Projet AU2 : Alimentation sans Interruption

Table of Contents

Première Partie : Modèle de commande et cahier des charges

Defintion des grandeurs electriques

```
L = 714e-6 ; % en Henry
R = 10 ; % en Ohm
C = 121.2e-6 ; % en Farade
Vd = 250 ; % en Volts
```

I. Définition de nos fonctions de transfert

1.1) on a

$$\begin{cases} \dot{x} = A_m \cdot x + B_m \cdot u \\ y = C_m \cdot x \end{cases} \underset{\text{en Laplace}}{\Rightarrow} \begin{cases} p \cdot x(p) = A_m \cdot x + B_m \cdot u(p) \\ y(p) = C_m \cdot x(p) \end{cases}$$

$$\Rightarrow \left(p \cdot I - A_m\right) x(p) = B_m \cdot u(p)$$

$$\Rightarrow \begin{bmatrix} p + \frac{1}{RC} & -\frac{1}{C} \\ \frac{1}{L} & p \end{bmatrix} \begin{bmatrix} V_o(p) \\ I_c(p) \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} \cdot u(p)$$

$$\Rightarrow \begin{bmatrix} pV_o(p) + \frac{V_o(p)}{RC} - \frac{I_c(p)}{C} \\ \frac{V_o(p)}{L} + p \cdot I_c(p) \end{bmatrix} = \begin{bmatrix} 0 \\ u(p)/L \end{bmatrix}$$

$$\Rightarrow \begin{cases} C p V_o(p) + \frac{V_o(p)}{R} = I_c(p) \\ \frac{V_o(p)}{L} + p \cdot I_c(p) = \frac{u(p)}{L} \end{cases}$$

$$\Rightarrow \begin{cases} V_o(p)\left(\frac{1}{L} + p \cdot \left(Cp + \frac{1}{R}\right)\right) = \frac{u(p)}{L} \\ V_o(p) = I_c(p) \times \frac{1}{C \cdot p + 1/R} \end{cases}$$

$$\Rightarrow \begin{cases} \frac{V_o(p)}{u(p)} = \frac{1}{L} \cdot \frac{1}{1/L + Cp^2 + p/R} \\ \frac{I_c(p) \cdot \frac{1}{C \cdot p + 1/R}}{u(p)} = \frac{1}{L} \cdot \frac{1}{1/L + Cp^2 + p/R} \end{cases}$$

$$\Rightarrow \boxed{\begin{aligned} F_1(p) &= \frac{V_o(p)}{u(p)} = \frac{R}{RLCp^2 + Lp + R} \\ F_2(p) &= \frac{I_c(p)}{u(p)} = \frac{1 + RCp}{RLCp^2 + Lp + R} \end{aligned}}$$

I.1 F1(p)

```
G1 = 1 ;
wo1 = 1/sqrt(L*C);
xio1 = (1/(2*R))*sqrt(L/C);
```

```
p = tf('s');
F1p = G1/(1 + (2*xio1/wo1) * p + (1/wo1^2)*p^2)
```

```
F1p =

                1
  ------------------------------
  8.654e-08 s^2 + 7.14e-05 s + 1

Continuous-time transfer function.
Model Properties
```

I.2 F2(p)

```
G2 = 1/R ;
wo2 = 1/sqrt(L*C);
xio2 = (1/2*R)*sqrt(L/C);
Tau2 = R*C;
p = tf('s');
F2p = (G2*(1+Tau2*p))/(1 + (2*xio1/wo1) * p + (1/wo1^2)*p^2)
```

```
F2p =

        0.0001212 s + 0.1
  ------------------------------
  8.654e-08 s^2 + 7.14e-05 s + 1

Continuous-time transfer function.
Model Properties
```

I.3 F3(p)

```
G3 = R ;
Tau3 = R*C ;
p = tf('s');
F3p = G3/(1+Tau3*p)
```

```
F3p =

        10
  --------------
  0.001212 s + 1

Continuous-time transfer function.
Model Properties
```

I.4 Gain statique, coefficient d'amortissement et pulsation propre de F1 et F2

II. Analyse des systèmes F2(p) et F3(p)

II.1 Analyse de F2(p)

II.1.1 Analyse temporelle : réponse indicielle

```
step_info = stepinfo(F2p);
depassement_F2p = step_info.Peak
```

```
depassement_F2p = 0.4319
```

```
tempsdereponse5p_F2p= step_info.SettlingTime
```

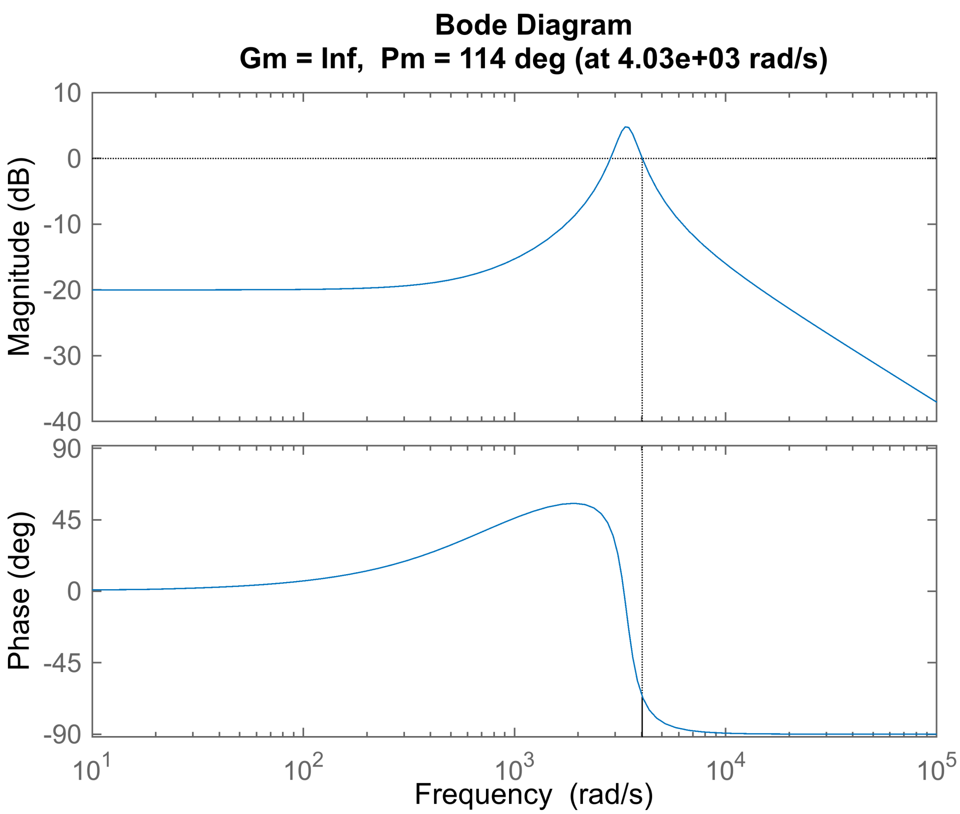
```
tempsdereponse5p_F2p = 0.0128
```

```
erreur_static = R/(1+R)
```

```
erreur_static = 0.9091
```

II.1.2 Analyse fréquencielle

```
margin(F2p) % marge de gain et marge de phase à partir du diagramme de Baud
```

