Centro Federal de Educação Tecnológica de Minas Gerais
Departamento de Computação
BACHAREL EM ENGENHARIA DE COMPUTAÇÃO

Laboratório de Controle Digital de Sistemas Dinâmicos

Trabalho final

Alunos: Jônatas R. Tonholo

Professor: Tales Argolo Jesus

Belo Horizonte

julho de 2015

Sumário

1 Resumo

Neste trabalho realizaremos controle digital no sistema da Figura 1 – Sistema Capacitivo de Segunda Ordem.

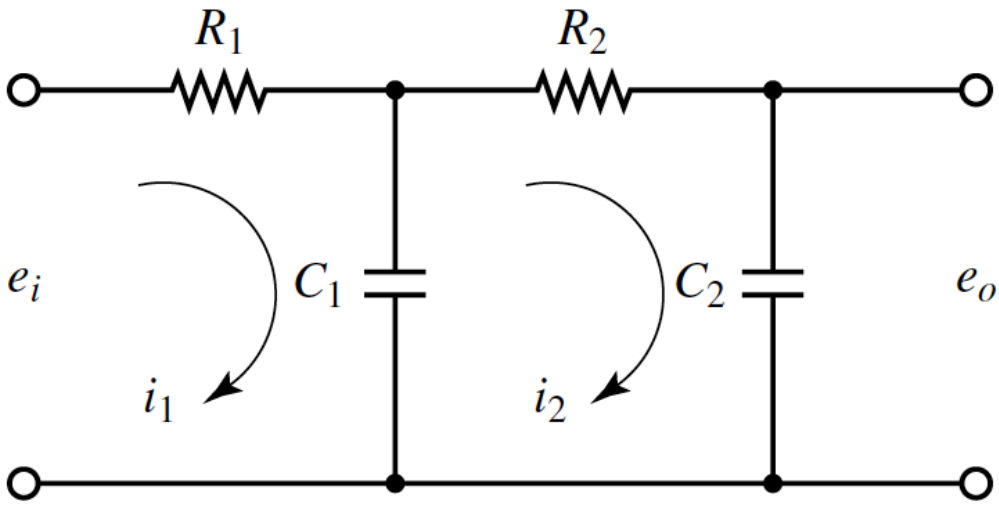


Figura 1 – Sistema Capacitivo de Segunda Ordem

Por possuir dois capacitores, o sistema é de segunda ordem.

2 Resultados

Representação em espaço de estados

Considerando $R_1 = R_2 = 100K\Omega$ e $C_1 = C_2 = 220nF$, utilizando a lei de kirchhoff chegou-se às equações *(i) e (ii).*

$$V_{c_1} = \frac{V_s - V_{c_2}}{RCs + 2} \qquad (i)$$

$$V_{c_2} = \frac{V_{c_1}}{RCs + 1} \qquad (ii)$$

A partir das equações *(i) e (ii) obteve-se as equações diferenciais (iii) e (iv)*

$$\dot{V}_{c_1}(t) = \frac{-2V_{c_1}(t) + V_{c_2}(t) + V_s(t)}{RC} \qquad (iii)$$

$$\dot{V}_{c_2}(t) = \frac{-V_{c_2}(t) + V_{c_1}(t)}{RC} \qquad (iv)$$

Com estes dados em mãos, obteve-se a representação em espaço de estados representados pelas equações (v) e (vi), onde *y(t)* é a saída do sistema, e $V_s(t)$ a entrada.

$$\begin{bmatrix}\dot{V}_{c_1}(t)\\ \dot{V}_{c_2}(t)\end{bmatrix} = \frac{1}{RC}\begin{bmatrix}-2 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}V_{c_1}(t)\\ V_{c_2}(t)\end{bmatrix} + \frac{1}{RC}V_s(t) \qquad (v)$$

$$y(t) = \begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}V_{c_1}(t)\\ V_{c_2}(t)\end{bmatrix} + [0]V_s(t) \qquad (vi)$$

Função de Transferência e Polos

Em posse da **Representação em espaço de estados** obtida, utilizando-se o Matlab® e a equação (vii), obteve-se a função de transferência de malha aberta do sistema (viii).

$$C(sI - A)^{-1}B + D \qquad (vii)$$

$$G(s) = \frac{2066}{s^2 + 136{,}4s + 2066} \qquad (viii)$$

Com polos em s_1 = -119,0015 e s_2 = -17,321

Simulação da resposta do sistema à uma entrada degrau e comparação com os dados obtidos experimentalmente

A Figura 2 nos mostra a resposta ao degrau de 5 volts para o sistema simulado e para o sistema amostrado.

