

Numéro d'inscription

40677

thème :La ville



Stabilité par  
effet dynamique  
grâce à un  
volant d'inertie



DAMMAK lyed  
en collaboration avec  
-METTALI Amine

# Enjeu :

- Aider les petits enfants à tenir l'équilibre en vélo
- Eviter/amortir les chutes des conducteurs



# Problématique:

- 
- 

Comment peut-on éviter des chutes brusques ?  
Comment peut-on assurer l'équilibre d'un vélo ?

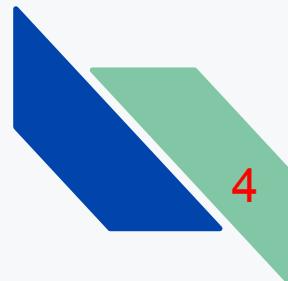
## Réponse:

Utiliser une roue inertielle asservie





## Plan:

- Présentation du problème et mise en place d'un modèle
  - Étude dynamique
  - Caractérisation du système
  - Simulation du modèle
  - Conclusion
- 

# I. Présentation du problème et mise en place d'un modèle

## I. 1- Observation:

Simplification du problème:

vélo en état statique



deux liaisons ponctuelles



Liaison cylindre plan

Liaison pivot

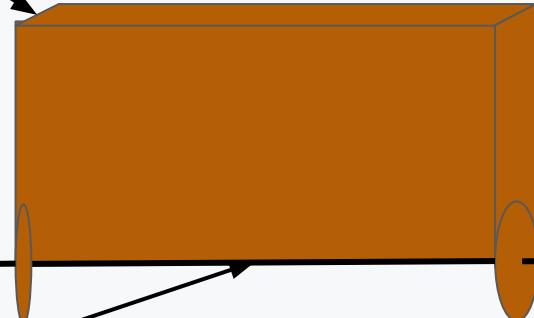


## I. 2- Conception :

système réel

vélo / plaque

modèle équivalent

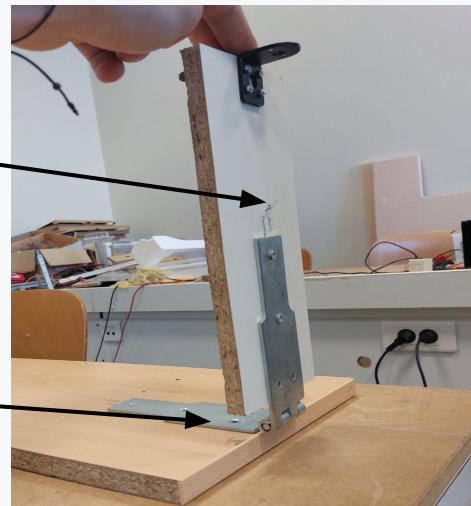


liaison pivot

bâti

plaque

charnière

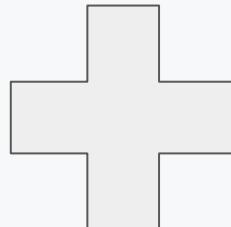


## I. 3- Mise en place d'un dispositif d'asservissement :



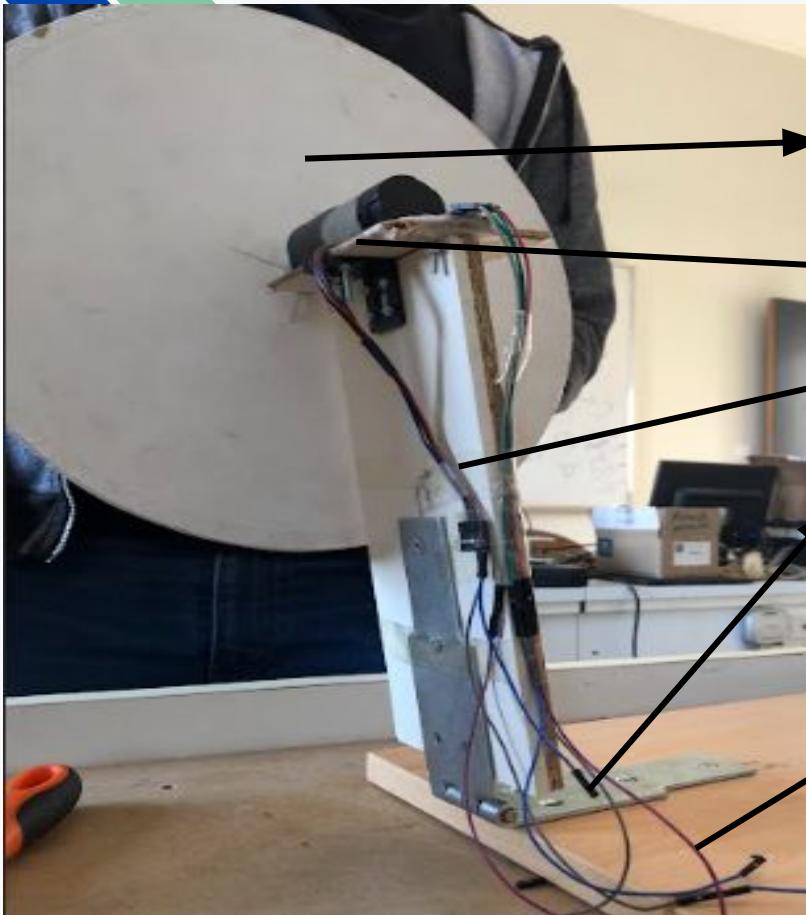
mise en place de la roue inertielle

une roue inertielle



comander la roue

# Modèle expérimental



constituants

roue inertie

moteur

plaque

charnière

bâti

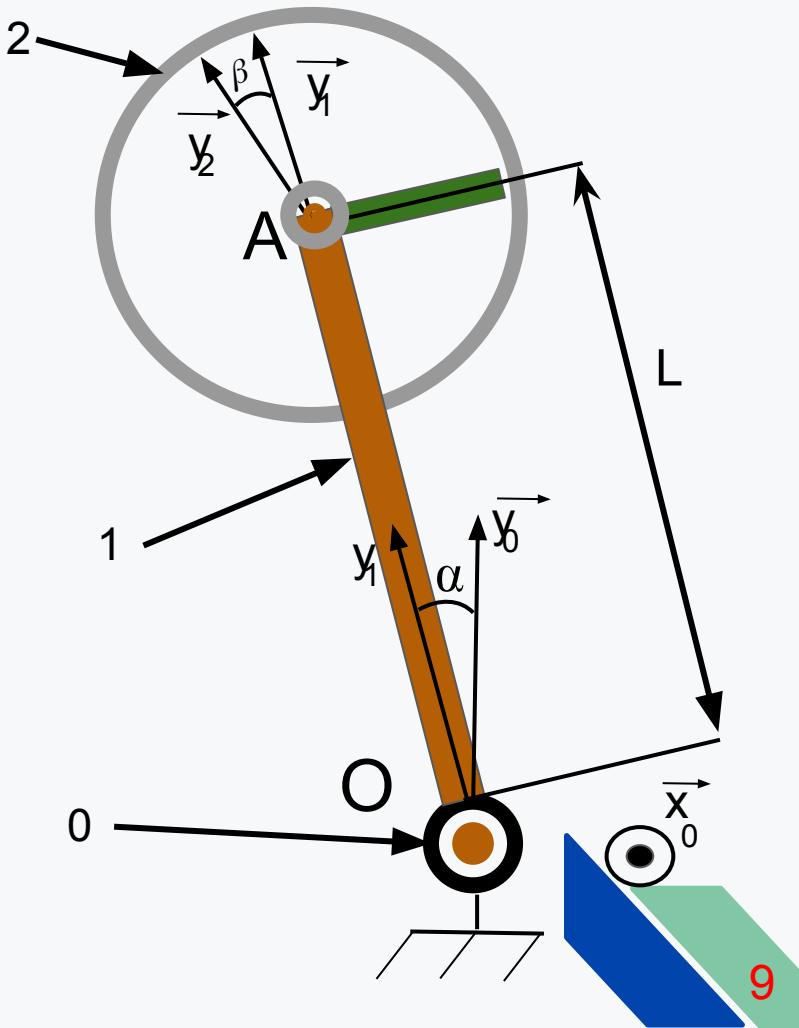
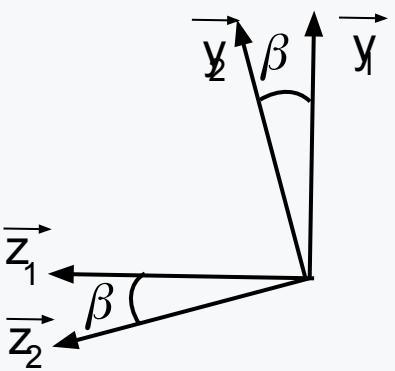
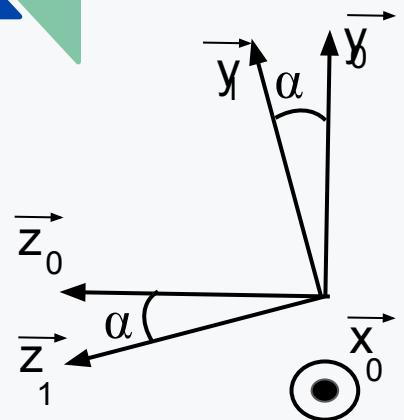
Classes d'  
équivalence

2

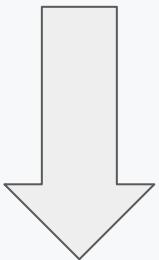
1

0

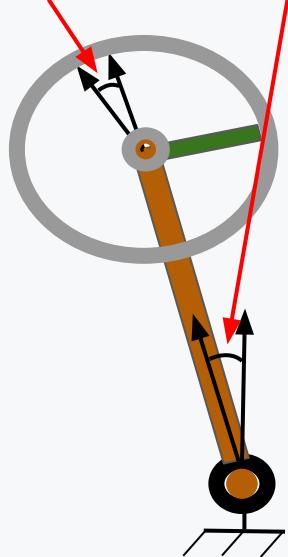
## Paramétrage



Commander le moteur directement !!!



commander  $\beta$  pour asservir  $\alpha$

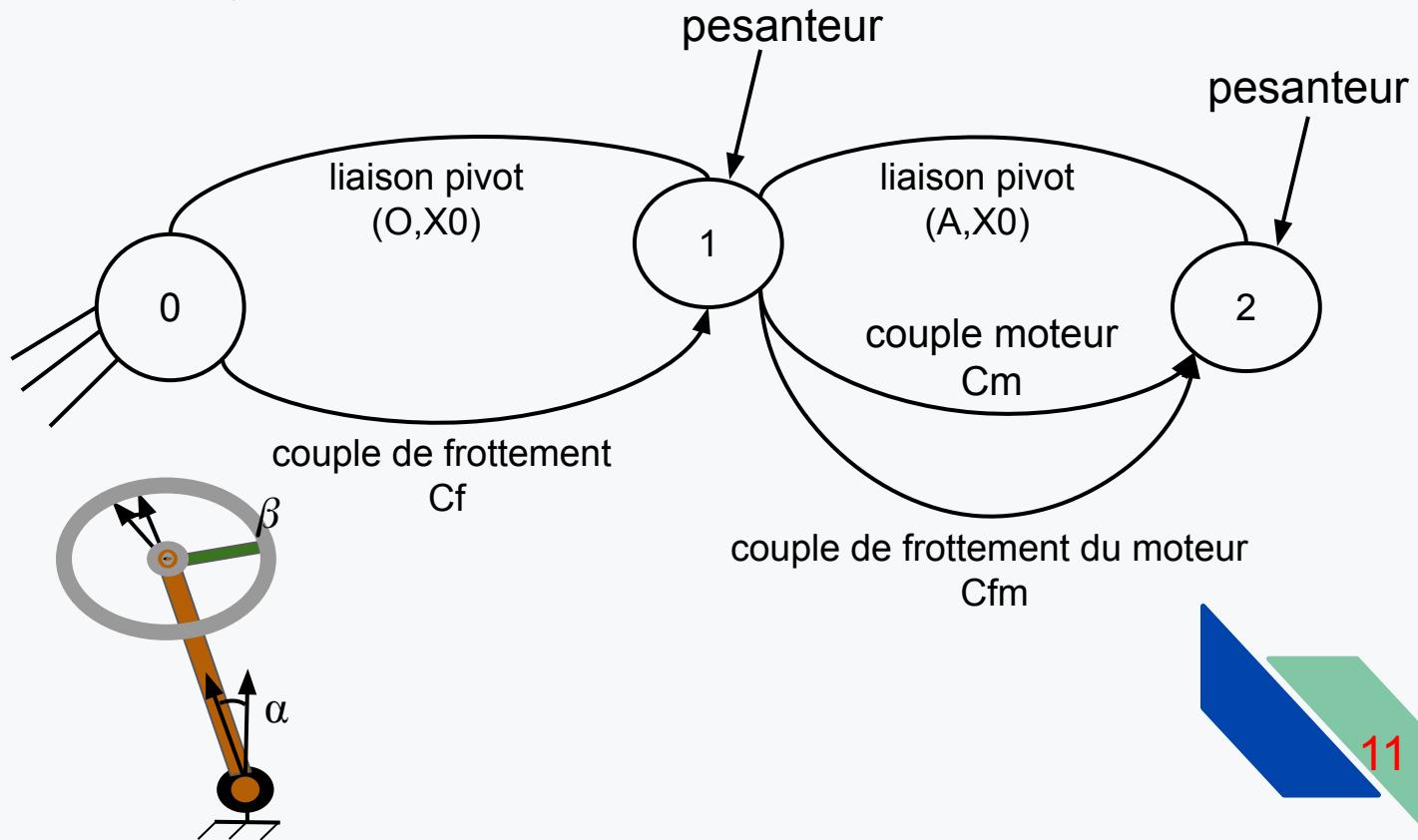


étudier et caractériser le système

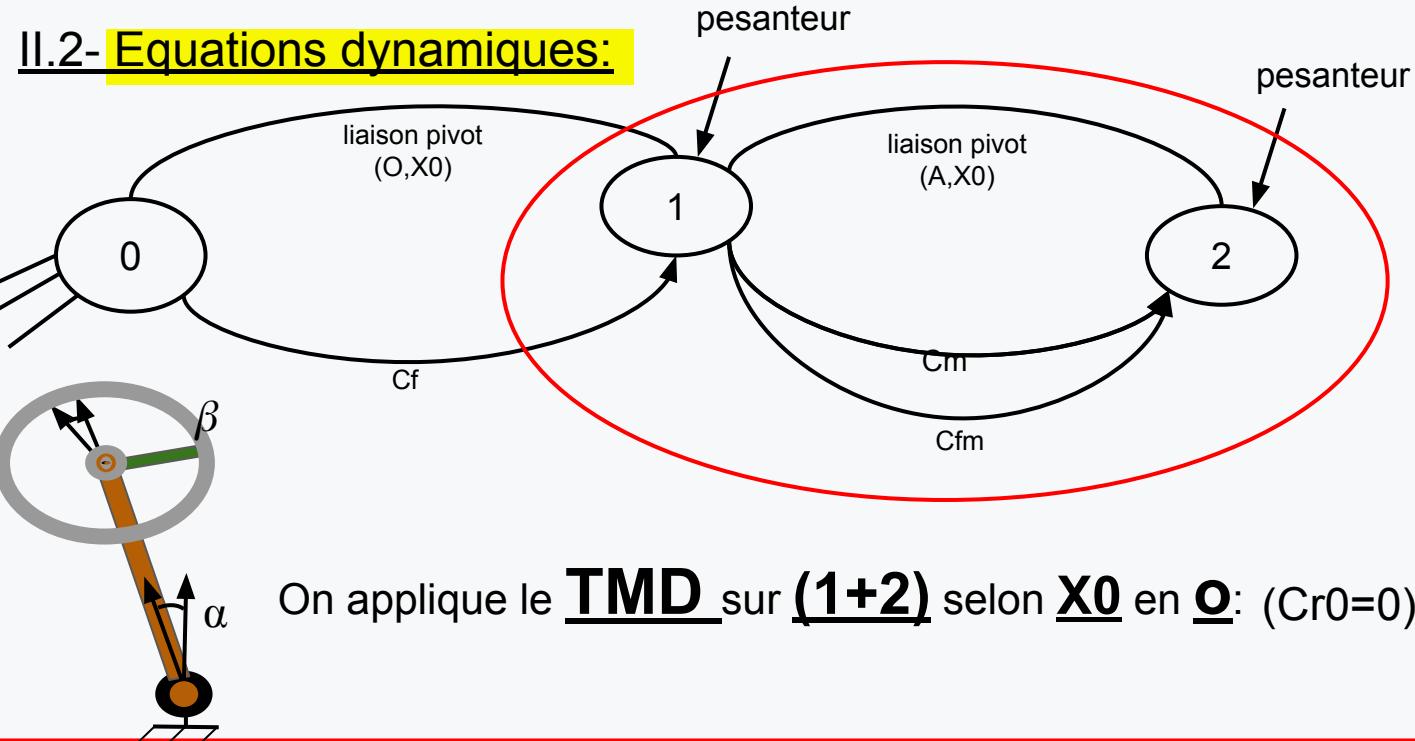
D'où le besoin de faire une étude plus élaborée

