# Laboratório 7

SEL0359 - Controle Digital - 2° Semestre de 2024

Prof. Marcos R. Fernandes

**Projeto de controladores na frequência - parte 2**

**Entrega:**[1] entregue um arquivo PDF com os gráficos e código fonte utilizado para cada questão além da resposta para as perguntas dos enunciados. Não esqueça de colocar título na figura para identificar o que cada figura representa, nome dos eixos, e legenda.

O objetivo dessa aula de laboratório é explorar recursos computacionais do Matlab para projeto de controladores na frequência para sistemas em tempo discreto.

**Dica: Confira sempre o help do matlab para cada comando antes de utiliza-lo!**

- dcgain: calcula nível dc do sistema.
- lsim: simula saída de um sistema LIT para entradas arbitrárias;
- pole: obtém polos da FT
- rlocus: obtém lugar das raízes
- stepinfo: calcula características temporais da resposta ao degrau (sobressinal, tempo de acomodação, tempo de subida, etc)

## 1 Atividade 1

Nessa atividade, o objetivo é avaliar o efeito da saturação do sinal de controle.

Para isso, considere um sistema de primeira ordem dado por

$$M\frac{dy}{dt} = -y + u$$

cuja função de transferência é

$$G(s) = \frac{1}{Ms+1}$$

Suponha que o sinal de controle seja limitado na forma

$$u_{\text{min}} \leq u \leq u_{\text{max}}$$

Adote uma estrutura de controle do tipo PID na forma

$$C(z) = K_p\Big(1 + \frac{T_s}{2T_I}\frac{z+1}{z-1} + \frac{T_D}{T_s}\frac{z-1}{z}\Big)$$

[1] Ultima atualização: 11/10/2024

Considere $T_s = 0.1s$. Faça sintonia para uma entrada em degrau unitário e comparações entre o comportamento do controlador PID nos seguintes casos:

1. Sem saturação e sem anti-windup;
2. Com saturação $0 \leq u \leq 5$ e sem anti-windup;
3. Com saturação $0 \leq u \leq 5$ e anti-windup;

Use o seguinte código Matlab como exemplo para simular esse sistema com um controlador PID.

```
close all
clear all
clc
%%
s=tf('s');
M=2;
G=1/(M*s+1);
Ts=0.1;
Gd=c2d(G,Ts,'zoh');
zpk(Gd)
%%
Kp=10;
Td=0;
Ti=.2;
%%
T=20;
td=0:Ts:T; %vetor tempo discreto
dt=0.0001; %amostragem continuo
t=0:dt:T; %vetor de tempo continuo
y=zeros(size(t)); %tempo continuo
r=ones(size(t)); %tempo continuo
u=zeros(size(td)); %tempo discreto
u_out=zeros(size(td)); %tempo discreto
up=zeros(size(td)); %tempo discreto
ud=zeros(size(td)); %tempo discreto
ui=zeros(size(td)); %tempo discreto
erro=zeros(size(td)); %tempo discreto
erroI=zeros(size(td)); %tempo discreto
N=Ts/dt; %razao das amostragens continuo/discreto
kd=1; %contador do tempo discreto
u_max=5;
u_min=0;
for k=2:numel(t)
    %% simula sistema em tempo continuo
    y(k)=y(k-1)+1/M*(-y(k-1)+u_out(kd))*dt;
```

```
    %% simula controle em tempo discreto
    if (mod(k,N)==0 || k==2) && kd<numel(u)
        kd=kd+1;
        erro(kd)=r(k)-y(k);
        %% sem anti-windup
        erroI(kd)=erro(kd);
        %% com anti-windup
        % if u_out(kd-1)≥u_max || u_out(kd-1) ≤ u_min
        %     erroI(kd)=0;
        % else
        %     erroI(kd)=erro(kd);
        % end
        %% Controle PID
        up(kd)=Kp*erro(kd); %proporcional
        ud(kd)=Kp*Td/Ts*(erro(kd)-erro(kd-1)); %derivativo (euler-backward)
        ui(kd)=ui(kd-1)+Kp*Ts/Ti*erroI(kd); %integrativo (euler-backward)
        u(kd)=up(kd)+ud(kd)+ui(kd);
        %% sem saturacao
        u_out(kd)=u(kd);
        %% com saturacao
        if u_out(kd)>u_max
            u_out(kd)=u_max;
        elseif u_out(kd)<u_min
            u_out(kd)=u_min;
        end

    end

end
f = figure;
f.Position = [0 100 1200 500];
subplot(1,3,1)
plot(t,y,'LineWidth',2)
hold on
plot(t,r,'LineStyle','--','Color','black')
title('Sinal de saida')
legend('Saida','Referencia')
subplot(1,3,2)
stairs(td,u,'LineWidth',2)
hold on
stairs(td,u_out,'LineWidth',2)
legend('u','u_{out}')
title('Sinal de controle')
subplot(1,3,3)
stairs(td,up,'LineWidth',1.5)
hold on
```

```
stairs(td,ud,'LineWidth',1.5)
stairs(td,ui,'LineWidth',1.5)
legend('Proporcional','Derivativo','Integral')
title('Termos de controle')
```

