DS – Saut de Félix Baumgartner et interpolation

Remarques : Chaque partie de chaque problème peut être traitée de façon indépendante. Vous pouvez travailler au choix sur Python 3.3 ou Scilab et vous pouvez consulter le formulaire Python/Scilab.

Il vous est demandé de rendre un programme commenté et exécutable sans erreur. Les questions notées ♣ sont à rédiger sur copie. Le programme sera enregistré au format "NOM_Prenom".

1 Chute libre de Félix Baumgartner

Le 14 octobre 2012, Felix Baumgartner réalise un saut en chute libre depuis une altitude vertigineuse de 39 000 m. Il dépasse le mur du son au cours de la chute, ce qui constitue une première !

Ce genre d'exploit est minutieusement préparé et des calculs préalables permettent de prévoir les temps de chute et les vitesses atteintes au cours du vol. Ces simulations permettent d'anticiper, puis sont comparées aux mesures faites durant le vol (figure 1).

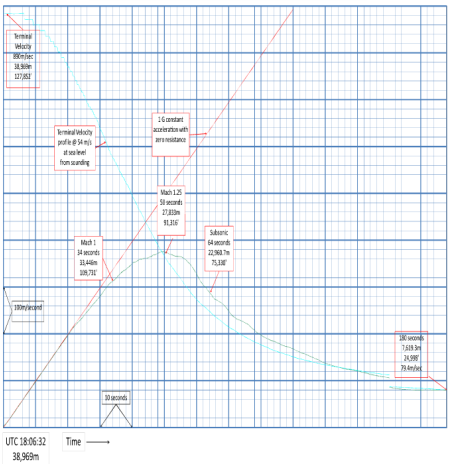

FIGURE 1 – Vue de la capsule et mesures de vol.

Ce problème vise à déterminer la loi de vitesse et d'altitude au cours de la chute, en tenant compte des paramètres physiques du vol : à cette altitude, la densité de l'air (et donc la trainée) change et la gravité évolue.

La trainée aérodynamique est modélisée par une loi quadratique (écoulement turbulent) sous la forme :

$$F_{\text{air}} = -\frac{1}{2}\rho(h)AC_x v^2$$

où $A = 0{,}45\,\text{m}^2$ est la surface apparente, $C_x = 0{,}8$ le coefficient de trainée dans l'air, $\rho(h)$ est la densité de l'air à l'altitude h et v la vitesse.

L'action de pesanteur est modélisée par une force centrale sous la forme :

$$P(h) = \frac{GMm}{(R+h)^2}$$

où $G = 6{,}73 \times 10^{-11}\,\text{N m}^2\,\text{kg}^{-2}$ est la constante de gravitation, $M = 6 \times 10^{24}\,\text{kg}$ la masse de la terre, $m = 80\,\text{kg}$ la masse du parachutiste, $R = 6{,}371 \times 10^{6}\,\text{m}$ le rayon de la terre et h l'altitude.

L'application du principe fondamental de la dynamique, projeté sur un axe vertical orienté vers le bas, s'écrit :

$$m\frac{dv}{dt} = -\frac{1}{2}\rho(h)AC_x v^2 + \frac{GMm}{(R+h)^2}$$

L'évolution de la densité de l'air en fonction de l'altitude h est approchée par la loi expérimentale :

$$\rho(h) = 1.433 * e^{-0.00013*h} \quad \text{en kg m}^{-3}$$

1.1 Petit entrainement préalable

Avant de réaliser l'intégration de l'équation différentielle précédente (assez complexe), on considère l'équation plus simple suivante, correspondant plutôt à la chute d'un caillou dans de l'eau :

$$m\frac{dv}{dt}(t) = mg - fv(t) \qquad \text{où } m = 1\,\text{kg},\ g = 9{,}81\,\text{m s}^{-2} \text{ et } f = 10\,\text{N s m}^{-1}.$$

Q1. Déterminer par une intégration par le schéma d'Euler *explicite* l'évolution de la vitesse $v(t)$ du caillou sur 5 secondes, en choisissant un pas de temps de 0,1 s.

Q2. Déterminer par une intégration par le schéma d'Euler *implicite* l'évolution de la vitesse $v(t)$ du caillou sur 5 secondes, en choisissant un pas de temps de 0,1 s.

Q3. Tracer sur la figure 1 les deux courbes obtenues.

1.2 Saut de Felix Baumgartner

Q4. Tracer sur la figure 2 la loi d'évolution de la densité de l'air en fonction de l'altitude sur l'intervalle $[0\,39\,000\,m]$ et vérifier qu'elle ne peut pas être considérée comme constante.

