Mathew Turnell

Modelagem, Simulação e Controle de um módulo termoelétrico de Peltier

Campina Grande - PB, Brasil

2013

Mathew Turnell

Modelagem, Simulação e Controle de um módulo termoelétrico de Peltier

Trabalho de conclusão de curso

Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Departamento de Engenharia Elétrica

Programa de Graduação em Engenharia Elétrica

Orientador: George A. Júnior

Campina Grande - PB, Brasil

2013

Mathew Turnell

Modelagem, Simulação e Controle de um módulo termoelétrico de Peltier

Trabalho de conclusão de curso

Trabalho aprovado. Campina Grande - PB, Brasil, 14 de Julho de 2013:

George A. Júnior
Orientador

Professor
Convidado 1

Campina Grande - PB, Brasil
2013

Agradecimentos

A meu orientador de estágio e tcc pelas correções e contribuições. A Rafael pela ajuda para implementar o processo de Peltier no simulador OPC. A Moisés pela ajuda nos testes com o sensor de corrente. Aos professores e colegas que me acompanharam nestes vários anos de curso e ajudaram a adiantar a minha conclusão. A minha família e todos que ajudaram a tornar suportável e mesmo prazeroso o ciclo de engenharia.

“"Success is not final...
Failure is not fatal...
It’s the courage to continue that counts."
Winston Churchill

Resumo

Este trabalho de conclusão de curso descreve a modelagem de um módulo termo-elétrico de Peltier utilizando as equações diferenciais parciais não-lineares que regem o seu funcionamento. As equações serão discretizadas utilizando a técnica das diferenças finitas o que permitirá sua simulação numérica no ambiente MATLAB/-Simulink. O modelo será validado utilizando a planta real do processo e os resultados serão utilizados para a síntese e teste de leis de controle.

Palavras-chaves: Modelagem. Simulação. Módulo Termo-elétrico de Peltier

Abstract

This work describes the modeling of a thermo-electric cooler using nonlinear partial differential equations that govern its operation. The equations are discretized using the finite difference method which allows for its numerical simulation using MATLAB / Simulink. The model will be validated using the process plant and will be used for the synthesis and testing of control laws.

Key-words: Simulation. Thermoelectric cooler. Peltier effect

Lista de ilustrações

Lista de tabelas

Sumário

1 Fundamentação Teórica - Definições

1.1 Calor

Em termodinâmica, calor é o processo de transferência de energia em que esta é transferida de um sistema para outro devido ao contato térmico e a uma diferença de temperatura. O termo energia térmica é utilizado para definir a energia de um corpo que aumenta com a temperatura. Temperatura é uma medida da energia interna ou entalpia e está associada ao grau de agitação das moléculas.

1.1.1 Calor Específico

O calor específico de um material é definido como a quantidade de calor necessária para aumentar em 1 Kelvin a temperatura de 1 g do material. Sua unidade é o Joule/g.K.

Para um material de massa m e calor específico c, a elevação de temperatura devido à absorção de uma quantidade dQ de calor num intervalo de tempo dt é dada por:

$$dQ = mcdT \tag{1.1}$$

Se uma taxa de calor $\frac{dQ}{dT}$ é fornecida, a taxa de variação de temperatura será então:

$$\frac{dQ}{dt} = mc\frac{dT}{dt} \tag{1.2}$$

1.2 Transferência de Calor

A transferência de calor ocorre espontaneamente de um corpo de maior temperatura para um de menor temperatura, e pode ocorrer por radiação, condução ou convecção.

1.2.1 Condução

Condução é a transferência de energia térmica entre interfaces adjacentes devido a um gradiente de temperatura que não envolve a transferência de matéria.

1.2.1.1 Lei de Fourier

A lei de Fourier para a transferência de calor diz que a taxa de transferência de calor através de uma superfície S é proporcional à sua área e ao gradiente de temperatura

sobre a mesma. Na forma integral:

$$\frac{dQ}{dt} = -k \oint_S \nabla T.\mathbf{dA} \tag{1.3}$$

onde:

$\frac{dQ}{dt}$ é o fluxo de calor em Watts, ou seja, a quantidade de calor por unidade de tempo que atravessa a superfície S.

k é a condutividade térmica do material, dada em W/mK.

dA é um elemento de superfície orientado.

∇T é o gradiente de temperatura no elemento de superfície.

Para uma superfície plana de área A a equação se reduz a:

$$\frac{dQ}{dt} = -kA\frac{dT}{dx} \tag{1.4}$$

$\frac{dT}{dx}$ é o gradiente de temperatura na direção perpendicular à superfície dado em K/m.

1.2.1.2 Condutância e Resistência Térmica

A partir da equação 1.4, se a superfície plana em questão tem espessura Δx e a distribuição de temperatura é uniforme, pode-se definir a sua condutância por:

$$U = \frac{kA}{\Delta x} \tag{1.5}$$

O inverso da condutância é denominado resistência térmica:

$$R = \frac{1}{U} \tag{1.6}$$

Ambas a condutância e resistência térmica podem ser medidas experimentalmente para um dado objeto. Assim, a Lei de Fourier pode ser reescrita como:

$$\frac{dQ}{dt} = -U\Delta T = -\frac{\Delta T}{R} \tag{1.7}$$

1.2.2 Convecção

Convecção é a transferência de calor devido ao deslocamento da matéria, mais especificamente, o movimento de fluidos. Há dois tipos de convecção:

- Convecção natural: Quando o deslocamento do fluido é causado pelo empuxo resultante de uma diferença de densidade, causada por variações na temperatura do fluido. Um exemplo simples é circulação de água em uma panela aquecida.
- Convecção forçada: Quando o fluido é forçado a se deslocar por um dispositivo externo como um ventilador ou bomba.

1.2.2.1 Lei de Newton do resfriamento

A lei de Newton para o resfriamento diz que a taxa de transferência de calor de um corpo, devido à convecção, é proporcional à diferença de temperatura entre o corpo e o ambiente e à sua área de contato.

$$\frac{dQ}{dt} = hA\Delta T \tag{1.8}$$

Onde:

$\frac{dQ}{dt}$ é o fluxo de calor em Watts, ou seja, a quantidade de calor por unidade de tempo que deixa o corpo.

h é o coeficiente de transferência térmica dado em W/m^2K.

ΔT é a diferença entre a temperatura do corpo e a temperatura ambienteK.

O valor de h pode ser determinado experimentalmente para diferentes objetos e depende do tipo de convecção.

1.2.3 Radiação

Radiação é a transferência de energia térmica através de ondas eletromagnéticas. Todo objeto com temperatura maior que o zero absoluto irradia energia.

1.3 Efeito Termoelétrico

O efeito termoelétrico é a conversão direta entre tensão e diferença de temperatura e vice-versa. Um dispositivo termoelétrico cria uma tensão quando uma diferença de temperatura é aplicada. Inversamente, quando uma tensão é aplicada aos seus terminais, uma diferença de temperatura é criada.

Os dispositivos termoelétricos podem ser utilizados para gerar eletricidade, medir temperatura, aquecer ou resfriar objetos.

Os principais efeitos termoelétricos são:

- O efeito Seebeck, descoberto pelo físico Estoniano Thomas Johann Seebeck

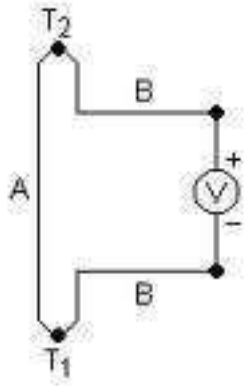

Figura 1: Par de materiais distintos A e B com junções em diferentes temperaturas

- O efeito Peltier, descoberto pelo físico Frances Jean Charles Athanase Peltier
- O efeito Thomson, descoberto pelo físico Britânico William Thomson (Lord Kelvin)

Estes efeitos estão relacionados com o processo de difusão de portadores de carga.

1.3.0.1 Difusão de portadores de carga

Quando as extremidades de um condutor estão em temperaturas diferentes, elétrons e lacunas se deslocam espontaneamente para a extremidade onde há uma menor concentração destes.

A carga elétrica líquida produzida não é nula pois a taxa de recombinação não é igual em ambas as extremidades. Isto se deve à dopagem do semicondutor, impurezas, imperfeições e fônons. Assim, há um acúmulo de cargas nas extremidades do condutor e surge uma diferença de potencial.

O campo elétrico gerado pelo acúmulo de cargas se opõe ao deslocamento de novas cargas e um equilíbrio é alcançado. Para obter uma maior diferença de potencial, é necessário aumentar a diferença de temperatura.

1.3.0.2 Efeito Seebeck

Quando um par de metais distintos (ou semicondutores) são unidos em uma junção cuja temperatura é diferente da do outro terminal, surge nestes uma tensão da ordem de microvolts por Kelvin. (Ver figura 1)

Em termos matemáticos:

$$V = \int_{T_1}^{T_2} S_B(T) - S_A(T) dT \tag{1.9}$$

Onde $S_A(T) e S_B(T)$ são os coeficientes de Seebeck dos materiais A e B e $T_2 > T_1$ são as temperaturas da junção quente e fria respectivamente. A unidade do coeficiente de Seebeck é o $\mu V/K$. Como se pode observar, estes coeficientes dependem do material e sua estrutura molecular e da temperatura absoluta.

Se estes coeficientes variam pouco entre a faixa de temperatura $T_1 a T_2$, a equação 1.9 pode ser aproximada por:

$$V = (S_B - S_A)(T_2 - T_1) \tag{1.10}$$

Pode-se observar que se os materiais A e B são iguais ou as temperaturas das junções são iguais, a tensão V observada é nula.

O efeito de Seebeck é o princípio de operação dos termopares, dispositivos utilizados para medir temperatura.

1.3.0.3 Efeito Peltier

O efeito Peltier diz que quando uma corrente I atravessa a junção de dois metais (ou semicondutores) distintos, há uma absorção ou emissão de calor na junção. Para dois materiais A e B tais que a corrente I flui de A para B:

$$\frac{dQ}{dt} = (\Pi_B - \Pi_A) I \tag{1.11}$$

Onde $\Pi_A e \Pi_B$ são os coeficientes de Peltier para os materiais A e B respectivamente e dependem da temperatura da Junção. Nota-se que se $\Pi_A > \Pi_B$ há geração de calor, caso contrário há absorção de calor.

Os coeficientes de Peltier e Seebeck estão relacionados por:

$$\Pi = ST \tag{1.12}$$

O efeito peltier é o princípio de operação dos dispositivos de refrigeração termoelétricos.

1.3.0.4 Efeito Thomson

O efeito Thomson diz que qualquer condutor de corrente submetido a um gradiente de temperatura produz ou absorve calor dependendo do material. A taxa de produção de calor por unidade de volume é:

$$q = \rho J^2 - \mu J \frac{dT}{dx} \tag{1.13}$$

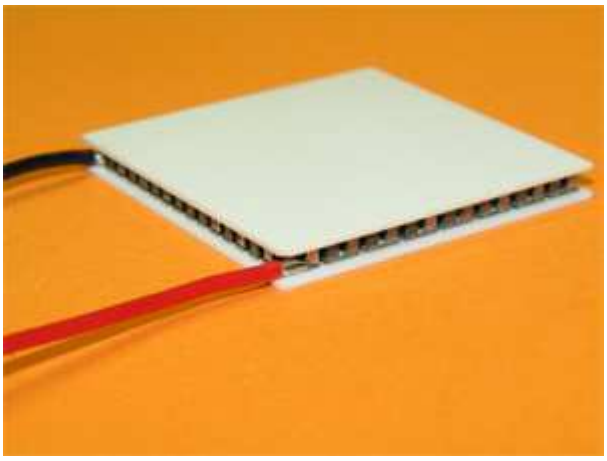

Figura 2: Um módulo de Peltier comercial

Onde:

ρ é a resistividade do condutor

J é a densidade de corrente

μ é o coeficiente de Thomson do material dado em J/pascal e depende da temperatura T

$\frac{dT}{dx}$ é o gradiente de temperatura do condutor

O coeficientes de Thomson e Seebeck estão relacionados por:

$$\mu = T\frac{dS}{dT} \tag{1.14}$$

1.4 Módulo de Peltier

Módulos de peltier são dispositivos que utilizam o efeito Peltier para transferir calor de um lado frio para um lado quente com a aplicação de uma corrente em seus terminais.

São frequentemente utilizados para o resfriamento de dispositivos eletrônicos como CPUs, sensores de infravermelho, manutenção de temperaturas de referência em termopares e refrigeradores.(HUANG; DUANG, 2000)

A eficiência de um módulo de peltier depende da quantidade de calor transferida, da diferença de temperatura entre seus lados e da corrente aplicada.

Um módulo de Peltier típico é composto de junções tipo P e N do semicondutor Teluro de Bismuto com um metal M segundo a figura 3. Duas junções tipo P e N formam o que se denomina uma junção ou par. O módulo possui várias junções conectadas eletricamente em série 4.

