Exam		
EPFL	**Course:**	Numerical Analysis and Computational Mathematics
	Lecturer:	Dr. Rafael Vázquez Hernández
	Date:	13/01/2020 **Duration:** 3h 00 min
Sciper:	**Student:**	**Section:**

Scores		
Presence/Answer	**Exercises**	**Total**
/1.00	**/5.00**	**/6.00**

Exercise 1	Exercise 2	Exercise 3	Exercise 4	Total
/20.00	**/17.00**	**/24.00**	**/24.00**	**/85.00**

PLEASE READ CAREFULLY ALL THE INSTRUCTIONS OF THE EXAM

- All the computations and the results, including the graphs, must be copied on this document to be returned to the examiner at the end of the exam. Report all the steps of your computations and provide justifications of the results.
- All the MATLAB commands used to solve the exercises must be completely reported on this document. Questions answered with just the final result or graph, incomplete justifications, computations or MATLAB commands will be considered incorrect.
- Please do not remove the staple from this document, and write everything with a blue or black pen. Answers written with a pencil will not be accepted.
- Scratch paper is provided for your personal notes, but it will not be considered for the evaluation of the exam. It must be returned at the end of the exam.
- All the electronic devices, other than the computer available in the room, are forbidden. The only software allowed during the exam is MATLAB, and a PDF reader for the MATLAB tutorial.
- A paper in A4 format with hand-written notes on a single side is allowed. Photocopies, copies of exercises and other written material are forbidden.
- All the necessary functions and files are in your Desktop. You are logged in using a local account. Do not log off until the end of the exam.

PLEASE SIGN THE FOLLOWING DECLARATIONS

- I hereby declare to have received the MATLAB functions and files:

  `I2c.m`, `newton.m`, `multistep.m`,`backward_euler_template.m`.

- I hereby declare that the computer, keyboard, mouse, the operative system and MATLAB are operating correctly.

Sciper: **Student:** **Section:**

```
function [ Ih ] = I2c( fun, a, b, N )
%  I2c approximate the integral of a function in the interval
%  [a,b] by means of the composite quadrature formula I_2^c
%  defined in exercise 2, question 3.
%
%  [ Ih ] = I2c( fun, a, b, N )
%  Inputs: fun = function handle,
%          a,b = extrema of the interval [a,b]
%          N = number of subintervals of [a,b] of the same size, N≥1
%              (the case N=1 corresponds to the simple formula)
%  Output: Ih = approximate value of the integral

H = ( b - a ) / N;

x_k = linspace( a, b, N + 1 );
x_bar_k = ( x_k( 1 : end - 1 ) + x_k( 2 : end ) ) / 2;

f_x_bar_k = fun( x_bar_k );
f_x_m1 = fun( x_k( 1 : end - 1 ) );
f_x_p1 = fun( x_k( 2 : end ) );

Ih = H * sum( f_x_p1 + 4*f_x_bar_k + f_x_m1 ) / 6;

return
```

```
function [ tv, uv ] = multistep ( fun, y0, y1, y2, t0, tf, Nh )
% Multistep method for the scalar ODE in the form
% y'(t) = f(t,y(t)),  t \in (t0,tf)
% y(t0) = y0, y(t0+h) = y1, y(t0+2h) = y2; with h = (tf-th)/Nh
%
%  [ tv, uv ] = multistep ( fun, y0, y1, y2, t0, tf, Nh )
%  Inputs: fun        = function handle for f(t,y), fun = @(t,y) ...
%          y0, y1, y2 = initial values
%          t0         = initial time
%          tf         = final time
%          Nh         = number of time subintervals
%  Output: tv         = vector of time steps (1 x (Nh+1))
%          uv         = vector of approximate solution at times tv

tv = linspace( t0, tf, Nh+1 );
h = ( tf-t0 ) / Nh;

uv = zeros( 1, Nh+1 );
uv( 1:3 ) = [y0, y1, y2];

for n = 3 : Nh
    fn = fun( tv( n ), uv( n ) );
    fnm1 = fun( tv( n-1 ), uv( n-1 ) );
    fnm2 = fun( tv( n-2 ), uv( n-2 ) );
    uv( n+1 ) = uv( n ) + h * ( 23*fn - 16*fnm1 + 5*fnm2 ) / 12;
end

return
```


