

Document-réponse

Nom et Prénom : CORRECTION

Question 1.

Il faut dissiper : $E_c = \frac{1}{2} \cdot M \cdot v_0^2 \approx 1,04 \text{ GJoules !}$

Question 2.

```

1 def F(vi):
2     return A0 + A1*vi**2
3
4 def eulerExplicite(Cini, FI):
5     V = [Cini] # liste des vitesses
6     t = [0]     # liste de temps
7     pas = 0.001 # dt en seconde
8     v = Cini    # vitesse à l'instant i
9     Wf = [0]    # liste d'énergie due aux actions de frottement
10    Wa = [0]     # liste d'énergie due aux actions aérodynamique
11    x = [0]      # liste de distance
12    Ec0 = 1/2*M*v0**2 # énergie cinétique initiale
13    while :      # A COMPLETER
14        v =      # A COMPLETER
15        x.append(    ) # A COMPLETER
16        V.append(v)
17        Wf.append(Wf[-1]+(A0*M*v*pas))
18        Wa.append(    ) # A COMPLETER
19        t.append(t[-1]+pas)
20    return V, x, t, Wf, Wa
21
22 V, x, t, Wf, Wa = eulerExplicite(v0,F)

```

$\rightarrow$ fonction qui renvoie $a(t) = A_0 + A_1 \cdot v^2(t)$

$v_{\text{suivant}} = v_{\text{précédent}} + \Delta t \cdot \left. \frac{dv}{dt} \right|_{\text{précédent}}$

$W_a = A_1 \cdot m \cdot \int_0^T v^3(t) \cdot dt$

```
ligne 13 while v > 0 :
```

```
ligne 15  x.append(x[-1] + pas * v)
```

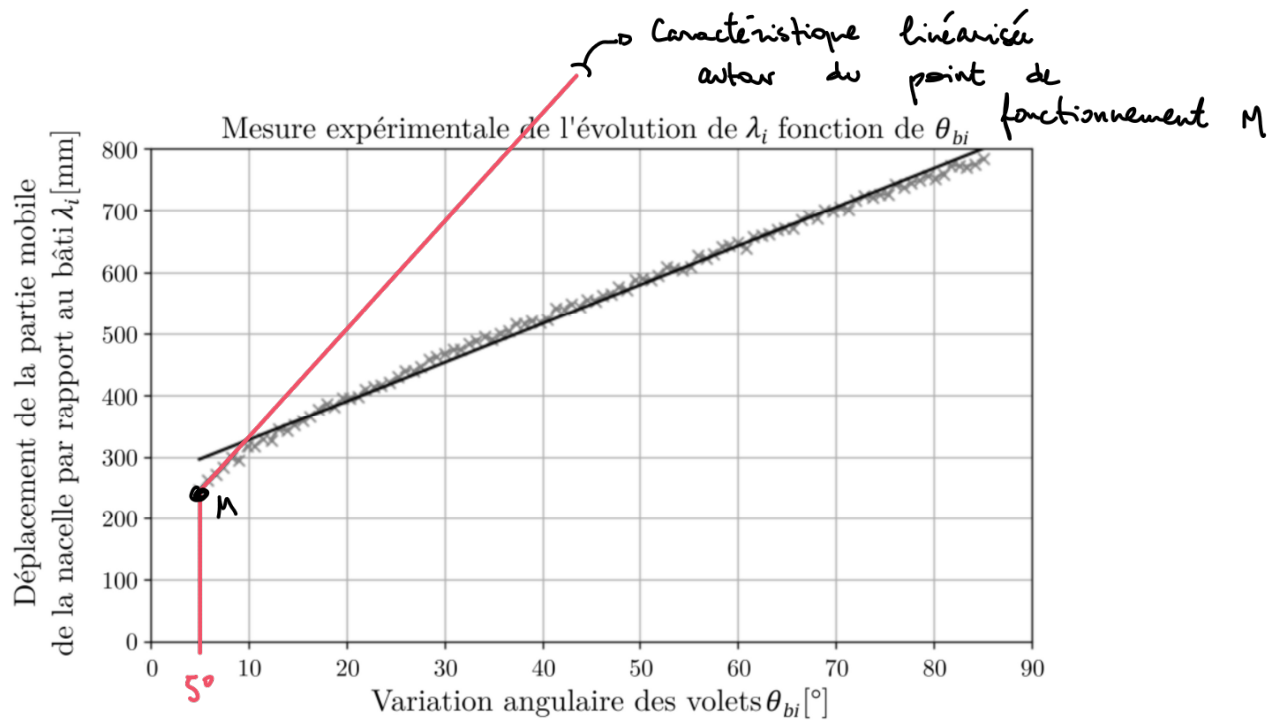
ligne 14 $v = v + pas * F(v)$

```
ligne 18 Na.append(Na[-1] + A1 * M * p0 * v**3)
```

Question 3.

	c) Sans inverseurs de poussée	d) Avec inverseurs de poussée
Énergie dissipée par les freins mécaniques (Joules)	429 MJ	186 MJ
Énergie dissipée par la traînée aérodynamique - sans inverseurs de poussée (Joules)	614 MJ	357 MJ
Énergie dissipée par les inverseurs de poussée (Joules)	0	500 MJ

Question 4.



Question 5.

Avantage	Le modèle est plutôt fiable sur l'ensemble de l'intervalle $[5^\circ, 85^\circ]$.
Inconvénient	Au voisinage du point de fonctionnement, le comportement semble non-linéaire. Le modèle est moins adapté.

Question 6.

```

1 import pylab as pl
2 ## Importation de la classe LinearRegression
3 from sklearn.linear_model import LinearRegression
4 ## Importation des données
5 f = open('mesure.txt','r')
6 L = f.readlines()
7 f.close()
8
9 theta_regression, lbda_regression = [], []
10 for i in range(len(L)):
11     theta, lbda = L[i].strip('\n').split(';')
12     theta_regression.append(float(theta))
13     lbda_regression.append(float(lbda))
14 X = pl.array(theta_regression)          # en degré
15 Y = pl.array(lbda_regression)
16
17 # mise en forme des données (theta) d'entrainements
18 X_train = X.reshape((-1,1))
19 # instancier modèle
20 model_linReg = LinearRegression()
21 # entrainer le modèle
22 model_linReg.                # A COMPLETER en utilisant la bonne méthode
23 # récupération des paramètres du modèle
24 a =                           # A COMPLETER
25 b =                           # A COMPLETER
26 def predict(x):
27     return a * x + b
28 fitLine = predict(X)
29
30 pl.plot(X, Y, '--k')
31 pl.plot(X, fitLine, c='grey')
32 pl.show()

```

ligne 22 `model_linReg.fit(X_train, Y)`

ligne 24 `a = model_linReg.coef_[0]`

ligne 25 `b = model_linReg.intercept_`

FIN