Un Jeu Olympique: le Kayak

El Karkouri Mohamed Yahya
18334

Sommaire

- Introduction
- Analyse théorique
- Approche expérimentale
- Résultats et synthèse

Introduction

Problématique

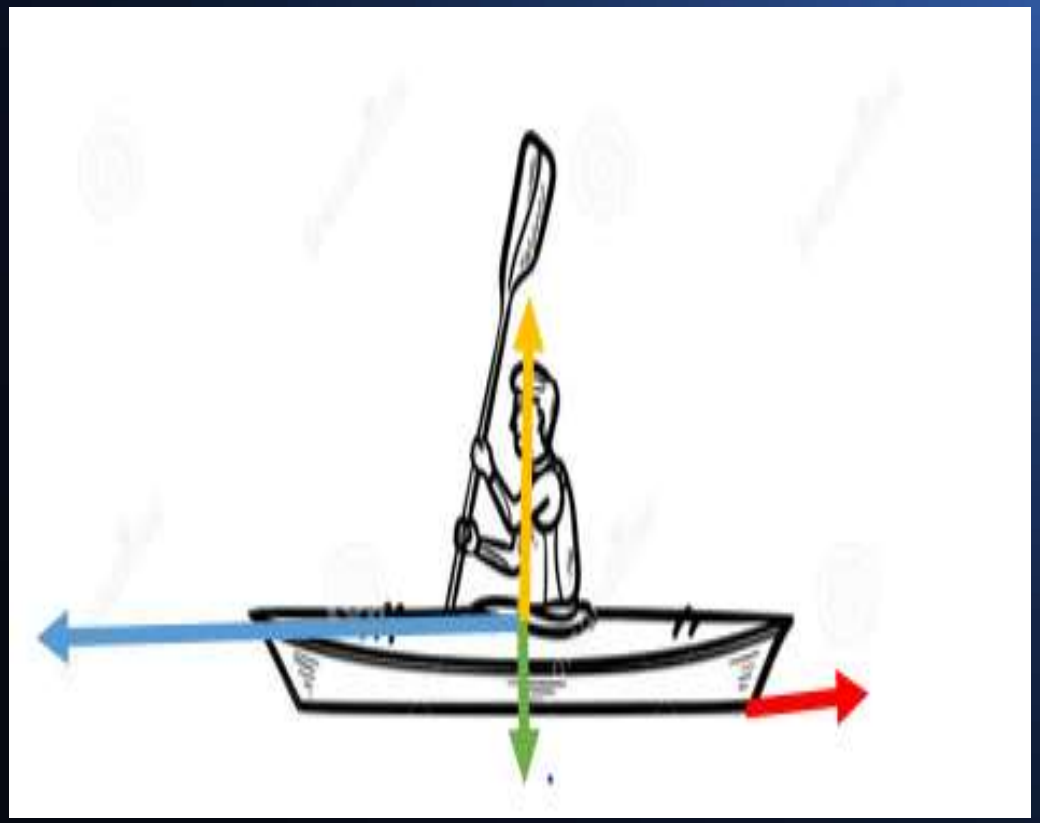
Comment améliorer les performances sportives des kayakistes lors des competitions de haut-niveau?

Etude mécanique

- Forces auxquelles est soumis le kayak
- Puissance et Efficacité

Bilan des forces

Système d'étude:{kayak+athlete+pagaie}



- Poids
- Poussée d'Archimède
- F.Trainée
  - C.hydrodynamique
  - C.aérodynamique
- F.de propulsion

Composante hydrodynamique

D'après l'hypothèse de Froude:

- Force de frottement pur: $$D_f = \frac{1}{2}\rho V_H^2 A_w C_f$$

- Action des vagues sur la coque: $$Dw = \frac{1}{2}\rho V_H^2 C_w \frac{\Delta^{5/3}}{L^{3\cdot}}$$

Composante hydrodynamique

- C_f : coefficient de frottement pure $Re = V_H \, {}^{L}/_{\nu}$
- C_w :coefficient des vagues $Fr = V_H / \sqrt{gL}$
- A_w :surface à l'eau $A_w = 2{,}5\sqrt{\Delta L}$