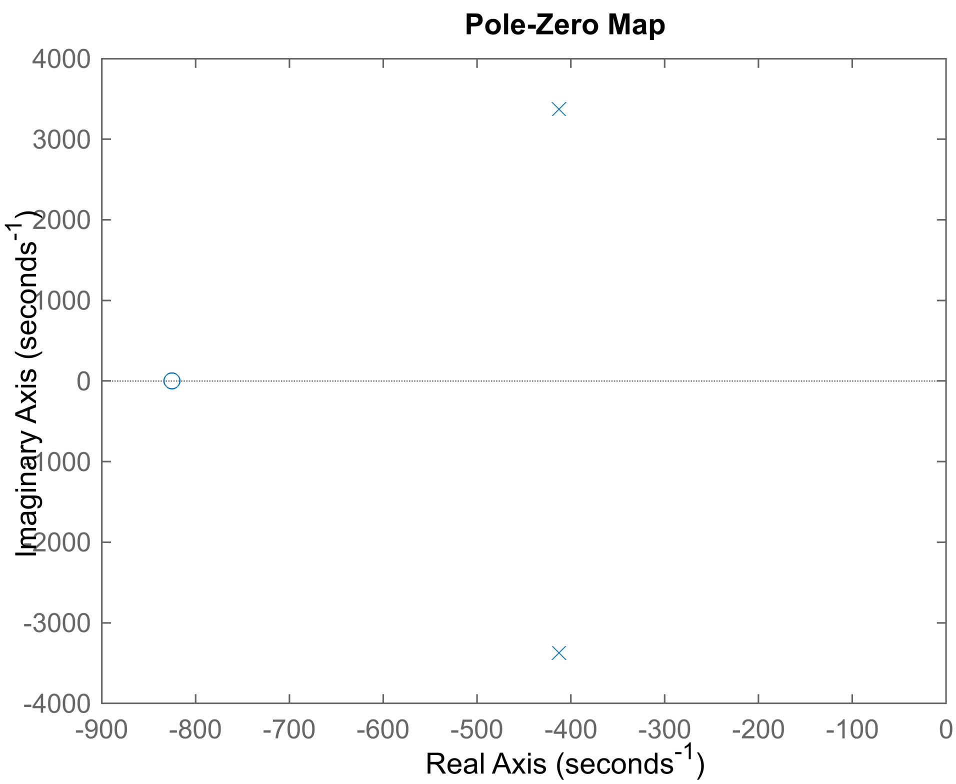


l'analyse fréquentielle montre qu'il y a résonnance à la fréquence de 3632Hz

la marge de gain est infinie et la marge de phase est de 114°

II.1.3 Analyse des pôles et des zéros

```
pzmap(F2p)
```



les pôles sont à partie réelle négatives => le processus est donc stable

le processus possède un zéro à p = -825,

II.2 Analyse de F3(p)

II.2.1 Analyse temporelle : réponse indicielle

```
step_info3 = stepinfo(F3p);
depassement_F3p = step_info3.Peak
```

```
depassement_F3p = 9.9934
```

```
tempsdereponse5p_F3p= step_info3.SettlingTime
```

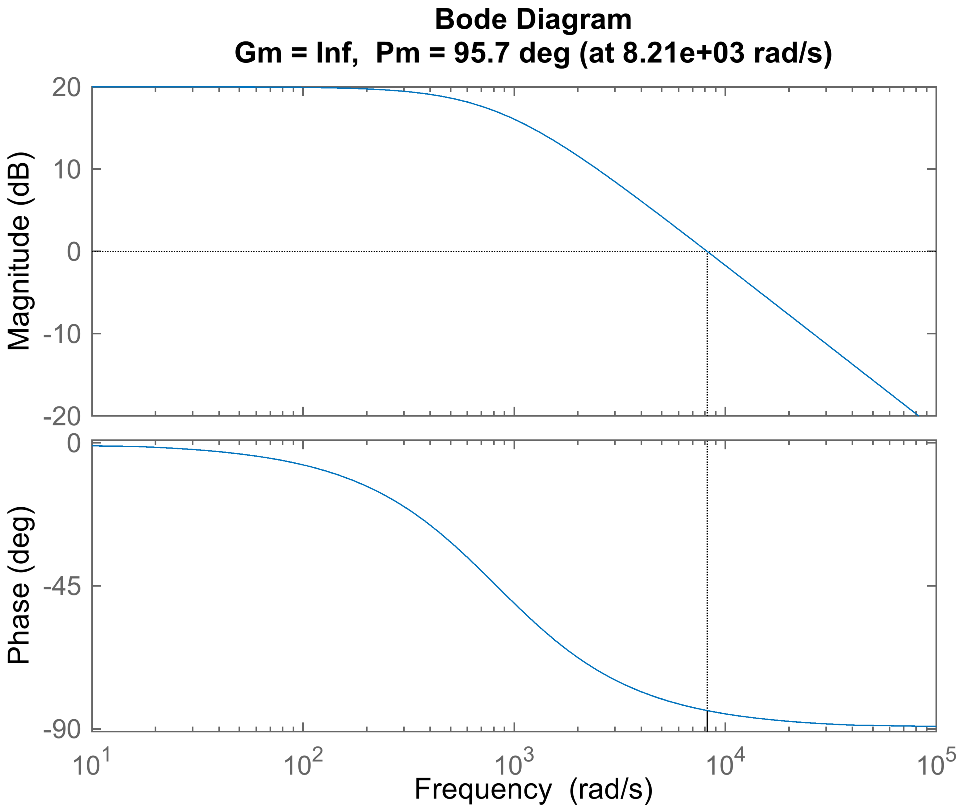
```
tempsdereponse5p_F3p = 0.0047
```

```
erreur_static3 = 1/(1+R)
```

```
erreur_static3 = 0.0909
```

II.1.2 Analyse fréquencielle

```
margin(F3p) % margin de gain et marge de phase
```

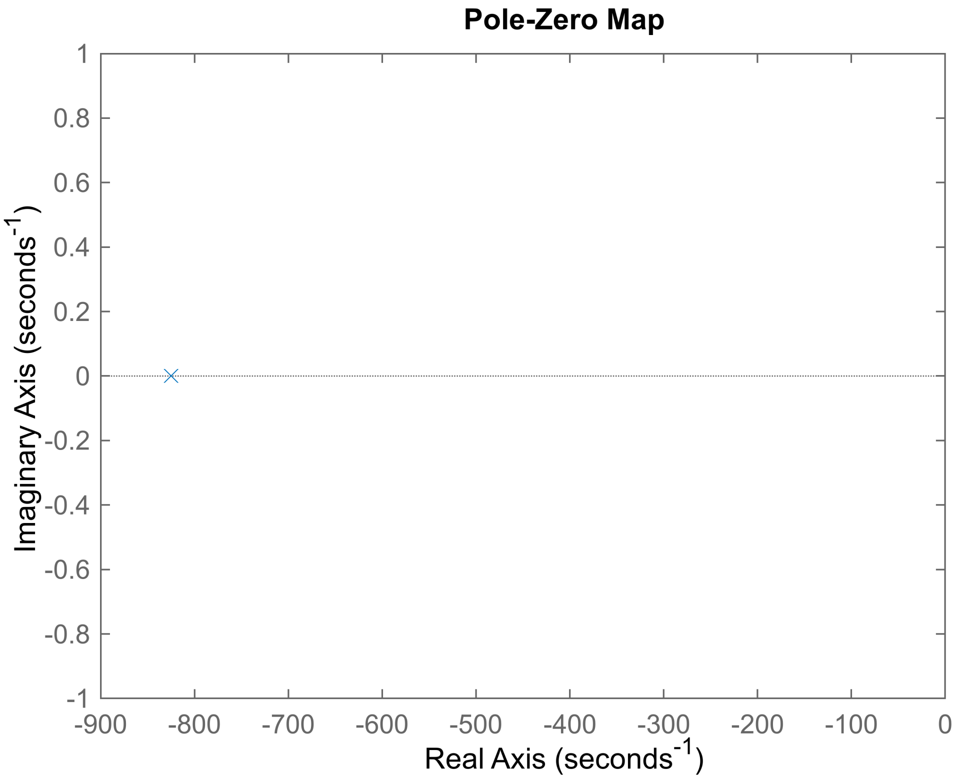


l'analyse fréquentielle montre que la fréquence de coupure à -3db est à 11.5kHz

la marge de gain est infinie et la marge de phase est de 95.7°

II.1.3 Analyse des pôles et des zéros

```
pzmap(F3p)
```



le pôle est à partie réelle négative => le processus est donc stable

le processus ne possède pas de zéro

III. Analyse des systèmes F2(z) et F3(z)

```
% dicrétisation du process avec bloqueur d'ordre 0
Te = 50e-6;
F2z = c2d(F2p,Te);
F2z.variable = 'z^-1'
```

```
F2z =

  0.06969 z^-1 - 0.06687 z^-2
  ----------------------------
  1 - 1.931 z^-1 + 0.9596 z^-2

Sample time: 5e-05 seconds
Discrete-time transfer function.
Model Properties
```

