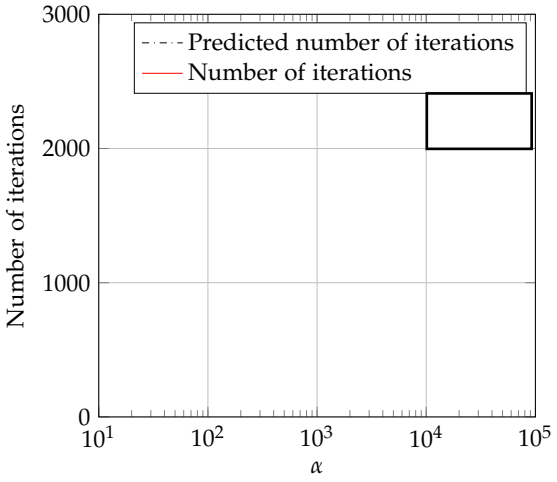
Homework 3

0. 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?"

1. **Exercise about basic QR-method.** Implement the basic QR-method. Apply it to `alpha_example.m` from the course web page. Measure the error with the maximum value below the diagonal `errfun=@(A) max(max(abs(tril(A,-1))))`.

   (a) Plot the number of iterations required to achieve error 10^{-10}, as a function of α. More precisely, generate the following plot (with `semilogx()`)



   (b) Suppose the eigenvalues are ordered by magnitude $|\lambda_1| < \cdots < |\lambda_m|$. From the lecture notes we know that the elements below the diagonal will asymptotically after n iterations be proportional to $|\lambda_i/\lambda_j|^n$ with $i < j$. For large α the error will be dominated by one particular choice of i and j. Which ones?

   (c) Use (b) to establish an estimated number of iterations required to reach a specified tolerance, for different choices of α. Add a plot of the predicted number of iterations in the plot generated in (a), for tolerance 10^{-10}, and discuss the result.

For the theoretical reasoning in (b) and (c) you may use the function `eig`

Hint for (c): Show that if the error behaves as $e_k = |\beta|^k$, then $e_N = \texttt{TOL}$ if $N = \ln(\texttt{TOL})/\ln(|\beta|)$.

2. **Exercises about Hessenberg reduction and shifted QR-method.**

   (a) Generalize the lemma about Householder reflectors in the lecture notes as follows. Given a vector $x \in \mathbb{R}^n$ and a vector $y \in \mathbb{R}^n$ with $y \neq 0$ and $x \neq 0$, derive a formula for a Householder reflector (represented by a normal direction $u \in \mathbb{R}^n$) such that $Px = \alpha y$ for some value α.

   Hint for (a): First derive a formula first for the case $\|y\| = 1$.

   (b) Implement a naive (inefficient) Hessenberg reduction by completing the program `naive_hessenberg_red.m` on the course web page.

   (c) Implement Algorithm 2 in the lecture notes and compare the computation time with the algorithm in (b). Carry out the comparison by computing a Hessenberg reduction of `A=alpha_example(1,m)`, which generates an $m \times m$-matrix. Complete the following table.

	CPU-time Algorithm 2	CPU-time of algorithm in (b)
m=10		
m=100		
m=200		
m=300		
m=400		

   (d) Let $\bar{H}$ be the result of one step of the shifted QR-method with shift σ for the matrix

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

   Run the shifted QR for two different choices of σ and complete the following table

ε	$\lvert\bar{h}_{2,1}\rvert$	
	$\sigma = 0$	$\sigma = a_{2,2}$
0.4	0.0961	0.0769
0.1		
0.01		
0.001		
10^{-4}		
10^{-5}		
10^{-6}		
10^{-7}		
10^{-8}		
10^{-9}		
10^{-10}		
0		

   Interpret the result in the table. What do the values in the table correspond to? Which choice of σ is better in this case?

   Hint: What does the shifted QR-method reduce to when you select $\sigma = 0$?

3. Download the template `schur_parlett.m` from the course web page.

(a) Complete the template code and compute $\sin(A)$ where

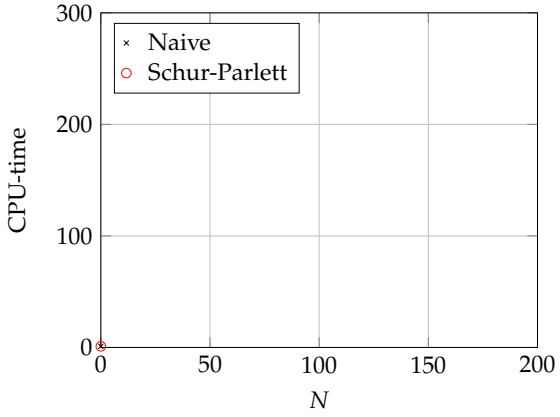
$$A = \begin{bmatrix} 1 & 4 & 4 \\ 3 & -1 & 3 \\ -1 & 4 & 4 \end{bmatrix}$$

(b) Let A be defined by `A=rand(100,100)+1im*rand(100,100); A=A/norm(A);`. Use the Schur-Parlett method from (a) to compute A^N, and compare with the naive method to compute A^N:

```
B=A; for i=1:N-1; B=B*A; end.
```

Complete the following figure. Increase N until you see that the best method changes, or see a tendency regarding which method will be better for large N.

