CPNA Lecture 14 - Solutions to Linear Simultaneous Equations

Mridul Sankar Barik

Jadavpur University

2023

Linear Systems I

- A **linear equation** in variables $x_1, x_2, \dots, x_n$ is an equation of the form
$$a_1x_1 + a_2x_2 + \ldots + a_nx_n = b$$
where $a_1, a_2, \dots, a_n$ and b are constant real numbers. The constant a_i is called the coefficient of x_i; and b is called the constant term of the equation
- A **system of linear equations** (or **linear system**) is a finite collection of linear equations in same set of variables. For instance, a linear system of n equations in n variables $x_1, x_2, \dots, x_n$ can be written as

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

Linear Systems II

- The system of linear equations can be written in matrix form

$$AX = B \tag{2}$$

where,

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}, X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \text{and } B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

- A **solution** of a linear system is a tuple $(s_1, s_2, \dots, s_n)$ of numbers that makes each equation a true statement when the values $s_1, s_2, \dots, s_n$ are substituted for $x_1, x_2, \dots, x_n$ respectively
- The set of all solutions of a linear system is called the **solution set** of the system

Solution of Linear Systems - Direct Methods I

- Yield exact solution in a finite number of arithmetic operations in absence of round-off errors
- In practice, we have finite number significant digits, so direct methods cannot lead to exact solutions
- Errors sometimes may lead to poor or even useless solutions
- Examples: Naive Gauss Elimination, Gauss-Jordon Elimination

Naive Gaussian Elimination I

- Reduces the system of equations to an equivalent upper triangular system which is then solved by back substitution
- The **augmented matrix** of the general linear system of equation 1 is

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & & & & \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{array}\right] \tag{3}$$

- The **coefficient matrix** of equation 1 is

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \tag{4}$$

Naive Gaussian Elimination II

- Forward Elimination of Unknowns
  - Reduce the set of equations to an upper triangular system
  - Eliminate the first unknown, x_1, from the second through the n^{th} equations
    - Multiply first row by a_{21}/a_{11} and subtract it from second row
    - Multiply first row by a_{31}/a_{11} and subtract it from third row
    - ...
    - We get the following

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ & a'_{22} & \dots & a'_{2n} & b'_2 \\ & \vdots & & & \\ & a'_{n2} & \dots & a'_{nn} & b'_n \end{array}\right]$$

    - a_{11} is called the pivot element
  - Repeat the above to eliminate the second unknown x_2 from third row onwards

Naive Gaussian Elimination III

- After $n-1$ iterations we get to an upper triangular matrix

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ & a_{22}^1 & \dots & a_{2n}^1 & b_2^1 \\ & \vdots & & & \\ & & \dots & a_{nn}^{n-1} & b_n^{n-1} \end{array}\right] \tag{5}$$

- Back Substitution
  - Last row can be solved as $x_n = \frac{b_n^{n-1}}{a_{nn}^{n-1}}$
  - The result can be back-substituted into the $(n-1)^{th}$ row to solve for $x_{n-1} = (b_{n-1} - a_{n-1,n}^{n-2} x_n)/a_{n-1,n-1}^{n-2}$
  - $\dots$
  - $x_1 = (b_1 - \sum_{j=2}^{n} a_{1j} x_j)/a_{11}$
  - General formula for obtaining the x's

$$x_i = \frac{b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{i-1} x_j}{a_{ii}^{(i-1)}} \text{ for } i = n-1, n-2, \dots 1 \tag{6}$$

Naive Gaussian Elimination IV

- Drawbacks
  - Division by Zero
    - During both elimination and back-substitution phase division by zero may occur
    - Pivoting technique partially avoids these problem
  - Round-Off Errors
    - Occurs due to limited significant digits
  - Ill-Conditioned Systems
    - Small changes in coefficients result in large changes in the solution
    - Implication $\Rightarrow$ wide range of answers can approximately satisfy the equations
  - Singular Systems
    - Determinant of a singular system is zero
    - After elimination stage the algorithm must check whether a zero diagonal element is created; if so, abort

