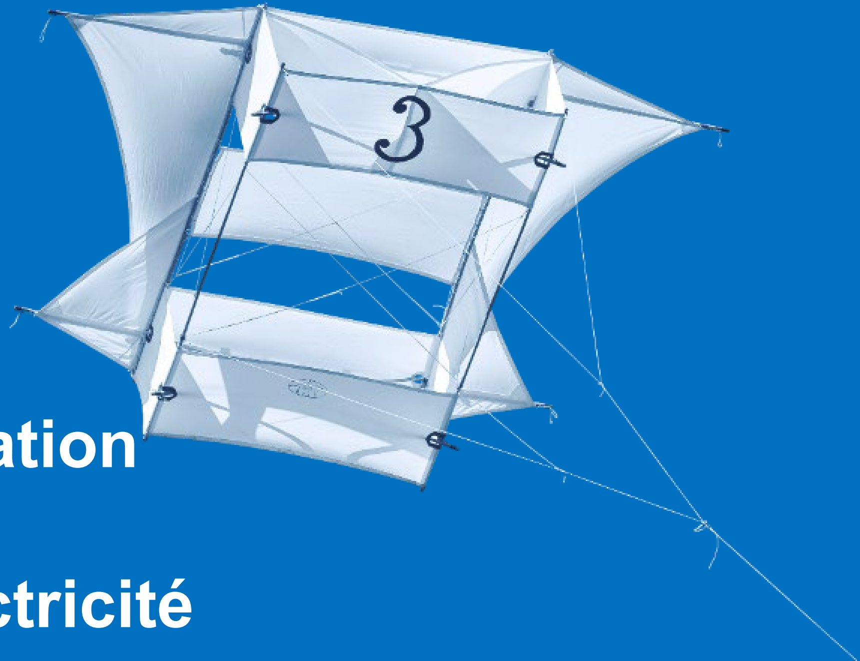
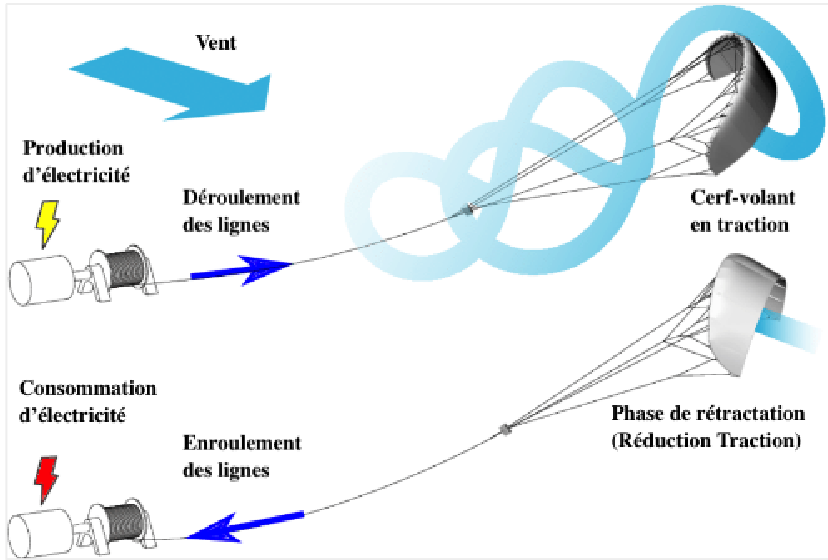


Etude & modélisation d'un système de production d'électricité via la traction d'un cerf-volant

Louis Libat - N° d'inscription : 2415

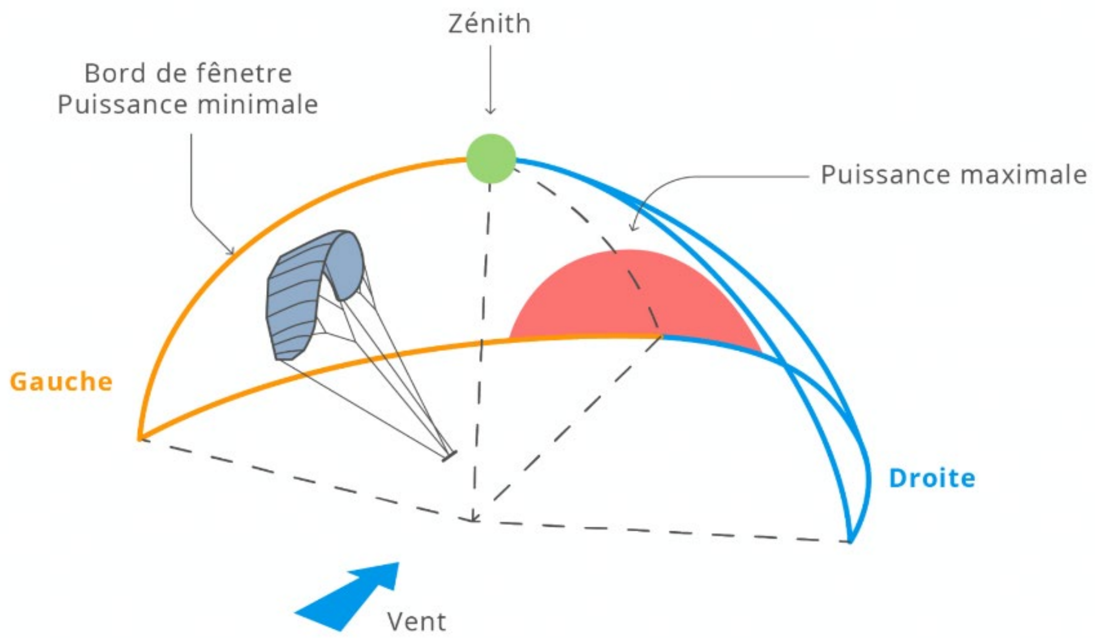
Introduction

- Vol de l’aile sur une orbite en forme de huit allongé
- Lignes enroulées autour d’un tambour relié au générateur
- Générateur électrique au sol
- Phase de rétractation : enroulement autour du tambour (moteur)
- Alternance de cycle de traction et de rétractation
- Nécessité de gérer la dynamique
  de l’aile (angle d’attaque)

