



# Analyse expérimentale et numérique d'un lancer sous effet Magnus

# Sommaire :

## I. Création du lanceur :

Initial : Introduction

1. Théorie électronique
2. Montage expérimental
3. Conditions initiales

## II. Description numérique des écoulements de fluide :

1. Modélisation du milieu
2. Modèle de viscosité et équation de transport
3. Résultats numériques

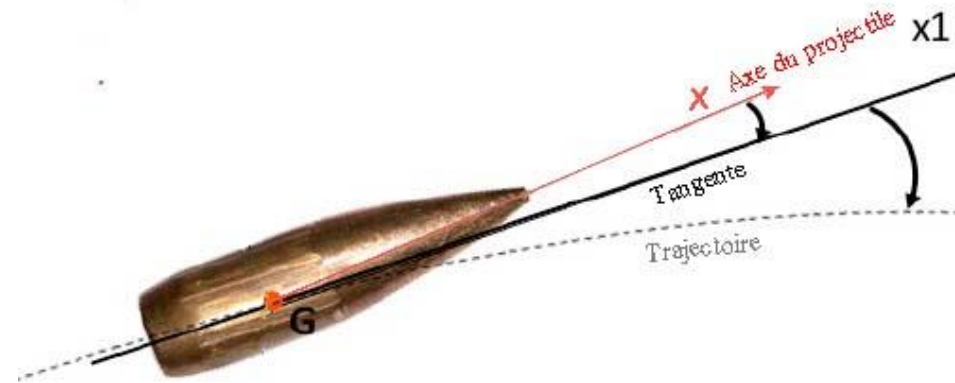
## III. Comparaison :