## 2 Atividade 2

Nessa atividade o objetivo projetar um controlador PID através do método do Lugar Geométrico das Raízes;

Para isso, considere a seguinte função de transferência como exemplo:

$$G(s) = \frac{2}{(s+2)(s+3)} \tag{1}$$

Deseja-se que o controle garanta um erro nulo para entrada em degrau e pólos dominantes em malha-fechada com $\xi = 0.7$ e $\omega_n = 2.5rad/s$ com tempo de amostragem $T_s = 0.3s$.

Assim, no plano-s, os pólos dominantes que atendem aos critérios de desempenho são

$$s_{1,2} = -\xi\omega_n \pm \omega_n\sqrt{1-\xi^2} = -1.75 \pm j1.7854$$

Logo, no plano-z, os pólos dominantes são

$$z_{1,2} = e^{s_{1,2}T_s} = 0.5824 \pm j0.2787$$

O equivalente discreto da planta, usando método de discretização degrau-invariante, é

$$G(z) = \mathcal{Z}\{\frac{1-e^{sT_s}}{s}G(s)\} = \frac{0.055568(z+0.6061)}{(z-0.5488)(z-0.4066)}$$

Adotando uma estrutura de controle do tipo PID, tem-se

$$C(z) = K_p\Big(1 + \frac{T_s}{2T_I}\frac{z+1}{z-1} + \frac{T_D}{T_s}\frac{z-1}{z}\Big)$$

em que foi adotado o método de Tustin para o integrador e o método Euler-Backward para o derivador. De forma equivalente, a função de transferência do PID pode ser colocada na forma

$$C(z) = \frac{K(z-c_1)(z-c_2)}{z(z-1)}$$

em que $c_1$ e $c_2$ são os zeros do controlador PID e $K$ o ganho total do controlador.

Uma vez obtido os valores de $c_1$ e $c_2$ juntamente com o ganho $K$, os parâmetros do controlador PID na forma usual são dados por

$$K_p = \frac{K}{2}(c_1 + c_2 - 3c_1c_2 + 1)$$
$$T_I = \frac{T_s}{2}\frac{1 + c_1 + c_2 - 3c_1c_2}{1 + c_1c_2 - c_1 - c_2}$$
$$T_D = 2T_s\frac{c_1c_2}{c_1 + c_2 - 3c_1c_2 + 1}$$

Logo, a sintonia do controlador PID pode ser realizada através da escolha da posição dos zeros do controlador e do ganho $K$. Assim, o PID pode ser sintonizado de forma analítica através do método do Lugar Geométrico das Raízes (LGR)!

Por simplicidade, pode-se adotar um dos zeros do controlador PID de forma a cancelar um pólo estável da planta. Escolhendo $c_1 = 0.5488$ fica faltando encontrar os valores de $c_2$ e do ganho $K$.

