Controlo de temperatura com o TClab

Pedro Magalhães (60302), Ricardo Coke (61368)

***Resumo*—In this article we present and discuss the results of the TCLab experiment proposed for the curricular unit of Controlo Digital (Digital Control) using MATLAB. TCLab is an Arduino with an extension featuring two transistors which act as heaters, and two temperature sensors. A first order plus time delay (FOPTD) model was identified for the TCLab system. Several types of control were implemented and different rules for controller tuning were compared. The types of control used were proportional (P), proportional derivative (PD), proportional integral (PI) and proporcional integral derivative (PID).**

***Index Terms*—Controlo Digital, TCLab, Arduino, MATLAB.**

I. Introdução

NO âmbito da unidade curricular de Controlo Digital foi proposta aos alunos a elaboração de um relatório de uma experiência usando o kit Temperature Control Lab (TCLab). O TCLab é uma aplicação de controlo de feedback com o micro-controlador Arduino que possui uma extensão com dois transístores que funcionam como aquecedores e dois sensores de temperatura que monitorizam durante as experiências. A energia térmica dos transístores é transferida para o sensor de temperatura e medida.

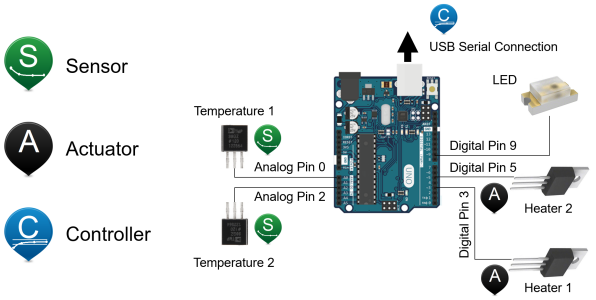

Figura 1: Kit TCLab

A experiência usando o kit TCLab proposta aos alunos consiste em:

- Controlo on-off.
- Teste de resposta ao degrau.
- Identificação de um modelo de primeira ordem mais atraso no tempo (FOPTD) do sistema pelo método dos 2 pontos.
- Teste de controladores P, PI, PD, e PID.
- Comparação entre os testes de controlo usando o kit TCLab e simulações no MATLAB usando o modelo FOPTD do sistema construído no Simulink.

Uma das grandes vantagens do kit TCLab usado neste projeto é que serve de alternativa às experiências práticas executadas em laboratório manuseando equipamentos caros em que os princípios dos controladores são postos de igual forma em teste e proporcionam os resultados próximos do que poderia ser esperado de um controlador real. [1]

II. Resultados e discussão

A. Parte I

1) Experiência 1 - Teste do LED: Nesta experiência foi testado o LED do kit TCLab, ligando e desligando ao longo de 5 segundos. Depois, foi controlada a luminosidade do LED durante 10 segundos, começando no limite máximo (mais brilhante), até desligar.

```
LED on and off repeatedly for 5 seconds
LED dimming from bright to off over 10 seconds
LED Test Complete
fx >>
```

Figura 2: Output do MATLAB

2) Experiência 2 - Teste de Temperatura: Foi ligado o aquecedor 1 a 30% e o aquecedor 2 a 60% durante 30 segundos, registando a temperatura inicial (antes do aquecimento) e final (depois do aquecimento).

```
Temperature 1: 22.3363 degC
Temperature 2: 22.3363 degC
Turn on Heater 1 to 30%, Heater 2 to 60%
Turn on LED for 30 seconds
Turn off Heaters
Temperature 1: 23.3138 degC
Temperature 2: 24.2913 degC
Temperature Test Complete
fx >>
```

Figura 3: Output do MATLAB

3) Experiência 3 - Controlo de Aquecimento: Foi ligado, 5 segundos depois do início, o aquecedor 1 a 60% da temperatura máxima, que equivale a 60°C, com a temperatura objetivo de 60°C durante 100 segundos. Após esse tempo, foi desligado o aquecedor 1. Após 150 segundos do início da experiência, foi ligado o aquecedor 2 a 80% da temperatura máxima, 80°C, com a temperatura objetivo de 80°C. Aos 205 segundos foi desligado o aquecedor 2. Durante o aquecimento do aquecedor 1 (dos 5 aos 105 segundos de experiência) verifica-se que há um pequeno aquecimento do aquecedor 2 dado que os dois aquecedor se encontram juntos e não há uma dissipação de calor efetiva a ser usada. Verifica-se também que o LED durante a experiência se liga quando a temperatura é superior a 30°C e vai aumentando a sua luminosidade até à máxima de acordo com o aumento da temperatura, sendo a luminosidade máxima atingida aos 100°C.

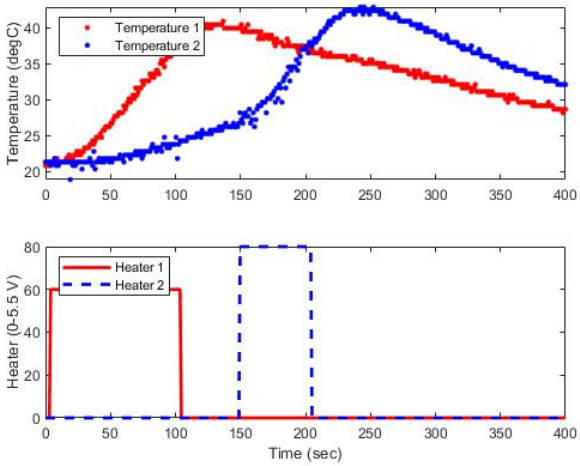

Figura 4: Registo das temperaturas e acionamento dos aquecedores ao longo do tempo

B. Parte II

1) Método dos dois pontos: Foi usado o método clássico dos dois pontos para a identificação de um modelo de primeira ordem com atraso no tempo (FOPTD) do sistema. Para este efeito, foi registada a resposta do sistema a um degrau em malha aberta, obtendo na saída uma função sigmoide.

