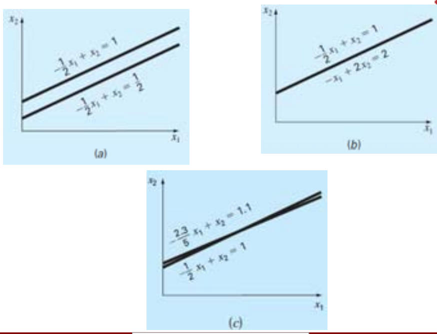
Lecture 09: Gauss Elimination



Outline

- Knowing how to solve small sets of linear equations with the graphical method and Cramer's rule.
- Understanding how to implement forward elimination and back substitution as in Gauss elimination.
- Understanding how to count flops to evaluate the efficiency of an algorithm.
- Understanding the concepts of singularity and ill condition.
- Understanding how partial pivoting is implemented and how it differs from complete pivoting.
- Knowing how to compute the determinant as part of the Gauss elimination algorithm with partial pivoting.
- Recognizing how the banded structure of a tridiagonal system can be exploited to obtain extremely efficient solutions.

Simultaneous Linear Algebraic Equations

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

- where the *a*'s are constant coefficients and the *b*'s are constants.

Gauss elimination

- Gauss elimination → combines equations to eliminate unknowns.
- Although it is one of the earliest methods for solving simultaneous equations, it remains among the most important algorithms in use today and is the basis for linear equation solving on many popular software packages.

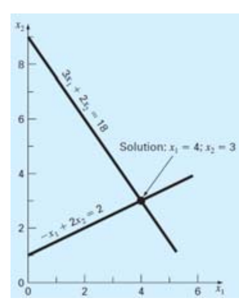
Solving Small Numbers of Equations

- For solving small ($n \le 3$) sets of simultaneous equations:
  - Graphical method
  - Cramer's rule
  - Elimination of unknowns
- These are called non-computer solution techniques.

Graphical Method

$$3x_1 + 2x_2 = 18$$
$$-x_1 + 2x_2 = 2$$

$$x_2 = -\frac{3}{2}x_1 + 9$$
$$x_2 = \frac{1}{2}x_1 + 1$$

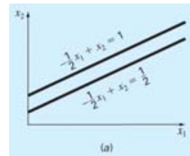


Graphical Method

- For three simultaneous equations, each equation would be represented by a plane in a three-dimensional coordinate system.
- The point where the three planes intersect would represent the solution.
- Beyond three equations, graphical methods break down and, consequently, have little practical value for solving simultaneous equations.
- However, they are useful in visualizing properties of the solutions.

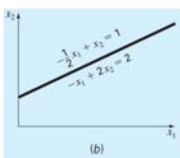
Graphical Method

- Three cases that can pose problems when solving sets of linear equations.



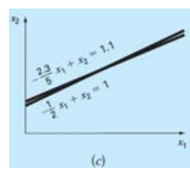
- Fig. *a* shows the case where the two equations represent parallel lines.
- For such situations, there is no solution because the lines never cross.

Graphical Method



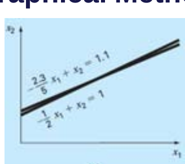
- Fig. *b* depicts the case where the two lines are coincident. For such situations there is an infinite number of solutions.
- Both types of systems are said to be singular.

Graphical Method



- In addition, systems that are very close to being singular (Fig. *c*) can also cause problems.
- These systems are said to be ill-conditioned.

Graphical Method



- Graphically, this corresponds to the fact that it is difficult to identify the exact point at which the lines intersect.
- Ill-conditioned systems will also pose problems when they are encountered during the numerical solution of linear equations.
- This is because they will be extremely sensitive to round-off error.

Determinants and Cramer's Rule

- Cramer's rule is another solution technique that is best suited to small numbers of equations.
- Determinant is used to implement Cramer's rule.
- In addition, the determinant has relevance to the evaluation of the ill-conditioning of a matrix.

Determinants

- Consider a set of three equations:

$$[A]\{x\} = \{b\}$$

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad D = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

- In contrast to a matrix, the determinant is a single number.

Determinants

- The determinant for two simultaneous equations

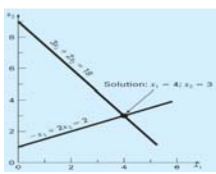
$$D = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \qquad D = a_{11}a_{22} - a_{12}a_{21}$$

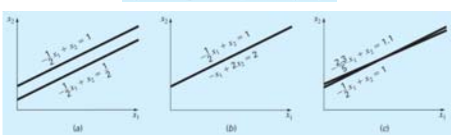
- For the third-order case, the determinant can be computed as

$$D = a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

- where the 2 × 2 determinants are called *minors.*

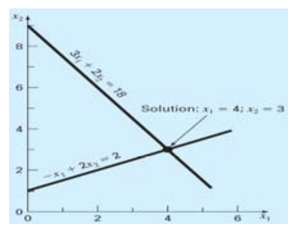
Example 1





Problem Statement. Compute values for the determinants of the systems represented in Figs. 9.1 and 9.2.