Composante aérodynamique

$$D_A = \frac{1}{2}\rho_A V_H^2 A_A$$

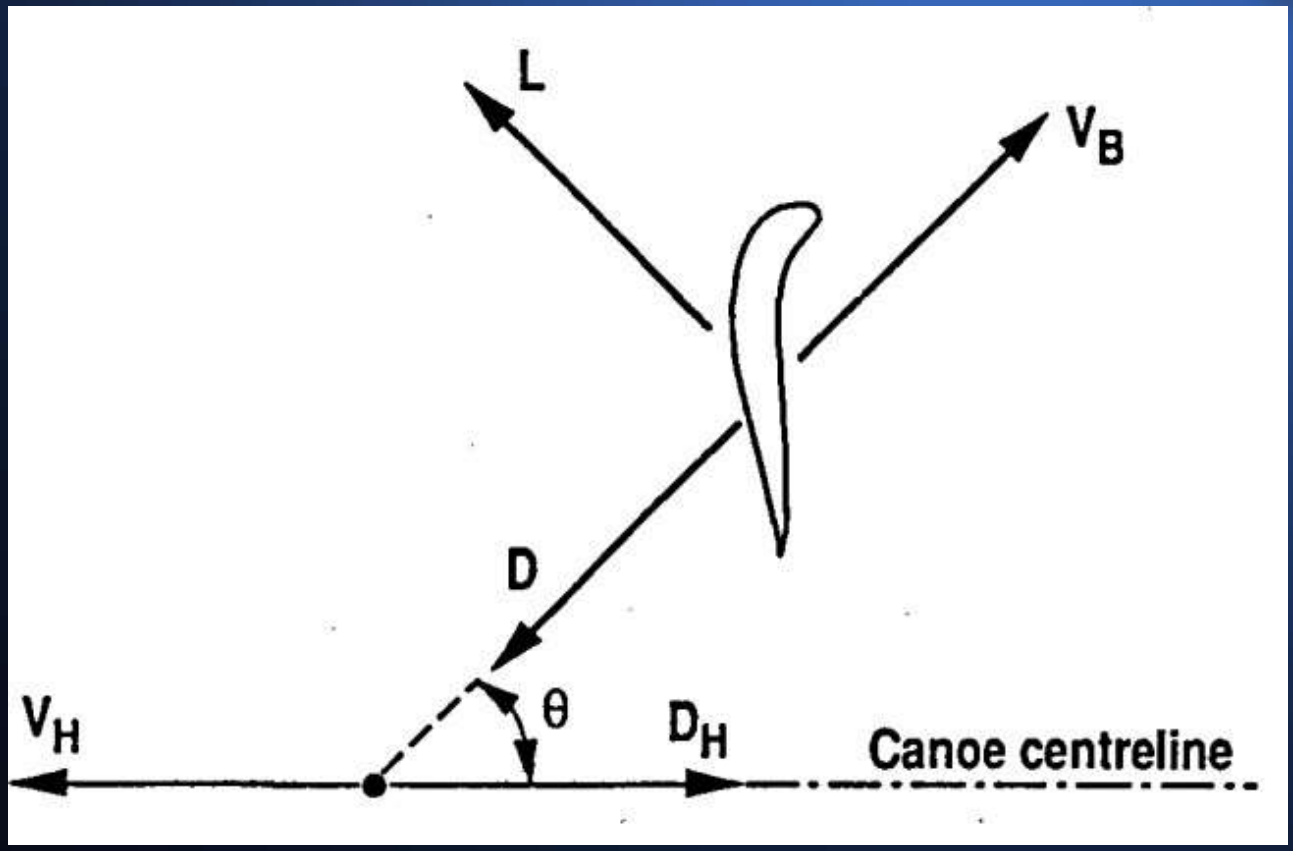
- A_A :aire frontale aérien

➢ Force de trainée totale:

$$D_H = \frac{1}{2}\rho V_H^2 A_H$$

- A_H: coefficient total

Efficacité



Efficacité

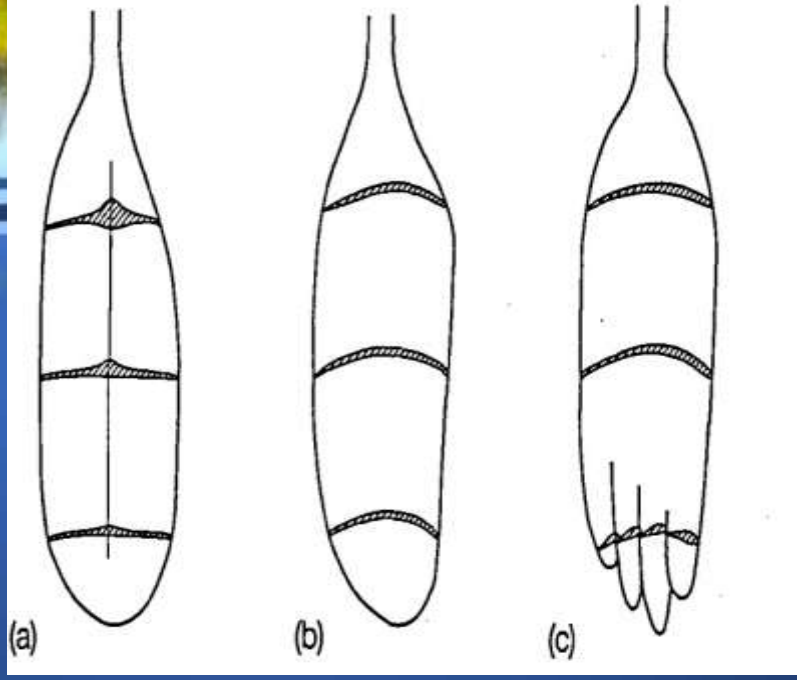
- Force de trainée totale opposée à la coque

$$D_H = \frac{1}{T}\int_0^T (D\cos\theta + L\sin\theta)\, dt$$

- Puissance à fournir par le pagayeur

$$P = D_H V_H + \frac{1}{T}\int_0^T D V_B\, dt$$

Efficacité



- Rendement

$$\eta = D_H \, V_H / P$$

- Influence de la forme de la paggaie

$$\frac{1}{\eta} = 1 + \frac{k}{cos\phi} \frac{A_H}{A_V} \frac{T \, V_H}{\sqrt{A_V}}$$

Puissance

- Puissance à fournir par le kayakiste

$$P = \frac{1}{2}\frac{\rho A_H {V_H}^3}{\eta}$$

$$\frac{dV_H}{V_H} = \frac{d\eta}{3\eta} - \frac{dA_H}{3A_H} + \frac{dP}{3P}$$

Approche expérimentale

But:

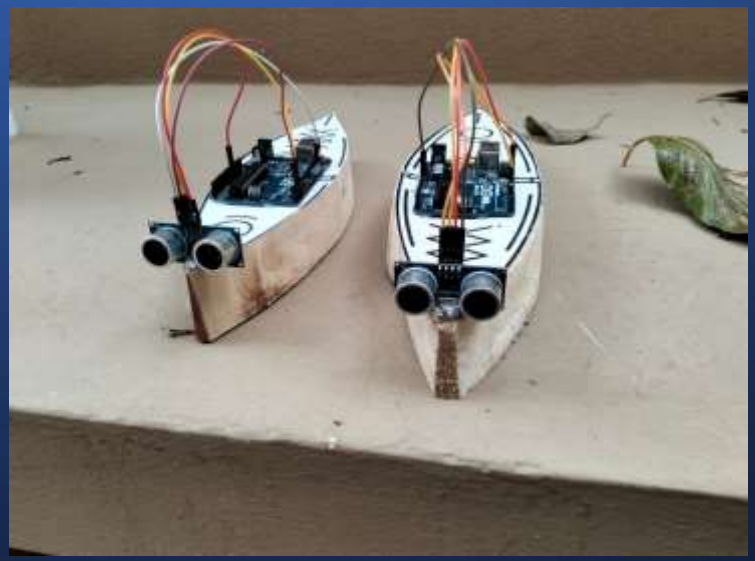
- Influence de la force de trainée sur l'évolution de la Vitesse
- Retrouver le coefficient de frottement expérimentalement

