Lecture 1: NONLINEAR EQUATIONS

We will start with the case of one equation with one unknown, which can be always written as

$$f(x) = 0$$

Finding all solutions graphically is easy: we plot $f(x)$ against x (using a wide enough range of x values, to make sure we have not missed anything); then each time the graph crosses (or touches) the x axis, the corresponding x coordinate provides a solution to the equation. The only difficulty with this approach is the accuracy – we will be lucky to get the first two digits of the correct answer. But, this will be enough to give a good estimate of the solution, which can ten be refined (to any accuracy) by the so called

Newton's Method

Expanding the graph of $f(x)$ around the point of intersect and observing only a small segment of it, the result will look almost like a straight line (with only slight curvature). We will also assume that the graph crosses the x axis at a non-zero angle (rather then just touching it). It is then obvious (draw the corresponding picture) that our initial estimate of the root (say x_0) can be improved by fitting a straight line with a slope of $f'(x_0)$ trough the point $[x_0, f(x_0)]$, and finding its intercept with the x axis. Since

$$y - f(x_0) = f'(x_0) \cdot (x - x_0)$$

is the equation of this straight line, solving

$$-f(x_0) = f'(x_0) \cdot (x - x_0)$$

for x yields the desired improved solution, namely:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

This yields a better, but not necessarily 10-digit accurate solution. But clearly, we can now apply the same idea again, using x_1 in place of x_0. This will result in

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

And again:

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}$$

etc., until the answers no longer change (within our usual, 10-digit accuracy).

One can show that this procedure is ***quadratically convergent***, which means that the number of correct digits roughly doubles in each ***iteration*** (one step of the procedure).

Example: To solve $e^x = 2 - x$

we plot the two functions (e^x and $2-x$) and note that they intersect in a point whose x coordinate is roughly equal to 1 (this is an alternate way of doing it – one does not even need Maple). Our x_0 is thus set to 1, $f(x) = e^x - 2 + x$ and $f'(x) = e^x + 1$. And we are ready to iterate:

$$\begin{aligned}
x_1 &= 1 - \frac{e - 2 + 1}{e + 1} = 0.53788\,28428 \\
x_2 &= x_1 - \frac{e^{x_1} - 2 + x_1}{e^{x_1} + 1} = 0.44561\,67486 \\
x_3 &= x_2 - \frac{e^{x_2} - 2 + x_2}{e^{x_2} + 1} = 0.44285\,67246 \\
x_4 &= x_3 - \frac{e^{x_3} - 2 + x_3}{e^{x_3} + 1} = 0.44285\,44011 \\
x_5 &= x_4 - \frac{e^{x_4} - 2 + x_4}{e^{x_4} + 1} = 0.44285\,44011
\end{aligned}$$

at which point the value no longer changes. Quadratic convergence is be clearly observed.

The method fails (convergence becomes extremely slow) when $f(x)$ only touches the x axis, without crossing it (this is an indication that both $f(x)$ and $f'(x)$ have a root at that point), or crosses it at 0 angle (an indication that $f''(x)$ is also equal to zero at that point). The best way to overcome this problem is: As soon as we notice it (from graph, or slow convergence of the standard technique), we switch to solving $f'(x) = 0$ instead. If that is still giving trouble, solve $f''(x) = 0$, etc. In the end (to be on the save side), we should verify that the final answer does meet $f(x) = 0$.

Example: We know that $f(x) = 1+\sin x$ has a root at $x = 1.5\,\pi = 4.\,71238\,8981$ ($\equiv 270^o$). If we try to find it by the regular technique (starting at $x_0 = 5$), we get

$$\begin{aligned}
x_1 &= x_0 - \frac{1+\sin x_0}{\cos x_0} = 4.\,85519\,4921\\
x_2 &= x_1 - \frac{1+\sin x_1}{\cos x_1} = 4.\,78367\,0356\\
x_3 &= x_2 - \frac{1+\sin x_2}{\cos x_2} = 4.\,74801\,457\\
x_4 &= x_3 - \frac{1+\sin x_3}{\cos x_3} = 4.\,73019\,9891\\
x_5 &= x_4 - \frac{1+\sin x_4}{\cos x_4} = 4.\,72129\,4199\\
x_6 &= x_5 - \frac{1+\sin x_5}{\cos x_5} = 4.\,71684\,156\\
x_7 &= x_6 - \frac{1+\sin x_6}{\cos x_6} = 4.\,71461\,527\\
&\vdots
\end{aligned}$$

Even though the procedure seems to converge to the correct answer, the process would obviously take forever.

If instead we solve $(1 + \sin x)' = \cos x = 0$, we get

$$\begin{aligned}
x_1 &= x_0 + \frac{\cos x_0}{\sin x_0} = 4.\,70418\,7084\\
x_2 &= x_1 + \frac{\cos x_1}{\sin x_1} = 4.\,71238\,9164\\
x_3 &= x_2 + \frac{\cos x_2}{\sin x_2} = 4.\,71238\,898\\
x_4 &= x_3 + \frac{\cos x_3}{\sin x_3} = 4.\,71238\,898
\end{aligned}$$

we get the exact answer in 3 iterations (even though an extra iteration is needed to confirm that). One can now easily verify that

$$1 + \sin 4.\,71238\,898 = 0$$

We now have a technique for finding roots of our orthogonal polynomials which, luckily, all must have ***simple real roots*** spreading over the original (A, B) interval (otherwise, polynomial roots require special consideration – we will not go into that).

Example: When dealing with the third-degree Laguerre polynomial, we had to rely onMaple to get its three roots. Now, we can do this on our own. Plotting

$$x^3 - 9x^2 + 18x - 6 = 0$$

indicates that there is a root near $x_0 = 6$. We thus get

$$x_1 = x_0 - \frac{x_0^3 - 9x_0^2 + 18x_0 - 6}{3x_0^2 - 18x_0 + 18} = \mathbf{6.}\,33333\,3333$$

$$x_2 = x_1 - \frac{x_1^3 - 9x_1^2 + 18x_1 - 6}{3x_1^2 - 18x_1 + 18} = \mathbf{6.29071}\,5373$$

$$x_3 = x_2 - \frac{x_2^3 - 9x_2^2 + 18x_2 - 6}{3x_2^2 - 18x_2 + 18} = \mathbf{6.28994}\,5332$$

$$x_4 = x_3 - \frac{x_3^3 - 9x_3^2 + 18x_3 - 6}{3x_3^2 - 18x_3 + 18} = \mathbf{6.28994\,5083}$$

Based on our experience with quadratic convergence, we don't have to verify that the last answer is correct.

