Chapitre[3] Résolution numérique des équations différentielles

Samir Kenouche - Département des Sciences de la Matière - UMKB

Méthodes Numériques et Programmation

Résumé

Après un bref aperçu sur l'interprétation géométrique d'une équation différentielle. L'objectif principal de ce chapitre est l'implémentation de méthodes numériques pour la résolution des équations différentielles. Les méthodes abordées sont dites **à un pas**, car le calcul de $y(x_{n+1})$ ne réclame que la valeur de $y(x_n)$ à l'instant précédent. Une méthode **à deux pas** utilisera à la fois $y(x_n)$ et $y(x_{n-1})$. Nous nous bornerons aux méthodes numériques d'Euler explicite, de Heun, de Crank - Nicolson et de Runge-Kutta classique d'ordre 4. La dernière section est dédiée aux séances de travaux pratiques.

Mots clés

Méthode d'Euler, de Heun, Runge Kutta, script Matlab®.

Table des matières

I. Introduction

Une équation différentielle est une équation dont l'inconnue est une fonction $y(x)$. La forme générale d'une telle équation s'écrit :

$$f(y^{(n)}, y^{(n-1)}, y^{(n-2)}, ..., y^{(1)}, y, x) = \varphi(x) \tag{1}$$

Avec, $y^{(n)}$ est la nième dérivée de la fonction y et $\varphi(x)$ désigne le second membre de l'équation différentielle. Dans le cas où $\varphi(x) = 0$, on dira que l'équation différentielle est homogène. L'existence d'une solution unique de l'équation différentielle est tributaire de l'imposition de certaines conditions initiales sur $y(x)$ et ses dérivées. Dans l'équation (1), les conditions initiales sont les valeurs de $y(a), y^{(1)}(a), y^{(2)}(a), ..., y^{(n)}(a)$. Cependant, il faut noter que très souvent la solution analytique n'existe pas, et on doit par conséquent approcher la solution exacte $y(x)$ par des méthodes numériques ou bien trouver des courbes intégrales géométriquement.

Le but majeur de ce chapitre est la résolution numérique des équations différentielle. Néanmoins, il a été jugé utile de faire un bref aperçu sur l'interprétation géométrique d'une équation différentielle. Graphiquement, trouver une

S. Kenouche est docteur en Physique de l'Université des Sciences et Techniques de Montpellier et docteur en Chimie de l'Université A. Mira de Béjaia.
Site web : voir http://www.sites.univ-biskra.dz/kenouche

Document corrigé, amélioré et actualisé le 19.09.2019.

solution d'une telle équation revient à tracer une courbe tangente aux coefficients directeurs des droites tangentes en chaque point, ce qui donne un champ de pentes. Afin d'appréhender cette notion, considérons l'équation $y^{'}(x) = x^2$. A chaque fois qu'on donne une valeur à x on trouve $y^{'}(x)$. D'un point de vue géométrique $y^{'}(x)$ exprime la pente (vecteur directeur unitaire) de la droite tangente au point $x = x_0$, soit $y(x) = y^{'}(x_0)\,(x-x_0)+y(x_0)$. Par conséquent, en calculant $y^{'}$ pour chaque point d'un plan, on obtient un champ de pentes donc des courbes intégrales.

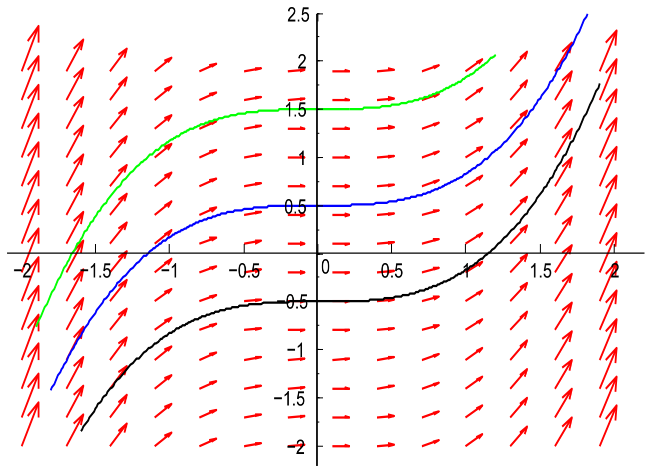

FIGURE 1: Champ de pentes associé à $y^{'}(x) = x^2$.

Par un point de coordonnées (x_0, y_0) ne passe qu'une seule solution. Cela signifie qu'il n'existe qu'une solution vérifiant la condition initiale $y(x_0) = y_0$. Notons également qu'une équation différentielle d'ordre un, possède une infinité de solutions. Ces solutions dépendent d'une constante $C \in \mathbb{R}$. Dans la figure ci-dessus, sont représentées trois solutions ou courbes intégrales, associées à $y^{'}(x) = x^2$, vérifiant les conditions initiales $y(1.15) = 1$ (courbe en couleur bleue), $y(1/4) = -1/2$ (courbe en couleur noire) et $y(1.20) = 2$ (courbe en couleur verte). La figure est obtenue selon ce script :

```
clear all ; close all ; clc ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Le 06/09/2019 - SAMIR KENOUCHE
% CHAMP DE PENTES ASSOCIE A y'(x) = x^2

u = -2 :0.30: 2 ; echelle = 1.2 ; pas = 0.5 ;
[x, y] = meshgrid(u) ; dy = x.^2 ; dx = ones(size(dy)) ;

quiver(x, y, dx, dy, echelle ,'color','r','LineWidth', 1) ;

c1 = 1/2 ; c2 = -1/2 ; c3 = 3/2 ;
figure(1) ; hold on ;

u1 = -1.8 :0.010: 1.9 ; plot(u1, (0.33.*u1.^3 + c1),'b') ; % SOLUTION 1
u2 = -1.6 :0.010: 1.9 ; plot(u2, (0.33.*u2.^3 + c2),'k') ; % SOLUTION 2
u3 = -1.9 :0.010: 1.2 ; plot(u3, (0.33.*u3.^3 + c3),'g') ; % SOLUTION 3
title('QUELQUES SOLUTIONS INTEGRALES') ; xlabel('dx') ; ylabel('dy') ;
```

Nous fermons cette parenthèse sur les courbes intégrales, focalisons nous désormais sur la résolution numérique des équations différentielles.

