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 permettant la résolution des équations différentielles ordinaires (EDO). Les méthodes abordées sont dites *à un pas*, car le calcul de $y(t_{n+1})$ ne réclame que la valeur de $y(t_n)$ à l'instant précédent. Une méthode *à deux pas* utilisera à la fois $y(t_n)$ et $y(t_{n-1})$. 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 $y(t)$. La forme générale d'une telle équation s'écrit :

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

Avec, $y^{(n)}$ est la nième dérivée de la fonction y et $\varphi(t)$ désigne le second membre de l'équation différentielle. Dans le cas où $\varphi(t) = 0$, 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 $y(t)$ 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 $y(t)$ par des méthodes numériques.

Dernière mise à jour le 09.09.2019.

A. *Problème de Cauchy*

Le problème de Cauchy consiste à trouver une fonction continûment dérivable $y : t \in \mathbb{R}^+ \longrightarrow y(t) \in \mathbb{R}$ vérifiant :

$$\begin{cases} y^{'}(t) = f(t, y(t)), \quad t > 0 \\ y(0) = y_0 \end{cases} \tag{2}$$

La première équation est une équation différentielle et la deuxième relation exprime une condition de Cauchy ou condition initiale. Le problème (2) est équivalent à l'équation intégrale :

$$y(t) = y_0 + \int_{t_0}^{t} f(u, y(u))\, du \tag{3}$$

a) ***Définition****: soit $f : \mathbb{I} \times \mathbb{R} \longmapsto \mathbb{R}$ une fonction donnée, s'il existe une constante $L > 0$ telle que :*

$$|f(t, u) - f(t, v)| \leq L\,|u - v| \tag{4}$$

$\forall\, u, v \in \mathbb{R}$ et $\forall\, t \in \mathbb{I}$ alors f est dite lipschitzienne de rapport L sur $\mathbb{I} \times \mathbb{R}$ ou simplement L-lipschitzienne.

b) ***Théorème****: si f est continue sur $\mathbb{I} \times \mathbb{R}$ et L-lipschitzienne par rapport à sa deuxième variable $y(t)$ alors le problème de Cauchy admet une solution unique sur $\mathbb{I}$, $\forall\, u(0) \in \mathbb{R}$.*

c) ***Démonstration****:* pour démontrer ce théorème nous considérons l'application $\varphi : (\mathbb{R} \to \mathbb{R}^n) \longmapsto (\mathbb{R} \to \mathbb{R}^n)$ qui est définie par :

$$\varphi(y)(t) = y_0 + \int_{t_0}^{t} f(u, y(u))\, du \tag{5}$$

De sorte que,

$$\varphi(y)(t_0) = y_0 + \underbrace{\int_{t_0}^{t_0} f(u, y(u))\, du}_{=0} = y_0 \tag{6}$$

Ainsi $\varphi(y)(t_0)$ satisfait toujours la condition initiale. En outre, si elle admet un point fixe, ie :

$$\varphi(y) = y \quad \Rightarrow \quad y(t) = y_0 + \int_{t_0}^{t} f(u, y(u))\, du \tag{7}$$

$$\Rightarrow \quad \frac{dy(t)}{dt} = 0 + \frac{d}{dt}\left[\int_{t_0}^{t} f(u, y(u))\, du\right] \quad \Rightarrow \quad y^{'}(t) = f(t, y(t)) \tag{8}$$

Donc un point fixe de l'application φ sera forcément une solution de l'équation différentielle avec la même condition initiale. La démonstration du théorème revient désormais à prouver que l'application φ est *contractante*. Cela signifie qu'elle va contracter l'espace des fonctions de sorte à rapprocher toute fonction d'une fonction qui solution du problème. Ci-dessous la démonstration :

$$\begin{aligned} \| \varphi^{(1)}(y_1)(t) - \varphi^{(1)}(y_2)(t) \| &= \| y_0 + \int_{t_0}^{t} f(u, y_1(u))\, du - y_0 - \int_{t_0}^{t} f(u, y_2(u))\, du \| \quad \text{inég. triang.} \\ &\leq \int_{t_0}^{t} \| f(u, y_1(u)) - f(u, y_2(u)) \|\, du \quad \text{avec } f \text{ est L-lipschitzienne} \\ &\leq L \int_{t_0}^{t} \underbrace{\| y_1(u) - y_2(u) \|}_{\leq \|y_1 - y_2\|_\infty} du \\ &\leq L\,(t - t_0)\ \| y_1 - y_2 \|_\infty \end{aligned}$$