Once we have a root of a cubic equation, we can ***deflate*** the polynomial by carrying out the following ***synthetic division:***

$$(x^3 - 9x^2 + 18x - 6) \div (x - 6.\,28994\,5083) =$$
$$x^2 - 2.710054917\,x + 0.9539034002$$

The remaining two roots can then be found easily to be:

$$\frac{2.710054917}{2} + \sqrt{(\frac{2.710054917}{2})^2 - 0.9539034002} = 2.\,29428\,0362$$

$$\frac{2.710054917}{2} - \sqrt{(\frac{2.710054917}{2})^2 - 0.9539034002} = 0.41577\,45565$$

in agreement with our previous example.

We will now extend this technique to solve the case of

Several Unknowns

We will start by trying to solve two nonlinear equations with two unknowns, which we will call x_1 and x_2 (collectively x, using a vector notation). The two equations can be always written in the following form:

$F_1(x_1, x_2) = 0$
$F_2(x_1, x_2) = 0$

where each F is now a function of two variables (our unknowns). The issue of finding a reasonably accurate starting (initial) solution is now a lot more difficult- plotting the two functions is still possible (even though now we need two **3D** plots), but extracting the information we need is a lot more difficult (and even this approach will ultimately fail, when we have 3 or more unknowns). There are various techniques capable of a ***global search*** (in n dimensional space, where n is the number of unknowns) for a good starting point; due to a lack of time, we have to bypass this step and simply assume that a reasonably good estimate is provided to us.

All we need to do then is to adapt the Newton's technique to two and more variables. Assuming that we have a pair of initial values x_{10} and x_{20} , each of the F functions can be (Taylor) expanded, around this point, as follows:

$$F_1(x_1, x_2) = F_1(x_{1_0}, x_{2_0}) + \frac{\partial F_1(x_{1_0}, x_{2_0})}{\partial x_1}(x_1 - x_{1_0}) + \frac{\partial F_1(x_{1_0}, x_{2_0})}{\partial x_2}(x_2 - x_{2_0}) + \ldots$$

$$F_2(x_1, x_2) = F_2(x_{1_0}, x_{2_0}) + \frac{\partial F_2(x_{1_0}, x_{2_0})}{\partial x_1}(x_1 - x_{1_0}) + \frac{\partial F_2(x_{1_0}, x_{2_0})}{\partial x_2}(x_2 - x_{2_0}) + \ldots$$

or, using a matrix notation,

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}(\mathbf{x}_{(0)}) + \left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\mathbf{x}_{(0)}} (\mathbf{x} - \mathbf{x}_{(0)}) + \ldots \qquad (9.1)$$

where $x_{(0)}$ is a column vector with components x_{10} and x_{20} , and $\partial F/\partial x$ denotes, rather symbolically, the following matrix of partial derivatives (called the ***Jacobian***)

$$\frac{\partial \mathbf{F}}{\partial \mathbf{x}} \equiv \begin{bmatrix} \dfrac{\partial F_1}{\partial x_1} & \dfrac{\partial F_1}{\partial x_2} \\ \dfrac{\partial F_2}{\partial x_1} & \dfrac{\partial F_2}{\partial x_2} \end{bmatrix}$$

of each equation (one per row) differentiated with respect of each variable (one per column). Making the left hand side of (9.1) equal to a zero vector, and solving for ***x*** yields:

$$\mathbf{x}_{(1)} = \mathbf{x}_{(0)} - \left[\left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\mathbf{x}_{(0)}}\right]^{-1} \mathbf{F}(\mathbf{x}_{(0)})$$

which constitutes one iteration. In this spirit we can continue:

$$\mathbf{x}_{(2)} = \mathbf{x}_{(1)} - \left[\left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\mathbf{x}_{(1)}}\right]^{-1} \mathbf{F}(\mathbf{x}_{(1)})$$

etc., until convergence is reached.

Example: Solve

$$x_1 \cos x_2 + 0.716 = 0$$
$$x_2 \sin x_1 - x_1^2 - 1.305 = 0$$

starting with $x_{10} = 1$ and $x_{20} = 3$. The Jacobian is clearly

$$\begin{bmatrix} \cos x_2 & -x_1 \sin x_2 \\ x_2 \cos x_1 - 2x_1 & \sin x_1 \end{bmatrix}$$

Evaluating the left hand side of each of our equations (using the initial values of x_1 and x_2) yields

$$\mathbf{F}_{(0)} = \begin{bmatrix} -0.27399\,24966 \\ 0.21941\,29540 \end{bmatrix}$$

similarly evaluating the ***Jacobian*** results in

$$\mathbb{J}_{(0)} = \begin{bmatrix} -0.98999\,24966 & -0.14112\,00081 \\ -0.37909\,3082 & 0.84147\,09848 \end{bmatrix}$$

We can now compute the values of x_{11} and x_{21} as the two components of

$$\mathbf{x}_{(1)} = \mathbf{x}_{(0)} - [\mathbb{J}_{(0)}]^{-1}\mathbf{F}_{(0)} = \begin{bmatrix} 0.77486\,46744 \\ 2.\,63782\,4472 \end{bmatrix}$$

where $x_{(0)}$ is the initial vector (with components 1 and 3). This completes the first iteration. We can now repeat the process, until the two values no longer change.

Even though we can still manage to do this with a calculator, we have clearly reached the point at which it may be better to delegate the routine but tedious computation to Maple. This can be done as follows:

```
F := [ x[1]*cos(x[2]) + 0.716, x[2]*sin(x[1]) - x[1]^2 - 1.305 ];

J := matrix(2, 2):

for i to 2 do  for j to 2 do  J[i, j] := diff(F[i], x[j]) end do  end do:

x := [1., 3.]:

with(linalg):

x := evalm(x -  linsolve(J, F) );
```

The last line computes the $x_{(1)}$ vector (instead of $\mathbf{J^{-1}F}$, we solve the equivalent set of equations). We find it convenient not to introduce a new name, but call both $x_{(0)}$ and $x_{(1)}$ simply x. This has the advantage that, by re-executing the last line, we will automatically get $x_{(2)}$ etc. In our case, executing the last line five times yields:

[0.7748646744, 2.637824472]
[0.7825429930, 2.719825617]
[0.7854682773, 2.717842406]
[0.7854606220, 2.717875728]
[0.7854606228, 2.717875728]

at which point the procedure has clearly converged (note that, due to the round off error, the last digit of either component may keep on changing, sometimes indefinitely).

It should be clear how the formulas extend to the case of three or more unknowns. Let us do an example which was in an year's exam.