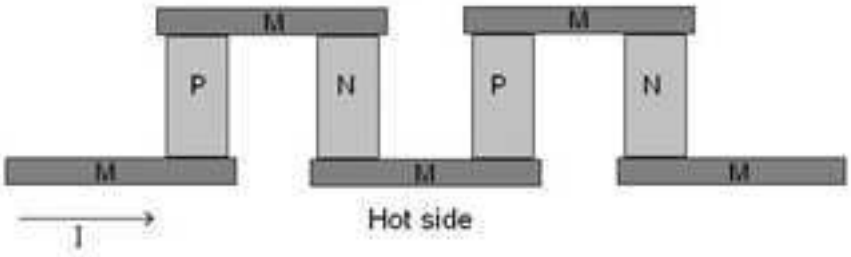

Figura 3: Secção transversal de um módulo de Peltier

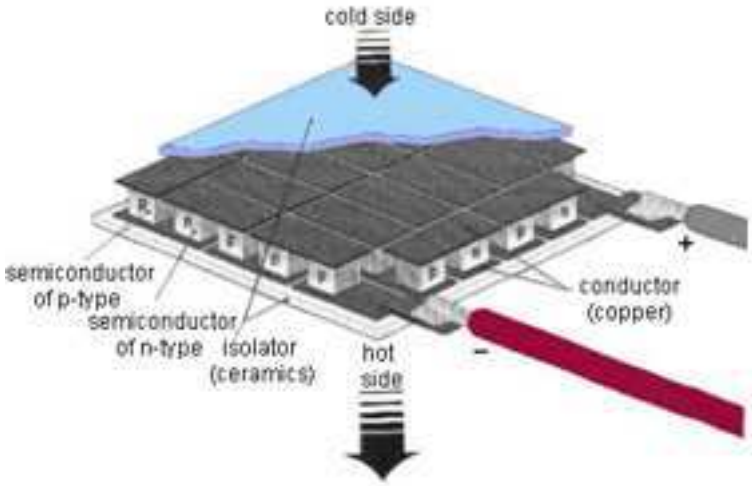

Figura 4: Junções conectadas eletricamente em sére e termicamente em paralelo

Os semicondutores tipo P e N e o metal M são tais que seus coeficientes de Seebeck obedecem:

$$\alpha_N > \alpha_M > \alpha_P \tag{1.15}$$

Segundo a equação 1.11, quando a corrente I atravessa cada junção da figura 3, calor é absorvido nas junções P-M e M-N e produzido nas junções M-P e N-M. Deste modo, cria-se uma região de absorção de calor (topo) e produção de calor (base). Várias junções são dispostas termicamente em paralelo para amplificar este efeito.

O módulo de peltier típico usa também duas pastilhas cerâmicas finas em ambos os lados que fornecem a isolação elétrica e a rigidez mecânica necessária. (ALVES, 2007)

2 Modelo do Refrigerador com módulo de Peltier

2.1 Descrição física

O processo modelado consiste de um módulo de peltier utilizado para resfriar uma placa metálica conectada a seu lado frio submetido a um aquecimento. Um dissipador e um ventilador (cooler) são conectados ao lado quente (ver figura 5).

A seguir, o modelo matemático do processo será desenvolvido.

2.2 Descrição matemática

(HUANG; DUANG, 2000) utilizam o princípio da conservação da energia para deduzir as equações descritas a seguir para o refrigerador de Peltier.

2.2.1 Material Semicondutor

Para um elemento tipo P ou N de condutor, temos a ação dos efeito Thomson e a condução de calor do lado quente para o lado frio do material.

Se A é a área de um elemento (ver figura 6), temos que para uma secção de volume $A\partial x$ a uma distância x do lado frio, o calor produzido pelo efeito Thomson (ver eq. 1.13) num intervalo ∂t é:

$$Q_T = (A\partial x\partial t)(\rho J^2 - \mu J\frac{\partial T(x,t)}{\partial x}) \tag{2.1}$$

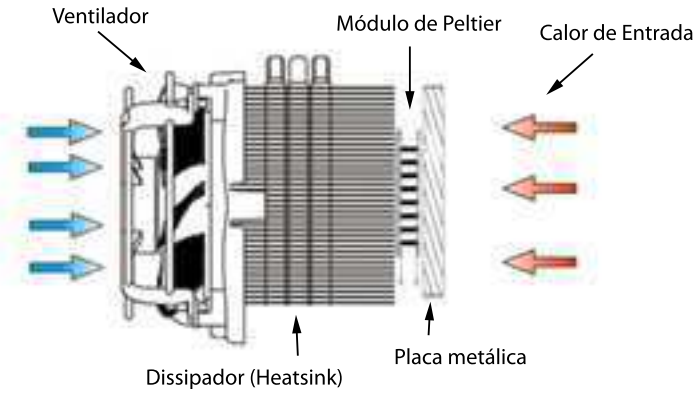

Figura 5: Processo Modelado

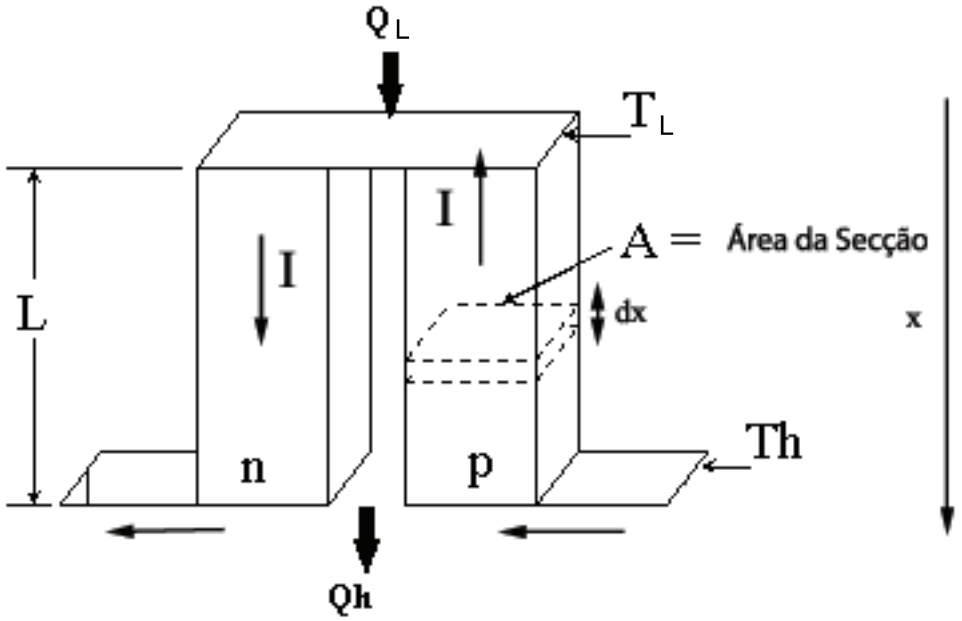

Figura 6: Uma junção PN

Onde:

$T(x,t)$ é a temperatura do semicondutor a uma distância x e instante t

ρ é sua resistividade

J é sua densidade de corrente

μ é seu coeficiente de Thomson

O calor transferido pela lei de Fourier para esta secção é(ver eq. 1.4):

$$Q_F = kA\frac{\partial T(x,t)}{\partial x}\partial t \tag{2.2}$$

Se C é o calor específico do semicondutor e γ é sua densidade, para uma variação de temperatura $\partial T(x,t)$ da secção num intervalo ∂t o calor absorvido é (ver eq. 1.2):

$$Q_C = C\gamma A\partial x\partial T(x,t) \tag{2.3}$$

Pela conservação da energia temos então:

$$Q_C = Q_T + Q_F \tag{2.4}$$

ou seja:

$$C\gamma A\partial x\partial T(x,t) = kA\frac{\partial T(x,t)}{\partial x}\partial t + (A\partial x\partial t)(\rho J^2 - \mu J\frac{\partial T(x,t)}{\partial x}) \tag{2.5}$$

Dividindo a equação por $A\partial x\partial t$ e utilizando a corrente ao invés da densidade de corrente ($I = JA$):

$$C\gamma\frac{\partial T(x,t)}{\partial t} = k\frac{\partial^2 T(x,t)}{\partial x^2} + \rho\frac{I^2}{A^2} - \mu\frac{I}{A}\frac{\partial T(x,t)}{\partial x} \tag{2.6}$$

2.2.2 Lados quente e frio

Para o conjunto formado pela placa cerâmica do lado frio e a placa metálica, a taxa de calor absorvido $\frac{dQ_p}{dt}$ pelo efeito de peltier é (eq. 1.11):

$$\frac{dQ_P}{dt} = -In\alpha_{PN}T_L \tag{2.7}$$

Onde $\alpha_{PN} = \alpha_P - \alpha_N$ é a diferença entre os coeficientes de Seebeck para o semicondutor tipo P e tipo N, n é o número de junções do módulo e T_L é a temperatura do lado frio.

A taxa de calor absorvido por condução $\frac{dQ_F}{dt}$ do lado quente para o lado frio segundo a lei de Fourier é (eq. 1.4):

$$\frac{dQ_F}{dt} = kA_m\frac{dT(x,t)}{dx}\Big|_{x=0} \tag{2.8}$$

Onde A_m é a área das placas cerâmicas do módulo de Peltier. Se M_L e M_C são as massas da placa metálica e da placa cerâmica respectivamente e C_L e C_C são seus calores específicos, a taxa de calor absorvida por estas massas $\frac{dQ_E}{dt}$ para uma taxa de variação de temperatura $\frac{dT_C}{dt}$ é dada pela eq. 1.2:

$$\frac{dQ_C}{dt} = (M_LC_L + M_CC_C)\frac{dT_L}{dt} \tag{2.9}$$

Assim, se Q_L é a taxa de calor fornecido ao lado frio, pela conservação da energia temos:

$$\frac{dQ_C}{dt} = Q_L + \frac{dQ_F}{dt} + \frac{dQ_P}{dt} \tag{2.10}$$

Ou seja:

$$(M_LC_L + M_CC_C)\frac{dT_L}{dt} = Q_L + kA_m\frac{dT(x,t)}{dx}\Big|_{x=0} - In\alpha_{PN}T_L \tag{2.11}$$

Onde:

I é a corrente fornecida ao módulo de Peltier.

Para o conjunto formado pela placa cerâmica do lado quente e o dissipador,a taxa de calor liberada pelo dissipador é segundo a eq. 1.6:

$$\frac{dQ_D}{dt} = -\frac{(T_H - T_a)}{R_{th}} \tag{2.12}$$

Onde R_th é a resistência térmica do dissipador, T_H e T_a são a temperatura do lado quente e do ambiente respectivamente. De maneira análoga à anterior, temos que as taxas de calor devido ao efeito peltier e condução são:

$$\frac{dQ_P}{dt} = +In\alpha_{PN}T_H \tag{2.13}$$

e

$$\frac{dQ_F}{dt} = -kA_m\frac{dT(x,t)}{dx}\Big|_{x=L} \tag{2.14}$$

Se M_F e M_H são as massas do dissipador e da placa cerâmica respectivamente e C_F e C_H são seus calores específicos, a taxa de calor absorvida por estas massas $\frac{dQ_E}{dt}$ para uma taxa de variação de temperatura $\frac{dT_H}{dt}$ é dada pela eq. 1.2:

$$\frac{dQ_H}{dt} = (M_FC_F + M_HC_H)\frac{dT_H}{dt} \tag{2.15}$$

Assim, temos:

$$\frac{dQ_H}{dt} = \frac{dQ_D}{dt} + \frac{dQ_F}{dt} + \frac{dQ_P}{dt} \tag{2.16}$$

$$(M_FC_F + M_HC_H)\frac{dT_H}{dt} = -\frac{(T_H - T_a)}{R_{th}} - kA_m\frac{dT(x,t)}{dx}\Big|_{x=L} + In\alpha_{PN}T_H \tag{2.17}$$

2.3 Equações do modelo

As equações do modelo são então:

$$C\gamma\frac{\partial T(x,t)}{\partial t} = k\frac{\partial^2 T(x,t)}{\partial x^2} + \rho\frac{I^2}{A^2} - \mu\frac{I}{A}\frac{\partial T(x,t)}{\partial x} \tag{2.18}$$

$$(M_L C_L + M_C C_C)\frac{dT_L}{dt} = Q_L + kA_m\frac{dT(x,t)}{dx}\Big|_{x=0} - In\alpha_{PN}T_L \tag{2.19}$$

$$(M_F C_F + M_H C_H)\frac{dT_H}{dt} = -\frac{(T_H - T_a)}{R_{th}} - kA_m\frac{dT(x,t)}{dx}\Big|_{x=L} + In\alpha_{PN}T_H \tag{2.20}$$

2.4 Valores dos parâmetros

Uma breve revisão bibliográfica foi feita para obter valores típicos dos parâmetros do modelo. (FORGHAN,) (MARLOW INDUSTRIES, INC.,).Vale ressaltar que embora seja interessante obter um modelo próximo ao processo real, os parâmetros foram escolhidos sem a realização de medidas da planta. Os valores estão indicados na tabela seguinte:

Tabela 1: Valores dos parâmetros

α_{PN}	$0.000530V/K$
ρ	$0.00001\Omega.m$
L	$0.0025m$
A	$0.00145m$
n	127 junções
k	$1.2W/m.K$
M_C, M_H	$0.05kg$
M_L	$0.7kg$
M_F	$0.6kg$
C_C, C_H	$1000W/kg.K$
C_L	$400W/kg.K$
C_F	$850W/kg.K$
γ	200
R_{th}	$0.25K/W$
T_a	25^oC

3 Simulação

3.1 Discretização do modelo

O método das diferenças finitas consiste em discretizar as equações diferencias parciais do sistema utilizando uma aproximação para os termos derivativos. Este é um método adequado tanto para sistemas lineares ou não-lineares. Considerando que a temperatura $T(x,t)$ é função da distância $x \in [0, L]$ e do tempo $t \in [0, \infty]$, começando dividindo estas duas dimensões em intervalos de comprimentos h_x e h_t respectivamente. A temperatura no ponto $T(x = ih_x, t = jh_t)$ passa a ser representada simplesemente por $T(x,t)$. No método explícito, as derivadas primeiras e segundas são então substituídas por:

$$\frac{\partial T(x,t)}{\partial x} = \frac{T_{i+1,j} - T_{i,j}}{h_x} \tag{3.1}$$

$$\frac{\partial^2 T(x,t)}{\partial x^2} = \frac{T_{i+1,j} + T_{i-1,j} - 2T_{i,j}}{h_x^2} \tag{3.2}$$

O mesmo é feito para as derivadas no tempo:

$$\frac{\partial T(x,t)}{\partial t} = \frac{T_{i,j+1} - T_{i,j}}{h_t} \tag{3.3}$$

