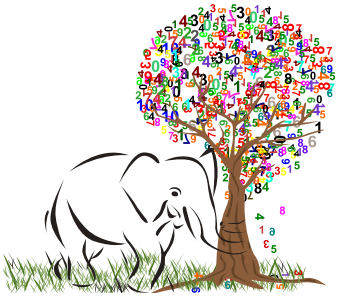

Chapitre[2] Résolution numérique des équations différentielles

Master Physique

SAMIR KENOUCHE - DÉPARTEMENT DES SCIENCES DE LA MATIÈRE - UMKB

MÉTHODES MATHÉMATIQUES ET ALGORITHMES POUR LA PHYSIQUE

Résumé

Dans ce chapitre, il sera question de la résolution numérique des équations différentielles. Les méthodes numériques 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, de Heun, de Crank-Nicolson et de Runge-Kutta classique d'ordre 4. Dans la dernière section, on abordera la résolution d'une équation aux dérivées partielles par la méthode des différences finies en deux dimensions. Par ailleurs, la programmation de ces algorithmes sera conduite par le biais de scripts Matlab®.

Mots clés

Méthode d'Euler, de Heun, Runge Kutta, équation aux dérivées partielles, script Matlab®.

Galilée disait " ... Le livre de la nature est écrit en langage mathématique "

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

S. Kenouche est docteur en Physique de l'Université de Montpellier et docteur en Chimie de l'Université de Béjaia.
Site web : voir http://www.sites.univ-biskra.dz/kenouche
Version améliorée et actualisée le 09.11.2018.

que très souvent la solution analytique n'existe pas, et on doit par conséquent approcher la solution exacte $y(x)$ par des méthodes numériques.

A. *Problème de Cauchy*

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

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

$$|f(t, u) - f(t, v)| \leq L\,|u - v| \tag{3}$$

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

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 $\mathbb{I}$, $\forall\, u(0) \in \mathbb{R}$.

Remarque : Dans ce qui suit, la variable t sera systématiquement remplacée par la variable x. Le formalisme mathématique demeure inchangé.

B. *Méthodes 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 l'équation différentielle :

$$y^{'} = f(x, y(x)) \tag{4}$$

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 :

$$\int_{x_n}^{x_{n+1}} f(x, y(x)) = y(x_{n+1}) - y(x_n) \tag{5}$$

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

$$\int_{x_n}^{x_{n+1}} f(x, y(x)) = u_{n+1} - u_n \tag{6}$$

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** :

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

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

$$\begin{cases} u_0 = y(x_0) = y_0 \\ u_{n+1} - u_n = h\,f(x_{n+1}, u_{n+1}) \quad \text{avec} \quad n \in \mathbb{N} \end{cases} \tag{8}$$

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

$$\left\{\begin{array}{l} u_0 = y(x_0) = y_0 \\ u_{n+1/2} \simeq u_n + \frac{h}{2} f(x_n, u_n) \\ u_{n+1} - u_n = h\, f(x_n + \frac{h}{2}, u_{n+\frac{1}{2}}) \quad \text{avec} \quad n \in \mathbb{N} \end{array}\right. \tag{9}$$

Exemple numérique

Soit à résoudre numériquement l'équation différentielle : $2\,y^{'} = x - y$ avec $y(0) = 1$. On utilisera à cet effet le schéma numérique d'Euler rétrograde :

$$\begin{aligned} u_{n+1} =& u_n + h\left(\frac{x_{n+1} - u_{n+1}}{2}\right) \\ 2\,u_{n+1} =& 2\,u_n + h\,x_{n+1} - h\,u_{n+1} \\ u_{n+1} =& \left(\frac{2\,u_n + h\,x_{n+1}}{2+h}\right) \end{aligned} \tag{10}$$

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 $[0 \quad 1]$. 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 :

$$\begin{aligned} u_1 &= \left(\frac{2\,u_0 + h\,x_1}{2+h}\right), \quad \text{avec} \quad h = 0.1 \\ u_2 &= \left(\frac{2\,u_1 + h\,x_2}{2+h}\right), \quad \text{avec} \quad h = 0.1 \\ u_3 &= \left(\frac{2\,u_2 + h\,x_3}{2+h}\right), \quad \text{avec} \quad h = 0.1 \\ u_4 &= \left(\frac{2\,u_3 + h\,x_4}{2+h}\right), \quad \text{avec} \quad h = 0.1 \\ \vdots &= \quad \vdots \quad\quad\quad\quad \vdots \\ u_{10} &= \left(\frac{2\,u_9 + h\,x_{10}}{2+h}\right), \quad \text{avec} \quad h = 0.1 \end{aligned}$$

TABLE I: Méthode d'Euler

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 15/11/2018
% 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') ;
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 15/11/2018
% 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') ;
xlabel('x') ; ylabel('y') ;
```

C. Erreur théorique

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

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

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 $y(x)\in\mathcal{C}^2$ sur $\mathbb{I}$ alors l'erreur $e_n = u(x_n) - u_n$ commise au point (x_n, u_n) est majorée par :

$$|e_n| \leq (e^{L\,(b-a)} - 1) \times \frac{h}{2\,L} \max_{x\in\mathbb{I}} |y(x)^{(2)}| \tag{11}$$

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

$$\begin{cases} y^{'}(x) = y(x) + 1 \quad x\in\mathbb{R} \\ y(0) = 1 \end{cases} \tag{12}$$

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 (3), il vient :

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

Évaluons l'erreur théorique donnée par (11), nous avons $y^{''}(x) = 2\,e^x$. Cette seconde dérivée est maximale pour $x = 1 \Rightarrow y^{''}(1) = 2\,e^1 = 5.4366$:

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

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 $\forall n\in\mathbb{N} : u_n = u_0\,r$:

- Si $\quad r > 0 \Rightarrow \begin{cases} \lim_{n\to+\infty} u_n = +\infty & \text{si} \quad u_0 > r \\ \lim_{n\to+\infty} u_n = -\infty & \text{si} \quad u_0 < r \end{cases}$
- Si $\quad |r| < 1 \Rightarrow \lim\limits_{n\to+\infty} u_n = 0$
- Si $\quad r = 1 \Rightarrow \lim\limits_{n\to+\infty} u_n = u_0$
- Si $\quad r \leq -1 \Rightarrow \lim\limits_{n\to+\infty} u_n \quad \nexists$

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

$$\begin{cases} y^{'}(x) = -\beta y(x) \quad \beta > 0 \\ y(0) = y_0 \end{cases} \tag{13}$$

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

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

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 $u_{n+1} = u_n + h\, f(x_n, u_n)$, ce qui donne :

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

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 :

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

Ainsi, 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 $y(x)$. 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.

$$\begin{cases} u_0 = y(x_0) = y_0 \\ u_{n+1} = u_n + \dfrac{h}{2}\left\{f(x_n, u_n) + f(x_n, u_n + h\, f(x_n, u_n))\right\} \quad \text{avec} \quad n \in \mathbb{N} \end{cases} \tag{14}$$

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 :

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

Finalement :

$$u_{n+1} = \left(\frac{4\, h - h^2}{8}\right) x_n + \left(\frac{8 - 4\, h + h^2}{8}\right) u_n \tag{15}$$

Ci-dessous le script Matlab® :

```
clc ; clear all ; close all ;
% Samir Kenouche - Le 15/11/2018
% 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 it = 1:n - 1

    un(it+1) = ((4*h - h.^2)/8)* xn(it) + ((8 - 4*h + h.^2)/8)*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') ;
xlabel('x') ; ylabel('y') ;
```

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 15/11/2018
% 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') ;
xlabel('x') ; ylabel('y') ;
```

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

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 :

$$\begin{cases} y_0 = y(x_0) = u_0 \\ u_{n+1} = u_n + \dfrac{h}{2}\left\{f(x_n, u_n) + f(x_{n+1}, u_{n+1})\right\} \quad \text{avec} \quad n \in \mathbb{N} \end{cases} \tag{16}$$

