University of Science
Faculty of Mathematics and Computer Science

Worksheet #3

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 for the Laplace operator

c) The 2D Finite Difference Matrix for the Laplace operator

d) Matrices defined in Matrix Market.

1. Exercise 1

With the codes that Prof. Halpern sent you, compare the incomplete Cholewski with an **ILU(0)** decomposition of the lap2D matrix. Let A be a small size sparse symmetric matrix ($n = 10$) . Comment the incomplete Cholewski and **ILU(0)**, **ILU(1)**

Conjugate gradient with preconditioner

```
function [x,res,iter]=CGprecd(C,A,b,x0,tol,maxiter)

%
% Gradient conjugue preconditionne:
%                     [x,e]=CGprecd(A,b,x0,tol,maxiter) resout Ax=b en
%                     utilisant l'algorithme du gradient conjugue
%                     preconditionne ar une matrice C^{-1}.
%                     tol est la tolerance, x0 le vecteur initial,
   maxiter le
%                     nombre d'iteratios maximal, et e contient l'erreur
%                     relative a chaque iteration.
%                     la matrice A doit etre symetrique definie positive.
%


r=b-A*x0;
res(1)=norm(r);
z=C\r;
p=z;
k=1;

while res(k)>tol  && k<maxiter

  Ap=A*p;                         % pour avoir seulement un produit
```

```
  pAp=p'*Ap;                % matrice vecteur par iteration
  num=z'*r;
  alpha=num/pAp;

  x=x0+alpha*p;
  x0=x;

  r=r-alpha*Ap;
  z=C\r;
  beta=z'*r/num;
  p=z+beta*p;
  res(k+1)= norm(r);
  k=k+1;

end

iter = k;
```

Main input:

```
n = 10; N = n*n;
A = lap2d(n,n);
xex = rand(N,1);
b = A*xex;
x0 = zeros(N,1);
tol = 10^(-12);
maxiter = N;
```

Compare number of iteration and error of difference method:

Method	Number of iterations	Error
Conjugate Gradient	45	2.7898e-15
CG preconditioning with Gauss Seidel	37	2.6953e-15
CG preconditioning with SSOR	36	1.7973e-15
CG preconditioning with Incomplete Cholewski	23	3.106e-15
CG preconditioning with L+U-speye(size(A))	24	2.476e-15
CG preconditioning with LU(0) decomposition	23	1.7581e-15
CG preconditioning with LU(1) decomposition	57	2.7105e-15

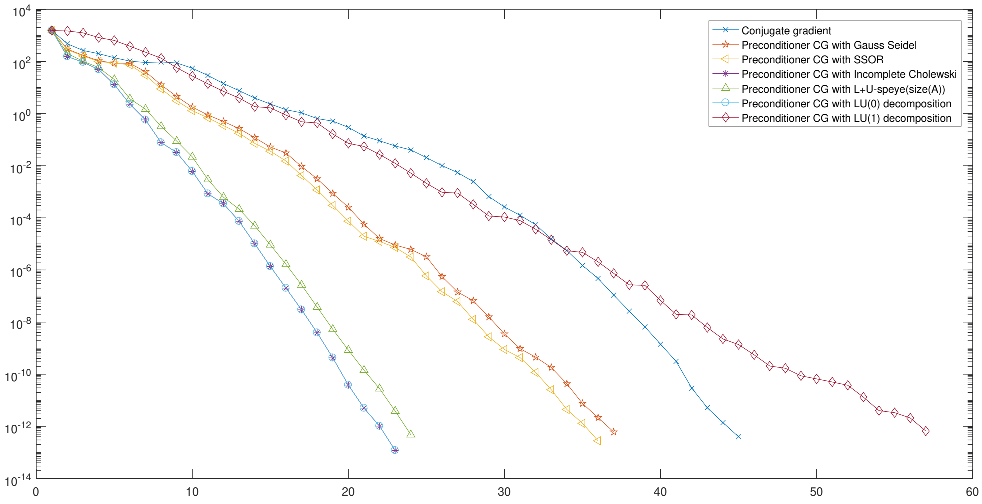


Figure 1: Comparison of value of residual of different method with *semilogy*

Using matrix of 1D Finite Difference Matrix for the Laplace operator

Main input:

```
n = 50;
A = lap1d(n);
```

Compare number of iteration and error of difference method:

Method	Number of iterations	Error
Conjugate Gradient	50	0.017617
CG preconditioning with Gauss Seidel	47	5.3029e-15
CG preconditioning with SSOR	46	6.6699e-15
CG preconditioning with Incomplete Cholewski	3	4.4365e-15
CG preconditioning with L+U-speye(size(A))	21	2.847e-14
CG preconditioning with LU(0) decomposition	3	4.4365e-15
CG preconditioning with LU(1) decomposition	50	3.3842e-09