Principe de l'expérience

$$M_e \frac{dV}{dt} = F_m(t) - F_D(t)$$

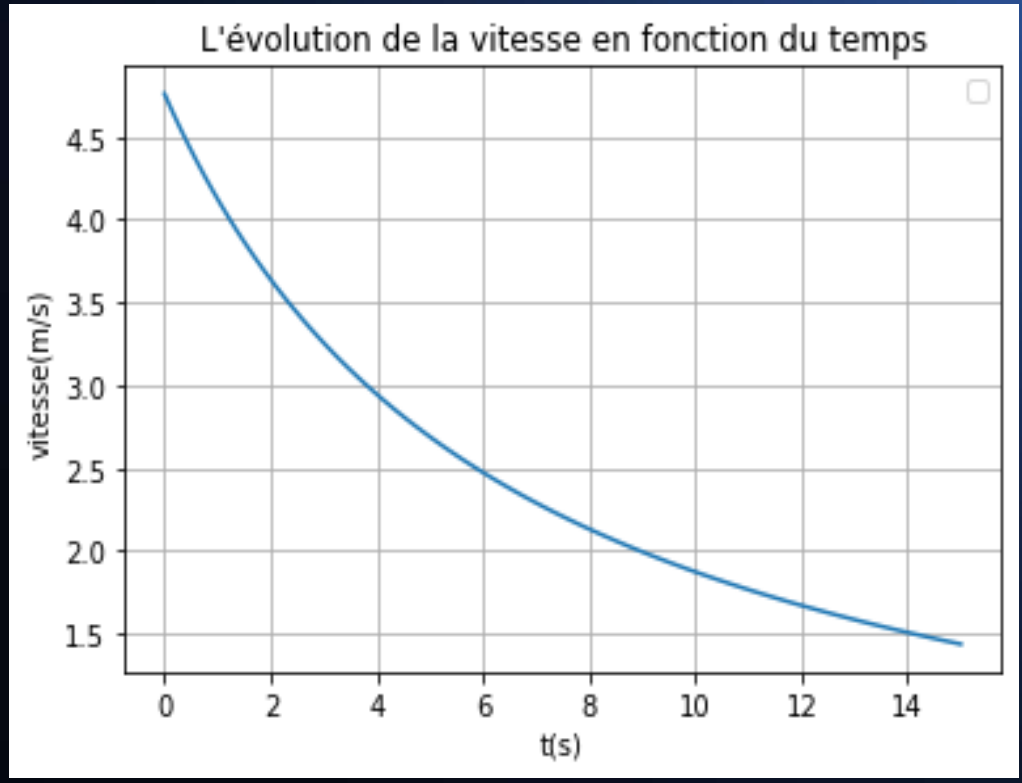
- on pose $Fm(t) = 0$

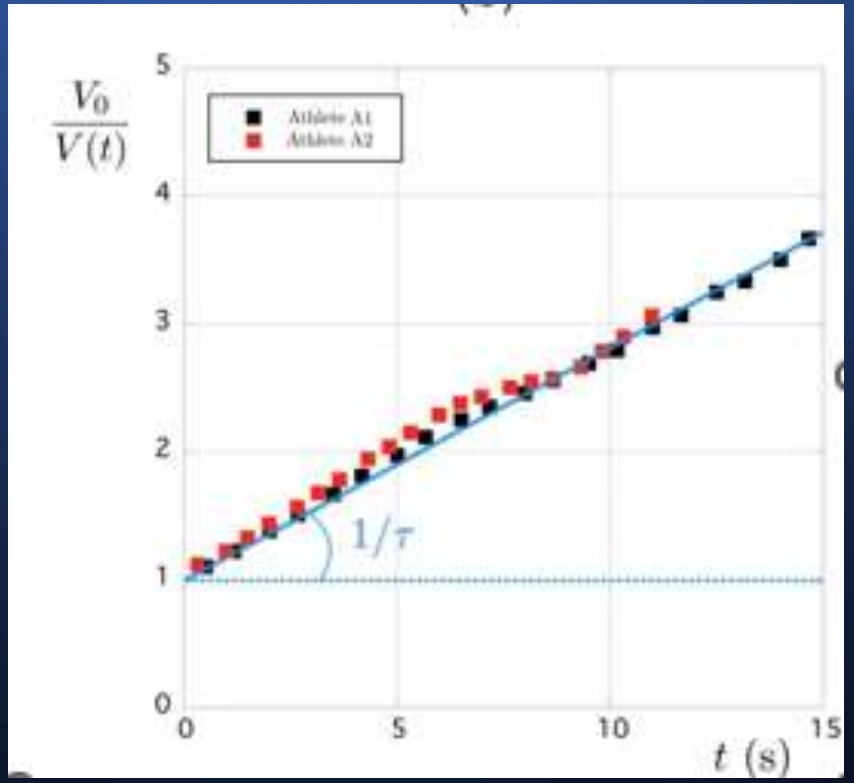
$$M_e \frac{dV}{dt} = -\frac{1}{2}\rho S C_D V^2$$



Attendus de l'expérience

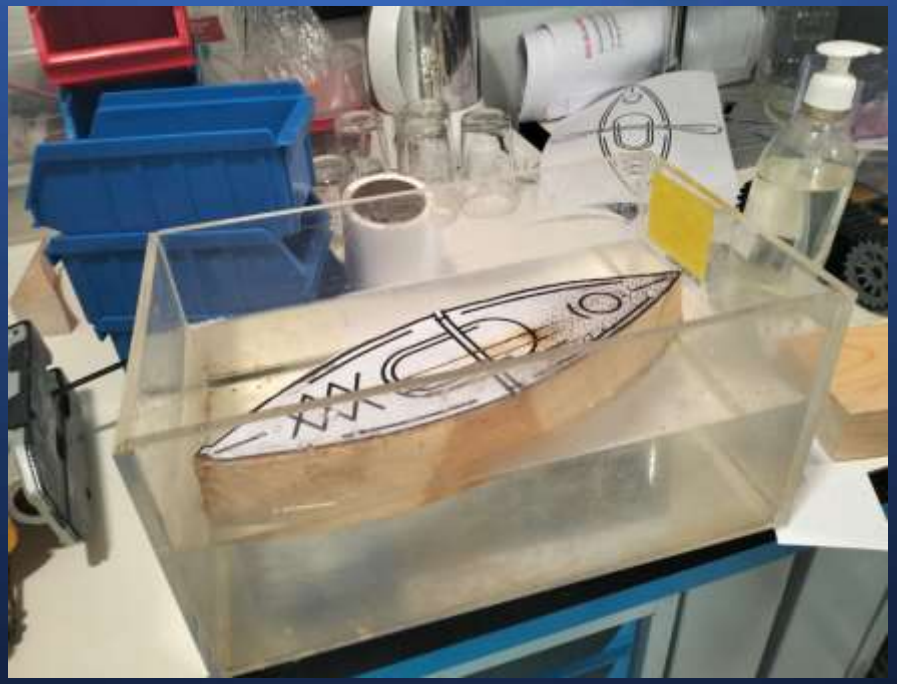
$$\frac{V_0}{V(t)} = 1 + \frac{t}{\tau} \qquad avec \qquad \tau = \frac{2M_e}{\rho S C_D V_0}$$



