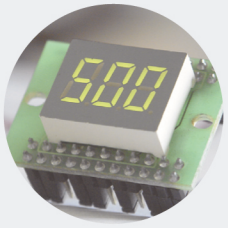
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Modelação e controlo de sistema de levitação magnética utilizando redes neuronais

BRUNO DANIEL ESTEVES DA SILVA
outubro de 2020

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Modelação e controlo de sistema de levitação magnética utilizando redes neuronais

Mestrado em Engenharia Eletrotécnica e de Computadores

Bruno Daniel Esteves da Silva

Orientação
Ramiro Sousa Barbosa

Ano Letivo: 2019-2020

Instituto Superior de Engenharia do Porto
Departamento de Engenharia Eletrotécnica
Rua Dr. António Bernardino de Almeida, 431, 4200-072 Porto

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Eletrotécnica e de Computadores

Candidato: Bruno Silva, Nº 1150610, 1150610@isep.ipp.pt
Orientação científica: Ramiro de Sousa Barbosa, rsb@isep.ipp.pt

Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Automação e Sistemas
Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
21 de Outubro de 2020

Agradecimentos

Inicialmente, quero agradecer ao Engenheiro Ramiro de Sousa Barbosa pelo tempo e recursos disponibilizados durante a orientação desta dissertação ao longo destes últimos meses.

Aos meus pais agradeço toda a ajuda durante a realização desta dissertação, e agradeço também o incentivo e confiança durante todos estes anos da minha formação académica.

Resumo

A levitação magnética consiste na suspensão de um objeto no ar, utilizando forças magnéticas de atração ou repulsão, contrariando assim a força gravítica aplicada nesse objeto. Nos tempos correntes é cada vez mais utilizada em áreas de transporte de passageiros e carga pois reduz drasticamente as perdas mecânicas que os sistemas convencionais acarretam. No entanto, sistemas que empregam levitação magnética necessitam de um algoritmo de controlo pois são, normalmente, instáveis e não lineares.

A utilização de algoritmos de controlo encontra-se presente na maior parte dos sistemas, existindo diversas formas de controlo. Atualmente ainda os controladores clássicos, como o controlo PID, são os mais utilizados, mas graças à evolução científica e tecnológica na área da inteligência artificial foram desenvolvidas técnicas de controlo inteligente, como o controlo neuronal. As redes neuronais são agora muito estudadas e utilizadas em áreas de processamento de imagem, problemas de otimização complexos, controlo de sistemas, entre outros.

Sendo assim, nesta dissertação são desenvolvidos, para controlar um sistema de levitação magnética, controladores clássicos e controladores neuronais de modelo inverso, modelo interno e modelo de referência e são comparados os seus desempenhos. É assim averiguada a capacidade dos controladores neuronais fazerem o sistema seguir um sinal de referência de maior amplitude, ao invés dos controladores clássicos, devido à não linearidade do sistema.

Por último, os controladores clássicos e neuronais desenvolvidos são implementados num micro-controlador, é desenvolvido um algoritmo na linguagem C que permite monitorizar o desempenho do sistema e é realizada uma comparação dos diferentes controladores testados.

Palavras-Chave: Levitação Magnética, Sistema não linear, Controlador em avanço, Controlador em avanço-atraso, Identificação Neuronal, Controlador neuronal.

Abstract

Magnetic levitation consists of the suspension of an object in the air, using magnetic forces of attraction or repulsion, thus counteracting the gravitational force applied to that object. In current times it is increasingly used in areas of passenger and cargo transportation because it drastically reduces the mechanical losses that conventional systems entail. However, systems that employ magnetic levitation require a control algorithm as they are normally unstable and nonlinear.

The use of control algorithms is present in most systems, with several forms of control. Currently, classical controllers, such as PID control, are still the most used, but thanks to scientific and technological developments in the field of artificial intelligence, intelligent control techniques have been developed, such as neural control. Neural networks are now widely studied and used in the areas of image processing, complex optimization problems, systems control, among others.

Thus, in this dissertation, classical controllers and inverse model, internal model and reference model neural controllers are developed to control a magnetic levitation system and their performances are compared. Thus, the ability of neural controllers to make the system follow a reference signal of greater amplitude is verified, unlike the classic controllers, due to the nonlinearity of the system.

Finally, the classical and neural controllers developed are implemented in a micro-controller, an algorithm is developed in C language that allows monitoring the system performance and a comparison of the different tested controllers is carried out.

Keywords: Magnetic levitation, Nonlinear system, Lead compensator, Lead-lag compensator, Neural identification, Neural controller.

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

Abreviatura	Descrição	Página
ADC	*Analog-to-Digital Converter*	49
ADU	*Analog-to-Digital Units*	56
bps	bits por segundo	98
CC	Corrente Contínua	8
CERN	*Conseil Européen pour la Recherche Nucléaire*	16
GPU	*Graphics Processing Unit*	26
IDE	*Integrated Development Environment*	105
ISEP	Instituto Superior de Engenharia do Porto	1
LED	*Light-Emitting Diode*	50
LHC	*Large Hadron Collider*	16
MEEC	Mestrado em Engenharia Eletrotécnica e Computadores	1
ML	*Machine Learning*	25
MOSFET	*Metal-oxide-semiconductor field-effect transistor*	47
MSE	*Mean Squared Error*	32
NARX	*Nonlinear AutoRegressive eXogenous*	36
PCB	*Printed Circuit Board*	46
PID	Proporcional-Integral-Derivativo	1
PWM	*Pulse-Width Modulation*	43
ReLU	*Rectified Linear Unit*	30
SRAM	*Static random-access memory*	48
TEDI	Tese/Dissertação	1
USART	*Universal Synchronous and Asynchronous Serial Receiver and Transmitter*	50
USB	*Universal Serial Bus*	44

Capítulo 1

Introdução

Neste capítulo é realizada a contextualização do projeto desenvolvido explicando o seu enquadramento com o curso em que está inserido e com as tecnologias que serão analisadas e utilizadas. De seguida são enumerados os objetivos a cumprir, com o fim de desenvolver o projeto. É também realizada a calendarização do plano de trabalho. Por fim é efetuada uma breve descrição de cada capítulo para uma melhor compreensão do relatório.

1.1 Contextualização

Este projeto surgiu no âmbito da disciplina de Tese/Dissertação (TEDI) do Mestrado em Engenharia Eletrotécnica e Computadores (MEEC) - ramo de Automação e Sistemas, do Instituto Superior de Engenharia do Porto (ISEP). A unidade curricular tem como objetivo integrar e aprofundar conhecimentos das várias áreas científicas adquiridos ao longo do curso.

O tema escolhido adveio do interesse em aplicar conceitos de modelação e controlo de sistemas num sistema não linear instável, utilizando uma técnica de controlo inteligente. Daí serem utilizadas redes neuronais para modelar e controlar uma sistema de levitação magnética.

Desde muito cedo que o controlo possui um papel fundamental na sociedade, estando atualmente presente na maior parte dos sistemas, existindo diversas técnicas de controlo. Dos algoritmos mais utilizados destaca-se o controlador Proporcional-Integral-Derivativo (PID) que faz parte do controlo clássico. No entanto, com a evolução científica e tecnológica no ramo da inteligência artificial existem agora técnicas de controlo intituladas de controlo inteligente. Aí se enquadram técnicas como controlo difuso, controlo fracionário e controlo neuronal.

Ao aplicar técnicas de inteligência artificial num sistema de controlo, obtém-se algoritmos capazes de resolver problemas que o controlo clássico não é capaz. Este tipo de controlo pode ser aplicado a diversos tipos de sistemas, no entanto, destaca-se o seu desempenho face a outras técnicas no controlo de sistemas não lineares e instáveis, dentro dos quais o tema deste trabalho: um sistema de levitação magnética.

Estes sistemas de levitação são cada vez mais utilizados em áreas de transporte de passageiros e carga pois reduzem drasticamente as perdas mecânicas que os sistemas convencionais acarretam. No entanto devido à dificuldade em estabilizar a suspensão de um objeto quer por repulsão quer por atracão magnética, onde a força gravítica é contrariada apenas por forças magnéticas, estes sistemas necessitam de um controlador robusto.

Assim esta dissertação pretende demonstrar todo o processo de implementação de um controlador neuronal aplicado a um sistema de levitação magnética, assim como alguns controladores clássicos de modo a comparar os resultados. É portanto constituída por várias fases, desde o estudo inicial de conceitos necessários à sua execução, até à construção e validação do protótipo.

1.2 Objectivos

O principal objetivo deste trabalho consiste na implementação de um sistema de levitação magnética, em que a posição do objeto suspenso é controlada com recurso a redes neuronais. Assim sendo, são definidos os seguintes objetivos adjacentes:

- Levantamento do estado da arte, por forma a contextualizar o tema estudado;
- Analisar o sistema de levitação magnética, de modo a obter o seu modelo matemático;
- Construção e treino de modelos de redes neuronais para o controlo do sistema;
- Testar diferentes controladores, analisando as suas diferenças.

1.3 Calendarização

O gráfico de Gant ilustrado na Figura 1.1 sumariza o plano de tarefas para este trabalho.

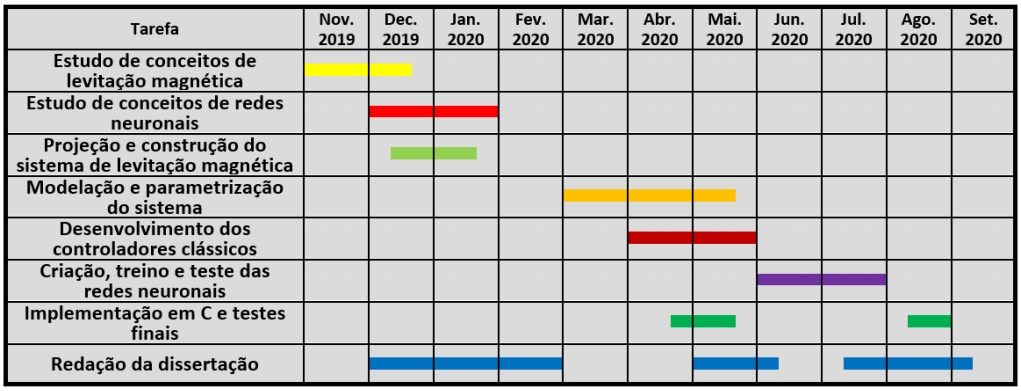


Figura 1.1: Gráfico de Gant da planificação do projeto

1.4 Estruturação do relatório

No Capítulo 1 é realizada uma contextualização do projeto desenvolvido onde é explicado o enquadramento com o curso e introdução aos temas abordados.

Ao longo do Capítulo 2 é feito um levantamento do estado da arte em termos de eletromagnetismo e levitação magnética. Neste capítulo explora-se os diferentes fenómenos eletromagnéticos que possibilitam a levitação magnética bem como as distintas fontes de campos magnéticos que podem ser utilizadas. Por fim detalha-se diferentes técnicas de levitação e suspensão magnética com destaque na suspensão magnética com electroíman de corrente contínua controlados.

De seguida no Capítulo 3, são introduzidas as redes neuronais e estudado o seu enquadramento em sistema de controlo. Inicialmente são expostos os pontos fulcrais que levaram à evolução das redes neuronais, desde a inspiração até às áreas onde são mais utilizadas nos tempos correntes. Seguidamente, são analisadas as suas características, arquiteturas, e algoritmos de treino. Por último, são estudadas as fases presentes no desenvolvimento de controladores neuronais, bem como algumas estruturas de controlo.

No decorrer do Capítulo 4 é descrita a arquitetura do sistema desenvolvido, descrevendo os diversos materiais utilizados e explicando a sua interligação. Aqui são expostas as características do sistema a desenvolver.

Durante o Capítulo 5 é estudado o sistema eletromagnético utilizado, de modo a realizar a sua modelação. De seguida são testados os sensores utilizados de forma a parametrizar o seu funcionamento e assim ser possível desenvolver alguns controladores clássicos que permitam obter dados de treino para as redes neuronais a desenvolver.

No Capítulo 6 são criadas e treinadas todas a redes neuronais deste relatório. Inicialmente são obtidos os dados de treino, utilizando os controladores do

capítulo anterior, para primeiro realizar a identificação neuronal. De seguida são criados alguns controladores neuronais e comparados o seu desempenho.

No seguinte Capítulo 7 é exposto e descrito o processo de programação de alguns dos controladores desenvolvidos num micro-controlador, utilizando a linguagem C. São explicadas as configurações realizadas, a implementação dos controladores e demonstrados os resultados obtidos com essa implementação.

Por último no Capítulo 8 são reunidas as principais conclusões e perspetivados futuros desenvolvimentos. Ao finalizar o documento, são expostas as referências bibliográficas e os anexos que dão suporte ao mesmo.

Capítulo 2

Conceitos eletromagnéticos

No presente capítulo é realizada um breve introdução aos sistemas de levitação magnética. Começando assim por descrever alguns fenómenos magnéticos e eletromagnéticos, sendo estes a base dos sistemas em estudo. De seguida é descrito o funcionamento de diversos métodos de levitação magnética e alguns dos seus usos mais preponderantes nos tempos correntes.

2.1 Eletromagnetismo

A força eletromagnética entre partículas é considerada uma das forças fundamentais da natureza, que, como o nome indica é causada pela interação de campos elétricos e magnéticos [1].

2.1.1 Campo elétrico

Toda a matéria é constituída por átomos, esses constituídos por um certo número de protões, de neutrões e de eletrões. Os protões são elétricamente carregados positivamente sendo que cada átomo mantém um número igual de eletrões (carregados negativamente) para se manter sem carga. Assim a carga elétrica (q) consegue ser quantificada, sendo que materiais podem encontrar-se carregados com diferentes intensidades dependendo da quantidade de eletrões presentes em relação aos protões desse material [1] [2].

Tal como Charles-Augustin de Coulomb comprovou em 1785, cargas carregadas elétricamente conseguem atrair ou repelir outras cargas carregadas, em que cargas de sinal oposto atraem-se, e cargas com o mesmo sinal repelem-se. Isto deve-se à existência de um campo elétrico em seu redor, a sua magnitude é proporcional à da carga que o gera. Com linhas de campo é possível representar

a sua magnitude e direção, onde na Figura 2.1 encontram-se representadas as linhas de campo de cargas elétricas pontuais no espaço [1] [2].

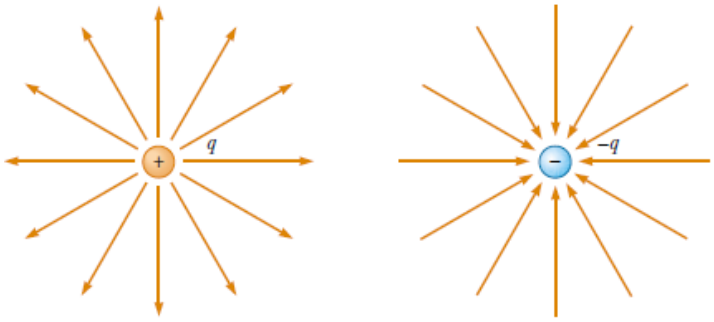


Figura 2.1: Campo elétrico criado por cargas carregadas elétricamente [1]

As suas linhas de campo elétrico começam nas partículas positivas e terminam nas negativas ou infinito caso sejam pontuais. A intensidade da força exercida entre cargas é proporcional às suas magnitudes e inversamente proporcional à distância entre cargas [1] [2].

2.1.2 Campo magnético

Embora cargas elétricas e pólos magnéticos sejam semelhantes em vários aspetos, a maior diferença é que os pólos magnéticos encontram-se sempre em pares, sendo impossível isolar um pólo magnético da mesma maneira que se isola um eletrão, onde caso se dividisse o íman em dois obter-se-iam dois novos ímanes com um pólo norte e sul cada. Tal se pode ver na Figura 2.2, as linhas de campo são também distintas, pois são linhas sempre fechadas, as quais começam no pólo norte e terminam no pólo sul, no exterior do íman, e no seu interior, começam no pólo sul e terminam no pólo norte [1] [2].

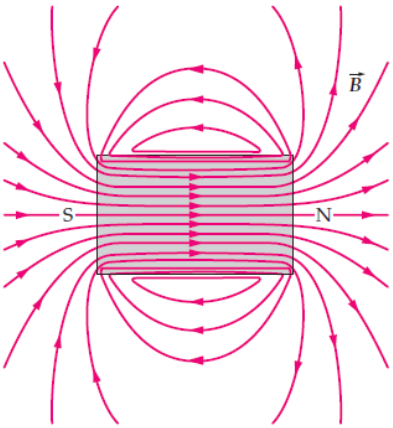


Figura 2.2: Campo magnético produzido por um íman [2]

2.1.3 Indução eletromagnética

Os casos anteriores tinham como foco cargas elétricas ou pólos magnéticos parados no tempo. No entanto, se uma partícula carregada elétricamente se encontrar em movimento, o qual origina um campo elétrico variável no tempo deixamos de ter apenas um campo elétrico, sendo gerado também um campo magnético na região do espaço. O campo gerado é então descrito pela equação (2.1) [1] [2].

$$\overrightarrow{B} = \frac{\mu_0}{4\pi} \frac{q\overrightarrow{v} \times \hat{r}}{r^2} \tag{2.1}$$

onde μ_0 é a constante da permeabilidade magnética do vazio e $\hat{r}$ é o vetor unitário paralelo ao vetor $\overrightarrow{r}$, como representado na Figura 2.3. A carga positiva q ao deslocar-se gera um campo magnético $\overrightarrow{B}$, num determinado ponto P, a uma distância $\overrightarrow{r}$ da carga. O que neste caso corresponderá a um campo magnético a entrar no plano formado entre a carga e o ponto.

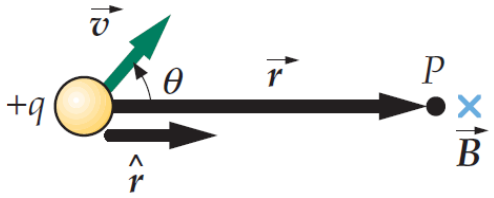


Figura 2.3: Campo magnético gerado por uma carga elétrica em movimento [2]

Este campo pode repelir ou atrair pólos magnéticos, quer sejam eles gerados por cargas elétricas ou por materiais magnéticos. Da mesma maneira que uma partícula elétrica cria uma campo magnético ao deslocar-se, também um condutor elétrico gera uma campo magnético quando percorrido por uma corrente elétrica, podendo ser afetado por outros campos magnéticos [1].

Assim a força aplicada numa partícula elétrica por um campo magnético é dada pela equação (2.2) [1] [2].

$$\overrightarrow{F} = q\overrightarrow{v} \times \overrightarrow{B} \tag{2.2}$$

onde $\overrightarrow{v}$ representa a velocidade a que a partícula se desloca no campo magnético $\overrightarrow{B}$. Sendo que o campo magnético consegue alterar a direção da velocidade da partícula que o atravessa, mas não alterar a energia cinética da partícula [1] [2].

De notar que enquanto a força elétrica atua na mesma direção do campo elétrico, a força magnética atua perpendicularmente ao plano formado entre o vetor velocidade e o vetor campo magnético, como visível na Figura 2.4 [1] [2].

Outra maneira que o campo magnético se relaciona com o campo elétrico, como Michael Faraday e Joseph Henry descobriram, é que ao variar um fluxo

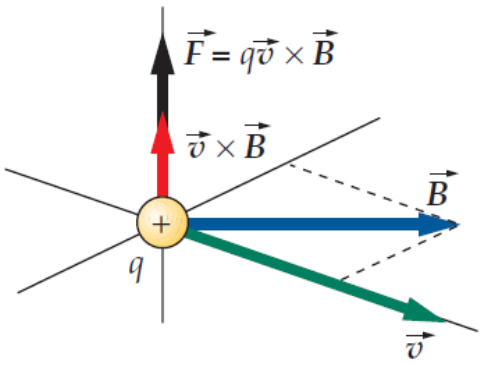


Figura 2.4: Força aplicada por campo magnético [2]

magnético que atravessa um condutor estacionário em *loop* fechado, é gerada uma corrente elétrica no condutor [1] [2].

A interação entre os campos elétricos e magnéticos pode dar-se de várias maneiras, inclusive a capacidade de um campo gerar o outro campo ao variar. Sendo assim torna-se relevante estudar fenómenos como o efeito de Hall, a lei de Ampère e a lei de Biot-Savart aplicáveis à levitação magnética e ao circuito em estudo nos próximos capítulos.

Efeito Hall

Este é um fenómeno muito utilizado em sensores de campo magnético, identificado inicialmente por Edwin Hall em 1879. Como verificado, um condutor é afetado ao ser atravessado por um campo magnético enquanto conduz uma corrente elétrica, pois é criada uma força perpendicular ao seu movimento. Essa força faz com que as partículas que atravessam o condutor sejam "empurradas" causando uma separação da carga, criando uma diferença de potencial no condutor. Diferença essa que pode ser utilizada para medir a magnitude do campo magnético [1] [2].

Ao ter então um condutor elétrico, como representado na Figura 2.5, de largura d e espessura t, percorrido por uma corrente contínua (CC) I enquanto atravessado por um campo magnético B, dá origem a que as partículas elétricas sejam separadas criando uma diferença de potencial ΔV_h [1] [2].

A tensão de Hall pode ser calculada utilizando a equação (2.3).

$$\Delta V = \frac{R_h I}{t} B \tag{2.3}$$

onde R_h é o coeficiente de Hall. Como é averiguável, é possível obter a densidade de fluxo através da tensão ou vice-versa conhecendo os restantes parâmetros como a corrente e a espessura do condutor [1][2]. No caso dos diversos sensores, esses

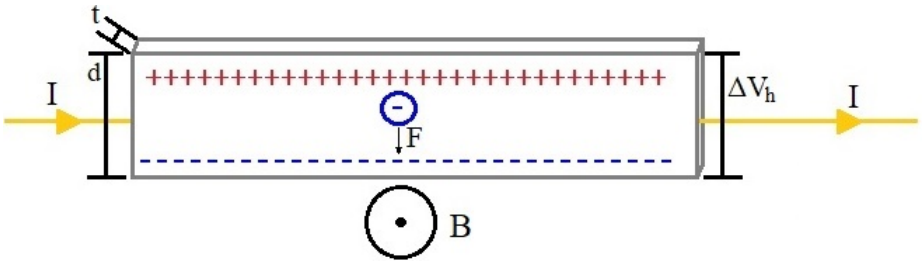


Figura 2.5: Efeito de Hall num condutor

parâmetros são conhecidos, sendo a sensibilidade normalmente dada em milésimas de Volt por Gauss.

Lei de Ampère

Como mencionado anteriormente, uma carga elétrica em movimento origina um campo magnético em seu redor, assim quando um condutor é percorrido por uma corrente elétrica I é também originado um campo magnético à sua volta. A lei de Ampère, nomeada após o físico e matemático francês Andre-Marie Ampère, descreve o campo magnético criado por uma CC em volta de um fio condutor longo e reto. Dada pela equação (2.4), esta lei relaciona o integral do campo magnético em torno da circunferência C numa superfície S atravessada por um fio com a corrente I_C que percorre esse mesmo fio, como representado na Figura 2.6 [1] [2].

$$\oint_C \vec{B}\,\vec{dl} = \mu_0 I_C \tag{2.4}$$

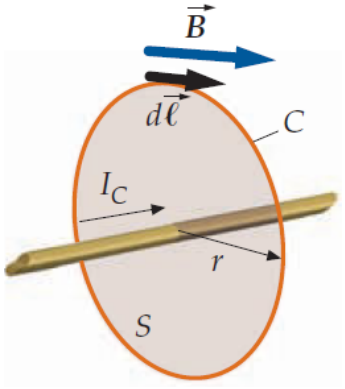


Figura 2.6: Campo magnético gerado segundo a lei de Ampère [2]

A equação (2.4), pode ser simplificada, pois como o campo magnético é constante, e o integral de $\vec{dl}$ é a circunferência C, o campo magnético gerado pela corrente na circunferência de raio r pode ser calculado com a equação 2.5.

$$B = \frac{\mu_0 I_C}{2\pi r} \tag{2.5}$$

No entanto esta lei só permite calcular o campo magnético produzido se existir simetria, pois se caso a circunferência C na Figura 2.6 tivesse como centro não o fio condutor mas sim qualquer outro ponto, o campo magnético não podia ser calculado com a lei de Ampère.

Lei de Biot-Savart

Em 1819, Jean-Baptiste Biot e Félix Savart estudaram este fenómeno e conseguiram chegar a uma expressão matemática que descreve o campo magnético em função da corrente que o produz a qualquer ponto do condutor. Essa expressão é dada pela equação (2.6) [1] [2] [3].

$$d\vec{B} = \frac{\mu_0}{4\pi} \frac{I d\vec{\ell} \times \hat{r}}{r^2} \tag{2.6}$$

onde $d\vec{B}$ é o campo magnético num ponto P_1 associado a um segmento condutor de comprimento $d\vec{\ell}$ percorrido por uma corrente contínua (CC) constante I. Analisando a equação (2.6) e a Figura 2.7, verifica-se que o campo magnético ($d\vec{B}$) é perpendicular ao plano formado pelo fio condutor percorrido pela corrente e pelo ponto P_1, onde se pretende calcular o campo magnético [1] [2] [3].

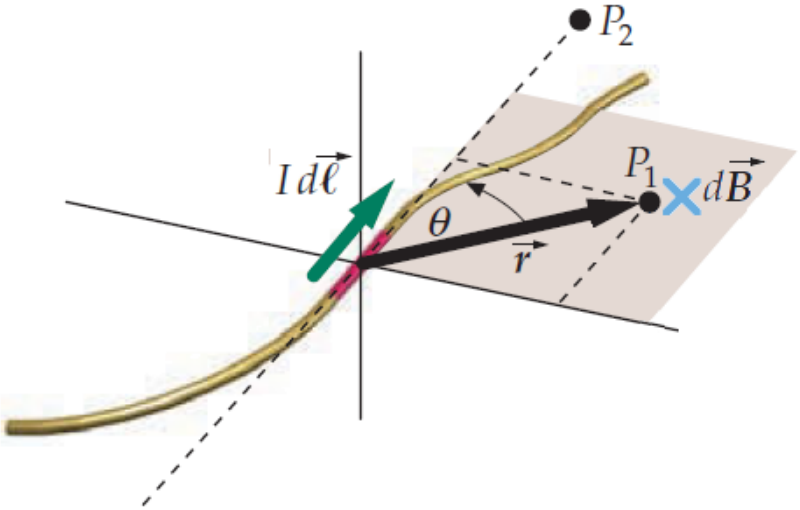


Figura 2.7: Lei de Biot-Savart [2]

Este fenómeno verifica-se independentemente da forma do condutor. Ao ter um condutor elétrico em *loop* de raio R percorrido por uma corrente I, como representado na Figura 2.8, o campo magnético gerado no centro do *loop* tem a direção do eixo do condutor.

A magnitude desse campo magnético pode ser calculado através da equação (2.7) [2]:

$$dB = \frac{\mu_0}{4\pi} \frac{I d\ell \times \sin\theta}{R^2} \tag{2.7}$$

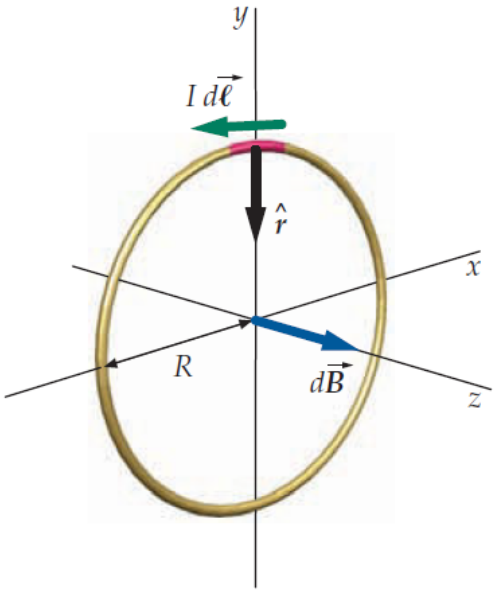


Figura 2.8: Lei de Biot-Savart num *loop* condutor [2]

onde θ é o ângulo entre $d\vec{\ell}$ e $\hat{r}$ o que neste caso são 90º, ou seja $\sin\theta = 1$. No entanto, como estudado anteriormente, o campo magnético pode ser representado por linhas de campo, não existindo apenas no centro do *loop* condutor, assim uma representação mais próxima é dada pela Figura 2.9 [1] [2].

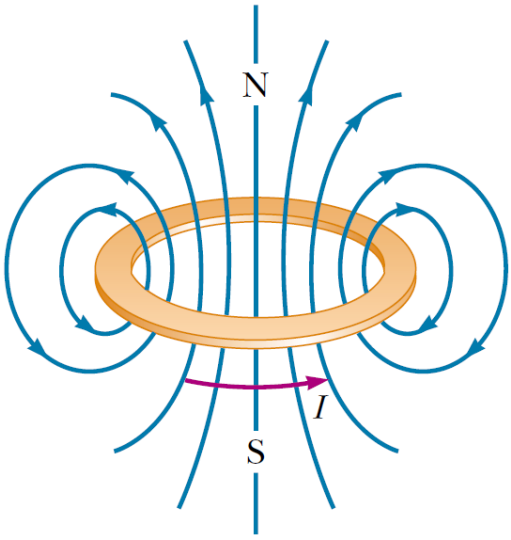


Figura 2.9: Campo gerado num *loop* condutor [1]

Ao colocar-se um fio condutor em espiral, com as espiras próximas entre si é obtido um solenóide, como representado na Figura 2.10. Ao ser percorrido por uma corrente é gerado um campo magnético cujas linhas de campo no interior do

enrolamento encontram-se mais juntas e quase paralelas, indicando que o campo magnético é intenso e uniforme [1] [2].

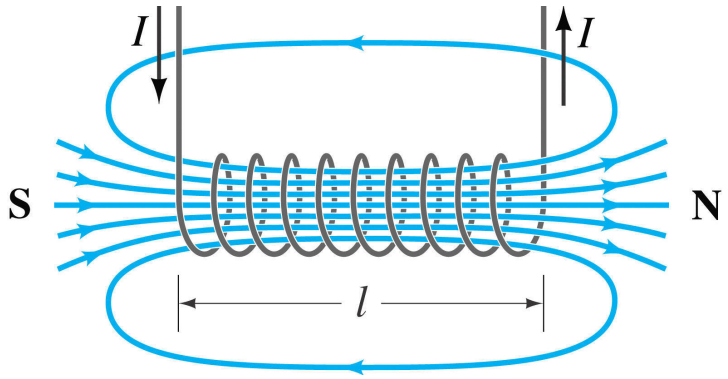


Figura 2.10: Solenóide [2]

Um solenóide é então normalmente caracterizado pelo seu comprimento ser substancialmente maior que o seu diâmetro, e por ter um determinado número de espiras (N) ao longo do seu comprimento. A disposição das espiras influencia o campo gerado, pois ao ter espiras mais próximas entre si, as linhas de campo entre elas tendem a cancelar-se porque os vetores de campo dos dois elementos estão em direções opostas. Isto leva a um campo magnético mais uniforme (o campo gerado por cada porção (espira) aproxima-se mais à de um *loop*) [1] [2].

Assim ao ter um solenóide de comprimento ℓ com N espiras de raio R, é possível calcular o campo magnético num ponto P que se encontre no eixo do solenóide. Para isso é utilizada a equação (2.8), onde P_1 e P_2 correspondem aos extremos do solenóide e são simétricos, pois o ponto médio do solenóide corresponde a 0 no eixo [1] [2].

$$B_P = \frac{1}{2}\mu_0 nI\left(\frac{P-P_1}{\sqrt{(P-P_1)^2+R^2}} - \frac{P-P_2}{\sqrt{(P-P_2)^2+R^2}}\right) \tag{2.8}$$

onde n corresponde a N/ℓ.

Ao explorar a equação (2.8), se P corresponder à posição central (igual a 0), ambas as frações dentro dos parênteses tendem para 1. Como a parcela da função dentro dos parênteses tenderá para 2, a equação (2.9) pode descrever o campo magnético na maior parte de um solenóide suficientemente comprido [2].

$$B_P = \mu_0 nI \tag{2.9}$$

Analisando a Figura 2.11 é possível ter uma melhor ideia do campo gerado ao longo do solenóide [2].

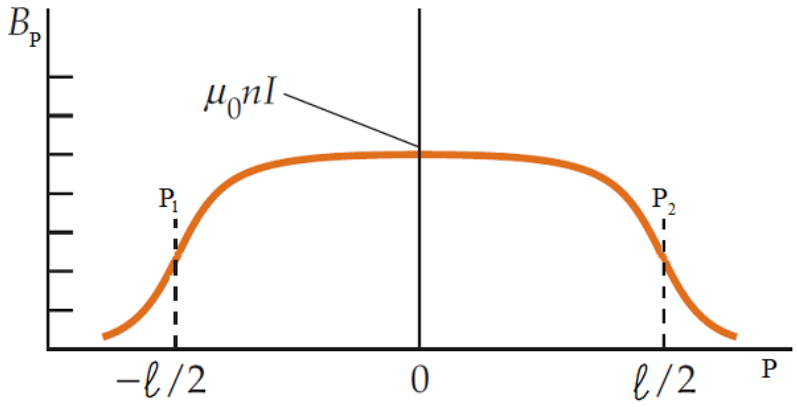


Figura 2.11: Campo magnético no interior de um solenóide [2]

2.2 Fontes de campo magnético

Existem diversas fontes e maneiras de gerar um campo magnético, como já discutido neste capítulo, trata-se no entanto necessário analisar as diferenças entre as múltiplas fontes de forma a compreender melhor as escolhas realizadas nos circuitos de levitação magnética.

Começando por analisar a componente magnética na matéria. Existem inúmeros materiais diferentes, estes tem reações distintas quando se encontram na presença de um campo magnético externo. Com o intuito de os comparar e distinguir é utilizada a equação 2.10 para calcular a permeabilidade magnética relativa de cada material utilizando como referência a permeabilidade magnética do vazio μ_0 [1] [2].

$$\mu_r = \frac{\mu}{\mu_0} \tag{2.10}$$

onde μ corresponde à permeabilidade de cada material, que indica a maior ou menor facilidade com que um material se deixa atravessar pelas linhas de força de um campo magnético B.

Assim quando um campo magnético externo é aplicado a diferentes materiais, vários tipos de comportamentos são observados, representando os vários tipos de materiais ferromagnéticos [1] [2] [3] [4] [5]:

- **Ferromagnéticos** - Estes materiais conservam grande parte da magnetização após o campo externo ser retirado. Alguns destes materiais possuem um momento magnético permanente na ausência de um campo externo e manifestam altas e permanentes magnetizações. São materiais com uma permeabilidade magnética relativa muito maior que 1 como por exemplo o ferro com uma permeabilidade magnética de 5000;

- **Paramagnéticos** - São caracterizados por terem um pequena permeabilidade magnética positiva pois os seus dipólos magnéticos alinham-se com o campo externo. Ao contrário dos materiais ferromagnéticos, os materiais paramagnéticos voltam ao estado original após o campo externo ser retirado, sendo que a sua permeabilidade magnética relativa também é ligeiramente maior que 1 como é o caso do alumínio;

- **Diamagnéticos** - Neste caso, os dipólos opõe-se ao campo magnético, o que origina um efeito magnético negativo de fraca permeabilidade magnética negativa. Assim, estes materiais tem uma permeabilidade magnética relativa ligeiramente menor que 1, como por exemplo o cobre.

Utilizando os conceitos estudados anteriormente de eletromagnetismo e de tipos de materiais, explora-se de seguida três tipos de tecnologias capazes de produzirem um campo magnético e de serem utilizadas em sistemas de levitação magnética.

2.2.1 Íman Permanente

Maior parte da população tem um conhecimento generalizado do que é um íman permanente, devido à sua elevada utilização em diversas aplicações mundanas, como dispositivos eletro-acústicos, instrumentos de medida, equipamentos médicos, componentes de microondas, motores elétricos, e inclusive brinquedos [4].

Um íman permanente ou íman natural é então caracterizado por criar o seu próprio campo magnético persistente através de propriedades magnéticas inerentes a si mesmo, como é o caso da magnetite que é um material já magnetizado na natureza. No entanto também é possível utilizar materiais ferromagnéticos que conseguem manter o campo magnético durante um extenso período de tempo após serem estimulados por um campo magnético externo, como representado na Figura 2.12 [3] [4] [5].

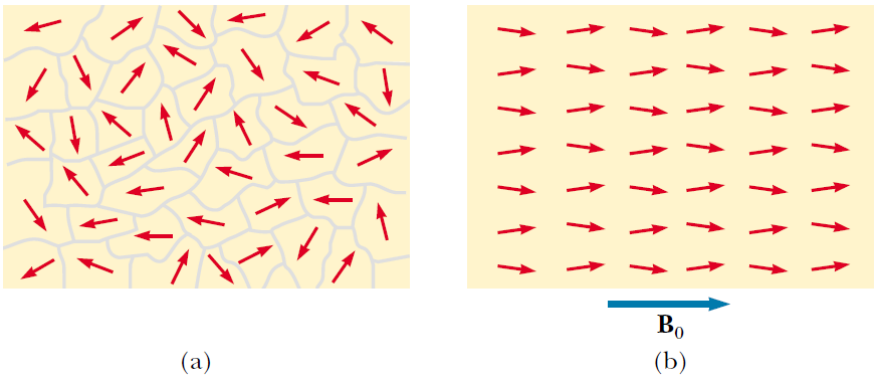


Figura 2.12: Momentos magnéticos: desalinhados (a) alinhados com B_0 (b) [1]

Inicialmente em (a) os momentos magnéticos atómicos encontram-se desalinhados e aleatórios, e de seguida em (b) ao ser aplicado o campo externo B_0, os momentos magnéticos alinham-se com esse campo e mantém esse alinhamento mesmo após ser retirado B_0. Inclusive são construídos com materiais que consigam manter o campo magnético mesmo quando o íman é submetido a um campo oposto [3] [4] [5].

Estes não necessitam de uma fonte de energia externa para continuar a produzir o campo magnético, sendo por isso utilizados em situações estáticas onde o campo magnético necessário é constante. Ao ser utilizado num circuito de levitação magnética, obtém-se um configuração simples e de baixo custo de manutenção [3].

São construídos a partir de materiais como ligas de alnico (Alumínio-Níquel-Cobalto), Samário-Cobalto, ou Neodímio-Ferro-Boro. A escolha do tipo de material depende da intensidade do campo que se pretende e da temperatura a que irá operar, pois os ímanes permanentes são sensíveis a temperaturas elevadas, podendo perder intensidade do campo magnético devido ao aquecimento [4] [5].

2.2.2 Supercondutores

Em 1911, o físico Holandês, Heike Kamerlingh-Onnes descobriu que existe uma classe de metais e componentes cuja resistência diminui para zero quando sujeitos a temperaturas abaixo de um certo valor identificado como temperatura crítica (T_c), como representado na Figura 2.13, tornando-os condutores de corrente ideais pois não causam dissipação de energia [1] [3] [6].

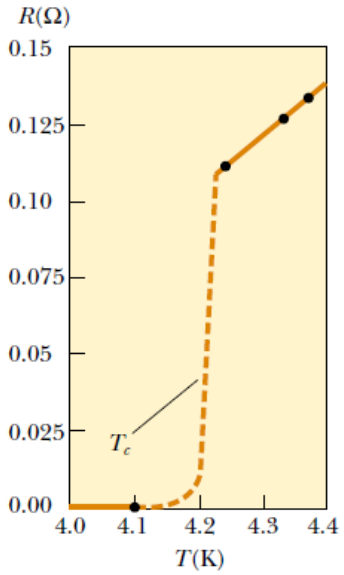


Figura 2.13: Resistência de um supercondutor a diferentes temperaturas [1]

A característica mais notável dos supercondutores é o facto de que mal uma corrente o percorra em circuito fechado, e ele se encontre abaixo de T_c, essa corrente irá persistir mesmo deixando de ser aplicado uma diferença de potencial. Essa corrente irá induzir um campo magnético persistente ao longo do tempo, desde que a temperatura seja mantida abaixo do limite crítico, tornando o supercondutor num íman permanente. Para esse fim tem de existir um sistema de refrigeração, como o de nitrogénio líquido visualizado na Figura 2.14 [1] [3] [7].

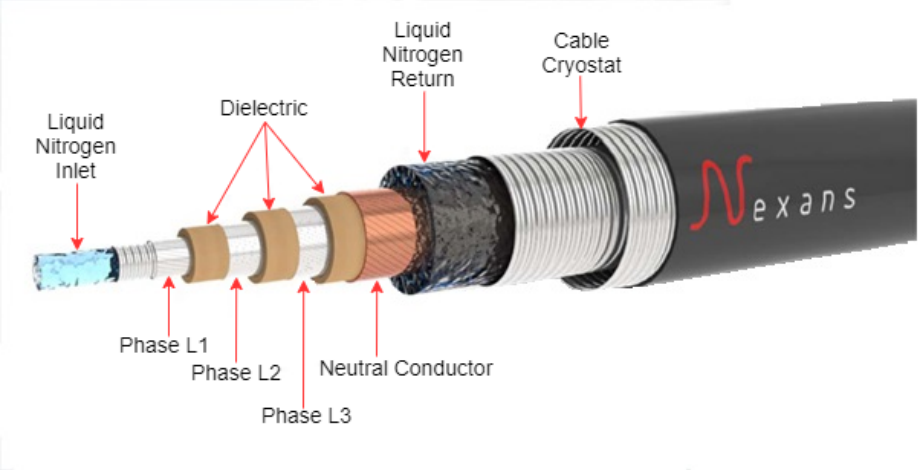


Figura 2.14: Cabo supercondutor [7]

Outro parâmetro que afeta a supercondutividade é o campo magnético ambiente, pois, se esse campo possuir uma determinada intensidade a supercondutividade é destruída. A esse dá-se o nome de campo crítico (H_c) e este depende da temperatura do supercondutor e de T_c. Sendo assim materiais com um elevado campo crítico e densidade de corrente são supercondutores apropriados. Sendo assim, o maior uso destes materiais é no desenvolvimento de ímanes, onde a magnitude dos campos magnéticos é cerca de dez vezes maior dos que os produzidos por ímanes permanentes. Estes ímanes podem também ser considerados armazenadores de energia desde que sejam mantidos abaixo de T_c [1] [3] [6].

Atualmente são muito utilizados em *scanners* de ressonância magnética, onde estão continuamente banhados em hélio líquido para se manterem abaixo de 4 graus Kelvin de modo a conservarem as propriedades supercondutoras. Estes materiais são também usados no acelerador de partículas *Large Hadron Collider* (LHC) da Organização Europeia para a Pesquisa Nuclear conhecida como CERN (antigo acrónimo para *Conseil Européen pour la Recherche Nucléaire*), que consiste num anel de 27 quilómetros de ímanes supercondutores [1] [6] [8].

As desvantagens associadas a este método advém da constante necessidade de manter o supercondutor extremamente frio, o que leva a que seja preciso sistemas de refrigeração dispendiosos, daí existir uma contínua investigação para

desenvolver materiais com propriedades supercondutores a temperaturas mais elevadas.

2.2.3 Eletroíman

O eletroíman é a tecnologia mais utilizada em sistemas de levitação magnética onde o campo magnético é produzido por uma corrente elétrica. Trata-se de um fio condutor enrolado num grande número de espiras pouco espaçadas de modo a criar um campo magnético, como um solenóide, no entanto este encontra-se enrolado à volta de dum núcleo de material ferromagnético, como representado na Figura 2.15 [3].

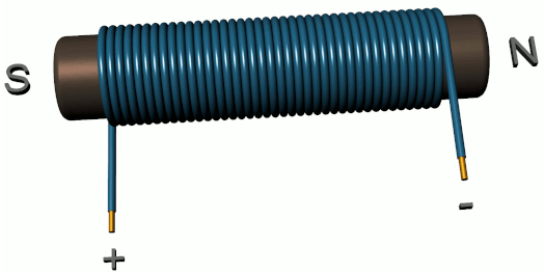


Figura 2.15: Eletroíman

O seu método de funcionamento é semelhante ao solenóide, em que de acordo com a lei de Biot-Savart, o condutor ao ser percorrido por uma corrente elétrica gera um campo magnético. O núcleo ferromagnético é utilizado para concentrar o fluxo magnético e tornando-se num íman mais forte [2] [3].

Assim ao aplicar uma corrente I é gerado o campo $\overrightarrow{B_I}$ no interior do solenóide dado pela equação (2.9), este campo por sua vez irá magnetizar o núcleo de material ferromagnético, que como estudado irá alinhar os seus momentos magnéticos de acordo com o campo externo, dando origem a um campo magnético $\overrightarrow{M}$. Somando os dois campos obtém-se o campo magnético total criado pela corrente I, dado pela equação (2.11) [2].

$$\overrightarrow{B} = \overrightarrow{B_I} + \mu_0 \overrightarrow{M} \tag{2.11}$$

A principal vantagem em utilizar eletroímanes é o facto de ser possível de variar rapidamente o campo magnético gerado, sendo apenas necessário controlar a corrente que atravessa o enrolamento. O próprio sentido do campo gerado depende também do sentido da corrente [3].

É também necessário ter em atenção que, mesmo utilizando um núcleo ferromagnético, algumas das linhas de campo são consideradas fugas de fluxo pois não atravessam o núcleo. Como representado na Figura 2.16, as linhas do campo

B_L não percorrem na totalidade o núcleo C, tomando um "atalho". Posto isto, este campo de fuga não contribui para a força exercida pelo eletroíman [3].

