Indian Institute of Technology Roorkee

MAB-103: Numerical Methods

Unit-II Solutions of system of linear equations Session-2025-26

Consider the simplest example

$$ax = b$$

where a and b are given real numbers, with $a \neq 0$, whose solution is

$$x = a^{-1}b$$

Next, we shall consider a different generalisation of this elementary problem:

Let A be an $n \times n$ matrix with a_{ij} as its entry in row i and column j and b a given column vector of size n with jth entry b_j; find a column vector x of size n such that $Ax = b$.

Denoting by x_i the ith entry of the vector x, we can also write $Ax = b$ in the following expanded form:

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

Recall that in order to ensure that for real numbers a and b the single linear equation $ax = b$ has a unique solution, we need to assume that $a \neq 0$. In the case of the above system of n linear equations in n unknowns we shall have to make an analogous assumption on the matrix A.

0.1 Gaussian elimination and backward substitution

We showcase the following in this section:

- How to solve linear equations when A is in upper triangular form. The algorithm is called backward substitution.
- How to transform a general system of linear equations into an upper triangular form, to which backward substitution can be applied. The algorithm is called Gaussian elimination.

0.1.1 Backward substitution

In specific instances, the matrix A exhibits structural properties that render the solution computationally efficient. For example, when A is diagonal, defined as diagonal, written as

$$A = \begin{pmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{pmatrix}$$

(by this notation we mean that the off-diagonal elements are all 0), then the linear equations are uncoupled, reading $a_{ii}x_i = b_i$, and the solution is obviously

$$x_i = \frac{b_i}{a_{ii}}, \quad i = 1, 2, \cdots, n$$

A more complex example of a special structure is an upper triangular matrix

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ & a_{22} & \ddots & \vdots \\ & & \ddots & \vdots \\ & & & a_{nn} \end{pmatrix}$$

where all elements above the main diagonal are zero: $a_{ij} = 0$ for all $i > j$. In this case each equation is possibly coupled only to those following it but not to those preceding it. Thus, we can solve an upper triangular system of equations backwards.

- **Algorithm: Backward Substitution.**
  Given an upper triangular matrix A and a right-hand-side $\mathbf{b}$,

$$\begin{aligned} &for\ k = n : -1 : 1 \\ &\quad x_k = \frac{b_k - \sum_{j=k+1}^{n} a_{kj} x_j}{a_{kk}} \\ &end \end{aligned}$$

- **Cost of backward substitution**
  What is the cost of this algorithm? In a simplistic way we just count each floating point operation (such as + and $*$) as a flop. The number of flops required here is

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

0.1.2 Forward substitution

A more complex example of a special structure is an upper triangular matrix

$$A = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

where all elements above the main diagonal are zero: $a_{ij} = 0$ for all $i < j$. The situation is similar to the upper triangular case, except that we apply forward substitution to compute a solution. The algorithm is given on this page.

- **Algorithm: Forward Substitution.**
  Given an upper triangular matrix A and a right-hand-side $\mathbf{b}$,

$$\begin{aligned} &for\ k = 1 : n \\ &\quad x_k = \frac{b_k - \sum_{j=1}^{k-1} a_{kj} x_j}{a_{kk}} \\ &end \end{aligned}$$

0.1.3 Gaussian elimination

We use elementary row transformations to reduce the given linear system to an equivalent upper triangular form. Then we solve the resulting system using backward substitution. The reason this works is that the solution to our problem $Ax = b$ is invariant to

- multiplication of a row by a constant
- subtraction of a multiple of one row from another, and
- row interchanges.

These claims can be easily verified directly. Simply recall that $Ax = b$ is a concise form for the system of equations

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

and check what each of these operations amounts to.
Performing each of these operations can be recorded directly on the augmented matrix

$$(A|b) = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & | & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & | & b_2 \\ \vdots & \vdots & \ddots & \vdots & | & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & | & b_n \end{pmatrix}$$

Eliminating one column at a time

- Eliminate the first column elements below the main diagonal:
  - Subtract $\frac{a_{21}}{a_{11}}\times$ the first row from the second row.
  - Subtract $\frac{a_{31}}{a_{11}}\times$ the first row from the third row.

    $\vdots$
  - Subtract $\frac{a_{n1}}{a_{11}}\times$ the first row from the nth row

  This produces

$$(A^{(1)}|b^{(1)}) = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & | & b_1 \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & | & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & | & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & | & b_n^{(1)} \end{pmatrix}$$

- Next, consider the $(n-1)\times n$ submatrix of $(A^{(1)}|b^{(1)})$ obtained by ignoring the first row and column. These are the elements that have been modified by the first stage of the process, described above, but not set to 0. Apply exactly the same step as before to this submatrix;

i.e., subtract $\frac{a_{32}}{a_{22}^{(1)}}\times$ the second row from the third row, etc.
After this second stage we have an augmented matrix of the form

$$(A^{(2)}|b^{(2)}) = \left(\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} & b_3^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{array}\right)$$

The superscripts in the above expression show for each element at which stage it has last been modified.

- Repeat the process. After $n-1$ such stages we obtain an upper triangular system

$$(A^{(n-1)}|b^{(n-1)}) = \left(\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} & b_3^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n-1)} & b_n^{(n-1)} \end{array}\right)$$

This procedure leads to the Gaussian elimination algorithm in its simplest form given on the next page. In stating the Gaussian elimination algorithm we have not bothered to zero out the elements below the main diagonal in the modified matrix A: this part of the matrix is not considered in the ensuing backward substitution. Of course, for this algorithm to work we need to assume that all those pivotal elements $a_{kk}^{(k-1)}$ by which we divide are nonzero and in fact not very close to 0 either. We deal with the question of ensuring this later

  - **Algorithm: Gaussian Elimination.**
    Given a real, nonsingular $n \times n$ matrix A and a vector b of size n, first transform into upper triangular form,

$$\begin{aligned} &for\ k = 1 : n-1 \\ &\quad for\ i = k+1 : n \\ &\qquad l_{ik} = \frac{a_{ik}}{a_{kk}} \\ &\quad for\ i = k+1 : n \\ &\qquad a_{ij} = a_{ij} - l_{ik}a_{kj} \\ &\quad end \\ &\qquad b_i = b_i - l_{ik}b_k \\ &\quad end \\ &end \end{aligned}$$

- The cost of the Gaussian elimination algorithm in terms of operation count is approximately

$$\sum_{k=1}^{n-1}((n-k) + 2(n-k)(n-k+1)) \approx 2\sum_{k=1}^{n-1}(n-k)^2 = \frac{2}{3}n^3 + O(n^2)$$

flops.