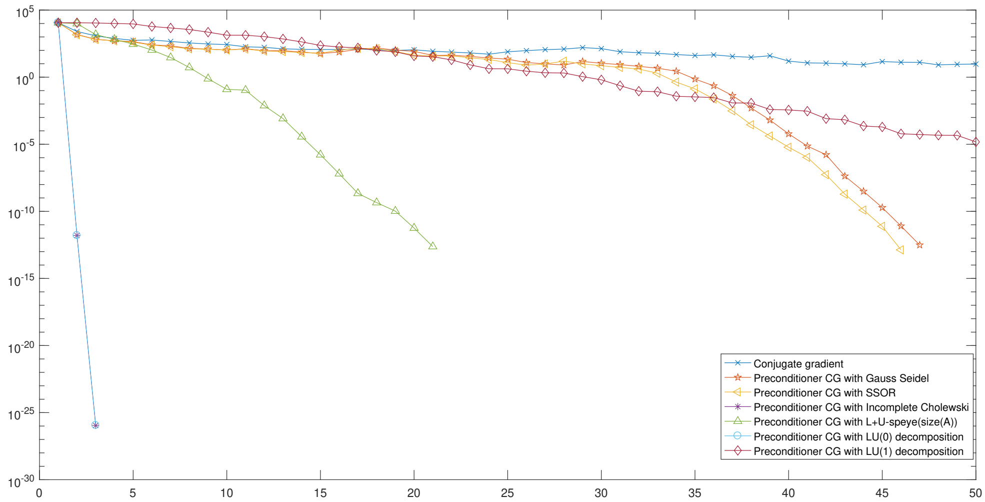


Figure 2: Comparison of value of residual of different method with *semilogy*

Using matrix defined in worksheet 1

Main input:

```
N = 100;
beta = 0.9;
alp = 2;
A = createMatrix(N,beta,alp);
A = A*A';
```

Compare number of iteration and error of difference method:

Method	Number of iterations	Error
Conjugate Gradient	100	0.25986
CG preconditioning with Gauss Seidel	16	1.4056e-13
CG preconditioning with SSOR	15	6.9301e-14
CG preconditioning with Incomplete Cholewski	3	3.2393e-14
CG preconditioning with L+U-speye(size(A))	100	4.9069e-05
CG preconditioning with LU(0) decomposition	3	1.9305e-14
CG preconditioning with LU(1) decomposition	100	1.6991e-05

