CHAPTER 11

11.1 The matrix to be evaluated is

$$\begin{bmatrix} 10 & 2 & -1 \\ -3 & -6 & 2 \\ 1 & 1 & 5 \end{bmatrix}$$

First, compute the *LU* decomposition. Multiply the first row by f_{21} = –3/10 = –0.3 and subtract the result from the second row to eliminate the a_{21} term. Then, multiply the first row by f_{31} = 1/10 = 0.1 and subtract the result from the third row to eliminate the a_{31} term. The result is

$$\begin{bmatrix} 10 & 2 & -1 \\ 0 & -5.4 & 1.7 \\ 0 & 0.8 & 5.1 \end{bmatrix}$$

Multiply the second row by f_{32} = 0.8/(–5.4) = –0.148148 and subtract the result from the third row to eliminate the a_{32} term.

$$\begin{bmatrix} 10 & 2 & -1 \\ 0 & -5.4 & 1.7 \\ 0 & 0 & 5.351852 \end{bmatrix}$$

Therefore, the *LU* decomposition is

$$[L]\{U\} = \begin{bmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.1 & -0.148148 & 1 \end{bmatrix}\begin{bmatrix} 10 & 2 & -1 \\ 0 & -5.4 & 1.7 \\ 0 & 0 & 5.351852 \end{bmatrix}$$

The first column of the matrix inverse can be determined by performing the forward-substitution solution procedure with a unit vector (with 1 in the first row) as the right-hand-side vector. Thus, the lower-triangular system, can be set up as,

$$\begin{bmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.1 & -0.148148 & 1 \end{bmatrix}\begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix}$$

and solved with forward substitution for $\{d\}^T$ = [1 0.3 –0.055556]. This vector can then be used as the right-hand side of the upper triangular system,

$$\begin{bmatrix} 10 & 2 & -1 \\ 0 & -5.4 & 1.7 \\ 0 & 0 & 5.351852 \end{bmatrix}\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0.3 \\ -0.055556 \end{Bmatrix}$$

which can be solved by back substitution for the first column of the matrix inverse,

$$[A]^{-1} = \begin{bmatrix} 0.110727 & 0 & 0 \\ -0.058824 & 0 & 0 \\ -0.010381 & 0 & 0 \end{bmatrix}$$

To determine the second column, Eq. (9.8) is formulated as

$$\begin{bmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.1 & -0.148148 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix}$$

This can be solved with forward substitution for $\{d\}^T = [0\ 1\ 0.148148]$, and the results are used with $[U]$ to determine $\{x\}$ by back substitution to generate the second column of the matrix inverse,

$$[A]^{-1} = \begin{bmatrix} 0.110727 & 0.038062 & 0 \\ -0.058824 & -0.176471 & 0 \\ -0.010381 & 0.027682 & 0 \end{bmatrix}$$

Finally, the same procedures can be implemented with $\{b\}^T = [0\ 0\ 1]$ to solve for $\{d\}^T = \lfloor 0 \quad 0 \quad 1 \rfloor$, and the results are used with $[U]$ to determine $\{x\}$ by back substitution to generate the third column of the matrix inverse,

$$[A]^{-1} = \begin{bmatrix} 0.110727 & 0.038062 & 0.00692 \\ -0.058824 & -0.176471 & 0.058824 \\ -0.010381 & 0.027682 & 0.186851 \end{bmatrix}$$

This result can be checked by multiplying it times the original matrix to give the identity matrix. The following MATLAB session can be used to implement this check,

```
>> A = [10 2 -1;-3 -6 2;1 1 5];
>> AI = [0.110727 0.038062 0.00692;
-0.058824 -0.176471 0.058824;
-0.010381 0.027682 0.186851];
>> A*AI

ans =
    1.0000   -0.0000   -0.0000
    0.0000    1.0000   -0.0000
   -0.0000    0.0000    1.0000
```

11.2 The system can be written in matrix form as

$$[A] = \begin{bmatrix} -8 & 1 & -2 \\ 2 & -6 & -1 \\ -3 & -1 & 7 \end{bmatrix} \qquad \{b\} = \begin{Bmatrix} -38 \\ -34 \\ -20 \end{Bmatrix}$$

