Trabalho Computacional 2 de Sistemas Realimentados

Componentes do grupo: Amanda S. Bassani e João Paulo B. da Rocha

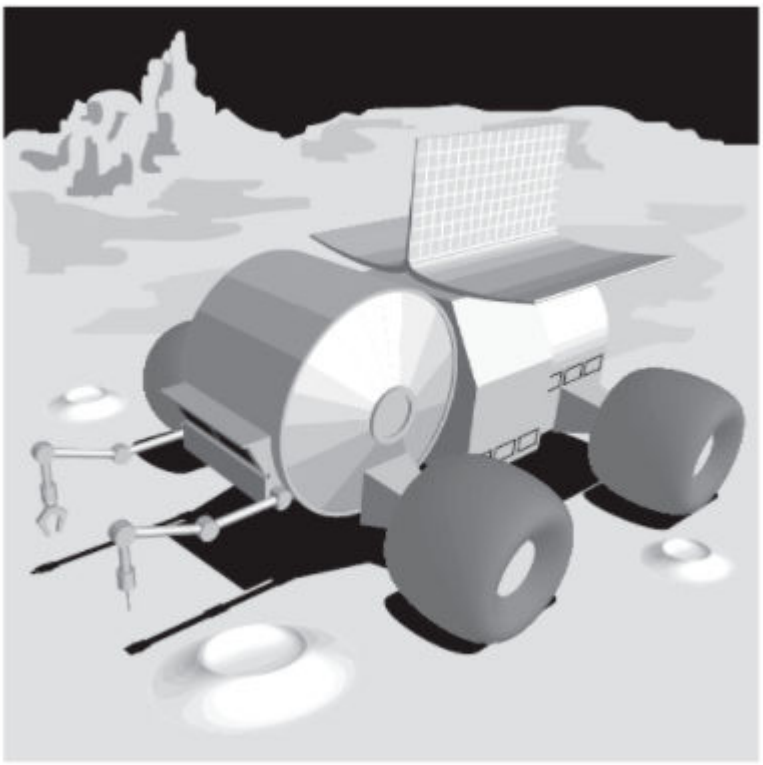
1.0 Inicialização

```
close all
clear all
clc
N = 11;
```

2.0 Projeto

A dinâmica do veiculo lunar (Sistemas de Controle Moderno 8ed. Dorf e Bishop. Capítulo 12 Sistemas de Controle Robusto. P12.10) é dada por:

$$G = \frac{100N}{(s + (25 - N))\left(s + \frac{25}{N}\right)} e^{-Ts}$$



2.1 Projetar controlador PID, considerando dois casos, com e sem atraso de transporte. Utilizando a resposta em frequência para que o sistema tenha as seguintes especificações:

- Erro ao degrau e ao distúrbio de degrau menores ou iguais a 0.1;
- Largura de banda FTMA maior possível;

- Margem de fase maior ou igual a 60º.

Caso 1. Sem atraso de transporte. T=0.

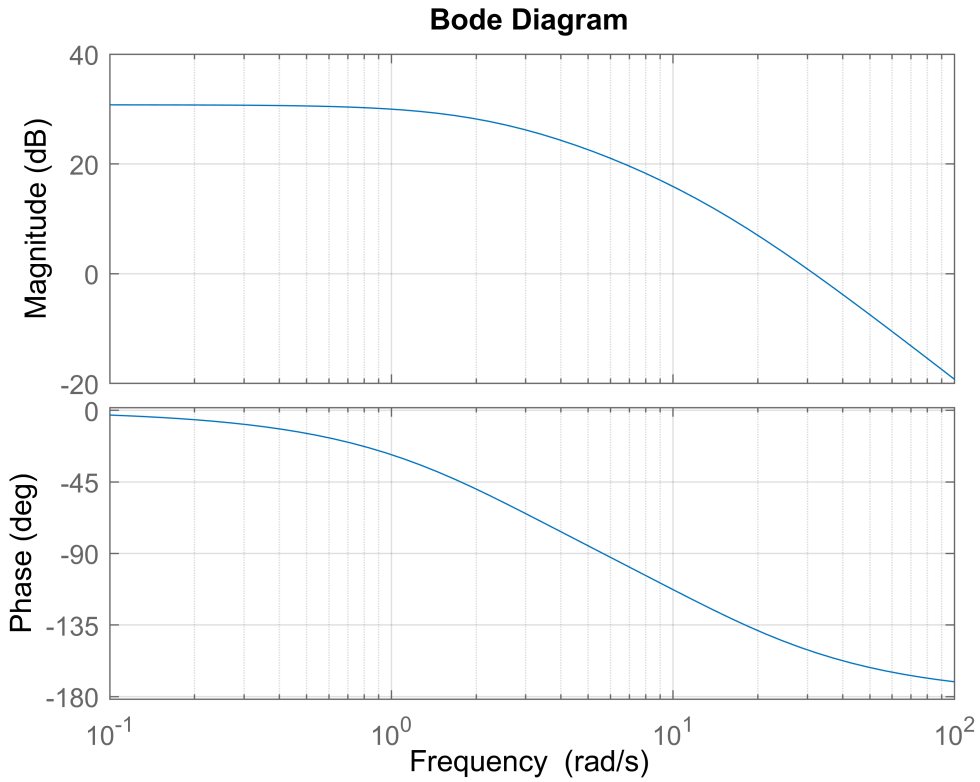
Caso 2. Com atraso de transporte. T=0.1/N.

Caso 1: T=0

```
syms s

T = 0; %Caso 1: T=0 | Caso 2: T=0.1/N
np = 100*N;
dp = conv([1 25-N],[1 25/N]);
gp = tf(np,dp,'InputDelay',T) % sistema tipo zero, erro ao degrau é diferente de zero e com atr
```

```
gp =

         1100
  --------------------
  s^2 + 16.27 s + 31.82

Continuous-time transfer function.
```

```
figure
w = logspace(-1,2);
bode(gp,w)
grid on
```



```
% 20*log(Kp) pelo gráfico de bode vale 30.8 dB.

%-------------------------------------------------------------------------------------------
% Calculo do erro em regime (s -> 0)
nps = poly2sym(np,s);
dps = poly2sym(dp,s);
gps = nps/dps;
    % Entrada degrau
sEs = 1/(1+gps);
ess = double(limit(sEs,s,0,'right'))
```

```
ess = 0.0281
```

```
    % Disturbio degrau
sEd = gps/(1+gps);
essd = double(limit(sEd,s,0,'right'))
```

```
essd = 0.9719
```

```
%-------------------------------------------------------------------------------------------
```

Com o aumento de K, diminui o erro em regime a entrada e a margem de fase. O erro a entrada degrau da planta atende as especificações, então qualquer valor escolhido para Kp do PI vai atender também.

Etapa 1: Calcular MG, MF e LB

```
[Gm,Pm,Wgm,Wpm]=margin(gp)
```

```
Gm = Inf
Pm = 27.9459
Wgm = Inf
Wpm = 31.6788
```

```
%MF de gp = 27.95 e largura de banda = 31.68
```

Etapa 2: Nova frequência de cruzamento de ganho

```
FaseGp = 25 - 180
```

```
FaseGp = -155
```

Olhando no gráfico de bode, a nova frequencia de cruzamento de ganho (w_0dB) deve ser 36.1 rad/s e o módulo nessa frequência é -2.12

Etapa 3: Calcular o ganho proporcional do PI (K_P)

```
% de maneira que G_MA=G_P×K_P tenha a margem de fase PM proximo de 25 graus na  nova
% frequência de cruzamento de ganho (w_0dB).
syms kpi
eq = 20*log10(kpi)-2.12;
kpi = double(solve(eq))
```

```
kpi = 1.2764
```

```
% kpi encontrado foi de 1.2764
```

```
gma=gp*kpi;
[Gma,Pma,Wgma,Wpma]=margin(gma)
```

```
Gma = Inf
Pma = 24.7683
Wgma = Inf
Wpma = 36.1488
```

```
gmf = feedback(gma,1);
% A margem de fase obtida foi de 24.77.

%-------------------------------------------------------------------------------------------
% Calculo do erro em regime (s -> 0)
s0 = 0;
ns = poly2sym(np*kpi,s);
dps = poly2sym(dp,s);
gps = ns/dps;

    % Entrada degrau
sEs = 1/(1+gps);
ess(1) = double(limit(sEs,s,0,'right'));
    % Disturbio degrau
sEd = gps/(1+gps);
essd(1) = double(limit(sEd,s,0,'right'));
%-------------------------------------------------------------------------------------------
```