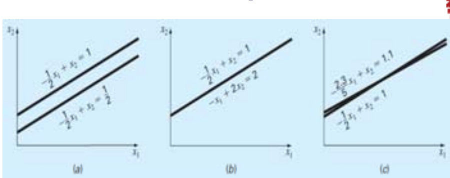
Example 1



Solution. For Fig. 9.1:

$$D = \begin{vmatrix} 3 & 2 \\ -1 & 2 \end{vmatrix} = 3(2) - 2(-1) = 8$$

Example 1



For Fig. 9.2*a*:

$$D = \begin{vmatrix} -\frac{1}{2} & 1 \\ -\frac{1}{2} & 1 \end{vmatrix} = -\frac{1}{2}(1) - 1\left(\frac{-1}{2}\right) = 0$$

For Fig. 9.2*b*:

$$D = \begin{vmatrix} -\frac{1}{2} & 1 \\ -1 & 2 \end{vmatrix} = -\frac{1}{2}(2) - 1(-1) = 0$$

For Fig. 9.2*c*:

$$D = \begin{vmatrix} -\frac{1}{2} & 1 \\ -\frac{2.3}{5} & 1 \end{vmatrix} = -\frac{1}{2}(1) - 1\left(\frac{-2.3}{5}\right) = -0.04$$

Cramer's Rule

- This rule states that each unknown in a system of linear algebraic equations may be expressed as a fraction of two determinants with denominator *D* and with the numerator obtained from *D* by replacing the column of coefficients of the unknown in question by the constants $b_1, b_2, \ldots, b_n$.

$$x_1 = \frac{\begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{32} & a_{33} \end{vmatrix}}{D}$$

Example 2

Problem Statement. Use Cramer's rule to solve

$$\begin{aligned} 0.3x_1 + 0.52x_2 + \quad x_3 &= -0.01 \\ 0.5x_1 + \quad x_2 + 1.9x_3 &= 0.67 \\ 0.1x_1 + 0.3\ x_2 + 0.5x_3 &= -0.44 \end{aligned}$$

Solution. The determinant D can be evaluated as [Eq. (9.1)]:

$$D = 0.3\begin{vmatrix} 1 & 1.9 \\ 0.3 & 0.5 \end{vmatrix} - 0.52\begin{vmatrix} 0.5 & 1.9 \\ 0.1 & 0.5 \end{vmatrix} + 1\begin{vmatrix} 0.5 & 1 \\ 0.1 & 0.3 \end{vmatrix} = -0.0022$$

The solution can be calculated as

$$x_1 = \frac{\begin{vmatrix} -0.01 & 0.52 & 1 \\ 0.67 & 1 & 1.9 \\ -0.44 & 0.3 & 0.5 \end{vmatrix}}{-0.0022} = \frac{0.03278}{-0.0022} = -14.9$$

$$x_2 = \frac{\begin{vmatrix} 0.3 & -0.01 & 1 \\ 0.5 & 0.67 & 1.9 \\ 0.1 & -0.44 & 0.5 \end{vmatrix}}{-0.0022} = \frac{0.0649}{-0.0022} = -29.5$$

$$x_3 = \frac{\begin{vmatrix} 0.3 & 0.52 & -0.01 \\ 0.5 & 1 & 0.67 \\ 0.1 & 0.3 & -0.44 \end{vmatrix}}{-0.0022} = \frac{-0.04356}{-0.0022} = 19.8$$

Cramer's Rule (Cont'd)

- For more than three equations, Cramer's rule becomes impractical because, as the number of equations increases, the determinants are time consuming to evaluate by hand (or by computer).
- Consequently, more efficient alternatives are used.

Elimination of Unknowns

- The elimination of unknowns by combining equations is an algebraic approach that can be illustrated for a set of two equations:

$$a_{11}x_1 + a_{12}x_2 = b_1$$

$$a_{21}x_1 + a_{22}x_2 = b_2$$

$$a_{21}a_{11}x_1 + a_{21}a_{12}x_2 = a_{21}b_1$$

$$a_{11}a_{21}x_1 + a_{11}a_{22}x_2 = a_{11}b_2$$

$$\boxed{a_{11}a_{22}x_2 - a_{21}a_{12}x_2 = a_{11}b_2 - a_{21}b_1}$$

$$\boxed{x_2 = \frac{a_{11}b_2 - a_{21}b_1}{a_{11}a_{22} - a_{21}a_{12}}}$$

$$\boxed{x_1 = \frac{a_{22}b_1 - a_{12}b_2}{a_{11}a_{22} - a_{21}a_{12}}}$$

Elimination of Unknowns

- The solutions follow directly from Cramer's rule:

$$\boxed{x_1 = \frac{a_{22}b_1 - a_{12}b_2}{a_{11}a_{22} - a_{21}a_{12}}} \qquad \boxed{x_2 = \frac{a_{11}b_2 - a_{21}b_1}{a_{11}a_{22} - a_{21}a_{12}}}$$

$$x_1 = \frac{\begin{vmatrix} b_1 & a_{12} \\ b_2 & a_{22} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}} = \frac{a_{22}b_1 - a_{12}b_2}{a_{11}a_{22} - a_{21}a_{12}}$$

$$x_2 = \frac{\begin{vmatrix} a_{11} & b_1 \\ a_{21} & b_2 \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}} = \frac{a_{11}b_2 - a_{21}b_1}{a_{11}a_{22} - a_{21}a_{12}}$$

Elimination of Unknowns

- The elimination of unknowns can be extended to systems with more than two or three equations.
- However, the numerous calculations that are required for larger systems make the method extremely tedious to implement by hand.
- However, as discussed next, the technique can be formalized and readily programmed for the computer.

