Binôme 1 : _____ Binôme 2 : _____

TP 7 : Capacités numériques : Résolution numérique d'équations différentielles et filtrage

Vous savez résoudre une équation différentielle linéaire d'ordre 1 de façon analytique. Il est aussi possible de le faire numériquement. Cette méthode numérique est tout particulièrement utile pour étudier des systèmes dont l'évolution est dirigée par des équations différentielles que l'on ne sait pas résoudre analytiquement. Ce TP-python propose de découvrir cette méthode numérique pour une équation différentielle d'ordre 1.

Codage

L'ensemble de ce TP numérique se fera à l'aide du Notebook dont le lien est donné ci-dessous.

https://capytale2.ac-paris.fr/web/c/1205-11869258/mcer

I. Méthode d'Euler

L'algorithme d'Euler permet de résoudre de façon approchée une équation différentielle du type

$$\begin{cases} \frac{\mathrm{d}X}{\mathrm{d}t} &= f(t,X) \\ X(t_0) &= X_0 \end{cases} \tag{1}$$

D'après la définition de la dérivée comme limite du taux d'accroissement : $\frac{\mathrm{d}X}{\mathrm{d}t}(t) = \lim_{h\to 0} \frac{X(t+h)-X(t)}{h}$.

Dans le cadre du schéma Euler explicite, au premier ordre, on peut approximer la dérivée à l'instant t par :

$$\frac{\mathrm{d}X}{\mathrm{d}t}(t) \approx \frac{X(t+h)-X(t)}{h}$$

soit

$$X(t+h) \approx X(t) + h \times f(t, X(t))$$

qui correspond au développement limité de X au premier ordre au voisinage de t.

La résolution numérique consiste à déterminer les valeurs X_i de X aux $(n+1)$ instants $(t_i)_{i\in[\![0,n]\!]}$ séparés de la durée h, soit aux instants : $t_i = t_0 + i \times h$, et $t_{i+1} = t_i + h$.

La valeurs approchées X_i de X à l'instant t_i sont définies par la relation de récurrence

$$X_{i+1} \approx X_i + h \times f(t_i, X_i)$$

Ainsi, connaissant la fonction f (forcément puisqu'on connait l'équation différentielle que l'on souhaite résoudre !) et la condition initiale $X(t_0) = X_0$, on peut déterminer $X_1 = X_0 + h \times f(t_0, X_0)$. Puis on peut déterminer X_2 : $X_2 = X_1 + h \times f(t_1, X_1)$, et ainsi de suite.

h est le pas de calcul, qu'il faudra choisir de façon pertinente. Il devra être suffisamment petit pour rendre compte des variations locales de f. En diminuant h on augmente la précision du calcul, mais on augmente aussi le temps de calcul.

I.1. Implémentation

Écrivons une fonction `euler` qui prendra en argument :

- la fonction `f` qui définit l'équation différentielle ;
- les bornes `t0` et `tf` de résolution ;
- le nombre `n` de pas de calcul ;
- la condition initiale `X0`

et renverra la tableau `X` des `n+1` valeurs approchées déterminées.

```python
# On commence par importer les bibliothèques nécessaires :
import numpy as np # pour utiliser les tableaux
import matplotlib.pyplot as plt # pour les représentations graphiques
def Euler(f,X0,t0,tf,n):
    """
    f : fonction définissant l'équation différentielle
    X0 : condition initiale
    t0 : instant initial et tf : instant final
    n : nombre de pas de calcul : h=(tf-t0)/n
    """
    h=                          # (à compléter) pas de calcul
    t=np.linspace(    ,    ,        ) #  (à compléter) liste des temps (n+1 instants de calculs) répartis régulièrement entre t0 et tf
    X=np.zeros(             ) #  (à compléter) tableau qui contient n+1 zéros au départ
    X[0]=       #  (à compléter)  le 1er élément du tableau est la condition initiale
    for i in range(    ,   ): #   (à compléter) il reste n instants à calculer
    # calcul du point suivant (élément i+1) à partir du point précédent (élément i) :
        X[i+1]=                                          #  (à compléter)
    return X #la fonction renvoie le tableau X des valeurs de X successives
```

