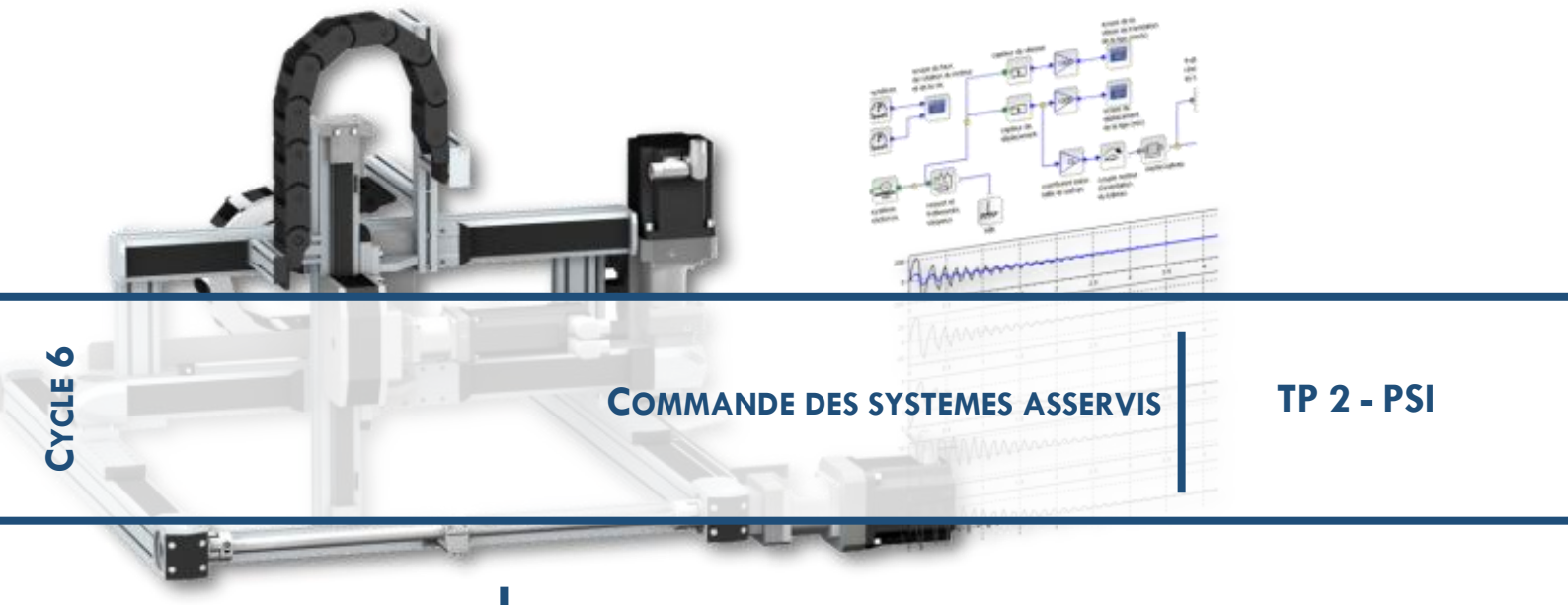


CYCLE 6

COMMANDE DES SYSTEMES ASSERVIS

TP 2 - PSI

Modélisation d'une MCC – Méthode d'Euler

1 PRESENTATION SIMULATION D'UN MCC

Il faut passer par un système de deux équations différentielles du premier ordre pour avoir i et w :

$$\begin{cases} u(t) = e(t) + Ri(t) + L\dfrac{di(t)}{dt} \\ e(t) = k_e\omega(t) \\ c_m(t) = k_c i(t) \\ c_f(t) = f\omega(t) \\ c_m(t) - c_f(t) - c_r(t) = J\dfrac{d\omega(t)}{dt} \end{cases} \quad ; \quad \begin{cases} \dfrac{di(t)}{dt} = \dfrac{u(t) - k_e\omega(t) - Ri(t)}{L} \\ \dfrac{d\omega(t)}{dt} = \dfrac{k_c i(t) - f\omega(t) - c_r(t)}{J} \end{cases}$$

