UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MOTOR DE INDUÇÃO LINEAR BLOQUEADO - OBTENÇÃO DA FORÇA DESEJADA ATRAVÉS DE ALIMENTAÇÃO NÃO SENOIDAL

MATHEUS GARCIA SOARES

MESTRADO

UBERLÂNDIA
2013

MATHEUS GARCIA SOARES

MOTOR DE INDUÇÃO LINEAR BLOQUEADO - OBTENÇÃO DA FORÇA DESEJADA ATRAVÉS DE ALIMENTAÇÃO NÃO SENOIDAL

Dissertação apresentada ao Departamento de Pós-Graduação da Faculdade de Engenharia Elétrica da Universidade Federal de Uberlândia (UFU) como parte dos requisitos para a obtenção do grau de Mestre em Engenharia Elétrica na área de Máquinas Elétricas.

Banca Examinadora:

Luciano Martins Neto, Dr. Orientador

Luciano Vieira Lima, Dr. UFU

Ricardo Silva Thé Pontes, Dr. UFC

UBERLÂNDIA

MARÇO, 2013

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE ENGENHARIA ELÉTRICA

PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MATHEUS GARCIA SOARES

MOTOR DE INDUÇÃO LINEAR BLOQUEADO - OBTENÇÃO DA FORÇA DESEJADA ATRAVÉS DE ALIMENTAÇÃO NÃO SENOIDAL

Dissertação apresentada ao Departamento de Pós-Graduação da Faculdade de Engenharia Elétrica da Universidade Federal de Uberlândia (UFU) como parte dos requisitos para a obtenção do grau de Mestre em Engenharia Elétrica na área de Máquinas Elétricas.

Luciano Martins Neto, Dr.	Alexandre Cardoso, Dr.
Orientador	Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

UBERLÂNDIA

MARÇO, 2013

Dedico este trabalho a Deus e a todos que me ajudaram nesta jornada.

Agradecimentos

Primeiramente agradeço a Deus por ter me guiado através do caminho até a realização do mestrado. Ao meu pai Adilson pelo incentivo e especialmente a minha mãe Cristina por me ajudar de forma incondicional e sempre zelar pelo meu bem estar.

Ao meu orientador, prof. Dr. Luciano Martins Neto, pela confiança depositada ao me aceitar como seu orientando. Neste dois anos me proporcionou uma oportunidade ímpar de aprendizado e crescimento, tanto profissional como pessoal.

Aos amigos Will e Pacheco, colegas do curso de pós-graduação, pela ajuda amizade e companherismo.

Ao Dr.André Luiz Gontijo por me conceder a oportunidade de dar continuidade em seu trabalho e pelo suporte prestado.

A Nayara pela atenção, compreensão e carinho dedicados. As minhas irmãs Naira e Livia pela paciência e carinho.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico-CNPQ pelo apoio financeiro.

Obrigado a todos que fizeram parte desta conquista, mesmo aqueles que não foram mencionados neste trabalho.

Muito Obrigado a Todos

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AG	Algoritmo Genético
MIL	Motor de Indução Linear
$v_x(t)$	Tensão Instantânea da Fase x
$i_x(t)$	Corrente Instantânea da Fase x
R_s	Resistência do Rotor do motor de Indução
R_r	Resistência do Estator do motor de Indução
L_{xy}	Indutância entre as fases X e Y
L_{DS}	Indutância de Dispersão do Estator
L_{DR}	Indutância de Dispersão do Linor
F(t)	Força Instantânea
T(t)	Conjugado Instantâneo
R	Raio do Motor rotativo
L	Comprimento
t	Tempo
θ	Posição Angular do Linor
P	Número de Polos
CC	Corrente Contínua
CA	Corrente Alternada

Resumo

O objetivo deste trabalho é aplicar um método heurístico para otimizar a escolha dos parâmetros de desbalanceamento das tensões de entrada de um motor de indução linear em baixas velocidades. O modelo matemático desenvolvido para o motor de indução linear considera o seu linor bloqueado, visando aplicações em baixas velocidades.

Através da aplicação de um programa de algoritmo genético é possível determinar a magnitude dos desbalanceamentos das tensões de entrada do motor linear. Os resultados experimentais obtidos são confrontados de forma satisfatória com os testes práticos realizados com um protótipo da máquina linear.

Palavras-chave: Motor de Indução Linear, Algoritmo Genético, controle de força, otimização.

Abstract

The objective of this work is to apply a heuristic method to optimize the choice of parameters of unbalanced input voltages of a linear induction motor at low speeds.The mathematical model developed for the linear induction motor considers his linor blocked, aiming applications at low speeds.

Applying a genetic algorithm program the magnitude of unbalance of the input voltage of the linear motor can be determine.The experimental results are compared satisfactorily with practical tests conducted with a prototype of the linear machine.

Keywords: Linear Induction Motor, Genetic Algorithm, force control, optimization

Capítulo 1

Introdução

1.1 Considerações Iniciais

Esta dissertação apresenta um aperfeiçoamento de um método já comprovado de controle de conjugado para motores de indução bloqueado. O método em questão foi desenvolvido inicialmente para o motor de indução trifásico rotativo e posteriormente adaptado ao motor linear. Foi comprovada a hipótese de que há um sistema de tensões trifásicas não senoidais, defasadas entre si de 120 graus, que aplicado aos terminais do motor gera um conjugado almejado, eliminando possíveis controle de malha fechada, ou seja, nenhum tipo de realimentação.

Este método foi desenvolvido por [GONTIJO 2011], onde a modelagem matemática utilizada se refere ao tradicional motor de indução trifásico rotativo, ou seja, seus enrolamentos trifásicos de estator e rotor são perfeitamente simétricos e equilibrados, podendo inclusive representar também o tradicional motor em gaiola de esquilo. É importante destacar que esta modelagem matemática foi, na referência citada, utilizada indistintamente para os motores rotativo e linear. Isto resultou em melhores resultados para o motor rotativo em relação ao linear, pois na realidade o motor linear possui uma assimetria inerente a sua própria construção, ficando, neste caso, a desejar a modelagem matemática utilizada. Para compensar esta questão, a referência citada utiliza o artifício de desbalancear o sistema de tensões trifásicas não senoidais aplicado ao motor linear, conseguindo com

isto, melhorar seus resultados.

O têrmo "aperfeiçoamento" utilizado no início deste capítulo se refere a dois assuntos a serem tratados nesta dissertação. Uma modelagem matemática mais apropriada, considerando toda a assimetria inerente do motor linear e a aplicação de técnicas computacionais de Algoritmos Genéticos para a otimização direcionada ao encontro de desbalanceamento de tensões em função da força desejada para o motor linear.

Este capítulo se destina a conceder ao leitor um referencial de consulta para que possa facilmente compreender a estrutura do trabalho e o desenvolvimento das ideias. A seguir, serão apresentados a justificativa e um detalhamento dos objetivos deste trabalho.

1.2 Justificativa

Existe uma grande gama de aplicações em que motores trabalham com velocidades extremamente baixas, ficando toda a atenção voltada para a força desenvolvida pelo motor. Nestas aplicações, a caracteristica da carga é exigir força e não velocidade. Muitos exemplos podem ser encontrados,como prensas, guilhotinas, compactação de materiais e atualmente aplicações voltadas para o campo da bioengenharia e biomecânica.

Muitas aplicações utilizam sistemas mecânicos para obterem a força necessária para a realização de uma determinada tarefa, porém na maioria dos casos estes são caros e demandam sempre uma manutenção dispendiosa.

A ideia basicamente é eliminar os sistemas mecânicos, utilizando somente o motor de indução linear, simplificando sua aquisição, instalação e manutenção.

A referência [GONTIJO 2011] mostra um caminho neste sentido, e este trabalho vem no sentido de colaborar para a melhoria deste caminho.

1.3 Objetivos

A seguir são apresentados os objetivos gerais e específicos deste trabalho, facilitando assim a compreensão de cada etapa.

1.3.1 Objetivo Geral

- Colaborar na viabilidade do uso do motor de indução linear de forma direta em aplicações que requerem força, em baixas velocidades.

1.3.2 Objetivos Específicos

- Desenvolver uma modelagem matemática para o motor linear,incluindo sua assimetria construtiva, tornando-a mais próxima da realidade.