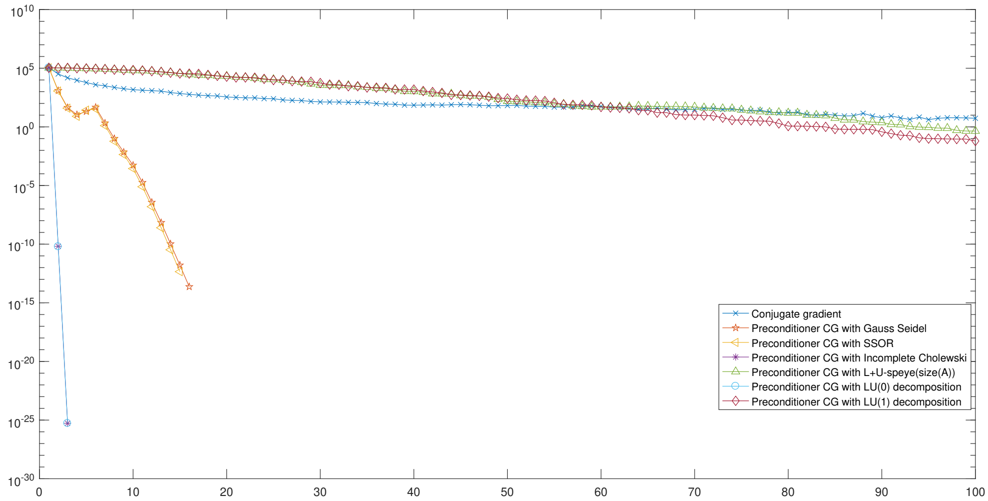


Figure 3: Comparison of value of residual of different method with *semilogy*

Using matrix from Matrix Market

1 FILE : 'hor131.mtx'

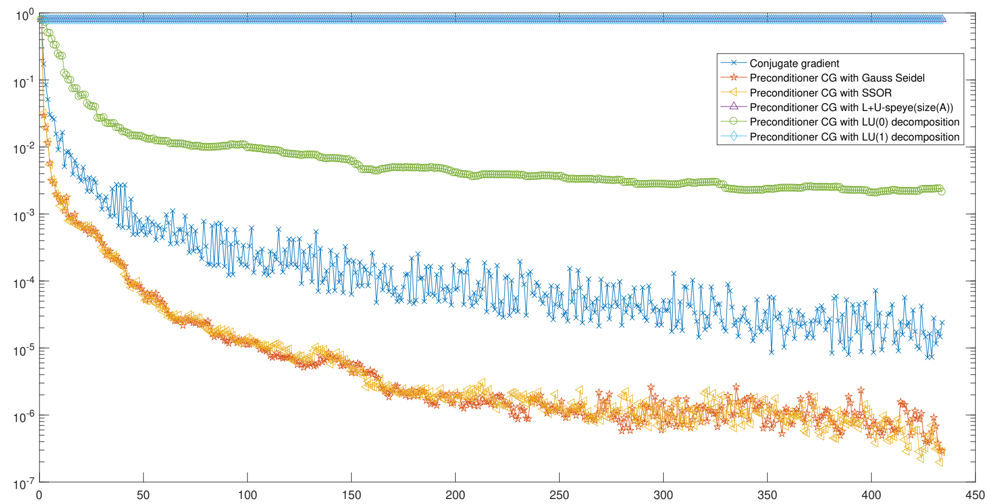


Figure 4: Comparison of value of residual of different method with *semilogy*

2 FILE : 'pde225.mtx'
Compare number of iteration and error of difference method:

Method	Number of iterations	Error
Conjugate Gradient	168	9.6661e-14
CG preconditioning with Gauss Seidel	87	7.0933e-14
CG preconditioning with SSOR	82	9.8406e-14
CG preconditioning with Incomplete Cholewski	32	2.1736e-13
CG preconditioning with L+U-speye(size(A))	47	9.8092e-14
CG preconditioning with LU(0) decomposition	32	2.0082e-13
CG preconditioning with LU(1) decomposition	225	4.6775

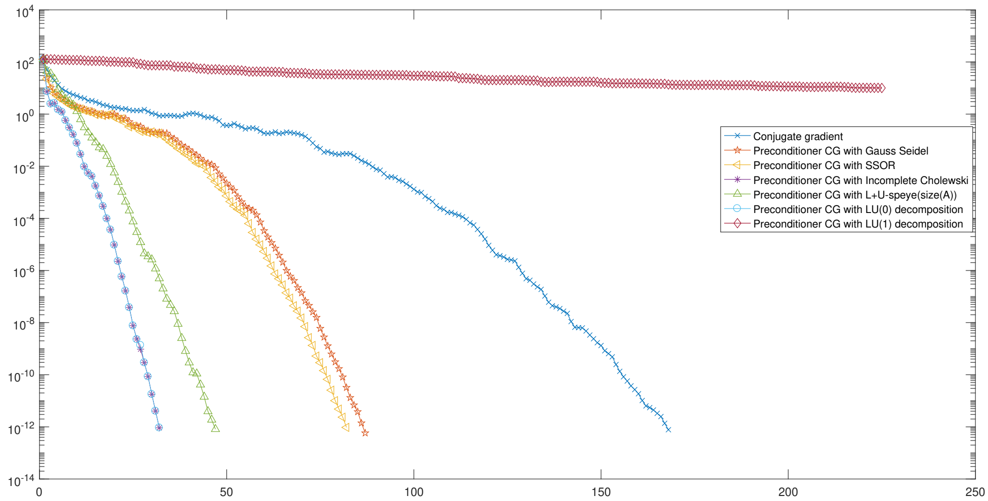


Figure 5: Comparison of value of residual of different method with *semilogy*

3 FILE : 'sherman4.mtx'

Method	Number of iterations	Error
Conjugate Gradient	1041	1.3094e-12
CG preconditioning with Gauss Seidel	292	5.677e-11
CG preconditioning with SSOR	283	4.667e-11
CG preconditioning with L+U-speye(size(A))	1104	16.3103
CG preconditioning with LU(0) decomposition	1104	2.9827e-07

