Exercise 9 – Solutions

1. The arithmetic complexities of these three algorithms:

   - GMRES: After N iteration steps:
     N multiplications with A: $N \times \texttt{nnz(A)}$ flops
     N solves with the preconditioner: $\mathcal{O}(N \times \texttt{nnz(A)})$ flops
     Orthogonalizations: $\sum_{j=1}^{N} j \approx \frac{1}{2}N^2$, i.e. $\mathcal{O}(N^2 n)$ flops, if A is $n \times n$.
     Memory: $N \times n$ for the Krylov basis, besides A and the preconditioner M.
   - GMRES(m): After $N = \texttt{iter(1)} \times \texttt{restart} + \texttt{iter(2)}$ iteration steps:
     Multiplications with A/solves with the preconditioner as with GMRES.
     Orthogonalizations: $\texttt{iter(1)} \times \frac{\texttt{restart}^2}{2} + \frac{\texttt{iter(2)}^2}{2}$
     Memory: $N \times \texttt{restart}$ for the Krylov basis, besides A and M.
   - MINRES: After N iteration steps:
     Multiplications with A/solves with the preconditioner as with GMRES.
     Orthogonalizations: $3N$.
     Memory: $4N$ for the few vectors of the Krylov basis, besides A and M.

   The most expensive portions of the GMRES/MINRES solver without preconditioning are the matrix-vector multiplication and the Gram–Schmidt orthogonalization. If there is preconditioning, then solving with the preconditioner, i.e.,$M\boldsymbol{z}_k = \boldsymbol{r}_k$, must also be added to get the total arithmetic complexity.

2. The vector $\boldsymbol{r}_0 = \boldsymbol{b}$ must be 'touched by just three eigenvectors. Two (simple) solutions are as follows:

   (1) You can generate a matrix that has just three eigenvalues.

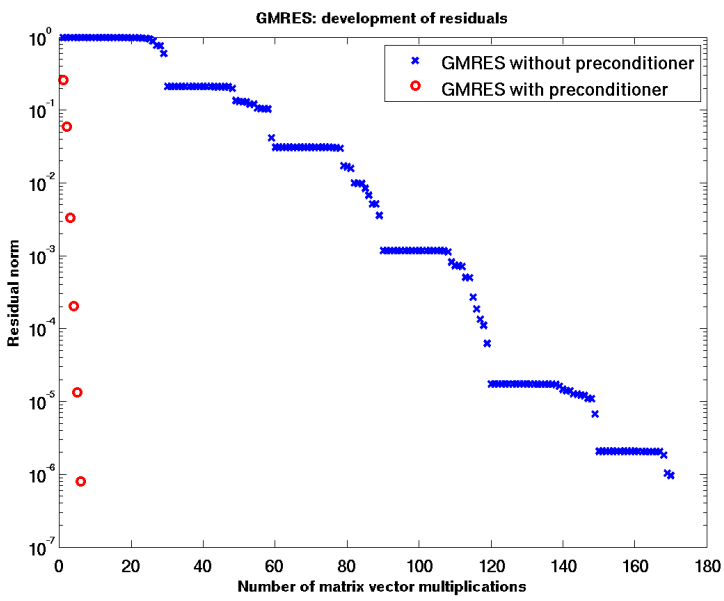
   (2) You can generate a diagonal matrix with all different diagonal elements (= eigenvalues) and have the vector $\boldsymbol{b}$ just 3 non-zero elements.

3. The following MATLAB script solves this problem. Notice that you need to set `restart=30` at least in order that the unpreconditioned GMRES algorithm can solve this problem. If `restart` is smaller then the residual norm is hardly reduced until `maxit` iterations are completed.

```
clear all
A = mmread('fs_183_6.mtx');
```

```
tol = 1e-6; n = length(A(:,1)); b = ones(n,1);
restart = 30;
maxit = 30;

[x1,flag1,relres,iter,resvec1] = gmres(A,b,restart,tol,maxit);
mvm1 = (iter(1)-1)*restart+iter(2);
fprintf('flag1 = %d, mvm1 = %d\n',flag1,mvm1)

%[L,U] = ilu(A);
L=tril(A);
[x2,flag2,relres,iter,resvec2] = gmres(A,b,restart,tol,maxit,L);
mvm2 = (iter(1)-1)*restart+iter(2);
fprintf('flag2 = %d, mvm2 = %d\n',flag2,mvm2)

close all

semilogy(0:mvm1,resvec1/resvec1(1),'bx','Linewidth',2)
hold on
semilogy(0:mvm2,resvec2/resvec2(1),'ro','Linewidth',2)
title('GMRES: development of residuals','FontWeight','demi')
xlabel('Number of matrix vector multiplications','FontWeight','demi')
ylabel('Residual norm','FontWeight','demi')
legend('GMRES without preconditioner',...
       'GMRES with Gauss-Seidel preconditioner','Location','SouthEast')

hold off
```

The figure shows how much preconditioning can help.



4. `run_gmres_with_matrix.m`: `gmres` with Gauss–Seidel preconditioner: $M = D + L$.

   `run_gmres_with_function_handle.m`: `gmres` with Jacobi preconditioner: $M = D$.