Example of Gaussian Elimination I

Use Gaussian Elimination to solve

$$\begin{aligned} 2x + y + z &= 10 \\ 3x + 2y + 3z &= 18 \\ x + 4y + 9z &= 16 \end{aligned}$$

The corresponding augmented matrix is

$$\left[\begin{array}{ccc|c} 2 & 1 & 1 & 10 \\ 3 & 2 & 3 & 18 \\ 1 & 4 & 9 & 16 \end{array}\right]$$

Eliminating first variable x from equation 2 and 3 by performing transformations $[R_2 - \frac{3}{2}R_1]$ and $[R_3 - \frac{1}{2}R_1]$

$$\left[\begin{array}{ccc|c} 2 & 1 & 1 & 10 \\ 0 & \frac{1}{2} & \frac{3}{2} & 3 \\ 0 & \frac{7}{2} & \frac{17}{2} & 11 \end{array}\right]$$

Example of Gaussian Elimination II

Eliminating second variable y from equation 3 by performing transformations $[R_3 - \frac{\frac{7}{2}}{\frac{1}{2}} R_2]$, we get the upper triangular form

$$\left[\begin{array}{ccc|c} 2 & 1 & 1 & 10 \\ 0 & \frac{1}{2} & \frac{3}{2} & 3 \\ 0 & 0 & -2 & -10 \end{array}\right]$$

By backward substitution we get $z = 5$, $y = -9$ and $x = 7$

Gaussian Elimination Algorithm I

Algorithm 1 Triangularization of n Equations in n Unknowns (Forward Elimination)

for $k = 1$ to $n - 1$ in steps of 1 **do**
    for $j = k + 1$ to n in steps of 1 **do**
        $u = a[j][k]/a[k][k]$
        for $i = k$ to $n + 1$ in steps of 1 **do**
            $a[j][i] = a[j][i] - u * a[k][i]$
        end for
    end for
end for

Gaussian Elimination Algorithm II

Algorithm 2 Backward Substitution

```
x[n] = a[n][n+1]/a[n][n]
for i = n-1 to 1 in steps of -1 do
    sum = 0
    for j = i+1 to n in steps of 1 do
        sum = sum + a[i][j] * x[j]
    end for
    x[i] = (a[i][n+1] - sum)/a[i][i]
end for
```

Gauss-Jordon Elimination I

- A variant of Gauss elimination
- When an unknown is eliminated, it is eliminated from all other equations rather than just the subsequent ones
- All rows are normalized by dividing them by their pivot elements
- The elimination step results in an identity matrix rather than a triangular matrix $\Rightarrow$ so, back substitution is not necessary
- All pitfalls and improvements in Gauss elimination also applies to the Gauss-Jordan method
- **Row Echelon Form**: A matrix A is said to be in row echelon form if the following conditions hold
  1. All of the rows containing nonzero entries sit above any rows whose entries are all zero
  2. The first nonzero entry of any row, called the leading entry of that row, is positioned to the right of the leading entry of the row above it

Gauss-Jordon Elimination II

- **Reduced Row Echelon Form**: A matrix A is said to be in reduced row echelon form if it is in row echelon form, and additionally it satisfies the following two properties:
  1. In any given nonzero row, the leading entry is equal to 1
  2. The leading entries are the only nonzero entries in their columns
- An augmented matrix in reduced row echelon form corresponds to a solution to the corresponding linear system

Gauss-Jordon Elimination III

Algorithm 3 Gauss-Jordon Method

```
for i = 1 to n in steps of 1 do
    j = i
    while a[i][i] == 0 & j ≤ n do
        Interchange i and (j + 1)^th row of matrix a
        j = j + 1
    end while
    f = a[i][i]
    for k = i to n + 1 in steps of 1 do
        a[i][k] = a[i][k]/f
    end for
    for k = 1 to n in steps of 1 do
        if k ≠ i then
            f = a[k][i]/a[i][i]
            for p = i to n + 1 in steps of 1 do
                a[k][p] = a[k][p] − f ∗ a[i][p]
            end for
        end if
    end for
end for
```

