Résolution approchée d'une équation différentielle ordinaire par la méthode d'EULER

TD 10

Compétences

- La méthode d'EULER
- Influence du pas de discrétisation et du nombre d'itérations sur la qualité des résultats et sur le temps de calcul
- Comparer les résultats obtenus avec les fonctions de résolution fournies par une bibliothèque numérique

Pré-conditions

Dans ce TD, on s'intéresse aux équations différentielles d'ordre 1 dont la fonction inconnue est soit à valeurs réelles, soit à valeurs vectorielles donnée par la forme canonique suivante :

$$\begin{cases} X'(t) = f(X(t), t) \\ X(t_0) = X_0 \end{cases} \tag{1}$$

où X est une application de I = [t0 ; t1], un segment de $\mathbb{R}$, dans $\mathbb{R}^n$ et f est une application de $\mathbb{R}^n \times I$ dans $\mathbb{R}^n$, on impose également une condition initiale $X(t_0)$.

Dans la plupart des cas, on ne sait pas résoudre explicitement ce type d'équation d'où la nécessité de mettre au point des méthodes numériques de résolution approchée.

Un exemple simple de résolution approchée en physique nucléaire

Dans cette première partie on cherche à confronter une méthode de résolution numérique des équations différentielles à valeurs réelles du type :

$$\begin{cases} x'(t) = f(x(t), t) \\ x(0) = x_0 \end{cases}$$

à une équation différentielle linéaire (scalaire) du premier ordre à coefficients constants dont on connait la forme de la solution. La décroissance radioactive conduit à ce type d'équation différentielle :

$$\begin{cases} N'(t) = -\lambda N(t) \\ N(0) = N_0 \end{cases}$$

dans ce cas la fonction $f(N(t), t) = -\lambda N(t)$ est à valeurs réelles et linéaire en N(t), avec λ une constante dépendant du noyau atomique.

À partir de la solution théorique

Pour ce genre d'équation on connait la solution théorique : $N(t) = N_0 e^{-\lambda t}$

1. L'iode 131 est utilisé en médecine, par exemple pour l'examen par scintigraphie des glandes surrénales, tracer avec *matplotlib* la courbe d'évolution de la quantité d'iode restant dans le corps en fonctions du temps pour les conditions initiales suivantes : $\lambda(^{131}\text{I}) = 8.55 \times 10^{-2}\text{jour}^{-1}$ et $N(0) = 4,6 \times 10^{21}$ noyaux

```python
def discretisation(a, b, N):
    '''
    Renvoie la liste des valeurs permettant  de
    dûcouper un intervalle en N parties ûgales
    ainsi que l'intervalle entre deux ûchantillons
    c'est û dire la pas h
    '''
    t = np.linspace(a, b , N+1)
    h = t[1] - t[0]
    return t, h
```

Cette méthode permet d'obtenir une liste de points de la variable temps : t, et la valeur du pas : h
On obtient N+1 valeurs pour N intervalles

2. À partir de quand l'iode 131 a-t-il complètement disparu dans le corps du patient ?

Résolution numérique approchée : la méthode d'EULER

Nous allons maintenant essayer d'approcher au plus juste la courbe théorique à partir d'une résolution numérique avec la méthode d'EULER. C'est sans doute la méthode la plus simple pour résoudre numériquement une équation différentielle, toutefois elle est limité par sa précision d'ordre 1.

Comment fait-on ? L'idée est d'exprimée la dérivée par une différence finie :

$$x'(t) \approx \frac{x(t+h) - x(t)}{h} \quad \text{pour } h \text{ petit}$$

c'est à dire

$$x(t+h) \approx x(t) + hx'(t) \approx x(t) + hf(x(t), t)$$

Pour la décroissance radioactive en partant du point (t_0, N_0), on obtient :

$$\begin{cases} t_1 = t_0 + h \\ N(t_1) = N_0 + hf(N_0, t_0) = N_0 - h \times \lambda \times N(t_0) \end{cases}$$

On construit ainsi la suite de points suivante :

$$\begin{cases} t_{k+1} = t_k + h \\ N(t_{k+1}) = N(t_k) + hf(N(t_k), t_k) \end{cases}$$

