CHAPTER 10

10.1 Matrix multiplication is distributive

$$[L]\{[U]\{X\}-\{D\}\}=[A]\{X\}-\{B\}$$

$$[L][U]\{X\}-[L]\{D\}=[A]\{X\}-\{B\}$$

Therefore, equating like terms,

$$[L][U]\{X\}=[A]\{X\}$$

$$[L]\{D\}=\{B\}$$

$$[L][U]=[A]$$

10.2 (a) The coefficient a_{21} is eliminated by multiplying row 1 by f_{21} = –3/10 = –0.3 and subtracting the result from row 2. a_{31} is eliminated by multiplying row 1 by f_{31} = 1/10 = 0.1 and subtracting the result from row 3. The factors f_{21} and f_{31} can be stored in a_{21} and a_{31}.

$$\begin{bmatrix} 10 & 2 & -1 \\ -0.3 & -5.4 & 1.7 \\ 0.1 & 0.8 & 5.1 \end{bmatrix}$$

a_{32} is eliminated by multiplying row 2 by f_{32} = 0.8/(–5.4) = –0.14815 and subtracting the result from row 3. The factor f_{32} can be stored in a_{32}.

$$\begin{bmatrix} 10 & 2 & -1 \\ -0.3 & -5.4 & 1.7 \\ 0.1 & -0.14815 & 5.351852 \end{bmatrix}$$

Therefore, the *LU* decomposition is

$$[L]=\begin{bmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.1 & -0.14815 & 1 \end{bmatrix} \qquad [U]=\begin{bmatrix} 10 & 2 & -1 \\ 0 & -5.4 & 1.7 \\ 0 & 0 & 5.351852 \end{bmatrix}$$

These two matrices can be multiplied to yield the original system. For example, using MATLAB to perform the multiplication gives

```
>> L=[1 0 0;-.3 1 0;0.1 -.14815 1];
>> U=[10 2 -1;0 -5.4 1.7;0 0 5.351852];
>> L*U

ans =
   10.0000    2.0000   -1.0000
   -3.0000   -6.0000    2.0000
    1.0000    1.0000    5.0000
```

(b) Forward substitution: $[L]\{D\} = \{B\}$

$$\begin{bmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.1 & -0.14815 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 27 \\ 61.5 \\ -21.5 \end{Bmatrix}$$

Solving yields $d_1 = 27$, $d_2 = -53.4$, and $d_3 = -32.1111$.

Back substitution:

$$\begin{bmatrix} 10 & 2 & -1 \\ 0 & -5.4 & 1.7 \\ 0 & 0 & 5.351852 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 27 \\ -53.4 \\ -32.1111 \end{Bmatrix}$$

$$x_3 = \frac{-32.1111}{5.351852} = -6$$

$$x_2 = \frac{-53.4 - 1.7(-6)}{-5.4} = 8$$

$$x_1 = \frac{27 - (-1)(-6) - 2(8)}{10} = 0.5$$

(c) Forward substitution: $[L]\{D\} = \{B\}$

$$\begin{bmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.1 & -0.14815 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 12 \\ 18 \\ -6 \end{Bmatrix}$$

Solving yields $d_1 = 12$, $d_2 = 21.6$, and $d_3 = -4$.

Back substitution:

$$\begin{bmatrix} 10 & 2 & -1 \\ 0 & -5.4 & 1.7 \\ 0 & 0 & 5.351852 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 12 \\ 21.6 \\ -4 \end{Bmatrix}$$

$$x_3 = \frac{-4}{5.351852} = -0.7474$$

$$x_2 = \frac{21.6 - 1.7(-0.7474)}{-5.4} = -4.23529$$

$$x_1 = \frac{12 - (-1)(-0.7474) - 2(-4.23529)}{10} = 1.972318$$

10.3 (a) The coefficient a_{21} is eliminated by multiplying row 1 by $f_{21} = -2/8 = -0.25$ and subtracting the result from row 2. a_{31} is eliminated by multiplying row 1 by $f_{31} = 2/8 = 0.25$ and subtracting the result from row 3. The factors f_{21} and f_{31} can be stored in a_{21} and a_{31}.

$$\begin{bmatrix} 8 & 4 & -1 \\ -0.25 & 6 & 0.75 \\ 0.25 & -2 & 6.25 \end{bmatrix}$$

a_{32} is eliminated by multiplying row 2 by $f_{32} = -2/6 = -0.33333$ and subtracting the result from row 3. The factor f_{32} can be stored in a_{32}.

$$\begin{bmatrix} 8 & 4 & -1 \\ -0.25 & 6 & 0.75 \\ 0.25 & -0.33333 & 6.5 \end{bmatrix}$$

Therefore, the *LU* decomposition is

$$[L] = \begin{bmatrix} 1 & 0 & 0 \\ -0.25 & 1 & 0 \\ 0.25 & -0.33333 & 1 \end{bmatrix} \qquad [U] = \begin{bmatrix} 8 & 4 & -1 \\ 0 & 6 & 0.75 \\ 0 & 0 & 6.5 \end{bmatrix}$$

Forward substitution: $[L]\{D\} = \{B\}$

$$\begin{bmatrix} 1 & 0 & 0 \\ -0.25 & 1 & 0 \\ 0.25 & -0.33333 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 11 \\ 4 \\ 7 \end{Bmatrix}$$

Solving yields $d_1 = 11$, $d_2 = 6.75$, and $d_3 = 6.5$.

