TP 1

Résolution numérique d'équations différentielles ordinaires - 1ère partie : la méthode d'Euler et rk4

1.1 La méthode d'Euler

Une équation différentielle du premier ordre satisfaite par une fonction $y(t)$ est de la forme,

$$\frac{dy(t)}{dt} = f\left(y(t),\, t\right) \tag{1.1}$$

où f est une fonction connue qui dépend du problème considéré. La méthode la plus simple pour résoudre numériquement une telle équation est la méthode d'Euler, figure 1.1 . La fonction y est calculée aux points $t_k = k\Delta t$ avec k entier, séparés par un petit intervalle Δt. La valeur de y au point $t_{k+1} = t_k + \Delta t$ est obtenue à partir de la valeur au point t_k par la formule,

$$y(t_{k+1}) = y(t_k) + f\left(y(t_k),\, t_k\right) \times \Delta t \tag{1.2}$$

A partir de la donnée d'une condition initiale $y(t_0)$, on peut ainsi calculer de proche en proche y pour des valeurs successives de t. Les questions 1-3 proposent des applications de cette méthodes à des cas très simples.

1. L'évolution dans le temps de la densité $x(t)$ d'un élément radioactif X subissant une réaction de désintégration $X \to Y$ obéit à l'équation différentielle

$$\frac{dx}{dt} = -kx \tag{1.3}$$

où k est le taux de désintégration. Écrire un petit programme qui calcule par la méthode d'Euler et enregistre x pour des valeurs de t entre 0 et $Tmax$. On prendra $k = 1$ et vous fixerez $Tmax$, Δt et $x(0)$. Tracer $x(t)$ et comparer avec la solution analytique de l'équation 1.3.

2. Supposons maintenant que l'élément Y, dont la densité est notée $y(t)$, se désintègre en un élément Z avec le tau k_2. On a alors,

$$\begin{aligned}\frac{dx}{dt} &= -kx \\ \frac{dy}{dt} &= kx - k_2 y\end{aligned} \tag{1.4}$$

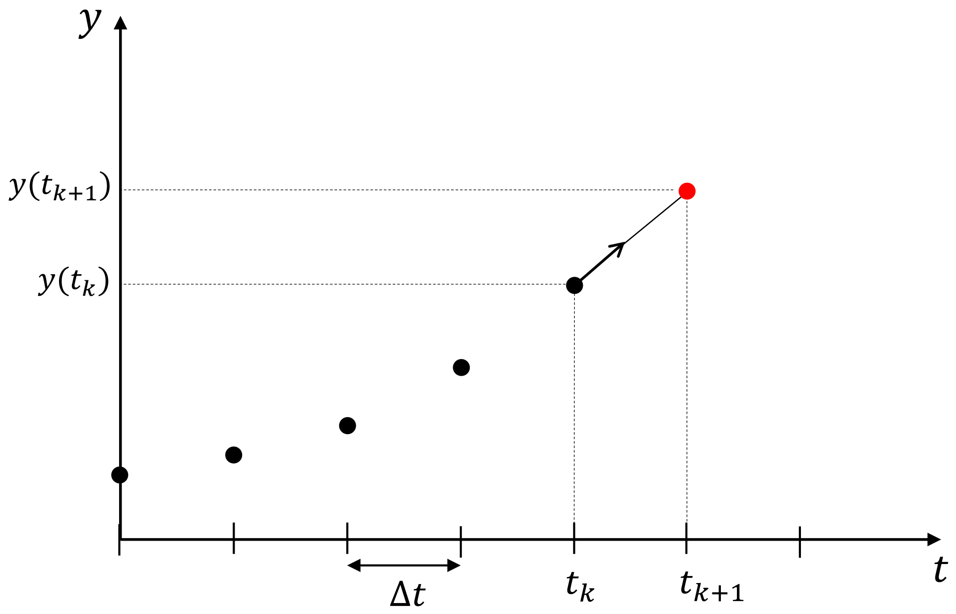

FIGURE 1.1 – Schéma méthode d'Euler.

Écrire un programme analogue au précédent qui calcule et enregistre $x(t)$ et $y(t)$ de $t = 0$ à $Tmax$. On prendra $k_2 = 0.1k$, $x(0) = 1$ et $y(0) = 0$. Tracer $x(t)$ et $y(t)$.

3. L'évolution d'un oscillateur harmonique est décrite par une équation du second ordre

$$\frac{d^2x}{dt^2} = -\omega_0^2 x \tag{1.5}$$

Une équation différentielle du second ordre peut être écrite sous forme de deux équations du premier ordre. Écrire ces deux équations et écrire un programme calculant $x(t)$. Vous fixerez les paramètres et les conditions initiales. Essayer plusieurs pas de temps Δt, qu'est-ce que vous observez sur un grand nombre de périodes ?

De manière générale, une équation différentielle d'ordre supérieur à un peut être transformée en un système d'équations du premier ordre.

Le but des quatre questions suivantes est de construire une fonction `euler` qui implémente l'algorithme d'Euler sur un pas et qui puisse être utilisée sans modification dans tout programme nécessitant de résoudre un système d'équations différentielles. Pour un ensemble de n équations différentielles du premier ordre satisfaites par n fonction $y_i(t)$ avec $i = 0, 1, .., n-1$,

$$\frac{dy_i(t)}{dt} = f_i\Big(\{y_j(t)\}, t\Big), \tag{1.6}$$

l'algorithme d'Euler pour le calcul des y_i au point $t + \Delta t$ à partir des valeurs en t peut être découpé en deux étapes :

1. le calcul de la dérivée de chaque fonction y_i au point t, donnée par $y_i'(t) = f_i\Big(\{y_j(t)\}, t\Big)$.
2. le calcul des y_i au point $t + dt$ avec la formule $y_i(t + \Delta t) = y_i(t) + y_i'(t) \times \Delta t$.

