate.

Problem 4

```
## problem7_2_4
from numpy import zeros,array
from run_kut5 import *
import matplotlib.pyplot as plt

def F(x,y):
    F = zeros(1)
    F[0] = x - 10.0*y[0]
    return F

x = 0.0                         # Start of integration
xStop = 10.0                    # End of integration
y = array([10.0])               # Init. values
h = 0.2                         # Initial step size
X,Y = integrate(F,x,y,xStop,h)
plt.plot(X,Y,'o-')
plt.xlabel('x'); plt.ylabel('y')
plt.grid(True)
plt.show()
input("\nPress return to exit")
```



Note the high density of points where y varies rapidly.

Problem 5

$$\ddot{y} = -\frac{c}{m}\dot{y} - \frac{k}{m}y \qquad y(0) = 0.01 \text{ m} \qquad \dot{y}(0) = 0$$

(a)

With $y = y_0$, $\dot{y} = y_1$ the equivalent first-order differential equations are

$$\begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}$$

These equations are of the form $\dot{\mathbf{y}} = -\mathbf{\Lambda}\mathbf{y}$, where

$$\mathbf{\Lambda} = \begin{bmatrix} 0 & -1 \\ k/m & c/m \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 450/2 & 460/2 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 225 & 230 \end{bmatrix}$$

The eigenvalues of $\mathbf{\Lambda}$ are the roots of

$$\begin{vmatrix} 0 - \lambda & -1 \\ 225 & 230 - \lambda \end{vmatrix} = 0 \qquad \lambda^2 - 230\lambda + 225 = 0$$

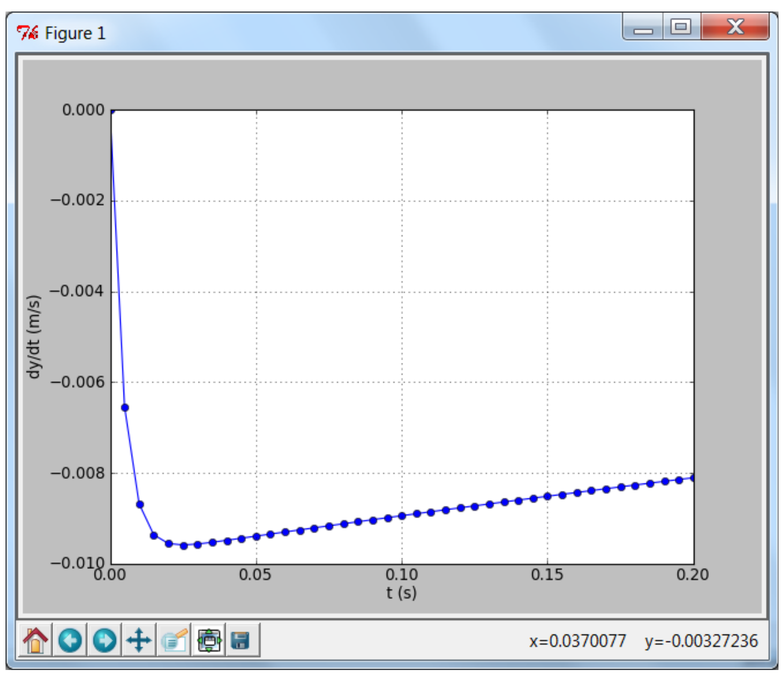
The solution is $\lambda_1 = 0.982\,458$, $\lambda_2 = 229.0175$. Since there is a large disparity in the eigenvalues, the problem is stiff. Numerical integration requires

$$h < \frac{2}{\lambda_2} = \frac{2}{229.0175} = 0.008\,733$$

A reasonable choice would be $h = 0.005$ ◀

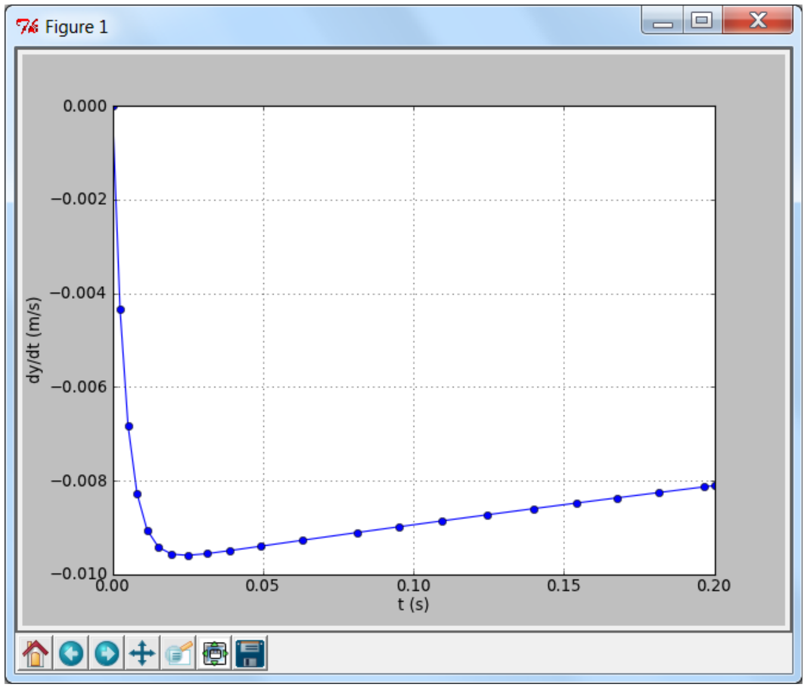
(b)

```
## problem7_2_5
from numpy import zeros,array
from run_kut4 import *
import matplotlib.pyplot as plt

def F(x,y):
    F = zeros(2)
    F[0] = y[1]
    F[1] = -225.0*y[0] - 230.0*y[1]
    return F

x = 0.0                          # Start of integration
xStop = 0.2                      # End of integration
y = array([0.01, 0.0])           # Init. values
h = 0.005                        # Step size

X,Y = integrate(F,x,y,xStop,h)
plt.plot(X,Y[:,1],'-o')
plt.xlabel('t (s)'); plt.ylabel('dy/dt (m/s)')
plt.grid(True); plt.show()
input("\nPress return to exit")
```



Problem 6

The program in Problem 5 was used with `run_kut4` in the `import` statement replaced by `run_kut5`.



Note the larger h used in the region $t > 0.05$ s than in Problem 5.

Problem 7

$$y'' = 16.81y$$

This problem was integrated (unsuccessfully) with the non-adaptive Runge-Kutta method in Problem 20, Problem Set 7.1. The analytical solution is

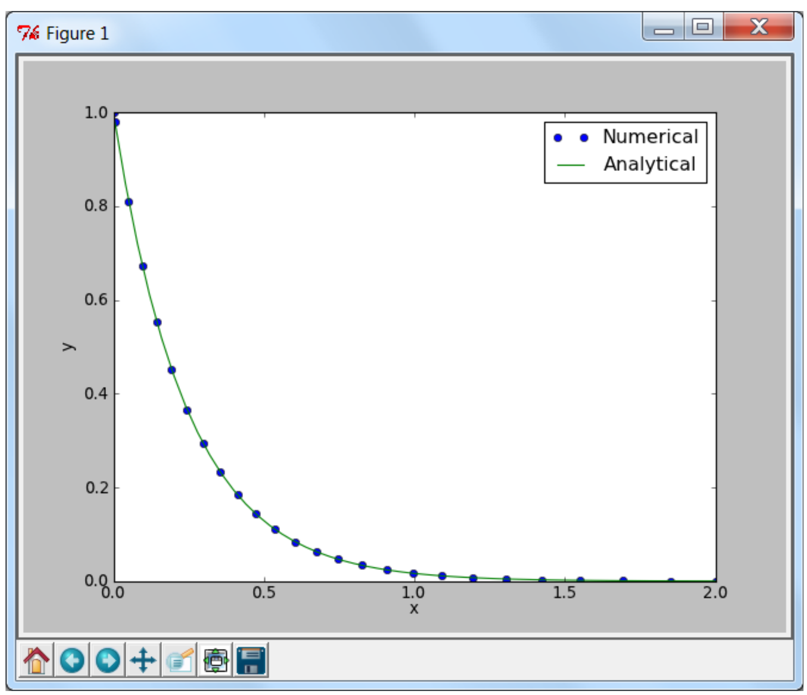
$$y = Ae^{4.1x} + Be^{-4.1x} \tag{a}$$

(a)

The initial conditions $y(0) = 1$, $y'(0) = -4.1$ yield $A = 0$, $B = 1$, so the the analytical solution becomes $y = e^{-4.1x}$.

```
## problem7_2_7
from numpy import zeros,array,arange,exp
from run_kut5 import *
```

```
import matplotlib.pyplot as plt

def F(x,y):
    F = zeros(2)
    F[0] = y[1]
    F[1] = 16.81*y[0]
    return F

x = 0.0                          # Start of integration
xStop = 2.0                      # End of integration
y = array([1.0, -4.1])           # Init. values
h = 0.005                        # Step size

xx = arange(0.0, 2.04, 0.04)
yy = exp(-4.1*xx)                # Analytical solution

X,Y = integrate(F,x,y,xStop,h)
plt.plot(X,Y[:,0],'o',xx,yy,'-')
plt.xlabel('x'); plt.ylabel('y')
plt.legend(('Numerical','Analytical'),loc=0)
plt.show()
input("\nPress return to exit")
```



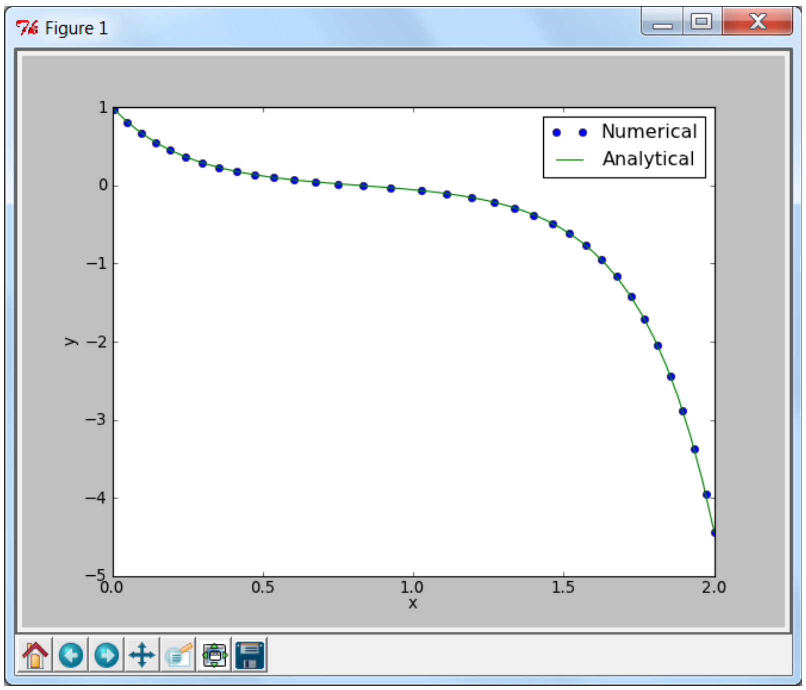
The numerical results reproduce the analytical solution quite closely. The effect of the dormant term $Ae^{4.1x}$ has not yet appeared in the numerical solution.

(b)

The initial conditions are $y(0) = 1$, $y'(0) = -4.11$. Substituting these conditions into Eq. (a) gives us $A = -1.2195 \times 10^{-3}$ and $B = 1.0012$. Thus the analytical solution is

$$y = -1.2195 \times 10^{-3} e^{4.1x} + 1.0012 e^{-4.1x}$$

The numerical solution was computed with the program in Part (a); the initial condition on y' and the expression for the analytical solution were the only changes.



The solution is initially dominated by the term $Be^{-4.1x}$, but the term $Ae^{4.1x}$ rapidly gains prominence beyond $x = 1$.

Problem 8

$$y'' = -y' + y^2 \qquad y(0) = 1 \qquad y'(0) = 0$$

```
## problem7_2_8
from numpy import zeros,array
from run_kut5 import *
from printSoln import *

def F(x,y):
```

```
    F = zeros(2)
    F[0] = y[1]
    F[1] = -y[1] + y[0]**2
    return F

x = 0.0                         # Start of integration
xStop = 3.5                     # End of integration
y = array([1.0, 0.0])           # Init. values
h = 0.05                        # Step size
freq = 25                       # Printout frequency

X,Y = integrate(F,x,y,xStop,h,tol=1.0e-6)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

The numerical integration was carried out with a per-step error tolerance 10^{-6}. Only every 20th step was printed.

```
      x           y[ 0 ]          y[ 1 ]
   0.0000e+00    1.0000e+00     0.0000e+00
   2.7596e+00    1.2151e+01     3.0193e+01
   3.2329e+00    8.0846e+01     5.6246e+02
   3.3571e+00    2.5505e+02     3.2261e+03
   3.4106e+00    5.8518e+02     1.1328e+04
   3.4393e+00    1.1265e+03     3.0425e+04
   3.4569e+00    1.9377e+03     6.8873e+04
   3.4685e+00    3.0804e+03     1.3837e+05
   3.4767e+00    4.6193e+03     2.5450e+05
   3.4826e+00    6.6212e+03     4.3727e+05
   3.4872e+00    9.1556e+03     7.1165e+05
   3.4907e+00    1.2294e+04     1.1081e+06
   3.4935e+00    1.6110e+04     1.6631e+06
   3.4958e+00    2.0679e+04     2.4198e+06
   3.4977e+00    2.6079e+04     3.4283e+06
   3.4992e+00    3.2389e+04     4.7464e+06
   3.5000e+00    3.6475e+04     5.6733e+06
```

Note that the step size h rapidly diminishes as x approaches 3.5. At the same time, y appears to approach infinity. If this is caused by numerical instability, the results should be sensitive to the per-step error tolerance `tol` used in the integration (tighter tolerance reduces the truncation error thus delaying the onset of instability). We tested for instability by re-running the program with `tol` set to 10^{-4} and 10^{-8}. Here are the results:

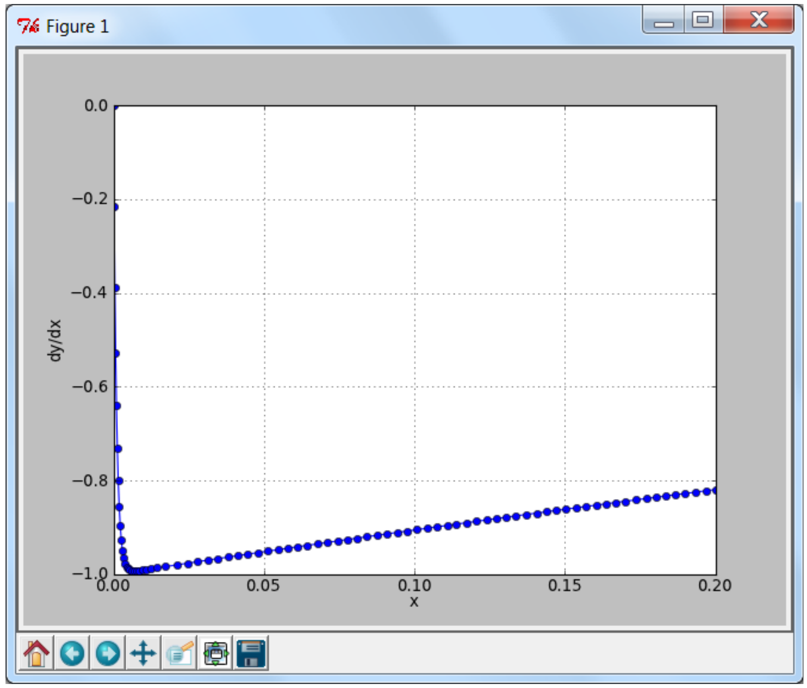
`tol`	$y(3.5)$
10^{-4}	$3.6414e+04$
10^{-6}	$3.6475e+04$
10^{-8}	$3.6475e+04$

Since changing `tol` made no significant difference, we conclude that the sudden increase in y is real.

Problem 9

$$y'' = -1001y' - 1000y \qquad y(0) = 1 \qquad y'(0) = 0$$

```
## problem7_2_9
from numpy import zeros,array
from run_kut5 import *
import matplotlib.pyplot as plt

def F(x,y):
    F = zeros(2)
    F[0] = y[1]
    F[1] = -1001.0*y[1] - 1000.0*y[0]
    return F

x = 0.0                         # Start of integration
xStop = 0.2                     # End of integration
y = array([1.0, 0.0])           # Init. values
h = 0.05                        # Step size

X,Y = integrate(F,x,y,xStop,h)
plt.plot(X,Y[:,1],'-o')
plt.xlabel('x'); plt.ylabel('dy/dx')
plt.grid(True); plt.show()
input("\nPress return to exit")
```



The adaptive quadrature had no trouble overcoming the stiffness of this problem.

Problem 10

$$y'' = -2y' - 3y \qquad y(0) = 0 \qquad y'(0) = \sqrt{2}$$

```
## problem7_2_10
from numpy import zeros,array,sin,sqrt,exp
from run_kut5 import *
from printSoln import *

def F(x,y):
    F = zeros(2)
    F[0] = y[1]
    F[1] = -2.0*y[1] - 3.0*y[0]
    return F

x = 0.0
y = array([0.0, sqrt(2)])
xStop = 5.0
h = 0.1
freq = 2
```

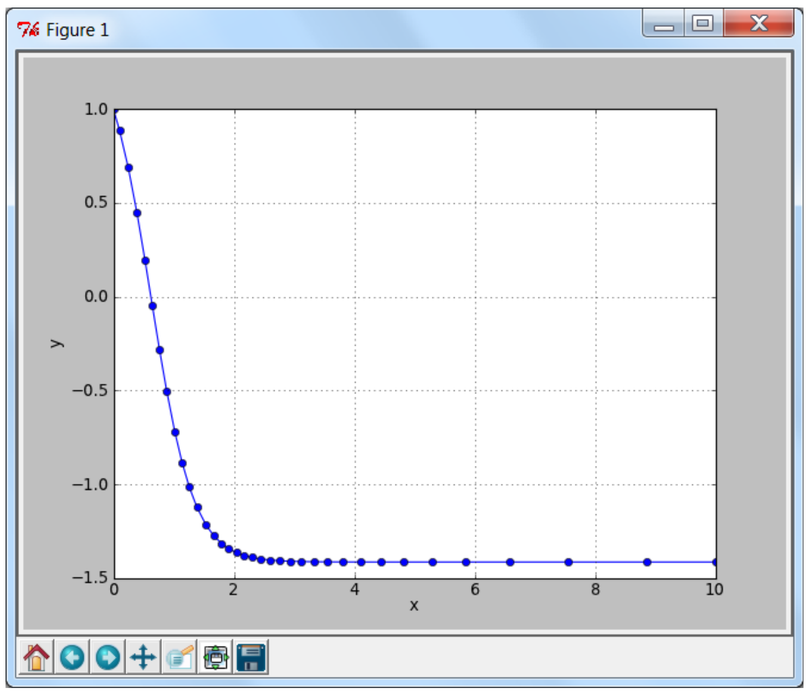
```
X,Y = integrate(F,x,y,xStop,h)
# Add a column to Y to make room for the analytical
# solution; store the result in array YY.
YY = zeros((len(Y),3))
for i in range(len(Y)):
    YY[i,0] = Y[i,0]; YY[i,1] = Y[i,1]
    YY[i,2] = exp(-X[i])*sin(sqrt(2)*X[i])
print("                Numerical y   Numerical y'" \
      "  Analytical y")
printSoln(X,YY,freq)
input("\nPress return to exit")
                Numerical y   Numerical y'  Analytical y

        x          y[ 0 ]        y[ 1 ]        y[ 2 ]
   0.0000e+00    0.0000e+00    1.4142e+00    0.0000e+00
   2.3761e-01    2.6001e-01    7.9274e-01    2.6001e-01
   5.1292e-01    3.9722e-01    2.3637e-01    3.9722e-01
   7.9427e-01    4.0741e-01   -1.3084e-01    4.0741e-01
   1.0897e+00    3.3617e-01   -3.2203e-01    3.3617e-01
   1.4092e+00    2.2290e-01   -3.6446e-01    2.2290e-01
   1.7684e+00    1.0197e-01   -2.9539e-01    1.0197e-01
   2.1856e+00    5.6936e-03   -1.6446e-01    5.6936e-03
   2.6197e+00   -3.8881e-02   -4.8208e-02   -3.8880e-02
   3.0511e+00   -4.3619e-02    1.7724e-02   -4.3618e-02
   3.5090e+00   -2.8995e-02    3.9471e-02   -2.8995e-02
   4.0295e+00   -9.8155e-03    3.0789e-02   -9.8153e-03
   4.6682e+00    2.9416e-03    9.6688e-03    2.9415e-03
   5.0000e+00    4.7766e-03    1.9451e-03    4.7763e-03
```

Problem 11

$$y'' = 2yy' \qquad y(0) = 1 \qquad y'(0) = -1$$

```
## problem7_2_11
from numpy import zeros,array
from run_kut5 import *
import matplotlib.pyplot as plt

def F(x,y):
    F = zeros(2)
    F[0] = y[1]
```

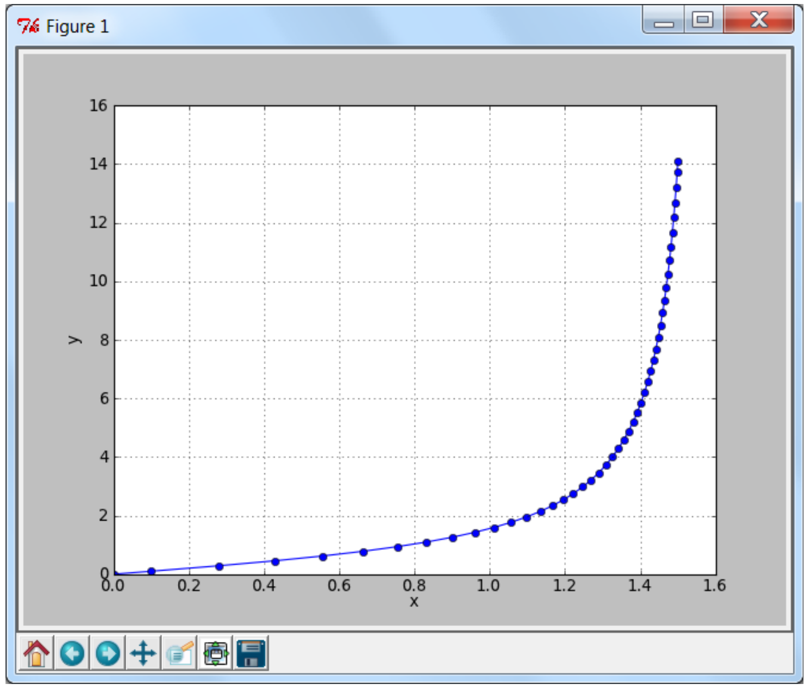
```
    F[1] = 2.0*y[0]*y[1]
    return F

x = 0.0
y = array([1.0, -1.0])
xStop = 10.0
h = 0.1
X,Y = integrate(F,x,y,xStop,h)
plt.plot(X,Y[:,0],'-o')
plt.xlabel('x'); plt.ylabel('y')
plt.grid(True); plt.show()
input("\nPress return to exit")
```



Problem 12

$$y'' = 2yy' \qquad y(0) = 0 \qquad y'(0) = 1$$

We used the program in Problem 11.



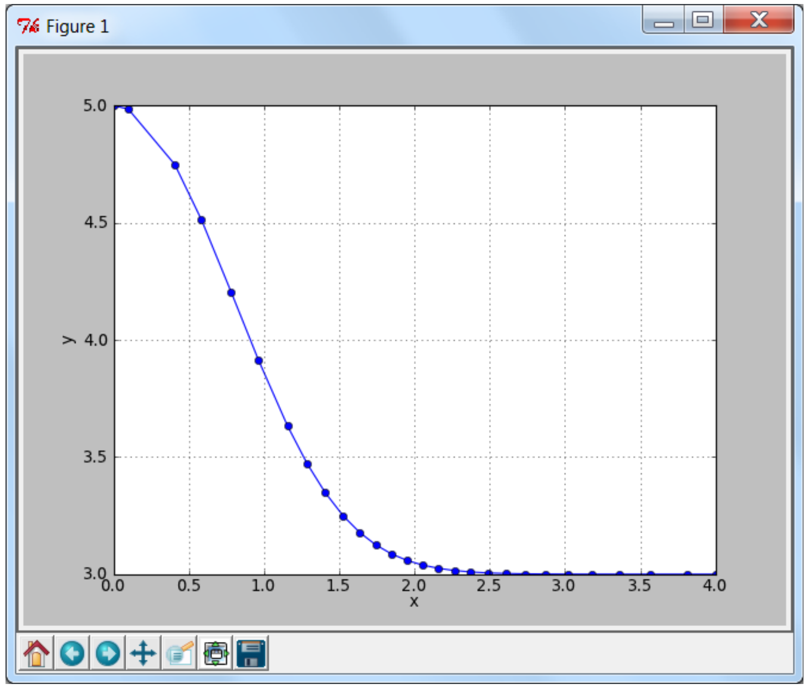
The analytical solution is $y = \tan x$.

Problem 13

$$y' = \left(\frac{9}{y} - y\right)x \qquad y(0) = 5$$

```
## problem7_2_13
from numpy import zeros,array
from run_kut5 import *
import matplotlib.pyplot as plt

def F(x,y):
    F = zeros(1)
    F[0] = (9.0/y[0] - y[0])*x
    return F

x = 0.0
y = array([5.0])
xStop = 4.0
h = 0.1
X,Y = integrate(F,x,y,xStop,h)
plt.plot(X,Y[:,0],'-o')
plt.xlabel('x'); plt.ylabel('y')
```

```
plt.grid(True); plt.show()
input("\nPress return to exit")
```

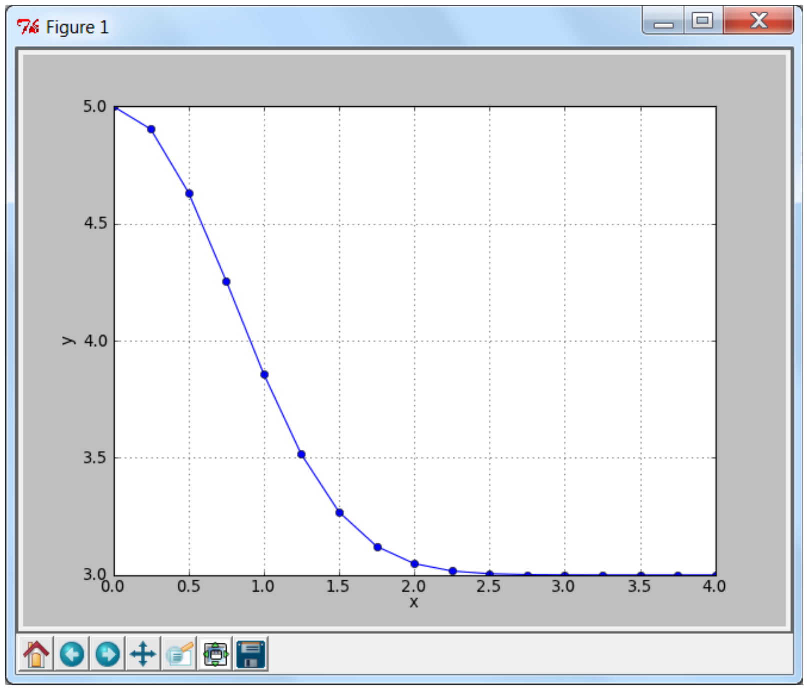


Problem 14

$$y' = \left(\frac{9}{y} - y\right)x \qquad y(0) = 5$$

```
## problem7_2_14
from bulStoer import *
from numpy import array,zeros
import matplotlib.pyplot as plt

def F(x,y):
    F = zeros(1)
    F[0] = (9.0/y[0] - y[0])*x
    return F

H = 0.25
xStop = 4.0
x = 0.0
y = array([5.0])
X,Y = bulStoer(F,x,y,xStop,H)
```

```
plt.plot(X,Y[:,0],'-o')
plt.xlabel('x'); plt.ylabel('y')
plt.grid(True); plt.show()
input("\nPress return to exit")
```

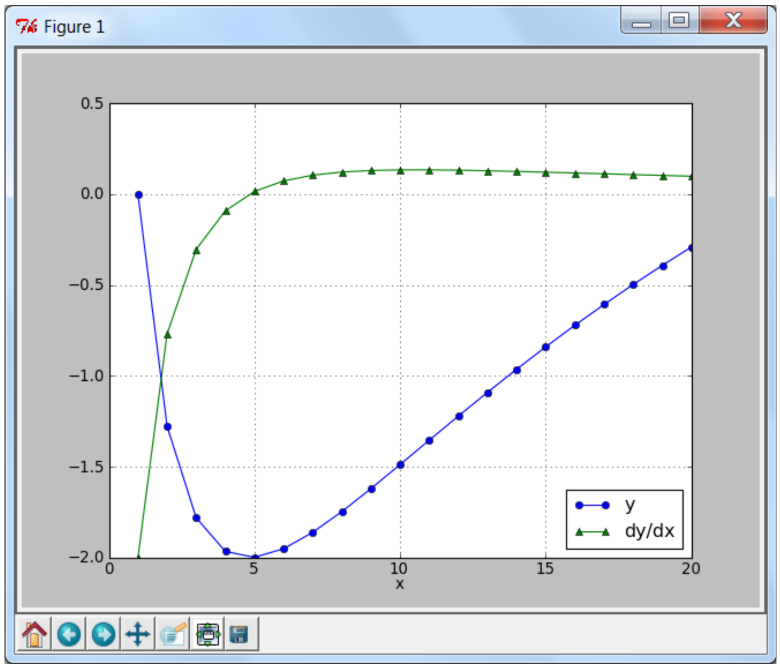


Problem 15

$$y'' = -\frac{1}{x}y' - \frac{1}{x^2}y \qquad y(1) = 0 \qquad y'(1) = -2$$

```
## problem7_2_15
from bulStoer import *
from numpy import array,zeros
import matplotlib.pyplot as plt

def F(x,y):
    F = zeros(2)
    F[0] = y[1]
    F[1] = -y[1]/x - y[0]/x**2
    return F

H = 1.0
xStop = 20.0
x = 1.0
```

```
y = array([0.0,-2.0])
X,Y = bulStoer(F,x,y,xStop,H)
plt.plot(X,Y[:,0],'-o',X,Y[:,1],'-^')
plt.xlabel('x')
plt.legend(('y','dy/dx'),loc=4)
plt.grid(True); plt.show()
input("\nPress return to exit")
```



Problem 16

$$\ddot{x} = \frac{c}{m}\frac{1}{x^2} - \frac{k}{m}(x - L) \qquad x(0) = L \qquad \dot{x}(0) = 0$$

In the program we use the notation $x = y_0$ and $\dot{x} = y_1$. An estimate of the period is $\tau = 2\pi/\sqrt{k/m} = 2\pi/\sqrt{120/1} = 0.57$ s, which is the period of a mass-spring system. We played is safe and used 0.6 s in the program.

