TP2 : Equations différentielles

Équations	Solutions
$y' - y = 0$	$f(x) = k \cdot e^{x}$
$y' - ay = 0$	$f(x) = k \cdot e^{ax}$
$y' - ay - b = 0$	$f(x) = k \cdot e^{ax} - \frac{b}{a}$

Noms 1 : kentsa meli habib edgar
Noms 2 : hamdane amine
Nom du prof : Cardinal Jean-Paul
Année : 2017/2018

1 Introduction

Parvenu au terme du tp2 qui concernait l'intégration numérique, nous nous rendons au tp3 qui traite la résolution d'équation différentielle notamment par le calcul de valeurs approchées par la méthode d'Euler. Nous utiliserons aussi des méthodes vues précédemment tel que l'intégration numérique pour arriver à certains résultats.

2 Résolution d'équation différentielle

2.1

2.1.1 Valeur de f dans l'exemple

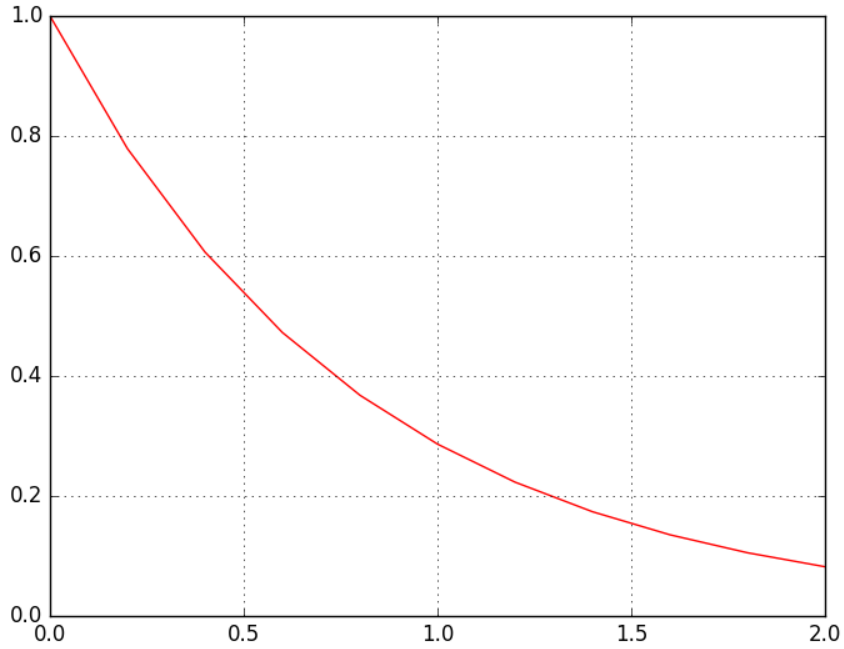
la fonction f vaut -u(t) dans cet exemple

2.1.2 Solution de l'EDO

Développé pas à on pas obtient la solution :
u'=-u <=> -(u'/u)=1 <=> ln u = t+cst <=> u= exp(t+cst)

2.1.3 Représentation graphique

nous avons représenté graphiquement u a l'aide de la fonction f en ouvrant la bibliothèque matplotlib, nous l'avons ensuite enregistré en format png sous le nom graphe1,png grâce a la commende plt.savefig() . Cela nous permet de sauvegarder notre figure dans nos documents ainsi nous pouvons co nsulter notre figure a tout moment sans le terminal ce qui nous evite d'utiliser la commende plt.show(). La représentation graphique de la fonction u est la suivante :



