UNIVERSIDADE ESTADUAL PAULISTA"JÚLIO DE MESQUITA FILHO
CÂMPUS DE ILHA SOLTEIRA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

RICARDO FERNANDO NUNES

MAPEAMENTO DA CINEMÁTICA INVERSA DE UM MANIPULADOR ROBÓTICO UTILIZANDO REDES NEURAIS ARTIFICIAIS CONFIGURADAS EM PARALELO

Ilha Solteira
2016

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

RICARDO FERNANDO NUNES

MAPEAMENTO DA CINEMÁTICA INVERSA DE UM MANIPULADOR ROBÓTICO UTILIZANDO REDES NEURAIS ARTIFICIAIS CONFIGURADAS EM PARALELO

Dissertação apresentada à Faculdade de Engenharia de Ilha Solteira – UNESP, como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica.
Área de Concentração: Automação.

Profa. Dra. Suely Cunha Amaro Mantovani
Orientadora

Ilha Solteira
2016

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

Nunes, Ricardo Fernando.
N972m Mapeamento da cinemática inversa de um manipulador robótico utilizando redes neurais artificiais configuradas em paralelo / Ricardo Fernando Nunes. -- Ilha Solteira: [s.n.], 2016
100 f. : il.

Dissertação (mestrado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de conhecimento: Automação, 2016

Orientador: Suely Cunha Amaro Mantovani
Inclui bibliografia

1. Manipuladores robóticos. 2. Cinemática direta e inversa. 3. Redes neurais artificiais. 4. Plataforma de desenvolvimento.

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Ilha Solteira

CERTIFICADO DE APROVAÇÃO

TÍTULO DA DISSERTAÇÃO: Mapeamento da cinemática inversa de um manipulador robótico utilizando Redes Neurais Artificiais configuradas em paralelo

AUTOR: RICARDO FERNANDO NUNES
ORIENTADORA: SUELY CUNHA AMARO MANTOVANI

Aprovado como parte das exigências para obtenção do Título de Mestre em ENGENHARIA ELÉTRICA, área: AUTOMAÇÃO, pela Comissão Examinadora:

Profa. Dra. SUELY CUNHA AMARO MANTOVANI
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. NOBUO OKI
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. TIAGO PEREIRA DO NASCIMENTO
Departamento de Sistemas de Computação / Universidade Federal da Paraíba

Ilha Solteira, 31 de março de 2016

AGRADECIMENTOS

Aos meus pais, Marcos Luiz Nunes e Conceição de Sousa Silva Nunes, meu irmão Eduardo Fernando Nunes e minha namorada Luana Almeida Gonzaga que sempre me apoiaram e incentivaram nesta caminhada.

Meus sinceros agradecimentos à minha orientadora profa. Dra. Suely Cunha Amaro Mantovani, primeiramente pela oportunidade, posteriormente pela paciência, incentivos e contribuições para o meu aprendizado e minha formação.

A FEIS-UNESP e seus funcionários, que contribuíram direta e indiretamente em minha formação profissional e pessoal, em especial ao técnico de laboratório Valdemir Chaves pela contribuição no protótipo desenvolvido.

Ao Instituto Federal de Educação, Ciência e Tecnologia do Estado de São Paulo (IFSP) pelo incentivo acadêmico.

Aos meus amigos de laboratório Jovanny Bedoya Guapacha, Rodolfo de Castro Silva, Thiago Rodrigues Pentiado, Hugo Bach Artico e Jean Rafael Camillo pelo compartilhamento de sabedoria, frustações e alegrias.

Agradeço também pelo convívio entre meus amigos de Ilha Solteira, Uiliam Nelson Lendzion Tomaz Alves, Diogo Ramalho de Oliveira, Alexandre Ataide Carniato, João Ricardo Lhullier Lugão e todos os meus amigos e companheiros de trabalho, Leonardo Ataide Carniato, Fernando Barros Rodrigues, Haislan Ranelli Santana Bernardes, Fabrício Marqui Sanches, Andryos da Silva Lemes, Tiago Veronese Ortunho, José Guilherme Magalini Santos Decanini, Willians França Leite, Charles de Souza Silva, Carlos Fernando Joventino, pelos auxílios, conselhos e momentos de descontração que passamos juntos.

RESUMO

Neste trabalho apresenta-se uma abordagem para o mapeamento da cinemática inversa utilizando Redes Neurais Artificiais do tipo Perceptron Multicamadas na configuração em paralelo, tendo como referência o protótipo de um manipulador robótico de 5 graus de liberdade, composto por sete servomotores controlado pela plataforma de desenvolvimento Intel® Galileo Gen 2. As equações da cinemática inversa, normalmente apresentam múltiplas soluções, desta forma, uma solução interessante e frequentemente encontrada na literatura são as Redes Neurais Artificiais (RNA) em razão da sua flexibilidade e capacidade de aprendizado por meio do treinamento. As Redes Neurais são capazes de entender a relação cinemática entre o sistema de coordenadas das juntas e a posição final da ferramenta do manipulador. Para avaliar a eficiência do método proposto foram realizadas simulações no software MATLAB, as quais demostram pelos resultados obtidos e comparações a uma RNA do tipo MLP simples, aproximadamente redução das médias dos erros das juntas em até 87,8% quando aplicado à trajetória e 80% quando aplicado a pontos distribuídos no volume de trabalho.

Palavras-chave: Manipuladores Robóticos. Cinemática Direta e Inversa. Redes Neurais Artificiais. Plataforma de Desenvolvimento.

ABSTRACT

This paper presents an approach to the mapping of inverse kinematics using Artificial Neural Networks Multilayer Perceptron in parallel configuration, in the prototype of a robotic manipulator 5 degrees of freedom, as reference, composed of seven servomotors controlled by development board Intel® Galileo Gen 2. The equations of inverse kinematics, usually have multiple solutions, therefore, an interesting solution and often found in the literature are the Artificial Neural Networks (ANN) because of their flexibility and learning capacity through training. Neural Networks are able to understand the kinematic relationship between the coordinate system of the joints and the final position of the manipulator tool. To evaluate the efficiency of the proposed, simulations in MATLAB software are performaded, that demonstrate by the results obtained and compared to a simple MLP type RNA, one reduction in mean errors of the joints by up to 87.8% when applied to the path and 80% when applied to points distributed in the work space.

Keywords: Robotic manipulators. Direct and inverse kinematics. Artificial neural network. Development board.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURA E SIGLAS

ANFIS	Adaptive Neuro-Fuzzy Inference System
BP	*Backpropagation*
D-H	Denavit–Hartenberg
DOF	Degrees of Freedom
EM	Electromagnetism-like Method
GDL	Grau(s) de Liberdade
MLP	Multilayer Perceptron
PD2	Proporcional Derivativo Duplo
PMC	Perceptron Multicamadas
RBF	Radial-Basis Function
RNA	Rede Neural Artificial – Redes Neurais Artificiais

SUMÁRIO

CAPÍTULO 1. INTRODUÇÃO

O tema “robótica” é um campo de pesquisa tecnológica multidisciplinar envolvendo as diversas engenharias - Elétrica, a Mecânica e a Engenharia Industrial, a Ciência da Computação, e a Matemática - isso explica a sua complexidade e a necessidade de conhecimentos nestas áreas, para a seu domínio e aplicação.

Os robôs têm sido observados em brinquedos, nas tarefas domésticas, na medicina, em explorações espaciais e outras tarefas de risco ao ser humano. Podem ser classificados primariamente como robôs manipuladores e robôs móveis.

A característica principal dos robôs móveis é a presença de uma base móvel que permite ao robô mover-se livremente em um ambiente. Os robôs móveis são usados principalmente em aplicações de serviços em que é necessária a capacidade de movimento autônomo, ao contrário dos manipuladores que são encontrados principalmente, na indústria.

A utilização de manipuladores robóticos nos tempos atuais, se tornou parte indispensável nas modernas fábricas e outros setores nos quais eles possam auxiliar em toda e qualquer tarefa visando a automação.

A importância dos manipuladores robóticos nas indústrias é principalmente, em função das tarefas repetitivas que eles podem executar, nas montadoras de veículos e de placas eletrônicas, carga e embaladoras de produtos em geral. Essa importância está relacionada com características como, o aumento na produção, padronização de produtos, possuírem força mecânica e precisão, resistência, apresentarem boa repetitividade e menos suscetíveis às falhas se instalados e programados corretamente.

Os robôs manipuladores consistem de uma sequência de elos que podem ser rígidos ou flexíveis, interconectados por meio de articulações chamadas de juntas. Geralmente são compostos por um braço que assegura mobilidade, um pulso e um efetuador (*end-effector),* ou garra, ao qual pode ser acoplada uma determinada ferramenta para realizar a tarefa desejada pelo robô. Cada junta apresentada na estrutura mecânica do manipulador contribui, normalmente, com um (1) grau de liberdade (GDL), ou em inglês, *Degrees of Freedom* (DOF) (FERNANDES, 2014).

O movimento em cada grau de liberdade se dá por atuadores, que podem ser pneumáticos, hidráulicos ou elétricos, ativados por um sinal de um controlador. Um controlador com base em um sinal de referência, se em um sistema de controle em malha

fechada, aplica um sinal de correção necessário para posicionar o manipulador de uma forma correta, após coleta de dados derivadas de sensores. Por outro lado, se o controlador pertencer a um sistema de controle em malha aberta, o acionamento dos atuadores se dá apenas por um sinal de referência e neste caso, espera-se que o manipulador se comporte de forma desejada ao longo do tempo.

Para um manipulador robótico executar uma determinada tarefa é necessário fazer a análise de seus modelos cinemáticos que descrevem matematicamente o movimento espacial realizado por cada elo do robô. Na cinemática de robôs estudam-se os movimentos do robô sem se importar com as forças que os causam.

Existem dois modelos cinemáticos, o modelo da cinemática direta e o da cinemática inversa. Quando se analisa seus movimentos por meio da cinemática, a complexidade geométrica aumenta bastante se o manipulador robótico apresenta muitos graus de liberdade, principalmente se a análise é baseada na cinemática inversa de robôs manipuladores.

A cinemática inversa apresenta grandes desafios devido as equações serem não lineares, poderem ou não apresentar solução ou múltiplas soluções, pois é comum a presença de redundâncias nesta análise para manipuladores com muitos graus de liberdade (CRAIG, 2012).

Alguns métodos tradicionais utilizados para calcular a cinemática inversa de manipuladores robóticos tais como, geométricos, numérico-interativos e os algébricos, são inadequados se a estrutura do manipulador for muito complexa (ALAVANDAR; NIGAM, 2008). Desta forma, abordagens alternativas buscam soluções e o uso de Redes Neurais Artificiais (RNA), é frequentemente encontrada na literatura para resolver a cinemática inversa, uma vez que são capazes de entender a relação do manipulador, em razão da sua flexibilidade e capacidade de aprendizado por meio do treinamento.

1.1 MOTIVAÇÃO DA PESQUISA

O modelo cinemático inverso, através do qual obtém-se o estado das juntas de um manipulador em função da posição desejada para a ferramenta, é fundamental para que os manipuladores robóticos, comerciais ou não, se posicionem de forma precisa para realizar suas tarefas.

Desta forma, tentando contribuir com as pesquisas nesta área, propõe-se o uso das RNAs para resolver o problema da cinemática inversa tendo como referência o protótipo do manipulador robótico de 5 GDL desenvolvido em laboratório.

A obtenção do modelo cinemático inverso, como mencionado anteriormente, envolve equações não lineares que pode admitir mais de uma solução. Em consequência, o uso de RNAs para esse tipo de aplicação tem sido frequente nos trabalhos pesquisados na literatura, devido, principalmente, à sua capacidade de aprendizado (CHIDDARWAR; BABU, 2010). Por meio do treinamento, a RNA é capaz de identificar a relação cinemática entre o sistema de coordenadas das juntas e a posição final da ferramenta e apresentar uma boa generalização de resposta.

1.2 OBJETIVOS

- Resolver (Mapear) a cinemática inversa para um manipulador robótico usando Redes Neurais Artificiais;
- Fazer simulações de RNAs para o aprendizado da cinemática inversa, delimitadas a um conjunto de pontos distribuídos no volume de trabalho e de trajetórias;

1.3 CONTRIBUIÇÕES

- Aplicar as soluções encontradas em um protótipo de um manipulador robótico com 5 GDL usando a placa de desenvolvimento, Intel® Galileo Gen 2;
- Contribuir para as pesquisas na área de robótica apresentando uma metodologia alternativa para o problema de cinemática inversa de manipuladores.

1.3 ORGANIZAÇÃO DO TEXTO

O texto encontra-se divido em sete capítulos, onde no capítulo 2, apresenta-se uma revisão dos principais artigos pesquisados sobre o assunto de Redes Neurais Artificiais aplicados a manipuladores robóticos. Apresentam-se no capítulo 3 os conceitos construtivos fundamentais e a abordagem cinemática e o problema do equacionamento da cinemática inversa para manipuladores robóticos. No capítulo 4 são descritos os principais conceitos de Redes Neurais Artificiais e o tipo de Rede Neural utilizada no trabalho. No capítulo 5 são descritas as etapas realizadas antes da implementação do algoritmo no protótipo, tais como, a geração dos padrões de treinamentos, algoritmos, simulações e seus resultados. No capítulo 6 são abordados os detalhes do hardware e software do protótipo do manipulador e o resultado

de uma simulação de trajetória. Finalmente, no capítulo 7 são feitas as conclusões e sugestões para trabalhos futuros.

CAPÍTULO 2. REVISÃO DE LITERATURA

Neste capítulo apresenta-se uma revisão de literatura, em ordem cronológica, de alguns dos mais relevantes trabalhos pesquisados sobre a utilização das Redes Neurais Artificiais (RNA) e suas características, aplicadas na resolução da cinemática inversa de manipuladores robóticos.

2.1 METODOLOGIAS COM RNAS PURAS PARA SOLUÇÃO DA CINEMÁTICA INVERSA

A utilização de RNA para solucionar a cinemática inversa de manipuladores não é um assunto recente, porém, ainda são muitos os trabalhos encontrados. Descrevem-se os mais relevantes e que deram embasamento para esta dissertação.

Em Morris e Mansor (1997) foram utilizadas Redes Neurais Artificiais de três camadas com treinamento via retropropagação ou *backpropagation* (BP), para a análise da cinemática inversa de manipuladores de dois e três graus de liberdade. Com o algoritmo de treinamento computacionalmente muito intenso para os computadores da época e para informações recorrentes, foram utilizadas tabelas de consultas (ou *lookup tables*), de forma que os resultados para o sistema foram considerados ótimos.

Barreto et al. (2002) usam uma rede Competitiva e Hebbiana Temporal para aprendizagem e reprodução de trajetórias. Utilizaram informação de contexto temporal para resolver os casos que ocorrem incertezas, ou seja, ambiguidades durante a fase de reprodução das sequências de trajetórias armazenadas. A rede passou por simulações e posteriormente foi implementada no robô PUMA 560. Comparações da performance do sistema de controle são realizadas com a performance de outras Redes Neurais não supervisionadas temporais. De forma geral, esta RNA consegue reproduzir as trajetórias de itens repetidos com precisão e com menor número de neurônios. Também é enfatizado que esta RNA faz uso eficiente de memória e possui certa robustez, ou seja, é tolerante a ruídos e falhas. Os autores classificaram sua aplicação como plausível.

Em Dávila e Read (2004) foi resolvido o problema de posicionamento de um manipulador comercial de 5 graus de liberdade, SCORBOT-ER II, via cinemática inversa, ao utilizar uma Rede Neural com topologia de 3 entradas, 2 camadas ocultas com 15 neurônios

cada e 4 neurônios na camada de saída. O treino desta RNA se deu pelo algoritmo de retropropagação, com os padrões obtidos pela cinemática direta, através da sistemática de Denavit-Hartenberg (D-H). Apesar do manipulador utilizado possuir 5 GDL, os autores desconsideram um grau de liberdade, pois este não possui influência no posicionamento do mesmo. Após o treinamento, a RNA demonstrou ter boa eficiência para disponibilizar os ângulos de juntas necessários para que o manipulador percorra, com exatidão considerável, uma trajetória simulada.

No artigo de Hasan et al. (2006) foi utilizado uma RNA *feedforward* com treinamento via *backpropagation* para controlar o movimento e solucionar a cinemática inversa de um manipulador comercial FANUC M-710i que possui 6 GDL. Neste artigo, os autores utilizaram dois conjuntos de dados – oriundos de duas trajetórias - para treinamento e validação da RNA. Para o primeiro conjunto, as médias absolutas em porcentagem das juntas, ficaram abaixo de 2,8% enquanto que, para o segundo, ficaram abaixo de 6,1%. Consideram que o uso de RNA pode ser vantajoso em relação à métodos de controle que dependam da modelagem do sistema robótico, desde que a RNA consiga alcançar um conjunto de pesos que proporciona, a partir de um conjunto de entradas, as saídas desejadas ou suficientemente próximas, respeitando uma certa tolerância.

Nóbrega Sobrinho (2011) utiliza a implementação de controladores inteligentes com RNA multicamada do tipo direto com treinamento via algoritmo de *backpropagation* realizado de forma *offline*, para o controle de posição de máquinas ferramentas em uma mesa de coordenadas com dois graus de liberdade, cujos resultados são considerados satisfatórios, pelo autor. O acionamento é realizado por motores de indução trifásicos, alimentados por inversores de frequência; os ensaios de acionamento são realizados com sinais tipo degrau para simular situações ponto a ponto e sinais tipo seno e cosseno para simular acompanhamento de trajetória. No primeiro caso, houve apenas um único erro de sobressinal de 1,245%, não houve erro de regime permanente e o tempo máximo de assentamento foi de 11,685 segundos. No segundo caso o erro máximo foi de 4,5% para um sinal com período de 20 segundos, sendo que sinais com maiores períodos obtiveram melhores resultados.

Semelhante ao trabalho de Nóbrega Sobrinho (2011), Lima (2012) utiliza RNA multicamadas com treinamento via *backpropagation*, que emula um controlador PD2 (Proporcional Derivativo Duplo), para controlar a posição de um manipulador robótico com 2 graus de liberdade, acionado por motores de indução e inversores de frequência. Entretanto, para determinar o número de neurônios na camada oculta (ou intermediária) foi utilizado um Algoritmo Genético (AG). As avaliações também foram realizadas sem simulação e de modo

experimental com sinais do tipo degrau e senoidais. Para o acionamento de controle de posição, os erros máximos foram de 0,11% e 0,1% para o braço e a base, respectivamente. Para o acompanhamento de trajetórias, os erros máximos foram respectivamente de 1,09% e 0,997%, para a base e o braço. Logo, o autor considerou como razoável a resposta do sistema para o controle do manipulador.

Em Camargo, Veraszto e Barreto (2014) foi apresentado um trabalho com finalidades didáticas utilizando RNAs para simular previamente o comportamento cinemático inverso de um manipulador robótico de dois graus de liberdade. Neste, desenvolveram o algoritmo com a Rede Perceptron Multicamadas (RPM) ou do inglês *Multilayer Perceptron* (MLP) composta por duas camadas intermediárias. Os resultados foram apresentados em forma de tabela na qual é possível perceber que os erros de cada ângulo de junta para cada padrão de treinamento se mantiveram relativamente baixos, portanto, a precisão que a RNA alcançou foi considerada satisfatória pelos autores.

Fernandes Junior (2014) usou em sua dissertação RNAs e AG para resolução da cinemática inversa de um manipulador robótico educacional com cinco GDL. Para o treinamento da Rede Neural, o espaço de trabalho do manipulador foi delimitado para evitar cálculos desnecessários durante o treinamento. Usou como padrão de treinamento as informações da cinemática direta - na entrada da Rede Neural os valores em coordenadas cartesianas, na saída, as variáveis das juntas. Obteve como resultados duas soluções para a Rede Neural, uma considerando quatro GDL (solução simplificada) e outra considerando cinco GDL (solução completa) do problema. Nesta utilizou ainda o AG para minimizar a distância quadrática entre o ponto atual da trajetória e o próximo ponto desejado. Novamente considerou dois testes, um cenário simples e outro mais complexo. O algoritmo foi aplicado em um manipulador educacional e usando como controlador a placa Arduino Uno Rev2. O autor classifica os resultados como satisfatórios e ressalta que o tempo de treinamento para determinar as melhores soluções é proibitivo no caso de uma aplicação em tempo real, embora possam ser aplicadas de modo *offline*.

Em Raj, Raglend e Anand (2015) teve-se como proposta a resolução da cinemática inversa de um manipulador robótico de cinco graus de liberdade por RNAs dos tipos *Feedfoward* e Função de Base Radial (em inglês, *Radial-Basis Function*). Ambas as Redes Neurais implementadas são compostas por três entradas e cinco saídas, referentes às coordenadas cartesianas e aos valores dos ângulos das juntas, respectivamente. Nos testes são apresentados dois padrões de treinamento para cada Rede Neural e os desempenhos destas são

comparados pelo gráfico de desempenho, épocas x erro quadrático médio e, neste caso, a Rede de Função de Base Radial obteve um melhor resultado.

Atualmente, os trabalhos envolvendo Redes Neurais Artificiais misturam técnicas de maneira a proporcionar o aumento do desempenho do sistema, sendo chamados de algoritmos híbridos – ou redes hibridas, ou seja, além dos conceitos provenientes de RNAs, utilizam outros algoritmos, tais como, algoritmo genético, Lógica *Fuzzy* ou alguma outra heurística de otimização e convergência. Apresentam-se a seguir, alguns exemplos destas técnicas, pesquisados.

2.2 METODOLOGIAS COM RNAS HÍBRIDAS E OUTRAS ABORDAGENS, PARA SOLUÇÃO DA CINEMÁTICA INVERSA

Como exemplo de um sistema híbrido tem-se o trabalho de Alavandar e Nigam (2008) que utilizam duas simulações computacionais para representar a eficiência da Rede ANFIS (Adaptive Neuro-Fuzzy Inference System) para manipuladores robóticos planares de 2 e 3 GDL. Segundo os autores, este método proporcionar uma boa convergência e um número reduzido de iterações e resulta em valores aceitáveis na diferença entre os ângulos obtidos e os previstos. Indicam métodos alternativos para melhorar o valor do erro.

Daya, Khawandi e Akoum (2010) utilizam em seu artigo, a técnica de agrupar seis RNAs do tipo MLP treinada com algoritmo de *backpropagation* para solucionar a cinemática inversa de um manipulador de dois graus de liberdade. Neste, cada RNA é usada para cada quadrante do espaço de trabalho do braço robótico, no total de quatro Redes Neurais; a quinta MLP é designada a classificar se a posição está ou não em uma região acessível do espaço de trabalho, e uma sexta para verificar as limitações de juntas do manipulador. Consideram, pelos resultados obtidos, que a proposta abordada pode ser uma estratégia para ser usada em futuros trabalhos.

Feng, Yao-Nan e Yi-Min (2012) utilizam um algoritmo de treinamento, chamado de Mecanismo de Aprendizado Rápido, para o treinamento de uma Rede Neural de única camada escondida com alimentação de entrada - ou do termo em inglês, *Single Hidden Layer Feedforward Neural Networks.* Para a aquisição do conjunto de treinamento sem múltiplas soluções da cinemática inversa é utiliza o algoritmo heurístico *Electromagnetism-like Method* (EM). Pelos testes realizados concluíram que, o algoritmo híbrido apresentou uma melhor velocidade de aprendizagem quando comparado aos algoritmos tradicionais, além de um melhor desempenho de generalização dos problemas da cinemática inversa.

No artigo de Pinheiro, Da Trindade e Pantoja (2013) foi utilizado o método numérico interativo da Jacobiana pseudo-inversa para solucionar o problema da cinemática inversa de um protótipo de um manipulador de quatro graus de liberdade. Por meio de simulações mostram que o método é eficaz, todavia, alertam que, para soluções em tempo real, o peso computacional deve ser considerado. Posteriormente, o algoritmo foi implementado em um protótipo de um manipulador. Sugerem para a continuação do trabalho o uso de Inteligência Artificial, como por exemplo, as RNAs para a minimização do erro e o tempo de resposta das soluções.

Em Jha, Biswal e Sahu (2014) são realizadas simulações com uma rede híbrida composta por uma MLP e o algoritmo de pesquisa gravitacional para resolver a cinemática inversa de robô manipulador PUMA, que possui 6 GDL. Neste trabalho tem-se como proposta a minimização do erro quadrático médio da solução. O algoritmo implementado é comparado com uma MLP com treinamento via *backpropagation*. A partir dos resultados é possível perceber que o algoritmo híbrido possui uma melhor taxa de convergência e um menor erro quadrático médio, o que significa uma RNA mais eficiente.

2.3 COMENTÁRIOS

Destaca-se desta revisão bibliográfica que a maioria dos trabalhos que envolvem aplicações de RNA para solucionar a cinemática inversa de um manipulador robótico utiliza para o aprendizado, trajetórias específicas, ou seja, um conjunto de pontos intermediários que interpolam o início e o final do percurso a ser percorrido. Essa metodologia é encontrada nos trabalhos de Morris e Mansor (1997), Barreto et al (2002), Hasan et al. (2006), Nóbrega Sobrinho (2011) e Lima (2012).

Por outro lado, são poucos os trabalhos que visam à utilização de RNA para mapeamento de entrada e saída para todo volume de trabalho (ou parte dele) de um manipulador robótico. Nos trabalhos de Camargo, Veraszto e Barreto (2014) e Fernandes Junior (2014) são utilizados um conjunto de padrões de treinamento derivados de pontos oriundos de todo o volume de trabalho de manipuladores de dois e cinco graus de liberdade, respectivamente.

Baseado na literatura estudada, especialmente, o artigo de Daya; Khawandi e Akoum (2010) implementa-se neste trabalho um conjunto de RNAs do tipo MLP configuradas em paralelo, para o mapeamento da cinemática inversa aplicada em trajetórias e em pontos

distribuídos em todo o espaço de trabalho considerado, para um protótipo de um manipulador robótico de 5 GDL.

CAPÍTULO 3. MANIPULADORES ROBÓTICOS: ASPECTOS CONSTRUTIVOS E MODELAGEM CINEMÁTICA

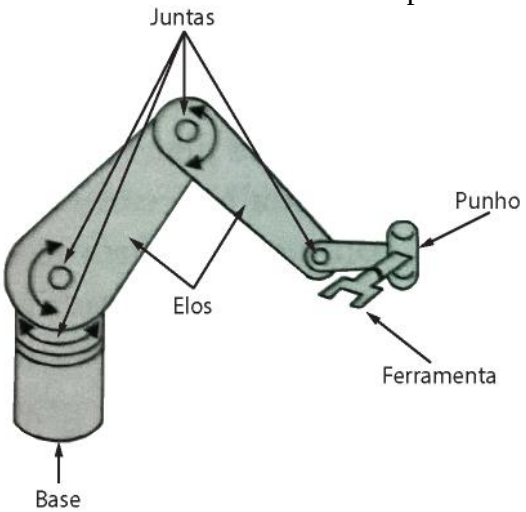
Faz-se neste capítulo uma abordagem sobre os principais conceitos que envolvem a construção de manipuladores robóticos, a modelagem da cinemática direta e inversa, utilizando como exemplo um manipulador robótico de 3 GDL. Na cinemática direta apresenta-se o método geométrico e pela sistemática de Denavit–Hartenberg, na cinemática inversa, os métodos geométricos e algébricos.

3.1 ASPECTOS CONSTRUTIVOS DOS MANIPULADORES ROBÓTICOS

Um manipulador robótico é considerado um robô industrial, programável e multifuncional, sendo projetado para inúmeras tarefas tais como, mover materiais, peças, ferramentas ou dispositivos específicos em movimentos variáveis.

De maneira geral, os manipuladores são montados sobre uma base fixa e formados por uma sequência de elos e juntas. O elo é a estrutura rígida e não móvel e a junta é a junção entre dois – ou mais – elos. Na extremidade da última junta tem-se o punho, no qual são montados os efetuadores que são ferramentas adequadas à realização de trabalhos específicos para os quais o manipulador foi construído. Na Figura 1 apresentam-se os elementos básicos de um manipulador robótico.

Figura 1 - Elementos básicos de um manipulador robótico.



Fonte: Adaptado de Niku (2010).

As juntas são a parte de maior atrito no manipulador robótico, não somente possibilitam a ligação de dois elos, mas também a liberdade de movimentação do braço, chamado de Grau de Liberdade.

3.1.1 Tipos de Juntas

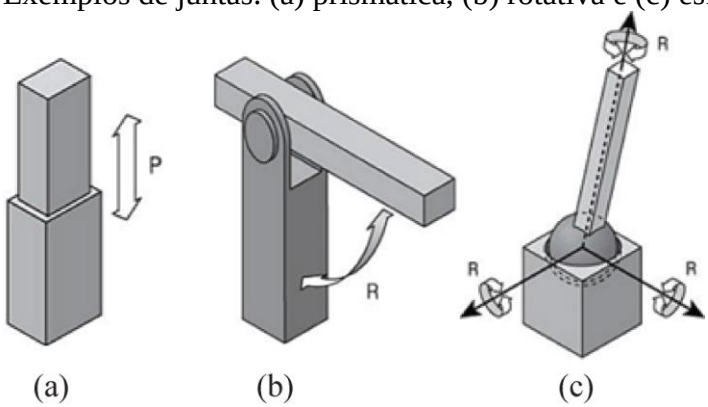
Existem 3 tipos de juntas principais: prismática (ou linear), rotativa (ou rotacional) e a junta esférica, ilustradas pela Figura 2. Porém, outros tipos de juntas podem ser obtidos a partir de juntas prismáticas e/ou rotativas.

A junta do tipo prismática ou linear, Figura 2(a), possibilita um grau de liberdade e proporciona a movimentação retilínea, é composta geralmente de duas hastes deslizantes (uma encaixada dentro da outra). Possui mais frequente aplicação em manipuladores por acionamentos hidráulicos e pneumáticos.

Rotativa ou rotacional, gira em torno de uma linha imaginária fixa, denominada eixo rotacional possibilitando 1 grau de liberdade, Figura 2(b).