0.2 LU Decomposition

In this section we show that the process of Gaussian elimination in fact decomposes the matrix A into a product $L \times U$ of a lower triangular matrix L and an upper triangular matrix U . Let us continue to assume that the elements $a_{kk}^{(k-1)}$ encountered in the Gaussian elimination process are all bounded safely away from 0. We will remove this assumption in the next section.

- **Elementary lower triangular matrices:**
  Consider the first stage of Gaussian elimination described above. These are elementary row opera- tions applied to the matrix A to zero out its first column below the $(1,1)$ entry. These operations are also applied to the right-hand-side vector b. Note, however, that the operations on b can actually be done at a later time because they do not affect those done on A. We can capture the effect of this first stage by defining the elementary $n \times n$ lower triangular matrix

$$M^{(1)} = \begin{pmatrix} 1 & & & & \\ -l_{21} & 1 & & & \\ -l_{31} & & 1 & & \\ \vdots & & & \ddots & \\ -l_{n1} & & & & 1 \end{pmatrix}$$

  Then the effect of this first stage is seen to produce

$$A^{(1)} = M^{(1)}A \quad \text{and} \quad b^{(1)} = M^{(1)}b$$

  Likewise, the effect of the second stage of Gaussian elimination, which is to zero out the second column of $A^{(1)}$ below the main diagonal, can be written as

$$A^{(2)} = M^{(2)}A^{(1)} = M^{(2)}M^{(1)}A \quad \text{and} \quad b^{(2)} = M^{(2)}b^{(1)} = M^{(2)}M^{(1)}b$$

  where

$$M^{(2)} = \begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ & -l_{32} & 1 & & & \\ & -l_{42} & & 1 & & \\ & \vdots & & & \ddots & \\ & -l_{n2} & & & & 1 \end{pmatrix}$$

- **Obtaining the LU decomposition :**
  This procedure is continued, and after $n-1$ such stages it transforms the matrix A into an upper triangular matrix, which we call U , by

$$U = A^{(n-1)} = M^{(n-1)} \cdots M^{(2)}M^{(1)}A.$$

  Likewise,

$$b^{(n-1)} = M^{(n-1)} \cdots M^{(2)}M^{(1)}b.$$

  Multiplying U by $[M^{(n-1)}]^{-1}$, then by $[M^{(n-2)}]^{-1}$, etc., we obtain

$$A = LU$$

where

$$L = [M^{(1)}]^{-1} \cdots [M^{(n-2)}]^{-1}[M^{(n-1)}]^{-1}$$

.

- Let us verify the above claims for the matrix

$$A = \begin{pmatrix} 1 & -1 & 3 \\ 1 & 1 & 0 \\ 3 & -2 & 1 \end{pmatrix}$$

  - The Gaussian elimination process for the first column yields $l_{21} = 1/1 = 1$ and $l_{31} = 3/1 = 3$, so

$$M^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix}, \quad A^{(1)} = M^{(1)}A = \begin{pmatrix} 1 & -1 & 3 \\ 0 & 2 & -3 \\ 0 & 1 & -8 \end{pmatrix}$$

  - The Gaussian elimination process for the second column yields $l_{32} = 1/2$, so

$$M^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1/2 & 1 \end{pmatrix}, \quad U = A^{(2)} = M^{(2)}A^{(1)} = \begin{pmatrix} 1 & -1 & 3 \\ 0 & 2 & -3 \\ 0 & 0 & -6.5 \end{pmatrix}$$

  - Note that

$$[M^{(1)}]^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix}, \quad [M^{(2)}]^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1/2 & 1 \end{pmatrix},$$

$$[M^{(1)}]^{-1}[M^{(2)}]^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & 1/2 & 1 \end{pmatrix} = L,$$

    and

$$LU = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & 1/2 & 1 \end{pmatrix} \begin{pmatrix} 1 & -1 & 3 \\ 0 & 2 & -3 \\ 0 & 0 & -6.5 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 3 \\ 1 & 1 & 0 \\ 3 & -2 & 1 \end{pmatrix} = A.$$

  - This explicitly specifies the LU decomposition of the given matrix A.

- Rather than constructing L and U by using the elimination steps, it is possible to solve directly for these matrices. Let us illustrate the direct computation of L and U in the case of $n = 3$. Write $A = LU$ as

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

  The above procedure of LU decompo- sition is called Doolittles method.

- We can also write $A = LU$ as

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} l11 & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{pmatrix}$$

  The above procedure of LU decomposition is called Crout's method.

- **Algorithm: Solving $Ax = b$ by LU Decomposition:**
  Given a real nonsingular matrix A, apply LU decomposition first:

$$A = LU$$

  Given also a right-hand-side vector b:

  - Forward substitution: solve

$$Ly = b$$

  - Backward substitution: solve

$$Ux = y$$

- **Symmetrizing the LU decomposition**
  Consider the LU decomposition of a symmetric positive definite matrix, written as

$$U = \begin{pmatrix} u_{11} & & & & \\ & u_{22} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & u_{nn} \end{pmatrix} \begin{pmatrix} 1 & \frac{u_{12}}{u_{11}} & \frac{u_{13}}{u_{11}} & \cdots & \frac{u_{1n}}{u_{11}} \\ & 1 & \frac{u_{23}}{u_{22}} & \cdots & \frac{u_{2n}}{u_{22}} \\ & & \ddots & & \vdots \\ & & & \ddots & \vdots \\ & & & & 1 \end{pmatrix} = D\tilde{U}$$

  so the LU decomposition reads

$$A = LD\tilde{U}.$$

  But transposing it we have

$$A = A^\top = \tilde{U}^\top D L^\top$$

  Since the decomposition is unique we must have $\tilde{U}^\top = L$, i.e., the decomposition reads

$$A = LDL^\top$$

  where L is unit lower triangular and D is diagonal with positive elements u_{kk}.
  It is also possible to write $D = D^{1/2}D^{1/2}$ with

$$D^{1/2} = diag(\sqrt{u_{11}}, \sqrt{u_{22}} \cdots, \sqrt{u_{nn}})$$

  whence the LU decomposition is written as

$$A = GG^\top$$

  with $G = LD^{1/2}$ a lower triangular matrix. This is called the Cholesky decomposition.

- Next we modify the algorithm of Gaussian elimination by pivoting, thus making it generally stable. The process of Gaussian elimination (or LU decomposition) described above cannot be applied unmodified to all nonsingular matrices.

  – For instance, the matrix
  $$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$
  is clearly nonsingular, and yet $a_{11} = 0$, so the algorithm breaks down immediately. Here is another example, slightly more subtle.