A. *Méthode d'Euler*

Afin d'atteindre la solution $y(x)$, sur l'intervalle $x \in [a, b]$, on choisit $n+1$ points dissemblables $x_0, x_1, x_2, ..., x_n$, avec $x_0 = a$ et $x_n = b$ et le pas de discrétisation est défini par $h = (b-a)/n$. La résolution numérique consiste à discrétiser l'axe des abscisses suivant $x_n = x_0 + h\, n$ ($n \in \mathbb{N}$). Ensuite on cherchera u_n comme approximation de y au point x_n, soit $y(x_n)$. Ainsi l'ensemble des approximations successives $\{u_0, u_1, u_2, ..., u_n\}$, où tout simplement $\{u_n\}_{n\in\mathbb{N}}$, constitue la solution numérique. Ces méthodes sont itératives donc la suite $\{u_n\}_{n\in\mathbb{N}}$ doit être initialisée afin de calculer ses successeurs. Soit une équation différentielle :

$$y^{'} = f(x, y(x)) \tag{2}$$

Trouver la solution de cette équation revient à calculer l'intégrale de $f(x, y(x))$ entre les bornes x_n et x_{n+1}, soit :

$$\int_{x_n}^{x_{n+1}} f(x, y(x)) = y(x_{n+1}) - y(x_n) \tag{3}$$

Cette intégrale s'écrit en fonction des approximations u_{n+1} et u_n :

$$\int_{x_n}^{x_{n+1}} f(x, y(x)) = u_{n+1} - u_n \tag{4}$$

Par conséquent, en fonction de la méthode d'intégration utilisée afin de résoudre l'intégrale (terme de gauche), on obtient un schéma numérique donné. En utilisant par exemple la méthode des rectangles à gauche, on obtient le schéma numérique d'Euler progressif :

$$\begin{cases} u_0 = y(x_0) = y_0 \\ u_{n+1} - u_n = h\, f(x_n, u_n) \quad \text{avec} \quad k \in \mathbb{N} \end{cases} \tag{5}$$

En utilisant la méthode des rectangles à droite, on obtient le schéma numérique d'Euler rétrograde :

$$\begin{cases} u_0 = y(x_0) = y_0 \\ u_{n+1} - u_n = h\, f(x_{n+1}, u_{n+1}) \quad \text{avec} \quad k \in \mathbb{N} \end{cases} \tag{6}$$

En utilisant la méthode du point milieu, on obtient le schéma numérique d'Euler modifié :

$$\begin{cases} u_0 = y(x_0) = y_0 \\ u_{n+1/2} \simeq u_n + \frac{h}{2}\, f(x_n, u_n) \\ u_{n+1} - u_n = h\, f(x_n + \frac{h}{2}, u_{n+\frac{1}{2}}) \quad \text{avec} \quad k \in \mathbb{N} \end{cases} \tag{7}$$

Exercice ❶ ☞ ®

Soit à résoudre numériquement l'équation différentielle :

$$\begin{cases} 2\, y^{'} = x - y \\ \\ y(0) = 1 \end{cases} \tag{8}$$

Nous utiliserons à cet effet le schéma numérique d'Euler rétrograde :

$$u_{n+1} = u_n + h \left(\frac{x_{n+1} - u_{n+1}}{2} \right) \Rightarrow 2\, u_{n+1} = 2\, u_n + h\, x_{n+1} - h\, u_{n+1}$$

$$\Rightarrow u_{n+1} = \left(\frac{2\, u_n + h\, x_{n+1}}{2 + h} \right) \tag{9}$$

Avec, $x_n = x_0 + n\, h$ et $x_{n+1} = x_0 + (n+1)\, h$. On discrétise l'axe des abscisses suivant 10 nœuds sur un intervalle $[0,\, 1]$. Le pas de discrétisation est $h = (1-0)/10 \Rightarrow h = 0.1$ et la condition initiale est $y(x_0 = 0) = 1 = u_0$. Nous avons appliqué le schéma numérique ci-dessus pour dix approximations successives :

$$
\begin{aligned}
u_1 &= \left(\frac{2\,u_0 + h\,x_1}{2+h}\right)\\
u_2 &= \left(\frac{2\,u_1 + h\,x_2}{2+h}\right)\\
u_3 &= \left(\frac{2\,u_2 + h\,x_3}{2+h}\right)\\
u_4 &= \left(\frac{2\,u_3 + h\,x_4}{2+h}\right)\\
\vdots & \qquad \vdots\\
u_{10} &= \left(\frac{2\,u_9 + h\,x_{10}}{2+h}\right)
\end{aligned}
$$

TABLE I: Résultats numériques

n	x_n	u_n
0.0000	0.0000	1.0000
1.0000	0.1000	0.9524
2.0000	0.2000	0.9118
3.0000	0.3000	0.8779
4.0000	0.4000	0.8504
5.0000	0.5000	0.8289
6.0000	0.6000	0.8133
7.0000	0.7000	0.8031
8.0000	0.8000	0.7982
9.0000	0.9000	0.7983
10.0000	1.0000	0.8031