A junta esférica é menos utilizada do que as outras duas em razão da sua dificuldade de acionamento e construção. Possibilita rotação em torno de 3 eixos, portanto, tem três GDL. Se for uma junta esférica passiva, é composta por um elo contendo uma parte esférica na ponta e outro elo contendo uma formação côncava para encaixe da parte esférica, mostrada na Figura 2(c). Se for uma junta esférica ativa, pode ser substituída por três juntas rotativas, desde que os três eixos de rotação se intersecionem, apresentando a mesma performance.

Figura 2 - Exemplos de juntas. (a) prismática, (b) rotativa e (c) esférica.



Fonte: Adaptado de Rosário (2005).

3.1.2 Graus de Liberdade e Volume de Trabalho

Os graus de liberdade são dados pela configuração de suas juntas, ou seja, são a quantidade de movimentos independentes que caracterizam o posicionamento completo de um objeto. O número de grau de liberdade total de um manipulador robótico é igual a somatória dos graus de liberdade concedidos pelas suas juntas.

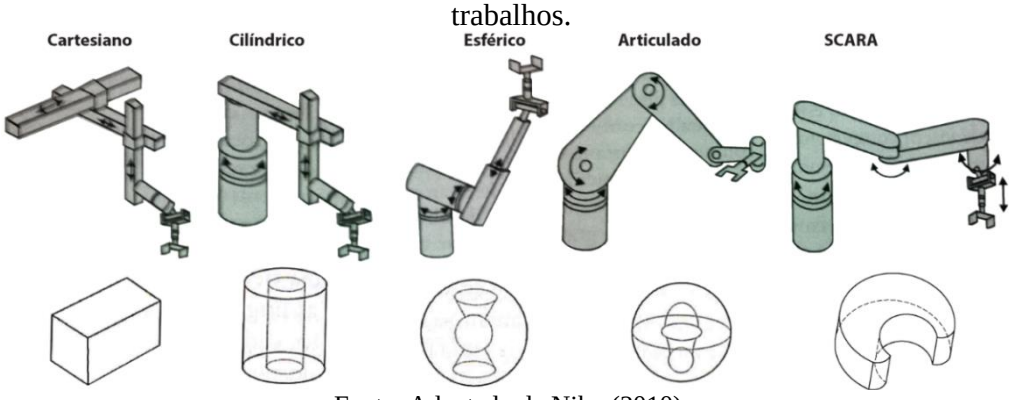
Quanto ao volume de trabalho, diz-se para o conjunto de todos os pontos do espaço que o punho de um robô pode alcançar, sendo seu limite definido pelo projeto estrutural do braço, pelo limite de movimento das juntas e o tamanho dos componentes do corpo, braço e punho. Convenciona-se utilizar o punho como referência para que não haja influência dos diferentes tamanhos de efetuadores que possam ser acopladas ao robô. Exemplos do volume de trabalho são mostrados na Figura 3, juntamente com as configurações básicas dos manipuladores robóticos.

Os manipuladores robóticos industriais possuem, normalmente, seis graus de liberdade para posicionar e orientar seus efetuadores em qualquer local dentro de seu volume de trabalho. Um braço robótico que contenha cinco ou menos graus de liberdade é limitado quanto a situar ou orientar seu órgão terminal.

3.1.3 Configurações Básicas dos Manipuladores Robóticos

Quanto as configurações, os robôs manipuladores são classificados de acordo com sua forma geométrica que considera as três juntas mais próximas à base. Existem cinco configurações principais de manipuladores, a cartesiana, cilíndrica, esférica (ou polar), articulada (ou de revolução) e SCARA, listadas também na Figura 3.

Figura 3 - Configurações básicas de manipuladores robóticos e seus respectivos volumes de trabalhos.



Fonte: Adaptado de Niku (2010).

O robô cartesiano realiza apenas movimentos lineares pois, trata-se de um robô que possui três juntas prismáticas. Apresenta um controle simples, pequeno volume de trabalho, mas com uma alta exatidão, além de um alto grau de rigidez mecânica.

O robô cilíndrico recebe esse nome pela forma do seu volume de trabalho, sendo constituído por uma junta rotacional e duas prismáticas. Essa configuração proporciona um volume de trabalho maior do que o de coordenadas cartesianas, todavia, é ligeiramente mais fraco na perspectiva da rigidez mecânica. Em consequência à presença da junta rotacional torna o controle um pouco mais complicado.

O esférico (ou polar) é constituído de duas juntas rotacionais e uma prismática, e assim como o cilíndrico, seu nome tem origem na forma de seu volume de trabalho. Comparando às outras configurações, esta possui um maior volume de trabalho, controle mais complexo e menor rigidez mecânica.

A quarta configuração, robô articulado ou robô de revolução, é constituída por três juntas de revolução, frequentemente utilizada por ter o maior volume de trabalho, do que as outras configurações. Apresenta um controle complexo e uma baixa rigidez mecânica, principalmente, em razão das variações do momento de inércia e carga.

O robô SCARA (*Selective Compliance Assembly Robot*) possui duas juntas de revolução e uma prismática, semelhante à configuração de robôs esféricos. É comumente utilizada para tarefas onde é necessária alta velocidade e precisão, como tarefas de *pick-and-place* e montagem.

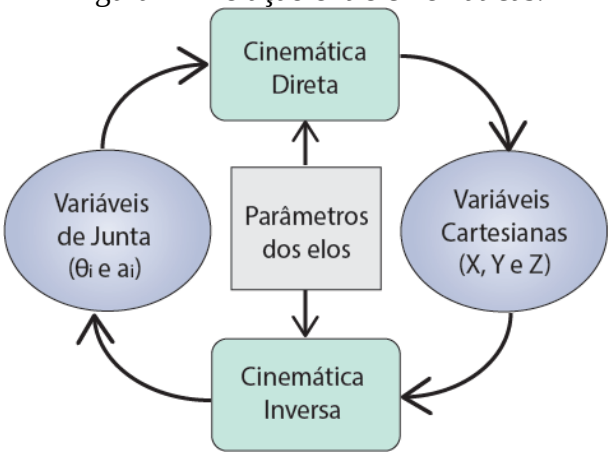
3.2 MODELAGEM CINEMÁTICA

De acordo com Craig (2012), a cinemática é um ramo da mecânica que estuda o posicionamento e a movimentação de corpos, sem ponderar as forças e os momentos presentes. Na robótica é possível, a partir do estudo cinemático e das características geométricas de cada elo do manipulador, relacionar o sistema de referência da ferramenta do manipulador com o sistema de referência fixo do espaço em que este está inserido.

Existem duas formas de tratar a cinemática de manipuladores: por meio da cinemática direta ou da cinemática inversa. Conhecendo as variáveis de cada junta e os parâmetros dos elos, a partir da cinemática direta é possível determinar a posição da ferramenta em um sistema de coordenadas cartesianas. Por outro lado, na cinemática inversa, sabendo-se a posição da ferramenta e os parâmetros dos elos é possível determinar uma

configuração das variáveis das juntas, que satisfaça o correto posicionamento do manipulador, representa-se a relação entre elas na Figura 4.

Figura 4 - Relação entre cinemáticas.



Fonte: Elaborada pelo autor.

Em manipuladores com juntas rotativas, existem variações em cada ângulo de junta (θ_i) e os comprimentos dos elos (a_i) são fixos. Por outro lado, em manipuladores com juntas prismáticas, as variações ocorrem apenas no comprimento das juntas. Neste trabalho tem-se o interesse no estudo de juntas rotativas, por serem mais usuais e onde o problema da cinemática inversa se torna mais complexo.

Antes de abordar os conceitos da cinemática direta e inversa, é necessário o embasamento sobre o sistema de posicionamento e transformações elementares.

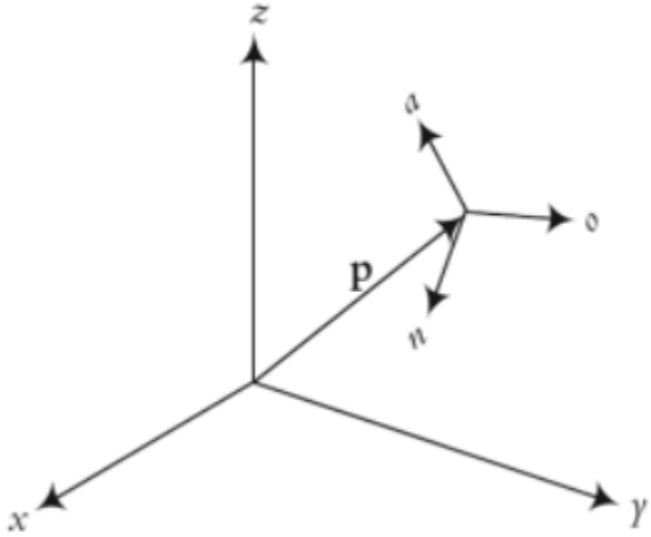
3.2.1 Sistema de Posicionamento e Orientação no Espaço

Com base na origem de um sistema de referência é possível localizar a posição de um ponto P no espaço tridimensional mediante a um vetor de três elementos, referentes as coordenadas x, y e z, dado na forma matricial por,

$$\boldsymbol{P} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{1}$$

Segundo Niku (2010) e com base na Figura 5, a orientação de um referencial no espaço é representada pela relação de coeficientes angulares (n, o e a) ao três elementos de coordenadas cartesianas de um referencial qualquer (fixo ou não), equação (2). A notação destes coeficientes é oriunda das palavras normal, orientação e abordagem, respectivamente.

Figura 5 - Referencial relacionado a um referencial fixo.

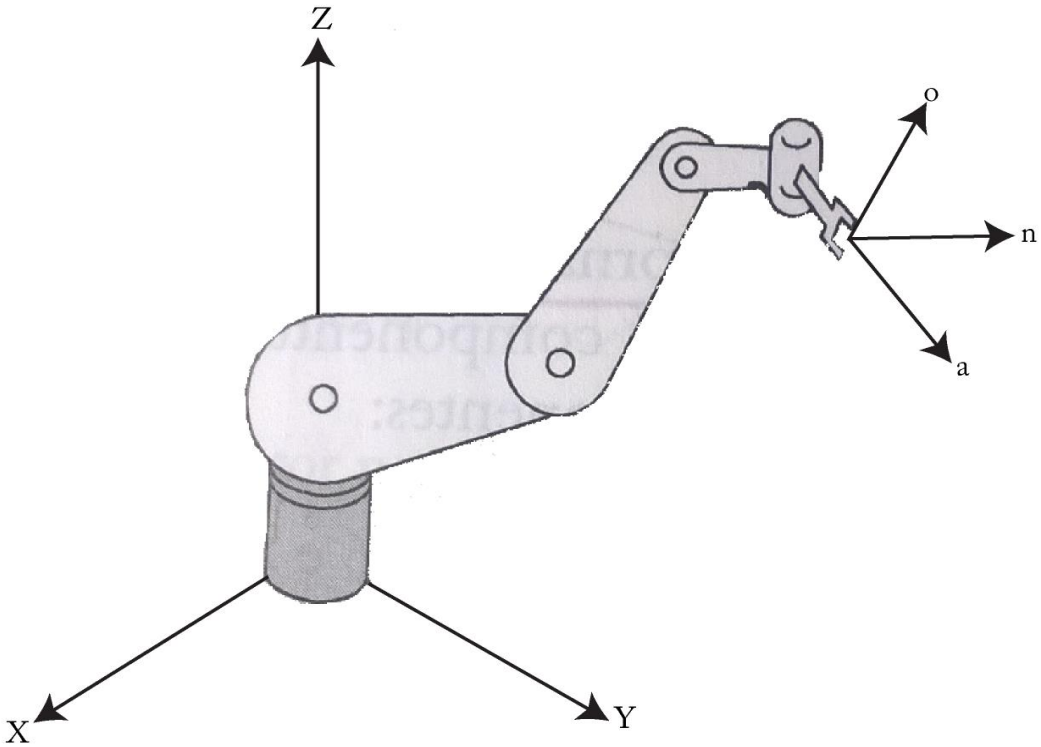


Fonte: Niku (2010)

$$F = \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} \quad (2)$$

A abordagem é realizada ao longo do eixo z da pinça, a orientação se dá em torno do eixo y e a normal entre estes está relacionada ao eixo x, conforme Figura 6.

Figura 6 - Eixos, normal, orientação e abordagem da garra do manipulador.



Fonte: Adaptado de Niku (2010)

Para representar um referencial em relação a outro de forma completa (ou até mesmo um referencial da origem), tanto a localização da sua origem como as direções de seus eixos devem ser especificados. Isto é possível ao se adicionar o vetor de posição representado pela equação (1), aos elementos da equação (2). Uma quarta linha com quatros elementos que representam fatores de perspectiva e escala é adicionada, completando a matriz homogênea 4x4, representada pela equação (3) (NIKU, 2010).

$$F = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Portanto, na matriz F, o referencial pode ser expresso por três vetores que descrevem seus vetores direcionais unitários e um quarto vetor que descreve a sua direção.

3.2.2 Transformações no Espaço

Uma transformação é definida como a realização de um movimento no espaço. Quando um referencial, por exemplo, um robô se move no espaço em relação a um sistema de referência fixo, representa-se este movimento de forma similar a uma representação de referencial. Uma transformação representa uma mudança na localização e orientação do robô (NIKU, 2010). A transformação pode ser por translação pura, rotação pura ou um movimento combinado de translações e rotações.

— **Representação de uma translação pura**

A transformação é considerada uma translação pura se um referencial (representando um robô) se move no espaço sem qualquer mudança na sua orientação. Neste caso, os vetores unitários direcionais permanecem na mesma direção, apenas a localização da origem do referencial é alterada, logo, os elementos de posição indicados pelo índice p variam, conforme a matriz (4), a seguir,

$$Trans = \begin{bmatrix} 1 & 0 & 0 & \Delta_x \\ 0 & 1 & 0 & \Delta_y \\ 0 & 0 & 1 & \Delta_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Onde as três primeiras colunas representam nenhum movimento rotacional (matriz identidade) e Δx, Δy, Δz são os três componentes de uma translação pura do vetor Δ em relação a x, y e z do sistema de referência.

A representação do novo referencial na forma matricial, pode ser encontrada pré-multiplicando-se o referencial por uma matriz que representa a transformação. A nova localização da estrutura será dada pela equação (5) e simbolicamente escrita como na equação (6),

$$F_{nova} = Trans(\Delta_x, \Delta_y, \Delta_z) \cdot F_{velha} \tag{5}$$

$$F_{nova} = \begin{bmatrix} 1 & 0 & 0 & \Delta_x \\ 0 & 1 & 0 & \Delta_y \\ 0 & 0 & 1 & \Delta_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & \Delta_x + p_x \\ n_y & o_y & a_y & \Delta_y + p_y \\ n_z & o_z & a_z & \Delta_z + p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Portanto, depois de uma translação pura os vetores de direção permanecem os mesmos, mas a nova localização do referencial é em $\Delta + \boldsymbol{p}$.

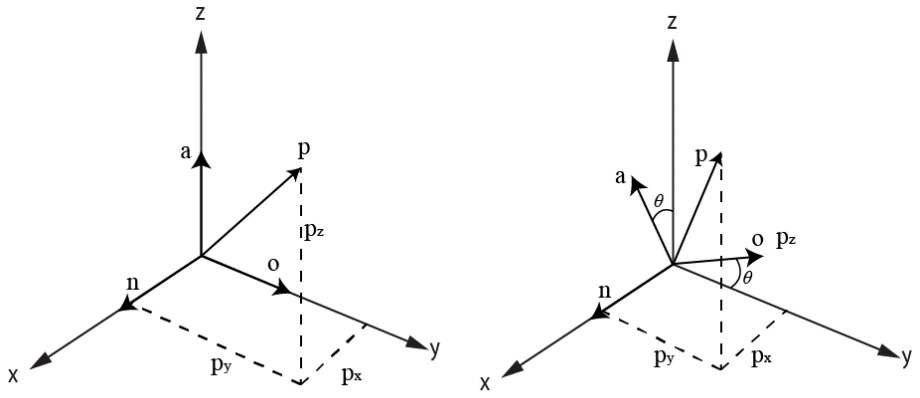
— **Representação de uma rotação pura em torno de um eixo**

Para simplificar a derivação de rotações em torno de um eixo, supõem-se que:

- O referencial está na origem do sistema de referência e é paralelo a ele;
- Um referencial F_{noa} localizado na origem do sistema de referência F_{xyz} gira de um ângulo θ em torno do eixo *x* do sistema de referência;
- Ligado ao sistema rotacional F_{noa} há um ponto p, com coordenadas p_x, p_y e p_z em relação ao sistema de referência e p_n, p_o e p_a em relação ao referencial em movimento.

O ponto p ligado ao referencial irá rodar com ele, à medida que o referencial gira em torno do eixo x. Antes da rotação, as coordenadas do ponto nos dois referenciais são as mesmas, após a rotação, as coordenadas p_n, p_o e p_a do ponto permanecem as mesmas no referencial girante F_{noa} , mas p_x, p_y e p_z serão diferentes no referencial F_{xyz}, conforme ilustrado pela Figura 7.

Figura 7 - Coordenadas de um ponto em um referencial rotativo antes e depois da rotação.



Antes da rotação Após a rotação

Fonte: Adaptado de Niku (2010)

As novas coordenadas do ponto em relação ao sistema de referência fixo, depois do referencial móvel ter girado em forma matricial é dada pela equação (7),

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sen\theta \\ 0 & sen\theta & cos\theta \end{bmatrix} . \begin{bmatrix} p_n \\ p_o \\ p_a \end{bmatrix} \tag{7}$$

A equação indica que as coordenadas do vetor $\boldsymbol{p}$ no novo referencial (girado) devem ser pré-multiplicadas pela matriz de rotação, para obter as coordenadas no referencial de referência. Esta matriz de rotação é válida apenas para uma rotação pura em torno do eixo x do sistema de referência e é denotada pela expressão (8),

$$p_{xyz} = Rot\ (x, \theta) \cdot p_{noa} \tag{8}$$

O mesmo desenvolvimento pode ser feito para a rotação de um referencial (fixo) em relação aos eixos y e z do referencial móvel, de forma a encontrar as equações (9) e (10),

$$Rot(Y, \theta) = \begin{bmatrix} cos\theta & 0 & sen\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sen\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

$$Rot(Z, \theta) = \begin{bmatrix} cos\theta & -sen\theta & 0 & 0 \\ sen\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

— **Representação de transformações combinadas**

Nas transformações combinadas uma série de translações e rotações sucessivas acontecem em relação aos eixos do sistema de referência fixo ou o movimento dos eixos do referencial atual.

Uma matriz homogênea que represente este tipo de combinado de movimento pode ser obtida pela multiplicação de suas combinações de transformações elementares, representadas pelas equações (6), (7), (9) e (10), na mesma sequência em que as transformações ocorrem.

3.2.3 Análise Geométrica e a Sistemática de Denavit – Hartenberg para representar equações da cinemática direta de robôs

A cinemática direta é utilizada quando se deseja conhecer a posição da ferramenta terminal do manipulador, por meio das configurações dos valores dos seus elos e juntas. A resolução de problemas que tratam da cinemática direta de manipuladores é considerada simples se comparada ao da forma inversa.

Sabendo-se os dados de cada elo e junta, e a posição da base do manipulador é possível obter a modelagem da cinemática direta. Existem diversas maneiras de realizar essa análise, entre elas estão: o modelamento geométrico, vetores locais e a sistemática de Denavit–Hartenberg ou parâmetros de D-H (ROSÁRIO, 2005).

Na modelagem de um robô por meio de D-H obtém-se a posição e a orientação finais do elemento terminal. Por outro lado, utilizando vetores locais obtém-se as posições e as orientações de diversos pontos de interesse que podem ser utilizados na construção gráfica do robô por meio de elementos primitivos.

Para as próximas análises, definem-se como notação inicial alguns parâmetros relacionados ao manipulador robótico, tais como,

— Juntas, J_i;

— Ângulos das juntas rotativas, θ_i;

— Comprimento de cada elo, a_i, onde $i=1, 2, ...n$;

— Número de articulações ou graus de liberdade, n.

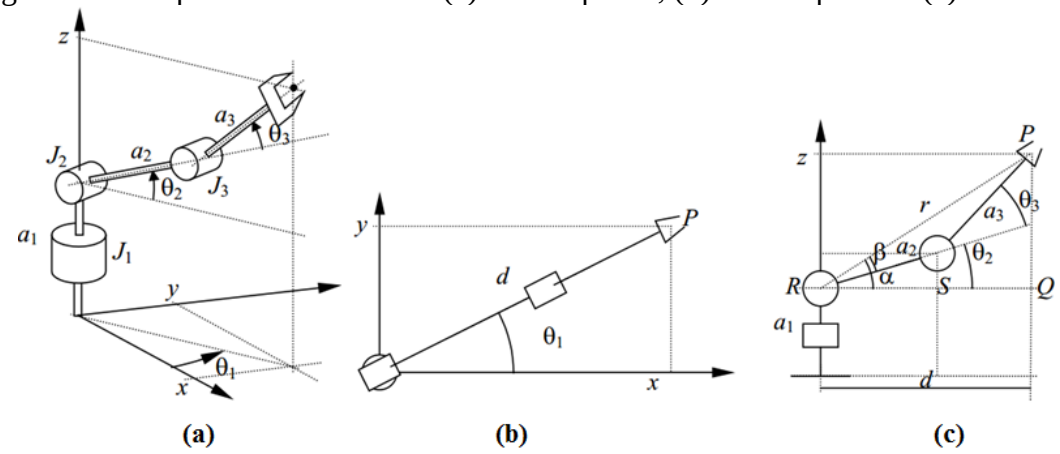
É possível determinar a cinemática direta de um manipulador robótico de 3 GDL de forma geométrica, conforme configuração dada na Figura 8, com todos os seus parâmetros.

— **Método geométrico para a cinemática direta**

Nesta figura, a junta J1 gira em torno do eixo z, enquanto as juntas J2 e J3 giram em torno do plano x, y; a distância do ponto P em termos do plano x, y é representada por d, Figura 8(b), que pode ser determinada por meio da vista lateral, Figura 8 (c), e conforme equação (11),

$$d = a_2 \cdot \cos(\theta_2) + a_3 \cdot \cos(\theta_2 + \theta_3) \tag{11}$$

Figura 8 - Manipulador com 3GDL. (a) vista espacial, (b) vista superior e (c) vista lateral.



Fonte: Adaptada de Carrara (2009).

Sabendo a equação de *d* pode-se obter por trigonometria, as coordenadas do ponto *P* que indicam a posição da ferramenta terminal do manipulador robótico por trigonometria, também usando as Figura 8(a) e (b) dadas pelas equações (12), (13) e (14),

$$x = d \cdot \cos(\theta_1) = (a_2 \cdot \cos(\theta_2) + a_3 \cdot \cos(\theta_2 + \theta_3)) \cdot \cos(\theta_1) \tag{12}$$

$$y = d \cdot \text{sen}(\theta_1) = (a_2 \cdot \cos(\theta_2) + a_3 \cdot \cos(\theta_2 + \theta_3)) \cdot \text{sen}(\theta_1) \tag{13}$$

$$z = a_1 + a_2 \cdot sen(\theta_2) + a_3 \cdot sen(\theta_2 + \theta_3) \tag{14}$$

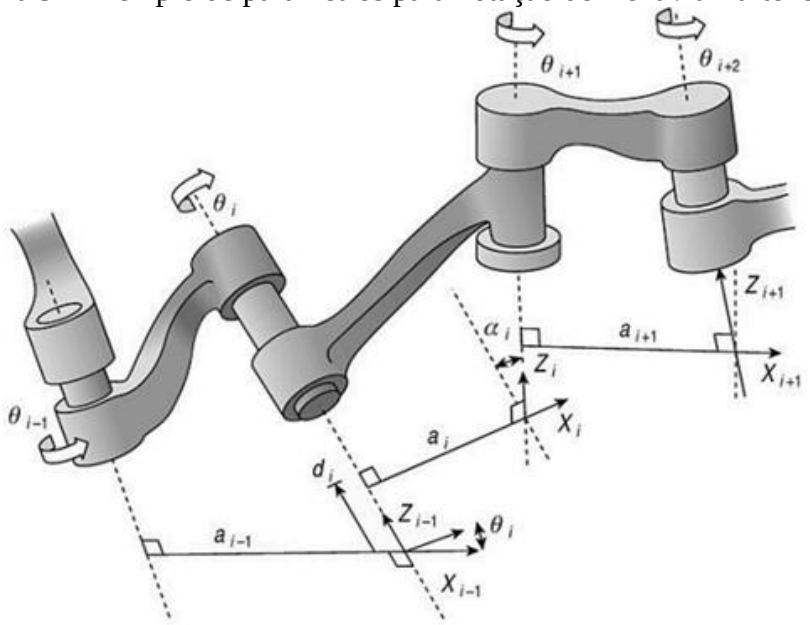
Para manipuladores com mais de 3 GDL, a cinemática direta via solução geométrica se torna mais complexa, desta forma, nesses casos, normalmente é recomendada a solução por

meio da sistemática de D-H que é uma ferramenta utilizada como um padrão de representação de manipuladores robóticos.

A sistemática de D-H para resolver a cinemática direta modela as equações de movimentos de manipuladores robóticos de n graus de liberdade, a partir dos dados dos elos e juntas, independentemente de quão complexo é a configuração do robô. A partir desta sistemática, além da análise de movimentos é possível realizar também a análise dinâmica, de força, etc. (NIKU, 2010).

Para modelar um robô com a sistemática de D-H deve-se primeiramente atribuir sistemas de referências de coordenadas cartesianas para cada junta, conforme mostrado na Figura 9. Se a junta é do tipo rotacional, o eixo Z_{i-1} é alocado ao longo do movimento da junta i em conformidade com a regra da mão direita, para rotações. Se a junta for do tipo prismática, este eixo é alocado ao longo do movimento linear. O eixo X_i é a normal comum entre os eixos Z_{i-1} e Z_i e apontado para fora de Z_{i-1}. O eixo Y_i completa o sistema conforme a regra da mão direita, porém, esse último não é utilizado na representação de D-H (ROSÁRIO, 2005).

Figura 9 - Exemplo de parâmetros para notação de Denavit-Hartenberg.



Fonte: Rosário (2005)

A definição de cada parâmetro de D-H utilizado é dada por:

- θ_i, ângulo da junta obtido entre os eixos X_{i-1} e X_i no eixo Z_{i-1};
- d_i, distância entre a origem do *(i-1)*-ésimo sistema de coordenadas até a intersecção do eixo Z_{i-1} com o eixo X_i ao longo do eixo Z_{i-1};
- a_i, a menor distância entre os eixos Z_{i-1} e Z_i (por meio do eixo X_i);

- α_i, ângulo entre os eixos Z_{i-1} e Z_i medidos no eixo X_i.

Por intermédio das definições dos quatro parâmetros apresentados, destaca-se que os parâmetros θ_i e d_i estão relacionados com os aspectos construtivos da junta i, enquanto que, os parâmetros a_i e α_i estão relacionados com os aspectos construtivos do elo i. Adicionalmente, se a junta i for do tipo rotacional, θ_i será a variável articular, enquanto que d_i será a variável articular de uma junta do tipo prismática.

Mostra-se na Tabela 1 a representação dos parâmetros de D-H. Esta tabela se torna uma ferramenta muito útil para a montagem da matriz de transformações homogêneas.

Tabela 1 - Tabela de parâmetros de D-H.

Junta	$\boldsymbol{\theta}$	$\boldsymbol{d}$	$\boldsymbol{a}$	$\boldsymbol{\alpha}$
1	θ_1	d_1	a_1	α_1
2	θ_2	d_2	a_2	α_2
n	θ_n	d_n	a_n	α_n

Fonte: Adaptado de Niku (2010).

As matrizes de transformações homogêneas de D-H relacionam o sistema de referência da junta atual, $(i-1)$-ésima, com o sistema de referência da próxima junta, i-ésima. Esta relação se dá pela realização de uma sequência de quatro movimentos que tem a função de deixar estas referências no mesmo ponto, tais como:

i. Rotação em um ângulo de θ_i o eixo Z_{i-1}. Este movimento deixará os eixos X_{i-1} e X_i paralelos;

ii. Transladar em uma distância d_i ao logo do eixo Z_{i-1} para deixar os eixos X_{i-1} e X_i colineares;

iii. Transladar ao logo do eixo X_i em uma distância a_1 para deixar os dois sistemas de referências no mesmo ponto;

iv. Finalmente, é necessário girar o eixo Z_{i-1} em torno do eixo X_i de um ângulo de α_i para alinhar o eixo Z_{i-1} com o eixo Z_i.

Após estes quatro movimentos, representados na equação (15), os pontos das $(i-1)$-ésima e i-ésima referências serão os mesmos. O resultado destas sucessivas transformações, se dá pela multiplicação das matrizes que representam cada transformação, representadas pela equação (16).

$$A_{i-1,i} = Rot(Z_{i-1}, \theta_i) \cdot Trans(0,0,d_i) \cdot Trans(a_i,0,0) \cdot Rot(X_i, \alpha_i)$$

$$= \begin{bmatrix} cos\theta_i & -sen\theta_i & 0 & 0 \\ sen\theta_i & cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha_i & -sen\alpha_i & 0 \\ 0 & sen\alpha_i & cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (15)$$

$$A_{i-1,i} = \begin{bmatrix} cos\theta_i & -cos\alpha_i \cdot sen\theta_i & sen\alpha_i \cdot sen\theta_i & a_i \cdot cos\theta_i \\ sen\theta_i & cos\alpha_i \cdot cos\theta_i & -sen\alpha_i \cdot cos\theta_i & a_i \cdot sen\theta_i \\ 0 & sen\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (16)$$

Desta forma, pode-se obter a matriz de transformação de todo o robô pela associação de cada referencial de junta sucessiva, ou seja, da relação da base até a primeira junta, da primeira à segunda, e assim por diante até o referencial de sistema de coordenadas do terminal efetuador (NIKU, 2010).

— Cinemática direta pela Sistemática de D-H

Ao aplicar a sistemática de D-H no robô exemplificado na Figura 8, obtêm-se a tabela de parâmetros, Tabela 2. Observa-se que a distância da junta d_1 recebe o valor da parcela a_1 pois, é considerada como o deslocamento da junta J_1, ou seja, a distância entre o sistema de coordenadas da base até a intersecção do eixo z_0 com o eixo x_1 ao longo do eixo z_0.

