Fiche n^0 2. Résolution numérique d'EDOs en langage C

Exercice 1 **– Schéma de Crank-Nicolson**
Etant donnés $a \in \mathbb{R}$ et $y_0 \in \mathbb{R}$, on souhaite résoudre de façon approchée l'équation aux dérivées ordinaires linéaire à coefficients constants suivante :

$$y' = ay \text{ avec } y(0) = y_0.$$

a) Quel était l'ordre des schémas d'Euler explicite et implicite vus dans la fiche n^0 1 ? Pouvez vous le prouver simplement en observant la valeur de y_n en fonction de y_0 et en la comparant à la solution exacte $y(n\Delta t)$?
b) Pour écrire un schéma d'ordre deux, on prouvera tout d'abord que

$$\frac{1}{\Delta t}\{y[(n+1)\Delta t] - y(n\Delta t)\}$$

est une approximation d'ordre deux de $y'[(n+1/2)\Delta t]$; on prouvera ensuite que

$$\frac{1}{2}\{y[(n+1)\Delta t] + y(n\Delta t)\}$$

est une approximation d'ordre deux de $y[(n+1/2)\Delta t]$. Déduisez en le schéma de Crank-Nicolson :

$$\frac{1}{\Delta t}(y_{n+1} - y_n) = \frac{a}{2}(y_{n+1} + y_n)$$

c) Dans le schéma de Crank-Nicolson, exprimez y_{n+1} en fonction de y_n, puis y_n en fonction de y_0 et vérifiez que y_n est bien une approximation à l'ordre deux de la solution exacte au temps $n\Delta t$. Sous quelle condition sur Δt ce schéma est-il monotone ?

---

a) Euler explicite et implicite sont d'ordre un. Montrons le directement dans notre cas : la solution exacte de l'EDO au temps T est $y(T) = y_0 \exp(aT)$ et la solution numérique par Euler explicite à l'itération $n = \frac{T}{\Delta t}$ est $y_n = y_0(1+a\Delta t)^n$. On cherche à montrer que $|y(T) - y_n| = \mathcal{O}(\Delta t)$. [1] Par le caractère localement Lipschitzien de l'exponentielle autour de aT, il suffit de montrer que $aT - n\ln(1+a\Delta t) = \mathcal{O}(\Delta t)$. C'est le cas comme le montre le développement limité $\ln(1+a\Delta t) = a\Delta t - (a\Delta t)^2/2 + \mathcal{O}(\Delta t^3)$. En utilisant $n = \frac{T}{\Delta t}$, on obtient donc $aT - n\ln(1+a\Delta t) = a^2 T\Delta t/2 + \mathcal{O}(\Delta t^2) = \mathcal{O}(\Delta t)$. Un calcul similaire montre qu'Euler implicite est aussi d'ordre un.

b) On a $y((n+1)\Delta t) = y((n+1/2)\Delta t) + (\Delta t/2)\, y'((n+1/2)\Delta t) + (\Delta t^2/8)\, y''((n+1/2)\Delta t) + \mathcal{O}(\Delta t^3)$ et $y(n\Delta t) = y((n+1/2)\Delta t) - (\Delta t/2)\, y'((n+1/2)\Delta t) + (\Delta t^2/8)\, y''((n+1/2)\Delta t) + \mathcal{O}(\Delta t^3)$
Par différence de ces deux quantités on obtient donc :
$\frac{1}{\Delta t}\{y[(n+1)\Delta t] - y(n\Delta t)\} = y'((n+1/2)\Delta t) + \mathcal{O}(\Delta t^2)$
Par leur somme, on obtient :
$\frac{1}{2}\{y[(n+1)\Delta t] + y(n\Delta t)\} = y((n+1/2)\Delta t) + \mathcal{O}(\Delta t^2)$
On écrit ensuite que $y'((n+1/2)\Delta t) = ay((n+1/2)\Delta t)$ (l'EDO est valable à tout instant et donc en particulier en $t = (n+1/2)\Delta t$). En remplaçant ces deux quantités par les égalités ci-dessus, on obtient
$\frac{1}{\Delta t}\{y[(n+1)\Delta t] - y(n\Delta t)\} = a\frac{1}{2}\{y[(n+1)\Delta t] + y(n\Delta t)\} + \mathcal{O}(\Delta t^2)$
Le schéma numérique découle de cette égalité en négligeant le terme en $\mathcal{O}(\Delta t^2)$ et en notant, pour tout n, y_n l'approximation de $y(n\Delta t)$.