Note que a função de transferência do ramo direto é

$$C(z)G(z) = \frac{K(z - c_2)}{z(z-1)}\frac{0.055568(z + 0.6061)}{(z - 0.4066)}$$

Para obter o valor de $c_2$, aplica-se a condição de fase do LGR:

$$\angle C(z)G(z) = \angle(z_1 - c_2) + \angle(z_1 + 0.6061) - \angle(z_1 - 0) - \angle(z_1 - 1) - \angle(z_1 - 0.4066) = 180^\circ$$

Isolando $c_2$, resulta em

$$c_2 = 0.2995$$

Por fim, para encontrar o ganho $K$, aplica-se a condição de módulo do LGR:

$$|C(z)G(z)| = \left|\frac{K(z - 0.2995)}{z(z-1)}\frac{0.055568(z + 0.6061)}{(z - 0.4066)}\right| = 1$$

Isolando $K$, resulta em

$$K = 4.6116$$

Portanto, o controlador PID projetado é

$$C(z) = 4.6116\frac{(z - 0.5488)(z - 0.2995)}{z(z-1)}$$

e os ganhos do PID usual são

$$Kp = 3.1248$$
$$Ti = 0.6431$$
$$Td = 0.0728$$

O seguinte código em Matlab obtém os cálculos apresentados acima e simula uma resposta ao degrau para o sistema em malha-fechada.

```
close all
clear all
clc
%%
s=tf('s');
G=2/((s+2)*(s+3));
Ts=0.3;
Gd=c2d(G,Ts,'zoh');
zpk(Gd)
xi=0.7;
wn=2.5;
s1=-xi*wn+1j*wn*sqrt(1-xi^2)
z1=exp(s1*Ts)
c1=0.5488
%% condicao de fase
P=angle(z1+0.6061)-angle(z1)-angle(z1-1)-angle(z1-0.4066);
%angle(z1-c2)=pi-P
c2=real(z1)-imag(z1)/tan(pi-P)
%% condicao de modulo
z=tf('z',Ts);
Ctemp=(z-c1)*(z-c2)/(z*(z-1));
K=1/abs(evalfr(Ctemp*Gd,z1))
%%
Cd=K*Ctemp;
Gf=Cd*Gd/(1+Cd*Gd)
p=pole(Gf);
figure
plot(real(p),imag(p),'xr','LineWidth',2)
hold on
zgrid(xi,wn,Ts)
axis equal
%%
Kp=K/2*(c1+c2-3*c1*c2+1)
Ti=Ts/2*(1+c1+c2-3*c1*c2)/(1+c1*c2-c1-c2)
Td=2*Ts*c1*c2/(c1+c2-3*c1*c2+1)
%%
figure
step(Gf)
```

Seguindo a mesma lógica, projete um controlador PID para

$$G(s) = \frac{s+2}{(s+4)(s+1)}$$

com $T_s = 0.2s$ que garanta $\xi = 0.65$ e $\omega_n = 3rad/s$.

# 3 Atividade 3

Nessa atividade, o objetivo é usar os recursos computacionais do Matlab para projetar um controlador para um servo-motor.

Para isso, considere um servo-motor cuja dinâmica em tempo contínuo é descrita pela EDO:

$$J\ddot{\theta} + (b + \frac{K_t K_b}{R})\dot{\theta} = K_t K_a \frac{u}{R}$$

em que

$$\begin{cases} J \rightarrow \text{inercia do rotor} \\ b \rightarrow \text{coef. de viscosidade} \\ R \rightarrow \text{resistência de armadura} \\ K_a \rightarrow \text{ganho do módulo de potência} \\ K_b \rightarrow \text{ganho de tensão induzida (eletromotriz)} \\ K_t \rightarrow \text{constante de torque do motor} \\ u \rightarrow \text{tensão de entrada} \\ \theta \rightarrow \text{posição angular do rotor} \end{cases}$$

Deseja-se projetar um controlador de tal forma que o sistema em malha fechada apresente sobressinal máximo de $16.3\%$ ($M_p \leq 16.3\%$) e tempo de pico $t_p = 1s$.