Sciper: **Student:** **Section:**

```
function [xvect, resvect, nit] = newton( fun, dfun, x0, tol, nmax )
% NEWTON Find a zero of a nonlinear scalar function.
%   [XVECT] = NEWTON(FUN,DFUN,X0,TOL,NMAX) finds a zero of the differentiable
%   function FUN using the Newton method and returns a vector XVECT containing
%   the successive approximations of the zero (iterates). DFUN is the derivative of FUN.
%   FUN and DFUN accept a real scalar input x and return a real scalar value;
%   FUN and DFUN can also be inline objects. X0 is the initial guess.
%   TOL is the tolerance on error allowed and NMAX the maximum number of iterations.
%   The stopping criterion based on the difference of successive iterates is used.
%   If the search fails a warning message is displayed.
%
%   [XVECT,RESVECT,NIT] = NEWTON(FUN,DFUN,X0,TOL,NMAX) also returns the vector
%   RESVECT of residual evaluations for each iterate, and NIT the number of iterations.
%   Note: the length of the vectors is equal to ( NIT + 1 ).

nit = 0;
xvect(nit+1) = x0;
resvect(nit+1) = fun( x0 );
err_estim = tol + 1;
while ( err_estim ≥ tol && nit < nmax )
    xvect(nit+2) = xvect(nit+1) − fun( xvect(nit+1) ) / dfun( xvect(nit+1) );
    resvect(nit+2) = fun( xvect(nit+2) );
    err_estim = abs( xvect(nit+2) − xvect(nit+1) ); % diff. successive iterates
    nit = nit + 1;
end

if err_estim ≥ tol
    warning(['Newton method stopped without converging to the desired tolerance, '...
             'the maximum number of iterations was reached.']);
end
return
```

```
function [ tv, uv ] = backward_euler( fun, dfun, y0, t0, tf, Nh )
% BACKWARD_EULER Backward Euler method for the scalar ODE in the form
% y'(t) = f(t,y(t)),  t \in (t0,tf)
% y(0) = y_0
%
%  [ tv, uv ] = backward_euler( fun, dfun, y0, t0, tf, h )
%  Inputs: fun    = function handle for f(t,y), fun = @(t,y) ...
%          dfun   = function handle for df(t,y), dfun = @(t,y) ...
%          y0     = initial value
%          t0     = initial time
%          tf     = final time
%          Nh     = number of subintervals
%  Output: tv     = vector of time steps (1 x (Nh+1))
%          uv     = vector of approximate solution at times tv

return
```


Sciper: **Student:** **Section:**

Exercise 1 (20 points)

Let $f : I = [a, b] \to \mathbb{R}$ a C^∞ function such that it admits a unique zero $\alpha \in I$, that is, $f(\alpha) = 0$.

a) Let us consider an iterative method for the approximation of α. Define the convergence order of the method, and the asymptotic convergence factor. Explain all the notation you use.

b) Describe the bisection method for the approximation of the zero of a function $f(x)$, and write the algorithm including the stopping criterion. Explain all the notation used.

c) Under which assumptions does the bisection method converge? What is the convergence order of the bisection method? Motivate your answers.

d) Describe the fixed point method for the approximation of a zero of a general differentiable function. Write the algorithm of the fixed point method, including a stopping criterion. Explain all the notation used.

e) Let the interval $I = [1, 2]$. The two functions $\Phi_1(x) = \frac{1}{2}\left(x + \frac{2}{x}\right)$ and $\Phi_2(x) = \frac{1}{x} + \frac{2}{x^3}$ have a fixed point at $\alpha = \sqrt{2}$. According to the theoretical results of convergence, which of the two functions is preferred to apply the fixed point method? Write explicitly the convergence theorem that justifies your answer.

Sciper: **Student:** **Section:**

Sciper: **Student:** **Section:**

Exercise 2 (17 points)

Let $a, b \in \mathbb{R}$ with $a < b$. Consider a function $f \in C^0([a, b])$.

a) Given the distinct nodes $x_0, x_1, x_2 \in [a, b]$, define the Lagrange interpolating polynomial of degree 2 associated to those nodes, called $\Pi_2 f$. Define the Lagrange characteristic polynomials associated to those nodes, and provide the expression of $\Pi_2 f$ in terms of the Lagrange characteristic polynomials. Explain all the notation you use.

b) Let us consider the equally spaced nodes $x_0 = a$, $x_1 = \frac{a+b}{2}$ and $x_2 = b$. Provide an upper bound for the interpolation error $e_2(f) := \max_{x\in[a,b]} |f(x) - \Pi_2 f(x)|$ that depends on $h := \frac{b-a}{2}$. Specify the assumptions on f.

c) For all $g \in C^0([a, b])$, let $I(g) := \int_a^b g(x)\,\mathrm{d}x$ and $I_2(g) := I(\Pi_2 g)$, computed with the equally spaced nodes of point b). What is the name of the quadrature formula given by I_2? What is the degree of exactness of this formula?

d) Let us consider I_2^c the composite version of the quadrature formula I_2. Write down its definition by explaining your notation. What is the convergence order of I_2^c?

e) Let $f(x) = \sin(3x)x^3$, for all $x \in [a, b] := [1, 4]$. We have $I(f) \approx -20.2206873008$. By using the given MATLAB function `I2c.m` that implements the quadrature formula I_2^c, verify the answer you gave in question d), that is, verify the convergence order of $I_2^c(f)$. Report the MATLAB code used and the results of the computations, including plots if needed.

f) Let again $[a, b] := [1, 4]$. Consider the quadrature formula defined by

$$I_w(g) = w_0 g(1) + w_1 g(2) + w_2 g(4),\ \forall g \in C^0([a, b]).$$

Find the values of the quadrature weights w_0, w_1 and w_2 such that I_w has degree of exactness equal to 2. You can use MATLAB to help you with the computation.

