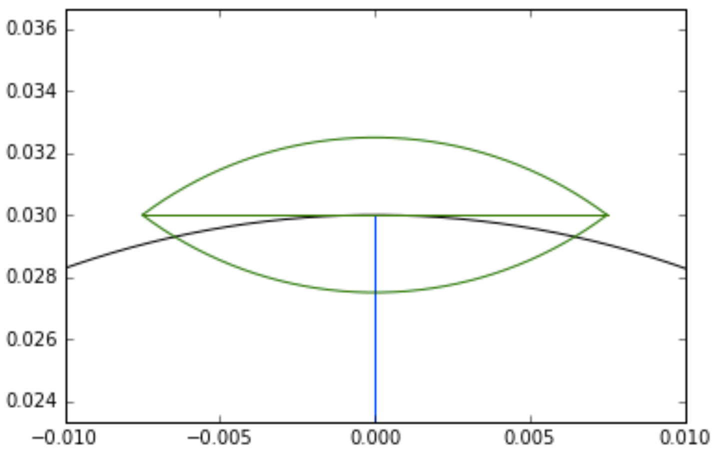
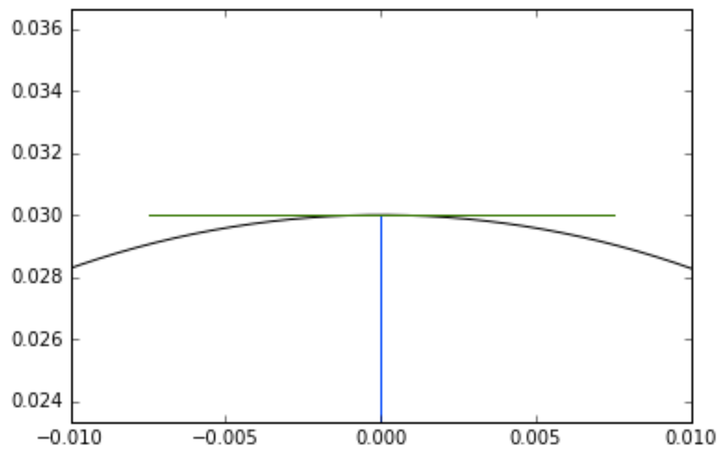
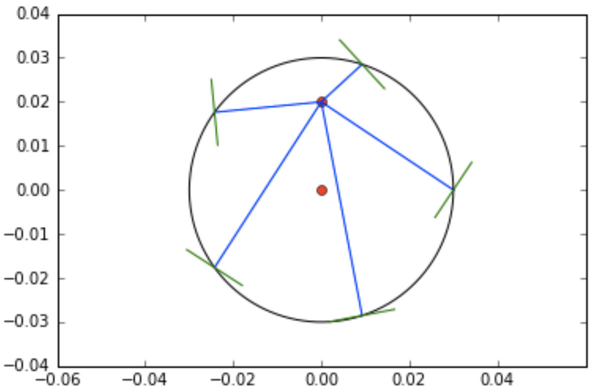
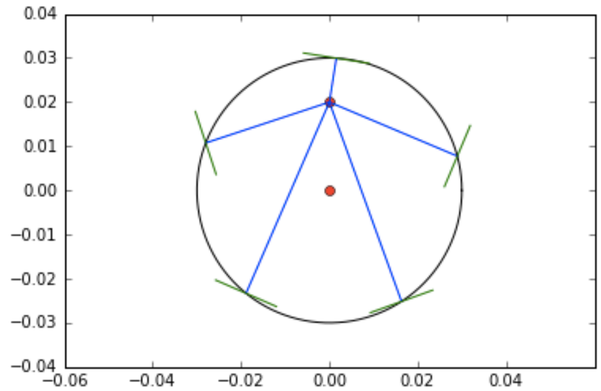
Le Propulseur Voith-Schneider

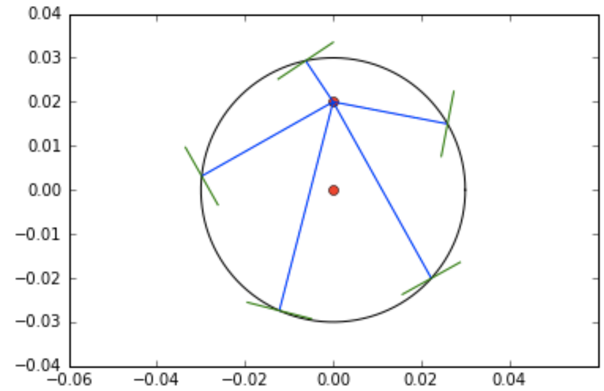
Principe de fonctionnement



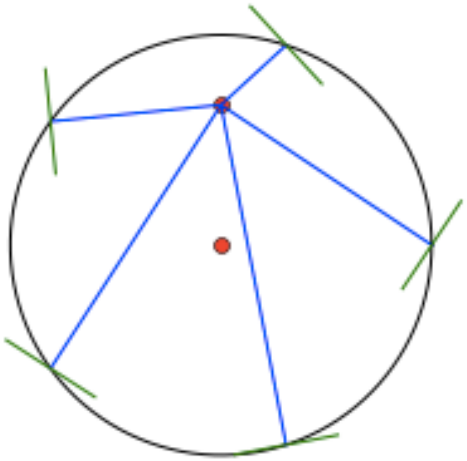


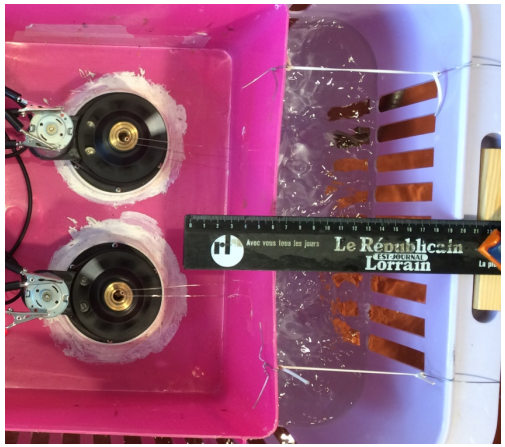






Optimalité : choix, contraintes, hasard





- I. Création d'une pale et étude de celle-ci en soufflerie
- II. Étude théorique des forces s'exerçant sur les pales
- III. Vérification de notre modèle grâce à une maquette

I. Création d'une pale et étude de celle-ci en soufflerie

- Estimation du nombre de Reynolds : $Re = \frac{\rho_{eau} \cdot V_{pale/eau} \cdot L_{Pale}}{\eta_{eau}}$

Système réel

Dimensions caractéristiques :

- Rayon du propulseur : 1 m
- Largeur d'une pale : 30 cm
- Vitesse d'un pale : 10 m/s

$$Re = \frac{1000 \cdot 10 \cdot 0{,}30}{10^{-3}} = 3 \cdot 10^{6}$$

→ Écoulement turbulent

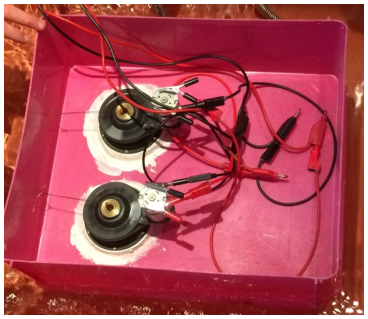
Maquette

Dimensions caractéristiques :

