Final exam		
EPFL	**Cours:**	Numerical Analysis - MA
	Teacher:	Prof. Daniel Kressner
	Date:	06.07.2023 **Duration:** 3h00
Sciper:	**Name:**	

Table of points					
Exercise 1	**Exercise 2**	**Exercise 3**	**Exercise 4**	**Exercise 5**	**Exercise 6**
/2	**/2**	**/2**	**/2**	**/7**	**/7**
Exercise 7	**Exercise 8**	**Total**			
/5	**/5**	**/32**			

Please read the following instructions carefully

- All your calculations, derivations, and results must be written on this document which will be given back to the examiner at the end of the exam. In particular, write down all the steps of your calculation/derivation and provide justification.
- Please, write everything with a blue or black pen (a pencil or a red pen are not allowed).
- Questions 1–4 are quiz questions. For these questions clearly mark the correct answer by circling it and provide a brief justification. (No points are deducted for wrong answers.)
- Please do not unstaple the copy of your exam.
- The quality of the presentation is taken into account.
- You may respond in English or French (but you have to use one of the two languages for the entire exam).
- For questions that concern Matlab/Python, please choose one of the two programming languages. Choose the same programming language for the entire exam.
- We provide you with scratch paper for your personal notes.
- You may bring one A4 page (both sides) of handwritten notes with you. All other material, including electronic devices, are forbidden.

Write your sciper and name on every sheet!

Sciper: **Name:**

Exercise 1 (2 points)

Which of the following expressions will return the most accurate approximation (when implemented in the order indicated) of $y = \frac{1}{x}(e^x - e^{-x})$ in double precision floating point arithmetic for small $x > 0$ (such as $x = 10^{-8}$)? Briefly justify your choice.

a) $z_1 = e^x$, $z_2 = e^{-x}$, $z_3 = z_1 - z_2$, $y = z_3/x$

b) $z_1 = \sinh(x)$, $z_2 = 2z_1$, $y = z_2/x$

c) $z_1 = e^{-2x}$, $z_2 = 1 - z_1$, $z_3 = z_2/x$, $y = z_3 e^x$

d) $z_1 = \log(x)$, $z_2 = e^x/e^{z_1}$, $z_3 = e^{-x}/e^{z_1}$, $y = z_2 - z_3$

Sciper: **Name:**

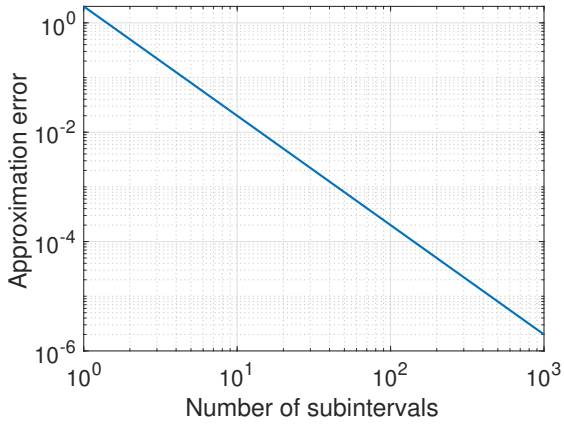
Exercise 2 (2 points)

Consider the integral

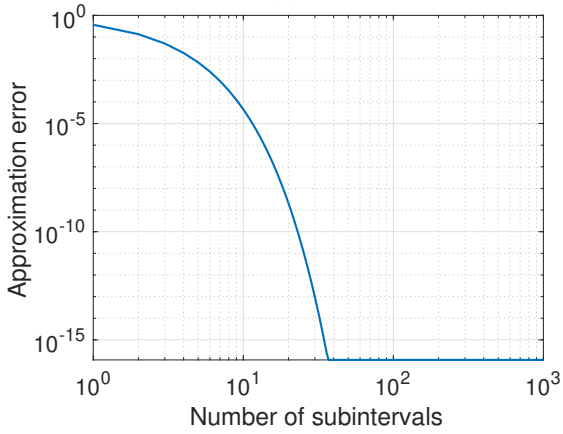
$$I = \int_{-1}^{1} (-2(x+1)^3 + 3(x+1)^2)\,\mathrm{d}x.$$

Let $Q_h^{(2)}$ denote the result of the composite Simpson rule on N subintervals applied to this integral, where $h = 2/N$. Which of the following graphs shows the quadrature error $|I - Q_h^{(2)}|$ vs. N? Briefly justify your choice.

