ADVNA HW2

Tomba Francesco

03/05/2021

1 Ex 1

From the picture 1 it is possible to point out that the convergence profiles obtained for `mypcg` and MATLAB native `pcg` coincide, see "star" and "circle" markers on the figure. For the test it has been used the discrete Laplacian with parameters in the title of the figure.

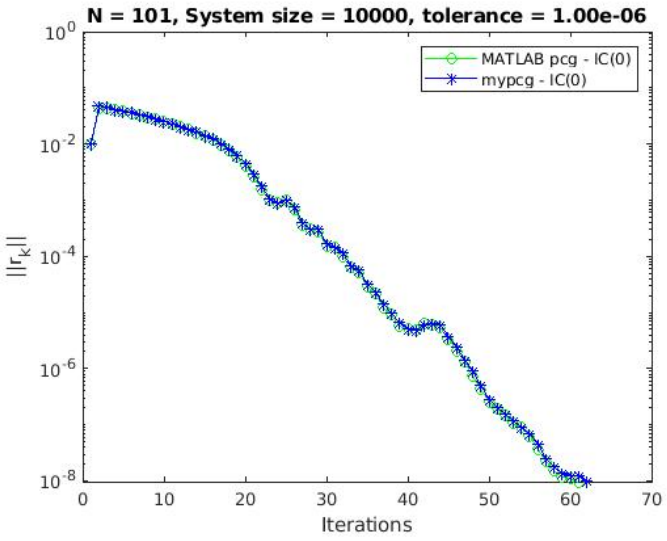


Figure 1: Convergence profiles of MATLAB native implementation of pcg and the personal one

2 Ex 2

The estimate for $\kappa(A)$ was calculated accordingly to the estimation derived in the lectures so:

$$\kappa(A) \approx \frac{4}{\pi^2} h^{-2} = \frac{4}{\pi^2}(N-1)^2$$

Now when N is doubled the condition number grows up as a quadratic function of N. From the theory we know also that the number of iteration expected k is:

$$k \approx \frac{\log 10}{2} p(\sqrt{\kappa} + 1) \propto \sqrt{\kappa} \propto (N - 1) \tag{1}$$

where p is the order of magnitude of the error reduction, between the starting point and the value at convergence. To experimentally prove the relation written before we can calculate for each value of the system size $N_i/\text{Iter}_i \approx 2.4 - 2.8$ Note this bound is not tight and the estimation not fully correct for a series of reasons, in a first time the fact that the theoretical bound for the error reduction in CG is derived from a series of approximation and upper bounds. Practically in fact in some cases the CG algorithm would perform much better than the bound. Note also that equation 1 is obtained manipulating the error reduction formula[1] for CG.

In conclusion we expect that the number of iteration will be directly proportional to the number of subdivisions N. In particular we expect to see that the number of iterations is roughly 2.5 times bigger than the number of subdivisions used in the discrete Laplacian.

			CG		PCG					
					IC		IC(1e-2)		IC(1e-3)	
N-1	n	k(A)	Iter	CPU	Iter	CPU	Iter	CPU	Iter	CPU
100	10000	4134	283	0.09	87	0.07	45	0.04	17	0.02
200	40000	16374	532	0.44	159	0.43	78	0.24	30	0.14
300	90000	36719	731	1.46	219	1.33	106	0.73	42	0.46
400	160000	65170	948	3.89	282	3.22	137	1.88	53	1.05

Figure 2: Table of the experimental results

3 Ex 3

The Jacobi preconditioner given a system described by a matrix A (SPD in order to apply CG and PCG) is by definition $M = \text{diag}(A)$. If we study the case of A being the discrete Laplacian of a 2D discretization of the unit square, M would be only the identity matrix multiplied by 4. In particular the preconditioned system will be:

$$M^{-1}Ax = M^{-1}b \quad \frac{1}{4}\mathbf{I}Ax = \frac{1}{4}b$$

Now the condition number of A and $M^{-1}A$ is the same due to the fact that given an u_λ the eigenvector associated to the eigenvalue λ, the following relation holds

$$M^{-1}A\ u_\lambda = \frac{1}{4}\mathbf{I}Au_\lambda = \frac{\lambda}{4}u_\lambda$$

and so calling λ^A a generic eigenvalue of matrix A:

[1]

$$\frac{||e_k||_A}{||e_0||_A} \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k \leq 10^{-p}$$

$$\kappa(M^{-1}A) = \frac{\frac{|\lambda^A_{max}|}{4}}{\frac{|\lambda^A_{min}|}{4}} = \frac{|\lambda^A_{max}|}{|\lambda^A_{min}|} = \kappa(A)$$

So i) the condition number of the preconditioned system is the same of the original one and ii) the spectrum is only rescaled and the eigenvalue structure and clustering is left untouched, so the iterations needed to solve the 2 systems will be equal, as seen in the experimental result.

			CG		PCG	
					Jacobi	
N-1	n	k(A)	Iter	CPU	Iter	CPU
100	10000	4134	283	0.13	283	0.11
200	40000	16374	532	0.43	532	0.71
300	90000	36719	731	1.32	731	2.31
400	160000	65170	948	3.35	948	5.33

Figure 3: Table of the experimental results

4 Ex 4

In order to obtain the following measures it was used the matrix `wathen` from the gallery, using the parameters on the exercise text. As it is possible to see from the measures in table 4 passing from CG to PCG and changing preconditioner leads to massive improvements (a factor 20) on the number of iteration needed to solve the problem.

		CG		PCG			
				Jacobi		IC(0)	
N-1	n	Iter	CPU	Iter	CPU	Iter	CPU
99	30401	270	0.62	242	0.65	10	0.03

Figure 4: Table of the experimental results

5 Ex 5

By following lectures notes, in order to calculate $\rho(H_J)$ in the estimation of the ω_{opt} parameter, the following relation was used.

$$H_J = \mathbf{I} - M^{-1}A \quad \text{with } M = \text{diag}(A) = 4\mathbf{I}$$

So the maximum eigenvalue in absolute value (so the spectral radius) of the iteration matrix for Jacobi will be of the form

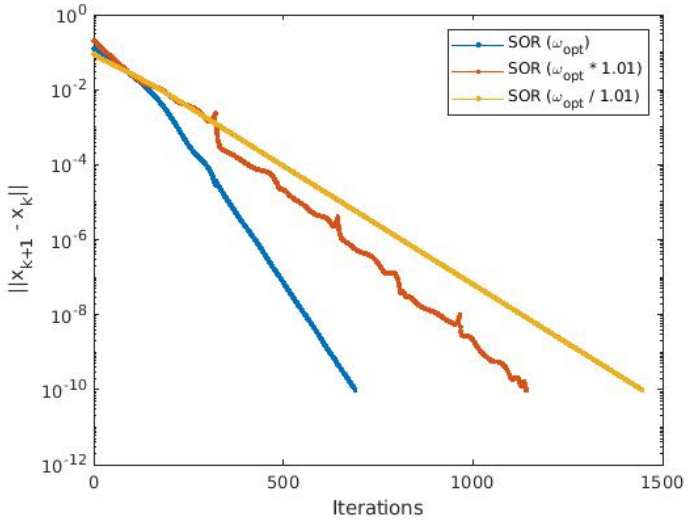
$$\rho(H_J) = \max_{i \in [1, N-1]} |(1 - 2\sin^2\left(\frac{i\pi}{2N}\right)|$$

The maximum is reached for $i = 1$ or $N-1$, so we need to compare only those cases, as proven in previous homeworks. The maximum number of iteration was fixed to be the size of the system since we know CG and PCG converge in at most that number of iteration, moreover, the number of iterations needed by Jacobi and Gauss Seidel could be very large, infact they both reach the cap of iterations ($2.5 \cdot 10^4$).

It is worthy to point out also the fact that even if CG does more than the double of iterations of PCG ICT(0) the CPU time is roughly the same, this is maybe due to the fact that a) there is some overhead associated with the introduction of the preconditioner and the CG algorithm is more optimized and b) since I ran this code on my laptop and the CPU time is on the order of the 100ms maybe the computer was busy doing other things, and so the code ran slower. A more accurate test should be done by repeating multiple times the measure, and averaging the results.

