UNIVERSIDADE CANDIDO MENDES – UCAM
PROGRAMA DE PÓS-GRADUAÇÃO EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL
CURSO DE MESTRADO EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL

Edson Simões Dos Santos

PROJETO E CONSTRUÇÃO DE UM CONTROLADOR ADAPTATIVO POR REALIMENTAÇÃO DE ESTADOS DE UM PÊNDULO INVERTIDO UTILIZANDO INTELIGÊNCIA COMPUTACIONAL

CAMPOS DOS GOYTACAZES, RJ
Novembro de 2013

UNIVERSIDADE CANDIDO MENDES - UCAM
PROGRAMA DE PÓS-GRADUAÇÃO EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL
CURSO DE MESTRADO EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL

Edson Simões Dos Santos

PROJETO E CONSTRUÇÃO DE UM CONTROLADOR ADAPTATIVO POR REALIMENTAÇÃO DE ESTADOS DE UM PÊNDULO INVERTIDO UTILIZANDO INTELIGÊNCIA COMPUTACIONAL

Dissertação apresentada ao Programa de Pós-Graduação em Pesquisa Operacional e Inteligência Computacional, da Universidade Candido Mendes – Campos dos Goytacazes/RJ, para obtenção do grau de MESTRE EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL.

Orientador: Prof. Ítalo de Oliveira Matias, D.Sc.

CAMPOS DOS GOYTACAZES, RJ
Novembro de 2013

EDSON SIMÕES DOS SANTOS

PROJETO E CONSTRUÇÃO DE UM CONTROLE ADAPTATIVO POR REALIMENTAÇÃO DE ESTADOS DE UM PÊNDULO INVERTIDO UTILIZANDO INTELIGÊNCIA COMPUTACIONAL

Dissertação apresentada ao Programa de Pós-Graduação em Pesquisa Operacional e Inteligência Computacional, da Universidade Candido Mendes – Campos dos Goytacazes/RJ, para obtenção do grau de MESTRE EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL.

Aprovada em 29 de novembro de 2013.

Orientador: Prof. Ítalo de Oliveira Matias, D.Sc.

BANCA EXAMINADORA

_____

Prof. Ítalo de Oliveira Matias, D.Sc.- orientador
Universidade Candido Mendes

_____

Prof. Eduardo Shimoda, D.Sc.
Universidade Candido Mendes

_____

Prof. Adelson Siqueira Carvalho, D.Sc.
Instituto Federal Fluminense

CAMPOS DOS GOYTACAZES / RJ
NOVEMBRO 2013

AGRADECIMENTOS

Agradeço a Deus pela sua infinita misericórdia.

Ao meu orientador, professor Ítalo de Oliveira Matias, pela compreensão e dedicação que foram fundamentais na concretização deste trabalho.

A minha querida namorada Lunara pelo carinho e apoio incondicional.

Aos alunos Fábio Barradas, Camila Lins, Elias Barreto e Ana Laura Lisboa, pela contribuição dada durante algumas fases no desenvolvimento deste projeto.

Aos meus amigos por me incentivar e pela força dada nos momentos de dificuldade.

Ao Instituto Federal de Educação, Ciência e Tecnologia Fluminense pela disponibilização de seus laboratórios.

“Saber muito não lhe torna inteligente. A inteligência se traduz na forma que você recolhe, julga, maneja e, sobretudo, onde e como aplica esta informação.”

Carl Sagan

RESUMO

PROJETO E CONSTRUÇÃO DE UM CONTROLE ADAPTATIVO POR REALIMENTAÇÃO DE ESTADOS DE UM PÊNDULO INVERTIDO UTILIZANDO INTELIGÊNCIA COMPUTACIONAL

Algoritmos de otimização baseados em métodos heurísticos têm sido largamente utilizados na sintonia automática de controladores, principalmente quando uma solução analítica não se apresenta viável em relação ao custo e tempo de obtenção de uma solução satisfatória. Também se torna crescente o emprego da Inteligência Computacional (IC) em demanda da complexidade dos sistemas e variações das condições de operação, exigindo cada vez mais sistemas de controle robustos e adaptativos a ambientes dinâmicos. O objetivo deste estudo é projetar um sistema de controle adaptativo utilizando inteligência computacional, em conjunto com um sistema capaz de alto-configurar a Rede Neural Artificial (RNA) utilizada no controlador, atendendo as restrições estipuladas no projeto. No presente trabalho aborda-se projeto e construção de um controlador adaptativo por ganho agendado, implementado via RNA. O projeto apresenta a obtenção e a representação do modelo não linear do pêndulo invertido em Espaço de Estados. O modelo do sistema é classificado como variante no tempo devido à variação de massa, e exige um controle adaptativo devido às variações dos parâmetros de operação, como tempo de acomodação e sobressinal. A estratégia de controle adotada é o controle por realimentação de estados, caracterizado pela alocação dos polos de malha fechada no plano complexo. Um Algoritmo Genético (AG) é responsável pela alocação dos polos, obtidos a partir de uma busca num espaço de solução não linear. Este algoritmo também é responsável pela configuração automática da RNA. O algoritmo evolutivo utilizado (AG) manipula a quantidade de camadas ocultas, número de neurônios das camadas e funções de ativação para obter uma melhoria na arquitetura da rede utilizada no problema proposto, sem que um conhecimento especialista do sistema seja exigido. Estratégia de *crossover*, mutação e elitismo, são utilizadas na evolução da população do AG. As operações de otimização neste trabalho ocorrem num espaço de busca limitado pelo autor. Após os testes, um simulador é construído com animação em 2D, permitindo ao usuário interagir com o modelo e acompanhar a resposta deste em tempo real. O sistema de controle adaptativo desenvolvido e implementado computacionalmente no presente trabalho atendeu as condições de variação de massa, sobressinal e tempo de acomodação, atuando dentro da faixa de operabilidade estabelecida em projeto. O método de configuração automática da RNA por AG obteve uma configuração de rede, e retornou o menor erro após o treinamento num determinado conjunto de soluções.

PALAVRAS-CHAVE: Controle Adaptativo, Algoritmo Genético, Inteligência Computacional, Redes Neurais Artificiais.

ABSTRACT

DESIGN AND CONSTRUCTION OF ADAPTIVE CONTROL BY STATE FEEDBACK OF AN INVERTED PENDULUM USING COMPUTATIONAL INTELLIGENCE

Optimization algorithms based on heuristics have been extensive used in automatic tuning of controllers, especially when an analytical solution is not available relation to cost and time of obtaining a satisfactory solution. The use of Computational intelligence (CI) increases by the demand system complexity and variations of the operating conditions, requiring increasingly robust control and adaptive in dynamic environment. This bacharol1s work aims to show the design and development of an adaptive controller for gain scheduled, implemented via Artificial Neural Network (ANN). The objective of this study is to design an adaptive control system using computational intelligence, together with a system capable of high-configure the Artificial Neural Network (ANN) used in the control, according the restrictions stipulated in the project .The project presents a nonlinear model of the inverted pendulum in State Space. The system model is classified as time-varying due to mass variation and is adaptive control due to the variation of operating parameters such as settling time and overshoot. The adopted control strategy is the state feedback control, characterized by the allocation of the closed-loop poles in the complex plane. A Genetic Algorithm (GA) is responsible for allocating the poles obtained from a search in the space of nonlinear solution. This algorithm is also responsible for the automatic configuration of the ANN. The evolutionary algorithm used (GA) manipulates the amount of hidden layers , number of layers of neurons and transfer functions for an improved network architecture used in the proposed problem without the necessity of great knowledge about the system. Crossover, mutation and elitism strategies are used in the GA population evolution. Operations for optimization are a search in a limited space by the author. After the tests, a simulator is developed with 2D animation, allowing the user to interact with the model and monitor this response in real time. The adaptive control system developed and implemented computationally in this work meets the conditions of mass variation, overshoot and settling time, acting within the range of operability established in the project. The method auto-configuration of ANN by GA obtained a network configuration that returned the smallest error after training in a set of particular solutions.

KEYWORDS: Adaptive Control, Genetic Algorithm, Computational Intelligence, Artificial Neural Networks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACO	Colony Optimization Algorithm
AE	Algoritmos evolutivos
AG	Algoritmo Genético
AI	Artificial Intelligence
ANN	Artificial Neural Network
EP	Evolutionary Programming
EPNet	Evolutionary Programming Network
GA	Genetic Algorithm
GPS	General Problem Solver
IA	Inteligência Artificial
MIMO	Multiple Input Multiple Output
MLP	Multilayer Perceptron
PID	Proporcional Integrativo e Derivativo
RNA	Rede Neural Artificial
SIMO	Single Input Multiple Output
SISO	Single Input Single Output
IC	Inteligência Computacional
CI	Computational intelligence
CE	computação evolutiva

SUMÁRIO

INTRODUÇÃO

O pêndulo invertido é um sistema didático muito utilizado na área de controle devido as suas características de instabilidade. A análise deste sistema possibilita trabalhar desde as teorias de modelagem para sistemas não lineares, linearização, aplicação de controle convencional quando linearizado e controle moderno.

Este sistema pode ser comparado ao controle de posição de um foguete na fase de lançamento, em que o objetivo é manter o foguete na posição vertical (OGATA, 2010). A linearização do modelo para este caso pode ser aplicado com eficiência devido ao foguete partir próximo da posição de equilíbrio e sofrer pequenos distúrbios. Após o lançamento, a massa do foguete diminui com a queima do combustível e com o desacoplamento dos propulsores, passando a se comportar diferentemente do momento em que se encontrava na base lançamento, ficando sujeito a distúrbios maiores. Nesta situação um modelo não linear e variante no tempo é que melhor descreve o sistema, contudo exige um controle mais sofisticado, bem como metodologia e teorias mais complexas.

As teorias de controle e estatísticas por algum tempo apresentaram certas limitações que motivaram o surgimento e crescimento da área de Inteligência Artificial (IA). Esta que inicialmente surgiu como uma rebelião a tais limitações, agora inclui estes campos em sua área de pesquisa (MCALLESTER, 1998, apud RUSSEL, 2003).

Segundo Russell (2003), nos últimos anos, uma revolução aconteceu na área de IA, no que tange o seu conteúdo e sua metodologia, sendo mais comum usar as teorias existentes, em vez de propor uma nova, fundamentando suas bases através de evidências experimentais rígidas e destacando a relevância nas aplicações em

sistemas reais. Estes sistemas podem ser distribuídos em diversas áreas. Atualmente a IA apresenta firmes métodos científicos. Cabe lembrar que não basta simplesmente aplicar tais metodologias, pois uma hipótese para ser aceita deve ser submetida a rigorosos testes empíricos, seus resultados devem ser acompanhados de analises estatísticos de acordo com sua relevância (COHEN, 1995, apud RUSSEL, 2003).

O estudo na área de controle moderno aplicado a sistemas não lineares tem sido utilizado em inúmeras áreas, possibilitando aplicações de soluções diversas, de maneira a contornar os problemas encontrados no controle convencional, que exige linearização de modelos, limitados a sistemas monovariáveis, e por não possuírem bom desempenho quando aplicado a sistemas com grandes variações de parâmetros ou condições de operação.

Para este tipo de sistema o controle adaptativo é recomendado para atender as restrições de projeto. Segundo Bolton (2010), o controle adaptativo é utilizado em situações de controle em que os parâmetros da planta variam com o tempo ou, talvez, com a carga. Assim, com a mudança da função de transferência da planta, é necessária uma nova sintonia do sistema de controle para que as condições de otimização sejam atendidas.

Segundo Rekdalsbakken (2006), vários modelos físicos de pêndulo invertido têm sido construídos e muitas estratégias de controle têm sido exploradas. Recentemente, o foco na construção de sistemas de controle tem sido a inteligência artificial (IA), métodos como a lógica *Fuzzy*, Redes Neurais Artificiais (RNA's) e Algoritmos Genéticos (AG's). Combinações destes métodos são também muito relevantes na área da computação evolutiva (CE).

A Figura 1 apresenta um sistema pêndulo invertido num base de carro.

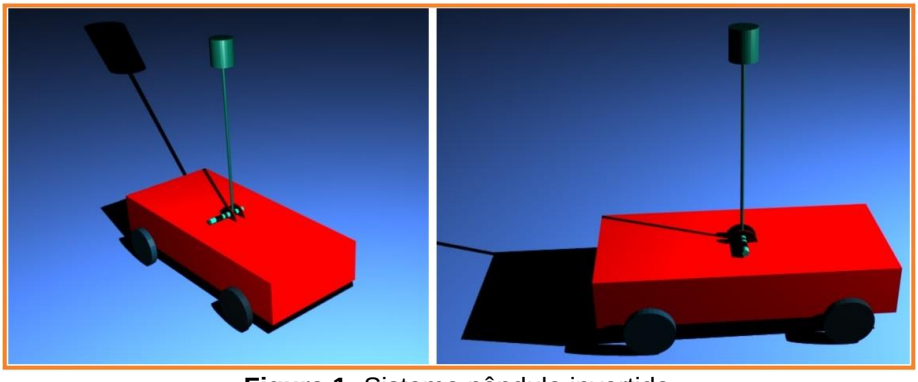


Figura 1: Sistema pêndulo invertido.

Fonte: Lins e De Souza, 2013. Adaptada pelo autor.

A computação evolutiva é uma área da ciência da computação que utiliza as ideias de evolução biológica para resolver problemas computacionais. Muitos destes problemas exigem uma procura num espaço enorme de possibilidades de soluções, tais como um vasto número de possíveis arranjos de circuitos de hardware, para uma configuração que produz comportamento desejado por um conjunto de equações, que ditará o comportamento de um sistema como um conjunto de regras a controlar um robô, fazendo com que este navegue num ambiente. Tais problemas computacionais muitas vezes requerem um sistema de controle adaptativo, permitindo manter um bom desempenho em um ambiente em mudança. Problemas como estes exigem soluções complexas que geralmente são difíceis para programadores humanos conceber. A rede neural artificial (RNA) é um exemplo desta filosofia evolutiva da computação (MITCHELL, 1999). Embora as RNA's apresentem bom desempenho na resolução destes problemas, ainda existe uma carência no quesito metodologia ou regras exatas para obter a melhor configuração da rede. Em sua maioria, a configuração se caracteriza mais como arte do que ciência, pois demanda uma considerável experiência de cada projetista.

Em alguns casos, os métodos exatos tendem a ser inadequados para problemas de elevada complexidade, pois geralmente possuem alto custo para sua obtenção e podem não estabelecer uma solução de boa qualidade dentro de um tempo razoável. Em contrapartida, os métodos heurísticos, apesar de não garantirem a melhor solução, costumam pelo menos obter soluções de boa qualidade em um tempo razoável (LAFETÁ, 2010).

Estudos na área de IA mostraram que os exemplos naturais de aprendizagem e adaptação são tesouros de procedimentos e estruturas para construção de algoritmos de busca robustos. Estes são procedimentos de busca probabilísticos projetados para trabalhar em grandes espaços. Destaque entre eles para o Algoritmo Genético (GOLBERG, 1988).

No presente trabalho o sistema pêndulo invertido será representado num carro-pêndulo. Será simulada a variação da massa do modelo, que exigirá um controle adaptativo, por se tratar de um sistema variante no tempo. A escolha do modelo não linear simplificado, desprezando atrito e momento de inércia, se dá em função de possibilitar a simulação com maiores distúrbios, obtendo uma resposta diferente do modelo linearizado, e por não haver a intenção, neste estudo, de ser implementado num sistema físico para comparação de resultados.

A otimização feita pelo método heurístico AG foi escolhida por apresentar fácil implementação computacional, obter soluções viáveis com maior simplicidade, exigindo somente uma função de custo. A RNA como parte do sistema de controle adaptativo apresenta menor custo computacional quando comparado à programação convencional estruturada para checagem e resposta das condicionais na variação do sistema. Para contornar o problema enfrentado na configuração da Rede Neural, um AG será utilizado como mecanismo de configuração e otimização, determinando a quantidade de camadas ocultas, suas funções de ativação, e também o número de neurônios nas camadas ocultas. Um AG também será utilizado para escolher a localização dos novos polos da equação característica, responsável por gerar o vetor de controle.

1.1 OBJETIVO

1.1.1 Objetivo Geral

Projetar e construir um controlador adaptativo para sistema pêndulo invertido de massa variável que apresenta variação nas condições de operação, atendendo os requisitos de sobressinal e tempo de acomodação estabelecidos no projeto, em conjunto será aplicado um método capaz de autoconfigurar a arquitetura de uma RNA para ser utilizada como controlador adaptativo.

1.1.2 Objetivos Específicos

Implementar um algoritmo de otimização (AG) para alocação dos polos da equação característica do sistema de controle adaptativo, por realimentação no espaço de estados.

Implementar AG para otimizar a configuração (funções de ativação, número de neurônios da camada oculta, quantidade de camadas ocultas) da RNA que será utilizado como base do controle agendado.

Prover animação num ambiente em 2D, que permita a visualização da

dinâmica do sistema em tempo real, e interface que permita ao usuário a inserção de distúrbio no sistema.

Analisar o comportamento do sistema de otimização *off-line* em relação às estratégias aplicadas.

Analisar o comportamento da RNA como base no controle agendado, destacar as avaliações positivas ou negativas encontradas durante a simulação.

Verificar se o controlador adaptativo consegue atender os requisitos estipulados no projeto.

1.2 JUSTIFICATIVA

Estudos sobre controle moderno têm sido realizados utilizando Inteligência Computacional. Investe-se cada vez mais em sistemas autônomos e com grande capacidade de adaptação em ambientes dinâmicos, sujeitos a grandes variações de operação. Uma busca constante é realizada na aplicação das metodologias já existentes em aplicações nas mais diversificadas áreas do mundo real, determinando sua viabilidade e desempenho. A área de controle moderno é um excelente laboratório para testes, devido a sua ampla aplicação nas diversas ferramentas e objetos, nos mais variados ambientes, alguns exemplos são encontrados na Figura 2.

Figura 2: a) Brinquedos bípedes; b) Veículo de um eixo; c) Exoesqueleto.
Fonte: a) http://www.informacaovirtual.com/ **b)** http://www.segway.com/
c) http://www.targethd.net/tecnologia

Na Figura 2.a tem-se como exemplo um robô de brinquedo bípede que exige um sistema de controle para seu equilíbrio. Na Figura 2.b um veículo de um eixo que requer um controle robusto para ter agilidade e proporcionar suavidade em sua condução e na Figura 2.c um exoesqueleto utilizado para propiciar equilíbrio ao deficiente físico, auxiliando-o em sua locomoção, substituindo a utilização de uma cadeira de rodas. Assim nos três exemplos de áreas distintas pode-se identificar as possibilidades de aplicação das teorias de controle moderno.

1.3 ABORDAGEM DO TEMA NA PESQUISA

Será apresentada uma aplicação específica para simulação computacional, realizando um estudo dirigido na implementação de um sistema de controle adaptativo, em que serão abordadas as metodologias de modelagem de sistemas físicos, simulação, otimização utilizando técnicas evolutivas e aplicação de sistemas inteligentes no reconhecimento de padrões de controle, mostrando a viabilidade de aplicação dessas metodologias no problema abordado.

1.4 ORGANIZAÇÃO E DISPOSIÇÃO DOS CAPÍTULOS

Capítulo 1: Introdução: Neste capítulo é apresentada uma contextualização dos temas abordados na área de controle, algoritmos evolutivos, inteligência artificial e Redes Neurais Artificiais. Expõem-se os objetivos, justificativas e a forma como o tema é organizado e abordado nos capítulos.

Capítulo 2: Pesquisa Temática. Neste capítulo é feito um breve levantamento histórico relacionado ao tema proposto na dissertação, sua evolução e metodologias utilizadas ao longo do tempo. Apresenta um estudo realizado sobre as publicações mais recentes dos temas abordados, como estes têm sido tratados academicamente, os métodos utilizados para obtenção de respostas para os problemas abordados e áreas de aplicações.

Capítulo 3: Revisão da Literatura. Neste capítulo faz-se uma breve introdução sobre IA, seu surgimento e sua evolução, áreas de atuação e definição do conceito IA. Aborda-se algoritmos evolutivos, algoritmos genéticos, sua metodologia, e uma exemplificação através de um pseudocódigo. Também introduz o conceito de RNA, sua arquitetura, seu funcionamento, comparação com o neurônio biológico, conceito de aprendizado, tipos de aprendizado e algoritmos de treinamento. Na teoria de controle moderno apresenta-se a sua representação e controle no espaço de estados, controle por realimentação de estados, método para alocação dos polos de malha fechada, controle adaptativo e ganho programado. Finaliza este capítulo com a conceituação de modelos, classifica alguns destes quanto ao seu tipo, define a ideia de simulação e apresenta de forma resumida a plataforma de simulação utilizada neste trabalho. Para o leitor familiarizado com o tema, a leitura deste capítulo torna-se dispensável.

Capítulo 4: Desenvolvimento. Neste capítulo é apresentado o modelo não linear do pêndulo invertido, a forma que se concebeu o projeto do AG, do ganho agendado com RNA e do controlador adaptativo. Apresenta alguns passos para a construção do simulador e identifica as estratégias utilizadas. Relata as configurações utilizadas na simulação para as diferentes metodologias.

Capítulo 5: Resultados e discussões. Neste capítulo são apresentados, através de gráficos, tabelas e quadros, os resultados obtidos após a simulação do sistema.

Capítulo 6: Considerações finais. Este capítulo apresenta a conclusão e sugestões para trabalhos futuros.

Conclusão. Neste utiliza-se os resultados apresentados para fazer uma análise e responder se os objetivos estipulados na apresentação deste trabalho foram alcançados e o quanto se alcançou.

Sugestões para trabalhos futuros. Neste são apresentadas algumas sugestões pra trabalhos futuros, de modo a aproveitar os dados gerados e a metodologia aplicada no desenvolvimento deste trabalho.

Referências Bibliográficas. Lista os trabalhos que serviram como base para o desenvolvimento desta pesquisa.

2 PESQUISA TEMÁTICA

2.1 ANÁLISE BIBLIOMÉTRICA

A seguir é apresentada uma breve análise bibliométrica sobre o tema abordado neste trabalho. Ressalta-se que as informações apresentadas são referentes a uma base de pesquisa específica num determinado período. O tema abordado é controle adaptativo de sistemas não lineares utilizando algoritmos de otimização e inteligência artificial.

Para esta pesquisa, foi utilizada a base SCOPUS no dia 28/02/2013. O tema Simulação Computacional com Controle Adaptativo de Sistemas Dinâmicos (***computer simulation***) *AND* (***adaptive control***) *AND* (***dynamical systems***) foi introduzido na ferramenta de busca, retornando 5.666 publicações, com o trabalho mais antigo publicado em 1974 por Davidson, J.M., Durbin, L.D com o tema "*Model reference adaptive control specification for a steam heated finned tube heat exchanger"*. Este trabalho objetivava melhorar o desempenho dinâmico no controle de um sistema cuja função de transferência do modelo variava em relação ao tempo.

A distribuição destas publicações foi levantada em categorias como quantidade de publicações na área de pesquisa, países de publicação e idioma de publicação. Quantidade de publicações em relação a áreas de publicação, mostrado na Tabela 1.

Tabela 1: Quantidades de publicações por área.

Classificação	Área Temática	Quantidade
1	Engenharia	3611
2	Ciência da Computação	2206
3	Matemática	1133
4	Física e Astronomia	621332
5	Neuro ciências	243

Fonte: Tabela extraída da base SCOPUS.
Disponível em:< http://www.scopus.com/.>. **Acesso em:** fevereiro de 2013.