As equações a serem discretizadas são repetidas aqui por conveniência. Para o lado frio:

$$(M_L C_L + M_C C_C)\frac{dT_L}{dt} = Q_L + kA_m \frac{dT(x,t)}{dx}\Big|_{x=0} - In\alpha_{PN} T_L \tag{3.4}$$

Cuja discretização fornece:

$$(M_L C_L + M_C C_C)\frac{T_{0,j+1} - T_{0,j}}{h_t} = Q_L + kA_m \frac{T_{1,j} - T_{0,j}}{h_x} - I\alpha_{PN} T_{0,j} \tag{3.5}$$

Para o lado quente:

$$(M_F C_F + M_H C_H)\frac{dT_H}{dt} = -\frac{(T_H - T_a)}{R_{th}} - kA_m \frac{dT(x,t)}{dx}\Big|_{x=L} + In\alpha_{PN} T_H \tag{3.6}$$

Cuja discretização fornece:

$$(M_F C_F + M_H C_H)\frac{T_{N,j+1} - T_{N,j}}{h_t} = -T_{N,j} - T_a R_{th} + kA_m \frac{T_{N,j} - T_{N-1,j}}{h_x} + I\alpha_{PN} T_{N,j} \tag{3.7}$$

E para o interior do módulo:

$$C\gamma\frac{\partial T(x,t)}{\partial t} = k\frac{\partial^2 T(x,t)}{\partial x^2} + \rho\frac{I^2}{A^2} - \mu\frac{I}{A}\frac{\partial T(x,t)}{\partial x} \tag{3.8}$$

Que discretizada se torna:

$$C\gamma\frac{T_{i,j+1} - T_{i,j}}{h_t} = k\frac{T_{i+1,j} + T_{i-1,j} - 2T_{i,j}}{h_x^2} + \frac{\rho I^2}{A^2} - \frac{\mu I}{A}\frac{T_{i+1,j} - T_{i,j}}{h_x} \tag{3.9}$$

Isolando os termos futuros dos termos passados, podemos calcular a cada passo de simulação os valores futuros a partir dos termos passados:

$$T_{i,j+1} = T_{i,j} + \frac{h_t}{C\gamma}\left[k\frac{(T_{i+1,j} + T_{i-1,j} - 2T_{i,j}}{h_x^2} + \frac{\rho I^2}{A^2} - \frac{\mu I}{A}\frac{(T_{i+1,j} - T_{i,j})}{h_x}\right] \tag{3.10}$$

$$T_{0,j+1} = T_{0,j} + \frac{h_t}{M_L C_L + M_C C_C}\left[Q_L + kA_m\frac{T_{1,j} - T_{0,j}}{h_x} - I\alpha_{PN}T_{0,j}\right] \tag{3.11}$$

$$T_{N,j+1} = T_{N,j} + \frac{h_t}{M_F C_F + M_H C_H}\left[-\frac{T_{N,j} - T_a}{R_t h} + kA_m\frac{T_{N,j} - T_{N-1,j}}{h_x} + I\alpha_{PN}T_{N,j}\right] \tag{3.12}$$

3.2 Simulação no Simulink

Para simular tais equações foi utilizado o programa Simulink. O passo de tempo escolhido foi $h_t = 0,01s$ e a expessura do módulo foi discretizada em 6 pontos distantes $h_x = L/5 = 0,001ms$ entre si.

A simulação das equações 3.10 a 3.12 foi feita utilizando o block S-function do Simulink. Este bloco está associado a um arquivo .m com funções que são chamadas na inicialização da simulação e a cada passo de simulação para atualizar os estados e as saídas do processo. A entrada do bloco é a corrente aplicada ao módulo assim como as pertubações (temperatura ambiente e calor fornecido) (Figura 7).

A seguir está a parte do código relativa à função mdlUpdate que calcula os valores futuros dos estados xq(1) a xq(6) utilizando os valores presentes dos estados x(1) a x(6), as entrada e(1) = I, e(2) = Ql e e(3) = Ta. O código completo se encontra nos anexos.

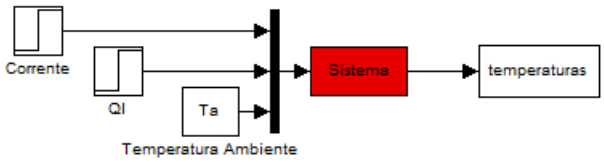

Figura 7: Modelo de simulação no Simulink

```
function sys=mdlUpdate(t,x,e,Ts)

I = e(1);
Ql = e(2);
Ta = e(3);

x1 = x(1);
x2 = x(2);
x3 = x(3);
x4 = x(4);
x5 = x(5);
x6 = x(6);

L = 0.005;
l = L/100;

n = 127;
Am = 0.00145;
A = Am/n;
gama = 200;
C = 700;

alpha = 0.000530;
k = 1.2;
p = 0.00001;
R = 0.25;

Ta = 25+273;

Cc = 500;
Ch = 500;
Cl = 400;
```

```
Cf = 850;

Ml = 0.34;
Mf = 0.6;
Mc = 0.05;
Mh = 0.05;

mc = Ml*Cl+Mc*Cc;
mh = Mf*Cf+Mh*Ch;

hx = 0.001; \%passo de discretizacao da expessura
ht = Ts; \%Timestep do block

xq = x;

\%Calculo das temperaturas nas extremidades (fronteiras)

xq(1) = 1/mc*(Ts*(Ql + k*Am*1/h*(x2-x1) -I*n*alpha*x1) + x1*mc);
xq(6) = 1/mh*(Ts*(-1*(x6-Ta)/R - k*Am*1/h*(x6-x5) +I*n*alpha*x6) + x6*mh)

for i=2:5 \%Calculo das temperaturas no interior
    xq(i) = x(i) + 1/(C*gama)*Ts*(k/h^2*(x(i+1)-2*x(i)+x(i-1))+p*I^2/A^2-
end

sys = [xq(1) xq(2) xq(3) xq(4) xq(5) xq(6)];
```

4 Simulador em C#

4.1 Simulador utilizando a interface OPC

4.1.1 Padrão OPC

A organização da informação em um processo industrial envolve três níveis: o nível de campo, o nível de processo e o nível de negócio. Para uma gerência efetiva do processo, é necessária a integração dos dados provenientes destes três níveis. Desta maneira, é fundamental que as ferramentas utilizadas neste contexto, como os sistemas SCADA, os bancos de dados, as planilhas eletrônicas, etc., possam interagir, trocando informações com o objetivo de consolidar a gerência integrada dos processos industriais. Diante desta necessidade de integração, foi criado o padrão OPC, com o objetivo de fornecer às aplicações um meio comum de acesso aos dados. (JúNIOR, 2005)

Antes do estabelecimento do padrão OPC, a integração entre os diversos sistemas era feita através de drivers de comunicação. No entanto, devido à grande variedade de equipamentos e de softwares, o desenvolvimento e a manutenção dos drivers demandavam cada vez mais esforços. Em resposta a este problema, foi criada a OPC Foundation, cujo objetivo seria desenvolver o padrão OPC, um padrão de acesso a dados aberto, capaz de diminuir a necessidade de drivers proprietários.

Após a criação do padrão OPC, cada fabricante de hardware pode disponibilizar apenas um driver OPC Server para seus equipamentos, enquanto os fornecedores de software utilizam um driver OPC Client, padronizando a comunicação entre os diferentes componentes do processo.

4.1.2 Programa do Simulador

O Laboratório de Instrumentação e Controle vem desenvolvendo um aplicativo para simulação de processos que utiliza o protocolo OPC. A idéia é de criar clientes OPC que executam o código de simulação em tempo-real e que comunicam os valores das variáveis do processo a um servidor OPC central. Este servidor pode ser acessado através da rede local por outros clientes OPC, sejam eles um modelo Simulink ou mesmo um Controlador Lógico Programável real. Desta maneira, o processo pode ser utilizado para o desenvolvimento e teste de leis de controle utilizando o próprio CLP num esquema apelidado Hardware in the Loop.

O mesmo algoritmo anterior desenvolvido para Matlab foi portado para a linguagem de programação C# que foi a liguagem utilizada para desenvolver o aplicativo. Uma

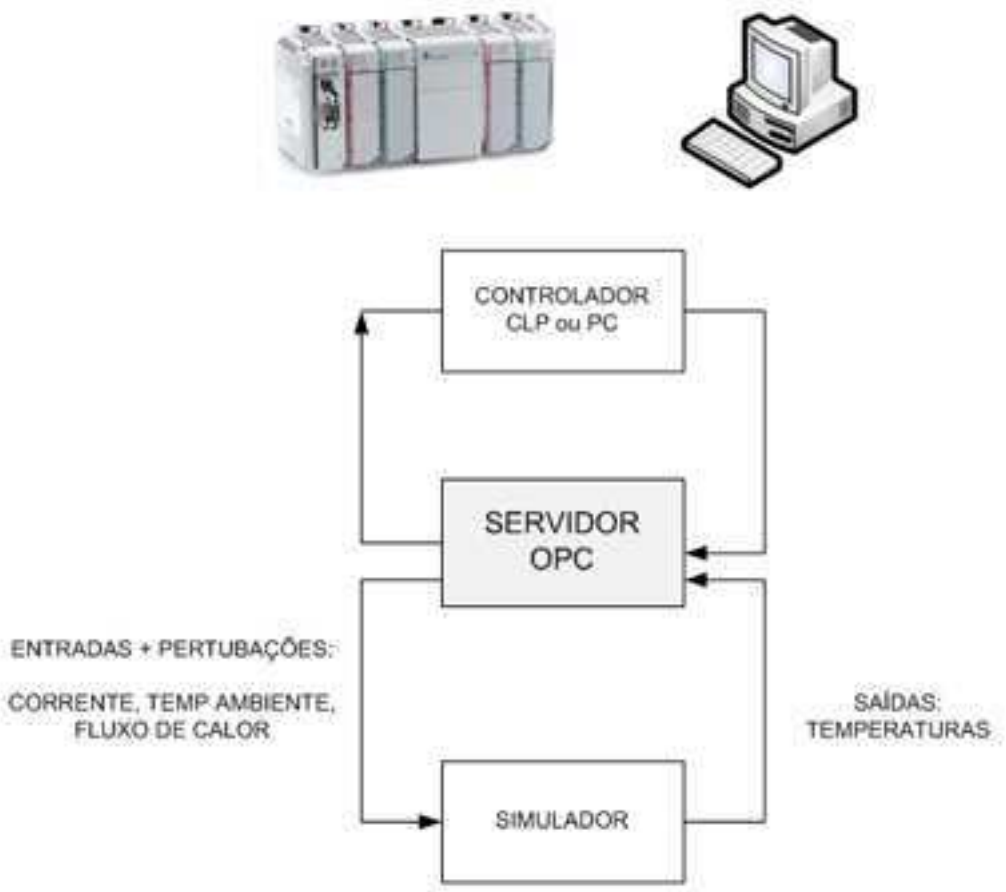

Figura 8: Esquema de Hardware in the Loop

boa referência para esta linguagem é (DEITEL, 2005). A figura 9 mostra a interface do simulador. A interface do simulador OPC é minimalista para manter um alto grau de conformidade e necessita de supervisórios para a visualização adequada de dados. O programa permite carregar diferentes processos na utilizando uma interface padrão para a definição de entradas, saídas e variáveis de estado. Uma vez configurado o processo, é necessário escolher um servidor OPC para hospedar as variáveis que poderão ser acessadas por qualquer outro aplicativo conectado ao servidor OPC pela rede.

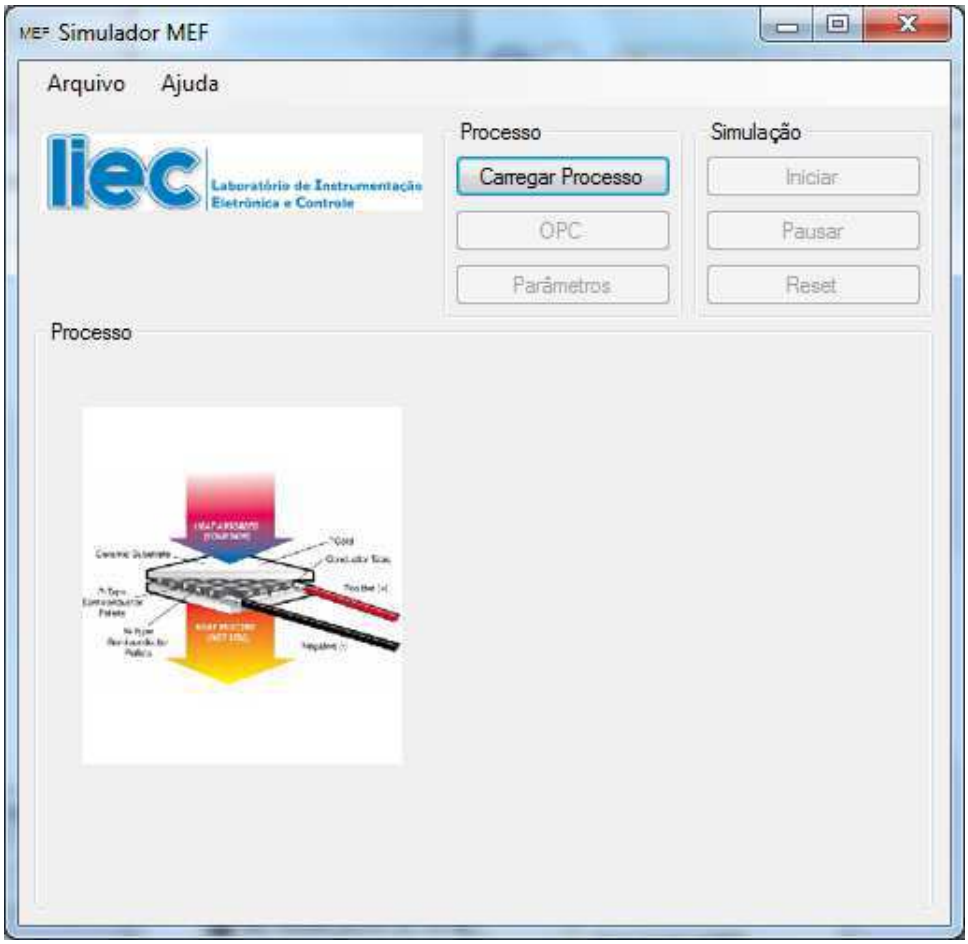

Figura 9: Aplicativo desenvolvido para simulação via OPC

5 Linerarização e identificação

5.1 Linearização do modelo em torno de um ponto de operação

O modelo do efeito peltier é não-linear devido aos termos da forma $nI\alpha_{PN}T$. Para aplicar os métodos de controle usuais faz-se necessário linearizar o modelo. O método escolhido foi o da linearização em torno de um ponto de operação.

