CL244 Tut3 Part B

Ayushman Choudhary (200020039)
Adhiraj Joshi (200020011)

Q1. (a)

i) Jacobi method :

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

$\therefore S = D \qquad T = -(L+U)$

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & -2 \end{bmatrix} \qquad T = \begin{bmatrix} 0 & -1 & -6 \\ -1 & 0 & 1 \\ -4 & -2 & 0 \end{bmatrix}$$

(ii) Gauss - Siedel :

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

$\therefore S = L+D \qquad T = -U$.

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 5 & 0 \\ 4 & 2 & -2 \end{bmatrix} \qquad T = \begin{bmatrix} 0 & -1 & -6 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

(iii) Successive Over Relaxation :

$$(D+\omega L)\, x^{(k+1)} = \big((1-\omega)D - \omega U\big)x^{k} + \omega b \qquad \omega = 1.2$$

$S = D + \omega L \qquad T = (1-\omega)D - \omega U$

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 1.2 & 5 & 0 \\ 4.8 & 2.4 & -2 \end{bmatrix} \qquad T = \begin{bmatrix} 0.2 & -1.2 & -7.2 \\ 0 & 1 & 1.2 \\ 0 & 0 & -0.4 \end{bmatrix}$$

Date / /
Page No.

(b) No convergence for Jacobi method within 100 iteration.

Gauss - Siedel method : $\lambda_{max} = -9.3659$

SOR method : $\lambda_{max} = -14.2084$

Table of Contents


```
clear all;
clear;
clc;
```

Jacobi Method

```
disp('Jacobi Method');
S= [
    1 0 0
    0 5 0
    0 0 -2 ];

T= [
    0 -1 -6
    -1 0 1
    -4 -2 0];

lamda_max=power_method(S,T)
lamda_min= 1./power_method(T,S)

cond_no= sqrt(abs(lamda_max./lamda_min))
```

Gauss- Siedel Method

```
disp('Gauss-Siedel Method');
S= [
    1 0 0
    1 5 0
    4 2 -2];

T= [
    0 -1 -6
    -1 0 1
    -4 -2 0];

lamda_max= power_method(S,T)
lamda_min= 1./power_method(T,S)

cond_no= sqrt(abs(lamda_max./lamda_min))
```

SOR

```
disp('SOR Method');
```

```
S= [
    1 0 0
    1.2 5 0
    4.8 2.4 -2];

T=[
    0.2 -1.2 -7.2
    0 1 1.2
    0 0 -0.4];

lamda_max= power_method(S,T)
lamda_min= 1./power_method(T,S)

cond_no= sqrt(abs(lamda_max./lamda_min))
```

Jacobi Method

```
No convergence within 100 iterations

lamda_max =

   NaN


lamda_min =

    0.0689


cond_no =

   NaN
```

Gauss-Siedel Method

```
lamda_max =

   -9.3659


lamda_min =

    0.0769


cond_no =

   11.0372
```

SOR Method

```
lamda_max =

  -14.2084
```

```
lamda_min =

   -0.0031

cond_no =

   67.6029
```

Published with MATLAB® R2021a

```
function lambda_max = power_method(S,T)

A=S\T;

Xo= [
    1
    2
    3];

counter=0;
prev_ratio=0;
t=100;
while (t~=0)
    Xk=A*Xo;
    curr_ratio= Xk(3)./Xo(3); % Ratio
    if (abs((curr_ratio-prev_ratio)./curr_ratio) < 1e-6) % comparison
        lambda_max=curr_ratio;
        break;
    end
    counter= counter+1;
    prev_ratio=curr_ratio;
    Xo=Xk;
    t=t-1;
end
if (t==0)
    disp('No convergence within 100 iterations');
    lambda_max=NaN;
end
end
```

Published with MATLAB® R2021a

Date __/__/__
Page No. ____

Q4.

$$A_1 = \begin{bmatrix} 4 & -1 & -1 & -1 \\ -1 & 4 & -1 & -1 \\ -1 & -1 & 4 & -1 \\ -1 & -1 & -1 & 4 \end{bmatrix}$$

Since matrix A is symmetric, we can use deflation method to obtain all eigen values

i.e. iterative use of power method.

$\because$ A is symmetric, its eigen vectors are orthogonal.

i.e. $x_i^T x_j = 0$ and $x_i^T x_i = 1$

$\hookrightarrow$ euclidean normalized.

if $|\lambda_1| > |\lambda_2| > |\lambda_3| - - - > |\lambda_n|$

we first compute λ_1 using power method.

Now, $A_2 = A_1 - \lambda_1 x_1 x_1^T$

eigen values of $A_2 = 0, \lambda_2, \lambda_3, \dots, \lambda_n$

Hence, using power method, we can calculate λ_2 and continue similarly.

```
clear all;
clear;
clc;

A= [
    4 -1 -1 -1
    -1 4 -1 -1
    -1 -1 4 -1
    -1 -1 -1 4 ];

eigen_values= zeros(1,4);

[eigen_values(1),x]= deflation(A);

for i= 2:4
    A= A - eigen_values(i-1).*(x*(x.'));
    [eigen_values(i),x]= deflation(A);
end

eigen_values
```

```
eigen_values =

    5.0000    1.0000    5.0000    5.0000
```

Published with MATLAB® R2021a

```
function [lambda_max,Xk] = deflation(A)

Xo= [
    0
    1
    2
    3];
counter=0;
prev_ratio=0;
t=100;
while (t~=0)
    Xk=A*Xo;
    curr_ratio= Xk(3)./Xo(3); % Ratio
    if (abs((curr_ratio-prev_ratio)./curr_ratio) < 1e-6) % comparison
        lambda_max=curr_ratio;
        Xk= Xk./norm(Xk); % Taking euclidean norm
        break;
    end
    counter= counter+1;
    prev_ratio=curr_ratio;
    Xo=Xk;
    t=t-1;
end
if (t==0)
    disp('No convergence within 100 iterations');
    lambda_max=NaN;
    Xk=NaN;
end
end
```

Published with MATLAB® R2021a