

## Worksheet #2

Professor: *Laurence Halpern - Juliette Ryan*

Student name: *Nguyen Tu Huy*

---

Course: *PUF - High Performance Computing*

Due date: *December 01th, 2022*

Let A be the

- a) matrix defined in worksheet 1
- b) The 1D Finite Difference Matrix
- c) Matrices defined in Matrix Market.

### 1. Exercise 1

In the restarting **gmres.m** you received , add the missing lines that transform the **Hessenberg** matrix into an upper matrix by applying Givens rotation and which also provides the error at iteration  $j + 1$ .

Check that your restarting **gmres.m** is correct using manufactured solutions.

- First , use a restart parameter that is greater than the matrix dimension. What does this imply?
- Analyse in terms of computing and memory storage , the effect of the restarting parameter.

Comment the efficiency of gmres compared to the previous iterative methods worksheet 1).

GMRES.m:

```
1 %*****
2 function [x,error,iter] = GMRES(A,b,x,m,itmax,epsi)
3 %*****
4 % gmres.m solves the linear system Ax=b
5 % using the Generalized Minimal residual ( GMRESm ) method with restarts
6 % input    A          REAL nonsymmetric positive definite matrix
7 %          x          REAL initial guess vector
8 %          b          REAL right hand side vector
9 %          m          INTEGER number of iterations between restarts
10 %          itmax      INTEGER maximum number of iterations
11 %          epsi       REAL error tolerance
12 %
13 % output   x          REAL solution vector
14 %          error      REAL error norm
15 %          iter       INTEGER number of iterations performed
16
```

```

17 % initialization
18
19 normb = norm(b);
20 if ( normb == 0.0 )
21     normb = 1.0;
22 end
23 % residual
24 r = b - A*x;
25 error(1) = norm(r)/normb;
26 if ( error(1) < epsi )
27     return;
28 end
29 [n,n] = size(A);
30 V(1:n,1:m+1) = zeros(n,m+1); % Vm+1=[Vm|qm+1]
31 H(1:m+1,1:m) = zeros(m+1,m); % Hessenberg matrix
32 cs(1:m) = zeros(m,1);
33 sn(1:m) = zeros(m,1);
34
35 e1 = zeros(n,1); % basic vector
36 e1(1) = 1.0;
37 iter = 1; % step of iterator
38 itt = 1;
39 while iter <= itmax % begin iteration
40     r = b-A*x;
41     V(:,1) = r/norm(r);
42     s = norm(r)*e1;
43     for j = 1:m % construct orthonormal
44         itt = itt+1; % basis using Gram-Schmidt
45         w = A*V(:,j);
46         for i = 1:j
47             H(i,j) = w'*V(:,i);
48             w = w - H(i,j)*V(:,i);
49         end
50         % size(H)
51         H(j+1,j) = norm(w);
52         V(:,j+1) = w/H(j+1,j);
53 % We tranform the Hessenberg matrix H into a triangular matrix by
54 % applying Givens rotation
55         for k = 1:j-1 % apply Givens rotation
56             temp = cs(k)*H(k,j) + sn(k)*H(k+1,j);
57             H(k+1,j) = -sn(k)*H(k,j) + cs(k)*H(k+1,j);
58             H(k,j) = temp;
59         end
60         cs(j) = H(j,j)/sqrt(H(j,j)^2 + H(j+1,j)^2);
61         sn(j) = H(j+1,j)/sqrt(H(j,j)^2 + H(j+1,j)^2);
62         temp = cs(j)*s(j); % approximate residual norm
63         s(j+1) = -sn(j)*s(j);
64         s(j) = temp;
65         H(j,j) = cs(j)*H(j,j) + sn(j)*H(j+1,j);
66         H(j+1,j) = 0.0;
67         % Which also provides the error at iteration j+1
68
69         error(it+1) = abs(s(j+1)) / normb;
70         if ( error(it+1) <= epsi ) % update approximation
71             y = H(1:j,1:j)\s(1:j); % and exit
72             x = x + V(:,1:j)*y;
73             % error(i+1) = abs(s(i+1)) / bnrm2;
74             break;

```

```

74         end
75     end
76
77     if ( error(itt+1) <= epsi ), break, end
78     y = H(1:m,1:m)\s(1:m);
79     % update approximation
80     x = x + V(:,1:m)*y;
81     % update approximation
82     % compute residual
83     r = b - A*x;
84     s(j+1) = norm(r);
85     error(itt+1) = s(j+1) / normb;           % check convergence
86     if ( error(itt+1) <= epsi )
87         break;
88     end
89     iter = iter+1;
90 end
91 iter = itt;

```

Matrix define in worksheet 1

Compare difference method:

Method	Number of iterations	Error	Time
GS Lower	42	3.1723e-08	0.012816
GS Upper	44	5.0821e-09	0.012816
Steepest Descent	1001	0.40714	0.053916
SOR	25	4.1048e-08	0.013954
CG	136	4.2446e-12	0.013785
GMRES	102	4.411e-14	0.045432

