TP3
Equations différentielles

7 décembre 2017

Réalisé par RAHARIJAONA Dylan et KUCAM Delphine

Encadré par Jean-Paul Cardinal

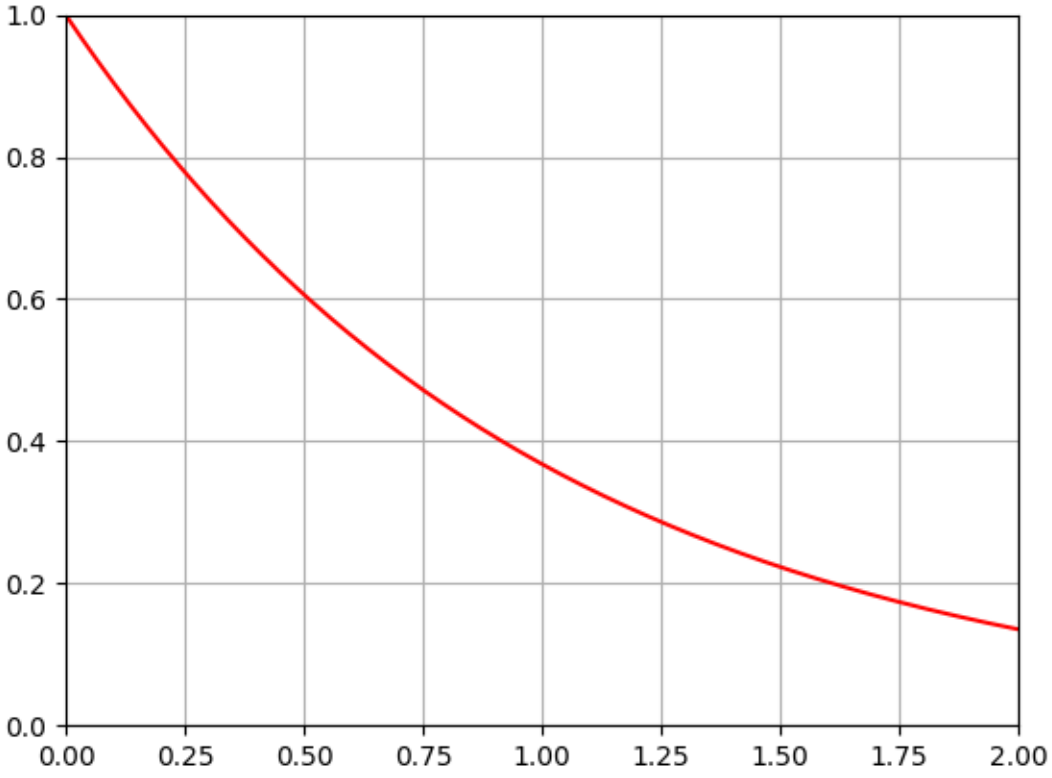
Le but de ce tp est de présenter plusieurs méthodes pour intégrer numériquement sur un intervalle $t \in [0, T]$, une équation différentielle ordinaire

1. Equation différentielle ordinaire

   (a) Dans cet exemple, f vaut -u(t).

   (b) Pour cette question, nous devons résoudre l'équation différentielle. Pour cela, nous avons fait quelques recherche sur Internet car nous avons oublié comment résoudre une équation différentielle. Par la suite, nous avons trouvé que u = exp(-t) puis nous avons dessiner son graphe sur l'intervalle $t \in [0, 2]$ que l'on vous a rendu en main propre.