Back substitution:

$$\begin{bmatrix} 8 & 4 & -1 \\ 0 & 6 & 0.75 \\ 0 & 0 & 6.5 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 11 \\ 6.75 \\ 6.5 \end{Bmatrix}$$

$$x_3 = \frac{6.5}{6.5} = 1$$

$$x_2 = \frac{6.75 - 0.75(1)}{6} = 1$$

$$x_1 = \frac{11 - (-1)(1) - 4(1)}{8} = 1$$

(b) The first column of the inverse can be computed by using $[L]\{D\} = \{B\}$

$$\begin{bmatrix} 1 & 0 & 0 \\ -0.25 & 1 & 0 \\ 0.25 & -0.33333 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix}$$

This can be solved for $d_1 = 1$, $d_2 = 0.25$, and $d_3 = -0.16667$. Then, we can implement back substitution

$$\begin{bmatrix} 8 & 4 & -1 \\ 0 & 6 & 0.75 \\ 0 & 0 & 6.5 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0.25 \\ -0.16667 \end{Bmatrix}$$

to yield the first column of the inverse

$$\{X\} = \begin{Bmatrix} 0.099359 \\ 0.0448718 \\ -0.025641 \end{Bmatrix}$$

For the second column use $\{B\}^T = \{0\ 1\ 0\}$ which gives $\{D\}^T = \{0\ 1\ 0.33333\}$. Back substitution then gives $\{X\}^T = \{-0.073718\ 0.160256\ 0.051282\}$.

For the third column use $\{B\}^T = \{0\ 0\ 1\}$ which gives $\{D\}^T = \{0\ 0\ 1\}$. Back substitution then gives $\{X\}^T = \{0.028846\ -0.01923\ 0.153846\}$.

Therefore, the matrix inverse is

$$[A]^{-1} = \begin{bmatrix} 0.099359 & -0.073718 & 0.028846 \\ 0.044872 & 0.160256 & -0.019231 \\ -0.025641 & 0.051282 & 0.153846 \end{bmatrix}$$

We can verify that this is correct by multiplying $[A][A]^{-1}$ to yield the identity matrix. For example, using MATLAB,

```
>> A=[8 4 -1;-2 5 1;2 -1 6];
>> AI=[0.099359 -0.073718 0.028846;
0.044872 0.160256 -0.019231;
-0.025641 0.051282 0.153846]
>> A*AI

ans =
    1.0000   -0.0000   -0.0000
    0.0000    1.0000   -0.0000
         0         0    1.0000
```

10.4 As the system is set up, we must first pivot by switching the first and third rows of $[A]$. Note that we must make the same switch for the right-hand-side vector $\{B\}$

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

The coefficient a_{21} is eliminated by multiplying row 1 by $f_{21} = -3/-8 = 0.375$ and subtracting the result from row 2. a_{31} is eliminated by multiplying row 1 by $f_{31} = 2/(-8) = -0.25$ and subtracting the result from row 3. The factors f_{21} and f_{31} can be stored in a_{21} and a_{31}.

$$[A] = \begin{bmatrix} -8 & 1 & -2 \\ 0.375 & -1.375 & 7.75 \\ -0.25 & -5.75 & -1.5 \end{bmatrix}$$

Next, we pivot by switching rows 2 and 3. Again, we must also make the same switch for the right-hand-side vector $\{B\}$

$$[A] = \begin{bmatrix} -8 & 1 & -2 \\ -0.25 & -5.75 & -1.5 \\ 0.375 & -1.375 & 7.75 \end{bmatrix} \qquad \{B\} = \begin{Bmatrix} -20 \\ -38 \\ -34 \end{Bmatrix}$$

a_{32} is eliminated by multiplying row 2 by $f_{32} = -1.375/(-5.75) = 0.23913$ and subtracting the result from row 3. The factor f_{32} can be stored in a_{32}.

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

Therefore, the *LU* decomposition is

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

Forward substitution: $[L]\{D\} = \{B\}$

$$\begin{bmatrix} 1 & 0 & 0 \\ -0.25 & 1 & 0 \\ 0.375 & 0.23913 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} -20 \\ -38 \\ -34 \end{Bmatrix}$$

Solving yields $d_1 = -20$, $d_2 = -43$, and $d_3 = -16.2174$.