Naive Gauss Elimination

- The elimination of unknowns consists of two steps:
  1. The equations were manipulated to eliminate one of the unknowns from the equations.

     The result of this elimination step was that we had one equation with one unknown.
  2. Consequently, this equation could be solved directly and the result back-substituted into one of the original equations to solve for the remaining unknown.

Naive Gauss Elimination

- Gauss elimination extends this approach to large sets of equations by developing a systematic scheme or algorithm to eliminate unknowns and to back-substitute.
- Although these techniques are ideally suited for implementation on computers, some modifications will be required to obtain a reliable algorithm.
- In particular, the computer program must avoid division by zero.
- The following method is called "naive" Gauss elimination because it does not avoid this problem.

Naive Gauss Elimination

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n &= b_n
\end{aligned}
$$

Naive Gauss Elimination

- Forward Elimination of Unknowns: The first phase is designed to reduce the set of equations to an upper triangular system.
- The initial step will be to eliminate the first unknown, x_1, from the second through the nth equations.

$$
\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{array}\right]
\;\Downarrow\;
\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ & a'_{22} & a'_{23} & b'_2 \\ & & a''_{33} & b''_3 \end{array}\right]
$$

Forward elimination

Naive Gauss Elimination

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n &= b_n
\end{aligned}
$$

- Multiply the 1st equation by a_{21}/a_{11} to give:

$$a_{21}x_1 + \frac{a_{21}}{a_{11}}a_{12}x_2 + \cdots + \frac{a_{21}}{a_{11}}a_{1n}x_n = \frac{a_{21}}{a_{11}}b_1$$

- Now, subtract this equation from the 2nd equation to give:

$$\left(a_{22} - \frac{a_{21}}{a_{11}}a_{12}\right)x_2 + \cdots + \left(a_{2n} - \frac{a_{21}}{a_{11}}a_{1n}\right)x_n = b_2 - \frac{a_{21}}{a_{11}}b_1$$

$$\boxed{a'_{22}x_2 + \cdots + a'_{2n}x_n = b'_2}$$

Naive Gauss Elimination

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n &= b'_2 \\
a'_{32}x_2 + a'_{33}x_3 + \cdots + a'_{3n}x_n &= b'_3 \\
&\vdots \\
a'_{n2}x_2 + a'_{n3}x_3 + \cdots + a'_{nn}x_n &= b'_n
\end{aligned}
$$

- For the foregoing steps, 1st equation is called the pivot equation and a_{11} is called the pivot coefficient or element.
- Note that the process of multiplying the first row by a_{21}/a_{11} is equivalent to dividing it by a_{11} and multiplying it by a_{21}.
- Sometimes the division operation is referred to as normalization.
- We make this distinction because a zero pivot element can interfere with normalization by causing a division by zero.

Naive Gauss Elimination

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n &= b'_2 \\
a''_{33}x_3 + \cdots + a''_{3n}x_n &= b''_3 \\
&\vdots \\
a''_{n3}x_3 + \cdots + a''_{nn}x_n &= b''_n
\end{aligned}
$$

- The double prime indicates that the elements have been modified twice.
- The final manipulation in the sequence is to use the $(n-1)$th equation to eliminate the x_{n-1} term from the nth equation.
- At this point, the system will have been transformed to an upper triangular system.

Naive Gauss Elimination

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n &= b'_2 \\
a''_{33}x_3 + \cdots + a''_{3n}x_n &= b''_3 \\
\vdots \qquad & \quad \vdots \\
a_{nn}^{(n-1)}x_n &= b_n^{(n-1)}
\end{aligned}
$$

```
DOFOR k = 1, n − 1
  DOFOR i = k + 1, n
    factor = a_i,k / a_k,k
    DOFOR j = k + 1 to n
      a_i,j = a_i,j − factor · a_k,j
    END DO
    b_i = b_i − factor · b_k
  END DO
END DO
```

- Notice that three nested loops provide a concise representation of the process.
- The outer loop moves down the matrix from one pivot row to the next.
- The middle loop moves below the pivot row to each of the subsequent rows where elimination is to take place.
- Finally, the innermost loop progresses across the columns to eliminate or transform the elements of a particular row.

Bangladesh University of Eng. & Tech. | Slide 31 of 78 | Industrial &Production Engineering

Naive Gauss Elimination

- Back Substitution:

$$
\left.
\begin{aligned}
x_3 &= b''_3/a''_{33} \\
x_2 &= (b'_2 - a'_{23}x_3)/a'_{22} \\
x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}
\end{aligned}
\right\} \text{Back substitution}
$$

$$a_{nn}^{(n-1)}x_n = b_n^{(n-1)} \qquad x_n = \frac{b_n^{(n-1)}}{a_{nn}^{(n-1)}}$$

$$x_i = \frac{b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)}x_j}{a_{ii}^{(i-1)}} \quad \text{for } i = n-1, n-2, \ldots, 1$$

Bangladesh University of Eng. & Tech. | Slide 32 of 78 | Industrial &Production Engineering

Naive Gauss Elimination

$$x_i = \frac{b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)}x_j}{a_{ii}^{(i-1)}}$$

```
x_n = b_n / a_n,n
DOFOR i = n − 1, 1, −1
  sum = b_i
  DOFOR j = i + 1, n
    sum = sum − a_i,j · x_j
  END DO
  x_i = sum / a_i,i
END DO
```

- A temporary variable, *sum*, is used to accumulate the summation from the equation for x_i.
- This results in a somewhat faster execution time than if the summation were accumulated in b_i.
- More importantly, it allows efficient improvement in precision if the variable, *sum*, is declared in double precision.