Tabela 2 - Tabela de parâmetros de D-H do exemplo da Figura 8.

Junta	$\boldsymbol{\theta}$	$\boldsymbol{d}$	$\boldsymbol{a}$	$\boldsymbol{\alpha}$
1	θ_1	a_1	0	90°
2	θ_2	0	a_2	0
3	θ_3	0	a_3	0

Fonte: Elaborada pelo autor.

Com os parâmetros do exemplo da Figura 8 determina-se a matriz de transformação de D-H. Esta é obtida pela multiplicação das matrizes que relacionam o sistema de referência da base com o da junta J_1 ($A_{0,1}$), da junta J_1 com a junta J_2 ($A_{1,2}$) e da junta J_2 com a junta J_3 ($A_{2,3}$), dadas pelas equações (17), (18) e (19), respectivamente. A matriz de transformação de D-H é apresentada na equação (20).

Para simplificação, as seguintes notações e relações trigonométricas são utilizadas:

$$C_i = \cos\theta_i \text{ e } S_i = sen\theta_i$$

$$C_{ij} = \cos(\theta_i \pm \theta_j) = \cos\theta_i \cdot cos\theta_j \mp sen\theta_i \cdot sen\theta_j$$

$$S_{ij} = \text{sen}(\theta_i \pm \theta_j) = \text{sen}\,\theta_i \cdot cos\theta_j \pm cos\theta_i \cdot sen\theta_j$$

$$A_{0,1} = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

$$A_{1,2} = \begin{bmatrix} C_2 & -S_2 & 0 & a_2 \cdot C_2 \\ S_2 & C_2 & 0 & a_2 \cdot S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$

$$A_{2,3} = \begin{bmatrix} C_3 & -S_3 & 0 & a_3 \cdot C_3 \\ S_3 & C_3 & 0 & a_3 \cdot S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

$$A_{0,3} = A_{0,1} \cdot A_{1,2} \cdot A_{2,3}$$

$$A_{0,3} = \begin{bmatrix} C_1 \cdot C_{23} & -C_1 \cdot S_{23} & S_1 & a_3 \cdot C_1 \cdot C_{23} + a_2 \cdot C_{12} \\ S_1 \cdot C_{23} & -S_1 \cdot S_{23} & -C_1 & a_3 \cdot S_1 \cdot C_{23} + a_2 \cdot S_1 \cdot C_2 \\ S_{23} & C_{23} & 0 & a_3 \cdot S_{23} + a_2 \cdot S_2 + a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{20}$$

Por essa matriz é possível identificar os valores do ponto em coordenadas cartesianas da posição final do manipulador pelos elementos da quarta coluna, equações (21) a (23),

$$P_x = a_3 \cdot C_1 \cdot C_{23} + a_2 \cdot C_{12} \tag{21}$$

$$P_y = a_3 \cdot S_1 \cdot C_{23} + a_2 \cdot S_1 \cdot C_2 \tag{22}$$

$$P_z = a_3 \cdot S_{23} + a_2 \cdot S_2 + a_1 \tag{23}$$

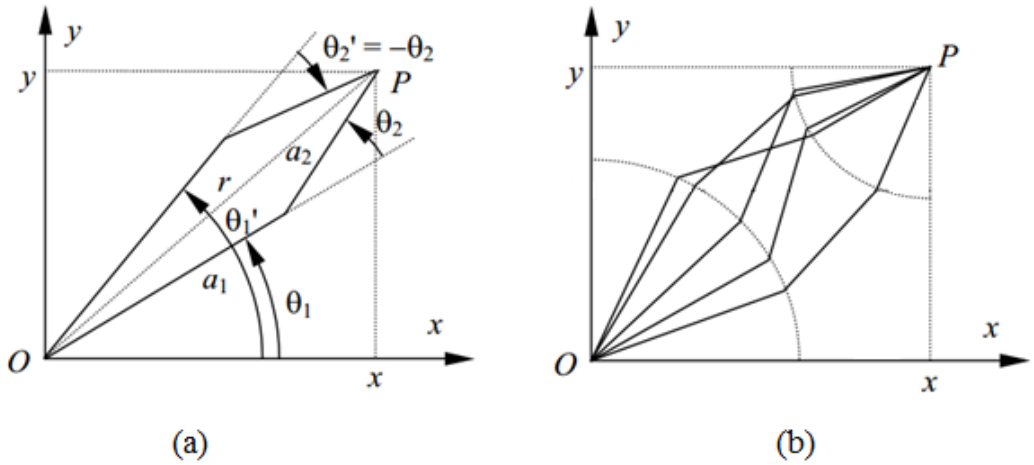
3.2.4 Análise Geométrica e Algébrica para a cinemática inversa

Diferente da cinemática direta que apresenta uma solução única, o problema da cinemática inversa fundamenta-se na não linearidade de suas funções e nas redundâncias que causam múltiplas soluções.

A cinemática inversa é utilizada quando se deseja determinar os ângulos das juntas de modo que satisfaça a posição final do terminal do manipulador no plano de coordenadas cartesianas, (x, y e z). Este modelamento matemático é frequentemente requerido para programação de robôs manipuladores, uma vez que é comum, por exemplo, que a ferramenta aborde uma peça qualquer, em determinada posição e orientação (CRAIG, 2012).

Sabendo-se a posição de origem de um manipulador, ou seja, o referencial de sua base representado por *O*, Figura 10(a) e (b), e a posição desejada no outro extremo do manipulador representada por *P*, existe múltiplas soluções que poderiam satisfazer a configuração dos ângulos das juntas. Portanto, existem redundâncias na solução da cinemática inversa, conforme os exemplos ilustrados na Figura 10(a) e (b), para manipuladores planares de dois e três GDL, respectivamente.

Figura 10 - Exemplos de redundâncias no plano x, y. (a) dois graus de liberdade e (b) três graus de liberdade.



Fonte: Adaptado de Carrara (2009).

Com base na Figura 10(a) percebe-se que o valor do ângulo θ_1 depende se o ângulo de θ_2 é positivo ou negativo, ou seja, se o manipulador está com o "cotovelo" para cima ou para baixo. No entanto, para o caso representado na Figura 10(b), o primeiro elo está com uma extremidade fixada na base e é possível posicionar a outra extremidade, em uma determinada faixa no espaço de trabalho que forma parte de um círculo, o mesmo ocorre com

o último elo, que deve estar posicionado na posição desejada. Logo, dependendo da sensibilidade do manipulador, podem existir infinitas soluções que satisfaçam o correto posicionamento do manipulador de forma que alcance determinada posição.

— **Cinemática inversa pelo método geométrico**

Para ilustrar a complexidade da análise da cinemática inversa, determina-se pela forma geométrica, os valores dos ângulos das juntas, θ_1, θ_2 e θ_3 para o manipulador robótico de 3 GDL, usando a configuração dada na Figura 8. Nesta, as relações trigonométricas definem as variáveis articulares com base nas variáveis cartesianas do ponto *P* e a orientação, α, em relação ao plano x, y.

O ângulo θ_1 pode ser calculado de forma direta pela Figura 8(b), equação (24), e a distância d é a hipotenusa do triângulo dado por x e y, equação (25),

$$\theta_1 = \arctan\frac{y}{x} \tag{24}$$

$$d = \sqrt{x^2 + y^2} \tag{25}$$

Visando calcular o ângulo θ_3 e utilizando a vista lateral, Figura 8(c), resolvendo o Teorema de Pitágoras no triângulo PQR e a Lei dos Cossenos no triângulo RSP, tem-se,

$$x^2 + y^2 + (z - a_1)^2 = {a_2}^2 + {a_3}^2 - 2 \cdot a_2 \cdot a_3 \cdot \cos(180 - \theta_3) \tag{26}$$

E sabendo que o $cos(180 - \theta_3) = -cos\,\theta_3$, é possível calcular θ_3 isolando-o em (26), tal que,

$$\theta_3 = \pm arcos\left(\frac{x^2 + y^2 + (z - a_1)^2 - {a_2}^2 - {a_3}^2}{2 \cdot a_2 \cdot a_3}\right) \tag{27}$$

Para o cálculo de θ_2 são necessários o ângulo de orientação α e o ângulo auxiliar β, Figura 8(c), de forma que, α e β são dados pelas equações (28) e (30), respectivamente,

$$\alpha = arctan\left(\frac{z - a_1}{\sqrt{x^2 + y^2}}\right) \tag{28}$$

$$\beta = arctan\left(\frac{a_3 \cdot sen(\theta_3)}{a_2 + a_3 \cdot cos(\theta_3)}\right) \tag{29}$$

Como θ_2 é dado pela relação $\theta_2 = \alpha - \beta$ e aplicando a fórmula da tangente da soma de arcos, obtém-se a equação (30),

$$\theta_2 = arctan\left(\frac{(z - a_1)\cdot(a_2 + a_3\cdot cos(\theta_3)) - \sqrt{x^2 + y^2}\cdot a_3\cdot sen(\theta_3)}{\sqrt{x^2 + y^2}\cdot(a_2 + a_3\cdot cos(\theta_3)) + (z - a_1)\cdot a_3\cdot sen(\theta_3)}\right) \quad (30)$$

Nota-se que o ângulo θ_3, equação (27), pode resultar em valores positivos e negativos, isto significa que o manipulador pode apresentar redundâncias no mesmo posicionamento com o "cotovelo" para cima ou para baixo.

— **Cinemática inversa pelo método algébrico**

A solução da cinemática inversa pela forma algébrica é obtida, basicamente, por manipulações algébricas e utilizando-se o mesmo exemplo da Figura 8, iguala-se os elementos da matriz do manipulador, equação (20), à uma matriz homogênea de valores de posição e orientação desejados, obtendo a equação (31),

$$A_{0,3} = A_{desejado}$$

$$\begin{bmatrix} C_1\cdot C_{23} & -C_1\cdot S_{23} & S_1 & a_3\cdot C_1\cdot C_{23} + a_2\cdot C_{12} \\ S_1\cdot C_{23} & -S_1\cdot S_{23} & -C_1 & a_3\cdot S_1\cdot C_{23} + a_2\cdot S_1\cdot C_2 \\ S_{23} & C_{23} & 0 & a_3\cdot S_{23} + a_2\cdot S_2 + a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (31)$$

Comparando as duas matrizes, observa-se que os elementos, (3,1), (3,2) e (3,4), são igualados a,

$$n_z = S_{23} \quad (32)$$

$$o_z = C_{23} \quad (33)$$

$$p_z = a_3\cdot S_{23} + a_2\cdot S_2 + a_1 \quad (34)$$

Manipulando algebricamente a equação (34), define-se θ_2 pela equação (35),

$$\theta_2 = arcsen\left(\frac{p_z - a_1 - a_3\cdot n_z}{a_2}\right) \quad (35)$$

Conhecendo θ_2, pode se obter θ_3 por,

$$\theta_3 = arctan\left(\frac{n_z}{o_z}\right) - \theta_2 \tag{36}$$

3.3 COMENTÁRIOS

Neste capítulo foram abordados os aspectos construtivos e a cinemática de manipuladores, visando entender a análise cinemática do manipulador implementado com juntas rotacionais e 5 GDL, e o uso da sua configuração para a solução da cinemática inversa por RNAs.

Pela modelagem matemática da cinemática apresentada, percebe-se que a análise da cinemática inversa através de métodos geométricos ou algébricos é complexa quando comparada às análises relacionadas à cinemática direta, por isso, a busca de métodos alternativos.

CAPÍTULO 4. REDES NEURAIS ARTIFICIAIS

Uma breve abordagem sobre às Redes Neurais Artificiais é apresentada neste tópico, com ênfase à Rede *Perceptron* Multicamadas, pois trata-se de uma categoria de RNA muito utilizada para aplicações que envolvem robôs manipuladores. Inicia-se pelo histórico, depois mostra-se o neurônio artificial, sua definição e suas principais características, parâmetros, algoritmo de treinamento e etc.

4.1 HISTÓRICO

Historicamente, o primeiro trabalho envolvendo um modelo matemático inspirado em um neurônio biológico foi desenvolvido por McCulloch e Pitts em 1943 (HAYKIN, 2001). Posteriormente, em 1949, Hebb propôs a primeira regra de treinamento de um neurônio artificial. Finalmente, em 1958, Rosenblatt idealizou o modelo básico do neurônio denominado de *Perceptron*.

A Rede *Perceptron* idealizada por Rosenblatt apresentava sinapses ajustáveis que podiam ser treinadas para reconhecimento de certos tipos de padrões. Descreveu uma topologia (estruturas de ligações entre os neurônios), propondo um algoritmo para treinar a Rede para executar determinados tipos de funções (DA SILVA; SPATTI; FLAUZINO, 2010).

Apesar do grande interesse inicial pelo estudo das RNAs, na década de 70 Minsky e Papert em 1969, mostraram que as Redes Perceptron não eram capazes de realizar tarefas consideradas simples, como a detecção de paridade, conectividade e simetria, problemas considerados linearmente separáveis, superados mais tarde.

Em 1982, Hopfield publicou um artigo que chamou a atenção para as propriedades associativas das RNAs. Hopfield mostrou a relação entre RNAs recorrentes auto-associativas e sistemas físicos. Anos mais tarde pesquisadores desenvolveram o algoritmo de treinamento *backpropagation*. O que também ajudou no ressurgimento do interesse pelas Redes Neurais foi o desenvolvimento de computadores mais potentes.

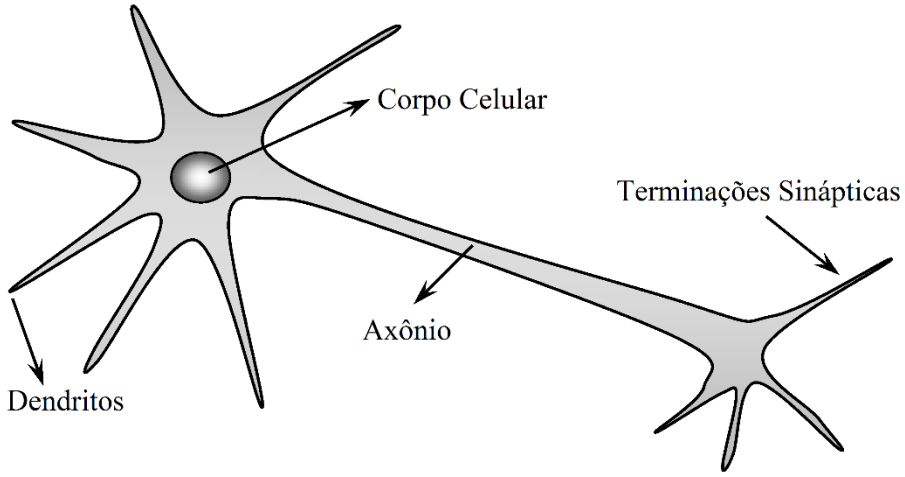
Atualmente, as Redes Neurais são uma importante metodologia aplicada em diversas áreas, cujo interesse dos pesquisadores está voltado ao desenvolvimento de técnicas de

aprendizagem com rápida convergência e a implementação de Redes Neurais em hardware (FERNANDES JUNIOR, 2014).

4.2 O NEURÔNIO ARTIFICIAL

Um neurônio artificial é uma estrutura básica para o processo de formação de uma Rede Neural, cuja inspiração é um neurônio biológico. O modelo biológico apresentado na Figura 11, é constituído pelo corpo celular que é responsável pela recepção e emissão dos sinais; o axônio possui a função de propagação dos impulsos nervosos; os dendritos são as ramificações do corpo celular das quais recebem impulsos nervosos e as sinapses são os pontos de contatos entre o axônio de um neurônio para dendritos de outros neurônios. Nas sinapses ocorrem as ponderações das transmissões dos sinais de ação (DA SILVA; SPATTI; FLAUZINO, 2010; HAYKIN, 2001).

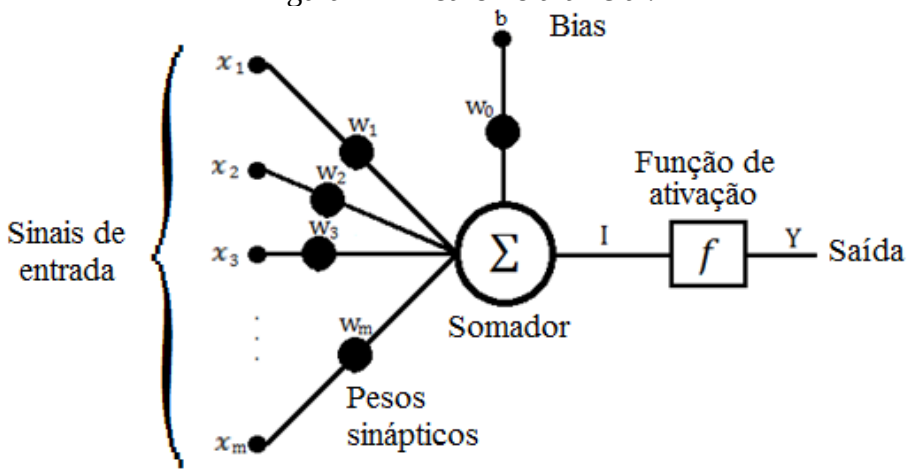
Figura 11 - Ilustração de um neurônio biológico.



Fonte: Adaptado de Da Silva, Spatti e Flauzino (2010)

De forma análoga ao modelo biológico, o neurônio artificial, ilustrado na Figura 12, é composto basicamente pelos mesmos elementos: as entradas, o conjunto de pesos sinápticos, um somador e uma função de ativação e sua saída. Tal como um neurônio biológico, o modelo artificial pode conter várias entradas, entretanto, somente uma saída. No entanto, sua saída pode ser propagada para um ou mais neurônios (HAYKIN, 2001).

Figura 12 - Neurônio artificial.



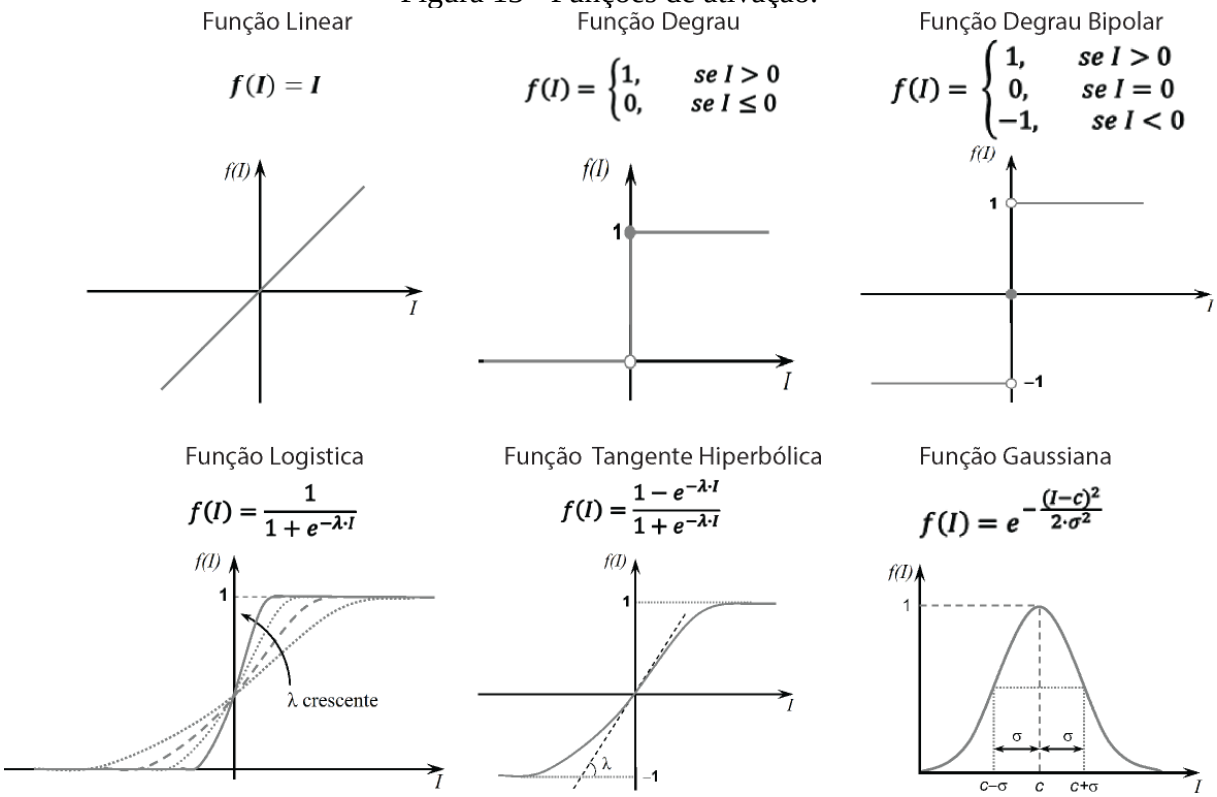
Fonte: Elaborada pelo autor.

Descreve-se a seguir, cada elemento de um neurônio artificial, mostrado na figura.

- Os sinais de entrada são provindos do meio externo de uma aplicação específica e usualmente são normalizados;
- Os pesos sinápticos ponderam cada variável de entrada quantificando sua relevância à funcionalidade do neurônio;
- O somador agrega todos os sinais de entrada ponderados, o Bias, produzindo um valor de saída intermediária, ou potencial de ativação (I);
- O Bias (b) é uma variável que delimita o patamar do resultado do somador;
- A Função de ativação processa e limita a saída do neurônio dentre um intervalo de valores específicos;
- A saída (Y) consiste no valor final produzido pelo neurônio em relação a um determinado conjunto de sinais de entrada.

As funções de ativação mais utilizadas são ilustradas pela Figura 13 juntamente com suas expressões matemáticas.

Figura 13 - Funções de ativação.



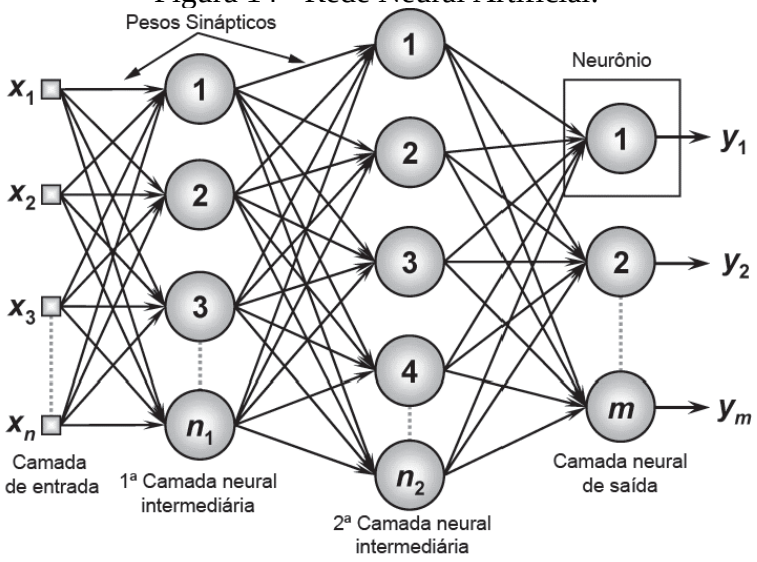
Fonte: Adaptado de Da Silva; Spatti; Flauzino (2010)

Estão divididas em dois grupos, funções parcialmente diferenciáveis e as funções totalmente diferenciáveis. Das funções apresentadas, apenas as funções do tipo degrau e a degrau bipolar são parcialmente diferenciáveis, esta classificação será melhor explorada no decorrer deste capítulo.

4.3 CARACTERÍSTICAS GERAIS DE REDES NEURAIS ARTIFICIAIS

O cérebro humano é um conjunto de neurônios biológicos organizados de forma paralela e, em geral, sua função constitui-se no processamento de informações. Por se tratar de uma analogia, as Redes Neurais Artificiais são compostas por neurônios artificiais também organizados de forma paralela e possuem a mesma finalidade do modelo biológico. Portanto, as RNAs são algoritmos compostos por conjuntos de regras e operações matemáticas bem definidas (DA SILVA; SPATTI; FLAUZINO, 2010). Na Figura 14 ilustra-se um exemplo de Rede Neural Artificial.

Figura 14 - Rede Neural Artificial.



Fonte: Adaptado de Da Silva, Spatti e Flauzino (2010)

Nesta figura, tipicamente para as RNAs, observa-se que:

- São compostas por um aglomerado de neurônios artificiais distribuídos em camadas;
- Podem conter uma ou mais camadas intermediárias – ou camadas escondidas;
- Os neurônios de uma camada específica são ligados aos neurônios das camadas vizinhas, por intermédio de um conjunto de interconexões definidas como pesos sinápticos, ou simplesmente, pesos;
- Podem conter um ou mais neurônios na camada de entrada ou um ou mais neurônios na camada de saída.

Não existe nenhuma relação de consenso geral entre o número de neurônios das camadas de entradas e saídas. Esses números são específicos e estão relacionados com cada necessidade de aplicação da RNA.

As RNAs podem ser utilizadas para diversos propósitos tais como, análise de séries temporais, modelagem e reconhecimento de padrões, controle de processos, área médica, área aeroespacial, robótica, entre outros. Esta quantidade de aplicações deriva-se da existência de algumas propriedades e capacidade que elas possuem como, habilidade de aprendizado e generalização de padrões (ainda que o dados estejam incompletos), ser linear ou não-linear, capacidade de mapeamento de entrada e saída, adaptabilidade, tolerância à falha, etc. (HAYKIN, 2001).

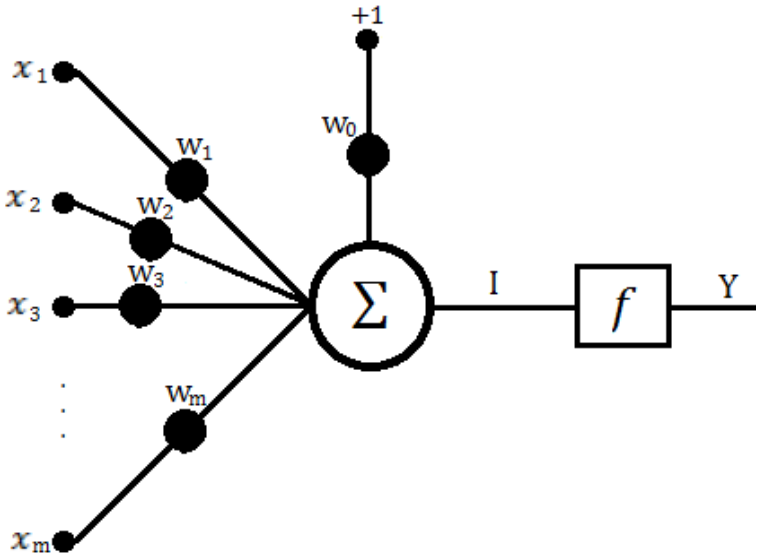
Uma RNA exige o processo de aprendizagem ou treinamento - nesta etapa, a RNA utiliza dados originados do ambiente para “aprender” relações entre padrões de entrada e saída. Nesse processo é realizado o ajuste nos valores dos pesos sinápticos com a finalidade de aumentar o desempenho da Rede Neural.

A aprendizagem pode ser realizada de forma supervisionada ou não supervisionada. Na aprendizagem supervisionada o ajuste dos pesos sinápticos é executado a partir dos padrões de entrada e suas relativas respostas desejadas, enquanto que, na não supervisionada a Rede Neural é capaz de aprender apenas com os valores dos padrões de entrada. Redes Neurais que utilizam o treinamento não supervisionado são caracterizadas como Redes auto-organizáveis (DA SILVA; SPATTI; FLAUZINO, 2010; HAYKIN, 2001).

4.4 O PERCEPTRON

O modelo *Perceptron* proposto por Rosenblatt foi o primeiro neurônio com um treinamento realizado de forma supervisionada. O *Perceptron* é constituído por um único neurônio que é capaz apenas de reconhecer padrões entre duas classes linearmente separáveis. Na Figura 15, apresenta-se o modelo de um *Perceptron* que é composto de um único neurônio com suas entradas e sinal de polarização (ou *bias*) e seus respectivos pesos sinápticos, um somador, sua função de ativação e uma saída.

Figura 15 - *Perceptron*.



Fonte: Elaborada pelo autor

O sinal de polarização juntamente com seu respectivo peso sináptico especifica um valor apropriado de disparo da função de ativação, pode ser unitário fixo, positivo ou negativo. O valor da saída intermediária *I* é representado pela soma ponderada das entradas do neurônio, equação (37),

$$I = \sum_{i=1}^{m} w_i x_i + w_0 \tag{37}$$

onde:

- $w_i x_i$ é a ponderação de um vetor de entrada, x com m elementos da camada (neurônios) pelos seus respectivos pesos sinápticos, w;
- w_o é a multiplicação do sinal de polarização com seu respectivo peso sináptico.

A saída do *Perceptron* é dada por Y que é obtida pela aplicação da função de ativação, em relação à saída intermediária, equação (38),

$$Y = f(I) \tag{38}$$

Para que a RNA seja capaz de resolver um sistema não linear, a função de ativação $f(.)$ do *Perceptron* também deve ser não linear (totalmente diferenciáveis).

4.5 REDE *PERCEPTRON* MULTICAMADAS

A Rede *Perceptron* Multicamadas (PMC), também conhecida como Rede *Multilayer Feedforward Network* ou *Multilayer Perceptron* (MLP), recebe esse nome em razão da associação dos *Perceptrons* em camadas, sendo composta de pelo menos, uma camada intermediária. Essa associação em camadas possibilita a Rede Neural resolver problemas mais complexos como classificações, reconhecimento de padrões, aproximações de funções, previsões de séries temporais, controles de processos, robótica, entre outras (DA SILVA; SPATTI; FLAUZINO, 2010; HAYKIN, 2001).

O número de neurônios de cada camada intermediária, assim como, o número de camadas da rede MLP diversifica-se de acordo com a necessidade de cada aplicação. Entretanto, de acordo com Minussi e Lotufo (2008), para não ocorrer estrangulamento de informações, o número de neurônios nas camadas intermediárias deve ser maior que os números de neurônios das camadas de entrada e saída.