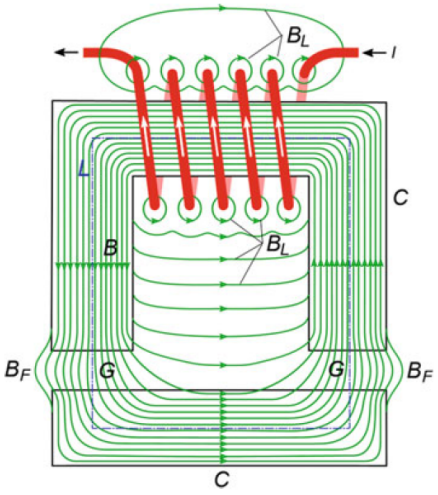


Figura 2.16: Fugas de fluxo num eletroíman [3]

Assim sendo, os eletroímanes são muito versáteis e capazes de serem utilizados em diversas aplicações. Requerem no entanto um sistema de controlo mais complexo mediante a precisão pretendida para o sistema em questão [3].

2.3 Levitação Magnética

Como explorado, pólos magnéticos repelem-se caso sejam iguais e atraem-se se forem opostos, como representado na Figura 2.17, sendo este o princípio base nos sistemas de levitação magnética [3].

A levitação magnética trata-se então da suspensão de um objeto no ar sem qualquer tipo de suporte físico com a exceção de campos magnéticos, em que uma força magnética é utilizada para contrariar os efeitos da aceleração gravítica, sendo este um fenómeno que atraiu cientistas e filósofos no passado [9].

O tipo de levitação depende da maneira de como a força aplicada está a ser utilizada para contrariar a gravidade, podendo tratar-se de suspensão magnética se caso seja usada força de atração, ou levitação magnética se for usado força de repulsão [9].

Para isso, de modo a manter o objeto suspenso, é necessário que o sistema de levitação seja também estável o suficiente para que o objeto não mude de posição nem orientação. Isto deve-se ao facto de que um objeto a levitar é um sistema muito instável, para isso é necessário um controlo que contrarie essa instabilidade. Graças aos avanços na eletrónica de potência do século passado e ao desenvolvimento de metodologias de controlo clássico e inteligente [9].

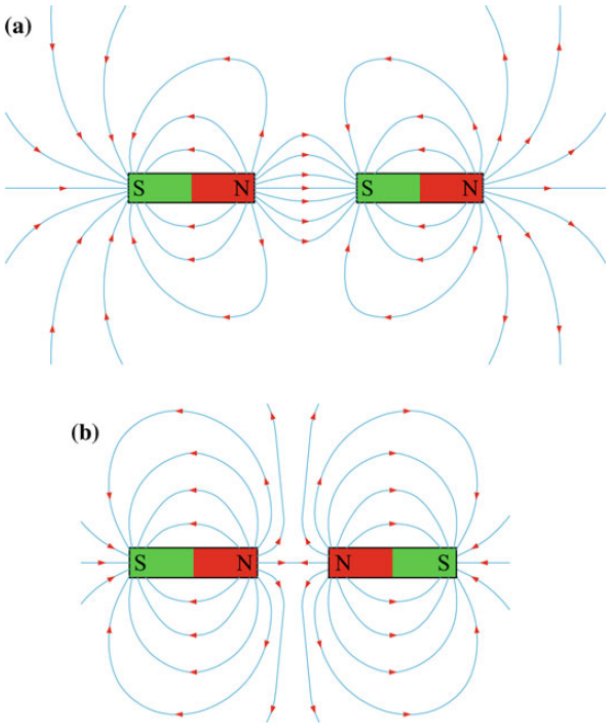


Figura 2.17: Atração (a) e repulsão (b) magnética [3]

A levitação magnética é, nos tempos correntes, uma área muito explorada no que toca a meios de transporte sem contacto físico, evitando atritos mecânicos, como por exemplo elevadores sem cabos, comboios Maglev (Figura 2.18) e pranchas de levitação (Figura 2.19). Esta também é aplicada na área de rolamentos sem fricção como os utilizados em motores elétricos [3].

Figura 2.18: Comboio Maglev Linimo [3]

É ainda explorada as possíveis aplicações da levitação magnética em certos

Figura 2.19: Prancha de levitação [3]

conceitos, como transportadores de carga (Figura 2.20), e veículos de transporte rápido pessoal (Figura 2.21), este último tem como objetivo não ser afetado por trânsito pois circula na sua própria linha, sendo capaz de transportar passageiros numa maneira rápida, segura e menos poluente [3].

Figura 2.20: Transportador de carga [3]

Figura 2.21: Transporte rápido pessoal [3]

Dependendo da complexidade do sistema desenvolvido e a sua funcionali-

dade, podem ser utilizados uma ou mais das diversas fontes de campo magnético descritas no Subcapítulo 2.2 anterior, sendo que todas tem as suas vantagens e desvantagens. Como neste trabalho será realizada a levitação com eletroíman, esta será mais aprofundada neste capítulo.

2.3.1 Técnicas de levitação magnética

Praticamente todas as fontes e métodos de produzir um campo magnético discutidos neste capítulo já foram investigados e explorados no passado quanto à sua capacidade de serem utilizados em circuitos de levitação magnética. Existem no entanto bastantes outros métodos que foram analisados, tendo maior parte deles apenas interesse académico, sendo esses [9]:

- Repulsão de materiais ferromagnéticos com ímanes permanentes;
- Levitação de materiais diamagnéticos;
- Levitação com ímanes supercondutores;
- Levitação com correntes de Eddy induzidas numa superfície ou corpo;
- Levitação a partir da força aplicada num condutor de corrente elétrica num campo magnético;
- Suspensão eletrostática utilizando um circuito RLC afinado;
- Suspensão entre um eletroíman e um corpo ferromagnético utilizando um circuito RLC afinado;
- Suspensão controlando eletroímanes CC e utilização de forças de atração.

Após alguns estudos e testes, o matemático inglês Samuel Earnshaw chegou ao seu teorema de que é impossível suspender ou levitar materiais com ímanes permanentes ou eletroímanes com uma corrente constante (ou seja campo magnético constante), excepto se o objeto que se pretende suspender ou levitar for de um material diamagnéticos, sendo mais tarde comprovado pelo físico alemão Werner Braunbeck [9].

Ao utilizar então materiais diamagnéticos, é possível obter sucesso a levitar objetos, no entanto mesmo os materiais com as características mais propícias (por exemplo grafite) só conseguem levitar em pequenas quantidades [9].

No caso da utilização de supercondutores para a levitação esta pode ser obtida de dois modos, tratando o supercondutor como um íman controlando o campo magnético gerado a partir da corrente que o atravessa, ou utilizando o efeito de Meissner. Este efeito, descoberto por Walther Meissner e Robert Ochsenfeld,

tornam alguns materiais supercondutores em diamagnéticos perfeitos, rejeitando completamente campos magnéticos externos, como representado na Figura 2.22 [3] [9].

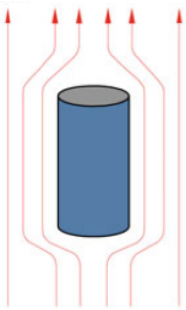


Figura 2.22: Efeito de Meissner num supercondutor [3]

A levitação por correntes de Eddy tem como base a lei de Lenz. Esta defende que ao movimentar um material condutor num campo magnético são induzidas correntes nesse material (correntes de Eddy), essas produzem por sua vez um campo magnético que se opõe ao externo, tal como demonstrado na Figura 2.23, onde ao movimentar a placa C no campo magnético B produzido pelo íman, é gerada a corrente I que produz o campo magnético representado a azul que repele o íman. O íman utilizado para neste método pode ser um íman supercondutor sendo esta outra maneira de como são utilizados na levitação magnética [3] [9].

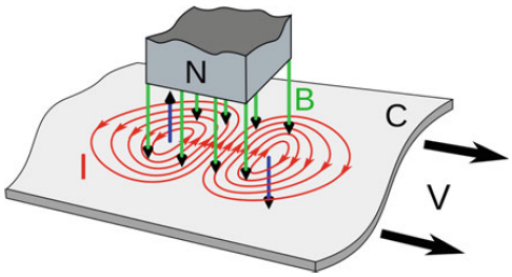


Figura 2.23: Levitação por correntes de Eddy [3]

Os estudos realizados na suspensão eletrostática com um circuito RLC determinaram que a elevada tensão necessária e potência reativa eram condições demasido adversas para a finalidade. Esse mesmo tipo de circuito foi estudado de modo a controlar a corrente de um eletroíman para realizar suspensão magnética, obtendo melhores resusltados. No entanto este circuito possui uma elevada potência reativa sendo ele maioritariamente indutivo [9].

2.3.2 Suspensão com eletroímanes CC controlados

A suspensão magnética controlando um eletroíman CC é o método mais utilizado nos tempos correntes, desde transportes terrestres avançados a rolamentos sem contactos para altas e baixas velocidades. Este é um método já estudado há mais de um século, mas que se demonstra em constante melhoria devido também a avanços tecnológicos em diversas áreas como eletrónica de potência, teoria de controlo e mecânica [9].

A construção destes sistemas é relativamente simples, no entanto precisa de um sistema de controlo em malha fechada devido à sua instabilidade inerente. Este controlador irá, após obter através de um sensor a posição do objeto suspenso em relação ao pólo do eletroíman, calcular a corrente que deve ser aplicada no circuito de modo a gerar o campo magnético necessário para colocar o objeto na posição pretendida. Para realizar este controlo é necessário estudar o sistema de modo a descrevê-lo com equações matemáticas, como a equação (2.12), para conseguir perceber como os parâmetros do sistema e a sua variação podem provocar o resultado pretendido [3] [9].

$$F_r = F_g - F_m(z, i) \tag{2.12}$$

onde F_r é a força resultante da soma da força gravitacional (F_g) com a força magnética (F_m) que depende da corrente (i) e da distância (z) do objeto ao eletroíman.

Mediante a técnica de controlo e a sintonia utilizada, características como a rigidez e o amortecimento da suspensão podem ser afinados, o que quando aplicado a meios de transportes como comboios Maglev, pode oferecer uma melhor viagem aos passageiros, o que também se deve à posição dos ímanes como representado na Figura 2.24. O mesmo não acontece nos sistemas com supercondutores pois esses têm um amortecimento intrínseco muito baixo [3] [9].

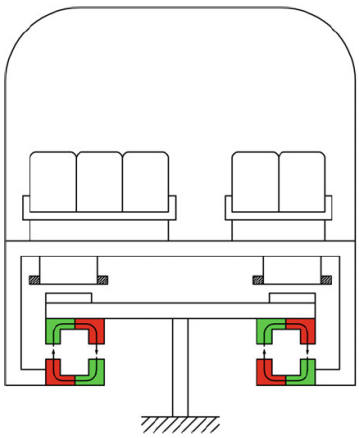


Figura 2.24: Posição dos ímanes num comboio Maglev [3]

Estes sistemas têm no entanto algumas desvantagens como as perdas por efeito de Joule mais elevadas, sendo necessário ter atenção à temperatura de funcionamento pois como o núcleo é um material ferromagnético, este pode perder as suas propriedades ao atingir temperaturas mais elevadas. Em situações como comboios onde o objeto levitado encontra-se em movimento são também geradas correntes de Eddy não lineares causando perturbações no sistema. Mesmo assim as vantagens superam as limitações, tornando esta numa técnica amplamente usada na área [3] [9].

Capítulo 3

Redes Neuronais

Neste terceiro capítulo realiza-se uma breve introdução e contextualização das redes neuronais bem como um estudo do seu enquadramento nos sistemas de controlo. Sendo assim será feita uma exposição dos conceitos base das redes neuronais e dos diferentes modelos para controlo de sistemas.

3.1 Contextualização histórica

As redes neuronais são uma área de *Machine Learning* (ML) no campo da inteligência artificial, que já é investigada desde a década de 50, que de uma maneira geral são sistemas computacionais inspirados pelo funcionamento do cérebro humano e a sua unidade fundamental, o neurónio. Essa inspiração deveu-se ao facto do cérebro humano ser capaz de processar bastante informação relativamente rápido, assim as principais características que motivaram a construção de redes neuronais são [10]:

- Adaptabilidade e tolerância a falhas;
- Capacidade de aprendizagem;
- Processamento paralelo.

A primeira tentativa de descrever um neurónio matematicamente foi realizada por Warren McCulloch e Walter Pitts em 1943, que mostraram que uma rede neuronal tinha a capacidade de computar qualquer função aritmética ou lógica, levando Donald Hebb a propor o primeiro mecanismo de aprendizagem em 1949. Isto levou a um crescido interesse pela área, onde graças a desenvolvimentos por parte de Frank Rosenblatt, Bernard Widrow e Ted Hoff obteve-se o modelo do

perceptrão e do neurónio linear adaptativo (ADALINE do inglês *Adaptive Linear Neuron*) bem com um algoritmo para o seu treino.

Mais tarde, no fim da década de 60, Marvin Minsky e Seymour Papert demonstraram as limitações do modelo ADALINE, levando a um período de obscuridade das redes neuronais. O interesse nesta tecnologia diminuiu devido à falta de ideias e limitações na capacidade de computação da época, sendo estas só voltadas a ser exploradas com mais intensidade na década de 80, graças à topologia de rede neuronal recorrente descrito por John Hopfield e ao algoritmo da retropropagação para treinar redes de múltiplas camadas de perceptrões proposto por David Rumelhart e James McClelland.

Graças à evolução na indústria dos videojogos, foram desenvolvidas unidades de processamento gráfico (GPU do inglês *Graphics Processing Units*) com alta capacidade de processamento paralelo, assim o tempo consumido nas operações com redes neuronais foi reduzido drasticamente, levando a um aumento na utilização destas técnicas.

No presente, são uma área bastante estudada com um vasto leque de aplicações e áreas, sendo muito utilizadas no processamento de imagens, problemas de otimização complexos, controlo de sistemas, reconhecimento de voz, monitorização e segurança, entre outros. Sendo relevante destacar que permitem aproximar e modelar o comportamento de sistemas não lineares e complexos através de funções compostas [11].

3.2 Arquitetura de redes

Enquanto que o cérebro é constituído por neurónios biológicos, as redes neuronais são compostas pelo equivalente artificial. Estes neurónios artificiais são fundamentais para a operação da rede neuronal pois são as suas unidades de processamento de informação. São constituídos por três elementos básicos:

- Sinapses de pesos variáveis;
- Somador de todas as entradas após serem multiplicadas pelos das sinapses;
- Função de ativação.

Um neurónio, como representado na Figura 3.1, possuí R entradas (p), cada uma dessas entradas é multiplicada por um peso sináptico (w) que representa o quanto essa entrada afeta na ativação do neurónio. De seguida todos essas multiplicações são somadas em conjunto com uma polaridade (b) que corresponde a um peso cuja entrada é sempre 1. Após a soma é utilizada a função de ativação f para calcular o valor de saída do neurónio. Esta função de ativação pode ser

uma função linear ou não linear, sendo que é escolhida de modo a satisfazer as especificações do problema a resolver, e que na maior parte das vezes são funções deriváveis pois os algoritmos de treino necessitam dessa derivada [10].

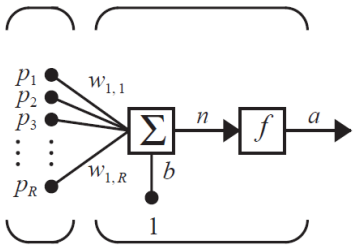


Figura 3.1: Neurónio artificial [11]

Assim o valor do neurónio pode ser calculado com a equação (3.1).

$$a = f(\mathbf{W}\mathbf{p} + b) \tag{3.1}$$

onde $\mathbf{W}$ é uma matriz 1xR com os pesos sinápticos e $\mathbf{p}$ é uma matriz Rx1 com as entradas do neurónio.

Como um neurónio, por muitas entradas que tenha, é bastante limitado em termos dos problemas que consegue resolver, foram criadas redes neuronais mono-camada (Figura 3.2). Estas são constituídas por S neurónios em paralelo, cada um com os seus pesos sinápticos e polaridades, tornando $\mathbf{W}$ num matriz SxR descrita pela equação (3.2) e $\mathbf{b}$ numa matriz Sx1.

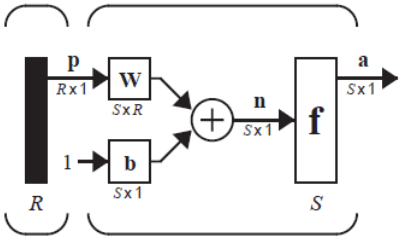


Figura 3.2: Rede mono-camada [11]

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix} \tag{3.2}$$

Estas redes mono-camada conseguem resolver apenas problemas linearmente separáveis, como completar padrões e eliminar ruídos, mas no entanto como Minsky e Papert demonstram, estas continuam muito limitadas [10]. Assim as redes neuronais mais utilizadas são as multi-camada caracterizadas por, como o

nome indica, possuírem duas ou mais camadas de neurónios em que a entrada de cada camada é a saída da anterior excepto no caso da primeira em que recebe a matriz **p**. A quantidade de neurónios em cada camada é independente entre camadas.

Uma rede com três camadas, como se verifica na Figura 3.3, tem uma saída dada pela equação (3.3) que depende de todas as entradas, pesos e polaridades das camadas anteriores. Normalmente essas camadas anteriores à camada de saída têm o nome de camadas escondidas [10] [11].

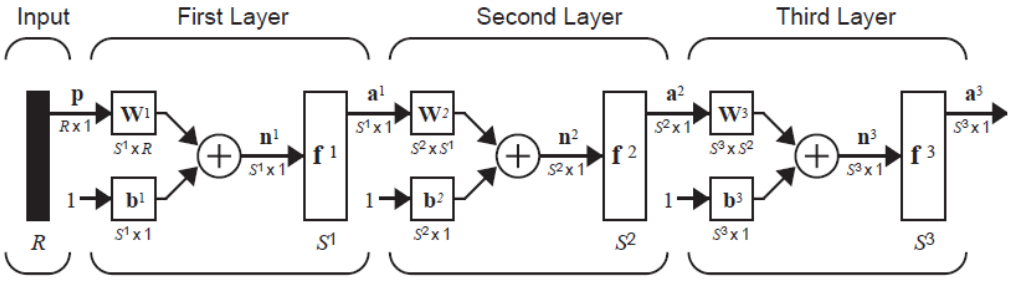


Figura 3.3: Rede multi-camada [11]

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3) \tag{3.3}$$

Todos estes modelos de redes analisados têm uma característica em comum, são estritamente proativos (*feedforward*) pois a saída que cada camada é a entrada da camada seguinte, ou seja as conexões são feitas num único sentido. Sendo assim, outra característica que se pode acrescentar a redes neuronais é a recorrência, adicionando a capacidade de utilizar valores de saídas de certas camadas como entradas anteriores. Para isso utilizam-se blocos como atrasos para a construção desse tipo de redes, como se pode ver na Figura 3.4 onde só é visível a camada de saída e as entradas da rede, sendo utilizado um bloco de atraso para realimentar a rede [11].

Todos estes parâmetros que definem a arquitetura da rede (recorrência, número de entradas, de saídas, de camadas e neurónios por camada) são definidos pelas especificações do problema em questão, sendo necessário uma ponderação da configuração requerida, pois quanto maior a rede mais complexos são os problemas que consegue resolver mas no entanto irá precisar de muitos mais recursos computacionais. Além disso uma rede demasiado complexa pode ficar demasiado ajustada aos dados de treino não conseguindo depois generalizar para novas situações [11].

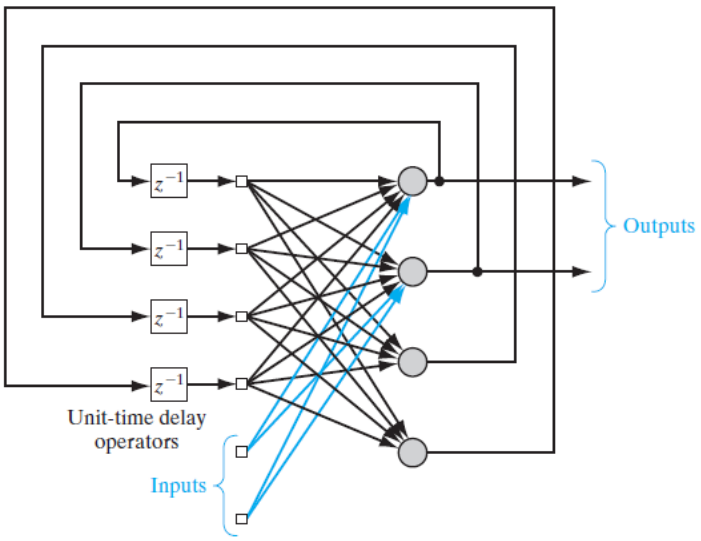


Figura 3.4: Rede recorrente [10]

3.2.1 Função de ativação

A escolha da função de ativação também tem impacto no desempenho da rede, e existem funções de ativação diferentes para diferentes fins. Sendo assim a escolha da função de ativação é realizada de modo a satisfazer as especificações do problema a resolver [11].

As funções não lineares, como a função binária e função bipolar da Figura 3.5, colocam o a saída do neurónio em apenas dois valores. Foram principalmente utilizadas nas primeiras estruturas de neurónios artificiais, enquanto que agora não são compatíveis com as técnicas de treino de redes neuronais baseadas no gradiente, pois não são deriváveis na origem e a sua derivada é 0 para todos os restantes valores.

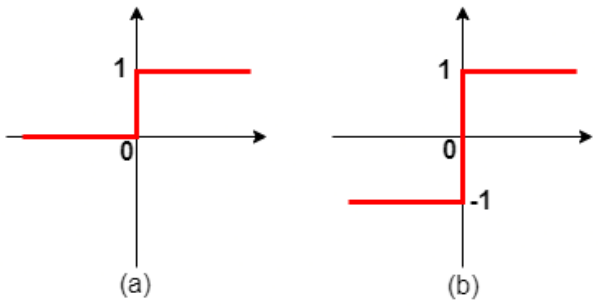


Figura 3.5: Funções de ativação:(a) binária, (b) bipolar

De seguida existem as funções lineares, muito utilizadas na arquitetura ADALINE, da Figura 3.6, onde é a função pode ser utilizada em conjunto com uma

saturação para limitar o valor de saída do neurónio como a função *Rectified Linerar Unit* (ReLU), sendo que esta não é diferenciável em 0. Computacionalmente, é possível utilizar esta função pois o algoritmo de optimização nunca chega ao mínimo global da função, mas sim a um valor bastante aproximado. Para pesos negativos o gradiente desta função é 0. É um opção bastante popular, pois nos últimos anos provou ser uma função de activação que torna a optimização do modelo mais rápida quando comparada com outras a seguir referidas [11].

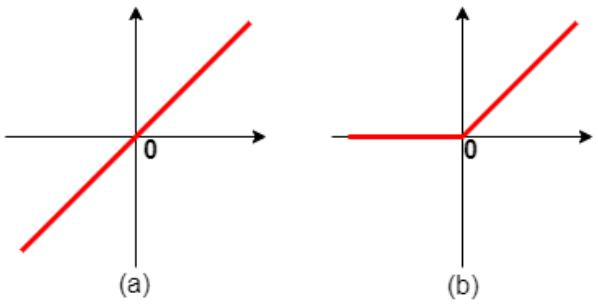


Figura 3.6: Funções de ativação:(a) linear, (b) ReLU

Outro grupo de funções de ativação são as funções logísticas da Figura 3.7, como a função sigmóide ou tangente hiperbólica, dadas pela equação (3.4) e equação (3.5), respetivamente. São funções suaves e por isso completamente deriváveis, daí a sua alta utilização nas redes neuronais implementadas nos tempos correntes, pois são compatíveis com a maior parte dos algoritmos de treino utilizados.

$$\sigma(a) = \frac{1}{1 + e^{-\lambda a}} \tag{3.4}$$

$$tanh(a) = \frac{1 - e^{-\lambda a}}{1 + e^{-\lambda a}} \tag{3.5}$$

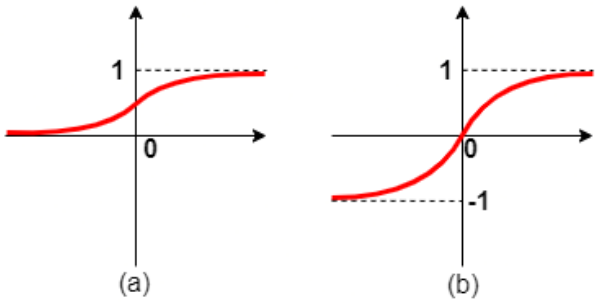


Figura 3.7: Funções de ativação:(a) sigmóide, (b) tangente hiperbólica

3.3 Treino de redes neuronais

Uma rede aprende quando melhora o seu desempenho após observar o mundo em que está inserida. Assim, o treino de uma rede consiste na alteração dos pesos das conexões e polaridades de cada neurónio de modo a atingirem um valor que permita alcançar o objetivo desejado para o problema em questão, mediante o algoritmo de treino. Existem, no entanto, vários algoritmos de treino para uma rede neuronal, sendo alguns deles [12]:

- Treino não supervisionado;
- Treino supervisionado;
- Treino semi supervisionado;
- Treino reforçado.

No treino sem supervisão a rede aprende padrões de entrada sem receber qualquer avaliação sobre as suas saídas, daí maior parte do uso desta técnica ser para ensinar a rede a agrupar entradas em categorias, o chamado *clustering* [12].

O treino reforçado é caracterizado pela rede aprender a partir de recompensas e repreensões, ou seja a sua saída é avaliada a cada entrada recebendo *feedback* positivo ou negativo, cabendo à rede detetar que ações levam ao *feedback* positivo. Este método de treino necessita no entanto de alguém que avalie as saídas da rede [12].

No caso do treino ser supervisionado é fornecido um grupo de pares entrada-saída, ou seja entradas cujas saídas são conhecidas. Assim é comparada a saída da rede com a saída objetivo, depois mediante a regra de aprendizagem definida, são alterados os pesos para o erro tender para zero, como na seguinte Figura 3.8. Ao fornecer dados de treino com erros, este passa a ser um treino semi supervisionado, pois a rede terá de detetar o erro [12].

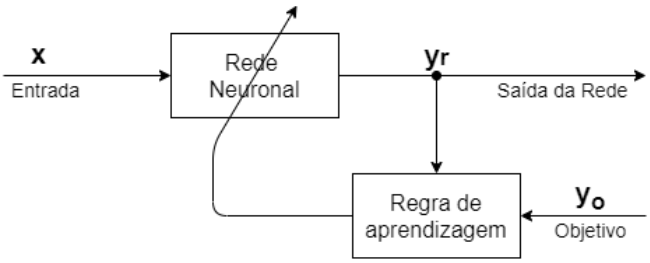


Figura 3.8: Treino supervisionado de uma rede neuronal

Nos dias de hoje a técnica mais utilizada é o treino supervisionado, por exemplo em áreas como *Deep Learning* para tarefas de classificação onde são treinadas redes com centenas de camadas sendo fornecidos milhares de dados de treino.

3.3.1 Retropropagação

Para treinar estes modelos são utilizados métodos iterativos baseados no gradiente, que minimizam a função de custo associada. O algoritmo da retropropagação, descoberto por Rumelhart e McClelland, é um método bastante utilizado no treino supervisionado de redes neuronais e é caracterizado por calcular, bastante eficientemente, o gradiente de uma função objetivo em relação aos pesos da rede para um par saída-entrada.

Este algoritmo, tal como o algoritmo de treino do ADALINE, usa como função objetivo a otimizar o erro quadrático médio (MSE do inglês *Mean Squared Error*), e deste modo calcular o desempenho do treino. O MSE é dado pela seguinte expressão:

$$F(x) = \frac{1}{n}\sum_{i}^{n}(y_o - y_r)^2 \tag{3.6}$$

onde x é a entrada da rede, y_o é o valor objetivo, y_r é o valor da rede e n é o número de amostras fornecidas à rede. Ao considerar Θ os parâmetros da rede, y_r pode ser descrito por $f(x, \Theta)$. Como é calculado o gradiente do MSE a cada iteração do treino, daí vem a necessidade dos neurónios possuírem funções de ativação deriváveis.

No caso da retropropagação, o sinal de erro é propagado no sentido retroativo, sendo calculado o quanto cada peso afeta a saída da rede. Ao analisar a equação (3.7) da soma das entradas no neurónio i da camada m, é possível verificar como todos os pesos anteriores afetam o seu valor [11].

$$n_i^m = \sum_{j=1}^{s^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m \tag{3.7}$$

Ao calcular a derivada do peso j do neurónio i da camada m e a sua polaridade, obtêm-se as seguintes equações (3.8) e (3.9) [11], respetivamente, verificando assim o que afetam:

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1} \tag{3.8}$$

$$\frac{\partial n_i^m}{\partial b_i^m} = 1 \tag{3.9}$$

Como era de esperar a polaridade apenas afeta uma entrada unitária, no entanto o peso já afeta os cálculos realizados nas camadas anteriores.

Para calcular a influência de um peso ou polaridade na variação da função objetivo usa-se a equação (3.10) e equação (3.11), respetivamente, onde é utilizada a derivada da função composta para esse fim [11].

$$\frac{\partial F}{\partial w_{i,j}^m} = \frac{\partial F}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m} \tag{3.10}$$

$$\frac{\partial F}{\partial b_i^m} = \frac{\partial F}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_i^m} \tag{3.11}$$

Ao assumir a seguinte equação:

$$s_i^m \equiv \frac{\partial F}{\partial n_i^m} \tag{3.12}$$

e ao utilizar a equação (3.8) e equação (3.9) anteriores, é possível utilizar o algoritmo da descida do gradiente para calcular os valores que os pesos e polaridades irão ter na iteração seguinte. Esse cálculo dá-se pela equação (3.13) para o peso e equação (3.14) para a polaridade [11].

$$w_{i,j}^m(n+1) = w_{i,j}^m(n) - \alpha s_i^m a_j^{m-1} \tag{3.13}$$

$$b_i^m(n+1) = b_i^m(n) - \alpha s_i^m \tag{3.14}$$

onde α é a taxa de aprendizagem. Esta afeta a variação causada pelo gradiente nos parâmetros da rede.

Em forma de matriz obtém-se a equação (3.15) e equação (3.16) para os pesos e polaridades, respetivamente, da camada m [11].

$$\mathbf{W}^m(n+1) = \mathbf{W}^m(n) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T \tag{3.15}$$

$$\mathbf{b}^m(n+1) = \mathbf{b}^m(n) - \alpha \mathbf{s}^m \tag{3.16}$$

onde $\mathbf{s}^m$ é agora a derivada parcial de F em função de todos os neurónios da camada m ($\mathbf{n}^m$). Assim a variação nos pesos é dada pela equação (3.17) e nas polaridades é a equação (3.18).

$$\Delta \mathbf{W}^m(n) = -\alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T \tag{3.17}$$

$$\Delta \mathbf{b}^m(n) = -\alpha \mathbf{s}^m \tag{3.18}$$

É necessário utilizar uma taxa de aprendizagem adequado, pois um valor alto demais causa com que a rede não convirja tornando-se instável, enquanto que um valor demasiado baixo aumenta a duração do treino, sendo ainda mais lento à medida que a derivada tende para um valor nulo [11]. Assim sendo um dos problemas do algoritmo da retropropagação é o tempo necessário para treinar a rede, quanto mais camadas e neurónios esta tiver.

3.3.2 Modificações heurísticas da retropropagação

Como o algoritmo da retropropagação básico demora bastante, foram estudados e analisadas diversas maneiras de acelerar a sua convergência. Existem agora diferentes técnicas para o atingir, sendo essas chamadas de modificações heurísticas.

Um método para evitar a instabilidade ao usar um valor mais alto de α é incluir um termo de momento de inércia. Este consiste em adicionar um parâmetro que tenha em conta a alteração realizada anteriormente aos pesos e polaridades, como expressado na equação (3.19) e equação (3.20).

$$\Delta \mathbf{W}^m(n) = \gamma \Delta \mathbf{W}^m(n-1) - (1-\gamma)\alpha \mathbf{s}^m(\mathbf{a}^{m-1})^T \tag{3.19}$$

$$\Delta \mathbf{b}^m(n) = \gamma \Delta \mathbf{b}^m(n-1) - (1-\gamma)\alpha \mathbf{s}^m \tag{3.20}$$

onde γ é um valor entre 0 e 1 chamado de coeficiente de momento.

Graças a esta alteração é possível acelerar o treino, aumentar a estabilidade e pode inclusive ajudar a ultrapassar mínimos locais na função objetivo [10][11]. Como se pode verificar na seguinte Figura 3.9, onde o treino é realizado numa rede com os mesmos parâmetros iniciais e com a mesma taxa de aprendizagem, o treino com momento é mais suave e atinge um melhor resultado em menos iterações do que o treino sem momento.

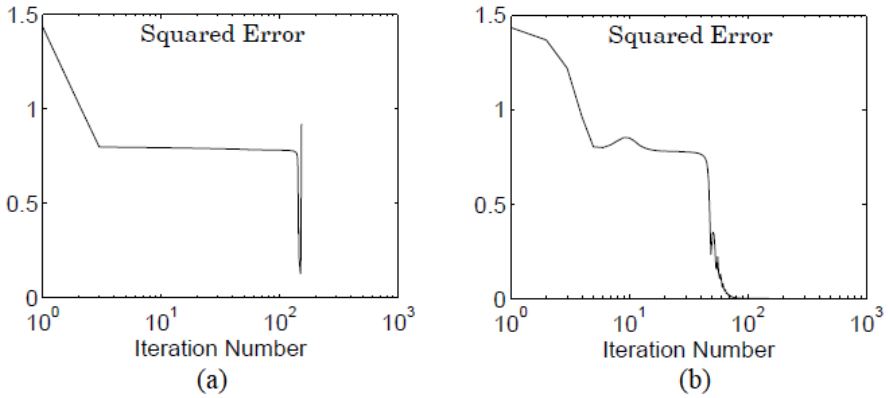


Figura 3.9: Erro durante treino:(a) Sem momento,(b) Com momento [11]

O segundo método é caracterizado por variar a taxa de aprendizagem mediante o gradiente. Neste método, após uma alteração dos pesos pode ocorrer uma das três reações [11]:

- Caso o erro aumente mais que uma certa percentagem ζ, então essa alteração é desprezada e a taxa de aprendizagem é multiplicada por $0 < \rho < 1$ e o coeficiente de momento é colocado a zero;
- Se o erro aumentar menos que ζ, a mudança é aceite mas a taxa de aprendizagem permanece constante e γ é colocado no valor original, caso tenha sido colocado a 0 anteriormente;
- Por último, caso o erro diminua, a mudança é aceite e α é multiplicado por um fator maior que 1, e caso γ tenha sido colocado a zero é colocado no seu valor original.

Na Figura 3.10 é possível ver como a taxa de aprendizagem varia mediante o erro, como por exemplo sempre que o erro diminui drasticamente a taxa de aprendizagem também diminui. De notar que a taxa é reduzida quando o erro chega a um certo valor mínimo.

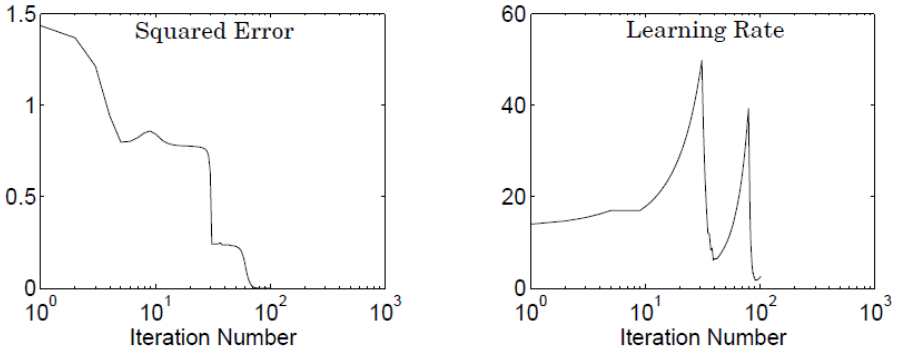


Figura 3.10: Taxa de aprendizagem variável [11]

Existem bastante algoritmos de treino de redes que aplicam estes métodos de diferentes maneiras, como por exemplo cada peso ter uma taxa de aprendizagem independente. Estas alterações tornam a convergência do algoritmo mais rápida mas requerem que mais parâmetros sejam definidos, o que causa que o desempenho do algoritmo dependa da correta definição de todos esses parâmetros.

3.4 Controlo com redes neuronais

As redes neuronais são muito utilizadas na identificação e controlo de sistemas dinâmicos graças às suas capacidades de aproximação universais. Nos sistemas

de controlo o objetivo é encontrar uma função de realimentação apropriada que mapeie as saídas medidas de modo a controlar as entradas, existindo para esse fim diversas arquiteturas que utilizam redes neuronais com múltiplas camadas [13] [14].

Normalmente no controlo neuronal de sistemas existem duas etapas, a identificação do sistema e o projeto do controlador. Na primeira fase utiliza-se uma rede neuronal para modelar o sistema que se pretende controlar, sendo que a rede é treinada com a estrutura representada na Figura 3.11 [14] [15].

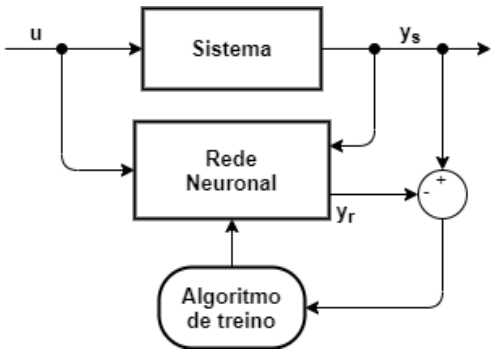


Figura 3.11: Estrutura para identificação do sistema

Como se pode ver a rede recebe duas entradas, uma para o sinal de controlo (u) e outra para saída do sistema, ambas no mesmo instante. Deste modo irá ser calculado o erro entre a saída da rede com a do sistema no instante seguinte. Na equação (3.21) encontra-se representada a função da saída da rede durante o treino.

$$y_r(n+1) = f[u(n), y_s(n)] \tag{3.21}$$

sendo que depois do treino, caso este tenha sido bem sucedido, é possível colocar como entrada a saída da rede ($y_r(n)$) em vez do sistema.

Um modelo de rede neuronal utilizado na identificação de um sistema que expande este conceito é o modelo *Nonlinear AutoRegressive eXogenous* (NARX). Este modelo utiliza também amostras do sinal de controlo e saída do sistema em instantes anteriores como na equação (3.22). É um modelo de duas camadas, a primeira com função de ativação tangente hiperbólica e um número de neurónios variável, e a segunda com um neurónio de função de ativação linear [14].

$$y_r(n+1) = f[u(n), u(n-1), ..., u(n-i), y_s(n), y_s(n-1), ..., y_s(n-j)] \tag{3.22}$$

onde i e j são números inteiros maiores que zero.

3.4.1 Estruturas de controlo

O segundo passo no controlo neuronal é utilizar o modelo obtido na identificação para treinar a rede que irá controlar o sistema. É também necessário escolher

que estrutura se pretende para o controlo, existindo diversos controladores, como por exemplo [13] [14] [15]:

- Modelo inverso direto;
- Estabilizador fixo;
- Controlador de modelo interno;
- Modelo preditivo;
- Modelo de referência;
- Inverso adaptativo.

Todos estes modelos são diferentes entre si, requerendo processos de treino diferentes.

Modelo inverso direto

Este é o modelo mais simples de controlo neuronal, sendo que a rede é treinada, como o nome indica, para modelar o inverso do sistema a controlar. Como representado na Figura 3.12, o controlador recebe como entrada a referência ($r(n)$) e a saída do sistema ($y(n)$), podendo também receber o sinal de controlo no instante anterior ($u(n-1)$) .

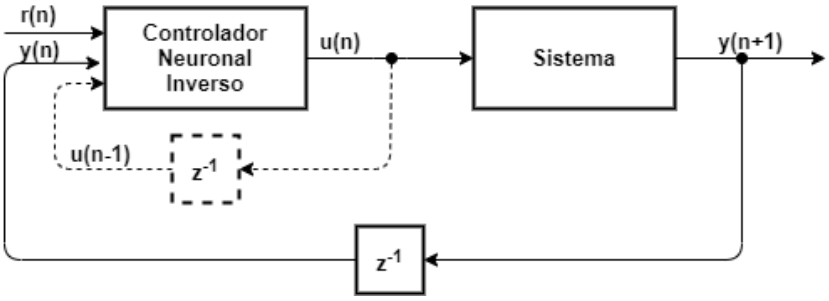


Figura 3.12: Estrutura do modelo inverso direto

Eventualmente, de modo a obter um controlador mais robusto e com um melhor desempenho é possível realimentar os sinais de controlo e saída do sistema dos instantes passados, passando a equação (3.23) a descrever a dinâmica inversa do sistema [15].

$$u(n) = f[r(n), y(n), ..., y(n-j), u(n-1), ..., u(n-1-i)] \tag{3.23}$$

Estabilizador fixo

Tal como no modelo anterior, aqui também é usado uma rede para modelar a dinâmica inversa do sistema, no entanto como representado na Figura 3.13,

é usado também um controlador estabilizador na malha de *feedback*. Assim o sinal de controlo que o sistema recebe é a soma da saída da rede com a saída do controlador estabilizador. De notar que o controlador neuronal recebe o sinal de referência como entrada e o sinal proveniente do controlador estabilizador como erro. Isto leva o algoritmo de otimização a treinar a rede de modo a diminuir sinal de erro, tornando o a rede num modelo inverso direto que irá substituir o controlador estabilizador [13].

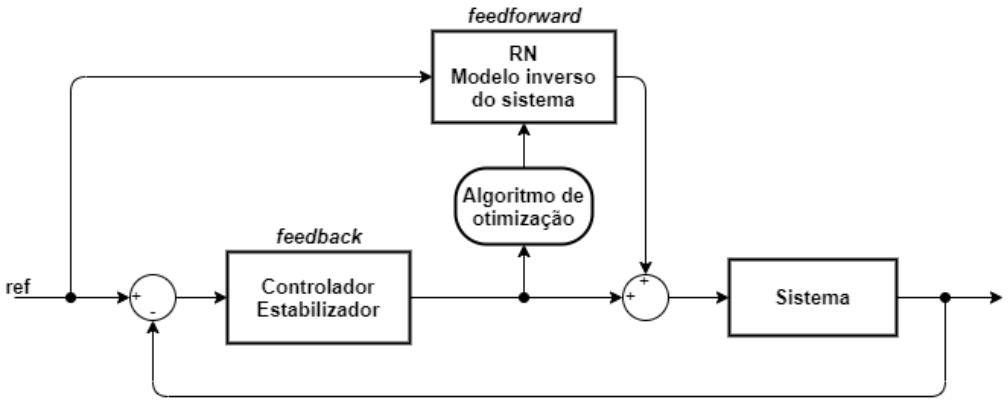


Figura 3.13: Controlador estabilizador fixo

A vantagem desta estrutura é a capacidade de começar com um sistema estável mesmo não tendo sido a rede treinada.

Controlador de modelo interno

No controlador de modelo interno, é usado o modelo inverso do sistema como controlador. No entanto, é também utilizado um rede neuronal do modelo do sistema e um filtro de primeira ordem, cuja função é garantir estabilidade em malha fechada. Neste modelo, cuja estrutura encontra-se representada na Figura 3.14, é calculado e filtrado o erro entre a saída do sistema e rede que o emula. De seguida retira-se o valor do erro ao valor de referência do controlo de modelo inverso.

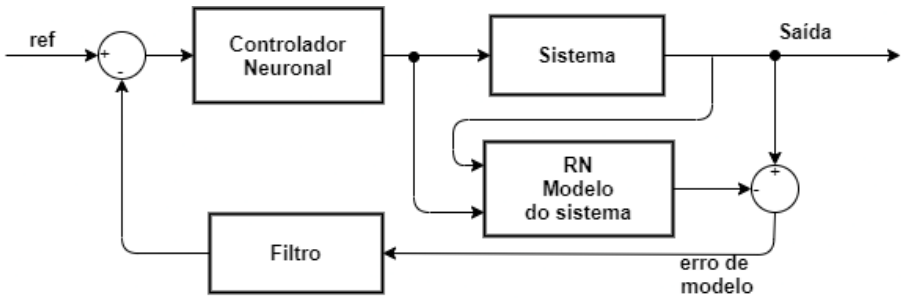


Figura 3.14: Controlador de modelo interno

Modelo preditivo

O modelo preditivo é caracterizado por calcular o sinal de controlo que irá otimizar a resposta do sistema para instantes futuros, tendo para isso de prever a resposta do sistema [13]. A sua estrutura encontra-se representada na Figura 3.15, onde é possível ver que é utilizado um modelo de referência de um sistema estável que fornece a resposta desejada ao algoritmo de otimização. Este algoritmo recebe também a resposta prevista pelo modelo neuronal do sistema, e deste modo aprimorar o controlador neuronal para que o sinal de controlo seja otimizado.

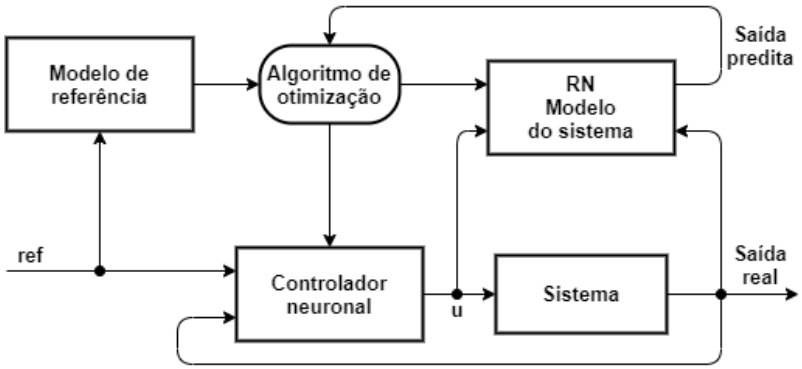


Figura 3.15: Modelo preditivo

Este método de controlo neuronal utiliza a técnica de *receding horizon* para calcular, como representado na equação (3.24), um parâmetro J que avalia a qualidade do sinal de controlo futuro, mediante as saídas previstas e anteriores. O algoritmo de otimização tem como objetivo minimizar esse parâmetro alterando o sinal de controlo e voltando a prever a resposta do sistema [14].

$$J = \sum_{j=N_1}^{N_2} (y_r(n+j) - y_m(n+j))^2 + \rho \sum_{j=1}^{N_u} (u'(n+j-1) - u'(n+j-2))^2 \tag{3.24}$$

onde N_1, N_2 e N_u definem os limites sobre os quais o erro e o sinal de controlo são analisados antes de serem aplicados ao sistema. O parâmetro ρ é utilizado para determinar o quanto a soma dos quadrados dos sinais de controlo afeta no avaliador de desempenho J.

