Kungl. Tekniska Högskolan **2012**
CSC / NA
Elias Jarlebring

Deadline 2012-06-15

Homework set 4.
Advanced numerical methods for science and engineering - DN3250

http://www.csc.kth.se/~eliasj/DN3250/

Jacobi-Davidson and ARPACK

---

1. (a) Download the matlab implementation of Jacobi-Davidson JDQR (jdqr.m) from

   `http://www.staff.science.uu.nl/~vorst102/JDQR.html`.

   (b) Load the matrix `poisson` from the matrix market with the command

   ```
   A=gallery('poisson', 80);
   ```

   and use jdqr to compute 5 eigenvalues. Read the help for the `jdqr.m` and generate input parameters such that the convergence is plotted.

   (c) Carry out the program with different choices of the parameters such that 10 eigenvalues are computed, BiCGStab is used to solve the linear system, and one step of the Gauss-Seidel iteration is used as a preconditioner. Plot the convergence and report the CPU time.
   Hint: You may use `jdqr(A,0,20,5,struct('Disp',1),[L U]);` with L, U corresponding to a Gauss-Seidel preconditioner.

2. Download `arnupd.m`, `arnoldi_sorensen.m` and `arnoldi_standard.m` from the course web page. Use the example described in the first lines in `arnupd.m`. Modify the code such that after every restart, a standard Arnoldi method (with `arnoldi_standard.m`) is executed started with the first vector in V. Print the norm of the difference compare the norm of the difference between the $p \times p$ part of the Hessenberg matrix generated by the restart and the Hessenberg matrix generated by the standard Arnoldi process. Explain your observation. Are they equal? Are they expected to be equal? Why?

3. (a) Download and install the ARPACK package. Hints for this can be found on the next page.

   (b) Compile the example "EXAMPLES/SIMPLE/dssimp" and change the code to compute 10 eigenvalues and change the tolerance to 10^{-10}. In the report for the homework, present the eigenvalues as well as the number of restarts necessary.

   (c) The example `dssimp` corresponds to a discretization of the 2-dimensional Laplacian on a square. Refine the discretization by changing the parameters in the code such that we have 150 unknowns in each direction. Increase the size of the Arnoldi factorization until it converges to 10 eigenvalues.

   (d) Read the implementation of the function `av` and `tv` and construct the corresponding sparse matrix in matlab. The matlab command `eigs` solves eigenvalue problems with restarted Arnoldi. Construct the matrix and solve the eigenvalue problem with

   ```
   tic; eigs(A,10,'lm'), toc
   ```

   You should get the same eigenvalues as with `dssimp`. Compare computation time with `dssimp`. The computation time can, e.g., be measured with

   ```
   time ./dssimp
   ```

   Explain differences in timing.

   (e) Change the size of the Arnoldi factorization and carry out timings for the size $20-35$ of the Arnoldi factorization. For what size of Arnoldi factorization do you get the minimum CPU time and still maintain convergence to all 10 eigenvalues? Give the timings in your report.
   We saw in the lectures that orthogonalization can be expensive in Arnoldi's method, at least if we do not carry out restarting. Analyze the timing-output of the algorithm and explain what component in the algorithm is more computationally demanding.

Hints for installation of ARPACK

(i) Download ARPACK (arpack96.tar.gz) from

`http://www.caam.rice.edu/software/ARPACK/download.html#ARPACK`

and unpack the package.

(ii) You first need to compile the ARPACK library by following the instructions in the file ARPACK/README. This involves several modifications in the file `ARmake.inc`. It might be helpful to start with this ARmake.inc

`http://www.csc.kth.se/~eliasj/DN3250/ARmake.inc`

which is adapted for a linux-like system where the BLAS libraries and LAPACK libraries are system-wide installed. If your setup is correct, you will get a file `libarpack_lnx.a` by running

```
make lib
```

(iii) Verify that the your build was successful by doing

```
cd EXAMPLES/SYM/
make ssdrv
./ssdrv1
```