SOLVING LINEAR SYSTEMS

We want to solve the linear system

$$\begin{aligned} a_{1,1}x_1 + \cdots + a_{1,n}x_n &= b_1 \\ &\vdots \\ a_{n,1}x_1 + \cdots + a_{n,n}x_n &= b_n \end{aligned}$$

This will be done by the method used in beginning algebra, by successively eliminating unknowns from equations, until eventually we have only one equation in one unknown. This process is known as *Gaussian elimination*. To put it onto a computer, however, we must be more precise than is generally the case in high school algebra.

We begin with the linear system

$$\begin{aligned} 3x_1 - 2x_2 - x_3 &= 0 \quad (E1) \\ 6x_1 - 2x_2 + 2x_3 &= 6 \quad (E2) \\ -9x_1 + 7x_2 + x_3 &= -1 \quad (E3) \end{aligned}$$

$$\begin{aligned} 3x_1 - 2x_2 - x_3 &= 0 && (E1) \\ 6x_1 - 2x_2 + 2x_3 &= 6 && (E2) \\ -9x_1 + 7x_2 + x_3 &= -1 && (E3) \end{aligned}$$

[1] Eliminate x_1 from equations $(E2)$ and $(E3)$. Subtract 2 times $(E1)$ from $(E2)$; and subtract -3 times $(E1)$ from $(E3)$. This yields

$$\begin{aligned} 3x_1 - 2x_2 - x_3 &= 0 && (E1) \\ 2x_2 + 4x_3 &= 6 && (E2) \\ x_2 - 2x_3 &= -1 && (E3) \end{aligned}$$

[2] Eliminate x_2 from equation $(E3)$. Subtract $\frac{1}{2}$ times $(E2)$ from $(E3)$. This yields

$$\begin{aligned} 3x_1 - 2x_2 - x_3 &= 0 && (E1) \\ 2x_2 + 4x_3 &= 6 && (E2) \\ -4x_3 &= -4 && (E3) \end{aligned}$$

Using back substitution, solve for x_3, x_2, and x_1, obtaining

$$x_3 = x_2 = x_1 = 1$$

In the computer, we work on the arrays rather than on the equations. To illustrate this, we repeat the preceding example using array notation.

The original system is $Ax = b$, with

$$A = \begin{bmatrix} 3 & -2 & -1 \\ 6 & -2 & 2 \\ -9 & 7 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 6 \\ -1 \end{bmatrix}$$

We often write these in combined form as an *augmented* matrix:

$$[A \mid b] = \left[\begin{array}{ccc|c} 3 & -2 & -1 & 0 \\ 6 & -2 & 2 & 6 \\ -9 & 7 & 1 & -1 \end{array}\right]$$

In step 1, we eliminate x_1 from equations 2 and 3. We multiply row 1 by 2 and subtract it from row 2; and we multiply row 1 by -3 and subtract it from row 3. This yields

$$\left[\begin{array}{ccc|c} 3 & -2 & -1 & 0 \\ 0 & 2 & 4 & 6 \\ 0 & 1 & -2 & -1 \end{array}\right]$$

$$\left[\begin{array}{rrr|r} 3 & -2 & -1 & 0 \\ 0 & 2 & 4 & 6 \\ 0 & 1 & -2 & -1 \end{array}\right]$$

In step 2, we eliminate x_2 from equation 3. We multiply row 2 by $\frac{1}{2}$ and subtract from row 3. This yields

$$\left[\begin{array}{rrr|r} 3 & -2 & -1 & 0 \\ 0 & 2 & 4 & 6 \\ 0 & 0 & -4 & -4 \end{array}\right]$$

Then we proceed with back substitution as previously.