Forward eliminate

$f_{21} = 2/(-8) = -0.25$ $\qquad$ $f_{31} = -3/(-8) = 0.375$

$$[A] = \begin{bmatrix} -8 & 1 & -2 \\ 0 & -5.75 & -1.5 \\ 0 & -1.375 & 7.75 \end{bmatrix}$$

Forward eliminate

f_{32} = –1.375/(–5.75) = 0.23913

$$[A] = \begin{bmatrix} -8 & 1 & -2 \\ 0 & -5.75 & -1.5 \\ 0 & 0 & 8.108696 \end{bmatrix}$$

Therefore, the *LU* decomposition is

$$[L]\{U\} = \begin{bmatrix} 1 & 0 & 0 \\ -0.25 & 1 & 0 \\ 0.375 & 0.23913 & 1 \end{bmatrix} \begin{bmatrix} -8 & 1 & -2 \\ 0 & -5.75 & -1.5 \\ 0 & 0 & 8.108696 \end{bmatrix}$$

The first column of the matrix inverse can be determined by performing the forward-substitution solution procedure with a unit vector (with 1 in the first row) as the right-hand-side vector. Thus, the lower-triangular system, can be set up as,

$$\begin{bmatrix} 1 & 0 & 0 \\ -0.25 & 1 & 0 \\ 0.375 & 0.23913 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix}$$

and solved with forward substitution for $\{d\}^T$ = [1 0.25 -0.434783]. This vector can then be used as the right-hand side of the upper triangular system,

$$\begin{bmatrix} -8 & 1 & -2 \\ 0 & -5.75 & -1.5 \\ 0 & 0 & 8.108696 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0.25 \\ -0.434783 \end{Bmatrix}$$

which can be solved by back substitution for the first column of the matrix inverse,

$$[A]^{-1} = \begin{bmatrix} -0.115282 & 0 & 0 \\ -0.029491 & 0 & 0 \\ -0.053619 & 0 & 0 \end{bmatrix}$$

To determine the second column, Eq. (9.8) is formulated as

$$\begin{bmatrix} 1 & 0 & 0 \\ -0.25 & 1 & 0 \\ 0.375 & 0.23913 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix}$$

This can be solved with forward substitution for $\{d\}^T$ = [0 1 -0.23913], and the results are used with [*U*] to determine {*x*} by back substitution to generate the second column of the matrix inverse,

$$[A]^{-1} = \begin{bmatrix} -0.115282 & -0.013405 & 0 \\ -0.029491 & -0.16622 & 0 \\ -0.053619 & -0.029491 & 0 \end{bmatrix}$$

Finally, the same procedures can be implemented with $\{b\}^T = [0\ 0\ 1]$ to solve for $\{d\}^T = [0\ 0\ 1]$, and the results are used with $[U]$ to determine $\{x\}$ by back substitution to generate the third column of the matrix inverse,

$$[A]^{-1} = \begin{bmatrix} -0.115282 & -0.013405 & -0.034853 \\ -0.029491 & -0.16622 & -0.032172 \\ -0.053619 & -0.029491 & 0.123324 \end{bmatrix}$$

11.3 The following solution is generated with MATLAB.

(a)
```
>> A = [15 -3 -1;-3 18 -6;-4 -1 12];
>> format long
>> AI = inv(A)

AI =
   0.07253886010363   0.01278065630397   0.01243523316062
   0.02072538860104   0.06079447322971   0.03212435233161
   0.02590673575130   0.00932642487047   0.09015544041451
```

(b)
```
>> b = [4000 1500 2400]';
>> format short
>> c = AI*b

c =
  339.1710
  251.1917
  333.9896
```

(c) The impact of a load to reactor 3 on the concentration of reactor 1 is specified by the element a_{13}^{-1} = 0.0124352. Therefore, the increase in the mass input to reactor 3 needed to induce a 10 g/m^3 rise in the concentration of reactor 1 can be computed as

$$\Delta b_3 = \frac{10}{0.0124352} = 804.1667 \frac{\text{g}}{\text{d}}$$

(d) The decrease in the concentration of the third reactor will be

$$\Delta c_3 = 0.0259067(500) + 0.009326(250) = 12.9534 + 2.3316 = 15.285 \frac{\text{g}}{\text{m}^3}$$