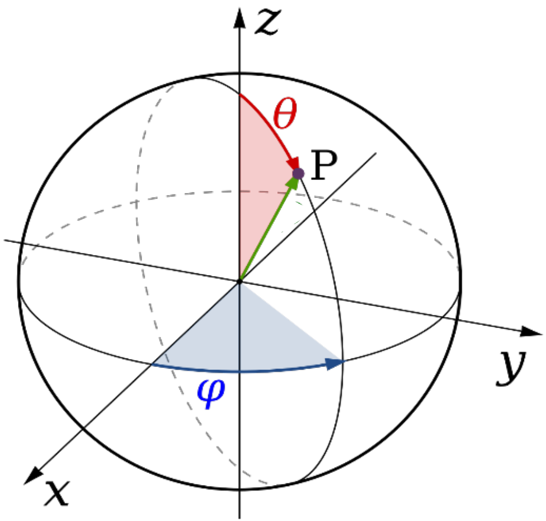


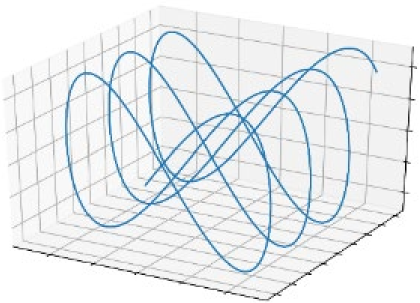
Adaptive Flight Path Control of Airborne Wind Energy Systems
Scientific Figure on ResearchGate – Roland Schmehl

Mise en place du dispositif aérien

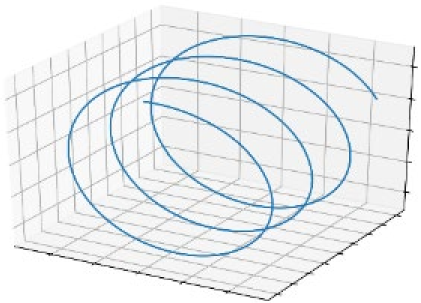


Paramétrage du vol





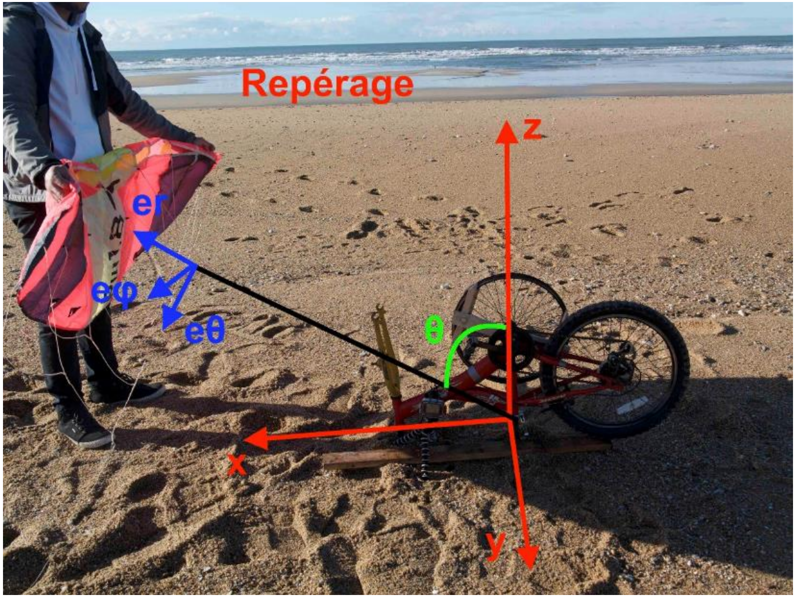
$$\begin{cases} \theta(t) = & 60 + 5 \cdot \sin(2t) \\ \phi(t) = & 5 \cdot \sin(t) \\ r(t) = & 4 \cdot t \end{cases}$$

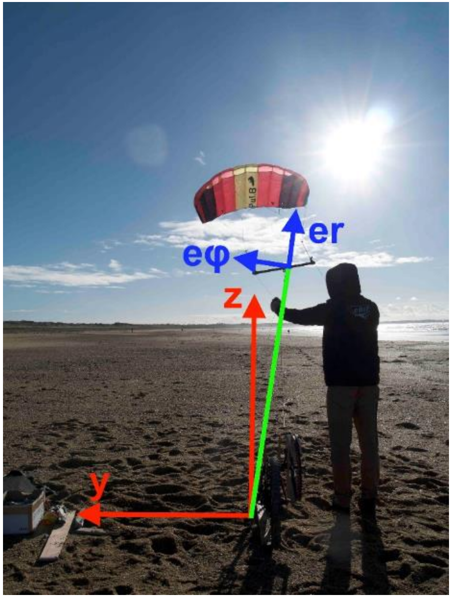


$$\begin{cases} \theta(t) = & 5 \cdot \cos(t) \\ \phi(t) = & 5 \cdot \sin(t) \\ r(t) = & 4 \cdot t \end{cases}$$