Países em relação à quantidade de publicações, representado na Tabela 2.

Tabela 2: Quantidades de publicações por país.

Classificação	País	Quantidade
1	Estados Unidos	1357
2	China	1332
3	Reino Unido	389
4	Taiwan	330
5	Japão	263
6	Alemanha	253
7	Canadá	249
8	França	229
9	Singapura	173
21	Brasil	70

Fonte: Tabela extraída da base SCOPUS.
Disponível em:< http://www.scopus.com/.>. **Acesso em:** fevereiro de 2013.

A quantidade de publicações por idioma é mostrada na Tabela 3.

Tabela 3: Quantidades de publicações por Língua de publicação.

Classificação	Língua	Quantidade
1	Inglês	5442
2	Chinês	213
3	Japonês	8
4	Português	4
5	Francês	2

Fonte: Tabela extraída da base SCOPUS.
Disponível em: <http://www.scopus.com/.>. **Acesso em:** fevereiro de 2013.

Na década de 90 o tema de pesquisa agregado a sistemas não lineares *AND* (***nonlinear***) começa a ser tratado com utilização de inteligência computacional. A busca retorna 4.262 publicações. A relação dos resultados encontrados por categorias são apresentadas nas tabelas 4, 5 e 6.

Tabela 4: Quantidades de publicações por área para sistemas não lineares.

Classificação	Área Temática	Quantidade
1	Engenharia	2817
2	Ciência da Computação	1679
3	Matemática	920
4	Física e Astronomia	211
5	Engenharia Química	243

Fonte: Tabela extraída da base SCOPUS.
Disponível em: <http://www.scopus.com/>**. Acesso em:** fevereiro de 2013.

Tabela 5: Quantidades de publicações por país para sistemas não lineares.

Classificação	País	Quantidade
1	China	1084
2	Estados Unidos	988
3	Taiwan	278
4	Reino Unido	273
5	Canadá	190
6	França	167
7	Japão	166
8	Alemanha	154
9	Singapura	143
20	Brasil	52

Fonte: Tabela extraída da base SCOPUS.
Disponível em:< http://www.scopus.com/.>. **Acesso em:** fevereiro de 2013.

Tabela 6: Quantidades de publicações por Língua de publicação para sistemas não lineares.

Classificação	Língua	Quantidade
1	Inglês	4097
2	Chinês	156
3	Japonês	5
4	Português	3
5	Francês	2

Fonte: Tabela extraída da base SCOPUS.
Disponível em:<http://www.scopus.com/>**. Acesso em:** fevereiro de 2013.

O trabalho "*Identification and control of dynamical systems using neural networks"* (1990) por Narendra, Kumpati S; Parthasarathy, Kannan aborda na simulação a eficácia da utilização das RNA's para identificação e controle de sistemas dinâmicos não lineares, mostrando-se viável como sistema de controle adaptativo. Este trabalho apresentou grande relevância, sendo citado em 3.396 artigos correlacionado ao tema como mostra o gráfico da Figura 3.

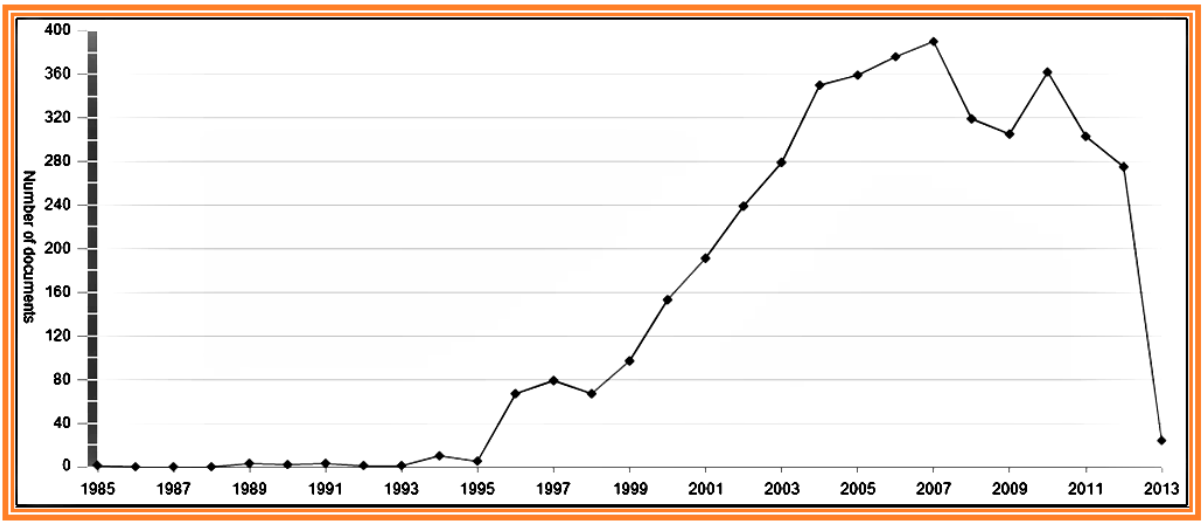


Figura 3: Citações do artigo *Identification and Control of Dynamical Systems using Neural Networks*.
Fonte: Gráfico extraída da base SCOPUS.
Disponível em: <http://www.scopus.com/>. **Acesso em:** fevereiro de 2013.

Dando continuidade na pesquisa no dia 01 / 03 / 2013, utilizando as mesmas palavras chaves, e adicionando a busca por Redes Neurais Artificiais *AND* (***neural network***), a busca retorna 2.300 publicações. O artigo "*Adaptive neural control of uncertain mimo nonlinear systems"* de Ge, S.S., Wang, C., reforça a eficácia da utilização das RNA's nos sistemas de controle adaptativo para sistemas de múltiplas entradas e múltiplas saídas (MIMO) com modelos não lineares. Publicado em 2004, este trabalho foi citado em 218 publicações. O Quadro 1 mostra as citações deste trabalho ao longo do tempo.

Quadro 1: Citação do artigo *Adaptive neural control of uncertain MIMO nonlinear systems*, ao longo dos anos.

Citações												
<2004	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	Subtotal	>2013	Total
0	3	15	16	24	30	33	37	32	23	5	**218**	0	218
	3	15	16	24	30	33	37	32	23	5	**218**		218

Fonte: Quadro extraído da base SCOPUS.
Disponível em:<http://www.scopus.com/.>. **Acesso em:** março de 2013.

Métodos híbridos também são muito utilizados para otimização destes sistemas. Um desses métodos que tem se apresentado com bastante eficiência, e se mostrado de fácil implementação, é a combinação das RNA's com AG (Algoritmo Genético). Ao refinar a busca com o algoritmo evolutivo *AND* (***genetic algorithm***), tem-se um retorno de 347 registros na base de busca. O artigo "*A hybrid neural-*

genetic multimodel parameter estimation algorithm", publicado em 1998, de Petridis, V., Paterakis, E., Kehagias, A, aborda este tema, implementando um algoritmo híbrido chamado de Neural-Genético para estimação de parâmetros de um manipulador robótico planar e uma estação de tratamento de água. Este Algoritmo híbrido também é utilizado na identificação de sistemas dinâmicos não lineares. A Rede Neural é utilizada para atribuir um valor de custo ou aptidão. O algoritmo Genético fica com a função de utilizar este custo aplicado a probabilidades de seleção para a produção de novas gerações de uma população. Este trabalho foi citado em 18 novas publicações.

Este trabalho apresentado como dissertação de mestrado na área de Inteligência Computacional e Pesquisa Operacional tem como principal objetivo a otimização e o controle adaptativo do pêndulo invertido, uma busca na base com a palavra chave *AND* (***inverted pendulum***), retornando 12 publicações, todas publicadas na língua inglesa.

A distribuição destes trabalhos em relação à área de publicação é dada na Tabela 7.

Tabela 7: Quantidades de publicações por área, para o sistema pêndulo invertido.

Classificação	Área Temática	Quantidade
1	Engenharia	9
2	Ciência da Computação	8
3	Matemática	3

Fonte: Tabela extraída da base SCOPUS.
Disponível em: <http://www.scopus.com/.>. **Acesso em:** março de 2013.

Relação de publicações do tema pesquisado em relação aos países de origem destas publicações e encontrado na Tabela 8.

Tabela 8: Quantidades de publicações por países, para o sistema pêndulo invertido.

Classificação	País	Quantidade
1	Itália	2
2	Coreia do Sul	2
3	Argélia	1
4	Canadá	1
5	China	1
6	Iran	1
7	Malásia	1
8	Paquistão	1
9	Taiwan	1
10	Turquia	1

Fonte: Tabela extraída da base SCOPUS.
Disponível em: <http://www.scopus.com/.>. **Acesso em:** março de 2013.

Os doze artigos retornados na pesquisa são apresentados no Anexo I:

Este foi o procedimento adotado na busca de textos científicos de referência ao tema abordado neste trabalho. O resumo dos principais artigos é abordado no Estado da Arte.

2.2 ESTADO DA ARTE

As informações apresentadas sobre o estado da arte está em ordem com a proximidade do tema desta trabalho, será apresentado na ordem do tema mais geral para o mais específico, ou seja para o mais proximo do tema desta dissertação.

Algoritmos de otimização têm sido largamente utilizados para sintonia automática de controladores. Os controladores Proporcional Integrativo e Derivativo (PID) classificados como controladores convencionais, são aplicados em sistemas lineares ou linearizáveis univariáveis com boa eficiência. Uma área de aplicação desta estratégia de controle é a área da robótica. Moghaddas (2012) aplica um AG na otimização da sintonia de um controlador PID em conjunto com equações de busca não lineares num pêndulo invertido de massa constante. No entanto, Dastranj et al. (2011), alerta que o método de projeto de controle linear é concebido com base na aplicação principal do amplo leque de frequência, mas em determinadas situações possui uma fraca aplicação por não conseguir compensar completamente o efeito não linear dos sistemas.

Em sistemas que possuem um maior range de operação, pode apresentar

comportamento não linear característico. Controladores utilizando métodos com Inteligência Artificiais como lógica nebulosa (Controlador *Fuzzy*) e RNA são aplicados com bom desempenho (BOGDANOV, 2004), pois além de possuírem ações de controle não linear, podem trabalhar com sistemas que apresentam respostas não lineares. Controles adaptativos também são aplicados para auto-regulagem utilizada em conjunto com algoritmos de otimização. O AG tem sido largamente utilizado como ferramenta de otimização em sistemas de controle por IA, pela facilidade de trabalhar em espaço de busca não linear. Um exemplo é o trabalho de Rekdalsbakken (2006), que utiliza o AG para atribuir valores de pesos e bias numa RNA utilizada como controlador num pêndulo invertido de massa constante. Mohamad et al. (2012), também aplica este método no controle de posição, num pêndulo virtual, também de massa constante, utilizando um controlador *Fuzzy*, sendo as funções de pertinência do controlador ajustadas pelo AG.

Sistemas como bípedes, também podem ser simplificados como um sistema pêndulo invertido para que seja projetado o seu sistema de controle. Ghorbani, Wu e Wang (2007), investigam a estabilidade de controle de um bípede, utilizando a simplificação anteriormente citada. O controle é implementado por uma Rede Neural que o estabiliza para uma posição próxima da vertical, e um controle PID faz o controle na região quando o erro tende a zero (região linearizável). O bípede tem massa constante, o AG utilizado tem a função de minimizar o fator custo de energia durante o processo.

Recentemente, pesquisas sobre rede neural tem recebido cada vez mais atenção, pelo potencial de aprender e reconstruir relações não lineares, e tem sido amplamente estudadas por pesquisadores da área de controle na indentificação, análise e projeto de sistemas de controle. O tamanho da rede, muitas vezes é medida pelo número de unidades de neurônios em camadas ocultas, que reflete na capacidade da RNA para aproximar uma função arbitrária. Surgindo assim a questão: qual é o tamanho necessário da rede neural para resolver um problema específico? Pesquisas têm sido realizadas para poder solucionar esta questão.

Yao (1997) utilizam um algoritmo de evolução denominado *EPNet* (*Evolutionary Programming Network*), para a evolução de Redes Neurais Artificiais. O algoritmo evolutivo utilizado na *EPNet* baseia-se na programação evolutiva de *Fogel's* (PE). O Algoritmo utilizado dá ênfase na mutação, evolui a arquitetura e

pesos das conexões. Para evitar o problema da permutação, o *crossover* não é adotado como estratégia, e sim a evolução dos indivíduos. A RNA utilizada é do tipo *feedforward* e a função de ativação é a função sigmóide.

Gutiérrez et al. (2005), utiliza um esquema de codificação construtivo indireto, com base na abordagem celular autônoma (*Cellular automata*), proposto para encontrar automaticamente uma arquitetura de RNA apropriado para um determinado problema. Um Algorítimo Genético é utilizado para geração de uma população e sua manutenção, atividades com crescimento celular e poda ocorrem durante a otimização. A função de ativação utilizada na RNA é uma função sigmoidal.

Rivero et al. (2008), utiliza um AG para configuração automática da RNA. Características como número de camadas ocultas, pesos e bias são alteradas para promover a evolução. Técnicas de *crossover* e mutação também são adotadas. A codificação utilizada é do tipo gráfica (*Graph Codification*). Este trabalho não menciona a função de ativação utilizada ou se esta sofre alteração.

Hassen (2012), sugere trabalhar com uma arquitetura de RNA fixa quadrada de tamanho significativo e suficiente para descrever a região de trabalho, utilizando como função de ativação uma função de base radial. Também propõe uma rede autoorganizável variando sua estrutura dinamicamente de modo a adicionar ou remover Funções Gaussianas de base radial, de modo a assegurar a precisão e aproximação desejada e ao mesmo tempo manter a complexidade de rede apropriada. O tamanho das redes é afetado pela distâcia entre os pontos necessários para descrever a função aproximada dentro de uma dada precisão.

No trabalho de Schuma (2013), o algoritmo de treinamento utilizado para essas redes é evolutivo. Uma população de RNA é gerada e mantida, uma função de aptidão para a aplicação específica é aplicada a cada rede na população. Redes com maior aptidão são preferencialmente selecionadas para a reprodução. A cada seleção, duas redes com relativamente alta aptidão são selecionadas e as operações de cruzamento e mutação são escolhidas com alguma probabilidade. As operações de cruzamento e de mutação não afetam apenas os pesos das sinapses, mas o número de neurônios e o número de sinapses. Não foi possível identificar neste trabalho a função de ativação utilizada.

A proposta do presente trabalho é acrescentar a variação de massa ao sistema, com a variação de requisitos de controle num modelo de simulação não

linear, numa faixa de operação, em conjunto com a construção de um controlador adaptativo. Para configuração automática da rede, adicionou como opção no algoritmo utilizado, quatro funções de ativação, além da opção de quantidade de camadas ocultas e números de neurônios nas camadas. Estas são as variações deste trabalho em relação às publicações apresentadas neste capítulo.

3 REVISÂO DE LITERATURA

3.1 INTELIGÊNCIA ARTIFICIAL

Segundo Luger (2004) definir Inteligência Artificial (IA) não é uma tarefa simples, começando na definição do conceito de inteligência em si, como: a inteligência é uma faculdade única ou apenas um nome dado a um conjunto de habilidades distintas e não correlacionadas? Qual o limite da aprendizagem sem um conhecimento prévio existente? Como a inteligência é representada no tecido de um ser vivo? O que rege o aprendizado? É possível definir inteligência a partir do comportamento observável ou é necessária à existência de uma forma de mecanismo interno particular? O que é necessário para projetar uma máquina inteligente? Tais perguntas levam a outras questões paradoxais num campo de estudo que inclui como objetivo sua própria definição, algo considerado apropriado por se tratar de uma área jovem com limites não claramente definidos como outros ramos mais antigos da ciência.

De acordo com Russell (2003), os primeiros trabalhos reconhecidos como IA foram realizados por Warren (1943), baseando-se em três fontes: conhecimento da fisiologia básica dos neurônios, análise formal da lógica proporcional criada por Russell e Whitehead, e a teoria da computação de Turing, em que um neurônio artificial responde ao estímulo emitido por um número suficiente de neurônios artificiais vizinhos, provocando uma mudança no seu estado. Allan Turing em seu trabalho publicado em 1950 com título *Computing Machinery Intelligence* foi o primeiro a apresentar uma visão mais completa da IA com o teste de Turing, aprendizagem de máquina, algoritmos genéticos e aprendizagem por esforço.

Segundo Turing (1950), quando se levanta a questão “máquinas podem

pensar?", pode-se levar à análise separada das palavras 'máquinas' e 'pensar' como usualmente são definidas, levando a conclusões não adequadas para a pergunta, e a tentativa de substituir tal questão por outra para tentar definir a questão levantada, só levará a outra questão que estará intimamente relacionada com a primeira. Propõe-se: ao invés de procurar uma definição para a questão, pode-se descrever através da solução de um determinado problema.

Com a dificuldade de definir a IA, torna-se mais fácil formar uma ideia através dos objetivos no estudo da IA, como proposto por Nascimento e Yoneyama (2004), que relatam que a IA busca prover máquinas com capacidade de realizar algumas atividades mentais do ser humano, através de alguns recursos computacionais com variadas arquiteturas. Estas atividades podem envolver a sensopercepção como tato, audição, visão, as capacidades intelectuais como aprendizagem, raciocínio dedutivo, memória, a linguagem sendo verbais ou gráficas; atenção, nas tomadas de decisão sob um estímulo, a orientação, tendo um conhecimento sob sua situação temporal e espacial num dado ambiente, a conceituação que busca captar a essência dos objetos e agrupá-los em classe.

De acordo com Bittencourt (2001), o objetivo central da IA é simultaneamente a criação de teorias e modelos para capacidade cognitiva, prática, com implementação computacional, podendo mostrar-se com objetivo semelhante a do estudo da psicologia, sendo diferenciado pela sua implementação computacional, tornando-o autônomo de certa forma. Desta forma o modelo ou teoria de IA não precisa ser provada através da comparação com os resultados previstos com o comportamento humano.

Desviando das considerações filosóficas que abrange a IA pode-se afirmar que existem duas áreas que concentram as atenções dos pesquisadores em IA: a representação de conhecimento e busca. A primeira trata de capturar um problema numa linguagem formal e adequada para que este seja manipulado num computador, fornecendo todo conhecimento necessário para um comportamento inteligente. A segunda é a técnica de solução que explora o espaço de estados do problema, estágios sucessivos e alternativos no processo de solução, que permite numa busca gerar diferentes alternativas de resposta para diferentes escolhas (LUGER, 2004).

Na construção de sistemas inteligentes existem duas principais linhas de pesquisas: a linha conexionista e a linha simbólica. A linha conexionista visa à

modelagem da inteligência humana através da simulação dos componentes do cérebro, e as interligações de seus neurônios. A linha simbólica segue a tradição lógica estabelecendo-se a manipulação simbólica de uma grande quantidade de fatos especializados num domínio restrito na construção de sistemas inteligentes (BITTENCOURT, 2001).

Na formalização do problema e a busca da solução, o conhecimento especialista é de suma importância para obtenção de sucesso na resolução de problemas específicos e complexos. No início do estudo da IA, utilizava-se o conhecimento genérico na solução de problemas gerais por meio de sistemas GPS *(General Problem Solver)*. Conhecidos como mecanismos de busca geral, tais abordagens foram chamadas de métodos fracos, pois não suportavam um aumento na escala do problema, apresentando um desempenho insatisfatório para problemas grandes e difíceis (RUSSELL, 2003). A solução de problemas complexos exige uma combinação de entendimento teórico e empírico sobre um problema específico, apoiado por um conjunto de regras heurísticas (conhecimento do maior número de estados possíveis assumidos no processo de solução) (LUGER, 2004). Entretanto, hoje, alguns pesquisadores acreditam que grandes conjuntos de "regras complexas" são demasiado custosas para os cientistas. Em vez disso, acredita-se que o melhor caminho para a inteligência artificial é através de um paradigma em que os seres humanos só devem escrever conjuntos de regras simples e fornecer um meio para que o sistema se adapte. Comportamentos complexos como a inteligência vão emergir da aplicação paralela e interação destas regras (MITCHELL, 1999). Assim, os sistemas especialistas devem ser construídos através da extração do conhecimento de um especialista humano utilizando IA para solução de problemas similares (LUGER, 2004).

3.2 - ALGORITMO GENÉTICO (AG)

Otimização é um mecanismo utilizado para determinação de uma ação que proporciona um máximo de benefício, medido por um critério de desempenho pré-estabelecido, em que o critério de desempenho deve ser minimizado caso represente um custo, ou maximizado, caso seja associado a retorno (NASCIMENTO JÚNIOR, 2004).

Segundo Bueno (2009), métodos heurísticos são algoritmos exploratórios que buscam resolver problemas que, geralmente, não envolvem a implementação computacional de um conhecimento especializado. Por este motivo, muitas vezes, esses métodos são classificados como “busca cega”. Uma solução ótima de um problema nem sempre é o alvo dos métodos heurísticos, uma vez que, tendo como ponto de partida uma solução viável, baseiam-se em sucessivas aproximações direcionadas a um ponto ótimo.

No estudo das metas heurísticas, podemos citar o surgimento do conceito de evolução. A evolução biológica é uma fonte atraente de inspiração para abordar difíceis problemas computacionais. A evolução é, na verdade, um método de busca entre um grande número de possibilidades, por exemplo, o conjunto de sequências de genes possíveis para - "Soluções" que permitem que os organismos, sobrevivam e se reproduzam em seus ambientes. Evolução também pode ser visto como um método para a adaptação à mudança de ambientes. As "regras" da evolução podem ser vistas de modo simples: As espécies evoluem por meio da variação aleatória (via mutação, recombinação e outros operadores), seguida pela seleção natural em que os mais aptos tendem a sobreviver e reproduzir-se, assim, propagar seu material genético para as gerações futuras. Este conjunto de regras é responsável pela extraordinária variedade e complexidade que vemos na biosfera. Existem várias abordagens que têm sido seguidas em matéria de Algoritmos evolutivos (AE). A forma mais utilizada de algoritmos evolutivos são algoritmos genéticos (AG's) (MITCHELL, 1999).

Com o aumento do recurso computacional, as técnicas de busca que anteriormente eram bastante simples, tornaram-se mais elaboradas e hoje existe um foco maior em buscas baseadas em população. Citando como exemplo, tem-se os Algoritmos Evolucionários (EA) e o algoritmo baseado em Colônia de Formigas (ACO) que hoje são utilizados em uma gama de problemas, tais como otimização numérica, combinatória, bioinformática, problemas de alocação, entre outros (SILVA, 2008).

De acordo com Fernandes (2008), os AG’s são métodos adaptativos, que podem ser usados para resolver problemas de busca e otimização. Este se baseia na teoria evolutiva postulado em Darwin (1859), onde as populações evoluem na natureza de acordo com os princípios de seleção natural e sobrevivência dos mais aptos. Imitando este processo, os AG’s são capazes de fornecer soluções para os

problemas do mundo real.

Segundo Michalewicz (1999), a evolução do programa (implementação do AG) se dá através de um algoritmo probabilístico, que contém uma população principal de indivíduos para cada interação. Cada indivíduo pertencente à população representa uma solução potencial para o problema. A mensuração do potencial de solução de cada indivíduo é dada através do '*fitness*', um valor numérico obtido através de uma função de avaliação '*fit*', mesurando sua aptidão em relação ao meio, que pode ser obtida através de um custo ou retorno. Enquanto uma condição de parada não for satisfeita, uma nova população é formada através da seleção individual, têm-se novos indivíduos são inseridos na população em substituição de alguns outros, fornecendo um novo conjunto de soluções.

