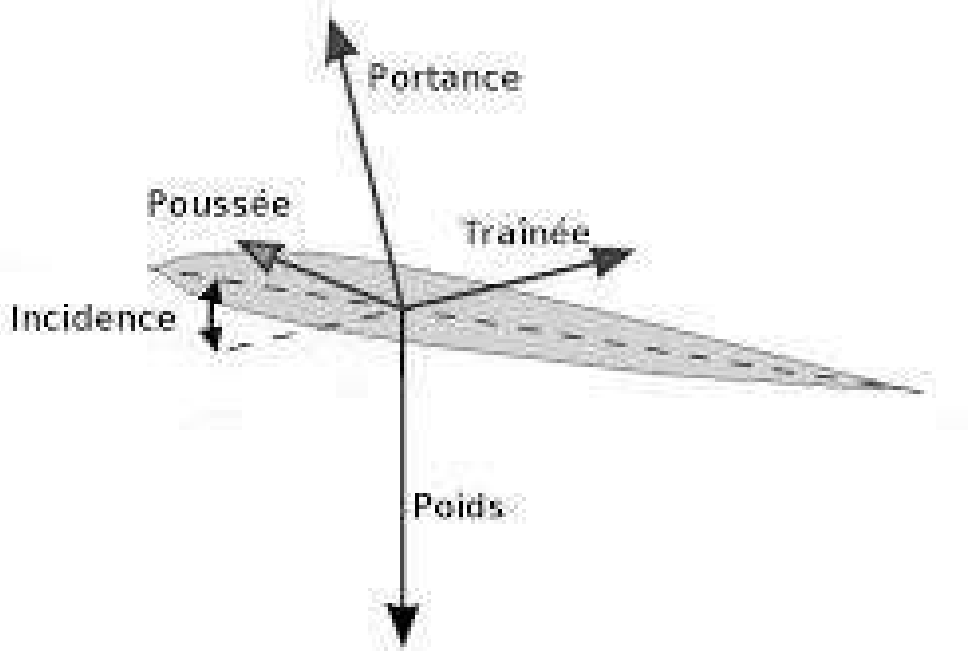
Du papier pour étudier l'aérodynamique

- Comment volent les avions
- Paramètres d'études et contextualisation
- Expériences sur le papier
- Expériences sur le vol
- Interprétations et conclusion

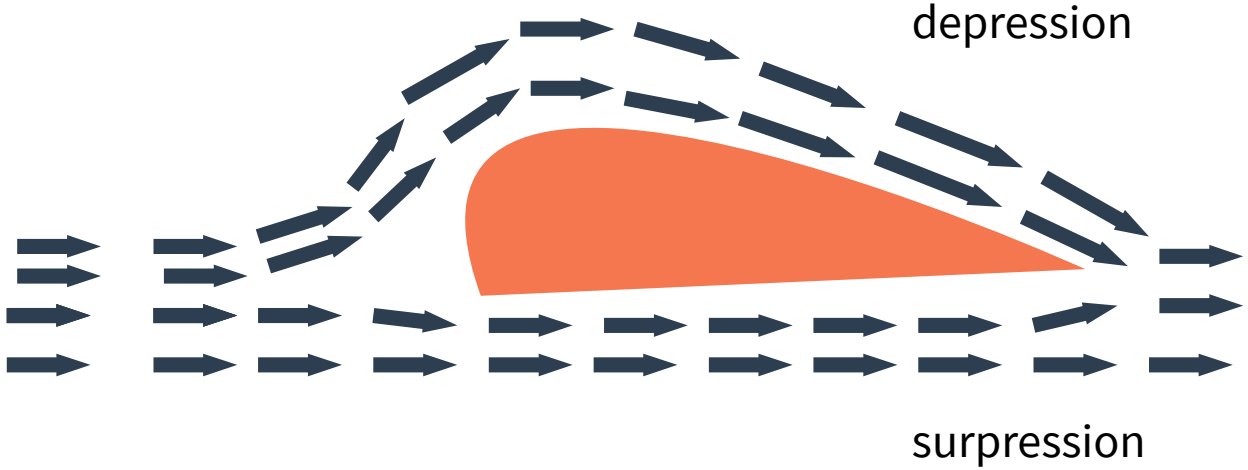
N° d'inscription: 7878

Comment volent les avions ?

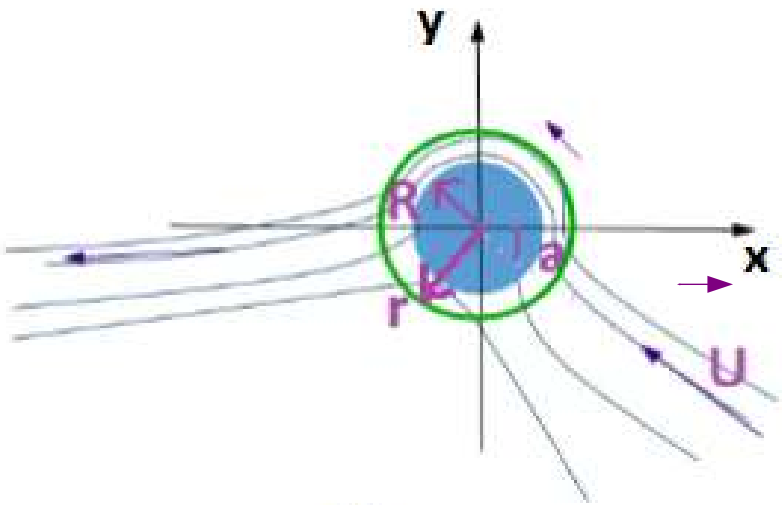


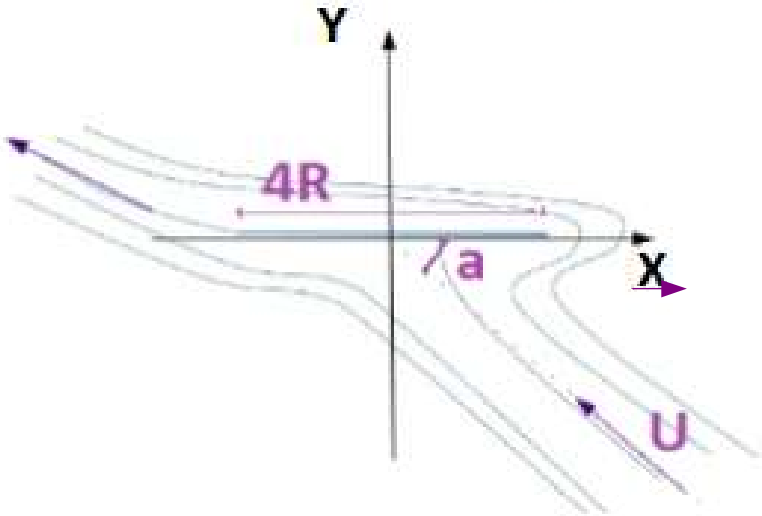
$$F_p = \frac{\rho U^2 S}{2} C_d$$

$$F_t = \frac{\rho U^2 S}{2} C_t$$



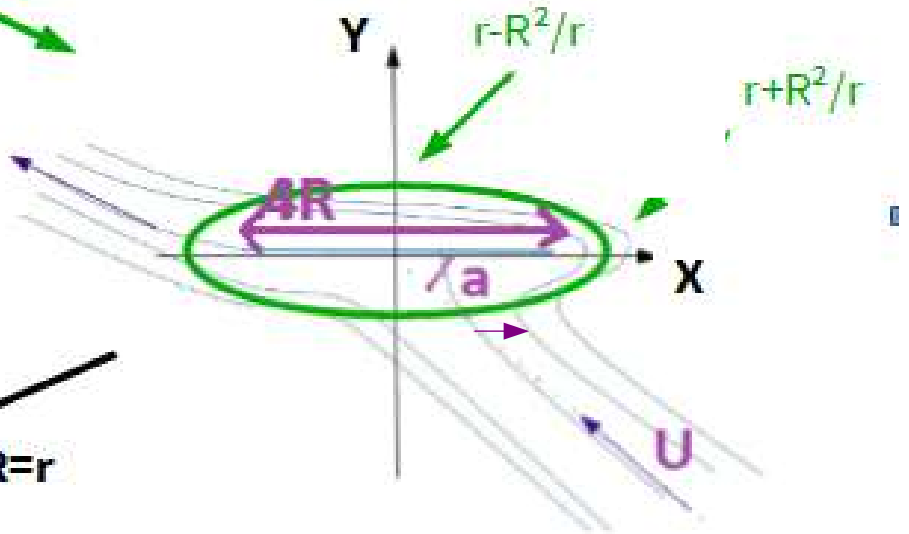
Comment volent les avions ?