- Utilizar a metodologia apresentada na referência [GONTIJO 2011] para a obtenção do sistema de tensões trifásicas não senoidais que alimentam o motor linear, substituindo a sua modelagem matemática do motor pela modelagem desenvolvida neste trabalho, especificadamente para o motor linear.

- Como já comentado anteriormente, o motor linear possui características construtivas que resultam em uma assimetria eletromagnética nas suas fases. Ao ser alimentado por um sistema trifásico de tensões balanceadas, mesmo que não sejam senoidais, que é o caso da referência [GONTIJO 2011] e também deste trabalho, a assimetria eletromagnética provoca uma componente negativa de força, que evidentemente prejudica o funcionamento do motor. Um artifício utilizado experimentalmente pela referência [GONTIJO 2011], foi desbalancear o sistema trifásico de tensões não senoidais. Variando o nível de desbalanceamento, foi possível observar que a componente negativa da força diminui, consequentemente aumentando a força resultante. Na própria referência foi também simulado computacionalmente este artifício, mostrando resultados positivos, mesmo com a modelagem matemática tradicional do motor de indução, ou seja, sem a sua assimetria. Desta forma o terceiro objetivo deste trabalho é desenvolver e aplicar um *software* que simula um gerador do sistema trifásico de tensões não senoidais e desbalanceadas, alimentando o motor linear, representado pela modelagem matemática que inclui a sua assimetria. Através de uma meta-heurística, buscar uma solução otimizada para o desbalanceamento das

tensões e força desejada; ou seja, obter qual nível de desbalanceamento das tensões não senoidais, que produz o valor mais próximo, dentro de uma certa precisão, da forma de onda temporal desejada para a força no motor linear.

1.4 Estrutura da dissertação

A dissertação é composta por seis capítulos, referencias bibliográficas e apêndice.

A introdução foi dividida em considerações iniciais, onde é apresentado uma descrição sucinta sobre o trabalho, o objetivo e a estrutura da dissertação, nos quais são evidenciados o objetivo e a forma de apresentação do trabalho.

O capítulo 2 faz um breve histórico do algoritmo genético, assim como todos os métodos de AG utilizados neste trabalho.

No capítulo 3, apresenta-se o equacionamento utilizado no motor de indução linear, e o desenvolvimento do modelo utilizado para o estudo do motor de indução linear.

O capítulo 4, apresenta os testes realizados com o modelo do motor de indução linear já acrescido do software do algoritmo genético, e em seguida são confrontados os valores de força encontrados no modelo, e em testes experimentais.

No capítulo 5, são expostas as conclusões, bem como as sugestões para trabalhos futuros.

Capítulo 2

O Algoritmo Genético

Este capítulo apresenta um breve histórico sobre algoritmos genéticos(AG) e descreve cada um dos métodos utilizados na construção do *software* do algoritmo genético. Conforme descrito no capítulo 1, no item 1.3.2, deve-se procurar otimizar a força desenvolvida pelo motor de indução linear através do desbalanceamento das suas tensões de fase. Este desbalanceamento matemático é obtido multiplicando as tensões de fase por constantes cujos valores são manipulados pelo AG afim de atingir o valor da força. Para referências futuras atribuiu-se os nomes Ka, Kb e Kc para as referidas constantes.

2.1 Uma Breve História Sobre AGs

A história dos algoritmos genéticos remete aos anos 40, quando os cientistas e pesquisadores começam a se basear em fenômenos naturais para criarem o ramo da inteligência artificial.

Uma das primeiras tentativas de associar otimização com a evolução natural foi realizada por [BOX 1957] quando apresentou um método de operação evolucionária, onde utiliza conceitos de variabilidade genética, mutação, e seleção natural para otimizar o processo de fabricação de uma indústria química.O seu trabalho sugeria mais um método do que um algoritmo para perturbar de forma sistemática duas ou três variáveis de controle de uma instalação, esta perturbação pode ser vista de modo análogo ao que entendemos

hoje como mutação e seleção.

Uma outra tentativa em aplicar processos evolutivos na resolução de problemas foi realizada pelo alemão Ingo Rechenberg, na primeira metade da década de 60, desenvolvendo as estratégias evolutivas. Mesmo não incluindo conceitos como o operador *crossover* e população maior, o trabalho de Rechenberg pode ser considerado pioneiro, por relacionar a computação evolucionária com as aplicações práticas.

Apesar de não ser o pioneiro em envestigar a a área, o americano John Henry Holland no final da década de 60 foi designado como o pai dos algoritmos genéticos.

Os algoritmos genéticos foram desenvolvidos por Holland, por seus colegas e por seus alunos na Universidade de Michigan. Sua pesquisa possuia dois objetivos, explicar de forma rigorosa os processos adaptativos dos sistemas naturais e projetar um sistema artificial em forma de software que contenha os principais mecanismos dos sistemas naturais [GOLDBERG 1989].

Em 1975 Holland publicou o seu primeiro livro,"Adaptation in Natural and Artificial Systems", seu trabalho apresenta os AGs como uma metáfora para os processos evolutivos, de forma que pudesse estudar a adaptação e a evolução presentes no mundo real, simulando dentro de um computador [LINDEN 2012].

Na década de 80 houve uma popularização dos algoritmos evolucionários no meio científico, isto fez com que surgissem as primeiras conferências dedicadas exclusivamente a este assunto.

Atualmente os algoritmos genéticos tem se beneficiado muito pela sua interdisciplinaridade. Cada vez mais pesquisadores de áreas diferentes a da computação buscam o auxílio dos AGs para a resolução de seus problemas.

2.2 O que são Algoritmos Genéticos ?

Algoritmos genéticos podem ser definido como sendo uma técnica computacional de busca baseada na heurística de um processo biológico de evolução natural. Os algoritmos genéticos utilizam de forma eficiente informações disponibilizadas pelo seu desenvolvedor

para especular sobre soluções que possuem um melhor desempenho para um problema proposto, o que o torna uma busca direcionada e não aleatória.

Segundo [GOLDBERG 1989] algoritmos genéticos são algoritmos de busca baseados em mecanismos de seleção natural e genética. Os algoritmos combinam o conceito da sobrevivência do indivíduo mais apto com estruturas que contém informações pertinentes ao problema para formar um algoritmo de busca.

De forma genérica a estrutura básica de um algoritmo genético consiste na criação de uma população de indivíduos que são selecionados e submetidos aos operadores genéticos: recombinação(*crossover*) e mutação. Estes operadores utilizam caracteristicas de cada indivíduo como parte da solução do problema proposto, desta forma as saídas geradas por estes operadores são avaliadas e o AG pode dizer se a solução foi alcançada. Eventualmente estes indivíduos vão passar por um processo de evolução e vão gerar novos indivíduos que podem caracterizar um boa solução para o problema.

De forma análoga a natureza as informações devem ser codificadas nas menores estruturas de um indivíduo, os cromossomos, e o crossover e a mutação se encarregarão de evoluir a população. A mutação cria diversidade, mudando aleatóriamente genes dentro de indivíduos e assim também como na natureza sua probabilidade de ocorrer é menor do que a da recombinação. Nas próximas seções deste capítulo serão detalhados a terminologia de AGs, o tipo de notação e os operadoes utilizados na contrução do programa do algoritmo genético.

2.3 Terminologia

Antes de prosseguir com mais explicações sobre os métodos de AG utilizados neste trabalho, é importante se familiarizar com a terminologia adotada. Como os algoritmos genéticos se assemelham a teoria da evolução das espécies, existe uma analogia muito forte entre a biologia e os termos utilizados para descrever os AGs.

Uma vez que os sistemas naturais funcionavam bem, Holland [HOLLAND 1975] procurou implementar algo semelhante para os sistemas artificiais. Nesta comparação descreve

o problema(ambiente de sobrevivência) como sendo uma função matemática e os indivíduos(cromossomos) mais fortes obtem valores mais altos dentro da função. Desta forma cada indivíduo corresponde a uma possivel solução.

Afim de manter a analogia com os sistemas de seleção natural, alguns termos da genética natural são utilizados nos sistemas artificiais. Desta forma, um indivíduo, mais especificamente neste trabalho caracterizado por um vetor de números é chamada de cromossomo, onde cada número corresponde a um gene, que por sua vez está inserido em uma determinada posição do vetor chamada de locus e possui um determinado valor designado de alelo. Para facilitar um primeiro contato com a analogia descrita acima, algumas ilustrações foram criadas. A Figura 2.1 mostra a menor unidade do AG, o gene.