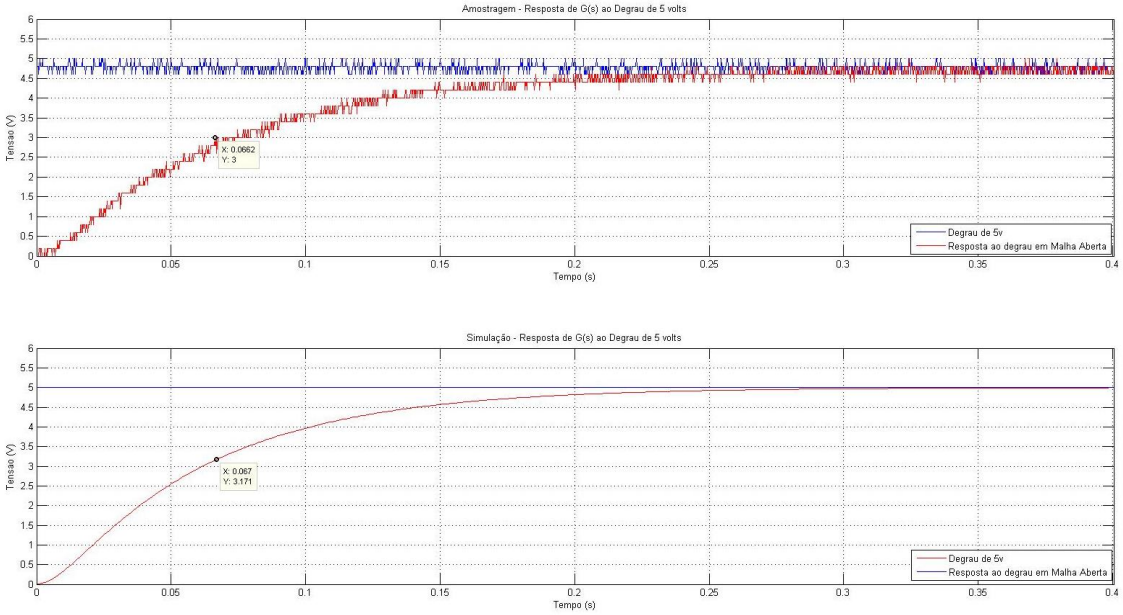


Figura 2 – Resposta ao degrau de 5 volts - Simulação e Amostragem

Em ambas as situações, a constante de tempo τ foi aproximadamente 0.66 segundos. Pode-se observar também que pode-se considerar que o sistema em malha aberta possui erro de posição em estado estacionário quase nulo.

Projeto de um controlador por alocação de Polos

Foi pedido que se projetasse um controlador por alocação de polos que tornasse o sistema oscilatório. A Figura 3 nos mostra o lugar das raízes de G(s), e a região de estabilidade em malha fechada.

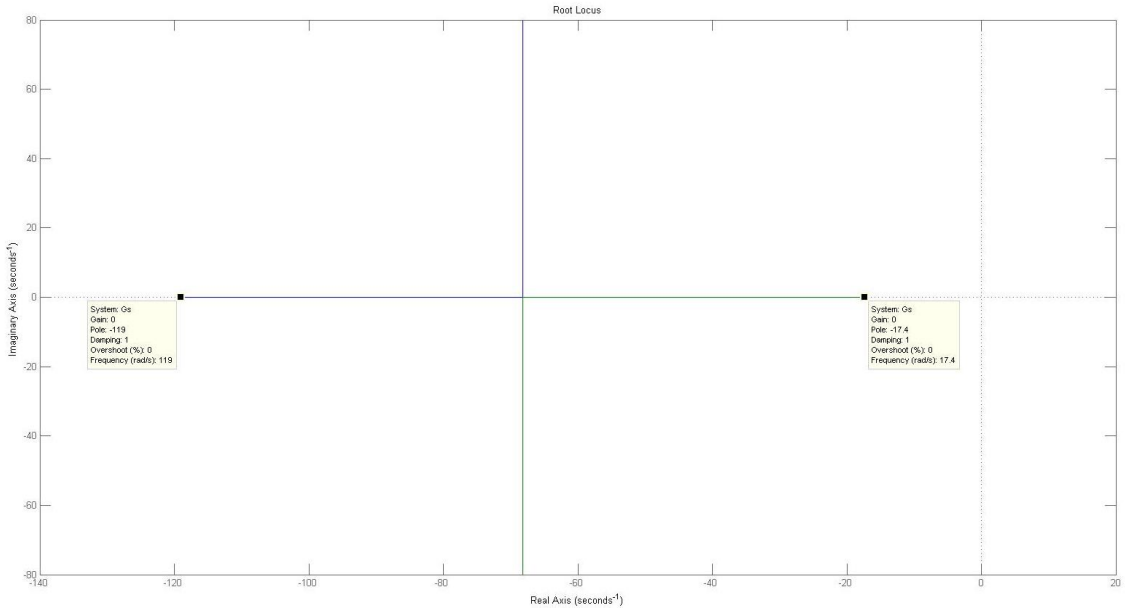


Figura 3 - Lugar das Raízes para G(s)

Para tornar o sistema oscilatório, mas estável, bastava introduzir no sistema um par de polos complexos conjugados com parte real negativa. Entretanto, é necessário observar que é interessante manter a ordem do sistema, portanto, como estratégia, anulamos os dois polos de G(s) e introduzimos novos polos. A Equação (ix) nos dá o controlador por alocação de polos:

$$K(s) = \frac{(s + 17.3621)(s + 119.0015)}{(s + 17.3621 + j50)(s + 17.3621 - j50)} \qquad (ix)$$

E por consequência, temos agora a seguinte configuração para o sistema mostrado na Figura

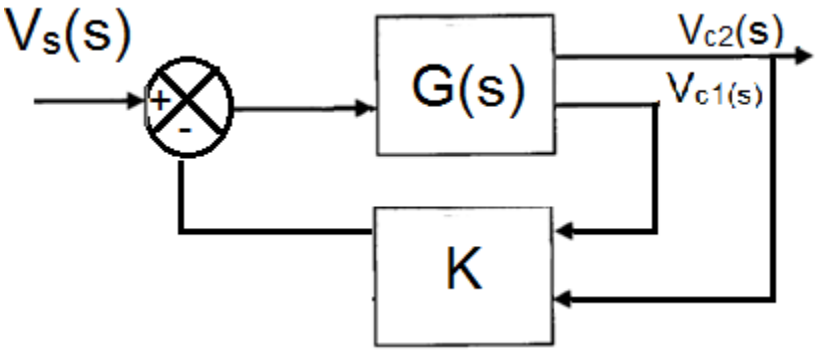


Figura 4 - Controlador Proporcional Por alocação de Polos

O sistema está neste formato pois o requisito era provocar uma oscilação em um sistema que era sobre amortecido. Mais adiante, introduziremos um controlador digital para corrigir o erro de posição no estado estacionário do sistema e a oscilação provocada.