```
## problem7_2_16
from numpy import zeros,array
from run_kut5 import *
from printSoln import *

def F(x,y):
    m = 1.0; c = 5.0; k = 120.0; L = 0.2
    F = zeros(2)
```

```
    F[0] = y[1]
    F[1] = c/m/y[0]**2 - k/m*(y[0] - L)
    return F

x = 0.0
y = array([0.2, 0.0])
xStop = 0.6
h = 0.1
X,Y = integrate(F,x,y,xStop,h)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

The printout below shows the two points that span the instant when the mass returns to the starting position $x = 0.2$ m for the first time.

```
       x           y[ 0 ]        y[ 1 ]
   3.7522e-01   2.0062e-01   -3.9224e-01
   3.8074e-01   2.0035e-01    2.9612e-01
```

Letting $t = 0.375\,22 + \Delta t$ be the time when $\dot{x} = 0$, we obtain from Taylor series

$$\dot{x}(0.375\,22 + \Delta t) = \dot{x}(0.375\,22) + \ddot{x}(0.375\,22)\,\Delta t = 0 \tag{a}$$

But

$$\ddot{x}(0.375\,22) = \frac{5}{1}\frac{1}{0.200\,62^2} - \frac{120}{1}(0.2 - 0.200\,62) = 124.30 \text{ m/s}^2$$

so the Eq. (a) is

$$0 = -0.392\ 24 + 124.30\Delta t \qquad \Delta t = 0.003\ 156 \text{ s}$$

Therefore, the period is $\tau = 0.375\,22 + 0.003\ 156 = 0.3784$ s ◀

Problem 17

$$\ddot{\theta} = \dot{\phi}^2 \sin\theta\cos\theta \qquad \ddot{\phi} = -2\dot{\theta}\dot{\phi}\cot\theta$$

$$\theta(0) = \frac{\pi}{12} \qquad \dot{\theta}(0) = 0 \qquad \phi(0) = 0 \qquad \dot{\phi} = 20 \text{ rad/s}$$

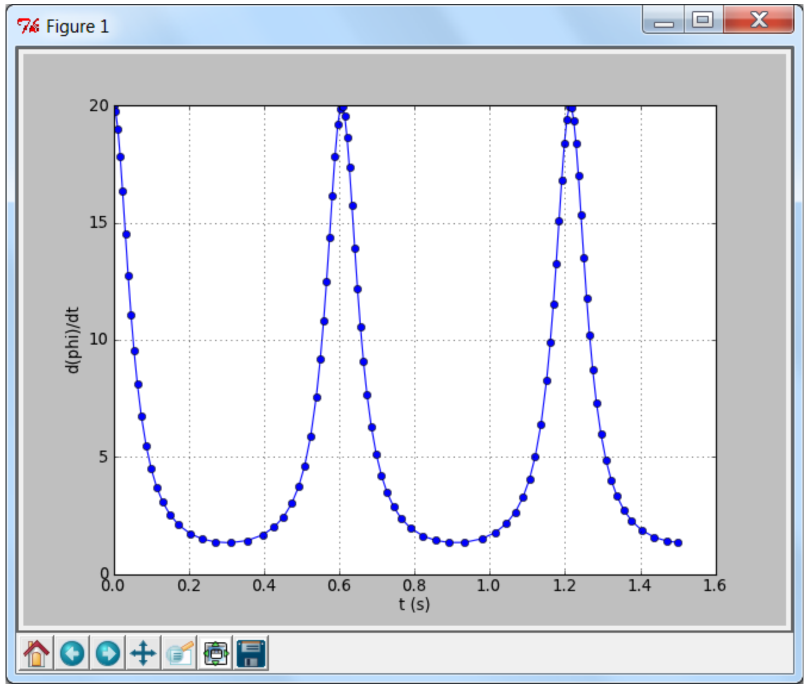
With the notation

$$\mathbf{y} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \\ \phi \\ \dot{\phi} \end{bmatrix}$$

the equivalent first-order differential equations are

$$\mathbf{F} = \begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_3^2 \sin y_0 \cos y_0 \\ y_3 \\ -2y_1 y_3 \cot y_0 \end{bmatrix}$$

```
## problem7_2_17
from numpy import zeros,array
from run_kut5 import *
import matplotlib.pyplot as plt

from math import pi,sin,cos

def F(x,y):
    F = zeros(4)
    F[0] = y[1]
    F[1] = (y[3]**2)*sin(y[0])*cos(y[0])
    F[2] = y[3]
    F[3] =-2.0*y[1]*y[3]*cos(y[0])/sin(y[0])
    return F

x = 0.0
y = array([pi/12.0, 0.0, 0.0, 20.0])
xStop = 1.5
h = 0.1
X,Y = integrate(F,x,y,xStop,h)
plt.plot(X,Y[:,3],'-o')
plt.xlabel('t (s)'); plt.ylabel('d(phi)/dt')
plt.grid(True); plt.show()
input("\nPress return to exit")
```



Problem 18

The equations in Example 7.11 were

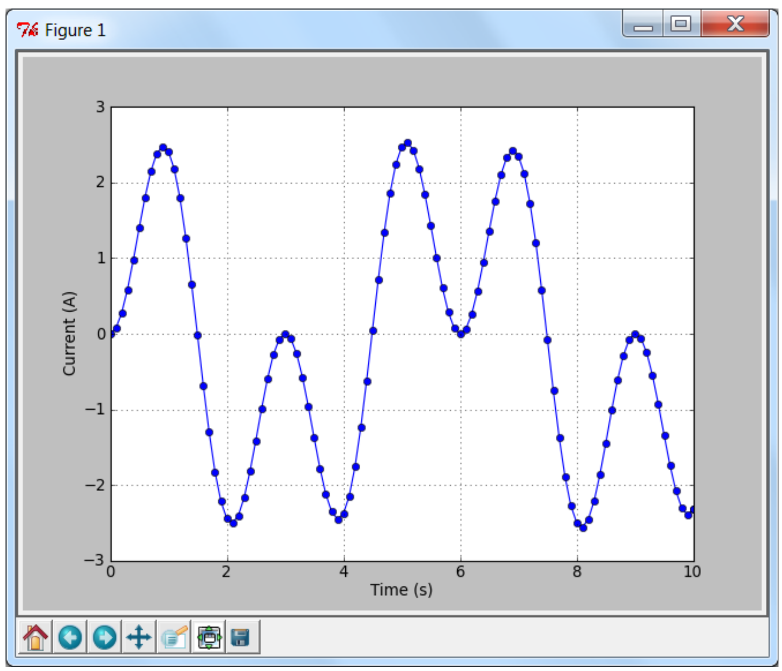
$$\frac{di}{dt} = \left[-Ri - \frac{q}{C} + E(t)\right]\frac{1}{L} \qquad \frac{dq}{dt} = i \qquad i(0) = q(0) = 0$$

Substituting $y_0 = q$, $y_1 = i$, $R = 0$, $C = 0.45$, $L = 2$ and $E(t) = 9\sin\pi t$ V, the differential equations become

$$\mathbf{F} = \begin{bmatrix} dq/dt \\ di/dt \end{bmatrix} = \begin{bmatrix} y_1 \\ \left[-y_0/0.45 + 9\sin\pi x\right]/2 \end{bmatrix}$$

```
#!/usr/bin/python
## problem7_2_18
from bulStoer import *
from math import sin,pi
import numpy as np
import matplotlib.pyplot as plt

def F(x,y):
    F = np.zeros(2)
    F[0] = y[1]
    F[1] = (-y[0]/0.45 + 9.0*sin(pi*x))/2.0
```

```
    return F

H = 0.1
xStop = 10.0
x = 0.0
y = np.array([0.0, 0.0])
X,Y = bulStoer(F,x,y,xStop,H)
plt.plot(X,Y[:,1],'o-')
plt.xlabel('Time (s)')
plt.ylabel('Current (A)')
plt.grid(True)
plt.show()
input("\nPress return to exit")
```

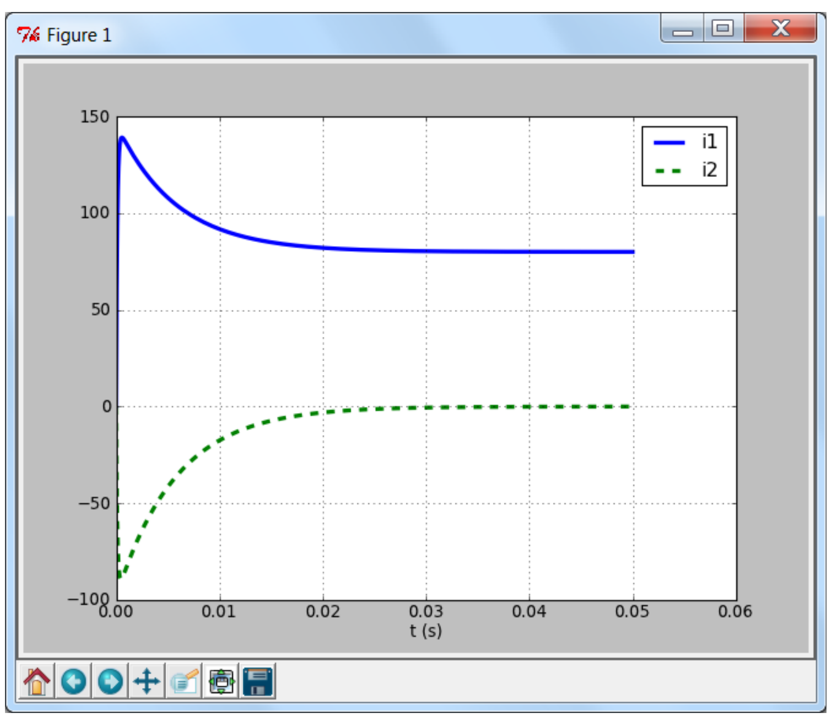


Problem 19

With constant E the equations in Problem 21, Problem Set 1 become

$$\begin{aligned}
\frac{di_1}{dt} &= (-3Ri_1 - 2Ri_2 + E)\frac{1}{L} \\
\frac{di_2}{dt} &= \left(-2\frac{di_1}{dt} - \frac{i_2}{RC}\right)\frac{1}{3} \\
i_1(0) &= i_2(0) = 0
\end{aligned}$$

```
## problem7_2_19
from numpy import zeros,array
from run_kut5 import *
import matplotlib.pyplot as plt

def F(x,y):
    R = 1.0; L = 0.2e-3; C = 3.5e-3; E = 240.0
    F = zeros(2)
    F[0] = (-3.0*R*y[0] - 2.0*R*y[1] + E)/L
    F[1] = (-2.0*F[0] - y[1]/R/C)/3.0
    return F

x = 0.0
y = array([0.0, 0.0])
xStop = 0.05
h = 0.001
X,Y = integrate(F,x,y,xStop,h)
plt.plot(X,Y[:,0],'-',X,Y[:,1],'--',linewidth=3)
plt.xlabel('t (s)')
plt.legend(('i1','i2'),loc=0)
plt.grid(True); plt.show()
input("\nPress return to exit")
```



Problem 20

$$
\begin{aligned}
L\frac{di_1}{dt} + R_1 i_1 + R_2(i_1 - i_2) &= E(t) \\
L\frac{di_2}{dt} + R_2(i_2 - i_1) + \frac{q_2}{C} &= 0 \\
\frac{dq_2}{dt} &= i_2
\end{aligned}
$$

$$q_2(0) = i_1(0) = i_2(0) = 0$$

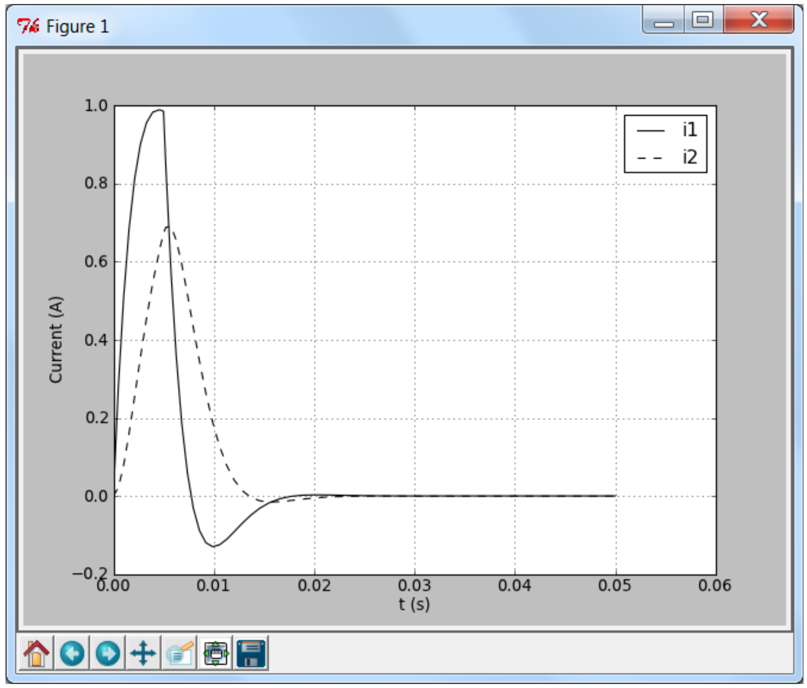
Using the notation

$$\mathbf{y} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} q_2 \\ i_1 \\ i_2 \end{bmatrix}$$

the equivalent first-order differential equations become

$$\mathbf{F} = \begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} y_2 \\ \left[-(R_1 + R_2)y_1 - R_2 y_2 + E\right]/L \\ \left[-y_0/C - R_2(y_2 - y_1)\right]/L \end{bmatrix}$$

```
## problem7_1_20
from numpy import zeros,array
from run_kut5 import *
import matplotlib.pyplot as plt

def F(x,y):
    R1 = 4.0; R2 = 10.0; L = 0.032; C = 0.53
    if x < 0.005: E = 20.0
    else: E = 0.0
    F = zeros(3)
    F[0] = y[2]
    F[1] = (-(R1 + R2)*y[1] - R2*y[2] + E)/L
    F[2] = (-y[0]/C - R2*(y[2] -y [1]))/L
    return F

x = 0.0
y = array([0.0, 0.0, 0.0])
xStop = 0.05
h = 0.001
X,Y = integrate(F,x,y,xStop,h)
plt.plot(X,Y[:,1],'-o',X,Y[:,2],'-^')
plt.xlabel('t (s)'); plt.ylabel('Current (A)')
plt.legend(('i1','i2'),loc=0)
plt.grid(True); plt.show()
input("\nPress return to exit")
```

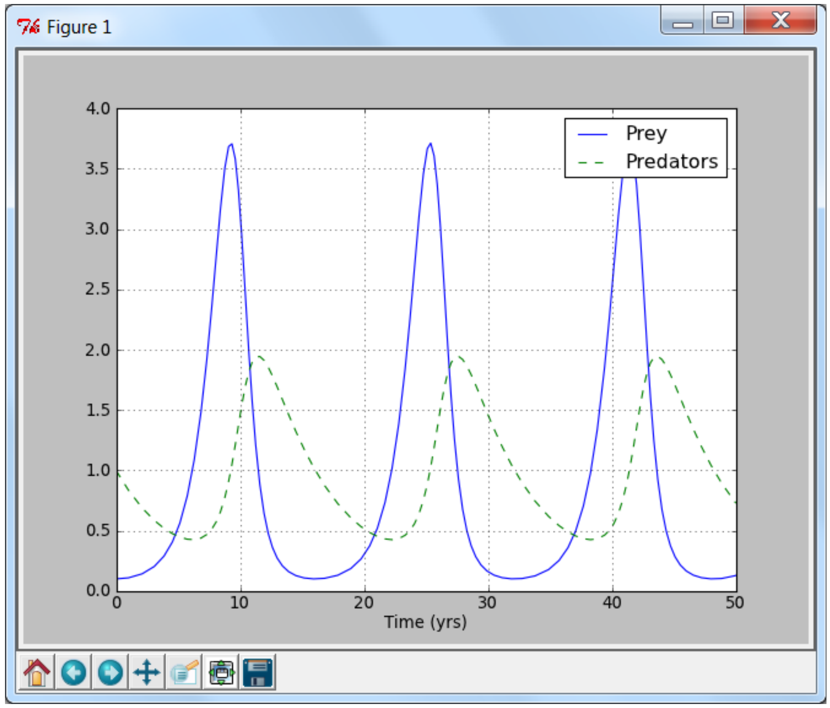


Problem 21

$$\dot{y}_0 = 1.0(y_0 - y_0 y_1) \qquad \dot{y}_1 = 0.2(-y_1 + y_0 y_1)$$
$$y_0(0) = 0.1 \qquad y_1(0) = 1$$

```
## problem7_2_21
from numpy import zeros,array
from run_kut5 import *
import matplotlib.pyplot as plt

def F(x,y):
    F = zeros(2)
    F[0] = y[0] - y[0]*y[1]
    F[1] = 0.2*(-y[1] + y[0]*y[1])
    return F

x = 0.0
y = array([0.1, 1.0])
xStop = 50.0
h = 1.0
X,Y = integrate(F,x,y,xStop,h)
plt.plot(X,Y[:,0],'-',X,Y[:,1],'--')
```

```
plt.xlabel('Time (yrs)')
plt.legend(('Prey','Predators'),loc=0)
plt.grid(True); plt.show()
input("\nPress return to exit")
```



Problem 22

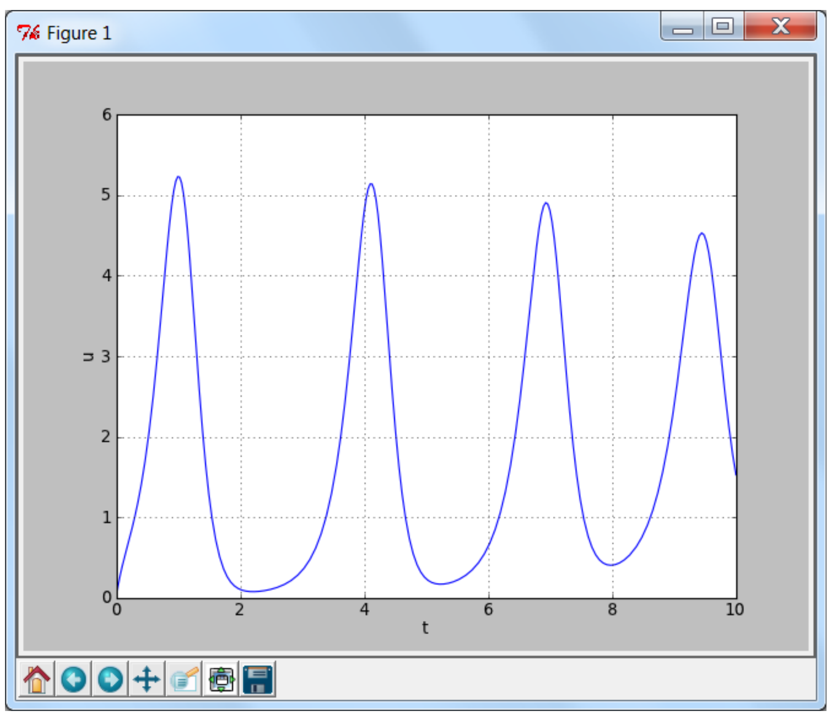
$$\dot{u} = -au + av \qquad \dot{v} = cu - v - uw \qquad \dot{w} = -bw + uv$$

$$u(0) = 0 \qquad v(0) = 1 \qquad w(0) = 2$$

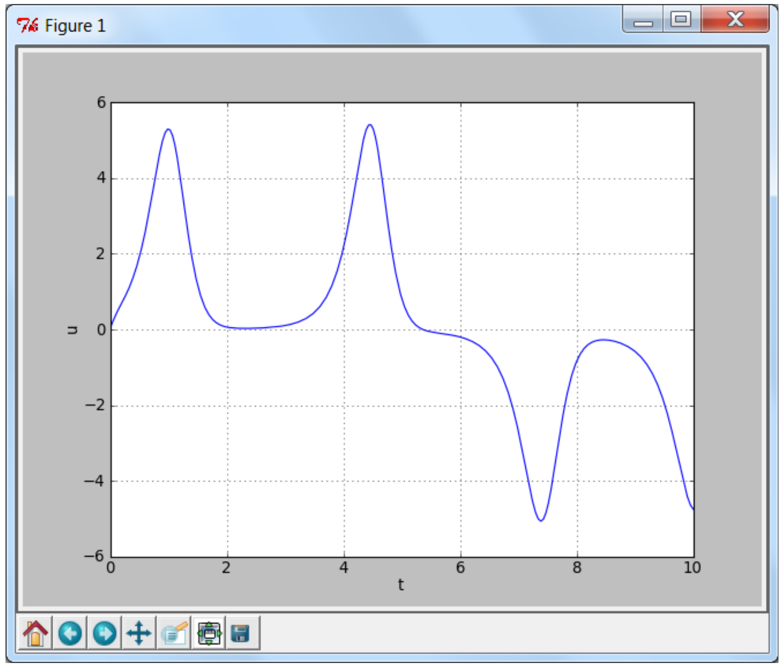
We use the notation $u = y_0$, $v = y_1$ and $w = y_2$.
With $c = 8.2$:

```
## problem7_2_22
from numpy import zeros,array
from run_kut5 import *
import matplotlib.pyplot as plt

def F(x,y):
    a = 5.0; b = 0.9; c = 8.2
    F = zeros(3)
    F[0] = -a*y[0] + a*y[1]
    F[1] = c*y[0] - y[1] - y[0]*y[2]
```

```
    F[2] = -b*y[2] + y[0]*y[1]
    return F

x = 0.0
y = array([0.0,1.0,2.0])
xStop = 10.0
h = 0.1
X,Y = integrate(F,x,y,xStop,h)
plt.plot(X,Y[:,0],'-')
plt.xlabel('t'); plt.ylabel('u')
plt.grid(True); plt.show()
raw_input("\nPress return to exit")
```



With $c = 8.3$:



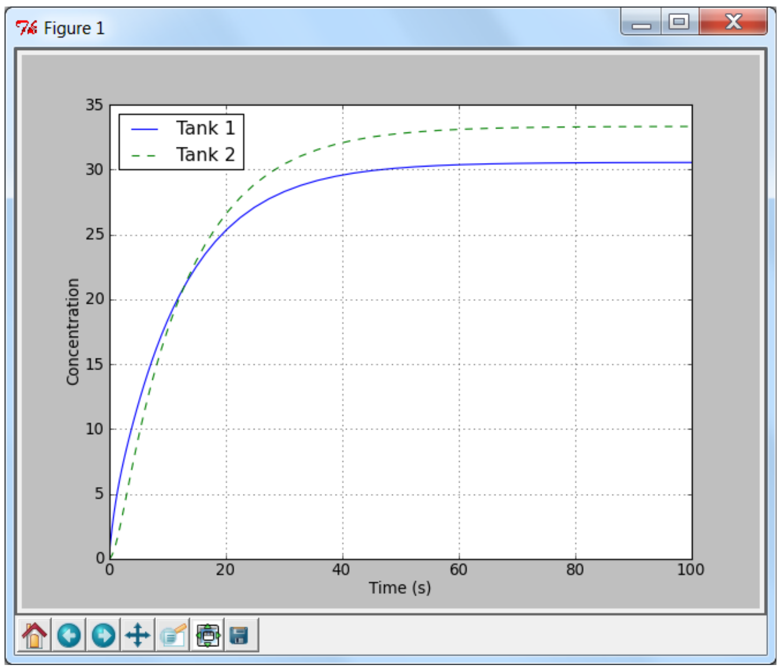
The solution seems to be very sensitive to the values of the parameters.

Problem 23

```
## problem7_2_23
from numpy import zeros
from run_kut5 import *
import matplotlib.pyplot as plt
from printSoln import *

def F(t,c):
    F = zeros(4)
    F[0] = -0.6*c[0] + 0.4*c[1] + 5.0
    F[1] = -0.7*c[1] + 0.3*c[2] + 0.4*c[3]
    F[2] = 0.4*c[0] - 0.4*c[2]
    F[3] = 0.2*c[0] + 0.1*c[2] - 0.4*c[3] + 5.0
    return F

t = 0.0
c = zeros(4)
tStop = 100.0
h = 1.0
T,C = integrate(F,t,c,tStop,h)
```

```
printSoln(T,C,0)
plt.plot(T,C[:,0],'-',T,C[:,1],'--')
plt.xlabel('Time (s)'); plt.ylabel('Concentration')
plt.legend(('Tank 1','Tank 2'),loc=0)
plt.grid(True); plt.show()
input("\nPress return to exit")
```

```
    x          y[ 0 ]        y[ 1 ]        y[ 2 ]        y[ 3 ]
  0.0000e+00    0.0000e+00    0.0000e+00    0.0000e+00    0.0000e+00
  1.0000e+02    3.0549e+01    3.3325e+01    3.0548e+01    3.5410e+01
```



In Prob. 21, Problem Set 2.2 the steady-state values were found to be

$$\begin{aligned} c_1 &= 30.556 \text{ mg/m}^3 \qquad c_2 = 33.333 \text{ mg/m}^3 \\ c_3 &= 30.556 \text{ mg/m}^3 \qquad c_4 = 35.417 \text{ mg/m}^3 \end{aligned}$$

Comparing these values with the printout, we conclude that the system is very close to the steady state after 100 s.

PROBLEM SET 8.1

Problem 1

$$y'' + y' - y = 0$$

We know the two solutions

$$\begin{aligned} y(0) &= 0 \qquad y'(0) = 1 \rightarrow y(1) = 0.741\,028 \\ y(0) &= 0 \qquad y'(0) = 0 \rightarrow y(1) = 0 \end{aligned}$$

We are looking for u so that

$$y(0) = 0 \qquad y'(0) = u \rightarrow y(1) = 1$$

By linear interpolation

$$u = \frac{1}{0.741\,028} = 1.349\,477 \quad \blacktriangleleft$$

Problem 2

$$y''' + y'' + 2y' = 6$$

The two known solutions are

$$\begin{aligned} y(0) &= 2 \qquad y'(0) = 0 \qquad y''(0) = 1 \rightarrow y(1) = 3.037\,65 \\ y(0) &= 2 \qquad y'(0) = 0 \qquad y''(0) = 0 \rightarrow y(1) = 2.723\,18 \end{aligned}$$

We have to find u so that

$$y(0) = 2 \qquad y'(0) = 0 \qquad y''(0) = u \rightarrow y(1) = 0$$

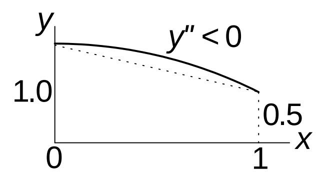
Linear interpolation yields

$$u = -\frac{2.723\,18}{3.037\,65 - 2.723\,18} = -8.659\,59 \quad \blacktriangleleft$$

Problem 3

(a)

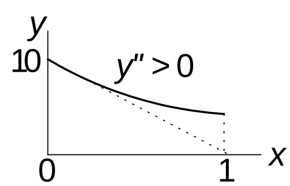
$$y'' = -e^{-y} \qquad y(0) = 1 \qquad y(1) = 0.5$$



Apparently $y'(0) < 0.5$; hence we estimate $y'(0) \approx 0.25$ ◀

(b)

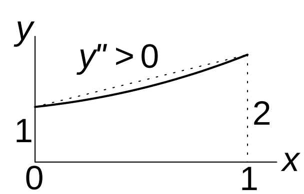
$$y'' = 4y^2 \qquad y(0) = 10 \qquad y'(1) = 0$$



We estimate $y'(0) \approx -10$ ◀

(c)

$$y'' = \cos(xy) \qquad y(0) = 1 \qquad y(1) = 2$$

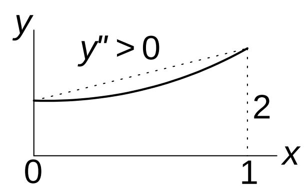


We see that $y'(0) < 1$, so that a reasonable quess is $y'(0) \approx 0.5$ ◀

Problem 4

(a)

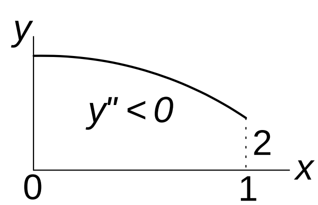
$$y'' = y^2 + xy \qquad y'(0) = 0 \qquad y(1) = 2$$



Estimate $y(0) \approx 1$ ◀

(b)

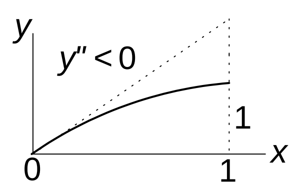
$$y'' = -\frac{2}{x}y' - y^2 \qquad y'(0) = 0 \qquad y(1) = 2$$



Estimate $y(0) \approx 4$ ◀

(c)

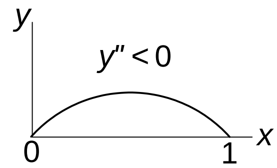
$$y'' = -x(y')^2 \qquad y'(0) = 2 \qquad y(1) = 1$$



Estimate $y(0) = 0$ ◀

Problem 5

$$y''' + 5y''y^2 = 0 \qquad y(0) = 0 \qquad y'(0) = 1 \qquad y(1) = 0$$



Assume that y is parabolic:

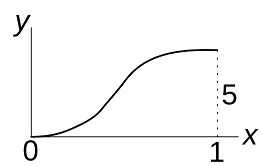
$$y = Cx(1-x) \qquad y' = C(1-2x)$$

But $y'(0) = 1$. Therefore, $C = 1$ and $y'' = -2$. Thus the estimate is

$$y''(0) \approx -2 \blacktriangleleft$$

Problem 6

$$y^{(4)} + 2y'' + y' \sin y = 0$$
$$y(0) = y'(0) = 0 \qquad y(1) = 5 \qquad y'(1) = 0$$



Assume that y is cubic:

$$\begin{aligned} y &= C_1x^2 + C_2x^3 \qquad y' = 2C_1x + 3C_2x^2 \\ y'' &= 2C_1 + 6C_2x \qquad y''' = 6C_2 \end{aligned}$$

The conditions $y(1) = 5$ and $y'(1) = 0$ yield

$$\begin{aligned} C_1 + C_2 &= 5 \\ 2C_1 + 3C_2 &= 0 \end{aligned}$$

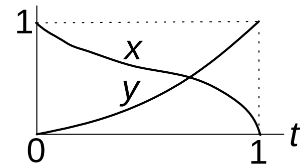
the solution of which is $C_1 = 15$, $C_2 = -10$. Thus the estimates are

$$\begin{aligned} y''(0) &= 2C_1 = 30 \blacktriangleleft \\ y'''(0) &= 6C_2 = -60 \blacktriangleleft \end{aligned}$$

Problem 7

$$\ddot{x} + 2x^2 - y = 0 \qquad \ddot{y} + y^2 - 2x = 1$$

$$x(0) = 1 \qquad x(1) = 0 \qquad y(0) = 0 \qquad y(1) = 1$$



We can obtain the following information about the curvatures from the differential equations and the boundary conditions:

$$\begin{aligned} \text{At } t &= 0\text{:} \qquad \ddot{x} = 1 \qquad \ddot{y} = 0 \\ \text{At } t &= 1\text{:} \qquad \ddot{x} = -2 \qquad \ddot{y} = 3 \end{aligned}$$

This information was used to draw the rough sketches of x and y. From these sketches we estimate that

$$\dot{x}(0) \approx -1 \blacktriangleleft \qquad \dot{y}(0) \approx 0.5 \blacktriangleleft$$

Problem 8

$$y'' = -(1 - 0.2x)\, y^2 \qquad y(0) = 0 \qquad y(\pi/2) = 1$$

```
## problem8_1_8
from numpy import zeros,array
from run_kut5 import *
from ridder import *
from printSoln import *
from math import pi

def initCond(u):  # Initial values of [y,y'];
                  # use 'u' if unknown
    return array([0.0, u])

def r(u):  # Boundary condition residuals
    X,Y = integrate(F,x,initCond(u),xStop,h)
    y = Y[len(Y) - 1]
    r = y[0] - 1.0
    return r

def F(x,y):  # First-order differential equations
    F = zeros(2)
    F[0] = y[1]
    F[1] = -(1.0 - 0.2*x)*y[0]**2
```

```
    return F

x = 0.0                      # Start of integration
xStop = pi/2.0               # End of integration
u1 = 0.6; u2 = 1.2           # Initial guesses for u
h = 0.1                      # Initial step size
freq = 1                     # Printout frequency
u = ridder(r,u1,u2,1.0e-5)
X,Y = integrate(F,x,initCond(u),xStop,h)
printSoln(X,Y,freq)
input("\nPress return to exit")
```

```
        x           y[ 0 ]         y[ 1 ]
   0.0000e+00    0.0000e+00    7.7880e-01
   1.0000e-01    7.7875e-02    7.7860e-01
   4.4835e-01    3.4725e-01    7.6190e-01
   7.1870e-01    5.4756e-01    7.1328e-01
   9.8976e-01    7.2940e-01    6.2073e-01
   1.2392e+00    8.6941e-01    4.9596e-01
   1.4802e+00    9.7139e-01    3.4636e-01
   1.5708e+00    1.0000e+00    2.8516e-01
```

Problem 9

$$y'' = -2y' - 3y^2 \qquad y(0) = 0 \qquad y(2) = -1$$

```
## problem8_1_9
from numpy import zeros,array
from run_kut5 import *
from ridder import *
from printSoln import *

def initCond(u):  # Initial values of [y,y'];
                  # use 'u' if unknown
    return array([0.0, u])

def r(u):  # Boundary condition residuals
    X,Y = integrate(F,x,initCond(u),xStop,h)
    y = Y[len(Y) - 1]
    r = y[0] + 1.0
    return r
```

```
def F(x,y):  # First-order differential equations
    F = zeros(2)
    F[0] = y[1]
    F[1] = -2.0*y[1] - 3.0*y[0]**2
    return F

x = 0.0                     # Start of integration
xStop = 2.0                 # End of integration
u1 = -1.5; u2 = -0.5        # Initial guesses for u
h = 0.1                     # Initial step size
freq = 1                    # Printout frequency
u = ridder(r,u1,u2,1.0e-5)
X,Y = integrate(F,x,initCond(u),xStop,h)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