No processo de evolução, a mutação é uma pequena transformação a que um indivíduo está sujeito em suas características. Ao sofrer uma mutação, este será considerado um novo indivíduo na população, com uma nova aptidão. Novos indivíduos também são formados através do cruzamento entre dois ou mais indivíduos da população, gerando descendentes que compartilham as características de seus pais. Quanto menor for sua aptidão, menor será a probabilidade de ser selecionado para reprodução. Desta forma, uma nova população é gerada com possíveis soluções, substituindo a anterior e apresentando características mais promissoras para a solução do problema. Desta forma as melhores características são repassadas para as novas gerações, enquanto as características não promissoras tenderão a extinção, convergindo para espaços de busca mais promissores (MICHALEWICZ, 1999).

O Algoritmo Genético pode ser implementado computacionalmente de diversas formas. Para melhor compreensão a representação binária será adotada para exemplificar através de um pseudocódigo e simbolicamente um AG simples, também conhecido como Algoritmo Genético canônico. O individuo binário é representado por um conjunto de gene denominado cromossomo. Cada gene é representado por um bit, e os subconjuntos de bits (genes) do cromossomo representam suas características como mostra a Figura 4.

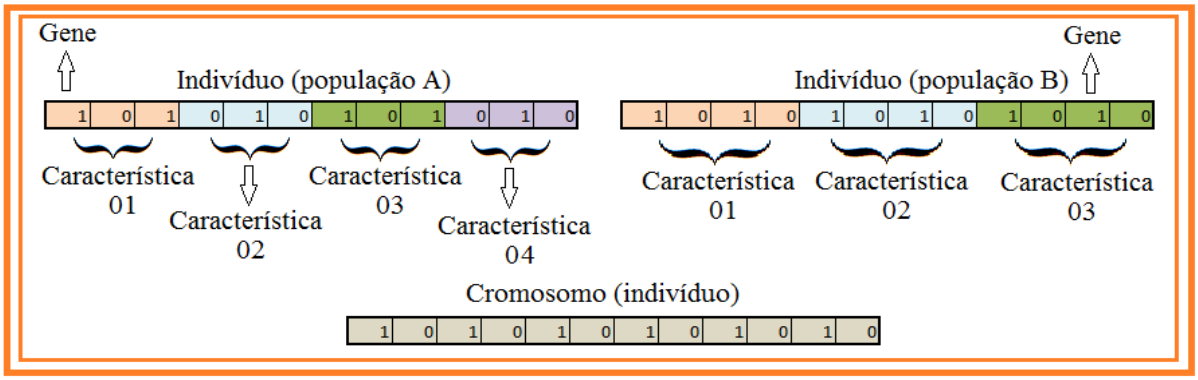


Figura 4: Representação de um indivíduo binário
Fonte: Próprio autor.

Na Figura 4, dois indivíduos de populações diferentes são representados por um cromossomo de 12 genes (bits). O indivíduo da população **A** possui 4 características, enquanto o indivíduo pertencente à população **B** possui um conjunto de 3 características. A quantidade de genes e características dos indivíduos de uma população depende da formulação do problema, sua precisão e outras particularidades do modelo.

Na Figura 5, é apresentado um fluxograma padrão do AG.

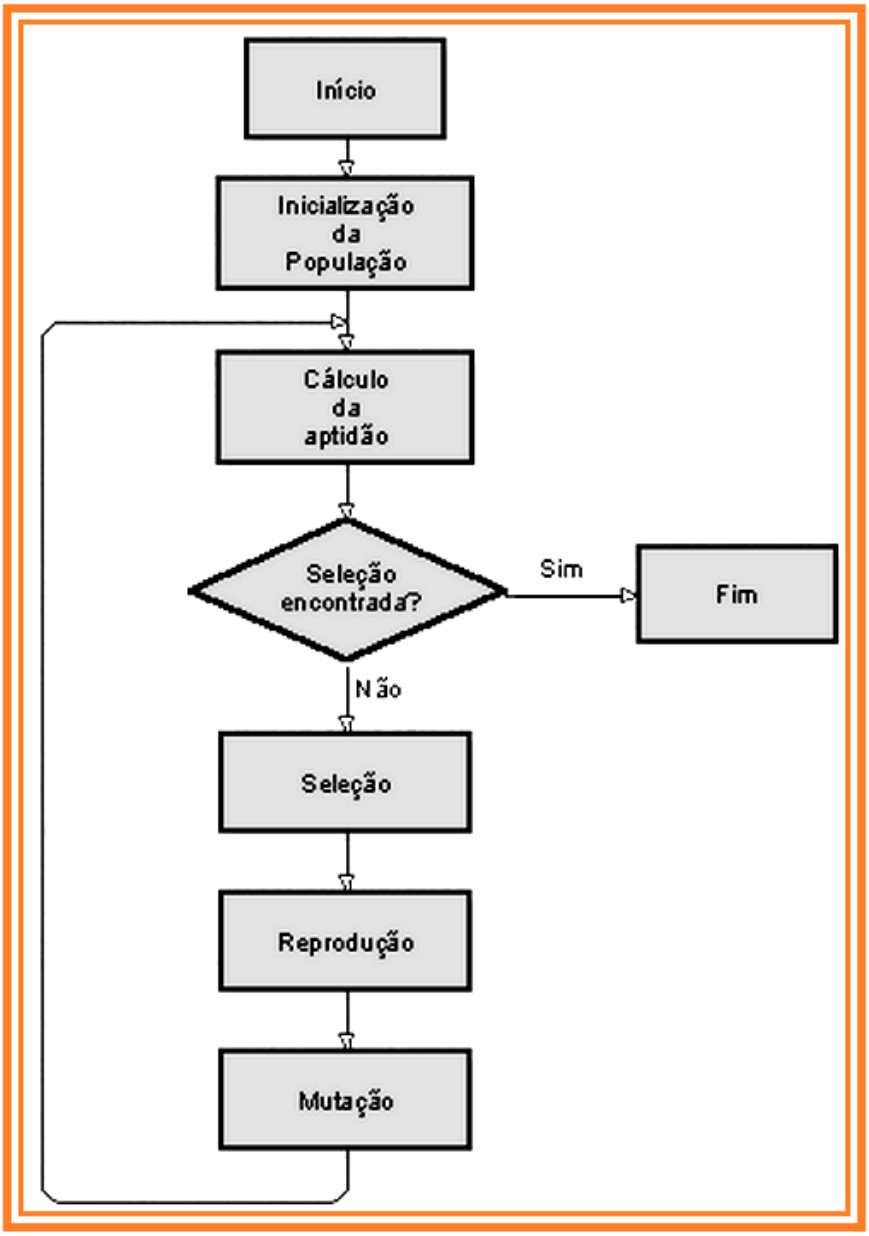


Figura 5: Fluxograma padrão
Fonte: Elaborada pelo autor.

3.2.1 Pseudocódigo

No pseudocódigo do AG abaixo, exemplifica e explica passo a passo as etapas do fluxograma anterior.

1- Início {Algoritmo Genético simples}

2- Gera população inicial de tamanho n: a população é gerada aleatoriamente com conjunto de características pré-definidas. Se tratando de problemas numéricos, estes possuem intervalos numéricos definidos, exemplificados na Figura 6.

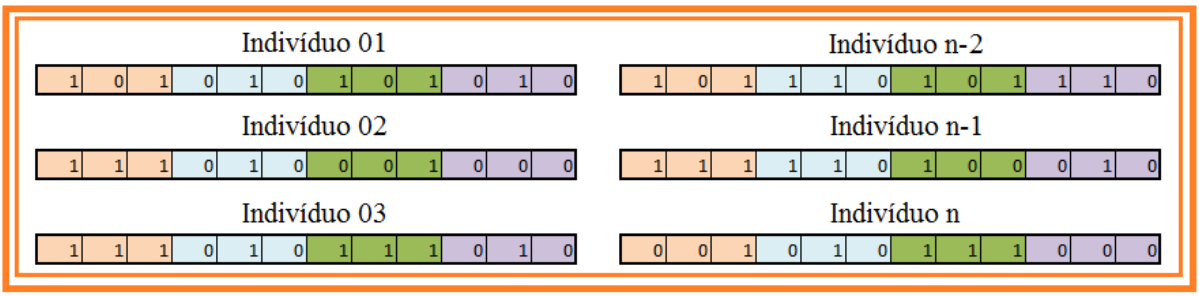


Figura 6: Indivíduos pertencentes à mesma população.
Fonte: Próprio autor.

A Figura 6 exemplifica indivíduos distintos pertencentes a uma população qualquer.

3- Calcula aptidão de cada indivíduo: atribui, em cada indivíduo, um valor de aptidão individual (*fitness*), através de uma função de avaliação. Em seguida, uma avaliação é feita em cada indivíduo em relação à população total (*fitness* global).

4- **Ciclo reprodutivo:** seleciona dois indivíduos da população baseado numa probabilidade de seleção proporcional à aptidão de cada indivíduo. Define-se uma taxa de reprodução em relação à população.

4.1- Seleção: o método roleta é um dos mais aplicados para seleção dos indivíduos, em que um valor aleatório é gerado numa faixa de 0% a 100% (entre 0 e 1), que corresponde a aptidão do indivíduo em relação à população. A Tabela 9 mostra a relação do *fitness* individual, relacionando-os com o *fitness* global.

Tabela 9: Tabela de indivíduos para seleção por meio de roleta.

Indivíduo	***Aptidão Global***	***Aptidão Acumulada***
1	0,1	0,10
2	0,15	0,25
3	0,05	0,30
n-2	0,11	0,82
n-1	0,02	0,93
n	0,07	1,00

Fonte: Elaborado pelo autor.

4.2- Cruzamento: variadas técnicas de cruzamento podem ser aplicadas, cada uma delas podendo dar um retorno particular em relação à melhora da espécie. O mais usual é que dois novos indivíduos (descendentes) sejam gerados a partir de dois pais. Seus cromossomos serão compostos pela soma das metades das

características genéticas dos indivíduos reprodutores, este processo é conhecido como *crossover*, como mostra na Figura 7.

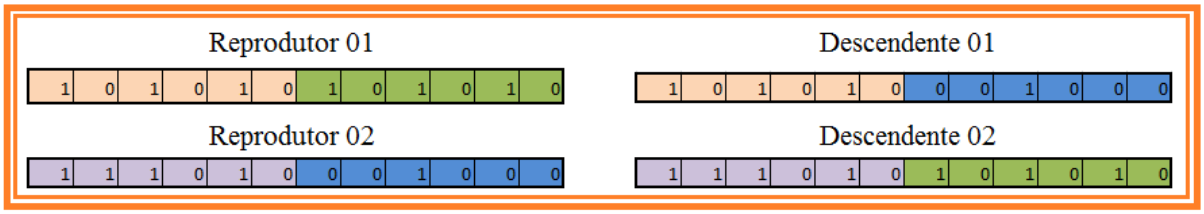


Figura 7: Ciclo de reprodução.
Fonte: Próprio autor.

A Figura 7 representa o *crossover* sendo aplicado na geração de dois descendentes. O primeiro filho recebe a primeira metade do material genético do reprodutor 01 e a segunda metade do material genético do reprodutor 02. O inverso é aplicado na geração do descendente 02.

5- Mutação: no processo de mutação, um ou mais genes podem ser selecionados para alteração da(s) característica(s) de um indivíduo. Uma taxa de mutação é pré-determinada em relação à população para possibilitar fugas de ótimos locais como mostra a Figura 8. O método roleta pode ser adotado para seleção do indivíduo que sofrerá a mutação.

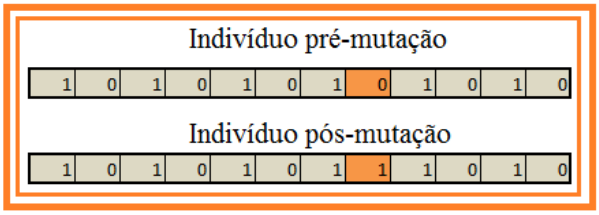


Figura 8: Processo de mutação de um indivíduo.
Fonte: Próprio autor.

A Figura 8 mostra um gene que sofre mutação, o bit tem seu valor modificado de 0 para 1. Após a mutação o valor de *fitness* individual e global é atualizado.

Michalewicz (1999), adota uma taxa de cruzamento de 0,25 garantindo a renovação de no máximo metade da população neste processo, e mutação de 0,01.

6- Caso não atenda a condição de parada retorna ao passo 5. A condição de parada pode ser uma condição de convergência ou atendimento de um conjunto de restrições.

7- Fim

Outras estratégias de *crossover*, mutação podem ser adotadas na

implementação dos AG’s, assim como taxas diferentes das mencionadas podem ser escolhidas no objetivo de melhorar a convergência do algoritmo.

3.3 REDES NEURAIS ARTIFICIAIS (RNA’S)

O modelo biológico do cérebro humano, constituído de neurônios e sistema nervoso, serviu como inspiração para o surgimento das Redes Neurais Artificiais (RNA), implementada através de uma arquitetura computacional com funcionamento similar. Em conjunto com a ideia central de Inteligência Artificial na década de 1950, também surgiu um conjunto de técnicas que permitiriam o desenvolvimento de sistemas suficientemente genéricos para resolver problemas gerais, condicionados à sua devida representação. Assim, as RNA’s, também chamadas de abordagem conexionista, surgiram como uma alternativa à abordagem simbólica, que se baseava na lógica (KRATZER, 1991).

Os primeiros modelos propostos por McCulloch (1943) tinham como objetivo imitar a realidade biológica. Tal preocupação hoje não é compartilhada por muitos pesquisadores. Com a teoria de que o aprendizado biológico baseava-se no reforço das ligações sinápticas entre os nodos, Hebb a descreve em forma matemática e, por meio da variação dos pesos de entrada dos nodos, demonstra a capacidade de aprendizagem de uma RNA. Mais tarde, Widrow e Hoff sugeriram uma regra de aprendizado conhecida como regra delta (AZEVEDO, 2000), mas foi em 1958, com Rosenblatt, que foi demostrado que, se fossem acrescidas sinapses ajustáveis às RNA's, essas poderiam ser treinadas para reconhecer certos tipos de padrões. A rede utilizada era chamada de Perceptron (ROSENBLATT, 1958).

O trabalho Minsky e Papert em 1969 rendeu duras críticas ao *perceptron*, expondo a sua incapacidade de resolver problemas que não eram linearmente separáveis, e o fato de que o modelo de aprendizado, que garantia a convergência para modelo *perceptrons* com única camada, não acontecia para redes multicamadas. A euforia, que havia sido perdida, basicamente só foi retomada em 1986, com o algoritmo de retropropagação do erro (*backpropagation)* (BRAGA, 2011).

3.3.1 Neurônios

Apesar da complexidade do cérebro humano, este é formado por estruturas relativamente simples, chamadas de neurônios. O neurônio biológico mostrado na Figura 9 é dividido em três seções: corpo celular, os dendritos e o axônio. Cada uma possui uma função específica, porém estes são complementares.

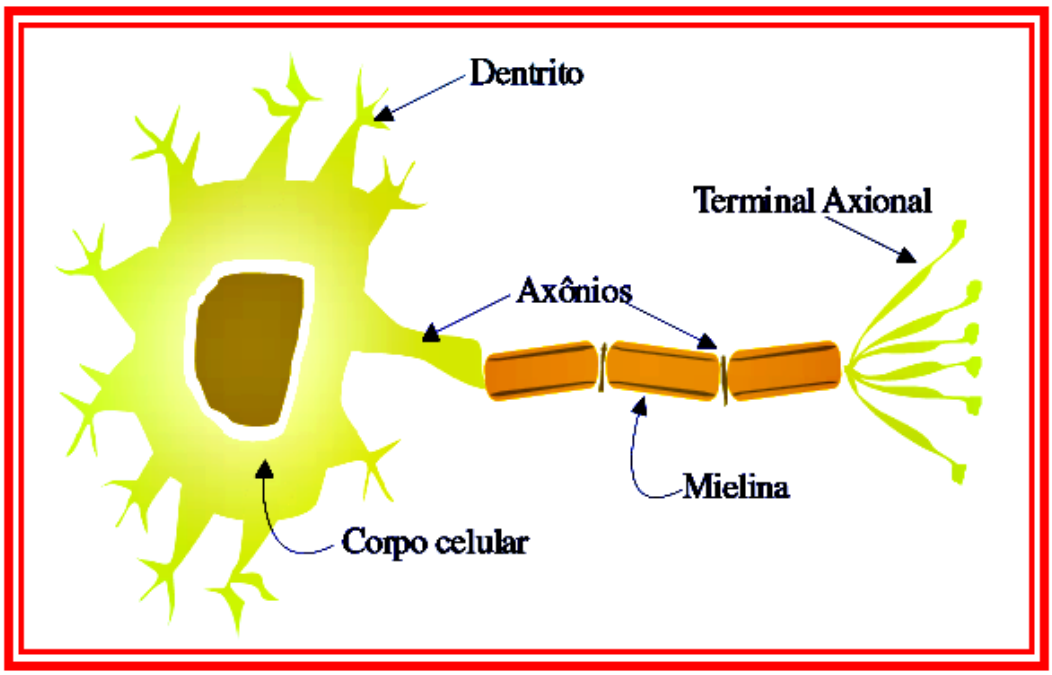


Figura 9: Modelo de neurônio biológico.
Fonte: Mota, (2007, p.25.)

Os dendritos têm por função receber as informações dos neurônios vizinhos conectados e conduzi-las até o corpo celular. Neste, a informação é processada e novos impulsos são gerados e transmitidos a outros neurônios através do axônio até os dendritos dos neurônios seguintes (BRAGA, 2011). Segundo Steiner (1995), o cérebro possui as seguintes características: é uma rede altamente interconectada; apresenta paralelismo maciço, muitos neurônios operam ao mesmo tempo; o processamento é distribuído de modo que a informação é não localizada, podendo um fato corresponder à ativação de um conjunto de neurônios; admite tolerância a falhas, assim, o prejuízo a poucos neurônios não afetam de modo significo a operação do cérebro.

O modelo de um neurônio artificial, mostrado na Figura 10, é dividido em três elementos básicos do modelo neural.

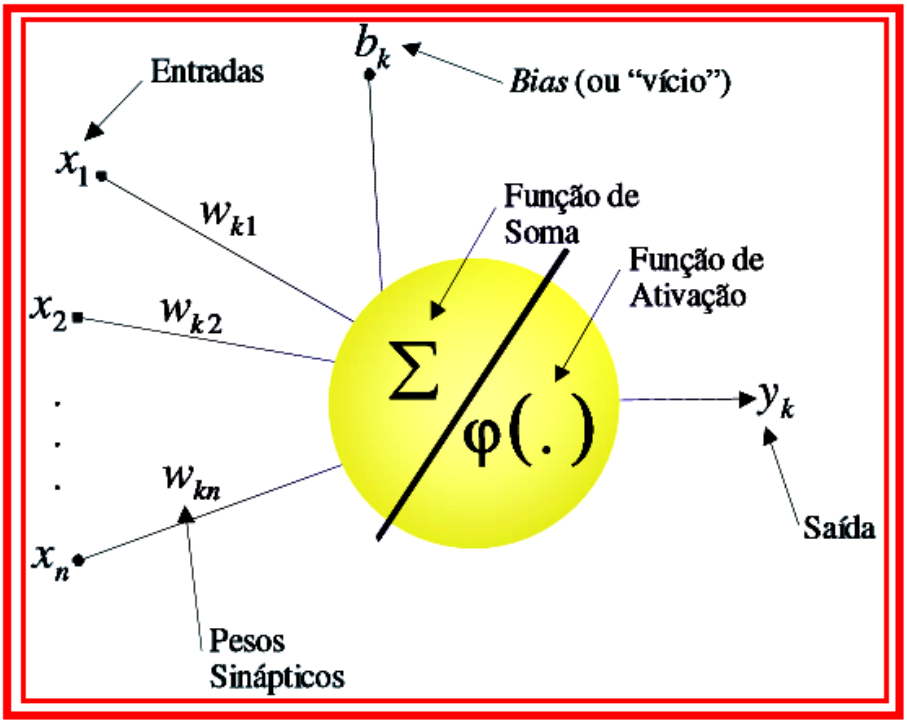


Figura 10: Modelo Neural Artificial.
Fonte: Mota (2007, p. 25.)

O conjunto de sinapses ou elos de conexão é caracterizado por um peso que será atribuído à entrada associada. O somador é responsável pela soma dos sinais de entrada ponderados pelas respectivas sinapses. A função de ativação restringe a amplitude do sinal de saída e o intervalo permissível de amplitude do sinal de saída (HAYKIN, 2007). Os gráficos da Figura 11 apresentam alguns exemplos de função de ativação como: tangente hiperbólica sigmóide (*tansig*), função sigmóide (*logsig*), função linear (*purelin*), função de base radial (*radbas*) e função degrau ou limiar (*hard-limit*).

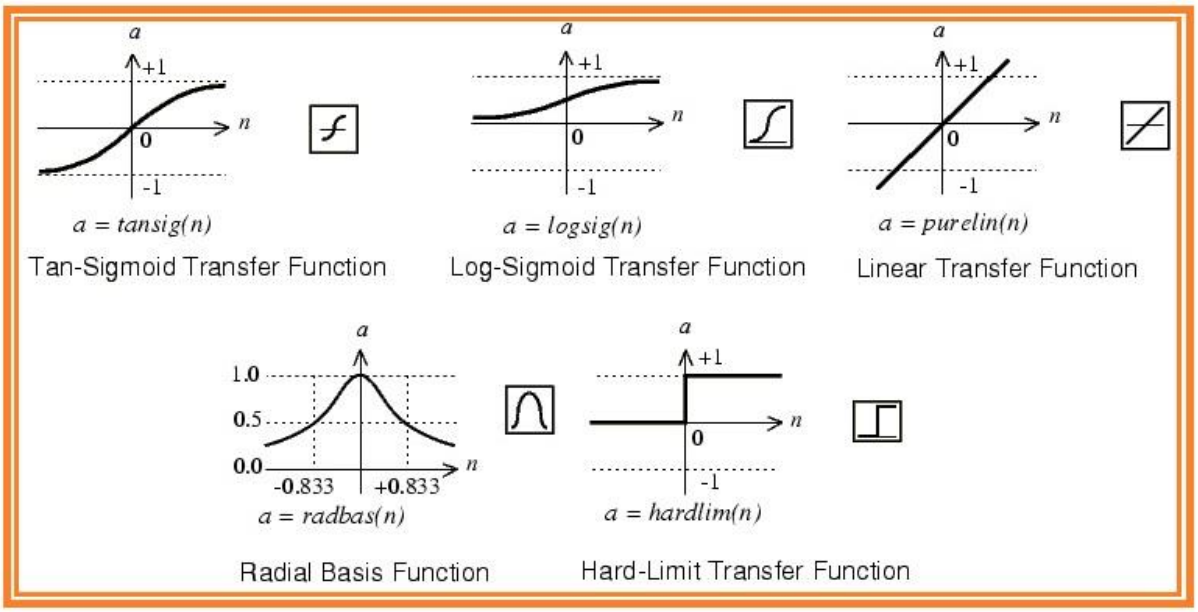


Figura 11: Funções de ativação.
Fonte: *Matlab* R2013a.
Disponível em: <http://www.mathworks.com/help>. **Acesso em:** outubro de 2013.

Normalmente, as funções não lineares são as mais utilizadas. Outra prática comum é a de utilizar uma mesma função de ativação para todos os elementos de uma mesma camada numa rede e, até mesmo, para toda a rede. Entretanto, não é uma regra (MOTA, 2007). Vale lembrar que as funções de ativação devem ser escolhidas de acordo com as características da massa de dados. Algumas funções de ativação requerem a normalização destes dados na entrada, consequentemente também emitindo uma resposta normalizada.

