Numerical Methods

Systems of linear equations

Contents

- Linear equations (first degree equation)
  - Algebraic equation in which each term is either a constant and (the first power of) a single variable. *Ex:* $ax + b = 0$
- Systems of Linear equations

  collection of two or more linear equations involving the same set of variables

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

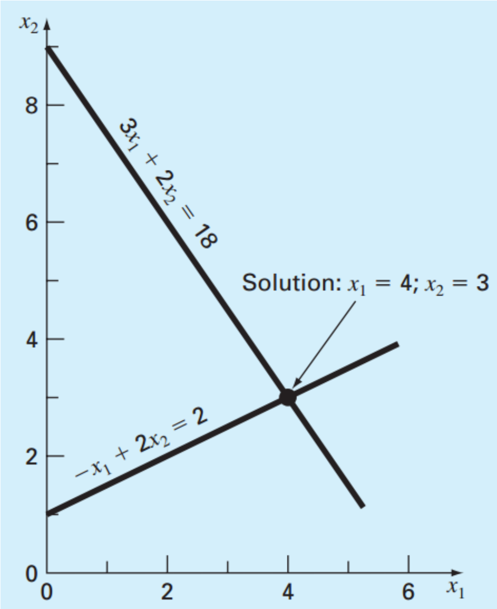
$n > 1$

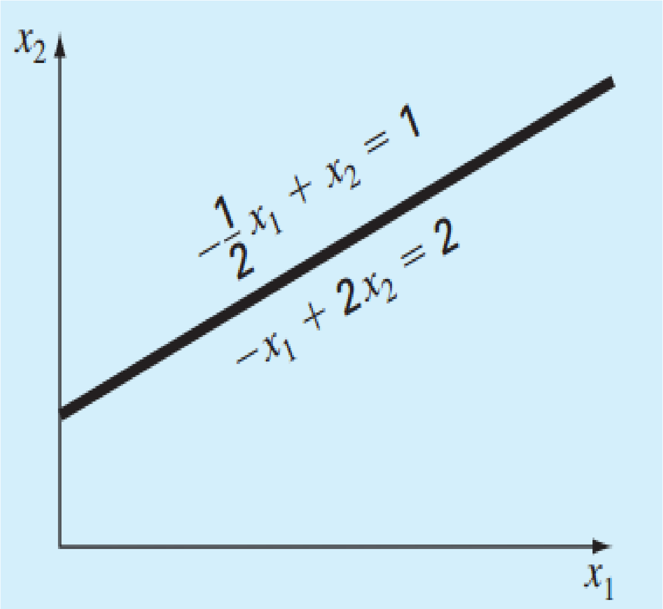
a's are constant coefficients, b's are constants, the x's are unknowns, and n is the number of equations

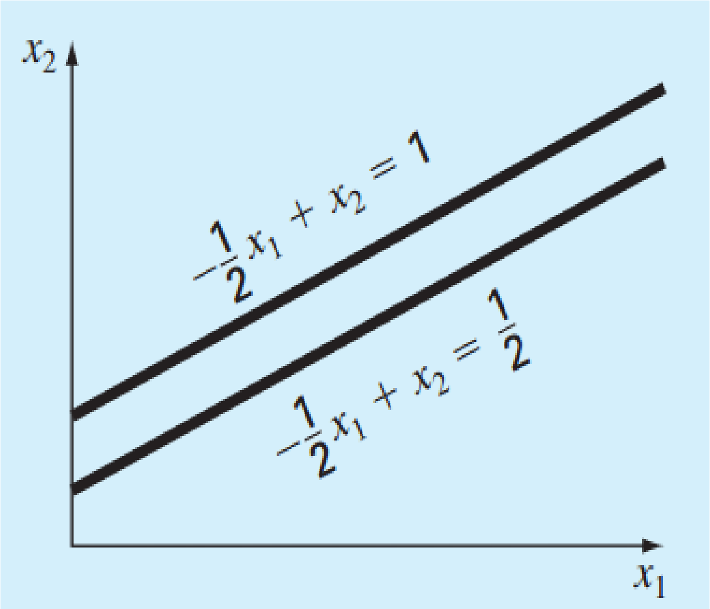
Systems of linear equations

- A linear system may behave in any one of three possible ways:
  - The system has a single unique solution
  - The system has infinitely many solutions
  - The system has no solution

HOW TO SOLVE ?







- Topics
  - Introduction of systems
  - Naïve Gauss Elimination
  - Jacobi method
  - Gauss-Seidel method

Objectives

- Understanding matrix notation, matrix multiplication, types of matrices
  identity, diagonal, symmetric, triangular
- Knowing how to represent a system of linear algebraic equations in matrix form
- Understanding how to use the Naïve Gauss elimination method

❖ Matrix Notation

- ➢ [A] is the shorthand notation for the matrix an *m* by *n* matrix
- ➢ a_{ij} designates an individual *element* of the matrix
- ➢ a_{11} , a_{22}, a_{33} : *the principal or main diagonal of the matrix*
- ➢ $a_{ij} = a_{ji}$: *symmetric matrix*
- ➢ $m = n$: *square matrices*

$$\begin{array}{ccc} & \begin{array}{ccccc} \phantom{a_{11}} & \phantom{a_{12}} & \text{Column 3} & & \phantom{a_{1n}} \\ & & \downarrow & & \end{array} & \\ [A] = & \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & \boxed{a_{23}} & \cdots & a_{2n} \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix} & \begin{array}{c} \\ \leftarrow \text{Row 2} \\ \\ \\ \\ \\ \end{array} \end{array}$$