## II. Etude dynamique

### II.1- Graphe de liaison:



## II.2- Equations dynamiques:

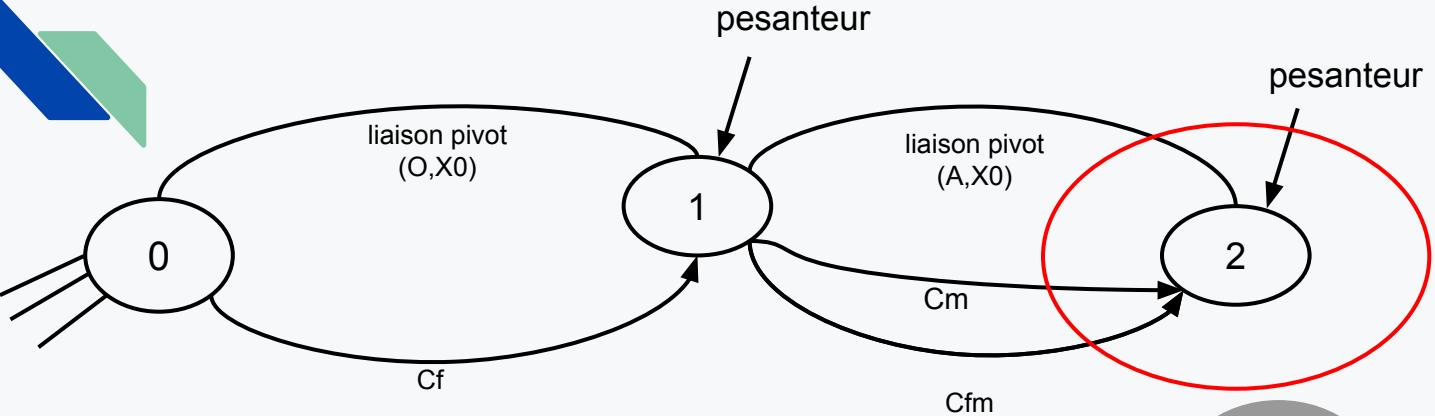


On applique le **TMD** sur **(1+2)** selon **X0** en **O**: ( $C_{r0}=0$ )

$$A_1 \frac{d^2\alpha}{dt^2} + A_2 \left( \frac{d^2\alpha}{dt^2} + \frac{d^2\beta}{dt^2} \right) + m_2 L^2 \left( \frac{d^2\alpha}{dt^2} \right) = g \sin(\alpha) (L m_2 + l_{g1} m_1) - f \frac{d\alpha}{dt}$$



$A_1, A_2, L, m_2, m_1, l_{g1}, f$



On applique le **TMD** sur **2** selon **X0** en **A**:  $(Cr0m=0)$

$$A_2 \left( \frac{d^2\alpha}{dt^2} + \frac{d^2\beta}{dt^2} \right) = C_m - f_m \frac{d\beta}{dt}$$

fm



## II. 3- Equations du moteur:

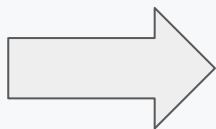
$$U = E + R_m i + L_m \frac{di}{dt}$$

$$E = K \frac{d\beta}{dt}$$

$$C_m = K i$$

$$A_2 \frac{d^2\beta}{dt^2} = C_m - C_r$$

$$C_r = C_{r0m} + f_m \frac{d\beta}{dt}$$



R<sub>m</sub>, L<sub>m</sub>, K, f<sub>m</sub>, C<sub>r0m</sub>

### III. Caractérisation du système

#### III. 1- Point méthodologique

##### Constantes

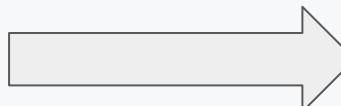
##### Expériences

$L, m_2, m_1$



mesures directes

$A_2$



Calcul

$lg_1$



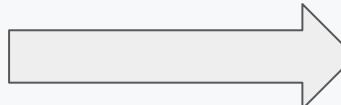
Programme informatique

$A_1, f, Cr_0$



pendule inverse

$K, R_m, L_m, f_m, Cr_{0m}$

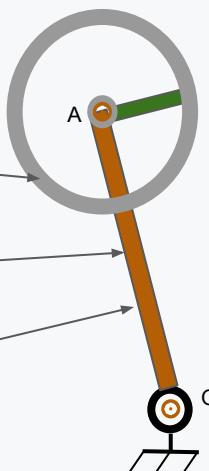


manipuler le moteur

### III.2- Phase expérimentale :

mesures directes

$$m_2 = 243.55 \pm 0.05 \text{ g}$$

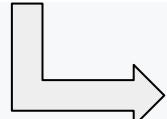


$$m_1 = 669.84 \pm 0.05 \text{ g}$$

$$L = 26.25 \pm 0.5 \text{ Cm}$$

Programme informatique

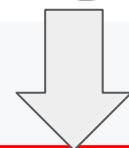
$$\overline{OG} = \frac{1}{\sum m_i} \sum_i m_i \overline{OG}_i$$



La roue est parfaitement homogène



$$A_2 = m_2 \times \frac{R^2}{2}$$



$$A_2 = 0.0033153 \pm 0.0003 \text{ Kg.m}^2$$

$$OG1.Y1 = lg1 = 17.52 \text{ Cm}$$

# A1, f,Cr0 : expérience de la pendule

$$A_1 \frac{d^2 \alpha}{dt^2} = \sin(\alpha) g l_{g1} m_1 - f \frac{d\alpha}{dt} - C_{r0}$$

$\alpha$  oscille autour de  $\pi$  ;

$$\tilde{\alpha} = \pi - \alpha$$



Tout calcul et simplifications faites:

$$\frac{d^2 \tilde{\alpha}}{dt^2} + \frac{f}{A_1} \frac{d\tilde{\alpha}}{dt} + \tilde{\alpha} \frac{g l_{g1} m_1}{A_1} = C_{r0} \rightarrow$$

$$w_0 = \sqrt{\frac{m_1 g l_{g1}}{A_1}}$$

$$\xi = \frac{f}{2w_0 A_1}$$

$\tilde{a}$  tend vers 0 en plus infini donc  $C_0=0$

### Objectif:

Avoir les variation de  $\alpha$  pour vérifier la validité de l'équation

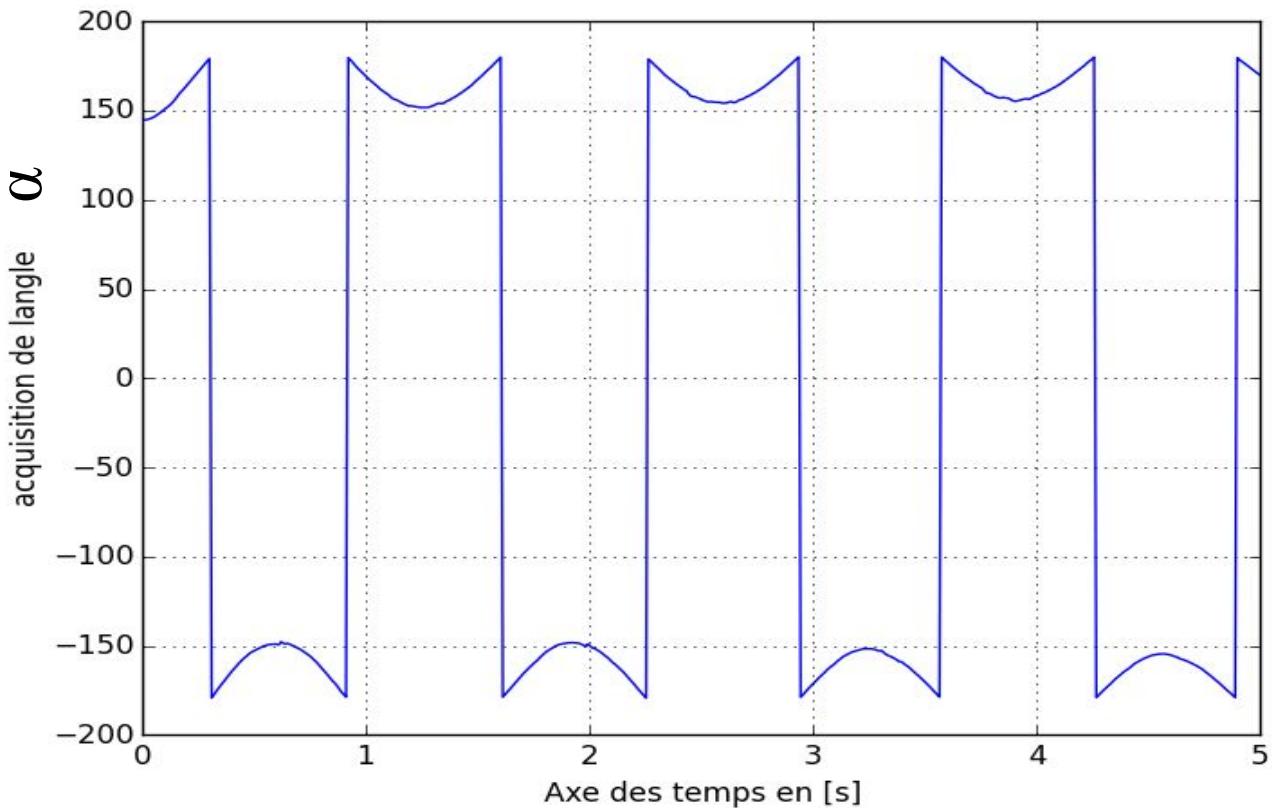


Placer une centrale inertie:  
durée d'acquisition:  $dt=10$  ms



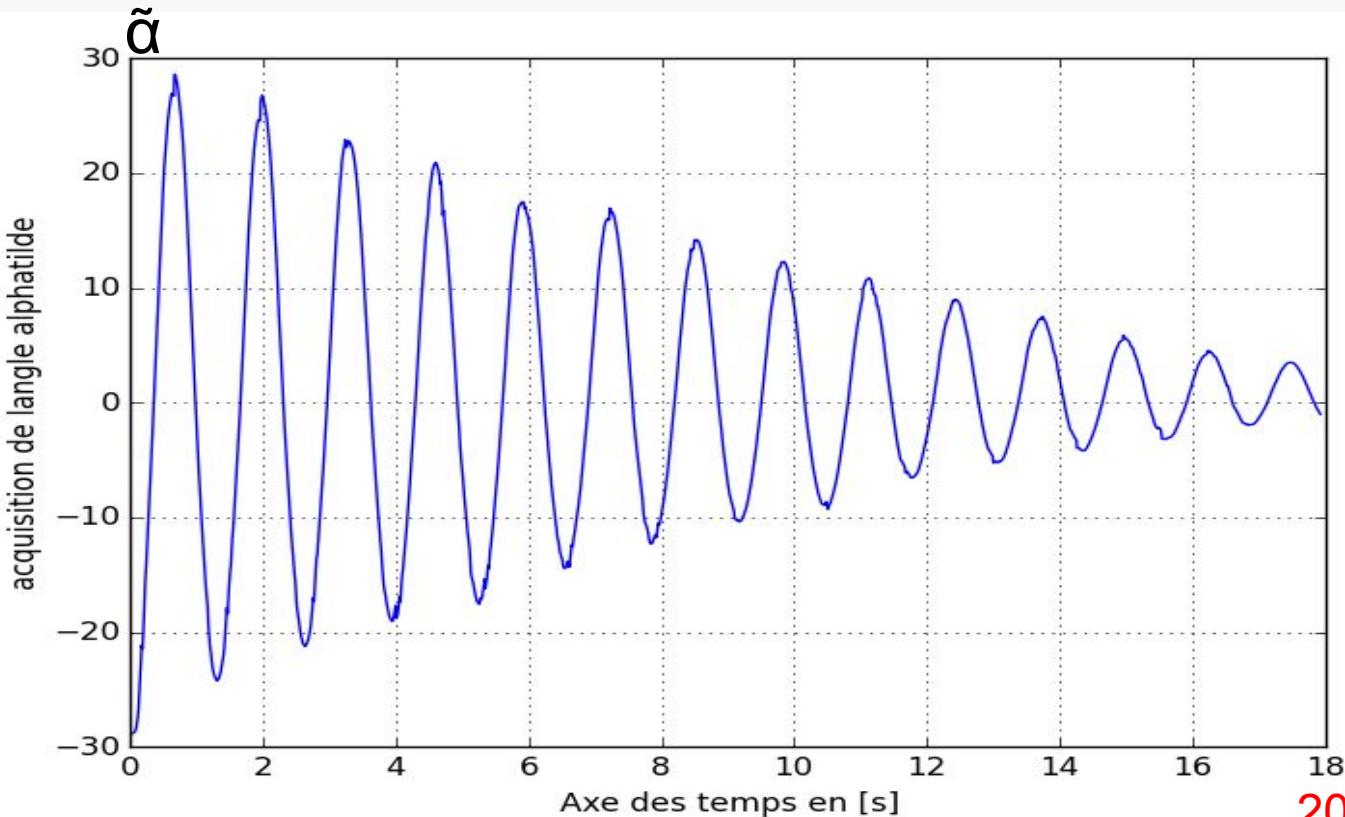
$$=\tilde{\alpha}$$

## Première acquisition



# Courbe traitée

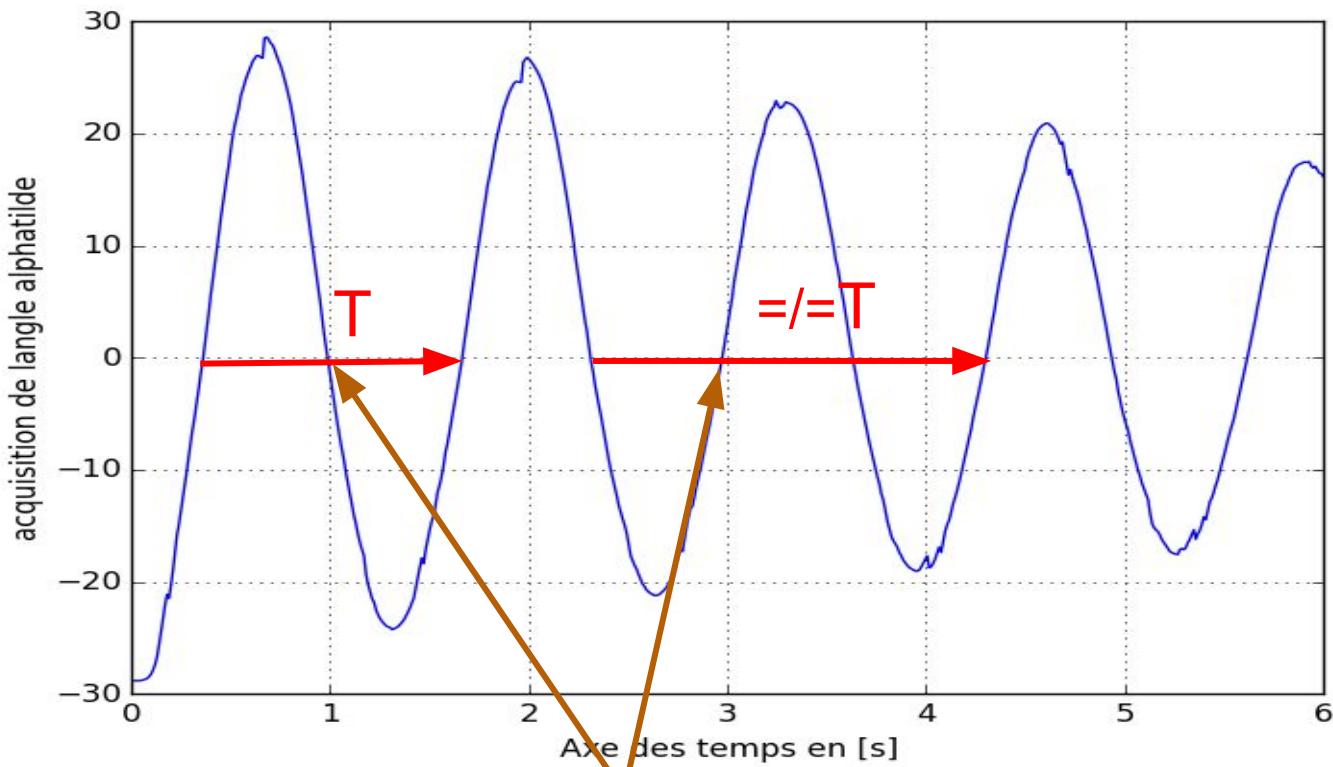
La fonction “traiter\_courbe” donne  $\tilde{\alpha}$  directement