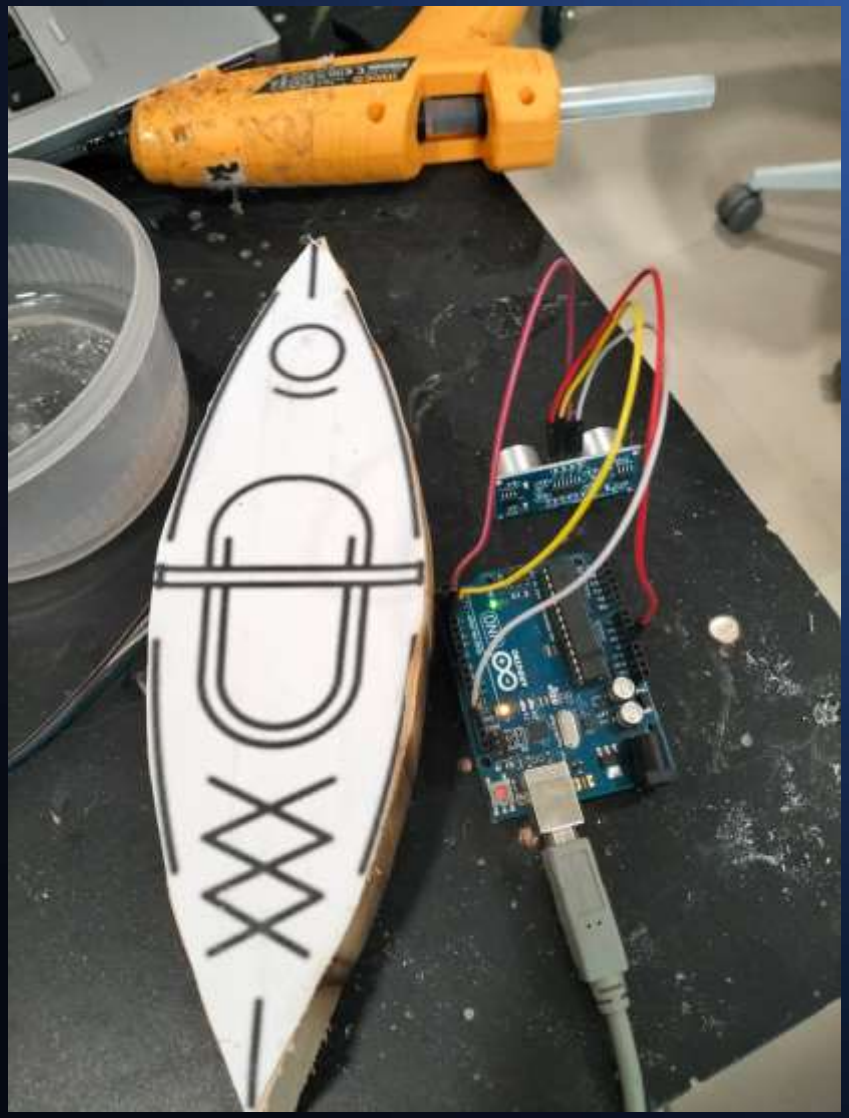


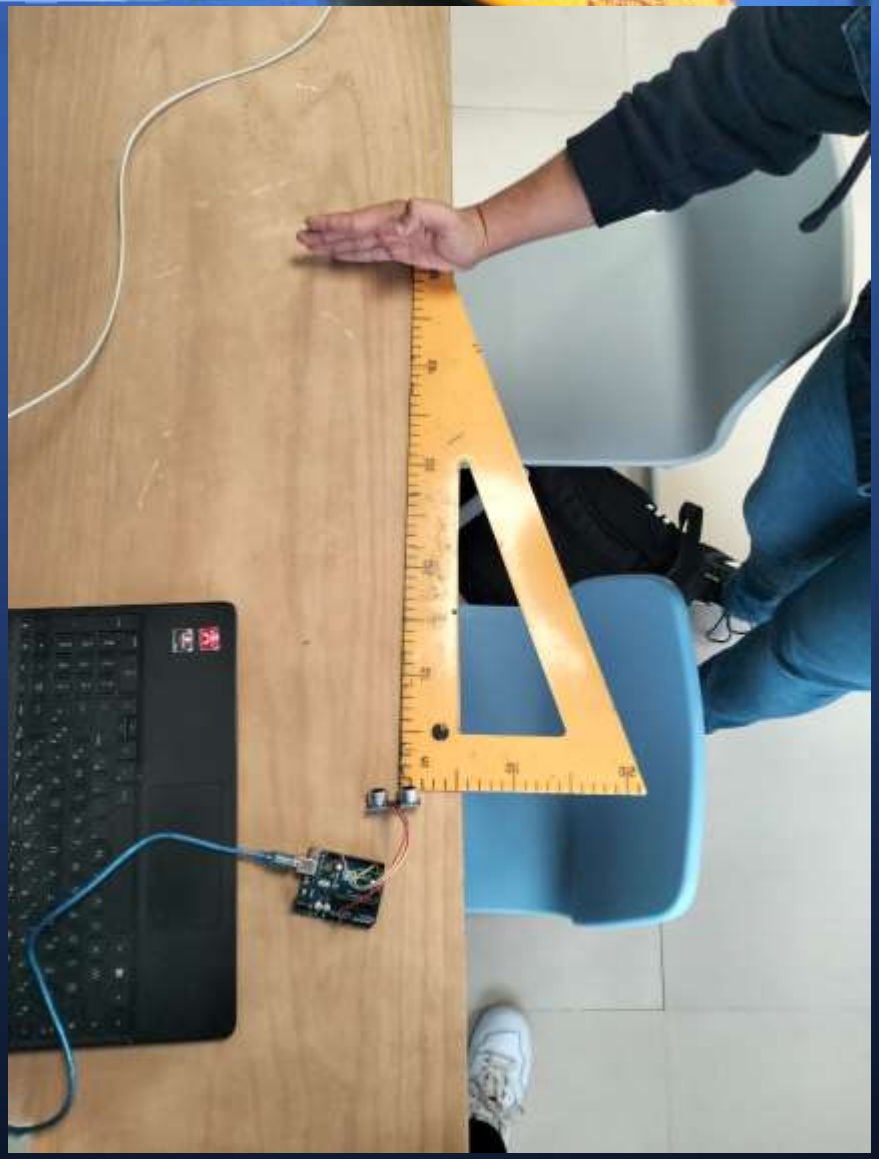
Réalisation

Matériel utilisé



Matériel utilisé



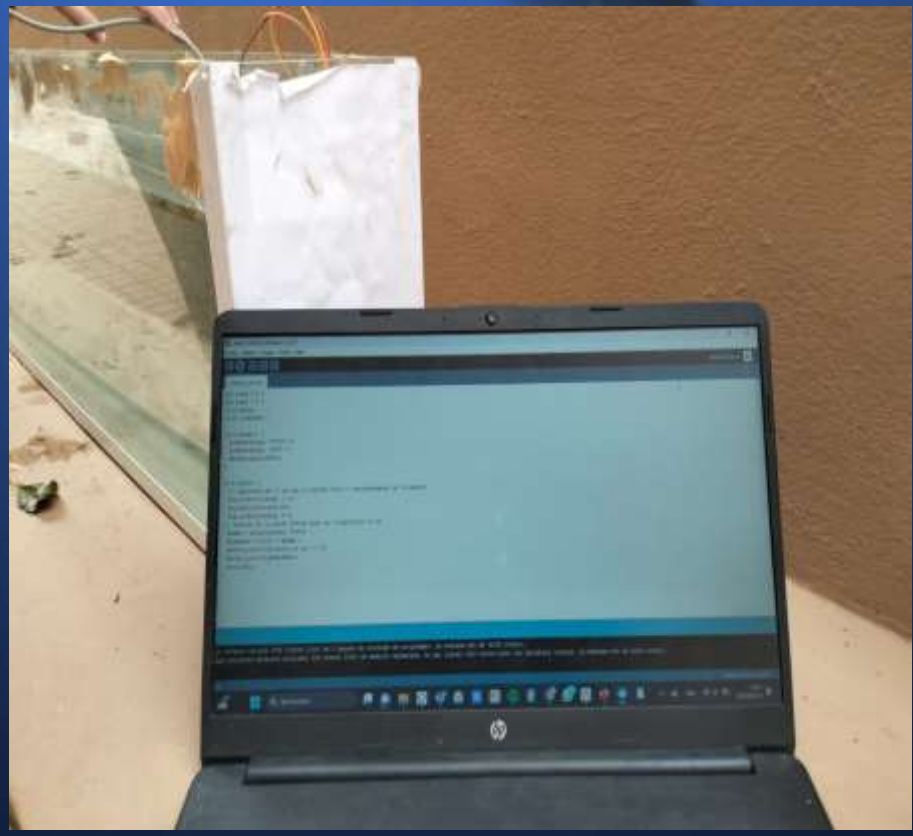


Code Arduino

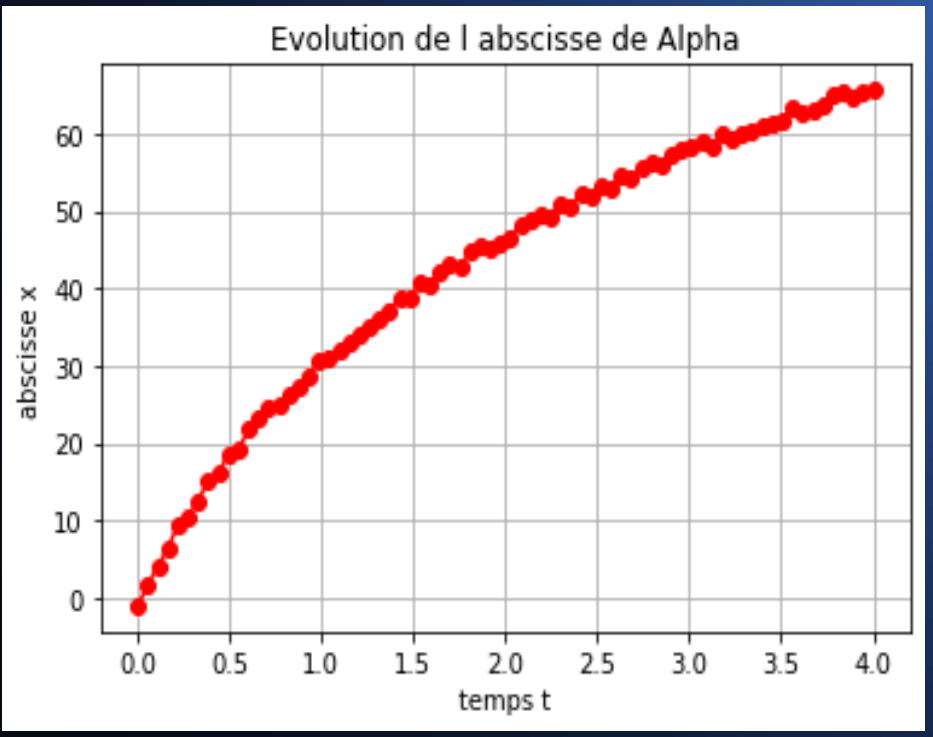
sketch_oct31a

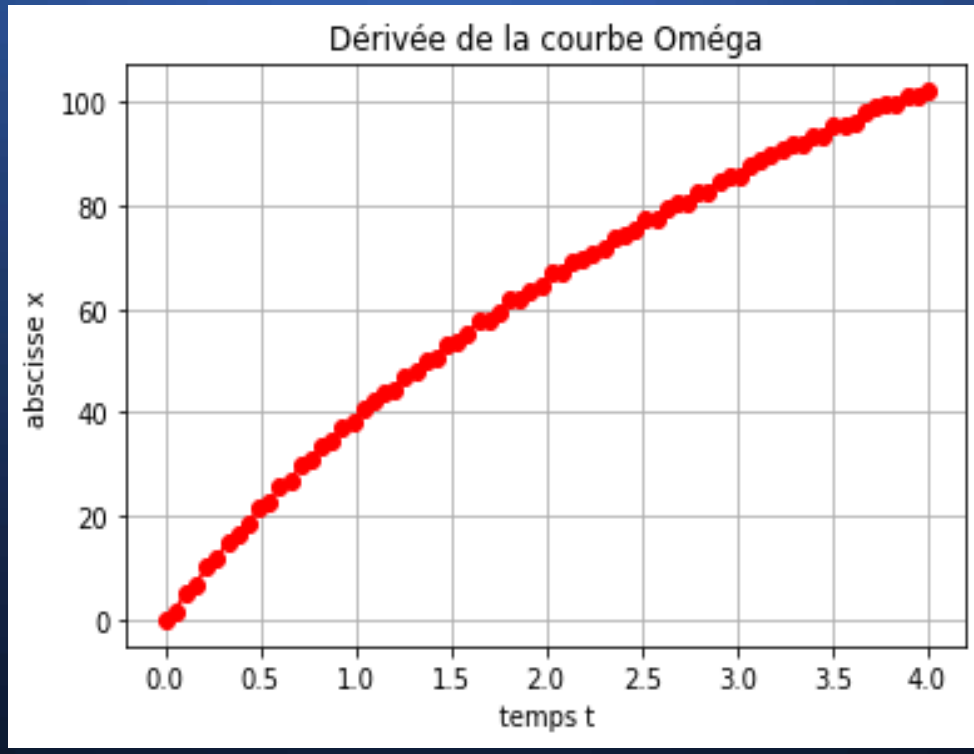
```
int trig = 3 ;
int echo = 2 ;
long duree;
float distance ;

void setup() {
  pinMode(trig, OUTPUT );
  pinMode(echo, INPUT );
  Serial.begin(9600);
}

void loop() {
  // impulsion de 10 us sur la broche trig => déclenchement de la mesure
  digitalWrite(trig, 1 );
  delayMicroseconds(10);
  digitalWrite(trig, 0 );
 // lecture de la durée d'état haut de l'impulsion en us
  duree = pulseIn(echo, HIGH);
  distance = 0.017 * duree ;
  Serial.print("distance en cm  : ");
  Serial.println(distance);
  delay(10);
}
```