Figura 2.1: Gene.

O desenvolvimento destas ilustrações seguem o mesmo padrão do algoritmo genético desenvolvido neste trabalho, sendo assim cada indivíduo contém 3 genes, que corresponde as variáveis Ka, Kb e Kc. A Figura 2.2 apresenta um suposto indivíduo composto de 3 genes.

Figura 2.2: Indivíduo.

Os algoritmos genéticos trabalham com um conjunto de indivíduos, assim uma típica população com cinco membros pode ser representada através da Figura 2.3.

A Tabela I mostra a relação entre as entidades dos sistemas naturais e as entidades dos algoritmos genéticos.

Os algoritmos genéticos são programas que trabalham em laços de repetição, ou seja, cada iteração do programa o AG utiliza os mesmos métodos de seleção, *crossover* e mutação para evoluirem a população a um estado que satisfaça o problema proposto. A cada

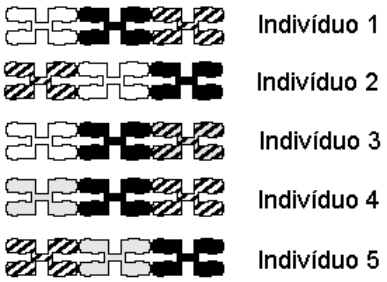

Figura 2.3: População.

Tabela I: Analogia entre Sistemas

Analogia entre Sitemas Naturais e Artificiais	
Linguagem de Sistemas Naturais	**Algoritmos Genéticos**
Cromossomo	Indivíduo, vetor numérico
Gene	Número
Alelo	Valor do Número
Locus	Posição do gene
População	Conjunto de Indivíduos, matriz numérica

uma destas iterações é dado o nome de geração.

2.4 Desenvolvimento do AG

Este item apresenta os principais componentes da criação do algoritmo genético. Onde serão descritos a criação da população, o método de seleção por torneio, a técnica de *simple crossover*[1] e a técnica da mutação simples.

2.4.1 O Indivíduo

Na codificação dos genes desenvolvidos para este trabalho, serão utilizados números reais. Desta forma cada indivíduo representa diretamente um conjunto de parâmetros a serem otimizados.

Ao utilizar uma representação em números reais, cada indivíduo possui todas as informações necessárias para a solução do problema, desta forma, pode-se dizer que cada indivíduo é uma possivel solução do problema proposto.

Cada indivíduo é representado por um vetor numérico de três posições, sendo cada

[1]Simple Crossover - palavra de origem inglesa, pode-se traduzi-la como cruzamento simples ou recombinação simples, este termo será utilizado em sua forma original, não havendo a sua tradução.

uma delas correspondente aos parâmetros Ka, Kb e Kc.

2.4.2 Criação da população

Para a criação da população utilizou-se uma estratégia simples, consistindo apenas em escolher uma quantidade X [2] de indivíduos de forma aleatória. A inicialização aleatória de maneira geral cria uma boa distribuição das soluções no espaço de busca [LINDEN 2012].

Como a população é um conjunto de indivíduos, a estrutura de cada um deles é importante, pois como citado em itens anteriores cada indivíduo consiste em uma possivel solução do problema.

2.4.3 Função de Avaliação(Aptidão)

Uma vez criada a população tem-se a necessidade de diferenciar cada indivíduo, pois como cada um representa uma solução possivel para o problema, deve-se saber qual apresenta uma melhor solução, e é neste ponto que a função de avaliação se faz necessária.

A função de avaliação é a maneira utilizada pelos algoritmos genético para determinar a qualidade de um indivíduo como a possivel solução do problema, de forma mais simples a função de avaliação pode ser vista como uma nota dada ao indivído pela resolução do problema. Esta nota será utilizada pelo método de seleção para determinar quais indivíduos são bons para sofrerem o processo de *crossover*.

Escolher somente os melhores indivíduos de uma determinada população baseando-se em suas notas é algo facil de se fazer, porém se faz necessária a utilização de um método de seleção para separar os indivíduos que sofrerão recombinação. Este método pode selecionar indivíduos bons e ruins, o que garante a variabilidade genética, pois indivíduos ruins ao sofrerem *crossover* e mutação podem gerar novos indivíduos bons.

[2] X deve ser um número inteiro e par

2.4.4 Seleção por Torneio

O método de seleção por torneio consiste em selecionar uma série de indivíduos da população para fazer com que eles entrem em competição direta pelo direito de ser o indivíduo que sofrerá o processo de recombinação, utilizando como arma a nota atribuida pela função de avaliação.

Os indivíduos são selecionados de forma aleatória para participarem do torneio, podendo até haver a repetição de um mesmo indivíduo. A única vantagem que os melhores indivíduos possuem é que se selecionados, eles vencerão o torneio. A Figura 2.4 mostra um exemplo da seleção por torneio.

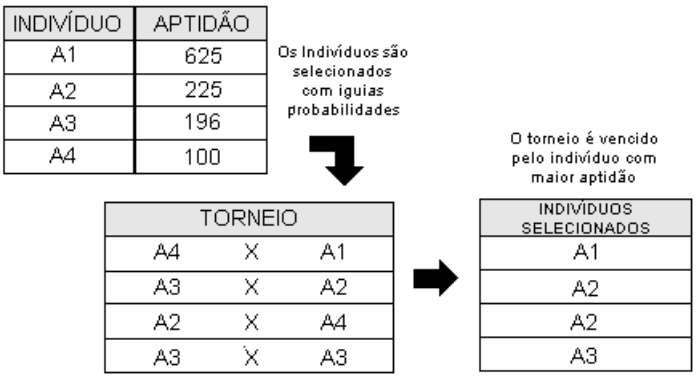

Figura 2.4: Seleção por Torneio.

Este método possui um parâmetro denominado "tamanho do torneio"(Z) que determina quantos indivíduos serão selecionados dentro da população para participarem do torneio.

O valor mínimo de Z é igual a 2, pois caso contrário não haveria competição com somenete um participante. O valor máximo de Z é determinado pelo tamanho da população, caso a Z seja atribuido este valor máximo o vencedor seria sempre o indivíduo com a melhor nota de avaliação dentro de toda a população.

Observe que na Figura 2.4 o valor de Z é igual a 2 pois são selecionado dois indivíduo para participarem do torneio. Este processo de escolha de participantes para o torneio se repete o mesmo mesmo número de vezes do tamanho da população, isto ocorre para que a população sempre possua o mesmo tamanho.

2.4.5 Operadores Genéticos

Dando continuidade nos estudos sobre AGs, se faz necessário a apresentação dos operadores genéticos utilizados no desenvolvimento deste trabalho.

Os operadores genéticos são responsáveis por transformar a população através de sucessivas gerações, guiando a busca para convergir em um resultado satisfatório.

Neste item serão apresentados os operadores de *simple crossover* e mutação simples, como seus próprios nomes já dizem, são as versões mais simples dos operadores genéticos.

Simple Crossover

Este operador de recombinação é o mais simples dentro do campo dos algoritmos genéticos, também chamado de *crossover* de um ponto. Uma vez concluido o processo de seleção, descrito no subitem 2.4.4 deste capítulo, dois indivíduos, que passarão a serem chamados de pais, são selecionados para sofrerem recombinação. O primeiro passo deste operador é a escolha de um ponto de corte, que nada mais é do que a posição entre dois genes de um cromossomo. Mudando o ponto de vista do lado biológico para o lado computacional, o ponto de corte será a posição entre dois indices de um vetor de tamanho n, sendo n a quantidade de indices deste vetor. Assim este vetor possuirá n-1 pontos de corte.

Na Figura 2.5 é mostrado um exemplo de pontos de corte para o cromossomo desenvolvido para AG deste trabalho, nota-se que este possui 3 genes e por conseguinte tem 2 pontos de corte possíveis.

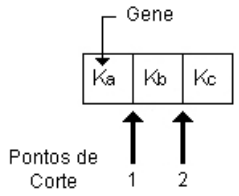

Figura 2.5: Pontos de corte em um vetor.

