23MAT204 – Mathematics for Intelligent Systems - 3
Practise Sheet-4
Gauss Jacobi and Gauss Siedel methods to numerically solve AX=B

Gauss – Jacobi Iteration method

Numerical Algorithm of Jacobi Method

Input: $A = [a_{ij}]$, $\boldsymbol{b}, \boldsymbol{XO} = \boldsymbol{x}^{(0)}$, tolerance TOL, maximum number of iterations N.

Step 1 Set $k = 1$

Step 2 while $(k \le N)$ do Steps 3-6

Step 3 For for $i = 1,2,, \dots n$

$$x_i = \frac{1}{a_{ii}}\left[\sum_{\substack{j=1,\\ j\neq i}}^{n}(-a_{ij}\boldsymbol{XO}_j) + b_i\right],$$

Step 4 If $||\boldsymbol{x} - \boldsymbol{XO}|| < TOL$, then OUTPUT $(x_1, x_2, x_3, \dots x_n)$;
STOP.

Step 5 Set $k = k + 1$.

Step 6 For for $i = 1,2,, \dots n$
Set $\boldsymbol{XO}_i = x_i$.

Step 7 OUTPUT $(x_1, x_2, x_3, \dots x_n)$;
STOP.

Gauss – Jacobi Iteration for A_{nxn} in matrix form

Consider to solve an $n \times n$ size system of linear equations $A\boldsymbol{x} = \boldsymbol{b}$ with $A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$ and $\boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$ for $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

We split A into

$$A = \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn} \end{bmatrix} - \begin{bmatrix} 0 & \dots & 0 & 0 \\ -a_{21} & \dots & 0 & 0 \\ \vdots & & \ddots & \vdots \\ -a_{n1} & \dots & -a_{n,n-1} & 0 \end{bmatrix} - \begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ 0 & 0 & & \vdots \\ \vdots & \vdots & \ddots & -a_{n-1,n} \\ 0 & 0 & \dots & 0 \end{bmatrix} = D - L - U$$

$A\boldsymbol{x} = \boldsymbol{b}$ is transformed into $(D - L - U)\boldsymbol{x} = \boldsymbol{b}$

$$D\boldsymbol{x} = (L + U)\boldsymbol{x} + \boldsymbol{b}$$

Assume D^{-1} exists and $D^{-1} = \begin{bmatrix} \frac{1}{a_{11}} & 0 & \dots & 0 \\ 0 & \frac{1}{a_{22}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{a_{nn}} \end{bmatrix}$

Then

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

The matrix form of Jacobi iterative method is

$$\boldsymbol{x}^{(k)} = D^{-1}(L + U)\boldsymbol{x}^{(k-1)} + D^{-1}\boldsymbol{b} \qquad k = 1,2,3, \dots$$

Define $T = D^{-1}(L + U)$ and $\boldsymbol{c} = D^{-1}\boldsymbol{b}$, Jacobi iteration method can also be written as

$$\boldsymbol{x}^{(k)} = T\boldsymbol{x}^{(k-1)} + \boldsymbol{c} \qquad k = 1,2,3, \dots$$

Example 1: Solve the below system of linear equations using Gauss-Jacobi method with initial solution as (0,0,0).

$$
\begin{aligned}
5x_1 - 2x_2 + 3x_3 &= -1 \\
-3x_1 + 9x_2 + x_3 &= 2 \\
2x_1 - x_2 - 7x_3 &= 3
\end{aligned}
$$

```
A=[5,-2,3;-3,9,1;2,-1,-7];
b=[-1;2;3];
n=size(A,1)
D=diag(diag(A));
L=-tril(A,-1);
% to generate the L matrix with negative values
% of A in lower triangular part and also with diagonal zero
U=-triu(A,1);
% to generate the U matrix with negative values
% of A in upper triangular part and also with diagonal zero
T=inv(D)*(L+U);
c=inv(D)*b;
x0=[0;0;0];
x1=T*x0+c
x2=T*x1+c
x3=T*x2+c
x4=T*x3+c
x5=T*x4+c
x6=T*x5+c
```

```
x1 = 3x1
   -0.2000
    0.2222
   -0.4286

x2 = 3x1
    0.1460
    0.2032
   -0.5175

x3 = 3x1
    0.1917
    0.3284
   -0.4159

x4 = 3x1
    0.1809
    0.3323
   -0.4207

x5 = 3x1
    0.1854
    0.3293
   -0.4244

x6 = 3x1
    0.1863
    0.3312
   -0.4226
```