c) A partir du schéma, on obtient $y_{n+1} = \frac{(1+\frac{a\Delta t}{2})}{(1-\frac{a\Delta t}{2})} y_n$. Ceci définit donc un suite géométrique et nous avons alors : $y_n = \left[\frac{(1+\frac{a\Delta t}{2})}{(1-\frac{a\Delta t}{2})}\right]^n y_0$.

1. La notation $|y(T) - y_n| = \mathcal{O}(\Delta t)$ signifie qu'il existe une constante $C \neq 0$ telle que $|y(T) - y_n| = C\Delta t(1+\varepsilon(\Delta t))$ avec $\lim_{\Delta t \longrightarrow 0} \varepsilon(\Delta t) = 0$.

Pour prouver l'ordre deux, il suffit de prouver que $aT - n[\ln(1+\frac{a\Delta t}{2}) - \ln(1-\frac{a\Delta t}{2})] = \mathcal{O}(\Delta t^2)$, ce qui s'obtient en effectuant un développement limité de $\ln(1+\frac{a\Delta t}{2})$ et de $\ln(1-\frac{a\Delta t}{2})]$ pour Δt petit.

Ce schéma (comme toute suite géométrique) est monotone si et seulement si sa raison est positive ou nulle (ou son itéré initial est nul). Si a est positif, la condition de monotonie est donc $\Delta t < \frac{2}{a}$. Lorsque a est négatif, la condition de monotonie est donc $\Delta t \leq \frac{2}{|a|}$.

Exercice 2 **– Schéma BDF d'ordre deux**

Pour résoudre de façon approchée une équation du type

$$y' = f(t,y) \text{ avec } y(0) = y_0,$$

la méthode "Backward Differentiation Formula" (BDF) d'ordre p approche y' au temps $(n+1)\Delta t$ à l'ordre p à l'aide d'une combinaison linéaire des $p+1$ valeurs $y[(n+1-i)\Delta t]$, avec $0 \leq i \leq p$ et évalue le membre de droite de l'équation au temps $(n+1)\Delta t$. On écrit donc :

$$\sum_{i=0}^{i=p} a_{i,p} y_{n+1-i} = f[(n+1)\Delta t, y_{n+1}]$$

en choisissant les coefficients $a_{i,p}$ de telle sorte que $\sum_{i=0}^{i=p} a_{i,p} y_{n+1-i}$ soit une approximation d'ordre p de $y'[(n+1)\Delta t]$

a) Cette méthode est elle implicite ou explicite ? Quelle est l'autre nom de cette méthode pour $p = 1$?

b) Montrer, en effectuant des développements de Taylor autour de $(n+1)\Delta t$, que le schéma BDF d'ordre deux pour l'équation $y' = ay$ s'écrit

$$\frac{1}{2}(3y_{n+1} - 4y_n + y_{n-1}) = \Delta t\, a y_{n+1}$$

Pour la question suivante, on rappelle que pour trouver la solution générale d'une suite récurrente à trois termes du type $au_{n+1} + bu_n + cu_{n-1} = 0$, (avec a, b et c dans $\mathbb{R}$), on résout l'équation du second degré $ar^2 + br + c = 0$. On a alors trois cas :

- *Si $\Delta > 0$, alors la solution générale est du type $u_n = A(r_1)^n + B(r_2)^n$, où r_1 et r_2 sont les solutions de l'équation du second degré.*
- *Si $\Delta = 0$ alors la solution générale est du type $u_n = (A+Bn)(r)^n$, où r est la solution double de l'équation du second degré.*
- *Si $\Delta < 0$, alors la solution générale est du type $u_n = \rho^n [A\cos(n\theta) + B\sin(n\theta)]$, où ρ est le module et θ l'argument d'une des deux solutions de l'équation du second degré (l'autre étant le complexe conjugué).*

c) Si $a < 0$ montrer que la suite (y_n) est stable pour toute valeur de Δt. Si $a > 0$ trouver une condition suffisante pour que la suite (y_n) soit non-oscillante.

