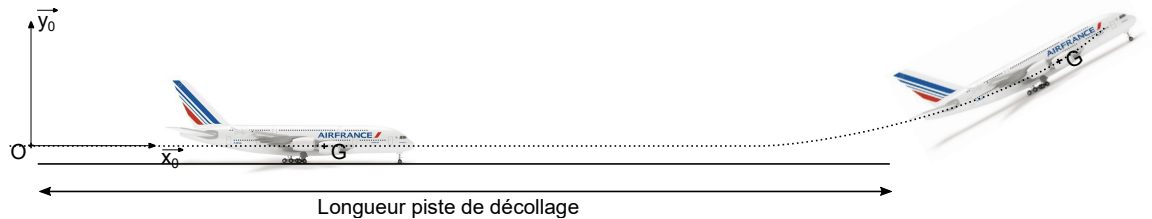


Simulation du décollage d'un avion

On cherche à simuler la trajectoire d'un avion lors de son décollage. Afin d'obtenir les équations de mouvement de l'avion, il est nécessaire d'utiliser le principe fondamental de la dynamique. Dans le cadre d'un modèle plan, on obtient donc trois équations différentielles. Moyennant quelques simplifications et hypothèses, il est possible de se ramener à deux équations différentielles : l'une représentant le mouvement de l'avion dans la direction horizontale et l'autre dans la direction verticale.



On cherche d'abord à déterminer la longueur de piste nécessaire afin que l'avion atteigne sa vitesse de décollage avant la fin de la piste.

Le constructeur annonce que l'avion (ici un A380) a les caractéristiques suivantes :

Masse à vide	$M = 550$ tonnes
Vitesse de croisière	900 km/h
Altitude de croisière	10 700 m
Vitesse de décollage	$v_{\text{décollage}} = 350$ km/h
Poussée des moteurs au démarrage	$F_m = 1210$ kN
Poussée des moteurs en croisière	$F_m = 250$ kN
Coefficient de frottement quadratique	$f_x = 1.7 \text{ N}/(\text{m/s})^2$

L'équation de mouvement dans la direction horizontale est la suivante :

$$F_m = M \cdot \ddot{x} + f_x \cdot \dot{x}^2 = M \cdot \dot{v}_x + f_x \cdot v_x^2$$

- avec le changement de variable $v_x = \dot{x}$ où x est l'abscisse du point G (centre de gravité de l'avion).
- on supposera également que F_m est égale à la poussée maximale si $v_x < v_{\text{décollage}}$ et égale à la poussée "de croisière" sinon.

On va résoudre cette équation différentielle non-linéaire en utilisant les modules de Python associés à la méthode d'Euler.

Toutes les bibliothèques nécessaires sont présentées sur les ressources fournies lors de l'oral des écoles centrales. Ces documents (ou du moins ceux nécessaires pour aujourd'hui) sont disponibles ici :

- pour l'analyse numérique : <https://www.concours-centrale-supelec.fr/CentraleSupélec/SujetsOral/Multi/Python-AN.pdf> (<https://www.concours-centrale-supelec.fr/CentraleSupélec/SujetsOral/Multi/Python-AN.pdf>)
- pour la réalisation de tracés : <https://www.concours-centrale-supelec.fr/>

Entrée [252]:

```
1 import numpy as np
2 import scipy.integrate as integr
3
4 Fm_max = 1210.e3 ## Force de poussée
5 Fm_croisiere = 250.e3 ## Force de poussée
6 v_decollage = 350*1000/3600 # en m/s
7 M = 550.e3 ## Masse
8 fx = 4. ## coefficient de frottement quadratique
9
10 ## nombre de subdivisions temporelles
11 n = 10000
12
13 ## Temps de simulation
14 T = 60.*20
15
16 ## Pas de temps
17 dt = T/n
18
19
```

Question 1. Déterminer la fonction F_{Hori} de telle sorte que $\dot{v}_x = F_{\text{Hori}}(v_x, t)$. Définir ensuite cette fonction ci-dessous :

Entrée [253]:

```
1 def Fhori(vx,t):
2     if vx<v_decollage:
3         Fm = Fm_max
4     else:
5         Fm = Fm_croisiere
6     return (Fm/M) - (fx/M)*vx**2
```

Question 2. Compléter les instructions ci-dessous afin de renvoyer la vitesse de l'avion et les temps correspondants. n représente ici le nombre de subdivisions temporelles.