Após a determinação do ponto de corte, os pais estão separados em duas partes: uma à esquerda do ponto de corte e a outra à direita. O novo indivíduo criado após o *crossover*

será chamado de filho. O primeiro filho gerado é composto pela união da parte do primeiro pai à esquerda do ponto de corte com a parte do segundo pai à direita do ponto de corte. O segundo filho è composto pelas partes que sobraram. Um exemplo deste processo pode ser visualizado na Figura 2.6.

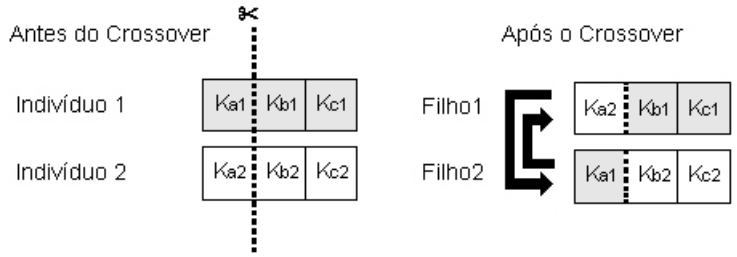

Figura 2.6: Simple Crossover.

Observando o processo de um ponto de vista matemático, pode-se descreve-lo como sendo a troca de componentes entre dois vetores de mesmo tamanho, criando assim dois novos vetores. O processo de *crossover* nem sempre ocorre todas as vezes, para isto existe um número de probabilidade atrelado a ele, chamado de taxa de *crossover*[3]. Após o processo de seleção dos pais, um número entre zero e um[4] é sorteado caso este número seja menor que a taxa de *crossover*, a recombinação ocorrerá.

Mutação Simples

Uma vez determinado os filhos, chega a vez do operador de mutação entrar em ação. O processo de mutação é bem simples, consistindo apenas na substituição de genes dentro de um cromossomo, por outro escolhido de forma aleatória. O novo valor deste gene será sorteado dentro de uma faixa numérica que vai de 0 até um valor limite escolhido pelo desenvolvedor do AG. A Figura 2.7 mostra o processo de mutação descrito. Observa-se que o segundo elemento do vetor que representa um indivíduo está sofrendo mutação e o seu valor foi alterado.

Este operador também possui, assim como o operador de *crossover* um número de probabilidade associado a ele, neste caso o número é extremamente baixo(na ordem de

[3]A taxa de *crossover* normalmente é definida em 75%
[4]Representam 0% e 100%

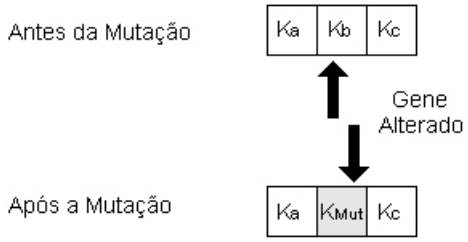

Figura 2.7: Mutação Simples.

1%), este número é chamado de taxa de mutação. De forma análoga ao processo de *crossover*, será sorteado um número entre 0 e 1. Caso este número seja menor que a taxa de mutação o operador entrará em ação. Este processo se repete para cada um dos genes que compõe os filhos.

O valor da Taxa de Mutação deve ser baixo. Caso se aumente muito este valor, a ocorrência da mutação se tornará maior, substituindo assim os valores de cada gene por números aleatórios, desta forma o algoritmo genético determinará a solução do problema de forma aleatória sem utilizar informações de gerações passadas.

Capítulo 3

Modelo Matemático e Simulador do Motor de Indução Linear - Linor Bloqueado

Este capítulo apresenta o equacionamento do motor de indução linear, bem como o desenvolvimento de seu simulador digital com o linor bloqueado. O propósito do modelo matemático é ser um modelo equivalente que se aproxime o máximo possível do motor de indução linear real.

3.1 Equacionamento com Linor Bloqueado

Neste item será estudado o modelo de um motor de indução linear com o linor bloqueado, levando em consideração a assimetria inerente a sua contrução, este modelo será utilizado na elaboração do simulador digital. O estudo e o desenvolvimento deste modelo se basearam nas referência [PONTES 2003] e [FITZGERALD et al. 2006].

Segundo [FITZGERALD et al. 2006] o estudo de máquinas lineares é muito similar à das máquinas rotativas. Tomando-se como base a modelagem da máquina rotativa, substituem-se as suas dimensões angulares por dimensões lineares, sua velocidade angular por linear, e seu conjugado por força. Assim as expressões para os parâmetros da máquina

linear são desenvolvidas de modo análogo aos parâmetros da máquina rotativa. Desta forma tem-se o desenvolvimento das equações da máquina linear.

A equação matricial da tensão em função da corrente no domínio do tempo, do motor de indução é apresentada em 3.1 .É oportuno lembrar que está sendo considerado a condição de linor bloqueado, e portanto a matriz [L] invariável no tempo.

$$[v(t)] = [R].[i(t)] + [L].\frac{d[i(t)]}{dt} \tag{3.1}$$

As matrizes de 3.2 a 3.5 são as representações de cada elemento da equação 3.1.

$$[v(t)] = \begin{bmatrix} v_{\mathrm{a}}(t) \\ v_{\mathrm{b}}(t) \\ v_{\mathrm{c}}(t) \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.2}$$

$$[i(t)] = \begin{bmatrix} i_{\mathrm{a}}(t) \\ i_{\mathrm{b}}(t) \\ i_{\mathrm{c}}(t) \\ i_{\mathrm{A}}(t) \\ i_{\mathrm{B}}(t) \\ i_{\mathrm{C}}(t) \end{bmatrix} \tag{3.3}$$

$$[R] = \begin{bmatrix} R_{\mathrm{S}} & 0 & 0 & 0 & 0 & 0 \\ 0 & R_{\mathrm{S}} & 0 & 0 & 0 & 0 \\ 0 & 0 & R_{\mathrm{S}} & 0 & 0 & 0 \\ 0 & 0 & 0 & R_{\mathrm{R}} & 0 & 0 \\ 0 & 0 & 0 & 0 & R_{\mathrm{R}} & 0 \\ 0 & 0 & 0 & 0 & 0 & R_{\mathrm{R}} \end{bmatrix} \tag{3.4}$$

$$[L] = \begin{bmatrix} L_{aa} + L_{da} & L_{ab} & L_{ac} & L_{aA} & L_{aB} & L_{aC} \\ L_{ba} & L_{bb} + L_{db} & L_{bc} & L_{bA} & L_{bB} & L_{bC} \\ L_{ca} & L_{cb} & L_{cc} + L_{dc} & L_{cA} & L_{cB} & L_{cC} \\ L_{Aa} & L_{Ab} & L_{Ac} & L_{AA} + L_{dA} & L_{AB} & L_{AC} \\ L_{Ba} & L_{Bb} & L_{Bc} & L_{BA} & L_{BB} + L_{dB} & L_{BC} \\ L_{Ca} & L_{Cb} & L_{Cc} & L_{CA} & L_{CB} & L_{CC} + L_{dC} \end{bmatrix} \tag{3.5}$$

Considerando a máquina linear como sendo a linearização geométrica de uma máquina rotativa, tem-se os seus enrolamentos esquematicamente apresentados na Figura 3.1.

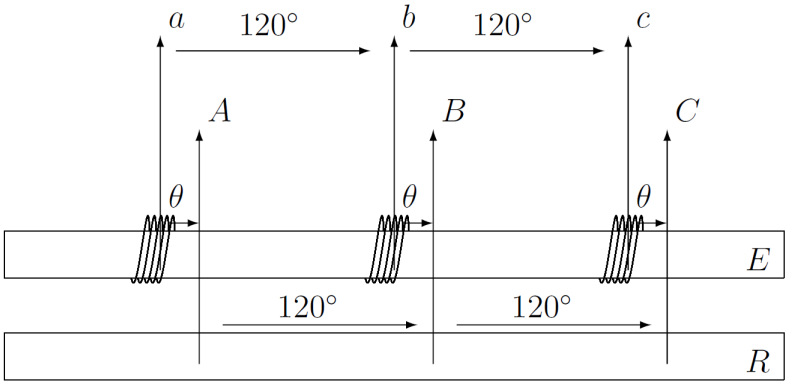

Figura 3.1: Enrolamentos Linearizados - estator E, rotor R.