Matrix Notation

- Diagonal matrix

$$[A] = \begin{bmatrix} a_{11} & & \\ & a_{22} & \\ & & a_{33} \end{bmatrix}$$

- Upper triangular matrix

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

- Lower triangular matrix

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

Matrix Notation

Identity matrix

$$[I] = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}$$

Banded matrix

$$[A] = \begin{bmatrix} a_{11} & a_{12} & & \\ a_{21} & a_{22} & a_{23} & \\ & a_{32} & a_{33} & a_{34} \\ & & a_{43} & a_{44} \end{bmatrix}$$

Matrix Notation

- Matrices with row dimension n = 1 => row vectors

$$[B] = [b_1 \quad b_2 \quad \cdots \quad b_m]$$

- Matrices with column dimension n = 1 => column vectors

$$[C] = \begin{bmatrix} c_1 \\ c_2 \\ \cdot \\ \cdot \\ \cdot \\ c_n \end{bmatrix}$$

Representing Linear Algebraic Equations in Matrix Form

- a 3 × 3 set of linear equations

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned} \Rightarrow x_1\begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \end{bmatrix} + x_2\begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \end{bmatrix} + x_3\begin{bmatrix} a_{13} \\ a_{23} \\ a_{33} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

- can be expressed as

$[A]\{x\} = \{b\}$ where $[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$ $\{b\} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$ $\{x\} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

$\Rightarrow$ $\{x\} = [A]^{-1}\{b\}$

- Gauss elimination (also known as row reduction)

The procedure consisted 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

Gauss elimination

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

$$\Downarrow$$

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

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

The two phases of Gauss elimination

The primes indicate the number of times that the coefficients and constants have been modified

Naïve Gauss elimination

1. Forward Elimination of Unknowns

a. Reduce the coefficient matrix [A] to an upper triangular system

b. Eliminate x_1 from the 2^{nd} to nth Eqns.

c. Eliminate x_2 from the 3^{rd} to nth Eqns.

d. Continue process until the nth equation has only 1 Non-Zero coefficient

2. Back-substituted

a. Starting from the last equation, => find x_n

b. Substitute to (n-1)th equation => find x_{n-1}

c. Each of the unknowns is found

Naïve Gauss elimination

The approach is designed to solve a general set of n equations

$$\boxed{a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1}$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2$$

$$\vdots \qquad \vdots$$

$$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n = b_n$$

$a_{11} = 0$ ⇒

- pivot equation
- a_{11} pivot coefficient

⇒ $a_{11} \neq 0$

Naïve 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

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1$$

Multiply by a_{21} / a_{11} to give $$a_{21}x_1 + \frac{a_{21}}{a_{11}}a_{12}x_2 + \frac{a_{21}}{a_{11}}a_{13}x_3 + \cdots + \frac{a_{21}}{a_{11}}a_{1n}x_n = \frac{a_{21}}{a_{11}}b_1$$

This equation can be subtracted from 2nd equation to give

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

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

❖ Forward Elimination of Unknowns

- The procedure is then repeated for the remaining equations
- 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:

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

⇨ can now be solved for x_n

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

❖ Back substitution

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

- This result can be back-substituted into the $(n - 1)$th equation to solve for x_{n-1}
- The procedure, which is repeated to evaluate the remaining x's, can be represented by the following formula:

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

For $i = n$-1, n-2 …1

Case study 1

Find and describe the set of solutions to the following system of linear equations

$$\begin{aligned} 2x + y - z &= 8 \\ -3x - y + 2z &= -11 \\ -2x + y + 2z &= -3 \end{aligned}$$

- The procedure consisted of steps:

Step 1: forward elimination

$$\begin{aligned} 2x + y - \quad z &= \quad 8 \\ -3x - y + 2z &= -11 \\ -2x + y + 2z &= \quad -3 \end{aligned} \Longrightarrow \left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right]$$

Step 2

$$\begin{aligned} L_2 + \frac{3}{2}L_1 &\to L_2 \\ L_3 + L_1 &\to L_3 \end{aligned} \Longrightarrow \left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ \boxed{0} & 1/2 & 1/2 & 1 \\ \boxed{0} & 2 & 1 & 5 \end{array}\right]$$

$$L_3 + -4L_2 \to L_3$$

- The procedure consisted of steps:

Step 3

$$L_3 + -4L_2 \to L_3 \quad \Longrightarrow \quad \left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ 0 & 1/2 & 1/2 & 1 \\ 0 & 0 & -1 & 1 \end{array}\right]$$

Step 4

Upper triangular form

$$\begin{aligned} 2x + \quad y - \quad z &= 8 \\ \frac{1}{2}y + \frac{1}{2}z &= 1 \\ \boxed{-z = 1} \end{aligned}$$

Back Substitution

- The procedure consisted of steps:

Step 5

$$-z = 1$$

Back Substitution

$$\begin{aligned} 2x + \ y \qquad &= 7 \\ \frac{1}{2}y \qquad &= \frac{3}{2} \\ -z &= 1 \end{aligned}$$

Step 6

$$\begin{aligned} 2x + y \quad &= 7 \\ \boxed{\begin{aligned} y \quad &= 3 \\ z &= -1 \end{aligned}} \end{aligned}$$

$$\begin{aligned} x \qquad &= 2 \\ y \quad &= 3 \\ z &= -1 \end{aligned}$$

Exercise

$$
\begin{aligned}
4x_1 + x_2 - x_3 &= 3 \\
2x_1 + 7x_2 + x_3 &= 19 \\
x_1 - 3x_2 + 12x_3 &= 31
\end{aligned}
$$

The exact solution is: $x_1 = 1$, $x_2 = 2$, $x_3 = 3$.

