TIPE 2021/2022

Thème :
SANTÉ / PRÉVENTION

Sujet :
CATAPULTAGE D'UN AÉRONEF POUR L'APPORT DE FOURNITURES MÉDICALES

Problématique :
COMMENT OPTIMISER LE DÉCOLLAGE D'UN AÉRONEF ?

Présenté par :
ETIENNE MARY

Table des matières

1. Contexte

Zipline est une entreprise d'avion autonome capable de livrer des consommables dans un rayon de 80 km.

Elle a été fondée avec la mission de permettre à chaque être humain d'avoir un accès instantané à des fournitures médicales vitales. Plus de 2 milliards de personnes dans le monde n'ont pas accès à ces fournitures par cause de manque d'infrastructures.
Leur première mission a commencé au Rwanda, à cause de la qualité de leurs routes ne permettant pas d'accès rapide et facile dans certains villages.

Leur système de catapultage permet à l'aéronef d'économiser 5% de sa batterie. La catapulte permet aussi de se dispenser d'une piste de décollage et d'atterrissage et donc d'économiser de la place.

Le sujet de ce TIPE sera donc de reproduire une catapulte permettant de faire décoller un planeur.
Nous étudierons l'influence de l'angle sur nos trajectoires et chercherons l'inclinaison adéquate de lancement pour permettre à notre planeur de parcourir la plus grande distance.

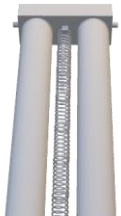
Drone Zipline larguant des poches de sangs

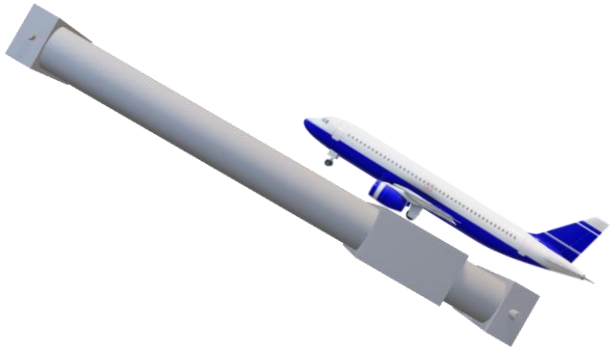
Préparation au catapultage d'un drone

2. Maquette & conception

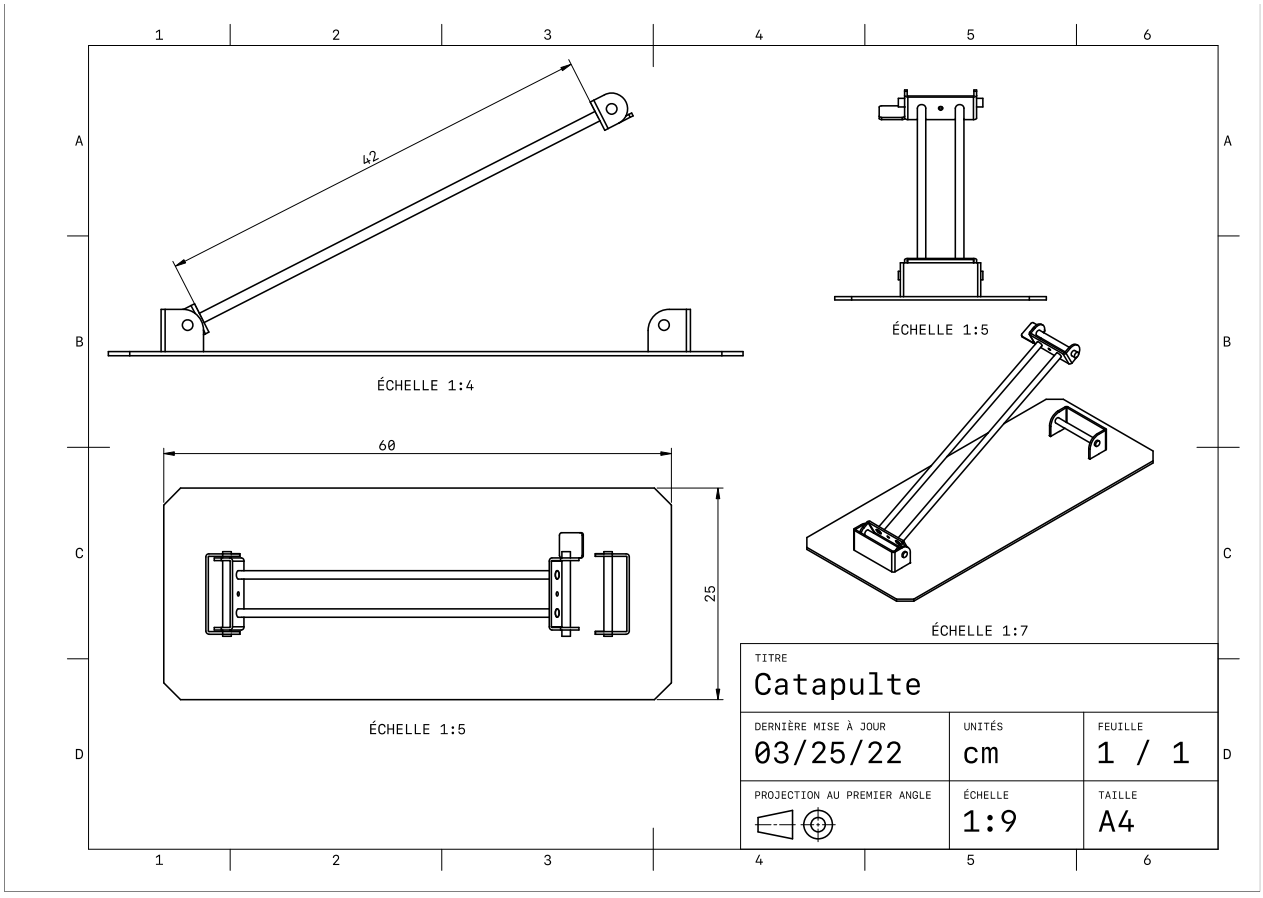
- **Étape 1 : Conception du modèle 3D**

La première étape était de mettre au clair mes idées du prototype.
La solution technique du système de catapultage était de mettre un coulisseau sur deux cylindres colinéaires tiré par un ressort.
La modélisation ci-dessous permet de donner une idée de cette solution.





Une fois le principe de fonctionnement validé, la conception du modèle 3D final pouvait être faite :



- **Étape 2 : Fabrication du prototype**

Pour procéder à la fabrication, un ami à moi ingénieur métallurgiste m'a aidé.
Nous avons envoyé le modèle 3D à une usine de découpe laser. Les pièces en INOX ont été reçus 2 semaines plus tard.

Pour les assembler, nous avons utilisé le principe de la soudure TIG (Tungsten Inert Gas).
Cette technique consiste à souder deux pièces ensembles à l'aide d'un arc électrique les chauffants à haute température (créant un bain de fusion) pour les souder ensemble. Le neutre (pince crocodile noire figure 2) est relié à la plaque principale pour créer un circuit fermé permettant au courant électrique de passer par l'électrode en Tungstène. Un gaz inerte (de l'Argon) est envoyé en même temps pour « stabiliser » l'arc électrique. Pour ce matériau l'intensité nécessaire à le faire fondre était proche des 90A.