Ci-dessous le script Matlab® :

```
clc ; clear all ; close all ;
% Samir KENOUCHE - Le 06/09/2019
% EULER IMPLICITE : y' = (x - y)/2
% SCHEMA NUMERIQUE : Un+1 = Un + F(Xn+1, Un+1)

dy = @(x,y) (x - y)./2 ; un(1) = 1 ; % CONDITION INITIALE
a = 0 ; b = 1 ; n = 10 ; h = (b-a)/n ; xn = a :h: b ;

for it = 1:n - 1
    un(it+1) = (1/(2+h))*(2*un(it) + h*xn(it)) ;
end

sol_exacte = 3.*exp(-xn/2) + xn - 2 ; figure('color',[1 1 1]) ;
plot(xn(1:end-1),un,'o-b') ; hold on ; plot(xn, sol_exacte,'-ro') ;
legend('SOLUTION APPROCHEE','SOLUTION EXACTE') ; xlabel('x') ; ylabel('y') ;
```

Très souvent pour des équations différentielles ayant des formules mathématiques compliquées, il sera difficile de retrouver analytiquement le schéma numérique. Dans ce cas il serait judicieux d'appliquer explicitement la formule d'Euler. Ci-dessous, le script Matlab® correspondant.

```
clc ; clear all ; close all ;
% Samir KENOUCHE - Le 06/09/2019
% FORMULE D'EULER - Eq : y' = (x - y)/2
% SCHEMA NUMERIQUE : Un+1 = Un + F(Xn, Un)

dy = @(x,y) (x - y)./2 ; un(1) = 1 ; % CONDITION INITIALE
a = 0 ; b = 1 ; n = 10 ; h = (b-a)/n ; xn = a:h:b;

for it = 1:n - 1
    un(it+1) = un(it) + h*dy(xn(it), un(it)) ;
end

sol_exacte = 3.*exp(-xn/2) + xn - 2 ; figure('color',[1 1 1]) ;
plot(xn(1:end-1),un,'o-b') ; hold on ; plot(xn, sol_exacte,'-ro') ;
legend('SOLUTION APPROCHEE','SOLUTION EXACTE') ; xlabel('x') ; ylabel('y') ;
```

B. Méthode de Heun

La Méthode de Heun est une version améliorée de celle d'Euler. L'erreur, sur le résultat, générée par cette méthode est proportionnelle à h^3, meilleur que celle de la méthode d'Euler. Néanmoins, la méthode de Heun réclame une double évaluation de la fonction f :

$$\begin{cases} u_0 = y(x_0) = y_0 \\ u_{n+1} = u_n + \dfrac{h}{2}\left\{f(x_n, u_n) + f(x_n, u_n + h\, f(x_n, u_n))\right\} \quad \text{avec} \quad k \in \mathbb{N} \end{cases} \tag{10}$$

Le schéma numérique de cette méthode résulte de l'application de la formule de quadrature du trapèze. Notons également que cette méthode fait partie des méthodes de Runge-Kutta explicites d'ordre deux. Afin d'illustrer le fonctionnement de cette méthode, reprenant l'équation différentielle de l'exemple numérique précédent et cherchons la formule analytique correspondante :

$$\begin{aligned} u_{n+1} &= u_n + \frac{h}{2}\left(\frac{x_n - u_n}{2}\right) + \frac{h}{2}\left(\frac{x_n - \left(u_n + h\left(\dfrac{x_n - u_n}{2}\right)\right)}{2}\right) \\ &= u_n + \frac{h}{2}\left(\frac{x_n - u_n}{2}\right) + \frac{h}{2}\left(\frac{x_n - \left(\dfrac{2\,u_n + h\,x_n - h\,u_n}{2}\right)}{2}\right) \\ &= u_n + \frac{h}{2}\left(\frac{x_n - u_n}{2}\right) + \frac{h}{2}\left(\frac{2\,x_n - 2\,u_n - h\,x_n + h\,u_n}{4}\right) \end{aligned}$$

Finalement on obtient :

$$u_{n+1} = \left(\frac{4\,h - h^2}{8}\right) x_n + \left(\frac{8 - 4\,h + h^2}{8}\right) u_n \tag{11}$$

Script Matlab® :

```
clc ; clear all ; close all ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SAMIR KENOUCHE - Le 06/09/2019
% Heun IMPLICITE : y' = (x - y)/2
% SCHEMA NUMERIQUE : Un+1 = Un + h/2 *(F(Xn, Un), F(Xn+1, Un+1))

lb = 0 ; ub = 1 ; nb = 10 ; h = (ub - lb)/nb ; n = 0:nb ; n = n(1:end-1) ;
xn = lb + n.*h ; U(1) = 1 ; it = 0 ;

while it < nb - 1
    it = it + 1 ;
    Un(it) = ((4*h - h.^2)/8)* xn(it) + ((8 - 4*h + h.^2)/8)*U(1) ;
    U(1) = Un(it) ; % MISE A JOUR
end

Un = [ub Un] ; sol_exacte = 3.*exp(-xn/2) + xn - 2 ; figure('color', [1 1 1])
  ;
plot(xn, Un,'-s') ; hold on ; plot(xn, sol_exacte,'-r') ;
legend('SOLUTION APPROCHEE','SOLUTION EXACTE') ; xlabel('xn') ; ylabel('Un') ;
```