Matlab

MATLAB Matrix Manipulations

1. Create a 3 × 3 matrix
>> A = [1 5 6;7 4 2;-3 6 7]

A = 1 5 6
7 4 2
-3 6 7

2. The transpose of [*A*] can be obtained using the ‘ operator:
>> A'

ans = 1 7 -3
5 4 6
6 2 7

3. Next we will create another 3 × 3 matrix on a row basis. First create three row vectors:

```
>> x = [8 6 9];
>> y = [-5 8 1];
>> z = [4 8 2];
```

MATLAB Matrix Manipulations

4. Then we can combine these to form the matrix:
>> B = [x; y; z]

B = 8 6 9
-5 8 1
4 8 2

5. We can add [*A*] and [*B*] together:
>> C = A+B

C = 9 11 15
2 12 3
1 14 9

6. Further, we can subtract [*B*] from [*C*] to arrive back at [*A*]:
>> A = C-B

A = 1 5 6
7 4 2
-3 6 7

MATLAB Matrix Manipulations

A = 1 5 6
7 4 2
-3 6 7

B = 8 6 9
-5 8 1
4 8 2

7. Because their inner dimensions are equal, [A] and [B] can be multiplied

>> A*B

ans = 7 94 26
44 90 71
-26 86 -7

8. multiplied on an element-by-element

>> A.*B

ans = 8 30 54
-35 32 2
-12 48 14

MATLAB Matrix Manipulations

9. The matrix inverse can be computed with the inv function:
>> AI = inv(A)

```
AI = 0.2462  0.0154 -0.2154
    -0.8462  0.3846  0.6154
     0.8308 -0.3231 -0.4769
```

10. To test that this is the correct result, the inverse can be multiplied by the original matrix to give the identity matrix:
>> A*AI

```
ans = 1.0000 -0.0000 -0.0000
      0.0000  1.0000 -0.0000
      0.0000 -0.0000  1.0000
```

11. The eye function can be used to generate an identity matrix:
>> I = eye(3)

```
I =  1 0 0
     0 1 0
     0 0 1
```

SOLVING LINEAR ALGEBRAIC EQUATIONS WITH MATLAB

MATLAB provides two direct ways to solve systems of linear algebraic equations. The most efficient way is to employ the backslash, or “left-division,” operator as in

```
>> x = A\b
```

The second is to use matrix inversion:

```
>> x = inv(A)*b
```

MATLAB Naïve Gauss Elimination

(a) forward elimination

```
FOR k = 1, n - 1
    FOR i =k + 1, n
        factor = ai,k / ak,k
         FOR j = k + 1 to n
            ai,j = ai,j  - factor . ak,j
        END
        bi =bi  - factor . bk
    END
END
```

(b) Back substitution

```
xn  = bn / an,n
FOR i = n - 1, 1, -1
    sum = bi
    FOR j = i + 1, n
        sum = sum - ai,j . xj
    END
        xi    =  sum / ai,i
END
```

```
clc
clear all
close all
%%
A = [2 1 -1; -3 -1 2; -2 1 2]
B= [8 -11 -3 ]
%
x = naiv_gauss(A,B)
% 2nd method

y = A\B'

% 3rd method

z = inv(A)*B'
```

Numerical Methods

Systems of linear equations (continue)

Contents

Topics

- Introduction of systems
- Gauss Elimination
  - Naïve Gauss Elimination
  - Gauss Elimination: Pivoting
- Iterative Methods
  - Gauss-Seidel method
  - Jacobi method

Gauss Elimination: Pivoting

1. The primary reason that the foregoing technique is called "naïve" is that during both the elimination and the back-substitution phases, it is possible that a division by zero can occur. For example, if we use naïve Gauss elimination to solve

$$\begin{aligned} \boxed{2x_2 + 3x_3 = 8} \\ 4x_1 + 6x_2 + 7x_3 = -3 \\ 2x_1 - 3x_2 + 6x_3 = 5 \end{aligned}$$

⇒ - pivot equation
- pivot coefficient $a_{11}=0$

2. Problems may also arise when the pivot coefficient is close, rather than exactly equal, to zero because if the magnitude of the pivot element is small compared to the other elements, then round-off errors can be introduced

Basis idea

- determine the coefficient with the largest absolute value in the column below the pivot element
- The rows can then be switched so that the largest element is the pivot element. This is called partial pivoting.

$$\boxed{0.0003x_1 + 3.0000x_2 = 2.0001} \qquad\qquad \boxed{1.0000x_1 + 1.0000x_2 = 1.0000}$$
$$1.0000x_1 + 1.0000x_2 = 1.0000 \quad \Longrightarrow \quad 0.0003x_1 + 3.0000x_2 = 2.0001$$

- pivot equation
- pivot coefficient $a_{11}=0.0003 < 1.0000$ $\Longrightarrow$ Pivoting is required at the first column!

