© *Jean-Baptiste APOUNG KAMGA <jean-baptiste.apoung@math.u-psud.fr>*

Fiche de TP: Summary of commands

Exercice-1 : Basic notions of Matlab and Linux

Q-1-1 : Write a **`Matlab`** script which

- reads in a terminal an integer n.
  (*Matlab command* **`fscanf`**)
- generate the files **`courbe_k.eps, k=1,...,n`** corresponding to the curves of functions

$$x \longmapsto \sin\left(\frac{\pi}{k}x\right), \quad k = 1, \ldots, n$$

  (*Matlab command* **`print`**)

Q-1-2 : By using a terminal,

- convert the files **`courbe_k.eps, k=1,...,n`** to **`.jpg`** and **`.png`** format.
  (*unix command* **`convert`**)
- assign only read access to the converted files.
  (*unix command* **`chmod`**)
- Put the name of the resulting files in the files **`liste_png.dat`** and **`liste_jpg.dat`** in such a way that the line number k of **`liste_png.dat`** (resp. **`liste_jpg.dat`**) is **`courbe_k.png`** (resp. **`courbe_k.jpg`**).
  (*unix commands* **`for, echo`** *and unix redirections* **`>, >>`**).

Exercice-2 : Solving Nonlinear Equation

Q-2-1 : Define through a **`Matlab`** file a function

$$\begin{cases} \mathbb{R} \times \mathbb{R}^2 & \rightarrow & \mathbb{R}^2 \\ (t, Y) & \mapsto & f(t, Y) \end{cases}$$

Q-2-2 : Make sure the selected function has a root for some $t \in \mathbb{R}$.

Q-2-3 : Find with the help of the Matlab function **`fsolve`**, the approximate values of the roots of $f(t, \cdot)$, for various values of t.

Q-2-4 : Modify the parameters of the function **`fsolve`** such that it makes a maximum ($nmax$) number of iterations, and such that it displays the norms of the residual at each iteration.
(*Check matlab command* **`optimset`**).

Exercice-3 : Solving ODE in Matlab

We wish to solve numerically the following ordinary differential equation which models a damped pendulum.

$$\ddot{x}(t) = -\frac{\eta}{m}\dot{x}(t) - \frac{k}{m}x(t), \qquad x(0) = a, \dot{x}(0) = b,$$

where the position of the pendulum is given by x , m is the mass of the pendulum, k is the spring constant (linear) and η the friction coefficient (linear).

Q-3-1 **:** Write the equation as a system of first order ordinary differential equations.

$$\dot{u}(t) = g(t, u(t)), \quad u(0) = u_0.$$

where u, u_0 and $g(\cdot, \cdot)$ are to be found.

Q-3-2 **:** Write an backward Euler scheme to solve this equation.
Écrire le schémas d'Euler implicite pour la résolution de l'équation ci-dessus.
Conclude that at each step t_n, we must solve a nonlinear equation $f(t_n, U) = 0$.

Q-3-3 **:** Write a **`Matlab`** script for solving the problem. (You must create graphics for some instants). The following values will be taken : $k/m = 1.5, \eta/m = 0.5, u_0 = (0, 2)$, the final time shall be $T = 20s$.

Exercice-4 : Basic Notions of LaTeX

Q-4-1 **:** Write one page (or two pages maximum) of a pdf document (A4 format) to present your results. (add graphics if possible).

Q-4-2 **:** Write one or two slides maximum using **`beamer`** to describe your results.

Exercice-5 : Try to understand what Matlab does in ode45

For this exercise we supply the following files

Listing 1: Exact solution file: exact.m

```
function res = exact(x)
 res = (- 2500.0*exp(-50.0*x) + 2500.0*cos(x) + 50.0*sin(x))/(2501.0);
end
```

Listing 2: Second membre fichier: f.m

```
function [res] = f(x,y)
 res = 50.0 *(cos(x) - y);
 end
```

Listing 3: Adaptive Euler file: adaptiveEuler.m

```
function [t,x] = adaptiveEuler(t0,x0,tol,h0,tf,scheme)
 t=t0;
 x =x0;
 tnex=t;
 xnex=x0;
 hnex=h0;
while tnex < tf
   hnext = min(hnext, tf - t(end));
  [tnex,xnex,hnex,success] = scheme(tnex,xnex,hnex,tol);
  if(true == success)
    t = [t,tnex];
    x = [x,xnex];
  end
end
```

Listing 4: For adaptive Euler file 1: myEuler.m

```
function [t,x,h,success] = myEuler(t0,x0,h0,tol)
%step one
xn = x0 + h0 * f(t0,x0);
%step two comparative solution
yn1 = x0 + (h0/2)  * f(t0,x0);
yn  = yn1 + (h0/2) * f(t0+h0/2,yn1);
% yn  = x0 + h0 * f(t0+h0/2, x0 + (h0/2) * f(t0,x0) );
%compare solution
```

```
err = abs(yn - xn);
if(err < tol)
  t = t0 + h0;
  h = 2.0*h0;
  x = xn;
  success = true;
else
  h = h0/2.0; % on peut faire mieux!
  t = t0;
  x = x0;
  success = false;
end
```

