Licence Renforcée L1 PC-SI, Équations différentielles
Feuille TP 1 – version 2
Thierry Champion

Dans ce TP on aborde les méthodes d'Euler explicite et implicite pour résoudre des équations différentielles ordinaires d'ordre 1, et plus précisément pour résoudre le problème de Cauchy

$$(C) \qquad \begin{cases} y(a) = y_a, \\ y' = f(t,y) \quad \text{sur } [a,b] \end{cases}$$

qu'on programmera sous Octave ou Python (cette fiche étant plutôt rédigée pour Octave).

Quelques instructions avant de commencer :

— Créez un dossier correspondant à ce module (par exemple **TP-R21**) : c'est dans ce dossier que vous sauverez les scripts correspondants à ce TP.

— Rédigez au fur et à mesure vos commentaires, remarques et réflexions sur les résultats obtenus.

1 Méthode d'Euler explicite

Le but de cette section est de programmer une fonction

```
[x,u]=euler_explicite(f,ya,a,b,n)
```

qui emploie le schéma d'Euler explicite pour résoudre le problème de Cauchy (C) en découpant l'intervalle $[a,b]$ en n intervalles $[x_i, x_{i+1}]$, d'extrémités $x_i := a + (i-1)\,\frac{b-a}{n}$ pour $i = 1, \ldots, n+1$. Il y a donc $n+1$ points de discrétisation :

$$x_1 = a,\ x_2,\ x_3,\ \ldots,\ x_n, x_{n+1} = b.$$

La donnée initiale y_a sera représentée par la variable `ya`, le vecteur ligne `x` doit contenir les points x_i pour $i = 1, \ldots, n+1$ et le vecteur `u` doit contenir les valeurs $u_i = u(x_i)$ de la solution approchée en ces points. On rappelle que ces valeurs approchées sont définies par la formule de récurrence suivante :

$$\begin{cases} u_1 = y_a \\ \forall i \geq 1, \quad u_{i+1} = u_i + h\, f(x_i, u_i) \end{cases} \tag{1}$$

où $h = \frac{b-a}{n}$ est le pas de discrisation.

1.1 Résolution exacte pour un cas simple

Résoudre le problème de Cauchy (C) dans le cas où $f(t,y) = f_1(y) := y$ et $y_a = 1$, c'est-à-dire dans le cas

$$(C) \qquad \begin{cases} y(a) = y_a, \\ y' = y \quad \text{sur } [a,b] \end{cases}$$

1.2 Quelques éléments de syntaxe Octave

Pour définir le vecteur $x = (x_1, \ldots, x_{n+1})$ des $n+1$ points de discrétisation de l'intervalle $[a,b]$, il suffit d'employer la commande :

```
a=0;
b=1;
n=10;
x=linspace(a,b,n+1);
```

Si $y = (y_1, \ldots, y_{n+1})$ est le vecteur des valeurs d'une fonction g aux abscisses x_i, c'est-à-dire $y_i = g(x_i)$ pour tout i, alors pour tracer le graphe de la fonction affine par morceaux qui passe par les points (x_i, y_i) il suffit d'exécuter la commande

```
plot(x,y,'r-');
```

La commande ci-dessus tracera le graphe en rouge (`r` pour "red") et en trait continu (c'est le `-`). Pour tracer le graphe en pointillés verts, il suffit de remplacer `'r-'` par `'g--'` (`g` pour "green"). Utiliser `b` pour tracer en bleu, et `k` pour tracer en noir ("black").

Pour tracer le graphe de la fonction $f : t \mapsto t^2$ sur $[0, 1]$, on pourra exécuter les commandes

```
# vecteur des abscisses
a=0;
b=1;
n=100;
x=linspace(a,b,n+1);
# definition de f et vecteur des ordonnees des valeurs de f
f=@(t)[t.**2];
y=f(x)
# courbe
plot(x,y,'r-');
```

Dans les instructions précédentes, on aurait pu remplacer la commande `y=f(x)` qui définit le vecteur `y` en appliquant `f` à toutes les coordonnées de `x` par la boucle suivante (plus longue à taper) :

```
# definition d'un verteur nul de taille n+1
y=zeros(1,n+1)
# boucle pour fixer les coordonnees de y
for i=1:n+1
  y(i)=f(x(i))
  endfor
```

1.3 Retour au problème de Cauchy pour la fonction f_1

Programmer la fonction $f(t, y) = f_1(y) = y$ du paragraphe 1.1 sous Octave, puis programmer la fonction `euler_explicite` demandée, en vous appuyant sur les commandes vues précédemment.