Example: We need to solve the following three equations (the algebraic, geometric and harmonic mean of three numbers is given, respectively, as):

$$\frac{x_1 + x_2 + x_3}{3} = 7$$

$$\sqrt[3]{x_1 x_2 x_3} = 4$$

$$\frac{3}{\frac{1}{x_1} + \frac{1}{x_2} + \frac{1}{x_3}} = \frac{16}{7}$$

For the initial values we will take $x_1 = 1.5$, $x_2 = 5$ *and* $x_3 = 10$. All we need to modify in our previous computer 'program' is the definition of F, thus (it does not hurt to simplify the equations):

```
F := [ x[1]+x[2]+x[3] −21, x[1] *x[2] *x[3] −64, 1/x[1]+1/x[2]+1/x[3] −21/16 ];
```

the dimensions (i.e. 3 instead of 2 in the next two lines), and the initial values. We will also need to rewrite the last line (to fix one of **Maple's** many quirks) as follows:

```
x := evalm(x – inverse(J)&* F );
```

As result, we get:

[.6989495801, 3.572619045, 16.72843138]
[.8746180084, 4.895879953, 15.22950204]
[.9791494539, 3.948814285, 16.07203626]
[.9992737556, 4.004281052, 15.99644519]
[.9999992176, 3.999999364, 16.00000142]
[.9999999991, 4.000000005, 16.00000000]

The exact answer thus seems to be $x_1 = 1$, $x_2 = 4$ *and* $x_3 = 16$ (this can be easily verified against the original equations).

If we used the equations in their original form, it would have taken us 9 iterations to reach the same conclusion.

The choice of the initial values is quite critical, see what would happen if we change x_{10} to 2.0 .

Lecture 2: Systems of Linear Algebraic Equations

Solve the simultaneous equations **Ax** = **b**

Notation

A system of algebraic equations has the form

$$\begin{aligned} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n &= b_1 \\ A_{21}x_1 + A_{22}x_2 + \cdots + A_{2n}x_n &= b_2 \\ A_{31}x_1 + A_{32}x_2 + \cdots + A_{3n}x_n &= b_3 \\ &\cdots \\ A_{n1}x_1 + A_{n2}x_2 + \cdots + A_{nn}x_n &= b_n \end{aligned} \qquad (2.1)$$

where the coefficients A_{ij} and the constants b_j are known, and x_i represent the unknowns. In matrix notation the equations are written as

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \qquad (2.2)$$

or, simply

$$\mathbf{Ax} = \mathbf{b} \qquad (2.3)$$

In summary, the modeling of linear systems invariably gives rise to equations of the form **Ax** = **b**, where **b** is the input and **x** represents the response of the system. The coefficient matrix **A**, which reflects the characteristics of the system, is independent of the input. In other words, if the input is changed, the equations have to be solved again with a different **b**, but the same **A**. Therefore, it is desirable to have an equation-solving algorithm that can handle any number of constant vectors with minimal computational effort.

Methods of Solution

There are two classes of methods for solving systems of linear algebraic equations:
direct and iterative methods. The common characteristic of ***direct methods*** is that they transform the original equations into *equivalent equations* (equations that have the same solution) that can be solved more easily. The transformation is carried out by applying the three operations listed below. These so-called *elementary operations* do not change the solution, but they may affect the determinant of the coefficient matrix as indicated in parenthesis.

- Exchanging two equations (changes sign of |**A**|).
- Multiplying an equation by a nonzero constant (multiplies |**A**| by the same constant).
- Multiplying an equation by a nonzero constant and then subtracting it from another equation (leaves |**A**| unchanged).

Iterative methods or *indirect methods* start with a guess of the solution **x**, and then repeatedly refine the solution until a certain convergence criterion is reached. Iterative methods are generally less efficient than their direct counterparts, but they do have significant computational advantages if the coefficient matrix is very large and sparsely populated (most coefficients are zero).

Overview of Direct Methods

Table 2.1 lists three popular direct methods, each of which uses elementary operations to produce its own final form of easy-to-solve equations.

Method	Initial form	Final form
Gauss elimination	**Ax** = **b**	**Ux** = **c**
LU decomposition	**Ax** = **b**	**LUx** = **b**
Gauss–Jordan elimination	**Ax** = **b**	**Ix** = **c**

Table 2.1

In the above table, **U** represents an upper triangular matrix, **L** is a lower triangular matrix, and **I** denotes the identity matrix. A square matrix is called *triangular*, if it contains only zero elements on one side of the principal diagonal. Thus a 3 × 3 upper triangular matrix has the form

$$\mathbf{U} = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix}$$

and a 3 × 3 lower triangular matrix appears as

$$\mathbf{L} = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{bmatrix}$$

Triangular matrices play an important role in linear algebra, since they simplify many computations. For example, consider the equations **Lx** = **c**, or

$$L_{11}x_1 = c_1$$
$$L_{21}x_1 + L_{22}x_2 = c_2$$
$$L_{31}x_1 + L_{32}x_2 + L_{33}x_3 = c_3$$
$$\ldots$$

If we solve the equations forwards, starting with the first equation, the computations are very easy, since each equation would contain only one unknown at a time. The solution would thus proceed as follows:

$$x_1 = c_1/L_{11}$$
$$x_2 = (c_2 - L_{21}x_1)/L_{22}$$
$$x_3 = (c_3 - L_{31}x_1 - L_{32}x_2)/L_{33}$$
$$..$$

This procedure is known as *forward substitution*. In a similar way, **Ux** = **c**, encountered in Gauss elimination, can easily be solved by *back substitution*, which starts with the last equation and proceeds backwards through the equations.

The equations **LUx** = **b**, which are associated with LU decomposition, can also be solved quickly if we replace them with two sets of equivalent equations: **Ly** = **b** and **Ux** = **y**. Now **Ly** = **b** can be solved for **y** by forward substitution, followed by the solution of **Ux** = **y** using back substitution.

The equations **Ix** = **c**, which are produced by Gauss–Jordan elimination, are equivalent to **x** = **c** (recall the identity **Ix** = **x**), so that **c** is already the solution.

EXAMPLE 2.1

Determine whether the matrix A is singular:

$$\mathbf{A} = \begin{bmatrix} 2.1 & -0.6 & 1.1 \\ 3.2 & 4.7 & -0.8 \\ 3.1 & -6.5 & 4.1 \end{bmatrix}$$

Solution Laplace's development (see Appendix A2) of the determinant about the first row of **A** yields

$$|\mathbf{A}| = 2.1\begin{vmatrix} 4.7 & -0.8 \\ -6.5 & 4.1 \end{vmatrix} + 0.6\begin{vmatrix} 3.2 & -0.8 \\ 3.1 & 4.1 \end{vmatrix} + 1.1\begin{vmatrix} 3.2 & 4.7 \\ 3.1 & -6.5 \end{vmatrix}$$
$$= 2.1(14.07) + 0.6(15.60) + 1.1(35.37) = 0$$

Since the determinant is zero, the matrix is singular. It can be verified that the singularity is due to the following row dependency: (row3) = (3 × row1)− (row2).