11.4 The mass balances can be written and the result written in matrix form as

$$\begin{bmatrix} 9 & 0 & -3 & 0 & 0 \\ -4 & 4 & 0 & 0 & 0 \\ 0 & -2 & 9 & 0 & 0 \\ 0 & -1 & -6 & 9 & -2 \\ -5 & -1 & 0 & 0 & 6 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{Bmatrix} = \begin{Bmatrix} Q_{01}c_{01} \\ 0 \\ Q_{03}c_{03} \\ 0 \\ 0 \end{Bmatrix}$$

MATLAB can then be used to determine the matrix inverse

```
>> Q = [9 0 -3 0 0;-4 4 0 0 0;0 -2 9 0 0;0 -1 -6 9 -2;-5 -1 0 0 6];
>> inv(Q)
ans =

    0.1200    0.0200    0.0400         0         0
    0.1200    0.2700    0.0400         0         0
    0.0267    0.0600    0.1200         0         0
    0.0578    0.0837    0.0933    0.1111    0.0370
    0.1200    0.0617    0.0400         0    0.1667
```

The concentration in reactor 5 can be computed using the elements of the matrix inverse as in,

$$c_5 = a_{51}^{-1}Q_{01}c_{01} + a_{53}^{-1}Q_{03}c_{03} = 0.1200(6)10 + 0.0400(7)20 = 7.2 + 2.8 = 10$$

11.5 The problem can be written in matrix form as

$$\begin{bmatrix} 0.866 & 0 & -0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0.866 & 0 & 0 & 0 \\ -0.866 & -1 & 0 & -1 & 0 & 0 \\ -0.5 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & -0.866 & 0 & 0 & -1 \end{bmatrix} \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ H_2 \\ V_2 \\ V_3 \end{Bmatrix} = \begin{Bmatrix} F_{1,h} \\ F_{1,v} \\ F_{2,h} \\ F_{2,v} \\ F_{3,h} \\ F_{3,v} \end{Bmatrix}$$

MATLAB can then be used to solve for the matrix inverse,

```
>> A = [0.866 0 -0.5 0 0 0;
0.5 0 0.866 0 0 0;
-0.866 -1 0 -1 0 0;
-0.5 0 0 0 -1 0;
0 1 0.5 0 0 0;
0 0 -0.866 0 0 -1];
>> AI = inv(A)

AI =
    0.8660    0.5000         0         0         0         0
    0.2500   -0.4330         0         0    1.0000         0
   -0.5000    0.8660         0         0         0         0
   -1.0000    0.0000   -1.0000         0   -1.0000         0
   -0.4330   -0.2500         0   -1.0000         0         0
    0.4330   -0.7500         0         0         0   -1.0000
```

The forces in the members resulting from the two forces can be computed using the elements of the matrix inverse as in,

$$F_1 = a_{12}^{-1}F_{1,v} + a_{15}^{-1}F_{3,h} = 0.5(-2000) + 0(-500) = -1000 + 0 = -1000$$

$$F_2 = a_{22}^{-1}F_{1,v} + a_{25}^{-1}F_{3,h} = -0.433(-2000) + 1(-500) = 866 - 500 = 366$$
$$F_3 = a_{32}^{-1}F_{1,v} + a_{35}^{-1}F_{3,h} = 0.866(-2000) + 0(-500) = -1732 + 0 = -1732$$

11.6 The matrix can be scaled by dividing each row by the element with the largest absolute value

```
>> A = [8/(-10) 2/(-10) 1;1 1/(-9) 3/(-9);1 -1/15 6/15]

A =
   -0.8000   -0.2000    1.0000
    1.0000   -0.1111   -0.3333
    1.0000   -0.0667    0.4000
```

MATLAB can then be used to determine each of the norms,

```
>> norm(A,'fro')
ans =
    1.9920

>> norm(A,1)
ans =
    2.8000

>> norm(A,inf)
ans =
     2
```

11.7 Prob. 11.2:

```
>> A = [-8 1 -2;2 -6 -1;-3 -1 7];
>> norm(A,'fro')

ans =
    13

>> norm(A,inf)

ans =
    11
```

Prob. 11.3:

```
>> A = [15 -3 -1;-3 18 -6;-4 -1 12]
>> norm(A,'fro')

ans =
   27.6586

>> norm(A,inf)

ans =
    27
```

11.8 Spectral norm

```
>> A = [1 4 9 16 25;4 9 16 25 36;9 16 25 36 49;16 25 36 49 64;25 36 49 64 81];
>> cond(A)
ans =
  2.5510e+017
```