- Rayon du propulseur : 3cm
- Largeur d'une pale : 1.5 cm
- Vitesse d'un pale : 1 m/s

$$Re = \frac{1000 \cdot 1 \cdot 0{,}015}{10^{-3}} = 1{,}5 \cdot 10^{4}$$

→ Écoulement turbulent

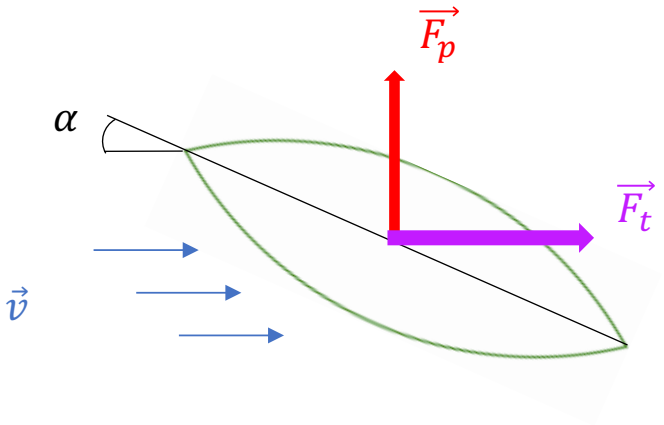


I. Création d'une pale et étude de celle-ci en soufflerie

Modèle retenu pour les pales du moteur:

$$F_t = \frac{1}{2} \cdot C_t \cdot \rho \cdot S \cdot v^2$$

$$F_p = \frac{1}{2} \cdot C_p \cdot \rho \cdot S \cdot v^2$$

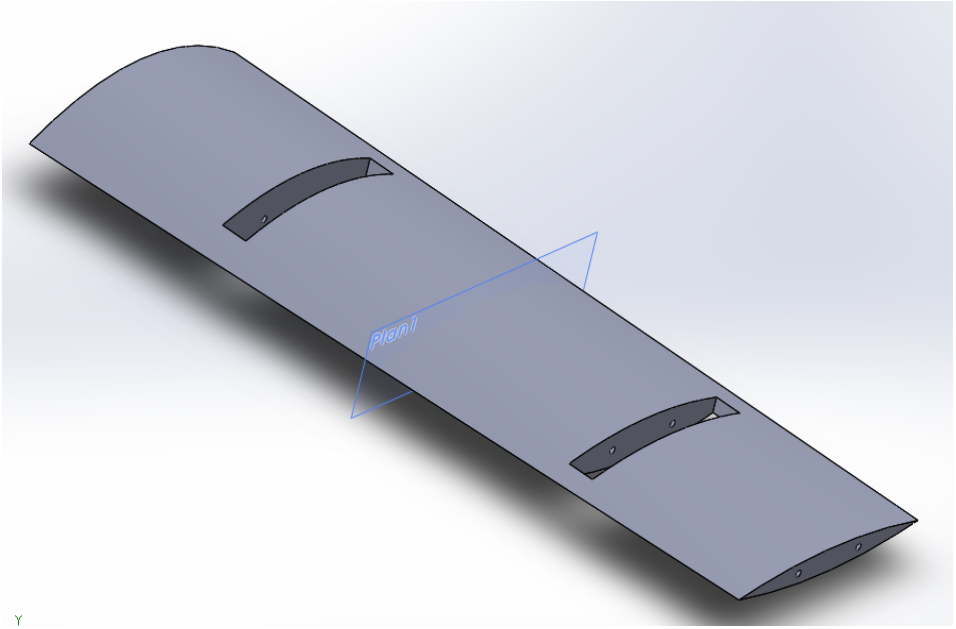


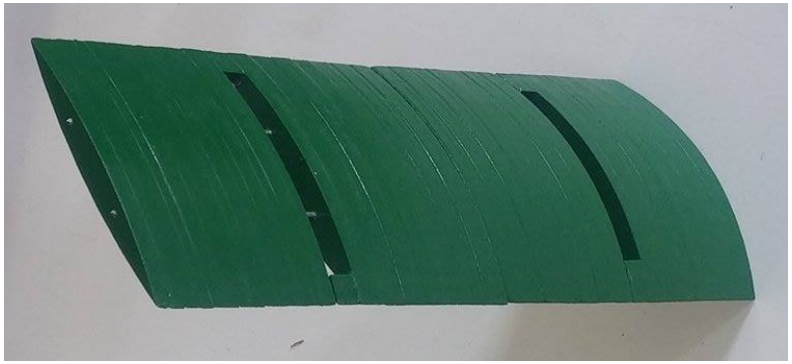
- C_t, C_p les coefficients de traînée et de portance (sans dimension), qui dépendent de l'incidence α

- Ces coefficients vont être déterminés en soufflerie

a) Création d'une pale à plus grande échelle

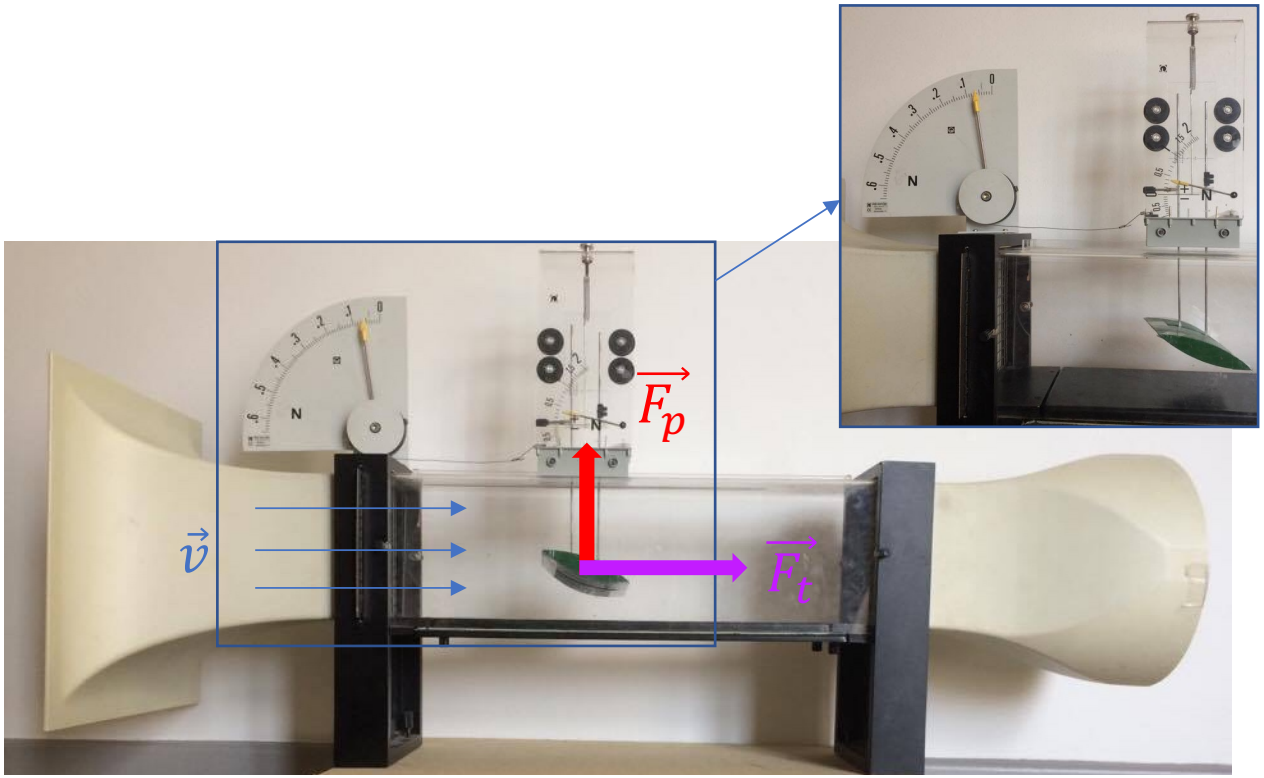
- À l'aide du logiciel Solidworks, création d'un modèle d'une pale.
- Réalisation de celle-ci par stratoconception à l'échelle 6/1.

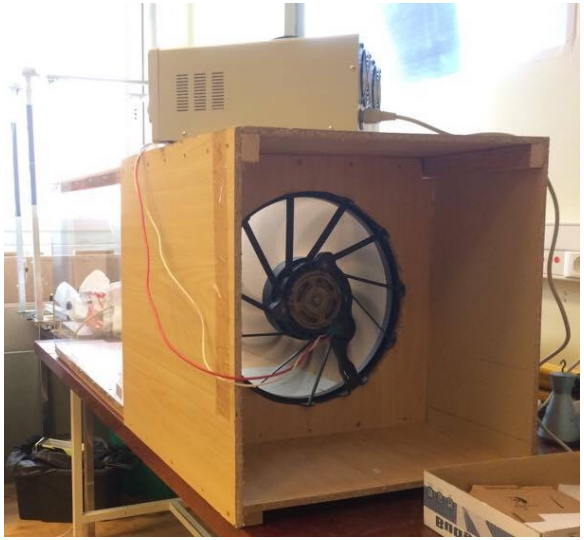




b) Détermination expérimentale des caractéristiques de la pale

- Expériences réalisées par mon binôme.
- Objectif : Établir une relation entre le carré de la vitesse du vent et les forces exercées sur la pale.