EXAMPLE 2.2

Solve the equations **Ax** = **b**, where

$$\mathbf{A} = \begin{bmatrix} 8 & -6 & 2 \\ -4 & 11 & -7 \\ 4 & -7 & 6 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 28 \\ -40 \\ 33 \end{bmatrix}$$

knowing that the LU decomposition of the coefficient matrix is (you should verify this)

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

Solution We first solve the equations **Ly** = **b** by forward substitution:

$$\begin{aligned} 2y_1 &= 28 & y_1 &= 28/2 = 14 \\ -y_1 + 2y_2 &= -40 & y_2 &= (-40 + y_1)/2 = (-40 + 14)/2 = -13 \\ y_1 - y_2 + y_3 &= 33 & y_3 &= 33 - y_1 + y_2 = 33 - 14 - 13 = 6 \end{aligned}$$

The solution **x** is then obtained from **Ux** = **y** by back substitution:

$$\begin{aligned} 2x_3 &= y_3 & x_3 &= y_3/2 = 6/2 = 3 \\ 4x_2 - 3x_3 &= y_2 & x_2 &= (y_2 + 3x_3)/4 = [-13 + 3(3)]/4 = -1 \\ 4x_1 - 3x_2 + x_3 &= y_1 & x_1 &= (y_1 + 3x_2 - x_3)/4 = [14 + 3(-1) - 3]/4 = 2 \end{aligned}$$

Hence the solution is $\mathbf{x} = [\,2 \;\; -1 \;\; 3\,]^T$

1. Gauss Elimination Method

Introduction

Gauss elimination is the most familiar method for solving simultaneous equations. It consists of two parts: the elimination phase and the solution phase. As indicated in Table 2.1, the function of the elimination phase is to transform the equations into the form **Ux** = **c**. The equations are then solved by back substitution. In order to illustrate the procedure, let us solve the equations

$$4x_1 - 2x_2 + x_3 = 11 \quad \text{(a)}$$
$$-2x_1 + 4x_2 - 2x_3 = -16 \quad \text{(b)}$$
$$x_1 - 2x_2 + 4x_3 = 17 \quad \text{(c)}$$

Elimination Phase The elimination phase utilizes only one of the elementary operations listed in Table 2.1 – multiplying one equation (say, equation j) by a constant λ and subtracting it from another equation (equation i). The symbolic representation of this operation is Eq. (i) ← Eq. (i) − λ × Eq. (j) (2.6)

The equation being subtracted, namely, Eq. (j), is called the *pivot equation*.

We start the elimination by taking Eq. (a) to be the pivot equation and choosing the multipliers λ so as to eliminate x_1 from Eqs. (b) and (c):

Eq. (b) ← Eq. (b) − (− 0.5) × Eq. (a)
Eq. (c) ← Eq. (c) − 0.25 × Eq. (a)

After this transformation, the equations become

$$4x_1 - 2x_2 + x_3 = 11 \quad \text{(a)}$$
$$3x_2 - 1.5x_3 = -10.5 \quad \text{(b)}$$
$$-1.5x_2 + 3.75x_3 = 14.25 \quad \text{(c)}$$

This completes the first pass. Now we pick (b) as the pivot equation and eliminate x_2 from (c):

Eq. (c) ← Eq. (c) − (− 0.5) × Eq.(b)

which yields the equations

$$4x_1 - 2x_2 + x_3 = 11 \quad \text{(a)}$$
$$3x_2 - 1.5x_3 = -10.5 \quad \text{(b)}$$
$$3x_3 = 9 \quad \text{(c)}$$

The elimination phase is now complete. The original equations have been replaced by equivalent equations that can be easily solved by back substitution.

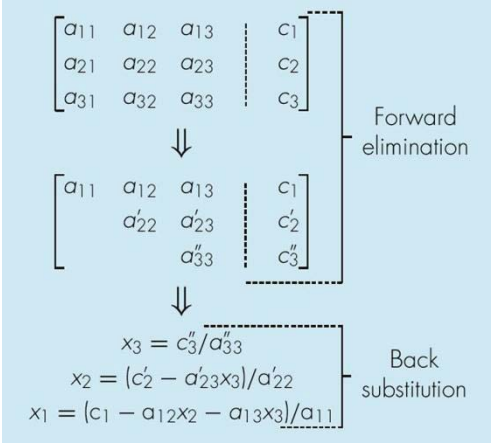
As pointed out before, the augmented coefficient matrix is a more convenient instrument for performing the computations. Thus the original equations would be written as

$$\left[\begin{array}{rrr|r} 4 & -2 & 1 & 11 \\ -2 & 4 & -2 & -16 \\ 1 & -2 & 4 & 17 \end{array}\right]$$

and the equivalent equations produced by the first and the second passes of Gauss elimination would appear as

$$\left[\begin{array}{rrr|r} 4 & -2 & 1 & 11.00 \\ 0 & 3 & -1.5 & -10.50 \\ 0 & -1.5 & 3.75 & 14.25 \end{array}\right]$$

$$\left[\begin{array}{rrr|r} 4 & -2 & 1 & 11.0 \\ 0 & 3 & -1.5 & -10.5 \\ 0 & 0 & 3 & 9.0 \end{array}\right]$$



It is important to note that the elementary row operation in Eq. (2.6) leaves the determinant of the coefficient matrix unchanged. This is rather fortunate, since the determinant of a triangular matrix is very easy to compute – it is the product of the diagonal elements (you can verify this quite easily). In other words,

$$|\mathbf{A}| = |\mathbf{U}| = U_{11} \times U_{22} \times \cdots \times U_{nn} \qquad (2.7)$$

Back Substitution Phase The unknowns can now be computed by back substitution in the manner described in the previous article. Solving Eqs. (c), (b), and (a) in that order, we get

$$x_3 = 9/3 = 3$$
$$x_2 = (-10.5 + 1.5x_3)/3 = [-10.5 + 1.5(3)]/3 = -2$$
$$x_1 = (11 + 2x_2 - x_3)/4 = [11 + 2(-2) - 3]/4 = 1$$