2.2 Méthode d'Euler

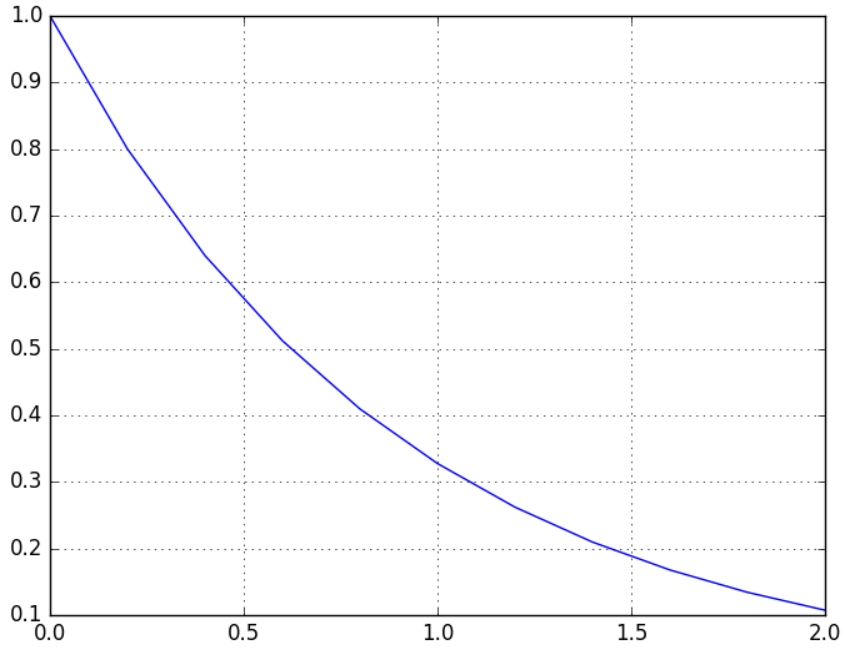
2.2.1 Calcul des n+1 premiers termes

A l'aide d'une boucle for nous avons représenté les n+1 premier termes de la suite u_k avec n= 10 et T=2, cela nous affiche une suite de nombre pour u_k que nous définissons de la manière suivante $u_{k+1} = u_k + hf(t_k, u_k)$ pour avoir toutes les valeur de u_k jusqu'à n. Nous bouclons donc avec une boucle for pour avoir u1,u2,u3 ..., u_n . Nous les stockons ensuite dans une liste U.

2.2.2 Représentation graphique de u

Comme nous l'avons expliqué dans la question précédente, nous avons stocké les points u_k dans une liste U et avec cette dernière nous avons tracé la fonction de la même manière que nous l'avons expliqué dans le 2.1.3. Nous

avons aussi sauvegardé notre graphique de la même manière. La représentation graphique de u est la suivante :



2.3 Écriture d'une fonction Euler

Nous avons créé un fichier tp3 fonction.py dans lequel nous allons écrire toutes nos fonctions. Notre fonction se présente sous la forme euler(f,u0,T,n) elle prend donc en argument une fonction f , une valeur initiale u0, un réel T et un entier n. Nous avons donc appris à renvoyer des numpy arrays dont nous ne connaissions pas l'existence et donc l'utilité. Cela nous a facilité la l'écriture de notre fonction euler.
L'implémentation de cette fonction est la suivante :

```
def euler(f,u0,T,n):
  h=float (T)/n
  xx = np.zeros(n)
  uu = np.zeros((n, 2))
  ui=u0
  xi=0.0
  for i in range (0,n):
    xi = xi + h
    ui += h*f(xi,ui)
    xx[i]=xi
    uu[i]=ui
  return xx,uu
```

2.4 Intégration d'EDO d'ordre supérieure

2.4.1 Écriture de la fonction F de l'EDO (5)

b) Nous avons écrit une fonction F qui prend en argument un flottant t et un numpy array U de taille 2. Dans le code, nous initialisons w a 1.0 , à u nous affectons la valeur de la première colonne du numpy array U et a v nous lui associons la 2eme colonne. Ensuite on crée un np.zeros de deux colonnes auquel on lui associe a la première colonne v et à la seconde $(-(w^2))u$. L'implémentation de la fonctio F est la suivante :

