```
% BOIF1302
% DESM1210
clc
clear all
close all

warning off

%% Ajout de la trajectoire
Profile_Tracking
%la matrice-colonne de temps en secondes est dans ttrk
%la matrice-colonne de consigne en degrés est dans utrk

%% Fonctions de transferts
TF_AZ_num = [1.59e09];
TF_AZ_den = [1 1020.51 25082.705 3102480.725 64155612.5 82700000 0];

TF_EL_num = [7.95e09];
TF_EL_den = [1 1020.51 37082.705 15346520.725 320776412.5 413500000 0];

TF_AZ = tf(TF_AZ_num, TF_AZ_den);
TF_EL = tf(TF_EL_num, TF_EL_den);

%% Variables Télescope A
Mp_A = 25; %En pourcent
Ts_A = 1; %sec
Tr_10_A = 0.18; %sec

%Erreurs
ERP_unitaire_A = 0;
ERP_rampe_AZ_A = 0.03; %deg
ERP_rampe_EL_A = 0; %deg
ERP_para_EL_A = 0.08; %deg

%% Variables Télescope B
BW_B = 10; %rad/s
PM_B = 50; %deg +- 1 deg

%Erreurs
ERP_rampe_B = 0.005; %deg
t_ERP_rampe_B = 8; %sec









%% Conception spécifications Télescope A
disp("_____________________________________ A AZ _____________________________________")
Phi = rad2deg(atan(-pi/log(Mp_A/100)));
Zeta = cosd(Phi);

%On doit trouver le plus grand des Omega_n
%Omega_n = (1+(1.1*Zeta)+(1.4*(Zeta^2)))/Tr_10_A
Omega_n = 4/(Ts_A*Zeta);

%On trouve Omega_a pour simplifier les P_etoile
```

```
Omega_a = Omega_n*sqrt(1-Zeta^2);

%Ajustement de P_etoile
Ajout_Omega_n = -1.3; %-1.3
Ajout_Omega_a = -5; %-5

%On trouve P_etoile
P_etoile_A = (-Zeta*Omega_n + Ajout_Omega_n) + (Omega_a + Ajout_Omega_a)*i;

%% Calcul pour Avance phase Azimut Télescope A
disp("__________________ Av __________________")
frsp = evalfr(TF_AZ, P_etoile_A);
Angle_A_AZ = (rad2deg(angle(frsp)));
clear frsp

%Calculs qui aideront dans les étapes suivantes
Angle_A_AZ = Angle_A_AZ - 360;
Delta_Phi_A_AZ = - 180 - Angle_A_AZ;
Alpha_A_AZ = 180 - Phi;

%Trouver les angles des distances
Phi_Z_A_AZ = (Alpha_A_AZ+Delta_Phi_A_AZ)/2;
Phi_P_A_AZ = (Alpha_A_AZ-Delta_Phi_A_AZ)/2;

%Trouver les poles et zeros
Z_A_AZ = real(P_etoile_A)-(imag(P_etoile_A)/tand(Phi_Z_A_AZ));
P_A_AZ = real(P_etoile_A)-(imag(P_etoile_A)/tand(Phi_P_A_AZ));

%Cree une sous fonction de transfert pour trouver le Ka
TF_Ka_A_AZ2 = tf([1 -Z_A_AZ], [1 -P_A_AZ]);
TF_Ka_A_AZ = TF_Ka_A_AZ2 * TF_AZ;

%Calcul du K_AvPh_AZ
K_AvPh_A_AZ = 1/abs(evalfr(TF_Ka_A_AZ, P_etoile_A));

%Temporaire pour rapport
TF_Ka_A_AZ2 = TF_Ka_A_AZ2 * K_AvPh_A_AZ;
clear TF_Ka_A_AZ2

%Nouvelle fonction de transfert d'avance de phase
TF_AvPh_A_AZ = TF_Ka_A_AZ * K_AvPh_A_AZ;

%% Calcul pour retard phase cascades Azimut Télescope A