Transformation de Joukovski

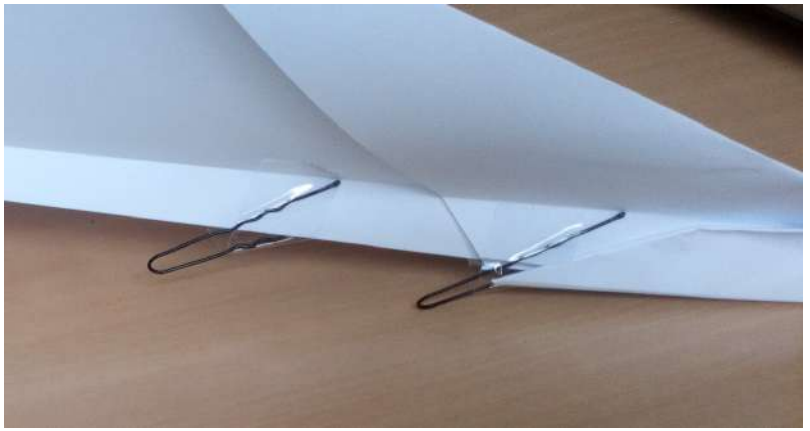
$$Z = X + iY = g(z) = z + \frac{R^2}{z}, z = x + iy$$

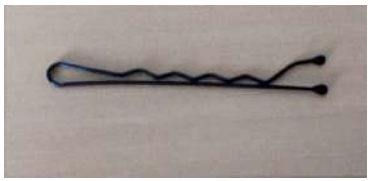


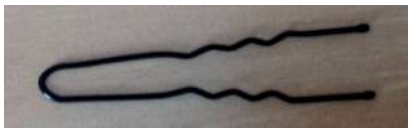
R=r

$$F_p = 4 R \pi \rho U^2 \sin \alpha$$

Construction d’une rampe et pince





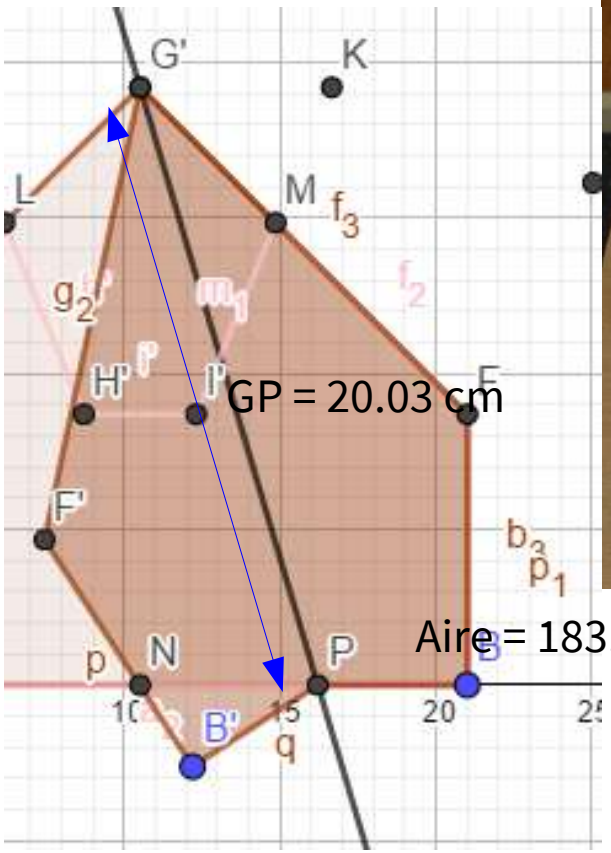


Les caractéristiques des papiers

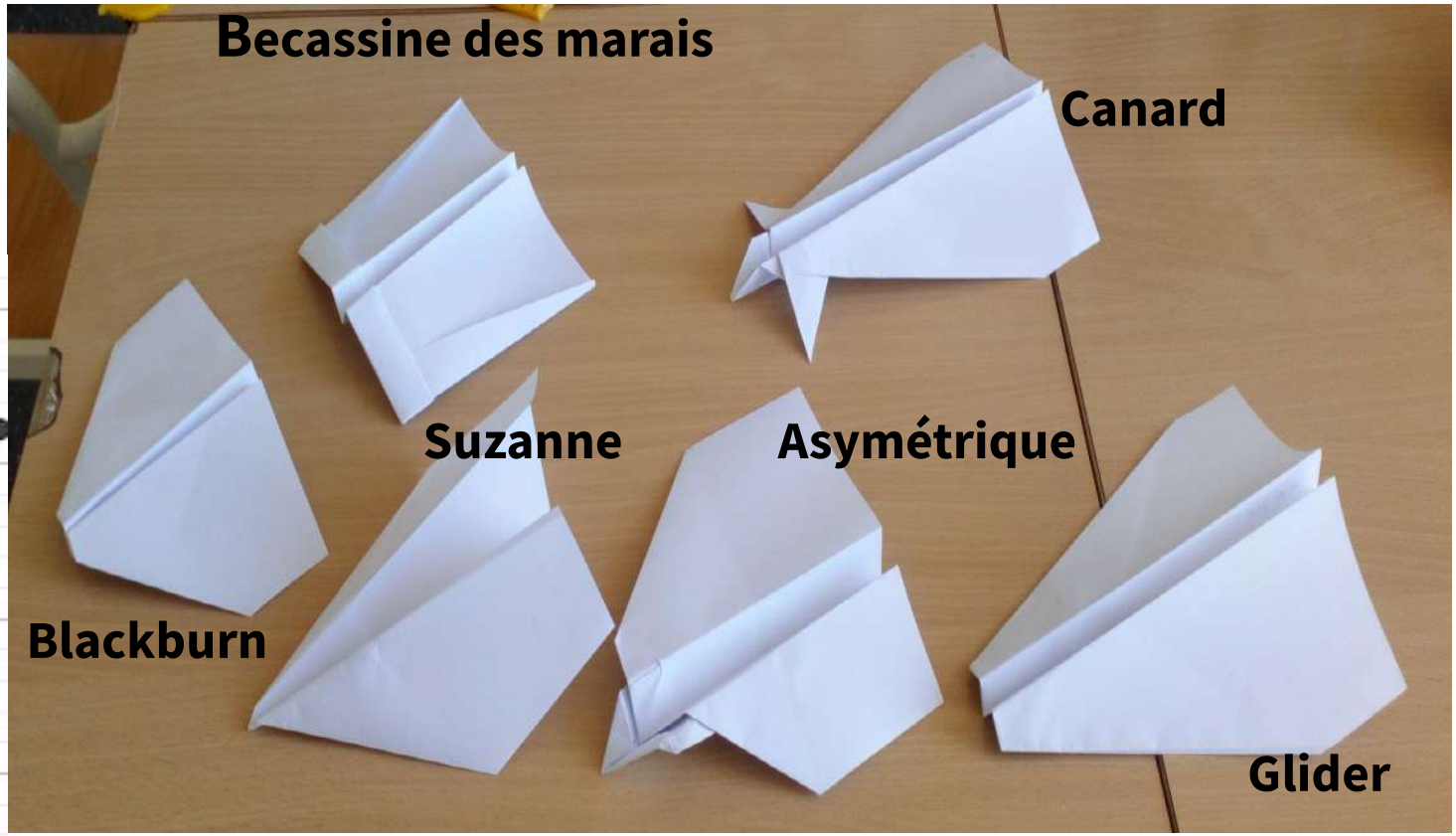
(G grammage, E épaisseur de la feuille de papier, H hauteur des aspérités, Iaf indice de perméabilité AFNOR, Ie résistance à l'éclatement, Id indice de déchirement)

Papier	G $g.m^{-2}$		E μm		H μm		Iaf (Bekk) 10cm3.m-2.Pa-1.s-1		Iaf (Brend) 10cm3.m-2.Pa-1.s-1		P pour Ie kPa		Ie kPa.g-1.m²	
	Moy	EC	Moy	EC	Moy	EC	Moy	EC	Moy	EC	Moy	EC	Moy	EC
Ikea	61.4	0.2	85	2	3.5	0.2	15	1	15	4	110	4	1.79	0.07
Cnew	44.1	0.6	62	2	5.8	0.5	23	3	29	4	67	0	1.52	0
Blanc1	62.8	0.6	83	1	5.7	0.3	57	4	64	10	194	10	3.1	0.2
Calqu	77.2	0.4	63.0	0.6	5.9	0.1	1.4	0.1	1.8	0.2	237	12	3.1	0.2
Immo	74.5	0.3	58	1	1.5	0.2	0.3	0.001	0.13	0.2	117	6	1.57	0.08
MidiL	43	1	55	1	4.8	0.6	21	4	20	6	70	0	1.63	0
Blanc2	82.8	0.5	99	2	5.5	0.7	50	4	39	4	236	12	2.9	0.1

Les modèles

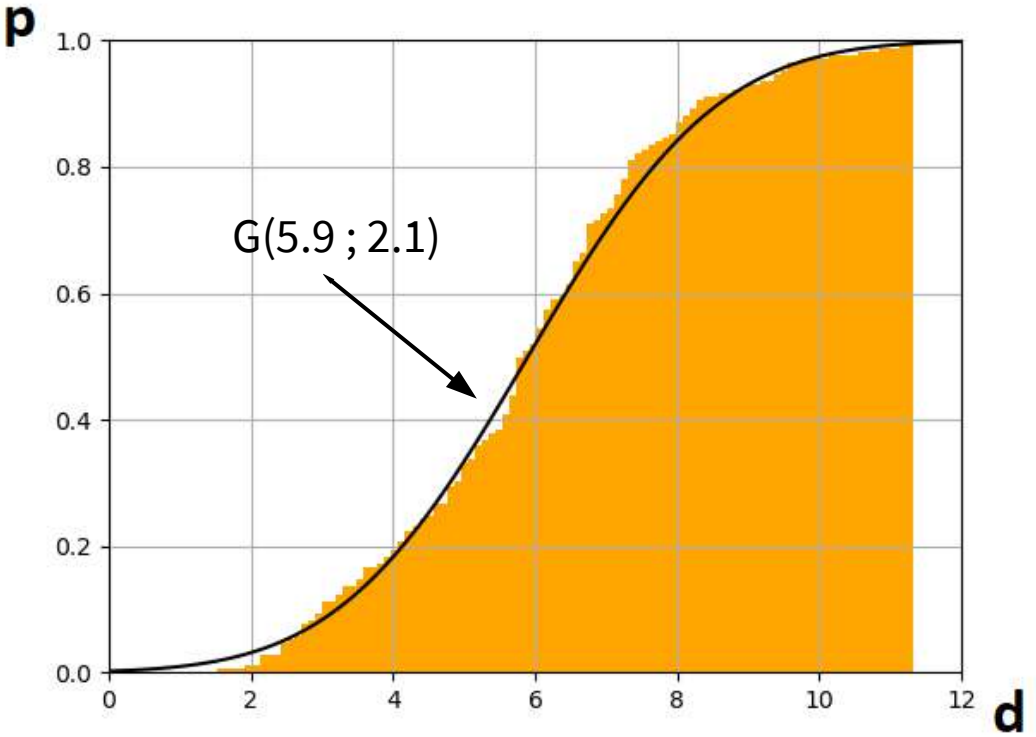


Aire = 183.44 cm²



Cadre et piste de vol

- Un gymnase



- Un couloir
- Une salle de classe avec tableau à craie

Expérience qualitative : ligne droite et distance

Paramètre étudié: le papier
Modèle: blackburn
La mesure: distance de vol et de glisse
Résultats:

Papier	Ikea	Cnews	Blanc 1
Moyenne (m)	6,11	5,55	6,84
Ecart type	1,6	1,2	2,4

Paramètre étudié: le modèle
Papier : blanc 80g
La mesure: distance de vol et de glisse
Résultats:

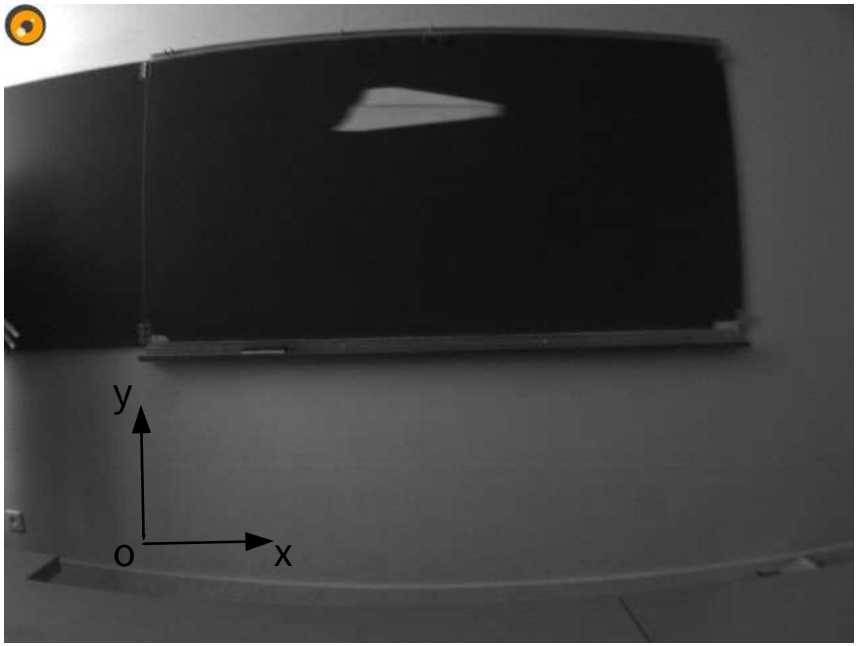
Pince 8 cm

Modèle	Classique	Glider
Moyenne (m)	8,75	6,19
Ecart-type	1,3	1,1

Pince 10cm

Modèle	Classique	Glider
Moyenne (m)	8,43	6,43
Ecart-type	0,9	1,2

Expériences quantitatives, mesure des forces



$$F_t = m a_x \cos(\alpha) + m(g + a_y)\sin(\alpha)$$

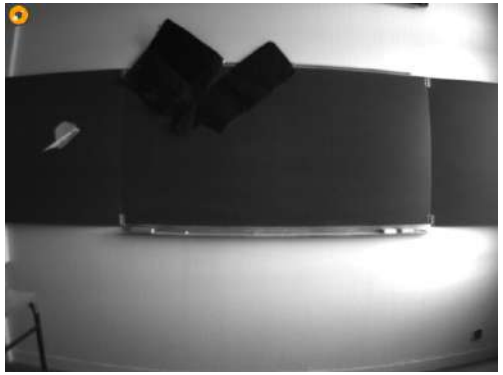
$$F_p = -m a_x \cos(\alpha) + m a_y \sin(\alpha)$$

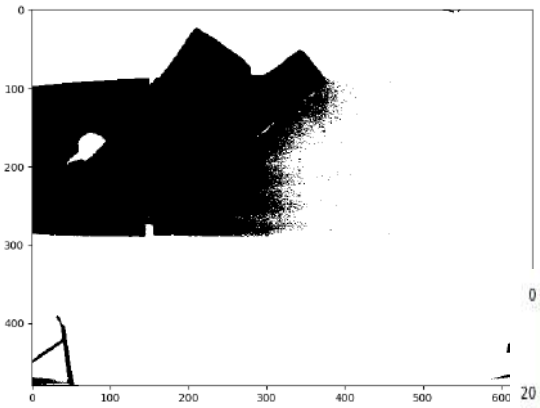
$$F_t = \frac{\rho U^2 S}{2} C_t$$

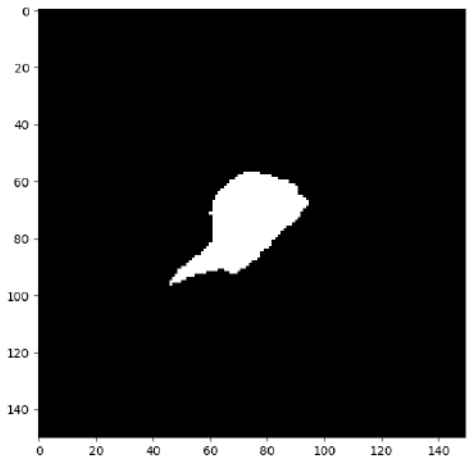
$$F_p = \frac{\rho U^2 S}{2} C_d$$

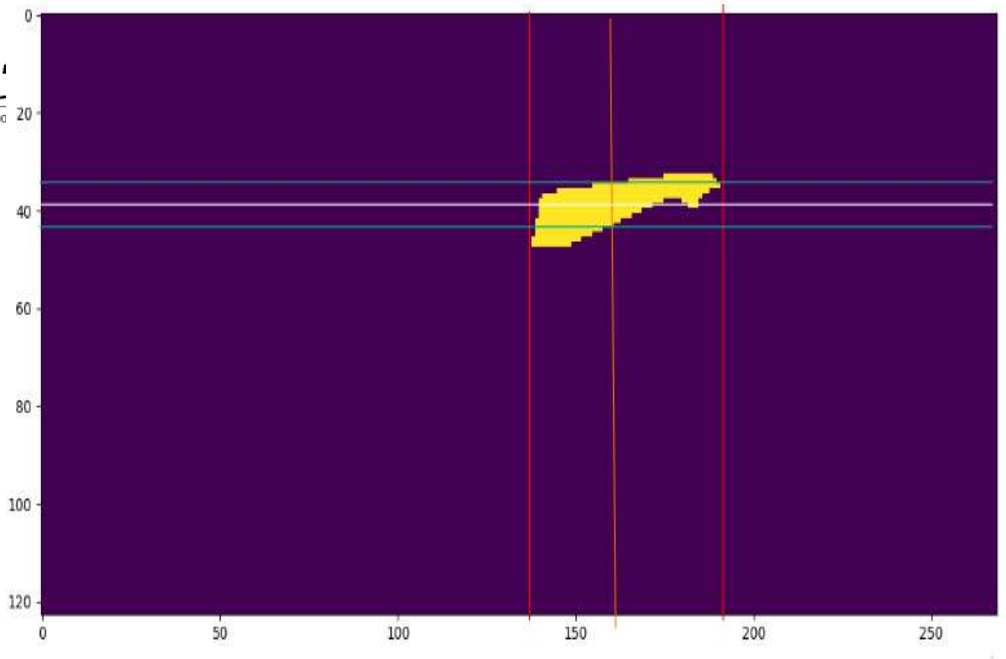
Expériences quantitatives, repérage de l'avion avec algorithme