Back substitution:

$$\begin{bmatrix} -8 & 1 & -2 \\ 0 & -5.75 & -1.5 \\ 0 & 0 & 8.108696 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} -20 \\ -43 \\ -16.2174 \end{Bmatrix}$$

$$x_3 = \frac{-16.2174}{8.108696} = -2$$

$$x_2 = \frac{-43 + 1.5(-2)}{-5.75} = 8$$

$$x_1 = \frac{-20 + 2(-2) - 8}{-8} = 4$$

10.5 The flop counts for *LU* decomposition can be determined in a similar fashion as was done for Gauss elimination. The major difference is that the elimination is only implemented for the left-hand side coefficients. Thus, for every iteration of the inner loop, there are n multiplications/divisions and $n - 1$ addition/subtractions. The computations can be summarized as

Outer Loop k	Inner Loop i	Addition/Subtraction flops	Multiplication/Division flops
1	2, n	$(n-1)(n-1)$	$(n-1)n$
2	3, n	$(n-2)(n-2)$	$(n-2)(n-1)$
.	.		
.	.		
.	.		
k	$k+1$, n	$(n-k)(n-k)$	$(n-k)(n+1-k)$
.	.		
.	.		
.	.		
$n-1$	n, n	(1)(1)	(1)(2)

Therefore, the total addition/subtraction flops for elimination can be computed as

$$\sum_{k=1}^{n-1}(n-k)(n-k) = \sum_{k=1}^{n-1}\left[n^2 - 2nk + k^2\right]$$

Applying some of the relationships from Eq. (8.14) yields

$$\sum_{k=1}^{n-1}\left[n^2 - 2nk + k^2\right] = \frac{n^3}{3} - \frac{n^2}{2} + \frac{n}{6}$$

A similar analysis for the multiplication/division flops yields

$$\sum_{k=1}^{n-1}(n-k)(n+1-k) = \frac{n^3}{3} - \frac{n}{3}$$

Summing these results gives

$$\frac{2n^3}{3} - \frac{n^2}{2} - \frac{n}{6}$$

For forward substitution, the numbers of multiplications and subtractions are the same and equal to

$$\sum_{i=1}^{n-1} i = \frac{(n-1)n}{2} = \frac{n^2}{2} - \frac{n}{2}$$

Back substitution is the same as for Gauss elimination: $n^2/2 - n/2$ subtractions and $n^2/2 + n/2$ multiplications/divisions. The entire number of flops can be summarized as

	Mult/Div	**Add/Subtr**	**Total**
Forward elimination	$\frac{n^3}{3} - \frac{n}{3}$	$\frac{n^3}{3} - \frac{n^2}{2} + \frac{n}{6}$	$\frac{2n^3}{3} - \frac{n^2}{2} - \frac{n}{6}$
Forward substitution	$\frac{n^2}{2} - \frac{n}{2}$	$\frac{n^2}{2} - \frac{n}{2}$	$n^2 - n$
Back substitution	$\frac{n^2}{2} + \frac{n}{2}$	$\frac{n^2}{2} - \frac{n}{2}$	n^2
Total	$\frac{n^3}{3} + n^2 - \frac{n}{3}$	$\frac{n^3}{3} + \frac{n^2}{2} - \frac{5n}{6}$	$\frac{2n^3}{3} + \frac{3n^2}{2} - \frac{7n}{6}$

Thus, the total number of flops is identical to that obtained with standard Gauss elimination.

10.6 First, we compute the *LU* decomposition. The coefficient a_{21} is eliminated by multiplying row 1 by $f_{21} = -3/10 = -0.3$ and subtracting the result from row 2. a_{31} is eliminated by multiplying row 1 by $f_{31} = 1/10 = 0.1$ and subtracting the result from row 3. The factors f_{21} and f_{31} can be stored in a_{21} and a_{31}.

$$\begin{bmatrix} 10 & 2 & -1 \\ -0.3 & -5.4 & 1.7 \\ 0.1 & 0.8 & 5.1 \end{bmatrix}$$

a_{32} is eliminated by multiplying row 2 by $f_{32} = 0.8/(-5.4) = -0.148148$ and subtracting the result from row 3. The factor f_{32} can be stored in a_{32}.

$$\begin{bmatrix} 10 & 2 & -1 \\ -0.3 & -5.4 & 1.7 \\ 0.1 & -0.148148 & 5.351852 \end{bmatrix}$$

Therefore, the *LU* decomposition is

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

The first column of the inverse can be computed by using $[L]\{D\} = \{B\}$

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

This can be solved for $d_1 = 1$, $d_2 = 0.3$, and $d_3 = -0.055556$. Then, we can implement back substitution

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

to yield the first column of the inverse

$$\{X\} = \begin{Bmatrix} 0.110727 \\ -0.058824 \\ -0.0103806 \end{Bmatrix}$$

For the second column use $\{B\}^T = \{0\ 1\ 0\}$ which gives $\{D\}^T = \{0\ 1\ 0.148148\}$. Back substitution then gives $\{X\}^T = \{0.038062\ -0.176471\ 0.027682\}$.