Method	Iterations	CPU	rel err	optional info
Jacobi	: 25600	15.31	7.606142e-03	
Gauss-Seidel	: 25600	18.57	5.814340e-05	
SOR (opt)	: 690	0.57	4.087808e-10	omega opt: 1.96 rho_j 0.9998 rho_omega 0.9615
SOR (2)	: 1141	0.95	2.603808e-10	omega: 1.98
SOR (3)	: 1446	1.17	1.004877e-09	omega: 1.94
Conj. Grad.	: 298	0.13	7.156420e-11	
PCG IC(0)	: 124	0.13	9.846513e-11	
PCG ICT	: 69	0.14	9.846513e-11	tolerance: 1.000000e-02

Figure 5: Table of the experimental results



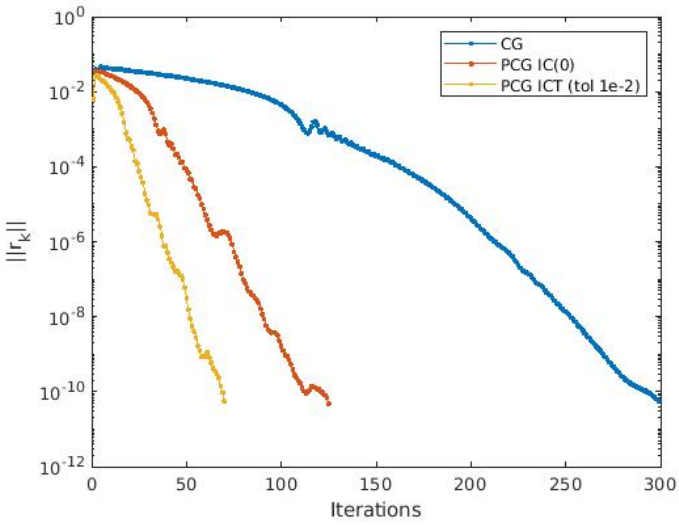


Figure 6: Convergence profiles, exercise 5

Optional question: By using the relation found in slides for the maximum and minimum eigenvalues of the discrete Laplacian we can find that expanding $\sin^2$ and $\cos^2$ for h going to zero up to the first non zero power of h we can find out that the spectral radius of the Jacobi iteration matrix associated with that system is:

$$\rho(H_J) \approx 1 - \frac{\pi^2 h^2}{2}$$

From this we can calculate the ω_{opt} parameter as:

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho_{H_J}^2}} \approx \frac{2}{1 + \sqrt{1 - (1 - \frac{\pi^2 h^2}{2})^2}}$$

Keeping always the first non zero
power of h and using $\sqrt{1+x} \approx 1 + \frac{1}{2}x$

$$\omega_{opt} = \frac{2}{1 + \pi h} \quad \rho_{H_\omega} = \frac{1 - \pi h}{1 + \pi h}$$

Now given a tolerance 10^{-p}, supposing to have a value of h small enough to drop off all constants in the subsequent formulas (Namely a $\log 2$ in k_{CG})

$$k_{SOR} \approx \frac{-p \log 10}{\log \rho_{H_\omega}}$$

$$k_{CG} \approx \frac{-p \log 10}{\log R} \quad \text{with } R = \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}$$

Using $\log(1-x) \approx x$ for x going to 0 and using the fact $\sqrt{\kappa} \approx 2/\pi h$

$$\frac{k_{SOR}}{k_{CG}} = \frac{\log R}{\log \rho_{H_\omega}} \approx \frac{\frac{1-\pi h/2}{1+\pi h/2}}{\frac{1-\pi h}{1+\pi h}} = \frac{2\pi h}{2+\pi h}\frac{1+\pi h}{2\pi h} \approx \frac{1}{2}$$

So theoretically CG would perform a factor 2 slower than SOR but actually as we can see from the measures that CG outperforms SOR.