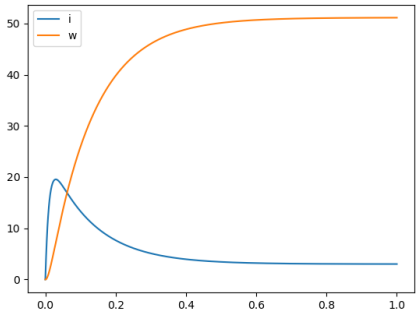
Question 1: Programmer la résolution Euler explicite des équations pour un échelon de tension de 10V sur 1 secondes avec 1000 points (on ne manipulera que des listes)

```
def Euler_Explicite(f,y0,t0,t1,dt):
    t = t0
    y = y0
    T = [t]
    Y = [y]
    i = 0
    while t < t1: # Important: < et non <=
        yp = f(y,t)
        y = [y[i] + yp[i]*dt for i in range(len(y))]
        i += 1
        t = t0 + i*dt # Important: Eviter les erreurs de t+=dt
        T.append(t)
        Y.append(y)
    return T,Y

def F(Y,t): # u variable globale
    R = 0.45
    L = 0.0046
    ke = 0.169
    kc = 0.17
    f = 0.01
```

```
    J = 0.01
    cr = 0
    i,w = Y
    di = (u-ke*w-R*i)/L
    dw = (kc*i-f*w-cr)/J
    Sol = [di,dw]
    return Sol

i0 = 0
w0 = 0
V0 = [i0,w0]
t0 = 0
t1 = 1
N = 1000
u = 10

dt = (t1-t0)/(N-1)
Lt,Y = Euler_Explicite(F,V0,t0,t1,dt)
Li = [Y[i][0] for i in range(len(Y))]
Lw = [Y[i][1] for i in range(len(Y))]
```

Question 2: Tracer l'évolution de la vitesse moteur et de son intensité en fonction du temps

```
import matplotlib.pyplot as plt
plt.close('all')
def f_Affiche(fig,Lx,Ly,Legende):
    plt.figure(fig)
    plt.plot(Lx,Ly,label=Legende)
    plt.legend()
    plt.show()
    plt.pause(0.0001)

f_Affiche(1,Lt,Li,'i')
f_Affiche(1,Lt,Lw,'w')
```

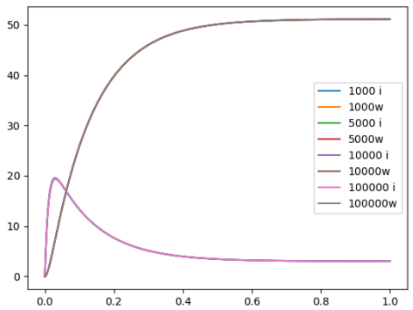


Question 3: Créer la fonction MCC(V0,T0,T1,dt) prenant en argument la liste d'état initial du MCC V0=[i0,w0] (intensité, vitesse), les temps T0 de début et T1 de fin de résolution et dt le pas de temps de la résolution d'Euler, et renvoyant les listes Lt, Li et Lw des temps, intensités et vitesses moteur

```
def MCC(V0,t0,t1,dt):
    Lt,Y = Euler_Explicite(F,V0,t0,t1,dt)
    Li = [Y[i][0] for i in range(len(Y))]
    Lw = [Y[i][1] for i in range(len(Y))]
    return Lt,Li,Lw
```

Question 4: Déterminer le nombre de points à utiliser pour obtenir une solution assez fiable sans que les calculs soient trop longs

```
for N in [1000,5000,10000,100000]:
    dt = (t1-t0)/(N-1)
    Lt,Li,Lw = MCC(V0,t0,t1,dt)
    f_Affiche(2,Lt,Li,str(N)+' i')
    f_Affiche(2,Lt,Lw,str(N)+'w')
```



En gros, à partir de 5000 points, ça évolue très peu.