Com esta nova configuração, temos a nova função de transferência (x) do sistema chamada $G_p(s)$.

$$G_p(s) = \frac{2066}{(s + 17.3621 + j50)(s + 17.3621 - j50)} \qquad (x)$$

Simulação do novo sistema a uma entrada degrau

A figura nos mostra a resposta de $G_p(s)$ à uma entrada degrau de 3 volts e a ação de controle, respectivametne.

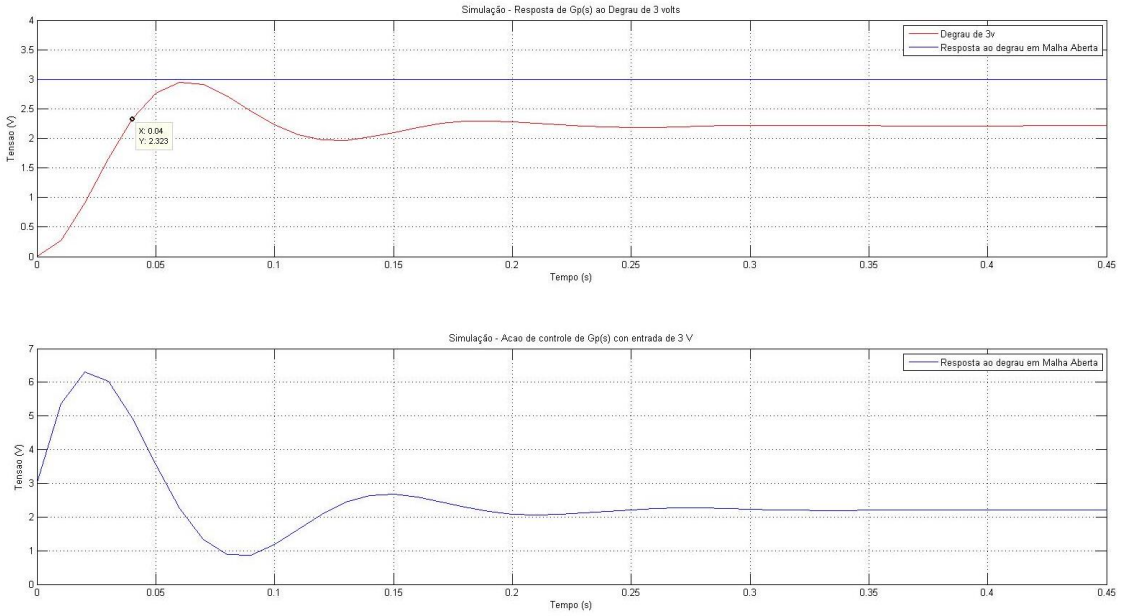


Figura 5 - Resposta de $G_p(s)$ à uma entrada degrau de 3V e Ação de controle do controlador K

Como se pode observar, o sistema ficou mais rápido, $\tau = 0.04s$, se tornou oscilatório como era previsto, entretanto ainda possui erro de posição no estado estacionário, o que era esperado, já que controladores proporcionais não corrigem este erro.

Controlador Proporcional por Alocação de Polos no Arduino

O Programa 1 é o projeto do sistema de controle da Figura 4.

Controlador Proporcional por Alocação de Polos

```
//Codigo simplificado

const int Ref = 0;      // Referencia sera conectado ao pino de entrada analogica A0

const int Sensor1 = 1; // O sensor sera conectado ao pino de entrada analogica A1

const int Sensor2 = 2; // O sensor sera conectado ao pino de entrada analogica A2

const int Atuador = 5; // O sinal de comando "analogico" (saida do controlador) pino de saida digital 5 (PWM)

double Valor_Referencia;

double Valor_Sensor1; // Variavel que armazenara o valor da saida (tensao no capacitor 1)

double Valor_Sensor2; // Variavel que armazenara o valor da saida (tensao no capacitor 2)

double Valor_Atuador; // Variavel que armazenara o valor da saida do controlador (acao de controle)

double D;

Double k1 = -2.2361;

double k2 = 2.5920;

double last_time;

void loop(){

  Valor_Sensor1 = analogRead(Sensor1); // Converte o valor de tensao numa palavra binaria de 10 bits (0V a 5V <---> 0 a 1023)

  Valor_Sensor1 = Valor_Sensor1/1023*5; // Mapeia de 10 bits para um escala de 0 a 5 V

  Valor_Sensor2 = analogRead(Sensor2); // Converte o valor de tensao numa palavra binaria de 10 bits (0V a 5V <---> 0 a 1023)

  Valor_Sensor2 = Valor_Sensor2/1023*5; // Mapeia de 10 bits para um escala de 0 a 5 V

  Valor_Referencia = analogRead(Ref); // Converte o valor de tensao numa palavra binaria de 10 bits (0V a 5V <---> 0 a 1023)

  Valor_Referencia = Valor_Referencia/1023*5; // Mapeia de 10 bits para um escala de 0 a 5 V

  D = k1*Valor_Sensor1 + k2*Valor_Sensor2; // EXPRESSAO DO CONTROLADOR

  Valor_Atuador = constrain((Valor_Referencia - D)* (255/5), 0, 255); // Restringe a faixa de 0V a 5V (0 a 255)

  analogWrite(Atuador, Valor_Atuador); // Escreve no pino 5 (Atuador), que simulará uma saida analogica via PWM

  }
```

Programa 1

A Figura 6 apresenta a resposta do sistema de controle por alocação de polos ao degrau de 3 volts. O sistema real apresentou uma frequência de oscilação menor em comparação com o

simulado, entretanto se comportou dentro do esperado e com constante de tempo $\tau = 0.046s$, o que é bem próximo do simulado.