Mise en oeuvre



Résultats

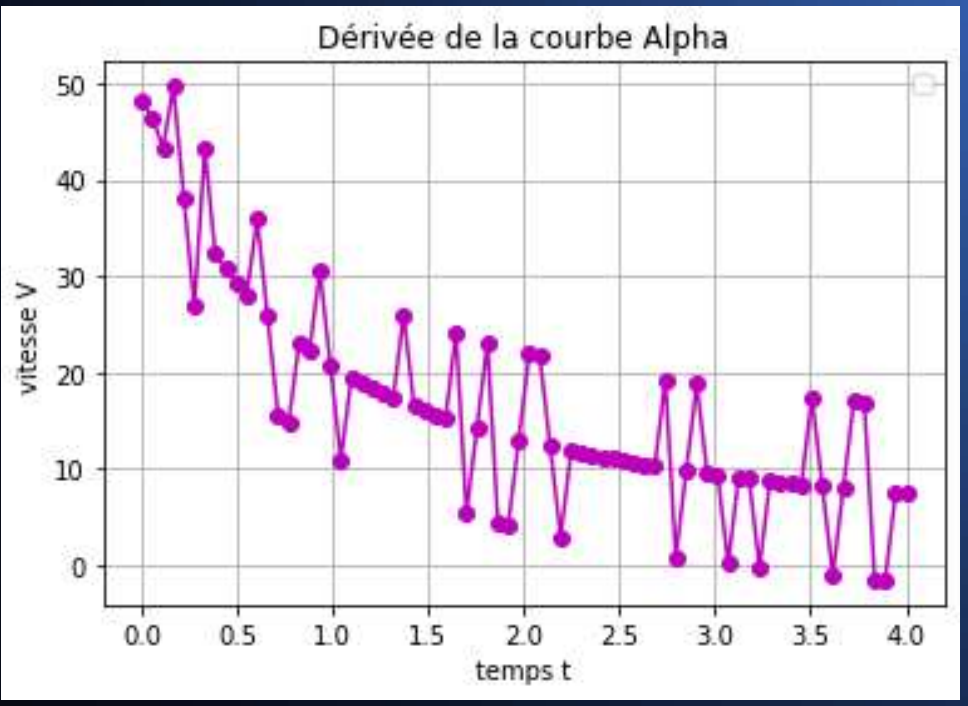


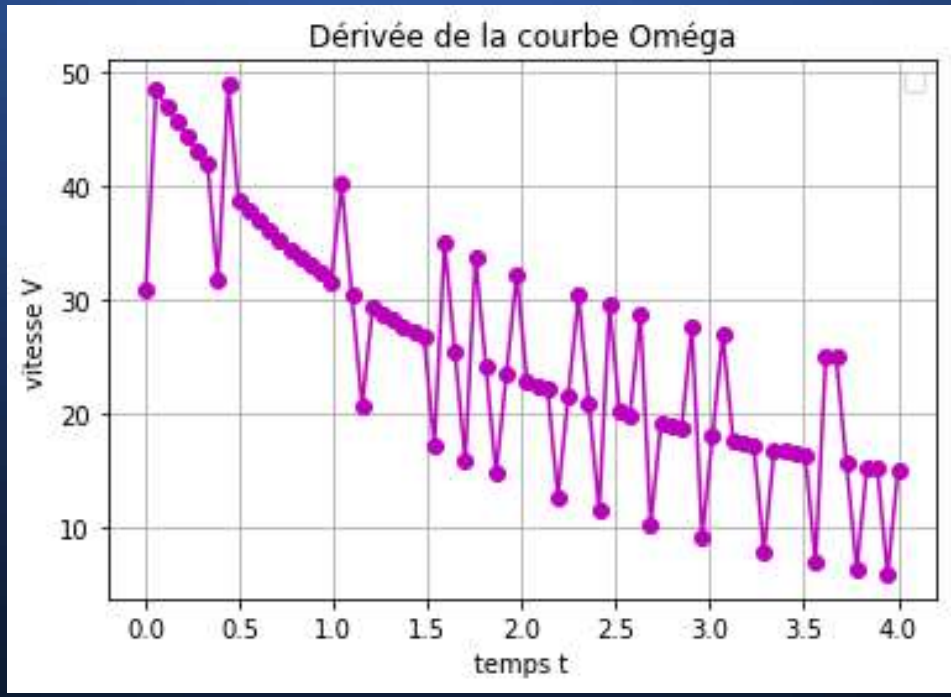


Code Python

```
5   @author: yahya
6   """
7
8   alpha=[(0.0, -1.0), (0.0547945205479452, 1.6377826632223518),(0.1095890410958904, 4.090624952083782),
9   omega=[(0.0, 0.0), (0.0547945205479452, 1.6967464490105058), (0.1095890410958904, 5.3110294476641755),
10
11  import matplotlib.pyplot as plt
12  import numpy as np
13  from sklearn.linear_model import LinearRegression
14
15  def tracer_courbe(points):
16      x = [point[0] for point in points]
17      y = [point[1] for point in points]
18      plt.plot(x, y, marker='o',color='r')
19      if points==alpha:
20          plt.title('Dérivée de la courbe Alpha')
21      else:
22          plt.title('Dérivée de la courbe Oméga')
23      plt.xlabel('temps t')
24      plt.ylabel('abscisse x')
25      plt.grid(True)
26      plt.show()
27  tracer_courbe(alpha)
28  tracer_courbe(omega)
```

