PMR3409 – CONTROLE II

EXPERIÊNCIA 6
PROJETO DE CONTROLE DE POSIÇÃO PARA PLANTA INDUSTRIAL

AUTOR: FELIPE LOPES DE SOUZA (ESTÁGIO PAE 2017)

1. INTRODUÇÃO

Os objetivos desta experiência são: traduzir requisitos de um projeto industrial em métricas usuais de controle como sobressinal, tempo de subida, tempo de assentamento e erro estático; projetar um controlador PI-D que satisfaça essas métricas, usando ferramentas computacionais; implementar esse controlador e verificar por simulação, utilizando a saída do sistema real para alimentar a simulação, se os requisitos do projeto industrial são satisfeitos.

Inicialmente você será apresentado ao problema de como projetar controladores SISO quando a planta do sistema foi identificada já incluindo a planta dos sensores, o que não é trivial.

Para justificar o emprego do controle, uma planta industrial será proposta, com requisitos próprios de como o sistema deve se comportar. A descrição do comportamento do sistema deverá ser traduzida para métricas utilizadas no projeto de controle clássico, como sobressinal e outras.

Em seguida, você deverá projetar um controlador que satisfaça os requisitos usando a ferramenta SISOTOOL do MATLAB.

Concluída a etapa de projeto, os controles obtidos deverão ser implementados e a resposta em tempo real do motor será utilizada para alimentar uma simulação da planta industrial inicialmente proposta, apresentando o conceito de Hardware-In-The-Loop (HIL) em projetos de engenharia.

2. CONTROLE SISO DA PLANTA PREVIAMENTE IDENTIFICADA

Em experiências anteriores, o sistema real, representado na Figura 1, foi modelado pelo diagrama de blocos apresentado na Figura 2.

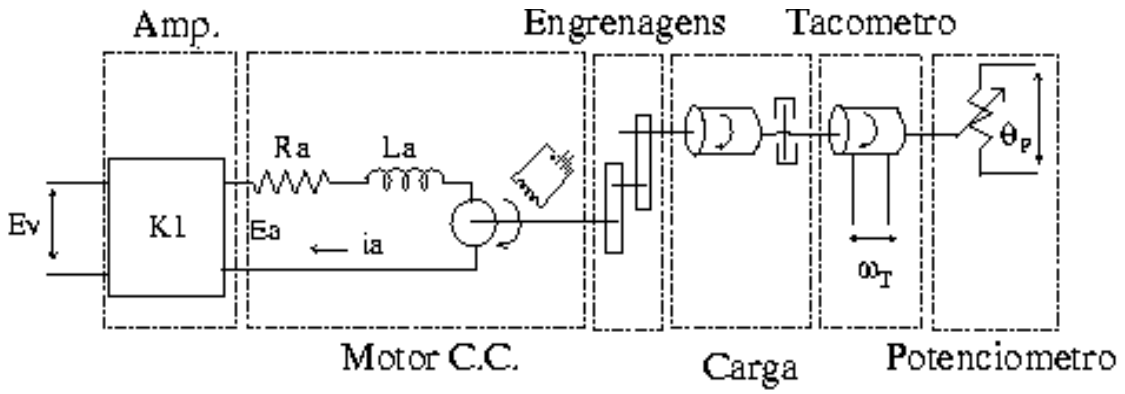

Figura 1. Diagrama esquemático do sistema real.

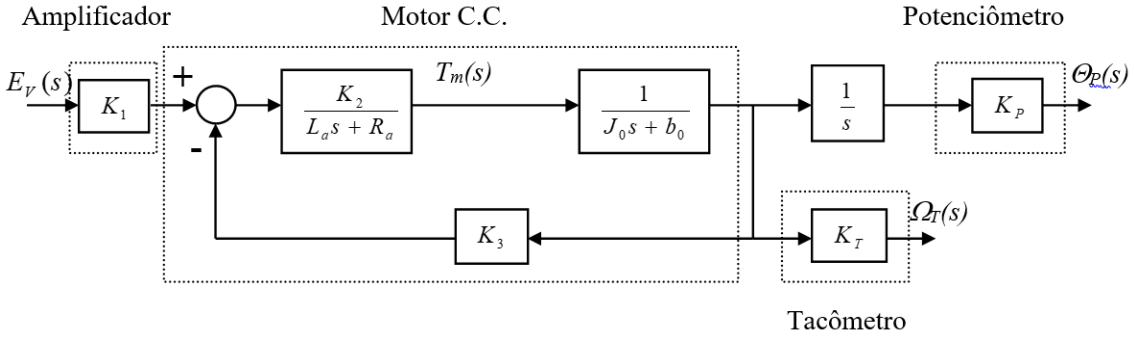

Figura 2. Diagrama de blocos do sistema.

O modelo foi, então, simplificado levando em conta que a indutância da armadura é pequena, sendo agora representado pelas Equações (1) e (2).

$$G_\omega(s) = \frac{\Omega_T(s)}{E_V(s)} = \frac{K_1 K_2 K_T}{R_a(J_0 s + b_0) + K_2 K_3} \quad (1)$$

$$G_\theta(s) = \frac{\Theta_P(s)}{E_V(s)} = \frac{K_1 K_2 K_P}{s[R_a(J_0 s + b_0) + K_2 K_3]} \quad (2)$$

Partindo-se dessas equações, o novo diagrama de blocos do sistema é o apresentado na Figura 3, em que $\theta_M(s)$ é a posição do motor e $\omega_M(s)$ a velocidade do motor.

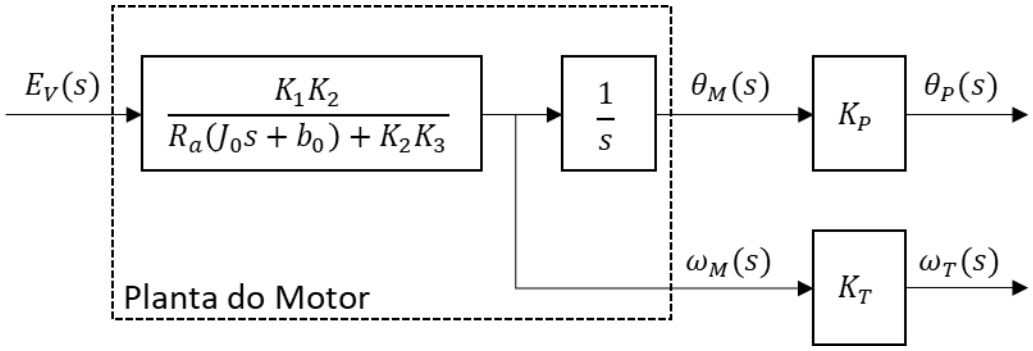

Figura 3. Planta simplificada do motor, e planta dos sensores.

É possível observar que o sistema, na forma como está, tem uma entrada, E_V, e duas saídas, θ_P e ω_T, o que exigiria, a princípio, o projeto de um controlador SIMO (Single-Input-Multiple-Output). Nesse caso, o objetivo do controle seria controlar os sinais do potenciômetro e do tacômetro através do sinal de input de voltagem.

Se o objetivo é controlar a posição do motor, entretanto, um simples pré-processamento dos sinais permite colocar o sistema novamente na forma de um controle SISO, permitindo a aplicação de técnicas clássicas, Figura 4.

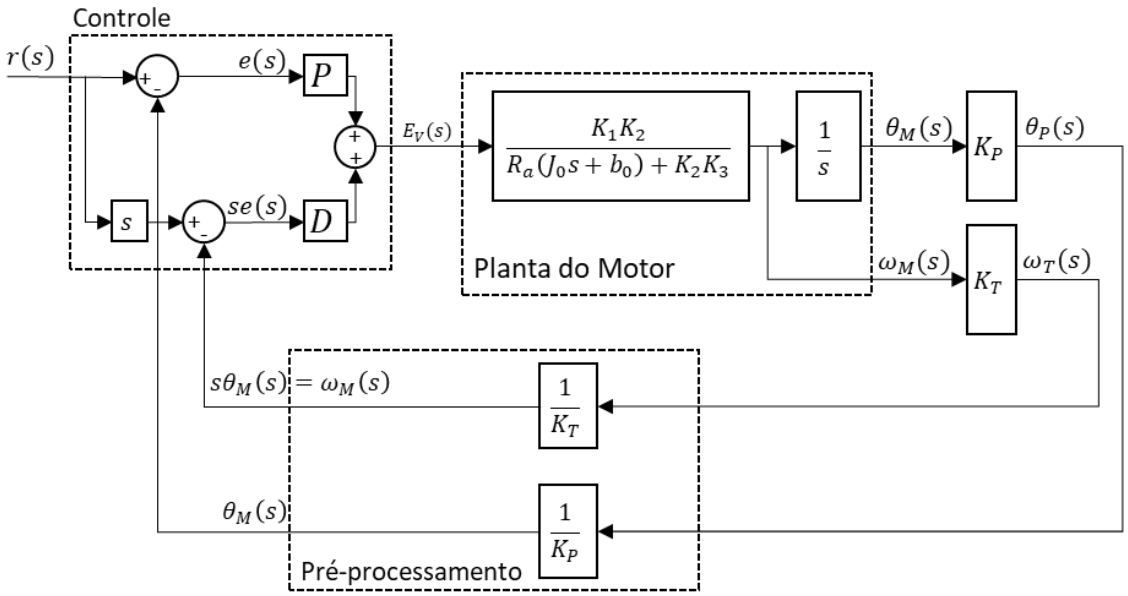

Figura 4. Pré-processamento permite obter a posição do motor e a derivada da posição, para que a informação seja usada em um controle PD.

