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

Cet exercice permet de mettre en application les notions de base de Matlab: (fonctions, graphiques, structures de contrôle,...). Une solution à l'exercice est fournie dans le Thème 2.

EDO: Mise en oeuvre de schémas à un pas implicites: estimation d'ordre et comparaisons

Thème - 1 *Sujet*

Exercice-1 **:** **Modèles**

On se propose dans cette fiche de mettre en oeuvre un schéma à un pas implicite. Sans nuire à la généralité, nous allons considérer les schémas dits θ-schémas.

L'équation différentielle ordinaire que nous considérons s'écrit sous la forme suivante :

$$\begin{cases} X'(t) = F(t, X(t)), \quad t \in]t_0, t_0 + T[. \\ X(t_0) = X^0 \in \mathbb{R}^d. \end{cases} \tag{1}$$

où F est une fonction donnée dans $C^1(\mathbb{R} \times \mathbb{R}^d, \mathbb{R}^d)$, $X^0 \in \mathbb{R}^d$ et où T est strictement positif et $t_0 \in \mathbb{R}$. On suppose que l'on dispose des dérivées partielles de $Y \mapsto F(t, Y)$.

Afin de résoudre numérique ce problème, on se donne $N \in \mathbb{N}^*$ et on pose $h = \frac{T}{N}$ le pas de discrétisation ce qui définit les instants $t_n = t_0 + nh, n = 0, \ldots, N$. On construit alors la suite X_n des valeurs approchées de $X(t_n)$ par le schéma :

$$\begin{cases} X_{n+1} &= X_n + h\big((1-\theta)F(t_n, X_n) + \theta F(t_{n+1}, X_{n+1})\big), \ 0 \leq n \leq N-1 \\ X_0 &= X^0, \end{cases} \tag{2}$$

appelé θ-schéma, où $\theta \in [0, 1]$

Q-1-1 **:** Montrer qu'à chaque itération n la détermination de X_{n+1} se ramène à la résolution d'un problème non linéaire que l'on peut écrire sous les deux formes suivantes :

- **Forme 1**

$$\mathcal{N}(X_{n+1}, F, X_n, t_n, h, \theta) = 0. \tag{3}$$

- **Forme 2**

$$X_{n+1} = \mathcal{P}(X_{n+1}, F, X_n, t_n, h, \theta). \tag{4}$$

Q-1-2 **:** Exprimer la matrice jacobienne de l'application $Z \mapsto \mathcal{N}(Z, F, X_n, t_n, h, \theta)$ en fonction des dérivées partielles de F par rapport à sa second variable.

Q-1-3 **:** Décrire l'algorithme de Newton pour déterminer la solution X_{n+1} de (11). L'algorithme sera initialisée par X_n (solution à l'instant t_n), utilisera un critère d'arrêt défini par un paramètre tol($ou\ \varepsilon$) (voir Cours M315) et imposera un nombre maximum `IterMax` d'itérations. On n'oubliera pas de vérifier que les matrices jacobiennes utilisées sont inversibles. (*On pour à ce stade passer à la question* **Q.2.1** *pour sa mise en oeuvre*).

Q-1-4 : Décrire la méthode de point fixe pour déterminer la solution X_{n+1} de (12). L'algorithme sera initialisée par X_n, et utilisera un critère d'arrêt défini par un paramètre tol($ou\ \varepsilon$) (identique à celui de Newton) et imposera un nombre maximum `IterMax` d'itérations. (*On pour à ce stade passer à la question* **Q.2.2** *pour sa mise en oeuvre*).

Exercice-2 : Mise en oeuvre

Pour une mise en oeuvre efficace du schéma numérique on adopte les conventions suivantes :

- L'EDO à résoudre est définie à travers une fonction `Matlab` de prototype

```
function [F, DF] = monEDO (t,y)
% F  est  la valeur de la fonction second-membre f  au point (t,y)
% DF est la dériée partielle (matrice jacobienne) par rapport a y de la f au point (t,y) (i.e df/dy )
```

- Un schéma numérique à un pas est défini par la donnée d'une fonction qui explique comment avancer d'un pas . Elle a pour prototype :

```
function [t1, x1, h1] = monSchema(f,t0,x0,h0)
%% ENTREES :
% f    : fonction second-membre f i.e  l' edo à résoudre (voir listing precedent : monEDO)
% t0   : est l'instant présent    (tn)
% x0   : la solution à l'instant present (xn)
% h    : le pas de discrétisation
%% SORTIES :
% t1   : l'instant suivant  (tn+1)
% x1   : la solution à l'instant suivant (xn+1)
% h    : le prochain pas à utiliser
```

Ainsi par exemple, le schéma d'Euler explicite s'écrira :

```
function [t1, x1, h1] = monSchemaEulerExplicite(f,t0,x0,h0)
t1 = t0 + h0;
x1 = x0 + h0 * f(t0,x0);
h1 = h0 ;
end
```

- La résolution effective de l'EDO se fait à travers la fonction suivante, qui fait appel au schéma numérique :

```
function [t,x,h]= solveurEDO(f,t0,tf,x0,N,monSchema)
%% ENTREES :
% f    : fonction second-membre f EDO à résoudre (voir listing precedent : monEDO)
% t0   : est l'instant initial
% tf   : instant final
% x0   : la solution initiale
% N    : le nombre de subdivisions à utiliser
%% SORTIES :
% t    : la suite des instants (tn)
% x    : la suite des solutions (xn) aux instants (tn)
% h    : le pas h de discretisation utilisé
t_temp = linspace(t0,tf,N+1);
h = t_temp(2) - t_temp(1);
t_temp = [];
x = [x0];
t = [t0];
for k = 1:N
%calcul
 [t1,x1,hh] = monSchema(f,t(k),x(:,k),h);
%stockage
  x = [x,x1];
  t = [t,t1];
end
end
```

Ainsi, on pourra résoudre une EDO par le schéma d'Euler explicite en procédant ainsi :

```
[t,x,h] = solveurEDO(f,t0,tf,x0,N,@monSchemaEulerExplicite)
```

Q-2-1 : Ecrire à présent une fonction `Matlab`

```
function [t1, x1, h1] = monThetaSchemaNewton(f, t0, x0, h0, tol, IterMax, theta)
```

qui avance le θ-schéma d'un pas en utilisant un algorithme de Newton pour déterminer la solution du problème non linéaire en X_{n+1} dans le θ-schéma.

Q-2-2 : Ecrire à présent une fonction Matlab

```
function [t1, x1, h1] = monThetaSchemaPicard(f, t0, x0, h0, tol, IterMax, theta)
```

qui avance le θ-schéma d'un pas en utilisant la méthode de point fixe pour déterminer la solution du problème non linéaire en X_{n+1} dans le θ-schéma.