Com base na da Figura 3.1 os valores de indutâncias da matriz 3.5 podem ser obtidos, expressões de 3.6 a 3.20. É importante frisar que o fato de se considerar, a princípio, as indutâncias diferentes entre si representam um primeiro passo para levar em consideração a assimetria do motor linear.

$$L_{ab} = L_{ba} = L_{ab}.\cos(\frac{2\pi}{3}) \tag{3.6}$$

$$L_{ac} = L_{ca} = L_{ac}.\cos(\frac{2\pi}{3}) \tag{3.7}$$

$$L_{bc} = L_{cb} = L_{bc}.\cos(\frac{2\pi}{3}) \tag{3.8}$$

$$L_{AB} = L_{BA} = L_{AB}.\cos(\frac{2\pi}{3}) \tag{3.9}$$

$$L_{AC} = L_{CA} = L_{AC}.\cos(\frac{2\pi}{3}) \tag{3.10}$$

$$L_{BC} = L_{CB} = L_{BC}.\cos(\frac{2\pi}{3}) \tag{3.11}$$

$$L_{\mathrm{aA}} = L_{\mathrm{Aa}} = L_{\mathrm{aA}}.\cos(\theta) \tag{3.12}$$

$$L_{\mathrm{aB}} = L_{\mathrm{Ba}} = L_{\mathrm{aB}}.\cos(\theta + \frac{2\pi}{3}) \tag{3.13}$$

$$L_{\mathrm{aC}} = L_{\mathrm{Ca}} = L_{\mathrm{aC}}.\cos(\theta - \frac{2\pi}{3}) \tag{3.14}$$

$$L_{\mathrm{bA}} = L_{\mathrm{Ab}} = L_{\mathrm{bA}}.\cos(\theta - \frac{2\pi}{3}) \tag{3.15}$$

$$L_{\mathrm{bB}} = L_{\mathrm{Bb}} = L_{\mathrm{bB}}.\cos(\theta) \tag{3.16}$$

$$L_{\mathrm{bC}} = L_{\mathrm{Cb}} = L_{\mathrm{bC}}.\cos(\theta + \frac{2\pi}{3}) \tag{3.17}$$

$$L_{\mathrm{cA}} = L_{\mathrm{Ac}} = L_{\mathrm{aC}}.\cos(\theta + \frac{2\pi}{3}) \tag{3.18}$$

$$L_{\mathrm{cB}} = L_{\mathrm{Bc}} = L_{\mathrm{bA}}.\cos(\theta - \frac{2\pi}{3}) \tag{3.19}$$

$$L_{\mathrm{cC}} = L_{\mathrm{Cc}} = L_{\mathrm{bB}}.\cos(\theta) \tag{3.20}$$

Para o motor rotativo o conjugado eletromagnético pode ser expresso por 3.21, expressão esta conhecida da conversão eletromagnética de energia e a título de comodidade pode-se referi-la a [BARBI 2013].

$$T(t) = \frac{P}{4}.\begin{bmatrix} i_{\mathrm{a}}(t) & i_{\mathrm{b}}(t) & i_{\mathrm{c}}(t) & i_{\mathrm{A}}(t) & i_{\mathrm{B}}(t) & i_{\mathrm{C}}(t) \end{bmatrix}.\frac{d[L]}{d\theta}.\begin{bmatrix} i_{\mathrm{a}}(t) \\ i_{\mathrm{b}}(t) \\ i_{\mathrm{c}}(t) \\ i_{\mathrm{A}}(t) \\ i_{\mathrm{B}}(t) \\ i_{\mathrm{C}}(t) \end{bmatrix} \tag{3.21}$$

A derivada de [L] em relação a θ está expressa em 3.22.

$$\frac{d[L]}{d\theta} = -\begin{bmatrix} 0 & 0 & 0 & L_{\mathrm{aA}}.\sin\theta & L_{\mathrm{aB}}.\sin(\theta+\frac{2\pi}{3}) & L_{\mathrm{aC}}.\sin(\theta-\frac{2\pi}{3}) \\ 0 & 0 & 0 & L_{\mathrm{bA}}.\sin(\theta-\frac{2\pi}{3}) & L_{\mathrm{bB}}.\sin\theta & L_{\mathrm{bC}}.\sin(\theta+\frac{2\pi}{3}) \\ 0 & 0 & 0 & L_{\mathrm{cA}}.\sin(\theta+\frac{2\pi}{3}) & L_{\mathrm{cB}}.\sin(\theta-\frac{2\pi}{3}) & L_{\mathrm{cC}}.\sin\theta \\ L_{\mathrm{Aa}}.\sin\theta & L_{\mathrm{Ab}}.\sin(\theta-\frac{2\pi}{3}) & L_{\mathrm{Ac}}.\sin(\theta+\frac{2\pi}{3}) & 0 & 0 & 0 \\ L_{\mathrm{Ba}}.\sin(\theta+\frac{2\pi}{3}) & L_{\mathrm{Bb}}.\sin\theta & L_{\mathrm{Bc}}.\sin(\theta-\frac{2\pi}{3}) & 0 & 0 & 0 \\ L_{\mathrm{Ca}}.\sin(\theta-\frac{2\pi}{3}) & L_{\mathrm{Cb}}.\sin(\theta+\frac{2\pi}{3}) & L_{\mathrm{cc}}.\sin\theta & 0 & 0 & 0 \end{bmatrix} \tag{3.22}$$

Substituindo 3.22 em 3.21 tem-se 3.23

$$T(t) = -\frac{P}{2}.\begin{Bmatrix} i_{\mathrm{a}}(t)[i_{\mathrm{A}}(t).L_{\mathrm{aA}}.\sin(\theta) + i_{\mathrm{B}}(t).L_{\mathrm{aB}}.\sin(\theta+\frac{2\pi}{3}) + i_{\mathrm{C}}(t).L_{\mathrm{aC}}.\sin(\theta-\frac{2\pi}{3})] + ... \\ i_{\mathrm{b}}(t)[i_{\mathrm{A}}(t).L_{\mathrm{bA}}.\sin(\theta-\frac{2\pi}{3}) + i_{\mathrm{B}}(t).L_{\mathrm{bB}}.\sin(\theta) + i_{\mathrm{C}}(t).L_{\mathrm{bC}}.\sin(\theta+\frac{2\pi}{3})] + ... \\ i_{\mathrm{c}}(t)[i_{\mathrm{A}}(t).L_{\mathrm{cA}}.\sin(\theta+\frac{2\pi}{3}) + i_{\mathrm{B}}(t).L_{\mathrm{cB}}.\sin(\theta-\frac{2\pi}{3}) + i_{\mathrm{C}}(t).L_{\mathrm{cC}}.\sin(\theta)] \end{Bmatrix} \tag{3.23}$$

3.1.1 Independência da Posição do Linor

A referência [GONTIJO 2011] mostra que para uma dada alimentação elétrica(sistemas de tensões trifásicas) do motor linear, com linor bloqueado, a sua resposta em termos de correntes elétricas e força independe da posição do linor "θ", que por sua vez é constante, pois o linor bloqueado implica em $\partial\theta/\partial$t igual a zero. Assim pode-se adotar qualquer qualquer valor para "θ". Portanto, para efeito deste trabalho adota-se θ igual a zero.