(b) Row-sum norm

```
>> cond(A,inf)

Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.250327e-018.
> In cond at 48

ans =
  9.8436e+017
```

11.9 (a) The matrix to be evaluated is

$$\begin{bmatrix} 16 & 4 & 1 \\ 4 & 2 & 1 \\ 49 & 7 & 1 \end{bmatrix}$$

The row-sum norm of this matrix is 49 + 7 + 1 = 57. The inverse is

$$\begin{bmatrix} -0.1667 & 0.1 & 0.0667 \\ 1.5 & -1.1 & -0.4 \\ -2.3333 & 2.8 & 0.5333 \end{bmatrix}$$

The row-sum norm of the inverse is $|-2.3333| + 2.8 + 0.5333 = 5.6667$. Therefore, the condition number is

Cond[A] = 57(5.6667) = 323

This can be verified with MATLAB,

```
>> A = [16 4 1;4 2 1;49 7 1];
>> cond(A,inf)
ans =
  323.0000
```

(b) Spectral norm:

```
>> A = [16 4 1;4 2 1;49 7 1];
>> cond(A)
ans =
  216.1294
```

Frobenius norm:

```
>> cond(A,'fro')
ans =
  217.4843
```

11.10 The spectral condition number can be evaluated as

```
>> A = hilb(10);
>> N = cond(A)
N =
  1.6025e+013
```

The digits of precision that could be lost due to ill-conditioning can be calculated as

```
>> c = log10(N)
c =
   13.2048
```

Thus, about 13 digits could be suspect. A right-hand side vector can be developed corresponding to a solution of ones:

```
>> b=[sum(A(1,:)); sum(A(2,:)); sum(A(3,:)); sum(A(4,:)); sum(A(5,:)); sum(A(6,:));
sum(A(7,:)); sum(A(8,:)); sum(A(9,:)); sum(A(10,:))]

b =
    2.9290
    2.0199
    1.6032
    1.3468
    1.1682
    1.0349
    0.9307
    0.8467
    0.7773
    0.7188
```

The solution can then be generated by left division

```
>> x = A\b

x =
    1.0000
    1.0000
    1.0000
    1.0000
    0.9999
    1.0003
    0.9995
    1.0005
    0.9997
    1.0001
```

The maximum and mean errors can be computed as

```
>> e=max(abs(x-1))

e =
  5.3822e-004

>> e=mean(abs(x-1))

e =
  1.8662e-004
```

Thus, some of the results are accurate to only about 3 to 4 significant digits. Because MATLAB represents numbers to 15 significant digits, this means that about 11 to 12 digits are suspect.

11.11 First, the Vandermonde matrix can be set up

```
>> x1 = 4;x2=2;x3=7;x4=10;x5=3;x6=5;
```

```
>> A = [x1^5 x1^4 x1^3 x1^2 x1 1;x2^5 x2^4 x2^3 x2^2 x2 1;x3^5 x3^4 x3^3 x3^2 x3
1;x4^5 x4^4 x4^3 x4^2 x4 1;x5^5 x5^4 x5^3 x5^2 x5 1;x6^5 x6^4 x6^3 x6^2 x6 1]

A =
        1024          256           64           16            4            1
          32           16            8            4            2            1
       16807         2401          343           49            7            1
      100000        10000         1000          100           10            1
         243           81           27            9            3            1
        3125          625          125           25            5            1
```

The spectral condition number can be evaluated as

```
>> N = cond(A)

N =
  1.4492e+007
```

The digits of precision that could be lost due to ill-conditioning can be calculated as

```
>> c = log10(N)

c =
    7.1611
```

Thus, about 7 digits might be suspect. A right-hand side vector can be developed corresponding to a solution of ones:

```
>> b=[sum(A(1,:));sum(A(2,:));sum(A(3,:));sum(A(4,:));sum(A(5,:)); sum(A(6,:))]

b =
        1365
          63
       19608
      111111
         364
        3906
```

The solution can then be generated by left division

```
>> format long
>> x=A\b

x =
   1.00000000000000
   0.99999999999991
   1.00000000000075
   0.99999999999703
   1.00000000000542
   0.99999999999630
```

The maximum and mean errors can be computed as

```
>> e = max(abs(x-1))
e =
    5.420774940034789e-012

>> e = mean(abs(x-1))
e =
    2.154110223528960e-012
```

Some of the results are accurate to about 12 significant digits. Because MATLAB represents numbers to about 15 significant digits, this means that about 3 digits are suspect. Thus, for this case, the condition number tends to exaggerate the impact of ill-conditioning.