L'équation différentielle est non linéaire et du second ordre (l'expression du frottement de l'air dépend de l'altitude). Pour l'intégration numérique, on pose le vecteur X suivant afin de mettre l'équation différentielle sous la forme d'une ODE :

$$X = \begin{bmatrix} v \\ h \end{bmatrix} \qquad \text{et} \qquad \frac{dX}{dt} = F(t, X)$$

L'altitude est bien évidement liée à la vitesse par l'équation $\frac{dh}{dt}(t) = -v(t)$.

Q5. ♣ Mettre le système d'équations physiques sous la forme précédente et exprimer le vecteur $F(t, X)$.

Q6. Déterminer par une intégration par le schéma d'Euler *explicite* l'évolution de la vitesse $v(t)$ et de la vitesse $h(t)$ sur les 120 premières secondes.

1.3 Comparaison aux mesures

Le fichier *mesure_chute_Baumgartner_TVH.csv* contient les données mesurées lors du saut par la centrale inertielle et le module GPS sur les 120 premières secondes de vol. Les trois colonnes du fichiers sont le temps t, la vitesse v et l'altitude h.

Q7. Lire le fichier et tracer la courbe de vitesse sur la figure 3. Si vous avez traité la partie précédente, superposer la courbe de vitesse mesurée et la courbe de vitesse simulée sur la figure 3.

2 Interpolation

Le design, l'imagerie ou encore la conception assistée par ordinateur manipulent beaucoup de courbes et surfaces définies par un ensemble de points de passage. Pour obtenir des valeurs entre les points de passage, il faut *interpoler*.

Les exercices suivants visent à mettre en oeuvre deux types d'interpolation sur les fonctions : l'interpolation polynomiale et l'interpolation par des splines. Nous verrons que l'interpolation polynomiale est simple mais qu'elle peut conduire à des oscillations dans certains cas. L'interpolation par splines est plus satisfaisante pour la plupart des situations.

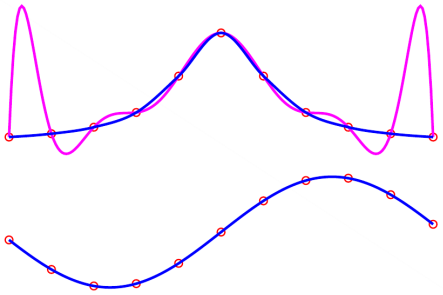

FIGURE 2 – Courbes test obtenues après interpolation polynomiale et spline.

Deux cas tests sont envisagés : une fonction sinus (`f1(x)`) et une fonction de Runge (`f2(x)`), échantillonnées en 11 points entre -1 et 1. Le programme suivant (proposé dans les deux langages) permet de tracer ces fonctions :

```python
def f1(x):
    return 0.5*sin(3*x)-0.8

def f2(x):
    return 1/(1+16*x**2)

# Echantillonnage de 11 points :
X=linspace(-1,1,11)
Y1=f1(X)
Y2=f2(X)

# Echantillonnage fin pour le tracé de la courbe
x=linspace(-1,1,200)

# Tracés
figure(1); clf();
plot(x,f1(x),"g") # tracé des courbes test en vert
plot(x,f2(x),"g")
plot(X,Y1,"ro")   # tracé des 11 points en rouge
plot(X,Y2,"ro")
```

```scilab
function y=f1(x)
    y=0.5*sin(3*x)-0.8;
endfunction
function y=f2(x)
    y=1 ./(1+16*x.^2);
endfunction

// Echantillonnage de 11 points :
X=(-1:0.2:1)';
Y1=f1(X);
Y2=f2(X);

// Echantillonnage fin pour le tracé de la courbe
x=(-1:0.01:1)';

// Tracés
figure(1); clf ;
plot(x,f1(x),"g") // tracé des courbes test en vert
plot(x,f2(x),"g")
plot(X,Y1,"ro")   // tracé des 11 points en rouge
plot(X,Y2,"ro")
```

2.1 Interpolation polynomiale

L'interpolation polynomiale consiste à déterminer le polynôme $P(x)$ de degré le moins élevé, passant par tous les points. S'il y a n points de passage, il suffit d'un polynôme de degré $n-1$, dont les n coefficients a_i sont solution du système :

$$\begin{cases} a_0 + a_1 x_1 + \cdots + a_{n-1} x_1^{n-1} = y_1 \\ \vdots \\ a_0 + a_1 x_j + \cdots + a_{n-1} x_j^{n-1} = y_j \\ \vdots \\ a_0 + a_1 x_n + \cdots + a_{n-1} x_n^{n-1} = y_n \end{cases}$$

Q8. ♣ Mettre le problème sous la forme matricielle $\mathbb{M}A = B$ où A est le vecteur colonne des coefficients du polynôme recherché : $A = (a_0, a_1, \cdots, a_{n-1})^T$. Exprimer la matrice $\mathbb{M}$ et le vecteur B.

Q9. Proposer un programme permettant de calculer les coefficients de $\mathbb{M}$.