Etapa 4: Escolher a frequência de corte do PI

```
% tal que o atraso de fase do PI ocorra um pouco abaixo da nova frequência de cruzamento de gan
% Em seguida, simular o PI com a planta, e verificar se a resposta é estável e rápida.

kii=kpi*Wpma/5;
pi=tf([kpi kii],[1 0])
```

```
pi =

  1.276 s + 9.228
  ---------------
        s

Continuous-time transfer function.
```

```
ggma=gp*pi;
[Gm,Pm,Wgm,Wpm]=margin(ggma)
```

```
Gm = Inf
Pm = 13.3366
Wgm = Inf
Wpm = 36.5229
```

```
%MF= 13.33 e LB=36.52 com PI
ggmf=feedback(ggma,1);
t=0.0:0.01:7;
y1=step(ggmf,t);
%step(ggmf); %resposta ao degrau
stepinfo(y1,t) % É rapida e estável
```

```
ans = struct with fields:
         RiseTime: 0.0304
    TransientTime: 0.8690
     SettlingTime: 0.8690
      SettlingMin: 0.5269
      SettlingMax: 1.7043
        Overshoot: 70.4252
       Undershoot: 0
             Peak: 1.7043
         PeakTime: 0.0900
```

```
%-------------------------------------------------------------------------------------------------
% Calculo do erro em regime (s -> 0)
pis = (kpi*s + kii)/s;

    % Entrada degrau
sEs = 1/(1+pis*gps);
ess(2) = double(limit(sEs,s,0,'right'));

    % Disturbio degrau
sEd = gps/(1+pis*gps);
essd(2) = double(limit(sEd,s,0,'right'));
%-------------------------------------------------------------------------------------------------
```

Etapa 5: Projetar PD > 1+sKdd

```
kdd=1/Wpma;
npid=conv([kpi kii],[kdd 1]);
dpid=[1 0];
pid=tf(npid,dpid)
```

```
pid =

  0.03531 s^2 + 1.532 s + 9.228
  -----------------------------
                s

Continuous-time transfer function.
```

```
gggma=gp*pid
```

```
gggma =

  38.84 s^2 + 1685 s + 1.015e04
  -----------------------------
    s^3 + 16.27 s^2 + 31.82 s

Continuous-time transfer function.
```

```
[Gm,Pm,Wgm,Wpm]=margin(gggma)
```

```
Gm = Inf
Pm = 63.1787
Wgm = NaN
Wpm = 47.3504
```

```
gggmf=feedback(gggma,1);
y2=step(gggmf,t);
```

```
stepinfo(y2,t);

%-------------------------------------------------------------------------------------------------
% Calculo do erro em regime (s -> 0)
npids = poly2sym(npid,s);
dpids = poly2sym(dpid,s);
pids = npids/dpids;

    % Entrada degrau
sEs = 1/(1+pids*gps);
ess(3) = double(limit(sEs,s,0,'right'));
    % Disturbio degrau
sEd = gps/(1+pids*gps);
essd(3) = double(limit(sEd,s,0,'right'));
%-------------------------------------------------------------------------------------------------

% Reprojetando o PD
kdd=5/Wpma;
npid=conv([kpi kii],[kdd 1]);
dpid=[1 0];
pid=tf(npid,dpid)
```

```
pid =

  0.1766 s^2 + 2.553 s + 9.228
  ----------------------------
              s

Continuous-time transfer function.
```

```
gggma2=gp*pid
```

```
gggma2 =

  194.2 s^2 + 2808 s + 1.015e04
  -----------------------------
    s^3 + 16.27 s^2 + 31.82 s

Continuous-time transfer function.
```

```
[Gm,Pm,Wgm,Wpm]=margin(gggma2)
```

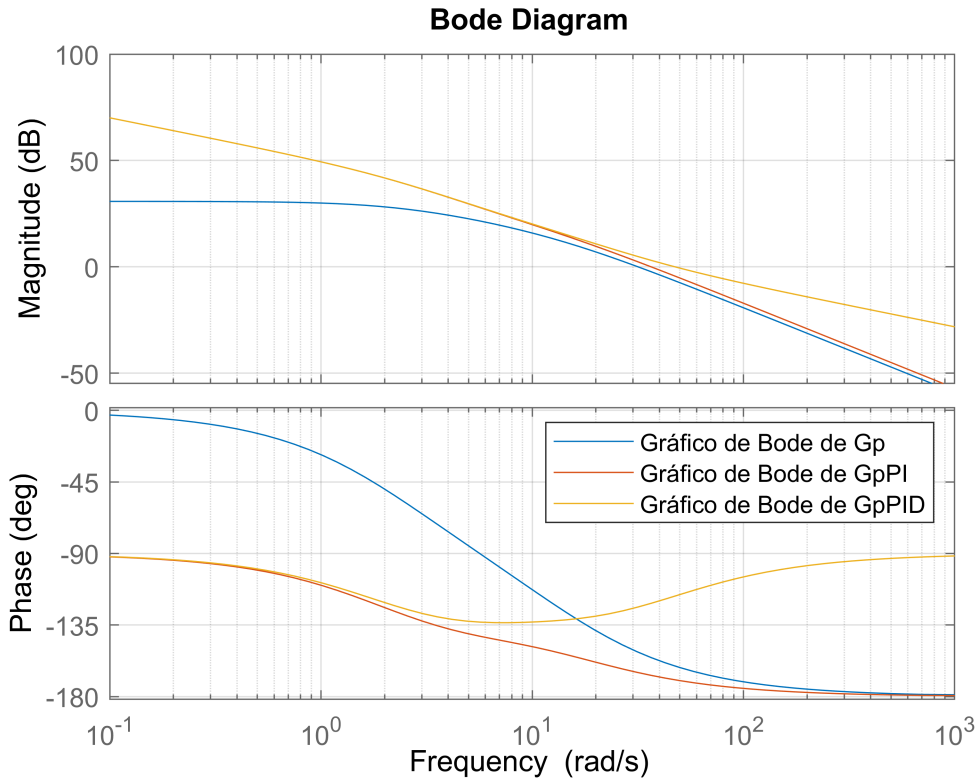
```
Gm = Inf
Pm = 90.5304
Wgm = NaN
Wpm = 193.9604
```

```
gggmf2=feedback(gggma2,1);
y3=step(gggmf2,t);
stepinfo(gggmf2)
```

```
ans = struct with fields:
         RiseTime: 0.0116
    TransientTime: 0.0220
     SettlingTime: 0.0220
      SettlingMin: 0.9010
      SettlingMax: 0.9989
        Overshoot: 0
       Undershoot: 0
```