Entrée [254]:

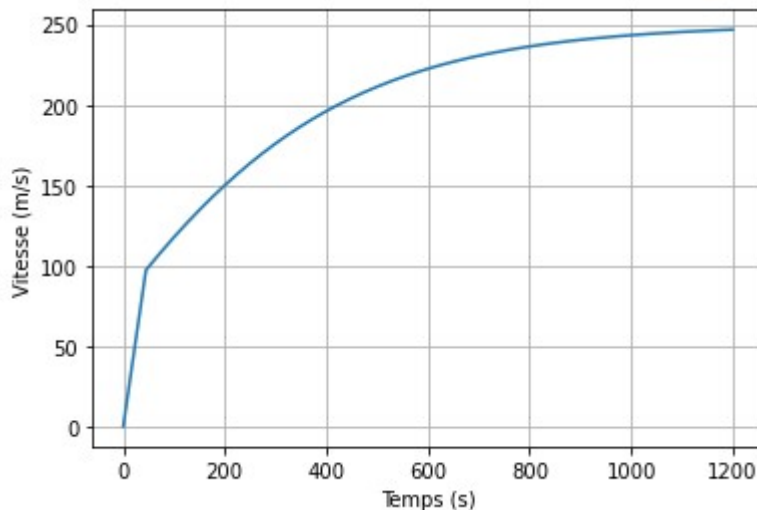
```
1 liste_T = np.arange(0, T+dt, dt)
2 vx0 = 0
3
4 liste_vx = integr.odeint(Fhori,vx0,liste_T)[: ,0]
5
6 print(liste_T)
7 print(liste_vx)
```

```
[0.00000e+00 1.20000e-01 2.40000e-01 ... 1.19976e+03 1.19988e+03
1.20000e+03]
[ 0.          0.26399996  0.52799981 ... 246.72377003 246.72518998
246.72660933]
```

Question 3. Tracer la vitesse de l'avion en fonction du temps. On affichera le nom des axes avec `plt.xlabel('nom et unité !')` et `plt.ylabel('nom et unité !')` où `plt` est l'importation de la bibliothèque `matplotlib.pyplot`.

Entrée [255]:

```
1 import matplotlib.pyplot as plt
2 plt.plot(liste_T,liste_vx)
3 plt.xlabel('Temps (s)')
4 plt.ylabel('Vitesse (m/s)')
5 plt.grid(True)
```



Question 4. On va maintenant obtenir la position horizontale de l'avion par intégration numérique avec une méthode des trapèzes. Compléter le code ci-dessous afin d'obtenir la liste des positions `liste_x`.

Entrée [256]:

```
1 liste_x = [0]
2 for i in range(1,n+1):
3     liste_x.append(liste_x[-1] + dt*(liste_vx[i] + liste_vx[i-1])/2)
4
5 #print(liste_x)
```

Question 5. Tracer maintenant la vitesse de l'avion en fonction de sa position. Ecrire les instructions permettant de déterminer numériquement une valeur approchée du temps de décollage et de la position atteinte par l'avion à cet instant.

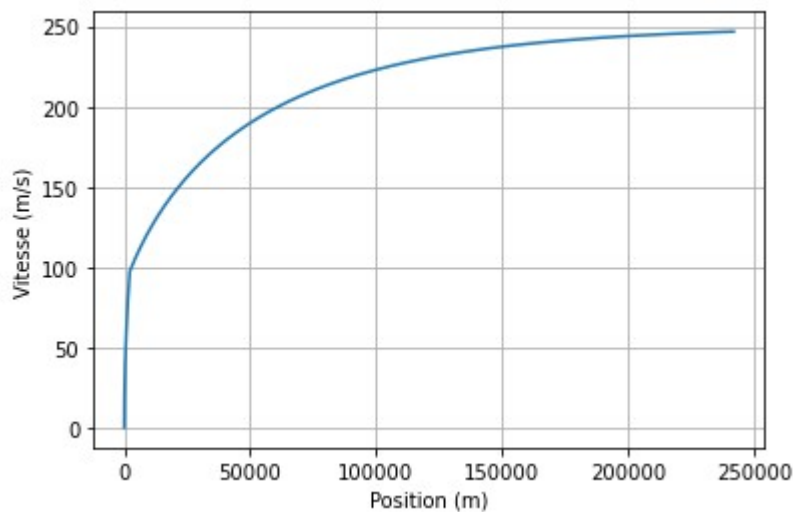
Question 5.bis. Créer une variable `tdecollage` qui sera le temps au moment du décollage.