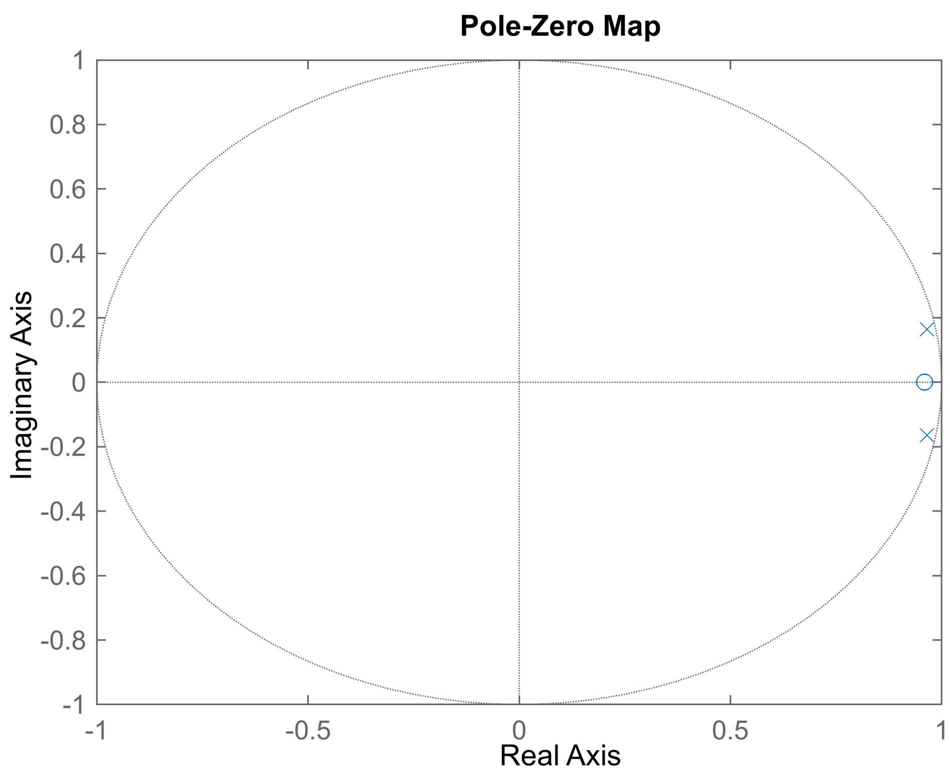
```
F3z = c2d(F3p, Te);
F3z.variable = 'z^-1'
```

```
F3z =

    0.4041 z^-1
  ---------------
```

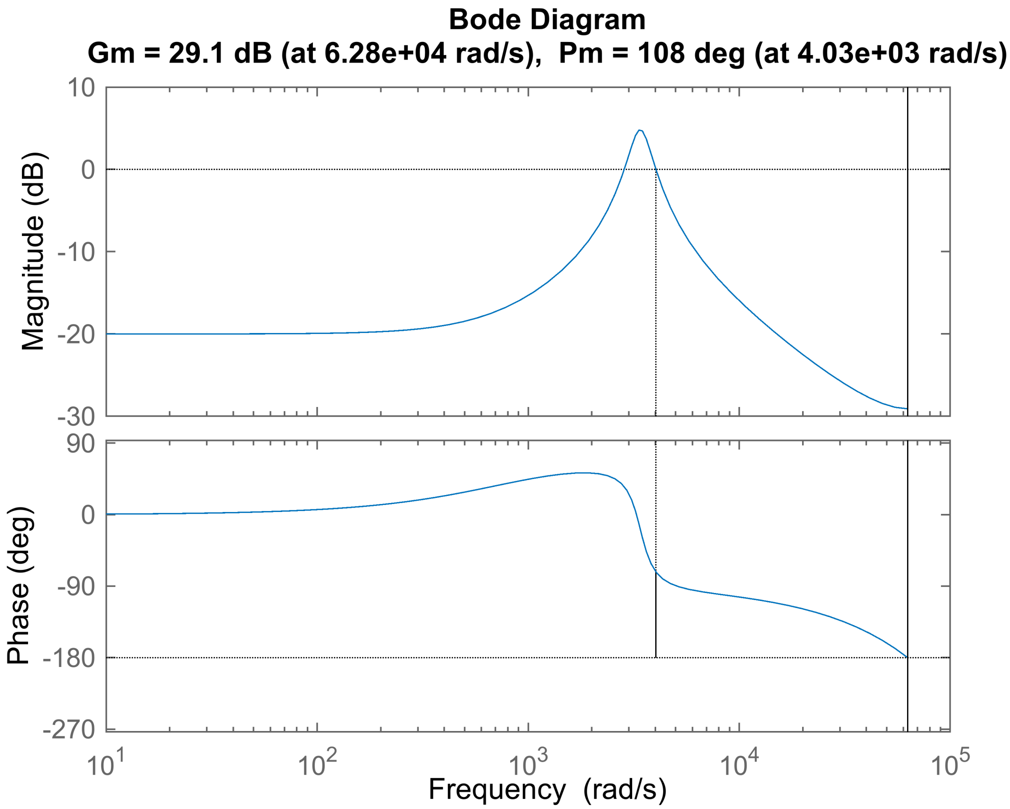
```
  1 - 0.9596 z^-1

Sample time: 5e-05 seconds
Discrete-time transfer function.
Model Properties
```

III.1 Analyse des marges de module et de retard, des zéros et des pôles F2(z)

```
pzmap(F2z)
```




```
margin(F2z) % nous donne marge de gain marge de phase
```



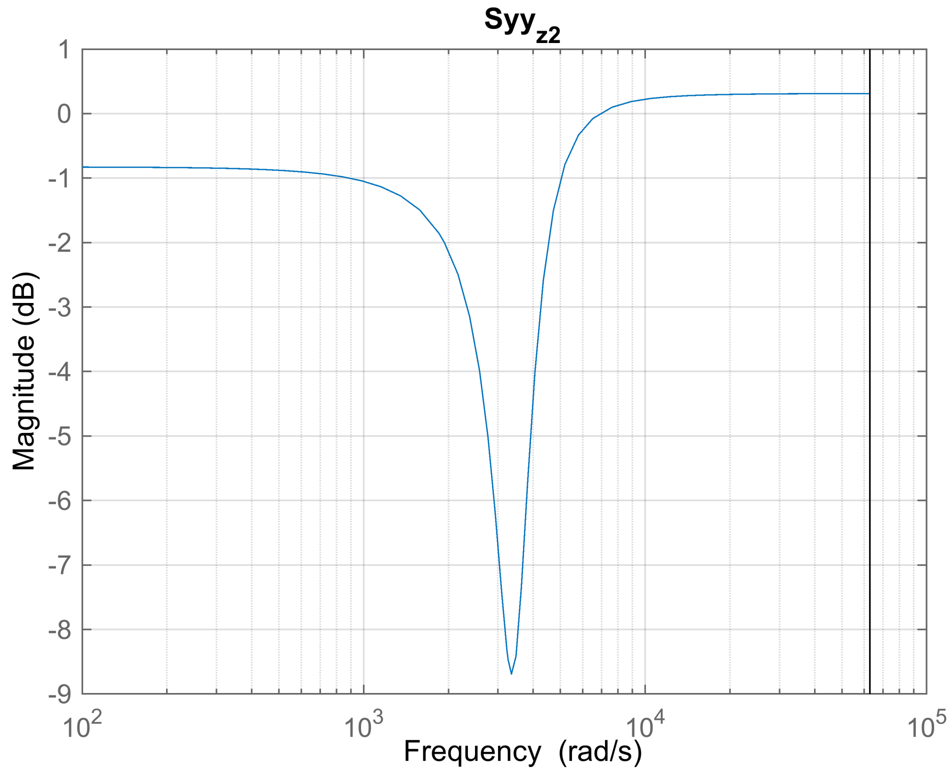

```
margin_delay2 = (108*pi)/(180*4.03e+3) % in secondes
```

```
margin_delay2 = 4.6773e-04
```

```
Syy_z_2 = 1/(1+ F2z) % fonction de sensibilité
```

```
Syy_z_2 =

  1 - 1.931 z^-1 + 0.9596 z^-2
  ----------------------------
  1 - 1.862 z^-1 + 0.8927 z^-2

Sample time: 5e-05 seconds
Discrete-time transfer function.
Model Properties
```

```
bodemag(Syy_z_2), grid,title('Syy_z_2')
```



les pôles sont à l'intérieur du cercle unité => le processus F2(z) est stable, ils sont proches du cercle unité