Exercice-3 : Validations : Evaluation de l'ordre

On considère pour la validation le système d'EDOs suivant :

$$\left\{ \begin{array}{rcl} x' & = & (x^2+y^2)y \\ y' & = & -(x^2+y^2)x. \end{array} \right. \tag{5}$$

Avec $t_0 = 0, T = \pi, (x(0), y(0)) = (1, 0)$

Q-3-1 : Vérifier que la solution exacte du problème est $(x(t), y(t)) = (cos(t), -sin(t))$.

Dans toute la suite on désignera par $e_N^\theta = \|(-1, 0) - X_N\|$ l'erreur commise au dernier point de calcul par le θ-schéma, avec N subdivisions.

Q-3-2 : Dans le cas où la méthode de Newton est utilisée avec `IterMax` $= 10$ et $tol = 1e-3$, tracer sur un même graphique la solution exacte et la solution fournie par le θ-Schéma lorsque $N = 50$, et $\theta \in \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$.

Q-3-3 : Reprendre la question précédente lorsque la méthode du point fixe est utilisée avec les mêmes paramètres, que la méthode de Newton.

Q-3-4 : On considère ici le θ-schéma avec la méthode de Newton où l'on a imposé: `IterMax` $= 10$ et $tol = 1e-3$. Tracer en échelle logarithmique e_N^θ en fonction de h, pour $\theta = 0$, $\theta = 0.5$, $\theta = 1$ et $N = 2^k, 1 \leq k \leq 12$. Que peut-on déduire sur l'ordre de précision des θ-schémas.

Q-3-5 : On considère à présent le θ-schéma avec la méthode du point fixe, où l' on a décidé de n'effectuer qu'une seule itération dans la méthode du point fixe. Tracer en échelle logarithmique e_N^θ en fonction de h, pour $\theta = 0$, $\theta = 0.5$, $\theta = 1$ et $N = 2^k, 1 \leq k \leq 12$. Que constatez-vous ?

Exercice-4 : Validations : Comparaisons

Le problème modèle est encore celui de l'exercice précédent. On suppose cette fois $\theta = 0.5$, dans ce cas le θ-schéma est connu sous le nom de schéma de Crank-Nicolson.

Comparer numériquement les trois schémas suivants ((*on pourra estimer leur ordre de convergence*)):

1. **Schéma 1** : Le Schéma de Crank-Nicolson dans lequel la méthode de Newton est utilisée pour résoudre les problèmes non-linéaires, avec les paramètres `IterMax = 10`, et `tol = 1e-3`.
2. **Schéma 2** :

$$\left\{ \begin{array}{rcl} X_{n+1}^* & = & X_n + hF(t_n, X_n), \\ X_{n+1} & = & X_n + \frac{h}{2}\left[F(t_n, X_n) + F(t_{n+1}, X_{n+1}^*)\right],\ 0 \leq n \leq N-1 \\ X_0 & = & X^0. \end{array} \right. \tag{6}$$

3. **Schéma 3** :

$$\left\{ \begin{array}{rcl} X_{n+1}^* & = & X_n + \frac{h}{2}\left[F(t_n, X_n) + F(t_{n+1}, X_n)\right], \\ X_{n+1} & = & X_n + \frac{h}{2}\left[F(t_n, X_n) + F(t_{n+1}, X_{n+1}^*)\right],\ 0 \leq n \leq N-1 \\ X_0 & = & X^0. \end{array} \right. \tag{7}$$

4. **Schéma 4** :

$$\left\{\begin{array}{rcl} X^*_{n+1} & = & X_n + hF(t_n + \frac{h}{2}, X_n + \frac{h}{2}F(t_n, X_n)), \\ X_{n+1} & = & X_n + \frac{h}{2}\left[F(t_n, X_n) + F(t_{n+1}, X^*_{n+1})\right],\ 0 \leq n \leq N-1 \\ X_0 & = & X^0. \end{array}\right. \tag{8}$$

Thème - 2 *Elément de Réponse*

Exercice-1 **:** **Modèles**

On se propose dans cette fiche de mettre en oeuvre un schéma à un pas implicite, pour la résolution d'une équation différentielle ordinaire de la forme

$$\begin{cases} X'(t) = F(t, X(t)), \quad t \in]t_0, t_0 + T[. \\ X(t_0) = X^0 \in \mathbb{R}^d. \end{cases} \tag{9}$$

où F est une fonction donnée dans $C^1(\mathbb{R} \times \mathbb{R}^d, \mathbb{R}^d)$, $X^0 \in \mathbb{R}^d$, T est strictement positif, $t_0 \in \mathbb{R}$ et l'on suppose disposer des dérivées partielles de $Y \mapsto F(t, Y)$ $\forall t \in [t_0, t_0 + T]$.

Le schéma considéré est le θ-schéma ($\theta \in [0, 1]$) défini sur une grille uniforme de pas $h = T/N$ de $[t_0, t_0 + T]$ par :

$$\begin{cases} X_{n+1} &= X_n + h\big((1-\theta)F(t_n, X_n) + \theta F(t_{n+1}, X_{n+1})\big), \ 0 \le n \le N-1 \\ X_0 &= X^0, \end{cases} \tag{10}$$

Q-1-1 **:** On constate qu'à chaque itération n la détermination de X_{n+1} se ramène à la résolution d'un problème non linéaire que l'on peut écrire sous l' une des deux formes suivantes :

- **Forme 1 :** $\boxed{\mathcal{N}(X_{n+1}, F, X_n, t_n, h, \theta) = 0}$ avec

$$\mathcal{N}(X_{n+1}, F, X_n, t_n, h, \theta) = X_{n+1} - X_n - h\big((1-\theta)F(t_n, X_n) + \theta F(t_{n+1}, X_{n+1})\big). \tag{11}$$

- **Forme 2 :** $\boxed{X_{n+1} = \mathcal{P}(X_{n+1}, F, X_n, t_n, h, \theta)}$ avec

$$\mathcal{P}(X_{n+1}, F, X_n, t_n, h, \theta) = X_n + h\big((1-\theta)F(t_n, X_n) + \theta F(t_{n+1}, X_{n+1})\big). \tag{12}$$

Q-1-2 **:** Pour résoudre le problème en X_{n+1} dans la forme 1 par une méthode de Newton, on a besoin de la matrice jacobienne de l'application $Z \mapsto \mathcal{N}(Z, F, X_n, t_n, h, \theta)$. Elle a l'expression suivante :
$\partial_Z \mathcal{N}(Z, F, X_n, t_n, h, \theta) = I_d - h\,\theta\,\partial_Z F(t_{n+1}, Z)$, où I_d est la matrice identité de dimension d. La matrice jacobienne n'est pas nécessaire si l'on considère la forme 2 et une méthode de type point fixe.