Utiliser la fonction `euler_explicite` pour résoudre numériquement (C) avec la fonction f_1 sur l'intervalle $[0, 2]$ pour les valeurs $y_a = 1$ et $n = 2^k$ avec $k \in \{2, \ldots, 10\}$. Pour chaque n, tracer les courbes représentatives des solutions approchées trouvées et de la solution exacte.

Calculer aussi l'erreur $|y(2) - u(2)|$ entre la valeur $y(2)$ pour la solution exacte et celle obtenue par la méthode d'Euler $u(2)$. Tracer le graphe de cette erreur (comme fonction de n) en échelle logarithmique.

1.4 Autres essais

Tester à nouveau votre programme avec la fonction $f(t,y) = f_2(t,y) := -y/(t+1)$ sur l'intervalle $[0,1]$ avec $y_a = 1/2$, en traçant les graphes et en comparant les valeurs obtenues en $t = 1$ avec la valeur de la solution exacte (on doit pour cela résoudre (C) dans ce cas).

2 Méthode d'Euler implicite - calcul exact

Le but de cette section est de programmer deux fonctions

```
[x,u]=euler_implicitef1(ya,a,b,n), [x,u]=euler_implicitef2(ya,a,b,n)
```

qui emploient chacune le schéma d'Euler implicite pour résoudre le problème de Cauchy (C) pour chacune des deux fonctions f_1 et f_2 de la section 1. On rappelle que ce schéma consiste à construire les valeurs approchées $u_i = u(x_i)$ par la formule de récurrence suivante :

$$\begin{cases} u_1 = y_a \\ \forall i \geq 1, \quad u_{i+1} = u_i + h\, f(x_{i+1}, u_{i+1}) \end{cases} \tag{2}$$

pour laquelle on doit donc résoudre à chaque étape l'équation *a priori* non linéaire

$$u_{i+1} = u_i + h\, f(x_{i+1}, u_{i+1}) \tag{3}$$

Or pour chacune des deux fonctions f_1 et f_2 on peut trouver une formule pour la solution u_{n+1} de cette équation. C'est ce qu'on va utiliser pour écrire la méthode d'Euleur implicite dans ces deux cas.

1. Pour chacune des deux fonctions f_1 et f_2, résoudre l'équation (3).
2. Programmer les deux méthodes `euler_implicitefi` pour $i = 1$ et $i = 2$ correspondantes.
3. Comparer les résultats avec ceux obtenus à la section 1 pour différentes valeurs de n (tracer les courbes obtenues pour les deux méthodes et la courbe exacte, et comparer les valeurs au temps terminal)

3 Méthode d'Euler implicite - cas général

Le but de cette section est de programmer la fonction

```
[x,u]=euler_implicite(f,ya,a,b,n)
```

qui emploie le schéma d'Euler implicite (2) rappeler ci-dessus pour résoudre le problème de Cauchy (C) dans le cas où on ne sait pas forcément résoudre exactement l'équation non-linéaire (3). Pour cela, on utilisera la fonction Octave `fzero` pour résoudre cette équation.

3.1 Résolution d'une équation non linéaire

Exécuter le script suivant sous Octave :

```
# definition de la fonction g(x)=x^2-2
g=@(x)[x.**2-2]
# resolution de g(x)=0 en partant du point 1
sol=fzero(g,1);
#affichage de la solution
```

```
sol
# resolution de g(x)=0 en partant du point -1 et affichage
sol=fzero(g,-1);
sol
```

La fonction `fzero` a deux arguments : une fonction g (dans le code ci-dessus notée `g`) et un point de départ x_0 (ci-dessus les valeurs 1 puis -1). Elle doit normalement retourner une solution l'équation $g(x) = 0$ proche de x_0. C'est bien ce qu'on observe ci-dessus.

En utilisant la fonction `fzero`, résoudre l'équation $sin(x) = x$. Plus précisément, il faut trouver toutes les solutions, en variant le point de départ x_0.

3.2 méthode d'Euler implicite - premiers tests

Programmer la fonction `euler_implicite`.

Tester cette fonction sur les fonctions f_1 et f_2 de la section 1, et comparer les résultats avec ceux des sections précédentes.

3.3 méthode d'Euler implicite - autre exemple

Utiliser cette méthode pour résoudre le problème de Cauchy (C) avec les données

$$f(t, y) = \frac{t+1}{t+2}\, y\,(2 - y), \qquad y_a = 0.1\,.$$

Cette solution se rapproche-t-elle de la solution du problème de Cauchy

$$f(t, y) = y\,(2 - y), \qquad y_a = 0.1\,.$$

pour de "grandes valeurs" du temps t ? (on se placera sur un intervalle du type $[n, n + 1]$ avec n "grand")