Filtrage par seuillage





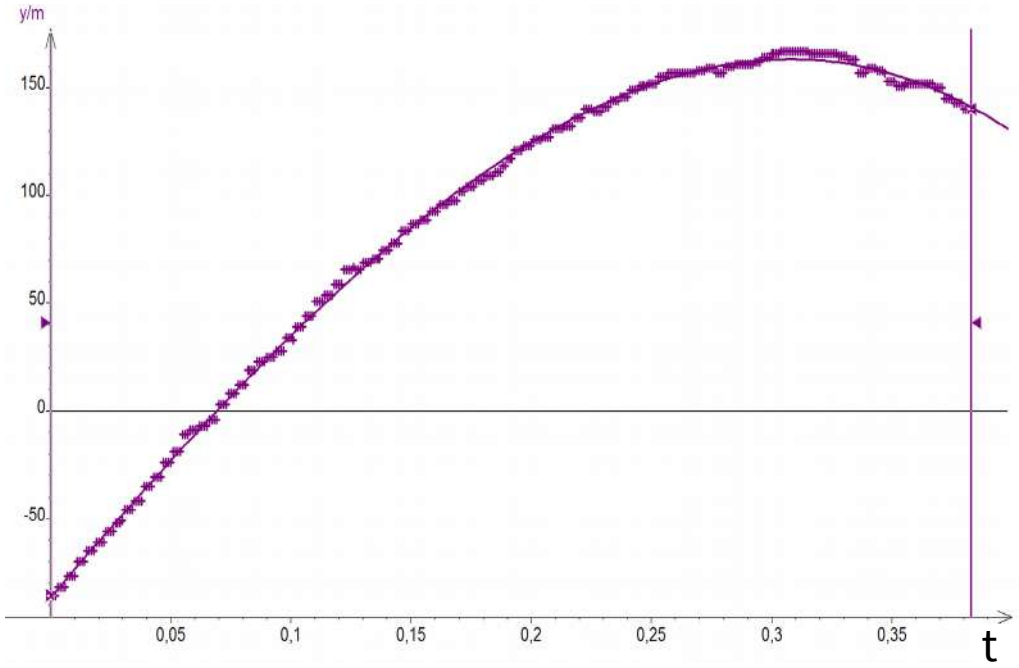




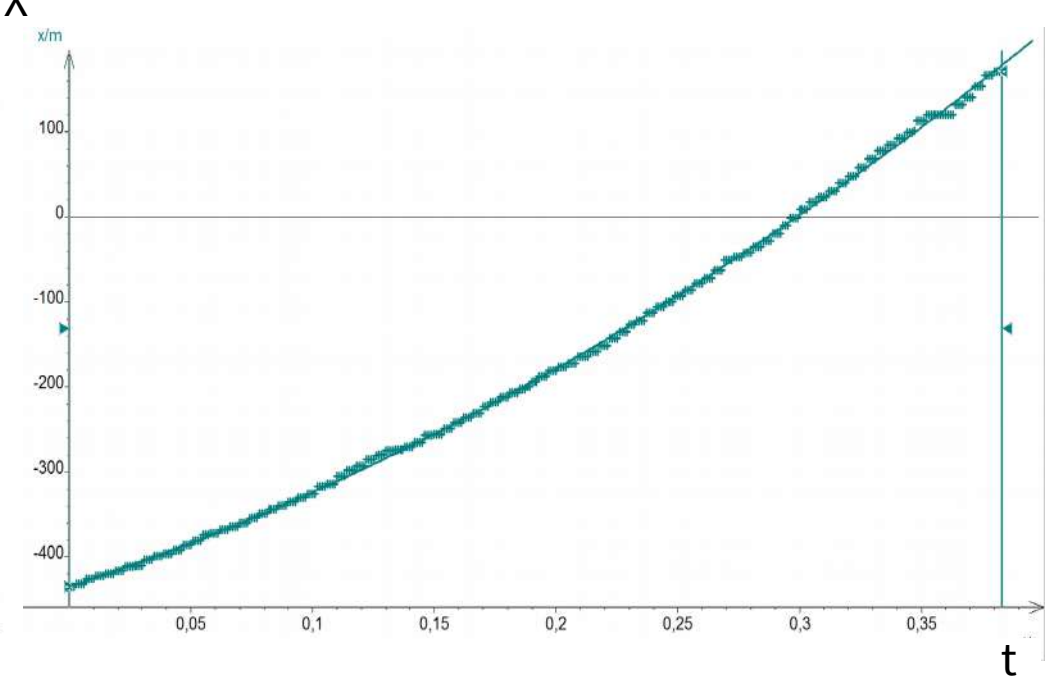
Expériences quantitatives, repérage de l'avion

Pointage manuel à l'aide de Regressi et modélisation :