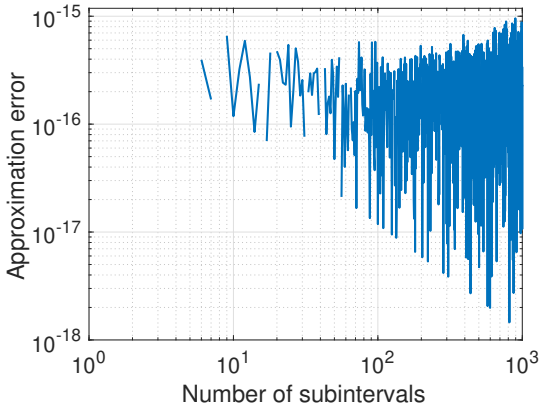
(a)



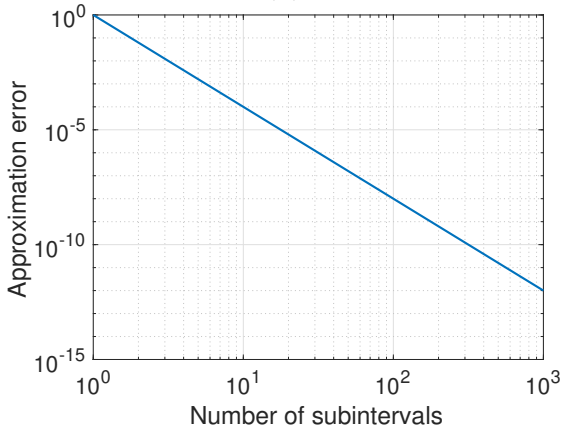
(b)



(c)



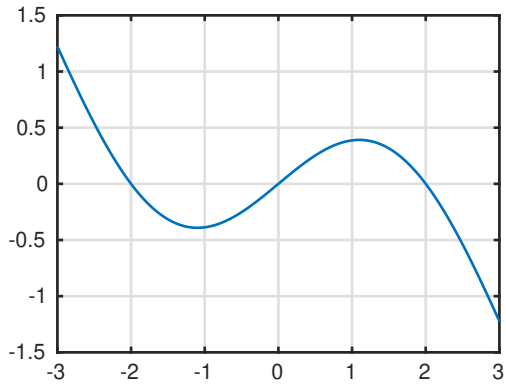
(d)



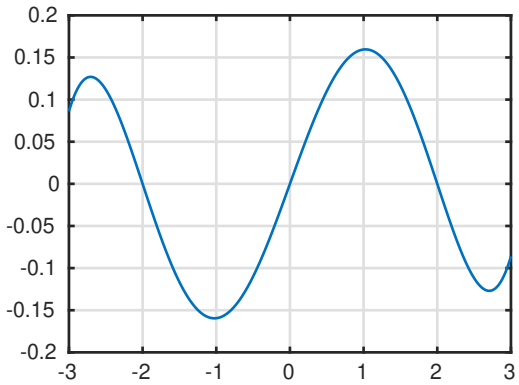
Exercise 3 (2 points)

Given the function $f(x) = \sin(x)$, let p_2 denote the quadratic polynomial that interpolates f at the interpolation nodes -2, 0, and 2. Which of the following four plots shows the error function $\text{error}(x) := f(x) - p_2(x)$? Briefly justify your choice.

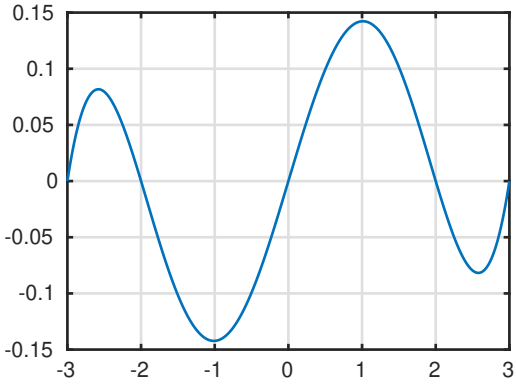
(a)



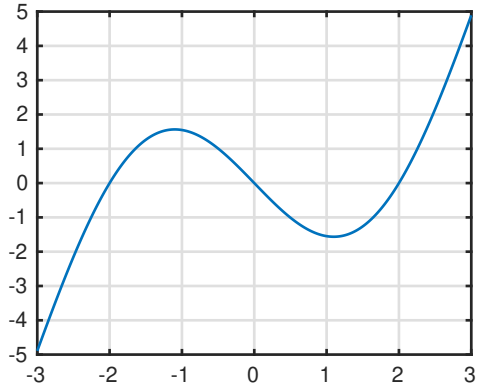
(b)



