Intégration d'équations différentielles ordinaires : Application à un problème de balistique en 2 dimensions

On se propose d'intégrer numériquement l'équation du mouvement d'un projectile lancé dans un plan à une vitesse $\mathbf{v}_0$ en utilisant différents algorithmes de résolution numérique des équations différentielles ordinaires.

1 Introduction : Intégration par différences finies

Considérons l'équation différentielle d'ordre 1 ci-dessous, où f est une fonction ayant les "bonnes" propriétés mathématiques requises à l'application des schémas numériques présentés :

$$\dot{x} = f(x,t) \tag{1}$$

Elle peut être intégrée numériquement de la façon suivante : le temps t est discrétisé par $t_i = i \times dt$ où i est un entier et Δt, le pas de temps élémentaire. Connaissant la position du projectile à l'instant t_i, on déduit la position à l'instant t_{i+1} en utilisant simplement la formule de Taylor. Ce qui donne :

$$\dot{x} = f(x,t) \longrightarrow \frac{x_{i+1} - x_i}{\Delta t} \simeq f(x_i, t_i) \tag{2}$$

soit,

$$x_{i+1} \simeq x_i + f(x_i, t_i)\Delta t \tag{3}$$

Noter que pour la résolution numérique, il faut connaître la position du projectile à un instant t_0 donné et qu'il faut appliquer itérativement la formule approchée qu'est l'équation (3) pour obtenir la solution en fonction du temps.

2 Équation du mouvement

On considère un projectile de masse M lancé à partir de l'origine ($x = 0$ et $y = 0$) à la vitesse initiale $\mathbf{v}_0$ faisant un angle θ avec l'horizontale. Le projectile est soumis à deux forces : son poids $\mathbf{P} = M\mathbf{g}$ et une force de frottement visqueux $\mathbf{f} = -\eta\mathbf{v}$. L'équation du mouvement s'écrit :

$$M\frac{d^2\mathbf{r}}{dt^2} = \mathbf{F} \qquad \text{soit :} \qquad M\frac{d^2\mathbf{r}}{dt^2} = M\mathbf{g} - \eta\mathbf{v}$$

En projetant sur chacun des axes et en notant les dérivations par rapport au temps avec des points, on a :

$$\left\{ \begin{array}{l} F_x = -\eta\,\dot{x} \\ F_y = -Mg - \eta\,\dot{y} \end{array} \right. \quad \text{soit pour l'équation du mouvement :} \quad \left\{ \begin{array}{l} M\,\ddot{x} = -\eta\,\dot{x} \\ M\,\ddot{y} = -Mg - \eta\,\dot{y}. \end{array} \right. \tag{4}$$

On obtient donc deux équations différentielles couplées que l'on va résoudre en se donnant des conditions initiales à $t = 0$.

Pour intégrer numériquement le système (4) on transforme celui-ci en un système de quatre équations différentielles du premier ordre en introduisant les fonctions $p = \dot{x}$ et $q = \dot{y}$:

$$\left\{\begin{array}{lcl} \dot{p} &=& f_1(p,q) \\ \dot{x} &=& p \\ \dot{q} &=& f_2(p,q) \\ \dot{y} &=& q \end{array}\right. \quad \text{avec} \quad \left\{\begin{array}{lcl} f_1(p,q) &=& -\dfrac{\eta}{M}\, p, \\ f_2(p,q) &=& -\dfrac{\eta}{M}\, q \;-\; g. \end{array}\right. \tag{5}$$

Le point d'impact du projectile avec le sol correspond au point $(x = x_{max},\, y = 0)$ sur la figure 1.