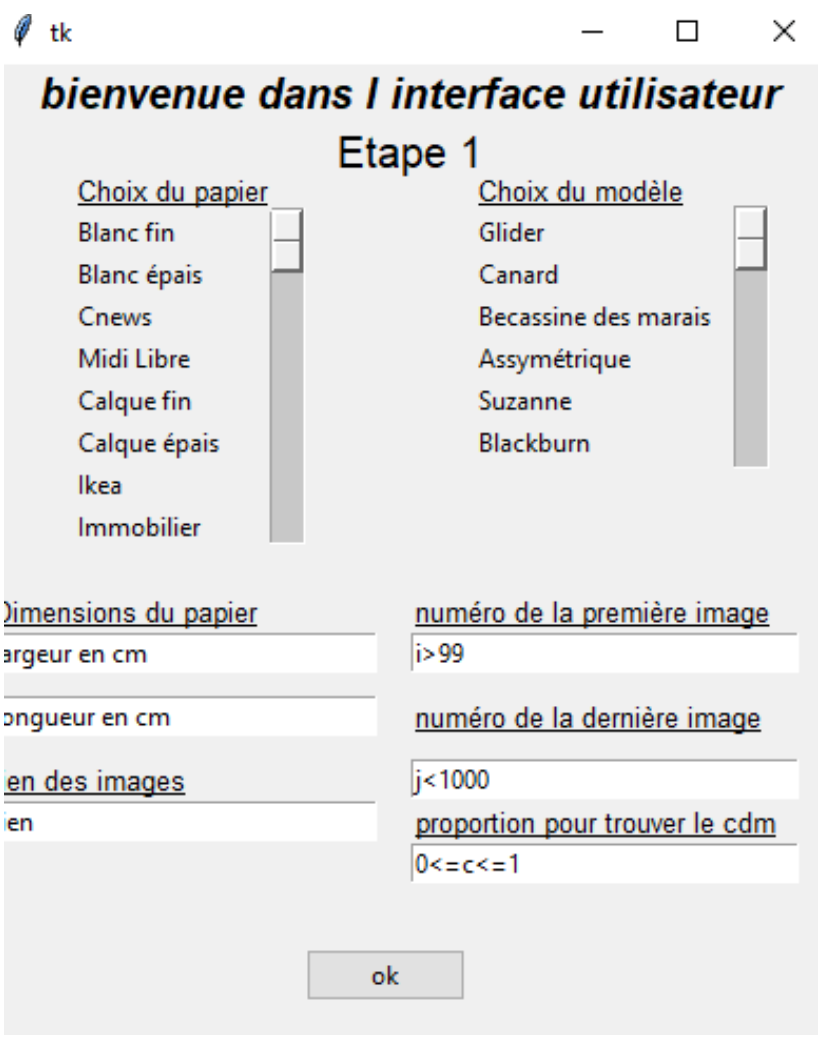
Y

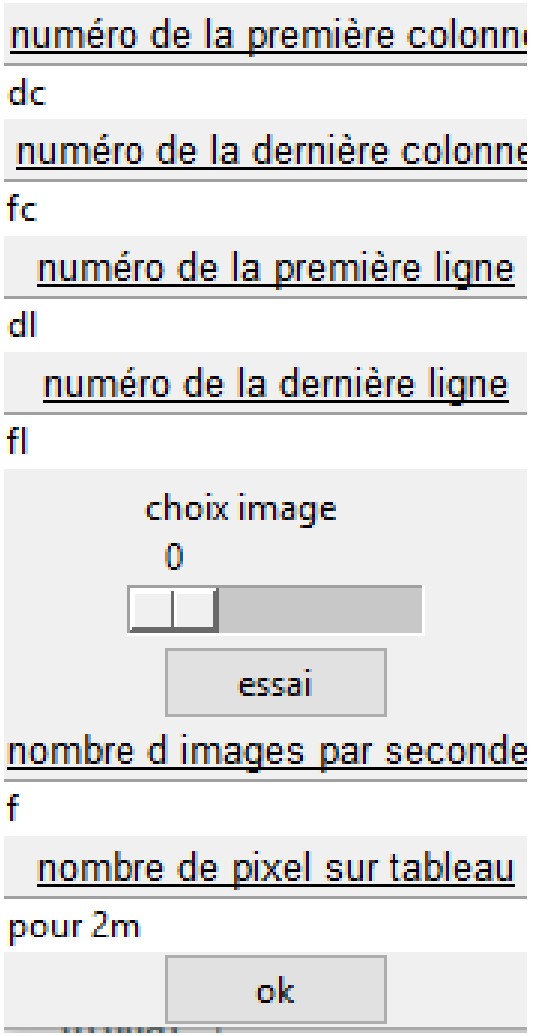


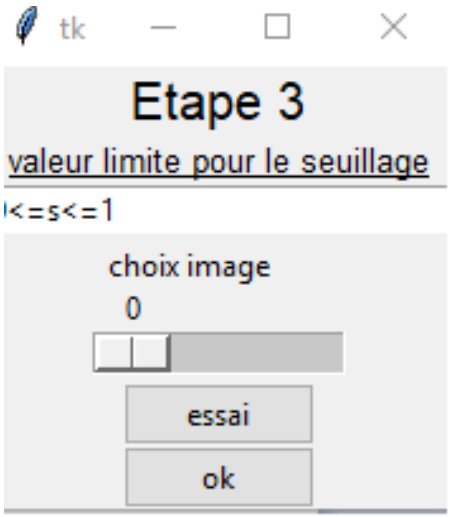
X

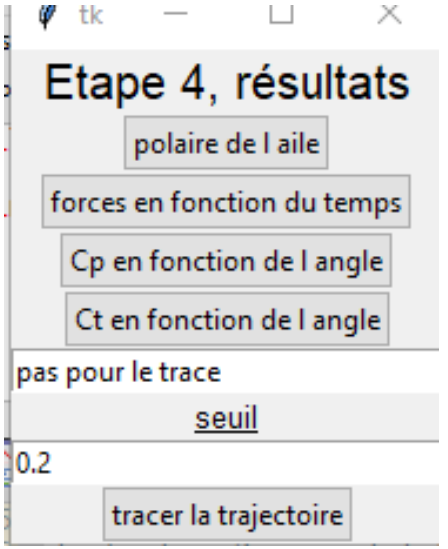


Interface graphique

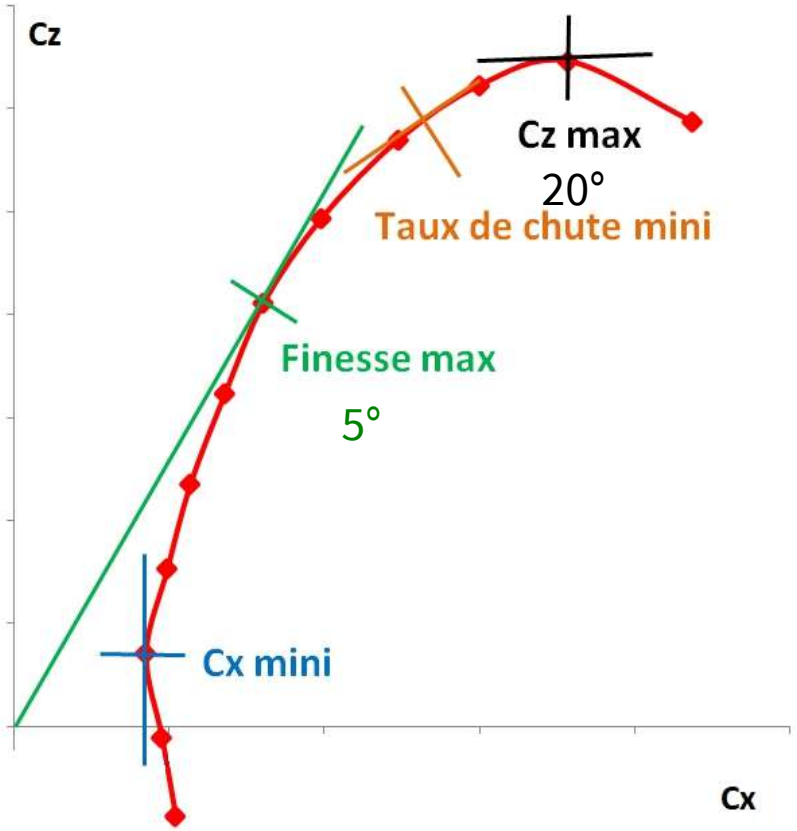


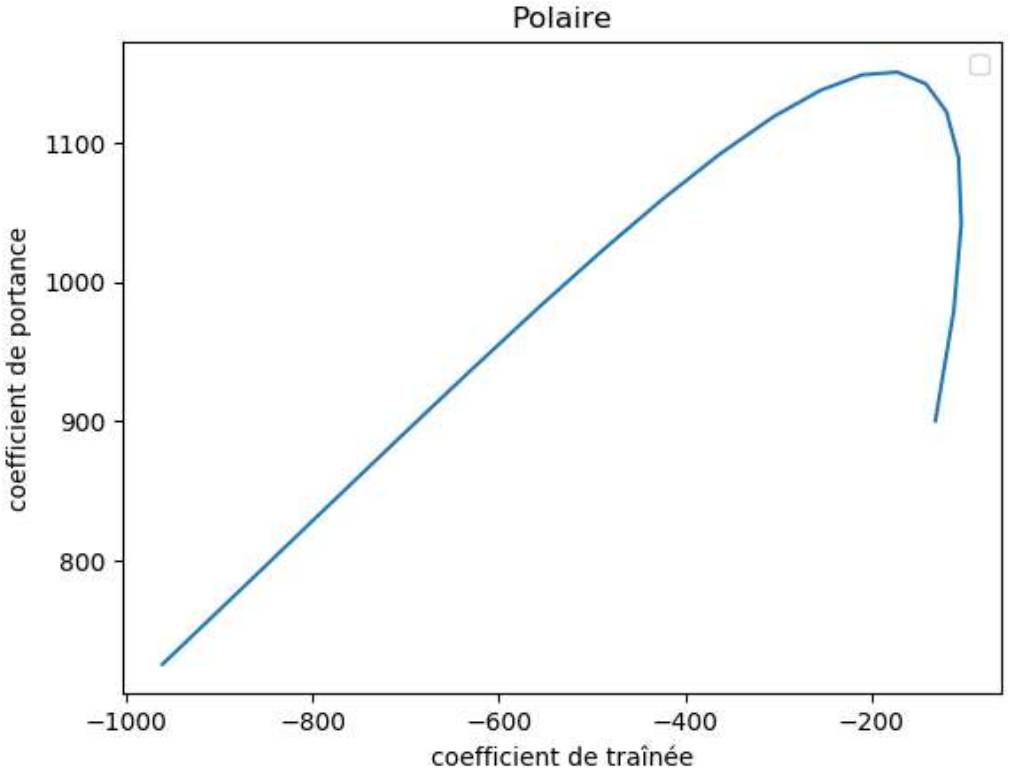






Expériences quantitatives : résultats, polaire d'aile

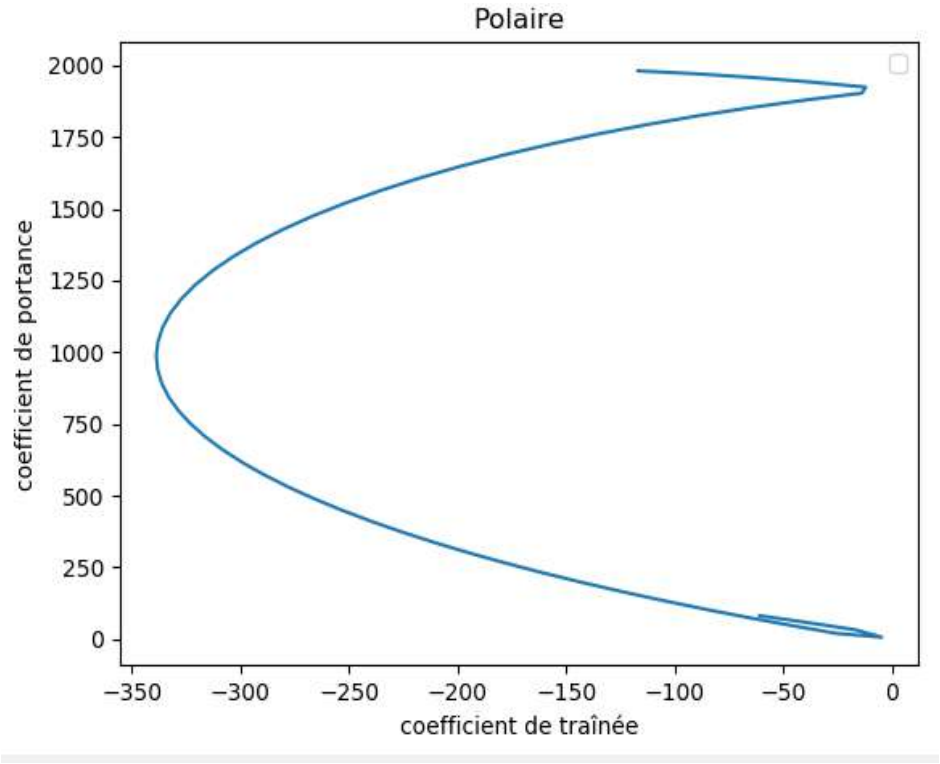




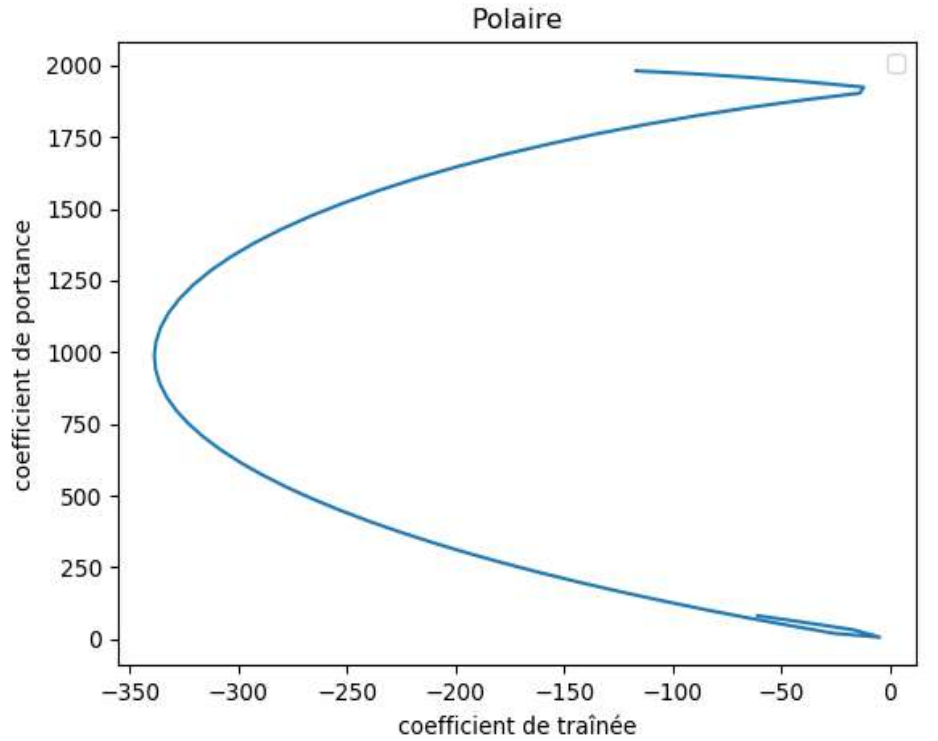
Courbe théorique

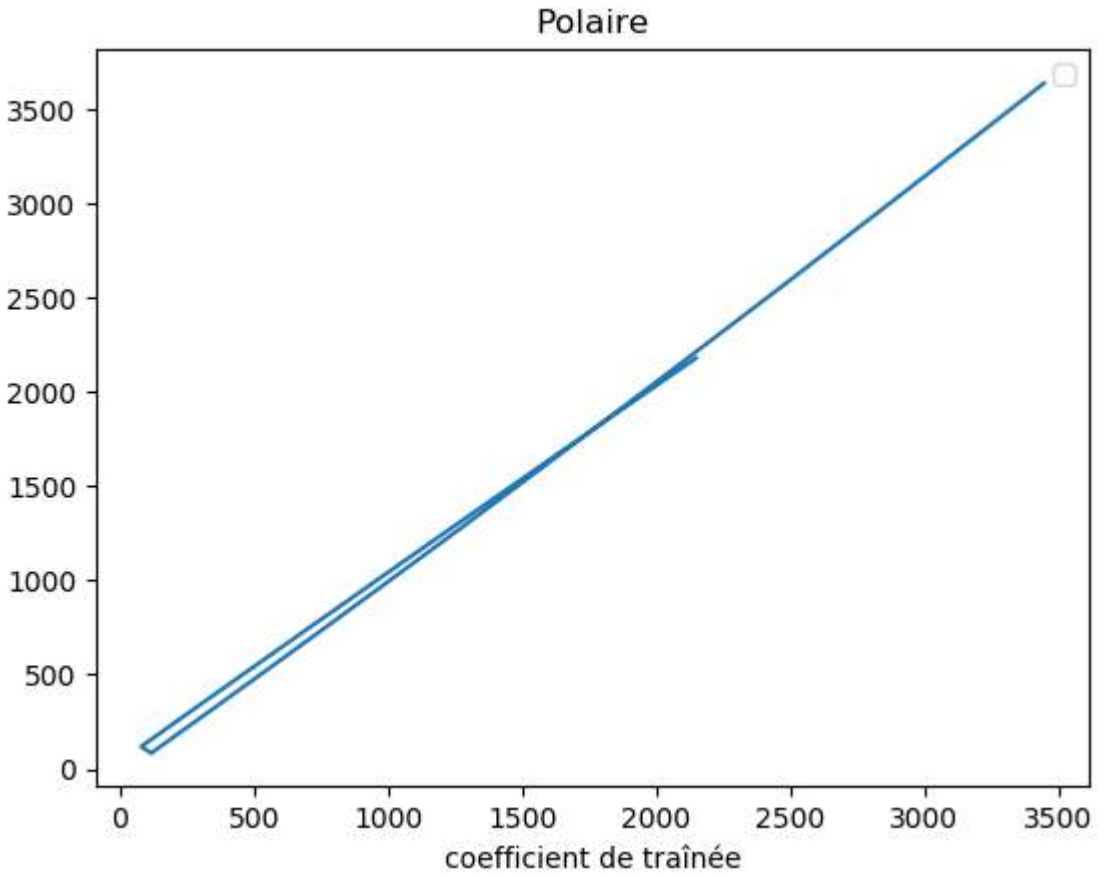
Courbe expérimentale obtenue

Expériences quantitatives : résultats



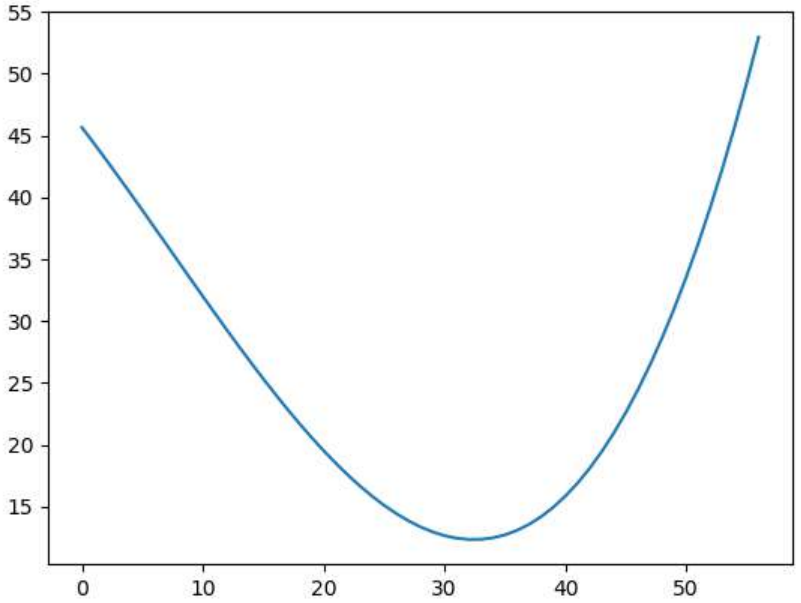
Expériences quantitatives : résultats



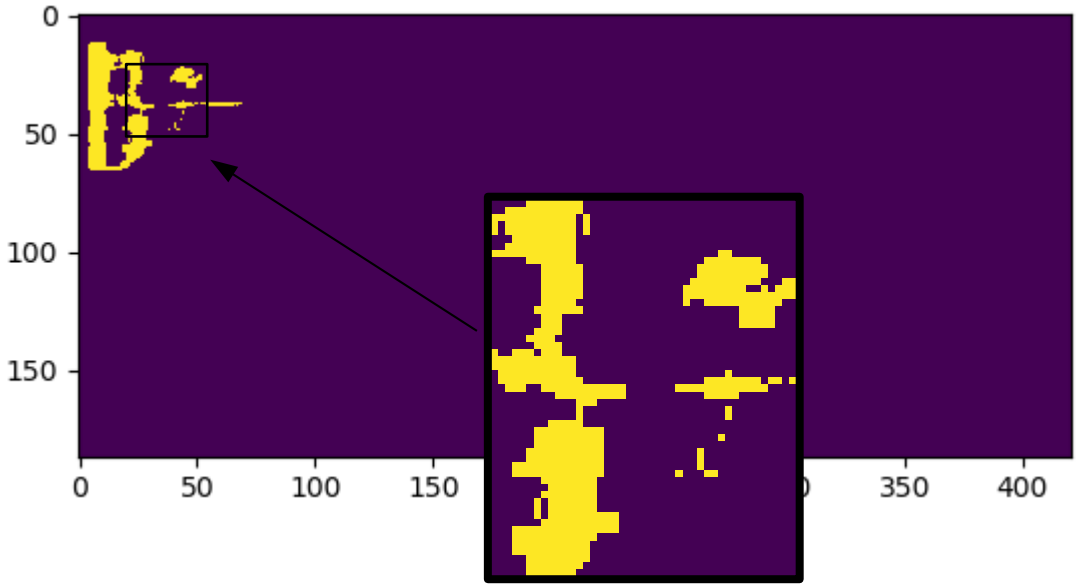


Expériences quantitatives : résultats

Problème d'injectivité



Problème de fermeture



Problème d'angle d'incidence trop grand

Alternatives et résultat

Angle : 5°	Glider	Canard	Becassine	Suzanne	Blackburn
Blanc 1	0,80	X	0,97	1,11	1,45
Calque e	1,44	0,81	0,88	0,93	1,32
