Controle PI de um sistema Pêndulo - Hélice

William Miranda Gusmão
Controle Digital - Sistemas Eletrônicos
Universidade de São Paulo
São Paulo-SP, Brasil
williamgusmao@hotmail.com

***Resumo*—Este projeto aborda o controle do ângulo de um sistema pêndulo-hélice que consiste em uma hélice acoplada a um motor DC preso na ponta da haste, que está fixada em um eixo. Foram feitos diversos testes aplicando degraus a fim de se obter um modelo aproximado do sistema para que fosse possível projetar o controlador com o auxílio do software MATLAB, realizar simulações e então aplicá-lo ao projeto prático.**

***Palavras chave*—hélice, motor, pêndulo, controle, PI**

I. Introdução

Sistemas com propulsão através de hélices, como é o exemplo dos drones, estão muito presentes no mercado e cada vez mais populares[1]. Eles são amplamente utilizados na área militar e estão presentes também no ramo de entregas, filmagem, fotografia, entre outros. Com um sistema Pêndulo - Hélice, é possível aplicar técnicas de modelagem, de controle, e estudar de forma mais simples um sistema com este tipo de propulsão[2].

II. Metodologia

A. Hardware

O hardware deste projeto é formado por um módulo Ponte H L298n para controlar a potência e acionar diretamente o motor DC com um sinal PWM, um potenciômetro de 10kR acoplado ao eixo do pêndulo para ler a posição angular da haste, uma fonte 12V 5A estabilizada e uma placa microcontrolada Nucleo H755ZI-Q, onde é utilizado o microcontrolador STM32H755ZI, que possui 2 cores, sendo um Cortex M7 e um Cortex M4[3,4].

B. Software

O software foi feito em Bare-Metal utilizando os 2 cores do microcontrolador. No core M7 está toda a parte relacionada a comunicação serial(RS232) entre a placa Nucleo e o computador. É enviado para o computador o sinal de saída, o sinal de controle e o setpoint com um tempo de amostragem de 10ms. O valor de setpoint é enviado do computador para o microcontrolador. No core M4 está a leitura A/D (Analógica/Digital), a geração do PWM e toda lógica de controle em si. O gatilho para a leitura do conversor A/D é feita por um dos Timers. Outro Timer foi configurado com um tempo de 10ms, onde todos os valores são atualizados, é feito a média dos valores de saída lidos até o momento, e o controle é realizado.

C. Planta

A planta a ser controlada é um sistema Pêndulo - Hélice, onde há um potenciômetro acoplado ao eixo da haste, e um motor DC com uma hélice acoplada na haste. É possível ver o sistema montado na figura 1, e um desenho que representa as principais partes do sistema na figura 2[5].

Figura 1. Pêndulo - Hélice Montado.

III. Identificação do sistema

A. Sinal de controle mínimo

O sistema em questão com propulsão a hélice necessita de uma excitação considerável para começar a se mover. A fim de eliminar qualquer atraso devido a este fator, foram feitos testes para estimar o menor sinal de controle com que a planta se movimenta. Foi desenvolvido um algoritmo no software para que se aplicasse um valor de PWM que aumentava em 1% a cada 5 segundos, iniciando-se em 37% e chegando até 44%. Este teste foi realizado 9 vezes, a fim de comparar os resultados para confirmar qual seria o sinal de controle inicial (r = 0). Após os testes, o valor de PWM inicial escolhido foi de 42%. É possível observar um dos testes realizados na figura 3.

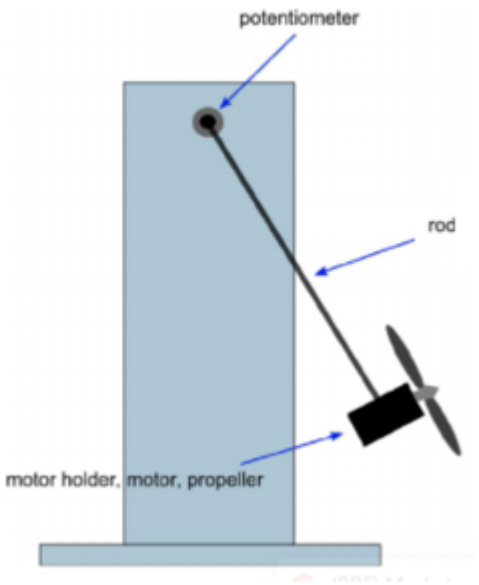

Figura 2. Pêndulo - Hélice Desenho.