For the third column use $\{B\}^T = \{0\ 0\ 1\}$ which gives $\{D\}^T = \{0\ 0\ 1\}$. Back substitution then gives $\{X\}^T = \{0.00692\ 0.058824\ 0.186851\}$.

Therefore, the matrix inverse is

$$[A]^{-1} = \begin{bmatrix} 0.110727 & 0.038062 & 0.006920 \\ -0.058824 & -0.176471 & 0.058824 \\ -0.010381 & 0.027682 & 0.186851 \end{bmatrix}$$

We can verify that this is correct by multiplying $[A][A]^{-1}$ to yield the identity matrix. For example, using MATLAB,

```
>> A=[10 2 -1;-3 -6 2;1 1 5];
>> AI=[0.110727 0.038062 0.006920;
-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
```

10.7 Equation 10.17 yields

$$l_{11} = 2 \qquad l_{21} = -1 \qquad l_{31} = 1$$

Equation 10.18 gives

$$u_{12} = \frac{a_{12}}{l_{11}} = -3 \qquad u_{13} = \frac{a_{13}}{l_{11}} = 0.5$$

Equation 10.19 gives

$$l_{22} = a_{22} - l_{21}u_{12} = 4 \qquad l_{32} = a_{32} - l_{31}u_{12} = 0$$

Equation 10.20 gives

$$u_{23} = \frac{a_{23} - l_{21}u_{13}}{l_{22}} = -0.125$$

Equation 10.21 gives

$$l_{33} = a_{33} - l_{31}u_{13} - l_{32}u_{23} = 1.5$$

Therefore, the *LU* decomposition is

$$[L] = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 4 & 0 \\ 1 & 0 & 1.5 \end{bmatrix} \qquad [U] = \begin{bmatrix} 1 & -3 & 0.5 \\ 0 & 1 & -0.125 \\ 0 & 0 & 1 \end{bmatrix}$$

These two matrices can be multiplied to yield the original system. For example, using MATLAB to perform the multiplication gives

```
>> L=[2 0 0;-1 4 0;1 0 1.5];
>> U=[1 -3 0.5;0 1 -0.125;0 0 1];
>> L*U

ans =
     2    -6     1
    -1     7    -1
     1    -3     2
```

10.8 (a) Using MATLAB, the matrix inverse can be computed as

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

AI =
    0.0725    0.0128    0.0124
    0.0207    0.0608    0.0321
    0.0259    0.0093    0.0902
```

(b)

```
>> B=[3800;1200;2350];
>> C=AI*B

C =
```

```
320.2073
227.2021
321.5026
```

(c) $\Delta W_3 = \dfrac{\Delta c_1}{a_{13}^{-1}} = \dfrac{10}{0.012435} = 804.1667$

(d) $\Delta c_3 = a_{31}^{-1}\Delta W_1 + a_{32}^{-1}\Delta W_2 = 0.025907(-500) + 0.009326(-250) = -15.285$

10.9 First we can scale the matrix to yield

$$[A] = \begin{bmatrix} -0.8 & -0.2 & 1 \\ 1 & -0.11111 & -0.33333 \\ 1 & -0.06667 & 0.4 \end{bmatrix}$$

Frobenius norm:

$$\|A\|_e = \sqrt{3.967901} = 1.991959$$

In order to compute the column-sum and row-sum norms, we can determine the sums of the absolute values of each of the columns and rows:

			row sums ↓
-0.8	-0.2	1	**2**
1	-0.11111	-0.33333	**1.44444**
1	-0.06667	0.4	**1.46667**
2.8	**0.37778**	**1.73333**	**←column sums**

Therefore, $\|A\|_1 = 2.8$ and $\|A\|_\infty = 2$.

10.10 For the system from Prob. 10.3, we can scale the matrix to yield

$$[A] = \begin{bmatrix} 1 & 0.5 & -0.125 \\ -0.4 & 1 & 0.2 \\ 0.33333 & -0.16667 & 1 \end{bmatrix}$$

Frobenius norm:

$$\|A\|_e = \sqrt{3.604514} = 1.898556$$

In order to compute the row-sum norm, we can determine the sum of the absolute values of each of the rows:

	row sums ↓

1	0.5	-0.125	**1.625**
-0.4	1	0.2	**1.6**
0.333333	-0.16667	1	**1.5**

Therefore, $\|A\|_\infty = 1.625$.

For the system from Prob. 10.4, we can scale the matrix to yield

$$[A] = \begin{bmatrix} -0.3333 & 1 & 0.16667 \\ -0.42857 & -0.14286 & 1 \\ 1 & -0.125 & 0.25 \end{bmatrix}$$

Frobenius norm:

$$\|A\|_e = \sqrt{3.421096} = 1.84962$$

In order to compute the row-sum norm, we can determine the sum of the absolute values of each of the rows:

			row sums ↓
-0.33333	1	0.166667	**1.5**
-0.42857	-0.14286	1	**1.571429**
1	-0.125	0.25	**1.375**

Therefore, $\|A\|_\infty = 1.571429$.

10.11 In order to compute the row-sum norm, we can determine the sum of the absolute values of each of the rows:

				row sums ↓
0.125	0.25	0.5	1	**1.875**
0.015625	0.625	0.25	1	**1.890625**
0.00463	0.02777	0.16667	1	**1.19907**
0.001953	0.015625	0.125	1	**1.142578**

Therefore, $\|A\|_\infty = 1.890625$ The matrix inverse can then be computed. For example, using MATLAB,

