Mock Exam (Solution)		
EPFL	**Course:** Numerical Analysis - Sections SIE-GC **Lecturer:** Daniel Kressner **Date:** 19/12/2024 **Duration:** 1h	
Sciper:	**Student:**	**Section:**

Do not turn the page before the start of the exam. Read carefully the instructions below.

EXAM RULES

- Write everything with a blue or black pen.
- Please write your surname, name, and sciper on **EVERY PAGE** of this document!
- All Python code and all the results (the plots and images if they are requested, too) **MUST BE TRANSCRIBED ON THESE SHEETS**, which must be submitted at the end of the exam.
- It is **NOT** possible to submit the Python/Jupyter code electronically.
- Scratch paper is provided for your personal notes, but it will not be considered for the evaluation of the exam.
- Exercise 1 is of single-choice type. It is the only exercise on this exam where you do not need to justify your answer.

AUTHORIZED MATERIAL

The only authorized material is **1 A4 cheat sheet (front and back) handwritten with pen/pencil**. No other sheets, notes or books (paper or electronics), or calculator, mobile phone, tablet, laptop or other electronic devices. The access to Internet (e-mail, websites) is prohibited.

VIRTUAL MACHINE LOGIN INSTRUCTIONS

Log-in credentials:

Username and password: `Your GASPAR account`

Virtual machine:

`SB-MATH-WIN11`

Exercise 1 - Single-choice questions

This is a single-choice exercise. **One, and only one, answer is correct for every question**. Clearly mark your answer choice with a cross.

Question 1 We want to find an approximate solution of the ordinary differential equation

$$\begin{cases} \dfrac{\mathrm{d}u(t)}{\mathrm{d}t} = -2u(t) + (t+1) \\ u(0) = u_0, \end{cases}$$

with the following scheme:

$$\begin{cases} u^{n+1} = \dfrac{1}{2\Delta t + 1} u^n + \Delta t \dfrac{(n+1)\Delta t + 1}{2\Delta t + 1} \\ u^0 = u_0. \end{cases}$$

Which scheme is it?

☐ Forward Euler (explicit).

☒ Backward Euler (implicit).

☐ Crank-Nicolson.

☐ Heun.

☐ None of the above.

Solution: Solve the backward finite differences approximation $\frac{u^{n+1}-u^n}{\Delta t} = -2u^{n+1} + (t_{n+1}+1)$ *for* u^{n+1} *where* $t_{n+1} = (n+1)\Delta t$.

Question 2 Consider the data

$$\begin{array}{ll} x_1 = -1 & y_1 = 12 \\ x_2 = 0 & y_2 = 12 \\ x_3 = 2 & y_3 = 6. \end{array}$$

The derivative at $x = 1$ of a quadratic polynomial (degree 2) that interpolates the data (x_i, y_i), $i = 1, 2, 3$, is

☒ -3

☐ 0

☐ 1

☐ 5

☐ 11

Solution: The Lagrange polynomial is $p(x) = y_1\phi_1(x) + y_2\phi_2(x) + y_3\phi_3(x) = 12\frac{x(x-2)}{3} + 12\frac{(x+1)(x-2)}{-2} + 6\frac{(x+1)x}{6} = -x^2 - x + 12$, *hence,* $p'(1) = -3$.

Question 3 Consider the first-order linear differential equation system

$$\frac{\mathrm{d}\mathbf{u}(t)}{\mathrm{d}t} = A\mathbf{u}(t) + \mathbf{b}(t), \quad \text{with} \quad A = \begin{pmatrix} -1 & 3 & 2 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}. \tag{1}$$

The forward Euler method to approximate the problem (1) is

☐ absolutely stable if and only if $0 < \Delta t < \frac{4}{5}$

☐ absolutely stable if and only if $0 < \Delta t < \frac{1}{2}$

☐ absolutely stable if and only if $0 < \Delta t < 1$

☐ absolutely stable if and only if $\Delta t > 0$ (unconditionally absolutely stable)

☒ never absolutely stable (not stable for any $\Delta t > 0$)

Solution: Two eigenvalues of A are $\lambda = \pm i$, meaning $|1 + \lambda \Delta t| > 1$ for all $\Delta t > 0$. See proof of Lemma 6.4 (lecture notes) for details.

Question 4 Let $f : (0, \infty) \to \mathbb{R}$ be defined by $f(x) = (x-1)^3 + 2\log(x) + 1$. We are interested in the unique fixed point $\alpha = 1$ of f. Under what condition on ω is the composite fixed point method defined for $\omega \in \mathbb{R}$ by

$$x^{(k+1)} = \phi(x^{(k)}) = (1-\omega)x^{(k)} + \omega f(x^{(k)})$$

of order 2?

☒ $\omega = -1$

☐ $\omega = 0$

☐ $\omega = 1$

☐ $\omega = 1/2$

☐ none of the values of ω proposed above

Solution: We have $\phi'(x) = (1-\omega) + \omega(3(x-1)^2 + \frac{2}{x})$. Therefore $\phi'(1) = 0 \iff \omega = -1$. Since for $\omega = -1$, $\phi^p(1) \neq 0$ for all $p \geq 2$, the method is of order 2 by Theorem 1.2 (lecture notes) if $\omega = -1$.

Exercise 2 - Implementation

Question 1 Complete the following Python function which implements the Heun method.

```
def heun(f, u_0, T, N):
    """
    Solves the ordinary differential equation
        u' = f(t,u), t in (0, T],
        u(0) = u_0
    using Heun's method on an equispaced grid of stepsize dt = T/N.

    Parameters
    ----------
    f : function
        The function whose zero we search
    u_0 : float
        Starting value.
    T : float > 0
        End point.
    N : int
        The number of sub-intervals

    Returns
    -------
    u : list or NumPy array
        Approximation of the solution [u_0, u_1, u_2,...,u_N]
    """

    dt = T / N
    ### BEGIN SOLUTION
    u = np.zeros(N + 1)
    u[0] = u_0
    ### END SOLUTION

    for n in range(N):
        ### BEGIN SOLUTION
        u[n + 1] = u[n] + dt / 2 * f(n * dt, u[n]) + dt / 2 * f((n + 1) * dt, u[n] + dt
            * f(n * dt, u[n]))
        ### END SOLUTION
    return u
```

Question 2 Complete the following Python function which implements the bisection method.

```python
def bisection(f, a, b, tol):
    """
    Finds a zero of the continuous function f in the interval [a, b] with
    the bisection method. The function f must be such that f(a) and f(b)
    have opposite signs.

    Parameters
    ----------
    f : function
        The function whose zero we search
    a : float
        Start point of the search interval [a, b]
    b : float > a
        End point of the search interval [a, b]
    tol : float > 0
        Tolerance to be used.

    Returns
    -------
    alpha : float
        The computed zero.
    niter : int
        Number of iterations.
    """

    alpha = a  # current approximate root
    k_min = int(np.ceil(np.log2((b - a) / tol) - 1))  # number of iterations needed
    x_k = (a + b) / 2  # mid-point of current search interval
    for k in range(k_min):
        ### BEGIN SOLUTION
        if f(x_k) * f(a) < 0:
            b = x_k
        else:
            a = x_k
        x_k = (a + b) / 2
        ### END SOLUTION

    alpha = x_k
    niter = k + 1

    return alpha, niter
```