© *J.-B.A.K. <jean-baptiste.apoung@math.u-psud.fr>*

Fiche de TP3 : Schémas à pas multiples

Thème - 1 *Génération de quelques schémas à pas multiples*

Exercice-1 **:**

Dans cet exercice on fournit un script **Mupad** pour générer quelques schémas à pas multiples les plus courants. *Cet exercice se déroulera sur l'environnement* ***Mupad***. Une fois connecté sur votre machine (de la salle de TP) commencez par lancer **Mupad**, et rentrer les fonctions données si dessous.

Adams-Bashforth

```
"@copyright: J.-B. APOUNG, Univ. ORSAY 2012
 @bref     : fonction MUPAD générant le schema de Nystrom à k pas
 @exemple  : Pour generer le schema a 4 pas faire:
 %> adams_bashforth(4);
"
adams_bashforth:=proc(k)
begin
 xl:=[t[n]-j*h $j=0..(k-1)];
 yl:=[f[n-j] $j=0..(k-1)];
 p:=interpolate(xl,yl,s);
 x[n+1] = x[n] + factor(int(p,s=t[n]..(t[n]+h)));
end_proc:
```

Adams-Moulton

```
"@copyright: J.-B. APOUNG, Univ. ORSAY 2012
 @bref     : fonction MUPAD générant le schema d' Adams-Moulton à k pas
 @exemple  : Pour generer le schema a 4 pas faire:
 %> adams_moulton(4);
"
adams_moulton:=proc(k)
begin
xl:=[t[n]-j*h $j=-1..(k-2)];
yl:=[f[n-j] $j=-1..(k-2)];
p:=interpolate(xl,yl,s);
x[n+1] = x[n] + factor(int(p,s=t[n]..(t[n]+h)));
end_proc:
```

Nyström

```
"@copyright: J.-B. APOUNG, Univ. ORSAY 2012
 @bref     : fonction MUPAD générant le schema de Nystrom à k pas
 @exemple  : Pour generer le schema a 4 pas faire:
 %> nystrom(4);
"
nystrom:=proc(k)
begin
 xl:=[t[n]-j*h $j=0..(k-1)];
 yl:=[f[n-j] $j=0..(k-1)];
 p:=interpolate(xl,yl,s);
 x[n+1] = x[n-1] + factor(int(p,s=(t[n]-h)..(t[n]+h)));
end_proc:
```

Milne-Simpson

```
"@copyright: J.-B. APOUNG, Univ. ORSAY 2012
 @bref     : fonction MUPAD générant le schema de Milne-Simpson à k pas
 @exemple  : Pour generer le schema a 4 pas faire:
 %> milne_simpson(4);
```

```
"
milne_simpson:=proc(k)
begin
 xl:=[t[n]-j*h $j=-1..(k-1)];
 yl:=[f[n-j] $j=-1..(k-1)];
 p:=interpolate(xl,yl,s);
 x[n+1] = x[n-1] + factor(int(p,s=(t[n]-h)..(t[n]+h)));
end_proc:
```

Différentiation-retrograde

```
"@copyright: J.-B. APOUNG, Univ. ORSAY 2012
 @bref     : fonction MUPAD générant le schema de Différentiation-retrograde à k pas
 @exemple  : Pour generer le schema a 4 pas faire:
 %> ER(4);
"
ER:=proc(k)
begin
 xl:=[t[n]-j*h $j=-1..(k-1)];
 yl:=[x[n-j] $j=-1..(k-1)];
 p:=interpolate(xl,yl,s);
 factor(h*limit(diff(p,s),s=t[n]+h))=h*f[n+1];
end_proc:
```

Q-1-1 : Pour chacune de ces fonctions, générer le schéma correspondant pour les pas $k = 1, 2, 3, 4, 5$. On rappelle qu'un exemple d'utilisation serait

```
"AM1:"; adams_moulton(1);
"AM3:"; adams_moulton(3);
"AM4:"; adams_moulton(4);
```

Exercice-2 :

Comparer les schémas obtenus ci-dessus avec des exemples vus en cours. Retrouver certains d'eux en utilisant les formules génériques reposant sur l'interpolation de Newton, vues en cours.

