Solution of Systems of Linear Equations

1. Previously, we learnt how to determine roots of systems of simultaneous equations in two variables.
2. These were non-linear equations that were tackled using a variation of bracketing and Newton-Raphson techniques.
3. However, if a set of equations are linear, it is often far more efficient to use a dedicated set of techniques that employ matrix algebra.

$$\begin{array}{c}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\
\cdot \qquad\qquad \cdot \\
\cdot \qquad\qquad \cdot \\
\cdot \qquad\qquad \cdot \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n
\end{array}$$

Structure of linear equation system

Solution of Systems of Linear Equations

4. The simplest example of a simultaneous equation system is that of two equations.

5. Usually systems of three or less equations can be solved graphically, by Cramer’s Rule or by elimination of unknown variables.

6. Typically, ***if there are*** common roots of any two equations, then they can be evaluated at the intersection point of the two functions when they are plotted.

7. Consider two such equations:

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

8. Firstly, we recognise that ***there are two equations and two unknowns, so that it will be possible to solve for both unknowns x_1 and x_2.*** (We assume here that the values of the coefficients aij and bi are all known.)

Solution of Systems of Linear Equations

1. To progress the solution further, we isolate a different one of each of the variables of interest in both equations as follows:

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

$$\Downarrow$$

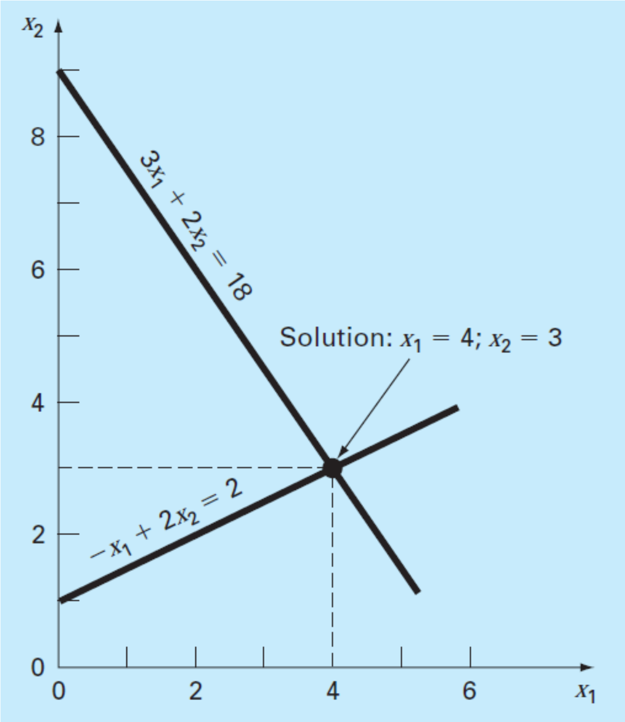
$$x_2 = -\left(\frac{a_{11}}{a_{12}}\right)x_1 + \frac{b_1}{a_{12}}$$
$$x_2 = -\left(\frac{a_{21}}{a_{22}}\right)x_1 + \frac{b_2}{a_{22}}$$

2. Note that these equations can now be plotted as two straight lines on a common plot, and if they possess solutions, then the solution will be visually inspectable by recording the co-ordinates of their intersection point (right).