Algorithm for Gauss Elimination Method

Elimination Phase Let us look at the equations at some instant during the elimination phase. Assume that the first k rows of **A** have already been transformed to upper-triangular form. Therefore, the current pivot equation is the kth equation, and all the equations below it are still to be transformed. This situation is depicted by the augmented coefficient matrix shown below. Note that the components of **A** are not the coefficients of the original equations (except for the first row), since they have been altered by the elimination procedure. The same applies to the components of the constant vector **b**.

$$\left[\begin{array}{ccccccccc|c} A_{11} & A_{12} & A_{13} & \cdots & A_{1k} & \cdots & A_{1j} & \cdots & A_{1n} & b_1 \\ 0 & A_{22} & A_{23} & \cdots & A_{2k} & \cdots & A_{2j} & \cdots & A_{2n} & b_2 \\ 0 & 0 & A_{33} & \cdots & A_{3k} & \cdots & A_{3j} & \cdots & A_{3n} & b_3 \\ \vdots & \vdots & \vdots & & \vdots & & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_{kk} & \cdots & A_{kj} & \cdots & A_{kn} & b_k \\ \hline \vdots & \vdots & \vdots & & \vdots & & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_{ik} & \cdots & A_{ij} & \cdots & A_{in} & b_i \\ \vdots & \vdots & \vdots & & \vdots & & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_{nk} & \cdots & A_{nj} & \cdots & A_{nn} & b_n \end{array}\right]$$

← pivot row

← row being transformed

Let the ith row be a typical row below the pivot equation that is to be transformed, meaning that the element A_{ik} is to be eliminated. We can achieve this by multiplying the pivot row by $\lambda = A_{ik}/A_{kk}$ and subtracting it from the ith row. The corresponding changes in the ith row are

$$A_{ij} \leftarrow A_{ij} - \lambda A_{kj}, \qquad j = k, k+1, \ldots, n \qquad (2.8a)$$
$$b_i \leftarrow b_i - \lambda b_k \qquad (2.8b)$$

In order to transform the entire coefficient matrix to upper-triangular form, k and I in Eqs. (2.8) must have the ranges $k = 1, 2, \ldots, n-1$ (chooses the pivot row), $i = k+1, k+2, \ldots, n$ (chooses the row to be transformed). The algorithm for the elimination phase now almost writes it self:

```
for k = 1: n-1
  for i= k+1: n
    if A(i,k) ~= 0
      lambda = A(i,k)/A(k,k);
      A(i,k+1: n) = A(i,k+1: n) - lambda*A(k,k+1: n);
      b(i)= b(i) - lambda*b(k);
    end
  end
end
```

In order to avoid unnecessary operations, the above algorithm departs slightly from Eqs. (2.8) in the following ways:

- If A_{ik} happens to be zero, the transformation of row i is skipped.
- The index j in Eq. (2.8a) starts with $k + 1$ rather than k. Therefore, A_{ik} is not replaced by zero, but retains its original value. As the solution phase never accesses the lower triangular portion of the coefficient matrix anyway, its contents are irrelevant.

Back Substitution Phase After Gauss elimination the augmented coefficient matrix has the form

$$[\mathbf{A}|\mathbf{b}] = \left[\begin{array}{ccccc|c} A_{11} & A_{12} & A_{13} & \cdots & A_{1n} & b_1 \\ 0 & A_{22} & A_{23} & \cdots & A_{2n} & b_2 \\ 0 & 0 & A_{33} & \cdots & A_{3n} & b_3 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_{nn} & b_n \end{array}\right]$$

The last equation, $A_{nn}x_n = b_n$, is solved first, yielding $x_n = b_n/A_n$ (2.9)

Consider now the stage of back substitution where $x_n, x_{n-1}, \ldots, x_{k+1}$ have already been computed (in that order), and we are about to determine x_k from the kth equation

$$A_{kk}x_k + A_{k,k+1}x_{k+1} + \cdots + A_{kn}x_n = b_k$$

The solution is

$$x_k = \left(b_k - \sum_{j=k+1}^{n} A_{kj}x_j\right)\frac{1}{A_{kk}}, \quad k = n-1, n-2, \ldots, 1 \qquad (2.10)$$

Equations (2.9) and (2.10) result in the following algorithm for back substitution:

```
for k = n: -1: 1
    b(k) = (b(k) - A(k,k+1: n)*b(k+1: n))/A(k,k);
end
```

Operation Count

The execution time of an algorithm depends largely on the number of long operations (multiplications and divisions) performed. It can be shown that Gauss elimination contains approximately $n^3/3$ such operations (n is the number of equations) in the elimination phase, and $n^2/2$ operations in back substitution. These numbers show that most of the computation time goes into the elimination phase. Moreover, the time increases very rapidly with the number of equations.

gauss

The function gauss combines the elimination and the back substitution phases. During back substitution, b is overwritten by the solution vector x, so that b contains the solution upon exit.

```
function [x,det] = gauss(A,b)
% Solves A*x = b by Gauss elimination and computes det(A).
% USAGE: [x,det] = gauss(A,b)
if size(b,2) > 1; b = b'; end        % b must be column vector
n = length(b);
for k = 1: n-1                       % Elimination phase
    for i= k+1: n
        if A(i,k) ~= 0
            lambda = A(i,k)/A(k,k);
```

```
            A(i,k+1: n) = A(i,k+1: n) - lambda*A(k,k+1: n);
            b(i)= b(i) - lambda*b(k);
        end
    end
end
if nargout == 2; det = prod(diag(A)); end
for k = n:-1:1                          % Back substitution phase
    b(k) = (b(k) - A(k,k+1: n)*b(k+1: n))/A(k,k);
end
x = b;
```

Multiple Sets of Equations

As mentioned before, it is frequently necessary to solve the equations $\mathbf{Ax} = \mathbf{b}$ for several constant vectors. Let there be m such constant vectors, denoted by $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$ and let the corresponding solution vectors be $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$. We denote multiple sets of equations by $\mathbf{AX} = \mathbf{B}$, where

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_m \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_m \end{bmatrix}$$

are $n \times m$ matrices whose columns consist of solution vectors and constant vectors, respectively.

An economical way to handle such equations during the elimination phase is to include all m constant vectors in the augmented coefficient matrix, so that they are transformed simultaneously with the coefficient matrix. The solutions are then obtained by back substitution in the usual manner, one vector at a time. It would be quite easy to make the corresponding changes in *gaussElimin.* However, the LU decomposition method, described in the next section, is more versatile in handling multiple constant vectors.