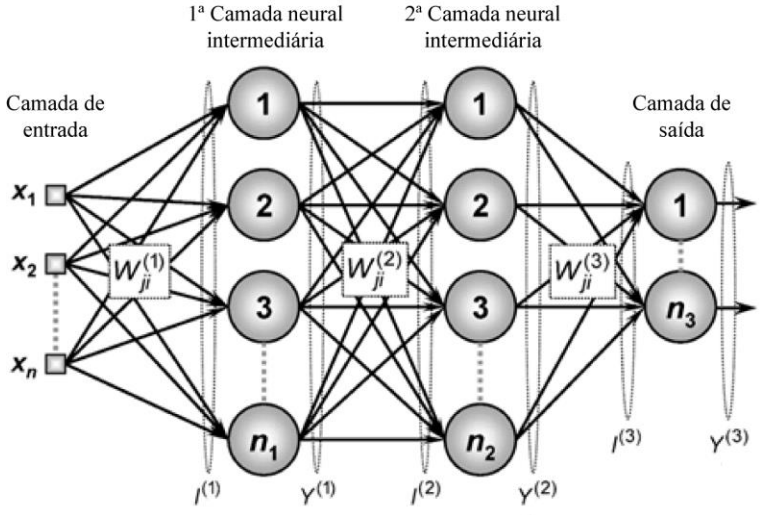
De forma complementar, Camargo, Veraszto e Barreto (2014) definem que uma Rede Neural com mais camadas, ou seja, mais neurônios nas camadas intermediárias podem apresentar um melhor processamento das informações, porém, se este número for muito elevado, pode existir lentidão desnecessária no treinamento da RNA.

De acordo com Feng, Yao-Nan e Yi-Min (2012), a MLP juntamente com a Rede de Função de Base Radial - ou do seu termo em inglês *Radial Basis Function* (RBF) - são as categorias de RNAs mais populares para a solução de problemas envolvendo a cinemática inversa de robôs manipuladores, em virtude de serem essencialmente simples e apresentarem bons resultados, utilizadas nos trabalhos de Jha, Biswal e Sahu (2014), Morris e Mansor (1997) e Camargo, Veraszto e Barreto (2014).

Outro fato que contribui para a popularidade da MLP é seu algoritmo de aprendizagem, ou treinamento, conhecido como algoritmo de Retropropagação do Erro ou *Error Backpropagation Algorithm* (EBP). De fato, é uma ferramenta relativamente simples de implementar que fornece significativos resultados.

Como em uma MLP as saídas dos neurônios de uma camada específica estão interligadas pelas entradas da próxima camada, o sinal de entrada é ramificado para a primeira camada intermediária, posteriormente para a próxima camada, seguindo desta forma até a última camada. Ilustra-se na Figura 16, uma rede MLP com duas camadas intermediárias com n entradas, n_1 neurônios na primeira camada neural intermediária, n_2 neurônios na segunda camada e n_3 neurônios na camada de saída.

Figura 16 - MLP com 2 camadas intermediárias.



Fonte: Adaptado de Da Silva, Spatti e Flauzino (2010)

Em relação ao processo de propagação do sinal de entrada e com base na Figura 16, tem-se que:

- O conjunto de pesos sinápticos que interligam uma entrada *i* com um neurônio *j* na camada *L* é representado como $W_{ji}^{(L)}$, uma vez que, a saída do neurônio em uma camada específica será as entradas ponderadas dos neurônios da camada seguinte;
- Do mesmo modo que ocorre no *Perceptron* de camada simples, as entradas dos neurônios localizados na camada *L* terão seus valores ponderados por um conjunto de pesos sinápticos;
- Para cada neurônio *j*, a somatória de todas as entradas ponderadas gera uma saída intermediária, que pode ser obtida pela equação (39).

$$I_j^{(L)} = \sum_{i=1}^{m} w_{ji} x_i + w_0 \tag{39}$$

Por meio do vetor $I^{(L)}$ representa-se as saídas intermediárias de todos os neurônios da camada L, Figura 16.

A saída intermediária $I_j^{(L)}$ do neurônio *j* na camada *L* é então utilizada pela função de ativação para gerar sua saída $Y_j^{(L)}$, de acordo com a equação (40).

$$Y_j^{(L)} = f(I_j^{(L)}) \tag{40}$$

As funções de ativação comumente utilizadas em Redes MLP são a função logística ou a tangente hiperbólica (Figura 13), esta última é expressa pela equação (41).

$$f(I) = \frac{1 - e^{-\lambda \cdot I}}{1 + e^{-\lambda \cdot I}} \tag{41}$$

Esta expressão possui a característica de apresentar valores entre -1 e 1, cuja constante λ está relacionada com a inclinação da função tangente hiperbólica, de modo que, se seu valor for muito elevado, a função de ativação irá se aproximar de uma função degrau.

A função de ativação deve ser não linear e contínua para ser capaz de mapear funções não lineares e para que o processo de cálculo do gradiente local, utilizado para o ajuste dos pesos, seja factível pois, neste cálculo é utilizada a diferenciação (DA SILVA; SPATTI; FLAUZINO, 2010; MINUSSI; LOTUFO, 2008).

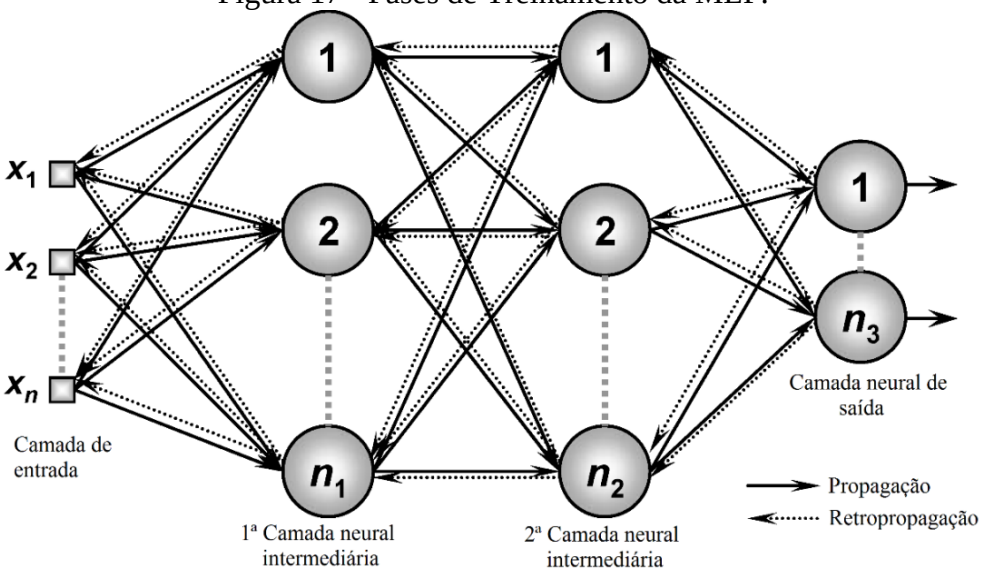
Uma Rede Neural deve ser configurada para que a aplicação de um conjunto de entradas produza um conjunto de saídas desejadas. Existem vários métodos para configurar o peso das conexões. Uma maneira é configurar os pesos explicitamente, usando conhecimentos a priori. Outra maneira é treinar e ensinar padrões à Rede Neural deixando que os pesos mudem de acordo com uma regra de treinamento (ou aprendizagem) que pode ser supervisionada ou não supervisionada.

No tipo de aprendizagem supervisionada a Rede Neural é treinada utilizando-se entradas e saídas de padrões desejados, onde os pares de entrada e saída podem ser fornecidos por uma fonte externa, ou pelo sistema que contém a Rede Neural. No caso da aprendizagem não supervisionada a saída é o resultado dos neurônios treinados que responde a um padrão de *clusters* existente dentro das entradas.

4.5.1 O Treinamento da Rede *Perceptron* Multicamadas

O algoritmo de treinamento da rede MLP é realizado de forma supervisionada constituindo-se em duas etapas: a propagação e a retropropagação, proposto por Werbos em 1974, e representadas na Figura 17 (HAYKIN, 2001).

Figura 17 - Fases de Treinamento da MLP.



Fonte: Adaptado de Da Silva, Spatti e Flauzino (2010).

Na etapa de propagação tendo um vetor de entrada, obtém-se os respectivos valores de saída, utilizando as equações (39) e (40) anteriormente citadas. Com exceção da camada de

entrada, que não possui neurônios (nem função de ativação), os valores de entrada são ramificados diretamente para a próxima camada.

O processo de propagação é realizado de forma que o sinal de entrada se ramifica camada por camada, até a camada neural de saída mantendo-se inalterados os valores dos pesos sinápticos. Após o processo de propagação são gerados os erros, enquanto que, na etapa de retropropagação são realizados os ajustes dos pesos.

Após a fase de propagação de uma determinada amostra de treinamento, o valor obtido para o neurônio j (equação (40)) na camada de saída, é comparado com seu valor desejado, d_j. A diferença entre o valor desejado e o valor obtido, o_j, gera um erro e_j, dado pela equação (42),

$$e_j = d_j - o_j \tag{42}$$

A partir deste resultado, inicia-se a etapa de retropropagação do erro, a qual irá ajustar os valores dos pesos sinápticos entre a camada de saída e a última camada intermediária.

O ajuste dos pesos pelo algoritmo de BP deriva-se da regra Delta, um aprendizado baseado no gradiente descedente, representada na equação (43),

$$w_{ji}^{(L)}(t+1) = w_{ji}^{(L)}(t) + \eta \cdot \delta_j^{(L)} \cdot Y_i^{(L-1)} \tag{43}$$

Sendo:

- t: o valor da interação atual, logo, $(t+1)$ é o valor da interação seguinte;
- η: taxa de treinamento da Rede Neural;
- i: índice do neurônio da camada anterior ao peso sináptico, ou índice da entrada;
- j: índice do neurônio da camada do peso sináptico;
- L: índice da camada do peso sináptico;
- $\delta_j^{(k)}$: gradiente local;
- $Y_i^{(L-1)}$: saída do neurônio da camada anterior.

A taxa de treinamento da Rede Neural está relacionada à velocidade de convergência do treinamento, entretanto, se seu valor for muito elevado, o treinamento pode se tornar instável. Para que não ocorra instabilidade na convergência do algoritmo é recomendável que a taxa de treinamento, η, esteja dentro do intervalo $0 < \eta < 1$ (MINUSSI; LOTUFO, 2008).

O valor do gradiente local é obtido por intermédio da equação (44), se L for o índice que representa a última camada, caso contrário, este seu valor é obtido pela equação (45),

$$\delta_j^{(L)} = e_j \cdot f'(I_j^{(L)}) \tag{44}$$

$$\delta_j^{(L)} = \left(\sum_{k=1}^{m} \delta_k^{(L+1)} \cdot w_{kj}^{(L+1)} \right) \cdot f'(I_j^{(L)}) \tag{45}$$

Sendo:

m: é o número de neurônios que compõe a camada $L + 1$ (próxima camada).

Observa-se pelas equações (44) e (45) a necessidade da função de ativação ser contínua devido a diferenciação para o cálculo do gradiente local no processo de treinamento via BP.

Adicionalmente, é frequente o uso do termo momento, ou momentum, α, no ajuste de pesos sinápticos no algoritmo de BP. Esse parâmetro melhora a convergência do treinamento quando a variação de pesos entre duas interações tem valor considerável pois, está relacionado com o valor do peso sináptico da interação anterior. Desta forma, o ajuste de pesos é representado pela equação (46),

$$w_{\mathrm{ji}}^{(\mathrm{L})}(\mathrm{t}+1) = \mathrm{w}_{\mathrm{ji}}^{(\mathrm{L})}(\mathrm{t}) + \alpha \cdot \left[w_{ji}^{(L)} - w_{ji}^{(L)}(t-1) \right] + \eta \cdot \delta_{\mathrm{j}}^{(\mathrm{L})} \cdot Y_i^{(L-1)} \tag{46}$$

Conclui-se então que, no processo de retropropagação são realizados os ajustes dos pesos sinápticos da última camada pelas equações (43) e (44); e da penúltima camada até a primeira por intermédio das equações (43) e (45). Caso, no ajuste de pesos sinápticos seja considerado o fator momento, utiliza-se a equação (46) em substituição a equação (43).

Os ajustes dos pesos sinápticos são feitos a cada padrão de treinamento disponível, adicionalmente, para um k-*ésimo* padrão de treinamento apresentado à Rede Neural, gera-se um erro de aproximação instantâneo (ou local), proveniente da equação (42), representado pela equação (47),

$$\varepsilon(k) = \frac{1}{2} \cdot \sum_{i=1}^{\omega} e_i^2(k) \tag{47}$$

Sendo ω o número de neurônios da camada de saída da rede.

Para obter o erro quadrático médio ou erro global da RNA na fase treinamento utiliza-se a equação (48),

$$\varepsilon_{méd} = \frac{1}{N} \cdot \sum_{k=1}^{N} \varepsilon(k) \tag{48}$$

Sendo *N* o número total de padrões apresentados à Rede Neural.

Um critério de parada do treinamento via algoritmo de BP é utilizado, sendo os mais comuns:

1. Quando o erro quadrático médio atinge um valor mínimo especificado;
2. Ou, quando a variação do erro quadrático médio de uma época para outra, se torna suficientemente pequena;
3. Ou ainda, quando o treinamento atinge um número determinado de épocas.

Entende-se como época a contagem de vezes que todos os padrões foram apresentados à Rede Neural no processo de treinamento.

O problema para o primeiro critério de convergência é que, se for especificado um valor muito baixo de erro, o tempo de aprendizagem pode ser muito longo. Todavia, no critério de parada do segundo e terceiro casos, pode resultar em um treinamento prematuro (HAYKIN, 2001).

4.6 COMENTÁRIOS

Neste capítulo foi apresentada a teoria básica sobre Redes Neurais com ênfase na Rede *Perceptron* Multicamadas. Esta Rede Neural é o tipo mais recomendo para resolver o problema da cinemática inversa, uma vez que se trata de um problema não linear.

CAPÍTULO 5. GERAÇÃO DE PADRÕES DE TREINAMENTO, ALGORITMOS DAS RNAS E RESULTADOS NA FASE OFFLINE

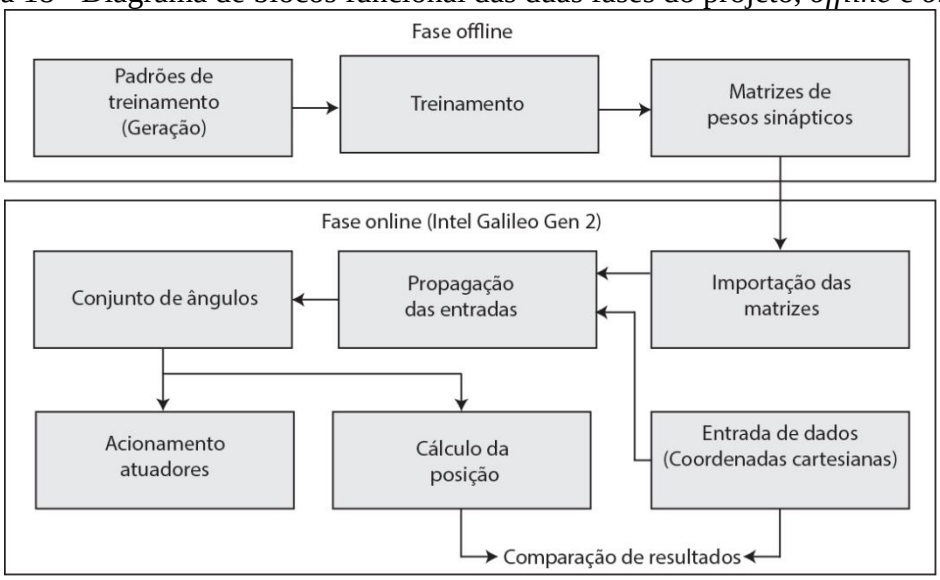
Neste capítulo apresenta-se a geração de padrões de treinamento, os algoritmos da RNAs, as simulações e seus resultados realizados na fase *offline*, ou seja, antes da implementação do algoritmo da RNA no protótipo do manipulador.

Inicia-se pelo diagrama de blocos que apresenta todas as etapas do projeto, incluindo as etapas da fase *online*.

5.1 DIAGRAMA DE BLOCOS DO DESENVOLVIMENTO DO PROJETO

O algoritmo da RNA desenvolvido foi aplicado conforme a sequência das etapas necessárias de projeto apresentado no diagrama de blocos da Figura 18, visando atingir os objetivos iniciais do trabalho, ou seja, o acionamento do manipulador robótico por cinemática inversa em um volume de trabalho limitado.

Figura 18 - Diagrama de blocos funcional das duas fases do projeto, *offline* e *online*.



Fonte: Elaborada pelo autor.

Nesse diagrama de bloco observa-se que o desenvolvimento foi dividido em duas fases uma de geração de padrões de treinamento e o treinamento visando obter o banco de conhecimentos das matrizes de pesos sinápticos. Na segunda etapa, em modo *online*, realiza-se a aplicação no protótipo usando uma plataforma (ou placa) de desenvolvimento. A divisão

em fases, *online* e *offline*, é devido ao fato do treinamento da Rede Neural exigir recursos computacionais e tempo elevados, para o treinamento e proibitivos para o processamento *online*.

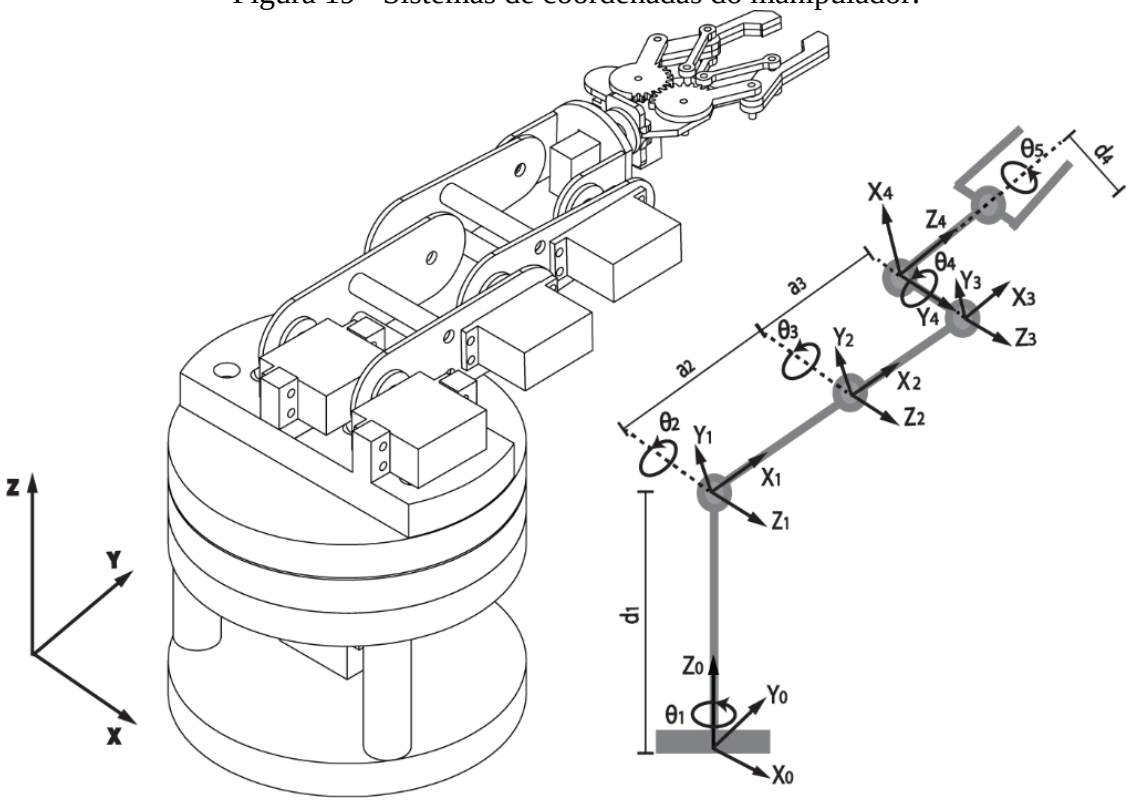
Todas as etapas na fase *offline* foram executadas em um *notebook* com um processador Intel Core I7-4510U operando em 2,6GHz, memória RAM de 8Gb usando o software MATLAB.

De forma a dar os subsídios para o entendimento do algoritmo desenvolvido, descreve-se a seguir, cada etapa da fase *offline* apresentada no diagrama de blocos.

5.2 CINEMÁTICA DIRETA DO PROTÓTIPO E DELIMITAÇÕES PARA A GERAÇÃO DE PADRÕES DE TREINAMENTOS

Para a geração dos padrões de treinamento utilizam-se a cinemática direta e a sistemática de D-H, para isso, é necessário ter previamente os dados dos parâmetros do protótipo do manipulador robótico, assim como definir se houver, as limitações de cada junta. Estes parâmetros são obtidos com base no sistema de referência de cada junta do protótipo, mostrado na Figura 19, e traduzidos na sistemática de D-H na Tabela 3.

Figura 19 - Sistemas de coordenadas do manipulador.



Fonte: Elaborada pelo autor.

Tabela 3 - Parâmetros de D-H do protótipo.

Junta	θ_i **[°]**	d_i **[mm]**	α_i**[°]**	a_i**[mm]**
1	θ_1	160	90	0
2	θ_2	0	0	80
3	θ_3	0	0	70
4	$(\theta_4 + 90)$	-41,75	90	0
5	--	156	0	0

Fonte: Elaborada pelo autor.

Na tabela observa-se que, apesar do protótipo possuir 5 GDL considera-se, para geração de padrões de treinamento (consequentemente à RNA) apenas 4 GDL, pois a proposta foi desenvolver um algoritmo capaz de posicionar o manipulador com uma tolerável precisão, em um sistema tridimensional de coordenadas cartesianas, sem contemplar a orientação.

Desta forma, o acionamento do giro do pulso não foi abordado para a geração dos padrões de treinamento (este grau de liberdade possui apenas a função de orientar o terminal efetuador). Ao eliminar um grau de liberdade o número de padrões de treinamento diminui exponencialmente quando se deseja mapear todo o espaço de trabalho do protótipo, devido a forma que estes pontos foram gerados no programa (sequência de laços *for*).

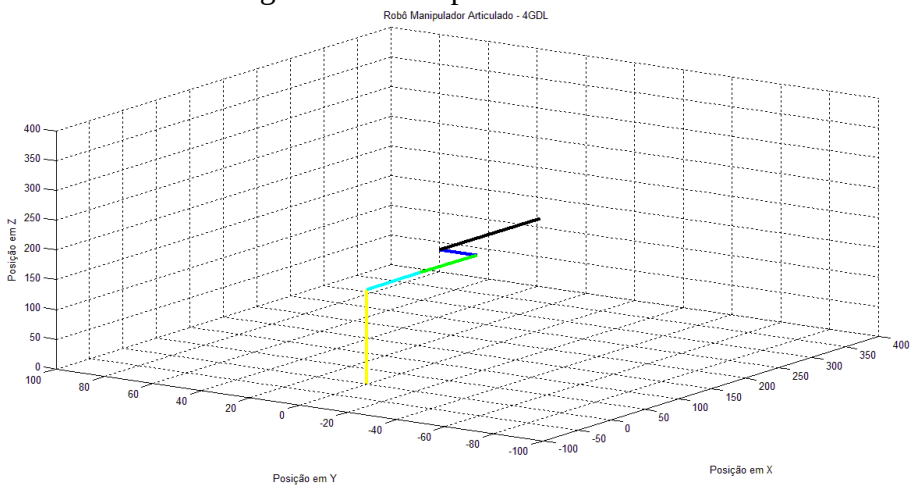
Embora as equações da cinemática direta aceitem qualquer valor de ângulo em uma junta rotacional de um manipulador, devido aos seus limites físicos e para diminuir as redundâncias, o seu volume de trabalho foi limitado aos valores de ângulos máximos e mínimos de cada junta, representados pelas expressões (49),

$$\begin{gathered} 0° \leq \theta_1 \leq 180° \\ 0° \leq \theta_2 \leq 90° \\ 0° \leq \theta_3 \leq 45° \\ -90° \leq \theta_4 \leq 0° \\ \theta_5 = 0° \end{gathered} \tag{49}$$

Esses limites foram considerados na geração do conjunto de treinamento da Rede Neural principalmente, para evitar cálculos desnecessários durante o treinamento.

Por intermédio dos parâmetros da tabela de D-H também é possível obter a simulação dos pontos geométricos do modelo do protótipo do manipulador no software MATLAB e sua representação visual, conforme ilustrado na Figura 20. Esta figura permite validar esses parâmetros e observar previamente o seu posicionamento.

Figura 20 - Manipulador simulado.



Fonte: Elaborada pelo autor.

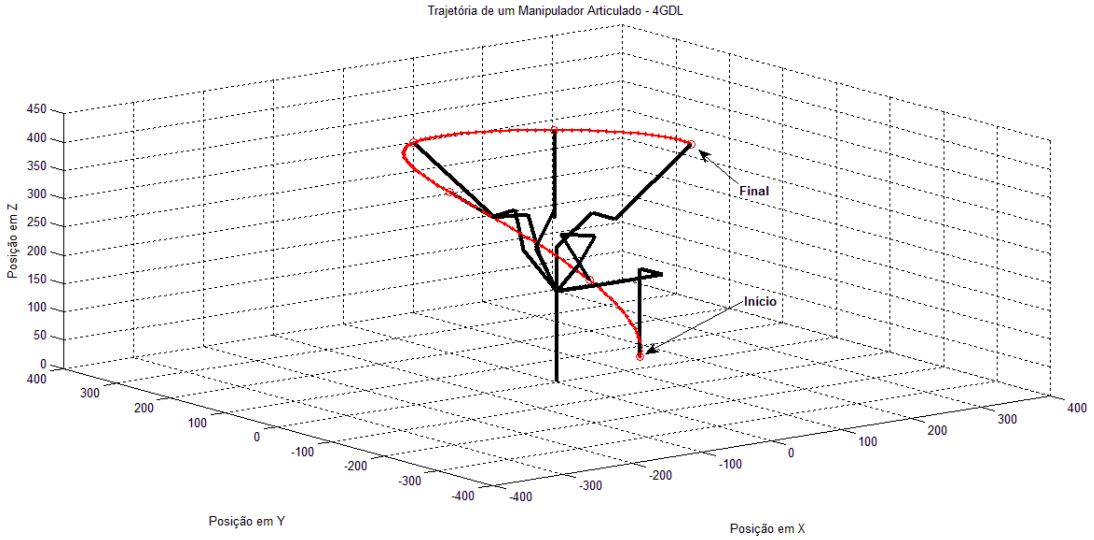
5.3 GERAÇÃO DE PADRÕES DE TREINAMENTOS

Após a obtenção dos parâmetros do manipulador, os padrões de treinamento são obtidos a partir de duas abordagens distintas, a geração de duas trajetórias e um conjunto de pontos distribuídos em um espaço de trabalho delimitado para o protótipo.

5.3.1 Geração das trajetórias

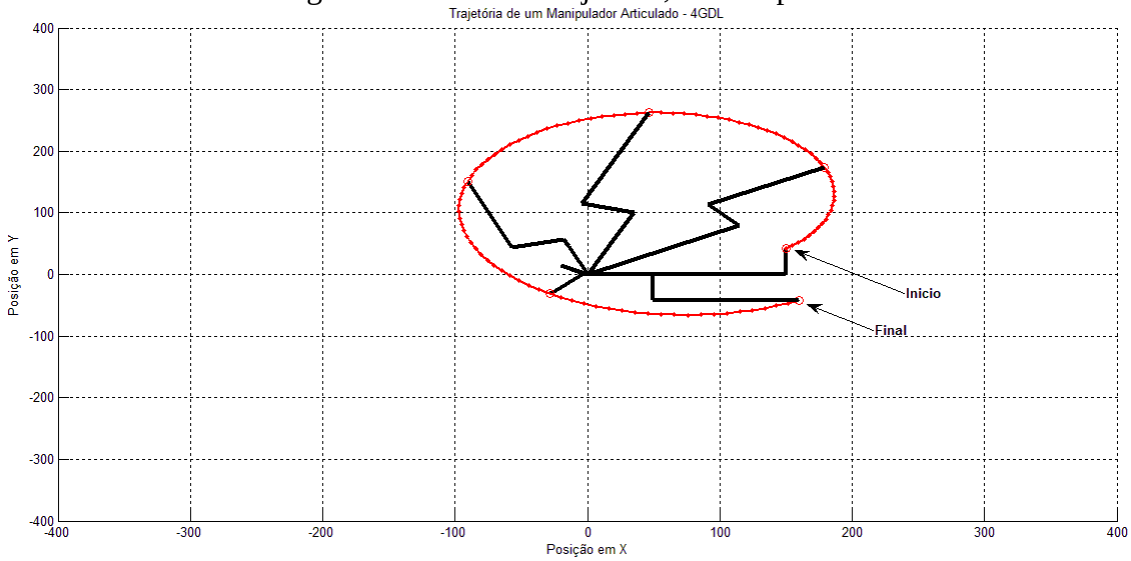
Na geração das duas trajetórias utiliza-se a cinemática direta, a primeira trajetória inicia-se com todos os valores de ângulos mínimos das juntas e finaliza-se com seus valores máximos, composta por um total de 100 pontos, dando como resultado uma pseudo-espiral no espaço tridimensional, conforme observado na Figura 21 e Figura 22, onde o conjunto de pontos gerados está representado na cor vermelha.

Figura 21 - Primeira trajetória, vista tridimensional.



Fonte: Elaborada pelo autor.