1. Confrontation statistique théorie expérience
2. Numérique et théorie analytique

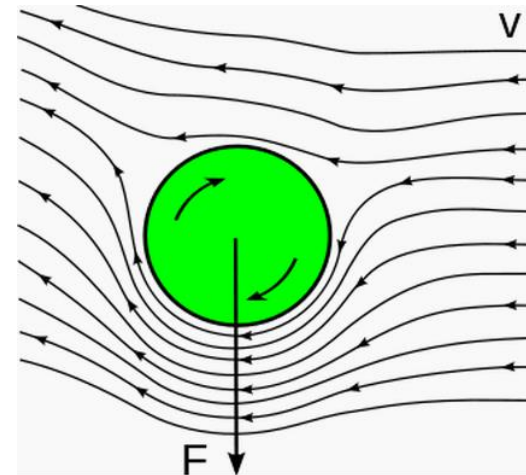
## IV. Conclusion

# Domaines d'exploitations :

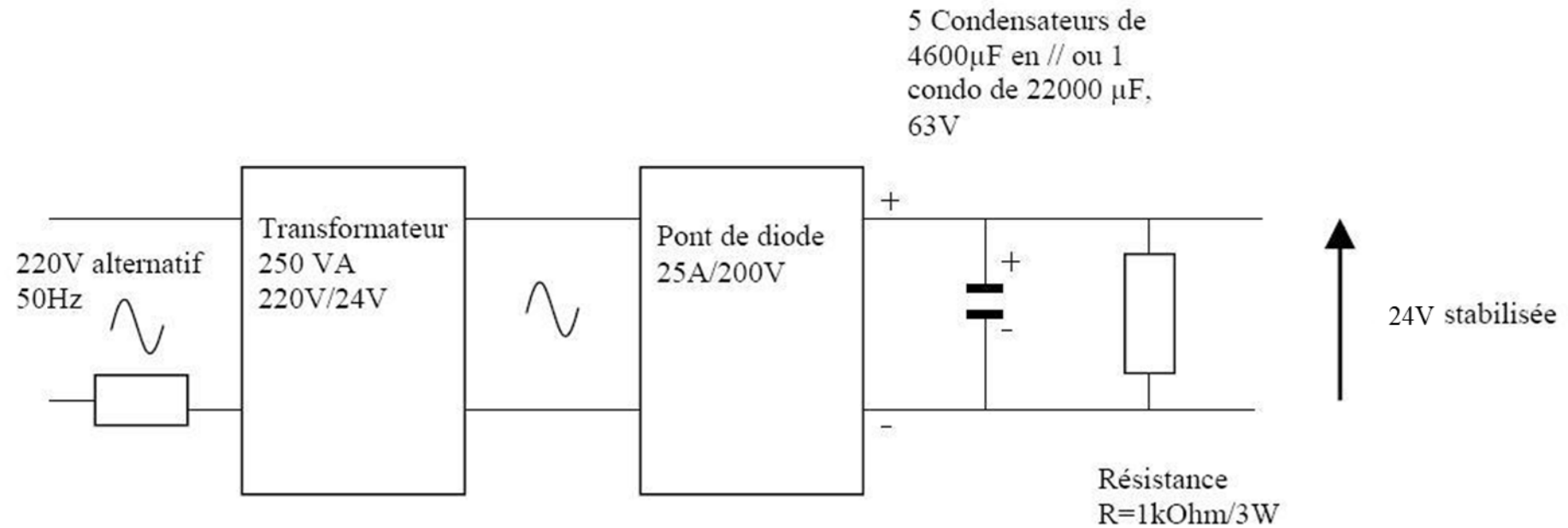
- Balistique



- Sport

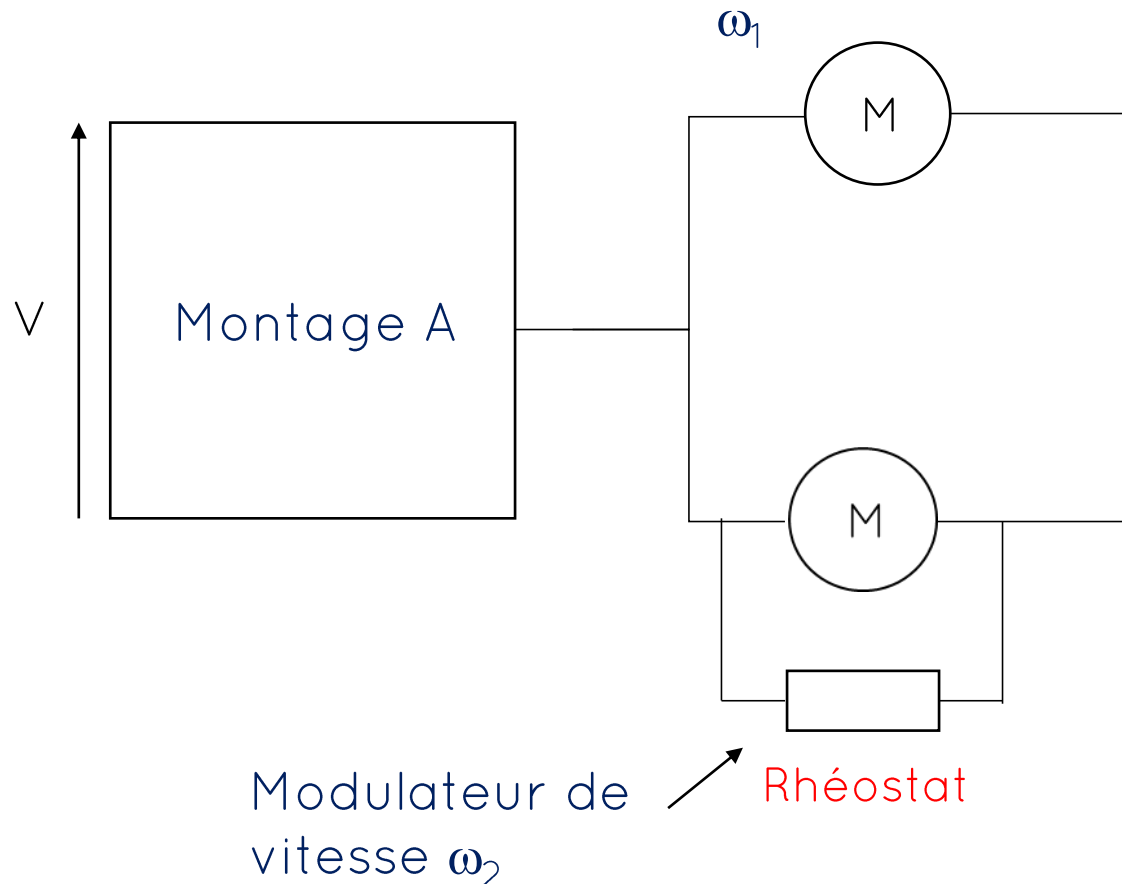


# 1. Théorie électronique :

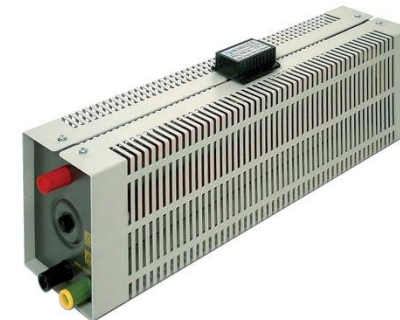


Montage A

## b) Schéma et composants :



Moteur 24V CC 6500t/minutes



Rhéostat

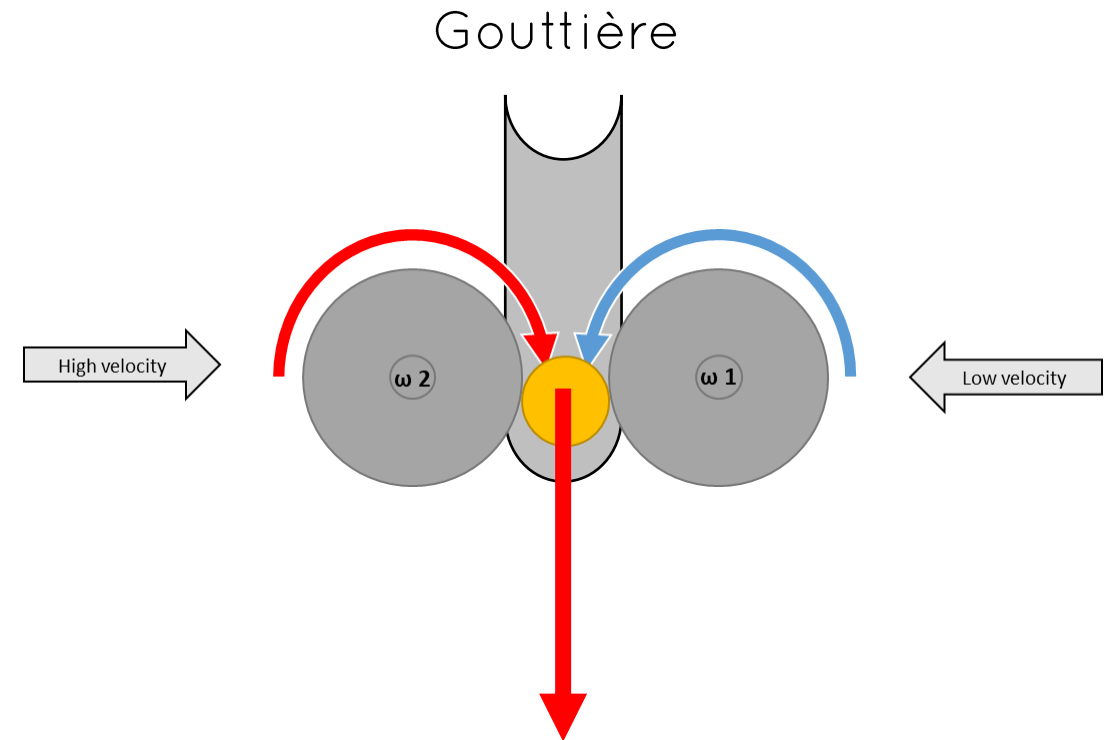
## 2. Montage expérimental :



Roues

Partie  
amovible

Rampe de lancement





### 3. Conditions initiales :

$$v_0 = 4,1 \pm 0,3 \text{ m.s}^{-1}, \text{ calculée avec Tracker}$$

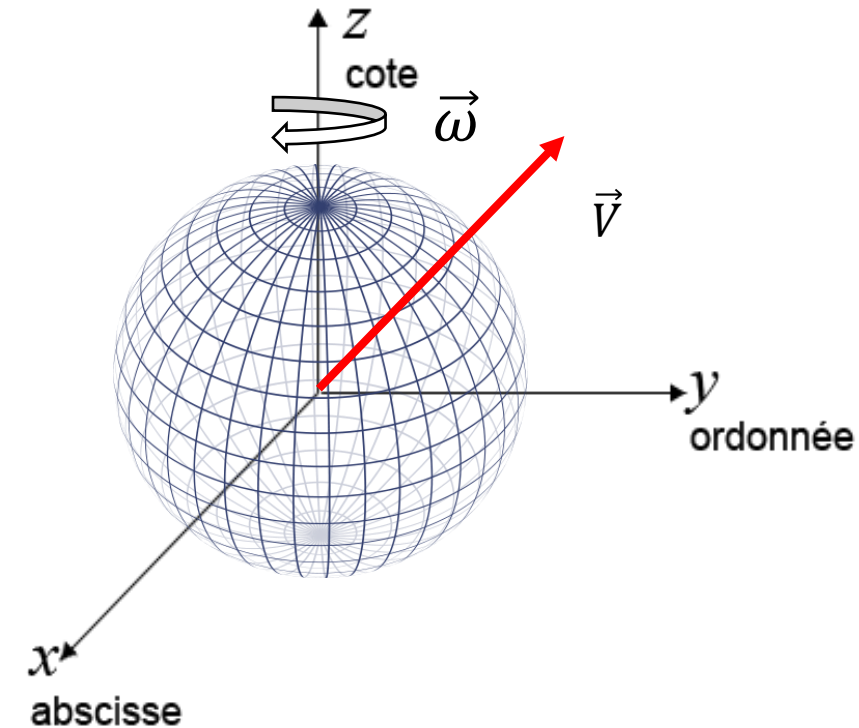
$$\omega = 99,4 \pm 1,6 \text{ rad.s}^{-1}$$