Comme précédemment, cette façon de procéder n'est faisable qui si la formule mathématique, découlant du schéma numérique, est relativement simple. Il est ainsi plus pertinent d'appliquer explicitement la formule de Heun. Ci-dessous, le script Matlab® correspondant.

```
clc ; clear all ; close all ;
% Samir KENOUCHE - Le 06/09/2019
% Heun - Eq : y' = (x - y)/2
% SCHEMA NUMERIQUE : Un+1 = Un + h/2[F(Xn+1, Un+1) + F(Xn, Un + h F(Xn,Un))]

dy = @(x,y) (x - y)./2 ; un(1) = 1 ; % CONDITION INITIALE
a = 0 ; b = 1 ; n = 10 ; h = (b-a)/n ; xn = a:h:b;

for i = 1:n - 1
     un(i+1) = un(i) + h/2*(dy(xn(i),un(i)) + dy(xn(i + 1),...
     un(i) + h*dy(xn(i),un(i)))) ;
end

sol_exacte = 3.*exp(-xn/2) + xn - 2 ; figure('color',[1 1 1]) ;
plot(xn(1:end-1),un,'o-b')  ; hold on ; plot(xn, sol_exacte,'-ro') ;
legend('SOLUTION APPROCHEE','SOLUTION EXACTE') ; xlabel('xn') ; ylabel('Un') ;
```

Ci-dessous, les résultats numériques pour dix approximations :

TABLE II: Méthode de Heun

n	x_n	u_n
0.0000	0.0000	1.0000
1.0000	0.1000	0.9537
2.0000	0.2000	0.9146
3.0000	0.3000	0.8823
4.0000	0.4000	0.8564
5.0000	0.5000	0.8367
6.0000	0.6000	0.8227
7.0000	0.7000	0.8144
8.0000	0.8000	0.8113
9.0000	0.9000	0.8133

C. Méthode de Crank - Nicolson

Cette méthode permet d'obtenir une plus grande précision en ce sens qu'elle génère des solutions numériques plus proches des solutions analytiques que les deux méthodes précédentes. Le schéma numérique est donné par :

$$\begin{cases} y_0 = y(x_0) = u_0 \\ u_{n+1} = u_n + \dfrac{h}{2}\left\{f(x_n, u_n) + f(x_{n+1}, u_{n+1})\right\} \quad \text{avec} \quad n \in \mathbb{N} \end{cases} \tag{12}$$

Cette méthode et celle d'Euler rétrograde sont inconditionnellement stables, moyennant certaines conditions de régularité sur les équations à résoudre.

D. Méthode de Runge–Kutta, d'ordre 4

La méthode de Runge-Kutta (classique) d'ordre 4, est une méthode explicite très populaire. Elle calcule la valeur de la fonction en quatre points intermédiaires selon :

$$\begin{cases} y_0 = y(x_0) = u_0 \\ u_{n+1} = u_n + \dfrac{h}{6}\left(f(x_n, u_{1n}) + 2\,f(x_n + \dfrac{h}{2}, u_{2n}) + 2\,f(x_n + \dfrac{h}{2}, u_{3n}) + f(x_{n+1}, u_{4n})\right) \quad \text{avec} \quad n \in \mathbb{N} \end{cases} \tag{13}$$

$$\begin{cases} u_{1n} = u_n \\ u_{2n} = u_n + \frac{h}{2}\,f(x_n, u_{1n}) \\ u_{3n} = u_n + \frac{h}{2}\,f(x_n + \frac{h}{2}, u_{2n}) \\ u_{4n} = u_n + h\,f(u_n + \frac{h}{2}, u_{3n}) \end{cases} \tag{14}$$

Notons que le nombre de termes retenus dans la série de `Taylor` définit l'ordre de la méthode de Runge-Kutta. Il en découle que cette méthode est d'ordre 4, s'arrête au terme $O(h^4)$ de la série de Taylor.

Exercice ❷ ☞ Ⓡ

1) Résoudre numériquement, par le biais des méthodes d'Euler, de Heun et de Runge-Kutta d'ordre 4, l'équation différentielle du premier ordre suivante :

$$\begin{cases} y^{'} = \dfrac{-y\,x^2 - y^2 + 2\,x}{1 - x^3} \\ y(0) = 1 \end{cases} \tag{15}$$

2) Afficher sur la même figure, la solution des trois méthodes.

Script Matlab® :

```
clear all ; close all ; clc ;
% LE 06/09/2019 Samir KENOUCHE : ALGORITHME PERMETTANT
% L'IMPLEMENTATION, SOUS MATLAB, DE METHODES NUMERIQUES (Euler, Heun,
% Runge-Kutta) POUR LA RESOLUTION D'EQUATIONS DIFFERENTIELLES

dy = @(xn, u) (-u.*xn.^2 - u.^2 + 2.*xn)./(1 - xn.^3) ;
a = 0 ; b = 1 ; n = 50 ; h = (b-a)/n ; xn = a :h: b ; u(1) = 1 ;

%%%%%%%%%%%%%%%%%%%%% METHODE DE Runge-Kutta d'ordre 4 %%%%%%%%%%%%%%%%%%%%%%%

for i = 1:n-1
   u1(i) = u(i) ;
   u2(i) = u(i) + h/2*dy(xn(i),u1(i)) ;

   u3(i) = u(i) + h/2*dy(xn(i) + h/2, u2(i)) ;
   u4(i) = u(i) + h*dy(xn(i) + h/2, u3(i)) ;

   u(i+1) =u(i) + h/6*(dy(xn(i),u1(i)) + 2*dy(xn(i) + h/2,...
   u2(i)) + 2*dy(xn(i) + h/2,u3(i)) + dy(xn(i+1),u4(i))) ;
end

figure('color',[1 1 1]) ; plot(xn(1:end-1),u,'o-g') ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% METHODE DE Heun %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dy = @(xn, u) (-u.*xn.^2 - u.^2 + 2.*xn)./(1 - xn.^3) ;
a = 0 ; b = 1 ; n = 50 ; h = (b-a)/n ; xn = a:h:b ;
epsilon = 0.0001 ; u(1) = 1 + epsilon ;

for i = 1:n - 1
  u(i+1) = u(i) + h/2*(dy(xn(i),u(i)) + dy(xn(i + 1),...
           u(i) + h*dy(xn(i),u(i))));
end

xn = xn(1:end-1) ; hold on ; plot(xn,u,'o-r') ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% METHODE D'Euler %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dy = @(xn, u) (-u.*xn.^2 - u.^2 + 2.*xn)./(1 - xn.^3) ;

a = 0 ; b = 1 ; n = 50 ; h = (b-a)/n ; xn = a:h:b ; u(1) = 1 ;

for i = 1:n - 1
  u(i+1) = u(i) + h/2*(dy(xn(i),u(i))) ;
end

hold on ; plot(xn(1:end-1),u,'o-b') ; legend('Runge-Kutta','Heun','Euler')
```