Les valeurs des fonctions y_i au point t sont stockées dans un tableau y[n]. L'étape 1 qui dépend de l'équation considérée (de la forme des f_i), est réalisée par une fonction,

```
deriv(int n, double t, double y[], double dy[])
```

qui à partir des $y_i(t)$ calcule les $y_i'(t)$ et les stocke dans le tableau dy[]. Notons que `deriv` prend aussi comme argument n et t qui peuvent être nécessaire dans certain cas pour calculer les dérivées.

4. Écrire la fonction `deriv` dans le cas de l'équation 1.4. (les paramètres physiques de l'équation différentielle sont déclarés en variables globales).

5. Écrire la fonction

```
euler(int n, double t, double y[], double dt)
```

qui reçoit en entrée les $y_i(t)$ dans le tableau y[], ainsi que les valeurs de n , t et Δt, calcule les $y_i(t + \Delta t)$ et les stocke dans le tableau y[] à la place des $y_i(t)$. Il faudra dans cette fonction déclarer un tableau dy[] et appeler la fonction `deriv` pour remplir ce tableau avec les $y_i'(t)$. Inclure les fonctions `deriv` et `euler` dans un programme complet permettant de traiter la question 2.

6. Créer une fonction `deriv3` permettant de résoudre l'équation 5 et modifier `euler` pour traiter la question 3.

7. Pour que la fonction `euler` puisse être utilisée sans aucune modification en toute circonstance, il faudrait pouvoir lui spécifier lorsqu'on l'appelle quelle fonction choisir pour calculer les y_i' (par exemple `deriv` ou `deriv3` ou autre). Autrement dit, il faut pouvoir passer en argument la fonction à utiliser pour calculer les y_i' . La fonction `euler` doit alors être de la forme,

```
euler(int n, double t, double y[], double dt, void f(int, double , double[], double[])
```

Écrire la fonction euler. Modifier la fonction `main` en conséquence et tester ce programme sur la question 2 et la question 3.

8. Écrire un programme utilisant la fonction euler pour calculer numériquement la trajectoire dans le plan $x - y$ d'une particule de masse m et charge q dans des champs électrique et magnétique uniformes, $\vec{E} = E\vec{u_x}$ et $\vec{B} = B\vec{u}_z$. On rappelle que l'équation du mouvement est

$$m\frac{d\vec{v}}{dt} = q\left(\vec{E} + \vec{v} \times \vec{B}\right) \tag{1.7}$$

Les unités sont choisies de telle sorte que $q/m = E = B = 1$. Quel est le nombre n d'équations différentielles ? Écrire ces équations et la fonction `deriv` correspondante.

1.2 Méthode de Runge-Kutta

Pour $y(t + \Delta t)$ à partir de $y(t)$, les méthodes de types Runge-Kutta estime la valeur de la dérivée y' au milieu de l'intervalle entre les points t et $t + \Delta t$. La version la plus simple pour résoudre l'équation 1.1 procède ainsi (voir figure 1.2) :

$$\begin{aligned} d_1 &= f(y(t), t) \\ y_p &= y(t) + d_1 \times \Delta t/2 \\ d_2 &= f(y_p, t + \Delta t/2) \\ y(t + \Delta t) &= y + d_2 \times \Delta t \end{aligned} \tag{1.8}$$

Cette méthode s'appelle Runge-Kutta d'ordre 2 car on peut montrer que les erreurs sont d'ordre $(\Delta t)^3$.

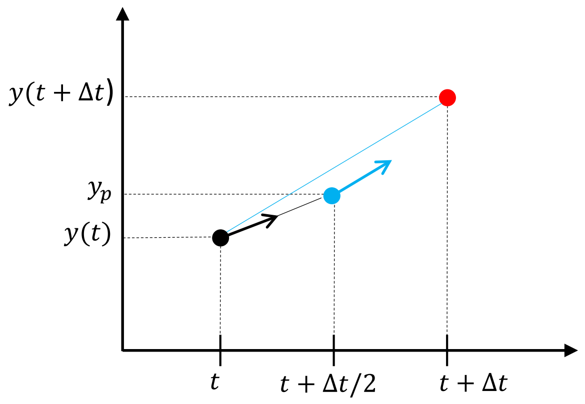

FIGURE 1.2 – Schéma méthode RK2

La méthode la plus couramment utilisée est Runge-Kutta d'ordre 4 (les erreurs sont d'ordres $(\Delta t)^5$) qui utilise 4 évaluations de la dérivée. La fonction est donnée dans le fichier rk4.cpp (voir sur moodle) et vous pouvez l'utiliser directement en la copiant dans votre programme.

9. Résoudre l'équation $\frac{d^2x}{dt^2} = -\omega_0^2 x$ en utilisant la fonction `euler` avec $\Delta t = 0.01s$. Tracer $x(t) - x_{th}(t)$ où $x(t)$ est la solution numérique et $x_{th}(t)$ est la solution analytique exacte. Que constatez-vous ? Diminuer le Δt. Le problème est-il résolu ?

10. Refaire la question précédente en utilisant à présent la fonction `rk4`. Que constatez-vous ?

1.3 *Pour aller plus loin :* implémenter rk2()

11. Ecrivez une fonction rk2() qui implémente l'algorithme RK2 décrit plus haut.

12. Comparer cette méthode avec la méthode d'Euler et de rk4 sur l'oscillateur harmonique.