Localisation et suivi de la source d'un signal vibratoire

Sommaire

Localiser la source d'un signal avec une précision optimale.

Piloter un dispositif de pointage asservi.

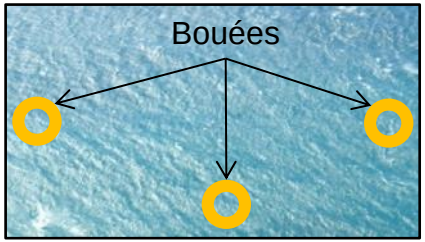
Localisation

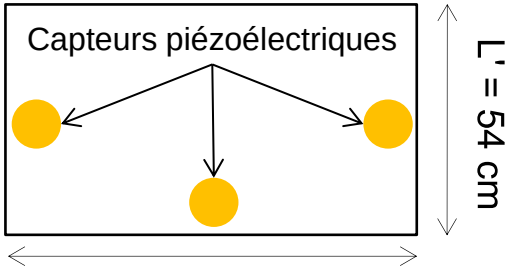
- Principe général
- Etude 1D
- Etude 2D

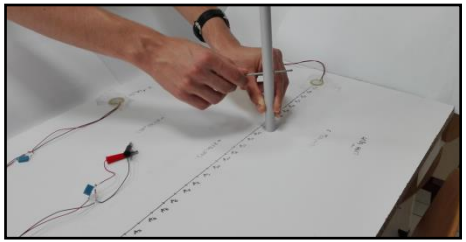
Asservissement

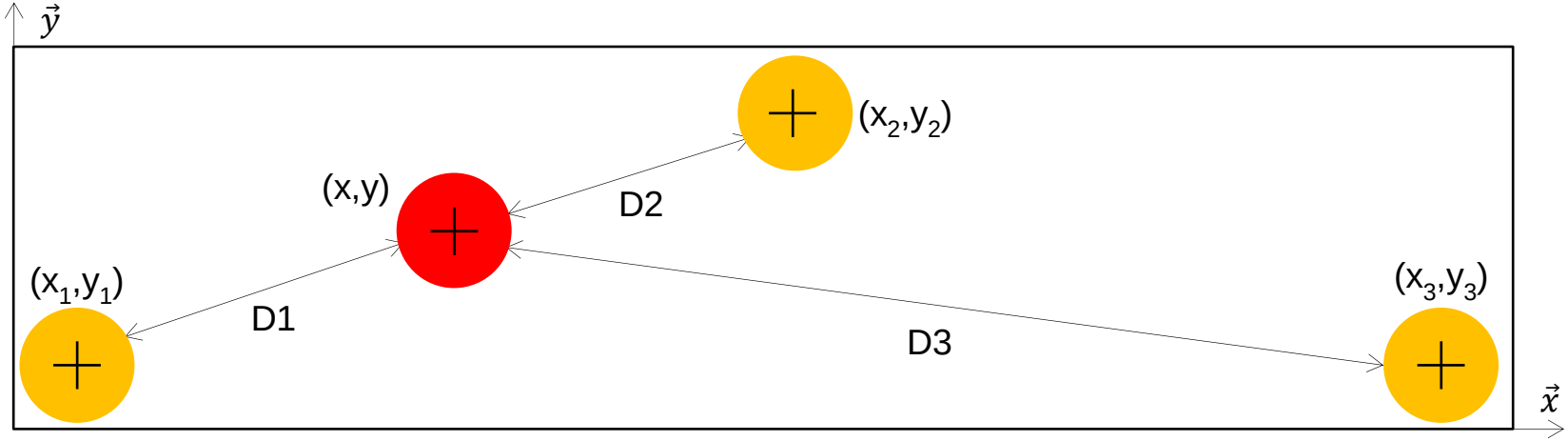
- Détermination des caractéristiques des moteurs utilisés
- Pointage 1D
- Pointage 2D