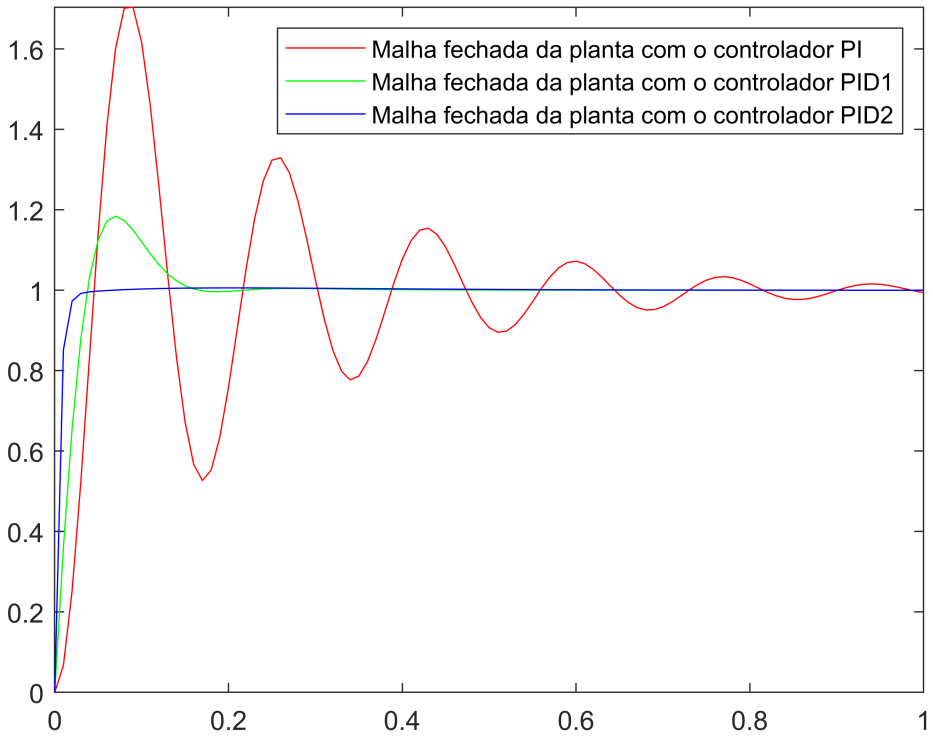
```
          Peak: 0.9989
      PeakTime: 0.0558
```

```
%-------------------------------------------------------------------------------------------------
% Calculo do erro em regime (s -> 0)
npids = poly2sym(npid,s);
dpids = poly2sym(dpid,s);
pids = npids/dpids;

    % Entrada degrau
sEs = 1/(1+pids*gps);
ess(4) = double(limit(sEs,s,0,'right'));
    % Disturbio degrau
sEd = gps/(1+pids*gps);
essd(4) = double(limit(sEd,s,0,'right'));
%-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
% Plots

figure
hold on
bode(gp);
bode(ggma);
bode(gggma);
grid on
legend('Gráfico de Bode de Gp','Gráfico de Bode de GpPI', 'Gráfico de Bode de GpPID');
```




```
figure
plot(t,y1,'r',t,y2,'g',t,y3,'b')
```

```
axis([0 1 0 inf])
legend('Malha fechada da planta com o controlador PI','Malha fechada da planta com o controlado
```




```
MG=0;MF=0;LB=0;OS=0;RT=0;ST=0;
[Gm,Pm,Wgm,Wpm]=margin(gma);
MG(1) = Gm;
MF(1) = Pm;
LB(1) = bandwidth(gmf);
OS(1)=stepinfo(gmf).Overshoot;
RT(1)=stepinfo(gmf).RiseTime;
ST(1)=stepinfo(gmf).SettlingTime;

[Gm,Pm,Wgm,Wpm]=margin(ggma);
MG(2) = Gm;
MF(2) = Pm;
LB(2) = bandwidth(ggmf);
OS(2)=stepinfo(ggmf).Overshoot;
RT(2)=stepinfo(ggmf).RiseTime;
ST(2)=stepinfo(ggmf).SettlingTime;

[Gm,Pm,Wgm,Wpm]=margin(gggma);
MG(3) = Gm;
MF(3) = Pm;
LB(3) = bandwidth(gggmf);
OS(3)=stepinfo(gggmf).Overshoot;
RT(3)=stepinfo(gggmf).RiseTime;
ST(3)=stepinfo(gggmf).SettlingTime;
```

```
[Gm,Pm,Wgm,Wpm]=margin(gggma2);
MG(4) = Gm;
MF(4) = Pm;
LB(4) = bandwidth(gggmf2);
OS(4)=stepinfo(gggmf2).Overshoot;
RT(4)=stepinfo(gggmf2).RiseTime;
ST(4)=stepinfo(gggmf2).SettlingTime;


col_Kp = [OS(1) RT(1) ess(1) essd(1) MG(1) MF(1) LB(1)]';
col_PI = [OS(2) RT(2) ess(2) essd(2) MG(2) MF(2) LB(2)]';
col_PID1 = [OS(3) RT(3) ess(3) essd(3) MG(3) MF(3) LB(3)]';
col_PID2 = [OS(4) RT(4) ess(4) essd(4) MG(4) MF(4) LB(4)]';
Tabela1=table(col_Kp,col_PI,col_PID1,col_PID2,'RowNames',{'Sobressinal (%)','T. Subida (s)','Er
```

Tabela1 = 7×4 table

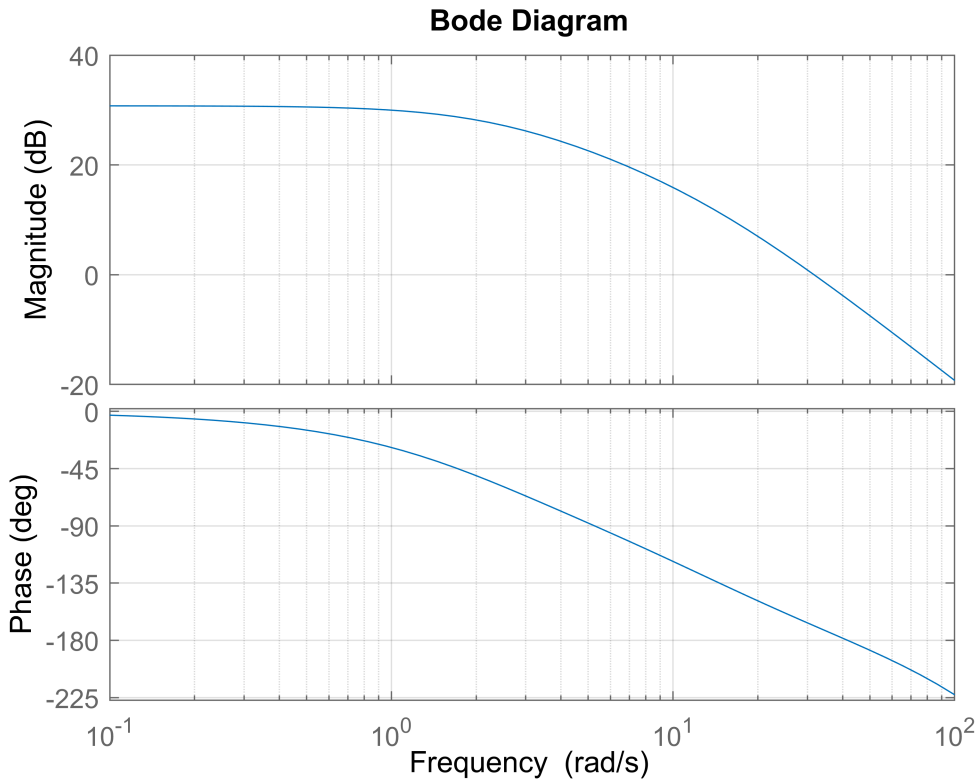
	Gp*Kp	Gp*PI	Gp*PID1	Gp*PID2
1 Sobressinal (%)	50.1243	71.4849	18.4296	0
2 T. Subida (s)	0.0323	0.0305	0.0286	0.0116
3 Erro à entrada degrau	0.0222	0	0	0
4 Erro ao disturbio de degrau	0.9778	0	0	0
5 MG	Inf	Inf	Inf	Inf
6 MF	24.7683	13.3366	63.1787	90.5304
7 LB	56.9235	57.3283	63.0945	191.6867

Caso 2: T=0.1/N

```
T = 0.1/N; %Caso 1: T=0 | Caso 2: T=0.1/N
np = 100*N;
dp = conv([1 25-N],[1 25/N]);
gpd = tf(np,dp,'InputDelay',T) % sistema tipo zero, erro ao degrau é diferente de zero e com at
```

```
gpd =

                          1100
  exp(-0.00909*s) * ---------------------
                    s^2 + 16.27 s + 31.82

Continuous-time transfer function.
```

```
w = logspace(-1,2);
figure
bode(gpd,w)
grid on
```




```
% 20*log(Kp) pelo gráfico de bode vale 30.8 dB.

%-------------------------------------------------------------------------------------------
% Calculo do erro em regime (s -> 0)
nps = poly2sym(np,s)*exp(-T*s);
dps = poly2sym(dp,s);
gps = nps/dps;
    % Entrada degrau
sEs = 1/(1+gps);
ess = double(limit(sEs,s,0,'right'))
```

```
ess = 0.0281
```

```
    % Disturbio degrau
sEd = gps/(1+gps);
essd = double(limit(sEd,s,0,'right'))
```

```
essd = 0.9719
```

```
%-------------------------------------------------------------------------------------------

% O módulo não se altera, logo segue o modelo anterior.
% Kp = 10^(30.8/20)
% ess = 1/(Kp+1) % erro em regime ao degrau sistema tipo 0
```

Com o aumento de K, diminui o erro em regime a entrada e a margem de fase. O erro a entrada degrau da planta atende as especificações, então qualquer valor escolhido para Kp do PI vai atender também.