Desta forma os valores de indutância se transformam em 3.24 a 3.32, aplicando-se "$\theta = 0$" nas expressões de 3.12 a 3.20.

$$L_{\mathrm{aA}} = L_{\mathrm{Aa}} \tag{3.24}$$

$$L_{\mathrm{aB}} = L_{\mathrm{Ba}} = -\frac{1}{2}.L_{\mathrm{aB}} \tag{3.25}$$

$$L_{\mathrm{aC}} = L_{\mathrm{Ca}} = -\frac{1}{2}.L_{\mathrm{aC}} \tag{3.26}$$

$$L_{\mathrm{bA}} = L_{\mathrm{Ab}} = -\frac{1}{2}.L_{\mathrm{bA}} \tag{3.27}$$

$$L_{\mathrm{bB}} = L_{\mathrm{Bb}} \tag{3.28}$$

$$L_{\mathrm{bC}} = L_{\mathrm{Cb}} = -\frac{1}{2}.L_{\mathrm{bC}} \tag{3.29}$$

$$L_{\mathrm{cA}} = L_{\mathrm{Ac}} = -\frac{1}{2}.L_{\mathrm{aC}} \tag{3.30}$$

$$L_{\mathrm{cB}} = L_{\mathrm{Bc}} = -\frac{1}{2}.L_{\mathrm{bA}} \tag{3.31}$$

$$L_{\mathrm{cC}} = L_{\mathrm{Cc}} \tag{3.32}$$

E suas derivadas na expressão matricial de 3.33.

$$\left.\frac{d[L]}{d\theta}\right|_{\theta=0} = \begin{bmatrix} 0 & 0 & 0 & 0 & \sqrt{3}.Lab & -\sqrt{3}.Lac \\ 0 & 0 & 0 & -\sqrt{3}.Lab & 0 & \sqrt{3}.Lab \\ 0 & 0 & 0 & \sqrt{3}.Lac & -\sqrt{3}.Lab & 0 \\ 0 & -\sqrt{3}.Lab & \sqrt{3}.Lac & 0 & 0 & 0 \\ \sqrt{3}.Lab & 0 & -\sqrt{3}.Lab & 0 & 0 & 0 \\ -\sqrt{3}.Lac & \sqrt{3}.Lab & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.33}$$

Para o conjugado T(t), aplica-se "$\theta = 0$" em 3.23 transformando-se em 3.34.

$$T(t) = -\frac{\sqrt{3}.P}{4}.\left\{\begin{array}{c} i_{\mathrm{a}}(t)[i_{\mathrm{B}}(t).L_{\mathrm{aB}} - i_{\mathrm{C}}(t).L_{\mathrm{aC}}] + i_{\mathrm{b}}(t)[i_{\mathrm{C}}(t).L_{\mathrm{bC}} - i_{\mathrm{A}}(t).L_{\mathrm{bA}}] + ... \\ i_{\mathrm{c}}(t)[i_{\mathrm{A}}(t).L_{\mathrm{cA}} - i_{\mathrm{B}}(t).L_{\mathrm{cB}}] \end{array}\right\} \tag{3.34}$$

No momento, se faz necessário obter a força do motor linear a partir do conjugado do motor rotativo. Esta conversão está apresentada a seguir.

3.1.2 Conversão do Conjugado em Força

A Figura 3.2 mostra de forma imaginária a construção do motor linear. Ela pode ser compreendida como sendo o resultado de uma máquina rotativa, cortada ao longo de um plano axial e linearizada.

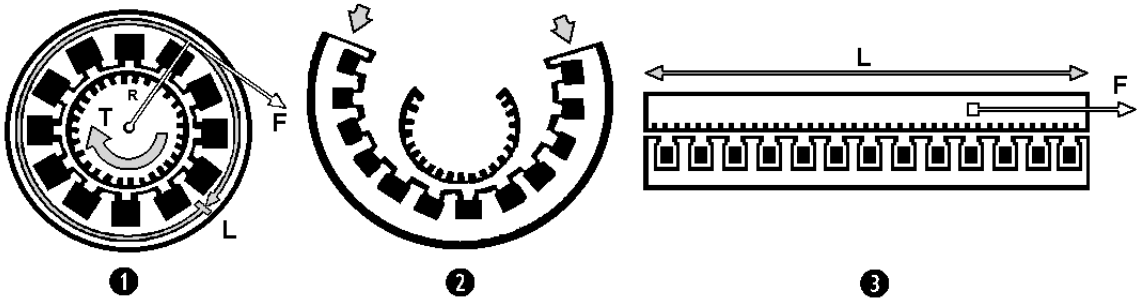

Figura 3.2: Transformação do conjugado em força

Nota-se na Figura 3.2 a ação da força tanto no motor rotativo como no motor linear.

A relação entre o conjugado e a força dependem do comprimento do motor linear. As equções de 3.35 a 3.37 mostram esta dependência. Na equação 3.35 a relação entre o raio do motor rotativo e a força determinam o seu conjugado.

$$T = R.F \tag{3.35}$$

O comprimento do motor rotativo é dado por 3.36.

$$L = 2.\pi.R \tag{3.36}$$

Substituindo 3.36 em 3.35 e isolando-se a força tem-se 3.37.

$$F = \frac{2.\pi}{L}.T \tag{3.37}$$

De 3.37 e 3.34 tem-se 3.38.

$$F(t) = -\frac{2.\pi}{L}.\frac{\sqrt{3}.P}{4}.\left\{\begin{array}{c} i_{\mathrm{a}}(t)[i_{\mathrm{B}}(t).L_{\mathrm{aB}} - i_{\mathrm{C}}(t).L_{\mathrm{aC}}] + i_{\mathrm{b}}(t)[i_{\mathrm{C}}(t).L_{\mathrm{bC}} - i_{\mathrm{A}}(t).L_{\mathrm{bA}}] + ... \\ i_{\mathrm{c}}(t)[i_{\mathrm{A}}(t).L_{\mathrm{cA}} - i_{\mathrm{B}}(t).L_{\mathrm{cB}}] \end{array}\right\} \tag{3.38}$$

3.2 Simulador Digital

O simulador foi construído utilizando-se o software Simulink®[1], que faz parte do programa computacional Matlab®[2].Para o desenvolvimento do modelo, utilizou-se a *toolbox* SimPowerSystems®[3], que possui modelos pré-definidos de circuitos elétricos.

O sistema modelado, apresentado na Figura 3.3, é composto pelos blocos, "Gerador de Harmônicos", "Motor de Indução Linear" e "Medição de tensões e correntes". Os outros componentes do sistema são utilizados para exibir e transferir dados.

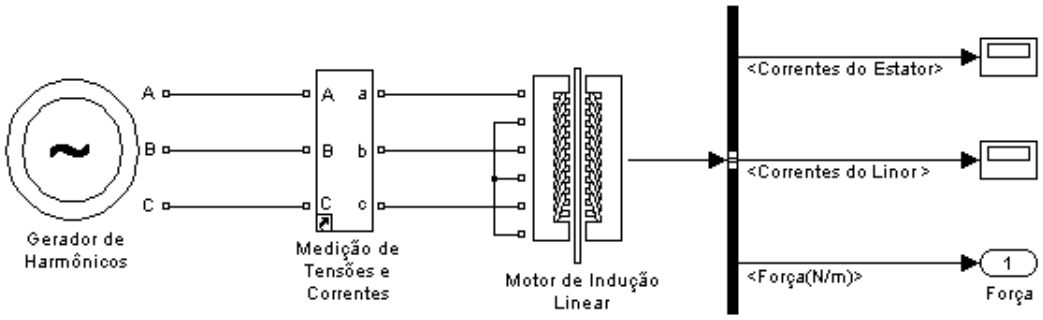

Figura 3.3: Modelo desenvolvido para o estudo da força.

O bloco "Medição e Correntes e Tensões" é definido em uma das bibliotecas do Simulink®. Por esta razão será dada maior ênfase para os blocos "Gerador de Harmônicos" e "Motor de Indução Linear" que foram construidos exclusivamente para o estudo da força produzida pelo motor de indução linear alimentado com tensões contendo harmônicas.

[1]Software desenvolvido pela compania MathWorks, é uma ferramenta para modelagem, simulação e análise de sistemas dinâmicos.

[2]MATrix LABoratory é um software destinado a fazer cálculos utilizando matrizes

[3] *Toolbox* dedicada exclusivamente à simulação de sistemas de energia/eletrônica de potência,permite simulação de sistemas em CC e CA

O bloco “Gerador de Harmônicos” é uma fonte de tensão trifásica que pode gerar tensões compostas de até 11 harmônicas. Após geradas as tensões, estas são injetadas no bloco correspondente a máquina linear. Este bloco possui cinco campos de entrada que definem as variáveis correspondentes a tensões, ângulos , frequência e desbalanceamento das amplitudes. Nos campos correspondentes às “Tensões Harmônicas” e “Ângulos das Harmônicas” o usuário inseri o valor das amplitudes e ângulos de cada componente harmônica que a fonte irá gerar.

O campo “Frequência” define a frequência das ondas geradas. Este campo deve estar relacionado com o tempo de simulação do Simulink® para que o resultado final seja satisfatório. O último campo, “Multiplicador”, aplica um desbalanceamento de amplitude nas tensões de saída do bloco. O usuário define quais são as magnitudes de desbalanceamento de cada uma das três fases. Cada uma das entradas do bloco “Gerador de Harmônicos” são mostradas na Figura 3.4.

