Computational Methods for Statistic with Applications Computer Exercise no. 7: Iterative solution methods, eigenvalue computations

NGSSC, LU, SLU, UU

September, 2011

The task of this computer lab is to test some iterative solution methods and obtain some experience regarding their rate of convergence.
At the end, the results of two exercises have to be sketched and commented, and given to the lab-consultant. It is recommended to use the publishing facilities, provided in $\mathbb{R}$ and `Matlab`.
Some of the tasks require some data files which can be downloaded via
`http://http://user.it.uu.se/~maya/Courses/NGSSC/STAT_Labs/MN/`.

You have at your disposal some matrices from Matrix Market (`*.mtx`). You can easily load them in $\mathbb{R}$ or `Matlab`, however elementary utility is provided (`hb_file_read`) for your convenience. The syntax to use it in `Matlab` is the following: `[A,b]=hb_file_read('rw5151');`.
It reads the matrix (as sparse) in A, loads a right-hand vector if such exists or, otherwise, returns a random vector b matching the size of the matrix. In most cases it is the matrix, which matters.

Exercise 1 (Convergence of the unpreconditioned CG method)
Consider the solution of $A\mathbf{x} = \mathbf{b}$ by the standard conjugate gradient, where

$$A = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The exact solution is $\hat{\mathbf{x}} = [1, 1, \cdots, 1]^T$. Starting with $\mathbf{x}^0 = [0, 0, \cdots, 0]^T$ run the unpreconditioned conjugate gradient method.
In the lab directory you find straightforward implementations of the unpreconditioned CG in $\mathbb{R}$ and `Matlab`, `cg.m` and `cg.r`, however you may use your own implementation of the standard CG method. Those, who work in `Matlab`, may use the `Matlab` CG, called for example as

```
x_it=pcg(A,b,1e-6,size(A,1));
```

The matrix A can be generated in various ways. One lazy possibility is the following:

```
A = 2*eye(n);
A = A + diag(-ones(n-1,1),1);
A = A + diag(-ones(n-1,1),-1);
A(n,n)=1;
```

1. Run experiments for a number of different sizes of the matrix A, say, $n = 10, 50, 100$.

   For how many iterations does the method converges? Monitor the current CG iterates (the approximations of the solution $\mathbf{x}$ produced during each iteration. What do you observe?

2. It is very instructive to derive the exact form of the CG iterates and see how the method proceeds. One finds that after k iterations

   $$\mathbf{x}^{(k)} = \left[\frac{k}{k+1}, \frac{k-1}{k+1}, \cdots, \frac{1}{k+1}, 0, \cdots, 0\right]^T$$

   for $1 \leq k \leq n-1$ and $\mathbf{x}^{(n)} = \hat{\mathbf{x}}$.

   Hence, the information travels one step at a time from left to right and it takes n steps before the last component has changed at all.

   Therefore, in order to speedup the convergence of the iterative solution methods, one needs good preconditioners.

Exercise 2 (A doubly stochastic matrix, PSMIGR 3: US inter-county migration 1965-1970)

This is a real unsymmetric sparse matrix of size 3140×3140 with 543162 nonzero entries. It is supplied to the Matrix Market (`http://math.nist.gov/MatrixMarket/`) by Paul Slater, University of California at Santa Barbara in November 1983 and originates from a demographical study.

1. Study the matrix - its sparsity pattern, conditioning, scaling. Compute all its eigenvalues and plot them. Is the matrix strongly nonsymmetric or not? Is it well- or ill-conditioned?

2. Solve a system with the `PSMIGR`-matrix using the GMRES iterative method, implemented by the `Matlab` commands `gmres`. If you have an implementation of GMRES in $\mathbb{R}$, use that one.

   ```
   [x,flag,relres,iter,resvec] = gmres(A,b,restart,tol,maxit);
   ```

   with $maxit = n$, $restart = n$, $tol = 1e-6$, n - equal to the size of A.

   Plot the convergence history of each run (for instance, as `semilogy(resvec,'o')`). Compute the exact solution by the direct `Matlab` solver and compare the difference between that one and the iteratively computed solution. Does the difference match the stopping tolerance?

3. Repeat the experiment with $restart = 10$. Plot the convergence history and compare with that when full GMRES was used.

   Did we save any computing resources? Which and how much?

4. Investigate the possibility to use some incomplete LU-factorization preconditioner, obtained using `Matlab` command `luinc` or `ilu` for newer `Matlab` versions. Try, for instance, `[L,U] = luinc(A,tau)` with `tau=0.1` or `tau=1e-5`. If it works, compare the convergence and the accuracy of the computed solution. What are your conclusions?