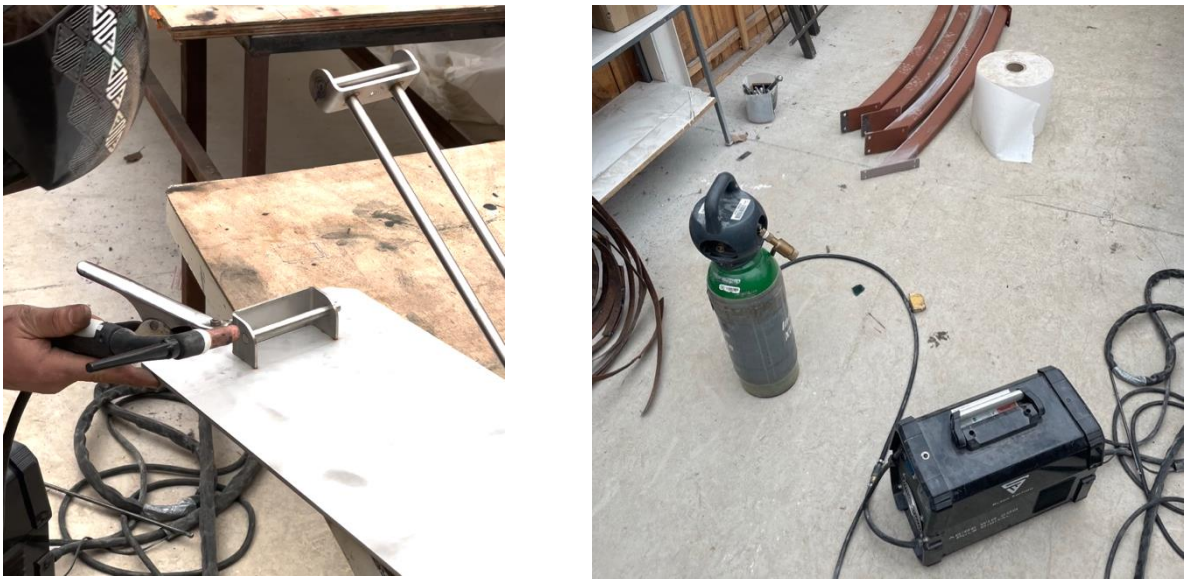
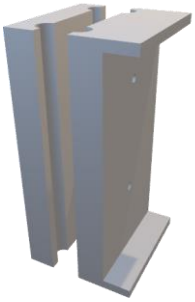


Figure 2 : Soudure TIG

- **Étape 3 : Conception du coulisseau et du système de catapultage**



La base de la catapulte terminée ; le modèle 3D du coulisseau permettant de transporter l'avion le long des cylindres pouvait être fait. Celui-ci allait être imprimé en 3D, de plus le coulisseau devait être démontable pour pouvoir l'installer sur les deux cylindres.

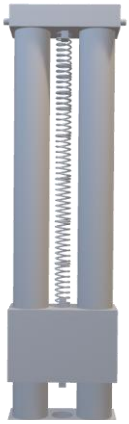


Figure 1 : Coulisseau

Pour le catapultage, la première idée était d'accrocher le ressort de bout en bout (figure 2).
Une fois les premiers tests effectués, les résultats n'étaient pas concluants dû au manque de place pour l'allongement du ressort.

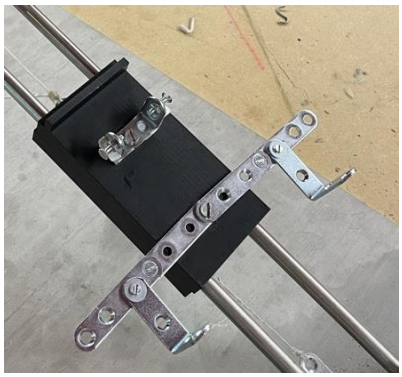
Figure 2

Figure 3

La seconde idée était donc de créer un système de poulie pour pouvoir doubler l'allongement du ressort (figure 3).
Celle-ci fut concluante lors des essais, cette solution a donc été adoptée pour le modèle final.

Pour assurer un bon décollage à notre planeur et l'adéquation de nos équations avec la réalité le prototype devait être le plus « parfait » possible. C'est-à-dire : limiter les frottements et transmettre d'une façon fluide et directe la vitesse du coulisseau au planeur. La difficulté est donc de maintenir l'avion droit tout en limitant les frottements lors du décollage et en lui transmettant le plus de vitesse possible.
Pour cela plusieurs systèmes ont été essayés :

- Le premier était d'utiliser des équerres pour tenir le dessous de l'avion horizontalement et de le pousser par les ailles arrières. Ce système créait trop de frottements et faussait nos équations.



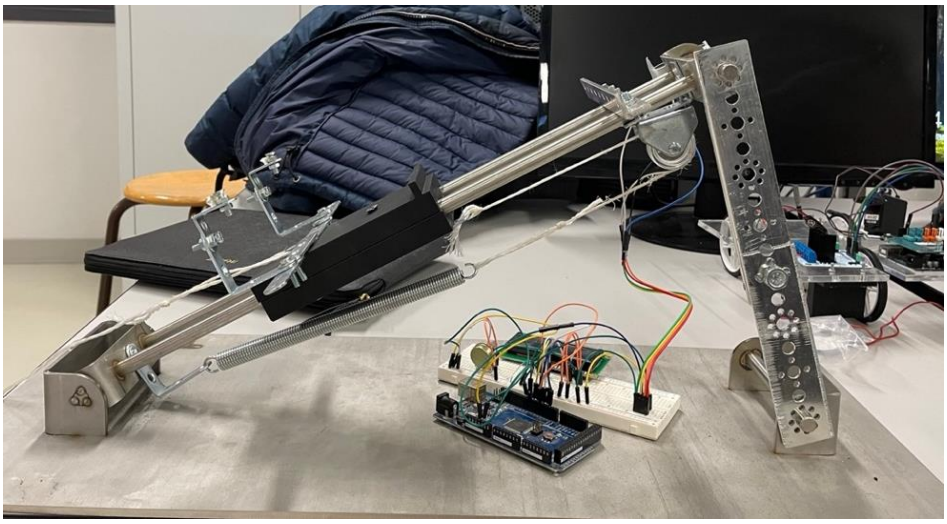
Solution de maintien n°1

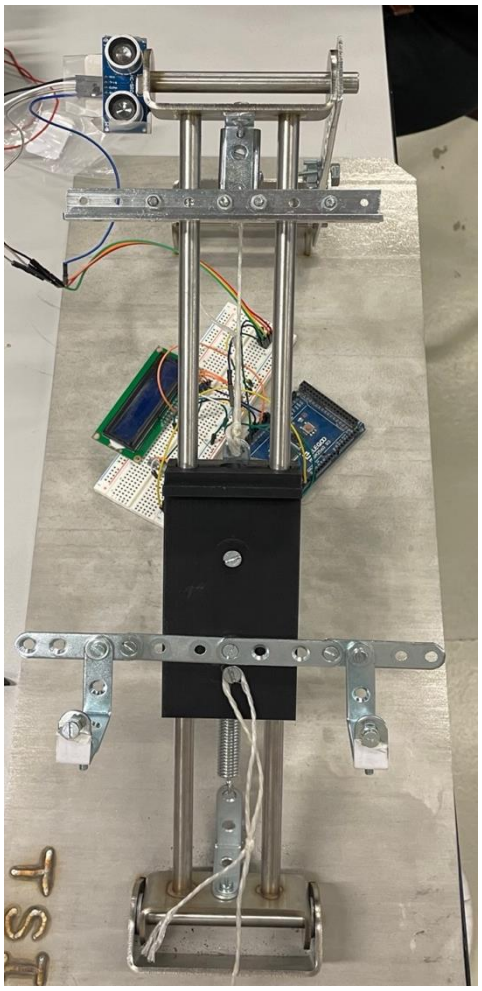
- Le second a donc été basé sur la même idée mais en enlevant les équerres de maintien. Celles-ci ont été remplacées par des équerres misent à l'envers sur lesquelles des vis et des écrous ont été installés pour permettre de régler le « serrage » des ailes arrière.

Cette solution, permettant de maintenir l'avion à l'horizontale tout en lui transmettant le plus de vitesse possible a été retenu.

Solution de maintien n°2

- **Étape 4 : Prototype final**





3. Détermination du coefficient de frottement visqueux

Problématique : Comment déterminer le coefficient de frottement visqueux de notre projectile ?