3. A numerical example is given as follows:

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

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

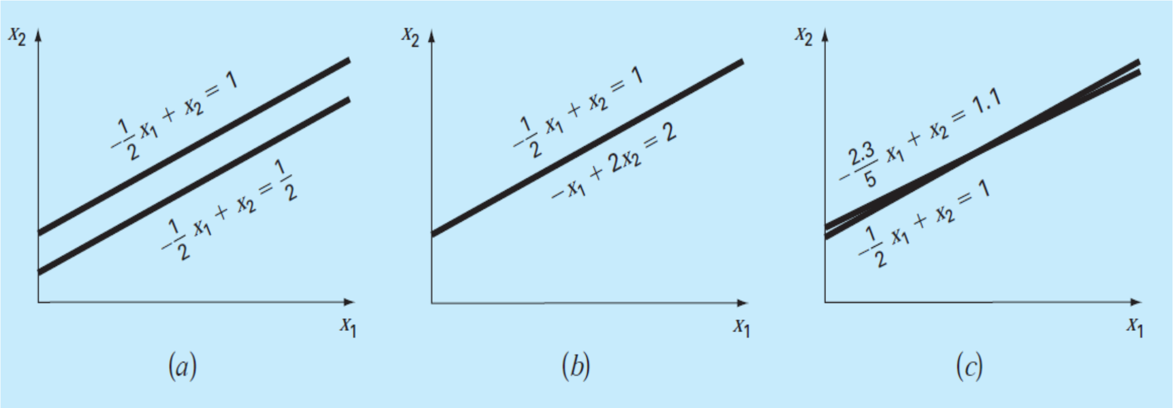


Graphical solution of two simultaneous equations

Solution of Systems of Linear Equations

1. The previous equation system was an example of a well-conditioned problem, i.e., the two equations possessed common solutions [x_1, x_2] represented graphically as an intersection of their two graphs.

2. Not all two-equation systems behave as well as this: in the graphs below, three situations are depicted:



3. a) No solution (no intersection); b) infinite solutions (concurring equations); c) ill-conditioned solution (slopes are too close to be efficiently detected visually)

Solution of Systems of Linear Equations

4. The first two scenarios are not resolvable as they are intrinsic properties of the equation system.

5. However, the third one is solvable, and it’s a question of finding a suitable technique to determine the solutions in a non-graphical manner (numerical).

6. The first technique we will look at is Cramer’s Rule which required famiarity with matrix algebra and determinants.

7. Given a set of linear equations expressed by: $[A]\{X\} = \{B\}$

8. [A] is the coefficient matrix

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

9. The determinant of A is given by:

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

Solution of Systems of Linear Equations

1. The determinant is not a matrix but one number generated by the following operation:

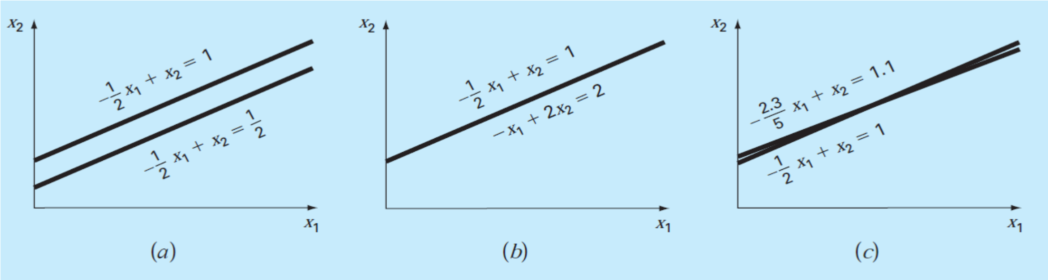
$$D = a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

2. In this case, the determinant is a function of various elements of the matrix and determinants of minors of matrix A.

3. A representative calculation of a determinant for one of the minors of A is:

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

4. Let us now calculate the determinants for each of the three scenarios in the following graphs: **Exercise**



Solution of Systems of Linear Equations

5. For a): $$D = \begin{vmatrix} -1/2 & 1 \\ -1/2 & 1 \end{vmatrix} = \frac{-1}{2}(1) - 1\left(\frac{-1}{2}\right) = 0$$

6. For b): $$D = \begin{vmatrix} -1/2 & 1 \\ -1 & 2 \end{vmatrix} = \frac{-1}{2}(2) - 1(-1) = 0$$

7. For c): $$D = \begin{vmatrix} -1/2 & 1 \\ -2.3/5 & 1 \end{vmatrix} = \frac{-1}{2}(1) - 1\left(\frac{-2.3}{5}\right) = -0.04$$

8. Straightaway we see that the value of the determinants for a) and b) are both zero. This is no coincidence but is a consequence of the fact that there is no solution of these two equation systems.