Para um sistema não-linear qualquer da forma:

$$\dot{x} = F(x,u) \qquad y = h(x,u)$$

A linearização em torno de um ponto de operação (x_e, u_e) consiste em obter matrizes A, BeC tais que para pequenas variações $\tilde{x} = x - x_e \qquad \tilde{u} = u - u_e$ em torno do ponto de operação:

$$\dot{\tilde{x}} = A\tilde{x} + B\tilde{u} \tag{5.1}$$

Estas matrizes são dadas por:

$$A = \frac{\partial F(x,u)}{\partial x}\bigg|_{u=u_e} \qquad B = \frac{\partial F(x,u)}{\partial u}\bigg|_{x=x_e} \qquad C = \frac{\partial h(x,u)}{\partial x}\bigg|_{u=u_e}$$

Como estamos interessados no controle de temperatura do lado da placa metálica, a variável de estado é $x = T_L$ e a entrada é a corrente $u = I$ a equação a linearizar é a equação 3.4:

$$(M_L C_L + M_C C_C)\frac{dT_L}{dt} = Q_L + kA_m \frac{dT(x,t)}{dx}\bigg|_{x=0} - In\alpha_{PN}T_L \tag{5.2}$$

Para simplificar a linearização, temos que aproximar o termo desconhecido $kA_m\frac{dT(x,t)}{dx}$. A aproximação é feita considerando a distribuição de temperatura ao longo do módulo linear e de modo que o termo é dado por:

$$kA_m \frac{dT(x,t)}{dx}\bigg|_{x=0} = kA_m \frac{(T_H - T_L)}{L} \tag{5.3}$$

Desta forma, $F(x,u)$ é dado por:

$$F(x,u) = \frac{1}{(M_L C_L + M_C C_C)}\left[Q_L + kA_m \frac{(T_H - T_L)}{L} - In\alpha_{PN}T_L\right] \tag{5.4}$$

Assim, os termos obtidos para a linearização são:

$$A = \frac{\partial F(x,u)}{\partial x}\Big|_{I=I_e} = \frac{1}{(M_L C_L + M_C C_C)}\left(-k\frac{A_m}{L} - \alpha_{PN} I_e\right) \tag{5.5}$$

$$B = \frac{\partial F(x,u)}{\partial u}\Big|_{T_L=T_{Le}} = \frac{1}{(M_L C_L + M_C C_C)}\left(-n\alpha_{PN} T_{Le}\right) \tag{5.6}$$

$$C = 1 \tag{5.7}$$

A equação em torno do ponto de operação é então:

$$\dot{\tilde{T}_L} = \frac{1}{(M_L C_L + M_C C_C)}\left(-k\frac{A_m \tilde{T_L}}{L} - n\alpha_{PN}(\tilde{T_L} I_e + T_{Le}\tilde{I})\right) \tag{5.8}$$

Utilizando a transformada de Laplace e considerando um ponto de operação qualquer T_{Le} I_e, esta equação se resume a um sistema de primeira ordem na forma:

$$y(s) = u(s)\frac{K}{1 + T_i s} \tag{5.9}$$

5.2 Identificação da planta

Para comparar o modelo simulado com a planta do processo, foi necessário realizar medidas de temperatura e corrente da planta disponibilizada pelo Laboratório de Instrumentação e Controle da UFCG (Figura 10).

A planta é composta de um módulo de peltier inserido entre a placa metálica e o dissipador que por sua vez está acoplado a ventiladores. A corrente é fornecida por circuitos de alimentação utilizando a técnica PWM. O controle do PWM e a medição de temperaturas é feito por Controladores Lógico Programáveis. Os dados são recuperados através do protocolo OPC que disponibiliza os dados em tempo real em um servidor que pode ser acessado via rede TCP/IP. (Figura 11).

Como o controle é feito em termos de porcentagem do sinal PWM, foi necessário desenvolver um circuito simples de medição de corrente (ver 13). Este foi realizado utilizando um sensor de corrente de efeito Hall que suporta até 20A RMS de corrente RMS e tem até 2500 V de isolação entre o circuito de corrente e o circuito de saída. A saída é composta de um sinal proporcional à corrente medida que foi ligado a uma das entradas analógicas do CLP. A figuras 12 mostra o esquemático do circuito.

Os testes realizados foram o degrau de corrente em torno do ponto de operação $T_{Le} = 298K$ $I_e = 3A$. A figura 14 mostram os dados obtidos para o resfriamento. O modelo obtido está suficientemente próximo à planta neste região de operação.

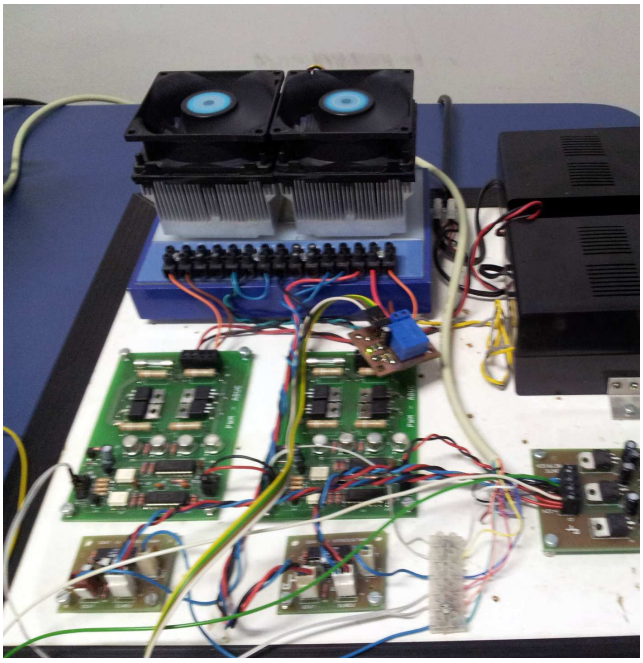

Figura 10: Planta do processo

Os dados obtidos foram utilizados na identifação do processo usando a System Identification Toolbox do Matlab. O modelos utilizados para comparação são de primeira e segunda ordem com atraso:

$$y(s) = u(s)\frac{Ke^{-T_d s}}{1 + T_1 s} \tag{5.10}$$

$$y(s) = u(s)\frac{Ke^{-T_d s}}{(1 + T_1 s)(1 + T_2 s)} \tag{5.11}$$

Os valores de constante obtidos são mostrados nas tabelas seguintes:

Tabela 2: Constantes de primeira ordem

	K	T_1	T_d
Planta	-4,861	329,7	24,37
Modelo	-6,47	288	29,5

Tabela 3: Constantes de segunda ordem

	K	T_1	T_2	T_d
Planta	-4,42	287	42,6	2,5
Modelo	-6,5239	363	244	11.5

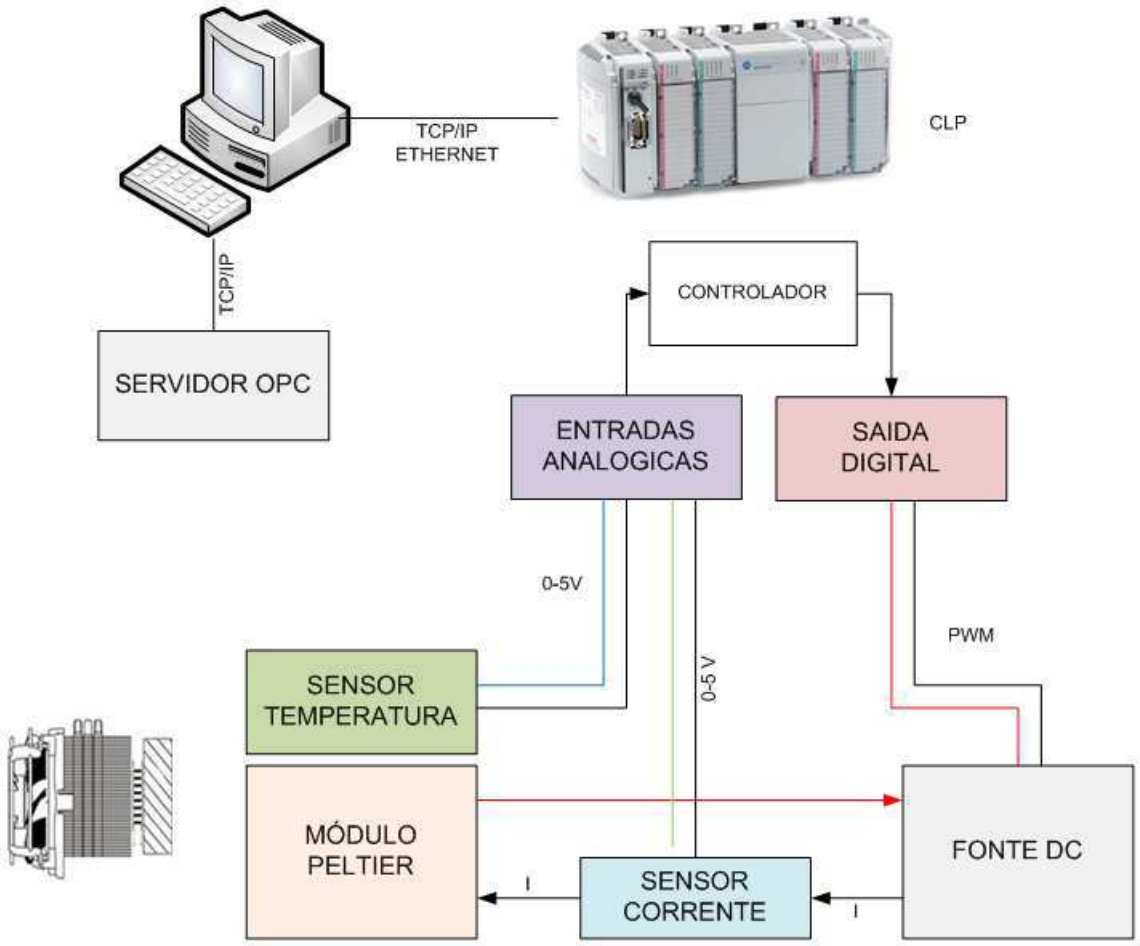

Figura 11: Planta do processo

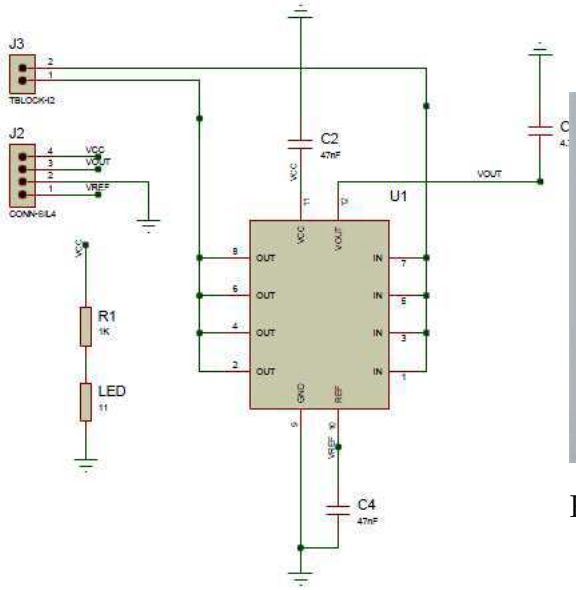

Figura 12: Esquemático

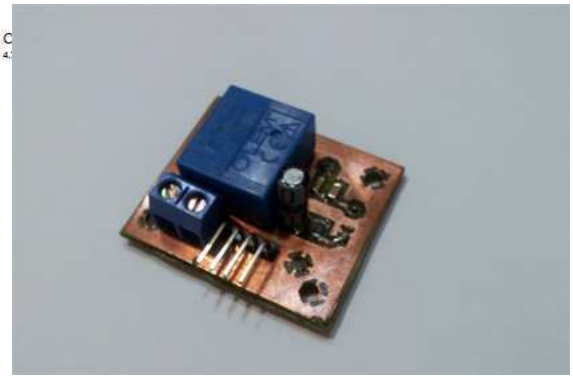

Figura 13: Placa de circuito impresso 2 faces

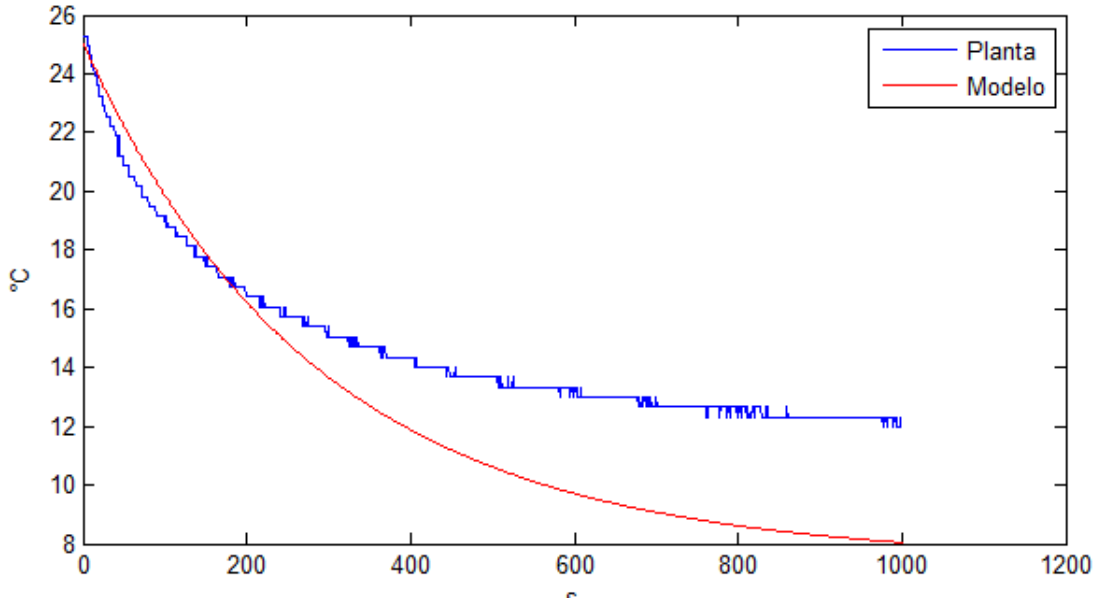

Figura 14: Comparação entre modelo e planta - Resfriamento

6 Controle

Nesta seção serão comparadas duas técnicas de controle: o controle PID utilizando a sintonia SIMC e os controles preditivo DMC (Dynamic Matrix Control) e MAC (Model Algoritmic Control). A variável manipulada é a corrente do módulo e a variável do processo é a temperatura T_L. A temperatura ambiente e o calor fornecido à placa metálica são consideradas pertubações do sistema. Duas boas referências utilizadas nesta parte foram (ÅSTRöM, 1996) e (OGATA, 2003).