1. À partir de la fonction prototype suivante `def euler(f, a, b, y0, h):` écrire une implémentation
   - f : $f(x(t), t)$
   - a : la valeur de t_0 (instant initial)
   - b : la valeur de t_n
   - y0 : condition initiale à t_0
   - h : le pas
2. Tester votre implémentation avec les pas suivant h = [10.0, 6.25, 5.0, 2.5, 1.25, 1.0, 0.5], que peut-on remarquer ?
3. Qu'en est-il si on choisit un pas $h = 10^{-6}$?
4. Considérons le problème de CAUCHY dans le cas particulier où $f(t, y(t)) = -\beta y(t)$ avec β un nombre réel positif donné. Vous avez vu en cours de mathématiques que sa solution est $y(t) = y_0 e^{-\beta t}$. Puisque β est positif, ce problème est numériquement bien posé : $y(t)$ décroît exponentiellement lorsque t tend vers l'infini. Pour discrétiser la demi-droite $t \leq 0$, nous choisissons un nombre réel $h > 0$ et nous posons $t_n = nh$ avec n = 0, 1, 2, . . .
   Dans ce cas le schéma d'EULER devient
   $$u_{n+1} = (1 - \beta h)u_n$$
   et par suite
   $$u_{n+1} = (1 - \beta h)^n u_0$$
   Bien que la solution $y(t)$ tend vers zéro lorsque t tend vers l'infini, nous voyons que si $u_0 = 0$ et $1 - \beta h < -1$ alors u_n tend vers l'infini en alternant de signe lorsque t tend vers l'infini. Nous dirons dans ce cas que le schéma d'EULER est instable.
   Calculer dans le cas de l'iode la valeur de h_{max} à partir de laquelle u_n tend vers l'infini. À partir de votre code commenter les résultats numériques obtenus autour de la valeur h_{max}
   Cette condition est appelée condition de stabilité : elle limite le pas h d'avance en t lorsqu'on utilise le schéma d'EULER progressif. Il existe un schéma d'EULER appelé rétrograde toujours stable, sans limitation sur h.

Circuit RC

Considérons l'équation différentielle linéaire du premier ordre à coefficients constants avec second membre suivante :

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

Cette équation différentielle très simple peut correspondre aux modèles physiques suivants

- charge d'un circuit RC
- montée en température d'un système thermique
- évolution de la vitesse d'un solide avec frottements

Dans le cas d'un circuit RC on a $u = E$ et $\tau = RC$, la solution de cette équation différentielle est donnée par :

$$u_c(t) = u(1 - e^{-\frac{t}{\tau}})$$

1. Écrire l'equation différentielle d'un circuit RC sous la forme canonique de l'équation (1)
2. Identifier $f(X(t), t)$
3. Montrer que
   $$\begin{cases} t_{k+1} = t_k + h \\ u_c(t_{k+1}) = u_c(t_k) + h \times \dfrac{E - u_c(t_k)}{RC} \end{cases}$$
4. Pour $u = 2\ V$ et $\tau = 10\ s$ à partir de quelle valeur de h la solution numérique devient-elle oscillante ? instable ?

Un module pour résoudre les équations différentielles : scipy.integrate.odeint

Dans le domaine de l'analyse numérique Python propose de nombreux modules. Les fonctions de ces modules sont en général bien mieux optimisées que les fonctions *maisons*. Pour les équations différentielles Python possède le solver générique assez sophistiqué : `odeint` contenu dans le module `scipy.integrate`. À importer en début de programme `from scipy.integrate import odeint` Il résout, en autre, avec une syntaxe relativement simple les équations différentielles du type $y' = f(y(t), t)$. Pour une documentation complète, comme à l'habitude je conseille la page officielle : http://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html

Utilisation de scipy.integrate.odeint

Pour utiliser odeint, il faut commencer par implémenter la fonction dérivée f(x(t),t) que l'on tire de la forme canonique (1). Cette fonction est ensuite passée en paramètre à `odeint` avec la condition initiale $x(t_0)$ et t une `list` ou un `array` pour chaque point du temps.

```
from scipy.integrate import odeint
def f(y, t):
    ''' fonction scalaire de l'ûquation
        diffûrentielle'''
    return -Lambda * y

x_ode = odeint(f, No, t)
plt.plot(t, x_ode, '--x',label='odeint')
plt.legend(loc='upper right')
```

1. Implémenter cette solution à la suite de votre travail
2. Commenter le résultat, vous ajouterez une légende à chaque courbe tracée.