The MATLAB command `A^N` for large N will actually not do the naive method. For large N MATLAB switches and uses an underlying procedure similar to Schur-Parlett.



Make appropriate sampling of the x-axis in the figure to support the conclusion, for instance $N = 10, 50, 100, 150, 200, \ldots$.

(c) What are the theoretical computational complexities of the two methods in (b) as a function of the size of the matrix n and N?

In other words, find p and q such that the number of flops $\sim \mathcal{O}(n^p N^q)$ for the two methods.

4. Let $A = \begin{bmatrix} \pi & 1 \\ 0 & \pi + \varepsilon \end{bmatrix}$ where $\varepsilon > 0$.

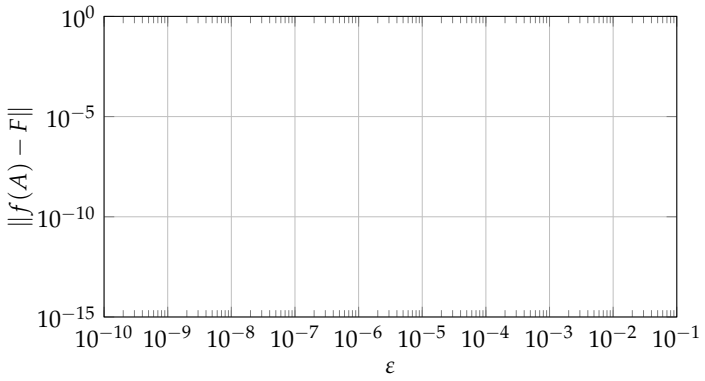
(a) Prove, for instance using the Jordan definition, for a general diagonalizable matrix, that if p is a polynomial which interpolates g in the eigenvalues of A, then $p(A) = g(A)$.

Find exact expressions for α and β when $p(z) = \alpha + \beta z$ for the matrix A above. That is, find a formula for α and β in terms of g, evaluated in the eigenvalues of A.

Due to noise in CPU-timing and memory issues, some sizes in problem (3b) may get spikes, where the method requires substantially more CPU-time.

(b) Give a formula for the exact value of $\exp(A)$ using (a).

(c) Let F be the result of computing $\exp(A)$ with the Jordan definition as in the last example in Section 4.1.2. Compare the exact result in (b) with the computed value for different ε. Generate the following figure (using `loglog()`) and explain the result.



Sample the x-axis such that you get equidistant point distribution in the log-log scale, for example $ev = 10.^(-10 : 0.1 : 1)$.

5. Scaling-and-squaring. We will now combine scaling-and-squaring with a Taylor approximation. Let the A-matrix be constructed like this

```
n=256; randn('seed',0);
A=randn(n,n);
A=A/norm(A); A=A-3*eye(n);
```

Consider the Taylor approximation

$$T_m(A) = \sum_{k=0}^{m} \frac{1}{k!} A^k$$

which can be computed with $m-1$ matrix-matrix products. Let j be the number of matrix-matrix products in the squaring phase of scaling and squaring.

Call the resulting approximation $P_{j,m}$.

(a) Complete Table 1 and Table 2.

Table 1: The total number of matrix-matrix products. This table can be filled out without any computation/implementation.

	j=0	j=1	j=2	j=3	j=4	j=5	j=6	j=7
m=1	0	1	2					
m=2								
m=3								
m=4								
m=5								
m=6								
m=7								

Table 2: $\log \|P_{j,m} - \exp(A)\|$, where `exp` is the matrix exponential and log is the 10-base logarithm (`log10` in matlab)

	j=0	j=1	j=2	j=3	j=4	j=5	j=6	j=7
m=1								
m=2			-1.4032					
m=3								
m=4								
m=5								
m=6								
m=7								

(b) Complete the following table. Hint: In your implementation, pay particular attention to only doing $m - 1$ matrix-matrix product when you evaluate T_m.

Table 3: CPU-time

	j=0	j=1	j=2	j=3	j=4	j=5	j=6	j=7
m=1								
m=2								
m=3								
m=4								
m=5								
m=6								
m=7								

(c) Discussion. Based on your tables, if you want an error smaller than 10^{-12} how many matrix-matrix products are needed? If we would only do Taylor approximation (which corresponds to the first column), how many multiplications would be needed? How much CPU-time?

6. Summarize in your own words (2-5 sentences for each video). You may use AI-level 1, but not higher.

   (a) 312: Householder reflectors that map x to the direction of e1

   (b) 325 QR-factorization of Hessenberg via Givens derivation

   (c) 333: Shifted QR-method with exact shifts

   (d) 344 Convergence of QR-method. Part 1. Simultaneous iteration: Normalized unnormalized (Week 4 quiz 5)

   (e) 345 Convergence of QR-method. Part 2. Convergence unnormalized simultaneous iteration for special case (Week 4 quiz 6)

   (f) 423 Newton SQRTM

   (g) Any other video in block 3-4.