- 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 most of the improvement comes from partial pivoting

❖ Advantages

a. avoiding division by zero

b. minimizes round-off error

Case study 1

Let's consider the following linear system

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

Use Gauss elimination to solve, recall that the true solution is $x_1 = 1/3$, $x_2 = 2/3$

Note that in this form the first pivot element, $a_{11} = 0.0003$, is very close to zero

Case study 1

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

Naïve Gauss elimination

Multiplying the first equation by 1/(0.0003) yields

$$x_1 + 10.000x_2 = 6667 \Rightarrow \begin{aligned} x_1 + 10.000x_2 &= 6667 \\ x_1 + x_2 &= 1 \end{aligned}$$

which can be used to eliminate x_1 from the second equation:

$$9999x_2 = 6666$$

which can be solved for $x_2 = 2/3$

This result can be substituted back into the first equation to evaluate $x_1 = \dfrac{2.0001 - 3(2/3)}{0.0003}$

Case study 1

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

However, due to subtractive cancellation, the result is very sensitive to the number of significant figures carried in the computation:

Significant Figures	x_2	x_1	Absolute Value of Percent 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

⇨ x_1 is highly dependent on the number of significant figures

Case study 1

$$0.0003x_1 + 3.0000x_2 = 2.0001 \quad (1)$$
$$1.0000x_1 + 1.0000x_2 = 1.0000 \quad (2)$$

Gauss elimination: Pivoting

On the other hand, if the equations are solved in reverse order, the row with the larger pivot element is normalized. The equations are

$$\begin{aligned} 1.0000x_1 + 1.0000x_2 &= 1.0000 \quad (2) \\ 0.0003x_1 + 3.0000x_2 &= 2.0001 \quad (1) \end{aligned} \Longrightarrow \begin{aligned} 1.0000x_1 + 1.0000x_2 &= 1.0000 \\ 2.9997x_2 &= 1.9998 \end{aligned}$$

Elimination and substitution yield $x_2 = 2/3$

$$x_1 = (1.0000 - x_2)/(1.0000)$$
$$x_2 = 2/3$$

Case study 1

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

This case is much less sensitive to the number of significant figures in the computation

Significant Figures	x_2	x_1	Absolute Value of Percent 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

$\Rightarrow$ Thus, a pivot strategy is much more satisfactory

Exercise

$$\begin{aligned} 2x_2 + 5x_3 &= 1 \\ 2x_1 + x_2 + x_3 &= 1 \\ 3x_1 + x_2 &= 2 \end{aligned}$$

The exact solution is: $x_1 = -2$, $x_2 = 8$, $x_3 = -3$.

- Topics
  - Introduction of systems
  - Gauss Elimination
    - Naïve Gauss Elimination
    - Gauss Elimination: Pivoting
  - Iterative Methods
    - Gauss-Seidel method
    - Jacobi method

- Iterative methods: is a mathematical procedure that generates a sequence of approximate solutions hopefully converging to the exact solution
  - Stationary
    - ✓Gauss-Seidel
    - ✓Jacobi
  - Non Stationary
    - ✓GCR, CG, GMRES…..

- Iterative methods
  - nonlinear equations
  - linear problems involving a large number of variables (sometimes of the order of millions)

Gauss-Seidel Method

- the most commonly used iterative method for solving linear equations
- Employ initial guesses
- iterates to obtain refined estimates of the solution
- Convergence can be checked using the criterion

⇒ round-off errors controlled by the number of iterations

Gauss-Seidel Method

Principle

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

If the diagonal elements are all nonzero,

- the first equation can be solved for x_1
- the second for x_2
- the third for x_3

From Equation 1 $\longrightarrow$ $$x_1 = \frac{b_1 - a_{12}x_2 - a_{13}x_3}{a_{11}}$$

From Equation 2 $\longrightarrow$ $$x_2 = \frac{b_2 - a_{21}x_1 - a_{23}x_3}{a_{22}}$$

From Equation 3 $\longrightarrow$ $$x_3 = \frac{b_3 - a_{31}x_1 - a_{32}x_2}{a_{33}}$$

A simple approach is to assume that the initial guesses are all zero

$$x_1^j = \frac{b_1 - a_{12}x_2^{j-1} - a_{13}x_3^{j-1}}{a_{11}}$$

$$x_2^j = \frac{b_2 - a_{21}x_1^{j} - a_{23}x_3^{j-1}}{a_{22}}$$

$$x_3^j = \frac{b_3 - a_{31}x_1^{j} - a_{32}x_2^{j}}{a_{33}}$$

where j and $j - 1$ are the present and previous iterations

⇨ As each new x value is computed, it is immediately used in the next equation to determine another x value

Gauss-Seidel Method

Convergence can be checked using the criterion that for all *i*

$$\varepsilon_{a,i} = \left| \frac{x_i^j - x_i^{j-1}}{x_i^j} \right| \times 100\% \leq \varepsilon_s$$

The iterations are stopped when the absolute relative approximate error is less than a prespecified tolerance for all unknowns

❖ Gauss-Seidel Method

If the following condition holds, Gauss-Seidel will converge

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|$$

Case study 1

Use the Gauss-Seidel method to obtain the solution for

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

Recall that the true solution is $x_1 = 3$, $x_2 = -2.5$, and $x_3 = 7$

The procedure consisted of steps

Step 1: First, solve each of the equations for its unknown on the diagonal

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

Step 2: By assuming that $x_2=0$ and $x_3=0$, Eq. (1) can be used to compute

$$x_1 = \frac{7.85 + 0.1(0) + 0.2(0)}{3} = 2.616667$$

Step 3: This value x_1, along with the assumed value of $x_3 = 0$, can be substituted into Eq. (2) to calculate

$$x_2 = \frac{-19.3 - 0.1(2.616667) + 0.3(0)}{7} = -2.794524$$

Step 4: The first iteration is completed by substituting the calculated values for x_1 and x_2 into Eq. (3) to yield