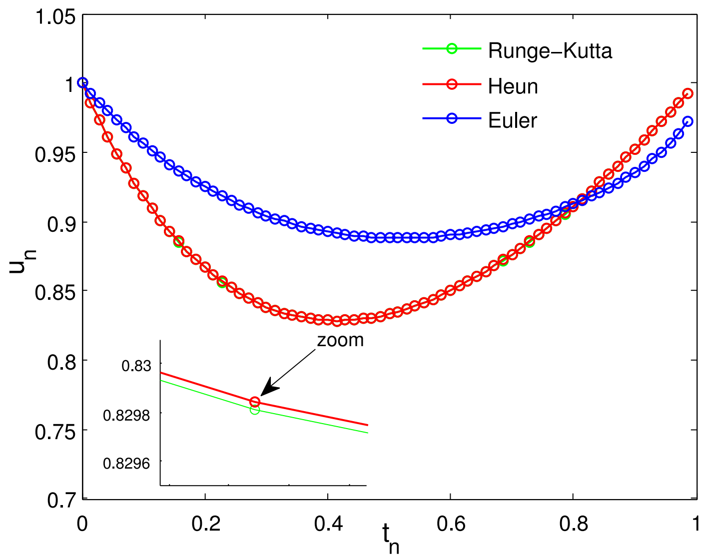

FIGURE 2: Solutions numériques par les méthodes d'Euler, de Heun et de Runge-Kutta d'odre 4

II. ÉQUATIONS DIFFÉRENTIELLES D'ORDRE n

N'importe quelle équation différentielle d'ordre n peut être ramenée à un systèmes de n équations du premier ordre. Soit l'équation différentielle du second ordre suivante :

$$\begin{cases} y^{''} = \dfrac{t\,y^{'}}{2} - y + 3 \\ y(0) = 1 \quad \text{et} \quad y^{'}(0) = 0 \end{cases} \tag{16}$$

Posons $u_1(t) = y(t)$ et $u_2(t) = y^{'}(t)$ il vient $u_2^{'}(t) = \dfrac{t\,u_2(t)}{2} - u_1(t) + 3$.

Exercice ❸ ☞ ®

1) Résoudre numériquement, au moyen de la méthode d'Euler, l'équation différentielle du second ordre (16).
2) Afficher sur la même figure, la solution numérique et la solution exacte, donnée par : $t^2 + 1$.
3) Afficher le graphe de la dérivée de la fonction $y(t)$.
4) Appliquer le même script Matlab® pour résoudre l'équation différentielle du troisième ordre suivante :

$$\begin{cases} y^{'''}(t) = 0.001\,\left(y^{''}(t) + (1 - y(t)^2)\right) \times y^{'}(t) + sin(t) \\ y(0) = 1, \quad y^{'}(0) = 5, \quad \text{et} \quad y^{''}(0) = 0 \end{cases} \tag{17}$$

5) Afficher sur la même figure, la solution $y(t)$ ainsi que ses première et deuxième dérivées.

La résolution numérique de ce système d'équations, par la méthode d'Euler, est donnée par le script Matlab® ci-dessous.

```
clear all ; clc ; close all ;

% 06/09/2019 Samir KENOUCHE : ALGORITHME PERMETTANT LA
% RESOLUTION D'EQUATIONS DIFFERENTIELLES DU SECOND ORDRE
% EULER - Eq : y'' = (t*y')/2 - y + 3
% SCHEMA NUMERIQUE : Un+1 = Un + h F(tn, Un)

a = 0 ; b = 1 ; n = 64 ; h = (b-a)/n ; tn = a :h: b ;
```

```
fun = @(tn,un)[un(2), (tn*un(2))/2 - un(1) + 3] ; un = [1 0] ;

for i = 1:n
    un(i+1,:) = un(i,:) + h*(fun(tn(i),un(i,:))) ;
end

sol_exacte = tn.^2 + 1 ; figure('color',[1 1 1]) ;
plot(tn, un(:,1),'o','MarkerSize',8) ; hold on ;
plot(tn,sol_exacte,'r','LineWidth',1.5) ; axis([-0.05 1.1 0.8 2.1])
ih1 = legend('SOLUTION NUMERIQUE','SOLUTION EXACTE');
set(ih1,'Interpreter','none','Location','NorthWest','Box','on',...
    'Color','none') ; xlabel('t') ; ylabel('y(tn)') ;
```