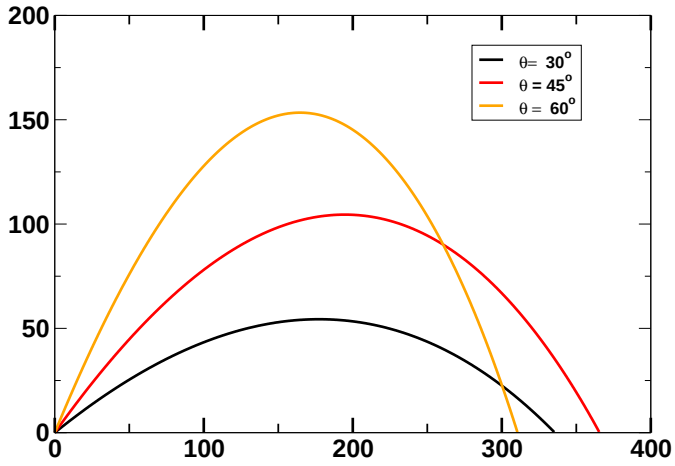

FIGURE 1 – Trajectoires d'un objet de 1 kg lancé à la vitesse initiale ($v_0 = 70$ m/s) avec une force de frottement visqueux $-\lambda v\mathbf{v}$ et non en $-\eta\mathbf{v}$ ($\lambda = 0,001$ kg/m) pour trois angles de tir θ différents.

On définira l'angle de tir optimal θ_{opt} comme l'angle correspondant à la valeur maximale de la courbe x_{max}=$h(\theta)$.

3 Intégration à l'aide de la méthode d'Euler

On transforme le système (4) de deux équations différentielles de second ordre en un système de quatre équations différentielles chacune de premier ordre en appliquant le même principe que dans la section 2, on obtient :

$$\left\{\begin{array}{lcll} p_{i+1} &=& p_i &+\; f_1(p_i,q_i)\,dt \\ x_{i+1} &=& x_i &+\; p_i\,dt \\ q_{i+1} &=& q_i &+\; f_2(p_i,q_i)\,dt \\ y_{i+1} &=& y_i &+\; q_i\,dt \end{array}\right. \tag{6}$$

Connaissant la position et la vitesse du projectile à l'instant t_0 et en résolvant de manière itérative le système d'équations (6), on déduit pas à pas la vitesse (x, y) et la position du projectile $(p, q) = (\dot{x}, \dot{y})$ à tout instant. On notera que la détermination de la position et de la vitesse à l'instant t_{i+1} dépend uniquement des valeurs de ces quantités à l'instant t_i.

4 Intégration à l'aide de la méthode d'Euler–Cromer

C'est une variante de la méthode d'Euler. Le système s'écrit sous la forme :

$$\left\{\begin{array}{lcll} p_{i+1} & = & p_i & + & f_1(p_i, q_i)\, dt \\ x_{i+1} & = & x_i & + & p_{i+1}\, dt \\ q_{i+1} & = & q_i & + & f_2(p_i, q_i)\, dt \\ y_{i+1} & = & y_i & + & q_{i+1}\, dt \end{array}\right. \tag{7}$$

Il diffère de (6) par l'utilisation des valeurs de la vitesse à l'instant t_{i+1} pour calculer la position au même instant, ce qui rend l'intégration plus stable.

5 Intégration à l'aide de la méthode de Runge-Kutta d'ordre 2 ou méthode du point milieu

Les méthodes de Runge-Kutta concernent les équations différentielles de la forme :

$$\frac{dy}{dt} = F(t, y)$$

où y est éventuellement un vecteur colonne $[y^1, y^2, \ldots, y^n]^T$ de dimension n, F est une fonction suffisamment régulière de t et de y. En combinant les développements de Taylor de F en des points bien choisis, on peut obtenir des relations d'extrapolation plus précises que les formules précédentes (6) et (7).