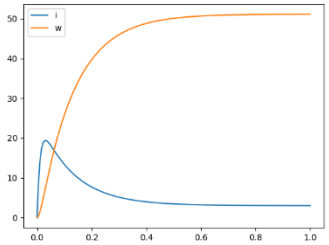
Question 5: En utilisant MCC, déterminer le pas de temps de simulation dts permettant de supposer que le résultat de la résolution discrète d'Euler est comparable à une résolution continue

```
N = 5000
dts = (t1-t0)/(N-1)
print('dts: ',dts)
```

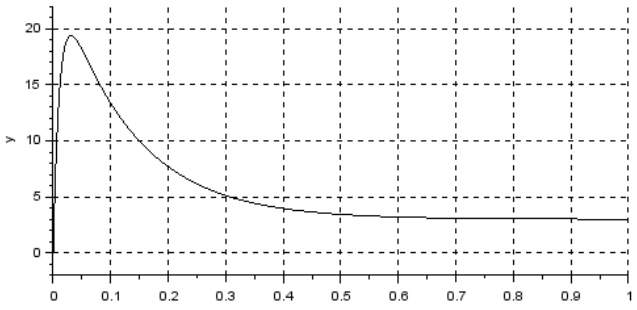
$$dts \approx 2.10^{-4}$$

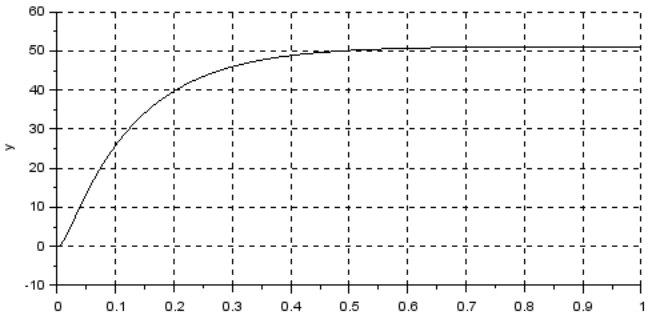
Question 6: Réaliser la résolution précédente en utilisant « odeint »

```
Lt = [t0+i*dts for i in range(N)]
from scipy.integrate import odeint
Sol = odeint(F,V0,Lt)
Li = [Sol[i][0] for i in range(len(Sol))]
Lw = [Sol[i][1] for i in range(len(Sol))]
f_Affiche(3,Lt,Li,'i')
f_Affiche(3,Lt,Lw,'w')
```



Question 7: Réaliser le modèle XCOS du MCC et comparer les résultats obtenus





2 SIMULATION D'ASSERVISSEMENT DU MCC

2.1 Objectif 1 – Simulation numérique d'un asservissement

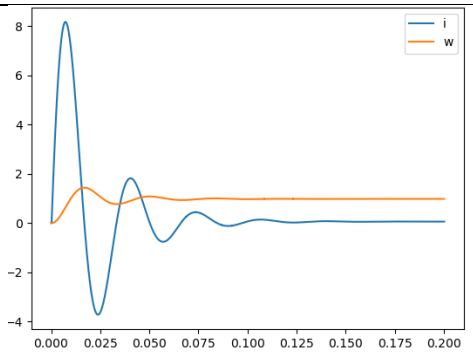
Question 1: Mettre en place le code permettant de réaliser la simulation de l'asservissement sur 0,2 seconde

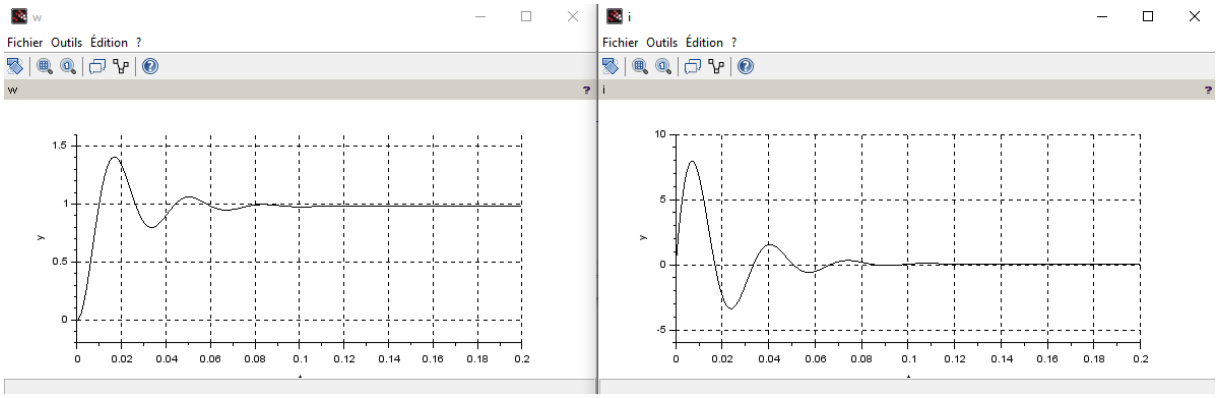
```
# Données de l'asservissement

Wc = 1      # Consigne (rd/s)
Kcapt = 10 # Gain capteur (V/rd/s)
Ka = Kcapt # Adaptateur (V/rd/s)
Uc = Ka*Wc # Consigne au comparateur (V)

# Données numériques

dts = 0.0002 # Pas de temps résolution MCC (s)
ti = 0       # Temps initial simulation
```

```
tf = 0.2      # Temps final simulation
n = 1         # Facteur sur le pas de temps utile pour les 2 objectifs
dt = n*dts    # Pas de temps de simulation

# Initialisation des données

i0 = 0
w0 = 0
u0 = 0
Li = [i0]
Lw = [w0]
Lu = [u0]
Lt = [ti]
Ltu = [ti]

# Boucle de simulation

k = 0 # Ne  pas mettre i
t = ti
while t < tf:
    i = Li[-1]
    w = Lw[-1]
    V = [i,w]
    k += 1
    t = ti + k*dt # Pas très important ici par rapport à t+=dt
    r = w * Kcapt
    eps = Uc - r
    u = eps # On mettra un correcteur ici
    Ltu.append(t)
    Lu.append(u)
    lt,li,lw = MCC(V,t,t+dt,dts)
    Li += li
    Lw += lw
    Lt += lt
```