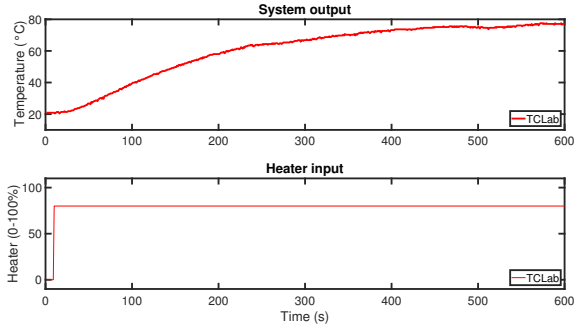

Figura 5: Resposta ao degrau em malha aberta

Começamos por determinar o primeiro ponto deste método (que ocorre aos 35.2%) e o segundo ponto (que ocorre aos 85.3%) da resposta obtida. Neste caso basta utilizar o gráfico criado pela saída do sistema utilizando as coordenadas x, que representa o tempo, e y, que representa a temperatura, das percentagens que se pretende obter. Especificamente (t35,y35) no caso do primeiro ponto que corresponde a 107 segundos e 40.83°C de temperatura e (t85,y85) no caso do segundo ponto, que corresponde a 328 segundos e 69.23°C de temperatura.

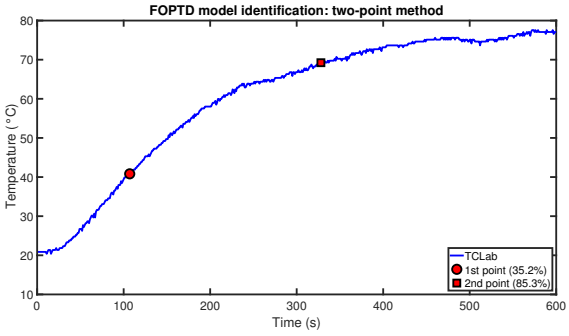

Figura 6: Representação dos dois pontos na resposta ao degrau

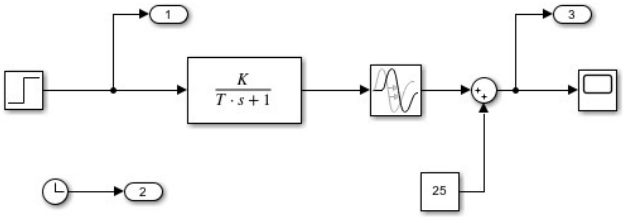

Figura 7: Modelo FOPTD da resposta ao degrau em malha aberta (Simulink)

A temperatura inicial e em regime permanente também foi determinada, sendo 20.87°C e 77.56°C respetivamente, obtidas através do programa em MATLAB. Quantos aos valores do ganho (K), constante de tempo (T) e o tempo de atraso (L) são determinados através das seguites fórmulas:

$$K = \frac{Yss - Yo}{Uss - Uo} = 0.7087 \tag{1}$$

$$T = 0.67(t_{85} - t_{35}) - t_0 = 148.07s \tag{2}$$

$$L = (1.3t_{85} - 0.29t_{35}) - t_0 = 33.88s \tag{3}$$

E o resultado do modelo de dois pontos FOPTD:

$$G_{2pt}(s) = \frac{0.7087}{1 + 148.07s} e^{-33.88s} \tag{4}$$

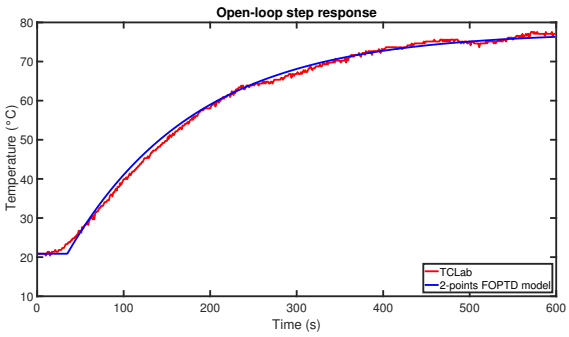

Figura 8: Comparação da resposta ao degrau em malha aberta entre os dados do TCLab e o modelo FOPTD obtido pelo método dos dois pontos

Para calcular o valor da integral do erro quadrático foram armazenados num vetor os dados de cada iteração correspondentes ao valor da temperatura e tempo que posteriormente vão ser subtraídos aos valores de cada ponto (correspondentes ao mesmo instante) no modelo de resposta ao degrau em malha aberta (simulink) que depois serão multiplicados, ponto a ponto. Fazendo o somatório desta multiplicação obtemos assim o integral do erro quadrático que no caso deste método foi de 550.6433.

2) Modelo Otimizado: A otimização por enxame de partículas (Particle Swarm Optimization, PSO) é um ramo da inteligência artificial que otimiza um problema iterativamente ao tentar melhorar a solução candidata com respeito a uma dada medida de qualidade [2]. No nosso caso através de uma adaptação de um dos add-ons do método PSO do MATLAB [3] a pesquisa foi conduzida assumindo intervalos para cada parâmetro (K,T e L) de [0.1 1.6],[120s 200s] e [16s 45s] respetivamente. Foram utilizadas 40 iterações, 20 partículas, as definições de inércia foram as que o MATLAB tinha por defeito [0.1 1.1] e os critério de minimização usado foi o da integração do erro quadrático.