En itérant une deuxième fois :

$$\begin{aligned}\| \varphi^{(2)}(y_1)(t) - \varphi^{(2)}(y_2)(t) \| &= \| y_0 + \int_{t_0}^{t} f(u, \varphi(y_1)(u))\, du - y_0 - \int_{t_0}^{t} f(u, \varphi(y_2)(u))\, du \| \\ &\leq \int_{t_0}^{t} \| f(u, \varphi(y_1)(u)) - f(u, \varphi(y_2)(u)) \| \, du \\ &\leq \int_{t_0}^{t} \| \varphi(y_1)(u) - \varphi(y_2)(u) \| \, du \\ &\leq L^2 \int_{t_0}^{t} \| y_1 - y_2 \|_\infty \, du \\ &\leq L^2 \frac{(t-t_0)^2}{2} \| y_1 - y_2 \|_\infty\end{aligned}$$

Par récurrence à la nième itération, il en résulte :

$$\| \varphi^{(n)}(y_1)(t) - \varphi^{(n)}(y_2)(t) \| \leq L^n \frac{(t-t_0)^n}{n} \| y_1 - y_2 \|_\infty \tag{9}$$

Le terme $L^n \dfrac{(t-t_0)^n}{n}$ tend vers zéro quand n tend vers l'infinie $\Rightarrow$ l'application φ est contractante. Ce théorème garantit l'unicité des solutions des équations différentielles pour une condition initiale donnée. Autrement dit, à deux conditions initiales différentes correspondent deux solutions différentes. Ceci est d'une importance majeure pour pouvoir prédire l'état d'un système à un instant ultérieur. Comme par exemple la prédiction de la trajectoire d'une particule à partir de l'instant courant.

Nous rappelons que pour des fonctions continues $f : \mathcal{D} \subset \mathbb{R} \longmapsto \mathbb{R}$ $\forall\ p \geq 1$, les normes p sont définies par :

$$\| f \|_p = \left[\int_{t \in \mathcal{D}} |f(t)|^p \, dt \right]^{1/p} \tag{10}$$

En l'occurrence la norme Euclidienne $p = 2$ est définie par :

$$\| f \|_2 = \left[\int_{t \in \mathcal{D}} |f(t)|^2 \, dt \right]^{1/2} \tag{11}$$

La norme infinie s'écrit :

$$\| f \|_\infty = \sup_{t \in \mathcal{D}} |f(t)| = \lim_{p \longrightarrow +\infty} |f(t)|_p \tag{12}$$

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 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 $y'(t) = t^2$. A chaque fois qu'on donne une valeur à t on trouve $y'(t)$. D'un point de vue géométrique $y'(t)$ exprime la pente (vecteur directeur unitaire) de la droite tangente au point $t = t_0$, soit $y(t) = y'(t_0)\,(t - t_0) + y(t_0)$. Par conséquent, en calculant y' pour chaque point d'un plan, on obtient un champ de pentes donc des courbes intégrales.

Par un point de coordonnées (t_0, y_0) ne passe qu'une seule solution. Cela signifie qu'il n'existe qu'une solution vérifiant la condition initiale $y(t_0) = y_0$. Notons également qu'une équation différentielle d'ordre 1 possède une

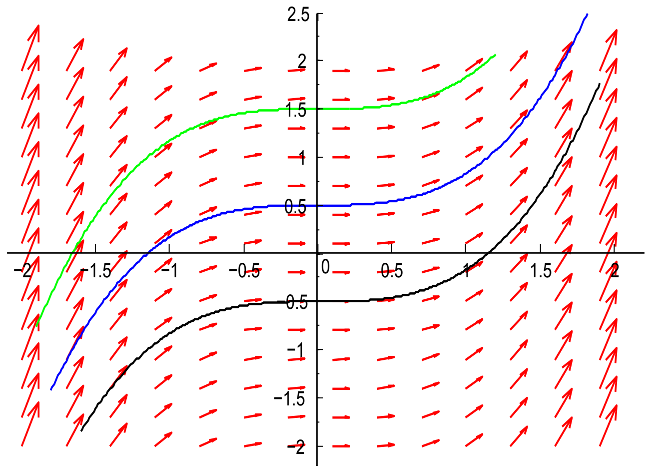

FIGURE 1: Champ de pentes associé à $y^{'}(t) = t^2$.

