Advanced Numerical Analysis Homework 2

Keerthana C J

May 8, 2020

1 Exercise 1

As a part of exercise 1, the PCG method has been implemented through mypcg function. For checking the correctness of the method, the FDM discretization of the Poisson problem on a square grid has been solved using both the mypcg function and the MATLAB inbuilt pcg function. The plot in Figure 1 shows the convergence for both the functions.

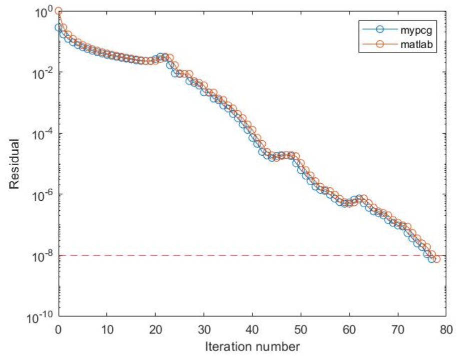

Figure 1: Implementation of PCG in MATLAB and comparison with inbuilt PCG of MATLAB

2 Exercise 2

Here, the FDM discretization of the Poisson problem is solved on a square grid for increasing number of nodes as 100,200,300 and 400. As the number of nodes(nx) increases, we know that

$$h = \frac{1}{nx+1}; nx = \frac{1}{h-1} \tag{1}$$

The condition number of the matrix A behaves as

$$\kappa(A) = \frac{4}{\pi^2}h^{-2} + O(1) \tag{2}$$

Hence, as the number of iterations for the CG method is proportional to $\frac{2}{\pi}h^{-1}\ \sqrt{n}$ for the 2D Laplacian, where n is the number of discretized equations and $n = nx^2$ The expected number of iterations for a CG method for a given tolerance 10^p is given as

$$k = \frac{\log 10}{2}p(\sqrt{\kappa}+1) \tag{3}$$

Therefore, condition number and iterations for the values of nx and tolerance 10^{-8} are given in the table 1 The use of preconditioners reduces the number of iterations. Hence the system was solved with the preconditioners such as IC(0), ICT with drop tolerance 10^{-2} and 10^{-3}. Their corresponding results along with the CPU time are presented in the table 2. We can see that the number of iterations almost doubled as the value of nx is doubled.

nx	$\kappa(A) = \frac{4}{\pi^2}nx^2$	Expected no. of iterations (k)
100	4052	259
200	16211	513
300	36475	768
400	64845	1023

Table 1: Condition number of the matrix and Expected number of iterations for the CG method as a fuction of the number of nodes nx

	CG		PCG					
			IC(0)		IC(10^{-2})		IC(10^{-3})	
nx	iter	CPU	iter	CPU	iter	CPU	iter	CPU
100	283	0.066	87	0.084	45	0.042	17	0.022
200	532	0.33	159	0.39	78	0.24	30	0.16
300	731	1.06	219	1.18	106	0.81	42	0.44
400	948	4.39	282	3.24	137	1.95	53	1.02

Table 2: Actual number of iterations and CPU times for the CG methods with different preconditioners.

3 Exercise 3

The Jacobi preconditioner has been tested for this exercise and the number of iterations are found same as that of the CG method without the preconditioner which can be seen in table 3.

nx	iter	CPU
100	283	0.087
200	532	0.59
300	731	1.82
400	948	5.73

Table 3: The no. of iterations and the CPU times for a Jacobi Preconditioned PCG

The Jacobi Preconditioner is the diagonal matrix of A and the condition number of the preconditioned matrix is given as $\kappa(P^{-1}A)$. Here the preconditioner is D and the diagonal of A is -4. Therefore the minimum$\lambda_{min}(P)$ and maximum$\lambda_{max}(P)$ eigen values of the preconditioner is -4. Hence the eigen values of the system for PCG becomes as in equation (4).

$$\lambda_{min}(P) = \lambda_{max}(P) = -4$$
$$\lambda_{min}(A) = \frac{2}{\pi^2}; \lambda_{max}(A) = \frac{8}{h^2}$$
$$\lambda_{min}(P^{-1}A) = \frac{-1}{4} \times \frac{2}{\pi^2}; \lambda_{max}(P^{-1}A) = \frac{-1}{4} \times \frac{8}{h^2}$$
$$\kappa(P^{-1}A) = \frac{\lambda_{max}(P^{-1}A)}{\lambda_{min}(P^{-1}A)} = \frac{4}{\pi^2}h^{-2} = \kappa(A) \tag{4}$$

Hence, the condition number of the problem is not altered by the Preconditioner which leads to the same condition as that of CG method with additional cost of inverting the Preconditioner, which can be observed as increased CPU time for the Jacobi Preconditioned CG.

4 Exercise 4

As a part of this exercise, a random SPD matrix has been solved corresponding to a random rhs vector using the PCG method with the options no preconditioner, jacobi preconditioner, IC(0), ICT with droptol 10^{-2} and another one with 10^{-3}. The comparison of the iteration profiles is shown in the Figure 2. As we can observe, here the Jacobi preconditioner has reduced the number of iterations as compared to the CG method. The IC preconditioner performs the best and as we see the denser the matrix, the iterations are reduced more.