Note that a better step size selection can be performed by taking into account the order p of the one step method:

Listing 5: For adaptive Euler file 2: myEulerBetter.m

```
function [t,x,h,success] = myEulerBetter(t0,x0,h0,tol)
 %le schema d'euler est d'ordre 1 donc
  order = 1;
 %step one
  xn = x0 + h0 * f(t0,x0);
%step two comparative solution
  yn1 = x0 + (h0/2)   * f(t0,x0);
  yn  = yn1 + (h0/2) * f(t0+h0/2,yn1);
  % yn  = x0 + h0 * f(t0+h0/2, x0 + (h0/2) * f(t0,x0) );
%compare solution
  [t,x,h,success] = myStepperController(t0,x0,h0,tol,xn,yn,order);
end
```

Listing 6: Adaptive Euler file:myStepperController.m

```
function [t,x,h,success] = myStepperController(t0,x0,h0,tol,x1,y1,p)
% [t,x,h,success] = myStepperController(t0,x0,h0,tol,x1,y1,p)
% fonction qui calcule le pas pas d'un schéma d'ordre p
% En entree :
% t0   : instant de départ pour le calcul de la solution x1
% x0   : solution de départ pour le calcul de x1
% h0   : pas utilisé pour calculer la solution x1
% x1   : solution calculée que l'on devra ou pas accepter
% tol  : tolérance permetant de valider la solution x1
% y1   : solution permettant d'approcher "l'erreur de consistance" définie par err = ||y1 - x1||
% p    : ordre de consistance du schéma considéré
% En sortie:
% t    : le nouvel instant de départ
% x    : la nouvelle solution de départ
% h    : nouveau pas de départ
% success : booléen signalant si la solution fournie a été acceptée
scale = abs(y1 - x1)/tol;
S = 0.9;
if(scale > 1.1)
  scale = max(0.2, S / (scale^(1.0/p)));
  h  = scale * h0;
  t  = t0;
  x  = x0;
  success = false;
elseif (scale < 0.5)
  err = max(1., min(5.0, S / (scale^(1.0/(p+1)))));
  h = err * h0;
  t = t0 + h0;
  x = x1;
  success = true;
else
  h = scale * h0;
  t = t0 + h0;
  x = x1;
  success = true;
end
```

Q-5-1 : Test this program and compare it to other schemes such as forward and backward Euler schemes. *You can use the following file*

Listing 7: Simple Euler file:testode.m

```
function testode()
```

```
h = 1.00974/50;
x = 0:h:1;
%linspace(0,10,100);
res = exact(x);
n=max(size(x));
Euler_exp= zeros(n,1);
Euler_imp = Euler_exp;
Euler_exp(1) = 0;
Euler_imp(1) = 0;
%Euler explicit

for i=1:n-1
  Euler_exp(i+1) = h*((1.0/h - 50.0)*Euler_exp(i) -50.0*cos(x(i)));
  Euler_imp(i+1) = (h/(1.0+50.0*h))*(Euler_imp(i)/h +50.0*cos(x(i+1)));
end

figure(1)
plot(x,res,'-*', x,Euler_exp,'-+',x,Euler_imp,'-o');
legend('E','E_E','E_I');

%%%%

tol = 1e-2;
t0  = 0;
x0  = 0;
h0  = h;
tf  = 1;

figure(2)
[t,xa]   = adaptiveEuler(t0,x0,tol,h0,tf,@myEuler);
[te,xab] = adaptiveEuler(t0,x0,tol,h0,tf,@myEulerBetter);

plot(x,res,'-', t,xa,'-+',te,xab,'-o');
legend('E','E_adapt','E_adaptBetter');

end
```

Q-5-2 : Try to apply the step control on problem of Exercise 3. Try also on other first step methods. (*Seek help for embedded Runge-Kutta type methods*).

Thème - 1 *Applications*

We study the motion of a planet subject to the attraction of a star placed at the origin of the selected coordinate system. Denote by $(x(t), y(t))$ the coordinates of the mass center of the planet in the motion plane. The fundamental principle of dynamic reads

$$x''(t) = -GM\frac{x(t)}{(x^2(t)+y^2(t))^{\frac{3}{2}}}, \quad y''(t) = -GM\frac{y(t)}{(x^2(t)+y^2(t))^{\frac{3}{2}}}$$

where G is the gravitational constant and M the mass of the star. The unit measure is selected such that $GM = 1$ and we consider the following initial condition : $x(0) = 0.2, y(0) = 0, x'(0) = 0$ et$y'(0) = 3$. The path described by the planet is an ellipse of semi-axis 1 and 0.6, covered in a period of $T = 2\pi$.