Données : S = 1,26 m² / m = 0,18 kg / Cp = 1,2 / V = 8 $m.s^{-1}$ / f = 8

Paramétrage du vol

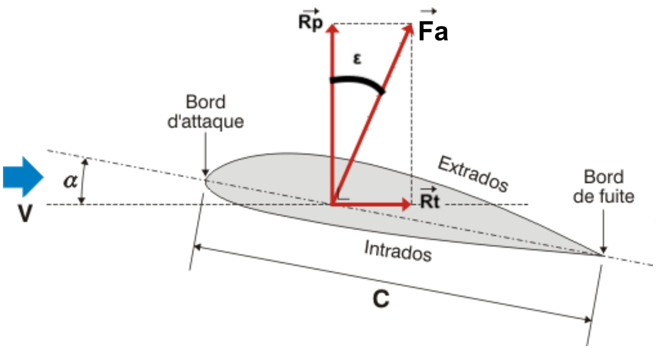


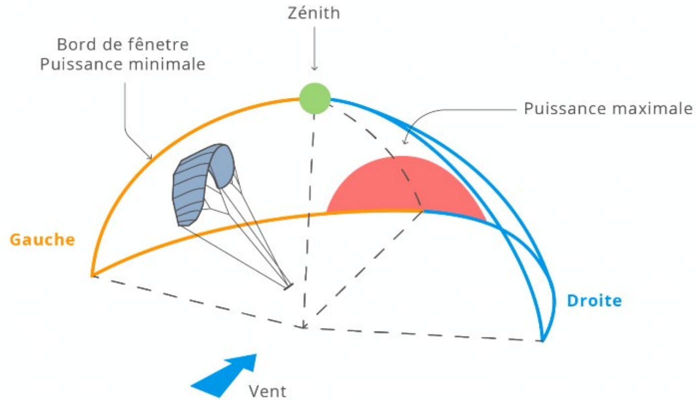


Zone de puissance

$$\tan(\epsilon) = \frac{C_t}{C_p} = \frac{1}{f}$$

$$V_a = V\frac{\cos(\phi)\sin(\theta)}{\sin(\epsilon)}$$





V_a croît avec la finesse. Force de traction plus élevée.
Intérêt de commander le vol en fonction de f.
Cerf-volant au zénith : V_a décroit - Utilisation du kite loin du zénith vers le sol.

Raccordement du cerf-volant

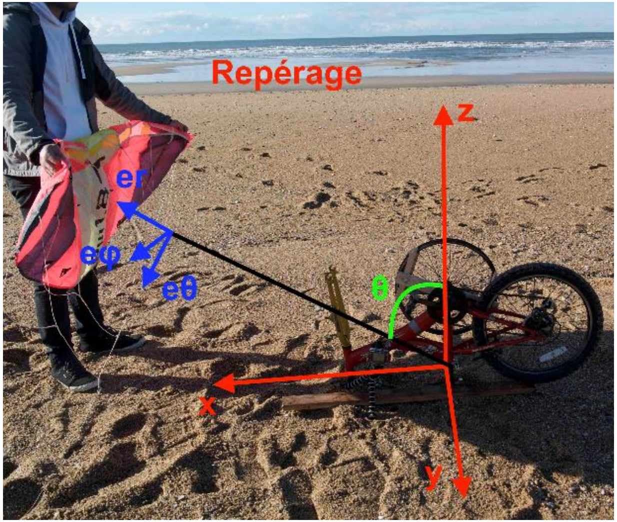
Raccordement du cerf - volant

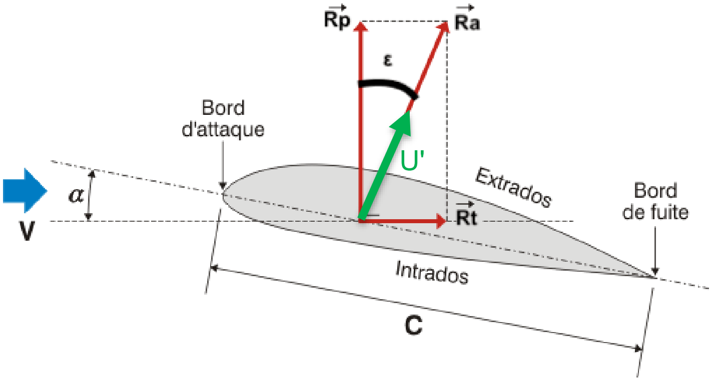
- Phase de décollage
- Répartition linéique de la force de traction
- Pilotage manuel du cerf-volant

Etude de θ

$$\vec{V_\infty} = V_\infty \vec{e_x} \quad \vec{V_a} = \vec{V_\infty} - \vec{V} \quad C_p = 2\pi \sin(\alpha)$$

$$\vec{F_a} = \rho_0 ab\pi \sin(\alpha) V_a^2 \vec{u'}$$





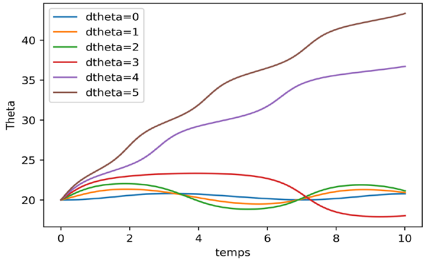
Théorème du moment cinétique en O à l'aile dans le référentiel Terrestre :

$$\ddot{\theta} = \frac{ab\pi\rho_0 \sin(\gamma)}{mR}\Big(V_\infty \cos(\theta+\gamma) - R\dot{\theta}\sin(\gamma)\Big) V_a\Big(\theta, \ddot{\theta}\Big) + \frac{g}{R}\cos(\theta)$$

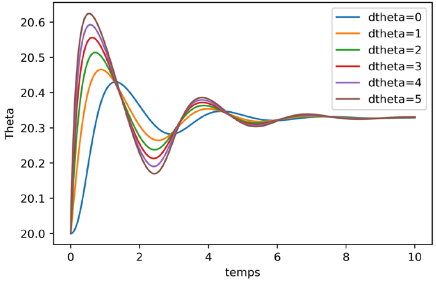
$$V_a\Big(\theta, \ddot{\theta}\Big) = \sqrt[2]{V_\infty^2 + R^2\dot{\theta}^2 + 2V_\infty R\dot{\theta}\sin(\theta)}$$

Evolution temporelle

$\Theta=f(t)$: Condition initiale θ_0 = 20° / θ'_0 variant. Fonction de l'angle γ



γ = 0.01°
Divergence de l'aile



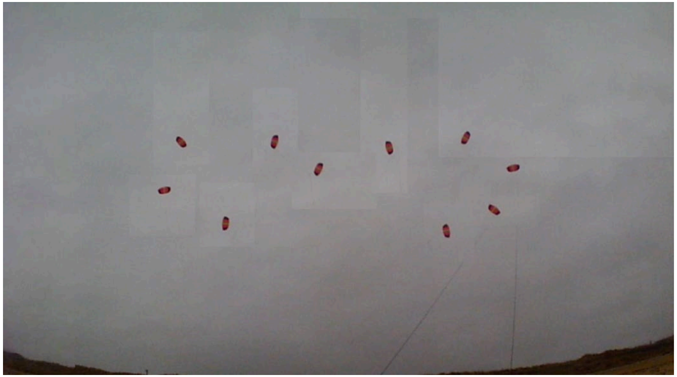
γ = 0.1°
L'aile oscille et tend
vers un équilibre stable