É possível observar, nesse caso, que o input da planta E_V obedece as seguintes equações:

$$E_V(s) = Pe(s) + Dse(s) \tag{3}$$

$$e(s) = r(s) - \theta_M(s) \tag{4}$$

O que equivale ao seguinte diagrama de blocos, Figura 5:

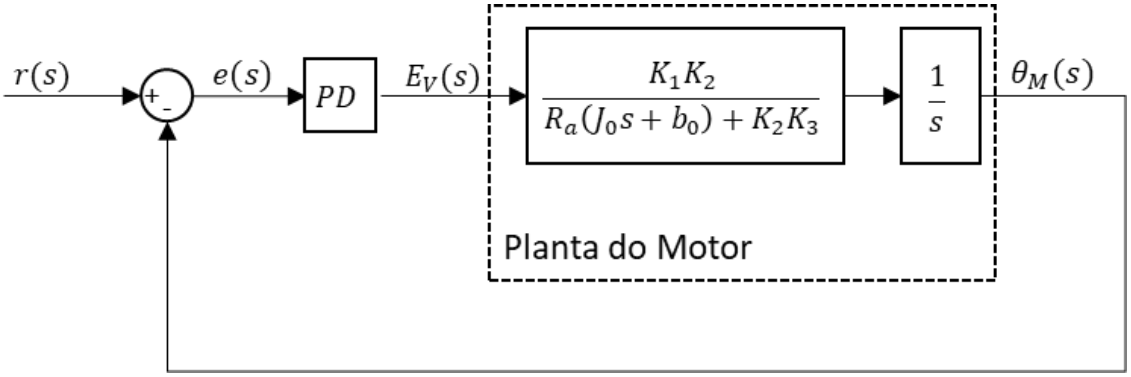

Figura 5. Rearranjo do diagrama de blocos anterior.

Ou seja, os ganhos P e D, para que o sistema atinja um determinado comportamento de θ_M dada a referência, podem ser determinados através de um projeto SISO clássico de controle proporcional-derivativo, utilizando a planta do motor de base.

O problema é que a planta do motor não é conhecida independentemente, pois a identificação foi feita incluindo os sensores na planta, obtendo-se as equações a seguir:

$$G_\omega(s) = \frac{\Omega_{\mathrm{T}}(s)}{E_V(s)} = \frac{K_\omega}{Ts + 1} \tag{5}$$

$$G_\theta(s) = \frac{\Theta_{\mathrm{P}}(s)}{E_V(s)} = \frac{K_\theta}{s(Ts + 1)} \tag{6}$$

$$K_\omega = \frac{K_2K_T}{R_ab_0 + K_2K_3}, K_\theta = \frac{K_2K_P}{R_ab_0 + K_2K_3}, T = \frac{R_aJ_0}{R_ab_0 + K_2K_3} \tag{7}$$

Ou seja, não é possível separar com certeza a parte do ganho devido a planta do motor e a parte devido aos sensores individuais.

Apesar disso, o mesmo princípio de pré-processamento utilizado anteriormente pode ser utilizado para as novas plantas, de forma que é possível obter um sinal e sua derivada, Figura 6.

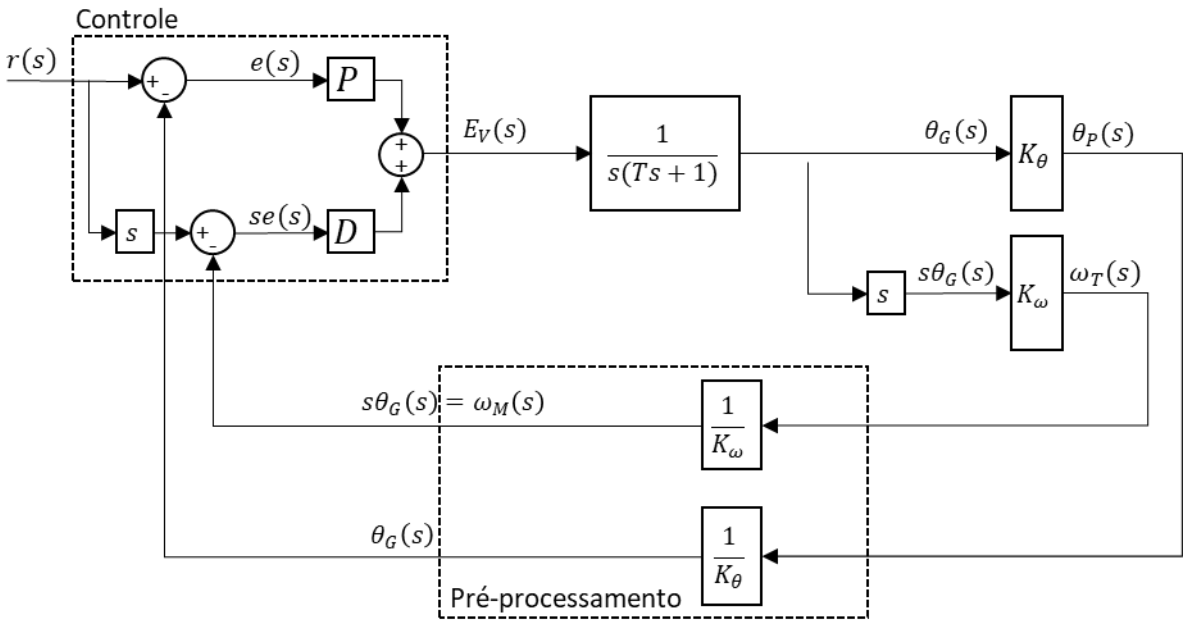

Figura 6. Pré-processamento dos sinais para obter a mesma estrutura discutida anteriormente.

Repare que, nesse caso, o sinal controlado $\theta_G(s)$ não tem um significado físico real, como a posição do motor. Apesar disso, a estrutura é exatamente a mesma que a apresentada anteriormente, e o diagrama de blocos pode ser rearranjado na forma apresentada na Figura 7.

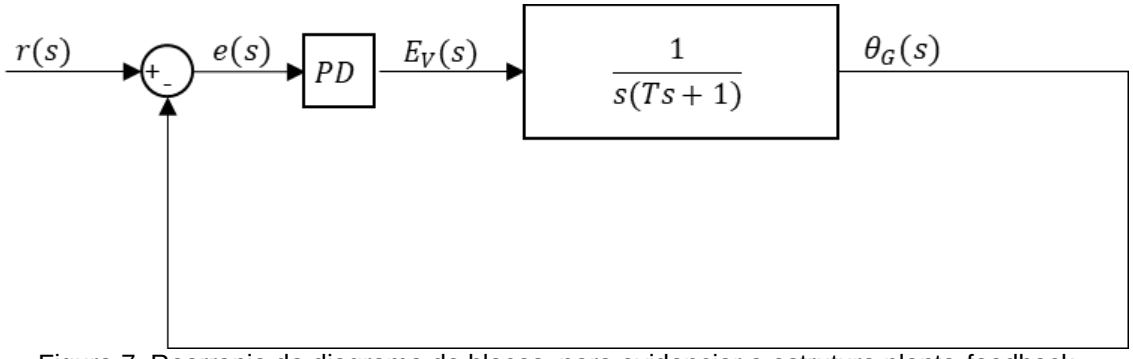

Figura 7. Rearranjo do diagrama de blocos, para evidenciar a estrutura planta-feedback-controlador.

Ou seja, pode-se determinar os ganhos P e D para que $\theta_G(s)$ siga a referência $r(s)$ com características de seguimento desejadas.

Embora o sinal $\theta_G(s)$ não represente um sinal real do sistema, ou seja, aparentemente não há vantagem em controlar tal sinal, é importante observar que ele está atrelado ao sinal de saída do potenciômetro através de uma constante:

$$\theta_P(s) = K_\theta \theta_G(s) \tag{8}$$

O que significa que o comportamento de $\theta_P(s)$ será exatamente o comportamento de $\theta_G(s)$, com diferença apenas na amplitude do sinal. Ou seja, projetar o controlador para que $\theta_G(s)$ siga $r_G(s)$ com características de seguimento desejada é equivalente a projetar um controlador para que $\theta_P(s)$ siga $K_\theta \cdot r_G(s) = r_P(s)$ com as mesmas características (mesmo sobressinal percentual, mesmo tempo de subida, mesmo tempo de assentamento, etc).

A afirmação acima pode ser facilmente provada utilizando o diagrama de blocos da Figura 8 para deduzir as Equações (9)-(12).

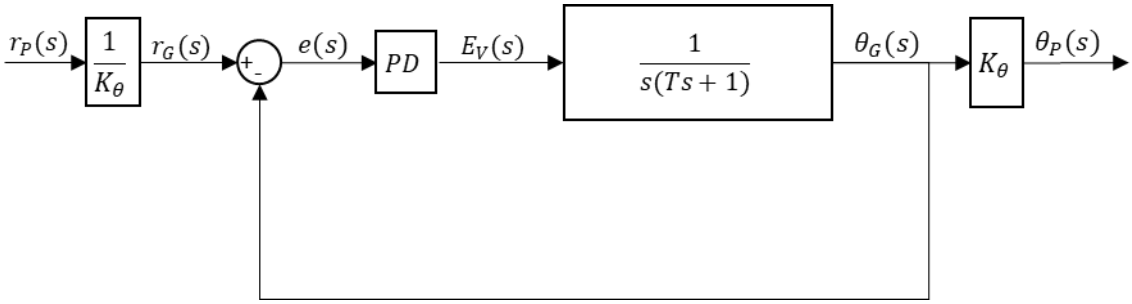

Figura 8. Diagrama de blocos evidenciando a relação entre $\theta_G(s)$ e $\theta_P(s)$.

Chamando $G = 1/s(Ts+1)$ e $F = PD$:

$$\theta_G(s) = \frac{GF}{1+GF} r_G(s) \tag{9}$$

$$\theta_P(s) = K_\theta \frac{GF}{1+GF} r_G(s) \tag{10}$$

$$\theta_P(s) = K_\theta \frac{GF}{1+GF} \frac{1}{K_\theta} r_P(s) \tag{11}$$

$$\theta_P(s) = \frac{GF}{1+GF} r_P(s) \tag{12}$$

Ou seja, para controlar a saída $\theta_P(s)$ para a referência $r_P(s)$, basta que se projete um controle para a saída $\theta_G(s)$ para a referência $r_G(s)$. Sendo assim, **durante a fase de projeto do controlador a planta usada será**:

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

Com as correções necessárias de pré-processamento (tanto da referência $r_G(s)$ como dos sinais $\theta_P(s)$ e $\omega_T(s)$) sendo feitas durante a fase de implementação do sistema.