disp("__________________ Re __________________")
Diviser = 10;

%Trouver les valeurs des numérateurs et denominateur
[num_temp, den_temp] = tfdata(TF_AvPh_A_AZ, 'v');

%Trouver les K_etoile avec erreurs
Kvel_A_AZ = (num_temp(end))/(den_temp(end-1));
Kvel_etoile_A_AZ = 1/ERP_rampe_AZ_A;
K_etoile_A_AZ = Kvel_etoile_A_AZ/Kvel_A_AZ;
clear num_temp den_temp

%Trouver poles et zeros
Z_RePh_A_AZ = real(P_etoile_A)/Diviser;
P_RePh_A_AZ = Z_RePh_A_AZ/K_etoile_A_AZ;

%Cree une sous fonction de transfert pour trouver le Ka
TF_Kr_A_AZ2 = tf([1 -Z_RePh_A_AZ], [1 -P_RePh_A_AZ]);
TF_Kr_A_AZ = TF_Kr_A_AZ2 * TF_AvPh_A_AZ;
```

```
%Calcul du K_RePh_AZ
K_RePh_A_AZ = 1/abs(evalfr(TF_Kr_A_AZ, P_etoile_A));
%on voit que c'est environ 1 donc on change pour a
K_RePh_A_AZ = 1;

%Temporaire pour rapport
TF_Kr_A_AZ2 = TF_Kr_A_AZ2 * K_RePh_A_AZ;
clear TF_Kr_A_AZ2

%Nouvelle fonction de transfert d'avance de phase et retard
TF_RePh_AvPh_A_AZ = TF_Kr_A_AZ * K_RePh_A_AZ;

% TF_Finale_A_AZ = TF_RePh_AvPh_A_AZ
% TF_Finale_BF_A_AZ = feedback(TF_Finale_A_AZ, 1);

%% Coupe bande AZ Téléscope A
disp("_________________ Coupe _____________")
Omega_o = 54.8; %Peak sur le bode
X = 0.2; %
Kfcp = 1;

num_temp = Kfcp*[1 1 Omega_o.^2];
den_temp = [1 2*X*Omega_o Omega_o.^2];

TF_Coupe_Bande_A_AZ = tf(num_temp, den_temp);

clear num_temp den_temp Omega_o X Kfcp

%Mettre Coupe-Bande sur les fonctions de transferts
TF_Finale_A_AZ = TF_Coupe_Bande_A_AZ * TF_RePh_AvPh_A_AZ;
TF_Finale_BF_A_AZ = feedback(TF_Finale_A_AZ, 1);


%% Demande pour rapport
%Rlocus du système avec les P desirer
% figure
% hold on
% rlocus(TF_AZ, "red")
% rlocus(TF_Finale_A_AZ, "blue")
% scatter(real(P_etoile_A), imag(P_etoile_A), 50, "*", 'black')
% scatter(real(P_etoile_A), imag(-P_etoile_A), 50, "*", 'black')
% legend(["Original", "Finale", "Pole1", "Pole2"])
% xlim([-30, 10])
% ylim([-100, 100])

%Réponse à l'échelon unitaire
% figure;
% step(TF_Finale_BF_A_AZ);

%Erreur à une rampe uniaire du système
temps = [0:0.1:10];
Rampe = [0:0.1:10];
y_Rampe = lsim(TF_Finale_BF_A_AZ, Rampe, temps);

% figure
% hold on
% box on
% % plot(temps, y_Rampe', "blue")
% plot(temps, Rampe-y_Rampe', "red")
% % plot(temps, Rampe, "black")
% % legend(["Réponse", "Erreur", "Rampe"]);
```

```
% title("Erreur à une rampe unitaire");
% ylabel("Amplitude");
% xlabel("Time (secondes)")

%Diagramme de bode du système
% figure
% hold on
% margin(TF_Finale_A_AZ, "blue")
% margin(TF_AZ, "red")