(c)



(d)



Exercise 4 (2 points)

We consider the linear system $A\mathbf{x} = \mathbf{b}$ with

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}.$$

For which values of $\alpha \in \mathbb{C}$ does the Richardson method with $P = I$ and initial vector $\mathbf{x}^{(0)} = \mathbf{0}$ converge for this linear system?

a) The Richardson method converges if and only if α is real and $0 < \alpha < 2$.

b) The Richardson method converges if and only if α takes the form $\alpha = \mathrm{i}\beta$ with $\beta \in \mathbb{R}$ and $\beta \neq 0$.

c) The Richardson method only converges for $\alpha = 0$.

d) The Richardson method does not converge for any $\alpha \in \mathbb{C}$.

Briefly justify your choice.

Exercise 5 (7 points)

Consider a function f and its integral over $[-1, 1]$

$$I[f] = \int_{-1}^{1} f(x)\,\mathrm{d}x$$

Consider the following quadrature rule to approximate $I[f]$:

$$Q_{[-1,1]}[f] = \frac{4}{3} f\left(-\frac{1}{2}\right) - \frac{2}{3} f(0) + \frac{4}{3} f\left(\frac{1}{2}\right).$$

a) Determine the order of $Q_{[-1,1]}$.

b) By noting

$$\int_c^d f(x)\,\mathrm{d}x = \frac{d-c}{2} \int_{-1}^{1} f\left(\frac{d-c}{2}x + \frac{c+d}{2}\right)\,\mathrm{d}x$$

derive from $Q_{[-1,1]}$ the corresponding quadrature rule $Q_{[c,d]}$ for a general interval $[c, d]$ with $c < d$.

c) The Matlab/Python function on the following two pages aims at implementing the *composite* quadrature rule corresponding to the quadrature rule from point b) but it misses some parts. Fill the missing lines of the code where indicated in the Matlab or the Python version.

Sciper: **Name:**

Matlab

```
function result = myint( a, b, N, f )
    % MYINT Composite quadrature rule
    %   myint( a, b, N, f ) approximates the integral of the function f
    %   in the interval [a b] using N subintervals.


    % Size h of subintervals
    h = (b - a) / N;


    % Compute extremes and midpoint of the subinterval
    x = linspace(a, b, 2*N + 1);

    % Integration nodes given by the midpoint of the subintervals
    x_mid = x(2:2:end);

    % Integration nodes given by the midpoint of the left half of the
    % subintervals
    % INSERT YOUR CODE HERE









    % Integration nodes given by the midpoint of the right half of the
    % subintervals
    % INSERT YOUR CODE HERE









    % Carry out composite rule
    % INSERT YOUR CODE HERE









    % END OF CODE
```

Sciper: **Name:**

Python

```
import numpy as np

def myint(a, b, N, f):
    # MYINT Composite quadrature rule
    # myint(a, b, N, f) approximates the integral of the function f
    # in the interval [a b] using N subintervals.

    # Size h of subintervals
    h = (b - a) / N

    # Compute extremes and midpoint of the subinterval
    x = np.linspace(a, b, 2*N + 1)

    # Integration nodes given by the midpoint of the subintervals
    x_mid = x[1::2]

    # Integration nodes given by the midpoint of the left half of the
    # subintervals
    # INSERT YOUR CODE HERE









    # Integration nodes given by the midpoint of the right half of the
    # subintervals
    # INSERT YOUR CODE HERE









    # Carry out composite rule
    # INSERT YOUR CODE HERE









    return result
```

Sciper: **Name:**

Exercise 6 (7 points)

Given a linear system $A\mathbf{x} = \mathbf{b}$ and a parameter ω, the *weighted Jacobi method* takes the form

$$x_i^{(k+1)} := \omega\left(b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}\, x_j^{(k)}\right)\Big/ a_{ii} + (1-\omega)x_i^{(k)}, \quad i = 1, \ldots, n.$$

for $k = 0, 1, 2, \ldots$ and a starting vector $\mathbf{x}^{(0)}$.

a) Show that the weighted Jacobi method can be written as a splitting method, that is, there is a splitting $A = P - N$ such that the weighted Jacobi method can be written as

$$\mathbf{x}^{(k+1)} = P^{-1}(N\mathbf{x}^{(k)} + \mathbf{b})$$

for $k = 0, 1, 2, \ldots$.