Localisation : principe général

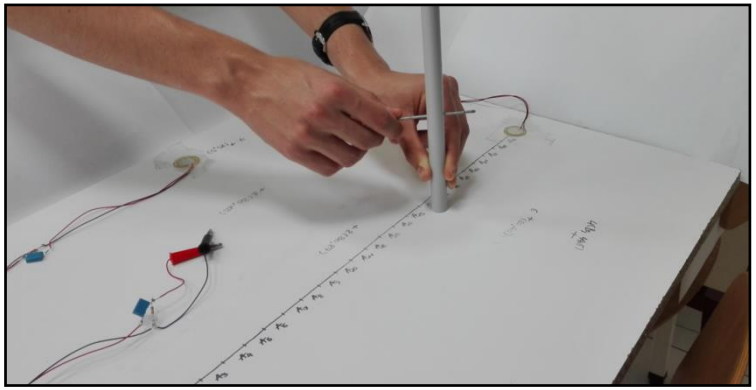


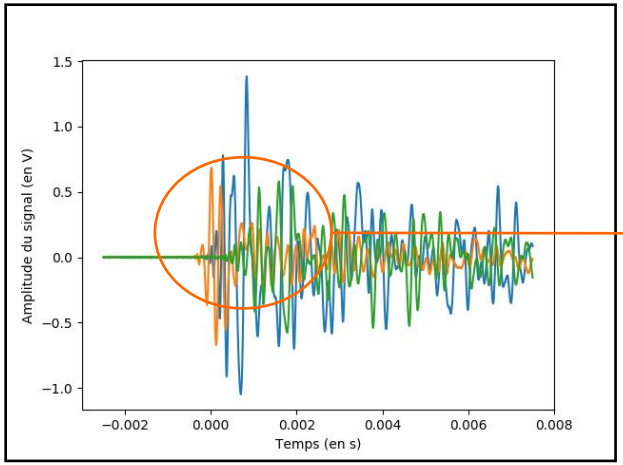


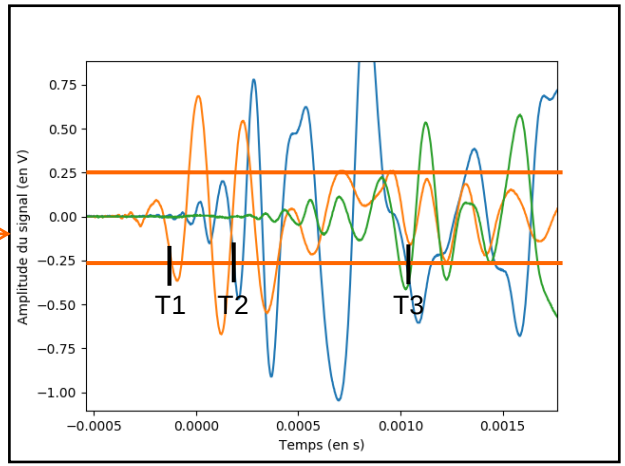




Localisation : principe général

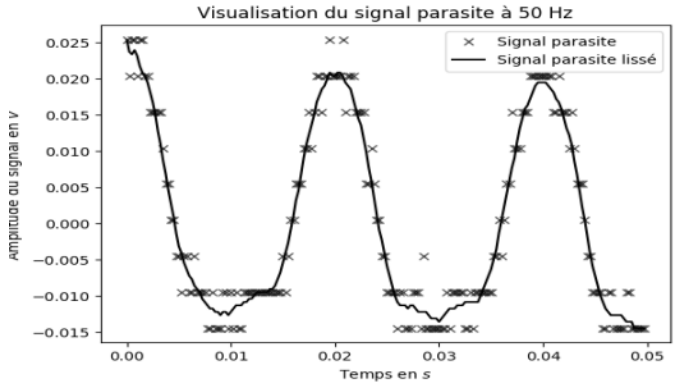




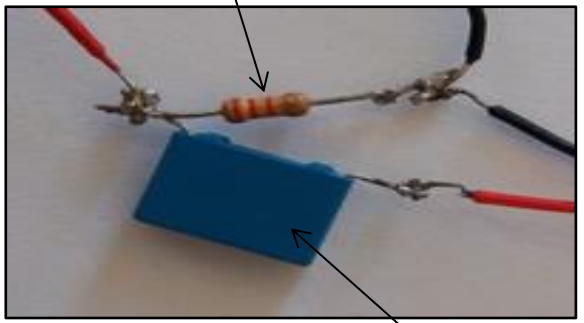


Localisation : Etude 1D

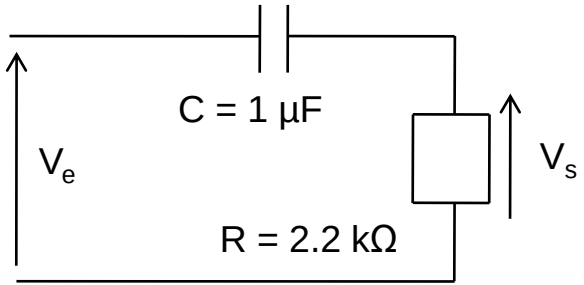
Signal parasite : fréquence f = 50 Hz



Résistance R



Condensateur C

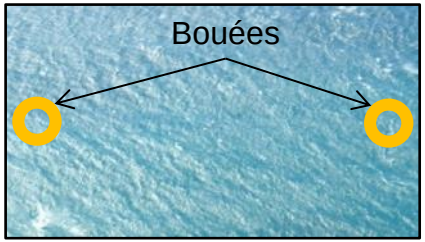


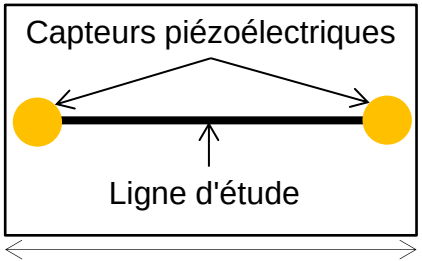
Filtre passe-haut :

Fréquence de coupure :

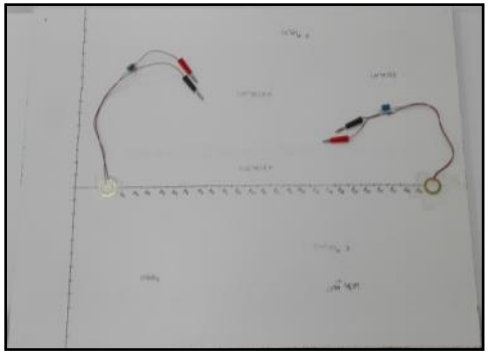
$$f_c = \frac{1}{2\pi RC} = 72{,}3\ Hz$$

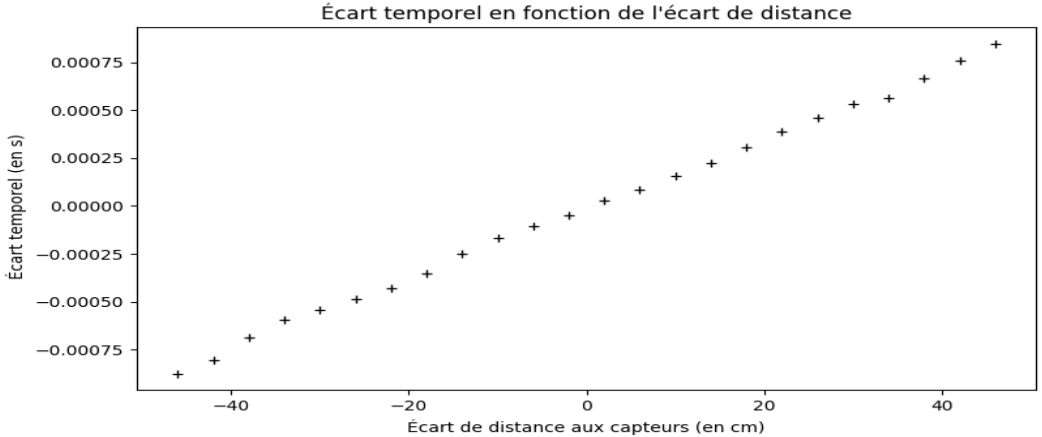
Localisation : Etude 1D





L = 50 cm





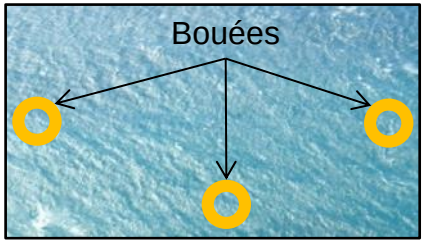
Equation de la droite :

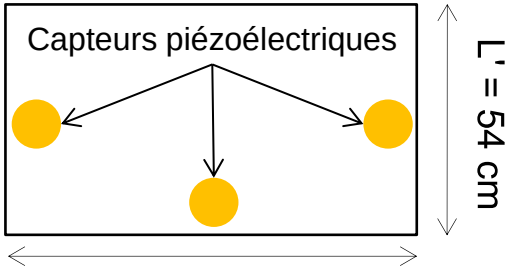
$$Y = 1{,}813.10^{-5}\,X\, - 1{,}4\,.10^{-5}\,secondes$$

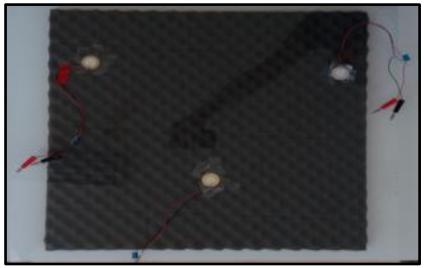
Vitesse des ondes dans le solide :