Nous avons besoin de connaître ce coefficient pour déterminer par la suite les équations de trajectoires.

a. Équation mécanique

Le but étant de déterminer le coefficient de frottement visqueux d'un objet quelconque de masse m, en le lâchant d'une hauteur h connu et en un temps t connu.
Avec un axe $\overrightarrow{e_x}$ ascendant :
PFD: $\vec{P} + \vec{f} = m\vec{a}$

$$-mg\overrightarrow{e_x} + \alpha v\overrightarrow{e_x} = m\frac{dv}{dt}\overrightarrow{e_x}$$

Sur $\overrightarrow{e_x}$:/ $-mg + \alpha v = m\frac{dv}{dt}$

$$\frac{dv}{dt} - \frac{\alpha}{m}v + g = 0\ (1)$$

$$SG_0 : v(t) = Ce^{\frac{\alpha}{m}t}$$

Par variation de la constante :
$v'(t) = C'(t)e^{\frac{\alpha}{m}t} + C(t) \times \frac{\alpha}{m} \times e^{\frac{\alpha}{m}t}$
$(1) \rightarrow C'(t)e^{\frac{\alpha}{m}t} + C(t) \times \frac{\alpha}{m} \times e^{\frac{\alpha}{m}t} - \frac{\alpha}{m}C(t)e^{\frac{\alpha}{m}t} + g = 0$
$\Leftrightarrow C'(t)e^{\frac{\alpha}{m}t} + g = 0$
$\Leftrightarrow C'(t) = -\frac{g}{e^{\frac{\alpha}{m}t}} = -ge^{-\frac{\alpha}{m}t}$
D'où : $C(t) = \frac{gm}{\alpha}e^{-\frac{\alpha}{m}t}$

D'où $v(t) = Ce^{\frac{\alpha}{m}t} + \frac{gm}{\alpha}$

$$v(0) = 0 \implies C = -\frac{gm}{\alpha}$$

D'où $v(t) = \frac{gm}{\alpha}(-e^{\frac{\alpha}{m}t} + 1)$
Or $v(t) = \frac{dx}{dt}$
En séparant les variables :

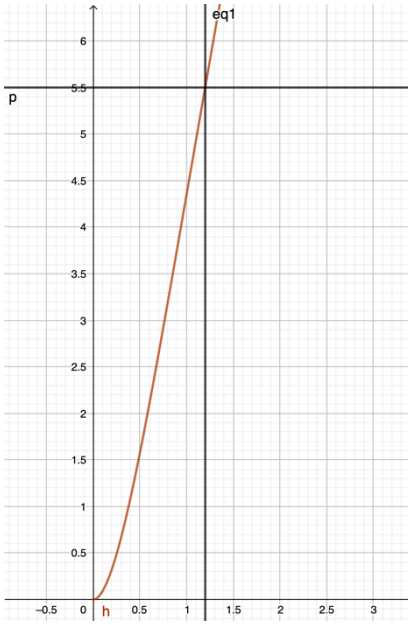
$$\int_0^{x(t)} dx = \frac{gm}{\alpha}\left(-e^{\frac{\alpha}{m}t} + 1\right).\int_t^0 dt$$

$$x(t) = -t\frac{gm}{\alpha}\left(-e^{\frac{\alpha}{m}t} + 1\right)$$

Notre seule inconnue dans cette expérience est alpha. Impossible de l'isoler … nous le chercherons par dichotomie.
Pour $\begin{cases} m = 8{,}2g \\ x = 5{,}5m \rightarrow \\ t = 1{,}2s \end{cases} \alpha = -0.0158$
En reportant sur Géogebra :

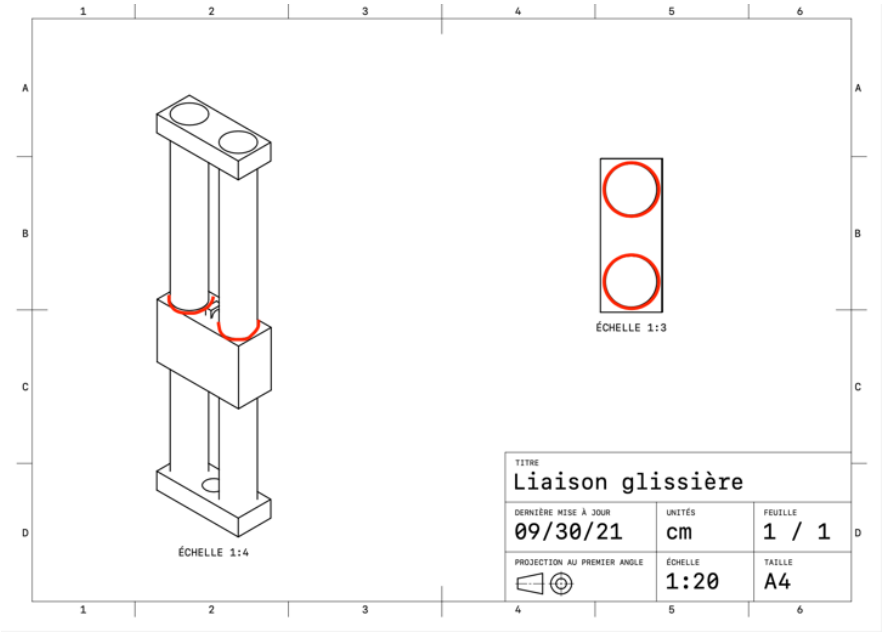
$$x(t) = -t\frac{9{,}81 \times 8{,}2.10^{-3}}{-0{,}0158}\left(-e^{\frac{-0{,}0158}{8{,}2.10^{-3}}t} + 1\right)$$

Résultat cohérent pour le coefficient de frottement visqueux issu de la dichotomie. (Voir annexe)



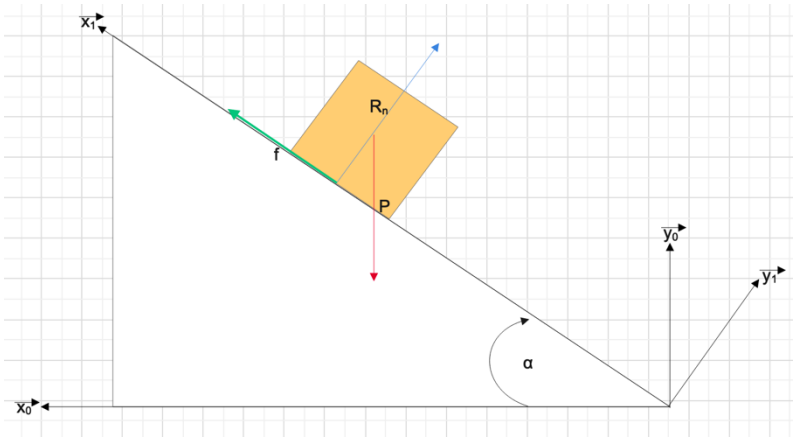
4. Détermination du coefficient de frottement sec

Problématique : Comment déterminer le coefficient de frottement sec de notre liaison glissière ?
Déterminer le coefficient de frottement sec expérimentalement grâce à une équation mécanique.
En rouge sur le schéma, les surfaces de frottements secs.



a. Équation mécanique

La but étant de trouver l'angle α limite pour lequel notre objet (coulisseau représenté par le bloc jaune) commence à bouger.



$$\vec{P} = -mg\overrightarrow{y_0}$$
$$\overrightarrow{R_n} = R_n\overrightarrow{y_1}$$

$\vec{f} = \mu R_n\overrightarrow{x_1}$, μ étant notre coefficient de frottement sec.