Etapa 1: Calcular MG, MF e LB

```
[Gm,Pm,Wgm,Wpm]=margin(gpd);
Pm
```

```
Pm = 11.4453
```

```
Wpm
```

```
Wpm = 31.6788
```

```
%MF de gp = 11.44 e largura de banda = 31.68
```

Etapa 2: Nova frequência de cruzamento de ganho

```
FaseGp = 25 - 180;
% Olhando no gráfico de bode, a nova frequencia de cruzamento de ganho (w_0dB)
% deve ser aproximadamente 24.4 rad/s e o módulo nessa frequência é 4.04
```

Etapa 3: Calcular o ganho proporcional do PI (K_P)

```
% de maneira que G_MA=G_P×K_Ptenha a margem de fase PM proximo de 25 graus
% na  nova frequência de cruzamento de ganho (w_0dB).

syms kpi
eq = 20*log10(kpi)+4.04;
kpi = double(solve(eq))
```

```
kpi = 0.6281
```

```
gma=gpd*kpi;
[Gma,Pma,Wgma,Wpma]=margin(gma);

gmf = feedback(gma,1);
%-------------------------------------------------------------------------------------------------
% Calculo do erro em regime (s -> 0)
ns = nps*kpi;
gps = ns/dps;

    % Entrada degrau
sEs = 1/(1+gps);
ess(1) = double(limit(sEs,s,0,'right'));
    % Disturbio degrau
sEd = gps/(1+gps);
essd(1) = double(limit(sEd,s,0,'right'));
%-------------------------------------------------------------------------------------------------
```

Etapa 4: Escolher a frequência de corte do PI

```
% tal que o atraso de fase do PI ocorra um pouco abaixo da nova frequência de cruzamento de gan
% Em seguida, simular o PI com a planta, e verificar se a resposta é estável e rápida.

kii=kpi*Wpma/5;
```

```
pi=tf([kpi kii],[1 0])
```

```
pi =

  0.6281 s + 3.069
  ----------------
         s

Continuous-time transfer function.
```

```
ggma=gpd*pi
```

```
ggma =

                          690.9 s + 3376
  exp(-0.00909*s) * -------------------------
                    s^3 + 16.27 s^2 + 31.82 s

Continuous-time transfer function.
```

```
[Gm,Pm,Wgm,Wpm]=margin(ggma)
```

```
Gm = 1.8271
Pm = 10.7464
Wgm = 34.3215
Wpm = 24.7003
```

```
ggmf=feedback(ggma,1);

t=0.0:0.01:7;
y1 = step(ggmf,t); % resposta ao degrau
stepinfo(y1,t) % É rapida e estável
```

```
ans = struct with fields:
         RiseTime: 0.0429
    TransientTime: 1.6313
     SettlingTime: 1.6313
      SettlingMin: 0.4439
      SettlingMax: 1.7846
        Overshoot: 78.4617
       Undershoot: 0
             Peak: 1.7846
         PeakTime: 0.1300
```

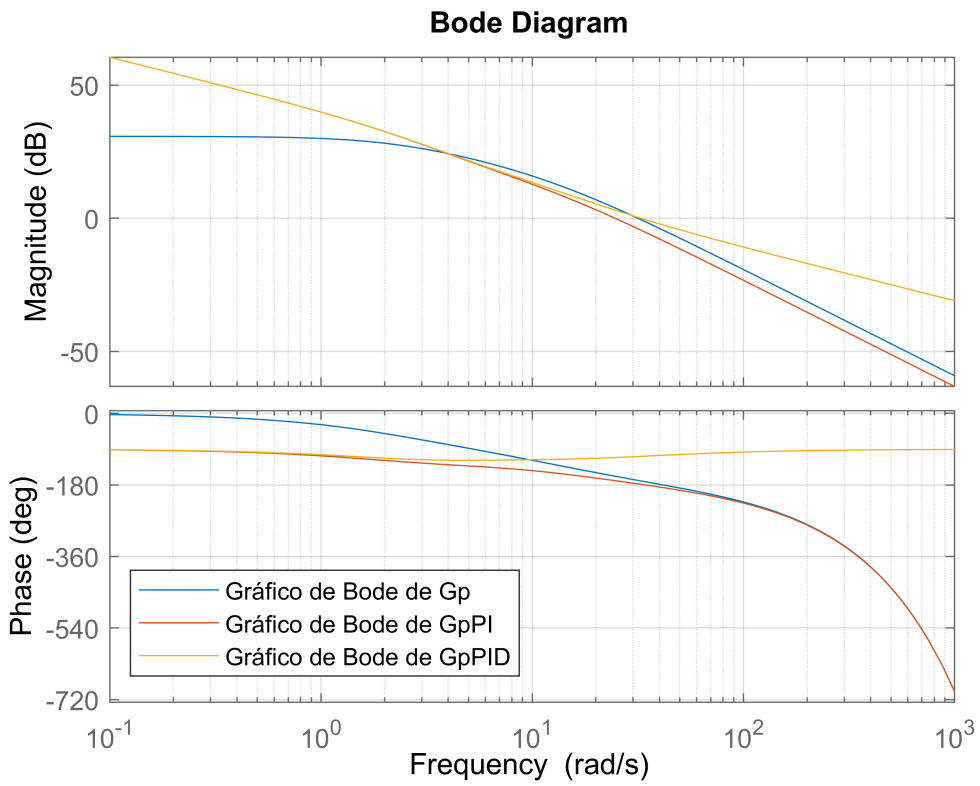
```
%-------------------------------------------------------------------------------------------
% Calculo do erro em regime (s -> 0)
pis = (kpi*s + kii)/s;

    % Entrada degrau
sEs = 1/(1+pis*gps);
ess(2) = double(limit(sEs,s,0,'right'));

    % Disturbio degrau
sEd = gps/(1+pis*gps);
essd(2) = double(limit(sEd,s,0,'right'));
%-------------------------------------------------------------------------------------------
```