Segundo Haykin (2007), o modo pelo qual os neurônios de uma rede estão estruturados se dá de modo íntimo ao algoritmo de aprendizado utilizado em seu treinamento. Basicamente, existem três tipos de arquitetura fundamentalmente diferentes: as redes de camada única alimentadas adiante, as redes de múltiplas camadas alimentadas adiante e as redes recorrentes.

Redes alimentadas adiante (*feedforward*): possui uma única camada apresentando como a forma mais simples de rede. Todo calculo é feito na camada de saída.

Redes alimentadas adiante (*feedforward*) com múltiplas camadas (*multilayer*): nesta, a camada de entrada continua não sendo considerada, apenas as camadas intermediárias e a camada de saída.

Redes alimentadas adiante com laços de realimentação (redes recorrentes): se distingue das anteriores por haver pelo menos um laço de realimentação. As três

estruturas são representadas na Figura 12.

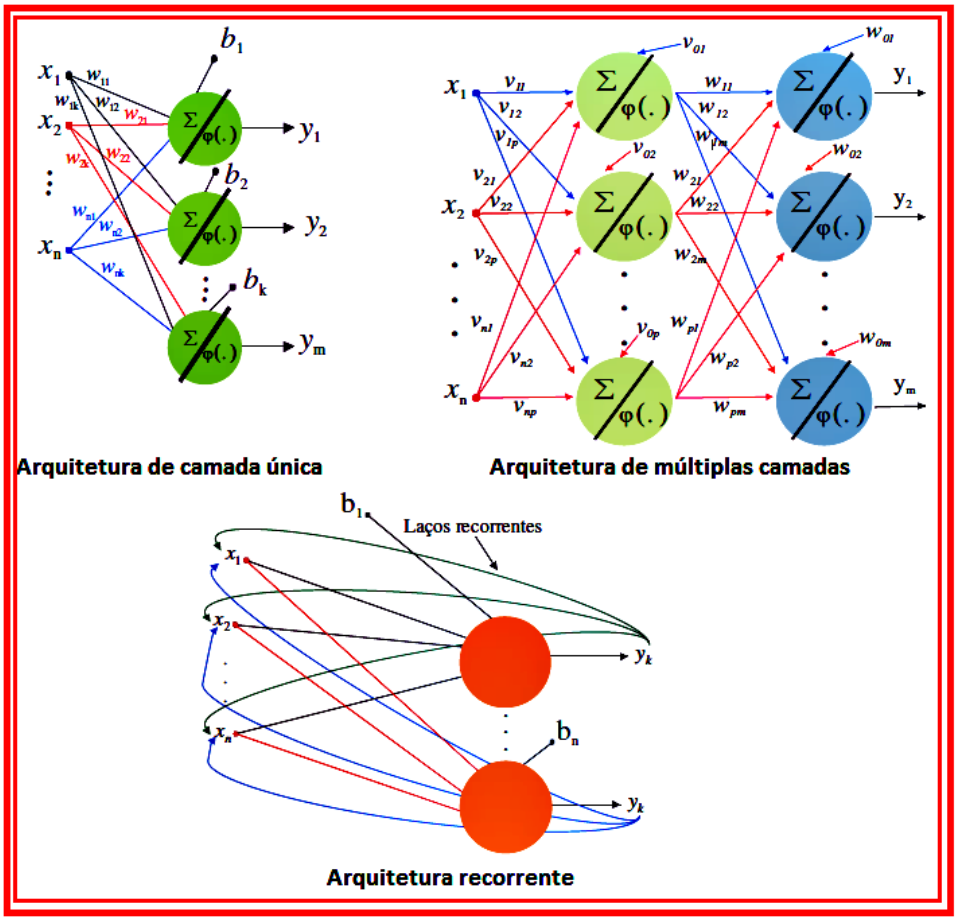


Figura 12: Arquitetura *feedforward*.
Fonte: Mota (2007, p.28).

3.3.2 Configuração das RNA's

Uma *feedforward Perceptron* de Múltiplas Camadas (MLP- *Multilayer Perceptron*), com duas ou mais camadas ocultas pode ser considerada como um aproximador universal, pois consegue realizar um mapeamento com relação entrada/saída, uma característica existencial. Isto porque, cada neurônio, na primeira camada, cria uma saliência e a próxima camada combina estas saliências em regiões disjuntas do espaço. Entretanto, um problema ainda sem solução, é a determinação do número ótimo de camadas escondidas e do número de neurônios em cada camada para um dado problema, faltando um mecanismo sistemático de se chegar à rede neural mais indicada em cada aplicação. Além dos desafios associados ao processo de otimização de parâmetros e determinação da estrutura

da RNA na etapa de aprendizado supervisionado, outro desafio está associado à capacidade de generalização da rede. Deste modo deve-se controlar a flexibilidade do mapeamento resultante, sem que este se apresente mais complexo do que o necessário para melhorar o desempenho no treinamento (GEMAN et al., 1992 apud RAVIV, 1999).

Propostas são apresentadas com abordagem evolutiva para a otimização da configuração estrutural das RNA's e similares, como EP (*evolutionary programming*) encontrado na literatura de YAO (1997), porém pelo fato de explorar uma região de busca contendo todas as arquiteturas possíveis, tornam-se extremamente custosas computacionalmente, embora sejam capazes de produzir bons resultados. Indivíduos diferentes, mas com mesma aptidão também podem levar a dificuldades numa abordagem evolutiva.

3.3.3 Aprendizado das RNA's

Segundo Fischler (1987), o conhecimento refere-se a um conjunto de informações armazenadas, ou modelos utilizados por uma pessoa ou máquina para interpretar, prever ou responder apropriadamente aos estímulos do mundo exterior. De acordo com Haykin (2007), são duas as principais características da representação do conhecimento: informação realmente tornada explicita e como a informação é codificada fisicamente para posterior utilização. A informação codificada servirá como um conjunto de dados ou amostra para treinamento da RNA através de um algoritmo de aprendizagem apropriado.

Algoritmos de aprendizagem é uma característica importante da RNA e seu algoritmo de aprendizagem. O processo de aprendizagem de uma rede neural é o ajuste dos pesos sobre os valores de entrada para uma determinada regra de aprendizado controlado. Dependendo da finalidade há uma regra de aprendizagem para cada configuração de RNA. Hoje, os mais utilizados são baseados em regras de Hebb. Dois tipos de aprendizagem podem ser citados: monitorada (supervisionada) e aprendizado não supervisionado. Neste último caso, a rede, durante o processo de aprendizagem, não controla o valor de saída com um valor de referência externo. No treinamento supervisionado, a RNA aprende com valores de entrada pré-classificados (padrões) para uma certa classe (saída padrão). O treino é

dado, neste caso, a partir de uma referência exterior, comparando a um alvo real, gerando um erro. Este erro é usado pela rede para o mapeamento dos valores de entrada para o aprendizado (ajustes dos valores de pesos e bias) de uma determinada classe. Aprender através de correção de erro é o mecanismo de aprendizagem mais utilizado (TAWIL,1999).

3.3.3.1 Algoritmo *Backpropagation*

O Algoritmo de aprendizagem *backpropagation*, utilizado no treinamento da rede neural multicamadas é um procedimento de treino supervisionado. Varias abordagens foram feitas na última década para a sua aplicabilidade em numerosas aplicações (BRYAN, 2011). Este possui um bom desempenho quando não se dispõe de hardware potente. O treinamento de uma rede por *backpropagation* possui os seguintes estágios: a apresentação dos padrões à rede, a comparação do valor obtido com o valor desejado, a retropropagação do erro e o ajuste dos pesos sinápticos. De maneira mais simples, pode-se dividir em duas as fases de treinamento, em que a primeira, chamada de *forward*, é utilizada para definir a saída da rede para um padrão de entrada e *backward*, que utiliza a saída da rede com a saída desejada para ajustar os pesos de suas conexões (BRAGA, 2011). A Figura 13 ilustra as duas fases.

Segundo Braga (2011), uma dificuldade encontrada numa rede MLP com o algoritmo *backpropagation* está associada à sensibilidade da característica da superfície do erro, em baixo gradiente e mínimos locais encontra dificuldades para convergência, exigindo a aplicação de outras abordagens para acelerar o algoritmo e escapar dos mínimos locais.

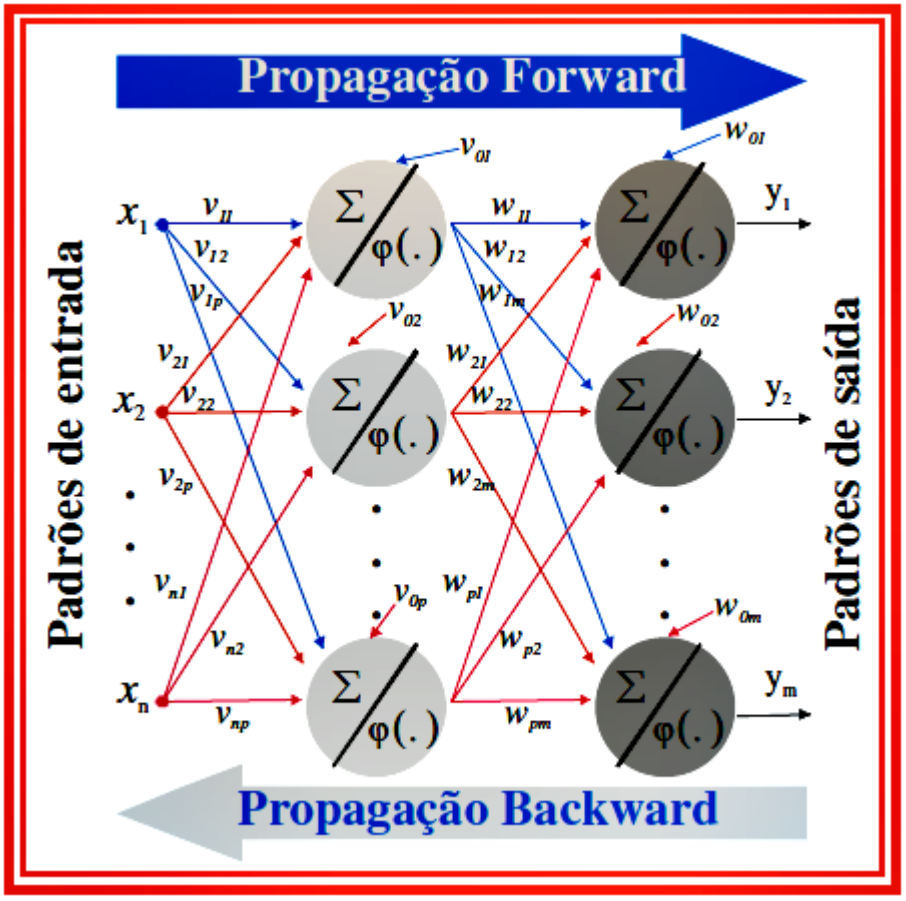


Figura 13: Fases do Algoritmo *backpropagation.*
Fonte: Mota, 2007, p.31.

3.3.3.2 Métodos de Segunda Ordem

Estes algoritmos são concebidos a partir de um rigor matemático baseado em modelos de otimização não linear, não apresentando o vínculo natural com a inspiração biológica inicialmente proposta para as RNA's. Hoje estes métodos são considerados os mais eficientes no treinamento de redes neurais do tipo MLP (SHEPHERD, 1997).

3.3.3.3 Método de Newton

O método de Newton pode ser considerado como o método local básico que utiliza informações de segunda ordem. Este aponta para um extremo local da função erro. Deve-se ter cuidado ao aplicar no treinamento das redes *perceptrons* multicamadas, pois este exigirá o cálculo da matriz Hessiana, exigindo a sua

inversão, que representa um elevado custo computacional, análise espectral e armazenagem de uma matriz quadrada que é da ordem do número de parâmetros P a serem ajustados (ZANETTI et al., 2007).

3.3.3.4 Levemberg Marquardt

O algoritmo proposto por *Levemberg Marquardt* é um dos algoritmos de segunda ordem mais rápidos para o treinamento de RNA's de tamanho moderado (JONES et al., 2005), sendo este uma variação do método de Newton (EDGAR, 1988) e melhora o método de *Gauss-Newton* por meio da utilização de uma taxa de aprendizado variável.

Uma explicação simplificada do cálculo do algoritmo de treinamento é mostrado neste trabalho, a partir da descrição de Hao Yuand et. al. (2011).

Como o método de *Newton* é aplicado para a atualização dos pesos, a fim de obter matriz Hessiana $\boldsymbol{H}$, a derivada de segunda da função erro total também deve ser calculado o que pode ser muito complicado. A fim de simplificar o processo de cálculo, a matriz Jacobiana $\boldsymbol{J}$ é introduzido como:

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w_1} & \frac{\partial e_{1,1}}{\partial w_2} & \cdots & \frac{\partial e_{1,1}}{\partial w_N} \\ \frac{\partial e_{1,2}}{\partial w_1} & \frac{\partial e_{1,2}}{\partial w_2} & \cdots & \frac{\partial e_{1,2}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{1,M}}{\partial w_1} & \frac{\partial e_{1,M}}{\partial w_2} & \cdots & \frac{\partial e_{1,M}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{P,1}}{\partial w_1} & \frac{\partial e_{P,1}}{\partial w_2} & \cdots & \frac{\partial e_{P,1}}{\partial w_N} \\ \frac{\partial e_{P,2}}{\partial w_1} & \frac{\partial e_{P,2}}{\partial w_2} & \cdots & \frac{\partial e_{P,2}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{P,M}}{\partial w_1} & \frac{\partial e_{P,M}}{\partial w_2} & \cdots & \frac{\partial e_{P,M}}{\partial w_N} \end{bmatrix}$$

O gradiente $\mathbf{g}$ utilizado para convergência pode ser encontrado através da relação da matriz jacobiana e da matriz de erro, sendo $\boldsymbol{g} = \boldsymbol{Je}$. O vetor erro $\mathbf{e}$ tem a

seguinte forma:

$$e = \begin{bmatrix} e_{1,1} \\ e_{1,2} \\ \cdots \\ e_{1,M} \\ \cdots \\ e_{P,1} \\ e_{P,2} \\ \cdots \\ e_{P,M} \end{bmatrix}$$

Como o pressuposto básico do método de *Newton*, tem-se que $\boldsymbol{H \approx J^T J}$.

Por combinação de equações pode-se extrair a regra de atualização do algoritmo de *Gauss-Newton* dada como:

$$w_{k+1} = w_k - \left(J_k^T J_k\right)^{-1} J_k e_k \qquad \text{eq. 01}$$

O algoritmo de *Levenberg-Marquardt* introduz outra aproximação da matriz Hessiana $\boldsymbol{H \approx J^T J + \mu I}$. Sendo $\boldsymbol{\mu}$ sempre positivo chamado coeficiente de combinação, $\boldsymbol{I}$ é a matriz identidade da própria equação, garantindo que os elementos da diagonal principal da matriz Hessiana aproximada será maior do que zero, garantindo que a matriz $\boldsymbol{H}$ é inversível. Assim a regra de atualização dos pesos no algoritmo de *Levenberg Marquardt* pode ser apresentado como:

$$w_{k+1} = w_k - \left(J_k^T J_k + \mu I\right)^{-1} J_k e_k \qquad \text{eq. 02}$$

Assim quando o coeficiente de combinação μ é quase zero, a equação z aproxima se para equação x, utilizando o método Gauss-Newton, quando o coeficiente de combinação $\boldsymbol{\mu}$ é muito grande utiliza o método do gradiente. A taxa de aprendizagem $\boldsymbol{\alpha}$ pode ser obtida da relação $\boldsymbol{\alpha = 1/\mu}$.

O algoritmo utilizado para o treinamento pelo método de *Levenberg Marquardt* é apresentado através do fluxograma da Figura 14.

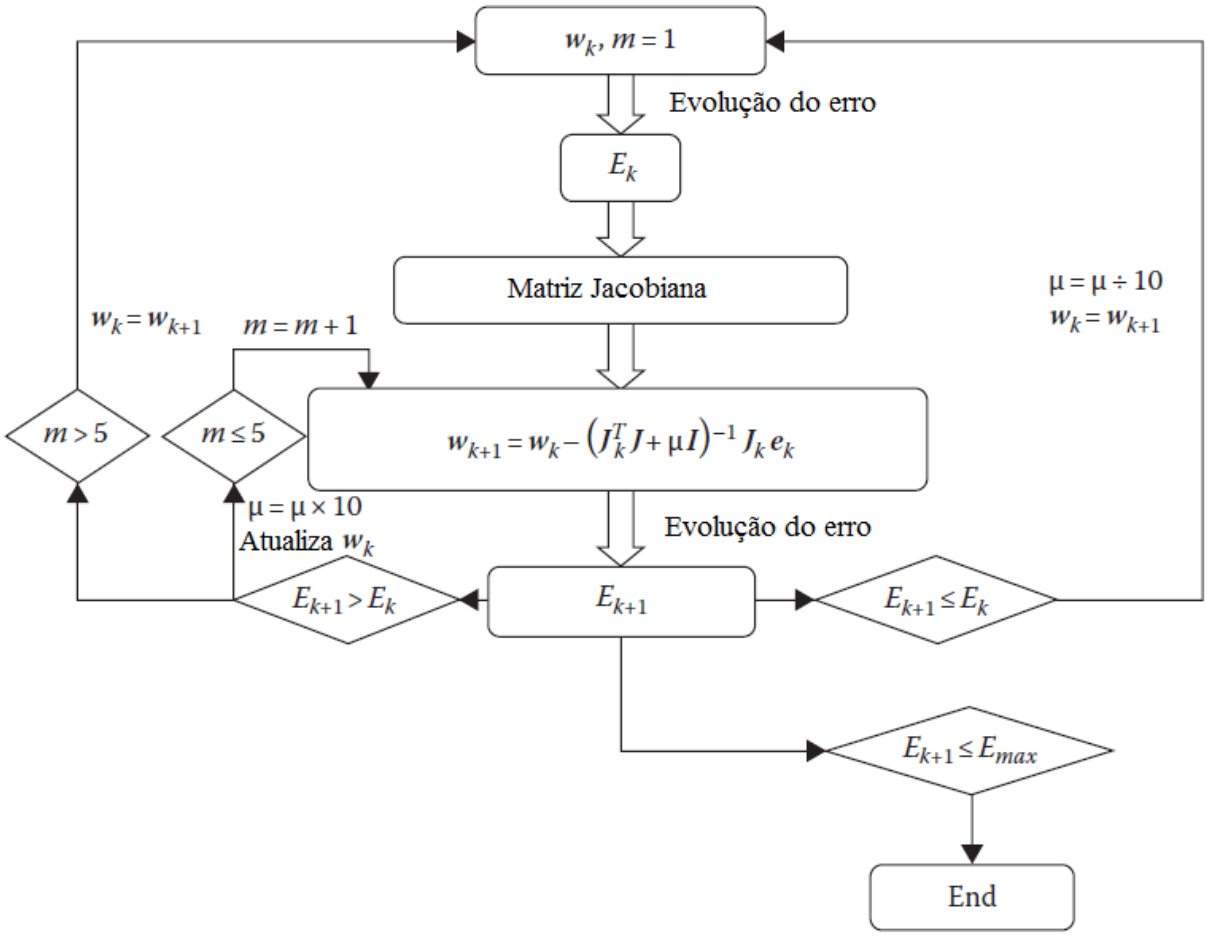


Figura 14: Fluxograma do Treinamento usando o algoritmo *Levenberg –Marquardt*.

Fonte: Hao Yuand et. al. (2011). Adaptada pelo autor.

3.4 CONTROLE MODERNO

Ogata (2010) diz que a teoria de controle moderno tem sido desenvolvida em razão da necessidade de atender às crescentes e rigorosas exigências de desempenho dos sistemas de controle, ao aumento da complexidade dos sistemas e ao acesso fácil e em larga escala aos computadores. Esta teoria é, também, essencialmente, uma abordagem no domínio de tempo, enquanto a teoria de controle convencional é uma abordagem no domínio da frequência complexa. A teoria de controle moderno enquadra sistemas de entradas e saídas múltiplas, lineares ou não lineares e variantes ou invariantes no tempo.

A representação de um sistema em Espaços de Estados consiste da representação de n funções de transferências na forma matricial. A modelagem de

um problema no espaço de estados é elaborada utilizando um conjunto de operações de transformação de estados para, a partir de um estado inicial, chegar a um estado final conforme desejado.

Para melhor compreensão são definidos alguns termos a seguir:

Estado – O estado de um sistema dinâmico como o menor conjunto de variáveis (variáveis de estado) (OGATA, 2010), tais que o conhecimento dessas variáveis num instante inicial, juntamente com o conhecimento da entrada que o sistema está submetido, determina completamente o comportamento do sistema em qualquer instante futuro (DORF, 2009).

Variáveis de estado – As variáveis de estado de um sistema dinâmico são aquelas que constituem o menor conjunto de variáveis capaz de determinar o estado desse sistema dinâmico. Este conjunto é capaz de descrever completamente o comportamento de um sistema dinâmico no futuro, desde que seja conhecido o seu estado inicial (OGATA, 2010). A quantidade de variáveis de estado é igual à quantidade de condições iniciais que está associado à ordem do sistema.

Dorf (2009) definem vetor de estado como a matriz coluna que consiste nas variáveis de estado.

Espaço de estados – Segundo Ogata (2010), é o espaço de *n* dimensional, cujos eixos coordenados são formados pelas variáveis de estado. Qualquer estado pode ser representado por um ponto no espaço de estados, definindo a configuração instantânea do sistema.

Equações no espaço de estados – A representação no espaço de estados envolve três tipos de variáveis que estão presentes na modelagem de sistemas dinâmicos: variáveis de entrada, variáveis de saída e variáveis de estado (OGATA, 2010).

Na representação no espaço de estados, a matriz **A** é denominada a matriz de estado do sistema, a matriz **X,** o vetor de estados composta pelas variáveis de estado, **B** é a matriz de entrada, **U** é o vetor de entrada, sinal proveniente de uma ação externa, vetor manipulado como sinal de controle. A matriz **C** é chamada de matriz de saída e **D** é a matriz de transmissão direta. A matriz composta pela derivada de X (**Ẋ**) representa a equação de estado, e **Y** à equação de saída. A seguir, são dadas as equações de estados e saída em espaço de estados.

$$\dot{X}(t) = AX(t) + Bu(t) \qquad \text{eq. (03)}$$

$$Y(t) = CX(t) + Du(t) \qquad \text{eq. (04)}$$

A equação de estado esta representada sob a forma simplificada para sistemas invariantes no tempo que admitam serem linearizadas num dado ponto.

A representação de um sistema no espaço de estados não é única no que diz respeito ao conjunto de variáveis de estados (DORF; BISHOP 2009). Também existem diversas formas de representação. A mais usual utilizada para controle por realimentação de estados é a forma canônica controlável, que possui uma relação de dualidade com a forma canônica observável. Dada uma representação de uma função de transferência.

$$G(s) = \frac{b_n s^n + b_{n-1} s^{n-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0}$$

A representação na forma canônica é dada:

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u$$

$$y = \begin{bmatrix} b_0 - a_0 b_n & b_1 - a_1 b_n & \cdots & b_{n-1} - a_{n-1} b_n \end{bmatrix} x + b_n u$$

A construção da matriz canônica é uma relação dos coeficientes da(s) função(ões) de transferência do sistema.