Une fois la matrice $\mathbb{M}$ et le vecteur B déterminés, le calcul des coefficients A du polynôme se fait par résolution du système : sous Scilab, `A=M\B` et sous Python, `A=linalg.solve(M,B)`.

Q10. Proposer un programme permettant de déterminer le vecteur colonne y correspondant à la valeur du polynôme pour chaque valeur du vecteur colonne x : $y = P(x)$.

Q11. Proposer alors une fonction `interp_poly(x,X,Y)` prennant en argument deux vecteurs de points de passages X et Y et un vecteurs x quelconque, et renvoyant le vecteur y correspondant à l'image de x par l'interpolation

polynomiale.

Q12. Mettre en oeuvre l'interpolation polynomiale sur les deux fonctions test et vérifier que l'interpolation polynomiale est très bonne sur la fonction sinus, mais qu'elle présente des oscillations sur la fonction de Runge.

Le paragraphe suivant envisage le cas d'une interpolation par splines, mieux adaptée dans la plupart des situations.

2.2 Interpolation par une spline

L'interpolation spline de n points (X, Y) consiste à déterminer une interpolation polynomiale par morceaux : sur chaque morceau $[X_i X_{i+1}]$, l'interpolation est représentée par un polynôme $P_i(x)$ de degrés trois (4 paramètres) satisfaisant aux conditions suivantes, $\forall i \in [\![2, n-2]\!]$:

— le polynôme passe par le point (X_i, Y_i), soit $P_i(X_i) = Y_i$,

— le polynôme passe par le point (X_{i+1}, Y_{i+1}), soit $P_i(X_{i+1}) = Y_{i+1}$,

— la pente (la dérivée) en X_i vaut $P_i'(X_i) = \dfrac{Y_{i+1} - Y_{i-1}}{X_{i+1} - X_{i-1}}$,

— la pente (la dérivée) en X_{i+1} vaut $P_i'(X_{i+1}) = \dfrac{Y_{i+2} - Y_i}{X_{i+2} - X_i}$.

Cette interpolation est de classe C^1, c'est-à-dire qu'elle est continue et que sa dérivée est aussi continue (aucun point anguleux).

Pour les points extrémités (points 1 et n), la pente ne peut pas être exprimée sous la forme précédente. La condition sur la pente est remplacée par une condition de courbure nulle :

— la courbure en X_1 est nulle, soit $P_1''(X_1) = 0$,

— la courbure en X_n est nulle, soit $P_{n-1}''(X_n) = 0$,

Les $n-1$ polynômes sont notés $P_i(x) = a_0^{(i)} + a_1^{(i)} x + a_2^{(i)} x^2 + a_3^{(i)} x^3$. Le vecteur colonne des paramètres de la spline est noté :

$$A = [a_0^{(1)}, a_1^{(1)}, a_2^{(1)}, a_3^{(1)}, a_0^{(2)}, a_1^{(2)}, a_2^{(2)}, a_3^{(2)}, \cdots, a_0^{(n-1)}, a_1^{(n-1)}, a_2^{(n-1)}, a_3^{(n-1)}]$$

Q13. ♣ Exprimer les deux conditions de passage en fonction des coefficients du polynôme et des coordonnées des points X_i, X_{i+1}, Y_i et Y_{i+1}. Exprimer de même les conditions sur la pente puis mettre les quatre équations sous forme matricielle :

$$\mathbb{M}_i \begin{bmatrix} a_0^{(i)} \\ a_1^{(i)} \\ a_2^{(i)} \\ a_3^{(i)} \end{bmatrix} = B_i$$

Exprimer la matrice $\mathbb{M}_i$ et le vecteur B_i.

Q14. ♣ Adapter le résultat précédent pour exprimer les matrices $\mathbb{M}_1$ et $\mathbb{M}_{n-1}$, ainsi que les vecteur B_1 et B_{n-1}, en replaçant la condition sur la pente par une condition de courbure nulle aux points extrémités.

Q15. ♣ Traduire le problème global sous forme matricielle $\mathbb{M}A = B$, en exprimant par blocs la matrice $\mathbb{M}$ et le vecteur B.

Q16. Proposer un programme permettant de calculer la matrice $\mathbb{M}$ et le vecteur B, puis de calculer A par résolution du système.

Q17. Proposer un programme permettant de déterminer le vecteur colonne y correspondant à la valeur du polynôme pour chaque valeur du vecteur colonne `x` : $y = P(x)$.

Q18. Proposer alors une fonction `interp_spline(x,X,Y)` prennant en argument deux vecteurs de points de passages X et Y et un vecteurs x quelconque, et renvoyant le vecteur y correspondant à l'image de x par l'interpolation polynomiale.

Q19. Mettre en oeuvre l'interpolation spline sur les deux fonctions test et vérifier que l'interpolation est très bonne dans les deux cas, contrairement à l'interpolation polynomiale de l'exercice précédent.