% legend(["Modifier", "Original"])

%Erreur sur la trajectoire
% figure
% hold on
% box on
% y = lsim(TF_Finale_BF_A_AZ, utrk, ttrk);
% plot(ttrk, y, "blue")
% plot(ttrk, utrk-y, "red")
% plot(ttrk, utrk, "black")
% legend(["Réponse", "Erreur", "Trajectoire"]);
% %legend(["Erreur"])
% title("Erreur sur la trajectoire");
% ylabel("Amplitude");
% xlabel("Time (secondes)")


%% Validation système AZ
disp("_________________ Valid _____________")
%On vérifie Mp < 30%    Tr<0.25sec      ts< 1.20sec
stepinfo(TF_Finale_BF_A_AZ, RiseTimeThreshold=[0 1]);

%on vérifie GM > 10 dB      RM > 0.09s
[Gm, Pm, wcg, wcp] = margin(TF_Finale_A_AZ);
Pm;
Gm = 20*log10(Gm);
Rm = (Pm/wcp)*(pi/180);

Z_A_AZ;
P_A_AZ;

K_AvPh_A_AZ;
TF_AvPh_A_AZ;

Kvel_A_AZ;
K_etoile_A_AZ;

P_RePh_A_AZ;
Z_RePh_A_AZ;

K_RePh_A_AZ;
TF_RePh_AvPh_A_AZ;

%disp("Temps Erreur Rampe");
lsiminfo(Rampe-y_Rampe', temps);

                    %Effacer les non utiliser
                    clear Gm Pm wcg wcp
                    %Clear les variables
                    clear Kvel_A_AZ Kvel_etoile_A_AZ K_etoile_A_AZ Z_RePh_A_AZ P_RePh_A_AZ ↙
TF_Kr_A_AZ
                    clear P_etoile_A Ajout_Omega_n Ajout_Omega_a
                    %Clear les variables
```

```
clear TF_Ka_A_AZ P_A_AZ Z_A_AZ Phi_P_A_AZ Phi_Z_A_AZ Angle_A_AZ
Delta_Phi_A_AZ Alpha_A_AZ
```

```
%% Calcul pour avance phase cascades Elevation Télescope A
disp("_____ A EL _____")
%Ajustement de P_etoile
Ajout_Omega_n = -3; %-3
Ajout_Omega_a = 8; %8
Diviser = 7.2; %7.2

disp("_____ Av _____")
%On trouve P_etoile
P_etoile_A = (-Zeta*Omega_n + Ajout_Omega_n) + (Omega_a + Ajout_Omega_a)*i;

frsp = evalfr(TF_EL, P_etoile_A);
Angle_A_EL = (rad2deg(angle(frsp)));
clear frsp

%Calculs qui aideront dans les étapes suivantes
Angle_A_EL = Angle_A_EL - 360;
Delta_Phi_A_EL = - 180 - Angle_A_EL;
Alpha_A_EL = 180 - Phi;

%Trouver les angles des distances
Phi_Z_A_EL = (Alpha_A_EL+Delta_Phi_A_EL)/2;
Phi_P_A_EL = (Alpha_A_EL-Delta_Phi_A_EL)/2;

%Trouver les poles et zeros
Z_A_EL = real(P_etoile_A)-(imag(P_etoile_A)/tand(Phi_Z_A_EL));
P_A_EL = real(P_etoile_A)-(imag(P_etoile_A)/tand(Phi_P_A_EL));

%Cree une sous fonction de transfert pour trouver le Ka
TF_Ka_A_EL2 = tf([1 -Z_A_EL], [1 -P_A_EL]);
TF_Ka_A_EL = TF_Ka_A_EL2 * TF_EL;

%Calcul du K_AvPh_AZ
K_AvPh_A_EL = 1/abs(evalfr(TF_Ka_A_EL, P_etoile_A));

%Temporaire pour rapport
```

```
TF_Ka_A_EL2 = TF_Ka_A_EL2 * K_AvPh_A_EL;
clear TF_Ka_A_EL2