For the general case, we reduce

$$[A \mid b] = \left[\begin{array}{ccc|c} a_{1,1}^{(1)} & \cdots & a_{1,n}^{(1)} & b_1^{(1)} \\ \vdots & \ddots & \vdots & \vdots \\ a_{n,1}^{(1)} & \cdots & a_{n,n}^{(1)} & b_n^{(1)} \end{array}\right]$$

in $n-1$ steps to the form

$$\left[\begin{array}{cccc|c} a_{1,1}^{(1)} & \cdots & & a_{1,n}^{(1)} & b_1^{(1)} \\ 0 & \ddots & & \vdots & \vdots \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & 0 & a_{n,n}^{(n)} & b_n^{(n)} \end{array}\right]$$

More simply, and introducing new notation, this is equivalent to the matrix-vector equation $Ux = g$:

$$\left[\begin{array}{cccc} u_{1,1} & \cdots & & u_{1,n} \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & 0 & u_{n,n} \end{array}\right] \left[\begin{array}{c} x_1 \\ \vdots \\ \vdots \\ x_n \end{array}\right] = \left[\begin{array}{c} g_1 \\ \vdots \\ \vdots \\ g_n \end{array}\right]$$

This is the linear system

$$\begin{aligned} u_{1,1}x_1 + u_{1,2}x_2 + \cdots + u_{1,n-1}x_{n-1} + u_{1,n}x_n &= g_1 \\ &\vdots \\ u_{n-1,n-1}x_{n-1} + u_{n-1,n}x_n &= g_{n-1} \\ u_{n,n}x_n &= g_n \end{aligned}$$

We solve for x_n, then x_{n-1}, and backwards to x_1. This process is called *back substitution*.

$$x_n = \frac{g_n}{u_{n,n}}$$

$$u_k = \frac{g_k - \left\{u_{k,k+1}x_{k+1} + \cdots + u_{k,n}x_n\right\}}{u_{k,k}}$$

for $k = n-1, ..., 1$. What we have done here is simply a more carefully defined and methodical version of what you have done in high school algebra.

How do we carry out the conversion of

$$\left[\begin{array}{ccc|c} a_{1,1}^{(1)} & \cdots & a_{1,n}^{(1)} & b_1^{(1)} \\ \vdots & \ddots & \vdots & \vdots \\ a_{n,1}^{(1)} & \cdots & a_{n,n}^{(1)} & b_n^{(1)} \end{array}\right]$$

to

$$\left[\begin{array}{ccccc|c} a_{1,1}^{(1)} & \cdots & & & a_{1,n}^{(1)} & b_1^{(1)} \\ 0 & \ddots & & & \vdots & \vdots \\ \vdots & & \ddots & & & \vdots \\ 0 & \cdots & 0 & a_{n,n}^{(n)} & & b_n^{(n)} \end{array}\right]$$

To help us keep track of the steps of this process, we will denote the initial system by

$$[A^{(1)} \mid b^{(1)}] = \left[\begin{array}{ccc|c} \underline{a_{1,1}^{(1)}} & \cdots & a_{1,n}^{(1)} & b_1^{(1)} \\ \vdots & \ddots & \vdots & \vdots \\ a_{n,1}^{(1)} & \cdots & a_{n,n}^{(1)} & b_n^{(1)} \end{array}\right]$$

Initially we will make the assumption that every *pivot element* will be nonzero; and later we remove this assumption.

Step 1. We will eliminate x_1 from equations 2 thru n. Begin by defining the *multipliers*

$$m_{i,1} = \frac{a_{i,1}^{(1)}}{a_{1,1}^{(1)}}, \quad i = 2, ..., n$$

Here we are assuming the *pivot element* $a_{1,1}^{(1)} \neq 0$. Then in succession, multiply $m_{i,1}$ times row 1 (called the *pivot row*) and subtract the result from row i. This yields new matrix elements

$$a_{i,j}^{(2)} = a_{i,j}^{(1)} - m_{i,1} a_{1,j}^{(1)}, \quad j = 2, ..., n$$

$$b_i^{(2)} = b_i^{(1)} - m_{i,1} b_1^{(1)}$$

for $i = 2, ..., n$.