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 :

$$\begin{cases} y_0 = y(x_0) = u_0 \\ u_{n+1} = u_n + \dfrac{h}{6}\left(f(x_n, u_{1n}) + 2\,f(x_n + \dfrac{h}{2}, u_{2n}) + 2\,f(x_n + \dfrac{h}{2}, u_{3n}) + f(x_{n+1}, u_{4n})\right) \quad \text{avec} \quad n \in \mathbb{N} \end{cases} \tag{17}$$

$$\begin{cases} u_{1n} = u_n \\ u_{2n} = u_n + \frac{h}{2}\,f(x_n, u_{1n}) \\ u_{3n} = u_n + \frac{h}{2}\,f(x_n + \frac{h}{2}, u_{2n}) \\ u_{4n} = u_n + h\,f(u_n + \frac{h}{2}, u_{3n}) \end{cases} \tag{18}$$

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 :

$$\begin{cases} y' = \dfrac{-y\,x^2 - y^2 + 2\,x}{1 - x^3} \\ y(0) = 1 \end{cases} \tag{19}$$

2) Afficher sur la même figure, la solution des trois méthodes.
3) Analyser l'erreur en fonction du pas de discrétisation pour la méthode de Runge-Kutta d'ordre 4.

4) Connaissant la solution exacte de l'équation différentielle ci-dessous. Proposer une démarche permettant la détermination de l'ordre de convergence de la méthode de Runge-Kutta. On donne :

$$\begin{cases} y' = \dfrac{x-y}{2} \\ y(0) = 1 \end{cases} \tag{20}$$

La solution exacte est : $3\,exp(-x/2) + x - 2$

5) Tracer le graphique donnant l'erreur relative, pour chaque pas de discrétisation, en fonction du nombre d'itérations.
6) Tracer le graphique donnant le maximum de l'erreur relative en fonction du pas de discrétisation.

Voici le script Matlab® :


```
clear all ; close all ; clc ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 17/11/2018 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 %%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
clear all ; clc ;
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')
```

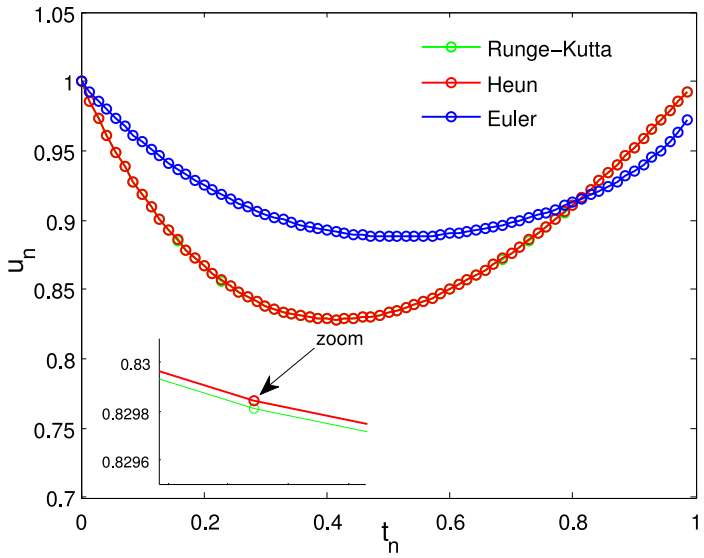

FIGURE 1: Solutions numériques obtenues par les méthodes d'Euler, de Heun et de Runge-Kutta d'odre 4

```
clear all ; close all ; clc ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 18/11/2018 Samir KENOUCHE : ANALYSE DE L'ERREUR EN
% FONCTION DU PAS DE DISCRITISATION DANS LE CADRE DE LA METHODE DE
% Runge-Kutta D'ORDRE 4

dy = inline('(xn - u)/2','xn','u');  % EQUATION DIFFERENTIELLE
sol_exacte = inline('3*exp(-xn/2) - 2 + xn','xn') ; % SA SOLUTION EXACTE
a = 0 ; b = 6 ; ik = 0 ; u(1) = 1  ;
col = {'k','b','r','m','c','g'};

for n = 60:10:100

    h = (b-a)/n ; xn = a :h: b;

  for i = 1: n - 1
```

```
   u1(i) = dy(xn(i),u(i)) + 2*dy(xn(i) + h/2,...
   u(i) + h/2*dy(xn(i),u(i))) ;

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

   u3(i) = dy(xn(i+1), u(i) + h*dy(xn(i)+h/2,...
   u(i)+h/2*dy(xn(i)+h/2,u(i) + h/2*dy(xn(i), u(i)))) ;

   u(i+1) = u(i) + h/6*(u1(i) + 2*u2(i) + u3(i)) ;

  end

ik = ik + 1;
xn = xn(1:end-1) ;
err  =  abs((sol_exacte(xn) - u)./sol_exacte(xn));

max_err(ik) = max(err) ;  pas(ik) = h ;
figure(1) ; hold on ; plot(err,col{ik}, 'LineWidth',1) ;
xlabel('NOMBRE D''ITERATIONS') ; ylabel('ERREUR RELATIVE')

end

figure(3) ; plot(xn,u,'o-') ; hold on ; plot(xn, sol_exacte(xn),'o-r') ;

for p = 1:6

[coeff, s] = polyfit(pas,max_err,p) ; % s ETANT L'ERREUR
% D'INTERPOLATION, POUR UN DEGRE p, GENEREE PAR LA FONCTION : polyfit
evalp = polyval(coeff, pas) ;
err_iterpolation(p) = max(abs(evalp - max_err));
end

[min_err_interp, degre_interp] = min(err_iterpolation);
coeff_opt = polyfit(pas,max_err,degre_interp);
eval_opt = polyval(coeff_opt, pas) ;

figure('color', [1 1 1]) ;
plot(pas, max_err,'r+','MarkerSize',10,'LineWidth',1)
hold on ; plot(pas,eval_opt) ; axis([0.05 0.11 0 7e-08])
xlabel('PAS DE DISCRETISATION') ; ylabel('MAXIMUM DE L''ERREUR RELATIVE')

str1 = {'CETTE METHODE EST D''ORDRE :', num2str(degre_interp)}
uicontrol('Style','text','Position',[260 80 150 60],...
          'BackgroundColor',[1 1 1],'String',str1) ;
```

L'affichage graphique restitué par le script ci-dessus est :