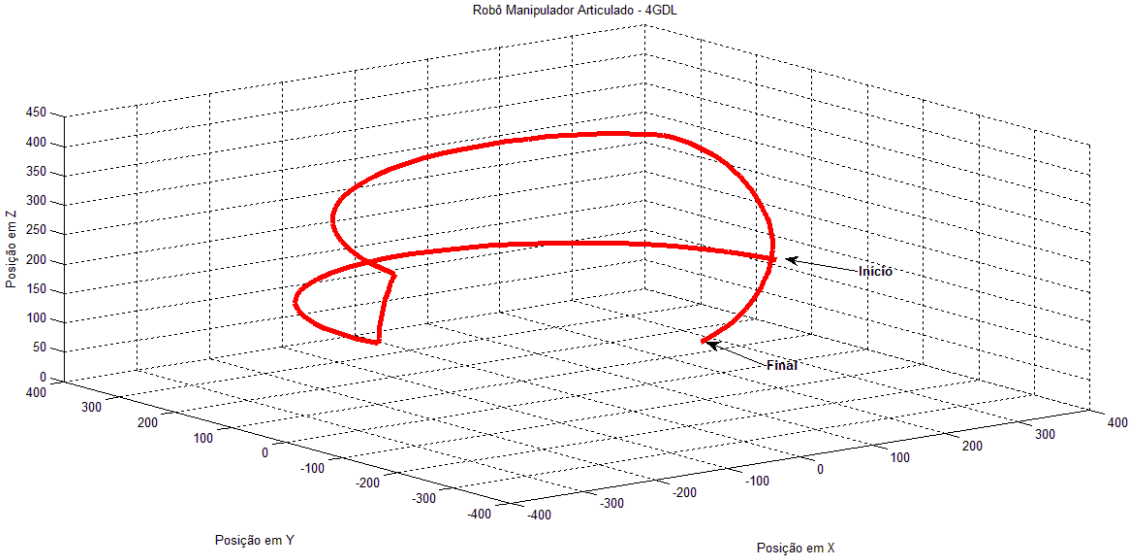
Figura 22 - Primeira trajetória, vista superior.



Fonte: Elaborada pelo autor.

A segunda trajetória, composta por 120 pontos, consiste de uma sequência de pontos que visa a simulação de uma trajetória mais próxima da realidade de trabalho do manipulador. Nesta, o manipulador robótico percorre três quadrantes do plano cartesiano, incrementando e posteriormente decrementando o ângulo da base (θ_2), com a variação dos outros ângulos de junta, de forma a apresentar o resultado mostrado na Figura 23.

Figura 23 - Segunda trajetória.



Fonte: Elaborada pelo autor.

5.3.2 Conjunto de pontos distribuídos no volume de trabalho delimitado

O conjunto de padrões de treinamento que abrange todo o volume de trabalho é encontrado para uma resolução definida pela faixa de trabalho versus número de pontos, dado na Tabela 4, utilizando uma sequência em série de quatros laços de repetições do tipo *for* na programação, sendo um laço para cada ângulo de junta considerada.

Tabela 4 - Resolução de pontos no espaço de trabalho.

Junta	Faixa de trabalho	Número de pontos	Resolução [graus/ponto]
1	180º	30	6
2	90º	18	5
3	45º	9	5
4	90º	6	15
5	--	1	--

Fonte: Elaborada pelo autor.

O total de pontos gerados nas coordenadas x, y e z relacionados às juntas e utilizados como padrões de treinamento são 29160 (dado pelo produto do número de pontos na tabela). Este número de padrões de treinamento depende do número de juntas e da resolução adotada para cada junta, e como consequência influencia no tamanho do volume de trabalho. Portanto,

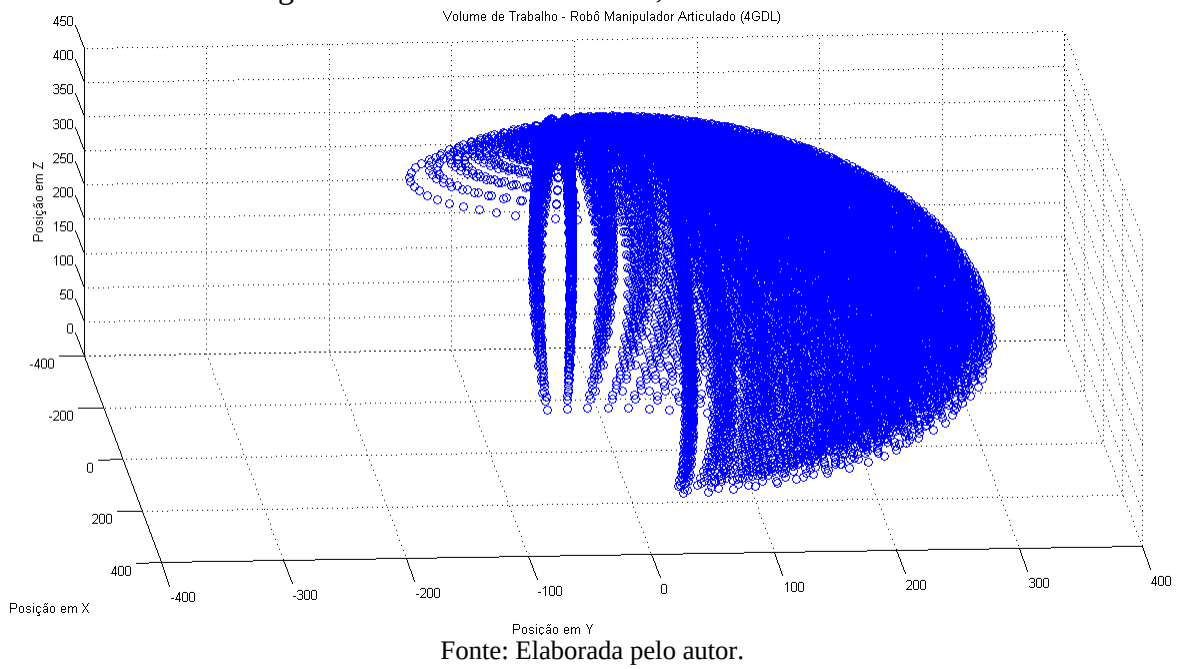
sem os limites físicos do manipulador e a delimitação do volume de trabalho, o conjunto de padrões de treinamento tende a aumentar de forma exponencial.

Para diminuir o número de padrões de treinamento e eliminar possíveis redundâncias implementa-se no programa uma lógica de verificação de redundâncias. O novo padrão gerado é armazenado, se forem diferentes de todos os outros já existentes no banco de dados, respeitando uma tolerância mínima considerada para cada eixo cartesiano. Por exemplo, um padrão $j = [X_j, Y_j, Z_j] = [100, 200{,}300]$ seria descartado se já existisse um padrão $n = [X_n, Y_n, Z_n] = [101, 198, 300]$ considerando a tolerância mínima de 2,1 mm.

Com a verificação de redundâncias foi eliminado um total de 21,82% dos padrões gerados inicialmente, 29160, considerando uma tolerância mínima de 2,5 mm.

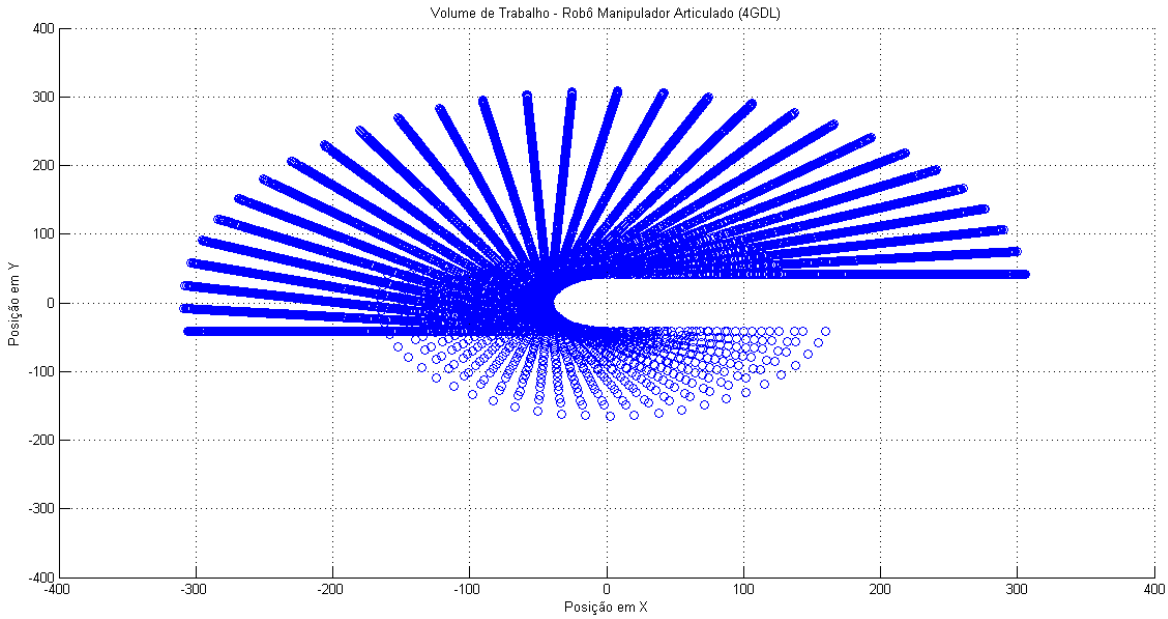
Encontrado o conjunto de pontos de padrões de treinamento pode-se observar de forma gráfica, o volume de trabalho delimitado para o protótipo do manipulador, ilustrado pela Figura 24 e Figura 25.

Figura 24 - Volume de trabalho, vista tridimensional.



Fonte: Elaborada pelo autor.

Figura 25 - Volume de trabalho, vista superior.



Fonte: Elaborada pelo autor.

5.4 ALGORITMOS DAS RNAS PARA O TREINAMENTO

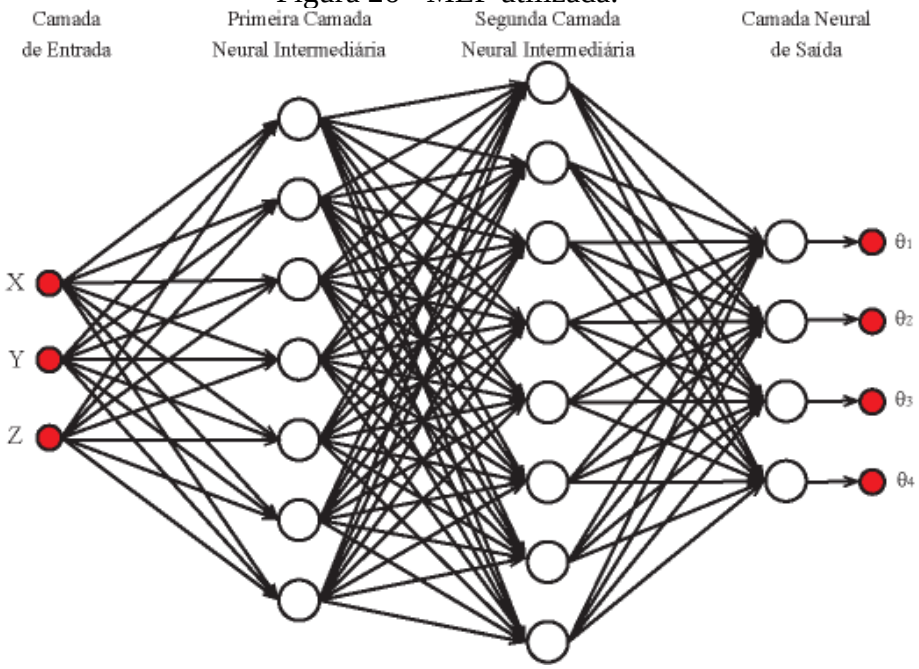
Tendo os valores de padrões de treinamento gerados pela cinemática direta, neste item apresentam-se os algoritmos da RNAs na fase *offline*, realizado para o treinamento e aprendizagem da relação cinemática do manipulador estudado.

Após o treinamento, o algoritmo deve ser capaz de fornecer um conjunto de valores de ângulos necessários para que o manipulador alcance - com uma tolerável precisão - qualquer ponto em coordenadas cartesianas, dentro do espaço de trabalho para o qual ele foi treinado.

Foram utilizadas duas técnicas de treinamento diferentes, em uma utiliza-se somente uma RNA e na outra são utilizadas nove RNAs, configuradas em paralelo.

Para ambos os casos, a RNA utilizada foi do tipo MLP (com treinamento via *backpropagation*), composta por 3 neurônios na camada de entrada - referentes às coordenadas em x, y e z, 7 e 8 neurônios na primeira e segunda camadas intermediárias, respectivamente, e 4 neurônios na camada de saída - referentes aos ângulos das juntas. A função de ativação empregada é do tipo tangente hiperbólica, a taxa de treinamento, η, igual a 0,75, e o fator momento, α, igual a 0,7. A arquitetura da RNA utilizada é ilustrada pela Figura 26.

Figura 26 - MLP utilizada.



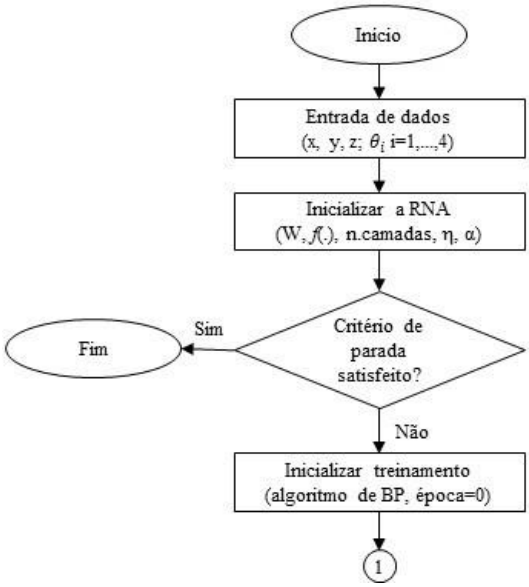
Fonte: Elaborada pelo autor.

Pela literatura estudada não foi encontrado um consenso geral sobre os valores dos parâmetros para otimização do rendimento das RNAs, portanto, neste trabalho os parâmetros citados foram definidos por diversas fontes, tais como, dicas em estudos e testes empíricos de otimização.

5.4.1 Algoritmo de treinamento considerando uma RNA

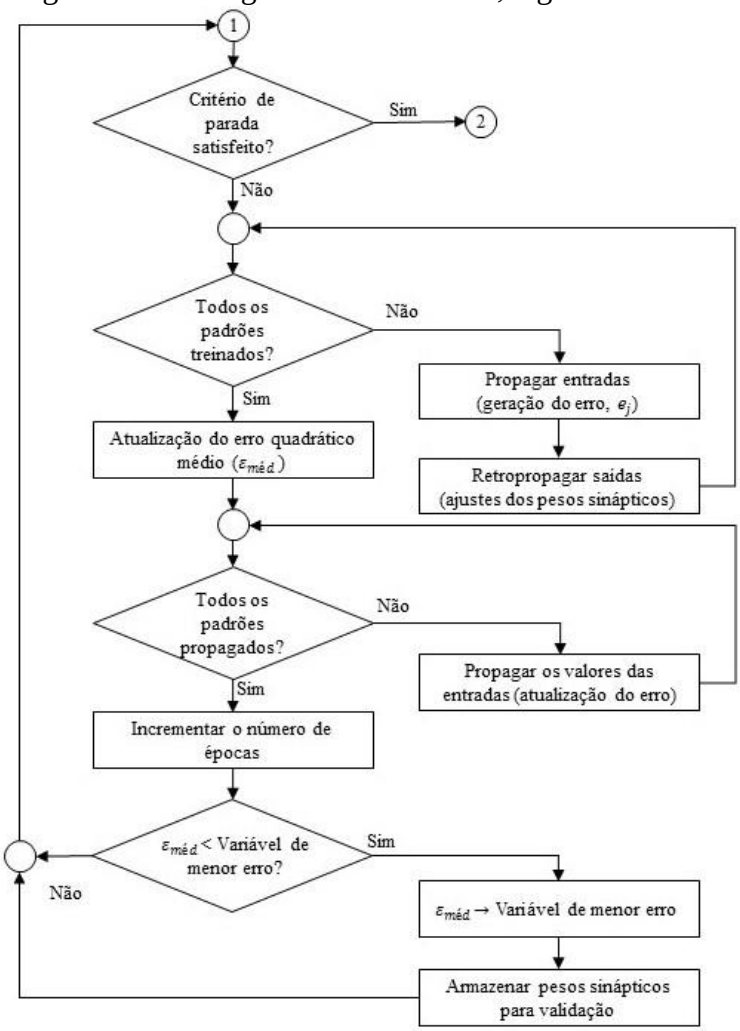
O algoritmo de treinamento completo considerando uma RNA encontra-se APÊNDICE A. Mostra-se a seguir, o mesmo algoritmo em fluxograma dividido em três partes: na Figura 27 tem-se o processo de inicialização, na Figura 28 o algoritmo de treinamento (BP) e na Figura 29 a validação dos resultados.

Figura 27 - Fluxograma de uma RNA, processo de inicialização.



Fonte: Elaborado pelo autor.

Figura 28 - Fluxograma de uma RNA, algoritmo de BP.



Fonte: Elaborado pelo autor.

Figura 29 - Fluxograma de uma RNA, validação dos resultados.

Fonte: Elaborado pelo autor.

5.4.2 Algoritmo de treinamento para RNAs configuradas em paralelo

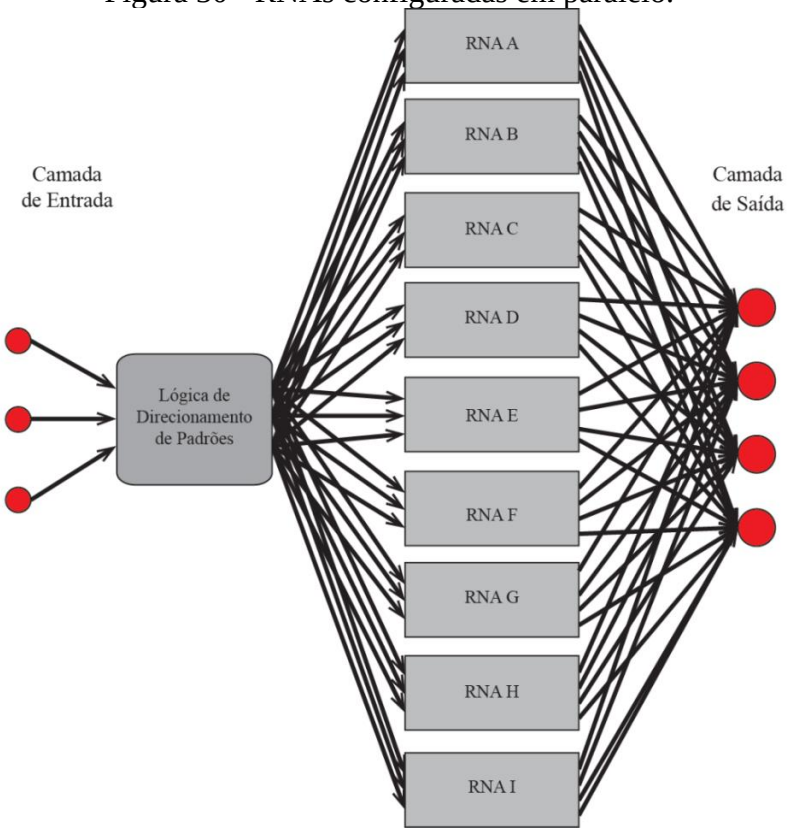
Para o desenvolvimento deste algoritmo foi adicionado um direcionador de padrões de treinamento que tem a finalidade de identificar qual RNA deve ser ativada para um padrão de treinamento específico, em relação à coordenada z do plano cartesiano, conforme definido na Tabela 5. A faixa de atuação de cada RNA no algoritmo foi determinada por meio de estudos da densidade de distribuição dos pontos no espaço de trabalho do manipulador robótico. Na Figura 30, ilustra-se a arquitetura do algoritmo com nove RNAs do tipo MLPs nomeadas de ‘A’ a ‘I’, configuradas em paralelo.

Tabela 5 - Discriminação do volume de trabalho para cada RNA

Rede Neural Artificial	**Faixa de operação em milímetros em relação à coordenada z**
RNA A	$Z \leq 100$
RNA B	$100 < Z \leq 150$
RNA C	$150 < Z \leq 200$
RNA D	$200 < Z \leq 250$
RNA E	$250 < Z \leq 300$
RNA F	$300 < Z \leq 350$
RNA G	$350 < Z \leq 390$
RNA H	$390 < Z \leq 430$
RNA I	$Z > 430$

Fonte: Elaborada pelo autor.

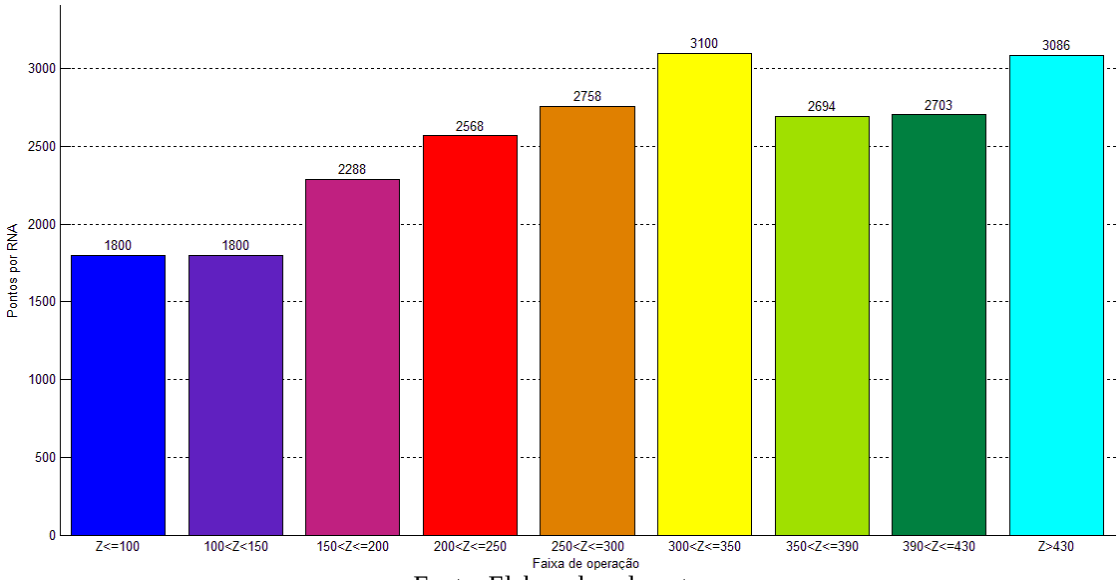
Figura 30 - RNAs configuradas em paralelo.



Fonte: Elaborada pelo autor.

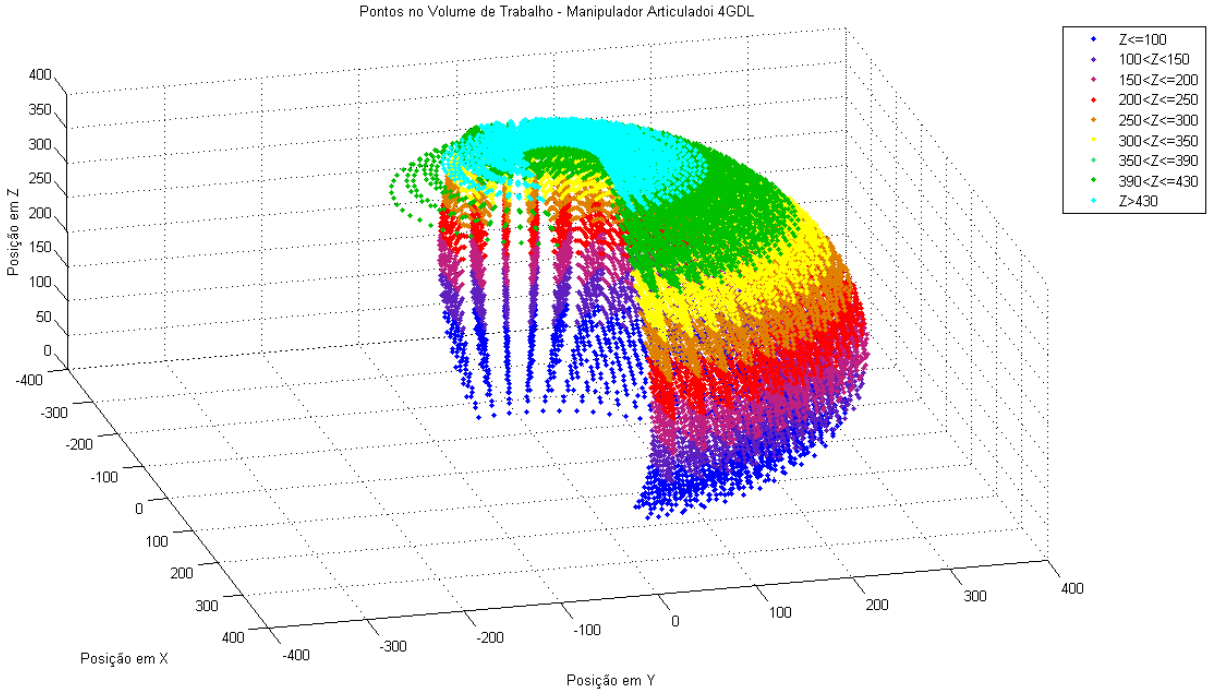
Graficamente, o resultado da densidade de pontos (padrões de treinamento) por RNA pode ser ilustrado pela Figura 31 e Figura 32. Nestas figuras observa-se que os números de pontos de padrões de treinamento gerados são distribuídos pelo conjunto de nove RNAs, de forma a facilitar a identificação de atuação de cada Rede Neural.

Figura 31 - Número de pontos para cada RNA.



Fonte: Elaborada pelo autor.

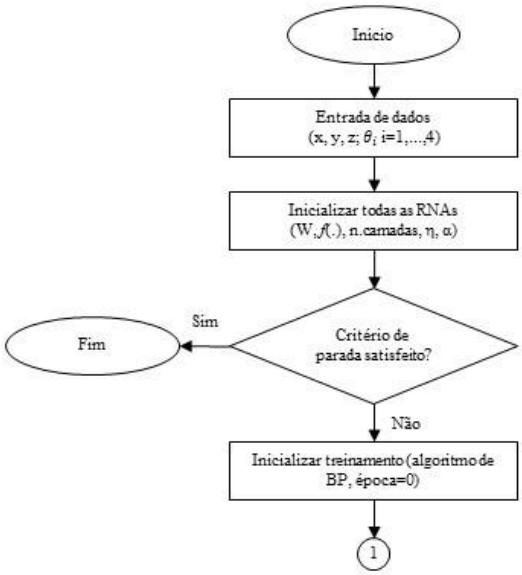
Figura 32 - Distribuição dos pontos por RNA no espaço de trabalho tridimensional.



Fonte: Elaborada pelo autor.

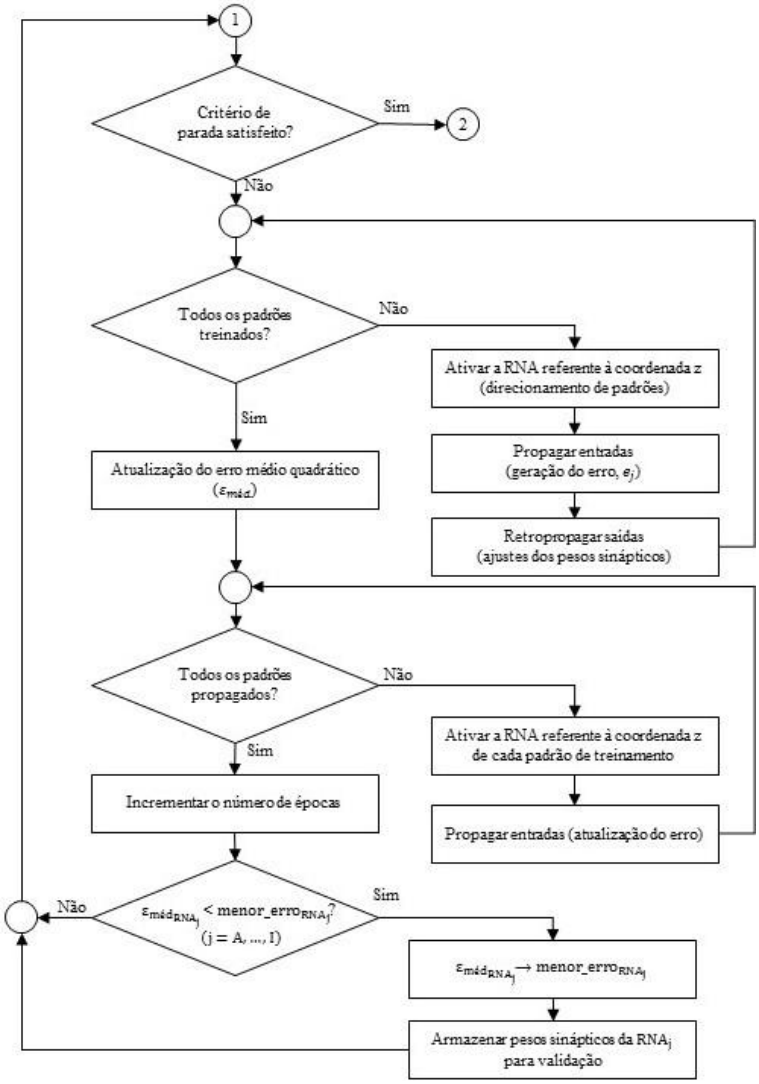
O algoritmo das RNAs configuradas em paralelo encontra-se no APÊNDICE B. Mostra-se a seguir, o mesmo algoritmo em fluxograma dividido em três partes: na Figura 33 tem-se o processo de inicialização, na Figura 34 o algoritmo de treinamento (BP) e na Figura 29Figura 35 a validação dos resultados.

Figura 33 - Fluxograma das MLP configuradas em paralelo, processo de inicialização.



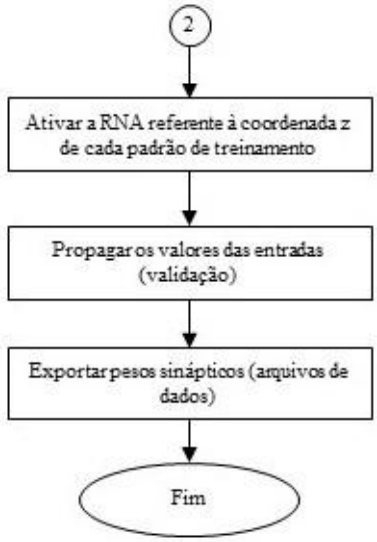
Fonte: Elaborado pelo autor.