Bangladesh University of Eng. & Tech. | Slide 33 of 78 | Industrial &Production Engineering

Naive Gauss Elimination: Operations Counting

- The execution time of Gauss elimination depends on the amount of *floating-point operations* (or *flops*) involved in the algorithm.
- On modern computers using math coprocessors, the time consumed to perform addition/subtraction and multiplication/division is about the same.
- Therefore, totaling up these operations provides insight into which parts of the algorithm are most time consuming and how computation time increases as the system gets larger.

Bangladesh University of Eng. & Tech. | Slide 34 of 78 | Industrial &Production Engineering

Naive Gauss Elimination: Operations Counting

- Before analyzing naive Gauss elimination, we will first define some quantities that facilitate operation counting:

$$\sum_{i=1}^{m} cf(i) = c\sum_{i=1}^{m} f(i) \qquad \sum_{i=1}^{m} f(i) + g(i) = \sum_{i=1}^{m} f(i) + \sum_{i=1}^{m} g(i)$$

$$\sum_{i=1}^{m} 1 = 1 + 1 + 1 + \cdots + 1 = m \qquad \sum_{i=k}^{m} 1 = m - k + 1$$

$$\sum_{i=1}^{m} i = 1 + 2 + 3 + \cdots + m = \frac{m(m+1)}{2} = \frac{m^2}{2} + O(m)$$

$$\sum_{i=1}^{m} i^2 = 1^2 + 2^2 + 3^2 + \cdots + m^2 = \frac{m(m+1)(2m+1)}{6} = \frac{m^3}{3} + O(m^2)$$

- where $O(m^n)$ means "terms of order m^n and lower."

Bangladesh University of Eng. & Tech. | Slide 35 of 78 | Industrial &Production Engineering

Naive Gauss Elimination: Operations Counting

```
DOFOR k = 1, n − 1
  DOFOR i = k + 1, n
    factor = a_i,k / a_k,k
    DOFOR j = k + 1 to n
      a_i,j = a_i,j − factor · a_k,j
    END DO
    b_i = b_i − factor · b_k
  END DO
END DO
```

- On the first pass through the outer loop, $k = 1$.
- Therefore, the limits on the middle loop are from $i = 2$ to n.
- Thus, the number of iterations of the middle loop will be

$$\sum_{i=2}^{n} 1 = n - 2 + 1 = n - 1$$

Bangladesh University of Eng. & Tech. | Slide 36 of 78 | Industrial &Production Engineering

Naive Gauss Elimination: Operations Counting

```
DOFOR k = 1, n − 1
 DOFOR i = k + 1, n
   factor = a_i,k / a_k,k
   DOFOR j = k + 1 to n
     a_i,j = a_i,j − factor · a_k,j
   END DO
   b_i = b_i − factor · b_k
 END DO
END DO
```

- For every one of these iterations, there is one division to define the factor.
- The interior loop then performs a single multiplication and subtraction for each iteration from $j = 2$ to n.
- Finally, there is one additional multiplication and subtraction for the right-hand-side value.

Naive Gauss Elimination: Operations Counting

```
DOFOR k = 1, n − 1
 DOFOR i = k + 1, n
   factor = a_i,k / a_k,k
   DOFOR j = k + 1 to n
     a_i,j = a_i,j − factor · a_k,j
   END DO
   b_i = b_i − factor · b_k
 END DO
END DO
```

- Thus, for every iteration of the middle loop, the number of multiplications is

$$1 + [n - 2 + 1] + 1 = 1 + n$$

Naive Gauss Elimination: Operations Counting

```
DOFOR k = 1, n − 1
 DOFOR i = k + 1, n
   factor = a_i,k / a_k,k
   DOFOR j = k + 1 to n
     a_i,j = a_i,j − factor · a_k,j
   END DO
   b_i = b_i − factor · b_k
 END DO
END DO
```

- The total multiplications for the first pass through the outer loop is therefore obtained by $[n - 1](1 + n)$.
- In like fashion, the number of subtractions is computed as $[n - 1](n)$.

Naive Gauss Elimination: Operations Counting

- Similar reasoning can be used to estimate the flops for the subsequent iterations of the outer loop.

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

Naive Gauss Elimination: Operations Counting

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

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

$$\boxed{n(n+1)\sum_{k=1}^{n-1} 1 - (2n+1)\sum_{k=1}^{n-1} k + \sum_{k=1}^{n-1} k^2}$$

$$[n^3 + O(n)] - [n^3 + O(n^2)] + \left[\frac{1}{3}n^3 + O(n^2)\right] = \frac{n^3}{3} + O(n)$$

Naive Gauss Elimination: Operations Counting

- A similar analysis for the multiplication/division flops yields

$$[n^3 + O(n^2)] - [n^3 + O(n)] + \left[\frac{1}{3}n^3 + O(n^2)\right] = \frac{n^3}{3} + O(n^2)$$

- The total number of flops is

$$\boxed{\frac{2n^3}{3} + O(n^2)}$$

- The result is written in this way because as n gets large, the $O(n^2)$ and lower terms become negligible.
- We are therefore justified in concluding that for large n, the effort involved in forward elimination converges on $2n^3/3$.

