MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ

MARIANA MULINARI PINHEIRO MACHADO

UTILIZAÇÃO DE HARDWARE-IN-THE-LOOP PARA REALIZAR O CONTROLE DE UM MOTOR A PARTIR DE UMA METAHEURÍSTICA

CURITIBA

2018

MARIANA MULINARI PINHEIRO MACHADO

UTILIZAÇÃO DE HARDWARE-IN-THE-LOOP PARA REALIZAR O CONTROLE DE UM MOTOR A PARTIR DE UMA METAHEURÍSTICA

Trabalho de conclusão de curso apresentado como requisito parcial para a obtenção do grau de engenheiro eletricista, no programa de graduação em engenharia elétrica da Universidade Federal do Paraná.
Orientador: Prof.Dr. Gideon Villar Leandro

CURITIBA

2018

AGRADECIMENTOS

Gostaria de agradecer ao professor orientador Prof. Dr. Gideon Villar Leandro por ter proposto o tema de pesquisa e por todo o apoio ao longo do desenvolvimento do trabalho. Também gostaria de agradecer à banca pelas sugestões e melhorias propostas que agregaram valor ao trabalho. Além disso gostaria de agradecer o apoio de toda a minha família e namorado durante esse tempo e por prestarem apoio para a conclusão desse trabalho.

RESUMO

Os controladores do tipo PID são os mais utilizados na indústria por sua simplicidade de implementação. Mesmo com tamanha popularidade, não existe até hoje uma solução ótima e única de determinar seus coeficientes. Nesse contexto tem-se estudado a possibilidade da utilização de metaheuristicas para sintonizar esses controladores. Nesse trabalho a sintonia do PID usando a metaheuristica ED (evolução diferencial) será comparada a dois outros tipos de sintonia mais difundidos (lugar das raizes e método de controle do modelo interno). Os três controladores serão utilizados para o controle de velocidade e posição de um motor universal e serão testados em situações em que se tem atraso no sinal. O sistema final de malha fechada será formado por software (Simulink) e hardware (motor, ponte H e sensores) partindo do princípio de hardware-in-the-loop. A comunicação entre software e hardware será feita pelo dispositivo DAQ NI USB 6009. Com a comparação dos três tipos de controladores pode-se concluir que a utilização da metaheurística para a sintonia tem bons resultados em detrimento das outras e uma implementação relativamente répida e fácil.

Palavras chave: PID, metaheuristica, motor, hardware-in-the-loop.

LISTA DE FIGURAS

LISTA DE SIGLAS

PID	Controlador proporcional, integral e derivativo.
ED	Metaheurística evolução diferencial.
IMC	Controlador por modelo de controle interno.
LR	Controlador por lugar das raízes.
PWM	Modulação por largura de pulsos.
ITAE	Integral do tempo multiplicado pelo módulo do erro.
ADC	Conversor analógico para digital.

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

Cada vez mais a indústria busca automatizar ao máximo os seus processos. Quanto mais automatização, menores são os custos e maior a precisão. Com o aumento desses processos também ocorre um aumento no número de malhas de controle. Apesar dos primeiros controladores do tipo PID terem sido idealizados por meados dos anos quarenta, eles ainda são a topologia de controladores mais usados hoje na indústria (OLIVEIRA P.B.M. ; PIRES, 2015). Mesmo que outras topologias com boas performances tenham sido criadas e estudadas desde então, os controladores do tipo PID são mais fáceis e simples de serem implementados além de serem os mais conhecidos.

Nesse cenário, torna-se crucial que os controladores sejam os mais robustos o possível para que tenham um bom desempenho até em se considerando situações adversas e não previstas (não linearidades, atrasos nos sinais, perturbações, etc.). Para garanti-lo é necessário que a sintonia dos parâmetros dos PID seja feita de maneira optimal e para isso inúmeros métodos foram e estão sendo estudados.

1.1 Problematização

Apesar de serem largamente utilizados ainda nos dias de hoje, e apesar do fato de inúmeras pesquisas já terem sido realizadas a respeito da sintonia de controladores PID, ainda não se tem nenhum método que seja tido como ideal para fazê-lo. Os métodos mais usuais e mais conhecidos são os métodos de tabela como Ziegler Nichols e Cohen Coon, porém esses métodos além de não serem robustos levam muito tempo para serem realizados uma vez que contam com um processo de tentativa e erro. Dessa forma, existe uma grande necessidade de encontrar outros métodos que levem menos tempo para fazer a sintonia dos parâmetros e que resultem em um controlador que permita escolher melhor a performance do sistema em malha fechada sendo mais robustos.

1.2 Hipótese

Nos últimos anos têm-se levantado a ideia da utilização de metaheurísticas para se fazer a otimização de sintonia de controladores PID.

Em (OLIVEIRA P.B.M. ; PIRES, 2015) um controlador posicast PID é obtido através de uma metaheurística de procura gravitacional. Um outro exemplo é em (SHAYANFAR H. A. ; SHAYEGHI, 2016) em que um PID do tipo estabilizador é sintonizado a partir da metaheurística otimização dos lobos cinzentos. Essa mesma metaheurística é usada (DAS K.R. ; DAS, 2015) para fazer o controle de velocidade de um motor. Uma outra metaheurística já utilizada para sintonia de controladores PID é a otimização por enxame de partículas tratada em (ANDRADE L. H. S. ; COSTA, 2014) para uma planta didática industrial. Em (SERAPIAO, 2011) é a metaheurística busca harmonica que é utilizada para fazer a sintonia. Um outro tipo

de metaheurística chamada Busca Tabu foi testada para a sintonia de PIDs para o controle do nível de um tanque em (GOMES, 2009). Em (FRANCA, 2009) a metaheurística escolhida para realizar a sintonia do controlador PID foi a *Iterated Local Search*.

Metaheurísticas são técnicas e algoritmos que com algum grau de aleatoriedade buscam otimizar tanto quanto possível problemas difíceis a partir de processos iterativos. Normalmente, o algoritmo vai buscar os valores que minimizem ou maximizem um dado critério de seleção.

Além disso, as metaheurísticas são úteis pelo fato de poderem ser usadas com qualquer tipo de sistemas uma vez que independem deles. O interesse em utilizá-las metaheurísticas para sintonia de controladores PID é o fato de que elas chegam em um resultado muito próximo do ótimo e não levam muito tempo para fazê-lo.

Nesse trabalho a metaheurística que será estudada será a Evolução Diferencial (ED). Ela será comparada com dois outros métodos convencionais de sintonia de controladores PID (IMC e LR) para a realização do controle de velocidade e posição de um motor a partir de *hardware-in-the-loop*. As malhas de posição e velocidade podem ser resumidas numa só pelo fato da malha de posição englobar a malha de velocidade (figura 1.1.).

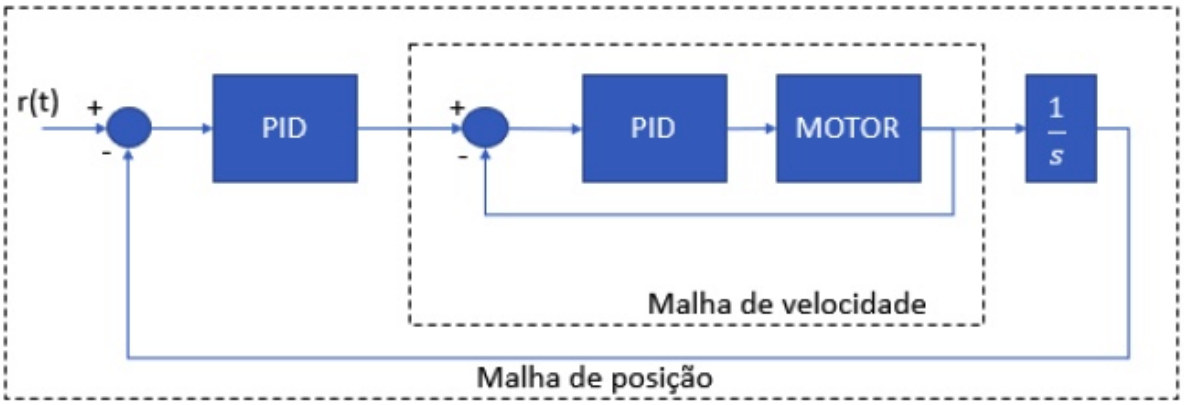

Figura 1.1: Malha geral de controle de posição a qual envolve a malha de controle de velocidade para um motor universal. Fonte: a autora, 2018.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral desse projeto é então de comparar a sintonia de controles do tipo PID feita a partir da metaheurística ED com outros métodos mais convencionais (IMC e LR) para controle de posição e velocidade de um motor a partir de *hardware-in-the-loop*.

1.3.2 Objetivos Específicos

- Identificação da função de transferência do motor universal.
- Sintonia dos controladores PID com os diferentes métodos.
- Simulação da malha.

- Implementação do sistema real utilizando-se do “hardware in the loop”.
- Comparação dos três tipos de sintonia dos controladores.

1.4 Justificativa

O interesse desse trabalho é estudar a eficácia e robustez da proposta de controladores PID sintonizados a partir de metaheurística comparando-os com os métodos mais conhecidos e mais difundidos. A relevância se da ao fato de que mesmo que esse tipo de controlador esteja em voga por muito tempo até agora, ainda não existe uma maneira ideal de fazer a sintonia de seus coeficientes. Dessa forma, como esse é o tipo de controlador mais usado na indústria, uma sintonia que de parâmetros mais robustos e em menos tempo é de grande interesse para o cenário atual.

1.5 Sequência de trabalho

Na sequência desse trabalho, serão abordados o hardware formado pelo motor, ponte H e sensores utilizados para realizar os testes dos controladores. Os métodos esccolhidos para realizar a sintonia dos controladores PID. E finalmente os resultados obtidos tanto em simulação com e sem atraso quanto em testes reais realizados pela aplicação do *hardware-in-the-loop*.

CAPÍTULO 2

HARDWARE

Como discutido o objetivo desse trabalho é de comparar a utilização de uma metaheurística para sintonizar coeficientes de um controlador PID. Essa comparação será feita a partir da aplicação desses controladores para controlar a velocidade e posição de um motor.

A malha de controle será feita a partir de *hardware-in-the-loop*. A ideia de *hardware-in-the-loop* é que a malha de controle se realiza entre o software e o hardware. Em que o software consiste dos controladores que enviam sinais de controle, para o caso desse projeto será o software Simulink. O hardware, o qual será estudado nesse capítulo é composto por um motor cuja velocidade e posição deseja-se controlar, uma ponte H que permite controlar a quantidade de tensão que chega ao motor, e os sensores de posição e velocidade que permitem realizar a realimentação e o fechamento da malha de controle.

2.1 Motor

O motor escolhido para a realização desse trabalho foi o motor universal modelo LE420BR (figura 2.1) por motivos de disponibilidade. Um motor universal pode operar tanto com corrente contínua quanto com corrente alternada. Para motores desse tipo as bobinas do estator são ligadas elétricamente ao rotor. Normalmente esse tipo de motor possui um alto torque de partida e pode atingir altas velocidades sem carga.

Figura 2.1: Esquemático utilizado para a ponte H. Fonte: Peluchi (2018)

Nesse trabalho o motor universal será utilizado apenas em aplicações de corrente contínua. Logo, a equação que o descreve (2.1) pode ser encontrada baseando-se na figura 2.2.

$$P_{motor}(s) = \frac{K}{(J \cdot s + b) \cdot (L \cdot s + R) + K^2} \tag{2.1}$$

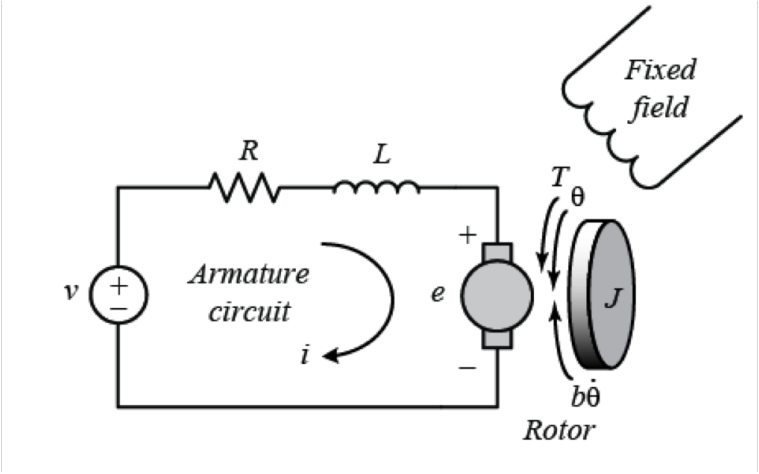

Figura 2.2: Modelo de um motor. Fonte: Matlab (2018)

2.2 Ponte H

Uma das partes importantes do hardware desse projeto é a ponte H. A ponte H é normalmente utilizada em sistemas em que se deseja controlar a posição de um motor pois ela permite que se mude o sentido de rotação do eixo do motor.

O circuito de ponte H utilizada para esse projeto se encontra na figura 2.3.

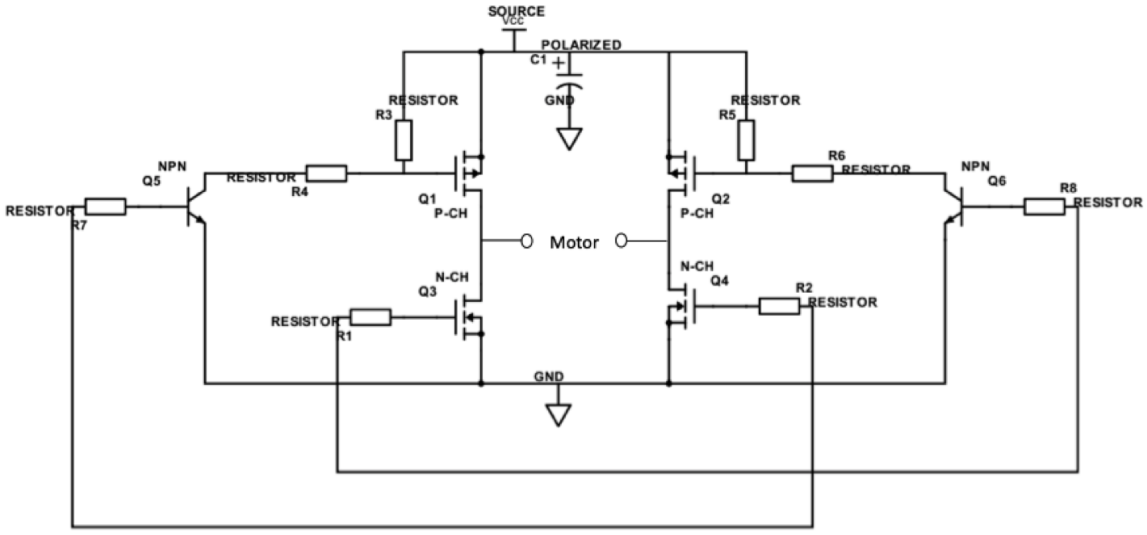

Figura 2.3: Esquemático utilizado para a ponte H. Fonte: A autora, 2018.

A ideia por trás do circuito é que existem dois sinais de controle que são aplicados na base dos transistores npn indicados na figura 2.3. Enquanto um desses sinais for do tipo PWM o outro estará a zero e vice versa. Os transistores quando recebem um sinal do tipo PWM na sua base (BJT) ou gate (MOSFET) trabalham como uma chave aberta ou fechada e quando rescebem o sinal 0 se mantém

fechados. Logo com os dois sinais PWM ocorrerá a ativação ou desativação simultânea dos transistores que estiverem na diagonal um do outro.