Figura 34 - Fluxograma das MLP configuradas em paralelo, algoritmo de BP.



Fonte: Elaborado pelo autor.

Figura 35 - Fluxograma das MLP configuradas em paralelo, validação dos resultados.



Fonte: Elaborado pelo autor.

5.5 SIMULAÇÕES PARA O ACOMPANHAMENTO DE TRAJETÓRIAS

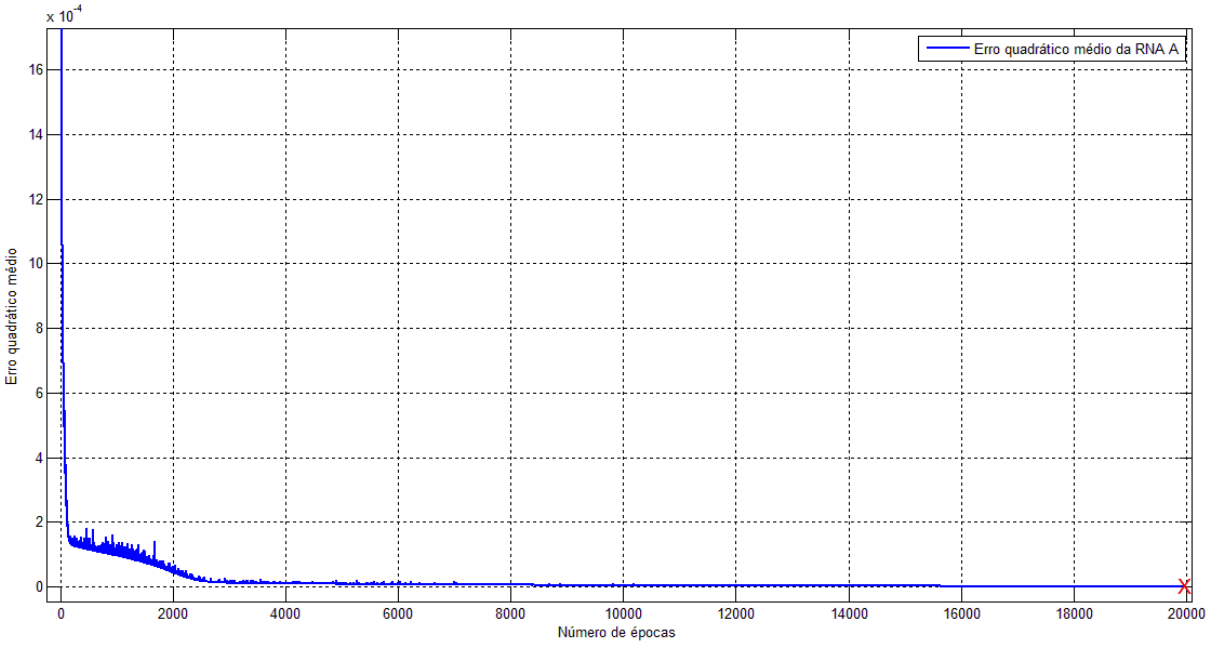
Com o objetivo de validar os algoritmos das RNAs abordados nos tópicos 0 e 0, ou seja, o algoritmo com somente uma Rede Neural e o algoritmo com o conjunto de Redes configuradas em paralelo, realiza-se os testes de treinamento e posteriormente, o acompanhamento de trajetórias.

Para validar as propostas das Redes Neurais utilizam-se os padrões de treinamento abordados no item 0. Para a primeira proposta utilizam-se os padrões de treinamento que representam a primeira e a segunda trajetórias, enquanto que para validar a segunda proposta de Rede Neural utilizam-se os padrões de treinamento que representam apenas a segunda trajetória.

5.5.1 Resultados da Simulação da primeira trajetória para uma RNA

Neste teste o desempenho da RNA durante o treinamento pode ser observado pelo gráfico da Figura 36, que apresenta a relação do erro médio quadrático por época, cujo treinamento foi interrompido na época de número 20.000. Depois do treinamento, a trajetória obtida em simulação e a desejada são apresentadas na Figura 37 e a variação dos ângulos obtidos e desejados durante a trajetória é mostrado na Figura 38.

Figura 36 - Erro versus época para treinamento da RNA para a primeira trajetória.



Fonte: Elaborada pelo autor

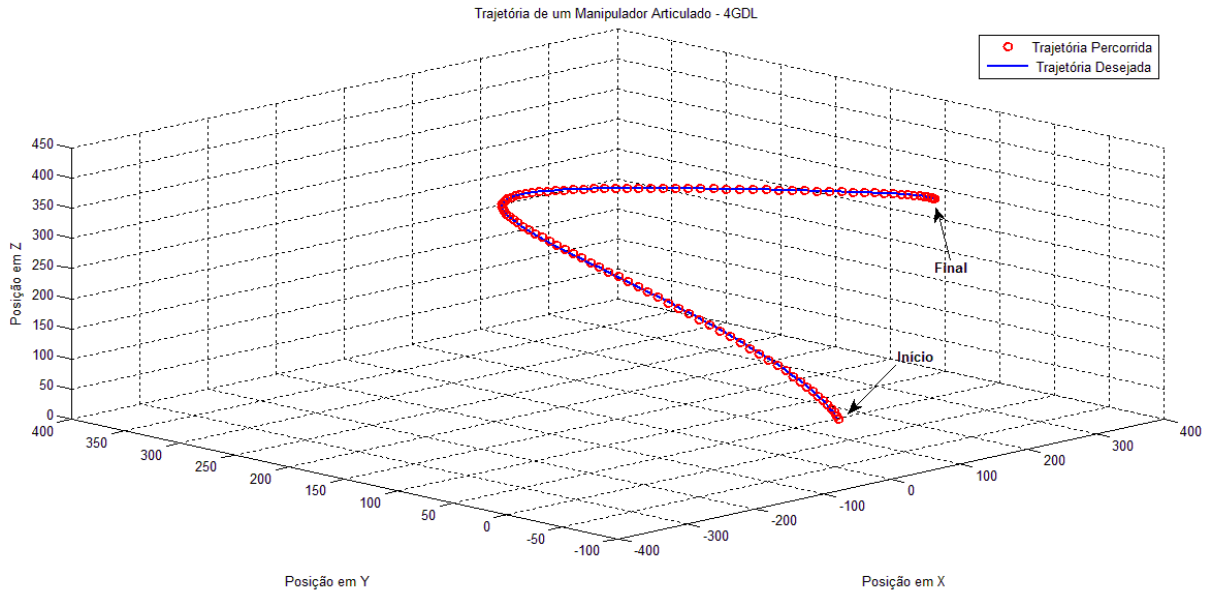
No treinamento a RNA apresentou um melhor desempenho na época de número 19.987, com o valor de erro quadrático médio de, aproximadamente, 10^{-6} (ponto ilustrado pela marca “X” em vermelho). Para validar a RNA, foram armazenados os pesos sinápticos desta época utilizados na fase de operação (ou propagação). Na fase de operação, os padrões de treinamento foram propagados novamente obtendo-se as médias absolutas de cada junta, dados na Tabela 6.

Tabela 6 - Primeira trajetória e uma RNA: médias aritméticas dos erros em valores absolutos para cada junta.

	Junta 1	**Junta 2**	**Junta 3**	**Junta 4**
Médias aritméticas dos erros	0,35º	0,14º	0,08º	0,17º

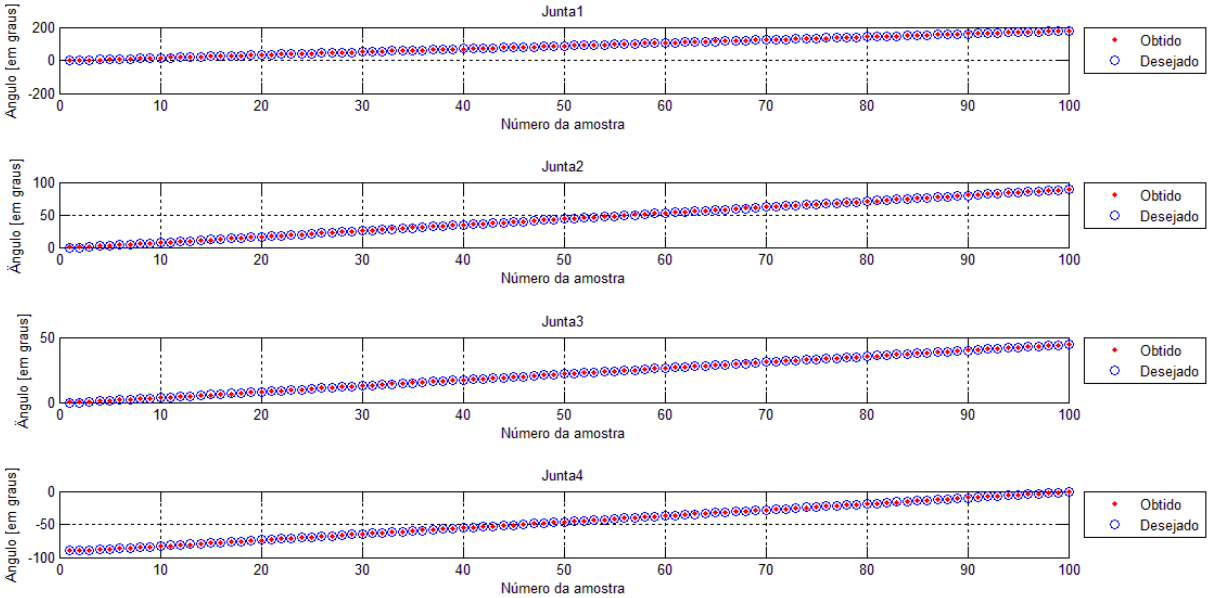
Fonte: Elaborada pelo autor.

Figura 37 - Primeira trajetória e uma RNA: percorrida e desejada



Fonte: Elaborada pelo autor.

Figura 38 - Variação dos ângulos durante a primeira trajetória para uma RNA.

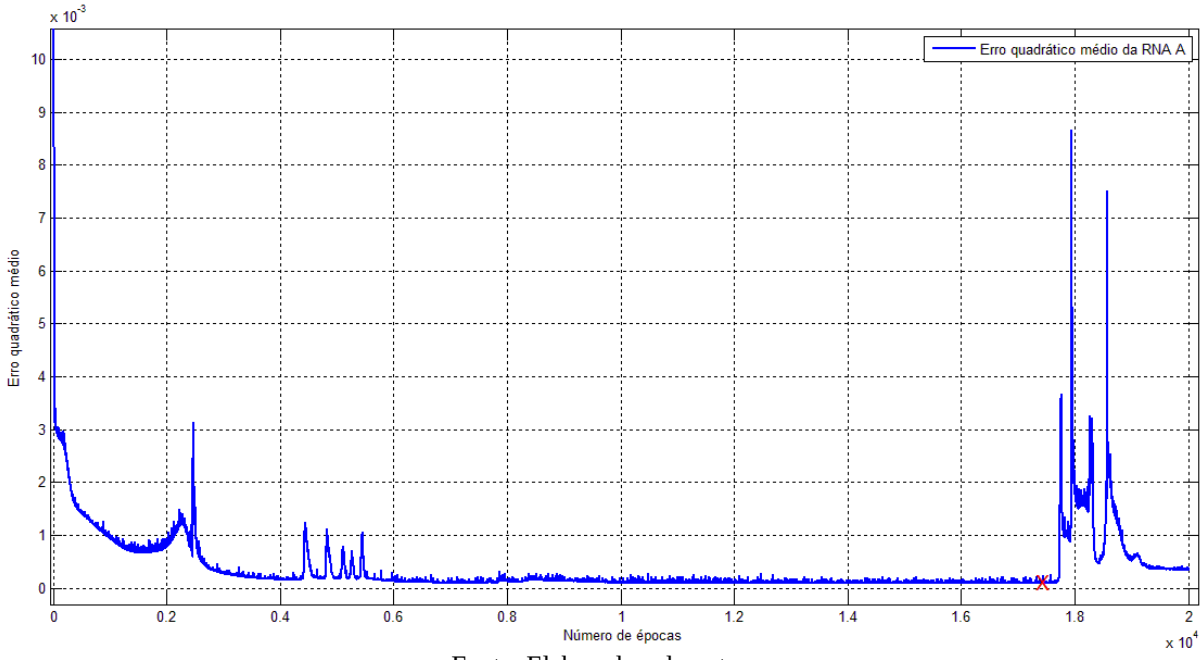


Fonte: Elaborada pelo autor.

5.5.2 Resultados da simulação da segunda trajetória para uma RNA

Para segunda trajetória, Figura 39, o processo de treinamento foi interrompido após obter a época de número 20000. A RNA obteve um melhor desempenho na época de número 17439 quando atingiu um erro quadrático médio de, aproximadamente, $9 \cdot 10^{-5}$. As matrizes de pesos desta época foram armazenadas para o processo de validação.

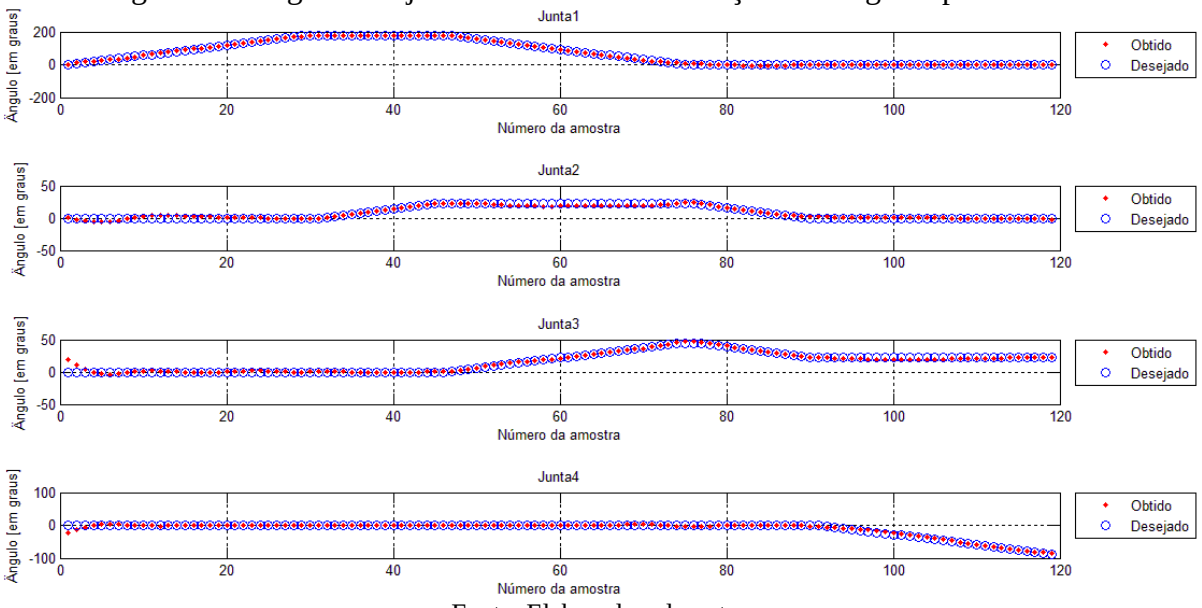
Figura 39 - Segunda trajetória e uma RNA: erro versus época.



Fonte: Elaborada pelo autor.

Após o treinamento, os padrões que representam a trajetória desejada foram propagados novamente para verificação da resposta do algoritmo. A variação dos ângulos das juntas - desejados e obtidos - no decorrer da trajetória está ilustrada na Figura 40. Após a validação da RNA foram obtidas as médias aritméticas dos erros, em graus e para cada junta, conforme Tabela 7. Finalmente, a representação da trajetória desejada e a percorrida é ilustrada pela Figura 41.

Figura 40 - Segunda trajetória e uma RNA: variação dos ângulos por amostra.



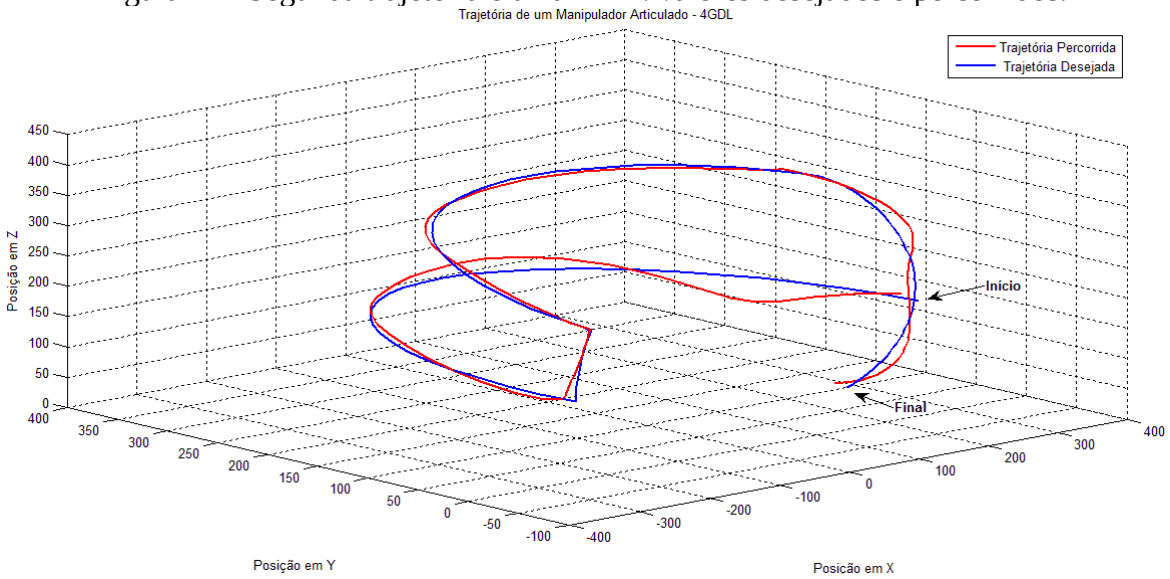
Fonte: Elaborada pelo autor.

Tabela 7 - Segunda trajetória e uma RNA: Médias aritméticas dos erros em valores absolutos, para cada junta.

	Junta 1	**Junta 2**	**Junta 3**	**Junta 4**
Médias aritméticas dos erros	1,92º	1,43º	1,27º	1,64º

Fonte: Elaborada pelo autor.

Figura 41 - Segunda trajetória e uma RNA: valores desejados e percorridos.

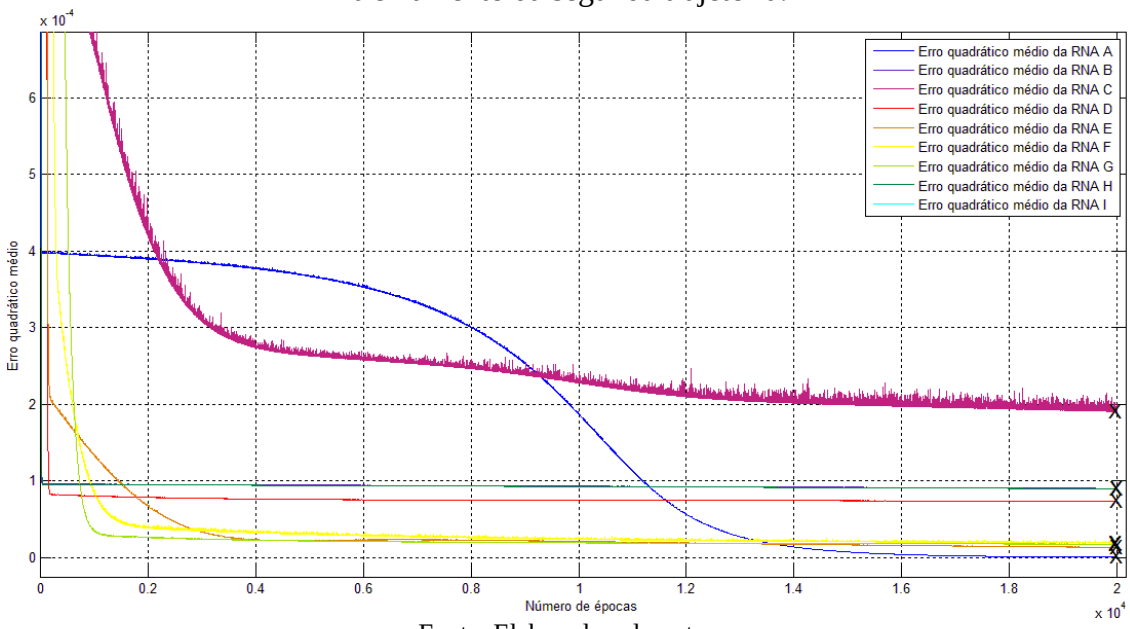


Fonte: Elaborada pelo autor.

5.5.3 Segunda trajetória com RNAs configuradas em paralelo

Um segundo treinamento foi realizado para a segunda trajetória, desta vez, o algoritmo utilizado foi o das RNAs do tipo MLP configuradas em paralelo, abordado no item 0. De mesma forma que os testes anteriores, o processo de treinamento foi também interrompido após obter a época de número 20000. O desempenho de cada RNA, durante o processo de treinamento, é representado pelo gráfico da Figura 42. Nota-se que não existe a curva da “RNA I” pois, esta não é ativada na segunda trajetória uma vez que apresenta pontos no espaço de trabalho com valor acima de 430 mm para a coordenada z.

Figura 42 - Curva de erro versus época de cada RNA configuradas em paralelo durante o treinamento da segunda trajetória.



Fonte: Elaborada pelo autor.

As matrizes de pesos das RNAs utilizadas no processo de validação do algoritmo foram obtidas pela Tabela 8, que representa os ápices de conhecimento de cada RNA. Na Tabela 9 apresenta-se a média aritmética de cada RNA, após a validação dos resultados. Na validação foram utilizados os mesmos padrões apresentados às RNAs na fase de treinamento, ou seja, todos os pontos que representam a trajetória.

Tabela 8 - Melhor desempenho de cada RNA para a segunda trajetória.

	RNA A	**RNA B**	**RNA C**	**RNA D**	**RNA E**	**RNA F**	**RNA G**	**RNA H**	**RNA I**
Menor erro	$4 \cdot 10^{-7}$	$9 \cdot 10^{-5}$	$2 \cdot 10^{-4}$	$7 \cdot 10^{-5}$	10^{-5}	$2 \cdot 10^{-5}$	$2 \cdot 10^{-5}$	$9 \cdot 10^{-5}$	-
Época	20000	19991	19986	19999	19998	19988	19988	19999	-

Fonte: Elaborada pelo autor.

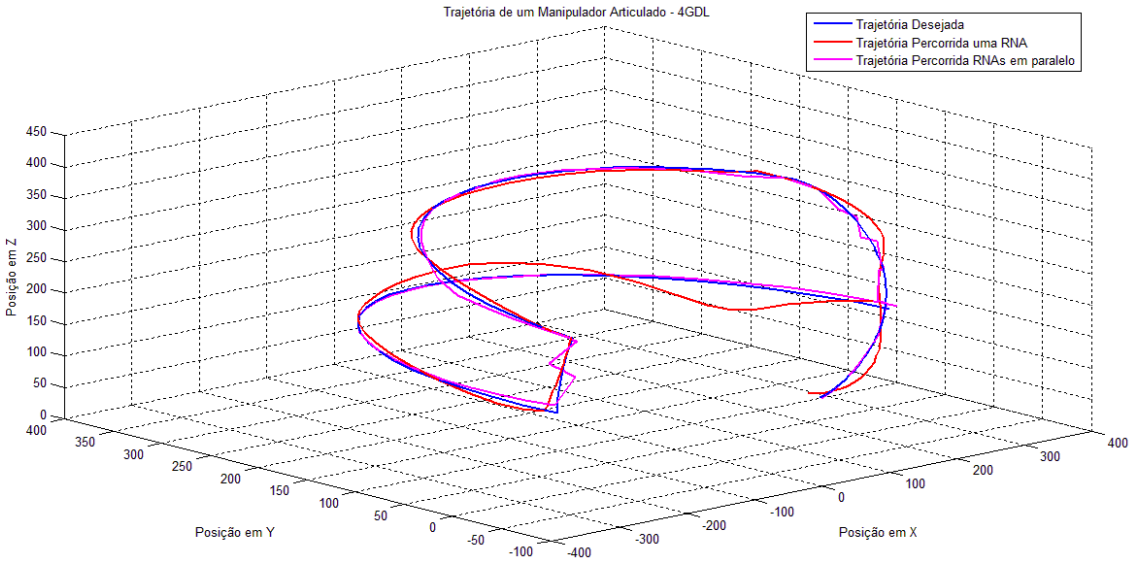
Tabela 9 - Média aritmética dos erros de cada RNA em valores absolutos, segunda trajetória.

	Junta 1	**Junta 2**	**Junta 3**	**Junta 4**
RNA A	0,19º	0,05º	0,11º	0,15º
RNA B	0,11º	0,05º	0,06º	4,45º
RNA C	1,00º	0,54º	2,77º	3,37º
RNA D	0,84º	1,40º	0,46º	2,43º
RNA E	0,80º	1,15º	0,41º	0,15º
RNA F	1,50º	0,68º	0,58º	0,02º
RNA G	0,48º	1,28º	1,90º	0,25º
RNA H	4,36º	0,53º	1,03º	0,02º
RNA I	-	-	-	-
Média das médias	**0,82º**	**0,61º**	**0,52º**	**0,20º**

Fonte: Elaborada pelo autor.

Com base nas médias de cada junta para uma RNA do tipo MLP apresentadas na Tabela 7 e pela média das médias apresentada na Tabela 8, nota-se que o algoritmo que utiliza MLPs configuradas em paralelo obteve um melhor desempenho, pois, apresentou menores médias de erros para cada junta, o que pode ser reforçado pela Figura 43, que ilustra os desempenhos dos testes citados em relação a trajetória desejada. Na junta 4 a redução do erro foi de, aproximadamente, 87,8%. Esta comparação é válida, pois, as RNAs utilizadas em ambos os algoritmos tiveram os mesmos parâmetros e os treinamentos foram interrompidos com o mesmo número de época.

Figura 43 – Comparando o desempenho entre algoritmos para a segunda trajetória.



Fonte: Elaborada pelo autor.

Com base na figura, observa-se pelo resultado da trajetória do algoritmo das RNA em paralelo, que ocorreu uma oscilação ou “ziguezague”, quando o movimento se deu basicamente pela coordenada z do plano cartesiano. Uma possível justificativa para esse movimento é a transição da ativação das RNAs de acordo com o valor da coordenada em questão.

Apresentados os desempenhos dos algoritmos das duas propostas de RNAs para o treinamento e acompanhamento de trajetórias, descrevem-se a seguir, os resultados das simulações do treinamento para o conjunto de pontos distribuídos em todo o volume de trabalho, programadas no MATLAB.

5.6 SIMULAÇÕES DO TREINAMENTO CONSIDERANDO O VOLUME DE TRABALHO

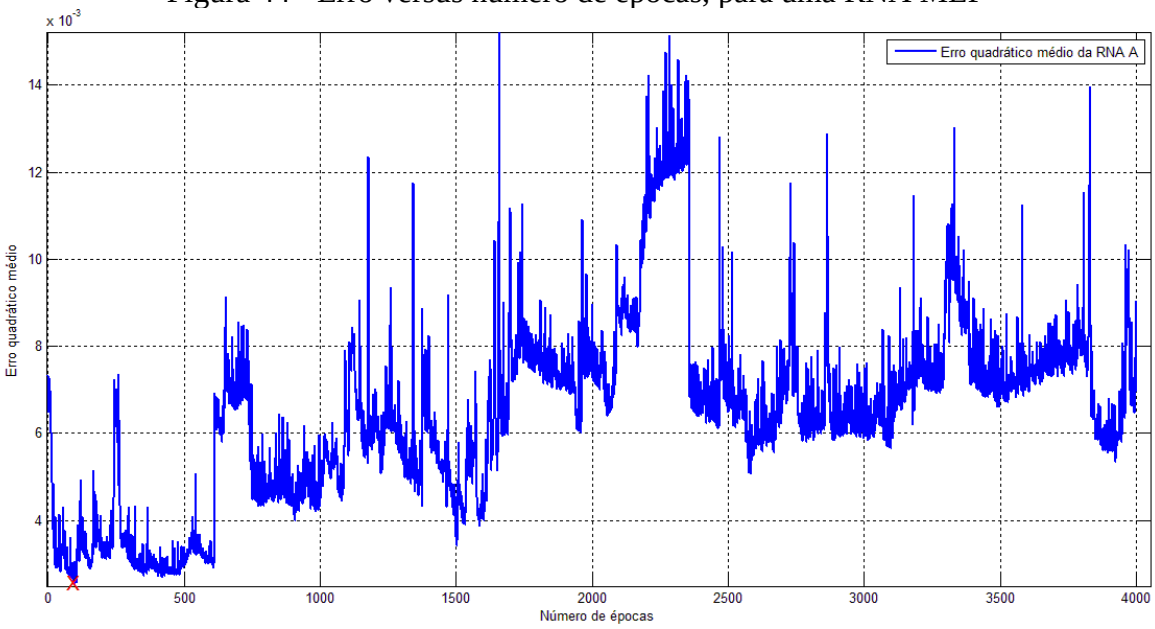
De forma a comparar o desempenho das duas Redes Neurais é realizado o mapeamento de padrões de entrada e saídas para o conjunto de pontos distribuídos em todo o volume de trabalho, utilizando primeiramente uma RNA MLP e posteriormente as RNAs com MLPs configuradas em paralelo.

5.6.1 Simulação para uma RNA MLP

Nesta simulação utiliza-se a RNA MLP com a mesma configuração adotada anteriormente, e o treinamento foi realizado até que atinja a época de número 4000. Este número de épocas inferior em relação aos testes com trajetórias justifica-se pela quantidade maior de padrões no espaço de trabalho, apresentados a Rede Neural. Esse treinamento foi realizado em aproximadamente, 10 horas e 30 minutos de processamento.

