Università della Svizzera italiana

Institute of Computational Science ICS

Numerical Computing **2020**

Student: Stefano Gonçalves Simao Discussed with: Stella Born

Solution for Project 5

Due date: Wednesday, December 02, 2020, 11:59 PM

Numerical Computing 2020 — Submission Instructions
(Please, notice that following instructions are mandatory: submissions that don't comply with, won't be considered)

- Assignments must be submitted to iCorsi (i.e. in electronic format).
- Provide both executable package and sources (e.g. C/C++ files, Matlab). If you are using libraries, please add them in the file. Sources must be organized in directories called:
  Project_number_lastname_firstname
  and the file must be called:
  project_number_lastname_firstname.zip
  project_number_lastname_firstname.pdf
- The TAs will grade your project by reviewing your project write-up, and looking at the implementation you attempted, and benchmarking your code's performance.
- You are allowed to discuss all questions with anyone you like; however: (i) your submission must list anyone you discussed problems with and (ii) you must write up your submission independently.

The purpose of this assignment is to gain insight on the theoretical and numerical properties of the Conjugate Gradient method. Here we use this method in an image processing application with the goal of deblur an image given the exact (noise-free) blurred image and the original transformation matrix. Note that the "noise-free" simplification is essential for us to solve this problem in the scope of this assignment.

General Questions [10 points]

1. Read the mini-project description and Section 7.4 of *A First Course on Numerical Methods.* ✓

2. What is the size of the matrix A?
   Matrix A has size 62500x62500. We know that generally A has dimension $n^2 \cdot n^2$, so here we have $250^2 \cdot 250^2$ with n = 250 the dimension of the image matrix.

3. How many diagonals bands does A have?
   Using the command $[l, h] = bandwidth(A)$ we have an upper and lower bands of 753 each. Adding the diagonal we have $753 + 753 + 1 = 1507$

4. What is the length of the vectorized blur image b?
   The length of b is $n \cdot n = 62500$

Properties of A [10 points]

1. If A is not symmetric, how would this affect $\tilde{A}$?
   We have that $\tilde{A} = A^T A$, then multiplying a non symmetric matrix like this results again in a symmetric matrix:
   $$(A^T A)^T = (A^T (A^T)^T) = A^T A.$$
   It is also positive-definite as
   $$x^T \tilde{A} x = x^T A^T A x = (Ax)^T Ax = \|Ax\|^2 > 0,$$
   as $Ax \neq 0$ with A non-singular. For these reasons we can see that it doesn't affect $\tilde{A}$.

2. Explain why solving $Ax = b$ for x is equivalent to minimizing $x^T Ax - b^T x$ over x.
   As we can see from Section 7.4 of *A First Course on Numerical Methods* we can see that $Ax = b$ is equivalent to finding $\boldsymbol{x}$ that minimizes
   $$\frac{1}{2}\boldsymbol{x}^T A \boldsymbol{x} - \boldsymbol{b}^T \boldsymbol{x}.$$
   We can see minimizing this means setting the gradient to zero:
   $$\nabla(\frac{1}{2}\boldsymbol{x}^T A \boldsymbol{x} - \boldsymbol{b}^T \boldsymbol{x}) = \boldsymbol{x}^T A^T - \boldsymbol{b}^T = 0$$
   and we have
   $$Ax = b$$
   as wanted.

Conjugate Gradient [40 points]

1. Write a function for the conjugate gradient solver `[x,rvec]=myCG(A,b,x0,max_itr,tol)`, where `x` and `rvec` are, respectively, the solution value and a vector containing the residual at every iteration.
   I did the implementation of the algorithm as described in the sheets, and used the relative residual error as used in the Example 7.10 of the book and also in Matlab for pcg. Note that in my implementation, the first residual norm is not stored in order to have a nicer plot in the next task.

2. In order to validate your implementation, solve the system defined by `A_test.mat` and `b_test.mat`. Plot the convergence (residual vs iteration).
   Here's the plot:

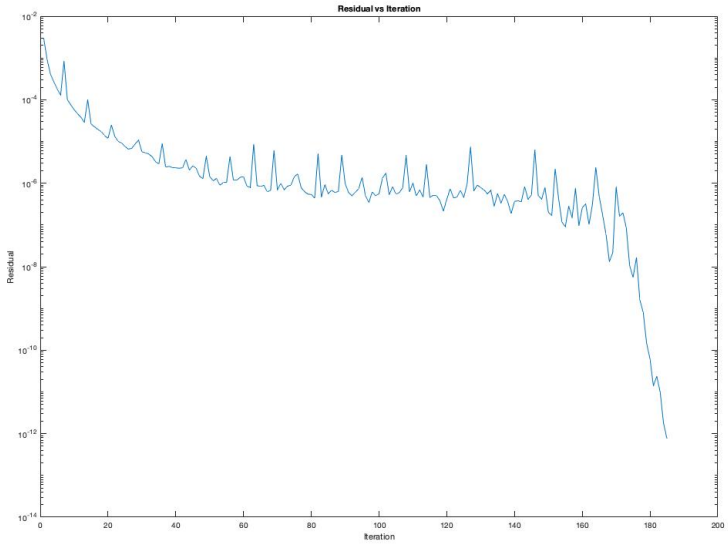

Figure 1: Convergence

We can see that after the first 50 iterations (where we have a good decrease of the residual), the residual oscillates around 10^{-6} until the 170th iteration where it drops rapidly to almost 10^{-12} in approx. 10 iterations. Here is the number of iterations where we can stop.

3. Plot the eigenvalues of `A_test.mat` and comment on the condition number and convergence rate.
   Here's the plot:

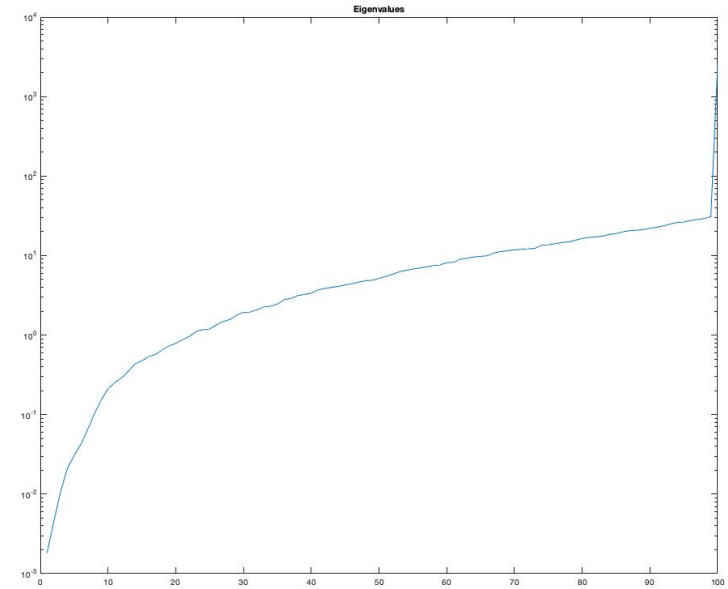

Figure 2: Eigenvalues of A

The condition number is 1.3700e+06. It is quite big and this means the matrix is ill-conditioned. This deteriorates the convergence rate as we have seen in the previous task: after a quite fast convergence of the residual from 10^{-2} to 10^{-6} in the first 50 iterations, then

it stays for 120 iterations on this magnitude before dropping to 10^{-12}.

Debluring problem [40 points]

1. Solve the debluring problem for the blurred image matrix `B.mat` and transformation matrix `A.mat` using your routine `myCG` and Matlab's preconditioned conjugate gradient `pcg`. As a preconditioner, use `ichol` to get the incomplete Cholesky factors and set routine type to `nofill` with $\alpha = 0.01$ for the diagonal shift (see Matlab documentation). Solve the system with both solvers using $max\_iter = 200$ $tol = 10^{-6}$. Plot the convergence (residual vs iteration) of each solver and display the original and final deblurred image.
   Here's the original:

Figure 3: Original Image

Here's the first deblurred image:

Figure 4: Conjugate Gradient deblurred Image

Here's the second deblurred image:

Figure 5: Preconditioned Conjugate Gradient deblurred Image

We can notice how the second image has a little less artifacts as it is closer to the real image. Nevertheless MyCG did a good job in deblurring the image.