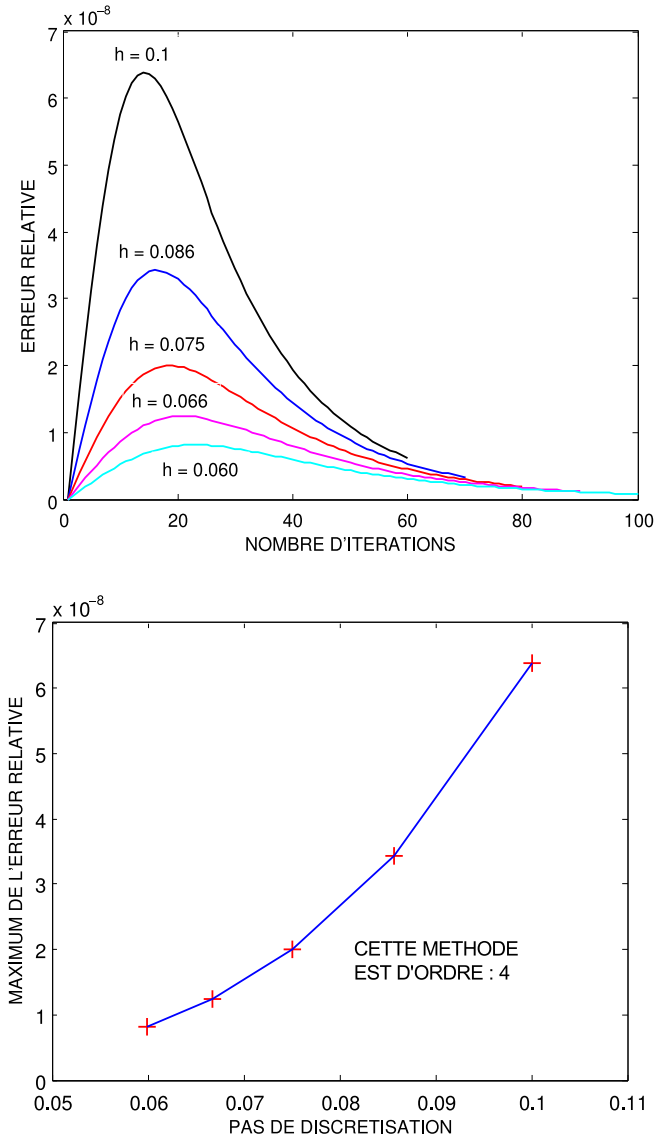

FIGURE 2: Évolution de l'erreur relative en fonction du pas de discrétisation

`s(p = 1) : normr = 9.3851e-09; s(p = 2) : normr = 7.4149e-10; s(p = 3) : normr = 1.6060e-11; s(p = 4) : normr = 9.9262e-24; s(p = 5) : normr = 1.9921e-23; s(p = 6) : normr = 2.8892e-23.` D'où le choix `p = 4`, qu'on retrouve aussi avec une démarche similaire en posant `err_iterpolation = max(abs(evalp - max_err))` et `[min_err_interp, degre_interp] = min(err_iterpolation)`. Il en ressort que `degre_interp = 4`. Avec un raisonnement analogue on peut aisément obtenir l'ordre des méthodes d'Euler et de Heun.

II. ÉQUATIONS DIFFÉRENTIELLES D'ORDRE n

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

$$\begin{cases} y'' = \dfrac{t\,y'}{2} - y + 3 \\ y(0) = 1 \quad \text{et} \quad y'(0) = 0 \end{cases} \tag{21}$$

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 ❷ ☞ ®

1) Résoudre numériquement, au moyen de la méthode d'Euler, l'équation différentielle du second ordre (21).
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 code Matlab® pour résoudre l'équation différentielle du troisième ordre suivante :

$$\begin{cases} y^{'''}(t) = 0.001\ \left(y^{''}(t) + (1 - y(t)^2)\right) \times y^{'}(t) + sin(t) \\ y(0) = 1, \quad y^{'}(0) = 5, \quad \text{et} \quad y^{''}(0) = 0 \end{cases} \tag{22}$$

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

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 ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 16/11/2018 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 :

$$\begin{cases} y^{'''}(t) = 0.001\ \left(y^{''}(t) + (1 - y(t)^2)\right) \times y^{'}(t) + sin(t) \\ y(0) = 1, \quad y^{'}(0) = 5, \quad \text{et} \quad y^{''}(0) = 0 \end{cases} \tag{23}$$

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 :

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

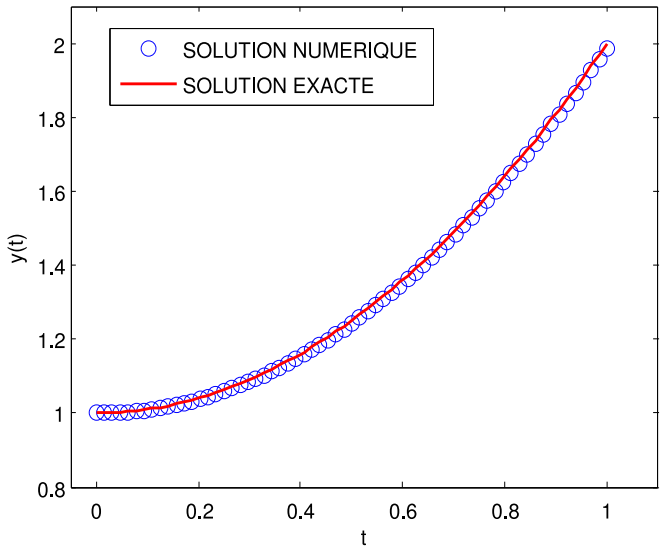

FIGURE 3: Solution exacte et solution numérique obtenues par la méthode d'Euler

```
clear all ; clc ; close all ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ALGORITHME PERMETTANT LA RESOLUTION D'EQUATIONS DIFFERENTIELLES
% DU 3EMME ORDRE
% Samir Kenouche - Le 16/11/2018
% 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') ;
```

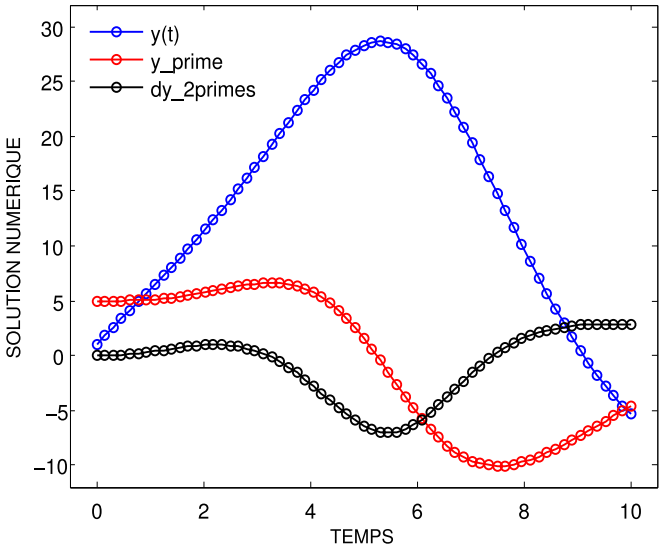

FIGURE 4: Équation différentielle du troisième ordre résolue par la méthode d'Euler

III. DIFFÉRENCES FINIES

A. *En dimension 1*

Dans cette section, nous considérons le problème de la corde élastique fixée aux extrémités $x = 0$ et $x = L$ telle que L est égale à une unité de longueur. La corde subit des déformations selon un mode vertical. L'amplitude des déformations est décrite par la fonction $u(x)$, ainsi le problème est formalisé mathématiquement par :

$$\begin{cases} -u''(x) = f(x), \quad 0 < x < L \\ \\ u(0) = 0 \\ u(L) = 0 \end{cases} \tag{24}$$

Autrement dit l'Eq. (24) est l'équation différentielle régissant la déformation linéaire de la corde élastique. Le second terme représente la source des déformations. Les conditions aux limites $u(0) = 0$ et $u(L) = 0$ traduisent le fait que la corde ne subit pas de déformations aux extrémités. Il s'agit d'une résolution numérique, donc le calcul d'une approximation de $u(x_i)$ au point x_i. Nous commençons par la discrétisation des abscisses.

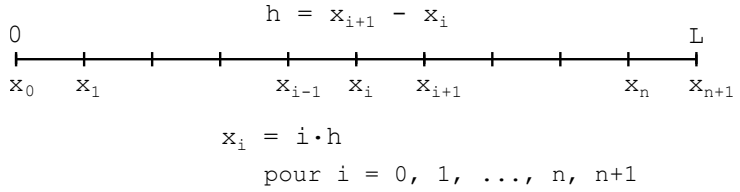

FIGURE 5: Discrétisation du problème aux limites 1 Dim

La discrétisation se fait avec un pas constant $h = x_{i+1} - x_i$ par conséquent tous les points x_i sont équidistants. La solution exacte au point x_i, soit $u(x_i)$ est inconnue. Nous cherchons des solutions approchées u_i qui sont des approximations de la solution exacte $u(x_i)$ au point x_i.