Ou seja, se o sinal de PWM for aplicado no transistor *Q5* e um sinal nulo for aplicado no transistor *Q6*, Os transistores *Q1* e *Q4* serão ativados e o motor girará em um sentido. Por outro lado, se o o sinal de PWM for aplicado no transistor *Q6* e um sinal nulo for aplicado no transistor *Q5*, Os transistores *Q2* e *Q3* serão ativados e o motor girará no sentido oposto.

O *dutycicle* do sinal de PWM serve para controlar o quanto da tensão máxima *VCC* será aplicada nos terminais do motor. Um *dutycicle* de 100% sigifica uma tensão de VCC aplicada ao motor, um *dutycicle* de 50% significa uma tensão equivalente a metade de VCC e assim sucessivamente.

Os componetes utilizados nesse projeto para a realização da ponte H se encontram na tabela a seguir:

Componente	**Valor**
R1, R2	1kΩ
R3, R5	4.7kΩ
R4,R6	1kΩ
R7, R8	2.2kΩ
C1	10 μF
Q1,Q2	IRF9540NPBF
Q3,Q4	IRFZ44NPBF
Q5,Q6	2N3904BU

2.3 Sensor de posição

O sensor de posição escolhido para esse projeto foi o encoder KY-040. Esse encoder possui três pinos denominados *A*,*B* e *C*. O que liga esses pinos são duas chaves um conectando *A* a *C* e o outra conectando *B* a *C*. Se o rotor for rotacionado em sentido horário, a chave que conecta *A* a *C* será a primeira a mudar de estado e para o sentido anti-horário a chave que conecta *B* a *C* será a primeira a mudar.

Logo, para determinar o sentido de rotação basta determinar qual chave mudou de estado em primeiro lugar. Um encoder possui um número fixo de posições por revolução, no caso do encoder utilizado para esse projeto são disponíveis 20 posições por revolução. Logo para encontrar o valor em angulos rotacionados basta aplicar a equação (2.2). Em que n_{pos} é o número de posições percorridas pelo rotor.

$$angulo = \frac{n_{pos} \cdot 2 \cdot \pi}{20} \tag{2.2}$$

Os pinos do encoder KY-040 utilizado se encontram na figura 2.4.

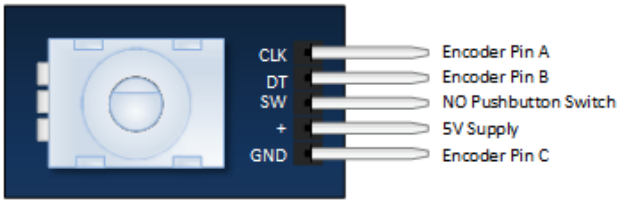

Figura 2.4: Pinos do encoder KY-040 utiliado. Fonte: Henry's-Bench (2018)

Para achar o número de posições basta verificar os pinos *A* (CLK) e *B* (DT) segundo o fluxograma da figura 2.5.

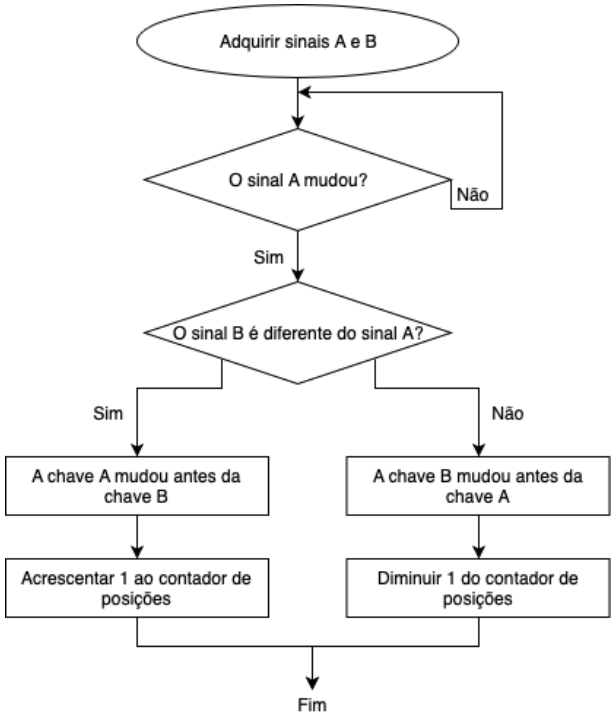

Figura 2.5: Fluxograma do algorítmo para obtenção das posições do encoder. Fonte: A autora, 2018.

2.4 Sensor de velocidade

Para o sensor de velocidade, o sensor Hall foi escolhido com um ímã no rotor. Foi decidido usar um sensor diferente do de posição pelo fato de que a operação de derivação da posição poderia ser complicada e tomar muito tempo.

Como o sensor Hall além de medir corrente pode também medir campo magnético, acoplando o ímã ao rotor, o campo magnético aumenta cada vez que o ímã se aproxima do sensor. Logo a velocidade pode ser calculada a partir do tempo entre dois picos do campo magnético (t_{picos}) a partir da equação (2.3).

$$velocidade = \frac{2 \cdot \pi}{t_{picos}} \tag{2.3}$$

O sensor Hall escolhido para medir a velocidade foi o 3144. O seu módulo apresentado na figura 2.6 possui uma saída analógica e uma saída digital. A saída digital é ideal para medir a velocidade uma vez que toda vez que medir um campo sua saída passa para o nível baixo.

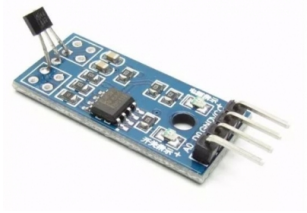

Figura 2.6: Sensor de efeito Hall A3144. Fonte: Cerne (2018)

2.5 NI USB 6009

O dispositivo da *National Instruments* NI USB 6009 é um dispositivo que permite a aquisição de dados. Possui 13 entradas ou saídas digitais, 2 entradas analógicas para sinais de até 150Hz e 8 saídas analógicas (14 bits, 48 kS/s).

Ele será o dispositivo usado para fazer a interface entre software e hardware na utilização *hardware-in-the-loop*. A escolha foi feita pela disponibilidade do dispositivo. Ele será usado para mandar os sinais de controle do controlador para o hardware e ler os sinais recebidos dos sensores de posição e velocidade. Além disso também será utilizado na identificação de sistemas para a aquisição dos sinais de entrada e saída da planta. Para tal basta instalar o *driver* DAQmx disponível gratuitamente para que o dispositivo possa se comunicar com o Matlab e o Simulink.

O componente NI USB 6009 é apresentado na figura 2.7.

Figura 2.7: NI USB 6009. Fonte: National-Instruments (2018)

CAPÍTULO 3

IDENTIFICAÇÃO DE SISTEMAS

Para que se possa sintonizar bem os controladores, é importante ter um modelo matemático preciso do hardware. Por isso a identificação de sistemas é um passo muito importante em sistemas de controle.

A identificação de sistemas é usada para achar a funções de transferência que descreve o comportamento do sistema. Ela liga uma entrada a uma saída a partir de uma função matemática.

O primeiro passo para se realizar a identificação de sistemas é a aquisição dos sinais de entrada e saída do motor (sistema em malha aberta). Para tal o dispositivo da National Instruments NI USB-6009 será utilizado para fazer a conexão entre o hardware (motor) e o software Matlab. Como o que se deseja controlar são velocidade e posição, os sinais que devem ser adquiridos devem relacionar a tensão aplicada ao motor à velocidade resultante. O sinal de posição pode ser adquirido a partir da integração dessa velocidade.

A estratégia utilizada para se obter a velocidade de rotação em função da tensão aplicada foi de utilizar como sensor de velocidade um motor CC cujo rotor foi acoplado ao rotor do motor universal que se deseja identificar. Nessa configuração, a velocidade de rotação do motor CC é proporcional à tensão em seus terminais. Um diagrama do sistema utilizado para a aquisição de pontos é mostrado na figura 3.1.

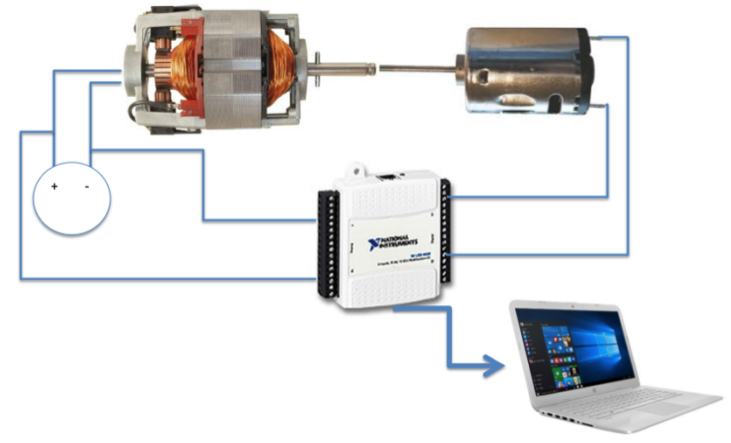

Figura 3.1: Esquemático do sistema utilizado para fazer a identificação de sistemas. Fonte: A autora, 2018.

Os sinais de entrada e saída foram obtidos com um tempo de amostragem de 0.1ms. Foram feitas aquisições com duas amplitudes diferentes de sinais de entrada: aproximadamente 12V e aproximadamente 20V. O interesse dos dois sets de amostras é de utilizar um para poder validar o outro. Os sinais de entrada e saída podem ser encontrados na figura 3.2.

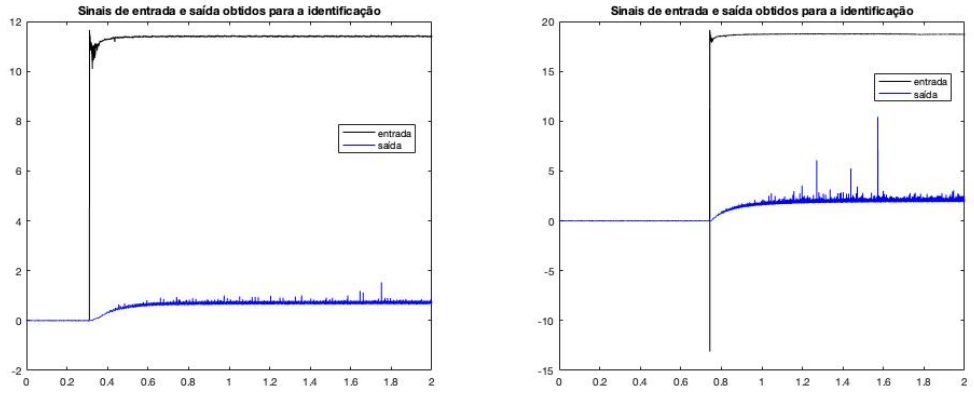

Figura 3.2: Sinais de entrada e saída obtidos na aquisição de dados para a identificação do sistema para uma tensão de aproximadamente 12V e 20V de entrada. Fonte: A autora, 2018.

Como os sinais apresentam um ruído importante, para realizar a identificação da função de transferência propriamente dita, foi necessário filtra-los. A filtragem foi feita por um filtro do tipo FIR com uma janela do tipo *kaiser* com frequências de corte de 80Hz e 120Hz. Os sinais filtrados se encontram na figura 3.3.

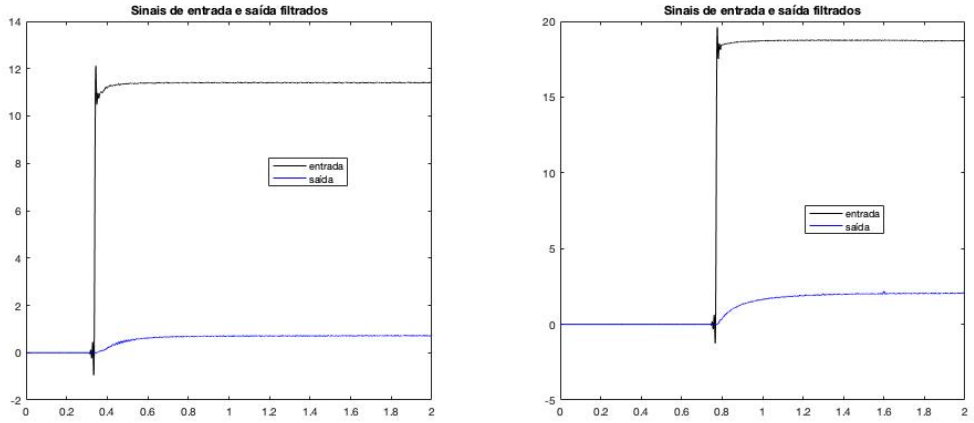

Figura 3.3: Sinais de entrada e saída filtrados. Fonte: A autora, 2018.

3.1 Métodos para Identificação de Sistemas

3.1.1 Identificação modelo ARMAX

Existem alguns métodos já difundidos para realizar a identificação de sistemas como para o modelo ARMAX o qual possui uma função de transferência do tipo dado pela equação (3.1).

$$A(q)y(k) = B(q)U(k) + C(q)y(k) \tag{3.1}$$

O Matlab possui uma função intrínseca utilizada para fazer a identificação de sistemas desse tipo chamada função ARMAX.

O melhor resultado foi obtido a partir de uma entrada de aproximadamente 12V e validação feita para uma entrada de aproximadamente 20V.

Os resultados se encontram na 3.4, o ajuste da identificação do sistema se encontra na figura 3.5.

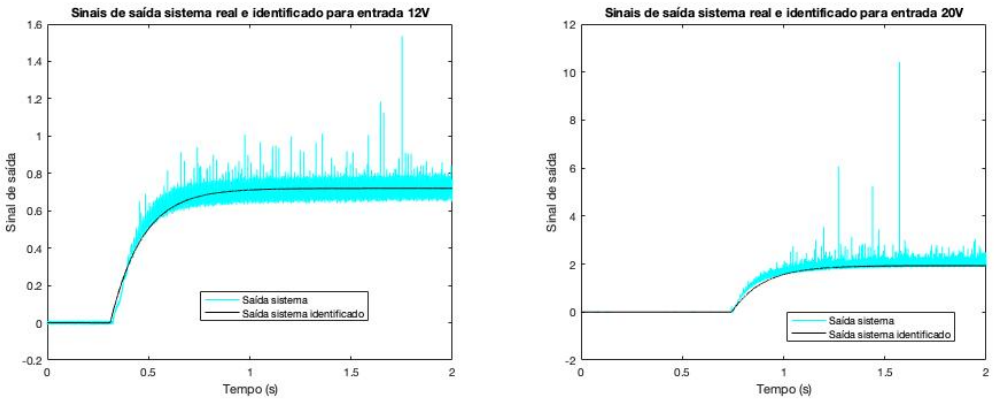

Figura 3.4: Saída do sistema identificado em relação ao sinal de saída do sistema aplicando 12V e 20V na entrada para identificação pela função ARMAX. Fonte: A autora, 2018.