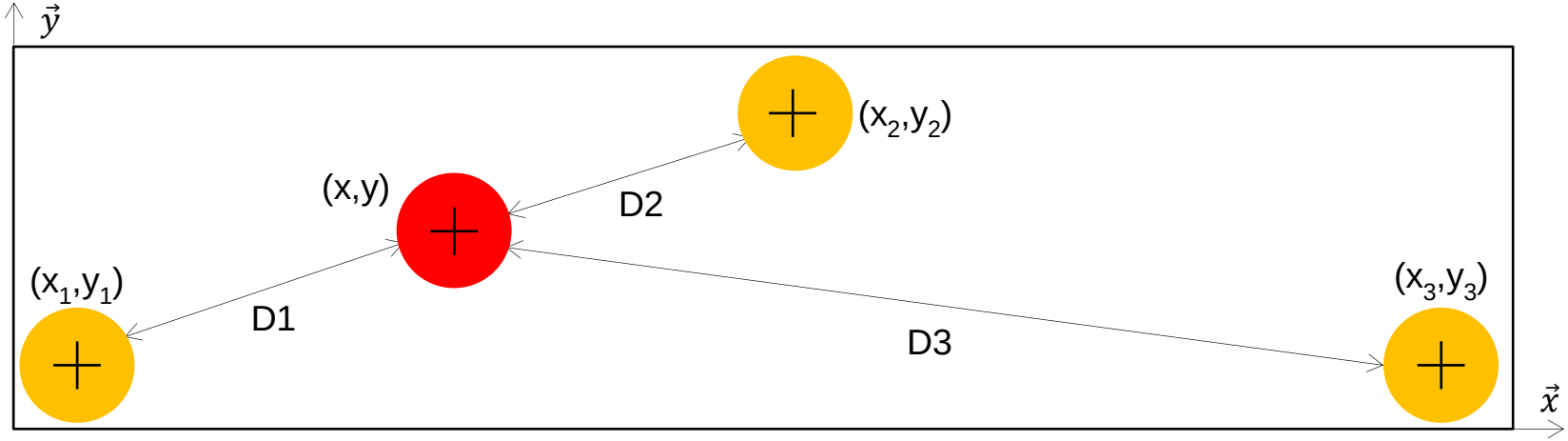
$$V = 552\, \pm 19\, m/s$$

Localisation : Etude 2D

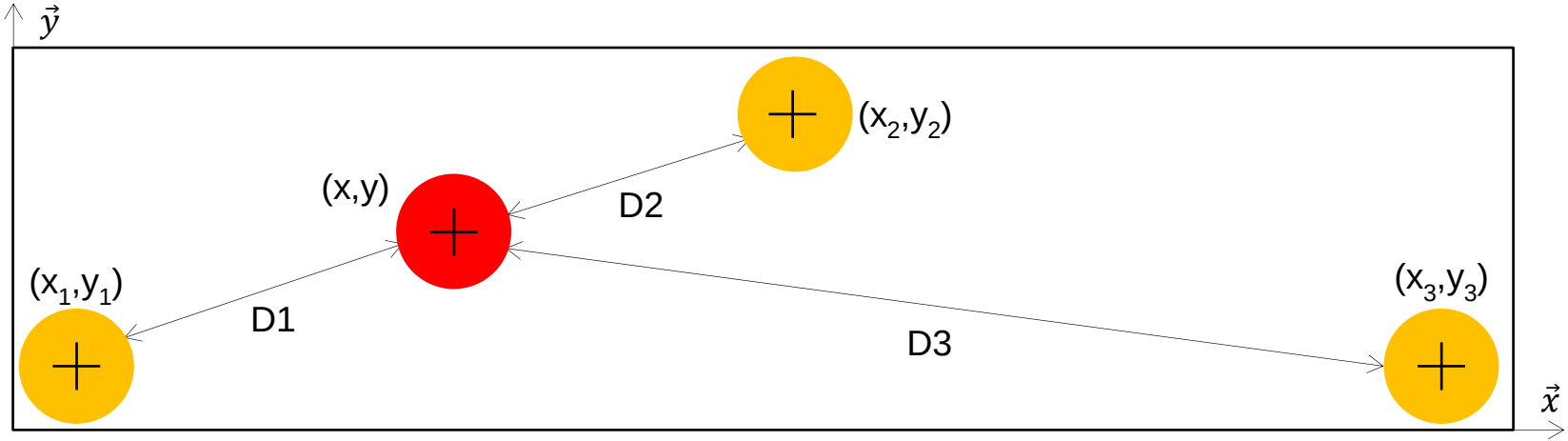






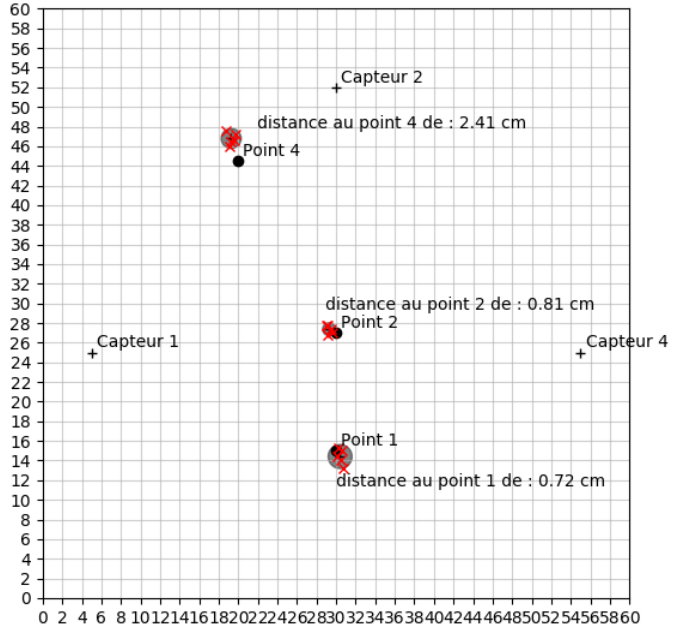


Localisation : Etude 2D

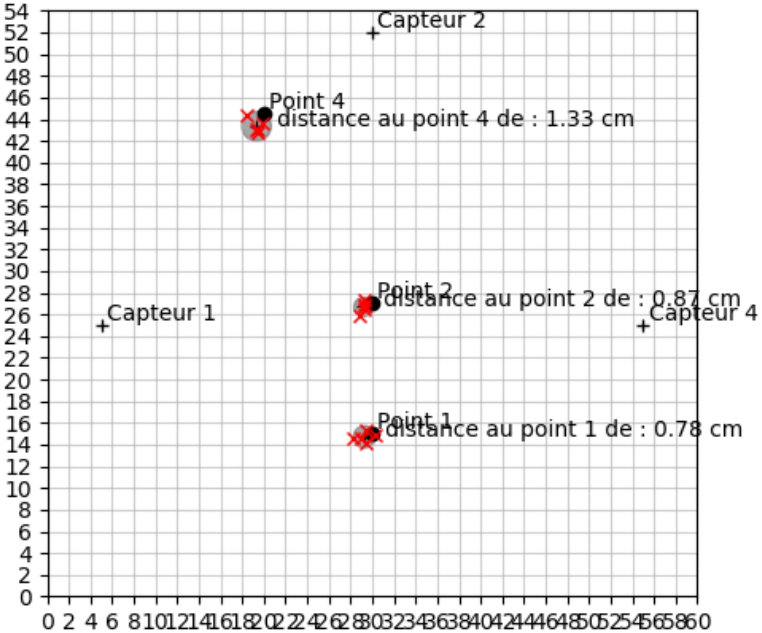


$$\begin{cases} D1 = \sqrt{(x-x_1)^2+(y-y_1)^2} \\ D2 = \sqrt{(x-x_2)^2+(y-y_2)^2} \\ D3 = \sqrt{(x-x_3)^2+(y-y_3)^2} \end{cases} \Rightarrow \begin{cases} D3-D1 = \sqrt{(x-x_3)^2+(y-y_3)^2} - \sqrt{(x-x_1)^2+(y-y_1)^2} \\ D2-D1 = \sqrt{(x-x_2)^2+(y-y_2)^2} - \sqrt{(x-x_1)^2+(y-y_1)^2} \end{cases}$$