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 $y^{'}(t) = t^2$, 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'(t) = t^2
u = -2 :0.30: 2 ; echelle = 1.2 ; pas = 0.5 ;
[t, y] = meshgrid(u) ; dy = t.^2 ; dt = ones(size(dy)) ;

quiver(t, y, dt, 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('dt') ; ylabel('dy') ;
```

Nous fermons cette parenthèse sur les courbes intégrales, focalisons nous désormais sur la résolution numérique des équations différentielles. Dans ce qui suit, nous décrirons quelques méthodes permettant de résoudre numériquement le problème de *Cauchy*. Nous nous bornerons aux méthodes dites à *un pas*. Pour de telles méthodes, le calcul de $y(t_{n+1})$ ne réclame que la valeur de $y(t_n)$ à l'instant précédent. Une méthode *à deux pas* utilisera à la fois $y(t_n)$ et $y(t_{n-1})$.

B. Méthode d'Euler

Afin d'atteindre la solution $y(t)$, sur l'intervalle $t \in [a\,; b]$, on choisit $n+1$ points dissemblables $t_0, t_1, t_2, ..., t_n$, avec $t_0 = a$ et $t_n = b$ et le pas de discrétisation est défini par $h = (b-a)/n$. La résolution numérique consiste à discrétiser l'axe des abscisses suivant $t_n = t_0 + h\,n$ $(n \in \mathbb{N})$. Ensuite on cherchera u_n comme approximation de

y au point t_n, soit $y(t_n)$. 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 l'équation différentielle :

$$y^{'} = f(t, y(t)) \tag{13}$$

Trouver la solution de cette équation revient à calculer l'intégrale de $f(t, y(t))$ entre les bornes t_n et t_{n+1}, soit :

$$\int_{t_n}^{t_{n+1}} f(t, y(t))\, dt = y(t_{n+1}) - y(t_n) \tag{14}$$

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

$$\int_{t_n}^{t_{n+1}} f(t, y(t))\, dt = u_{n+1} - u_n \tag{15}$$

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** :

$$\begin{cases} u_0 = y(t_0) = y_0 \\ u_{n+1} - u_n = h\, f(t_n, u_n) \quad \text{avec} \quad n \in \mathbb{N} \end{cases} \tag{16}$$

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

$$\begin{cases} u_0 = y(t_0) = y_0 \\ u_{n+1} - u_n = h\, f(t_{n+1}, u_{n+1}) \quad \text{avec} \quad n \in \mathbb{N} \end{cases} \tag{17}$$

Avec la méthode du point milieu, on aura le schéma numérique **d'Euler modifié** :

$$\begin{cases} u_0 = y(t_0) = y_0 \\ u_{n+1/2} \simeq u_n + \frac{h}{2}\, f(t_n, u_n) \\ u_{n+1} - u_n = h\, f(t_n + \frac{h}{2}, u_{n+\frac{1}{2}}) \quad \text{avec} \quad n \in \mathbb{N} \end{cases} \tag{18}$$

Exercice ❶ ☞ ®

Soit à résoudre numériquement l'équation différentielle : $2\, y^{'} = t - y$ avec $y(0) = 1$. On utilisera à cet effet le schéma numérique d'Euler rétrograde :

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

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

Avec $t_n = t_0 + n\, h$ et $t_{n+1} = t_0 + (n+1)\, h$. On discrétise l'axe des abscisses suivant 10 nœuds sur un intervalle $[0\,;\, 1]$. Le pas de discrétisation est $h = (1-0)/10 \Rightarrow h = 0.10$ et la condition initiale est $y(t_0 = 0) = 1 = u_0$. Nous avons appliqué le schéma numérique ci-dessus pour dix approximations successives :

$$
\begin{aligned}
u_1 &= \left(\frac{2\,u_0 + h\,t_1}{2+h}\right)\\
u_2 &= \left(\frac{2\,u_1 + h\,t_2}{2+h}\right)\\
u_3 &= \left(\frac{2\,u_2 + h\,t_3}{2+h}\right)\\
u_4 &= \left(\frac{2\,u_3 + h\,t_4}{2+h}\right)\\
&\vdots \qquad\quad \vdots\\
u_{10} &= \left(\frac{2\,u_9 + h\,t_{10}}{2+h}\right)
\end{aligned}
$$

TABLE I: Méthode d'Euler

n	t_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 02/10/2019
% EULER IMPLICITE : y' = (t - y)/2
% SCHEMA NUMERIQUE : Un+1 = Un + F(tn+1, Un+1)

dy = @(t,y) (t - y)./2 ; un(1) = 1 ; % CONDITION INITIALE

a = 0 ; b = 1 ; n = 10 ; h = (b-a)/n ; tn = a:h:b ; % DISCRETISATION

for it = 1:n - 1

    un(it+1) = (1/(2+h))*(2*un(it) + h*tn(it)) ;
end

sol_exacte = 3.*exp(-tn/2) + tn - 2 ; figure('color',[1 1 1]) ;
plot(tn(1:end-1),un,'o-b')  ; hold on ; plot(tn, sol_exacte,'-ro') ;
xlabel('t') ; ylabel('y') ; legend('SOLUTION APPROCHEE','SOLUTION EXACTE') ;
```