Note that the index j does not include $j = 1$. The reason is that with the definition of the multiplier $m_{i,1}$, it is automatic that

$$a_{i,1}^{(2)} = a_{i,1}^{(1)} - m_{i,1} a_{1,1}^{(1)} = 0, \quad i = 2, ..., n$$

The augmented matrix now is

$$[A^{(2)} \mid b^{(2)}] = \left[\begin{array}{cccc|c} a_{1,1}^{(1)} & a_{1,2}^{(1)} & \cdots & a_{1,n}^{(1)} & b_1^{(1)} \\ 0 & \underline{a_{2,2}^{(2)}} & & a_{2,n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n,2}^{(2)} & \cdots & a_{n,n}^{(2)} & b_n^{(2)} \end{array}\right]$$

Step k: Assume that for $i = 1, ..., k-1$ the unknown x_i has been eliminated from equations $i+1$ thru n. We have the augmented matrix

$$[A^{(k)} \mid b^{(k)}] = \left[\begin{array}{ccccccc|c} a_{1,1}^{(1)} & a_{1,2}^{(1)} & & \cdots & & & a_{1,n}^{(1)} & b_1^{(1)} \\ 0 & a_{2,2}^{(2)} & & \cdots & & & a_{2,n}^{(2)} & b_2^{(2)} \\ & \ddots & \ddots & & & & \vdots & \vdots \\ \vdots & & 0 & \underline{a_{k,k}^{(k)}} & \cdots & & a_{k,n}^{(k)} & b_k^{(k)} \\ & & \vdots & \vdots & \ddots & & \vdots & \vdots \\ 0 & \cdots & 0 & a_{n,k}^{(k)} & \cdots & & a_{n,n}^{(k)} & b_n^{(k)} \end{array}\right]$$

We want to eliminate unknown x_k from equations $k+1$ thru n. Begin by defining the multipliers

$$m_{i,k} = \frac{a_{i,k}^{(k)}}{a_{k,k}^{(k)}}, \quad i = k+1, ..., n$$

The pivot element is $a_{k,k}^{(k)}$, and we assume it is nonzero. Using these multipliers, we eliminate x_k from equations $k+1$ thru n. Multiply $m_{i,k}$ times row k (the *pivot row*) and subtract from row i, for $i = k+1$ thru n.

$$a_{i,j}^{(k+1)} = a_{i,j}^{(k)} - m_{i,k} a_{k,j}^{(k)}, \quad j = k+1, ..., n$$

$$b_i^{(k+1)} = b_i^{(k)} - m_{i,k} b_k^{(k)}$$

for $i = k+1, ..., n$. This yields the augmented matrix

$[A^{(k+1)} \mid b^{(k+1)}]$:

$$\left[\begin{array}{ccccccc|c}
a_{1,1}^{(1)} & & & & \cdots & & a_{1,n}^{(1)} & b_1^{(1)} \\
0 & \ddots & & & & & \vdots & \vdots \\
 & & a_{k,k}^{(k)} & a_{k,k+1}^{(k)} & \cdots & a_{k,n}^{(k)} & b_k^{(k)} \\
\vdots & & 0 & \underline{a_{k+1,k+1}^{(k+1)}} & & a_{k+1,n}^{(k+1)} & b_{k+1}^{(k+1)} \\
 & & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & \cdots & 0 & a_{n,k+1}^{(k+1)} & \cdots & a_{n,n}^{(k+1)} & b_n^{(k+1)}
\end{array}\right]$$

Doing this for $k = 1, 2, ..., n-1$ leads to the upper triangular system with the augmented matrix

$$\left[\begin{array}{cccc|c}
a_{1,1}^{(1)} & \cdots & & a_{1,n}^{(1)} & b_1^{(1)} \\
0 & \ddots & & \vdots & \vdots \\
\vdots & & \ddots & & \vdots \\
0 & \cdots & 0 & a_{n,n}^{(n)} & b_n^{(n)}
\end{array}\right]$$

We later remove the assumption

$$a_{k,k}^{(k)} \neq 0, \qquad k = 1, 2, ..., n$$

QUESTIONS

- How do we remove the assumption on the pivot elements?