Utilisons une formule des différences finies centrée afin d'approcher $u''(x)$, soit :

$$-\frac{u(x_{i-1}) - 2\,u(x_i) + u(x_{i+1})}{h^2} = f(x_i) + O(h^2) \tag{25}$$

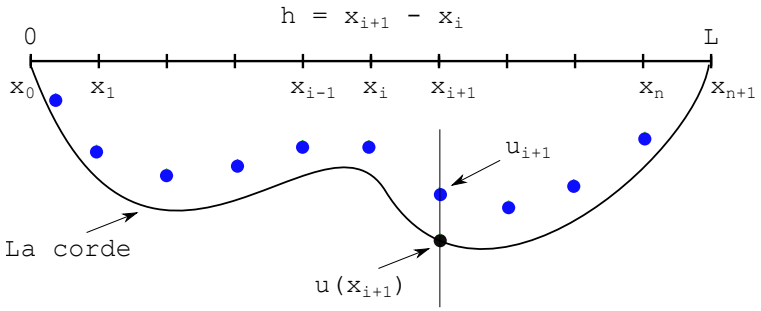

FIGURE 6: Solution exacte $u(x_i)$ versus solution approchée u_i

Le terme $O(h^2)$ stipule que lorsqu'on divise h par une constante a, l'erreur $|u(x_i) - u_i|$ est divisée par a^2. C'est le résultat du théorème suivant :

Si $u(x)$ est de classe $\mathcal{C}^4$ sur $[0\ L]$, alors $\exists\ c \in \mathbb{R}^+$ telle que $\forall\ 0 < h < L$ nous avons :

$$\max_{1 \leq i \leq 1} |u(x_i) - u_i| \leq c\, h^2$$
$$\max_{1 \leq i \leq 1} |u(x_i) - u_i| \leq \frac{1}{96} \max_{0 \leq x \leq L} |u(x)^{(4)}|$$

Ce théorème affirme que l'erreur d'intégration est majorée par la quatrième dérivée de la fonction $u(x)$ et plus h est petit plus on s'approche de la solution exacte. Écrivons maintenant la même formule des différences finies pour les approximations u_i, il vient :

$$\begin{cases} -\dfrac{u_{i-1} - 2\,u_i + u_{i+1}}{h^2} = f(x_i), \quad i = 1, 2, ..., n \\ \\ u(x_0) = u_0 \\ u(L) = u_{n+1} \end{cases} \tag{26}$$

Le terme $O(h^2)$ n'est pas pris en considération dans les calculs. Le schéma (26) correspond à la résolution d'un système linéaire.

$$A_n\,(\mathbb{R}^n \times \mathbb{R}^n)\,u_n\,(\mathbb{R}^n) = f_n\,(\mathbb{R}^n) \tag{27}$$

Explicitons le schéma (26) pour $n = 4$ et pour des conditions aux limites $u_0 = \alpha$ et $u_{n+1} = \beta$. Ces conditions aux limites signifient qu'à $x = 0$ la corde subit une déformation constante égale à la valeur α et à l'autre extrémité $x = L$, la corde subit aussi une déformation constante égale à la valeur β. Nous aurions pu prendre par exemple $u_0 = 0$ et $u_{n+1} = 0$, cela indique que la corde ne subit aucune déformation aux extrémités. Nous préférons prendre un cas général avec les conditions $u_0 = \alpha$ et $u_{n+1} = \beta$.

$$\begin{cases} \dfrac{-u_0+2\,u_1-u_2}{h^2}=f(x_1), \quad i=1 \\ \dfrac{-u_1+2\,u_2-u_3}{h^2}=f(x_2), \quad i=2 \\ \dfrac{-u_2+2\,u_3-u_4}{h^2}=f(x_3), \quad i=3 \\ \dfrac{-u_3+2\,u_4-u_5}{h^2}=f(x_4), \quad i=4 \end{cases} \Rightarrow \begin{cases} -\alpha+2\,u_1-u_2=h^2\,f(x_1), \quad i=1 \\ -u_1+2\,u_2-u_3=h^2\,f(x_2), \quad i=2 \\ -u_2+2\,u_3-u_4=h^2\,f(x_3), \quad i=3 \\ -u_3+2\,u_4-\beta=h^2\,f(x_4), \quad i=4 \end{cases}$$

$$\Rightarrow \begin{cases} 2\,u_1-u_2=h^2\,f(x_1)+\alpha, \quad i=1 \\ -u_1+2\,u_2-u_3=h^2\,f(x_2), \quad i=2 \\ -u_2+2\,u_3-u_4=h^2\,f(x_3), \quad i=3 \\ -u_3+2\,u_4=h^2\,f(x_4)+\beta, \quad i=4 \end{cases} \tag{28}$$

En adoptant une notation matricielle, le problème peut s'écrire :

$$A_n\,u_n=f_n \tag{29}$$

Avec,

$$A_n=\frac{1}{h^2}\begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix} \in \mathbb{R}^4 \times \mathbb{R}^4 \tag{30}$$

$$f_n=\begin{pmatrix} h^2\,f(x_1)+\alpha \\ h^2\,f(x_2) \\ h^2\,f(x_3) \\ h^2\,f(x_1)+\beta \end{pmatrix} \in \mathbb{R}^4 \tag{31}$$

La matrice A_n est tridiagonale, symétrique et définie positive. Le vecteur des valeurs de la solution (inconnues) aux points x_i est donné

$$u_h=\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} \in \mathbb{R}^4 \tag{32}$$

Ce schéma se généralise pour i = $\{1, 2, ..., n\}$, selon :

$$A_n = \begin{pmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{pmatrix} \in \mathbb{R}^n \times \mathbb{R}^n \quad f_n = \begin{pmatrix} h^2\, f(x_1) + \alpha \\ f(x_2) \\ \vdots \\ f(x_{n-1}) \\ h^2\, f(x_n) + \beta \end{pmatrix} \in \mathbb{R}^n \quad u_n = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{pmatrix} \in \mathbb{R}^n$$

Exercice ❸ ☞ ®

Soit l'équation :

$$\begin{cases} -u^{''}(x) = f(x) \quad 0 \leq x \leq 1 \\ \\ u(0) = 0 \\ u(1) = 0 \end{cases} \tag{33}$$

Le second terme de l'équation vaut :

$$f(x) = e^{3\,x^2} \times (x+1) \tag{34}$$

1) Résoudre numériquement pour n = 100 l'équation (33) par la méthode des différences finies.

Voici le script Matlab®

```
clear all ; clc ; close all ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SAMIR KENOUCHE - RESOLUTION NUMERIQUE DU PROBLEME AUX LIMITES DE LA CORDE
% ELASTIQUE : - u''(x) = f(x) AVEC LES CONDITIONS u(0) = 0 et u(1) = 0

np = 100 ; pas_x = 1/(np+1) ; xi = 0: pas_x :1 ; % DISCRETISATION
fx = @(xi) exp(3.*xi.^2).*(xi + 1) ;

sur_diag = diag(ones(np - 1, 1) ,1)*(-1) ;  % SUR-DIAGONALE
des_diag = diag(ones(np - 1, 1) ,-1)*(-1) ; % SOUS-DIAGONALE
in_diag = diag(ones(np, 1))*(2) ;           % DIAGONALE PRINCIPALE

An = sur_diag + des_diag + in_diag ;        % MATRICE An
fn = fx(xi(2:end-1)) ;                      % SOURCE DE LA DEFORMATION

un = inv(An)*fn' ;                          % CALCUL DES APPROXIMATIONS
un = [0 un' 0] ;                      % ON RAJOUTE LES CONDITIONS AUX LIMITES

fig1 = figure('color',[1 1 1]) ; plot(xi, un,'o') ;
xlabel('x_i') ; ylabel('u_i') ; title('SOLUTION APPROCHEE') ;
```

Considérons désormais des fonctions $c(x)$ et $f(x)$ continues sur l'intervalle $[a,b]$. On se propose de résoudre, par les différences finies, l'équation de la convection-diffusion. Soient α et β deux constantes $\in \mathbb{R}$. Le problème consiste à trouver une fonction $u(x)$ deux fois dérivable sur l'intervalle $[a,b]$ et qui satisfait :

$$\begin{cases} -u^{''}(x) + c(x)\,u^{'}(x) = f(x), \quad x \in [a,b] \\ u(a) = 0 \quad \text{et} \quad u(b) = 0 \end{cases} \tag{35}$$