Considere

$$\begin{cases} J = 5.3 * 10^{-7} \\ b = 7.7 * 10^{-6} \\ R = 2.6\Omega \\ K_a = 1 \\ K_b = 7.67 * 10^{-3} \\ K_t = 7.67 * 10^{-3} \end{cases}$$

1. Determine um tempo de amostragem 10x menor que o tempo de oscilação amortecida ($T_d = \frac{1}{\omega_d}$) do sistema em malha fechada.

2. Encontre os pólos dominantes de 2° que atendem os critérios de desempenho no plano-z;

3. Obtenha a função de transferência equivalente em tempo discreto para esse servo-motor considerando a presença de um ZOH.

4. Escolha uma estrutura de controlador, ex: PD, PI, PID, Avanço/Atraso de fase. Justifique sua escolha;
5. Faça a sintonia do controlador escolhido no item anterior de forma a atender os critérios de desempenho. (Dica: use LGR)
6. Valide o projeto do controlador através de testes via simulação de diferentes níveis de entrada em degrau sequencial (dica: use o comando **gensig** do matlab).

# 4 Atividade 4

Nessa atividade o objetivo é fazer a sintonia de um controlador PID usando algoritmo de otimização Particle Swarm (PSO).

Para isso, considere o sistema dado por

$$G(s) = \frac{1}{Ms + 1}$$

Adotando uma estrutura de controle do tipo PID na forma

$$C(z) = K_p\Big(1 + \frac{T_s}{2T_I}\frac{z+1}{z-1} + \frac{T_D}{T_s}\frac{z-1}{z}\Big)$$

busca-se encontrar os valores dos parâmetros $K_p, T_D$ e $T_I$ de forma que o controle PID atenda os seguintes critérios de desempenho:

1. $M_p \leq 15\%$
2. $\min \int_0^T |e(\tau)| d\tau$

Para utilizar o algoritmo PSO, deve-se definir uma função custo $J$ que atenda os critérios de projeto.

O Algoritmo PSO, de forma simplificada, consiste em criar um conjunto de soluções candidatas e então "rastrear" as melhores soluções até atingir convergência.

Para cada iteração, atualiza-se as partículas da seguinte forma

$$p^{i+1} = p^i + v^i \tag{2}$$

$$v^{i+1} = \alpha v^i + \underbrace{\beta_1 r_1 (p^i_{\text{best}} - p^i)}_{\text{aprendizado individual}} + \underbrace{\beta_2 r_2 (g_{\text{best}} - p^i)}_{\text{aprendizado do grupo}} \tag{3}$$

em que $\alpha, \beta_1, \beta_2$ são ajustes do algoritmo PSO e $r_1 \sim \mathcal{U}[0,1], r_2 \sim \mathcal{U}[0,1]$ são variáveis aleatórias uniforme que aumentam a variabilidade da busca por novas soluções.

$$p^i = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix}, \quad v^i = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix};$$

são os vetores de parâmetros e respectiva velocidades para cada partícula.

No caso de um controlador PID, o vetor de parâmetros é

$$p^i = \begin{bmatrix} K_p \\ T_D \\ T_I \end{bmatrix}$$

Para cada parâmetro é importante definir a faixa de valores a serem explorados:

$$\begin{aligned} K_p^{\text{min}} &\leq K_p \leq K_p^{\text{max}} \\ T_D^{\text{min}} &\leq T_D \leq T_D^{\text{max}} \\ T_I^{\text{min}} &\leq T_I \leq T_I^{\text{max}} \end{aligned}$$