Localisation : Etude 2D

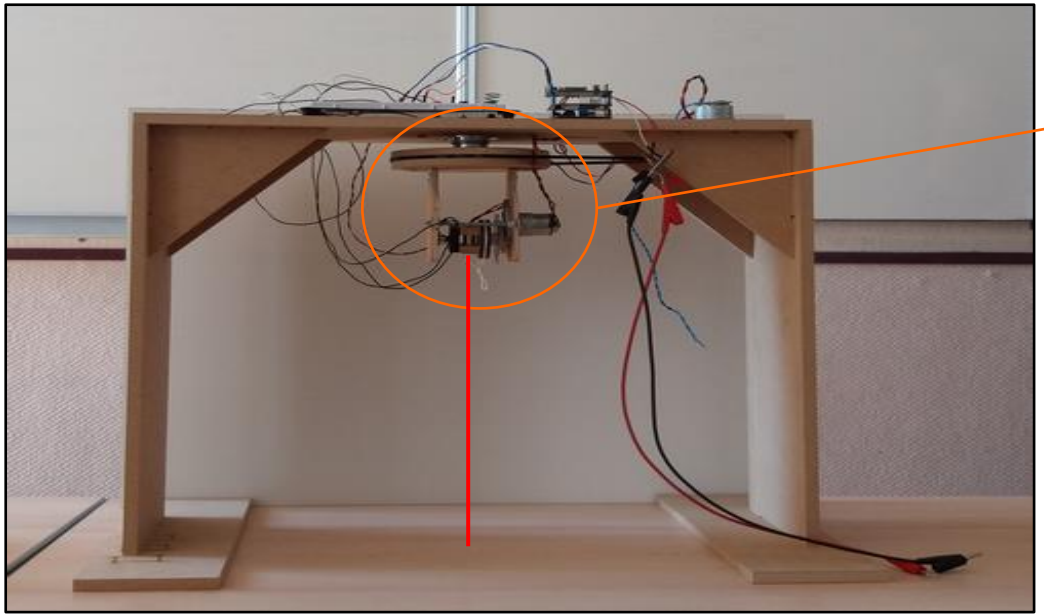


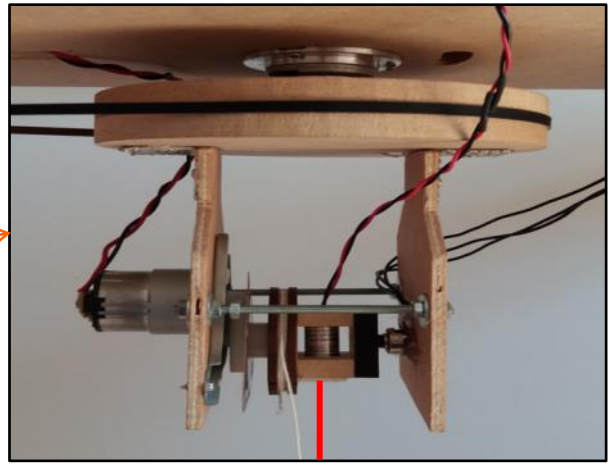
Résultats de la localisation dans le cas du bois

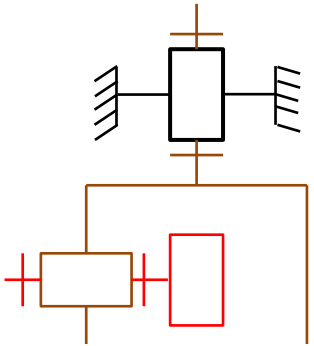


Résultats de la localisation dans le cas du Plexiglas

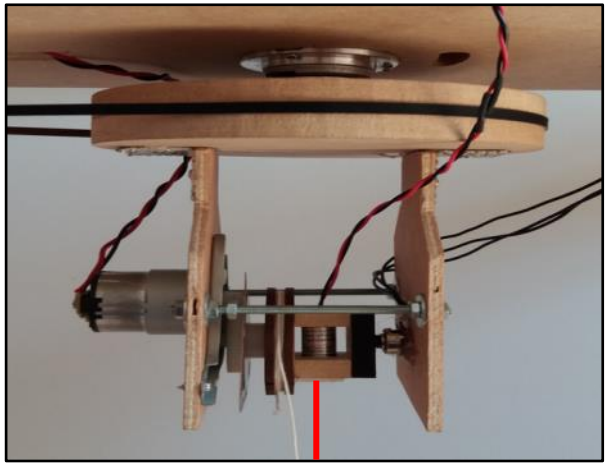
Asservissement

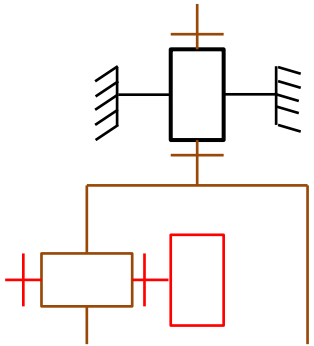






Asservissement

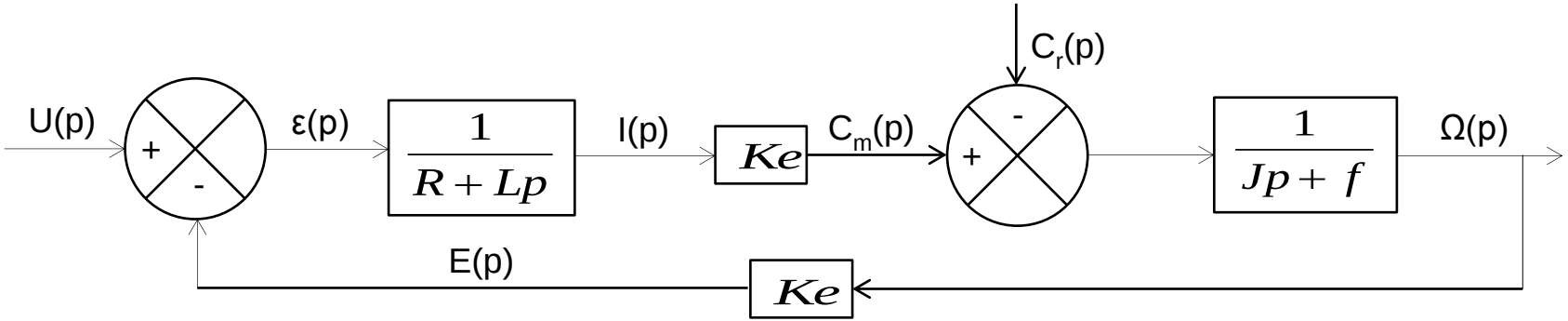




Cahier des charges

Critère	Demande
Précision	$\varepsilon \leq 1\ cm$
Rapidité	$t_{r5\%} \leq 10\ s$
Stabilité	Stable
Dépassement	Sans dépassement

Asservissement : étude des moteurs



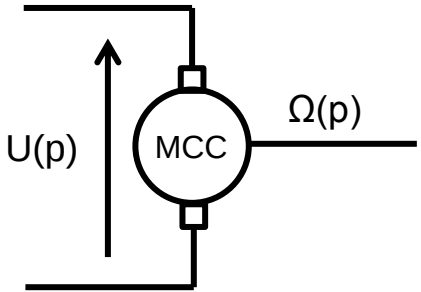
Résistance : $R = \frac{U}{I} = 8{,}458 \pm 0{,}020\ \Omega$

Constante de couplage : $Ke = \frac{E}{\Omega} = (7{,}68 \pm 0{,}20).10^{-3}\ Vs/rad$

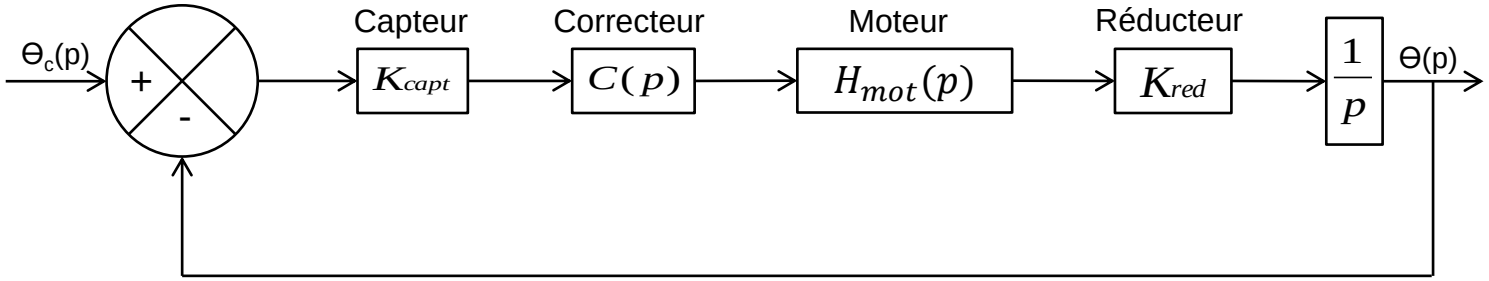
Coefficient de frottement : $f = (1{,}74 \pm 0{,}05).10^{-5}\ Nms/rad$

