Homework 1

Information about homework:

- If correct solutions to questions 2-5 are handed in before the homework 1 deadline, 1 bonus point will be awarded for the final exam. The deadline is visible in CANVAS. If solutions to questions 2-5 handed in before that date are not correct, they have to be redone, but the second time without yielding bonus points for the exam. Solutions must be clearly written, and easy to follow. If not, they will not generate bonus points, and must be redone. It is highly recommended that the answers are done on a computer (for instance LaTeX or word). Task 2-5 can be done in groups of two (single person groups are allow). Please submit, not later than midnight of the date of the homework 1 deadline:
  - written solutions (in the form of a PDF-file)
  - your matlab (or julia) programs by

  The files should be uploaded in CANVAS → SF2524 → Assignments → Homework 1

- If task 1 is solved before the *wiki-deadline*, 0.5 wiki-bonus will be awarded. If you collect y wiki-bonus points during the course, the grade limits for grade A and B will be reduced by y points. The wiki tasks are reported by you by specifying questions numbers (before wiki deadline) in
  CANVAS → SF2524 → Assignments → Wiki part homework 1

---

1. Create problems and solutions on the course training wiki. This task is optional but can give wiki-bonus. In the wiki, create
   - in Block 1-2: x exercise problems (per person), without solutions.
   - in Block 1-2: x solutions (per person) to problems which do not yet have a solution. Don't do the problems you created.

   If you are attending SF2524 $x = 2$ if you are attending SF3580 $x = 3$. In order to get good problem ideas, you might want to do the wiki-problems in parallell or after solving the problems below.

Course training wiki: `http://gragg.math.kth.se/sf2524/merge_group_pages2.php?name=21457`
QR-code link:

2. Consider the following matrix:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 2 \\ 3 & 2 & 9 \end{bmatrix}.$$

Plot the eigenvalue error as a function of iterate for the following algorithms in a *semilog* plot, and relate with the convergence theory.

You may in this exercise use `eig(A)` as a reference/exact solution.

(a) Power method with the starting vector $x_0^T = (1, 1, 1)$.

(b) Rayleigh qoutient iteration with the starting vector $x_0^T = (1, 1, 1)$.

(c) Change the entry of the matrix in element row 1, column 3, to $a_{1,3} = 4$ and again run the Rayleigh qoutient iteration with $x_0^T = (1, 1, 1)$. Explain why we obtain slower convergence in comparison to (b).

3. In this exercise we will investigate the performance of different versions of the Gram-Schmidt orthogonalization when it is combined with the Arnoldi method. Consider the matrix A constructed with the following command

```
rand('seed',0); A=gallery('wathen',nn,nn);        (1)
```

(a) Modify the orthogonalization in `arnoldi.m` available from the course we page. Apply it to (1) and generate the values in the following table, where `time` is the CPU-time and `orth`=$\|Q_m^T Q_m - I\|$ is an indicator of the orthogonality of the basis, and m the number of iterations in Arnoldi's method.

Use tic-toc to measure performance in MATLAB. In order to get reliable estimates, you may need the tricks described here: `http://se.mathworks.com/help/matlab/matlab_prog/measure-performance-of-your-program.html`.

	single GS		modified GS		double GS		triple GS	
	time	orth	time	orth	time	orth	time	orth
$m = 5$								
$m = 10$								
$m = 20$								
$m = 50$								
$m = 100$								

Select `nn`≥ 10 such that the maximum computation time for $m = 100$ is approximately 10 minutes on your computer.

(b) Interpret the result in (a). Is one version "best" in this setting? If so, in what sense is it "best"? Discuss the fairness of the comparison.

4. We shall here investigate a primitive variant of the Arnoldi method. Let K_m be a matrix with the iterates of the power method, that is, let $K_m := \left[b, Ab/\|Ab\|, \ldots, A^{m-1}b/\|A^{m-1}b\|\right] \in \mathbb{R}^{n\times m}$.

Background Galerkin method: The Galerkin method is a fundamental technique in the finite element method for partial differential equations. The approximation is generated from the bilinear form $a(\cdot,\cdot)$ and $f(\cdot)$ on vector space $V = \text{span}(v_1, \ldots, v_m)$ and is defined by the equality

$$\sum_{i=1}^{m} w_i a(v_i, v_j) = f(v_j)$$

for all v_i, $i = 1, \ldots, m$.