PFS :

$$-mg\overrightarrow{y_0} + R_n\overrightarrow{y_1} + \mu R_n\overrightarrow{x_1} = 0$$
$$\overrightarrow{y_0} = \sin\alpha\,\overrightarrow{x_1} + \cos\alpha\,\overrightarrow{y_1}$$

D'où :

$$-mg(\sin\alpha\,\overrightarrow{x_1} + \cos\alpha\,\overrightarrow{y_1}) + R_n\overrightarrow{y_1} + \mu R_n\overrightarrow{x_1} = 0$$

$\overrightarrow{x_1}/: -mgsin\,\alpha + \mu R_n = 0$ (1)
$\overrightarrow{y_1}/: -mgcos\,\alpha + R_n = 0 \iff R_n = mgcos\,\alpha$

En réinjectant R_n dans (1) : $-mgsin\,\alpha + \mu mgcos\,\alpha = 0$

$$\iff \mu = \tan(\alpha)$$

Conclusion : $\mu = \tan(\alpha)$

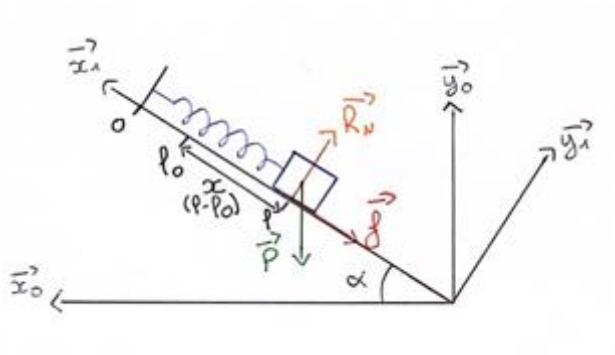
5. Détermination de la vitesse de sortie

Problématique : Comment déterminer la vitesse de notre projectile en sortie de catapulte ?

a. Équation mécanique

i. Théorème de l'énergie cinétique

Bilan des forces : $\begin{cases} \vec{P} = -mg\overrightarrow{y_0} \\ \overrightarrow{F_r} = k(l-l_0)\overrightarrow{x_1} = kx\overrightarrow{x_1} \\ \overrightarrow{R_n} = R_n\overrightarrow{y_1} \\ \vec{f} = -\mu R_n\overrightarrow{x_1} \end{cases}$



On a posé : $x = l - l_0$

$\overrightarrow{y_0} = \sin\alpha\overrightarrow{x_1} + \cos\alpha\,\overrightarrow{y_1}$

D'où $\vec{P} = -mg(\sin\alpha\overrightarrow{x_1} + \cos\alpha\,\overrightarrow{y_1})$

On applique le théorème de l'énergie cinétique en commençant par le calcul du travail des forces :

$$W(\vec{P}) = \int \vec{P}dx\overrightarrow{x_1} = -mxgsin\ \alpha$$

$$W(\overrightarrow{F_r}) = \int \overrightarrow{F_r}dx\overrightarrow{x_1} = k\left(\frac{x^2}{2}\right)$$

$$W(\vec{f}) = \int \vec{f}dx\overrightarrow{x_1} = -\mu R_n x = -\mu xmgcos\ \alpha$$

$W(\overrightarrow{R_n}) = 0$ car la réaction normale est toujours orthogonale au déplacement.

TEC :

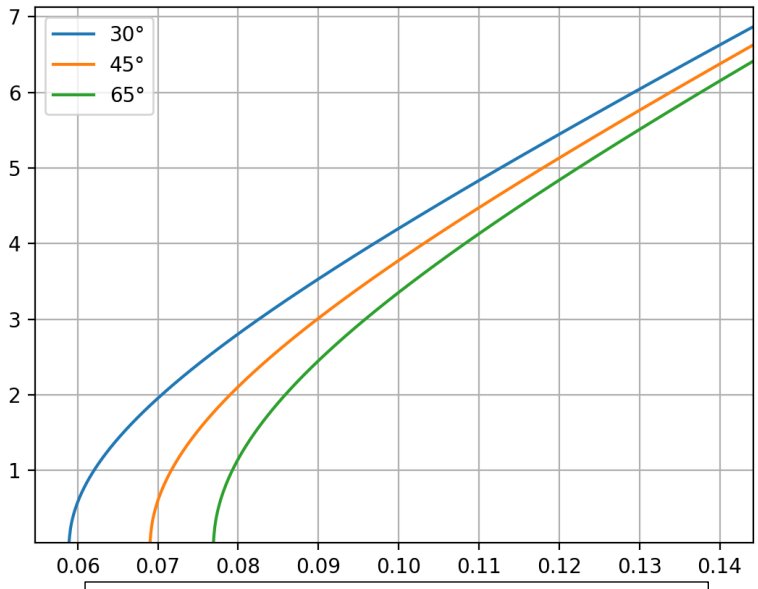
$$\Delta Ec_{l\to l_0} = \frac{1}{2}m \times v_{l_0}^2 - 0$$

$$= -mgxsin\ \alpha - \mu mgcos(\alpha)x + k\left(\frac{x^2}{2}\right)$$

$$v_{l_0}^2 = -2gxsin\ \alpha - 2\mu gcos(\alpha)x + \frac{2k}{m}\left(\frac{x^2}{2}\right)$$

$$\textbf{Donc : } \boldsymbol{v_{l_0} = \sqrt{-2gxsin\ \alpha - 2\mu gcos(\alpha)x + \frac{2k}{m}\left(\frac{x^2}{2}\right)}}$$

Résultats des simulations :



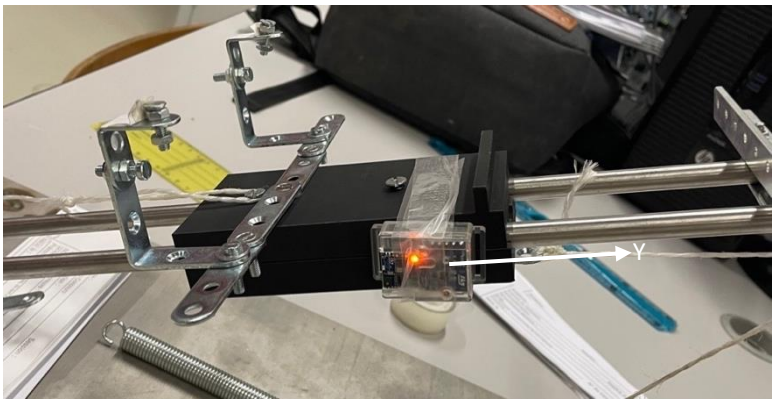
Vitesse (m/s) en fonction de l'allongement du ressort

b. Protocole expérimentale pour la mesure de vitesse

Problématique : Comment mesurer la vitesse de sortie de notre catapulte ?

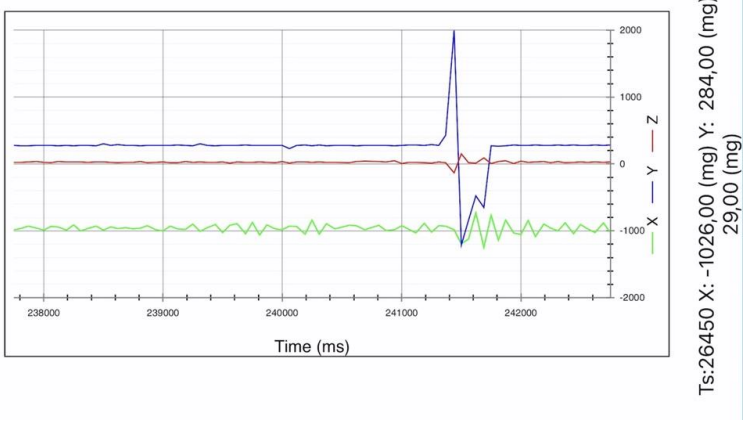
i. Accéléromètre

Le principe est de raccorder un accéléromètre sur notre coulisseau pour suivre l'évolution de notre accélération durant la phase de décollage. (Figure de droite)