Comme précédemment, ce type de problème est dénommé **problème aux limites**. Cette dénomination provient du fait que la fonction $u(x)$ doit satisfaire les conditions aux limites, $u(a) = 0$ et $u(b) = 0$, posées aux bornes de l'intervalle $[a,b]$. Afin de résoudre numériquement (trouver la solution approchée) le système (35), nous recourrons à la méthode des différences finies. Dans cette méthode numérique, l'intervalle $[a,b]$ sur lequel nous cherchons la solution $u(x)$ est discrétisé en $n+1$ sous-intervalles équidistants de longueur h avec $x_i = x_0 + i\,h$ et $i = 1, 2, 3, ..., n$. On cherche alors en chacun de ces points une valeur approchée, notée u_i, de $u(xi)$. Ainsi, le système continu initial est substitué par un système discret. L'idée de base de la méthode des différences finies consiste à remplacer l'équation différentielle (35) par un système de n équations algébriques. Ce système d'équations est obtenu en écrivant cette équation différentielle en chaque point de discrétisation x_i, et en substituant également à chaque valeur $u^{''}(x)$ l'approximation de la dérivée seconde :

$$u^{''}(x) \approx \frac{u(x_{i-1}) - 2\,u(x_i) + u(x_{i+1})}{h^2} + O(h^2) \tag{36}$$

Et la dérivée $u^{'}(x)$ est approchée par

$$u^{'}(x) \approx \frac{u(x_{i+1}) - u(x_{i-1})}{2\,h} + O(h) \tag{37}$$

Ainsi, l'équation différentielle (35) est réécrite suivant :

$$\begin{cases} -\dfrac{u(x_{i-1}) - 2\,u(x_i) + u(x_{i+1})}{h^2} + c(x_i)\,\dfrac{u(x_{i+1}) - u(x_{i-1})}{2\,h} = f(x_i) \quad i \in \{1, ..., n\} \\ u_0 = 0 \quad \text{et} \quad u_n = 0 \end{cases} \tag{38}$$

Comme précédemment en adoptant une notation matricielle après quelques réarrangements, le problème peut s'écrire :

$$A_h\,u_h = b_h \tag{39}$$

Avec $A_h = A_1 + A_2$,

$$A_1 = \begin{pmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{pmatrix} \tag{40}$$

$$A_2 = \frac{r\,h}{2}\begin{pmatrix} c(x_1) & 0 & \dots & 0 \\ 0 & c(x_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & c(x_n) \end{pmatrix} \times \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ -1 & 0 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 0 & 1 \\ 0 & \dots & 0 & -1 & 0 \end{pmatrix} \tag{41}$$

$$f_h = h^2 \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{n-1}) \\ f(x_n) \end{pmatrix} \tag{42}$$

Le vecteur des valeurs de la solution (inconnues) aux points x_i est donné

$$u_h = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{pmatrix} \in \mathbb{R}^n \tag{43}$$

On peut mettre en évidence le fait que la matrice A_h est inversible (sous l'hypothèse que la fonction $c(x) \geq 0$. La matrice A_h est symétrique définie positive). Les matrices A_1 et A_2 sont tridiagonales. Afin d'obtenir la solution discrète, les valeurs du vecteur u_h, il va falloir résoudre le système linéaire tridiagonal (39).

Exercice ❹ ☞ Ⓡ

Soit l'équation de *convection-diffusion* :

$$\begin{cases} -u^{''}(x) + r\,x\,u^{'}(x) = f(x) \\ \\ x \in [0,1] \quad \text{et} \quad u(0) = 0, \quad u(1) = 0 \end{cases} \tag{44}$$

Le second terme de l'équation vaut :

$$f(x) = \frac{(r^2\,e^{(r\,x)}\,(x-1))}{(1-e^r) + r\,x} \tag{45}$$

La solution exacte est donnée par :

$$\text{funex} = \frac{x - (1 - e^{(r\,x)})}{(1-e^r)} \tag{46}$$

1) Résoudre numériquement l'équation (44) par la méthode des différences finies.
2) Tracer, sur la même figure, les solutions exacte et numérique pour `n=64` et `r=1/2`.
3) Étudier l'erreur en fonction du nombre de sous-intervalles de discrétisation et du paramètre `r`.

4) Représenter graphiquement cette erreur en fonction de `n+1` et des valeurs du paramètre `r`.

Voici le script Matlab®

```
clear all ; clc ; close all ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 19/11/2018 Samir KENOUCHE : ALGORITHME PERMETTANT
% L'IMPLEMENTATION, SOUS MATLAB, DE LA METHODE DES DIFFERENCES FINIES EN
% DIMENSION 1

fx = @(x,r) (r.^2.*exp(r.*x).*(x-1))./(1 - exp(r)) + r.*x ;
n = 64 ; r = 1/2 ; h = 1/(n+1) ; xh = 0 : h: 1 ;
cx = inline('x') ; funex = @(x,r) x-(1-exp(r.*x))./(1- exp(r)) ;

fn = h.^2.*fx(xh(1:n),r) ; cx = cx(xh(1:n)).*r ;

sur_diag = diag(ones(n-1 ,1) ,1) ; sur_diag(sur_diag == 1) = -1 ;
des_diag = diag(ones(n-1 ,1) , -1) ; des_diag(des_diag == 1) = -1 ;
in_diag = diag(ones(n ,1)) ; in_diag(in_diag == 1) = 2 ;

A1 = sur_diag + des_diag + in_diag ; a1 = diag(ones(n ,1)) ;
a1(a1 == 1) = cx ;

sur_diag_a2 = diag(ones(n-1 ,1) ,1) ; sur_diag_a2(sur_diag_a2 == 1) = 1 ;
des_diag_a2 = diag(ones(n-1 ,1) , -1) ; des_diag_a2(des_diag_a2 == 1) = -1 ;
in_diag_a2 = diag(ones(n ,1)) ; in_diag_a2(in_diag_a2 == 1) = 0 ;

a2 = sur_diag_a2 + des_diag_a2 + in_diag_a2 ;
A2 = (r*h/2).*(a1*a2) ; An = A1 + A2 ;

un = fn*inv(An) ;  % RESOLUTION DU SYSTEME LINEAIRE
uh = [0 , un, 0] ; % SOL. FINALE - PRISE EN COMPTE DES CONDITIONS INITIALES

%%%%%%%%%%%%%%%%%%%%%%%%%%%% AFFICHAGE GRAPHIQUE %%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure('color',[1 1 1])
plot(xh,uh,'o','MarkerSize',7,'LineWidth',1) ; hold on ;
xk = 0:0.001:1 ; plot(xk,funex(xk,r),'r','LineWidth',1.2)
axis([-0.1 1.1 -0.005 0.07]) ;
ih =legend('SOLUTION NUMERIQUE','SOLUTION EXACTE') ;
set(ih,'Interpreter','none','Location','South','Box','on',...
    'Color','none') ; xlabel('x','FontSize',12) ; ylabel('u(x)','FontSize',12)

msg1 = strcat('r= ', num2str(r)) ;
gtext(msg1) % cliquer sur la figure pour afficher : msg1
msg2 = strcat('n= ', num2str(n)) ;
gtext(msg2) % cliquer sur la figure pour afficher : msg2
```

En déroulant le script ci-dessus pour `n = 4`. Les résultats s'affichent sur la fenêtre des commandes, suivant :

```
A1 =

     2    -1     0     0
    -1     2    -1     0
     0    -1     2    -1
     0     0    -1     2

A2 =

         0         0         0         0
   -0.0050         0    0.0050         0
         0   -0.0100         0    0.0100
         0         0   -0.0150         0

An =

    2.0000   -1.0000         0         0
   -1.0050    2.0000   -0.9950         0
         0   -1.0100    2.0000   -0.9900
         0         0   -1.0150    2.0000

un =

    0.0353    0.0548    0.0562    0.0380

uh =

    0    0.0353    0.0548    0.0562    0.0380    0
```