Résultats

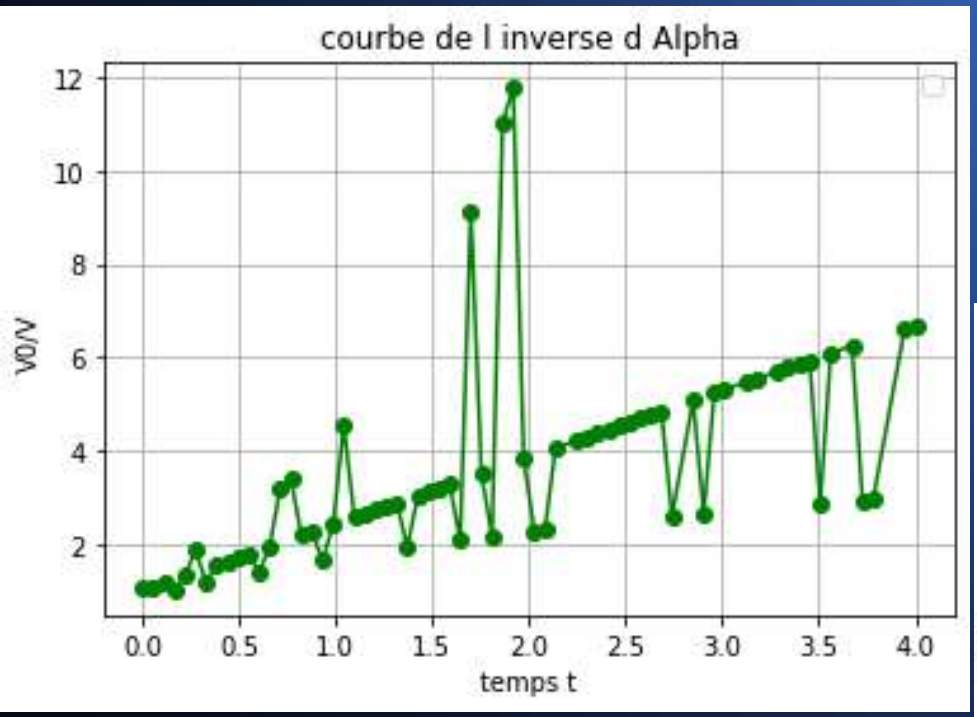


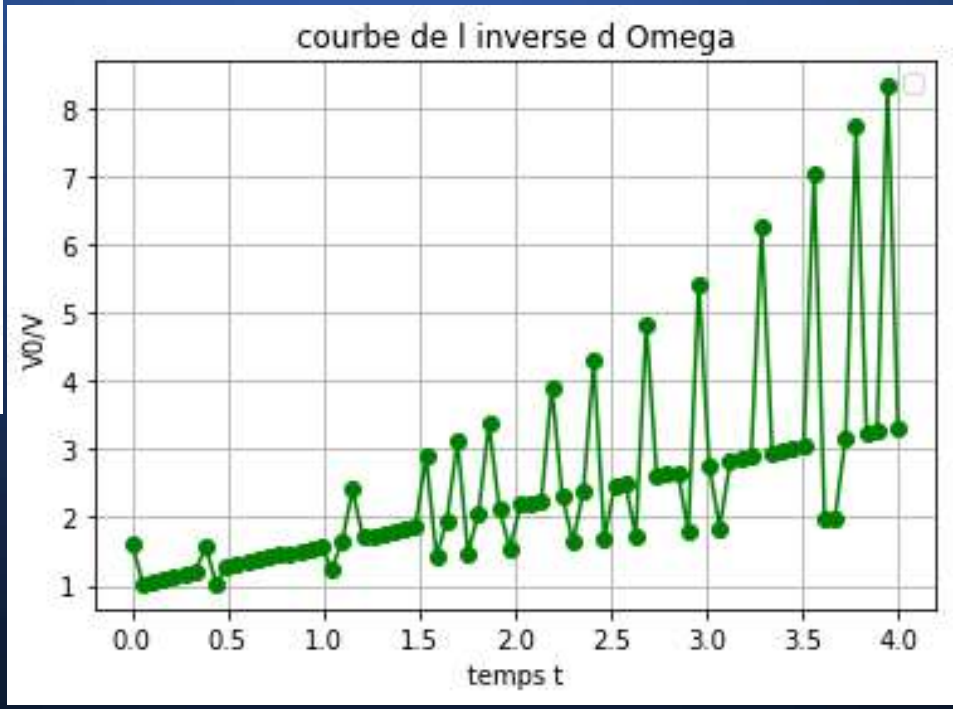


Code Python

```
30  def g(v):
31      #V0=50 cm/s
32      return 50/v
33
34  def dérivée(points):
35      x = [point[0] for point in points]
36      y = [point[1] for point in points]
37      y_prime = np.gradient(y, x)
38      plt.plot(x, y_prime,marker='o',color='m')
39      if points==alpha:
40          plt.title('Dérivée de la courbe Alpha')
41      else:
42          plt.title('Dérivée de la courbe Oméga')
43      plt.xlabel('temps t')
44      plt.ylabel('vitesse V')
45      plt.legend()
46      plt.grid(True)
47      plt.show()
48  dérivée(alpha)
49  dérivée(omega)
```

