MATH 612: CM4ES&FM

Spring'14 | Coding assignment # 2 | Due March 21

1. For this piece of work, you will need to start with your own code of the steepest descent and the conjugate gradient methods. They are the algorithms shown at the end of the slides of the first week. Code them and test them. They solve systems $Ax = b$, where A is real symmetric and positive definite. In addition to the approximate solution, *they should output the iteration number where the algorithm stopped.* For the next collection of tests, you'll first need to create a large $m \times m$ real unitary matrix Q. With this fixed matrix (do not change it from experiment to experiment), you'll create a matrix

$$A = Q\Lambda Q^*, \qquad \Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_m \end{bmatrix} \qquad \text{with} \qquad \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m > 0.$$

   This is the general form of a symmetric positive definite matrix. The numbers λ_j are the eigenvalues and, at the same time, the singular values of A. The next ingredient is a fixed non-zero vector $x \in \mathbb{R}^m$, which you will keep for all your computations. Make it somewhat equilibrated, with not too many crazy changes of magnitude in its entries. Then you compute $b = Ax$. In summary, Q and x are fixed and then, for each choice of $\lambda_1, \ldots, \lambda_m$, you compute A and b. Then you solve $Ax = b$ for a prescribed tolerance, starting always with $x_0 = b$, and count how many iterations you needed to get the result.

   A warning word now. You will need a somewhat large value of m to preceive many effects. Try several until you start seeing things. The same applies to the tolerance. If the tolerance (which is a relative value) is too large, you might not see the desired effects.

   (a) Take $\lambda_1 = C$, $\lambda_m = 1$, and all the other eigenvalues equispaced between then. Solve and display the number of iterations as $C \to \infty$.

   (b) Take $\lambda_1 = C \to \infty$ and all other eigenvalues equispaced between 1 and 2. Solve and display the number of iterations.

2. Make a function `conditionNumber(A)` that computes the $2-$condition number of a matrix using power iteration for A^*A and for $(A^{-1})^*A^{-1}$ withouth ever computing A^*A and A^{-1}. Systems can be solved with the backslash command. Everything should be in the same function. There should not be appeals to other pre-exisiting functions, like the `TwoNorm` function you coded for the previous assignment. You can use expand that piece of code though, as long as it complies with the rule that everything is in the same function. Test it and compare it with Matlab's own `cond`.