Como é de esperar este método requer mais capacidade computacional que os anteriores, principalmente quanto maior for o intervalo de previsão, mesmo após o treino.

Modelo de referência

Este modelo utiliza um controlador neuronal para fazer o sistema seguir a resposta de um modelo de referência, como analisada na Figura 3.16, sendo que

o controlador neuronal tenta minimizar o erro entre a saída do modelo de referência e do sistema. É possível verificar a utilização de um modelo neuronal do sistema que é usado para calcular um erro de modelo, o qual serve para afinar o controlador caso o sistema sofra alguma alteração [13] [14].

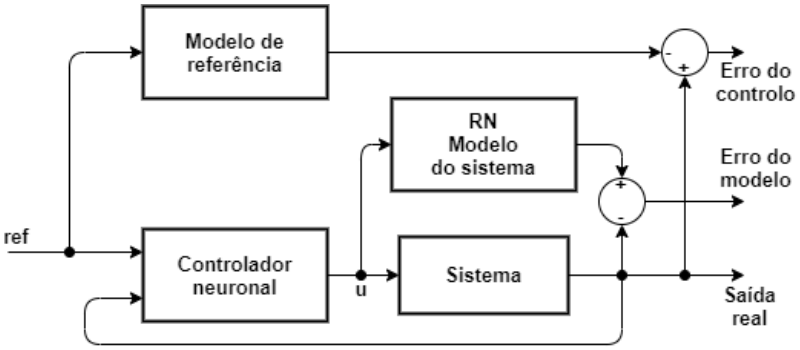


Figura 3.16: Modelo de referência

Com o modelo de referência não é necessária uma capacidade computacional acrescida depois do treino, como a técnica anterior, pois não existe a previsão do funcionamento do sistema para sinais de controlo futuros.

Inverso adaptativo

Por último, existe o modelo inverso adaptativo, representado na Figura 3.17, que utiliza três redes neuronais diferentes, uma com o modelo do sistema, outra com o modelo inverso e ainda o controlador. Neste método, o algoritmo de adaptação recebe o erro de seguimento entre o modelo de referência e o sistema, alterando os parâmetros do controlador de modo a reduzir esse erro [13].

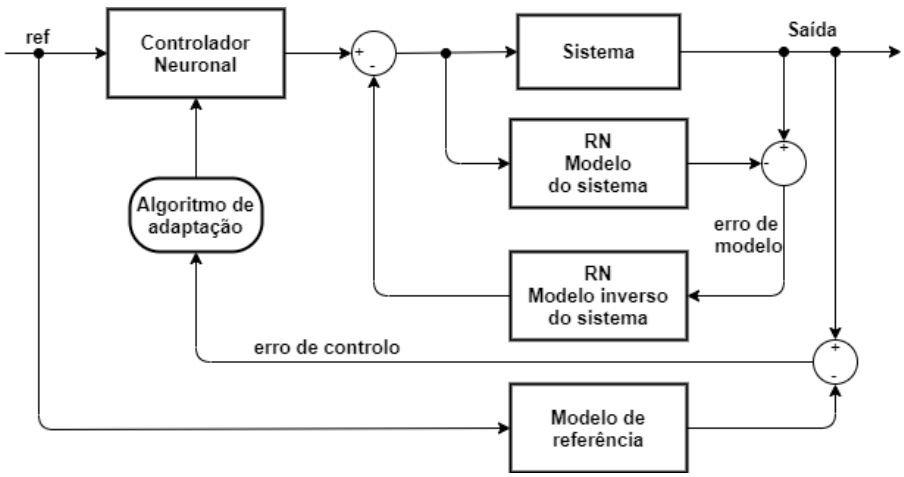


Figura 3.17: Modelo de inverso adaptativo

Outra funcionalidade deste método é o cancelamento de ruídos e perturbações

na leitura da saída do sistema, a qual é conseguida com a utilização das redes com o modelo do sistema e com o modelo inverso. Como o modelo neuronal do sistema recebe o mesmo sinal de controlo que o sistema, a diferença entre as suas saídas irá ser considerado um erro devido a perturbações e ruídos. Esse sinal de erro será processado por uma rede neuronal a modelar a dinâmica inversa do sistema, o que por sua vez irá produzir um sinal que será subtraído ao proveniente do controlador. Assim sendo o sinal de controlo aplicado ao sistema irá ter em conta a perturbação e irá tentar eliminá-la [13].

Como referido, existem bastantes configurações para a utilização de redes neuronais na modelação e controlo de sistemas. É por isso necessário escolher a melhor opção, mediante a precisão que se pretende e a capacidade de computação disponível.

Capítulo 4

Arquitetura do sistema

No atual capítulo é identificada e especificada a arquitetura geral do sistema, onde são tidas em consideração cada uma das suas componentes, explicando as suas funcionalidades, modo de funcionamento, componentes utilizados e outros possíveis dados que se consideraram relevantes.

4.1 Arquitetura geral

Como referido no Capítulo 1 o principal objetivo do trabalho é o controlo de um sistema de levitação magnética com redes neuronais, sendo primeiro simulado o sistema e controlo e subsequentemente implementado o algoritmo num micro-controlador na linguagem C. A arquitetura geral do sistema desenvolvido é descrita pela Figura 4.1, onde são representadas as ligações entre as diferentes partes do sistema.

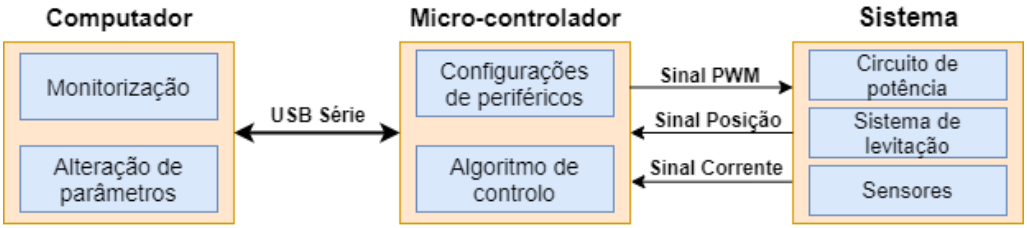


Figura 4.1: Arquitetura geral do sistema

É então possível verificar que o micro-controlador envia um sinal de *Pulse-Width Modulation* (PWM) para o sistema que permite controlar a posição no sistema de levitação e este último por sua vez, graças ao sensor de posição e sensor de corrente, envia dois sinais analógicos com uma determinada tensão ao longo do tempo representando a posição e a corrente, respetivamente. Também

é visível uma comunicação série bidirecional, através de um *Universal Serial Bus* (USB) entre o micro-controlador e o computador, com o intuito de trocar dados de modo a poder alterar parâmetros como a posição de referência e visualizar o desempenho do sistema ao longo do tempo.

Assim é possível construir um diagrama de blocos mais pormenorizado, como na Figura 4.2, identificando cada componente do sistema.

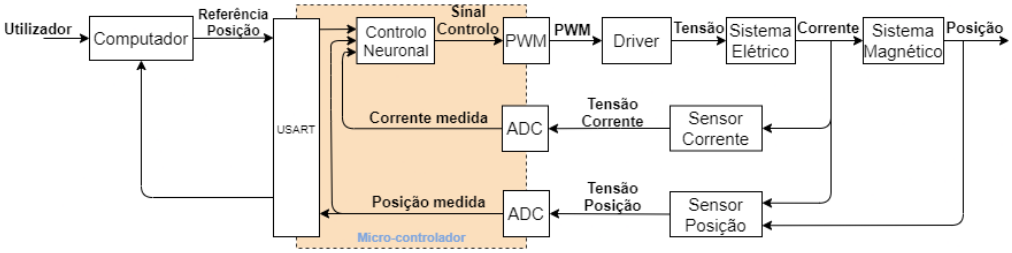


Figura 4.2: Diagrama de blocos do sistema

A partir deste diagrama será feita a análise de cada bloco de modo a identificar os seus requisitos necessários e as suas funções bem como explicar o seu modo de funcionamento.

4.2 Sistema de levitação, circuito de potência e sensores

Neste caso, o *driver* de potência é utilizado para converter o sinal de PWM proveniente do micro-controlador num sinal apropriado para o sistema de levitação utilizado, pois como era de esperar o micro-controlador não consegue fornecer ao sistema a corrente necessária sem se danificar.

4.2.1 Sistema de levitação magnética

O sistema de levitação magnética utilizado neste projeto, representado na Figura 4.3, é constituído por um eletroíman e um disco magnético.

Este conjunto de eletroíman e disco magnético é comercializado pela empresa Zeltom com o fim de ser utilizado em aplicações educacionais para o teste de sistemas de controlo.

O eletroíman trata-se de uma bobina com inúmeras espiras, cujas características encontram-se descritas na Tabela 4.1, em que a corrente que o percorre irá ser controlada de modo a suspender o disco magnético de neodímio. O disco magnético por sua vez tem 12,7 milímetros de diâmetro e 3,2 milímetros de espessura, pesando 3 gramas.

De salientar que a escolha deste sistema não adveio de nenhuma necessidade específica, sendo que este foi o disponibilizado para realização deste projeto. Além

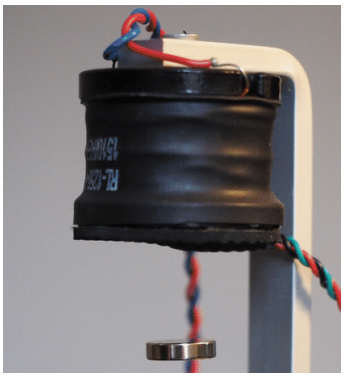


Figura 4.3: Sistema de levitação

Tabela 4.1: Características do eletroíman

Indutância (L_e)	15 mH
Resistência (R_e)	3,0 Ω
Corrente máxima	1,5 A
Diâmetro	37,3 mm
Comprimento	28,5 mm

disso, como é um sistema construído para realizar levitação magnética, os componentes nele presentes já se encontram corretamente dimensionados para um funcionamento dentro dos limites do sistema.

4.2.2 Sensor de posição

Para obter a posição do íman ao longo do tempo, é utilizado o sensor linear de efeito de Hall A1324 da Allegro Microsystems. Este sensor, cujo *pinout* é representado na Figura 4.4, é alimentado com 5 V para gerar a corrente que irá ser afetada pela corrente criada pelo campo magnético externo por indução magnética, a corrente resultante irá criar a tensão de Hall, como explicado no Capítulo 2.

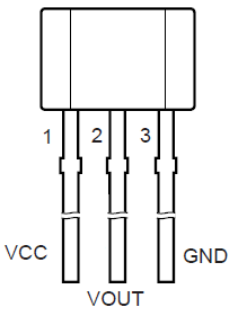


Figura 4.4: *Pinout* do sensor de efeito Hall A1324 [16]

A disposição do sensor é representada na Figura 4.5, onde se pode ver que é colocado entre o íman e eletroíman, junto a este último. Assim a tensão produzida irá variar mediante a distância (z) entre o sensor e íman, medida pelo microcontrolador. De notar, que o sensor também é afetado pelo campo magnético gerado pelo eletroíman, sendo que é necessário conseguir calcular essa parcela para obter corretamente a distância, como demonstrado mais à frente.

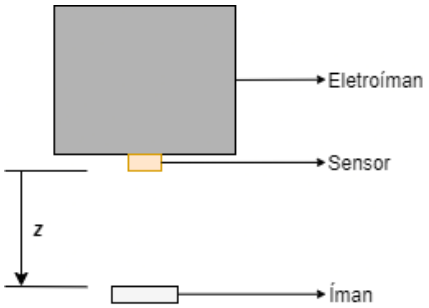


Figura 4.5: Disposição do sensor com o sistema de levitação

4.2.3 Sensor de corrente

Para conseguir medir melhor a influência causada pelo campo magnético do eletroíman, é utilizado o sensor linear de efeito Hall ACS711 também da Allegro Microsystems para medir a corrente que percorre o eletroíman.

Este é um sensor, também alimentado com 5 V, capaz de medir correntes entre -15,5 e 15,5 A, o que inclui claramente os 1,5 A máximos que podem percorrer o eletroíman. Para evitar variações na tensão de alimentação é utilizado um condensador de 0,1 μF entre o pino Vcc e GND [17]. É também utilizada uma resistência de 10 kΩ entre o pino Vcc e o pino $\overline{\text{FAULT}}$ pois este último desce para um nível de tensão baixo quando a corrente ultrapassa o máximo [17]. Assim, o esquemático da Figura 4.6 serviu para construir a *Printed Circuit Board* (PCB) da Figura 4.7.

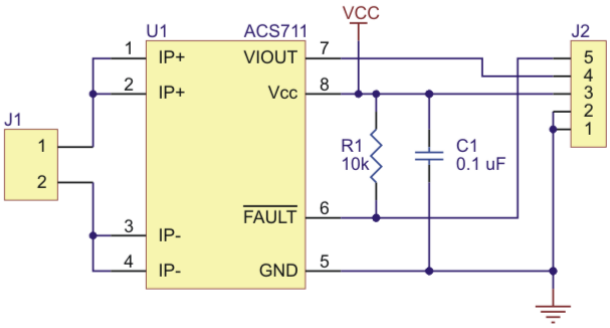


Figura 4.6: Esquemático do sensor ACS711 [18]

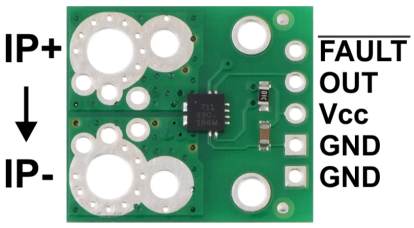


Figura 4.7: Placa de circuito impresso com sensor ACS711 [18]

4.2.4 Circuito eletrónico de potência

O controlo da posição do íman é feito com recurso à alteração da tensão fornecida ao eletroíman, sendo para esse efeito utilizado um *driver* de potência que permita a interligação entre o sistema de levitação e o micro-controlador.

Assim, é utilizado o circuito da Figura 4.8 onde o eletroíman, representado por uma série R_e e L_e dentro do retângulo laranja, utiliza um díodo de proteção (D) contra polarizações inversas devido à natureza indutiva da carga.

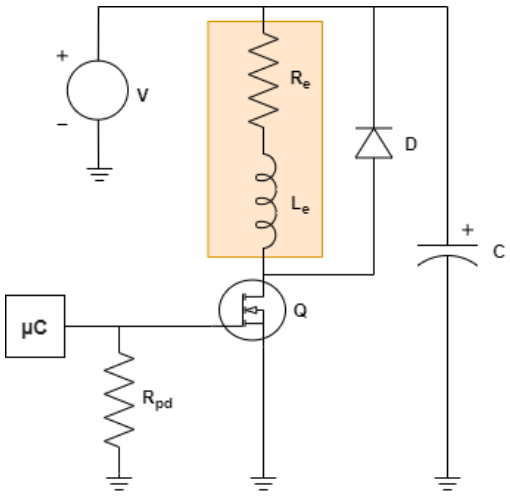


Figura 4.8: *Driver* de potência

Neste circuito é utilizado um *Metal-Oxide-Semiconductor Field-Effect Transistor* (MOSFET) como interruptor (Q), sendo o eletroíman ligado ao seu dreno e a massa do sistema à fonte. A sua porta estabelece a ligação com o micro-controlador, onde é utilizada uma resistência de *pull-down* (R_{pd}) para certificar que o MOSFET se encontra na região de corte quando o pino do micro-controlador se encontra num nível lógico baixo. É também usado um condensador (C) para garantir que a tensão fornecida pela fonte de alimentação se mantém constante de modo a não influenciar o desempenho do sistema.

Sendo assim, foram utilizados os seguintes componentes:

- **D** - Díodo *Schottky* SB520 da Diotec caracterizado por bloquear tensões até 20 V e aguentar uma corrente direta de 5 A;
- $\mathbf{R_{pd}}$ - Resistência de 4,7 kΩ e 250 mW
- **Q** - MOSFET IRF530N que suporta uma tensão de 100 V entre o dreno e fonte e uma corrente direta de 11 A;
- **C** - Condensador de 1000 μF de 16 V.

O circuito será alimentado com 5 V para, independentemente do sinal de controlo, a corrente não ultrapassar o limite suportado pelo eletroíman. Estes 5 V são provenientes do circuito regulador de tensão da Figura 4.9, onde é utilizado um LM7805CV da STMicroelectronics [19].

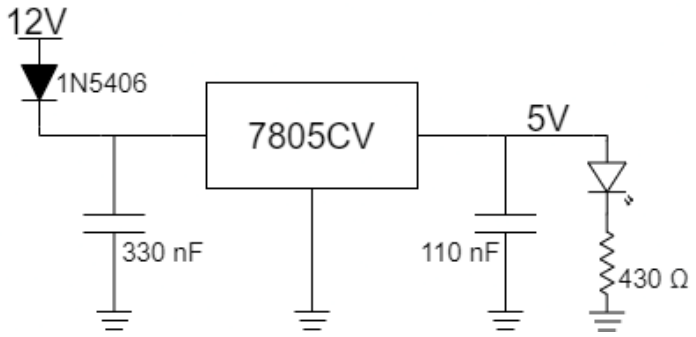


Figura 4.9: Circuito de alimentação com o regulador 7805CV

4.3 Micro-controlador

O micro-controlador tem como principais funções a configuração de periféricos, comunicação com o computador, processamento do algoritmo de controlo neuronal e leitura do sinal proveniente do sensor.

Para o caso do presente projeto foi escolhido o Arduino Mega 2560 com o micro-controlador ATmega2560 de 8 bits da Microchip.

Este micro-controlador foi selecionado principalmente devido à quantidade de memória de programa e *Static Random-Access Memory* (SRAM) disponível pois, como foi evidenciado no Capítulo 3, a utilização de redes neuronais necessita de uma elevada capacidade computacional. As suas características são as apresentadas na Tabela 4.2.

Outros requisitos a ter em conta são frequência de funcionamento elevada, que permita a atualização rápida do sistema face ao sinal do controlo, uma porta que permita a comunicação série com o computador e ainda um periférico que permita a geração de um PWM a ser aplicado aos terminais do *driver* de potência.

Tabela 4.2: Características do ATmega2560

Nome	Valor
Tensão de funcionamento	5 V
Frequência de relógio	16 MHz
Memória de programa (*Flash memory*)	256 KB
SRAM	8 KB
EEPROM	4 KB
Temporizadores	2×8 bit, 4×16 bit
Periféricos de comunicação	4×UART, 5×SPI, 1×I^2C
Periféricos de Captura/Comparação/PWM	4×Input Capture, 4×, 16×PWM

Assim, graças a todas estas funcionalidades, considera-se a escolha deste micro-controlador adequada para a realização do trabalho.

4.3.1 Controlo Neuronal

Para implementar este bloco é necessário compreender os conceitos de redes neuronais abordados no Capítulo anterior e as diferentes arquiteturas de controlo possíveis. Neste caso, este bloco requer uma posição de referência (proveniente do computador) e uma leitura da posição do íman permanente (adquirida através do sensor de efeito de Hall) por forma ao algoritmo neuronal computar a ação a tomar. A sua saída, após efetuados todos os cálculos, é um sinal de controlo que será convertido num sinal PWM no pino do micro-controlador.

Como forma de testar a validade do algoritmo é necessário observar as curvas de resposta do sistema, que podem ser obtidas através de ferramentas como por exemplo o MATLAB/Simulink. A partir destas é possível ajustar os parâmetros do controlador, por forma a que a resposta do sistema seja adequada ao funcionamento pretendido do sistema.

4.3.2 Periféricos

Como representado na Figura 4.2, são utilizados alguns periféricos do micro-controlador para ligar o sistema ao computador.

Começando pelo bloco de PWM, este converte o sinal de controlo num sinal proveniente do algoritmo de controlo neuronal num sinal PWM entre 0 e 5 V, com um certo *Duty Cycle* entre 0% e 100%. Este sinal é aplicado na porta do MOSFET presente no circuito eletrónico de potência para alterar a tensão fornecida ao eletroíman e por conseguinte a corrente que gera o campo magnético.

O bloco de *Analog-to-Digital Converter* (ADC) realiza a conversão do sinal proveniente do sensor. Neste micro-controlador o ADC tem uma resolução de 10 bits, assim a partir da equação (4.1) é possível calcular a menor variação que se

consegue ler com o sensor que, como é possível verificar é suficientemente baixa para ter um bom controlo.

$$R = \frac{5V}{2^{10}} = 4,88 \quad \text{mV} \tag{4.1}$$

Para comunicar com o computador é utilizado o *Universal Synchronous and Asynchronous Serial Receiver and Transmitter* (USART), pois permite estabelecer uma comunicação *full-duplex* com uma elevada taxa de transmissão. Esta comunicação será utilizada para transmitir dados como a posição de referência para o micro-controlador e a posição atual do íman para o computador.

4.4 Construção do sistema eletrónico

A construção do sistema descrito é relativamente simples, sendo implementado numa *breadboard*, como é possível verificar na Figura 4.10, possibilitando a realização dos testes descritos nos capítulos seguintes. De notar que é utilizado um dissipador térmico no regulador de tensão para a manter a temperatura no regular abaixo do limite definido, prevenindo assim que a corrente seja limitada. A possível implementação numa PCB encontra-se descrita no Anexo A.

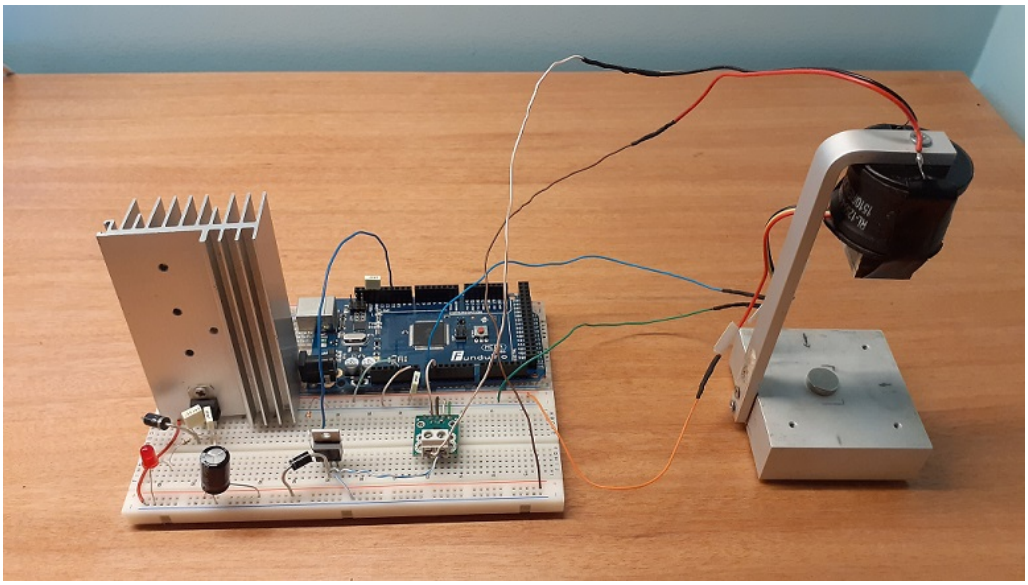


Figura 4.10: Sistema implementado nas *breadboards*

É utilizado um *on-board Light-Emitting Diode* (LED) da placa Arduino, que corresponde ao pino digital 13 da placa ou pino PB7 do micro-controlador ATmega2560, para verificar se o algoritmo implementado funciona corretamente. Este LED irá ser controlado dentro da rotina de interrupção que ditará o período de amostragem.

Capítulo 5

Modelação e análise do sistema

No presente capítulo é analisado o sistema de levitação magnética de modo a realizar a sua modelação, tanto por função de transferência bem como por espaços de estados. Deste modo serão criados alguns controladores de modo a comprovar a correta modelação do sistema de levitação.

5.1 Modelação do sistema eletromagnético

O sistema em questão controla o campo magnético gerado pelo eletroíman de modo a levitar o disco de neodímio. Este sistema é caracterizado por ser não linear e altamente instável em malha aberta. Por isso, de modo a obter uma levitação constante e estável do íman, é necessário projetar um controlador robusto em malha fechada que neutralize essas características do sistema.

Qualquer tentativa de projetar um controlador deve começar com uma previsão do desempenho do sistema a controlar. Essa previsão é baseada numa descrição matemática das características dinâmicas do sistema chamada de modelo matemático. Para muitos sistemas físicos, os modelos matemáticos úteis são descritos em termos de equações diferenciais.

Após obter as equações diferenciais do sistema em análise, é necessário utilizar a transformada de Laplace para chegar à função de transferência do sistema, que representa o quociente entre a transformada de Laplace da resposta sobre a da entrada.

Neste caso, o sistema será controlado através da tensão fornecida aos terminais do sistema elétrico, o que originará a corrente que gera o campo magnético. Assim, o sistema eletromagnético pode ser representado pela Figura 5.1.

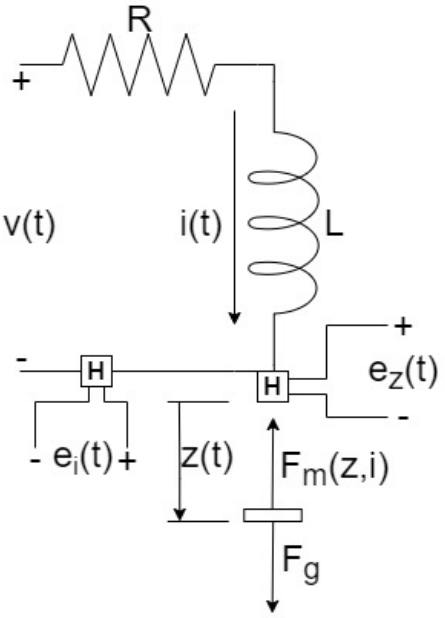


Figura 5.1: Sistema a controlar

O sinal $v(t)$ é a tensão fornecida, $i(t)$ a corrente gerada, $e_z(t)$ e $e_i(t)$ a tensão nos terminais dos sensores Hall de posição e corrente, respetivamente, $z(t)$ a distância do disco magnético ao sensor, F_m a força magnética gerada pelo campo magnético do eletroíman e F_g a força causada pela gravidade.

No sistema magnético, a força de atração causado pelo campo magnético gerado depende da corrente que percorre o eletroíman e da distância a que o íman se encontra do pólo do eletroíman. Como é de esperar, quanto maior for a corrente mais intenso será o campo gerado e à medida que a distância aumenta a influência do campo do eletroíman diminuirá. Assim a força magnética pode ser descrita através da equação (5.1) [20][21].

$$F_m(z,i) = k\frac{i(t)}{z(t)^4} \tag{5.1}$$

onde k representa a constante magnética característica do sistema em questão.

O objetivo é gerar uma força magnética que anule a força gravítica, ou seja, a força resultante no disco tem de ser nula. De acordo com a segunda lei de Newton, a força resultante num corpo com uma certa massa (m) é o produto de m com a sua aceleração. Assim, a força resultante no íman pode ser descrita pela equação (5.2), tendo em conta a equação (2.12) que foi previamente explicada no Capítulo 2.

$$m\frac{\mathrm{d}^2 z(t)}{\mathrm{d}t^2} = mg - k\frac{i(t)}{z(t)^4} \tag{5.2}$$

onde g é a aceleração gravítica.

Como é possível observar, o sistema é não linear, então para conseguir desenvolver um controlador é necessário linearizar o sistema em torno do seu ponto de equilíbrio, ou seja para uma corrente e posição específicas [22]. Sendo que o sistema se iria encontrar nas condições da equação (5.3) para a aceleração, equação (5.4) para a posição e equação (5.5) para a corrente.

$$\frac{\mathrm{d}^2 z(t)}{\mathrm{d}t^2} = 0 \tag{5.3}$$

$$z(t) = z_{eq} \tag{5.4}$$

$$i(t) = i_{eq} \tag{5.5}$$

onde i_{eq} é a corrente de equilíbrio e z_{eq} a posição de equilíbrio.

Como a sua aceleração no ponto de equilíbrio é zero, a força resultante no disco é nula, obtendo a equação (5.6) onde se confirma que no ponto de equilíbrio a força gravítica é igual à força de atração magnética.

$$mg = k\frac{i_{eq}}{z_{eq}^4} \tag{5.6}$$

Reorganizando a equação é possível chegar ao valor da constante magnética, sendo esta descrita pela equação (5.7), que depende da posição e corrente, pois estas por sua vez dependem do objeto que se pretende levitar.

$$k = mg\frac{z_{eq}^4}{i_{eq}} \tag{5.7}$$

Para linearizar a força magnética em torno de z_{eq} e i_{eq} é utilizada a expansão em série de Taylor. No entanto é necessário ter em conta que a corrente e posição apenas podem variar ligeiramente em torno do ponto de equilíbrio. Assim a corrente dada pela equação (5.8) e a posição na equação (5.9) são a soma do seu valor de equilíbrio com um pequena variação.

$$i(t) = i_{eq} + \Delta i(t) \tag{5.8}$$

$$z(t) = z_{eq} + \Delta z(t) \tag{5.9}$$

Ao substituir z na aceleração obtém-se a equação (5.10), pois a derivada de z_{eq} é zero.

$$\frac{\mathrm{d}^2(z_{eq}+\Delta z(t))}{\mathrm{d}t^2} = \frac{\mathrm{d}^2\Delta z(t)}{\mathrm{d}t^2} \tag{5.10}$$

A expansão de Taylor da força magnética em torno do ponto de equilíbrio é então dada pela equação (5.11).

$$F_m(z(t),i(t)) = F_m(z_{eq},i_{eq}) + \frac{\partial F_m(z_{eq},i_{eq})}{\partial z(t)}(z(t)-z_{eq}) + \frac{\partial F_m(z_{eq},i_{eq})}{\partial i(t)}(i(t)-i_{eq}) \tag{5.11}$$

Calculando as derivadas parciais e simplificando para as pequenas variações das variáveis, obtém-se a equação (5.12).

$$F_m(\Delta z(t),\Delta i(t)) = k\frac{i_{eq}}{z_{eq}^4} - k\frac{4i_{eq}}{z_{eq}^5}\Delta z(t) + k\frac{1}{z_{eq}^4}\Delta i(t) \tag{5.12}$$

Assim a força resultante é dada pela equação (5.13), e como provado na equação (5.6) a força gravítica é igual à força magnética no ponto de equilíbrio, o que simplifica a força resultante originando a equação (5.14).

$$m\frac{\mathrm{d}^2\Delta z(t)}{\mathrm{d}t^2} = mg - k\frac{i_{eq}}{z_{eq}^4} + k\frac{4i_{eq}}{z_{eq}^5}\Delta z(t) - k\frac{1}{z_{eq}^4}\Delta i(t) \tag{5.13}$$

$$m\frac{\mathrm{d}^2\Delta z(t)}{\mathrm{d}t^2} = k\frac{4i_{eq}}{z_{eq}^5}\Delta z(t) - k\frac{1}{z_{eq}^4}\Delta i(t) \tag{5.14}$$

Ao substituir k pela equação (5.7) é possível simplificar ainda mais o sistema, obtendo a equação (5.15) da aceleração do disco.

$$\frac{\mathrm{d}^2\Delta z(t)}{\mathrm{d}t^2} = \frac{4g}{z_{eq}}\Delta z(t) - \frac{g}{i_{eq}}\Delta i(t) \tag{5.15}$$

Analisando agora o sistema elétrico, este é constituído por uma resistência e indutância em série, o que pode ser descrito pela equação (5.16) da tensão aplicada em função da corrente gerada.

$$v(t) = Ri(t) + L\frac{\mathrm{d}i(t)}{\mathrm{d}t} \tag{5.16}$$

Tal como a corrente e a posição, também a tensão aplicada é a soma do seu valor de equilíbrio com uma pequena variação. Ou seja, no ponto de equilíbrio a tensão aplicada obedece à condição da equação (5.17).

$$v_{eq} + \Delta v(t) = R(i_{eq}+\Delta i(t)) + L\frac{\mathrm{d}(i_{eq}+\Delta i(t))}{\mathrm{d}t} \tag{5.17}$$

Desenvolvendo a equação obtém-se as parcelas constantes da equação (5.18), que permite calcular o valor de equilíbrio da tensão, e as parcelas das pequenas variações da equação (5.19).

$$v_{eq} = Ri_{eq} \tag{5.18}$$

$$\Delta v(t) = R\Delta i(t) + L\frac{\mathrm{d}\Delta i(t)}{\mathrm{d}t} \tag{5.19}$$

Com as duas equações diferenciais do sistema linearizadas em torno do ponto de equilíbrio é possível utilizar a transformada de Laplace para obter a função de transferência na equação (5.20) da variação da distância em função da variação da tensão.

$$\frac{\Delta Z(s)}{\Delta V(s)} = \frac{-\frac{g}{i_{eq}}}{(Ls+R)(s^2 - \frac{4g}{z_{eq}})} \tag{5.20}$$

Para representar o sistema em espaço de estados considera-se como entrada a variação da tensão e como saída a variação da posição do disco. Para a matriz de estados $\mathbf{x}$ considera-se que $[x_1\ x_2\ x_3]^T$ é igual a $[i\ z\ \dot{z}]^T$, assim é possível descrever o sistema com a equação de estados (5.21) e a equação de saída (5.22).

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -\frac{R}{L} & 0 & 0 \\ 0 & 0 & 1 \\ -\frac{g}{i_{eq}} & \frac{4g}{z_{eq}} & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} \frac{1}{L} \\ 0 \\ 0 \end{bmatrix} u(t) \tag{5.21}$$

$$y(t) = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \mathbf{x}(t) \tag{5.22}$$

Quanto aos sensores, a tensão produzida pelo sensor de corrente pode ser descrita pela equação (5.23) enquanto que a tensão do sensor de posição é dada pela equação (5.24). Ambos são sensores de efeito de Hall, sendo que produzem uma tensão mediante o campo magnético resultante. Outra característica que possuem é a sua tensão de repouso, ou seja a tensão que eles produzem quando nenhum campo é aplicado, o que em ambos, segundo [16] e [17], é próxima de metade da tensão de alimentação.

$$e_i(t) = mf(i(t)) + b \tag{5.23}$$

$$e_z(t) = \alpha + \beta g(z(t)) + \gamma f(i(t)) \tag{5.24}$$

onde m, b, α, β, e γ são as constantes dos sensores a determinar.

Como é possível aferir, ambos dependem de uma parcela constante (tensão de repouso) e de uma constante multiplicada por uma função da corrente. No entanto, o sensor de distância depende também de uma parcela em função da distância.

5.2 Parametrização do sistema

Para conseguir obter os valores da corrente e posição de equilíbrio, é agora necessário analisar o funcionamento dos sensores e sistema e assim obter os seus parâmetros. Em todos os testes foi utilizado um Arduino Mega 2560 para aplicar ao sistema um sinal PWM de 1000 Hz e para ler os sensores.

Começando por analisar os sensores em repouso, ou seja não está nenhum campo magnético a ser aplicado, chega-se aos valores da Tabela 5.1, onde os ADCs utilizados são de 10 bits, cujas medições são quantificadas com *Analog-to-Digital Units* (ADU). É possível ver que ambos os sensores têm valores muito próximos de metade da tensão de alimentação.

Tabela 5.1: Tensão medida nos sensores em repouso

Sensor de posição		Sensor de corrente	
ADC_z(ADU)	e_z(V)	ADC_i(ADU)	e_i(V)
508	2,4829	516	2,5220

De seguida foi aplicado um PWM de 8 bits ao sistema desde 0 a 5 V. Assim foi possível obter a influência da corrente em cada sensor. Os valores obtidos encontram-se na Tabela B.1 presente no Anexo B. De notar que aos valores obtidos com o ADC subtraiu-se o valor de repouso constante para se perceber melhor a influência da corrente. Foi então construído o gráfico da Figura 5.2, da tensão de cada sensor mediante a corrente.

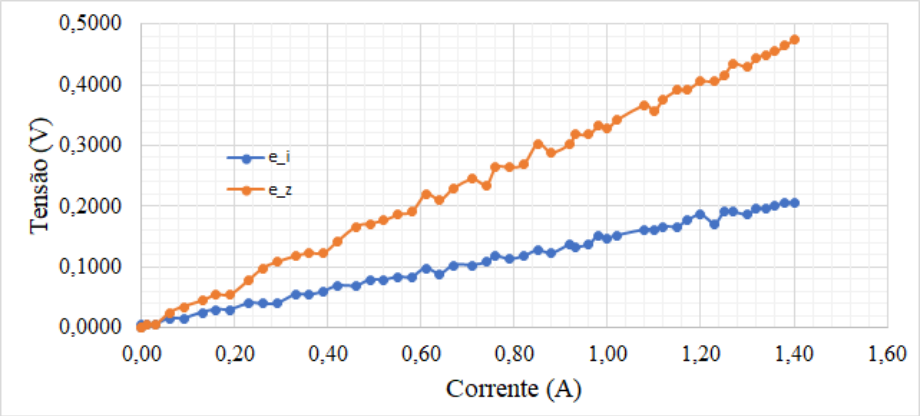


Figura 5.2: Influência da corrente em cada sensor

Analisando o gráfico e utilizando as ferramentas do Microsoft Excel foi realizada uma regressão linear obtendo a equação (5.25) e equação (5.26) para a influência da corrente para o sensor de corrente e distância, respetivamente.

$$e_{ii}(t) = 0,1447i(t) + 0,0029 \tag{5.25}$$

$$e_{zi}(t) = 0,3339i(t) - 0,0020 \tag{5.26}$$

onde m=0,1447 V/A e γ=0,3339 V/A.

Verifica-se que ambos têm uma relação proporcional com a corrente, onde a parcela constante pode ser desprezada pois na equação final de cada sensor esse valor corresponde ao valor de repouso.

O último teste aos sensores trata de analisar a influência do disco magnético no sensor de distância. Para isso foi colocado o disco na base do sistema a 8,7 cm do sensor e depois foram utilizadas as peças de madeira da Figura 5.3 para reduzir gradualmente a distância.

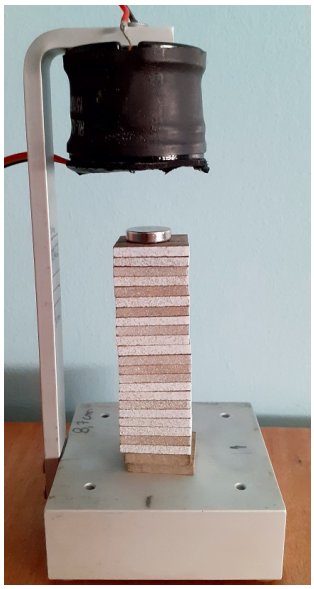


Figura 5.3: Medições da distância do íman ao sensor

A espessura de cada peça foi cuidadosamente medida com um paquímetro de modo a construir a Tabela 5.2. Tal como na influência da corrente, foi subtraído ao valor medido o valor de repouso do sensor para obter a variação devido à distância.

Com estes valores foi construído o gráfico da Figura 5.4, onde se averigua que a tensão aumenta à medida que a distância diminui, o que significa que as linhas de campo do campo magnético do íman têm o mesmo sentido que as do

Tabela 5.2: Influência da distância no sensor

z(cm)	z(m)	ADC(ADU)	e_z
8,700	0,08700	1	0,0049
8,450	0,08450	1,3	0,0064
8,165	0,08165	1,6	0,0078
7,880	0,07880	2	0,0098
7,595	0,07595	2,3	0,0112
7,325	0,07325	2,6	0,0127
7,055	0,07055	3	0,0147
6,800	0,06800	3	0,0147
6,540	0,06540	3,5	0,0171
6,270	0,06270	4	0,0196
6,000	0,06000	4,5	0,0220
5,730	0,05730	5	0,0244
5,455	0,05455	6	0,0293
5,170	0,05170	7	0,0342
4,895	0,04895	8,5	0,0415
4,610	0,04610	10,8	0,0528
4,310	0,04310	13	0,0635
4,000	0,04000	17	0,0831
3,735	0,03735	20	0,0978
3,435	0,03435	26	0,1271
3,150	0,03150	35	0,1711
2,870	0,02870	46	0,2248
2,670	0,02670	61	0,2981
2,340	0,02340	84	0,4106
2,055	0,02055	125	0,6109

campo produzido pelo eletroíman, pois ambos aumentam a tensão do sensor. Assim conclui-se que os campos magnéticos atraem-se pois o íman e eletroíman encontram-se de lados opostos do sensor.

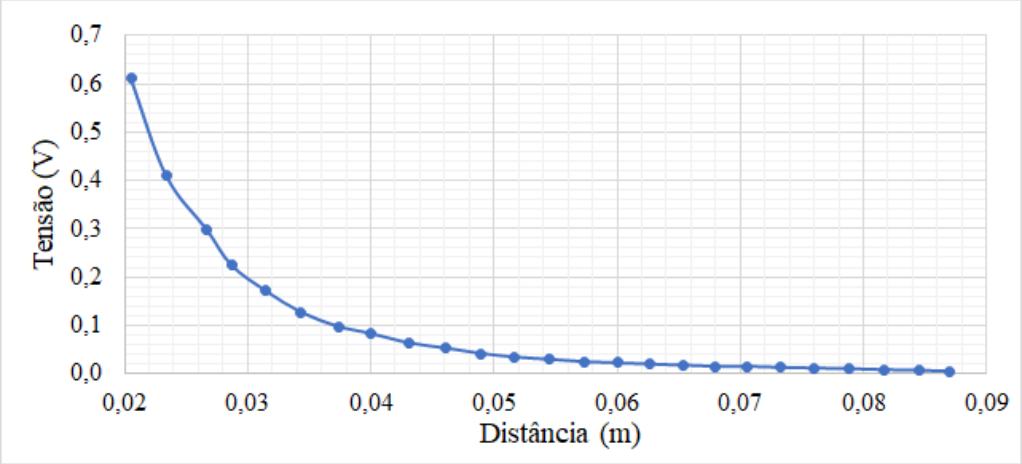


Figura 5.4: Influência da distância no sensor de posição

Novamente, utilizando o Microsoft Excel foi possível obter a linha de tendência que melhor aproxima a influência da distância, chegando assim à equação (5.27).

$$e_{zz}(t) = 2,7431 \times 10^{-6} \frac{1}{z(t)^{3,1896}} \tag{5.27}$$

De modo a simplificar a equação anterior, obteve-se a linha de tendência para a influência da distância entre os 2 cm e 4 cm que se encontra dentro da zona de funcionamento do sistema, sendo dada pela equação (5.28).

$$e_{zz2}(t) = 4,4941 \times 10^{-6} \frac{1}{z(t)^3} \tag{5.28}$$

onde β=4,4941×10^{-6} Vm^3.

Com estas análises é possível chegar à equação (5.29) e equação (5.30) do sensor de corrente e sensor de distância, respetivamente, juntando a influência da corrente e distância ao valor de repouso.

$$e_i(t) = 0,1447i(t) + 2,5220 \tag{5.29}$$

$$e_z(t) = 2,4829 + 4,4941 \times 10^{-6} \frac{1}{z(t)^3} + 0,3339i(t) \tag{5.30}$$

Com os sensores parametrizados é necessário obter a corrente para a posição de equilíbrio. Para isso foi realizado o teste de aumentar gradualmente a tensão aplicada ao eletroíman até o disco ser atraído, sendo isto realizado para diferentes distâncias. Na Tabela 5.3 encontram-se os valores obtidos, onde $\mathrm{I_{rms}}$ é o valor obtido com o multímetro e I o valor calculado com a tensão aplicada e resistência.

Tabela 5.3: Corrente necessária para atrair o íman a diferentes distâncias

z(cm)	$\mathrm{V_{PWM}}$(V)	$\mathrm{I_{rms}}$(A)	I(A)	$\mathrm{ADC_i}$(ADU)	$\mathrm{e_i}$(V)	$\mathrm{ADC_z}$(ADU)	$\mathrm{e_z}$(V)
2,055	0,569	0,12	0,318	520	2,542	644	3,148
2,340	1,078	0,29	0,419	525	2,566	613	2,996
2,670	1,471	0,42	0,534	529	2,586	598	2,923
2,870	1,902	0,56	0,696	533	2,605	592	2,893
3,150	2,392	0,72	0,845	538	2,630	592	2,893
3,435	3,039	0,91	1,075	542	2,649	597	2,918
3,735	3,686	1,1	1,298	549	2,683	606	2,962
4,000	4,569	1,32	1,582	555	2,713	614	3,001

Analisando os valores e o desempenho do sistema, todos os valores que se encontram na Tabela 5.3 foram retirados quando o disco ascendeu até ao sensor, ou seja para uma força magnética magnética maior que a força gravítica. No entanto, é nos instantes anteriores que o disco se encontrava no ponto de equilíbrio.

Para a distância pretendida de 2 cm a corrente fornecida é cerca de 0,27 A. Neste caso é escolhida a corrente mediante a tensão fornecida e não a corrente eficaz medida com o multímetro. Como é de esperar é uma corrente ligeiramente mais baixa que os 0,32 A calculados.

Como será utilizado um micro-controlador para implementar os algoritmos de controlo, torna-se necessário reduzir e simplificar os cálculos realizados. Os parâmetros do sistema têm então a tensão convertida de V para ADU, e também

para se obter uma melhor resolução do desempenho do sistema a distância será medida em centímetros. Os parâmetros convertidos do sistema encontram-se na Tabela 5.4.