O desempenho da RNA durante a fase de treinamento é representado pelo gráfico da relação entre erro quadrático médio por época ilustrado na Figura 44. Durante o treinamento a RNA apresentou um melhor desempenho na época de número 94, com um erro quadrático médio de 0,0025 – representado pela marcação em “X” no gráfico. As matrizes de pesos geradas nesta época foram armazenadas e utilizadas no processo de validação do algoritmo que constou na propagação de todos os pontos gerados em todo o espaço de trabalho do manipulador. Na Tabela 10 mostram-se os resultados desta RNA, e as médias absolutas dos erros, em graus, para cada junta.

Figura 44 - Erro versus número de épocas, para uma RNA MLP



Fonte: Elaborado pelo autor.

Tabela 10 - Médias aritméticas dos erros em valores absolutos da RNA treinada para todo o volume de trabalho.

	Junta 1	**Junta 2**	**Junta 3**	**Junta 4**
Média aritméticas dos erros	7,84º	8,74º	12,17º	10,33º

Fonte: Elaborado pelo autor.

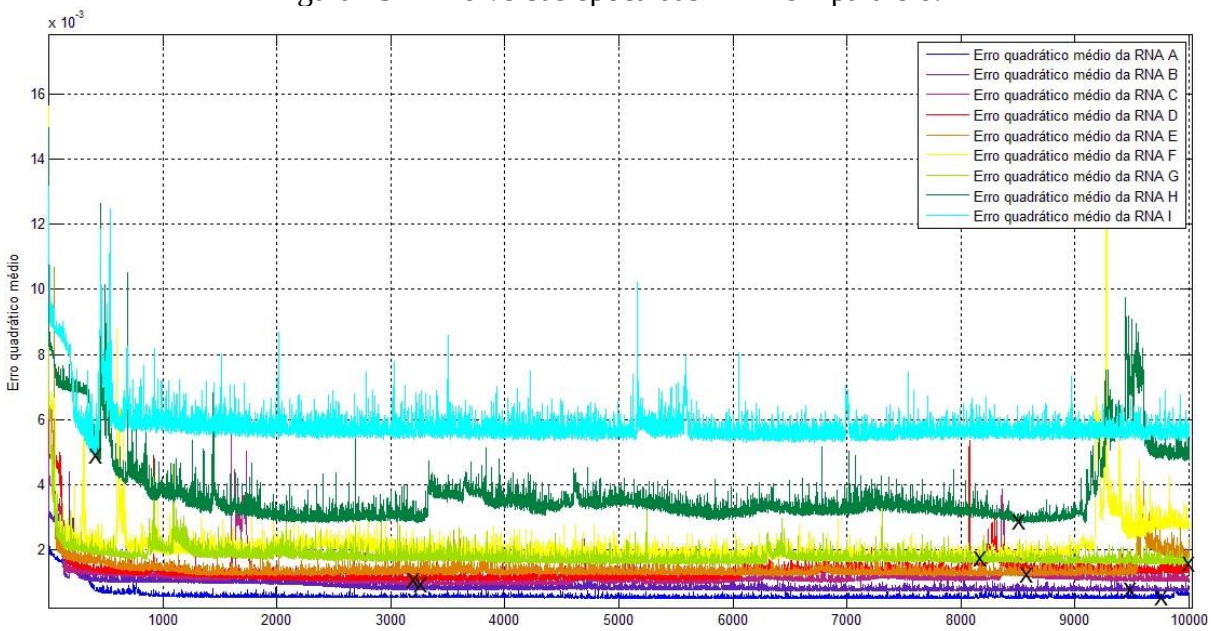
Com base no desempenho da RNA durante o treinamento, Figura 44, observa-se que o algoritmo composto por uma RNA se tornou incapaz de melhorar o aprendizado da relação cinemática do manipulador robótico, apresentando um comportamento de saturação (época de número 500, aproximadamente) de conhecimento.

5.6.2 Simulação com MLPs em Paralelo

Nesta simulação utiliza-se o algoritmo das RNAs MLPs configuradas em paralelo abordado no item 0, cujo treinamento foi interrompido na época de número 10000, com tempo de processamento de aproximadamente 27 horas. O melhor desempenho, representado pelo menor erro quadrático médio, marcado com “X” é mostrado na Figura 45, cujos valores e nas suas respectivas épocas são apresentados na Tabela 11.

O conjunto de pesos sinápticos destes pontos foram utilizados na fase de operação e as médias dos erros, em graus, de cada junta de todas as Redes Neurais são apresentadas em uma nova tabela, Tabela 12.

Figura 45 - Erro versus época das RNA em paralelo.



Fonte: Elaborado pelo autor.

Tabela 11 - Melhor desempenho de cada RNA para o volume de trabalho

	RNA A	**RNA B**	**RNA C**	**RNA D**	**RNA E**	**RNA F**	**RNA G**	**RNA H**	**RNA I**
Menor erro	0,0005	0,0007	0,0009	0,0011	0,0012	0,0017	0,0015	0,0028	0,0049
Época	9762	9491	3270	3204	8583	8173	9997	8516	414

Fonte: Elaborado pelo autor.

Tabela 12 - Médias aritméticas dos erros das juntas em valores absolutos para cada RNA

	Junta 1	**Junta 2**	**Junta 3**	**Junta 4**
RNA A	1,22º	4,42º	7,65º	3,44º
RNA B	0,80º	5,56º	9,19º	4,17º
RNA C	1,19º	6,28º	10,74º	4,84º
RNA D	0,76º	7,30º	11,90º	4,90º
RNA E	1,55º	7,86º	11,63º	5,78º
RNA F	3,14º	8,92º	11,82º	8,69º
RNA G	2,02º	8,67º	11,85º	8,07º
RNA H	7,78º	9,14º	10,26º	10,64º
RNA I	20,45º	9,87º	7,79º	14,73º
Média das médias	**1,55º**	**7,86º**	**10,74º**	**5,78º**

Fonte: Elaborado pelo autor.

Com base na média aritmética do erro das juntas do algoritmo que utiliza uma RNA, representado na Tabela 10 e nas médias das médias das RNA configuradas em paralelo, representados pela Tabela 12, pode-se constatar que, novamente, o algoritmo que utiliza a configuração de RNAs em paralelo obteve um melhor desempenho no aprendizado da relação cinemática do manipulador em estudo. Na junta 1, a redução do erro foi de aproximadamente de 80%. É importante salientar que, neste caso, não foi usado no treinamento o mesmo número de épocas, pois como já observado anteriormente, o algoritmo com uma RNA apresentou um comportamento de saturação em torno de 500 (e menor erro na época 94), descartando a necessidade de um treinamento mais longo.

5.7 COMENTÁRIOS

Neste capítulo foram gerados os padrões de treinamento para duas trajetórias e um conjunto de pontos distribuídos no volume de trabalho para dois algoritmos de RNAs, um por uma RNA simples e outro pela configuração em paralelo de 9 RNAs, ambos os casos, utilizando a mesma topologia de RNA, a MLP. Foram apresentados os resultados das simulações no software MATLAB.

CAPÍTULO 6. O PROTÓTIPO E A IMPLEMENTAÇÃO EM HARDWARE DO ALGORITMO DAS RNAS CONFIGURADAS EM PARALELO

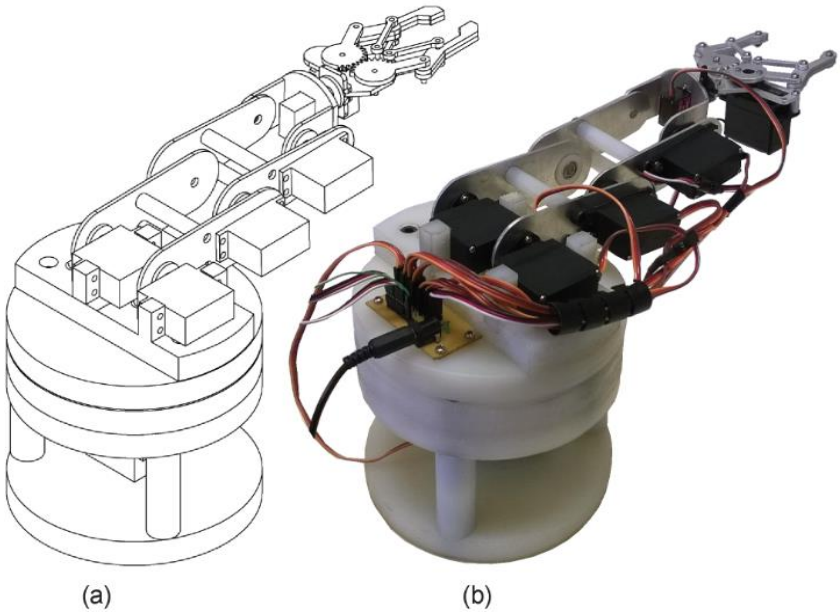
Após o treinamento das RNAs obtidos no modo *offline*, neste capítulo descrevem-se os detalhes do hardware na fase *online*, que se refere ao interfaceamento da plataforma Intel Galileo Gen 2 com o *notebook* e manipulador, servomotores, linguagem de programação, fonte de alimentação, memória, etc.

Em seguida apresentam-se o algoritmo *online*, da RNA MLP configurada em paralelo aplicado ao manipulador, para realizar seu posicionamento via a cinemática inversa e o resultado dos testes simulando uma trajetória.

6.1 O PROTÓTIPO

O manipulador robótico foi projetado no software de CAD SolidWorks, conforme modelo apresentado na Figura 46(a), cujo modelo real é mostrado na Figura 46(b), confeccionado nos laboratórios do DEE-FEIS-UNESP e do IFSP Câmpus Presidente Epitácio, suas características são representadas no APÊNDICE C.

Figura 46 - Arquitetura do manipulador robótico. (a) Modelo (b) Protótipo.



Fonte: Elaborada pelo autor.

Trata-se de um manipulador articulado (RRR) de 5 GDL, sendo três graus de liberdades necessários para o movimento do “braço” e dois para o “pulso”. Os elos são de

alumínio, material comumente utilizado na área da robótica, por ser resistente e leve. No conjunto da base foi usado o nylon, material de fácil usinagem que proporciona ao manipulador sustentação e confiabilidade na realização de suas tarefas.

Os movimentos do protótipo são proporcionados pelo acionamento de sete servomotores, sendo seis para o movimento do braço, antebraço e punho do manipulador e um para garra. Seus parâmetros e a forma de como foram alocados no protótipo são apresentados na Tabela 13.

Tabela 13 - Dados dos servos utilizados.

Servomotor	**Alocação**	**Torque a 4.8V** (kgf·cm)	**Peso (g)**
MG995 (3) Futaba	Base, antebraço, punho (vertical)	9,4	55
MG996R (2) TowerPRO	Braço	9,4	55
MG90S (1) TOwerPRO	Punho (giro)	1,8	13,4
S3003 (1) TowerPRO	Garra	3,2	37,2

Fonte: Elaborada pelo autor.

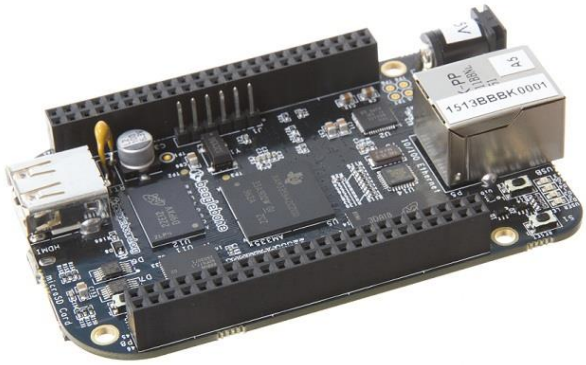
6.2 PLATAFORMAS DE DESENVOLVIMENTO

Para controlar um processo dedicado a tarefas específicas, um manipulador robótico, por exemplo, utilizam-se, normalmente placas de desenvolvimento chamadas de *Single Board Computers* (SBCs) ou *Single Board Controller* (SBCt). Estas são computadores montados em uma única placa, constituídos por microprocessador, memórias, entradas e saídas (I/Os), comunicação serial, etc., com objetivos educacionais, para o desenvolvimento de protótipos e/ou compor sistemas embarcados. São em geral, computadores simples, com pouca ou nenhuma possibilidade de expansão. Dentre as SBCs presentes no mercado, tem-se a *BeagleBone Black*, *RaspberryPi*, Arduino e mais recentemente a Intel® GalileoGen 2.

6.2.1 BeagleBone Black

A BeagleBone Black (BBB) foi desenvolvida pela Fundação *BeagleBoard*, nos Estados Unidos. Entre outros componentes, a BBB, Figura 47, está baseado no SoC (*System on Chip*) processador Sitara XAM3359AZCZ100 de 1GHz, da Texas Instruments, memória RAM 512 MB DDR3L de 800 MHz, conexões externas através de portas USB Host, Mini-USBClient e 1 10/100 Mbps Ethernet. Possui 65 pinos de propósitos gerais, ou do inglês-*General Purpose Input/Output* (GPIO), oito delas podendo ser configuradas como PWM (*Pulse Width Modulation*). Um SoC (*System-on-a-chip*) é um chip que contem quase todos os componentes básicos para um computador, como memória RAM e processador. Na Tabela 14 são resumidas as informações técnicas do BBB.

Figura 47 – BeagleBone Black.



Fonte: Beagleboard (2015).

Tabela 14 - Informações técnicas do BeagleBone Black

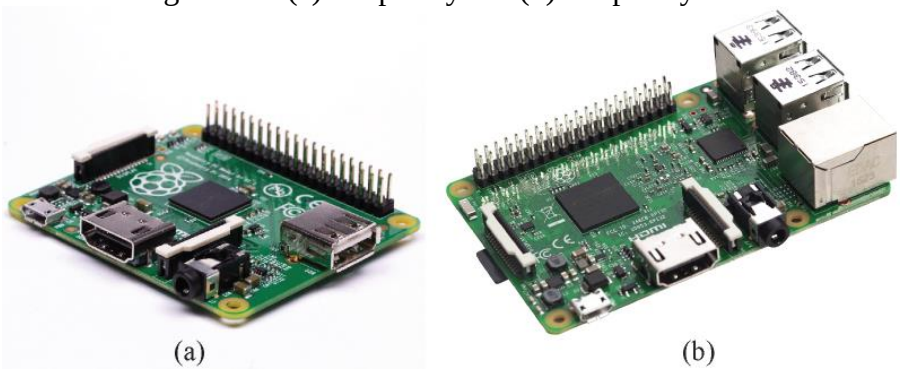
Processador	ARM Cortex A8 1GHz
RAM	512 MB DDR3L 800 MHz
Armazenamento	4 GB on-board eMMC (expansível via MicroSD)
Saídas de vídeo	1 Micro-HDMI
Saídas de áudio	Stereo através do HDMI
Sistemas operacionais	Angstrom (Default), Ubuntu, Android, ArchLinux, Gentoo, Minix, RISC OS, outros
GPIO	65 Pinos
Periféricos	1 USB Host, 1 Mini-USBClient, 1 10/100 Mbps Ethernet
Alimentação	5V, via miniUSB, USB ou Fonte DC

Fonte: Rodrigues; Mantovani (2015).

6.2.2 RaspberryPi

O RaspberryPi (RPi) é um SBC desenvolvido pela Fundação RaspberryPi - localizada na Inglaterra, sem fins lucrativos. O objetivo desta fundação, através da RPi é de motivar nas escolas, o ensino de programação e Ciências da Computação, por isso, esta placa possui preço baixo. O primeiro modelo foi lançado em 2012 e, atualmente, são comercializados alguns modelos, entre eles o Raspberry Pi 3 B e o Raspberry A+, ambos mostrados na Figura 48 (RASPBERRY PI FOUNDATION, 2015).

Figura 48 - (a) Raspberry A+ (b) RaspberryPi 3 B.



Fonte: Raspberry Pi Foundation (2015).

Suas principais características são apresentados na Tabela 15. A diferença principal entre os dois modelos apresentados é o poder de processamento e as dimensões por conta das conexões externas, o modelo A+ é mais barato e leve do que o modelo 3 B.

Tabela 15 - Principais características do Raspberry modelos A+ e 3 B.

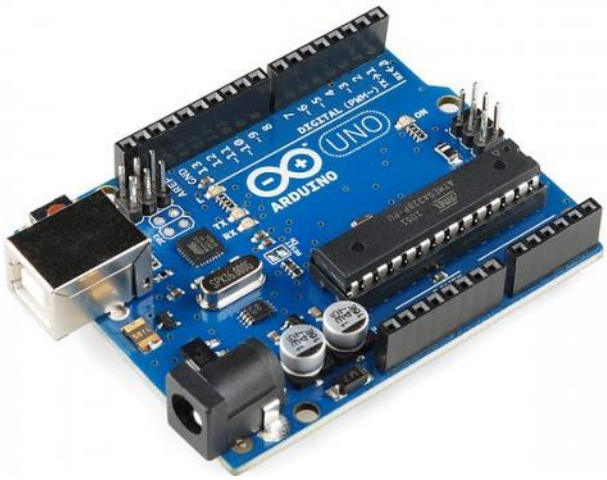
	Raspberry Pi A+	**Raspberry Pi 3 B**
Processador	Broadcom BCM2835 32Bit SoC Single Core 700 MHz	Broadcom BCM2837 64bit ARMv7 Quad Core 1.2GHz
RAM	256 MB SDRAM 400 MHz	1GB SDRAM, 400 MHz
Armazenamento	microSD Card, expansível até 32GB	microSD Card, expansível até 64GB
Saídas de vídeo	Jack de 3,5 mm e HDMI	Jack de 3,5 mm e HDMI
Saídas de áudio	Stereo, Jack de 3,5 mm de áudio	Stereo, Jack de 3,5 mm de áudio
USB	1 porta	4 portas
Sistemas operacionais	Linux	Linux e Windows 10
GPIO	40 pinos	40 pinos
Outros periféricos	UART, I²C, SPI	Wi-Fi embutido, Bluetooth 4.1, Ethernet, CSI camera e DSI display.
Alimentação	5V e 1.8A	5V e 2,5 A
Dimensões	65mm x 56mm	85mm x 56mm

Fonte: (RASPBERRY PI FOUNDATION, 2015).

6.2.3 Arduino

Arduino é uma plataforma de prototipagem de hardware e software abertos (em inglês, *open source*), cujo objetivo do projeto Arduino é incentivar projetos de baixos custos com uma ferramenta fácil para prototipagem rápida, destinado a estudantes sem experiência em eletrônica e programação. O Arduino é muito popular e tem sido utilizado em inúmeras aplicações, inclusive a robótica. Dentre os modelos das placas Arduino, uma das mais populares é o modelo UNO, representado na Figura 49, baseada no microcontrolador ATmega328P, suas características principais são apresentadas na Tabela 16.

Figura 49 - Arduino UNO.



Fonte: (ARDUINO, 2016)

Tabela 16 - Principais características da plataforma Arduino UNO.

Microcontrolador	ATmega328P 16 MHz
SRAM	2Kb (ATmega328P)
Armazenamento	Memória Flash 32 KB (ATmega328P)
USB	USB host e USB cliente
Sistemas operacionais	Linux, Windows e Mac via Arduino (IDE)
GPIO	20 pinos, 14 digitais (6 com saída PWM 8 bit) e 6 entradas analógicas
Periféricos	I^2C
Alimentação	de 6V a 12V
Dimensões	68,6 mm x 53,4 mm

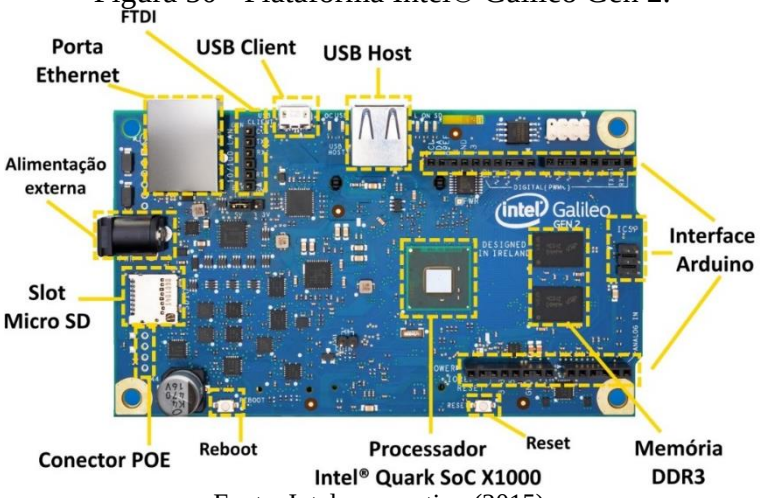
Fonte: (ARDUINO, 2016)

6.2.4 Intel® Galileo Gen 2

A plataforma Intel® Galileo Gen 2 mostrada na Figura 50 é classificada como uma *Single Board Controller* (SBCt), ou seja, são consideradas como controladores montados em uma única placa, constituídos por microprocessador, memórias, entradas e saídas (I/Os), comunicação serial, etc., fabricada com objetivos educacionais, para o desenvolvimento de protótipos e/ou compor sistemas embarcados. São em geral, computadores simples, com pouca ou nenhuma possibilidade de expansão.

Esta é uma plataforma de *hardware* livre sucessora da Intel® Galileo, apresentando compatibilidade em software, hardware e pinos com placas da família Arduino, sendo que suas principais características são apresentadas na Tabela 17.

Figura 50 - Plataforma Intel® Galileo Gen 2.



Fonte: Intel corporation (2015).

Tabela 17 - Principais características da Intel Galileo Gen 2.

Processador	Intel® Quark™ SoC X1000 (16K Cache, 400 MHz)
RAM	256 MB DDR3
Armazenamento	slot microSD, expansível até 32GB
USB	USB host e USB cliente
Sistemas operacionais	Linux, além de Windows e Mac via Arduino (IDE)
GPIO	20 pinos, 14 digitais (6 com saída PWM 8/12-bit) e 6 entradas analógicas
Outros periféricos	FTDI, mini-PCI Express e Ethernet
Alimentação	de 7V a 15V
Dimensões	124 mm x 72 mm

Fonte: (INTEL CORPORATION, 2015; RAMON, 2014).

Pode ser programada em Python ou também em C, C++ e Node.js. Além das características apresentadas na tabela esta placa possui também memória SRAM embarcada de 512 KB, NOR Flash de 8 MB e EEPROM padrão de 8 KB on-board (INTEL CORPORATION, 2015).

Optou-se por esta placa para compor o controlador do manipulador robótico deste trabalho, por conter os recursos de hardware e software necessários para o sistema robótico, tais como, número de saídas PWM, capacidade de processamento e de memória suficientes, interfaceamento via cabo ethernet e a possibilidade de programação em linguagens de alto nível, como a linguagem Python, utilizada na realização dos programas de testes da fase *online*.

6.3 A MONTAGEM COM O PROTÓTIPO

Visando a aplicação das RNAs MLPs e o posicionamento do manipulador por cinemática inversa, fez-se a montagem com o protótipo da forma mostrada nas Figura 51 e Figura 52. Nestas tem-se o sistema formado pelos seguintes componentes:

- Manipulador robótico projetado;
- Plataforma de controle Intel Galileo Gen 2;
- *Notebook*;
- Placa de interface entre o manipulador e o Intel Galileo Gen 2;
- Cabo Ethernet para comunicação entre a plataforma de controle e o *notebook*;
- Cartão microSD;
- Fonte de alimentação externa.

Para realizar o acionamento do manipulador e a etapa de propagação *online* do algoritmo desenvolvido, a plataforma de controle Intel® Galileo Gen 2 é ligada a um *notebook* via um cabo ethernet para o acesso, usando o protocolo de rede SSH (Secure Shell), e por intermédio do software de emulação de terminal, PuTTy.

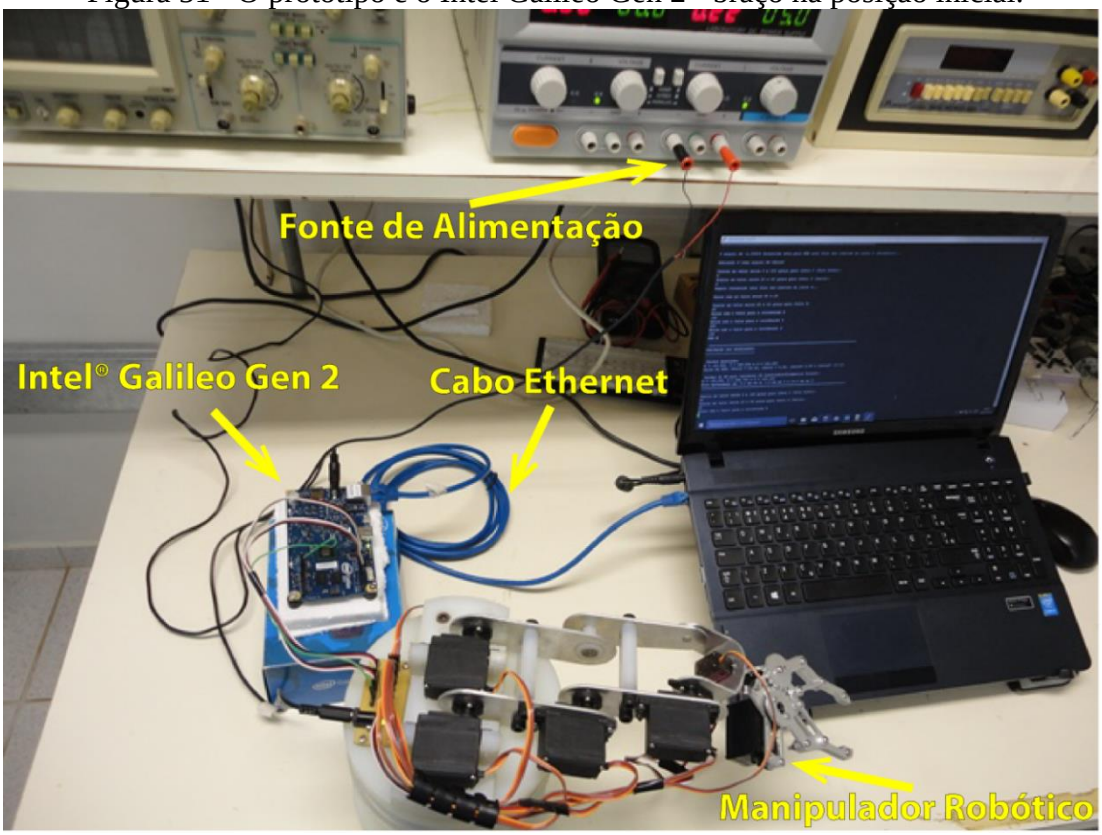
O *notebook* é utilizado para a visualização, entrada de dados e comandos por teclado para o sistema, pois, na fase *online*, o manipulador robótico é acionado com base nos parâmetros desejados pelo usuário.

Uma interface entre o manipulador e a plataforma de controle se dá por uma placa, cujo circuito esquemático é apresentado no APÊNDICE D, que recebe alimentação da interface por uma fonte externa de 5 VCC e alimenta os servomotores, localizados nas juntas

do manipulador, e interliga os terminais de acionamento. De cada servomotor derivam-se três terminais, o positivo (alimentação), negativo (comum) e o sinal de controle feito por PWM. Os terminais dos servomotores são conectados às entradas GPIO da plataforma de controle, sendo que, dois servos, alocados na junta 2 (braço) do manipulador, são acionados juntos.

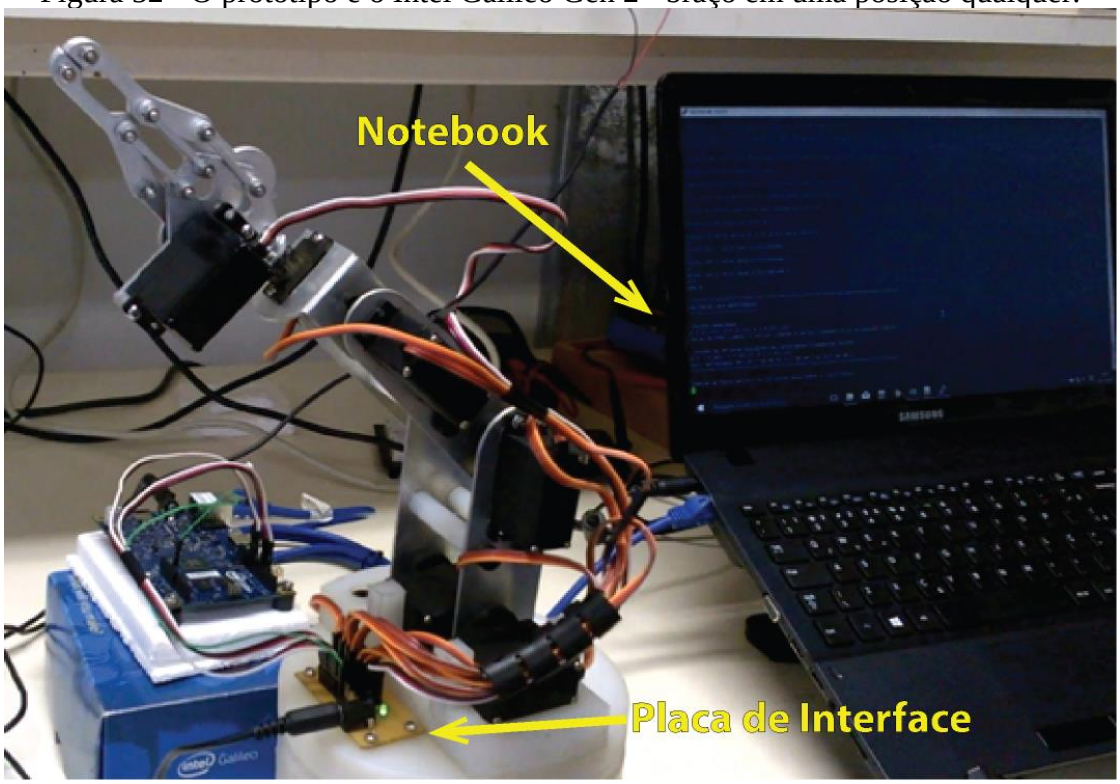
O acionamento dos atuadores (servomotores) é executado por um programa residente na placa Galileo Gen 2 por meio de seis sinais de PWM configurado em 50Hz, com *duty cycle* variando de 5 a 10 por cento do período.