3. PROJETO USANDO SISOTOOL

O modelo de controle proposto é da seguinte forma:

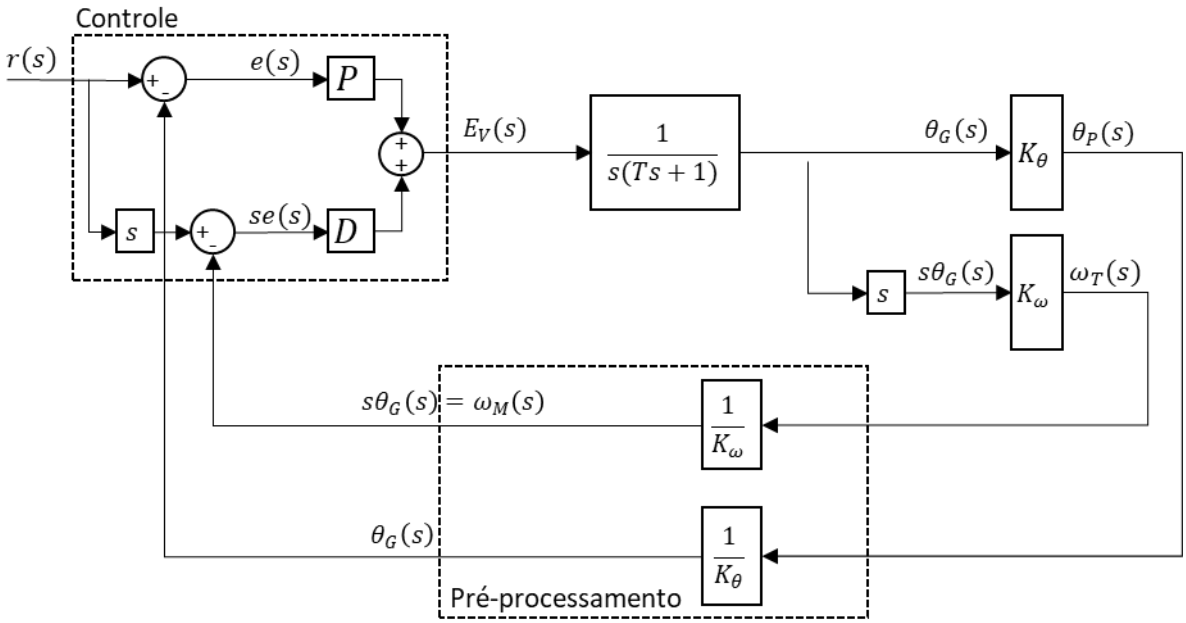

Figura 9. Controle do tipo PD.

Entretanto, sabe-se que é necessário um filtro derivativo para que a implementação real de controles PID possam ser feitas (como explicado na experiência anterior).

Como o sinal será uma referência constante, entretanto, uma outra abordagem é possível, já que $\frac{dr(t)}{dt} = 0$, quando $t \neq 0$. Isso significa que:

$$\frac{de(t)}{dt} = \frac{dr(t)}{dt} - \frac{d\theta_G(t)}{dt} = -\frac{d\theta_G(t)}{dt}, t \neq 0 \quad (13)$$

Ou seja, eu posso simplesmente multiplicar o ganho derivativo pelo oposto da saída do tacômetro corrigida, ao que se denominada controle PI-D:

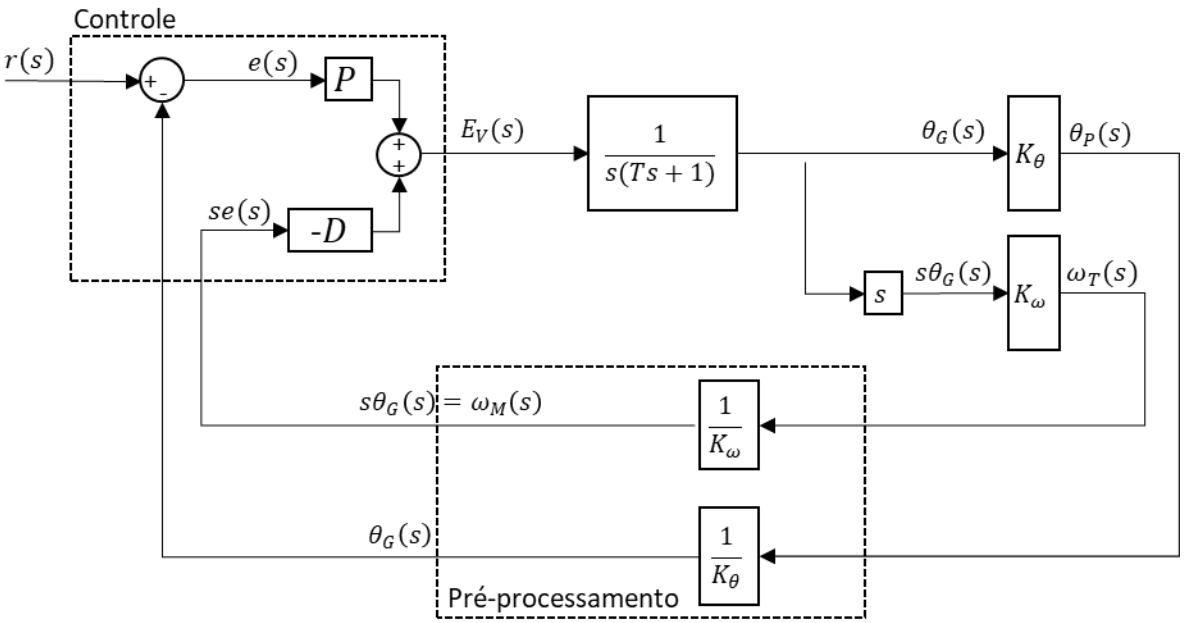

Figura 10. Controle do tipo PI-D.

O controle do tipo PI-D tem aproximadamente as mesmas características de um controle projetado do tipo PID sem filtro derivativo, entretanto, existe um ligeiro atraso na resposta (já que a derivada do degrau de input no instante zero tende a causar um ganho elevado no controlador logo no instante inicial).

Sendo assim, para o projeto do controlador para essa experiência é necessário que se considere uma outra arquitetura na plataforma SISOTOOL, que leve em conta essa diferença entre o controlador PID e o PI-D.

Rearranjando o diagrama de blocos é possível colocar em evidência como a nova arquitetura deve ser:

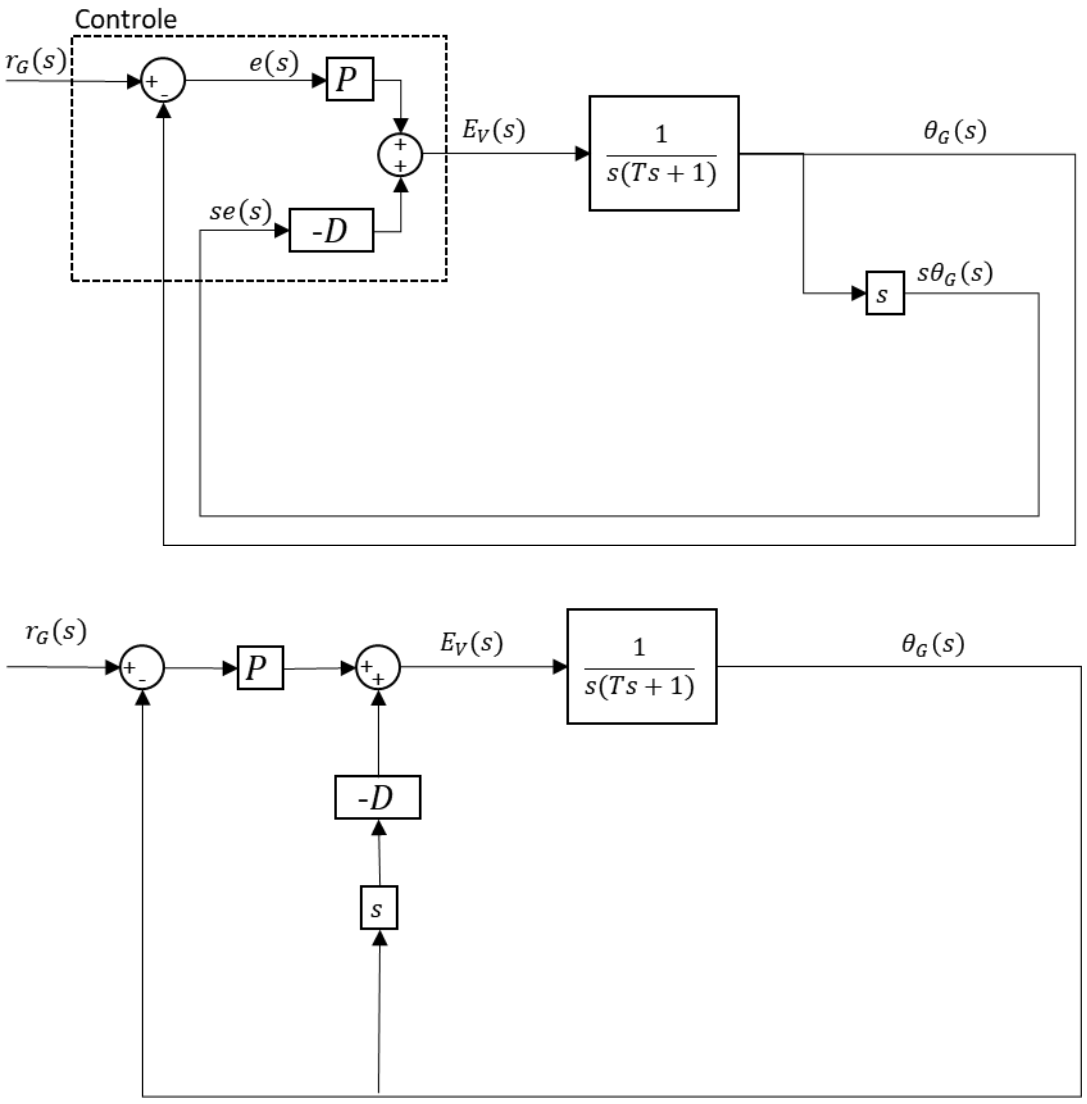

Figura 11. Rearranjando diagramas de blocos para evidenciar nova arquitetura.