Question 2: Tracer l'évolution de l'intensité et de la vitesse du moteur au cours du temps et les comparer à une simulation du même système réalisée sur XCOS

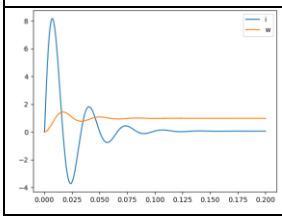
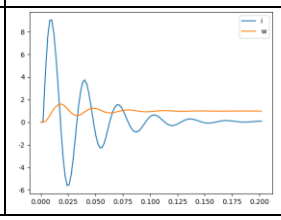
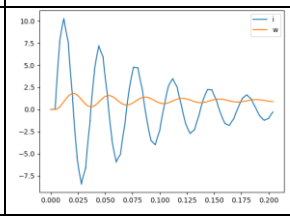
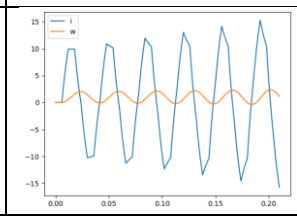
```
f_Affiche(3,Lt,Li,'i')
f_Affiche(3,Lt,Lw,'w')
```





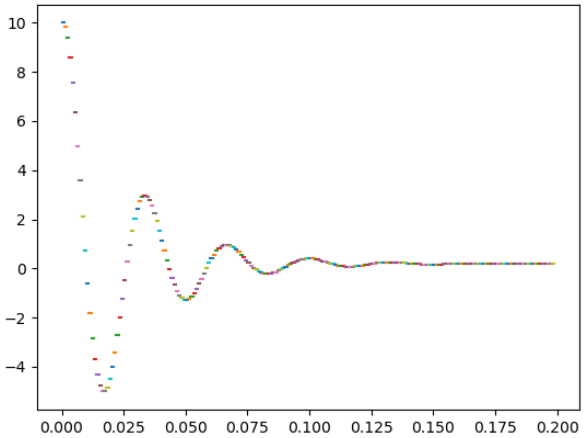
2.2 Objectif 2 – Effets d'un asservissement numérique sur un système réel

Question 3: En adaptant légèrement le code réalisé dans l'objectif 1, simuler l'asservissement du MCC pour des valeurs de n dans [1,10,20,30] – On tracera la tension de consigne du moteur sous la forme d'une courbe en marches d'escaliers

1	10	20	30
			

```
def Affiche_Creneaux(fig,X,Y):
    plt.figure(fig)
    for i in range(len(X)-2):
        plt.plot(X[i:i+2],[Y[i+1],Y[i+1]])
    plt.show()

Affiche_Creneaux(4,Ltu,Lu)
```

Exemple de courbe de tension moteur :



Question 4: Conclure sur l'effet de la fréquence de traitement de la carte de commande sur le comportement du système

Lorsque la fréquence de traitement de la carte de commande est trop faible, on fait apparaître du retard et de l'instabilité dans les réponses.