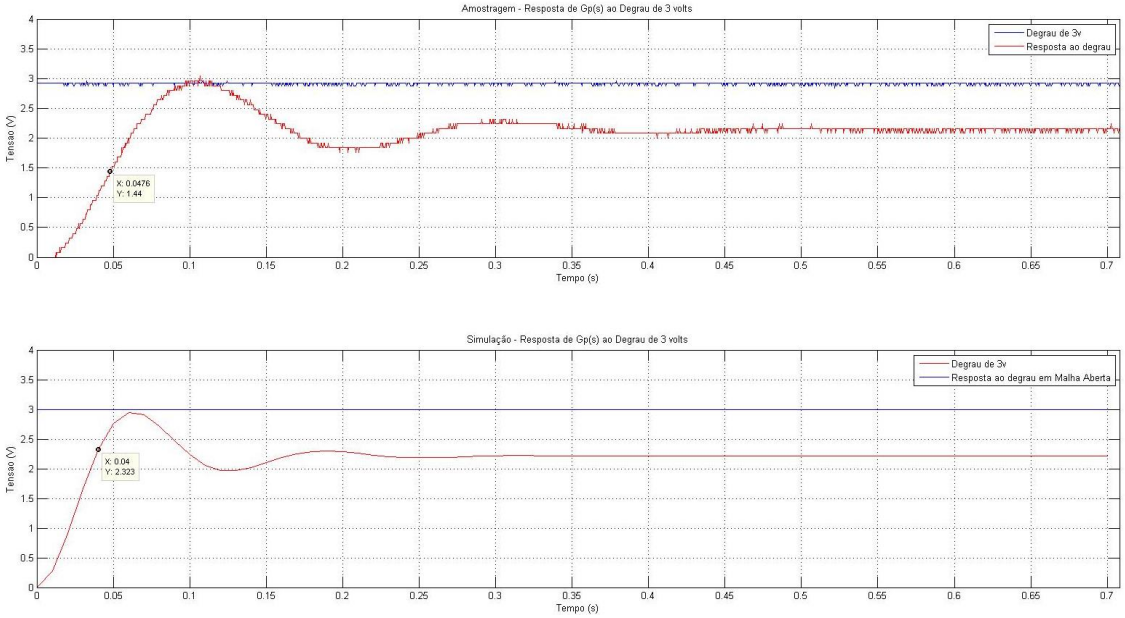


Figura 6 - Amostragem e simulação do sistema $G_p(s)$

Escolhendo o Tempo de amostragem e Função de Transferência Pulsada G(z)

Na sessão **Projeto de um controlador por alocação de Polos** foi apresentada a função de Transferência de malha fechada com o controlador K, equações *(ix) e (x)*. Também foi obtido uma constante de tempo para o sistema $\tau = 0.04s$. Entretanto, essa constante de tempo não foi observada em laboratório, devido à limitações de conhecimento sobre a precisão do *Data Cursor*, e foi considerada $\tau = 0.0332s$ obtida manualmente no toolbox *Figure* do *Matlab*®.

Em posse desta nova constante de tempo, foi calculado o tempo de amostragem **T** = 0.00664s, i.e., 20% de τ. Com este tempo de amostragem, através das ferramentas do *Matlab*®, obteve-se a função de transferência pulsada (xi):

$$G(z) = \frac{0.04181z + 0.03871}{z^2 - 1.685z + 0.7941} \qquad (xi)$$

A Figura 7 apresenta o novo sistema com o segurador de ordem zero e o amostrador T utilizados para obter G(z).

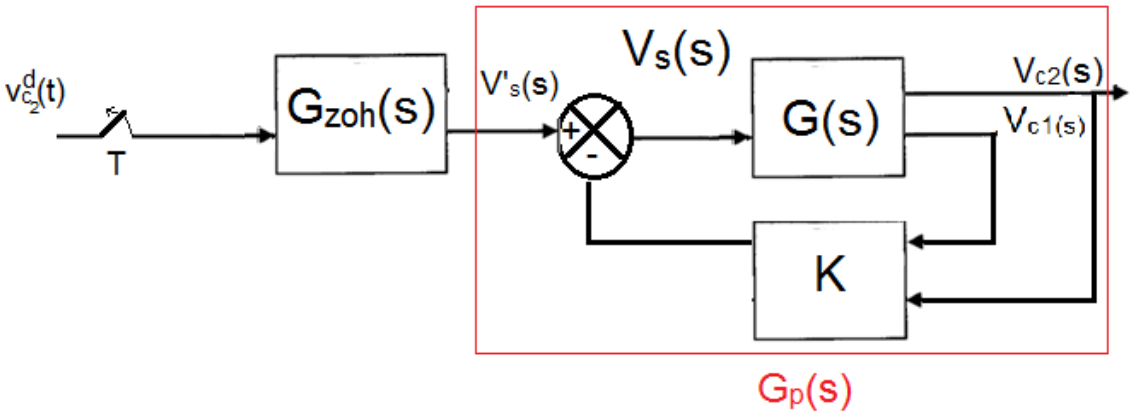


Figura 7 - Inserção do amostrador T e segurador de ordem zero $G_{zoh}(s)$ para obter G(z)

Simulação – Resposta de G(z) ao degrau de 3 volts

A Figura 8 apresenta a resposta de G(z), equação *(xi)* à um degrau de 3 volts e a resposta de $G_p(s)$, equação *(x).*