On peut réécrire la méthode d'Euler sous une forme compacte, en notant comme précédemment le pas de temps par dt :

$$\begin{array}{lcl} k_1 & = & F(t_i, y_i)dt \\ y_{i+1} & = & y_i + k_1 + \Theta(dt^2) \end{array} \tag{8}$$

Pour améliorer cette estimation et obtenir une erreur en $\Theta(dt^3)$ et non en $\Theta(dt^2)$, on utilise un point intermédiaire en $t_i + \frac{1}{2}dt$ (noté $t_{i+1/2}$) et en $y_i + (1/2)k_1$ (noté $y_{i+1/2}$). On calcule d'abord une première évaluation du terme F que l'on utilise ensuite afin d'obtenir une meilleure estimation de F au pas de temps suivant. Ce qui donne :

$$\begin{array}{lcl} k_1 & = & F(t_i, y_i)dt \\ k_2 & = & F(t_i + \frac{1}{2}dt, y_i + (1/2)k_1)dt = F(t_{i+1/2}, y_{i+1/2})dt \\ y_{i+1} & = & y_i + k_2 + \Theta(dt^3) \end{array} \tag{9}$$

Pour le système d'équations différentielles (4), on a la correspondance :

$$\left\{\begin{array}{lcl} y^1 & = & p \\ y^2 & = & x \\ y^3 & = & q \\ y^4 & = & y \end{array}\right. \tag{10}$$

On peut donc réécrire explicitement le système (9) pour notre exemple en fonction de p, x, q, et y.

1. On calcule d'abord les termes :

$$k_1^p = f_1(p_i, q_i)dt \qquad \text{et} \qquad k_1^q = f_2(p_i, q_i)dt \tag{11}$$

qui correspondent aux accroissements des composantes de la vitesse à l'instant t_i. On fait de même pour les accroissements liés à la position k_1^x et k_1^y (mais c'est plus simple car les fonctions f_i sont alors égales à l'identité (voir formules (5) et (9)).

$$k_1^x = p_i dt \qquad \text{et} \qquad k_1^y = q_i dt \tag{12}$$

2. On calcule ensuite les accroissements précédents à l'instant $(t_{i+1/2})$:

$$k_2^p = f_1(p_i + \frac{1}{2}k_1^p, q_i + \frac{1}{2}k_1^q)dt \qquad \text{et} \qquad k_2^q = f_2(p_i + \frac{1}{2}k_1^p, q_i + \frac{1}{2}k_1^q)dt \tag{13}$$

On fait de même pour les accroissements liés à la position k_2^x et k_2^y (mais de nouveau c'est plus simple pour les mêmes raisons que précédemment (voir formules (5) et (9)).

$$k_2^x = (p_i + \frac{1}{2}k_1^p)dt \qquad \text{et} \qquad k_2^y = (q_i + \frac{1}{2}k_1^q)dt \tag{14}$$

3. Puis on déduit les composantes de la vitesse à l'instant t_{i+1} (cf. dernière ligne des formules 9) par

$$p_{i+1} = p_i + k_2^p \qquad \text{et} \qquad q_{i+1} = q_i + k_2^q \tag{15}$$

De même pour la position à l'instant t_{i+1} :

$$x_{i+1} = x_i + k_2^x \qquad \text{et} \qquad y_{i+1} = y_i + k_2^y \tag{16}$$

4. Finalement la méthode Runge–Kutta d'ordre 2 donne les équations suivantes pour évaluer les quatre composantes à l'instant t_{i+1} :

$$\left\{\begin{array}{lcl} k_1^p &=& f_1(p_i, q_i)dt \\ k_1^x &=& p_i dt \\ k_1^q &=& f_2(p_i, q_i)dt \\ k_1^y &=& q_i dt \\ \\ k_2^p &=& f_1(p_i + (1/2)k_1^p, q_i + (1/2)k_1^q)dt \\ k_2^x &=& p_i dt + (1/2)k_1^p dt \\ k_2^q &=& f_2(p_i + (1/2)k_1^p, q_i + (1/2)k_1^q)dt \\ k_2^y &=& q_i dt + (1/2)k_1^q dt \\ \\ p_{i+1} &=& p_i + k_2^p \\ x_{i+1} &=& x_i + k_2^x \\ q_{i+1} &=& q_i + k_2^q \\ y_{i+1} &=& y_i + k_2^y \end{array}\right. \tag{17}$$

6 Programmation

On veut écrire un programme en `python` qui permette de déterminer la trajectoire du projectile et de calculer la distance atteinte en fonction de l'angle de tir connaissant les conditions initiales définies au début du programme.