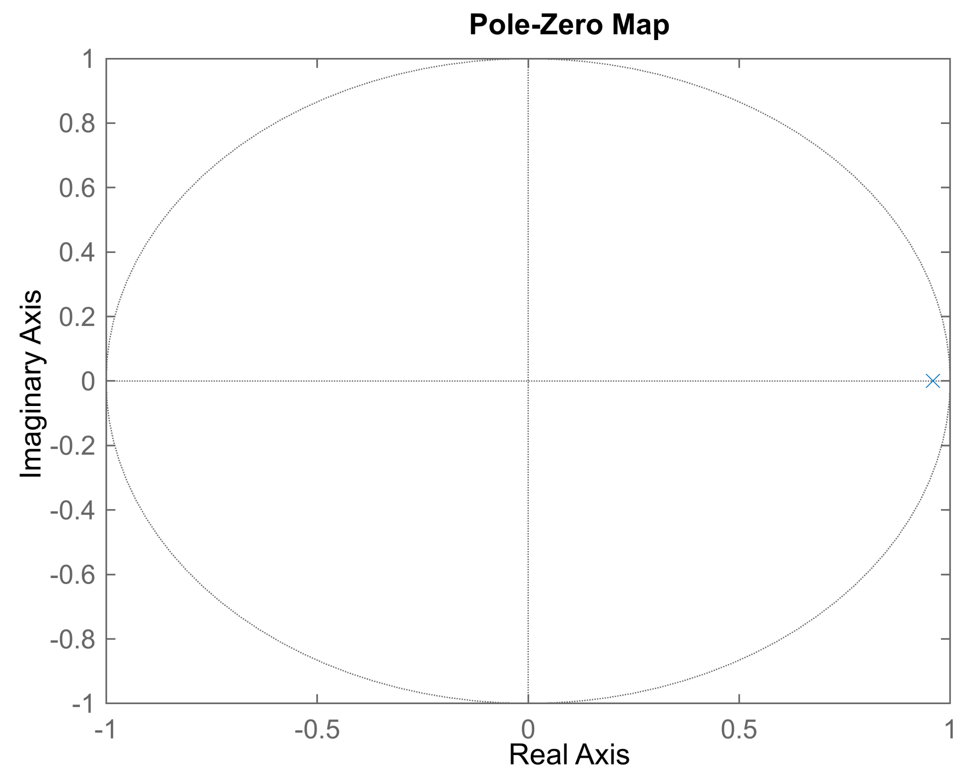
Les zéros sont stables non oscillatoires.

on détermine la marge de module à partir de la fonction de sensibilité Syy_z_2 , on 1/**Δ**M = max(Syy_z_2) = 2 => **Δ**M = 0.5 >=0.5 le processus est donc robuste vis à vis d'une erreur sur le gain statique du processus

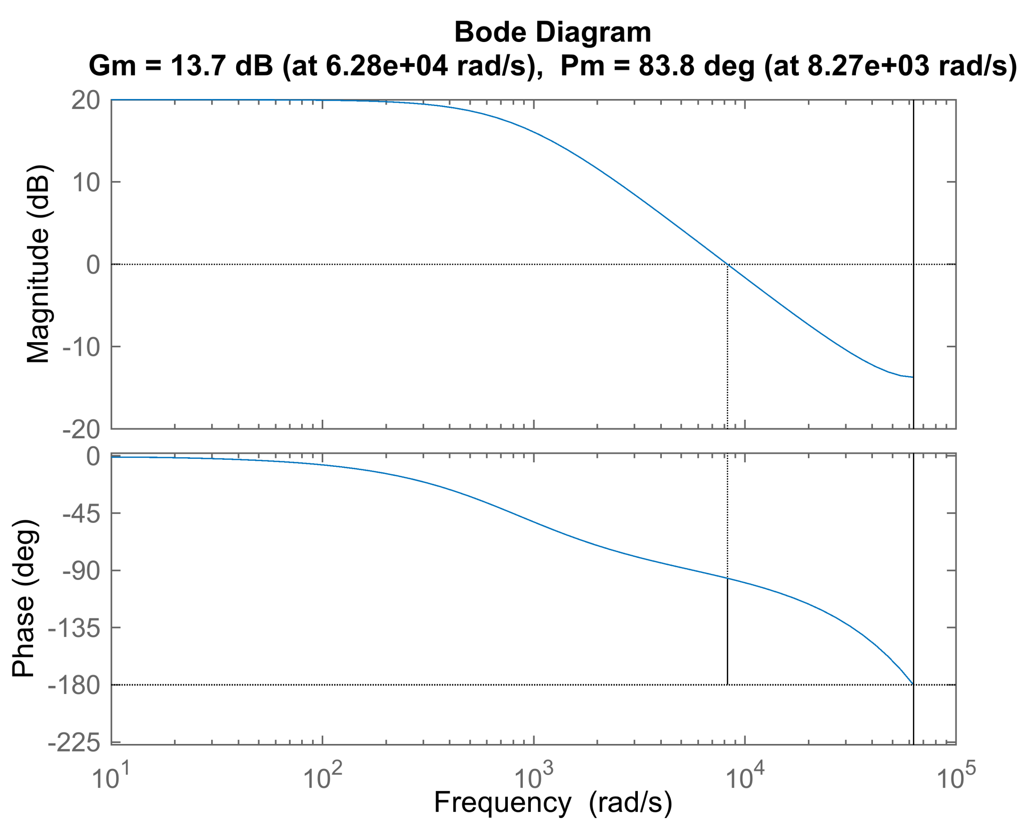
la marge de retard est de 47ms >> à 2*Te et Tp=2*pi/w =0.76ms et marge de retard >>Tp/2 => le processus discrétisé est donc robuste vis à vis d'un retard pur dans le systèmeà partir du diagramme de Baud

III.2 Analyse des marges de module et de retard, des zéros et des pôles F3(z)

```
pzmap(F3z)
```




```
margin(F3z) % nous donne marge de gain marge de phase
```