%Nouvelle fonction de transfert d'avance de phase
TF_AvPh_A_EL = TF_Ka_A_EL * K_AvPh_A_EL;

%% Calcul pour PI Elevation Télescope A
disp("__________________ PI __________________")

%Trouver les K_etoile avec erreurs
[num_temp, den_temp] = tfdata(TF_AvPh_A_EL, 'v');
Kvel_EL = (num_temp(end))/(den_temp(end-1));
clear num_temp den_temp

%Trouver le Ki
Ki_A_EL = 1 / (Kvel_EL * ERP_para_EL_A);

%Trouver les poles et zeros
Z_Re_A_EL = real(P_etoile_A) / Diviser;

%Calcul du K_PI_EL
K_Re_A_EL = -Ki_A_EL/Z_Re_A_EL;

%Cree une sous fonction de transfert pour trouver le Kpi
TF_Kpi_A_EL = K_Re_A_EL * tf([1 -Z_Re_A_EL], [1 0]);

%Temporaire pour rapport
TF_Kpi_A_EL;

%Nouvelle fonction de transfert de PI
TF_AvPh_PI_A_EL = TF_Kpi_A_EL * TF_AvPh_A_EL;

% TF_Finale_A_EL = TF_AvPh_PI_A_EL;
% TF_Finale_BF_A_EL = feedback(TF_Finale_A_EL, 1);

%% Coupe bande EL Téléscope A
disp("__________________ Coupe _____________")
Omega_o = 122.5; %Peak sur le bode
X = 0.2; %gere la largeur
Kfcp = 1;

num_temp = Kfcp*[1 1 Omega_o.^2];
den_temp = [1 2*X*Omega_o Omega_o.^2];

TF_Coupe_Bande_A_EL = tf(num_temp, den_temp);

clear num_temp den_temp Omega_o X Kfcp

%Mettre Coupe-Bande sur les fonctions de transferts
TF_Finale_A_EL = TF_Coupe_Bande_A_EL * TF_AvPh_PI_A_EL;
TF_Finale_BF_A_EL = feedback(TF_Finale_A_EL, 1);

%% Demande pour rapport
%Rlocus du système avec les P desirer
% figure
% hold on
% rlocus(TF_EL, "red")
% rlocus(TF_Finale_A_EL, "blue")
% scatter(real(P_etoile_A), imag(P_etoile_A), 50, "*", 'black')
% scatter(real(P_etoile_A), imag(-P_etoile_A), 50, "*", 'black')
% legend(["Original", "Finale", "Pole1", "Pole2"])
% xlim([-30, 10])
```

```
% ylim([-100, 100])

%Réponse à l'échelon unitaire
% figure;
% step(feedback(TF_Finale_A_EL,1));

%Erreur à une parabole uniaire du système
temps = [0:0.1:30];
Parabole = [0:0.1:30].^2;
Parabole = Parabole./2;
y_Parabole = lsim(TF_Finale_BF_A_EL, Parabole, temps);

% figure
% hold on
% box on
% plot(temps, y_Parabole', "blue")
% plot(temps, Parabole-y_Parabole', "red")
% plot(temps, Parabole, "black")
% legend(["Réponse", "Erreur", "Parabole"]);
% title("Erreur à une rampe unitaire");
% ylabel("Amplitude");
% xlabel("Time (secondes)")

%Diagramme de bode du système
% figure
% hold on
% margin(TF_Finale_A_EL, "blue")
% margin(TF_EL, "red")
% legend(["Modifier", "Original"])

%Erreur sur la trajectoire
% figure
% hold on
% box on
% y = lsim(TF_Finale_BF_A_EL, utrk, ttrk);
% %plot(ttrk, y, "blue")