  – Consider the system
  $$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ x_1 + x_2 + 2x_3 &= 2 \\ x_1 + 2x_2 + 2x_3 &= 1. \end{aligned}$$
  The matrix A defined by these equations is nonsingular (indeed, the unique solution is $x = (1, -1, 1)^\top$), also all elements of A are nonzero, and yet Gaussian elimination yields after the first stage
  $$\left(\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 1 \end{array}\right) \Longrightarrow \left(\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right)$$
  Next, $a_{22}^{(1)} = 0$ and we cannot proceed further.
  In this case there is an obvious remedy: simply interchange the second and the third rows! This yields
  $$\left(\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 1 \\ 1 & 1 & 2 & 2 \end{array}\right) \Longrightarrow \left(\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right)$$
  and backward substitution proceeds as usual.

  – In the above examples the process broke down because a zero pivotal element $a_{kk}^{(k-1)}$ was encountered. But $a_{kk}^{(k-1)}$ does not have to exactly equal 0 for trouble to arise: undesirable accumulation of roundoff error may arise also when $a_{kk}^{(k-)}$ is near 0.

  – Consider a perturbation of the previous example that reads
  $$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ x_1 + 1.0001x_2 + 2x_3 &= 2 \\ x_1 + 2x_2 + 2x_3 &= 1. \end{aligned}$$
  The exact solution, correct to 5 digits, is $x \approx (1, -1.0001, 1.0001)^\top$. Now, Gaussian elimination in exact arithmetic can be completed and yields
  $$\left(\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 1 & 1.0001 & 2 & 2 \\ 1 & 2 & 2 & 1 \end{array}\right) \Longrightarrow \left(\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 0 & 0.0001 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right) \Longrightarrow \left(\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 0 & 0.0001 & 1 & 1 \\ 0 & 0 & -9999 & -10000 \end{array}\right)$$

Assume next that we are using floating point arithmetic with base $\beta = 10$ and precision $t = 3$. Then backward substitution gives

$$x_3 = 1,\ x_2 = 0,\ x_1 = 0.$$

On the other hand, if we interchange the second and third rows we obtain

$$\begin{pmatrix} 1 & 1 & 1 & | & 1 \\ 1 & 2 & 2 & | & 1 \\ 1 & 1.0001 & 2 & | & 2 \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 1 & 1 & | & 1 \\ 0 & 1 & 1 & | & 0 \\ 0 & 0.0001 & 1 & | & 1 \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 1 & 1 & | & 1 \\ 0 & 1 & 1 & | & 0 \\ 0 & 0 & .9999 & | & 1 \end{pmatrix}$$

Now backward substitution with 3 decimal digits gives

$$x_3 = 1.000,\ x_2 = -1.000,\ x_1 = 1.000$$

which is correct, in that the difference between this solution and the exact solution is less than the rounding unit.

- **Partial pivoting**
  The problem highlighted in these simple examples is real and general. Recall from last unit that roundoff errors may be unduly magnified if divided by a small value. Here, if a pivotal element $a_{kk}^{(k-1)}$ is small in magnitude, then roundoff errors are amplified, both in subsequent stages of the Gaussian elimination process and (even more apparently) in the backward substitution phase.

  These simple examples also suggest a strategy to resolve the difficulty, called partial pivoting: as the elimination proceeds for $k = 1, \cdots, n-1$, at each stage k choose $q = q(k)$ as the smallest integer for which

$$|a_{qk}^{(k-1)}| = \max_{k \le i \le n} |a_{ik}^{(k-1)}|,$$

  and interchange rows k and q. Then proceed with the elimination process.

- We could modify the partial pivoting strategy into one called scaled partial pivoting. Thus, define initially for each row i of A a size

$$s_i = \max_{1 \le j \le n} |a_{ij}|.$$

  Then, at each stage k of the Gaussian elimination procedure, choose $q = q(k)$ as the smallest integer

$$\frac{|a_{qk}^{(k-1)}|}{s_q} = \max_{k \le i \le n} \frac{|a_{ik}^{(k-1)}|}{s_i}$$

  and interchange rows k and q.

- A strategy that does guarantee stability is complete pivoting: at each stage choose q and r as the smallest integers for which

$$|a_{qr}^{(k-1)}| = \max_{k \le i,j \le n} |a_{ij}^{(k-1)}|,$$

  and interchange both row q and column r with row i and column j, respectively. However, this strategy is significantly more expensive than partial pivoting.

Solution Using Gaussian Elimination with Partial Pivoting

Given the system:

$$\begin{aligned} 3x_2 + 5x_3 &= 1.20736 \\ 3x_1 - 4x_2 &= -2.34066 \\ 5x_1 + 6x_3 &= -0.329193 \end{aligned}$$

Step 1: Augmented Matrix

$$\left(\begin{array}{ccc|c} 0 & 3 & 5 & 1.20736 \\ 3 & -4 & 0 & -2.34066 \\ 5 & 0 & 6 & -0.329193 \end{array}\right)$$

Step 2: Partial Pivoting (Swap Rows)

Swap row 1 and row 3:

$$\left(\begin{array}{ccc|c} 5 & 0 & 6 & -0.329193 \\ 3 & -4 & 0 & -2.34066 \\ 0 & 3 & 5 & 1.20736 \end{array}\right)$$

Step 3: Eliminate x_1 from Row 2

Multiplier: $m_{21} = \frac{3}{5} = 0.6$
Row 2 $\leftarrow$ Row 2 $- 0.6\times$ Row 1:

$$\left(\begin{array}{ccc|c} 5 & 0 & 6 & -0.329193 \\ 0 & -4 & -3.6 & -2.1431442 \\ 0 & 3 & 5 & 1.20736 \end{array}\right)$$

Step 4: Eliminate x_2 from Row 3

Multiplier: $m_{32} = \frac{3}{-4} = -0.75$
Row 3 $\leftarrow$ Row 3 $- (-0.75)\times$ Row 2:

$$\left(\begin{array}{ccc|c} 5 & 0 & 6 & -0.329193 \\ 0 & -4 & -3.6 & -2.1431442 \\ 0 & 0 & 2.3 & -0.39999 \end{array}\right)$$

Step 5: Back Substitution

$$\begin{aligned} x_3 &= \frac{-0.39999}{2.3} \approx -0.17391 \approx -0.174 \\ x_2 &= \frac{-2.14314 - (-3.6)(-0.174)}{-4} \approx 0.69238 \approx 0.692 \\ x_1 &= \frac{-0.329193 - 6(-0.174)}{5} \approx 0.14296 \approx 0.143 \end{aligned}$$

Solution of the Linear System Using Gaussian Elimination with Scaling

We solve the system:

$$\begin{cases} 3x + 2y + 100z = 105 \\ -x + 3y + 100z = 102 \\ x + 2y - z = 2 \end{cases}$$

Step 1: Augmented Matrix

$$\left(\begin{array}{ccc|c} 3 & 2 & 100 & 105 \\ -1 & 3 & 100 & 102 \\ 1 & 2 & -1 & 2 \end{array}\right)$$