$$x_3 = \frac{71.4 - 0.3(2.616667) + 0.2(-2.794524)}{10} = 7.005610$$

Step 5: For the second iteration, the same process is repeated to compute

$$x_1 = \frac{7.85 + 0.1(-2.794524) + 0.2(7.005610)}{3} = 2.990557$$

$$x_2 = \frac{-19.3 - 0.1(2.990557) + 0.3(7.005610)}{7} = -2.499625$$

$$x_3 = \frac{71.4 - 0.3(2.990557) + 0.2(-2.499625)}{10} = 7.000291$$

estimate the error $|\varepsilon_{a,1}| = \left|\frac{2.990557 - 2.616667}{2.990557}\right| 100\% = 12.5\%$

and $|\varepsilon_{a,2}| = 11.8\%$ $|\varepsilon_{a,3}| = 0.076\%.$

⇨ Additional iterations could be applied to improve the solutions

Exercise

$$
\begin{aligned}
4x_1 + x_2 - x_3 &= 3 \\
2x_1 + 7x_2 + x_3 &= 19 \\
x_1 - 3x_2 + 12x_3 &= 31
\end{aligned}
$$

0	0	0	0
1	0,75	2,50	3,15
2	0,91	**2,00**	3,01
3	1,00	2,00	3,00
4	1,00	2,00	3,00

The exact solution is: $x_1 = 1$, $x_2 = 2$, $x_3 = 3$.

Jacobi Method

- an algorithm for determining the solutions of a diagonally dominant system of linear equations
- transforms a matrix to a diagonal matrix by eliminating off-diagonal terms in a systematic fashion
- requires an infinite number of operations because the removal of each nonzero element often creates a new nonzero value at a previous zero element
- the approach is iterative in that it is repeated until the off-diagonal terms are "sufficiently" small
- An alternative approach, with Gauss-Seidel, utilizes a somewhat different tactic

- The difference between the Gauss-Seidel method and Jacobi iteration

First iteration

(a)

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$
$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$
$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

Second iteration

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$
$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$
$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

(b)

First iteration

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$
$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$
$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

Second iteration

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$
$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$
$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

System

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

Initial guesses

$$x^0 = \begin{bmatrix} x_1^0 \\ x_2^0 \\ \vdots \\ x_n^0 \end{bmatrix}$$

First iteration

$$x_1^1 = \frac{1}{a_{11}}(b_1 - a_{12}x_2^0 - \cdots - a_{1n}x_n^0)$$

$$x_2^1 = \frac{1}{a_{22}}(b_2 - a_{21}x_1^0 - a_{23}x_3^0 - \cdots - a_{2n}x_n^0)$$

$$x_n^1 = \frac{1}{a_{nn}}(b_n - a_{n1}x_1^0 - a_{n2}x_2^0 - \cdots - a_{nn-1}x_{n-1}^0)$$

(k+1)th iteration - Generalization

$$x_i^{k+1} = \frac{1}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^k - \sum_{j=i+1}^{n} a_{ij}x_j^k\right]$$

$$\Rightarrow \quad \boxed{x^{(k+1)} = D^{-1}\left(L+U\right)x^{(k)} + D^{-1}b}$$

Case study 2

Solve the following linear system using the Jacobi-Iterative method

$$
\begin{aligned}
10x_1 - x_2 + 2x_3 &= 6, \\
-x_1 + 11x_2 - x_3 + 3x_4 &= 25, \\
2x_1 - x_2 + 10x_3 - x_4 &= -11, \\
3x_2 - x_3 + 8x_4 &= 15.
\end{aligned}
$$

Case study 2

$$\begin{aligned} 10x_1 - x_2 + 2x_3 &= 6, \\ -x_1 + 11x_2 - x_3 + 3x_4 &= 25, \\ 2x_1 - x_2 + 10x_3 - x_4 &= -11, \\ 3x_2 - x_3 + 8x_4 &= 15. \end{aligned}$$

If we choose (0, 0, 0, 0) as the initial approximation, then the first approximate solution is given by

$$\begin{aligned} x_1 &= (6 - 0 - 0)/10 = 0.6, \\ x_2 &= (25 - 0 - 0)/11 = 25/11 = 2.2727, \\ x_3 &= (-11 - 0 - 0)/10 = -1.1, \\ x_4 &= (15 - 0 - 0)/8 = 1.875. \end{aligned}$$

Case study 2

- Using the approximations obtained, the iterative procedure is repeated until the desired accuracy has been reached
- The following are the approximated solutions after five iterations

x_1	x_2	x_3	x_4
0.6	2.27272	-1.1	1.875
1.04727	1.7159	-0.80522	0.88522
0.93263	2.05330	-1.0493	1.13088
1.01519	1.95369	-0.9681	0.97384
0.98899	2.0114	-1.0102	1.02135

The exact solution of the system is $(1, 2, -1, 1)$

Gauss-Seidel Method: Matlab

$$x_1^{\text{new}} = \frac{b_1}{a_{11}} - \frac{a_{12}}{a_{11}} x_2^{\text{old}} - \frac{a_{13}}{a_{11}} x_3^{\text{old}}$$

$$x_2^{\text{new}} = \frac{b_2}{a_{22}} - \frac{a_{21}}{a_{22}} x_1^{\text{new}} - \frac{a_{23}}{a_{22}} x_3^{\text{old}}$$

$$x_3^{\text{new}} = \frac{b_3}{a_{33}} - \frac{a_{31}}{a_{33}} x_1^{\text{new}} - \frac{a_{32}}{a_{33}} x_2^{\text{new}}$$

the solution can be expressed concisely in matrix form as

$$\{x\} = \{d\} - [C]\{x\}$$

where

$$\{d\} = \begin{Bmatrix} b_1/a_{11} \\ b_2/a_{22} \\ b_3/a_{33} \end{Bmatrix} \qquad [C] = \begin{bmatrix} 0 & a_{12}/a_{11} & a_{13}/a_{11} \\ a_{21}/a_{22} & 0 & a_{23}/a_{22} \\ a_{31}/a_{33} & a_{32}/a_{33} & 0 \end{bmatrix}$$

Convergence Criterion for the Gauss-Seidel Method

- it was sometimes non-convergent (diverge)
- when it converged, it often did so very slowly.