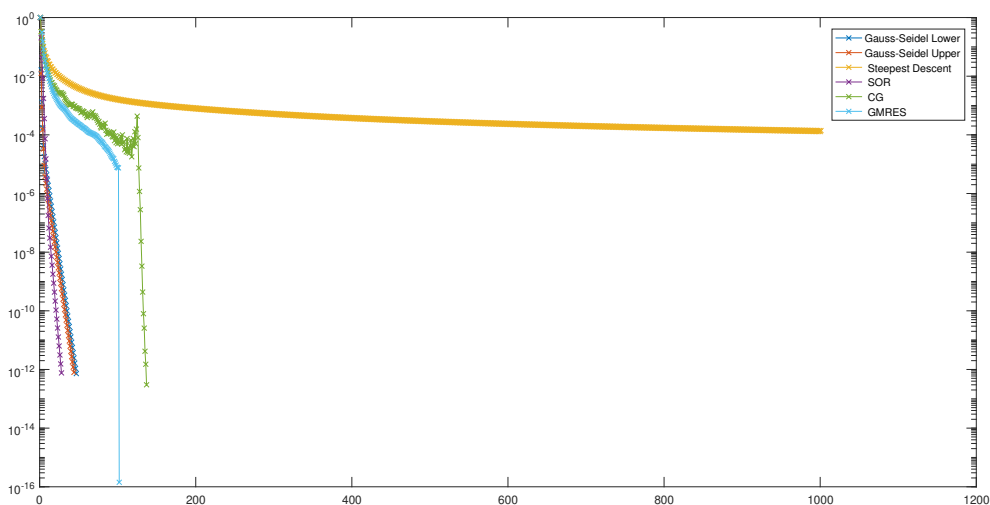


Figure 1: Comparison of value of  $\|x_{approx} - x_{exact}\|$  of different method with *semilogy*

## Matrix define Laplace 1D

### Compare difference method:

Method	Number of iterations	Error	Time
GS Lower	1001	4.8297	0.022161
GS Upper	1001	4.841	0.022161
Steepest Descent	1001	4.8702	0.070996
SOR	1001	4.8091	0.046463
CG	405	2.4533e-09	0.029979
GMRES	102	2.4673e-10	0.064383

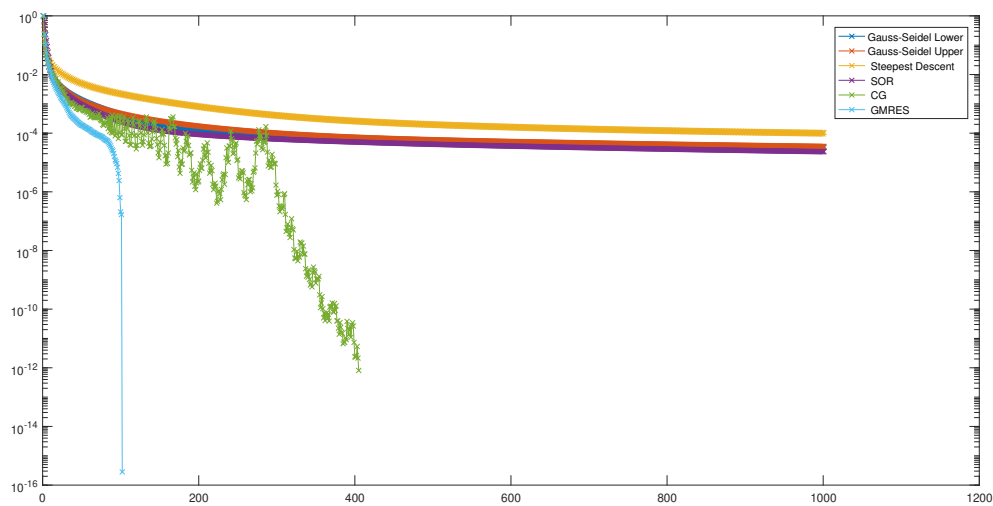


Figure 2: Comparison of value of  $\|x_{approx} - x_{exact}\|$  of different method with *semilogy*

### Using matrix from Matrix Market

1 FILE : 'hor131.mtx'

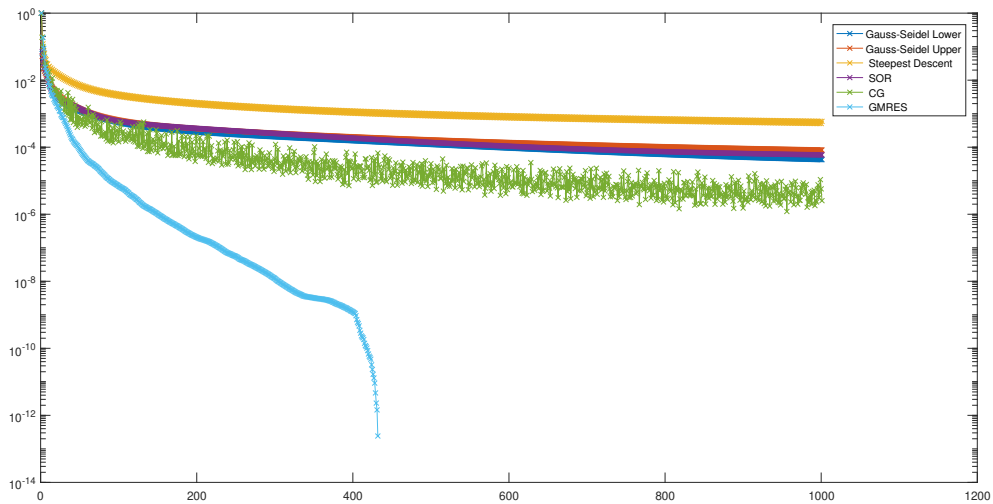


Figure 3: Comparison of value of  $\|x_{approx} - x_{exact}\|$  of different method with *semilogy*