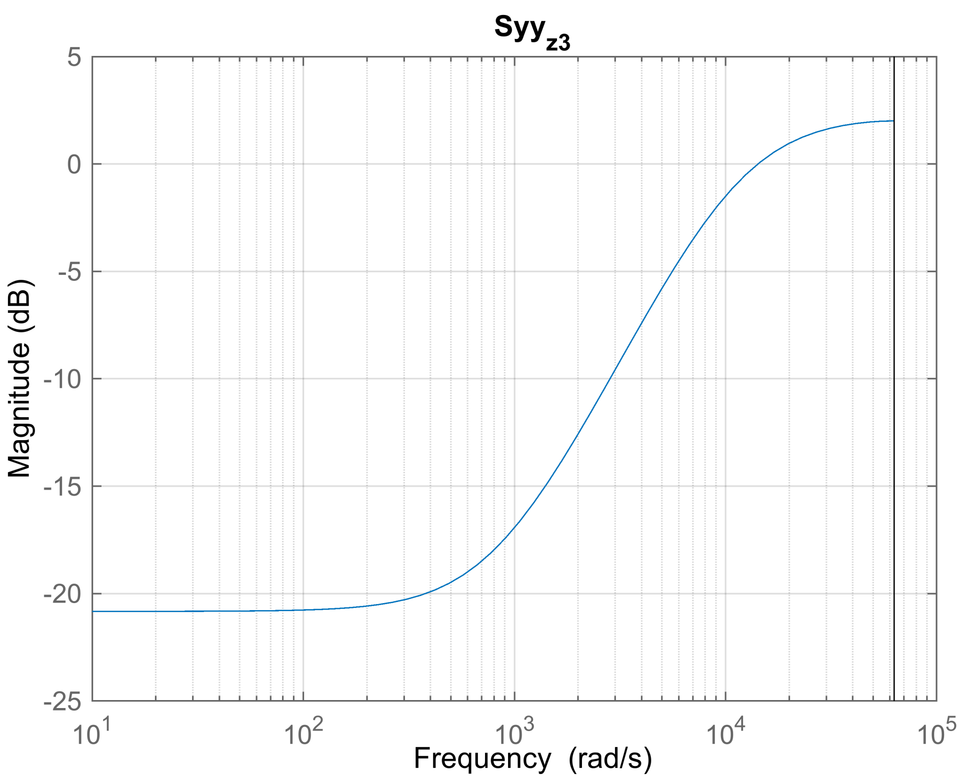
```
margin_delay3 = (83.8*pi)/(180*8.27e+3) % in secondes
```

```
margin_delay3 = 1.7685e-04
```

```
Syy_z_3 = 1/(1+ F3z) % fonction de sensibilité
```

```
Syy_z_3 =

  1 - 0.9596 z^-1
  ---------------
  1 - 0.5554 z^-1

Sample time: 5e-05 seconds
Discrete-time transfer function.
Model Properties
```

```
bodemag(Syy_z_3), grid,title('Syy_z_3')
```



le pôle est à l'intérieur du cercle unité => le processus F3(z) est stable, il est proche du cercle unité

Le processus ne possède pas de zéro.

On détermine la marge de module à partir de la fonction de sensibilité Syy_z_3 , on 1/**Δ**M = max(Syy_z_3) = 0.3 => **Δ**M = 3.33 > 0.5 le processus est donc robuste vis à vis d'une erreur sur le gain statique du processus

la marge de retard est de 177ms >> à 2*Te et tau =1.2ms et marge de retard >>Tp/2 => le processus discrétisé est donc robuste vis à vis d'un retard pur dans le systèmeà partir du diagramme de Baud

Deuxième Partie : Commande de Courant I

```
cdc_z_I =  filt([0 1],1,Te) ;% LE Cahier des charge est une réponse pile dont la
fct de transfert est z^-1
z_1 = filt([0 1],1,Te);
[Bz_I,Az_I] = tfdata(F2z,'v');
zpk(F2z) % permet de déterminer Biz et BsZ
```

```
ans =

  0.069695 z^-1 (1-0.9595z^-1)
  ----------------------------
  (1 - 1.931z^-1 + 0.9596z^-2)

Sample time: 5e-05 seconds
Discrete-time zero/pole/gain model.
Model Properties
```

I. Synthèse du correcteur avec placement des pôles et des zéros

I.1 Identification des zéros du processus

Le processus possède un zéro non oscillatoire, qu'on va considérer comme stable

```
zerooo = zero(F2z);
Bsz_I = [1 zerooo];
[Biz_I, reste]=deconv(Bz_I,Bsz_I);

Bi1_I =polyval(Biz_I,1);
BizsurB1z = filt ( Biz_I, Bi1_I , Te);
cdc_a_I = BizsurB1z/z_1*cdc_z_I ;
minreal(cdc_a_I) % on s'implifie le cahier de charges adapté
```

```
ans =

  z^-1

Sample time: 5e-05 seconds
Discrete-time transfer function.
Model Properties
```

On retrouve que cdc_a = cdc , ceci est prévisible vu qu'on n'a pas de zéros instables

I.2Calcul du correcteur avec le cahier des charges adapté

```
Kz_I  =1/F2z*cdc_a_I/(1-cdc_a_I);
```

Simplification des correcteurs

```
zpk(Kz_I);
minreal(Kz_I,1.e-4);
```

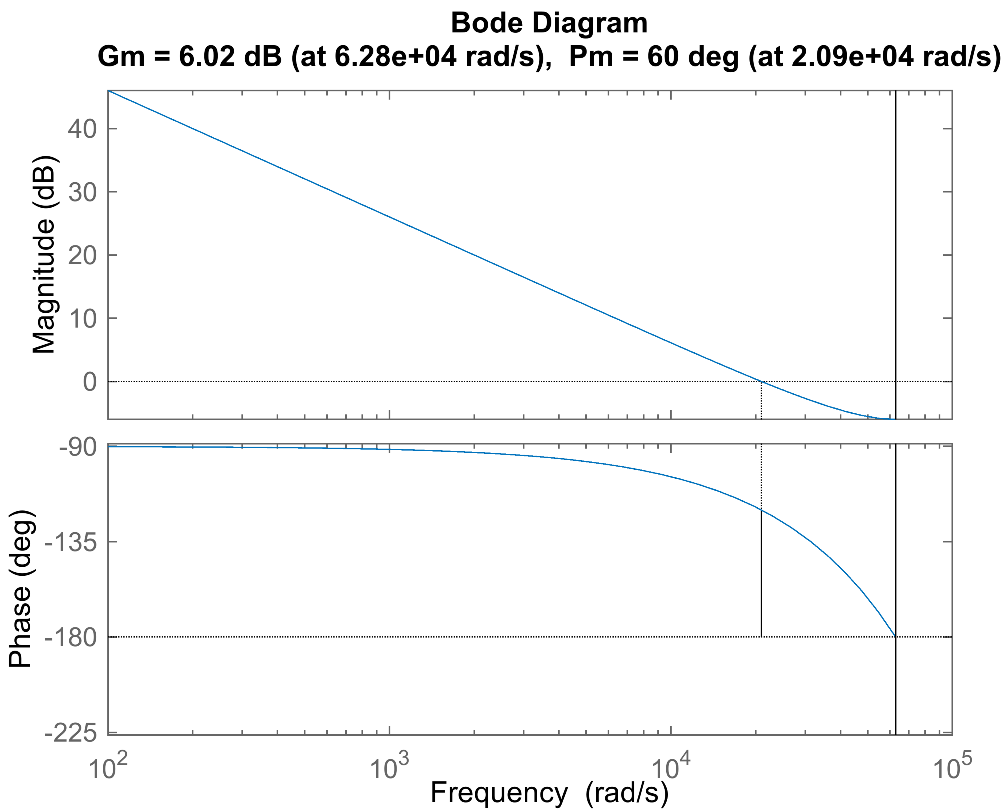
I.3 Vérification du correcteur synthétisé

```
%Vérifier le correcteur trouvé
%FTBO et FTBF
```

```
FTBO_I=F2z*Kz_I;
figure
FTBF_I=(Kz_I*F2z)/(1+Kz_I*F2z);
```

