Advanced Numerical Analysis Homework 3

Keerthana C J

May 28, 2020

1 Exercise 2

As a part of Exercise 1, the GMRES algorithm has been implemented and now, it is compared along with the matlab inbuilt function. The matrix 'mat13041.rig' was solved using both the algorithms and the convergence profiles for both the functions are shown in the figure 1. The matlab function takes about 527 iterations & 9.7s while the mygmres takes 511 iterations & 14.5s.

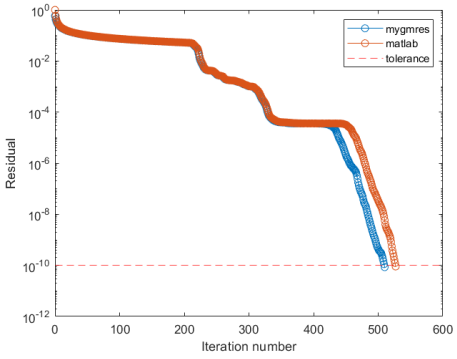

Figure 1: Implementation of GMRES in MATLAB and comparison with inbuilt GMRES of MATLAB

2 Exercise 3

The preconditioned GMRES has been implemented in the function myprecgmres.m and compared with matlab inbuilt preconditioned gmres() function. Three types of preconditioning has been done namely the left, right and the split. An ILU preconditioner has been used for solving the problem same as that of Exercise 2.

The results are given in the table of figure 2 and the convergence profiles are shown in the figure 3.

The right preconditioning uses the real residual while the other preconditioning techniques work with the preconditioned residual.Hence the actual residual can be greater than the tolerance. All the preconditioning techniques takes almost the same number of iterations.

3 Exercise 4

The restart is generally used to minimize the memory space requirements needed for the GMRES method. Although, the restart can significantly slow down the convergence or can lead to divergence. This exercise focuses on the restart parameter and the results are shown in the table of figure 4. As the restart value increases, the number of iterations and the time reduces. After a threshold, the iterations remain constant. From the convergence profiles in figure 5, we see that for very low restart values, the solution diverges.

```
>> notebook2

T =

  6×5 table
```

preconditioner	iterations	timings	real_residuals	preconditioned_residuals
"myprecgmres_left"	93	12.837	4.0978e-11	6.2631e-11
"matlab_left"	93	0.45408	4.0985e-11	6.2665e-11
"myprecgmres_right"	92	12.719	9.2566e-11	9.2566e-11
"matlab_right"	92	0.43921	9.2536e-11	9.2536e-11
"myprecgmres_split"	93	14.253	3.7426e-11	7.7861e-11
"matlab_split"	93	0.38793	3.7427e-11	7.7852e-11

Figure 2: Results of Implemented GMRES function and Matlab Inbuilt GMRES function

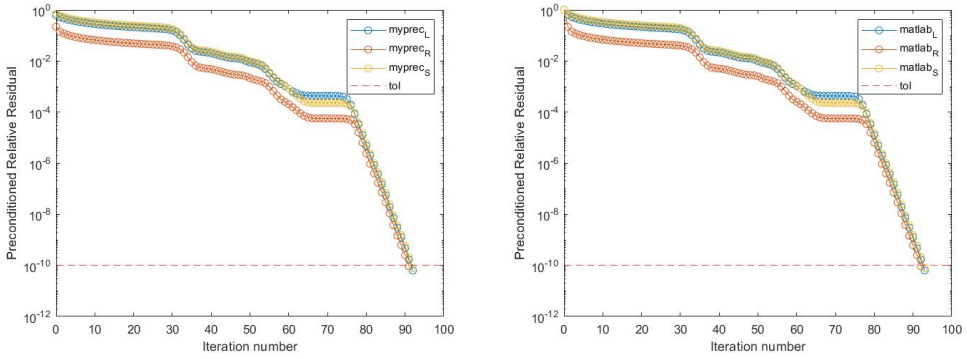

Figure 3: Convergence profiles of implemented Preconditioned GMRES (Left) and inbuilt GMRES(Right) of MATLAB

```
>> notebook3

T =

  5×10 table
```

restart	iter_L	relres_L	left_time	iter_R	relres_R	right_time	iter_S	relres_S	split_time
10	500	3.7718e-07	0.85238	500	2.7017e-08	0.73824	500	0.0015817	0.68442
20	739	9.614e-13	1.3739	461	8.412e-13	0.83335	1000	0.0003962	1.5987
30	88	6.7203e-13	0.17883	89	4.263e-13	0.17763	89	7.8093e-13	0.18442
50	41	4.8414e-13	0.11096	40	7.4341e-13	0.10196	41	6.7106e-13	0.093608
100	41	4.8414e-13	0.10013	40	7.4341e-13	0.097606	41	6.7106e-13	0.091177

Figure 4: Results of Implemented GMRES function and Matlab Inbuilt GMRES function

4 Exercise 5

We know that the denser the preconditioner faster is the convergence, but the denser preconditoner takes more time for factorization. This exercise helps us to understand that the denser preconditioner would be required when the time taken for factorization can be masked by the time taken for solution. As the density of preconditioner increases, the iterations do decrases and so the soultion time, but the total time involving both factorization and the solution stays constant. Hence, the denser preconditioner can help in memory requirements by reducing the number of iterations but the total time to solution might not be compromised.