## 2 FILE : 'pde225.mtx'

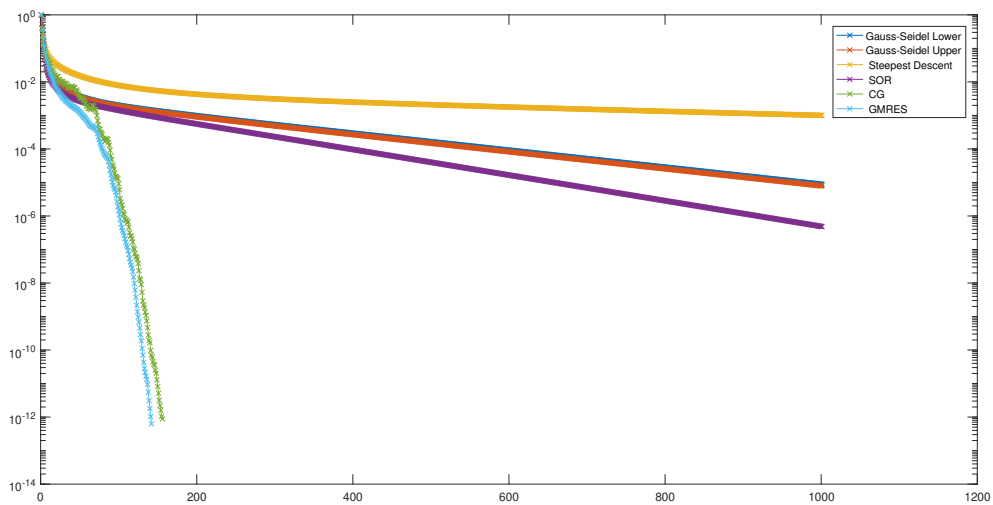


Figure 4: Comparison of value of  $\|x_{approx} - x_{exact}\|$  of different method with *semilogy*

## 3 FILE : 'saylr4.mtx'

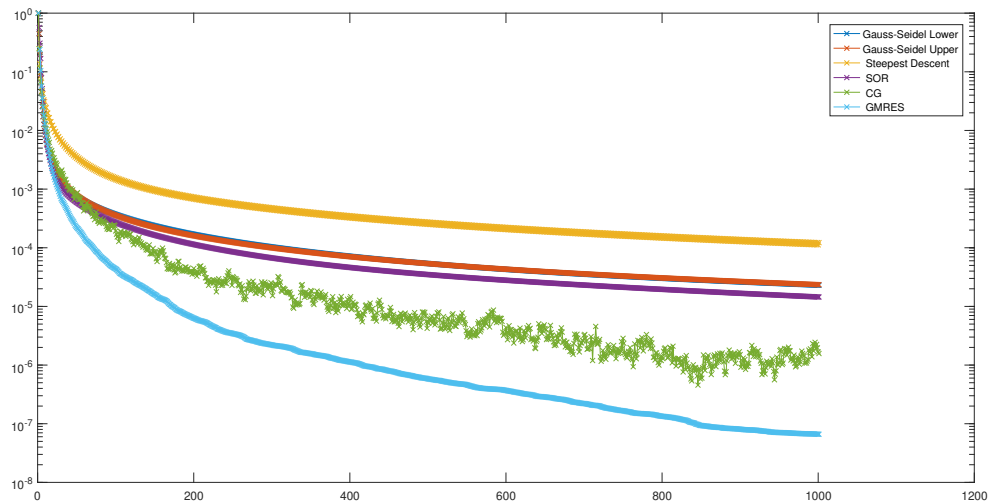


Figure 5: Comparison of value of  $\|x_{approx} - x_{exact}\|$  of different method with *semilog*