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 02/10/2019
% FORMULE D'EULER - Eq : y' = (t - y)/2
% SCHEMA NUMERIQUE : Un+1 = Un + F(tn, Un)

dy = @(t,y) (t - y)./2 ; un(1) = 1 ; % CONDITION INITIALE

a = 0 ; b = 1 ; n = 10 ; h = (b-a)/n ; tn = a:h:b;

for it = 1:n - 1

    un(it+1) = un(it) + h*dy(tn(it), un(it)) ;
end

sol_exacte = 3.*exp(-tn/2) + tn - 2 ; figure('color',[1 1 1]) ;
plot(tn(1:end-1),un,'o-b')  ; hold on ; plot(tn, sol_exacte,'-ro') ;
xlabel('t') ; ylabel('y') ; legend('SOLUTION APPROCHEE','SOLUTION EXACTE') ;
```

C. *Erreur théorique*

La convergence des deux schémas d'Euler est d'ordre un par rapport à h :

$$e_n = |u(t_n) - u_n| = O(h)$$

Cela signifie que si je divise par deux le pas h, l'erreur sera divisée par deux également. Dans cette configuration le temps de calcul est multiplié par deux.

Théorème *: si f est continue sur $\mathbb{I} \times \mathbb{R}$, L-lipschitzienne par rapport à sa deuxième variable et $y(t) \in \mathcal{C}^2$ sur $\mathbb{I}$ alors l'erreur $e_n = u(t_n) - u_n$ commise au point (t_n, u_n) est majorée par :*

$$|e_n| \leq (e^{L\,(b-a)} - 1) \times \frac{h}{2\,L} \underset{t \in \mathbb{I}}{max} |y(t)^{(2)}| \tag{19}$$

Avec a et b sont respectivement les bornes inférieure et supérieure d'intégration. Afin de calculer cette erreur, soit le problème de Cauchy suivant :

$$\begin{cases} y^{'}(t) = y(t) + 1 \quad t \in \mathbb{R} \\ y(0) = 1 \end{cases} \tag{20}$$

1) Montrer que f est L-lipschitzienne.
2) Calculer l'erreur théorique de la méthode d'Euler pour un pas $h = 0.1$.

Tenant compte de (4), il vient :

$$\begin{aligned} |f(t,u) - f(t,v)| &= |t + u - t - v| \\ &= |u - v| \Rightarrow L = 1 \end{aligned}$$

Évaluons l'erreur théorique donnée par (19), nous avons $y^{''}(t) = 2\,e^t$. Cette seconde dérivée est maximale pour $x = 1 \Rightarrow y^{''}(1) = 2\,e^1 = 5.4366$:

$$|e_n| \leq (e^{1\,(1-0)} - 1) \times \frac{h}{2 \times 1} \times \frac{5.4366}{2 \times 1} \times 0.1$$
$$\leq 0.4665$$

D. Stabilité des schémas numériques

Avant d'entamer la discussion sur la notion de stabilité, il a été jugé judicieux d'introduire un bref rappel sur les suites. Soit u_n une suite géométrique de terme u_0 et de raison r. Nous avons $\forall n \in \mathbb{N} : u_n = u_0\, r$:

$$\text{o} \quad \text{Si} \quad r > 0 \Rightarrow \begin{cases} \lim_{n\to+\infty} u_n = +\infty & \text{si} \quad u_0 > r \\ \lim_{n\to+\infty} u_n = -\infty & \text{si} \quad u_0 < r \end{cases}$$

$$\text{o} \quad \text{Si} \quad |r| < 1 \Rightarrow \lim_{n\to+\infty} u_n = 0$$

$$\text{o} \quad \text{Si} \quad r = 1 \Rightarrow \lim_{n\to+\infty} u_n = u_0$$

$$\text{o} \quad \text{Si} \quad r \leq -1 \Rightarrow \lim_{n\to+\infty} u_n \quad \nexists$$

Dans ce qui suit on se propose d'étudier la stabilité des schémas d'Euler, progressif et rétrograde sur l'équation différentielle :

$$\begin{cases} y^{'}(t) = -\beta y(t) \quad \beta > 0 \\ y(0) = y_0 \end{cases} \tag{21}$$

La solution de ce problème de Cauchy est triviale, soit : $y(t) = y_0\, e^{-\beta\, t}$ ainsi :

$$\lim_{t\to+\infty} y(t) = 0 \quad (P_1)$$

Les schémas numériques d'Euler permettent de calculer les approximations $\{u_n\}_{\in\mathbb{N}}$. Nous souhaitons vérifier la propriété P_1 pour les approximations $\{u_1, u_2, u_3, ..., u_n\}$. Autrement dit on cherche :

$$\lim_{n\to+\infty} u_n = 0$$

Commençons par le schéma d'Euler progressif $u_{n+1} = u_n + h\, f(t_n, u_n)$, ce qui donne :

$$u_{n+1} = (1 - \beta\, h)\, u_n$$
$$u_1 = (1 - \beta\, h)\, u_0$$
$$u_2 = (1 - \beta\, h)\, u_1 = (1 - \beta\, h)^2\, u_0$$

En généralisant on obtient la suite géométrique : $u_{n+1} = (1 - \beta\, h)^n\, u_0$ de raison $(1 - \beta\, h)$ et de premier terme u_0. Cherchons la propriété P_1 :

$$\lim_{n\to+\infty} u_n = 0 \Leftrightarrow |1 - \beta\, h| < 1 \Leftrightarrow \beta\, h < 2 \Leftrightarrow h < \frac{2}{\beta}$$

La convergence de cette suite est conditionnée par $h < \frac{2}{\beta}$. Si nous prenons par exemple $h > \frac{2}{\beta}$ la suite divergera, ce qui contredit la solution exacte $y(t)$. Examinons désormais le schéma d'Euler rétrograde. De façon analogue que précédemment on aura :

$$u_{n+1} = u_n \left(\frac{1}{1 + \beta\, h}\right) = u_0 \left(\frac{1}{1 + \beta\, h}\right)^{n+1}$$

Cherchons la propriété P_1 :

$$\lim_{n\to+\infty} u_{n+1} = 0 \Leftrightarrow \frac{1}{1+\beta\, h} < 1$$

Ce qui est toujours vérifié $\forall\, h$. Par conséquent il n'existe aucune condition sur la pas de discrétisation, contrairement au schéma progressif. Ainsi, le schéma rétrograde est inconditionnellement stable.

E. Méthode de Heun

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

$$\begin{cases} u_0 = y(t_0) = y_0 \\ u_{n+1} = u_n + \dfrac{h}{2}\left\{f(t_n, u_n) + f(t_n, u_n + h\, f(t_n, u_n))\right\} \quad \text{avec} \quad n \in \mathbb{N} \end{cases} \tag{22}$$

Le schéma numérique de cette méthode résulte de l'application de la formule de quadrature du trapèze. Notons également que la méthode de Heun 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 :

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

Finalement :

$$u_{n+1} = \left(\frac{4\,h - h^2}{8}\right) t_n + \left(\frac{8 - 4\,h + h^2}{8}\right) u_n \tag{23}$$

Ci-dessous le script Matlab® :