6 Ex 6

Also in this case the application of the preconditioner leads to a massive improvement in performances (a factor 9), in this case the relative error norm shows a more irregular decreasing trend. This behaviour maybe is caused by the spectral structure of the `Apache1` matrix.

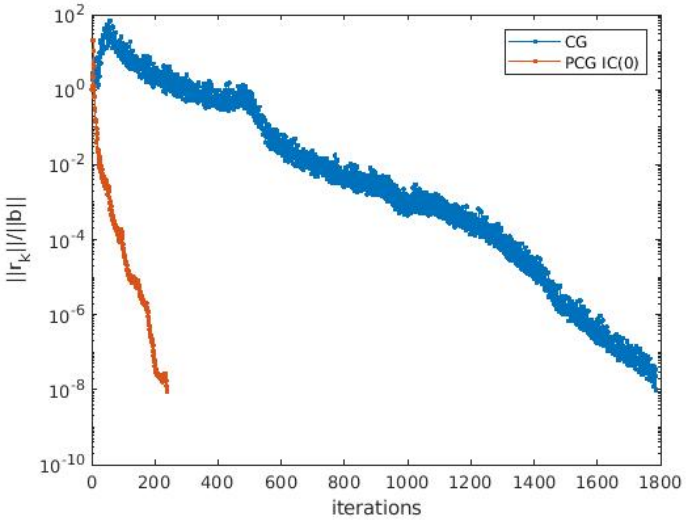


Figure 7: Convergence profiles, exercise 6. `Apache1` matrix

7 Ex 7

In figure 8 the convergence profiles requested.

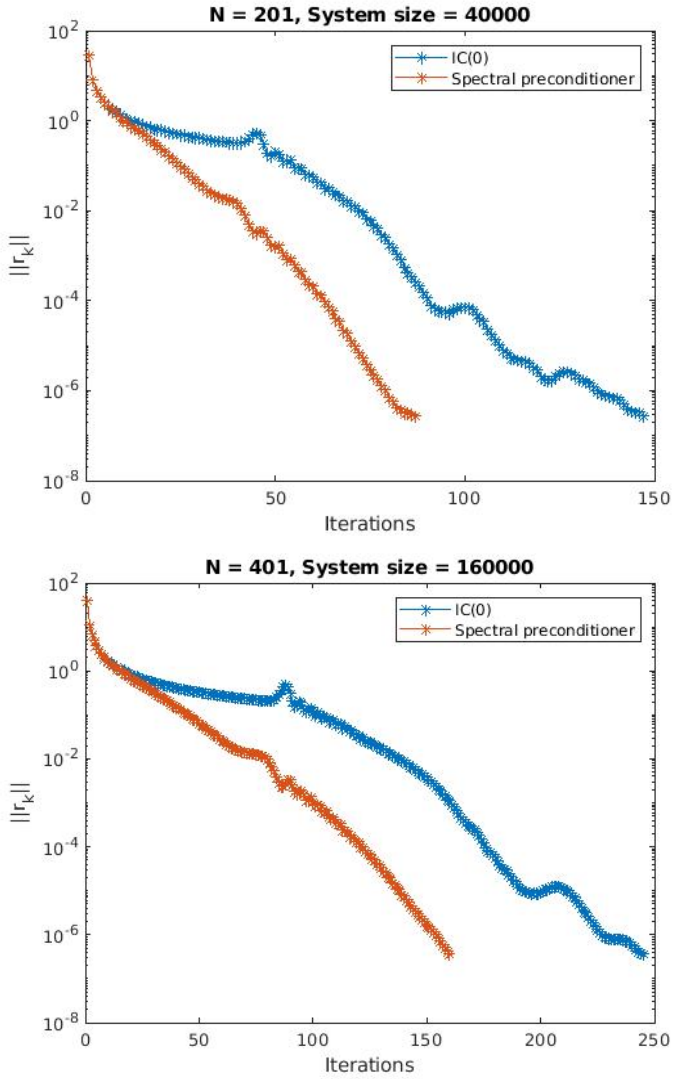


Figure 8: Convergence profiles, exercise 7. Spectral Preconditioner

Now let examine how the addition of the matrix WHW^T to P_0 affects the the spectrum of the matrix P_0A. Defining $P = P_0 + WHW^T$ with $H = (W^TAW)^{-1}$ and W be the matrix whose columns are the eigenvectors of P_0A If we define $\mathbf{w_i}$ the eigenvector of P_0A associated with eigenvalue λ_i, so the matrix W will satisfy the following relation. Let assume also $\lambda_1 < \lambda_2 < \cdots < \lambda_n$