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

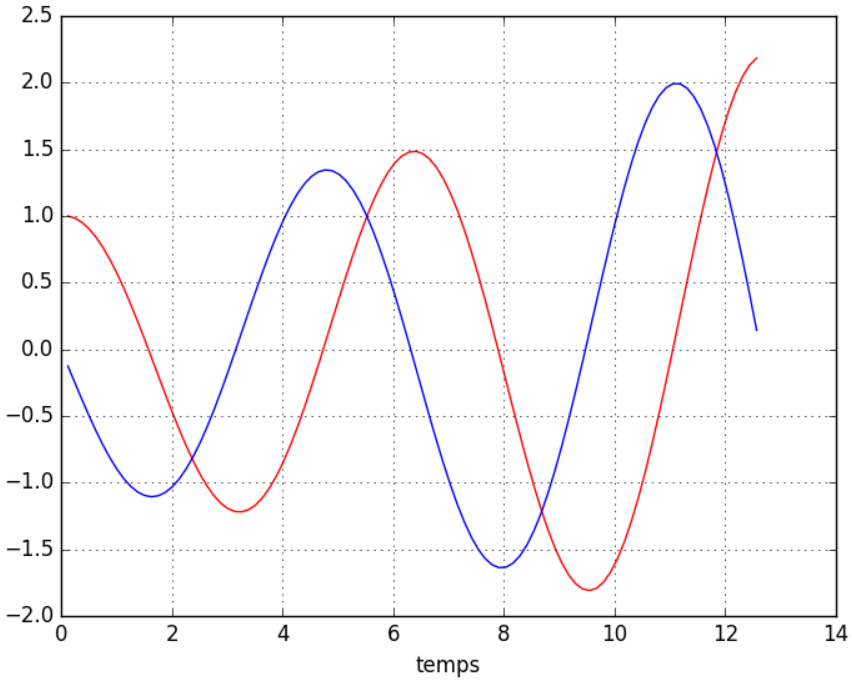
2.4.2 Fonction Euler2

Sur le même principe que Euler nous avons crée une fonction euleur2 qui prend cette fois ci une fonction de même type que notre fonction F. Cette fonction Euler nous renvoie deux numpy arrays tt de taille n+1 contenant les valeurs ti et UU de taille 2*(n+1) contenant les Ui obtenue par la méthode d'Euler.
L'implémentation de la fonction Euler2 est la suivante :

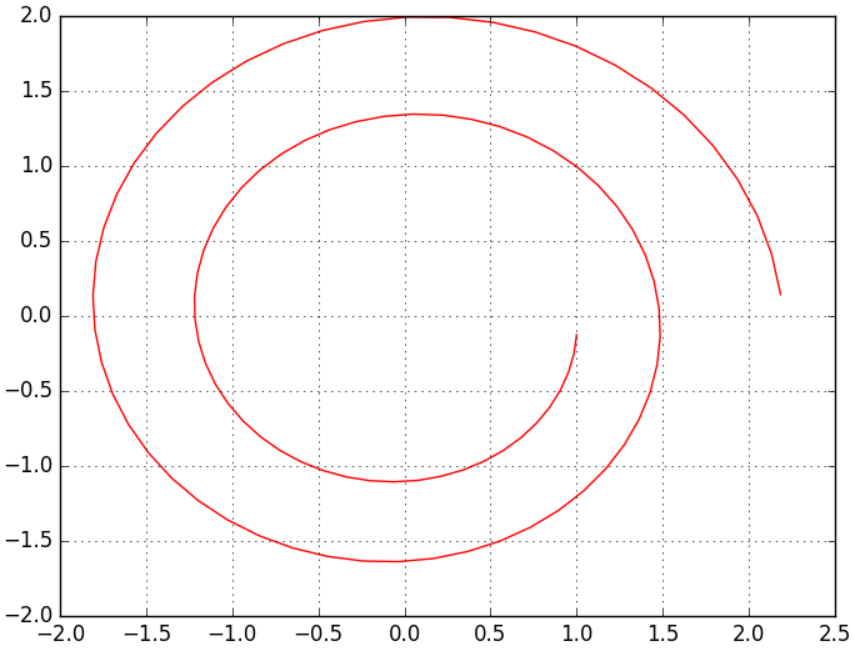
```
def euler2(F, U0, T, n):
  h = float(T)/n
  tt = np.zeros(n)
  UU = np.zeros((n, 2))
  ti = 0.0
  Ui = U0
  for i in range(n):
    ti = ti + h
    Ui += h*F(ti, Ui)

    print(Ui)
    tt[i] = ti
    UU[i, :] = Ui
  return tt, UU
```

2.4.3

Pour finir nous avons mis sur le même graphique les courbes représentants la position en fonction du temps en rouge et la vitesse en fonction du temps en bleu dans le graphe5 que ne voyons ci-dessous.



Ensuite nous avons tracé la position en fonction de la vitesse dans graphe6 cependant nous avons eu quelques problèmes au niveau de l'affichage des graphiques, toutes les courbes ce mélangeaient et se mettait sur le même graphe nous avons donc utiliser un plt.clf() avant chaque plot afin que les plot ne se mélangent pas. On peut remarquer qu'en augmentant n la spiral du graphique se resserre de plus en plus. Le graphe est le suivant :



3 Conclusion

À l'issue de ce tp nous avons pu réviser le principe de l'équation différentiel et de son utilité dans la mécanique en dehors des heures de tp afin de mieux comprendre le sujet du tp.
En revenant sur ce qui nous concerne(python), nous avons appris a à résoudre des equations différentielles d'ordre 1 et d'ordre supérieures à 1 grace à la méthode d'Euler et graces à d'autres fonctions utilisants la fonction Euler.