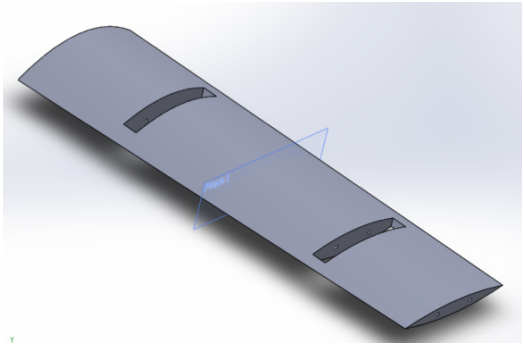


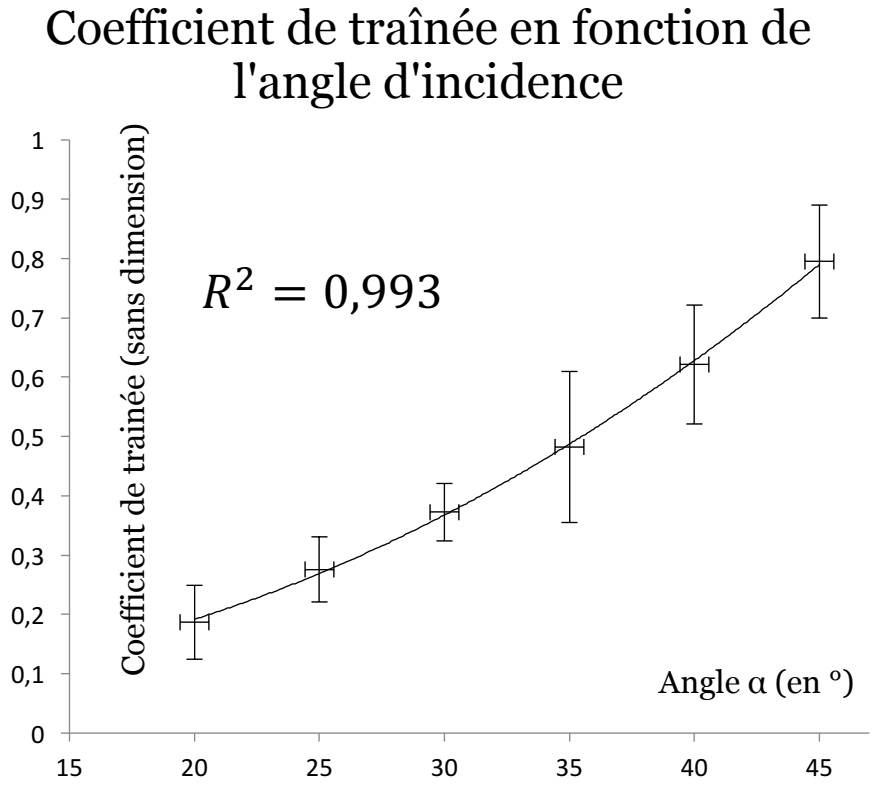


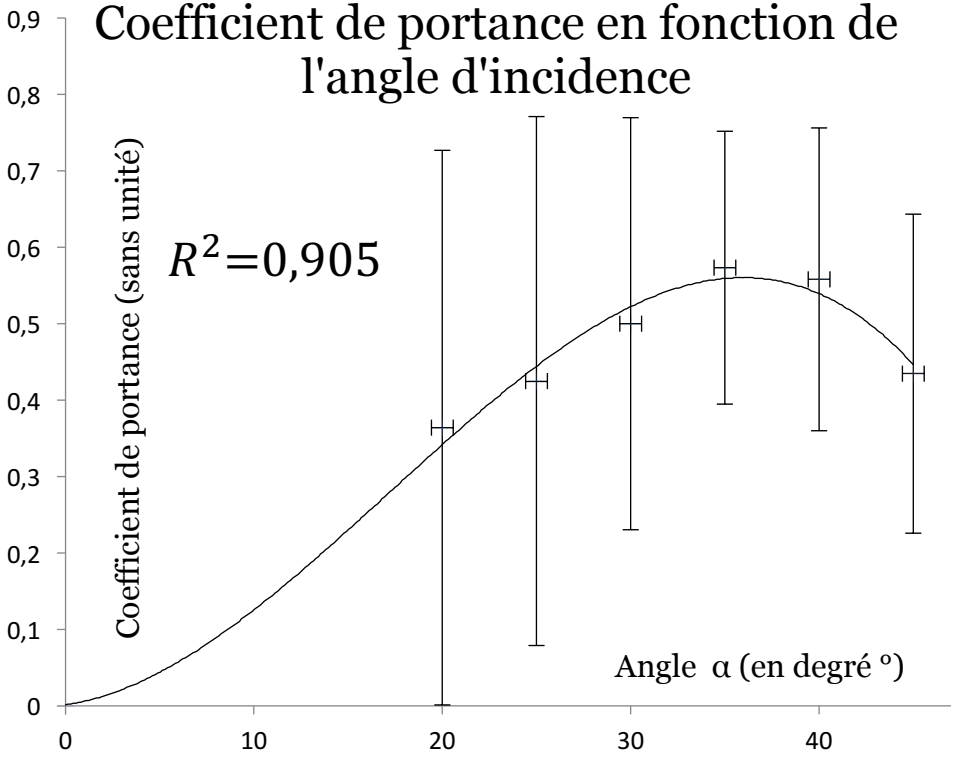
c) Modèles mathématiques retenus

$$C_t = 4{,}24 \cdot 10^{-4} \cdot \alpha^2 - 3{,}61 \cdot 10^{-3} \cdot \alpha + 9{,}46 \cdot 10^{-2}$$

$$C_p = -1{,}75 \cdot 10^{-5} \cdot \alpha^3 + 7{,}12 \cdot 10^{-4} \cdot \alpha^2 + 1{,}19 \cdot 10^{-2} \cdot \alpha$$

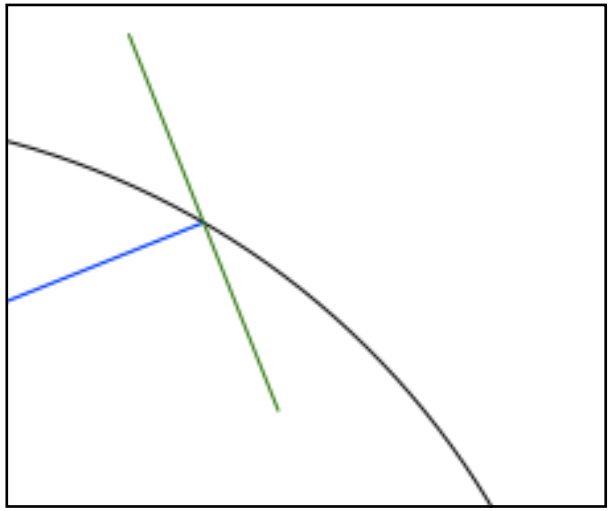


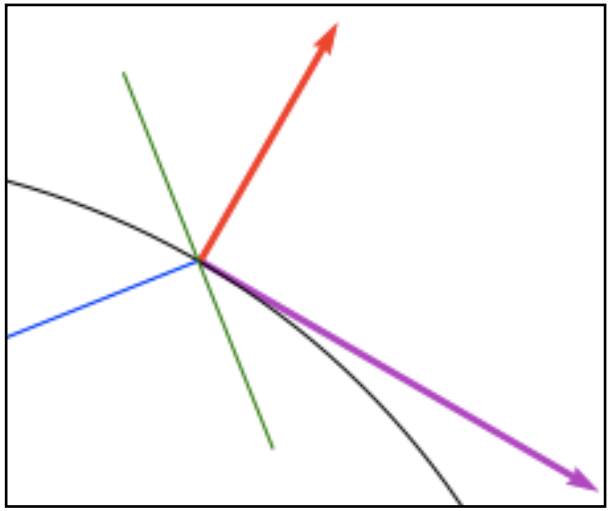


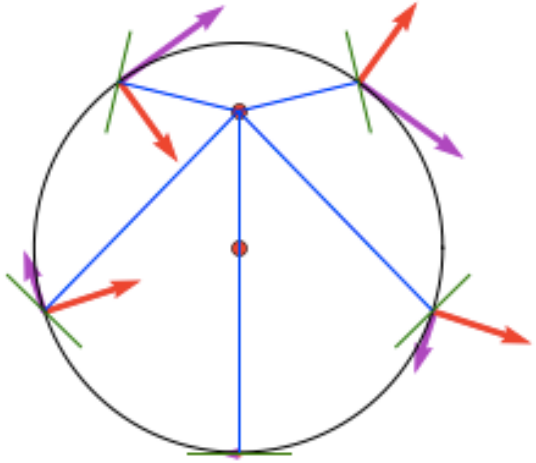


II. Etude théorique des forces s'exerçant sur les pales

- Utilisation de Python pour les calculs complexes.
- Représentation des pales à différents instants.
- Utilisation des modèles pour C_t et C_p afin de déterminer les efforts développés.