#### 4 FILE : 'sherman4.mtx'

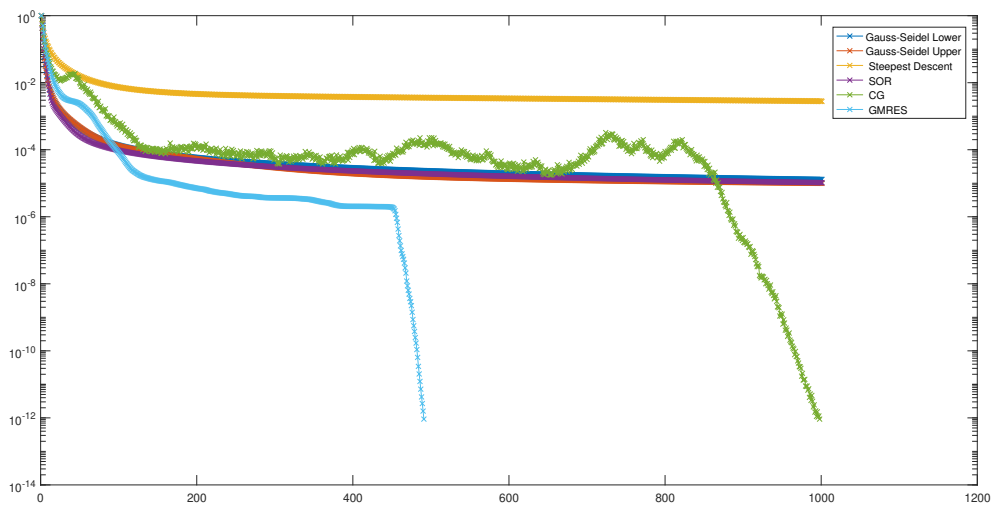


Figure 6: Comparison of value of  $\|x_{approx} - x_{exact}\|$  of different method with *semilog*

#### 5 FILE : 'tub1000.mtx'

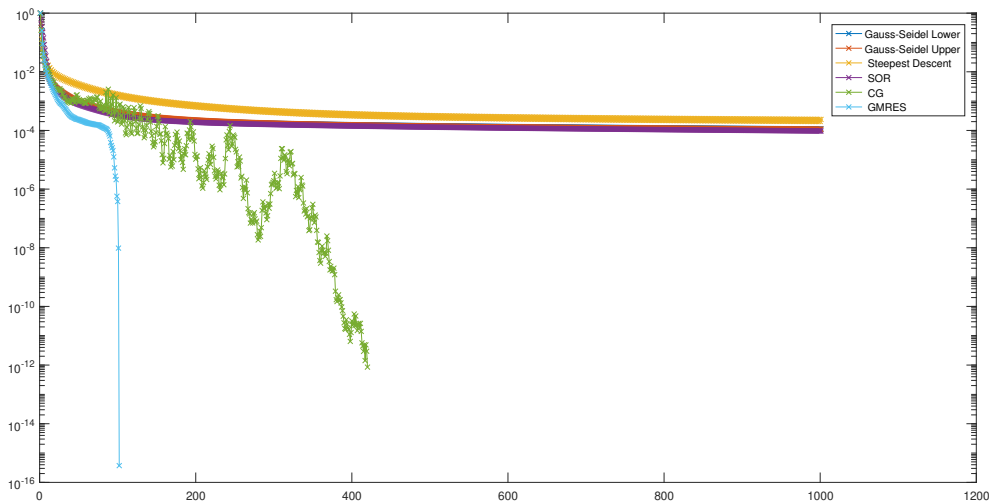


Figure 7: Comparison of value of  $\|x_{approx} - x_{exact}\|$  of different method with *semilogy*

**Commend:** For different case of matrix above, GMRES method show the best result among all method.

## 2. Exercise 2

**Left preconditioner:** Solve  $MAx = Mb$ .

**Right preconditioner:** Solve  $AMy = b$ , and then  $x = My$ .

Using the above **gmres.m**, modify it to **precgmres.m** so as to be able to use a right preconditioner  $M$  that can be changed at each restart of gmres.

Validate your **precgmres.m** defining  $M = \text{diag}(A)$

**Left preconditioner:**

```

1 function [x, error, iter] = Left_PRECGMRES( A,b,x,M,m,itmax,epsi)
2 % PRECGMRES.m solves the preconditioner linear system
3 % The left preconditioner solve MAX = b
4 % The right preconditioner solve AMy = b, then x = My
5 % Using the above gmres.m, modify it to precgmres.m so as to be able to
   use
6 % a right preconditioner M that can be changed at each restart of gmres.
7 % input    A          REAL nonsymmetric positive definite matrix
8 %          x          REAL initial guess vector
9 %          b          REAL right hand side vector
10 %          M          REAL preconditioner matrix
11 %          numb_m     INTEGER number of iterations between restarts
12 %          max_it     INTEGER maximum number of iterations
13 %          tol        REAL error tolerance
14 %
15 % output   x          REAL solution vector
16 %          error      REAL error norm
17 %          iter       INTEGER number of iterations performed
18 % initialization
19
20 normb = norm(b);