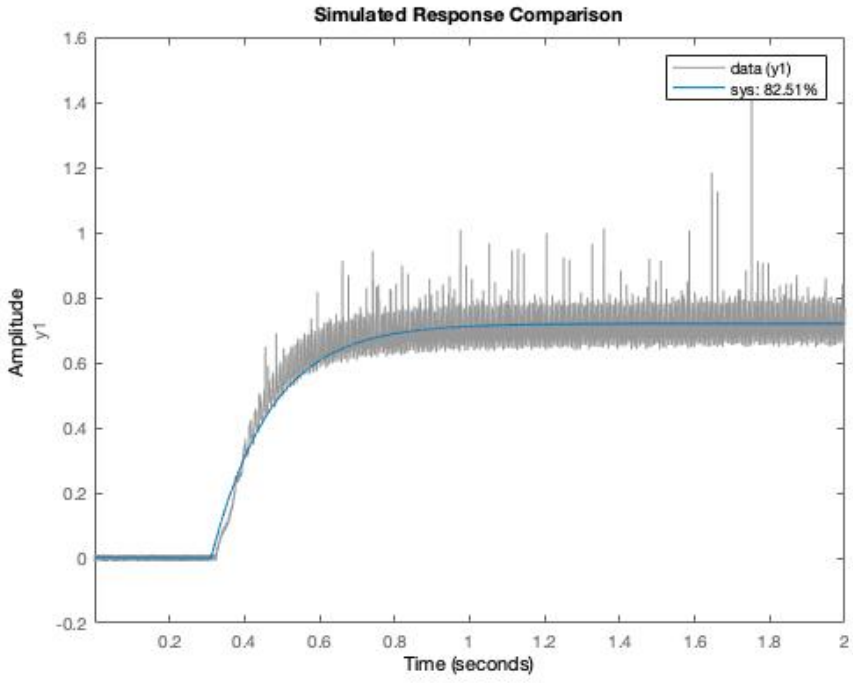

Figura 3.5: Ajuste da identificação de sistema pela função ARMAX. Fonte: A autora, 2018.

3.1.2 Metaheurística ED (Evolução Diferencial)

Uma vez que nesse projeto metaheurísticas estão sendo utilizadas, e uma vez que elas podem ser aplicadas a todos os tipos de sistema, elas também vão ser utilizadas

para fazer a identificação do motor universal uma vez que a identificação nada mais é do que um problema de otimização.

Existem inúmeras metaheurísticas usadas hoje. A maioria delas inspira-se em eventos da natureza e da biologia para inspirar as iterações do método. As metaheurísticas mais utilizadas são as “evolucionárias” que se baseiam no comportamento de cromossomos e outras dinâmicas da evolução. Dentre elas a mais conhecida é o Algorítmo Genético (GA). Além dessas exitem vários outros tipos de metaheurísticas como por exemplo a Grey Wolf Optimization (SHAYANFAR H. A. ; SHAYEGHI, 2016) que se baseia em como uma alcatéia de lobos cinzentos se comporta em bando.

Nesse trabalho a metaheurística que será estudada será a Evolução Diferencial (ED). Essa metaheurística é parte integrante dos algorítimos “evolucionários” como o GA. Ela também é inspirada no comportamentos dos cromossomos como para o algoritmo genético, mas ao contrário dele a seleção das melhores soluções é feita depois de mutar e fazer o cross-over da população inicial.

Como se trata algoritmo evolucionário, a ideia geral por traz da metaheurística ED é começar com uma população inicial de ”cromossomos”criada aleatoriamente e fazer com que essa população evolua a partir de mutações e *cross-over*. Ao final os ”cromossomos”selecionados serão aqueles que minimizem ou maximizem um certo critério pré determinado.

Em geral, o algoritmo inspirado no trabalho de (COSTA, 2012) pode ser descrito pelo fluxograma da figura 3.6.

A população inicial é dada por um conjunto de cromossomos em forma de matriz cujo número de linhas *i* mostra o tamanho da população (P definida pelo usuário) e cujo número de colunas *j* mostra o número de variáveis a serem determinadas. Cada um desses cromossomos é gerado a partir da equação (3.2).

$$X_{i,j} = X_{min} + rand[0,1] \cdot X_{max} - X_{min} \tag{3.2}$$

Em que $X_{i,j}$ representa o cromossomo da linha *i* e coluna *j*, X_{max} e X_{min} representam respectivamente o valor máximo e mínimo que os coeficientes podem atingir.

O próximo passo seria a mutação. Para tal três linhas de cromossomos X_α, X_β e X_γ são selecionados aleatóriamente para criar o vetor de cromossomos mutados *v*. Isso é feito de acordo com a equação (3.3).

$$v_{i,:} = X_\alpha + F_e \cdot X_\beta - X_\gamma \tag{3.3}$$

Nesse caso, F_e é o *Fator de escala* definido por uma constante.

Para que o critério de valores máximos e mínimos seja respeitado, é necessário verificar se o vetor resultante não está fora do intervalo definido por eles. Se não estiver contido no intervalo, é necessário substituir o valor por um dos limites.

Uma vez feita a mutação, o cross-over deve ser realizado. Nessa parte, um novo vetor *u* será criado a partir dos cromossomos originais da linha *i* ou a partir do vetor de cromossomos mutados a partir do *cross-over binomial* o qual é calculado a partir do sitema de equações:

$$\begin{cases} u_{i,:} = v_{i,:}, & se \;\; r < \eta_{CR} \\ u_{i,:} = X_{i,:}, & se \;\; r \geq \eta_{CR} \end{cases}$$

Em que *r* é um valor aleatório entre [0,1] e η_{CR} é a constante de cross-over.

Uma vez feito o cross-over, a nova linha de cromossomos criados deve ser avaliada com o critério de decisão para ver se ela é melhor e deve substituir a linha atual ou para ver se ela é pior e deve ser descartada.

O critério escolhido para a decisão foi o ITAE (Integral do módulo do erro multiplicado pelo tempo), dado pela equação (3.4).

$$ITAE = \int_0^T t \cdot |e(t)|\, dt \tag{3.4}$$

Esse critério deve ser minimizado pela solução ótima. Para o caso da utilização da metaheurística para a identificação de sistemas o erro é dado pela diferença entre o sinal de saída do sistema identificado e o sinal de saída do sistema real. Para o caso da sintonia do controlador PID, o erro é dado pela diferença entre o sinal de saída do sistema em malha fechada formado pelo controlador e o sistema e o sinal de entrada.

Logo a solução que tiver o menor ITAE (entre a nova linha de cromossomos e a linha anterior) é mantida enquanto a outra é descartada.

Esse processo continua até que toda a população seja avaliada pelo número de gerações desejado. No final a linha de cromossomos selecionada é aquela que possui o menor ITAE e os cromossomos de cada coluna da linha são ou os coeficientes da função de transferencia sendo identificada ou os coeficientes do controlador PID.

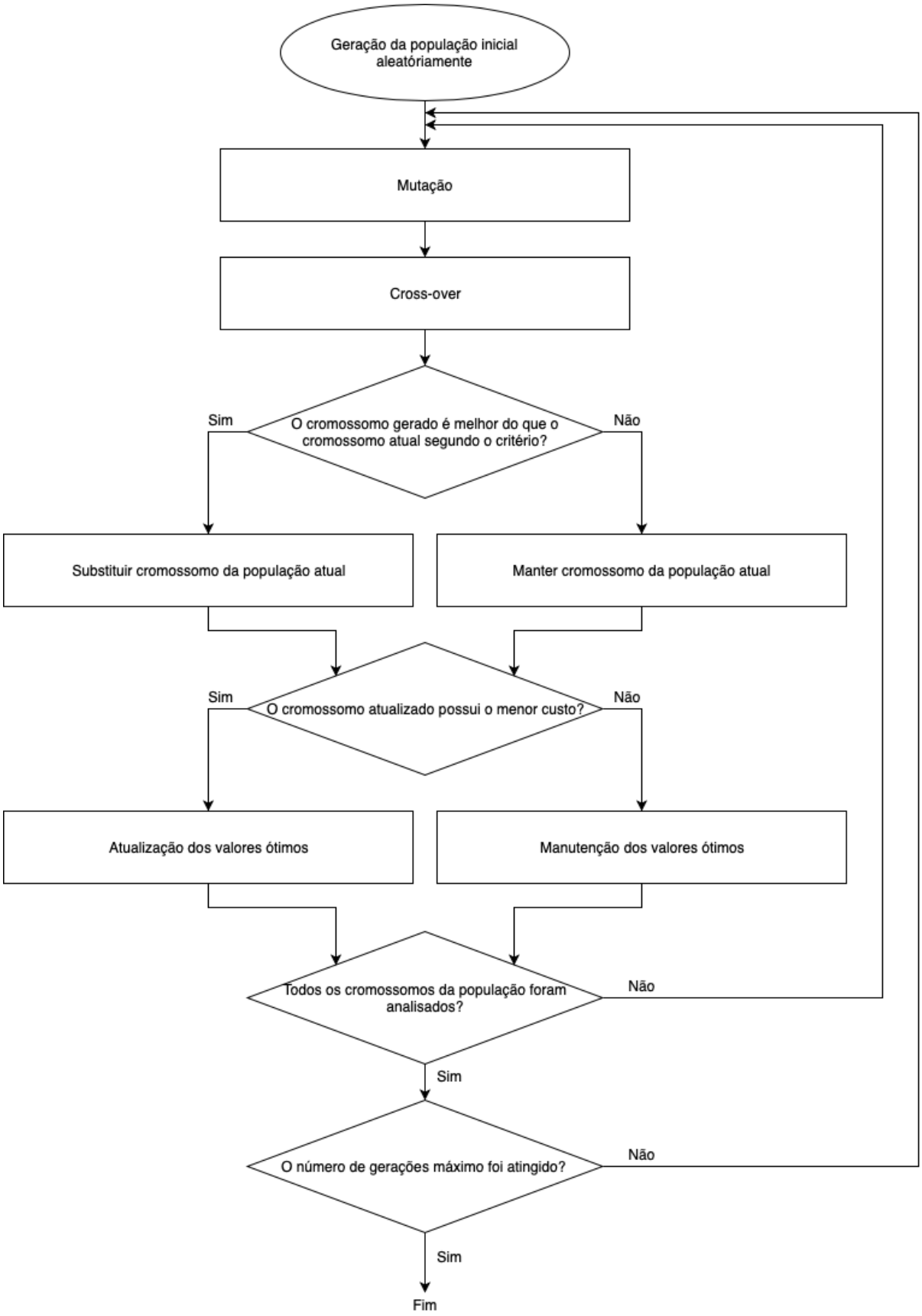

Figura 3.6: Fluxograma do algorítmo da metaheurística ED. Fonte: A autora, 2018.

Aplicando esse método, a melhor resposta obtida foi para identificação feita com sinal de entrada de aproximadamente 12V e validação feita para sinal de entrada de aproximadamente 20V com apenas um ajuste no ganho.

Os resultados se encontram na figura 3.7, o ajuste da resposta é dado na figura 3.8 e a convergência se encontra na figura 3.9.

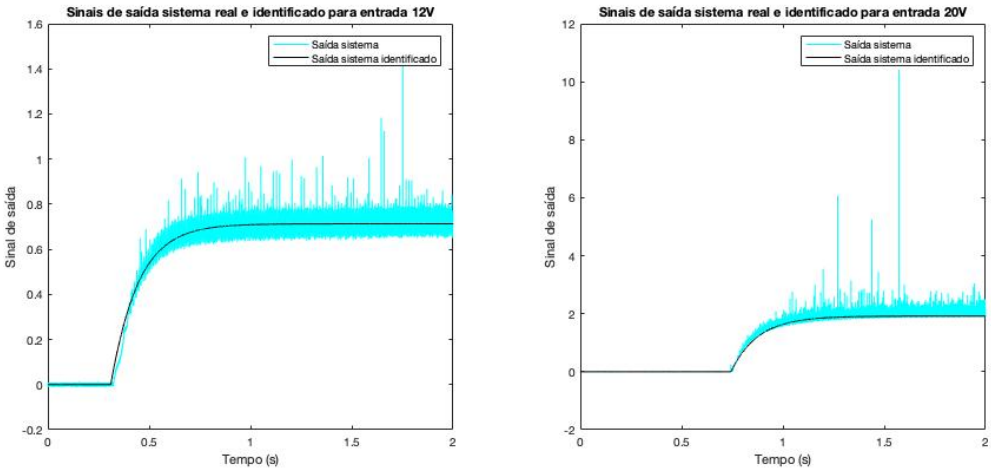

Figura 3.7: Saída do sistema identificado em relação ao sinal de saída do sistema aplicando 12V e 20V na entrada para identificação por metaheurística. Fonte: A autora, 2018.

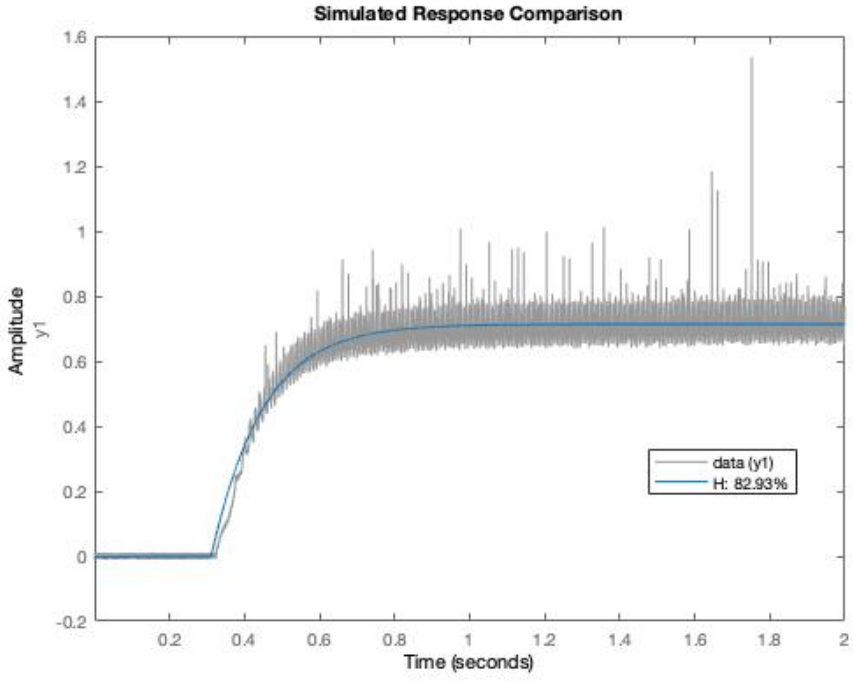

Figura 3.8: Ajuste da identificação de sistema por metaheurística. Fonte: A autora, 2018.

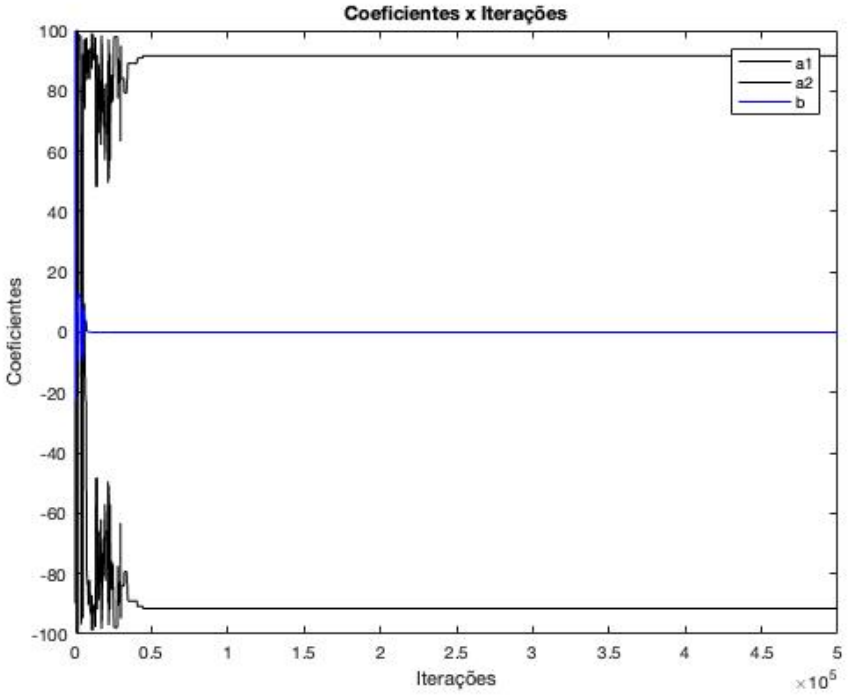

Figura 3.9: Convergencia dos coeficientes. Fonte: A autora, 2018.