3 CORRECTEURS NUMERIQUES

On pourrait directement résoudre cela avec Euler si les objectifs n'étaient que la simulation d'un asservissement sur ordinateur (ex avec correcteur PI):

$$\frac{du(t)}{dt} = K_p \frac{de(t)}{dt} + K_i e(t) \quad ; \quad e(t) = U_c - K_{capt}\,\omega(t)$$

$$\frac{du(t)}{dt} = -K_p K_{capt} \frac{d\,\omega(t)}{dt} + K_i \left(U_c - K_{capt}\,\omega(t)\right)$$

$$\begin{cases} \dfrac{di(t)}{dt} = \dfrac{u(t) - k_e\omega(t) - Ri(t)}{L} \\ \dfrac{d\omega(t)}{dt} = \dfrac{k_c i(t) - f\omega(t) - c_r(t)}{J} \\ \dfrac{du(t)}{dt} = -K_p K_{capt} \dfrac{d\,\omega(t)}{dt} + K_i \left(U_c - K_{capt}\,\omega(t)\right) \end{cases}$$

Question 5: En utilisant la méthode des équations aux différentes, déterminer pour ces trois correcteurs l'expression de u_k en fonction de u_{k-1}, e_k et e_{k-1}

<table>
<tr><td colspan="2">Correcteur proportionnel : K_p</td></tr>
<tr><td colspan="2">$u(t) = K_p e(t)$</td></tr>
<tr><td colspan="2">$u_k = K_p e_k$</td></tr>
<tr><td colspan="2">Correcteur PI : $K_p + \frac{K_i}{p}$</td></tr>
<tr><td>$\frac{U(p)}{E(p)} = K_p + \frac{K_i}{p}$
$U(p) = K_p E(p) + K_i \frac{E(p)}{p}$
$u(t) = K_p e(t) + K_i \int_0^t e(u)du$
CIN : La primitive $E(t)$ de $e(t)$ s'annule en 0 avec $E(t) = \int_0^t e(u)du$
$u(t) = K_p e(t) + K_i \int_0^{t-T_e} e(u)du + K_i \int_{t-T_e}^t e(u)du$
On avait au pas précédent :
$u(t-T_e) = K_p e(t-T_e) + K_i \int_0^{t-T_e} e(u)du$
Soit :
$\int_0^{t-T_e} e(u)du = \frac{u(t-T_e) - K_p e(t-T_e)}{K_i}$
Par ailleurs, en prenant l'intégration à droite :
$\int_{t-T_e}^t e(u)du = T_e e(t)$
On obtient :
$u(t) = K_p e(t) + K_i \frac{u(t-T_e) - K_p e(t-T_e)}{K_i} + K_i T_e e(t)$
$u(t) = u(t-T_e) + (K_p + K_i T_e)e(t) - K_p e(t-T_e)$</td><td>$\frac{U(p)}{E(p)} = \frac{K_p p + K_i}{p}$
$pU(p) = K_p pE(p) + K_i E(p)$
$\frac{du(t)}{dt} = K_p \frac{de(t)}{dt} + K_i e(t)$
$\frac{u(t) - u(t-T_e)}{T_e} = K_p \frac{e(t) - e(t-T_e)}{T_e} + K_i e(t)$
$u(t) - u(t-T_e) = (K_p + T_e K_i)e(t) - K_p e(t-T_e)$
$u(t) = u(t-T_e) + (K_p + T_e K_i)e(t) - K_p e(t-T_e)$

On remarque qu'il vaut mieux éviter de faire apparaître les intégrations</td></tr>
<tr><td colspan="2">$u_k = u_{k-1} + (K_p + T_e K_i)e_k - K_p e_{k-1}$
$u_k = A u_{k-1} + B e_k - C e_{k-1} \quad ; \quad \begin{cases} A = 1 \\ B = K_p + T_e K_i \\ C = K_p \end{cases}$</td></tr>
</table>

