Analyse Numérique

TP 3 : Résolutions numériques d'équations différentielles ordinaires

Dans ce TP, on souhaite approcher par différentes méthodes la solution du problème de Cauchy suivant :

$$\begin{cases} y'(t) = f(t, y(t)), & \forall t \in [0, T] \\ y(0) = y_0 \end{cases}$$

Pour cela on considère une discrétisation uniforme de l'intervalle $[0, T]$ de pas $h = \frac{T}{N}$ et donnée par $t = (t_0, \ldots, t_N)$ avec $t_{n+1} = t_n + h$. Pour valider nos méthodes numériques on considèrera les fonctions suivantes :

$$f_1(t, x) = -x, \quad f_2(t, x) = -2tx,$$

qui correspondront aux équations

$$y'(t) = -y(t), \quad y'(t) = -2ty(t).$$

Les solutions exactes associées sont les fonctions

$$y_1(t) = \exp(-t), \quad y_2(t) = \exp(-t^2)$$

En effet, vous pouvez vérifier par un calcul de dérivées que $y_1'(t) = -y_1(t) = f_1(t, y_1(t))$ et $y_2'(t) = -2ty_2(t) = f_2(t, y_2(t))$.

Calcul d'erreur sur l'intervalle $[0, T]$ **:** Dans la suite on aura d'une part y un vecteur solution approchée $y = (y_1, \ldots, y_N)$ et d'autre part z le vecteur solution exacte discrétisée $z = (z_1, \ldots, z_N)$. On choisit comme manière de calculer l'erreur la norme infinie, c'est à dire :

$$err = \|y - z\|_\infty = \max_{i \in \{1,\ldots,N\}} |(y - z)_i|$$

Cette erreur indique la valeur maximale des écarts entre les fonctions exactes et approchées associées sur l'intervalle $[0, T]$

Méthodes explicites

On rappelle la formule du schéma d'Euler explicite

$$y_{n+1} = y_n + hf(t_n, y_n), \quad y_0 \text{ donné.}$$

1. Créer un répertoire '/tp3edo' et créer à l'intérieur un fichier $main.m$. Implémenter la fonction *eulerExp.m* qui prend en entrée un réel $tfin$, une fonction f, un entier N et un réel $ydebut$ et en sortie renvoie deux vecteurs de taille N+1 y résultat du schéma d'Euler explicite et t les temps discrétisés. On utilisera la syntaxe suivante :

```
function [y,t]=eulerExp(tfin,f,N,ydebut)
 y=?;
 t=?;
 y(1)=?
 h=?
 for i=1:?
  tt=?;
  yy=?;
  y(i+1)=yy+h*f(tt,yy);
  t(i+1)=tt+h;
 end
end
```

On testera la fonction avec les lignes suivantes.

```
f=@(t,y) -y;
N=32;
ydebut=1;
tfin=20;
disp("temps final")
disp(tfin)
disp("pas de discretisation")
disp(tfin/N)
%calcul Euler explicite
[y,t]=eulerExp(tfin,f,N,ydebut);
%solution exacte
yex=exp(-t);
%calcul erreur
err=?;
disp("erreur Euler explicite")
disp(err)
plot(t,y,"b",t,yex,"r");
legend("euler exp", "exacte")
```

Au niveau de l'erreur, testez des valeurs de $N = 32, 64, 128\ldots$, l'erreur doit tendre vers 0. Au niveau graphique, pour $N = 16$, vous devriez obtenir la courbe de la colonne gauche du graphique donné dans la suite de l'énoncé.