6.1 Consignes générales pour les différentes questions

- On chargera le module `numpy` que l'on utilisera pour gérer les tableaux.
- La boucle de récurrence sur le temps s'effectue dans le corps du programme (et non dans une fonction *ad hoc*). Cette boucle sera répétée **tant que** $y \geq 0$.
- On utilisera des tableaux pour stocker les valeurs des variables d'intérêt pour chaque t_i.
- Pour chaque méthode (voir les sections 3 à 5), les formules de récurrence sont mises sous forme de fonctions séparées comme indiqué ci-après.
- La position (x, y) et la vitesse $(\dot{x}, \dot{y})$ à un instant t_i quelconque sont stockées dans les variables `x, y, xp` et `yp`. À chaque pas de temps de la récurrence, l'instant t_i est remplacé par l'instant t_{i+1}. L'historique de la trajectoire est alors réalisé en ajoutant les valeurs `x, y, xp` et `yp` aux quatre tableaux de la position (`tab_x` et `tab_y`) et de la vitesse (`tab_xp` et `tab_yp`) de l'objet à l'instant t_{i+1}. On utilisera pour cela la méthode `append` adaptée aux tableaux `python`.

6.2 Consignes propres à la programmation des fonctions nécessaires

Les fonctions utilisées pour implémenter les formules de récurrence ainsi que les fonctions

`def f1 (xp, yp)` et `def f2 (xp, yp)`

— renvoyant la valeur des fonctions définies par le système d'équations (5) — seront définies en début de programme (après les commandes d'import de modules).

- Pour appliquer la méthode d'Euler, définir la fonction
  `def euler (dt, x, y, xp, yp)`
  qui calcule la position et la vitesse du projectile au pas de temps $i + 1$ à l'aide du système d'équations (6).
  Cette fonction retournera un tableau contenant les valeurs x_{i+1}, y_{i+1}, p_{i+1} et q_{i+1}
- Pour appliquer la méthode d'Euler-Cromer, faire appel à la fonction `euler_cromer` qui permet d'intégrer le système (7) de la même manière que précédemment :
  `def euler_cromer (dt, x, y, xp, yp)`
- Les fonctions `euler` et `euler_cromer` feront appel à `f1` et `f2`.
- Pour la méthode de Runge-Kutta d'ordre 2, écrire la fonction
  `def rk2 (dt, x , y, xp , yp)`
  qui calcule la position et la vitesse du projectile au pas de temps $i + 1$ à l'aide du système d'équations (17). La fonction `rk2` fera appel à `f1` et `f2`.

6.3 Test des différentes méthodes : programmation

Dans le fichier `balistique1.py`, définir les constantes suivantes : `MASSE, DT, V0` et `G` telles que :

$$\texttt{MASSE} = 1 \text{ kg} \,;\, \texttt{DT} = 0.001 \text{ s} \,;\, \texttt{G} = 9.81 \text{ m/s}^2 \,;\, \texttt{V0} = 70 \text{ m/s}.$$

Pour chacune des trois méthodes : initialiser la valeur de l'angle de tir à $\theta = \pi/4$ et stocker les x_i et y_i pour chaque pas de temps dans des tableaux de résultats comme indiqué en début d'énoncé (par exemple `tab_x_euler`, `tab_x_cromer`, `tab_x_rk2`, `tab_y_euler`...). Faire le calcul avec `ETA` $= 0.01$ kg/s puis avec `ETA` $= 0.0$ kg/s pour comparer les deux courbes $y = f(x)$.