I.2. Régime transitoire du premier ordre en électricité

On étudie la charge du condensateur à travers une résistance R, par un générateur de fem E constante. La tension aux bornes du condensateur vérifie l'équation différentielle

$$\frac{\mathrm{d}u_C}{\mathrm{d}t} + \frac{u_C}{RC} = \frac{E}{RC}$$

Le condensateur est initialement déchargé, donc $u_C(0) = 0$. On souhaite résoudre cette équation différentielle en utilisant la méthode d'Euler.

1. Récrire l'équation différentielle sous la forme de celle de l'Éq 1 , et identifier la fonction f définissant ici l'équation différentielle.
2. Définir, dans l'éditeur, les grandeurs nécessaires : $E = 5\,\mathrm{V}$, $R = 1000\,\Omega$ et $C = 100\,\mathrm{nF}$.
3. Définir, en python, la fonction `f_charge(t,uc)` qui prend en argument deux flottants (l'instant `t` et la tension `uc`) et renvoie $\frac{\mathrm{d}u_C}{\mathrm{d}t}$ définie par l'équation différentielle (autrement dit la valeur de $f(t, u_c)$ définie précédemment).
4. Écrire l'instruction permettant de récupérer, dans `solution` le tableau des valeurs de u_C par application de la fonction `Euler`. On choisira $t_0 = 0$, $t_f = 10 \times RC$, $n = 10000$.
5. Représenter l'évolution de la tension aux bornes du condensateur.
6. Tracer sur un même graphe, l'évolution de la tension obtenue pour $R = 1 \times 10^2\,\Omega, 1 \times 10^3\,\Omega, 1 \times 10^4\,\Omega$.

```python
t=np.linspace(t0,tf,n+1) # instants
plt.plot(t,solution) # représentation des valeurs de uc en fonction du temps
plt.title('Titre')
plt.xlabel('abscisse (unité)')
plt.ylabel('ordonnée(unités)')
plt.show() # affiche le graphique
```

7. Reprendre toutes les questions en choisissant comme condition initiale $u_C(0) = \frac{E}{2}$.

II. Action d'un filtre

On souhaite étudier l'effet d'un filtre sur un signal $e(t)$ dont on connaît le spectre de Fourier. Ce dernier est constitué uniquement des harmoniques impaires et dont l'amplitude varie en $1/n$ avec n le numéro de l'harmonique. On montre que $e(t)$ peut alors s'écrire sous la forme

$$e(t) = A_0^e + \sum_{n=1}^{N_{max}} A_{2n+1}^e \sin\left(2(2n+1)\pi f \times t\right) \qquad \text{avec} \qquad A_n^e = \frac{4}{n\pi}. \tag{2}$$

8. Montrer que $\langle e(t) \rangle = A_0^e$.
   Le filtre est caractérisé par sa fonction de transfert $H(f, f_c)$ dont on connaît également l'expression :
   $$\underline{H}(f, f_c) = \frac{1}{1 + j\frac{f}{f_c}}.$$
   $\underline{H}$ dépend à la fois de la fréquence f du signal que le filtre reçoit en entrée ainsi que de f_c la fréquence de coupure du filtre.
9. Rappeler l'intérêt de l'étude sinusoïdale et de la décomposition en série de Fourier.

En sortie du filtre, on obtient le signal

$$s(t) = \sum_{n=0}^{N_{max}} A_{2n+1}^s \sin\left(2\pi\left(2n+1\right) f \times t + \phi_{2n+1}\right) \qquad \text{avec} \qquad \begin{cases} A_{2n+1}^s = A_{2n+1}^e \left|H((2n+1)f)\right| \\ \phi_{2n+1} = \arg\left(H((2n+1)f)\right). \end{cases} \tag{3}$$

II.1. Implémentation

On cherche ici à écrire une fonction `Filtrage` qui prend en argument la fréquence `f` du signal sinusoïdal $e(t)$ à filtrer ainsi que la fréquence de coupure `fc` du filtre.
Rigoureusement parlant, il faut sommer les harmoniques jusqu'à l'infini. En pratique, on s'arrêtera à $N_{max} = 5000$.

