Homework 1

If correct solutions are handed in before the homework 1 deadline, and canvas peer-review is done correctly, bonus point will be awarded for the final exam. If an incomplete or incorrect solution is handed in, you will need to redo the homework, and might obtain partial bonus.
It is highly recommended that the answers are done on a computer (for instance LaTeX or word). All tasks should be done in groups of two (single person groups are allowed). Please submit, by the deadline specified in CANVAS

- written solutions (in the form of a PDF-file). CANVAS → SF2524 → Assignments → Homework 1
- your matlab (or julia) programs for the problems. CANVAS → SF2524 → Assignments → Homework 1 sourcecode

---

1. Specify here your AI usage level (0-3) and when applicable further explanation. The level can also be specified per solution.
   See CANVAS → SF2524 → "Can I use generative AI?"

2. Consider the following matrix:

$$B = \begin{bmatrix} 4 & 1 & 1 \\ 1 & 3 & 0 \\ 1 & 0 & 2 \end{bmatrix}.$$

   Plot the eigenvalue error as a function of the iterate (in a *semilog* plot) for the following algorithms, and discuss the results in relation to the convergence theory, for example by describing the relation with a property in the plot with a specific page or theorem in the lecture notes.

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

   (a) **Power method.** Use the starting vector $x_0^T = (1, 1, 1)$. Comment on how the convergence rate relates to the ratio $|\lambda_2/\lambda_1|$.

   (b) **Inverse iteration with a fixed shift.** Run inverse iteration with the shift $\mu = 3$ and starting vector $x_0^T = (1, 1, 1)$. Identify which eigenvalue the iteration converges to, and compare the rate with the power method.

(c) **Rayleigh quotient iteration.** Use the same starting vector $x_0^T = (1, 1, 1)$. What is the observed convergence rate?

(d) **Perturbed matrix.** Change the entry in row 2, column 1, to $b_{2,1} = 1.5$, and again run the Rayleigh quotient iteration with the same starting vector. Explain why convergence is slower compared to part (c).