Visualiser les trajectoires. Sauvegarder les courbes dans trois fichiers : `euler_graphe1.pdf`, `cromer_graphe1.pdf` et `rk2_graphe1.pdf`.

6.4 Comparaison avec `scipy`

Utiliser la fonction `scipy.integrate.ode` pour produire les mêmes tableaux que précédemment avec la méthode Runge Kutta d'ordre 4. Pour cela, on se référera à la documentation officielle de la fonction disponible sur le web :
`https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.ode.html`

6.5 Point d'impact

Copier le fichier `balistique1.py` sous `balistique2.py`.
On souhaite tracer la courbe du point d'impact x_{max} en fonction de l'angle de tir θ. Dans le fichier source `balistique2.py`, faire varier l'angle de tir de 0 à 90° par pas de 1 degré et stocker dans un tableau `tab_rk2_impact` (si la méthode de Runge-Kutta d'ordre 2 est choisie) la distance d'impact en fonction de l'angle de tir. Pour l'une des méthodes d'intégration de votre choix, tracer et sauvegarder la courbe $x_{max} = f(\theta)$.

6.6 Vitesse de convergence

Copier le fichier `balistique2.py` sous `balistique3.py`.
Dans le fichier `balistique3.py`, on fait varier la valeur du pas de temps pour une méthode donnée. Pour chaque valeur, on fait varier l'angle de tir θ de 0 à 90° par pas de 1 degré ($N = 91$) et on calcule l'«erreur» moyenne $\langle\varepsilon\rangle_\theta$ définie comme :

$$\langle\varepsilon\rangle_\theta = \frac{1}{N}\sum_{i=1}^{N}\left|x_{max}(\theta_i) - x_{max}^{exact}(\theta_i)\right|$$

Les valeurs de $x_{max}^{exact}(\theta_i)$ sont celles obtenues pour `DT=0.0001` avec la fonction `scipy.integrate.ode` préalablement stockées dans un tableau `numpy array` de 91 éléments appelé `xmax_converge`. Pour faire varier le pas de temps on utilise une variable `delta` telle que

$$\texttt{delta} = \frac{\texttt{DT}}{a^n} \tag{18}$$

où $a = 1.1$ et $n = 1, 2, 3, ...$ est un entier supérieur à 1. La constante `DT` est fixée à 1.0. Pour chacune des trois méthodes d'intégration, faire varier la valeur du pas de temps `delta` en utilisant la formule (18). Utiliser une boucle «tant que» dont la condition de sortie est vérifiée pour $\langle\varepsilon\rangle_\theta < 0.1$. Lorsque la précision est atteinte, afficher la valeur du pas temps `delta` ainsi que la valeur finale de `n`, stocker les valeurs obtenues pour les 3 méthodes en commentaire dans votre programme. Tracer les 3 courbes de convergence de l'erreur en fonction de n dans le même graphe `comparaison_convergence.pdf`. Conclure sur la vitesse de convergence des trois méthodes d'intégration.

6.7 Stabilité des algorithmes proposés

Pour aller plus loin dans l'étude de la stabilité des algorithmes proposés, on rajoute au fichier de sortie l'énergie potentielle, l'énergie cinétique, et l'énergie totale du projectile :

Pour les trois méthodes d'intégration, en fixant $\theta = \pi/3$ et `ETA` $= 0.0$ kg/s, et pour `DT` $=$ 0.075 s, représenter l'énergie totale en fonction du temps. Commenter.

6.8 Cas d'un frottement quadratique en vitesse

On améliorera la description physique du frottement en passant d'une dépendance linéaire à une dépendance quadratique dans la vitesse. On ne peut plus alors résoudre analytiquement et simplement le problème. Ainsi, pour cette question on prendra :

$$\mathbf{f} = -\lambda|\mathbf{v}|\mathbf{v} \tag{19}$$

Adapter le programme pour refaire des simulations avec ce type de frottement. On définira les nouvelles fonctions `f1_quadra` et `f2_quadra`. Prendre $\lambda = 0.1$ kg.m^{-1}.