---

a) Le membre de gauche et le membre de droite de l'égalité définissant le schéma font tous les deux intervenir y_{n+1}. Le schéma est donc implicite. Pour $p = 1$, l'approximation de la dérivée est obtenue à l'aide de $y[(n+1)\Delta t]$ et $y(n\Delta t)$. C'est donc l'approximation classique $y'[(n+1)\Delta t] \approx \frac{1}{\Delta t}\{y[(n+1)\Delta t] - y(n\Delta t)\}$ et le schéma obtenu est Euler implicite.

b) Effectuons les développements limités suivants :

$y(n\Delta t) = y[(n+1)\Delta t] - \Delta t y'[(n+1)\Delta t] + (\Delta t^2/2) y''[(n+1)\Delta t] + \mathcal{O}(\Delta t^3)$

$y[(n-1)\Delta t] = y[(n+1)\Delta t] - 2\Delta t y'[(n+1)\Delta t] + (2\Delta t^2) y''[(n+1)\Delta t] + \mathcal{O}(\Delta t^3)$

On obtient donc

$$\frac{1}{2\Delta t}\{3y[(n+1)\Delta t] - 4y(n\Delta t) + y[(n-1)\Delta t]\} = y'[(n+1)\Delta t] + \mathcal{O}(\Delta t^2)$$

On écrit ensuite que $y'[(n+1)\Delta t] = ay[(n+1)\Delta t]$ (l'EDO est valable à tout instant et donc en particulier en $t = (n+1)\Delta t$). En remplaçant le membre de gauche par l'égalité ci-dessus, on obtient

$$\frac{1}{2\Delta t}\{3y[(n+1)\Delta t] - 4y(n\Delta t) + y[(n-1)\Delta t]\} = ay[(n+1)\Delta t] + \mathcal{O}(\Delta t^2)$$

Le schéma numérique découle de cette égalité en négligeant le terme en $\mathcal{O}(\Delta t^2)$ et en notant, pour tout n, y_n l'approximation de $y(n\Delta t)$.

c) Le schéma s'écrit aussi

$$(3 - 2a\Delta t)y_{n+1} - 4y_n + y_{n-1} = 0$$

Le discriminant de l'équation caractéristique associée est : $\Delta = 4(1 + 2a\Delta t)$.

Si $a > 0$, ce discriminant est toujours positif strictement. Il suffit que les deux solutions soient positives pour que la suite soit non-oscillante. Une condition équivalente est que leur produit et leur somme soient tous les deux positifs. Or le produit des racines est égal à $\frac{1}{3-2a\Delta t}$ et leur somme à $\frac{4}{3-2a\Delta t}$. La condition obtenue est donc $3 - 2a\Delta t > 0$, soit $\Delta t < \frac{3}{2a}$.

Si $a < 0$, on a deux cas :

- soit $\Delta \geq 0$, auquel cas les deux racines sont positives (car $(3 - 2a\Delta t) > 0$, voir raisonnement ci-dessus). Il suffit de prouver que la plus grande d'entre elles est strictement inférieure à 1 pour prouver la stabilité. Ceci se ramène à prouver que $4 + 2\sqrt{1 + 2a\Delta t} \leq 2(3 - 2a\Delta t)$, ce qui équivaut à $\sqrt{1 + 2a\Delta t} \leq 1 - 2a\Delta t$, ce qui est toujours vrai lorsque a est négatif.
- soit $\Delta < 0$, auquel cas les deux racines sont complexes conjuguées et il suffit de prouver que leur module est inférieur ou égal à un pour avoir la stabilité. Or le carré du module est égal au produit des racines, qui vaut $\frac{1}{3-2a\Delta t}$. Or a est négatif, et donc cette quantité est strictement inférieure à 1/3, et donc à 1.