Sciper: **Student:** **Section:**

Sciper: **Student:** **Section:**

Sciper: **Student:** **Section:**

Exercise 3 (24 points)

We want to find $\mathbf{x} \in \mathbb{R}^n$, the solution of the linear system $A\mathbf{x} = \mathbf{b}$ with $A \in \mathbb{R}^{n \times n}$, $\mathbf{b} \in \mathbb{R}^n$ and $n \geq 1$.

a) Let A be the Hilbert matrix, given by the coefficients $a_{ij} = 1/(i + j - 1)$. Assign the matrix in MATLAB for $n = 7$. Is it possible to apply the LU factorization to A? And the Cholesky factorization? Motivate the answers.

b) According to the answer to point a), which one of the two factorization methods would you use to solve the linear system with the matrix A? Motivate the answer.

c) Explain all the necessary steps to solve the linear system with the factorization method selected in the previous point.

d) Define the relative error and the relative residual. Is the relative residual a good estimate of the error for the linear system considered above? Motivate the answer.

e) We consider the preconditioned gradient method, and we set as preconditioner the matrix $P = D$, the diagonal part of A, and as the first iterate we set $\mathbf{x}^{(0)} = \mathbf{0}$. We know that the exact solution is $\mathbf{x} = (1, 0, 0, 0, 1, 0, 0)^\top$. How many iterations should we compute to ensure that the absolute error is below 10^{-6}? Is the matrix D a good preconditioner for the Hilbert matrix? Motivate the answers.

f) Let us denote by E the matrix formed by the upper and lower diagonal of A. That is,

$$e_{ij} = \begin{cases} a_{ij} & \text{if } i = j + 1 \text{ or } i = j - 1, \\ 0 & \text{otherwise.} \end{cases}$$

Assign the matrix E in MATLAB. Then, study graphically which value of $\theta \in [0.25, 0.35]$ yields the fastest convergence of the preconditioned gradient method setting as preconditioner the matrix $P = \theta D + (1 - \theta)E$, and for any given $\mathbf{x}^{(0)}$.

Sciper: **Student:** **Section:**

Exercise 4 (24 points)

Let us consider the following Cauchy (initial value) problem:

$$\text{find } y : I \subset \mathbb{R} \longrightarrow \mathbb{R}, \text{ such that } \begin{cases} y'(t) = f(t, y(t)), & t \in I \\ y(t_0) = y_0, \end{cases}$$

with the interval $I = (t_0, t_f)$, the initial datum is $y_0 \in \mathbb{R}$ and a function $f : I \times \mathbb{R} \longrightarrow \mathbb{R}$. We assume that $f(t, y)$ is at least continuous in both arguments, and Lipschitz-continuous in the second argument.

a) Approximate the solution of the Cauchy problem by means of the backward Euler method, with a partition of the interval I into N_h subintervals of the same length h. Report the expression of the approximate solution at the general discrete time $t_n = t_0 + nh$, for $n \geq 0$. Is the method explicit or implicit? Motivate the answer.

b) For the model problem $f(t, y(t)) = \lambda y(t)$ with $\lambda < 0$, is the method conditionally or unconditionally absolutely stable? In the former case, report the stability condition in terms of the interval size h. Is the method $\mathcal{A}$-stable? Motivate the answers.

c) Using the function template `backward_euler_template.m`, and eventually the function `newton.m`, implement a MATLAB function to approximate the solution of the Cauchy problem with backward Euler method.

d) Given $u_0, u_1, \ldots, u_p$, multistep methods with $p + 1$ steps are given by the expression

$$u_{n+1} = \sum_{j=0}^{p} a_j u_{n-j} + h \sum_{j=-1}^{p} b_j f(t_{n-j}, u_{n-j}),$$

with real coefficients $\{a_j\}_{j=0}^{p}$ and $\{b_j\}_{j=-1}^{p}$. Under which condition is the multistep method given in the previous expression an implicit method?

e) The function `multistep.m` includes the implementation of a particular multistep method. Determine, using the expression above, the number of steps and the coefficients a_j and b_j of the implemented method.

f) Let us consider the Cauchy problem with

$$f(t, y) = \left(\lambda + \frac{1}{(1 + t^2)(1 + \arctan(t))} \right) y,$$

and initial condition $y_0 = 0$, and let us set $t_0 = 0$, $t_f = 3$. In this case the exact solution is given by

$$y(t) = (1 + \arctan(t)) e^{\lambda t}.$$

Let $\lambda = -1$. Compute the solution for interval sizes $h = \frac{1}{16}, \frac{1}{32}, \frac{1}{64}$ with the multistep method given in function `multistep.m`, giving the input values u_1 and u_2 as the exact solution, i.e., $u_1 = y(h)$, $u_2 = y(2h)$. Report the computed solution and the error at $t_f = 3$, for the three values of h. By using the computed errors, estimate the order of convergence of the method, either algebraically or graphically.

g) Assuming that the exact solution is not known, which method would you use to compute the values u_1 and u_2 of the multistep method? Motivate the answer.

Sciper: **Student:** **Section:**

Sciper: **Student:** **Section:**