2. On rappelle les formules associées au schéma Runge Kutta 2 (RK2) :

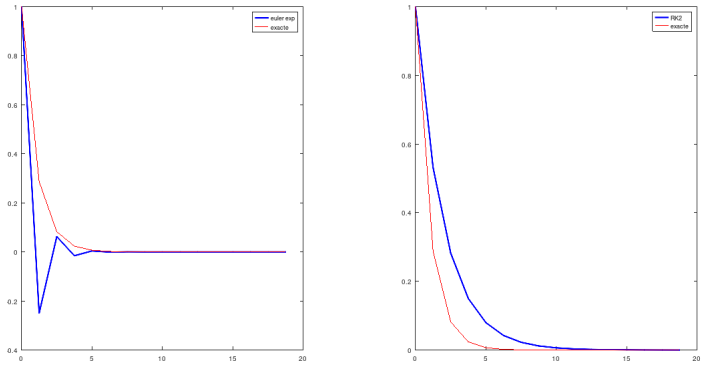
$$\begin{aligned} y_{n+1} &= y_n + hf(t_{n+1/2}, y_{n+1/2}) \\ y_{n+1/2} &= y_n + hf(t_n, y_n) \\ t_{n+1/2} &= t_n + \tfrac{h}{2} \end{aligned}$$

A partir de votre fonction *eulerExp.m*, sauvegardez la comme *rk2.m* afin de définir une nouvelle fonciton *[y,t]=rk2(tfin,f,N,ydebut)* qui prend les mêmes entrées que précemment et en sortie renvoie y, résultat du schéma RK2 et t les temps discrétisés. On introduira deux nouvelles variables *thalf* et *yhalf* et pour la mise à jour dans la boucle *for*, on utilisera la syntaxe suivante :

```
y(i)=yy+h*f(thalf,yhalf);
```

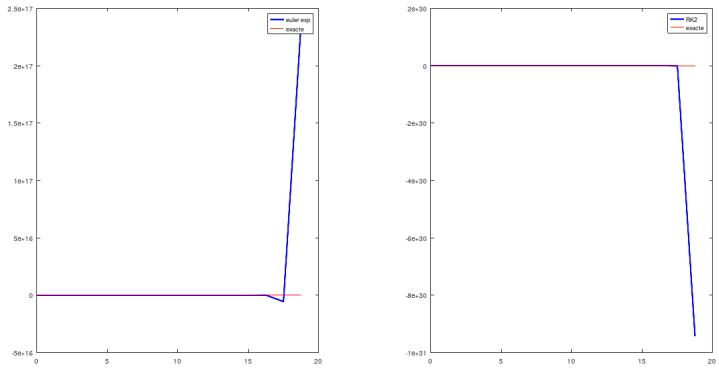
Pour tester cette fonction, reprenenez votre fichier *main.m* d'Euler explicite et ajoutez un test pour RK2, utilisez *subplot(1,2,1)*, *subplot(1,2,2)* pour afficher les résultats Explicite et RK2 sur deux colonnes.

Au niveau erreur, vous devriez obtenir une erreur pour la méthode RK2 inférieure à la méthode Euler explicite car la théorie qui nous dit que RK2 est un schéma ordre 2 et EE d'ordre 1. Au niveau graphique, pour $N = 16$ vous devriez obtenir les courbes suivantes



Vérifiez qu'avec $N = 32$ le pic négatif de la solution approchée par la méthode d'Euler explicite a disparu.

3. Sauvegardez votre fichier $main.m$ comme un nouveau fichier $testGaussienne.m$ et modifiez la fonction associée à l'équation différentielle $f(t,y) = -2ty$ et la solution exacte associée qui sera alors $y(t) = \exp(-t^2)$. Pour $N = 16, 32, 64, 128$, constatez que les solutions données par Euler explicite et RK2 explosent. Par exemple pour $N = 16$, vous devriez obtenir les courbes suivantes :



Vérifiez que pour $N = 256$, vous retrouvez des solutions approchées convenables.

Conclusions sur les méthodes explicites On observe ici l'existence d'un seuil h_c tel que

- ▷ Si $h < h_c$, alors les méthodes explicites sont stables,
- ▷ Si $h \geq h_c$, alors les résultats explosent.

Dans la suite, nous allons nous intéresser à des méthodes implicites, qui elles n'auront pas de restrictions sur h pour avoir de la stabilité.