- Semblant de trajectoire similaire à des oscillations amorties
- Ne peut exister dans la pratique

Mettre en place un dispositif de contrôle de la trajectoire

Expérience n°1 (1/3)

Mesure Tension (N)

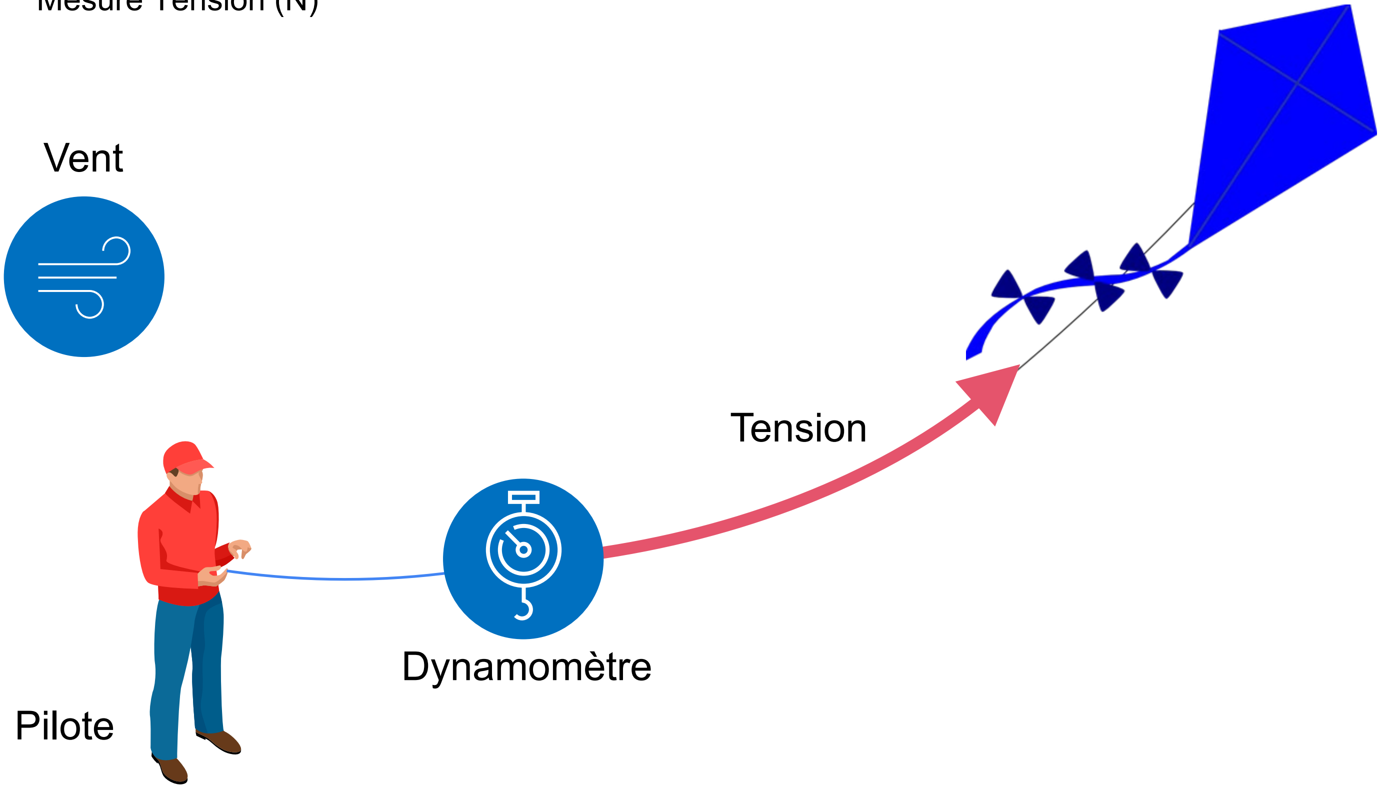


Vol de Type Huit

Vol de Type Cercle

Expérience n°1 (2/3)

Mesure Tension (N)

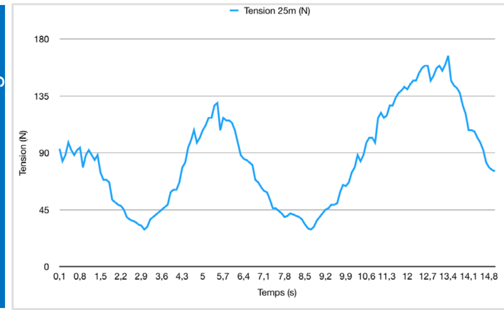


Expérience n°1 (3/3)

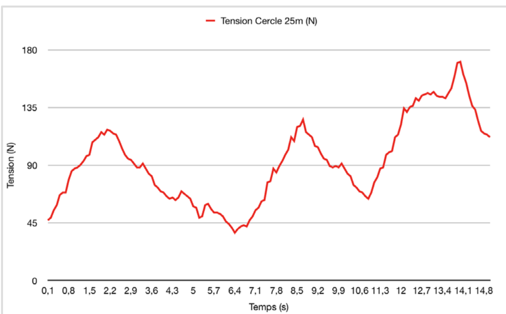
Mesure Tension (N)