Thème - 2 *Programmation de schémas à pas multiples : cas explicites*

Sans nuire à la généralité, on va considérer l'équation différentielle

$$x'(t) = 4t\, x(t)^{\frac{1}{2}}, \quad t \in]0, 2[, \quad x(0) = 1, \tag{1}$$

de solution exacte $x(t) = (1 + x^2)^2$, (fournie pour les besoins de tests de convergence).
On posera $f(t, y) = 4t\, y^{\frac{1}{2}}$, de sorte que l'équation (1) s'écrive $x'(t) = f(t, x(t))$.

On considère aussi le schéma à 2 pas suivant pour sa discrétisation :

$$\begin{cases} x_{n+2} - (1+a)x_{n+1} + a x_n = \frac{1}{2}h\Big[(3-a)f_{n+1} - (b+a)f_n\Big], \quad n = 0, \ldots, N-2, \\ x_0 = 1; \quad x_1 \quad \text{(à déterminer)}. \end{cases} \tag{2}$$

où a, b sont des réels à fournir dans la suite. On rappelle les notations $f_j = f(t_j, x_j), \quad t_j = jh, \quad j = 0, \ldots, N$.

Exercice-1 : Préparation du démarrage (ou initialisation).

Il est question de fournir une fonction qui permette de fournir les données initiales manquantes pour le schéma (2).

Q-1-1 : Ecrire une fonction **Matlab** qui complète les données initiales afin que le schéma à k pas puisse démarrer :

```
function [tn,xn,fn] = monInitMultiPas(f,t0,tf,x0,h,k)
%ENTREE:
% f  -> la fonction second-membre de l'EDO
% t0 -> l'instant initiale
% tf -> l'instant final (juste pour nous assurer qu'on ne sort pas du domaine de calcul
% x0 -> solution initiale
```

```
% h  -> le pas de temps de la méthode
% k  -> le nombre de pas de la méthode (schema à k pas)
%SORTIE:
% tn  -> [t0, ..., tk-1]
% xn  -> [x0, ..., xk-1]
% fn  -> [f0, ..., fk-1]
```

Cette fonction fera exceptionnellement usage de la solution exacte $x(t) = (1 + t^2)^2$ du problème. Il faut noter toutefois que ce n'est pas toujours possible en pratique, puisqu'on ne dispose pas de la solution exacte.

Q-1-2 : Proposer aussi une fonction

```
function [tn,xn,fn] = monInitMultiPasRK4(f,t0,tf,x0,h,k)
```

Qui utilise cette-fois le schéma de Runge-Kutta d'ordre 4 pour réaliser le même travail. Cette fonction est-elle suffisante pour le schéma considéré ?

Exercice-2 : Production de solutions.

On commence par dire au moyen d'une fonction, comment le schéma fournit la prochaine solution. Cette fonction est ensuite appelée dans une boucle contrôlée par le domaine d'intégration, pour fournir la totalité des solutions. On utilise encore une fonction pour cette tâche.

Q-2-1 : Ecrire une fonction **Matlab**

```
function [tn1,xn1,fn1] = monSchemaMultiPas(f, ltn, lxn, lfn, h, k, param)
% fonction qui avance le schéma multipas d'un pas.
% On suppose disposer de lxn=[x0, ..., xn] et on veut fabriquer la prochaine xn+1, tn+1 et fn+1
%ENTREE:
% f  -> la fonction second-membre de l'EDO
% ltn -> la liste des  instants
% lxn -> la liste des  solutions aux instants ltn
% lfn -> la liste des  valeurs f(tn,xn) (afin d' éviter l'évaluation de f(.,.) qui peut être couteux!)
% h  -> le pas de temps de la méthode (le pas etant constant on peut l'éviter)
% k  -> le nombre de pas de la méthode
% param -> les parametres par exemple param=[a,b]
% SORTIE:
% tn1  ->  prochain instant i.e tn+1 =tn(end) + h
% xn1  ->  prochaine solution (c-est-à-dire xn+1)
% fn1  ->   f(tn1,xn1)
```