11.12 (a) The solution can be developed using your own software or a package. For example, using MATLAB,

```
>> A=[13.422 0 0 0;
-13.422 12.252 0 0;
0 -12.252 12.377 0;
0 0 -12.377 11.797];
>> W=[750.5 300 102 30]';
>> AI=inv(A)

AI =
    0.0745         0         0         0
    0.0816    0.0816         0         0
    0.0808    0.0808    0.0808         0
    0.0848    0.0848    0.0848    0.0848

>> C=AI*W
C =
   55.9157
   85.7411
   93.1163
  100.2373
```

(b) The element of the matrix that relates the concentration of Havasu (lake 4) to the loading of Powell (lake 1) is $a_{41}^{-1} = 0.084767$. This value can be used to compute how much the loading to Lake Powell must be reduced in order for the chloride concentration of Lake Havasu to be 75 as

$$\Delta W_1 = \frac{\Delta c_4}{a_{41}^{-1}} = \frac{100.2373 - 75}{0.084767} = 297.725$$

(c) First, normalize the matrix to give

$$[A] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -0.91283 & 0 & 0 \\ 0 & -0.9899 & 1 & 0 \\ 0 & 0 & 1 & -0.95314 \end{bmatrix}$$

The column-sum norm for this matrix is 2. The inverse of the matrix can be computed as

$$[A]^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1.095495 & -1.09549 & 0 & 0 \\ 1.084431 & -1.08443 & 1 & 0 \\ 1.137747 & -1.13775 & 1.049165 & -1.04917 \end{bmatrix}$$

The column-sum norm for the inverse can be computed as 4.317672. The condition number is, therefore, 2(4.317672) = 8.635345. This means that less than 1 digit is suspect [$\log_{10}(8.635345) = 0.93628$]. Interestingly, if the original matrix is unscaled, the same condition number results.

11.13 (a) When MATLAB is used to determine the inverse, the following error message suggests that the matrix is ill-conditioned:

```
>> A=[1 2 3;4 5 6;7 8 9];
>> inv(A)
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.541976e-018.

ans =
  1.0e+016 *
   -0.4504    0.9007   -0.4504
    0.9007   -1.8014    0.9007
   -0.4504    0.9007   -0.4504

The high condition number reinforces this conclusion:

>> cond(A)
ans =
  3.8131e+016
```

(b) However, when one of the coefficients is changed slightly, the system becomes well-conditioned:

```
>> A=[1 2 3;4 5 6;7 8 9.1];
>> inv(A)
ans =
    8.3333  -19.3333   10.0000
  -18.6667   39.6667  -20.0000
   10.0000  -20.0000   10.0000

>> cond(A)
ans =
  994.8787
```

11.14 The five simultaneous equations can be set up as

$$
\begin{aligned}
1.6\times10^{9}p_1 + 8\times10^{6}p_2 + 4\times10^{4}p_3 + 200p_4 + p_5 &= 0.746 \\
3.90625\times10^{9}p_1 + 1.5625\times10^{7}p_2 + 6.25\times10^{4}p_3 + 250p_4 + p_5 &= 0.675 \\
8.1\times10^{9}p_1 + 2.7\times10^{7}p_2 + 9\times10^{4}p_3 + 300p_4 + p_5 &= 0.616 \\
2.56\times10^{10}p_1 + 6.4\times10^{7}p_2 + 16\times10^{4}p_3 + 400p_4 + p_5 &= 0.525 \\
6.25\times10^{10}p_1 + 1.25\times10^{8}p_2 + 25\times10^{4}p_3 + 500p_4 + p_5 &= 0.457
\end{aligned}
$$