Sendo assim, a seguinte arquitetura deve ser selecionada na ferramenta SISOTOOL:

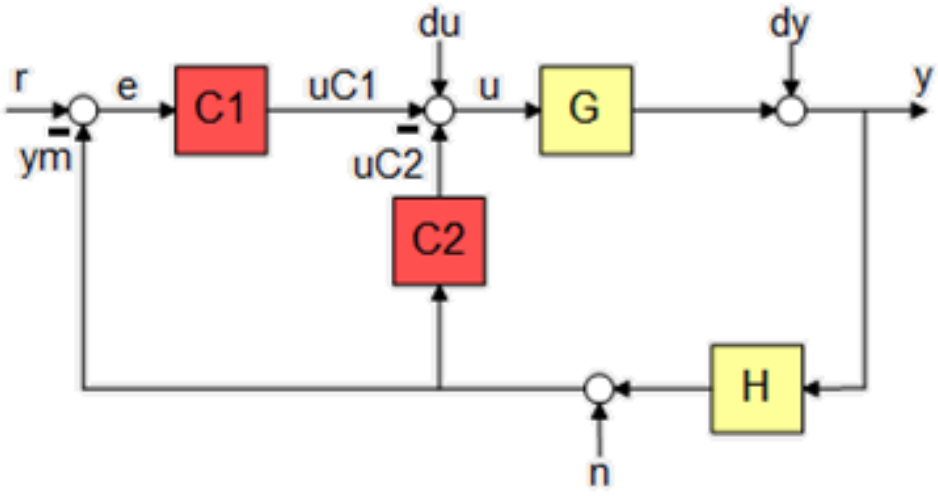

Figura 12. Arquitetura equivalente no sisotool.

Quando o controle é do tipo proporcional:

$$C_1(s) = P, C_2(s) = 0 \tag{14}$$

Quando o controlador é do tipo P-D:

$$C_1(s) = P, C_2(s) = Ds \tag{15}$$

Ou seja, um differentiator em C_2.

Quando o controlador é do tipo PI-D:

$$C_1(s) = \frac{I(s + \frac{P}{I})}{s}, C_2(s) = Ds \tag{16}$$

Ou seja, um integrator e um zero real em C_1.

Nesse caso, é possível ajustar os ganhos em dois diagramas de lugar das raízes, um para o loop C_1 e outro para o loop C_2.

Por fim, é importante enfatizar como deve ser a implementação. Que é ilustrada na figura a seguir:

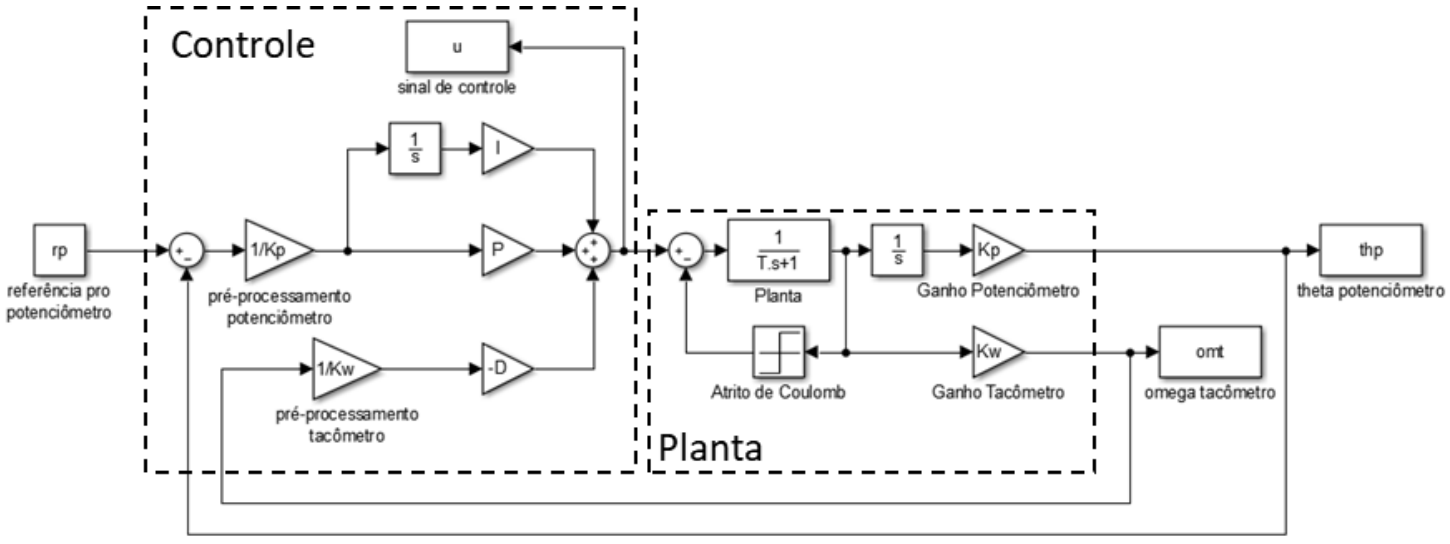

Figura 13. Implementação do controle.

Ou seja, inicialmente adquiro os sinais θ_P e ω_T, então processo os mesmos para obter $e = (\theta_P - r_P)/K_P$, $se = -\omega_T/K_\omega$; integro o sinal de erro (essencialmente somando o novo sinal de erro multiplicado por Δt, ou seja, $\int e_k = \int e_{k-1} + e \cdot \Delta t$); e, por fim, calculo $u = P \cdot e + I \cdot \int e + D \cdot se$.

4. CÁLCULOS PRELIMINARES

Para essa etapa da experiência a função base que será utilizada é a função osciloscópio.m. A função retorna 5 vetores:

1. tVec: vetor com os instantes de tempo nos quais a função controle foi executada, também servindo como referência de tempo para os outros vetores;
2. ref: vetor com referências externas ao computador (não será utilizado nessa experiência, mas pode ser utilizado quando o sinal de referência for setado por um gerado de funções, por exemplo);
3. pot: vetor com medidas do potenciômetro;
4. tac: vetor com medidas do tacômetro;
5. uctrl: vetor com valores de controle enviados ao motor.

Como input, a função requer apenas dois parâmetros

1. Fa: frequência de amostragem desejada, em Hz;
2. funControle: handle para a função controle.

Em MATLAB é possível passar uma função como argumento de outra função usando o prefixo @. A ideia é que com o mesmo algoritmo osciloscopio.m seja possível executar o controle usando diferentes configurações de controlador. Para tanto, a função de controle deve ser definida como:

```
function [uk,ik] = myFunControle(tk,tkm,rk,pk,wk,ikm)

    % INPUTS:
    % tk: instante atual da chamada da função
    % tkm: instante anterior da chamada da função (dt = tk-tkm)
    % rk: sinal de referência externa
    % pk: sinal atual do potenciômetro
    % wk: sinal atual do tacômetro
    % ikm: integral do erro no instante anterior
    %
    % OUTPUTS:
    % uk: sinal de controle no instante atual (calculado na função)
    % ik: integral do erro no instante atual (calculado na função)


end % function myFunControle
```

Para entender melhor o conceito, abra a função controlFunExample, compreenda a mesma, e em seguida rode, no command window:

```
[tVec,ref,pot,tac,uctrl] = osciloscopio(50,@controlFunExample);
```

Será possível ver, em tempo real, um controle do tipo PID atuando no motor e levando o mesmo de -3V a 3V de forma alternada (repare que a referência é setada internamente, e não externamente, de acordo com o tempo de simulação).

4.1. Validação de K_θ, K_ω e T

Usando a função anterior, é possível, inicialmente, validar as constantes do motor. Para tanto, abra a função calculaConstantesKwTKp e perceba que a mesma é simplesmente um sinal de controle constante. Em seguida, execute o comando a seguir e aguarde ao menos 1s após o sistema entrar em regime estacionário (ou seja, velocidade não mudar mais) e aperte stop:

```
[tVec,ref,pot,tac,uctrl] = osciloscopio(50,@calculaConstantesKwTKp);
```

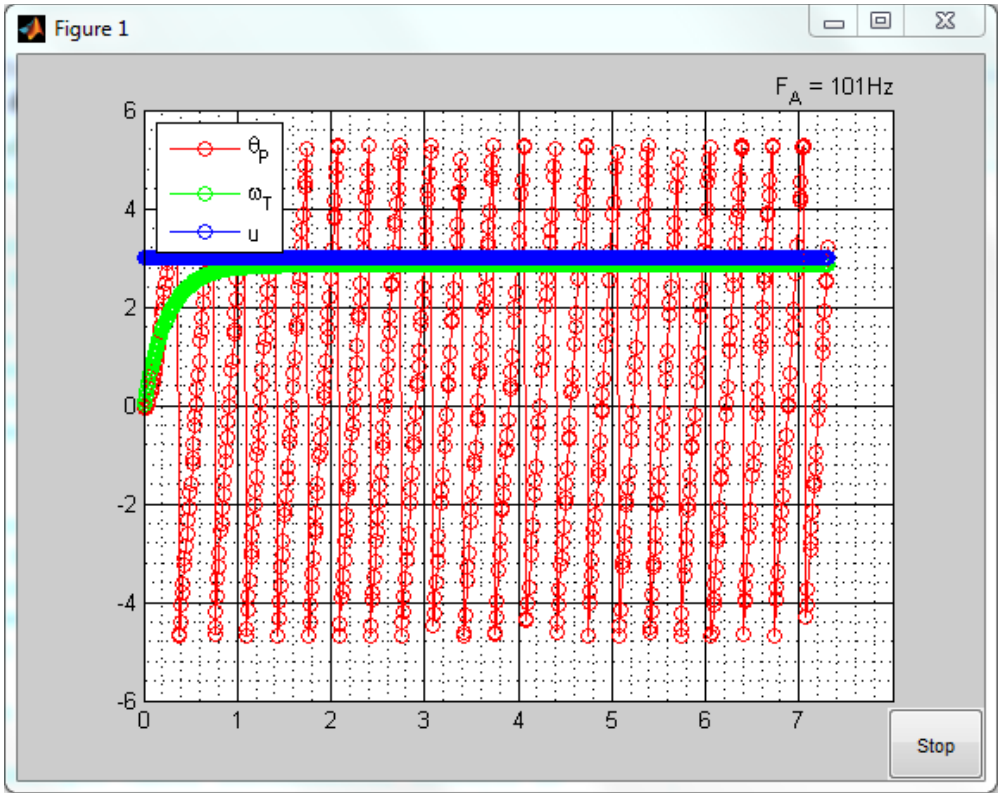

Figura 14. Resultado esperado da execução do comando.