Step 2: Scaling Factors

Compute the maximum absolute value in each row:

$$s_1 = \max(|3|, |2|, |100|) = 100$$
$$s_2 = \max(|-1|, |3|, |100|) = 100$$
$$s_3 = \max(|1|, |2|, |-1|) = 2$$

Step 3: Partial Pivoting with Scaling

For Column 1, compute scaled ratios:

$$\frac{|3|}{100} = 0.03, \quad \frac{|-1|}{100} = 0.01, \quad \frac{|1|}{2} = 0.5$$

Row 3 has largest ratio $\Rightarrow$ Swap Row 1 and Row 3

New matrix:

$$\left(\begin{array}{ccc|c} 0.03 & 0.02 & 1 & 1.05 \\ -0.01 & 0.03 & 1 & 1.02 \\ 0.5 & 1 & -0.5 & 1 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 0.5 & 1 & -0.5 & 1 \\ -0.01 & 0.03 & 1 & 1.02 \\ 0.03 & 0.02 & 1 & 1.05 \end{array}\right)$$

Step 4: Eliminate x from Rows 2 and 3

- Row 2 $\leftarrow$ Row 2 + 0.02 Row 1:

$$\left(\begin{array}{ccc|c} 0.5 & 1 & -0.5 & 1 \\ 0 & 0.05 & 0.99 & 1.04 \\ 0.03 & 0.02 & 1 & 1.05 \end{array}\right)$$

- Row 3 $\leftarrow$ Row 3 - 0.06 $\times$ Row 1:

$$\left(\begin{array}{ccc|c} 0.5 & 1 & -0.5 & 1 \\ 0 & 0.05 & 0.99 & 1.04 \\ 0 & -0.04 & 1.03 & .99 \end{array}\right)$$

Step 5: Partial Pivoting for Column 2

Scaled ratios for Column 2:

$$\frac{|0.05|}{0.99} = 0.05, \quad \frac{|-0.04|}{1.03} = 0.03$$

No row swap needed.

Step 6: Eliminate y from Row 3

Row 3 $\leftarrow$ Row 3 + $\frac{4}{5}$ $\times$ Row 2:

$$\left(\begin{array}{ccc|c} 0.5 & 1 & -0.5 & 1 \\ 0 & 0.05 & 0.99 & 1.04 \\ 0 & 0 & 1.822 & 1.822 \end{array}\right)$$

Step 7: Back Substitution

$$\begin{aligned} 1.822z = 1.822 &\Rightarrow z = 1 \\ 0.05y + 0.99(1) = 1.04 &\Rightarrow y = 1 \\ 0.5x + 1(1) - 1(1) = 2 &\Rightarrow x = 1 \end{aligned}$$

Solution Using Doolittle's LU Decomposition

Given the system:

$$\begin{cases} x + y + z = 1 \\ 4x + 3y - z = 6 \\ 3x + 5y + 3z = 4 \end{cases}$$

Step 1: Matrix Representation

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 4 & 3 & -1 \\ 3 & 5 & 3 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 6 \\ 4 \end{pmatrix}$$

Step 2: LU Decomposition

Find L and U such that $A = LU$ where:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

Step 3: Compute Elements

$$\begin{aligned} u_{11} = 1, \quad u_{12} = 1, \quad u_{13} = 1 \\ l_{21} = 4, \quad l_{31} = 3 \\ u_{22} = -1, \quad u_{23} = -5 \end{aligned}$$

$$l_{32} = -2$$
$$u_{33} = -10$$

Step 4: Forward Substitution ($Ly = b$)

$$\begin{cases} y_1 = 1 \\ y_2 = 2 \\ y_3 = 5 \end{cases}$$

Step 5: Backward Substitution ($Ux = y$)

$$\begin{cases} z = -0.5 \\ y = 0.5 \\ x = 1 \end{cases}$$

Final Solution

$$\boxed{x = 1}, \quad \boxed{y = 0.5}, \quad \boxed{z = -0.5}$$

Problem

Assume the LU decomposition of a square matrix $A \in \mathbb{R}^{500\times 500}$ takes 5 seconds. How many systems of the form

$$A\mathbf{x}_i = \mathbf{b}_i, \quad i = 1, 2, \ldots, k$$

can be solved in the next 6 seconds after the decomposition is complete?

Solution

Once the LU decomposition is computed, each system $A\mathbf{x} = \mathbf{b}$ can be solved by performing:

1. Forward substitution to solve $L\mathbf{y} = \mathbf{b}$
2. Backward substitution to solve $U\mathbf{x} = \mathbf{y}$

Each substitution step is an $\mathcal{O}(n^2)$ operation, so the total cost to solve one system is proportional to n^2. Let c denote the constant of proportionality. Then the LU decomposition time is approximately:

$$T_{\text{LU}} = \frac{2}{3}n^3 \cdot c.$$

Given that $T_{\text{LU}} = 5$ seconds for $n = 500$, we solve for c:

$$5 = \frac{2}{3} \cdot (500)^3 \cdot c \quad \Rightarrow \quad c = \frac{5 \cdot 3}{2 \cdot 500^3} = \frac{15}{250 \cdot 10^6} = 6 \times 10^{-8}.$$

Now, the time required to solve one linear system is:

$$T_{\text{solve}} = 2n^2 \cdot c = 2 \cdot (500)^2 \cdot 6 \times 10^{-8} = 2 \cdot 250{,}000 \cdot 6 \times 10^{-8} = 0.03 \text{ seconds}.$$

Thus, the number of systems that can be solved in 6 seconds is:

$$k = \frac{6}{0.03} = 200.$$

Conclusion

200 systems can be solved in 6 seconds after the LU decomposition is complete.

Conclusion

200 systems can be solved in 6 seconds after the LU decomposition is complete.

