TP n° 22 : Méthode d'Euler

L'objectif de ce TP est de programmer la méthode d'Euler ainsi que différents algorithmes de résolution d'équations différentielles de la forme

$$\begin{cases} y(t_0) = y_0 \\ y' = F(t,y) \end{cases}$$

et de comparer leur efficacité.

1 Méthode d'Euler

1. Au brouillon, et si possible sans regarder le cours, se remémorer l'idée qui se cache derrière la méthode d'Euler et retrouver le schéma numérique d'intégration. Rappelons que l'on va chercher à approcher la fonction y en certains points t_k par une valeur y_k. Il s'agit donc d'exprimer y_{k+1} en fonction de F, y_k, t_k et du pas $h = t_{k+1} - t_k$.
2. Implémenter une fonction `euler(F, y0, a, b, h)` résolvant numériquement par la méthode d'Euler explicite le problème de Cauchy

$$\begin{cases} y(a) = y_0 \\ y' = F(t,y) \end{cases}$$

sur $[a,b]$. Cette fonction devra renvoyer un couple formé des listes (ou des tableaux NUMPY) du temps discrétisé (t_k) avec un pas h et des valeurs (y_k) approchées de y correspondantes.
3. Vérifier pour l'équation différentielle

$$\begin{cases} y(0) = 0 \\ y' = y \end{cases} \tag{1}$$

sur $[0,1]$ avec $h \in \{0.1, 0.01, 0.001\}$. Tracer sur un même graphe la solution exacte et la solution approchée obtenue.

4. Vérifier pour l'équation différentielle

$$\begin{cases} y(0) = 0 \\ y' + y = t \end{cases} \tag{2}$$

sur $[0,1]$ avec $h \in \{0.1, 0.01, 0.001\}$. Tracer sur un même graphe la solution exacte — que vous savez calculer — et la solution approchée obtenue.

5. Essayer pour l'équation différentielle

$$\begin{cases} y(0) = 1 \\ y' = \sin(t)\sin(y) \end{cases} \tag{3}$$

sur $[0,42]$ avec $h \in \{0.1, 0.01, 0.001\}$. Tracer la solution approchée obtenue.

2 Utilisation d'une bibliothèque

La bibliothèque `scipy`, que vous devez avoir maintenant réussi à installer, fournit une fonction `odeint` permettant le calcul approché d'une solution d'équation différentielle. Elle se trouve dans le sous-module `integrate`. Pour l'utiliser, on peut par exemple faire :

PYTHON
```
from scipy.integrate import odeint
```

Allez parcourir rapidement la documentation pour comprendre son fonctionnement :

- `http://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html`
- ou bien en console `help(odeint)`

⚠ Pour ne pas faire comme tout le monde, `odeint` prend en argument une fonction $F(y,t)$ et non $F(t,y)$, à moins de passer en argument supplémentaire `tfirst=True`.

6. Superposer sur les graphes obtenus précédemment pour les équations (1), (2) et (3) la solution approchée calculée par `odeint`.

3 Autres schémas d'intégration

Pour résoudre l'équation $y' = F(t,y)$ par la méthode d'Euler, on a procédé exactement de la même manière que lors du cours sur l'intégration numérique. En intégrant l'équation $y'(t) = F(t,y(t))$ entre t_k et t_{k+1}, on se retrouve avec :

$$y_{k+1} = y_k + \int_{t_k}^{t_{k+1}} F(t,y(t))dt$$

On peut alors utiliser les différentes formules de quadrature que nous avons rencontrées dans le chapitre sur l'intégration numérique.