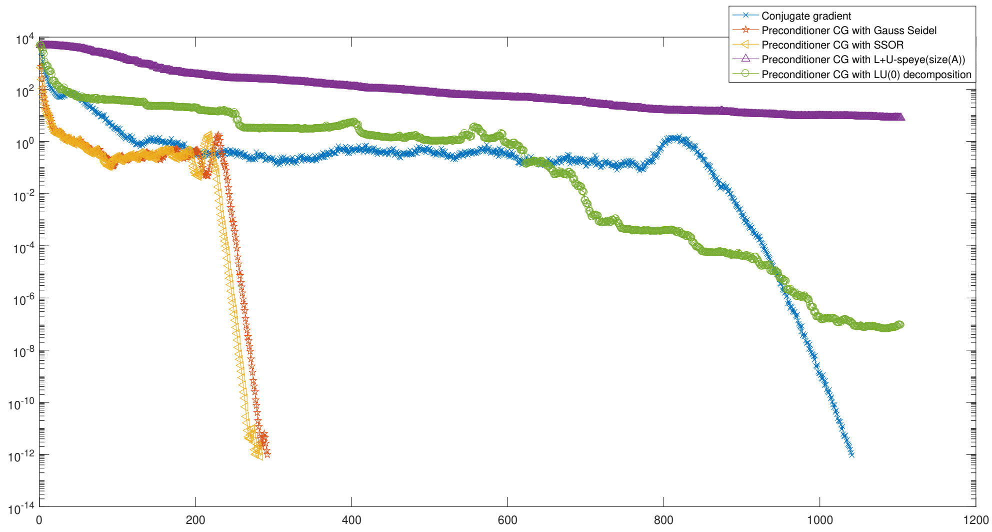


Figure 6: Comparison of value of residual of different method with *semilogy*

4 FILE : 'tub1000.mtx'

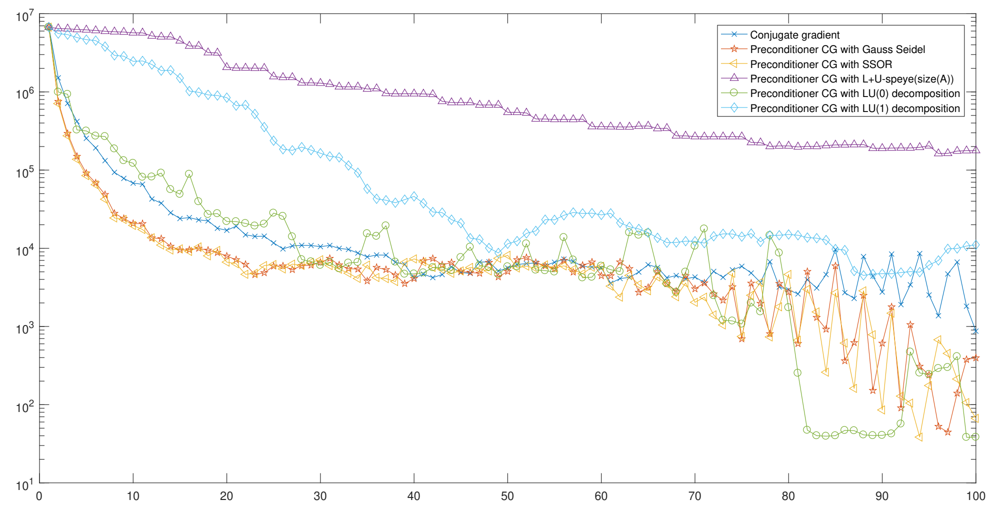


Figure 7: Comparison of value of residual of different method with *semilogy*

Exercise 2 and Exercise 3 already done in ws2.

2. Exercise 4

Look into the matlab function schur that performs a Schur decomposition of a matrix. (help schur)

Schur decomposition:

$[U, T] = \text{schur}(X)$ produces a *quasitriangular* **Schur** matrix T and a *unitary* matrix U so that X = U*T*U' and U'*U = EYE(SIZE(U)). X must be square.

T = schur(X) returns just the Schur matrix T.

If X is real, two different decompositions are available. schur(X,'real') has the real eigenvalues on the diagonal and the complex eigenvalues in 2-by-2 blocks on the diagonal. schur(X,'complex') is triangular and is complex if X has complex eigenvalues. schur(X,'real') is the default.

If X is complex, the complex Schur form is returned in matrix T. The complex Schur form is upper triangular with the eigenvalues of X on the diagonal. The second input is ignored in this case.

3. Exercise 5

Following the algorithm below, modify precgmres.m to obtain a function that performs a Deflation Preconditioned Gmres

Algorithm : DEFLGMRES

Require: Choose x_0, $M = I_n$

1: $r_0 = b - Ax$, $\beta = \|r_0\|$, $v_1 := r_0/\beta$

2: Generate the Arnoldi basis applied to AM^{-1}

and the associated Hessenberg matrix $\tilde{H}_m$ starting with v_1

3: Compute y_m which minimises $\left\|\beta e_1 - \tilde{H}_m y\right\|$ and $x_m = x_0 + M^{-1}V_m y_m$

4: If convergence Stop, else set ;

$x_0 = x_m$

Compute 1 Schur vectors of H_m noted S_l;

Compute the approximation of $|\lambda_n|$;

Orthogonalize $V_m S_l$ against U;

Increase U with $V_m S_l$;

$T = U^T A U$;

$M^{-1} = I_n + U\left(|\lambda_n|\, T^{-1} - I_r\right)U^T$;