Quando a execução é interrompida o valor em regime estacionário (média do último segundo) de cada um dos sinais é mostrado na tela:

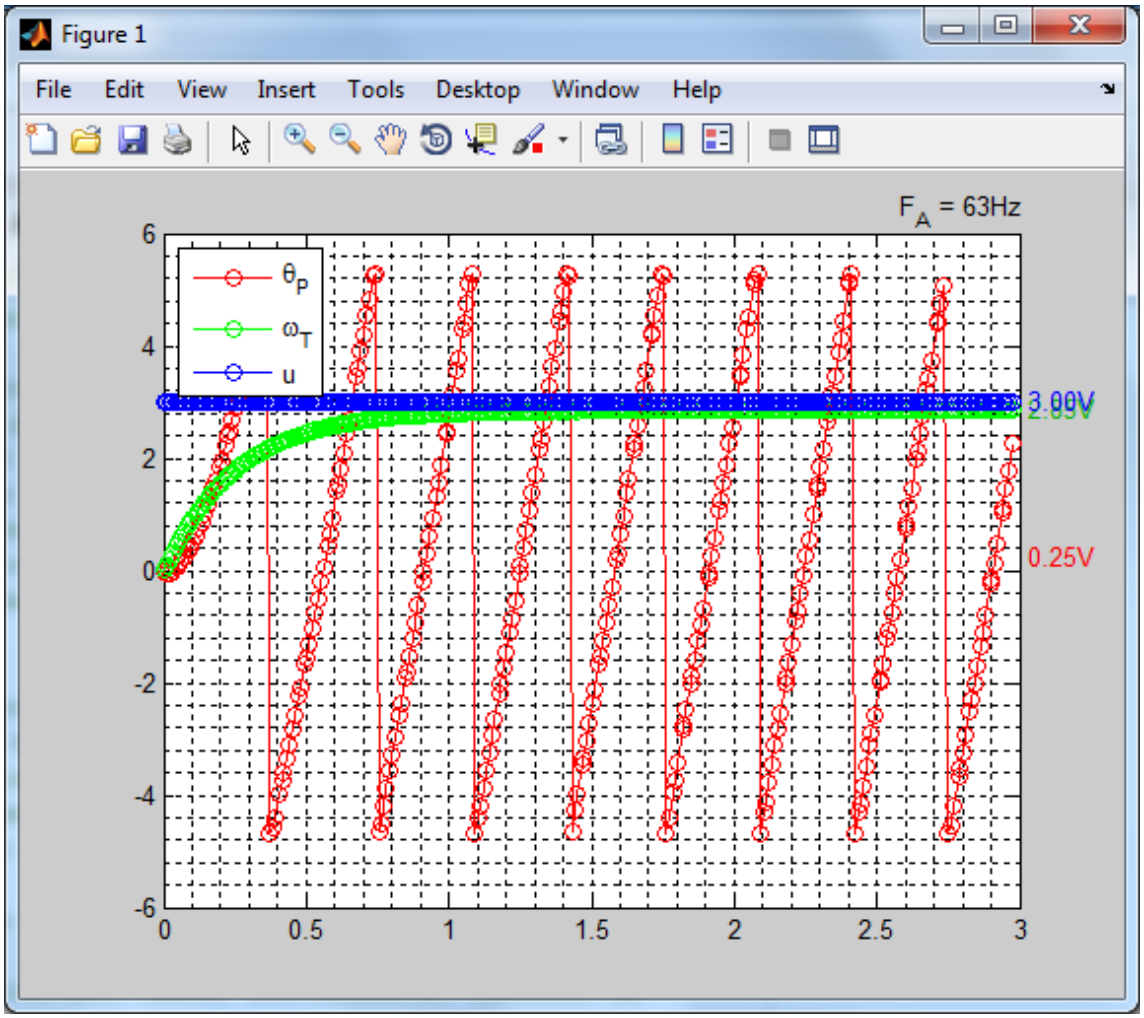

Figura 15. Quando a execução é interrompida é possível adquirir o valor em regime estacionário dos sinais.

Os primeiros valores de interesse são os valores u_∞ e ω_∞, ou seja, do exemplo:

$$u_\infty = 3V$$

$$\omega_\infty = 2.85V$$

Com esses valores a primeira constante do motor já pode ser obtida:

$$K_\omega = \frac{\omega_\infty}{u_\infty} = 0.95$$

A constante de tempo T pode ser calculada através de uma nova figura normalizada. Para isso, crie uma nova figure e plote a resposta do tacômetro normalizada pelo valor ω_∞, ou seja:

```
uinf = 3;
winf = 2.85;
figure;plot(tVec,tac/winf);
```

Através da ferramenta “Data Cursor” é possível selecionar o valor que se encontra aproximadamente em 0.623 (ou 62.3% do sinal). O instante de tempo que isso ocorre é a constante de tempo do motor:

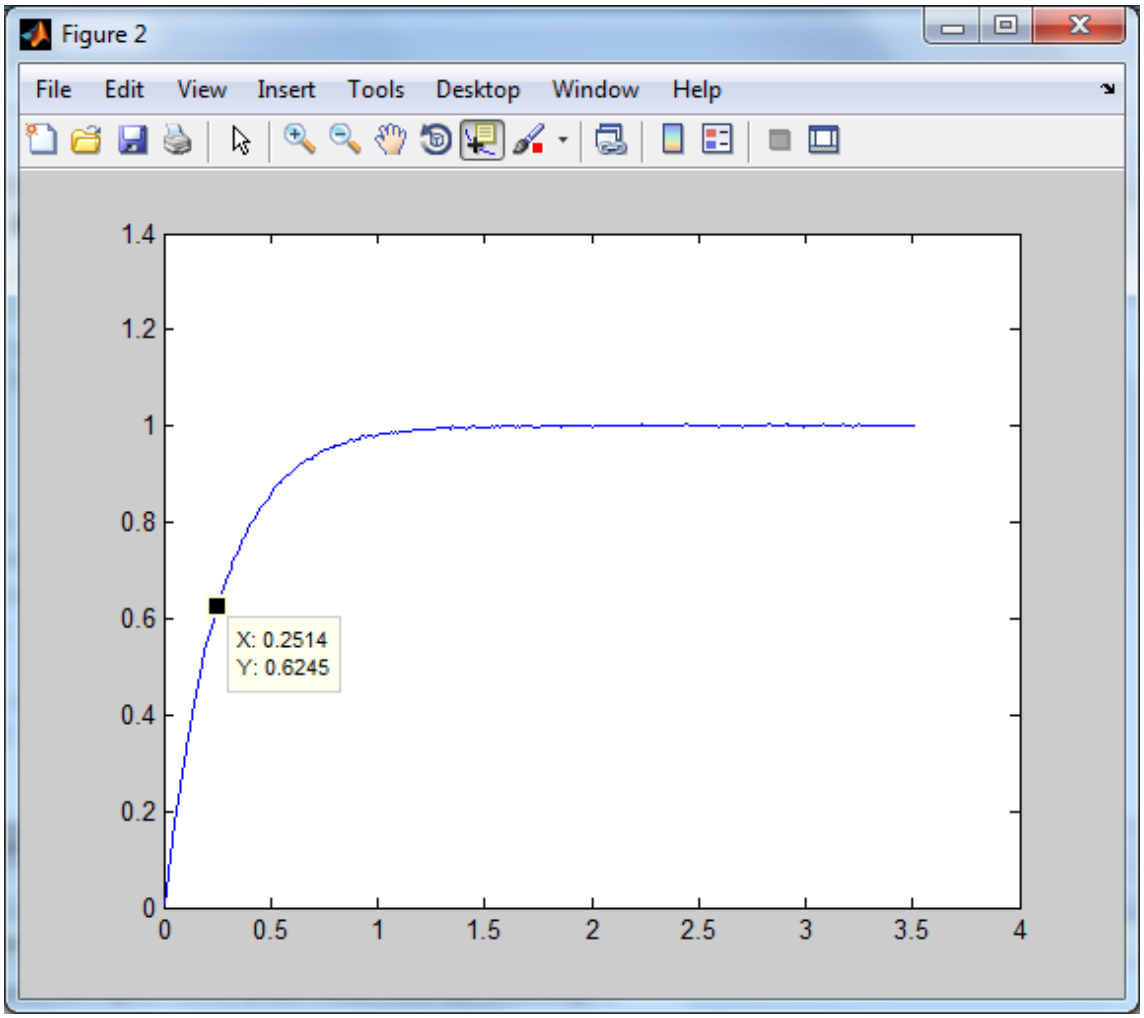

Figura 16. Determinação do parâmetro T do motor.

No exemplo, $T = 0.25s$.