3.1.3 Função de transferência do sistema

Mesmo com uma pequena diferença, o modelo identificado pela metaheurística é mais preciso. Isso se da tanto pelo fato de ele possuir um melhor ajuste (*fit*) tanto pelo fato de ele possuir uma resposta visivelmente melhor para o conjunto de pontos de validação para entrada 20V. Dessa a função de transferência é dada pela equação (3.5).

$$H(s) = \frac{0.0044}{0.0703 + 0.0093 \cdot s} \tag{3.5}$$

Além disso, o sistema final encontrado foi estipulado de primeira ordem, mas para poder representar da melhor maneira possível o sistema em simulação, foi decidido que seria mais fácil representar o motor por sua função de transferência geral dada pela equação (2.1) apresentada no capítulo 2.

O motivo pelo qual o sistema pode ser representado como de primeira ordem é o fato de que a resistencia e indutância do motor são muito pequenas. Basta então encorpora-las na função de transferência. De fato, ao medi-las a resistencia apresentom um valor de 15Ω e a indutancia um valor de 35mH.

A função de transferencia final é mostrada pela equação (3.6).

$$H(s) = \frac{0.0044}{0.0703 + 0.0093 \cdot s} \cdot \frac{a}{L \cdot s + R} = \frac{0.1222}{0.0006001\hat{s}2 + 0.2617 \cdot s + 1.954} \tag{3.6}$$

Onde *a* é uma constante. Dessa forma as variáveis da função (2.1) podem ser facilmente encontradas e o motor facilmente identificado.

CAPÍTULO 4

SINTONIA DOS CONTROLADORES PID

Uma vez definidos os componentes do hardware e a função de transferência do motor, o próximo passo seria a implementação dos controladores para ambas as malhas de velocidade e posição.

4.1 Controladores PID

Como dito previamente, controles do tipo PID surgiram há muito tempo (1940) mas continuam sendo os controladores mais usados na indústria hoje devido a sua simplicidade. Um controlador desse tipo possui três componentes: Uma componente proporcional (P), uma componente derivativa (D) e uma componente integrativa (I), daí o nome PID. Cada parte tem uma função específica as quais são mostradas na figura 4.1.

	K_p	K_i	K_d
Tempo de subida	Diminui	Diminui	Muda pouco
Sobressinal	Aumenta	Aumenta	Diminui
Tempo acomodação	Muda pouco	Aumenta	Diminui
Erro no equilibrio	Diminui	Elimina	Nenhum

Figura 4.1: Efeito de cada ganho do controlador PID em alguns parâmetros da resposta. Fonte: SERAPIÃO, A. B. S. , 2011.

A equação de um controlador PID é dada pela equação (4.1).

$$C(s) = \frac{K_i}{s} + Kp + Kd \cdot s \tag{4.1}$$

4.1.1 Controlador PID por controle de modelo interno IMC

O controlador por modelo interno (IMC) é uma técnica de sintonia de controladores PID que tem por objetivo desconsiderar erros de modelagem do sistema. Parte-se de uma estrutura do tipo representado na figura (4.2) e reestrutura-se a malha para que ela fique da forma mostrada na figura (4.3).

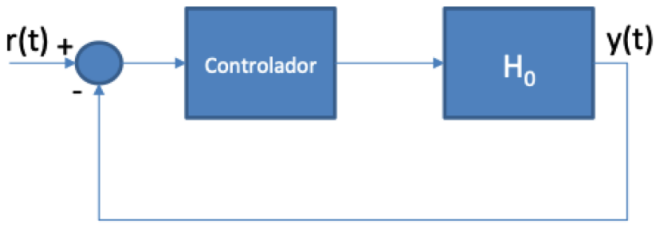

Figura 4.2: Malha original formada por controlador mais função de transferência. Fonte: A autora, 2018.

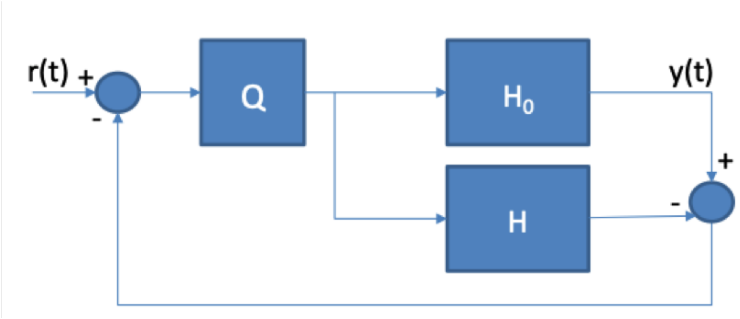

Figura 4.3: Malha reestruturada para projeto do controlador PID a partir do IMC. Fonte: A autora, 2018.

Nesse caso, *H* é o modelo identificado do sistema e H_0 a função de transferência real. O controlador *C* é encontrado a partir do bloco *Q*, como indicado pela equação (4.2).

$$C(s) = \frac{Q(s)}{1 - Q(s) \cdot H(s)} \tag{4.2}$$

Por sua vez, a partir do rearranjo da malha e levando em conta critérios de estabilidade e causalidade, a equação que descreve *Q* é dada por (4.3).

$$Q(s) = \frac{1}{H_{-}(s)} \cdot F(s) \tag{4.3}$$

Em que $H_{-}(s)$ é a função de transferência sem contar atrasos e zeros de fase não mínima e *F(s)* é um filtro de função de transferência (4.4) que serve para deixar a função de transferência causal a partir de um *n* suficientemente grande para tornar Q causal e uma constante α a ser escolhida.

$$F(s) = \frac{1}{(\alpha \cdot s + 1)^n} \tag{4.4}$$

O controlador final é composto pelo controlador PID já descrito seguido de um filtro.

Seguindo esse passo a passo, um *script* foi criado no software Matlab para realizar as equações necessárias. O fluxograma que descreve o código é descrito na figura 4.4.

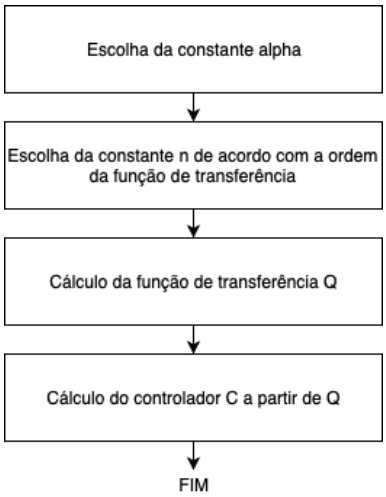

Figura 4.4: Fluxograma para o código IMC. Fonte: A autora, 2018.

Os valores de α para as sintonias foi escolhido a partir de tentativa e erro e os valores de *n* foram escolhidos a partir da análise da função de transferência da malha. O α para a sintonia do controlador de velocidade foi escolhido como 2 e para o controlador de posição como 0.1. O valor de *n* para deixar o primeiro controlador causal foi de 2 e para deixar o segundo causal foi de 3.

O controlador de velocidade obtido para esse caso se encontra na equação (4.5) e o controlador de posição se encontra na equação (4.6), no caso as equações representam um controlador PID seguido de um filtro. A resposta em malha fechada de velocidade e posição se encontram na figura 4.5. As equações foram simplificadas pelo fato do controlador final possuir grau alto com coeficientes muito próximos de zero.

$$C(s) = \frac{3.602 \cdot 10^{-7}s^4 + 0.0003141s^3 + 0.07085s^2 + 1.023s + 3.819}{4.693 \cdot 10^{-7}s^4 + 0.0002164s^3 + 0.006645s^2 + 0.03821s} \tag{4.5}$$

$$C(s) = \frac{0.0001656s^7 + 0.001947s^6 + 0.01471s^5 + 0.06941s^4 + 0.1864s^3 + 0.2177s^2}{3.288 \cdot 10^{-5}s^7 + 0.0004224s^6 + 0.003486s^5 + 0.01789s^4 + 0.05198s^3 + 0.06531s^2} \tag{4.6}$$

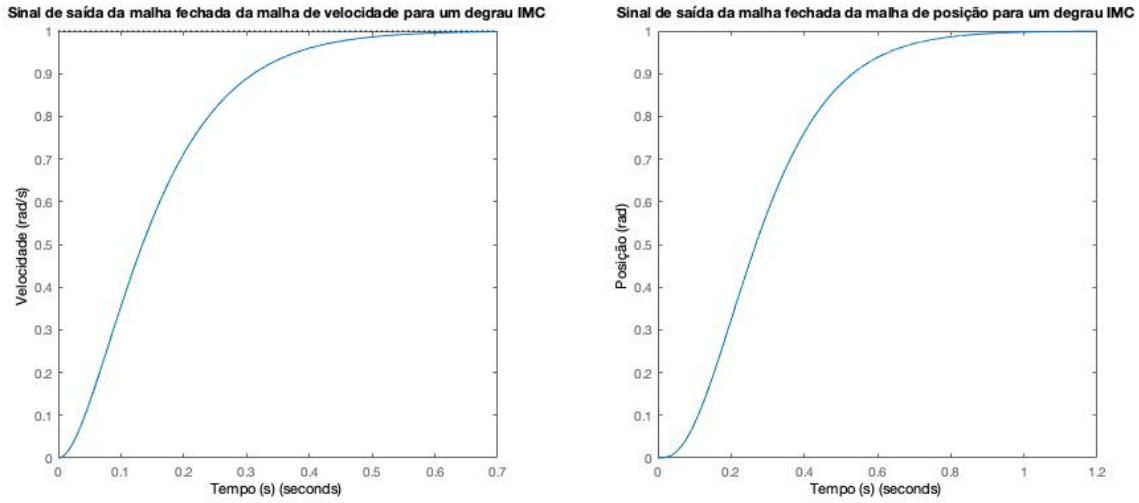

Figura 4.5: Saida da malha de velocidade e posição para um degrau para sintonia IMC. Fonte: A autora, 2018.

4.1.2 Controlador PID sintonizado por lugar das raízes

A ideia de se fazer a sintonia de um controlador PID a partir da análise do lugar das raízes vem do fato de que os polos de uma função de transferência controlam de maneira direta o comportamento do sistema em termos de tempos, sobre-sinais e erros.

Dessa forma, o objetivo é de analisar os polos em malha fechada do sistema formado pela planta e pelo controlador, uma vez que eles estabelecem as performances desejadas em malha fechada. A sintonia dos coeficientes será feita baseando-se nisso.

Para determinar o polo a partir das performances desejadas tem-se as relações dadas na equação (4.7).

$$s = -(cos(\xi))^{-1} + -j \cdot w_d \tag{4.7}$$

Em que ξ é o coeficiente de amortecimento e w_d é a frequência natural amortecida.

Uma vez determinados os polos, para que se possa determinar os coeficientes do controlador PID deve-se aplicar os critérios de ângulo (4.8) e módulo (4.9).

$$\angle C(s)H(s) = +-180^o \tag{4.8}$$

$$|C(s)H(s)| = 1 \tag{4.9}$$

Seguindo o passo a passo descrito, os controladores podem ser encontrados. Porém para esse caso, ao invés da utilização de um *script* nesse caso, o programa Matlab possui uma função chamada *sisotool* que abre uma interface que permite a realização do design de controladores de forma iterativa.

A interface permite a análise dos polos do sistema mais o controlador e permite ainda que o usuário escolha as características desejadas de resposta a partir

de linhas complementares que delimitam o plano dos zeros e polos. A chamada da função e a interface utilizada se encontram na figura 4.6.

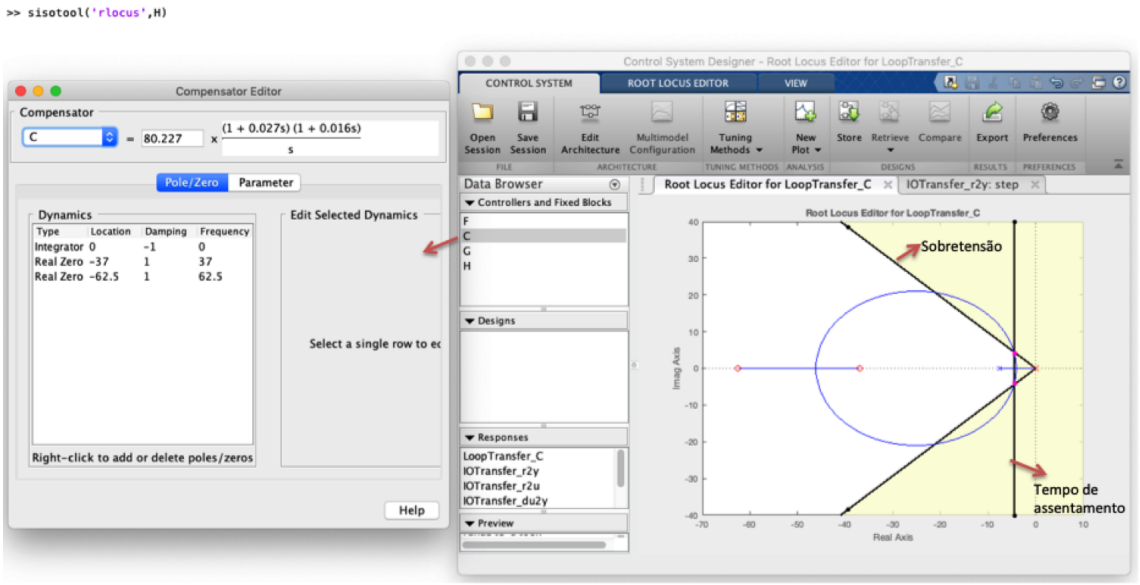

Figura 4.6: Interface *sisotool* do programa Matlab utilizada para sintonia do controlador PID por lugar das raízes. Fonte: A autora, 2018.

Destacadas nas figuras estão as linhas de critérios utilizadas para um tempo de assentamento de 0.8s para a malha de velocidade e de 0.1s para a malha de posição e uma sobretenção de 5% considerados para a sintonia dos dois controladores. O controlador pode ser projetado clicando em *C* como indicado na figura. Basta adicionar dois zeros reais e um integrador e move-los no Root Locus Editor para que eles encontrem as performances desejadas previamente definidas.