Calque f	1,87	1,13	1,00	1,31	1,69
Cnews	X	0,75	2,51	0,53	X
Midil	1,12	0,99	X	1,30	X
Ikea	0,89	0,02	0,93	2,18	0,82
Immob	4,59	1,00	0,08	0,95	1,40

Angle : 20°	Glider	Canard	Becassine	Suzanne	Blackburn
Blanc e	X	0,25	X	0,74	0,91
Calque e	1,18	X	0,28	0,59	X
Calque f	X	X	0,01	0,60	1,54
Cnews	1,39	0,12	0,18	0,98	X
Midil	0,98	X	4,80	1,40	X
Ikea	X	1,08	1,11	0,09	0,68
Immob	2,02	0,99	0,97	0,2	1,20

Alternatives et résultat

Angle : 5°	Glider	Canard	Becassine	Suzanne	Blackburn
Blanc 1	0,80	X	0,97	1,11	1,45
~~Calque e~~	~~1,44~~	~~0,81~~	~~0,88~~	~~0,55~~	~~1,52~~
Calque f	1,87	1,13	1,00	1,31	1,69
Cnews	X	0,75	2,51	0,53	X
Midil	1,12	0,99	X	1,30	X
Ikea	0,89	0,02	0,93	2,18	0,82
Immob	4,59	1,00	0,08	0,95	1,40

Angle : 20°	Glider	Canard	Becassine	Suzanne	Blackburn
Blanc e	X	0,25	X	0,74	0,91
~~Calque e~~	~~1,18~~	~~X~~	~~0,28~~	~~0,59~~	~~X~~
Calque f	X	X	0,01	0,60	1,54
Cnews	1,39	0,12	0,18	0,98	X
Midil	0,98	X	4,80	1,40	X
Ikea	X	1,08	1,11	0,09	0,68
Immob	2,02	0,99	0,97	0,2	1,20