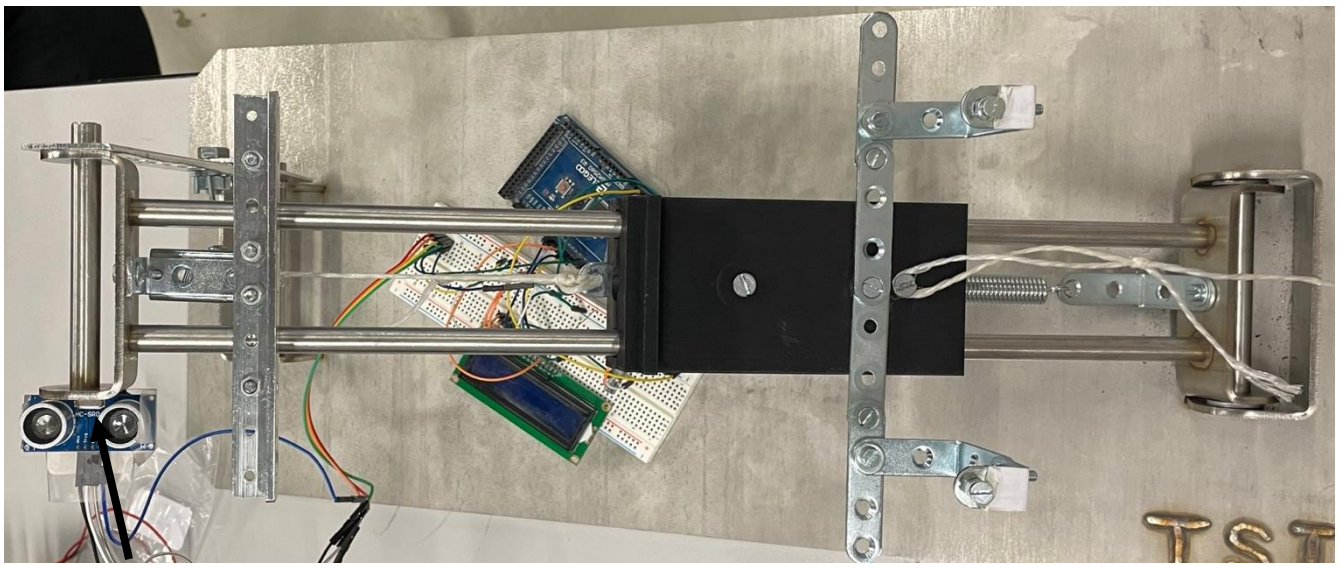


Le graphique nous donne l'accélération en mg (seulement suivant Y - bleu)
Et le temps en ms.
Le pique montre bien que le capteur sature à l'instant ou la catapulte part.

Le capteur n'est donc pas assez performant pour notre utilisation.
Par manque de moyen on ne retiendra pas cette solution.



ii. Capteur à ultrason



L'accéléromètre n'étant pas concluant, nous mesurerons la vitesse à l'aide d'un capteur à ultra-son.
Le but premier de ce capteur est de nous donner la distance face à lui en utilisant un système d'envoie et de réception d'ondes sonores.

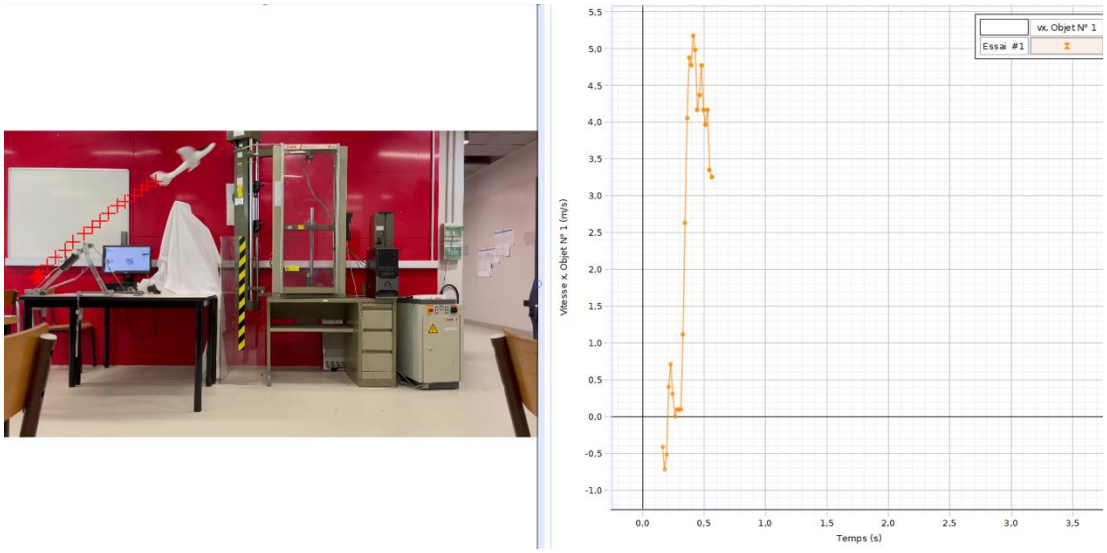
L'idée pour connaitre la vitesse est de mesurer le temps pendant lequel notre capteur détecte une différence d'hauteur (dû à l'aile), comme nous connaissons sa largeur et le temps de passage nous pouvons en déduire la vitesse.

Algorithme :

```
Boucle :
        -Mesurer la distance
                -Si distance < 20cm :
                        -Variable avion_present = 1
                        -lancer un chrono

                -Si distance>20 & avion_present=1 :
                        -lancer un deuxième minuteur
                        -Faire la différence
                        -Calculer notre notre vitesse à partir de la largeur de l'aile
                -Sinon :
                        -On recommence
```

c. Différence entre les résultats théoriques et expérimentaux



Résultat de la chronophotographie

Pour un angle de 45degré, la chronophotographie donne une vitesse de 5.2m/s tandis que le capteur affiche une vitesse de 4.3 m/s.
Nos calculs donnent une vitesse 5,13 m/s.

Le capteur n'est donc pas concluant. Nous garderons notre modèle physique.

6. Détermination des équations de trajectoires

a. Cas pour un objet quelconque

Problématique : détermination de la trajectoire d'un objet lancé avec une vitesse

Considérons notre avion comme un objet quelconque sans force de portance.

Bilan des forces : $\begin{cases} \vec{P} = -mg\vec{z} \\ \vec{f} = -\mu v\overrightarrow{x_1}, \mu\ coeff\ frot.visqeux \end{cases}$

$$\begin{cases} \vec{P} = -mg\vec{z} \\ \vec{f} = -\mu v(\sin\alpha\,\vec{z} + \cos\alpha\,\vec{x}) \end{cases}$$

Application du PFD :

$$\begin{cases} \vec{x}: ma_x + \mu v cos\ \alpha \\ \vec{z}: ma_z + \mu v sin\ \alpha + mg = 0 \end{cases}$$
$$\Leftrightarrow \begin{cases} \vec{x}: a_x + \dfrac{\mu}{m}\ v cos\ \alpha \\ \vec{z}: a_z + \dfrac{\mu}{m}\ v sin\ \alpha + g = 0 \end{cases}$$

Première équation :

$$\frac{dv_x}{dt} + \frac{\mu}{m}\cos\alpha\ v_x = 0$$

Deuxième équation :

$$\frac{dv_z}{dt} + \frac{\mu}{m}\sin\alpha\ v_z + g = 0$$

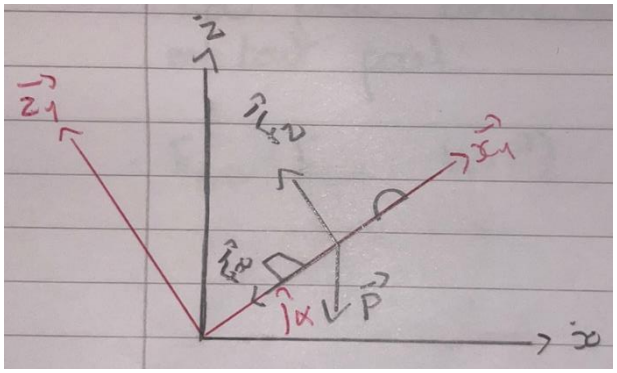
b. Cas pour l'avion

Problématique : détermination de la trajectoire du planeur

Cette fois ci, considérons notre avion comme un objet aérodynamique possédant une force de portance et de trainée.