Given the system:

$$\begin{cases} 3x + 2y + 4z = 7 \\ 2x + y + z = 4 \\ 3x + 5y + 3z = 3 \end{cases}$$

Step 1: Matrix Representation

$$A = \begin{pmatrix} 3 & 2 & 4 \\ 2 & 1 & 1 \\ 3 & 5 & 3 \end{pmatrix}, \quad b = \begin{pmatrix} 7 \\ 4 \\ 3 \end{pmatrix}$$

Step 2: LU Decomposition

Find L and U such that $A = LU$ where:

$$L = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix}, \quad U = \begin{pmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{pmatrix}$$

Step 3: Compute Elements

$$\begin{aligned} l_{11} &= 3, \quad l_{21} = 2, \quad l_{31} = 3 \\ u_{12} &= \tfrac{2}{3} = .6666, \quad u_{13} = \tfrac{4}{3} = 1.3333 \\ l_{22} &= -\tfrac{1}{3} = 0.3333, \quad l_{32} = 3 \\ u_{23} &= 5 \\ l_{33} &= -16 \end{aligned}$$

Step 4: Forward Substitution ($Ly = b$)

$$\begin{cases} y_1 = \frac{7}{3} = 2.3333 \\ y_2 = 2 \\ y_3 = \frac{2}{3} \end{cases}$$

Step 5: Backward Substitution ($Ux = y$)

$$\begin{cases} z = \frac{2}{3} \\ y = -\frac{50}{3} \\ x = \frac{65}{3} \end{cases}$$

Final Solution

$$\boxed{x = \tfrac{65}{3}}, \quad \boxed{y = -\tfrac{50}{3}}, \quad \boxed{z = \tfrac{4}{3}}$$

Final Solution

$$\boxed{x = 1}, \quad \boxed{y = 1}, \quad \boxed{z = 1}$$

Step 1: Augmented Matrix

$$\left(\begin{array}{ccc|c} 3 & 2 & 100 & 105 \\ -1 & 3 & 100 & 102 \\ 1 & 2 & -1 & 2 \end{array}\right)$$

Step 2: Scaling Factors

Compute the maximum absolute value in each row:

$$s_1 = \max(|3|, |2|, |100|) = 100$$
$$s_2 = \max(|-1|, |3|, |100|) = 100$$
$$s_3 = \max(|1|, |2|, |-1|) = 2$$

Step 3: Partial Pivoting with Scaling

For Column 1, compute scaled ratios:

$$\frac{|3|}{100} = 0.03, \quad \frac{|-1|}{100} = 0.01, \quad \frac{|1|}{2} = 0.5$$

Row 3 has largest ratio $\Rightarrow$ Swap Row 1 and Row 3

New matrix:

$$\left(\begin{array}{ccc|c} 1 & 2 & -1 & 2 \\ -1 & 3 & 100 & 102 \\ 3 & 2 & 100 & 105 \end{array}\right)$$

Step 4: Eliminate x from Rows 2 and 3

- Row 2 $\leftarrow$ Row 2 + Row 1:

$$\left(\begin{array}{ccc|c} 1 & 2 & -1 & 2 \\ 0 & 5 & 99 & 104 \\ 3 & 2 & 100 & 105 \end{array}\right)$$

- Row 3 $\leftarrow$ Row 3 - 3 $\times$ Row 1:

$$\left(\begin{array}{ccc|c} 1 & 2 & -1 & 2 \\ 0 & 5 & 99 & 104 \\ 0 & -4 & 103 & 99 \end{array}\right)$$

Step 5: Partial Pivoting for Column 2

Scaled ratios for Column 2:

$$\frac{|5|}{5} = 1, \quad \frac{|-4|}{4} = 1$$

No row swap needed.

Step 6: Eliminate y from Row 3

Row 3 $\leftarrow$ Row 3 + $\frac{4}{5}$ $\times$ Row 2:

$$\left(\begin{array}{ccc|c} 1 & 2 & -1 & 2 \\ 0 & 5 & 99 & 104 \\ 0 & 0 & 182.2 & 182.2 \end{array}\right)$$

Step 7: Back Substitution

$$\begin{aligned} 182.2z = 182.2 &\Rightarrow z = 1 \\ 5y + 99(1) = 104 &\Rightarrow y = 1 \\ x + 2(1) - 1(1) = 2 &\Rightarrow x = 1 \end{aligned}$$

Verification

Substitute $(x, y, z) = (1, 1, 1)$ into original equations:

$$\begin{cases} 3(1) + 2(1) + 100(1) = 105 \\ -1(1) + 3(1) + 100(1) = 102 \\ 1(1) + 2(1) - 1(1) = 2 \end{cases}$$

All equations are satisfied.

Vector Norms

A vector norm is a function $\|\cdot\|$ from $\mathbb{R}^n$ to $\mathbb{R}$ satisfying:

1. Positive definiteness: $\|\mathbf{x}\| \geq 0, \quad \|\mathbf{x}\| = 0 \iff \mathbf{x} = \mathbf{0}$.
2. Homogeneity: $\|\alpha\mathbf{x}\| = |\alpha| \, \|\mathbf{x}\|$.
3. Triangle inequality: $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$.

Common vector norms:

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|, \quad \|\mathbf{x}\|_2 = \left(\sum_{i=1}^{n} |x_i|^2\right)^{1/2}, \quad \|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|.$$

Matrix Norms

A matrix norm $\|A\|$ satisfies:

1. Positive definiteness: $\|A\| \geq 0, \quad \|A\| = 0 \iff A = 0.$
2. Homogeneity: $\|\alpha A\| = |\alpha| \, \|A\|.$
3. Triangle inequality: $\|A + B\| \leq \|A\| + \|B\|.$
4. Submultiplicativity: $\|AB\| \leq \|A\| \, \|B\|.$

The induced (operator) norm based on a vector norm $\| \cdot \|_v$ is:

$$\|A\| = \max_{\mathbf{x} \neq 0} \frac{\|A\mathbf{x}\|_v}{\|\mathbf{x}\|_v}.$$

Common induced norms:

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^{n} |a_{ij}|, \quad \|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^{n} |a_{ij}|, \quad \|A\|_2 = \sqrt{\rho(A^T A)}.$$

Vector Norms Example

Let

$$\mathbf{x} = (3, -4, 12)^T.$$

$$\|\mathbf{x}\|_1 = |3| + |-4| + |12| = 19,$$

$$\|\mathbf{x}\|_2 = \sqrt{3^2 + (-4)^2 + 12^2} = \sqrt{169} = 13,$$

$$\|\mathbf{x}\|_\infty = \max(3, 4, 12) = 12.$$

Matrix Norms Example

Let

$$A = \begin{bmatrix} 1 & -2 & 3 \\ -4 & 5 & -6 \end{bmatrix}.$$

Column sums:

$$5, \quad 7, \quad 9 \quad \Rightarrow \quad \|A\|_1 = 9.$$

Row sums:

$$6, \quad 15 \quad \Rightarrow \quad \|A\|_\infty = 15.$$

Spectral norm:

$$\|A\|_2 \approx 9.508.$$

Condition Number of a Matrix

Let A be a nonsingular matrix. The *condition number* of A with respect to a matrix norm $\|\cdot\|$ is defined as

$$\kappa(A) = \|A\| \cdot \|A^{-1}\|.$$

In the 2-norm (spectral norm) this becomes

$$\kappa_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)},$$

where $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ are the largest and smallest singular values of A, respectively.