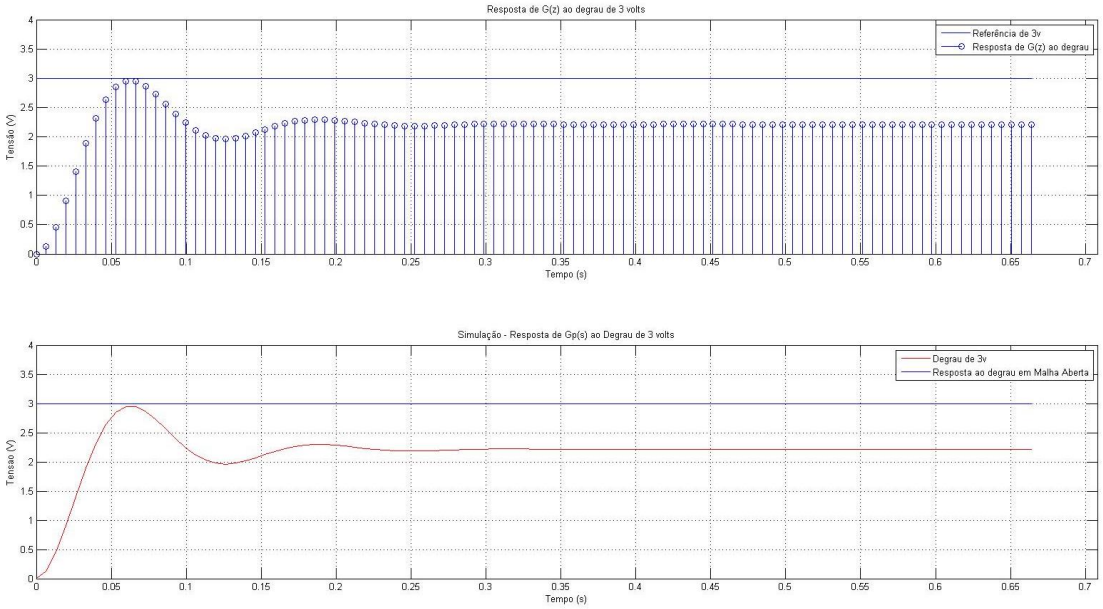


Figura 8 - Resposta de G(z) e $G_p(s)$ ao degrau de 3 volts

Através da *toolbox lsiminfo,* obtivemos que o tempo se setting de $G_p(s)$ é de 22.0291 segundos, enquanto que de G(z) é de 32.6627 segundos. Ambos obtiveram amplitudes máximas muito próximas, a primeira 2.9472 volts e a segunda 2.9456 volts.

Projeto do controlador PI

Visando eliminar as oscilações do sistema e na expectativa de deixar o sistema mais rápido, foi utilizado a toolbox *PID Tuner* do *Matlab*®. A Figura 9 mostra os parâmetros encontrados e a resposta deste controlador à um degrau.

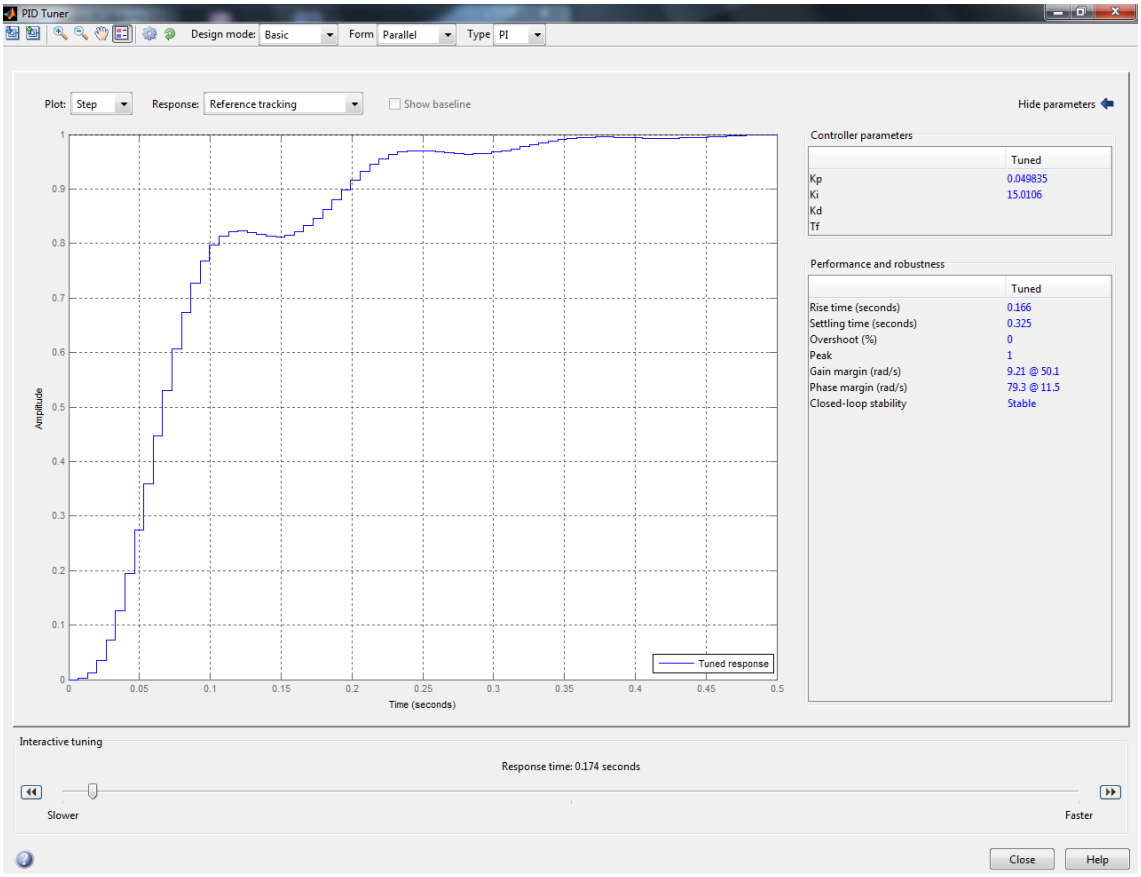


Figura 9 - Projeto de um controlador PI utilizando a toolbox PID Tuner

Foi inserido um tempo de resposta de 0.174 segundos, o que gera sobressinal de 0%, conforme se pode observar nos dados da Figura 9. Com este tempo de resposta, foram obtidos $K_p = 0.049835$ e $K_i = 15.0106$.