```

```

21 if ( normb == 0.0 )
22     normb = 1.0;
23 end
24
25 % residual
26 r = M*(b - A*x);
27 error(1) = norm(r)/normb;
28 if ( error(1) < epsi )
29     return;
30 end
31 [n,n] = size(A);
32 V(1:n,1:m+1) = zeros(n,m+1);    % Vm+1=[Vm|qm+1]
33 H(1:m+1,1:m) = zeros(m+1,m);    % Hessenberg matrix
34 cs(1:m) = zeros(m,1);
35 sn(1:m) = zeros(m,1);
36
37 e1 = zeros(n,1);                % basic vector
38 e1(1) = 1.0;
39 iter = 1;                        % step of iterator
40 ittt = 1;
41 while iter <= itmax              % begin iteration
42     r = M*(b - A*x);
43     V(:,1) = r/norm(r);
44     s = norm(r)*e1;
45     for j = 1:m                  % construct orthonormal
46         ittt = ittt + 1;         % basis using Gram-Schmidt
47         w = M*A*V(:,j);
48         for i = 1:j
49             H(i,j) = w'*V(:,i);
50             w = w - H(i,j)*V(:,i);
51         end
52         % size(H)
53         H(j+1,j) = norm(w);
54         V(:,j+1) = w/H(j+1,j);
55         % We tranform the Hessenberg matrix H into a triangular matrix
56         % by applying Givens rotation
57         for k = 1:j-1            % apply Givens
58             rotation
59                 temp = cs(k)*H(k,j) + sn(k)*H(k+1,j);
60                 H(k+1,j) = -sn(k)*H(k,j) + cs(k)*H(k+1,j);
61                 H(k,j) = temp;
62             end
63             cs(j) = H(j,j)/sqrt(H(j,j)^2 + H(j+1,j)^2);
64             sn(j) = H(j+1,j)/sqrt(H(j,j)^2 + H(j+1,j)^2);
65             temp = cs(j)*s(j);    % approximate
66         residual norm
67             s(j+1) = -sn(j)*s(j);
68             s(j) = temp;
69             H(j,j) = cs(j)*H(j,j) + sn(j)*H(j+1,j);
70             H(j+1,j) = 0.0;
71             % Which also provides the error at iteration j+1
72             error(ittt+1) = abs(s(j+1)) / normb;
73
74             if ( error(ittt+1) <= epsi )    % update
75                 approximation
76                     y = H(1:j,1:j)\s(1:j);    % and exit
77                     x = x + V(:,1:j)*y;

```



```

75         % error(i+1) = abs(s(i+1)) / bnorm2;
76         break;
77     end
78 end
79
80     if ( error(itt+1) <= epsi ), break, end
81     y = H(1:m,1:m)\s(1:m);
82 %     x = x + V*y; % update
83     approximation
84     x = x + V(:,1:m)*y; % update
85     approximation
86     % compute residual
87     r = M*(b - A*x);
88     s(j+1) = norm(r);
89     error(itt+1) = s(j+1) / normb; % check
90     convergence
91     if ( error(itt+1) <= epsi )
92         break;
93     end
94     iter = iter+1;
95 end
96 iter = itt;

```

### Right preconditioner:

```

1 function [x, error, iter] = Right_PRECGMRES( A,b,y,M,m,itmax,epsi)
2 % PRECGMRES.m solves the preconditioner linear system
3 % The left preconditioner solve Mx = b
4 % The right preconditioner solve AMy = b, then x = My
5 % Using the above gmres.m, modify it to precgmres.m so as to be able to
6 % use
7 % a right preconditioner M that can be changed at each restart of gmres.
8 % input    A        REAL nonsymmetric positive definite matrix
9 %          x        REAL initial guess vector
10 %          b        REAL right hand side vector
11 %          M        REAL preconditioner matrix
12 %          numb_m   INTEGER number of iterations between restarts
13 %          max_it   INTEGER maximum number of iterations
14 %          tol      REAL error tolerance
15 % output   x        REAL solution vector
16 %          error    REAL error norm
17 %          iter     INTEGER number of iterations performed
18 % initialization
19
20 normb = norm(b);
21 if ( normb == 0.0 )
22     normb = 1.0;
23 end
24
25 % residual
26 r = b - A*M*y;
27 error(1) = norm(r)/normb;
28 if ( error(1) < epsi )
29     return;
30 end
31 [n,n] = size(A);
32 V(1:n,1:m+1) = zeros(n,m+1); % Vm+1=[Vm|qm+1]
33 H(1:m+1,1:m) = zeros(m+1,m); % Hessenberg matrix

```

```

34 cs(1:m) = zeros(m,1);
35 sn(1:m) = zeros(m,1);
36
37 e1      = zeros(n,1);           % basic vector
38 e1(1) = 1.0;
39 iter = 1;                       % step of iterator
40 itt = 1;
41 while iter <= itmax             % begin iteration
42     r = b - A*M*y;
43     V(:,1) = r/norm(r);
44     s = norm(r)*e1;
45     for j = 1:m                 % construct orthonormal
46         itt = itt + 1;         % basis using Gram-Schmidt
47         w = A*M*V(:,j);
48         for i = 1:j
49             H(i,j) = w'*V(:,i);
50             w = w - H(i,j)*V(:,i);
51         end
52         % size(H)
53         H(j+1,j) = norm(w);
54         V(:,j+1) = w/H(j+1,j);
55         % We transform the Hessenberg matrix H into a triangular matrix
by applying Givens rotation
56         for k = 1:j-1          % apply Givens
rotation
57             temp      = cs(k)*H(k,j) + sn(k)*H(k+1,j);
58             H(k+1,j) = -sn(k)*H(k,j) + cs(k)*H(k+1,j);
59             H(k,j)   = temp;
60         end
61         cs(j) = H(j,j)/sqrt(H(j,j)^2 + H(j+1,j)^2);
62         sn(j) = H(j+1,j)/sqrt(H(j,j)^2 + H(j+1,j)^2);
63         temp  = cs(j)*s(j);           % approximate
residual norm
64         s(j+1) = -sn(j)*s(j);
65         s(j)   = temp;
66         H(j,j) = cs(j)*H(j,j) + sn(j)*H(j+1,j);
67         H(j+1,j) = 0.0;
68         % Which also provides the error at iteration j+1
69
70         error(itt+1) = abs(s(j+1)) / normb;
71
72         if ( error(itt+1) <= epsi )   % update
approximation
73             z = H(1:j,1:j)\s(1:j);    % and exit
74             y = y + V(:,1:j)*z;
75             % error(i+1) = abs(s(i+1)) / bnrm2;
76             break;
77         end
78     end
79
80     if ( error(itt+1) <= epsi ), break, end
81     z = H(1:m,1:m)\s(1:m);
82 %     x = x + V*y;                     % update
approximation
83     y = y + V(:,1:m)*z;               % update
approximation
84     % compute residual
85     r = b - A*M*y;