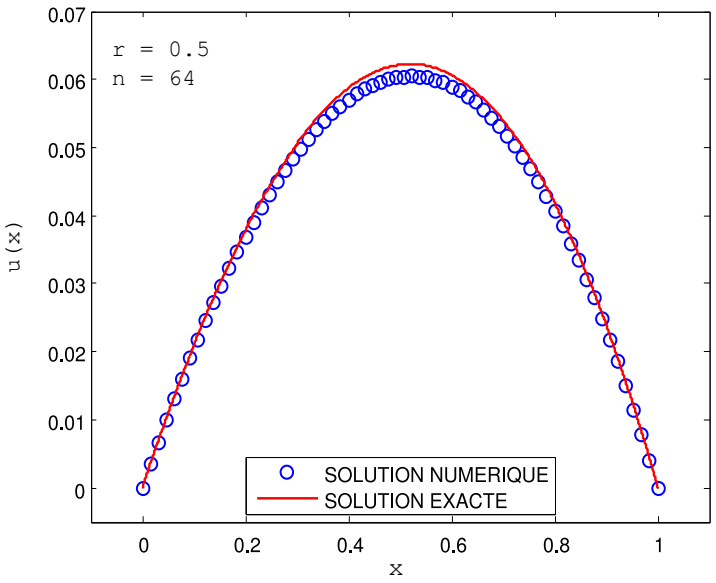

FIGURE 7: Solution exacte et solution numérique obtenues par différences finies

```
clear all ; clc ; close all ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  18/11/2018 Samir KENOUCHE : ALGORITHME PERMETTANT
% L'ANALYSE DE L'ERREUR EN FONCTION DU NOMBRES DE SOUS-INTERVALLES DE
% DISCRETISATION
fx = @(x,r) (r.^2.*exp(r.*x).*(x-1))./(1 - exp(r)) + r.*x ;  r = 1/2 ;
cx1 = inline('x') ; funex = @(x,r) x-(1-exp(r.*x))./(1- exp(r)) ;
n = 5 : 5 : 120 ;

for ik = 1:length(n)

  h = 1/(n(ik)+1) ; xh = 0:h: 1 ;
  fn = h.^2.*fx(xh(1:n(ik)),r) ; cx = cx1(xh(1:n(ik))).*r ;

 sur_diag = diag(ones(n(ik)-1 ,1) ,1) ; sur_diag(sur_diag == 1) = -1 ;
 des_diag = diag(ones(n(ik)-1 ,1) , -1) ; des_diag(des_diag == 1) = -1 ;
 in_diag = diag(ones(n(ik), 1)) ; in_diag(in_diag == 1) = 2 ;

 A1 = sur_diag + des_diag + in_diag ; a1 = diag(ones(n(ik) ,1)) ;
 a1(a1 == 1) = cx ;

 sur_diag_a2 = diag(ones(n(ik)-1 ,1) ,1) ; sur_diag_a2(sur_diag_a2 == 1) = 1 ;
 des_diag_a2 = diag(ones(n(ik)-1 ,1) , -1) ;
 des_diag_a2(des_diag_a2 == 1) = -1 ;
 in_diag_a2 = diag(ones(n(ik) ,1)) ; in_diag_a2(in_diag_a2 == 1) = 0 ;

 a2 = sur_diag_a2 + des_diag_a2 + in_diag_a2 ;
 A2 = (r*h/2).*(a1*a2) ; An = A1 + A2 ;

 un = fn*inv(An) ; % RESOLUTION DU SYSTEME LINEAIRE
 uh = [0 , un, 0] ; % SOL. FINALE - PRISE EN COMPTE DES CONDITIONS INITIALES

 err(ik) = max(abs(uh - funex(xh,r))) ;

end

figure('color',[1 1 1]) ; hold on ; box on ;
loglog(n+1,err,'o','MarkerSize',7,'LineWidth',1) ; % ECHELLE LOGARITHMIQUE
xlabel('n + 1','FontSize',12) ; ylabel('Erreur absolue','FontSize',12) ;
```

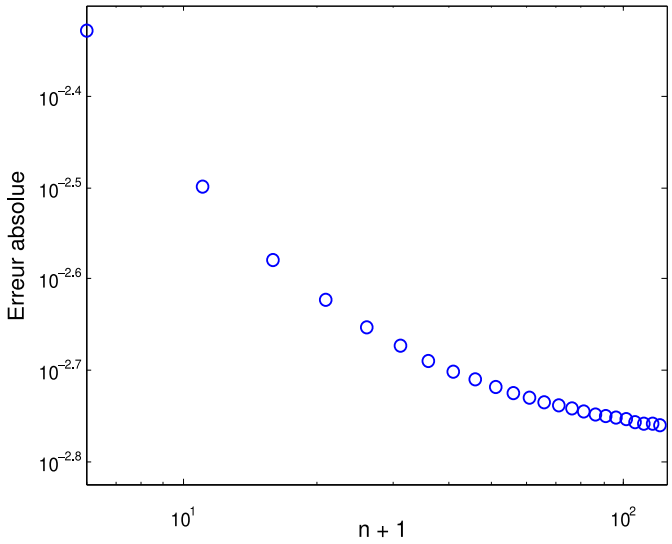

FIGURE 8: Erreur absolue versus le nombre d'intervalles de discrétisation.


```
clear all ; clc ; close all ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  18/11/2018 Samir KENOUCHE : ALGORITHME PERMETTANT
% L'ANALYSE DE L'ERREUR EN FONCTION DU NOMBRES DE SOUS-INTERVALLES DE
% DISCRETISATION
fx = @(x,r) (r.^2.*exp(r.*x).*(x-1))./(1 - exp(r)) + r.*x ;
cx1 = inline('x') ; funex = @(x,r) x-(1-exp(r.*x))./(1- exp(r)) ;
n = 64 ;

for r = 1:40

  h = 1/(n + 1) ; xh = 0:h: 1 ;
  fn = (h.^2).*fx(xh(1:n),r) ; cx = cx1(xh(1:n)).*r ;

 sur_diag = diag(ones(n - 1 ,1) ,1) ; sur_diag(sur_diag == 1) = -1 ;
 des_diag = diag(ones(n - 1 ,1) , -1) ; des_diag(des_diag == 1) = -1 ;
 in_diag = diag(ones(n, 1)) ; in_diag(in_diag == 1) = 2 ;

 A1 = sur_diag + des_diag + in_diag ; a1 = diag(ones(n, 1)) ;
 a1(a1 == 1) = cx ;

 sur_diag_a2 = diag(ones(n - 1 ,1) ,1) ; sur_diag_a2(sur_diag_a2 == 1) = 1 ;
 des_diag_a2 = diag(ones(n - 1 ,1) , -1) ;
 des_diag_a2(des_diag_a2 == 1) = -1 ;
 in_diag_a2 = diag(ones(n ,1)) ; in_diag_a2(in_diag_a2 == 1) = 0 ;

 a2 = sur_diag_a2 + des_diag_a2 + in_diag_a2 ;
 A2 = (r*h/2).*(a1*a2) ; An = A1 + A2 ;

 un = fn*inv(An) ; % RESOLUTION DU SYSTEME LINEAIRE
 uh = [0 , un, 0] ; % SOL. FINALE - PRISE EN COMPTE DES CONDITIONS INITIALES
```

```
 err_r = max(abs(uh - funex(xh, r))) ;

 figure(1) ; hold on ; box on ;
 semilogy(r,err_r,'o','MarkerSize',7,'LineWidth',1)
 % ECHELLE SEMI-LOGARITHMIQUE
 xlabel('Valeur de r','FontSize',12) ;
 ylabel('Erreur absolue','FontSize',12) ;

end
```

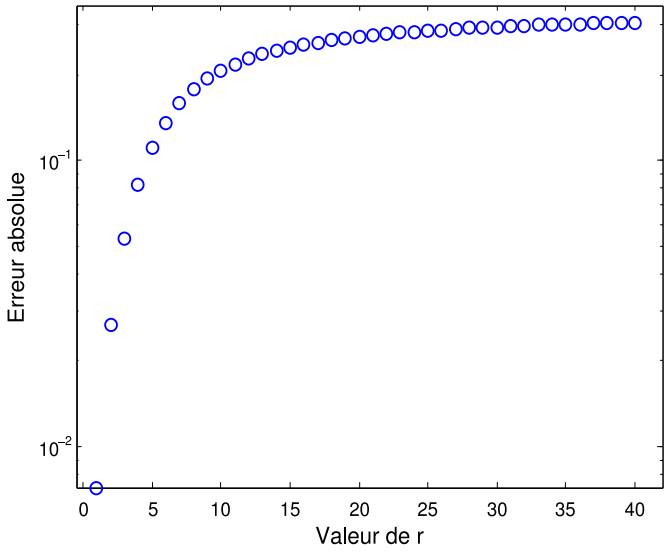

FIGURE 9: Erreur absolue en fonction du paramètre r.