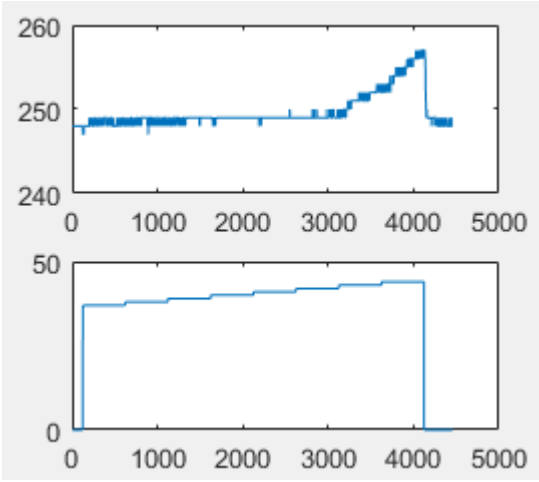

Figura 3. Teste para definição de PWM mínimo.

B. Resposta ao Degrau e Aquisição

Para identificar um modelo aproximado da planta, foi aplicado inicialmente um PWM de 42%, e depois um degrau de 60% (r = 0 , r = 18). Este procedimento foi realizado 11 vezes. Após analisar os dados obtidos e verificar a repetibilidade da resposta da planta ao degrau, um conjunto de dados foi escolhido a fim de servir como base para a geração do modelo. A resposta ao degrau e a entrada podem ser observadas na figura 4.

C. Identificação do modelo

Para auxiliar na identificação do modelo, foi utilizada a ferramenta System Identification Toolbox. A partir dos dados coletados no item (A), foi estimado um modelo de 3ª Ordem

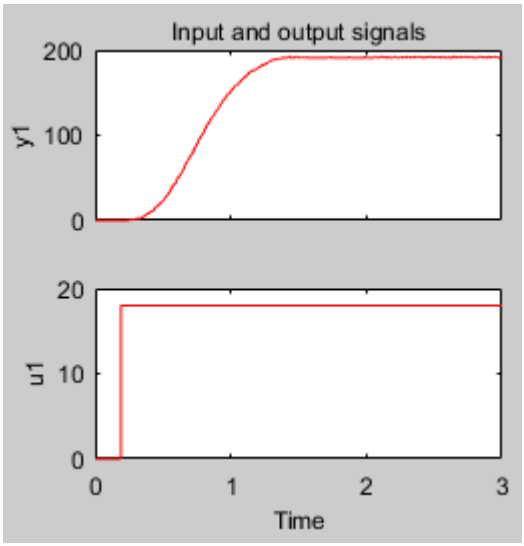

Figura 4. Resposta da planta ao degrau de 18%.

para esta planta sem atraso de transporte, com um Fit de 98,35%. O modelo estimado foi:

$$G(s) = \frac{-23.05s + 688.9}{s^3 + 8.093s^2 + 37.17s + 64.47}$$

IV. Projeto do Controlador

O controlador escolhido para este projeto é o PI (Proporcional - Integral). Controlador PI[6]:

$$PI = P(1 + \frac{1}{Ti} * Ts * \frac{1}{z-1})$$

Para este projeto estamos considerando:

$$Ki = \frac{1}{Ti}$$

Para auxiliar no projeto do controlador, foi utilizada a ferramenta Sisotool do MATLAB, que permite alterar os ganhos em tempo real enquanto mostra o lugar das raízes e a resposta do sistema em malha fechada a um degrau unitário. O seguinte controlador foi projetado:

$$C(s) = \frac{0.04654s + 0.1551}{s}$$

É possível ver o a resposta do sistema em malha fechada e o lugar das raízes na figura 5.

V. Simulações

Para que sejam feitas simulações, e em seguida embarcar o controlador a fim de realizar testes na planta física, deve-se discretizar o controlador. Para isso foi utilizado a função c2d do MATLAB, obtendo-se o seguinte controlador no plano Z:

$$C(z) = \frac{0.04654z + 0.04499}{z - 1}$$

Sample time: 0.01 seconds.