O código em Matlab a seguir implementa esse algoritmo PSO para sintonia de um controlador PID.

```
close all
clear all
clc
%%
Ts=0.2;
s=tf('s');
T=2;
G=1/((s+3)*(T*s+1));
Gd=c2d(G,Ts,'zoh');
Mp_max=0.15;
ts_max=10;
tp_max=2;
td=0:Ts:30;
function cost=J(p,Gd,Ts,Mp_max,ts_max,tp_max,td)
Kp=p(1);
Td=p(2);
Ti=p(3);
z=tf('z',Ts);
Cd=Kp*(1+Ts/(2*Ti)*(z+1)/(z-1)+Td/Ts*(z-1)/z);
Gf=Cd*Gd/(1+Cd*Gd);
```

```
poles=pole(Gf);
if max(abs(poles))≤1
info=stepinfo(Gf);
ts=info.SettlingTime;
tp=info.PeakTime;
Mp=info.Overshoot;
y=step(Gd,td);

err=1-y;
c1=10;
c2=100;

if Mp/100>Mp_max
    cost=inf;
else
    cost=c1*(Mp_max-Mp/100)^2+c2*sum(abs(err));
end
else
    cost = inf;
end
end
%%
Kp_min=0;
Kp_max=15;
Td_min=1;
Td_max=2;
Ti_min=0;
Ti_max=2;
%%
N=30;
p(1,:)=Kp_min+(Kp_max-Kp_min)*rand(1,N); %particulas (Kp)
p(2,:)=Td_min+(Td_max-Td_min)*rand(1,N); %particulas (Td)
p(3,:)=Ti_min+(Ti_max-Ti_min)*rand(1,N); %particulas (Ti)
p_best=p; % aprendizado individual
v=2*randn(3,N); %velocidades

%parametros do PSO
alpha=0.5; %inercia
beta1=0.1; %aprendizado individual
beta2=0.3; %aprendizado do grupo
iter=50;
c_best=zeros(iter,1);

f = figure;
f.Position = [0 100 1200 500];
for j=1:iter
%avalia desempenho de todas as particulas
```

```
c=zeros(N,1);
for i=1:N
    c(i)=J(p(:,i),Gd,Ts,Mp_max,ts_max,tp_max,td);
    if c(i)< J(p_best(:,i),Gd,Ts,Mp_max,ts_max,tp_max,td)
        p_best(:,i)=p(:,i);
    end
end
[c_best(j),best]=min(c);
p_global=p(:,best); %melhor solucao global
%%
subplot(1,3,1)
plot(c_best(1:j),'o-','LineWidth',1.5)
xlabel('iter')
ylabel('J(p)')
grid on
title(['iter' num2str(j)])
subplot(1,3,2)
hold off
plot3(p(1,:),p(2,:),p(3,:),'or','LineWidth',3);
hold on
quiver3(p(1,:),p(2,:),p(3,:),v(1,:),v(2,:),v(3,:),'g','LineWidth',1);
xlabel('Kp')
ylabel('Td')
zlabel('Ti')
xlim([Kp_min Kp_max])
ylim([Td_min Td_max])
zlim([Ti_min Td_max])
grid on
subplot(1,3,3)
hold off
z=tf('z',Ts);
Kp=p_global(1);
Td=p_global(2);
Ti=p_global(3);
Cd=Kp*(1+Ts/(2*Ti)*(z+1)/(z-1)+Td/Ts*(z-1)/z);
Gf=Cd*Gd/(1+Cd*Gd);
y=step(Gf,td);
stairs(td,y,'LineWidth',1.5)
hold on
line([0 max(td)], [1 1]+Mp_max,'linestyle','--','color','black')
line([ts_max ts_max], [0 1+Mp_max],'linestyle','--','color','black')
drawnow
%%
for i=1:N
    p(:,i)=p(:,i)+v(:,i);
    r1=rand;
    r2=rand;
```

```
        v(:,i)=alpha*v(:,i)+beta1*r1*(p_best(:,i)-p(:,i))+beta2*r2*(p_global-p(:,i));
        %% garante intervalos
        if p(1,i)<Kp_min
            p(1,i)=Kp_min;
        end
        if p(1,i)>Kp_max
            p(1,i)=Kp_max;
        end
        if p(2,i)<Td_min
            p(2,i)=Td_min;
        end
        if p(2,i)>Td_max
            p(2,i)=Td_max;
        end
         if p(3,i)<Ti_min
            p(3,i)=Ti_min;
        end
        if p(3,i)>Ti_max
            p(3,i)=Ti_max;
        end
    end

end
```

Tente reproduzir o resultado acima e aplicar em algum exemplo já estudado de sintonia de controladores.