Alternatives et résultat

Angle : 5°	Glider	Canard	Becassine	Suzanne	Blackburn
Blanc 1	0,80	X	0,97	1,11	1,45
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Calque f	1,87	1,13	1,00	1,31	1,69
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Midil	1,12	0,99	X	1,30	X
Ikea	0,89	0,02	0,93	2,18	0,82
Immob	4,59	1,00	0,08	0,95	1,40

Angle : 20°	Glider	Canard	Becassine	Suzanne	Blackburn
Blanc e	X	0,25	X	0,74	0,91
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Calque f	X	X	0,01	0,60	1,54
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Midil	0,98	X	4,80	1,40	X
Ikea	X	1,08	1,11	0,09	0,68
Immob	2,02	0,99	0,97	0,2	1,20

Alternatives et résultat

Angle : 5°	Glider	Canard	Becassine	Suzanne	Blackburn
Blanc 1	0,80	X	0,97	1,11	1,45
~~Calque e~~	~~1,44~~	~~0,81~~	~~0,88~~	~~0,93~~	~~1,32~~
Calque f	1,87	1,13	1,00	1,31	1,69
~~Chews~~	~~X~~	~~0,75~~	~~2,51~~	~~0,55~~	~~X~~
Midil	1,12	0,99	X	1,30	X
~~Ikea~~	~~0,85~~	~~0,82~~	~~0,93~~	~~2,16~~	~~0,82~~
Immob	4,59	1,00	0,08	0,95	1,40

Angle : 20°	Glider	Canard	Becassine	Suzanne	Blackburn
Blanc e	X	0,25	X	0,74	0,91
~~Calque e~~	~~1,16~~	~~X~~	~~0,28~~	~~0,59~~	~~X~~
Calque f	X	X	0,01	0,60	1,54
~~Chews~~	~~1,39~~	~~0,12~~	~~0,16~~	~~0,98~~	~~X~~
Midil	0,98	X	4,80	1,40	X
~~Ikea~~	~~X~~	~~1,06~~	~~1,11~~	~~0,69~~	~~0,68~~
Immob	2,02	0,99	0,97	0,2	1,20

Alternatives et résultat

Angle : 5°	Glider	Canard	Becassine	Suzanne	Blackburn
Blanc 1	0,80	X	0,97	1,11	1,45
~~Calque e~~	~~1,44~~	~~0,81~~	~~0,88~~	~~0,55~~	~~1,52~~
~~Calque f~~	~~1,07~~	~~1,15~~	~~1,08~~	~~1,31~~	~~1,69~~
~~Chews~~	~~X~~	~~0,75~~	~~2,51~~	~~0,55~~	~~X~~
Midil	1,12	0,99	X	1,30	X
~~Ikea~~	~~0,85~~	~~0,82~~	~~0,55~~	~~2,16~~	~~0,82~~
Immob	4,59	1,00	0,08	0,95	1,40

Angle : 20°	Glider	Canard	Becassine	Suzanne	Blackburn
Blanc e	X	0,25	X	0,74	0,91
~~Calque e~~	~~1,18~~	~~X~~	~~0,28~~	~~0,59~~	~~X~~
~~Calque f~~	~~X~~	~~X~~	~~0,81~~	~~0,68~~	~~1,54~~
~~Chews~~	~~1,39~~	~~0,12~~	~~0,18~~	~~0,98~~	~~X~~
Midil	0,98	X	4,80	1,40	X
~~Ikea~~	~~X~~	~~1,06~~	~~1,11~~	~~0,09~~	~~0,68~~
Immob	2,02	0,99	0,97	0,2	1,20

Alternatives et résultat

Angle : 5°	Glider	Canard	Becassine	Suzanne	Blackburn
~~Blanc 1~~	~~0,86~~	~~X~~	~~0,87~~	~~1,11~~	~~1,45~~
~~Calque c~~	~~1,44~~	~~0,81~~	~~0,88~~	~~0,55~~	~~1,52~~
~~Calque f~~	~~1,07~~	~~1,15~~	~~1,08~~	~~1,31~~	~~1,69~~
~~Chews~~	~~X~~	~~0,75~~	~~2,51~~	~~0,55~~	~~X~~
Midil	1,12	0,99	X	1,30	X
~~Ikea~~	~~0,85~~	~~0,82~~	~~0,55~~	~~2,16~~	~~0,82~~
Immob	4,59	1,00	0,08	0,95	1,40

Angle : 20°	Glider	Canard	Becassine	Suzanne	Blackburn
~~Blanc c~~	~~X~~	~~0,25~~	~~X~~	~~0,74~~	~~0,51~~
~~Calque c~~	~~1,16~~	~~X~~	~~0,28~~	~~0,59~~	~~X~~
~~Calque f~~	~~X~~	~~X~~	~~0,81~~	~~0,68~~	~~1,54~~
~~Chews~~	~~1,55~~	~~0,12~~	~~0,16~~	~~0,58~~	~~X~~
Midil	0,98	X	4,80	1,40	X
~~Ikea~~	~~X~~	~~1,06~~	~~1,11~~	~~0,05~~	~~0,68~~
Immob	2,02	0,99	0,97	0,2	1,20

Alternatives et résultat

Angle : 5°	Glider	Canard	Becassine	Suzanne	Blackburn
Blanc 1	0,80	X	0,97	1,11	1,45
Calque e	1,44	0,61	0,66	0,55	1,52
Calque f	1,07	1,15	1,08	1,31	1,63
Chews	X	0,75	2,51	0,55	X
Midli	1,12	0,99	X	1,30	X
Ikea	0,85	0,62	0,55	2,16	0,62
Immob	4,59	1,00	0,08	0,95	1,40

Angle : 20°	Glider	Canard	Becassine	Suzanne	Blackburn
Blanc e	X	0,25	X	0,74	0,51
Calque e	1,16	X	0,28	0,59	X
Calque f	X	X	0,81	0,68	1,54
Chews	1,55	0,12	0,16	0,58	X
Midli	0,56	X	4,60	1,40	X
Ikea	X	1,66	1,11	0,05	0,66
Immob	2,02	0,99	0,97	0,2	1,20