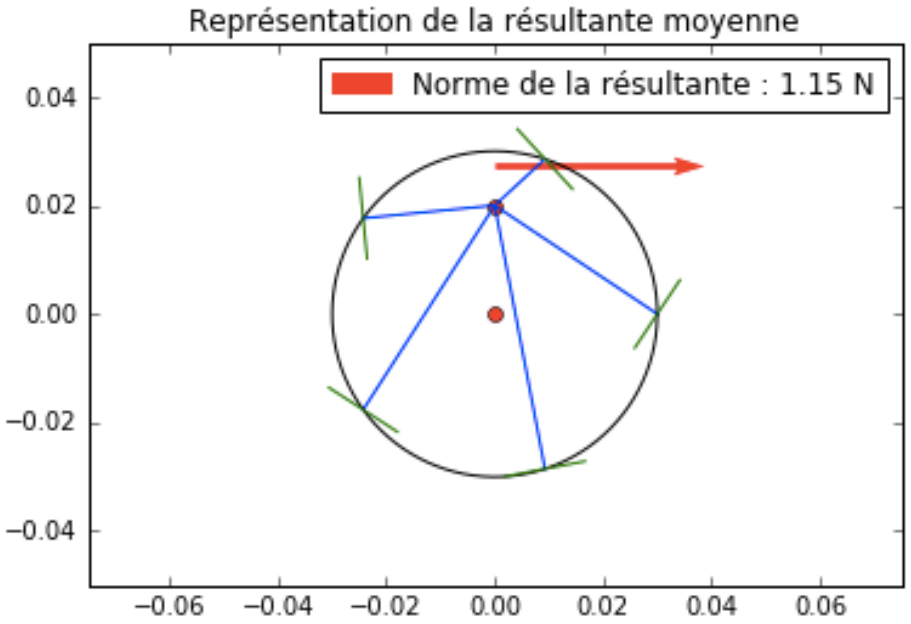


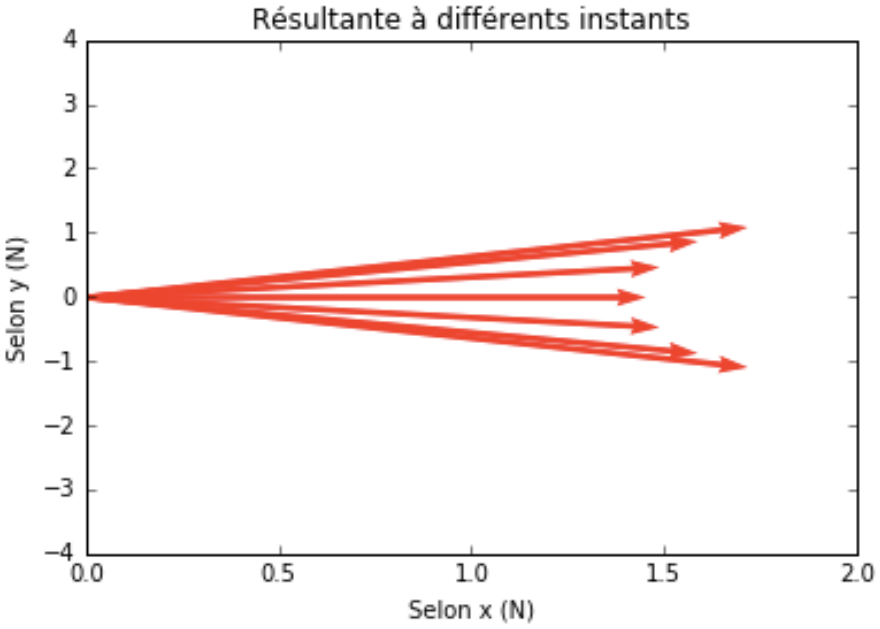




a) Résultante totale

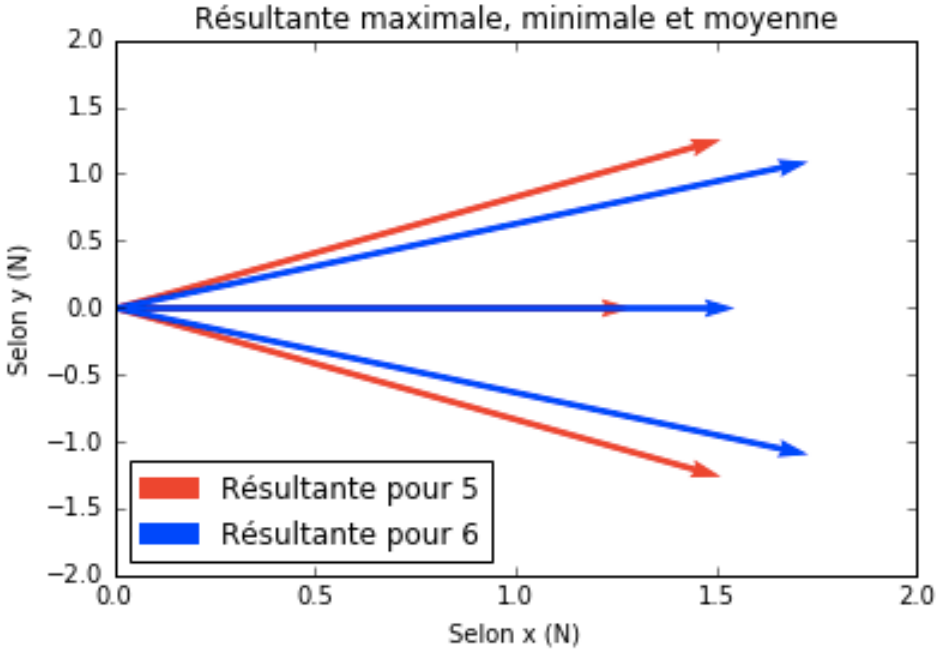
- On remarque que la résultante n'est pas constante au cours du mouvement. Elle a une période de 1/5eme de la durée de révolution.
- On peut calculer une résultante moyenne car sa période est très courte pour une utilisation classique du moteur.