Primeiramente, as leis de controle serão testadas no ambiente simulink utilizando o modelo desenvolvido. O mesmo procedimento utilizando o bloco s-function do simulink foi utilizado para implementar estes controladores no modelo.

6.1 Sintonia SIMC

O método de sintonia SIMC considera o modelo de processo da forma:

$$y(s) = u(s)\frac{Ke^{-\theta s}}{(1+T_1 s)(1+T_2 s)} \tag{6.1}$$

e um controlador PID na forma de cascata:

$$c(s) = K_c\frac{T_i s + 1}{T_i s}(T_d s + 1) \tag{6.2}$$

Os parâmetros do controlador PID são escolhidos segundo as equações (**??**):

$$K_c = \frac{1}{K}\frac{T_1}{\theta + T_c} \tag{6.3}$$

$$T_I = min\{T_1, 4(T_c + \theta)\} \tag{6.4}$$

$$T_d = T_2 \tag{6.5}$$

onde T_c é um parâmetro a ser escolhido e que influencia o tempo de resposta e a robustez do sistema em malha-fechada. Utilizando o modelo previamente identificado, foram escolhidos $T_c = \frac{T_1}{2}$ e $T_c = \frac{T_1}{3}$. Os ganhos do controlador são respectivamente:

Os valores obtidos para a margem de ganho e margem de fase para estas configurações são:

Tabela 4: Constantes de segunda ordem

	Kc	T_i	T_d
SIMC1	-3.84	160	30
SIMC2	-2.90	160	30

Tabela 5: Constantes de segunda ordem

	Margem de ganho	Margem de fase
SIMC1	15,5 dB	62 deg
SIMC2	17,9 dB	70 deg

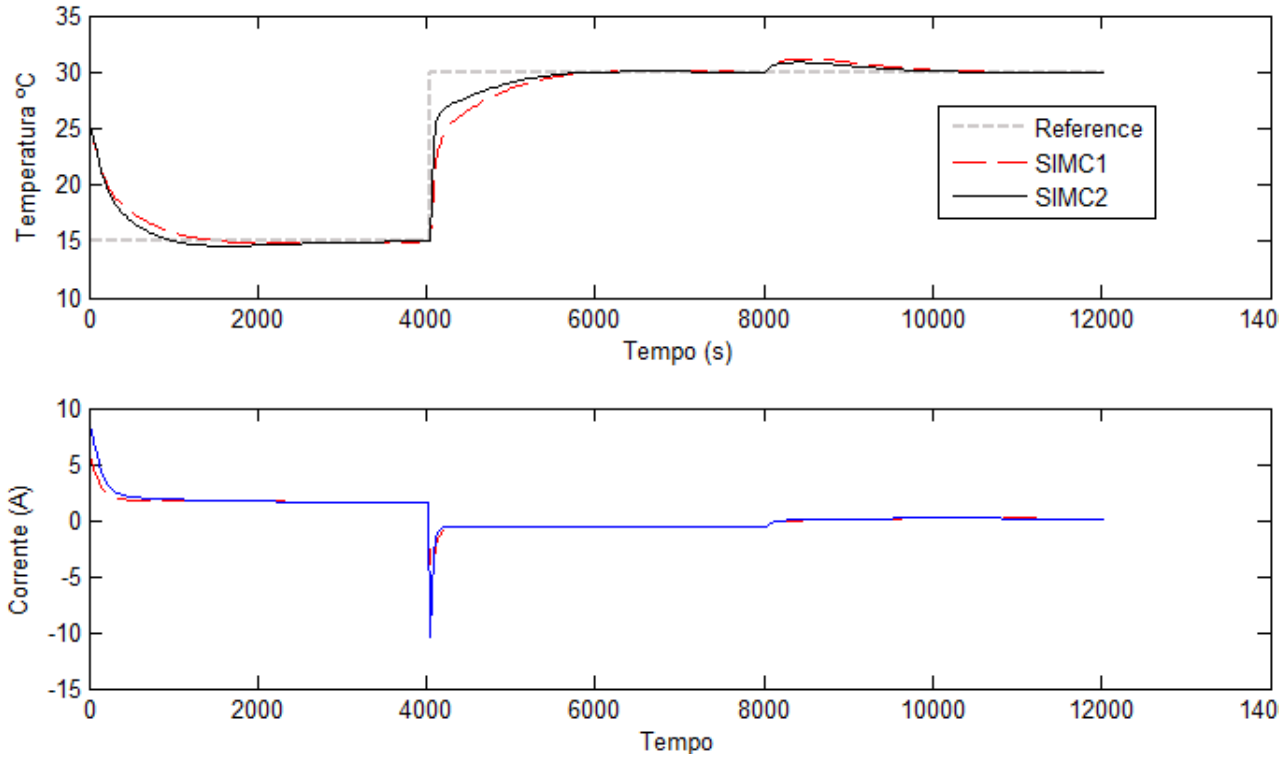

Figura 15: Resposta ao degrau do controle PID SIMC

A figura 15 mostra a resposta à referência em degrau dos dois pids elaborados. No instante t = 8000 é aplicada uma perturbação na forma do calor Q_L fornecido ao módulo. Obervar como a escolha de um T_c menor provoca uma resposta mais rápida.

A planta real também foi controlada utilizando a técnica PID SIMC (Figura 16). Para isto foi necessário reidentificar o sistema tendo agora como entrada a percentagem de PWM e não mais a corrente. Os valores obtidos para a identificação estão mostrados na tabela ??. Os valores das constantes obtidas para o controlador assim como os parâmetros

Figura 16: Planta utilizada para o controle

de robustez estão mostrados nas tabelas ?? e ??. A figura 17 mostra o resultado obtido.

Tabela 6: Constantes de segunda ordem eq planta

	K	T_1	T_2	T_d
Aquecimento	-0.6113	159	30	30
Resfriamento	-0,20	633	45	30

Tabela 7: Robustez do controlador SIMC PID aplicado à planta

	Margem de ganho	Margem de fase
SIMC1	13,9 dB	61,9 deg
SIMC2	17,3 dB	69,6 deg

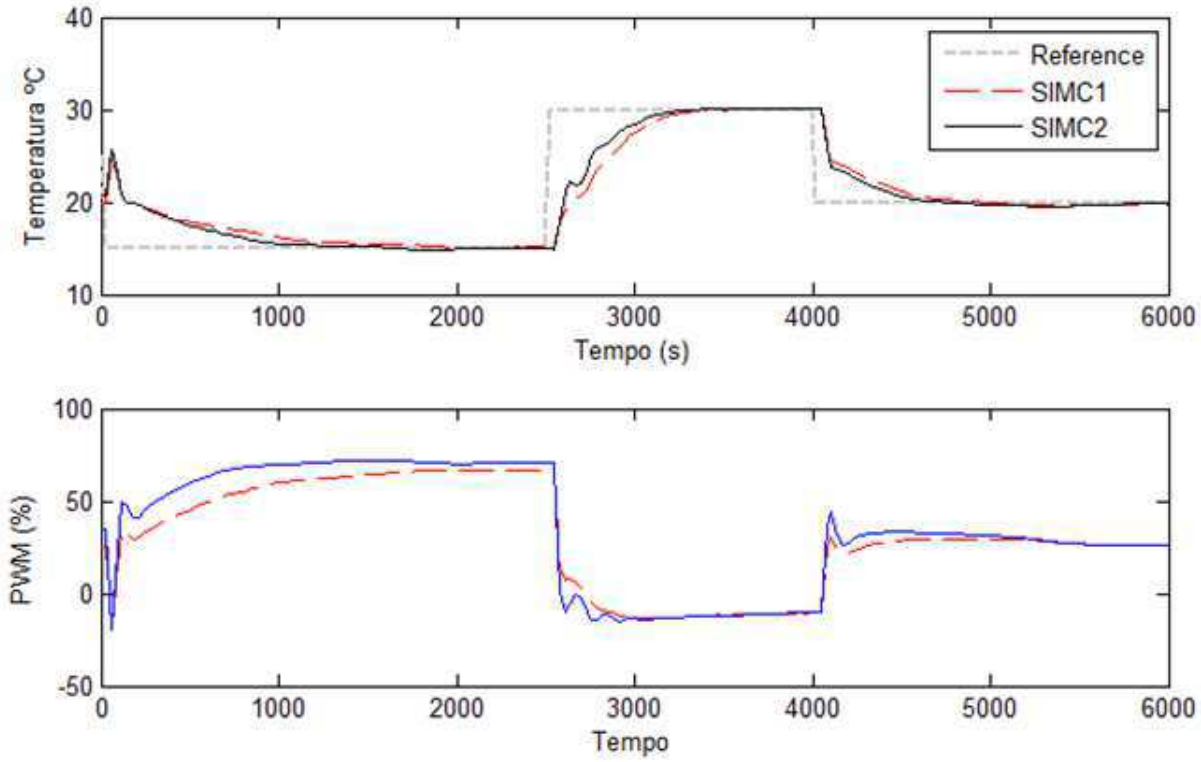

Figura 17: Resultados para o PID SIMC aplicado à planta

Tabela 8: Constantes do controlador SIMC PID aplicado à planta

	Kc	T_i	T_d
SIMC1	-2.90	160	30,3
SIMC2	-3.8399	160	30,3

6.2 Controle Preditivo

6.2.1 Introdução

O controle preditivo é uma técnica que surgiu na década de 70 e tem evoluído desde então. Tem sido bastante utilizado desde o início na indústria petroquímica. Uma des suas vantagens porém é a dificuldade teórica de provar sua estabilidade e robustez. Várias técnicas de controle preditivo existem mas todas consistem em utilizar um modelo do processo para prever a evolução da saída. A ação de controle é calculada de maneira a minimizar um critério energético que leva em consideração o erro da saída assim como o esforço de controle (Figura 18).

A estratégia do horizonte recuante é utilizada: a cada instante, a saíde do sistema é prevista e a ação de controle é calculada para um horizonte de tempo, porém somente

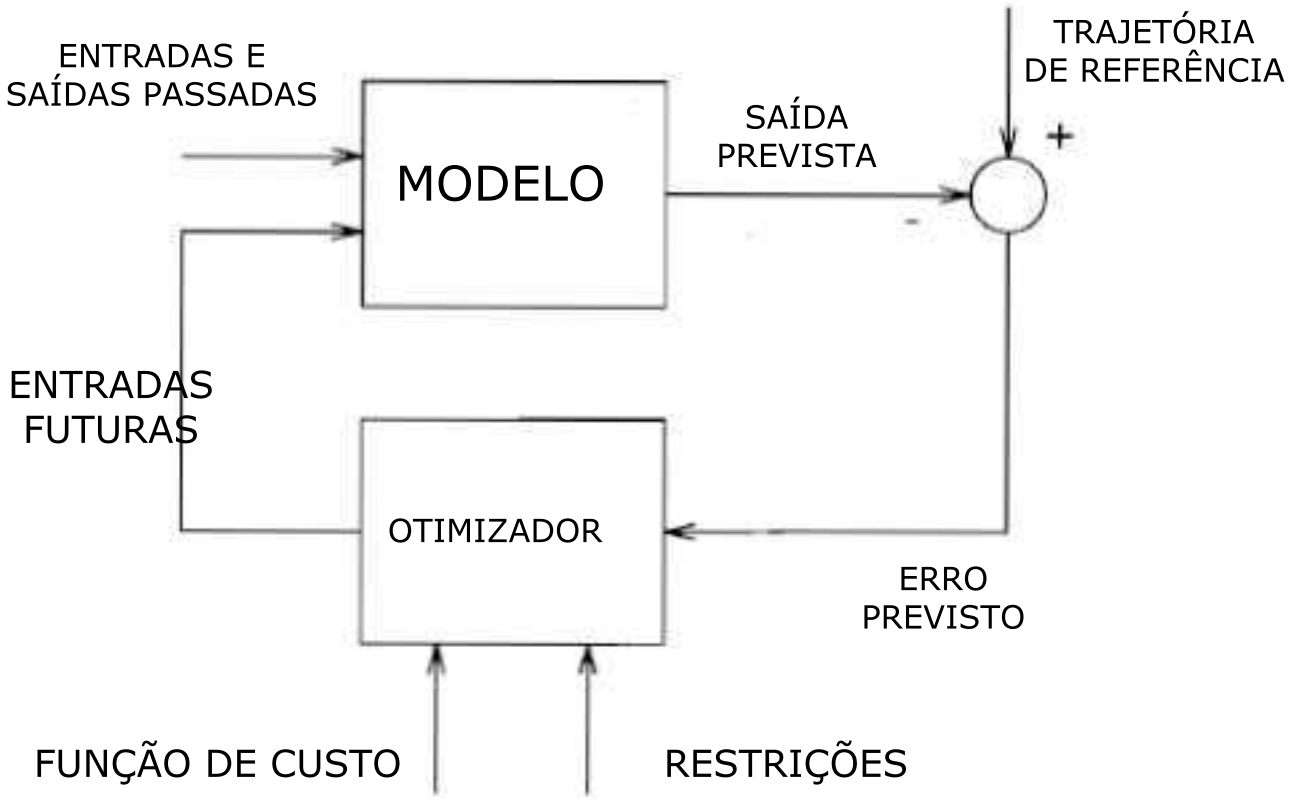

Figura 18: Esquema do controle preditivo

a ação de controle correspondente ao instante seguinte é aplicada ao sistema e o cálculo é refeito no instante seguinte (Figura 19). A referência do sistema é uma sequência aleatória $w(t+k)$. Caso as entradas sejam do tipo degrau, pode-se escolher um $1 > \alpha > 0$ de modo que:

$$w(t+k) = \alpha w(t+k-1) + (1-\alpha)r(t+k) \tag{6.6}$$

onde $r(t+k)$ é o valor da referência. Isto permite controlar a velocidade de convergência da referência gerando respostas mais rápidas ou mais suaves.