la période: T

l'enveloppe des oscillations:  
 $y=B \cdot \exp(-\xi \cdot W_0 \cdot t)$



déetecter les 0  
puis calculer  $T$

prendre les valeurs cohérentes ?

Programme periode dans l'annexe 7

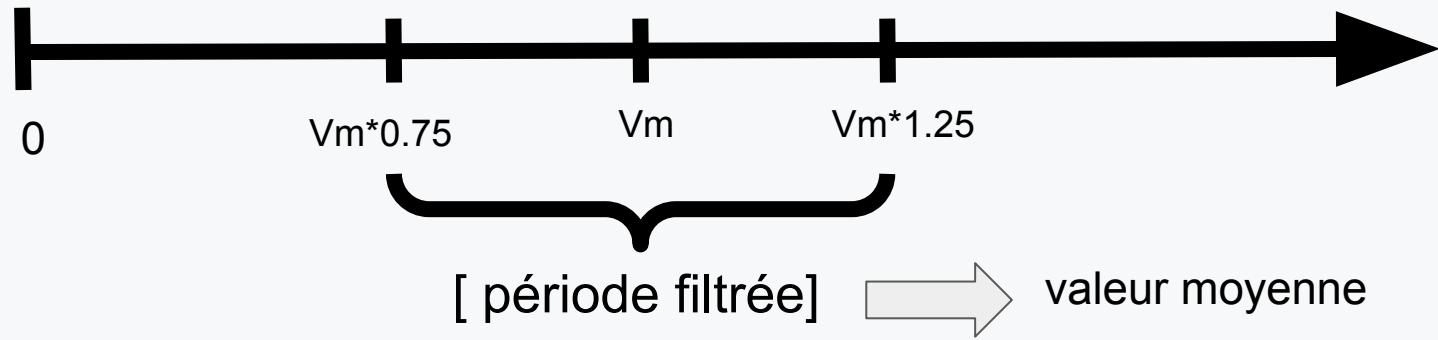


On stocke ces valeurs dans la liste période= []



Certaines valeurs ne sont pas logiques → valeur moyenne:  $V_m$

filtre logique



T=1.3s

enveloppe



déetecter les sommets



appliquer une régression

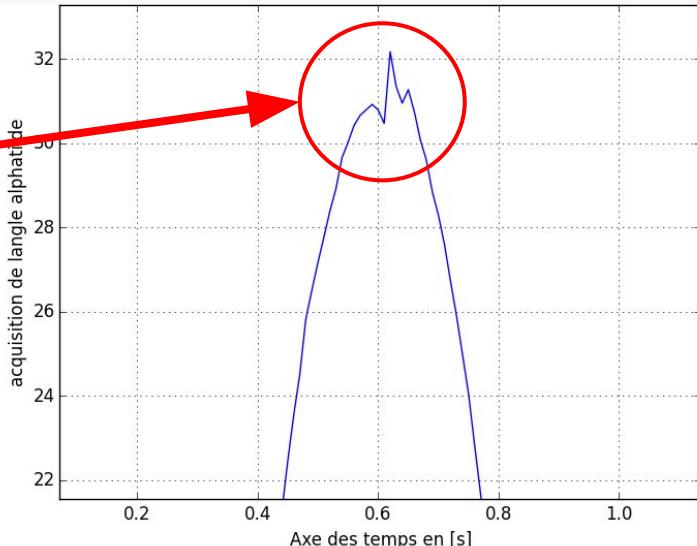
bruit



filtre passe bas  
(  $W_c=3*Wr$  )

$$H(jw) = \frac{S}{E} = \frac{1}{1+j\frac{w}{3*Wr}}$$

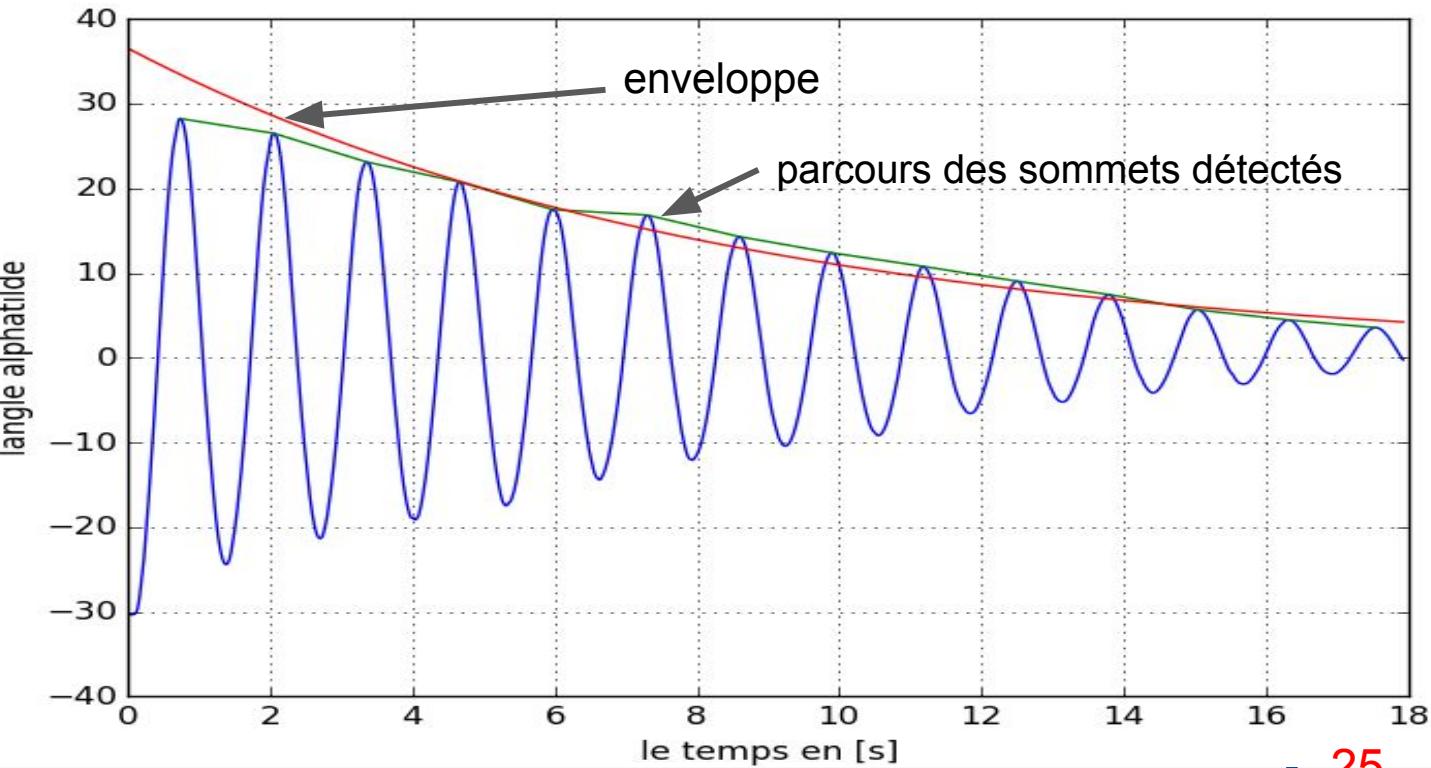
$$\frac{dS}{dt} \frac{1}{3Wr} + S = E$$



Méthode d'Euler

programme filtrercourbes dans l'annexe 7

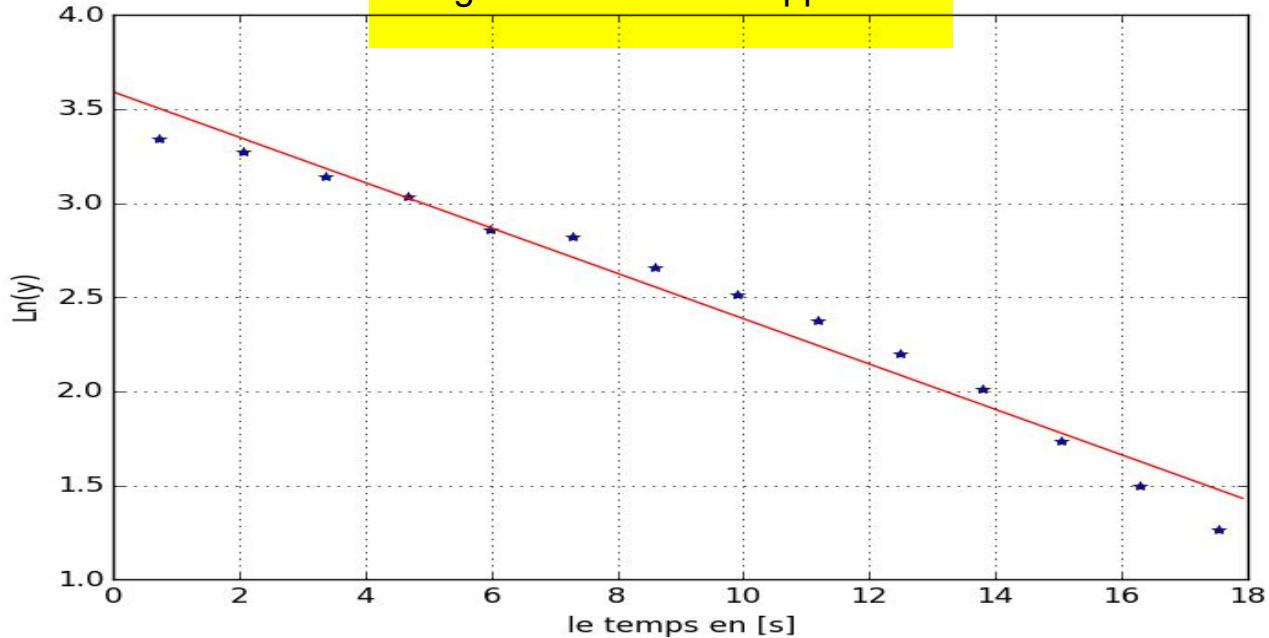
## Le programme courbe\_et\_enveloppe donne



25

programme courbe\_et\_enveloppe dans l'annexe 7

## régression de l'enveloppe

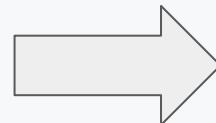


```
In [50]: courbe_et_enveloppe(Lle[:1700])
Out[50]: 0.11445832564549079
```



$$\xi^* W_0 = 0.1146 \pm 0.08 \text{ s}^{-1}$$

On résout le système à deux inconnus



$$A1=0.04617 \pm 0.004 \text{ Kg.m}^2$$
$$f=0.011 \pm 0.002 \text{ Kg.m}^2.\text{s}^{-1}$$

# Déterminer : K , Rm, Lm, fm,Cr0

manipuler le moteur

$$U = E + R_m i + L_m \frac{di}{dt}$$

$$E = K \frac{d\beta}{dt}$$

$$C_m = K i$$

$$A_2 \frac{d^2\beta}{dt^2} = C_m - C_r$$

$$C_r = C_{r0m} + f_m \frac{d\beta}{dt}$$

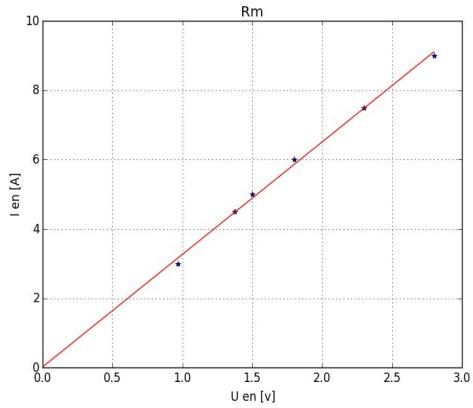
neutraliser le terme à identifier



acquisition des valeurs

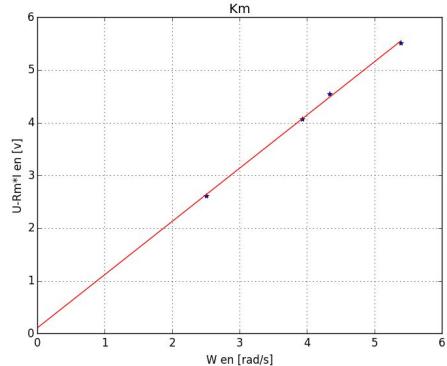
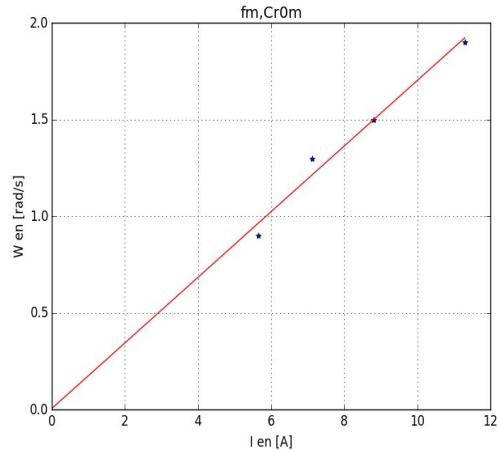


régression linéaire



$R_m = 3.2 \pm 0.18 \Omega$

```
In [82]: Rmp(I2,U)
Out[82]: 3.2785304469408416
```



$K = 1.04 \pm 0.14 \text{ V.s/rad}$

```
In [89]: Km(wk,uk)
Out[89]: 1.0424443891719259
```



relation quasi-linéaire:  
 $Cr0m=0$

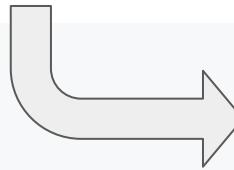
$fm = 0.18 \pm 0.04 \text{ Kg.m}^2.\text{s}^{-1}$

```
In [93]: fm_Crm(W,I1)
Out[93]: (0.17689610772347111, 0.0057783796230724254)
```