EXAMPLE 2.3

Use Gauss elimination to solve the equations $\mathbf{AX} = \mathbf{B}$, where

$$\mathbf{A} = \begin{bmatrix} 6 & -4 & 1 \\ -4 & 6 & -4 \\ 1 & -4 & 6 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} -14 & 22 \\ 36 & -18 \\ 6 & 7 \end{bmatrix}$$

Solution The augmented coefficient matrix is

$$\left[\begin{array}{ccc|cc} 6 & -4 & 1 & -14 & 22 \\ -4 & 6 & -4 & 36 & -18 \\ 1 & -4 & 6 & 6 & 7 \end{array}\right]$$

The elimination phase consists of the following two passes:

$$\text{row } 2 \leftarrow \text{row } 2 + (2/3) \times \text{row } 1$$
$$\text{row } 3 \leftarrow \text{row } 3 - (1/6) \times \text{row } 1$$

$$\left[\begin{array}{ccc|cc} 6 & -4 & 1 & -14 & 22 \\ 0 & 10/3 & -10/3 & 80/3 & -10/3 \\ 0 & -10/3 & 35/6 & 25/3 & 10/3 \end{array}\right]$$

and

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

$$\left[\begin{array}{ccc|cc} 6 & -4 & 1 & -14 & 22 \\ 0 & 10/3 & -10/3 & 80/3 & -10/3 \\ 0 & 0 & 5/2 & 35 & 0 \end{array}\right]$$

In the solution phase, we first compute $\mathbf{x}_1$ by back substitution:

$$X_{31} = \frac{35}{5/2} = 14$$

$$X_{21} = \frac{80/3 + (10/3)X_{31}}{10/3} = \frac{80/3 + (10/3)14}{10/3} = 22$$

$$X_{11} = \frac{-14 + 4X_{21} - X_{31}}{6} = \frac{-14 + 4(22) - 14}{6} = 10$$

Thus the first solution vector is $\mathbf{x}_1 = \begin{bmatrix} X_{11} & X_{21} & X_{31} \end{bmatrix}^T = \begin{bmatrix} 10 & 22 & 14 \end{bmatrix}^T$

The second solution vector is computed next, also using back substitution:

$$X_{32} = 0$$

$$X_{22} = \frac{-10/3 + (10/3)X_{32}}{10/3} = \frac{-10/3 + 0}{10/3} = -1$$

$$X_{12} = \frac{22 + 4X_{22} - X_{32}}{6} = \frac{22 + 4(-1) - 0}{6} = 3$$

Therefore, $\mathbf{x}_2 = \begin{bmatrix} X_{12} & X_{22} & X_{32} \end{bmatrix}^T = \begin{bmatrix} 3 & -1 & 0 \end{bmatrix}^T$

2. LU Decomposition Methods

Introduction

It is possible to show that any square matrix **A** can be expressed as a product of a lower triangular matrix **L** and an upper triangular matrix **U**: **A** = **LU** (2.11)

The process of computing **L** and **U** for a given **A** is known as *LU decomposition* or *LU factorization*. LU decomposition is not unique (the combinations of **L** and **U** for a prescribed **A** are endless), unless certain constraints are placed on **L** or **U**. These constraints distinguish one type of decomposition from another. Three commonly used decompositions are listed in Table 2.2.

Name	Constraints
Doolittle's decomposition	$L_{ii} = 1, \quad i = 1, 2, \ldots, n$
Crout's decomposition	$U_{ii} = 1, \quad i = 1, 2, \ldots, n$
Choleski's decomposition	$\mathbf{L} = \mathbf{U}^T$

Table 2.2

After decomposing **A**, it is easy to solve the equations **Ax** = **b**, as pointed out in Section 2.1. We first rewrite the equations as **LUx** = **b**. Upon using the notation **Ux** = **y**, the equations become **Ly** = **b** which can be solved for **y** by forward substitution. Then **Ux** = **y** will yield **x** by the back substitution process.

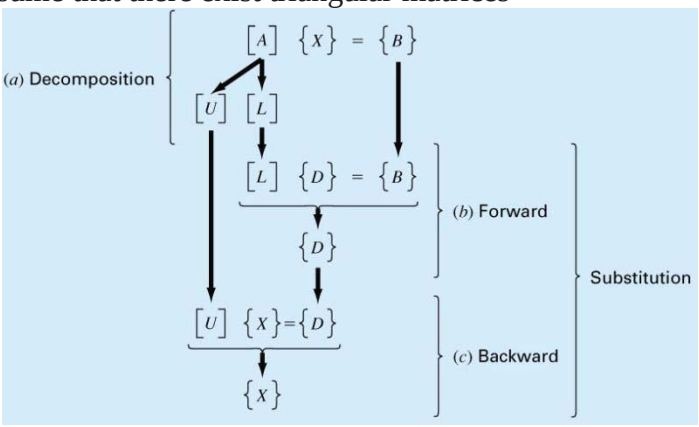
The advantage of LU decomposition over the Gauss elimination method is that once **A** is decomposed, we can solve **Ax** = **b** for as many constant vectors **b** as we please. The cost of each additional solution is relatively small, since the forward and back substitution operations are much less time-consuming than the decomposition process.

Doolittle's Decomposition Method

Decomposition Phase

Doolittle's decomposition is closely related to Gauss elimination. In order to illustrate the relationship, consider a 3 × 3 matrix **A** and assume that there exist triangular matrices

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ L_{21} & 1 & 0 \\ L_{31} & L_{32} & 1 \end{bmatrix} \qquad \mathbf{U} = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix}$$



such that **A** = **LU**. After completing the multiplication on the right-hand side, we get

$$\mathbf{A} = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ U_{11}L_{21} & U_{12}L_{21} + U_{22} & U_{13}L_{21} + U_{23} \\ U_{11}L_{31} & U_{12}L_{31} + U_{22}L_{32} & U_{13}L_{31} + U_{23}L_{32} + U_{33} \end{bmatrix} \qquad (2.12)$$

Let us now apply Gauss elimination to Eq. (2.12). The first pass of the elimination procedure consists of choosing the first row as the pivot row and applying the elementary operations

row 2 ← row 2 − L_{21} × row 1 (eliminates A_{21})
row 3 ← row 3 − L_{31} × row 1 (eliminates A_{31})

The result is

$$\mathbf{A}' = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & U_{22}L_{32} & U_{23}L_{32} + U_{33} \end{bmatrix}$$

In the next pass, we take the second row as the pivot row and utilize the operation

row 3 ← row 3 − L_{32} × row 2 (eliminates A_{32})

ending up with

$$\mathbf{A}'' = \mathbf{U} = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix}$$

The foregoing illustration reveals two important features of Doolittle's decomposition:

- The matrix **U** is identical to the upper triangular matrix that results from Gauss elimination.
- The off-diagonal elements of **L** are the pivot equation multipliers used during Gauss elimination; that is, L_{ij} is the multiplier that eliminated A_{ij} .