- How many operations are involved in this procedure?

- How much error is there in the computed solution due to rounding errors in the calculations?

- How does the machine architecture affect the implementation of this algorithm.

PARTIAL PIVOTING

Recall the reduction of

$$[A^{(1)} \mid b^{(1)}] = \left[\begin{array}{ccc|c} a_{1,1}^{(1)} & \cdots & a_{1,n}^{(1)} & b_1^{(1)} \\ \vdots & \ddots & \vdots & \vdots \\ a_{n,1}^{(1)} & \cdots & a_{n,n}^{(1)} & b_n^{(1)} \end{array}\right]$$

to

$$[A^{(2)} \mid b^{(2)}] = \left[\begin{array}{cccc|c} a_{1,1}^{(1)} & a_{1,2}^{(1)} & \cdots & a_{1,n}^{(1)} & b_1^{(1)} \\ 0 & a_{2,2}^{(2)} & & a_{2,n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n,2}^{(2)} & \cdots & a_{n,n}^{(2)} & b_n^{(2)} \end{array}\right]$$

What if $a_{1,1}^{(1)} = 0$? In that case we look for an equation in which the x_1 is present. To do this in such a way as to avoid zero the maximum extant possible, we do the following.

Look at all the elements in the first column,

$$a_{1,1}^{(1)}, a_{2,1}^{(1)}, ..., a_{n,1}^{(1)}$$

and pick the largest in size. Say it is

$$\left|a_{k,1}^{(1)}\right| = \max_{j=1,...,n} \left|a_{j,1}^{(1)}\right|$$

Then interchange equations 1 and k, which means interchanging rows 1 and k in the augmented matrix $[A^{(1)} \mid b^{(1)}]$. Then proceed with the elimination of x_1 from equations 2 thru n as before.

Having obtained

$$[A^{(2)} \mid b^{(2)}] = \left[\begin{array}{cccc|c} a_{1,1}^{(1)} & a_{1,2}^{(1)} & \cdots & a_{1,n}^{(1)} & b_1^{(1)} \\ 0 & a_{2,2}^{(2)} & & a_{2,n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n,2}^{(2)} & \cdots & a_{n,n}^{(2)} & b_n^{(2)} \end{array}\right]$$

what if $a_{2,2}^{(2)} = 0$? Then we proceed as before.

Among the elements

$$a_{2,2}^{(2)}, a_{3,2}^{(2)}, \ldots, a_{n,2}^{(2)}$$

pick the one of largest size:

$$\left|a_{k,2}^{(2)}\right| = \max_{j=2,\ldots,n} \left|a_{j,2}^{(2)}\right|$$

Interchange rows 2 and k. Then proceed as before to eliminate x_2 from equations 3 thru n, thus obtaining

$$[A^{(3)} \mid b^{(3)}] = \left[\begin{array}{ccccc|c} a_{1,1}^{(1)} & a_{1,2}^{(1)} & a_{1,3}^{(1)} & \cdots & a_{1,n}^{(1)} & b_1^{(1)} \\ 0 & a_{2,2}^{(2)} & a_{2,3}^{(2)} & \cdots & a_{2,n}^{(2)} & b_2^{(2)} \\ 0 & 0 & a_{3,3}^{(3)} & \cdots & a_{3,n}^{(3)} & b_3^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & a_{n,3}^{(3)} & \cdots & a_{n,n}^{(3)} & b_n^{(3)} \end{array}\right]$$

This is done at every stage of the elimination process. This technique is called *partial pivoting*, and it is a part of most Gaussian elimination programs (including the one in the text).