Para o caso da sintonia feita por lugar das raízes, a função de transferência do controlador de velocidade é dada pela equação (4.10) e do controlador de posição é dada pela equação (4.11). As respostas das malhas fechadas se encontram na figura 4.7.

$$C(s) = 0.04549 \cdot s + \frac{105.3}{s} + 4.527 \tag{4.10}$$

$$C(s) = 52.26 \cdot s + \frac{52.26}{s} + 104.5 \tag{4.11}$$

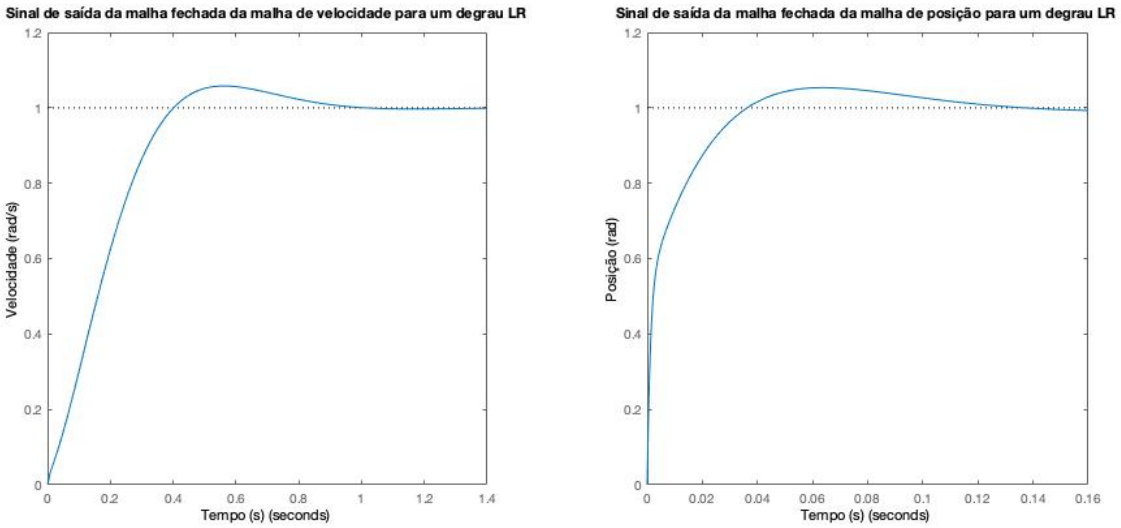

Figura 4.7: Saida da malha de velocidade e posição para um degrau para sintonia LR. Fonte: A autora, 2018.

4.1.3 PID pela metaheurística ED

Para sintonizar o controlador a partir da metaheurística ED foi criado um *script* no Matlab que segue o fluxograma descrito na figura 3.6. Para o caso da sintonia do controlador PID, o critério de custo de cada solução calculado pelo ITAE considera o erro entre a resposta do sistema em malha fechada e a referência do sistema.

Os valores limites para coeficientes foram entre 0 150 para o controlador de velocidade e entre 0 e 100 para o controlador de posição. O fator de escala F_e utilizado foi de 0.6 e a constante de cross-over η_{CR} foi de 0.7.

Aplicando a sintonia por metaheurística ED, as equações dos controladores de velocidade e posição se encontram em (4.12) e (4.13) respectivamente. As respostas das malhas para um degrau de entrada se encontram na figura 4.8.

$$C(s) = 1 \cdot 10^{-6} \cdot s + \frac{150}{s} + 21.8 \tag{4.12}$$

$$C(s) = 9.878 \cdot s + \frac{0.001}{s} + 100 \tag{4.13}$$

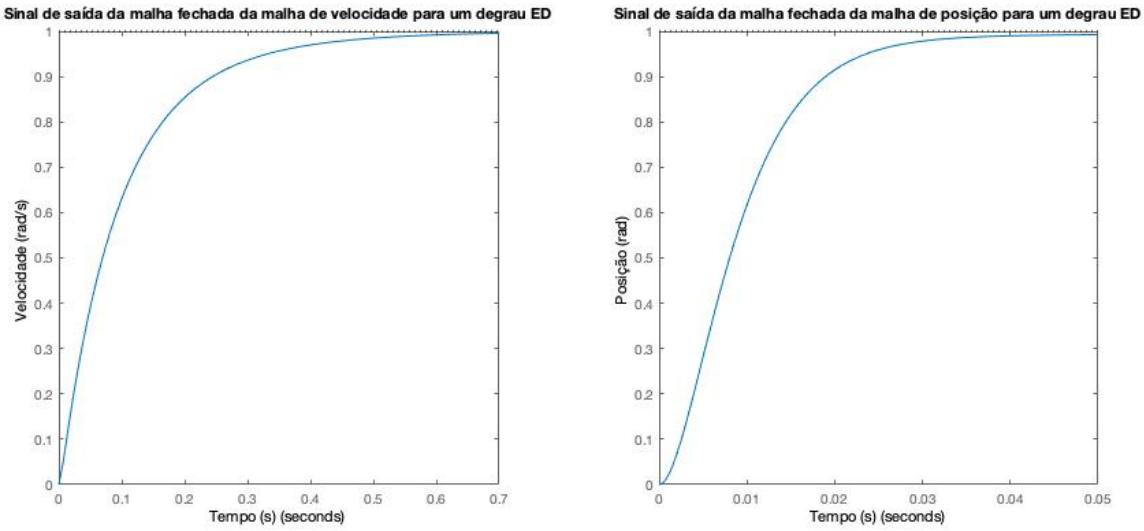

Figura 4.8: Saida da malha de velocidade e posição para um degrau para sintonia ED.
Fonte: A autora, 2018.

CAPÍTULO 5

RESULTADOS

5.1 Simulação do sistema

Uma vez determinados o sistema e os controladores é necessário simular todo o sistema antes de passar para a realização do *hardware-in-the-loop* propriamente dito.

5.1.1 Simulação do hardware

O hardware para esse projeto é formado pelo motor universal que se deseja controlar, uma ponte H de mosfets utilizada para possibilitar a rotação do motor em dois sentidos e os sensores de posição e velocidade.

Para simular o motor a escolha foi feita de utilizar a estrutura de um motor CC do Simulink. Em que os parâmetros são obtidos a partir da função de transferencia identificada. Além disso foram usados os componentes mosfet e resistencias e sensor ideal de posição e velocidade do Simscape. A simulação do hardware se encontra na figura 5.1.

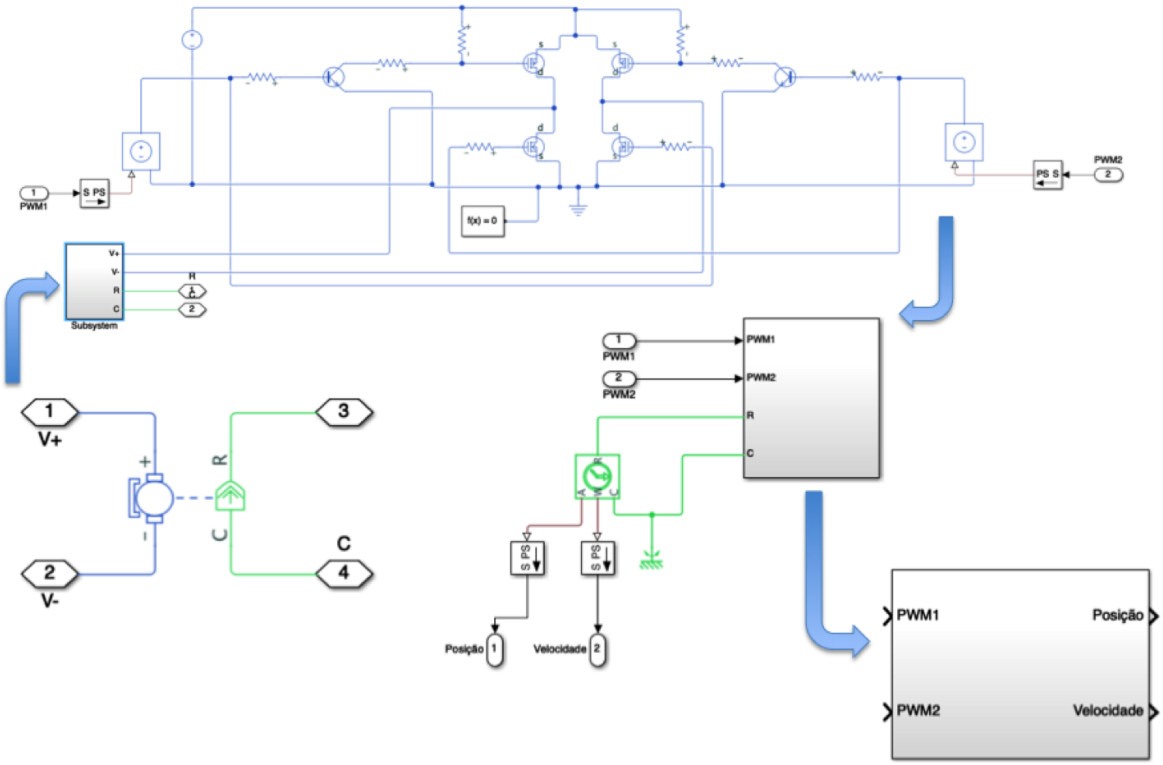

Figura 5.1: Simulação do hardware no Simulink. Fonte: A autora, 2018.

Essa parte da simulação será mais tarde substituida na prética pelo verdadeiro hardware.

5.1.2 Transfromação do sinal de controle de velocidade em sinais de PWM

Como o controlador de velocidade foi feito considerando uma função de transferência que relaciona a tensão de entrada com a velocidade de rotação, a sua saída equivale a tensão que deve ser aplicada ao motor. Essa tensão pode ser tanto positiva quanto negativa. O sinal referencia o sentido de rotação.

Como se tratam de dois sentidos de rotação foi decidido utilizar a ponte H. Para controlá-la é necessário gerar dois sinais de PWM, um para rotação em um sentido e outro para rotação em outro sentido. Além disso enquanto um PWM está ativado o outro deve ser nulo e é preciso garantir um atraso enquanto o rotor muda de posição para não curto-circuitar o motor. Logo, o sinal de tensão na saída do controlador de velocidade deve ser transformado em *dutycicle*. Esse sinal com o valor de *dutycicle* pode ser aplicado diretamente ao bloco do Simulink *PWM Generator*. A amplitude da tensão de saída desse bloco é booleana e para a simulação vai ser multiplicada por 5 para controlar os transistores bipolares.

Além disso, o sinal de saída do controlador deve ser saturado para que ele não ultrapasse a tensão de alimentação máxima da ponte H. O esquemático do Simulink utilizado para realizar os sinais PWM se encontra na figura 5.2.

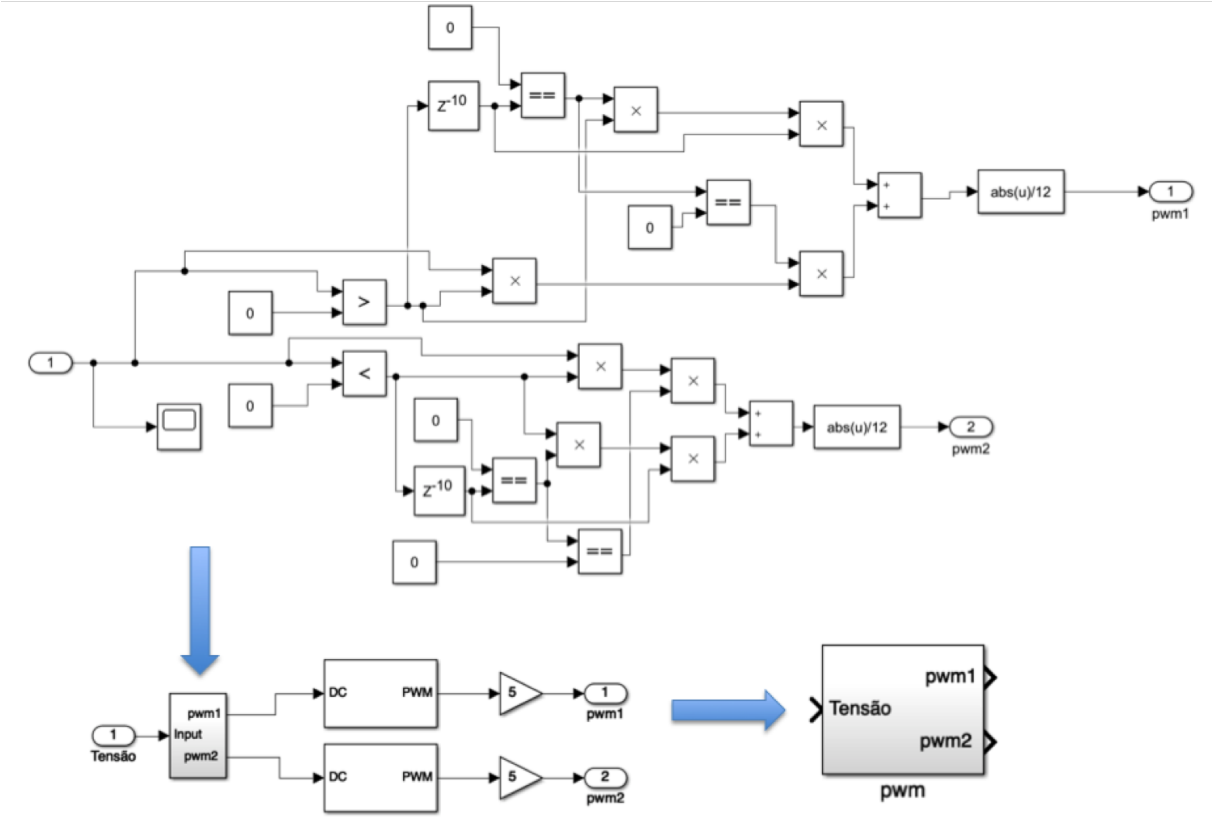

Figura 5.2: Transformação do sinal de velocidade em sinal de pwm para a ponte H. Fonte: A autora, 2018.

5.1.3 Simulação completa

A partir da simulação do motor e da transformação do sinal de controle em dois sinais de PWM, basta fechar as malhas de posição e de velocidade para poder realizar a simulação. A simulação completa se encontra na figura 5.3.

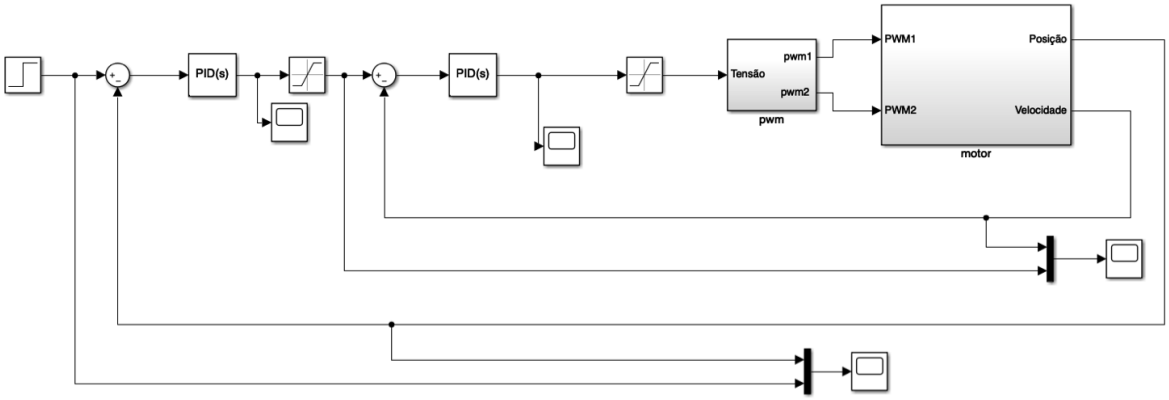

Figura 5.3: Simulação completa do sitema: Malhas de velocidade e posição. Fonte: A autora, 2018.