Correcteur à avance de phase : $\frac{1+aTp}{1+Tp}$; $a > 1$

$$\frac{U(p)}{E(p)} = \frac{1+aTp}{1+Tp}$$
$$E(p) + aTpE(p) = U(p) + TpU(p)$$
$$e(t) + aT\frac{de(t)}{dt} = u(t) + T\frac{du(t)}{dt}$$
$$e(t) + aT\frac{e(t) - e(t-T_e)}{T_e} = u(t) + T\frac{u(t) - u(t-T_e)}{T_e}$$
$$T_e e(t) + aTe(t) - aTe(t-T_e) = T_e u(t) + Tu(t) - Tu(t-T_e)$$
$$(T_e + aT)e(t) - aTe(t-T_e) = (T_e + T)u(t) - Tu(t-T_e)$$
$$(T_e + T)u(t) = (T_e + aT)e(t) - aTe(t-T_e) + Tu(t-T_e)$$
$$u(t) = \frac{T_e + aT}{T_e + T}e(t) - \frac{aT}{T_e + T}e(t-T_e) + \frac{T}{T_e + T}u(t-T_e)$$

$$u_k = \frac{T}{T_e + T}u_{k-1} + \frac{T_e + aT}{T_e + T}e_k - \frac{aT}{T_e + T}e_{k-1}$$

$$u_k = Au_{k-1} + Be_k - Ce_{k-1} \quad ; \quad \begin{cases} A = \frac{T}{T_e + T} \\ B = \frac{T_e + aT}{T_e + T} \\ C = \frac{aT}{T_e + T} \end{cases}$$

Question 6: Proposer les trois fonctions python CP, CPI et CAP des correcteurs numériques étudiés en précisant les arguments nécessaires

P

```
def CP(ek,Kp):
    uk = Kp*ek
    return uk
```

PI

```
def CPI(ukm,ek,ekm,Te,Kp,Ki):
    A = 1
    B = Kp+Te*Ki
    C = Kp
    uk = A*ukm + B*ek - C*ekm
    return uk
```

AP

```
def CAP(ukm,ek,ekm,Te,a,T):
    A = T/(Te+T)
    B = (Te+a*T)/(Te+T)
    C = a*T/(Te+T)
    uk = A*ukm + B*ek - C*ekm
    return uk
```

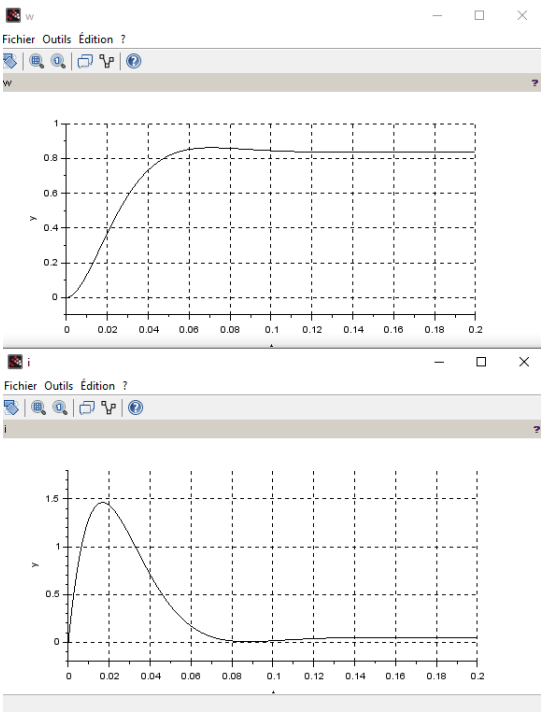
Question 7: En utilisant le code de l'exercice 2, ajouter ces 3 correcteurs à l'asservissement numérique réalisé et vérifier leur fonctionnement avec XCOS