Consequences of partial pivoting. Recall the definition of the elements obtained in the process of eliminating x_1 from equations 2 thru n.

$$m_{i,1} = \frac{a_{i,1}^{(1)}}{a_{1,1}^{(1)}}, \quad i = 2, ..., n$$

$$\begin{aligned} a_{i,j}^{(2)} &= a_{i,j}^{(1)} - m_{i,1} a_{1,j}^{(1)}, \quad j = 2, ..., n \\ b_i^{(2)} &= b_i^{(1)} - m_{i,1} b_1^{(1)} \end{aligned}$$

for $i = 2, ..., n$. By our definition of the pivot element $a_{1,1}^{(1)}$, we have

$$\left| m_{i,1} \right| \leq 1, \qquad i = 2, ..., n$$

Thus in the calculation of $a_{i,j}^{(2)}$ and $b_i^{(2)}$, we have that the elements do not grow rapidly in size. This is in comparison to what might happen otherwise, in which the multipliers $m_{i,1}$ might have been very large. This property is true of the multipliers at very step of the elimination process:

$$\left| m_{i,k} \right| \leq 1, \qquad i = k+1, ..., n, \quad k = 1, ..., n-1$$

The property

$$\left|m_{i,k}\right| \leq 1, \qquad i = k+1, ..., n$$

leads to good error propagation properties in Gaussian elimination with partial pivoting. The only error in Gaussian elimination is that derived from the rounding errors in the arithmetic operations. For example, at the first elimination step (eliminating x_1 from equations 2 thru n),

$$\begin{aligned} a_{i,j}^{(2)} &= a_{i,j}^{(1)} - m_{i,1} a_{1,j}^{(1)}, \quad j = 2, ..., n \\ b_i^{(2)} &= b_i^{(1)} - m_{i,1} b_1^{(1)} \end{aligned}$$

The above property on the size of the multipliers prevents these numbers and the errors in their calculation from growing as rapidly as they might if no partial pivoting was used.

As an example of the improvement in accuracy obtained with partial pivoting, see the example on pages 272-273.

OPERATION COUNTS

One of the major ways in which we compare the efficiency of different numerical methods is to count the number of needed arithmetic operations. For solving the linear system

$$\begin{aligned} a_{1,1}x_1 + \cdots + a_{1,n}x_n &= b_1 \\ &\vdots \\ a_{n,1}x_1 + \cdots + a_{n,n}x_n &= b_n \end{aligned}$$

using Gaussian elimination, we have the following operation counts.

1. $A \to U$, where we are converting $Ax = b$ to $Ux = g$:

Divisions	$\dfrac{n(n-1)}{2}$
Additions	$\dfrac{n(n-1)(2n-1)}{6}$
Multiplications	$\dfrac{n(n-1)(2n-1)}{6}$

2. $b \to g$:

Additions	$\frac{n(n-1)}{2}$
Multiplications	$\frac{n(n-1)}{2}$

3. Solving $Ux = g$:

Divisions	n
Additions	$\frac{n(n-1)}{2}$
Multiplications	$\frac{n(n-1)}{2}$

On some machines, the cost of a division is much more than that of a multiplication; whereas on others there is not any important difference. We assume the latter; and then the operation costs are as follows.

$$\begin{aligned} MD(A \to U) &= \frac{n\left(n^2-1\right)}{3} \\ MD(b \to g) &= \frac{n(n-1)}{2} \\ MD(\text{Find } x) &= \frac{n(n+1)}{2} \end{aligned}$$

$$
\begin{aligned}
AS(A \to U) &= \frac{n(n-1)(2n-1)}{6} \\
AS(b \to g) &= \frac{n(n-1)}{2} \\
AS(\text{Find } x) &= \frac{n(n-1)}{2}
\end{aligned}
$$

Thus the total number of operations is

$$
\begin{array}{cc}
\text{Additions} & \dfrac{2n^3+3n^2-5n}{6} \\
\left(\begin{array}{c}\text{Multiplications}\\ \text{and Divisions}\end{array}\right) & \dfrac{n^3+3n^2-n}{3}
\end{array}
$$

Both are around $\frac{1}{3}n^3$, and thus the total operations account is approximately

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

What happens to the cost when n is doubled?

