Numerical Analysis and Computational Mathematics

Fall Semester 2024 – CSE Section

Prof. Laura Grigori

Assistant: Israa Fakih

Session 13 – December 11, 2024

Ordinary differential equations

Exercise I (Theoretical)

Discuss the existence and uniqueness of the solution $y(t)$ of the Cauchy problem

$$\text{find } y : [0,\infty) \to \mathbb{R} \quad : \quad \begin{cases} y'(t) = -\arctan(y) & \text{for all } t > 0, \\ y(0) = 1, \end{cases} \tag{1}$$

Exercise II (MATLAB)

Consider the Cauchy problem

$$\text{find } y : I \subset \mathbb{R} \to \mathbb{R} \quad : \quad \begin{cases} y'(t) = f(t, y(t)) & \text{for all } t \in I, \\ y(t_0) = y_0, \end{cases} \tag{2}$$

where $I = (t_0, t_f)$ is the integration interval, $f : I \times \mathbb{R} \to \mathbb{R}$ is a given continuous function, and $y_0 \in \mathbb{R}$ is the initial datum.

a) Write the MATLAB function `runge_kutta_4.m` that implements the explicit 4-stage Runge-Kutta method RK4. Use the function `runge_kutta_4_template.m` as template.

```
function [ tv, uv ] = runge_kutta_4( fun, y0, t0, tf, Nh )
% RUNGE_KUTTA_4 Runge-Kutta 4, explicit method for the scalar ODE in the
% form:
% y'(t) = f(t,y(t)),  t \in (t0,tf)
% y(0) = y_0
%
%  [ tv, uv ] = runge_kutta_4( fun, y0, t0, tf, Nh )
%  Inputs: fun    = function handle for f(t,y), fun = @(t,y) ...
%          y0     = initial value
%          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
%

return
```

b) Use the functions `forward_euler.m`, `heun.m` (from Series 12), and `runge_kutta_4.m` to solve problem (2) for $f(t,y) = \left(\frac{\cos(t)}{\alpha+\sin(t)} + \beta\right) y$, $y_0 = \alpha$, $t_0 = 0$, $t_f = 30$, $\alpha = 1.5$, and $\beta = -0.2$. Set $N_h = 30$ and compare the numerical solutions with the exact solution $y(t) = (\alpha + \sin(t))\, e^{\beta(t-t0)}$.

c) Compute the errors $e_n^{FE} = \left|y(t_n) - u_n^{FE}\right|$, $e_n^{H} = \left|y(t_n) - u_n^{H}\right|$, and $e_n^{RK4} = \left|y(t_n) - u_n^{RK4}\right|$. Select n such that the computed errors correspond to time $\bar{t} = 10$ for increasing values of the number of subintervals $N_h = 30, 60, 120, 240, 480, 960$. Plot the computed errors vs h and graphically deduce the convergence orders of the methods.

d) Repeat point b) to solve the Cauchy problem corresponding to $f(t,y) = \lambda y$, with $\lambda = -0.525$, $t_0 = 0$, $t_f = 40$, and $y_0 = 1$. Vary the number of subintervals N_h and try to identify empirically which values of h yield (absolutely) stable numerical methods.

Exercise III (MATLAB)

Consider a system of first order ODEs:

$$\text{find } \mathbf{y} \ : I \subset \mathbb{R} \to \mathbb{R}^m \qquad : \qquad \begin{cases} \mathbf{y}'(t) = \mathbf{F}(t, \mathbf{y}(t)) & \text{for all } t \in I, \\ \mathbf{y}(t_0) = \mathbf{y}_0, \end{cases} \tag{3}$$

where $m \geq 1$, $I = (t_0, t_f)$ is the integration interval, $\mathbf{F} \ : I \times \mathbb{R}^m \to \mathbb{R}^m$ is a given vector-valued function, and $\mathbf{y}_0 \in \mathbb{R}^m$ is the initial datum. If $\mathbf{F}$ is of the form

$$\mathbf{F}(t, \mathbf{y}) = A\,\mathbf{y} + \mathbf{g}(t), \tag{4}$$

for some matrix $A \in \mathbb{R}^{m \times m}$ and a vector-valued function $\mathbf{g} \ : \ I \to \mathbb{R}^m$, then the system of ODEs (3) is said to be linear and non-homogeneous with constant coefficients.

a) Write the MATLAB function `forward_euler_system.m` that implements the *forward Euler* method for the solution of (3). Use the function `forward_euler_system_template.m` as template.

```
function [ tv, uv ] = forward_euler_system( fun, y0, t0, tf, Nh )
% FORWARD_EULER_SYSTEM Forward Euler method for solving a system of ODEs
% in the form:
% y'(t) = F(t,y(t)),   t \in (t0,tf)
% y(0) = y_0
%
% y, y_0 are vectors of size (m x 1)
%
%  [ tv, uv ] = forward_euler_system( fun, y0, t0, tf, Nh )
%  Inputs: fun     = function handle for F(t,y), fun = @(t,y) ...
%                    a vector of size (m x 1) must be returned by fun
%          y0      = initial vector of size (m x 1)
%          t0      = initial time
%          tf      = final time
```

```
%          Nh     = number of time subintervals
%  Output: tv     = vector of time steps (1 x (Nh+1))
%          uv     = matrix of the approximate solution at times tv
%                   size (m x (Nh+1))


return
```

b) Write the MATLAB function `backward_euler_system_nhcc.m` that implements the *backward Euler* method for the solution of a problem (3) that is linear and non-homogeneous with constant coefficients, see (4). Use the function `backward_euler_system_nhcc_template.m` as template.

```
function [ tv, uv ] = backward_euler_system_nhcc( A, g, y0, t0, tf, Nh )
% BACKWARD_EULER_SYSTEM_NHCC Backward Euler method for solving a system of
% ODEs in the nonhomogeneous with constant coefficients form:
% y'(t) = A y(t) + g(t),  t \in (t0,tf)
% y(0) = y_0
%
% y, y_0 are vectors of size (m x 1)
%
%  [ tv, uv ] = backward_euler_system_nhcc( A, g, y0, t0, tf, Nh )
%  Inputs: A      = square matrix of size (m x m)
%          g      = function handle for g(t), g = @(t) ...
%                   a vector of size (m x 1) must be returned by g
%          y0     = initial vector of size (m x 1)
%          t0     = initial time
%          tf     = final time
%          Nh     = number of time subintervals
%  Output: tv     = vector of time steps (1 x (Nh+1))
%          uv     = matrix of the approximate solution at times tv
%                   size (m x (Nh+1))

return
```

c) Use the functions `forward_euler_system.m` and `bacward_euler_system_nhcc.m` to solve (3), with $\mathbf{F}$ of the form (4), where $A = -\begin{bmatrix} 3 & 1 \\ 1 & 1 \end{bmatrix}$, $\mathbf{g}(t) = \mathbf{0}$, $y_0 = (1, 0.6)^T$, $t_0 = 0$, and $t_f = 5$. Set $N_h = 25$ and plot the numerical solutions.

d) Repeat point c) with smaller values of N_h. Discuss the results obtained in terms of the absolute stability of the forward and backward Euler methods.