$$Re = \frac{\mu UL}{\eta} \sim 1,4 * 10^4 \text{ l'approche sera turbulente}$$

$$\text{Nombre de Mach : } Ma = \frac{V}{a} = 0,012, \text{ l'écoulement est subsonique}$$

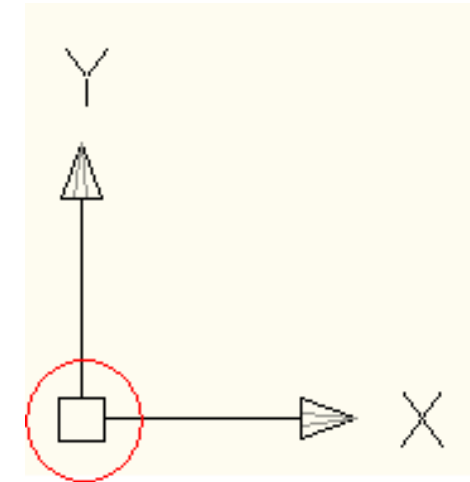
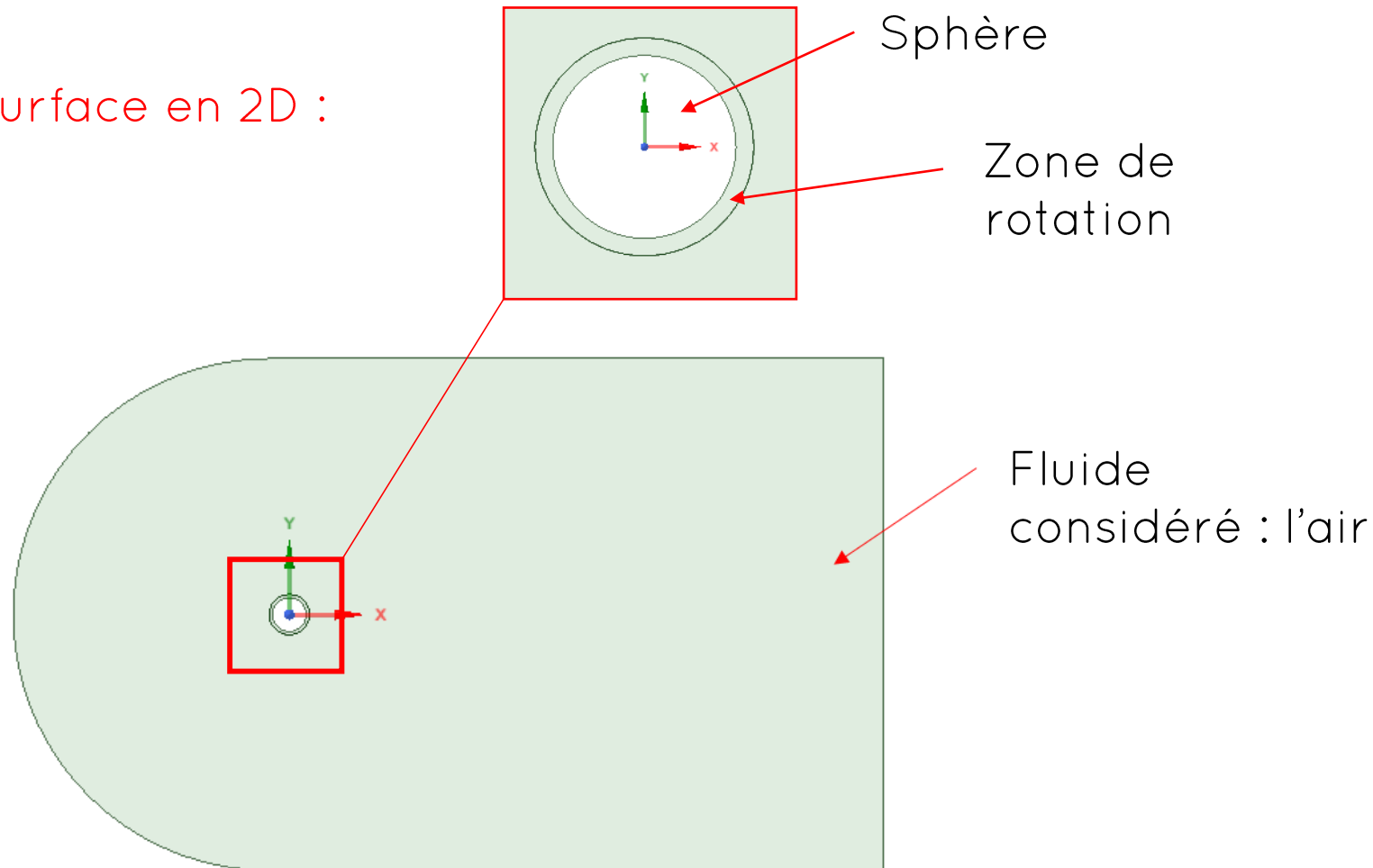
$Ma < 0,3$ , l'écoulement sera donc considéré comme incompressible