Exercice 3 – **Implémentation en Langage C**

a) Ecrire une fonction crank_nicolson *qui prendra comme arguments :*

`le nombre de pas de temps,`
`la valeur du pas de temps,`
`la valeur de la condition initiale,`
`le coefficient` a `de l'équation,`

et qui renverra comme valeur l'approximation de $y(N\Delta t)$ *par l'algorithme de Crank-Nicolson.*

Comme lors de la séance précédente, on prendra soin d'écrire deux fichiers : l'un (crank_nicolson.h) avec la déclaration de la fonction et l'autre (crank_nicolson.c) qui implémentera effectivement cette fonction.

b) Ecrire une fonction bdf_ordre2 *qui aura les mêmes arguments que ci-dessus et qui renverra comme valeur l'approximation de* $y(N\Delta t)$ *par l'algorithme BDF d'ordre deux.* Attention : *vous devrez dans cette fonction faire appel à une autre méthode pour calculer* y_1*. Vous pourrez par exemple utiliser l'une des trois autres méthodes que vous avez déjà programmées.*

c) Reprendre votre programme principal main.c et le modifier de telle sorte qu'il réponde au souhait suivant :

On souhaite étudier la convergence numérique de chacune des méthodes lorsque le pas de temps tend vers 0*. Pour cela on va fixer le temps final* T *de la simulation et faire les calculs approchés en prenant* N*, puis* $2N$*, puis* $4N$ *..., jusqu'à* $2^k N$ *pas de temps, où* k *sera un entier lu dans le fichier de données.*

On souhaite sauvegarder les résultats des calculs en écrivant dans un fichier, pour chacun des pas de temps (sur une nouvelle ligne pour chaque pas de temps), dans cet ordre :

`la valeur du pas de temps,`

la valeur absolue de l'erreur entre la solution exacte au temps T et son approximation par le schéma d'Euler explicite,
la valeur absolue de l'erreur entre la solution exacte au temps T et son approximation par le schéma d'Euler implicite,
la valeur absolue de l'erreur entre la solution exacte au temps T et son approximation par le schéma de rank-Nicolson
la valeur absolue de l'erreur entre la solution exacte au temps T et son approximation par le schéma BDF d'odre deux

On pourra utiliser gnuplot *pour visualiser, en échelle* log-log *l'évolution de l'erreur pour ces schémas. Retrouve-t-on les ordres attendus ? (NB : Pour utiliser* gnuplot*, après avoir tapé la commande* `gnuplot` *puis* `set logscale` *pour passer en échelle* log-log*, taper par exemple*
`plot "fichier_resultats" using 1:2 with lines , "fichier_resultats" using 1:3 with lines`

On donne les indications suivantes pour la lecture du fichier de données et l'ouverture du fichier de résultat dans lequel on écrira les résultats :

```
---------------- main_fich.c -----------------------


#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "euler_imp.h"
#include "euler_exp.h"
#include "crank_nicolson.h"
#include "bdf_ordre2.h"


int main(int argc, char *argv[])

{

    if (argc != 3) {

    printf("\nLe programme doit recevoir deux et seulement deux arguments\n");

    return(EXIT_FAILURE);

    }

int N ;
int k;
double temps_final;
double a ;
double y0 ;


FILE* fichier_lec = NULL;
FILE* fichier_ecr = NULL;


fichier_lec = fopen(argv[1],"r");  // ouverture du fichier de donnees en lecture
```

```
    if (fichier_lec == NULL)
    {
         printf("Impossible d'ouvrir le fichier %s \n",argv[1]);
         return(EXIT_FAILURE);
    }

fichier_ecr = fopen(argv[2],"w+"); // ouverture (et creation eventuelle) du fichier
                                   // de resultats en ecriture


fscanf(fichier_lec, "%d %d %lf %lf %lf", &N , &k,  &temps_final, &a , &y0);
```

Pour écrire dans le fichier de résultats, on utilisera la commande

```
fprintf(fichier_ecr, "%g %g %g %g %g \n", dt, e_exp, e_imp, e_cni, e_bdf);
// on prefere le format %g qui ecrit les nombres en format scientifique: x.xxxxxexx
```

où `dt` *est la valeur du pas de temps, et* `e_exp, e_imp, e_cni, e_bdf` *sont les valeurs absolues des erreurs entre la solution exacte au temps T et son approximation par les différents schémas.*

On n'oubliera pas, en fin de programme main, de fermer les fichiers avec la commande

```
fclose(fichier_lec);
fclose(fichier_ecr);
```

Le fichier `fichier_de_donnees` *devra comprendre : deux entiers, suivi de trois doubles.*

La ligne de commande sera `./executable fichier_de_donnees fichier_de_resultats`