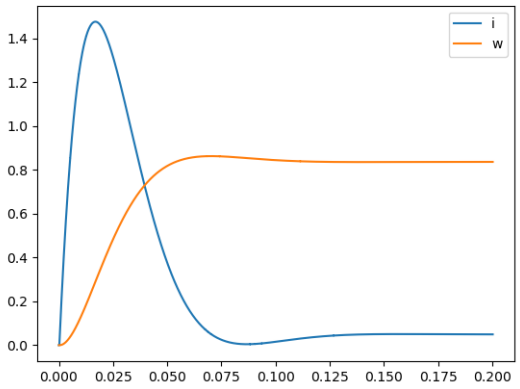
Code pour les 3 simulations :

```
i0 = 0
w0 = 0
u0 = 0
Li = [i0]
Lw = [w0]
Lu = [u0]
Lt = [ti]
Ltu = [ti]

# Boucle de simulation

k = 0
t = ti
u = u0
ek = 0 # Ou Uc
while t < tf:
    i = Li[-1]
    w = Lw[-1]
    V = [i,w]
    k += 1
    t = ti + k*dt
    r = w * Kcapt
    ekm = ek
    ek = Uc - r
    ukm = u
    # u = CP(ek,0.1)
    u = CPI(ukm,ek,ekm,dt,0.4,7.29)
    # u = CAP(ukm,ek,ekm,dt,2.04,0.0038)
    Ltu.append(t)
    Lu.append(u)
    lt,li,lw = MCC(V,t,t+dt,dts)
    Li += li
    Lw += lw
    Lt += lt

f_Affiche(5,Lt,Li,'i')
f_Affiche(5,Lt,Lw,'w')
Affiche_Creneaux(6,Ltu,Lu)
```

Correction proportionnelle

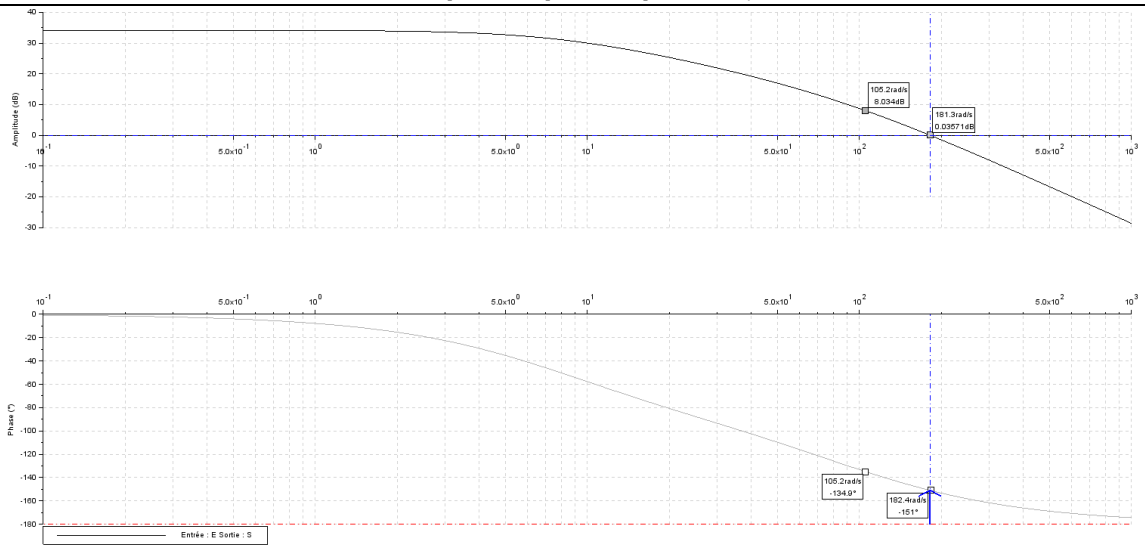
$K_p = 0{,}1$





Correction PI

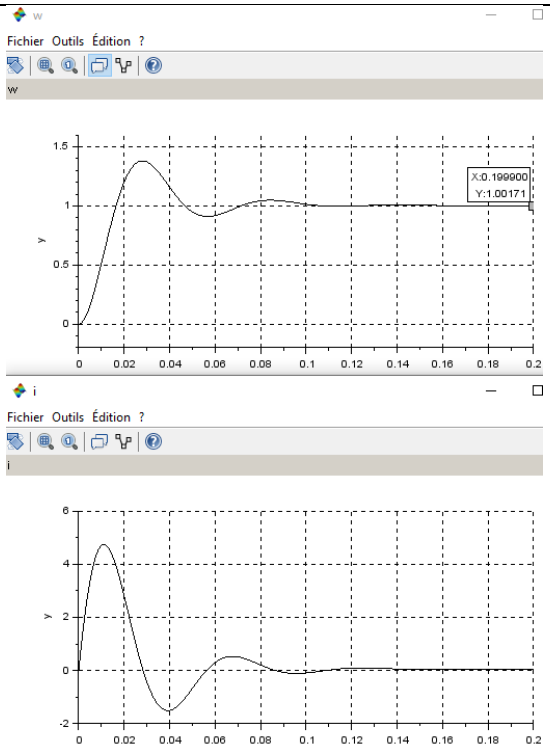
$$K_p + \frac{K_i}{p} = \frac{K_p p + K_i}{p} = \frac{K_i}{p}\left(1 + \frac{K_p}{K_i} p\right)$$