Another idea for easy coding of Gauss-Jacobi method

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

$$x_i^{k+1} = x_i^k + \frac{1}{a_{ii}}\left[b_i - \sum_{i=1}^{j=n} a_{ij}x_j^k\right]$$
$$= x_i^k + \frac{1}{a_{ii}}\left[b_i - \left(\textit{dotproduct}\ \text{of ith row of A with old x vector}\right)\right]$$

```
for i=1:nRow
          xnew(i)=xold(i)+ (b(i)-A(i,:)*xold)/A(i,i);
End
xold=xnew
```

Example 2: Solve the system using Gauss-Jacobi method

$$\begin{aligned}
-4x_1 + 2x_2 + x_3 &= -4\\
x_1 - 4x_2 + x_3 + x_4 &= 11\\
2x_1 + x_2 - 4x_3 + x_4 + 2x_5 &= -16\\
x_2 + x_3 - 4x_4 + x_5 &= 11\\
x_3 + 2x_4 - 4x_5 &= -4
\end{aligned}$$

```
%Gauss Jacobi Iteration Method
clc;
clear all;
A=[-4 2 1 0 0;1 -4 1 1 0;2 1 -4 1 2;0 1 1 -4 1;0 0 1 2 -4];
 b=[-4 11 -16 11 -4];
 maxIter=1000;
 errorLimit=0.00001;
 resLimit=0.00001;
 x=[1,1,1,1,1]';

[xnew,k,relError]=my_Jacobi(A,x,b,maxIter,errorLimit,resLimit);
%Check the result
display('the solution vector is')
xnew'
display('recomputed b is')
(A*xnew)'
display('original b is')
b
```

```matlab
function [xnew,k,relError]=my_Jacobi(A,x,b,maxIter,errorLimit,resLimit)
    [nRow,nCol]=size(A);
    xold=x(:);  % convert into column vector if it is not
    b=b(:);% convert into column vector if it is not
    k=0;
    relError=zeros(maxIter,1);
    Notsolved=true;
    xnew=zeros(size(xold));
    while Notsolved
          k=k+1;
          for i=1:nRow
            xnew(i)=xold(i)+ (b(i)-A(i,:)*xold)/A(i,i);
           end
        currentError=norm(xnew-xold);
        relError(k)=currentError/norm(xnew);
        if norm(b-A*xnew)<=resLimit || currentError<=errorLimit || k>maxIter
            Notsolved=false;
        else
            xold=xnew;
        end
    end
end
```

```
the solution vector after all iterations is
ans =
   1.0000   -2.0000    4.0001   -2.0000    1.0000
recomputed b is
ans =
  -4.0000   11.0000  -16.0000   11.0000   -4.0000
original b is
b =
   -4    11   -16    11    -4
```

Gauss – Siedel Method for solving AX=B:

Numerical Algorithm of Gauss-Seidel Method

Input: $A = [a_{ij}]$, $\boldsymbol{b}, \boldsymbol{XO} = \boldsymbol{x}^{(0)}$, tolerance TOL, maximum number of iterations N.

Step 1 Set $k = 1$

Step 2 while ($k \leq N$) do Steps 3-6

Step 3 For for $i = 1,2,, \ldots n$

$$x_i = \frac{1}{a_{ii}}\left[-\sum_{j=1}^{i-1}(a_{ij}x_j) - \sum_{j=i+1}^{n}(a_{ij}\boldsymbol{XO}_j) + b_i\right],$$

Step 4 If $||\boldsymbol{x} - \boldsymbol{XO}|| < TOL$, then OUTPUT $(x_1, x_2, x_3, \ldots x_n)$;

STOP.

Step 5 Set $k = k + 1$.

Step 6 For for $i = 1,2,, \ldots n$

Set $\boldsymbol{XO}_i = x_i$.

Step 7 OUTPUT $(x_1, x_2, x_3, \ldots x_n)$;

STOP.