Ainsi, sous forme de fonctions Matlab, ces deux fonctions auront les expressions suivantes :

Listing 1: fonction $\mathcal{N}$

```
function [Nc, DzNc] = Ncursif(Z,f,t0,X0,h,theta)
[F0, DF0] = f(t0,X0);
[F1, DF1] = f(t0+h,Z);
dim = length(Z);
Nc  = Z - X0 - h * ((1-theta)* F0  + theta * F1);
DzNc= eye(dim)  - h * theta * DF1;
end
```

Listing 2: fonction $\mathcal{P}$

```
function [Pc] = Pcursif(Z,f,t0,X0,h,theta)
F0  = f(t0,X0);

F1  = f(t0+h,Z);

Nc  = X0  + h * ( (1-theta)* F0  + theta * F1 );
end
```

On peut remarquer que dans cette mise en oeuvre l'évaluation de la fonction $\mathcal{N}$ retourne directement sa jacobienne, ce qui diffère de l'approche habituelle non optimale consistant à fournir séparément la fonction et sa jacobienne dans un algorithme de Newton. En effet en pratique dans un problème du type $g(x) = 0$, l'évaluation de la fonction g peut nécessiter l'acquisition des ressources (connexion à une base de données) et libération des ressources (déconnexion de cette base de données). L'acquisition des ressources étant coûteuse, il vaut mieux les réduire au maximum.

Q-1-3 **:** Un algorithme de Newton est facile à mettre en place pour déterminer X_{n+1} dans la forme 1. Nous le rappelons ici par une fonction `Newton` dans le cadre d'une équation $g(x) = 0$.

Listing 3: Code matlab pour l'algorithme de Newton

```
function [x,N] = Newton(g,x0,tol,iterMax)
x = x0;
iter = 0;
conv = false;
while(false == conv)
 [gn,Dgn] = g(x);
 delta    = Dgn\gn ;
 x = x - delta;
 iter = iter + 1;
 conv = (norm(delta)< tol * norm(x)) ||(iter > iterMax);
end
N = iter;
end
```

L'utilisation de cette fonction dans le cadre de $\mathcal{N}$, est faite dans le Listing 7.

**Q-1-4 : ** De même pour déterminer la solution X_{n+1} de (12) on met en place un algorithme de point fixe, que nous rappelons ici à travers une fonction Matlab `Picard` pour une équation générale $x = g(x)$.

Listing 4: Code matlab pour l'algorithme du point fixe

```
function [x,N] = Picard(g,x0,tol,iterMax)
x = x0;
iter = 0;
conv = false;
while(false == conv)
 xx = g(x);
 delta = xx - x;
 x = xx;
 iter = iter + 1;
 conv = (norm(delta)< tol * norm(x)) ||(iter > iterMax);
end
N = iter;
end
```

L'utilisation de cette fonction dans le cadre de $\mathcal{P}$ est faite dans le Listing 8.

Exercice-2 : ** **Mise en oeuvre

La disposition de la fonction

Listing 5: Definition du solveur générique attendant un schéma

```
function [t,x,h]= solveurEDO(f,t0,tf,x0,N,monSchema)
```

permet de simplifier considérablement la mise en oeuvre d'un schéma à un pas, en la réduisant à la définition d'une fonction de prototype

Listing 6: Interface d'un schema pour le solveur générique

```
function [t1, x1, h1] = monSchema(f,t0,x0,h0)
```

qui décrit comment avancer le schéma d'un pas.

**Q-2-1 : ** Lorsque le θ-schéma utilise un algorithme de Newton pour résoudre le problème non linéaire, la fonction d'avancement d'un pas sera extraite de la fonction suivante :

Listing 7: Definition du θ-schema avec Newton pour solveur non linéaire

```
1 function [t1, x1, h1] = monThetaSChemaNewton(f, t0, x0, h0, tol, IterMax,theta)
2 t1 = t0 + h0;
3 gPourNewton = @(Z) Ncursif(Z,f,t0,x0,h0,theta);
4 [x1,N] = Newton(gPourNewton,x0,tol,IterMax);
5 h1 = h0 ;
6 end
```

On remarque que cette fonction a beaucoup plus d'arguments que n'exige le prototype du Listing 6. Ceci ne pose pas de problème majeur car sous Matlab on peut capturer certains paramètres et ajuster le prototype des fonctions. Voir par exemple ce qui est fait dans le Listing 10 à la ligne 15.

Q-2-2 : Lorsque le θ-schéma utilise un algorithme point fixe pour résoudre le problème non linéaire, la fonction d'avancement d'un pas sera déduite de la fonction suivante :

Listing 8: Definition du θ-schema avec point fixe pour solveur non linéaire

```
function [t1, x1, h1] = monThetaSchemaPicard(f, t0, x0, h0, tol, IterMax, theta)
t1 = t0 + h0;
[F0, DF0] = f(t0, x0);
gPourPicard = @(Z) Pcursif(Z,f,t0,x0,h0,theta);
[x1,N] = Picard(gPourPicard,x0,tol,IterMax);
h1 = h0 ;
end
```

Ici aussi afin d'adapter son interface à celle du Listing 6, voir par exemple le Listing 11 ligne 15.

Exercice-3 : Validations : Evaluation de l'ordre

On considère pour la validation le système d'EDOs suivant :

$$\left\{ \begin{array}{rcl} x' & = & (x^2+y^2)y \\ y' & = & -(x^2+y^2)x. \end{array} \right. \tag{13}$$

Avec $t_0 = 0, T = \pi, (x(0), y(0)) = (1, 0)$.

La fonction Matlab implémentant le problème selon les hypothèses (disponibilité des dérivées partielles) est donnée par :

Listing 9: monEDO

```
function [f,df] = monEDO(t,x)
x1 = x(1);
x2 = x(2);
u       = x1.^2 + x2.^2;
f  = [u * x2; - u * x1 ];
df(1,:) = [ 2 * x1 * x2, (u + 2*x2.^2)];
df(2,:) = [(-u - 2*x1.^2) , -2*x2*x1];
end
```

Q-3-1 : La solution exacte du problème est $(x(t), y(t)) = (cos(t), -sin(t))$.

Q-3-2 : Pour tester le cas où la méthode de Newton est utilisée avec `IterMax` $= 10$ et $tol = 1e-3$, on utilise le listing suivant :

Listing 10: Script pour question Q-3-2

```
1  function [res]  = question32(figNum)
2  theta=[0,0.25,0.5,0.75,1];
3  col     = ['r';'*b';'+g';'m';'--k'];
4  tol     = 1e-3;
5  IterMax = 10;
6  t0      = 0;
7  tf      = pi;
8  x0      = [1;0];
9  N       = 50 ;
10
11 figure(figNum); clf; hold off;
12
13 % solutions approchees
14 for i=1:length(theta)
15  monSchema = @(f, t0, x0, h0) monThetaSChemaNewton(f, t0, x0, h0, tol, IterMax,theta(i));
16  [t,x,h]   = solveurEDO(@monEDO,t0,tf,x0,N,monSchema);
17  plot(x(1,:), x(2,:),col(i,:)); hold on;
18 end
19 % solution exacte
20 x = cos(t); y = -sin(t);
21 plot(x,y,'k');
22
```

```
legend('theta = 0','theta = 0.25','theta =0.5','theta =0.75', 'theta = 1', 'sol exacte');
title('Solution avec methode de Newton');
hold off;

% impression du graphique
print('Q32SolutionNewton.eps','-depsc');
res =1;
end
```

Le résultat de l'exécution est reporté dans la figure FIGURE 1.