```
clc ; clear all ; close all ;
% Samir KENOUCHE - Le 02/10/2019
% Heun - Eq : y' = (t - y)/2
% SCHEMA NUMERIQUE : Un+1 = Un + h/2[F(tn+1, Un+1) + F(tn, Un + h F(tn,Un))]

dy = @(t,y) (t - y)./2 ; un(1) = 1 ; % CONDITION INITIALE

a = 0 ; b = 1 ; n = 10 ; h = (b-a)/n ; tn = a:h:b;

for it = 1:n - 1

    un(it+1) = ((4*h - h.^2)/8)* tn(it) + ((8 - 4*h + h.^2)/8)*un(it) ;

end
```

```
sol_exacte = 3.*exp(-tn/2) + tn - 2 ; figure('color',[1 1 1]) ;
plot(tn(1:end-1),un,'o-b') ; hold on ; plot(tn, sol_exacte,'-ro') ;
xlabel('t') ; ylabel('y') ; legend('SOLUTION APPROCHEE','SOLUTION EXACTE') ;
```

Comme précédemment, cette façon de procéder n'est faisable que 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 02/10/2019
% Heun - Eq : y' = (t - y)/2
% SCHEMA NUMERIQUE : Un+1 = Un + h/2[F(tn+1, Un+1) + F(tn, Un + h F(tn,Un))]

dy = @(t,y) (t - y)./2 ; un(1) = 1 ; % CONDITION INITIALE


a = 0 ; b = 1 ; n = 10 ; h = (b-a)/n ; tn = a:h:b;

for i = 1:n - 1

    un(i+1) = un(i) + h/2*(dy(tn(i),un(i)) + dy(tn(i + 1),...
     un(i) + h*dy(tn(i),un(i))));
end

sol_exacte = 3.*exp(-tn/2) + tn - 2 ; figure('color',[1 1 1]) ;
plot(tn(1:end-1),un,'o-b') ; hold on ; plot(tn, sol_exacte,'-ro') ;
xlabel('t') ; ylabel('y') ; legend('SOLUTION APPROCHEE','SOLUTION EXACTE') ;
```

Ci-dessous, les résultats numériques pour dix approximations successives :