Dado:

$\mathrm{x} \rightarrow$ vetor de estado (vetor n)
$\mathrm{u} \rightarrow$ vetor de controle (vetor r)
$\mathrm{y} \rightarrow$ vetor de saída (vetor m)
$\mathrm{A} \rightarrow$ matriz $n \times n$
$\mathrm{B} \rightarrow$ matriz $n \times r$
$\mathrm{C} \rightarrow$ matriz $m \times n$
$\mathrm{D} \rightarrow$ matriz $m \times r$

Na construção de sistemas de controle por realimentação de estados é necessário que o sistema seja de estado completamente controlável, condição necessária para que cada ação de controle afete o estado do sistema. Também é necessário que as variáveis de estado estejam disponíveis para realimentação. Caso num sistema real, alguma destas variáveis não possam ser medidas diretamente, as mesmas deverão ser estimadas. Logo, exigirá que o sistema seja observável. Em simulações puramente computacionais, o sistema deve ser observável.

Segundo Ogata (2010), para estabelecer se o sistema é completamente controlável, é necessário que os vetores da matriz de controlabilidade [B, AB, ... , $A^{n-1}B$] sejam linearmente independentes, ou seja, o posto da matriz é igual ao número de linhas da matriz de controlabilidade. Para observabilidade a mesma idéia é aplicada na matriz de controlabilidade [C^t, C^tA^t, ... , $C^t(A^t)^{n-1}$].

3.4.1 Projeto de Sistemas de Controle no Espaço de Estados

Segundo Dorf (2009), para a realização do projeto de controle por realimentação de estado, tem que haver a possibilidade de todos os polos do sistema em malha fechada serem arbitrariamente posicionados no plano complexo. Se o sistema for controlável e observável, então pode-se realizar o objetivo de projeto de alocar os polos precisamente nas posições desejadas para se atender às especificações de desempenho.

No projeto do controlador, basicamente casa-se os coeficientes da equação característica do sistema com a da nova equação resultante da alocação dos novos polos. Este é obtido através de um vetor de ganho ou sinal de controle que será do tipo ***u = - Kx(t)***, o que significa que o sinal de controle é determinado por um estado instantâneo (configuração instantânea do sistema). A matriz $K_{1\times n}$ é denominada matriz de ganho de realimentação de estado (OGATA, 2010).

Adotando ***u = - Kx(t)*** a nova equação de estado passa a ser:

$$\dot{x} = (\mathrm{A} - \mathrm{Bk})x \qquad \text{eq. (05)}$$

A partir da equação característica da matriz A, tem-se,

$$|sI - A| = s^n + a_1 s^{n-1} + ... + a_{n-1} s + a_n \qquad \text{form. (01)}$$

deve-se determinar os valores de $a_1, a_2, ..., a_n$.

Utilizando os autovalores desejados (polos desejados de malha fechada), escreve-se o polinômio característico desejado:

$$(s - \mu_1)(s - \mu_2)...(s - \mu_n) = s^n + \alpha_1 s^{n-1} + ... + \alpha_{n-1} s + \alpha_n \qquad \text{form. (02)}$$

e determina-se os valores de $\alpha_1, \alpha_2, ..., \alpha_n$.

A matriz de ganho K de realimentação de estado requerida pode, então, ser determinada por:

$$K = [\alpha_n - a_n \quad \vdots \quad \alpha_{n-1} - a_{n-1} \quad \vdots \quad ... \quad \vdots \quad \alpha_2 - a_2 \quad \vdots \quad \alpha_1 - a_1]^{T^{-1}} \qquad \text{form. (03)}$$

Dispondo de um bom modelo, o desafio para projetar o controlador é encontrar a alocação dos novos polos no plano complexo que dê um resultado satisfatório em relação às restrições de projeto.

3.4.2 Controle Adaptativo

Desde o momento de lançamento até a entrada em órbita de um foguete, variações severas acontecem no ambiente de navegação, como variação da gravidade, densidade do meio, arraste aerodinâmico, perda de massa e outras, exigindo um sistema de controle robusto que se adapte para suportar tais mudanças e manter-se em condições de operabilidade dentro do especificado em projeto.

Franklin, (1998), relata que as variações na planta de um sistema de controle podem causar impactos severos no desempenho e estabilidade. Por este motivo, os projetos de sistema de controle têm um algoritmo que se reformula de acordo com as alterações na planta. Este é o principal assunto do controle adaptativo. Vale a pena notar que sempre existe uma amigável discussão entre os engenheiros de controle sobre esta definição. Uma definição é que o controlador adaptativo é um

sistema de controle que é projetado de um ponto de vista adaptativo. Outra é que o controle adaptativo é qualquer sistema de controle que monitora e realiza ajustes para melhorar a performance. Segundo Åström (1995), o controle adaptativo pode ser definido como um sistema de controle composto de um loop rápido para controle e um loop lento ajuste de controle. Nesse caso, a definição do termo adaptativo é para “modificar de acordo com as mudanças de circunstâncias”. Quase todos os sistemas de controles adaptativos modificam-se sob as mudanças de circunstâncias.

Segundo Bolton (2010), o controle adaptativo é utilizado em situações de controle em que os parâmetros da planta variam com o tempo ou, talvez com a carga. Com a mudança da função de transferência da planta, é necessária uma nova sintonia do sistema de controle para que as condições de otimização sejam atendidas.

Para se projetar um sistema de controle, devem ser definidos quatro critérios básicos:

1) Um modelo de planta que seja controlável e o range de sua validade.
2) Os valores dos parâmetros do modelo e suas esperadas variações.
3) Os objetivos de desempenho
4) As restrições do projeto, como os custos das ações de controle, limites de autoridade do controle e o custo pretendido do controlador.

3.4.3 Ganho programado

Segundo Flanklin, (1998), os algoritmos de controles são escolhidos baseados em algumas condições de operação, assim o ganho programado é classificado como adaptativo. Desde que os parâmetros de controle sejam projetados off-line com a informação da prioridade do sistema, a principal contribuição do ganho programado é a identificação adequada do vetor de controle que será usado.

As simulações são geralmente usadas para examinar a transição de uma região para a outra, ou seja, deve-se usar um ganho fixo, ou um ganho programado. Deve-se notar que frequentes questões de estabilidade são contornadas pela existência de várias pequenas regiões e assegurando que o controlador passará de uma para outra sem ficar preso em “fronteiras”. Assim, o controlador será capaz de

realizar o controle em diversas regiões sejam elas de ganho fixo ou agendado. Está fundamentado que, para poucas regiões que são alteradas raramente, ou possuem parâmetros que são alterados muito devagar, a estabilidade do sistema pode ser assegurada se cada parte da lei de linearidade for examinada, isso se essas leis forem fixadas. Portanto, o ganho programado é geralmente considerado como dependente de alguma variação lenta de parâmetro (FLANKLIN, 1998).

Segundo o autor (op. Cit.), este método é um sofisticado conceito, frequentemente usado para avaliar a estabilidade do sistema. Na prática, a análise de estabilidade do ganho programado consiste na realização de um acompanhamento mais próximo dos sistemas de simulações.

3.5 METODOLOGIA DA SIMULAÇÃO

Segundo Freitas Filho (2008), a simulação permite realizar estudos sobre um determinado sistema modelado para responder algumas questões do tipo: o que aconteceria se? O principal objetivo no uso deste método é que tais perguntas podem ser respondidas sem que o sistema sob investigação, sofra qualquer distúrbio, uma vez que a simulação seja computacional, permitindo que tais estudos sejam realizados sobre sistemas que ainda não existem, direcionando a desenvolvimento de sistemas eficientes, antes que alterações físicas sejam realizadas. Um estudo simulado permite a percepção de comportamentos sutis no modelo em que no sistema real poderia passar despercebido, e, com a possibilidade de aplicar animações, também permite que o comportamento do modelo seja visualizado.

Um modelo é uma abstração da realidade, aproximando do verdadeiro comportamento do sistema, mas sempre com menor complexidade que o sistema real. A complexidade de um sistema real é principalmente relacionada à natureza dinâmica e na aleatoriedade, que podem ser reproduzidos com boa fidelidade pelo modelo computacional (CHWIF, 2007).

A simulação pode ser caracterizada em três categorias básicas: simulação Monte Carlo, em que o tempo não é considerado explicitamente como variável, e utiliza-se de geradores de números aleatórios para simular sistemas físicos ou matemáticos; simulação contínua, em que os estados das variáveis mudam

continuamente no tempo e simulação discreta, que mudam seu estado em momentos discretos no tempo, a partir da ocorrência de um evento. Em alguns modelos de simulação pode haver a necessidade de combinar as três categorias bases, o que é denominado simulação híbrida ou combinada (CHWIF, 2007).

3.5.1 Sistemas e modelos

Representar através de um modelo um sistema ou fenômeno observável pode se apresentar como um desafio, dependendo de sua complexidade, principalmente quando surge a necessidade de se obter um modelo matemático a partir de dados observado, que é chamado de modelagem empírica, que requer pouco ou nenhum conhecimento prévio do sistema, e não partindo das equações matemáticas que regem a física do processo, chamadas de modelagem fenomenológica ou conceitual que requer conhecimento aprofundado do fenômeno e exige grande demanda de tempo, podendo se tornar inviável. Nos dois tipos de modelo, técnicas e requisitos distintos são utilizadas para sua obtenção (AGUIRRE, 2007).

Freitas Filho (2008) nos diz que a modelagem pressupõe um processo de criação e descrição, que possui um certo grau de abstração, direcionando para uma simplificação do modelo real. Geralmente estas descrições são feitas na forma matemática ou lógica que, no seu todo, chamamos de modelo. Simulação é um dos muitos métodos que podemos utilizar para analisar e estudar sistemas os problemas em questão. Os modelos matemáticos não são os únicos utilizados no estudo de simulação. Outros tipos podem ser empregados como modelos físicos, como maquete, protótipos, túnel de vento e outros. Os modelos computacionais são utilizados para a simulação computacional. Sua aplicação depende da natureza do sistema em estudo, como exemplo, existem modelos computacionais voltados para simulação de sistemas modelados de modo discreto e para sistemas modelados de modo continuo no tempo.

Segundo Taylor (1970 apud CHWIF, 2006), sistema é como um conjunto de objetos que podem encontrar ou definir algum tipo de relação, atuando e interagindo, com o propósito de alcançar determinado objetivo ou propósito lógico.

O modelo desenvolvido para um determinado sistema é apenas uma representação aproximada, consequentemente não existe o modelo do sistema, mas

sim uma família de modelos com características e desempenhos variados (AGUIRRE, 2007), contendo em si apenas algumas características do sistema real.

3.5.2 Tipos de modelos

Modelos **estáticos ou dinâmicos**: Os modelos estáticos relacionam às variáveis sem quantificar sua dependência temporal. Se a evolução do sistema for temporal, é desejado um modelo dinâmico (AGUIRRE, 2007). Vale lembrar que estamos classificando a obtenção do modelo e não o sistema, pois, em muitos casos, para um mesmo sistema, podemos ter modelos de classificações diferentes, dependendo do interesse em estudo.

Modelos determinísticos: em sua modelagem não são inseridos componentes de incerteza de nenhuma natureza (AGUIRRE, 2007). Neste tipo de modelo se obtém a mesma resposta para diferentes simulações, se as condições das variáveis de estado e o conjunto de dados utilizado para alimentar o modelo forem os mesmos.

Modelos estocásticos: são modeladas incertezas na forma de variáveis aleatórias, (AGUIRRE, 2007). Neste modelo, mesmo que as condições de uma simulação anterior sejam mantidas, o modelo retornará valores numéricos diferentes, sempre que a semente do gerador de número aleatório for mudada. É importante ressaltar que valores diferentes obtidos numa simulação não significam exatamente resultados diferentes, pois em muitas das vezes se faz necessário recorrer às estatísticas para comprovar a diferença dos resultados.

A caracterização de um modelo é dada em função de como as variáveis de estado de um sistema mudam no tempo. Esta mudança se dará de modo discreta ou contínua. Nos **modelos de mudança discreta:** as variáveis que representam o estado do sistema só mudam em pontos bem definidos de tempo, mesmo que ocorra variação do tempo. Nos modelos de mudança contínua, as variáveis que representam o estado do sistema podem variar continuamente junto com a variação do tempo (FREITAS FILHO, 2008).

Segundo Freitas Filho (2008), os modelos também podem ser classificados em função do seu propósito e tipo de processo decisório envolvido, diferenciando das classificações anteriores que envolviam as características físicas do sistema que

era atribuída ao modelo. Estes podem ser do tipo: **modelos voltados à previsão:** podem utilizar a simulação partindo de algumas suposições do seu comportamento no presente para determinar seu estado em um ponto futuro; **modelos voltados à investigação:** baseia-se em simulações voltadas a levantamento de informações e ou hipóteses sobre a natureza do fenômeno estudado e seu comportamento; **modelos voltados à comparação:** é um dos mais populares. Avalia-se a performance obtida nas mudanças das variáveis de controle, a partir da comparação entre os resultados obtidos em diferentes rodadas de simulação; **modelos específicos:** são projetados e utilizados no auxílio à tomada de decisão em diferentes níveis gerenciais, podendo ser aplicados em situações únicas e específicas dentro de uma empresa e outros .

3.5.3 Plataforma de simulação

O *MATLAB*® e o *Simulink*® são ambientes integrados que permitem a análise, simulação e revisão dos modelos em qualquer ambiente e em qualquer ponto (KARRIS, 2008). Utilizado na simulação de sistemas dinâmicos, o *Simulink*® opera com o *MATLAB*® com o objetivo de permitir a especificação de diferentes sistemas dinâmicos e determinar as suas condições de simulação. Dispõe de uma biblioteca de módulos básicos, utilizados na operação dos diagramas que especificam o sistema (POLLONI et al., 2010).

4 DESENVOLVIMENTO

4.1 MODELO NÃO LINEAR DO PÊNDULO INVERTIDO

O pêndulo invertido foi modelado analiticamente como um sistema dinâmico, determinístico e continuo no tempo. Segundo Ogata (2010) o modelo mecânico do sistema pode ser obtido conforme mostrado na Figura 12.

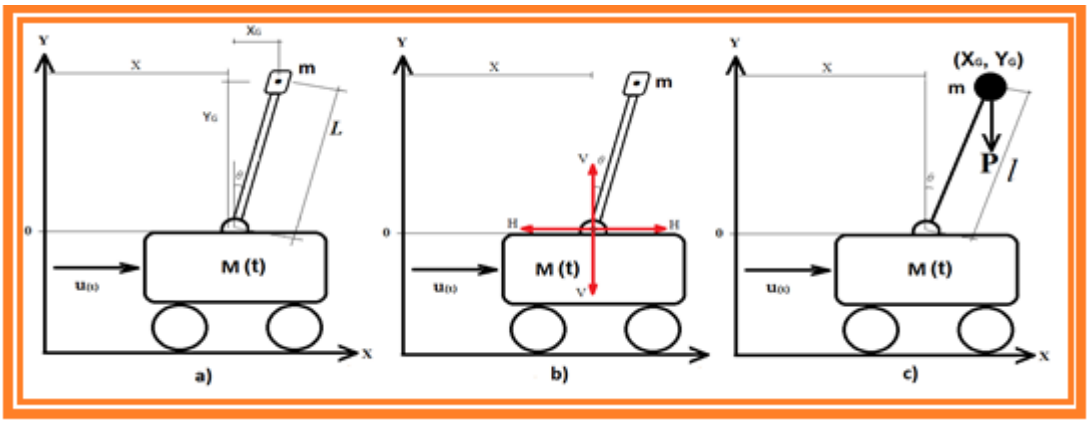


Figura 15: a) Sistema simplificado do pêndulo invertido; b) Representação do movimento vertical e horizontal; c) Centroide da haste.
Fonte: Ogata, (2010, p.61-62). Adaptada pelo autor.

A Figura 15.a mostra o pêndulo com a grandeza força como entrada do sistema u(t) responsável pela movimentação horizontal do corro. Esta força terá sua intensidade variada de acordo com o sinal de controle. A massa do carrinho M(t) é variante no tempo, simulando uma perda de massa. A massa da haste m é considerada constante. A Figura 15.b mostra a força horizontal H e vertical V de contra momento no ponto de apoio da haste. A Figura 15.c mostra a haste sendo

simplificada com a concentração da massa no centro gravitacional que possui comprimento l.

A modelagem do sistema pode ser obtida através das análises do momento angular na haste e da força resultante no carro conforme explica Ogata (2010). Este sistema é classificado como tipo SIMO (*Single Input, Multiple outputs*), em que o vetor de controle dado pela força u(t) é a entrada do sistema e as variáveis de estado ângulo, frequência angular, posição e velocidade linear do carro são as saídas do sistema.

As variáveis presentes no modelo são:

θ - inclinação da haste em relação à linha vertical;

(X_G, Y_G) - Centro de gravidade do pêndulo (*CG*);

ℓ - Distância entre uma extremidade e o centro de gravidade do pêndulo;

M - Massa da plataforma (carro);

m - Massa do pêndulo, concentrada num ponto;

I - Momento de inércia do pêndulo em relação ao *CG*;

u(t) - Força externa, contínua no tempo, aplicada pelo motor à plataforma.

V = força referente ao eixo vertical;

H = força referente ao eixo horizontal;

mg = força peso.

Modelando o sistema obtemos as seguintes equações:

$$M\ddot{x} = u(t) - m\ddot{x} - m\ell\ddot{\theta}\cos\theta + m\dot{\theta}^2\sin\theta \qquad \text{eq. (06)}$$

$$I\ddot{\theta} = m\ell\sin\theta\left[g + \left(\ddot{\theta}\sin\theta + \dot{\theta}^2\cos\theta\right)\right] - \ell\cos\theta\left[m\left(\ddot{x} + \ell\ddot{\theta}\cos\theta - \ell\dot{\theta}^2\sin\theta\right)\right] \qquad \text{eq. (07)}$$

4.2 MODELAGEM NO ESPAÇO DE ESTADOS

Por se tratar de um sistema com quatro variáveis de entrada e saída, consideraremos o nosso vetor x, como:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Onde:

$x_1 = \theta$ (movimentos em função do ângulo) e $x_3 = x$ (movimentos em função da posição)

Considerando:

$\dot{x}_1 = \dot{\theta} = x_2$

$\dot{x}_3 = \dot{x} = x_4$

$\ddot{x}_1 = \ddot{\theta} = \dot{x}_2$

$\ddot{x}_3 = \ddot{x} = \dot{x}_4$

Resumindo, tem-se:

$x_1 = \theta = Posição\ angular$

$x_2 = \dot{\theta} = Frequência\ angular$

$x_3 = x = Posição$

$x_4 = \dot{x} = Velocidade\ linear$

Considerando $I = 0$ e substituindo:

$\ddot{\theta} = \dot{x}_2\ e\ \ddot{x} = \dot{x}_4\ temos,$

$$\dot{x}_2 = \frac{(M+m)*\left(g*sen(x_1)+(x_2)^2*sen(x_1)*cos(x_1)*(1+L)\right)-cos(x_1)*u-\left(m*L*sen(x_1)*cos(x_1)*(x_2)^2\right)}{-M*L*\left(cos(x_1)\right)^2-(M+m)*\left(sen(x_1)\right)^2-L*\left(cos(x_1)\right)^2}$$

eq. (08)

$$\dot{x}_4 = \frac{\left[\frac{U+m*L*sen(x_1)*(x_2)^2}{(M+m)}-m*L*cos(x_1)*\left((M+m)\left(g*sen(x_1)+(x_2)^2*sen(x_1)*cos(x_1)*(1+L)\right)-cos(x_1)*u-m*L*sen(x_1)*cos(x_1)*(x_2)^2\right]}{-m*L*\left(cos(x_1)\right)^2-(M+m)^2*\left(\left(sen(x_1)\right)^2-L*\left(cos(x_1)\right)^2\right)}$$

eq. (09)

Admitindo:

$\dot{x}_1 = x_2\ e\ \dot{x}_3 = x_4$

A representação no espaço de estados e dado pelas seguintes matrizes:

Admitindo:

A matriz de estados **A**.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & a_{42} & 0 & 0 \end{bmatrix}$$

Onde,

$$a_{22} = \left[\frac{\left\{ \left((M+m) * \frac{[g * sen(x_1))}{x_1} + (x_2) * sen(x_1) * cos(x_1) * (1+L) \right] - m * L * sen(x_1) * cos(x_1) * (x_2) \right\}}{\left(-m * L * (cos(x_1))^2 - (M+m) \left[(sen(x_1))^2 - L * (cos(x_1))^2 \right) \right)} \right] \quad \text{eq.(10)}$$

$$a_{42} = \frac{\left(\frac{m * L * sen((x_1)) * x_2}{M+m} - m * L * cos((x_1)) * \left((M+m) \left(g * \frac{sen((x_1))}{x_1} + x_2 * sen((x_1)) * cos((x_1)) * (1+L) \right) - m * L * sen((x_1)) * cos((x_1 \right. \right.}{- m * L * \left(cos((x_1)) \right)^2 - (M+m)^2 * \left(\left(sen((x_1)) \right)^2 - L * \left(cos((x_1)) \right)^2 \right)}$$

$$\text{eq.(11)}$$

Matriz de entrada **B**.

$$B = \begin{bmatrix} 0 \\ b_2 \\ 0 \\ b_4 \end{bmatrix}$$

Onde:

$$b_2 = \frac{-cos(x_1)}{-m * l * (cos(x_1))^2 - (M+m) * \left(\left((sen(x_1))^2 \right) - l * (cos(x_1))^2 \right)} \quad \text{eq.(12)}$$

e,

$$b_4 = \left(\left(\frac{1}{M+m} \right) - \left(\frac{cos(x_1)}{(M+m)*\left(-m*l*(cos(x_1))^2 - (M+m)*\left(\left((sen\,(x_1))^2 \right) - l*(cos(x_1))^2 \right) \right)} \right) \right) \quad \text{eq.}(13)$$

A matriz de saída C.

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A matriz de transmissão direta D.

$$D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Com a representação das matrizes, podemos então formular as equações no espaço de estados e projetar o sistema de controle. A representação no Espaço de Estados na forma canônica é obtida numericamente por uma matriz de transformação **T**. A representação da mesma não será mostrada devido ao tamanho das equações obtidas na transformação.

A função característica do sistema é:

$$S^4 + \left(\frac{mgsen(x_1) + x_2^2 \cos x_1\, sen\,(x_1) + Mgsen\,(x_1) + \ell x_2^2 \cos(x_1) sen\,(x_1) - \ell m x_2^2 \cos(x_1) sen\,(x_1)}{Mx_2 sen(x_1)^2 + mx_2 sen(x_1)^2 - M\ell x_2 cos(x_1)^2} \right) S^3 \quad \text{eq.}(14)$$

Assim o vetor de ganho **K** pode ser calculado com a seguinte equação:

$$K = [\alpha_n - \gamma_n \vdots \alpha_{n-1} - \gamma_{n-1} \vdots \cdots \vdots \alpha_2 - \gamma_2 \vdots \alpha_1 - \gamma_1] T^{-1} \qquad \text{form.}(04)$$

Onde os coeficientes da nova equação característica e representado por α e γ os coeficientes da antiga equação característica.

$$K = [\alpha_n - 0 \vdots \alpha_{n-1} - 0 \vdots \alpha_2 - 0 \vdots$$
$$\alpha_1 - \left(\frac{mgsen(x_1) + x_2^2 \cos x_1\, sen\,(x_1) + Mgsen\,(x_1) + \ell x_2^2 \cos(x_1) sen\,(x_1) - \ell m x_2^2 \cos(x_1) sen\,(x_1)}{Mx_2 sen(x_1)^2 + mx_2 sen(x_1)^2 - M\ell x_2 cos(x_1)^2} \right)] T^{-1} \quad \text{eq.}(15)$$

T é a matriz de transformação de qualquer base, para a base canônica controlável. Se a representação do sistema já estiver na base canônica controlável T será uma matriz identidade.