Tabela 5.4: Parâmetros convertidos do sistema

Parâmetros	Valor	Unidades
α	508	ADU
β	1090	ADUcm^3
γ	68,274	ADU/A
m	29,569	ADU/A
b	516	ADU
z_{eq}	2	cm
e_{eq}	0,27	A
k	174,4	cm^{5}Kg/s^{2}A

Com estes valores, as novas equações dos sensores passam a ser a equação (5.31) e equação (5.32) para o sensor de corrente e distância, respetivamente.

$$ADC_i(t) = 29,569i(t) + 516 \tag{5.31}$$

$$ADC_z(t) = 508 + 1090\frac{1}{z(t)^3} + 68,274i(t) \tag{5.32}$$

Por último, foi utilizado a ferramenta Solver do Microsoft Excel para otimizar os parâmetros dos sensores. Para isso foram calculados os valores da corrente e distância para diferentes pontos de funcionamento do sistema com as equações anteriores (5.31) e (5.32). De seguida foi calculado o erro entre os valores medidos e calculados. O Solver foi utilizado para variar os valores dos parâmetros de modo a reduzir a soma do módulo dos erros. Na Tabela 5.5 encontra-se a comparação entre os valores reais, calculados e otimizados.

Tabela 5.5: Valores da distância otimizados

Ponto	z(cm)	ADC$_i$ (ADU)	ADC$_z$ (ADU)	z equação (cm)	z otimizado (cm)	Erro	Erro após otimização
1	2,055	520	644	2,044	2,061	0,0107	0,0059
2	2,340	525	613	2,338	2,357	0,0015	0,0168
3	2,670	529	598	2,612	2,632	0,0579	0,0376
4	2,870	533	592	2,871	2,894	0,0006	0,0239
5	3,150	538	592	3,154	3,184	0,0043	0,0338
6	3,435	542	597	3,288	3,325	0,1466	0,1104
7	3,735	549	606	3,580	3,634	0,1555	0,1010

8	4,000	555	614	3,916	4,000	0,0838	0,0000
9	2,055	517	638	2,032	2,047	0,0234	0,0077
10	2,340	517	595	2,323	2,338	0,0174	0,0022
11	3,150	517	543	3,146	3,150	0,0038	0,0000
					Total	0,5056	0,3393

Para uma melhor perceção das diferenças entre os valores otimizados, calculados e reais, na Figura 5.5 encontram-se desenhados os valores da distância obtidos em cada ponto para cada um dos casos. É possível observar que com os valores otimizados obtém-se valores da distância mais próximos do real, principalmente à medida que o íman se afasta do sensor.

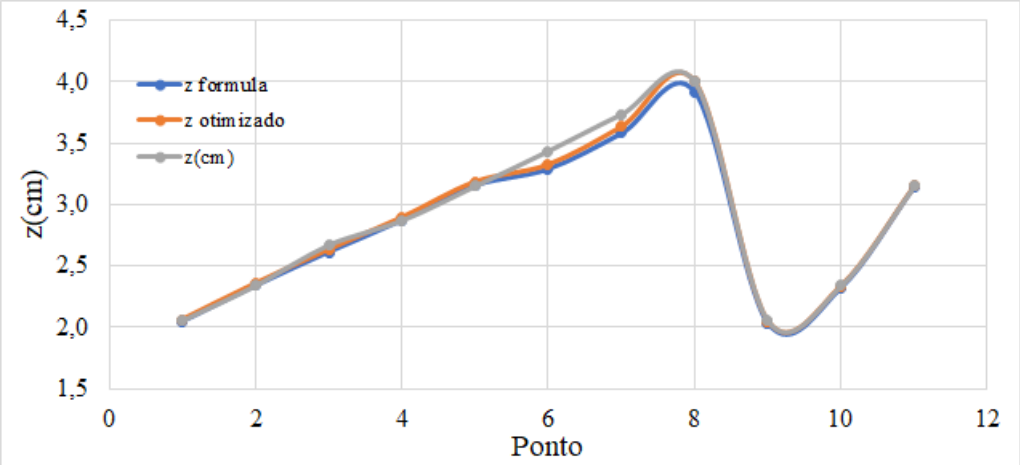


Figura 5.5: Comparação entre os valores medidos, calculados e otimizados

Os parâmetros otimizados dos sensores são os da Tabela 5.6, onde é possível verificar que estes se encontram próximos dos valores obtidos com as linearizações anteriores.

Tabela 5.6: Parâmetros otimizados dos sensores

Parâmetros	Valor	Unidades
α	507,045	ADU
β	1123,801	ADUcm^3
γ	69,474	ADU/A
m	30,084	ADU/A
b	516,343	ADU

Como o objetivo é medir a corrente e a distância, ambas as equações têm de ser reorganizadas em função do objetivo. Obteve-se assim a equação (5.33) para a corrente e a equação (5.34) para a distância.

$$i(t) = 0,03324ADC_i(t) - 17,1632 \tag{5.33}$$

$$z(t) = \sqrt[3]{\frac{1123,801}{ADC_z(t) - 507,045 - 69,474i(t)}} \tag{5.34}$$

5.3 Projeto de controladores clássicos

Para treinar uma rede neuronal é necessário dados de treino, e como explicado no Capítulo 3 o primeiro passo é o treino de uma rede que emule o comportamento do sistema. Essa é uma rede que recebe como dados de treino, valores da saída do sistema e do sinal aplicado ao sistema. Neste caso como o sistema é instável, é impossível aplicar um sinal aleatório em malha aberta ao sistema sem que ele tenda para a instabilidade, e por essa razão neste sub-capítulo são projetados controladores que permitam excitar e controlar o sistema e desse modo obter dados de treino para a rede. De notar que todos os cálculos foram realizados com o código MATLAB do Anexo C.

Começando por analisar o sistema, a sua função de transferência é dada pela equação (5.35), onde é possível ver que possuí um pólo positivo em 44,29 o que prova a instabilidade do sistema.

$$G(s) = \frac{-2,4222 \times 10^5}{(s + 200)(s + 44,29)(s - 44,29)} \tag{5.35}$$

Por forma a ser mais simples projetar os controladores, é possível simplificar o sistema realizando a aproximação das raízes dominantes. Segundo [23] e [24], os pólos do sistema que estejam posicionados mais próximo do eixo imaginário são considerados dominantes, pois a resposta transitória do sistema é caracterizada principalmente por esses pólos. Neste caso como o quociente entre o pólo mais afastado do eixo imaginário ($s = -193,3$) e o pólo mais próximo ($s = -44,29$) é maior que 4, é possível rescrever a função de transferência como na equação (5.36), tendo em atenção que é necessário manter o ganho estático do sistema.

$$G_{pd}(s) = \frac{-2,4222 \times 10^5}{200(s + 44,29)(s - 44,29)} = \frac{-1211,1}{(s + 44,29)(s - 44,29)} \tag{5.36}$$

Na Figura 5.6 encontram-se representados os lugares geométricos de raízes do sistema com todos os pólos e do sistema simplificados, ou seja apenas com os pólos dominantes. Antes de projetar o primeiro controlador é necessário discretizar o sistema para o período de amostragem utilizado, neste caso o valor escolhido foi 3 ms. Este valor foi selecionado para permitir a realização de várias leituras com o ADC do micro controlador. Ao utilizar o método *zero-order-hold* da equação

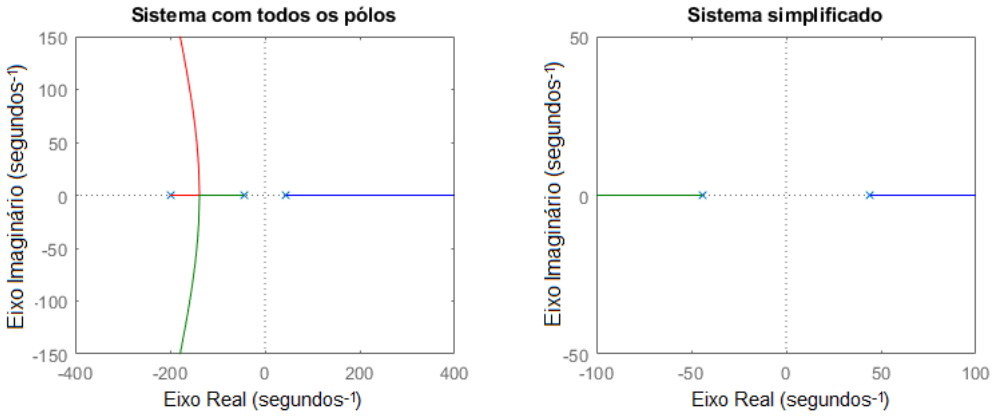


Figura 5.6: Lugar de Raízes no plano s

(5.37) obtém-se a equação (5.38) e equação (5.39) para o sistema com todos os pólos, $G(z)$ e simplificado, $G_{pd}(z)$.

$$H(z) = (1 - z^{-1})Z\left\{\frac{H(s)}{s}\right\} \tag{5.37}$$

$$G(z) = \frac{-9,4519 \times 10^{-3}(z + 3,241)(z + 0,2287)}{(z - 1,142)(z - 0,8756)(z - 0,5488)} \tag{5.38}$$

$$G_{pd}(z) = \frac{-5,458 \times 10^{-2}(z + 1)}{(z - 1,142)(z - 0,8756)} \tag{5.39}$$

Na Figura 5.7 é possível ver o lugar de raízes discreto de cada sistema, onde se verifica o pólo fora do círculo unitário.

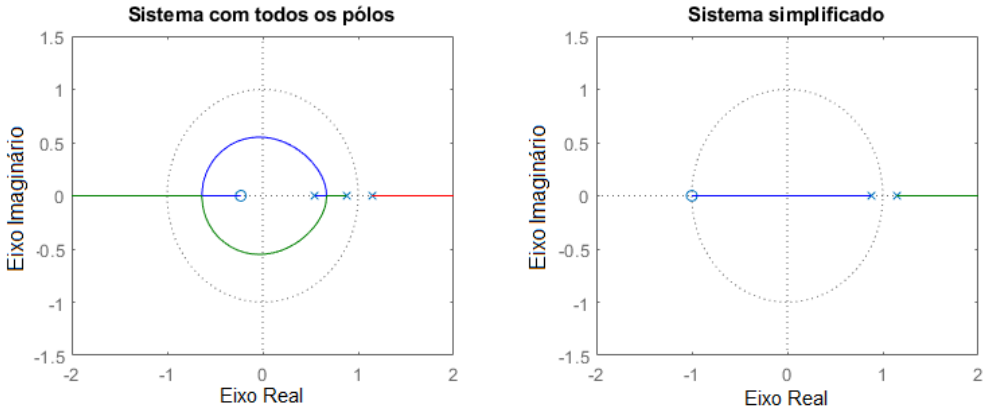


Figura 5.7: Lugar de Raízes no plano z

5.3.1 Controlador em avanço

O primeiro controlador projetado trata-se de um controlador em avanço, cuja função de transferência discreta é dada pela equação (5.40). Este melhora a estabilidade do sistema pois desloca o lugar de raízes para a esquerda [23] [24].

$$D(z) = K_{av}\frac{z-\alpha}{z-\beta},\ (\alpha > \beta) \tag{5.40}$$

Pretende-se uma resposta do sistema com um tempo de subida igual a dez vezes o período de amostragem e com menos de 5 % de *overshoot*, ou seja:

- MP=0,05;
- t_r=10×T=0,03 s.

Assim o sistema terá um coeficiente de amortecimento dada pela equação (5.41), uma frequência natural calculada com a equação (5.42) e um tempo de estabelecimento determinado pela equação (5.43).

$$\zeta = 0,6 \times (1 - MP) = 0,5700 \tag{5.41}$$

$$w_n = \frac{1,8}{t_r} = 60,0 \text{ rad/s} \tag{5.42}$$

$$t_s = \frac{4,6}{\zeta \times w_n} = 0,1345 \text{ s} \tag{5.43}$$

Considerando as caraterísticas calculadas, os pólos correspondentes do sistema no plano z são dados pela equação (5.44).

$$z_{1,2} = e^{-\zeta \cdot w_n \cdot T \pm j \cdot w_n \cdot T \cdot \sqrt{1-\zeta^2}} = 0,8926 \pm j \cdot 0,1330 \tag{5.44}$$

De modo a anular o pólo do sistema em $z = 0,8756$, o zero do controlador assume esse valor, ou seja $\alpha = 0,8756$, sendo a função de transferência do sistema em malha aberta descrita pela equação (5.45).

$$D_{av}(z)G_{pd}(z) = \frac{-5,458 \times 10^{-2} K_{av}(z+1)}{(z-1,142)(z-\beta)} \tag{5.45}$$

O pólo do controlador tem de ser calculado de modo a obedecer à condição de fase da equação (5.46) para o sistema nos pólos pretendidos [23] [24].

$$\angle D_{av}(z_1)G_{pd}(z_1) = -\pi \tag{5.46}$$

Assim é obtido o valor de 0,6806 para β, sendo agora a equação (5.47) a função de transferência do sistema com o controlador.

$$D_{av}(z)G_{pd}(z) = \frac{-5,458 \times 10^{-2} K_{av}(z+1)}{(z-1,142)(z-0,6806)} \tag{5.47}$$

Para agora calcular o ganho do controlador é utilizada a condição de módulo da equação (5.48) [23] [24], chegando assim a $K_{av} = -6,8316$.

$$|D_{av}(z_1)G_{pd}(z_1)| = 1 \tag{5.48}$$

O controlador em avanço é então descrito pela função de transferência da equação (5.49) tornando a série do controlador com o sistema na equação (5.50) para o sistema simplificado e equação (5.51) para o sistema com todos os pólos.

$$D(z) = -6,8316\frac{z-0,8756}{z-0,6806} \tag{5.49}$$

$$D_{av}(z)G_{pd}(z) = \frac{0,03729z + 0,03729}{z^2 - 1,823z + 0,7772} \tag{5.50}$$

$$D_{av}(z)G(z) = \frac{0,006457z^2 + 0,0224z + 0,004786}{z^3 - 2,371z^2 + 1,777z - 0,4266} \tag{5.51}$$

Visualizando o lugar geométrico de raízes dos dois sistemas na Figura 5.8, é possível ver que no sistema simplificado os pólos desejados são atingidos, e no sistema com todos os pólos o comportamento do sistema será próximo do objetivo.

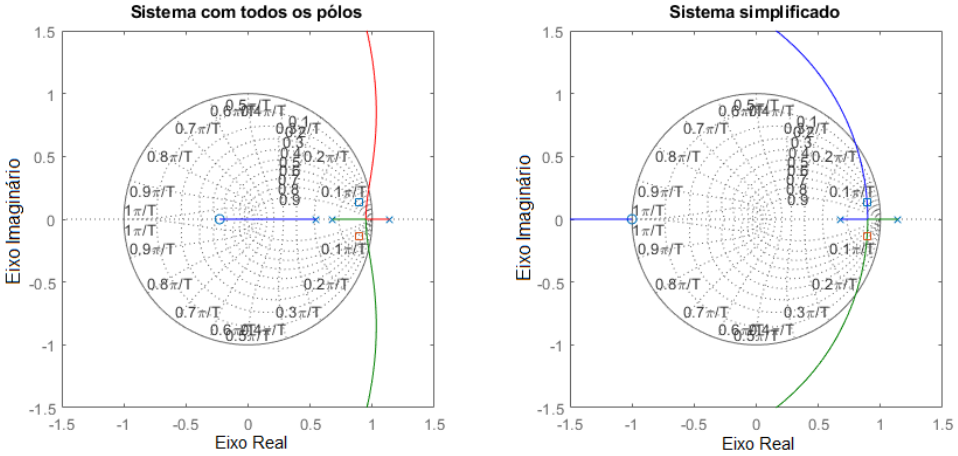


Figura 5.8: Lugar de Raízes dos sistemas com controlador em avanço

Ao aplicar a cada sistema um degrau unitário, na Figura 5.9 é visível que o regime transitório de cada sistema é diferente, sendo o sistema com todos os

pólos mais oscilatório. No entanto ambos os sistema tendem para o mesmo valor em regime permanente, apresentando um erro em regime permanente de cerca de 1,55 cm.

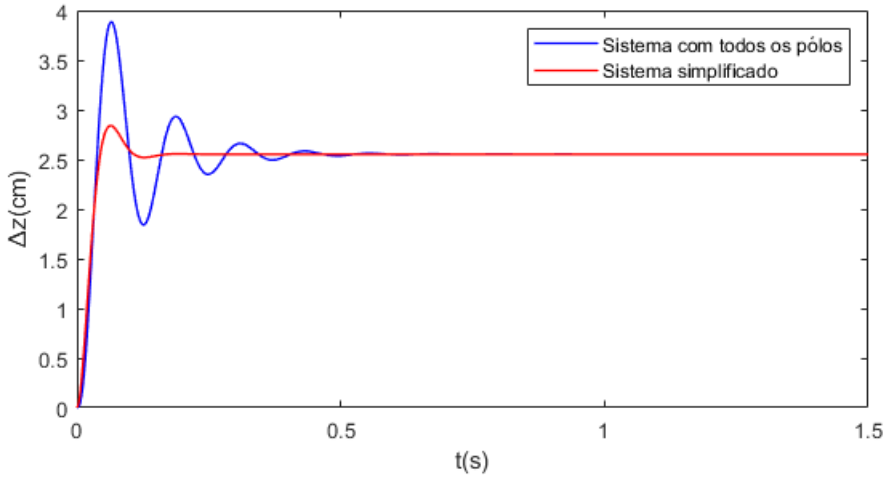


Figura 5.9: Resposta a um degrau unitário dos sistemas com controlador em avanço

5.3.2 Controlador em avanço-atraso

De modo a reduzir ou até eliminar o erro em regime permanente é possível adicionar um controlador em atraso (equação (5.52)) em série com o controlador em avanço projetado anteriormente. Deste modo é possível manter a estabilidade do sistema e ao mesmo tempo diminuir o erro em regime permanente. No entanto para manter a resposta transitória do sistema o pólo e zero do controlador têm de estar muito próximos do eixo imaginário em $s = 0$.

$$D_{at}(s) = K_{at}\frac{s-\alpha}{s-\beta},\ (\alpha > \beta) \tag{5.52}$$

Começando por converter o controlador em avanço para o domínio contínuo, obtém-se a função de transferência de equação (5.53). Assim a função de transferência do controlador em avanço em série com o sistema é a equação (5.54) para o sistema simplificado e equação (5.55) para o sistema com todos os pólos.

$$D_{av}(s) = 0,5\frac{(s+49,96)}{(s+128,3)} \tag{5.53}$$

$$D_{av}(s)G_{pd}(s) = \frac{8273,7s+4,133\times10^5}{s^3+128,3s^2-1962s-2,516\times10^5} \tag{5.54}$$

$$D_{av}(s)G(s) = \frac{1,6548 \times 10^6 s + 8,267 \times 10^7}{s^4 + 328,3s^3 + 2,369 \times 10^4 s^2 - 6,441 \times 10^5 - 5,034 \times 10^7} \tag{5.55}$$

O erro em regime permanente para um degrau unitário como entrada do sistema com ambos os controladores é obtido com a equação (5.56) [23], onde ao calcular o limite obtém-se a equação (5.57), para um ganho unitário no controlador em atraso ($K_{at} = 1$).

$$E_{ss} = \lim_{s \to 0} \frac{s}{1 + D_{at}(s)D_{av}(s)G(s)} \frac{1}{s} \tag{5.56}$$

$$E_{ss} = \frac{DG_{den}(0) \times \beta}{DG_{den}(0) \times \beta + DG_{num}(0) \times \alpha} \tag{5.57}$$

onde DG_{den} é o denominador e DG_{num} o numerador do sistema com controlador em avanço.

Como o objetivo do controlador em atraso é reduzir o erro mantendo a resposta transitória do sistema, o ganho e raízes do controlador em atraso tem de causar o mínimo impacto possível no lugar de raízes, daí o ganho K_{at} ser 1. Para obter o pólo e zero do controlador é possível reorganizar a fórmula anterior para obter a equação (5.58) do rácio entre os dois valores em função do erro pretendido [25].

$$\frac{\alpha}{\beta} = \frac{DG_{den}(0) - E_{ss} \times DG_{den}(0)}{E_{ss} \times DG_{num}(0)} \tag{5.58}$$

Neste caso, para um erro de -0,05 cm é obtido a equação (5.59).

$$\frac{\alpha}{\beta} = \frac{-5,034 \times 10^7 + 0,05 \times -5,034 \times 10^7}{-0,05 \times 8,267 \times 10^7} = 12,7855 \tag{5.59}$$

O erro negativo serve para evitar que o disco oscile acima num valor menor que a referência, pois isso significa que está mais próximo do eletroíman o que pode causar com que o disco sofra uma força de atração entre o seu campo magnético e o próprio material do eletroíman.

Para este controlador em atraso não afetar a resposta transitória do sistema o seu impacto na condição de fase tem de ser mínimo, ou seja o zero e pólo têm de estar próximos um do outro visto que afetam de maneira oposta a fase do sistema. Como explicado em [25], o zero do controlador em atraso deve ser menor que a parte real dos pólos dominantes do sistema, e o quociente entre eles de 50 a 100.

Os pólos do sistema são os equivalentes contínuos dos pólos discretos calculados anteriormente com a equação (5.44). A equação (5.60) é utilizada para obter os pólos contínuos, onde os parâmetros do sistema são os mesmos.

$$s_{1,2} = -\zeta \cdot w_n \pm j \cdot w_n \cdot \sqrt{1-\zeta^2} = -34,2 \pm j \cdot 49,2987 \tag{5.60}$$

Assim com a equação (5.61) é obtido o valor do zero e através da equação (5.62) é obtido o pólo do controlador em atraso.

$$\alpha = \frac{Re\{s_1\}}{50} = \frac{-34,2}{50} = -0,6840 \tag{5.61}$$

$$\beta = \frac{-0,6840}{12,7855} = -0,0535 \tag{5.62}$$

Com estes valores, o controlador em atraso é dado pela função de transferência contínua da equação (5.63), o que é aproximadamente um controlador integral pois o pólo encontra-se quase em $s = 0$.

$$D_{at}(s) = \frac{s+0,6840}{s+0,0535} \tag{5.63}$$

Ao discretizar o controlador para o período de amostragem de 3 ms através do método *zero-order-hold* é obtido o controlador integral da equação (5.64).

$$D_{at}(z) = \frac{z-0,9979}{z-1} \tag{5.64}$$

Assim a função de transferência do controlador avanço-atraso é dada pela equação (5.65), sendo a equação (5.66) a função de transferência de malha aberta do sistema com o controlador.

$$D_{av-at}(z) = \frac{-6,823z^2+12,8z-5,969}{z^2-1,68z+0,6804} \tag{5.65}$$

$$D_{av-at}(z)G(z) = \frac{0,006457z^3+0,01596z^2-0,01757z-0,004776}{z^4-3,371z^3+4,149z^2-2,204z+0,4265} \tag{5.66}$$

Como é aferido na Figura 5.10 do lugar geométrico de raízes de ambos os sistemas (simplificado e com todos os pólos), com o controlador avanço-atraso ainda são cumpridas as especificações utilizadas para projetar o controlador em avanço. Isto é verificado pelo facto do lugar geométrico de raízes sobrepor-se aos pólos discretos correspondentes às especificações.

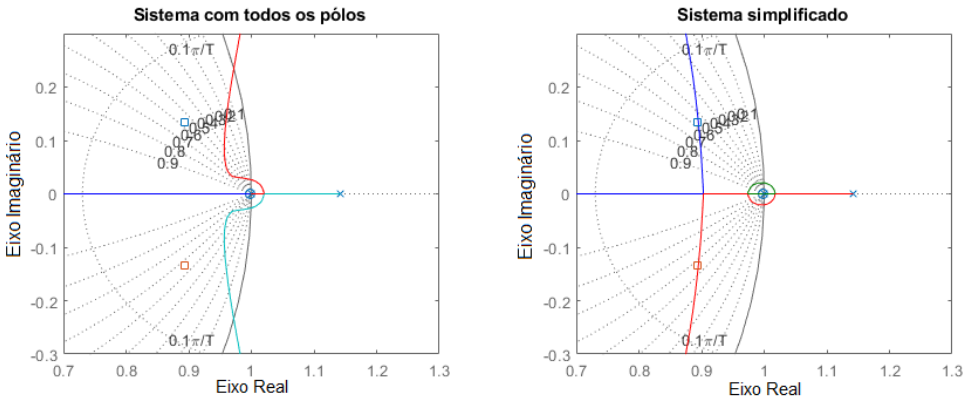


Figura 5.10: Lugar de Raízes dos sistemas com controlador em avanço-atraso

Ao aplicar um degrau unitário, é possível comparar a resposta do sistema controlado com apenas o controlador em avanço e com o controlador avanço-atraso. Na Figura 5.11 encontram-se representadas ambas as respostas, onde é possível ver que ao adicionar o controlador em atraso não foi afetada a resposta inicial do sistema, apresentando o mesmo *overshoot*. No entanto, após a primeira oscilação o sistema com o controlador avanço-atraso começa lentamente a eliminar o erro do sistema tendendo para o valor de referência.

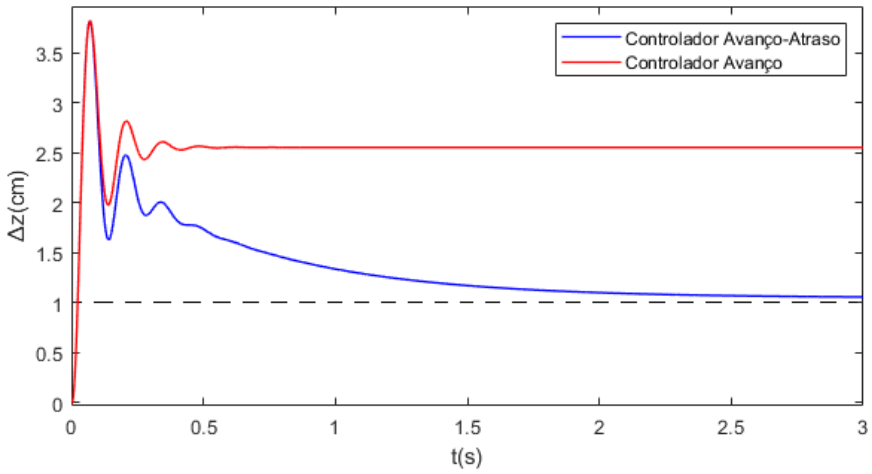


Figura 5.11: Comparação dos controladores a um degrau unitário

5.4 Testes dos controladores clássicos

Até agora os controladores projetados, foram simulados a controlar o sistema linearizado em torno do ponto de funcionamento, no entanto o sistema real é não linear. Por isso é necessário realizar as simulações com o sistema não linear

de modo a comprovar o funcionamento adequado dos controladores. Para esse fim foi utilizado o diagrama Simulink da Figura 5.12, onde é possível ver que ao sinal de controlo proveniente do controlador é adicionado um constante. Essa constante corresponde ao valor da tensão de equilíbrio calculada com a equação (5.18). É também possível averiguar que o sistema magnético usado é não linear.

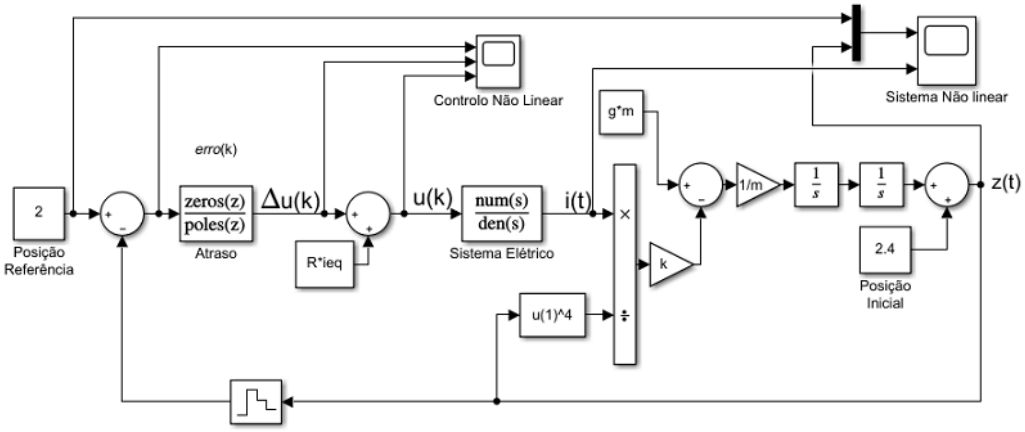


Figura 5.12: Diagrama Simulink do sistema não linear

Com este diagrama foram testados ambos os controladores. Na Figura 5.13 encontra-se a resposta do sistema com o controlador em avanço e na Figura 5.14 a do controlador em avanço-atraso.

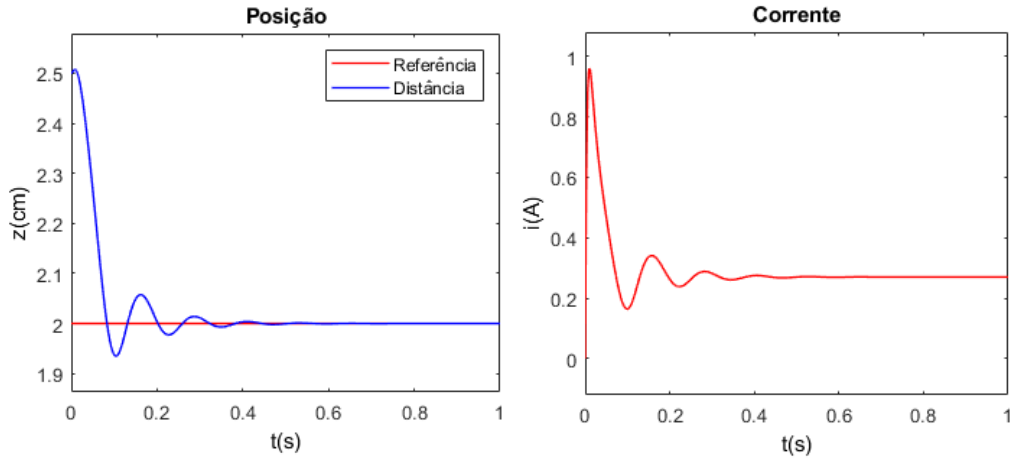


Figura 5.13: Resposta do sistema não linear com o controlador em avanço

Analisando os resultados, conclui-se que ambos os controladores têm um bom desempenho pois conseguem colocar o sistema na posição de referência, sendo que normalmente um controlador em avanço apresenta um erro em regime permanente. De notar que também a corrente no sistema nunca ultrapassa o limite de 1,5 A e se mantém constante em regime permanente.

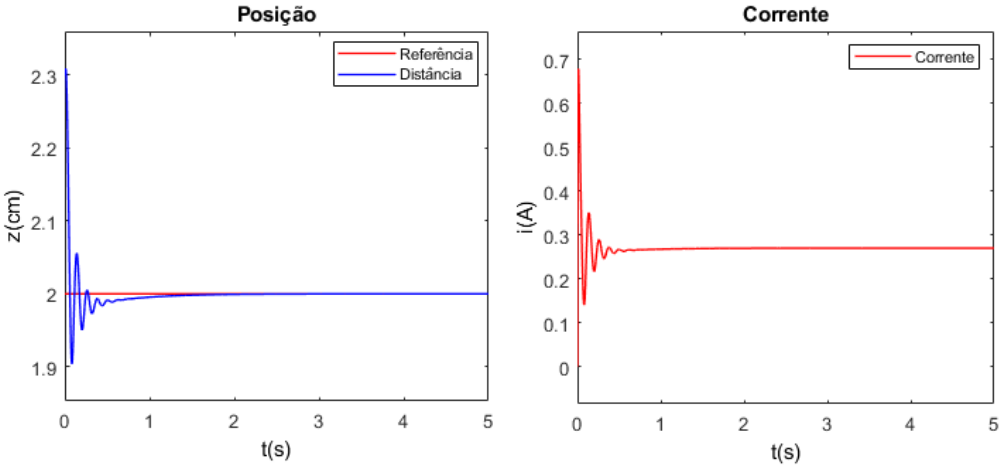


Figura 5.14: Resposta do sistema não linear com o controlador em avanço-atraso

O último teste a realizar é com o sistema real, para isso é utilizado o diagrama Simulink em conjunto com o "Add-on" *Simulink Support Package for Arduino Hardware* para controlar o Arduino Mega 2560. Este "Add-on" permite testar os controladores projetados facilmente antes da sua implementação no micro-controlador. Como é possível aferir o pino 54 do Arduino realiza a medição do sensor de Hall de posição enquanto que o pino 55 mede o sensor de corrente, o pino 11 é utilizado para aplicar o sinal PWM na porta do MOSFET.

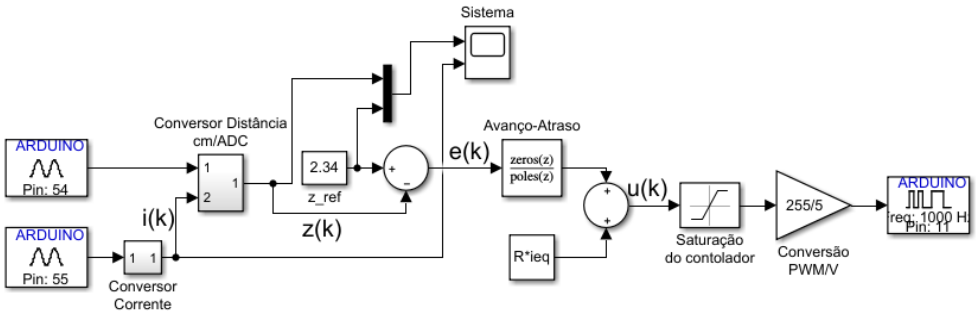


Figura 5.15: Diagrama Simulink utilizado para testar sistema

Neste diagrama Simulink, o bloco "Conversor Distância" e "Conversor Corrente" aplicam a equação (5.34) e equação (5.33), respetivamente.

Tal como anteriormente, foi testado em primeiro lugar o controlador em avanço e de seguida o controlador em avanço-atraso, estando as respostas do sistema na Figura 5.16 e Figura 5.17, respetivamente. De notar que a distância de referência é 2,34 cm para permitir que o sistema oscile sem que o disco seja atraído pelo material do eletroíman. Ambos os controladores foram testados com o disco numa posição inicial de 3,1 cm.

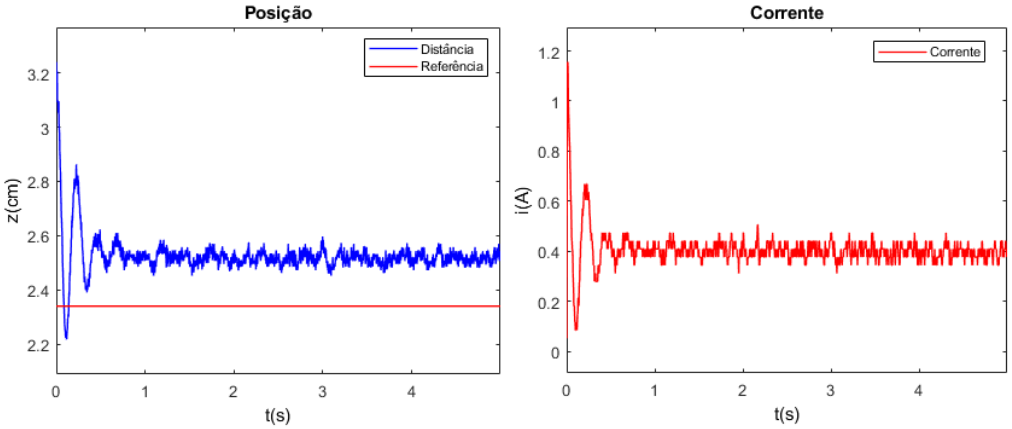


Figura 5.16: Resposta do sistema real com o controlador em avanço

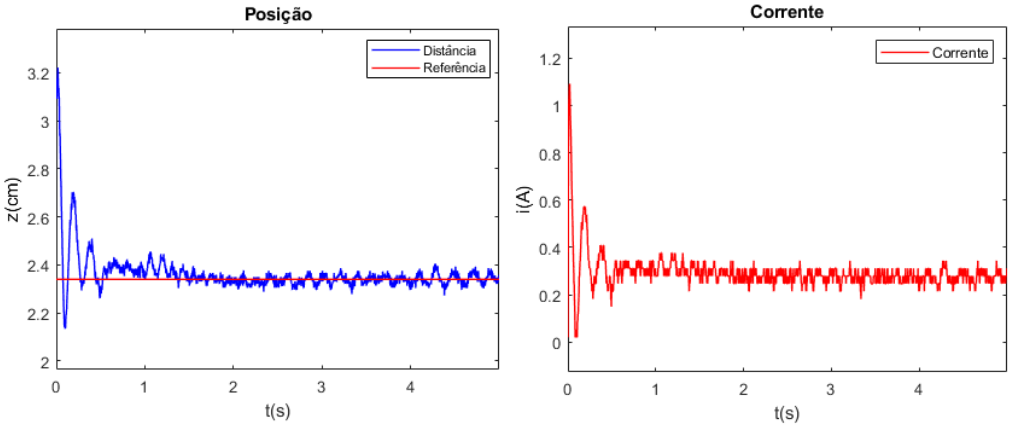


Figura 5.17: Resposta do sistema real com o controlador em avanço-atraso

Como é possível observar, ambos os controlador conseguem obter a resposta prevista, onde a diferença entre respostas é a existência de um erro em regime permanente, sendo que o controlador em avanço-atraso consegue manter o sistema na posição de referência. As principais diferenças entre estas respostas e as simuladas, é a presença de ruído nas medições, o que era de esperar. Nota-se a semelhança na resposta dos controladores no sistema real, ambos atingem o regime estacionário em aproximadamente o mesmo tempo pois possuem um regime transitório análogo.

Com estes controladores operacionais é agora possível aplicar sinais mais apropriados ao sistema, de modo a obter dados de treino para as redes neuronais necessárias para o controlo.

Capítulo 6

Identificação e controlo neuronal

Neste capítulo são utilizados os conceitos de redes neuronais analisados anteriormente para criar e treinar algumas redes neuronais que realizem a identificação do sistema de levitação magnético, emulando assim o seu comportamento. De seguida serão desenvolvidos alguns controladores constituídos por redes neuronais e testado o seu funcionamento. Deste modo será analisado o desempenho de cada um e comparado entre eles.

6.1 Identificação do sistema eletromagnético

Como mencionado no Capítulo 3, existem duas etapas no controlo neuronal de um sistema, sendo a primeira etapa a identificação do sistema a controlar. Esta implica treinar uma rede neuronal que modele o sistema e a resposta que este teria a qualquer sinal de controlo.

Para esse fim, é primeiro necessário obter dados de treino adequados, ou seja, aplicar ao sistema real um sinal de controlo que varie aleatoriamente, com diferentes amplitudes e frequências [14]. Deste modo será possível obter amostras do sinal de controlo aplicado e da distância do sistema ao longo do tempo.

Com o objetivo de obter os dados de treino, foi criado o sinal de referência da Figura 6.1, onde entre os 0 e 15 segundos e entre os 30 e 40 segundos é também adicionado um sinal de ruído de alta frequência com distribuição Gaussiana de média 0 e variância 0,001. Neste caso é criado um sinal de referência aleatório em vez de um sinal de controlo, pois como o sistema é altamente instável, sem o controlador era extremamente difícil criar um sinal de controlo aleatório que conseguisse estimular o sistema e ao mesmo tempo resultar numa resposta estável.

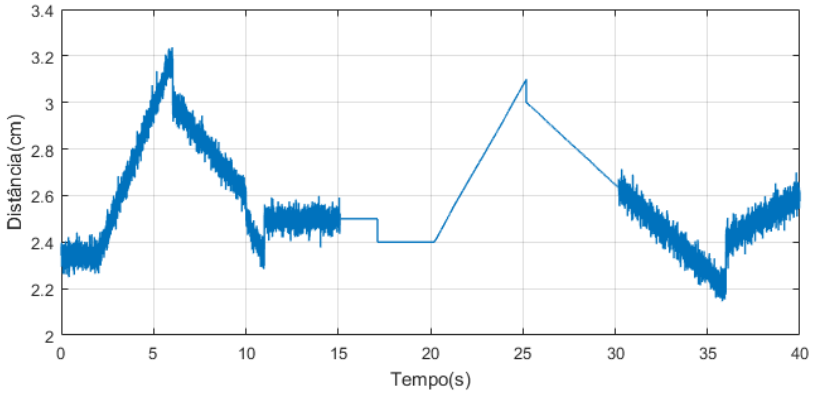


Figura 6.1: Sinal de referência para obter os dados de treino

Utilizando o controlador em Avanço-Atraso da equação (5.65) é realizado o controlo do sistema para o sinal de referência criado, obtendo a resposta da Figura 6.2 e o sinal de controlo da Figura 6.3.

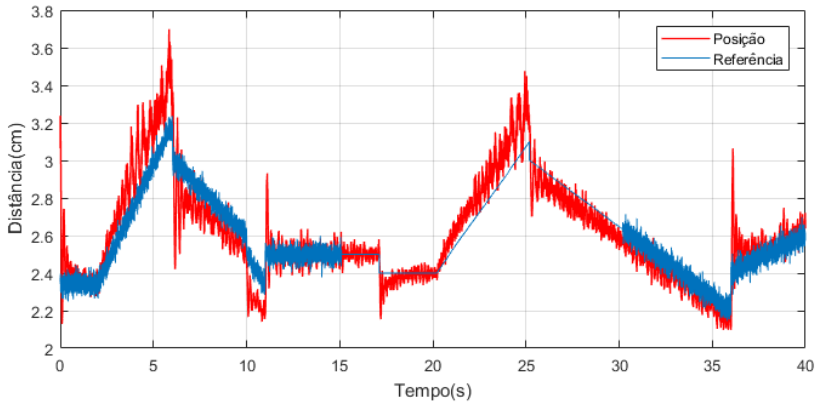


Figura 6.2: Posição do sistema ao longo do tempo para os dados de treino

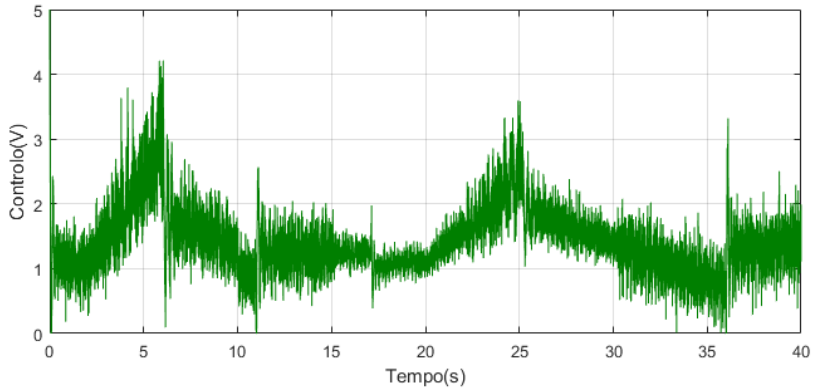


Figura 6.3: Sinal de controlo aplicado ao sistema para os dados de treino

Para então proceder à identificação do sistema é utilizado o código MATLAB do Anexo D, onde é criada e treinada uma rede NARX. A escolha deste tipo de rede advém do facto do sistema ser altamente instável e não linear. Assim com este tipo de arquitetura de rede serão tidos em conta os instantes anteriores da saída do sistema e do sinal controlo, *feedback delay* e *input delay*, respetivamente. No código é possível escolher quantos *delays* se pretende e o número de neurónios na camada escondida, possibilitando assim que sejam criadas, treinadas e simuladas diversas redes de identificação.

De notar que o algoritmo de treino utilizado é a retro-propagação de Levenberg-Marquardt (`trainlm`), para evitar que o treino fique preso em mínimos locais para o conjunto de dados utilizados, onde a variação dos pesos da rede numa determinada iteração é dada pela equação (6.1) [26].

$$\Delta \mathbf{W} = -\left[\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}\right]^{-1}\mathbf{J}^T e \tag{6.1}$$

onde e é o erro da rede e $\mathbf{J}$ é a matriz Jacobiana dos erros da rede em relação aos pesos. O parâmetro μ é inicializado com um valor pequeno e é aumentado com um fator de 10 se o desempenho da rede não melhorar em qualquer iteração. No caso do desempenho da rede melhorar, μ diminui por um fator de 10. E se μ atingir um valor demasiado elevado, o algoritmo termina, pois indica que o desempenho já não melhora [26].

Assim são criadas e treinadas redes para diferentes combinações de número de neurónios (n) na camada escondida e diferentes números de *delays* (d) do sinal de controlo e saída do sistema. O resultado do treino de cada rede pode ser observado na Tabela 6.1, onde por exemplo a rede 10d15n, da Figura 6.4, possui 10 *delays* e 15 neurónios na camada escondida.

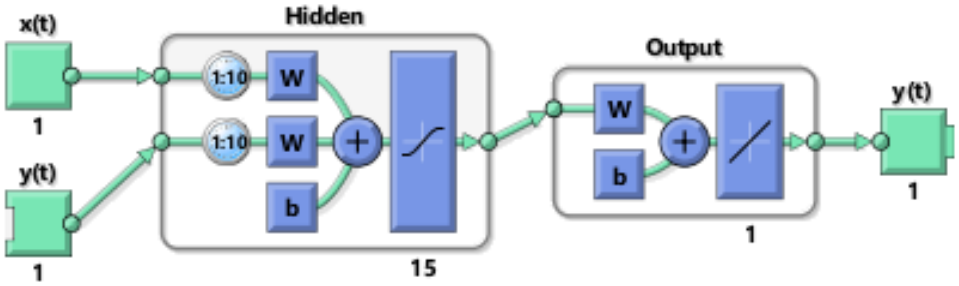


Figura 6.4: Rede de identificação 10d15n

Como se verifica, as redes com 10 ou 15 *delays* possuem um melhor MSE após o treino, mesmo com menos neurónios na camada escondida.

Tabela 6.1: MSE das redes para identificação após o treino

Rede	MSE treino	Rede	MSE treino
3d5n	8,8604E-04	10d5n	8,0709E-04
3d10n	8,7738E-04	10d10n	7,8236E-04
3d15n	8,6531E-04	10d15n	7,7818E-04
5d5n	8,7772E-04	15d5n	7,5261E-04
5d10n	8,3748E-04	15d10n	7,3700E-04
5d15n	8,1346E-04	15d15n	7,0542E-04

De modo a visualizar melhor as diferenças entre as redes, é utilizado o diagrama Simulink da Figura 6.5 para simular as redes de identificação do sistema com o controlador em Avanço-Atraso.

