Date / /
Page No.

CL249 - Computational Methods Lab
Assignment 4 : Iterative Techniques

Submitted by :

AYUSHMAN CHOUDHARY
(200020039)

Q. Given a 15x15 matrix A and vector b, solve the equation :

$$Ax = b$$

using the Gauss-Seidel and Jacobi method of iterative technique. Use suitable initial guess and tolerance value for converge.

Compare the no. of operations for Jacobi, Gauss Seidel and Gauss Elimination method.

Method

(i) Jacobi Method

It is an iterative method to calculate solutions of the eq^n

$$Ax = b$$

Starting with an initial guess of

$$x^0 = (0\ 0\ 0\ \cdots\ 0)^T$$

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

$$\therefore \quad x_i^{(1)} = \frac{b_i - \sum\limits_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^0}{a_{ii}}$$

For general k,

$$x_i^{(k+1)} = \frac{b_i - \sum\limits_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)}}{a_{ii}}$$

Date / /
Page No.

Check for convergence ⇒

$$\max \left| \frac{x_i^{(k+1)} - x_i^{(k)}}{x^{(k+1)}} \right| < \text{tolerance.}$$

Gauss Seidel

Gauss Seidel is similar to Jacobi method, except that we use the latest value available for x in each iteration.

i.e.

After calculating x_1^1, latest value

$$x_2^1 = \frac{b_2 - \left(a_{11}^{?} x_1^1 + \sum_{j=3}^{n} a_{2j} x_j^0 \right)}{a_{22}}$$

∴ for general k,

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

Use same convergence criteria.

PSEUDOCODE

main.m

```
load A matrix from file
Initialize b vector

call Jacobi_method
call Gauss_seidel method
call Gauss Elimination method.
```

Jacobi_method.m

```
sz = length of b
x0 = x1 = zero vector

operations = 0
iter = 1e4.
ensure pivoting i.e. |Aii| >= |Aji| ∀ j > i
while (iter ≠ 0)

    loop i = 1 to sz

        temp = 0
        loop j = 1 to sz
            if (j==i) skip (continue)
            temp = temp + A(i,j)·x0(j)
        end loop
    x1(i) = (b(i) - temp)/A(i,i)

    update operations.
    end loop
```

% calculate max error

$$max\_error = \max \left| \frac{x_1(j) - x_0(j)}{x_1(j)} \right| \quad \forall j \in [1, SZ]$$

% convergence cond.

```
if (max_error < tol)
    return x1.
else
    x0 = x1.
    iter = iter - 1.
end loop (while)
```

gauss-seidel.m

```
SZ = length of b
x0 = x1 = zero vectors

op = 0
iter = 1e6
ensure pivoting i.e. |aii| >= |aji| for all j > i
while (iter != 0)

    loop i = 1 to SZ
        temp = 0

        loop j = 1 to i - 1.
            temp = temp + A(i,j) . x1(j)
        end loop
```

loop j = i+1 to sz
    temp = temp + A(i,j) · x_0(j)
end loop.

x_1(i) = (b(i) − temp) / A(i,i)

update op

end loop
~~end while (loop)~~

% calculate max error

$$\text{max\_error} = \max \left| \frac{x_1(j) - x_0(j)}{x_1(j)} \right| \quad \forall j \in [1, sz]$$

% convergence condition

if (max_error < tol)
    return x_1

else
    x_0 = x_1
    iter = iter − 1

end loop (while)

```
clear all;
clear;
clc;

A= load('A.txt'); % load matrix A
b= ones(size(A,1),1); % Initialize vector b
roll=39;
b=b*(roll+2);

%%%% Jacobi Method
fprintf('For Jacobi method: \n\n');
y= jacobi_method(A,b);
fprintf('x:\n');
disp(y);

%%%% Gauss Seidel Method
fprintf('For Gauss Seidel method: \n\n');
x= gauss_seidel(A,b);
fprintf('x:\n');
disp(x);
fprintf('A*x=\n');
disp(A*x);

%%%% Gauss Elimination Method
fprintf('For Gauss Elimination method: \n\n');
z= Gauss_elimination(A,b);
z= z.';
fprintf('x:\n');
disp(z);
fprintf('A*x=\n');
disp(A*z);
```

```
For Jacobi method:

No. of operations: 594645
x:
   NaN
   NaN
   NaN
   NaN
   NaN
   NaN
   NaN
   NaN
   NaN
   NaN
   Inf
  -Inf
   Inf
  -Inf
   Inf
```

For Gauss Seidel method:

```
No. of operations: 70724910
x:
   1.0e+05 *

    0.0123
    0.0676
    0.1603
    0.2849
    0.4366
    0.6109
    0.8036
    1.0111
    1.2300
    1.4575
    1.6912
    1.9290
    2.1693
    2.4108
    2.6527

A*x=
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
```

For Gauss Elimination method:

```
No. of operation = 2570
x:
   1.0e+05 *

    0.0123
    0.0677
    0.1603
    0.2850
    0.4367
    0.6109
    0.8036
    1.0111
    1.2300
```

```
    1.4576
    1.6913
    1.9291
    2.1693
    2.4108
    2.6527

A*x=
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
   41.0000
```