```
       x          y[ 0 ]        y[ 1 ]
   0.0000e+00    0.0000e+00   -9.9420e-01
   1.0000e-01   -9.0130e-02   -8.1480e-01
   2.1272e-01   -1.7262e-01   -6.5598e-01
   3.2213e-01   -2.3773e-01   -5.3984e-01
   4.3180e-01   -2.9200e-01   -4.5460e-01
   5.4341e-01   -3.3913e-01   -3.9386e-01
   6.5808e-01   -3.8178e-01   -3.5340e-01
   7.7668e-01   -4.2215e-01   -3.3025e-01
   8.9993e-01   -4.6222e-01   -3.2249e-01
   1.0284e+00   -5.0393e-01   -3.2905e-01
   1.1626e+00   -5.4932e-01   -3.4986e-01
   1.3024e+00   -6.0060e-01   -3.8590e-01
   1.4475e+00   -6.6026e-01   -4.3934e-01
   1.5962e+00   -7.3086e-01   -5.1363e-01
   1.7457e+00   -8.1471e-01   -6.1311e-01
   1.8920e+00   -9.1343e-01   -7.4254e-01
   2.0000e+00   -1.0000e+00   -8.6542e-01
```

Problem 10

$$y'' = -\sin y - 1 \qquad y(0) = y(\pi) = 0$$

```
## problem8_1_10
from numpy import zeros,array
from bulStoer import *
```

```
from ridder import *
from printSoln import *
from math import sin,pi

def initCond(u):  # Initial values of [y,y'];
                  # use 'u' if unknown
    return array([0.0, u])

def r(u):  # Boundary condition residuals
    X,Y = bulStoer(F,x,initCond(u),xStop,H)
    y = Y[len(Y) - 1]
    r = y[0]
    return r

def F(x,y):  # First-order differential equations
    F = zeros(2)
    F[0] = y[1]
    F[1] = -sin(y[0]) - 1.0
    return F

x = 0.0                    # Start of integration
xStop = pi                 # End of integration
u1 = 2.0; u2 = 3.0         # Initial guesses for u
H = 0.5                    # Step size
freq = 1                   # Printout frequency
u = ridder(r,u1,u2,1.0e-5)
X,Y = bulStoer(F,x,initCond(u),xStop,H)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

x	y[0]	y[1]
0.0000e+00	0.0000e+00	2.8047e+00
5.0000e-01	1.2266e+00	2.0219e+00
1.0000e+00	1.9900e+00	1.0356e+00
1.5000e+00	2.2767e+00	1.2454e-01
2.0000e+00	2.1176e+00	-7.6891e-01
2.5000e+00	1.4931e+00	-1.7422e+00
3.0000e+00	3.8580e-01	-2.6359e+00
3.1416e+00	7.8855e-09	-2.8047e+00

Problem 11

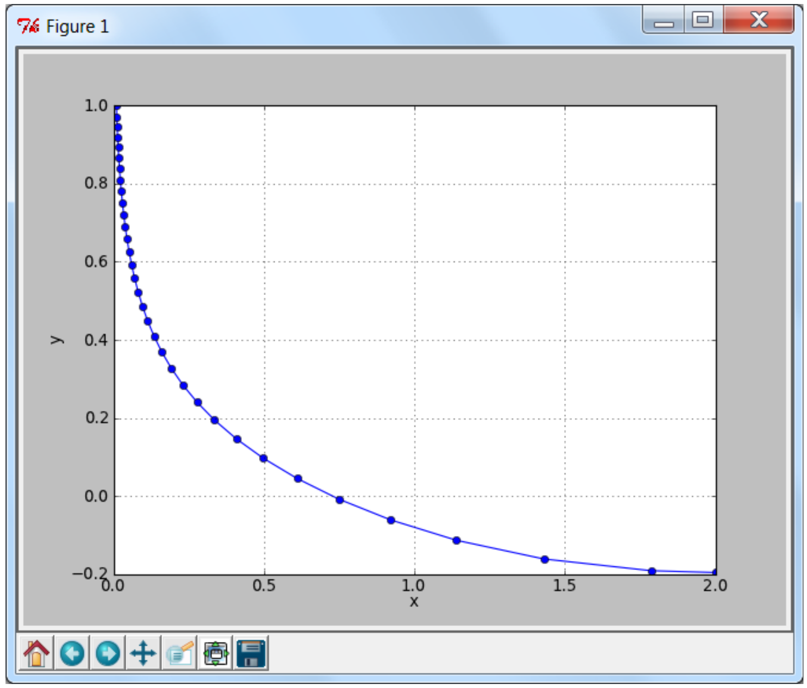
$$y'' = -\frac{1}{x}y' - y \qquad y(0.01) = 1 \qquad y'(2) = 0$$

We chose the adaptive Runge-Kutta method for integration.

```
## problem8_1_11
from numpy import zeros,array
from run_kut5 import *
from linInterp import *
from printSoln import *
import matplotlib.pyplot as plt

def initCond(u):  # Initial values of [y,y'];
                  # use 'u' if unknown
    return array([1.0, u])

def r(u):  # Boundary condition residuals
    X,Y = integrate(F,x,initCond(u),xStop,h)
    y = Y[len(Y) - 1]
    r = y[1]
    return r

def F(x,y):  # First-order differential equations
\qquad \qquad F = zeros(2)
    F[0] = y[1]
    F[1] = -y[1]/x - y[0]
    return F

x = 0.01                      # Start of integration
xStop = 2.0                   # End of integration
u1 = -50.0; u2 = 0.0          # Initial guesses for u
h = 0.1                       # Initial step size
freq = 0                      # Printout frequency
u = linInterp(r,u1,u2)
X,Y = integrate(F,x,initCond(u),xStop,h)
printSoln(X,Y,freq)
plt.plot(X,Y[:,0],'-o')
plt.xlabel('x'); plt.ylabel('y')
plt.grid(True); plt.show()
input("\nPress return to exit")
```

```
        x         y[ 0 ]        y[ 1 ]
   1.0000e-02    1.0000e+00   -2.2582e+01
```

```
2.0000e+00    -1.9573e-01    -2.4833e-07
```



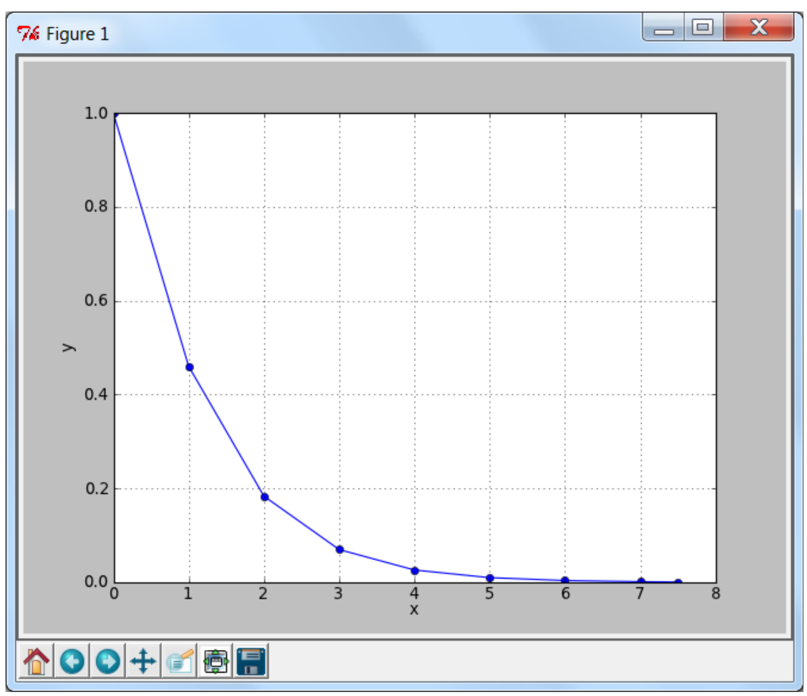
Problem 12

$$y'' = (1 - e^{-x})y \qquad y(0) = 1 \qquad y(\infty) = 0$$

```
## problem8_1_12
from numpy import zeros,array
from bulStoer import *
from linInterp import *
from printSoln import *
from math import exp
import matplotlib.pyplot as plt


def initCond(u):  # Initial values of [y,y'];
                  # use 'u' if unknown
    return array([1.0, u])

def r(u):  # Boundary condition residuals
    X,Y = bulStoer(F,x,initCond(u),xStop,H)
    y = Y[len(Y) - 1]
    r = y[0]
```

```
    return r

def F(x,y):  # First-order differential equations
    F = zeros(2)
    F[0] = y[1]
    F[1] = (1.0 - exp(-x))*y[0]
    return F

x = 0.0                         # Start of integration
xStop = 7.5                     # End of integration
u1 = -1.0; u2 = 0.0             # Initial guesses for u
H = 1.0                         # Initial step size
freq = 0                        # Printout frequency
u = linInterp(r,u1,u2)
X,Y = bulStoer(F,x,initCond(u),xStop,H)
printSoln(X,Y,freq)
plt.plot(X,Y[:,0],'-o')
plt.xlabel('x'); plt.ylabel('y')
plt.grid(True); plt.show()
input("\nPress return to exit")
```

```
        x            y[ 0 ]         y[ 1 ]
   0.0000e+00     1.0000e+00    -6.3455e-01
   7.5000e+00     1.0289e-04    -1.4638e-03
```



The results did not change significantly when the program was run with `xStop`

= 5.0.

Problem 13

$$
\begin{aligned}
y''' &= -\frac{1}{x}y'' + \frac{1}{x^2}y' + 0.1(y')^3 \\
y(1) &= 0 \qquad y''(1) = 0 \qquad y(2) = 1
\end{aligned}
$$

We chose the adaptive Runge-Kutta method for integration.

```
## problem8_1_13
from numpy import zeros,array
from run_kut5 import *
from ridder import *
from printSoln import *

def initCond(u):  # Initial values of [y,y',y''];
                  # use 'u' if unknown
    return array([0.0, u, 0.0])

def r(u):  # Boundary condition residual--see Eq. (8.3)
    X,Y = integrate(F,xStart,initCond(u),xStop,h)
    y = Y[len(Y) - 1]
    r = y[0] - 1.0
    return r

def F(x,y):  # First-order differential equations
    F = zeros(3)
    F[0] = y[1]
    F[1] = y[2]
    F[2] = -y[2]/x + y[1]/x**2 + 0.1*y[1]**3
    return F

xStart = 1.0        # Start of integration
xStop = 2.0         # End of integration
u1 = 0.0; u2 = 2.0  # Trial values
h = 0.05            # Initial step size
freq = 2            # Printout frequency
u = ridder(r,u1,u2)
X,Y = integrate(F,xStart,initCond(u),xStop,h)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

x	y[0]	y[1]	y[2]
1.0000e+00	0.0000e+00	9.0112e-01	0.0000e+00
1.2025e+00	1.8369e-01	9.1792e-01	1.5291e-01
1.5582e+00	5.2358e-01	1.0020e+00	3.0430e-01
2.0000e+00	1.0000e+00	1.1635e+00	4.2129e-01

Problem 14

$$y''' = -4y'' - 6y' + 10$$
$$y(0) = y''(0) = 0 \qquad y(3) - y'(3) = 5$$

```
## problem8_1_14
from numpy import zeros,array
from run_kut5 import *
from linInterp import *
from printSoln import *

def initCond(u):  # Initial values of [y,y',y''];
                  # use 'u' if unknown
    return array([0.0, u, 0.0])

def r(u):  # Boundary condition residual--see Eq. (8.3)
    X,Y = integrate(F,xStart,initCond(u),xStop,h)
    y = Y[len(Y) - 1]
    r = y[0] - y[1] - 5.0
    return r

def F(x,y):  # First-order differential equations
    F = zeros(3)
    F[0] = y[1]
    F[1] = y[2]
    F[2] = -4.0*y[2] - 6.0*y[1] + 10.0
    return F

xStart = 0.0        # Start of integration
xStop = 3.0         # End of integration
u1 = 0.0; u2 = 2.0  # Trial values
h = 0.05            # Initial step size
freq = 2            # Printout frequency
u = linInterp(r,u1,u2)
X,Y = integrate(F,xStart,initCond(u),xStop,h)
```

```
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

```
       x          y[ 0 ]        y[ 1 ]        y[ 2 ]
   0.0000e+00    0.0000e+00    4.1461e+00    0.0000e+00
   1.3493e-01    5.5413e-01    4.0331e+00   -1.5233e+00
   3.0370e-01    1.2079e+00    3.6902e+00   -2.3863e+00
   4.7036e-01    1.7885e+00    3.2730e+00   -2.5344e+00
   6.3850e-01    2.3037e+00    2.8627e+00   -2.3034e+00
   8.1138e-01    2.7661e+00    2.4986e+00   -1.8928e+00
   9.9196e-01    3.1890e+00    2.1991e+00   -1.4264e+00
   1.1834e+00    3.5866e+00    1.9700e+00   -9.8142e-01
   1.3895e+00    3.9748e+00    1.8086e+00   -6.0314e-01
   1.6160e+00    4.3717e+00    1.7071e+00   -3.1367e-01
   1.8721e+00    4.8010e+00    1.6544e+00   -1.1801e-01
   2.1754e+00    5.2994e+00    1.6377e+00   -8.8302e-03
   2.5579e+00    5.9266e+00    1.6438e+00    2.8919e-02
   2.9652e+00    6.5985e+00    1.6553e+00    2.4274e-02
   3.0000e+00    6.6561e+00    1.6561e+00    2.3251e-02
```

Problem 15

$$
\begin{aligned}
y''' &= -2y'' - \sin y \\
y(-1) &= 0 \qquad y'(-1) = -1 \qquad y'(1) = 1
\end{aligned}
$$

```
## problem8_1_15
from numpy import zeros,array
from run_kut5 import *
from ridder import *
from printSoln import *
from math import sin

def initCond(u):  # Initial values of [y,y',y''];
                  # use 'u' if unknown
    return array([0.0, -1.0, u])

def r(u):  # Boundary condition residual--see Eq. (8.3)
    X,Y = integrate(F,xStart,initCond(u),xStop,h)
    y = Y[len(Y) - 1]
    r = y[1] - 1.0
    return r
```

```
def F(x,y):  # First-order differential equations
    F = zeros(3)
    F[0] = y[1]
    F[1] = y[2]
    F[2] =-2.0*y[2] - sin(y[0])
    return F

xStart = -1.0       # Start of integration
xStop = 1.0         # End of integration
u1 = 2.0; u2 = 6.0  # Trial values
h = 0.05            # Initial step size
freq = 2            # Printout frequency
u = ridder(r,u1,u2)
X,Y = integrate(F,xStart,initCond(u),xStop,h)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

```
        x           y[ 0 ]         y[ 1 ]         y[ 2 ]
  -1.0000e+00    0.0000e+00   -1.0000e+00    4.3791e+00
  -8.6166e-01   -1.0003e-01   -4.7044e-01    3.3278e+00
  -6.7867e-01   -1.3651e-01    4.1312e-02    2.3272e+00
  -4.8073e-01   -8.8056e-02    4.2389e-01    1.5855e+00
  -2.6549e-01    3.5252e-02    7.0222e-01    1.0356e+00
  -2.9648e-02    2.2546e-01    8.9462e-01    6.2107e-01
   2.3136e-01    4.7614e-01    1.0121e+00    2.9721e-01
   5.2346e-01    7.8042e-01    1.0585e+00    3.3759e-02
   8.5216e-01    1.1260e+00    1.0323e+00   -1.8063e-01
   1.0000e+00    1.2763e+00    1.0000e+00   -2.5381e-01
```

Problem 16

$$\begin{aligned} y''' &= -2y'' - \sin y \\ y(-1) &= 0 \qquad y(0) = 0 \qquad y(1) = 1 \end{aligned}$$

The Bulirsch-Stoer method for integration is a must here since it gives us control over the integration increment (we need the computed y at exactly $x = 0$).

```
## problem8_1_16
from numpy import zeros,array
from bulStoer import *
```

```
from newtonRaphson2 import *
from printSoln import *
from math import sin

def initCond(u):  # Initial values of [y,y',y''];
                  # use 'u' if unknown
    return array([0.0, u[0], u[1]])

def r(u):  # Boundary condition residual--see Eq. (8.6)
    r = zeros(len(u))
    X,Y = bulStoer(F,xStart,initCond(u),xStop,H)
    yMiddle = Y[(len(Y) - 1)/2]
    yEnd = Y[len(Y) - 1]
    r[0] = yMiddle[0]
    r[1] = yEnd[0] - 1.0
    return r

def F(x,y):  # First-order differential equations
    F = zeros(3)
    F[0] = y[1]
    F[1] = y[2]
    F[2] = -2.0*y[2] - sin(y[0])
    return F

xStart = -1.0          # Start of integration
xStop = 1.0            # End of integration
u = array([-0.5, 1.0]) # Trial values
H = 0.2                # Step size
freq = 1               #Printout frequency
u = newtonRaphson2(r,u,1.0e-4)
X,Y = bulStoer(F,xStart,initCond(u),xStop,H)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

x	y[0]	y[1]	y[2]
-1.0000e+00	0.0000e+00	-1.4947e+00	5.1960e+00
-8.0000e-01	-2.0751e-01	-6.3667e-01	3.5035e+00
-6.0000e-01	-2.7298e-01	-5.4985e-02	2.3895e+00
-4.0000e-01	-2.4164e-01	3.4358e-01	1.6446e+00
-2.0000e-01	-1.4375e-01	6.1838e-01	1.1342e+00
-5.5511e-17	-2.7984e-11	8.0705e-01	7.7181e-01
2.0000e-01	1.7493e-01	9.3326e-01	5.0237e-01
4.0000e-01	3.7014e-01	1.0119e+00	2.9162e-01
6.0000e-01	5.7715e-01	1.0525e+00	1.1952e-01
8.0000e-01	7.8903e-01	1.0615e+00	-2.4616e-02

```
   1.0000e+00    1.0000e+00    1.0442e+00   -1.4557e-01
```

Problem 17

$$y^{(4)} = -xy^2$$

$$y(0) = 5 \qquad y''(0) = 0 \qquad y'(1) = 0 \qquad y'''(1) = 2$$

```
## problem8_1_17
from numpy import zeros,array
from run_kut5 import *
from newtonRaphson2 import *
from printSoln import *

def initCond(u):  # Initial values of [y,y',y'',y'''];
                  # use 'u' if unknown
    return array([5.0, u[0], 0.0, u[1]])

def r(u):  # Boundary condition residuals-- see Eq. (8.7)
    r = zeros(len(u))
    X,Y = integrate(F,x,initCond(u),xStop,h)
    y = Y[len(Y) - 1]
    r[0] = y[1]
    r[1] = y[3] - 2.0
    return r

def F(x,y):  # First-order differential equations
    F = zeros(4)
    F[0] = y[1]
    F[1] = y[2]
    F[2] = y[3]
    F[3] = -x*y[0]**2
    return F

x = 0.0                      # Start of integration
xStop = 1.0                  # End of integration
u = array([-2.0, 6.0])       # Initial guesses for u
h = 0.1                      # Initial step size
freq = 1                     # Printout frequency
u = newtonRaphson2(r,u,1.0e-5)
X,Y = integrate(F,x,initCond(u),xStop,h)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

x	y[0]	y[1]	y[2]	y[3]
0.0000e+00	5.0000e+00	-3.2007e+00	0.0000e+00	7.6621e+00
1.0000e-01	4.6812e+00	-3.1625e+00	7.6231e-01	7.5475e+00
2.1540e-01	4.3233e+00	-3.0250e+00	1.6142e+00	7.1822e+00
3.3299e-01	3.9806e+00	-2.7867e+00	2.4278e+00	6.6300e+00
4.5188e-01	3.6683e+00	-2.4527e+00	3.1766e+00	5.9512e+00
5.7147e-01	3.3993e+00	-2.0321e+00	3.8434e+00	5.1908e+00
6.9121e-01	3.1850e+00	-1.5365e+00	4.4167e+00	4.3753e+00
8.1038e-01	3.0344e+00	-9.8113e-01	4.8873e+00	3.5141e+00
9.2814e-01	2.9536e+00	-3.8331e-01	5.2480e+00	2.6006e+00
1.0000e+00	2.9398e+00	-2.4420e-10	5.4136e+00	2.0000e+00

Problem 18

$$y^{(4)} = -2yy''$$

$$y(0) = y'(0) = 0 \qquad y(4) = 0 \qquad y'(4) = 1$$

```
## problem8_1_18
from numpy import zeros,array
from run_kut5 import *
from newtonRaphson2 import *
from printSoln import *

def initCond(u):  # Initial values of [y,y',y'',y'''];
                  # use 'u' if unknown
    return array([0.0, 0.0, u[0], u[1]])

def r(u):  # Boundary condition residuals-- see Eq. (8.7)
    r = zeros(len(u))
    X,Y = integrate(F,x,initCond(u),xStop,h)
    y = Y[len(Y) - 1]
    r[0] = y[0]
    r[1] = y[1] - 1.0
    return r

def F(x,y):  # First-order differential equations
    F = zeros(4)
    F[0] = y[1]
    F[1] = y[2]
    F[2] = y[3]
    F[3] = -2.0*y[0]*y[2]
```

```
    return F

x = 0.0                       # Start of integration
xStop = 4.0                   # End of integration
u = array([-0.3, 0.3])        # Initial guesses for u
h = 0.5                       # Initial step size
freq = 1                      # Printout frequency
u = newtonRaphson2(r,u,1.0e-5)
X,Y = integrate(F,x,initCond(u),xStop,h)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

```
        x          y[ 0 ]        y[ 1 ]        y[ 2 ]        y[ 3 ]
   0.0000e+00    0.0000e+00    0.0000e+00   -3.6774e-01    2.6832e-01
   3.3099e-01   -1.8523e-02   -1.0703e-01   -2.7903e-01    2.6706e-01
   6.5941e-01   -6.7150e-02   -1.8435e-01   -1.9218e-01    2.6100e-01
   1.0175e+00   -1.4351e-01   -2.3665e-01   -1.0055e-01    2.5056e-01
   1.4111e+00   -2.4193e-01   -2.5707e-01   -3.6883e-03    2.4335e-01
   1.8316e+00   -3.4732e-01   -2.3696e-01    1.0023e-01    2.5600e-01
   2.1886e+00   -4.2353e-01   -1.8416e-01    1.9796e-01    2.9730e-01
   2.5456e+00   -4.7427e-01   -9.3056e-02    3.1741e-01    3.7995e-01
   2.8780e+00   -4.8517e-01    3.5540e-02    4.6305e-01    5.0433e-01
   3.1921e+00   -4.4838e-01    2.0831e-01    6.4585e-01    6.6683e-01
   3.4429e+00   -3.7394e-01    3.9290e-01    8.3207e-01    8.1947e-01
   3.6660e+00   -2.6402e-01    6.0002e-01    1.0300e+00    9.5239e-01
   3.8896e+00   -1.0226e-01    8.5509e-01    1.2547e+00    1.0469e+00
   4.0000e+00    1.1125e-06    1.0000e+00    1.3714e+00    1.0619e+00
```

Problem 19

$$\ddot{x} = -\frac{c}{m} v\,\dot{x} \qquad \ddot{y} = -\frac{c}{m} v\,\dot{y} - g \qquad v = \sqrt{\dot{x}^2 + \dot{y}^2}$$

$$x(0) = y(0) = 0 \qquad x(10\text{ s}) = 8000\,\text{m} \qquad y(10\text{ s}) = 0$$

We use the notation

$$\mathbf{y} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}$$

```
## problem8_1_19
from numpy import zeros,array
from bulStoer import *
from newtonRaphson2 import *
```

```
from printSoln import *
from math import sqrt

def initCond(u):  # Initial values of [y,y',y'',y'''];
                  # use 'u' if unknown
    return array([0.0, u[0], 0.0, u[1]])

def r(u):  # Boundary condition residuals-- see Eq. (8.7)
    r = zeros(len(u))
    X,Y = bulStoer(F,x,initCond(u),xStop,H)
    y = Y[len(Y) - 1]
    r[0] = y[0] - 8000.0
    r[1] = y[2]
    return r

def F(x,y):  # First-order differential equations
    F = zeros(4)
    c = 3.2e-4; m = 20.0; g = 9.80665
    v = sqrt(y[1]**2 + y[3]**2)
    F[0] = y[1]
    F[1] = -c/m*v*y[1]
    F[2] = y[3]
    F[3] = -c/m*v*y[3] - g
    return F

x = 0.0                       # Start of integration
xStop = 10.0                  # End of integration
u = array([1000.0, 600.0])    # Initial guesses for u
H = 2.0                       # Step size
freq = 0                      # Printout frequency
u = newtonRaphson2(r,u,1.0e-5)
X,Y = bulStoer(F,x,initCond(u),xStop,H)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

x	y[0]	y[1]	y[2]	y[3]
0.0000e+00	0.0000e+00	8.5349e+02	0.0000e+00	5.0150e+01
1.0000e+01	8.0000e+03	7.5089e+02	-1.5760e-07	-4.8051e+01

$$v_0 = \sqrt{\dot{x}_0^2 + \dot{y}_0^2} = \sqrt{853.49^2 + 501.50^2} = 989.9 \text{ m/s} \blacktriangleleft$$

$$\theta = \tan^{-1}\frac{501.50}{853.49} = 0.531\,24 \text{ rad} = 30.44^\circ \blacktriangleleft$$

Problem 20

$$y^{(4)} = \beta y'' + 1 \qquad y(0) = y''(0) = y(1) = y''(1) = 0$$

where $()' = d()/d\xi$.

(a)

```
## problem8_1_20
from numpy import zeros,array
from bulStoer import *
from newtonRaphson2 import *
from printSoln import *

def initCond(u):  # Initial values of [y,y',y'',y'''];
                  # use 'u' if unknown
    return array([0.0, u[0], 0.0, u[1]])

def r(u):  # Boundary condition residuals-- see Eq. (8.7)
    r = zeros(len(u))
    X,Y = bulStoer(F,x,initCond(u),xStop,H)
    y = Y[len(Y) - 1]
    r[0] = y[0]
    r[1] = y[2]
    return r

def F(x,y):  # First-order differential equations
    F = zeros(4)
    beta = 1.65929
    F[0] = y[1]
    F[1] = y[2]
    F[2] = y[3]
    F[3] = beta*y[2] + 1.0
    return F

x = 0.0                      # Start of integration
xStop = 1.0                  # End of integration
u = array([1.0, 1.0])        # Initial guesses for u
H = 0.25                     # Step size
freq = 2                     # Printout frequency
u = newtonRaphson2(r,u,1.0e-5)
X,Y = bulStoer(F,x,initCond(u),xStop,H)
printSoln(X,Y,freq)
input(''\nPress return to exit'')
```

x	y[0]	y[1]	y[2]	y[3]
0.0000e+00	0.0000e+00	3.5747e-02	0.0000e+00	-4.4069e-01
5.0000e-01	1.1141e-02	-7.0978e-08	-1.0651e-01	-1.1789e-07
1.0000e+00	-4.9879e-08	-3.5747e-02	-8.2884e-08	4.4069e-01

$$v_{\max} = \frac{w_0 L^4}{EI} y(0.5) = 0.001114 \frac{w_0 L^4}{EI} \quad \blacktriangleleft$$

(b)

Running the same program with $\beta = -1.65929$ resulted in

x	y[0]	y[1]	y[2]	y[3]
0.0000e+00	0.0000e+00	4.9976e-02	0.0000e+00	-5.8292e-01
5.0000e-01	1.5661e-02	3.3490e-08	-1.5099e-01	-5.5488e-08
1.0000e+00	2.5840e-11	-4.9976e-02	3.8372e-11	5.8292e-01

$$v_{\max} = 0.001566 \frac{w_0 L^4}{EI} \quad \blacktriangleleft$$

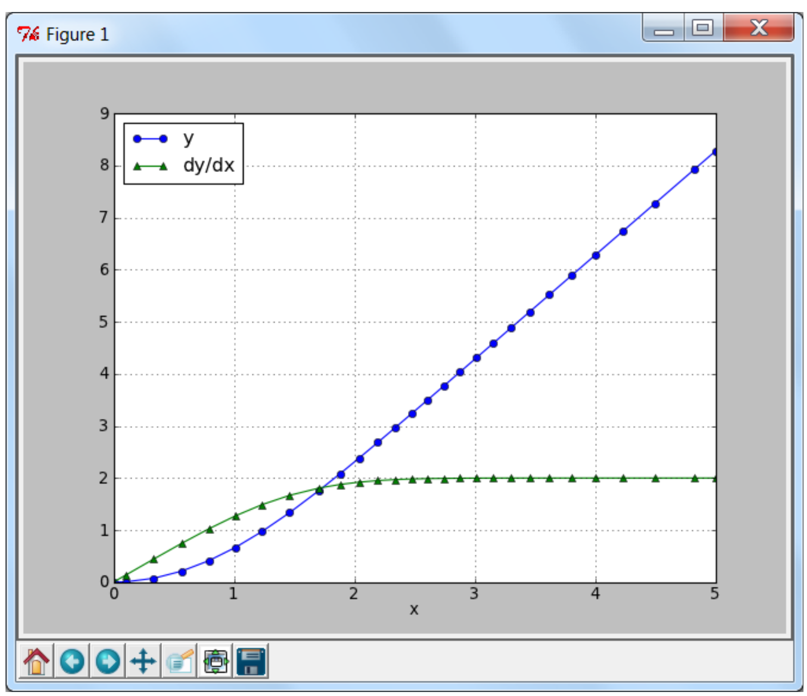
Problem 21

$$y''' = -yy'' \qquad y(0) = y'(0) = 0 \qquad y'(\infty) = 2$$

```
## problem8_1_21
from numpy import zeros,array
from run_kut5 import *
from ridder import *
from printSoln import *
import matplotlib.pyplot as plt


def initCond(u):  # Initial values of [y,y',y"];
                  # use 'u' if unknown
    return array([0.0, 0.0, u])

def r(u):  # Boundary condition residual--see Eq. (8.3)
    X,Y = integrate(F,xStart,initCond(u),xStop,h)
    y = Y[len(Y) - 1]
    r = y[1] - 2.0
    return r

def F(x,y):  # First-order differential equations
```

```
    F = zeros(3)
    F[0] = y[1]
    F[1] = y[2]
    F[2] = -y[0]*y[2]
    return F

xStart = 0.0          # Start of integration
xStop = 5.0           # End of integration (close enough to inf.)
u1 = 0.0; u2 = 2.0    # Trial values of u
h = 0.1               # initial step size
freq = 0              # printout frequency
u = ridder(r,u1,u2)
X,Y = integrate(F,xStart,initCond(u),xStop,h)
printSoln(X,Y,freq)
plt.plot(X,Y[:,0],'-o',X,Y[:,1],'-^')
plt.xlabel('x')
plt.legend(('y','dy/dx'),loc=0)
plt.grid(True); plt.show()
input("\nPress return to exit")
```

```
       x           y[ 0 ]        y[ 1 ]        y[ 2 ]
   0.0000e+00    0.0000e+00    0.0000e+00    1.3282e+00
   5.0000e+00    8.2792e+00    2.0000e+00    1.1161e-07
```



Problem 22

As in Example 8.4, we introduce the dimensionless variables

$$\xi = \frac{x}{L} \qquad y = \frac{EI}{w_0 L^4} v$$

which transforms the differential equation into

$$\frac{d^4 y}{d\xi^4} = 1 + \xi$$

Here is the modified function `shoot4` that solves the problem:

```
## problem8_1_22
from numpy import zeros,array
from bulStoer import *
from newtonRaphson2 import *
from printSoln import *
import matplotlib.pyplot as plt

def initCond(u):  # Initial values of [y,y',y",y"'];
                  # use 'u' if unknown
    return array([0.0, u[0], 0.0, u[1]])

def r(u):  # Boundary condition residuals--see Eq. (8.7)
    r = zeros(len(u))
    X,Y = bulStoer(F,xStart,initCond(u),xStop,H)
    y = Y[len(Y) - 1]
    r[0] = y[0]
    r[1] = y[1]
    return r

def F(x,y):  # First-order differential equations
    F = zeros(4)
    F[0] = y[1]
    F[1] = y[2]
    F[2] = y[3]
    F[3] = 1.0 + x
    return F

xStart = 0.0                # Start of integration
xStop = 1.0                 # End of integration
u = array([0.0, 1.0])       # Initial guess for {u}
H = 0.05                    # Printout incremant
freq = 0                    # Printout frequency
```

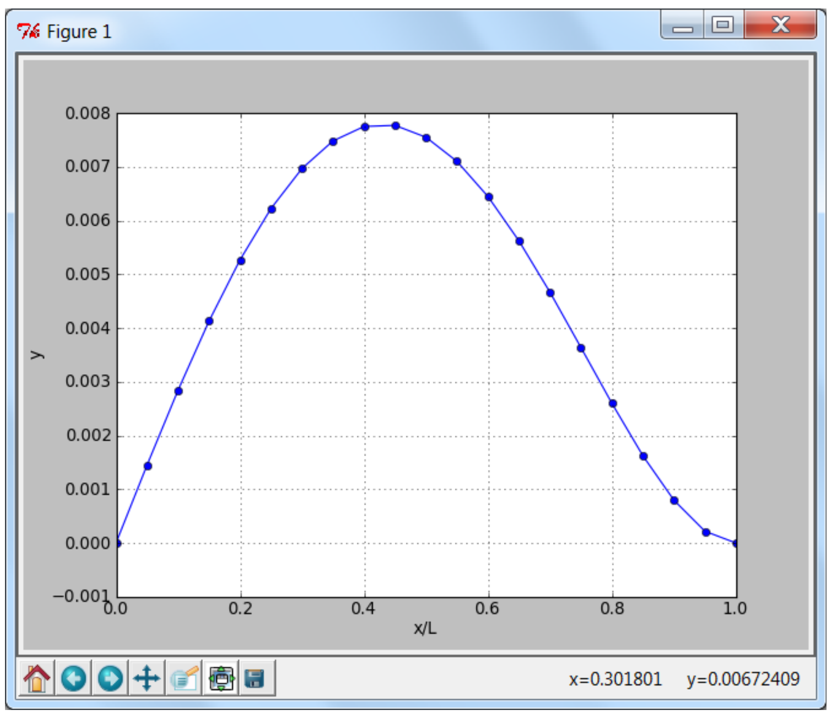
```
u = newtonRaphson2(r,u,1.0e-4)
X,Y = bulStoer(F,xStart,initCond(u),xStop,H)
printSoln(X,Y,freq)
plt.plot(X,Y[:,0],'-o')
plt.xlabel('x/L'); plt.ylabel('y')
plt.grid(True); plt.show()
input("\nPress return to exit")
```

```
     x            y[ 0 ]        y[ 1 ]        y[ 2 ]        y[ 3 ]
0.0000e+00    0.0000e+00    2.9167e-02    0.0000e+00   -4.7500e-01
1.0000e+00   -2.8331e-12   -1.4570e-12    1.9167e-01    1.0250e+00
```

Below is the plot of the dimensionless displacement

$$y = \frac{EI}{w_0 L^4} v$$

v being the actual displacement.



PROBLEM SET 8.2

Problem 1

$$y'' = (2+x)y \qquad y(0) = 0 \qquad y'(1) = 5$$

With $f = (2+x)y$ Eqs. (8.11) become

$$\begin{aligned} y_0 &= 0 \\ y_{i-1} - 2y_i + y_{i+1} - h^2(2+x_i)y_i &= 0, \quad i = 1, 2, \ldots, m-1 \\ 2y_{m-1} - 2y_m - h^2(2+x_m)y_m &= 0 \end{aligned}$$

$$\begin{aligned} y_0 &= 0 \\ y_{i-1} - \left[2 + h^2(2+x_i)\right] y_i + y_{i+1} &= 0, \quad i = 1, 2, \ldots, m-1 \\ 2y_{m-1} - \left[2 + h^2(2+x_m)\right] y_m - y_m &= 0 \end{aligned}$$

Problem 2

$$y'' = y + x^2 \qquad y(0) = 0 \qquad y(1) = 1$$

Using $f = y + x^2$, Eqs. (8.11) become

$$\begin{aligned} y_0 &= 0 \\ y_{i-1} - 2y_i + y_{i+1} - h^2(y_i + x_i^2) &= 0, \quad i = 1, 2, \ldots, m-1 \\ y_m - 1 &= 0 \end{aligned}$$

$$\begin{aligned} y_0 &= 0 \\ y_{i-1} - \left(2 + h^2\right) y_i + y_{i+1} &= h^2 x_i^2, \quad i = 1, 2, \ldots, m-1 \\ y_m &= 1 \end{aligned}$$

Problem 3

$$y'' = e^{-x} y' \qquad y(0) = 1 \qquad y(1) = 0$$

With $f = e^{-x}y$ Eqs. (8.11) are

$$\begin{aligned} y_0 &= 1 \\ y_{i-1} - 2y_i + y_{i+1} - h^2 e^{-x_i} y_i &= 0, \quad i = 1, 2, \ldots, m-1 \\ y_m &= 0 \end{aligned}$$

$$\begin{aligned} y_0 &= 1 \\ y_{i-1} - \left(2 + h^2 e^{-x_i}\right) y_i + y_{i+1} &= 0, \quad i = 1, 2, \ldots, m-1 \\ y_m &= 0 \end{aligned}$$

Problem 4

$$y^{(4)} = y'' - y \qquad y(0) = y(1) = 0 \qquad y'(0) = 1 \qquad y'(1) = -1$$

Substituting $f = y'' - y$ into Eqs. (8.13) we get

$$\begin{aligned} y_{-2} - 4y_{-1} + 6y_0 - 4y_1 + y_2 - h^4 \left(\frac{y_{-1} - 2y_0 + y_1}{h^2} - y_0 \right) &= 0 \\ y_{-1} - 4y_0 + 6y_1 - 4y_2 + y_3 - h^4 \left(\frac{y_0 - 2y_1 + y_2}{h^2} - y_1 \right) &= 0 \\ y_0 - 4y_1 + 6y_2 - 4y_3 + y_4 - h^4 \left(\frac{y_1 - 2y_2 + y_3}{h^2} - y_2 \right) &= 0 \\ &\vdots \\ y_{m-3} - 4y_{m-2} + 6y_{m-1} - 4y_m + y_{m+1} - h^4 \left(\frac{y_{m-2} - 2y_{m-1} + y_m}{h^2} - y_{m-1} \right) &= 0 \\ y_{m-2} - 4y_{m-1} + 6y_m - 4y_{m+1} + y_{m+2} - h^4 \left(\frac{y_{m-1} - 2y_m + y_{m+1}}{h^2} - y_m \right) &= 0 \end{aligned}$$

From Table 8.1 equivalent the boundary conditions are

$$\begin{aligned} y_0 &= 0 \qquad y_{-1} = y_1 - 2h \\ y_m &= 0 \qquad y_{m+1} = y_{m-1} - 2h \end{aligned}$$

Therefore, the finite difference equations become

$$\begin{aligned} y_0 &= 0 \\ -\left(4 + h^2\right) y_0 + \left(7 + 2h^2 + h^4\right) y_1 - \left(4 + h^2\right) y_2 + y_3 &= 2h \\ y_0 - (4 + h^2) y_1 + (6 + 2h^2 + h^4) y_2 - (4 + h^2) y_3 + y_4 &= 0 \\ &\vdots \\ y_{m-3} - \left(4 + h^2\right) y_{m-2} + \left(7 + 2h^2 + h^4\right) y_{m-1} - \left(4 + h^2\right) y_m &= 2h \\ y_m &= 0 \end{aligned}$$

Problem 5

$$y^{(4)} = -9y + x \qquad y(0) = y''(0) = 0 \qquad y'(0) = y'''(0) = 0$$

With $f = -9y + x$ Eqs. (8.13) yield

$$\begin{aligned}
y_{-2} - 4y_{-1} + 6y_0 - 4y_1 + y_2 - h^4\left(-9y_0 + x_0\right) &= 0 \\
y_{-1} - 4y_0 + 6y_1 - 4y_2 + y_3 - h^4\left(-9y_1 + x_1\right) &= 0 \\
y_0 - 4y_1 + 6y_2 - 4y_3 + y_4 - h^4\left(-9y_2 + x_2\right) &= 0 \\
&\vdots \\
y_{m-3} - 4y_{m-2} + 6y_{m-1} - 4y_m + y_{m+1} - h^4\left(-9y_{m-1} + x_{m-1}\right) &= 0 \\
y_{m-2} - 4y_{m-1} + 6y_m - 4y_{m+1} + y_{m+2} - h^4\left(-9y_m + x_m\right) &= 0
\end{aligned}$$

According to Table 8.1 the boundary conditions are equivalent to

$$\begin{aligned}
y_0 &= 0 \qquad y_{-1} = 2y_0 - y_1 \\
y_{m+1} &= y_{m-1} \qquad y_{m+2} = 2y_{m+1} - 2y_{m-1} + y_{m-1} = y_{m-2}
\end{aligned}$$

The finite difference equations now become

$$\begin{aligned}
y_0 &= 0 \\
-2y_0 + (5 + 9h^4)y_1 - 4y_2 + y_3 &= h^4 x_1 \\
y_0 - 4y_1 + (6 + 9h^4)y_2 - 4y_3 + y_4 &= h^4 x_2 \\
&\vdots \\
y_{m-3} - 4y_{m-2} + (7 + 9h^4)y_{m-1} - 4y_m &= h^4 x_{m-1} \\
2y_{m-2} - 8y_{m-1} + (6 + 9h^4)y_m &= h^4 x_m
\end{aligned}$$

Problem 6

$$y'' = xy \qquad y(1) = 1.5 \qquad y(2) = 3$$

The finite difference equations are

$$\begin{aligned}
y_0 &= 1.5 \\
y_{i-1} - 2y_i + y_{i+1} - h^2 x_i y_i &= 0, \quad i = 1, 2, \ldots, m-1 \\
y_m &= 3
\end{aligned}$$

or

$$\begin{aligned}
y_0 &= 1.5 \\
y_{i-1} - \left(2 + h^2 x_i\right) y_i + y_{i+1} &= 0, \quad i = 1, 2, \ldots, m-1 \\
y_m &= 3
\end{aligned}$$

```
## problem8_2_6
from numpy import zeros,ones,array
from LUdecomp3 import *

def equations(x,h,m): # Set up finite difference eqs.
    h2 = h*h
    d = ones(m + 1)
    c = ones(m)
    e = ones(m)
    b = zeros(m+1)
    for i in range(1,m): d[i] = -2.0 - x[i]*h2
    d[0] = 1.0;   e[0] = 0.0; b[0] = 1.5
    d[m] = 1.0; c[m-1] = 0.0; b[m] = 3.0
    return c,d,e,b

xStart = 1.0        # x at left end
xStop = 2.0         # x at right end
m = 20              # Number of mesh spaces

h = (xStop - xStart)/m
x = zeros(m + 1)
for i in range(m + 1): x[i] = xStart + h*i
c,d,e,b = equations(x,h,m)
LUdecomp3(c,d,e)
LUsolve3(c,d,e,b)
print('\n        x              y')
for i in range(0,m + 1,2):
    print('{:14.5e} {:14.5e}'.format(x[i],b[i]))
input("\nPress return to exit")
```

The program prints every second point:

```
        x              y
   1.00000e+00    1.50000e+00
   1.10000e+00    1.53725e+00
   1.20000e+00    1.59142e+00
   1.30000e+00    1.66472e+00
   1.40000e+00    1.75968e+00
   1.50000e+00    1.87931e+00
   1.60000e+00    2.02718e+00
   1.70000e+00    2.20753e+00
   1.80000e+00    2.42548e+00
   1.90000e+00    2.68716e+00
   2.00000e+00    3.00000e+00
```

Problem 7

$$y'' = -2y' - y \qquad y(0) = 0 \qquad y(1) = 1$$

The finite difference equations are

$$\begin{aligned} y_0 &= 0 \\ y_{i-1} - 2y_i + y_{i+1} - h^2\left(-2\frac{y_{i+1} - y_{i-1}}{2h} - y_i\right) &= 0, \quad i = 1, 2, \ldots, m-1 \\ y_m &= 1 \end{aligned}$$

or

$$\begin{aligned} y_0 &= 0 \\ (1-h)y_{i-1} - (2-h^2)y_i + (1+h)y_{i+1} &= 0, \quad i = 1, 2, \ldots, m-1 \\ y_m &= 1 \end{aligned}$$

```
## problem8_2_7
from numpy import zeros,ones,array
from LUdecomp3 import *
from math import exp

def equations(x,h,m):
  # Set up tridiagonal finite difference eqs.
    h2 = h*h
    d = ones(m + 1)*(-2.0 + h2)
    c = ones(m)*(1.0 - h)
    e = ones(m)*(1.0 + h)
    b = zeros(m+1)
    d[0] = 1.0; e[0] = 0.0
    d[m] = 1.0; c[m-1] = 0.0; b[m] = 1.0
    return c,d,e,b

xStart = 0.0        # x at left end
xStop = 1.0         # x at right end
m = 20              # Number of mesh spaces

h = (xStop - xStart)/m
x = zeros(m + 1)
for i in range(m + 1): x[i] = xStart + h*i
c,d,e,b = equations(x,h,m)
LUdecomp3(c,d,e)
LUsolve3(c,d,e,b)
print("\n        x              y        Analytical y")
for i in range(0,m + 1,2):
```

```
    print('{:14.5e} {:14.5e} {:14.5e}' \
          .format(x[i],b[i],x[i]*exp(1.0 - x[i])))
input("\nPress return to exit")

       x              y          Analytical y
   0.00000e+00    0.00000e+00    0.00000e+00
   1.00000e-01    2.46120e-01    2.45960e-01
   2.00000e-01    4.45361e-01    4.45108e-01
   3.00000e-01    6.04421e-01    6.04126e-01
   4.00000e-01    7.29149e-01    7.28848e-01
   5.00000e-01    8.24640e-01    8.24361e-01
   6.00000e-01    8.95334e-01    8.95095e-01
   7.00000e-01    9.45087e-01    9.44901e-01
   8.00000e-01    9.77249e-01    9.77122e-01
   9.00000e-01    9.94717e-01    9.94654e-01
   1.00000e+00    1.00000e+00    1.00000e+00
```

Problem 8

$$y'' = -\frac{1}{x}y' - \frac{1}{x^2}y \qquad y(1) = 0 \qquad y(2) = 0.638\,961$$

The finite difference equations are

$$\begin{aligned} y_0 &= 0 \\ y_{i-1} - 2y_i + y_{i+1} - h^2\left(-\frac{y_{i+1} - y_{i-1}}{2hx_i} - \frac{y_i}{x_i^2}\right) &= 0, \quad i = 1, 2, \ldots, m-1 \\ y_m &= 0.638\,961 \end{aligned}$$

or

$$\begin{aligned} y_0 &= 0 \\ \left(1 - \frac{h}{2x_i}\right)y_{i-1} - \left(2 - \frac{h^2}{x_i^2}\right)y_i + \left(1 + \frac{h}{2x_i}\right)y_{i+1} &= 0, \quad i = 1, 2, \ldots, m-1 \\ y_m &= 0.638\,961 \end{aligned}$$

Here are the finite difference equations:

```
## problem8_2_8
from numpy import zeros,ones,array
from LUdecomp3 import *
from math import log,sin

def equations(x,h,m): # Set up finite difference eqs.
```

```
    h2 = h*h
    d = zeros(m + 1)
    c = zeros(m)
    e = zeros(m)
    b = zeros(m+1)
    for i in range(m):
        c[i] = 1.0 - 0.5*h/x[i+1]
        e[i] = 1.0 + 0.5*h/x[i]
        d[i] = -2.0 +h2/x[i]**2
    d[0] = 1.0;   e[0] = 0.0; b[0] = 0.0
    d[m] = 1.0; c[m-1] = 0.0; b[m] = 0.638961
    return c,d,e,b

xStart = 1.0        # x at left end
xStop = 2.0         # x at right end
m = 20              # Number of mesh spaces

h = (xStop - xStart)/m
x = zeros(m + 1)
for i in range(m + 1):
    x[i] = xStart + h*i
c,d,e,b = equations(x,h,m)
LUdecomp3(c,d,e)
LUsolve3(c,d,e,b)
print("\n        x              y        Analytical y")
for i in range(0,m + 1,2):
    print('{:14.5e} {:14.5e} {:14.5e}'.format(x[i],b[i],sin(log(x[i]))))
input("\nPress return to exit")
```

Every other point of the numerical and analytical solution is printed:

```
       x              y        Analytical y
   1.00000e+00    0.00000e+00   0.00000e+00
   1.10000e+00    9.51699e-02   9.51659e-02
   1.20000e+00    1.81319e-01   1.81313e-01
   1.30000e+00    2.59370e-01   2.59365e-01
   1.40000e+00    3.30164e-01   3.30159e-01
   1.50000e+00    3.94450e-01   3.94446e-01
   1.60000e+00    4.52892e-01   4.52890e-01
   1.70000e+00    5.06077e-01   5.06075e-01
   1.80000e+00    5.54521e-01   5.54521e-01
   1.90000e+00    5.98682e-01   5.98681e-01
   2.00000e+00    6.38961e-01   6.38961e-01
```

Problem 9

$$y'' = y^2 \sin y \qquad y'(0) = 0 \qquad y(\pi) = 1$$

The finite difference equations are

$$\begin{aligned} -2y_0 + 2y_1 - h^2 F(x_0, y_0, y_0') &= 0 \\ y_{i-1} - 2y_i + y_{i+1} - h^2 F(x_i, y_i, y_i') &= 0, \quad i = 1, 2, \ldots, m-1 \\ y_m &= 1 \end{aligned}$$

In arriving at the first equation, we utilize the equivalent boundary condition $y_{-1} = y_1$. The quadratic $y = (x/\pi)^2$ was chosen for the starting solution (note that its satisfies the prescribed boundary conditions).

```
## problem8_2_9
from numpy import zeros,array,arange
from newtonRaphson2 import *
from math import sin,pi

def residual(y):  # Residuals of finite diff. Eqs. (8.11)
    r = zeros(m + 1)
    hh = h**2
    r[0] = -2.0*(y[0] - y[1]) - hh*F(x[0],y[0],0.0)
    r[m] = y[m] - 1.0
    for i in range(1,m):
        r[i] = y[i-1] - 2.0*y[i] + y[i+1]  \
             - hh*F(x[i],y[i],(y[i+1] - y[i-1])/2.0/h)
    return r

def F(x,y,yPrime):  # Differential eqn. y" = F(x,y,y')
    F = (y**2)*sin(y)
    return F

def startSoln(x): # Starting solution y(x)
    y = zeros(m + 1)
    for i in range(m + 1): y[i] = (x[i]/pi)**2
    return y

xStart = 0.0            # x at left end
xStop = pi              # x at right end
m = 20                  # Number of mesh intevals

h = (xStop - xStart)/m
x = arange(xStart,xStop + h,h)
y = newtonRaphson2(residual,startSoln(x),1.0e-5)
```

```
print("\n        x               y")
for i in range(0,m + 1,2):
    print("{:14.5e} {:14.5e}".format(x[i],y[i]))
input("\nPress return to exit")
```

The program prints every other point of the solution:

```
        x               y
  0.00000e+00    4.13382e-01
  3.14159e-01    4.16779e-01
  6.28319e-01    4.27138e-01
  9.42478e-01    4.44982e-01
  1.25664e+00    4.71274e-01
  1.57080e+00    5.07572e-01
  1.88496e+00    5.56311e-01
  2.19911e+00    6.21320e-01
  2.51327e+00    7.08753e-01
  2.82743e+00    8.28917e-01
  3.14159e+00    1.00000e+00
```

Problem 10

$$y'' = -2y(2xy' + y) \qquad y(0) = \frac{1}{2} \qquad y'(1) = -\frac{2}{9}$$

The finite difference equations are

$$\begin{aligned} y_0 &= 0.5 \\ y_{i-1} - 2y_i + y_{i+1} - h^2 F(x_i, y_i, y_i') &= 0, \quad i = 1, 2, \ldots, m-1 \\ y_{m-1} - 2y_m + y_{m+1} - h^2 F(x_m, y_m, y_m') &= 0 \end{aligned}$$

The bounday condition $y_m' = -2/9$ is equivalent to

$$\frac{y_{m+1} - y_{m-1}}{2h} = -\frac{2}{9} \qquad y_{m+1} = y_{m-1} - \frac{4}{9}h$$

so that the last finite difference equation becomes

$$2y_{m-1} - 2y_m - \frac{4}{9}h - h^2 F\left(x_m, y_m, -\frac{2}{9}\right) = 0$$

```
## problem8_2_10
from numpy import zeros,array,arange
from newtonRaphson2 import *

def residual(y):  # Residuals of finite diff. Eqs. (8.11)
```

```
    r = zeros(m + 1)
    c = 2.0/9.0
    hh = h**2
    r[0] = y[0] - 0.5
    r[m] = 2.0*(y[m-1] - y[m]) - 2.0*c*h - hh*F(x[m],y[m],-c)
    for i in range(1,m):
        r[i] = y[i-1] - 2.0*y[i] + y[i+1]  \
             - hh*F(x[i],y[i],(y[i+1] - y[i-1])/2.0/h)
    return r

def F(x,y,yPrime):  # Differential eqn. y" = F(x,y,y')
    F = -2.0*y*(2.0*x*yPrime + y)
    return F

def startSoln(x): # Starting solution y(x)
    y = zeros(m + 1)
    c = 2.0/9.0
    for i in range(m + 1): y[i] = 0.5 - c*x[i]
    return y

xStart = 0.0            # x at left end
xStop = 1.0             # x at right end
m = 20                  # Number of mesh intevals

h = (xStop - xStart)/m
x = arange(xStart,xStop + h,h)
y = newtonRaphson2(residual,startSoln(x),1.0e-5)
print("\n        x               y        Analytical y")
for i in range(0,m + 1,2):
    print("{:14.5e} {:14.5e} {:14.5e}"  \
          .format(x[i],y[i],1.0/(2.0 + x[i]**2)))
input("\nPress return to exit")
```

The program prints every other solution point together with the analytical solution:

```
        x               y        Analytical y
   0.00000e+00    5.00000e-01    5.00000e-01
   1.00000e-01    4.97461e-01    4.97512e-01
   2.00000e-01    4.90089e-01    4.90196e-01
   3.00000e-01    4.78306e-01    4.78469e-01
   4.00000e-01    4.62746e-01    4.62963e-01
   5.00000e-01    4.44178e-01    4.44444e-01
   6.00000e-01    4.23420e-01    4.23729e-01
   7.00000e-01    4.01262e-01    4.01606e-01
   8.00000e-01    3.78416e-01    3.78788e-01
```

```
9.00000e-01    3.55479e-01    3.55872e-01
1.00000e+00    3.32925e-01    3.33333e-01
```

Problem 11

$$y'' = \begin{cases} -0.25x & 0 < x < 0.25 \\ -\dfrac{0.25}{\gamma}\left[x - 2\left(x - 0.25\right)^2\right] & 0.25 < x < 0.5 \end{cases}$$

$$y(0) = 0 \qquad y'(0.5) = 0$$

The finite difference equations are

$$\begin{aligned} y_0 &= 0 \\ y_{i-1} - 2y_i + y_{i+1} &= \begin{cases} -0.25x_i h^2 & 0 < x_i < 0.25 \\ -\dfrac{0.25}{\gamma}\left[x_i - 2\left(x_i - 0.25\right)^2\right] h^2 & 0.25 < x_i < 0.5 \end{cases} \\ y_{m-1} - 2y_m + y_{m+1} &= -\frac{0.25}{\gamma}\left[x_m - 2\left(x_m - 0.25\right)^2\right] h^2 \end{aligned}$$

The bounday condition $y'_m = 0$ is equivalent to $y_{m+1} = y_{m-1}$ so that the last finite difference equation becomes

$$2y_{m-1} - 2y_m = -\frac{0.25}{\gamma}\left[x_m - 2\left(x_m - 0.25\right)^2\right] h^2$$

```
## problem8_2_11
from numpy import zeros,ones,array
from LUdecomp3 import *

def equations(x,h,m): # Set up finite difference eqs.
    h2 = h*h
    d = ones(m + 1)*(-2.0)
    c = ones(m)
    e = ones(m)
    b = zeros(m+1)
    for i in range(int(m/2+1)): b[i] = -0.25*x[i]*h2
    for i in range(int(m/2+1),m+1):
        b[i] = -0.25*(x[i]-2.0*(x[i]-0.25)**2)*h2/gamma
  # If there is a node at x = 0.25, average the values of RHS's
    if m % 2 == 0: b[m/2] = -0.125*(1.0 + 1.0/gamma)* x[m/2]*h2
    d[0] = 1.0; e[0] = 0.0; b[0] = 0.0
    c[m-1] = 2.0
```

```
    return c,d,e,b

xStart = 0.0        # x at left end
xStop = 0.5         # x at right end
m = 20              # Number of mesh spaces
gamma = 1.5

h = (xStop - xStart)/m
x = zeros(m + 1)
for i in range(m + 1):
    x[i] = xStart + h*i
c,d,e,b = equations(x,h,m)
LUdecomp3(c,d,e)
LUsolve3(c,d,e,b)
print("y(0.5) =",b[m])
input("\nPress return to exit")
```

```
y(0.5) = 0.00662434895833
```

Thus the numerical solution gives

$$v_{\max} = 0.006624\frac{w_0L^4}{EI} \quad \blacktriangleleft$$

whereas the analytical solution is

$$v_{\max} = \frac{61}{9216}\frac{w_0L^4}{EI} = 0.006619\frac{w_0L^4}{EI}$$

Problem 12

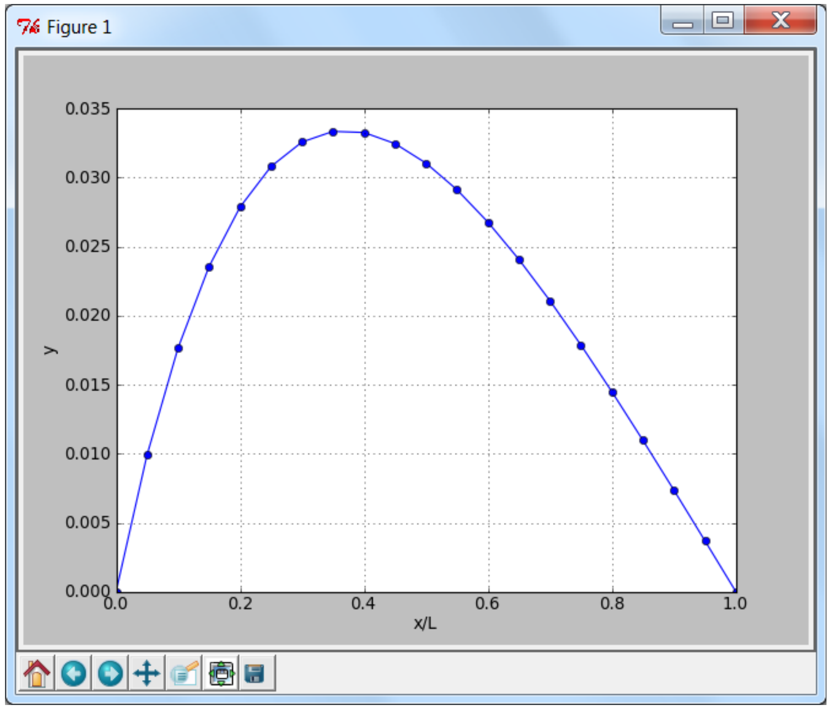
$$y'' = -\frac{1-x}{1+[(\delta-1)x]^4} \qquad y(0) = y(1) = 0$$

The finite difference equations are

$$\begin{aligned} y_0 &= 0 \\ y_{i-1} - 2y_i + y_{i+1} &= -\frac{1-x_i}{1+[(\delta-1)x_i]^4}h^2, \quad i = 1,2,\ldots,m-1 \\ y_m &= 0 \end{aligned}$$

```
## problem8_2_12
from numpy import zeros,ones,array
from LUdecomp3 import *
import matplotlib.pyplot as plt
```

```
def equations(x,h,m): # Set up finite difference eqs.
    h2 = h*h
    d = ones(m + 1)*(-2.0)
    c = ones(m)
    e = ones(m)
    b = zeros(m+1)
    for i in range(1,m):
        b[i] = -h2*(1.0 - x[i])/(1.0 + (delta - 1.0)*x[i])**4
    d[0] = 1.0;   e[0] = 0.0
    d[m] = 1.0; c[m-1] = 0.0
    return c,d,e,b

xStart = 0.0        # x at left end
xStop = 1.0         # x at right end
m = 20              # Number of mesh spaces
delta = 1.5

h = (xStop - xStart)/m
x = zeros(m + 1)
for i in range(m + 1):
    x[i] = xStart + h*i
c,d,e,b = equations(x,h,m)
LUdecomp3(c,d,e)
LUsolve3(c,d,e,b)
plt.plot(x,b,'-o')
plt.xlabel('x/L'); plt.ylabel('y')
plt.grid(True); plt.show()
input("\nPress return to exit")
```



Problem 13

$$y^{(4)} = x \qquad y(0) = y''(0) = y(1) = y''(1) = 0$$

Taking into account the boundary conditions $y_{-1} = -y_1$ and $y_{m+1} = -y_{m-1}$, the finite difference equations are

$$\begin{aligned}
y_0 &= 0 \\
-4y_0 + 5y_1 - 4y_2 + y_3 &= h^4 x_1 \\
y_{i-2} - 4y_{i-1} + 6y_i - 4y_{i+1} + y_{i+2} &= h^4 x_i, \quad i = 2, 3, \ldots, m-2 \\
y_{m-3} - 4y_{m-2} + 5y_{m-1} - 4y_m &= h^4 x_{m-1} \\
y_m &= 0
\end{aligned}$$

```
## problem8_2_13
from numpy import zeros,ones,array,arange
from LUdecomp5 import *

def equations(x,h,m): # Set up finite difference eqs.
    h4 = h**4
    d = ones(m + 1)*6.0
    e = ones(m)*(-4.0)
    f = ones(m-1)
```

```
    b = zeros(m+1)
    d[0] = 1.0; e[0] = 0.0; f[0] = 0.0
    d[1] = 5.0
    d[m-1] = 5.0
    d[m] = 1.0; e[m-1] = 0.0; f[m-2] = 0.0
    for i in range(1,m): b[i] = x[i]*h4
    return d,e,f,b

xStart = 0.0        # x at left end
xStop = 1.0         # x at right end
m = 20              # Number of mesh spaces

h = (xStop - xStart)/m
x = arange(xStart,xStop + h,h)
d,e,f,b = equations(x,h,m)
d,e,f = LUdecomp5(d,e,f)
y = LUsolve5(d,e,f,b)
print("\n        x              y")
for i in range(m + 1):
    print("{:14.5e} {:14.5e}".format(x[i],y[i]))
input("\nPress return to exit")
```

Only the points needed for the computation of end slopes and mid-span displacement are shown below.