9. For c) however, there is a solution, but as it is very close to zero, it can be said that the problem is ill-conditioned.

10. The first two systems are said to be singular, and the latter is almost singular.

Cramer's Rule for up to Three Equations

1. Cramer's rule allows the solution of each unknown to be written as follows for the example of x_1:

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

2. **Example:** Take the following system of three equations:

$$0.3x_1 + 0.52x_2 + x_3 = -0.01$$
$$0.5x_1 + x_2 + 1.9x_3 = 0.67$$
$$0.1x_1 + 0.3x_2 + 0.5x_3 = -0.44$$

3. First, the determinant of the system is written from the nine coefficients of the three equations as follows:

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

Cramer's Rule for up to Three Equations

1. Then, the three minors of the system can be written down as follows:

$$A_1 = \begin{vmatrix} 1 & 1.9 \\ 0.3 & 0.5 \end{vmatrix} = 1(0.5) - 1.9(0.3) = -0.07$$

$$A_2 = \begin{vmatrix} 0.5 & 1.9 \\ 0.1 & 0.5 \end{vmatrix} = 0.5(0.5) - 1.9(0.1) = 0.06$$

$$A_3 = \begin{vmatrix} 0.5 & 1 \\ 0.1 & 0.3 \end{vmatrix} = 0.5(0.3) - 1(0.1) = 0.05$$

2. The determinant is now calculated as follows from the coefficients and the minors:

$$D = 0.3(-0.07) - 0.52(0.06) + 1(0.05) = -0.0022$$

3. Now, Cramer's Rule is used to calculate the three solutions for x_1, x_2, and x_3 as follows:

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

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

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

Elimination of Unknowns

1. Cramer's rule is 100% effective in cases where a solution exists (i.e., the system is not singular), but it is not computationally efficient for systems involving more than three equations. (Try doing this for four equations).

2. A more effective technique is the elimination of unknowns: here we illustrate the use of this for two equations to start with:

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

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

3. The basis of this technique is to multiply both equations with different coefficients from the other equation to try to eliminate one unknown from the system, and solve for the other one first.

4. In this case, we multiply the first equation by a_{21}, and the lower one by a_{11} to eliminate x_1 from the system and solve for x_2:

Elimination of Unknowns

$$a_{11}a_{21}x_1 + a_{12}a_{21}x_2 = b_1a_{21}$$

$$a_{21}a_{11}x_1 + a_{22}a_{11}x_2 = b_2a_{11}$$

4. Then, we subtract the lower equation from the upper one to enable us to solve for x_2:

$$a_{22}a_{11}x_2 - a_{12}a_{21}x_2 = b_2a_{11} - b_1a_{21}$$

4. We isolate x_2 from the above as follows:

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

4. Now we reverse substitute the value of x_2 into either of the original two equations to solve for x_1: e.g. if into the first one, then:

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

7. Examination of the solutions to both equations will show that they are simply variations of Cramer’s Rule!

Naïve (Simple) Gauss Elimination

1. Naïve Gauss Elimination is a technique that systematises the process of ***forward elimination*** and ***reverse substitution*** that we witnessed in the elimination of unknowns technique.

2. Consider a system of linear equations as follows:

$$
\begin{array}{l}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \\
\qquad\quad\;\, \vdots \qquad\qquad\qquad\;\; \vdots \\
a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n = b_n
\end{array}
$$

3. The first stage in the Gauss method is to implement the following manipulation of equation 1 in this series as follows:

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

4. Now this equation is subtracted from the second in the series to give

Naïve (Simple) Gauss Elimination

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

5. The coefficients in the original entries of the coefficient matrix are now replaced so that the new equation above is labelled