Entrée [257]:

```
1 plt.plot(liste_x,liste_vx)
2 plt.xlabel('Position (m)')
3 plt.ylabel('Vitesse (m/s)')
4 plt.grid(True)
5
6
7 k = 0
8 v_decollage = 350 * 1000/3600 ## en m/s
9 while liste_vx[k] < v_decollage:
10     k = k + 1
11 print('distance décollage =',liste_x[k],'m')
12 tdecollage = liste_T[k]
13 print('temps décollage =',liste_T[k],'s')
```

distance décollage = 2192.1186688691737 m

temps décollage = 44.76 s



Question 6. Une piste de décollage classique mesure environ 3 km de long. Conclure.

On cherche maintenant à simuler le mouvement vertical de l'avion. L'équation de mouvement dans la direction verticale est la suivante :

- Si $F_p > M \cdot g$ Alors $F_p - M \cdot g = M \cdot \dot{v}_y + f_y \cdot v_y^2$
- Sinon $v_y = 0$

F_p est la portance de l'avion (homogène à une force). f_y est le coefficient quadratique de frottement dans la direction verticale, on prendra : $f_y = 200000 \text{ N}/(\text{m/s})^2$. Cette portance est une fonction de l'angle d'inclinaison des gouvernes de l'avion et de la vitesse horizontale de celui-ci. On écrira :

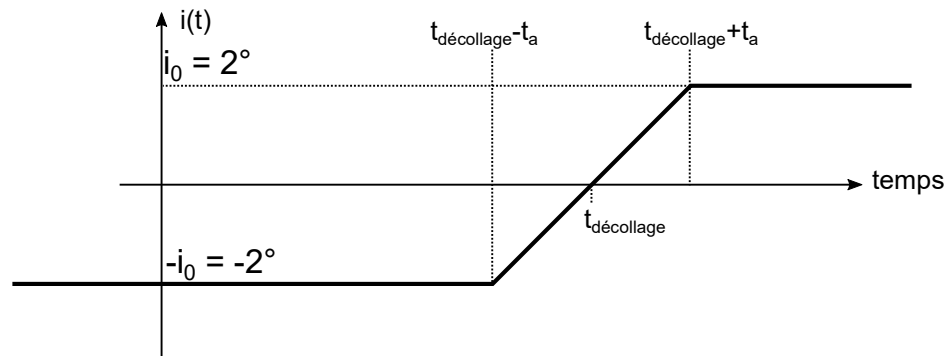
$$F_p = K \cdot k_i(i) \cdot v_x^2$$

avec i , l'angle d'inclinaison des gouvernes.

On donne $K = 290 \text{ N}/(\text{m/s})^2$, un coefficient relatif à l'aérodynamique des ailes.

L'angle d'inclinaison des gouvernes de l'avion est pilotée par le pilote. Si la vitesse de l'avion est inférieure à la vitesse de décollage, les gouvernes sont inclinées de telle manière que la force de portance soit négative. Cela permet de "plaquer" l'avion sur le sol. Si la

vitesse est suffisante, dans ce cas, il faut augmenter l'angle d'incidence de l'avion pour forcer le décollage. Le pilotage retenu est défini sur le graphique ci-dessous (on prendra $t_a = 10$ s) :



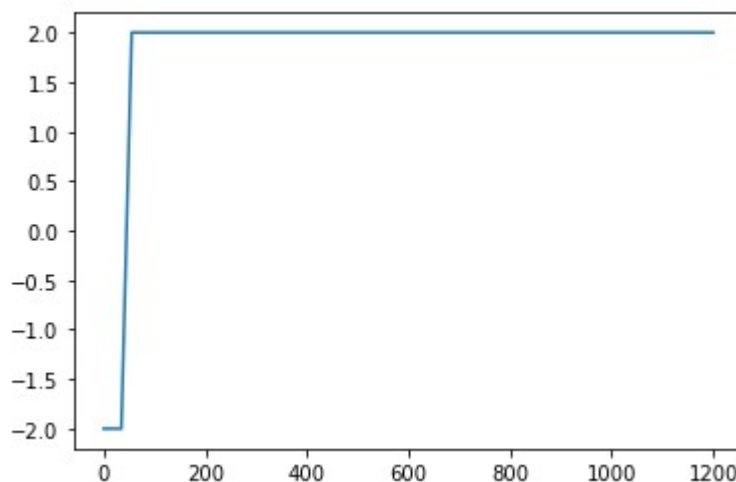
Question 7. Compléter la fonction `incidence(t)` ci-dessous.

```
Entrée [258]: 1 def incidence(t):
2     i0 = 2.
3     ta = 10.0
4     if t < (tdecollage - ta):
5         return -i0
6     elif t > (tdecollage + ta):
7         return i0
8     else:
9         return i0*(t - tdecollage)/ta
```