% plot(ttrk, utrk-y, "red")
% %plot(ttrk, utrk, "black")
% %legend(["Réponse", "Erreur", "Trajectoire"]);
% legend(["Erreur"])
% title("Erreur sur la trajectoire");
% ylabel("Amplitude");
% xlabel("Time (secondes)")

%% Validation système EL
disp("_________________ Valid _____________")
%On vérifie Mp < 30%    Tr<0.25sec      ts< 1.20sec
stepinfo(TF_Finale_BF_A_EL, RiseTimeThreshold=[0 1]);

%on vérifie GM > 10 dB      RM > 0.09s
[Gm, Pm, wcg, wcp] = margin(TF_Finale_A_EL);
Pm;
Gm = 20*log10(Gm);
Rm = (Pm/wcp)*(pi/180);

Z_A_EL;
P_A_EL;

K_AvPh_A_EL;
TF_AvPh_A_EL;

Kvel_EL;
```

```
Z_Re_A_EL;

TF_AvPh_PI_A_EL;

%disp("Temps Erreur Parabole");
lsiminfo(Parabole-y_Parabole', temps);

                    %Effacer les non utiliser
                    clear Gm Pm wcg wcp Zeta Phi Rm Omega_a Omega_n Parabole Rampe temps
                    %Clear les variables
                    clear Kvel_EL Kvel_etoile_EL K_etoile_EL Z_RePh_EL P_RePh_EL TF_Ka_A_EL
                    clear P_etoile_A Ajout_Omega_n Ajout_Omega_a
                    %Clear les variables
                    clear TF_Ka_A_EL P_A_AZ Z_A_EL Phi_P_A_EL Phi_Z_A_EL Angle_A_EL↙
Delta_Phi_A_EL Alpha_A_EL
```

```
%% Conception spécifications Télescope B
disp("_____ B AZ _____")
%Ajustement
Ajout_Angle = -4.5; %-4.5
Multi_TF = 0.58; %0.58
Ajout_Omega_g = 2.3; %2.3
Diviser = 9; %9

%Calcul des valeurs demander pour le reste des calculs
Zeta_B = (0.5)*sqrt(tand(PM_B)*sind(PM_B));

Omega_g_B_AZ = BW_B * (sqrt(sqrt(1+(4*Zeta_B^4))-(2*Zeta_B^2))/(sqrt((1-(2*Zeta_B^2))+sqrt↙
((4*Zeta_B^4)-(4*Zeta_B^2)+2))));
Omega_g_B_AZ = Omega_g_B_AZ + Ajout_Omega_g;

%% Calcul pour avance phase Azimut Télescope B
disp("_____ Av _____")
K_etoile_B_AZ = 1 / abs(evalfr(TF_AZ, (Omega_g_B_AZ*i)));
PM_B_AZ = rad2deg(angle(evalfr(TF_AZ*K_etoile_B_AZ, (Omega_g_B_AZ*i)))) - 360 + 180;
```

```
Delta_phi_B_AZ = PM_B - PM_B_AZ + 5 + Ajout_Angle;
Alpha_B_AZ = (1 - sind(Delta_phi_B_AZ)) / (1 + sind(Delta_phi_B_AZ));

T_B_AZ = 1 / (Omega_g_B_AZ * sqrt(Alpha_B_AZ));

%On trouve pole et zeros
Z_B_AZ = -1 / T_B_AZ;
P_B_AZ = - 1 / (Alpha_B_AZ * T_B_AZ);

K_AvPh_A_AZ = Multi_TF * (K_etoile_B_AZ / sqrt(Alpha_B_AZ));

%Pour rapport
TF_AvPh_B_AZ2 =  K_AvPh_A_AZ * tf([1 -Z_B_AZ], [1 -P_B_AZ]);

%Fonctiond de transfert
TF_AvPh_B_AZ = TF_AvPh_B_AZ2 * TF_AZ;
[num_temp, den_temp] = tfdata(TF_AvPh_B_AZ, 'v');

%% Calcul pour retard phase Azimut Téléscope B
disp("_________________ Re _________________")