$Lm = 1.900 \pm 0.003 \text{ mH}$

## Simplification des équations

$$A_1 \frac{d^2\alpha}{dt^2} + A_2 \left( \frac{d^2\alpha}{dt^2} + \frac{d^2\beta}{dt^2} \right) + m_2 L^2 \left( \frac{d^2\alpha}{dt^2} \right) = g \sin(\alpha) (L m_2 + l_{g1} m_1) - f \frac{d\alpha}{dt}$$



$$\hat{A}_1 \frac{d^2\alpha}{dt^2} + A_2 \frac{d^2\beta}{dt^2} = \alpha C_{eq} - f \frac{d\alpha}{dt} \quad (1)$$

$$A_2 \left( \frac{d^2\alpha}{dt^2} + \frac{d^2\beta}{dt^2} \right) = C_m - f_m \frac{d\beta}{dt} \quad (2)$$

$$U = E + R_m i + L_m \frac{di}{dt} \quad (3)$$

$$E = K \frac{d\beta}{dt}$$

$$C_m = K i$$



$$U = K \frac{d\beta}{dt} + \frac{R_m}{K} C_m + \frac{L_m}{K} \frac{dC_m}{dt}$$

## IV. Simulation du modèle:

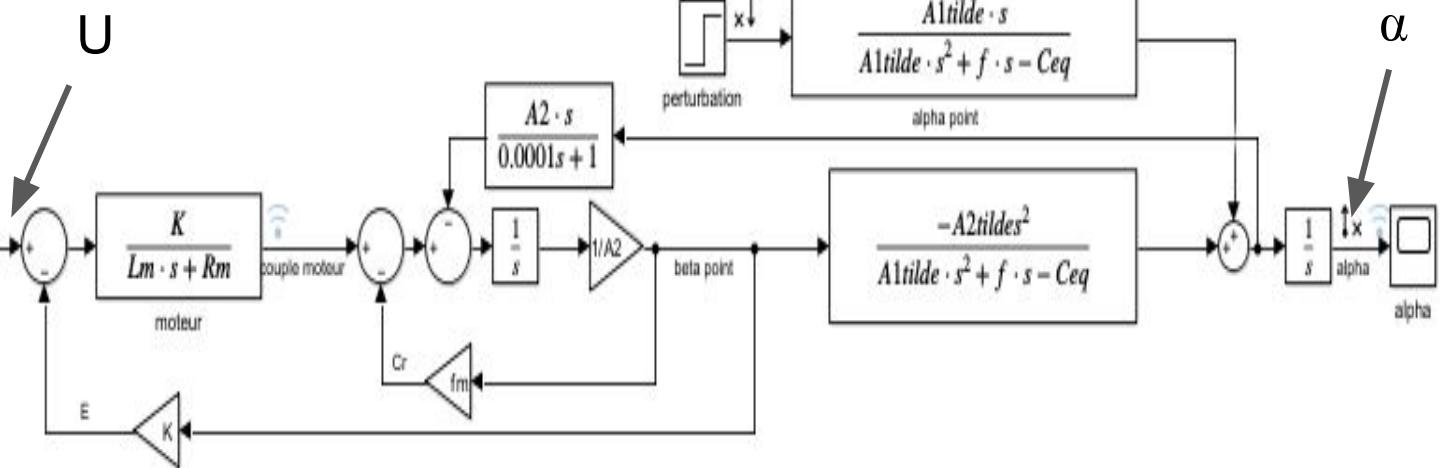
→ domaine de Laplace

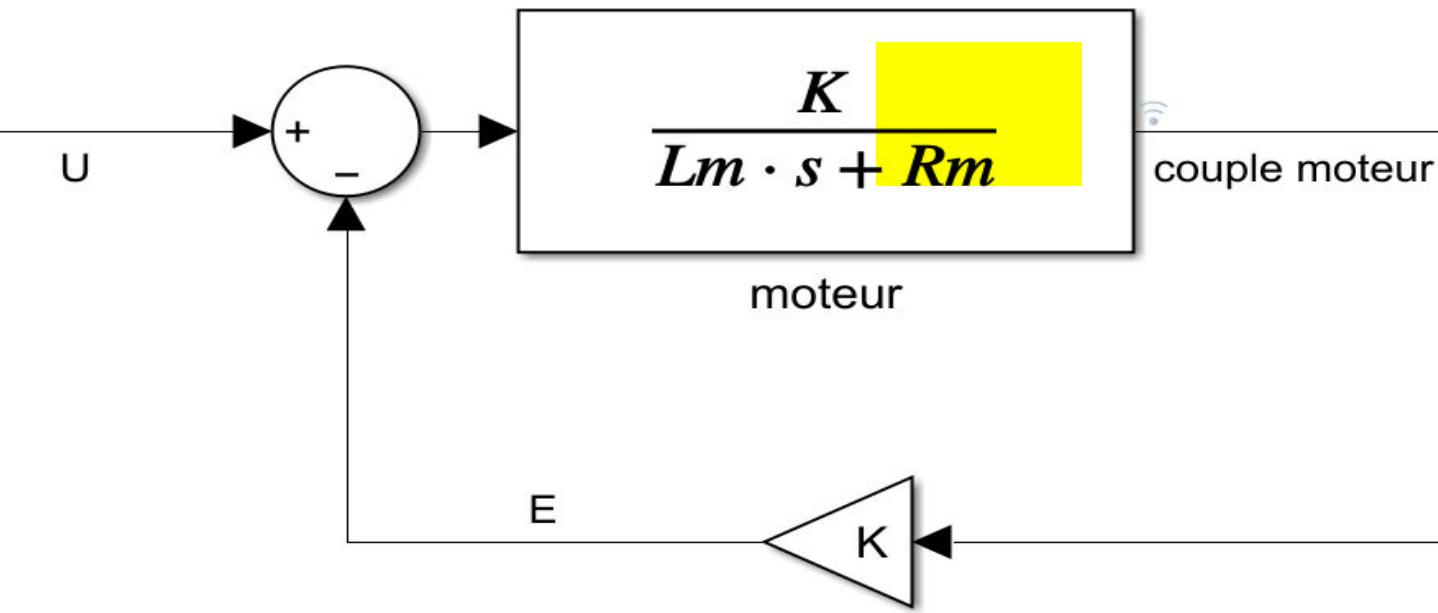
Les conditions de Heaviside sont valides sauf pour  $\alpha$

$$\frac{d^2\alpha}{dt^2} = p^2 \alpha(p) - \alpha v$$

$\alpha v$ : vitesse initiale : perturbation

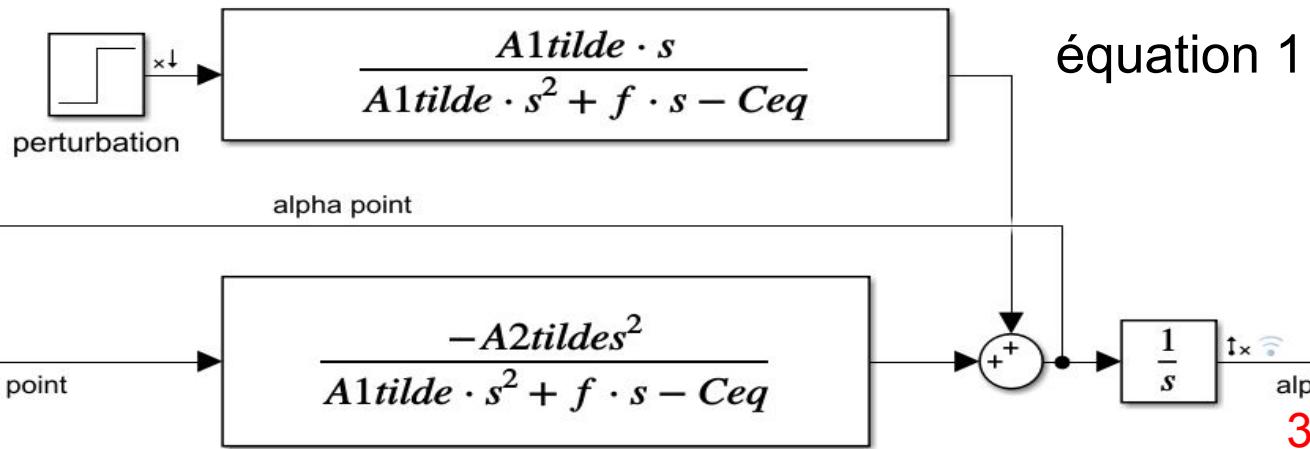
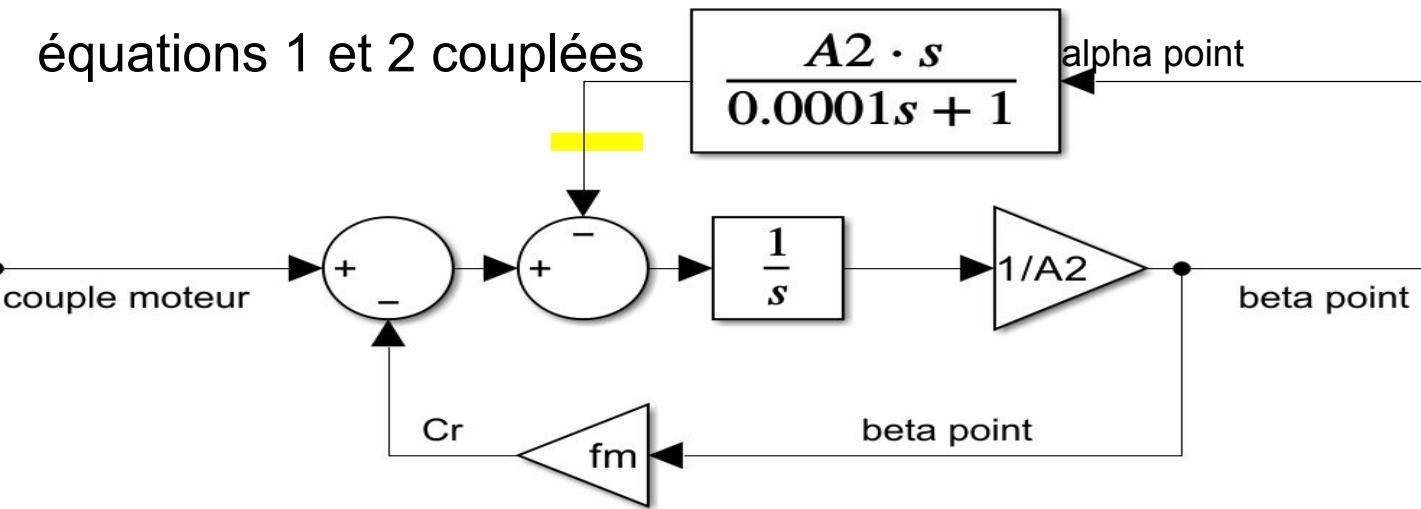
# Boucle ouverte