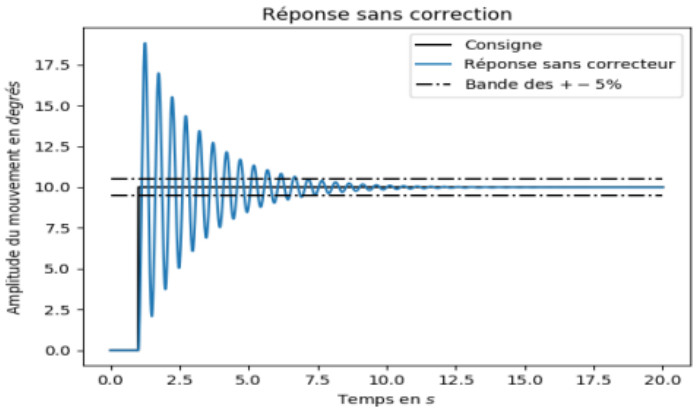
Moment d'inertie : $J = 2{,}57.10^{-5}\ kg.m^2$

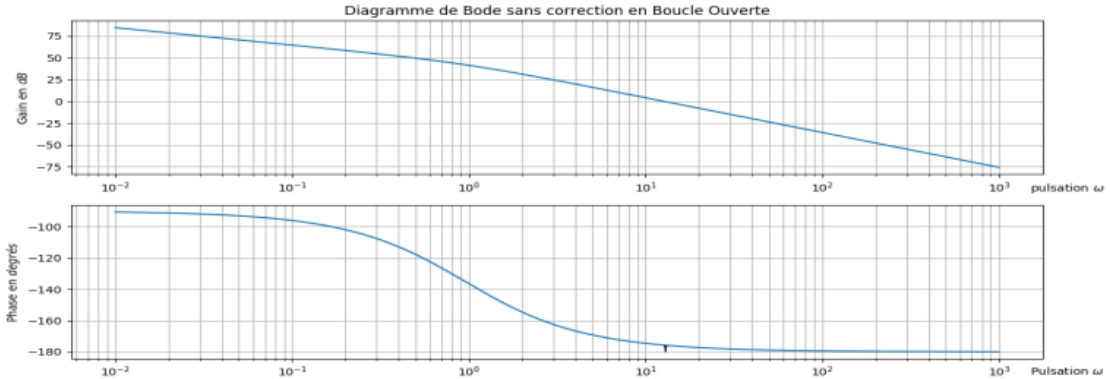
Couple résistant : $C_r = 2{,}5.10^{-4}\ Nm$



Asservissement : Pointage 1D

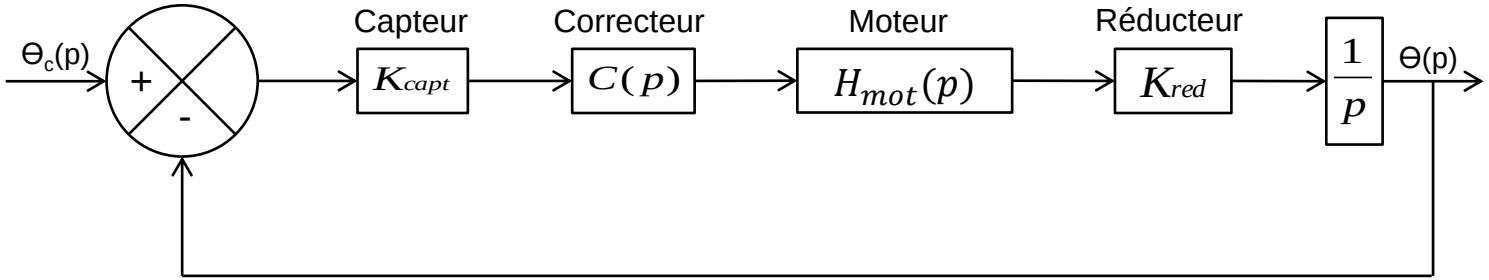




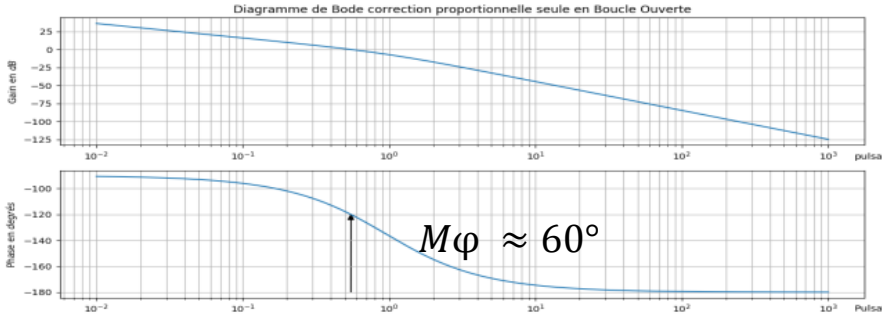


$M\varphi \approx 5°$

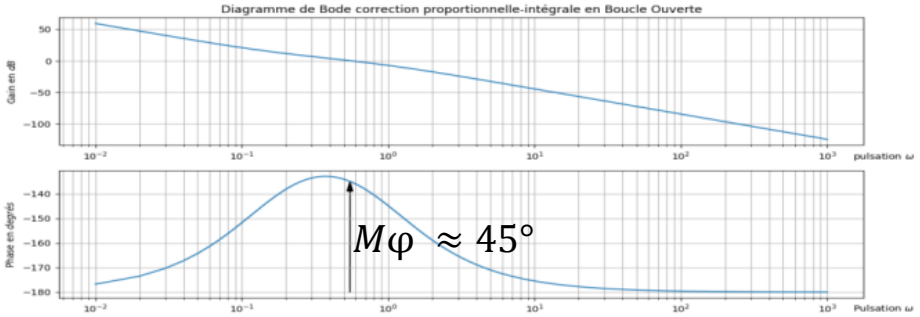
Asservissement : Pointage 1D



Implémentation d'un correcteur proportionnel intégral : $C(p) = Ki * (1 + \frac{1}{Tip})$



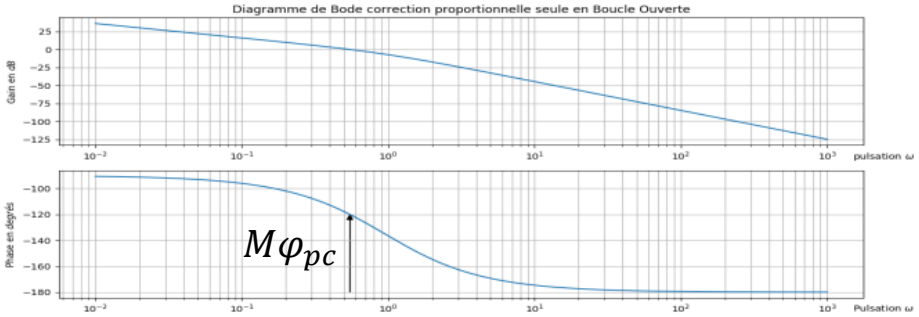
Seulement proportionnel



Proportionnel et intégral

Asservissement : Pointage 1D

Seulement proportionnel



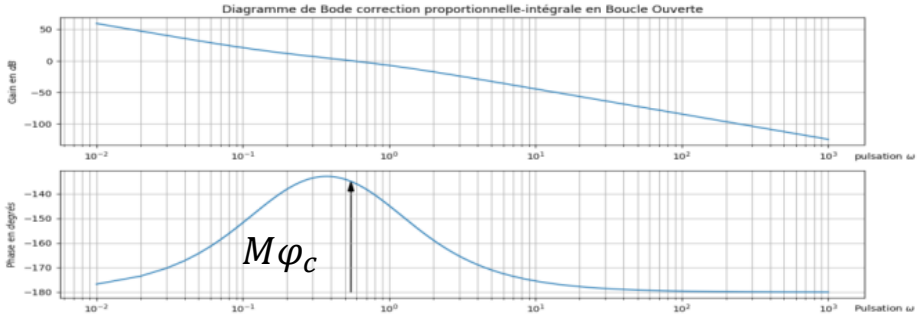
$$M\varphi = \arg(H_{nc}(j\omega_{0dB})) + 180°$$