MATLAB can then be used to solve for the coefficients,

```
>> format short g
>> A=[200^4 200^3 200^2 200 1
250^4 250^3 250^2 250 1
300^4 300^3 300^2 300 1
400^4 400^3 400^2 400 1
500^4 500^3 500^2 500 1]

A =
     1.6e+009        8e+006        40000          200            1
  3.9063e+009  1.5625e+007        62500          250            1
     8.1e+009     2.7e+007        90000          300            1
    2.56e+010     6.4e+007     1.6e+005          400            1
    6.25e+010    1.25e+008     2.5e+005          500            1

>> b=[0.746;0.675;0.616;0.525;0.457];
>> format long g
```

```
>> p=A\b
p =
     1.33333333333201e-012
    -4.53333333333155e-009
     5.29666666666581e-006
      -0.00317366666666649
          1.20299999999999

>> cond(A)
ans =
          11711898982423.4
```

Thus, because the condition number is so high, the system seems to be ill-conditioned. This implies that this might not be a very reliable method for fitting polynomials. Because this is generally true for higher-order polynomials, other approaches are commonly employed as will be described subsequently in Chap. 15.

11.15 (a) The balances for reactors 2 and 3 can be written as

$$Q_{2,in}c_{2,in} - Q_{2,1}c_2 + Q_{1,2}c_1 + Q_{3,2}c_3 - kV_2c_2 = 0$$
$$-Q_{3,2}c_3 - Q_{3,\text{out}}c_3 + Q_{1,3}c_1 - kV_3c_3 = 0$$

(b) The parameters can be substituted into the mass balances

$$100(10) - 5c_1 - 117c_1 + 22c_2 - 0.1(100)c_1 = 0$$
$$10(200) - 22c_2 + 5c_1 + 7c_3 - 0.1(50)c_2 = 0$$
$$-7c_3 - 110c_3 + 117c_1 - 0.1(150)c_3 = 0$$

Collecting terms

$$132c_1 - 22c_2 = 1000$$
$$-5c_1 + 27c_2 - 7c_3 = 2000$$
$$-117c_1 + 132c_3 = 0$$

or in matrix form

$$\begin{bmatrix} 132 & -22 & 0 \\ -5 & 27 & -7 \\ -117 & 0 & 132 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \end{Bmatrix} = \begin{Bmatrix} 1000 \\ 2000 \\ 0 \end{Bmatrix}$$

(c) Using MATLAB

```
>> A=[132 -22 0;-5 27 -7;-117 0 132];
>> [L,U]=lu(A)
L =
    1.0000         0         0
   -0.0379    1.0000         0
   -0.8864   -0.7452    1.0000
U =
  132.0000  -22.0000         0
         0   26.1667   -7.0000
         0         0  126.7834
```

(d)

```
>> b=[1;0;0]; d1=L\b
d1 =
   1.00000000000000
   0.03787878787879
   0.91459177764910
>> c1=U\d1
c1 =
   0.00813865862849
   0.00337740631637
   0.00721381105707
>> b=[0;1;0]; d2=L\b
d2 =
                  0
   1.00000000000000
   0.74522292993631
>> c2=U\d2
c2 =
   0.00663149962321
   0.03978899773926
   0.00587792012057
>> b=[0;0;1]; d3=L\b
d3 =
     0
     0
     1
>> c3=U\d3
c3 =
   0.00035167043456
   0.00211002260739
   0.00788746546094
>> Ainv=[c1 c2 c3]
Ainv =
   0.00813865862849   0.00663149962321   0.00035167043456
   0.00337740631637   0.03978899773926   0.00211002260739
   0.00721381105707   0.00587792012057   0.00788746546094
```

(e) (*i*)

```
>> b=[1000;2000;0];
>> c=Ainv*b
c =
  21.40165787490580
  82.95540179488936
  18.96965129821196
```

(*ii*) $\Delta c_1 = a_{12}^{-1} \times \Delta W_2 = (0.0066315) \times (-2000) = -13.263$

(*iii*) $c_3 = a_{31}^{-1} \times W_1 + a_{32}^{-1} \times W_2 = (0.0072138) \times (2000) + (0.0058779) \times (1000) = 20.3055$

11.16 (a) Here is a script to compute the matrix inverse:

```
K = [150 -100 0;-100 150 -50;0 -50 50]
KI = inv(K)

K =
   150  -100     0
  -100   150   -50
     0   -50    50
KI =
    0.0200    0.0200    0.0200
    0.0200    0.0300    0.0300
    0.0200    0.0300    0.0500
```

(b) $\Delta x_1 = a_{13}^{-1} \times \Delta m_1 g = 0.02 \times (100\times 9.81) = 19.62$ m