Além da saturação já mencionada na saída do controlador de posição, também foi necessário adicionar uma saturação na saída do controlador de posição. No caso, o sinal de saída do controlador de posição é a referência de velocidade. Como a tensão de alimentação do motor é limitada, a velocidade máxima de rotação também será limitada por isso a necessidade de uma saturação.

5.1.4 Simulação de atrasos

Para poder realmente comparar as performances dos controladores escolhidos, é necessário analisar seu comportamento quando um atraso é instroduzido ao sistema. Para introduzir esse atraso, o bloco *delay* do Matlab foi utilizado em que se escolhe como argumento quantas amostras o bloco atrasará. Foram feitas simulações com atraso na realimentação e no sinal de controle. O caso de atraso do sinal de controle é mostrado na figura 5.4.

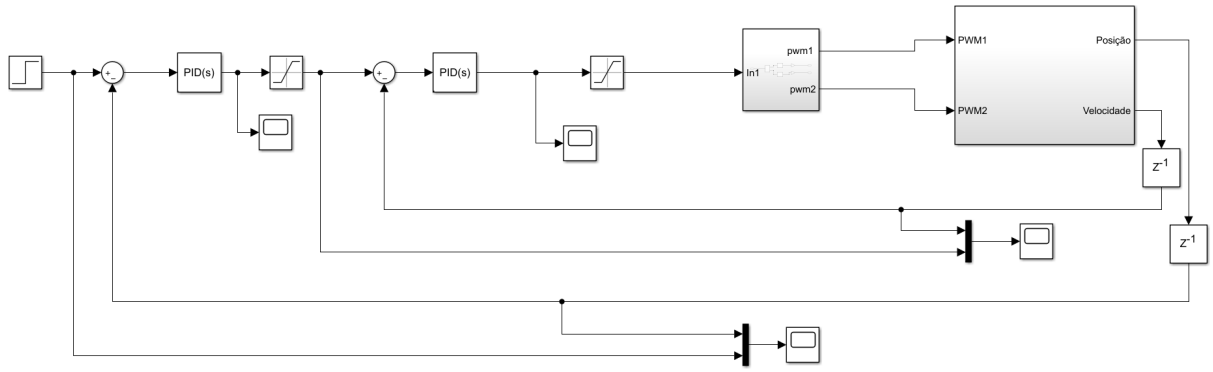

Figura 5.4: Simulação com atraso na realimentação de algumas amostras. Fonte: A autora, 2018.

O número de amostras de atraso foi escolhido de acordo com o tempo de amostragem com valores de 5 e 10 amostras para 5ms e 25 e 50 amostras para 1ms.

Uma vez montadas as malhas no Simulink, foi necessária a realização de simulações. Dessa maneira se pode garantir que os controladores escolhidos ainda apresentam uma boa resposta mesmo considerando as saturações necessárias e também, garantir que o sinal de controle seja bem transformado em sinais de PWM para controlar a ponte H.

Além disso a simulação permite analisar a resposta do sistema em situações em que se tem atraso de sinal.

5.1.5 PID por IMC

A resposta em simulação do controlador PID sintonisado a partir do controle por modelo interno IMC se encontra na figura 5.5.

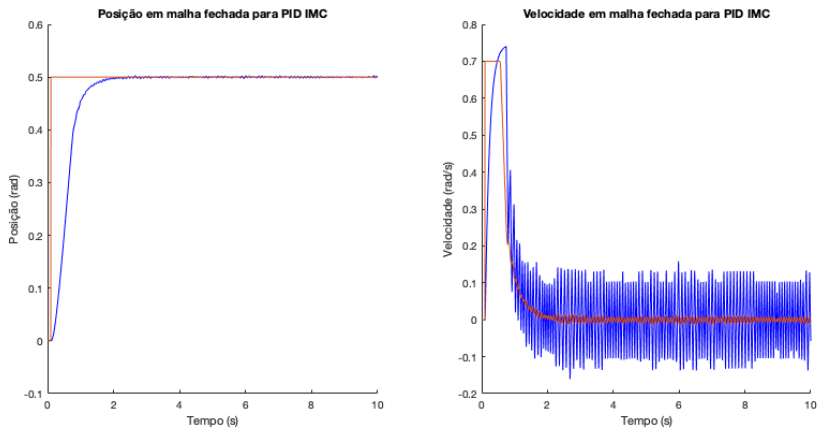

Figura 5.5: Saida da malha de posição e velocidade (azul) em relação às referências (laranja) para sintonia IMC. Fonte: A autora, 2018.

Pode-se notar que as saturações da simulação fazem com que a resposta em malha fechada torne-se mais lentas do que as previstas. E que a resposta em posição

nesse caso é melhor do que a resposta em velocidade porque o sinal de referencia muda mais devagar.

O próximo passo consiste em analisar a resposta quando se adiciona um atraso na realimentação e um atraso no sinal de comando. Os atrasos foram colocados em termos de amostras perdidas em dois valores para dois tempos de amostragem diferentes (5 e 10 amostras para 5ms e 25 e 50 amostras para 1ms).

Em geral, as respostas para atraso na realimentação foram as mesmas do que para atraso no sinal de controle. As respostas com atraso são mostradas na figura 5.6. Pode-se notar que o sistema ainda tem uma resposta aceitável principalmente em se tratando de posição uma vez que o sinal de referência para esse caso é mais lento do que para a velocidade.

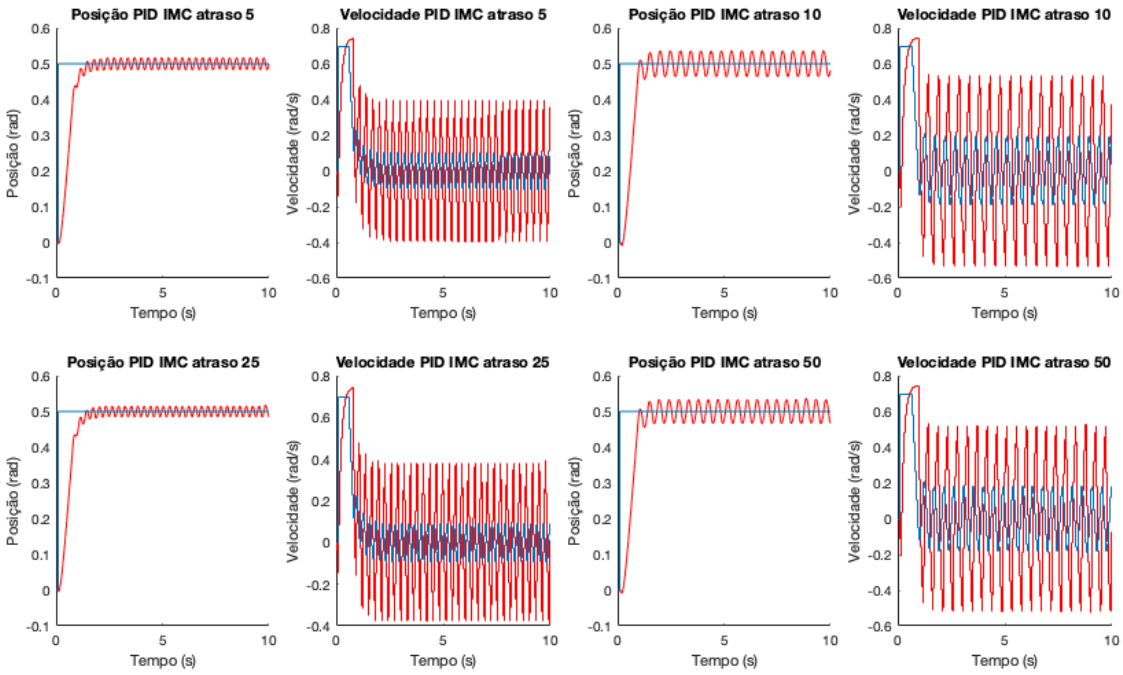

Figura 5.6: Saida da malha de posição e velocidade (laranja) em relação às referências (azul) para sintonia IMC com atraso. Fonte: A autora, 2018.

5.1.6 PID por lugar das raízes LR

A resposta em simulação do controlador PID sintonisado a partir do lugar das raízes LR se encontra na figura 5.7.

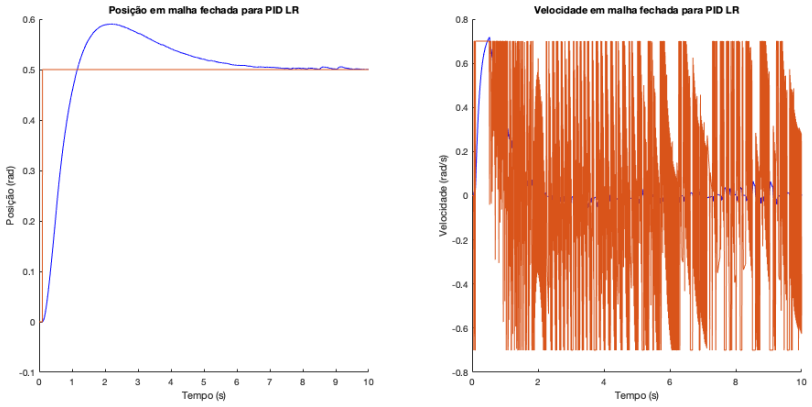

Figura 5.7: Saida da malha de posição e velocidade (azul) em relação às referências (laranja) para sintonia LR. Fonte: A autora, 2018.

Pode-se notar também para esse caso que as saturações da simulação fazem com que a resposta em malha fechada torne-se mais lentas. Em comparação com a sintonia IMC, o sinal de controle na saída do controlador tem uma amplitude bem mais elevada. Mais uma vez, a velocidade tem resposta pior do que a posição pelo fato de que sua referência muda com uma velocidade maior.

Aplicando o mesmo atraso do que o caso anterior, as respostas são mostradas na figura 5.8. Pode-se notar mais uma vez que o sistema ainda tem uma resposta aceitável principalmente em se tratando de posição uma vez que o sinal de referência para esse caso é mais lento do que para a velocidade.

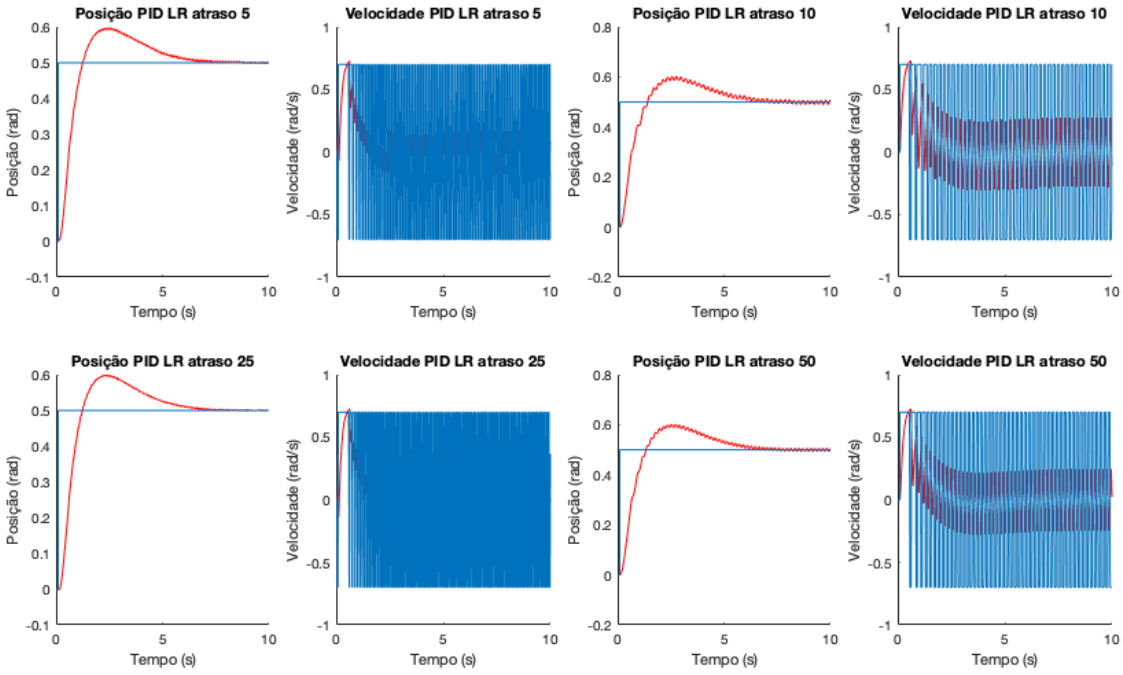

Figura 5.8: Saida da malha de posição e velocidade (laranja) em relação às referências (azul) para sintonia LR com atraso. Fonte: A autora, 2018.

5.1.7 PID por metaheurística ED

A resposta em simulação do controlador PID sintonisado a partir da metaheurística ED se encontra na figura 5.9.

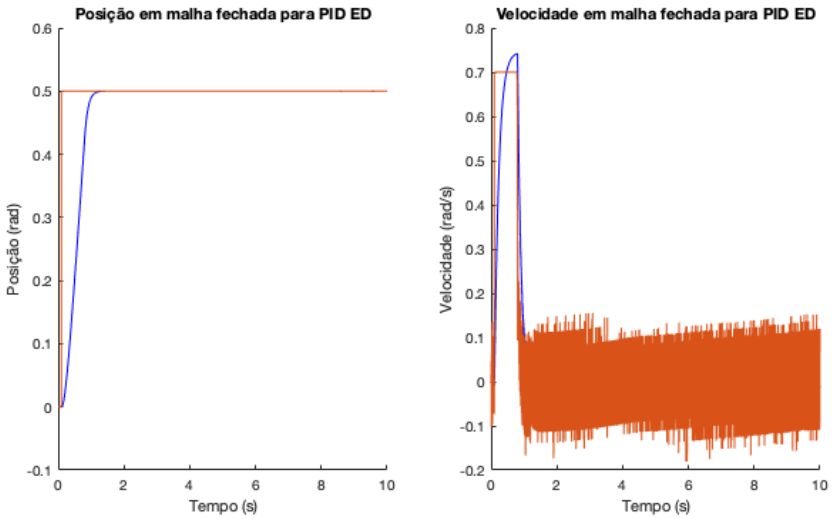

Figura 5.9: Saida da malha de posição e velocidade (azul) em relação às referências (laranja) para sintonia ED. Fonte: A autora, 2018.

Mais uma vez as saturações da simulação fazem com que a resposta em malha fechada torne-se mais lentas. O sinal de controle para esse caso ainda é maior do que para o PID sintonisado por IMC, mas menor do que para o LR.

As respostas com atraso para esse caso são mostradas na figura 5.10. Mais uma vez o sistema ainda tem uma resposta aceitável principalmente em se tratando de posição.