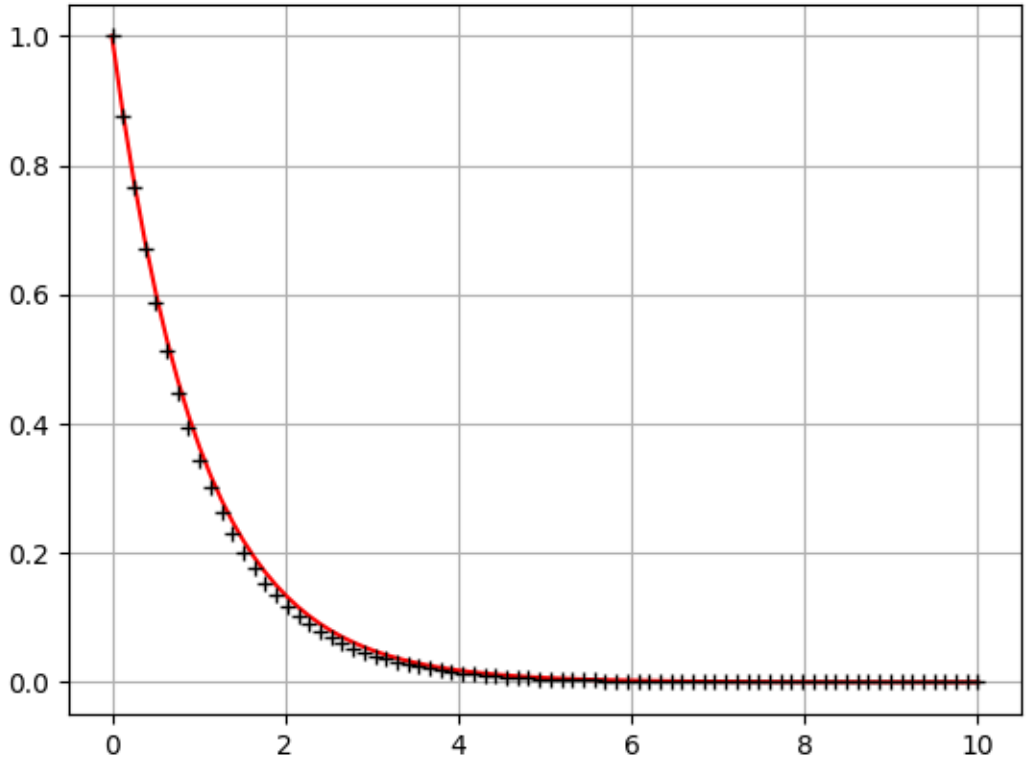
   (c) De même, nous avons représenté u(t) sur ordinateur à l'aide de matplotlib.
   Voici ce que l'on obtient :



2. Calcul approché de u au moyen de la méthode d'Euler
   (a) En calculant les $n+1$ premiers termes de la suite (u_k), on obtient :

$$u_{(k+1)} = u_0 + h * \sum_{i=0}^{k} f(t_i, u_i)$$

   (b) Nous avons représenté sur le même graphique la solution exacte u et les points (t_k, u_k) et nous avons obtenus :



En rouge, nous avons la solution de u et les points noirs correspondent aux points (t_k, u_k)

3. Euler
   Nous avons créé une fonction euler qui renvoie deux numpy.array tt et uu. Nous avons dû la modifiée plusieurs fois car il y avais plusieurs petites fautes d'inattention (corrigés immédiatement après).

4. (a) Voici la fonction f de l'exemple (2) :

```
def f(t, u) :
        return -u
```

   (b) Après avoir appliqué la fonction euler à l'exemple (2) (tt,uu=euler(f, u0, T, n)), on remarque que nous retrouvons les mêmes résultats que précedemment.

   (c) Nous avons résolus à la main les EDOs donnés, exceptionnelement la 3-ème EDO n'a pas pu être traité par souci de difficulté beaucoup trop élevé, ainsi nous avons approché le site WolframAlpha pour la calculer.

   Pour la première EDO, nous trouvons u(t) = $e^{-t} + t - 1$

   Pour la seconde EDO, nous trouvons u(t) = $-1/t^2$

   Pour la troisième EDO, nous avons regardé la réponse sur Internet.