O resultado do método otimizado (ISE_{PSO}=282.065) apresentou praticamente metade do valor obtido no método de dois pontos (ISE_{2pt} = 550.6433). Os valores de K,T e L obtidos através deste método foram hl0.7243, 162.4602 e 33.3980 respetivamente. Comparando os resultados dos dados obtidos pelo TCLab, do modelo de dois pontos e do modelo otimizado resultou o seguinte gráfico:

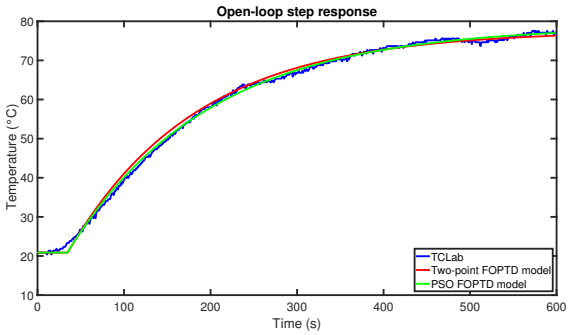

Figura 9: Comparação entre os dados do TCLab, o modelo FOPTD pelo método dos dois pontos, e o modelo FOPTD por PSO

Em que podemos verificar que o método otimizado acompanha de forma mais efetiva os dados reais.

3) Regras de Sintonia Adicionais: A partir dos novos valores de K,T e L usaremos novas regras de sintonia para além das de Cohen-Coon também, apenas para o controlador PID, de modo a verificar qual a saída mais eficaz entre elas.

- **Método IAE**, O valor a ser minimizado é dado pela interação do erro absoluto, sendo a área dada pela diferença entre o sinal de entrada e o de saída.

$$J = IAE = \int_0^\infty |e(t)|dt \tag{5}$$

$$K_p = (\frac{1.435}{K})(\frac{T}{L})^{-0.921} = 0.4615 \tag{6}$$

$$\frac{1}{T_i} = (\frac{0.878}{T})(\frac{L}{T})^{-0.749} = 0,0177 \tag{7}$$

$$T_d = (T * 0.482)(\frac{L}{T})^{-1.137} = 12.9612 \tag{8}$$

- **Método ITAE**, Este critério consiste na multiplicação entre o tempo e o erro absoluto, no entanto este critério de minimização acarreta um problema devido à perturbação que ocorre aos 400 segundos visto que é feita uma multiplicação pelo tempo e iria resultar numa desigualdade na integral do erro, tendo portanto de adaptar esta solução dividindo a resposta em duas partes: a SPT(Set Point Tracker) e LDR (Load Disturbance Rejection).

$$K_p = (\frac{1.357}{K})(\frac{T}{L})^{-0.947} = 0.4188 \tag{9}$$

$$\frac{1}{T_i} = (\frac{0.842}{T})(\frac{L}{T})^{-0.738} = 0.0167 \tag{10}$$

$$T_d = (T * 0.381)(\frac{L}{T})^{-0.995} = 12.8257 \tag{11}$$

- **Método AMIGO**, Neste caso trata-se de um método com abordagem computacional que tem como critério de otimização a maximização da parte integrativa do controlador, tendo como objetivo melhorar a resposta ao set-point.

$$K_p = (\frac{1}{K})(0.2 + 0.45\frac{T}{L}) = 3.2983 \tag{12}$$

$$T_i = (\frac{0.4L + 0.8T}{L + 0.1T})L = 96.4234 \tag{13}$$

$$T_d = \frac{0.5LT}{0.3L + T} = 15.7289 \tag{14}$$

- **Método S-IMC**, Para implementar este método foram calculadas novas constantes na Forma Série (PI): K'p, T'i e T'd, e para uma resposta mais agressiva usaremos a constante Tc com metade do valor de L. Existe também um fator de conversão que permite transformar as constantes anteriormente calculadas para o método PI em constantes para o método PID (Kp,Td e Ti).

$$K'_p = (\frac{1}{K})(\frac{T}{T_c + L}) = 4.4773 \tag{15}$$

$$T'_i = min(T, 4(T_c + L)) = 162.4602 \tag{16}$$

$$T'_d = \frac{L}{3} = 11.1327 \tag{17}$$

$$f = 1 + \frac{T'd}{T'i} = 1.0685 \tag{18}$$

$$K_p = K'_p * f = 4.7841 \tag{19}$$

$$T_i = T_i' * f = 173.5929 \tag{20}$$

$$T_d = \frac{T'd}{f} = 10.4187 \tag{21}$$

Nos últimos dois métodos foi necessário utilizar uma configuração do modelo em blocos diferente em que o ganho proporcional (Kp) é posicionado depois do primeiro bloco de subtração.

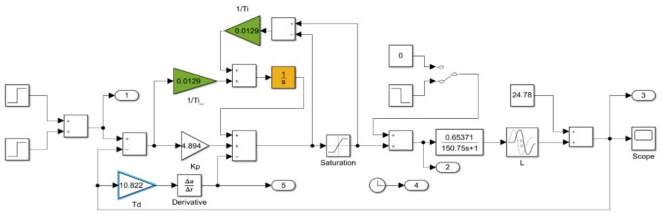

Figura 10: Modelo do sistema com controlador PID para as regras de sintonia AMIGO e S-IMC

III. Controladores

1) Controlo on-off: O Controlo On-Off mantém a temperatura alvo, onde o sistema está completamente funcional ou desligado, ou seja, o controlador é ligado e mede a temperatura, quando a temperatura desejada é alcançada, o controlador desliga. Inicialmente vai ter o objetivo de elevar a temperatura (primeiros 300 segundos) até 30°C desligando assim que a temperatura se verificar igual a esse objetivo. No entanto com o encerramento do funcionamento é expectável que a temperatura desça até que seja inferior a 30°C e o transístor seja forçado a ligar novamente. Isto vai provocar um pequeno conflito de vários on-off's seguidos, uma vez que a temperatura objetivo é atingida a primeira vez, que pode ser resolvida com a implementação de uma banda de histerese cujo objetivo é dar uma pequena "folga" de alguns graus antes que o transístor seja forçado a ligar ou desligar para compensar a temperatura atual.