```
       x              y
   0.00000e+00    0.00000e+00
   5.00000e-02    9.70484e-04
   1.00000e-01    1.92019e-03

   5.00000e-01    6.52344e-03

   9.00000e-01    2.17669e-03
   9.50000e-01    1.10764e-03
   1.00000e+00    0.00000e+00
```

$$y(0.5) = 6.523 \times 10^{-3}$$

From Tables 5.3:

$$\begin{aligned} y'(0) &\approx \frac{-3y(0) + 4y(0.05) - y(0.1)}{2h} \\ &= \frac{-3(0) + 4(0.970484) - 1.92019}{0.1} \times 10^{-3} = 19.62 \times 10^{-3} \end{aligned}$$

$$
\begin{aligned}
y'(1) &= \frac{y(0.9) - 4y(0.95) + 3y(1.0)}{2h} \\
&= \frac{2.17669 - 4(1.10764) + 3(0)}{0.1} \times 10^{-3} = -22.54 \times 10^{-3}
\end{aligned}
$$

Therefore, the mid-span displacement and the end slopes are (the numbers in parenthesis are the numerical factors obtained from the analytical solution):

$$
\begin{aligned}
v(0.5L) &= 6.523 \times 10^{-3} \frac{w_0 L^4}{EI} \quad \blacktriangleleft \qquad (6.510) \\
\left.\frac{dv}{dx}\right|_{x=0} &= 19.62 \times 10^{-3} \frac{w_0 L^3}{EI} \quad \blacktriangleleft \qquad (19.44) \\
\left.\frac{dv}{dx}\right|_{x=L} &= -22.54 \times 10^{-3} \frac{w_0 L^3}{EI} \quad \blacktriangleleft \qquad (-22.22)
\end{aligned}
$$

Problem 14

$$
y^{(4)} = \beta y'' + 1 \qquad y(0) = y''(0) = y(1) = y''(1) = 0
$$

The finite difference equations are

$$
\begin{aligned}
y_0 &= 0 \\
y_{-1} - 4y_0 + 6y_1 - 4y_2 + y_3 - h^4 \left(\beta \frac{y_0 - 2y_1 + y_2}{h^2} + 1 \right) &= 0 \\
y_0 - 4y_1 + 6y_2 - 4y_3 + y_4 - h^4 \left(\beta \frac{y_1 - 2y_2 + y_3}{h^2} + 1 \right) &= 0 \\
&\vdots \\
y_{m-3} - 4y_{m-2} + 6y_{m-1} - 4y_m + y_{m+1} - h^4 \left(\beta \frac{y_{m-2} - 2y_{m-1} + y_m}{h^2} + 1 \right) &= 0 \\
y_m &= 0
\end{aligned}
$$

After using the boundary conditions $y_{-1} = -y_1$ and $y_{m+1} = -y_{m-1}$, we get

$$
\begin{aligned}
y_0 &= 0 \\
-\left(4 + h^2\beta\right) y_0 + \left(5 + 2h^2\beta\right) y_1 - \left(4 + h^2\beta\right) y_1 + y_3 &= h^4 \\
y_0 - \left(4 + h^2\beta\right) y_1 + \left(6 + 2h^2\beta\right) y_2 - \left(4 + h^2\beta\right) y_3 + y_4 &= h^4 \\
&\vdots \\
y_{m-3} - \left(4 + h^2\beta\right) y_{m-2} + \left(5 + 2h^2\beta\right) y_{m-1} - \left(4 + h^2\beta\right) y_m &= h^4 \\
y_m &= 0
\end{aligned}
$$

(a)

```
## problem8_2_14
from numpy import zeros,ones,array
from LUdecomp5 import *

def equations(x,h,m): # Set up finite difference eqs.
    h2 = h*h
    h4 = h2*h2
    d = ones(m + 1)*(6.0 + 2.0*h2*beta)
    e = ones(m)*(-4.0 - h2*beta)
    f = ones(m-1)
    b = ones(m+1)*h4
    d[0] = 1.0; e[0] = 0.0; f[0] = 0.0; b[0] = 0.0
    d[1] = 5.0 + 2.0*h2*beta
    d[m-1] = 5.0 + 2.0*h2*beta
    d[m] = 1.0; e[m-1] = 0.0; f[m-2] = 0.0; b[m] = 0.0
    return d,e,f,b

xStart = 0.0        # x at left end
xStop = 1.0         # x at right end
m = 20              # Number of mesh spaces
beta = 1.65929

h = (xStop - xStart)/m
x = zeros(m + 1)
for i in range(m + 1): x[i] = xStart + h*i
d,e,f,b = equations(x,h,m)
LUdecomp5(d,e,f)
LUsolve5(d,e,f,b)
print("\n        x               y")
for i in range(0,m + 1,10):
    print("{:14.5e} {:14.5e}".format(x[i],b[i]))
input("\nPress return to exit")
```

```
       x              y
   0.00000e+00    0.00000e+00
   5.00000e-01    1.11594e-02
   1.00000e+00    0.00000e+00
```

$$v_{\max} = 0.01116\frac{w_0 L^4}{EI} \blacktriangleleft$$

(b)

Running the program with negative β yields

```
      x              y
0.00000e+00    0.00000e+00
5.00000e-01    1.56995e-02
1.00000e+00    0.00000e+00
```

Problem 15

$$y^{(4)} = -\gamma y + 1 \qquad y(0) = y''(0) = y(1) = y''(1) = 0$$

The finite difference equations are

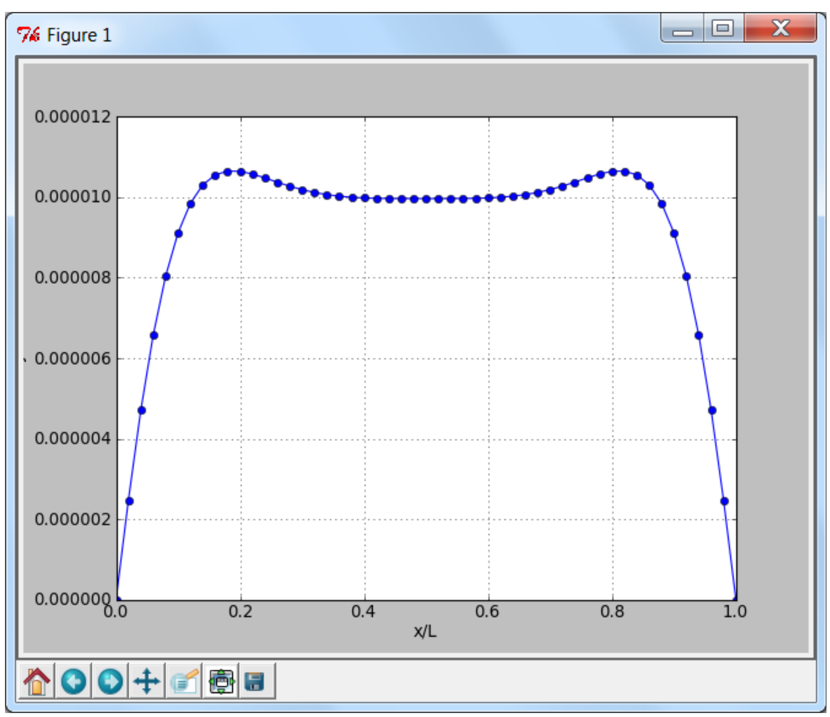
$$\begin{aligned}
y_0 &= 0 \\
y_{-1} - 4y_0 + 6y_1 - 4y_2 + y_3 - h^4\left(-\gamma y_1 + 1\right) &= 0 \\
y_0 - 4y_1 + 6y_2 - 4y_3 + y_4 - h^4\left(-\gamma y_2 + 1\right) &= 0 \\
&\vdots \\
y_{m-3} - 4y_{m-2} + 6y_{m-1} - 4y_m + y_{m+1} - h^4\left(-\gamma y_{m-1} + 1\right) &= 0 \\
y_m &= 0
\end{aligned}$$

With the boundary conditions $y_{-1} = -y_1$ and $y_{m+1} = -y_{m-1}$ these equations become

$$\begin{aligned}
y_0 &= 0 \\
-4y_0 + \left(5 + h^4\gamma\right) y_1 - 4y_2 + y_3 &= h^4 \\
y_0 - 4y_1 + \left(6 + h^4\gamma\right) y_2 - 4y_3 + y_4 &= h^4 \\
&\vdots \\
y_{m-3} - 4y_{m-2} + \left(5 + h^4\gamma\right) y_{m-1} - 4y_m &= h^4 \\
y_m &= 0
\end{aligned}$$

```
## problem8_2_15
from numpy import zeros,ones,array
from LUdecomp5 import *
import matplotlib.pyplot as plt

def equations(x,h,m): # Set up finite difference eqs.
    h4 = h**4
    d = ones(m + 1)*(6.0 + h4*gamma)
    e = ones(m)*(-4.0)
```

```
    f = ones(m-1)
    b = ones(m+1)*h4
    d[0] = 1.0; e[0] = 0.0; f[0] = 0.0; b[0] = 0.0
    d[1]   = 5.0 + h4*gamma
    d[m-1] = 5.0 + h4*gamma
    d[m] = 1.0; e[m-1] = 0.0; f[m-2] = 0.0; b[m] = 0.0
    return d,e,f,b

xStart = 0.0          # x at left end
xStop = 1.0           # x at right end
m = 50                # Number of mesh spaces
gamma = 1.0e5

h = (xStop - xStart)/m
x = zeros(m + 1)
for i in range(m + 1):
    x[i] = xStart + h*i
d,e,f,b = equations(x,h,m)
LUdecomp5(d,e,f)
LUsolve5(d,e,f,b)
plt.plot(x,b,'-o')
plt.xlabel('x/L'); plt.ylabel('y')
plt.grid(True); plt.show()
input("\nPress return to exit")
```



Problem 16

$$
\begin{aligned}
y^{(4)} &= \begin{cases} -\gamma y & \text{in } 0 < x < 0.25 \\ -\gamma y + 1 & \text{in } 0.25 < x < 0.5 \end{cases} \\
y''(0) &= y'''(0) = y'(0.5) = y'''(0.5) = 0
\end{aligned}
$$

The finite difference equations are

$$
\begin{aligned}
y_{-2} - 4y_{-1} + 6y_0 - 4y_1 + y_2 - h^4\left(-\gamma y_0\right) &= 0 \\
y_{-1} - 4y_0 + 6y_1 - 4y_2 + y_3 - h^4\left(-\gamma y_1\right) &= 0 \\
y_0 - 4y_1 + 6y_2 - 4y_3 + y_4 - h^4\left(-\gamma y_2\right) &= 0 \\
&\vdots \\
y_{m-3} - 4y_{m-2} + 6y_{m-1} - 4y_m + y_{m+1} - h^4\left(-\gamma y_{m-1} + 1\right) &= 0 \\
y_{m-2} - 4y_{m-1} + 6y_m - 4y_{m+1} + y_{m+2} - h^4\left(-\gamma y_m + 1\right) &= 0
\end{aligned}
$$

Substituting the equivalent boundary conditions in Table 8.1:

$$
\begin{aligned}
y_{-1} &= 2y_0 - y_1 \qquad y_{-2} = 2y_{-1} - 2y_1 + y_2 = 4y_0 - 4y_1 + y_2 \\
y_{m+1} &= y_{m-1} \qquad y_{m+2} = 2y_{m+1} - 2y_{m-1} + y_{m-2} = y_{m-2}
\end{aligned}
$$

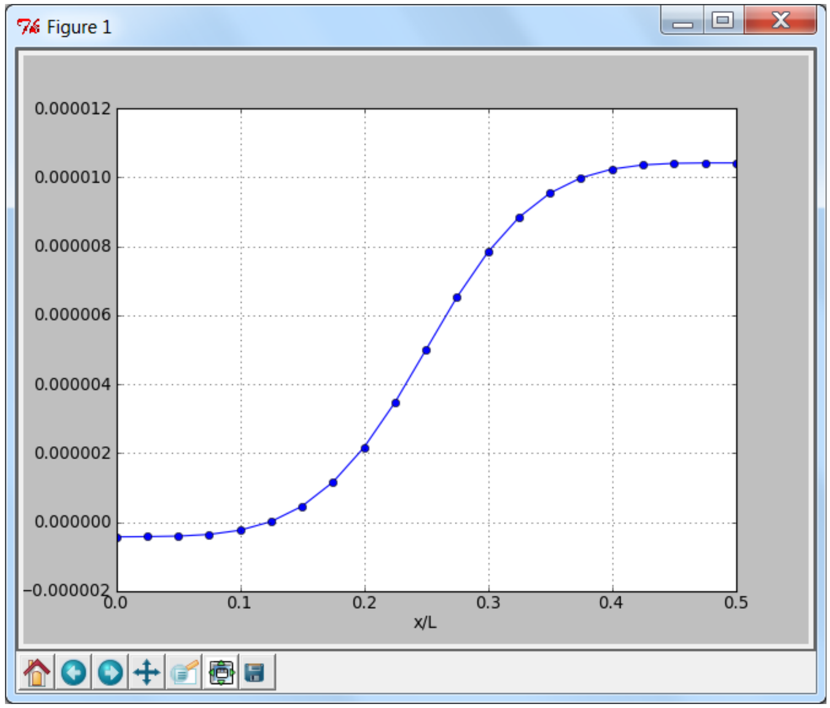
the finite difference equations become

$$
\begin{aligned}
\left(2 + h^4\gamma\right) y_0 - 4y_1 + 2y_2 &= 0 \\
-2y_0 + \left(5 + h^4\gamma\right) y_1 - 4y_2 + y_3 &= 0 \\
y_0 - 4y_1 + \left(6 + h^4\gamma\right) y_2 - 4y_3 + y_4 &= 0 \\
&\vdots \\
y_{m-3} - 4y_{m-2} + \left(7 + h^4\gamma\right) y_{m-1} - 4y_m &= h^4 \\
2y_{m-2} - 8y_{m-1} + \left(6 + h^4\gamma\right) y_m &= h^4
\end{aligned}
$$

To obtain a symmetric coefficient matrix, the first and last equations must be divided by 2.

```
## problem8_2_16
from numpy import zeros,ones,array
from LUdecomp5 import *
import matplotlib.pyplot as plt

def equations(x,h,m): # Set up finite difference eqs.
    h4 = h**4
    d = ones(m + 1)*(6.0 + h4*gamma)
    e = ones(m)*(-4.0)
    f = ones(m-1)
```

```
    b = zeros(m+1)
    d[0] = 1.0 + 0.5*h4*gamma
    d[1] = 5.0 + h4*gamma
    e[0] = -2.0
    d[m-1] = 7.0 + h4*gamma
    d[m] = 3.0 + 0.5*h4*gamma
    for i in range(int(m/2+1),m+1): b[i] = h4
    if m % 2 == 0: b[m/2] = 0.5*h4
    b[m] = 0.5*h4
    return d,e,f,b

xStart = 0.0        # x at left end
xStop = 0.5         # x at right end
m = 20              # Number of mesh spaces
gamma = 1.0e5

h = (xStop - xStart)/m
x = zeros(m + 1)
for i in range(m + 1): x[i] = xStart + h*i
d,e,f,b = equations(x,h,m)
LUdecomp5(d,e,f)
LUsolve5(d,e,f,b)
plt.plot(x,b,'-o')
plt.xlabel('x/L'); plt.ylabel('y')
plt.grid(True); plt.show()
input("\nPress return to exit")
```



Problem 17

Introducing the central finite difference approximations into the differential equation

$$y'' = r(x) + s(x)y + t(x)y'$$

we obtain

$$\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} = r_i + s_i y_i + t_i \frac{y_{i+1} - y_{i-1}}{2h}$$

where we used the notation $r_i = r(x_i)$ etc. After collecting terms, the finite difference equations become

$$\left(1 + \frac{h}{2}t_i\right) y_{i-1} - \left(2 + h^2 s_i\right) y_i + \left(1 - \frac{h}{2}t_i\right) y_{i+1} = h^2 r_i$$

The first and last of these equations are modified when the boundary conditions are introduced. The boundary conditions at the left end are prescribed by `alpha = (code,value)`, where `code` specifies the variable on which the condition is imposed (`code = 0` refers to y and `code = 1` refers to y') and `value` is the prescribed value. The boundary conditions at the right end are prescibed by `beta = (code,value)` in the same manner.

```
## problem8_2_17
```

```
from numpy import zeros,ones,array
from LUdecomp3 import *

def equations(coeffts,alpha,beta): # Finite difference eqs.
    b = zeros(m + 1)
    d = zeros(m + 1)
    c = ones(m)
    e = ones(m)
    for i in range(1,m):
        r,s,t = coeffts(x[i])
        c[i-1] = 1.0 + h/2.0*t
        d[i] = -(2.0 + h2*s)
        e[i] = 1.0 - h/2.0*t
        b[i] = h2*r
    code,value = alpha
    if code == 0: # y(xStart) is given
        d[0] = 1.0
        e[0] = 0.0
        b[0] = value
    else:         # y'(xStart) is given
        r,s,t = coeffts(x[0])
        d[0] = -(2.0 + h2*s)
        e[0] = 2.0
        b[0] = h2*(r + t*value) + 2.0*h*value
    code,value = beta
    if code == 0: # y(xStop) is given
        d[m] = 1.0
        c[m-1] = 0.0
        b[m] = value
    else:         # y'(xStop) is given
        r,s,t = coeffts(x[m])
        d[m] = -(2.0 + h2*s)
        c[m-1] = 2.0
        b[m] = h2*(r + t*value) - 2.0*h*value
    return c,d,e,b

def coeffts(x):
    # Specify r,s,t in y" = r(x) + s(x)y + t(x)y'
    r = 0.0
    s = -1.0/x**2
    t = -1.0/x
    return r,s,t

xStart = 1.0            # x at left end
xStop = 2.0             # x at right end
```

```
m = 20                  # Number of mesh spaces
alpha = (0, 0.0)        # Bound. cond. at xStart (code,value)
beta = (0, 0.638961)    # Bound. cond. at xStop  (code,value)
# If y is given, use code = 0; if y' is given, use code = 1
freq = 2                # Printout frequency

h = (xStop - xStart)/m
h2 = h*h
x = zeros(m + 1)
for i in range(m + 1):
    x[i] = xStart + h*i
c,d,e,b = equations(coeffts,alpha,beta)
LUdecomp3(c,d,e)
LUsolve3(c,d,e,b)
print("        x               y")
for i in range(0,m + 1,freq):
    print("{:14.5e} {:14.5e}".format(x[i],b[i]))
input("\nPress return to exit")
```

```
        x               y
   1.00000e+00    0.00000e+00
   1.10000e+00    9.51699e-02
   1.20000e+00    1.81319e-01
   1.30000e+00    2.59370e-01
   1.40000e+00    3.30164e-01
   1.50000e+00    3.94450e-01
   1.60000e+00    4.52892e-01
   1.70000e+00    5.06077e-01
   1.80000e+00    5.54521e-01
   1.90000e+00    5.98682e-01
   2.00000e+00    6.38961e-01
```

Problem 18

It is convenient to introduce he variable $x = r/a$. The differential equation and the boundary conditions then become

$$\frac{d^2T}{dx^2} = -\frac{1}{x}\frac{dT}{dx} \qquad T|_{x=0.5} = 0 \qquad T|_{x=1} = 200^\circ \text{ C}$$

Using 11 mesh points, the finite difference equations, Eqs. (8.11), are

$$\begin{aligned} T_1 &= 0 \\ T_{i-1} - 2T_i + T_{i+1} - h^2\left(-\frac{1}{x_i}\frac{T_{i+1}-T_{i-1}}{2h}\right) &= 0, \quad i = 2, 3, \ldots 10 \\ T_{11} &= 200 \end{aligned}$$

or

$$\begin{aligned} T_1 &= 0 \\ \left(1 - \frac{h}{2x_i}\right)T_{i-1} - 2T_i + \left(1 + \frac{h}{2x_i}\right)T_{i+1} &= 0, \quad i = 2, 3, \cdots 10 \\ T_{11} &= 200 \end{aligned}$$

The following program is based on Example 8.6. It utilizes the tridiagonal structure of the equations.

```
## problem8_2_18
from numpy import zeros,ones,arange
from LUdecomp3 import *
from math import log

def equations(x,h,m): # Set up finite difference eqs.
    h2 = h*h
    d = ones(m + 1)*(-2.0)
    c = zeros(m)
    e = zeros(m)
    for i in range(m):
        c[i] = 1.0 - h/2.0/x[i+1]
        e[i] = 1.0 + h/2.0/x[i]
    b = zeros(m+1)
    d[0] = 1.0
    d[m] = 1.0
    b[m] = 200.0
    c[m-1] = 0.0
    e[0] = 0.0
    return c,d,e,b

# Numerical solution
xStart = 0.5        # x at left end
xStop = 1.0         # x at right end
m = 10              # Number of mesh spaces
h = (xStop - xStart)/m
x = arange(xStart,xStop + h,h)
c,d,e,b = equations(x,h,m)
c,d,e = LUdecomp3(c,d,e)
```

```
y = LUsolve3(c,d,e,b)

# Analytical solution
y_anal = zeros(m+1)
for i in range(m+1):
    y_anal[i] = 200*(1.0 - log(x[i])/log(0.5))

print("\n        x          y_numeric      y_analytic")
for i in range(m + 1):
    print("{:14.5e} {:14.5e} {:14.5e}".format(x[i],y[i],y_anal[i]))
input("\nPress return to exit")
```

```
        x          y_numeric      y_analytic
   5.00000e-01    0.00000e+00    0.00000e+00
   5.50000e-01    2.74923e+01    2.75007e+01
   6.00000e-01    5.25939e+01    5.26069e+01
   6.50000e-01    7.56874e+01    7.57023e+01
   7.00000e-01    9.70703e+01    9.70854e+01
   7.50000e-01    1.16979e+02    1.16993e+02
   8.00000e-01    1.35602e+02    1.35614e+02
   8.50000e-01    1.53097e+02    1.53107e+02
   9.00000e-01    1.69593e+02    1.69599e+02
   9.50000e-01    1.85196e+02    1.85200e+02
   1.00000e+00    2.00000e+02    2.00000e+02
```

PROBLEM SET 9.1

Problem 1

$$\mathbf{A} = \begin{bmatrix} 7 & 3 & 1 \\ 3 & 9 & 6 \\ 1 & 6 & 8 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

From Eqs. (9.26):

$$\mathbf{L} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{bmatrix} \qquad \mathbf{L}^{-1} = \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/3 & 0 \\ 0 & 0 & 1/2 \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} 7/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1 \\ 1/4 & 1 & 2 \end{bmatrix} \quad \blacktriangleleft$$

$$\mathbf{x} = (\mathbf{L}^{-1})^T \mathbf{z} = \begin{bmatrix} z_1/2 \\ z_2/3 \\ z_3/2 \end{bmatrix} \quad \blacktriangleleft$$

Problem 2

$$\mathbf{A} = \begin{bmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

Choleski's decomposition of $\mathbf{B}$:

$$\begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \begin{bmatrix} L_{11} & L_{21} & L_{31} \\ 0 & L_{22} & L_{32} \\ 0 & 0 & L_{33} \end{bmatrix} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

$$\begin{array}{lll} L_{11}^2 = 2 & & L_{11} = \sqrt{2} \\ L_{11}L_{21} = -1 & \sqrt{2}L_{21} = -1 & L_{21} = -1/\sqrt{2} \\ L_{31}L_{11} = 0 & & L_{31} = 0 \\ L_{21}^2 + L_{22}^2 = 2 & 1/2 + L_{22}^2 = 2 & L_{22} = \sqrt{3/2} \\ L_{31}L_{21} + L_{32}L_{22} = -1 & L_{32}\sqrt{3/2} = -1 & L_{32} = -\sqrt{2/3} \\ L_{31}^2 + L_{32}^2 + L_{33}^2 = 1 & 2/3 + L_{33}^2 = 1 & L_{33} = \sqrt{1/3} \end{array}$$

$$\mathbf{L} = \begin{bmatrix} \sqrt{2} & 0 & 0 \\ \sqrt{1/2} & \sqrt{3/2} & 0 \\ 0 & -\sqrt{2/3} & \sqrt{1/3} \end{bmatrix}$$

Inversion of $\mathbf{L}$:

$$\begin{bmatrix} L_{11}^{-1} & 0 & 0 \\ L_{21}^{-1} & L_{22}^{-1} & 0 \\ L_{31}^{-1} & L_{32}^{-1} & L_{33}^{-1} \end{bmatrix} \begin{bmatrix} \sqrt{2} & 0 & 0 \\ \sqrt{1/2} & \sqrt{3/2} & 0 \\ 0 & -\sqrt{2/3} & \sqrt{1/3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{array}{lll}
L_{11}^{-1}\sqrt{2} = 1 & & L_{11}^{-1} = \sqrt{1/2} \\
L_{22}^{-1}\sqrt{3/2} = 1 & & L_{22}^{-1} = \sqrt{2/3} \\
L_{21}^{-1}\sqrt{2} - L_{22}^{-1}\sqrt{1/2} = 0 & L_{21}^{-1}\sqrt{2} - \sqrt{2/3}\sqrt{1/2} = 0 & L_{21}^{-1} = \sqrt{1/6} \\
L_{33}^{-1}\sqrt{1/3} = 1 & & L_{33}^{-1} = \sqrt{3} \\
L_{32}^{-1}\sqrt{3/2} - L_{33}^{-1}\sqrt{2/3} = 0 & L_{32}^{-1}\sqrt{3/2} - \sqrt{3}\sqrt{2/3} = 0 & L_{32}^{-1} = \sqrt{4/3} \\
L_{31}^{-1}\sqrt{2} - L_{32}^{-1}\sqrt{1/2} = 0 & L_{31}^{-1}\sqrt{2} - \sqrt{4/3}\sqrt{1/2} = 0 & L_{31}^{-1} = \sqrt{1/3}
\end{array}$$

$$\mathbf{L}^{-1} = \begin{bmatrix} \sqrt{1/2} & 0 & 0 \\ \sqrt{1/6} & \sqrt{2/3} & 0 \\ \sqrt{1/3} & \sqrt{4/3} & \sqrt{3} \end{bmatrix}$$

$$\begin{aligned}
\mathbf{H} &= \begin{bmatrix} \sqrt{1/2} & 0 & 0 \\ \sqrt{1/6} & \sqrt{2/3} & 0 \\ \sqrt{1/3} & \sqrt{4/3} & \sqrt{3} \end{bmatrix} \begin{bmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} \sqrt{1/2} & \sqrt{1/6} & \sqrt{1/3} \\ 0 & \sqrt{2/3} & \sqrt{4/3} \\ 0 & 0 & \sqrt{3} \end{bmatrix} \\
&= \begin{bmatrix} 2 & \sqrt{1/3} & \sqrt{2/3} \\ \sqrt{1/3} & 8/3 & 5\sqrt{2}/3 \\ \sqrt{2/3} & 5\sqrt{2}/3 & 40/3 \end{bmatrix} = \begin{bmatrix} 2.0000 & 0.5774 & 0.8165 \\ 0.5774 & 2.6667 & 2.3570 \\ 0.8165 & 2.3570 & 13.3333 \end{bmatrix} \blacktriangleleft
\end{aligned}$$

Problem 3

$$\begin{aligned}
\mathbf{A}^* &= \mathbf{A} - s\mathbf{B} = \begin{bmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix} - 2.5 \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \\
&= \begin{bmatrix} -1.0 & 1.5 & 0 \\ 1.5 & -1.0 & 1.5 \\ 0 & 1.5 & 1.5 \end{bmatrix}
\end{aligned}$$

<u>First iteration</u>

$$\mathbf{Bv} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}$$

Solve $\mathbf{A}^*\mathbf{z} = \mathbf{Bv}$:

$$\begin{bmatrix} -1.0 & 1.5 & 0 \\ 1.5 & -1.0 & 1.5 \\ 0 & 1.5 & 1.5 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix} \qquad \mathbf{z} = \begin{bmatrix} -14 \\ -8 \\ 8 \end{bmatrix}$$

$$v = \frac{\mathbf{z}}{|\mathbf{z}|} = \begin{bmatrix} -14 \\ -8 \\ 8 \end{bmatrix} \frac{1}{18} = \begin{bmatrix} -0.7778 \\ -0.4444 \\ 0.4444 \end{bmatrix}$$

$$\lambda = s - \frac{1}{|\mathbf{z}|} = 2.5 - \frac{1}{18} = 2.4444$$

<u>Second iteration</u>

$$\mathbf{Bv} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} -0.7778 \\ -0.4444 \\ 0.4444 \end{bmatrix} = \begin{bmatrix} -1.1112 \\ -0.5554 \\ 0.8888 \end{bmatrix}$$

Solve $\mathbf{A}^*\mathbf{z} = \mathbf{Bv}$:

$$\begin{bmatrix} -1.0 & 1.5 & 0 \\ 1.5 & -1.0 & 1.5 \\ 0 & 1.5 & 1.5 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} -1.1112 \\ -0.5554 \\ 0.8888 \end{bmatrix} \qquad \mathbf{z} = \begin{bmatrix} 19.778 \\ 12.444 \\ -11.852 \end{bmatrix}$$

$$v = \frac{\mathbf{z}}{|\mathbf{z}|} = \begin{bmatrix} 19.778 \\ 12.444 \\ -11.852 \end{bmatrix} \frac{1}{26.20} = \begin{bmatrix} 0.7549 \\ 0.4750 \\ -0.4524 \end{bmatrix}$$

$$\lambda = s - \frac{1}{|\mathbf{z}|} = 2.5 - \frac{1}{26.20} = 2.4618 \blacktriangleleft$$

Problem 4

$$\mathbf{S} = \begin{bmatrix} 150 & -60 & 0 \\ -60 & 120 & 0 \\ 0 & 0 & 80 \end{bmatrix} \text{MPa}$$

The characteristic equation is

$$|\mathbf{S} - \lambda\mathbf{I}| = 0 \qquad \begin{vmatrix} 150-\lambda & -60 & 0 \\ -60 & 120-\lambda & 0 \\ 0 & 0 & 80-\lambda \end{vmatrix} = 0$$

$$\begin{aligned} (80-\lambda)\left[(150-\lambda)(120-\lambda) - 60^2\right] &= 0 \\ (80-\lambda)(14400 - 270\lambda + \lambda^2) &= 0 \end{aligned}$$

The solution (principal stresses) is

$$\lambda_1 = 73.15 \text{ MPa} \qquad \lambda_2 = 80 \text{ MPa} \qquad \lambda_3 = 196.85 \text{ MPa} \blacktriangleleft$$

Problem 5

$$\begin{aligned} kL(\theta_2-\theta_1)-mg\theta_1 &= mL\ddot{\theta}_1 \\ -kL(\theta_2-\theta_1)-2mg\theta_2 &= 2mL\ddot{\theta}_2 \end{aligned}$$

Substituting $\theta_i = x_i \sin\omega t$ we get

$$\begin{aligned} [kL(x_2-x_1)-mgx_1]\sin\omega t &= -\omega^2 mLx_1\sin\omega t \\ [-kL(x_2-x_1)-2mgx_2]\sin\omega t &= -2\omega^2 mLx_2\sin\omega t \end{aligned}$$

$$\begin{bmatrix} kL+mg & -kL \\ -kL & kL+2mg \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \omega^2 mL\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\begin{bmatrix} 1+mg/(kL) & -1 \\ -1 & 1+2mg/(kL) \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \omega^2\frac{m}{k}\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Using

$$\frac{mg}{kL} = \frac{0.25(9.80665)}{20(0.75)} = 0.16344 \qquad \lambda = \omega^2\frac{m}{k}$$

the equations of motion become

$$\begin{bmatrix} 1.16344-\lambda & -1 \\ -1 & 1.32688-2\lambda \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{a}$$

The characteristic equation is

$$\begin{aligned} \begin{vmatrix} 1.16344-\lambda & -1 \\ -1 & 1.32688-2\lambda \end{vmatrix} &= 0 \\ (1.16344-\lambda)(1.32688-2\lambda)-1 &= 0 \\ 0.54375-3.65376\lambda+2\lambda^2 &= 0 \end{aligned}$$

$$\lambda_1 = 0.16344 \qquad \lambda_2 = 1.66344$$

The circular frequencies are

$$\begin{aligned} \omega_1 &= \sqrt{\lambda_1\frac{k}{m}} = \sqrt{0.16344\frac{20}{0.25}} = 3.616 \text{ rad/s} \blacktriangleleft \\ \omega_2 &= \sqrt{\lambda_2\frac{k}{m}} = \sqrt{1.66344\frac{20}{0.25}} = 11.536 \text{ rad/s} \blacktriangleleft \end{aligned}$$

Substituting $\lambda = \lambda_1$, Eq. (a) becomes

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

yielding $x_1 = x_2$. Hence the (normalized) relative amplitudes of the first mode are

$$x_1 = x_2 = \frac{1}{\sqrt{2}} \blacktriangleleft$$

With $\lambda = \lambda_2$, Eq. (a) is

$$\begin{bmatrix} -0.5 & -1 \\ -1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

which gives $x_1 = -2x_2$, so that the relative amplitudes of the second mode are

$$x_1 = \frac{1}{\sqrt{5}} \qquad x_2 = -\frac{2}{\sqrt{5}} \blacktriangleleft$$

Problem 6

$$\begin{aligned} 3i_1 - i_2 - i_3 &= -LC\frac{d^2 i_1}{dt^2} \\ -i_1 + i_2 &= -LC\frac{d^2 i_2}{dt^2} \\ -i_1 + i_3 &= -LC\frac{d^2 i_3}{dt^2} \end{aligned}$$

Let $i_k = x_k \sin \omega t$. Then the equations become (after cancelling $\sin \omega t$)

$$\begin{aligned} 3x_1 - x_2 - x_3 &= \omega^2 LC x_1 \\ -x_1 + x_2 &= \omega^2 LC x_2 \\ -x_1 + x_3 &= \omega^2 LC x_3 \end{aligned}$$

or

$$\begin{bmatrix} 3 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{a}$$

where $\lambda = \omega^2 LC$. The characteristic equation is

$$\begin{vmatrix} 3-\lambda & -1 & -1 \\ -1 & 1-\lambda & 0 \\ -1 & 0 & 1-\lambda \end{vmatrix} = 0$$

$$\begin{aligned} 1 - 5\lambda + 5\lambda^2 - \lambda^3 &= 0 \\ (1-\lambda)\left(\lambda^2 - 4\lambda + 1\right) &= 0 \end{aligned}$$

$$\lambda_1 = 0.2679 \qquad \lambda_2 = 1 \qquad \lambda_3 = 3.7321$$

The circular frequencies are

$$\begin{aligned}
\omega_1 &= \sqrt{\frac{\lambda_1}{LC}} = \sqrt{\frac{0.2679}{LC}} = \frac{0.5176}{\sqrt{LC}} \\
\omega_2 &= \sqrt{\frac{\lambda_2}{LC}} = \sqrt{\frac{1}{LC}} = \frac{1}{\sqrt{LC}} \\
\omega_3 &= \sqrt{\frac{\lambda_3}{LC}} = \sqrt{\frac{3.7321}{LC}} = \frac{1.9319}{\sqrt{LC}}
\end{aligned}$$

First mode: substitute λ_1 into Eqs. (a):

$$\begin{bmatrix} 2.7321 & -1 & -1 \\ -1 & 0.7321 & 0 \\ -1 & 0 & 0.7321 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Choosing $x_1 = 1$, the second and third equations yield

$$\begin{aligned}
-1 + 0.7321x_2 &= 0 \qquad x_2 = 1.3659 \\
-1 + 0.7321x_3 &= 0 \qquad x_3 = 1.3659
\end{aligned}$$

After normalizing we have

$$\mathbf{x} = \begin{bmatrix} 0.4597 & 0.6280 & 0.6280 \end{bmatrix}^T \blacktriangleleft$$

Second mode: substituting λ_2 into Eqs. (a) we get

$$\begin{bmatrix} 2 & -1 & -1 \\ -1 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$x_1 = 0 \qquad x_2 = x_3$$

$$\mathbf{x} = \begin{bmatrix} 0 & 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}^T \blacktriangleleft$$

Third mode: with $\lambda = \lambda_3$ Eqs. (a) are

$$\begin{bmatrix} -0.7321 & -1 & -1 \\ -1 & -2.7321 & 0 \\ -1 & 0 & -2.7321 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Choosing $x_1 = 1$, the second and third equations yield

$$\begin{aligned}
-1 - 2.7321x_2 &= 0 \qquad x_2 = -0.3660 \\
-1 - 2.7321x3 &= 0 \qquad x_3 = -0.3660
\end{aligned}$$

After normalizing we have

$$\mathbf{x} = \begin{bmatrix} 0.8881 & -0.3250 & -0.3250 \end{bmatrix}^T \blacktriangleleft$$

Problem 7

$$\mathbf{A} = \begin{bmatrix} 4 & -1 & 0 & 1 \\ -1 & 6 & -2 & 0 \\ 0 & -2 & 3 & 2 \\ 1 & 0 & 2 & 4 \end{bmatrix}$$

From Eqs. (9.15)-(9.17) and (9.19):

$$\begin{aligned}
\phi &= -\frac{A_{11} - A_{44}}{2A_{14}} = -\frac{4-4}{2(1)} = 0 \\
t &= \frac{\operatorname{sgn}(\phi)}{|\phi| + \sqrt{\phi^2+1}} = \frac{-1}{|0| + \sqrt{0^2+1}} = -1 \\
c &= \frac{1}{\sqrt{1+t^2}} = \frac{1}{\sqrt{1+(-1)^2}} = 0.7071 \qquad s = tc = -0.7071 \\
\tau &= \frac{s}{1+c} = \frac{-0.7071}{1+0.7071} = -0.4142
\end{aligned}$$

From Eq. (9.18):

$$\begin{aligned}
A_{11}^* &= A_{11} - tA_{14} = 4 - (-1)(1) = 5 \\
A_{44}^* &= A_{44} + tA_{14} = 4 + (-1)(1) = 3 \\
A_{12}^* &= A_{12} - s(A_{42} + \tau A_{12}) = -1 - (-0.7071)\left[0 + (-0.4142)(-1)\right] \\
&= -0.7071 \\
A_{13}^* &= A_{13} - s(A_{43} + \tau A_{13}) = 0 - (-0.7071)\left[2 + (-0.4142)(0)\right] = 1.4142 \\
A_{42}^* &= A_{42} + s(A_{12} - \tau A_{42}) = 0 + (-0.7071)\left[-1 - (-0.4142)(0)\right] = 0.7071 \\
A_{43}^* &= A_{43} + s(A_{13} - \tau A_{43}) = 2 + (-0.7071)\left[0 - (-0.4142)(2)\right] = 1.4142
\end{aligned}$$

$$\mathbf{A}^* = \begin{bmatrix} 5.0000 & -0.7071 & 1.4142 & 0 \\ -0.7071 & 6.0000 & -2.0000 & 0.7071 \\ 1.4142 & -2.0000 & 3.0000 & 1.4142 \\ 0 & 0.7071 & 1.4142 & 3.0000 \end{bmatrix} \blacktriangleleft$$

Problem 8