```

```

86     s(j+1) = norm(r);
87     error(itt+1) = s(j+1) / normb;                                % check
88     convergence
89     if ( error(itt+1) <= epsi )
90         break;
91     end
92     iter = iter+1;
93 end
94 iter = itt;
95 x = M*y;

```

We using **right preconditioner** to apply equation  $Ax = b$  become  $AMy = b$  and  $x = My$ .

Where  $M = \text{diag}(\text{diag}(A))$

Method	error	iter	Time (seconds)
GMRES	1.006e-14	101	0.047792
Right preconditioner	2.3357e-13	101	0.060019

Table 1: Compare 2 method method

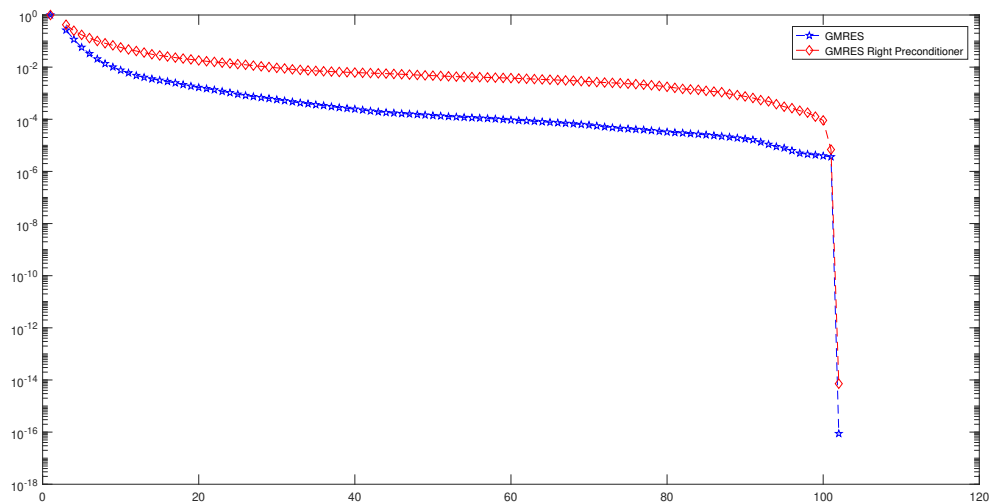


Figure 8: Comparison of GMRES and PRECGMRES

**Commend:** Although GMRES convergence faster, but PRECGMRES show us a better result (the approx is more close to exact solution)

### 3. Exercise 3

Look into the matlab function **ILU** that performs an ILU(0) approximation of  $A$ .  $[L, U] = \text{ILU}(A)$  Use this ILU(0) as a right preconditioner in `precgmres.m` to solve  $Ax = b$  as in TP1

#### Answer

Using **ILU** function, we have:

$$\begin{aligned}[L, U] &= \text{ilu}(A) \\ M &= \text{inv}(L * U)\end{aligned}$$

Matrix in worksheet 1:

Method	error	iter	Time (seconds)
GMRES	2.4282e-14	101	0.040265
Right preconditioner	8.4696e-15	2	0.03

Table 2: Compare 2 method method

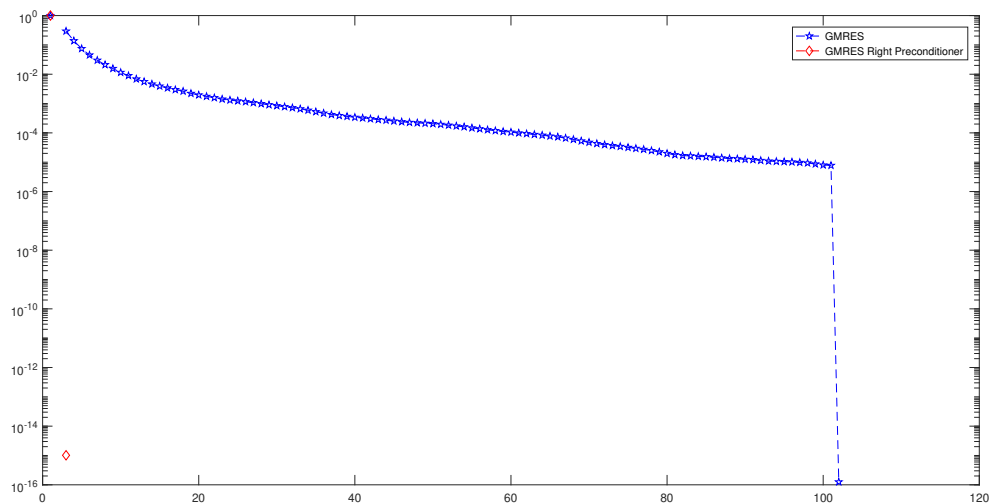


Figure 9: Comparison of GMRES and PRECGMRES

Matrix Market  
File 'hor131.mtx'