Etapa 5: Projetar PD > 1+sKdd

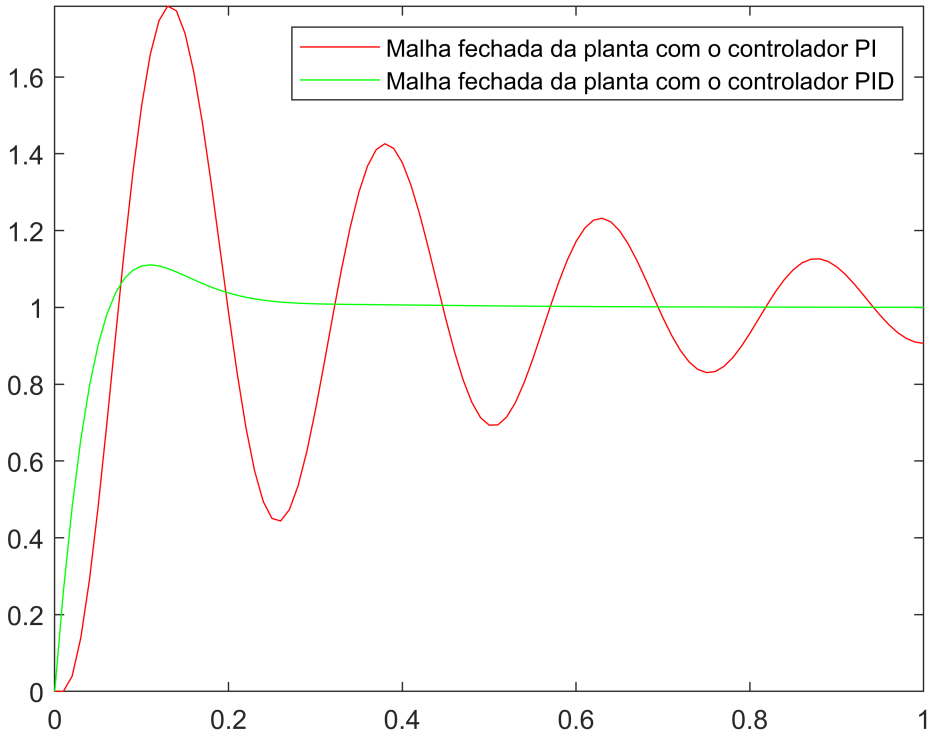
```
kdd=1/Wpma;
npid=conv([kpi kii],[kdd 1]);
dpid=[1 0];
pid=tf(npid,dpid);
gggma=gp*pid;
[Gm,Pm,Wgm,Wpm]=margin(gggma)
```

```
Gm = Inf
Pm = 71.9005
Wgm = NaN
Wpm = 32.7266
```

```

gggmf=feedback(gggma,1);
y2=step(gggmf,t);
stepinfo(y2,t);

%-----------------------------------------------------------------------------------------
% Calculo do erro em regime (s -> 0)
npids = poly2sym(npid,s);
dpids = poly2sym(dpid,s);
pids = npids/dpids;

    % Entrada degrau
sEs = 1/(1+pids*gps);
ess(3) = double(limit(sEs,s,0,'right'));
    % Disturbio degrau
sEd = gps/(1+pids*gps);
essd(3) = double(limit(sEd,s,0,'right'));
%-----------------------------------------------------------------------------------------

%-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
% Plots

figure
hold on
bode(gpd);
bode(ggma);
bode(gggma);
grid on
legend('Gráfico de Bode de Gp','Gráfico de Bode de GpPI', 'Gráfico de Bode de GpPID','Position'
```




```
figure
plot(t,y1,'r',t,y2,'g')
axis([0 1 0 inf])
legend()
legend('Malha fechada da planta com o controlador PI','Malha fechada da planta com o controlado
```




```
MG=0;MF=0;LB=0;OS=0;RT=0;ST=0;
[Gm,Pm,Wgm,Wpm]=margin(gma);
MG(1) = Gm;
MF(1) = Pm;
LB(1) = bandwidth(gmf);
OS(1)=stepinfo(gmf).Overshoot;
RT(1)=stepinfo(gmf).RiseTime;
ST(1)=stepinfo(gmf).SettlingTime;

[Gm,Pm,Wgm,Wpm]=margin(ggma);
MG(2) = Gm;
MF(2) = Pm;
LB(2) = bandwidth(ggmf);
OS(2)=stepinfo(ggmf).Overshoot;
RT(2)=stepinfo(ggmf).RiseTime;
ST(2)=stepinfo(ggmf).SettlingTime;

[Gm,Pm,Wgm,Wpm]=margin(gggma);
MG(3) = Gm;
MF(3) = Pm;
LB(3) = bandwidth(gggmf);
OS(3)=stepinfo(gggmf).Overshoot;
RT(3)=stepinfo(gggmf).RiseTime;
ST(3)=stepinfo(gggmf).SettlingTime;


col_Kp = [OS(1) RT(1) ess(1) essd(1) MG(1) MF(1) LB(1)]';
```

```
col_PI = [OS(2) RT(2) ess(2) essd(2) MG(2) MF(2) LB(2)]';
col_PID = [OS(3) RT(3) ess(3) essd(3) MG(3) MF(3) LB(3)]';
Tabela1=table(col_Kp,col_PI,col_PID,'RowNames',{'Sobressinal (%)','T. Subida (s)','Erro à entra
```

Tabela1 = 7×3 table

	Gp*Kp	Gp*PI	Gp*PID
1 Sobressinal (%)	55.4731	77.6577	11.0855
2 T. Subida (s)	0.0450	0.0426	0.0461
3 Erro à entrada degrau	0.0440	0	0
4 Erro ao disturbio de degrau	0.9560	0	0
5 MG	2.6539	1.8271	Inf
6 MF	22.4051	10.7464	71.9005
7 LB	40.1701	39.7858	40.8244

2.2 Para o caso em que T=0 seg, multiplique o controlador projetado pela FT G do veículo lunar e determine a equação de estados deste sistema em malha aberta. Em seguida, desenvolva um código no matlab, semelhante ao fornecido para simular o avião (programa "simulaviao.m" fornecido em anexo), para simular o sistema de controle de direção do veículo lunar em malha fechada no espaço de estados. Considere nas simulações os seguintes casos:

Caso 1. Adicionar ruído Gaussiano na saída do sistema de controle, referente ao sensor do veículo lunar.

Caso 2. Sistema de controle sem o ruído de medição.

2.3 Compare a resposta ao degrau obtida usando o simulador desenvolvido no item 1.2 (sem o ruído de medição), com a resposta ao degrau (sobressinal, tempo de subida) obtida no item 1.1 para T=0 s.

2.4 Controlador Avanço-Atraso

Caso 1: T=0

Etapa 1: Determinar ganho k1 para atender as especificações do erro em regime e da largura de banda

```
%-------------------------------------------------------------------------------------------
% Calculo do erro em regime (s -> 0)
nps = poly2sym(np,s);
dps = poly2sym(dp,s);
gps = nps/dps;
    % Entrada degrau
sEs = 1/(1+gps);
ess = double(limit(sEs,s,0,'right'));
    % Disturbio degrau
sEd = gps/(1+gps);
```

```
essd = double(limit(sEd,s,0,'right'));
%-------------------------------------------------------------------------------------------------
% 20*log(Kp) pelo gráfico de bode vale 30.8 dB.
Kp = 10^(30.8/20)
```

```
Kp = 34.6737
```

```
ess = 1/(Kp+1) % erro em regime ao degrau sistema tipo 0
```

```
ess = 0.0280
```

Com o aumento de K, diminui o erro em regime a entrada e a margem de fase. O erro ao disturbio será reduzido em passos adiante quando pudermos aumentar a margem de fase.

```
k_atraso = 1;

%-------------------------------------------------------------------------------------------------
% Calculo do erro em regime (s -> 0)
ns = poly2sym(np*k_atraso,s);
dps = poly2sym(dp,s);
gps = ns/dps;

    % Entrada degrau
sEs = 1/(1+gps);
ess(1) = double(limit(sEs,s,0,'right'));
    % Disturbio degrau
sEd = gps/(1+gps);
essd(1) = double(limit(sEd,s,0,'right'));
%-------------------------------------------------------------------------------------------------
```

Etapa 2: Projetar o controlador atraso de fase

Passo 2: MG, MF, frequências de cruzamento de ganho e de fase e a largura de banda

```
[Gm,Pm,Wgm,Wpm]=margin(k_atraso*gp);
Pm
```

```
Pm = 27.9459
```

```
Wpm
```

```
Wpm = 31.6788
```

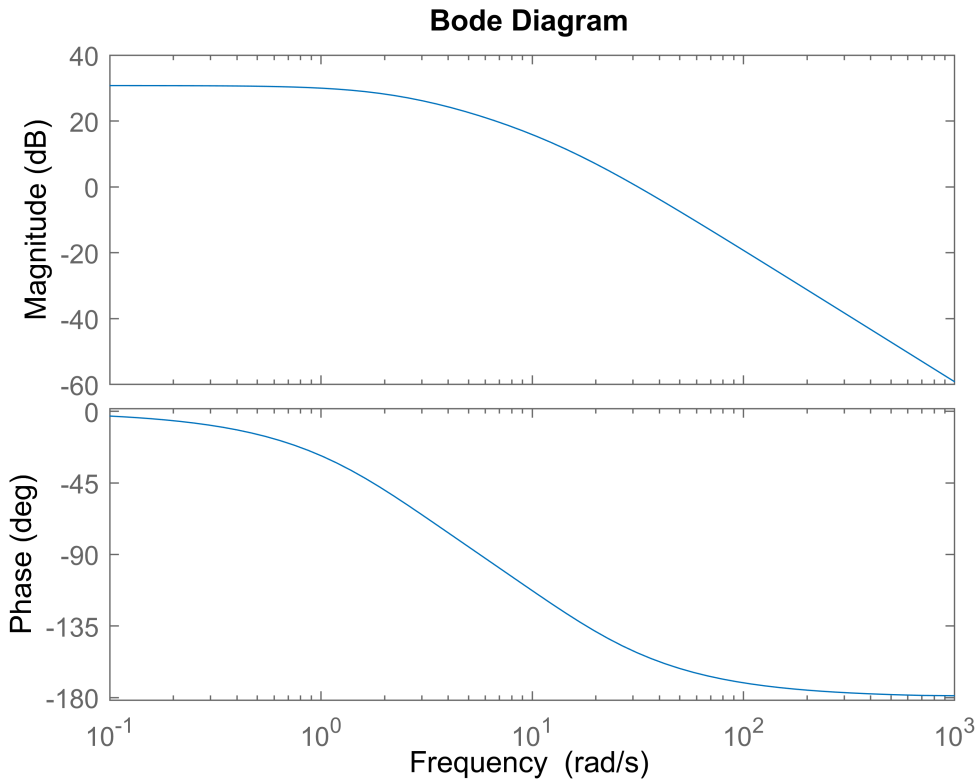
```
% MF de gp = 27.95 e largura de banda = 31.68
```