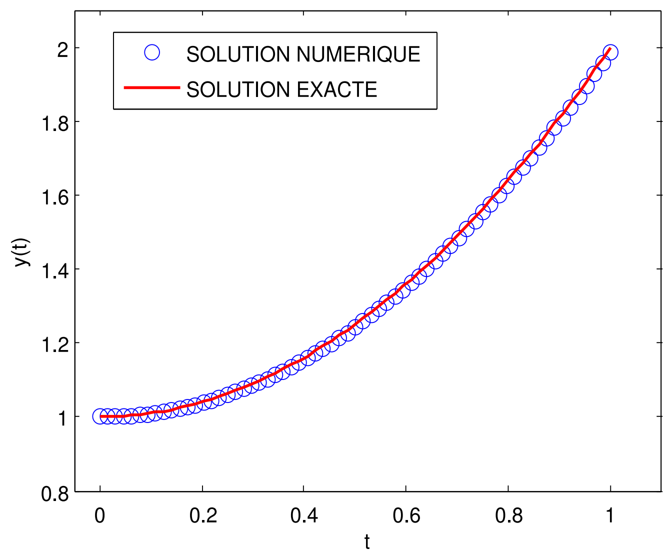

FIGURE 3: Solution exacte et solution numérique obtenues par la méthode d'Euler

Nous appliquons le même script Matlab® pour résoudre l'équation différentielle du troisième ordre suivante :

$$\begin{cases} y^{'''}(t) = 0.001\,\left(y^{''}(t) + (1 - y(t)^2)\right) \times y^{'}(t) + sin(t) \\ y(0) = 1, \quad y^{'}(0) = 5, \quad \text{et} \quad y^{''}(0) = 0 \end{cases} \tag{18}$$

De la même façon que précédemment, posons $u_1(t) = y(t)$, $u_2(t) = y^{'}(t)$ et $u_3(t) = y^{''}(t) \Rightarrow y^{'''}(t) = u_3^{'}(t)$. Il en ressort que :

$$u_3^{'}(t) = 1e-03\,\left(u_3(t) + (1 - u_1(t)^2)\right) \times u_2(t) + \sin(t)$$

```
clear all ; clc ; close all ;
% ALGORITHME PERMETTANT LA RESOLUTION D'EQUATIONS DIFFERENTIELLES
% DU 3EMME ORDRE
% Samir KENOUCHE - Le 06/09/2019
% EULER - Eq : y''' = alpha (y'' + 1 - y^2)*y' + sin(t)
% AVEC y(0) = 1, y'(0) = 5 et y''(0) = 0
```

```
% SCHEMA NUMERIQUE : Un+1 = Un + h F(tn, Un)

a = 0 ; b = 10 ; n = 64 ; h = (b-a)/n ; tn = a :h: b ; alpha = 1e-03 ;
fun = @(tn,u)[u(2), u(3), alpha*(u(3) + (1 - u(1).^2))*u(2) + sin(tn)] ;
u = [1 5 0] ;

for i=1:n

  u(i+1,:) = u(i,:) + h*fun(tn(i),u(i,:)) ;

end

figure('color',[1 1 1]) ;

plot(tn, u(:,1),'-o','MarkerSize',6,'LineWidth',1) ; hold on ;
plot(tn, u(:,2),'r-o','MarkerSize',6,'LineWidth',1) ;
plot(tn, u(:,3),'k-o','MarkerSize',6,'LineWidth',1) ;
axis([-1/2 10.5 -12 32]) ; ih1 = legend('y(t)','y_prime','dy_2primes') ;

set(ih1,'Interpreter','none','Location','NorthWest','Box','off',...
    'Color','none') ; xlabel('TEMPS') ; ylabel('SOLUTION NUMERIQUE') ;
```

Ci-dessous le résultats renvoyés par le script :

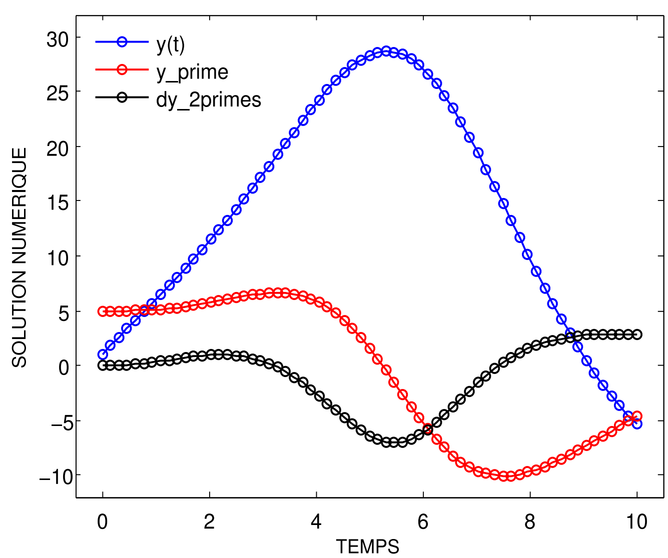

FIGURE 4: Équation différentielle du troisième ordre résolue par la méthode d'Euler

III. Travaux pratiques avec des fonctions Matlab prédéfinies

Matlab® comprend un certain nombre de *solveurs* destinés à la résolution d'équations différentielles. Les plus utilisés sont `ode23` et `ode45`, qui sont basés sur la méthode de *Runge-Kutta* explicite à un pas. Le *solveur* `ode113`, utilise la méthode de *Adams-Bashforth-Moulton* multi-pas. Les autres *solveurs* sont `ode15s, ode23s, ode23t, ode23tb`. Ils ont tous la même syntaxe :

```
[t, y] = solveur(eqs, [ti; tf], yinit, opts)
```

Cette syntaxe renvoie la solution `y` au temps `t`. L'argument `eqs` est le système d'équations différentielles. Ce système peut être défini de plusieurs manières. Soit à travers un fichier *M-file*, dans ce cas on doit rajouter l'identifiant `@eqs`. Soit à travers une fonction `inline` ou bien au moyen de la *fonction anonyme*. Ce système d'équations différentielles est résolu sur l'intervalle `[ti ; tf]`, avec les conditions initiales `yinit = [y(ti) ; y(tf)]`. L'argument d'entrée `opts`, de type *srtucture*, compte les options d'optimisation indiquées dans `odeset`, sa syntaxe est donnée par :

```
opts = odeset('Property1', value1, 'Property2', value2, ...)
```

Chaque propriété est suivie de sa valeur. À titre illustratif, la propriété `('Stats','on', ...)` affiche à la fin de l'exécution, des statistiques relatives au calcul effectué. Pour plus d'informations sur les paramètres d'optimisation, taper *help odeset* en ligne de commande.

