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

Fiche de TP: récapitulatif des commandes

Exercice-1 : Notions de base de Matlab et Linux

Q-1-1 : Écrire un script **`Matlab`** qui

- lit au terminal un entier n.
  (*Se renseigner sur la commande* ***`fscanf`***)
- génère les fichiers **`courbe_k.eps, k=1,...,n`** correspondant aux courbes des fonctions

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

(*Se renseigner sur la commande* ***`print`***)

Q-1-2 : En utilisant un terminal,

- convertir les fichiers **`courbe_k.eps, k=1,...,n`** au format **`.jpg`** et **`.png`**.
  (*Se renseigner sur la commande unix* ***`convert`***)
- attribuer uniquement les droits en lecture aux fichiers convertis.
  (*Se renseigner sur la commande unix* ***`chmod`***)
- mettre les noms des fichiers ainsi obtenus dans **`liste_png.dat`** et **`liste_jpg.dat`** de sorte que la ligne numéro k de **`liste_png.dat`** (resp. **`liste_jpg.dat`**) soit **`courbe_k.png`** (resp. **`courbe_k.jpg`**).
  (*Se renseigner sur les commandes unix* ***`for, echo`*** *et les redirections* **`>, >>`**).

Exercice-2 : Résolution des équations non-linaires

Q-2-1 : Définir à travers un fichier **`Matlab`** une fonction

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

Q-2-2 : Le choix de la fonction vous revenant, tester sa bonne définition en l'évaluant en quelques points. Assurez-vous, quitte à la modifier en conséquence, pour qu'elle admette une racine pour des $t \in \mathbb{R}$.

Q-2-3 : Déterminer à l'aide de **`fsolve`**, les valeurs approchées des racines de $f(t, \cdot)$, pour diverses valeurs de t.

Q-2-4 : Modifier les paramètres de **`fsolve`** pour que cette fonction fasse un nombre maximum ($nmax$) d' itérations, et affiche les normes des résidus à chaque itération.
(*Se renseigner sur la commande* ***`optimset`**).

Exercice-3 : Résolution d'EDO avec Matlab

On désire résoudre numériquement l'équation différentielle décrivant le mouvement d'un pendule amorti :

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

où la position du pendule est donnée par x , m est la masse du pendule, k la constante de raideur du ressort (linéaire) et η le coefficient de frottement (linéaire).

Q-3-1 : Mettre cette équation sous la forme

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

où u, u_0 et $g(\cdot, \cdot)$ sont à déterminer.

Q-3-2 : Écrire le schémas d'Euler implicite pour la résolution de l'équation ci-dessus.
Conclure qu'à chaque instant t_n, on est amené à résoudre une équation non linéaire $f(t_n, U) = 0$.

Q-3-3 : Écrire un script **Matlab** pour la résolution ce problème. (On créera des graphiques pour certains instants). On prendra les valeurs suivantes: $k/m = 1.5, \eta/m = 0.5, u_0 = (0, 2)$, le temps final sera $T = 20s$.

Exercice-4 : Notions de base de LaTeX

Q-4-1 : Écrire une page (ou deux pages maximum) d'un document .pdf (format A4) commentant vos résultats. (Mettre les courbes si possible).

Q-4-2 : Écrire deux pages maximum d'une présentation, commentant vos résultats. On utilisera **beamer** pour la génération de la présentation au format .pdf.

Exercice-5 : Essai de compréhension de ce que fait Matlab (ode45)

Pour cet exercice on propose les fichiers suivants

Listing 1: Solution exacte fichier: 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: Euler adaptatif fichier: 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
  [tnex,xnex,hnex,success] = scheme(tnex,xnex,hnex,tol);
  if(true == success)
    t = [t,tnex];
    x = [x,xnex];
  end
end
```

Listing 4: Pour Euler adaptatif fichier 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);
%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; %ceci n'est pas sophistique, on peut faire mieux!
  t = t0;
  x = x0;
  success = false;
end
```

On peut améliorer la détermination du pas en prenant en compte l'ordre p du schéma.

Listing 5: Pour Euler adaptatif fichier 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: Euler adaptatif fichier:myStepperController.m

```
[t,x,h,success] = myStepperController(t0,x0,h0,tol,y0,p)
% [t,x,h,success] = myStepperController(t0,x0,h0,tol,x1,y1,p)
% function 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 + h;
  x = x1;
  success = true;
else
  h = scale * h0;
  t = t0 + h;
  x = x1;
  success = true;
end
```

Q-5-1 : Tester ce programme et comparer ce schéma adaptatif aux schémas d'Euler explicite et implicite.*On pourra se servir du fichier*

Listing 7: Euler simple fichier: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 : En vous servant de la question précédente, appliquer le schéma de contrôle de pas à l' éxercice de l' oscillateur et à des variantes de schéma de votre choix.

Thème - 1 *Applications*

On étudie le mouvement d'une planète soumise à l'attraction d'un astre que l'on place à l'origine du repère choisi. Désignant par $(x(t), y(t))$ les coordonnées du centre de gravité de la planète dans le plan du mouvement, le principe fondamental de la dynamique s' écrit

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

où Gest la constante de gravitation et M est la masse de l' astre. On choisit les unités de mesure de façon que $GM = 1$ et on considère les données de Cauchy suivantes: $x(0) = 0.2, y(0) = 0, x'(0) = 0$ et$y'(0) = 3$. La trajectoire décrite par la planète est alors une ellipse de demi-axes 1 et 0.6, parcourue avec une période $T = 2\pi$.

Exercice-1 : Ecrire ces équations sous la forme d'un système différentiel d'ordre 1, de la forme

$$\begin{cases} Y'(t) &= F(Y(t)), \quad t > 0 \\ Y(0) &= Y_0, \end{cases} \tag{1}$$

où $Y : \mathbb{R}^+ \to \mathbb{R}^4$ et $F : \mathbb{R}^4 \to \mathbb{R}^4$.

Exercice-2 : On souhaite résoudre numériquement le système proposé sur l'intervalle $[0, T]$ ($T > 0$ étant fixé). Pour celà, on se donne $N \geq 1$ et on pose $k = T/N$ avec $t_n = nk$. On calcule la valeur approchée de la solution de (1) en les points t_n en utilisant le schéma d'Euler explicite, sous la forme

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

avec $Y_0 = Y(0)$. Ecrire un programme **matlab** qui permet d'effectuer la résolution de (1) à l'aide de ce schéma sur $[0, 2\pi]$. On pourra tracer la trajectoire approchée et la trajectoire exacte sur un même graphe.

Exercice-3 : Faire de même avec les deux autres schémas suivants (on initialise de la même façon en posant $Y_0 = Y(0)$) :

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

(schéma du point milieu) et

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

(schéma de Runge-Kutta d'ordre 4).

Exercice-4 : Comparer les performances respectives de ces schémas. On pourra étudier l'influence du pas de temps sur le caractère fermé de la trajectoire et retrouver numériquement l'ordre de chaque schéma.

Exercice-5 : On propose à présent plusieurs vatriantes de mise à jour de la solution dans le schéma de Runge-Kutta, selon la table ci-dessous. Comparer les sur le problème ci-dessus.

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

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

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

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

Exercice-6 : Allez plus loin.

Q-6-1 : Rechercher un schéma dit **symplectique** (c'est-à dire conservant l'aire et l'orientation) et évaluer le sur le problème considéré.

Q-6-2 : Rechercher et implémenter un schéma dit de **Rosenbrock exponentiel** et évaluer le sur le problème considéré. **On insistera sur la version avec contrôle de pas.**