(a) The approximation stemming from Galerkin method applied to the bilinear form associated with the eigenvalue problem ($a(u, v) =$

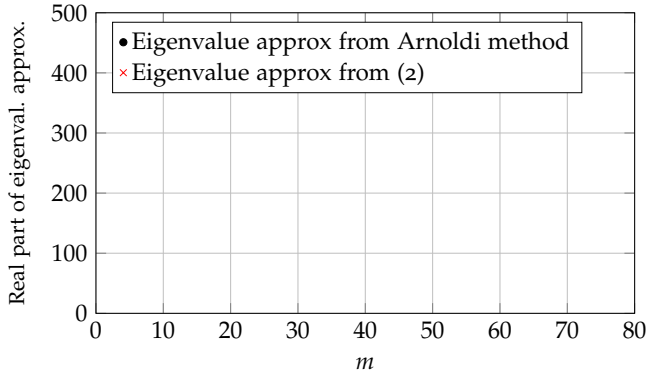
$u^T Av - \mu u^T v$, $f(v) = 0$) for the subspace spanned by the columns of K_m satisfies

$$\mu K_m^T K_m w = K_m^T A K_m w \tag{2}$$

You may assume that $\mathcal{K}_m(A, b)$ has dimension m. Prove that the approximation (2) is identical to the approximation generated by Arnoldi's method for eigenvalue problem. No prior knowledge about the Galerkin method is required to solve the problem.

Hint: Recall that if V_m is an orthogonal basis of $\mathcal{K}_m(A, b)$, then the Arnoldi approximation is eigenvalues of $V_m^T A V_m$. Maybe there exists a non-singular matrix R such that $K_m = V_m R$? Show that $R^T R = K_m^T K_m$ by considering $V_m^T V_m$.

(b) By solving (2) directly with `eig`, we have a method which requires only computation of $K_m^T K_m$ and $K_m^T A K_M$, and does not require any orthogonalization. Apply this method to (1) and compare with the Arnoldi method. Use double GS as orthogonalization. Generate the following figure for $m = 1, 2, \ldots$: Use a random starting vector, with the same starting vector for all simulations.



(c) Interpret the result of (a)-(b). What do we expect in exact arithmetic? What do we observe? Which approach is better? Why?

5. Download and load the matrix

   http://www.math.kth.se/na/SF2524/matber14/Bwedge.mat

Use the command `load('Bwedge.mat')` to access the matrix B and eigenvalues.

(a) The eigenvalues are given in the variable `B_eigvals`. Plot the eigenvalues and mark the eigenvalues you will have the fastest convergence with Arnoldi's method.

(b) In the same figures as in (a), indicate circles and estimate a convergence factor. The convergence factor is here meant $\alpha < 1$ such that the error $\sim \alpha^m$.

(c) Generate figures with the Ritz values for the Arnoldi method, with double GS for $m = 2, 4, 8, 10, 20, 30, 40$. Clearly indicate which eigenvalue you expect fast convergence based on (a). How many iterations are needed in order to get an eigenvalue error approximately 10^{-10}

For those of you doing the homework in the Julia programming language, the mat-file can be loaded in this way:

```
julia> ;wget http://www.math...14/Bwedge.mat
julia> Pkg.add("MAT")
julia> using MAT
julia> bw=MAT.matread("Bwedge.mat")
```

Modify your algorithm such that it carries out shift-and-invert Arnoldi method with shift $\sigma = -11 + 2i$.

(d) Plot the eigenvalues of the transformed matrix $B = (A - \sigma I)^{-1}$ and indicate as in (a)

(e) How many iterations are needed to achieve eigenvalue accuracy 10^{-10}. Is it faster or slower than (b)? Does the theory predict faster or slower than (c)? You can justify your reasoning with figures.

Only for PhD students taking the course *SF3580 Numerical linear algebra*:

6. Exercise about restarting. Use the matrix Bwedge.mat in the simulations.

(a) Implement an explicit restarting strategy for the Arnoldi method as follows. Take a linear combination (with unit weights) of the Ritz vectors generated by m iterations of the Arnoldi method as a new starting vector. Use the Ritz vectors corresponding to the k largest Ritz values. Extract the Ritz vectors as follows:

```
[Q,H]=arnoldi(A,b,m);
[V,D]=eig(H(1:end-1,1:end)); d=diag(D);
[Y,I]=sort(-abs(d));
ritz_vals=d(I(1:k));
ritz_vecs=Q(:,1:end-1)*V(:,I(1:k));
```

Carry out the experiments for $m = 10$, $k = 5$ and $m = 20$, $k = 10$ with 100 restarts. Plot the eigenvalue approximation as a function of restart. Does the restart strategy work in practice?

(b) Use the reference code `arnupd.m` and `arnoldi_sorensen.m` which are implementations of an implicit restarting strategy. Run the algorithm with same restarting parameters as in (a) and explain the difference. Implicit restarting is in general to prefer over explicit restarting. Why?

The programs `arnoldi_sorensen.m` and `arnupd.m` are reference implementations of *Implicit application of polynomial filters in a k-step arnoldi method, D. C. Sorensen, SIAM J. Matrix Anal. Appl. Vol. 13, No. 1, pp. 357-385, 1992*. It is not necessary to read the details of the paper to solve the homework. The programs are available from http://www.math.kth.se/~eliasj/NLA/arnupd.m Another good reference for implicit restarting are provided in the lecture notes: http://people.inf.ethz.ch/arbenz/ewp/Lnotes/chapter10.pdf