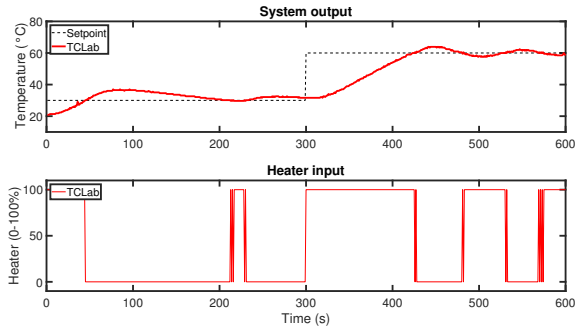

Figura 11: Controlo on-off sem banda de histerese

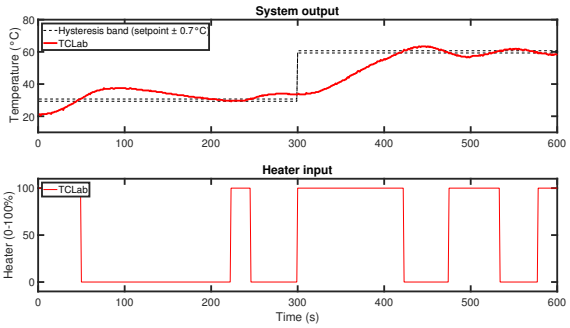

Figura 12: Controlo on-off com banda de histerese

Numa segunda fase, após 5 minutos do início do programa, a temperatura objetivo passará a ser 60°C tendo os transístores de aquecer novamente até se verificar uma temperatura igual ao objetivo. Sempre que é necessário fazer um aquecimento de modo a atingir a temperatura objetivo os transístores são colocados a aquecer a 100 %. Deste programa também resultou um ficheiro de texto que inclui todas as medidas de 1 em 1 segundo até o fim (10 minutos) que registou todas as medidas do tempo atual, funcionamento ou não do aquecedor (transístor ligado a 100% ou a 0%), temperatura do transístor 1 e temperatura do transístor 2 por essa ordem.

2) Resposta ao degrau em malha aberta: Desta vez o controlador vai ligar aos 10 segundos o aquecedor 1 de modo a atingir a temperatura objetivo de 80°C e vai funcionar durante 10 minutos. Verificamos que apesar do aquecedor 2 nunca ligar, a temperatura deste transístor acaba por subir porque está muito próximo do transístor 1, que está em funcionamento, e não havendo um sistema eficaz de dissipação de calor acaba por subir ligeiramente a sua temperatura (na casa dos 20°C).

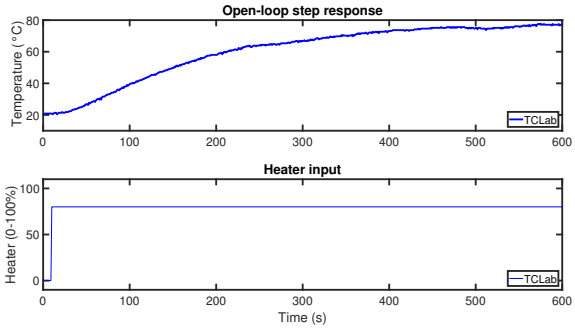

Figura 13: Resposta ao degrau em malha aberta

Do mesmo modo que a experiência anterior, também é registado todas as medidas num ficheiro de texto organizado com o tempo decorrido, funcionamento ou não dos dois transístores (aquecedor 1 e 2 respetivamente) e temperatura registada em ambos os transístores (1 e 2 respetivamente).

3) Controlador P: Este controlador consiste em projetar o ganho proporcional Kp no modelo de 1º ordem com atraso no tempo, usando as regras de Cohen-Coon A ação proporcional produz um sinal de saída que é proporcional à amplitude do erro e(t).

Para o domínio dos tempos vamos usar o modelo em simulink para fazer esta aproximação:

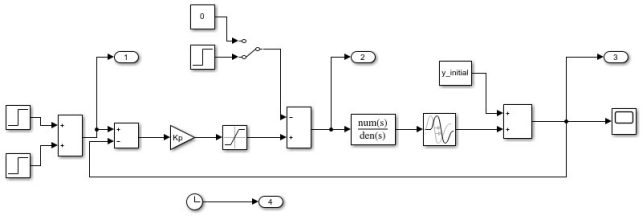

Figura 14: Modelo do sistema com controlador P

Foi preciso ajustar a saturação para variar entre os valores de 0 e 100, dois degraus que vão ser aplicados aos 0 segundos (início com amplitude de 25) e 100 segundos (amplitude de 35) obtendo no total uma amplitude de 60º que é o desejado aos 100 segundos.