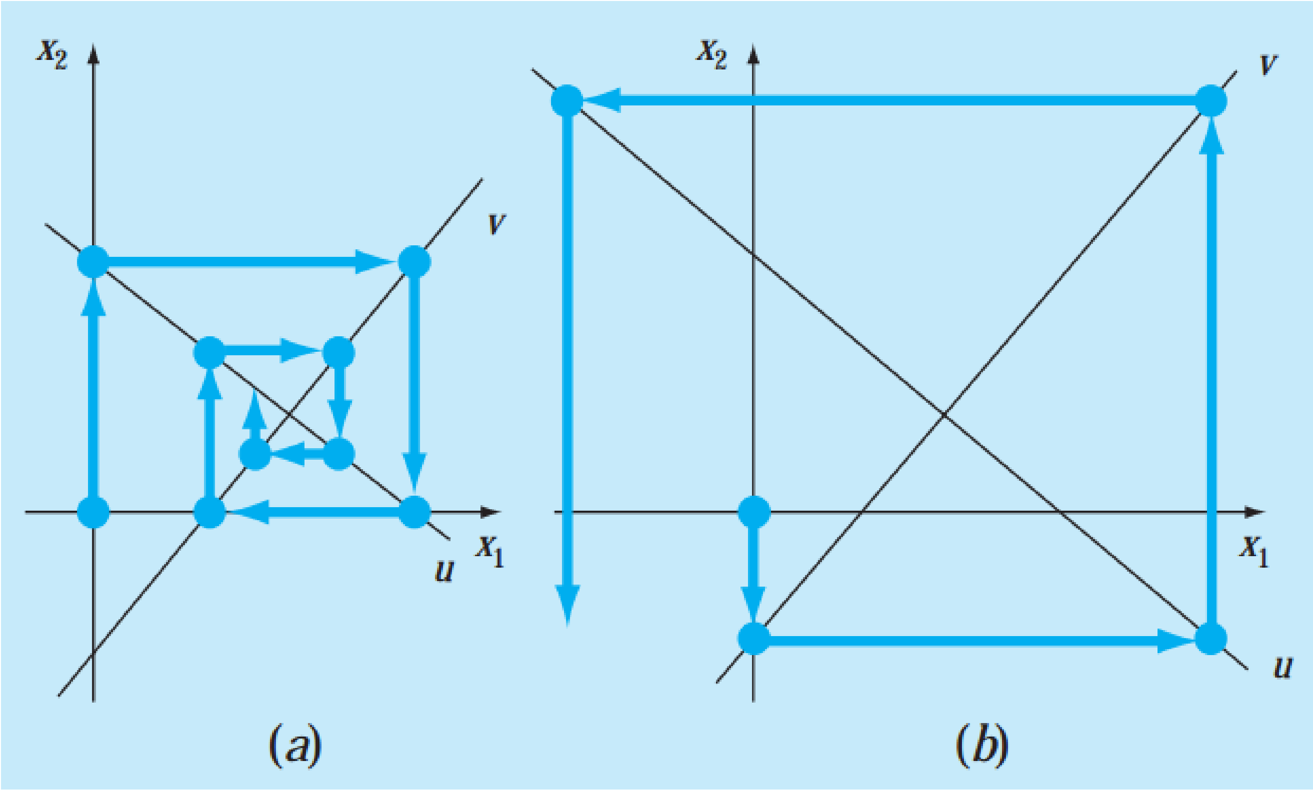
It can be shown that if the following condition holds, Gauss-Seidel will converge:

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|$$

Diagonally dominant. This criterion is sufficient but not necessary for convergence

Convergence Criterion for the Gauss-Seidel Method

- it was sometimes non-convergent (diverge)
- when it converged, it often did so very slowly.



(a) convergence
b) divergence
of the Gauss-Seidel method

Convergence Criterion for the Gauss-Seidel Method

It can be shown that if the following condition holds, Gauss-Seidel will converge:

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|$$

- The systems are callled *Diagonally dominant*
- This criterion is sufficient but not necessary for convergence

Improvement of Convergence Using Relaxation

- Relaxation represents a slight modification of the Gauss-Seidel method and is designed to enhance convergence.
- After each new value of x is computed, that value is modified by a weighted average of the results of the previous and the present iterations:

$$x_i^{\text{new}} = \lambda x_i^{\text{new}} + (1 - \lambda) x_i^{\text{old}}$$

where λ is a weighting factor that is assigned a value between 0 and 2.

If $\lambda = 1$, $(1 - \lambda)$ is equal to 0 and the result is unmodified

Improvement of Convergence Using Relaxation

$$x_i^{\text{new}} = \lambda x_i^{\text{new}} + (1 - \lambda) x_i^{\text{old}}$$

- if λ : between 0 and 1, the result is a weighted average of the present and the previous results. This type of modification is called underrelaxation
- It is typically employed to make a nonconvergent system converge or to hasten convergence by dampening out oscillations.