```
>> A=[0.125 0.25 0.5 1;
0.015625 0.625 0.25 1;
0.00463 0.02777 0.16667 1;
0.001953 0.015625 0.125 1]
>> AI=inv(A)

AI =
   10.2329   -2.2339  -85.3872   77.3883
```

```
    -0.1008     1.7674    -4.3949     2.7283
    -0.6280    -0.3716    30.7645   -29.7649
     0.0601     0.0232    -3.6101     4.5268
```

The row-sum norm can then be computed by determining the sum of the absolute values of each of the rows. The result is $\|A^{-1}\|_\infty = 175.2423$. Therefore, the condition number can be computed as

$$\text{Cond}[A] = 1.890625(175.2423) = 331.3174$$

This corresponds to $\log_{10}(331.3174) = 2.52$ suspect digits.

10.12 (a) In order to compute the row-sum norm, we can determine the sum of the absolute values of each of the rows:

					row sums ↓
1	4	9	16	25	**55**
4	9	16	25	36	**90**
9	16	25	36	49	**135**
16	25	36	49	64	**190**
25	36	49	64	81	**255**

Therefore, $\|A\|_\infty = 255$. The matrix inverse can then be computed. For example, using MATLAB,

```
>> 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];
>> AI=inv(A)
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 9.944077e-019.
AI =
  1.0e+015 *
   -0.2800    0.6573   -0.2919   -0.2681    0.1827
    0.5211   -1.3275    0.8562    0.1859   -0.2357
    0.1168    0.0389   -0.8173    1.0508   -0.3892
   -0.6767    1.2756    0.2335   -1.5870    0.7546
    0.3189   -0.6443    0.0195    0.6184   -0.3124
```

Notice that MATLAB alerts us that the matrix is ill-conditioned. This is also strongly suggested by the fact that the elements are so large.

The row-sum norm can then be computed by determining the sum of the absolute values of each of the rows. The result is $\|A^{-1}\|_\infty = 4.5274\times 10^{15}$. Therefore, the condition number can be computed as

$$\text{Cond}[A] = 255(4.5274 \times 10^{15}) = 1.1545 \times 10^{18}$$

This corresponds to $\log_{10}(1.1545 \times 10^{18}) = 18.06$ suspect digits. Thus, the suspect digits are more than the number of significant digits for the double precision representation used in MATLAB (15-16 digits). Consequently, we can conclude that this matrix is highly ill-conditioned.

It should be noted that if you used Excel for this computation you would have arrived at a slightly different result of Cond[A] = 1.263×10^{18}.

(b) First, the matrix is scaled. For example, using MATLAB,

```
>> A=[1/25 4/25 9/25 16/25 25/25;
4/36 9/36 16/36 25/26 36/36;
9/49 16/49 25/49 36/49 49/49;
16/64 25/64 36/64 49/64 64/64;
25/81 36/81 49/81 64/81 81/81]

A =
    0.0400    0.1600    0.3600    0.6400    1.0000
    0.1111    0.2500    0.4444    0.9615    1.0000
    0.1837    0.3265    0.5102    0.7347    1.0000
    0.2500    0.3906    0.5625    0.7656    1.0000
    0.3086    0.4444    0.6049    0.7901    1.0000
```

The row-sum norm can be computed as 3.1481. Next, we can invert the matrix,

```
>> AI=inv(A)
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 2.230462e-018.