Dans le cas où le coefficient de frottement est non-nul, comment pourrait-on procéder afin de tester le comportement des méthodes. Implémenter cette procédure et commenter les résultats.

Annexe : Intégration à l'aide de la méthode de Runge–Kutta d'ordre 4

Les méthodes de Runge-Kutta concernent les équations différentielles de la forme :

$$\frac{dy}{dt} = F(t, y)$$

où F est une fonction suffisamment régulière de la variable d'intégration t et de la fonction recherchée y. En combinant les développements de Taylor de F en des points bien choisis, on peut obtenir des relations d'extrapolation plus précises que les formules précédentes. La combinaison la plus souvent employée est une formulation d'ordre 4 par rapport au pas d'intégration temporelle h, l'erreur commise est alors en $\Theta(h^5)$:

$$y_{i+1} \simeq y_i + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}$$

avec

$$\begin{aligned} k_1 &= h\,F(t_i,\, y_i) \\ k_2 &= h\,F(t_i + \frac{h}{2},\, y_i + \frac{k_1}{2}) \\ k_3 &= h\,F(t_i + \frac{h}{2},\, y_i + \frac{k_2}{2}) \\ k_4 &= h\,F(t_i + h,\, y_i + k_3) \end{aligned}$$

Ces formules restent vraies pour des systèmes vectoriels différentiels en remplaçant les scalaires par des vecteurs (notés en lettres grasses) partout où cela est nécessaire.

$$\frac{d\mathbf{w}}{dt} = \mathbf{F}(t,\, \mathbf{w}) \quad \Rightarrow \quad \left\{ \begin{array}{lcl} \mathbf{k}_1 &=& h\,\mathbf{F}(t_i, \mathbf{w}_i) \\ \mathbf{k}_2 &=& h\,\mathbf{F}(t_i + \dfrac{h}{2}, \mathbf{w}_i + \dfrac{\mathbf{k}_1}{2}) \\ \mathbf{k}_3 &=& h\,\mathbf{F}(t_i + \dfrac{h}{2}, \mathbf{w}_i + \dfrac{\mathbf{k}_2}{2}) \\ \mathbf{k}_4 &=& h\,\mathbf{F}(t_i + h, \mathbf{w}_i + \mathbf{k}_3) \end{array} \right. \tag{20}$$

Pour la méthode de RK d'ordre 4, la valeur du vecteur $\mathbf{w}$ à l'instant t_{i+1} est donnée par :

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \frac{\mathbf{k}_1}{6} + \frac{\mathbf{k}_2}{3} + \frac{\mathbf{k}_3}{3} + \frac{\mathbf{k}_4}{6} \tag{21}$$

Ici, on peut définir le vecteur $\mathbf{w}$ par ses composantes [1] :

$$\mathbf{w} \;=\; \left\{ \begin{array}{lcl} w_0 &=& p \\ w_1 &=& x \\ w_2 &=& q \\ w_3 &=& y \end{array} \right. \qquad \text{avec} \qquad \dot{\mathbf{w}} \;=\; \mathbf{F}(t,\, \mathbf{w}) \quad \Rightarrow \quad \left\{ \begin{array}{lcl} \dot{w_0} &=& f_1(w_0) \\ \dot{w_1} &=& w_0 \\ \dot{w_2} &=& f_2(w_2) \\ \dot{w_3} &=& w_2 \end{array} \right. \tag{22}$$

On remarque que dans notre problème (cf. eq. (5)), les fonctions f_1 et f_2 ne dépendent pas du temps et que chacune ne dépend que d'une seule variable w_i.

---

1. On ne confondra pas l'indice de w qui repère la composante avec celui de $\mathbf{w}$ qui repère le pas d'intégration par rapport au temps.