A matrix with $\kappa(A) \approx 1$ is called *well-conditioned*, while a matrix with a large condition number is called *ill-conditioned*. The condition number provides an upper bound on the relative error:

$$\frac{\|\delta \mathbf{x}\|}{\|\mathbf{x}\|} \lesssim \kappa(A) \cdot \frac{\|\delta \mathbf{b}\|}{\|\mathbf{b}\|}.$$

Example 1: Well-Conditioned Matrix

Let

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

We have

$$\|A\|_2 = \max(|2|, |1|) = 2, \quad A^{-1} = \begin{bmatrix} 1/2 & 0 \\ 0 & 1 \end{bmatrix}, \quad \|A^{-1}\|_2 = 1.$$

Thus,

$$\kappa_2(A) = 2 \times 1 = 2.$$

Example 2: Ill-Conditioned Matrix

Let

$$B = \begin{bmatrix} 1 & 0.99 \\ 0.99 & 0.98 \end{bmatrix}.$$

The determinant is

$$\det(B) = 1 \cdot 0.98 - 0.99 \cdot 0.99 = -0.0001,$$

which is very small, so B^{-1} will have large entries. Numerically,

$$\|B\|_2 \approx 1.99, \quad \|B^{-1}\|_2 \approx 2 \times 10^4,$$

so

$$\kappa_2(B) \approx 4 \times 10^4.$$

Example 3: Identity Matrix

For

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

we have

$$\|I\|_2 = 1, \quad \|I^{-1}\|_2 = 1, \quad \kappa_2(I) = 1.$$

Definition

A square matrix $A = [a_{ij}] \in \mathbb{R}^{n \times n}$ is called **strictly diagonally dominant** if:

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad \text{for all } i = 1, 2, \ldots, n.$$

That is, in each row of A, the magnitude of the diagonal entry is strictly greater than the sum of the magnitudes of the other (non-diagonal) entries.

Interpretation

Geometrically, strict diagonal dominance means that each equation in the linear system $A\mathbf{x} = \mathbf{b}$ has its main variable coefficient significantly larger (in magnitude) than the coefficients of other variables in that equation. This property often guarantees stability and convergence of iterative methods such as the Gauss–Seidel and Jacobi methods.

Example

Consider the matrix:

$$A = \begin{bmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 3 \end{bmatrix}.$$

Check strict diagonal dominance:

$$\begin{aligned} |4| &> |-1| + |0| = 1 && \text{(True)} \\ |4| &> |-1| + |-1| = 2 && \text{(True)} \\ |3| &> |0| + |-1| = 1 && \text{(True)} \end{aligned}$$

Since all three inequalities hold, A is strictly diagonally dominant.

Non-example

The matrix

$$B = \begin{bmatrix} 2 & 3 \\ 1 & 1 \end{bmatrix}$$

fails the first row test:

$$|2| \not> |3| \quad \Rightarrow \quad \text{Not strictly diagonally dominant.}$$

Applications

Strictly diagonally dominant matrices are important because:

- They guarantee nonsingularity (invertibility).
- They ensure convergence of iterative solvers.
- They often arise from discretizations of PDEs (e.g., Laplace equation).

Definition

A real symmetric matrix $A \in \mathbb{R}^{n \times n}$ is said to be **positive definite** if:

$$\mathbf{x}^T A \mathbf{x} > 0 \quad \forall\, \mathbf{x} \in \mathbb{R}^n,\ \mathbf{x} \neq \mathbf{0}.$$

Similarly, A is **positive semi-definite** if:

$$\mathbf{x}^T A \mathbf{x} \geq 0 \quad \forall\, \mathbf{x} \in \mathbb{R}^n.$$

Equivalent Conditions

For a real symmetric matrix A, the following are equivalent:

1. $\mathbf{x}^T A \mathbf{x} > 0$ for all nonzero $\mathbf{x}$.
2. All eigenvalues $\lambda_i > 0$.
3. All leading principal minors are positive (Sylvester's criterion): $\det(A_k) > 0$ for $k = 1, 2, \ldots, n$, where A_k is the top-left $k \times k$ submatrix of A.

Example

Consider the matrix:

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

Quadratic form check

Let $\mathbf{x} = (x_1, x_2)^T$. Then:

$$\mathbf{x}^T A \mathbf{x} = 2x_1^2 - 2x_1 x_2 + 2x_2^2 = (x_1 - x_2)^2 + x_1^2 + x_2^2 > 0$$

for all nonzero $\mathbf{x}$.

Eigenvalue check

We solve $\det(A - \lambda I) = 0$:

$$\det \begin{bmatrix} 2-\lambda & -1 \\ -1 & 2-\lambda \end{bmatrix} = (2-\lambda)^2 - 1 = 0,$$

which simplifies to:

$$\lambda^2 - 4\lambda + 3 = 0.$$

Thus:

$$\lambda = 1,\ 3 > 0.$$

All eigenvalues are positive.

Principal minors check

$$\det([2]) = 2 > 0,$$
$$\det(A) = (2)(2) - (-1)(-1) = 4 - 1 = 3 > 0.$$

Since all leading principal minors are positive, A is positive definite.

0.3 Iterative methods

Loosely speaking, discretization of one- and two-dimensional problems yields linear systems that are small enough to be efficiently solved with direct methods. However, three-dimensional problems usually leads to linear systems that are too large and expensive for direct methods. Instead, cheaper iterative methods must be utilized. Unlike direct methods, iterative methods do not have a fixed number of floating point operations attached to them for computing the solution x to a linear system. Instead, a sequence of approximations $x^{(k)}$ is sought successively, such that $x^{(k)} \to x$ in the limit $k \to \infty$. Of course, the unspoken hope is that convergence in one way or another will be reached with only a small number of iterations.

0.3.1 Basic Iterative methods

It is actually quite simple to create a framework for a set of basic iterative methods. To this end consider, again, the linear system $Ax = b$, and let us split A into

$$A = M - K$$

where M is any non-singular matrix, and K the remainder $K = M - A$. This gives

$$\begin{aligned} (M - K)x &= b \\ Mx &= Kx + b \\ x &= M^{-1}Kx + M^{-1}b \end{aligned}$$

Now, if we have a starting guess $x^{(0)}$ for x, this suggests the iteration

$$x^{(k+1)} = M^{-1}Kx^{(k)} + M^{-1}b.$$

- Although we do not know for which linear systems the iteration scheme (6.24) converges, if any, let us tacitly take it as our basic iterative method for solving linear systems.
- For this algorithm to be computationally practical, it is important that the splitting of A is chosen such that $M^{-1}K$ and $M^{-1}b$ are easy to calculate, or at least their action on any given vector. Recall that we do not want to compute any inverses.
- In the following we shall study splittings of A of the form

  $$A = D - L - U$$

  where D is the diagonal of A, and $-U$ and $-L$ the strictly upper and lower triangular part of A, respectively. This leads to two classical iterative methods, known as the Jacobi and the Gauss-Seidel methods.