Résultats

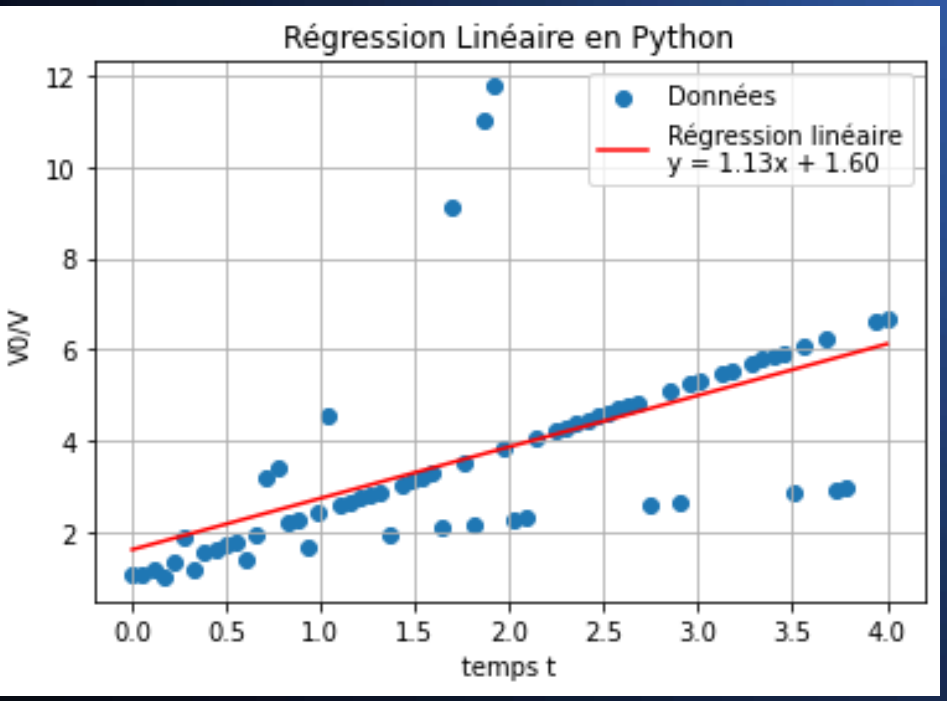


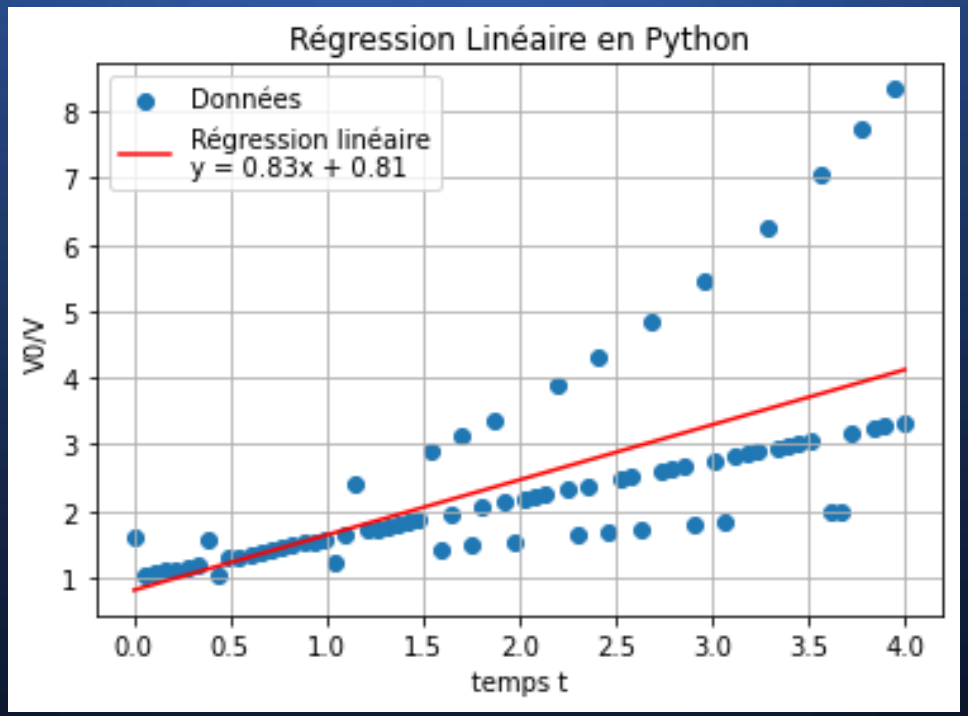


Code Python

```
51      def condition(d):
52          for e in d:
53              if e<0 or cnd(e,d):
54                  return False
55          return True
56      def cnd(r,points):
57          return np.abs(r)>12
58      def inverse(points):
59          x = [point[0] for point in points]
60          y = [point[1] for point in points]
61          y_prime = list(np.gradient(y, x))
62          inv=list(map(g,y_prime))
63          #on utilise un filtre de valeur
64          while not condition(inv):
65              for elt in inv:
66                  if elt<0 or cnd(elt,points):
67                      x.remove(x[inv.index(elt)])
68                      inv.remove(elt)
69          plt.plot(x, inv,marker='o',color='g')
70          if points==alpha:
71              plt.title('courbe de l inverse d Alpha')
72          else:
73              plt.title('courbe de l inverse d Omega')
74          plt.xlabel('temps t')
75          plt.ylabel('V0/V')
76          plt.legend()
77          plt.grid(True)
78          plt.show()
79      inverse(alpha)
80      inverse(omega)
81
```

Résultats





Code Python

```
82  def régressionlinéaire(points):
83      x = [point[0] for point in points]
84      y = [point[1] for point in points]
85      y_prime = list(np.gradient(y, x))
86      inv=list(map(g,y_prime))
87      #on utilise un filtre de valeur
88      while not condition(inv):
89          for elt in inv:
90              if elt<0 or cnd(elt,points):
91                  x.remove(x[inv.index(elt)])
92                  inv.remove(elt)
93      X=np.array(x).reshape(-1,1)
94      y=np.array(inv).reshape(-1,1)
95      model = LinearRegression()
96      model.fit(X, y)
97      slope = model.coef_[0][0]
98      intercept = model.intercept_[0]
99      y_pred = model.predict(X)
100     plt.scatter(X, y, label='Données')
101     plt.plot(X, y_pred, color='red', label=f'Régression Linéaire\ny = {slope:.2f}x + {intercept:.2f}')
102     plt.xlabel('temps t')
103     plt.ylabel('V0/V')
104     plt.title('Régression Linéaire en Python')
105     plt.legend()
106     plt.grid(True)
107     plt.show()
108 régressionlinéaire(alpha)
109 régressionlinéaire(omega)
```

Résultats

Coefficient de frottement(m^2) :

- Alpha: $7{,}1.10-3$
- Omega:$5{,}97.10-3$

Incertitude:

- Capteur HC-SR04: 4% à 13%
- Hypothèses simplificatrices(force des vagues,mise en oeuvre…)

Paramètres:

$\rho = 10^3$ kg/ m^3

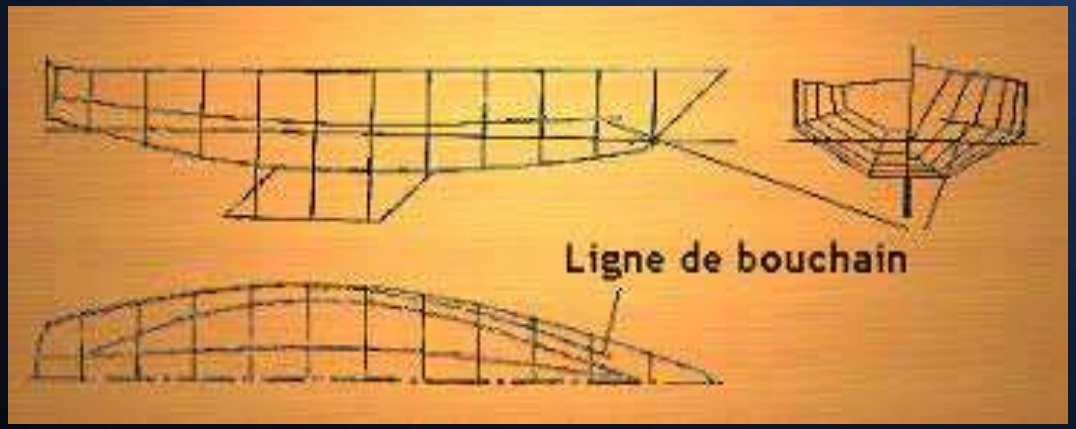
M_1=2 kg

M_2=1,3 kg

V_0=0,5m/s

Synthèse

- Forme optimale:
- ➢ minimisant la force de trainée
- ➢ Inhérent aux caractéristiques géométriques de la coque uniquement
- ➢ Matériau et poids fixés par la réglementation
- ➢ Dans l'idéal une coque à bouchains mince



MERCI POUR VOTRE ATTENTION