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

6. Note: the first equation remains unchanged in the revised matrix after this and subsequent operations.

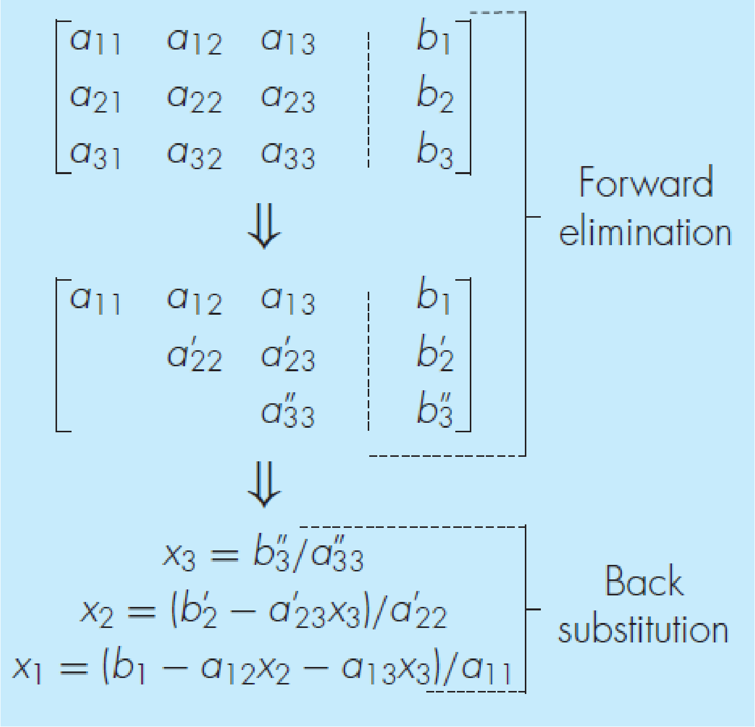
7. The second phase is a repeat of this manipulation from the third equation downwards, to eliminate the second unknown in all of these equations.

8. The resulting series of equations is represented by the following:

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

Naïve Gauss Elimination

1. Implementation of NGE for a three equation system is shown in the figure:



$$\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{array}\right]$$

$$\Downarrow$$

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

Forward elimination

$$\Downarrow$$

$$x_3 = b''_3/a''_{33}$$
$$x_2 = (b'_2 - a'_{23}x_3)/a'_{22}$$
$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$