(c) The position of the third jumper as a function of an additional force applied to the third jumper can be formulated in terms of the matrix inverse as

$$x_3 = x_{3,\text{original}} + a_{33}^{-1}\Delta F_3$$

which can be solved for

$$\Delta F_3 = \frac{x_3 - x_{3,\text{original}}}{a_{33}^{-1}}$$

Recall from Example 8.2 that the original position of the third jumper was 131.6130 m. Thus, the additional force is

$$\Delta F_3 = \frac{140 - 131.6130}{0.05} = 167.74 \text{ N}$$

11.17 The current can be computed as

$$i_{52} = a_{25}^{-1} V_6 + a_{26}^{-1} V_1$$

The matrix inverse can be computed as

```
>> A=[1 1 1 0 0 0
0 -1 0 1 -1 0
0 0 -1 0 0 1
0 0 0 0 1 -1
0 10 -10 0 -15 -5
5 -10 0 -20 0 0];
>> AI=inv(A)

AI =
      0.84615      0.61538      0.76923      0.73077    0.0076923     0.030769
      0.11538     -0.46154    -0.076923     -0.17308     0.019231    -0.023077
     0.038462     -0.15385     -0.69231     -0.55769    -0.026923   -0.0076923
      0.15385      0.38462      0.23077      0.26923   -0.0076923    -0.030769
     0.038462     -0.15385      0.30769      0.44231    -0.026923   -0.0076923
     0.038462     -0.15385      0.30769     -0.55769    -0.026923   -0.0076923
```

Therefore,

$$i_{52} = 0.019231(200) - 0.023077(100) = 1.5385$$

11.18 (a) First, flow balances can be used to compute $Q_{13} = 100$, $Q_{23} = 100$, $Q_{34} = 150$, and $Q_{4,\text{out}} = 150$. Then, steady-state mass balances can be written for the rooms as

$$W_1 - Q_{12}c_1 - Q_{13}c_1 + E_{13}(c_3 - c_1)$$
$$W_2 + Q_{12}c_1 - Q_{23}c_2 + E_{23}(c_3 - c_2)$$
$$Q_{13}c_1 + Q_{23}c_2 - Q_{34}c_3 - Q_{3,\text{out}}c_3 + E_{13}(c_1 - c_3) + E_{23}(c_2 - c_3) + E_{34}(c_4 - c_3)$$
$$W_4 + Q_{34}c_3 - Q_{4,\text{out}}c_4 + E_{34}(c_3 - c_4)$$

Substituting parameters

$$150 - 50c_1 - 100c_1 + 50(c_3 - c_1)$$
$$2000 + 50c_1 - 100c_2 + 50(c_3 - c_2)$$
$$100c_1 + 100c_2 - 150c_3 - 50c_3 + 50(c_1 - c_3) + 50(c_2 - c_3) + 90(c_4 - c_3)$$
$$5000 + 150c_3 - 150c_4 + 90(c_3 - c_4)$$

Collecting terms and expressing in matrix form

$$\begin{bmatrix} 200 & 0 & -50 & 0 \\ -50 & 150 & -50 & 0 \\ -150 & -150 & 390 & -90 \\ 0 & 0 & -240 & 240 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{Bmatrix} = \begin{Bmatrix} 150 \\ 2000 \\ 0 \\ 5000 \end{Bmatrix}$$

(b) The matrix inverse can be computed and used to solve for the concentrations as

```
clear,clc
format short g
A=[200 0 -50 0;-50 150 -50 0;-150 -150 390 -90; 0 0 -240 240];
b=[150 2000 0 5000]';
AI=inv(A)
c=AI*b

AI =
      0.00625      0.00125      0.00125   0.00046875
      0.00375      0.00875    0.0020833   0.00078125
        0.005        0.005        0.005     0.001875
        0.005        0.005        0.005    0.0060417

c =
       5.7813
       21.969
       20.125
       40.958
```

(c) The concentration of the second room as a function of the change in load to the fourth room can be formulated in terms of the matrix inverse as

$$c_2 = c_{2,\text{original}} + a_{24}^{-1} \Delta W_4$$

which can be solved for

$$\Delta W_4 = \frac{c_2 - c_{2,\text{original}}}{a_{24}^{-1}}$$

Substituting values gives

$$\Delta W_4 = \frac{20 - 21.969}{0.00078125} = -2520$$