TABLE II: Méthode de Heun

n	t_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

F. Méthode de Crank - Nicolson

Cette méthode permet d'obtenir une plus grande précision (elles génèrent des solutions numériques plus proches des solutions analytiques) que les deux méthodes précédentes. Le schéma numérique est donné par :

$$\begin{cases} y_0 = y(t_0) = u_0 \\ u_{n+1} = u_n + \dfrac{h}{2}\left\{f(t_n, u_n) + f(t_{n+1}, u_{n+1})\right\} \quad \text{avec} \quad n \in \mathbb{N} \end{cases} \tag{24}$$

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

G. 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 :

$$\begin{cases} y_0 = y(t_0) = u_0 \\ u_{n+1} = u_n + \dfrac{h}{6}\left(f(t_n, u_{1n}) + 2\,f(t_n + \dfrac{h}{2}, u_{2n}) + 2\,f(t_n + \dfrac{h}{2}, u_{3n}) + f(t_{n+1}, u_{4n})\right) \end{cases} \quad \text{avec} \quad n \in \mathbb{N} \tag{25}$$

$$\begin{cases} u_{1n} = u_n \\ u_{2n} = u_n + \frac{h}{2}\,f(t_n, u_{1n}) \\ u_{3n} = u_n + \frac{h}{2}\,f(t_n + \frac{h}{2}, u_{2n}) \\ u_{4n} = u_n + h\,f(u_n + \frac{h}{2}, u_{3n}) \end{cases} \tag{26}$$

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

Exercice ❶ ☞ ®

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 :

$$\begin{cases} y' = \dfrac{-y\,t^2 - y^2 + 2\,t}{1 - t^3} \\ y(0) = 1 \end{cases} \tag{27}$$

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

Voici le script Matlab® :

```
clear all ; close all ; clc ;
% 02/10/2019 Samir KENOUCHE : ALGORITHME PERMETTANT
% L'IMPLEMENTATION, SOUS MATLAB, DE METHODES NUMERIQUES (Euler, Heun,
% Runge-Kutta) POUR LA RESOLUTION D'EQUATIONS DIFFERENTIELLES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dy = @(tn, u) (-u.*tn.^2 - u.^2 + 2.*tn)./(1 - tn.^3) ;
a = 0 ; b = 1 ; n = 50 ; h = (b-a)/n ; tn = 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(tn(i),u1(i)) ;

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

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

```
figure('color',[1 1 1]) ; plot(tn(1:end-1),u,'o-g') ;
%%%%%%%%%%%%%%%%%%%%%%%%%% METHODE DE Heun %%%%%%%%%%%%%%%%%%%%%%%%%%
dy = @(tn, u) (-u.*tn.^2 - u.^2 + 2.*tn)./(1 - tn.^3) ;
a = 0 ; b = 1 ; n = 50 ; h = (b-a)/n ; tn = a:h:b;
epsilon = 0.0001 ; u(1) = 1 + epsilon;

for i = 1:n - 1

  u(i+1) = u(i) + h/2*(dy(tn(i),u(i)) + dy(tn(i + 1),...
     u(i) + h*dy(tn(i),u(i))));
end

tn = tn(1:end-1) ; hold on ; plot(tn,u,'o-r') ;
%%%%%%%%%%%%%%%%%%%%%%%%%% METHODE D'Euler %%%%%%%%%%%%%%%%%%%%%%%%%%
clear all ; clc ;
dy = @(tn, u) (-u.*tn.^2 - u.^2 + 2.*tn)./(1 - tn.^3) ;

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

for i = 1:n - 1

  u(i+1) = u(i) + h/2*(dy(tn(i),u(i))) ;

end

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

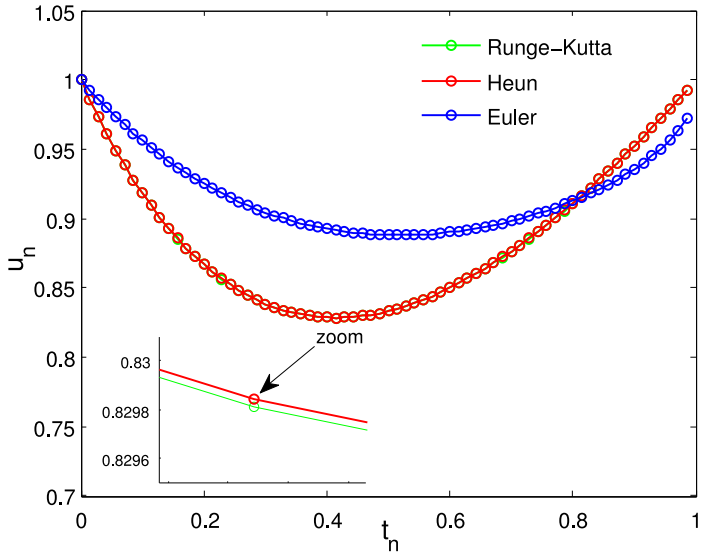

FIGURE 2: Solutions numériques obtenues par les méthodes d'Euler, de Heun et de Runge-Kutta d'odre 4