```
## problem9_1_8
from numpy import array
from jacobi import *

a = array([[4, -1, -2],[-1, 3, 3],[-2, 3, 1]])*1.0
lam,x = jacobi(a)
```

```
print("Eigenvalues:\n", lam)
print("Eigenvectors:\n",x)
input("\nPress return to exit")

Eigenvalues:
[ 6.69558869  2.69161093 -1.38719961]
Eigenvectors:
[[ 0.61015618  0.76356158  0.21138388]
 [-0.59228156  0.61680524 -0.51841476]
 [-0.52622428  0.19111519  0.82859097]]
```

Problem 9

```
## problem9_1_9
from numpy import array
from jacobi import *

a = array([[ 4, -2,  1, -1],  \
           [-2,  4, -2,  1],  \
           [ 1, -2,  4, -2],  \
           [-1,  1, -2,  4]])*1.0
lam,x = jacobi(a)
print("Eigenvalues:\n", lam)
print("Eigenvectors:\n",x)
input("\nPress return to exit")

Eigenvalues:
[ 8.54138127  1.38196601  2.45861873  3.61803399]
Eigenvectors:
[[ 0.45705607  0.37174803 -0.5395366   0.60150095]
 [-0.5395366   0.60150096 -0.45705607 -0.37174803]
 [ 0.5395366   0.60150096  0.45705607 -0.37174803]
 [-0.45705607  0.37174803  0.5395366   0.60150096]]
```

Problem 10

```
## problem9_1_10
from numpy import zeros,array,dot
from LUdecomp import *
from math import sqrt
from random import random

def powerMethod(a,tol=1.0e-6):
    n = len(a)
    v = zeros(n)
    for i in range(n): v[i] = random()
    v =v/sqrt(dot(v,v))
    for i in range(25):
        z = dot(a,v)
        zMag = sqrt(dot(z,z))
        z = z/zMag
        if sqrt(dot(v - z,v - z)) < tol: break
        v = z
    if dot(v,z) > 0.0: lam = zMag
    else: lam = -zMag
    return lam,z

a = array([[ 4, -2,  1, -1],  \
           [-2,  4, -2,  1],  \
           [ 1, -2,  4, -2],  \
           [-1,  1, -2,  4]])*1.0
lam,x = powerMethod(a)
print("Eigenvalue =",lam)
print("Eigenvector =",x)
input("\nPress return to exit")

Eigenvalue = 8.541381265141277
Eigenvector = [-0.45705568  0.53953638 -0.53953685  0.45705645]
```

Problem 11

```
## problem9_1_11
from numpy import array
from inversePower import *
```

```
a = array([[ 4, -2,  1, -1],  \
           [-2,  4, -2,  1],  \
           [ 1, -2,  4, -2],  \
           [-1,  1, -2,  4]])*1.0
lam,x = inversePower(a,0.0)
print("Eigenvalue =", lam)
print("Eigenvector =",x)
input("\nPress return to exit")

Eigenvalue = 1.3819660112520518
Eigenvector = [ 0.37174865  0.60150148  0.60150043  0.37174742]
```

Problem 12

```
##  problem9_1_12
from numpy import array,dot
from jacobi import *
from stdForm import *
from math import sqrt

a = array([[1.4, 0.8, 0.4],  \
           [0.8, 6.6, 0.8],  \
           [0.4, 0.8, 5.0]])
b = array([[ 0.4, -0.1,  0.0],  \
           [-0.1,  0.4, -0.1],  \
           [ 0.0, -0.1,  0.4]])
h,t = stdForm(a,b)        # Transform into std. form
lam,z = jacobi(h,1.0e-12) # Solve by Jacobi's method
x = dot(t,z)              # Recover eigenvals. of orig. prob.
for i in range(len(x)):   # Normalize eigenvectors
    xMag = sqrt(dot(x[:,i],x[:,i]))
    x[:,i] = x[:,i]/xMag
print("Eigenvalues:\n",lam)
print("Eigenvectors:\n",x)
input("Press return to exit")

Eigenvalues:
[  2.92765173  25.59804434   9.90287536]
Eigenvectors:
[[ 0.98102277  0.32245473 -0.18709892]
 [-0.18761435  0.78544019 -0.46142271]
 [-0.04894062  0.52830546  0.86722724]]
```

Problem 13

```
## problem9_1_13
from numpy import array,dot
from inversePower import *
from stdForm import *
from math import sqrt

a = array([[1.4, 0.8, 0.4],  \
           [0.8, 6.6, 0.8],  \
           [0.4, 0.8, 5.0]])
b = array([[ 0.4, -0.1,  0.0],  \
           [-0.1,  0.4, -0.1],  \
           [ 0.0, -0.1,  0.4]])
h,t = stdForm(a,b)            # Transform into std. form
lam,z = inversePower(h,0.0) # Solve by inverse power mthd.
x = dot(t,z)                  # Recover eigenvalue of orig. prob.
x = x/sqrt(dot(x,x))          # Normalize eigenvector
print("Eigenvalue =",lam)
print("Eigenvector =",x)
input("Press return to exit")

Eigenvalue = 2.9276517279146326
Eigenvector = [ 0.98102278 -0.18761427 -0.04894077]
```

Problem 14

```
## problem9_1_14
from numpy import array,dot
from jacobi import *
from sortJacobi import *
from math import sqrt

a = array([[ 11.0, 2.0,  3.0,  1.0,  4.0, 2.0],  \
           [  2.0, 9.0,  3.0,  5.0,  2.0, 1.0],  \
           [  3.0, 3.0, 15.0,  4.0,  3.0, 2.0],  \
           [  1.0, 5.0,  4.0, 12.0,  4.0, 3.0],  \
           [  4.0, 2.0,  3.0,  4.0, 17.0, 5.0],  \
           [  2.0, 1.0,  2.0,  3.0,  5.0, 8.0]])
lam,x = jacobi(a)
sortJacobi(lam,x)
```

```
print("Eigenvalues:")
for i in range (len(a)): print("{:8.4f}".format(lam[i]),end=" ")
print("\n\nEigenvectors:")
for i in range(len(a)):
    for j in range(len(a)):
        print("{:8.4f}".format(x[i,j]),end=" ")
    print()
input ("Press return to exit")
```

```
Eigenvalues:
  4.4636   5.9889   8.7119  10.9767  13.8675  27.9913

Eigenvectors:
 -0.2380   0.1537   0.7247  -0.5314   0.1213   0.3121
  0.6234  -0.3858   0.4368   0.3089  -0.3001   0.2938
  0.0251  -0.0554  -0.4416  -0.4987  -0.5901   0.4521
 -0.5653   0.2082   0.0785   0.6054  -0.2871   0.4266
 -0.0416  -0.4034  -0.2640   0.0328   0.6522   0.5826
  0.4825   0.7864  -0.1147   0.0769   0.1981   0.3009
```

Problem 15

Because $\mathbf{B}$ is not positive definite, the eigenvalue problem $\mathbf{Ax} = \lambda\mathbf{Bx}$ cannot be transformed into the standard form, since Choleski's decomposition $\mathbf{B} = \mathbf{LL}^T$ would fail. We can, however, interchange the roles of $\mathbf{A}$ and $\mathbf{B}$ by dividing both sides of the problem by λ. The result is the eigenvalue problem $\mathbf{Bx} = (1/\lambda)\mathbf{Ax}$. As $\mathbf{A}$ is positive definite, we have no trouble decomposing it

```
## problem9_1_15
from numpy import array
from stdForm import *
from jacobi import *

a = array([[ 6.0, -4.0,  1.0,  0.0], \
           [-4.0,  6.0, -4.0,  1.0], \
           [ 1.0, -4.0,  6.0, -4.0], \
           [ 0.0,  1.0, -4.0,  7.0]])
b = array([[ 1.0, -2.0,  3.0, -1.0], \
           [-2.0,  6.0, -2.0,  3.0], \
           [ 3.0, -2.0,  6.0, -2.0], \
           [-1.0,  3.0, -2.0,  9.0]])
h,q = stdForm(b,a)
```

```
lam,x = jacobi(h)
print("Eigenvalues:\n",1.0/lam)
input("\nPress return to exit")

Eigenvalues:
[-7.29608125  1.31712064  0.92887446  0.10397837]
```

Problem 16

(a)

Here the eigenvalue problem is $\mathbf{Ax} = \lambda\mathbf{Bx}$, where $\mathbf{B}$ is a diagonal matrix. Using the notation in Eq. (9.25), the diagonal terms of $\mathbf{B}$ are $\beta_1 = \beta_2 = \cdots = \beta_{n-1} = 1$, $\beta_n = 1/2$. Equation (9.26b) is

$$H_{ij} = \frac{A_{ij}}{\sqrt{\beta_i \beta_j}}$$

The differences between $\mathbf{H}$ and $\mathbf{A}$ are confined to the last row and column:

$$\begin{aligned} H_{in} &= H_{ni} = \sqrt{2}A_{in}, \quad i = 1, 2, \ldots, n-1 \\ H_{nn} &= 2A_{nn} \end{aligned}$$

Thus

$$\mathbf{H} = \begin{bmatrix} 7 & -4 & 1 & 0 & 0 & \cdots & 0 \\ -4 & 6 & -4 & 1 & 0 & \cdots & 0 \\ 1 & -4 & 6 & -4 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & -4 & 6 & -4 & \sqrt{2} \\ 0 & \cdots & 0 & 1 & -4 & 5 & -2\sqrt{2} \\ 0 & \cdots & 0 & 0 & \sqrt{2} & -2\sqrt{2} & 2 \end{bmatrix} \blacktriangleleft$$

The transformation is

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & \sqrt{2} \end{bmatrix} \mathbf{z} \quad \blacktriangleleft$$

(b)

```
## problem9_1_16
from numpy import zeros
```

```
from jacobi import *
+from sortJacobi import *
from math import sqrt

n = 10
a = zeros((n,n))
for i in range(n-2):
    a[i,i] = 6.0
    a[i,i+1] = -4.0
    a[i,i+2] = 1.0
a[0,0] = 7.0
a[n-2,n-2] = 5.0
a[n-2,n-1] = -2.0*sqrt(2.0)
a[n-1,n-1] = 2.0
a[n-3,n-1] = sqrt(2.0)
lam,x = jacobi(a)
sortJacobi(lam,x)
x[n-1,0:2] = sqrt(2)*x[n-1,0:2]
print("Circular frequencies (units of sqrt(EI/gamma)/L**2):")
print(sqrt(lam[0])*n**2,sqrt(lam[1])*n**2)
print("Eigenvectors:")
print(x[0:n,0])
print(x[0:n,1])
input("\nPress return to exit")
```

```
Circular frequencies (units of sqrt(EI/gamma)/L**2):
3.4865965640051746 21.13353824617363
Eigenvectors:
[ 0.01096236  0.04084113  0.08664132  0.14541759  0.21432992
  0.29071506  0.37217034  0.45664644  0.54254651  0.6288288 ]
[ 0.06423825  0.19506603  0.33346537  0.42913051  0.44664912
  0.36977497  0.20221035 -0.03582729 -0.31508929 -0.60792711]
```

$$\omega_1 = 3.487\sqrt{\frac{EI}{\gamma}}\frac{1}{L^2} \blacktriangleleft \qquad \omega_2 = 21.134\sqrt{\frac{EI}{\gamma}}\frac{1}{L^2} \blacktriangleleft$$

Problem 17

The eigenvalue problem is $A\mathbf{u} = \lambda\mathbf{B}\mathbf{u}$, where $\mathbf{B}$ is a diagonal matrix and $\mathbf{A}$ is tridiagonal: $\mathbf{A} = [\mathbf{c}\backslash\mathbf{d}\backslash\mathbf{e}]$.

```
## problem9_1_17
```

```
from numpy import dot,array,zeros,ones
from LUdecomp3 import *
from math import sqrt
from random import random

def inversePower3(d,c,b,tol=1.0e-6):
    n = len(d)
    e = c.copy()
    v = zeros(n)
    z = zeros(n)
    for i in range(n):           # Seed [v] with random numbers
        v[i] = random()
    v =v/sqrt(dot(v,v))          # Normalize [v]
    LUdecomp3(c,d,e)             # Decompose [A]
    for i in range(30):          # Begin iterations
        for k in range(n):       # Form [B][v]
            z[k] = b[k]*v[k]
        LUsolve3(c,d,e,z)        # Solve [A][z] = [B][v]
        zMag = sqrt(dot(z,z))    # Normalize [z]
        z = z/zMag
        if sqrt(dot(v - z,v - z)) < tol: break
        v = z
    if dot(v,z) > 0.0:           # Detect change in sign of [v]
        lam = 1.0/zMag
    else: lam = -1.0/zMag
    return lam

n = 10             # n must be even
d = ones(n)*2.0
d[n-1] = 1.0
c = ones(n-1)*(-1.0)
b = ones(n)
b[n/2-1] = 1.0/1.5
b[n/2:n-1] = 0.5
b[n-1] = 0.25
lam = inversePower3(d,c,b)
print("PL^2/EI =",lam*(2.0*n)**2)
input("\nPress return to enter")

PL^2/EI = 16.71408681469039
```

Problem 18

$$\begin{bmatrix} 6 & 5 & 3 \\ 3 & 3 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \frac{P}{kL} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

The problem can be transformed into standard form by the operations

$$\begin{aligned} \text{row 1} &\leftarrow \text{row 1} - \text{row 2} \\ \text{row 3} &\leftarrow \text{row 2} - \text{row 3} \end{aligned}$$

This yields

$$\begin{bmatrix} 3 & 2 & 1 \\ 2 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \frac{P}{kL} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

```
## problem9_1_18
from numpy import array
from inversePower import *

a = array([[3, 2, 1],[2, 2, 1],[1, 1, 1]])*1.0
lam,x = inversePower(a,0.0)
print("Buckling load =",lam,"kL")
print("Buckling mode =",x,"rad")
input("Press return to exit")
```

```
Buckling load = 0.3079785283702163 kL
Buckling mode = [-0.32798574  0.73697648 -0.59100848] rad
```

Problem 19

$$\begin{aligned} k\left(-2u_1 + u_2\right) &= m\ddot{u}_1 \\ k(u_1 - 2u_2 + u_3) &= 3m\ddot{u}_2 \\ k(u_2 - 2u_3) &= 2m\ddot{u}_3 \end{aligned}$$

Substituting $u_i = x_i \sin \omega t$, the equations of motion become (after cancelling $\sin \omega t$)

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \omega^2 \frac{m}{k} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

```
## problem 9_1_19
from jacobi import *
from numpy import array,dot,sqrt

m = array([1, 3, 2])*1.0    # Specify masses
m = sqrt(m)
a = array([[2, -1, 0],[-1, 2, -1],[0, -1, 2]])*1.0
for i in range(3):          # Form matrix [H} of std. problem
    for j in range(3):
        a[i,j] = a[i,j]/(m[i]*m[j])
lam,x = jacobi(a)           # Solve by Jacobi's method
for i in range(3):          # Recover eigenvectors of [A]
    x[i] = x[i]/m[i]
for i in range(3):          # Normalize eigenvectors
    x[:,i] = x[:,i]/sqrt(dot(x[:,i],x[:,i]))
print("Circular frequencies (units of sqrt(k/m)):")
for i in range(3): print("{:10.5f}".format(sqrt(lam[i])),end=" ")
print("\n\nRelative displacements:")
for i in range(3):
    for j in range(3):
        print("{:10.5f}".format(x[i,j]),end=" ")
    print()
input("\n Press return to exit")
```

```
Circular frequencies (units of sqrt(k/m)):
   1.49430    0.50281    1.08670

Relative displacements:
   0.96983    0.42955   -0.38362
  -0.22590    0.75050   -0.31422
   0.09161    0.50222    0.86839
```

Problem 20

$$\begin{aligned}
L\frac{d^2 i_1}{dt^2} + \frac{1}{C}i_1 + \frac{2}{C}(i_1 - i_2) &= 0 \\
L\frac{d^2 i_2}{dt^2} + \frac{2}{C}(i_2 - i_1) + \frac{3}{C}(i_2 - i_3) &= 0 \\
L\frac{d^2 i_3}{dt^2} + \frac{3}{C}(i_3 - i_2) + \frac{4}{C}(i_3 - i_4) &= 0 \\
L\frac{d^2 i_4}{dt^2} + \frac{4}{C}(i_4 - i_3) + \frac{5}{C}i_4 &= 0
\end{aligned}$$

Substituting $i_k = x_k \sin \omega t$, we get (after cancelling $\sin \omega t$)

$$\begin{bmatrix} 3 & -2 & 0 & 0 \\ -2 & 5 & -3 & 0 \\ 0 & -3 & 7 & -4 \\ 0 & 0 & -4 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \omega^2 LC \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

```
## problem9_1_20
from jacobi import *
from numpy import array,sqrt

a = array([[ 3, -2,  0,  0], \
           [-2,  5, -3,  0], \
           [ 0, -3,  7, -4], \
           [ 0,  0, -4,  9]])*1.0
lam,x = jacobi(a)
print("Circular frequencies (units of 1/sqrt(LC)):")
print(sqrt(lam))
input("\n Press return to exit")
```

```
Circular frequencies (units of 1/sqrt(LC)):
[ 0.95139975  1.84115409  2.65790497  3.55535249]
```

Problem 21

$$\begin{aligned} L\frac{d^2 i_1}{dt^2} + L\left(\frac{d^2 i_1}{dt^2} - \frac{d^2 i_2}{dt^2}\right) + \frac{1}{C} i_1 &= 0 \\ L\left(\frac{d^2 i_2}{dt^2} - \frac{d^2 i_1}{dt^2}\right) + L\left(\frac{d^2 i_2}{dt^2} - \frac{d^2 i_3}{dt^2}\right) + \frac{2}{C} &= 0 \\ L\left(\frac{d^2 i_3}{dt^2} - \frac{d^2 i_2}{dt^2}\right) + L\left(\frac{d^2 i_3}{dt^2} - \frac{d^2 i_4}{dt^2}\right) + \frac{3}{C} i_3 &= 0 \\ L\left(\frac{d^2 i_4}{dt^2} - \frac{d^2 i_3}{dt^2}\right) + L\frac{d^2 i_4}{dt^2} + \frac{4}{C} i_4 &= 0 \end{aligned}$$

After substituting $i_k = x_k \sin \omega t$, we get

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \omega^2 LC \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

This can be written as

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \lambda \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

where $\lambda = (\omega^2 LC)^{-1}$.

```
## problem 9_1_21
from jacobi import *
from numpy import array,sqrt

beta = array([1, 2, 3, 4])*1.0  # Diagonal terms of [B}
beta = sqrt(beta)
a = array([[ 2, -1,  0,  0], \
           [-1,  2, -1,  0], \
           [ 0, -1,  2, -1], \
           [ 0,  0, -1,  2]])*1.0
for i in range(4):              # Form matrix [H} of std. problem
    for j in range(4):
        a[i,j] = a[i,j]/(beta[i]*beta[j])
lam,x = jacobi(a)               # Solve by Jacobi's method
print("Circular frequencies (units of 1/sqrt(CL)):")
print(1.0/sqrt(lam))
input("\n Press return to exit")
```

```
Circular frequencies (units of 1/sqrt(CL)):
[ 0.64711605  2.61701309  0.96278991  1.34369599]
```

Problem 22

```
##problem9_1_22
from numpy import array,transpose,diagonal,dot
from choleski import *
from math import sqrt

def LRmethod(a):
    n = len(a)
    lamMagOld = sqrt(dot(diagonal(a),diagonal(a)))
    for i in range(50):
        choleski(a)
        for i in range(n-1):    # choleski(a) does not zero
            a[i,i+1:n] = 0.0    #    upper half of [A]
        a = dot(transpose(a),a)
        lamMag = sqrt(dot(diagonal(a),diagonal(a)))
        if abs(lamMag - lamMagOld) < 1.0e-6: return diagonal(a)
        else: lamMagOld = lamMag
    print("LR method did not converge")

a = array([[4.0, 3.0, 1.0], \
           [3.0, 4.0, 2.0], \
           [1.0, 2.0, 3.0]])
lam = LRmethod(a)
print("Eigenvalues:\n",lam)
input("Press return to exit")