Exercice ❶ ☞ ®

1) Résoudre symboliquement et numériquement au moyen du *solveur* `ode23`, l'équation différentielle du second ordre suivante :

$$\begin{cases} y^{''}(t) + 3\,y = 4\,\sin(t) + \exp(-t) \\ y(0) = 1, \quad y^{'}(0) = 1 \end{cases} \tag{19}$$

2) Afficher sur la même figure, la solution numérique et la solution analytique.

Script Matlab®

```
clear all ; clc ; close all ;

% LE 06/09/2019 - SAMIR KENOUCHE

%%%%%%%%%%%%%%%%%%%%%%%%%%%% SOLUTION SYMBOLIQUE %%%%%%%%%%%%%%%%%%%%%%%%%%%
syms t

ySym = dsolve('D2y + 3*y = 4*sin(t) + exp(-t)','y(0) = 1','Dy(0) = 1','t')

%%%%%%%%%%%%%%%%%%%%%%%%%%%% SOLUTION NUMERIQUE %%%%%%%%%%%%%%%%%%%%%%%%%%%%
a = 0 ; b = 10 ; n = 200 ; h = (b-a)/n ; tn = a :h: b ; y = [1 ; 1] ;

eqs = @(tn,y) [y(2); - 3*y(1) + 4*sin(tn) + exp(-tn)] ;
opts = odeset('Stats','on','RelTol', 1e-3) ;
[tn, ysol] = ode23(eqs, [tn(1) ; tn(end)], y, opts) ; % SOLVEUR ode23

%%%%%%%%%%%%%%%%%%%%%%%%%%%% AFFICHAGE GRAPHIQUE %%%%%%%%%%%%%%%%%%%%%%%%%%%
figure('color', [1 1 1]) ;
plot(tn, ysol(:,1),'o-','LineWidth',1) ; hold  on ;
plot(tn, double(subs(ySym, t, tn)),'o-r','LineWidth',1)  ;
```

```
xlabel('t','FontSize',12) ; ylabel('y(t)','FontSize',12) ;
ih1 = legend('SOLUTION NUMERIQUE','SOLUTION ANALYTIQUE') ;
set(ih1,'Interpreter','none','Location','NorthWest','Box','off', ...
    'Color','none') ;
```

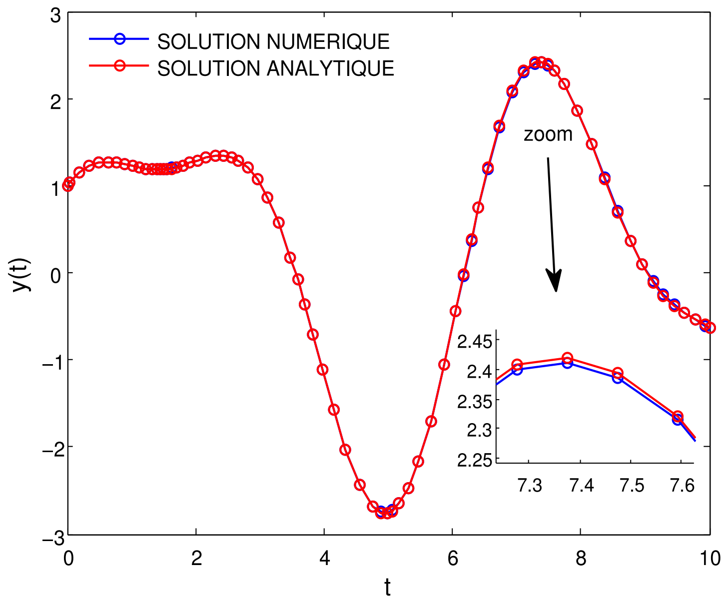

FIGURE 5: Solutions analytique et numérique générées par le solveur `ode23`

Une autre façon de faire est de créer des fonctions imbriquées dans un fichier *M-file*, selon la syntaxe suivante :

```
function [] = eqs_sol(t,y)

tsol = [t(1) ; t(end)] ;

opts = odeset('Stats','on','RelTol', 1e-3) ;
[t , ysol] = ode23(@eqs, tsol, y, opts)  ;
plot(t,ysol(:,1))
function dydt = eqs(t, y)
dydt = [y(2) ; - 3*y(1) + 4*sin(t) + exp(-t)] ;

end
end
```

Qu'il faudra ensuite appeler, en tapant dans la fenêtre des commandes : `eqs_sol(t, [1 ; 1])`. Cette fonction accepte deux arguments en entrée, le vecteur `t` et les conditions initiales `[1 ; 1]`. Le fichier doit être sauvegardé sous le nom `eqs_sol.m`. Nous allons désormais résoudre une équation différentielle du second ordre avec paramètre variable, noté α :

Exercice ❷ ☞ Ⓡ

1) Résoudre numériquement au moyen du *solveur* `ode45`, l'équation différentielle du second ordre suivante :

$$\begin{cases} y^{''}(t) - \alpha\,(1-y^2)\,y^{'} = -y \\ y(0) = 1, \quad y^{'}(0) = 0 \end{cases} \tag{20}$$

2) Afficher sur la même figure, la solution numérique pour différentes valeurs de α. Il en est de même pour les fonctions dérivées. Avec, le paramètre α qui prend ses valeurs de 1.5 à 4 par pas de 0.5.