   We explore the impact of the restarting technique on the performance of the `gmres` solver in `run_gmres_restart_tech.m`

```
% FEM17- Exercise9-4
N=30000; A=2; B=3; C=8; D=3.5; E=4.5;
% (4a) Generate a sample Toeppen matrix
P = gallery('toeppen',N, A, B, C, D, E);
% (4b) Right hand side of the linear equation
b=rand(N,1);

tol = 1.0e-9;   % Tolerance
maxit = 1000; % Maximum iterations
restart=[1,2,4,6,8,16,32];

total_iter=[];
telapsed=[];
for i=1:7
   % (4c) total number of iterations = (iter(1)-1) x restart + iter(2)
  [total_iter(i),telapsed(i)] = restart_tech(restart(i),P,b,tol,maxit);
end
restart
total_iter
telapsed
```

   In `restart_tech.m` the statement `[x,flag,relres,iter] = gmres(P, b, restart, tol, maxit)` returns both the outer and inner iteration numbers at which `x` was computed, where $0 \leq$ `iter(1)` $\leq$ `maxit` and $0 \leq$ `iter(2)` $\leq$ `restart`.

   On my Laptop (2.8 GHz Intel Core i7) I get the following numbers

`restart` =	1	2	4	6	8	16	32
`total_iter` =	290	182	173	151	143	135	133
t_{elapsed} =	0.3464	0.1922	0.1791	0.1561	0.1751	0.2082	0.3033

   - In Figure 1, the number of iterations of GMRES method decreases when restart is increased, but the rate of decreasing is getting smaller and smaller, and finally the number of iterations keeps constant at a munimun value(92 in this example). explanation: When the restart value grows, so does the Krylov searching space. And the method tends to converge within fewer steps. However, this is at the cost of increased memory consumption, which is directly proportional to restart.
   - In Figure 2, the execution time of GMRES method first reduces when restart increases. But after it reaches a minimum value, it starts to increase.

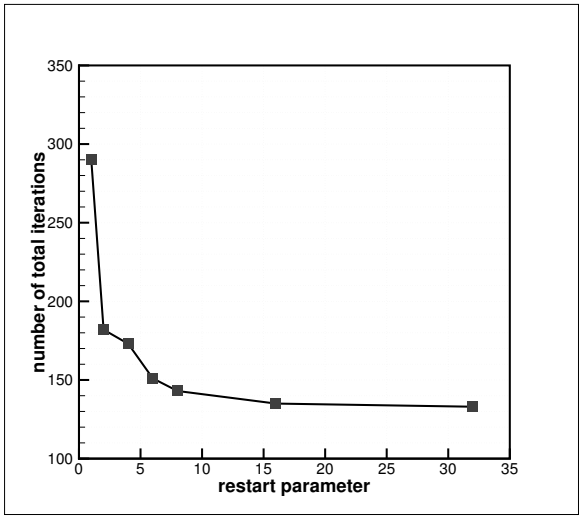


Figure 1: Number of iterations vs. restart of GMRES method

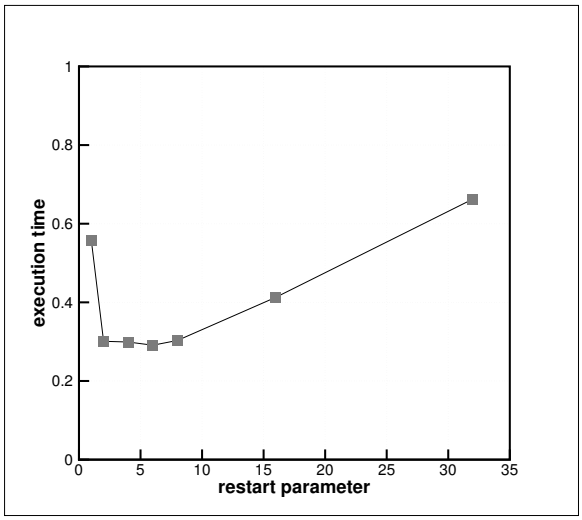


Figure 2: Execution time vs. restart of GMRES method

Explanation: In the `gmres` solver, the most expensive portions are routines of Matrix-Vector multiplication and the Gram–Schmidt orthogonalization. The complexity per iteration step is about $O_{\mathrm{MV}} = O(\mathtt{m}\times\mathtt{restart})$ for the Matrix-Vector multiplication and $O_{\mathrm{ortho}} = O(\frac{\mathtt{restart}^2}{2}\times N)$ for the Gram–Schmidt orthogonalization. The execution time can be evaluated by *number of iterations* $\times(O_{\mathrm{MV}}+O_{\mathrm{ortho}})\times t_{\mathrm{mul}}$, where t_{mul} is the time for a scalar multiplication. When restart starts from 1, as the restart rises, the number of iterations drops sharply as presented in Figure 1 in spite of the cost for the orthogonalization and matrix-vector multiplication routines increases in a moderate rate, therefore the execution time decreases. However, when restart continues to grow, the number of iterations just drops a little or even keeps constant at last, while on the other hand the cost for the orthogonalization and matrix-vector multiplication routines still grows at a moderate rate. Therefore the execution time increases. According to the experimental results on my machine, the optimal restart value is 6, that minimizes the execution time.

5. The solution of this problem is given in `ex9_5.m`. The experimental results are again machine dependent. In general, if a good preconditioner is applied to the iterative solver, the number of iterations and the execution time will both reduce. Preconditioning is an essential part for the success of the iterative solvers. You will learn more details of preconditioning techniques in the future lectures.