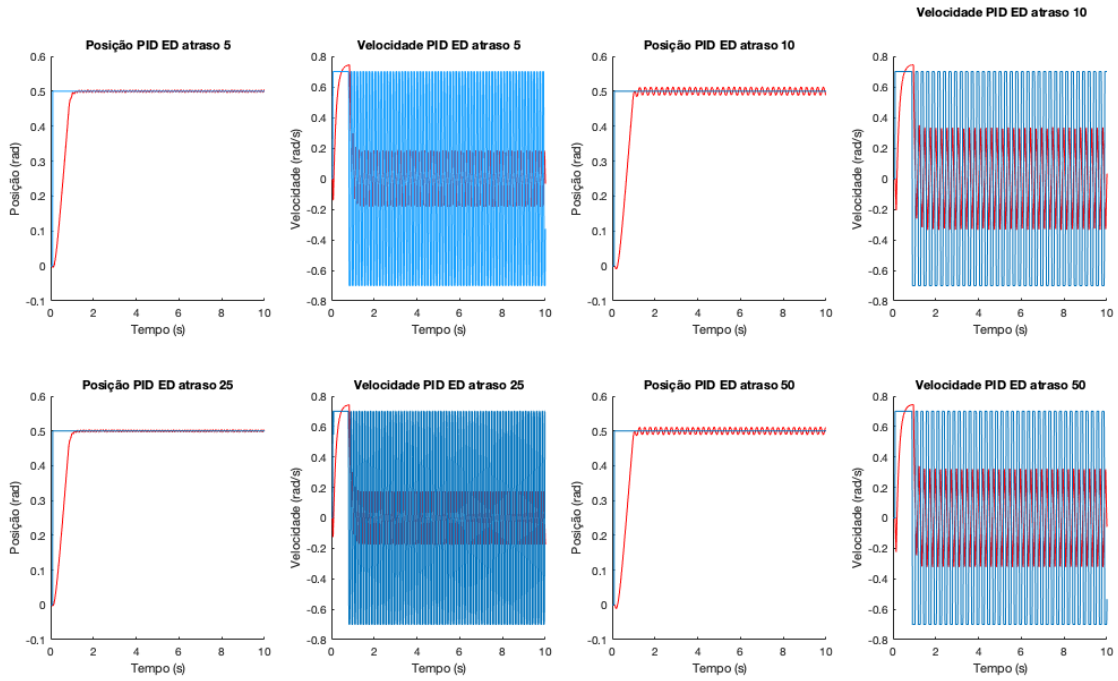

Figura 5.10: Saida da malha de posição e velocidade (laranja) em relação às referências (azul) para sintonia ED com atraso. Fonte: A autora, 2018.

5.2 *Hardware-in-the-loop com NI USB 6009*

Uma vez feitas as simulações, o passo seguinte seria a realização do controle do motor propriamente dito a partir da utilização do *hardware-in-the-loop*.

Como mencionado anteriormente, o que muda da simulação para o sistema real é a substituição do bloco *motor* no Simulink pelo hardware (próprio motor, ponte H e sensores de posição e velocidade). O dispositivo que faz o intermédio entre software e hardware é o *NI USB 6009*, que enviará sinais analógicos e receberá sinais digitais. O esquemático que liga o hardware ao software é mostrado na figura 5.11.

Dessa maneira a malha de controle é formada pelo software (Simulink) e pelo hardware (motor, ponte H e sensores). O Simulink é o responsável por gerar os sinais de controle (velocidade e posição) em que o sinal de controle de velocidade é transformado em sinais de pwm que serão transferidos pelo dispositivo *NI USB 6009* até a ponte H ligada ao motor. A realimentação que fechará a malha então é feita pelos sensores de velocidade e posição.

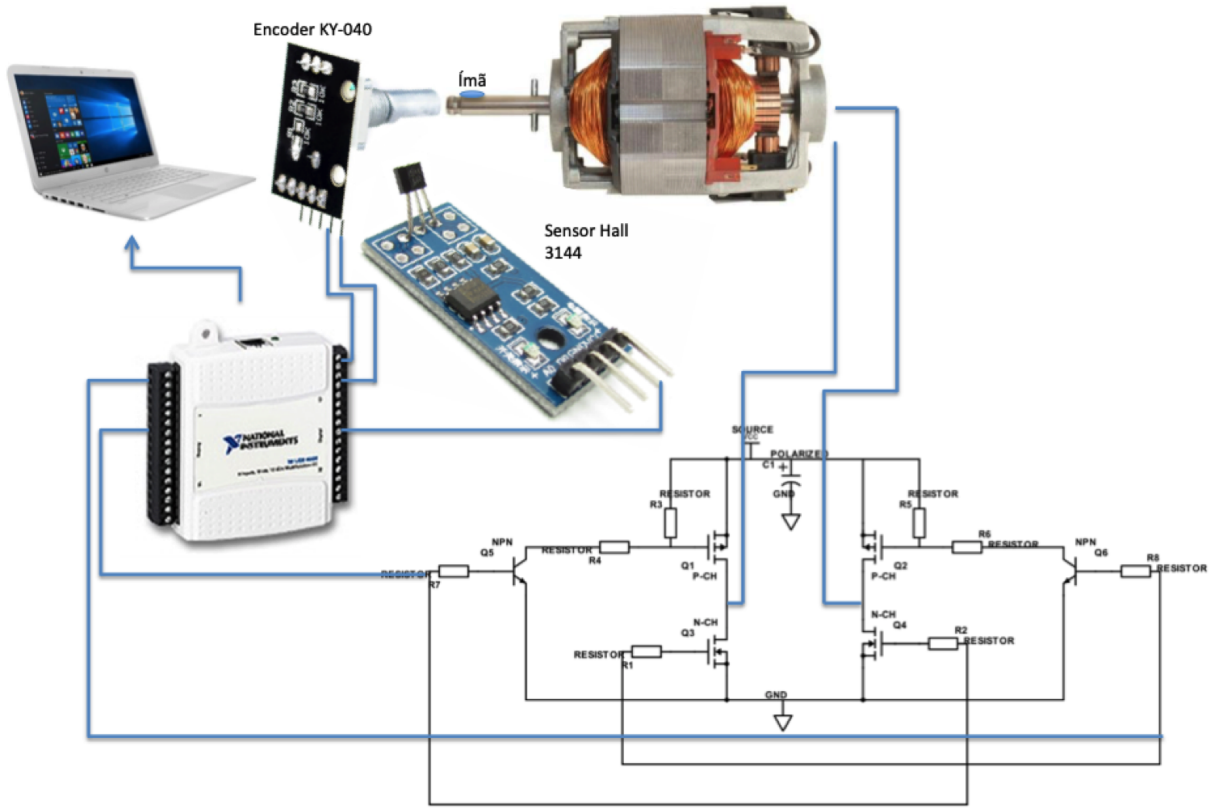

Figura 5.11: Esquemático utilizado para a realização do *hardware-in-the-loop*. Fonte: A autora, 2018.

Como a escolha foi feita de ligar os dois sensores diretamente ao software, os seus sinais devem ser tratados e transformados em sinais de posição angular em *rad* e velocidade angular em *rad/s*.

Para isso um bloco de tratamento desses sinais foi criado no Simulink seguindo a lógica apresentada na fundamentação teórica. O bloco resultante é apresentado na figura 5.12.

No esquemético, o bloco do Simulink responsável por adquirir os sinais do dispositivo *NI USB 6009* é o *Digital Input* disponível em *Data Acquisition Toolbox* do Simulink. Para que o circuito funcione, é necessário instalar no computador que possui o Matlab o suporte *NI-DAQmx* para que o Simulink reconheça o dispositivo.

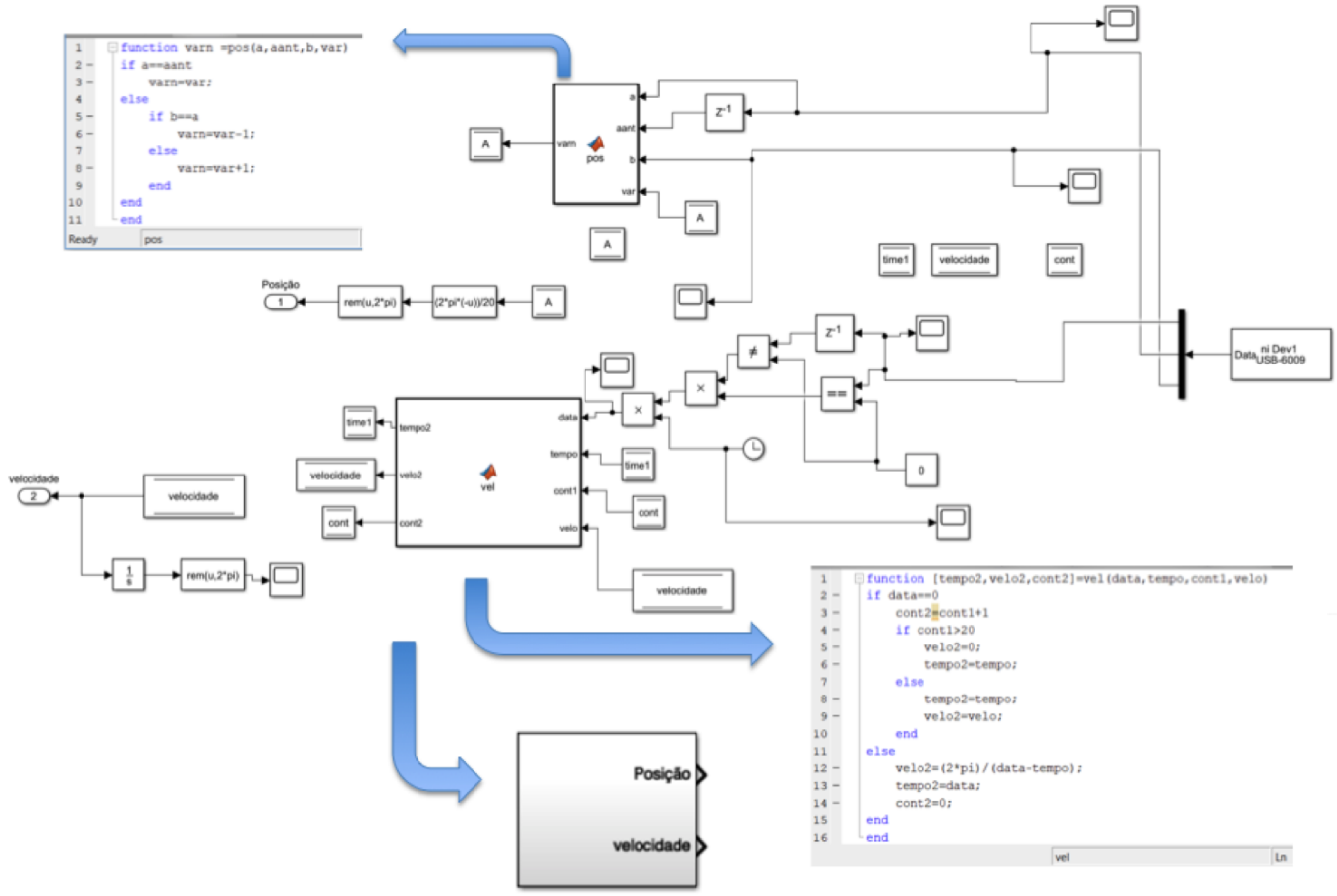

Figura 5.12: Esquemático para a transformação dos sinais fornecidos pelos sensores em velocidade e posição. Fonte: A autora, 2018.

O esquemático completo da parte software do *hardware-in-the-loop* no Simulink se encontra na figura 5.13. Para enviar os sinais de PWM foram usadas as duas saídas analógicas do dispositivo *NI USB 6009* pois as saídas digitais necessitavam de um resistor de *pull-up*. As saídas do sistema foram incorporadas ao bloco de velocidade e posição sendo ligadas ao *NI USB 6009* através do *Digital Output* disponível em *Data Acquisition Toolbox* do Simulink.

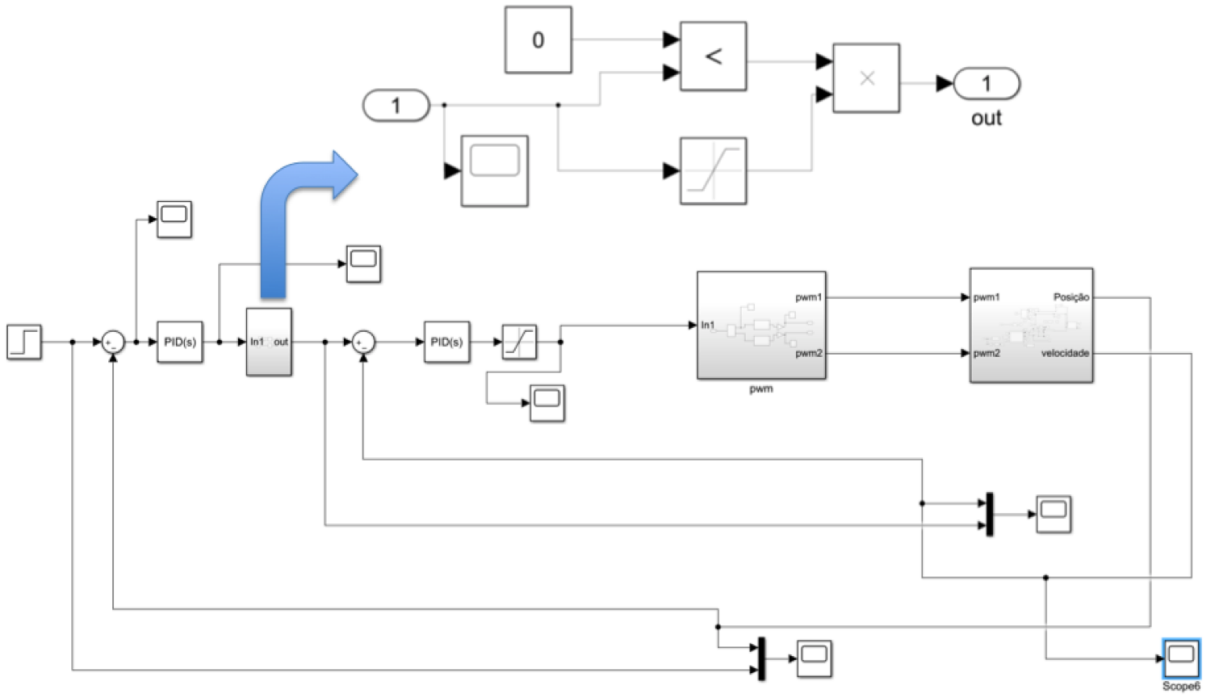

Figura 5.13: Esquemático completo do *hardware-in-the-loop* com saturação. Fonte: A autora, 2018.

Para o caso real do *hardware-in-the-loop* a saturação para a referência de velocidade ainda deve ser mantida e deve incluir além da velocidade máxima permitida com a máxima tensão de alimentação, uma velocidade mínima para a mínima tensão de alimentação do motor necessária para que ele rotacione. O bloco usado para essa saturação se encontra destacado na figura 5.13.

Ao começar a simulação por *hardware-in-the-loop* notou-se que para o motor escolhido (motor universal), a rotação só poderia ser realizada em um sentido. Isso acontece pelo fato de o motor universal aceitar como tensão de entrada tensões CC e CA. Logo, a ponte H continua sendo usada para controlar a tensão no motor mas ela não pode inverter o sentido. Isso significa que um dos sinais de controle será sempre zero.

O tempo de amostragem para a realização do *hardware-in-the-loop* foi escolhido como 5ms.

Dessa forma, apenas os controladores que não apresentam overshoot possuem uma resposta satisfatória.

A resposta em posição e velocidade para as malhas fechadas com controladores sintonizados pelo controle por modelo interno (IMC) se encontram na figura 5.14.