A equações (xii) e (xiii) nos apresentam a expressão geral de um controlador PID digital discreto:

$$c_k = K_p e_k + K_d \left(\frac{e_k - e_{k-1}}{T}\right) + K_i C_k^I \qquad (xii)$$

$$C_k^I = C_{k-1}^I + K_i T e_k \qquad (xiii)$$

Em posse de K_p e K_i, obtemos as equações (xiv) (xv) para o controlador PI.

$$c_k = K_p e_k + K_i C_k^I \qquad (xiv)$$

$$C_k^I = C_{k-1}^I + K_i T e_k \qquad (xv)$$

São essas as expressões utilizadas na codificação do controlador no Arduino.

A equação (xvi) nos dá a expressão geral do controlador PID Digital em Z.

$$D(z) = K_p + \frac{K_d}{T}\left(\frac{z-1}{z}\right) + K_i T\left(\frac{z}{z-1}\right) \qquad (xvi)$$

Partindo dela, chegamos no controlador projetado (xvii):

$$D(z) = 0.049835 + 0.0997\left(\frac{z}{z-1}\right) \qquad \textit{(xvii)}$$

Através do *Matlab* geramos a função de transferência em malha fechada com o controlador PI (xviii).

$$M(z) = \frac{0.0062513\ (z + 0.9258)\ (z - 0.3333)}{(z - 0.9249)\ (z^2\ -\ 1.754z\ +\ 0.8606)} \qquad \textit{(xviii)}$$

A Figura 10 nos apresenta a configuração do sistema de malha fechada com o controlador D(z).

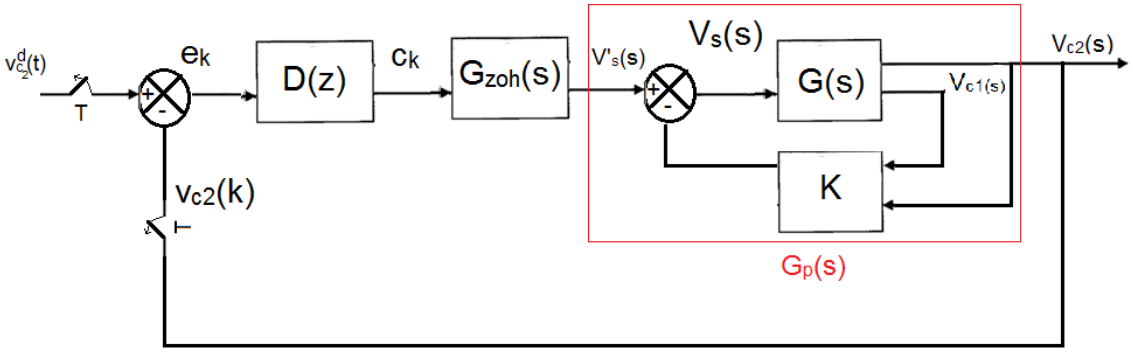
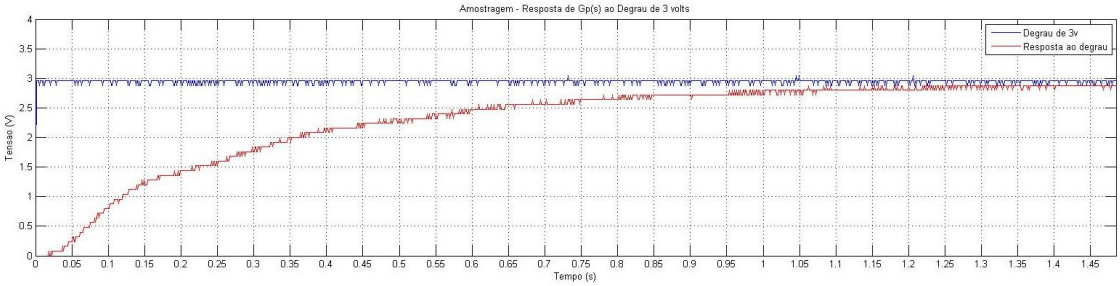
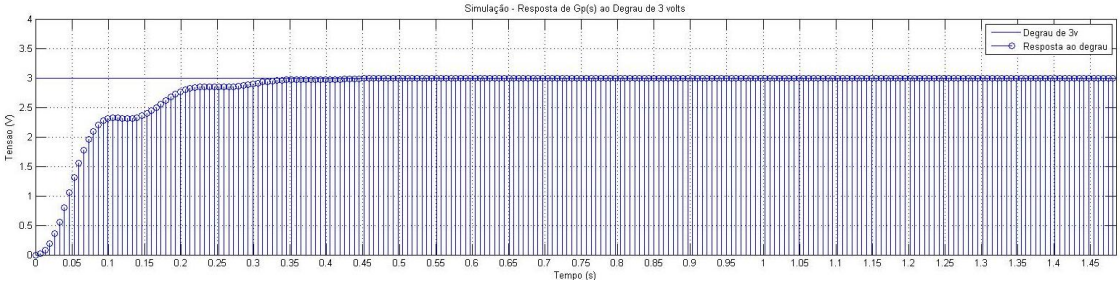


Figura 10 - Sistema de controle digital de malha fechada

A f nos apresenta a resposta de G(z) ao degrau de 3 volts obtidas na amostragem e na simulação.





Os resultados obtidos na simulação foram muito divergentes dos obtidos na amostragem. Em tese teríamos um sistema extremamente rápido, mas não foi o que aconteceu. Enquanto na simulação esperava-se um tempo de setting de aproximadamente 0.45 segundos, o sistema amostrado apresentou um tempo de setting bem maior, mas mesmo

assim, conseguiu-se melhorias no sistema, pois não existe mais sobressinal e nem erro de posição no estado estacionário.

Por fim, foi implementado o Programa 2, o controlador PI no arduino.