Ce sera notre seule hypothèse quand à la nature du fluide



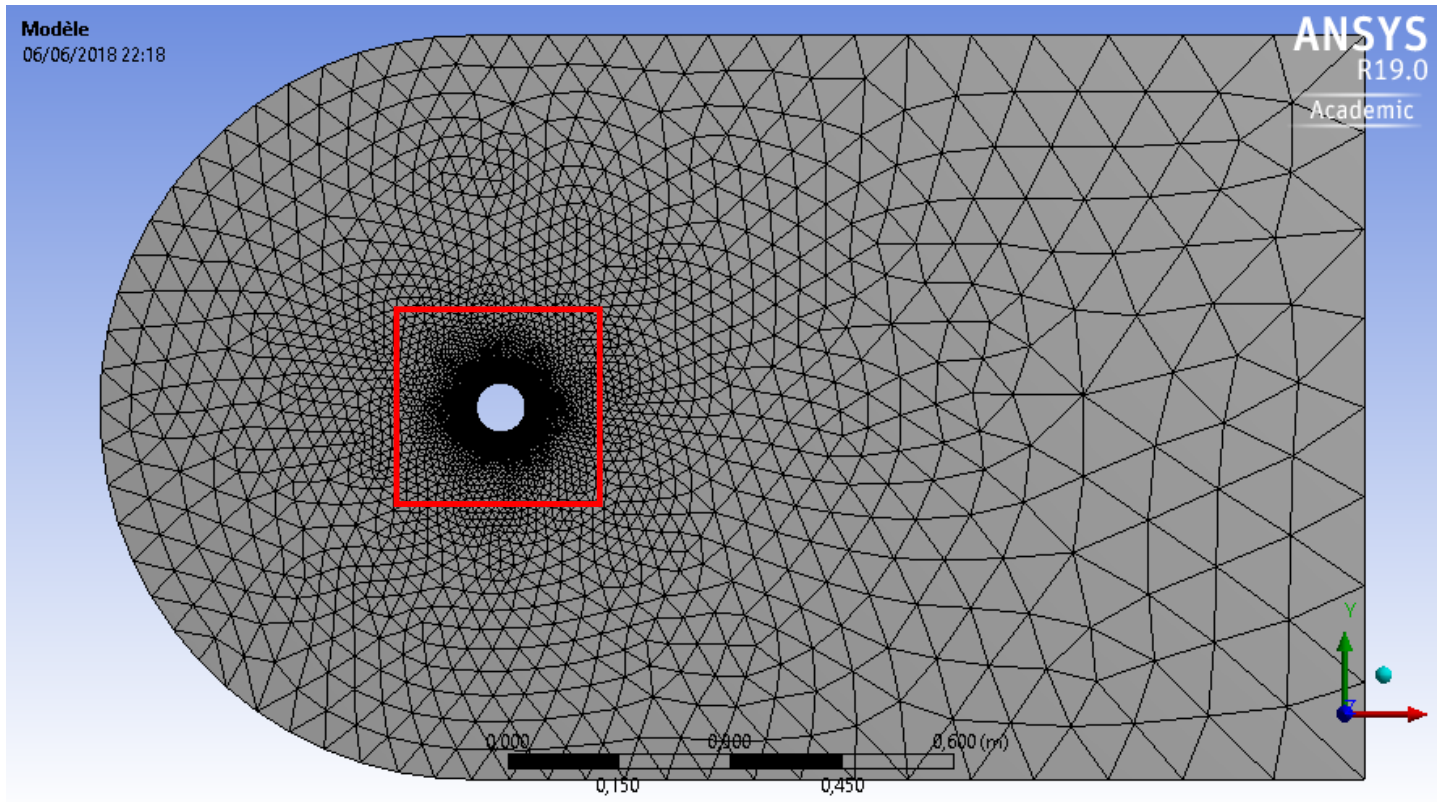
# 1. Modélisation du milieu :

Surface en 2D :

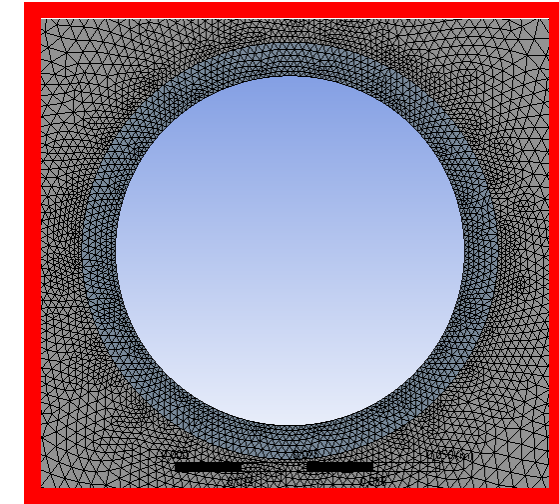




## b) Maillage :



- Méthode des triangles seuls
- Maillage des interfaces fluide/fluide en rotation
- Taille des unités



Maillage plus fin pour  
une meilleure approche  
descriptive

Objectif :

- Découper l'espace en milliers d'éléments finis pour appliquer une analyse numérique ultérieure

## 2. Un modèle de viscosité :

### Realizable k-ε

### Equations de transport :

#### Energie cinétique :

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k u_i)}{\partial x_i} = \frac{\partial}{\partial x_j} \left[ \frac{\mu_t}{\sigma_k} \frac{\partial k}{\partial x_j} \right] + 2\mu_t E_{ij} E_{ij} - \rho \epsilon$$

#### Dissipation :

$$\frac{\partial(\rho \epsilon)}{\partial t} + \frac{\partial(\rho \epsilon u_i)}{\partial x_i} = \frac{\partial}{\partial x_j} \left[ \frac{\mu_t}{\sigma_\epsilon} \frac{\partial \epsilon}{\partial x_j} \right] + C_{1\epsilon} \frac{\epsilon}{k} 2\mu_t E_{ij} E_{ij} - C_{2\epsilon} \rho \frac{\epsilon^2}{k}$$

### Variables :

- $U_i$  la vitesse dans la direction considérée
- $E_{ij}$  le module d'Young
- $\mu_t$  eddy viscosity