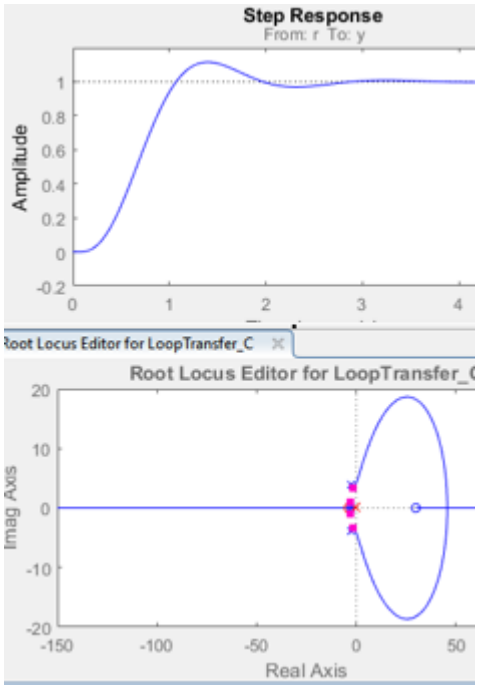

Figura 5. Resposta ao Degrau e Lugar das Raízes.

A partir disso, podemos chegar nos valores de Kp e Ki do controlador PI: Kp = 0.04654 Ki = 3.3326

No software SIMULINK, o controle foi feito utilizando o MATLAB Function com um algoritmo para o controlador PI. As figuras 6 e 7 mostram o diagrama de blocos e o algoritmo utilizado para simulação:

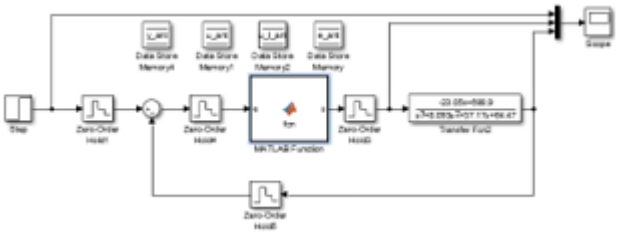

Figura 6. Diagrama de blocos.

Os resultados obtidos da simulação foram satisfatórios, onde houve um sobressinal de aproximadamente 13% e um tempo de acomodação em aproximadamente 2,6s. Na figura 8 é possível ver a simulação realizada no SIMULINK, onde a linha amarela é o setpoint, a linha azul é o sinal de controle e a linha vermelha é o valor de saída.

VI. Experimentos Práticos

A. Implementação do Controlador no Software

Na hora de implementar o algoritmo no microcontrolador, alguns valores precisaram se adequar ao que é necessário no sistema físico. Como o o valor mínimo de PWM para o motor é de 42%, foi preciso colocar um offset no valor de PWM que é passado para a saída do microcontrolador. Foi definido

```
function u  = fcn(e)
global u_ant; global e_ant; global u_I_ant;
K_P = 0.04654;
K_I = 3.3326;
Ts = 0.01;
sat = 57;
if (u_ant >= sat || u_ant <= -sat)
    e_i = 0;
else
    e_i = e;
end
u_P = K_P*e;
u_I = (K_P*Ts*K_I)* e_i + u_I_ant;
u = u_P + u_I;
u_ant = u; e_ant = e; u_I_ant = u_I;
```

Figura 7. Algoritmo PI.

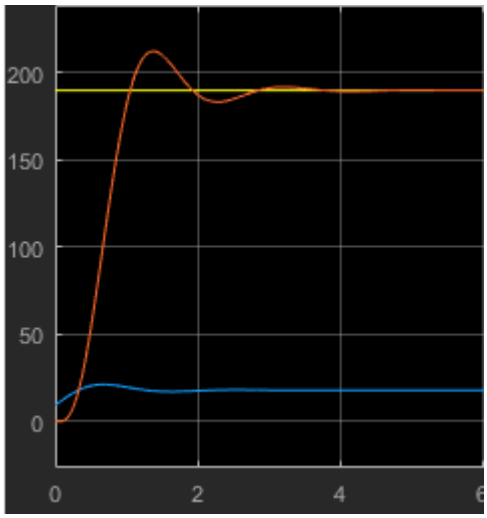

Figura 8. Simulação do sistema em Malha Fechada.

também a região de saturação para a saída do controlador, levando em consideração que o valor mínimo para o PWM é de 42% e o valor maxímo é 100%. O valor de saída não possui uma unidade física de medida conhecida, e sim um valor que inicia-se em 269 (posição inicial da Planta) e pode variar até aproximadamente 600. Foi inserido um offset de -269 no valor de saída para que a variação seja de 0 a aproximadamente 300. A figura 9 mostra o script do controlador.

