

# Modélisation d'un mouvement de chute

F.Baumgartner a réalisé un saut depuis une altitude de 38969m. Pendant cette phase de chute le champ de pesanteur s'appliquant au sauteur a varié, la masse volumique de l'air l'environnant également



Le poids  $P$  du sauteur, de masse  $m = 110\text{kg}$ , est fonction de l'altitude  $z$  de ce dernier vis-à-vis de la surface de la Terre :  $\vec{P} = -\frac{GMm}{(R+z)^2} \vec{u}_z$  (où  $\vec{u}_z$  est suivant la verticale ascendante).

Où  $G = 6,37 \times 10^{-11} \text{N} \cdot \text{m}^2 \cdot \text{kg}^{-2}$ ,

$M = 6 \times 10^{24} \text{kg}$ ,

et  $R = 6,371 \times 10^6 \text{m}$ .

On propose un modèle pour exprimer la force de frottement de la forme :  $\vec{F} = ke^{-\alpha z} v^2 \vec{u}_z$  où  $k = 0,2 \text{N} \cdot \text{s}^2 \cdot \text{m}^{-2}$  et  $\alpha = 10 \times 10^{-5} \text{m}^{-1}$ .

1) Obtenir l'équation scalaire du mouvement de chute verticale de Felix.

L'équation précédente est non linéaire mais il est possible de la résoudre en se ramenant à un problème de Cauchy pour lequel il existe un vecteur  $\vec{X} = \begin{bmatrix} z \\ v \end{bmatrix}$  et une fonction  $\vec{f}$  telle que :  $\frac{d\vec{X}}{dt} = \vec{f}$

2) Trouver l'expression de  $\vec{f}$

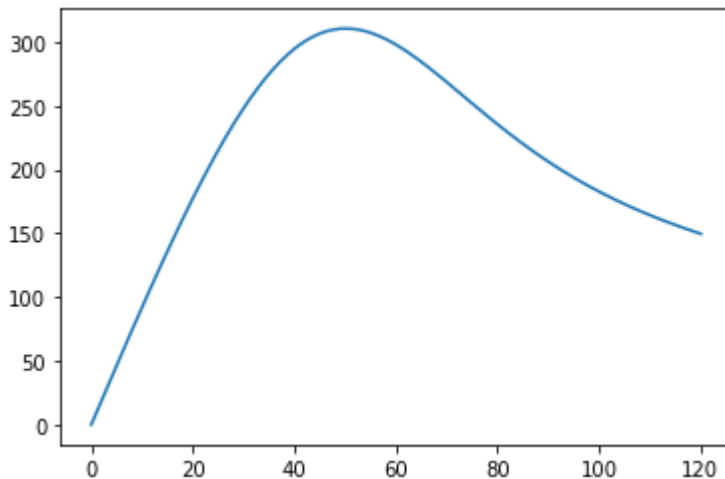
3) Déterminer avec une intégration par le schéma d'Euler explicite l'évolution de la valeur absolue de la vitesse  $v(t)$  ainsi que la position  $z(t)$ . On prendra un pas temporel  $T_e = 0,5\text{s}$  sur 120s (241 échantillons) et une vitesse initiale nulle.

In [1]:

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 Te=0.5
5 N=241
6 t=np.linspace(0,120,241)
7 X=np.zeros((2,N))
8 G=6.37*10**-11
9 M=6*10**24
10 m=110
11 R=6371*10**3
12 k=0.2
13 alpha=10**-4
14 h=38969
15 X[0,0]=h
16 for i in range(len(t)-1):
17     X[0,i+1]=X[0,i]+Te*X[1,i]
18     X[1,i+1]=X[1,i]+Te*(-G*M/(R+X[0,i])**2+k/m*np.exp(-alpha*X[0,i])*X[1,i]**2)
19 plt.plot(t,abs(X[1,:]))
20 plt.show()

```



Vous avez à disposition le fichier felix.csv dans lequel vous avez accès aux temps d'acquisition, aux vitesses et aux positions expérimentales de F. Baumgartner.

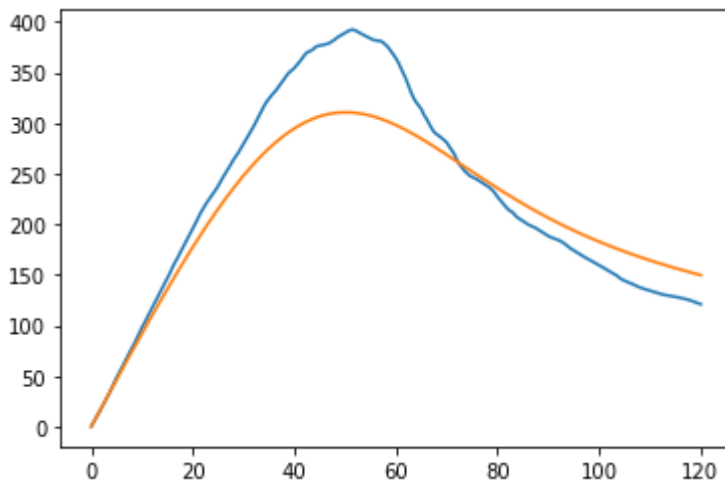
4) Superposer sur un même graphe la vitesse mesurée et la vitesse théorique et commenter votre modèle proposé pour la force de frottement.

In [2]:

```

1 f=open("C:/Users/User/Documents/travail/TSI2_2022_2023/physique/oraux/physiqu
2 tab=f.readlines()
3 tab_v=np.zeros(len(tab))
4 for i in range(len(tab)):
5     ligne=tab[i]
6     liste=ligne.split(",")
7     v=float(liste[1])
8     tab_v[i]=v
9 plt.plot(t,tab_v)
10 plt.plot(t,abs(X[1,:]))
11 plt.show()

```



Afin d'améliorer le modèle précédent, nous allons mettre en œuvre la méthode des moindres carrés et chercher la valeur de  $\alpha$  la plus adaptée à notre cas. Toujours avec le schéma d'Euler explicite, on peut écrire la vitesse comme une fonction dépendant de  $\alpha$ .

In [3]:

```

1 def v(alpha) :
2     X=np.zeros((2,N))
3     X[0,0]=h
4     X[1,0]=0
5     for i in range(len(t)-1):
6         X[0,i+1]=X[0,i]+Te*X[1,i]
7         X[1,i+1]=X[1,i]+Te*(-G*M/(R+X[0,i])**2+k/m*np.exp(-alpha*X[0,i]))*X[1,i]
8     return -X[1,:]
9

```

4) Ecrire une fonction erreur(alpha) qui renvoie la somme du carré des écarts entre les points expérimentaux et la loi souhaitée.

In [4]:

```

1 def erreur(alpha) :
2     return np.sum((v(alpha)-tab_v)**2)
3 print(erreur(10**-4))

```

347240.29304076516

Typesetting math: 100%

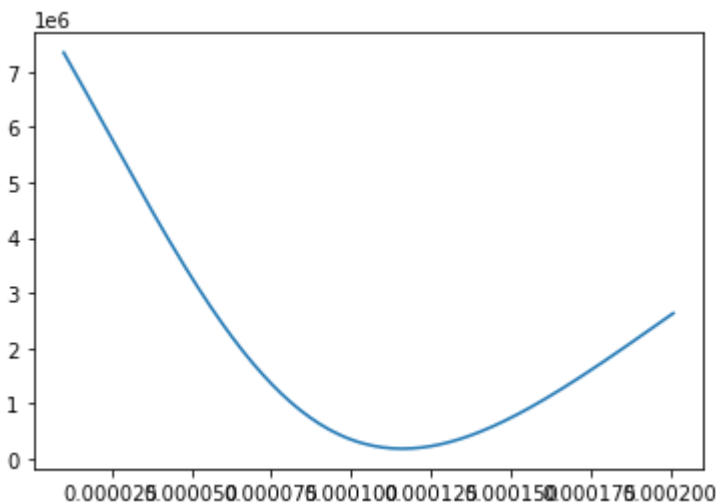
Cette fonction erreur doit être minimisée, c'est-à-dire qu'il faut trouver une valeur du paramètre  $\alpha$  qui minimise l'erreur dans l'intervalle  $I = [10^{-5}, 20 \times 10^{-5}]$

5) Ecrire une fonction Liste\_erreur(tab\_alpha) qui retourne une liste de valeurs d'erreurs (notée liste\_e) associée à un tableau de valeurs de  $\alpha$  (appelé tab\_alpha) comprise dans  $I$  (on prendra un pas de  $10^{-6}$ )

6) Obtenir un graphe donnant l'évolution de l'erreur entre le modèle et les résultats expérimentaux pour  $\alpha$  dans  $I$

In [7]:

```
1 pas=10**-6
2 tab_alpha=np.arange(10**-5,20*10**-5+pas,pas)
3 def Liste_erreur(tab_alpha):
4     liste_e=[]
5     for i in tab_alpha:
6         liste_e.append(erreur(i))
7     return liste_e
8 plt.plot(tab_alpha,Liste_erreur(tab_alpha))
9 plt.show()
```



7) Expliquer l'intérêt des lignes de code ci-dessous:

In [8]:

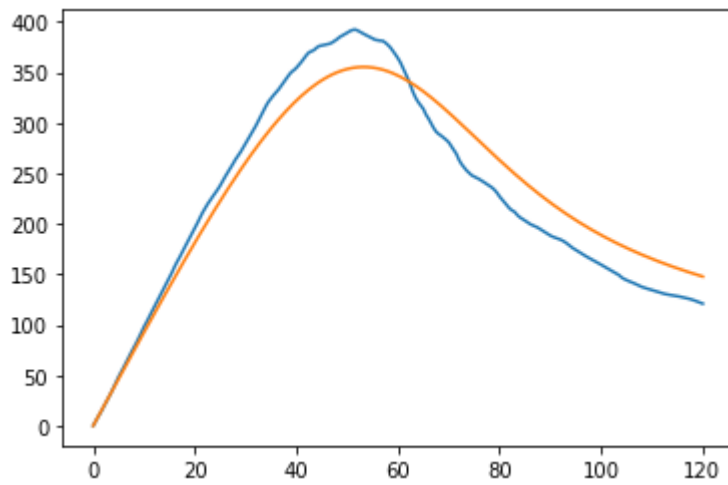
```
1 tab_alpha=np.arange(10**-5,20*10**-5+pas,pas)
2 liste_e=Liste_erreur(tab_alpha)
3 i=0
4 while liste_e[i+1]<liste_e[i]:
5     i=i+1
6 print(tab_alpha[i])
7
```

0.00011600000000000005

On trouve la valeur de alpha optimale :

In [9]:

```
1 plt.plot(t,tab_v)
2 plt.plot(t,v(0.00011600000000000005))
3 plt.show()
```



In [ ]:

1