$$\mu_t = \rho C_\mu \frac{k^2}{\epsilon}$$

$$C_\mu = 0.09$$

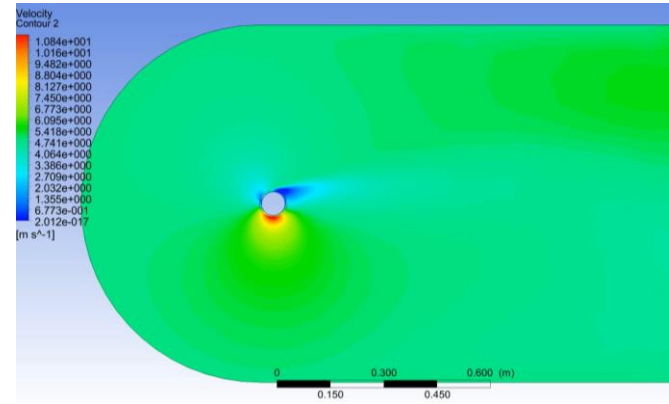
$$\sigma_k = 1.00$$

$$\sigma_\epsilon = 1.30$$

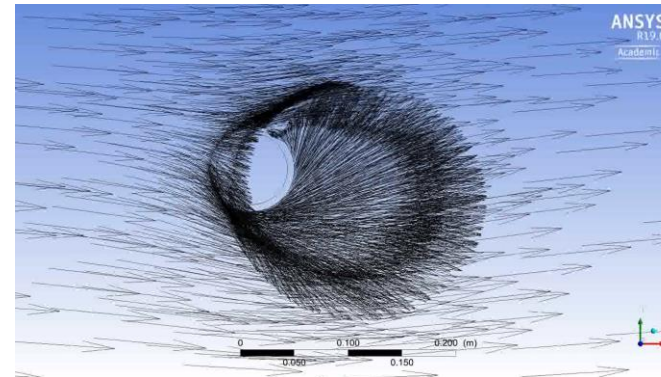
$$C_{1\epsilon} = 1.44$$

$$C_{2\epsilon} = 1.92$$

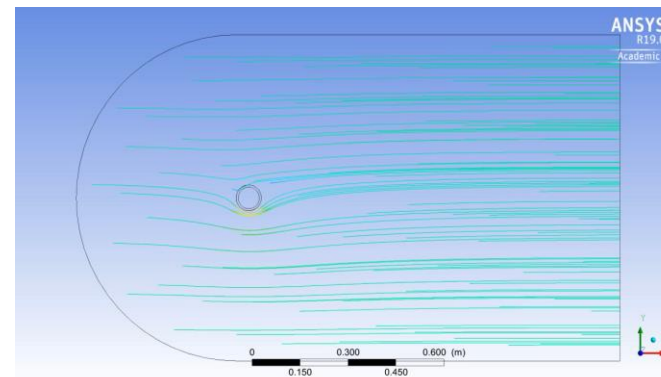
Champ des vitesses :



Champ de vecteurs vitesse :

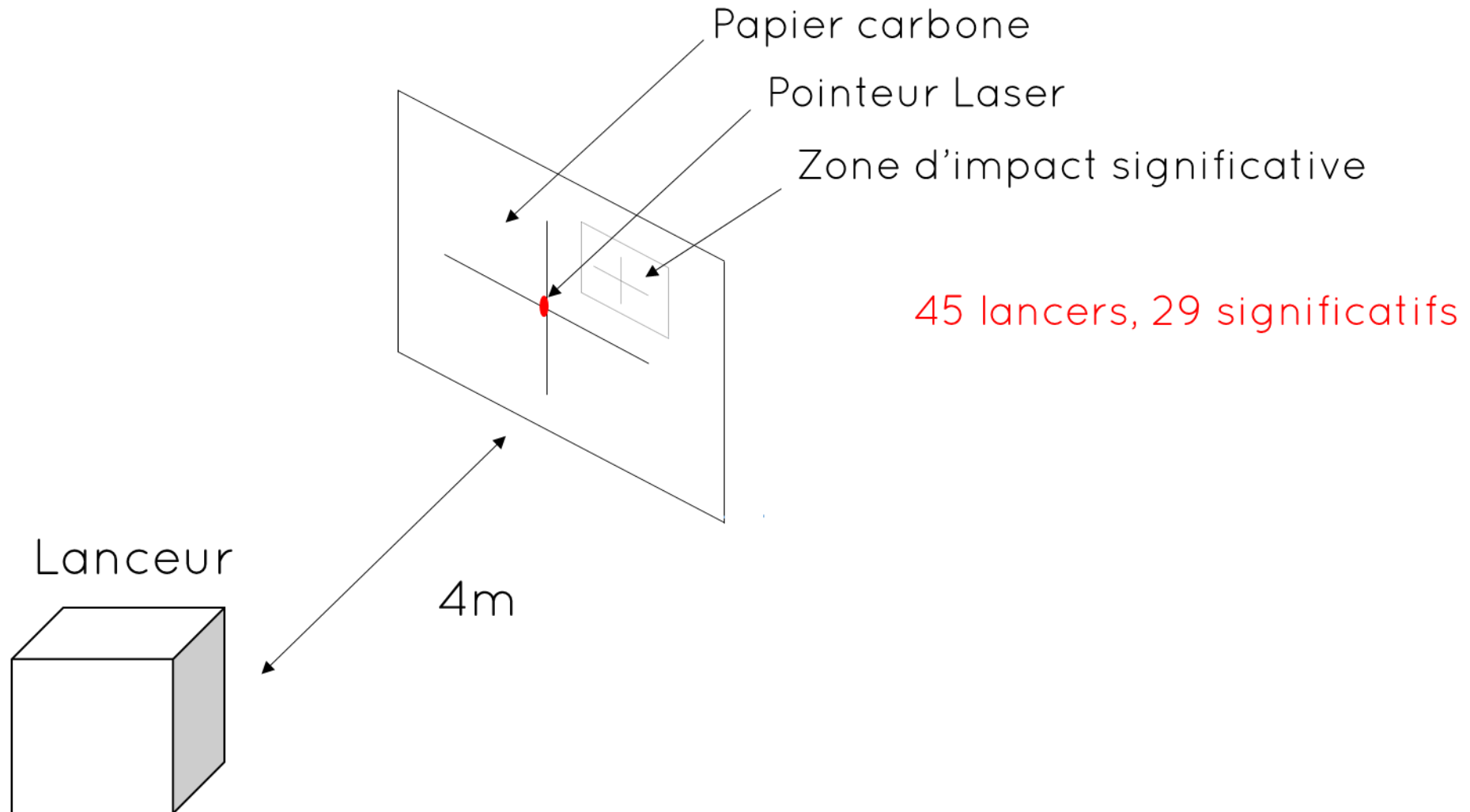


Lignes de courant :