**Q-3-3 : ** Pour reprendre la question précédente lorsque la méthode du point fixe est utilisée avec les mêmes paramètres, que la méthode de Newton on utilise le listing suivant :

Listing 11: Script Pour question Q-3-3

```
function [res]  = question33(figNum)
theta=[0,0.25,0.5,0.75,1];
col     = ['r';'*b';'+g';'m';'--k'];
tol     = 1e-3;
IterMax = 10;
t0      = 0;
tf      = pi;
x0      = [1;0];
N       = 50 ;

figure(figNum); clf; hold off;

% solutions approchees
for i=1:length(theta)
 monSchema = @(f, t0, x0, h0) monThetaSChemaPicard(f, t0, x0, h0, tol, IterMax,theta(i));
 [t,x,h]   = solveurEDO(@monEDO,t0,tf,x0,N,monSchema);
 plot(x(1,:), x(2,:),col(i,:)); hold on;
end

% solution exacte
x = cos(t); y = -sin(t);
plot(x,y,'k');

legend('theta = 0','theta = 0.25','theta =0.5','theta =0.75', 'theta = 1', 'sol exacte');
title('Solution avec methode de Point fixe');
hold off;

% impression du graphique
print('Q33SolutionPointFixe.eps','-depsc');
res =1;
end
```

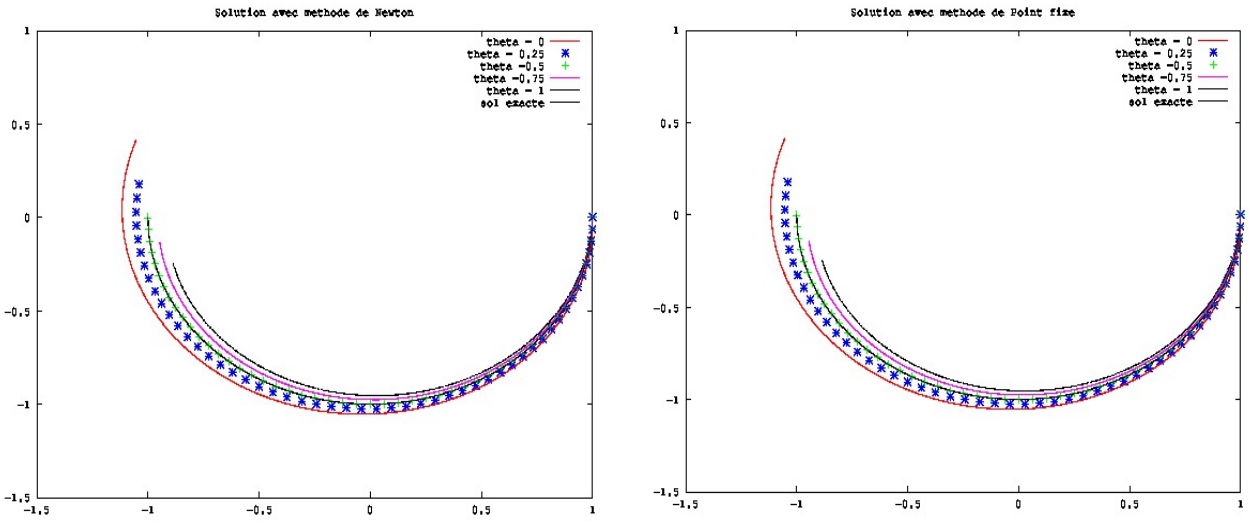

Figure 1: Graphiques pour question Q-3-2 (gauche) et question Q-3-3 (droite)

Q-3-4 : On s'intéresse à présent à l'ordre des θ-schémas. On convient d'utiliser la méthode de Newton où l'on a imposé: `IterMax` $= 10$ et $tol = 1e-3$. On peut alors représenter en échelle logarithmique $e_N^\theta = \|(-1,0) - X_N\|$ en fonction de h, pour $\theta = 0$, $\theta = 0.5$, $\theta = 1$ et $N = 2^k, 1 \leq k \leq 12$.

La fonction suivante réalise l'expérience pour une valeur de θ.

Listing 12: Script Pour question Q-3-4

```
function [res]  = question34CasUnTheta(theta, figNum)
% theta  : valeur de theta
% figNum : numero de la figure
%-------------------------------------------------------

tol       = 1e-3;
IterMax   = 10;
t0        = 0;
tf        = pi;
x0        = [1;0];
N         = 50 ;
monSchema = @(f, t0, x0, h0) monThetaSChemaNewton(f, t0, x0, h0, tol, IterMax,theta);

H = []; % pour stocker les pas
E = []; % pour stocker les erreurs

for k=1:12
  N = 2^k ;
  [t,x,h] = solveurEDO(@monEDO,t0,tf,x0,N,monSchema);
  H = [H,h];
  E = [E, norm(x(:,end) - [-1;0])];
end

% calcul des log
logH = log(H);
logE = log(E);

% droites de pente 1 et 2 passant par le dernier point
x   = linspace(min(log(H)), max(log(H)), 30);
y1  = (x - log(H(end))) + log(E(end));
y2  = 2 * (x - log(H(end))) + log(E(end));

%
figure(figNum); clf; hold off;
plot(logH,logE,'-*'); hold on;      % courbe log log
plot(x,y1,'g',x,y2,'--r');          % droites de pente 1 et 2
xlabel('Log(h)'); ylabel('Log(E)');
legend('log(E)', strcat('droite de pente = ','1'), strcat('droite de pente =', '2'));
title(strcat('Courbe pour theta = ', num2str(theta)));

% Impression du graphique au format .eps couleur
nomFichier = strcat('Q34CourbeTheta', num2str(theta),'.eps');
print(nomFichier,'-depsc');
res=1;
end
```