Improvement of Convergence Using Relaxation

$$x_i^{\text{new}} = \lambda x_i^{\text{new}} + (1 - \lambda) x_i^{\text{old}}$$

- if λ is set at a value between 1 and 2. This type of modification is called overrelaxation
- accelerate the convergence of an already convergent system.
- The approach is also called successive or simultaneous overrelaxation, or SOR.

Improvement of Convergence Using Relaxation

$$x_i^{\text{new}} = \lambda x_i^{\text{new}} + (1 - \lambda) x_i^{\text{old}}$$

- The choice of a proper value for λ is highly problem-specific and is often determined empirically.
- For a single solution of a set of equations it is often unnecessary.
- However, if the system under study is to be solved repeatedly, the efficiency introduced by a wise choice of λ can be extremely important.
- Good examples are the very large systems of partial differential equations that often arise when modeling continuous variations of variables

Improvement of Convergence Using Relaxation

Gauss-Seidel Method with Relaxation

Problem Statement. Solve the following system with Gauss-Seidel using overrelaxation ($\lambda = 1.2$) and a stopping criterion of $\varepsilon_s = 10\%$:

$$
\begin{aligned}
-3x_1 + 12x_2 &= 9 \\
10x_1 - \quad 2x_2 &= 8
\end{aligned}
$$

Improvement of Convergence Using Relaxation

Solution. First rearrange the equations so that they are diagonally dominant and solve the first equation for x_1 and the second for x_2:

$$x_1 = \frac{8 + 2x_2}{10} = 0.8 + 0.2x_2$$
$$x_2 = \frac{9 + 3x_1}{12} = 0.75 + 0.25x_1$$

First iteration: Using initial guesses of $x_1 = x_2 = 0$, we can solve for x_1:

$$x_1 = 0.8 + 0.2(0) = 0.8$$

Before solving for x_2, we first apply relaxation to our result for x_1:

$$x_{1,r} = 1.2(0.8) - 0.2(0) = 0.96$$

We use the subscript r to indicate that this is the "relaxed" value. This result is then used to compute x_2:

$$x_2 = 0.75 + 0.25(0.96) = 0.99$$

We then apply relaxation to this result to give

$$x_{2,r} = 1.2(0.99) - 0.2(0) = 1.188$$

At this point, we could compute estimated errors with Eq. (12.2). However, since we started with assumed values of zero, the errors for both variables will be 100%.

Second iteration: Using the same procedure as for the first iteration, the second iteration yields

$$x_1 = 0.8 + 0.2(1.188) = 1.0376$$
$$x_{1,r} = 1.2(1.0376) - 0.2(0.96) = 1.05312$$
$$\varepsilon_{a,1} = \left| \frac{1.05312 - 0.96}{1.05312} \right| \times 100\% = 8.84\%$$
$$x_2 = 0.75 + 0.25(1.05312) = 1.01328$$
$$x_{2,r} = 1.2(1.01328) - 0.2(1.188) = 0.978336$$
$$\varepsilon_{a,2} = \left| \frac{0.978336 - 1.188}{0.978336} \right| \times 100\% = 21.43\%$$

Because we have now have nonzero values from the first iteration, we can compute approximate error estimates as each new value is computed. At this point, although the error estimate for the first unknown has fallen below the 10% stopping criterion, the second has not. Hence, we must implement another iteration.

Third iteration:

$$x_1 = 0.8 + 0.2(0.978336) = 0.995667$$
$$x_{1,r} = 1.2(0.995667) - 0.2(1.05312) = 0.984177$$
$$\varepsilon_{a,1} = \left|\frac{0.984177 - 1.05312}{0.984177}\right| \times 100\% = 7.01\%$$

$$x_2 = 0.75 + 0.25(0.984177) = 0.996044$$
$$x_{2,r} = 1.2(0.996044) - 0.2(0.978336) = 0.999586$$
$$\varepsilon_{a,2} = \left|\frac{0.999586 - 0.978336}{0.999586}\right| \times 100\% = 2.13\%$$

At this point, we can terminate the computation because both error estimates have fallen below the 10% stopping criterion. The results at this juncture, $x_1 = 0.984177$ and $x_2 = 0.999586$, are converging on the exact solution of $x_1 = x_2 = 1$.

Case study

Solve the following system using three iterations with Gauss-Seidel using overrelaxation $(\lambda = 1.25)$

$$3x_1 + 8x_2 = 11$$
$$6x_1 - x_2 = 5$$

Recall that the true solution is $x_1 =$, $x_2 =$

Matlab

SOLVING LINEAR ALGEBRAIC EQUATIONS WITH MATLAB

MATLAB provides two direct ways to solve systems of linear algebraic equations. The most efficient way is to employ the backslash, or “left-division,” operator as in

```
>> x = A\b
>> x = mldivide(A,B)
```

The second is to use matrix inversion:

```
>> x = inv(A)*b
```

MATLAB Naïve Gauss Elimination

(a) forward elimination

```
FOR k = 1, n - 1
    FOR i =k + 1, n
        factor = ai,k / ak,k
        FOR j = k + 1 to n
            ai,j = ai,j  - factor . ak,j
        END
        bi =bi  - factor . bk
    END
END
```

(b) Back substitution

```
xn  = bn / an,n
FOR i = n - 1, 1, -1
    sum = bi
    FOR j = i + 1, n
        sum = sum - ai,j . xj
    END
    xi    =  sum / ai,i
END
```