Vol Huit - 50m ligne

Vol Huit - 25m Ligne



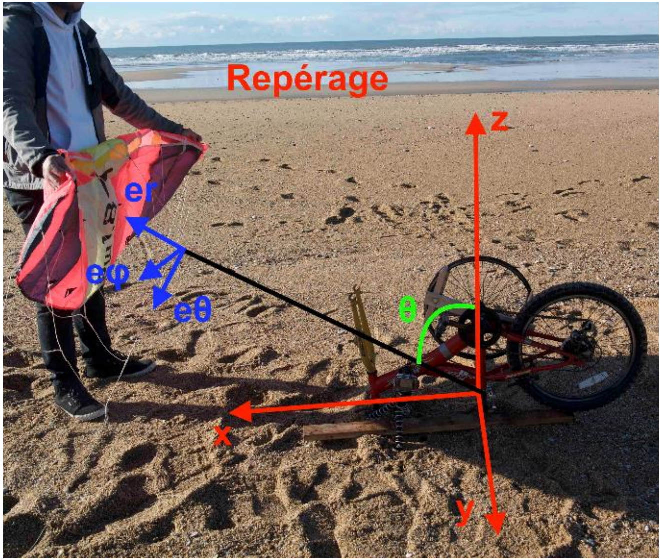
Vol Cercle - 25m Ligne



Incertitudes :

- Lecture Dynamomètre : ulect = 0.05
- Variation de theta pas totalement identique
- Incertitude : 5 % de la valeur
- Ecart Modèle-Expérience : 7 %

Déroulement du câble (1/2)

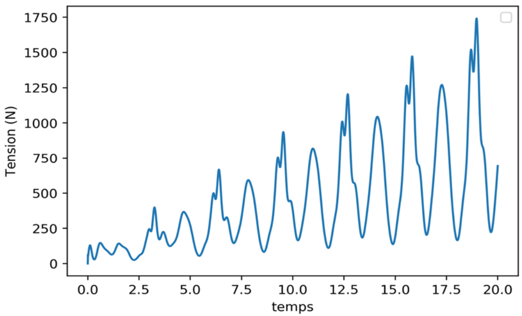


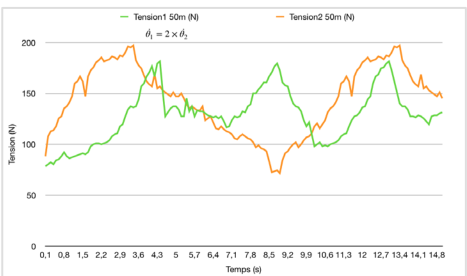
Théorème de la résultante dynamique à l'aile dans le référentiel terrestre :

$$\vec{T} = T\vec{e_r} \quad \vec{P} = m\vec{g} \quad \vec{F_a} = \frac{1}{2}\rho_0 S V_a C_p \vec{V_a}$$

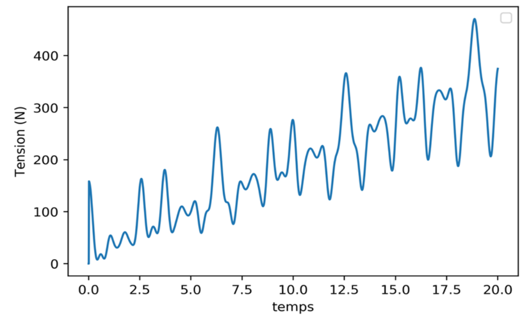
$$\begin{cases} m\left(\ddot{r} - r\dot{\theta}^2 - r\sin^2(\theta)\dot{\phi}^2\right) & = F_r \\ m\left(r\ddot{\theta} + 2\dot{r}\dot{\theta} - r\sin(\theta)\cos(\theta)\dot{\phi}^2\right) & = F_\theta \\ m\left(r\dot{\phi}\sin(\theta) + 2r\dot{\theta}\dot{\phi}\cos(\theta) + 2\dot{r}\dot{\phi}\sin(\theta)\right) & = F_\phi \end{cases}$$

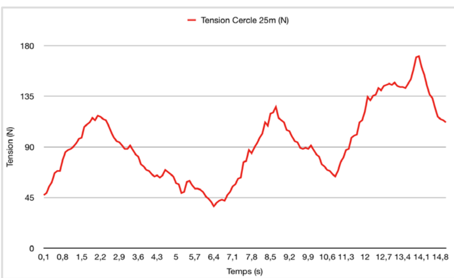
Déroulement du câble (2/2)





Vol Huit - V_∞ = 10m/s - r'= 4m/s





Vol Cercle - V_∞ = 10m/s - r'= 4m/s

Phase de rétractation

Aile au zénith : soumis à son poids et force aérodynamique

$$\begin{cases} \dot{r} < & 0 \\ \ddot{r} = & 0 \\ \dot{\theta} = & 0 \\ \dot{\phi} = & 0 \end{cases}$$

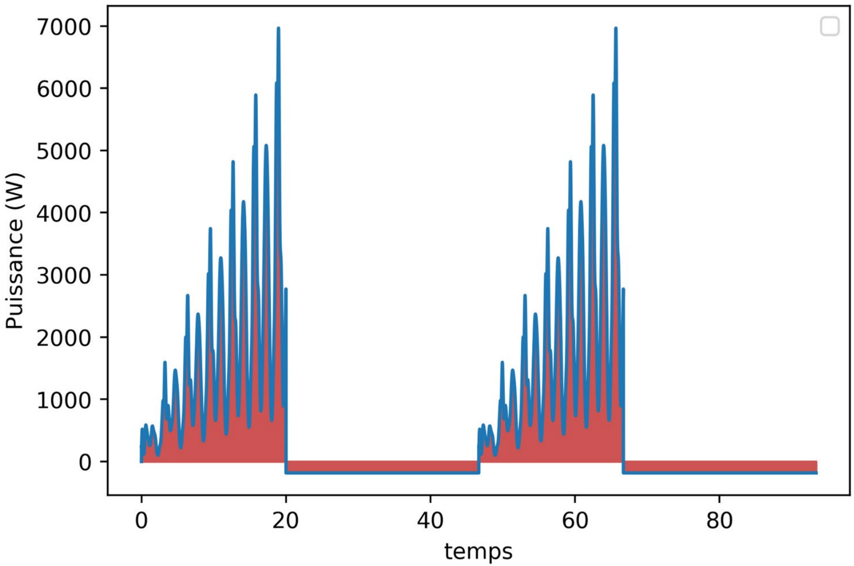
$$P = \left(-mg + \frac{1}{2}\rho_0 S C_p \cdot \dot{r}^2\right) \cdot \dot{r}$$