Bilan des forces : $\begin{cases} \vec{P} = -mg\vec{z} \\ \overrightarrow{F_x} = -\frac{1}{2}\rho S v^2 C_x \overrightarrow{x_1} \\ \overrightarrow{F_z} = \frac{1}{2} S v^2 C_z \overrightarrow{z_1} \end{cases}$

$$\begin{cases} \vec{P} = -mg\vec{z} \\ \overrightarrow{F_x} = -\dfrac{1}{2}\rho S v^2 C_x(\sin\alpha\,\vec{z} + \cos\alpha\,\vec{x}) \\ \overrightarrow{F_z} = \dfrac{1}{2} S v^2 C_z(-\sin\alpha\,\vec{x} + \cos\alpha\,\vec{z}) \end{cases}$$



Application du PFD :

$$\begin{cases} \vec{x}: -\dfrac{1}{2}\rho S {v^2}_x C_x \cos\alpha - \dfrac{1}{2} S {v^2}_x C_z \sin\alpha = ma_x \\ \vec{z}: -\dfrac{1}{2}\rho S {v^2}_z C_x \sin\alpha + \dfrac{1}{2} S {v^2}_z C_z \cos\alpha - mg = ma_z \end{cases}$$
$$\Leftrightarrow \begin{cases} \vec{x}: \dfrac{1}{2}\rho S {v^2}_x C_x \cos\alpha + \dfrac{1}{2} S {v^2}_x C_z \sin\alpha + m\dfrac{dv_x}{dt} = 0 \\ \vec{z}: \dfrac{1}{2}\rho S {v^2}_z C_x \sin\alpha - \dfrac{1}{2} S {v^2}_z C_z \cos\alpha + mg + m\dfrac{dv_z}{dt} = 0 \end{cases}$$

Première équation :

$$\frac{1}{2m}\rho S(C_x\cos\alpha + C_z\sin\alpha)v_x^2 + \frac{dv_x}{dt} = 0$$

Seconde équation :

$$\frac{1}{2m}\rho S(C_x\sin\alpha - C_z\cos\alpha)\ {v^2}_z + g + \frac{dv_z}{dt} = 0$$

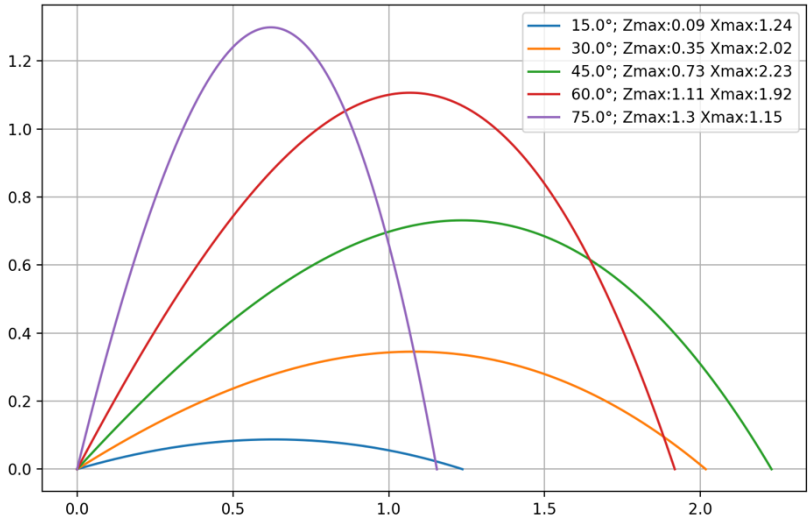
Nos deux équations dépendent de notre angle alpha qui varie en fonction du temps à chaque instant. Aucun théorème physique peut nous permettre de trouver une relation entre notre angle et la vitesse/position. Nous avons donc 3 inconnues pour 2 équations, plusieurs solutions sont envisageables :

- Résoudre les équations différentielles en prenant alpa constant (angle de décollage)
- Approximer notre angle en fonction de notre vitesse (suivant z). (Vz = vitesse sur z →Hauter ; Alphai : angle de lancement)
    - Si Vz = 0 →Alpha = 0
    - Si Vz= Vz/2 →Alpha = Alphai/2
    - Si Vz=-Vz/2 →Alpha =- Alphai/2
    - En dehors →Alpha = Alphai

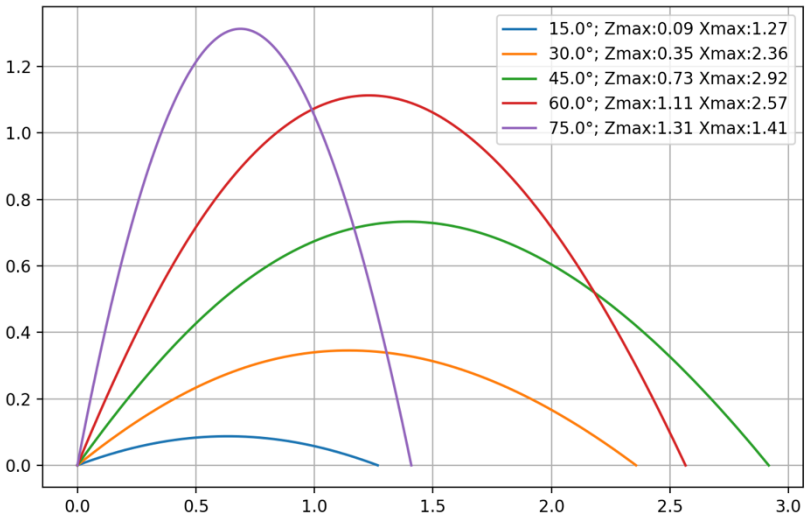
Ces équations n'étants pas linéaires avec des coefficients non constants nous les résoudrons à l'aide d'un programme informatique. (Voir annexes)

c. Résultats des programmes

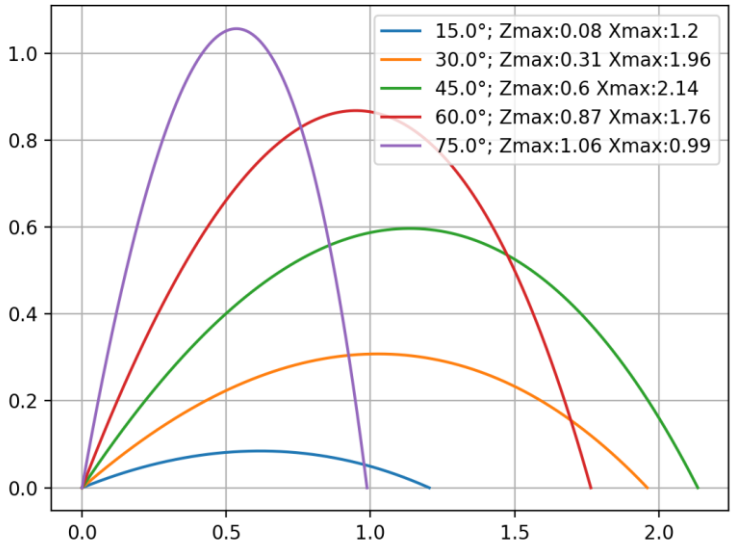
Résultats des 3 programmes pour une vitesse initiale de 5 m/s.