Method	error	iter	Time (seconds)
GMRES	3.2943e-06	427	0.35305
Right preconditioner	2.4151e-07	44	0.59493

Table 3: Compare 2 method method

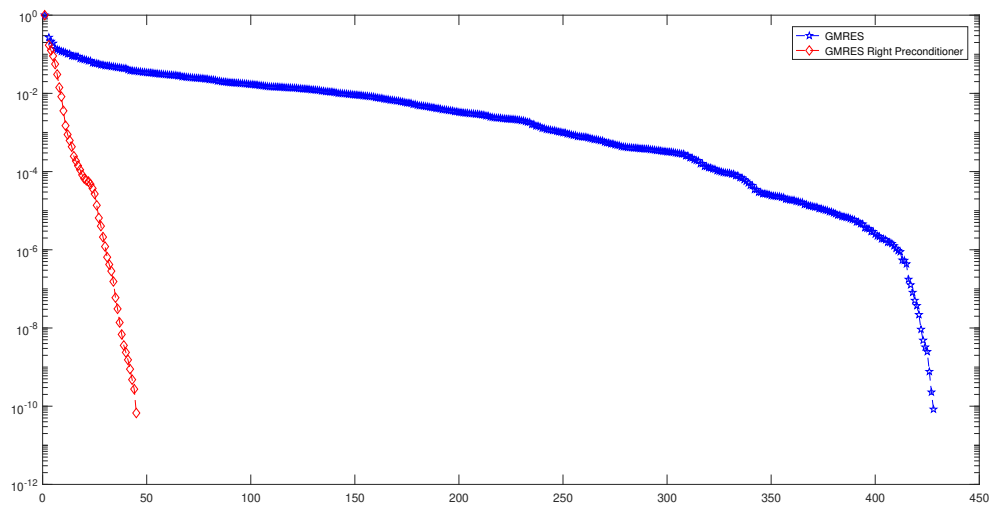


Figure 10: Comparison of GMRES and PRECGMRES

**File 'pde225.mtx'**

Method	error	iter	Time (seconds)
GMRES	1.0984e-09	76	0.033946
Right preconditioner	7.9933e-10	21	0.068774

Table 4: Compare 2 method method

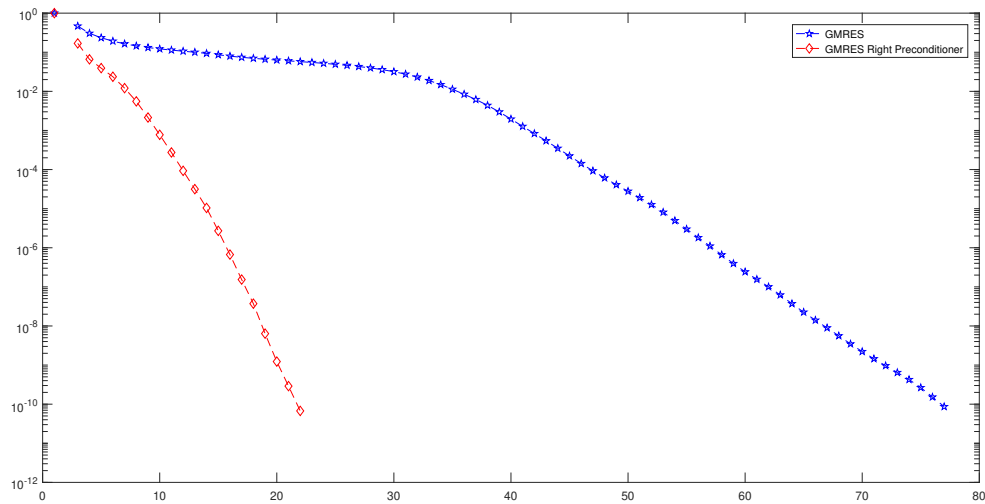


Figure 11: Comparison of GMRES and PRECGMRES

File 'saylr4.mtx'

Method	error	iter	Time (seconds)
GMRES	2.5793e-05	1684	41.2416
Right preconditioner	8.3633e-05	52	38.0274