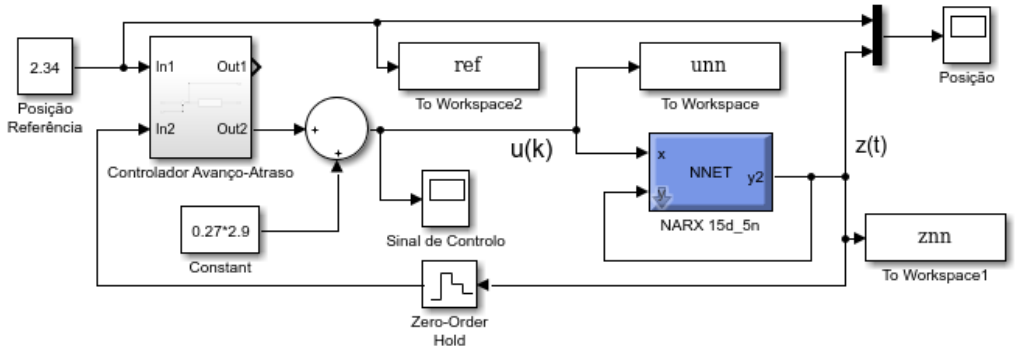


Figura 6.5: Diagrama Simulink para simular as redes de identificação

Na Figura 6.6 é possível analisar a resposta das redes com melhor desempenho para a entrada de referência utilizada para obter os dados de treino, e na Figura 6.7 os sinais de controlo aplicados a cada uma das redes.

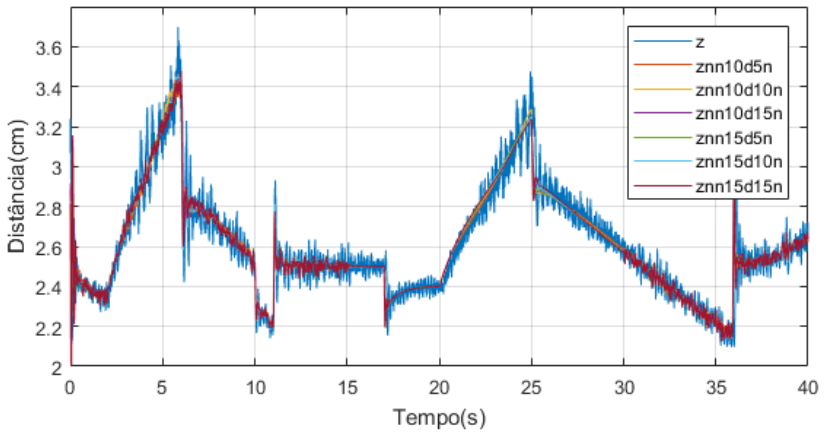


Figura 6.6: Respostas das redes de identificação para entrada de treino

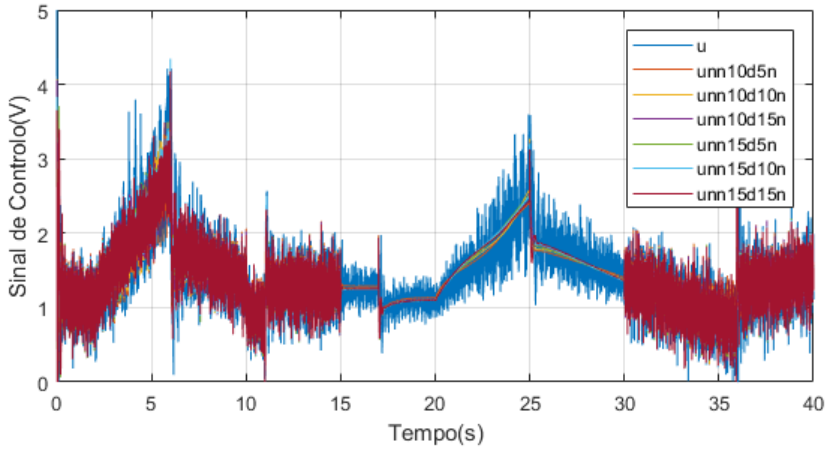


Figura 6.7: Sinal de controlo aplicado nas redes de identificação para entrada de treino

Como é possível observar as redes têm um desempenho satisfatório, possuindo respostas com menos ruído em relação ao sistema. O que é ideal, sendo que assim as redes aprenderam a emular o comportamento do sistema, ignorando o ruído, caso contrário podiam apresentar uma resposta instável.

Com o fim de testar a capacidade de emulação das redes para sinais distintos dos dados de treino, foi aplicado ao sistema real com o controlador em Avanço-Atraso os sinais de referência em degrau, sinusoidal e em serra. As respostas obtidas encontram-se na Figura 6.8.

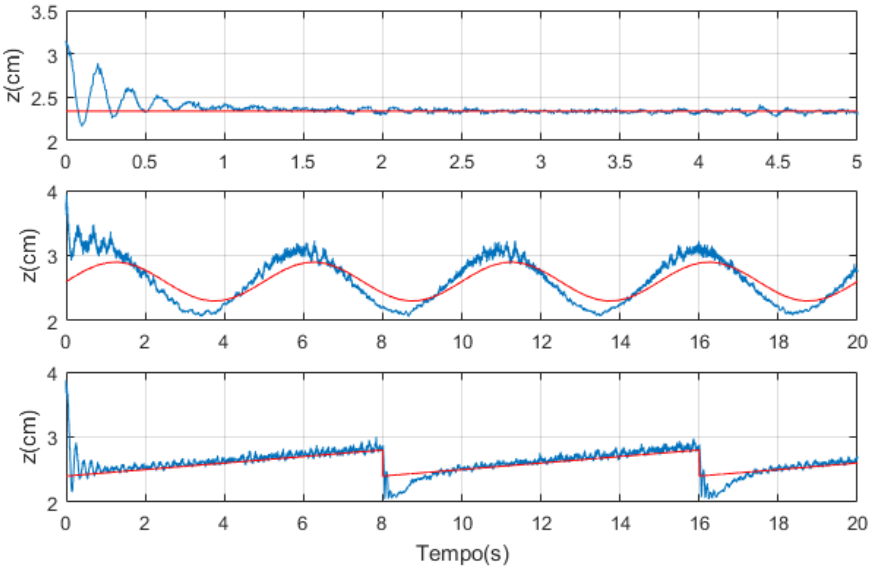


Figura 6.8: Resposta do sistema a entrada em degrau, sinusoidal e serra com controlador em Avanço-Atraso

De seguida utilizando os valores obtidos da distância do sistema e sinal de controlo aplicado para os três sinais de referência, simulam-se as redes neuronais de identificação treinadas de modo a analisar o seu desempenho. Na Tabela 6.2 encontram-se os valores dos MSE calculados com o código MATLAB do Anexo D para as redes com 10 e 15 *delays* para os três sinais de referência. Além disso, no Anexo E encontram-se representadas as respostas das redes obtidas com o diagrama Simulink da Figura 6.5.

Tabela 6.2: MSE das redes de identificação para diferentes sinais de referência

Rede	MSE degrau	MSE sinusóide	MSE serra	Total
10d5n	2,5969E-04	9,0931E-04	6,6006E-04	1,8291E-03
10d10n	2,5293E-04	8,7981E-04	6,6029E-04	1,7930E-03
10d15n	2,5085E-04	8,8896E-04	6,6100E-04	1,8008E-03
15d5n	2,4060E-04	8,7159E-04	6,9810E-04	1,8103E-03
15d10n	2,4077E-04	8,9053E-04	6,9631E-04	1,8276E-03
15d15n	2,4030E-04	8,9365E-04	6,9988E-04	1,8338E-03

Analisando os valores da tabela, é possível aferir que as redes com 10 *delays* possuem um melhor desempenho nos sinais de referência sinusoidal e serra, tendo as redes com 15 *delays* um melhor desempenho na referência em degrau. Sendo assim, a rede 10d10n é a que apresenta o melhor desempenho. Na Figura 6.9 encontra-se representada a resposta desta rede a azul e as três referências a vermelho.

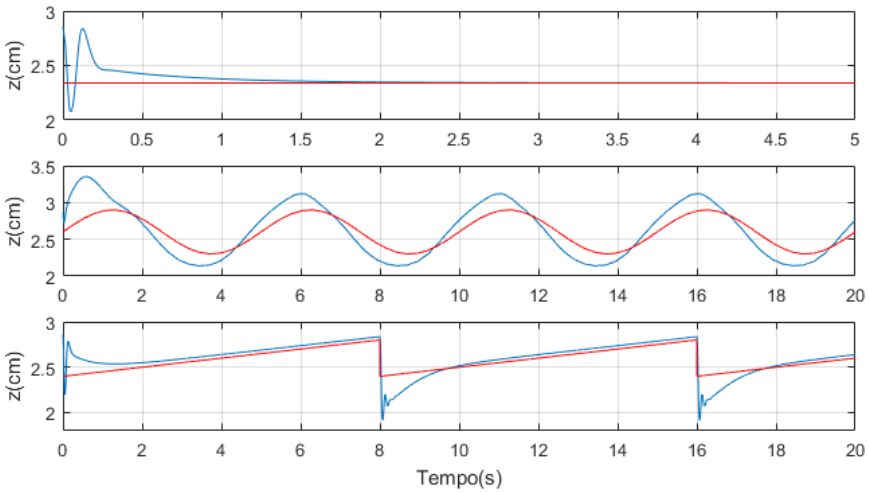


Figura 6.9: Resposta da rede 10d10n para a entrada em degrau, sinusoidal e serra com controlador em Avanço-Atraso

6.2 Controlo neuronal

Com a rede de identificação treinada e testada, procede-se ao desenvolvimento de controladores neuronais. Para isso utilizar-se-á como dados de treino, as mesmas amostras do sinal de controlo, referência e resposta do sistema controlado com o controlador em Avanço-Atraso para o sinal de referência aleatório.

Existem vários controladores neuronais, sendo o mais simples o modelo inverso direto, que é uma rede que descreve a dinâmica inversa do sistema. Tal como justificado na identificação do sistema, este trata-se de um sistema não linear e instável. Assim também na criação e treino da rede do modelo inverso serão utilizadas várias amostras do sinal de controlo e posição do sistema em instantes anteriores, sendo a função do controlador dada pela equação 3.23.

6.2.1 Modelo inverso

Através do código MATLAB desenvolvido do Anexo D, são criadas algumas redes e treinadas de modo a aprenderem a dinâmica inversa do sistema. Inicialmente realiza-se um pré-processamento dos dados, proporcionando todos os valores entre 0 e 1. De seguida, criam-se redes de duas camadas, cuja primeira camada possuí um número variável de neurónios com função de ativação sigmóide e a segunda camada é constituída por um neurónio com função de ativação ReLU. A escolha do ReLU advém do controlador não poder fornecer sinais de tensão negativos.

Estas redes de duas camadas variam entre si no número de neurónios e *delays* do sinal de controlo e posição do sistema. Por exemplo a rede 5s10d é constituída por cinco neurónios na primeira camada e 10 amostras do sinal de controlo e distância do sistema, como representado na Figura 6.10, onde se observa que a rede possuí 21 entradas, 20 provenientes dos *delays* e o restante com o valor do sinal de referência.

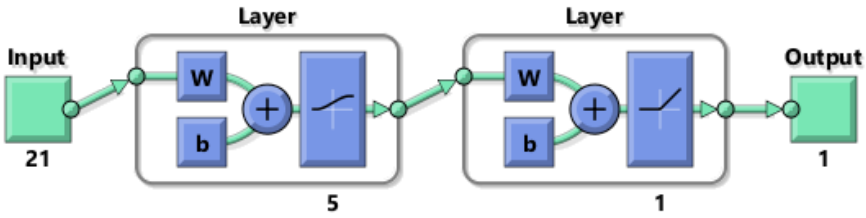


Figura 6.10: Rede de controlo 5s10d

O algoritmo de treino utilizado é novamente a retro-propagação de Levenberg-Marquardt (`trainlm`). Assim são treinadas redes com 5, 10 e 15 neurónios na primeira camada e com 5 e 10 *delays*. O MSE de treino para essas redes encontra-se na Tabela 6.3, onde como é possível verificar, as redes com 5 *delays* possuem um MSE mais baixo que as redes com 10 *delays*.

Tabela 6.3: MSE das redes de modelo inverso de duas camadas após o treino

Rede	MSE treino
5s5d	8,0172E-08
10s5d	2,6406E-07
15s5d	4,5901E-07
5s10d	5,3251E-06
10s10d	1,2461E-02
15s10d	2,1370E-04

De modo a confirmar o resultado do treino, foram testadas as redes de modelo inverso de duas camadas para as referências em degrau, sinusoidal e em serra. Na Tabela 6.4 é possível verificar o desempenho de cada rede para cada referência, bem como o desempenho em geral.

Tabela 6.4: MSE das redes de modelo inverso de duas camadas para diferentes sinais de referência

Rede	MSE degrau	MSE sinusóide	MSE serra	Total
5s5d	2,8573E-05	2,3002E-04	3,4357E-04	6,0217E-04
10s5d	3,1472E-05	3,2420E-04	4,9581E-04	8,5148E-04
15s5d	3,6304E-05	6,7784E-04	7,6431E-04	1,4785E-03
5s10d	1,2165E-04	3,9774E-04	5,7153E-04	1,0909E-03
10s10d	1,5032E-04	5,1957E-04	1,1914E-03	1,8613E-03
15s10d	1,3643E-04	1,0770E-03	2,1435E-03	3,3569E-03

Como era de esperar pelos resultados obtidos no treino, a rede 5s5d possuí o melhor desempenho de todas as redes desenvolvidas. Para visualizar o resultado da rede 5s5d a controlar o sistema é utilizado o diagrama Simulink da Figura 6.11. Com este é simulada a rede para os sinais de referência em degrau, sinusoidal e em serra, obtendo as respostas do sistema da Figura 6.12, onde a azul encontra-se representada a referência fornecida e a vermelho a saída do sistema, e na Figura 6.13 estão representados os respetivos sinais de controlo.

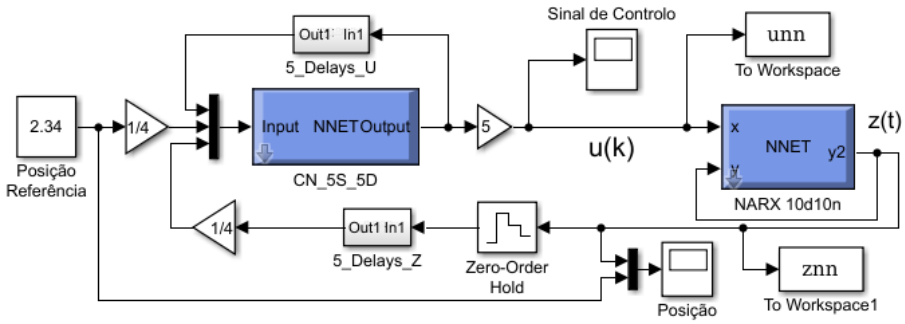


Figura 6.11: Diagrama Simulink para simular a rede 5s5d de modelo inverso

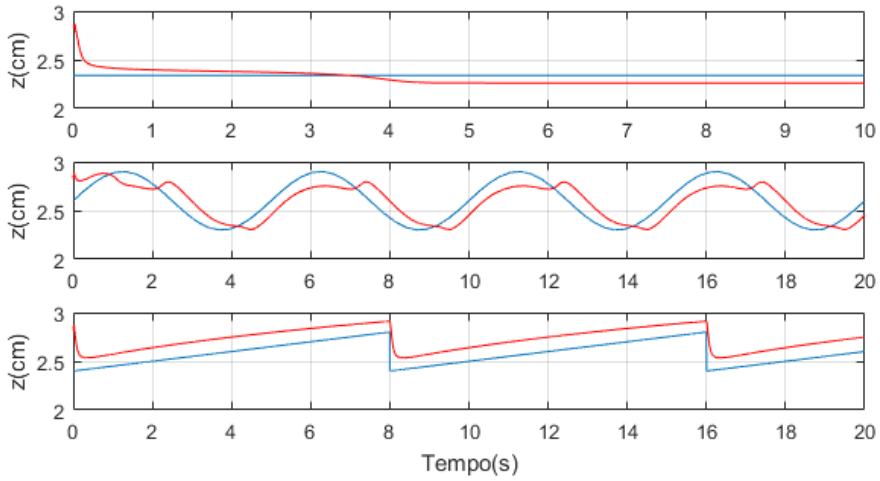


Figura 6.12: Respostas do sistema controlado pela rede 5s5d para as diferentes entradas

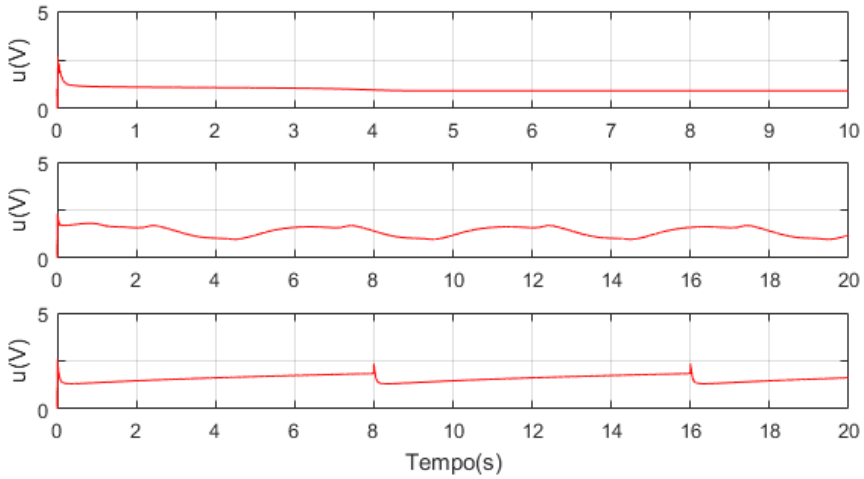


Figura 6.13: Sinal de controlo fornecido pela rede 5s5d para as diferentes entradas

É possível verificar que a rede consegue controlar o sistema de forma satisfatória, embora possua um ligeiro erro em regime permanente e uma distorção no controlo da referência sinusoidal.

De modo a tentar melhorar estas imperfeições, foram criadas e treinadas redes de três camadas, onde a primeira camada é constituída por um neurónio com função de ativação tangente hiperbólica (1th) e as duas camadas seguintes são iguais às redes de modelo inverso de duas camadas desenvolvidas, como por exemplo a rede 1th5s5d da Figura 6.14. Para esse fim é utilizado o código MATLAB do Anexo D, sendo realizado o mesmo pré-processamento e utilizado o mesmo algoritmo de treino que nos casos anteriores.

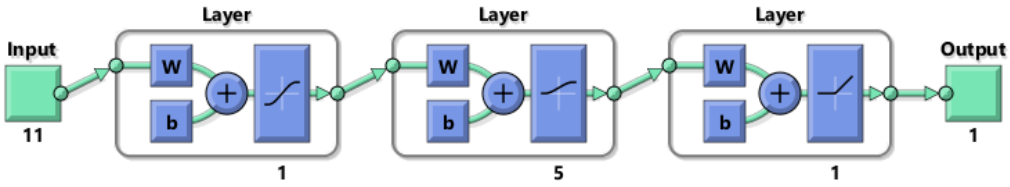


Figura 6.14: Rede de controlo 1th5s5d

A Tabela 6.5 apresenta o MSE obtido para cada rede para o treino com o sinal aleatório.

Tabela 6.5: MSE das redes de modelo inverso de três camadas após o treino

Rede	MSE treino
1th5s5d	1,3623E-04
1th10s5d	4,0668E-06
1th15s5d	1,4950E-05
1th5s10d	4,5508E-05
1th10s10d	9,1546E-09
1th15s10d	2,1092E-05

Ao analisar os valores obtidos, é possível verificar que as redes com 10 amostras do sinal de controlo e distância do sistema (*delays*) possuem melhores desempenhos, sendo de destacar a rede 1th10s10d com um MSE muito menor que as outras redes. De modo a confirmar estes resultados é utilizado o código MATLAB do Anexo D para calcular o desempenho das redes para as referências em degrau, sinusoidal e em serra. Na Tabela 6.6 encontram-se os valores de MSE obtidos.

Tabela 6.6: MSE das redes de modelo inverso de três camadas para diferentes sinais de referência

Rede	MSE degrau	MSE sinusóide	MSE serra	Total
1th5s5d	2,1019E-05	1,5618E-04	3,0189E-04	4,7908E-04
1th10s5d	2,0237E-05	1,5752E-04	3,0275E-04	4,8050E-04
1th15s5d	2,7159E-05	1,5577E-04	3,0127E-04	4,8420E-04
1th5s10d	9,8642E-05	2,6887E-04	5,2893E-04	8,9644E-04
1th10s10d	9,8300E-05	2,7111E-04	5,2967E-04	8,9908E-04
1th15s10d	1,9305E-02	2,8199E-04	2,5666E-02	4,5253E-02

Contrariamente ao que se obteve no treino, ao testar com as diferentes entradas, são obtidos melhores desempenhos nas redes de 5 *delays*, o que leva a concluir que os baixos valores de MSE das redes de 10 *delays* para os dados de treino eram sintomas de redes sobre-ajustadas. Deste modo a rede com melhor desempenho é a rede 1th5s5d, pois embora possua um desempenho muito pró-

ximo das outras duas redes com 5 *delays*, é a rede mais pequena, sendo que requer menos recursos computacionais. De notar que tanto nas redes de modelo inverso de duas camadas como nas de três camadas, as redes mais pequenas obtiveram melhores resultados.

Utilizando o diagrama Simulink da Figura 6.11 é simulada a rede 1th5s5d para as referências em degrau, sinusoidal e em serra, obtendo a Figura 6.15 e Figura 6.16, para as respostas do sistema e sinais de controlo, respetivamente.

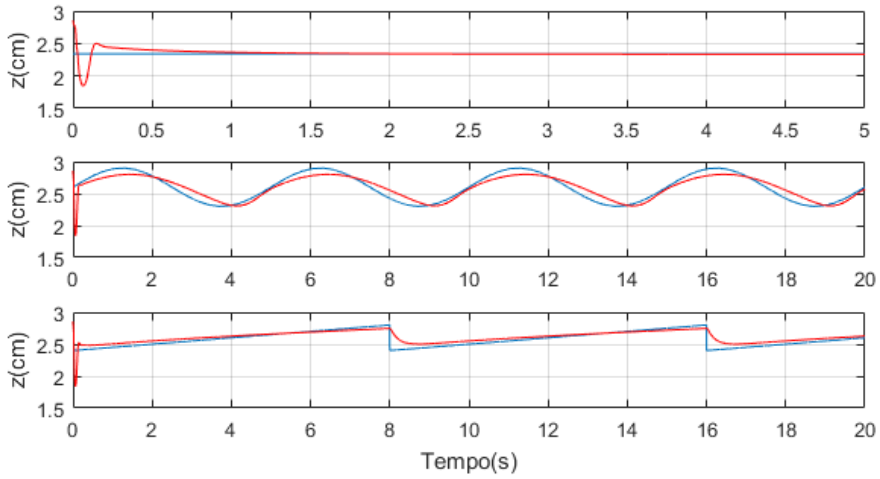


Figura 6.15: Respostas do sistema controlado pela rede 1th5s5d para as diferentes entradas

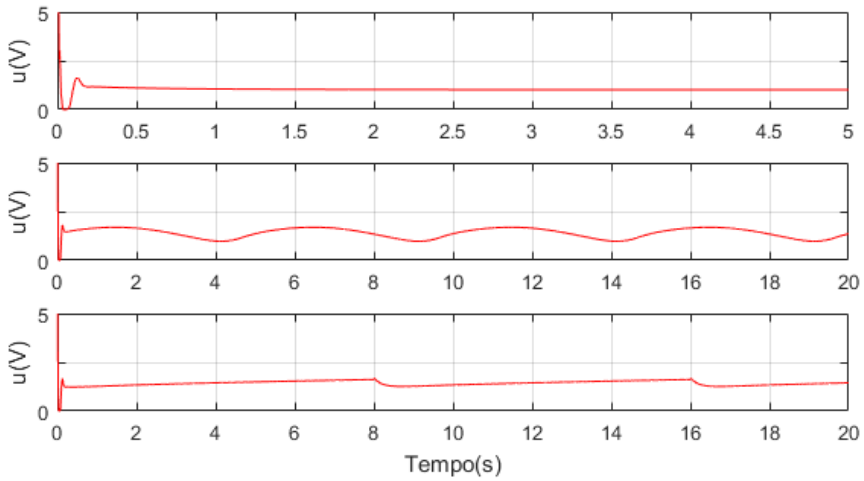


Figura 6.16: Sinal de controlo fornecido pela rede 1th5s5d para as diferentes entradas

Como se verifica, com a adição de uma camada de um neurónio com função de ativação tangente hiperbólica são obtidos bons resultados de controlo do sistema.

Em comparação com a rede 5s5d são corrigidas certas imperfeições principalmente na referência sinusoidal, e reduzido também o erro em regime permanente.

De modo a testar o controlador no sistema real, é novamente utilizado o diagrama Simulink da Figura 5.15 com o *Simulink Support Package for Arduino Hardware*. Na Figura 6.17 encontra-se a resposta do sistema real às diferentes referências, onde a azul encontra-se representada a referência fornecida e a vermelho a saída do sistema. Na Figura 6.18 estão representados os sinais de controlo fornecidos pela rede 1th5s5d para cada uma das referências.

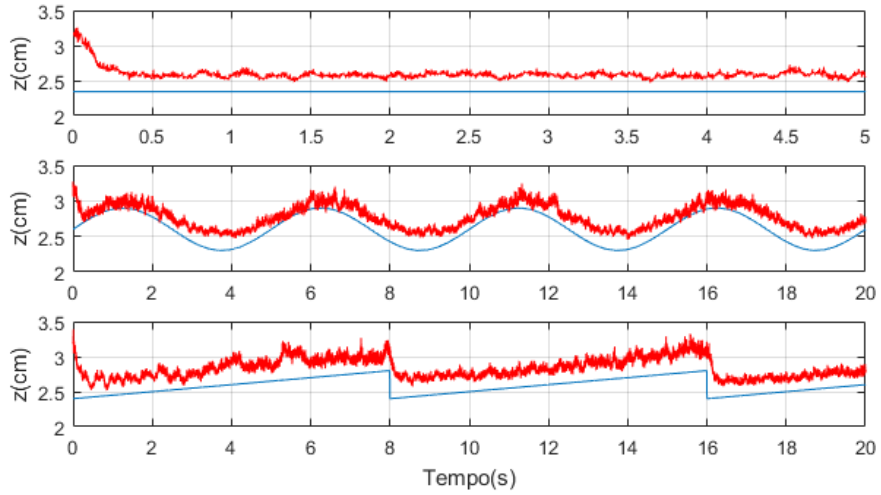


Figura 6.17: Respostas do sistema real controlado pela rede 1th5s5d para as diferentes entradas

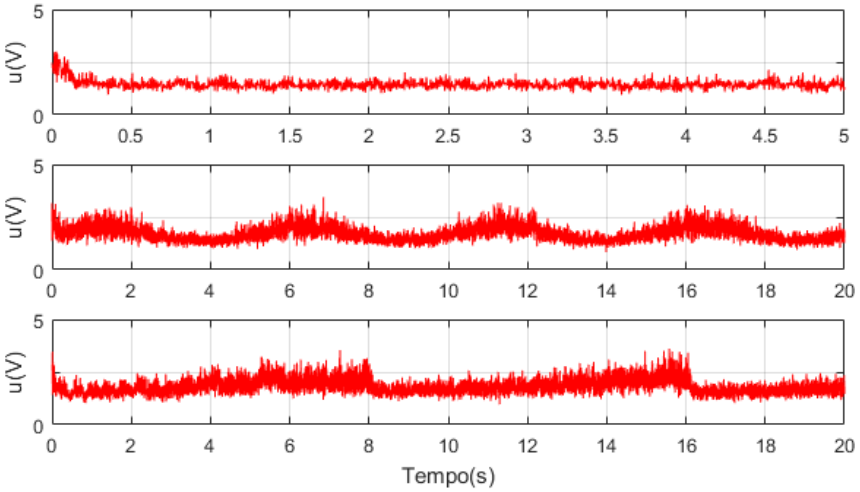


Figura 6.18: Sinal de controlo fornecido ao sistema real pela rede 1th5s5d para as diferentes entradas

É possível observar que existe um ligeiro desvio entre os resultados experimentais e simulados, principalmente em termos do erro em regime permanente, visto que no sistema real a rede neuronal consegue seguir a forma do sinal de referência.

6.2.2 Modelo inverso com bloco adaptativo

De modo a melhorar a resposta, em termos do erro em regime permanente, do sistema quando controlado por uma rede de modelo inverso é possível utilizar um bloco adaptativo. Este bloco consiste num simples integrador, ou seja aumentando o ganho a resposta será mais rápida mas também mais oscilatória.

Na Figura 6.19 encontra-se o controlador construído no Simulink, em que o valor do bloco adaptativo é adicionado à saída da rede. Assim é possível obter a estabilidade alcançada pelo controlador neuronal de modelo inverso com a redução do erro em regime permanente [15].

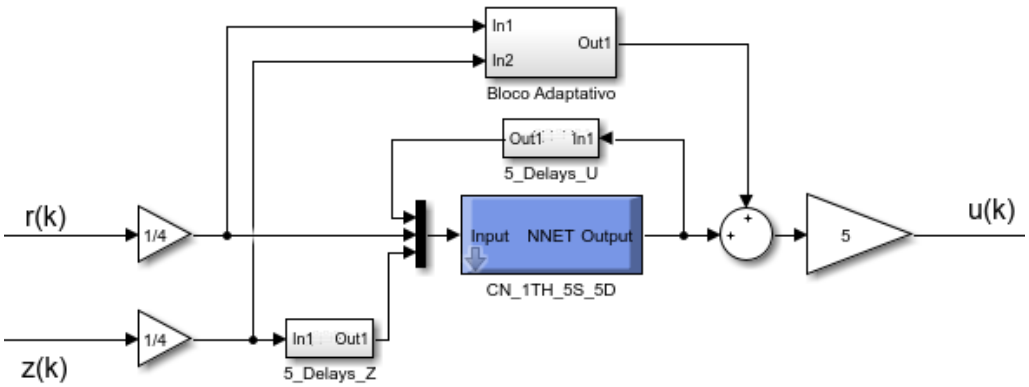


Figura 6.19: Controlador neuronal de modelo inverso 1th5s5d com bloco adaptativo

O bloco adaptativo implementa a equação (6.2), onde o parâmetro β é um ganho ajustável.

$$B = \beta \sum_{i=0}^{k} (z(i) - r(i)) \tag{6.2}$$

Na Figura 6.20 encontra-se a resposta do sistema real às diferentes referências, controlado pelo controlador de modelo inverso com bloco adaptativo, utilizando um β de 0,25. Na Figura 6.21 estão representados os sinais de controlo aplicados para cada uma das referências.

Como se verifica, com a adição do bloco adaptativo, obtém-se uma resposta bastante satisfatória. O erro em regime permanente é praticamente nulo, exis-

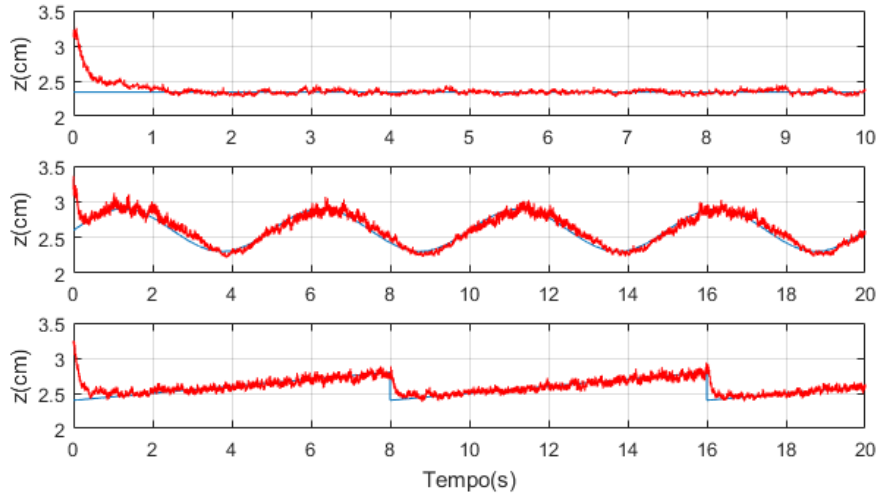


Figura 6.20: Respostas do sistema real controlado pela rede 1th5s5d com bloco adaptativo para as diferentes entradas

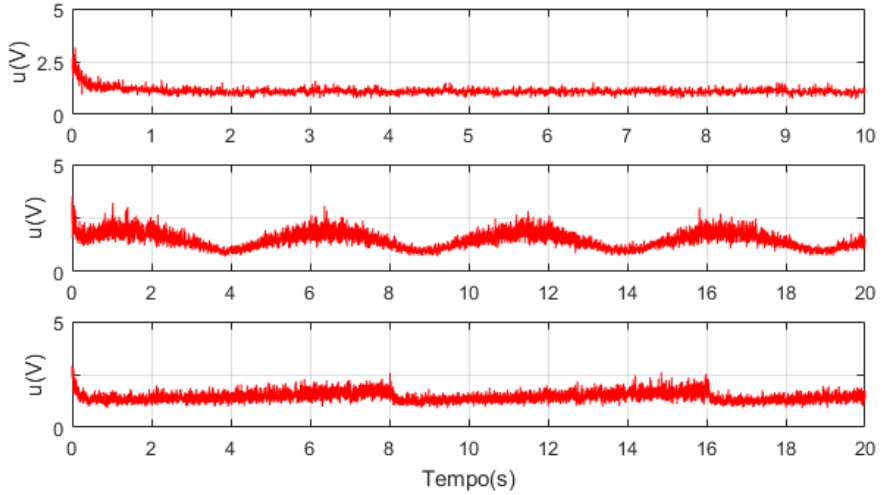


Figura 6.21: Sinal de controlo fornecido ao sistema real pela rede 1th5s5d com bloco adaptativo para as diferentes entradas

tindo apenas um ligeiro desvio quando o valor da referência varia rapidamente, como é o caso da referência em serra. A resposta do sistema é bastante estável não sendo visíveis quaisquer oscilações no disco magnético que indiquem que o sistema esteja a tornar-se instável.

6.2.3 Modelo interno

Com a rede de identificação e a rede de controlo por modelo inverso treinadas e aprovadas, é possível desenvolver um controlador de modelo interno. Este consiste

num controlador que subtrai o erro entre o sistema e a rede de identificação ao valor de referência do controlador de modelo inverso, de modo a tentar reduzir o ruído [13] [15].

Antes de implementar este controlador é necessário um filtro discreto, para o valor do erro entre o sistema e rede de identificação. Para isso é projetado o filtro passa-baixo da equação (6.3).

$$H(z) = \frac{a}{1-(1-a)z^{-1}},\ 0 < a < 1 \tag{6.3}$$

O valor de a pode ser calculado através da equação (6.4), onde é necessário garantir que o valor da constante de tempo (τ_F) seja superior a cinco vezes o valor de T, ou seja pelo menos 0,015 segundos.

$$a = \frac{T}{\tau_F + T} \tag{6.4}$$

Assim, utilizando um τ_F igual a 0,20 segundos é obtido um a de 0,1304. Resulta assim o filtro passa-baixo discreto da equação 6.5.

$$H(z) = \frac{0,1304}{1-0.8696z^{-1}} \tag{6.5}$$

Com o filtro projetado, é possível construir o diagrama Simulink da Figura 6.22 do controlador de modelo interno. Como este controlador tem como objetivo atenuar as perturbações no sistema, é adicionado ruído com distribuição normal à saída do sistema, com média de 0 e variância de 0,005.

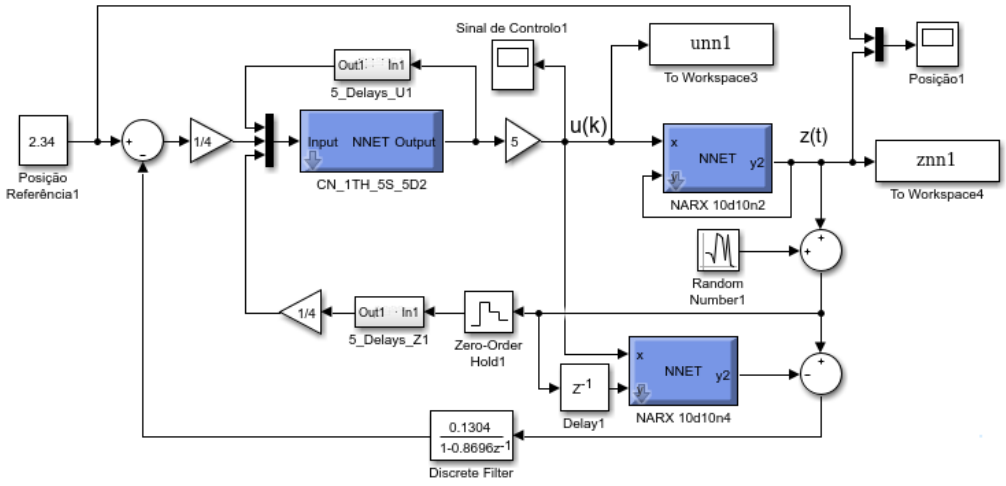


Figura 6.22: Diagrama Simulink do controlador de modelo interno

É possível agora simular o sistema para as diferentes entradas de referência, de modo a comparar com o controlador inverso. Assim nas seguintes figuras encontram-se as respostas do sistema controlado pelo controlador inverso a azul e pelo controlador de modelo interno a vermelho. Na Figura 6.23 encontra-se a resposta ao degrau, na Figura 6.24 a resposta à sinusóide e na Figura 6.25 a resposta a uma referência em serra.

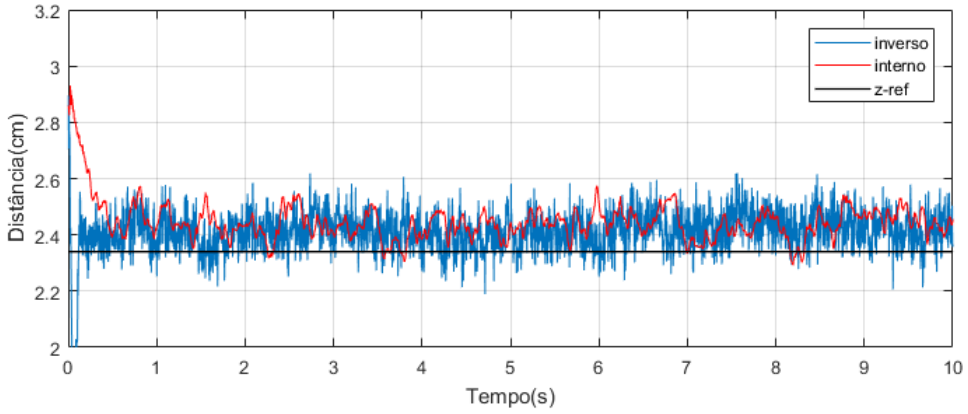


Figura 6.23: Resposta do sistema controlado pelo controlador de modelo interno para um referência em degrau

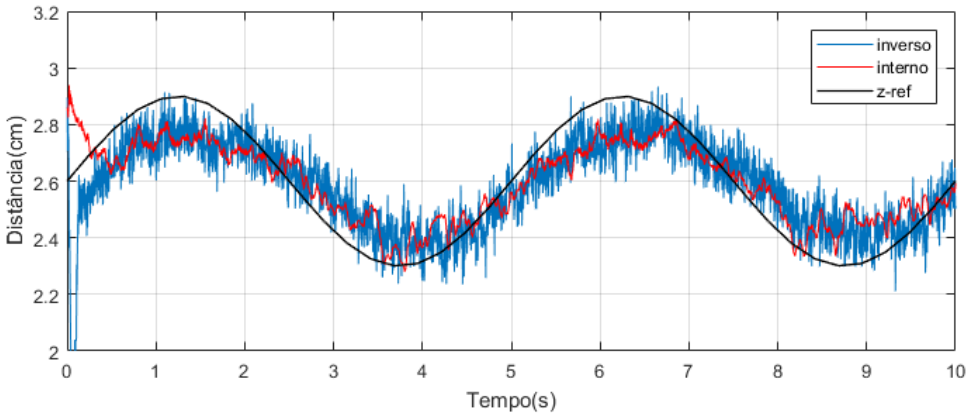


Figura 6.24: Resposta do sistema controlado pelo controlador de modelo interno para uma referência sinusoidal

O sinal de controlo é semelhante entre os dois controladores, possuindo uma amplitude e forma idêntica, tal como mostrado na Figura 6.26 para uma referência sinusoidal.

No entanto, como é possível averiguar nos gráficos das Figuras 6.23, 6.24 e 6.25, as oscilações são bastante reduzidas no sistema controlado pelo controlador de modelo interno, sendo visíveis pequenas distorções no sinal de posição do controlador de modelo interno. Esta diminuição da oscilação do sistema, deve-se

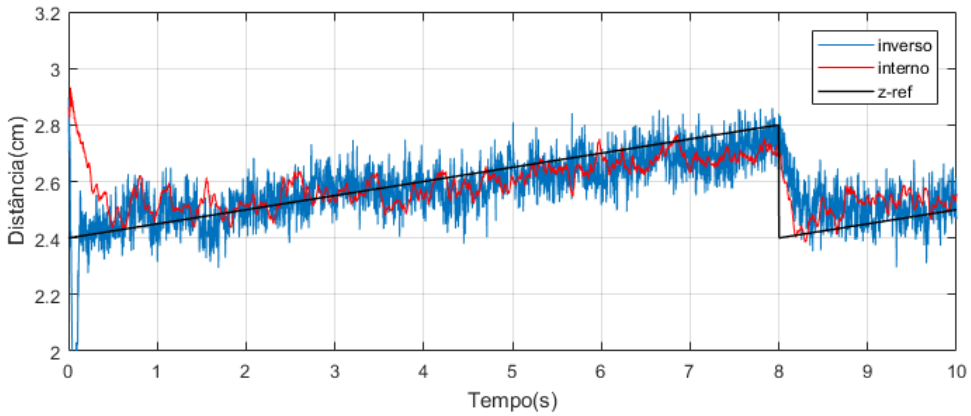


Figura 6.25: Resposta do sistema controlado pelo controlador de modelo interno para uma referência em serra

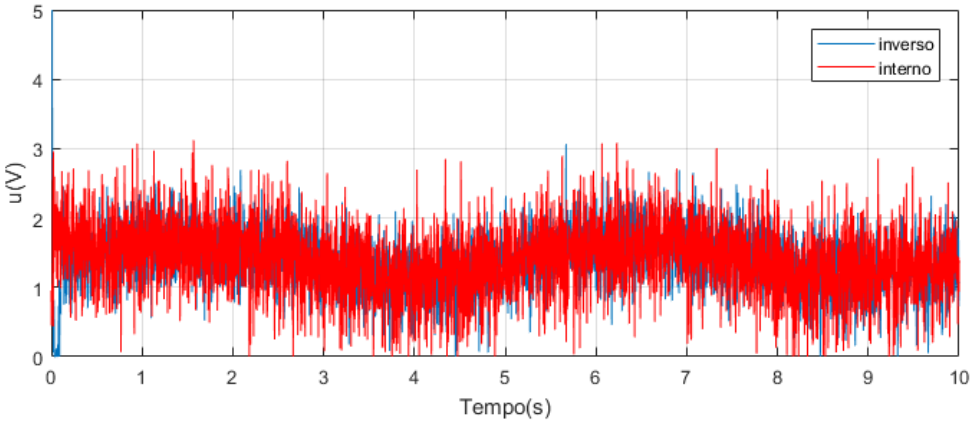


Figura 6.26: Controlo aplicado pelo controlador de modelo interno para uma referência sinusoidal

ao facto de que antes de ser calculado o sinal de controlo a aplicar no instante atual, a rede de identificação calcula a posição do eletroíman mediante posição e sinal de controlo anteriores. A diferença entre a rede de identificação e o sistema é retirado à referência fornecida à rede de modelo inverso.

Nenhum dos controladores utilizou um bloco adaptativo para reduzir o erro em regime permanente, mas como é de esperar, ao adicionar o bloco o erro presente nas respostas irá reduzir. É possível verificar na Figura 6.27 a resposta do sistema controlado por cada um dos controladores com bloco adaptativo e na Figura 6.28 os respetivos sinais de controlo. Como se confirma a resposta obtida com ambos os controladores possui um erro menor, sendo que a resposta do controlador de modelo interno apresenta menos ruído.