Exercice-1 : Write the equations as a system of first order ordinary differential:

$$\left\{ \begin{array}{rcl} Y'(t) & = & F(Y(t)), \quad t > 0 \\ Y(0) & = & Y_0, \end{array} \right. \tag{1}$$

where $Y : \mathbb{R}^+ \to \mathbb{R}^4$ and $F : \mathbb{R}^4 \to \mathbb{R}^4$.

Exercice-2 : We wish to numerically solve the system on the domain $[0, T]$ (with $T > 0$ fixed). To this end, let $N \geq 1$ set $k = T/N$ and define $t_n = nk$. We compute the approximated value of the solution of (1) at instants t_n by the following Euler scheme:

$$Y_{n+1} = Y_n + kF(Y_n), \quad n = 0, \ldots, N-1$$

where $Y_0 = Y(0)$. Write a **matlab** program which uses that scheme to approximate the solution of (1) on $[0, 2\pi]$. Display on the same figure the approximated and the exact path.

Exercice-3 : Do the same with the following schemes.

$$Y_* = Y_n + \frac{1}{2}kF(Y_n), \quad Y_{n+1} = Y_n + kF(Y_*), \quad n = 0, \ldots, N-1$$

(midpoint method) and

$$\begin{cases} F_1 = F(Y_n), \quad F_2 = F(Y_n + \frac{1}{2}kF_1) \\ F_3 = F(Y_n + \frac{1}{2}kF_2), \quad F_4 = F(Y_n + kF_3) \\ Y_{n+1} = Y_n + \frac{k}{6}\left(F_1 + 2F_2 + 2F_3 + F_4\right), \quad n = 0, \dots, N-1. \end{cases}$$

(4 order Runge-Kutta).

Exercice-4 **:** Compare the performance of each those methods: by studying the influence of the step on the closed nature of the trajectory, estimate the convergence order of each method.

Exercice-5 **:** Compare the performance of the following exotic Runge-Kutta schemes on the previous problem.

$$(\text{RK4-CM:}) \begin{cases} F_1 = F(Y_n), \quad F_2 = F(Y_n + \frac{1}{2}kF_1) \\ F_3 = F(Y_n + \frac{kF_1}{12} + \frac{11kF_2}{24}), \quad F_4 = F(Y_n + \frac{kF_1}{12} - \frac{25kF_2}{132} + \frac{73kF_3}{66}) \\ Y_{n+1} = Y_n + \frac{2k}{9}\left(\frac{F_1^2+F_1F_2+F_2^2}{F_1+F_2} + \frac{F_2^2+F_2F_3+F_3^2}{F_2+F_3} + \frac{F_3^2+F_3F_4+F_4^2}{F_3+F_4}\right) \quad n = 0, \dots, N-1. \end{cases}$$

$$(\text{RK4-CoM:}) \begin{cases} F_1 = F(Y_n), \quad F_2 = F(Y_n + \frac{1}{2}kF_1) \\ F_3 = F(Y_n + \frac{kF_1}{8} + \frac{3kF_2}{8}), \quad F_4 = F(Y_n + \frac{kF_1}{4} - \frac{3kF_2}{4} + \frac{3kF_3}{2}) \\ Y_{n+1} = Y_n + \frac{k}{3}\left(\frac{F_1^2F_2^2}{F_1+F_2} + \frac{F_2^2F_3^2}{F_2+F_3} + \frac{F_3^2F_4^2}{F_3+F_4}\right) \quad n = 0, \dots, N-1. \end{cases}$$

$$(\text{RK4-HM:}) \begin{cases} F_1 = F(Y_n), \quad F_2 = F(Y_n + \frac{1}{2}kF_1) \\ F_3 = F(Y_n - \frac{kF_1}{8} + \frac{5kF_2}{8}), \quad F_4 = F(Y_n - \frac{kF_1}{4} + \frac{7kF_2}{20} + \frac{9kF_3}{10}) \\ Y_{n+1} = Y_n + \frac{2k}{3}\left(\frac{F_1F_2}{F_1+F_2} + \frac{F_2F_3}{F_2+F_3} + \frac{F_3F_4}{F_3+F_4}\right) \quad n = 0, \dots, N-1. \end{cases}$$

$$(\text{RK4-HeM:}) \begin{cases} F_1 = F(Y_n), \quad F_2 = F(Y_n + \frac{1}{2}kF_1) \\ F_3 = F(Y_n - \frac{kF_1}{48} + \frac{25kF_2}{48}), \quad F_4 = F(Y_n - \frac{kF_1}{24} + \frac{47kF_2}{600} + \frac{289kF_3}{300}) \\ Y_{n+1} = Y_n + \frac{k}{9}\left(F_1 + 2(F_1 + F_2) + F_4 + \sqrt{|F_1F_2|} + \sqrt{|F_2F_3|} + \sqrt{|F_3F_4|}\right) \quad n = 0, \dots, N-1. \end{cases}$$

Exercice-6 **: Extension.**

Q-6-1 **:** Search for **symplectic** type schemes (that is schemes that can preserve the area an the orientation) and try them on the considered problem.

Q-6-2 **:** Search and implement a scheme called **Exponential Rosenbrock**. Try versions with step control.