bloc moteur

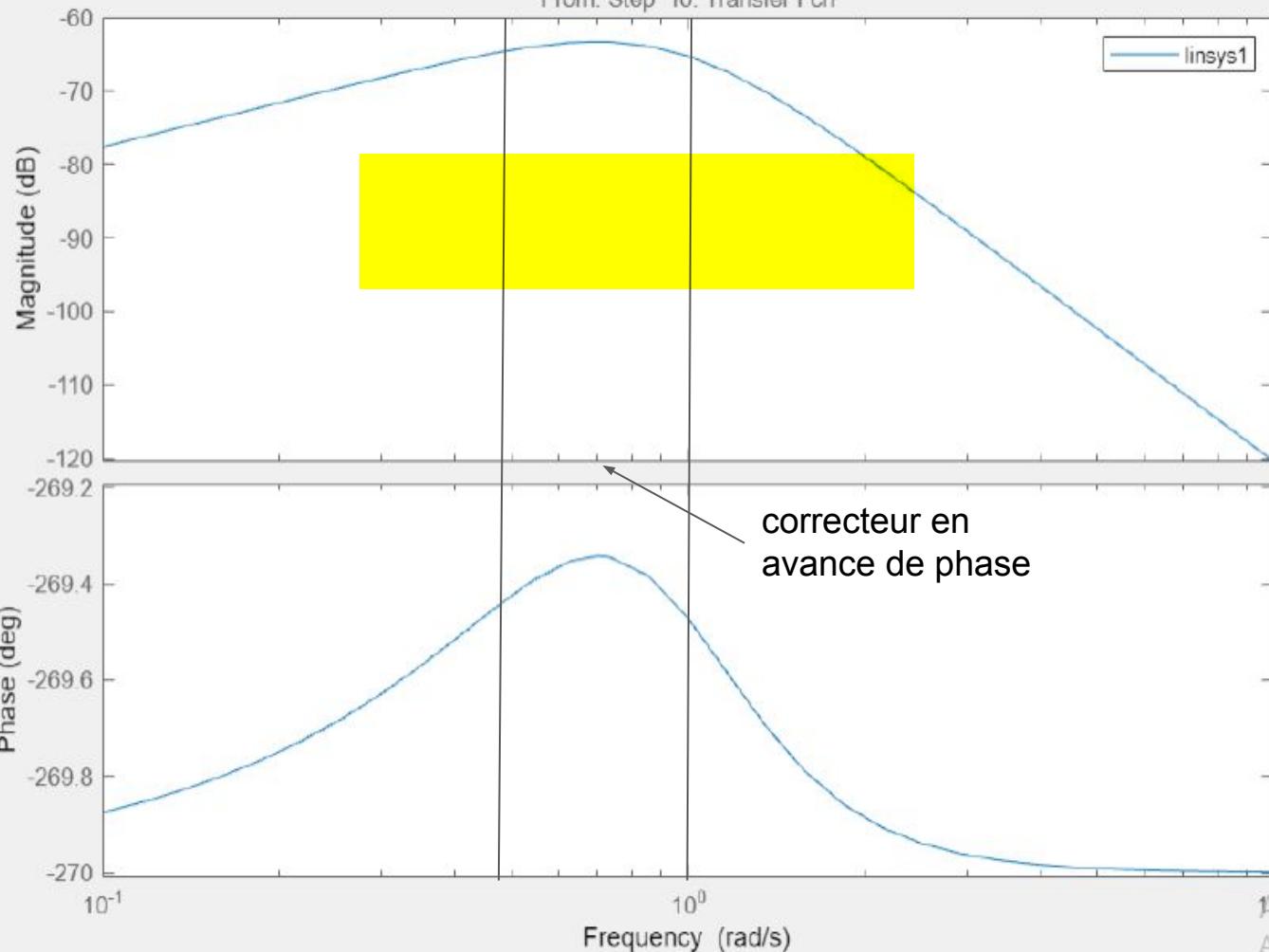
équations 1 et 2 couplées



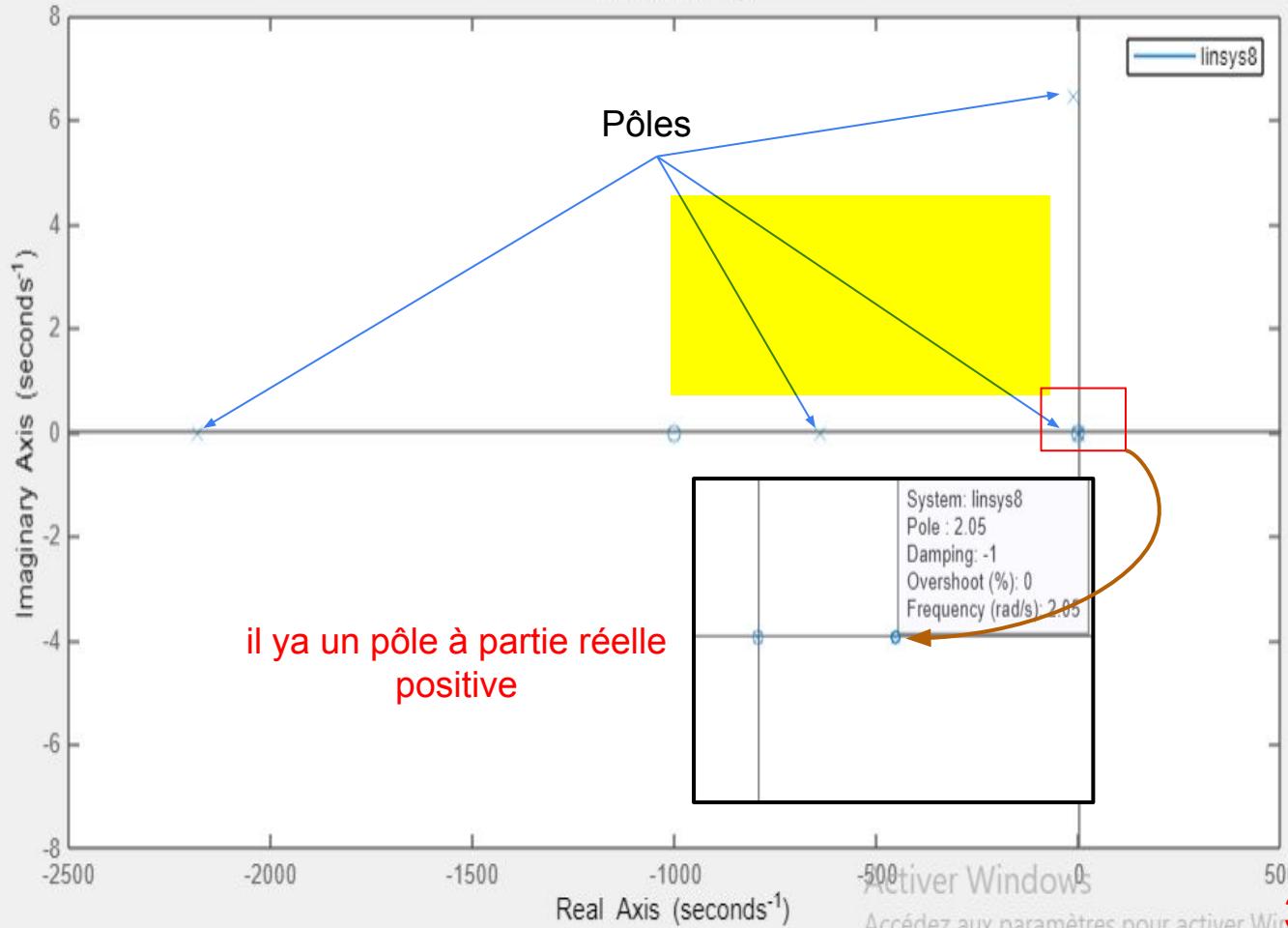
## Diagrammes de Bode de la FTBO

Bode Diagram

From: Step To: Transfer Fcn

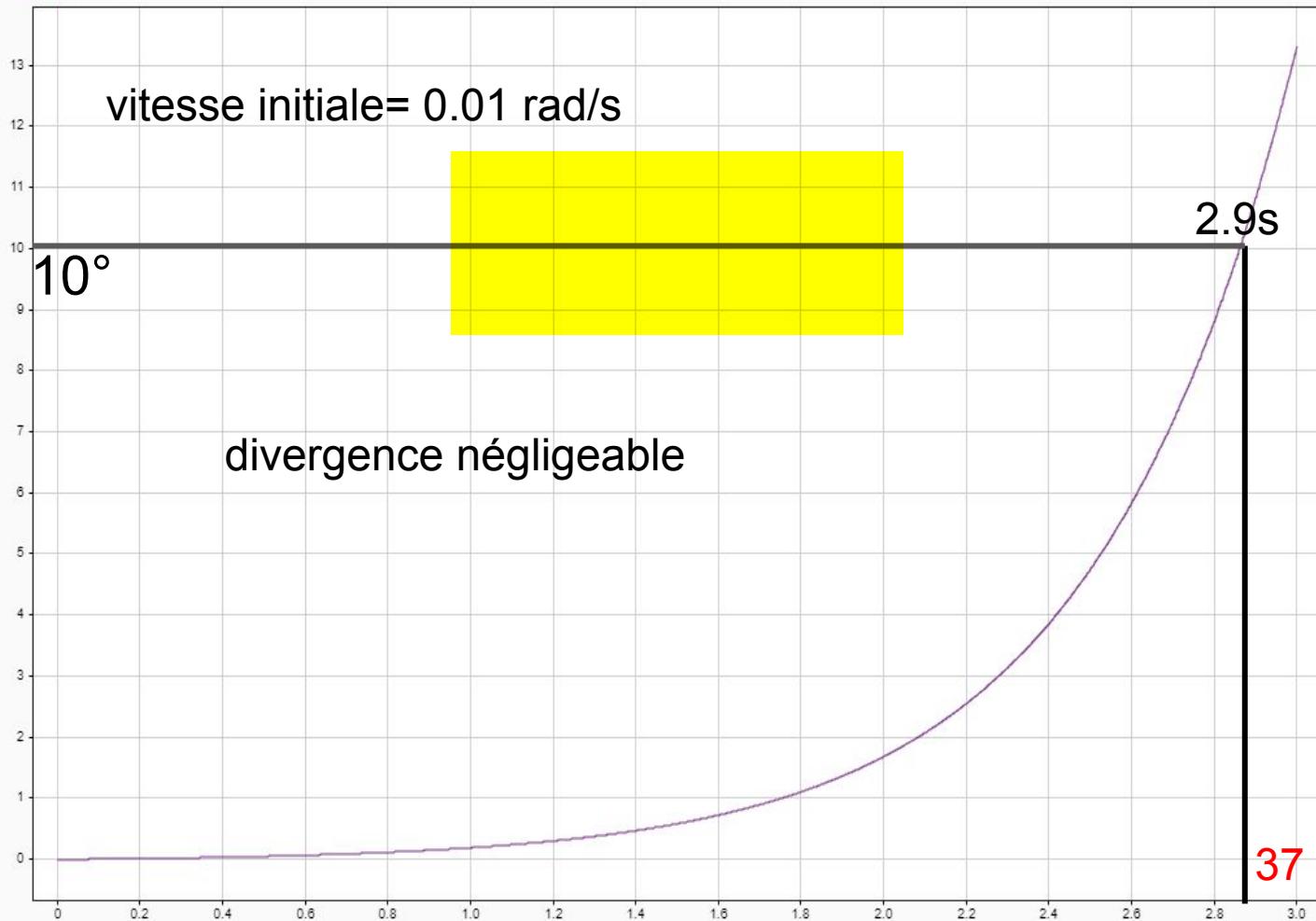


Pole-Zero Map

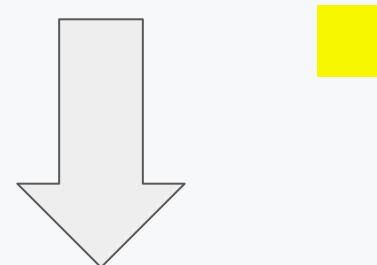


# alpha en deg

■ alpha en deg

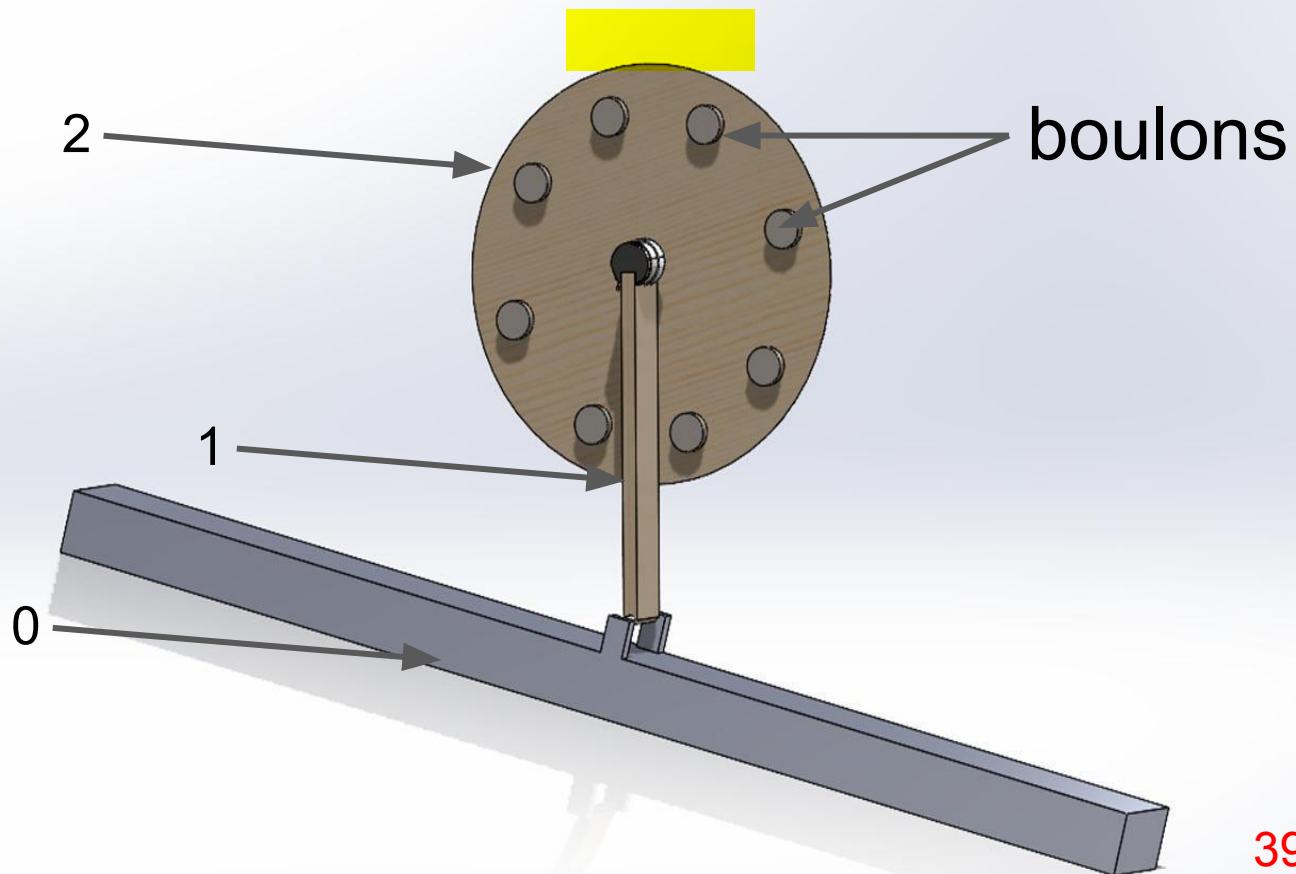


# Le modèle diverge

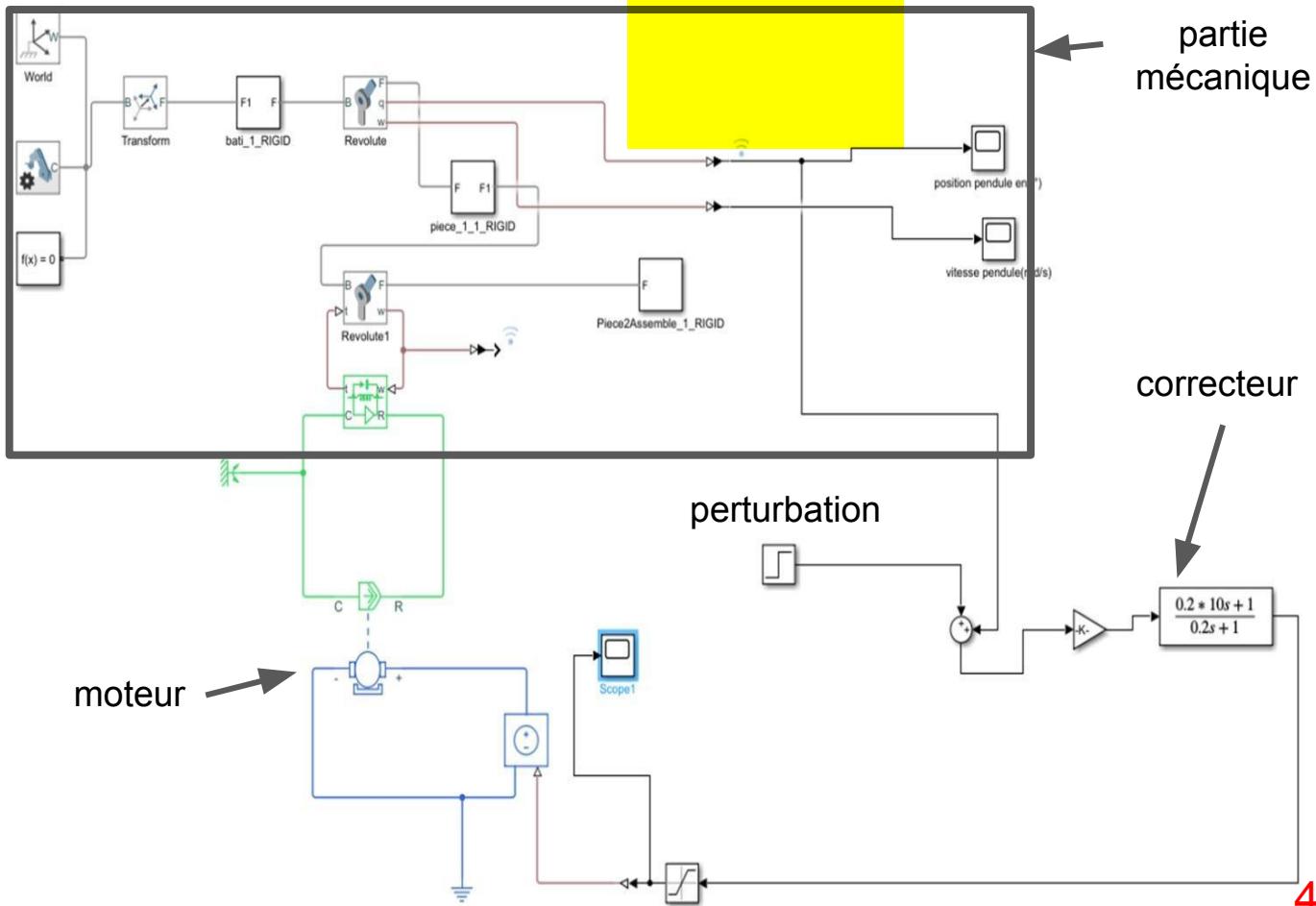


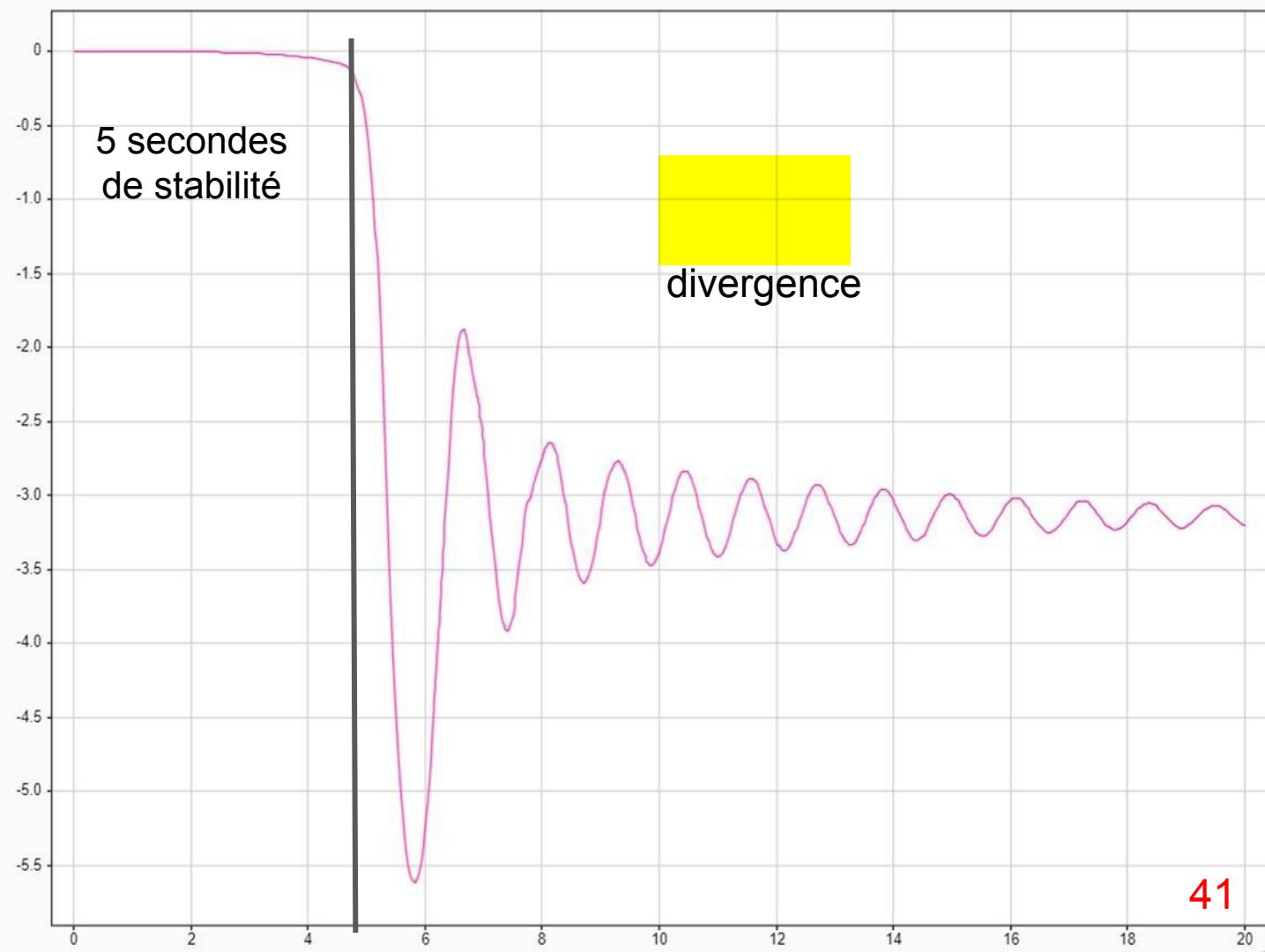
modèle linéaire ?

Mon collègue s'est intéressé à solidworks pour faire une simulation sur matlab  
avantage : dépasser le besoin de la linéarisation et avoir un moteur plus puissant



# Bloc multi-physique





5 secondes de stabilité

La tension augmente en exponentielle

Le moteur est saturée il ne plus augmenter la tension

-24 V



L'implémentation de la simulation n'est  
pas à valider

-

le système tombe directement

# Échec et limitations



centrale inertie

+10 ms

carte arduino

+20 ms

## V. conclusion:

pour un système asservi par une technologie suffisamment rapide

[redacted]  
la stabilisation nécessite une étude plus approfondie

Mais

L'amortissement de la chute est une implémentation envisageable du système



**Fin de la**  
**présentation**

**Candidat**  
**40677**

# **annexes**

# Annexe 1

## Liaison pivot ?

Roulement à billes



Charnière



caractéristiques

Réf 67555166

- il y a un frottement sec important
- pas de jeux
- pas de frottement sec
- un jeux plus important
- masse plus importante

# Annexe 2

On applique le TMD sur (1+2) selon X0 en O:

moment dynamique de 1+2 selon  $\vec{n}_0$  en O:

$$\begin{aligned} \times \vec{\delta}(8,1+2/0) \cdot \vec{n}_0 &= \vec{\delta}(8,1/0) \cdot \vec{n}_0 + \vec{\delta}(8,2/0) \cdot \vec{n}_0 \\ \rightarrow \vec{\sigma}(8,1/0) \cdot \vec{n}_0 &= m_1 (\vec{O} G_1 \wedge \vec{V}(8,1/0)) \cdot \vec{n}_0 + [\vec{\tau}(8/0)]_{B_1} \left( \begin{array}{c} \ddot{\alpha} \\ 0 \\ 0 \end{array} \right) \cdot \vec{n}_0 \\ &= \vec{\alpha} \\ \vec{\delta}(8,1/0) \cdot \vec{n}_0 &= \frac{d \vec{\sigma}(8,1/0)}{dt} \Big|_0 \cdot \vec{n}_0 + m_1 \vec{V}(8,1/0) \wedge \vec{V}(G_1, 1/0) \cdot \vec{n}_0 \\ \Rightarrow \vec{\delta}(8,1/0) \cdot \vec{n}_0 &= A_1 \ddot{\alpha} \\ \vec{\sigma}(A,2/0) \cdot \vec{n}_0 &= m_2 (\vec{OA} \wedge \vec{V}(A,2/0)) \cdot \vec{n}_0 + [\vec{\tau}(A/0)]_{B_2} \left( \begin{array}{c} \ddot{\alpha} \\ \ddot{\beta} \\ 0 \end{array} \right) \cdot \vec{n}_0 \\ &= A_2 (\ddot{\alpha} + \ddot{\beta}) \cdot \vec{n}_0 \\ (\vec{\sigma}(8,2/0)) &= \vec{\sigma}(A,2/0) + m_2 \vec{OA} \wedge \vec{V}(A,2/0) \cdot \vec{n}_0 \end{aligned}$$

$$\begin{aligned} \vec{OA} &= L \vec{y}_1 + v_{OA} \vec{n}_0 \\ \Rightarrow \vec{V}(A,2/0) &= \frac{d \vec{OA}}{dt} \Big|_0 = L \frac{d \vec{y}_1}{dt} \Big|_0 + \vec{\alpha} \\ &= \ddot{\alpha} L \vec{z}_1 \end{aligned}$$

Ainsi  $(\vec{OA} \wedge \vec{V}(A,2/0)) \cdot \vec{n}_0 = (\ddot{\alpha} L^2 \vec{n}_0) \cdot \vec{n}_0 = \ddot{\alpha} L^2$

$$d'où \vec{\sigma}(8,2/0) \cdot \vec{n}_0 = A_2 (\ddot{\alpha} + \ddot{\beta}) + \ddot{\alpha} L^2 m_2 \vec{n}_0$$

$$\vec{n}_0 \cdot \vec{\delta}(8,2/0) = \frac{d \vec{\sigma}(8,2/0)}{dt} \Big|_0 \cdot \vec{n}_0 + m_2 \vec{V}(8,2/0) \wedge \vec{V}(A,2/0) \cdot \vec{n}_0$$

$$\vec{\delta}(8,2/0) \cdot \vec{n}_0 = A_2 (\ddot{\alpha} + \ddot{\beta}) + \ddot{\alpha} L^2 m_2$$

Moment des actions mécaniques

$$*\overrightarrow{\mathcal{M}}_{\overrightarrow{P}_1}(0) = \overrightarrow{\mathcal{M}}_{\overrightarrow{P}_1}(G_1) + \overrightarrow{OG_1} \wedge \overrightarrow{P}_1 \\ = \overrightarrow{O} + \overrightarrow{OG_1} \wedge m_1 g \overrightarrow{y_0}$$

$$\Rightarrow \overrightarrow{\mathcal{M}}_{\overrightarrow{P}_1}(0) \cdot \overrightarrow{\kappa_0} = -(\overrightarrow{OG_1} \wedge m_1 g \overrightarrow{y_0}) \cdot \overrightarrow{\kappa_0} = -m_1 g (\overrightarrow{y_0} \wedge \overrightarrow{\kappa_0}) \cdot \overrightarrow{OG_1} \\ = +m_1 g \overrightarrow{z_0} \cdot \overrightarrow{OG_1} = \underline{m_1 g l_{g_1} \sin(\alpha)}$$