0.3.2 The Jacobi Method

Jacobi iteration is defined by choosing $M = D$ and $K = L + U$, which gives the iteration scheme

$$x^{(k+1)} = D^{-1}(L + U)x^{(k)} + D^{-1}b.$$

We observe that D is easy to invert, since it is a diagonal matrix. For example, $n = 3$, we have

$$\begin{pmatrix} x_1^{k+1} \\ x_2^{k+1} \\ x_3^{k+1} \end{pmatrix} = \begin{pmatrix} a_{11}^{-1} & & \\ & a_{22}^{-1} & \\ & & a_{33}^{-1} \end{pmatrix} \begin{pmatrix} 0 & -a_{12} & -a_{13} \\ -a_{21} & 0 & -a_{23} \\ -a_{31} & -a_{32} & 0 \end{pmatrix} \begin{pmatrix} x_1^k \\ x_2^k \\ x_3^k \end{pmatrix} + \begin{pmatrix} a_{11}^{-1} & & \\ & a_{22}^{-1} & \\ & & a_{33}^{-1} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}.$$

Hence, it gives

$$\begin{aligned} x_1^{k+1} &= a_{11}^{-1}(-a_{12}x_2^k - a_{13}x_3^k + b_1), \\ x_2^{k+1} &= a_{22}^{-1}(-a_{21}x_1^k - a_{23}x_3^k + b_2), \\ x_3^{k+1} &= a_{33}^{-1}(-a_{31}x_1^k - a_{32}x_2^k + b_3). \end{aligned}$$

Let's solve the following system of linear equations using the Jacobi method:

$$\begin{aligned} 4x_1 - x_2 + 0x_3 &= 15, \\ -x_1 + 4x_2 - x_3 &= 10, \\ 0x_1 - x_2 + 4x_3 &= 10. \end{aligned}$$

This corresponds to the matrix form:

$$A\mathbf{x} = \mathbf{b},$$

where

$$A = \begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 15 \\ 10 \\ 10 \end{pmatrix}.$$

The Jacobi method updates each component of $\mathbf{x}$ iteratively as follows:

$$\begin{aligned} x_1^{(k+1)} &= \frac{1}{4}\left(15 + x_2^{(k)}\right), \\ x_2^{(k+1)} &= \frac{1}{4}\left(10 + x_1^{(k)} + x_3^{(k)}\right), \\ x_3^{(k+1)} &= \frac{1}{4}\left(10 + x_2^{(k)}\right). \end{aligned}$$

Given an initial guess $\mathbf{x}^{(0)} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^{\mathrm{T}}$, the iterations proceed until the solution converges.

0.3.3 The Gauss-Seidel Method

In the Gauss-Seidel method for $M = D - L$ and $K = U$, which gives the iteration scheme

$$x^{(k+1)} = (D - L)^{-1}Ux^{(k)} + (D - L)^{-1}b.$$

We observe that since $D - L$ has a lower triangular structure, the effect of $(D - L)^{-1}$ can be computed by forward elimination. For example, $n = 3$, we have

$$\begin{aligned} x_1^{k+1} &= a_{11}^{-1}(-a_{12}x_2^k - a_{13}x_3^k + b_1), \\ x_2^{k+1} &= a_{22}^{-1}(-a_{21}x_1^{k+1} - a_{23}x_3^k + b_2), \\ x_3^{k+1} &= a_{33}^{-1}(-a_{31}x_1^{k+1} - a_{32}x_2^{k+1} + b_3). \end{aligned}$$

Consider the linear system:

$$\begin{aligned} 4x_1 - x_2 + 0x_3 &= 15, \\ -1x_1 + 4x_2 - 1x_3 &= 10, \\ 0x_1 - 1x_2 + 4x_3 &= 10. \end{aligned}$$

This can be written in matrix form as:

$$\begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 15 \\ 10 \\ 10 \end{pmatrix}.$$

Using the Gauss-Seidel method, the iterative updates for each variable are:

$$\begin{aligned} x_1^{(k+1)} &= \frac{1}{4}\left(15 + x_2^{(k)}\right), \\ x_2^{(k+1)} &= \frac{1}{4}\left(10 + x_1^{(k+1)} + x_3^{(k)}\right), \\ x_3^{(k+1)} &= \frac{1}{4}\left(10 + x_2^{(k+1)}\right). \end{aligned}$$

Starting with an initial guess $\mathbf{x}^{(0)} = (0, 0, 0)^\top$, the iterations proceed as follows:

- Iteration 1:

$$\begin{aligned} x_1^{(1)} &= \frac{1}{4}(15 + 0) = 3.75, \\ x_2^{(1)} &= \frac{1}{4}(10 + 3.75 + 0) = 3.4375, \\ x_3^{(1)} &= \frac{1}{4}(10 + 3.4375) = 3.359375. \end{aligned}$$

- Iteration 2:

$$\begin{aligned} x_1^{(2)} &= \frac{1}{4}(15 + 3.4375) = 4.609375, \\ x_2^{(2)} &= \frac{1}{4}(10 + 4.609375 + 3.359375) = 4.4921875, \\ x_3^{(2)} &= \frac{1}{4}(10 + 4.4921875) = 3.623046875. \end{aligned}$$

- Continue iterating until convergence.

0.3.4 Convergence Analysis

We now return to the question of convergence of our basic iterative method. By inspection, we see that it can be rewritten as

$$x^{(k+1)} = Hx^{(k)} + c \tag{1}$$

where $H = M^{-1}K$ is the relation matrix, and $c = M^{-1}b$. Let $e^{(k)} = x - x^k$ be the error after k iterations. A relation between the errors in successive approximations can be be derived by subtracting $x = Hx + c$ from (1)

$$e^{(k+1)} = x^{k+1} - x = H(x^{(k)} - x) = \cdots = H^{k+1}(x^{(0)} - x) = H^{k+1}e^0$$

Here, for convergence, we require $H^{k+1}e \to 0$ as $k \to \infty$ for any e. It turns out that this requirement is equivalent to $\rho(H) < 1$, where $\rho(H) = \max_{1 \le k \le n} |\lambda_k(H)|$ is the so-called spectral radius of H, that is, the magnitude of the extremal eigenvalue of H.

- Both Jacobi and Gauss-Seidel methods converge if A is strictly diagonally dominant.
- The Gauss-Seidel method also converges if A is SPD.

Convergence of the Jacobi Method

Given the system

$$A\mathbf{x} = \mathbf{b}, \quad A \in \mathbb{R}^{n \times n},$$

write

$$A = D - L - U,$$

where D is diagonal, L is strictly lower triangular, and U is strictly upper triangular.