Question 8. Tracer i en fonction du temps.

```
Entrée [259]: 1 liste_incidence = [incidence(liste_T[i]) for i in range(0, len(liste_T))
2
3 plt.plot(liste_T, liste_incidence)
```

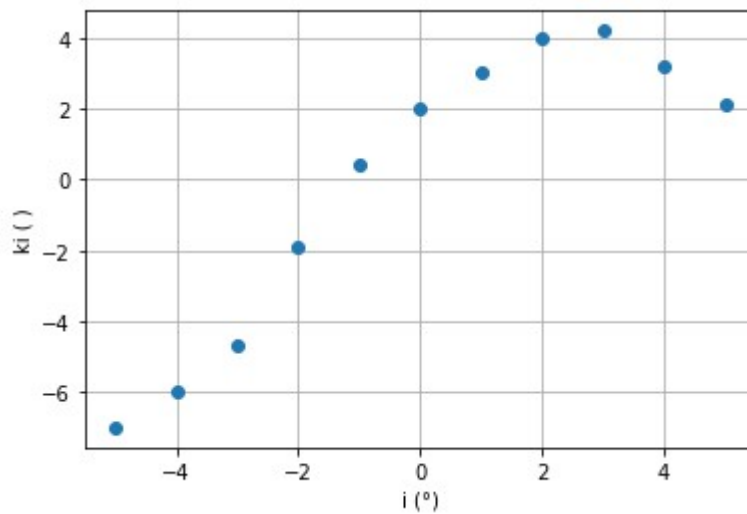
Out[259]: [<matplotlib.lines.Line2D at 0x14a9183aa30>]



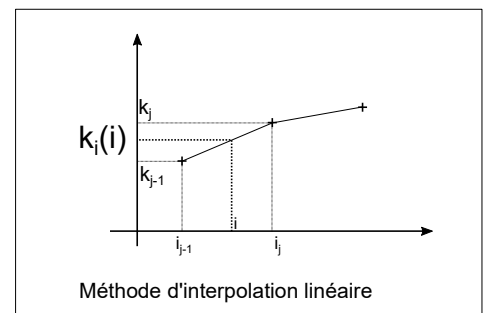
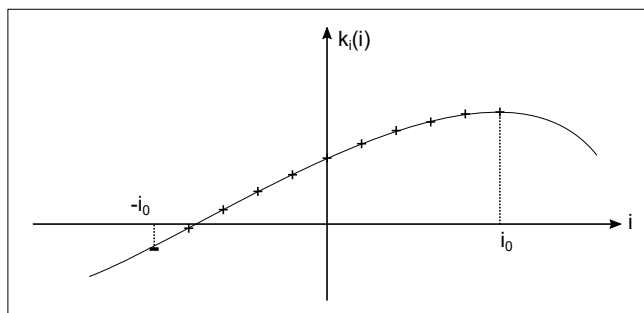
L'évolution de k_i en fonction de l'angle i a été obtenue en soufflerie. On donne les valeurs tabulées suivantes :

Entrée [260]:

```
1 i_doc = [-5,-4,-3,-2,-1,0,1,2,3,4,5]
2 ki_doc = [-7.,-6.,-4.7,-1.9,0.4,2.0,3.0,4.0,4.2,3.2,2.1]
3 plt.plot(i_doc,ki_doc,'o')
4 plt.xlabel('i (°)')
5 plt.ylabel('ki ( )')
6 plt.grid(True)
```



On cherche dans un premier temps à écrire une fonction permettant de calculer la force de portance quelque soit l'angle d'inclinaison des gouvernes. On va donc interpoler les valeurs tabulées en faisant l'hypothèse que k_i évolue de manière linéaire entre deux valeurs (voir schéma ci-dessous).



La documentation de la fonction `interp` de la bibliothèque `numpy` est donnée ici : <https://numpy.org/devdocs/reference/generated/numpy.interp.html#numpy.interp> (<https://numpy.org/devdocs/reference/generated/numpy.interp.html#numpy.interp>).

Question 9. Compléter la fonction ci-dessous afin qu'elle renvoie, pour une valeur de i quelconque, la valeur de k_i interpolée à partir des données des listes `i_doc` et `ki_doc`.

Entrée [261]:

```
1 def ki(i,i_doc,ki_doc):
2     return np.interp(i, i_doc, ki_doc)
3
4
5 print(ki(0.63,i_doc,ki_doc))
```

2.63

Question 9.bis. Compléter la fonction $F_p(i, v_x)$ qui renvoie la force de portance en fonction de l'angle d'inclinaison des gouvernes et de la vitesse horizontale.