Go To 1 ;

```
%**********************************************
function[x,error,iter, itv] = DEFLGMRESR(A,b,x,M,m,l,itmax,epsi)
%**********************************************
% DEFLGMRESR.m solves the linear system Ax=b
% using the Generalized Minimal residual ( GMRESm ) method with restarts
    .
%  With Right preconditioning and Schur decomposition
%
% input    A         REAL nonsymmetric positive definite matrix
%          x         REAL initial guess vector
%          bb         REAL right hand side vector
%          m         INTEGER number of iterations between restarts
%          l
%          itmax   INTEGER maximum number of iterations
%          epsi       REAL error tolerance
%          M          Right Preconditioner
```

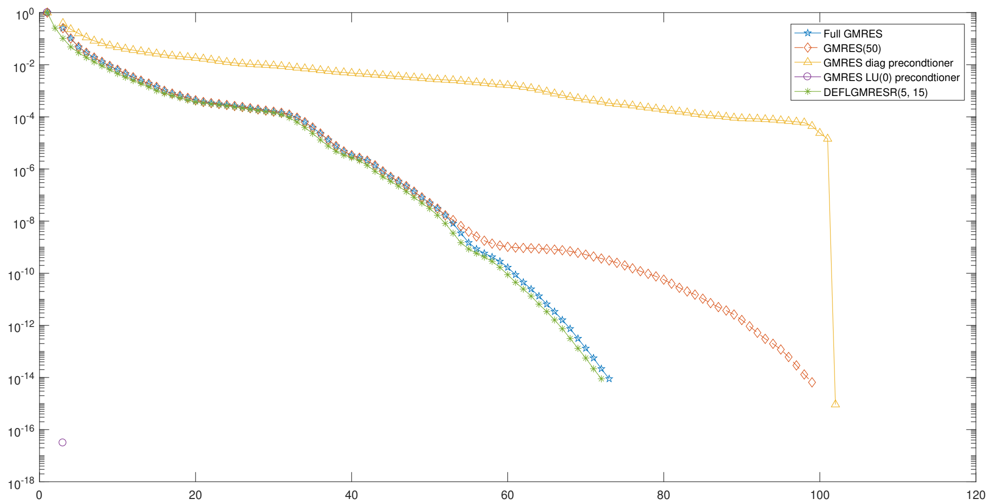
```
%
% output   x        REAL solution vector
%          error    REAL error norm
%          iter     INTEGER number of iterations performed
%          itv Total number of iterations


% initialization
normb = norm( b );
if  ( normb == 0.0 ),
    normb = 1.0;
end
% invM = inv(M);
% residual
r = b - A*(M\x);
error(1) = norm( r )/normb;

if ( error(1) < epsi )
    return;
end

n= size(A,1);
V(1:n,1:m+1) = zeros(n,m+1);
H(1:m+1,1:m) = zeros(m+1,m);   % Hessenberg matrix
U = [];
cs(1:m) = zeros(m,1);
sn(1:m) = zeros(m,1);
e1    = zeros(n,1); % basic vector
e1(1) = 1.0;
iter=1;  % step of iterator
itv= 0 ; % Total iterations
while iter <= itmax                              % begin iteration
      r = b - A*(M\x);
      V(:,1)=r/norm(r);
      s = norm(r)*e1;
   for j = 1:m                                   % construct orthonormal
         itv = itv +1 ;                          % basis using Gram-
   Schmidt
       w = A*(M\V(:,j));
     for i = 1:j
         H(i,j)= w'*V(:,i);
         w = w - H(i,j)*V(:,i);
       end
     H(j+1,j) = norm(w);
     V(:,j+1) = w/H(j+1,j);
   % We tranform the Hessenberg matrix H into a triangular matrix by
   applying Givens rotation
     for i = 1:j-1
           temp     =  cs(i)*H(i,j) + sn(i)*H(i+1,j);
           H(i+1,j) = -sn(i)*H(i,j) + cs(i)*H(i+1,j);
           H(i,j)   = temp;
       end
     [cs(j),sn(j)] = rotmat( H(j,j), H(j+1,j) ); % form i-th rotation
   matrix
       temp   = cs(j)*s(j);
       s(j+1) = -sn(j)*s(j);
     s(j) = temp;
       H(j,j) = cs(j)*H(j,j) + sn(j)*H(j+1,j);
```

```
        H(j+1,j) = 0.0;  %eliminate H(j+1,j)
      error(j+1) = abs(s(j+1)) / normb;
      if ( error(j+1) <= epsi )                      % update approximation
         y = H(1:j,1:j) \ s(1:j);                    % and exit
           x = x + V(:,1:j)*y;
         % error(i+1) = abs(s(i+1)) / bnrm2;
         break;
      end
    end

    if ( error(j+1) <= epsi ), break, end
    y = H(1:m,1:m)\s(1:m);
%     x = x + V*y;                                      % update approximation
    x = x + V(:,1:j)*y;                                 % update approximation

    % Compute Shur vectors of H noted Sl, l = 2
    [eigvals] = eig(H(1:m,1:m));
    eigval_new = [];
    for i = 1:length(eigvals)
        if (eigvals(i) ~= min(eigvals))
            eigval_new = [eigval_new;eigvals(i)];
        end
    end
    min_eigval = [min(eigvals); min(eigval_new)];
    [U_Hess,T] = schur(H(1:m,1:m));
    Sl = zeros(m,l);
    for i = 1:l
        for jj = 1:size(T,2)
            if T(jj,jj) == min_eigval(i)
                Sl(:,i) = T(:,jj);
            end
        end
    end
    k = size(U,2);
    W = V(:,1:m)*Sl;
    U = [U W];
    % Orthogonalize VmSl against U
    U(:,1) = U(:,1) / sqrt(U(:,1)'*U(:,1));
    for i = 2:k
        for ii = 1:k-1
            U(:,i) = U(:,i) - ( U(:,ii)'*U(:,i) )/( U(:,ii)'*U(:,ii) )*U(:,ii);
        end
        U(:,i) = U(:,i) / sqrt(U(:,i)'*U(:,i));
    end
   T = U'*A*U;
   invM = eye(n,n) + U*(abs(max(eig(full(A))))*inv(T) - eye(size(T,1)))*U';

    % compute residual
    r = b - A*(M\x);
    s(j+1) = norm(r);
    error(j+1) = s(j+1) / normb;                           % check convergence
    if ( error(j+1) <= epsi )
        break;
```

```
    end
    iter=iter+1;
end
x = M\x;     % true solution after preconditioning
```