Example of Gauss-Jordon Elimination I

Use Gauss-Jordon Elimination to solve

$$\begin{aligned} x + y + z &= 5 \\ 2x + 3y + 5z &= 8 \\ 4x + 5z &= 2 \end{aligned}$$

The corresponding augmented matrix is

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 2 & 3 & 5 & 8 \\ 4 & 0 & 5 & 2 \end{array}\right]$$

Dividing R_1 by it's pivot element $a_{11} = 1$ or $[R_1 \leftarrow \frac{R_1}{1}]$

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 2 & 3 & 5 & 8 \\ 4 & 0 & 5 & 2 \end{array}\right]$$

Example of Gauss-Jordon Elimination II

Eliminating first variable x from equation 2 and 3 by performing transformations $[R_2 \leftarrow R_2 - 2R_1]$ and $[R_3 \leftarrow R_3 - 4R_1]$

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & -4 & 1 & -18 \end{array}\right]$$

Dividing R_2 by it's pivot element $a_{22} = 1$ or $[R_2 \leftarrow \frac{R_2}{1}]$

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & -4 & 1 & -18 \end{array}\right]$$

Eliminating second variable y from equation 1 and 3 by performing transformations $[R_1 \leftarrow R_1 - R_2]$ and $[R_3 \leftarrow R_3 - (-4)R_2]$

$$\left[\begin{array}{ccc|c} 1 & 0 & -2 & 7 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 13 & -26 \end{array}\right]$$

Example of Gauss-Jordon Elimination III

Dividing R_3 by it's pivot element $a_{33} = 13$ or $[R_3 \leftarrow \frac{R_3}{13}]$

$$\left[\begin{array}{ccc|c} 1 & 0 & -2 & 7 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 1 & -2 \end{array}\right]$$

Eliminating third variable z from equation 1 and 2 by performing transformations $[R_1 \leftarrow R_1 - (-2)R_3]$ and $[R_2 \leftarrow R_2 - 3R_2]$

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

Now, we directly get the solution as $x = 3$, $y = 4$ and $z = -2$

Matrix Inversion Using Gauss-Jordon Elimination I

- Let A be an invertible $n \times n$ matrix
- Suppose that a sequence of elementary row-operations reduces A to the identity matrix
- Then the same sequence of elementary row-operations when applied to the identity matrix yields A^{-1}
- Apply the Gauss-Jordan method to the matrix $[A \;\; I_n]$
- Suppose the row reduced echelon form of the matrix $[A \;\; I_n]$ is $[B \;\; C]$
- If $B = I_n$, then $A^{-1} = C$ or else A is not invertible

Solution of Linear Systems - Iterative Method I

- Iterative methods start with an approximation to the true solution and if convergent derive a sequence of closer approximations till the required accuracy is obtained
- Amount of computation is dependent on the accuracy required
- Let the system of linear equations be given by

$$\left\{\begin{array}{c} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{nn}x_n = b_n \end{array}\right. \tag{7}$$

- We assume the diagonal elements (a_{ii}) to be non zero
- If not, then the equations should be rearranged

Solution of Linear Systems - Iterative Method II

- We can rewrite the equations as

$$\left\{\begin{array}{c} x_1 = \frac{b_1}{a_{11}} - \frac{a_{12}}{a_{11}}x_2 - \frac{a_{13}}{a_{11}}x_3 - \dots - \frac{a_{1n}}{a_{11}}x_n \\ x_2 = \frac{b_2}{a_{22}} - \frac{a_{21}}{a_{22}}x_1 - \frac{a_{23}}{a_{22}}x_3 - \dots - \frac{a_{2n}}{a_{22}}x_n \\ \vdots \\ x_n = \frac{b_n}{a_{nn}} - \frac{a_{n1}}{a_{nn}}x_1 - \frac{a_{n2}}{a_{nn}}x_2 - \dots - \frac{a_{n,(n-1)}}{a_{nn}}x_{n-1} \end{array}\right. \tag{8}$$