Voici le script Matlab :

```
clc ; clear all ; close all ;
% LE 06/09/2019 - SAMIR KENOUCHE

a = 1 ; b = 25 ; n = 100 ; h = (b-a)/n ; t = a :h: b ;

y = [1 ; 0] ; alpha = [1.5 ; 2.0 ; 2.50 ; 3.0 ; 3.5 ; 4.0] ;
col = {'o-k','o-m','-co','o-g','o-','o-r'} ;

for ik = 1 : numel(alpha)

    eqs = @(t,y) [y(2); alpha(ik)*(1 - y(1).^2)*y(2) - y(1)] ;
    opts = odeset('Stats','on','RelTol', 1e-3) ;
    [t, ySol] = ode45(eqs, [t(1) ; t(end)], y, opts) ;

    ysol = squeeze(ySol(:,1)) ; ysolDer = squeeze(ySol(:,2)) ;
    figure(1) ; plot(t, ysol, col{ik}) ; hold on ; % SOLUTIONS
    xlabel('t','FontSize',12) ; ylabel('y(t)','FontSize',12)
    figure(2) ; plot(t, ysolDer, col{ik}) ; hold on ; % DERIVEES
    xlabel('t','FontSize',12) ; ylabel('y^{''}(t)','FontSize',12)

end
```


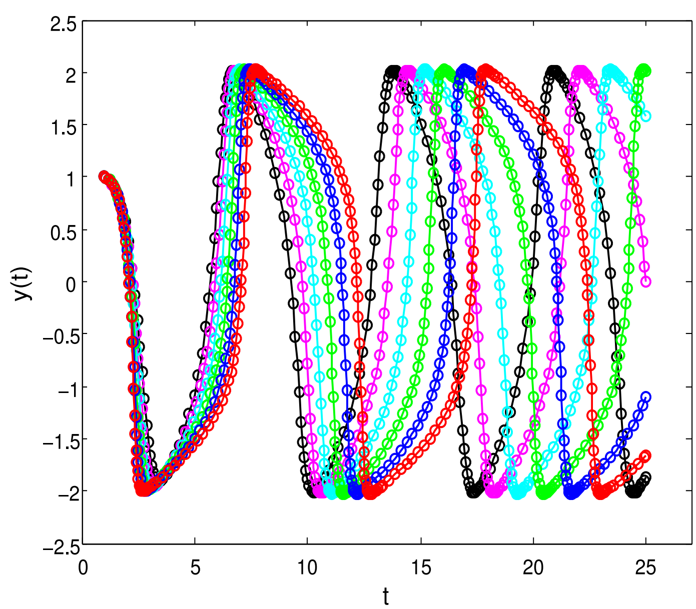

FIGURE 6: Solution numérique $y(t)$ pour différentes valeurs de α

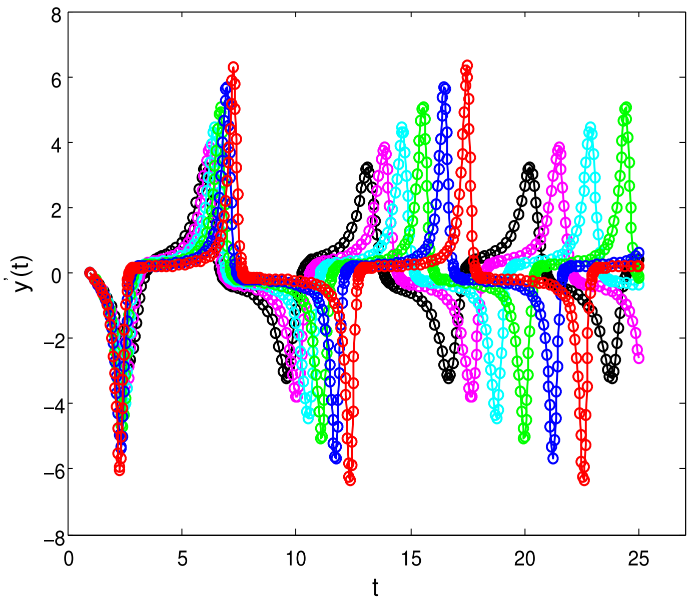

FIGURE 7: Dérivée première de la solution numérique $y(t)$ pour différentes valeurs de α

Exercice ❸ ☞ Ⓢ

1) Résoudre numériquement les équations différentielles suivantes :

$$\begin{cases} y^{'} + y\,\cos(x) = \cos(x), \quad y(0) = 1 \\ y^{'} = x + \cos(y), \quad y(0) = 0 \\ y^{''} + 2\,y^{'} = -2y, \quad y(0) = 2, y^{'}(0) = -3 \\ y^{''} - y^{'} - 2y = 2\,x\exp(-x) + x^2, \quad y(0) = 0, y^{'}(0) = 1 \\ y^{''} + y = 3\,x^2 - 4\sin(x), \quad y(0) = 0, y^{'}(0) = 1 \\ y^{'''} + y^{'} = x, \quad y(0) = 0, y^{'}(0) = 1, y^{''}(0) = 1 \\ y^{''''} - y = 8\,\exp(x), \quad y(0) = 0, y^{'}(0) = 2, y^{''}(0) = 4, y^{'''}(0) = 6 \end{cases} \tag{21}$$

2) Afficher sur la même figure, la solution numérique ainsi que les fonctions dérivées.