I.4Analyse de la robustesse

```
L_I=Kz_I*F2z;
margin(L_I);
```

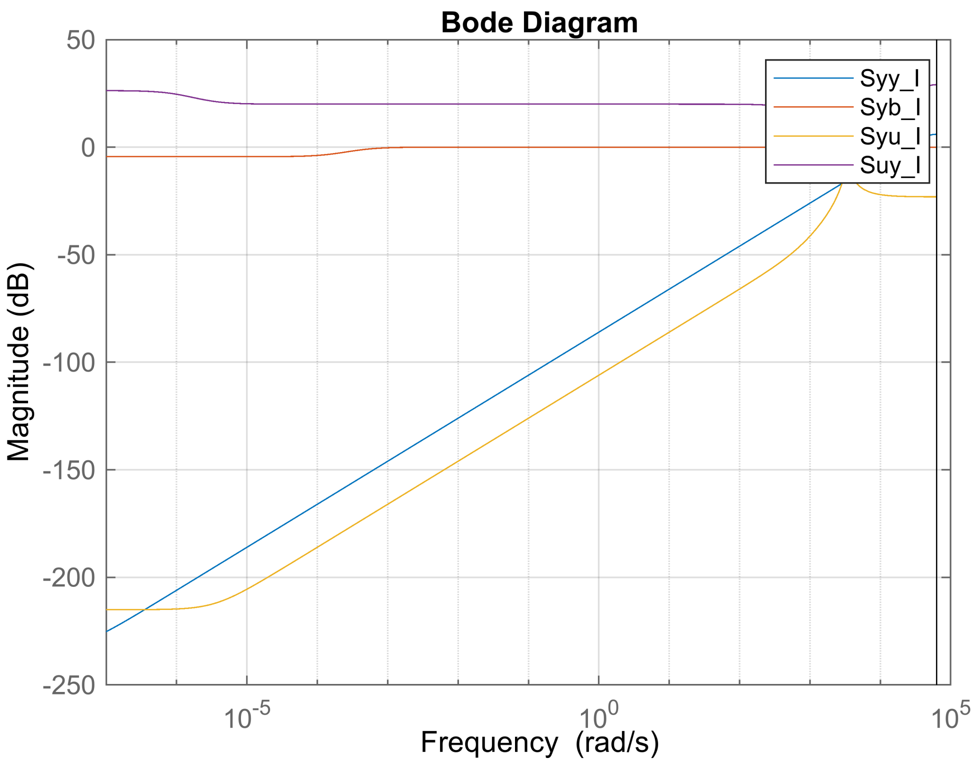



```
margin_delay_FTBO_I = (60*pi)/(180*2.09e+4) % in secondes
```

```
margin_delay_FTBO_I = 5.0105e-05
```

La marge de retard est 50ms >> 2*Te et Tp = 2*pi/w = 0.3ms << 2*marge de module

```
S_I=1/(1+L_I);
Syy_I=S_I;
Syu_I=F2z/(1+L_I);
Syb_I=-L_I/(1+L_I);
Suy_I=Kz_I/(1+L_I);
bodemag(Syy_I,Syb_I,Syu_I,Suy_I),grid,legend
```



La marge de module est **Δ**M = 0.17 < 0.5 , le correcteur n'est pas robuste vis à vis d'une erreur sur le gain statique du processus.

On remarque que la fonction de sensibilité Syb_I qui est la fonction de sensibilité de la sortie par rapport aux bruits de mesure ne rejette aucune de ces perturbations.

Le gain est de 0dB => On recopie tout bruit de mesure sur la sortie

On rejette les bruits de commande avec une réponse pile, mais pas comme indiqué dans le cahier des charges avec un comportement du premier ordre avec un temps caractéristique de 0.4ms

- ***Par contre les bruits de mesure ne sont pas rejettés et sont directement copiés sur la sortie***

On ne respecte donc pas le cahier des charges, on ne choisira pas ce correcteur.

II Synthèse de correcteur par modèle interne

```
Jz_I = filt(Biz_I,Bi1_I,Te); % discretre transfer fucntion filt(numerator, 
denominator, TE)
```

Calculer C(z)

```
Cz_I= Jz_I/F2z;
zpk(Cz_I); % Forme factorisée, important  à faire pour supprimer le zéro instble
minreal(Cz_I,1.e-4)
```

```
ans =

  14.35 - 27.71 z^-1 + 13.77 z^-2
  -------------------------------
          1 - 0.9595 z^-1

Sample time: 5e-05 seconds
Discrete-time transfer function.
Model Properties
```

Calcul de Gz pour la dynamique en asserv

```
Gz_I = cdc_z_I/z_1
```

```
Gz_I =

  1

Static gain.
Model Properties
```

Calcul de Fz pour la dynamique en régulation

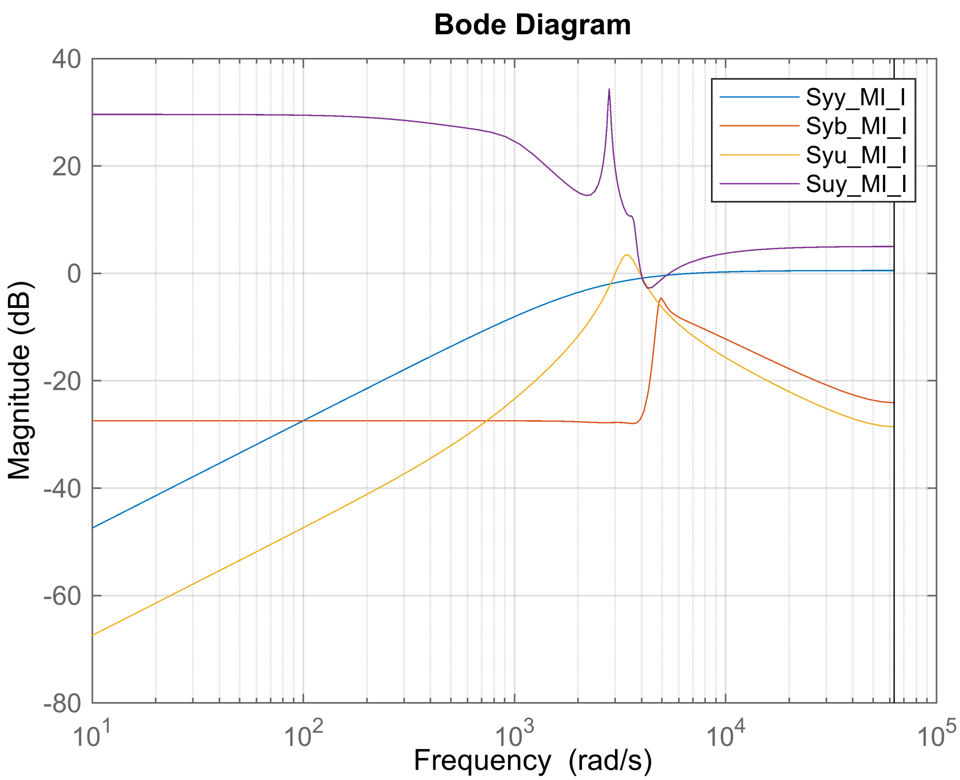
```
tau_y = 4e-4 ;
Cdc_reg_I = tf(1,[tau_y,1]);
Cdc_regz_I =c2d(Cdc_reg_I,Te);
Cdc_regz_I.variable = 'z^-1';
Fz_I = Cdc_regz_I/z_1
```

```
Fz_I =

      0.1175
  ---------------
  1 - 0.8825 z^-1

Sample time: 5e-05 seconds
Discrete-time transfer function.
Model Properties
```

II.1 Analyse de la robustesse de la commande par modèle interne

Rappelons que le cahier des charges nous impose une régulation avec une dynamique 1/(1+04.e-3p) et un rejet des bruits de mesures à au moins -10db pour les HF > 1.5Khz<=> 9420 rad.s^-1 .

```
K2z_I = Cz_I*Fz_I/(1-Cz_I*Fz_I*F2z);
Syy_MI_I = 1/(1+K2z_I*F2z);
Syu_MI_I = F2z  /(1+K2z_I*F2z);
Syb_MI_I = -F2z*K2z_I/(1+K2z_I*F2z);
Suy_MI_I = -K2z_I/(1+K2z_I*F2z);
bodemag(Syy_MI_I,Syb_MI_I,Syu_MI_I,Suy_MI_I),grid,legend
```



L'analyse de la robustesse nous renseigne sur plusieurs paramètre :

- la marge de module : **Δ**M = 1>0.5
- Le rejet des perturbations de mesure indiqué par la fonction Syb_MI_I nous renseigne qu'on rejette effectivement les bruits de mesure supérieur à 1500Hz (9420 rad.s^-1).