Méthode implicite

On rappelle le schéma d'Euler implicite

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$$

Ce schéma se réécrit de la manière suivante :

$$g(y_{n+1}) = 0, \quad \text{avec } g(y) = y - hf(t_{n+1}, y) - y_n.$$

On a donc besoin d'une méthode qui permette de trouver les zéros d'une fonction. Une méthode classique et efficace s'appelle la méthode de Newton.

Méthode de Newton On cherche à résoudre numériquement $f(x) = 0$. La méthode de Newton consiste à calculer la suite numérique suivante :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad x_0 \text{ donné arbitrairement.}$$

En pratique pour implémenter cet algorithme on se fixe un seuil de précision ε et on s'arrête lorsque x_{n+1} satisfait la condition suivante :

$$|x_{n+1} - x_n| < \varepsilon.$$

1. Dans une fichier *newton.m*, définir une fonction *z=newton(crit,xinit,f,df)*, qui prend en entrées deux réels *crit,xinit* et deux fonctions *f, df*. En sortie elle renvoie *z*, un zéro de la fonction f calculé par la méthode de Newton. On utilisera la syntaxe suivante :

```
function z=newton(crit,xinit,f,df)
 x=?;
 xx=?;
 err=abs(xx-x);
 while (?)
  x=?;
  xx=?;
  err=abs(xx-x);
 end
 z=xx;
end
```

Pour tester cette fonction, écrire un script *testNewton.m* en utilisant la syntaxe suivante :

```
crit=0.9                          xinit=0
f=@(x) (x-1)*(x-2); %x^2-3x+2     y=newton(crit,xinit,f,df)
df=@(x) 2*x-3                     xinit=3
                                  y=newton(crit,xinit,f,df)
```

Testez les valeurs de $crit = 0.9, 0.5, 0.1, 0.01$, vous devriez converger vers les valeurs des racines 1 et 2 du polynôme $(x-1)(x-2)$.

2. On rappelle que la fonction g associée à la méthode d'Euler implicite est définie de la manière suivante : $g(y) = y - hf(t_{n+1}, y) - y_n$. On suppose également connue la dérivée de $f(t,y)$ par rapport à la variable y, qu'on note $df(t,y)$. Définir une fonction *eulerImp.m* qui prend en entrée un réel *tfin*, deux fonctions *f, df*, un entier N et un réel *ydebut*. En sortie elle renvoie y calculée par la méthode d'Euler implicite et t le vecteur des temps discrétisés associé. Vous utiliserez la syntaxe suivante :

```
function [y,t]=eulerImp(tfin,f,df,N,ydebut)
y=zeros(1,N+1);
t=zeros(1,N+1);
y(1)=ydebut;
h=tfin/N;
for i=1:N
 yy=y(i);
 tt=t(i);
 g=@(y) ?-h*f(?,?)-?;
 dg=@(y) ?-h*df(?,?);
 crit=1e-6
 xinit=yy;
 y(i+1)=?
 t(i+1)=tt+h;
end
end
```

Pour tester cette fonction, dans un script *testImplicite.m* faîtes un test similaire à celui d'Euler explicite, à la fonction *f=@(t,y) -y;* il vous faudra ajouter la dérivée par rapport à la seconde variable y de f qu'on déclarera comme *df=@(t,y) ? ;*.

Au niveau erreur, testez avec $N = 16, 32, 64 \ldots$, l'erreur doit tendre vers 0. Au niveau graphique, pour $N = 16$ vous devriez obtenir la courbe donnée ci-après dans le colonne de gauche des deux graphiques donnés ci-après.

3. On rappelle les formules du schéma des trapèzes

$$y_{n+1} = y_n + \frac{h}{2}\left(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right)$$

Donnez en commentaire la fonction g et da dérivée dg qui permettent de réécrire le schéma sous la forme $g(y_{n+1}) = 0$.