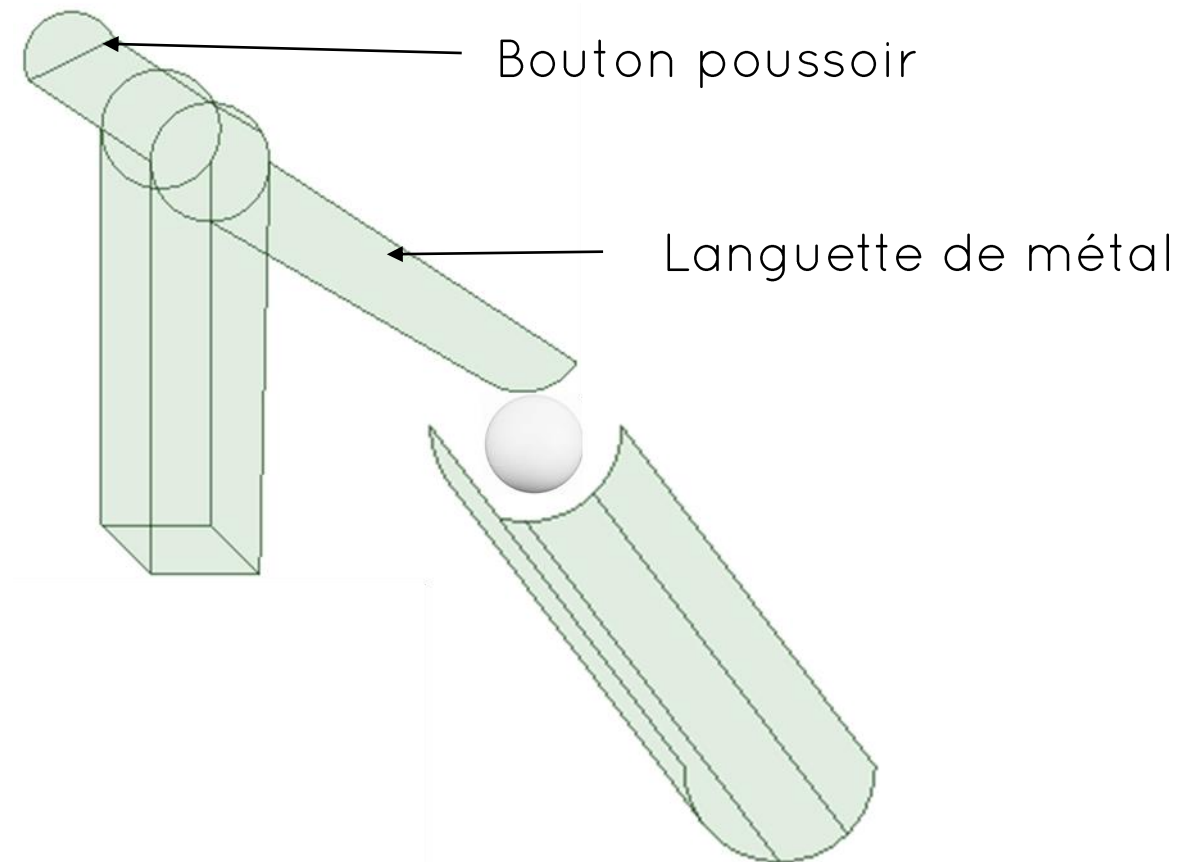


### 3. Résultats numériques

# 1. Expérience :



## b) Réduction de l'incertitude des conditions initiales du lancer



## 2. Résultats et incertitudes :

$u$  représente la moyenne de la déviation de la trajectoire expérimentale par rapport à une trajectoire linéaire

$$u_{\text{relatif}} = u_{\text{moyen}} \pm \Delta u$$

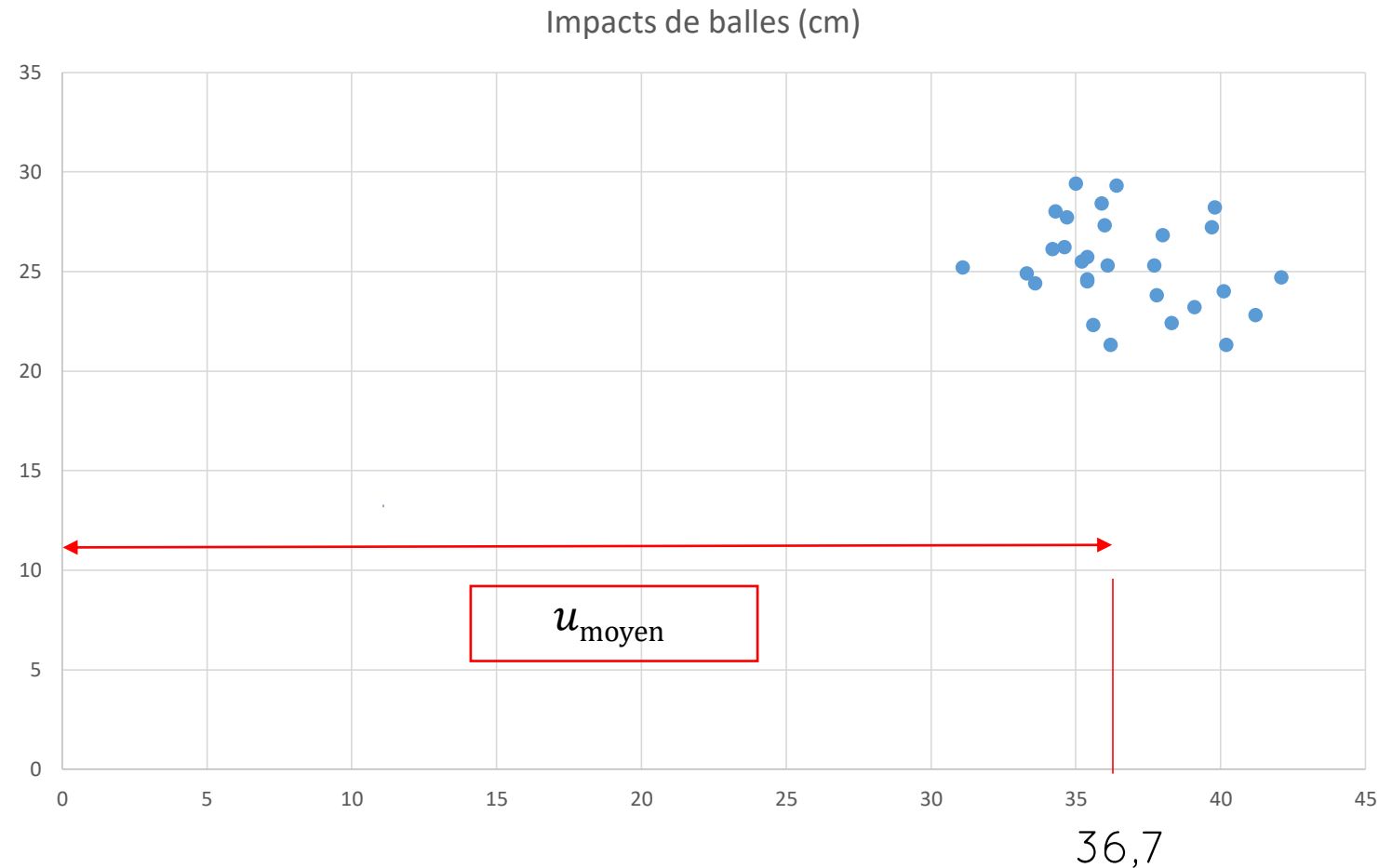
$$\text{or } u_{\text{moyen}} = 36,7 \text{ cm}$$