Table 5: Compare 2 method method

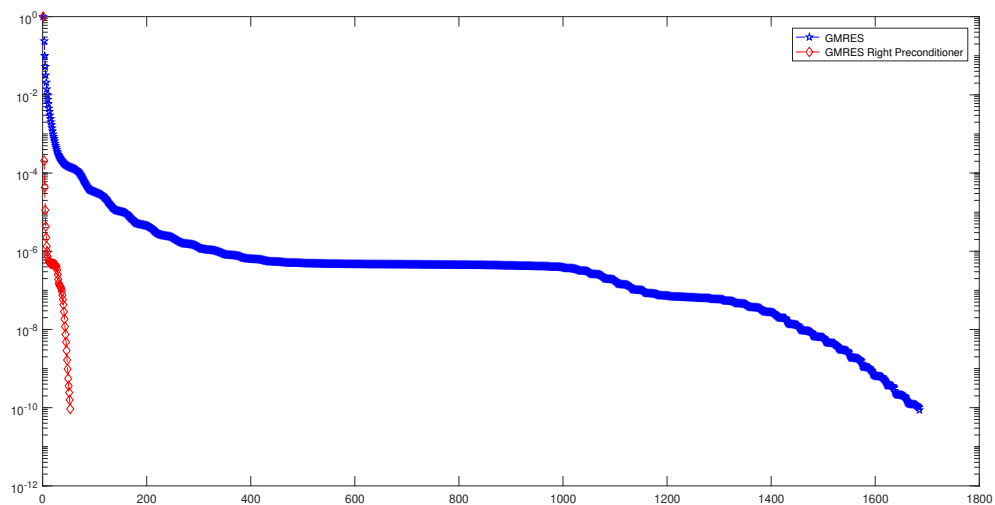


Figure 12: Comparison of GMRES and PRECGMRES

**File 'sherman4.mtx'**

Method	error	iter	Time (seconds)
GMRES	7.8963e-09	137	0.11605
Right preconditioner	4.3628e-09	38	0.61389

Table 6: Compare 2 method method

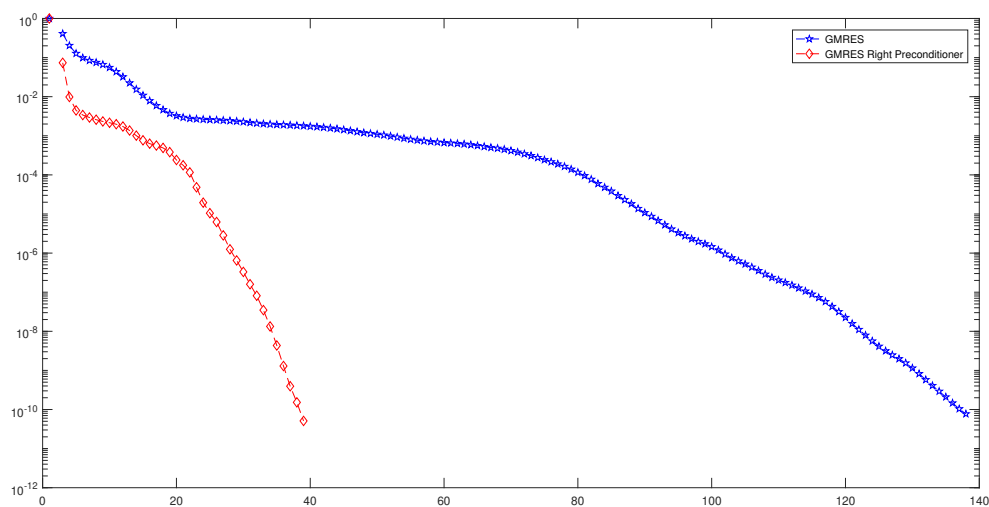


Figure 13: Comparison of GMRES and PRECGMRES

**File 'tub100.mtx'**

Method	error	iter	Time (seconds)
GMRES	4.4087e-13	101	0.046172
Right preconditioner	3.4059e-11	17	0.035072

Table 7: Compare 2 method method

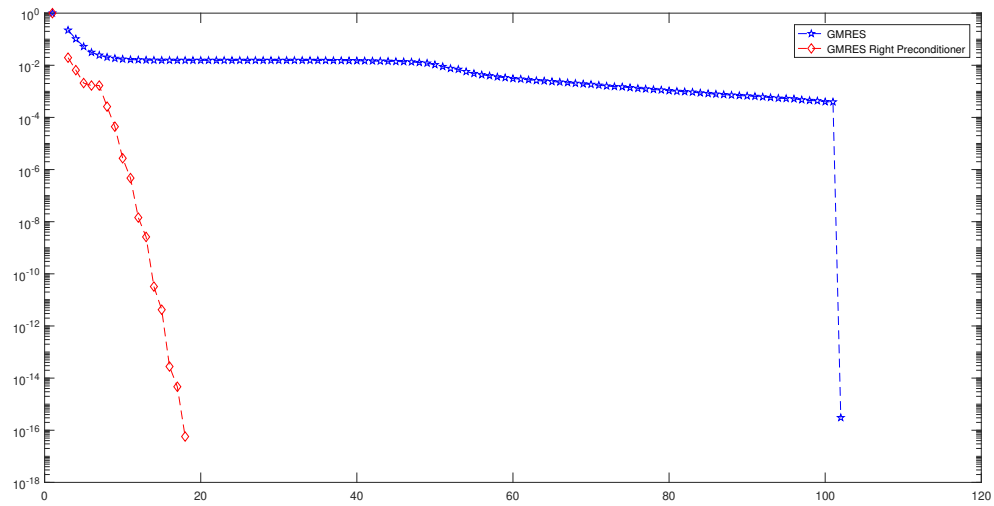


Figure 14: Comparison of GMRES and PRECGMRES