Este controlador requer uma elevada capacidade de computação, não sendo possível implementar no sistema real, com o tempo de amostragem definido de 3

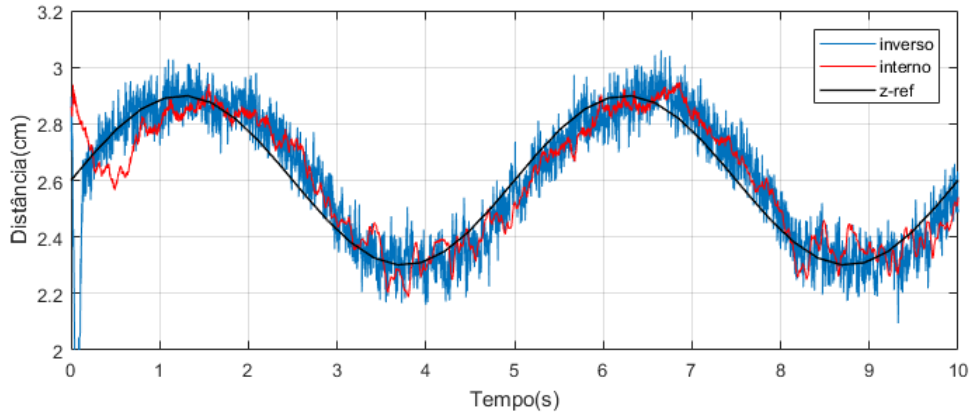


Figura 6.27: Resposta do sistema controlado pelo controlador de modelo interno com bloco adaptativo para uma referência sinusoidal

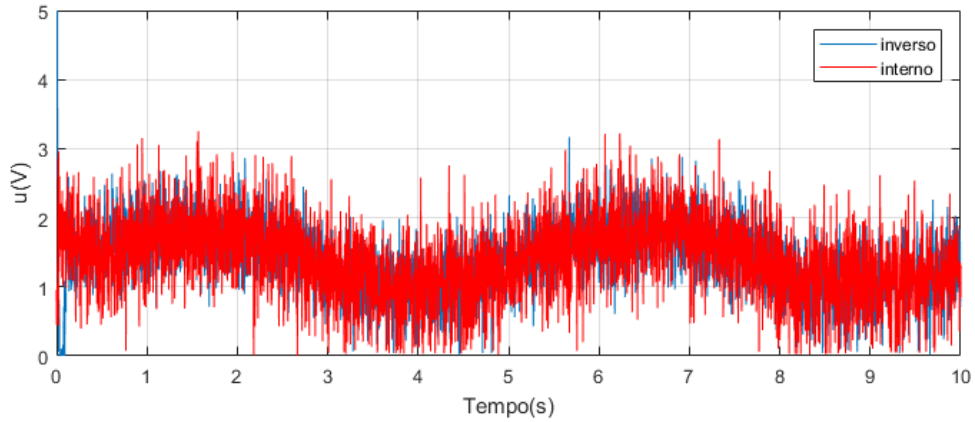


Figura 6.28: Sinal de controlo aplicado pelo controlador de modelo interno com bloco adaptativo para uma referência sinusoidal

milissegundos. Aumentar o tempo de amostragem demonstrou tornar o funcionamento do sistema instável. No entanto, tendo em conta que a rede de identificação consegue emular o funcionamento do sistema satisfatoriamente, é previsível que no sistema real a resposta seja semelhante à resposta simulada.

6.2.4 Modelo de referência

Para obter o controlador de modelo de referência, é utilizado o código MATLAB do Anexo E. Como este controlador tem como objetivo controlar o sistema segundo um modelo pretendido, o desempenho do controlador durante o treino representa o erro entre o modelo de referência e o sistema, como representado na Figura 3.16. Para isso é criada a rede de cinco camadas da Figura 6.29, na qual as primeiras três correspondem ao controlador as e duas últimas ao sistema.

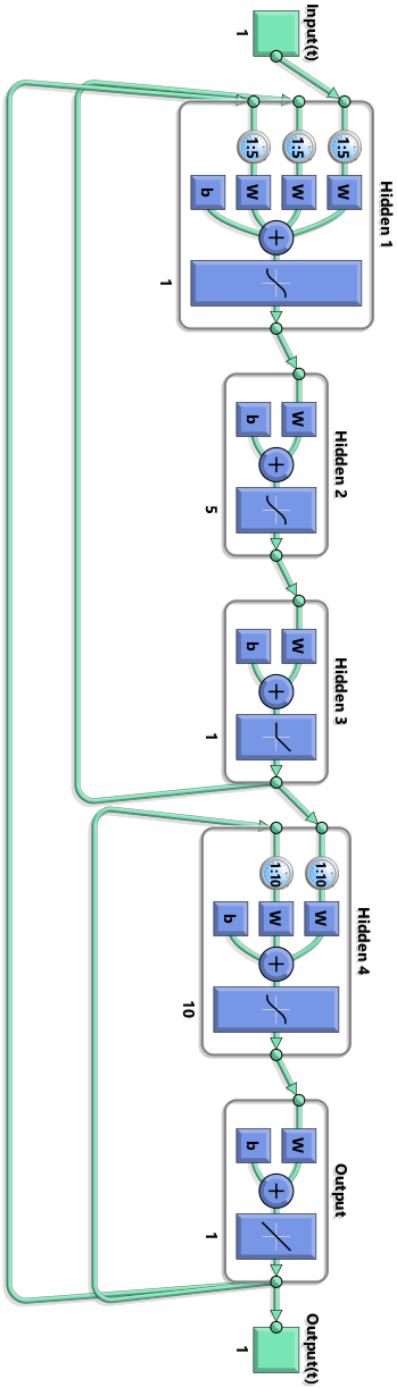


Figura 6.29: Rede do controlador de modelo de referência

O primeiro passo no projeto deste controlador é desenvolver uma rede que identifique o sistema a controlar. Como este passo foi realizado anteriormente no sub-capítulo 6.1, será utilizada a rede de identificação 10d10n que obteve um bom desempenho emulando o sistema corretamente. Esta é uma rede de duas

camadas com amostras dos sinais de controlo e posição nos 10 instantes anteriores (*delays*).

De modo a treinar o controlador é então utilizada a rede de 5 camadas criada, sendo as últimas duas iguais à rede de identificação, ou seja a quarta camada possuí 10 neurónios de função de ativação sigmóide e a quinta apenas um neurónio de função linear. Quanto ao controlador, a primeira camada da rede é constituída por um neurónio de função de ativação tangente hiperbólica, a segunda por cinco neurónios de função sigmóide e a terceira apenas um neurónio com função ReLU. Em relação aos sinais na entrada, esta rede recebe amostras da referência, posição do sistema e sinal de controlo, dos cinco instantes anteriores, sendo a posição do sistema e sinal de controlo realimentados da quinta e terceira camada, respetivamente. Depois de criar a rede, são copiados os valores dos pesos sinápticos e polaridades da rede 10d10n para as duas últimas camadas, e de modo a treinar corretamente o controlador é desativada a aprendizagem nessas duas últimas camadas. Assim o sistema não varia durante o treino, sendo as camadas do controlador adaptadas de modo ao sistema seguir os valores pretendidos segundo a referência fornecida.

Como dados de treino são utilizadas amostras do sinal de referência aleatório fornecido ao controlador inverso com bloco adaptativo e a respetivas amostras da respostas do sistema real. Assim, utilizando a retro-propagação de Levenberg-Marquardt (`trainlm`), é treinada a rede durante 50 iterações e obtido um desempenho (MSE) de 0,0025. A resposta da rede encontra-se representada na Figura 6.30, sendo a resposta da rede a saída do sistema e como se verifica, o controlador consegue fazer o sistema seguir a referência com um erro em regime quase nulo. O sinal de controlo neste controlador é o sinal fornecido da terceira camada para a quarta camada.

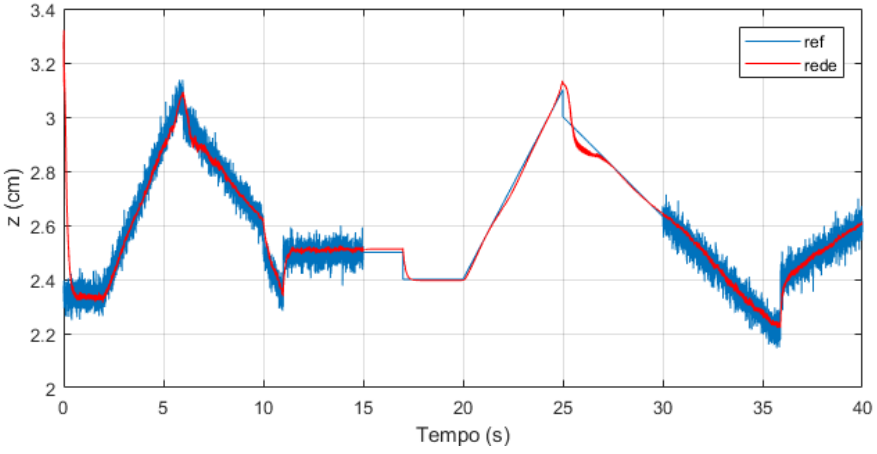


Figura 6.30: Resposta da rede de modelo de referência à referência aleatória de treino

De modo a confirmar que o treino da rede foi bem sucedido, é testada a rede para os sinais de referência utilizados anteriormente: a entrada em degrau, o dente de serra e sinusóide. O desempenho da rede para cada referência encontra-se na seguinte Tabela 6.7, onde como se averigua o desempenho é semelhante para todas as referências.

Tabela 6.7: MSE da rede de modelo de referência para diferentes sinais de referência

Sinal de referência	MSE
Treino	2,5257E-3
Degrau	1,0413E-3
Dente de serra	1,7806E-3
Sinusóide	2,2322E-3

Com o intuito de visualizar o desempenho obtido, a resposta do sistema para cada referência encontra-se na seguinte Figura 6.31, e como se verifica, o sistema possui respostas satisfatórias para cada sinal, estas quase não possuem erro em regime permanente, o *overshoot* é nulo e o tempo de estabelecimento é baixo.

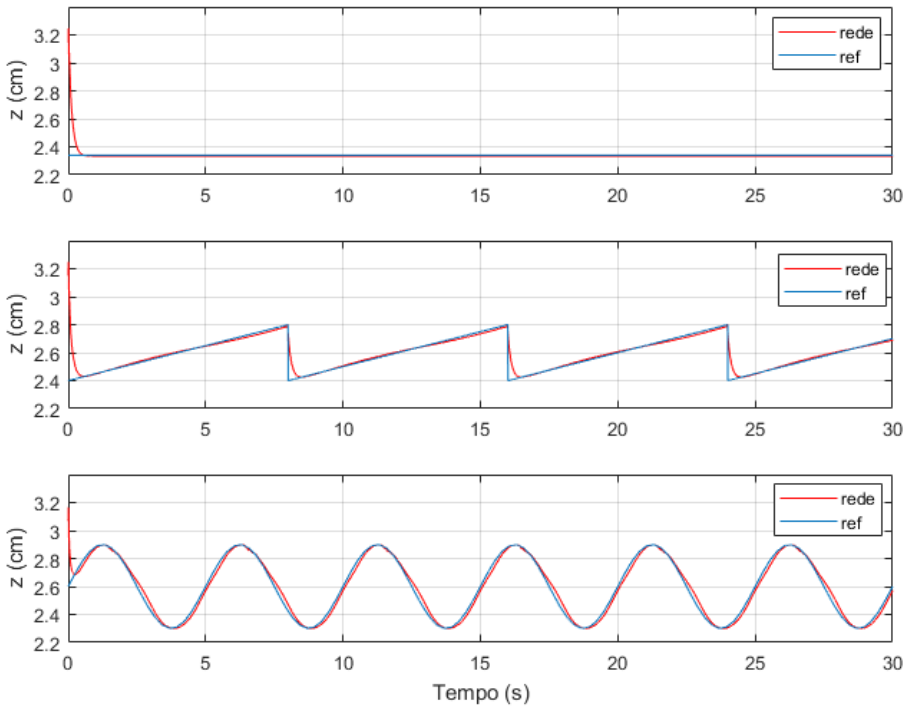


Figura 6.31: Resposta da rede de modelo de referência para as diferentes entradas

De modo a analisar o sinal de controlo e desempenho do controlador quando as medições possuem ruído, é desenvolvido o diagrama Simulink da Figura 6.32 onde como se verifica, é utilizada a rede de cinco camadas desenvolvida com a

diferença que é adicionado ruído com uma distribuição normal, com média de 0 e variância de 0,005, à saída do sistema.

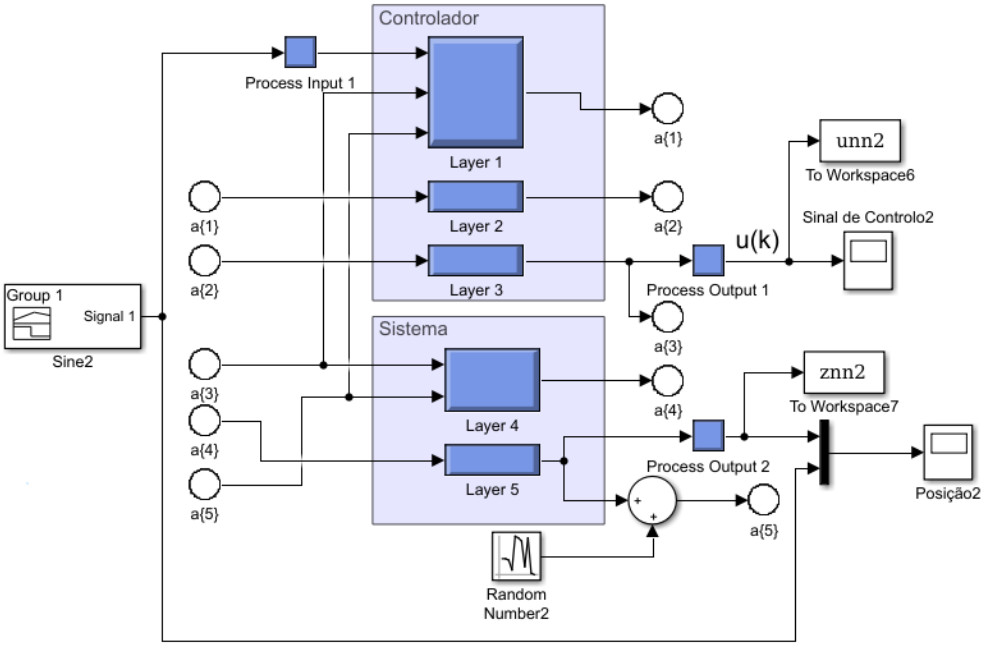


Figura 6.32: Diagrama Simulink do controlador de modelo de referência

Com este diagrama é possível simular o sistema para diferentes entradas com e sem ruído acrescentado. Na Figura 6.33 e na Figura 6.34 encontram-se representados a resposta do sistema e o sinal de controlo, respetivamente, para uma referência sinusoidal.

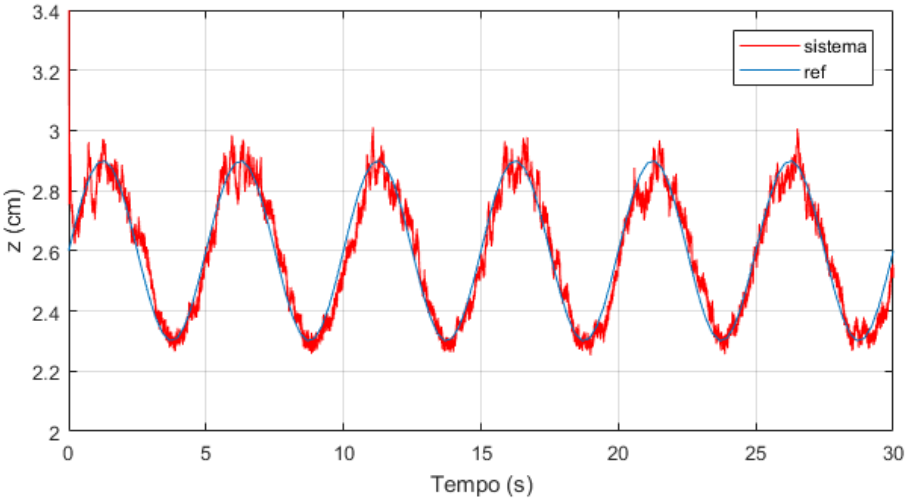


Figura 6.33: Resposta do sistema controlado pelo controlador de modelo de referência para uma referência sinusoidal

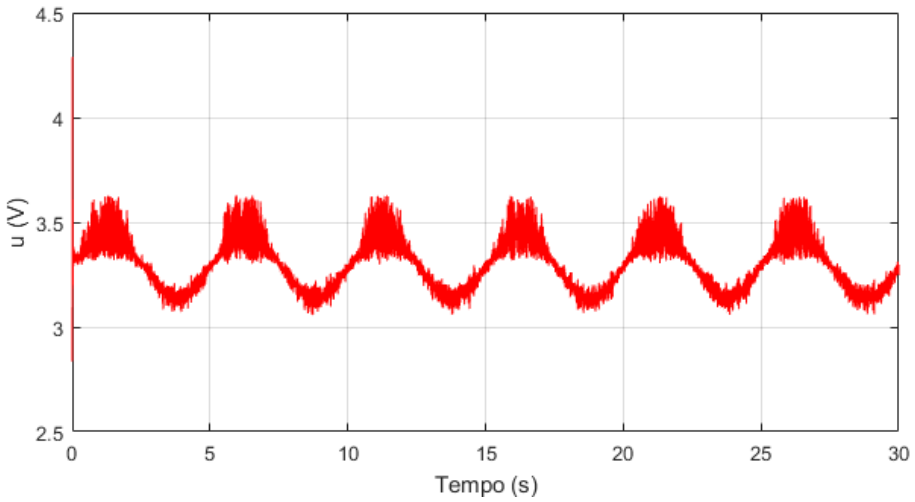


Figura 6.34: Sinal de controlo aplicado pelo controlador de modelo de referência para uma referência sinusoidal

Como se observa, o controlador mantém o sistema a seguir a referência mesmo com a adição do ruído na saída. Ao analisar o sinal de controlo é possível ver o controlador oscilar mais nas secções em que o sistema se encontra mais afastado do sensor. É notável a capacidade deste controlador conseguir controlar o sistema de modo a praticamente não existir um erro em regime permanente sem a necessidade de um bloco adaptativo, o que não aconteceu nos controladores anteriores.

Tal como o controlador de modelo interno, também este requer uma capacidade de computação acima da disponível, não sendo possível implementar no sistema real, com o tempo de amostragem definido de 3 milissegundos. Mas com as simulações realizadas é possível aferir que este controlador, ao ser treinado mediante um modelo de referência pretendido, aprende a controlar o sistema para seguir a referência em vez de aprender a dinâmica inversa.

Capítulo 7

Implementação prática em C

No capítulo atual encontra-se detalhada a implementação dos diferentes controladores analisados anteriormente. É desenvolvido e explicado o algoritmo na linguagem C onde esses controladores são implementados, com recurso ao micro-controlador ATmega2560. É também analisado todas as configurações realizadas no micro-controlador de modo a obter o protótipo final.

7.1 Configurações do Micro-controlador

Com o objetivo de implementar o algoritmo de controlo utilizado no micro-controlador e a sua comunicação com o computador, é necessário realizar as corretas configurações do ATmega2560.

Para estabelecer o período de amostragem para realizar o controlo do sistema é configurado o TIMER0 no modo de funcionamento *Clear Timer on Compare*, assim o contador incrementa o seu valor até atingir o valor colocado no registo OCR0A. A interrupção ativada neste modo de funcionamento acontece sempre que o valor do registo OCR0A é atingido, e esta interrupção necessita de ocorrer a cada 3 ms. Deste modo o TIMER0 é configurado com o *prescaller* de 256 e o valor do registo OCR0A é calculado através da equação (7.1) para um período de 6 ms (duas interrupções).

$$OCR0A = \frac{16 \times 10^6}{166,667 \times 256 \times 2} - 1 \tag{7.1}$$

onde a frequência de relógio é dividida pelo produto da frequência pretendida com o *prescaller* multiplicado por 2 e a este quociente é subtraído 1, sendo obtido um OCR0A de 186. A frequência pretendida é calculada em função do período de 6 ms.

O TIMER1 tem como função gerar o sinal que controla o MOSFET. Para isso, é configurado no modo de funcionamento *Fast* PWM de modo a contar até ao valor colocado no registo ICR1, é também ativado o pino OC1A em modo não invertido, ou seja o pino é colocado a 1 quando o valor do contador é superior ao valor do registo OCR1A. De modo a obter a frequência desejada de 1000 Hz é necessário calcular o valor a colocar no registo ICR1, para isso utiliza-se a equação (7.2), onde a frequência de relógio é dividida pela frequência pretendida para o PWM multiplicada por um *prescaller* de 1.

$$ICR1 = \frac{16 \times 10^6}{1000 \times 1} - 1 \tag{7.2}$$

É obtido assim um ICR1 de 15999 pois este registo é de 16 bits. Com isto é gerado um PWM de 1000 Hz onde será variado o *duty cycle* consoante o resultado do controlador, sendo que um incremento de 1 no registo OCR1A equivale a um aumento de $0,00625\%$ no sinal de controlo.

A USART0 tem como função realizar uma comunicação série com o computador (através da porta USB). Assim definiu-se com uma taxa de envio (*baudrate*) de 250000 bits por segundo (bps). Configura-se uma comunicação assíncrona sem paridade, um *stop* bit e com *double speed* ativado. É então ativado o recetor e transmissor, e de modo a ler o que é recebido do computador, apenas quando é recebido algo é ativada a interrupção de receção. Para definir o *baudrate* é utilizada a equação (7.3) para calcular o valor a colocar no registo UBRR0.

$$UBRR0 = \frac{16 \times 10^6}{8 \times 250000} - 1 \tag{7.3}$$

É assim obtido um UBRR0 de 7, sendo que a *baudrate* obtida é igual à pretendida, ou seja 0,0% de erro.

O ADC é configurado com um *division factor* de 128, obtendo assim uma frequência de 125 kHz. É também definido de modo a ser obtida a conversão com 10 bits. Como referência é utilizada a tensão de 5 V do pino AVCC fornecida pelo micro-controlador.

Por último são configurados os pinos necessários para controlar o MOSFET e obter os valores dos sensores. Assim são definidos como saída os pinos:

- Pino PB5 (OC1A) para controlar o MOSFET;
- Pino PB7 para o LED de 3 ms controlado na interrupção do TIMER0.

Como entrada, encontram-se definidos os pinos PF0 e PF1 correspondendo ao ADC0 e ADC1, respetivamente. O ADC0 converte o valor do sensor de posição e o ADC1 o sensor de corrente.

As principais configurações realizadas encontram-se apresentadas, de forma sucinta, na Tabela 7.1.

Tabela 7.1: Configuração do ATmega2560

USART0		
Assíncrona	250000 bps	8 bit de trasnmissão
1 *Stop* bit	Sem paridade	0,0% de erro
TIMER0		
CTC	166,6 Hz	Interrupção Ativada
TIMER1		
Fast PWM	1000 Hz	Sem Interrupção
ADC		
10 bit	125 kHz	AREF=AVCC

7.2 Controladores discretos

O controladores do sistema encontram-se na mesma função, realizada a cada 3 ms. O funcionamento dessa função é dado pelo fluxograma da Figura 7.1, onde como se verifica, primeiro é calculado o erro utilizado pelos controladores clássicos e pelo bloco adaptativo.

O próximo passo verifica qual dos controladores é o selecionado e realiza os cálculos necessários nesse controlador, que no caso dos controladores neuronais é chamada a função com a rede implementada.

Por último é atualizado o vetor dos sinais de controlo (presente e quatro anteriores), saturado o valor entre 0 e 5, e por fim calculado o *duty cycle* do PWM a aplicar.

7.2.1 Controladores clássicos

De modo a implementar os controladores clássicos desenvolvidos anteriormente é primeiro necessário utilizar a transformada de Z inversa para obter as equações às diferenças para cada controlador, visto que estes já se encontram no domínio discreto.

A saída do controlador em avanço dado pela equação (5.40) pode ser descrito pela equação (7.4).

$$\Delta U(z) = \beta_{av} z^{-1} \Delta U(z) - \alpha_{av} K_{av} z^{-1} E(z) + K_{av} E(z) \tag{7.4}$$

Assim é possível realizar a transformada de Z inversa obtendo a equação às diferenças (7.5) do sinal de controlo a aplicar no instante k. Este depende do sinal de erro no instante k e $k-1$ e do sinal de controlo no instante anterior.

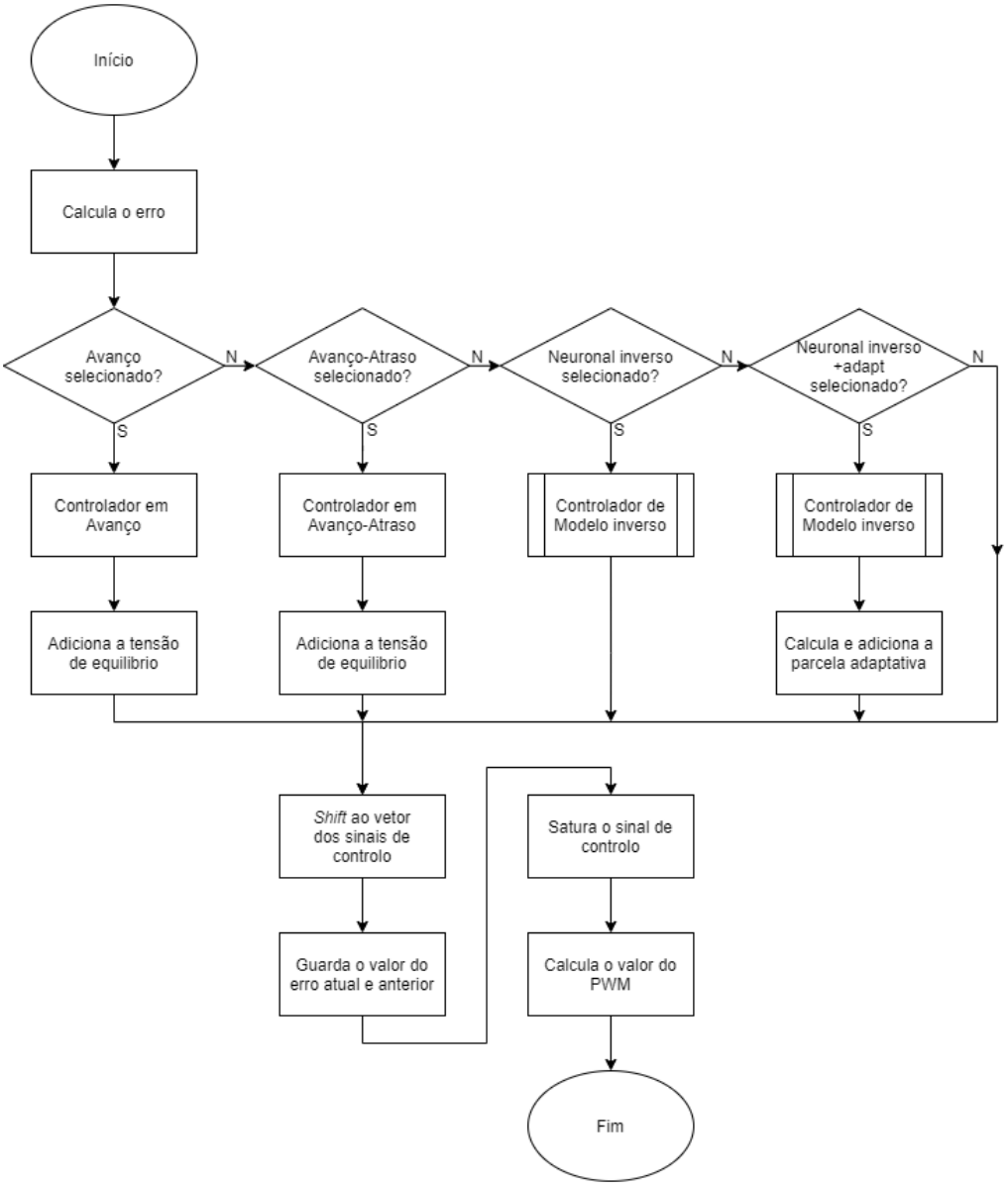


Figura 7.1: Fluxograma da função de controlo

$$\Delta u(k) = \beta_{av}\Delta u(k-1) - \alpha_{av}K_{av}e(k-1) + K_{av}e(k) \tag{7.5}$$

No caso do controlador em avanço-atraso dado pela equação (7.6), onde $K_{at} = 1$, ao desenvolver a sua função de transferência discreta é obtida a equação (7.7) para a variação do sinal de controlo.

$$D(z) = \frac{\Delta U(z)}{E(z)} = K_{av}\frac{z-\alpha_{av}}{z-\beta_{av}} \times K_{at}\frac{z-\alpha_{at}}{z-\beta_{at}} \tag{7.6}$$

$$\Delta U(z) = Az^{-1}\Delta U(z) - Bz^{-2}\Delta U(z) + K_{av}E(z) - Cz^{-1}E(z) + Dz^{-2}E(z) \tag{7.7}$$

onde:

- $A = \beta_{av} + \beta_{at}$;
- $B = \beta_{av}\beta_{at}$;
- $C = (\alpha_{av} + \alpha_{at})K_{av}$;
- $D = \alpha_{av}\alpha_{at}K_{av}$.

Ao realizar a transformada de Z inversa é obtida a equação às diferenças (7.8) do sinal de controlo a implementar no micro-controlador. Como se verifica, este controlador utiliza amostras do sinal de controlo e erro nos instantes atuais e anteriores.

$$\Delta u(k) = A\Delta u(k-1) - B\Delta u(k-2) + K_{av}e(k) - Ce(k-1) + De(k-2) \tag{7.8}$$

Ambos estes controladores, como explicado, calculam a variação do sinal de controlo em torno do seu valor de equilíbrio, sendo necessário em ambos adicionar 0,783 antes de ser calculado o *duty cycle* do sinal PWM a aplicar no MOSFET.

7.2.2 Controladores neuronais

Quanto aos controladores neuronais é implementado o modelo inverso sem e com o bloco adaptativo. Para esse fim, é necessário criar os vetores com os pesos e polaridades para cada camada da rede.

Em ambos os controladores neuronais é calculado o sinal de controlo a ser aplicado pela rede de modelo inverso de três camadas com 5 *delays* e 5 neurónios com função de ativação sigmóide na segunda camada. Assim são definidos os cálculos realizados pela rede na função do excerto de código seguinte:

```
//Rede neuronal de modelo inverso
float  CN_1TH_5S_5D()
{
    unsigned char i=0;//para os ciclos  for

    CN_SUM1=0.0;//Reset aos somatorios da camada 1
    for  (i=0; i>5; i++)//de 0 a 4
        CN_SUM2[i]=0.0;//Reset aos somatorios da camada 2
    CN_SUM_out=0.0;//Reset aos somatorios da camada 3
```

```
    //primeira camada
    for( i=0; i<5; i++)//de 0 a 4
    {
        CN_SUM1+=W1_CN[i]*(u_delays[i]/5);//calcula para os valores de u
        CN_SUM1+=W1_CN[6+i]*(z_delays[i]/4);//calcula para os valores de z
    }
        CN_SUM1+=W1_CN[5]*(z_ref/4);//calcula para a referencia
        CN_SUM1+=B1_CN;//adiciona a polaridade
        CN_L1=atv_tanh(CN_SUM1);//Saida da primeira camada

    //segunda camada
    for( i=0; i<5; i++)//de 0 a 4
    {
        CN_SUM2[i]=W2_CN[i]*CN_L1+B2_CN[i];//calcula o somatorio
        CN_L2[i]=atv_sig(CN_SUM2[i]);//calcula saida de cada neuronio
    }

    //terceira camada
    for( i=0; i<5; i++)//de 0 a 4
        CN_SUM_out+=W3_CN[i]*CN_L2[i];//calcula o somatorio
    CN_out=CN_SUM_out+B3_CN;//Saida da ultima camada

    return (CN_out*5);//devolve o sinal de controlo
}
```

Esta é a função chamada na função de controlo cujo fluxograma foi analisado na Figura 7.1, onde W1_CN, W2_CN, W3_CN são os vetores com os pesos para cada camada e B1_CN, B2_CN, B3_CN as polaridades.

Como se pode ver é primeiro calculado o somatório de cada camada e de seguida é aplicada a função de ativação, sendo estas definidas nas seguintes macros do excerto de código seguinte:

```
//Macros
#define atv_tanh(val_f) ((2/(1+exp(-2*val_f)))-1) //tangente hiperblica
#define atv_sig( val_f ) (1/(1+exp(-1*val_f)))      //sigmoide
```

O sinal de controlo que a rede retorna irá ser saturado entre 0 e 5 na função de controlo e só depois irá ser calculado o *duty cycle* a aplicar. Caso seja selecionado o controlador neuronal com o bloco adaptativo, a parcela do bloco é calculada dentro da função de controlo logo após a função da rede neuronal retornar o valor do sinal de controlo.

7.3 Funcionamento do algoritmo

O funcionamento do programa é dado pelo fluxograma da Figura 7.2, onde é possível averiguar que antes do micro-controlador entrar no ciclo infinito são realizadas as configurações do sistema e calculados os coeficientes dos controladores.

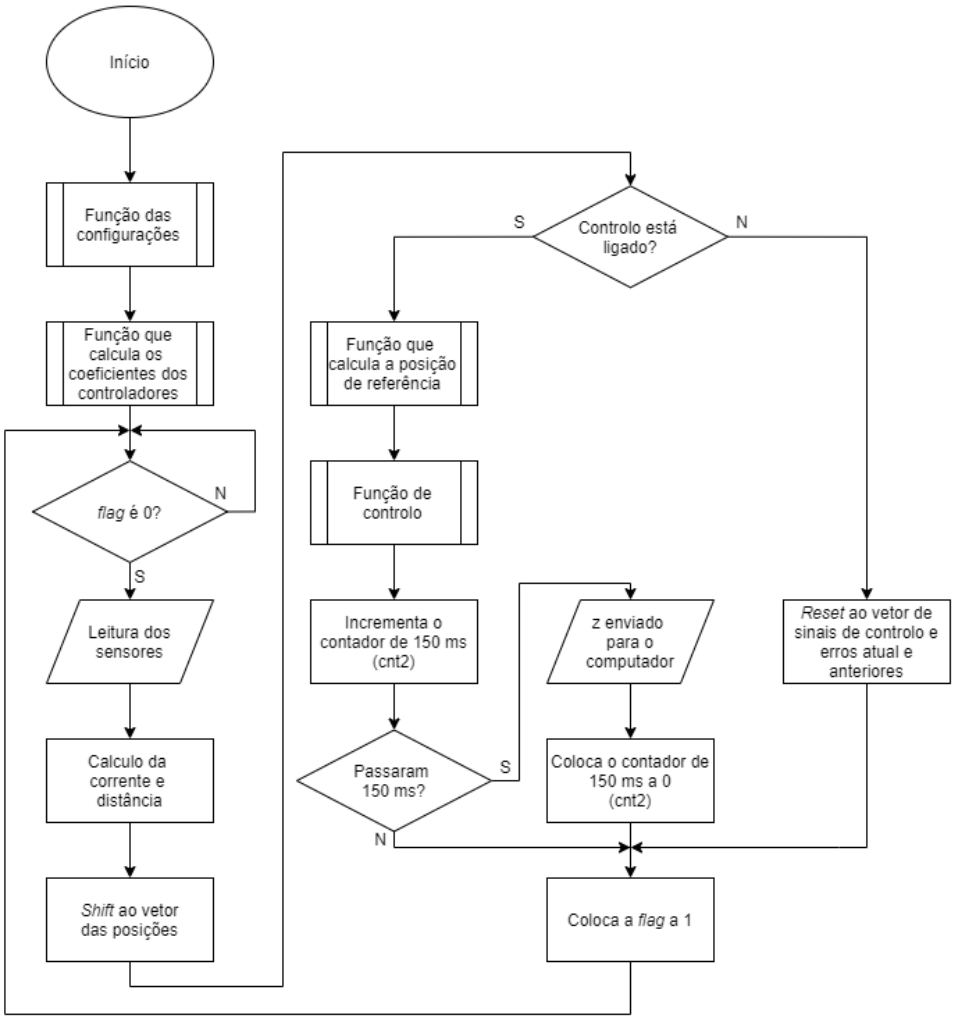


Figura 7.2: Fluxograma do funcionamento do algoritmo C

De seguida é verificada uma *flag* que caso seja 0 é iniciado um ciclo, caso o valor seja 1 o algoritmo continuará a realizar a verificação da *flag* até que o seu valor altere para 0 dentro da interrupção do TIMER0. O funcionamento da interrupção é dado pelo fluxograma da Figura 7.3 onde, como se verifica, a cada 3 ms é alterado o valor do *duty cycle* do sinal de PWM aplicado no MOSFET, incrementado e verificado o contador de 500 ms (`cnt1`) do LED *status* e no fim colocada a *flag* a 0.

Após a confirmação que a *flag* se encontra com o valor 0, é realizada a leitura e conversão dos sensores de posição e corrente, sendo a corrente calculada. Ao obter o valor da corrente é então calculada a distância do íman permanente ao sensor de posição. Para este fim são utilizadas as equações explícitas na secção

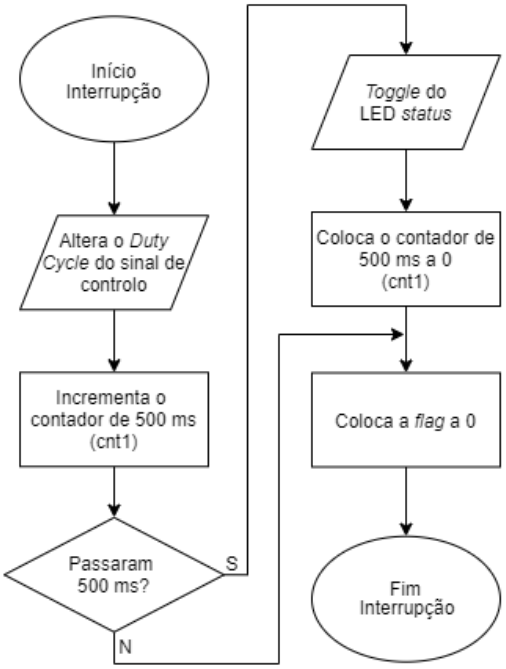


Figura 7.3: Fluxograma da interrupção do TIMER0

5.2 da parametrização do sistema.

Com a distância calculada é atualizado o vetor de posições, o qual possuí a posição atual e a posição nos quatro instantes anteriores. Posteriormente, é verificado se o controlo do sistema foi ativado, se sim é calculado o valor de referência para a distância, mediante o sinal escolhido, podendo este ser:

- Um valor constante de 2,25 cm;
- Uma onda quadrada de 0,2 Hz que varia entre 2,25 cm e 3,3 cm;
- Uma onda em dente de serra que aumenta de 2,4 cm até 3,4 cm a cada 5 s.

Este cálculo é realizado dentro de uma função cujo funcionamento é dado pelo fluxograma da Figura 7.4.

Após calcular o valor de referência é iniciada a função de controlo descrita anteriormente onde é calculado o valor do *duty cycle* a aplicar. Se seguida é incrementado um contador (`cnt2`) de modo a enviar o valor da posição calculado para o computador a cada 150 ms.

Caso o controlo esteja desligado, o sinal de controlo, e o vetor com os sinais de controlo nos instantes anteriores e o valor do erro são colocados a zero. Este *reset* é realizado de modo a evitar erros quando o controlo for novamente ativado.

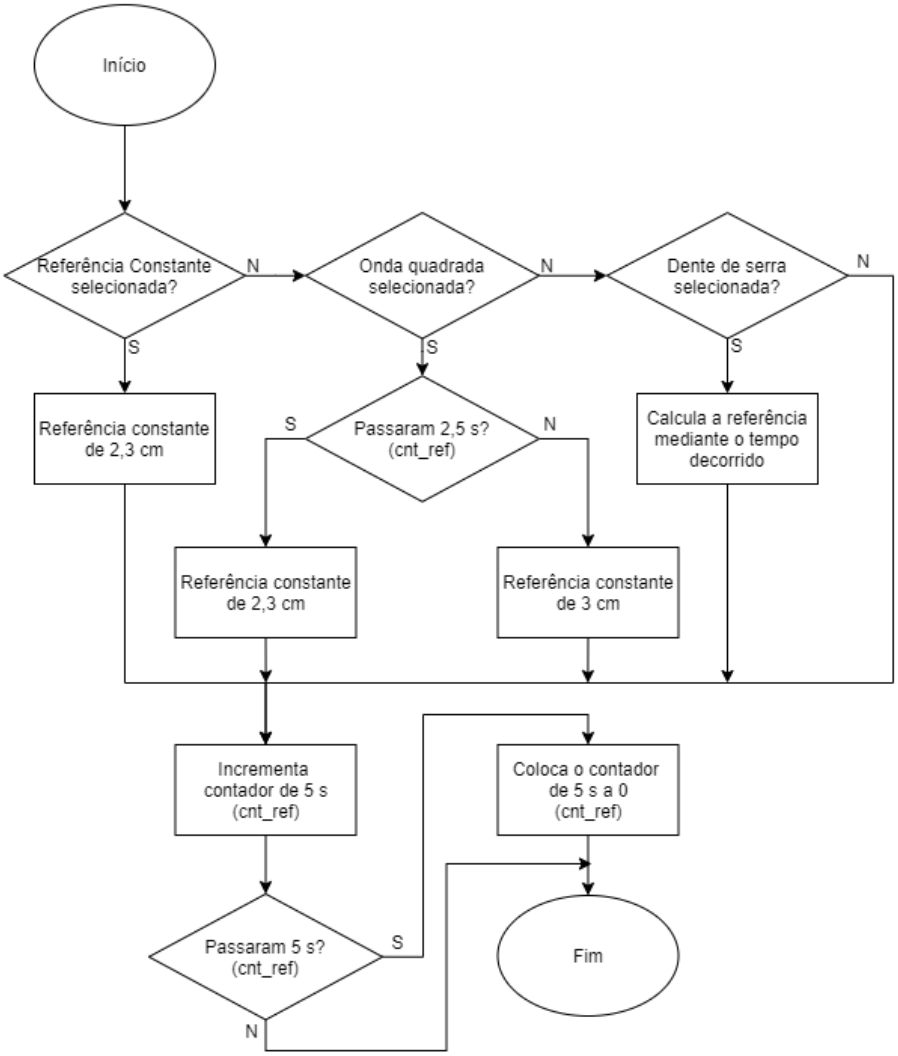


Figura 7.4: Fluxograma do funcionamento da função que calcula a distância de referência

Por último a *flag* é colocada a 1, *flag* esta que é colocada a 0 dentro da interrupção do TIMER0. Assim todo este ciclo, desde a leitura dos sensores até à alteração da *flag*, só acontece uma vez a cada 3 ms.

A seleção do controlador, da ativação e desativação do controlo, e a escolha do tipo de referência são realizados com recurso a um computador. Para este fim é utilizado o monitor série do Arduino *Integrated Development Environment* (IDE) para enviar os carateres necessários a realizar as ações pretendidas, sendo o protocolo de comunicação utilizado representado na Tabela 7.2. Assim sempre que um carater é enviado, é ativada a interrupção de receção da USART0 onde é colocado em prática a opção correspondente.

Tabela 7.2: Carateres enviados para o ATmega2560

Carater enviado	Função
a	Ativar controlo
b	Desativar controlo
i	Referência constante
o	Referência onda quadrada
p	Referência dente serra
z	Avanço
x	Avanço-Atraso
c	Modelo Inverso
v	Modelo Inverso + Adapt

Todo o código C desenvolvido para o funcionamento descrito encontra-se no Anexo F.

7.4 Testes e resultados

São realizados testes a cada controlador para cada referência, começando pela entrada em degrau. Como se observa na Figura 7.5 é definido o controlo, neste caso o controlador neuronal de modelo inverso, de seguida escolhida a referência em degrau, e por último ativado o controlo começando assim o micro-controlador a realizar o controlo do sistema e o envio da posição a cada 0,15 segundos. De seguida os dados recebidos no monitor série são processados utilizando o MATLAB de modo a obter os gráficos com os diferentes controladores sobrepostos.

```
Controlador Neuronal de Modelo Inverso
Referencia em degrau
ON
3.537
3.311
2.563
2.607
2.589
2.572
2.585
2.546
```

Figura 7.5: Monitor série com informação

O processo é realizado para cada controlador, obtendo os resultados da Figura 7.6 para as respostas ao degrau de 2,3 cm.

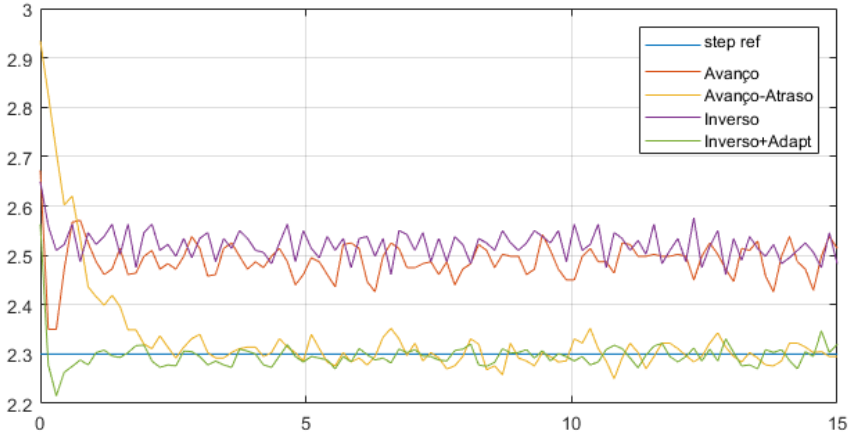


Figura 7.6: Respostas ao degrau

Ao comparar as respostas de cada controlador, é possível averiguar que o controlador em avanço-atraso e o controlador neuronal de modelo inverso com bloco adaptativo possuem uma resposta sem erro em regime permanente, sendo que o sistema varia ligeiramente em torno do valor de referência. É também possível observar que os controladores neuronais possuem um menor tempo de estabelecimento.

No teste com a referência de onda quadrada da Figura 7.7, apenas é possível controlar o sistema com os controladores neuronais. Os controladores clássicos foram projetados para o sistema linearizado em torno de um ponto de equilíbrio, assim quando a referência mudava para os 3 cm o controlador não era capaz de calcular o valor correto a aplicar, causando com que o íman permanente deixasse de ser atraído o suficiente para contrariar a gravidade, acabando por cair.