A constante de ganho proporcional determinada foi:

$$K_p = \frac{1}{K}\frac{T}{L}(1 + \frac{L}{3T}) = 6.6372 \tag{22}$$

Os resultados obtidos foram os seguintes:

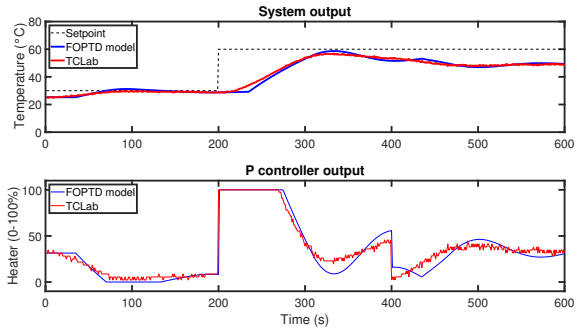

Figura 15: Controlo P

Conseguimos verificar que existe um erro em regime estacionário, que se traduz na diferença entre o sinal obtido pelo controlador e o sinal esperado teoricamente (set point). Esta diferença não é nula e deve-se à impossibilidade prática do controlador P eliminar o erro em regime estacionário, situação que poderá ser solucionada com a adição de uma ação integrativa como iremos verificar posteriormente. Todos os resultados obtidos pelo tclab são controlos digitais sendo os sinais discretos, sendo o tempo um múltiplo do tempo de amostragem que neste caso é representado por T. O arduíno é responsável por fazer a conversão digital-analógico para o transístor. É utilizada também uma regra que limita o valor do atuador de 0 a 100 para o caso dos valores saírem desse intervalo. Comparando os valores obtidos simulados e os reais (através do TCLab) conseguimos verificar que existe uma boa aproximação de ambos os sinais tendo valores praticamente idênticos durante todo o período de teste.

A função transferência correspondente a este controlador é:

$$G_c(s) = \frac{U(s)}{E(s)} = K_p \tag{23}$$

4) Controlador PI: O controlador proporcional integral apresenta duas componentes, a ação proporcional e a ação integral. No caso da ação integral que é a adição face ao controlador anterior é produzido um sinal de saída que é proporcional à magnitude e duração do erro. Deste modo conseguimos corrigir o off-set gerado pela ação proporcional e aceleramos a resposta do sistema permitindo obter o valor de referência de forma mais célere. No entanto um dos problemas presentes neste controlador é a saturação do controlador que vai provocar uma acumulação do integral do erro, fenómeno conhecido como ”Windup”que provoca um ”over-shoot”no resultado final. A solução para este fenómeno utilizada neste trabalho foi a do ”Anti-Windup”que consiste em parar a acumulação integrativa do erro. De modo a podermos aproximar o integral contínuo pela forma discreta no domínio digital, usamos um método recursivo denominado regra retangular para trás em que é traçado um retângulo com base na amostra anterior (k-1) e na amostra atual (k) com base T (tempo decorrido entre amostra anterior e amostra atual, neste caso 1 segundo) e altura ek (que representa o erro). O somatório de todos estes rectângulos vai representar o somatório integral do erro.

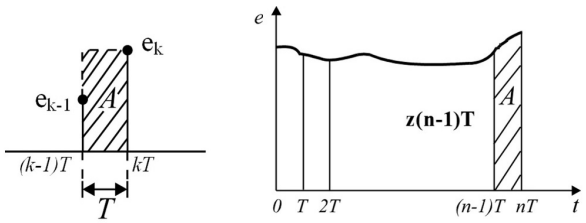

Figura 16: Regra retangular para trás.

- O erro é portanto a diferença entre a entrada do sistema e a saída.
- A ação proporcional corresponde à constante do ganho proporcional Kp (26) multiplicada pelo erro.
- A integral do erro é o retângulo calculado através do método em cima (tempo entre amostra multiplicado pelo erro atual) somado com o ”retângulo”anterior (no caso da primeira iteração este integral corresponde apenas ao valor do erro multiplicado pelo tempo entre amostras).
- A ação integrativa corresponde ao ganho proporcional dividido pela constante de tempo integrativa multiplicada pelo integral do erro.
- A saída do controlador corresponde à soma da ação proporcional com a ação integrativa.

Se o valor antes do atuador (soma da parte proporcional com a parte integrativa) for igual à saída (estiver entre 1 e 100 %) significa que o es vai ser igual a zero, logo:

$$K_i = \frac{K_p}{T_i} \tag{24}$$

Se o valor antes do atuador for superior à saída significa que o erro vai ser diferente de zero e o atuador vai estar saturado. Nesta situação é necessário parar a integração:

$$I = \frac{K_p}{T_i s}E + \frac{1}{T_i s}E_s = 0 <=> E_s = -\frac{K}{s}s \tag{25}$$

Quanto ao modelo do Simulink aproveitamos o modelo do controlador proporcional em que bastou adicionar o "Anti-Windup"de modo a poder "resetar"a Integração caso necessário, e uma perturbação (com um degrau) antes da função transferência de modo a que atue no instante 380 segundos.

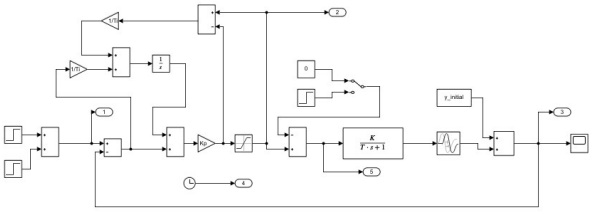

Figura 17: Modelo do sistema com controlador PI

Os resultados da simulação foram os seguintes:

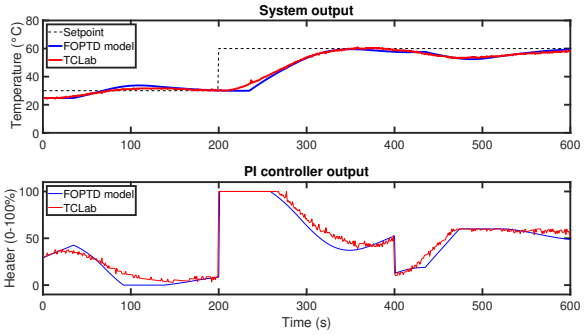

Figura 18: Controlo PI

Podemos observar que o erro em regime estacionário é nulo, mesmo após a perturbação aplicada aos 400 segundos o controlador consegue "recuperar"e eliminar o erro em regime estacionário no final do tempo de amostragem. Ao contrário da implementação deste controlador sem o Anti-Windup já não existe o over-shoot depois do primeiro degrau e já não existe tanta oscilação também. No caso do segundo gráfico (Pi Controller output) verifica-se que por volta dos 300 segundos sem o Anti-Windup a saturação obrigava o sinal a manter-se mais alto durante mais tempo o que neste caso não se verificou, obtendo uma resposta mais perto do pretendido mais cedo.

As constantes do ganho proporcional e tempo integrativo foram obtidas pelas regras de sintonia de Cohen-Coon em que:

$$K_p = \frac{1}{K}\frac{T}{L}(0.9 + \frac{L}{12T}) = 5.6677 \tag{26}$$

$$T_i = L + \frac{3T + 3L}{9T + 20L} = 76.5793 \tag{27}$$

$$K_i = \frac{K_p}{T_i} = 0.0740 \tag{28}$$

A função transferência da qual foi feita uma aproximação foi:

$$G_c = K_p(1 + \frac{1}{sT_i}) \tag{29}$$

A expressão da saída deste controlador é dada por:

$$G_c = K_p e(kT) + \frac{K_p}{T_i}\sum_{i=1}^{k} Te(iT) \tag{30}$$

5) Controlador PD: Relativamente ao controlador PD é acrescentada a ação derivativa à ação proporcional, havendo portanto a soma da parte derivada do erro com a parte proporcional do erro.

No domínio de Laplace a função transferência é a seguinte

$$G_c(s) = K_p(1 + sT_d) \tag{31}$$

No modelo simulink utilizado foi necessário fazer uma aproximação da ação derivada através do bloco derivativo presente no programa.

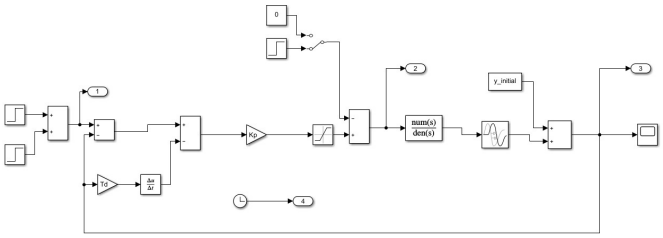

Figura 19: Modelo do sistema com controlador PD

De forma a aproximar digitalmente a derivada foi utilizada a seguinte função:

$$\frac{de(t)}{dt}\Big|_{t=kT} = \frac{e(kT) - e((k-1)T)}{T} \tag{32}$$

No programa MATLAB a derivada da saída vai corresponder à subtração da saída atual pela saída da iteração anterior a dividir pela constante de tempo e a ação derivativa à multiplicação da constante proporcional pela constante de tempo derivativo e derivada da saída. No entanto este controlador possuí uma particularidade, que é o salto derivativo, que ocorre devido à derivada de uma função degrau resultar numa mudança drástica de valor. A solução usada para contornar este erro foi aplicando a derivada não ao sinal do erro mas sim à saída controlada. Esta solução requer que o seu sinal seja negativo quando somado com a ação proporcional.

Usando as regras de Cohen-Coon obtemos as constantes do ganho proporcional e tempo derivativa com as seguintes grandezas:

$$K_p = \frac{1}{K}\frac{T}{L}(1.25 + \frac{L}{6T}) = 7.9437 \tag{33}$$

$$T_d = L\frac{6T - 2L}{22T + 3L} = 8.2770 \tag{34}$$

$$K_d = K_p T_d = 65.75 \tag{35}$$

$$D_k = K_p * T_d * D_{yk} \tag{36}$$

$$D_k = (\frac{T_f}{T_f + T_s}) * D_{(k-1)} + (\frac{T_s}{T_f + T_s}) * D_k \tag{37}$$

Os resultados obtidos pelo MATLAB e pela simulação do Simulink sem filtro e com filtro foram os seguintes:

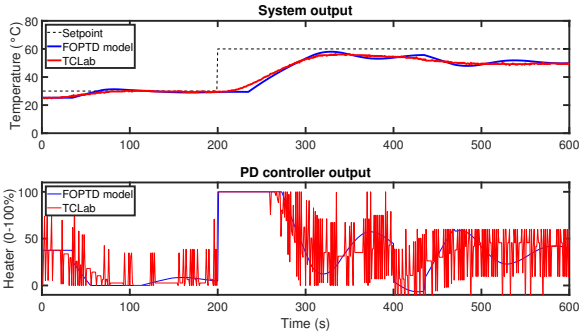

Figura 20: Controlo PD sem filtro

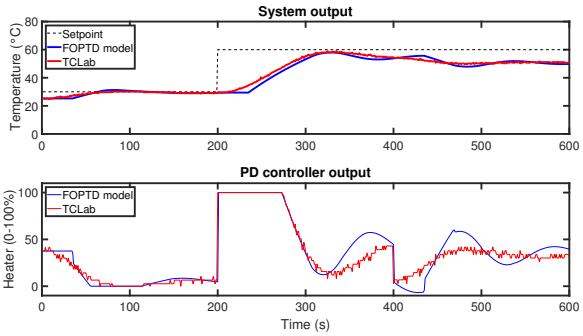

Figura 21: Controlo PD com filtro

A ação derivativa fornece uma correção antecipada do erro, diminuindo o tempo de resposta e melhorando a estabilidade do sistema. A constante de tempo derivativa é inversamente proporcional à velocidade de variação da variável controlada indicando que a ação derivativa não deve ser utilizada em processos nos quais o sistema deve responder rapidamente a uma perturbação, nem em processos que apresentem muito ruído no sinal de medido, pois levaria o processo à instabilidade sendo necessário aplicar um filtro tal como foi no nosso caso. Consegue-se verificar que após aplicado o degrau existe um erro em regime permanente no sinal como seria de esperar deste controlador que poderá ser corrigido aplicando uma ação integrativa, como vamos verificar posteriormente no controlador PID.

6) Controlador PID: O controlador PID apresenta as 3 componentes descritas anteriormente sendo elas a ação proporcional, a ação derivativa e a ação integrativa fazendo assim com que o sinal de erro seja minimizado pela ação proporcional, zerado pela ação integral e obtido com uma velocidade antecipativa pela ação derivativa.

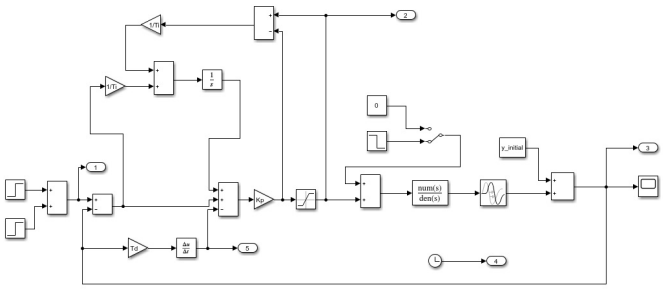

Figura 22: Modelo do sistema com controlador PID

Usando as regras de Cohen-Coon obtemos as constantes de tempo proporcional, integrativa e derivativa:

$$K_p = \frac{1}{K}\frac{T}{L}\left(\frac{4}{3} + \frac{L}{4T}\right) = 8.5752 \tag{38}$$

$$T_i = L\frac{32 + 6\left(\frac{L}{T}\right)}{13 + 8\left(\frac{L}{T}\right)} = 76.2398 \tag{39}$$

$$T_d = L\frac{4}{11 + 2\left(\frac{L}{T}\right)} = 11.8279 \tag{40}$$

A função transferência no domínio de Laplace do controlador PID ideal, da qual vai ser feita uma aproximação é a seguinte:

$$G_c(s) = K_p\left(1 + \frac{1}{sT_i} + sT_d\right) \tag{41}$$

Tal como o caso do controlador PD, esta experiência foi testada com e sem filtro obtendo saídas ligeiramente diferentes com muito menos ruído no caso do teste com filtro, como seria de esperar.

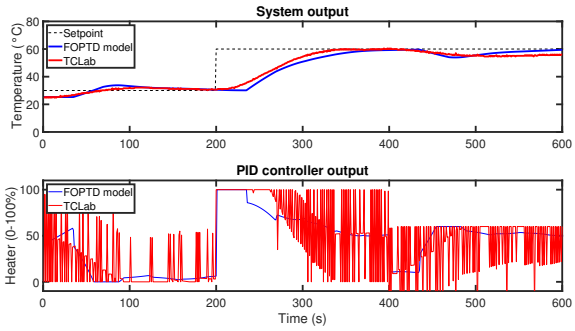

Figura 23: Controlo PID sem filtro

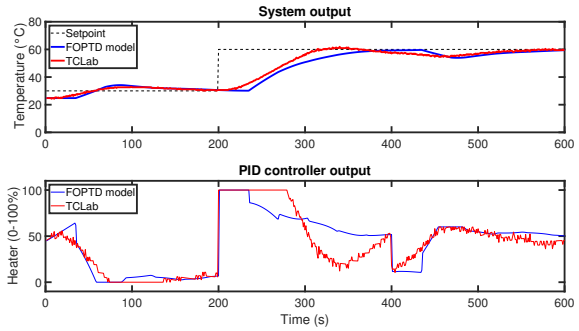

Figura 24: Controlo PID com filtro

Houve algumas melhorias na saída obtida com a inclusão da ação integrativa face ao controlador PD, obtendo de forma mais rápida o valor pretendido (sem erro em regime estacionário) no caso do controlador PID com filtro. Através do modelo otimizado obtemos resultados ligeiramente diferentes:

- **Regra de Sintonia Cohen-Coon**

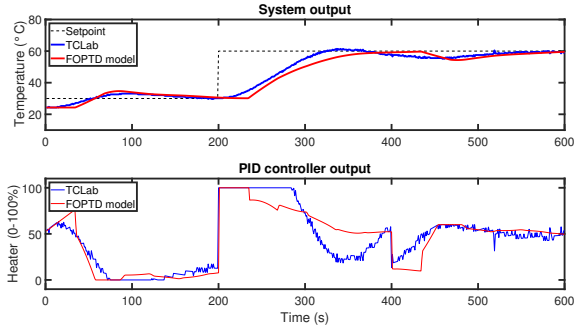

Figura 25: Controlo PID usando o modelo FOPTD PSO e regras de sintonia de Cohen-Coon

Como se pode verificar não houve melhorias significativas face ao modelo de dois pontos em nenhum aspeto.