Remarque : Dans la pratique, pour des besoins d'économie de mémoire, on limite les entrées ***ltn, lxn, lfn****, aux k dernières valeurs lorsque le schéma à k pas.*

Q-2-2 : Ecrire une fonction **Matlab**, **monMultiPas** qui calcule la suite des solutions produites par le schéma.

```
function [t,x] = monMultiPas(f, t0, tf, x0, h, param)
% f->second-membre de l'edo ; (t0,x0)-> données initiales; tf-> temps final; h-> pas; param-> parametres.
% t-> liste des instants générés
% x-> liste des solutions générées aux instants t
```

Cette fonction ferra appel à la fonction **monSchemaMultiPas** précédente. On peut donc faire le choix de la passer en paramètre, pour plus de généricité. C'est-à-dire, opter pour le prototype

```
function [t,x] = monMultiPas(f, t0, tf, x0, h, param, Schema)
```

Exercice-3 : **Validations**

Q-3-1 : Evaluer la fonction pour $h = 0.1, b = 1$, et afficher la suite

```
instant              solution exacte                     solution                                             solution
 t                          x(t)                           a=0                                                  a=-5
-----------------------------------------------------------------------------------------------------------
 .                    .                                   .                                                    .
 .                    .                                   .                                                    .
 .                    .                                   .                                                    .
```

On peut aussi choisir de représenter sur une même figure les courbes des différentes solutions.

Q-3-2 : Reprendre l'exercice pour $h = 0.025$. Conclure que le schéma diverge pour $a = -5$.
Ce constat était-il prévisible au regard de la stabilité ?

Q-3-3 : On prend à présent $a = 0$ et $b = -1$. Reprendre l'exercice précédent en affichant cette fois-ci

```
 t                x(t)                    solution                    solution                          solution
                                           h=0.1                       h=0.05                            h=0.025
-----------------------------------------------------------------------------------------------------------
 .                 .                       .                        .                                    .
 .                 .                       .                        .                                    .
 .                 .                       .                        .                                    .
```

On peut aussi choisir de représenter sur une même figure les courbes des différentes solutions.

En déduire que le schéma n'est pas convergent pour ces valeurs de a et b.
Ce constat était-il prévisible au regard de la consistance ?

Thème - 3 *Evaluation numérique d'un critère de stabilité de quelques schémas à pas multiples*

Dans cette partie on souhaite vérifier numériquement la stabilité en domaine bornée, de quelques schémas à pas multiples. On rappelle le résultat :

Un schéma à k pas, de premier polynôme caractéristique ϱ, lorsqu'il est constructible, est stable (ou 0-stable) si et seulement si les racines de ϱ sont toutes de module inférieur ou égale à 1, celles de module 1 étant simples.

On appelle **polynôme adjoint** *de* $P(z) = \sum_{k=0}^{n} a_k z^k$*, le polynôme défini par* $P^*(z) = \sum_{k=0}^{n} \bar{a}_{n-k} z^k$*, où* $\bar{a}$ *désigne le complexe conjugué de* a*.*

Un polynôme P(z) est dit de **Schur** *si et seulement si* $|P^*(0)| > |P(0)|$ *et* $Q(z) = \frac{1}{z}\left(P^*(0)P(z) - P(0)P^*(z)\right)$ *est de Schur. (On suppose que* $a_n \neq 0, a_0 \neq 0$*, sinon il faut diviser par la bonne puissance de z.).*

L'information que nous voulons exploiter, est celle selon laquelle, *un polynôme P est de Schur si toutes ses racines sont de module strictement inférieur à 1.* On dispose alors d'une condition *suffisante* de stabilité.

Exercice-1 :

Q-1-1 : Ecrire une fonction **Matlab** qui prend les coefficients du polynôme P par ordre de monôme croissant

```
function [Q] = suivantPolySchur(P)
```

et qui retourne les coefficients du polynôme $Q(z) = \frac{1}{z}\left(P^*(0)P(z) - P(0)P^*(z)\right)$. Il faudra simplifier ce polynôme tant que $Q(0) = 0$. C'est-à-dire si par exemple $Q(z) = 3z^4 + 2z^3 + z^2$ alors on le remplace par $Q(z) = 3z^2 + 2z + 1$.