Gauss – Siedel Method in matrix form:

$$x_i^{(k)} = \frac{1}{a_{ii}}\left[-\sum_{j=1}^{i-1}(a_{ij}x_j^{(k)}) - \sum_{j=i+1}^{n}(a_{ij}x_j^{(k-1)}) + b_i\right], \qquad \text{for } i = 1,2,,\dots n$$

Namely,

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

Matrix form of Gauss-Seidel method.

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

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

Define $T_g = (D-L)^{-1}U$ and $\boldsymbol{c}_g = (D-L)^{-1}\boldsymbol{b}$, Gauss-Seidel method can be written as

$$\boldsymbol{x}^{(k)} = T_g\boldsymbol{x}^{(k-1)} + \boldsymbol{c}_g \qquad k = 1,2,3,\dots$$

Example 3: Solve the below system of linear equations using Gauss-Siedel method with initial solution as (0,0,0).

$$\begin{aligned}
5x_1 - 2x_2 + 3x_3 &= -1 \\
-3x_1 + 9x_2 + x_3 &= 2 \\
2x_1 - x_2 - 7x_3 &= 3
\end{aligned}$$

```
A=[5,-2,3;-3,9,1;2,-1,-7];
b=[-1;2;3];
n=size(A,1)
D=diag(diag(A));
L=tril(A,-1);
U=triu(A,1);
Tg=inv(D-L)*U
cg=inv(D-L)*b;
x0=[0;0;0];
x1=Tg*x0+cg
x2=Tg*x1+cg
x3=Tg*x2+cg
x4=Tg*x3+cg
x5=Tg*x4+cg
x6=Tg*x5+cg
```

```
x1 = 3×1
   -0.2000
    0.1556
   -0.5079

x2 = 3×1
    0.1670
    0.3343
   -0.4286

x3 = 3×1
    0.1909
    0.3335
   -0.4217

x4 = 3×1
    0.1864
    0.3312
   -0.4226

x5 = 3×1
    0.1861
    0.3312
   -0.4227

x6 = 3×1
    0.1861
    0.3312
   -0.4227
```

Gauss-Siedel gives the solution in lesser number of iterations than the Jacobi method.

For this problem the accuracy obtained in the fourth iteration of Gauss Siedel was obtained only in the sixth iteration of Gauss-Jacobi

Another idea for easy coding of Gauss-Siedel method

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

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

Slight modification in the code of Gauss Jacobi will give the code for Gauss- Siedel:

```
xnew=xold;  %  the crucial step
for i=1:nRow
          xnew(i)=xnew(i)+ (b(i)-A(i,:)*xnew)/A(i,i);
End
xold=xnew
```

Example 4: Solve the system using Gauss-Siedel method

$$-4x_1 + 2x_2 + x_3 = -4$$
$$x_1 - 4x_2 + x_3 + x_4 = 11$$
$$2x_1 + x_2 - 4x_3 + x_4 + 2x_5 = -16$$
$$x_2 + x_3 - 4x_4 + x_5 = 11$$
$$x_3 + 2x_4 - 4x_5 = -4$$

Practice Questions:

1. Solve the given problem by both Gauss-Jacobi and Gauss Siedel methods with initial vector as (1,1,1,1).
$$9x_1 + 2x_2 + x_3 + x_4 = 7$$
$$x_1 - 9x_2 + 2x_3 + x_4 = -2$$
$$2x_1 + x_2 + 5x_3 + x_4 = 14$$
$$x_2 + 2x_3 + 9x_4 = 14$$
In how many iterations the solution could be obtained using both methods?

2. Generate a random integer square matrix A of order 9. Obtain a vector b, such that Ax=b, with x=[a,b,c,c,a,c,c,b,a]^T, where: a is the last two digits of your registration number, b is your date of birth, c is your month of birth.
    a. Solve the system AX=b, using Gauss Elimination using rref.
    b. Solve the system AX=b, using Gauss-Jacobi iteration with starting point as origin. Verify in how many iterations you are getting the exact solution.
    c. Solve the system AX=b, using Gauss-Siedel iteration with starting point as origin. Verify in how many iterations you are getting the exact solution.