Solving $Ax = b$ **and** $Ax = c$. What is the cost? Only the modification of the right side is different in these two cases. Thus the additional cost is

$$\begin{pmatrix} MD(b \to g) \\ MD(\text{Find } x) \end{pmatrix} = n^2$$

$$\begin{pmatrix} AS(b \to g) \\ AS(\text{Find } x) \end{pmatrix} = n(n-1)$$

The total is around $2n^2$ operations, which is quite a bit smaller than $\frac{2}{3}n^3$ when n is even moderately large, say $n = 100$.

Thus one can solve the linear system $Ax = c$ at little additional cost to that for solving $Ax = b$. This has important consequences when it comes to estimation of the error in computed solutions.

CALCULATING THE MATRIX INVERSE

Consider finding the inverse of a 3×3 matrix

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} = \Big[A_{*,1}, A_{*,2}, A_{*,3}\Big]$$

We want to find a matrix

$$X = \Big[X_{*,1}, X_{*,2}, X_{*,3}\Big]$$

for which

$$AX = I$$

$$A\Big[X_{*,1}, X_{*,2}, X_{*,3}\Big] = [e_1, e_2, e_3]$$

$$\Big[AX_{*,1}, AX_{*,2}, AX_{*,3}\Big] = [e_1, e_2, e_3]$$

This means we want to solve

$$AX_{*,1} = e_1, \quad AX_{*,2} = e_2, \quad AX_{*,3} = e_3$$

We want to solve three linear systems, all with the same matrix of coefficients A.

In augmented matrix notation, we want to work with

$$[A \mid I]$$

$$\left[\begin{array}{ccc|ccc} a_{1,1} & a_{1,2} & a_{1,3} & 1 & 0 & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & 0 & 1 & 0 \\ a_{3,1} & a_{3,2} & a_{3,3} & 0 & 0 & 1 \end{array}\right]$$

Then we proceed as before with a single linear system, only now we have three right hand sides. We first introduce zeros into positions 2 and 3 of column 1; and then we introduce zero into position 3 of column 2. Following that, we will need to solve three upper triangular linear systems by back substitution. See the numerical example on pages 274-276.

MATRIX INVERSE EXAMPLE

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

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

$m_{2,1} = 1 \downarrow \quad m_{3,1} = 1$

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

$$\downarrow$$

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

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

Then by using back substitution to solve for each column of the inverse, we obtain

$$A^{-1} = \begin{bmatrix} \frac{1}{6} & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{6} & \frac{1}{3} & -\frac{1}{2} \\ -\frac{1}{3} & \frac{1}{3} & 0 \end{bmatrix}$$

COST OF MATRIX INVERSION

In calculating A^{-1}, we are solving for the matrix $X = \left[X_{*,1}, X_{*,2}, \ldots, X_{*,n}\right]$ where

$$A\left[X_{*,1}, X_{*,2}, \ldots, X_{*,n}\right] = [e_1, e_2, \ldots, e_n]$$

and e_j is column j of the identity matrix. Thus we are solving n linear systems

$$AX_{*,1} = e_1, \quad AX_{*,2} = e_2, \ldots, \quad AX_{*,n} = e_n \quad (1)$$

all with the same coefficient matrix. Returning to the earlier operation counts for solving a single linear system, we have the following.

Cost of triangulating A: approx. $\frac{2}{3}n^3$ operations
Cost of solving $Ax = b$: $2n^2$ operations

Thus solving the n linear systems in (1) costs approximately

$$\frac{2}{3}n^3 + n\left(2n^2\right) = \frac{8}{3}n^3 \quad \text{operations, approximately}$$

It costs approximately four times as many operations to invert A as to solve a single system. With attention to the form of the right-hand sides in (1) this can be reduced to $2n^3$ operations.

Matlab MATRIX OPERATIONS

To solve the linear system $Ax = b$ in Matlab, use

$$x = A \setminus b$$

In Matlab, the command

$$\text{inv}\,(A)$$

will calculate the inverse of A.

There are many matrix operations built into Matlab, both for general matrices and for special classes of matrices. We do not discuss those here, but recommend the student to investigate these thru the Matlab help options.