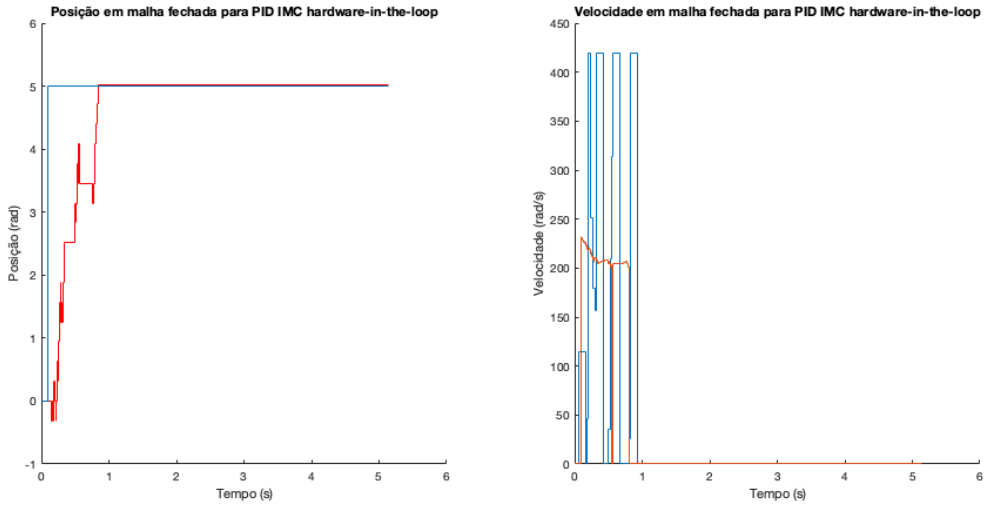

Figura 5.14: Saida da malha de posição e velocidade (laranja) em relação às referências (azul) para sintonia IMC no caso *hardware-in-the-loop*. Fonte: A autora, 2018.

A resposta em posição e velocidade para as malhas fechadas com controladores sintonizados pelo lugar das raízes (LR) se encontram na figura 5.15.

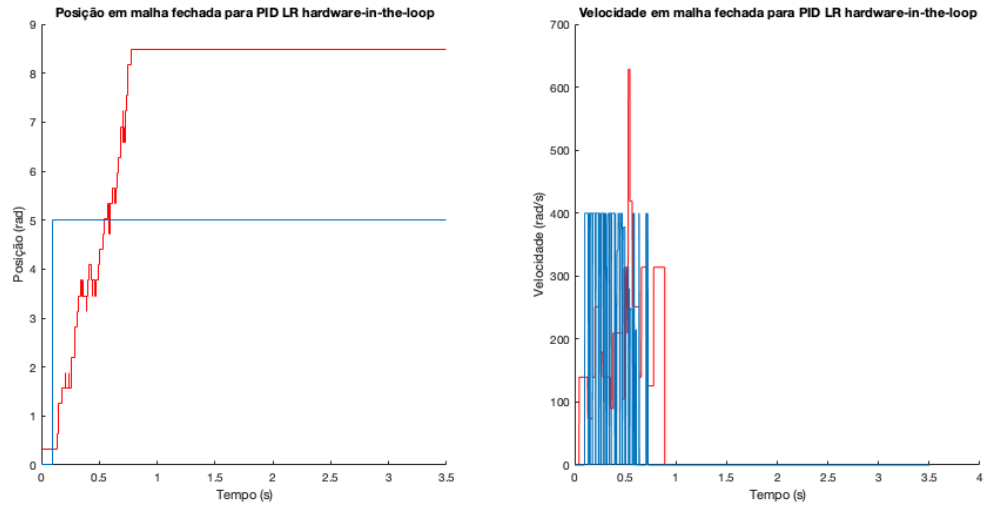

Figura 5.15: Saida da malha de posição e velocidade (laranja) em relação às referências (azul) para sintonia LR no caso *hardware-in-the-loop*. Fonte: A autora, 2018.

A resposta em posição e velocidade para as malhas fechadas com controladores sintonizados pela metaheurística ED se encontram na figura 5.16.

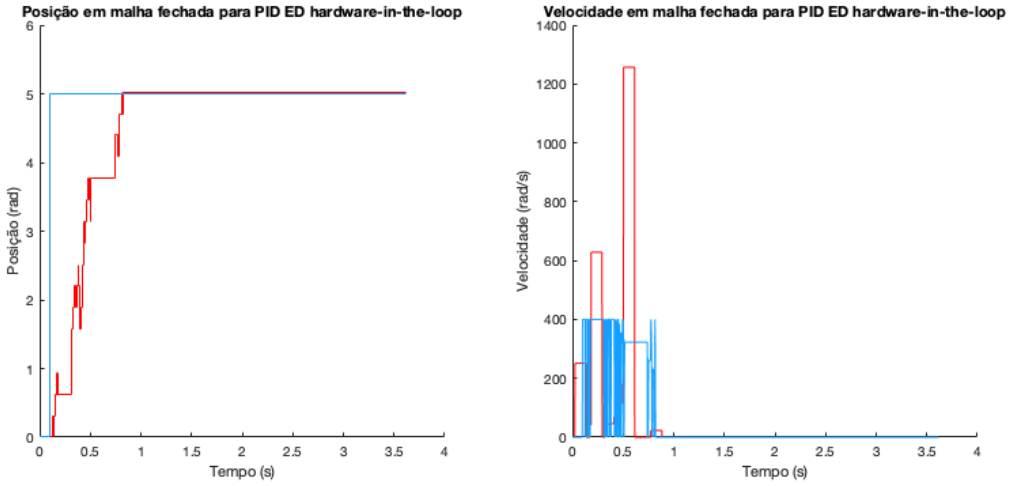

Figura 5.16: Saida da malha de posição e velocidade (laranja) em relação às referências (azul) para sintonia ED no caso *hardware-in-the-loop*. Fonte: A autora, 2018.

Apesar de que as respostas parecem, em primeira vista, condizentes com as obtidas em simulação, o dispositivo NI USB 6009 não permite uma sincronização ideal entre software e hardware. Isso acontece pois o dispositivo possui apenas um ADC o que faz com que, quando existem vários sinais a serem enviados e recebidos, apenas um seja transmitido. Logo, o dispositivo NI USB 6009 não é suportado toolbox *Simulink Desktop Real Time* disponível no Simulink.

Mesmo assim o dispositivo pode continuar sendo utilizado em aplicações em que não se exija um tempo de aquisição importante.

O motor universal apresenta um torque inicial muito alto bem como velocidades mínimas altas o que faz com que a posição mude rapidamente sem que ela seja necessáriamente contabilizada e processada corretamente pelo software. Mesmo que com os resultados obtidos com o NI USB 6009 e o *Data Acquisition Toolbox* do Simulink se possa ver que o controlador funciona corretamente parando o motor uma vez que a referência é atingida ou ultrapassada, não se pode garantir que a posição do motor real condiz com a obtida nem que o tempo esteja correto uma vez que não se pode fazer a simulação em tempo real.

5.3 *Hardware-in-the-loop com STM32F103*

Para fazer uma análise em tempo real a partir da toolbox *Simulink Desktop Real Time* a ideia foi de substituir o dispositivo NI USB 6009 pelo microcontrolador STM32F103 seguido de um chip FTDI que faz a conversão UART - USB possibilitando a comunicação serial.

A toolbox *Simulink Desktop Real Time* suporta comunicação serial com qualquer dispositivo que o faça e é compatível a utilização do *External Mode* que permite uma mais alta performance utilizando o coder do Simulink para fazer um link com as entradas e sídas externas.

Para tal, os blocos de entrada e saída digitais e analógicas da figura 5.12 antes adicionados a partir da *Data Acquisition Toolbox* serão substituidos pelos blocos

Packet Input e *Packet Output* do *Simulink Desktop Real Time*.

O microcontrolador nesse caso serve apenas para fazer a comunicação entre o Simulink e o hardware e apenas vai mandar ou receber do Simulink um sinal do tipo *uint8*. A comunicação serial é composta por um sinal de transmissão e um sinal de recepção. No sinal de transmissão do Simulink para o microcontrolador devem estar contidos os sinais de PWM e o sinal recebido pelo Simulink enviado pelo microcontrolador conterá as informações dos sensores.

Cada sensor será representado por um *bit* na palavra *uint8* recebida pelo Simulink. E cada sinal de PWM também será representado por um *bit* na palavra *uint8* enviada pelo Simulink.

A única adição de blocos no Simulink seriam blocos que reconheçam as palavras recebidas e as convertam nos estados dos sensores e que transformem as informações de PWM em apenas uma palavra. Isso é possível graças ao fato de todos os sinais serem digitais.

O esquemático final do *hardware-in-the-loop* utilizado nesse caso se encontra na figura 5.17.

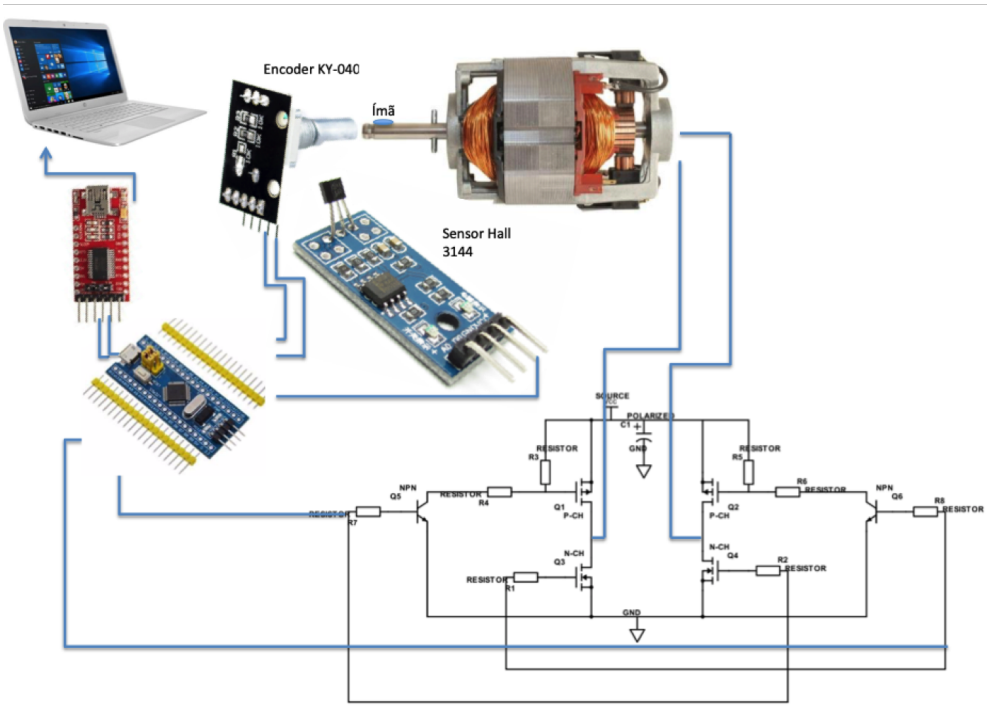

Figura 5.17: Esquemático utilizado para a realização do *hardware-in-the-loop* com STM32F103. Fonte: A autora, 2018.

Com esse novo esquemático a aquisição de dados em tempo real se tornou possível. Porém, mesmo não tendo o problema anterior de envio de penas um sinal, os resultados obtidos continuaram sendo insatisfatórios para gerar conclusões sobre o funcionamento dos controladores em *hardware-in-the-loop* pois o tempo de processamento dos sinais no Simulink é muito elevado para a velocidade de geração dos dados, principalmente os de posição. Isso se torna crucial para o caso do motor universal em não se pode reverter o sentido de rotação para o ajuste de posição.

CAPÍTULO 6

CONCLUSÃO

Considerando os resultados obtidos a partir das sintonias dos controladores PID de velocidade e posição, pode-se concluir que os três métodos escolhidos obtiveram boas performances.

Tanto a sintonia IMC, LR e pela metaheurística ED geraram controladores capazes de obter características em malha fechada muito similares e satisfatórias. Mesmo quando testados em simulação em que se consideraram não linearidades como saturações e atrasos, provaram-se robustos o suficiente.

Dessa forma pode-se concluir que o uso da metahuerística ED para fazer a sintonia de controladores PID mostra-se uma solução a altura das já existentes com a vantagem da simplicidade uma vez que o código será sempre o mesmo independente do sistema.

Além disso, a metaheurística ED mostrou-se uma ótima opção para realizar a identificação de sistemas. Além de apresentar um *fit* melhor do que o método de identificação para sistemas ARMAX, sua implementação é mais simples.

Em contra partida, quanto aos testes do *hardware-in-the-loop* propriamente dito, percebeu-se que o motor universal talvez não seja a melhor escolha de sistema para esse tipo de análise pelo fato de seu alto torque de partida e velocidade não permitirem que o Simulink tome o tempo necessário para fazer o tratamento de sinais e pelo fato de não permitir a rotação em dois sentidos.

Para melhorar esse trabalho seria interessante transferir o processamento de dados hoje feito no próprio Simulink para o hardware. Dessa forma o software Simulink seria apenas responsável por transmitir os sinais de controle gerados pelo PID. Dessa forma o tempo de processamento do Simulink poderia ser reduzido.

Outra sugestão seria refazer os processos para um motor de mais baixo torque e velocidade que permita os dois sentidos de rotação como um motor CC. Isso reduziria o tempo de amostragem necessário e possibilitaria correções no caso de sobre-sinais.

BIBLIOGRAFIA

ANDRADE L. H. S. ; COSTA, B. L. G. . A. B. A. de. Pso aplicado à sintonia do controlador pi/pid da malha de nível de uma planta didática industrial. *XI IEEE/IAS International Conference on Industry Applications*, dezembro 2014.

CERNE. *Módulo sensor de efeiro Hall*. 2018. <https://www.cernecomponenteeletronico.com.br/loja/arduino?it=modulo-sensor-de-efeito-hall-a3144e-a3144. Acesso em 26/11/2018.

COSTA, B. L. G. Sintonia de controladores pid utilizando a metaheurística evolução diferencial. *XVII Seminário de Iniciação Científica e Tecnológica da UTFPR*, novembro 2012.

DAS K.R. ; DAS, D. . D. J. Optimal tuning of pid controller using gwo algorithm for speed control in dc motor. *2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI)*, IEEE, outubro 2015.

FRANCA, Y. M. M. Iterated local search aplicado à sintonia de controladores pid. *Monografia apresentada na Universidade Federal de Ouro Preto Escola de Minas*, agosto 2009.

GOMES, H. R. N. Busca tabu aplicada ao problema de sintonia de controladores pid. *Monografia apresentada na Universidade Federal de Ouro Preto Escola de Minas*, agosto 2009.

HENRY'S-BENCH. *Keyes KY-040 Arduino Rotary Encoder User Manual*. 2018. <http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/keyes-ky-040-arduino-rotary-encoder-user-manual/. Acesso em 26/11/2018.

MATLAB. *Control tutorials for Matlab*. 2018. <http://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeedsection=SystemModeling. Acesso em 26/11/2018.

NATIONAL-INSTRUMENTS. *NI USB 6009*. 2018. <http://www.ni.com/pt-br/support/model.usb-6009.html. Acesso em 26/11/2018.

OLIVEIRA P.B.M. ; PIRES, E. . N. P. Design of posicast pid control systems using a gravitational search algorithm. *Elsevier*, v. 167, novembro 2015.

PELUCHI, P. C. *Motores Elétricos Princípios e Fundamentos*. 2018. <https://www.ebah.com.br/content/ABAAAewp0AI/mototes-eletricos. Acesso em 26/11/2018.

SERAPIAO, A. B. S. Busca harmônica aplicada à sintonia automática de controladores pid. *X SBAI – Simpósio Brasileiro de Automação Inteligente*, setembro 2011.

SHAYANFAR H. A. ; SHAYEGHI, H. . Y. A. Pid type stabilizer design using grey wolfe optimization algorithm. *The Steering Committee of The World Congress in Computer*

Science, Computer Engineering and Applied Computing (WorldComp), 2016.