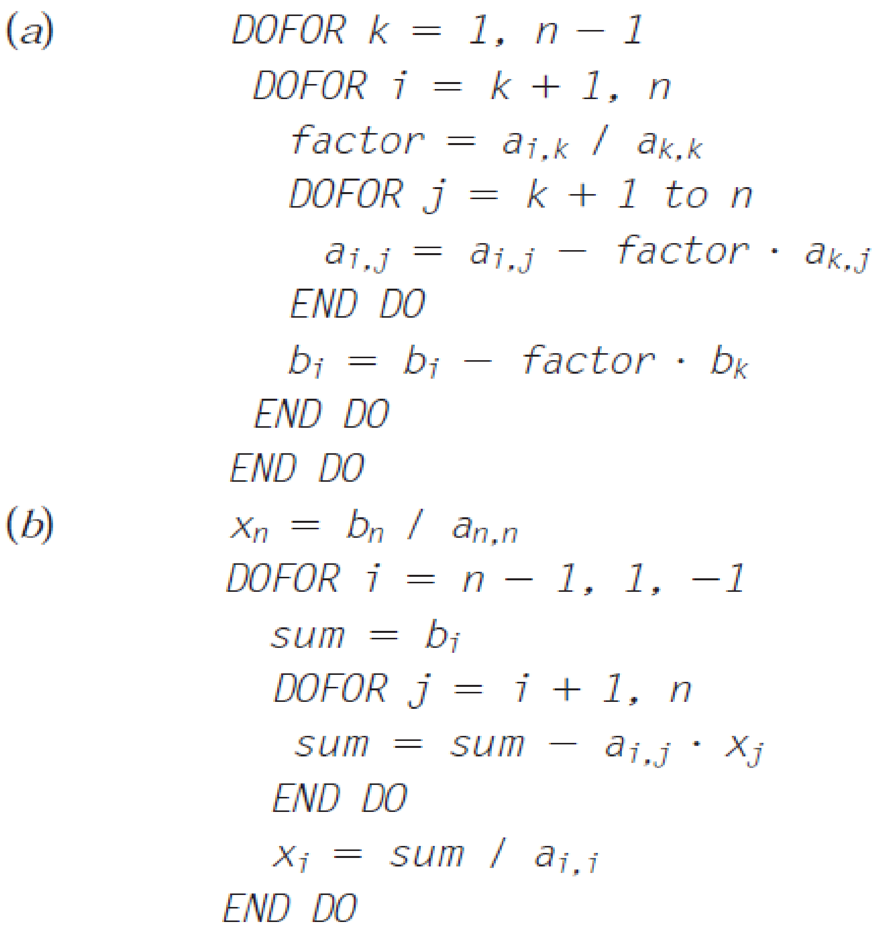
Back substitution

2. The series of manipulations results in an upper triangular matrix of coefficients. The number of primes on each elements denotes the number of changes that have been made to the original entry at that location.

Naïve Gauss Elimination

3. After the forward eliminations are complete, the unknown variable in the last modified equation can be solved for directly. In this case it is x_3.
4. One x_3 is solved, the remaining unknown in the previous equation is solved, i.e., $x_{2.}$
5. Then, x_1 can be calculated from these values of x_2 and x_3 from the first equation.
6. The same process applies to any system of n linear equations that one solves using NGE.
7. Unlike Cramer’s technique, this technique can be computerised as it is a recognisable, repeatable algorithm.
8. The pseudocode for this technique is given in the next slide:

Naïve Gauss Elimination: Pseudocode



```
(a)   DOFOR $k = 1, n - 1$
        DOFOR $i = k + 1, n$
          $factor = a_{i,k} / a_{k,k}$
          DOFOR $j = k + 1$ to $n$
            $a_{i,j} = a_{i,j} - factor \cdot a_{k,j}$
          END DO
          $b_i = b_i - factor \cdot b_k$
        END DO
      END DO
(b)   $x_n = b_n / a_{n,n}$
      DOFOR $i = n - 1, 1, -1$
        $sum = b_i$
        DOFOR $j = i + 1, n$
          $sum = sum - a_{i,j} \cdot x_j$
        END DO
        $x_i = sum / a_{i,i}$
      END DO
```

Pseudocode for Naïve Gauss Elimination

Naïve Gauss Elimination: Disadvantages.

1. Although NGE works on solvable systems, it suffers from some disadvantages in this form as follows:

- **It requires a high number of floating point operations** or ‘flops’. Thus, it is computationally expensive.
- **NGE and all elimination techniques will crash as they will not be able to handle divisions by zero** (i.e., where certain equations have no terms in a given variable, $x_{i.}$
- **NGE can accumulate inaccuracies due to round-off error**: every decimal operation, no matter to how many decimal places, incurs some intrinsic relative error, and these accumulate to high total error over multiple calculations.
- **Ill-conditioned systems**, where small changes in coefficients induce large changes in solution **are not suited to solution using NGE** and elimination methods **because of round-off error**.

Computational Methods and Modelling

Lecture 7: Dr. Edward McCarthy

Topic 1: Solution of Simultaneous Equations using Matrix Methods for Computers

THE UNIVERSITY *of* EDINBURGH
School of Engineering

Ill-conditioned systems and their pitfalls.

1. To illustrate the dangers of using NGE with an ill-conditioned system, study the following example:

$$x_1 + 2x_2 = 10$$

$$1.1x_1 + 2x_2 = 10.4$$

2. Compute the two apparent solutions using the two elimination formulae we discussed earlier:

$$x_2 = \frac{a_{11}b_2 - a_{21}b_1}{a_{11}a_{22} - a_{12}a_{21}} \qquad x_2 = \frac{1(10.4) - 1.1(10)}{1(2) - 2(1.1)} = 3$$

$$x_1 = \frac{a_{22}b_1 - a_{12}b_2}{a_{11}a_{22} - a_{12}a_{21}} \qquad x_1 = \frac{2(10) - 2(10.4)}{1(2) - 2(1.1)} = 4$$

3. Now introduce a slight change to a coefficient, say a_{21}, and the apparent values of x_1, x_2 are dramatically different!