It is usual practice to store the multipliers in the lower triangular portion of the coefficient matrix, replacing the coefficients as they are eliminated (L_{ij} replacing A_{ij}).
The diagonal elements of **L** do not have to be stored, since it is understood that each of them is unity. The final form of the coefficient matrix would thus be the following mixture of **L** and **U**:

$$[\mathbf{L}\backslash\mathbf{U}] = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ L_{21} & U_{22} & U_{23} \\ L_{31} & L_{32} & U_{33} \end{bmatrix} \qquad (2.13)$$

The algorithm for Doolittle's decomposition is thus identical to the Gauss elimination procedure in gauss, except that each multiplier λ is now stored in the lower triangular portion of **A**.
The number of long operations in L/U decomposition is the same as in Gauss elimination, namely, $n^3/3$.

LUdec

In this version of LU decomposition the original **A** is destroyed and replaced by its decomposed form[**L****U**].

```
function A = LUdec(A)
% LU decomposition of matrix A; returns A = [L\U].
% USAGE: A = LUdec(A)
n = size(A,1);
for k = 1: n-1
    for i = k+1: n
        if A(i,k) ~= 0.0
          lambda = A(i,k)/A(k,k);
          A(i,k+1: n) = A(i,k+1: n) - lambda*A(k,k+1: n);
          A(i,k) = lambda;
        end
    end
end
```

Solution Phase

Consider now the procedure for the solution of **Ly** = **b** by forward substitution. The scalar form of the equations is (recall that $L_{ii} = 1$)

$$\begin{aligned} y_1 &= b_1 \\ L_{21}y_1 + y_2 &= b_2 \\ &\cdots \\ L_{k1}y_1 + L_{k2}y_2 + \cdots + L_{k,k-1}y_{k-1} + y_k &= b_k \\ &\cdots \end{aligned}$$

Solving the kth equation for y_k yields

$$y_k = b_k - \sum_{j=1}^{k-1} L_{kj}y_j, \quad k = 2, 3, \ldots, n \qquad (2.14)$$

which gives us the forward substitution algorithm:

```
for k = 2: n
    b(k)= b(k) - A(k,1:k-1)*y(1:k-1);
end
```

The back substitution phase for solving **Ux** = **y** is identical to what was used in the Gauss elimination method.

LUsol

This function carries out the solution phase (forward and back substitutions). It is assumed that the original coefficient matrix has been decomposed, so that the input is **A** = [**L****U**]. The contents of **b** are replaced by **y** during forward substitution. Similarly, back substitution overwrites **y** with the solution **x**.

```
function x = LUsol(A,b)
% Solves L*U*b = x, where A contains both L and U;
% that is, A has the form [L\U].
% USAGE: x = LUsol(A,b)

if size(b,2) > 1; b = b'; end
n = length(b);
for k = 2: n
    b(k) = b(k) - A(k,1:k-1)*b(1:k-1);
end
for k = n: -1: 1
    b(k) = (b(k) - A(k,k+1:n)*b(k+1:n))/A(k,k);
end
x = b;
```

Choleski's Decomposition Method

Choleski's decomposition **A** = **LL**^{T} has two limitations:

- Since the matrix product **LL**^{T} is always symmetric, Choleski's decomposition can be applied only to *symmetric* matrices.
- The decomposition process involves taking square roots of certain combinations of the elements of **A**. It can be shown that square roots of negative numbers can be avoided only if **A** is *positive definite*.

Choleski's decomposition contains approximately $n^3/6$ long operations plus n square root computations. This is about half the number of operations required in LU decomposition. The relative efficiency of Choleski's decomposition is due to its exploitation of symmetry.

Let us start by looking at Choleski's decomposition

$$\mathbf{A} = \mathbf{LL}^T \qquad (2.15)$$

of a 3 ×3 matrix:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \begin{bmatrix} L_{11} & L_{21} & L_{31} \\ 0 & L_{22} & L_{32} \\ 0 & 0 & L_{33} \end{bmatrix}$$

After completing the matrix multiplication on the right hand side, we get

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} L_{11}^2 & L_{11}L_{21} & L_{11}L_{31} \\ L_{11}L_{21} & L_{21}^2 + L_{22}^2 & L_{21}L_{31} + L_{22}L_{32} \\ L_{11}L_{31} & L_{21}L_{31} + L_{22}L_{32} & L_{31}^2 + L_{32}^2 + L_{33}^2 \end{bmatrix} \qquad (2.16)$$

Note that the right-hand-side matrix is symmetric, as pointed out before. Equating the matrices **A** and $\mathbf{LL}^T$ element-by-element, we obtain six equations (due to symmetry only lower or upper triangular elements have to be considered) in the six unknown components of **L**. By solving these equations in a certain order, it is possible to have only one unknown in each equation.

Consider the lower triangular portion of each matrix in Eq. (2.16) (the upper triangular portion would do as well). By equating the elements in the first column, starting with the first row and proceeding downward, we can compute L_{11}, L_{21}, and L_{31} in that order:

$$A_{11} = L_{11}^2 \qquad L_{11} = \sqrt{A_{11}}$$

$$A_{21} = L_{11}L_{21} \qquad L_{21} = A_{21}/L_{11}$$

$$A_{31} = L_{11}L_{31} \qquad L_{31} = A_{31}/L_{11}$$

The second column, starting with second row, yields L_{22} and L_{32}:

$$A_{22} = L_{21}^2 + L_{22}^2 \qquad L_{22} = \sqrt{A_{22} - L_{21}^2}$$

$$A_{32} = L_{21}L_{31} + L_{22}L_{32} \qquad L_{32} = (A_{32} - L_{21}L_{31})/L_{22}$$

Finally, the third column, third row, gives us L_{33}:

$$A_{33} = L_{31}^2 + L_{32}^2 + L_{33}^2 \qquad L_{33} = \sqrt{A_{33} - L_{31}^2 - L_{32}^2}$$

We can now extrapolate the results for an $n \times n$ matrix. We observe that a typical element in the lower triangular portion of $\mathbf{LL}^T$ is of the form

$$(\mathbf{LL}^T)_{ij} = L_{i1}L_{j1} + L_{i2}L_{j2} + \cdots + L_{ij}L_{jj} = \sum_{k=1}^{j} L_{ik}L_{jk}, \quad i \geq j$$

Equating this term to the corresponding element of **A** yields

$$A_{ij} = \sum_{k=1}^{j} L_{ik}L_{jk}, \quad i = j, j+1, \ldots, n, \quad j = 1, 2, \ldots, n \qquad (2.17)$$