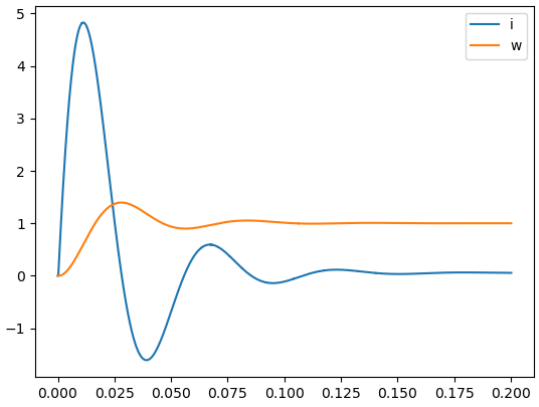


$$\omega_{c_0} = 182{,}3\ rd.s^{-1}$$

$$\frac{K_i}{K_p} = \frac{\omega_{c_0}}{10} = 18{,}23 \quad ; \quad 20\log K_p = TG = -8$$

$$K_p = 10^{-\frac{8}{20}} = 0{,}4 \quad ; \quad K_i = 18{,}23 K_p = 7{,}29$$

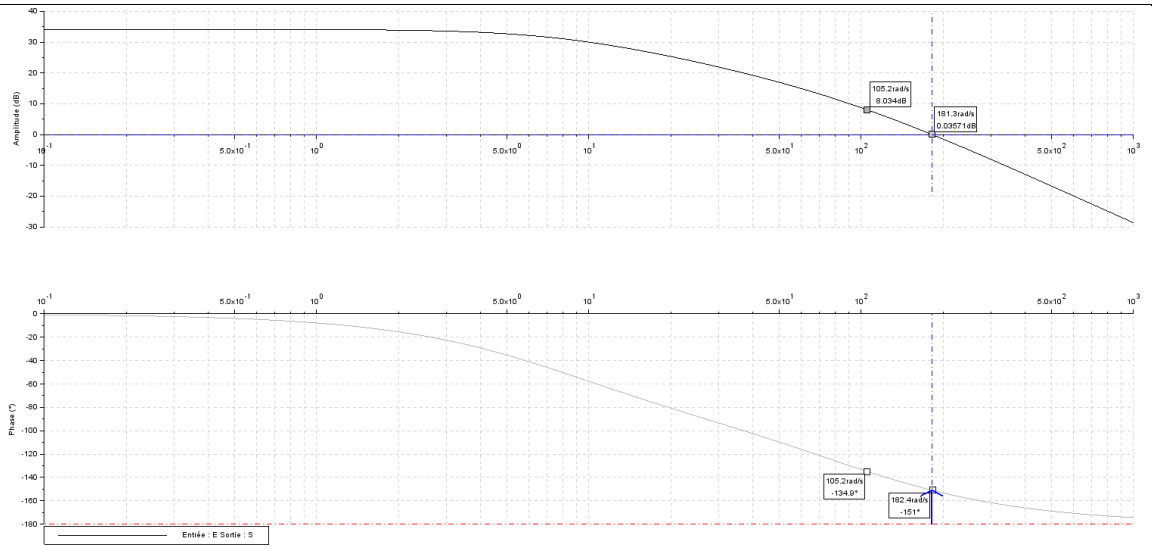





```
>>> Lw[-1]
1.0018187313076155
```

Correction AP

$$\frac{1+aTp}{1+Tp}$$




```
Marge de gain infinie ; Marge de phase : 29.07°
```

Remontons la phase d'environ $\theta = 20°$ en $\omega_{c_0} = 182{,}3\ rd.s^{-1}$

$$a = \frac{1+\sin\theta}{1-\sin\theta} = 2{,}04 \quad ; \quad T = \frac{1}{\omega_{c_0}\sqrt{a}} = \frac{1}{182{,}3\ \sqrt{2{,}04}} = 0{,}0038$$