Université Paris-Saclay - Année 2023-2024
Cours accéléré d'analyse numérique - M2 AMS

1 Pour aller plus loin

1.1 Un exemple de méthode implicite : simulation du système de Van der Pol par la méthode d'Euler implicite en utilisant la méthode de Newton.

On considère le problème de Cauchy pour le système de Van der Pol :

$$\begin{cases} x'(t) = \varepsilon \left(x(t) - \frac{x(t)^3}{3} \right) + y(t), \\ y'(t) = -x(t), \end{cases}$$

de donnée initiale

$$x(0) = x_0, \quad y(0) = y_0.$$

Nous allons calculer une solution approchée de ce problème par la **méthode d'Euler implicite**, dont l'itération est donnée par

$$Y^{n+1} = Y^n + hf(t^{n+1}, Y^{n+1}).$$

Pour ce faire, nous allons utiliser la méthode de Newton pour résoudre l'équation non linéaire définissant l'itération de la méthode d'Euler implicite.

Le système de Van der Pol est un exemple d'un problème dit « raide ». Il s'agit de problèmes dont la solution présente d'importantes variations dans un intervalle de temps très court. Nous verrons qu'au contraire des méthodes explicites, les méthodes implicites se comportent en général bien vis à vis de problèmes dits « raides ».

Définition des fonctions du système.

Exercice 1.
Écrire le système de Van der Pol sous la forme

$$Y' = F(Y),$$

avec $Y = (x, y)^t$ et $F : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$. Définir la fonction F, ainsi que sa matrice Jacobienne DF en complétant les fonctions python suivantes (respecter la structure donnée : F retourne un array de taille 2 de composantes F_1 et F_2 et DF un tableau de taille 2×2 avec les dérivées partielles de F) .

```
# ep= A completer avec la valeur du parametre epsilon

def F(Y):
    x=Y[0]
    y=Y[1]
    #F1=... # A completer : premiere composante de F
    #F2=... # A completer : deuxieme composante de F
    return np.array([F1,F2])

def DF(Y):
    x=Y[0]
    y=Y[1]
    # DFL1=np.array([ ... , ... ]) # A completer : premiere ligne de la matrice
                                   # jacobienne de F en Y=(x,y)
    # DFL2=... # A completer : deuxieme ligne de la matrice jacobienne de F
    return np.array([DFL1,DFL2])
```

La méthode de Newton.

Soit $G : \mathbb{R}^n \longrightarrow \mathbb{R}^n$. La méthode de Newton pour résoudre l'équation $G(Y) = 0$ est la méthode itérative définie par la suite

$$Y_{n+1} = Y_n - (DG(Y_n))^{-1} G(Y_n),$$

avec Y_0 donné. Sous certaines conditions, si Y_0 est proche d'un zéro de G, la suite $\{Y_n\}_n$ converge vers ce zéro.

On va définir une fonction python qui calcule une solution approchée de l'équation $G(Y) = 0$ par la méthode de Newton. Une telle fonction doit dépendre de :

- une valeur *eps* définissant la tolérance avec laquelle on va calculer une solution de $G(Y) = 0$ (ici on va calculer une valeur approchée Y^N de Y tel que $G(Y) = 0$ vérifiant $|G(Y^N)| \leq eps$) ;
- la fonction G et sa matrice jacobienne DG ;
- l'initialisation Y_0 ;
- un nombre maximal *nmax* d'itérations à ne pas dépasser.

Exercice 2.
Compléter la fonction python suivante permettant de calculer une valeur approchée d'un zéro d'une fonction G par la méthode de Newton :

```
def Newton(eps,G,DG,Y0,nmax):
    tol=np.linalg.norm(G(Y0))
    n=1 # n = nombre d'iterations qui seront effectuees
    Y=Y0
    while (tol>eps and n<nmax):
        # A completer ....
    return n,Y
```

Pour implémenter la méthode d'Euler implicite pour le système de Van der Pol, on remarque que l'itération de la méthode s'écrit sous la forme

$$Y^{n+1} = Y^n + hF(Y^{n+1}).$$

On peut donc la re-écrire sous la forme

$$G_n(Y^{n+1}) = 0,$$

où $G_n(Y) = Y - Y^n - hF(Y)$. Pour implémenter le méthode d'Euler implicite, on doit donc à chaque itération trouver la solution Y de $G_n(Y) = 0$. La fonction G_n dépend de Y^n et elle change alors à chaque itération. On peut la définir à chaque itération à partir de la fonction F en utilisant la fonction *lambda* de python.

Exercice 3. Construire une fonction donnant la solution approché Y de l'EDO $Y' = F(Y)$, de condition initiale $y(t_0) = Y0$, associée à une discrétisation uniforme de pas h de l'intervalle $[t_0, tf]$, en suivant le modèle suivant :

```
def EI(F,t0,tf,Y0,h):
    #... A completer
    return Y # vecteur contenant les valeurs approchees
             # obtenues par la methode d'Euler implicite
```

Exercice 4. On considère $\varepsilon = 10,\ x_0 = 2,\ y_0 = 0$. Compléter votre code pour obtenir la solution approchée obtenue par la méthode d'Euler implicite dans l'intervalle $[0, 30]$ et avec un pas $h = 0.05$. Représenter la solution approchée x ainsi obtenue, en fonction du temps.

Dans un autre graphique, représenter dans le plan (x, y) les trajectoires des solutions approchées obtenues.

Exercice 5. Refaire l'exercice en utilisant les méthodes programmées auparavant. On pourra aussi utiliser un solveur d'équations différentielles ordinaires fournis par python, la fonction `odeint` du package `scipy.integrate` de python. Comparer les résultats obtenus.