Para checar se o sistema identificado até então está correto, é possível plotar a função de transferência em cima do gráfico atual:

```
T = 0.25;
s = zpk('s');
Gteo = 1/(T*s + 1);
hols on;step(Gteo);
```

Se a identificação foi feita corretamente, ambos os gráficos devem ser equivalentes:

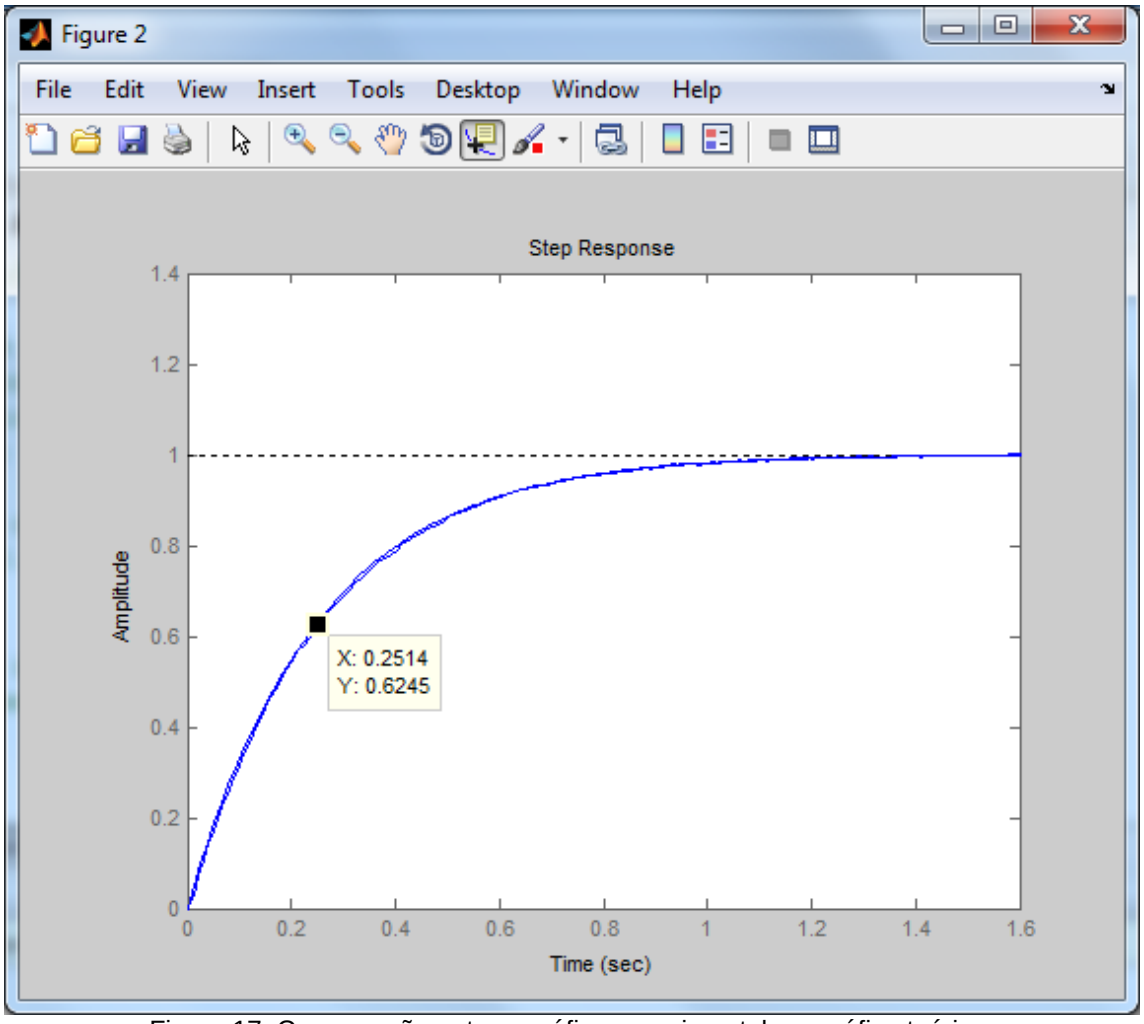

Figura 17. Comparação entre o gráfico experimental e o gráfico teórico.

Para a identificação da constante K_P do motor, atrelada ao sinal do potenciômetro, é necessário separar uma região do sinal que operou dentro do limite linear. Para tanto, é possível usar o comando “Data Cursor” e verificar os índices do vetor pot em que isso ocorre. Para criar dois “Data Cursor” basta clicar com o botão direito e selecionar “Create New Data Tip”.

```
figure;plot(pot)
```

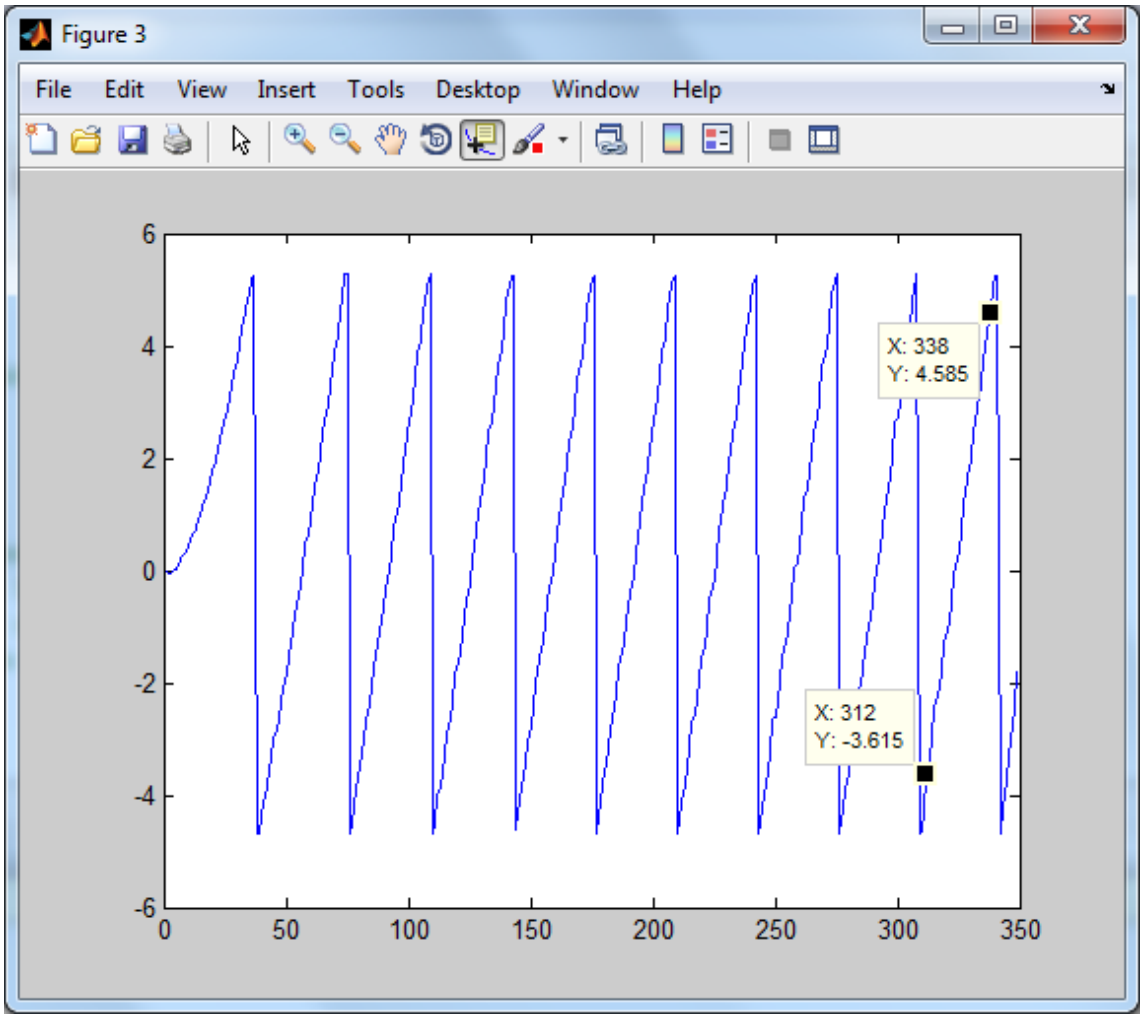

Figura 18. Repare que quando o sinal é plotado apenas com o comando pot, o eixo X contém o número do índice em que o valor ocorre.

Com os limites definidos, é possível plotar o valor do potenciômetro apenas nessa região (esse plot deve ser feito em função do tempo), usando o comando:

```
figure;plot(tVec(312:338),pot(312:338),'o-');
```

No regime estacionário, a velocidade é constante, ou seja, o sinal do potenciômetro vira uma reta com inclinação $K_P \cdot u_\infty$. Tal inclinação pode ser obtida no menu da figura, Tools > Basic Fitting > Linear (certifique-se de marcar “Show Equation”).

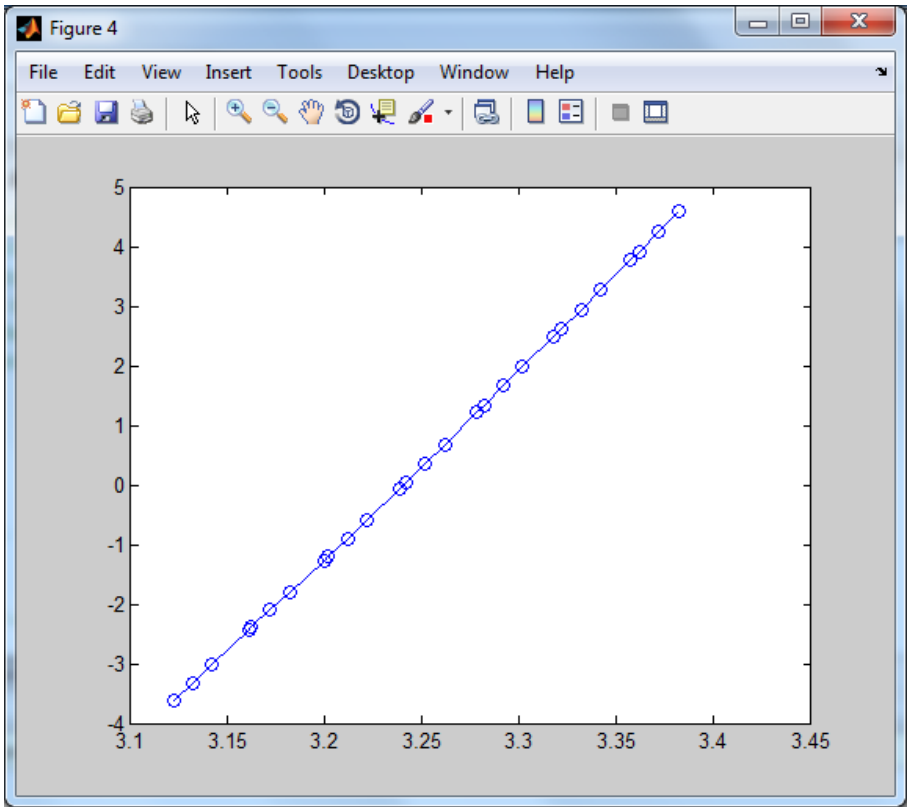

Figura 19. Plot contendo apenas a região selecionada do potenciômetro.