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 :

$$\begin{cases} y'' = \dfrac{t\,y'}{2} - y + 3 \\ y(0) = 1 \quad \text{et} \quad y'(0) = 0 \end{cases} \tag{28}$$

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 (28).
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 :

$$\begin{cases} y'''(t) = 0.001\,\left(y''(t) + (1 - y(t)^2)\right) \times y'(t) + sin(t) \\ y(0) = 1, \quad y'(0) = 5, \quad \text{et} \quad y''(0) = 0 \end{cases} \tag{29}$$

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)') ;
```

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

$$\begin{cases} y'''(t) = 0.001\,\left(y''(t) + (1 - y(t)^2)\right) \times y'(t) + sin(t) \\ y(0) = 1, \quad y'(0) = 5, \quad \text{et} \quad y''(0) = 0 \end{cases} \tag{30}$$

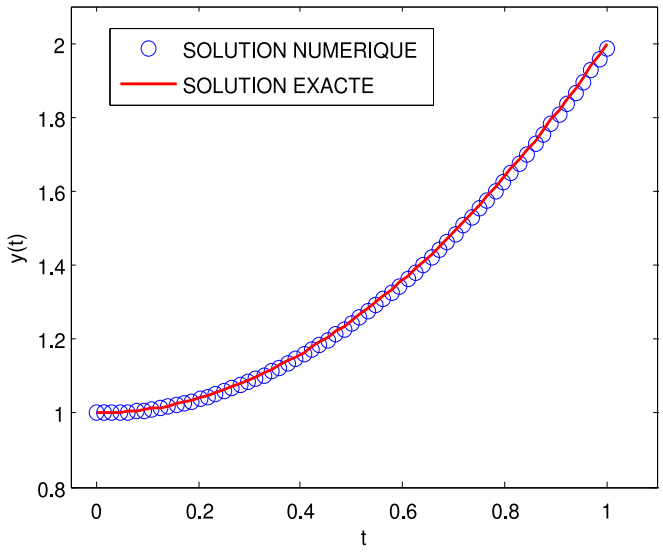

FIGURE 3: Solutions exacte et numérique obtenues par la méthode d'Euler

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 les résultats renvoyés par le script :

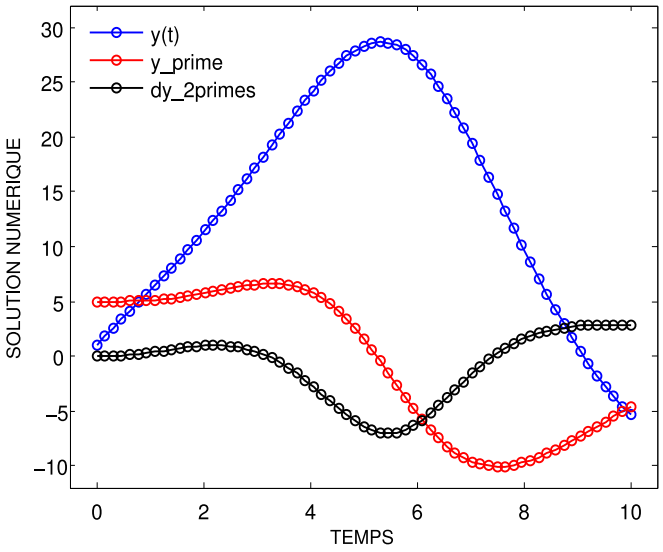

FIGURE 4: Équation différentielle du troisième ordre résolue par la méthode d'Euler

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 :

$$\begin{cases} y''(t) + 3\,y = 4\,\sin(t) + \exp(-t) \\ y(0) = 1, \quad y'(0) = 1 \end{cases} \tag{31}$$

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') ;
```

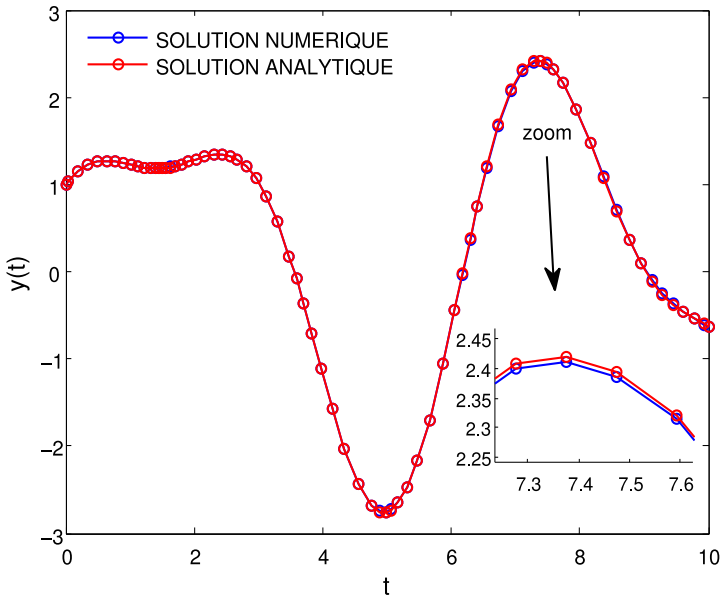

FIGURE 5: Solutions analytique et numérique générées par le solveur `ode23`

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 :