5. Equation différentielle d'un ordre supérieur

   (a) Voici le résultat obtenu après avoir calculé à la main l'équation différentielle ordinaire : La solution de l'EDO est u(t) = $cos(wt)$

   (b) En prenant w=1.0, u_0=1.0, v_0 = 0.0, U_0 = np.array([u_0, v_0]), et U = U_0, voici la fonction F :

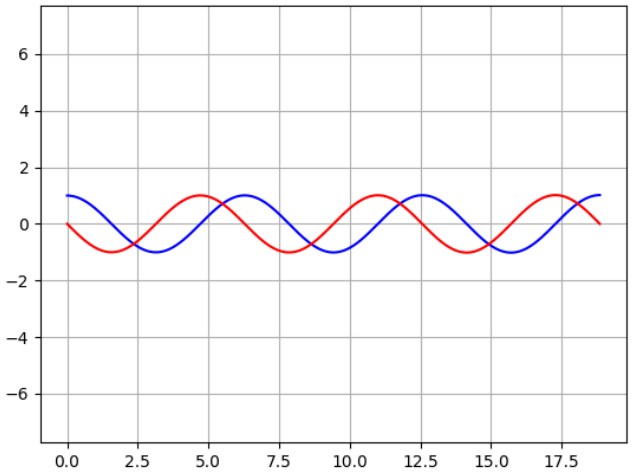
```
def F(t, U):
    w = 1.0
    u = U[0]
    v = U[1]
    U_out = np.zeros(2)
    U_out[0] = v
    U_out[1] = -w**2*u

    return U_out
```

   (c) Nous avons modifié la fonction euler, pour qu'elle prenne 4 arguments et retourne deux `numpy arrays`, tt contenant les valeurs t_k et UU contenant les valeurs U_k, obtenues par la méthode d'Euler. (Pour T = $6 * \pi$ et $n = 10000$)

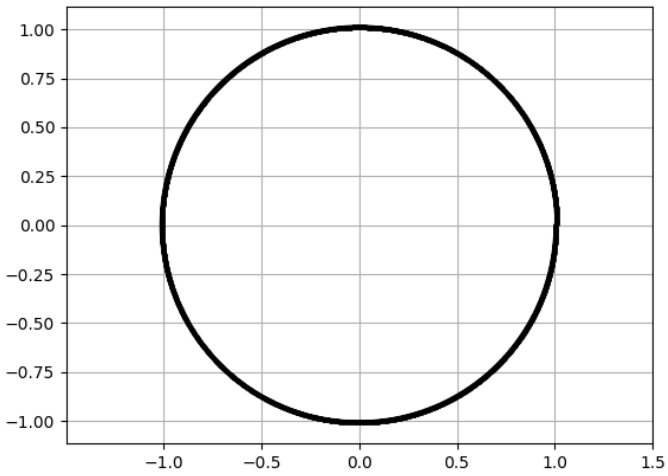
```
def euler2(F, U0, T, n):
        U = U0
        d = U0.shape[0]
        t = 0.0
        h = T/n
        tt = np.zeros(n)
        UU = np.zeros((d, n))
        for i in range(n):
                if(d==1):
                        UU = U
                else :
                        UU [:, i] = U
                tt[i] = t
                U = U + h*F(t,U)
                t+= h
        return tt, UU
```

(d) Nous devons résoudre numériquement l'EDO suivante : $u"(t) + \omega^2 u(t) = 0$. On trouve $u(t) = u'(0)sin(\omega x) + u(0)cos(\omega x)$ avec u'(0) = 0 et u(0) = 1.0 ce qui équivaut à $u(t) = cos(\omega x)$

(e) Lorsque nous avons édité les graphiques, nous avons mis 2 graphes sur le même graphique, ainsi nous avons seulement 2 graphiques :



Ce graphique représente la position u et la vitesse v en fonction du temps t.



Sur ce graphe, nous avons pris n = 10000, cependant plus n diminue plus le graphe ressemblera à un escargot en effet une erreur se propage quand n diminue.

Voyons un graphique où n = 200 :