Controlador Proporcional Integral

```
//(...)Codigo simplificado

double D;

Double k1 = -2.2361;

double k2 = 2.5920;


double Erro; // Variavel que armazenara o sinal de erro (Valor_referencia - Valor_Sensor)

double I = 0;

double P;

const double Kp = 0.049835;

const double Ki = 15.0106;

const double T = 0.00664; // Tempo de amostragem em milissegundos

double last_time;

void loop(){

 Valor_Sensor1 = analogRead(Sensor1); // Converte o valor de tensao numa palavra binaria de 10 bits (0V a 5V <---> 0 a 1023)

 Valor_Sensor1 = Valor_Sensor1/1023*5; // Mapeia de 10 bits para um escala de 0 a 5 V

 Valor_Sensor2 = analogRead(Sensor2); // Converte o valor de tensao numa palavra binaria de 10 bits (0V a 5V <---> 0 a 1023)

 Valor_Sensor2 = Valor_Sensor2/1023*5; // Mapeia de 10 bits para um escala de 0 a 5 V

 Valor_Referencia = analogRead(Ref); // Converte o valor de tensao numa palavra binaria de 10 bits (0V a 5V <---> 0 a 1023)

 Valor_Referencia = Valor_Referencia/1023*5; // Mapeia de 10 bits para um escala de 0 a 5 V

 D = k1*Valor_Sensor1 + k2*Valor_Sensor2; // EXPRESSAO DO CONTROLADOR

  if ((millis() - last_time) > 1000*T) {

 last_time = millis();

 Erro = (Valor_Referencia - Valor_Sensor2);

 Erro = Erro*255/5; // Mapeia de uma escala de 0 a 5 V para 8 bits

 P = Kp*Erro;

 I = I + Ki * T * Erro;

 Valor_Atuador = constrain(((P+I) - D)* (255/5), 0, 255); // Restringe o valor na faixa de 0V a 5V (0 a 255)

 analogWrite(Atuador, Valor_Atuador); // Escreve no pino 5 (Atuador), que simulara  uma saida analogica via PWM

  }

}
```

Programa 2

Projeto do controlador de Dahlin

Assim como na sessão **Projeto do controlador PI**, foi projetado um controlador de Dahlin que tornou o sistema mais rápido e eliminou o erro de posição no estado estacionário.

A equação (xix) nos apresenta o controlador de Dahlin.

$$D(z) = \frac{1}{G(z)} \frac{1 - e^{-\frac{T}{\tau_d}}}{(1 - e^{-\frac{T}{\tau_d}})z^{-k}} z^{-k}, k \geq 1 \qquad (xix)$$

Como o sistema não possui tempo morto, k=1. Escolheu-se τ_d=0,15 segundos.

Utilizando o *Matlab*®, chegou-se a equação (xx) para o controlador de Dahlin projetado:

$$D(z) = \frac{1.0356\,(z^2 - 1.685z + 0.7941)}{(z + 0.9258)\,(z - 1)} \qquad (xx)$$

Consequentemente, nossa malha fechada agora possui nova equação (xxi).

$$M(z) = \frac{0.043301}{(z - 0.9567)} \qquad (xxi)$$

A figura nos apresenta as simulações do sistema apresentado na Figura 10, onde agora D(z) é o nosso controlador de Dahlin.

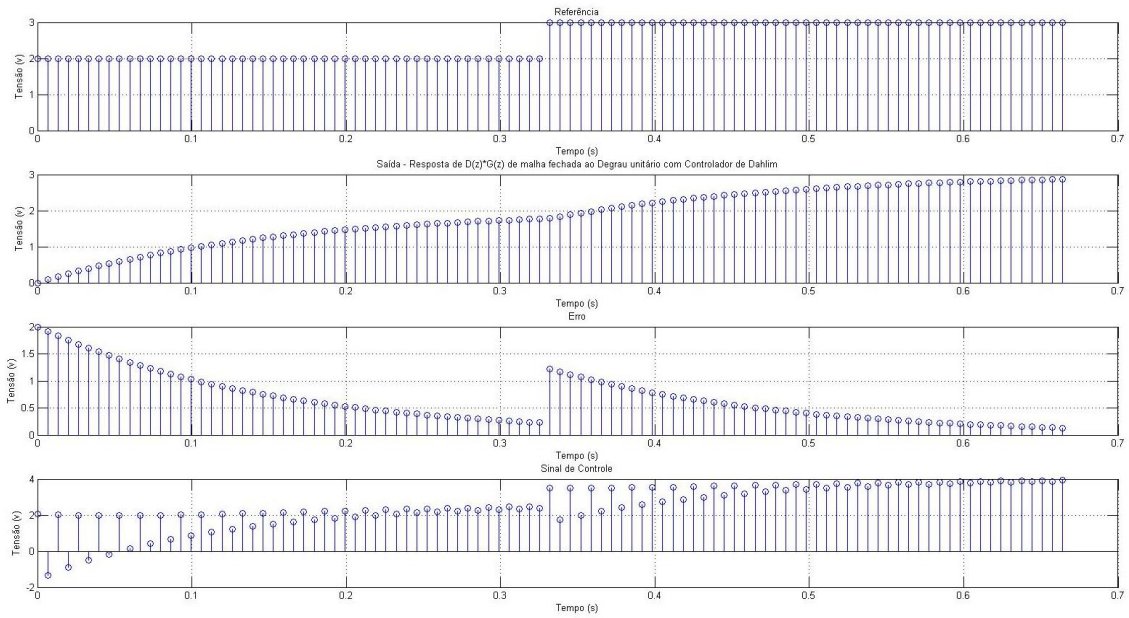


Figura 11 - Resposta de M(z) ao degrau de 3 volts

Como a ação de controle apresentou oscilações negativas a princípio, foi necessário realizar um offset na referência, de tal modo que o controlador digital (Arduino) não saturasse a saída de referência.

A Figura 12 nos mostra a amostragem do sistema.