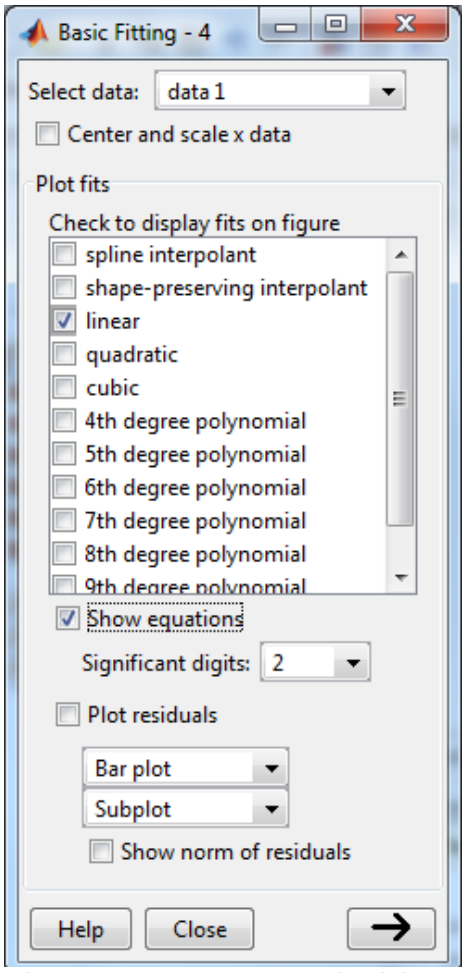

Figura 20. Ferramenta "Basic Fitting".

Com isso, é possível obter a inclinação da reta:

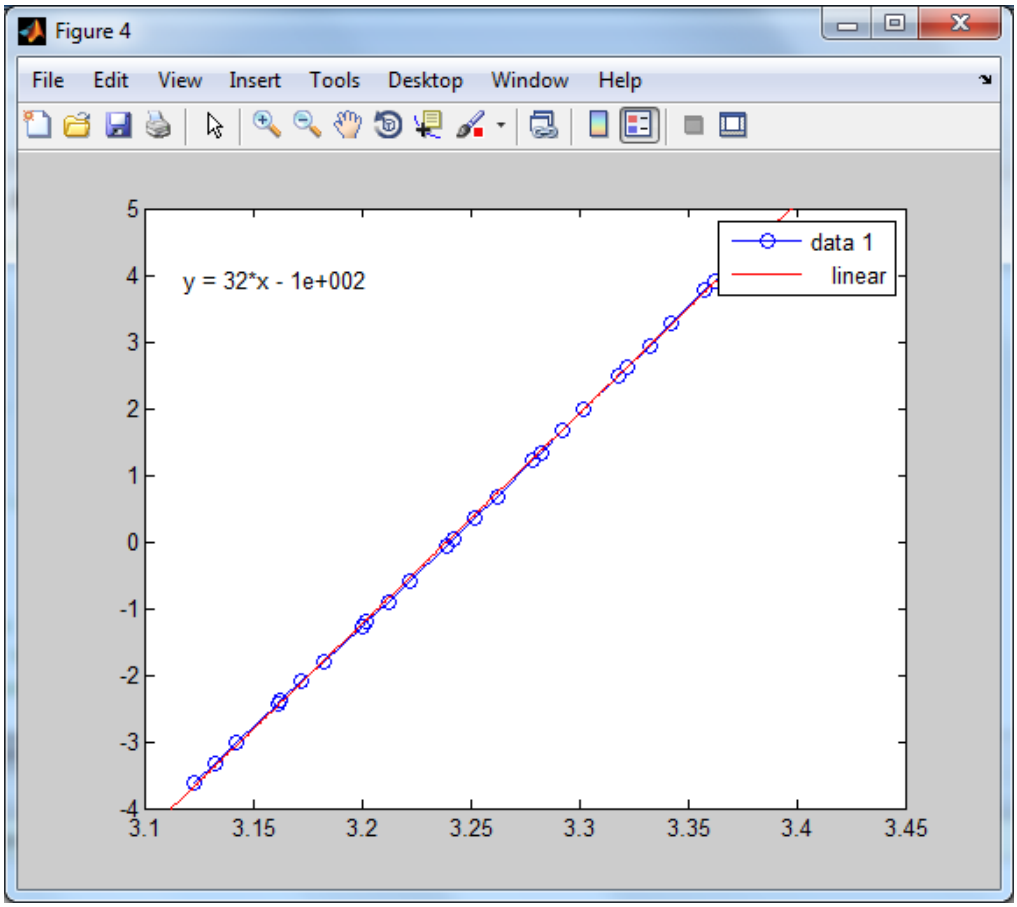

Figura 21. Inclinação do sinal do potenciômetro na região linear.

Com isso calcula-se o ganho K_P:

$$K_P = \frac{\alpha}{u_\infty} = 10.7$$

4.2. Levantamento do Atrito de Coulomb

Outra propriedade importante do motor, que até então não foi levantada, é o atrito de Coulomb.

Como explicado na experiência anterior, o atrito pode ser modelado como um distúrbio na entrada da planta. Sendo assim, uma forma simples de levantar o mesmo é usar um controle bem lento, para que não haja velocidade excessiva, e verificar qual é o valor de u para o qual o sistema para de se mexer. Para tanto, estude a função calculaAtrito.m, que contém um controle do tipo PD bem lento, e rode a mesma com a função osciloscópio até que o

sistema pare de se mover. Assim que o sistema para de se mover, é possível determinar o distúrbio du do sistema:

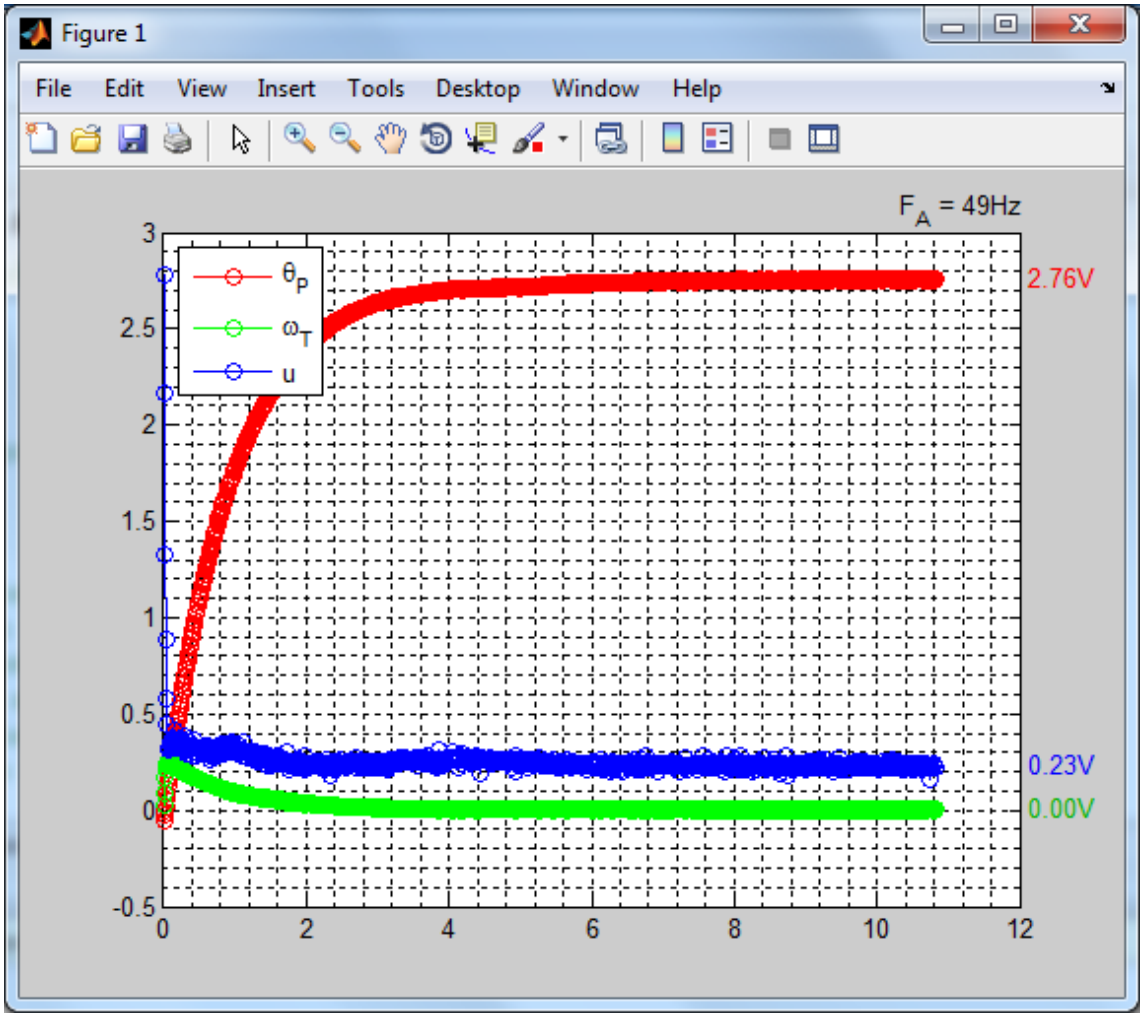

Figura 22. Controle lento, que permite encontrar o u para o qual o sistema não se move mais.

O sistema para de se mover quando $u = du$, sendo assim, para o caso do exemplo:

$$du = 0.23V$$

Com os parâmetros levantados até então, é possível fazer um projeto mais preciso de controlador.

5. PROPOSTA DE PLANTA INDUSTRIAL

Para a experiência em questão, assuma que o motor está acoplado a uma mão manipuladora em uma fábrica de bolos, cuja responsabilidade é tirar o bolo assado do forno e colocá-lo em uma esteira de saída, em que o mesmo será recheado e decorado. A Figura 23 ilustra a planta.