AI =
  1.0e+016 *
   -0.0730         0    0.8581   -1.4945    0.7093
    0.1946   -0.0000   -2.2884    3.9852   -1.8914
   -0.1459         0    1.7163   -2.9889    1.4186
   -0.0000    0.0000   -0.0000   -0.0000    0.0000
    0.0243   -0.0000   -0.2860    0.4982   -0.2364
```

The row-sum norm of the inverse can be computed as 8.3596×10^{16}. The condition number can then be computed as

$$\text{Cond}[A] = 3.1481(8.3596 \times 10^{16}) = 2.6317 \times 10^{17}$$

This corresponds to $\log_{10}(2.6317 \times 10^{17}) = 17.42$ suspect digits. Thus, as with **(a)**, the suspect digits are more than the number of significant digits for the double precision representation used in MATLAB (15-16 digits). Consequently, we again can conclude that this matrix is highly ill-conditioned.

It should be noted that if you used Excel for this computation you would have arrived at a slightly different result of Cond[A] = 1.3742×10^{17}.

10.13 In order to compute the row-sum norm of the normalized 5×5 Hilbert matrix, we can determine the sum of the absolute values of each of the rows:

					row sums ↓
1	0.500000	0.333333	0.250000	0.200000	**2.283333**
1	0.666667	0.500000	0.400000	0.333333	**2.9**
1	0.750000	0.600000	0.500000	0.428571	**3.278571**
1	0.800000	0.666667	0.571429	0.500000	**3.538095**
1	0.833333	0.714286	0.625000	0.555556	**3.728175**

Therefore, $\|A\|_\infty = 3.728175$. The matrix inverse can then be computed and the row sums calculated as

					row sums ↓
25	-150	350	-350	126	**1001**
-300	2400	-6300	6720	-2520	**18240**
1050	-9450	26460	-29400	11340	**77700**
-1400	13440	-39200	44800	-17640	**116480**
630	-6300	18900	-22050	8820	**56700**

The result is $\|A^{-1}\|_\infty = 116{,}480$. Therefore, the condition number can be computed as

$$\text{Cond}[A] = 3.728175(116{,}480) = 434{,}258$$

This corresponds to $\log_{10}(434{,}258) = 5.64$ suspect digits.

10.14 The matrix to be evaluated can be computed by substituting the *x* values into the Vandermonde matrix to give

$$[A] = \begin{bmatrix} 16 & 4 & 1 \\ 4 & 2 & 1 \\ 49 & 7 & 1 \end{bmatrix}$$

We can then scale the matrix by dividing each row by its maximum element,

$$[A] = \begin{bmatrix} 1 & 0.25 & 0.0625 \\ 1 & 0.5 & 0.25 \\ 1 & 0.142857 & 0.020408 \end{bmatrix}$$

In order to compute the row-sum norm, we can determine the sum of the absolute values of each of the rows:

			row sums ↓
1	0.25	0.0625	**1.3125**

1	0.5	0.25	**1.75**
1	0.142857	0.020408	**1.163265**

Therefore, $\|A\|_\infty = 1.75$. The matrix inverse can then be computed and the row sums calculated as

			row sums ↓
-2.66667	0.4	3.266667	**6.333333**
24	-4.4	-19.6	**48**
-37.3333	11.2	26.13333	**74.66667**

The result is $\|A^{-1}\|_\infty = 74.66667$. Therefore, the condition number can be computed as

$$\text{Cond}[A] = 1.75(74.66667) = 130.6667$$

This result can be checked with MATLAB,

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

ans =
  130.6667
```

(b) MATLAB can be used to compute the spectral and Frobenius condition numbers,

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

ans =
  102.7443

>> cond(A,'fro')

ans =
  104.2971
```

[Note: If you did not scale the original matrix, the results are: $\text{Cond}[A]_\infty = 323$, $\text{Cond}[A]_2 = 216.1294$, and $\text{Cond}[A]_e = 217.4843$]

10.15 Here is a VBA program that implements *LU* decomposition. It is set up to solve Example 10.1.

```
Option Explicit

Sub LUDTest()
Dim n As Integer, er As Integer, i As Integer, j As Integer
```

```
Dim a(3, 3) As Double, b(3) As Double, x(3) As Double
Dim tol As Double
n = 3
a(1, 1) = 3: a(1, 2) = -0.1: a(1, 3) = -0.2
a(2, 1) = 0.1: a(2, 2) = 7: a(2, 3) = -0.3
a(3, 1) = 0.3: a(3, 2) = -0.2: a(3, 3) = 10
b(1) = 7.85: b(2) = -19.3: b(3) = 71.4
tol = 0.000001
Call LUD(a, b, n, x, tol, er)
'output results to worksheet
Sheets("Sheet1").Select
Range("a3").Select
For i = 1 To n
  ActiveCell.Value = x(i)
  ActiveCell.Offset(1, 0).Select
Next i
Range("a3").Select

End Sub

Sub LUD(a, b, n, x, tol, er)
Dim i As Integer, j As Integer
Dim o(3) As Double, s(3) As Double
Call Decompose(a, n, tol, o, s, er)
If er = 0 Then
  Call Substitute(a, o, n, b, x)
Else
  MsgBox "ill-conditioned system"
  End
End If
End Sub

Sub Decompose(a, n, tol, o, s, er)
Dim i As Integer, j As Integer, k As Integer
Dim factor As Double
For i = 1 To n
  o(i) = i
  s(i) = Abs(a(i, 1))
  For j = 2 To n
    If Abs(a(i, j)) > s(i) Then s(i) = Abs(a(i, j))
  Next j
Next i
For k = 1 To n - 1
  Call Pivot(a, o, s, n, k)
  If Abs(a(o(k), k) / s(o(k))) < tol Then
    er = -1
    Exit For
  End If
  For i = k + 1 To n
    factor = a(o(i), k) / a(o(k), k)
    a(o(i), k) = factor
    For j = k + 1 To n
      a(o(i), j) = a(o(i), j) - factor * a(o(k), j)
    Next j
  Next i
Next k
If (Abs(a(o(k), k) / s(o(k))) < tol) Then er = -1
End Sub