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

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

Ill-conditioned systems and their pitfalls.

4. An error check with these erroneous values will not necessarily alert one to the fact that they are the incorrect solutions.

5. Another factor to consider is that the determinant of a system can scale dramatically if the system is multiplied across by a common factor.

6. This also applies to an ill-conditioned system, so that now, a determinant may appear to have a large value, but the system will be no more well-conditioned than before.

7. To combat this, one solution is to scale the equation system so that the maximum coefficient of any element in any row is 1.

8. This has the effect of producing a determinant value that more accurately reflects the proximity of the system to singularity.

9. Other techniques for improving solutions to systems of linear equations include:

Techniques for improving Solutions to LESs

1. There are a number of techniques used to mitigate the issues discussed above:

   a. **Use of more significant figures** in computations to reduce round-off error.

   b. **Use of partial or full pivoting:** This is the practice of a) identifying zero coefficients, or near-zero coefficients of elements in a given row, and switching that row with one containing much larger coefficients in the same position. (Exchanging both rows and columns is full pivoting, though full pivoting is generally avoided as it complicates the computation).

2. An example of the critical benefit of partial pivoting is given as follows:

$$0.0003x_1 + 3.0000x_2 = 2.0001$$
$$1.0000x_1 + 1.0000x_2 = 1.0000$$

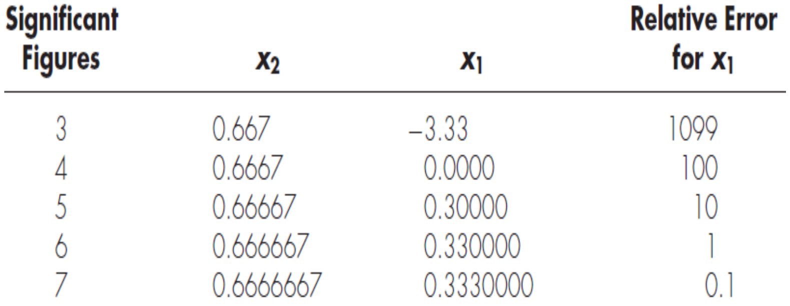
Techniques for improving Solutions to LESs

2. The coefficient of the leading element, x_1, is clearly very low, and one suspects that the problem as currently posed is ill conditioned.

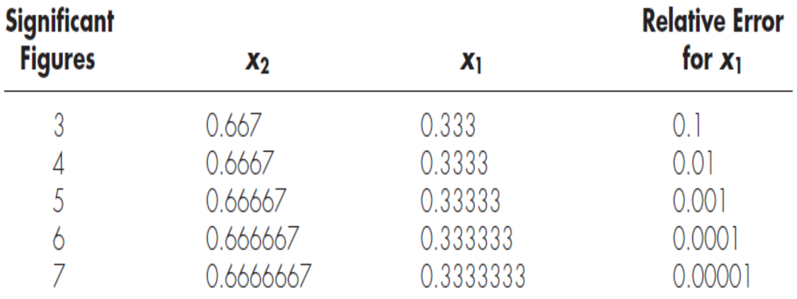
3. Partial pivoting of the solution identifies a larger coefficient in the second equation (i.e., 1) that would not lead to numerical complications caused by near-zero denominators. The new system is:

$$\begin{aligned} 1.0000x_1 + 1.0000x_2 &= 1.0000 \\ 0.0003x_1 + 3.0000x_2 &= 2.0001 \end{aligned}$$

4. Implementing a full solution of both systems of equations delivers the same solutions, but the error introduced by using different significant figures is much lower.