%On trouve les erreurs
Kvel_B_AZ = num_temp(end)/den_temp(end-1);
Kvel_etoile_B_AZ = 1 / (ERP_rampe_B);
clear num_temp den_temp

%Calcul du K_etoile
K_etoile_B_AZ = Kvel_etoile_B_AZ / Kvel_B_AZ;

%Trouver valeur de T pour fonction de transfert
T_B_AZ = Diviser / Omega_g_B_AZ;

%Trouver poles et zeros
Z_B_AZ = -1 / T_B_AZ;
P_B_AZ = -1 / (K_etoile_B_AZ * T_B_AZ);

%Kr
Kr = 1;

%Fonction de transfert
TF_RePh_B_AZ2 = Kr * tf([1 -Z_B_AZ], [1 -P_B_AZ]);

%Fonction de transfert finale
TF_Av_Re_B_AZ = TF_AvPh_B_AZ * TF_RePh_B_AZ2;

% TF_Finale_B_AZ = TF_Av_Re_B_AZ;
% TF_Finale_BF_B_AZ = feedback(TF_Finale_B_AZ, 1);

%% Coupe bande AZ Téléscope B
disp("_________________ Coupe _____________")
Omega_o = 54.8; %Peak sur le bode
X = 0.2;
Kfcp = 1;

num_temp = Kfcp*[1 1 Omega_o.^2];
den_temp = [1 2*X*Omega_o Omega_o.^2];

TF_Coupe_Bande_B_AZ = tf(num_temp, den_temp);

clear num_temp den_temp Omega_o X Kfcp
```

```
%Mettre Coupe-Bande sur les fonctions de transferts
TF_Finale_B_AZ = TF_Coupe_Bande_B_AZ * TF_Av_Re_B_AZ;
TF_Finale_BF_B_AZ = feedback(TF_Finale_B_AZ, 1);

%% Demande pour rapport
%Diagramme de bode du système
% figure
% hold on
% margin(TF_Finale_B_AZ, "blue")
% margin(TF_AZ, "red")
% legend(["Modifier", "Original"])

%Réponse à l'échelon unitaire
% figure;
% step(TF_Finale_BF_B_AZ);

%Erreur à une rampe uniaire du système
temps = [0:0.1:10];
Rampe = [0:0.1:10];
y_Rampe = lsim(TF_Finale_BF_B_AZ, Rampe, temps);

% figure
% hold on
% box on
% plot(temps, y_Rampe', "blue")
% plot(temps, Rampe-y_Rampe', "red")
% plot(temps, Rampe, "black")
% legend(["Réponse", "Erreur", "Rampe"]);
% title("Erreur à une rampe unitaire");
% ylabel("Amplitude");
% xlabel("Time (secondes)")

%Erreur sur la trajectoire
% figure
% hold on
% box on
% y = lsim(TF_Finale_BF_B_AZ, utrk, ttrk);
% plot(ttrk, y, "blue")
% plot(ttrk, utrk-y, "red")
% plot(ttrk, utrk, "black")
% legend(["Réponse", "Erreur", "Trajectoire"]);
% %legend(["Erreur"])
% title("Erreur sur la trajectoire");
% ylabel("Amplitude");
% xlabel("Time (secondes)")


%% Validation système AZ
disp("_________________ Valid _____________")
%on vérifie GM > 10 dB      RM > 0.09s
[Gm, Pm, wcg, wcp] = margin(TF_Finale_B_AZ);
Pm;
Gm = 20*log10(Gm);

[Gm, Pm, wcg, wcp] = margin(5*TF_Finale_B_AZ);
Rm = (Pm/wcp)*(pi/180);

BW_B_Calculer = bandwidth(TF_Finale_BF_B_AZ);

%disp("Temps Erreur Rampe à 8 sec");
Erreur = Rampe(find(temps == 8))-y_Rampe(find(temps == 8));