Naive Gauss Elimination: Operations Counting

```
x_n = b_n / a_n,n
DOFOR i = n − 1, 1, −1
  sum = b_i
  DOFOR j = i + 1, n
    sum = sum − a_i,j · x_j
  END DO
  x_i = sum / a_i,i
END DO
```

- Because only a single loop is used, back substitution is much simpler to evaluate.
- The number of addition/subtraction flops is equal to $n(n-1)/2$.

Naive Gauss Elimination: Operations Counting

```
x_n = b_n / a_n,n
DOFOR i = n − 1, 1, −1
  sum = b_i
  DOFOR j = i + 1, n
    sum = sum − a_i,j · x_j
  END DO
  x_i = sum / a_i,i
END DO
```

- Because of the extra division prior to the loop, the number of multiplication/division flops is $n(n+1)/2$.
- These can be added to arrive at a total of

$$\boxed{n^2 + O(n)}$$

Naive Gauss Elimination: Operations Counting

- Thus, the total effort in naive Gauss elimination can be represented as

$$\underbrace{\frac{2n^3}{3} + O(n^2)}_{\text{Forward elimination}} + \underbrace{n^2 + O(n)}_{\text{Backward substitution}} \xrightarrow{\text{as } n \text{ increases}} \frac{2n^3}{3} + O(n^2)$$

- Two useful general conclusions can be drawn from this analysis:
  1. As the system gets larger, the computation time increases greatly.
  2. Most of the effort is incurred in the elimination step.

Naive Gauss Elimination: Operations Counting

n	Elimination	Back Substitution	Total Flops	$2n^3/3$	Percent Due to Elimination
10	705	100	805	667	87.58%
100	671550	10000	681550	666667	98.53%
1000	6.67×10^8	1×10^6	6.68×10^8	6.67×10^8	99.85%

- The amount of flops increases nearly three orders of magnitude for every order of magnitude increase in the dimension.

Pitfalls of Elimination Methods

- **Division by Zero**
- The primary reason that the foregoing technique is called "naive" is that during both the elimination and the back-substitution phases, it is possible that a division by zero can occur.

$$2x_2 + 3x_3 = 8$$
$$4x_1 + 6x_2 + 7x_3 = -3$$
$$2x_1 + x_2 + 6x_3 = 5$$

- Problems also can arise when a coefficient is very close to zero.
- The technique of pivoting has been developed to partially avoid these problems.

Pitfalls of Elimination Methods

- **Round-Off Errors**
- The problem of round-off error can become particularly important when large numbers of equations are to be solved.
- This is due to the fact that every result is dependent on previous results.
- Consequently, an error in the early steps will tend to propagate—that is, it will cause errors in subsequent steps.

Pitfalls of Elimination Methods

- **Round-Off Errors**
- Specifying the system size where round-off error becomes significant is complicated by the fact that the type of computer and the properties of the equations are determining factors.
- A rough rule of thumb is that round-off error may be important when dealing with 100 or more equations.
- In any event, you should always substitute your answers back into the original equations to check whether a substantial error has occurred.
- However, as discussed below, the magnitudes of the coefficients themselves can influence whether such an error check ensures a reliable result.

Pitfalls of Elimination Methods

- **Ill-Conditioned Systems**
- The adequacy of the solution depends on the condition of the system.
- *Well-conditioned systems are those where a small change in one or more of the coefficients results* in a similar small change in the solution.
- *Ill-conditioned systems are those where small* changes in coefficients result in large changes in the solution.
- An alternative interpretation of ill-conditioning is that a wide range of answers can approximately satisfy the equations.
- Because round-off errors can induce small changes in the coefficients, these artificial changes can lead to large solution errors for ill-conditioned systems.

Example 2

Ill-Conditioned Systems

Problem Statement. Solve the following system:

$$x_1 + 2x_2 = 10 \quad \text{(E9.6.1)}$$
$$1.1x_1 + 2x_2 = 10.4 \quad \text{(E9.6.2)}$$

Then, solve it again, but with the coefficient of x_1 in the second equation modified slightly to 1.05.

Solution. Using Eqs. (9.10) and (9.11), the solution is

$$x_1 = \frac{2(10) - 2(10.4)}{1(2) - 2(1.1)} = 4 \qquad x_2 = \frac{1(10.4) - 1.1(10)}{1(2) - 2(1.1)} = 3$$

However, with the slight change of the coefficient a_{21} from 1.1 to 1.05, the result is changed dramatically to

$$x_1 = \frac{2(10) - 2(10.4)}{1(2) - 2(1.05)} = 8 \qquad x_2 = \frac{1(10.4) - 1.1(10)}{1(2) - 2(1.05)} = 1$$

Example 2

$$8 + 2(1) = 10 = 10$$
$$1.1(8) + 2(1) = 10.8 \cong 10.4$$

Therefore, although $x_1 = 8$ and $x_2 = 1$ is not the true solution to the original problem, the error check is close enough to possibly mislead you into believing that your solutions are adequate.