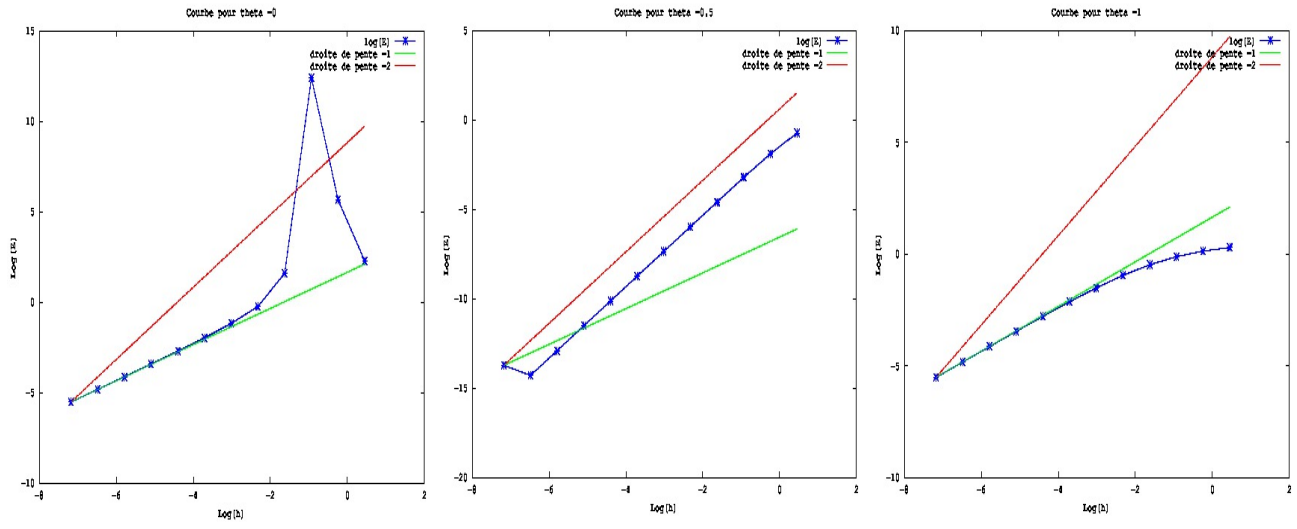

Figure 2: Graphique pour question **Q-3-4**. Erreurs en fonction du pas en échelle logarithmique. On peut observer l'ordre 1 pour les θ-schémas avec $\theta = 0$ et $\theta = 1$, et l'ordre 2 pour le θ-schéma lorsque $\theta = 0.5$

Q-3-5 : Pour réaliser une expérience analogue lorsque le θ-schéma utilise la méthode du point fixe où l'on n'effectue qu'une seule itération dans la méthode du point fixe, on fournit la fonction suivante :

Listing 13: Script Pour question Q-3-5

```
function [res]  = question35CasUnTheta(theta, figNum)
% theta  : valeur de theta
% figNum : numero de la figure
%-----------------------------------------------------

tol       = 1e-3;
IterMax   = 1;          % C'EST ICI  QU'IL Y A EU MODIFICATION
t0        = 0;
tf        = pi;
x0        = [1;0];

monSchema = @(f, t0, x0, h0) monThetaSChemaPicard(f, t0, x0, h0, tol, IterMax,theta);

H = []; % pour stocker les pas
E = []; % pour stocker les erreurs

for k=1:12
  N = 2^k ;
  [t,x,h] = solveurEDO(@monEDO,t0,tf,x0,N,monSchema);
  H = [H,h];
  E = [E, norm(x(:,end) - [-1;0])];
end

% calcul des log
logH = log(H);
logE = log(E);

% droites de pente 1 et 2 passant par le dernier point
x   = linspace(min(log(H)), max(log(H)), 30);
y1  = (x - log(H(end))) + log(E(end));
y2  = 2 * (x - log(H(end))) + log(E(end));

%
figure(figNum); clf; hold off;
plot(logH,logE,'-*'); hold on;      % courbe log log
plot(x,y1,'g',x,y2,'--r');          % droites de pente 1 et 2
xlabel('Log(h)'); ylabel('Log(E)');
legend('log(E)', strcat('droite de pente = ','1'), strcat('droite de pente =', '2'));
title(strcat('Courbe pour theta = ', num2str(theta)));

% Impression du graphique au format .eps couleur
nomFichier = strcat('Q35CourbeTheta', num2str(theta),'.eps'); % MODIFICATION FAITE ICI AUSSI
print(nomFichier,'-depsc');
res=1;
end
```

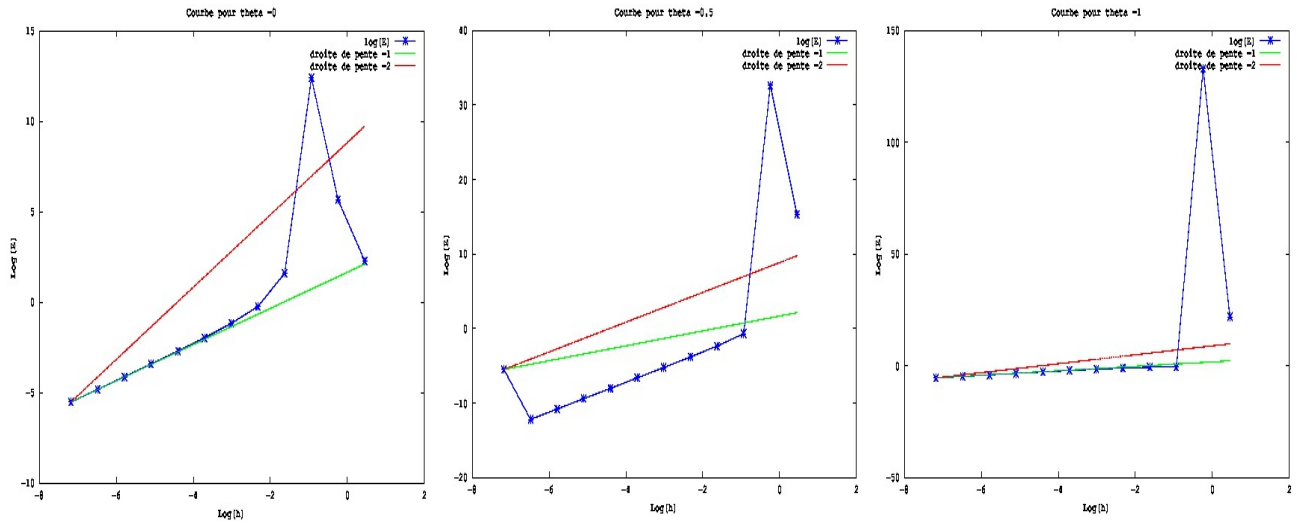

Figure 3: Graphique pour question **Q-3-5**. Erreurs en fonction du pas en échelle logarithmique. Même si les courbes sont moins explicites, on aboutit à la même conclusion qu'avec la FIGURE2.

Exercice-4 : Validations : Comparaisons

On considère à présent le schéma de Crank-Nicolson. C'est un schéma implicite d'ordre 2 comme on l'a observé ($\theta = 0.5$). On souhaite analyser le comportement de certaines méthodes d'explicitation basées sur des techniques du type prédiction-correction. On considère alors les schémas suivants (*c'est un exercice intéressant d'écrire ces schémas comme des schémas de Runge-Kutta*) :

1. **Schéma 1** : Le Schéma de Crank-Nicolson dans lequel la méthode de Newton est utilisée pour résoudre les problèmes non-linéaires, avec les paramètres `IterMax = 10`, et `tol = 1e-3`.