% lsiminfo(Rampe-y_Rampe', temps)
```

```
%On enleve les non-utiliser
clear Pm Gm Rm BW_B_Calculer Erreur
clear BW_B PM_B Ajout_BW Ajout_PM Zeta_B Omega_g_B_AZ
clear T_B_AZ Alpha_B_AZ Delta_phi_B_AZ PM_B_AZ K_etoile_B_AZ
clear Z_B_AZ P_B_AZ K_AvPh_A_AZ TF_AvPh_B_AZ2 TF_AvPh_B_AZ
clear Kvel_B_AZ Kvel_etoile_B_AZ K_etoile_B_AZ
clear T_B_AZ TF_RePh_B_AZ2 TF_Av_Re_B_AZ TF_Finale_B_AZ TF_Finale_B_AZ
clear Ajout_Angle Multi_TF Ajout_Omega_g
```

```
%% Conception spécifications Télescope B
disp("_____ B EL _____");
%Reset
BW_B = 10; %rad/s
PM_B = 50; %deg +- 1 deg

%Ajustement
Ajout_Angle = -4.5; %-4.5
Multi_TF = 0.7; %0.7
Ajout_Omega_g = 2.3;    %2.3
Diviser = 10; %10

%Calcul des valeurs demander pour le reste des calculs
Zeta_B = (0.5)*sqrt(tand(PM_B)*sind(PM_B));

Omega_g_B_EL = BW_B * (sqrt(sqrt(1+(4*Zeta_B^4))-(2*Zeta_B^2))/(sqrt((1-(2*Zeta_B^2))+sqrt
((4*Zeta_B^4)-(4*Zeta_B^2)+2))));
Omega_g_B_EL = Omega_g_B_EL + Ajout_Omega_g;

%% Calcul pour avance phase Elevation Télescope B
disp("_____ Av _____")
K_etoile_B_EL = 1 / abs(evalfr(TF_EL, (Omega_g_B_EL*i)));
PM_B_EL = rad2deg(angle(evalfr(TF_EL*K_etoile_B_EL, (Omega_g_B_EL*i)))) - 360 + 180;

Delta_phi_B_EL = PM_B - PM_B_EL + 5 + Ajout_Angle;
Alpha_B_EL = (1 - sind(Delta_phi_B_EL)) / (1 + sind(Delta_phi_B_EL));

T_B_EL = 1 / (Omega_g_B_EL * sqrt(Alpha_B_EL));

%On trouve pole et zeros
Z_B_EL = -1 / T_B_EL;
P_B_EL = - 1 / (Alpha_B_EL * T_B_EL);

K_AvPh_A_EL = Multi_TF * (K_etoile_B_EL / sqrt(Alpha_B_EL));
```

```
%Pour rapport
TF_AvPh_B_EL2 = K_AvPh_A_EL * tf([1 -Z_B_EL], [1 -P_B_EL]);

%Fonctiond de transfert
TF_AvPh_B_EL = TF_AvPh_B_EL2 * TF_EL;
[num_temp, den_temp] = tfdata(TF_AvPh_B_EL, 'v');

%% Calcul pour retard phase Elevtion Téléscope B
disp("__________________ Re __________________")

%On trouve les erreurs
Kvel_B_EL = num_temp(end)/den_temp(end-1);
Kvel_etoile_B_EL = 1 / (ERP_rampe_B);
clear num_temp den_temp

%Calcul du K_etoile
K_etoile_B_EL = Kvel_etoile_B_EL / Kvel_B_EL;

%Trouver valeur de T pour fonction de transfert
T_B_EL = Diviser / Omega_g_B_EL;

%Trouver poles et zeros
Z_B_EL = -1 / T_B_EL;
P_B_EL = -1 / (K_etoile_B_EL * T_B_EL);

%Kr
Kr = 1;

%Fonction de transfert
TF_RePh_B_EL2 = Kr * tf([1 -Z_B_EL], [1 -P_B_EL]);

%Fonction de transfert finale
TF_Av_Re_B_EL = TF_AvPh_B_EL * TF_RePh_B_EL2;

% TF_Finale_B_EL = TF_Av_Re_B_EL;
% TF_Finale_BF_B_EL = feedback(TF_Finale_B_EL, 1);

%% Coupe bande AZ Téléscope B