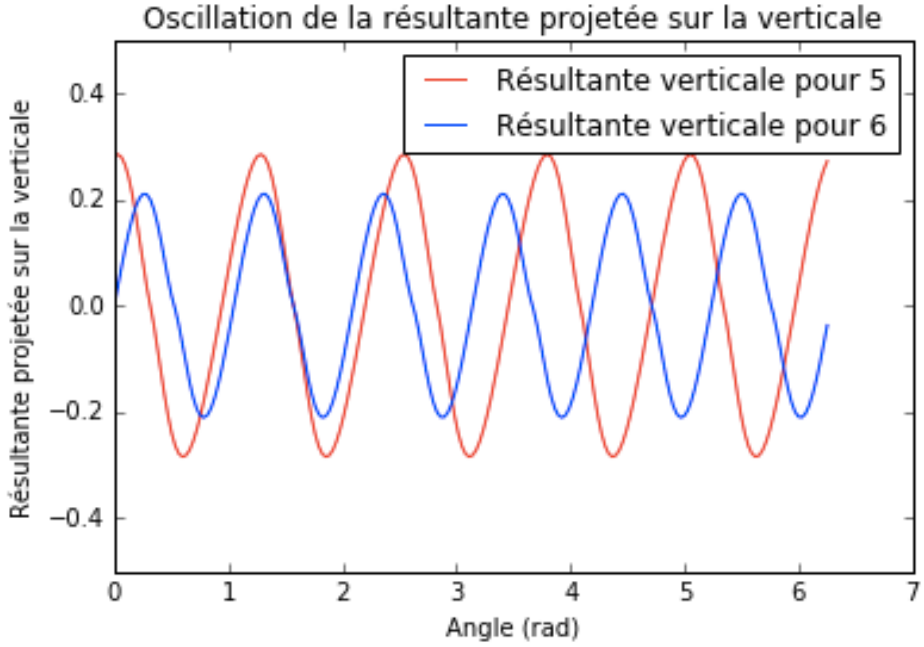




b) Influence du nombre de pales sur les variations de la résultante

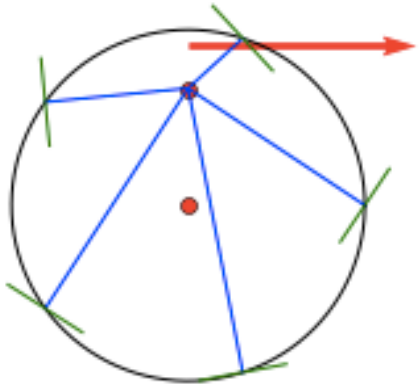
- Du point de vue de la conception, le nombre de pale a une influence sur le comportement de la résultante.
- On peut visualiser à plusieurs instants celle-ci grâce à Python, pour différents paramètres.

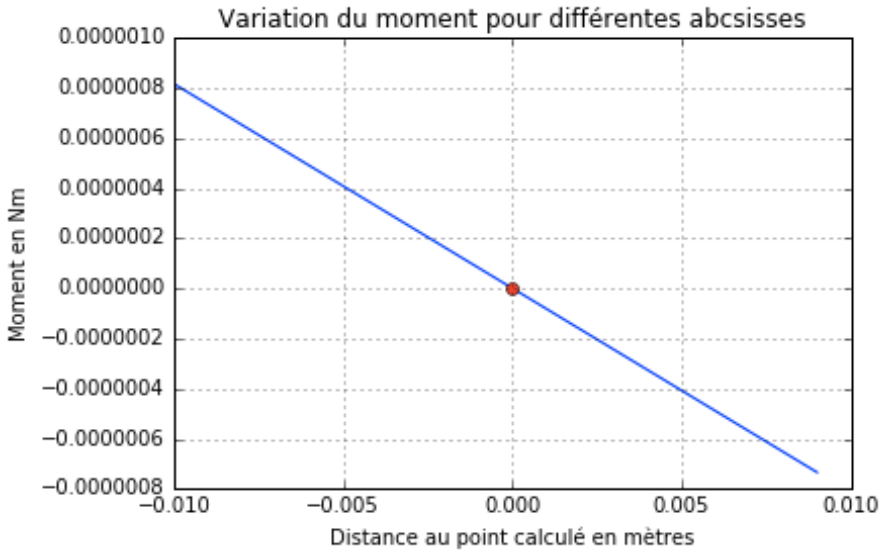


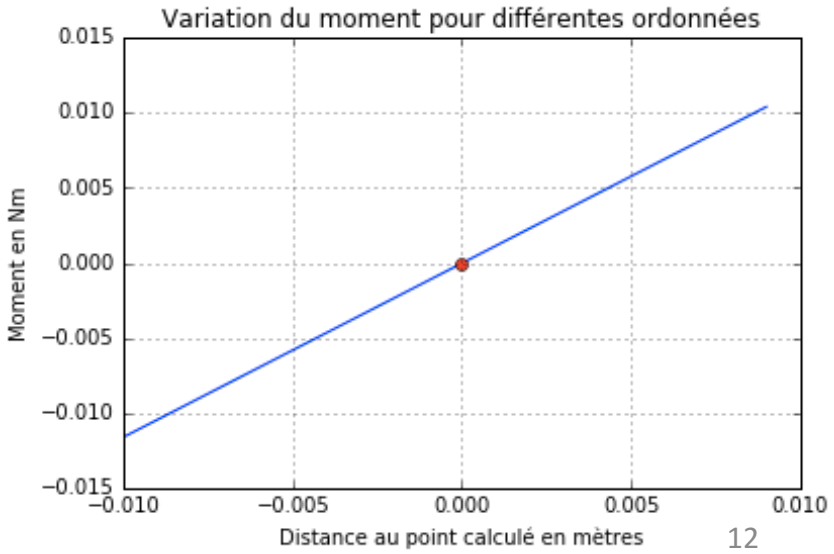


c) Point d'application de la résultante

- Il peut être important pour le pilotage de connaître le point d'application de la résultante moyenne.
- On résout ce problème numériquement : on trouve le point (0, 2.7 mm)