Listing 14: Fonction Matlab pour le Schema1

```
function [t1, x1, h1] = Schema1(f, t0, x0, h0)
tol     = 1e-3;
IterMax = 10;
theta   = 0.5;
[t1, x1, h1] = monThetaSChemaNewton(f, t0, x0, h0, tol, IterMax,theta);
end
```

2. **Schéma 2** : (*On prédit par le schéma d'Euler explicite: c'est le schéma de HEUN*)

$$\left\{\begin{array}{rcl} X^*_{n+1} & = & X_n + hF(t_n, X_n), \\ X_{n+1} & = & X_n + \frac{h}{2}\left[F(t_n, X_n) + F(t_{n+1}, X^*_{n+1})\right],\ 0 \leq n \leq N-1 \\ X_0 & = & X^0. \end{array}\right. \tag{14}$$

Listing 15: Fonction Matlab pour le Schema2

```
 function [t1, x1, h1] = Schema2(f, t0, x0, h0)
t1     = t0 + h0;
x1star = x0 + h0 * f(t0,x0);
x1     = x0 + (h0/2) * (f(t0,x0) + f(t1,x1star) );
h1     = h0 ;
end
```

3. **Schéma 3** : (*On prédit par une itération de la méthode du point fixe)*

$$\left\{\begin{array}{rcl} X^*_{n+1} & = & X_n + \frac{h}{2}\left[F(t_n, X_n) + F(t_{n+1}, X_n)\right], \\ X_{n+1} & = & X_n + \frac{h}{2}\left[F(t_n, X_n) + F(t_{n+1}, X^*_{n+1})\right],\ 0 \leq n \leq N-1 \\ X_0 & = & X^0. \end{array}\right. \tag{15}$$

Listing 16: Fonction Matlab pour le Schema3

```
function [t1, x1, h1] = Schema3(f, t0, x0, h0)
t1       = t0 + h0;
x1star   = x0 + (h0/2) * (f(t0,x0) + f(t1,x0) );
x1       = x0 + (h0/2) * (f(t0,x0) + f(t1,x1star) );
h1       = h0 ;
end
```

4. **Schéma 4** :(*On prédit par le schéma du point milieu qui est d'ordre 2)*

$$\left\{\begin{array}{rcl} X^*_{n+1} & = & X_n + hF(t_n + \frac{h}{2}, X_n + \frac{h}{2}F(t_n, X_n)), \\ X_{n+1} & = & X_n + \frac{h}{2}\left[F(t_n, X_n) + F(t_{n+1}, X^*_{n+1})\right],\ 0 \leq n \leq N-1 \\ X_0 & = & X^0. \end{array}\right. \tag{16}$$

Listing 17: Fonction Matlab pour le Schema4

```
 function [t1, x1, h1] = Schema4(f, t0, x0, h0)
t1       = t0 + h0;
x1star   = x0 + h0 * (f(t0 + h0/2.0, x0 + (h0/2.0) * f(t0,x0)) );
x1       = x0 + (h0/2) * (f(t0,x0) + f(t1,x1star) );
h1       = h0 ;
end
```

Pour évaluer l'ordre de ces schémas on fournit le Listing 18, qui produit la figure FIGURE 4.

Listing 18: Script Pour question 4

```
function [res] = question4(figNum)
t0 = 0;
tf = pi;
x0 =[1;0];
H  = [];
E1 = []; %Pour schema 1
E2 = []; %Pour schema 2
E3 = []; %Pour schema 3
E4 = []; %Pour schema 4

for k=1:12
  N = 2^k ;
  [t,x1,h] = solveurEDO(@monEDO,t0,tf,x0,N,@Schema1);
  H  = [H,h];
  E1 = [E1, norm(x1(:,end) - [-1;0])];
  [t,x2,h] = solveurEDO(@monEDO,t0,tf,x0,N,@Schema2);
  E2 = [E2, norm(x2(:,end) - [-1;0])];
  [t,x3,h] = solveurEDO(@monEDO,t0,tf,x0,N,@Schema3);
  E3 = [E3, norm(x3(:,end) - [-1;0])];
  [t,x4,h] = solveurEDO(@monEDO,t0,tf,x0,N,@Schema4);
  E4 = [E4, norm(x4(:,end) - [-1;0])];
end

logH  = log(H);
logE1 = log(E1);
logE2 = log(E2);
logE3 = log(E3);
logE4 = log(E4);

x  = linspace(min(log(H)), max(log(H)), 30);
y2 = (x - log(H(end))) + log(E1(end));
y4 = 2 * (x - log(H(end))) + log(E1(end));

figure(figNum);
hold off;

plot(logH,logE1,'*r', logH,logE1,'+b', logH,logE1,'og', logH(2:end),logE4(2:end),'sm');

hold on
plot(x,y2,'g',x,y4,'--r');
xlabel('Log(h)');
ylabel('Log(E)');
legend('Schema1','Schema2','Schema3','Schema4', strcat('droite de pente = ','1'), strcat('droite de pente ='↩
    , '2'));
title('Graphique de comparaison des 4 schemas');

hold off;
print('Q4Comparaison.eps','-depsc');
res=1;
end
```

Nous fournissons en annexe le listing complet du tp (Listing 19) qui décrit le contenu du fichier tp2.m.
(*Rappelons qu'il est aussi possible de détacher les fonctions qui s'y trouvent dans des fichiers séparés.*)

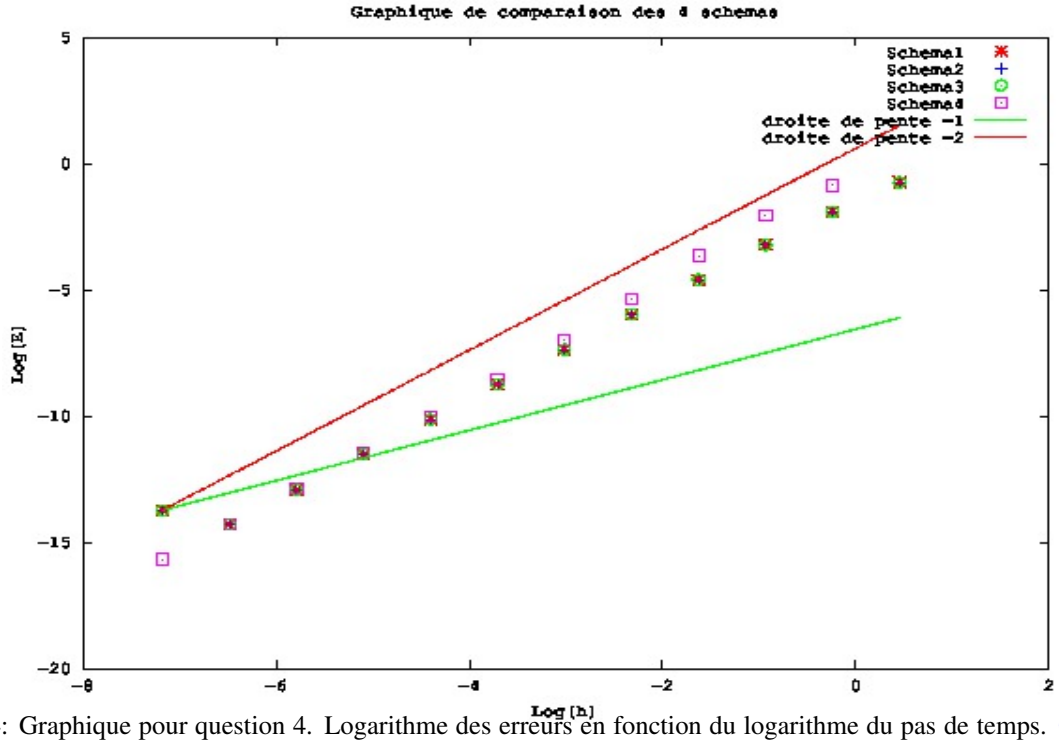

Figure 4: Graphique pour question 4. Logarithme des erreurs en fonction du logarithme du pas de temps. On observe que l'ordre 2 est préservé.