Pitfalls of Elimination Methods

- **Ill-Conditioned Systems**
- We can mathematically characterize this situation by writing the two equations in general form:

$$a_{11}x_1 + a_{12}x_2 = b_1$$
$$a_{21}x_1 + a_{22}x_2 = b_2$$

$$x_2 = -\frac{a_{11}}{a_{12}}x_1 + \frac{b_1}{a_{12}}$$
$$x_2 = -\frac{a_{21}}{a_{22}}x_1 + \frac{b_2}{a_{22}}$$

- Consequently, if the slopes are nearly equal,

$$\frac{a_{11}}{a_{12}} \cong \frac{a_{21}}{a_{22}}$$

Pitfalls of Elimination Methods

- **Ill-Conditioned Systems**

$$\frac{a_{11}}{a_{12}} \cong \frac{a_{21}}{a_{22}} \qquad a_{11}a_{22} \cong a_{12}a_{21} \qquad \boxed{a_{11}a_{22} - a_{12}a_{21} \cong 0}$$

- We arrive at the general conclusion that an ill-conditioned system is one with a determinant close to zero.
- It is difficult to specify how close to zero the determinant must be to indicate ill-conditioning.
- This is complicated by the fact that the determinant can be changed by multiplying one or more of the equations by a scale factor without changing the solution.
- Consequently, the determinant is a relative value that is influenced by the magnitude of the coefficients.

Example 3

Problem Statement. Evaluate the determinant of the following systems:

(a) From Example 9.1:

$$3x_1 + 2x_2 = 18 \quad \text{(E9.7.1)}$$
$$-x_1 + 2x_2 = 2 \quad \text{(E9.7.2)}$$

(b) From Example 9.6:

$$x_1 + 2x_2 = 10 \quad \text{(E9.7.3)}$$
$$1.1x_1 + 2x_2 = 10.4 \quad \text{(E9.7.4)}$$

(c) Repeat (*b*) but with the equations multiplied by 10.

Solution.

(a) The determinant of Eqs. (E9.7.1) and (E9.7.2), which are well-conditioned, is

$$D = 3(2) - 2(-1) = 8$$

(b) The determinant of Eqs. (E9.7.3) and (E9.7.4), which are ill-conditioned, is

$$D = 1(2) - 2(1.1) = -0.2$$

Example 3

(c) The results of (*a*) and (*b*) seem to bear out the contention that ill-conditioned systems have near-zero determinants. However, suppose that the ill-conditioned system in (*b*) is multiplied by 10 to give

$$10x_1 + 20x_2 = 100$$
$$11x_1 + 20x_2 = 104$$

The multiplication of an equation by a constant has no effect on its solution. In addition, it is still ill-conditioned. This can be verified by the fact that multiplying by a constant has no effect on the graphical solution. However, the determinant is dramatically affected:

$$D = 10(20) - 20(11) = -20$$

Not only has it been raised two orders of magnitude, but it is now over twice as large as the determinant of the well-conditioned system in (*a*).

Pitfalls of Elimination Methods

- **Ill-Conditioned Systems**
- The magnitude of the coefficients interjects a scale effect that complicates the relationship between system condition and determinant size.
- One way to partially circumvent this difficulty is to scale the equations so that the maximum element in any row is equal to 1.

Example 4

Problem Statement. Evaluate the determinant of the following systems:

(a) From Example 9.1:

$$3x_1 + 2x_2 = 18 \quad \text{(E9.7.1)}$$
$$-x_1 + 2x_2 = 2 \quad \text{(E9.7.2)}$$

(b) From Example 9.6:

$$x_1 + 2x_2 = 10 \quad \text{(E9.7.3)}$$
$$1.1x_1 + 2x_2 = 10.4 \quad \text{(E9.7.4)}$$

(c) Repeat (*b*) but with the equations multiplied by 10.

Scaling

Problem Statement. Scale the systems of equations in Example 9.7 to a maximum value of 1 and recompute their determinants.

Example 4

Solution.

(a) For the well-conditioned system, scaling results in

$$x_1 + 0.667x_2 = 6$$
$$-0.5x_1 + \quad x_2 = 1$$

for which the determinant is

$$D = 1(1) - 0.667(-0.5) = 1.333$$

(b) For the ill-conditioned system, scaling gives

$$0.5x_1 + x_2 = 5$$
$$0.55x_1 + x_2 = 5.2$$

for which the determinant is

$$D = 0.5(1) - 1(0.55) = -0.05$$

(c) For the last case, scaling changes the system to the same form as in (*b*) and the determinant is also −0.05. Thus, the scale effect is removed.