3.1 Méthode d'Euler explicite

7. Que se passe-t-il si on utilise la méthode des rectangles à gauche pour approcher l'intégrale ?

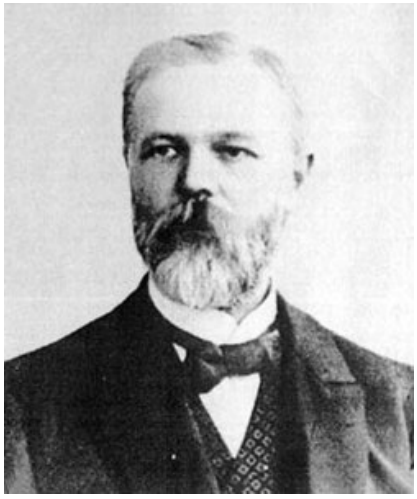
3.2 Méthode d'Euler implicite

8. Quelle équation liant F, y_k, y_{k+1}, t_{k+1} et h obtient-on en utilisant la méthode des rectangles à droite. Peut-on en déduire simplement y_{k+1} en général ?

L'équation obtenue dans la question précédente peut se mettre sous la forme $f(y_{k+1}) = 0$. Si on ne sait pas résoudre cette équation en général, on peut utiliser une des méthodes de résolution approchée du cours sur la résolution numérique d'équation sur les réels. Cette méthode s'appelle la *méthode d'Euler implicite*. On peut s'en souvenir en remarquant que y_{k+1} est implicite et non explicite ici.

9. Implémenter une fonction `euler_implicite(F, y0, a, b, h, epsilon)` qui implémente la méthode d'Euler implicite. Pour résoudre numériquement l'équation $f(y_{k+1}) = 0$, on utilisera la méthode de NEWTON, que l'on retrouvera au brouillon avant de la réimplémenter sans tricher, avec un critère d'arrêt $|x_{i+1} - x_i| \leq \epsilon$, où `epsilon` est passé en argument.
10. Superposer également sur les graphes obtenus précédemment pour les équations (1), (2) et (3) la solution approchée calculée par cette méthode.

3.3 Méthode de HEUN



11. Appliquer la méthode des trapèzes pour approcher l'intégrale, puis utiliser le schéma d'Euler explicite pour éliminer le y_{k+1} qui se trouve comme argument de la fonction F. Montrer que le schéma numérique obtenu est :

$$y_{k+1} = y_k + h\frac{F(t_k, y_k) + F\left(t_{k+1}, y_k + hF(t_k, y_k)\right)}{2}$$

Cette méthode s'appelle la *méthode de Heun*.

12. Implémenter une fonction `heun(F, y_0, a, b, h)` mettant en œuvre cette méthode.
13. À nouveau, superposer les résultats obtenus par cette méthode sur les graphes pour les trois exemples d'équation différentielle vus précédemment.

4 Tracé de l'erreur

On souhaite tracer sur un même graphe bilogarithmique (`plt.loglog()`) l'erreur globale $\max_k |y(t_k) - y_k|$ correspondant aux méthodes étudiées pour l'équation différentielle (1), en fonction du pas h pour $h \in \{10^{-1}, 10^{-2}, \dots, 10^{-6}\}$, ainsi que pour la fonction `odeint`.

14. Comprendre et compléter la fonction suivante.

PYTHON

```
def erreur():
    a, b, y0 = 0, 1, 0
    F = lambda t, y: ...
    exacte = lambda t: ... # solution exacte
    H = 10. ** (-np.arange(..., ...))
    methodes = [(euler,"Euler explicite"),
                (euler_implicite, "Euler implicite",
                (heun,"Heun"),
                (odeint,"odeint")]
    # Tracé des erreurs
    for schema, nom in methodes:
        err = [] # liste d'erreurs pour chaque val. de h
        for h in H:
            if nom == "odeint":
                t = np.arange(... , ... , ...)
                approche = odeint(lambda y, t: ...,
                                  ... ,...)[:, 0]
            else:
                t, approche = ...
            err.append(...)
        plt.loglog(... ,... , label=...)

    # Témoins
    for i in (1,2):
        plt.loglog(H, H**i, ':', label="$h^{}$".format(i))

    plt.legend(loc='best')
    plt.savefig('erreur')
    plt.show()
```