.

Listing 19: Script complet: fichier ImplicitOneStep.m

```
function [res] = ImplicitOneStep()

%% question 3
% question 3-2
figNum = 1;
[res]  = question32(figNum);

% question 3-3
figNum = 2;
[res]  = question33(figNum);

% question 3-4
  theta  = 0;   figNum = 3;
  [res]  = question34CasUnTheta(theta, figNum);
  theta  = 0.5; figNum = 4;
  [res]  = question34CasUnTheta(theta, figNum);
  theta  = 1;   figNum = 5;
  [res]  = question34CasUnTheta(theta, figNum);

% question 3-5
  theta  = 0;   figNum = 6;
  [res]  = question35CasUnTheta(theta, figNum);
  theta  = 0.5; figNum = 7;
  [res]  = question35CasUnTheta(theta, figNum);
  theta  = 1;   figNum = 8;
  [res]  = question35CasUnTheta(theta, figNum);

%% question 4
  figNum = 9;
  [res]  = question4(figNum);
res =1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%  question 3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%=============================================================
function [res]  = question32(figNum)
theta=[0,0.25,0.5,0.75,1];
col     = ['r';'*b';'+g';'m';'--k'];
tol     = 1e-3;
IterMax = 10;
t0      = 0;
tf      = pi;
x0      = [1;0];
N       = 50 ;
```

```
figure(figNum); clf; hold off;

% solutions approchees
for i=1:length(theta)
 monSchema = @(f, t0, x0, h0) monThetaSChemaNewton(f, t0, x0, h0, tol, IterMax,theta(i));
 [t,x,h]   = solveurEDO(@monEDO,t0,tf,x0,N,monSchema);
 plot(x(1,:), x(2,:),col(i,:)); hold on;
end
% solution exacte
x = cos(t); y = -sin(t);
plot(x,y,'k');

legend('theta = 0','theta = 0.25','theta =0.5','theta =0.75', 'theta = 1', 'sol exacte');
title('Solution avec methode de Newton');
hold off;

% impression du graphique
print('Q32SolutionNewton.eps','-depsc');
res =1;
end
%%=====================================================================
function [res]  = question33(figNum)
theta=[0,0.25,0.5,0.75,1];
col     = ['r';'*b';'+g';'m';'--k'];
tol     = 1e-3;
IterMax = 10;
t0      = 0;
tf      = pi;
x0      = [1;0];
N       = 50 ;

figure(figNum); clf; hold off;

% solutions approchees
for i=1:length(theta)
 monSchema = @(f, t0, x0, h0) monThetaSChemaPicard(f, t0, x0, h0, tol, IterMax,theta(i));
 [t,x,h]   = solveurEDO(@monEDO,t0,tf,x0,N,monSchema);
 plot(x(1,:), x(2,:),col(i,:)); hold on;
end

% solution exacte
x = cos(t); y = -sin(t);
plot(x,y,'k');

legend('theta = 0','theta = 0.25','theta =0.5','theta =0.75', 'theta = 1', 'sol exacte');
title('Solution avec methode de Point fixe');
hold off;

% impression du graphique
print('Q33SolutionPointFixe.eps','-depsc');
res =1;
end
%%=====================================================================
function [res]  = question34CasUnTheta(theta, figNum)
% theta  : valeur de theta
% figNum : numero de la figure
%-------------------------------------------------------

tol       = 1e-3;
IterMax   = 10;
t0        = 0;
tf        = pi;
x0        = [1;0];
N         = 50 ;
monSchema = @(f, t0, x0, h0) monThetaSChemaNewton(f, t0, x0, h0, tol, IterMax,theta);

H = []; % pour stocker les pas
E = []; % pour stocker les erreurs

for k=1:12
  N = 2^k ;
  [t,x,h] = solveurEDO(@monEDO,t0,tf,x0,N,monSchema);
  H = [H,h];
  E = [E, norm(x(:,end) - [-1;0])];
end

% calcul des log
logH = log(H);
logE = log(E);

% droites de pente 1 et 2 passant par le dernier point
x   = linspace(min(log(H)), max(log(H)), 30);
y1  = (x - log(H(end))) + log(E(end));
y2  = 2 * (x - log(H(end))) + log(E(end));

%
figure(figNum); clf; hold off;
```

```
plot(logH,logE,'-*'); hold on;      % courbe log log
plot(x,y1,'g',x,y2,'--r');          % droites de pente 1 et 2
xlabel('Log(h)'); ylabel('Log(E)');
legend('log(E)', strcat('droite de pente = ','1'), strcat('droite de pente =', '2'));
title(strcat('Courbe pour theta = ', num2str(theta)));

% Impression du graphique au format .eps couleur
nomFichier = strcat('Q34CourbeTheta', num2str(theta),'.eps');
print(nomFichier,'-depsc');
res=1;
end
%%=====================================================================
function [res]  = question35CasUnTheta(theta, figNum)
% theta  : valeur de theta
% figNum : numero de la figure
%-----------------------------------------------

tol       = 1e-3;
IterMax   = 1;          % C'EST ICI  QU'IL Y A EU MODIFICATION
t0        = 0;
tf        = pi;
x0        = [1;0];

monSchema = @(f, t0, x0, h0) monThetaSChemaPicard(f, t0, x0, h0, tol, IterMax,theta);

H = []; % pour stocker les pas
E = []; % pour stocker les erreurs

for k=1:12
  N = 2^k ;
  [t,x,h] = solveurEDO(@monEDO,t0,tf,x0,N,monSchema);
  H = [H,h];
  E = [E, norm(x(:,end) - [-1;0])];
end

% calcul des log
logH = log(H);
logE = log(E);

% droites de pente 1 et 2 passant par le dernier point
x   = linspace(min(log(H)), max(log(H)), 30);
y1  = (x - log(H(end))) + log(E(end));
y2  = 2 * (x - log(H(end))) + log(E(end));

%
figure(figNum); clf; hold off;
plot(logH,logE,'-*'); hold on;      % courbe log log
plot(x,y1,'g',x,y2,'--r');          % droites de pente 1 et 2
xlabel('Log(h)'); ylabel('Log(E)');
legend('log(E)', strcat('droite de pente = ','1'), strcat('droite de pente =', '2'));
title(strcat('Courbe pour theta = ', num2str(theta)));

% Impression du graphique au format .eps couleur