- **Regra de Sintonia IAE**,

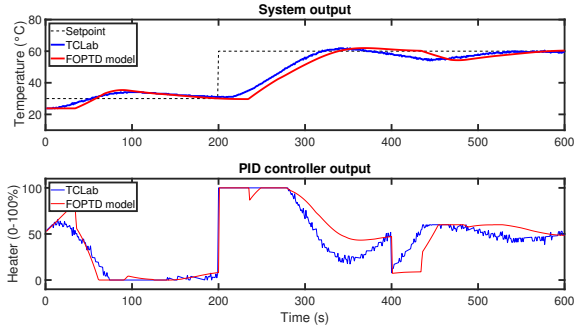

Figura 26: Controlo PID usando o modelo FOPTD PSO e regras de sintonia de IAE

Esta regra de sintonia conseguiu obter resultados ligeiramente melhores que o método de CC, com os valores bastante mais próximos no intervalo de 200 a 400 segundos.

- **Regra de Sintonia ITAE**,

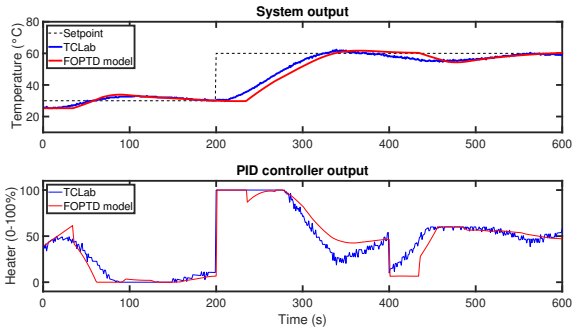

Figura 27: Controlo PID usando o modelo FOPTD PSO e regras de sintonia de ITAE

De todos os métodos este foi o que consideramos melhor, dado que o resultado final não apresenta erro em regime estacionário, e os resultados do tclab e simulação na saída do controlador PID estão mais próximos que as outras regras acabando também com os dois sinais pouco desfasados.

- **Regra de Sintonia AMIGO**,

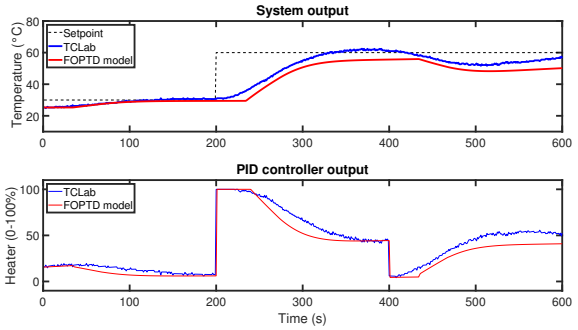

Figura 28: Controlo PID usando o modelo FOPTD PSO e regras de sintonia de AMIGO

Este método foi o que obteve melhores resultados no gráfico da saída do controlador PID, estando ambos os sinais a acompanhar-se no entanto acabam com um erro em regime estacionário grande face aos outros métodos.

- **Regra de Sintonia S-IMC**,

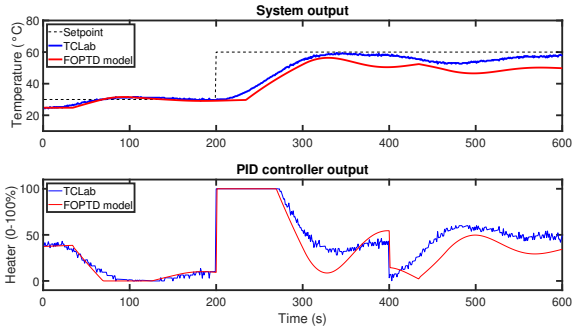

Figura 29: Controlo PID usando o modelo FOPTD PSO e regras de sintonia de S-IMC

IV. Conclusões

Com esta análise dos vários controladores, é possível verificar através dos resultados a influência que cada controlador tem no sinal de saída, havendo algumas situações particulares no caso do controlador derivativo e integrativo que foram solucionadas através de métodos adicionais (Derivative Kick e Anti-Windup). Podemos então concluir que a ação proporcional produz um sinal de saída que é proporcional à amplitude do erro e, a ação integrativa é responsável pela eliminação do erro em regime permanente e a ação derivativa fornece uma correção antecipada do erro, diminuindo o tempo de resposta e melhorando a estabilidade do sistema. Quanto ao método otimizado usando o algoritmo PSO obtivemos resultados melhores no que toca à grandeza do integral do erro quadrático. Relativamente às novas regras de sintonia usadas para o controlador PID podemos concluir que o melhor método foi o de integração multiplicação entre o tempo e o erro absoluto (ITAE) obtendo um valor final com erro em regime estacionário praticamente nulo, recuperando da perturbação imposta aos 400 segundos e obtendo valores muito poucos desfasados entre os sinais do modelo e dados reais.

Em relação ao método como o relatório foi planeado foi sem dúvida uma alternativa extremamente útil que nos proporcionou a experiência de ensino que obteríamos num laboratório real, sem parte da responsabilidade a manusear equipamento caro mas que acaba por trazer resultados bem próximos da realidade e pela mobilidade de podermos realizar este tipo de experiências em qualquer sítio a qualquer hora, só temos a agradecer a iniciativa do professor Paulo Oliveira responsável pela unidade curricular de Controlo Digital e apoiar que este estilo de proatividade no meio do ensino continue a ser desenvolvida.

Referências

[1] P. M. Oliveira and J. D. Hedengren, "An apmonitor temperature lab pid control experiment for undergraduate students," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2019, pp. 790–797.

[2] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-International Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[3] . MathWorks. Particle swarm optimization. [Online]. Available: https://www.mathworks.com/help/gads/particleswarm.html