Sub Pivot(a, o, s, n, k)
Dim ii As Integer, p As Integer
Dim big As Double, dummy As Double
```

```
p = k
big = Abs(a(o(k), k) / s(o(k)))
For ii = k + 1 To n
  dummy = Abs(a(o(ii), k) / s(o(ii)))
  If dummy > big Then
    big = dummy
    p = ii
  End If
Next ii
dummy = o(p)
o(p) = o(k)
o(k) = dummy
End Sub

Sub Substitute(a, o, n, b, x)
Dim k As Integer, i As Integer, j As Integer
Dim sum As Double, factor As Double
For k = 1 To n - 1
  For i = k + 1 To n
    factor = a(o(i), k)
    b(o(i)) = b(o(i)) - factor * b(o(k))
  Next i
Next k
x(n) = b(o(n)) / a(o(n), n)
For i = n - 1 To 1 Step -1
  sum = 0
  For j = i + 1 To n
    sum = sum + a(o(i), j) * x(j)
  Next j
  x(i) = (b(o(i)) - sum) / a(o(i), i)
Next i
End Sub
```

10.16 Here is a VBA program that uses *LU* decomposition to determine the matrix inverse. It is set up to solve Example 10.3.

```
Option Explicit

Sub LUInverseTest()
Dim n As Integer, er As Integer, i As Integer, j As Integer
Dim a(3, 3) As Double, b(3) As Double, x(3) As Double
Dim tol As Double, ai(3, 3) As Double
n = 3
a(1, 1) = 3: a(1, 2) = -0.1: a(1, 3) = -0.2
a(2, 1) = 0.1: a(2, 2) = 7: a(2, 3) = -0.3
a(3, 1) = 0.3: a(3, 2) = -0.2: a(3, 3) = 10
tol = 0.000001
Call LUDminv(a, b, n, x, tol, er, ai)
If er = 0 Then
  Range("a1").Select
  For i = 1 To n
    For j = 1 To n
      ActiveCell.Value = ai(i, j)
      ActiveCell.Offset(0, 1).Select
    Next j
    ActiveCell.Offset(1, -n).Select
  Next i
  Range("a1").Select
Else
  MsgBox "ill-conditioned system"
End If
```

```
End Sub

Sub LUDminv(a, b, n, x, tol, er, ai)
Dim i As Integer, j As Integer
Dim o(3) As Double, s(3) As Double
Call Decompose(a, n, tol, o, s, er)
If er = 0 Then
  For i = 1 To n
    For j = 1 To n
      If i = j Then
        b(j) = 1
      Else
        b(j) = 0
      End If
    Next j
    Call Substitute(a, o, n, b, x)
    For j = 1 To n
      ai(j, i) = x(j)
    Next j
  Next i
End If
End Sub

Sub Decompose(a, n, tol, o, s, er)
Dim i As Integer, j As Integer, k As Integer
Dim factor As Double
For i = 1 To n
  o(i) = i
  s(i) = Abs(a(i, 1))
  For j = 2 To n
    If Abs(a(i, j)) > s(i) Then s(i) = Abs(a(i, j))
  Next j
Next i
For k = 1 To n - 1
  Call Pivot(a, o, s, n, k)
  If Abs(a(o(k), k) / s(o(k))) < tol Then
    er = -1
    Exit For
  End If
  For i = k + 1 To n
    factor = a(o(i), k) / a(o(k), k)
    a(o(i), k) = factor
    For j = k + 1 To n
      a(o(i), j) = a(o(i), j) - factor * a(o(k), j)
    Next j
  Next i
Next k
If (Abs(a(o(k), k) / s(o(k))) < tol) Then er = -1
End Sub

Sub Pivot(a, o, s, n, k)
Dim ii As Integer, p As Integer
Dim big As Double, dummy As Double
p = k
big = Abs(a(o(k), k) / s(o(k)))
For ii = k + 1 To n
  dummy = Abs(a(o(ii), k) / s(o(ii)))
  If dummy > big Then
    big = dummy
    p = ii
  End If
Next ii
```