disp("__________________ Coupe _____________")
Omega_o = 122.5; %Peak sur le bode
X = 0.2; %
Kfcp = 1;

num_temp = Kfcp*[1 1 Omega_o.^2];
den_temp = [1 2*X*Omega_o Omega_o.^2];

TF_Coupe_Bande_B_EL = tf(num_temp, den_temp);

clear num_temp den_temp Omega_o X Kfcp

%Mettre Coupe-Bande sur les fonctions de transferts
TF_Finale_B_EL = TF_Coupe_Bande_B_EL * TF_Av_Re_B_EL;
TF_Finale_BF_B_EL = feedback(TF_Finale_B_EL, 1);

%% Demande pour rapport
%Diagramme de bode du système
% figure
% hold on
% margin(TF_Finale_B_EL, "blue")
% margin(TF_EL, "red")
% legend(["Modifier", "Original"])
```

```
%Réponse à l'échelon unitaire
% figure;
% step(TF_Finale_BF_B_EL);

%Erreur à une rampe uniaire du système
temps = [0:0.1:10];
Rampe = [0:0.1:10];
y_Rampe = lsim(TF_Finale_BF_B_EL, Rampe, temps);

% figure
% hold on
% box on
% plot(temps, y_Rampe', "blue")
% plot(temps, Rampe-y_Rampe', "red")
% plot(temps, Rampe, "black")
% legend(["Réponse", "Erreur", "Rampe"]);
% title("Erreur à une rampe unitaire");
% ylabel("Amplitude");
% xlabel("Time (secondes)")

figure
hold on
box on
% plot(temps, y_Rampe', "blue")
plot(temps, Rampe-y_Rampe', "red")
% plot(temps, Rampe, "black")
% legend(["Réponse", "Erreur", "Rampe"]);
title("Erreur à une rampe unitaire");
ylabel("Amplitude");
xlabel("Time (secondes)")

%Erreur sur la trajectoire
% figure
% hold on
% box on
% y = lsim(TF_Finale_BF_B_EL, utrk, ttrk);
% plot(ttrk, y, "blue")
% plot(ttrk, utrk-y, "red")
% plot(ttrk, utrk, "black")
% legend(["Réponse", "Erreur", "Trajectoire"]);
% %legend(["Erreur"])
% title("Erreur sur la trajectoire");
% ylabel("Amplitude");
% xlabel("Time (secondes)")


%% Validation système EL
disp("_________________ Valid _____________")
%on vérifie GM > 10 dB      RM > 0.09s
[Gm, Pm, wcg, wcp] = margin(TF_Finale_B_EL);
Pm;
Gm = 20*log10(Gm);

[Gm, Pm, wcg, wcp] = margin(5*TF_Finale_B_EL);
Rm = (Pm/wcp)*(pi/180);

BW_B_Calculer = bandwidth(TF_Finale_BF_B_EL);

%disp("Temps Erreur Rampe à 8 sec");
Erreur = Rampe(find(temps == 8))-y_Rampe(find(temps == 8));
% lsiminfo(Rampe-y_Rampe', temps)
```

```
%On enleve les non-utiliser
clear BW_B PM_B Ajout_BW Ajout_PM Zeta_B Omega_g_B_EL
clear T_B_EL Alpha_B_EL Delta_phi_B_EL PM_B_EL K_etoile_B_EL
clear Z_B_EL P_B_EL K_AvPh_A_EL TF_AvPh_B_EL2 TF_AvPh_B_EL
clear Kvel_B_EL Kvel_etoile_B_EL K_etoile_B_EL
clear T_B_EL TF_RePh_B_AZ2 TF_Av_Re_B_EL TF_Finale_B_EL TF_Finale_B_EL



%Assurer la fin du document
disp("Hello World")
```