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)
```

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.

(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.

	single GS (MV version)		single GS (for version)		modified GS (for version)	
	time	orth	time	orth	time	orth
$m = 5$						
$m = 10$						
$m = 20$						
$m = 50$						
$m = 100$						

for-version: version with for-loop

MV-version: version with a matrix vector product

Use only MV version for this table:

	single GS		double GS		triple GS	
	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 1 minute on your computer (or you run out of memory). Try to make each version as efficient as possible.

(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 $K_m := \left[b, Ab/\|Ab\|, \ldots, A^{m-1}b/\|A^{m-1}b\|\right] \in \mathbb{R}^{n\times m}$.

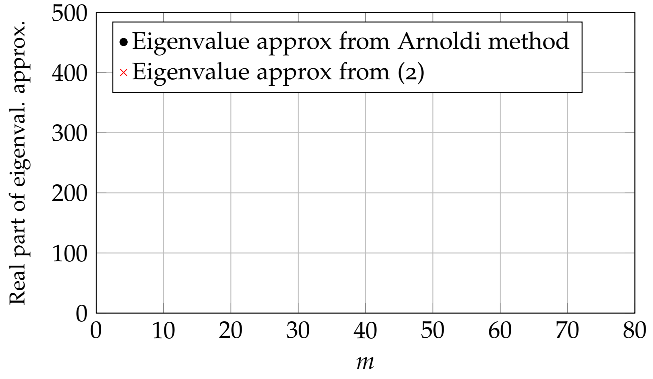
(a) We will let eigenvalues of A be approximated by eigenvalues of the matrix $(K_m^T K_m)^{-1} K_m^T A K_m$:

$$\mu w = (K_m^T K_m)^{-1} K_m^T A K_m w \tag{2}$$

You may assume that $K_m^T K_m$ is invertible. Prove that the approximation (2) is identical to the approximation generated by Arnoldi's method for eigenvalue problem. You may use any theorem in the course.

Hint: ALW wiki 1-40 of HT22: `http://jarlebring.ddns.net/~jarl/active_learning/active_learning.php?id=8&contents=2`

(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. Note that the matrix in the right-hand side is $m \times m$ which is a small matrix and you can use `eig`. Apply this method to (1) with `nn=30` 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. Use the matrix in the previous exercise.



Note that $K_m^T A K_m$ has m eigenvalues, so the plot will consist of several curves/points.

(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

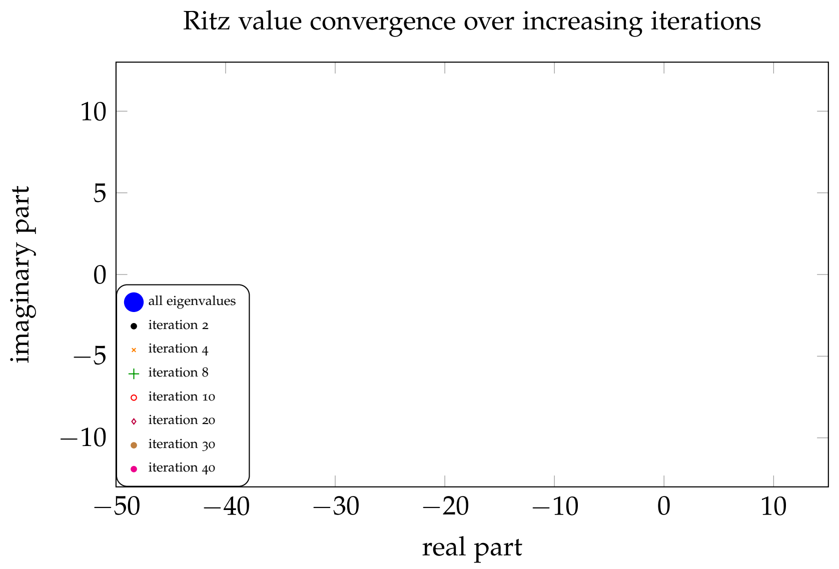
MATLAB users: Use the command `load('Bwedge.mat')` to access the matrix B and eigenvalues.

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

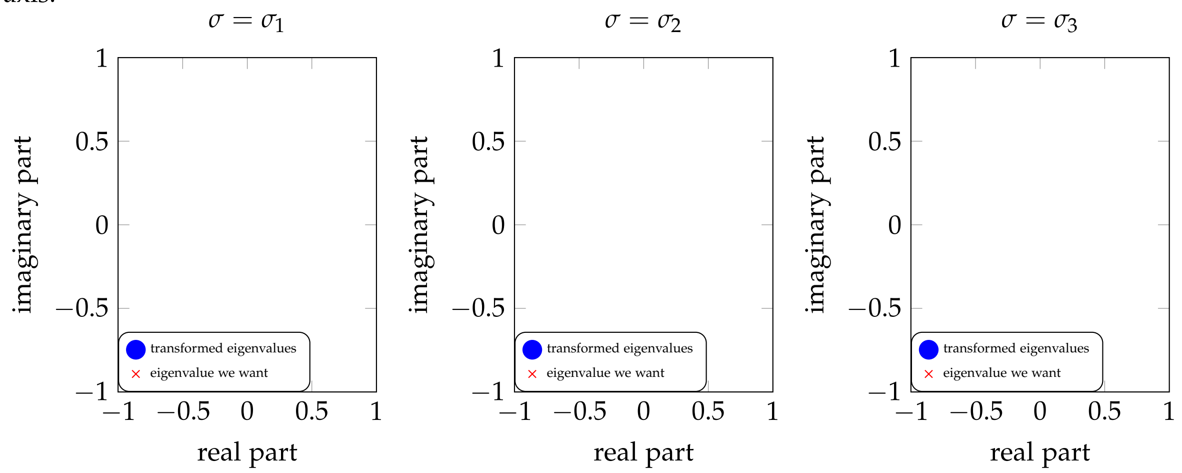
(b) In the same figures as in (a), indicate circles and estimate a convergence factor. The convergence factor is here meant $\alpha < 1$ such that the error $\sim \alpha^m$. One circle for each eigenvalue.

Julia users: see file instructions CANVAS → SF2524 → Modules → Julia instructions

(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}. You can visualize the Ritz values in this type of plot.



(d) This problem concerns the shift-and-invert Arnoldi method. The matrix has an eigenvalue close to $\lambda_* \approx -9.8 + 2i$, that we want to compute. In this case a precise value of the eigenvalue can be found in the `B_eigvals` variable. We want to study how the convergence depends on the shift. Plot in three separate figures the transformed eigenvalues; that is the eigenvalues of $(A - \sigma I)^{-1}$ for the σ-values given in the table below. Mark in the figures the location of the λ_* after the transformation. For this task you do not need to compute eigenvalues, but only transform given data. The figures can look like below, but you may need to adjust the axis.



(e) Implement shift-and-invert Arnoldi method. Use the one-vector as a starting vector. Fill out the following table with the with the eigenvalue errors for specific the eigenvalue we want: the difference between the exact eigenvalue (λ_*) and the closest computed approximation. Relate what you see in the table to the figures you plotted in task (d).

	Standard AM	$\sigma_1 = -10$	$\sigma_2 = -7+2i$	$\sigma_3 = -9.8+1.5i$
$m = 10$	2.5			
$m = 20$	0.6			
$m = 30$	$2 \cdot 10^{-4}$			

MATLAB hint: If you want to find the element in a vector xv which is closest to a given value z you can use `min` with two return parameters:

```
>> [~,I]=min(abs(xv-z));
>> xclosest=xv(I);
Julia hint:  (The I variable is
reserved for identity)
julia> i=argmin(abs.(xv-z));
julia> xclosest=xv[i];
```

6. You will find video material "SF2524 -> Video mateiral in this course". Summarize in your own words. You may use AI-level 1, but not higher. Approximately 100 words per video:
   (a) 101 Block 1 applications
   (b) 140 Rayleigh-Ritz method
   (c) 169 Breakdown
   (d) 168 Shift-and-invert Arnoldi method for eigenvalue problems
7. (A reminder. Not an actual exercise.) Please note that the "Week X" quizzes are mandatory, and are ideally completed the corresponding week but not later than one week after the exam.
8. Eigenfaces and iterative eigenvalue methods. We use the *Olivetti Faces* dataset (400 grayscale images, each 64×64 pixels).
   https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
   MAT-file: https://canvas.kth.se/courses/59848/files.
   Load and visualize it as follows:

```
load('olivetti_faces.mat')
imagesc(faces(:,:,345)); colormap("gray"); axis image;
```

Preprocessing. The variable `faces` has size $64 \times 64 \times 400$; technically a tensor where each slice `faces(:,:,k)` is one image. Construct the data matrix

$$X = [\, x_1, x_2, \ldots, x_m \,] \in \mathbb{R}^{n \times m}, \quad n = 64^2,\ m = 400.$$

The reshape command can compute the vectorization of each slice
`X = reshape(faces, 64*64, 400);`

Each column of X contains the pixel intensities of one face.

Compute and subtract the mean image:

```
x_mean = mean(X,2);
Xc = X - x_mean;
```

Context: This technique is known as principal component analysis (PCA), typically computed via the singular value decomposition (SVD). The SVD can be obtained using the Arnoldi method, but here we compute eigenvalues of C directly to illustrate the use of iterative methods and matrix vector products.

Define the symmetric matrix

$$C = \frac{1}{m-1} X_c X_c^T \in \mathbb{R}^{4096 \times 4096}.$$

This matrix describes how pixel intensities vary across images. Its dominant eigenvectors, corresponding to the largest eigenvalues, are called *eigenfaces*.

(a) Load the data and display a face of your choice.

(b) Determine the sizes of X_c and C. Compare computation times for:

(1) Forming $C = X_c X_c^T$ given X_c.

(2) Computing Cb for 100 different vectors b, given C.

(3) Computing $X_c(X_c^T b)$ for 100 different b, given X_c.

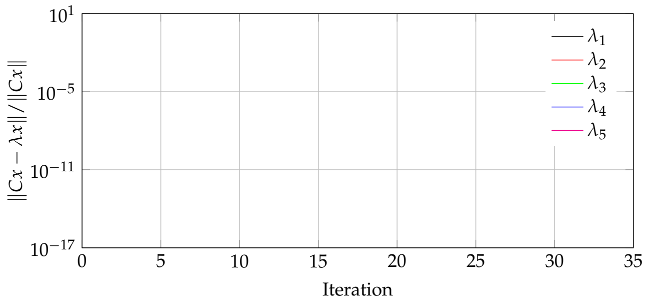
(c) **Power Method.** Use the power method to find the largest eigenpair (λ_1, v_1) of C. Plot the eigenvalue error $|\lambda_k - \lambda_*|/|\lambda_*|$ versus iteration k in a semilog plot, where λ_* is a reference solution. Start with a vector of ones and perform 30 iterations. Compare two implementations: one based on (b.1-2) and one based on (b.3). Which one is faster? Display the eigenface corresponding to the largest eigenvalue.

(d) **Arnoldi Method.** Use the Arnoldi method to compute the five largest eigenvalues $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$ and corresponding eigenvectors, using the efficient matrix-vector product. We will use the relative residual to quantify the error

$$\frac{\|Cx - \lambda x\|}{\|Cx\|}. \qquad (3)$$

We use a relative error measure in (3) since both $\|C\|$ and λ_1 are large; absolute errors may be misleading.

Plot the convergence (relative residual vs. iteration) on a semilog scale and display the computed five eigenfaces.



(e) Discuss the difference of the methods in (c) and (d). Which is preferable?

(f) **Halloween special.**🎃 Among the first 20 eigenfaces, which one looks the spookiest? Use any method to compute the eigenfaces. You might get a spookier face if you plot the negative of an eigenvector.

The following problems are only for *FSF3580 Numerical linear algebra* (PhD students).

- If you have taken SF2524 as a master student: You do not need to solve the SF2524 part above, but you need to solve all of the problems below.
- Otherwise: You need to solve the SF2524-part above, but may skip one of the questions 1-4. Additionaly, it is sufficient to solve one of the problems below.

9. 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`

10. SVD: Singular value decomposition. If you have not heard of SVD, you may need to look it up, in particular the Eckart-Young theorem.

    (a) What are the relationship between the singular values of the matrix $A \in \mathbb{R}^{n\times m}$ and the eigenvalues and eigenvectors of

$$B = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}.$$

    (b) Download the movie `india_driving1` from `https://canvas.kth.se/groups/303121/files/folder/fsf3580`. Reshape the movie to a matrix. Compute the SVD (with `svd(A)`) and construct the best rank one approximation

$$\tilde{A} = u_1 s_1 v_1^T$$

    According to Eckart-Young theorem, this corresponds to the largest

Note that even if the eigenvector of B is normalized, the two components correspond to singular vectors need to be normalized in a post-processing step.

See CANVAS "Julia instructions" for how to load and manipulate videos frames in Julia.

singular value and corresponding vector. Visualize and interpret the first column of $\tilde{A}$ as an image. Explain (briefly) why this is image looks this way.

(c) Download the larger movie file `market`. Reshape the movie to a matrix. The `svd` does not work on this matrix (on most computers) since the matrix is too large and the `svd`-algorithm runs out of memory. Use (a) and the Lanczos method to compute the singular vector and solve visualize as in (b).