Significant Figures	x_2	x_1	Relative Error for x_1
3	0.667	−3.33	1099
4	0.6667	0.0000	100
5	0.66667	0.30000	10
6	0.666667	0.330000	1
7	0.6666667	0.3330000	0.1



Significant Figures	x_2	x_1	Relative Error for x_1
3	0.667	0.333	0.1
4	0.6667	0.3333	0.01
5	0.66667	0.33333	0.001
6	0.666667	0.333333	0.0001
7	0.6666667	0.3333333	0.00001

Techniques for improving Solutions to LESs

1. A pseudocode to implement partial pivoting is given as follows:

```
$p = k$
$big = |a_{k,k}|$
DOFOR $ii = k+1, n$
    $dummy = |a_{ii,k}|$
    IF $(dummy > big)$
        $big = dummy$
        $p = ii$
    END IF
END DO
IF $(p \neq k)$
    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
```

p = row number of pivot element

big = current pivot element

Define absolute value (modulus) of the pivot element a_{kk}

Define dummy variable for modulus of $a_{ii,k}$

Test whether dummy is greater than the pivot element $a_{k,k}$

If yes, assign dummy as new pivot element

First loop checks if current pivot element big is still bigger than the elements in rows below it.

Second loop switches row with largest pivot element with the one previously largest at the start of the first loop.

Python Code for Gauss Elimination

A Python Code for Gauss Elimination without partial pivoting is as follows

```
import numpy as np

def linearsolver(A,b):
    n = len(A)

#Initialise solution vector as an empty array
    x = np.zeros(n)

#Join A and use concatenate to form an augmented
coefficient matrix
    M = np.concatenate((A,b.T), axis=1)

    for k in range(n):
        for i in range(k,n):
            if abs(M[i][k]) > abs(M[k][k]):
                M[[k,i]] = M[[i,k]]
            else:
                pass
                for j in range(k+1,n):
                    q = M[j][k] / M[k][k]
                    for m in range(n+1):
                        M[j][m] +=  -q * M[k][m]
```

```
#Python starts indexing with 0, so the
last element is n-1
    x[n-1] =M[n-1][n]/M[n-1][n-1]


#We need to start at n-2, because of
Python indexing
    for i in range (n-2,-1,-1):
        z = M[i][n]
        for j in range(i+1,n):
            z = z  - M[i][j]*x[j]
        x[i] = z/M[i][i]

    return x

#Initialise the matrices to be solved.
A=np.array([[10., 15., 25],[4., 5., 6],
[25, 3, 8]])
b=np.array([[34., 25., 15]])
print(linearsolver(A,b))
```

Gauss Elimination with Partial Pivoting

A Python Code for Gauss Elimination with partial pivoting is as follows (A. Fragkou)

```
def column(m, c):
  return [m[i][c] for i in range(len(m))]

def row(m, r):
  return m[r][:]

def height(m):
  return len(m)

def width(m):
  return len(m[0])

def print_matrix(m):
  for i in range(len(m)):
    print(m[i])

def gaussian_elimination_with_pivot(m):

# forward elimination
  n = height(m)
  for i in range(n):
    pivot(m, n, i)
    for j in range(i+1, n):
      m[j] = [m[j][k] - m[i][k]*m[j][i]/m[i][i] for
k in range(n+1)]
```

```
if m[n-1][n-1] == 0:
raise ValueError('No unique solution')

  # backward substitution
  x = [0] * n
  for i in range(n-1, -1, -1):
    s = sum(m[i][j] * x[j] for j in
range(i, n))
    x[i] = (m[i][n] - s) / m[i][i]
  return x

def pivot(m, n, i):
    max = -1e100
    for r in range(i, n):
        if max < abs(m[r][i]):
            max_row = r
            max = abs(m[r][i])
    m[i], m[max_row] = m[max_row], m[i]

if __name__ == '__main__':
  #m = [[0,-2,6,-10], [-1,3,-6,5], [4,-
12,8,12]]
  #m = [[1,-1,3,2], [3,-3,1,-1], [1,1,0,3]]
  m = [[1,-1,3,2], [6,-6,2,-2], [1,1,0,3]]
  print(gaussian_elimination_with_pivot(m))
```