$$\Delta u = \sqrt{(\Delta u_{\text{lecture}})^2 + (\Delta u_{\text{mesure}})^2}$$

$$\Delta u = 2,9 \text{ cm } (\sigma = 1,02)$$

$\sigma$ , coefficient de Student

$$u_{\text{relatif}} = [36,7 \pm 2,9] \text{ cm}$$



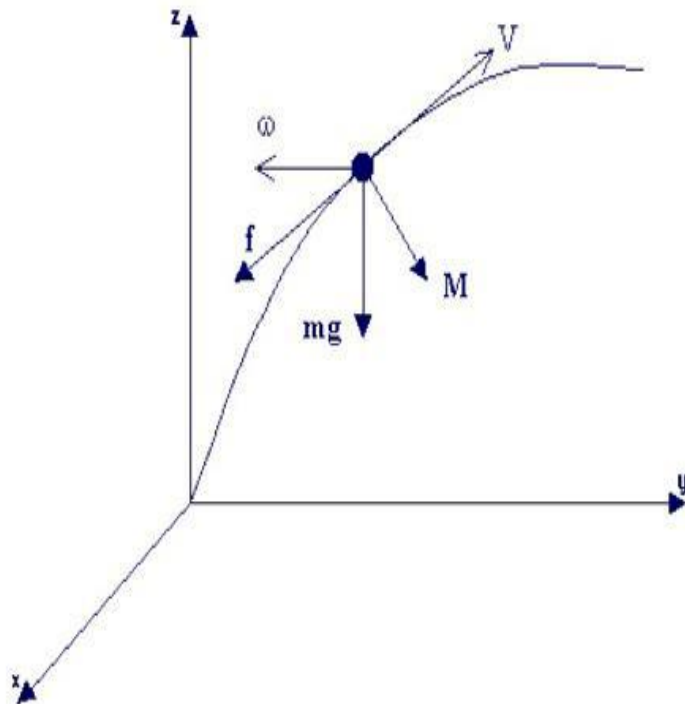


# Analyse pythonnesque de la trajectoire :

## 1. Théorie analytique

Equations couplées :

$$\begin{aligned} m\ddot{x} &= -k\sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} + a(\omega_y \dot{z} - \omega_z \dot{y}) \\ m\ddot{y} &= -k\sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} + a(\omega_z \dot{x} - \omega_x \dot{z}) \\ m\ddot{z} &= -mg - k\sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} + a(\omega_x \dot{y} - \omega_y \dot{x}) \end{aligned}$$



$V$  : vitesse de la balle

$\omega$  : vitesse angulaire de rotation de la balle

$mg$  : poids de la balle

$f$  : frottement de l'air

$M$  : force de Magnus  
perpendiculaire à  $V$  et à  $\omega$

Découplage :

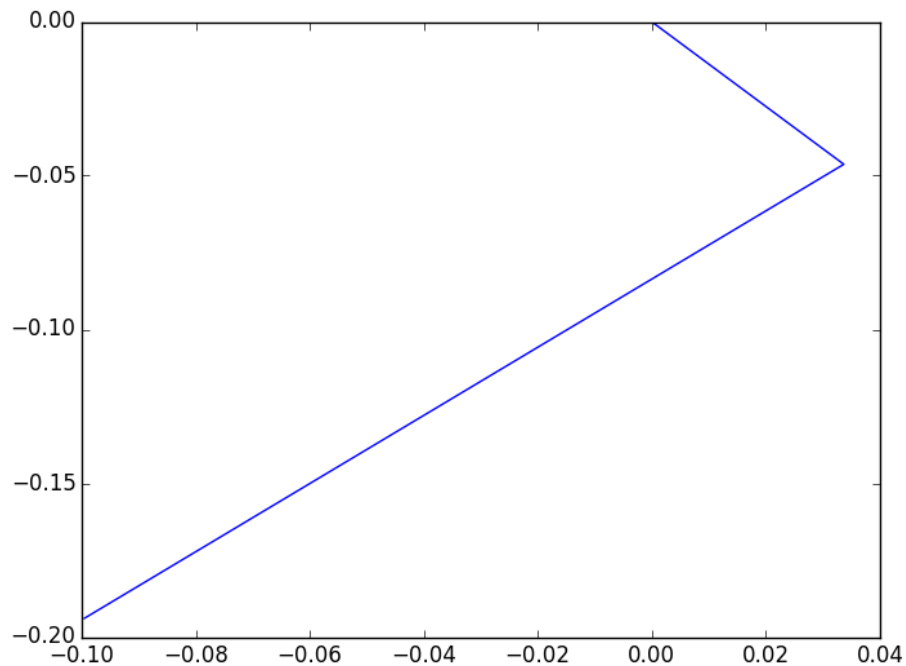
- Rotation de la balle selon une unique direction
- $V$  constante
- Mouvement plan

Solutions analytiques :

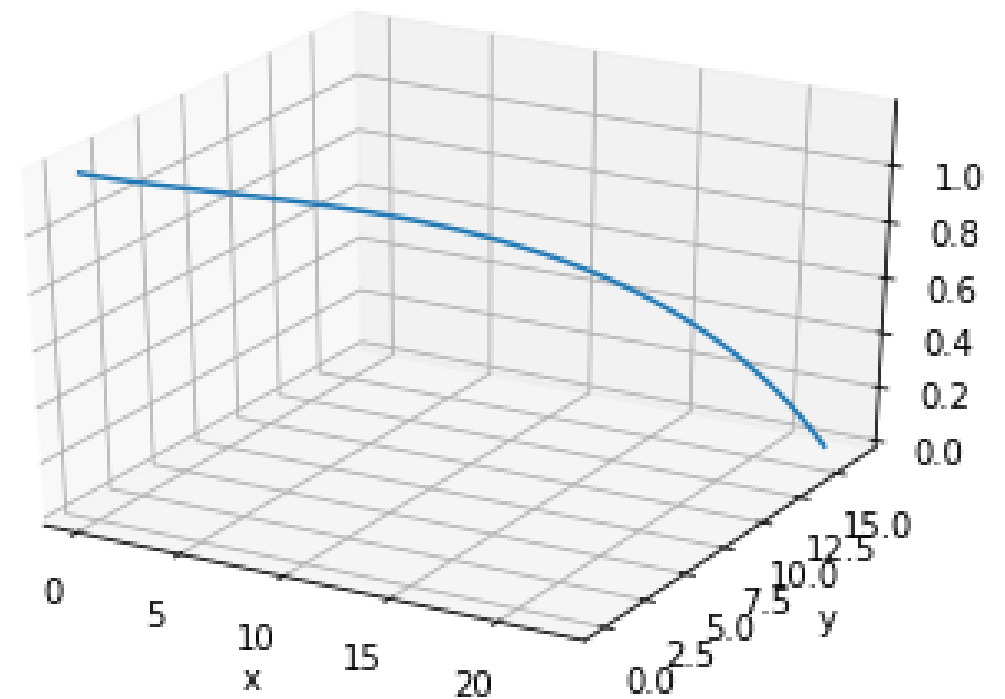
- $x(t) = e^{-Ct}(\alpha \cos(Kt) - \beta \sin(Kt)) - \frac{gKt}{C^2 + K^2} - \alpha$
- $z(t) = e^{-Ct}(\alpha \sin(Kt) a - \beta \cos(Kt)) - \frac{gCt}{C^2 + K^2} - \beta$