Duty Factor : Ratio durée de déroulement sur durée totale du cycle

$$D = \frac{\Delta t_o}{\Delta t_i + \Delta t_o} \quad \Delta t_i = \frac{\Delta l}{v_i},\ \Delta t_o = \frac{\Delta l}{v_o} \quad \eta = \frac{E_{m,o} - E_{m,i}}{E_{m,o}}$$

- Vent Faible : Vitesse déroulement faible - Meilleur Duty Factor
- Vent Fort : Vitesse déroulement Faible (Fort Couple) + Vitesse d'enroulement élevée
- Vitesse de Rétractation trop élevée peut faire chuter le rendement

Cycle Complet



Vol en Huit :
V_∞ = 10m/s
r'= 4m/s
r'_{retra} = - 3m/s

Estimer la vitesse de rétractation optimale afin de rembobiner le câble en dépensant le moins d'énergie

Construction du treuil

Ligne reliée
au cerf-volant

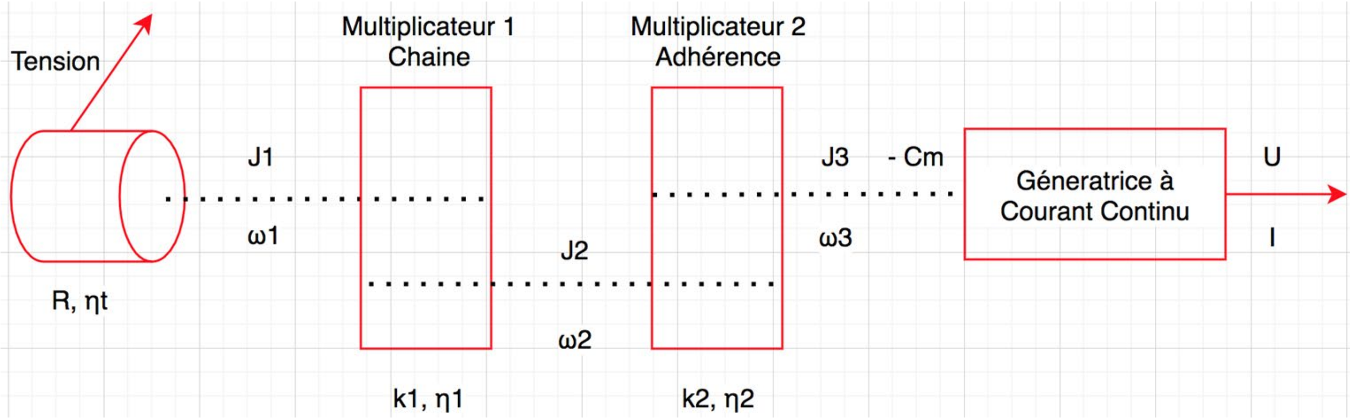
Tambour

Transmission
par chaine

Volant d'inertie

Contact par adhérence
volant et moteur

Transmission de Puissance



$$E_c(\Sigma)_{/Rg} = \frac{1}{2}\omega_3^2\left(J_3 + \frac{J_2}{k_2^2} + \frac{J_1}{(k_1 k_2)^2}\right)$$

$$P\left(\overline{\Sigma} \rightarrow \Sigma\right)_{/Rg} = F_{Tract}V_{deroul} - \Gamma_{em}\omega_3$$

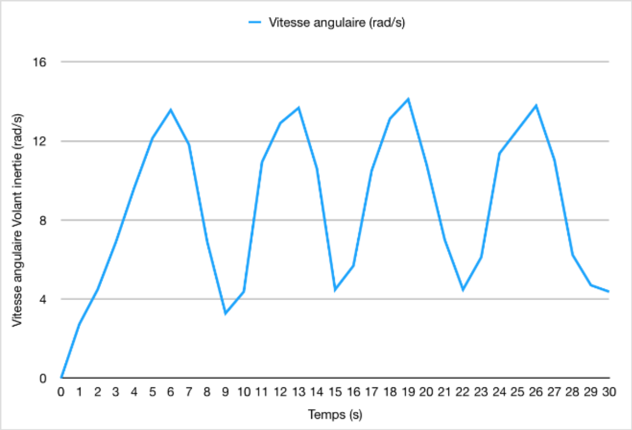
$$-\Gamma_{em} = -F_{Tract}\frac{R}{k_1 k_2 \eta_1 \eta_2 \eta_t} + \dot{\omega}_3 J_{eq,p}$$

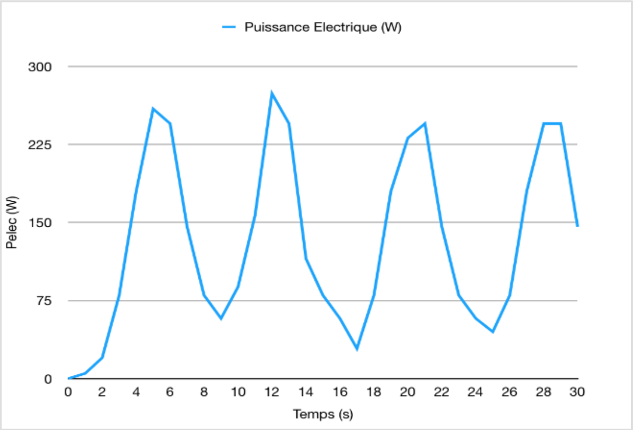
$$J_{eq,p} = J_3 + \frac{J_2}{k_2^2\eta_2} + \frac{J_1}{(k_1 k_2)^2\eta_1\eta_2\eta_t}$$

Expérience n°2 en images

Résultats expérience n°2

Vol Huit - 50m de ligne





- Variation de puissance liée à variation de tension (N) dans les lignes
- Solution : Accumulateur / Amortisseur Hydraulique

Synthèse
et perspectives

Synthèse

Synthèse :