b) Let A be symmetric positive definite. Show that the weighted Jacobi method converges to $\mathbf{x}$ for any starting vector $\mathbf{x}^{(0)}$ if and only if $0 < \omega < \frac{2}{\lambda_{\max}(D^{-1}A)}$, where $\lambda_{\max}(\cdot)$ denotes the largest eigenvalue of a matrix. *Note:* You may use the fact that the eigenvalues of $D^{-1}A$ are real and positive.

c) Show that the weighted Jacobi method with $\omega = 2/3$ converges to $\mathbf{x}$ when applied to the symmetric positive definite $n \times n$ matrix

$$A = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & -1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}$$

for any $\mathbf{b}$ and any $\mathbf{x}^{(0)}$.
Hint: Use part b) and techniques from the lecture to bound the eigenvalues of a matrix.

Exercise 7 (5 points)

a) For given data $t_1, \dots, t_m \in \mathbb{R}$, $y_1, \dots, y_m \in \mathbb{R}$, we consider the problem of determining parameters α, β that solve the following minimization problem:

$$\min_{\alpha,\beta\in\mathbb{R}} \sum_{i=1}^{m} (\alpha e^{t_i} + \beta - y_i)^2 + \alpha^2 + \beta^2$$

Determine a matrix $A \in \mathbb{R}^{m\times 2}$ and a vector $\mathbf{b} \in \mathbb{R}^m$ such that this minimization problem is equivalent to

$$\min_{\mathbf{x}\in\mathbb{R}^2} \|A\mathbf{x} - \mathbf{b}\|_2^2 + \|\mathbf{x}\|_2^2.$$

b) We now consider the more general problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|_2^2 + \|\mathbf{x}\|_2^2. \tag{1}$$

for an arbitrary matrix $A \in \mathbb{R}^{m\times n}$ and a vector $\mathbf{b} \in \mathbb{R}^m$.
(i) Show that the linear system

$$(A^\top A + I)\mathbf{x} = A^\top \mathbf{b} \tag{2}$$

has a unique solution for *any* matrix A and vector $\mathbf{b}$.
(ii) Show that $\mathbf{x}$ minimizes (1) if and only if it solves (2).
Note: You may assume, if necessary, that $f(\mathbf{x}) := \|A\mathbf{x} - \mathbf{b}\|_2^2 + \|\mathbf{x}\|_2^2$ is convex.

c) The following Matlab/Python code aims at using the (economic) QR factorization to solve (1), but it contains one mistake. Identify and explain what is wrong in the code. Correct the Matlab or the Python function such that it works properly; you may directly write into the code below.

Matlab

```
[Q,R] = qr( [A; eye(n)], 0 );


x = R \ ( Q'*b );
```

Python

```
import numpy as np
Q, R = np.linalg.qr(np.concatenate((A, np.eye(n))), 'reduced')


x = np.linalg.solve(R, np.transpose(Q) @ b))
```

Exercise 8 (5 points)

The Discrete Hartley Transform (DHT) of a vector $\mathbf{y} = \begin{pmatrix} y_0 & y_1 & \cdots & y_{n-1} \end{pmatrix}^\top \in \mathbb{R}^n$ is the vector $\mathbf{h} = \begin{pmatrix} h_0 & h_1 & \cdots & h_{n-1} \end{pmatrix}^\top \in \mathbb{R}^n$ defined by

$$h_k = \sum_{j=0}^{n-1} y_j \left[\cos\left(\frac{2\pi}{n} jk\right) + \sin\left(\frac{2\pi}{n} jk\right) \right], \qquad k = 0, \ldots n-1.$$

a) Let $\mathbf{z} = F_n \mathbf{y}$, where F_n is the $n \times n$ matrix with entries $f_{kj} = \exp(-\frac{\mathrm{i}2\pi}{n} jk)$ for $j, k = 0, \ldots, n-1$. Show that

$$\begin{aligned} z_k &= \frac{h_k + h_{n-k}}{2} + \mathrm{i}\frac{h_{n-k} - h_k}{2}, \quad k = 0, \ldots, n-1, \\ h_k &= \mathsf{Re}\big((1+\mathrm{i})z_k\big), \quad k = 0, \ldots, n-1, \end{aligned}$$

where Re denotes the real part of a complex number.

b) Based on part a), write a (short) Matlab or Python code that carries out the DHT in $O(n \log n)$ operations. You may use any of the algorithms and Matlab/Python functions discussed in the lecture.