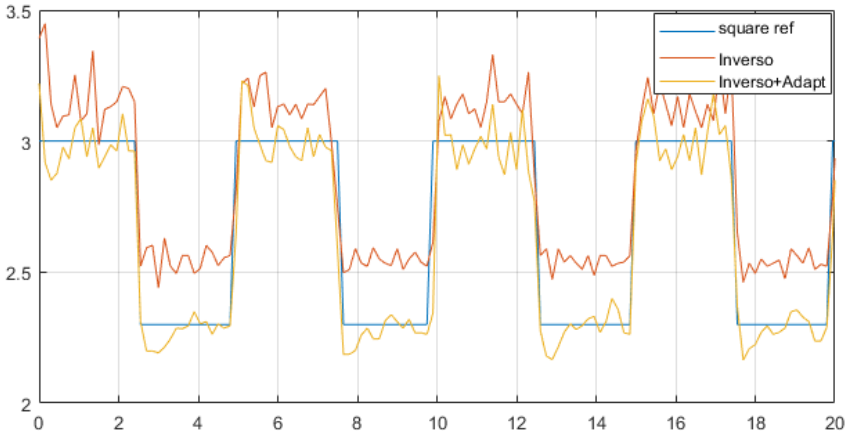


Figura 7.7: Respostas à onda quadrada

Em relação aos dois controladores neuronais, como é de esperar o controla-

dor com o bloco adaptativo consegue seguir melhor a forma de onda, possuindo um menor erro em relação à referência. No entanto o controlador com bloco adaptativo possuí um ligeiro *overshoot* sempre que existe a alteração do valor de referência.

Ao testar a referência em dente de serra, é encontrada a mesma situação em que os controladores clássicos não conseguem seguir a referência a partir de um certo valor. Na Figura 7.8 encontram-se as respostas dos controladores neuronais.

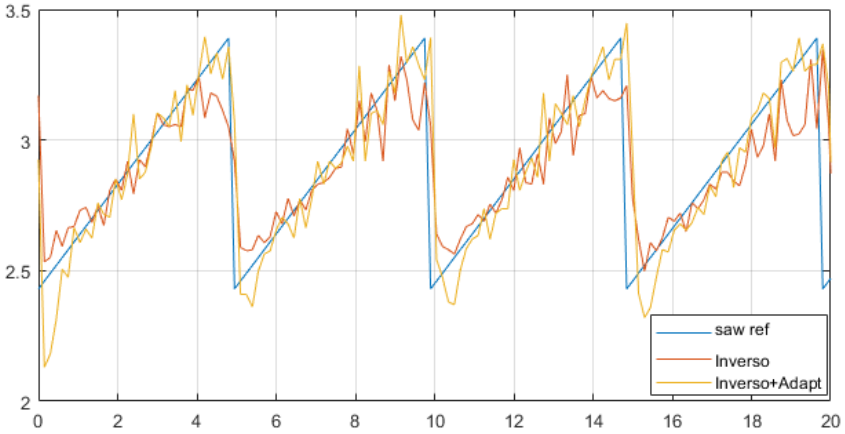


Figura 7.8: Respostas ao dente de serra

Novamente, o controlador com o bloco adaptativo obtém uma resposta melhor, seguindo a referência na totalidade, enquanto que o controlador neuronal sem bloco adaptativo não segue corretamente nas extremidades do sinal.

Resumidamente, com estes testes foi possível verificar que com os controladores neuronais é possível controlar o sistema em amplitudes maiores, mesmo sendo este um sistema não linear. Este facto é comprovado pelos controladores clássicos não conseguirem levitar o íman permanente caso o sistema saísse do ponto de funcionamento para o qual os controladores foram projetados. Ou seja, o modelo inverso do sistema consegue modelar corretamente a dinâmica inversa do sistema, existindo sempre alguns desvios face à entrada de referência.

O controlador de modelo inverso com bloco adaptativo demonstra como é possível obter um controlar robusto para um sistema altamente instável e não linear. Além do mais é possível implementar este controlador num micro-controlador com capacidades computacionais mais reduzidas em relação a opções mais dispendiosas.

Capítulo 8

Considerações finais

Ao longo do presente capítulo são apresentadas as conclusões que permitem apoiar as escolhas de desenvolvimento realizadas ao longo do projeto. É ainda efetuada uma sinopse das consequências mais notáveis e relevância do trabalho, terminando com propostas de desenvolvimentos futuros.

8.1 Conclusões

No início desta dissertação foram estudados diferentes conceitos importantes à realização da mesma, incluindo os fenómenos eletromagnéticos que provocam levitação magnética, diferentes tipos de levitação e as suas principais diferenças. Foram também estudados os principais conceitos de redes neuronais, desde a sua origem até aos tempos correntes, sendo focado em especial os diferentes tipos de controladores neuronais utilizados.

Após esse estudo foi projetado o sistema a utilizar, primeiro foi desenhado um diagrama de blocos genérico do sistema e de seguida explicada a função de cada bloco, sendo deste modo obtido um melhor entendimento de cada parte do sistema desenvolvido. Com cada componente do sistema analisado foi construído o protótipo final com o sistema de levitação magnética, circuito de potência, sensores e micro-controlador, neste caso a placa Arduino Mega 2560 com um ATmega2560 de 8 bits da Microchip.

Para poder implementar os controladores neuronais pretendidos foi primeiro necessário projetar controladores que conseguissem manter o sistema num funcionamento estável e ao mesmo tempo aplicar um sinal de controlo aleatório, isto é com amplitudes e frequências aleatórias, para assim conseguir obter dados de treino para treinar as redes neuronais. Isto deveu-se ao facto do sistema de levitação magnética ser extremamente instável e não linear.

De modo a contornar a não linearidade do sistema, a sua modelação foi realizada em torno de um ponto de funcionamento, para uma corrente e posição de equilíbrio, obtendo no fim a sua função de transferência que descrevia a dinâmica da distância do disco magnético ao sensor em função da tensão aplicada. São também obtidas as equações da tensão que os sensores produzem mediante a posição do disco magnético e a corrente que percorre o eletroíman.

Antes do desenvolvimento dos controladores clássicos foi parametrizado o sistema, nomeadamente as constantes das equações da tensão dos sensores. Para isso foi medida a tensão de repouso dos sensores, de seguida foi analisada a influência que pequenos incrementos de corrente causam na tensão de ambos os sensores, verificando que o sensor de posição é mais sensível à variação de corrente. Para o sensor de posição foi ainda necessário verificar a influência da distância, sendo diminuída a distância entre o disco e sensor gradualmente obtendo uma relação inversa entre a distância e a tensão, o que era de esperar. De modo a verificar e afinar as constantes obtidas, foi utilizado a ferramenta Solver do Microsoft Excel para otimizar os seus valores.

Com o sistema modelado e parametrizado foi possível desenvolver e testar os controladores clássicos. Foi primeiro projetado um controlador em avanço sendo utilizada a condição de fase e módulo para obter o seu pólo e ganho, respetivamente. Ao simular este controlador, este mostrou ser capaz de controlar o sistema de maneira estável possuindo no entanto um erro em regime permanente. Assim, de modo a tentar corrigir esse problema foi desenvolvido um controlador em avanço-atraso que, graças à técnica utilizada de adicionar, ao controlador em avanço, um controlador em atraso com as suas raízes muito próximas de 1 no plano discreto foi obtido um controlador com a resposta transitória semelhante ao controlador em avanço mas com um erro em regime permanente reduzido drasticamente.

Testando no sistema real, os resultados foram de acordo com os esperados, ambos os controladores conseguirem controlar corretamente o sistema, sendo que o controlador em avanço-atraso não possuía erro em regime permanente para a entrada em degrau utilizada. De notar que para testar no sistema real foi necessário acrescentar a tensão de equilíbrio ao sinal de controlo calculado pelos controladores, pois estes foram projetados para o sistema linearizado em torno do ponto de equilíbrio. Com os controladores clássicos desenvolvidos, foi possível obter dados de treino para as redes neuronais pretendidas.

Primeiro foram criadas e treinadas diversas redes de identificação do sistema, ou seja emulavam o seu comportamento. Estas redes de duas camadas, a primeira com função de ativação tangente hiperbólica e a segunda com apenas um neurónio de função linear, variavam entre si devido ao número de neurónios na primeira camada e número de *delays* do sinal de controlo e saída do sistema. Ao testar

as redes para diferentes entradas, foi concluído que a rede com 10 *delays* e 10 neurónios na primeira camada (10d10n) obtinha as melhores respostas sem sobreajustar, sendo esta a rede utilizada como modelo dos sistemas para todas as simulações futuras.

Quanto aos controladores neuronais, foram treinadas redes de modo a aprender a emular a dinâmica inversa do sistema, daí o controlador ser chamado de modelo inverso. Assim, inicialmente foram treinadas redes com *delays* do sinal de controlo e duas camadas, a primeira com função de ativação sigmóide e a segunda com um neurónio de função ReLU, mas mesmo a rede com 5 neurónios e 5 *delays* com o melhor desempenho apresentou algumas imperfeições no controlo do sistema. Para melhorar essas imperfeições foram criadas e treinadas redes com três camadas, a primeira constituída por apenas 1 neurónio com função de ativação tangente hiperbólica, a segunda constituída por um número variável de neurónios com função sigmóide e a última com um neurónio de função ReLU. Estas redes obtiveram resultados melhores, sendo de destacar a rede com 5 neurónios na segunda camada e 5 *delays* que, ao testar no sistema real, obteve resultados satisfatórios pois conseguiu controlar o sistema numa grande amplitude de valores para diferentes sinais de referência. As respostas possuíam um pequeno tempo de subida e baixo *overshoot*, no entanto eram caracterizadas por terem um ligeiro erro em regime permanente.

De modo a corrigir os problemas do controlador de modelo inverso, foi adicionado um bloco adaptativo ao controlador, bloco este que consiste num integrador que soma o erro entre a referência e a posição atual. Ao testar o novo controlador foram obtidas respostas do sistema quase sem erro em regime permanente, provando que um controlador neuronal de modelo inverso com um integrador adicional mostra ser uma configuração viável para um sistema não linear. Foi também construído um controlador neuronal de modelo interno utilizando o modelo inverso do sistema (1th5s5d) e o modelo identificado (10d10n), sendo que apenas foi possível simular este controlador devido a limitações de *hardware*. No entanto verificou-se que nas simulações, este controlador conseguia reduzir o ruído no desempenho do sistema e quando agregado a um bloco adaptativo conseguia fazer o sistema seguir a posição de referência quase sem erro em regime permanente e sem o ruído aleatório que o controlador inverso com bloco adaptativo possuía.

O último controlador neuronal desenvolvido tratou-se do controlador de modelo de referência, sendo este treinado de forma diferente em relação aos anteriores. Primeiramente, é necessária uma rede de identificação, para cumprir este requisito foi utilizada a rede 10d10n, que possuía o melhor desempenho das redes de identificação desenvolvidas. De seguida foram copiados os valores dos pesos sinápticos e polaridades da rede 10d10n nas duas últimas camadas de uma rede de cinco camadas, onde as primeiras três camadas correspondem ao controlador.

Como entrada este controlador usou amostras nos cinco instantes anteriores do sinal de controlo, posição do sistema e referência, sendo a posição realimentada da saída da quinta camada e o sinal de controlo realimentado da saída da terceira camada. Esta rede de cinco camadas foi configurada de modo a desativar a aprendizagem nas duas últimas camadas, mantendo assim o sistema constante, sendo que os pesos e polaridades do controlador (primeiras três camadas) variaram de modo ao sistema sistema seguir a resposta do modelo de referência. O modelo utilizado foi a resposta do sistema real controlado pelo controlador de modelo inverso com bloco adaptativo para a referência aleatória. Após treinar a rede, o controlador conseguiu controlar o sistema com uma resposta bastante satisfatória, mantendo o sistema sem erro em regime permanente, sem ser necessário um bloco adaptativo.

Com os controladores todos desenvolvidos e testados, foi implementado um algoritmo de controlo no micro-controlador definido no projeto do sistema (ATmega2560) na linguagem C, realizando as configurações necessárias para que este comunique de forma bidirecional com o computador, efetue a conversão da tensão dos sensores e calcule o *duty cycle* do sinal de PWM a aplicar no MOSFET de modo a controlar a posição do disco magnético. Nos testes realizados foi possível aferir as capacidades de controlo dos controladores neuronais em comparação com os controladores clássicos. Os controladores neuronais de modelo inverso possuem um menor tempo de estabelecimento que os controladores clássicos, e no caso destes últimos, quando a posição de referência distanciava-se muito do valor de equilíbrio (na referência de onda quadrada e em dente de serra) os controladores clássicos deixavam de conseguir manter o sistema em levitação estável. Ao contrário os controladores neuronais, após o seu treino são, capazes de controlar os sistemas seguindo sinais de referência com grandes amplitudes, provando que as redes neuronais de pequenas dimensões podem ser controladores robustos capazes de modelarem a dinâmica inversa de sistemas não lineares e controlar os ditos sistemas.

8.2 Desenvolvimentos futuros

Com o intuito de melhorar os resultados obtidos e a qualidade do sistema desenvolvido, aconselha-se a implementação física do sistema com recurso a uma placa de circuito impresso, deste modo era reduzido o tempo a verificar as ligações entre componentes do sistema. Outra melhoria possível, seria a alteração do micro-controlador utilizado para um com maiores capacidades computacionais, pois com uma frequência de relógio maior e mais memória seria possível implementar controladores neuronais mais complexos e analisar as suas características. No entanto esta alteração necessitaria de ser bem estudada pois seria necessário alterar o algoritmo de controlo de modo a este ser compatível.

Referências Bibliográficas

[1] D. Halliday, R. Resnick, and J. Walker, *Fundamentals of Physics*. Willey, 8^{th} ed., 2007. [Quoted on p. 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]

[2] P. A. Tipler and G. Mosca, *Physics for Scientists and Engineers with Modern Physics*. New York, United States of America: W. H. Freeman and Company, 6^{th} ed., 2008. [Quoted on p. 5, 6, 7, 8, 9, 10, 11, 12, 13, 17]

[3] H. Hyung-Suk and K. Dong-Sung, *Magnetic Levitation - Maglev Technology and Applications*, vol. 13. New York, United States of America: Springer, 2016. [Quoted on p. 10, 13, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24]

[4] L. C. A. B. de Pinho, *Materiais Magnéticos e suas Aplicações*. PhD thesis, 2009. [Quoted on p. 13, 14, 15]

[5] C. Heck, *Magnetic Materials and Their Applications.* Hungary: London Butterworths, 1974. [Quoted on p. 13, 14, 15]

[6] A. Devred, *Practical low-temperature superconductors for electromagnets*. CERN Yellow Reports: Monographs, Geneva: CERN, 2004. [Quoted on p. 15, 16]

[7] "World's longest superconductor cable works without a hitch." Disponível em: `https://eandt.theiet.org/content/articles/2014/10/worlds-longest-superconductor-cable-works-without-a-hitch/`, 2014. Acedido: 15/01/2020. [Quoted on p. 16]

[8] "The large hadron collider." Disponível em: `https://home.cern/science/accelerators/large-hadron-collider`, 2020. Acedido: 03/02/2020. [Quoted on p. 16]

[9] B. Jayawant, "Electromagnetic suspension and levitation," *Reports on Progress in Physics*, vol. 44, no. 4, p. 74, 1981. [Quoted on p. 18, 21, 22, 23, 24]

[10] S. Haykin, *Neural Networks and Learning Machines*. Upper Saddle River, New Jersey: Pearson Prentice Hall, 3^{rd} ed., 2009. [Quoted on p. 25, 27, 28, 29, 34]

[11] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús, *Neural Networks Design*. 2^{nd} ed., 2014. [Quoted on p. 26, 27, 28, 29, 30, 32, 33, 34, 35]

[12] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, New Jersey: Pearson Prentice Hall, 3^{rd} ed., 2010. [Quoted on p. 31]

[13] M. T. Hagan and H. B. Demuth, "Neural networks for control," *Proceedings of the American Control Conference*, vol. 3, pp. 1642–1656, 1999. [Quoted on p. 36, 37, 38, 39, 40, 41, 87]

[14] M. T. Hagan, H. B. Demuth, and O. D. E. Jesús, "An introduction to the use of neural networks in control systems," *International Journal of Robust and Nonlinear Control*, vol. 12, no. 11, pp. 959–985, 2002. [Quoted on p. 36, 37, 39, 40, 73]

[15] S. Kajan, "Neural controllers for nonlinear systems in Matlab," pp. 1–10, 2008. [Quoted on p. 36, 37, 85, 87]

[16] A. Microsystems, "*Datasheet* a1324." Disponível em: `https://www.allegromicro.com/en/Products/Sense/Linear-and-Angular-Position/Linear-Position-Sensor-ICs/A1324-5-6`. Acedido: 9/12/2012. [Quoted on p. 45, 55]

[17] A. Microsystems, "*Datasheet* acs711." Disponível em: `https://pdf1.alldatasheet.com/datasheet-pdf/view/22437/STMICROELECTRONICS/L298.html`. Acedido: 22/5/2020. [Quoted on p. 46, 55]

[18] "Pololu - acs711ex current sensor carrier -15.5a to +15.5a." Disponível em: `https://www.pololu.com/product/2452`, 2020. Acedido: 13/04/2020. [Quoted on p. 46, 47]

[19] STMicroelectronics, "*Datasheet* lm7805cv." Disponível em: `https://pdf1.alldatasheet.com/datasheet-pdf/view/22634/STMICROELECTRONICS/L7805CV.html`. Acedido: 22/5/2020. [Quoted on p. 48]

[20] Z. LLC, "Electromagnetic Levitation System," tech. rep., Zeltom LLC, Belleville USA, 2009. [Quoted on p. 52]

[21] J. H. M. Myung - Gon Yoon, "A Simple Analog Controller for a Magnetic Levitation Kit," *International Journal of Engineering Research and Technology (IJERT)*, vol. 5, no. 3, pp. 94–97, 2016. [Quoted on p. 52]

[22] K. Ogata, *System Dynamics*. Upper Saddle River, New Jersey: Prentice Hall Ptr, 4^{th} ed., 2004. [Quoted on p. 53]

[23] K. Ogata, *Modern Control Engineering*. Upper Saddle River, New Jersey: Prentice Hall, 5^{th} ed., 2010. [Quoted on p. 62, 64, 65, 67]

[24] William, J. Palm III, *System Dynamics*. New York: The McGraw-Hill, 3^{rd} ed., 2010. [Quoted on p. 62, 64, 65]

[25] G. Beale, "Classical systems and control theory - compensator design to improve steady-state error using root locus." Disponível em: `https://ece.gmu.edu/~gbeale/ece_421/comp_root_ess.pdf`. Acedido: 25/05/2020. [Quoted on p. 67]

[26] A. H. Jafari and M. T. Hagan, "Application of new training methods for neural model reference control," *Engineering Applications of Artificial Intelligence*, vol. 74, no. June, pp. 312–321, 2018. [Quoted on p. 75]

Anexo A

Circuito e placa de circuito impresso

Para desenhar o circuito da Figura A.1 e a PCB da Figura A.2 foi utilizado o programa EasyEDA. Os componentes representados por J são terminais de parafuso com o objetivo de ligar os fios externos.

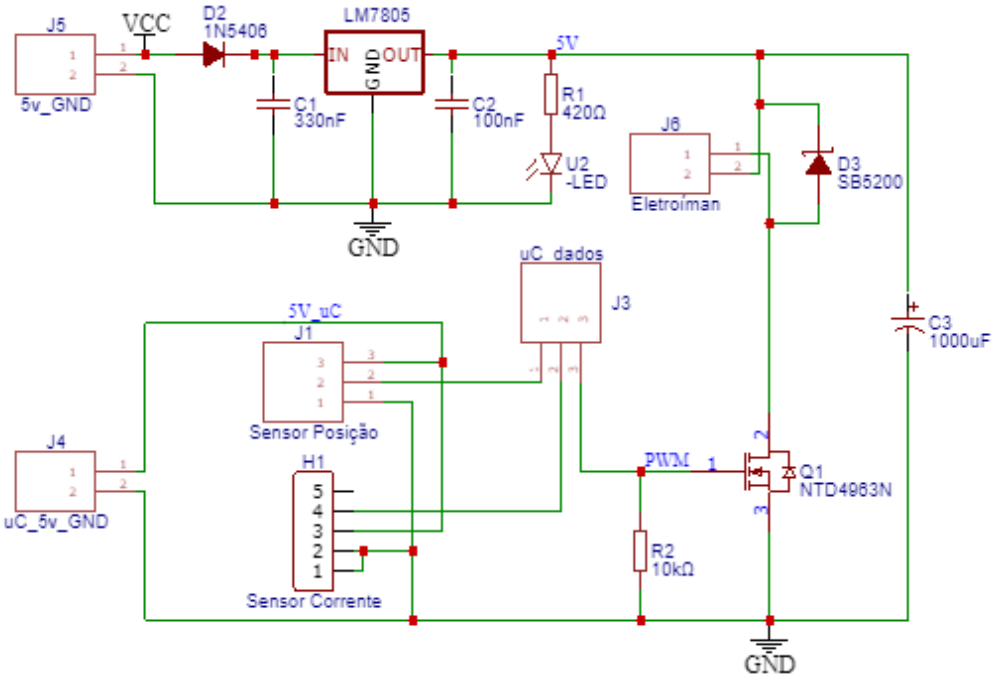


Figura A.1: Circuito eletrónico desenhado

Neste caso os componentes seriam colocados na parte superior da placa e as

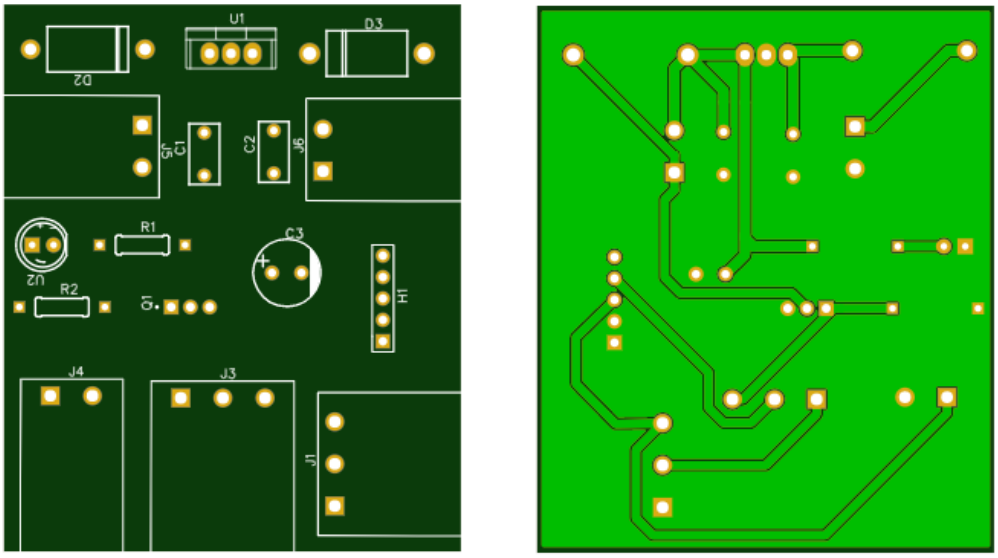


Figura A.2: PCB projetada: (a) vista superior, (b) vista inferior

pistas de cobre na parte inferior da placa. Como se trata de uma placa simples com uma só face de cobre, é possível ver que o *ground* ocupa maior parte do cobre, isto para reduzir ruídos.

Anexo B

Influência da corrente no sensores de efeito Hall

Tabela B.1: Influência da corrente nos sensores

PWM	V_{PWM}(V)	I_{rms}(A)	I(A)	ADC_i	e_i(V)	ADC_z	e_z(V)
0	0,0000	0,00	0,00	0	0,0000	0	0,0000
5	0,0980	0,00	0,03	1	0,0049	0	0,0000
10	0,1961	0,01	0,07	1	0,0049	1	0,0049
15	0,2941	0,03	0,10	1	0,0049	1	0,0049
20	0,3922	0,06	0,14	3	0,0147	5	0,0244
25	0,4902	0,09	0,17	3	0,0147	7	0,0342
30	0,5882	0,13	0,20	5	0,0244	9	0,0440
35	0,6863	0,16	0,24	6	0,0293	11	0,0538
40	0,7843	0,19	0,27	6	0,0293	11	0,0538
45	0,8824	0,23	0,30	8	0,0391	16	0,0782
50	0,9804	0,26	0,34	8	0,0391	20	0,0978
55	1,0784	0,29	0,37	8	0,0391	22	0,1075
60	1,1765	0,33	0,41	11	0,0538	24	0,1173
65	1,2745	0,36	0,44	11	0,0538	25	0,1222
70	1,3725	0,39	0,47	12	0,0587	25	0,1222
75	1,4706	0,42	0,51	14	0,0684	29	0,1417
80	1,5686	0,46	0,54	14	0,0684	34	0,1662
85	1,6667	0,49	0,57	16	0,0782	35	0,1711
90	1,7647	0,52	0,61	16	0,0782	36	0,1760
95	1,8627	0,55	0,64	17	0,0831	38	0,1857
100	1,9608	0,58	0,68	17	0,0831	39	0,1906

105	2,0588	0,61	0,71	20	0,0978	45	0,2199
110	2,1569	0,64	0,74	18	0,0880	43	0,2102
115	2,2549	0,67	0,78	21	0,1026	47	0,2297
120	2,3529	0,71	0,81	21	0,1026	50	0,2444
125	2,4510	0,74	0,85	22	0,1075	48	0,2346
130	2,5490	0,76	0,88	24	0,1173	54	0,2639
135	2,6471	0,79	0,91	23	0,1124	54	0,2639
140	2,7451	0,82	0,95	24	0,1173	55	0,2688
145	2,8431	0,85	0,98	26	0,1271	62	0,3030
150	2,9412	0,88	1,01	25	0,1222	59	0,2884
155	3,0392	0,92	1,05	28	0,1369	62	0,3030
160	3,1373	0,93	1,08	27	0,1320	65	0,3177
165	3,2353	0,96	1,12	28	0,1369	65	0,3177
170	3,3333	0,98	1,15	31	0,1515	68	0,3324
175	3,4314	1,00	1,18	30	0,1466	67	0,3275
180	3,5294	1,02	1,22	31	0,1515	70	0,3421
185	3,6275	1,08	1,25	33	0,1613	75	0,3666
190	3,7255	1,10	1,28	33	0,1613	73	0,3568
195	3,8235	1,12	1,32	34	0,1662	77	0,3763
200	3,9216	1,15	1,35	34	0,1662	80	0,3910
205	4,0196	1,17	1,39	36	0,1760	80	0,3910
210	4,1176	1,20	1,42	38	0,1857	83	0,4057
215	4,2157	1,23	1,45	35	0,1711	83	0,4057
220	4,3137	1,25	1,49	39	0,1906	85	0,4154
225	4,4118	1,27	1,52	39	0,1906	89	0,4350
230	4,5098	1,30	1,56	38	0,1857	88	0,4301
235	4,6078	1,32	1,59	40	0,1955	91	0,4448
240	4,7059	1,34	1,62	40	0,1955	92	0,4497
245	4,8039	1,36	1,66	41	0,2004	93	0,4545
250	4,9020	1,38	1,69	42	0,2053	95	0,4643
255	5,0000	1,40	1,72	42	0,2053	97	0,4741

Anexo C

Código MATLAB utilizado para modelar o sistema e projetar os controladores clássicos

```
%% Modelacao e simulacao do sistema de levitacao%%%%%%%%%%%%%%
%% Parametros do sistema
zeq=2                %distancia ao iman em centimetros
ieq=0.27             %corrente necessaria em amperes
g=981                %aceleracao gravitica cm/s^2
R=3                  %resistencia do eletroiman em ohms
L=0.015              %indutancia do eletroiman em henrys
T=0.003              %Periodo de amostragem
m=0.003              %massa do disco
k=m*g*(zeq^4)/ieq %constante magnetica


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Funcao de Transferencia %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Gs_tf=tf([-g/ieq],[L R -L*4*g/zeq -R*4*g/zeq])%Sistema continuo
Gs=zpk(Gs_tf)%para mostrar na forma zpk
%dados do sistema
zs=[]
ps=[44.29 -44.29]%polos dominantes
ks=-2.4222e+05
%Retirando o polo no dominante (s=-193.33) necessario
%manter o ganho estatico do sistema
Gs_pd=zpk(zs,ps,ks/(R/L))%Sistema aproximado sem o polo nao dominante
dcgain(Gs)
dcgain(Gs_pd)%comparar os ganhos
```

```
%%%%%Discretizar o sistema%%%%%
Gz=c2d(Gs,T)%Sistema com todos os polos
Gz_pd=c2d(Gs_pd,T)%Sistema sem o polo nao dominante
%dados do sistema
zz=[-3.241 -0.2287]
pz=[0.5488 0.8756 1.142]%polos do sistema ordem crescente
kz=-0.00094519
zz_pd=[-1]
pz_pd=[0.8756 1.142]%polos do sistema ordem crescente
kz_pd=-0.005458


%%%%%Lugar geometrico de raizes dos diferentes sistemas%%%%%
figure
subplot(2,2,1)
rlocus(Gs)%G(s)
axis([-400 400 -150 150])
subplot(2,2,2)
rlocus(Gs_pd)%G_pd(s)
axis([-100 100 -50 50])
subplot(2,2,3)
rlocus(Gz)%G(z)
axis([-2 2 -1.5 1.5])
subplot(2,2,4)
rlocus(Gz_pd)%G_pd(z)
axis([-2 2 -1.5 1.5])



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Controlador em avanco %%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Resposta desejada
MP=0.05             %overshoot menor que 10%
zeta=0.6*(1-MP) %coeficiente de amortecimento
tr=T*10             %tempo de subida (s)
wn=1.8/tr           %frequencia natural (rad/s)
ts=4.6/(zeta*wn) %tempo de estabelecimento (s)

%Polos correspondentes no plano z
r=exp(-zeta*wn*T)
theta=wn*T*sqrt(1-zeta^2)
z2=r*exp(-j*theta)
z1=r*exp(j*theta)

%Alpha-----zero que anula o polo z=0.8756
Alpha_av=pz_pd(1)
%Beta------polo a calcular /D(z)G_pd(z)=-pi
phi1=atan(imag(z1)/(abs(zz_pd)+real(z1)))
theta1=pi-atan(imag(z1)/(pz_pd(2)-real(z1)))
thetaB=pi-theta1+phi1
```

```
Beta_av=real(z1)-imag(z1)/tan(thetaB)
%k-----ganho do controlador |D(z)G_pd(z)|=1 quando z=z1
K_av=(1/kz_pd)*(abs((z1-Beta_av)*(z1-pz_pd(2))))/(abs(z1+abs(zz_pd)))

Dz_av=zpk([Alpha_av],[Beta_av],K_av,T)%Controlador em avano

%Sistema+controlador malha aberta
DGz_av=Dz_av*Gz %Sistema com todos os polos
DGz_av_pd=Dz_av*Gz_pd %Sistema com apenas os polos dominantes


%lugar de raizes  discreto
figure
subplot(1,2,1)
rlocus(DGz_av)%Sistema com todos os polos
hold on
axis([-1 1 -1 1])
plot(real(z1),imag(z1),'s') %Para dar plot aos polos desejados
plot(real(z2),imag(z2),'s') %E possivel ver que os polos desejados sao atingidos
zgrid
subplot(1,2,2)
rlocus(DGz_av_pd)%Sistema com apenas os polos dominantes
hold on
axis([-1 1 -1 1])
plot(real(z1),imag(z1),'s') %Para dar plot aos polos desejados
plot(real(z2),imag(z2),'s') %E  possvel  ver que os polos desejados estao proximos
zgrid


%Sistema em malha fechada
DGz_av_cl=feedback(DGz_av,1)          %Sistema com todos os polos
DGz_av_pd_cl=feedback(DGz_av_pd,1)%Sistema com apenas os polos dominantes

%Resposta ao step
[y1,t1]=step(DGz_av_cl,1.5);
[y2,t2]=step(DGz_av_pd_cl,1.5);
figure
plot(t1,y1,'b','linewidth',1);
hold on
plot(t2,y2,'r','linewidth',1);

ESS=abs(1-y1(end)) %Erro em regime permanente no sistema com todos os polos
ESS=abs(1-y2(end)) %Erro em regime permanente no sistema com apenas os polos dominantes


Ds_av=d2c(Dz_av)          %Controlador em avano continuo
DGs_av=Ds_av*Gs           %Sistema contnuo  com todos os polos
DGs_av_pd=Ds_av*Gs_pd %Sistema com apenas os plos dominantes Gnum/Gden

%Polos no plano s
r=-zeta*wn
```

```
theta=wn*sqrt(1-zeta^2)
s2=r-j*theta
s1=r+j*theta

%lugar de raizes continuo do sistema com o controlador em avano
figure
subplot(1,2,1)
rlocus(DGs_av)
hold on
axis([-300 50 -100 100])
plot(real(s1),imag(s1),'s') %para dar plot aos polos desejados
plot(real(s2),imag(s2),'s') %e possivel ver que os polos desejados sao quase atingidos
subplot(1,2,2)
rlocus(DGs_av_pd)
hold on
axis([-300 50 -100 100])
plot(real(s1),imag(s1),'s') %para dar plot aos polos desejados
plot(real(s2),imag(s2),'s') %e possivel ver que os polos desejados  so  atingidos



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Controlador Avano-Atraso %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ESS_obj=-0.05 %Erro em regime permanente desejado

%racio z/p
%(Gden(0)-Ess_obj*Gden(0))/(Ess_obj*Gnum(0))
racio_zp=(-50334761.21-ESS_obj*-50334761.21)/(ESS_obj*82673808)

z=real(s1)/50
p=z/racio_zp

Ds_at=zpk([z],[p],1)%Controlador em atraso

%sistema em malha aberta
DGs=DGs_av*Ds_at          %Sistema continuo com todos os polos
DGs_pd=DGs_av_pd*Ds_at %Sistema com apenas os polos dominantes Gnum/Gden

%lugar de  razes  continuo do sistema com o controlador em avano-atraso
figure
rlocus(DGs)
hold on
axis([-300 50 -100 100])
plot(real(s1),imag(s1),'s') %para dar plot aos polos desejados
plot(real(s2),imag(s2),'s') %e possivel ver que os polos desejados sao quase atingidos
figure
rlocus(DGs_pd)
hold on
axis([-300 50 -100 100])
plot(real(s1),imag(s1),'s') %para dar plot aos polos desejados
```

```
plot(real(s2),imag(s2),'s') %e possivel ver que os polos desejados sao atingidos


%sistema em malha fechada
DGs_cl=feedback(DGs,1)          %sistema com todos os polos
DGs_av_cl=feedback(DGs_av,1)%sistema com apenas os polos dominantes

[y1,t1]=step(DGs_cl,3);      %resposta ao step
[y2,t2]=step(DGs_av_cl,3); %resposta ao step
figure
plot(t1,y1,'b','linewidth',1);
hold on
plot(t2,y2,'r','linewidth',1);

ESS=abs(1-y1(end)) %Erro em regime permanente no sistema com todos os polos
ESS=abs(1-y2(end)) %Erro em regime permanente no sistema com apenas os polos dominantes


Dz_at=c2d(Ds_at,T)%controlador em atraso discreto
DDz=Dz_at*Dz_av%controlador em avano-atraso discreto
DGz=Dz_at*DGz_av
DGz_pd=Dz_at*DGz_av_pd

%lugar de raizes discreto do sistema com o controlador em avano-atraso
figure
subplot(1,2,1)
rlocus(DGz)%sistema com todos os polos
hold on
axis([0.7 1.3 -0.3 0.3])
plot(real(z1),imag(z1),'s') %para dar plot aos polos desejados
plot(real(z2),imag(z2),'s') %e possivel ver que os polos desejados sao atingidos
zgrid
subplot(1,2,2)
rlocus(DGz_pd)%sistema com apenas os polos dominantes
hold on
axis([0.7 1.3 -0.3 0.3])
plot(real(z1),imag(z1),'s') %para dar plot aos polos desejados
plot(real(z2),imag(z2),'s') %e possivel ver que os polos desejados estao proximos
zgrid
```