10. Définir `nmax` dans la fonction `Filtrage`. Définir ensuite un tableau `N` contenant tous les numéros des harmoniques entre 1 et `nmax`. On utilisera à profit la fonction `np.arange`.
11. Définir `Ae` qui contient les amplitudes A_{2n+1}^e du signal d'entrée pour chacune de ses harmoniques.
12. Définir `A0 = 1`.
13. Compléter le script suivant définissant la fonction `H(f,fc)` correspondant à la fonction du transfert du filtre pour une entrée dont la fréquence est `f`.

```
1      j = complex(0,1)  # Définition du nombre complexe j
2      def H(f,fc):
3          """Fonction de transfert du filtre étudié
4          f : fréquence du signal en entrée
5          fc : fréquence de coupure du filtre"""
6          s =  """À compléter"""
7          return s
```

14. Définir la grandeur `gain` contenant le module de la fonction de transfert pour chacune des harmoniques du signal d'entrée. On utilisera la fonction `np.abs`.
15. Définir la grandeur `phase` contenant l'argument de la fonction de transfert pour chacune des harmoniques du signal d'entrée. On utilisera la fonction `np.angle`.
16. Définir l'amplitude `As` et la phase `phis` du signal de sortie contenant chacun l'amplitude et la phase pour chacune des harmoniques du signal d'entrée.
17. Définir la valeur moyenne du signal de sortie `A0s`.
18. On souhaite tracer l'évolution des signaux d'entrée et de sortie pendant deux périodes. Définir en conséquence le temps `t` en complétant `t = np.linspace(... , ... , 500)`.
19. Définir ensuite `e` et `s` correspondants respectivement aux signaux d'entrée $e(t)$ et de sortie $s(t)$. On s'aidera des Éq 2 et 3.
20. Tracer enfin sur un même graphe, l'évolution de $e(t)$ et de $s(t)$ en complétant les instructions suivantes :

```
1      fig,ax = plt.subplots(1,2)
2
3      ax[0].plot(...,..., label='entree')
```

```
    ax[0].plot(...,..., label='sortie')
    ax[0].set_xlabel('abscisse (unite)')
    ax[0].set_ylabel('ordonnee (unite)')
    ax[0].legend()

    # Tracé des spectres NE PAS MODIFIER
    ax[1].loglog(N*f,Ae, label='Spectre de e(t)')
    ax[1].loglog(N*f,gain, label='Gain du filtre')
    ax[1].loglog(N*f,As, label='Spectre de s(t)')
    ax[1].set_xlabel('frequence (Hz)')
    ax[1].set_ylabel('amplitude')
    ax[1].legend()
    ax[1].grid(which='both')

    fig.show()
```

II.2. Analyse du signal filtré $s(t)$

21. Exécuter la fonction `Filtrage(1e3,1e4)` où $f = 1\,\text{kHz}$ et $f_c = 10\,\text{kHz}$. À quoi correspond le signal d'entrée $e(t)$? Et le signal de sortie ?
22. Exécuter la fonction en faisant varier la fréquence du signal d'entrée $f = 10\,\text{kHz}, 100\,\text{kHz}, 1\,\text{MHz}$. Expliquer l'allure du signal de sortie $s(t)$.
23. Interpréter les différents spectres obtenus.
24. En déduire le type de filtre étudié.

II.3. Pour aller plus loin...

25. Écrire une nouvelle fonction `FiltragePH` analogue à la fonction `Filtrage` mais dont la fonction de transfert du filtre s'écrit à présent
$$\underline{H}(f) = \frac{j\frac{f}{f_c}}{1+\frac{f}{f_c}}.$$
26. Déterminer de quel type de filtre s'agit-il.