4.3 PROJETO DO CONTROLADOR ADAPTATIVO

Para projetar um sistema de controle adaptativo é necessário dispor de um mecanismo de checagem do sistema para identificar sua variação, e em conjunto um mecanismo para adequar o sistema de controle a esta nova condição identificada.

4.3.1 Algoritmo de Otimização

A metodologia proposta por Michalewicz (1999), foi utilizada neste trabalho para implementação computacional do AG. Pequenas adaptações para o crossover foram feitas no algoritmo utilizado e mostrado em seu pseudocódigo. Um benefício em trabalhar com AG é que este não depende da obtenção de gradiente para a otimização. Neste trabalho o cálculo do gradiente teria alto custo matemático.

O AG binário utilizado neste estudo possui uma população pré-determinada. Cada indivíduo possui três conjuntos de genes que são responsáveis por três características distintas: o primeiro conjunto de genes representa a característica (valor) da parte real do polo complexo '**a**', o segundo conjunto de genes representa a característica (valor) da parte imaginária '**b**' e o terceiro conjunto representa a característica (valor) do polo real duplo '**c**', como mostrado na Figura 16, estas características são geradas aleatoriamente, dentro de um intervalo de valores pré-definidos.

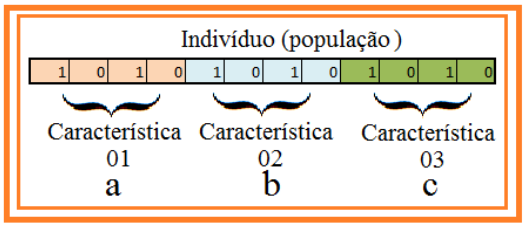


Figura 16: Cromossomo representado na base binária.
Fonte: Próprio Autor.

O grande desafio para projetar um controlador no espaço de estados é definir o local dos polos que atenda os requisitos de projeto, como observado no trabalho de Pasemann (1998), onde optou-se por utilizar uma RNA para balanceamento (alocação) dos polos no projeto do controlador de um pêndulo invertido com massa constante. Neste trabalho, um AG é utilizado ao invés de uma RNA.

A sintonia do controlador é dada pelos novos ganhos do vetor de controle u(t). Para encontrar as novas alocações de polos e consequentemente um novo vetor de ganho, o Algoritmo Genético é utilizado para determinar os valores dos polos complexos conjugados e dos dois polos reais. Para simplificar a busca por otimização do AG, serão determinados polos reais iguais (polos reais duplos). O controle do pêndulo é feito na posição vertical (ângulo igual à zero) e posição horizontal igual à zero.

A Figura 16 ‘a’ tem representação binária 10101 e valor inteiro é 21, ‘b’ tem representação binária 01010 e valor inteiro é 10 e ‘c’ tem representação binária 10101 e valor inteiro é 21. A representação dos polos nesta ilustração seria P=(S+21+10i)(S+21-10i)(S+21)(S+21), após a representação, com a nova alocação dos polos, o fitness é calculado.

Assim a função característica P com a nova alocação dos polos, pode ser descrita da seguinte forma:

$$P = (S + a + bi)\ (S + a - bi)\ (S + c)\ (S + c) \qquad \text{eq.}(16)$$

$$P = S^4 + (2a+2c)\ S^3 + (a^2+4ac+b^2+c^2)\ S^2 + [2ac^2 + (2a^2+2b^2)\ c)]\ S + c^2(a^2+b^2) \qquad \text{eq.}(17)$$

E a nova alocação dos polos dada por:

$$P = (s-\mu_1)(s-\mu_2)\ldots(s-\mu_n) = s^n + \alpha_1 s^{n-1} + \ldots + \alpha_{n-1} s + \alpha_n \qquad \text{eq.}(18)$$

$\alpha_1 = (2a+2c);\ \alpha_2 = (a^2+4ac+b^2+c^2);\ \alpha_3 = [2ac^2 + (2a^2+2b^2)c];\ \alpha_4 = c^2(a^2+b^2)$

Cabe lembrar que o vetor de controle será:

$$K = [\alpha_n - 0 \vdots \alpha_{n-1} - 0 \vdots \alpha_2 - 0 \vdots \alpha_1 - (\frac{mgsen(x_1)+x_2^2 \cos x_1 \, sen\,(x_1)+Mgsen(x_1)+\ell x_2^2 \cos(x_1) sen\,(x_1)-\ell m x_2^2 \cos(x_1) sen\,(x_1)}{Mx_2 sen(x_1)^2+mx_2 sen(x_1)^2-M\ell x_2 cos(x_1)^2})]T^{-1} \quad \text{eq.(15)}$$

Como a massa do sistema varia com o tempo M(t), em conjunto com os critérios de tempo de acomodação e sobressinal, novas alocações para os polos precisam ser determinados para satisfazer os critérios. Assim, os coeficientes da função da nova alocação dos polos α_n também variam com a nova alocação. Para solucionar este problema de variação nas condições de operação e variação dos parâmetros, um controle adaptativo é empregado.

Com o vetor de ganho definido, é aplicado um distúrbio padrão no sistema através da equação de estado com a nova alocação de polos.

$$\dot{x} = (A - Bk)x \qquad \text{eq.(05)}$$

$$Y = CX \qquad \text{eq.(04)}$$

O *fitness* é calculado através dos valores do sobressinal (mp) e tempo de acomodação (ts) obtidos para o ângulo, relacionando-os com os requisitos (Mp e Ts).

$$\mathbf{fit = [(|mp-Mp|)10]^5 + [(|ts-Ts|)4]^5} \qquad \text{form.(05)}$$

Considera-se uma solução aceitável quando os polos fornecem uma diferença de sobressinal de **0,1** ([(|mp-Mp| ≤ 0,1) e tempo de acomodação de **0,25** segundos ([(|ts-Ts| ≤ 0,25). Na equação do *fitness* na primeira parte [(|mp-Mp|)10]^5 aplicando a teoria de limite, quando |mp-Mp| = 0,1, [(|mp-Mp|)10]^5 = 1; para |mp-Mp| > 0,1 ou para |mp-Mp| < 0,1 tende a ampliar o valor ([(|mp-Mp|)10]^5 > 1), para |mp-Mp| = 0, [(|mp-Mp|)10]^5 = 0. A mesma operação é aplicada na segunda metade da equação do *fitness* [(|ts-Ts|)4]^5. Assim consegue-se evidenciar o indivíduo que não atende simultaneamente as duas condições. O expoente com valor igual a cinco, foi escolhido de modo empírico, após apresentar a distinção entre individuas que tinham uma diferença de 10% dentro da tolerância e 10% fora desta (fit = 0,9^5 + 1,1^5 = 2,20), atribuindo-lhes uma penalização estipulada pelo autor de 10%, ou seja, a soma dos dois parâmetros seria 0,90+1,10=2 acrescentando a penalização 2,20, resultado próximo ao obtido com expoente igual a cinco. A tabela 10 compara estes valores.

Tabela 10: Diferença dos parâmetros com penalização.

Diferenças entre parâmetros		**Expoente**				
mp-MP	ts-TS	2	3	4	5	6
0,90	1,10	2,02	2,06	2,12	2,20	2,30
1,00	1,00	2,00	2,00	2,00	2,00	2,00

Fonte: próprio autor.

O indivíduo que obtiver fit = 0 é considerado o indivíduo perfeito é a melhor solução, mas não é única, pois pode existir vários indivíduos perfeitos numa mesma população, se 0 < fit ≤ 1 indivíduo atende as restrições (fornece solução aceitável, garantia de solução), se 1 < fit ≤ 2 o indivíduo tem a possibilidade de atender as restrições (indivíduos com potencial de solução, mas não tem garantia de solução, pois neste caso o indivíduo pode atender apenas uma das restrições), os indivíduos com fit >2 não atendem as restrições. A função de custo foi concebida de maneira que caracteriza problema de otimização como problema de minimização.

Neste AG as gerações são criadas a partir do cruzamento correspondente a 25% dos melhores indivíduos da população (*fitness* mais baixo). A estratégia utilizada é mostrada na Figura 17.

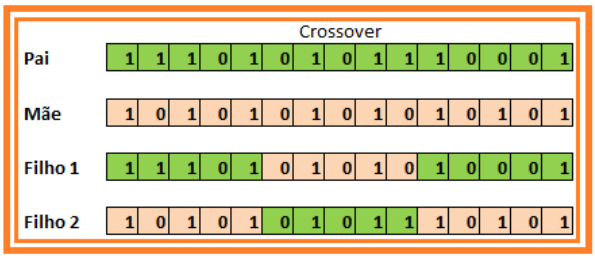


Figura 17: Geração de descendentes por crossover.
Fonte: Próprio Autor.

Na Figura 17, o cruzamento ocorre da seguinte maneira: divide-se o cromossomo em três partes, onde um descendente recebe duas partes do pai e uma da mãe e o segundo recebe duas da mãe e uma parte do pai.

A mutação é aplicada em 5% dos indivíduos da população global através do método de roleta (valor arbitrado), onde um gene é modificado de uma posição aleatória do cromossomo.

O número de gerações também é pré-determinado. O AG utilizado neste trabalho tem o seguinte pseudocódigo:

1- **Gera população** (aleatoriamente num intervalo pré-estabelecido)

2- **Atribui *fitness*** individual e calcula o *fitness* global
3- **Aplica o Crossover**
4- **Aplica a Mutação**
5- **Atualiza *fitness*** individual e global
6- **Seleciona melhor indivíduo**
7- **Se** condição de n° de gerações não atendida e não houver indivíduo(s) com *fitness* menor que 1, retorna ao passo **3**.
8- **Fim**

4.3.2 Projeto da Rede Neural

Projeto 1

Para configurar a RNA, foi utilizado o mesmo AG citado anteriormente. O AG retorna três valores **a**, **b** e **c**. A característica "**a"** representa a quantidade de camadas ocultas com a função de ativação *tansig*; "**b**" representa a quantidade de camadas ocultas com a função de ativação *radbas* e "**c**", a quantidade de camadas ocultas com a função de ativação *purelin*. O número de neurônios (**n**) em cada camada oculta está relacionado à quantidade de camadas ocultas, tal que **n= 2(característica)+5**, sendo o número mínimo de neurônios em uma camada igual a 7 quando se tem uma camada oculta referente a uma função de ativação, e o número máximo igual a 19 para 7 camadas ocultas para uma função de ativação. Se **a**, **b** e **c** igual a 7, teremos 21 camadas ocultas, cada uma com 19 neurônios.

A RNA utilizada neste segundo projeto é do tipo *feedforward*. Em seu treinamento utiliza-se o algoritmo de aprendizagem *Levemberg- Marquartd*, onde os valores de pesos e bias iniciais são atribuídos aleatoriamente. A rede possui três camadas de entrada (ts, mp, M) e três camadas de saída (a, b, c). A função de custo retorna a performance obtida no trino da RNA.

O número máximo de iterações foi estipulado em 3000, gradiente mínimo de 10^{-20}, falha na validação e checagem aceitável igual a 50, o desempenho da rede é calculado através do erro médio quadrático (*mse*).

Projeto 2

Um segundo AG foi utilizado para configuração da RNA. Este pode ser acessado pelo comando do *Matlab* ga(@função_de_avaliação, numero_variáveis, A,b), onde A e b são matrizes das inequações (**A≤ b**), estas matrizes determinam a range das variáveis. A função de avaliação possui nove parâmetros a, b, c, A, B, C, α, β, γ.

Os parâmetros a, b e c são responsáveis pela quantidade de camadas ocultas. A, B e C são parâmetros responsáveis pelas funções de ativação que serão utilizadas nas camadas ocultas (a, b e c), que poderão ser: tangente sigmoide, base radial, linear e função logarítmica sigmoide. O número de neurônios de cada camada é definido pelos parâmetros α, β e γ.

Este algoritmo é iniciado com uma população de 20 indivíduos, a nova população será gerada com 80% dos indivíduos provenientes de crossover. A mutação aleatória é aplicada num valor de 1 de desvio padrão, utilizando uma curva Gaussiana na população inicial, este valor decresce linearmente para 0 ao final das gerações. Dois melhores indivíduos são passados para nova população utilizando a estratégia de elitismo. Ocorre uma migração de 20% da população antiga para a nova população. O número máximo de gerações foi limitado em 100. A função de custo retorna a performance obtida no treino da RNA. A configuração do AG pode ser visualizada inserindo o comando *options* = *gaoptimset*(@ga) no *workspace*.

Os parâmetros da RNA não foram alterados, sendo iguais para os dois projetos.

4.3.3 Projeto do Ganho Agendado

No projeto do controlador adaptativo por ganho agendado, é utilizada uma RNA, que recebe as variações de tempo de acomodação, sobressinal e massa e aloca os polos para o controle por realimentação de estados.

4.4 SIMULAÇÃO

4.4.1 Configuração do controlador

A seguir são dadas as configurações utilizadas na simulação com o AG na busca dos polos.

1- O intervalo inicial de ‘a’ ‘b’ e ‘c’ varia de 0,1 a 150,0 com precisão de três casas decimais.

2- De acordo com Aguirre (2007), bons resultados são obtidos quando o periodo de amostragem é escolhido com valores entre um décimo e um terço da menor constante de tempo do sistema. O valor de c máximo encontrado foi de 10,061, logo, a constante de tempo é de 0,0994s. Nos testes realizados constatou-se que para um periodo de amostragem maior que 0,01, obtinha-se tempos de acomodação e sobresinais distintos com variação em 0,1, e para valores menores que 0,01 estas variações eram menores que 0,001. Neste trabalho o período de amostragem foi aproximado pra 0,01s, uma vez que os critérios de tempo de acomodação e sobresinal fora estipulados com 2 algarismo significativo.

3- Após encontrar o primeiro indivíduo que satisfaça as restrições de prejeto, este é utilizado como ponto para formar uma região de busca local para os novos indivíduos com limites superior e inferior igual a 10. Se o primeiro encontrado é [5,5701, 149,8177, 80,0921], a nova região de busca será: (5,5701-10) ≤ a ≤ (5,5701+10), (149,8177-10) ≤ b ≤ (149,8177+10) e (80,0921-10) ≤ c ≤ (80,0921+10). Para resultados de limite inferior menor que zero (5,5701-10), este é fixado em zero. Com esta estratégia a região de busca torna-se variável.

4- Determinou-se uma população inicial de 125 indivíduos, um limite de 50 gerações para a primeira busca, visto que a busca ocorre numa região maior (global), e 15 gerações para as demais, pois as buscas passam a ser local.

5- As variações de tempo de acomodação, sobressinal e massa são representadas pelos blocos do simulink Matlab, um ambiente de simulação dinâmica, na Figura 18.

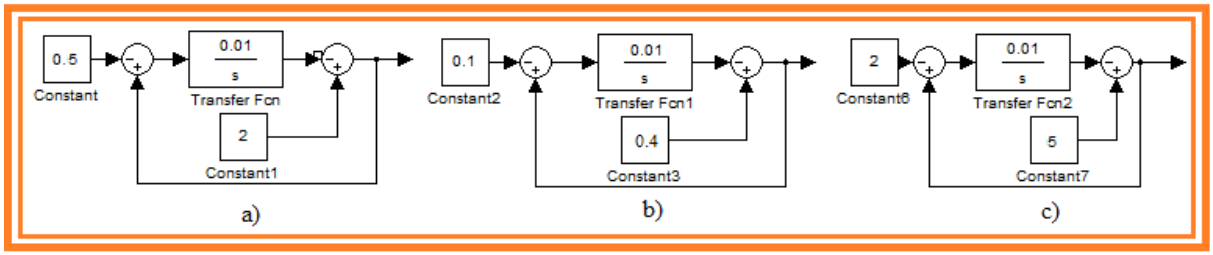


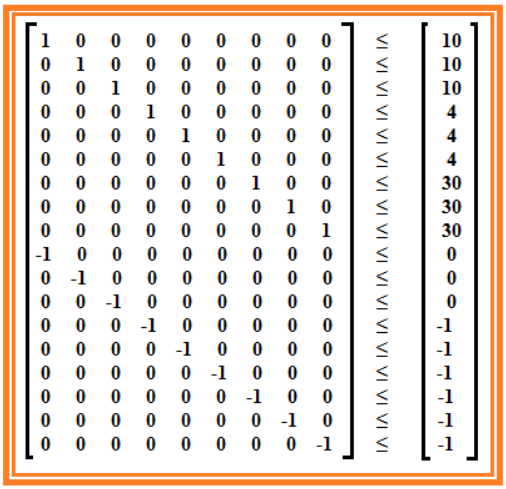
Figura 18: a) Tempo de acomodação b) Sobressinal c) Massa.
Fonte: *Simulink, Matlab* 2012b.

Na Figura 18.a o tempo de acomodação tem valor inicial de 2,0s e fica mais rígido com o passar do tempo, estabilizando próximo do valor de 0,5s. Na Figura 18.b o valor inicial do sobressinal é igual a 0,4 no início da simulação (requisito inicial do sistema), tende a 0,1 para tempos maiores que 600s. A Figura 18.c mostra a variação da massa do carro ao decorrer da simulação representada pelo diagrama de blocos. Este começa com valor de massa inicial de 5,0kg, e tende a se estabilizar no valor de 2,0kg após um intervalo de tempo de 600s.

4.4.2 Configuração do Projeto da RNA

No projeto 1 adotou-se as seguintes configurações: 0≤ **a** ≤ 7, 0≤ **b** ≤ 7, 0≤ **c** ≤ 7. Estas variáveis possuem valores inteiros, portanto ajustou-se o AG para 0, casa decimal de precisão. Optou-se por limitar o tamanho da RNA entre 20 a 30 camadas ocultas devido ao custo computacional requerido. A população inicial foi de 64, um limite de gerações foi definido igual a 5, taxa de crossover e mutação são as mesmas adotadas no projeto de alocação de polos. Não foi utilizada a estratégia de elitismo neste AG.

No projeto 2 utilizou-se a seguinte configuração: a Figura 19 mostra as matrizes **A** e **b**.



$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{bmatrix} \leq \begin{bmatrix}
10 \\ 10 \\ 10 \\ 4 \\ 4 \\ 4 \\ 30 \\ 30 \\ 30 \\ 0 \\ 0 \\ 0 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1
\end{bmatrix}$$

Figura 19: Matriz A e b de inequações.
Fonte: Próprio Autor.

A função de avaliação deste AG possui nove variáveis (a,b,c,A,B,C,α,β,γ), o intervalo destas são definidos pelas matrizes de inequação, 0≤ **a** ≤ 10, 0≤ **b** ≤ 10, 0≤ **c** ≤ 10, 1≤ **A** ≤ 4, 1≤ **B** ≤ 4, 1≤ **C** ≤ 4, 1≤ **α** ≤ 30, 1≤ **β** ≤ 30, 1≤ **γ** ≤ 30. Estas variáveis trabalham somente com valores inteiros, portanto ajustou-se o AG para 0 casa decimal de precisão.

Para treino nos projetos 1 e 2 das RNA's foi utilizado o mesmo conjunto de polos encontrados pelo AG.

5 RESULTADOS E DISCUSSÕES

Com o AG, montou-se uma tabela de polos para o ganho agendado (implementado por meio de uma RNA) que atende as restrições de projeto.

A Tabela 11 foi construída após uma otimização. A menor tabela que pode ser construída com os dados dos Ganhos Agendados é uma tabela que contenha somente indivíduos perfeitos. A tabela utilizada para treino foi a menor encontrada, porém não há garantia de ser a menor tabela que pode ser construída atendendo as restrições. Os polos utilizados no aprendizado são mostrados na Tabela 11.

Tabela 11: Polos utilizados no Treinamento da RNA.

A	b	c	Ts			mp			M		
5,764	67,188	2,151	1,830	**≤ ts ≤**	2,000	0,366	**≤ ts ≤**	0,400	4,661	**≤ ts ≤**	5,000
6,296	68,639	10,061	1,657	**≤ ts <**	1,830	0,331	**≤ ts <**	0,366	4,313	**≤ ts <**	4,661
6,218	67,345	0,453	1,634	**≤ ts <**	1,657	0,327	**≤ ts <**	0,331	4,267	**≤ ts <**	4,313
7,551	61,267	3,708	1,410	**≤ ts <**	1,634	0,282	**≤ ts <**	0,327	3,820	**≤ ts <**	4,267
8,884	52,286	1,120	1,245	**≤ ts <**	1,410	0,249	**≤ ts <**	0,282	3,490	**≤ ts <**	3,820
12,806	54,835	2,375	1,128	**≤ ts <**	1,245	0,226	**≤ ts <**	0,249	3,257	**≤ ts <**	3,490
20,531	47,345	2,845	0,818	**≤ ts <**	1,128	0,164	**≤ ts <**	0,226	2,637	**≤ ts <**	3,257
24,257	52,482	1,512	0,695	**≤ ts <**	0,818	0,139	**≤ ts <**	0,164	2,390	**≤ ts <**	2,637
30,884	54,875	0,806	0,550	**≤ ts <**	0,695	0,110	**≤ ts <**	0,139	2,100	**≤ ts <**	2,390
26,610	48,012	0,178	0,500	**≤ ts <**	0,550	0,100	**≤ ts <**	0,110	2,000	**≤ ts <**	2,100

Fonte: Elaborada pelo autor.

O AG utilizado no **projeto 1** retornou à seguinte configuração de RNA: 7 camadas ocultas, 19 neurônios em cada camada oculta, todas com a função de ativação tangente sigmoide.

O erro médio quadrático (*fitness*) encontrado após o treino foi igual a 0,0697. A Tabela 12 mostra a evolução do erro, até encontrar o melhor indivíduo ***n***.

Tabela 12: Evolução dos melhores indivíduos.

Indivíduo	**n-10**	**n-9**	**n-8**	**n-7**	**n-6**	**n-5**	**n-4**	**n-3**	**n-2**	**n-1**	**n**
Erro	22,71	2,41	1,66	1,51	1,33	1,30	1,09	1,00	0,99	0,96	0,07

Fonte: Elaborada pelo autor.

A Figura 20 mostra a evolução dos melhores indivíduos através do gráfico.