The range of indices shown limits the elements to the lower triangular part. For the first column (j = 1), we obtain from Eq. (2.17)

$$L_{11} = \sqrt{A_{11}} \qquad L_{i1} = A_{i1}/L_{11}, \quad i = 2, 3, \ldots, n \qquad (2.18)$$

Proceeding to other columns, we observe that the unknown in Eq. (2.17) is L_{ij} (the other elements of **L** appearing in the equation have already been computed). Taking the term containing L_{ij} outside the summation in Eq. (2.17), we obtain

$$A_{ij} = \sum_{k=1}^{j-1} L_{ik}L_{jk} + L_{ij}L_{jj}$$

If $i = j$ (a diagonal term), the solution is

$$L_{jj} = \sqrt{A_{jj} - \sum_{k=1}^{j-1} L_{jk}^2}, \quad j = 2, 3, \ldots, n \qquad (2.19)$$

For a non diagonal term, we get

$$L_{ij} = \left(A_{ij} - \sum_{k=1}^{j-1} L_{ik} L_{jk} \right) / L_{jj}, \quad j = 2, 3, \ldots, n-1, \quad i = j+1, j+2, \ldots, n \qquad (2.20)$$

choleski

Note that in Eqs. (2.19) and (2.20), A_{ij} appears only in the formula for L_{ij} . Therefore, once L_{ij} has been computed, A_{ij} is no longer needed. This makes it possible to write the elements of **L** over the lower triangular portion of **A** as they are computed. The elements above the principal diagonal of **A** will remain untouched. At the conclusion of decomposition, **L** is extracted with the MATLAB command tril(A). If a negative $L^2_{j\,j}$ is encountered during decomposition, an error message is printed and the program is terminated.

```
function L = choleski(A)
% Computes L in Choleski's decomposition A = LL'.
% USAGE: L = choleski(A)
n = size(A,1);
for j = 1: n
    temp = A(j,j) - dot(A(j,1:j-1),A(j,1:j-1));
    if temp < 0.0
        error('Matrix is not positive definite')
    end
    A(j,j) = sqrt(temp);
    for i = j+1: n
        A(i,j)=(A(i,j) - dot(A(i,1:j-1),A(j,1:j-1)))/A(j,j);
    end
end
L = tril(A)
```

choleskiSol

After the coefficient matrix **A** has been decomposed, the solution of **Ax** = **b** can be obtained by the usual forward and back substitution operations. The following function (given without derivation) carries out the solution phase.

```
function x = choleskiSol(L,b)
% Solves [L][L']{x} = {b}
% USAGE: x = choleskiSol(L,b)
n = length(b);
if size(b,2) > 1; b = b'; end          % {b} must be column vector
for k = 1: n                           % Solution of [L]{y} = {b}
    b(k) = (b(k) - dot(L(k,1:k-1),b(1:k-1)'))/L(k,k);
end
for k = n: -1: 1                       % Solution of {L}'{x} = {y}
    b(k) = (b(k) - dot(L(k+1:n,k),b(k+1:n)))/L(k,k);
end
x = b;
```

Other Methods

Crout's Decomposition

Recall that the various decompositions **A** = **LU** are characterized by the constraints placed on the elements of **L** or **U**. In Doolittle's decomposition the diagonal elements of **L** were set to 1. An equally viable method is Crout's decomposition, where the 1s lie on diagonal of **U**. There is little difference in the performance of the two methods.

Gauss–Jordan Elimination

The Gauss–Jordan method is essentially Gauss elimination taken to its limit. In the Gauss elimination method only the equations that lie below the pivot equation are transformed. In the Gauss–Jordan method the elimination is also carried out on equations above the pivot equation, resulting in a diagonal coefficient matrix.

The main disadvantage ofGauss–Jordan elimination is that it involves about $n^3/2$ long operations, which is 1.5 times the number required in Gauss elimination.

EXAMPLE 2.5

Use Doolittle's decomposition method to solve the equations **Ax** = **b,** where

$$\mathbf{A}=\begin{bmatrix}1 & 4 & 1\\ 1 & 6 & -1\\ 2 & -1 & 2\end{bmatrix} \qquad \mathbf{b}=\begin{bmatrix}7\\ 13\\ 5\end{bmatrix}$$

Solution We first decompose **A** by Gauss elimination. The first pass consists of the elementary operations

$$\text{row } 2 \leftarrow \text{row } 2 - 1 \times \text{row } 1 \text{ (eliminates } A_{21})$$
$$\text{row } 3 \leftarrow \text{row } 3 - 2 \times \text{row } 1 \text{ (eliminates } A_{31})$$

Storing the multipliers $L_{21} = 1$ and $L_{31} = 2$ in place of the eliminated terms, we Obtain

$$\mathbf{A}'=\begin{bmatrix}1 & 4 & 1\\ [1] & 2 & -2\\ [2] & -9 & 0\end{bmatrix}$$

where the multipliers are shown in brackets.

The second pass of Gauss elimination uses the operation

$$\text{row } 3 \leftarrow \text{row } 3 - (-4.5) \times \text{row } 2 \text{ (eliminates } A_{32})$$

Since we do not wish to touch the multipliers, the bracketed terms are excluded from the operation. Storing the multiplier $L_{32} = -4.5$ in place of A_{32}, we get

$$\mathbf{A}''=[\mathbf{L}\backslash\mathbf{U}]=\begin{bmatrix}1 & 4 & 1\\ [1] & 2 & -2\\ [2] & [-4.5] & -9\end{bmatrix}$$

The decomposition is now complete, with

$$\mathbf{L}=\begin{bmatrix}1 & 0 & 0\\ 1 & 1 & 0\\ 2 & -4.5 & 1\end{bmatrix} \qquad \mathbf{U}=\begin{bmatrix}1 & 4 & 1\\ 0 & 2 & -2\\ 0 & 0 & -9\end{bmatrix}$$

Solution of **Ly** = **b** by forward substitution comes next. The augmented coefficient form of the equations is

$$[\mathbf{L}|\mathbf{b}]=\left[\begin{array}{ccc|c}1 & 0 & 0 & 7\\ 1 & 1 & 0 & 13\\ 2 & -4.5 & 1 & 5\end{array}\right]$$

The solution is

$$y_1 = 7$$
$$y_2 = 13 - y_1 = 13 - 7 = 6$$
$$y_3 = 5 - 2y_1 + 4.5y_2 = 5 - 2(7) + 4.5(6) = 18$$

Finally, the equations **Ux** = **y**, or

$$[\mathbf{U}|\mathbf{y}]=\left[\begin{array}{ccc|c}1 & 4 & 1 & 7\\ 0 & 2 & -2 & 6\\ 0 & 0 & -9 & 18\end{array}\right]$$