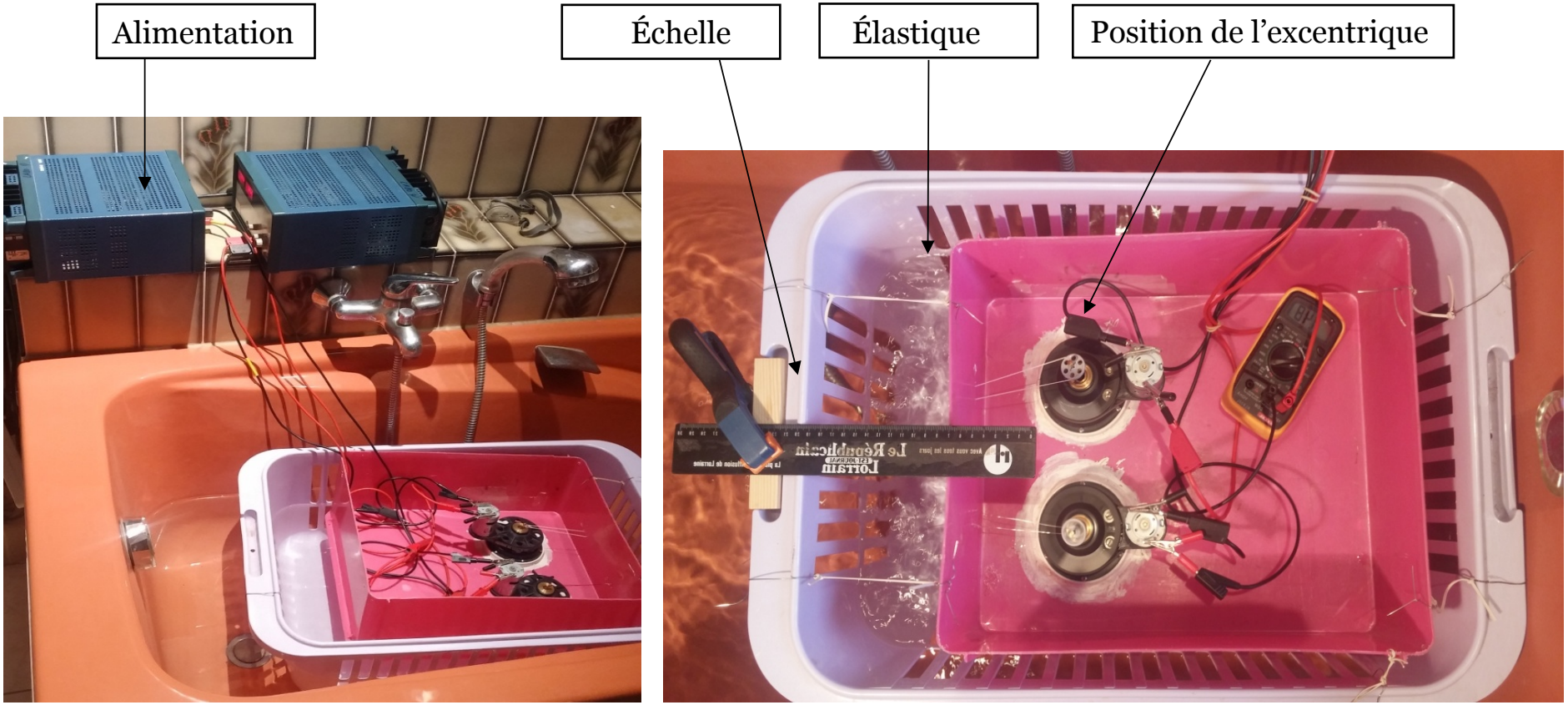






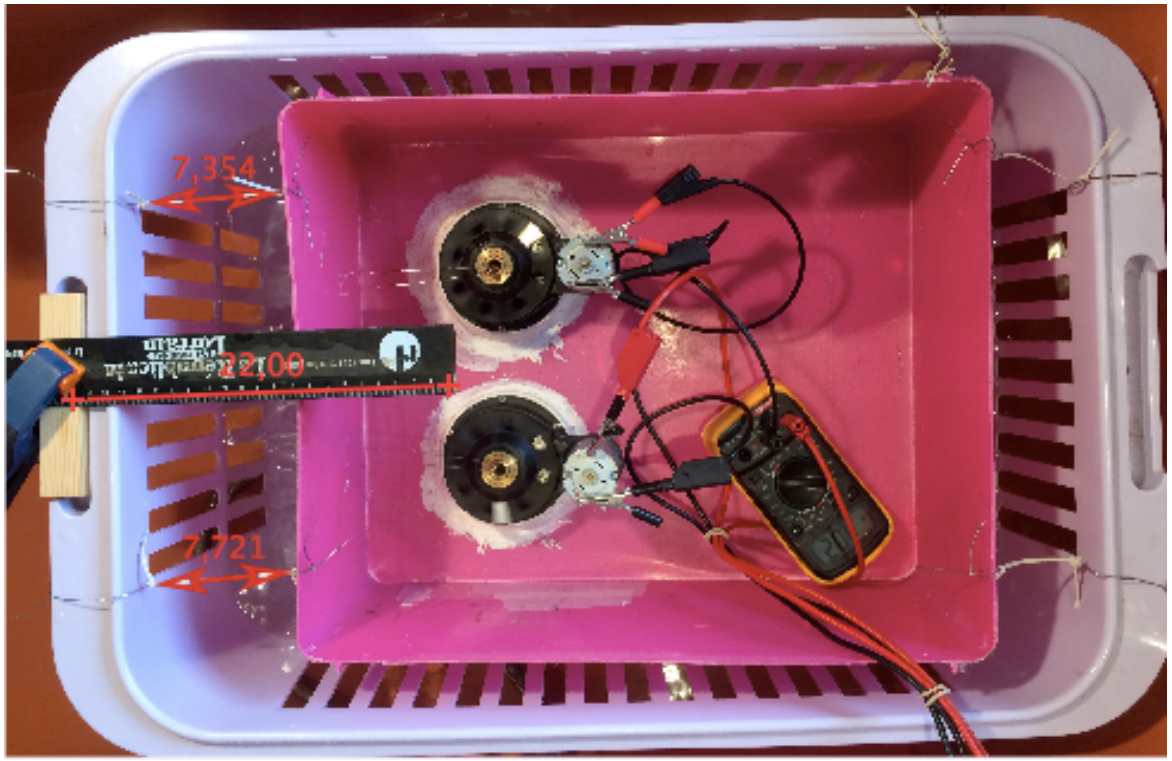
III. Vérification de notre modèle grâce à une maquette

a) Expérience

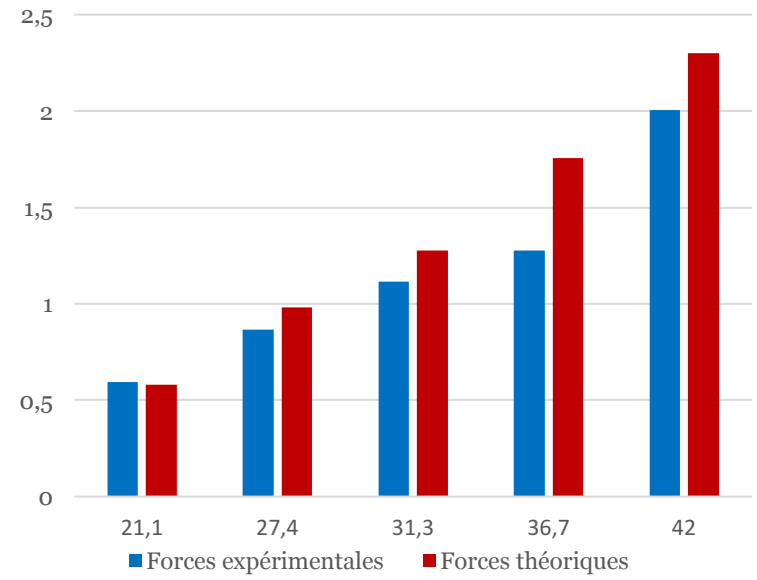


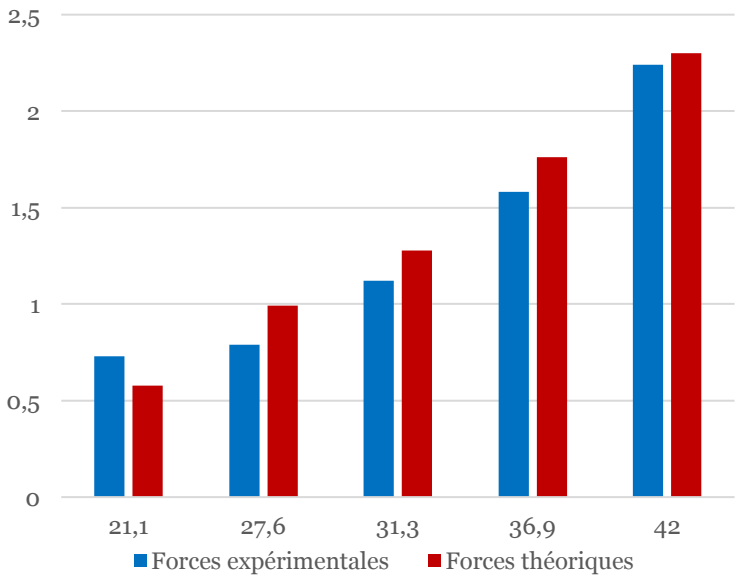
b) Exploitation des résultats

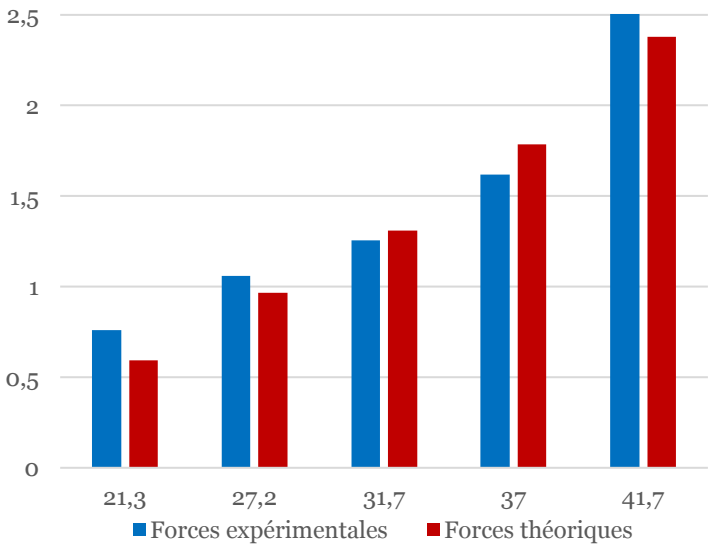
- Détermination de la raideur des élastiques par mon binôme.
- À l'aide du logiciel Tracker, mesurer l'élongation de chaque élastique et en déduire les efforts développés par le VSP

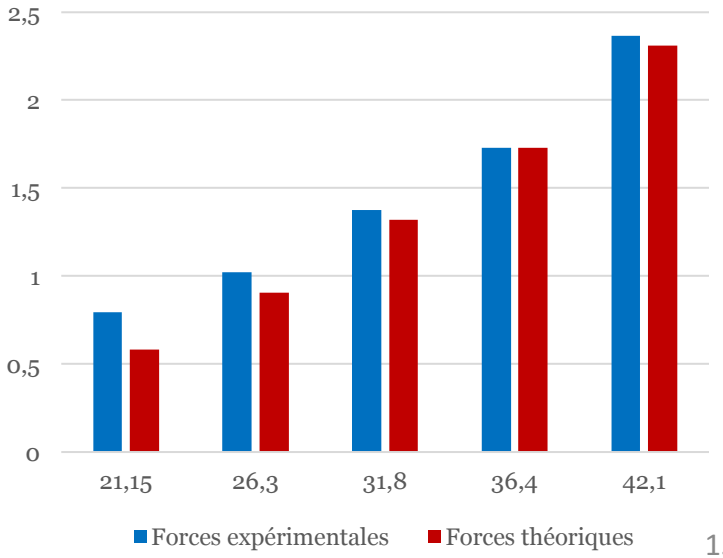


c) Comparaison des résultats









d) Conclusion et erreurs

Erreurs possibles commises :

- Fortes incertitudes sur les coefficients C_t et C_p
- Incertitude sur la direction des forces expérimentales
- Non prise en compte des turbulences et des remous.