$$\begin{cases} y''(t) - \alpha\,(1-y^2)\,y' = -y \\ y(0) = 1, \quad y'(0) = 0 \end{cases} \tag{32}$$

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
```

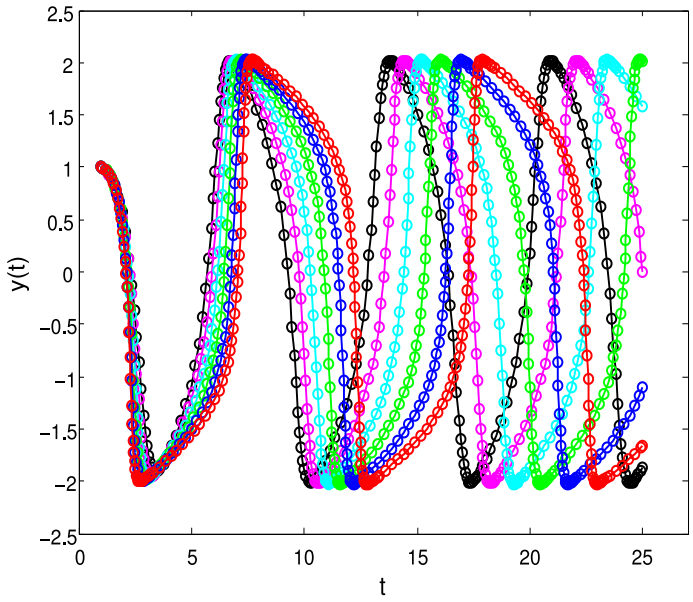

FIGURE 6: Solution numérique $y(t)$ pour différentes valeurs de α

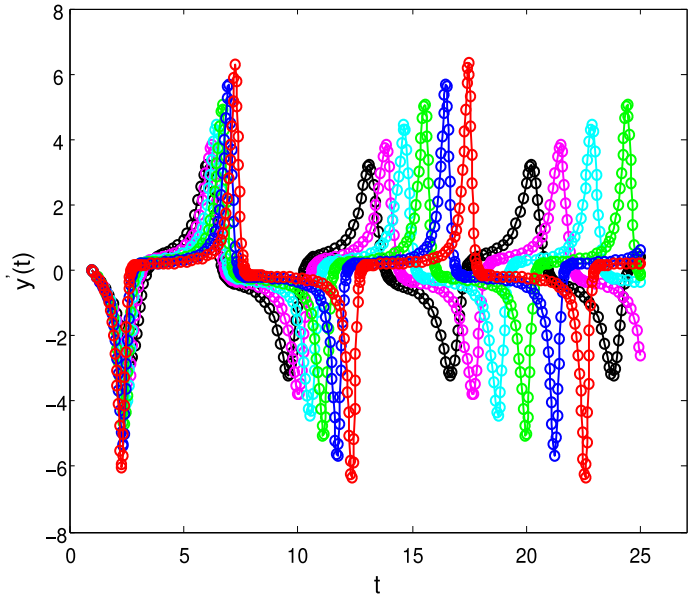

FIGURE 7: Dérivée première de la solution numérique $y(t)$ pour différentes valeurs de α

Exercice ❸ ☞ Ⓢ

1) Résoudre numériquement les équations différentielles suivantes :

$$\begin{cases} y^{'} + y\,\cos(x) = \cos(x), \quad y(0) = 1 \\ y^{'} = x + \cos(y), \quad y(0) = 0 \\ y^{''} + 2\,y^{'} = -2y, \quad y(0) = 2, y^{'}(0) = -3 \\ y^{''} - y^{'} - 2y = 2\,x\exp(-x) + x^2, \quad y(0) = 0, y^{'}(0) = 1 \\ y^{''} + y = 3\,x^2 - 4\sin(x), \quad y(0) = 0, y^{'}(0) = 1 \\ y^{'''} + y^{'} = x, \quad y(0) = 0, y^{'}(0) = 1, y^{''}(0) = 1 \\ y^{''''} - y = 8\,\exp(x), \quad y(0) = 0, y^{'}(0) = 2, y^{''}(0) = 4, y^{'''}(0) = 6 \end{cases} \tag{33}$$

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