Published with MATLAB® R2021a

```
function x = jacobi_method(A,b)

sz= length(b);
Xo= zeros(sz,1);
X1= zeros(sz,1);

iter=1e4;
op=0;

% Pivoting
for i=1:sz
    diagonal_max=A(i,:);
    max_row=i;
    for k= i+1 : sz
        if(abs(A(k,i)) > abs(diagonal_max(i))) % Condition
            diagonal_max=A(k,:);
            max_row=k;
        end
    end

    A(max_row,:)=A(i,:); % Updating pivot row to max in column
    A(i,:)=diagonal_max;
    temp=b(i);
    b(i)=b(max_row); % Updating pivot row for b vector
    b(max_row)=temp;
end

% Jacobi Method
while(iter)
    for i=1:sz
        temp_sum=0;
        for j=1:sz
            if (j==i)
                continue;
            end
            temp_sum = temp_sum + A(i,j).*Xo(j); % taking sum of old
 values
        end
        op= op + 2.*(sz-1);
        X1(i)= (b(i) - temp_sum)./A(i,i); % Jacobi formula
        op= op + 1;
    end
    max_err=0;
    for k=1:sz
       temp_err= abs((X1(k)-Xo(k))./X1(k));
       if(temp_err > max_err) % Checking max error
           max_err= temp_err;
       end
    end
    if(max_err < 1e-10) % Convergence condition
       x=X1;
       fprintf('No. of operations: %i\n',op);
```

```
        return;
     end
    Xo= X1; % Replacing old values with new values
   iter=iter-1;
end
x= NaN;
disp('Did not converge');
fprintf('No. of operations: %i\n',op);
end
```

Published with MATLAB® R2021a

```
function x= gauss_seidel(A, b)

x0= zeros(size(A,1),1);
sz=size(x0,1);
x1=x0;
iter=1e6;
op=0;

% Pivoting
for i=1:sz
    diagonal_max=A(i,:);
    max_row=i;
    for k= i+1 : sz
        if(abs(A(k,i)) > abs(diagonal_max(i))) % Condition
            diagonal_max=A(k,:);
            max_row=k;
        end
    end

    A(max_row,:)=A(i,:); % Updating pivot row to max in column
    A(i,:)=diagonal_max;
    temp=b(i);
    b(i)=b(max_row); % Updating pivot row for b vector
    b(max_row)=temp;
end

% Gauss Seidel Method
while (iter)
    iter=iter-1;
    for i=1:sz
        temp_sum=0;
        for j=1:i-1
                temp_sum=temp_sum + A(i,j).*x1(j); % Taking sum of new
 values
        end
        for j=i+1:sz
                temp_sum=temp_sum + A(i,j).*x0(j); % Taking sum of old
 values
        end
        op= op + 2.*(sz-1);
        x1(i)=(b(i) - temp_sum)./A(i,i); % Gauss Seidel Formula
        op= op + 1;
    end
    max_err=0;
    for k=1:sz
       temp_err= abs((x1(k)-x0(k))./x1(k));
       if(temp_err > max_err) % Checking max error
           max_err= temp_err;
       end
    end
    if(max_err < 1e-11) % Convergence condition
       x=x1;
```

```
        fprintf('No. of operations: %i\n',op);
        return;
    end
    x0= x1; % Replacing old values with new values
end
disp('Did not converge\n');
fprintf('No. of operations: %i\n',op);
end
```

Published with MATLAB® R2021a

Comments & Remarks.

x vector for Jacobi method does not converge.

This is because the spectral radius > 1 for given matrix. (As taught in CL244).

Answer is obtained in Gauss Seidel method, but number of operations are remarkably higher than Gauss Eliminations due to many iterations.

for $\epsilon = 1e-10$, operations = 6.06 cr

as compared to Gauss Elimination's 2570

For perfect convergence, $\epsilon = 1e-11$.

No. of operations = 7.07 cr

$A^*x = b$ for Gauss Seidel & Gauss Elimination

```
function X= Gauss_elimination(A,B)
% Length of various matrices
col_A= size(A,2);
Y=[A B]; % Y= [A|B] format to minimize separate operations on A and B.
row_Y= size(Y,1);
col_Y= size(Y,2);

counter=0;
for i= 1 : row_Y
    % Pivot and largest diagonal element Condition
    diagonal_max=Y(i,:);
    max_row=i;
    for k= i+1 : row_Y
        if(abs(Y(k,i)) > diagonal_max(i)) % Condition
            diagonal_max=Y(k,:);
            max_row=k;
        end
    end
    Y(max_row,:)=Y(i,:); % Updating pivot value to max in column
    Y(i,:)=diagonal_max;

    % Gauss-elimination method
    if abs(Y(i,i)) > 1e-4 % Condition to ensure no operation is done
 on NULL element
        for j=i+1 : row_Y

            factor= Y(j,i)./Y(i,i); % calculating factor
            counter= counter+1;
            Y(j,:)=Y(j,:)-factor.*Y(i,:); % updating subsequent rows
            counter = counter + 2.*(col_Y - i);
        end
    end
end
% Back Substitution
X = zeros(1,col_A); % Initializing X vector to 0
col_X= size(X,2);
for i=row_Y:-1:1
    temp= sum(Y(i,i+1:col_Y-1).*X(i+1:col_X));
    counter= counter + 2.*(col_X -i) -1;
    X(i)= (Y(i,col_Y)-temp)./Y(i,i); % Backsubstitution formula
    counter= counter +2;
end
fprintf('No. of operation = %i\n', counter);
end
```

Published with MATLAB® R2021a