Annexes

```python
import math
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.patches as mpatches
plt.clf()

n=5 #nombre de pales
T=np.arange(0,2*np.pi/n,0.001)
e=0.02   #excentrique
l=0.015/2 #largeur de la demi-pale
r=0.03 #rayon du cercle
rho=1000     #masse volumique
S=0.015*0.05 #surface ailaire
w=42.1 #vitesse de rotation

def x(th):          #distance excentrique-acroche de la pale
    return math.sqrt(e**2+r**2-2*e*r*math.sin(th))

def gamma(th):  #angle pale-tangente au cercle
    resu=(2*r**2-2*e*r*np.sin(th))/(2*x(th)*r)
    if resu>=1:
        resu=1
    elif resu<=-1:
        resu=-1
    return math.acos(resu)

def Cz(th):
    a=gamma(th)*180/np.pi
    return -1.75*10**(-5)*a**3+7.12*10**(-4)*a**2+1.19*10**-2*a

def Cx(th):
    a=gamma(th)*180/np.pi
    return 4.24*10**(-4)*a**2 - 3.61*10**(-3)*a + 9.46*10**(-2)

def P(th):  #extremité 1 de la pale
    if 0<=th%(2*np.pi)<=np.pi/2 or 3*np.pi/2<=th%(np.pi*2)<=2*np.pi:
        return [(r-l*np.sin(gamma(th)))*np.cos(th)+l*np.cos(gamma(th))*np.sin(th),(r-l*np.sin(gamma(th)))*np.sin(th)-l*np.cos(gamma(th))*np.cos(th)]
    else:
        return [(r+l*np.sin(gamma(th)))*np.cos(th)+l*np.cos(gamma(th))*np.sin(th),(r+l*np.sin(gamma(th)))*np.sin(th)-l*np.cos(gamma(th))*np.cos(th)]

def Q(th):      #extrémité 2 de la pale
    if 0<=th%(2*np.pi)<=np.pi/2 or 3*np.pi/2<=th%(np.pi*2)<=2*np.pi:
        return [(r+l*np.sin(gamma(th)))*np.cos(th)-l*np.cos(gamma(th))*np.sin(th),(r+l*np.sin(gamma(th)))*np.sin(th)+l*np.cos(gamma(th))*np.cos(th)]
    else:
        return [(r-l*np.sin(gamma(th)))*np.cos(th)-l*np.cos(gamma(th))*np.sin(th),(r-l*np.sin(gamma(th)))*np.sin(th)+l*np.cos(gamma(th))*np.cos(th)]

def Fx(th):
    return 0.5*rho*S*(r*w)**2*Cx(th)

def Fz(th):
    if 0<=th%(2*np.pi)<=np.pi/2 or 3*np.pi/2<=th%(np.pi*2)<=2*np.pi:
        return 0.5*rho*S*(r*w)**2*Cz(th)
    else:
        return - 0.5*rho*S*(r*w)**2*Cz(th)
```

```
C=np.arange(0,2*np.pi,0.001)    #tracé du cercle
X=[r*np.cos(c) for c in C]
Y=[r*np.sin(c) for c in C]
plt.plot(X,Y,'k')
plt.plot([0,0],[0,e],'or')  #tracé de l'excentrique

for i in range(-2,3):   #tracé des rayons et des pales
    plt.plot([0,r*np.cos(np.pi*2/5*i)],[e,np.sin(np.pi*2/5*i)*r],'b')
    plt.plot([P(np.pi*2/5*i)[0],Q(np.pi*2/5*i)[0]],[P(np.pi*2/5*i)[1],Q(np.pi*2/5*i)[1]],'g')

Res=[0,0]   #résultante
MomentEnO1=0 #moment en O de coordonnées(0,0)
MomentEnO2=0 #moment en un deuxième point (0,e) permettant de déterminer le centre de poussée
MomentEnC=0 #moment en C
L=np.arange(-0.01,0.01,0.001) #variation d'abscisse ou d'ordonnée
M=np.array([0]*len(L))

for t in T:
    for i in range(-2,3):   #tracé des vecteurs force
        th=t+np.pi*2/5*i
        FX=Fx(th)*np.sin(th)+Fz(th)*np.cos(th)
        Res[0]+=FX
        FY=Fz(th)*np.sin(th)-Fx(th)*np.cos(th)
        Res[1]+=FY
        plt.quiver(r*np.cos(th),r*np.sin(th),Fx(th)*np.sin(th),-Fx(th)*np.cos(th),angles='xy', scale_units='xy', scale=20,color='r')
        plt.quiver(r*np.cos(th),r*np.sin(th),Fz(th)*np.cos(th),Fz(th)*np.sin(th),angles='xy', scale_units='xy', scale=20,color='r')
        MomentEnO1+=MomentEnO(FX,FY,r*np.cos(th),r*np.sin(th),0,0)
        MomentEnO2+=MomentEnO(FX,FY,r*np.cos(th),r*np.sin(th),0,e)
        MomentEnC+=MomentEnO(FX,FY,r*np.cos(th),r*np.sin(th),0,0.0275080276881)
        M=M+np.array([MomentEnO(FX,FY,r*np.cos(th),r*np.sin(th),a,0.0275080276881) for a in L])  #graphes centre de poussée

Res[0]=Res[0]/len(T)    #moyenne de la résultante totale
Res[1]=Res[1]/len(T)
M=M/len(T)

def courbeMoment():
    plt.plot(L,M) #graphes centre de poussée
    plt.title("Variation du moment pour différentes abcsisses")
    plt.xlabel("Distance au point calculé")
    plt.ylabel("Moment")
    plt.grid(True)
    plt.show()

print('Norme résultante :',(Res[0]**2+Res[1]**2)**0.5)
print('Norme moment en C:',MomentEnC/len(T))
print('Y centre de poussée',-MomentEnO1/len(T)/Res[0])

plt.quiver(0,0.0275080276881,Res[0],Res[1],angles='xy', scale_units='xy', scale=30,color='r')    #tracer résultante
red_patch = mpatches.Patch(color='red', label='Norme de la résultante : '+str(round((Res[0]**2+Res[1]**2)**0.5,2))+' N')
plt.legend(handles=[red_patch])
plt.axis('Equal')
plt.axis([-0.05,0.05,-0.05,0.05])
plt.title("Représentation de la résultante moyenne")
plt.show()
```

```
74
75 FY=[[] for i in range(n)]   #liste de la résultante selon y pour chaque pale
76 FX=[[] for i in range(n)]
77 Ry_n=[]
78 Theta=np.arange(0,np.pi*2,np.pi*2/resolution)  #angle theta qui varie
79
80 for i in range(n):
81     for th in Theta:    #représentation de la résultante à plusieurs instants
82         FY[i]+=[Fz(np.pi*2/n*i+th)*np.sin(np.pi*2/n*i+th)-Fx(np.pi*2/n*i+th)*np.cos(np.pi*2/n*i+th)]
83         FX[i]+=[Fx(np.pi*2/n*i+th)*np.sin(np.pi*2/n*i+th)+Fz(np.pi*2/n*i+th)*np.cos(np.pi*2/n*i+th)]
84
85 for th in range(len(FX[0])):
86     resu=0
87     for i in range(len(FX)):
88         resu+=FY[i][th]
89     Ry_n+=[resu]
90
91 FY=[[] for i in range(n+1)]   #liste de la résultante selon y pour chaque pale pour n+1
92 FX=[[] for i in range(n+1)]
93 Theta=np.arange(0,np.pi*2,np.pi*2/resolution)
94 Ry_np1=[]
95
96 for i in range(n+1):
97     for th in Theta:    #représentation de la résultante à plusieurs instants pour n+1 pales
98         FY[i]+=[Fz(np.pi*2/(n+1)*i+th)*np.sin(np.pi*2/(n+1)*i+th)-Fx(np.pi*2/(n+1)*i+th)*np.cos(np.pi*2/(n+1)*i+th)]
99         FX[i]+=[Fx(np.pi*2/(n+1)*i+th)*np.sin(np.pi*2/(n+1)*i+th)+Fz(np.pi*2/(n+1)*i+th)*np.cos(np.pi*2/(n+1)*i+th)]
100
101 for th in range(len(FY[0])):        #idem pour n+1 pales
102     resu=0
103     for i in range(len(FX)):
104         resu+=FY[i][th]
105     Ry_np1+=[resu]
106
107 plt.figure(1)
108 plt.title("Oscillation de la résultante projetée sur la verticale")
109 plt.xlabel("Angle (rad)")
110 plt.ylabel("Résultante projetée sur la verticale")
111 plt.plot(Theta,Ry_n,color='r',label='Résultante verticale pour '+str(n))
112 plt.axis([0,7,-0.5,0.5])
113 plt.plot(Theta,Ry_np1,color='b',label='Résultante verticale pour '+str(n+1))
114 plt.axis([0,7,-0.5,0.5])
115 plt.legend(loc='best')
116 plt.show()
```

```
73 for ti in np.arange(-np.pi*2/n*3/10,np.pi*2/n*4/10,np.pi*2/n/10):    #représentation de la résultante à plusieurs instants
74     FX,FY=0,0
75     for i in range(n):
76         if ti!=np.pi*2/n*5/10:
77             th=ti+np.pi*2/n*i+t
78             FX+=Fx(th)*np.sin(th)+Fz(th)*np.cos(th)
79             FY+=Fz(th)*np.sin(th)-Fx(th)*np.cos(th)
80     plt.quiver(0,0,FX,FY,angles='xy', scale_units='xy', scale=1,color='b')
81
82 plt.axis([0,2,-4,4])
83 plt.title("Résultante à différents instants")
84 plt.xlabel("Selon x (N)")
85 plt.ylabel("Selon y (N)")
86 plt.show()
```