## 2. Résultats analytiques et numériques :

Trajectoire sur Python avec  
les équations analytiques  
trouvées précédemment



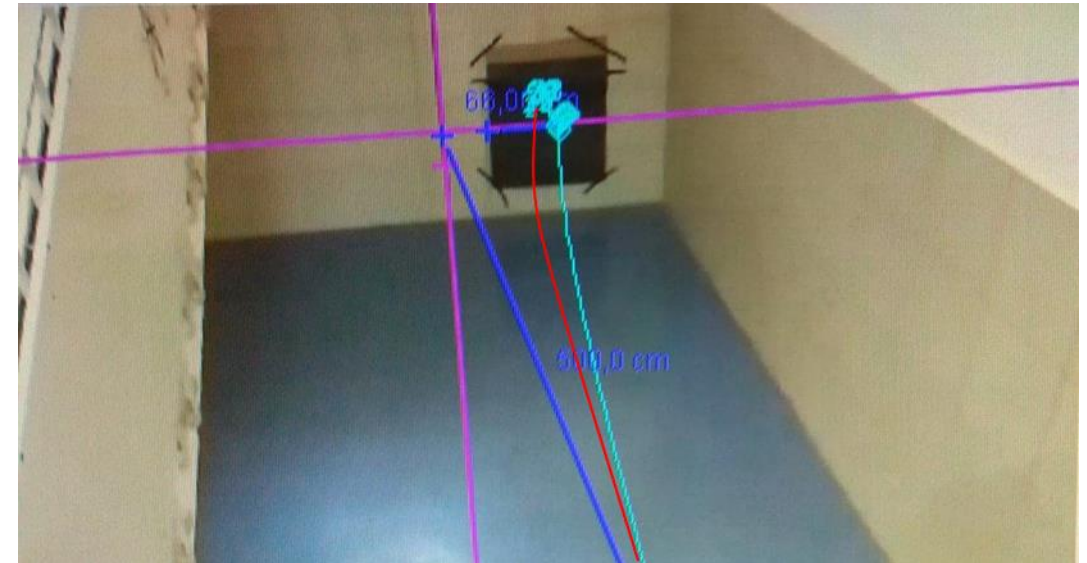
Trajectoire avec la  
méthode d'Euler sans  
découplage analytique :



## 2. Tendence à la théorie :

Théorie : 33,4 cm de déviation  
Notre lanceur : 36,7 cm de déviation  
Incertitude par rapport à la théorie = 9,8%

En rouge, la théorie analytique  
En bleu clair, la trajectoire  
expérimentale avec Tracker



# Conclusion :

Modèle	Simulation	Analytique
Précision de la gouttière, rabotage et finition du diamètre	Utilisation de la 3D pour améliorer la qualité descriptive des écoulements de fluide	Supprimer des hypothèses comme la planéité du mouvement
Nouveau système qui permet de réduire l'incertitude des conditions initiales du lancers	Créer une simulation numérique de la trajectoire sur Fluent même	La vitesse n'est pas constante, introduire une vitesse variable selon les paramètres x y z
Améliorer les matériaux des roues utilisées pour une meilleure préhension : caoutchouc vulcanisé	Meilleur maillage, utiliser des modèles de viscosité différents comme k- $\omega$	

```
from numpy import *
from matplotlib.pyplot import *

c=340.6
K=327.2
a=-0.035
b=0.044
g=9,81

t=linspace(0,10,101)

def x(t):
    return (exp(-c*t)*(a*cos(K*t)-b*sin(K*t))-(g*K*t)/(c**2+K**2)-a)

def z(t):
    return (exp(-c*t)*(b*cos(K*t)+a*sin(K*t))-(g*c*t)/(c**2+K**2)-b)

plot(x(t),z(t))
show()
```

```

from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from numpy import*

g = 9.81
m = 40*e-3
k= 0.72
l = 1.8*1e-1

x0,y0,z0 = 0,0,1.20
vx0,vy0,vz0 = 4,0,0

wx,wy,wz =0,0,100

a=0
n=101

def Euler(a,b,n): #Equations couplées
    t=linspace(a,b,n)
    x=[x0];y=[y0];z=[z0]
    vx=[vx0];vy=[vy0];vz=[vz0]
    for i in range (n-1):
        dt =t[i+1]-t[i]
        vx+=(vx[i]-(k/m)*sqrt(vx[i]**2+vy[i]**2+vz[i]**2)*vy[i]*dt+(l/m)*(wy*vz[i]-wz*vy[i])*dt]
        vy+=(vy[i]-(k/m)*sqrt(vx[i]**2+vy[i]**2+vz[i]**2)*vx[i]*dt+(l/m)*(wz*vx[i]-wx*vz[i])*dt]
        vz+=(vz[i]-(k/m)*sqrt(vx[i]**2+vy[i]**2+vz[i]**2)*vz[i]*dt+(l/m)*(wx*vy[i]-wy*vx[i])*dt]
        x +=[x[i]+vx[i]*dt] #Decouplage
        y +=[y[i]+vy[i]*dt]
        z +=[z[i]+vz[i]*dt]
    return (x,y,z)

def trajectoire(a,h,n):
    b=a
    while Euler(a,b,n)[2][-1]>=h: #La condition >h est ici imposée pour que la balle ne traverse pas le sol. On a alors
    quand z=0,h=0
        b+=1/n
    fig = plt.figure()
    ax = fig.gca(projection='3d')
    ax.plot(Euler(a,b,n)[0],Euler(a,b,n)[1],Euler(a,b,n)[2])
    plt.xlabel('x')
    plt.ylabel('y')
    plt.show()

trajectoire(a,0,n)

```