A diferença entre o controle preditivo e técnicas de controle tradicionais como o PID pode ser entendida com a analogia do motorista. A cada instante, o motorista conhece a velocidade que deseja manter em um horizonte de tempo próximo e levando em conta seu conhecimento das características do carro, decide as próximas ações de controle a tomar (aceleração, frenagem e direção). Esta é a abordagem do controle preditivo. Analogamente, o controle PID seria equivalente a conduzir o carro não com base na previsão do que vai acontecer mas olhando para erros passados (olhando pelo retrovisor - Figura 20. (CAMACHO, 1996)

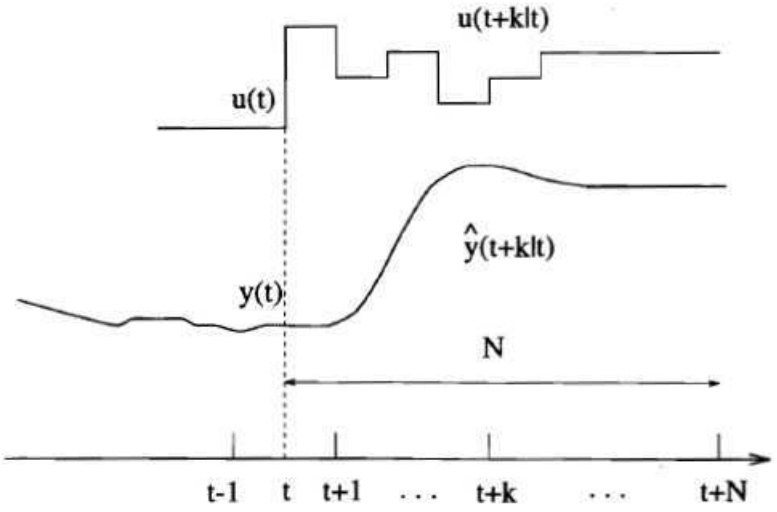

Figura 19: Receding horizon

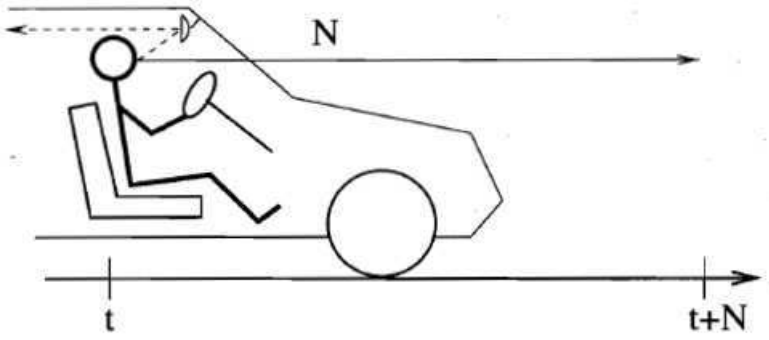

Figura 20: Analogia do motorista

6.2.2 Algoritmo DMC - Dynamic Matrix Control

A técnica de controle preditivo DMC foi desenvolvida no fim da década de 1970 por engenheiros da Shell e têm tido vasta aplicação na industria petroquímica devido a sua simplicidade de implementação e sintonia.

A derivação do algoritmo começa considerando que todo sinal $y(t)$ pode ser representado da seguinte maneira:

$$y(t) = \sum_{i=1}^{\infty} g_i \Delta u(t-i) \tag{6.7}$$

Onde os coeficientes g_i são otidos com a aplicação de uma entrada $u(t)$ em degrau unitário e medindo o valor da saída nos instantes $t+h, t+2h, t+kh$ onde h é o intervalo de amostragem. Utilizando esta representação, o valor previsto da saída no instante t+kh

é dado por:

$$\hat{y}(t+k|t) = \sum_{i=1}^{\infty} g_i \Delta u(t+k-i) + \hat{n}(t+k|t) \tag{6.8}$$

$$= \sum_{i=1}^{k} g_i \Delta u(t+k-i) + \sum_{i=k+1}^{\infty} g_i \Delta u(t+k-i) + \hat{n}(t+k|t) \tag{6.9}$$

Aqui $y(t+k|t)$ representa o valor previsto de $y(t)$ no instante $t+k$ calculado no instante $t+k$. $\hat{n}(t+k|t)$ é o valor previsto da pertubação. Considerando que a pertubação permanece constante, $\hat{n}(t+k|t) = \hat{n}(t|t) = y_m(t) - \hat{y}(t|t)$ onde $y_m(t)$ é o valor medido da saída. Obtemos:

$$\hat{y}(t+k|t) = \sum_{i=1}^{k} g_i \Delta u(t+k-i) + \sum_{i=k+1}^{\infty} g_i \Delta u(t+k-i) + \left(y_m(t) - \sum_{i=1}^{\infty} g_i \Delta u(t-i) \right) = \sum_{i=1}^{k} g_i \Delta u(t+k-i) + f(t \tag{6.10}$$

Com:

$$f(t+k) = y_m(t) + \sum_{i=1}^{\infty} (g_{k+i} - g_i) \Delta u(t-i) \tag{6.11}$$

$f(t+k)$ só depende das entradas passadas do sistema e por isso é denominada resposta livre do sistema considerando que a ação de controle permanecesse constante. Se o sistema é asintóticamente estável temos para um certo $N positivo$

$g_{k+i} - g_i = 0$ para $i > N$

Assim, a resposta livre é dada por:

$$f(t+k) = y_m(t) + \sum_{i=1}^{N} (g_{k+i} - g_i) \Delta u(t-i) \tag{6.12}$$

Os valores previstos de $y(t)$ são dados por:

$$\hat{y}(t+1|t) = g_1 \Delta u(t) + f(t+1) \tag{6.13}$$

$$\hat{y}(t+2|t) = g_2 \Delta u(t) + g_1 \Delta u(t+1) + f(t+2) \tag{6.14}$$

$$\hat{y}(t+p|t) = \sum_{i=1}^{m} g_i \Delta u(t+p-i) + f(t+p) \tag{6.15}$$

Aqui, p é o número de intervalos que compõem o horizonte de predição. Definindo as matrizes:

$$G = \begin{bmatrix} g_1 & 0 & \dots & 0 \\ g_2 & g_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_m & g_{m-1} & \cdots & g_1 \\ \vdots & \vdots & \ddots & \vdots \\ g_p & g_{p-1} & \cdots & g_{p-m+1} \end{bmatrix} u = \begin{bmatrix} \Delta u(t) \\ \Delta u(t+1) \\ \vdots \\ \Delta u(t+p-m) \end{bmatrix} \qquad f = \begin{bmatrix} f(t+1) \\ f(t+2) \\ \vdots \\ f(t+p) \end{bmatrix} \tag{6.16}$$

O valor previsto de y pode ser dado pela seguinte forma matricial:

$$\hat{\mathbf{y}} = \mathbf{G}\mathbf{u} + \mathbf{f} \tag{6.17}$$

O critério que será utilizado para calcular a ação de controle é dado por:

$$J = \sum_{i=1}^{p} \left[\hat{y}(t+j|t) - w(t+j)\right]^2 + \sum_{i=1}^{m} \lambda \left[\Delta u(t+j-1)\right]^2 \tag{6.18}$$

Este critério leva em conta a previsão do erro entre a saída prevista $y(t|t+k)$ e a trajetória da referência $w(t+k)$ assim como o esforço de controle ponderado pela constante λ. Quanto maior o valor de λ maior o custo do controle $u(t)$ e portanto menor o valor das ações de controle calculadas. Na forma matricial:

$$J = (Gu + f - w)^T (Gu + f - w) + \lambda u^T u \tag{6.19}$$

$$= u^T G^T G u - 2u^T G^T (w - f) + (f - w)^T (f - w) + \lambda u^T u \tag{6.20}$$

Para minimizar J calcula-se a derivada com relação a u e iguala-se o resultado a zero:

$$\frac{\partial J}{\partial u} = 2TG^T G u - 2G^T (w - f) + 2\lambda u = 0 \tag{6.21}$$

$$(G^T G + \lambda I)u = G^T (w - f) \tag{6.22}$$

Nestas condições, o vetor de controle é dado por:

$$u = (G^T G + \lambda I)^{-1} G^T (w - f) \tag{6.23}$$

Este algoritmo foi implementado no código Matlab que se encontra em anexo.

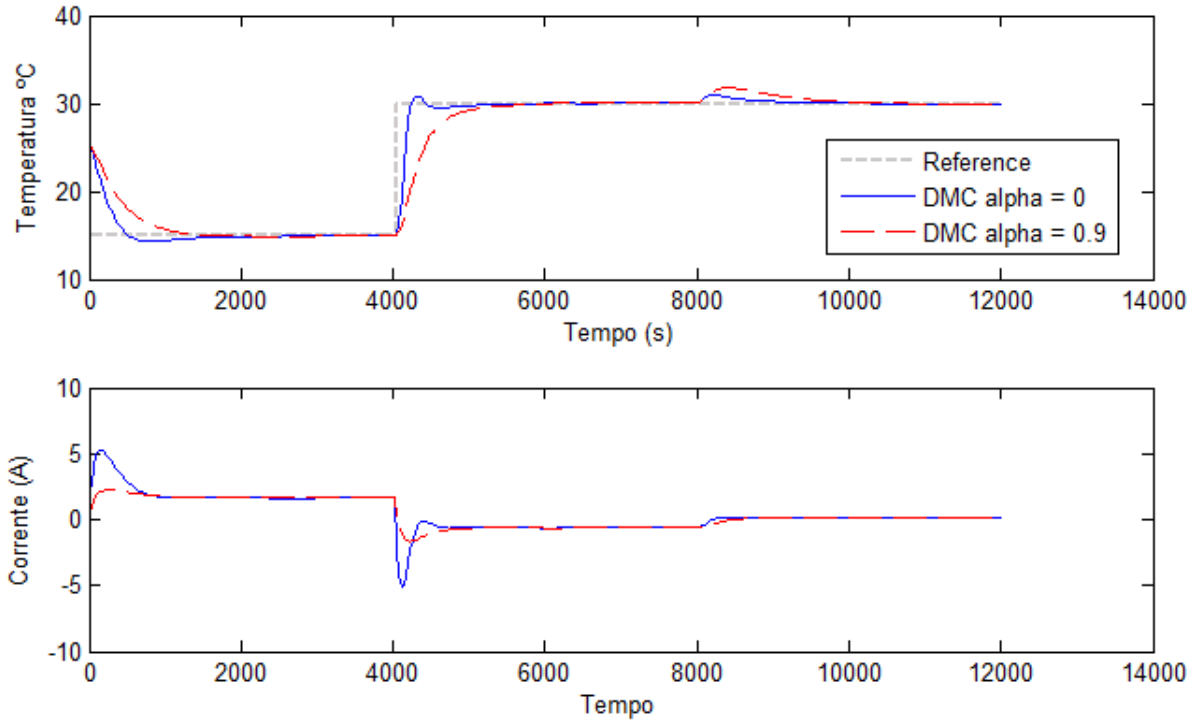

Figura 21: Variação de alpha - Controlador DMC

6.2.3 Parâmetros

Para construir a matrix G foi utilizada a resposta a um degrau de corrente unitário $I = 1A$ com condições iniciais $T_L = T_H = 25C$. Como a constante de tempo de primeira ordem do sistema simulado é de entorno de $500s$, o tempo de amostragem escolhido foi de $h = 30s$.

Para testar a lei de controle, foi utilizado um horizonte de previsão $n = 10$ e um horizonte de controle $m = 5$. A constante $\lambda = 10$ foi escolhida de modo a limitar a corrente na faixa de 0 a $10A$. Correntes maiores provocam aquecimento do módulo e tornam o sistema em malha fechada instável. A resposta ao degrau se estabiliza para um tempo entorno de 1500 s o que nos dá uma matrix G 10x5 facilmente inversível.

A figura 21 mostra como varia a ação de controle com a escolha da constante α.