$$*\overrightarrow{\mathcal{M}}_{\overrightarrow{P}_2}(0) = \overrightarrow{\mathcal{M}}_{\overrightarrow{P}_2}(A) + \overrightarrow{OA} \wedge \overrightarrow{P}_2 \cdot \overrightarrow{\kappa_0} = \overrightarrow{OG_1} \cdot \overrightarrow{y_1}$$

$$\overrightarrow{\mathcal{M}}_{\overrightarrow{P}_2}(0) \cdot \overrightarrow{\kappa_0} = -m_2 g (\overrightarrow{y_0} \cdot \overrightarrow{\kappa_0}) \overrightarrow{OA} = +m_2 g \overrightarrow{z_0} \cdot \overrightarrow{OA} = \underline{m_2 g L \sin(\alpha)}$$

$$\overrightarrow{E_6}(0) = -6 \dot{\alpha} \overrightarrow{\kappa_0}$$

d'où l'équation

$$A_1 \frac{d^2 \alpha}{dt^2} + A_2 \left( \frac{d^2 \alpha}{dt^2} + \frac{d^2 \beta}{dt^2} \right) + m_2 L^2 \left( \frac{d^2 \alpha}{dt^2} \right) = g \sin(\alpha) (L m_2 + l_{g_1} m_1) - f \frac{da}{dt}$$

On applique le **TMD** sur **2** selon **X0** en **A**:

$$\vec{\delta}(A, 2/0) \cdot \vec{u}_0 = \frac{d \vec{G}(A, 2/0)}{dt} \Big|_0 \cdot \vec{u}_0 + m_2 \underbrace{(\vec{v}(A, 2/0) - \vec{v}(A, 2/0))}_{= \vec{0}} \cdot \vec{u}_0 \\ = A_2 (\ddot{\alpha} + \dot{\beta})$$

$$\vec{C}_m(A) \cdot \vec{u}_0 = C_m$$

$$\vec{C}_f(A) \cdot \vec{u}_0 = - f_m \dot{\beta} \vec{u}_0$$

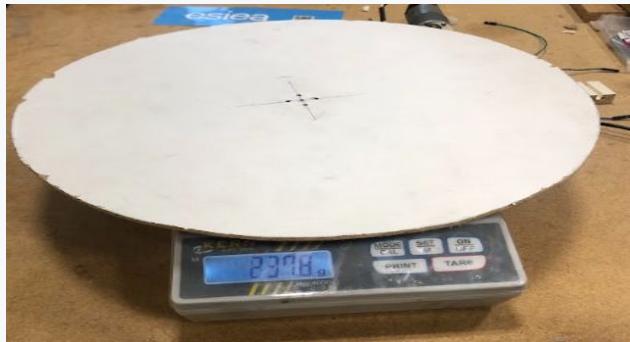
$$A_2 \left( \frac{d^2 \alpha}{dt^2} + \frac{d^2 \beta}{dt^2} \right) = C_m - f_m \frac{d\beta}{dt}$$

# Annexe 3

mesures directes de L m<sub>2</sub> , m<sub>1</sub>

$$m_2 = 0.24355 \text{ Kg}$$

$$L = 26.25 \text{ Cm}$$



Classe 1:

$$\text{Moteur: } m(\text{moteur}) = 0.2632 \text{ Kg}$$

$$\text{Plaque: } m(\text{plaque}) = 0.25243 \text{ Kg}$$

$$\frac{1}{2} \text{ charnière: } m(\frac{1}{2} \text{ charnière}) = 0.15421 \text{ Kg}$$

$$m_1 = m_{\text{moteur}} + m_{\text{plaque}} + m_{\frac{1}{2} \text{ charnière}} \rightarrow m_1 = 0.66984 \text{ Kg}$$





La roue est parfaitement homogène  
donc son inertie par rapport à son axe  
est

$$A_2 = m_2 \times \frac{R^2}{2}$$

Avec  
 $m_2 = 0.24355 \text{ Kg}$   
 $R(\text{roue}) = 16.5\text{Cm}$

Donc  
 $A_2 = 0.0033153 \text{ Kg.m}^2$

# Annexe 4

```
1  lol=0.2625
2  lal=0.1075
3  e0l= 0.0125
4  f1=0.02+0.5*0.0015
5  Lm=0.044
6  lm=0.075
7  D1={"plaque":[[lal*0.5,lol*0.5,0],0.25243],"1/2charnière":[[0,0.0606,-0.5*e0l],0.15421],"moteur":[[0.5*lal-
lm/2,lol+Lm/2,0] , 0.2632]}
8  D2={"roue":[[f1+lal*0.5,lol,0],0.24355]}
9  #les centres de gravité sont exprimés dans la base B1
10 import numpy as np
11 def G(D):
12     m=0
13     x,y,z=0,0,0
14     for L in D.values():
15         m+=L[1]
16         x+=L[0][0]*L[1]
17         y+=L[0][1]*L[1]
18         z+=L[0][2]*L[1]
19     x=x/m
20     y=y/m
21     z=z/m
22     return [[x,y,z],m]
23 #ce systeme permet de calculer le centre de gravite de chaque système, il prend en argument un dictionnaire qui décrit la
classe d'équivalence et donne une liste [[les coordonnées du centre de gravité], masse totale du système] dans la base 1
24
```

# Annexe 5

## Expérience de la pendule inverse

On reprend la première équation où on a isolé 1+2:

$$A_1 \frac{d^2\alpha}{dt^2} + A_2 \left( \frac{d^2\alpha}{dt^2} + \frac{d^2\beta}{dt^2} \right) + m_2 L^2 \left( \frac{d^2\alpha}{dt^2} \right) = g \sin(\alpha) (L m_2 + l_{g1} m_1) - f \frac{d\alpha}{dt}$$

on fait comme si la pièce 2 n'existait pas

C.-à-d.,  $A_2=0$   $m_2=0$

Donc on obtient (avec considération de Cr0)

$\alpha$  oscille autour de  $\pi$

On fait un changement de variable  $\tilde{\alpha}=\pi-\alpha \Leftrightarrow \alpha = \pi - \tilde{\alpha}$

$\tilde{\alpha}$  oscille autour de 0

On fait l'approximation  $\sin(\tilde{\alpha}) = \tilde{\alpha}$

Donc l'équation devient

$$\frac{d^2\tilde{\alpha}}{dt^2} + \frac{f}{A_1} \frac{d\tilde{\alpha}}{dt} + \tilde{\alpha} \frac{g l_{g1} m_1}{A_1} = C_{r0}$$

## Annexe 6



Centrale inertielle:  
mpu6050



Carte arduino Méga

$$w_r = w_0 \sqrt{1 - \xi^2}$$

la solution est de la forme :

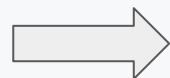
$$\tilde{\alpha}(t) = B e^{-\xi w_0 t} \cos(w_r t + \varphi)$$



chercher:

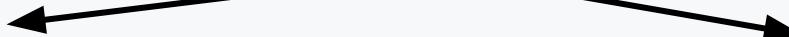
- la période: T
- l'enveloppe des oscillations:  
 $y=B \cdot \exp(-\xi \cdot W_0 \cdot t)$

filtre



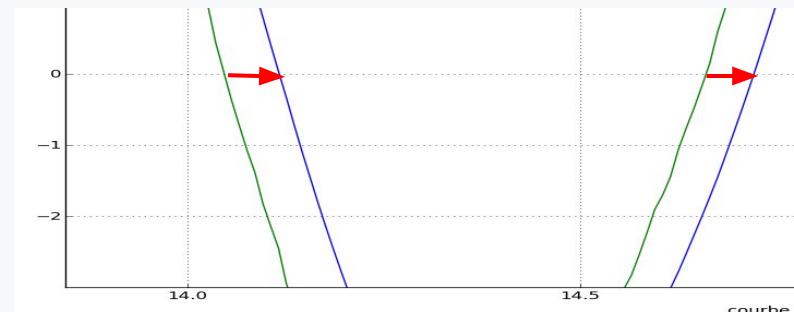
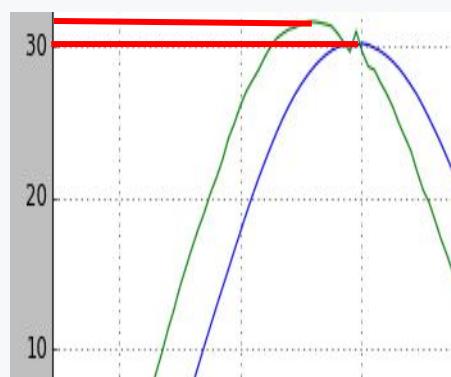
corriger le filtre

## Annexe 7

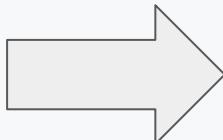


régler l'amplitude

$$K_{cor} = 1/H(W_r) = \sqrt{1+1/9}$$



Le programme “depht” calcule le déphasage entre les deux courbes



correction de la courbe filtrée :

le programme “correction\_courbe\_filtree”

programme correction\_courbe\_filtre dans l'annexe 7

programme depht dans l'annexe 7

```
15 import numpy as np
16 import matplotlib.pyplot as plt
17
18 lgl=0.16655703376925832
19 dt=0.01 #durée d'aquisition
20 g=9.8
21 ml= 0.66984
22 sin=[np.sin(i*dt*2)+np.cos(i*dt*4)*0.1 for i in range(len(L2))]
23 def traitecourbe(H):
24     Ll=[]
25     for i in H:
26         if i<-90:
27             Ll.append(i+180)
28         elif i>90:
29             Ll.append (i-180)
30         else:
31             Ll.append(i)
32     return Ll
33 #ce système permet de traiter les listes , en cetrant les valeurs sur 0
```

```
34 def courbes(L,A=None,B=None,C=None,D=None):
35     for H in (L,A,B,C,D):
36         if H is not None:
37             L1=traitercourbe(H)
38             X=np.linspace(0,len(L1)*dt, len(L1))
39             Y=np.array(L1)
40             plt.plot(X,Y)
41             plt.grid()
42             plt.xlabel('Axe des temps en [s]')
43             plt.ylabel('acquisition de l''angle alphatilde ')
44     plt.show()
45 #ce système permet d'afficher les courbes traitées
```

```
47 def periode(liste): #done et marche bien
48     L=traitercourbe(liste) #traiter la liste
49     indzeros=[] #les indices ou la courbe s'annule
50     for i in range(len(L)-1): #reperer les indices d'annulation
51         if L[i]*L[i+1]<=0:
52             indzeros.append(i)
53     periode=[] #liste des periodes
54     for i in range(len(indzeros)-2): #calculer la periode
55         periode.append(dt*(indzeros[i+2]-indzeros[i]))
56     MoyT=np.mean(periode) #la valeur moyenne de la periode
57     periodef=[] # filtrer les valeurs des periodes non raisonnables
58     for T in periode:
59         if T>=MoyT*0.75 and T<= MoyT*1.25:
60             periodef.append(T)
61     return np.mean(periodef) #donner la valeur moyenne raisonnable
62
63 #on applique un filtre passe bas pour lisser la courbe et la rendre plus exploitable, les bruits d'aquisition
   seront atténués
```

```
65 def filtrercourbes(L):#done et marche bien
66     wc=3*(2*np.pi/periode(L)) #fréquence de coupure/de filtre , elle est égale à 3 fois la fréquence
d'oscillation
67     LN=traitercourbe(L) #courbe traitée ou ramenée à 0
68     s=[LN[0]] # boucle qui donne s: les valeurs filtrées
69     for i in range(0,len(LN)-1):
70         s.append(LN[i]*wc*dt+s[i]*(1-wc*dt))
71     X=np.linspace(0,len(LN)*dt, len(LN))
72     #plt.plot(X,s)
73     #plt.xlabel('courbe filtrée')
74     #plt.plot(X,LN)
75     #plt.grid()
76     #plt.show()
77     return s      # courbe filtrée et bruit éliminé
```

```

81 def depht(L): #done et marche bien // cette fonction permet de calculer le déphasage en temps pour la
course filtrer
82     CF=filtrercourbes(L)#courbe filtrée
83     C=traitercourbe(L) #courbe normale
84     # processus de reperage les indices d'annulation de la courbe originale et filtrée
85     indzerosF=[] #les indices pour lesquels la courbe filtrée s'annule
86     indzerosC=[] #les indices pour lesquels la courbe originale s'annule
87     for i in range(len(CF)-1):
88         if CF[i]*CF[i+1]<0:
89             indzerosF.append(i)
90     for i in range(len(C)-1):
91         if C[i]*C[i+1]<0:
92             indzerosC.append(i)
93
94     depht=[] #cette liste contient chaque déphasage existant
95     for i in range(min(len(indzerosC),len(indzerosF))):
96         depht.append(dt*abs(indzerosF[i]-indzerosC[i]))
97
98     #pour prendre que les valeurs significatives on prend que les valeurs autour de la valeur moyenne, les
valeurs loing de cette valeur seront éliminé
99     moydepht=np.mean(depht)#valeur moyenne avant brut
100    dephtF=[]
101    moydepht=np.mean(depht)
102    for depht in depht:
103        if depht>=moydepht*0.5 and depht<=moydepht*1.5:
104            dephtF.append(depht)
105            # print(str(depht))
106    return int((0+np.mean(dephtF))/dt)*dt #valeur moyenne nette du déphasage approximée aux chiffres
significatifs de dt
107
108 #la fonction correction_courbe_filtree permet de régler ce déphasage et utilise un correcteur (gain pour
corriger l'amplitude , à la fin de cette fonction la courbe sera parfaitement exploitable

```

```
110 def correction_courbe_filtree (L): #done et marche bien
111     LF=filtrercourbes(L) #courbe après filtrage
112     #Kcor=max(traitercourbe(L))/max(LF)
113     Kcor=(np.sqrt(1+1/9)) #correcteur de l'amplitude en gain
114     #print(Kcor)
115     LF=[i*Kcor for i in LF] #courbe filtré corrigée en amplitude
116     deltat=dephet(L) #déphasage en temps
117     Xl=np.linspace(0,len(L)*dt,len(L)) # courbe du temps
118     X=[i-deltat for i in Xl] # régler le déphasage en temps
119     #ppp=X[0]
120     #X=[i-ppp for i in X]
121     #print (X[0])
122     #plt.plot(Xl,traitercourbe(L)) # afficher courbe originale
123     #plt.plot(X,LF) # afficher courbe filtrée corrigée
124     return LF #courbe filtrée corrigée
125
126 def estsommet(s,i):#done et marche bien
127     if s[i]>s[i+1] and s[i]>s[i-1] and s[i]>s[i+2] and s[i]>s[i-2] and s[i]>s[i+3] and s[i]>s[i-3] and
128         s[i]>s[i+4] and s[i]>s[i-4]:
129         return True
130     else:
131         return False
132 #permet de vérifier si le point d'indexe i est un sommet de s , s doit être la liste filtrée
```

```

133 def courbe_et_enveloppe(L): #done et marche bien
134     s=correction_courbe_filtree(L)
135     #processus de determination des indexes de sommets
136     indsommets=[]
137     for i in range(4,len(s)-4):
138         if estsommets(s,i):
139             indsommets.append(i)
140     #processus d'affichage des sommets
141     Yr=[]#les valeurs des sommets
142     Xr=[]#les instants correspondants aux sommets
143     for i in range(len(indsommets)):# remplissage de Yr et Xr
144         Yr.append(s[indsommets[i]])
145         Xr.append(dt*indsommets[i])
146     lnYr=[np.log(i) for i in Yr]
147     # coefficient de regression # l'enveloppe est sous la forme ( exp((-a*t)+b))
148     a,b=np.polyfit(Xr,lnYr,1)
149     a=-a # coefficient de regression # l'enveloppe est sous la forme ( exp((-a*t)+b))
150     X=np.linspace(0,dt*len(s),len(s))
151     enveloppe=np.exp(-a*i*dt)*np.exp(b+a*depth(s)) for i in range(len(s)) #les valeurs de l'enveloppe d'après la regression
152     plt.figure(' la courbe et son enveloppe associé')
153     plt.plot(X,s)
154     plt.plot(Xr,Yr)#afficher le parcours des sommets détectées
155     plt.plot(X,enveloppe)#afficher l'enveloppe
156     plt.ylabel('l''angle alphanotilde')
157     plt.xlabel('le temps en [s]')
158     plt.grid()
159     plt.figure('la regression')
160     plt.ylabel('Ln(y)')
161     plt.xlabel('le temps en [s]')
162     plt.plot(Xr,lnYr,'*',X,[-a*i*dt+b for i in range(len(X))],'-r')
163     plt.grid()
164     plt.show()
165     return a #a='Xi*W0
166 #ce système permet de déduire xi selon la formule de dépassement ce système donne des valeurs épartillées et ne donne pas une conclusion pertinante

```

```

168 def regression(L): #done et marche bien
169     s=correction_courbe_filtree(L)
170     #processus de determination des indexes de sommets
171     indsommets=[]
172     for i in range(4,len(s)-4):
173         if estsommets(s,i):
174             indsommets.append(i)
175     #processus d'affichage des sommets
176     Yr=[]#les valeurs des sommets
177     Xr=[]#les instants correspondants aux sommets
178     for i in range(len(indsommets)):# remplissage de Yr et Xr
179         Yr.append(s[indsommets[i]])
180         Xr.append(dt*indsommets[i])
181     lnYr=[np.log(i) for i in Yr]
182     # coefficient de regression # l'enveloppe est sous la forme ( exp((-a*t)+b))
183     a,b=np.polyfit(Xr,lnYr,1)
184     a=-a # coefficient de regression # l'enveloppe est sous la forme ( exp((-a*t)+b))
185     return a #a=Xi*W0
186 def Xi_w0(L): #done et marche bien
187     XiW0,Wr=regression(L),2*np.pi/periode(L) # extraire Xi*W0 et la pulsation
188     W0=np.sqrt(Wr**2+XiW0**2) # la pulsation propre
189     Xi=XiW0/W0 # coefficient d'amortissement
190     return Xi,W0 # ce program permet de determiner le coefficient d'amortissement à partir d'une acquisition de valeur
191
192 def f_A1(L):#done et marche bien
193     xi,W0=Xi_w0(L)
194     A1=m1*g*Tg1/(W0**2)
195     f=xi**2*W0*A1
196     return f,A1 #ce programme permet de determiner les coefficinets f et At
197

```

On résout le système:  
qu'on implémente sur  
python:

$$w_r = \frac{2\pi}{T}$$

$$w_r = w_0 \sqrt{1 - \xi^2}$$

$$\xi w_0 = a$$

$$w_0 = \sqrt{\frac{m_1 g l_{g1}}{A_1}}$$

$$\xi = \frac{f}{2w_0 A_1}$$

$$T=1.3 \text{ s}$$

$$a= 0.1146 \text{ s}^{-1}$$

$$m_1= 0.66984 \text{ Kg}$$

$$g=9.8 \text{ m/s}^2$$

$$l_{g1}=17.52 \text{ Cm}$$



$$A_1=0.04617 \pm 0.004 \text{ Kg.m}^2$$

$$f=0.011 \pm 0.002 \text{ Kg.m}^2.\text{s}^{-1}$$

In [52]: f\_A1(Lle)

Out[52]: (0.011137976850621885, 0.046168756167250649)

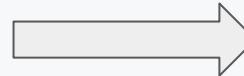
## Annexe 8

Déterminer : K , Rm, Lm, fm,Cr0 : manipuler le moteur

$$U = E + R_m i + L_m \frac{di}{dt}$$

$$E = K \frac{d\beta}{dt}$$

protocole pour déterminer Rm :

- on bloque le moteur:  $E = 0$
- on applique une tension U au borne du générateur bien déterminée
- on attend le régime permanent  $di/dt=0$
- on mesure l'intensité du moteur
- on fait une régression linéaire  on trouve Rm

$$U = R_m i$$

# Outil de mesure



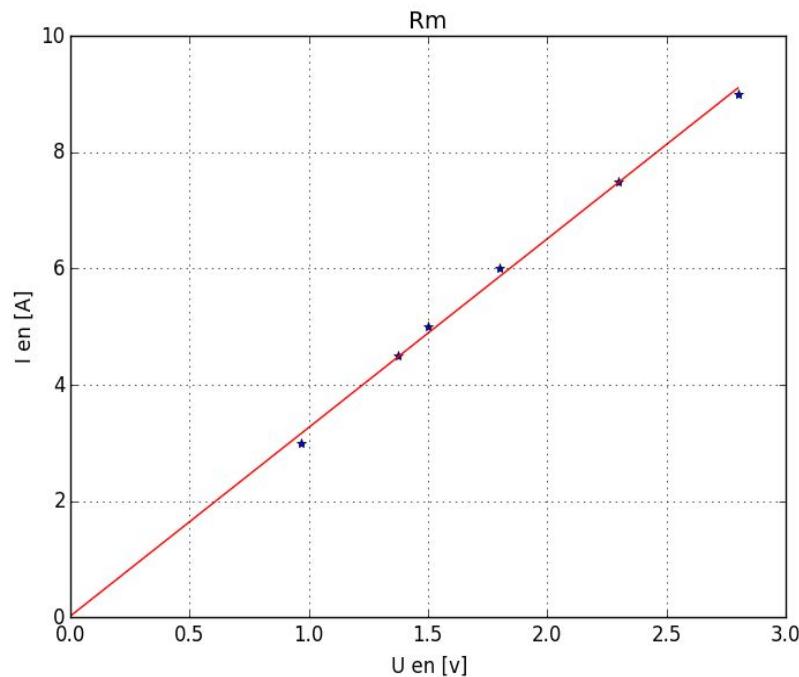
pince ampèremétrique  
Référence:  
fluke i30s manual

rapport de gain = 100 mv/A

La tension est assurée à travers une alimentation stabilisée

$$R_m = 3.2785 \text{ Ohm}$$

U_m en [v]	I_m en [A]
3	1
4,5	1,375
5	1,5
6	1,8
7,5	2,3
9	2,8



```
In [82]: Rmp(I2,U)
Out[82]: 3.2785304469408416
```

```
5
6 I2=[1,1.375,1.5,1.8,2.3,2.8] #en A
7 U=[3,4.5,5,6,7.5,9] #en V
8 def Rm(Xd,Yd):
9     X=np.array(Xd)#Tableaux des mesures
10    Y=np.array(Yd)
11    a,b=np.polyfit(X,Y,1)#Résultats de la régression associée à ce tirage
12    N=1000
13    Xl=np.linspace(0,X[-1],N)
14    Yl=np.array([a*Xl[i]+b for i in range(len(Xl))])
15    #plt.close()#Affichage de la droite de régression et des points
16    #plt.plot(X,Y,'*',Xl,Yl,'-r')
17    #plt.grid()
18    #plt.show()
19    return a, b
20 Rm=Rm(I2,U)[0]
```

$$U = E + R_m i + L_m \frac{di}{dt}$$

$$E = K \frac{d\beta}{dt}$$

Protocole pour déterminer K:

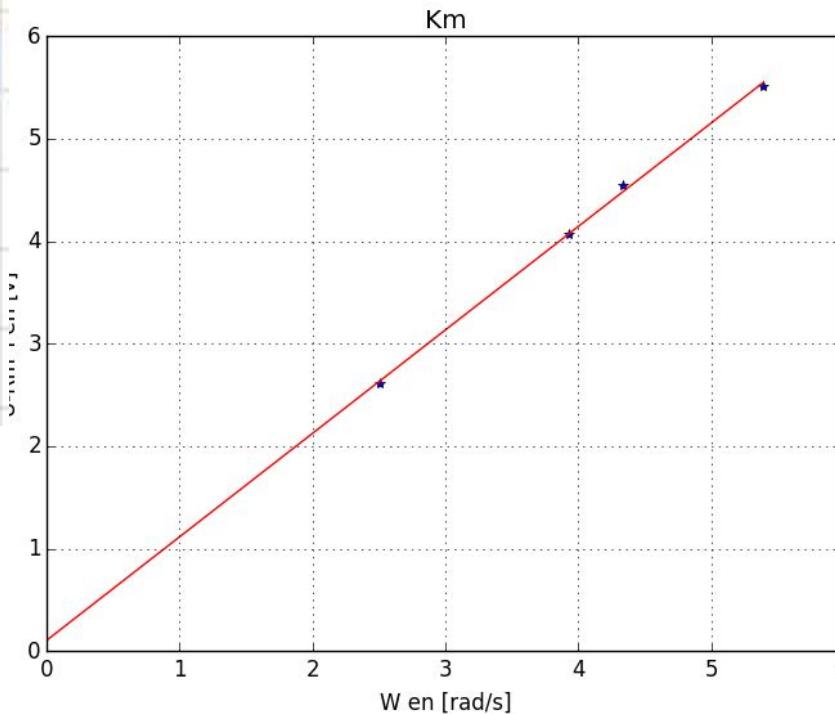
On reprend la même expérience mais en laissant le moteur tourner.