- Estimer un ordre de grandeur de l'énergie produite par ce système
- Comprendre le fonctionnement de type "yo-yo" d'une machine
- Cerner les difficultés liés à la réalisation et au maintien du vol du cerf-volant

Bilan :

- Cerf-Volant Plus grand
- Haute Altitude (Vent plus régulier et plus fort).
- Coût Équipement : 4x moins cher (Prévisions) - Simplicité de fabrication et construction

Problèmes :

- Phase de décollage dangereuse
- Compensation des effets de la variation de tension (N) dans les lignes

Perspectives

Amélioration du Modèle :

- Etudier l'influence des rotations au cours du huit afin d'améliorer
- Prendre en compte la traînée du câble : Non négligeable pour de grande longueur et métal

- Mettre en place un dispositif de commande automatisé du cerf-volant (Réalisation des huits, ré-enroulage)
- Algorithme de prévision et de correction de trajectoire (MPPT)
- Etudier le principe d'amortisseur hydraulique permettant de compenser les variations de fréquences

Annexes

Annexes (1/3)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
    Created on Sat Oct 28 17:53:09 2020

@author: Louis
"""

from math import *
import matplotlib.pyplot as plt
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
from scipy import *
from pylab import *
from scipy.integrate import *

# Tracer plt.plot(abscisse, ordonnée)

def liste_temps(pas,tmax):
    l=[]
    a=1
    l.append(0)
    while l[-1]<tmax:
        l.append(a*pas)
        a+=1
    return l

# Vol en Huit (Paramétrage Thèses) fonctions
# pas = 0,01
def theta(t):
    return 60+5*sin(2*t)

def phi(t):
    return 5*sin(t)

def r(t):
    return 4*t #0.3m/s d'après littérature

# Vol en cercle fonctions

def thetacercle(rayon, t):
    return rayon*cos(t)

def phicercle(rayon,t):
    return rayon*sin(t)

def rpoint(tab):
    rpoint=[0]
```

1

```
    for i in range(len(tab)-1):
        rpoint.append((tab[i+1]-tab[i])/0.01)
    return rpoint

#Vol en huit tab
temps=liste_temps(0.01,20)
tabtheta = [theta(t) for t in temps]
tabphi = [phi(t) for t in temps]
tabr = [r(t) for t in temps]
tabrpoint = rpoint(tabr)

#Vol en cercle tab
tabthetacercle = [thetacercle(5,t) for t in temps]
tabphicercle = [phicercle(5,t) for t in temps]

# Transformation en système cartésien
def listex(lister,listetheta,listephi):
    x=[]
    for i in range(len(lister)):
        x.append(lister[i]*sin(listetheta[i])*cos(listephi[i]))
    return x

def listey(lister,listetheta,listephi):
    y=[]
    for i in range(len(lister)):
        y.append(lister[i]*sin(listetheta[i])*sin(listephi[i]))
    return y

def listez(lister,listetheta,listephi):
    z=[]
    for i in range(len(lister)):
        z.append(lister[i]*cos(listetheta[i]))
    return z

tabx = listex(tabr,tabtheta,tabphi)
taby = listey(tabr,tabtheta,tabphi)
tabz = listez(tabr,tabtheta,tabphi)

def evolution_3d(x,y,z):
    fig= plt.figure()
    ax = fig.gca(projection='3d') # Affichage en 3d
    ax.plot(x,y,z, label ='Courbe en 3d')
    plt.tight_layout()
    ax.set_yticklabels([])
    ax.set_xticklabels([])
    ax.set_zticklabels([])
    #plt.savefig("Plot generated using Matplotlib.png",dpi=400)
    return plt.show
```

2

Annexes (2/3)

```python
# Répartion du vent en fonction de l'altidue et du vent connu à
# une altitude connue
def distributionvent(vconnue, altconnue, alpha, z):
    return vconnue*(z/altconnue)**(alpha)

altitude = [i for i in range(1001)]
tabvent=[distributionvent(1,10,1/7,z) for z in altitude]

# Evolution de la tension de corde
# Voile Deca pw 1.8m
mkite = 0,18
g = 9.81
L = 20
rhoair = 1.225
ct= 0.15
cp = 1.2
S = 1.26

def tension(lister,listerpoint,listetheta,listephi,m,rho,s,cp,v):
    T=[0]
    for i in range(len(lister)-1):
        poids = -m*9.81*cos(listetheta[i])
        aero = 0.5*rho*s*(cp)*(v*sin(listetheta[i])*cos(listephi[i])-lis
        acce = -m*((listerpoint[i+1]-listerpoint[i])/0.01-lister[i]*((li
        A=poids+aero+acce
        T.append(abs(A))
    return T

#Tensions standarts pour vol 8 et O. Valide.
tensionstandarthuit = tension(tabr,tabrpoint,tabtheta,tabphi,2.5,1.225,5
tensionstandartcercle = tension(tabr,tabrpoint,tabthetacercle,tabphicerc
"""Possibilité de tracé tension en fonction temps et voir augmentation d
tension ou sinon tracé de tension en fonction de theta pour voir zone de
puissance
"""
"""
CC : Vol en huit genere plus de puissance
"""
# Tensions Modeles Expérimental
tensionsexperimentalhuit = tension(tabr,tabrpoint,tabtheta,tabphi,0.18,1
tensionsexperimentalcercle = tension(tabr,tabrpoint,tabthetacercle,tabph

"""
Modèle Valide. Cohérence avec expérience
"""
def puissancemechanique(tension,vitesse):
    P=[]
```

3

```python
    for i in range(len(tension)):
        val = tension[i]*vitesse[i]
        P.append(val)
    return P

def puissancemoyenne(tabpuissance):
    return (sum(tabpuissance))/len(tabpuissance)