Passo 3: Nova frequência de cruzamento de ganho

```
FaseGp = 30 - 180
```

```
FaseGp = -150
```

```
bode(k_atraso*gp)
```




```
% Olhando no gráfico de bode, a nova frequencia de cruzamento de ganho (w_0dB)
% deve ser 29.35 rad/s e o módulo nessa frequência é 1,2.
mod = 1.2;
```

Passo 4: Obter a constante alpha resolvendo a equação

```
syms alpha1
eq = 20*log10(mod) + 20*log10(alpha1);
alpha1 = double(solve(eq))
```

```
alpha1 = 0.8333
```

Passo 5: Escolher a frequência de corte do zero do compensador uma década abaixo (29.35/10 = 2.935) da nova frequência de cruzamento de ganho e encontrar a frequência de corte do polo do compensador

```
syms tau1
eq = 1/(alpha1*tau1) - 2.935;
tau1 = double(solve(eq))
```

```
tau1 = 0.4089
```

```
1/(alpha1*tau1) % freq de corte do zero do compensador
```

```
ans = 2.9350
```

```
1/(tau1) % freq de corte do polo do compensador
```

```
ans = 2.4458
```

```
syms jw
g_atrasos = k_atraso*(1+jw*alpha1*tau1)/(1+jw*tau1);

%-------------------------------------------------------------------------------------------------
% Calculo do erro em regime (s -> 0)
g_atrasos = subs(g_atrasos,jw,s);
    % Entrada degrau
sEs = 1/(1+g_atrasos*gps);
ess(1) = double(limit(sEs,s,0,'right'));

    % Disturbio degrau
sEd = gps/(1+g_atrasos*gps);
essd(1) = double(limit(sEd,s,0,'right'));
%-------------------------------------------------------------------------------------------------
[num,den]=numden(g_atrasos);
g_atraso = tf(sym2poly(num),sym2poly(den))
```

```
g_atraso =

  200 s + 587
  -----------
  240 s + 587

Continuous-time transfer function.
```

Etapa 3:

```
gma_linha = g_atraso * gp;
[Gm,Pm,Wgm,Wpm]=margin(gma_linha)
```

```
Gm = Inf
Pm = 29.5822
Wgm = Inf
Wpm = 28.6790
```

```
gmf_linha = feedback(gma_linha,1);
y1 = 0;
t=0:.001:1;
y1 = step(gmf_linha,t);
% Margem de fase alcançada foi de 29.58
```

Etapa 4:

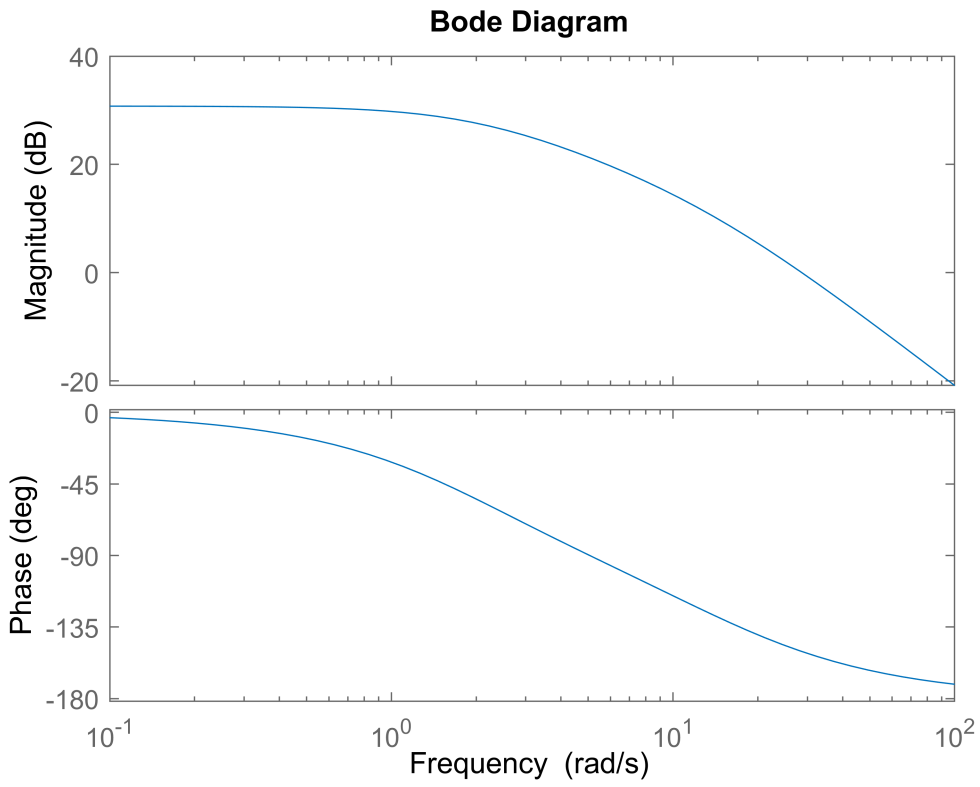
```
k_avanco = 1;
FaseGma = 70 - Pm; % margem de fase especificada 60, +10 de folga
syms alpha2
eq=(alpha2-1)/(2*sqrt(alpha2))-tand(FaseGma);
alpha2 = double(solve(eq))
```

```
alpha2 = 4.6876
```

```
% 5:
mod = -10*log10(alpha2)
```

```
mod = -6.7095
```

```
bode(gma_linha,w)
```




```
% Olhando no diagrama de Bode, wm será 46,52 rad/s.
wm = 46.52;
% 6:
tau2 = 1/(wm*sqrt(alpha2));
1/(alpha2*tau2) % freq de corte do zero do compensador
```

```
ans = 21.4865
```

```
1/(tau2) % freq de corte do polo do compensador
```

```
ans = 100.7195
```

```
g_avancos = k_avanco*(1+jw*alpha2*tau2)/(1+jw*tau2);

%-------------------------------------------------------------------------------------------
% Calculo do erro em regime (s -> 0)
g_avancos = subs(g_avancos,jw,s);
    % Entrada degrau
sEs = 1/(1+g_avancos*g_atrasos*gps);
ess(2) = double(limit(sEs,s,0,'right'));

    % Disturbio degrau
sEd = gps/(1+g_avancos*g_atrasos*gps);
essd(2) = double(limit(sEd,s,0,'right'));
%-------------------------------------------------------------------------------------------

[num,den]=numden(g_avancos);
```

```
g_avanco = tf(sym2poly(num),sym2poly(den))
```

```
g_avanco =

  7.552e30 s + 1.623e32
  ---------------------
  1.611e30 s + 1.623e32

Continuous-time transfer function.
```

```
gma = g_avanco * g_atraso * gp;
[Gm,Pm,Wgm,Wpm]=margin(gma)
```

```
Gm = Inf
Pm = 61.1605
Wgm = Inf
Wpm = 41.9438
```

```
gmf = feedback(gma,1);
up5(2) = stepinfo(gmf).Overshoot; %vetor para questão 5
tr5(2) = stepinfo(gmf).RiseTime;  %vetor para questão 5
t=0:.001:1;
y2 = 0;
y2 = step(gmf,t);
%-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
% Plots
figure
plot(t,y1,'r',t,y2,'g')
axis([0 1 0 inf])
legend('Malha fechada da planta com o controlador Atraso','Malha fechada da planta com o contro
```

```
MG=0;MF=0;LB=0;OS=0;RT=0;ST=0;
[Gm,Pm,Wgm,Wpm]=margin(gma_linha);
MG(1) = Gm;
MF(1) = Pm;
LB(1) = bandwidth(gmf_linha);
OS(1)=stepinfo(gmf_linha).Overshoot;
RT(1)=stepinfo(gmf_linha).RiseTime;
ST(1)=stepinfo(gmf_linha).SettlingTime;

[Gm,Pm,Wgm,Wpm]=margin(gma);
MG(2) = Gm;
MF(2) = Pm;
LB(2) = bandwidth(gmf);
OS(2)=stepinfo(gmf).Overshoot;
RT(2)=stepinfo(gmf).RiseTime;
ST(2)=stepinfo(gmf).SettlingTime;

col_atraso = [OS(1) RT(1) ess(1) essd(1) MG(1) MF(1) LB(1)]';
col_avancoatraso = [OS(2) RT(2) ess(2) essd(2) MG(2) MF(2) LB(2)]';
Tabela1=table(col_atraso,col_avancoatraso,'RowNames',{'Sobressinal (%)','T. Subida (s)','Erro à
```