Q-1-2 : Ecrire une fonction **Matlab**, dénommée **critereSchur** qui prend en entrée un polynôme p identifié par ses coefficients et qui retourne un nombre positif si ce polynôme est de *Schur*, et un nombre négatif sinon. Cette fonction aura pour prototype :

```
function [res] = critereSchur(P)
```

Cette fonction pourra s'appeler de manière récursive.

Exercice-2 :

Q-2-1 : Utiliser votre implémentation de la fonction **critereSchur** définie précédemment pour étudier la 0-stabilité des schémas de différentiation rétrogrades à k pas, avec $k = 1, 2, 3, 5, 6, 7$. On prendra soin bien avant, en l'expliquant, de diviser le premier polynôme caractéristique du schéma $\varrho(z)$ par $z - 1$.

Q-2-2 : Est-il nécessaire d'utiliser cette méthode pour étudier la 0-stabilité des schémas d'Adams, de Nyström et de Miles ?

Thème - 4 *Schémas à pas multiples : implicite*

Exercice-1 : Le schéma d'Adams-Moulton à k pas s'écrit sous la forme

$$(AMk)\left\{\begin{array}{l} x_{n+k} - x_{n+k-1} = h\sum_{j=0}^{k}\beta_j^* f_{n+j}, \quad n = 0, \ldots, N-k, \\ x_0, x_1, \ldots, x_{k-1} \quad \text{donnés.} \end{array}\right. \tag{3}$$

Dans la suite on s'intéresse au schéma à 5 pas.

Q-1-1 : Rappeler les coefficients β_i^* du schéma d'Adams-Moulton à 5 pas. Affirmer que ce schéma est implicite.

Q-1-2 : Montrer que ce schéma est convergent. Quelle est son ordre ? On précisera la constante d'erreur.

Exercice-2 : Le schéma d'Adams-Bashforth à k pas s'écrit sous la forme

$$(ABk)\left\{\begin{array}{l} x_{n+k} - x_{n+k-1} = h\sum_{j=0}^{k-1}\beta_j f_{n+j}, \quad n = 0, \ldots, N-k, \\ x_0, x_1, \ldots, x_{k-1} \quad \text{donnés.} \end{array}\right. \tag{4}$$

Q-2-1 : Rappeler les coefficients β_i du schéma d'Adams-Bashforth à 5 pas. Affirmer que ce schéma est explicite.

Q-2-2 : Ecrire un script **matlab** qui calcule numériquement l'ordre de ce schéma. On utilisera le problème modèle suivant

$$x'(t) = -\omega(x(t) - a(t)), \quad 0 \leq x \leq 1, x(0) = 1, \qquad \omega > 0. \tag{5}$$

On prendra différemment $\omega = 1, 10, 50$, ainsi que $a(t) = t^2, a(t) = e^{-t}, a(t) = e^t$. Les valeurs de démarrage du schéma seront calculées à partir de la solution exacte que l'on calculera.

Exercice-3 :

On substitue au schéma implicite (3), le schéma de Prédiction-Evaluation-Correction-Evaluation (PECE) suivant :

$$(PCk)\left\{\begin{array}{l} \left\{\begin{array}{ll} P: & x_{n+k}^* - x_{n+k-1} = h\sum_{j=0}^{k-1}\beta_j f_{n+j}, \\ E: & f_{n+k}^* = f(t_{n+k}, x_{n+k}^*), \\ C: & x_{n+k} - x_{n+k-1} = h\sum_{j=0}^{k-1}\beta_j^* f_{n+j} + h\beta_k^* f_{n+k}^*, \\ E: & f_{n+k} = f(t_{n+k}, x_{n+k}) \end{array}\right\} \quad n = 0, \ldots, N-k, \\ x_0, x_1, \ldots, x_{k-1} \quad \text{donnés.} \end{array}\right. \tag{6}$$

Q-3-1 **:** Ecrire ce schéma pour k = 5 et montrer qu'on obtient un schéma explicite.

Q-3-2 **:** Calculer numériquement l'ordre de ce schéma. On pourra se servir du problème modèle (5).