# Puissances mécaniques standarts pour vol 8 et O. Valide Cohérent pour
# avec theses (notamment celle à déployement 0.5m/s) mais pour le cercle
tabpuissancestandarthuit = puissancemechanique(tensionstandarthuit,tabrp
tabpuissancestandartcercle = puissancemechanique(tensionstandartcercle,t

tabpuissanceexperimentalhuit = puissancemechanique(tensionsexperimentalh
tabpuissanceexperimentalcercle = puissancemechanique(tensionsexperimenta
""" Possibilité de tracé en fonction du temps et de theta
"""

def tracegraphe(temps,tension):
    plt.plot(temps,tension)
    plt.legend()
    plt.xlabel('Angle Phi (°)')
    plt.ylabel('Va (m/s)')
    plt.savefig(" ",dpi=400)
    plt.grid()
    plt.title('Va (m/s) en fonction de phi (°)')
    plt.show()

# Phase de rétractation
"""Kite au Zénith rpoint<0 donc P=Frpoint<0
Kite statique => rpointpoint =0, thetapoint = 0, phi point 0
Donc theta = 0, phi = 0 avec constant prise à zéro (au zénith)
Equation tension devient : T =-mg-1/2pS(Cp)rpoint^2
"""
def rretractation(t,altitudemax,vitessederetractation):
    return altitudemax+vitessederetractation*t

tempsdescente = liste_temps(0.01,80/3)
tabrretractation= [rretractation(t,80,-3) for t in tempsdescente]
# pour revenir à 0m d'altitude à une vitesse de 2,3m/s. Pour changer
# altitudemax changer durer simu

def puissanceretractation(listetempsdescente,vdescente,m,rho,s,cp):
    P=[]
    for i in range(len(listetempsdescente)):
        poids = -m*9.81
        aero = 0.5*rho*s*(cp)*(vdescente)**2
        p = (poids+aero)*vdescente
```

4

Annexes (3/3)

```python
            P.append(p)
        return P

tabpuissanceretra = puissanceretractation(tempsdescente,[-3],2.5,1.2,5,1

# Pour marquer la qté d'énergie : affiche couche couleur
# sous la courbe ax.fill_between(t, 0, function(t))

# Affichage

def trace_puissance(t,td,p,pretra):
    T = liste_temps(0.01,20+80/3+0.01+20+0.01+80/3+0.01)
    P=p+pretra+p+pretra
    plt.plot(T,P)
    plt.fill_between(T, P, color='#cd5353')
    plt.legend()
    plt.xlabel('temps')
    plt.ylabel('Puissance (W)')
    plt.savefig("Plot generated using Matplotlib5.png",dpi=400)
    plt.grid()
    plt.title('Puissance (W) en fonction du temps')
    plt.show()
    b = np.trapz(P,T,dx = 0.01)
    print(b)

# Phase Transmission de Puissance
Rayontreuil = 25.4
k_1 = 2.86
k_2 = 5.00
n_1 = 0.9
n_2 = 0.9
n_t = 0.9
J_1 = 0.111
J_2 = 0.216
J_3 = 0.00003

def vitesse_rotation(k_1,k_2,r,listevitesse):
    omega = []
    for i in range(len(listevitesse)):
        a = k_1*k_2*listevitesse[i]/r
        omega.append(a)
    return omega

tabvitesserotationhuit = (k_1,k_2,Rayontreuil,tabrpoint)

def couple_em(listeforce,listevitesse,R,k_1,k_2,n_1,n_2,n_t,J_3,J_2,J_1)
    C =[]
    for i in range(len(listeforce)):
```

5

```python
        a = listeforce[i]*R/(k_1*k_2*n_1*n_2*n_t)
        b = (J_3+J_2/(k_2**2*n_1)+J_1/(k_1**2*k_2**2*n_1*n_2*n_t))
        c = k_1*k_2/R*((listevitesse[i]-listevitesse[i-1])/0.01)
        d = a+b*c
        C.append(d)
    return C

#Modeles
tabcoupleemhuit = couple_em(tensionsexperimentalhuit,tabrpoint,Rayontreu

# Etude theta : theta(0) = 20° Faire Varier gamma
a = 1
b = 1.5
rho = 1.292
m = 0.18
R = 25
V = 10
g = 9.81
gamma = 0.01

def F(Y,t):
    return [Y[1], ((a*b*rho*sin(gamma))/(m*R))*(V*cos(Y[0]+gamma)-R*Y[1]

def solution(v,tmax):
    T = np.linspace(0,10,301)
    Z = odeint(F,[20,v],T)
    X = [z[0] for z in Z]
    V = [z[1] for z in Z]
    return X,V,T

def trace_angle(vmax,tmax):
    for dtheta in range(0,vmax+1):
        X,V,T = solution(dtheta,tmax)
        plt.plot(T,X,label = 'dtheta={}'.format(dtheta))
    plt.legend()
    plt.xlabel('temps')
    plt.ylabel('Theta')
    #plt.savefig("Plot generated using Matplotlib3.png",dpi=400)
    plt.grid()
    plt.title('Theta en fonction du temps')
    plt.show()

"""
CC :
Pour gamma < gammacritique (<0.01): l'aile s'eleve et n'arrete pas de s'
fonction de la vitesse angulaire initiale. Elle dépasse meme 90°. Les
hypothèses ne sont plus vérifiés, le vent arrive au-dessus de l'aile, et
les câbles se détendent. Cette solution n'existe pas dans la réalité
```

6

```python
Pour gamma > gammacritique (>1) : l'aile retombe au sol
Pour gamma = gamma critique (entre 0.1 et 1) : soit trajectoire oscillat
amorties(0.1). au dela l'aile atteint son equilibre en un temps d'ordre
Resultats valables dans le cas 2D Fluide parfait. La présence de viscosi
ne change pas fondamentalement les résultats mais tend à diminuer l'angl
d'équilibre.

"""

# Zones De Puissance
# Va= V*cos(phi)cos(theta)/sin(e)

def vitesse_apparente(V,lt,lp):
    va = []
    for i in range(len(lt)):
        elt = -V*(cos(lp[i])*sin(lt[i]))/sin(0.12) # f=Cp/ct = 1/tan(e)
        va.append(elt)
    return va

tabvahuit = vitesse_apparente(2,tabtheta,tabphi)
tabvacercle = vitesse_apparente(2,tabthetacercle,tabphicercle)
```

7