Entrée [262]:

```
1 def Fp(i,vx):
2     K = 290
3     return K*ki(i,i_doc,ki_doc)*vx**2
4
5
6 print(Fp(0.63,87))
```

5772876.3

Question 10. De la même manière, créer une fonction $F_{vx}(t, \text{liste_T}, \text{liste_vx})$ qui renvoie la valeur de la vitesse horizontale pour n'importe quel temps t à partir des variables `liste_T` et `liste_vx` obtenues précédemment.

Entrée [263]:

```
1 def F_vx(t,liste_T,liste_vx):
2     return np.interp(t,liste_T,liste_vx)
3
4
5 print(F_vx(40,liste_T,liste_vx))
```

87.25667319087349

Question 11. Déterminer la fonction F_{Vert} de telle sorte que $\dot{v}_y = F_{\text{Vert}}(v_y, t)$. Compléter ensuite la fonction `FVert` ci-dessous.

Entrée [274]:

```
1 g = 9.81 #N/kg
2 fy = 4.e6 #N/(m/s)2
3 def Fvert(vy,t):
4     portance = Fp(incidence(t),F_vx(t,liste_T,liste_vx))
5     if portance < M*g:
6         return 0.
7     else:
8         return (portance - M*g)/M - (fy/M)*vy
9
10
11 #t = 60
12 #incidence(t)
13 #print(Fvert(0,t))
```

Question 12. Compléter les instructions ci-dessous afin de calculer la vitesse verticale de l'avion.

Entrée [275]:

```
1 vy0 = 0
2
3 liste_vy = integr.odeint(Fvert,vy0,liste_T)[: ,0]
4
5 print(liste_T)
6 print(liste_vy)
```

```
[0.00000e+00 1.20000e-01 2.40000e-01 ... 1.19976e+03 1.19988e+03
1.20000e+03]
[ 0.          0.          0.          ... 16.30395136 16.30415468
16.3043578 ]
```

Question 13. En déduire, par intégration, la position verticale de l'avion.

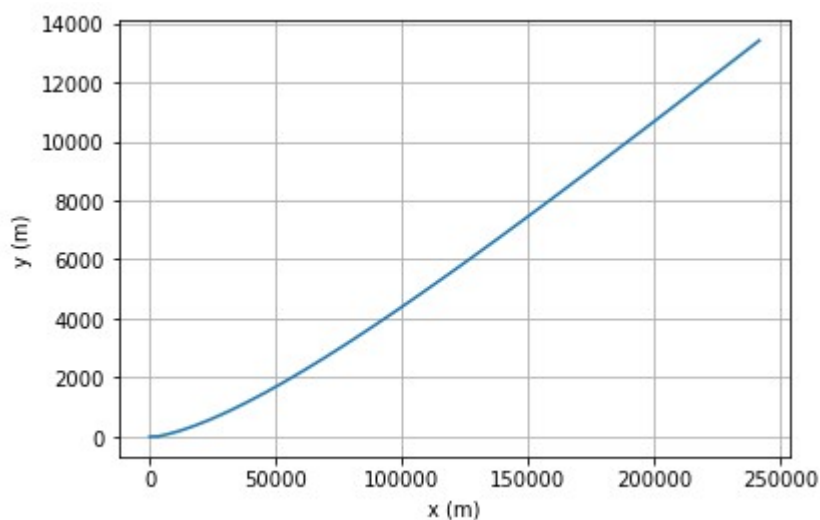
Entrée [276]:

```
1 liste_y = [0]
2 for i in range(1,n+1):
3     liste_y.append(liste_y[-1] + dt*(liste_vy[i] + liste_vy[i-1])/2)
4
5 #print(liste_x)
```

Question 14. Tracer la trajectoire de l'avion.

Entrée [277]:

```
1 plt.plot(liste_x,liste_y,)
2 plt.xlabel('x (m)')
3 plt.ylabel('y (m)')
4 plt.grid(True)
```



Question 15. En modifiant le temps de simulation, évaluer le temps que prend l'avion pour atteindre son altitude de croisière.

Entrée [279]:

```
1 k = 0
2 h_croisiere = 10700 ## en m/s
3 while liste_y[k] < h_croisiere:
4     k = k + 1
5 tcroisiere = liste_T[k]
6 print('temps croisière =',liste_T[k]/60,'minutes')
```

temps croisière = 17.2 minutes

Entrée []:

1

Entrée []:

1