$$M\varphi_{pc} = \arg\left(H_{nc}\left(j\omega_{0dB,pc}\right)\right) + 180°$$

$$G_{dB}\left(H_{pc}\left(j\omega_{0dB,pc}\right)\right) = 0 = 20.\log(Ki) + G_{dB}\left(H_{nc}\left(j\omega_{0dB,pc}\right)\right)$$

$$Ki = 10^{\frac{-G_{dB}(H_{nc}(j\omega_{0dB,pc}))}{20}}$$

Proportionnel et intégral

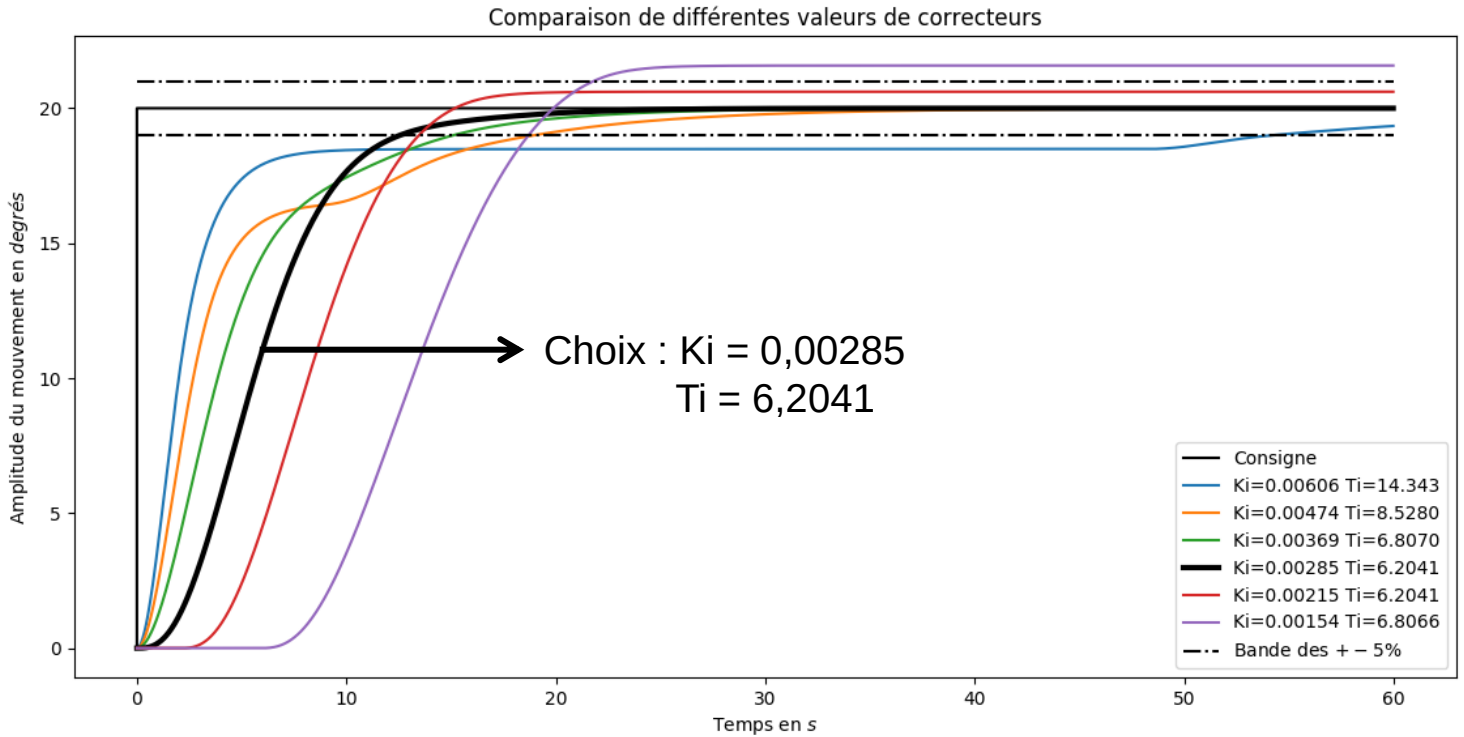


$$M\varphi_c = M\varphi_{pc} + \arg\left(C\left(j\omega_{0dB,pc}\right)\right)$$

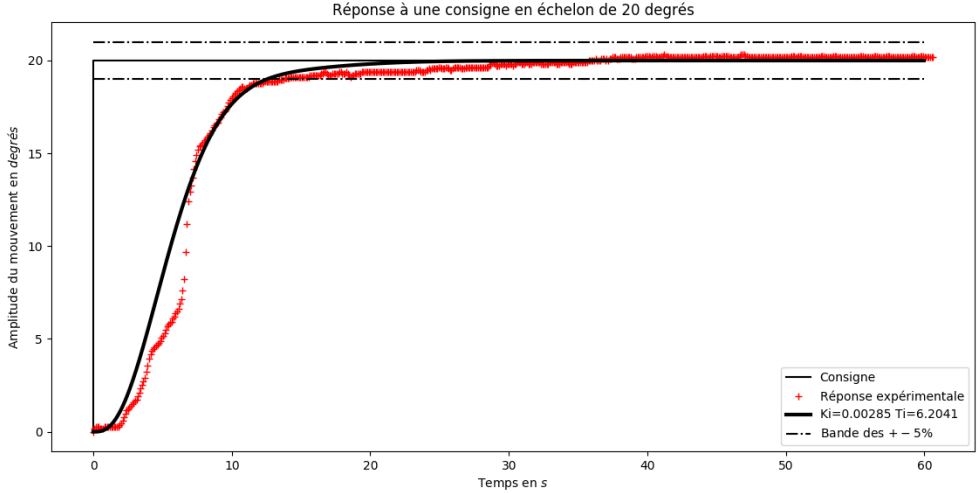
$$\arctan\left(Ti.\,\omega_{0dB,pc}\right) - 90° = M\varphi_c - M\varphi_{pc}$$