Figura 51 - O protótipo e o Intel Galileo Gen 2 - braço na posição inicial.



Fonte: Elaborada pelo autor.

Figura 52 - O protótipo e o Intel Galileo Gen 2 - braço em uma posição qualquer.



Fonte: Elaborada pelo autor.

Um cartão microSD (*Secure Digital Card*) de 16GB conectado na plataforma de controle armazena o Sistema Operacional do Linux (Yocto), as matrizes de pesos sinápticos e o arquivo na linguagem Python que contém o algoritmo da fase *online*, que implementa a RNA e faz o acionamento dos servomotores.

6.4 APRESENTAÇÃO DO ALGORITMO NA FASE *ONLINE*

Inicia-se a fase *online* pela importação das matrizes dos valores dos pesos sinápticos, arquivos .csv (*Comma-separated values*), uns dos formatos reconhecidos pela linguagem Python, os quais representam o conhecimento das RNAs. Nesta fase é processada apenas a propagação do algoritmo da RNA configuradas em paralelo, implementado na plataforma Intel® Galileo Gen 2 e destaca-se que todos os ângulos são considerados para o acionamento do manipulador, inclusive o ângulo θ_5, responsável pelo movimento de giro do pulso, e o θ_6, ângulo de abertura da garra. Descreve-se a seguir o algoritmo da fase *online*:

1. Início: Importar as bibliotecas de programação e definir as portas de saída;
2. Importar as matrizes de pesos sinápticos;
3. Mover manipulador posição inicial ($\theta_1 = \theta_2 = \theta_3 = \theta_4 = \theta_5 = \theta_6(Garra) = 0$);

4. Solicitar ao usuário os valores das coordenadas x, y e z;

5. Identificar a RNA a ser ativada, de “A” a “I”, normalizar os valores e propagá-los até a saída;

6. Validar os valores de saída da RNA por cinemática direta (posição obtida) e comparar com os dados solicitados no item **4.** deste algoritmo;

7. Acionar a junta k do manipulador, sendo, $1 \leq k \leq 4$ com base nas respostas obtidas pela RNA ativada, se os ângulos fornecidos respeitarem os limites de junta k, caso contrário:

7.1. Adotar o valor de $\theta_{k_{mín}}$ para θ_k, se o valor de ângulo fornecido pela RNA ativada for menor que $\theta_{k_{mín}}$.

7.2. Adotar o valor de $\theta_{k_{max}}$ para θ_k se o valor de ângulo fornecido pela RNA ativada for maior que $\theta_{k_{máx}}$.

8. Solicitar ao usuário um valor de θ_5 dentro dos limites da junta 5, caso contrário:

8.1. Solicitar novamente ao usuário um valor de θ_5 dentro dos limites da junta 5.

9. Solicitar ao usuário um valor de θ_6 dentro dos limites da garra, caso contrário:

9.1. Solicitar novamente ao usuário um valor de θ_6 dentro dos limites da junta 6.

10. Mover o manipulador de acordo com configurações de ângulos (obtidos pela RNA e solicitados pelo usuário);

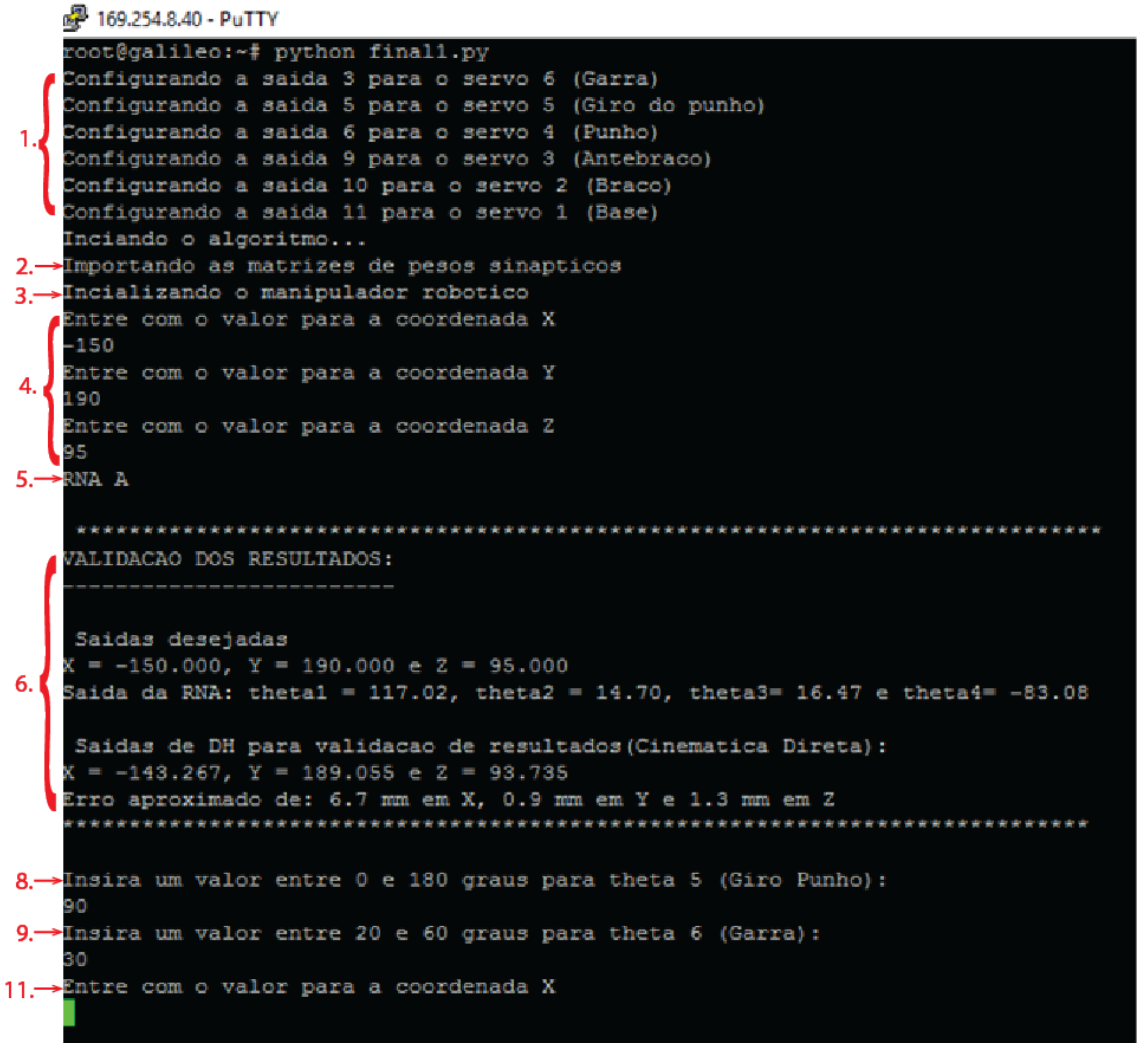
11. Retornar para o item **4.**

Deste algoritmo, algumas observações podem ser feitas:

- A partir do item **4.** o algoritmo trabalha com um loop de repetição, entretanto, a resposta da RNA anterior não interfere na próxima resposta pois, a propagação é realizada de forma independente. Por outro lado, por se tratar de acionamento sem realimentação sensorial, optou-se por controlar a velocidade dos servomotores pelo acionamento atenuado do valor do ângulo, desta forma, a resposta anterior é necessária, evitando acionamentos bruscos;
- Por segurança, em todas as etapas onde envolvem acionamentos das juntas do manipulador é realizado uma verificação prévia de seus valores;
- O ângulo para abertura ou fechamento da garra deve atender uma faixa de valores entre 20 e 60 graus, determinada de forma empírica. Esta informação é impressa na tela do *notebook*, no momento da solicitação, assim como a faixa de valores de θ_5.

A execução das etapas descritas no algoritmo da fase *online* é apresentada na tela do *notebook*, Figura 53.

Figura 53 - Interface do Galileo Gen 2 no ambiente Linux.



Fonte: Elaborada pelo autor.

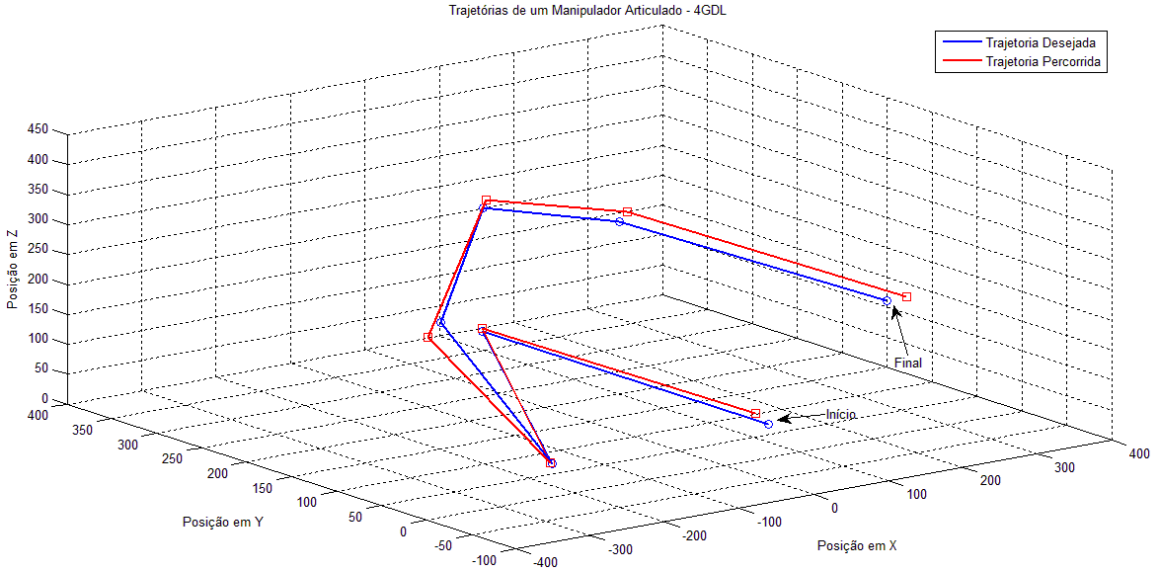
6.5 SIMULANDO UMA TRAJETÓRIA COM O ALGORITMO DA RNA, APLICADO AO MANIPULADOR

Com o objetivo de mostrar o desempenho e a precisão obtida pelo algoritmo da Rede Neural para resolver a cinemática inversa do manipulador robótico implementado, foram escolhidos sete pontos, (x, y e z), dentro do volume de trabalho treinado, de modo a formar uma trajetória. Estes pontos foram apresentados ao algoritmo aplicado ao manipulador como pontos desejados. A resposta do algoritmo gerou uma segunda trajetória por onde o manipulador supostamente posicionou a garra. Com esses dados, os desejados e os percorridos, obtiveram-se no MATLAB as trajetórias ilustradas na Figura 54.

Ressalta-se que a trajetória percorrida não foi adquirida pela posição do manipulador real, mas pelos valores dos ângulos de juntas obtidos pelo algoritmo na plataforma de controle que são utilizados para calcular, via cinemática direta, a suposta posição do manipulador. Portanto, alguns fatores que poderiam interferir no posicionamento final do manipulador real como, por exemplo, a presença de ruídos no sinal de controle dos atuadores, folgas nas juntas, dentre outros, não foram mensurados. Para a medição direta do posicionamento do manipulador real no volume de trabalho, algum outro instrumento deveria ser utilizado como, por exemplo, tratamento de imagem.

Na Tabela 18 apresentam-se os valores das coordenadas dos pontos utilizados e seus respectivos valores obtidos pela resposta do algoritmo e calculado via cinemática direta.

Figura 54 - Resposta do algoritmo *online* aos pontos solicitados.



Fonte: Elaborada pelo autor.

Tabela 18 - Validação dos pontos para teste *online*

Pontos desejados [mm]			**Pontos obtidos [mm]**			**Erro [mm]**		
x	**y**	**z**	**x**	**y**	**z**	**x**	**y**	**z**
300,6	68,2	173,1	303,3	48,1	189,6	2,7	20,1	16,5
141,0	233,0	262,5	137,4	222,0	285,2	3,6	11,0	22,7
-94,3	190,5	359,0	-103,1	179,3	379,8	8,8	11,2	20,8
-258,8	101,2	249,5	-261,3	113,2	219,4	2,5	11,9	30,1
-249,0	-15,8	68,2	-253,1	-16,2	69,8	4,1	0,4	1,6
-41,8	235,0	119,9	-36,7	238,4	121,8	5,0	3,4	1,9
140,5	67,2	4,0	148,5	87,4	10,7	8,1	20,2	6,7

Fonte: Elaborada pelo autor.

6.6 COMENTÁRIOS

Neste capítulo foram apresentados o protótipo e a montagem dos componentes do sistema, incluindo a plataforma de desenvolvimento para a aplicação do algoritmo da RNA, MLP configuradas em paralelo para uma trajetória.

CAPÍTULO 7. CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

Este trabalho teve como objetivo apresentar uma alternativa para resolver a cinemática inversa utilizando Redes Neurais Artificiais do tipo MLP, tendo como referência um protótipo de um manipulador robótico de 5GDL, composto por sete servomotores controlado pela plataforma Intel® Galileo Gen 2.

Com as soluções da cinemática inversa pode-se determinar o valor de cada articulação, a fim de colocar o robô manipulador na posição e orientação desejadas. Desta forma, as RNAs são capazes de entender a relação cinemática entre o sistema de coordenadas das juntas e a posição final da ferramenta do manipulador, em razão da sua flexibilidade e capacidade de aprendizado por meio do treinamento.

Após o treinamento, a utilização de RNAs em sistemas embarcados se torna vantajosa pois, a etapa de propagação dos padrões na fase *online* exige menor custo computacional do que na etapa de treinamento, viabilizando aplicações deste algoritmo em plataformas com poder de processamento intermediário.

Visando obter bons resultados na implementação das Redes Neurais, diversos parâmetros devem ser considerados, como por exemplo, o tipo de RNA utilizada, o valor da taxa de treinamento e do momento, entre outros, cujos valores em sua maioria são determinados experimentalmente, por não haver um consenso sobre os melhores valores.

Para determinar os padrões de treinamento apresentados às RNAs foram considerados quatro, dos cinco graus de liberdade do manipulador em referência, devido não somente ao interesse no posicionamento da ferramenta terminal do manipulador, mas também para diminuir a complexidade do algoritmo.

Os padrões de treinamentos utilizados têm origem em duas abordagens distintas, uma por trajetórias específicas e outra, por um conjunto de pontos distribuídos no espaço de trabalho. Inerente a qualquer RNA, o treinamento é feito *offline*, devido ao alto custo computacional.

Foram implementadas duas arquiteturas de RNAs, uma RNA simples e um conjunto de nove RNAs configuradas em paralelo, ambas são compostas por RNAs de mesmas características. As RNAs configuradas em paralelo são discretizadas em seu volume de trabalho em relação à coordenada z do plano cartesiano. Os resultados obtidos para esta solução foram melhores, quando comparada a uma única RNA para resolver todo o volume.

O algoritmo composto por uma RNA simples, quando aplicado ao aprendizado de trajetórias, resultou em baixos valores de erros médios. Entretanto, no treinamento pelo conjunto de pontos distribuídos no volume de trabalho, obteve um menor rendimento no aprendizado da relação cinemática do manipulador pois, apresentou um comportamento de saturação do conhecimento.

A proposta do algoritmo composto por nove RNAs configuradas em paralelo apresentou menores erros médios das juntas, tanto para o aprendizado da trajetória quanto para o conjunto de pontos distribuídos no volume de trabalho. Outra vantagem desta proposta é a possibilidade de identificar qual RNA não apresenta solução satisfatória e atuar na mesma de forma a otimizar os seus resultados.

Portanto, a proposta de configurar RNAs em paralelo apresentada neste trabalho demonstrou melhor rendimento quando comparada à uma RNA simples e pode representar uma boa estratégia para futuros trabalhos no mesmo domínio, independente dos graus de liberdade do manipulador.

Outra alternativa para futuros trabalhos é discretizar de outra forma, o volume de trabalho, por exemplo, Redes Neurais especializadas em relação à coordenada z e por quadrante ou ainda, utilizar algoritmos inteligentes, como Algoritmo Genético ou Lógica Fuzzy para otimizar os parâmetros de cada RNA, melhorando a convergência e o tempo de processamento.

REFERÊNCIAS

ALAVANDAR, S.; NIGAM, M. J. Neuro-Fuzzy based Approach for Inverse Kinematics Solution of Industrial Robot Manipulators. **Int. J. of Computers, Communications & Control**, Oradea, v. 3, n. 3, p. 224–234, 2008.

ARDUINO. **Arduino**. [S. l.: s. n.], 2016. Disponível em: <https://www.arduino.cc/>. Acesso em: 10 abr. 2016.

BARRETO, G. D. A. et al. Implementação de um sistema de controle para o robô puma 560 usando uma rede neural auto-organizável. **Sba: Controle & Automação**, Heidelberg, v. 13, n. 2, p. 141–155, 2002.

BEAGLEBOARD. **Beagleboard:** BeagleBoneBlack. [S. l.], 2015.

CAMARGO, J. T. F. de; VERASZTO, E. V.; BARRETO, G. **Simulação do modelo cinemático inverso de um robô através do uso de redes neurais artificiais:** um complemento ao ensino de robótica. [S. l.]: COBENGE, 2014.

CHIDDARWAR, S. S.; BABU, N. R. Comparison of RBF and MLP neural networks to solve inverse kinematic problem for 6R serial robot by a fusion approach. **Engineering Applications of Artificial Intelligence**, Kidlington, v. 23, n. 7, p. 1083–1092, 2010.

CRAIG, J. J. **Robótica**. 3. ed. São Paulo: Pearson, 2012.

DA SILVA, I. N.; SPATTI, D. H.; FLAUZINO, R. A. **Redes neurais artificiais:** para engenharia e ciências aplicadas curso prático. São Paulo: Artliber, 2010.

DÁVILA, V. M. H.; READ, J. S. B. Cinemática inversa de un manipulador robótico con redes neuronales. **Encuentro de Investigacion en Ingeniería Elétrica**, p. 5, 2004.

DAYA, B.; KHAWANDI, S.; AKOUM, M. Applying Neural Network Architecture for Inverse Kinematics Problem in Robotics. **Journal of Software Engineering and Applications**, Irvine, v. 03, n. 03, p. 230–239, 2010.

FENG, Y.; YAO-NAN, W.; YI-MIN, Y. Inverse Kinematics Solution for Robot Manipulator based on Neural Network under Joint Subspace. **International Journal of Computers Communications & Control**, Oradea, v. 7, n. 3, p. 459–472, 2012.

FERNANDES JUNIOR, F. E. **Estudo e implementação de redes neurais e algoritmos genéticos para resolução de cinemática inversa de um manipulador robótico com 5 graus de liberdade**. Campinas: Universidade Estadual de Campinas, 2014.

HASAN, A. T. et al. An adaptive-learning algorithm to solve the inverse kinematics problem of a 6 D.O.F serial robot manipulator. **Advances in Engineering Software**, Kidlington, v. 37, n. 7, p. 432–438, 2006.

HAYKIN, S. S. **Redes neurais:** princípios e prática. 2. ed. Porto Alegre: Bookman Companhia, 2001.

INTEL CORPORATION. **Intel galileo gen 2 development board.** [S. l.], 2015.

JHA, P.; BISWAL, B. B.; SAHU, O. P. S. Inverse kinematic solution of robot manipulator using hybrid neural network. **International Journal of Materials Science and Engineering**, Singapore, v. 3, n. 1, p. 31-38, 2014.

LIMA, T. L. DE V. **Controlador neural com camada oculta definida por meio de algoritmo genético aplicado ao posicionamento de um manipulador robótico**. João Pessoa: Universidade Federal da Paraíba, 2012.

MINUSSI, C. R.; LOTUFO, A. D. P. **Redes neurais:** introdução e principais conceitos. Ilha Solteira: [s. n.], 2008.

MORRIS, A. S.; MANSOR, A. Finding the inverse kinematics of manipulator arm using artificial neural network with lookup table. **Robotica**, Cambridge, v. 15, n. 6, p. 617–625, 1997.

NIKU, S. **Introduction to robotics**. New Jersey: John Wiley & Sons, 2010.

NÓBREGA SOBRINHO, C. A. **Controlador neural aplicado a um sistema posicionador acionado por motores de indução trifásicos**. João Pessoa: Universidade Federal da Paraíba, 2011.

PINHEIRO, T. C. F.; DA TRINDADE, M. R. P.; PANTOJA, B. R. Cinemática inversa de um manipulado robótico de quatro graus de liberda- de utilizando método numérico iterativo da jacobiana pseudo-inversa. In: SBAI, 11., 2013, Fortaleza. **Anais…** Fortaleza: SBAI, 2013. p. 1–5.

RAJ, D. R.; RAGLEND, I. J.; ANAND, M. Inverse kinematics solution of a five joint robot using feed forward and radial basis function neural network. In: INTERNATIONAL CONFERENCE ON COMPUTATION OF POWER, ENERGY, INFORMATION AND COMMUNICATION, 2015, [S. l.]. **Proceedings…** Piscataway: IEEE, 2015. p. 117-122.

RAMON, M. C. **Intel ® Galileo and Intel ® Galileo Gen 2 -API Features and Arduino Projects for Linux Programers**. New York: Apress Media, 2014.

RASPBERRY PI FOUNDATION. **Raspbarry Pi**. [S. l.: s. n.], 2016. Disponível em: <https://www.raspberrypi.org/>. Acesso em: 9 abr. 2016.

RODRIGUES, P. T.; MANTOVANI, S. C. A. **Abordagem cinemática para o controle de um manipulador robótico de 3gl usando raspberry pi**. Ilha Solteira: UNESP, 2015.

ROSÁRIO, J. M. **Princípios de mecatrônica**. São Paulo: Pearson Prentice Hall, 2005.

APÊNDICE A - ALGORITMO CONSIDERANDO UMA RNA

1. Entradas de dados (número de padrões de treinamento; conjunto de entradas e suas respectivas respostas desejadas);

2. Iniciar a RNA (número de camadas, número de neurônios em cada camada; função de ativação e seus parâmetros; inicialização das matrizes de pesos com valores randômicos normalizados entre –1 e +1);

3. Verificar se o critério de parada foi satisfeito, se não, iniciar o treinamento de rede - algoritmo de BP (iniciar o contador de épocas igual a zero, definir os valores da taxa de treinamento e do termo momento) enquanto o critério de parada não for atendido, prosseguir:

3.1. Para cada conjunto de treinamento até que todos sejam apresentados:

3.1.1. Propagar os valores das entradas, camada por camada até a camada de saída por meio das equações (39) e (40) e calcular o conjunto do erro de cada neurônio da camada de saída pela equação (42);

3.1.2. Por meio da equação (44) calcular o gradiente local de cada neurônio na camada saída por intermédio do seu erro, equação (42);

3.1.3. Ajustar os pesos da última camada por meio da equação (46);

3.1.4. Calcular o gradiente local da segunda camada intermediária por meio da equação (45);

3.1.5. Ajustar os pesos da segunda camada intermediária pela equação (46);

3.1.6. Calcular o gradiente local da primeira camada intermediária por meio da equação (45);

3.1.7. Ajustar os pesos da primeira camada intermediária por meio da equação (46);

3.2. Propagar todo o conjunto de treinamento pelas equações (39) e (40) e atualizar o erro quadrático médio por meio das equações (42), (47) e (48);

3.3. Incrementar em uma unidade o valor do contador de épocas;

3.4. Comparar se o atual valor de erro quadrático médio é o menor de todo o treinamento, se for:

3.4.1. Erro médio quadrático atual → variável de menor erro. E atualizar a época do menor erro;

3.4.2. Armazenar o conjunto de pesos sinápticos desta época.

4. Início da fase de operação.

4.1. Propagar os valores das entradas, camada por camada até a camada de saída, por meio das equações (39) e (40); disponibilizar as saídas da rede para todo o conjunto de amostras, utilizando os pesos sinápticos armazenados no item **3.4.2** deste algoritmo.

5. Exportação dos pesos sinápticos por meio de arquivos de dados.

APÊNDICE B - ALGORITMO DAS RNAS CONFIGURADAS EM PARALELO

1. Entradas de dados (número de padrões de treinamento; conjunto de entradas e suas respectivas respostas desejadas);

2. Definir parâmetros das MLPs (número de neurônios, taxa de aprendizado, fator momento, função de ativação e pesos sinápticos iniciais);

3. Verificar se o critério de parada foi satisfeito, se não, iniciar o treinamento de rede - algoritmo de BP (iniciar o contador de épocas igual a zero, definir os valores da taxa de treinamento e do termo momento) enquanto o critério de parada não for atendido, prosseguir:

3.1. Para cada padrão de treinamento, até que todos sejam apresentados:

3.1.1. Desnormalizar a entrada referente à coordenada z e utilizar a lógica de direcionamento de padrões para enviar à MLP específica, os dados de treinamento: parâmetros da MLP (item 2) e o conjunto de entradas e saídas;

3.1.2. Na RNA: Propagar e retropropagar o padrão de treinamento e devolver o conjunto de pesos sinápticos após o treino;

3.2. Atualização dos erros quadráticos médios: propagar todos os padrões de entrada para a saída, utilizando a lógica de direcionamento de padrões;

3.3. Incrementar o número de épocas;

3.4. Se o erro quadrático médio atual de cada MLP for menor do que o menor registrado:

3.4.1. Armazenar o conjunto de pesos sinápticos e atualizar o menor erro quadrático médio registrado;

4. Fase de operação: propagar todos os padrões de entrada para a saída, utilizando a lógica de direcionamento de padrões;

5. Exportação dos pesos sinápticos por meio de arquivos de dados.

APÊNDICE C - CARACTERÍSTICAS CONTRUTIVAS DO PROTOTIPO

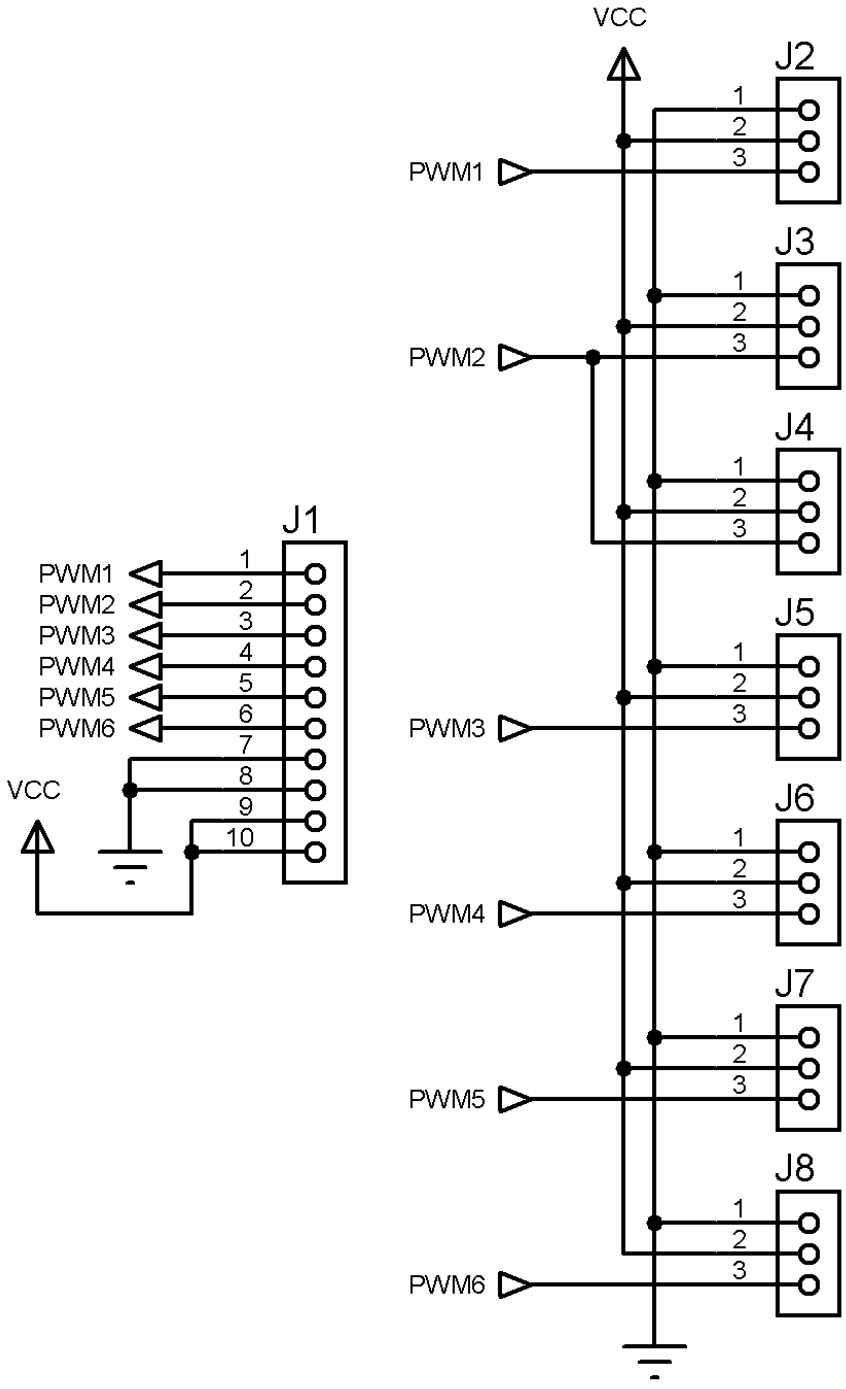
Tabela 19 – Características Construtivas do Prototipo

Número de eixos	5	
Alimentação	5 VCC/3A	
Alcance máximo	Com a garra	Sem a garra
	466 mm	310 mm
Peso	2641 g	
Base (Diâmetro)	150 mm	
Capacidade de carga	300g (na garra)	
Movimento dos eixos	**Eixo**	**Faixa de trabalho**
	1	180°
	2	90°
	3	45°
	4	90°
	5	180°

Fonte: Elaboração do autor.

APÊNDICE D - PLACA DE INTERFACE DOS TERMINAIS DOS SERVOMOTORES

Figura 55 - Placa de interface dos terminais dos servomotores



Fonte: Elaboração do autor.