A planta real também foi controlada utilizando a técnica DMC e os blocos de comunicação OPC do Simulink (Figura 22). Os resultados estão na figura 23.

6.2.4 Model Algoritmic Control - MAC

O método MAC é bastante parecido com o algoritmo DMC, a diferença está basicamente no uso da resposta ao impulso na identificação do sistema ao invés da resposta ao degrau. Tem sido bastante aplicado na industria devido a sua simplicidade de identificação

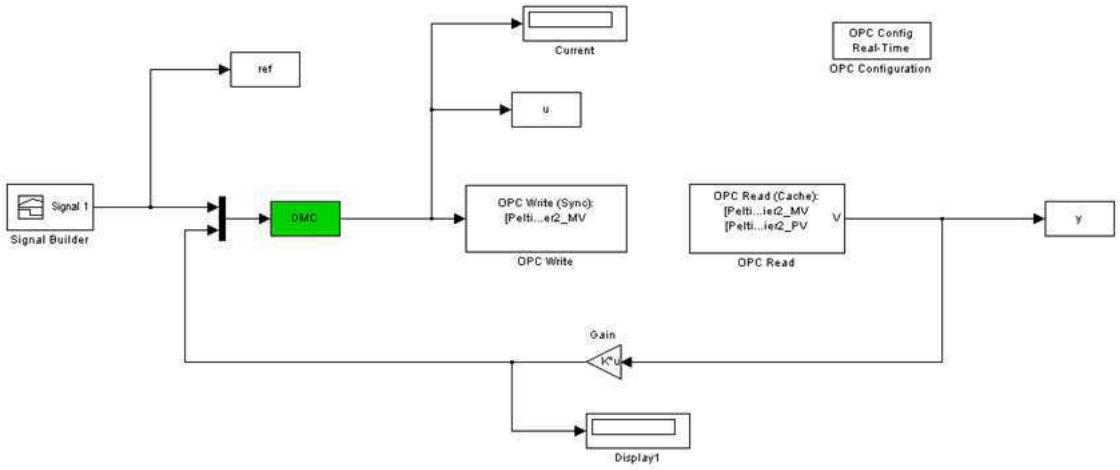

Figura 22: Controle DMC da planta via OPC

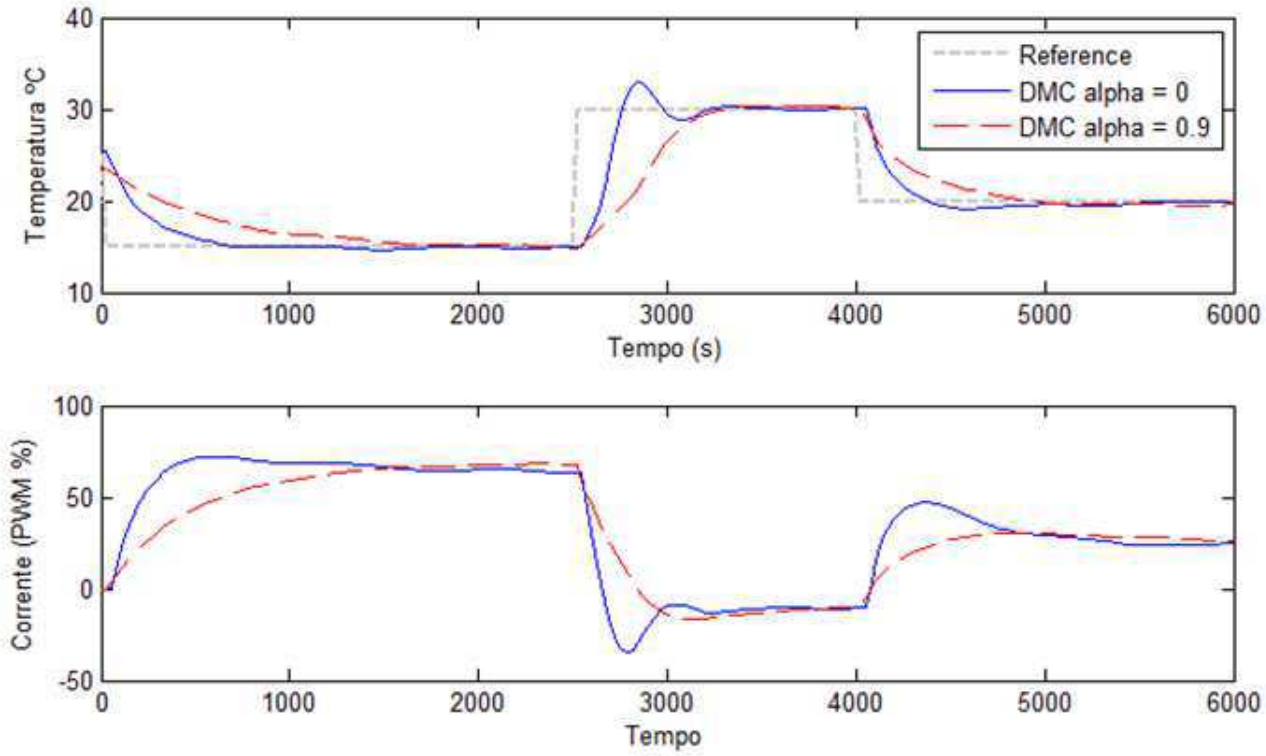

Figura 23: Controle DMC da planta via OPC

e implementação.

Um sistema com resposta ao impulso finita pode ser representado por:

$$y(t) = \sum_{i=1}^{N} h_i u(t-i) \tag{6.24}$$

Utilizando esta representação, a estimação da saída pode ser feita segundo:

$$\hat{y}(t+k|t) = \sum_{i=1}^{N} h_i u(t+k-i) + \hat{n}(t+k|t) \tag{6.25}$$

onde $\hat{n}$ é a pertubação. O somatório pode ser dividido em dois termos:

$$f_r(t+k) = \sum_{i=k+1}^{N} h_i u(t+k-i) \tag{6.26}$$

chamado de resposta livre e dependende somente das entradas passadas e

$$f_o(t+k) = \sum_{i=1}^{k} h_i u(t+k-i) \tag{6.27}$$

chamado de resposta forçada e dependende somente das entradas futuras e presente. Supondo uma pertubação constante temos:

$\hat{n}(t+k|t) = \hat{n}(t|t) = y(t) - \sum_{i=1}^{N} h_i u(t-i)$

Ou seja:

$$\hat{y}(t+k|t) = f_r(t+k) + f_o(t+k) + \hat{n}(t+k|t) \tag{6.28}$$

Definindo as matrizes:

$$u_+ = \begin{bmatrix} u(t) \\ u(t+1) \\ \vdots \\ u(t+M-1) \end{bmatrix} u_- = \begin{bmatrix} u(t-N+1) \\ u(t-N+2) \\ \vdots \\ u(t-1) \end{bmatrix} \quad n = \begin{bmatrix} \hat{n}(t+1) \\ \hat{n}(t+2) \\ \vdots \\ \hat{n}(t+M) \end{bmatrix} w = \begin{bmatrix} w(t+1) \\ w(t+2) \\ \vdots \\ w(t+M) \end{bmatrix} \quad y = \begin{bmatrix} \hat{y}(t+1) \\ \hat{y}(t+2) \\ \vdots \\ \hat{y}(t+M) \end{bmatrix} \tag{6.29}$$

$$H_1 = \begin{bmatrix} h_1 & 0 & \dots & 0 \\ h_2 & h_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ h_M & h_{M-1} & \dots & h_1 \end{bmatrix} H_2 = \begin{bmatrix} h_N & \dots & h_i & \dots & h_2 \\ 0 & \dots & h_j & \dots & h_3 \\ \dots & \ddots & \dots & \dots & \dots \\ 0 & \dots & h_N & \dots & h_{M+1} \end{bmatrix} \tag{6.30}$$

A saída estimada pode ser dada por:

$$y = H_1 u_+ + H_2 u- + n \tag{6.31}$$

Assim como para o algoritmo DMC é definido um critério energético função do erro e do esforço de controle dado por:

$$J = (w - y)^T (w - y) + \lambda u_+^T u_+ \tag{6.32}$$

Ou ainda:

$$J = (w - f - H_1 u_+)^T (w - f - H_1 u_+) + \lambda u_+^T u_+ \tag{6.33}$$

O critério é minimizado derivando em função do termo u_+

$$\frac{\partial J}{\partial u_+} = 2H_1^T H_1 u_+ - 2H_1^T (w - f) + 2\lambda u_+ = 0 \tag{6.34}$$

O que dá:

$$(H_1^T + H_1 + \lambda I) u_+ = H_1^T (w - f) \tag{6.35}$$

Ou finalmente, a lei de controle:

$$u_+ = (H_1^T + H_1 + \lambda I)^{-1} H_1^T (w - f) \tag{6.36}$$

Este algoritmo foi implementado no código Matlab que se encontra em anexo.

6.2.5 Parâmetros

Para construir as matrizes H_1 e H_2 foi utilizada a resposta a um pulso de corrente unitário $I = 1A$ com condições iniciais $T_L = T_H = 25C$. O tempo de amostragem foi novamente escolhido como $h = 30s$.

Para testar a lei de controle, foi utilizado um horizonte de previsão $m = 30$. A constante $\lambda = 1$ foi escolhida de modo a limitar a corrente na faixa de 0 a $10A$. Correntes maiores provocam aquecimento do módulo e tornam o sistema em malha fechada instável. A resposta ao pulso se estabiliza para um tempo entorno de 2000 s o que nos dá matrizes $H_1 30x30$ e $H_2 30x65$ facilmente inversíveis. A figura 24 mostra como varia a ação de controle com a escolha da constante α.

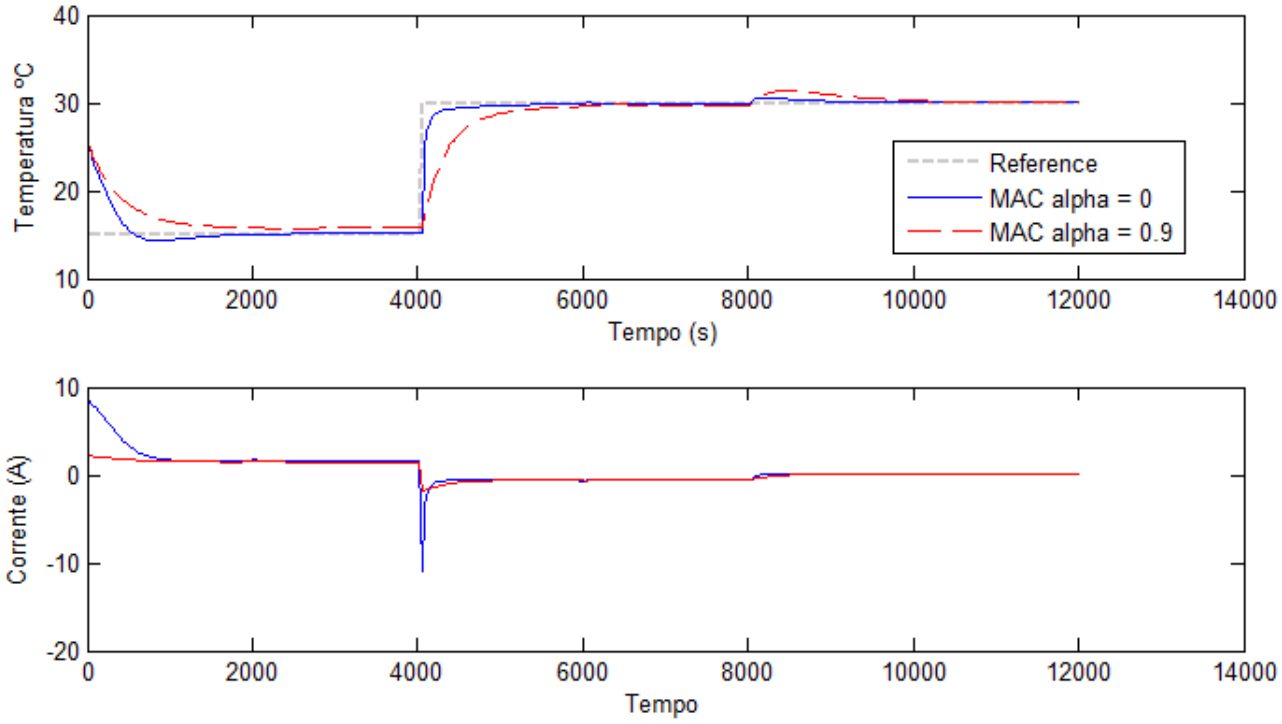

Figura 24: Variação de alpha - Controlador MAC

6.3 Resultados

A figura 25 mostra a comparação entre os controles implementados. Uma pertubação na forma de calor fornecido ao lado frio de 2W é aplicada no instante de tempo 300. Pode-se ver que para este processo os controladores preditivos apresentam um menor tempo de subida que aquele propcionado pelo controlador PID sintonizado com a regra SIMC.