Exercise 3 (Eigenvalue problem)
You are given two matrices, which originate from Markov Chain models - rw136 (136x136) and rw5151 (5151x5151). Those are referred to as *transition matrices* and are (right) stochastic matrices, thus, their row sum is (nearly) equal to the vector $[1, 1, \cdots, 1]$,

$$\sum_{j=1} nA(k, j) = 1, \text{ for any } k = 1, 2, \cdots, n.$$

The primal interest is in the steady state probabilities of the chain, which is ordinarily the appropriately scaled left eigenvector corresponding to the eigenvalue unity (see `http://en.wikipedia.org/wiki/Stochastic_matrix` about existence and unicity of such vector etc., `http://math.nist.gov/MatrixMarket/data/NEP/mvmrwk/mvmrwk.html`, `http://www.cise.ufl.edu/research/sparse/matrices/Bai/rw136.html` or `http://www.cise.ufl.edu/research/sparse/matrices/Bai/rw5151.html`). The left eigenvector, corresponding to eigenvalue 1, satisfies the relation

$$\mathbf{q}A = \mathbf{q}.$$

By transposing the above, we see that $\mathbf{q}$ is the eigenvector of A^T, corresponding to the eigenvalue 1.

Tasks:

1. Load the two matrices and compute all their eigenvalues. Find the left eigenvector, corresponding to the eigenvalue equal to 1.

2. Are the matrices stochastic? Is this influence the stationary vector? Suggestion: make the matrices exactly left stochastic and recompute the stationary eigenvector. What is the order of the difference in the vectors?

   How much time does it take to compute all the eigenvalues and eigenvectors for the two sizes?

3. In `Matlab`, for sparse matrices, one can use the command `eigs` to compute a few eigensolutions at some part of the spectrum. One can try `[V,D]=svds(A,5,1)`, which will compute the 5 eigensolutions, closest to 1. This, clearly, saves computational time and memory.

   How does one compute only part of the spectrum in $\mathbb{R}$?

4. Use the routine `Arnoldi.m` to compute the matrices V and H in the Arnoldi decomposition $AV = VH$. The eigenvalues of H must be the same as those of A and V must have mutually orthogonal column vectors. Is that true? Comment on that.

Exercise 4 (In case you have nothing else to do)
Have some fun with the iterative solution methods: Peaks and plateaus

Background: In the so-called Galerkin methods, typical representatives of which are the Full Orthogonalization method (FOM), Biconjugate gradient (BICG) method and the Lanczos method, the approximation $\mathrm{x}^{(k)}$ is chosen such that the corresponding residual is orthogonal to a certain space. The methods in this class suffer some numerical instabilities and various minimum- and quasi-minimum residual methods have been developed to overcome these instabilities.
Examples of such pairs of methods are
1: GMRES and FOM
2: QMR and BICG
3: Minres and Lanczos
4: GCG-RM and GCG-OR.
The convergence behaviour of pairs of such methods shows striking similarities. One essential result shown is a relation between residuals of these pairs, called the effect of peaks and plateaus. Whenever a peak appears in the convergence plot of the orthogonal residual method, there is a plateau under it for the corresponding norm-minimizing method. Several peaks may sit on the top of what appears to be a single plateau, where there are unacceptably small improvements in the residual norm over a number of consecutive iterations. In addition, whenever the residual norm plot for the norm-minimizing method rapidly decreases, the corresponding residual norm plot for the orthogonal residual method also rapidly decreases. Thus, the corresponding residual plots appear to track each other if the arithmetic used is exact. In finite precision arithmetic one extremely high peak could lead to loss of accuracy and then the convergence plot of the orthogonal method will not meet the convergence plot of the corresponding minimal residual method at the end of the plateau under this peak.
Denote by $\mathbf{r}^{(k)}$ and $\mathbf{r}^{(k,G)}$ the residual of a minimal residual method and the residual of the corresponding Galerkin orthogonal residual method. It is shown that if the residuals are computed exactly at step k, they satisfy the following relation

$$||\mathbf{r}^{(k,G)}|| \geq \frac{||\mathbf{r}^{(k)}||}{\sqrt{1-\frac{||\mathbf{r}^{(k)}||^2}{||\mathbf{r}^{(k-1)}||^2}}}.$$

Whenever $\frac{||\mathbf{r}^{(k)}||^2}{||\mathbf{r}^{(k-1)}||^2}$ is close to unity, then $||\mathbf{r}^{(k,G)}||$ is much larger than $||\mathbf{r}^{(k)}||$.
The following test illustrates the effect of peaks and plateaus.

1. load the matrix `A_pp` and the corresponding rhs vector `b_pp`.
2. Solve the corresponding system `A_pp*x=b_pp` by the BICG and the QMR methods:

```
[x_qmr,f_qmr,rres_qmr,it_qmr,rvec_qmr] = qmr(A_pp,b_pp,1e-6,size(A_pp,1));
[x_bcg,f_bcg,rres_bcg,it_bcg,rvec_bcg] = bicg(A_pp,b_pp,1e-6,size(A_pp,1));
```

Plot the convergence histories of both methods on one figure.

```
semilogy(rvec_qmr,'o-')
hold
semilogy(rvec_bcg,'*-r')
```

What do you see?

3. Check whether the accuracy of the solution is affected from the erratic convergence behaviour of the BICG method.