Anexo D

Código MATLAB desenvolvido para criar, treinar e simular redes neuronais

```
%%% Identificacao e Controlo Neuronal%%%%%%%%%%%%%%%%%%
%% Dados a utilizar
clear
load Dados_de_treino
%carrega os valores da distancia(z), corrente(i) e sinal de controlo(u)
t=[0:length(u)-1]*T;%cria vetor com os valor de tempo

figure
subplot(2,2,1)
plot(t,z,'b','linewidth',1)%posicao
hold on, grid on
plot(t,ref,'r','linewidth',1)%posicao
xlabel ('Tempo(s)')
ylabel(' Distncia (cm)')
subplot(2,2,2)
plot(t,u,'B','linewidth',1)%u
hold on, grid on
xlabel ('Tempo(s)')
ylabel('Controlo(V)')
subplot(2,2,3)
plot(t,i,'r','linewidth',1)%i
hold on, grid on
xlabel ('Tempo(s)')
ylabel('Corrente(A)')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Identificacao do sistema %%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%Preparacao dos dados de treino%%%%%%%
U=num2cell(u);%Converte os dados para um vetor de celulas standard
Z=num2cell(z);%para o treino de redes neuronais

inputDelays = 1:5; %numero de amostras do sinal de x (n=10)
feedbackDelays = 1:5; %numero de amostras do sinal de x (m=10)
hiddenLayerSize = 5; %5 neuronios na camada escondida
net = narxnet(inputDelays,feedbackDelays,hiddenLayerSize,'open','trainlm');
%cria uma rede y(k+1)=f[x(k),x(k-1),...,x(k-n),y(k),y(k-1),...,y(k-m)]
%open -> para a realimentacao dos valores de saida da rede ser externa
%trainlm -> algoritmo de Levenberg-Marquardt para o treino
view(net)%para ver a rede

[Xs,Xi,Ai,Ts] = preparets(net,U,{},Z);%para redes com 10delays
%cria as matrixes de entrada dos valores de u e z de k a k-n e m
%Xs->Matriz de entrada da rede com os valores de u e z apartir de k=10
%Xi->Matriz dos delays de entrada com os 10 primeiro valores de u e z
%Ai->Matriz dos delays de camada
%Ts->Matriz objetivo com os valores de z apartir de k=10

%funcao de performance
net.performFcn='mse';%Mean Squared Error

%lista de plots
net.plotFcns={'plotperform','plottrainstate', 'plotresponse'};

net.trainparam.goal=5e-5;%Tolerancia do erro;
net.trainparam.epochs=5000;%Numero maximo de iteracoes
net.trainparam.show=50; %Resultado e mostrado a cada 50 iteracoes
net.trainparam.lr=0.05;%Taxa de aprendizagem

[net,tr]=train(net,Xs,Ts,Xi,Ai);%treino da rede
%simula a rede
y=net(Xs,Xi,Ai);
%calcula o vetor de erro para os diferentes pontos
e=gsubtract(Ts,y);
%calcula a performance(MSE) da rede treinada
performance=perform(net,Ts,y)

gensim(net,T) %criao bloco simulink da rede


%% Teste das rede de   identificao
clear
load Redes_ident

load dados_step
%%%%%%%Preparacao dos dados de treino%%%%%%%
U=num2cell(u);%Converte os dados para um vetor de celulas standard
Z=num2cell(z);%para o treino de redes neuronais
```

```
[Xs,Xi,Ai,Ts] = preparets(NARX_10d10n,U,{},Z);%para redes com 10delays
%simula a rede
y=NARX_10d5n(Xs,Xi,Ai);
%calcula a performance da rede treinada
performance_10d5n_step=perform(NARX_10d5n,Ts,y);
y=NARX_10d10n(Xs,Xi,Ai);
performance_10d10n_step=perform(NARX_10d10n,Ts,y);
y=NARX_10d15n(Xs,Xi,Ai);
performance_10d15n_step=perform(NARX_10d15n,Ts,y);

[Xs,Xi,Ai,Ts] = preparets(NARX_15d10n,U,{},Z);%para redes com 10delays
%simula a rede
y=NARX_15d5n(Xs,Xi,Ai);
%calcula a performance da rede treinada
performance_15d5n_step=perform(NARX_15d5n,Ts,y);
y=NARX_15d10n(Xs,Xi,Ai);
performance_15d10n_step=perform(NARX_15d10n,Ts,y);
y=NARX_15d15n(Xs,Xi,Ai);
performance_15d15n_step=perform(NARX_15d15n,Ts,y);

load dados_sine
%%%%%%%Preparacao dos dados de treino%%%%%%%
U=num2cell(u);%Converte os dados para um vetor de celulas standard
Z=num2cell(z);%para o treino de redes neuronais
[Xs,Xi,Ai,Ts] = preparets(NARX_10d10n,U,{},Z);%para redes com 10delays
%simula a rede
y=NARX_10d5n(Xs,Xi,Ai);
%calcula a performance da rede treinada
performance_10d5n_sine=perform(NARX_10d5n,Ts,y);
y=NARX_10d10n(Xs,Xi,Ai);
performance_10d10n_sine=perform(NARX_10d10n,Ts,y);
y=NARX_10d15n(Xs,Xi,Ai);
performance_10d15n_sine=perform(NARX_10d15n,Ts,y);

[Xs,Xi,Ai,Ts] = preparets(NARX_15d10n,U,{},Z);%para redes com 10delays
%simula a rede
y=NARX_15d5n(Xs,Xi,Ai);
%calcula a performance da rede treinada
performance_15d5n_sine=perform(NARX_15d5n,Ts,y);
y=NARX_15d10n(Xs,Xi,Ai);
performance_15d10n_sine=perform(NARX_15d10n,Ts,y);
y=NARX_15d15n(Xs,Xi,Ai);
performance_15d15n_sine=perform(NARX_15d15n,Ts,y);

load dados_saw
%%%%%%%Preparacao dos dados de treino%%%%%%%
U=num2cell(u);%Converte os dados para um vetor de celulas standard
Z=num2cell(z);%para o treino de redes neuronais
[Xs,Xi,Ai,Ts] = preparets(NARX_10d10n,U,{},Z);%para redes com 10delays
%simula a rede
y=NARX_10d5n(Xs,Xi,Ai);
```

```
%calcula a performance da rede treinada
performance_10d5n_saw=perform(NARX_10d5n,Ts,y);
y=NARX_10d10n(Xs,Xi,Ai);
performance_10d10n_saw=perform(NARX_10d10n,Ts,y);
y=NARX_10d15n(Xs,Xi,Ai);
performance_10d15n_saw=perform(NARX_10d15n,Ts,y);

[Xs,Xi,Ai,Ts] = preparets(NARX_15d10n,U,{},Z);%para redes com 15delays
%simula a rede
y=NARX_15d5n(Xs,Xi,Ai);
%calcula a performance da rede treinada
performance_15d5n_saw=perform(NARX_15d5n,Ts,y);
y=NARX_15d10n(Xs,Xi,Ai);
performance_15d10n_saw=perform(NARX_15d10n,Ts,y);
y=NARX_15d15n(Xs,Xi,Ai);
performance_15d15n_saw=perform(NARX_15d15n,Ts,y);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Controlo do sistema por modelo inverso %%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
load Dados_de_treino

N = length(u);
%k comeca em 11 para poder adicionar 10 amostras anteriores dos sinais
for k = 11:(N-1)
    NNinput(1,k)=u(k-1)/5;%O sinal de controlo anterior
    NNinput(2,k)=u(k-2)/5;%O sinal de controlo no instante k-1
    NNinput(3,k)=u(k-3)/5;%O sinal de controlo no instante k-2
    NNinput(4,k)=u(k-4)/5;%O sinal de controlo no instante k-3
    NNinput(5,k)=u(k-5)/5;%O sinal de controlo no instante k-4
    NNinput(6,k)=u(k-6)/5;%O sinal de controlo no instante k-5
    NNinput(7,k)=u(k-7)/5;%O sinal de controlo no instante k-6
    NNinput(8,k)=u(k-8)/5;%O sinal de controlo no instante k-7
    NNinput(9,k)=u(k-9)/5;%O sinal de controlo no instante k-9
    NNinput(10,k)=u(k-10)/5;%O sinal de controlo no instante k-10
    NNinput(11,k)=ref(k+1)/4;%A distancia pretendida
    NNinput(12,k)=z(k)/4;%A distancia atual em k
    NNinput(13,k)=z(k-1)/4;%A distancia em k-1
    NNinput(14,k)=z(k-2)/4;%A distancia em k-2
    NNinput(15,k)=z(k-3)/4;%A distancia em k-3
    NNinput(16,k)=z(k-4)/4;%A distancia em k-4
    NNinput(17,k)=z(k-5)/4;%A distancia em k-5
    NNinput(18,k)=z(k-6)/4;%A distancia em k-6
    NNinput(19,k)=z(k-7)/4;%A distancia em k-7
    NNinput(20,k)=z(k-8)/4;%A distancia em k-8
    NNinput(21,k)=z(k-9)/4;%A distancia em k-9
    NNoutput(k)=u(k)/5;%O sinal de controlo pretendido
end
```

```
net = newff(minmax(NNinput),[2 5 1],{'tansig','logsig','poslin'});
net.trainFcn='trainlm';%algoritmo de treino
net.plotFcns={'plotperform','plotresponse','plottrainstate'}%lista de plots
%view(net)

net.trainparam.epochs=500;%Numero maximo de iteracoes
net.trainparam.min_grad=1e-10;%Numero gradiente minimo
% Treino da RN
[net,tr] = train(net,NNinput,NNoutput);

unn = sim(net,NNinput);%simula a rede

figure
plot(u)
hold on
plot(unn)

gensim(net,0.003)


%% Teste das redes de controlo de modelo inverso
clear
load Redes_control

load dados_step
%  preparao
NNinput=0;
NNoutput=0;
N = length(u);
%k comeca em 6 para poder adicionar 5 amostras anteriores dos sinais
%(Divisoes realizadas para colocar os dados entre [-1 e 1])
for k = 6:(N-10)
    NNinput(1,k)=u(k-1)/5;%O sinal de controlo anterior
    NNinput(2,k)=u(k-2)/5;%O sinal de controlo no instante k-1
    NNinput(3,k)=u(k-3)/5;%O sinal de controlo no instante k-2
    NNinput(4,k)=u(k-4)/5;%O sinal de controlo no instante k-3
    NNinput(5,k)=u(k-5)/5;%O sinal de controlo no instante k-4
    NNinput(6,k)=ref(k+1)/4;%A distancia pretendida
    NNinput(7,k)=z(k)/4;%A distancia atual
    NNinput(8,k)=z(k-1)/4;%A distancia em k-1
    NNinput(9,k)=z(k-2)/4;%A distancia em k-2
    NNinput(10,k)=z(k-3)/4;%A distancia em k-3
    NNinput(11,k)=z(k-4)/4;%A distancia em k-4
    NNoutput(k)=u(k)/5;%O sinal de controlo pretendido
end

%simula a rede
unn = sim(CN_5S_5D,NNinput);
%calcula a performance da rede treinada
performance_5s5d_step=perform(CN_5S_5D,NNoutput,unn);
unn = sim(CN_10S_5D,NNinput);
```

```
performance_10s5d_step=perform(CN_10S_5D,NNoutput,unn);
unn = sim(CN_15S_5D,NNinput);
performance_15s5d_step=perform(CN_15S_5D,NNoutput,unn);
unn = sim(CN_1TH_5S_5D,NNinput);
performance_1th5s5d_step=perform(CN_1TH_5S_5D,NNoutput,unn);
unn = sim(CN_1TH_10S_5D,NNinput);
performance_1th10s5d_step=perform(CN_1TH_10S_5D,NNoutput,unn);
unn = sim(CN_1TH_15S_5D,NNinput);
performance_1th15s5d_step=perform(CN_1TH_15S_5D,NNoutput,unn);

%k comeca em 11 para poder adicionar 10 amostras anteriores dos sinais
for k = 11:(N-1)
    NNinput(1,k)=u(k-1)/5;%O sinal de controlo anterior
    NNinput(2,k)=u(k-2)/5;%O sinal de controlo no instante k-1
    NNinput(3,k)=u(k-3)/5;%O sinal de controlo no instante k-2
    NNinput(4,k)=u(k-4)/5;%O sinal de controlo no instante k-3
    NNinput(5,k)=u(k-5)/5;%O sinal de controlo no instante k-4
    NNinput(6,k)=u(k-6)/5;%O sinal de controlo no instante k-5
    NNinput(7,k)=u(k-7)/5;%O sinal de controlo no instante k-6
    NNinput(8,k)=u(k-8)/5;%O sinal de controlo no instante k-7
    NNinput(9,k)=u(k-9)/5;%O sinal de controlo no instante k-9
    NNinput(10,k)=u(k-10)/5;%O sinal de controlo no instante k-10
    NNinput(11,k)=ref(k+1)/4;%A distancia pretendida
    NNinput(12,k)=z(k)/4;%A distancia atual em k
    NNinput(13,k)=z(k-1)/4;%A distancia em k-1
    NNinput(14,k)=z(k-2)/4;%A distancia em k-2
    NNinput(15,k)=z(k-3)/4;%A distancia em k-3
    NNinput(16,k)=z(k-4)/4;%A distancia em k-4
    NNinput(17,k)=z(k-5)/4;%A distancia em k-5
    NNinput(18,k)=z(k-6)/4;%A distancia em k-6
    NNinput(19,k)=z(k-7)/4;%A distancia em k-7
    NNinput(20,k)=z(k-8)/4;%A distancia em k-8
    NNinput(21,k)=z(k-9)/4;%A distancia em k-9
    NNoutput(k)=u(k)/5;%O sinal de controlo pretendido
end

%simula a rede
unn = sim(CN_5S_10D,NNinput);
%calcula a performance da rede treinada
performance_5s10d_step=perform(CN_5S_10D,NNoutput,unn);
unn = sim(CN_10S_10D,NNinput);
performance_10s10d_step=perform(CN_10S_10D,NNoutput,unn);
unn = sim(CN_15S_10D,NNinput);
performance_15s10d_step=perform(CN_15S_10D,NNoutput,unn);
unn = sim(CN_1TH_5S_10D,NNinput);
performance_1th5s10d_step=perform(CN_1TH_5S_10D,NNoutput,unn);
unn = sim(CN_1TH_10S_10D,NNinput);
performance_1th10s10d_step=perform(CN_1TH_10S_10D,NNoutput,unn);
unn = sim(CN_1TH_15S_10D,NNinput);
performance_1th15s10d_step=perform(CN_1TH_15S_10D,NNoutput,unn);
```

```
load dados_sine
% preparao
NNinput=0;
NNoutput=0;
N = length(u);
for k = 6:(N-10)
    NNinput(1,k)=u(k-1)/5;%O sinal de controlo anterior
    NNinput(2,k)=u(k-2)/5;%O sinal de controlo no instante k-1
    NNinput(3,k)=u(k-3)/5;%O sinal de controlo no instante k-2
    NNinput(4,k)=u(k-4)/5;%O sinal de controlo no instante k-3
    NNinput(5,k)=u(k-5)/5;%O sinal de controlo no instante k-4
    NNinput(6,k)=ref(k+1)/4;%A distancia pretendida
    NNinput(7,k)=z(k)/4;%A distancia atual
    NNinput(8,k)=z(k-1)/4;%A distancia em k-1
    NNinput(9,k)=z(k-2)/4;%A distancia em k-2
    NNinput(10,k)=z(k-3)/4;%A distancia em k-3
    NNinput(11,k)=z(k-4)/4;%A distancia em k-4
    NNoutput(k)=u(k)/5;%O sinal de controlo pretendido
end

%simula a rede
unn = sim(CN_5S_5D,NNinput);
%calcula a performance da rede treinada
performance_5s5d_sine=perform(CN_5S_5D,NNoutput,unn);
unn = sim(CN_10S_5D,NNinput);
performance_10s5d_sine=perform(CN_10S_5D,NNoutput,unn);
unn = sim(CN_15S_5D,NNinput);
performance_15s5d_sine=perform(CN_15S_5D,NNoutput,unn);
unn = sim(CN_1TH_5S_5D,NNinput);
performance_1th5s5d_sine=perform(CN_1TH_5S_5D,NNoutput,unn);
unn = sim(CN_1TH_10S_5D,NNinput);
performance_1th10s5d_sine=perform(CN_1TH_10S_5D,NNoutput,unn);
unn = sim(CN_1TH_15S_5D,NNinput);
performance_1th15s5d_sine=perform(CN_1TH_15S_5D,NNoutput,unn);

for k = 11:(N-1)
    NNinput(1,k)=u(k-1)/5;%O sinal de controlo anterior
    NNinput(2,k)=u(k-2)/5;%O sinal de controlo no instante k-1
    NNinput(3,k)=u(k-3)/5;%O sinal de controlo no instante k-2
    NNinput(4,k)=u(k-4)/5;%O sinal de controlo no instante k-3
    NNinput(5,k)=u(k-5)/5;%O sinal de controlo no instante k-4
    NNinput(6,k)=u(k-6)/5;%O sinal de controlo no instante k-5
    NNinput(7,k)=u(k-7)/5;%O sinal de controlo no instante k-6
    NNinput(8,k)=u(k-8)/5;%O sinal de controlo no instante k-7
    NNinput(9,k)=u(k-9)/5;%O sinal de controlo no instante k-9
    NNinput(10,k)=u(k-10)/5;%O sinal de controlo no instante k-10
    NNinput(11,k)=ref(k+1)/4;%A distancia pretendida
    NNinput(12,k)=z(k)/4;%A distancia atual em k
    NNinput(13,k)=z(k-1)/4;%A distancia em k-1
    NNinput(14,k)=z(k-2)/4;%A distancia em k-2
    NNinput(15,k)=z(k-3)/4;%A distancia em k-3
```

```
    NNinput(16,k)=z(k-4)/4;%A distancia em k-4
    NNinput(17,k)=z(k-5)/4;%A distancia em k-5
    NNinput(18,k)=z(k-6)/4;%A distancia em k-6
    NNinput(19,k)=z(k-7)/4;%A distancia em k-7
    NNinput(20,k)=z(k-8)/4;%A distancia em k-8
    NNinput(21,k)=z(k-9)/4;%A distancia em k-9
    NNoutput(k)=u(k)/5;%O sinal de controlo pretendido
end

%simula a rede
unn = sim(CN_5S_10D,NNinput);
%calcula a performance da rede treinada
performance_5s10d_sine=perform(CN_5S_10D,NNoutput,unn);
unn = sim(CN_10S_10D,NNinput);
performance_10s10d_sine=perform(CN_10S_10D,NNoutput,unn);
unn = sim(CN_15S_10D,NNinput);
performance_15s10d_sine=perform(CN_15S_10D,NNoutput,unn);
unn = sim(CN_1TH_5S_10D,NNinput);
performance_1th5s10d_sine=perform(CN_1TH_5S_10D,NNoutput,unn);
unn = sim(CN_1TH_10S_10D,NNinput);
performance_1th10s10d_sine=perform(CN_1TH_10S_10D,NNoutput,unn);
unn = sim(CN_1TH_15S_10D,NNinput);
performance_1th15s10d_sine=perform(CN_1TH_15S_10D,NNoutput,unn);

load dados_saw
%  preparao
NNinput=0;
NNoutput=0;
N = length(u);
for k = 6:(N-10)
    NNinput(1,k)=u(k-1)/5;%O sinal de controlo anterior
    NNinput(2,k)=u(k-2)/5;%O sinal de controlo no instante k-1
    NNinput(3,k)=u(k-3)/5;%O sinal de controlo no instante k-2
    NNinput(4,k)=u(k-4)/5;%O sinal de controlo no instante k-3
    NNinput(5,k)=u(k-5)/5;%O sinal de controlo no instante k-4
    NNinput(6,k)=ref(k+1)/4;%A distancia pretendida
    NNinput(7,k)=z(k)/4;%A distancia atual
    NNinput(8,k)=z(k-1)/4;%A distancia em k-1
    NNinput(9,k)=z(k-2)/4;%A distancia em k-2
    NNinput(10,k)=z(k-3)/4;%A distancia em k-3
    NNinput(11,k)=z(k-4)/4;%A distancia em k-4
    NNoutput(k)=u(k)/5;%O sinal de controlo pretendido
end

%simula a rede
unn = sim(CN_5S_5D,NNinput);
%calcula a performance da rede treinada
performance_5s5d_saw=perform(CN_5S_5D,NNoutput,unn);
unn = sim(CN_10S_5D,NNinput);
performance_10s5d_saw=perform(CN_10S_5D,NNoutput,unn);
unn = sim(CN_15S_5D,NNinput);
```

```
performance_15s5d_saw=perform(CN_15S_5D,NNoutput,unn);
unn = sim(CN_1TH_5S_5D,NNinput);
performance_1th5s5d_saw=perform(CN_1TH_5S_5D,NNoutput,unn);
unn = sim(CN_1TH_10S_5D,NNinput);
performance_1th10s5d_saw=perform(CN_1TH_10S_5D,NNoutput,unn);
unn = sim(CN_1TH_15S_5D,NNinput);
performance_1th15s5d_saw=perform(CN_1TH_15S_5D,NNoutput,unn);

for k = 11:(N-1)
    NNinput(1,k)=u(k-1)/5;%O sinal de controlo anterior
    NNinput(2,k)=u(k-2)/5;%O sinal de controlo no instante k-1
    NNinput(3,k)=u(k-3)/5;%O sinal de controlo no instante k-2
    NNinput(4,k)=u(k-4)/5;%O sinal de controlo no instante k-3
    NNinput(5,k)=u(k-5)/5;%O sinal de controlo no instante k-4
    NNinput(6,k)=u(k-6)/5;%O sinal de controlo no instante k-5
    NNinput(7,k)=u(k-7)/5;%O sinal de controlo no instante k-6
    NNinput(8,k)=u(k-8)/5;%O sinal de controlo no instante k-7
    NNinput(9,k)=u(k-9)/5;%O sinal de controlo no instante k-9
    NNinput(10,k)=u(k-10)/5;%O sinal de controlo no instante k-10
    NNinput(11,k)=ref(k+1)/4;%A distancia pretendida
    NNinput(12,k)=z(k)/4;%A distancia atual em k
    NNinput(13,k)=z(k-1)/4;%A distancia em k-1
    NNinput(14,k)=z(k-2)/4;%A distancia em k-2
    NNinput(15,k)=z(k-3)/4;%A distancia em k-3
    NNinput(16,k)=z(k-4)/4;%A distancia em k-4
    NNinput(17,k)=z(k-5)/4;%A distancia em k-5
    NNinput(18,k)=z(k-6)/4;%A distancia em k-6
    NNinput(19,k)=z(k-7)/4;%A distancia em k-7
    NNinput(20,k)=z(k-8)/4;%A distancia em k-8
    NNinput(21,k)=z(k-9)/4;%A distancia em k-9
    NNoutput(k)=u(k)/5;%O sinal de controlo pretendido
end

%simula a rede
unn = sim(CN_5S_10D,NNinput);
%calcula a performance da rede treinada
performance_5s10d_saw=perform(CN_5S_10D,NNoutput,unn);
unn = sim(CN_10S_10D,NNinput);
performance_10s10d_saw=perform(CN_10S_10D,NNoutput,unn);
unn = sim(CN_15S_10D,NNinput);
performance_15s10d_saw=perform(CN_15S_10D,NNoutput,unn);
unn = sim(CN_1TH_5S_10D,NNinput);
performance_1th5s10d_saw=perform(CN_1TH_5S_10D,NNoutput,unn);
unn = sim(CN_1TH_10S_10D,NNinput);
performance_1th10s10d_saw=perform(CN_1TH_10S_10D,NNoutput,unn);
unn = sim(CN_1TH_15S_10D,NNinput);
performance_1th15s10d_saw=perform(CN_1TH_15S_10D,NNoutput,unn);


%% Para copiar quando necessrio  diferentes
%Dados de treino(Divisoes realizadas para colocar os dados entre [-1 e 1])
```

```
%k comeca em 6 para poder adicionar 5 amostras anteriores dos sinais
for k = 6:(N-10)
    NNinput(1,k)=u(k-1)/5;%O sinal de controlo anterior
    NNinput(2,k)=u(k-2)/5;%O sinal de controlo no instante k-1
    NNinput(3,k)=u(k-3)/5;%O sinal de controlo no instante k-2
    NNinput(4,k)=u(k-4)/5;%O sinal de controlo no instante k-3
    NNinput(5,k)=u(k-5)/5;%O sinal de controlo no instante k-4
    NNinput(6,k)=ref(k+1)/4;%A distancia pretendida
    NNinput(7,k)=z(k)/4;%A distancia atual
    NNinput(8,k)=z(k-1)/4;%A distancia em k-1
    NNinput(9,k)=z(k-2)/4;%A distancia em k-2
    NNinput(10,k)=z(k-3)/4;%A distancia em k-3
    NNinput(11,k)=z(k-4)/4;%A distancia em k-4
    NNoutput(k)=u(k)/5;%O sinal de controlo pretendido
end

%k comeca em 11 para poder adicionar 10 amostras anteriores dos sinais
for k = 11:(N-1)
    NNinput(1,k)=u(k-1)/5;%O sinal de controlo anterior
    NNinput(2,k)=u(k-2)/5;%O sinal de controlo no instante k-1
    NNinput(3,k)=u(k-3)/5;%O sinal de controlo no instante k-2
    NNinput(4,k)=u(k-4)/5;%O sinal de controlo no instante k-3
    NNinput(5,k)=u(k-5)/5;%O sinal de controlo no instante k-4
    NNinput(6,k)=u(k-6)/5;%O sinal de controlo no instante k-5
    NNinput(7,k)=u(k-7)/5;%O sinal de controlo no instante k-6
    NNinput(8,k)=u(k-8)/5;%O sinal de controlo no instante k-7
    NNinput(9,k)=u(k-9)/5;%O sinal de controlo no instante k-9
    NNinput(10,k)=u(k-10)/5;%O sinal de controlo no instante k-10
    NNinput(11,k)=ref(k+1)/4;%A distancia pretendida
    NNinput(12,k)=z(k)/4;%A distancia atual em k
    NNinput(13,k)=z(k-1)/4;%A distancia em k-1
    NNinput(14,k)=z(k-2)/4;%A distancia em k-2
    NNinput(15,k)=z(k-3)/4;%A distancia em k-3
    NNinput(16,k)=z(k-4)/4;%A distancia em k-4
    NNinput(17,k)=z(k-5)/4;%A distancia em k-5
    NNinput(18,k)=z(k-6)/4;%A distancia em k-6
    NNinput(19,k)=z(k-7)/4;%A distancia em k-7
    NNinput(20,k)=z(k-8)/4;%A distancia em k-8
    NNinput(21,k)=z(k-9)/4;%A distancia em k-9
    NNoutput(k)=u(k)/5;%O sinal de controlo pretendido
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Treino da rede de Controlo de Modelo de Referencia 5 camadas %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load Dados_de_treino

%numero de neuronios
S1=10;
S2=5;
```

```
%delays
d1 = [1:10];
d2 = [1:5];

narx_net_closed = closeloop(narx_net);
%view(narx_net_closed)

%cria a rede de de 5 camadas
mrac_net = feedforwardnet([1 S2 1 S1]);
%Estabelece as ligaes entre camadas
mrac_net.layerConnect = [0 0 1 0 1;1 0 0 0 0 ;0 1 0 0 0;0 0 1 0 1;0 0 0 1 0];
mrac_net.outputs{5}.feedbackMode = 'closed';%malha fechada
% funcao de ativacao
mrac_net.layers{2}.transferFcn = 'tansig';
mrac_net.layers{3}.transferFcn = 'poslin';
%delays na quarta camada
mrac_net.layerWeights{4,5}.delays = d1;
mrac_net.layerWeights{4,3}.delays = d1;
% Para no alteraros valores das ultimas duas camadas
mrac_net.layerWeights{4,2}.learn = 0;
mrac_net.layerWeights{4,4}.learn = 0;
mrac_net.layerWeights{5,3}.learn = 0;
mrac_net.biases{4}.learn = 0;
mrac_net.biases{5}.learn = 0;
mrac_net.name = 'Model Reference Control Network';
%delays na primeira camada
mrac_net.layerWeights{1,3}.delays = d2;
mrac_net.layerWeights{1,5}.delays = d2;
mrac_net.inputWeights{1}.delays = d2;

%dados de treino para o controlador de modelo de referncia
refin =num2cell(refnn');
refout=num2cell(znn');
mrac_net = configure(mrac_net,refin,refout);

%copia os valores da rede de identificao para as duas ultimas camadas
mrac_net.LW{4,3} = narx_net_closed.IW{1};
mrac_net.LW{4,5} = narx_net_closed.LW{1,2};
mrac_net.b{4} = narx_net_closed.b{1};
mrac_net.LW{5,4} = narx_net_closed.LW{2,1};
mrac_net.b{5} = narx_net_closed.b{2};
mrac_net.LW{3,1} = zeros(size(mrac_net.LW{3,1}));
mrac_net.b{3} = 0;
%view(mrac_net)

mrac_net.plotFcns ={'plotperform','plottrainstate', 'plotresponse'};
mrac_net.trainFcn = 'trainlm';

[x_tot, xi_tot, ai_tot, t_tot] = preparets(mrac_net,refin,{},refout);
mrac_net.trainParam.epochs = 100;
mrac_net.trainParam.min_grad = 1e-100;
```

```
mrac_net.trainParam.mu_max = 1e+100;
%treina a rede
[mrac_net,tr] = train(mrac_net,x_tot,t_tot,xi_tot,ai_tot)

%simula a rede
y=mrac_net(x_tot,xi_tot,ai_tot);
%calcula o vetor de erro para os diferentes pontos
e=gsubtract(t_tot,y);
%calcula a performance(MSE) da rede treinada
performance=perform(mrac_net,t_tot,y)
plot(znn)
grid on
hold on
plot(cell2mat(y))

%% Teste da rede de modelo de referencia
clear
load dados_controlo_reference

% teste step
refin=num2cell(ref_step_robust');
refout=num2cell(z_step_robust');
[x_tot,xi_tot,ai_tot,t_tot] = preparets(mrac_net,refin,{},refout);
%simula a rede
y=mrac_net(x_tot,xi_tot,ai_tot);
%calcula o vetor de erro para os diferentes pontos
e=gsubtract(t_tot,y);
%calcula a performance(MSE) da rede treinada
performance=perform(mrac_net,t_tot,y)
%cria os vetores do tempo para cada um dos dados
time1=[0:length(ref_step_robust)-1]*0.003;
time2=[0:length(y)-1]*0.003;
figure
subplot(3,1,1)
plot(time2',cell2mat(y),'r')
axis([0 30 2.2 3.4])
grid on
hold on
plot(time1',ref_step_robust)

% teste saw
refin=num2cell(ref_saw_robust');
refout=num2cell(z_saw_robust');
[x_tot,xi_tot,ai_tot,t_tot] = preparets(mrac_net,refin,{},refout);
y=mrac_net(x_tot,xi_tot,ai_tot);
e=gsubtract(t_tot,y);
performance=perform(mrac_net,t_tot,y)
time1=[0:length(ref_saw_robust)-1]*0.003;
time2=[0:length(y)-1]*0.003;
subplot(3,1,2)
plot(time2',cell2mat(y),'r')
```

```
axis([0  30  2.2  3.4])
grid on
hold on
plot(time1',ref_saw_robust)

%teste sine
refin=num2cell(ref_sine_robust');
refout=num2cell(z_sine_robust');
[x_tot,xi_tot,ai_tot,t_tot] = preparets(mrac_net,refin,{},refout);
y=mrac_net(x_tot,xi_tot,ai_tot);
e=gsubtract(t_tot,y);
performance=perform(mrac_net,t_tot,y)
time1=[0:length(ref_sine_robust)-1]*0.003;
time2=[0:length(y)-1]*0.003;
subplot(3,1,3)
plot(time2',cell2mat(y),'r')
axis([0  30  2.2  3.4])
grid on
hold on
plot(time1', ref_sine_robust)

%teste treino
refin=num2cell(refnn');
refout=num2cell(znn');
[x_tot,xi_tot,ai_tot,t_tot] = preparets(mrac_net,refin,{},refout);
y=mrac_net(x_tot,xi_tot,ai_tot);
e=gsubtract(t_tot,y);
performance=perform(mrac_net,t_tot,y)
time1=[0:length(refnn)-1]*0.003;
time2=[0:length(y)-1]*0.003;
figure
plot(time1',refnn)
axis([0  40  2  3.4])
grid on
hold on
plot(time2',cell2mat(y),'r')

figure
plot(time1',CN_ref_sine.znn2','r')
axis([0  30  2  3.4])
grid on
hold on
plot(time1', ref_sine_robust)

figure
plot(time1',CN_ref_sine.unn2','r')
axis([0  30  2.5  4.5])
grid on
```

Anexo E

Simulação das redes neuronais de identificação para outras entradas de referência

Este Anexo tem o intuito de comprovar o funcionamento satisfatório das redes de identificação treinadas. Neste encontram-se representadas as respostas das redes de identificação para diferentes sinais de referência e os respetivos sinais de controlo aplicados. Os sinais de referência incluem degrau, sinusoidal e serra.

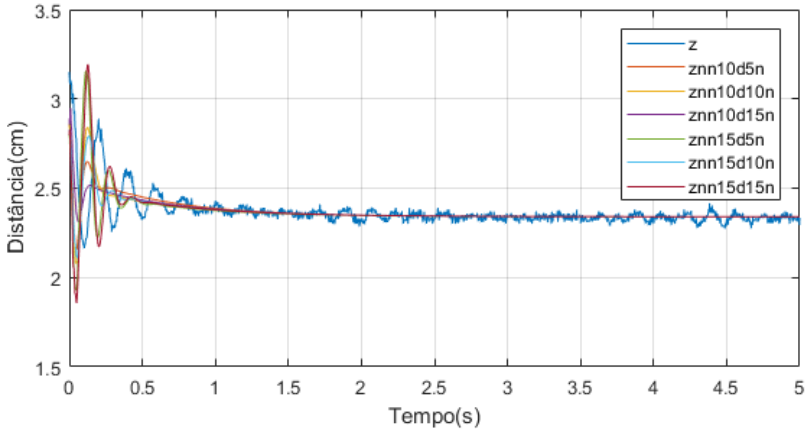


Figura E.1: Respostas do sistema e redes de identificação para entrada em degrau

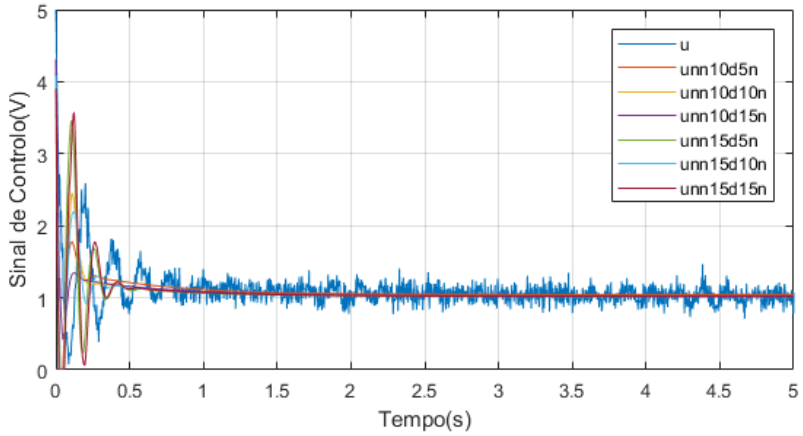


Figura E.2: Sinal de controlo aplicado no sistema e redes de identificação para entrada em degrau

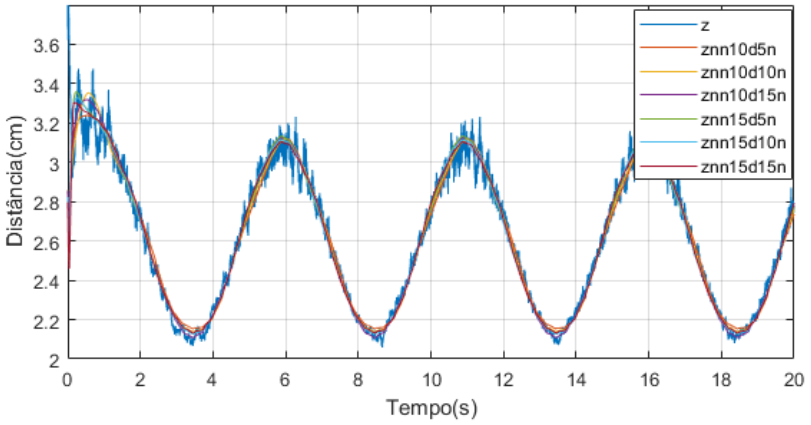


Figura E.3: Respostas do sistema e redes de identificação para entrada sinusoidal

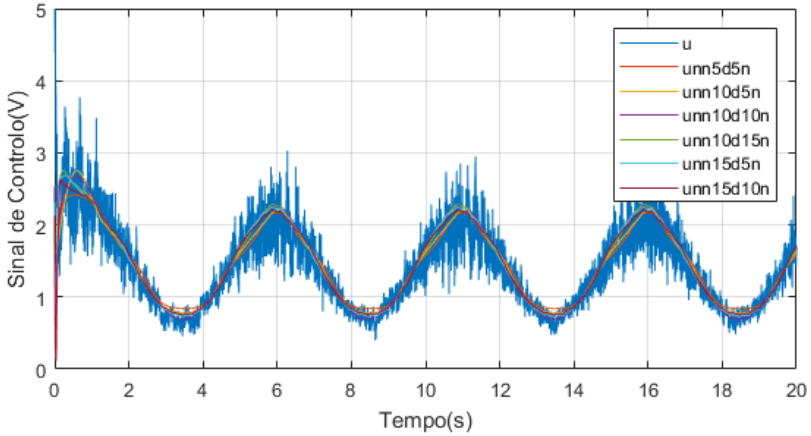


Figura E.4: Sinal de controlo aplicado no sistema e redes de identificação para entrada sinusoidal

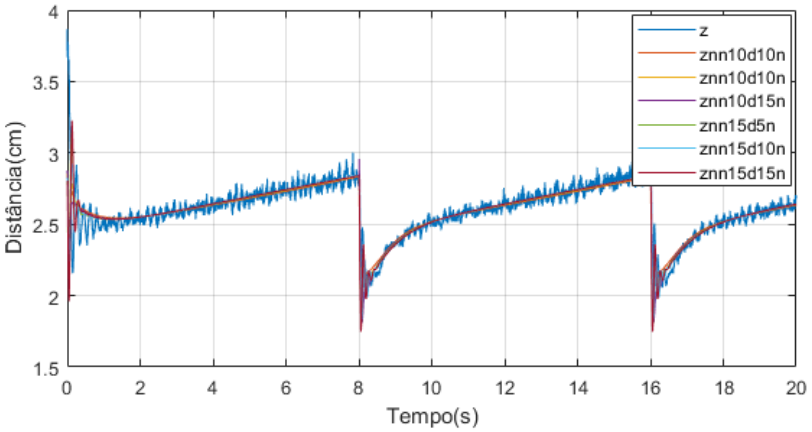


Figura E.5: Respostas do sistema e redes de identificação para entrada em serra

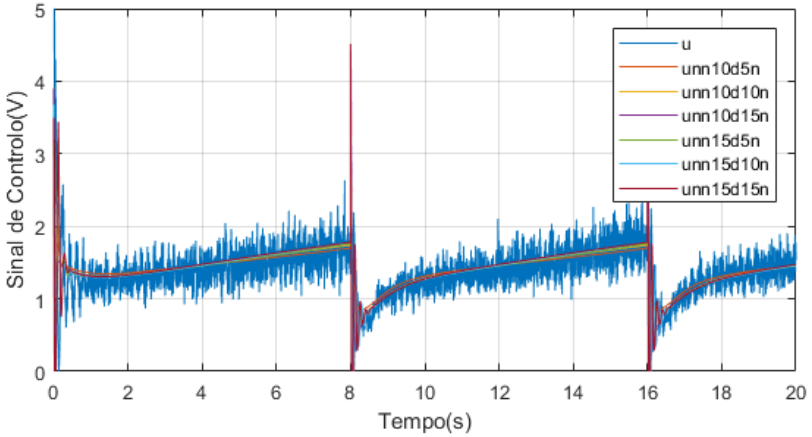


Figura E.6: Sinal de controlo aplicado no sistema e redes de identificação para entrada em serra

Anexo F

Código C desenvolvido para o ATmega2560

```
/*
 * Maglev.c
 *
 * Created: 24/04/2020 15:28:41
 * Author : Bruno Silva 1150610
 */
#define F_CPU 16000000UL      //Frequencia de clock

#include <avr/io.h>
#include <avr/interrupt.h>
#include <math.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

#define BAUD  250000          //Baudrate da usart
#define FOSC  16000000        //Para configurar USART

//Macros
#define atv_tanh(val_f) ((2/(1+exp(-2*val_f)))-1) //tangente hiperbolica
#define atv_sig( val_f ) (1/(1+exp(-1*val_f)))    //sigmoide

// Variveis ------------------------
unsigned int ADC_z=0, ADC_i=0;//Guardam a media das 5 medicoes
unsigned int z_i=0, z_d=0;//Para imprimir o float com o sprintf
unsigned int u_i=0, u_d=0;//Para imprimir o float com o sprintf
unsigned int cnt_ref=0;//Para calculo da posicao de referencia
char data[40];//Para enviar pela USART0 /// Contador para o LED de status2
float Coef_sine=0.0;//Para calculo  posio  de referencia
float AK=0.0;//Coeficientes do controlador Avano
```

```
float A=0.0, B=0.0, C=0.0, D=0.0, E=0.0;//Coeficientes do controlador Avano-atraso
float i=0.0, z=0.0;//Corrente ///Distancia
float erro=0.0, erro_old=0.0, erro_old2=0.0;//Erro no instante k e k-1
float u=0.0, u_sat=0.0;//Sinal de controlo no instante k e saturado entre 0-5V

// Redes Neuronais //
float z_delays[5]={3.0, 3.0, 3.0, 3.0, 3.0};
float u_delays[5]={0.0, 0.0, 0.0, 0.0, 0.0};
//Para modelo inverso
float CN_L1=0.0,CN_out=0.0;//saida de cada camada
float CN_L2[5]={0.0, 0.0, 0.0, 0.0, 0.0};
float CN_SUM1=0.0,CN_SUM_out=0.0;//Somatorio de cada camada
float CN_SUM2[5]={0.0, 0.0, 0.0, 0.0, 0.0};

//Volatiles porque sao (ou podem ser) usadas em interrupcoes
//Flag para controlar ciclo de controlo /// Contador para o LED de status
volatile unsigned char flag=0, cnt=0;
//Caso ON==1 realiza controlo /// Controler determina o controlador /// Ref determina a referencia
volatile unsigned int ON=0, Controler=0, Ref=0;
volatile unsigned int pwm=0;//Valor do pwm (0-15999)
volatile float z_ref=2.25;//Distancia de referencia
volatile float u_adpt=0.0;//Para o bloco adaptativo
volatile int USART_receive_buffer=0;//Buffer de rececao para a USART
//-------------------------------------

//Constantes-------------------------------
const float Gpwm=3199.8;//Conversao de Volts para o valor a comparar com o TIMER1
//Parametros do sensor de distncia
const float alpha=507.045, beta=1123.801, gam=69.474;
//alpha(ADU) /// beta(ADU*cm^3) /// gamma(ADU/A)
//Parametros do sensor de corrente
const float G_si=0.03324, C_si=-17.16324;//G_si=ganho i/ADC /// C_si=constante A

//Obtidos para R=2.9/ieq=0.27/zeq=2
const float ueq=0.783; //Tensao de equilibrio R*ieq
//Parametros Controlador em Avano
const float A_avc=0.8756, B_avc=0.6806, K_avc=-6.8316; //A=Zero /// B=polo /// K=Ganho
//Parametros Controlador Avano-Atraso
const float A_at=0.9979, B_at=1.0, K_at=1.0;

//Pesos e polaridades Rede de Controlo modelo inverso 1TH_5S_5D
const float W1_CN[11]={0.0623,0.2784,0.3393,0.4790,0.2591,-0.4790,
                       20.3203,-5.1671,-4.4871,-3.3678,-3.4983},
            W2_CN[5]={49.2489,7.5915,0.2844,-105.0548,16.2385},
            W3_CN[5]={3.7307,0.5573,4.1638,-0.0083,0.0921};

const float B1_CN=-2.5505,
            B2_CN[5]={-51.2138,-7.8737,-0.8983,-46.1446,10.8738},
            B3_CN=-1.0189;

//Bloco adapatativo
```

```
const float adpt=0.7/4.0;
const float Coef_saw=0.0006;
//-------------------------------------

// Funes ------------------------------
void Config();//Configuracoes de perifericos
void Coefs();//Calcula Coeficientes
void Send_data(char *buffer);//Enviar dados pela USART0
unsigned int ADC_read(unsigned char chn);//Realiza uma leitura do ADC
void Control();//Contem os ciclos de controlo
void Reference();//Calcula a referencia mediante o sinal escolhido
float CN_1TH_5S_5D();//Rede neuronal de modelo inverso
float NARX_10d10n();//Rede neuronal de identificacao
//-------------------------------------

int main(void){
    Config();//Configuracoes iniciais
    Coefs();
    ADC_read(0);//Para inicializar o ADC(primeira conversao mais lenta)

    unsigned int Measure_z=0, Measure_i=0, cycles=0;//Guardam as leituras do ADC
    float Cr=0, den=0, frac=0;
    unsigned char k=0, j=0, cnt2=0;

    TCNT0=0;//Reset Counter
    TCNT1=0;//Reset Counter

    //Ciclo infinito
    while(1){
        //Faz um ciclo depois espera pela interrupcao
        while(flag==0){
            //Realiza 5 medicoes de cada sensor
            for (j=0;j<3;j++){
                Measure_i+=ADC_read(1);//Sensor de Corrente
                Measure_z+=ADC_read(0);//Sensor de Distancia
            }

            //Media dos valores
            ADC_i=Measure_i/3;
            ADC_z=Measure_z/3;

            //Reset dos valores
            Measure_z=0;
            Measure_i=0;

            //Calculo da corrente
            i=(G_si*ADC_i)+C_si;//Corrente em A
            if(i<0){i=0;}

            //Calculo da distancia
            Cr=i*gam;//Influencia da corrente
```

```
            den=ADC_z-alpha-Cr;//Denominador
            frac=beta/den;//Quociente
            z=cbrt(frac);//Distancia em cm

            for( k=0; k<4; k++)//de 0 a 4
                z_delays[4-k]=z_delays[3-k];//da shift ao valores de z
            z_delays[0]=z;//guarda o valor mais recente de z

            if(ON==1){
                Reference();
                Control();//Realiza o controlo do sistema
                cnt2++;
                cycles++;
                if(cnt2==50){//caso tenha passado 0,15 segundos
                    cnt2=0;
                    z_i=(int)z;
                    z_d=(z-z_i)*1000.00;
                    sprintf(data,"%d.%d   \n\r", z_i ,abs(z_d));
                    Send_data(data);
                }
            }
            else{
                erro=0;
                erro_old=0;
                u=0;
                u_sat=0;
                pwm=0;
                u_adpt=0;
                for(k=0; k>4; k++)
                    u_delays[k]=0;
            }
            flag=1;//Coloca a 1 para sair do ciclo while
        }
    }
}

//Configuracoes
void Config(){
    // Configurao de I/O
    DDRB=(1<<PB3)|(1<<PB5)|(1<<PB7);
    //Pino B5-PWM(OC1A) /// Pino B7-LED status /// Pino B3-LED debug
    DDRF&=~(1<<PF0)&&~(PF1);
    //Pino F0-ADC0 /// Pino F1-ADC1

    //Timer0 de 3ms para algoritmo de controlo.
    OCR0A=186;
    TCCR0A=(1<<WGM01);
    //OC0A e OC0B Disconnected /// CTC
    TCCR0B=(1<<CS02);
    //Prescaller=256=>f=166.6667Hz
    TIMSK0=(1<<OCIE0A);
```

```
    //Timer/Counter0 Output Compare Match A Interrupt Enable

    //Timer1 para gerar PWM
    TCCR1A=(1<<COM1A1)|(1<<WGM11);
    //Clear OC1A on compare match, set OC1A at BOTTOM (non-inverting mode)
    TCCR1B=(1<<CS10)|(1<<WGM13)|(1<<WGM12);
    //No Prescaller /// Fast-PWM
    ICR1=15999;//Set TOP->F=1000Hz
    OCR1A=0;//Duty Cycle(0-15999)(0-5V)

    //  Configurao  do conversor AD
    ADMUX=(1<<REFS0);
    //AVCC with external capacitor at AREF pin /// 10bits
    ADCSRA=(1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
    //ADC Enabled /// Divison Factor=128=>Fadc=125kHz

    //  Configurao  da USART - pag 218/222
    UCSR0A=(1<<U2X0);
    //Double the USART Transmission Speed
    UCSR0B=(1<<RXCIE0)|(1<<RXEN0)|(1<<TXEN0);
    //RX Complete Interrupt Enable /// Receiver Enable /// Transmitter Enable
    UCSR0C=(1<<UCSZ01)|(1<<UCSZ00);
    //Asynchronous /// 1 Stop bit /// Disabled parity /// 8-bit
    unsigned int ubrr;
    ubrr=(FOSC/(8*BAUD))-1;//Calcula o ubrr pretendido Fosc=16MHz e BAUD 250000
    UBRR0H=(unsigned char)(ubrr>>8);
    UBRR0L=(unsigned char)ubrr;//Para estas defies obtm -se 0.0% de erro

    sei (); //Ativa as  interrupes
}

//Coeficientes
void Coefs(){
    //Para o controlador em Avano
    AK=K_avc*A_avc;

    //Para o controlador em Avano-Atraso
    E=K_at*K_avc;
    A=B_at+B_avc;
    B=B_at*B_avc;
    C=(A_at+A_avc)*E;
    D=A_at*A_avc*E;

    Coef_sine=(2*3.14159)/1667;
    //1667  o coeficiente do periodo da sinusoide pelo periodo de amostragem 5/0.003
}

//Envia os dados pela USART0
void Send_data(char *buffer){
    unsigned char i=0;
    while(buffer[i]!='\0')
```

```
    {
        //Espera que o buffer de envio fique vazio
        while (( UCSR0A & (1<<UDRE0)) == 0);
        UDR0 = buffer[i];// Pe os dados no buffer de envio
        i++;
    }
}

//Conversao e Leitura do ADC
unsigned int ADC_read(unsigned char chn){
    ADMUX = (ADMUX & 0xFE)|chn;//Seleciona o conversor

    ADCSRA |= (1<<ADSC);//Inicia a converso
    while(ADCSRA & (1<<ADSC));//Espera que a converso termine

    return (ADC);//Retorna o valor convertido
}

//Ciclo de Controlo
void Control(){
    unsigned char i=0;
    float temp1=0,temp2=0;

    erro=z_ref-z;//Calcula o erro

    switch(Controler)
    {
        case 0://Controlador em Avanco
            temp1=u_delays[0]-ueq;
            u=(B_avc*temp1)-(AK*erro_old)+(K_avc*erro);//Calcula o valor do sinal de controlo
            u=u+ueq;//Adiciona a tensao de equilibrio
            u_adpt=0;
            break;
        case 1://Controlador Avano-Atraso
            temp1=u_delays[0]-ueq;
            temp2=u_delays[1]-ueq;
            u=(A*temp1)-(B*temp2)+(D*erro_old2)-(C*erro_old)+(E*erro);
            u=u+ueq;//Adiciona a tensao de equilibrio
            u_adpt=0;
            break;
        case 2://Controlador Neuronal de Modelo Inverso
            u=CN_1TH_5S_5D();
            u_adpt=0;
            break;
        case 3://Controlador Neuronal de Modelo Inverso com bloco adaptativo
            u=CN_1TH_5S_5D();
            u+=u_adpt;
            break;
    }

    for( i=0; i<4; i++)//de 0 a 3
```

```
            u_delays[4-i]=u_delays[3-i];//da shift ao valores de u
        u_delays[0]=u-u_adpt;//guarda o valor mais recente de u

        u_adpt-=(adpt*erro*0.003)*5;

        erro_old2=erro_old;//Guarda o erro k-2
        erro_old=erro;//Guarda o erro k-1

        // Saturao do controlo
        if(u>5.0){u_sat=5.0;}
        if(u<0.0){u_sat=0.0;}
        else{u_sat=u;}

        pwm=(int)round(u_sat*Gpwm);//Calcula o valor do pwm(0-15999)
}

// Posio de Referencia
void Reference(){
        switch(Ref)
        {
                case 0://Step
                        z_ref=2.3;
                        break;
                case 1://Square
                        if (cnt_ref>=833)
                                z_ref=2.3;
                        else
                                z_ref=3.0;
                        break;
                case 2://Saw
                        z_ref=(Coef_saw*cnt_ref)+2.4;
                        break;
        }
        cnt_ref++;
        if(cnt_ref>=1667)
                cnt_ref=0;
}

//Rede neuronal de modelo inverso
float CN_1TH_5S_5D(){
        unsigned char i=0;//para os ciclos for

        CN_SUM1=0.0;//Reset aos somatorios da camada 1
        for (i=0; i>5; i++)//de 0 a 4
                CN_SUM2[i]=0.0;//Reset aos somatorios da camada 2
        CN_SUM_out=0.0;//Reset aos somatorios da camada 3

        //primeira camada
        for( i=0; i<5; i++)//de 0 a 4
        {
                CN_SUM1+=W1_CN[i]*(u_delays[i]/5);//calcula para os valores de u
```

```
        CN_SUM1+=W1_CN[6+i]*(z_delays[i]/4);//calcula para os valores de z
    }
    CN_SUM1+=W1_CN[5]*(z_ref/4);//calcula para a referencia
    CN_SUM1+=B1_CN;//adiciona a polaridade
    CN_L1=atv_tanh(CN_SUM1);//Saida da primeira camada

    //segunda camada
    for( i=0; i<5; i++)//de 0 a 4
    {
        CN_SUM2[i]=W2_CN[i]*CN_L1+B2_CN[i];//calcula o somatorio
        CN_L2[i]=atv_sig(CN_SUM2[i]);//calcula saida de cada neuronio
    }

    //terceira camada
    for( i=0; i<5; i++)//de 0 a 4
        CN_SUM_out+=W3_CN[i]*CN_L2[i];//calcula o somatorio
    CN_out=CN_SUM_out+B3_CN;//Saida da ultima camada

    return (CN_out*5);//devolve o sinal de controlo
}

//Interrupcao a cada 3ms
ISR(TIMER0_COMPA_vect){
    OCR1A=pwm;
    cnt++;
    if (cnt==167)
    {
        PORTB^=(1<<PB7);
        cnt=0;
    }
    flag=0;//Coloca a zero para voltar a fazer as medicoes
}

//Interrupcao de rececao da USART0
ISR(USART0_RX_vect){
    USART_receive_buffer=UDR0;
    switch (USART_receive_buffer)
    {
        case 'a':
            ON=1;//Comea o controlo
            Send_data("ON\r\n");
            break;
        case 'b':
            ON=0;//Desliga o controlo
            Send_data("OFF\r\n");
            break;
        case 'z':
            Controler=0;
            //Seleciona o controlador Av
            u_adpt=0;
            Send_data("Controlador em Avano\r\n");
```

```
            break;
        case 'x':
            Controler=1;
            //Seleciona o controlador AvAt
            u_adpt=0;
            Send_data("Controlador Avano-PI\r\n");
            break;
        case 'c':
            Controler=2;
            //Seleciona o controlador neuronal de modelo inverso
            u_adpt=0;
            Send_data("Controlador Neuronal de Modelo Inverso\r\n");
            break;
        case 'v':
            Controler=3;
            //Seleciona o controlador neuronal de modelo inverso com bloco adaptativo
            Send_data("Controlador Neuronal de Modelo Inverso com bloco adaptativo\r\n");
            break;
            case 'i':
            Ref=0;
            //Seleciona a referencia em step
            Send_data("Referencia em degrau\r\n");
            break;
        case 'o':
            Ref=1;
            //Seleciona a referencia quadrada
            Send_data("Referencia em onda quadrada\r\n");
            break;
        case 'p':
            Ref=2;
            //Seleciona a referencia em serra
            Send_data("Referencia em dente de serra\r\n");
            break;
        default:
            Send_data("Invalido\r\n");
            break;
    }
    USART_receive_buffer=0;
}
```