Eigenvalues:
[ 7.92692778  2.3856316   0.68744062]
```

PROBLEM SET 9.2

Problem 1

(a)

$$\mathbf{A} = \begin{bmatrix} 10 & 4 & -1 \\ 4 & 2 & 3 \\ -1 & 3 & 6 \end{bmatrix}$$

$$\mathbf{a} = \begin{bmatrix} 10 \\ 2 \\ 6 \end{bmatrix} \qquad \mathbf{r} = \begin{bmatrix} 4+1 \\ 4+3 \\ 1+3 \end{bmatrix} = \begin{bmatrix} 5 \\ 7 \\ 4 \end{bmatrix}$$

$$\begin{aligned} \lambda_{\min} &\geq \min_i(a_i - r_i) = 2 - 7 = -5 \ \blacktriangleleft \\ \lambda_{\max} &\leq \max_i(a_i + r_i) = 10 + 5 = 15 \ \blacktriangleleft \end{aligned}$$

(The actual eigenvalues are $\lambda_{\min} = -1.066$ and $\lambda_{\max} = 11.667$).

(b)

$$\mathbf{B} = \begin{bmatrix} 4 & 2 & -2 \\ 2 & 5 & 3 \\ -2 & 3 & 4 \end{bmatrix}$$

$$\mathbf{a} = \begin{bmatrix} 4 \\ 5 \\ 4 \end{bmatrix} \qquad \mathbf{r} = \begin{bmatrix} 2+2 \\ 2+3 \\ 2+3 \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \\ 5 \end{bmatrix}$$

$$\begin{aligned} \lambda_{\min} &\geq \min_i(a_i - r_i) = 4 - 5 = -1 \ \blacktriangleleft \\ \lambda_{\max} &\leq \max_i(a_i + r_i) = 5 + 5 = 10 \ \blacktriangleleft \end{aligned}$$

(The actual eigenvalues are $\lambda_{\min} = -0.365$ and $\lambda_{\max} = 7.565$).

Problem 2

$$P_4(\lambda) = |\mathbf{A} - \lambda\mathbf{I}| = \begin{vmatrix} 5-\lambda & -2 & 0 & 0 \\ -2 & 4-\lambda & -1 & 0 \\ 0 & -1 & 4-\lambda & -2 \\ 0 & 0 & -2 & 5-\lambda \end{vmatrix}$$

Using $\lambda = 2$:

$$\begin{aligned}
P_0(2) &= 1 \\
P_1(2) &= d_1 - \lambda = 5 - 2 = 3 \\
P_2(2) &= (d_2 - \lambda)P_1(2) - c_1^2 P_0(2) = (4-2)3 - (-2)^2 1 = 2 \\
P_3(2) &= (d_3 - \lambda)P_2(2) - c_2^2 P_1(2) = (4-2)2 - (-1)^2 3 = 1 \\
P_4(2) &= (d_4 - \lambda)P_3(2) - c_3^2 P_2(2) = (5-2)1 - (-2)^2 2 = -5
\end{aligned}$$

There is one sign change in this sequence. Therefore, one eigenvalue is less than 2.

Using $\lambda = 4$:

$$\begin{aligned}
P_0(4) &= 1 \\
P_1(4) &= d_1 - \lambda = 5 - 4 = 1 \\
P_2(4) &= (d_2 - \lambda)P_1(4) - c_1^2 P_0(4) = (4-4)3 - (-2)^2 1 = -4 \\
P_3(4) &= (d_3 - \lambda)P_2(4) - c_2^2 P_1(4) = (4-4)(-4) - (-1)^2 1 = -1 \\
P_4(4) &= (d_4 - \lambda)P_3(4) - c_3^2 P_2(4) = (5-4)(-1) - (-2)^2(-4) = 15
\end{aligned}$$

Since there are 2 sign changes in this sequence, there are 2 eigenvalues are smaller than 4.

It follows that there is one eigenvalue between 2 and 4.

Problem 3

$$\mathbf{A} = \begin{bmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix}$$

Find global bounds with Gerschgorin's theorem:

$$\mathbf{a} = \begin{bmatrix} 4 \\ 4 \\ 4 \end{bmatrix} \qquad \mathbf{r} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

$$\begin{aligned}
\lambda_{\min} &\geq \min_i (a_i - r_i) = 4 - 2 = 2 \\
\lambda_{\max} &\leq \max_i (a_i + r_i) = 4 + 2 = 6
\end{aligned}$$

Use Sturm sequence to determine intermediate bounds:

With $\lambda = 4$:

$$\begin{aligned}
P_0(4) &= 1 \\
P_1(4) &= d_1 - \lambda = 4 - 4 = 0 \\
P_2(4) &= (d_2 - \lambda)P_1(4) - c_1^2 P_0(4) = (4-4)(0) - (-1)^2 1 = -1 \\
P_3(4) &= (d_3 - \lambda)P_2(4) - c_2^2 P_1(4) = (4-3)(-1) - (-1)^2 0 = 0
\end{aligned}$$

The zero result indicates that $\lambda = 4$ is an eigenvalue.
With $\lambda = 3$:

$$\begin{aligned}
P_0(3) &= 1 \\
P_1(3) &= d_1 - \lambda = 4 - 3 = 1 \\
P_2(3) &= (d_2 - \lambda)P_1(3) - c_1^2 P_0(3) = (4-3)1 - (-1)^2 1 = 0 \\
P_3(3) &= (d_3 - \lambda)P_2(3) - c_2^2 P_1(3) = (4-3)0 - (-1)^2 1 = -1
\end{aligned}$$

There is one sign change in this sequence; hence one of the eigenvalues is smaller than 3.
In conclusion:

$$2 \le \lambda_1 \le 4 \qquad \lambda_2 = 4 \qquad 4 \le \lambda_3 \le 6 \ \blacktriangleleft$$

Problem 4

$$\mathbf{A} = \begin{bmatrix} 6 & 1 & 0 \\ 1 & 8 & 2 \\ 0 & 2 & 9 \end{bmatrix}$$

Find global bounds with Gerschgorin's theorem:

$$\mathbf{a} = \begin{bmatrix} 6 \\ 8 \\ 9 \end{bmatrix} \qquad \mathbf{r} = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$$

$$\begin{aligned}
\lambda_{\min} &\ge \min_i (a_i - r_i) = 6 - 1 = 5 \\
\lambda_{\max} &\le \max_i (a_i + r_i) = 8 + 3 = 11
\end{aligned}$$

Use Sturm sequence to determine intermediate bounds:
With $\lambda = 8$:

$$\begin{aligned}
P_0(8) &= 1 \\
P_1(8) &= d_1 - \lambda = 6 - 8 = -2 \\
P_2(8) &= (d_2 - \lambda)P_1(8) - c_1^2 P_0(8) = (8-8)(-2) - 1^2\,(1) = -1 \\
P_3(8) &= (d_3 - \lambda)P_2(8) - c_2^2 P_1(8) = (9-8)(-1) - 2^2(-2) = 7
\end{aligned}$$

The sign changes indicate that there are 2 eigenvalues smaller than 8.
With $\lambda = 6$:

$$\begin{aligned}
P_0(6) &= 1 \\
P_1(6) &= d_1 - \lambda = 6 - 6 = 0 \\
P_2(6) &= (d_2 - \lambda)P_1(6) - c_1^2 P_0(6) = (8-6)0 - 1^2\,(1) = -1 \\
P_3(6) &= (d_3 - \lambda)P_2(6) - c_2^2 P_1(6) = (9-6)(-1) - 2^2(0) = -3
\end{aligned}$$

There is one sign change in this sequence; hence one of the eigenvalues is smaller than 6.
In conclusion:

$$5 \leq \lambda_1 \leq 6 \qquad 6 \leq \lambda_2 \leq 8 \qquad 8 \leq \lambda_3 \leq 11 \quad \blacktriangleleft$$

Problem 5

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Find global bounds with Gerschgorin's theorem:

$$\mathbf{a} = \begin{bmatrix} 2 \\ 2 \\ 2 \\ 1 \end{bmatrix} \qquad \mathbf{r} = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 1 \end{bmatrix}$$

$$\begin{aligned} \lambda_{\min} &\geq \min_i (a_i - r_i) = 2 - 2 = 0 \\ \lambda_{\max} &\leq \max_i (a_i + r_i) = 2 + 2 = 4 \end{aligned}$$

Use Sturm sequence to determine intermediate bounds:
<u>With $\lambda = 2$</u>:

$$\begin{aligned} P_0(2) &= 1 \\ P_1(2) &= d_1 - \lambda = 2 - 2 = 0 \\ P_2(2) &= (d_2 - \lambda)P_1(2) - c_1^2 P_0(2) = (2-2)(0) - (-1)^2\, 1 = -1 \\ P_3(2) &= (d_3 - \lambda)P_2(2) - c_2^2 P_1(2) = (2-2)(1) - (-1)^2(0) = 0 \\ P_4(2) &= (d_4 - \lambda)P_3(2) - c_3^2 P_2(2) = (1-2)(0) - (-1)^2 1 = 1 \end{aligned}$$

The two sign changes indicate that there are 2 eigenvalues smaller than 2.
<u>With $\lambda = 3$</u>:

$$\begin{aligned} P_0(3) &= 1 \\ P_1(3) &= d_1 - \lambda = 2 - 3 = -1 \\ P_2(3) &= (d_2 - \lambda)P_1(3) - c_1^2 P_0(3) = (2-3)(-1) - (-1)^2\,(1) = 0 \\ P_3(3) &= (d_3 - \lambda)P_2(3) - c_2^2 P_1(3) = (2-3)(0) - (-1)^2(-1) = 1 \\ P_4(3) &= (d_4 - \lambda)P_3(3) - c_3^2 P_2(3) = (1-3)(1) - (-1)^2(0) = -2 \end{aligned}$$

There are 3 sign change in this sequence; hence 3 of the eigenvalues are smaller than 3.

With $\lambda = 1$:

$$\begin{aligned}
P_0(1) &= 1 \\
P_1(1) &= d_1 - \lambda = 2 - 1 = 1 \\
P_2(1) &= (d_2 - \lambda)P_1(1) - c_1^2 P_0(1) = (2-1)(1) - (-1)^2\,(1) = 0 \\
P_3(1) &= (d_3 - \lambda)P_2(1) - c_2^2 P_1(1) = (2-1)(0) - (-1)^2(1) = -1 \\
P_4(1) &= (d_4 - \lambda)P_3(1) - c_3^2 P_2(1) = (1-1)(-1) - (-1)^2(0) = 0
\end{aligned}$$

$\lambda = 1$ is an eigenvalue
In conclusion:

$$0 \le \lambda_1 \le 1 \qquad \lambda_2 = 1 \qquad 2 \le \lambda_3 \le 3 \qquad 3 \le \lambda_4 \le 4 \quad \blacktriangleleft$$

Problem 6

$$\mathbf{A} = \begin{bmatrix} 12 & 4 & 3 \\ 4 & 9 & 3 \\ 3 & 3 & 15 \end{bmatrix}$$

$$\mathbf{A}' = \begin{bmatrix} 9 & 3 \\ 3 & 15 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} 4 \\ 3 \end{bmatrix} \qquad k = |\mathbf{x}| = 5 \text{ (note that } x_1 > 0)$$

$$\mathbf{u} = \begin{bmatrix} k + x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 9 \\ 3 \end{bmatrix} \qquad \mathbf{u}\mathbf{u}^T = \begin{bmatrix} 81 & 27 \\ 27 & 9 \end{bmatrix} \qquad H = \frac{1}{2}\,|\mathbf{u}|^2 = 45$$

$$\mathbf{Q} = \mathbf{I} - \frac{\mathbf{u}\mathbf{u}^T}{H} = \begin{bmatrix} 1 - 81/45 & -27/45 \\ -27/45 & 1 - 9/45 \end{bmatrix} = \begin{bmatrix} -0.8 & -0.6 \\ -0.6 & 0.8 \end{bmatrix}$$

$$\begin{aligned}
\mathbf{Q}^T\mathbf{A}' &= \begin{bmatrix} -0.8 & -0.6 \\ -0.6 & 0.8 \end{bmatrix} \begin{bmatrix} 9 & 3 \\ 3 & 15 \end{bmatrix} = \begin{bmatrix} -9.0 & -11.4 \\ -3.0 & 10.2 \end{bmatrix} \\
\mathbf{Q}^T\mathbf{A}'\mathbf{Q} &= \begin{bmatrix} -9.0 & -11.4 \\ -3.0 & 10.2 \end{bmatrix} \begin{bmatrix} -0.8 & -0.6 \\ -0.6 & 0.8 \end{bmatrix} = \begin{bmatrix} 14.04 & -3.72 \\ -3.72 & 9.96 \end{bmatrix}
\end{aligned}$$

$$\mathbf{A} \leftarrow \begin{bmatrix} A_{11} & (\mathbf{Q}\mathbf{x})^T \\ \mathbf{Q}\mathbf{x} & \mathbf{Q}^T\mathbf{A}'\mathbf{Q} \end{bmatrix} = \begin{bmatrix} 12 & -5 & 0 \\ -5 & 14.04 & -3.72 \\ 0 & -3.72 & 9.96 \end{bmatrix} \quad \blacktriangleleft$$

Problem 7

$$\mathbf{A} = \begin{bmatrix} 4 & -2 & 1 & -1 \\ -2 & 4 & -2 & 1 \\ 1 & -2 & 4 & -2 \\ -1 & 1 & -2 & 4 \end{bmatrix}$$

First round:

$$\mathbf{A}' = \begin{bmatrix} 4 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 4 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} -2 \\ 1 \\ -1 \end{bmatrix} \qquad k = -|\mathbf{x}| = -\sqrt{6} = -2.4495$$

$$\mathbf{u} = \begin{bmatrix} k + x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -4.4495 \\ 1 \\ -1 \end{bmatrix} \qquad \mathbf{u}\mathbf{u}^T = \begin{bmatrix} 19.7981 & -4.4495 & 4.4495 \\ -4.4495 & 1 & -1 \\ 4.4495 & -1 & 1 \end{bmatrix}$$

$$H = \frac{1}{2}|\mathbf{u}|^2 = \frac{1}{2}(4.4495^2 + 2) = 10.8990$$

$$\mathbf{Q} = \mathbf{I} - \frac{\mathbf{u}\mathbf{u}^T}{H} = \begin{bmatrix} -0.8165 & 0.4082 & -0.4082 \\ 0.4082 & 0.9082 & 0.0918 \\ -0.4082 & 0.0918 & 0.9082 \end{bmatrix}$$

$$\mathbf{Q}^T\mathbf{A}' = \begin{bmatrix} -4.4906 & 4.0822 & -3.2657 \\ -0.0918 & 2.6328 & -1.0410 \\ -0.9082 & -0.6328 & 3.0410 \end{bmatrix}$$

$$\mathbf{Q}^T\mathbf{A}'\mathbf{Q} = \begin{bmatrix} 6.6660 & 1.5746 & -0.7581 \\ 1.5746 & 2.2581 & -0.6663 \\ -0.7581 & -0.6663 & 3.0745 \end{bmatrix}$$

$$\mathbf{A} \leftarrow \begin{bmatrix} A_{11} & (\mathbf{Qx})^T \\ \mathbf{Qx} & \mathbf{Q}^T\mathbf{A}'\mathbf{Q} \end{bmatrix} = \begin{bmatrix} 4 & 2.4495 & 0 & 0 \\ 2.4495 & 6.6660 & 1.5746 & -0.7581 \\ 0 & 1.5746 & 2.2581 & -0.6663 \\ 0 & -0.7581 & -0.6663 & 3.0745 \end{bmatrix}$$

Second round:

$$\mathbf{A}' = \begin{bmatrix} 2.2581 & -0.6663 \\ -0.6663 & 3.0745 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} 1.5746 \\ -0.7581 \end{bmatrix}$$

$$k = |\mathbf{x}| = \sqrt{1.5746^2 + 0.7581^2} = 1.7476$$

$$\mathbf{u} = \begin{bmatrix} k + x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3.3222 \\ -0.7581 \end{bmatrix} \qquad \mathbf{u}\mathbf{u}^T = \begin{bmatrix} 11.0370 & -2.5186 \\ -2.5186 & 0.5747 \end{bmatrix}$$

$$H = \frac{1}{2}|\mathbf{u}|^2 = \frac{1}{2}(3.3222^2 + 0.7581^2) = 5.8059$$

$$\mathbf{Q} = \mathbf{I} - \frac{\mathbf{u}\mathbf{u}^T}{H} = \begin{bmatrix} -0.9010 & 0.4338 \\ 0.4338 & 0.9010 \end{bmatrix}$$

$$\mathbf{Q}^T\mathbf{A}' = \begin{bmatrix} -2.3236 & 1.9341 \\ 0.3792 & 2.4811 \end{bmatrix} \qquad \mathbf{Q}^T\mathbf{A}'\mathbf{Q} = \begin{bmatrix} 2.9326 & 0.7346 \\ 0.7346 & 2.4000 \end{bmatrix}$$

$$\mathbf{A} \leftarrow \begin{bmatrix} A_{11} & A_{12} & \mathbf{0}^T \\ A_{21} & A_{22} & (\mathbf{Qx})^T \\ \mathbf{0} & \mathbf{Qx} & \mathbf{Q}^T\mathbf{A}'\mathbf{Q} \end{bmatrix} = \begin{bmatrix} 4 & 2.4495 & 0 & 0 \\ 2.4495 & 6.6660 & -1.7476 & 0 \\ 0 & -1.7476 & 2.9326 & 0.7346 \\ 0 & 0 & 0.7346 & 2.4000 \end{bmatrix} \quad \blacktriangleleft$$

Problem 8

$$\mathbf{A} = \begin{bmatrix} 6 & 2 & 0 & 0 & 0 \\ 2 & 5 & 2 & 0 & 0 \\ 0 & 2 & 7 & 4 & 0 \\ 0 & 0 & 4 & 6 & 1 \\ 0 & 0 & 0 & 1 & 3 \end{bmatrix}$$

```
## problem9_2_8
from numpy import array
from eigenvals3 import *

d = array([6.0, 5.0, 7.0, 6.0, 3.0])
c = array([2.0, 2.0, 4.0, 1.0])
print(eigenvals3(d,c,5))
input("Press return to exit")

[ 1.43562719  2.93780551  4.15153739  7.44731309 11.02771683]
```

Problem 9

```
## problem9_2_9
from numpy import array
from householder import *
from eigenvals3 import *

a = array([[ 4, -1,  0, 1], \
           [-1,  6, -2, 0], \
           [ 0, -2,  3, 2], \
           [ 1,  0,  2, 4]])*1.0
d,c = householder(a)
print("Eigenvalues:\n",eigenvals3(d,c,2))
input("Press return to exit")

Eigenvalues:
[ 0.69385207  3.80559373]
```

Problem 10

```
## problem9_2_10
from numpy import array, zeros,dot
from householder import *
from eigenvals3 import *
from inversePower3 import *

N = 3
a = array([[ 7.0, -4.0,  3.0, -2.0,  1.0,  0.0], \
           [-4.0,  8.0, -4.0,  3.0, -2.0,  1.0], \
           [ 3.0, -4.0,  9.0, -4.0,  3.0, -2.0], \
           [-2.0,  3.0, -4.0, 10.0, -4.0,  3.0], \
           [ 1.0, -2.0,  3.0, -4.0, 11.0, -4.0], \
           [ 0.0,  1.0, -2.0,  3.0, -4.0, 12.0]])
xx = zeros((len(a),N))
d,c = householder(a)
p = computeP(a)
lambdas = eigenvals3(d,c,N)
for i in range(N):
    lam,xx[:,i] = inversePower3(d,c,lambdas[i]*1.0001)
xx = dot(p,xx)
print("Eigenvalues:\n",lambdas)
print("\nEigenvectors:\n",xx)
input("Press return to exit")
```

```
Eigenvalues:
[ 3.37684507  4.75931985  6.0368161 ]

Eigenvectors:
[[ 0.69038451 -0.37799658  0.26973816]
 [ 0.71514996  0.26749257 -0.24859356]
 [ 0.0950872   0.8386303  -0.03741361]
 [-0.02413926  0.27805223  0.80295336]
 [ 0.02568968 -0.04284676  0.46396587]
 [-0.04056509  0.05582685 -0.06359527]]
```

Problem 11

The elements of a $n \times n$ Hilbert matrix are

$$A_{ij} = \frac{1}{i+j+1}, \quad i,j = 0,1,\ldots,n-1$$

```
## problem9_2_11
from householder import *
from eigenvals3 import *
from numpy import zeros

n = 6  # Size of matrix
N = 2  # Number of eigenvalues
a = zeros((n,n))
for i in range(n):
    for j in range(n):
        a[i,j] = 1.0/(i+j+1)
d,c = householder(a)
print("Eigenvalues:\n",eigenvals3(d,c,N))
input("Press return to exit")
```

```
Eigenvalues:
[  1.81260083e-06   1.25707571e-05]
```

The solution is not reliable due to ill-conditioning.

Problem 12

```
## problem9_2_12
from numpy import ones,zeros
from sturmSeq import *
from gerschgorin import *
from householder import *

def lamRange(d,c,N):
  # This function brackets N largest eigenvalues
    n = len(d)
    lamMin,lamMax = gerschgorin(d,c)
    r = ones(N+1)
    r[N] = lamMax
  # Search for eigenvalues in ascending order
```

```
    for k in range(N):
      # First bisection of interval(lamMin,lamMax)
        lam = (lamMax + lamMin)/2.0
        h = (lamMax - lamMin)/2.0
        for i in range(1000):
          # Find number of eigenvalues less than lam
            p = sturmSeq(d,c,lam)
            numLam = numLambdas(p)
          # Bisect again & find the half containing lam
            h = h/2.0
            if numLam < n - N + k: lam = lam + h
            elif numLam > n - N + k: lam = lam - h
            else: break
      # If eigenvalue located, change the lower limit
      # of search and record it in [r]
        lamMin = lam
        r[k] = lam
    return r

n = 6
a = ones((n,n))
for i in range(n):
    for j in range(6):
        a[i,j] = 1.0/(i+j+1)
d,c = householder(a)
print("Brackets on eigenvalues:\n",lamRange(d,c,2))
input("\nPress return to exit")

Brackets on eigenvalues:
[ 0.15921438  0.93010287  1.70099136]
```

Problem 13

Substituting $u_i(t) = y_i \sin \omega t$ we get the non-standard eigenvalue problem $\mathbf{Ay} = \lambda \mathbf{By}$, where $\lambda = \omega^2/k$. The matrix $\mathbf{A}$ is tridiagonal: $\mathbf{A} = [\mathbf{c}\backslash\mathbf{d}\backslash\mathbf{c}]$, where

$$\mathbf{d} = \begin{bmatrix} k_1 + k_2 \\ k_2 + k_3 \\ \vdots \\ k_{n-1} + k_n \\ k_n \end{bmatrix} \qquad \mathbf{c} = \begin{bmatrix} -k_2 \\ -k_3 \\ \vdots \\ -k_n \end{bmatrix}$$

and B is diagonal:

$$B = \begin{bmatrix} m_1 & 0 & 0 & \cdots & 0 \\ 0 & m_2 & 0 & \cdots & 0 \\ 0 & 0 & m_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & & m_n \end{bmatrix}$$

```
## problem 9_2_13
from eigenvals3 import *
from inversePower3 import *
import numpy as np
k = 500.0e3
m = np.array([1.0, 1.0, 1.0, 8.0, 1.0, 1.0, 8.0])
n = len(m)                    # Number of masses (DOF)
m = 1.0/np.sqrt(m)            # Used in transformation
N = 2                         # Number of eigenvalues reqd.
d = np.ones(n)*2.0
d[n-1] = 1.0
c = -np.ones(n-1)
xx = np.zeros((n,N))          # Storage for eigenvectors
# Transform the problem into the standard form
for i in range(n): d[i] = d[i]*m[i]*m[i]
for i in range(n-1): c[i] = c[i]*m[i]*m[i+1]
lam = eigenvals3(d,c,N)       # Compute eigenvalues
# Compute eigenvectors
for i in range(N):
    s = lam[i]*1.0000001      # Shift close to eigenvalue
    eVal,x = inversePower3(d,c,s)
    x = x*m                   # Eigenvctors of orig. problem
    x = x/np.sqrt(np.dot(x,x)) # Normalize eigenvectors
    xx[:,i] = x               # Store in array [xx]
# Format and print results
print("Circular frequencies (in rad/s)):")
for i in range(N): print("{:10.6f}".format(np.sqrt(lam[i]*k)),end=" ")
print("\n\nRelative displacements:")
for i in range(n):
    for j in range(N):
        print("{:10.6f}".format(xx[i,j]),end=" ")
    print()
input("\n Press return to exit"))
```

```
Circular frequencies (in rad/s)):
 74.268887 215.306539
```

```
Relative displacements:
  0.097824  -0.182732
  0.194570  -0.348522
  0.289168  -0.481999
  0.380577  -0.570788
  0.438398  -0.236218
  0.491384   0.120253
  0.538948   0.465575
```

Problem 14

Substituting $u_i(t) = y_i \sin \omega t$ we get the standard eigenvalue problem $\mathbf{Ay} = \lambda\mathbf{y}$, where $\lambda = m\omega^2$ and $\mathbf{A} = [\mathbf{c}\backslash\mathbf{d}\backslash\mathbf{c}]$ is tridiagonal with

$$\mathbf{d} = \begin{bmatrix} k_1 + k_2 \\ k_2 + k_3 \\ \vdots \\ k_{n-1} + k_n \\ k_n \end{bmatrix} \qquad \mathbf{c} = \begin{bmatrix} -k_2 \\ -k_3 \\ \vdots \\ \not{k}_n \end{bmatrix}$$

```
## problem9_2_14
from eigenvals3 import *
from inversePower3 import *
import numpy as np
k = np.array([400.0, 400.0, 400.0, 0.2, 400.0, 400.0, 200.0])*1000.0
N = 2                    # Number of eigenvalues reqd.
m = 2.0                  # Mass in kg
n = len(k)               # Number of equations
d = np.zeros(n)
c = np.zeros(n-1)
xx = np.zeros((n,N))     # Storage for eigenvectors
for i in range(n-1):     # Set up vectors d and e
    d[i] = k[i]+ k[i+1]
    c[i] = -k[i+1]
d[n-1] = k[n-1]
lam = eigenvals3(d,c,N)  # Compute eigenvelues
for i in range(N):       # Compute eigenvectors
    eVal,x = inversePower3(d,c,lam[i]*1.0001)
    xx[:,i] = x
print("Circular freqiencies in rad/s:\n",np.sqrt(lam/m))
print("\nRelative displacements:\n",xx)
input("Press return to exit")
```

```
Circular freqiencies in rad/s:
[   4.99508294  199.16513625]

Relative displacements:
[[  2.49695298e-04   3.28079871e-01]
 [  4.99359446e-04   5.91090428e-01]
 [  7.48961296e-04   7.36867800e-01]
 [  4.99765790e-01   9.21258977e-05]
 [  4.99952950e-01  -2.94533615e-04]
 [  5.00077739e-01  -6.22777168e-04]
 [  5.00202544e-01  -1.03222880e-03]]
```

The first mode reflects the weak coupling: the four masses on the right move in unison as the remaining three masses are almost stationary.

Problem 15

(a)

This is a non-standard eigenvalue problem $\mathbf{Ax} = \lambda\mathbf{Bx}$. The matrix $\mathbf{B}$ is empty except for its diagonal $\boldsymbol{\beta} = \begin{bmatrix} 1 & 1 & \cdots & 1 & 1/2 \end{bmatrix}^T$. According to Eq. (9.26b) the matrix $\mathbf{H}$ of the standard problem is

$$H_{ij} = \frac{A_{ij}}{\sqrt{\beta_i \beta_j}}$$

In this case, the differences between $\mathbf{H}$ and $\mathbf{A}$ are confined to the last row and column:

$$\begin{aligned} H_{in} &= H_{ni} = \sqrt{2}A_{in}, \quad i = 1, 2, \ldots, n-1 \\ H_{nn} &= 2A_{nn} \end{aligned}$$

Thus

$$\mathbf{H} = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -\sqrt{2} \\ & & & -\sqrt{2} & 2 \end{bmatrix} \blacktriangleleft$$

The transformation is

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & \sqrt{2} \end{bmatrix} \mathbf{z} \blacktriangleleft$$

(b)

```
## problem9_2_15
from numpy import array,ones
from LUdecomp3 import *
from inversePower3 import *
from math import sqrt

n = array([10, 100, 1000])
for i in range(len(n)):
    d = ones(n[i])*2.0
    c = ones(n[i]-1)*(-1.0)
    c[n[i]-2] = -sqrt(2.0)
    eVal,x = inversePower3(d,c,0.0,1.0e-9)
print("\nn =",n[i])
    print("Cicular frequency in units of sqrt(E/rho)/L:")
    print(sqrt(eVal)*n[i])
input("Press return to exit")
```

```
n = 10
Cicular frequency in units of sqrt(E/rho)/L:
1.56918191456

n = 100
Cicular frequency in units of sqrt(E/rho)/L:
1.57078017774

n = 1000
Cicular frequency in units of sqrt(E/rho)/L:
1.5707961653
```

The analytical solution is

$$\omega_1 = \frac{\pi}{2}\sqrt{\frac{E}{\rho}}\frac{1}{L} = 1.570\,79\,6\sqrt{\frac{E}{\rho}}\frac{1}{L}$$

Problem 16

```
## problem9_2_16
from numpy import zeros,dot
from stdForm import *
from householder import *
from eigenvals3 import *
```

```
from inversePower3 import *

N = 3                          # Number of eigenvalues
n = 25                         # Number of equations
param = 1000.0                 # param = kL**4/EI
alpha = param/(n+1)**4
a = zeros((n,n))
b = zeros((n,n))
for i in range(n):
    a[i,i] = 6.0 + alpha
    b[i,i] = 2.0
a[0,0] = 5.0 + alpha
a[n-1,n-1] = 5.0 + alpha
for i in range(n-1):
    a[i,i+1] = -4.0; a[i+1,i] = -4.0
    b[i,i+1] = -1.0; b[i+1,i] = -1.0
for i in range(n-2):
    a[i,i+2] = 1.0; a[i+2,i] = 1.0

xx = zeros((n,N))
h,t1 = stdForm(a,b)             # Transform to [H]{x} = lam{x}
d,c = householder(h)            # Tridiagonalize [H]
t2 = computeP(h)                # Compute transformation matrix
lam = eigenvals3(d,c,N)         # Compute eigenvalues
for i in range(N):              # Compute eigenvectors
    eVal,x = inversePower3(d,c,lam[i]*1.0001)
    xx[:,i] = x
xx = dot(t2,xx)  # Recover eigenvectors of the
xx = dot(t1,xx)  # original problem
print("PL**2/EI at buckling:\n",lam*(n+1)**2)
print("\nEigenvectors:\n",xx)
input("Press return to exit")
```

```
PL**2/EI at buckling:
[  64.74059371   99.2400375   111.30215007]

Eigenvectors:
[[ -2.75327904e-01  -2.72807078e-01   2.76844088e-01]
 [ -5.34654750e-01  -5.10158098e-01   5.49651165e-01]
 [ -7.62909406e-01  -6.81205138e-01   8.14443091e-01]
 [ -9.46826541e-01  -7.63717639e-01   1.06735860e+00]
 [ -1.07571756e+00  -7.46971657e-01   1.30470962e+00]
 [ -1.14209178e+00  -6.33143625e-01   1.52303504e+00]
 [ -1.14209178e+00  -4.37027490e-01   1.71915117e+00]
 [ -1.07571756e+00  -1.84111979e-01   1.89019821e+00]
```

```
[ -9.46826541e-01   9.27321093e-02   2.03368191e+00]
[ -7.62909406e-01   3.57524035e-01   2.14750994e+00]
[ -5.34654750e-01   5.75849451e-01   2.23002244e+00]
[ -2.75327904e-01   7.19333144e-01   2.28001619e+00]
[ -3.71533360e-12   7.69326897e-01   2.29676218e+00]
[  2.75327904e-01   7.19333144e-01   2.28001619e+00]
[  5.34654750e-01   5.75849451e-01   2.23002244e+00]
[  7.62909406e-01   3.57524035e-01   2.14750994e+00]
[  9.46826541e-01   9.27321095e-02   2.03368191e+00]
[  1.07571756e+00  -1.84111978e-01   1.89019821e+00]
[  1.14209178e+00  -4.37027490e-01   1.71915117e+00]
[  1.14209178e+00  -6.33143625e-01   1.52303504e+00]
[  1.07571756e+00  -7.46971657e-01   1.30470962e+00]
[  9.46826541e-01  -7.63717639e-01   1.06735860e+00]
[  7.62909406e-01  -6.81205138e-01   8.14443091e-01]
[  5.34654750e-01  -5.10158098e-01   5.49651165e-01]
[  2.75327904e-01  -2.72807078e-01   2.76844088e-01]]
```

Problem 17

```
## problem9_2_17
from householder3 import *
from eigenvals3 import *
from numpy import zeros

N = 5        # Number of eigenvalues requested
n = 20       # Number of equations
a = zeros((n,n))
for i in range(n):
    a[i,i] = 2.0
    if i <= n-2: a[i+1,i] = a[i,i+1] = 1.0
a[n-1,0] = a[0,n-1] = 1.0

d,c = householder(a)        # Tridiagonalize [A]
lam = eigenvals3(d,c,N)     # Compute eigenvalues
print("Eigenvalues:\n",lam)
input("Press return to exit")

Eigenvalues:
 [  1.91903785e-16   9.78869674e-02   9.78869674e-02   3.81966011e-01
   3.81966011e-01]
```

Problem 18

Substituting $\xi = x/L$, the differential equation

$$\frac{d^2\theta}{dx^2} + \gamma^2 \left(1 - \frac{x}{L}\right)^2 \theta = 0$$

becomes

$$\frac{d^2\theta}{d\xi^2} + \gamma^2 L^2 (1 - \xi)^2 \theta = 0$$

According to Eqs. (8.11) the finite difference approximation is (m is the number of intervals)

$$\begin{aligned} \theta_0 &= 0 \\ \theta_{i-1} - 2\theta_i + \theta_{i+1} + h^2\gamma^2 L^2 (1 - \xi_i)^2 \theta_i &= 0 \quad i = 1, 2, \ldots, m-1 \\ 2\theta_{m-1} - 2\theta_m - h^2\gamma^2 L^2 (1 - \xi_m)^2 \theta_m + 2h(0) &= 0 \end{aligned}$$

But $1 - \xi_m = 0$, so that the last equation is simply $\theta_m = \theta_{m-1}$. Substituting the first and last equations into the remaining equations, we obtain the following $(m-1) \times (m-1)$ matrix eigenvalue problem

$$\begin{aligned} 2\theta_1 - \theta_2 &= \lambda (1 - \xi_1)^2 \theta_1 \\ -\theta_{i-1} + 2\theta_i - \theta_{i+1} &= \lambda (1 - \xi_i)^2 \theta_i \quad i = 2, 3, \ldots, m-2 \\ -\theta_{m-2} + \theta_{m-1} &= \lambda (1 - \xi_{m-1})^2 \theta_{m-1} \end{aligned}$$

where

$$\lambda = h^2 \gamma^2 L^2 = \frac{P^2 L^4 h^2}{(GJ)(EI_z)}$$

Note that the problem is tridiagonal but not of standard form.

```
## problem9_2_18
from numpy import ones,zeros
from math import sqrt
from inversePower3 import *

m = 50                  # Use 50 intervals
h = 1.0/m
n = m - 1               # Size of matrix after boundary
                        # conditions are applied
# Set up matrices
d = ones(n)*2.0
d[n-1] = 1.0
c = ones(n-1)*(-1.0)
b = zeros(n)
for i in range(n):
```

```
    b[i] = (1.0-h*(i+1))**2

# Transform to std. form using Eq. (9.26b)
d[n-1] = d[n-1]/b[n-1]
for i in range(n-1):
    d[i] = d[i]/b[i]
    c[i] = c[i]/sqrt(b[i]*b[i+1])

eVal,eVec = inversePower3(d,c,0.0)
print('Lowest eigenvalue =',eVal)
input("Press return to exit")
```

```
Lowest eigenvalue = 0.0064391022913483096
```

$$
\begin{aligned}
P_{cr} &= \frac{\sqrt{\lambda (GJ)(EI_z)}}{hL^2} = \frac{\sqrt{0.006\ 439\ 1 (GJ)(EI_z)}}{0.02L^2} \\
&= 4.012 \frac{\sqrt{(GJ)(EI_z)}}{L^2}
\end{aligned}
$$

which agrees well with the analytical solution.

Problem 19

```
## problem9_2_19
from householder import *
from eigenvals3 import *
from rootsearch import *
from ridder import *
import numpy

def f(z):
    a = np.array([[z,4,3,5,2,1],
                  [4,z,2,4,3,4],
                  [3,2,z,4,1,8],
                  [5,4,4,z,2,5],
                  [2,3,1,2,z,3],
                  [1,4,8,5,3,z]],float)

    d,c = householder(a)       # Tridiagonalize matrix
    lam = eigenvals3(d,c,1)    # Compute smallest eigenvalue
    return lam[0] - 1.0
```

```
z1,z2 = rootsearch(f,0.0,20.0,1.0)
print('z =',ridder(f,z1,z2))
input("Press return to exit")
```

```
z = 10.1902509257
```

PROBLEM SET 10.1

Problem 1

$$V = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

Letting $x = \sigma/r$, the function to be minimized is $f(x) = x^{12} - x^6$.

```
## problem10_1_1
from goldSearch import *

def f(x):
    return x**12 -x**6

x1,x2 = bracket(f,0.0,0.02)
x,fMin = search(f,x1,x2)
print("x =",x)
print("f(x) =",fMin)
input ("Press return to exit")

x = 0.8908987174369027
f(x) = -0.25
```

Analytical solution:

$$\begin{aligned}\frac{df}{dx} &= 12x^{11} - 6x^5 = 0 \qquad \left(2x^6 - 1\right)x = 0 \\ x &= 2^{-1/6} = 0.890\,899 \;\blacktriangleleft\; \text{Checks}\end{aligned}$$

Problem 2

We are to minimize the function $f(\sigma) = (27 - 18\sigma + 2\sigma^2)e^{-\sigma/3}$.