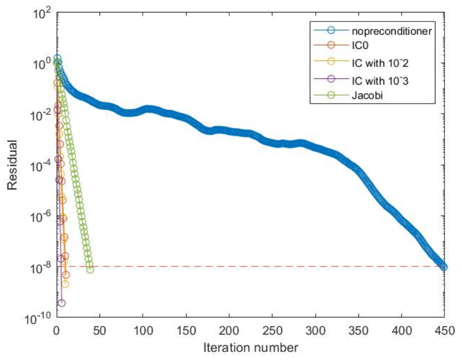

Figure 2: Comparison of various preconditioners

5 Exercise 5

In this section, the apache matrix has been solved first without a conditioner setting the rhs corresponding to an exact solution of all ones. The residual norm achieves the specified tolerance. We find out the maximum entry(a) in the matrix and scale the system accordingly by dividing it by a.The scaling doesn't seem to have any improvement on the number of iterations as shown in figure 3

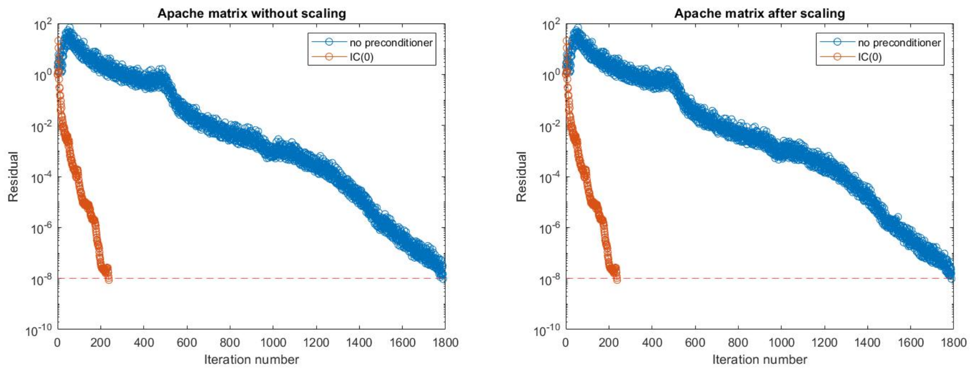

Figure 3: Effect of scaling a system by dividing it with the maximum value of the matrix entry

Experimenting on a structural matrix, when an IC factorization breaks down with error 'nonpositive pivot encountered', the matrix can solved using a diagonal compensation given as follows:

```
alpha = max(sum(abs(A),2)./diag(A))-2
L1 = ichol(A, struct('type','ict','droptol',1e-3,'diagcomp',alpha));
[x,flag,relres,iter,resvec] = pcg(A,b,1e-6,100,L1,L1');
```

Choosing a very small value of $\alpha(\approx 0.1)$ reduces the number of iterations drastically. A structural matrix and is shown in figure

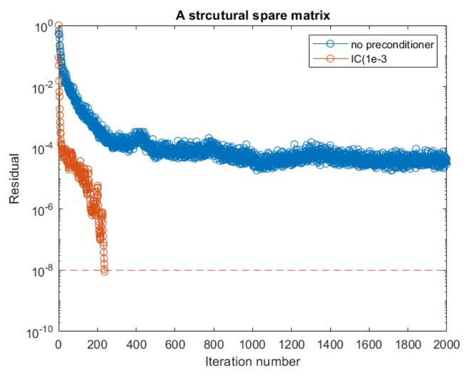

Figure 4: A structural matrix solved with 'diagcomp'

6 Exercise 6

The spectral preconditioner has been implemented as a function handle here and the same is compared with the IC(0) preconditioner for the Poisson problem with two different number of nodes. The main purpose of the function handle is to reduce the memory reuqirement by not explicity forming a preconditioner matrix. The plots in the figure 5 shows the spectral preconditoner against the IC(0) preconditioner.

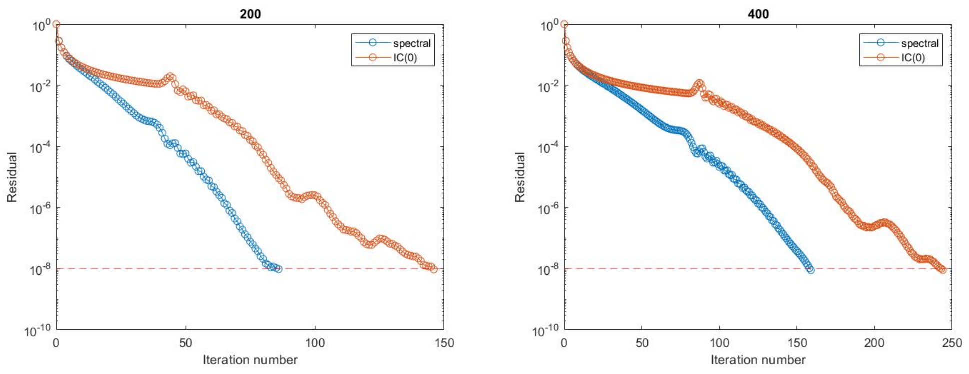

Figure 5: Spectral and IC(0) preconditioners for the Poisson problems with $nx = 200$ and $nx = 400$