nomFichier = strcat('Q35CourbeTheta', num2str(theta),'.eps'); % MODIFICATION FAITE ICI AUSSI
print(nomFichier,'-depsc');
res=1;
end
%%%%%%%%%%%%%%%%%% monThetaSChemaNewton %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [t1, x1, h1] = monThetaSChemaNewton(f, t0, x0, h0, tol, IterMax,theta)
t1 = t0 + h0;
gPourNewton = @(Z) Ncursif(Z,f,t0,x0,h0,theta);
[x1,N] = Newton(gPourNewton,x0,tol,IterMax);
h1 = h0 ;
end
%------- fonction Ncursif ------
function [Nc, DzNc] = Ncursif(Z,f,t0,X0,h,theta)
[F0, DF0] = f(t0,X0);
[F1,   DF1] = f(t0+h,Z);
dim = length(Z);
Nc  = Z - X0  - h * ( (1-theta)* F0  + theta * F1 );
DzNc = eye(dim)  - h * theta * DF1;
end
%%%%%%%%%%%%%%%%%% monThetaSChemaPicard %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [t1, x1, h1] = monThetaSChemaPicard(f, t0, x0, h0, tol, IterMax, theta)
t1 = t0 + h0;
[F0, DF0] = f(t0, x0);
gPourPicard = @(Z) Pcursif(Z,f,t0,x0,h0,theta);
[x1,N] = Picard(gPourPicard,x0,tol,IterMax);
h1 = h0 ;
end
%------- fonction Pcursif ------
function [Pc] = Pcursif(Z,f,t0,X0,h,theta)
F0  = f(t0,X0);

F1  = f(t0+h,Z);

Pc  = X0  + h * ( (1-theta)* F0  + theta * F1 );
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% question 4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [res] = question4(figNum)
t0 = 0;
tf = pi;
x0 =[1;0];
H  = [];
E1 = []; %Pour schema 1
E2 = []; %Pour schema 2
E3 = []; %Pour schema 3
E4 = []; %Pour schema 4

for k=1:12
  N = 2^k ;
  [t,x1,h] = solveurEDO(@monEDO,t0,tf,x0,N,@Schema1);
  H  = [H,h];
  E1 = [E1, norm(x1(:,end) - [-1;0])];
  [t,x2,h] = solveurEDO(@monEDO,t0,tf,x0,N,@Schema2);
  E2 = [E2, norm(x2(:,end) - [-1;0])];
  [t,x3,h] = solveurEDO(@monEDO,t0,tf,x0,N,@Schema3);
  E3 = [E3, norm(x3(:,end) - [-1;0])];
  [t,x4,h] = solveurEDO(@monEDO,t0,tf,x0,N,@Schema4);
  E4 = [E4, norm(x4(:,end) - [-1;0])];
end

logH  = log(H);
logE1 = log(E1);
logE2 = log(E2);
logE3 = log(E3);
logE4 = log(E4);

x  = linspace(min(log(H)), max(log(H)), 30);
y2 = (x - log(H(end))) + log(E1(end));
y4 = 2 * (x - log(H(end))) + log(E1(end));

figure(figNum);
hold off;

plot(logH,logE1,'*r', logH,logE1,'+b', logH,logE1,'og', logH(2:end),logE4(2:end),'sm');

hold on
plot(x,y2,'g',x,y4,'--r');
xlabel('Log(h)');
ylabel('Log(E)');
legend('Schema1','Schema2','Schema3','Schema4', strcat('droite de pente = ','1'), strcat('droite de pente ='↩
     , '2'));
title('Graphique de comparaison des 4 schemas');

hold off;
print('Q4Comparaison.eps','-depsc');
res=1;
end
%=============================================================================
function [t1, x1, h1] = Schema1(f, t0, x0, h0)
tol     = 1e-3;
IterMax = 10;
theta   = 0.5;
[t1, x1, h1] = monThetaSChemaNewton(f, t0, x0, h0, tol, IterMax,theta);
end
%------------------------------------------------------------------------------
function [t1, x1, h1] = Schema2(f, t0, x0, h0)
t1     = t0 + h0;
x1star = x0 + h0 * f(t0,x0);
x1     = x0 + (h0/2) * (f(t0,x0) + f(t1,x1star) );
h1     = h0 ;
end
%-----------------------------------------------------------------------------
function [t1, x1, h1] = Schema3(f, t0, x0, h0)
t1       = t0 + h0;
x1star   = x0 + (h0/2) * (f(t0,x0) + f(t1,x0) );
x1       = x0 + (h0/2) * (f(t0,x0) + f(t1,x1star) );
h1       = h0 ;
end
%-----------------------------------------------------------------------------
function [t1, x1, h1] = Schema4(f, t0, x0, h0)
t1       = t0 + h0;
x1star   = x0 + h0 * (f(t0 + h0/2.0, x0 + (h0/2.0) * f(t0,x0)) );
x1       = x0 + (h0/2) * (f(t0,x0) + f(t1,x1star) );
h1       = h0 ;
end
%-----------------------------------------------------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% UTILITAIRES %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%% Solveur d'EDO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [t,x,h]= solveurEDO(f,t0,tf,x0,N,schema)
```

```
t_temp=linspace(t0,tf,N+1);
h=t_temp(2) - t_temp(1);
t_temp =[];
x= [x0];
t= [t0];
for k=1:N
  [t1,x1,hh] = schema(f,t(k),x(:,k),h);
  x=[x,x1];
  t=[t,t1];
end
end
%%%%%%%%%%%%%%%%% Methode de Newton %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x,N] = Newton(g,x0,tol,iterMax)
x = x0;
iter = 0;
conv = false;
while(false == conv)
 [gn,Dgn] = g(x);
 delta    = Dgn\gn ;
 x = x - delta;
 iter = iter + 1;
 conv = (norm(delta)< tol * norm(x)) ||(iter > iterMax);
end
N = iter;
end
%%%%%%%%%%%%%%%% Methode de Point Fixe %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x,N] = Picard(g,x0,tol,iterMax)
x = x0;
iter = 0;
conv = false;
while(false == conv)
 %calcul
 xx = g(x);
 delta = xx - x;
 x = xx;
 iter = iter + 1;
 conv = (norm(delta)< tol * norm(x)) ||(iter > iterMax);
end
N = iter;
end
%%%%%%%%%%%%%%%%%%%% monSchemaEulerExplicite %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [t1, x1, h1] = monSchemaEulerExplicite(f, t0, x0, h0)
t1 = t0 + h0;
x1 = x0 + h0 * f(t0,x0);
h1 = h0 ;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [f,df] = monEDO(t,x)
x1 = x(1);
x2 = x(2);
u       = x1.^2 + x2.^2;
f  = [u * x2; - u * x1 ];
df(1,:) = [ 2 * x1 * x2, (u + 2*x2.^2)];
df(2,:) = [(-u - 2*x1.^2) , -2*x2*x1];
end
%%%%%%%%%%%%%%%%%%%%%%%%% FIN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```