Tabela1 = 7×2 table

	Gp*Atraso	Gp*Avanço-Atraso
1 Sobressinal (%)	43.5057	11.3874
2 T. Subida (s)	0.0415	0.0302
3 Erro à entrada degrau	0.0281	0.0281
4 Erro ao disturbio de degrau	0.9719	0.9719
5 MG	Inf	Inf
6 MF	29.5822	61.1605
7 LB	45.4398	67.1075

Caso 2: T=0.1/N

Etapa 1: Determinar ganho k1 para atender as especificações do erro em regime e da largura de banda

```
% bode(gpd)
% 20*log(Kp) pelo gráfico de bode vale 30.8 dB.
%-------------------------------------------------------------------------------------------
% Calculo do erro em regime (s -> 0)
nps = poly2sym(np,s)*exp(-T*s);
dps = poly2sym(dp,s);
gps = nps/dps;
    % Entrada degrau
sEs = 1/(1+gps);
ess = double(limit(sEs,s,0,'right'))
```

```
ess = 0.0281
```

```
    % Disturbio degrau
sEd = gps/(1+gps);
essd = double(limit(sEd,s,0,'right'))
```

```
essd = 0.9719
```

```
%-----------------------------------------------------------------------------------------------------

Kp = 10^(30.8/20)
```

```
Kp = 34.6737
```

```
ess = 1/(Kp+1) % erro em regime ao degrau sistema tipo 0
```

```
ess = 0.0280
```

```
% Com o aumento de K, diminui o erro em regime e a margem de fase e aumenta
% a largura de banda.
k_atraso = 1;
```

Etapa 2: Projetar o controlador atraso de fase

Passo 2: MG, MF, frequências de cruzamento de ganho e de fase e a largura de banda

```
[Gm,Pm,Wgm,Wpm]=margin(k_atraso*gpd);
Pm
```

```
Pm = 11.4453
```

```
Wpm
```

```
Wpm = 31.6788
```

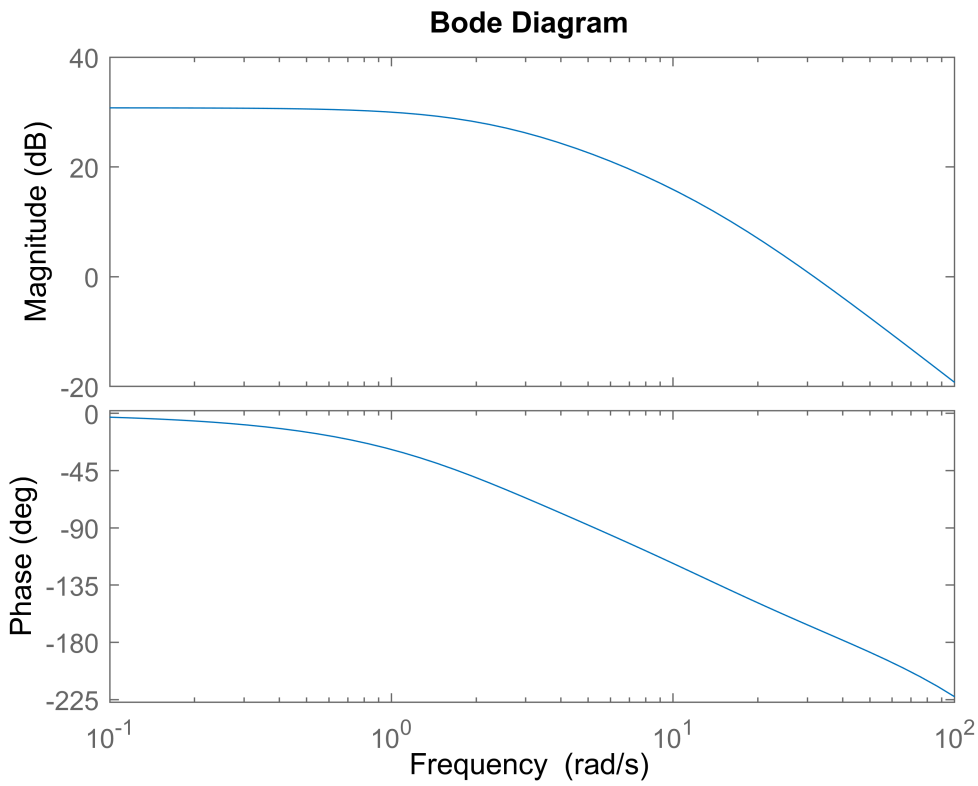
```
% MF de gp = 11.44 e largura de banda = 31.68
```

Passo 3: Nova frequência de cruzamento de ganho

```
FaseGp = 30 - 180
```

```
FaseGp = -150
```

```
bode(k_atraso*gpd,w)
```




```
% Olhando no gráfico de bode, a nova frequencia de cruzamento de ganho (w_0dB)
% deve ser 20.5 rad/s e o módulo nessa frequência é 6,6.
mod = 6.6;
```

Passo 4: Obter a constante alpha resolvendo a equação

```
syms alpha1
eq = 20*log10(mod) + 20*log10(alpha1);
alpha1 = double(solve(eq))
```

```
alpha1 = 0.1515
```

```
% Etapa 5: Escolher a frequência de corte do zero do compensador uma década
% abaixo (20.5/10 = 2.05) da nova frequência de cruzamento de ganho e
% encontrar a frequência de corte do polo do compensador
syms tau1
eq = 1/(alpha1*tau1) - 2.05;
tau1 = double(solve(eq))
```

```
tau1 = 3.2195
```

```
1/(alpha1*tau1) % freq de corte do zero do compensador
```

```
ans = 2.0500
```

```
1/(tau1) % freq de corte do polo do compensador
```

```
ans = 0.3106
```

```
syms jw
```

```
g_atrasos = k_atraso*(1+jw*alpha1*tau1)/(1+jw*tau1);
%-------------------------------------------------------------------------------------------------
% Calculo do erro em regime (s -> 0)
g_atrasos = subs(g_atrasos,jw,s);
% Entrada degrau
sEs = 1/(1+g_atrasos*gps);
ess(1) = double(limit(sEs,s,0,'right'));
% Disturbio degrau
sEd = gps/(1+g_atrasos*gps);
essd(1) = double(limit(sEd,s,0,'right'));
%-------------------------------------------------------------------------------------------------
[num,den]=numden(g_atrasos);
g_atraso = tf(sym2poly(num),sym2poly(den))
```

```
g_atraso =

  20 s + 41
  ----------
  132 s + 41

Continuous-time transfer function.
```

Etapa 3:

```
gmad_linha = g_atraso * gpd;
[Gm,Pm,Wgm,Wpm]=margin(gmad_linha)
```

```
Gm = 9.8002
Pm = 53.2299
Wgm = 39.2142
Wpm = 9.7247
```

```
% Margem de fase alcançada foi de 53.23
gmfd_linha = feedback(gmad_linha,1);
y1 = 0;
t=0:.001:5;
y1 = step(gmfd_linha,t);
```

Etapa 4:

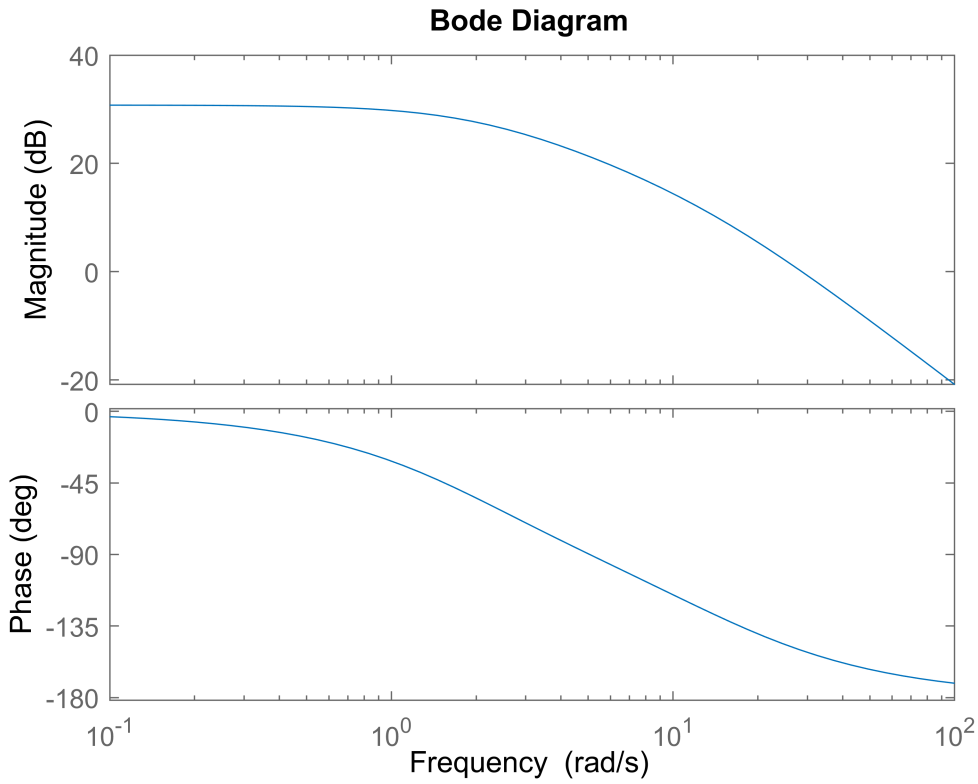
```
k_avanco = 1;
FaseGma = 70 - Pm; % margem de fase especificada 60, +10 de folga
syms alpha2
eq=(alpha2-1)/(2*sqrt(alpha2))-tand(FaseGma);
alpha2 = double(solve(eq))
```

```
alpha2 = 1.8111
```

```
% 5:
mod = -10*log10(alpha2)
```

```
mod = -2.5794
```

```
bode(gma_linha,w)
```




```
% Olhando no diagrama de Bode, wm será 12,1 rad/s.
wm = 12.1;
% 6:
tau2 = 1/(wm*sqrt(alpha2));
1/(alpha2*tau2) % freq de corte do zero do compensador
```

```
ans = 8.9911
```

```
1/(tau2) % freq de corte do polo do compensador
```

```
ans = 16.2838
```

```
g_avancos = k_avanco*(1+jw*alpha2*tau2)/(1+jw*tau2);
%-----------------------------------------------------------------------------------------------
% Calculo do erro em regime (s -> 0)
g_avancos = subs(g_avancos,jw,s);
% Entrada degrau
sEs = 1/(1+g_avancos*g_atrasos*gps);
ess(2) = double(limit(sEs,s,0,'right'));
% Disturbio degrau
sEd = gps/(1+g_avancos*g_atrasos*gps);
essd(2) = double(limit(sEd,s,0,'right'));
%-----------------------------------------------------------------------------------------------
[num,den]=numden(g_avancos);
g_avanco = tf(sym2poly(num),sym2poly(den))
```

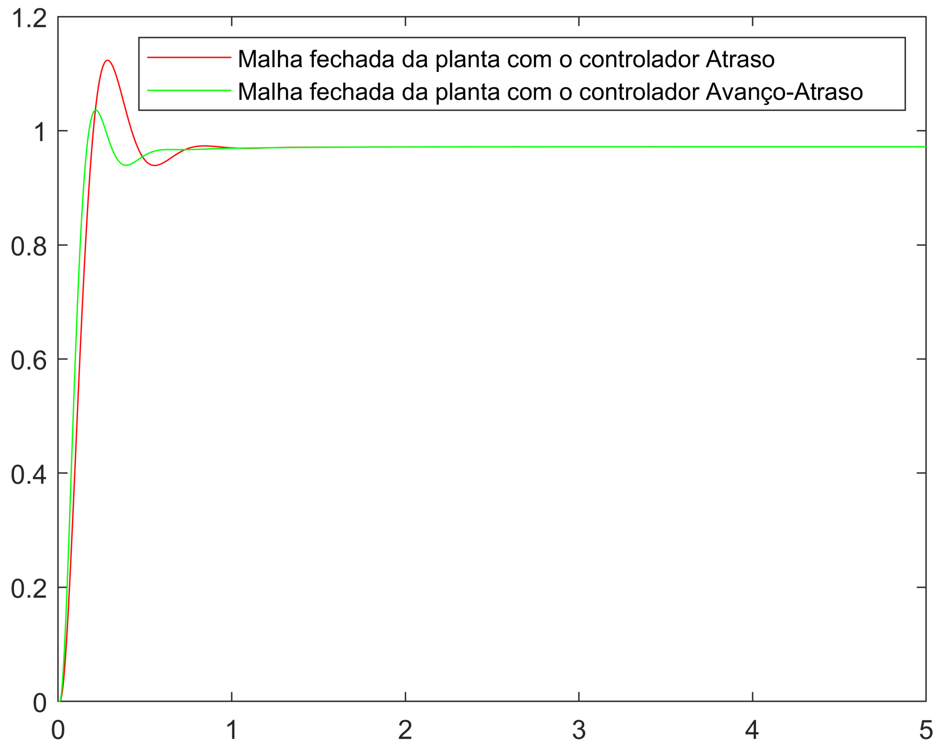
```
g_avanco =

  1.805e31 s + 1.623e32
```

```
  ---------------------
  9.964e30 s + 1.623e32

Continuous-time transfer function.
```

```
gmad = g_avanco * g_atraso * gpd;
[Gm,Pm,Wgm,Wpm]=margin(gmad)
```

```
Gm = 8.2621
Pm = 62.1815
Wgm = 48.0240
Wpm = 12.0812
```

```
gmfd = feedback(gmad,1);
t=0:.001:5;
y2 = 0;
y2 = step(gmfd,t);
%-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
% Plots
figure
plot(t,y1,'r',t,y2,'g')
legend('Malha fechada da planta com o controlador Atraso','Malha fechada da planta com o contro
```




```
MG=0;MF=0;LB=0;OS=0;RT=0;ST=0;
[Gm,Pm,Wgm,Wpm]=margin(gmad_linha);
MG(1) = Gm;
MF(1) = Pm;
LB(1) = bandwidth(gmfd_linha);
OS(1)=stepinfo(gmfd_linha).Overshoot;
RT(1)=stepinfo(gmfd_linha).RiseTime;
```

```
ST(1)=stepinfo(gmfd_linha).SettlingTime;

[Gm,Pm,Wgm,Wpm]=margin(gmad);
MG(2) = Gm;
MF(2) = Pm;
LB(2) = bandwidth(gmfd);
OS(2)=stepinfo(gmfd).Overshoot;
RT(2)=stepinfo(gmfd).RiseTime;
ST(2)=stepinfo(gmfd).SettlingTime;

% Reorganização de dados
col_atraso = [OS(1) RT(1) ess(1) essd(1) MG(1) MF(1) LB(1)]';
col_avancoatraso = [OS(2) RT(2) ess(2) essd(2) MG(2) MF(2) LB(2)]';
Tabela1=table(col_atraso,col_avancoatraso,'RowNames',{'Sobressinal (%)','T. Subida (s)','Erro à
```

Tabela1 = 7×2 table

	Gp*Atraso	Gp*Avanço-Atraso
1 Sobressinal (%)	15.6076	6.6008
2 T. Subida (s)	0.1256	0.1037
3 Erro à entrada degrau	0.0281	0.0281
4 Erro ao disturbio de degrau	0.9719	0.9719
5 MG	9.8002	8.2621
6 MF	53.2299	62.1815
7 LB	16.6621	21.2910

2.5 Compare as respostas à entrada degrau (sobressinal, tempo de subida) e o erro em regime às entradas degrau para os dois sistemas com os controladores obtidos nos itens 1.1 e 1.4 para T=0.0 s.

```
ctrl = {'PID';'Avanço-Atraso'} ;
tab1 = table(up5',tr5','VariableNames',{'Overshoot','Rise Time'},RowNames=ctrl)
```

tab1 = 2×2 table

	Overshoot	Rise Time
1 PID	0	0
2 Avanço-Atraso	11.3874	0.0302