Résultat 1 : Alpha constant



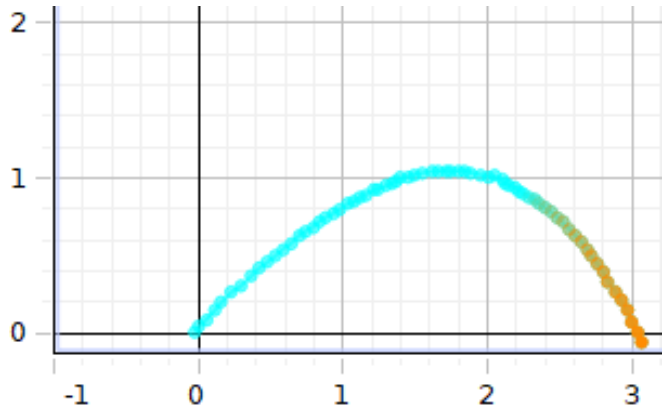
Résultat 2 : Alpha approximé

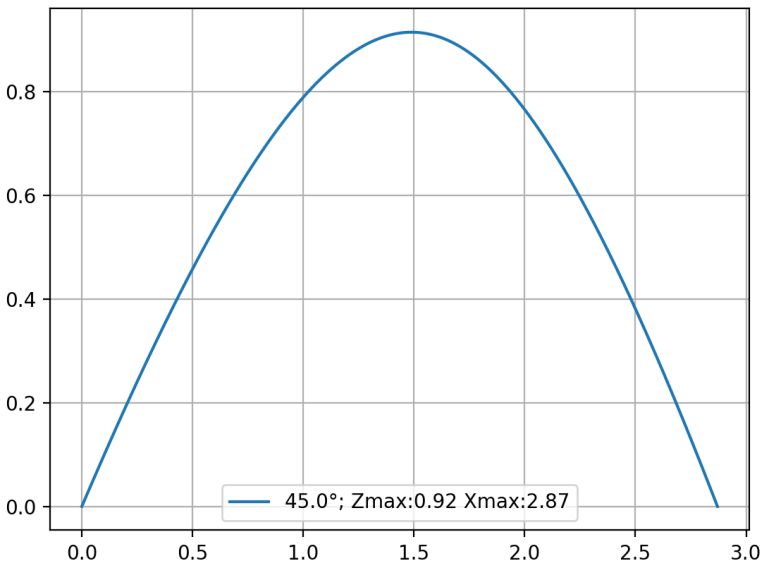


Résultat 3 : Objet non aérodynamique

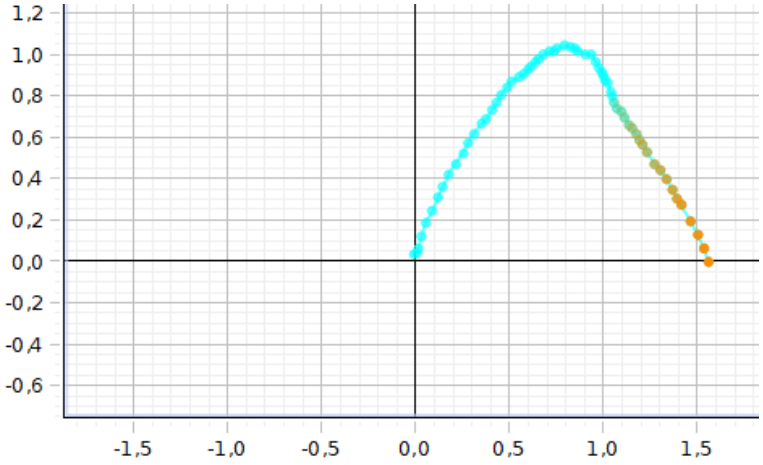
d. Mesures expérimentale (chronophotographie)

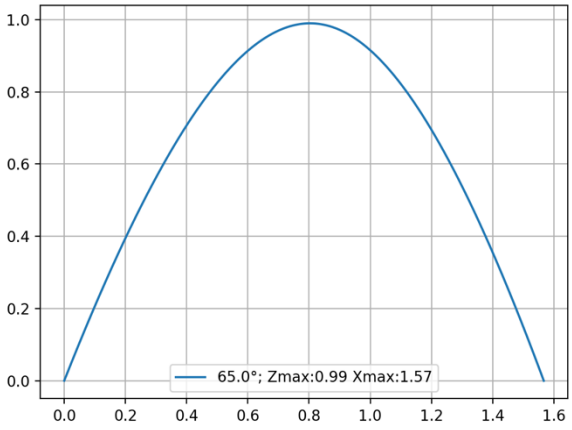
Pour un angle de 45 degrés et une vitesse initiale de 5,2 m/s. À gauche la chronophotographie et à droite notre résultat expérimentale.



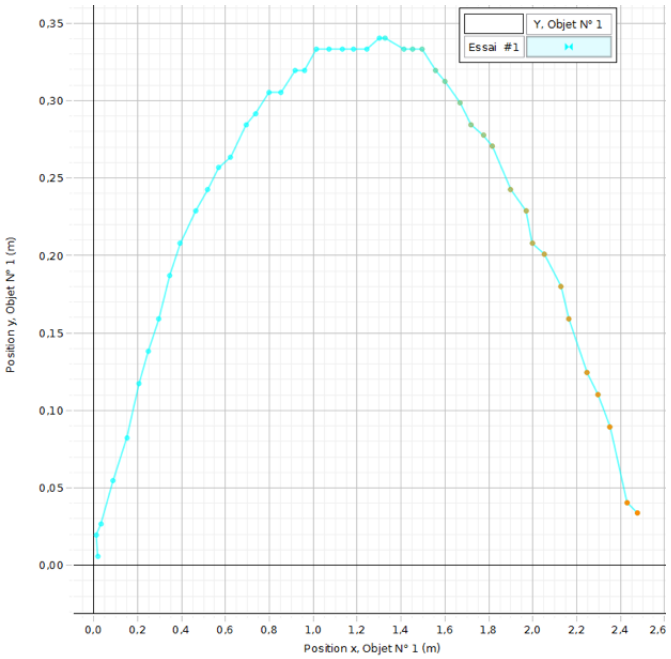


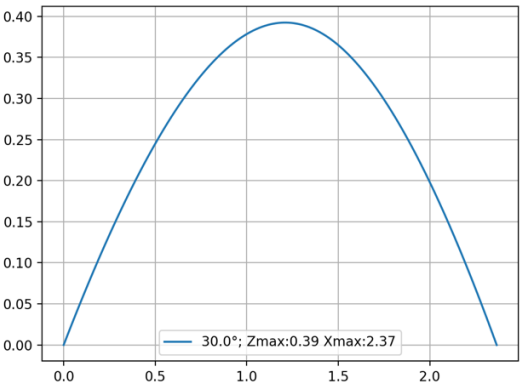
Angle de 65 degrés et vitesse initiale de 4.6 m/s.





Angle de 30 degrés et vitesse initiale de 5.4 m/s.