Supplément : avec une démarche analogue on peut résoudre également le problème de la flexion simple dont la formulation mathématique est donnée par :

$$\begin{cases} -u''(x) + c(x)\,u(x) = f(x), \quad x \in [a,b] \\ u(a) = \alpha \quad \text{et} \quad u(b) = \beta \end{cases} \tag{47}$$

En utilisant la formule des différences finies centrées le problème devient :

$$\begin{cases} -\dfrac{u(x_{i-1}) - 2\,u(x_i) + u(x_{i+1})}{h^2} + c(x_i)\,u_i = f(x_i) \quad i \in \{1,...,n\} \\ u_0 = \alpha \quad \text{et} \quad u_n = \beta \end{cases} \tag{48}$$

En adoptant une notation matricielle :

$$A_h\,u_h = b_h \tag{49}$$

Avec,

$$A_h = A_h^{(0)} + \begin{pmatrix} c(x_1) & 0 & \dots & 0 \\ 0 & c(x_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & c(x_n) \end{pmatrix} \tag{50}$$

$$A_h^{(0)} = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{pmatrix} \tag{51}$$

Les deux matrices ci-dessus peuvent se combiner pour donner :

$$A_h = \frac{1}{h^2}\begin{pmatrix} 2+c(x_1)\,h^2 & -1 & 0 & \dots & 0 \\ -1 & 2+c(x_2)\,h^2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2+c(x_{n-1})\,h^2 & -1 \\ 0 & \dots & 0 & -1 & 2+c(x_n)\,h^2 \end{pmatrix} \tag{52}$$

$$b_h = \begin{pmatrix} f(x_1)+\alpha\,h^{-2} \\ f(x_2) \\ \vdots \\ f(x_{n-1}) \\ f(x_n)+\beta\,h^{-2} \end{pmatrix} \tag{53}$$

Le vecteur des valeurs de la solution (inconnues) aux points x_i est donné

$$u_h = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{pmatrix} \in \mathbb{R}^n \tag{54}$$

1) Problème non-linéaire: Considérons le problème non-linéaire suivant :

$$\begin{cases} -u''(x) + x\,u(x)^3 = f(x), \quad 0 < x < L \\ \\ u(0) = \alpha \\ u(L) = \beta \end{cases} \tag{55}$$

Contrairement au cas linéaire (problème (24)) où le terme $x\,u(x)^3$ n'existe pas, dans le problème (55) nous avons une relation non-linéaire entre la source de la déformation $f(x)$ et l'amplitude de la déformation $u(x)$. Autrement dit, si j'applique par exemple une force $f(x)$ deux fois plus grande, l'amplitude de la déformation $u(x)$ n'est pas doublée. En appliquant la formule des différences finies centrées il vient :

$$\begin{cases} -\dfrac{u_{i-1} - 2\,u_i + u_{i+1}}{h^2} + x_i\,u_i^3 = f(x_i) \quad i \in \{1, ..., n\} \\ u_0 = \alpha \quad \text{et} \quad u_{n+1} = \beta \end{cases} \tag{56}$$

Contrairement au problème (24), ici nous cherchons à résoudre un système de n équations non-linaires. Il existe dans la littérature plusieurs méthodes itératives pour effectuer ce calcul. On citera les méthodes de Newton, de fausse position ou de la sécante. Je renvoie, les lecteurs intéressés par ces méthodes, à mon cours d'analyse numérique que je dispense aux deuxièmes années des filières physique et chimie. Ce cours est disponible en version pdf.

Pour n = 4, on obtient le système d'équations :

$$\begin{cases} \dfrac{-u_0+2\,u_1-u_2}{h^2}+x_1\,u_1^3=f(x_1), & i=1\\ \dfrac{-u_1+2\,u_2-u_3}{h^2}+x_2\,u_2^3=f(x_2), & i=2\\ \dfrac{-u_2+2\,u_3-u_4}{h^2}+x_3\,u_3^3=f(x_3), & i=3\\ \dfrac{-u_3+2\,u_4-u_5}{h^2}+x_4\,u_4^3=f(x_4), & i=4 \end{cases} \Rightarrow \begin{cases} -\alpha+2\,u_1-u_2+h^2\,x_1\,u_1^3-h^2\,f(x_1)=0\\ -u_1+2\,u_2-u_3+h^2\,x_2\,u_2^3-h^2\,f(x_2)=0\\ -u_2+2\,u_3-u_4+h^2\,x_3\,u_3^3-h^2\,f(x_3)=0\\ -u_3+2\,u_4-\beta+h^2\,x_4\,u_4^3-h^2\,f(x_4)=0 \end{cases}$$

En posant $u_0=0$ et $u_{n+1}=0$ (pas de déformations aux extrémités) on obtient :

$$\begin{cases} 2\,u_1-u_2+h^2\,x_1\,u_1^3-h^2\,f(x_1)=0\\ -u_1+2\,u_2-u_3+h^2\,x_2\,u_2^3-h^2\,f(x_2)=0\\ -u_2+2\,u_3-u_4+h^2\,x_3\,u_3^3-h^2\,f(x_3)=0\\ -u_3+2\,u_4+h^2\,x_4\,u_4^3-h^2\,f(x_4)=0 \end{cases}$$

On peut par exemple résoudre ce système par la méthode de Newton, c'est une méthode itérative d'ordre deux donc elle converge rapidement. Nous allons illustrer cette méthode à l'aide d'un exemple. Soit à résoudre le système d'équations non-linéaires suivant :

$$f(U)=\begin{cases} f_1(u_1,u_2)=2\,u_1-u_2+e^{u_1}=0\\ f_2(u_1,u_2)=-u_1+2\,u_2+e^{u_2}=0 \end{cases} \tag{57}$$

Nous cherchons u_1 et u_2 telle que $f(U)=0$, avec le vecteur $U=(u_1,u_2)^T$. Le schéma numérique de la méthode de Newton pour résoudre le système (57) est :

$$U_{k+1}=U_k-\frac{f(U)}{\nabla f(U)}$$
$$[u_1^{k+1},u_2^{k+1}]=[u_1^k,u_2^k]-\frac{f(u_1,u_2)}{\nabla f(u_1,u_2)}$$

Avec $\nabla f(u_1,u_2)$ est la matrice Jacobienne :

$$\nabla f(u_1,u_2)=\begin{pmatrix} \dfrac{\partial f_1}{\partial u_1} & \dfrac{\partial f_1}{\partial u_2}\\ \dfrac{\partial f_2}{\partial u_1} & \dfrac{\partial f_2}{\partial u_2} \end{pmatrix} \tag{58}$$

On donne une valeur initiale au vecteur U_k ensuite on calcule les successeurs U_{k+1}. On cesse les itérations une fois le test d'arrêt est positif pour une tolérance donnée, par exemple $|-\frac{f(U)}{\nabla f(U)}|<\epsilon$.