```
dummy = o(p)
o(p) = o(k)
o(k) = dummy
End Sub

Sub Substitute(a, o, n, b, x)
Dim k As Integer, i As Integer, j As Integer
Dim sum As Double, factor As Double
For k = 1 To n - 1
  For i = k + 1 To n
    factor = a(o(i), k)
    b(o(i)) = b(o(i)) - factor * b(o(k))
  Next i
Next k
x(n) = b(o(n)) / a(o(n), n)
For i = n - 1 To 1 Step -1
  sum = 0
  For j = i + 1 To n
    sum = sum + a(o(i), j) * x(j)
  Next j
  x(i) = (b(o(i)) - sum) / a(o(i), i)
Next i
End Sub
```

10.17 The problem can be set up as

$$2\Delta x_1 + 5\Delta x_2 + \Delta x_3 = -5 - (-3) = -2$$

$$6\Delta x_1 + 2\Delta x_2 + \Delta x_3 = 12 - 14 = -2$$

$$\Delta x_1 + 2\Delta x_2 + \Delta x_3 = 3 - 4 = -1$$

which can be solved for $\Delta x_1 = -0.2$, $\Delta x_2 = -0.26667$, and $\Delta x_3 = -0.26667$. These can be used to yield the corrected results

$$x_1 = 2 - 0.2 = 1.8$$
$$x_2 = -3 - 0.26667 = -3.26667$$
$$x_3 = 8 - 0.26667 = 7.73333$$

These results are exact.

10.18

$$\vec{A} \cdot \vec{B} = 0 \Rightarrow -4a + 2b = 3 \quad (1)$$
$$\vec{A} \cdot \vec{C} = 0 \Rightarrow 2a - 3c = -6 \quad (2)$$
$$\vec{B} \cdot \vec{C} = 2 \Rightarrow 3b + c = 10 \quad (3)$$

Solve the three equations using Matlab:

```
>> A=[-4 2 0; 2 0 -3; 0 3 1]
b=[3; -6; 10]
x=inv(A)*b

x = 0.525
    2.550
```

```
   2.350
```

Therefore, a = 0.525, b = 2.550, and c = 2.350.

10.19

$$(\vec{A}\times\vec{B}) = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ a & b & c \\ -2 & 1 & -4 \end{vmatrix} = (-4b-c)\vec{i} - (-4a+2c)\vec{j} + (a+2b)\vec{k}$$

$$(\vec{A}\times\vec{C}) = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ a & b & c \\ 1 & 3 & 2 \end{vmatrix} = (2b-3c)\vec{i} - (2a-c)\vec{j} + (3a-b)\vec{k}$$

$$(\vec{A}\times\vec{B}) + (\vec{A}\times\vec{C}) = (-2b-4c)\vec{i} - (-2a+c)\vec{j} + (4a+b)\vec{k}$$

Therefore,

$$(-2b-4c)\vec{i} + (2a-c)\vec{j} + (4a+b)\vec{k} = (5a+6)\vec{i} + (3b-2)\vec{j} + (-4c+1)\vec{k}$$

We get the following set of equations $\Rightarrow$

$$-2b-4c = 5a+6 \quad \Rightarrow \quad -5a-2b-4c = 6 \qquad (1)$$
$$2a-c = 3b-2 \quad \Rightarrow \quad 2a-3b-c = -2 \qquad (2)$$
$$4a+b = -4c+1 \quad \Rightarrow \quad 4a+b+4c = 1 \qquad (3)$$

<u>In Matlab:</u>

```
>> A=[-5  -2  -4 ; 2  -3  -1 ; 4  1  4];
>> B=[6 ; -2 ; 1];
>> x = A\B

x =
   -3.6522
   -3.3478
    4.7391
```

a = –3.6522, b = –3.3478, c = 4.7391

10.20

(I) $f(0)=1 \Rightarrow a(0)+b=1 \Rightarrow b=1$

$f(2)=1 \Rightarrow c(2)+d=1 \Rightarrow 2c+d=1$

(II) If f is continuous, then at x = 1

$$ax+b=cx+d \Rightarrow a(1)+b=c(1)+d \Rightarrow a+b-c-d=0$$

(III) $a+b=4$

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 4 \end{bmatrix}$$

These can be solved using MATLAB

```
>> A=[0 1 0 0;0 0 2 1;1 1 -1 -1;1 1 0 0];
>> B=[1;1;0;4];
>> A\B

ans =
     3
     1
    -3
     7
```

Thus, $a = 3$, $b = 1$, $c = –3$, and $d = 7$.

10.21 MATLAB provides a handy way to solve this problem.

(*a*)

```
>> a=hilb(3)

a =
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
    0.3333    0.2500    0.2000

>> x=[1 1 1]'

x =
     1
     1
     1

>> b=a*x

b =
    1.8333
    1.0833
    0.7833

>> format long e
>> x=a\b

x =
    9.999999999999992e-001
    1.000000000000006e+000
    9.999999999999939e-001
```

(*b*)

```
>> a=hilb(7);
>> x=ones(7,1);
>> b=a*x;
```

```
>> x=a\b

x =
    9.999999999927329e-001
    1.000000000289139e+000
    9.999999972198158e-001
    1.000000010794369e+000
    9.999999802287199e-001
    1.000000017073336e+000
    9.999999943967310e-001
```

(*c*)

```
>> a=hilb(10);
>> x=ones(10,1);
>> b=a*x;
>> x=a\b

x =
    9.999999993518614e-001
    1.000000053255573e+000
    9.999989124259656e-001
    1.000009539399602e+000
    9.999558816980565e-001
    1.000118062679701e+000
    9.998108238105067e-001
    1.000179021813331e+000
    9.999077593295230e-001
    1.000019946488826e+000
```