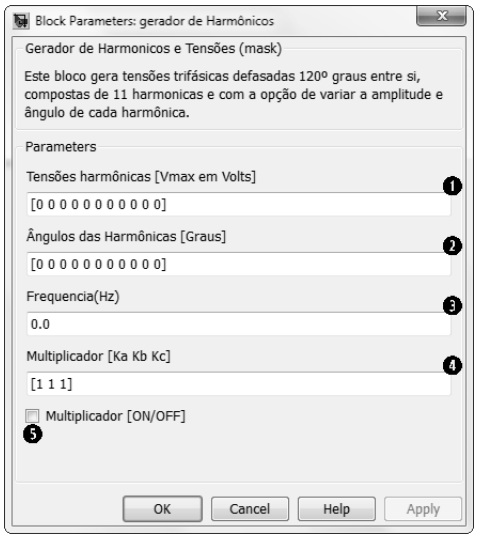

Figura 3.4: Janela com os parâmetros de entrada do bloco “Gerador de Harmônicos”.

Ao se expandir o bloco “Gerador de Harmônicos”, visualiza-se o diagrama de blocos mostrado na Figura 3.5, o principal bloco deste diagrama é o bloco “Tensoes”. Este é um bloco especial do Simulink®, chamado *Embedded MATLAB Function*[4], que permite ao desenvolvedor utilizar códigos da linguagem do Matlab® ou códigos em linguagem C

[4]Para a utilização deste bloco são necessárias a instalação de dois arquivos, o Visual Studio 2008 Express Edition e o Microsoft Windows SDK

para manipular as entradas e saídas de dados. O código fonte utilizado neste bloco se encontra no Apêndice A deste trabalho.

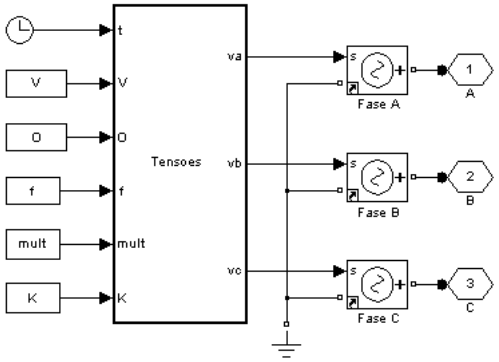

Figura 3.5: Diagrama de blocos do "Gerador de Harmônicos".

As entradas "V" e "O", contidas no diagrama de blocos, mostrado na Figura 3.5, são vetores contendo, respectivamente, as amplitudes das tensões e os ângulos iniciais para cada harmônico. Estas duas grandezas são utilizadas nos cálculos para gerarem a tensão trifásica que alimentará o motor de indução linear.Além destas duas entradas o bloco "Tensoes" possui a entrada "f" que concede o valor da frequência, e a entrada "K" que é um vetor de três posições que contém o valor de desbalanceamento das amplitudes das tensões de saída de cada fase.

O modelo do motor de indução linear com linor bloqueado, contido no bloco "Motor de indução Linear", possui cinco parâmetros de entrada, estes parâmetros podem ser visualizados na Figura 3.6. Todos os parâmetros são inseridos no formato de vetor, a primeira entrada contempla os valores das resistências do estator e do linor, a segunda, terceira e quarta entradas são destinadas, respectivamente, aos valores das reatâncias de dispersão, próprias e mútuas. No último campo de entrada são definidos os parâmetros da máquina, neste campo o usuário determina o número de polos do motor.

O bloco do motor linear pode ser visto de forma mais detalhada através da Figura 3.7. O barramento de entrada é responsável por multiplexar as seis entradas para o bloco principal do modelo, o bloco "MIL", este bloco também possui características especiais e é chamado de *s-function*. É no bloco "MIL" que o modelo do motor de indução linear desenvolvido no primeiro item do capítulo 3 foi aplicado, este bloco é responsável pela

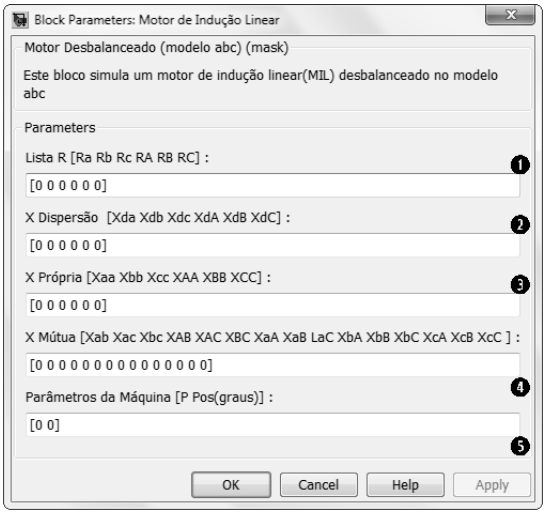

Figura 3.6: Janela com os parâmetros de entrada do bloco "Motor de Indução Linear".

resolução do sistema em espaço de estados do motor. O bloco "medidas" demultiplexa as saídas do bloco "MIL" e as disponibiliza para o bloco "enrolamento", que por sua vez transformam os sinais em corrente através de fontes de corrente controladas.

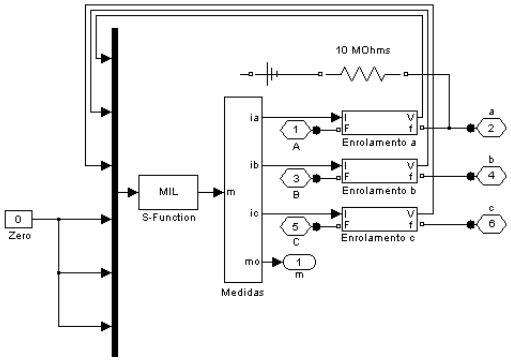

Figura 3.7: Diagrama de blocos "Motor de Indução Linear".

Qualquer bloco do tipo *s-function* utiliza-se de um arquivo.m para ser configurado, sendo assim o código desenvolvido para a simulação do motor deve ser conectado com o bloco da *s-function*. Abrindo a janela do bloco "MIL", exibida na Figura 3.8, observa-se três campos configuráveis, sendo os dois primeiros os mais importantes. O campo *" S-function name"* deve receber como parâmetro o mesmo nome do arquivo.m que contem o código da modelagem do motor, já o campo *"S-function parameters"* deve receber o nome dos parâmetros de entrada que serão utilizados na mesma modelagem.

Assim o bloco *s-function* "MIL" conecta o Simulink® com o código do modelo. Para

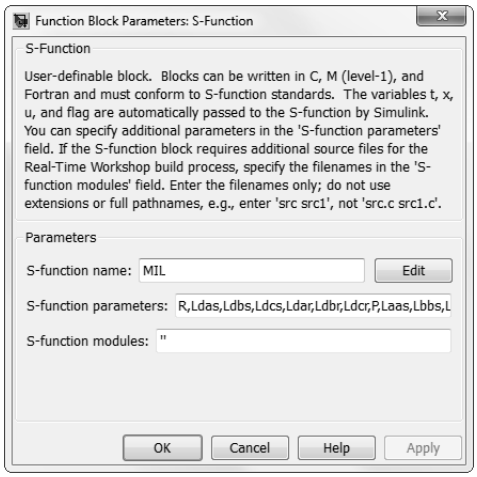

Figura 3.8: Janela de parâmetros de configuração da *s-function* "MIL".

que o modelo desenvolvido possa ser utilizado, este deve ser modificado em um sistema em espaço de estados do motor. Tomando como ponto de partida a equação 3.1, pode-se isolar a derivada da corrente de maneira que a equação matricial fique igual a 3.39. Esta expressão pode ser reescrita na forma da equação 3.40 , podendo ser resolvida por métodos computacionais, tais como o ode23tb[5], presente no MATLAB®. O código fonte do bloco "MIL" está disponível no apêndice B deste trabalho.

$$\frac{d[i(t)]}{dt} = [L]^{-1}.[[v(t)] - [R].[i(t)]] \tag{3.39}$$

$$\frac{dX}{dt} = A - (B.X) \tag{3.40}$$

O bloco "Medidas", apresentado na Figura 3.9, multiplexa a saída do bloco "MIL". Esta operação se torna necessária pois o bloco "MIL" gera uma saída em forma de vetor contendo sete posições. As seis primeiras posições do vetor correspondem as correntes do estator e do linor, estas correntes são disponibilizadas para o bloco "Enrolamento".A sétima e última posição do vetor recebe o valor da força elétrica.