Conclusion

Listing de mes codes

- Exploitation statistique de l'expérience du gymnase et test gaussien
- Code commun aux tests des détections de contours et à l'algorithme principal
- Comparaison de méthodes de détections de contours
- Algorithme principal: les fonctions pour traiter une image puis le film complet
- Fonctions annexes pour visualiser la trajectoire et calculer un coefficient de corrélation
- Interface utilisateur

```
001| #### import des modules
002| import matplotlib.pyplot as plt
003| import matplotlib.image as mpimg
004| import numpy as np
005| from pylab import *
006| from scipy.signal import savgol_filter #pour lissage des données
007| import scipy.stats # pour l'expérience du gymnase
008| # pour les interfaces utilisateurs sous Tkinter
009| from tkinter import *
010| from tkinter import ttk
011| from tkinter.messagebox import *
012| from tkinter import font
```

```
026| ## test Gaussienne - comparaison courbe et histogramme
027| x=np.linspace(0,12.0,100)
028| plt.plot(x,scipy.stats.norm.pdf(x,5.9,2.1))
029| plt.grid()
030| plt.xlim(0,12)
031| plt.ylim(0,0.4)
032| plt.hist(resultats1et2,bins=10, normed=1)
033| plt.show()
034|
035| ## test Gaussienne - comparaison des fonctions de répartition
036| x=np.linspace(0,12.0,100)
037| plt.plot(x,scipy.stats.norm.cdf(x,5.9,2.1), 'black')
038| plt.grid()
039| plt.xlim(0,12)
040| plt.ylim(0,1)
041| plt.hist(resultats1et2,bins=100,cumulative = True, normed=1, color='orange')
042| plt.show()
```

```
049| # couper l'image m entre les lignes dl et fl et entre les colonnes dc et df et ne
garde que la première composante de la matrice
050| def couper_image(m,dc,fc,dl,fl):
051|     new_m=[]
052|     for i in range(fl-dl+1):
053|         inte=[]
054|         for j in range(fc-dc+1):
055|             inte.append(m[i+dl][j+dc][0])
056|         new_m.append(inte)
057|     return(new_m)
058|
059| # pour seuiller l'image en deux couleurs
060| def seuillage(m,seuil):
061|     m1=[]
062|     for i in range(len(m)):
063|         m2=[]
064|         for j in range(len(m[0])):
065|             if m[i][j]>seuil:
066|                 m2.append(1)
067|             else:
068|                 m2.append(0)
069|         m1.append(m2)
070|     return(m1)
071|
072| # pour passer de la matrice de travail à une matrice format image png
073| def retour_mat_image(m):
074|     new_m=[]
075|     nbr_col=len(m[0])
076|     nbr_lign=len(m)
077|     for i in range(nbr_lign):
078|         inte=[]
079|         for j in range(nbr_col):
080|             inte.append([m[i][j],m[i][j],m[i][j],1])
081|         new_m.append(inte)
082|     return(new_m)
```

```
183| # cherche les colonnes limites droite et gauche de l'avion
184| def chercher_limite(m):
185|     ll=len(m)
186|     lc=len(m[0])
187|     mini=lc-1
188|     maxi=0
189|     for i in range(ll):
190|         for j in range(lc):
191|             if m[i][j]==1:
192|                 if j<mini:
193|                     mini=j
194|                 if j>maxi:
195|                     maxi=j
196|     return([mini,maxi])
197|
198| # même principe mais cherche les limites "haute et basse" dans une colonne donnée
199| def chercher_haut_bas(m,c):
200|     min=len(m[0])-1
201|     max=0
202|     for i in range(len(m)):
203|         if m[i][c]==1:
204|             if i<min:
205|                 min=i
206|             if i>max:
207|                 max=i
208|     return([min,max])
209|
210| # On a deux version de la fonction traitement image
211| # version 1
212|
213| # Donne une approximation du centre de masse de l'avion, pour cela l'utilisateur
rentre une valeur 0<p<1 qui donne la place du centre de masse de l'avion (si p=0 il
est en min, si p=1 il est en max et sinon proportion de longueur)
214| def centre_de_masse(m,p):
215|     tl=chercher_limite(m)
216|     cdm_colonne=tl[1]*p+(1-p)*tl[0]
217|     tc=chercher_haut_bas(m,tl[0])
218|     cdm_ligne=int((tc[1]+tc[0])/2)
219|     return([cdm_ligne,int(cdm_colonne)])
220| # fonctionnement: on calcule les limites de l'avion (colonnes et lignes) puis on
en déduit approximativement la colonne de son centre de masse. Dans cette colonne on
cherche alors les limites de l'avion et on moyenne. On obtient ainsi les coordonnées
du centre de masse de l'avion en pixel
```

```python
# fonction qui regroupe tout
def traitement_image_1(link,k,c,dc,fc,dl,fl,s):
    #k est le numero de l'image a traité, entre 100 et 999, c représente la proportion pour le centre de masse, dc et fc (resp dl et fl) les délimitations des colonnes (resp des lignes) à traiter et s le seuil à utiliser
    lien=link+str(k)+".png"
    i0=mpimg.imread(lien)
    i1=couper_image(i0,dc,fc,dl,fl)
    i2=seuillage(i1,s)
    cdm=centre_de_masse(i2,c)
    abscimax=chercher_limite(i2)[1]
    colomax=(chercher_haut_bas(i2,abscimax))[0]
    t2=arctan((cdm[0]-colomax)/(abscimax-cdm[1]))
    return([cdm,t2]) # en coordonnées pixel

# version 2
def traitement_image(link,k,c,dc,fc,dl,fl,s):
    lien=link+str(k)+".png"
    i0=mpimg.imread(lien)
    i1=couper_image(i0,dc,fc,dl,fl)
    i2=seuillage(i1,s)
    n=len(i2)
    m=len(i2[0])
    x=0
    x0=0
    y=0
    y0=0
    nbr_val=0
    for i in range(n):
        for j in range(m):
            if i2[i][j]==1:
                x0+=i
                y0+=j
                nbr_val+=1
                if i>x:
                    x=i
                    y=j
                elif i==x and j>y:
                    y=j
    x0=int(x0/nbr_val)
    y0=int(y0/nbr_val)
    angle=arctan((y-y0)/(x-x0))
    return([[x0,y0],angle]) # renvoie les coordonnées du centre de masse et alpha en rad
```

```python
# calcul des vitesses
def vitesses(t,dt,dl): #s'applique sur une liste d'une seule composante. Donc sur liste_cdm_y OU sur liste_cdm_x
    n=len(t)
    t_v=[]
    for k in range(n-1):
        vk=(t[k+1]-t[k])*dl/dt
        t_v.append(vk)
    return(t_v) # en m/s

# calcul des vitesses et des accélérations d'une liste à une seule composante
def vit_acc(t,dt,dl):
    t_v=vitesses(t,dt,dl)
    n=len(t_v)
    t_a=[]
    for k in range(n-1):
        ak=(t_v[k+1]-t_v[k])/dt
        t_a.append(ak)
    return((t_v),(t_a)) # en m/s²

# calcul de l'angle entre le sol et l'avion
def calcul_TO(tvx,tvy):
    n=len(tvx)
    t_TO=[]
    for k in range(n):
        if tvx[k]==0 and tvy[k]>=0:
            t_TO.append(pi/2)
        elif tvx[k]==0 and tvy[k]<0:
            t_To.append(-pi/2)
        else:
            t_TO.append(arctan(tvy[k]/tvx[k]))
    return(t_TO)  # en rad car les fonctions python sont faites comme cela. On changera l échelle plus tard pour l affichage

# calcul de Ft, Fp pour tout les instants d'étude
def calcul_ft_fp(tx,ty,dt,dl):
    ft_list=[]
    fp_list=[]
    [tvx,tax]=vit_acc(tx,dt,dl)
    [tvy,tay]=vit_acc(ty,dt,dl)
    To=calcul_TO(tvx,tvy)
    nt=min(len(tax),len(tay))
    for i in range(nt):
        ft_list.append(abs(m*tax[i]*cos(To[i])+m*(g+tay[i])*sin(To[i])))
        fp_list.append(abs(-m*tax[i]*cos(To[i])+m*tay[i]*sin(To[i])))
    return(ft_list,fp_list)

def calcul_ct_cp(tx,ty,ft,fp,rho,surface,dt,dl):
    [tvx,tax]=vit_acc(tx,dt,dl)
    [tvy,tay]=vit_acc(ty,dt,dl)
    ct_list=[]
    cp_list=[]
    for i in range(len(ft)):
        ct_list.append(2*ft[i]/(rho*surface*(tvx[i]**2+tvy[i]**2)))
        cp_list.append(2*fp[i]/(rho*surface*(tvx[i]**2+tvy[i]**2)))
    return(ct_list,cp_list)

# calcul écart valeurs pointées et modélisation
def coef_correlation(x,y):
    n=len(x)
    xy=[]
    xx=[]
    yy=[]
    for i in range(n):
        xy.append(x[i]*y[i])
        xx.append(x[i]*x[i])
        yy.append(y[i]*y[i])
    ex=sum(x)/n
    ey=sum(y)/n
    exy=sum(xy)/n
    exx=sum(xx)/n
    eyy=sum(yy)/n
    covxy=exy-ex*ey
    ecartype_x=np.sqrt(exx-ex*ex)
    ecartype_y=np.sqrt(eyy-ey*ey)
    return(covxy/(ecartype_x*ecartype_y))
```

```
375| # Les codes qui vont suivre nécessitent un éclairage constant (type lumière
naturelle)
376| ## calcule les listes des abscisses (x) et  des ordonnées (y)
377| list_cdm_x=[]
378| list_cdm_y=[]
379| for k in range(j-i):
380|     trait=traitement_image(link,k+i,c,dc,fc,dl,fl,s)[0]
381|     list_cdm_y.append(fl-trait[0])
382|     list_cdm_x.append(trait[1])
383| # permet de tracer les abscisses en fonction du temps et de lisser les données
384| list_cdm_x_2= savgol_filter(list_cdm_x, n, 3)# window size 51, polynomial order 2
385| # permet de tracer les ordonnées en fonction du temps et de lisser les données
386| list_cdm_y_2 = savgol_filter(list_cdm_y, n, 3)# window size 51, polynomial order
3
387| # calcul et courbes des coefficients avec lissage
388| cft,cfp=calcul_ft_fp(list_cdm_x_2,list_cdm_y_2)
389|
390| plt.plot(cft,cfp)
391| plt.show()
392| print(coef_correlation(list_cdm_x,list_cdm_x_2))
```

```
795|  # etape 4, pointage et modélisation
796|  list_cdm_x=[]
797|  angle=[]
798|  list_cdm_y=[]
799|  for k in range(j-i):
800|      trait=traitement_image(link,k+i,c,dc,fc,dl,fl,s)
801|      list_cdm_y.append(fl-trait[0][0])
802|      list_cdm_x.append(trait[0][1])
803|      angle.append(trait[1]*180/pi)
804|
805|  list_cdm_x_2= savgol_filter(list_cdm_x,n,3)
806|  list_cdm_y_2 = savgol_filter(list_cdm_y,n,3)
807|
808|  if (coef_correlation(list_cdm_x,list_cdm_x_2))>0.99 and
(coef_correlation(list_cdm_y,list_cdm_y_2)):
809|      ftl,fpl=calcul_ft_fp(list_cdm_x_2,list_cdm_y_2,dt_im,dl_im)
810|
cft,cfp=calcul_ct_cp(list_cdm_x_2,list_cdm_y_2,ftl,fpl,rho,s_aile,dt_im,dl_im)
811|      cfp2=[]
812|      cft2=[]
813|      for k in range(len(cfp)):
814|          cfp2.append(cfp[k])
815|          cft2.append(-cft[k])
816|      [tvx,tax]=vit_acc(list_cdm_x_2,dt_im,dl_im)
817|      [tvy,tay]=vit_acc(list_cdm_y_2,dt_im,dl_im)
818|      To=calcul_T0(tvx,tvy)
819|      To2=[]
820|      for k in range(min(len(To),len(angle))-1):
821|          a=abs(angle[len(angle)-(k+1)]-To[len(To)-(k+1)]*180/pi)
822|          if a>=0 and a<90:
823|              To2.append(abs(angle[len(To)-(k+1)]-To[len(To)-(k+1)]*180/pi))
824|          else:
825|               To2.append(abs(angle[len(To)-(k+1)]-To[len(To)-(k+1)]*180/pi-90))
826|      To2=savgol_filter(To2,len(To2),3)
827|  else:
828|       showinfo("Alerte", 'problème de modélisation merci de reessayer ou d executer
manuellement le code')
```