Résolution numériques des équations différentielles - II

Objectifs :

- étudier expérimentalement la stabilité de la méthode d'Euler pour une équation raide
- programmer la méthodes d'Euler implicite

1 Stabilité du schéma d'Euler

On considère l'équation différentielle suivante :

$$\begin{cases} y' = -150y + 30 \\ y(0) = \dfrac{1}{5} \end{cases} \tag{1}$$

La solution exacte de cette équation est $y(x) = \dfrac{1}{5}$. Si au lieu de prendre comme condition initiale $y(0) = \dfrac{1}{5}$ on perturbe la condition initiale en prenant $y(0) = \dfrac{1}{5} + \varepsilon$ alors la solution exacte devient $y(x) = \dfrac{1}{5} + \varepsilon e^{-150x}$. Pour ε petit cette nouvelle solution est très proche de celle trouvée avec la condition initiale non perturbée. Une méthode numérique est dite stable si l'on retrouve cette propriété au niveau de la solution numérique : la solution numérique trouvée pour la condition initiale $y(0) = \dfrac{1}{5} + \varepsilon$ doit être proche de celle trouvée pour la condition initiale $y(0) = \dfrac{1}{5}$.

Nous allons tester la stabilité du schéma d'Euler explicite sur cette équation.

1. Appliquer le schéma d'Euler explicite (déjà programmé) à l'équation (1) sur l'intervalle $[0, 1]$. Tracer le graphe de la solution obtenue.
2. Appliquer le schéma d'Euler explicite à l'équation précédente en utilisant la condition initiale $y(0) = \dfrac{1}{5} + \varepsilon$ avec $\varepsilon = 10^{-10}$ pour plusieurs valeurs de n (nombre pas de discrétisation). Tracer les graphes des résultats obtenus. Qu'observe-t-on ?
3. Que peut-on déduire de la stabilité du schéma d'Euler dans ce cas ?

2 Schéma d'Euler implicite

Au lieu d'utiliser le schéma d'Euler explicite nous allons utiliser le schéma d'Euler implicite défini par :

$$y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}) \tag{2}$$

Pour l'équation étudiée nous avons :

$$f(x, y) = -150y + 30$$

La seule différence avec le schéma d'Euler est, qu'à chaque itération, nous devons résoudre l'équation implicite 2 pour obtenir y_{n+1}. Pour ce faire nous utiliserons la méthode de Newton.

1. La méthode de Newton permet de résoudre une équation du type $g(x) = 0$, ce qui n'est pas le cas de l'équation 2. Mettre l'équation 2 sous la forme $g(x) = 0$ et rappeler l'algorithme de Newton.
2. Programmer le schéma d'Euler implicite. Pour ce faire vous devrez rajouter trois fonctions matlab (ou scilab) au programme précédent :
   - une fonction `dfdy(x,y)` admettant `x` et `y` comme paramètres et renvoyant la valeur de $\dfrac{\partial f}{\partial y}$
   - une fonction `newton(f,dfdy,xn,yn,h,eps)` dont les paramètres sont ; la fonction f, `f`, sa dérivée par rapport à y, `dfdy`, le point de départ de l'algorithme de Newton (`xn`, `yn`), le pas d'intégration `h` et la précision demandée `eps` et qui renvoie la valeur approchée de la solution de 2.
   - une fonction `euler_implicite(f,dfdy,x0,y0,x1,n)` semblable à la fonction `euler_explicite(f,x0,y0,x1,n)` déjà programmée.
3. Etudier la stabilité du schéma d'Euler implicite. Que remarque-t-on ?

Indications 1. Vous complèterez les fonctions Scilab suivantes :

```
// Calcul de df/dy pour l'EDO y'=f(x,y)
// Paramètres d'entrée:
//    x valeur de la variable
//    y valeur de y(x)
// Valeur renvoyée:
//    df/dy (x,y)
function [val]=dfdy(x,y)
  val = ...;
endfunction

// Résolution par la méthode de Newton de
```

```
// l'équation  implicite associée au schéma d'Euler:
// Paramètres d'entrée:
//    f fonction f(x,y)
//    dfdy fonction df/dy (x,y)
//    xn,yn point de départ
//    h pas
//    eps précision
// Valeur renvoyée:
//    y valeur approchée de ...
function [y]=newton(f,dfdy,xn,yn,h,eps)
    y = yn;
    fn = ...;
    while (abs(fn)>eps) do
        fn = ...;
        dfn = ...;
        y = y  -...;
    end
endfunction

// Schéma d'Euler implicite pour l'EDO y'=f(x,y) avec
// y(x0) = y0 sur l'intervalle [x0, x1]
// et pas de temps fixe h=(x1-x0)/n
// y(xn+1) = y(xn) + hf(xn+1, y(xn+1))
// Paramètres d'entrée:
//    f fonction f(x,y)
//    x0 point de départ
//    y0 condition initiale y0=y(x0)
//    x1 point d'arrivée
//    n nombre de points de discrétisation
// Valeurs renvoyées:
//    x vecteur contenant les points de discrétisation xi
//    y vecteur contenant les valeurs de la solution
//      approch?e yi=y(xi)
function [x,y]=euler_implicite(f,dfdy,x0,y0,x1,n)
    // Initialisation des vecteurs x et y
    x=[x0 zeros(1,n)];
    y=[y0 zeros(1,n)];
    // Pas de temps
    h=(x1-x0)/n;
    // Précision
    eps=1e-8;
```

```
    // Calcul de y(xi)
    for i=1:n
        x(i+1)=x(i)+h;
        y(i+1)=...;
    end
endfunction
```