7. In the course file area you have access to a mysterious data-file: `mysterious_images.mat`. You have the information that each column in the matrix corresponds to an image, vectorized in the usual way. The width and height are also given.

```
W=68; H=82;
x=X(:,5); myimage=reshape(x,W,H); % Load image 5
imagesc(myimage); % Plot the image
```

   (a) Pick a favoriate image and plot it.

   (b) Make an animation where you plot one image after another, and see what is going on. As an answer to this subquestion, explain what the video shows - no need to plot the animation in your report.

   (c) Do some detective work. What is this data set? Google might help, for example Google lens. When was this data-set created? What music should be played in the background? Hint: They are "8-bit images".

   (d) Compute the Estrada index of the graph with the adjacency matrix

   `A=create_adjacency(X,0)`

   Plot the 5 most central images and plot the 5 least central images in the sense of Estrada index.

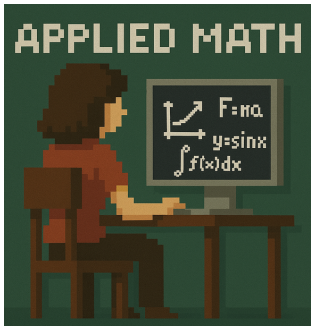
   (e) In problem (d) you need to compute the matrix exponential. Instead of computing the matrix exponential, with the builtin command, now use the truncated Taylor series. Select the truncation such that you are only doing 2 matrix-matrix multiplications. Answer:

   - Compare the computational time in d) and e). You need to run your code several times and form an average
   - How accurate is your method in (e)? If you order the images by centrality, which image is the first where the methods are different?

   (f) Optional. Use different cut-off values and visualize the graph. `plot(graph(A))`. `cutoff` describes if we should mark remove some image connections and it is the second parameter in the `create_adjacency` function. Use cutoff=0,0.5,0.6,0.7,0.9.

For problem (b): You may want to add a command drawnow in each for-loop in order for the image to be displayed.

What you need to know about Estrada index is explained in the slides of the first lecture for block 4.



Artistic low-resolution images is nowadays known as "pixel art".

In problem (e), you might ask: What is the highest degree polynomials that you can use with $m = 2$ multiplications? This question and its generalization for general m is the topic of current research - of great importance for example in the optimization for deep learning https://arxiv.org/pdf/2505.16932

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

8. **Exercise about exploitation of structure in specific application.** The purpose of this exercise is to learn some techniques to derive more efficient methods by taking problem-specific structure into account. (The new method you will derive is not necessarily in general the best for this problem-type.)

   (a) Prove that

   $$\frac{d}{dt}\exp(tA) = A\exp(tA) = \exp(tA)A$$

   (b) Let $G(t) := \exp(-tA)B\exp(tA)$ and let $[\cdot,\cdot]$ denote a commutator, i.e., $[A,B] := AB - BA$. Show that

   $$G(t) = B + t[B,A] + \frac{t^2}{2!}[[B,A],A] + \frac{t^3}{3!}[[[B,A],A],A] + \cdots \quad (*)$$

   Connection with current research: In the field of quantum chemistry, the relation (*) for $t = 1$ is commonly called the Baker-Campbell-Hausdorff expansion. It is fundamental in one of the leading numerical methods in that field - the so-called coupled cluster approach.

   (c) Suppose A is anti-symmetric $A^T = -A$. Let

   $$P := \int_0^\tau \exp(tA^T)B\exp(tA)\,dt$$

   Derive an expression for P involving commutators of A and B.

   The quantity P is called a Gramian, and it is often used in system and control in order to study controllability, observability and to derive optimal control as well as carrying out "model order reduction".

   (d) Let $C_k = [C_{k-1}, A]$ with $C_0 = B$. Show that $\|C_k\| \leq 2^k\|A\|^k\|B\|$.

   (e) Suppose $\|A\| < \frac{1}{2}$ and $t \leq 1$. Let G_N be the truncation of G,

   $$G_N(t) := B + t[B,A] + \cdots + \frac{t^N}{N!}[\cdots[[B,A],A]\cdots,A].$$

   Derive a bound for $\|G_N(t) - G(t)\|$, which converges to zero as $N \to \infty$ for any $t \in [0,1]$.

   (f) Combine (c)-(e) and derive a numerical method to compute P where A is anti-symmetric and $\|A\| < 1/2$. Construct the algorithm such that the user can specify a tolerance.

   (g) Compare your numerical method with the naive numerical integration approach:

   ```
   P=integral(@(t) expm(t*A')*B*expm(t*A),0,T,'arrayvalued',true);
   ```

   Use $\tau = 1$ and the matrices generated by:

   ```
   A=gallery('neumann',20^2); A=A-A'; A=A/(2*norm(A,1));
   B=sprandn(length(A),length(A),0.05);
   ```

   The MAT-file for "gallery Neumann" is available in the course files area. How much better is the new method?

   Not a part of the exercise: Can you derive a similar algorithm which does not require the matrix to be anti-symmetric?

9. Will appear in a later version