Determinant Evaluation Using Gauss Elimination

- The method is based on the fact that the determinant of a triangular matrix can be simply computed as the product of its diagonal elements:

$$\boxed{D = a_{11}a_{22}a_{33}\cdots a_{nn}}$$

$$D = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{vmatrix} \qquad D = a_{11}\begin{vmatrix} a_{22} & a_{23} \\ 0 & a_{33} \end{vmatrix} - a_{12}\begin{vmatrix} 0 & a_{23} \\ 0 & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} 0 & a_{22} \\ 0 & 0 \end{vmatrix}$$

$$D = a_{11}a_{22}a_{33} - a_{12}(0) + a_{13}(0) = a_{11}a_{22}a_{33}$$

Determinant Evaluation Using Gauss Elimination

- Recall that the forward-elimination step of Gauss elimination results in an upper triangular system.
- Because the value of the determinant is not changed by the forward-elimination process, the determinant can be simply evaluated at the end of this step via

$$D = a_{11} a'_{22} a''_{33} \cdots a_{nn}^{(n-1)}$$

- There is a slight modification to the above approach when the program employs partial pivoting.
- For this case, the determinant changes sign every time a row is pivoted.

$$D = a_{11} a'_{22} a''_{33} \cdots a_{nn}^{(n-1)} (-1)^p$$

Where *p* represents the number of times that rows are pivoted.

Pitfalls of Elimination Methods

- **Singular Systems**
- When the two equations are identical, we would lose one degree of freedom, and would be dealing with the impossible case of $n-1$ equations with n unknowns.
- Such cases might not be obvious to you, particularly when dealing with large equation sets.
- Consequently, it would be nice to have some way of automatically detecting singularity.
- A computer algorithm can test to discern whether a zero diagonal element is created during the elimination stage.
- If one is discovered, the calculation can be immediately terminated and a message displayed alerting the user.

Techniques for Improving Solutions

- **Use of More Significant Figures**
- **Pivoting**
- **Scaling**

Techniques for Improving Solutions

- **Pivoting**
- Before each row is normalized, it is advantageous to determine the largest available coefficient in the column below the pivot element.
- The rows can then be switched so that the largest element is the pivot element → *partial pivoting*.
- If columns as well as rows are searched for the largest element and then switched, the procedure is called *complete pivoting*.
- Complete pivoting is rarely used because switching columns changes the order of the *x*'s and, consequently, adds significant and usually unjustified complexity to the computer program.
- Aside from avoiding division by zero, pivoting also minimizes round-off error.
- As such, it also serves as a partial remedy for ill-conditioning.

Techniques for Improving Solutions

- **Pivoting**
- General-purpose computer programs must include a pivot strategy.
- The algorithm consists of two major loops.
- After storing the current pivot element and its row number as the variables, *big* and *p*, the first loop compares the pivot element with the elements below it to check whether any of these is larger than the pivot element.
- The second loop switches the original pivot row with the one with the largest element so that the latter becomes the new pivot row.

```
p = k
big = |a_k,k|
DOFOR ii = k+1, n
  dummy = |a_ii,k|
  IF (dummy > big)
     big = dummy
     p = ii
  END IF
END DO
IF (p ≠ k)
  DOFOR jj = k, n
     dummy = a_p,jj
     a_p,jj = a_k,jj
     a_k,jj = dummy
  END DO
  dummy = b_p
  b_p = b_k
  b_k = dummy
END IF
```


Techniques for Improving Solutions

- **Scaling**
- Scaling has value in standardizing the size of the determinant.
- Besides, it has utility in minimizing round-off errors for those cases where some of the equations in a system have much larger coefficients than others.
- Such situations are frequently encountered in engineering practice when widely different units are used in the development of simultaneous equations.
- As long as each equation is consistent, the system will be technically correct and solvable.
- However, the use of widely differing units can lead to coefficients of widely differing magnitudes.
- This, in turn, can have an impact on round-off error as it affects pivoting.

Example 3

Effect of Scaling on Pivoting and Round-Off

Problem Statement.

(a) Solve the following set of equations using Gauss elimination and a pivoting strategy:

$$2x_1 + 100{,}000x_2 = 100{,}000$$
$$x_1 + \quad x_2 = 2$$

(b) Repeat the solution after scaling the equations so that the maximum coefficient in each row is 1.

(c) Finally, use the scaled coefficients to determine whether pivoting is necessary. However, actually solve the equations with the original coefficient values. For all cases, retain only three significant figures. Note that the correct answers are $x_1 = 1.00002$ and $x_2 = 0.99998$ or, for three significant figures, $x_1 = x_2 = 1.00$.

Example 3

(a) Without scaling, forward elimination is applied to give

$$2x_1 + 100{,}000x_2 = 100{,}000$$
$$-50{,}000x_2 = -50{,}000$$

which can be solved by back substitution for

$$x_2 = 1.00$$
$$x_1 = 0.00$$

Although x_2 is correct, x_1 is 100 percent in error because of round-off.

Example 3

(b) Scaling transforms the original equations to

$$0.00002x_1 + x_2 = 1$$
$$x_1 + x_2 = 2$$

Therefore, the rows should be pivoted to put the greatest value on the diagonal.

$$x_1 + x_2 = 2$$
$$0.00002x_1 + x_2 = 1$$

Forward elimination yields

$$x_1 + x_2 = 2$$
$$x_2 = 1.00$$

which can be solved for

$$x_1 = x_2 = 1$$

Thus, scaling leads to the correct answer.

Example 3

(c) The scaled coefficients indicate that pivoting is necessary. We therefore pivot but retain the original coefficients to give

$$x_1 + \quad x_2 = 2$$
$$2x_1 + 100{,}000x_2 = 100{,}000$$

Forward elimination yields

$$x_1 + \quad x_2 = 2$$
$$100{,}000x_2 = 100{,}000$$

which can be solved for the correct answer: $x_1 = x_2 = 1$. Thus, scaling was useful in determining whether pivoting was necessary, but the equations themselves did not require scaling to arrive at a correct result.