B. Resultados obtidos

Nos testes práticos, primeiramente foi enviado ao controlador um setpoint de 185. Após o valor de entrada ser alcançado, um peso foi inserido sobre a haste do pêndulo, causando um distúrbio. O ângulo caiu consideravelmente, mas logo em seguida retornou ao valor de setpoint. Ao retirar o peso, o distúrbio causado teve uma influência ainda maior no sistema. A saída apresentou um sobressinal de aproximadamente 22%

```
y = adc_average - 269; // Offset de -269 para o sinal de saída
e = (setpoint-y);         // Cálculo do erro
if (u_ant >= (100-42) || u_ant <= (0)) //Anti Windup
  e_i = 0;
else
  e_i = e;
u_p = Kp*e; // Parcela proporcional
u_i = (Kp*Ts*Ki)* e_i + u_i_ant;  //Parcela integrativa
                                  //com anti windup
u = u_p + u_i + 0.5;  // Saída do controlador somado a um
                      // fator de arredondamento
if(u > (100-42))  //Região de saturação
  u = 100-42;
else if(u < (0))
  u = 0;
u_ant = u; // Atualiza u_ant para a próxima iteração
e_ant = e; // Atualiza e_ant para a próxima iteração
u_i_ant = u_i; // Atualiza u_i_ant para a próxima iteração
pwm_out = u+42; // Atualiza PWM com a saída do controlador +offset
```

Figura 9. Script do controlador embarcado.

e um tempo de acomodação de aproximadamente 5,5s. Nas figuras 10 e 11 é possível ver a aquisição do teste realizado.

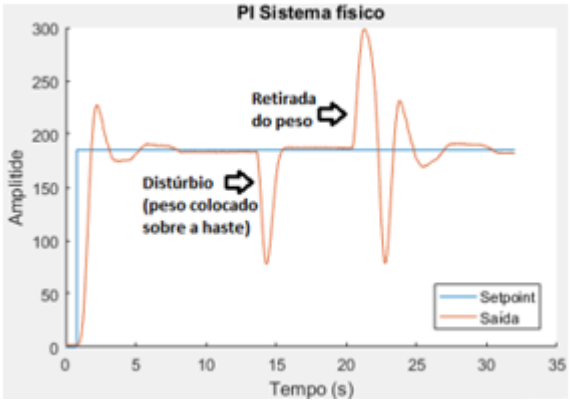

Figura 10. Testes práticos: Saída x Entrada.

Figura 11. Testes práticos: Sinal de controle.

VII. Conclusão

Foi possível controlar o sistema com um controlador PI e, apesar apesar do comportamento lento que resultou em uma resposta não muito eficiente contra distúrbios, os resultados foram satisfatórios e podem servir como base para se aprofundar mais nesse tipo de sistema. Para projetos futuros seria interessante acrescentar a parcela Derivativa com o objetivo de obter uma resposta mais rápida e atenuar o efeito causado pelos distúrbios. Uma outra proposta para melhorar o desempenho do controlador seria diminuir a resolução do PWM, que atualmente é 1%, pois a planta possui um ganho relativamente alto. Foi observado também que um modelo de 3ª ordem representa com fidelidade a planta em questão, porém isso resulta em um sistema muito complexo e dificulta o controle da mesma.

Referências

[1] M. Yoon, ”Stabilization of a Propeller - Driven Pendulum”, International Journal of Engineering Research and Technology (IJERT), Vol. 5 Issue 01, pp. 230-233, 2016.

[2] T. Huba, T. Malatinec, M. Huba, ”Propeller-Pendulum for Nonlinear UAVs Control”, International Journal of Online and Biomedical Engineering (iJOE), Vol. 9, pp. 42-46, 2013.

[3] STMicroelectronics, ”STM32H7 Nucleo-144 boards (MB1364) - User manual”, Jun. 2020.

[4] F. Guimarães, ”Guia definitivo de uso da Ponte H L298N”, Mundo Projetado, Eletrônica, Abril 2018.

[5] E. T. Enikov, G. Campa, ”Mechatronic Aeropendulum: Demonstration ofLinear and Nonlinear Feedback Control PrinciplesWith MATLAB/Simulink Real-Time Windows Target”, IEEE TRANSACTIONS ON EDUCATION, Vol. 55, pp. 538-545, Nov. 2012.

[6] MathWorks Inc., ”Discrete-Time Proportional-Integral-Derivative (PID) Controllers”.