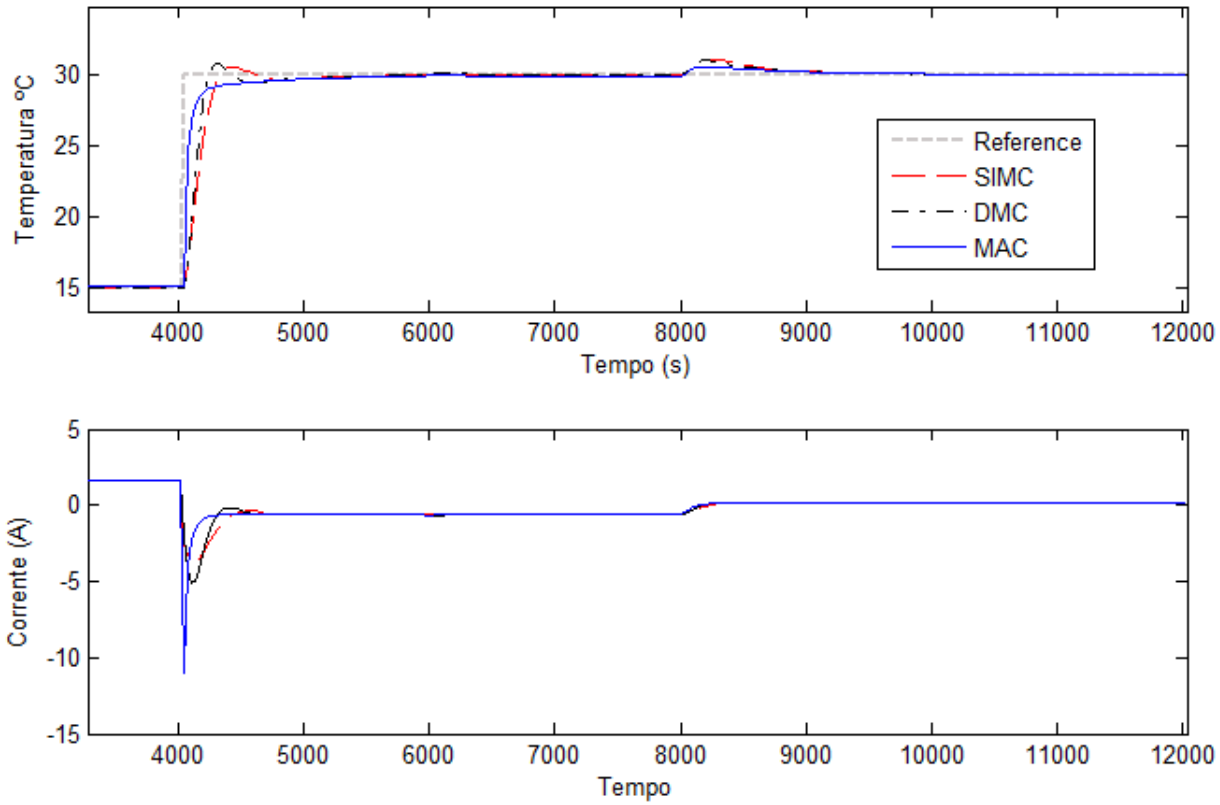

Figura 25: Comparação entre os controladores obtidos

7 Conclusão

Este trabalho de conclusão de curso consistiu na modelagem de um módulo termoelétrico de Peltier utilizando as equações diferenciais parciais não-lineares que regem o seu funcionamento. Foi mostrado que estas equações podem ser simuladas em forma discretizadas utilizando a técnica das diferenças finitas.

Mesmo em se tratando de um modelo não-linear, mostrou-se que a concepção de leis de controle utilizando um modelo linear em torno de um ponto de operação rende resultados satisfatório. Foi feita uma breve introdução dos métodos de controle preditivos. A aplicação dos algoritmos DMC e MAC mostrou que estes podem ter um menor tempo de resposta com relação ao método PID SIMC, embora a análise de estabilidade e robustez seja demasiadamente complexa e não foi abordada.

Um trabalho futuro consiste em aplicar os controladores desenvolvidos à planta real utilizando a interface OPC.

Referências

ALVES, E. Propriedades físicas do semicondutor bi2te3. 2007. 17

CAMACHO, E. F. *Model Predictive Control*. 1st. ed. London: Springer, 1996. 39

DEITEL, P. J. *Visual C sharp 2005 How to Program*. 2nd. ed. [S.l.]: Pearson, 2005. 28

FORGHAN, F. Experimental and theoretical investigation of thermal performance of heat sinks. 22

HUANG, B.; DUANG, C. System dynamic model and temperature control of a thermoelectric cooler. *International Journal of Refrigeration*, v. 23, n. 1, p. 197–207, 2000. 16, 18

JúNIOR, G. A. Avaliação do padrão opc para automação utilizando dispositivos móveis. 2005. 27

MARLOW INDUSTRIES, INC. *TECHNICAL DATA SHEET RC12-2.5 Single-Stage Thermoelectric Cooler*. USA. 22

OGATA, K. *Engenharia de controle moderno*. 4th. ed. São Paulo: Prentice Hall, 2003. 35

ÅSTRöM, K. J. *Computer-Controlled Systems: Theory and Design*. 3rd. ed. [S.l.]: Prentice Hall, 1996. 35

8 Anexos

8.1 Código de simulação do sistema no Simulink

```
function [sys,x0,str,ts] = Sistema(t,x,e,flag,x0,Ts)
\%
\% S function para simulaxOo do modulo de peltier :
\%
\%
\% Entradas :
\%   e(1) = i
\%   e(2) = Ql
\%   e(3) = Ta
\%
\% Estados :
\%   x(1) a x(6) = Temperaturas (Kelvin)
\%
\% Saidas :
\%   x(1) a x(6) = Temperaturas (Kelvin)
\%
\% Parametros passados na inicializacao :
\%  x0 : Temperatura inicial do modulo

switch flag,

  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  \% Initialization \%
  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(x0,Ts);

  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  \% Derivatives \%
  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  case 2,
  sys=mdlUpdate(t,x,e,Ts);
```

```
  \%\%\%\%\%\%\%\%\%\%\%\%
  \% Outputs \%
  \%\%\%\%\%\%\%\%\%\%\%\%
  case 3,
    sys=mdlOutputs(t,x,e);

  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  \% Unhandled flags \%
  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  case { 1, 4, 9 },
    sys = [];

  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  \% Unexpected flags \%
  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);

end
\% end csfunc

\%
\%=============================================================================
\% mdlInitializeSizes
\% Return the sizes, initial conditions, and sample times for the S-funct
\%=============================================================================
\%
function [sys,x0,str,ts]=mdlInitializeSizes(x0,Ts)

sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 6;
sizes.NumOutputs     = 6;
sizes.NumInputs      = 3;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
x0  = [x0] ;              \% Valor inicial do estado
```

```
str = [];
ts  = [Ts 0];

\% end mdlInitializeSizes
\%
\%=============================================================================
\% mdlDerivatives
\% Return the derivatives for the continuous states.
\%=============================================================================
\%

function sys=mdlUpdate(t,x,e,Ts)

I = e(1);
Ql = e(2);
Ta = e(3);

x1 = x(1);
x2 = x(2);
x3 = x(3);
x4 = x(4);
x5 = x(5);
x6 = x(6);

L = 0.005;
l = L/100;

n = 127;
Am = 0.00145;
A = Am/n;
gama = 200;
C = 700;

alpha = 0.000530;
k = 1.2;
p = 0.00001;
R = 0.25;

Ta = 25+273;
```

```
Cc = 1000;
Ch = 1000;
Cl = 400;
Cf = 850;

Ml = 0.7;
Mf = 0.6;
Mc = 0.05;
Mh = 0.05;

mc = Ml*Cl+Mc*Cc;
mh = Mf*Cf+Mh*Ch;

hx = 0.001; \%passo de discretizacao da expessura
ht = Ts; \%Timestep do block

Ql = Ql + (Ta-x(1))/(R);

xq = x;

\%Calculo das temperaturas nas extremidades (fronteiras)

xq(1) = 1/mc*(Ts*(Ql + k*Am*1/h*(x2-x1) -I*n*alpha*x1) + x1*mc);
xq(6) = 1/mh*(Ts*(-1*(x6-Ta)/R - k*Am*1/h*(x6-x5) +I*n*alpha*x6) + x6*mh)

for i=2:5 \%Calculo das temperaturas no interior
    xq(i) = x(i) + 1/(C*gama)*Ts*(k/h^2*(x(i+1)-2*x(i)+x(i-1))+p*I^2/A^2-
end

sys = [xq(1) xq(2) xq(3) xq(4) xq(5) xq(6)];

\% end mdlDerivatives
\%
\%=============================================================================
\% mdlOutputs
\% Retorna as saidas.
\%=============================================================================
\%
```

```
function sys=mdlOutputs(t,x,e)

sys = [x(1) x(2) x(3) x(4) x(5) x(6)];

\% end mdlOutputs
```

8.2 Código de simulação do controle DMC no Simulink

```
function [sys,x0,str,ts] = DMC(t,x,e,flag,Ts,G_,gi_)

global G;
global uvec;
global yvec;
global deltau;
global gi;
global ym;
global f;

switch flag,

  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  \% Initialization \%
  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(Ts);



    G = G_;
    gi = gi_;

    deltau = zeros(length(gi),1);
    uvec = zeros(length(gi),1);
    f = zeros(length(G),1);

  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  \% Derivatives \%
  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  case 2,
    sys=mdlUpdate(t,x,e,Ts);
```

```
  \%\%\%\%\%\%\%\%\%\%\%
  \% Outputs \%
  \%\%\%\%\%\%\%\%\%\%\%
  case 3,
    sys=mdlOutputs(t,x,e);

  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  \% Unhandled flags \%
  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  case { 1, 4, 9 },
    sys = [];

  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  \% Unexpected flags \%
  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);

end
\% end csfunc

\%
\%=============================================================================
\% mdlInitializeSizes
\% Return the sizes, initial conditions, and sample times for the S-funct
\%=============================================================================
\%
function [sys,x0,str,ts]=mdlInitializeSizes(Ts)

sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 1;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
x0  = zeros(1,sizes.NumDiscStates) ;
```

```
str = [];
ts  = [Ts 0];

\% end mdlInitializeSizes
\%
\%=============================================================================

\%=============================================================================
\%
function sys=mdlUpdate(t,x,e,Ts)



sys = [];

\% end mdlDerivatives
\%
\%=============================================================================
\% mdlOutputs
\% Return the block outputs.
\%=============================================================================
\%
function sys=mdlOutputs(t,x,e)


global G;
global uvec;
global yvec;
global deltau;
global gi;
global ym;
global f;

y_ref = e(1);
y_mes  = e(2);

for k =1:length(f)
```

```
    sum = 0;

    for i=1:length(gi)
        if(k+i > length(gi))
            sum = sum + (gi(length(gi))-gi(i))*deltau(i);
        else
            sum = sum + (gi(k+i)-gi(i))*deltau(i);
        end
    end
    f(k) = y_mes + sum;
end

\% weights = 20*ones(length(G)-1,1);
\% lambdas = diag([1; weights]);

lambda = 0.5;
alpha = 0;

w = zeros(length(f),1);

w(1) = y_mes;
w(1) = alpha*w(1)+(1-alpha)*y_ref;

for k =2:(length(w))
    w(k) = alpha*w(k-1)+(1-alpha)*y_ref;
end

u = (G'*G+lambda*eye(min(size(G))))^(-1)*G'*(w-f);
deltauaction = u(1);
uaction = deltauaction + uvec(1);

if(uaction < -100)
    uaction = -100;
end
if(uaction > 100)
    uaction = 100;
end

yobs = G*u+f;
```

```
for k =fliplr(2:length(uvec))
    uvec(k) = uvec(k-1);
end

uvec(1) = uaction;

for k =1:length(deltau)-1
    deltau(k) = uvec(k)-uvec(k+1);
end

output = 50 - uaction /2;

sys = [ output ];

\% end mdlOutputs
```

8.3 Código de simulação do controle MAC no Simulink

```
function [sys,x0,str,ts] = Commande(t,x,e,flag,Ts,H1_,H2_,h_)

global H1;
global H2;
global h;
global u_;
global n;
global f;
global m;

switch flag,

  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  \% Initialization \%
  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(Ts);

    H1 = H1_;
    H2 = H2_;
    h = h_;
```

```
    m = length(H1);
    u_ = zeros(length(h),1);
    n = zeros(m,1)
    f = zeros(length(h),1);

  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  \% Derivatives \%
  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  case 2,
    sys=mdlUpdate(t,x,e,Ts);

  \%\%\%\%\%\%\%\%\%\%\%\%
  \% Outputs \%
  \%\%\%\%\%\%\%\%\%\%\%\%
  case 3,
    sys=mdlOutputs(t,x,e);

  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  \% Unhandled flags \%
  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  case { 1, 4, 9 },
    sys = [];

  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  \% Unexpected flags \%
  \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);

end
\% end csfunc

\%
\%=============================================================================
\% mdlInitializeSizes
\% Return the sizes, initial conditions, and sample times for the S-funct
\%=============================================================================
\%
function [sys,x0,str,ts]=mdlInitializeSizes(Ts)
```

```
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 1;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
x0  = zeros(1,sizes.NumDiscStates) ;
str = [];
ts  = [Ts 0];

\% end mdlInitializeSizes
\%
\%=============================================================================

\%=============================================================================
\%

function sys=mdlUpdate(t,x,e,Ts)

sys = [];

\% end mdlDerivatives
\%
\%=============================================================================
\% mdlOutputs
\% Return the block outputs.
\%=============================================================================
\%
function sys=mdlOutputs(t,x,e)



global H1;
global H2;
global h;
global n;
```

```
global m;
global u_;
global f;

y_ref = e(1);
y_mes  = e(2);


for k =1:m
    sum = 0;
    for j=1:length(h)
        sum = sum + h(j)*u_(length(h)+1-j);
    end

    n(k) = y_mes - sum;
end

f = H2*u_(2:end) + n;

lambda = 0.01;
alpha = 0;

w = zeros(length(f),1);

w(1) = y_mes;
w(1) = alpha*w(1)+(1-alpha)*y_ref;

for k =2:length(w)
    w(k) = alpha*w(k-1)+(1-alpha)*y_ref;
end

u =(H1'*H1+lambda*eye(length(H1)))^-1*H1'*(w-f);

ycal = H1*u+f;

uaction = u(1);

if(uaction < -100)
    uaction = -100;
```

```
end
if(uaction > 100)
    uaction = 100;
end

for k = 1:(length(u_)-1)
    u_(k) = u_(k+1);
end

output = 50 - uaction /2;

u_(end) = uaction;

sys = [ output ];

\% end mdlOutputs
```