$$Ti = \frac{\tan(90° + M\varphi_c - M\varphi_{pc})}{\omega_{0dB,pc}}$$

Asservissement : Pointage 1D

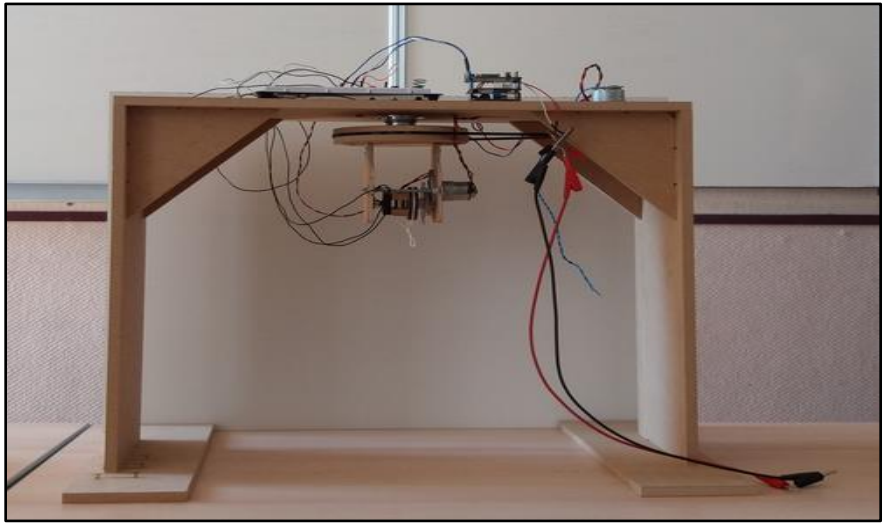


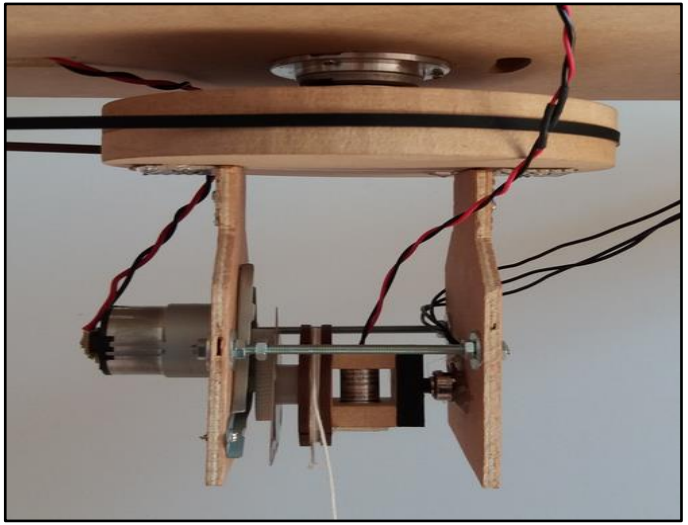
Asservissement : Pointage 1D



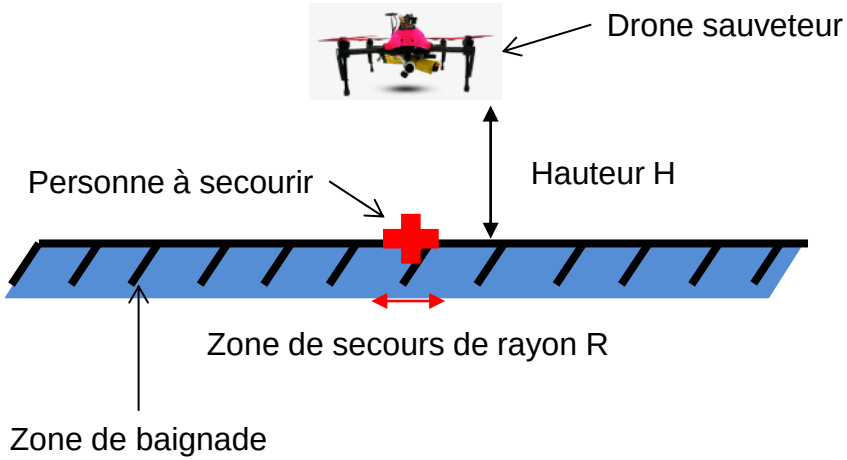
Critère	Demande	Résultat
Précision	$\varepsilon \leq 1\ cm$	$\varepsilon \leq 0{,}5\ cm$
Rapidité	$t_{r5\%} \leq 10\ s$	$t_{r5\%} \leq 12\ s$
Stabilité	Stable	Stable
Dépassement	Sans dépassement	Sans dépassement

Asservissement : Pointage 2D





Conclusion et Limites



Système réel	Maquette
H = 5 m	H = 35 cm
R = 50 cm	R = 3,5 cm

Critère	Conclusion
Stabilité	✓
Dépassement	✓
Précision	✓
Rapidité	✗

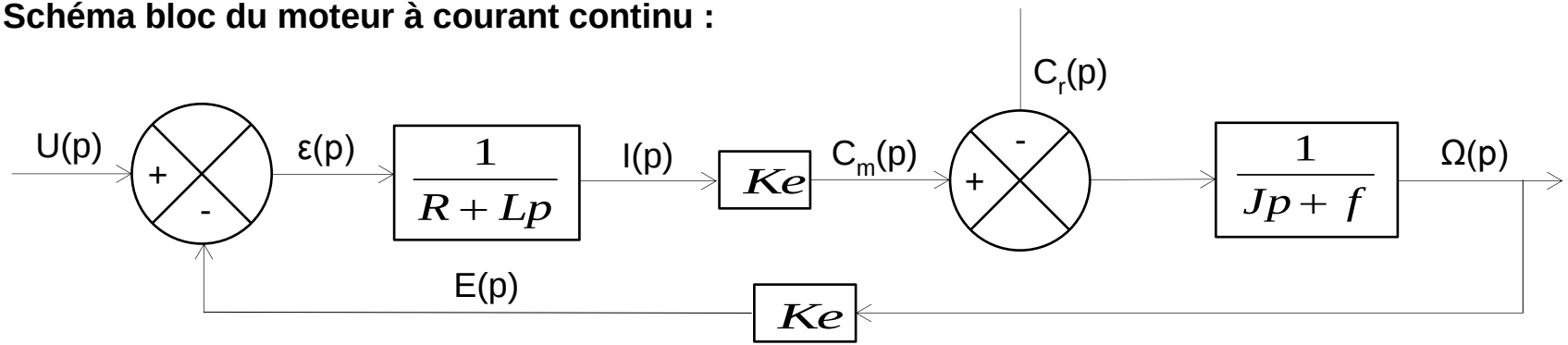
Merci

Annexe 1 : Moteur à courant continu

Equations caractéristiques du moteur à courant continu :

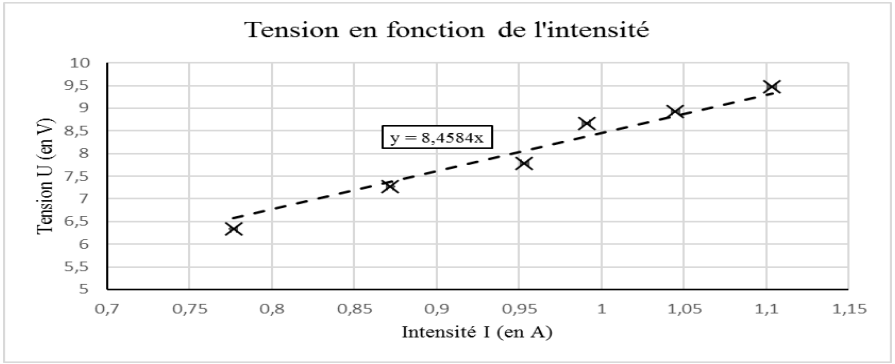
Domaine temporel	Domaine de Laplace
$u(t) = e(t) + R*i(t) + L*di(t)/dt$	$U(p) = E(p) + (R+Lp) * I(p)$
$C_m(t) = Ke * i(t)$	$C_m(p) = Ke * I(p)$
$e(t) = Ke * \omega(t)$	$E(p) = Ke * \Omega(p)$
$J*d\omega(t)/dt = C_m(t) - C_r(t) – f * \omega(t)$	$(Jp + f) * \Omega(p) = C_m(p) - C_r(p)$

Schéma bloc du moteur à courant continu :



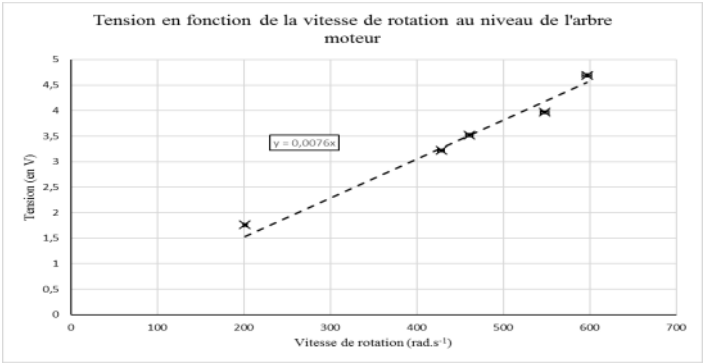
Annexe 2 : Détermination des caractéristiques 1/2