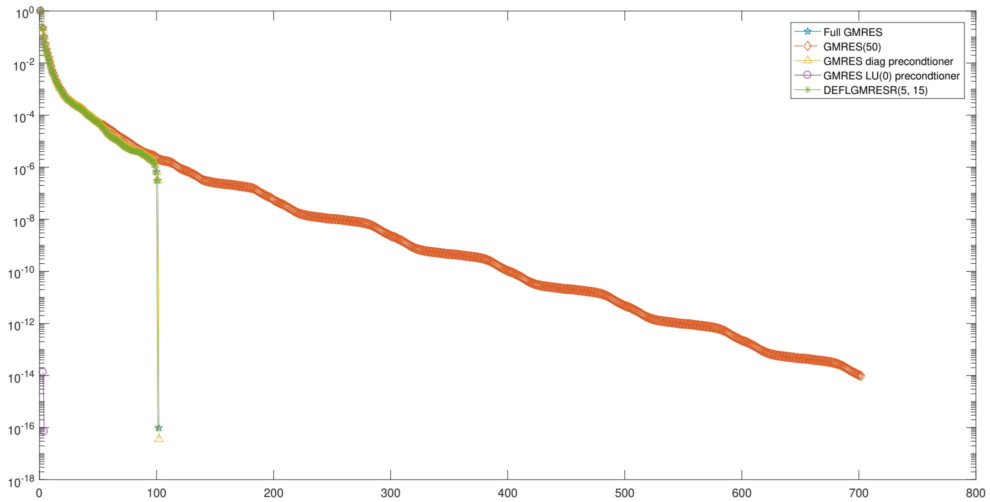
Matrix in worksheet 1:

Method	error	iter	Time (seconds)
Full GMRES	74	6.925e-12	0.044242
GMRES(50)	101	1.6963e-11	0.029979
GMRES diag precondtioner	101	2.6924e-12	0.044937
GMRES LU(0) precondtioner	2	2.8741e-13	0.0097826
DEFLGMRESR(5, 15)	72	6.925e-12	0.048682



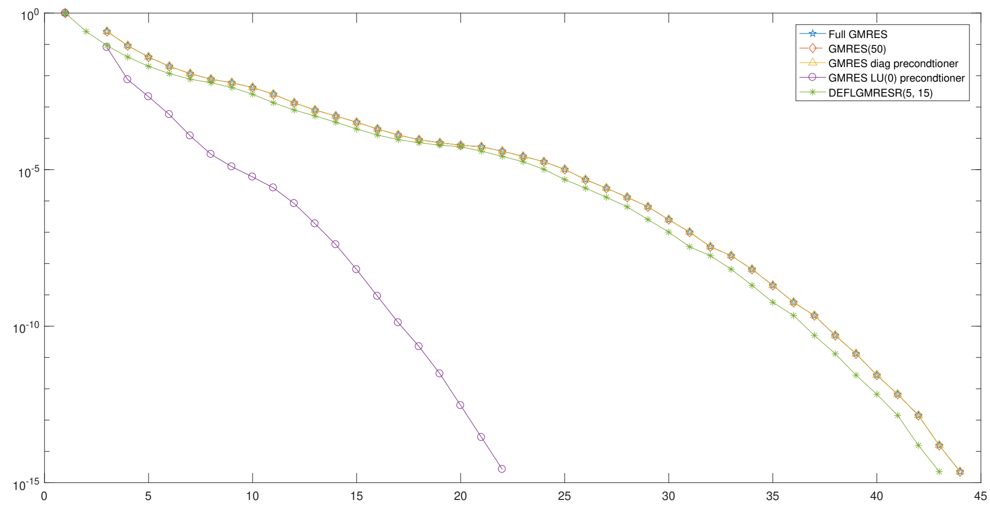
Matrix Laplace 1d:

Method	error	iter	Time (seconds)
Full GMRES	102	2.7435e-11	0.053413
GMRES(50)	702	2.3873e-09	0.090297
GMRES diag precondtioner	101	2.6785e-11	0.041346
GMRES LU(0) precondtioner	3	2.325e-10	0.0087148
DEFLGMRESR(5, 15)	100	2.7435e-11	0.065002



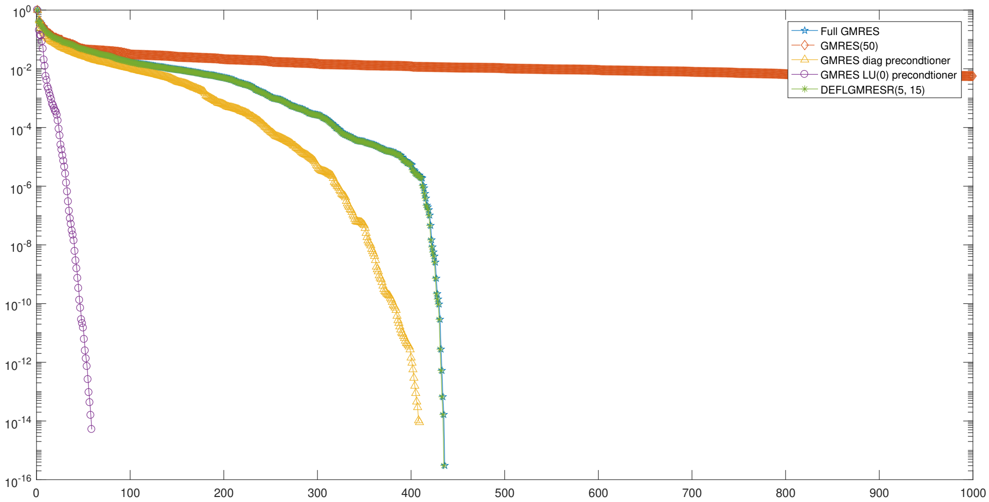
Matrix Laplace 2d:

Method	error	iter	Time (seconds)
Full GMRES	44	1.0666e-12	0.034555
GMRES(50)	44	1.0666e-12	0.010509
GMRES diag precondtioner	43	1.1324e-12	0.034507
GMRES LU(0) precondtioner	21	2.161e-12	0.019295
DEFLGMRESR(5, 15)	42	1.0666e-12	0.041744



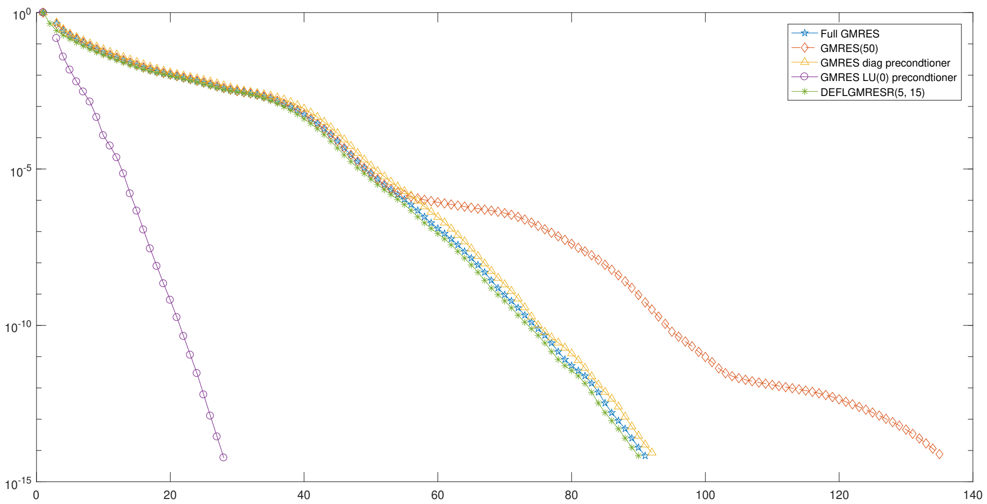
Matrix Market File 'hor131.mtx'

Method	error	iter	Time (seconds)
Full GMRES	436	2.0039e-11	0.44593
GMRES(50)	5002	187.66	0.59435
GMRES diag precondtioner	408	6.2006e-09	0.39461
GMRES LU(0) precondtioner	58	7.7192e-10	0.97
DEFLGMRESR(5, 15)	434	2.0039e-11	0.48991