MATLAB Naïve Gauss Elimination

(a) forward elimination

```
% forward elimination
for k = 1:n-1
for i = k+1:n
factor = Aug(i,k)/Aug(k,k);
Aug(i,k:nb) = Aug(i,k:nb)-
factor*Aug(k,k:nb);
end
end
```

(b) Back substitution

```
% back substitution
x = zeros(n,1);
x(n) = Aug(n,nb)/Aug(n,n);
for i = n-1:-1:1
x(i) = (Aug(i,nb)-
Aug(i,i+1:n)*x(i+1:n))/Aug(i,i);
end
```

MATLAB Naïve Gauss Elimination

Use naive Gauss elimination to solve the following system

$$
\begin{aligned}
7x_1 + 2x_2 - 3x_3 &= -12 \\
2x_1 + 5x_2 - 3x_3 &= -20 \\
x_1 - x_2 - 6x_3 &= -26
\end{aligned}
$$

MATLAB Gauss Elimination-Pivoting

```
function x = GaussPivot(A,b)
% GaussPivot: Gauss elimination pivoting
% x = GaussPivot(A,b): Gauss elimination with
pivoting.
% input:
% A = coefficient matrix
% b = right hand side vector
% output:
% x = solution vector
[m,n]=size(A);
if m~=n, error('Matrix A must be square');
end
nb=n+1;
Aug=[A b];
```

MATLAB Gauss Elimination-Pivoting

The max function has the syntax [y,i] = max(x) where y is the largest element in the vector x, and *i* is the index corresponding to that element

```
% forward elimination
for k = 1:n-1
% partial pivoting
[big,i]=max(abs(Aug(k:n,k)));
ipr=i+k-1;
if ipr~=k
Aug([k,ipr],:)=Aug([ipr,k],:);
end
for i = k+1:n
factor=Aug(i,k)/Aug(k,k);
Aug(i,k:nb)=Aug(i,k:nb)-
factor*Aug(k,k:nb);
end
end
```

```
% back substitution
x=zeros(n,1);
x(n)=Aug(n,nb)/Aug(n,n);
for i = n-1:-1:1
x(i)=(Aug(i,nb)-
Aug(i,i+1:n)*x(i+1:n))/Aug(i,i);
end
```

MATLAB Gauss Elimination-Pivoting

Use Gauss elimination with partial pivoting to solve the following system

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

MATLAB Gauss-Seidel

Before developing an algorithm, let us first recast Gauss-Seidel in a form that is compatible with MATLAB's ability to perform matrix operations.

$$
\begin{aligned}
x_1^{\text{new}} &= \frac{b_1}{a_{11}} & & & -\frac{a_{12}}{a_{11}}x_2^{\text{old}} - \frac{a_{13}}{a_{11}}x_3^{\text{old}} \\
x_2^{\text{new}} &= \frac{b_2}{a_{22}} & -\frac{a_{21}}{a_{22}}x_1^{\text{new}} & & -\frac{a_{23}}{a_{22}}x_3^{\text{old}} \\
x_3^{\text{new}} &= \frac{b_3}{a_{33}} & -\frac{a_{31}}{a_{33}}x_1^{\text{new}} & -\frac{a_{32}}{a_{33}}x_2^{\text{new}} &
\end{aligned}
$$

Notice that the solution can be expressed concisely in matrix form as $\{x\} = \{d\} - [C]\{x\}$

where
$$
\{d\} = \begin{Bmatrix} b_1/a_{11} \\ b_2/a_{22} \\ b_3/a_{33} \end{Bmatrix} \qquad [C] = \begin{bmatrix} 0 & a_{12}/a_{11} & a_{13}/a_{11} \\ a_{21}/a_{22} & 0 & a_{23}/a_{22} \\ a_{31}/a_{33} & a_{32}/a_{33} & 0 \end{bmatrix}
$$

MATLAB Gauss-Seidel

Exercise 1

Use Matlab and the Gauss-Seidel method to solve the following system

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

MATLAB Gauss-Seidel

Exercise 2

Use Matlab and the Gauss-Seidel method to solve the following system until the percent relative error falls below $\varepsilon_s = 5\%$

$$\begin{aligned} 10x_1 + 2x_2 - x_3 &= 27 \\ -3x_1 - 6x_2 + 2x_3 &= -61.5 \\ x_1 + x_2 + 5x_3 &= -21.5 \end{aligned}$$

MATLAB Jacobi - Iterative

Convert the system: $Ax = B$

into the equivalent system: $x = Cx + d$

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_1 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_1 + a_{33}x_3 &= b_3 \end{aligned}$$

$$\begin{aligned} x_1 &= \qquad\qquad\quad -\frac{a_{12}}{a_{11}}x_2 - \frac{a_{13}}{a_{11}}x_3 + \frac{b_1}{a_{11}} \\ x_2 &= -\frac{a_{21}}{a_{22}}x_1 \qquad\qquad - \frac{a_{23}}{a_{22}}x_3 + \frac{b_2}{a_{22}} \\ x_3 &= -\frac{a_{31}}{a_{33}}x_1 - \frac{a_{32}}{a_{33}}x_2 \qquad\qquad + \frac{b_3}{a_{33}} \end{aligned}$$

Generate a sequence of approximation

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots \qquad x^{(k)} = Cx^{(k-1)} + d$$

Exercise

Write the Matlab function to solve the system of linear equations using

- Jacobi method
- Gauss-Seidel method

Application: solve the following system

$$\begin{aligned} 2x_1 + x_2 - x_3 &= 8 \\ -3x_1 - x_2 + 2x_3 &= -11 \\ -2x_1 + x_2 + 2x_3 &= -3 \end{aligned}$$