7. Annexes

a. Dichotomie pour la recherche du coefficient de frottement visqueux

```
1.  from math import *
2.  def deplacement(x):
3.      m = 8.2*10**-3
4.      g = 9.81
5.      t = 1.2
6.      fonction = -(m*g*t)/(x) * (-exp(x*t/m) + 1)
7.      valeur = 5.5
8.      return fonction,valeur
9.
10. def dico(a,b,delta):
11.     i = 1
12.     while i < delta:
13.         m = (a+b)/2
14.         print(m,deplacement(m)[0])
15.         if(deplacement(m)[0] < deplacement(m)[1]):
16.             a = m
17.             #entre m et b
18.         else:
19.             b = m
20.             #entre a et m
21.         i+=1
22.     print(m)
23.
24. dico(-1,0,100)
25.
```

Résultat : -0.0158

a. Programme pour mesurer la vitesse

```
#include <LiquidCrystal.h>
const int rs = 2, en = 3, d4 = 4, d5 = 5, d6 = 6, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
const byte TRIGGER_PIN = 8;                 // Broche TRIGGER
const byte ECHO_PIN = 9;                    // Broche ECHO
const unsigned long MEASURE_TIMEOUT = 25000UL; // 25ms = ~8m à 340m/s
const float SOUND_SPEED = 340.0 / 1000;       /* Vitesse du son dans l'air en mm/us */
int passage1 = 0;                           // pour aiguiller la loop
int passage2 = 0;                           // pour aiguiller la loop
unsigned long temps1 = 0;
unsigned long temps2 = 0;
unsigned long temps  = 0;
float longueur = 9;
float vitesse = 0;

void setup()
{
  lcd.begin(16, 2);
  analogWrite(8, 15);
  Serial.begin(115200);
  pinMode(TRIGGER_PIN, OUTPUT);
  digitalWrite(TRIGGER_PIN, LOW);
  pinMode(ECHO_PIN, INPUT);
  Serial.println("fin setup ");
}

void loop()
{
  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER_PIN, LOW);
  long measure = pulseIn(ECHO_PIN, HIGH, MEASURE_TIMEOUT);
  float distance_mm = measure / 2.0 * SOUND_SPEED;
  if (passage1 == 0)
  {
    if (distance_mm < 150)//l'objet passe
    {
      temps1   = millis();
      passage1 = 1;
      passage2 = 1;
    }
  }
  if (passage2 == 1)
  {
    digitalWrite(TRIGGER_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIGGER_PIN, LOW);
    long measure = pulseIn(ECHO_PIN, HIGH, MEASURE_TIMEOUT);
    float distance_mm = measure / 2.0 * SOUND_SPEED;
    if ( distance_mm > 200)
```

```
51.     {
52.       temps2 = millis();
53.       temps = temps2 - temps1;
54.       lcd.clear();
55.       lcd.print("Temps = ");
56.       vitesse = (longueur*10/temps);
57.       Serial.print("V =  ");
58.       Serial.println(vitesse);
59.       lcd.print(temps);
60.       lcd.setCursor(0,1);
61.       lcd.print("Vitesse = ");
62.       lcd.print(vitesse);
63.       Serial.println(temps);
64.       passage1 = 0;
65.       passage2 = 0;
66.     }
67.   }
68. }
69.
```

a. Programme pour calculer notre vitesse de sortie

```
#TEC
from math import *
from turtle import position
import matplotlib.pyplot as plt
import scipy.integrate as it
#PFD
m = 0.45 #masse
k = 1400 #raideur
angles = [pi/6, pi/4,65*pi/180] #angles lancement
g = 9.81
mu = 1

x=[]
v=[]
i = 0
while i <0.15: #création de notre liste des abs
    x.append(i)
    i+=0.0001

def f(x,mn):
    x = float(x)
    return(sqrt(-4*m*g*sin(angles[mn])-4*m*mu*g*cos(angles[mn])*x+2*k*x**2))

for z in range(len(angles)):
    for o in range(len(x)): #calcul de la vitesse pour chaque x
        try:
            v.append(f(x[o],z))
        except ValueError:
            v.append(0)
    plt.plot(x,v,label=str(round(angles[z]*180/pi))+'°')
    for o in range(len(x)):
        if x[o] > 0.1199 and x[o] < 0.1201:
            print("V="+str(v[o])+" m/s pour "+str(round(angles[z]*180/pi))+'°')
    v = []

plt.legend()
plt.grid(visible=True, which='major', axis='both')
plt.show()
```

a. Programme pour résoudre les équations différentielles du mouvement

```
1.  from math import *
```

```
2.  from turtle import position
3.  import matplotlib.pyplot as plt
4.  import scipy.integrate as it
5.
6.  maxtemps = 40 #plage de temps sur laquelle les calculs sont faits
7.  vitesseinitiale = 4.5 #vitesse de lancement en sortie de la catapulte
8.  variationangle = pi/12 #angle de lancement
9.  masse = 40*(10**-3) #masse de l'avion
10. g = 9.81 #constante de force de gravitation
11. cx = 0.005 #coeff frottement sur x - 0,005 à 0,010
12. cz = 0.6 #coeff portance ailes - 0.3 à 0.7
13. p = 1.225 #densité de l'air
14. S = 0.030 #surface des ailes
15. precision = 0.0001 #précision des calculs (plus c petit mieux c)
16. #calcul des coefficients en fonction des différents angles
17. constantex = []
18. constantez = []
19. angle=[]
20. for i in range(int((pi/2)/variationangle)): #
21.     constantex.append(round((1/(2*masse))*p*S*(cx*cos(i*variationangle)+cz*sin(i*variationangle)),90))
22.     constantez.append(round((1/(2*masse))*p*S*(cx*sin(i*variationangle)-cz*cos(i*variationangle)),90))
23.     angle.append(round(i*variationangle,90))
24. print(angle)
25. print(constantex)
26. print(constantez)
27. #recherche maximum d'une liste
28. def maximum(L):
29.     max = L[0]
30.     for i in range(len(L)):
31.         if L[i] > max:
32.             max = L[i]
33.     return max
34. #boucle pour calculer à chaque angle en fonction de notre variable variation angle
35. for k in range(len(angle)):
36.     if(angle[k] != 0):
37.         vitessex = vitesseinitiale*cos(angle[k])
38.         vitessez = vitesseinitiale*sin(angle[k])
39.         def F1(y,x,c): #y' = ...  equa diff sur x
40.             return -c*y**2
41.         def F2(y,x,c): #y' =  ... equa diff sur z
42.             return -c*y**2-9.81
43.
44.         def euleur(F, y0, dx, xmax,c):
45.             xs = [0]
46.             ys = [y0]
47.             while xs[-1] < xmax:
48.                 try:
49.                     x = xs[-1] + dx
50.                     y = ys[-1] + F(ys[-1], xs[-1],c)*dx
51.                     xs.append(x)
52.                     ys.append(y)
```

```
                except OverflowError: #en cas d'erreur pour certains angles
                    print("Angle impossible"+str(angle[k]))
                    xs.clear()
                    ys.clear()
                    xs=[0]
                    ys=[0]
                    break
            return xs,ys

        lx,ly = euleur(F1,vitessex,precision,maxtemps,constantex[k])
        lxx,lyy = euleur(F2,vitessez,precision,maxtemps,constantez[k])

        #plt.plot(lxx,lyy,label=str(round(angle[k]*180/pi,1)))
        if len(lyy) >= 5:
            #recherche position par intégration comme nos équas diff étaient sur la vitesse
            def position(varx,vary,dt,condition=0):
                temps=[0]
                location = [0]
                tot = maxtemps/dt
                y=0
                for i in range(int(tot)-1):
                    if y>= condition:
                        x = dt+i*dt
                        pos = x * len(varx)/maxtemps
                        posp = (x-dt)*len(varx)/maxtemps
                        y = y+ vary[int(pos)]*dt+(vary[int(posp)]-vary[int(pos)])*dt*1/2
                        location.append(y)
                        temps.append(x)
                return(temps,location)
            temps,location = position(lx,ly,precision)
            tempsz,locationz = position(lxx,lyy,precision)
            #tracer t en fonction de x (avec l'equa diff sur x):
            # sert pour reinjecter dans la position de z
            def positioninverse(x,y):
                y,x=x,y
                return(x,y)
            temps2,location2 = positioninverse(temps,location)
            #tracer z en fonction de x (et non de t) --> en position
            #on vérfie les longueurs des listes à cause de la condition z>0
            if len(tempsz) < len(temps2):
                temps2 = temps2[:len(tempsz)]
            #z en fonction de x
            newx = temps2
            newz=locationz
            maxx = round(newx[-1],2)
            maxz = round(maximum(newz),2)
            plt.plot(newx,newz,label=str(round(angle[k]*180/pi,1))+'°;'+' Zmax:'+str(maxz)+' Xmax:'+str(maxx))
    plt.legend()
    plt.grid(visible=True, which='major', axis='both')
    plt.show()
```