```
## problem10_1_2
from goldSearch import *
from math import exp

def f(x):
    return (27.0 - 18.0*x + 2.0*x**2)*exp(-x/3.0)
```

```
x1,x2 = bracket(f,0.0,0.01)
x,fMin = search(f,x1,x2)
print("x =",x)
print("f(x) =",fMin)
input ("Press return to exit")
```

```
x = 3.531372993699028
f(x) = -3.5819473201969343
```

Analytical solution:

$$
\begin{aligned}
\frac{df}{d\sigma} &= 0 \\
-\frac{1}{3}(27 - 18\sigma + 2\sigma^2)e^{-\sigma/3} + (-18 + 4\sigma)e^{-\sigma/3} &= 0 \\
-\frac{1}{3}\left(81 - 30\sigma + 2\sigma^2\right)e^{-\sigma/3} &= 0
\end{aligned}
$$

$$
\sigma = \frac{30 \pm \sqrt{30^2 - 4(81)(2)}}{4} = \frac{30 \pm 15.87451}{4} = \begin{cases} 11.468\,63 \\ 3.531\,373 \quad \blacktriangleleft \text{ Checks} \end{cases}
$$

Problem 3

$$
f(p) = \int_0^{\pi} \sin x \cos px \, dx
$$

```
## problem10_1_3
from goldSearch import *
from romberg import *
from math import pi,sin,cos

def f(p):
    def func(t): return sin(t)*cos(p*t)
    I,nPanels = romberg(func,0.0,pi)
    return I

pStart = 0.0
p1,p2 = bracket(f,pStart,0.01)
p,pMin = search(f,p1,p2)
print("p =",p)
print("f(p) =",pMin)
input ("Press return to exit")
```

```
p = 1.6521885158867344
f(p) = -0.844125033109
```

Problem 4

$$\begin{aligned} R_1 i_1 + R_3 i_1 + R(i_1 - i_2) &= E \\ R_2 i_2 + R_4 i_2 + R_5 i_2 + R(i_2 - i_1) &= 0 \end{aligned}$$

The power dissipated by the resistor is $P(R) = R(i_1 - i_2)^2$. Since we want to maximize the power, we must minimize $-P(R)$.

```
## problem10_1_4
from goldSearch import *
from gaussElimin import *
from numpy import array,zeros

def f(R):
    R1 = 2.0; R2 = 3.6; R3 = 1.5; R4 = 1.8
    R5 = 1.2; E = 120.0
    a = zeros((2,2),float)
    b = zeros((2),float)
    a[0,0] = R1 + R3 + R
    a[0,1] = -R; a[1,0] = -R
    a[1,1] = R2 + R4 + R5 + R
    b[0] = E; b[1] = 0.0
    i = gaussElimin(a,b)
    P = R*(i[0] - i[1])**2
    return -P

Rmin,Rmax = bracket(f,1.0,0.01)
R,Pmin = search(f,Rmin,Rmax)
print("Resistance =",R,"ohms")
print("Power =",-Pmin,"watts")
input ("Press return to exit")

Resistance = 2.2871286570975604 ohms
Power = 672.135785007 watts
```

Problem 5

$$T = \frac{q}{2\pi}\left(\frac{\ln(r/a)}{k} + \frac{1}{hr}\right) + T_\infty$$

```
## problem10_1_5
from goldSearch import *
from math import log,pi

def T(r):
    return q*(log(r/a)/k + 1.0/(h*r))/(2.0*pi) + Tinf

a = 0.005; h = 20.0; k = 0.16; q = 50.0; Tinf = 280.0
r1,r2 = bracket(T,0.006,0.0001)
r,TMin = search(T,r1,r2)
print("r =",r,"m")
print("T(r) =",TMin,"K")
input ("Press return to exit")

r = 0.008000000000874632 m
T(r) = 353.11198248671394 K
```

Problem 6

$$F(x,y) = (x-1)^2 + (y-1)^2 \qquad x+y \le 1 \qquad x \ge 0.6$$

```
## problem10_1_6
from powell import *
from numpy import array

def F(x): return (x[0] -1 )**2 + (x[1] - 1)**2

def Fstar(x):   # Penalized merit function
    c1 = max(0.0, x[0] + x[1] - 1.0)
    c2 = min(0.0, x[0] - 0.6)
    return F(x) + lam*(c1**2 + c2**2)

lam = 100.0
xStart = array([1.0, 1.0])
x,nIter = powell(Fstar,xStart,0.01)
print("x =",x)
print("F(x) =",F(x))
```

```
print("Number of cycles =",nIter)
input ("Press return to exit")
```

```
x = [ 0.59809727  0.40782448]
F(x) = 0.512197645795
Number of cycles = 3
```

Running the program again with

```
lam = 10000.0
xStart = array([0.59809727,0.40782448])
```

we obtain

```
x = [ 0.59998001  0.40007998]
F(x) = 0.519920020263
Number of cycles = 3
```

Problem 7

$$F(x,y) = 6x^2 + y^3 + xy \qquad y \geq 0$$

```
## problem10_1_7
from powell import *
from math import pi, sqrt
from numpy import array

def F(x):
    lam = 100.0
    F = 6.0*x[0]**2 + x[1]**3 + x[0]*x[1]
    c1 = min(0.0, x[1])
    return F + lam*(c1**2)

xStart = array([1.0, -1.0])
x,nIter = powell(F,xStart,0.01)
print("x =",x)
print("Number of cycles =",nIter)
print("F(x) =",F(x))
input ("Press return to exit")
```

```
x = [-0.00231481  0.02777778]
Number of cycles = 5
F(x) = -1.07167352538e-05
```

Note that the constraint $y \geq 0$ was not active at the optimal point.
Analytical solution:

$$\begin{aligned} \frac{\partial F}{\partial x} &= 0 \qquad 12x + y = 0 \qquad y = -12x \\ \frac{\partial F}{\partial y} &= 0 \qquad 3y^2 + x = 0 \qquad 3(-12x)^2 + x = 0 \end{aligned}$$

$$432x^2 + x = 0 \qquad x = \begin{cases} 0 \\ -0.002\,314\,82 \end{cases} \qquad y = \begin{cases} 0 \\ 0.027\,777\,8 \end{cases}$$

The solution $x = -0.002\,314\,82$, $y = 0.027\,777\,8$ is the optimal one since it results in smaller F than $x = y = 0$.

Problem 8

We use the program in Problem 7 with $c1$ changed to:

```
c1 = min(0.0, x[1] + 2.0)
```

The result is

```
x = [ 0.17206767 -2.06481207]
Number of cycles = 3
F(x) = -8.56080397812
```

Since the constraint $y \geq -2$ is violated significantly, we run the program again with the following changes:

```
lam = 10000.0
xStart = array([0.17206767, -2.06481207])
```

obtaining

```
x = [ 0.1667174 -2.0006087]
Number of cycles = 3
F(x) = -8.17036959852
```

Analytical solution:
As the constraint $y \geq -2$ is active at the optimal point, we set $y = -2$ in the merit function:

$$F = 6x^2 + (-2)^3 + x(-2) = 6x^2 - 8 - 2x$$

$$\frac{dF}{dx} = 0 \qquad 12x - 2 = 0 \qquad x = \frac{1}{6} = 0.166\,667$$

Problem 9

```
## problem10_1_9
from powell import *
from numpy import array
from math import sqrt

def F(x):                    # Merit function (distance**2)
    return (x[0] - 1)**2 + (x[1] - 2)**2

def Fstar(x):                # Penalized merit function
    c = x[1] - x[0]**2       # Point x must be on parabola
    return  F(x) + lam*c**2

lam = 100.0
xStart = array([2.0, 2.0])
x,numIter = powell(Fstar,xStart,0.01)
print("Intersection point =",x)
print("Minimum distance =", sqrt(F(x)))
print("Constraint y - x**2 =", x[1] - x[0]**2)
print("Number of cycles =",numIter)
input ("Press return to exit")
```

```
Intersection point = [ 1.36557917  1.86614502]
Minimum distance = 0.3893138647716713
Constraint y - x**2 = 0.00133855010812
Number of cycles = 5
```

To get closer to the constaint $y - x^2 = 0$, we run the program again with the data

```
lam = 10000.0
xStart = array([1.36557917, 1.86614502])
```

This yields

```
Intersection point = [ 1.36602094  1.8660266 ]
Minimum distance = 0.3897694175407153
Constraint y - x**2 = 1.33973633374e-05
Number of cycles = 3
```

Problem 10

The location of the centroid is

$$\begin{aligned} d &= \frac{\sum A_i d_i}{\sum A_i} = \frac{(0.4)^2(0.2) - (0.2x)(0.4 - x/2)}{(0.4)^2 - 0.2x} \\ &= \frac{0.032 - 0.08x + 0.1x^2}{0.16 - 0.2x} \end{aligned}$$

```
## problem10_1_10
from goldSearch import *

def f(x):
    return (0.032 - 0.08*x + 0.1*x**2) \
           /(0.16 - 0.2*x)

xStart = 0.2
x1,x2 = bracket(f,xStart,0.01)
x,d = search(f,x1,x2)
print("x =",x,"m")
print("d =",d,"m")
input ("Press return to exit")

x = 0.2343145715181848 m
d = 0.165685424949238 m
```

Problem 11

The mass of water in the vessel is

$$M_w = \pi r^2 x\gamma = \pi(0.25)^2 x(1000) = 62.5\pi x \text{ kg}$$

The height of the combined centroid above the floor of the vessel is

$$\begin{aligned} d &= \frac{\sum m_i d_i}{\sum m_i} = \frac{M(0.43H) + M_w(x/2)}{M + M_w} \\ &= \frac{115(0.43)(0.8) + 62.5\pi x^2/2}{115 + 62.5\pi x} = \frac{39.56 + 31.25\pi x^2}{115 + 62.5\pi x} \text{ m} \end{aligned}$$

```
## problem10_1_11
from goldSearch import *
from math import pi
```

```
def f(x):
    return (39.56 + 31.25*pi*x**2)/(115.0 + 62.5*pi*x)

xStart = 0.4
x1,x2 = bracket(f,xStart,0.01)
x,d = search(f,x1,x2)
print("x =",x,"m")
print("d =",d,"m")
input("Press return to exit")
```

```
x = 0.27801569882152866 m
d = 0.278015691338397 m
```

Note that $x = d$ when d is minimized.

Problem 12

In the program below we use the notation $a = x_0$ and $b = x_1$.

```
## problem10_1_12
from powell import *
from numpy import array

def F(x):                # Merit function (cardboard area)
    return 4.0*x[0]*x[1] + x[0]*x[0]

def Fstar(x):            # Penalized merit function
    V = x[0]*x[0]*x[1]   # Volume of box
    c1 =  V - 1.0
    return F(x) + lam*(c1**2)

lam = 100.0
xStart = array([1.0, 1.0])
x,nIter = powell(Fstar,xStart,0.01)
print("a =",x[0],"m")
print("b =",x[1],"m")
print("Cardboard area =",F(x),"m**2")
print("Volume of box =", x[0]*x[0]*x[1],"m**3")
print("Number of cycles =",nIter)
input ("Press return to exit")
```

```
a = 1.25318253962 m
```

```
b = 0.626591302134 m
Cardboard area = 4.71139959486 m**2
Volume of box = 0.984040635164 m**3
Number of cycles = 7
```

Running the program again with the changes

```
lam = 10000.0
xStart = array([1.25318253962, 0.626591302134])
```

results in

```
a = 1.25985437487 m
b = 0.629927189392 m
Cardboard area = 4.76169914752 m**2
Volume of box = 0.999841251507 m**3
Number of cycles = 5
```

The true solution is $a = 2b = \sqrt[3]{2} = 1.259\,921$.

Problem 13

```
## problem10_1_13
from powell import *
from math import sqrt
from numpy import array

def F(x):  # Merit function (potential energy)
    a = 150.0; b = 50.0; k = 0.6; P = 5.0
    delAB = sqrt((a + x[0])**2 + x[1]**2) - a
    delBC = sqrt((b - x[0])**2 + x[1]**2) - b
    return -P*x[1] + 0.5*k*(a + b)*(delAB**2/a + delBC**2/b)

xStart = array([0.0, 0.0])
x,nIter = powell(F,xStart)
print("u =",x[0],"mm")
print("v =",x[1],"mm")
print("Number of cycles =",nIter)
input("Press return to exit")
```

```
u = 5.21061762081 mm
v = 28.3750880943 mm
Number of cycles = 4
```

Problem 14

```
## problem10_1_14
from powell import *
from math import sin,cos,pi
from numpy import array
# We use the notation A = x[0], theta = x[1]

def F(x):  # Merit function (structural volume)
    return b*x[0]/cos(x[1])

def response(x):
    d = P*b/(2.0*E*x[0]*sin(2.0*x[1])*sin(x[1])) # Displacement
    s = P/(2.0*x[0]*sin(x[1]))                   # Stress
    return d,s

def Fstar(x):  # Penalized merit function
  # Scale displt. constraint by 1.0e6 to match the magnitude
  # of the stress constraint
    d,s = response(x)
    c1 = max(0.0, d - dAll)*1.0e6
    c2 = max(0.0, s - sAll)
    return F(x) + lam*(c1**2 + c2**2)

b = 4.0; P = 50.0e3; E = 200.0e9
sAll = 150.0e6      # Allowable stress
dAll = 0.005        # Allowable displt.

lam = 1.0
xStart = array([0.0005, 0.8])
x,nIter = powell(Fstar,xStart,0.00001)
d,s = response(x)
print("A      =",x[0]*1.0e6,"mm**2")
print("Theta  =",x[1]*180.0/pi,"deg")
print("Volume =",F(x),"m**3")
print("Stress =",s*1.0e-6,"MPa")
print("Displt =",d*1.0e3,"mm")
print("Number of cycles =",nIter)
input ("Press return to exit")
```

```
A      = 232.334815407 mm**2
Theta  = 45.8365781993 deg
Volume = 0.00133390204337 m**3
```

```
Stress = 150.0 MPa
Displt = 3.00127959758 mm
Number of cycles = 2
```

Note that only the stress constraint is active.

Problem 15

Using the program in Problem 14 with the displacement constraint changed to `dAll = 0.0025` we get

```
A      = 278.920616767 mm**2
Theta  = 45.8365961387 deg
Volume = 0.00160136525356 m**3
Stress = 124.946703895 MPa
Displt = 2.5000000003 mm
Number of cycles = 2
```

Here the optimal design is governed by the displacement constraint.

Problem 16

```
## problem10_1_16
from powell import *
from math import pi
from numpy import array
# We use the notation r1 = x[0], r2 = x[1]

def F(x):  # Merit function (structural volume)
    return  pi*L*(x[0]**2 + x[1]**2)

def response(x):
    d = 4.0*P*L**3/(3.0*pi*E)    \
        *(7.0/x[0]**4 + 1.0/x[1]**4) # Displacement
    s1 = 8.0*P*L/(pi*x[0]**3)        # Stress 1
    s2 = 4.0*P*L/(pi*x[1]**3)        # Stress 2
    return d,s1,s2

def Fstar(x):  # Penalized merit function
    d,s1,s2 = response(x)
```

```
    # Scale displt. constraint by 1.0e6 to match the magnitude
    # of the stress constraint
    c1 = max(0.0, d - dAll)*1.0e6
    c2 = max(0.0, s1 - sAll)
    c3 = max(0.0, s2 - sAll)
    return F(x) + lam*(c1**2 + c2**2 + c3**2)

L = 1.0; E = 200.0e9; P = 10.0e3
sAll = 180.0e6  # Allowable stress
dAll = 0.025    # Allowable displacement

lam = 1.0
xStart = array([0.1, 0.1])
x,nIter = powell(Fstar,xStart,0.001)
d,s1,s2 = response(x)
print("r1 =",x[0],"m")
print("r2 =",x[1],"m")
print("Volume  =",F(x),"m**3")
print("Stress1 =",s1*1.0e-6,"MPa")
print("Stress2 =",s2*1.0e-6,"MPa")
print("Displt  =",d,"m")
print("Number of cycles =",nIter)
input ("Press return to exit")
```

```
r1 = 0.052106176146 m
r2 = 0.0457383295326 m
Volume  = 0.0151017878792 m**3
Stress1 = 179.999999729 MPa
Stress2 = 133.066649298 MPa
Displt  = 0.0250000000006 m
Number of cycles = 2
```

The displacement constraint and one of the stress constraints are active at the optimal point.

Problem 17

```
## problem10_1_17
from downhill import *
from numpy import array

def F(x):
```

```
    return  2.0*x[0]**2 + 3.0*x[1]**2 + x[2]**2  \
           + x[0]*x[1] + x[0]*x[2] - 2.0*x[1]

x = array([0.0, 0.0, 0.0])
x = downhill(F,x)
print("x,y,z =",x)
print("F(x,y,z) =",F(x))
input("Press return to exit")
```

```
x,y,z = [-0.10000075  0.34999998  0.05000037]
F(x,y,z) = -0.349999999999
```

Analytical solution:

$$\begin{aligned} \frac{\partial F}{\partial x} &= 0 \qquad 4x + y - z = 0 \\ \frac{\partial F}{\partial y} &= 0 \qquad x + 6y = 2 \\ \frac{\partial F}{\partial z} &= 0 \qquad x + 2z = 0 \end{aligned}$$

The solution of these equations is $x = -0.1$, $y = 0.35$, $z = 0.05$.

Problem 18

$$V = \pi r^2 \left(\frac{b}{3} + h \right) \qquad S = \pi r \left(2h + \sqrt{b^2 + r^2} \right)$$

The most efficient way to obtain the optimal design is to solve the volume constraint $V = 1$ for h and substitute the result into the expression for S:

$$\pi r^2 \left(\frac{b}{3} + h \right) = 1 \qquad h = \frac{1}{\pi r^2} - \frac{b}{3}$$

$$S = \pi r \left(\frac{2}{\pi r^2} - \frac{2b}{3} + \sqrt{b^2 + r^2} \right)$$

We now have an unconstrained optimization problem) in the two variables.

```
## problem10_1_18
from powell import *
from math import pi, sqrt
from numpy import array
# Notation: b = x[0], r = x[1]
```

```
def F(x):
    return pi*x[1]*(2.0/pi/x[1]**2   \
                    - 2.0*x[0]/3.0   \
                    + sqrt(x[0]**2 + x[1]**2))

xStart = array([0.5,0.5])
x,nIter = powell(F,xStart,0.001)
h=1.0/pi/x[1]**2 - x[0]/3.0
print("b =",x[0],"m")
print("r =",x[1],"m")
print("h =",h,"m")
print("Surface area =",F(x),"m**2")
print("Number of cycles =",nIter)
input("Press return to exit")
```

```
b = 0.673556113751 m
r = 0.753058635874 m
h = 0.336778062472 m
Surface area = 3.98375353114 m**2
Number of cycles = 4
```

Problem 19

```
## problem10_1_19
from powell import *
from gaussElimin import *
from numpy import array,zeros
# Notation: A1 = x[0], A2 = x[1], A3 = x[2]

def F(x):          # Merit function (material volume)
    return 4.0*x[0] + 5.0*x[1] + 3.0*x[2]

def response(x):  # Computes stresses
    load = 200.0e3
    a = zeros((3,3))
    b = zeros(3)
    s = zeros(3)
    a[0,0] = 1.0;      a[0,1] = 0.8
    a[1,1] = 0.6;      a[1,2] = 1.0
    a[2,0] = 3.2/x[0]; a[2,1] = -5.0/x[1]
    a[2,2] = 1.8/x[2]
    b[0] = load; b[1] = load
```

```
    P = gaussElimin(a,b)
    for i in range(3): s[i] = P[i]/x[i]
    return s

def Fstar(x):     # Penalized merit function
    sAll = 150.0e6
    s = response(x)
    c1 = max(0.0, s[0] - sAll)
    c2 = max(0.0, s[1] - sAll)
    c3 = max(0.0, s[2] - sAll)
    return F(x) + lam*(c1**2 + c2**2 + c3**2)

lam = 1.0
xStart = array([0.001, 0.001, 0.001])
x,nIter = powell(Fstar,xStart,0.000005)
print("Areas    =",x,"m**2")
print("Stresses =",response(x)*1.0e-6,"MPa")
print("Volume   =", F(x),"m**3")
print("Num. of cycles =", nIter)
input ("Press return to exit")
```

```
Areas    = [ 0.00052032  0.00101627  0.00072357] m**2
Stresses = [ 150.  150.  150.] MPa
Volume   = 0.00933333333334 m**3
Num. of cycles = 4
```

In this problem the minimum weight is the fully-stressed design (all members are stressed to their allowable limits), but this is not always the case.
The numerical solution obtained above is *not unique*. It is not hard to show that the number of solutions is infinite, all of them being fully stressed and having the same weight. Therefore, the design produced by the program is completely dependent on the starting point.

Problem 20

```
## problem10_1_20
from powell import *
from math import pi, sqrt,sin,cos
from numpy import array

def V(x):      # Potential energy
    return -1.0e3*(60.0*sin(x[0]) + 45.0*sin(x[1]))
```

```
def c(x):      # Constraints
    return \
    array([1.2*cos(x[0]) + 1.5*cos(x[1]) + cos(x[2]) - 3.5, \
           1.2*sin(x[0]) + 1.5*sin(x[1]) + sin(x[2])])

def F(x):      # Penalized merit function
    lam = 1.0e9
    c1,c2 = c(x)
    return V(x) + lam*(c1**2 + c2**2)

xStart = array([0.5, 0.0, -0.5])
x,nIter = powell(F,xStart,0.01)
print("Angles =",x*180.0/pi,"deg")
print("Potential enegy =",V(x),"N.m")
print("Constraints =",c(x))
print("Number of cycles =",nIter)
input ("Press return to exit")
```

```
Angles = [ 21.36211695   1.24916744 -28.02127941] deg
Potential enegy = -22836.682421722835
Constraints = [ -2.70762308e-05   1.44096196e-05]
Number of cycles = 9
```

Usually a large penalty multiplier λ results in many iterative cycles. This is not the case here thanks to good starting values of the angles.

Alternative solution

As pointed out in Art. 10.1, solutions to optimization problems with equality constraints may also be obtained by the method of Lagrange multipliers.
We have the merit function

$$F = (-6.0 \sin\theta_1 - 4.5 \sin\theta_2) \times 10^4$$

and the constraints

$$\begin{aligned} g_1 &= 1.2\cos\theta_1 + 1.5\cos\theta_2 + \cos\theta_3 - 3.5 = 0 && \text{(a)} \\ g_2 &= 1.2\sin\theta_1 + 1.5\sin\theta_2 + \sin\theta_3 = 0 && \text{(b)} \end{aligned}$$

Since F can be scaled without affecting θ_1 and θ_2, we drop the factor 10^4. The equations $\nabla F^*(\mathbf{x}) = 0$ are

$$\begin{aligned} -6.0\cos\theta_1 - 1.2\lambda_1\sin\theta_1 + 1.2\lambda_2\cos\theta_1 &= 0 \\ -4.5\cos\theta_2 - 1.5\lambda_1\sin\theta_2 + 1.5\lambda_2\cos\theta_2 &= 0 \\ -\lambda_1\sin\theta_3 + \lambda_2\cos\theta_3 &= 0 \end{aligned}$$

These equations together with Eqs. (a) and (b) are coded in the function below using the notation $\mathbf{x} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \lambda_1 & \lambda_2 \end{bmatrix}^T$.

```
## problem10_1_20_lagrange
from newtonRaphson2 import *
from math import pi
from numpy import array,sin,cos

def F(x):
    s = sin(x); c = cos(x);
    y = array([-6.0*c[0] - 1.2*x[3]*s[0] + 1.2*x[4]*c[0], \
               -4.5*c[1] - 1.5*x[3]*s[1] + 1.5*x[4]*c[1], \
               -x[3]*s[2] + x[4]*c[2], \
               1.2*c[0] + 1.5*c[1] + c[2] - 3.5, \
               1.2*s[0] + 1.5*s[1] + s[2]])
    return y

xStart = array([0.5, 0.1, -0.5, 0.0, 0.0])
x = newtonRaphson2(F,xStart)
print("Angles =",x[0:3]*180.0/pi,"deg")
print("Multipliers =",x[3:5])
input ("Press return to exit")
```

```
Angles = [ 21.36042458   1.24887315 -28.01957224] deg
Multipliers = [-5.41566993  2.88193622]
```

These angles are reasonably close to the values obtained with Powell's method.

Problem 21

With $A_3 = A_1$ the displacement equations become

$$\frac{E}{4L}\begin{bmatrix} 6A_1 & 2\sqrt{3}A_1 \\ 2\sqrt{3}A_1 & 2A_1 + 8A_2 \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} P \\ 2P \end{bmatrix} \tag{a}$$

Introducing the dimensionless variables

$$x_i = \frac{E\delta}{PL}A_i \qquad u' = \frac{u}{\delta} \qquad v' = \frac{v}{\delta} \tag{b}$$

we can write Eqs. (a) as

$$\begin{bmatrix} 6x_1 & 2\sqrt{3}x_1 \\ 2\sqrt{3}x_1 & 2x_1 + 8x_2 \end{bmatrix}\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} 4 \\ 8 \end{bmatrix} \tag{c}$$

The structural volume is

$$V = 2LA_1 + 0.5LA_2 = \frac{PL^2}{2E\delta}(4x_1 + x_2)$$

Our task is to minimize $4x_1 + x_2$ subject to the constraints $u' \leq 1$ and $v' \leq 1$.

```
## Problem10_1_21
from numpy import zeros,array
from math import sqrt
from gaussElimin import *
from downhill import *

def displ(x):  # Set up and solve displacement eqs.
    a = zeros((2,2))
    a[0,0] = 6.0*x[0]; a[0,1] = 2.0*sqrt(3.0)*x[0]
    a[1,0] = a[0,1];   a[1,1] = 2.0*x[0] + 8.0*x[1]
    b = array([4.0,8.0])
    return gaussElimin(a,b)

def volume(x): return 4.0*x[0] + x[1]

def F(x):      # Penalized merit function
    [u,v] = displ(x)
    c1 = max(0.0,u - 1.0)
    c2 = max(0.0,v - 1.0)
    return volume(x) + lam*(c1**2 + c2**2)

lam = 10000.0  # Small multiplier will not work here
xStart = array([1.0,1.0])
x = downhill(F,xStart)
print('x =',x)
print('[u_prime,v_prime] =',displ(x))
print('Volume =',volume(x))
input ("Press return to exit")
```

The output of the program is

```
x = [ 0.4226254   0.71127814]
[u_prime,v_prime] = [ 1.00005288  1.00006569]
Volume = 2.40177974151
```

Note that u' and v' are close enough to the constraints $u' \leq 1$ and $v' \leq 1$. Therefore, the optimal design is

$$A_1 = 0.4226\frac{PL}{E\delta} \qquad A_2 = 0.7113\frac{PL}{E\delta} \quad \blacktriangleleft$$

with the structural volume

$$V = \frac{PL^2}{2E\delta}(2.402) = 1.201\frac{PL^2}{E\delta}$$

Alternative Solution If we had known beforehand that both displacement constraints are active at optimal design (a pretty safe bet), we could have substituted $u' = v' = 1$ into Eqs. (c) and solved for x_1 and x_2, thereby bypassing the optimization procedure.

Problem 22

The displacement equations are

$$\frac{E}{4L}\begin{bmatrix} 3A_1 + 3A_3 & \sqrt{3}A_1 + \sqrt{3}A_3 \\ \sqrt{3}A_1 + \sqrt{3}A_3 & A_1 + 8A_2 + A_3 \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} P \\ 2P \end{bmatrix}$$

Using the nondimensional variables in Eqs. (b) of Problem 21, the these equations become

$$\begin{bmatrix} 3x_1 + 3x_3 & \sqrt{3}x_1 + \sqrt{3}x_3 \\ \sqrt{3}x_1 + \sqrt{3}x_3 & x_1 + 8x_2 + x_3 \end{bmatrix}\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} 4 \\ 8 \end{bmatrix}$$

The structural volume is

$$V = LA_1 + \frac{L}{2}A_2 + LA_3 = \frac{PL^2}{2E\delta}(2x_1 + x_2 + 2x_3)$$

The task of optimization is to minimize $4x_1 + x_2$ subject to the constraints $u' \leq 1$ and $v' \leq 1$. We also need the additional constraints $x_i \geq 0$ to avoid negative cross-sectional areas.

```
## Problem10_1_22
from numpy import zeros,array,sum
from math import sqrt
from gaussElimin import *
from downhill import *

def displ(x):  # Set up and solve displacement eqs.
    a = zeros((2,2))
    a[0,0] = 3.0*x[0] + 3.0*x[2]; a[0,1] = sqrt(3.0)*(x[0] + x[2])
    a[1,0] = a[0,1];              a[1,1] = x[0] + 8.0*x[1] + x[2]
    b = array([4.0,8.0])
    return gaussElimin(a,b)

def volume(x): return 2.0*x[0] + x[1] + 2.0*x[2]

def F(x):      # Penalized merit function
    c = zeros(5)
```

```
    [u,v] = displ(x)
    c[0] = max(0.0,u - 1.0)
    c[1] = max(0.0,v - 1.0)
    for i in range (2,5):
        c[i] = min(0.0,x[i-2])
    return volume(x) + lam*(sum(c**2))

lam = 10000.0  # Small multiplier will not work here
xStart = array([1.0,1.0,1.0])
x = downhill(F,xStart)
print('x =',x)
print('[u_prime,v_prime] =',displ(x))
print('Volume =',volume(x))
input ("Press return to exit")
```

The output of the program is

```
x = [ 0.81594725  0.71127842  0.02930379]
[u_prime,v_prime] = [ 1.00005266  1.0000653 ]
Volume = 2.4017804998
```

Since the constraints $u' \leq 1$ and $v' \leq 1$ are very close to being satisfied, the design is satisfactory. The optimal cross-sectional areas are

$$A_1 = 0.8159\frac{PL}{E\delta} \qquad A_2 = 0.7113\frac{PL}{E\delta} \qquad A_3 = 0.0293\frac{PL}{E\delta} \quad \blacktriangleleft$$

The structural volume is

$$V = \frac{PL^2}{2E\delta}(2.402) = 1.201\frac{PL^2}{tE\delta}$$

which is the same as in Problem 21.

Note Added Note that A_3 is quite small, which begs the question: can it be removed altogether? The answer is "yes". The result would be a statically determinate truss of about the same strucural volume as the one above.

Problem 23

We have to maximize $I = bh^3/12$ with the constraint $b^2 + h^2 = d^2$. Introducing the variables $x_1 = b/d$ and $x_2 = h/d$, the problem becomes:

$$\text{minimize } F = -x_1 x_2^3 \text{ subject to } x_1^2 + x_2^2 - 1 = 0$$

```
## problem10_1_23
from numpy import array
from downhill import *

def c(x):  # Constraint (must be zero)
    return x[0]**2 + x[1]**2 - 1.0

def F(x):  # Penalized merit function
    return -x[0]*(x[1]**3) + lam*(c(x)**2)

lam = 10000.0
xStart = array([1.0,1.0])
x = downhill(F,xStart)
print('x =',x)
print('Constraint =',c(x))
input("Press return to exit")
```

```
x = [ 0.50000746  0.86603985]
Constraint = 3.24933545905e-005
```

The optimal dimansions are

$$b = \frac{1}{2}d \qquad h = \frac{\sqrt{3}}{2}d \ \blacktriangleleft$$

the moment of inertia being

$$I = \frac{1}{12}\left(\frac{\sqrt{3}}{2}\right)^3 d^4 = \frac{\sqrt{3}}{32}d^4$$

<u>Alternative solution</u>:
The equations to be solved are—see Eqs. (10.2):

$$\begin{aligned}\frac{\partial}{\partial x_1}\left[F(x) + \lambda g(x)\right] &= \frac{\partial}{\partial x_1}\left[-x_1 x_2^3 + \lambda(x_1^2 + x_2^2 - 1)\right] \\ &= -x_2^3 + 2\lambda x_1 = 0\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial x_2}\left[F(x) + \lambda g(x)\right] &= \frac{\partial}{\partial x_2}\left[-x_1 x_2^3 + \lambda(x_1^2 + x_2^2 - 1)\right] \\ &= -3x_1 x_2^2 + 2\lambda x_2 = 0\end{aligned}$$

$$g(x) = x_1^2 + x_2^2 - 1 = 0$$

Letting $\lambda = x_0$ results in the following program:

```
## problem 10_1_23_calc
from numpy import array
from newtonRaphson2 import *

def F(x):
    y = array([-x[2]**3 + 2.0*x[0]*x[1],           \
               -3.0*x[1]*x[2]**2 + 2.0*x[0]*x[2], \
               + x[1]**2 + x[2]**2 - 1.0])
    return y

xStart = array([1.0, 1.0, 1.0])
x = newtonRaphson2(F,xStart)
print('x =',x[1:3])
input("Press return to exit")
```

The result is

```
x = [ 0.5        0.8660254]
```