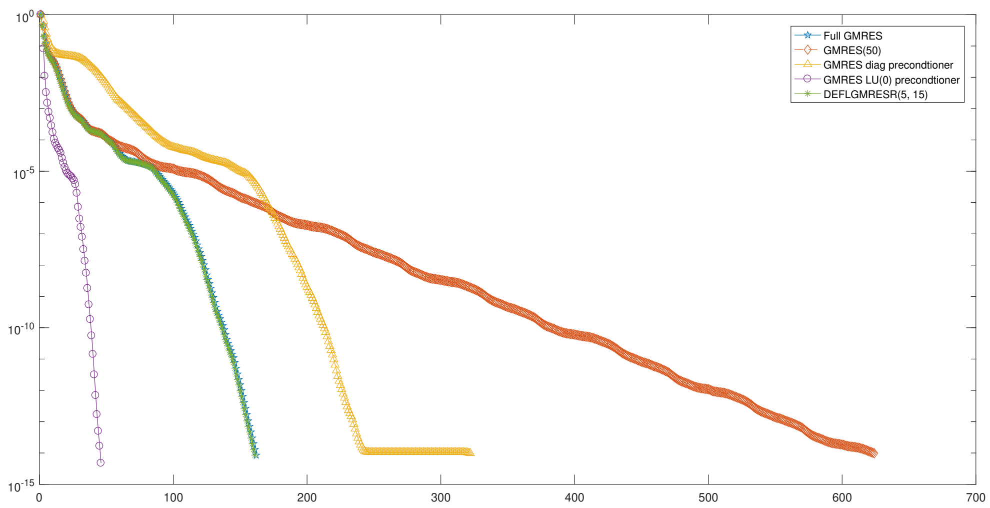
File 'pde225.mtx'

Method	error	iter	Time (seconds)
Full GMRES	91	4.9422e-12	0.063074
GMRES(50)	135	1.0202e-11	0.038621
GMRES diag precondtioner	91	1.0492e-11	0.048552
GMRES LU(0) precondtioner	27	8.1211e-12	0.060863
DEFLGMRESR(5, 15)	89	4.9422e-12	0.081825



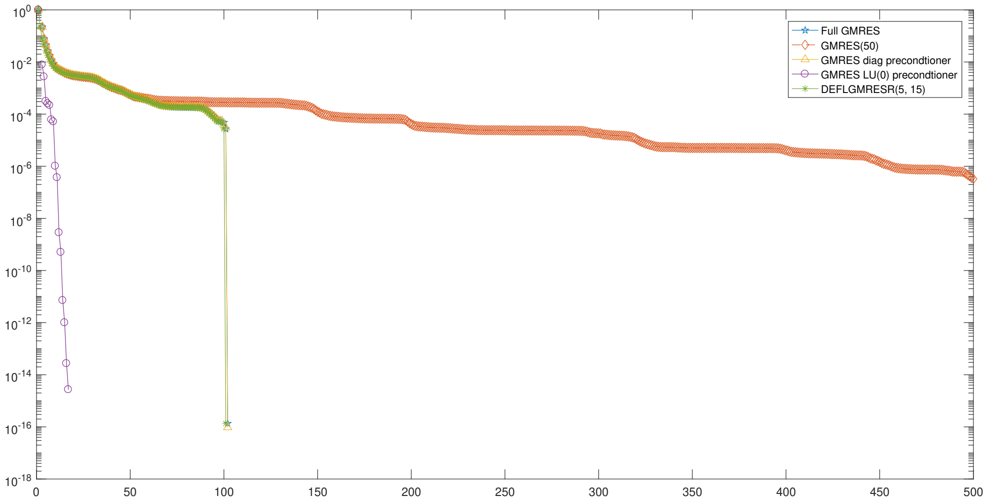
File 'sherman4.mtx'

Method	error	iter	Time (seconds)
Full GMRES	162	5.7276e-11	0.15503
GMRES(50)	624	3.3599e-09	0.10939
GMRES diag precondtioner	321	1.1896e-10	0.35844
GMRES LU(0) precondtioner	45	5.8692e-11	0.70376
DEFLGMRESR(5, 15)	160	5.7276e-11	0.14505



File 'tub100.mtx'

Method	error	iter	Time (seconds)
Full GMRES	102	1.9944e-11	0.041509
GMRES(50)	5002	2.4806e-07	0.46914
GMRES diag precondtioner	101	1.2226e-10	0.048416
GMRES LU(0) precondtioner	16	4.3016e-10	0.015504
DEFLGMRESR(5, 15)	100	1.9944e-11	0.070726



Commend: Those method give us solution to solve $Ax = b$, where A is not necessary symmetric. Among all, GMRES method using LU(0) as precondtioner give the best behaviour.