- Suppose the vector $X = [x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \dots, x_n^{(1)}]$ be a first approximation to the unknowns $x_1, x_2, x_3, \dots, x_n$
- So, the second approximation is obtained as

$$\left\{\begin{array}{c} x_1^{(2)} = \frac{b_1}{a_{11}} - \frac{a_{12}}{a_{11}}x_2^{(1)} - \frac{a_{13}}{a_{11}}x_3^{(1)} - \dots - \frac{a_{1n}}{a_{11}}x_n^{(1)} \\ x_2^{(2)} = \frac{b_2}{a_{22}} - \frac{a_{21}}{a_{22}}x_1^{(1)} - \frac{a_{23}}{a_{22}}x_3^{(1)} - \dots - \frac{a_{2n}}{a_{22}}x_n^{(1)} \\ \vdots \\ x_n^{(2)} = \frac{b_n}{a_{nn}} - \frac{a_{n1}}{a_{nn}}x_1^{(1)} - \frac{a_{n2}}{a_{nn}}x_2^{(1)} - \dots - \frac{a_{n,(n-1)}}{a_{nn}}x_{n-1}^{(1)} \end{array}\right. \tag{9}$$

Solution of Linear Systems - Iterative Method III

- If we write equation 9 in the matrix form $X = BX + C$ then the iteration formula may be written as $X^{(r+1)} = BX^{(r)} + C$
- In actual computation, solution vector $X^{(r+1)}$ is obtained element wise

Jacobi's Method I

- The iterative formula for the computation of solution by Jacobi's method is

$$x_i^{(r+1)} = \left(- \sum_{j=1, j \neq i}^{n} a_{ij} x_j^{(r)} + b_i \right) / a_{ii} \text{ for } i = 1, 2, 3, \ldots, n \tag{10}$$

  provided $a_{ii} \neq 0$
- Also known as *method of simultaneous displacements*

Jacobi's Method II

Algorithm 4: Jacobi's Method

input $a \to$ augmented matrix of order $n \times (n+1)$, $e \to$ allowed relative error in the result, $maxit \to$ the maximum number of iterations

output $x \to$ solution vector

for $i = 1$ to n in steps of 1 **do**
    $x[i] = 0$
end for
for $iter = 1$ to $maxit$ in steps of 1 **do**
    $big = 0$
    for $i = 1$ to n in steps of 1 **do**
        $sum = 0$
        for $j = 1$ to n in steps of 1 **do**
            if $j \neq i$ **then**
                $sum = sum + a[i][j] * x[j]$
            end if
        end for
        $temp = (a[i][n+1] - sum)/a[i][i]$
        $relerror = \mid (x[i] - temp)/temp \mid$
        if $relerror > big$ **then**

Jacobi's Method III

```
            big = relerror
        end if
        x'[i] = temp
    end for
    for i = 1 to n in steps of 1 do
        x[i] = x'[i]
    end for
    if big ≤ e then
        Write "Converges to a solution"
        Stop
    end if
end for
Write "Does not converge in maxit number of iterations"
```

Jacobi's Method IV

- The Jacobi iterative method works fine with well-conditioned linear systems
- If the linear system is ill-conditioned, it is most probably that the Jacobi method will fail to converge
- The Jacobi method can generally be used for solving linear systems in which the coefficient matrix is **diagonally dominant**
  - For each row, the absolute value of the diagonal term is greater than the sum of absolute values of other terms