Here's the convergences:

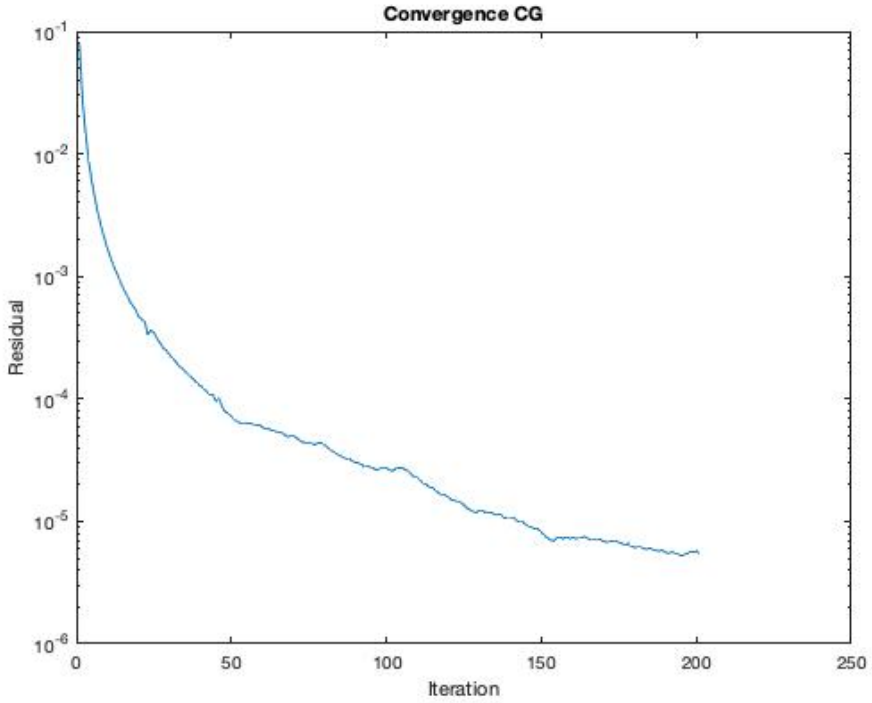

Figure 6: Conjugate Gradient Convergence

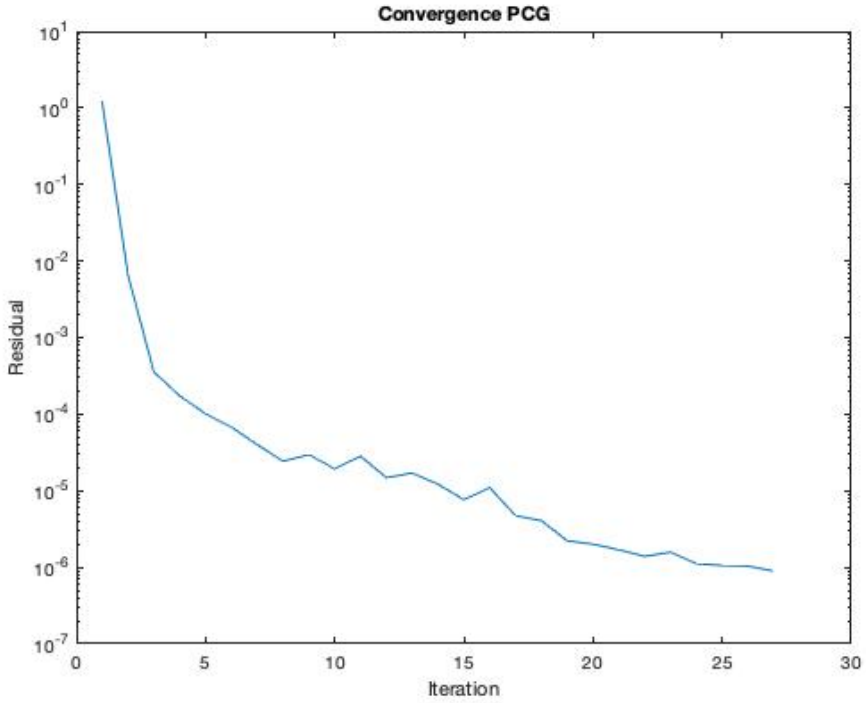

Figure 7: Preconditioned Conjugate Gradient Convergence

We can clearly see the difference here. PCG in less than 30 iterations comes to a residual of 10^{-6} when my implementation it is only able to reach 10^{-5} in 200 iterations. This performance difference is given by the mitigation of the fact that A is ill-conditioned with very high

condition number, by using preconditioning.

2. When would `pcg` be worth the added computational cost? What about if you are debluring lots of images with the same blur operator?
   When the condition number is large (the matrix is ill-conditioned), the iterative method converges very slowly, in this situation it is worth preconditioning the matrix and use PCG. If we need to deblur a lot of images that share the same matrix A, we only need to compute the right side of this preconditioning equation every time, the left side needs to be computed only once:
   $$(L^{-1}\tilde{A}L^{-1})(Lx) = L^{-1}\tilde{b}.$$
   This can help the convergence but only if matrix A has a really large condition number, as PCG really speeds up the convergence.