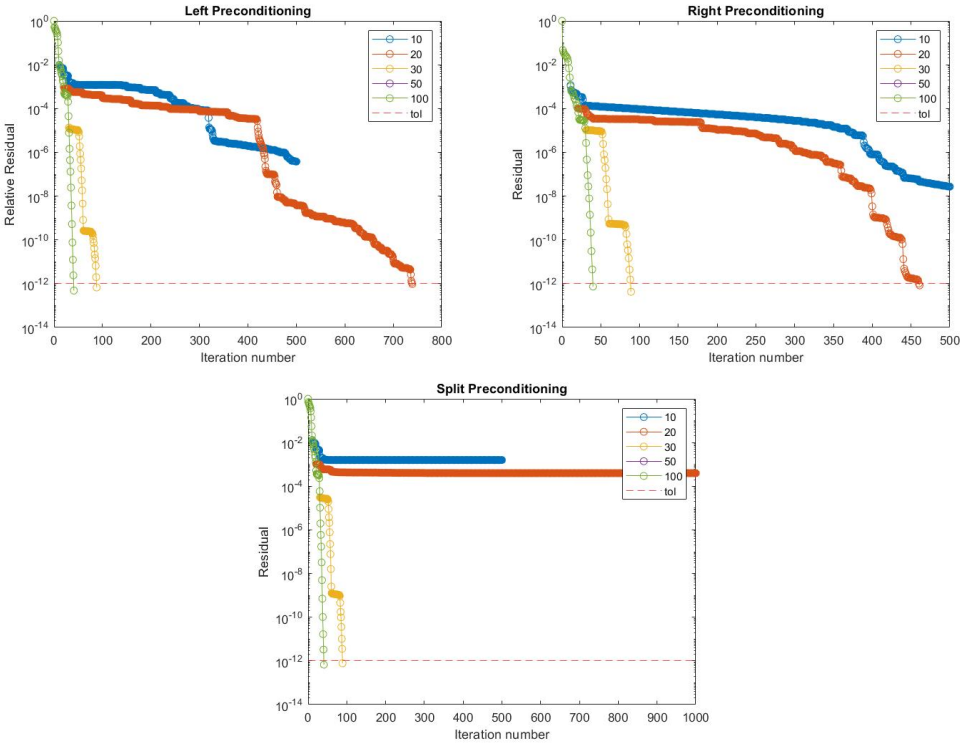

Figure 5: Implementation of Preconditioned GMRES (Left) in MATLAB and comparison with inbuilt GMRES(Right) of MATLAB

5 Exercise 6

The MINRES problem has been solved here using the Matlab inbuilt function for a constrained optimization problem. The convergence profile is shown in the figure 7. The MINRES problem took about 2.3s to be solved with 424 iterations.

T =

6×7 table

droptol	tprec	tsol	tcpu	iters	relres_whole	rho_whole
0.02	0.94185	2.7584	3.7002	436	9.9037e-11	0.44356
0.01	0.95686	1.2276	3.7002	176	9.5257e-11	0.57235
0.003	0.99657	0.49647	3.7002	60	9.0987e-11	0.93376
0.001	1.062	0.28106	3.7002	32	5.6977e-11	1.4488
0.0001	1.5621	0.22974	3.7002	15	2.8405e-11	3.5327
1e-05	4.223	0.23035	3.7002	8	9.3986e-12	8.5559

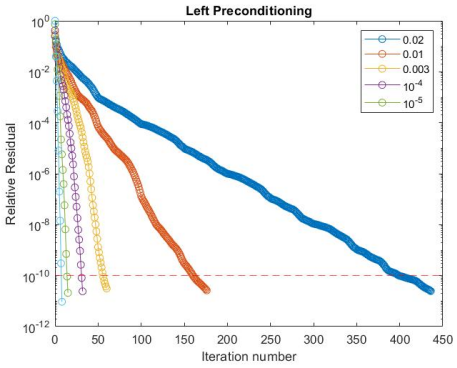

Figure 6: Table(Top) of values comparing the results of different dense preconditioners and Figure(Bottom) shows the convergence profiles for the same

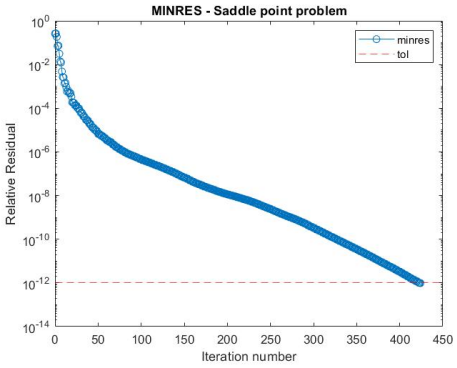

Figure 7: Effect of scaling a system by dividing it with the maximum value of the matrix entry