Gauss-Seidel Method I

- Improves Jacobi's method (faster convergence) by a simple modification
- Uses an improved component as soon as it is available
- Also known as *method of successive displacements*
- The iterative formula for the computation of solution by Gauss Seidel method is

$$x_i^{(r+1)} = \left(-\sum_{j=1}^{i-1} a_{ij} x_j^{(r+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(r)} + b_i \right) / a_{ii} \text{ for } i = 1, 2, 3, \ldots \tag{11}$$

provided $a_{ii} \neq 0$

Gauss-Seidel Method II

Algorithm 5: Gauss-Seidel Method

input $a \rightarrow$ augmented matrix of order $n \times (n+1)$, $e \rightarrow$ allowed relative error in the result, *maxit* $\rightarrow$ the maximum number of iterations

output $x \rightarrow$ solution vector

```
for i = 1 to n in steps of 1 do
    x[i] = 0
end for
for iter = 1 to maxit in steps of 1 do
    big = 0
    for i = 1 to n in steps of 1 do
        sum = 0
        for j = 1 to n in steps of 1 do
            if j ≠ i then
                sum = sum + a[i][j] * x[j]
            end if
        end for
        temp = (a[i][n + 1] − sum)/a[i][i]
        relerror = | (x[i] − temp)/temp |
        if relerror > big then
```

Gauss-Seidel Method III

```
            big = relerror
        end if
        x[i] = temp
    end for
    if big ≤ e then
        Write "Converges to a solution"
        Stop
    end if
end for
Write "Does not converge in maxit number of iterations"
```

Example of Iterative Method I

Use Jacobi's / Gauss-Seidel Method to solve

$$
\begin{aligned}
10x_1 - 2x_2 - x_3 - x_4 &= 3 \\
-2x_1 + 10x_2 - x_3 - x_4 &= 15 \\
-x_1 - x_2 + 10x_3 2x_4 &= 27 \\
-x_1 - x_2 - 2x_3 + 10x_4 &= -9
\end{aligned}
$$

We rewrite the equations as

$$
\begin{aligned}
x_1 &= 0.3 + 0.2x_2 + 0.1x_3 + 0.1x_4 \\
x_2 &= 1.5 + 0.2x_1 + 0.1x_3 + 0.1x_4 \\
x_3 &= 2.7 + 0.1x_1 + 0.1x_2 + 0.2x_4 \\
x_4 &= -0.9 + 0.1x_1 + 0.1x_2 + 0.2x_3
\end{aligned}
$$

Initial solution vector $x = [0, 0, 0, 0]$

Example of Iterative Method II

n	x_1	x_2	x_3	x_4
1	0.300000	1.500000	2.700000	-0.900000
2	0.780000	1.740000	2.700000	-0.180000
3	0.900000	1.908000	2.916000	-0.108000
4	0.962400	1.960800	2.959200	-0.036000
5	0.984480	1.984800	2.985120	-0.015840
6	0.993888	1.993824	2.993760	-0.006048
7	0.997536	1.997549	2.997562	-0.002477
8	0.999018	1.999016	2.999013	-0.000979
9	0.999607	1.999607	2.999608	-0.000394
10	0.999843	1.999843	2.999843	-0.000157
11	0.999937	1.999937	2.999937	-0.000063
12	0.999975	1.999975	2.999975	-0.000025
13	0.999990	1.999990	2.999990	-0.000010
14	0.999996	1.999996	2.999996	-0.000004
15	0.999998	1.999998	2.999998	-0.000002
16	0.999999	1.999999	2.999999	-0.000001
17	1.000000	2.000000	3.000000	-0.000000

Table 1: Jacobi's Method

Example of Iterative Method III

n	x_1	x_2	x_3	x_4
1	0.300000	1.560000	2.886000	-0.136800
2	0.886920	1.952304	2.956562	-0.024765
3	0.983641	1.989908	2.992402	-0.004165
4	0.996805	1.998185	2.998666	-0.000768
5	0.999427	1.999675	2.999757	-0.000138
6	0.999897	1.999941	2.999956	-0.000025
7	0.999981	1.999989	2.999992	-0.000005
8	0.999997	1.999998	2.999999	-0.000001
9	0.999999	2.000000	3.000000	-0.000000
10	1.000000	2.000000	3.000000	-0.000000

Table 2: Gauss-Seidel Method

Clearly, Gauss-Seidel method converges faster than Jacobi's method