Techniques for Improving Solutions

- **Scaling**
- In the previous example, scaling has utility in minimizing round-off.
- However, it should be noted that scaling itself also leads to round-off.
- For example, given the equation

$$2x_1 + 300{,}000x_2 = 1 \qquad 0.00000667x_1 + x_2 = 0.00000333$$

- Thus, scaling introduces a round-off error to the first coefficient and the right-hand-side constant.
- For this reason, it is sometimes suggested that scaling should be used to calculate scaled values for the coefficients solely as a criterion for pivoting.
- The original coefficient values are retained for the actual elimination and substitution computations.

Gauss-Jordan

- The Gauss-Jordan method is a variation of Gauss elimination.
- The major difference is that when an unknown is eliminated in the Gauss-Jordan method, it is eliminated from all other equations rather than just the subsequent ones.
- In addition, all rows are normalized by dividing them by their pivot elements.
- Thus, the elimination step results in an identity matrix rather than a triangular matrix.
- Consequently, it is not necessary to employ back substitution to obtain the solution.

Gauss-Jordan

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \vdots & b_1 \\ a_{21} & a_{22} & a_{23} & \vdots & b_2 \\ a_{31} & a_{32} & a_{33} & \vdots & b_3 \end{bmatrix}$$

$$\downarrow$$

$$\begin{bmatrix} 1 & 0 & 0 & \vdots & b_1^{(n)} \\ 0 & 1 & 0 & \vdots & b_2^{(n)} \\ 0 & 0 & 1 & \vdots & b_3^{(n)} \end{bmatrix}$$

$$\downarrow$$

$$\begin{aligned} x_1 &= b_1^{(n)} \\ x_2 &= b_2^{(n)} \\ x_3 &= b_3^{(n)} \end{aligned}$$

Example 4

Gauss-Jordan Method

Problem Statement. Use the Gauss-Jordan technique to solve the same system as in Example 9.5:

$$\begin{aligned} 3x_1 - 0.1x_2 - 0.2x_3 &= 7.85 \\ 0.1x_1 + 7x_2 - 0.3x_3 &= -19.3 \\ 0.3x_1 - 0.2x_2 + 10x_3 &= 71.4 \end{aligned}$$

Solution. First, express the coefficients and the right-hand side as an augmented matrix:

$$\begin{bmatrix} 3 & -0.1 & -0.2 & 7.85 \\ 0.1 & 7 & -0.3 & -19.3 \\ 0.3 & -0.2 & 10 & 71.4 \end{bmatrix}$$

Example 4

Solution. First, express the coefficients and the right-hand side as an augmented matrix:

$$\begin{bmatrix} 3 & -0.1 & -0.2 & 7.85 \\ 0.1 & 7 & -0.3 & -19.3 \\ 0.3 & -0.2 & 10 & 71.4 \end{bmatrix}$$

Then normalize the first row by dividing it by the pivot element, 3, to yield

$$\begin{bmatrix} 1 & -0.0333333 & -0.066667 & 2.61667 \\ 0.1 & 7 & -0.3 & -19.3 \\ 0.3 & -0.2 & 10 & 71.4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -0.0333333 & -0.066667 & 2.61667 \\ 0 & 7.00333 & -0.293333 & -19.5617 \\ 0 & -0.190000 & 10.0200 & 70.6150 \end{bmatrix}$$

Example 4

$$\begin{bmatrix} 1 & -0.0333333 & -0.066667 & 2.61667 \\ 0 & 7.00333 & -0.293333 & -19.5617 \\ 0 & -0.190000 & 10.0200 & 70.6150 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -0.0333333 & -0.066667 & 2.61667 \\ 0 & 1 & -0.0418848 & -2.79320 \\ 0 & -0.190000 & 10.0200 & 70.6150 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -0.0680629 & 2.52356 \\ 0 & 1 & -0.0418848 & -2.79320 \\ 0 & 0 & 10.01200 & 70.0843 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -0.0680629 & 2.52356 \\ 0 & 1 & -0.0418848 & -2.79320 \\ 0 & 0 & 1 & 7.0000 \end{bmatrix}$$

$$\boxed{\begin{bmatrix} 1 & 0 & 0 & 3.0000 \\ 0 & 1 & 0 & -2.5000 \\ 0 & 0 & 1 & 7.0000 \end{bmatrix}}$$

Gauss-Jordan

- Although the Gauss-Jordan technique and Gauss elimination might appear almost identical, the former requires more work.
- Using a similar approach as discussed earlier, it can be determined that the number of flops involved in naive Gauss-Jordan is

$$n^3 + n^2 - n \xrightarrow{\text{as } n \text{ increases}} n^3 + O(n^2)$$

- Thus, Gauss-Jordan involves approximately 50 percent more operations than Gauss elimination.

Assignment-9

- Write a pseudo code for Gauss-Jordan method.
- Prove that the number of flops involved in naive Gauss-Jordan is

$$n^3 + n^2 - n \xrightarrow{\text{as } n \text{ increases}} n^3 + O(n^2)$$

- Problems: 9.7, 9.11.