Conclusion : Nous respectons le cahier des charges

La simulation valide bien le cdc :

Voir fichier simulink Intern_sami_fahim :

- Pour l'asservissement choisir une entrée sinusoidale
- On peut rajouter des perturbations à différents niveau
- Pour observer la validation du cdc en régulation mettre une entrée nulle et rajouter une perturbation sur la sortie

Troisième partie : Commande RST pour la boucle de tension

III.1 Implémentation de la commande RST

Pour ce qui est du cahier de charge en asservissement pour la tension on ne pourra choisir que z^-2 comme le plus rapide qu'on puisse avoir car le processus contient une période de retard.

Si on avait pris un cdc en réponse pile, Gz_U ne serait pas réalisable car le degré de son numérateur est strictement supérieur à celui de son dénominateur.

```
z_d = z_1; % le processus contient une période de retard dû à la boucle interne de
courant
cdc_z_U = filt([0 0 1],1,Te);
Gz_U = cdc_z_U/(z_1*z_d)
```

```
Gz_U =

  1

Static gain.
Model Properties
```

Calculer A1 pour savoir si on doit précaractériser Sz

```
[Bz_U,Az_U] = tfdata(F3z,'v');
A1_U=polyval(Az_U,1) ;  %=0,0404
Bsz_U = 1;
[Biz_U, reste]=deconv(Bz_U,Bsz_U);
%A1<> de 0, il faut précaractériser Sp_U(z)
%Car il n'y a pas d'intégrateur dans le processus
```

précaractérisation de Sp_U=1-z^(-1) pour éliminer l'erreur statique

```
Sp_U=[1 -1];
Rp=[1 0]; % rejet des bruits de mesure avec au moins -10dB pour les hautes
fréquences >1.5khz
% On ne précaractérise pas dans un premier temps Rp
```

Déterminer les dégrés de polynôme Pz_U, Dz_U et Fz_U

```
np=3;     %np=Nc+Ne+d-1=2+1+1-1
nd=1;     %Nd=Ne+d-1=1+1-1
nf=1;     %Nf=Nc-1=2-1
```

Détermination de P(z)_U

Déterminer les pôles principaux à partir de CdCreg

```
%Traduire le CdC en régulation pour calculer Pp(z)
% *Comportement en régulation du premier ordre avec une constante de temps de
% 2.1 ms.*
tauy_U=2.1e-3;  %constante de temps de 1ms
CdC_reg_U=tf(1,[tauy_U,1]) ;  %cdC de régulation en continu
CdC_regz_U=c2d(CdC_reg_U,Te);
CdC_regz_U.variable='z^-1';
[num_reg_U,den_reg_U]=tfdata(CdC_regz_U,'v');
Pp_U=den_reg_U;
```

Déterminer Pz(z)

Az est stable

```
Pz_U=Az_U;
```

Pas de renforcement de robustesse

```
Pa_U=1;

% Au final P(z)=Pp(z)*Pz(z)*Pa(z) avec Pa(z)=1
P_z_U=conv(Pp_U,Pz_U);   % Produit de deux polynomes

P_z_U=[P_z_U, 0 ]  %compléter les 0 qui manquent pour atteindre np
```

```
P_z_U = 1×4
    1.0000   -1.9361    0.9370         0
```

Calculer Cz et Ez

```
Cz_U=conv(Az_U,Sp_U);   %C(z)=A(z)*Sp(z)
c1=Cz_U(2);
c2=Cz_U(3);
```

```
Ez_U=conv(Biz_U,Rp);
e1=Ez_U(2);   %b1
```

Construre la matrice M

La construction de la matrice M se fait à partir de la résolution de l'équation de Bezout.

Le calcul est détaillé ci dessous :

$$P(z) = C(z) \cdot D(z) + E(z) \cdot F(z)$$

on a

$$C(z) = 1 + c_1 z^{-1} + c_2 z^{-2}$$

$$E(z) = e_1 z^{-1}$$

$$P(z) = 1 + p_1 z^{-1} + p_2 z^{-2} + 0 z^{-3}$$

$$D(z) = S_1(z) = 1 + d_1 z^{-1} + d_2 z^{-2}$$

$$F(z) = R(z) = r_0$$

$$C(z) \cdot D(z) = 1 + z^{-1}(c_1 + d_1) + z^{-2}(c_2 + c_1 d_1 + d_2) + z^{-3}(c_2 d_1 + d_2 c_1) + d_2 c_2 z^{-4}$$

$$E(z) \cdot F(z) = r_0 e_1 z^{-1}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ c_1 & 1 & 0 & e_1 \\ c_2 & c_1 & 1 & 0 \\ 0 & c_2 & c_1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ d_1 \\ d_2 \\ r_0 \end{bmatrix} = \begin{bmatrix} 1 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

$$M \quad \underline{X} \quad = \quad \underline{P}$$

```
M=[1 0 0 0 ; c1 1 0 0  ; c2 c1 e1 0; 0 c2 0 e1 ]
```

```
M = 4×4
    1.0000         0         0         0
   -1.9596    1.0000         0         0
    0.9596   -1.9596    0.4041         0
         0    0.9596         0    0.4041
```

```
X=inv(M)*(P_z_U)' ;    %P'=P transposé
```

récuérer les coefficients de D(z) et F(z)

```
d0=X(1);
d1=X(2);
f0=X(3);
f1=X(4);
```

Construire R(z) et S(z) T(z)

```
Rz=[f0 f1];  %Rp(Z)=1 Rz=Rp(z)*F(z)=F(z)
Sprim_z_U=conv(Sp_U, [d0 d1]);  %S'(z)=Sp(z)*S1(z)=Sp(z)*D(z)
Sz=conv(Sprim_z_U,Bsz_U);   % changement de variable invers
```

T(z)=P(z)/Bi(1)

```
Bi1_U=polyval(Biz_U,1);
Tz_U=P_z_U/Bi1_U;

Sz_U=filt(1,Sz,Te)
```

```
Sz_U =

                1
  ------------------------------
  1 - 0.9765 z^-1 - 0.02353 z^-2

Sample time: 5e-05 seconds
Discrete-time transfer function.
Model Properties
```

```
Rz_U=filt(Rz,1,Te)
```

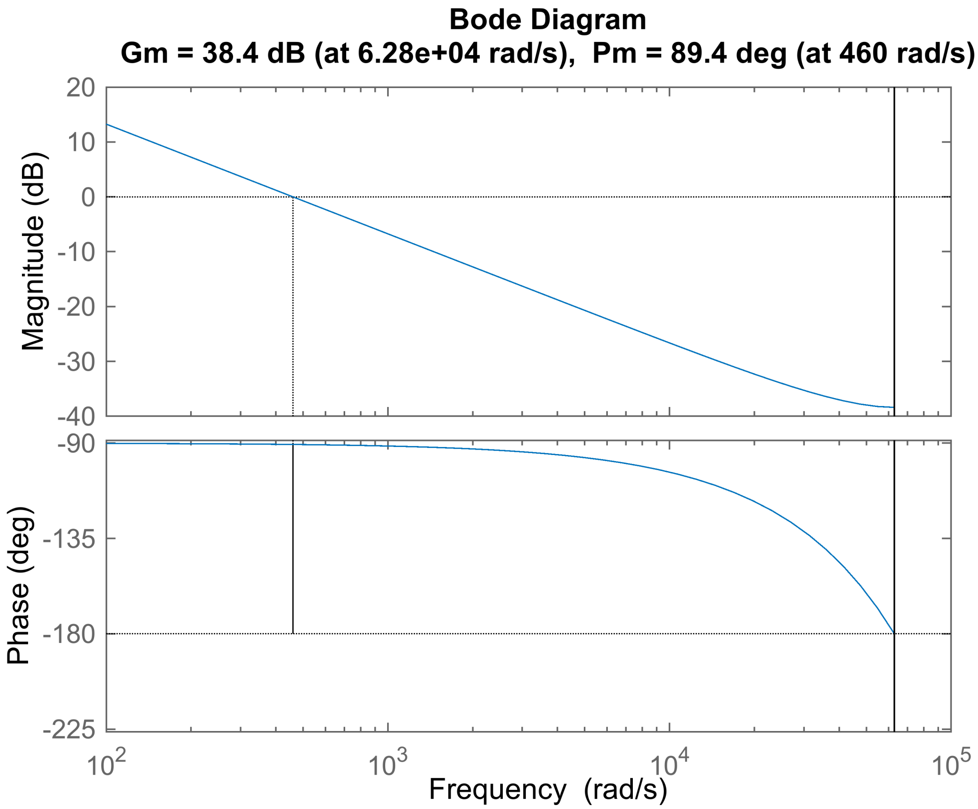
```
Rz_U =

  0.05822 - 0.05586 z^-1

Sample time: 5e-05 seconds
Discrete-time transfer function.
Model Properties
```

```
Tz_U=filt(Tz_U,1,Te)
```

```
Tz_U =

  2.474 - 4.79 z^-1 + 2.318 z^-2

Sample time: 5e-05 seconds
Discrete-time transfer function.
Model Properties
```