Détermination de la résistance R



$$R = \frac{U}{I} = 8{,}458 \pm 0{,}020\ \Omega$$

Détermination de la constante de couplage Ke



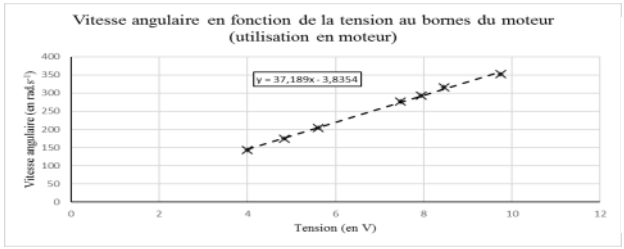
$$Ke = \frac{E}{\Omega} = (7{,}68 \pm 0{,}20) * 10^{-3}\ Vs/rad$$

Annexe 2 : Détermination des caractéristiques 2/2

Par lecture de schéma bloc : $\Omega(p) = \frac{1}{1+p(\underbrace{\frac{RJ}{Ke^2+Rf}}_{\tau})} * (\underbrace{\frac{Ke}{Ke^2+Rf}}_{\alpha} U(p) - \underbrace{\frac{R}{Ke^2+Rf}}_{\beta} C_r(p))$

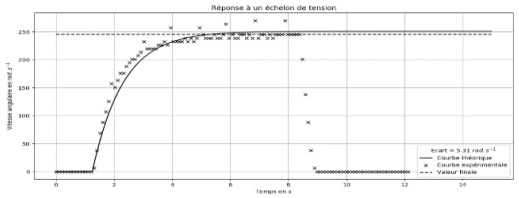
$\tau = \frac{RJ}{Ke^2+Rf}$ $\quad \alpha = \frac{Ke}{Ke^2+Rf}$ $\quad \beta = \frac{R}{Ke^2+Rf}$

Détermination du coefficient de frottement f



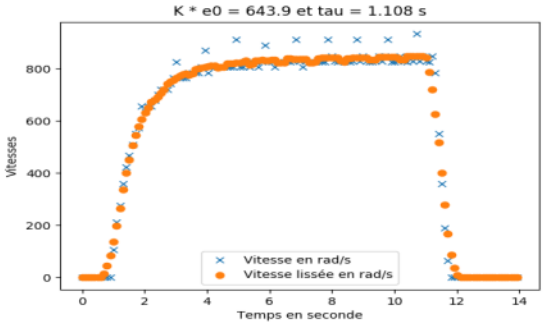
$f = (1{,}74 \pm 0{,}05) * 10^{-5}\ \mu Nms/rad$

Détermination du couple résistant C_r



$C_r = 2{,}5 * 10^{-4}\ Nm$

Détermination du moment d'inertie J



$J = 2{,}57 * 10^{-5}\ kg.m^2$

Annexe 3 : Programme localisation 1D

- Définir l'indice du temps de début d'acquisition

```
def temps_seuil(liste) :
    """Retourne l'indice correspondant au temps de début d'acquisition du signal"""
    j = 0
    while abs(liste[j]) < 0.03 :
        j = j+1
    return j
```

- Calculer l'écart temporel entre les débuts d'acquisition des 2 capteurs

```
def valeur_moyenne_case_k(k) :
    donnees = [0]*len(L[k-1])
    temps_1 = [0]*len(L[k-1])
    temps_2 = [0]*len(L[k-1])
    amplitude_1 = [0]*len(L[k-1])
    amplitude_2 = [0]*len(L[k-1])

    temps_initial = [0]*len(L[k-1])
    temps_retard = [0]*len(L[k-1])
    ecart = [0]*len(L[k-1])
    """Initialise la taille des listes."""

    for i in range (len(L[k-1])) :
        chemin='A'+str(k)+'\\'
        donnees[i] = np.loadtxt(chemin+L[k-1][i], skiprows = 1)
        temps_1[i] = donnees[i][:,0]
        temps_2[i] = donnees[i][:,2]
        amplitude_1[i] = donnees[i][:,1]
        amplitude_2[i] = donnees[i][:,3]
    """Recupère les différentes gradeurs mesurées par les capteurs & associe à chaque colonne une grandeur"""

    for j in range (len(L[k-1])) :
        temps_initial[j] = temps_1[j][temps_seuil(amplitude_1[j])]
        temps_retard[j] = temps_2[j][temps_seuil(amplitude_2[j])]
        ecart[j] = temps_initial[j]-temps_retard[j]
    """Donne l'écart temporel entre les signaux des deux capteurs"""

    return(ecart)
```

- Retourner la moyenne des écarts temporels pour chaque case

```
def ecart_moyen() :
    """Retourne la liste contenant la moyenne des écarts temporels pour chaque case"""
    M = []
    for i in range (1,25) :
        somme_ecart_case_k = 0
        for j in range (0,5) :
            somme_ecart_case_k += valeur_moyenne_case_k(i)[j]
        ecart_moyen_case_k = (somme_ecart_case_k)/5
        M.append(ecart_moyen_case_k)
    return M
```

- Tracer le graphique pour obtenir la vitesse des ondes dans le solide

```
plt.figure()
abscisse = ecart_dist
ordonnee = ecart_moy
plt.plot(abscisse, ordonnee, '+', color = 'black') #Trace l'écart moyen en fonction de la case
plt.xlabel('Écart de distance aux capteurs (en cm)')
plt.ylabel(' Écart temporel (en s)')
plt.title("Écart temporel en fonction de l'écart de distance")
plt.show(False)

from scipy.stats import linregress
pente, ordonnee_a_origine, coefficient_de_regression, p_value, erreur_standard = linregress(abscisse, ordonnee)
print(pente, ordonnee_a_origine, coefficient_de_regression)
print("La droite a pour équation y=" + str(pente) + "x" + str(ordonnee_a_origine))
print("La vitesse des ondes dans le solide est de " + str(10**(-2)/pente) + " m/s")
```

Annexe 4 : Programme localisation 2D

- Retourner les temps de début d'acquisition des 3 capteurs

```
def recupere_donnees() :
    donnees = np.loadtxt("Point1_11.txt", skiprows = 1) #Charge les données reçues sous forme de tableau
    temps_1_recu = donnees[:,0] #Liste associée au temps du capteur 1
    temps_2_recu = donnees[:,2] #Liste associée au temps du capteur 2
    temps_3_recu = donnees[:,4] #Liste associée au temps du capteur 3
    amplitude_1_recu = donnees[:,1] #Liste associée à l'amplitude reçue par le capteur 1
    amplitude_2_recu = donnees[:,3] #Liste associée à l'amplitude reçue par le capteur 2
    amplitude_3_recu = donnees[:,5] #Liste associée à l'amplitude reçue par le capteur 3

    temps_deb_1 = temps_1_recu[temps_seuil(amplitude_1_recu)] #Temps de début d'acquisition du signal par le capteur 1
    temps_deb_2 = temps_2_recu[temps_seuil(amplitude_2_recu)] #Temps de début d'acquisition du signal par le capteur 2
    temps_deb_3 = temps_3_recu[temps_seuil(amplitude_3_recu)] #Temps de début d'acquisition du signal par le capteur 3
    print(temps_deb_1, temps_deb_2, temps_deb_3)
    return temps_deb_1, temps_deb_2, temps_deb_3
```

- Retourner les coordonnées recherchées

```
T1, T2, T3 = recupere_donnees()


D1 = vitesse*T1
D2 = vitesse*T2
D3 = vitesse*T3


def f(L) : #L est la liste contenant les coordonnées du point recherché : L élément de R**2
    x,y = L[0],L[1]
    return [ sqrt((x-x3)**2 + (y-y3)**2) - sqrt((x-x1)**2 + (y-y1)**2) - (D3-D1),
             sqrt((x-x3)**2 + (y-y3)**2) - sqrt((x-x2)**2 + (y-y2)**2) - (D3-D2) ]


coordonnées = fsolve(f,[25,25])
print(coordonnées)
```