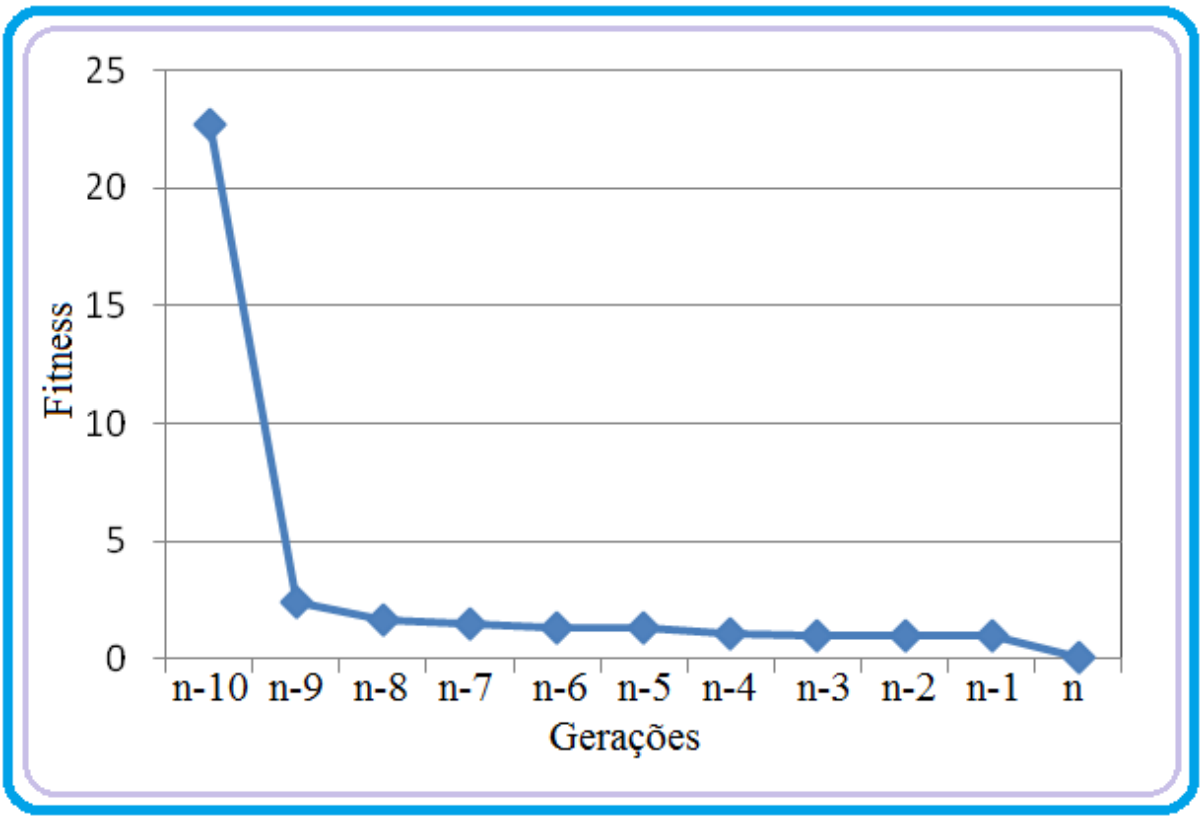


Figura 20: Gráfico da evolução dos melhores indivíduos.
Fonte: Elaborado pelo autor.

O AG utilizado no **projeto 2** retornou a seguinte configuração de RNA: 5 camadas ocultas com 5 neurônios em cada camada, com a função de ativação tangente sigmoide.

A configuração do melhor indivíduo encontrado neste projeto é mostrada no Quadro 2.

Quadro 2: Configuração do melhor indivíduo.

Camadas Ocultas			Função de Ativação			N° de Neurônios		
Camada 1	Camada 2	Camada 3	Camada 1	Camada 2	Camada 3	Camada 1	Camada 2	Camada 3
5	0	0	1	2	3	5	3	4

Fonte: Elaborada pelo autor.

O indivíduo do Quadro 2 possui um total de 5 camadas ocultas (5+0+0), composto pela função de ativação 1 (tangente sigmoide), o primeiro conjunto de camadas possui 5 neurônios, como as camadas 2 e 3 são iguais a zero, as outras características tornam-se sem efeito (células em vermelho).

O erro médio quadrático encontrado após o treinamento foi igual a 0,8704.

Para Construir o controlador Adaptativo por ganho Agendado escolheu-se a configuração do projeto 1, por ter retornado o menor erro. Para melhorar o desempenho no treinamento da RNA, aumentou-se a matriz de dados utilizada no treinamento, onde a nova matriz passa a ter dados repetidos. Os dados para o treinamento foram tratados de modo que, cada entrada distinta possui saída singular, garantindo o ineditismo dos dados utilizados na aprendizagem, não foi necessário inserir atraso nos dados para treinamento da rede por não ser influenciada pela taxa de variação das entradas. A função que a RNA aproxima é uma função do tipo *f(ts,mp,M),* onde *ts(t), mp(t)* e *M(t),* assim a rede varia em função dos requisitos de operação do sistema e da massa. A realimentação da rede não foi necessária, pois nenhuma relação anterior, e (ou) velocidade interferem na alocação dos novos polos do sistema.

O erro médio quadrático retornado após o treino foi igual a 0,0340, em 148 iterações. A matriz de pesos e bias pode ser extraída da rede através do comando *getwb(net),* a matriz extraída da RNA após o treino possui a dimensão de 2416 linhas e 1 coluna, esta não foi apresentada neste trabalho em virtude do seu tamanho. Para gerar o bloco da RNA no *simulink* após o treino, utiliza-se o comando *gensim(net)*.

Os resultados obtidos na plataforma de simulação com o controlador adaptativo em relação a variável de estado ângulo pode ser visualizado através das Figuras 21, 22, e 23.

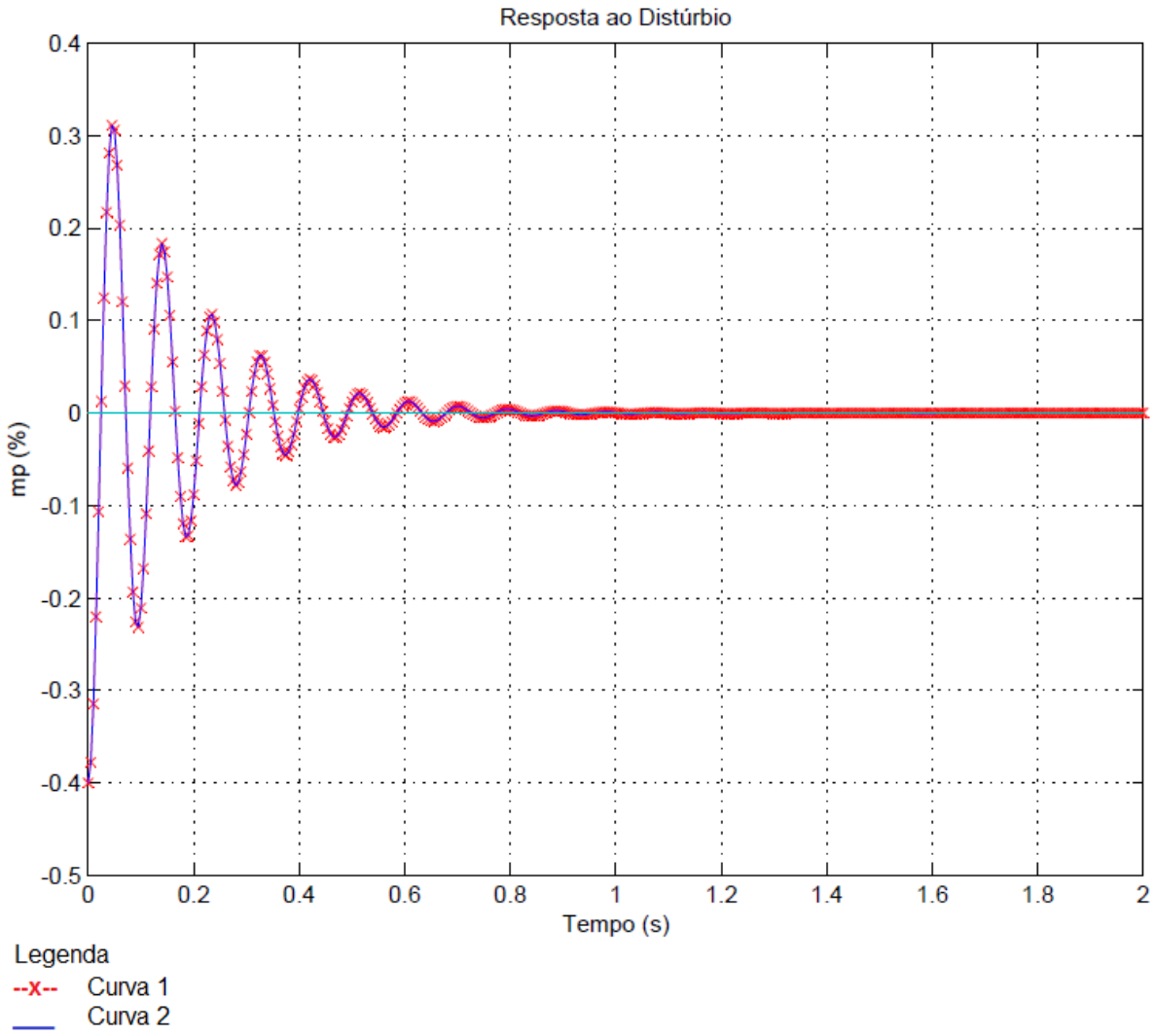


Figura 21: Gráfico da resposta do sistema ao distúrbio.
Fonte: Elaborado pelo autor.

O gráfico da Figura 21 representa a resposta do sistema ao distúrbio pra os requisitos ts=2,000, mp=0,400 e M=5,000:

Resposta com polos (5,7639; 67,1883; 2,1507) alocados com o AG, retorna ts=2,08, mp=0,311 e *fitness*=0,558 (curva 1).

Resposta com polos (5,7741; 67,2108; 2,0849) alocados com o Controlador com Ganho Agendado (RNA), retorna ts=2,06, mp=0,308 e *fitness*=0,565 (curva 2). As duas curvas estão no mesmo gráfico da Figura 21, porém por não haver diferença significativa estas se sobrepõem.

No gráfico da Figura 22, representa a resposta do sistema ao distúrbio pra os requisitos ts=1,2524, mp=0,2505 e M=3,5047.

Resposta com polos (8,8843; 52,2863; 1,1196) alocados com o AG, retorna ts=1,260, mp=0,424 e *fitness*=0,843, curva 1. Resposta com polos (8,7162; 52,2287; 1,2708)

alocados com o Controlador com Ganho Agendado (RNA), retorna ts=1,440, mp=0,352 e *fitness*=1,305, curva 2.

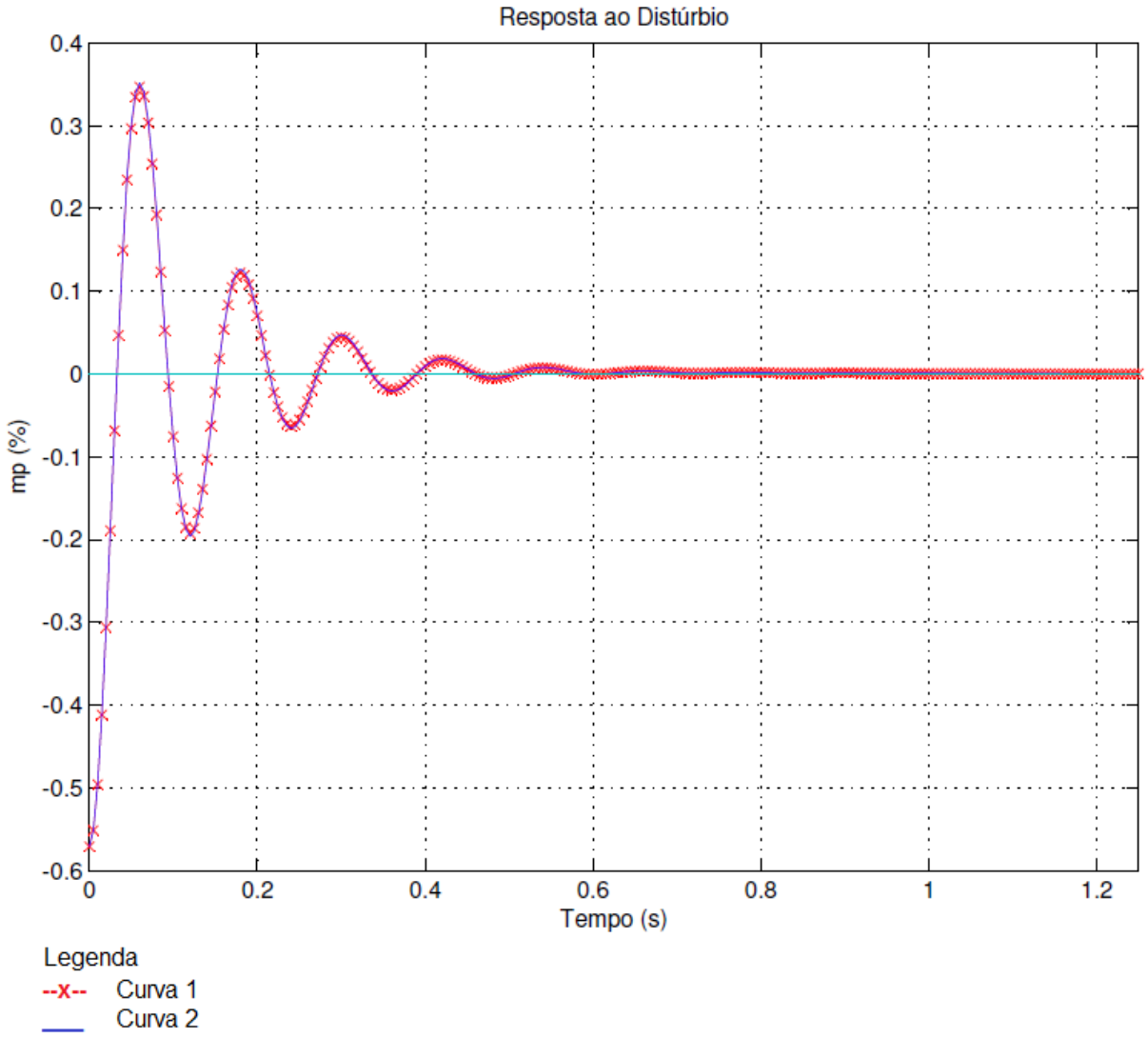


Figura 22*:* Gráfico da resposta do sistema ao distúrbio, após variações das variáveis de estado.
Fonte: Elaborado pelo autor.

Ao final da simulação, obtém-se o gráfico da Figura 23, representa a resposta do sistema ao distúrbio pra os requisitos ts= 0,5042, mp= 0,1008 e M= 2,0085.

Resposta com polos (26,6098; 48,0118; 0,1784) alocados com o AG, retorna ts=0,560, mp=0,182 e *fitness*=0,350, curva 1.

Resposta com polos (26,6098; 48,0118; 0,1784) alocados com o Controlador com Ganho Agendado (RNA), retorna ts=0,560, mp=0,183 e *fitness*=0,373, curva 2.

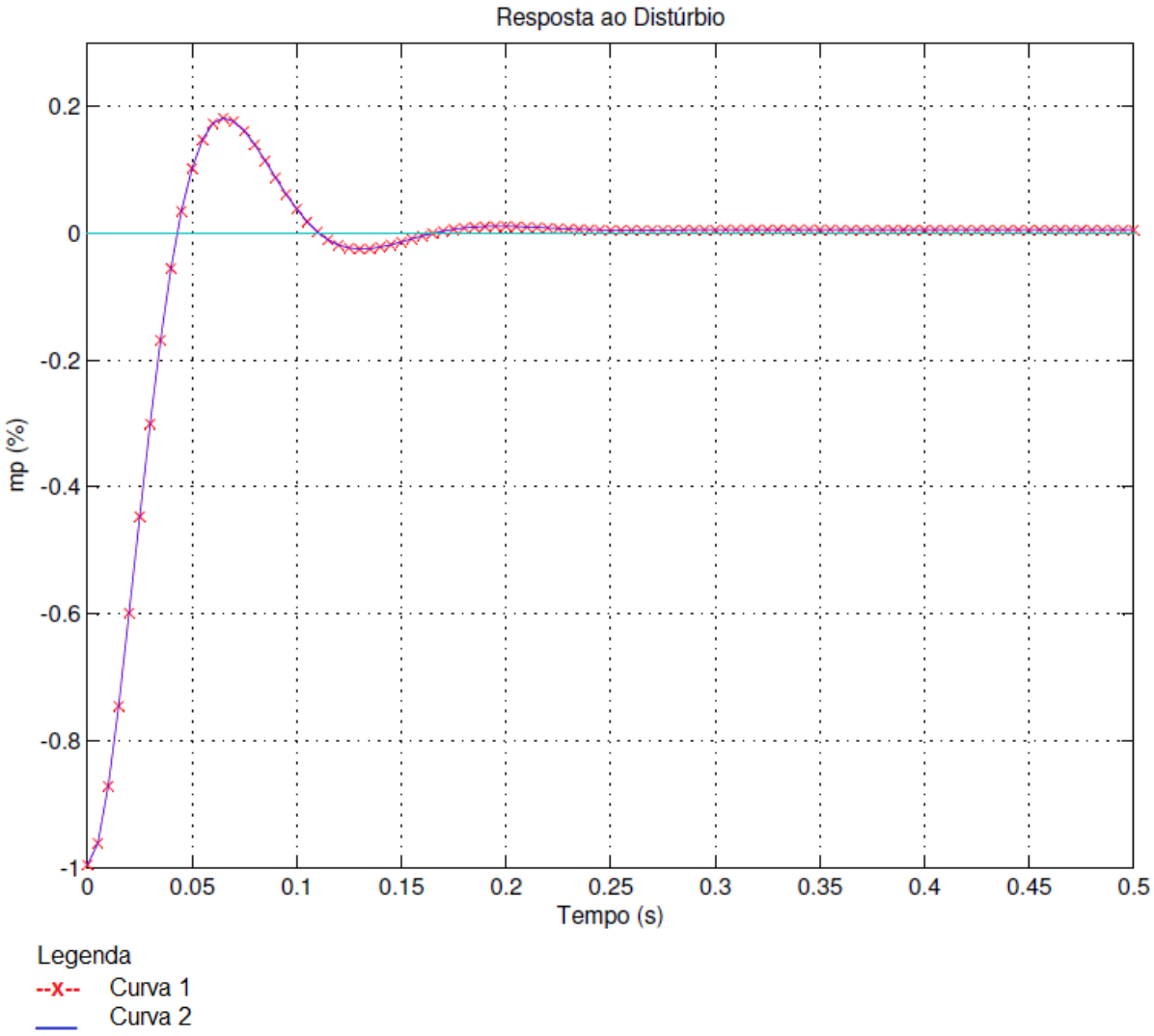


Figura 23: Gráfico da resposta do sistema ao distúrbio, ao final da simulação.
Fonte: Elaborado pelo autor.

A simulação foi construída na plataforma de simulação contínua *Simulink Matlab*. A Figura 24 mostra a configuração do modelo através da linguagem de diagramas de blocos.

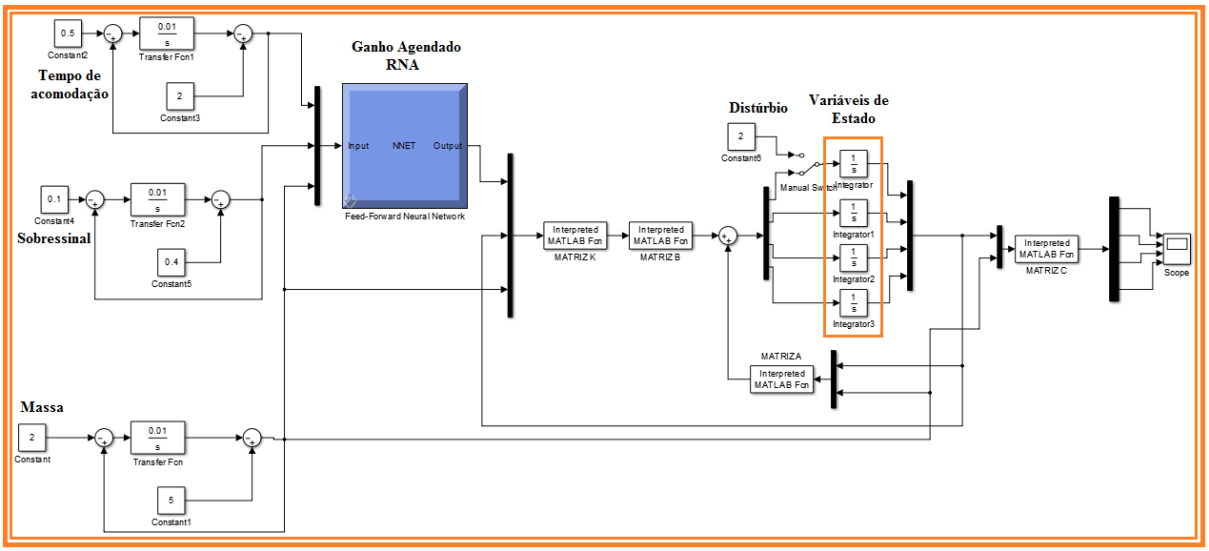


Figura 24: Sistema representado por diagrama de blocos.
Fonte: *Simulink, Matlab* 2012b.

Na Figura 24, tem-se o controle por ganho agendado através da RNA, blocos responsáveis por variar o tempo de acomodação, sobressinal e massa. Os integradores representam as variáveis de estado e uma chave para inserir o distúrbio no sistema;

A Figura 25 apresenta a animação feita no *wonderware Intouch* versão acadêmica.

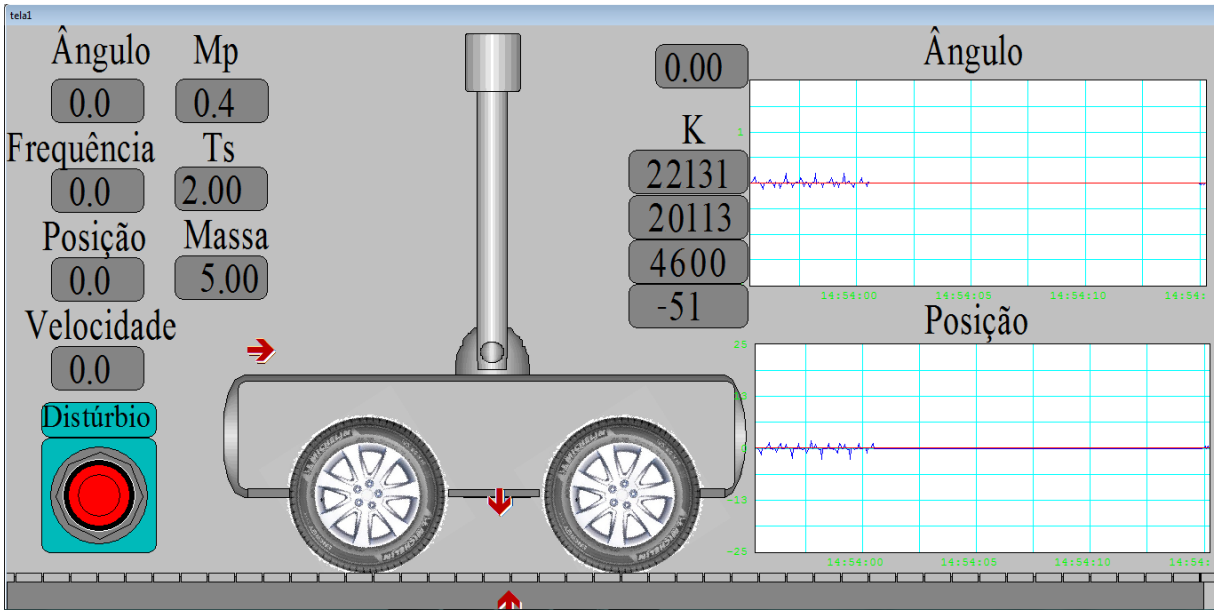


Figura 25: Animação do pêndulo invertido.
Fonte: *Wonderware Intouch 9.5*, versão acadêmica.

Para validação e verificação do modelo, utilizaram-se dados com respostas conhecidas para alimentação do modelo. A animação em 2D também foi utilizada

como ferramenta de verificação, pois segundo Freitas Filho (2008) esta ferramenta empregada no estudo simulado permite a percepção de comportamentos sutis no modelo em que no sistema real poderia passar desapercebido.