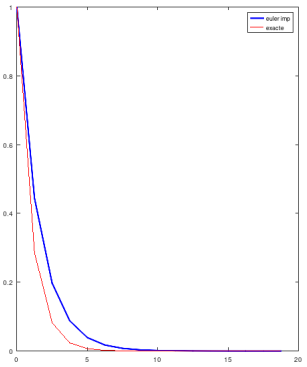
```
%{
Effacez cette ligne et remplacez les ? par tn,tn+1 ou yn
g(y)= ?-0.5*h*f(?,?)-?-0.5*h*f(?,?)
dg(y)= ?-0.5*h*df(?,?)
%}
```

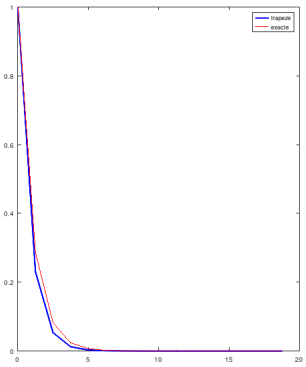
4. A partir de votre fonction *eulerImp.m*, sauvegardez la comme *trapeze.m* afin de définir une nouvelle fonciton *[y,t]=trapeze(tfin,f,df,N,ydebut)* qui prend les mêmes entrées que précemment et en sortie renvoie y, résultat du schéma des trapèzes et t les temps discrétisés. Les seules variables à modifier sont g et dg

```
g=@(y) ?-0.5*h*f(?,?)-?-0.5*h*f(?,?);
dg=@(y) ?-0.5*h*df(?,?);
```

on utilisera la question précédente pour remplacer les ? par y, yy, tt ou $tt+h$.

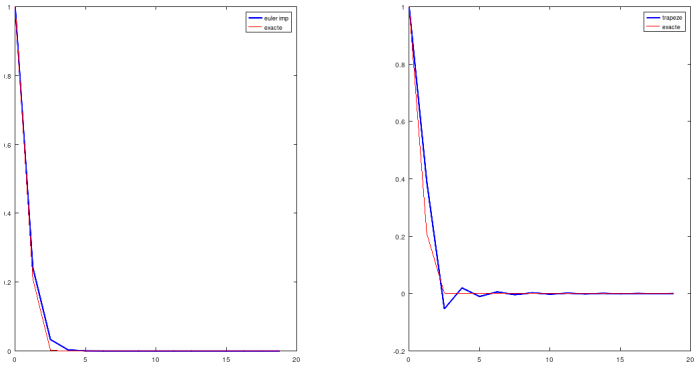
Pour tester cette fonction, reprendre le test d'Euler implicite et ajoutez un test pour le schéma des trapèzes, vous devriez obtenir des erreurs plus petites pour les trapèzes car la théorie nous dit que le schéma des trapèzes est un schéma ordre 2 et le schéma Euler implicite d'ordre 1. Au niveau graphique, pour $N = 16$ vous devriez obtenir les courbes suivantes :





On remarque que pour ces mêmes paramètres Euler explicite présentait un pic négatif, ce qui n'est plus le cas pour Euler implicite.

5. Sauvegardez votre fichier $testImpliciteTrapeze.m$ comme un nouveau fichier $testGaussienneImpliciteTrapeze.m$ et modifiez la fonction associée à l'équation différentielle $f(t,y) = -2ty$, mettre à jour la dérivée de f par rapport à y, $df(t,y) = -2t$ et la solution exacte associée qui sera alors $y(t) = \exp(-t^2)$.
   Testez les paramètres $tfin = 20$ et $N = 16$, vous devriez obtenir les courbes suivantes :



   Pour ce même choix de paramètres en explicite les résultats explosaient !

Conclusions pour les méthodes implicites Nous avons constaté que les deux méthodes implicites donnent des résultats satisfaisant pour un choix de paramètres où les méthodes explicites explosaient. Comme rien n'est parfait, ces méthodes implicites ont pour inconvénient de devoir trouver les zéros d'une fonction par méthode de Newton. Dans certaines cas cette étape peut s'avérer coûteuse en temps de calcul voire impossible si la méthode de Newton est mise en échec. Mais cela dépasse largement le cadre de ce cours et relève de la recherche actuelle en analyse numérique, où l'on cherche de nouvelles méthodes adaptées à des problèmes compliqués où les méthodes présentées ici ne seraient pas satisfaisantes.