En régime permanent l'équation finale devient

$$U - R_m i = K \frac{d\beta}{dt}$$

- la tension et l'intensité sont acquises avec les même outils
- $d\beta/dt$  est calculée manuellement avec une vidéo prise par téléphone

$U_{en}$ [V]	$I_{en}$ [A]	$W_{en}$ [rad/s]
3	0,117	2,65
4,5	0,133	3,93
5	0,139	4,34
6	0,147	5,29



In [89]:  $Km(Wk, Uk)$

Out[89]: 1.0424443891719259

$$K = 1.04 \text{ v.s/rad}$$

```
23 Uk=[3,4.5,5,6]
24 Ik=[0.117,0.133,0.139,0.147]
25 Uk=[Uk[i]-Rm*Ik[i] for i in range(len(Uk))]
26 Wk=[2.554747547,3.92925409,4.337724873,5.29482169]
27 def Km(Xd,Yd):
28     X=np.array(Xd)#Tableaux des mesures
29     Y=np.array(Yd)
30     a,b=np.polyfit(X,Y,1)#Résultats de la régression associée à ce tirage
31     N=1000
32     X1=np.linspace(0,X[-1],N)
33     Y1=np.array([a*X1[i]+b for i in range(len(X1))])
34     #plt.close()#Affichage de la droite de régression et des points
35     #plt.plot(X,Y,'*',X1,Y1,'-r')
36     #plt.grid()
37     #plt.show()
38     return a, b
39 K=Km(Wk,Uk)[0] # constante du moteur
```

# Protocole pour déterminer $f_m$ , Cr0m:

$$A_2 \frac{d^2\beta}{dt^2} = C_m - C_r$$

$$C_m = K i$$

$$C_r = C_{r0m} + f_m \frac{d\beta}{dt}$$

- On applique une tension au moteur
- On attend le régime permanent  $d^2\beta/dt^2=0$
- On calcule l'intensité avec la pince ampèremétrique
- On mesure la vitesse de rotation du moteur avec un tachymètre

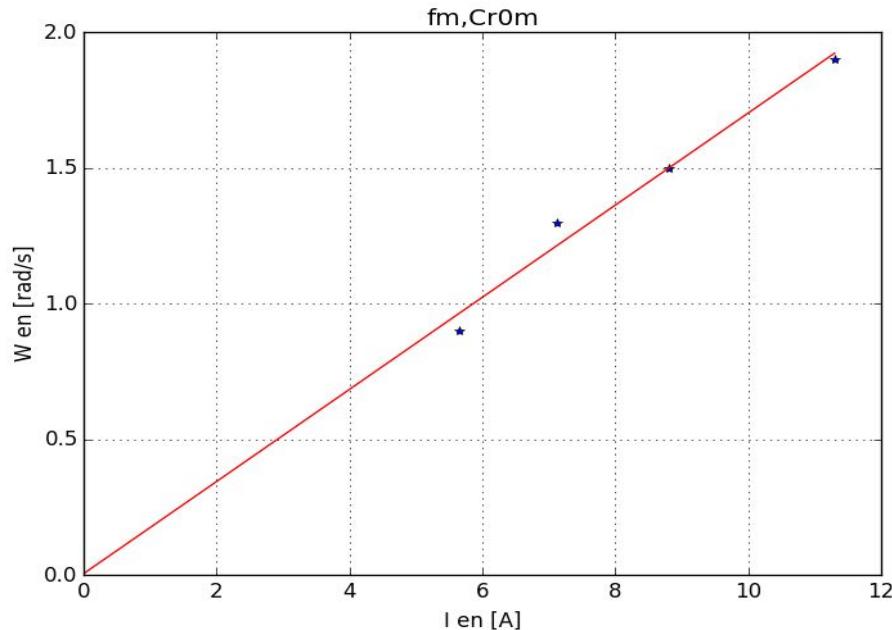
tachymètre

toute les relations confondues:

$$i = \frac{C_{r0m}}{K} + \frac{f_m}{K} \frac{d\beta}{dt}$$



I en [A]	W en [rad/s]
2	11,3
1,5	8,8
1,3	7,12
0,7	5,65



In [93]: fm\_Crm(W,I1)

Out[93]: (0.17689610772347111, 0.0057783796230724254)

relation quasi-linéaire:

donc  $Cr0m=0$  ,  $fm=0.177 \text{ Kg.m}^2.\text{s}^{-1}$

```
41 W=[5.65,7.12,8.8,11.3] # W du moteur rad.s-1 FSK
42 I1=[0.9,1.3,1.5,1.9] # I du moteur en A FSK
43
44 def fm_Crm(Xd,Yd):
45     X=np.array(Xd)#Tableaux des mesures
46     Y=np.array(Yd)
47     a,b=np.polyfit(X,Y,1)#Résultats de la régression associée à ce tirage
48     N=1000
49     X1=np.linspace(0,X[-1],N)
50     Y1=np.array([a*X1[i]+b for i in range(len(X1))])
51     #plt.close()#Affichage de la droite de régression et des points
52     #plt.plot(X,Y,'*',X1,Y1,'-r')
53     #plt.grid()
54     #plt.show()
55     return a*K, b*K
```

## Annexe 9

En effet

$$\tilde{A}1 = A1 + A2 + m2 * L^2$$

$$\tilde{A}2 = A2$$

$$C_{eq} = g * (L * m2 + m1 * lg1)$$

## Annexe 10

### IV. Simulation du modèle:

Les conditions de Heaviside sont valides sauf pour  $\alpha$

La transformée de Laplace d'une dérivée seconde est donnée par :

$$\mathcal{L} \left[ \frac{d^2 f(t)}{dt^2} \right] = p^2 F(p) - p f(0^+) - \frac{df(0^+)}{dt}$$

$$\frac{d^2 \alpha}{dt^2} = p^2 \alpha - \frac{d\alpha}{dt}|_{(0)} \quad (0) = p^2 \alpha(p) - \alpha v \\ \text{en effet: } \frac{d\alpha}{dt}|_{(0)} = 0$$

# alpha en deg

■ alpha en deg

vitesse initiale= 0.001 rad/s

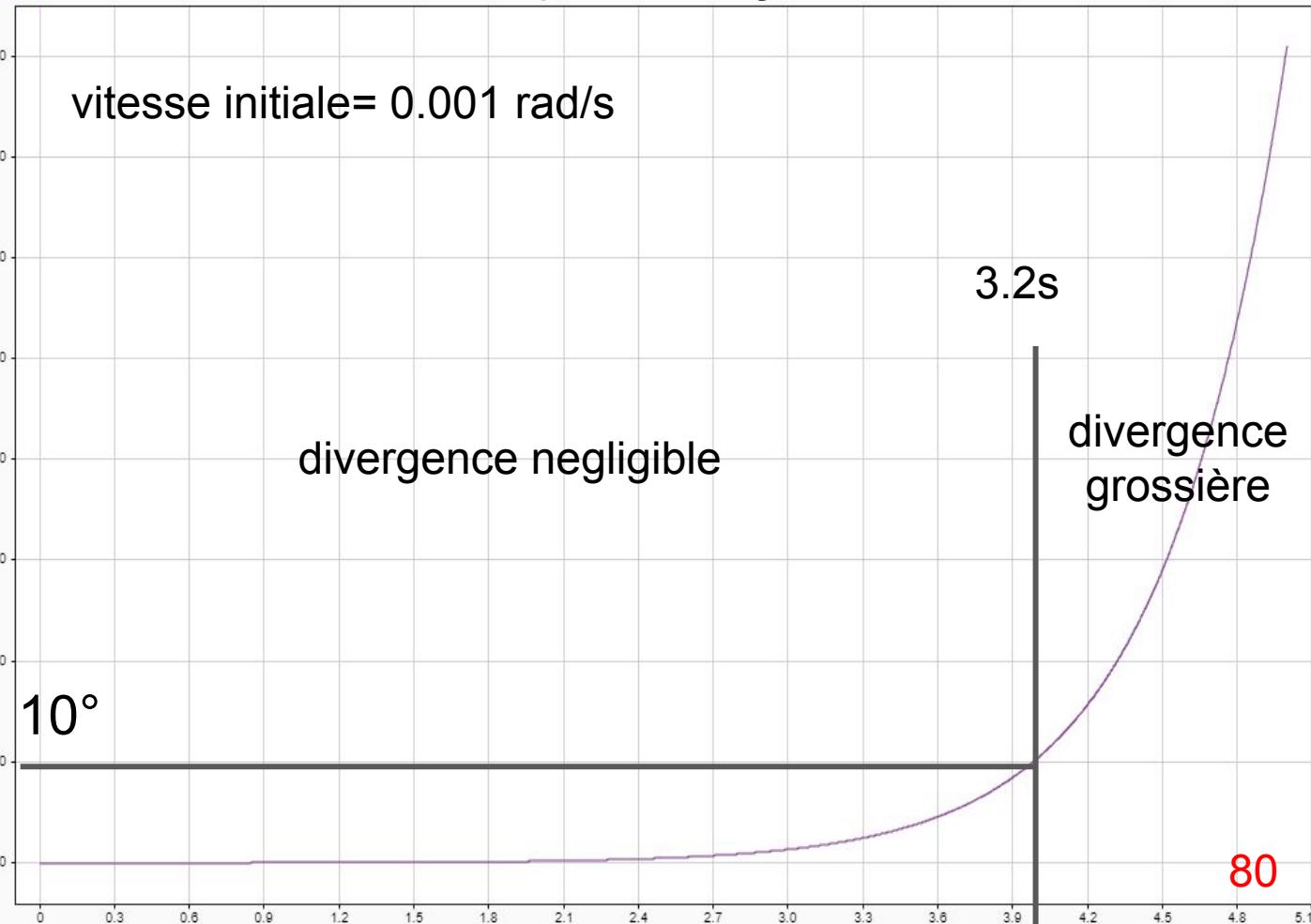
divergence negligible

10°

3.2s

divergence  
grossière

80



■ alpha en deg

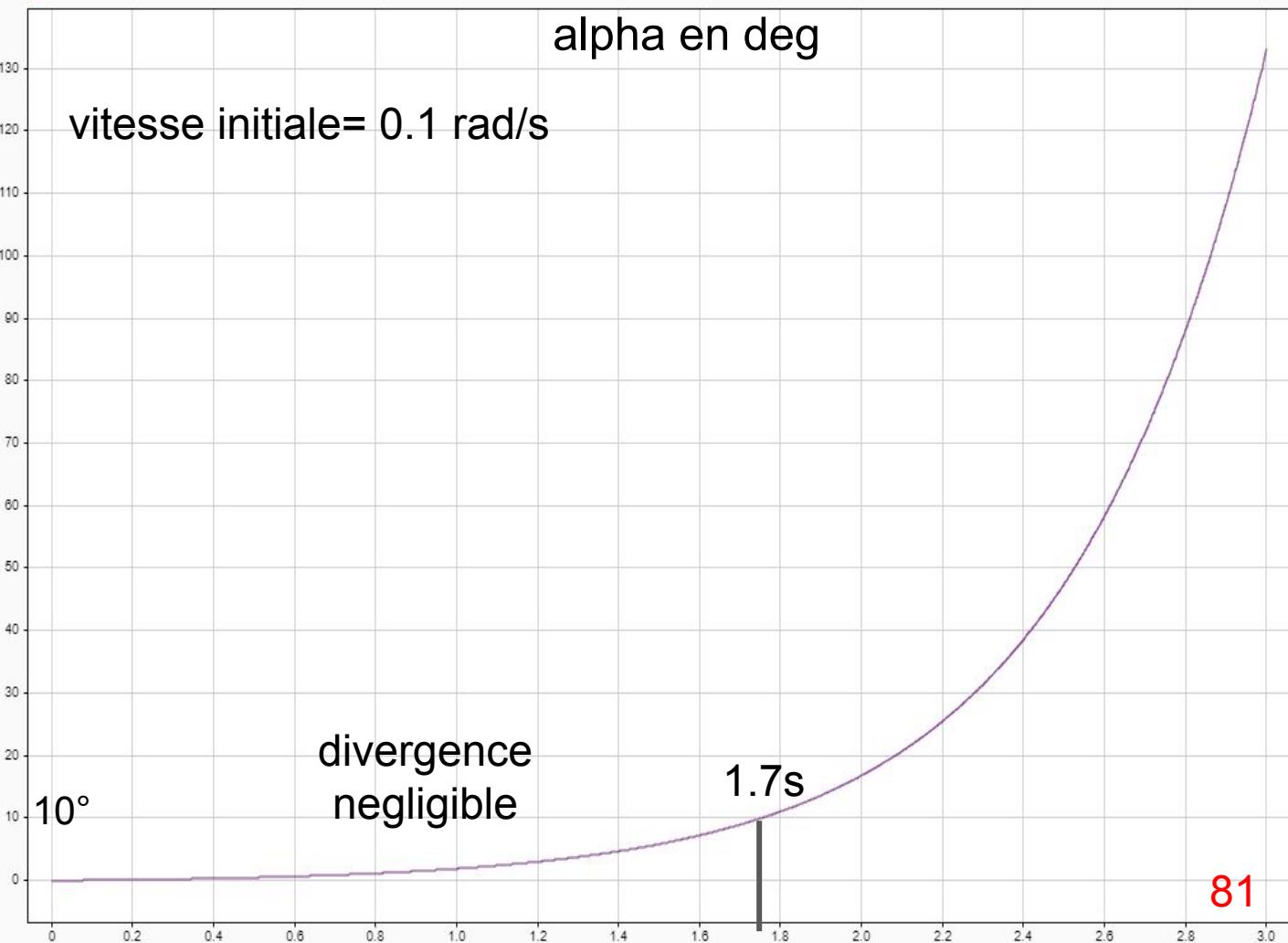
alpha en deg

vitesse initiale= 0.1 rad/s

$10^\circ$   
divergence  
negligible

1.7s

81



Fin de  
l'annexe

Fin  
du projet

Candidat  
40677