The Jacobi iteration is

$$\mathbf{x}^{(k+1)} = D^{-1}(L + U)\mathbf{x}^{(k)} + D^{-1}\mathbf{b},$$

with iteration matrix

$$H_J = D^{-1}(L + U).$$

Convergence Condition

The Jacobi method converges if and only if

$$\rho(H_J) < 1,$$

where $\rho(\cdot)$ denotes the spectral radius.

Sufficient Condition

If A is strictly diagonally dominant, i.e.,

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \quad \forall i,$$

then

$$\|H_J\|_\infty = \max_i \frac{\sum_{j \neq i} |a_{ij}|}{|a_{ii}|} < 1$$

and hence $\rho(H_J) < 1$, guaranteeing convergence.

Convergence of the Gauss-Seidel Method

For the **Gauss-Seidel method**, the iteration matrix is given by:

$$H_{GS} = (D - L)^{-1}U,$$

where $A = D - L - U$ is the decomposition of matrix A into its diagonal (D), strictly lower triangular (L), and strictly upper triangular (U) parts.

The eigenvalues λ of H_{GS} satisfy the characteristic equation:

$$\det(H_{GS} - \lambda I) = \det\left((D - L)^{-1}U - \lambda I\right) = 0.$$

Using properties of determinants, this can be rewritten as:

$$\det\left((D - L)^{-1}(U - \lambda(D - L))\right) = 0.$$

Since $\det((D - L)^{-1}) \neq 0$, this simplifies to:

$$\det(U - \lambda(D - L)) = 0.$$

Multiplying by (-1) and rearranging terms, we obtain:

$$\det(\lambda D - \lambda L - U) = 0.$$

Let us denote:

$$A_\lambda = \lambda D - \lambda L - U.$$

Key Observation: Diagonal Dominance

If $A = D - L - U$ is **strictly diagonally dominant**, then for any λ with $|\lambda| \geq 1$, the matrix A_λ remains strictly diagonally dominant.

Proof of Diagonal Dominance: For each row i, the diagonal entry of A_λ is λa_{ii}, and the off-diagonal entries are scaled by λ (for L) or remain unchanged (for U). Since $|\lambda| \geq 1$, the magnitude of the diagonal entry $|\lambda a_{ii}|$ grows at least as fast as the sum of the magnitudes of the off-diagonal entries. Thus:

$$|\lambda a_{ii}| > |\lambda| \sum_{j<i} |a_{ij}| + \sum_{j>i} |a_{ij}| \geq \sum_{j \neq i} |(A_\lambda)_{ij}|,$$

which ensures that A_λ is strictly diagonally dominant.

Implications for Eigenvalues

A strictly diagonally dominant matrix is **nonsingular** (its determinant cannot be zero). Therefore:

$$\det(A_\lambda) \neq 0 \quad \text{for} \quad |\lambda| \geq 1.$$

However, the eigenvalue equation requires $\det(A_\lambda) = 0$. This leads to a contradiction unless $|\lambda| < 1$.

Conclusion: All eigenvalues λ of H_{GS} must satisfy $|\lambda| < 1$, which guarantees the **convergence** of the Gauss-Seidel method for strictly diagonally dominant matrices.

Extension to the Jacobi Method

The proof for the **Jacobi method** follows a similar reasoning. The iteration matrix for Jacobi is:

$$H_J = D^{-1}(L + U).$$

The eigenvalue analysis leads to:

$$\det(\lambda D - L - U) = 0.$$

If A is strictly diagonally dominant, then for $|\lambda| \geq 1$, the matrix $\lambda D - L - U$ is also strictly diagonally dominant and thus nonsingular. Hence, $|\lambda| < 1$, proving convergence.

0.4 Successive over-relaxation (SOR) method

- The SOR method is an enhancement of the Gauss-Seidel method, introducing a relaxation factor to accelerate convergence.
- The SOR method builds upon the Gauss-Seidel method, which iteratively updates each component of the solution vector by solving the corresponding equation assuming all other components remain fixed.
- Relaxation Factor (ω): SOR introduces a relaxation factor ω(where $1 \leq \omega < 2$) to modify the update rule, allowing for over-relaxation and faster convergence: If $\omega = 1$, the method reduces to the Gauss-Seidel method. If $1 < \omega < 2$, the method is over-relaxed, often leading to faster convergence. If $\omega > 2$ or $\omega < 1$, the method may diverge.
- SOR Algorithm: Given a system $Ax = b$, where A is decomposed as $A = D - L - U$ (with D being the diagonal, L the lower triangular, and U the upper triangular parts of A), the SOR iteration is defined by:

  $$x^{k+1} = Hx^k + c,$$

  where

  $$H = (D - \omega L)^{-1}((1 - \omega)D + \omega U), \quad c = \omega(D - \omega L)^{-1}r^k.$$

  The alternative form is as follows:

  $$x_i^{k+1} = (1 - \omega)x_i^k + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}x_j^{k+1} - \sum_{j>i} a_{ij}x_j^k\right)$$

  for $i = 1, 2, \cdots, n$, where x_i^{k+1} is the updated value of the $i - th$ component of the solution at iteration $k + 1$.
- Consider the system of linear equations:

  $$\begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 15 \\ 10 \\ 10 \end{pmatrix}$$

  We apply the Successive Over-Relaxation (SOR) method with an initial guess $\mathbf{x}^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ and relaxation factor $\omega = 1.25$.

Iteration 1

$$x_1^{(1)} = \frac{1.25}{4}(15 - (-1)\cdot 0 - 0) = 4.6875$$

$$x_2^{(1)} = \frac{1.25}{4}(10 - (-1)\cdot 4.6875 - (-1)\cdot 0) = 4.78515625$$

$$x_3^{(1)} = \frac{1.25}{4}(10 - (-1)\cdot 4.78515625) = 4.9932861328125$$

Thus,

$$\mathbf{x}^{(1)} = \begin{pmatrix} 4.6875 \\ 4.78515625 \\ 4.9932861328125 \end{pmatrix}.$$

Iteration 2

$$x_1^{(2)} = (1 - 1.25)\cdot 4.6875 + \frac{1.25}{4}(15 - (-1)\cdot 4.78515625 - 0) = 4.95758056640625$$

$$x_2^{(2)} = (1-1.25)\cdot 4.78515625 + \frac{1.25}{4}(10-(-1)\cdot 4.95758056640625-(-1)\cdot 4.9932861328125) = 5.0005626678$$

$$x_3^{(2)} = (1-1.25)\cdot 4.9932861328125 + \frac{1.25}{4}(10-(-1)\cdot 5.00056266784668) = 5.006518870592117$$

Thus,

$$\mathbf{x}^{(2)} = \begin{pmatrix} 4.95758056640625 \\ 5.00056266784668 \\ 5.006518870592117 \end{pmatrix}.$$