B. En dimension 2

Il existe une myriade de problèmes de physique et de chimie admettant comme formulation mathématique une équation aux dérivées partielles. Cette dernière est une équation dont l'inconnue est une fonction de plusieurs variables. Dans l'équation même apparait la fonction de plusieurs variables recherchée ainsi que ses dérivées partielles. Soit $f : (x,t) \in [0,1] \times \mathbb{R}^+ \longmapsto f(x,t) \in \mathbb{R}$ une fonction continue, nous considérons un problème parabolique consistant à déterminer $u : (x,t) \in [0,1] \times \mathbb{R}^+ \longmapsto u(x,t) \in \mathbb{R}$ qui satisfait :

$$\frac{\partial u(x,t)}{\partial t} - \frac{\partial^2 u(x,t)}{\partial x^2} = \rho(x,t) \tag{59}$$

$$\begin{cases} u(x,0) = u_0(x) \quad 0 \leq x \leq 1 \\ u(0,t) = u(L,t) = 0 \quad 0 \leq t \end{cases} \tag{60}$$

C'est l'équation de la diffusion de la chaleur, avec $\rho(x,t)$ est la source de chaleur. On cherche à déterminer la quantité de chaleur fournie au point x à l'instant t. A partir d'un développement de Taylor on démontre ces approximations des dérivées partielles :

$$\frac{\partial u(x,t)}{\partial t} \approx \frac{u(x_i,t_{j+1}) - u(x_i,t_j)}{\Delta t} + O(\Delta t) \tag{61}$$

$$\frac{\partial u(x,t)}{\partial t} \approx \frac{u(x_i,t_j) - u(x_i,t_{j-1})}{\Delta t} - O(\Delta t) \tag{62}$$

$$\frac{\partial^2 u(x,t)}{\partial x^2} \approx \frac{u(x_{i-1},t_j) - 2\,u(x_i,t_j) + u(x_{i+1},t_j)}{h^2} + O(h^2) \tag{63}$$

Nous utiliserons ces schémas des différences finies afin d'approcher $u(x,t)$ du problème ci-dessus. En effet, à partir des équations (62), (63) et en posant $u(x_i,t_j) = u(i,j)$ on obtient :

$$\frac{u(i,j) - u(i,j-1)}{\Delta t} - \frac{u(i-1,j) - 2u(i,j) + u(i+1,j)}{h^2} = f(i,j)$$

$$h^2\,u(i,j) - h^2\,u(i,j-1) - \Delta t\,u(i-1,j) + 2\,\Delta t\,u(i,j) - \Delta t\,u(i+1,j) = f(i,j)\,\Delta t\,h^2$$

Tenant compte des conditions $u(0,t) = u(L,t) = 0$ on obtient :

$$h^2\,u(i,j) - h^2\,u(i,j-1) - \cancel{\Delta t\,u(i-1,j)} + 2\,\Delta t\,u(i,j) - \cancel{\Delta t\,u(i+1,j)} = f(i,j)\,\Delta t\,h^2$$

$$(h^2 + 2\,\Delta t)\,u(i,j) = h^2\,u(i,j-1) + f(i,j)\,\Delta t\,h^2$$

$$(I + \frac{2\,\Delta t}{h^2})\,u(i,j) = u(i,j-1) + f(i,j)\,\Delta t$$

$$(I + \frac{A_n\,\Delta t}{h^2})\,u(i,j) = u(i,j-1) + f(i,j)\,\Delta t$$

Tenant compte de la condition $u(x,0)$ le schéma numérique final devient :

$$(I + \frac{A_n\,\Delta t}{h^2})\,u(i,j+1) = u(i,j) + f(i,j)\,\Delta t$$

Avec I est la matrice identité et A_n est une matrice tridiagonale valant :

$$A_n = \begin{pmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{pmatrix} \in \mathbb{R}^n \times \mathbb{R}^n$$

Voici le script Matlab®


```
clear all ; clc ; close all ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% LE 02/12/2018 Samir KENOUCHE : RESOLUTION DE L'EQUATION DE LA CHALEUR PAR
% DIFFERENCES FINIES EN DIMENSION 2

nb = 10 ; pas_temps = 0.01 ; pas_x = 1/(nb+1) ; T = 0.7 ;
xi = 0: pas_x :1 ; ti = 0: pas_temps :T ; ux = sin(pi.*xi) ;

u = zeros([numel(xi)-2 numel(ti)]) ; u(:, 1) = ux(2:end-1)' ;
fx = -2 + 6.*xi ; fn = zeros(size(u)) ;

mat_diag = 2*diag(ones(nb,1))-diag(ones(nb-1 ,1),1)-diag(ones(nb-1,1),-1) ;
my_mat = (eye(nb)+(pas_temps/pas_x.^2).*mat_diag) ; it = 1 ;

while  it < numel(ti)

     fn(:, it) = fx(2:end-1)' ;
     u(:, it+1) = my_mat \(u(:,it) + pas_temps*fn(:,it)) ; % DIVISION A GAUCHE
     it = it + 1 ;
end

ui = cat(1,zeros([1 numel(ti)]),u,zeros([1 numel(ti)])); % AJOUT DES CONDITION
                                                         % INITIALE ET AU LIMITE
fig1 = figure('color',[1 1 1]) ; [xn, tn] = meshgrid(xi, ti) ;
surf(xn, tn, ui') ; xlabel('Distance') ; ylabel('Temps') ; view(114, 18) ;

fig2 = figure('color',[1 1 1]) ; contourf(xn, tn, ui') ;
xlabel('Distance') ; ylabel('Temps') ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%% PARTIE : ANIMATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure('Renderer','zbuffer') ; [xn, tn] = meshgrid(xi, ti) ;
colormap(jet) ; xlabel('Distance') ; ylabel('Temps') ;

for in = 1:30
    surf(exp(-0.008*in)*ui'.^2,ui')
    my_animation(in) = getframe ;
end
```

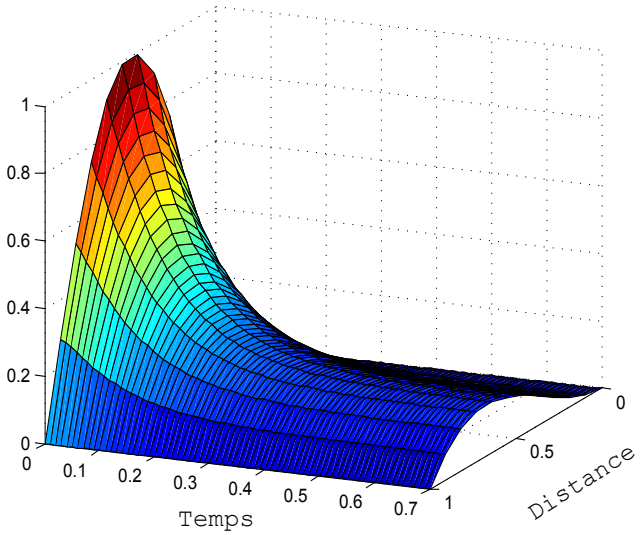

FIGURE 10: Surface de la solution approchée

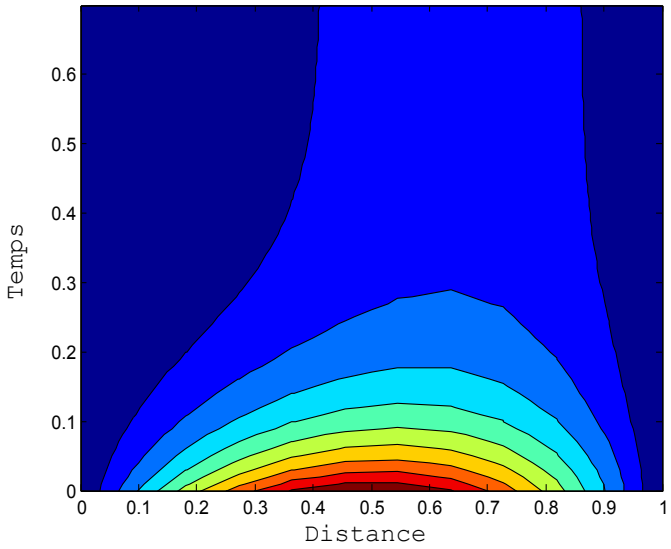

FIGURE 11: Courbe de niveaux de la solution approchée