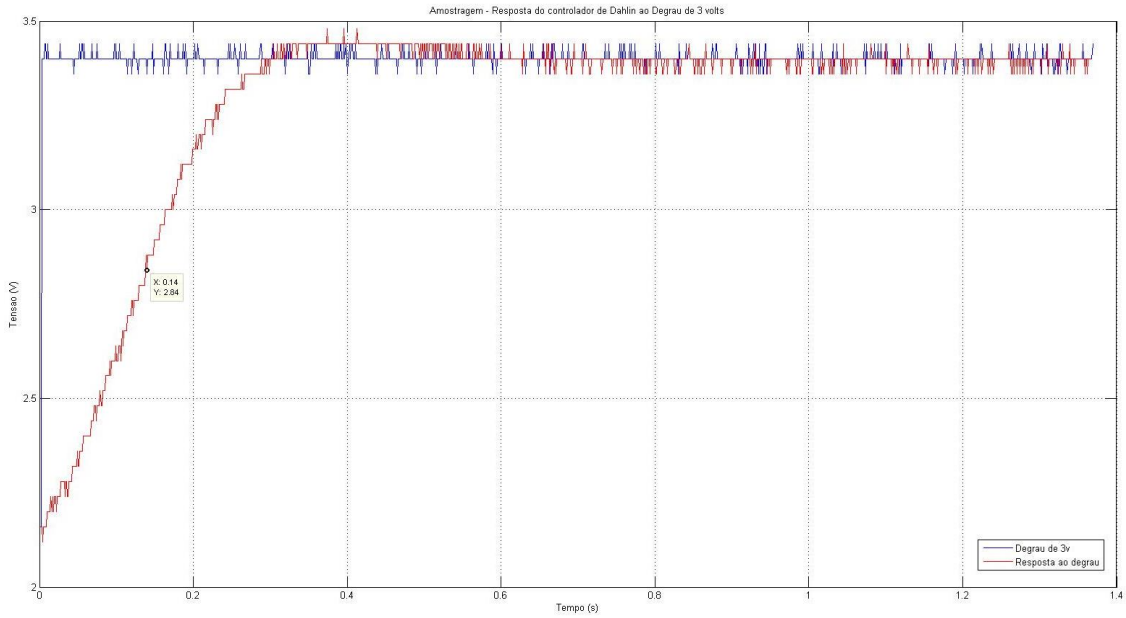


Figura 12 – Amostragem da Resposta de M(z) ao degrau de 3 volts com offset

Como se pode observar na Figura 12, foi preciso realizar um offset no gerador de sinais para que o sistema se comportasse da maneira desejada. O que de fato ocorreu. O sistema tornou-se bem mais rápido, estável com uma constante de tempo de aproximadamente 0.14 segundos.

Para implementar o controlador digital, foi preciso encontrar a equação de diferenças do sistema. A partir da equação (xx), chegou-se na seguinte equação de diferenças:

$$c_k = 0.0742c_{k-1} + 0.9258c_{k-2} + 2.9742e_k - 5.01152e_{k-1} + 2.3618e_{k-2} \qquad (xxii)$$

Com a equação (xxii) foi implementado o Programa 3, Controlador de Dahlin o no Arduino.

Controlador de Dahlin

```
//(...)Codigo simplificado

double D;

Double k1 = -2.2361;

double k2 = 2.5920;

double Erro; // Variavel que armazenara o sinal de erro (Valor_referencia - Valor_Sensor)

const double T = 0.00664; // Tempo de amostragem em milissegundos

double ck=0;  //c(k)

double ck1=0; //c(k-1)

double ck2=0; //c(k-2)

double ek=0;  //e(k)

double ek1=0; //e(k-1)

double ek2=0; //e(k-2)

double last_time;

void loop(){

 Valor_Sensor1 = analogRead(Sensor1); // Converte o valor de tensao numa palavra binaria de 10 bits (0V a 5V <---> 0 a 1023)

 Valor_Sensor1 = Valor_Sensor1/1023*5; // Mapeia de 10 bits para um escala de 0 a 5 V

 Valor_Sensor2 = analogRead(Sensor2); // Converte o valor de tensao numa palavra binaria de 10 bits (0V a 5V <---> 0 a 1023)

 Valor_Sensor2 = Valor_Sensor2/1023*5; // Mapeia de 10 bits para um escala de 0 a 5 V

 Valor_Referencia = analogRead(Ref); // Converte o valor de tensao numa palavra binaria de 10 bits (0V a 5V <---> 0 a 1023)

 Valor_Referencia = Valor_Referencia/1023*5; // Mapeia de 10 bits para um escala de 0 a 5 V

 D = k1*Valor_Sensor1 + k2*Valor_Sensor2; // EXPRESSAO DO CONTROLADOR

  if ((millis() - last_time) > 1000*T) {

 last_time = millis();

 Erro = (Valor_Referencia - Valor_Sensor2);

 Erro = Erro*255/5; // Mapeia de uma escala de 0 a 5 V para 8 bits


 ck = 0.074242981308880*ck1 + 0.925757018691120*ck2 +   1.035588322898393*ek - 1.744865183567332*ek1 + 0.822341931913884*ek2;

 Valor_Atuador = constrain(ck,0, 255); // Restringe o valor na faixa de 0V a 5V (0 a 255)

 analogWrite(Atuador, Valor_Atuador); // Escreve no pino 5 (Atuador), que simulara  uma saida analogica via PWM

  }

}
```

Programa 3

Referências

[1] Phillips, Charles L and H. Troy Neagle. Digital System Analysis and Design. Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[2] R. C. Dorf and R. H. Bishop. Sistemas de Controle Modernos. LTC Editora, Rio de Janeiro, 2001.

[3] K. Ogata. Engenharia de Controle Moderno. Prentice Hall do Brasil, S.P., 1998.

[4] Pedro Dinis Gaspar, António Espírito Santo, J. A. M. Felippe de Souza, APONTAMENTOS DE MATLAB CONTROL SYSTEM Toolbox, Edição Abril 2002