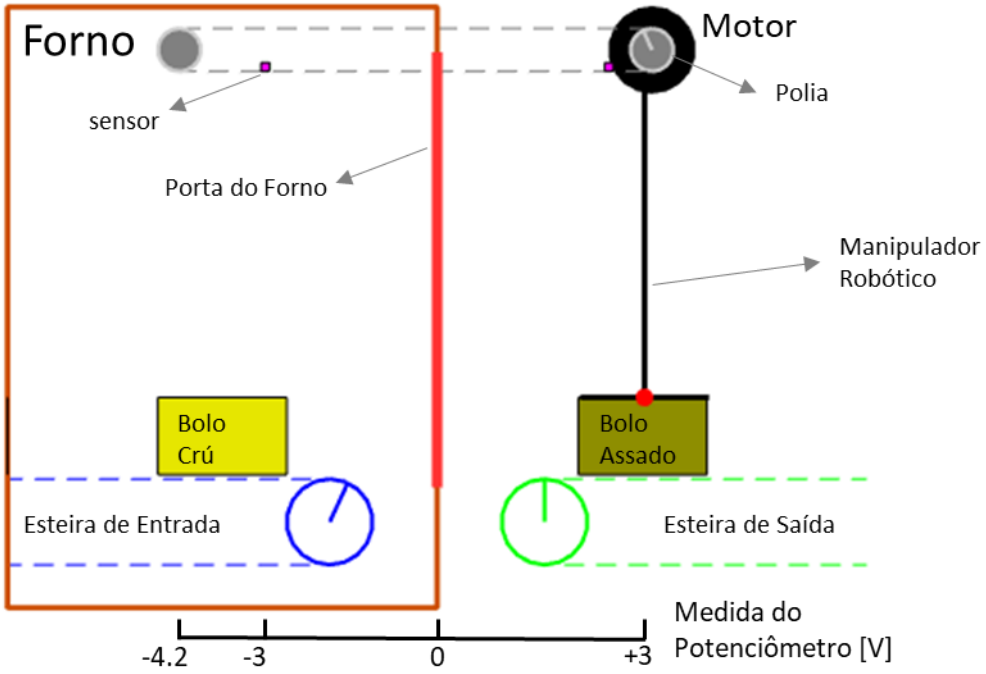

Figura 23. Planta industrial em uma fábrica de bolos.

Seu objetivo é projetar o controle do motor, sendo que o contratante estabelece as seguintes propriedades desejadas do sistema:

1. A mão manipuladora pode chegar no máximo no limite das polias, mas nunca deve ultrapassar esse limite (ou seja, só é permitido que o potenciômetro se mantenha na faixa -4.2 até 3V);
2. Após deixar o bolo assado na esteira de saída, a mão retorna para a posição -3V para esperar até que o novo bolo seja assado. A porta do forno fica aberta do momento que a mão começa a se mover, ou seja, do momento que a mesma sai da posição +3V, até o momento que ela cruza um sensor que se encontra na posição -1.8V. É

importante que o forno não fique aberto por mais que 0.6s para que a temperatura não caia abaixo do necessário e o bolo não fique cru. O manipulador só consegue pegar o bolo se ele se encontra a uma distância inferior a 0.06V do bolo (alta precisão é necessária, para que o manipulador não estrague o bolo);

3. Assim que o sistema registra que o bolo foi assado, ele aciona a mão para que a mesma leve o bolo assado do centro do forno (-3V) até a esteira de saída (+3V). O bolo deve estar longe do calor do forno (que irradia enquanto a porta estiver aberta) em até 1s. A porta fica aberta do momento que a mão começa a se mexer até o momento em que a mesma cruza um sensor localizado em +1.8V. O manipulador consegue liberar o bolo assim que ele se encontra parado a uma distância de ao menos 0.36V da posição final desejada de +3V (como o bolo é deixado na esteira, o posicionamento final pode ser mais grosseiro).
4. A esteira de entrada no forno começa a posicionar o novo bolo assim que o manipulador retira o bolo que estava no centro. Ela leva cerca de 1s para posicionar o bolo no centro do forno, e o bolo leva cerca de 5s para assar completamente, após atingir o centro do forno, ou seja, a mão manipuladora deve ser capaz de estar posicionada para pegar o bolo assado antes desse instante.

6. EXERCÍCIOS

6.1. Tradução para Requisitos Formais

Interprete as restrições da planta colocadas pelo contratante na forma de requisitos formais tradicionais em controle clássico, como: sobressinal percentual, tempo de subida, tempo de assentamento, erro estacionário, etc.

Exemplo:

- Sobressinal: $M_P < 5\%$
- Tempo de subida: subir 50% até 3s
- Tempo de assentamento: $t_s < 3s, tal\ que\ |e(t_s)| < 1\%$
- Erro estacionário: $|e(\infty)| < 0.5\%, ou\ 0.03V$

Perceba que existem duas estratégias possíveis:

1. Estratégia 1: Projetar um único controlador que satisfaça todas as condições necessárias;
2. Estratégia 2: Projetar dois controladores, um para quando o sistema deve ir de -3V para +3V e outro para quando o sistema precisa ir de +3V para -3V; chaveando os controladores de acordo com o sinal de referência desejado.

Os requisitos formais devem ser definidos para as duas possibilidades (ou seja, para a estratégia 1 haverá um grupo de requisitos formais que deve ser satisfeito em qualquer situação e para a estratégia 2 serão definidos dois grupos de requisitos formais, um para quando a mão vai de -3V para +3V e outro para quando a mesma vai de +3V para -3V).

6.2. Projeto do Controlador

O objetivo do exercício é projetar um controlador que satisfaça as condições anteriores usando a ferramenta SISOTOOL. Para facilitar as contas, tenha em mente que o erro máximo em um dado instante de tempo devido ao atrito é calculado pela fórmula:

$$du \cdot du2y(t) \cdot K_P = e_{atrito}(t)$$

Em que $du2y$ é o valor da função de transferência de du para y, e du é o valor calculado do atrito (no exemplo da apostila, 0.25V). Sendo assim, se desejo limitar o erro em e_{max}, em um dado instante de tempo:

$$du \cdot du2y(t) \cdot K_P < e_{max}$$

$$du2y(t) < \frac{e_{max}}{K_P \cdot du}$$

Outra fórmula que simplifica as análises é ter em mente que, para controladores PID:

$$u_{max} = P \cdot \frac{A_{degrau}}{K_P} = P \cdot \frac{6}{K_P}$$

$$P < K_P \frac{u_{sat}}{6}, para\ evitar\ saturação$$

Fora isso:

$$u_{max} = \frac{6V}{K_P} \cdot \max(r2u)$$

CONTROLADOR P

Inicialmente verifique se é possível utilizar apenas um controlador do tipo P para cada grupo de restrições (o grupo de restrições da estratégia 1 e os dois grupos de restrições da estratégia 2). Justifique.

CONTROLADOR PD

Verifique se é possível projetar apenas um controlador PD que satisfaça todas as condições (ou seja, verifique se é possível seguir a estratégia 1). Atente para o fato de que o controlador **nunca** deve saturar (ou seja, se o sistema sair de -3V e ir para 3V o sinal de controle não deve ultrapassar 5V em valor absoluto).

Se não for possível projetar um único controlador que satisfaça todas as condições, verifique se é possível satisfazer as restrições de algum grupo da estratégia 2.

CONTROLADOR PI-D

Projete o(s) controlador(es) PI-D para satisfazer as restrições que ainda não puderam ser satisfeitas com os controladores anteriores. Novamente, é importante que o controlador **nunca** sature dentro da operação definida. A aproximação a seguir ainda é válida (**justifique, informalmente, por quê**).

$$u_{max} = P \cdot \frac{A_{degrau}}{K_\theta} = P \cdot \frac{[3V - (-3V)]}{K_\theta} = \frac{6P}{K_\theta}$$

DICA: caso seja necessário otimizar o controle, e o mesmo não convirja, entre na aba Optimization, Optimization Options, e configure Optimization Method para Pattern Search, ao invés de Gradient Descent. Essa configuração permite que o otimizador procure controles quando o chute inicial fica um pouco mais grosseiro.

IMPLEMENTAÇÃO

Implemente o(s) controle(s) obtido(s) e verifique a resposta em degrau do sistema usando a função osciloscopio. As restrições são cumpridas como foram projetadas?

Se não, justifique a discrepância e reprojete o controlador, usando coeficientes de segurança (é natural que esse seja um projeto iterativo, e poderia ser deito usando uma simulação em simulink ao invés do motor real).

6.3. Hardware-in-the-Loop

A estratégia Hardware-in-the-Loop consiste em acoplar o sistema de controle real a uma simulação da planta industrial, de forma que seja possível verificar com certeza se os requisitos serão cumpridos, sendo uma prática comum em diversos ramos da engenharia como: engenharia automotiva, robótica, sistemas de potência e sistemas offshore.

Usualmente essa técnica é empregada para testar apenas o controlador desenvolvido, principalmente quando o mesmo é implementado em um sistema embarcado, de forma que o sistema embarcado é acoplado a planta simulada para a análise do comportamento. No nosso caso, o sistema real será o controlador e o motor, e a planta simulada será o processo industrial.

Erros identificados dessa forma apresentam risco zero. Um erro de sobressinal nesse caso, por exemplo, poderia significar um dano material a planta, representando elevados custos. Sendo assim, o esperado é que técnicas como o HIL sejam cada vez mais empregados na indústria.

É importante ressaltar que é imprescindível que a planta tenha um modelo fidedigno. Nessa experiência, a simulação é bem simplificada, sem levar em conta vários modelos da planta como a carga no motor, efeitos não-lineares como atrito de coulomb, perturbações, ruído de medida, etc, então espera-se que o exercício seja tomado mais como ilustração do que como exemplo real da prática do HIL.

Concluídas todas as considerações, o objetivo agora é rodar o controle usando a função HardwareInTheLoop.m, que simula a planta industrial, e visualizar o sistema funcionando (a implementação é feita da mesma forma que foi feita para função osciloscópio.m, entretanto, agora **a referência externa deve ser usada** sendo setada pela planta. Em caso de dúvida, verifique a função controlFunExampleHIL.m). O movimento do motor é replicado em tempo real na simulação do sistema, e é possível verificar se os requisitos estão sendo cumpridos ou se algum problema está interferindo no sistema (como quebra do sistema por erro de sobressinal, bolo queimado por ficar tempo demais no forno ou bolo cru devido à queda de temperatura do forno por ficar tempo demais aberto).

Inclua uma foto do resultado da simulação no relatório, deixando a planta rodar até que 10 bolos sejam produzidos.