III.2 Analyse de la robustesse

```
K_RST=filt(Rz,Sz,Te);
Lyy_RST=K_RST*F3z;

margin(Lyy_RST)
```



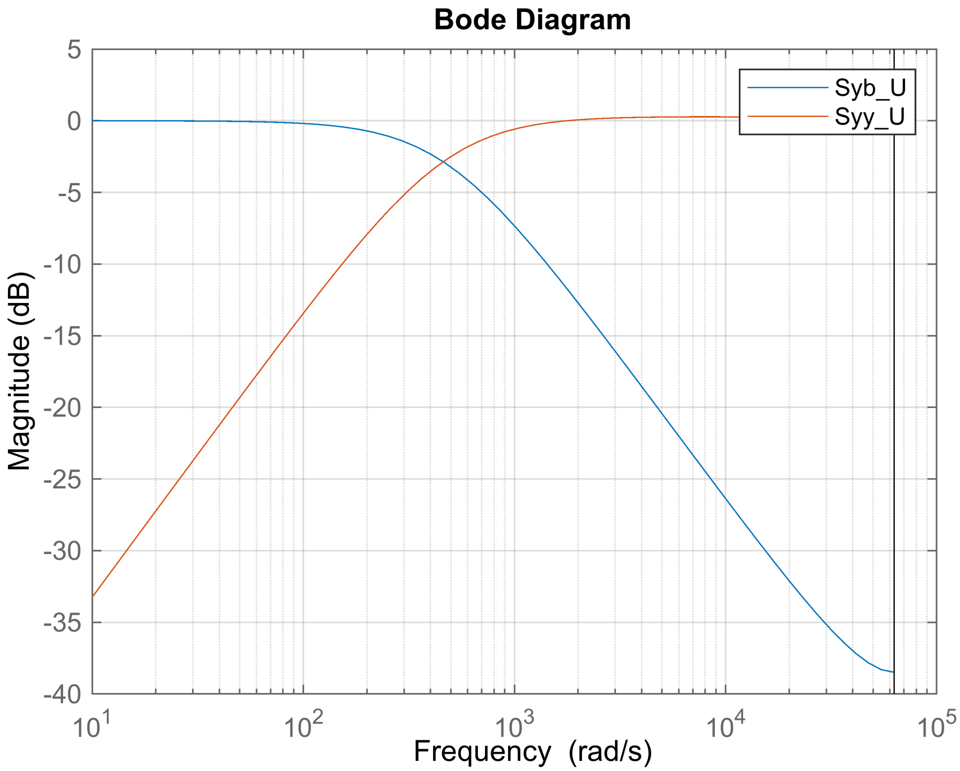

```
allmargin(Lyy_RST)
```

```
ans = struct with fields:
     GainMargin: 83.0040
    GMFrequency: 6.2832e+04
    PhaseMargin: 89.3715
    PMFrequency: 459.8832
    DelayMargin: 67.8358
    DMFrequency: 459.8832
         Stable: 1
```

La marge de phase est de 67.8ms >> 2*Te, ainsi la commande RST est robuste aux retards

Analyse des fonction de sensibilité

```
% creating the transfer functions
Bz_U_tf = filt(Bz_U,1,Te);
Az_U_tf = filt(Az_U,1,Te);
Sz_U_tf = 1/Sz_U;
Syb_U = -(Rz_U*Bz_U_tf*z_d)/(Az_U_tf*Sz_U_tf+Bz_U_tf*Rz_U*z_d);
Syy_U = (Az_U_tf*Sz_U_tf)/(Az_U_tf*Sz_U_tf+Bz_U_tf*Rz_U*z_d);
bodemag(Syb_U,Syy_U),grid,legend
```



La marge de module calculé à partir du max de Syy est **Δ**M = 3.7 > 0.5 => Robuste

La figure nous montre qu'on atténue tous les bruits de mesure d'au moins 10db au dela 2000 rad.s-1. Ce qui respecte largement le cahier des charges.

La simulation valide bien le cahier des charges :

Voir fichier simulink RST_sami_fahim_R2 :

- Pour l'asservissement choisir une entrée sinusoidale
- On peut rajouter des perturbations à différents niveau
- Pour observer la validation du cdc en régulation mettre une entrée nulle et rajouter une perturbation sur la sortie

Quatrième partie : Implémentation de la commande

IV.1 La différence entre le modèle ASI et le modèle précédent :

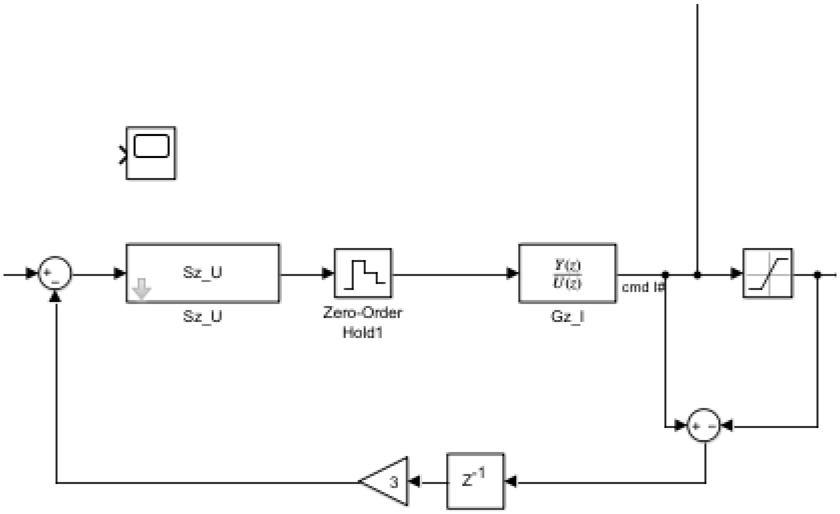
- Les résistances parasites ne sont pas négligées
- La sortie de notre correcteur de courant est un rapport cyclique et non un courant
- Le processus n'est plus linéaire, on n'a plus un modèle idéale, mais on est passé au modèle réel.

IV.2. Simulation du modèle Réel

Voir le fichier simulink ASI_sami_fahim_R2022a.slx

IV.3 Implémentation de l'anti saturation

La technique implémenté est un anti-windup, voici le schéma de la simulation zoomé
sur l'anti saturation de l'intégrale, représenté par la fonction Sz_U de la commande RST.



Conclusion

Le projet nous permet d'impléter un correcteur pour le contrôle du courant et de la tension d'une alimentation sans interruption.

Cette commande est adapté pour les systèmes linéraires, cependant le notre ne l'est pas parfaitement, nous devons implémenter des commande plus sophistiqué pour améliorer les résultats. On pourra utiliser **Contrôle prédictif basé sur le modèle (MPC - Model Predictive Control) par exemple.**