$$P_0AW = P_0A\begin{pmatrix}\mathbf{w_1} & \mathbf{w_2} & \dots & \mathbf{w_p}\end{pmatrix} = \begin{pmatrix}\lambda_1\mathbf{w_1} & \lambda_2\mathbf{w_2} & \dots & \lambda_p\mathbf{w_p}\end{pmatrix} = W\Lambda \tag{2}$$

Where Λ is a p x p matrix containing on the diagonal the first p eigenvalues

$$\Lambda = \begin{pmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_p \end{pmatrix}$$

Now lets see how the first p eigenvalues of P_0A behave when PA is applied to them, lets consider so the expression:

$$PAW = (P_0A)W + (WHW^T)AW = W\Lambda + W(W^TAW)^{-1}(W^TAW)$$
$$= W\Lambda + W = W(\Lambda + \mathbf{I}_p)$$

So $\mathbf{w_1}, ..., \mathbf{w_p}$ are the eigenvectors of PA associated with the eigenvalues $\lambda_1 + 1, ..., \lambda_p + 1$. Since the convergence speed is determined by the condition number of PA and the condition number is determined by the smallest and the biggest eigenvalues, by adding the term WHW^T the p smallest eigenvalues are "pushed up by one" and so the condition number of PA is smaller than the one of P_0A; so the convergence speed is better. To finally prove that statement we have to see also what happens to the eigenvectors $\mathbf{v}_{p+1} \dots \mathbf{v}_n$ associated to the eigenvalues $\lambda_{p+1} \dots \lambda_n$. We are assuming all eigenvalues of P_0A are distinct. In particular they follow:

$$PA\mathbf{v_j} = P_0A\mathbf{v_j} + WHW^T\mathbf{v_j} = \lambda_j\mathbf{v_j} + WHW^TA\mathbf{v_j}$$

The goal is to prove that the correction to P_0 leaves unchanged the eigenvalues $\lambda_{p+1} \dots \lambda_n$, so that $WHW^TA\mathbf{v_j} = 0$.

It is sufficient to prove that:

$$\mathbf{w_i}^TA\mathbf{v_j} = 0 \quad \forall i = 1 \dots p \quad \forall j = p+1 \dots n$$

In the case of $P_0 = (LL^T)^{-1}$ which is symmetric the previous equation can be rewritten in 2 ways:

$$\mathbf{w_i}^TA\mathbf{v_j} = \mathbf{w_i}^TP_0^{-1}\lambda_i\mathbf{v_i} = \lambda_j\mathbf{w_i}^TP_0^{-1}\mathbf{v_i}$$
$$\mathbf{w_i}^TA\mathbf{v_j} = (A\mathbf{w_i})^T\mathbf{v_j} = \lambda_j\mathbf{w_i}^TP_0^{-1}\mathbf{v_j}$$

Calling $\mathbf{u}_{i,j} = \mathbf{w_i}^TP_0^{-1}\mathbf{v_i}$

$$\lambda_i\mathbf{u} = \lambda_j\mathbf{u}$$

Since we have assumed $\lambda_i \neq \lambda_j$ it follows that $\mathbf{u} = 0$, so resuming

$$\lambda_{PA} = \begin{cases} \lambda_i + 1 & \text{if } i = 1 \dots p \\ \lambda_i & \text{if } i = p+1 \dots n \end{cases}$$

So the first eigenvalues of P_0A are shifted up, and the remaining are left unchanged, as consequences i) the condition number of PA is smaller w.r.t. the one of P_0A ii) the eigenvalues live in a smaller interval, so they are more clustered. As consequence of these 2 facts the convergence speed is better.