A Figura 25 mostra a tela com a animação em 2D do simulador do pêndulo invertido. Nesta tela pode-se acompanhar a variação de massa do sistema, a mudança dos requisitos sobressinal Mp e tempo de acomodação Ts, valores instantâneos de ângulo de inclinação da haste, frequência angular, posição, velocidade linear do carro, gráficos de tendência real e a dinâmica do carro pêndulo . A seta horizontal vermelha presente na Figura 25 do simulador representa o sentido em que a força está atuando no carro, a seta vertical superior indica a posição do carro e a seta vertical inferior indica o referencial zero horizontal. Quando estas duas setas se alinham, significa que o carro se encontra na posição zero.

Durante a simulação, é possível interagir com o sistema através do botão com título distúrbio. Quando utilizado, este muda sua cor de vermelho, Figura 25, para verde, apresentado na Figura 26.

Figura 26: Botão para inserção de distúrbios.
Fonte: *Wonderware Intouch 9.5*, versão acadêmica.

O distúrbio inserido no sistema é transitório para que o pêndulo volte para o equilíbrio. Na Figura 25, o gráfico ângulo representa a posição angular (dado em grau) com escala de -2 a 2, em relação ao tempo (dado em segundo). O gráfico com título Posição descreve a posição (dado em metro) com escala de -25 a 25, em relação ao tempo (dado em segundo).

CONSIDERAÇÕES FINAIS

Neste capítulo são apresentadas as conclusões dos resultados obtidos e sugestões para trabalhos futuros.

6.1 CONCLUSÃO

Os resultados obtidos pelo AG na alocação de polos se mostraram viáveis, uma vez que atenderam os critérios impostos no inicio do trabalho como: variação entre o sobressinal obtido e desejado menor igual a 0,1 e variação do tempo de acomodação desejado e obtido menor igual a 0,25s. Mesmo apresentando uma região de busca não linear e com certo grau de complexidade, a construção da função de avaliação é relativamente simples, o resultado é obtido num tempo satisfatório para a otimização *off-line*.

Sem a necessidade dos novos polos encontrados que atendam as novas condições de operações estarem próximos dos polos anteriores, a estratégia com busca global seguida de buscas locais se mostrou eficiente, retornando uma solução num menor intervalo de tempo quando comparado a implementação, que utilizava somente buscas globais, bastando fornecer o melhor indivíduo para estabelecer uma nova e menor região de busca (elitismo), em casos de ótimos locais sem atender os requisitos uma busca global era aplicada.

O AG também mostrou-se satisfatório na configuração da RNA, mesmo não tendo a garantia desta ser a melhor configuração, o método permite encontrar uma

configuração que atenda os objetivos com o menor erro após o treino das arquiteturas analisadas. Desta forma pode-se obter uma configuração sem que fosse fundamental o conhecimento de um especialista em relação às funções de ativação, a quantidade de camadas ocultas e o número de neurônios destas camadas para configurar a RNA.

A RNA não foi utilizada como controlador pela complexidade da massa de dados que esta exigiria, pois a superfície que descreve o comportamento do sinal da ação de controle para determinado distúrbio, combinado com mudança de massa, tempo de acomodação e sobressinal é de extrema complexidade exigindo uma grande quantidade de camadas ocultas e um grande número de neurônios nestas camadas para minimização do erro, algo que demanda muita capacidade de processamento, apresentando um elevado tempo de resposta, inviabilizando a simulação devido ao tempo de resposta da RNA ser maior que o período mínimo de amostragem do sistema.

A RNA não foi utilizada para interpolação na alocação dos polos, por exigir um intenso mapeamento para gerar uma tabela de dados capaz de representar significativamente a superfície do modelo com as mudanças das condições de operação. Na tentativa de interpolar polos com distâncias significativas entre si, para atender diferentes condições de operação (sobressinal e tempo de acomodação), este retornará um resultado insatisfatório com condições de operação diferentes do requerido para a massa de dados utilizada neste trabalho.

Embora tenha sido considerado satisfatório o resultado obtido pelo AG na configuração da RNA, tendo retornado a configuração de rede com o menor erro dentre as configurações presentes na otimização, o comando utilizado para a aplicação do algoritmo de aprendizagem *train(net,x,y)* aloca inicialmente um valor aleatório de pesos e bias, podendo retornar um valor significativamente diferente entre dois treinos consecutivos com o mesmo tipo de RNA e com as mesmas configurações, dificultando a avaliação de aptidão de indivíduos iguais ou similares.

O controle agendado apresentou bom desempenho com a alocação fixa dos polos, pois todos os polos apresentados atendiam os critérios de sobressinal e tempo de acomodação estabelecidos no início deste trabalho. Também comportando-se diferentemente de quando se tem um ganho fixo para um modelo linear, quando a estratégia de ganho fixo é aplicada a um modelo não linear, não consegue manter o melhor desempenho estipulado no projeto, pois o método

consiste na alocação dos polos e este varia com o ganho constante devido à variação angular. No controlador desenvolvido os polos alocados não variam com a variação angular, reproduzindo com melhor precisão o estipulado em projeto.

Mesmo que a solução em Espaços de Estados seja baseada em sistemas lineares, a solução deste modelo baseia-se na linearização, em cada instante de operação aumentando sua eficiência, apresentando uma solução numérica a para cada posição angular.

6.2 SUGESTÕES PARA TRABALHOS FUTUROS

Para projetos que objetivam a implementação num sistema real, algumas considerações devem ser feitas como: extrair um modelo com maior precisão do sistema levando em consideração momento de inércia, coeficiente de atrito cinético e dinâmico. A função de avaliação deve ter uma penalização para indivíduos que apresentem um sinal de controle superior ao valor máximo disponibilizado pelo atuador.

Num conjunto de soluções que atendam as restrições de projeto, pode-se otimizar o que consuma menor energia.

Simular o controle adaptativo com inteligência artificial otimizado num pêndulo invertido com dois graus de liberdade.

Construir um pêndulo invertido utilizando a placa microcontrolada Arduino com sistema de controle embarcado, visto que, muitas aplicações tem sido realizadas utilizando esta ferramenta de software livre.

Aplicar uma otimização *on-line* para o sistema pêndulo invertido, com variação lenta dos requisitos de operação e massa ao longo do tempo.

REFERÊNCIAS BIBLIOGRÁFICAS.

AGUIRRE, L. A**. Introdução a Identificação de Sistemas: técnicas lineares e não lineares aplicadas a sistema real**. 3. ed. Belo Horizonte: UFMG, 2007.

ASTROM, K. J.; WITTENMARK, B. **Adaptive Control Systems**. 2.ed. Massachusetts: Addison-Wesley, 1995.

BITTENCOURT, G. **Inteligência Artificial: ferramentas e teorias**. 2.ed. Florianópolis: UFSC, 2001.

BOGDANOV, A. **Optimal Control of a Double Inverted Pendulum on a Cart**. Technical Report CSE-04-006, December 2004.

BOLTON, W. **Mecatrônica - Uma abordagem multidisciplinar**. 4. ed. Porto Alegre: Bookmann, 2010.

CHWIF, L.; MEDINA, A. C**. Modelagem e Simulação de Eventos Discretos: teoria e prática**. 2. ed. São Paulo: Bravarte, 2007.

DASTRANJ, M. R. et al. Design of optimal PID controller using genetic algorithm. **Australian Journal of Basic and Applied Sciences,** 5(10), p.996-1001, 2011.

DAVIDSON, J. M., & DURBIN, L. D. **Model Reference Adaptive Control Specification For a Steam Heated Finned Tube Heat Exchanger**. Texas, USA: A&M University, 1974. p.474-483, 1974.

DIP G.; PRAHLAD V.; KIEN P. D. **Genetic algorithm-based optimal bipedal walking gait synthesis considering tradeoff between stability margin and speed**, Robótica, v. 27, n. 3, p.355–365, 2009. Disponível em :< journals.cambridge.org/article_S026357470800475X>, acessado em: 25/01/2014.

DORF, R. C.; BISHOP, R. H. **Sistemas de Controle Moderno**.11.ed. Rio de Janeiro: LTC,. 2009.

EDGAR, T. F. E HIMMELBLAU, D. M. **Optimization of Chemical Processes**. New York: Mc Graw Hill Book, 1988.

ELE LITEX 2: Disciplina EE-09 Inteligência Artificial. **Síntese de uma das Áreas da Inteligência Artificial.** Disponível em: <http://www.ele.ita.br/~flavios/listex2.htm>. Acesso em: 22/ out/2013.

FERNANDES, A. M. **Inteligência Artificial: noções gerais**. 3. ed. Florianópolis: Visual Books, 2008.

FRANKLIN, G. F.; POWELL, J. D.; WORKMAN, M. L. **Digital Control of Dynamic Systems.** 3.ed. Menlo Park, California: Ellis-Kagle Press, 1998.

FREITAS FILHO, P. J. **Introdução à Modelagem e Simulação de Sistemas com Aplicação em Arena.** 2. ed. Florianópolis: Visual Books, 2008.

GE, S. S., WANG, C**. Adaptive neural control of uncertain MIMO nonlinear systems**. IEEE Transactions on Neural Networks, 15(3), p.674-692, 2004. Disponível: <http://www.ncbi.nlm.nih.gov/pubmed/15384555>. Acessado em 18/out/2013.

GEMAN, S.; BIENENSTOCK, E. DOURSAT, R. Neural Networks and the Bias/Variance Dilemma. **Journal Neural Computation**, v. 4, n. 1, p. 1-58, 1992. Disponível: < http://www.mitpressjournals.org/loi/neco>. Acessado em 18/out/2013.

GHORBANI, R.; WU, Q.; WANG, G. G. Nearly optimal neural network stabilization of bipedal standing using genetic algorithm. **Journal Engineering Applications of Artificial Intelligence**, n. 20, p. 473–480, 2007. Disponível <http://www.journals.elsevier.com/engineering-applications-of-artificial-intelligence/>. Acessado em 18/out/2013.

GOLBERG D.E., HOLLAND J.H.Genetic algorithms and machine learning. Mach Learn, v. 3, p. 95–99, 1988. Disponível: <http://link.springer.com/article/10.1023%2FA%3A1022602019183>. Acessado em 18/out/2013.

GORSE, D., SHEPHERD, A. J., TAYLOR, J. G. **The new ERA in supervised learning. Neural Network**, 10(2), p. 343-352, 1997. Disponível em: <http://discovery.ucl.ac.uk/79386/>. Acessado em 18/out/2013.

GUTIÉRREZ, G.; et al. Non-Direct Encoding Method Based on Cellular Automata to Design Neural Network Architectures**. Computing and Informatics**, V. 24, 1001–1023, 2005-Oct-18. Disponível em: <http://e-archivo.uc3m.es/bitstream/handle/10016/4118/non-irect_encoding_CI_2005.pdf>. Acessado em 18/out/2013.

HAO YUAND; BOGDAN M.; WILAMOWSKI.**The industrial Electronics Handbook: Intelligent Systems**. 2. ed. Boca Raton: CRC Press, 2011.

HASSEN, M.; MOHAMED, C. Variable Structure Neural Networks for Real Time Approximation of Continuous Time Dynamical Systems Using Evolutionary Artificial Potential Fields. **Wseas Transactions on Systems**, n. 2, p. 75–84, 2012. Disponível: < http://www.wseas.org/multimedia/journals/systems/2012/54-886.pdf>. Acessado em 18/out/2013.

HAYKIN, S. **Redes Neurais: princípios e práticas**. 2 ed. Porto Alegre: Bookmann, 2007.

HUNG L. C.; CHUNG H. Y. **Decoupled control using neural network-based sliding-mode controller for nonlinear systems.** Expert Systems with Applications, vl. 32, no. 4, p. 1168–1182, 2007. Disponível em: < www.sciencedirect.com/.../pii/S0957417406000935>, acessado em 25/01/2014.

HUNG L.;CHUNG **H. Decoupled sliding-mode with fuzzy-neural network controller for nonlinear systems.** International Journal of Approximate Reasoning 46 , p. 74–97, 2007. <www.sciencedirect.com/.../pii/S0957417406000935>, acessado em 25/01/2014.

INFORMÁTICA VIRTUAL. Tenha o seu próprio Robô. Disponível em: < http://www.informacaovirtual.com/tecnologia/tenha-seu-proprio-robo>. Acesso em 25/out/2013.

JONES, D. M., WATTON, J., E BROWN, K. J. **Comparison of hot rolled steel mechanical property prediction models using linear multiple regression, non-linear multiple regression and non-linear artifical neural networks**. 32(5), p.435–442, 2005. Disponível: <http://www.ingentaconnect.com/content/maney/ias/2005/00000032/00000005/art00014>. Acessado em 18/out/2013.

JUANG J.-G. **Fuzzy neural network approaches for robotic gait synthesis.** IEEE Transactions on Systems, Man and Cybernetics, Part B. p.594-601, Aug 2000. Disponível: <ieeexplore.ieee.org/iel5/3477/18734/00865178.pdf>, acessado em 25/01/2014.

KAYNAK O.; ERBATUR K.; ERTUGNRL M. **The fusion of computationally intelligent methodologies and sliding-mode control-a survey.** IEEE Trans. on Industrial Electronics, vol. 48, no. 1, pp. 4-17, Feb 2001. Disponível: <ieeexplore.ieee.org/iel5/3477/18734/00865178.pdf>, acessado em 25/01/2014.

KRATZER, K. P. **Neuronale Netze – Grundlagen und Anwendungen**. Disponível em < http://link.springer.com/chapter/10.1007%2F978-3-322-89950-7_4>. Acessado em 18/out/2013.

LAFETÁ, Thiago Fiallho Queiroz. **Enxame de Partículas Aplicado ao Problema Geral de Dimensionamento de Lotes**. Monografia (Ciência da Computação). Universidade Federal de Lavras, Lavras-MG, 2010.

LINS, C., De SOUSA, F,V., **Desenvolvimento de um Simulador Didático de um Pêndulo Invertido com Modelagem e Controle em Espaço de Estados.** Monografia (Engenharia de Controle e Automação). Instituto Federal Fluminense, Campos dos Goytacazes-RJ, 106 p., 2013.

LUGER, G. F. **Inteligência Artificial: estruturar e estratégias para solução de problemas complexos**. 4 ed. Porto Alegre: Bookmann, 2004.

MATWORKS. **Multilayer neural network architecture**. Disponível em: <http://www.mathworks.com/help/nnet/ug/multilayer-neural-network-rchitecture.html>. Acesso em: 22 de outubro de 2013.

MICHALEWICZ, W. **Genetic Algorithms + Data Structures = Evolution Programs**. 3.ed. New York: Springer, 1999.

MITCHELL, M, TAYLOR, C. E. Evolutionary computation: An overview. **Annual Review of Ecology and Systematics**, v.30, p. 593-616, 1999. Disponível em: <http://www.annualreviews.org/doi/abs/10.1146/annurev.ecolsys.30.1.593>. Acessado em 18/out/2013.

MOGHADDAS, M. et al. Design of Optimal PID Controller for Inverted Pendulum Using Genetic Algorithm**. International Journal of Innovation, Management and Technology**, V. 3, No. 4, August 2012. Disponível em: <http://www.ijimt.org/papers/271-D0373.pdf>. Acessado em 18/out/2013.

MOHAMAD, R. D. et al. Robust Control of Inverted Pendulum Using Fuzzy Sliding Mode Control and Genetic Algorithm. **International Journal of Information and electronics Engineering**. v. 2, n. 5, September 2012. Disponível em: <http://www.ijiee.org/papers/205-S240.pdf>. Acessado em 18/out/2013.

MUHAMMAD A. **Fuzzy and evolutionary modelling of nonlinear control systems.** Mathematical and Computer Modelling, v. 33, Issues 4–5, p. 533–551, February–March 2001. Disponível: <dl.acm.org/citation.cfm?id=2258782>, acessado em 25/01/2014.

NARENDRA, K. S; PARTHASARATHY, K. Identification and Control of Dynamical Systems Using Neural Networks. **IEEE Transactions on Neural Networks**, 1(1), p.4-27, 1990. Disponível em: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=80202&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D80202>. Acessado em 18/out/2013.

NASCIMENTO JÚNIOR., C. L.; YONEYAMA, T. **Inteligência Artificial em Controle e Automação.** 3. ed. São Paulo: Edgar Blucher, 2004.

NANDI G. C.; IJSPEERT A. J.; CHAKRABORTY P.; NANDI A. **Development of Adaptive Modular Active Leg (AMAL) using bipedal robotics technology.** Robotics and Autonomous Systems 57(6-7): p.603-616, 2009.Disponível em < www.sciencedirect.com/.../pii/S0921889009000402>, acessado em 25/01/2014.

NGUYEN V. B., MORRIS A. S. **Using a genetic algorithm to fully optimise a fuzzy logic controller for a two-link-flexible robot arm**. Robótica 27(5): p.677-687, 2009.Disponível em: <dl.acm.org/citation.cfm?id=2258782>, acessado em 25/01/2014.

OGATA, K. **Engenharia de Controle Moderno**. 5. ed. São Paulo: Pearson, 2010.

PASEMANN, F. Evolving Neurocontrollers for Balancing an Inverted Pendulum. **Network, Computation in Neural Systems**, 9, p.495-511, 1998. Disponível: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.6851>. Acessado em 18/out/2013.

PETRIDIS, V., PATERAKIS, E; KEHAGIAS, A. A hybrid Neural-Genetic Multimodel Parameter Estimation Algorithm**. IEEE Transactions on Neural Networks**, 9(5), p.862-876, 1998. Disponível em: < http://www.ncbi.nlm.nih.gov/pubmed/18255772>. Acessado em 18/out/2013.

POLLONI, E.; FEDELI, R.; PERES, F. E**. Introdução à ciência da computação**. 2.ed. São Paulo: Cengage Learning, 2010.

QIAN, S. AND LEE, Y.C. **Adaptive Stochastic Cellular Automata: Applications**. Physica D, 45(1-3): 181-188, 1990. Disponível em: < www.sciencedirect.com/.../pii/016727899090181N>, acessado em 25/01/2014.

RAVIV, Y. INTRATOR, N. Variance reduction via noise and bias constraints. In: **Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems**. London, Springer-Verlag, p. 163-175, 1999.

REKDALSBAKKEN, W. Feedback Control of an Inverted Pendulum with the use of Artificial Intelligence. Computational Cybernetics. **IEEE International Conference**. p. 1–6, 2006. Disponível em: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4097653&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4097653>. Acessado em 18/out/2013.

RIVERO, D; et al. Artificial Neural Network Development by means of Genetic Programming with Graph Codification. **World Academy of Science, Engineering and Technology**, v. 21, p. 880–885, 2008. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.193.5826&rep=rep1&type=pdf>. Acessado em 18/out/2013.

ROSENBLATT, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. **Psychological Review**. V. 65, p. 386-408, 1958. Disponível em: <http://psycnet.apa.org/index.cfm?fa=buy.optionToBuy&id=1959-09865-001>. Acessado em 18/out/2013.
RUSSELL, S.; NORVIG, P. **Inteligência Artificial**. 2. ed. Rio de Janeiro: Elsevier, 2003.

SAMAN R., BRYAN A. T. A new formulation forfeed forward Neural Networks. **IEEE Trans. NeuralNetw**., v.22, 10, p. 1588-1598, 2011. Disponível em: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5993545&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F72%2F4359168%2F05993545.pdf%3Farnumber%3D5993545>. Acessado em 18/out/2013.

SEGWAY SIMPLY MOVING: Vehicle patrol. Disponível em: <http://www.segway.com/patrol>. Acesso em 25/out/2013.

SCHUMA, C. D.; BIRDWELL, J. D. Variable structure dynamic artificial neural networks. **Biologically Inspired Cognitive Architectures**. n. 6, p. 126–130, 2013. Disponível em:< http://www.citeulike.org/user/jpsantos/article/12417518>. Acessado

em 18/out/2013

SILVA, T. **Estratégias de Aplicações Sequenciais e Paralelas da Metaheurística Otimização por Enxame de Partículas ao Problema do Caixeiro Viajante**. Mestrado (Engenharia de Produção). Universidade Federal do Rio Grande do Norte, Natal-RN, 2008.

TARGETHD: TECNOLOGIA. **O esqueleto mecânico Rex poderá substituir as cadeiras de roda no futuro.** Disponível em: < http://www.targethd.net/tecnologia >. Acesso em 25/out/2013.

TURING, A. M. Computing Machinery Intelligence. **Computing machinery and intelligence. Mind**, v. 59, n. 236, p. 433–460, 1950. Disponível em: <http://www.loebner.net/Prizef/TuringArticle.html>. Acessado em 18/out/2013

YAO, X; LIU, Y. A new evolutionary system for evolving artificial neural networks. **IEEE Transactions on Neural Networks**, v. 8, n.3, p. 694-713, 1997. Disponível em: < http://www.cs.bham.ac.uk/~axk/evoNN2.pdf>. Acessado em 18/out/2013

ZANETTI, S.S; et al. Estimating evapotranspiration using artificial neural network and minimum climatological data**. Journal of Irrigation and Drainage Engineering, Reston**, v.33, n.2, p.83-89, 2007. Disponível em: < http://scitation.aip.org/>. Acessado em 18/out/2013

WORDEN K.; STASZEWSKI W.J.; HENSMAN J.J. **Natural computing for mechanical systems research: A tutorial overview.** Mechanical Systems and Signal Processing, 25(1): p.4-111, 2011. Disponível em: www.sciencedirect.com/science/article/pii/S0888327010002499>, acessado em 25/01/2014.

WONG C. C , HER S. M. **A self-generating method for fuzzy system design.** Fuzzy Sets Syst. 103: p.13–25, 1999. Disponível em < dl.acm.org/citation.cfm?id=314970>, acessado em 25/01/2014.

WU J. C.; LIU T. S. **Fuzzy control of rider-motorcycle system using genetic algorithm and auto-tuning.** Mechatronics, 5(4):441–455, June 1995. Disponível em: www.sciencedirect.com/science/.../09574158950000>, acessado em 25/01/2014.

ANEXO I: ARTIGOS CIENTÍFICOS

Os doze artigos retornados na pesquisa utilizando os filtros citados na pesquisa bibliográfica são apresentados a seguir:

1. QIAN, S. AND LEE, Y.C. *Adaptive Stochastic Cellular Automata: Applications*, 1990;
2. WU J. C.; LIU T. S. *Fuzzy control of rider-motorcycle system using genetic algorithm and auto-tuning,* 1995;
3. WONG C. C , HER S. M. *A self-generating method for fuzzy system design. Fuzzy*, 1999;
4. JUANG J.-G. *Fuzzy neural network approaches for robotic gait synthesis*, 2000;
5. MUHAMMAD A. Fuzzy and evolutionary modelling of nonlinear control systems, 2001;
6. KAYNAK O.; ERBATUR K.; ERTUGNRL M. *The fusion of computationally intelligent methodologies and sliding-mode control-a survey*, 2001;
7. HUNG L. C.; CHUNG H. Y. *Decoupled control using neural network-based sliding-mode controller for nonlinear systems*, 2007;
8. HUNG L.;CHUNG H. *Decoupled sliding-mode with fuzzy-neural network controller for nonlinear systems*, 2007;
9. NANDI G. C.; IJSPEERT A. J.; CHAKRABORTY P.; NANDI A. *Development of Adaptive Modular Active Leg (AMAL) using bipedal robotics technology,* 2009;
10. DIP G.; PRAHLAD V.; KIEN P. D. *Genetic algorithm-based optimal bipedal*

walking gait synthesis considering tradeoff between stability margin and speed, 2009;

11. NGUYEN V. B., MORRIS A. S. *Using a genetic algorithm to fully optimise a fuzzy logic controller for a two-link-flexible robot arm,* 2009;
12. WORDEN K.; STASZEWSKI W.J.; HENSMAN J.J. *Natural computing for mechanical systems research: A tutorial overview,* 2011.