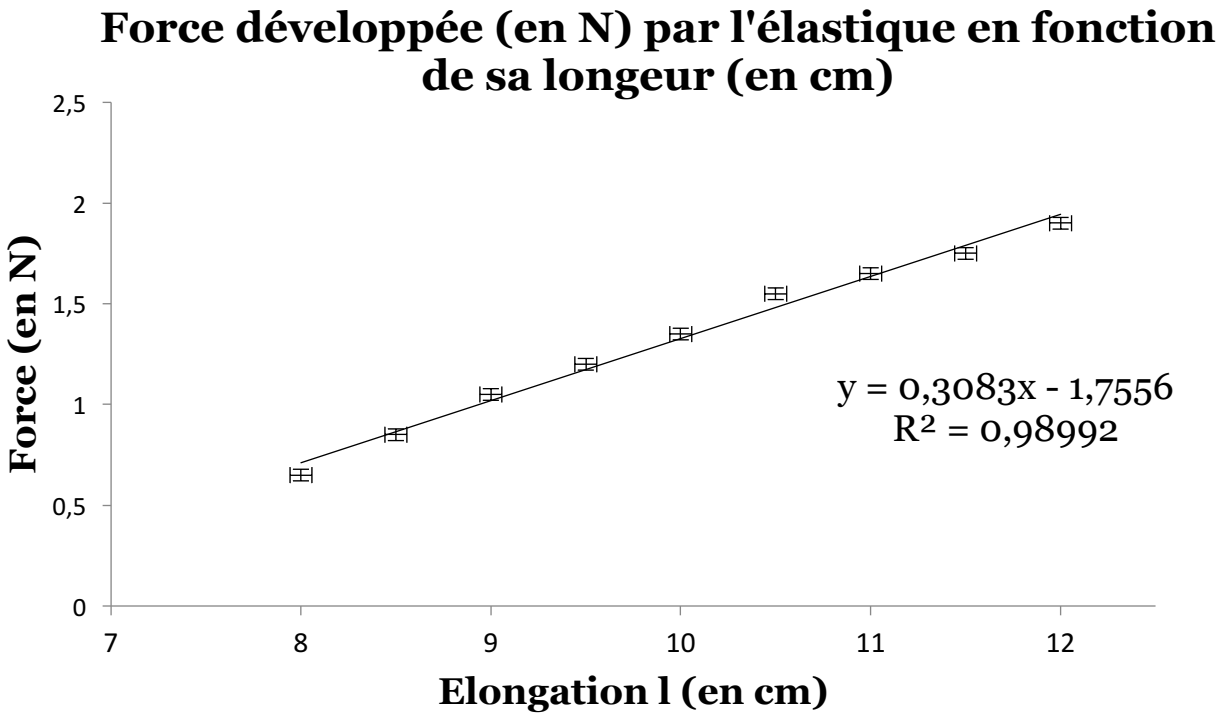
```
Fmax=[0,0]
Fmin=[0,0]
F0=[0,0]

for ti in np.arange(0,np.pi*2/n,np.pi*2/n/1000):    #représentation de la résultante à plusieurs instants
    (Fy,FX)=(0,0)
    for i in range(n):
        th=ti+np.pi*2/n*i
        Fy+=Fz(th)*np.sin(th) - Fx(th)*np.cos(th)
        FX+=Fz(th)*np.cos(th) + Fx(th)*np.sin(th)
    if Fy>=Fmax[1]:
        Fmax[1]=Fy
        Fmax[0]=FX
    if Fy<=Fmin[1]:
        Fmin[1]=Fy
        Fmin[0]=FX
    F0[0]+=FX
    F0[1]+=Fy

Longueur=len(np.arange(0,np.pi*2/n,np.pi*2/n/1000))
plt.quiver(0,0,Fmin[0],Fmin[1],angles='xy', scale_units='xy', scale=1,color='r')
plt.quiver(0,0,Fmax[0],Fmax[1],angles='xy', scale_units='xy', scale=1,color='r')
plt.quiver(0,0,F0[0]/Longueur,F0[1]/Longueur,angles='xy', scale_units='xy', scale=1,color='r')
Fmax2=[0,0]
Fmin2=[0,0]
F02=[0,0]

for ti in np.arange(0,np.pi*2/(n+1),np.pi*2/(n+1)/1000):    #représentation de la résultante à plusieurs instants
    (Fy,FX)=(0,0)
    for i in range(n+1):
        th=ti+np.pi*2/(n+1)*i
        Fy+=Fz(th)*np.sin(th) - Fx(th)*np.cos(th)
        FX+=Fz(th)*np.cos(th) + Fx(th)*np.sin(th)
    if Fy>=Fmax2[1]:
        Fmax2[1]=Fy
        Fmax2[0]=FX
    if Fy<=Fmin2[1]:
        Fmin2[1]=Fy
        Fmin2[0]=FX
    F02[0]+=FX
    F02[1]+=Fy

Longueur=len(np.arange(0,np.pi*2/(n+1),np.pi*2/(n+1)/1000))
plt.quiver(0,0,Fmin2[0],Fmin2[1],angles='xy', scale_units='xy', scale=1,color='b')
plt.quiver(0,0,Fmax2[0],Fmax2[1],angles='xy', scale_units='xy', scale=1,color='b')
plt.quiver(0,0,F02[0]/Longueur,F02[1]/Longueur,angles='xy', scale_units='xy', scale=1,color='b')
red_patch = mpatches.Patch(color='red', label='Résultante pour '+str(n))
blue_patch = mpatches.Patch(color='blue', label='Résultante pour '+str(n+1))
plt.legend(handles=[red_patch]+[blue_patch],loc=3)
plt.title("Résultante maximale, minimale et moyenne")
plt.xlabel("Selon x (N)")
plt.ylabel("Selon y (N)")
plt.axis([0,2,-2,2])
plt.show()
```


Force développée (en N) par l'élastique en fonction de sa longueur (en cm)
Force (en N)
2,5
2
1,5
1
0,5
0
7
8
9
10
11
12
Elongation l (en cm)
y = 0,3083x - 1,7556
R² = 0,98992