O bloco "Enrolamento" será o ultimo bloco a ser apresentado,ele simula o enrolamento de cada fase do estator da máquina, a Figura 3.10 mostra em detalhe este bloco.O bloco

[5]Resolve equações diferenciais difíceis, método de baixa ordem.

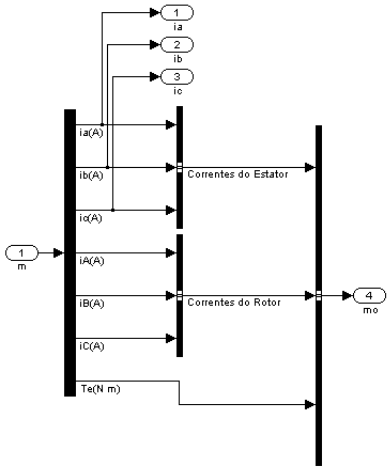

Figura 3.9: Diagrama de blocos "Medidas".

"Enrolamento" recebe como entrada um sinal de corrente, proveniente do bloco "Medidas" e devolve como saída um sinal de tensão.

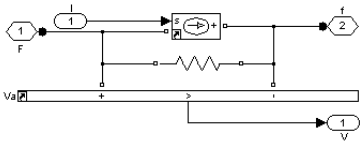

Figura 3.10: Diagrama de blocos "Enrolamento".

3.3 Software do AG

Este item se dedica a apresentar o *software* do algoritmo genético utilizado para encontrar os valores das constantes de desbalanceamento das tensões de fase do motor de indução linear, denominadas Ka, Kb e Kc no início do capítulo 2, podendo assim obter uma solução próxima da desejada. O código fonte deste *software* está disponível no apêndice C, e se encontra subdividido em seis partes do C.1 ao C.6.

Para o desenvolvimento do programa foi utilizado o *software* MATLAB®, devido ao fato de ser compatível com o modelo do motor desenvolvido no *software* Simulink®. Desta forma foi possível alterar os parâmetros desejáveis dentro do modelo da máquina, e também receber seus dados de saída. Assim o programa do algoritmo genético ficou responsável por inserir e receber os dados do simulador digital. A Figura 3.11 mostra o

fluxograma do programa desenvolvido, utilizando a técnica de algoritmo genético.

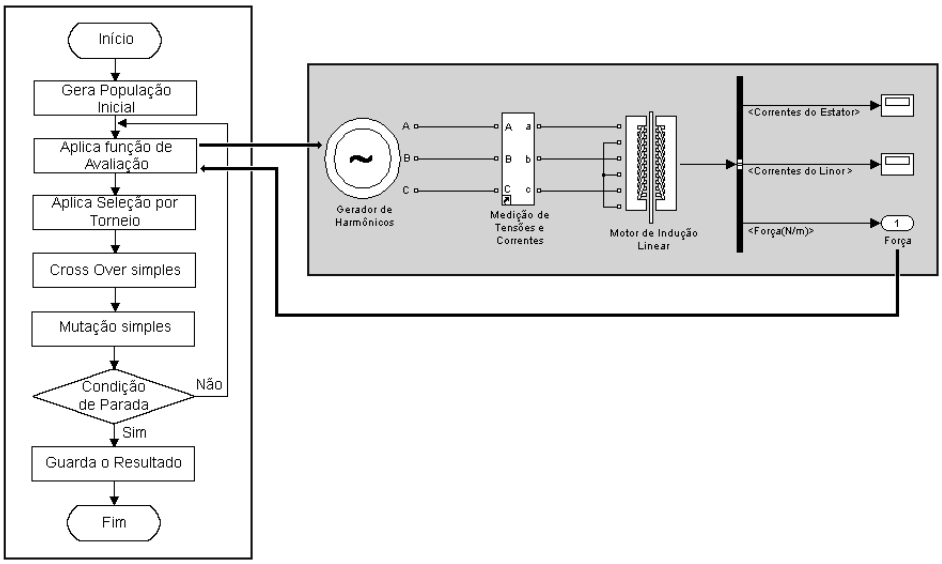

Figura 3.11: Fluxograma do programa do algoritmo genético

O *software* se apresenta de forma modular, ou seja, cada função criada para o programa foi desenvolvida em um arquivo diferente. A escolha deste tipo de construção de software facilita o seu desenvolvimento, e também a inserção de novas funções ou a modificação das já existentes.

O apêndice C.1 apresenta a função principal do programa do AG, que contém alguns parâmetros importantes para o desempenho do programa. Caso os parâmetros de entrada do algoritmo genético forem ajustados de forma incondizente com o problema, a resposta final pode demorar muito para ser obtida ou nunca ser encontrada. A seguir cada um destes parâmetros serão explicados. Os parâmetros são denominados por: “n”,“lim”, “f”, “z”, “Tcruza”, “Tmuta”, “Gmax” e “erro”.

O parâmetro “n” determina o tamanho da população de números que estão inseridos dentro do espaço de busca [6]. Segundo [LINDEN 2012] deve-se ter cuidado ao escolher o valor do tamanho da população, pois pode resultar um alto custo computacional [7], na ordem de n^2, onde n é o tamanho da população. A Figura 3.12 mostra uma representação de uma população contendo três indivíduos, note que cada um é composto por três genes

[6] espaço de busca: conjunto de números onde se encontra a solução desejada.
[7] custo computacional: nível exigência do computador para executar uma determinada tarefa

que correspondem aos valores de Ka, Kb e Kc, anteriormente apresentados.

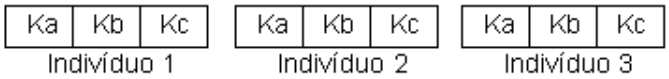

Figura 3.12: Representação de uma população de três indivíduos

O parâmetro "lim" limita o valor numérico dos genes sorteados de cada indivíduo. Ao se definir o valor deste parâmetro, deve-se tomar cuidado para que a tensão de entrada do motor não ultrapasse o seu valor nominal. Desta forma pode-se assegurar que o algoritmo genético encontra sempre valores válidos para as constantes Ka, Kb e Kc.

O parâmetro "f" representa o valor da força desejada produzida pelo motor linear. Este parâmetro é determinado pelo tipo de pulso de força escolhido para o motor desempenhar.

O parâmetro "z" já foi apresentado no capítulo 2. Ele pertence ao método de seleção por torneio. Seu valor está vinculado ao tamanho da população, não podendo ser maior que ela. Caso se escolha um valor alto para "z" ,em relação ao tamanho da população, pode ocorrer uma prematura convergência do resultado para um mínimo local, ou seja, o resultado ficaria travado em um determinado valor diferente do desejado.

Os parâmetros "Tcruza" e "Tmuta" definem as taxas de *crossover* e mutação, respectivamente, que por sua vez determinam a frequência de ocorrência.

O critério de parada por número de gerações é determinado por "Gmax", enquanto que o "erro" comanda o final da execução do programa.

Capítulo 4

Resultados

Este capítulo apresenta o conjunto simuladores digital e algoritmo genético(AG), este último utilizado para encontrar os parâmetros de desbalanceamento do sistema de tensões trifásicas. A seguir faz-se uma explanação sobre os parâmetros utilizadas no simulador digital, e uma avaliação do mesmo. Finalmente são apresentados os resultados obtidos pelo conjunto Simulador Digital e AG com a respectiva análise.

4.1 Simulador AG

Como visto anteriormente no capítulo 2 o programa do AG necessita de uma função que avalia cada indivíduo da população, atribuindo uma nota a cada um. Esta nota, também chamada de aptidão, comunica ao programa, simulador digital do motor de indução linear, qual dos indivíduos está mais próximo da solução.

Atuando como função de avaliação, o simulador digital necessita dos indivíduos, desta forma cada um deles será inserido no campo "multiplicador", pertencente ao bloco "Gerador de Harmônicos". Este campo aguarda como entrada um vetor numérico de três posições. O desenvolvimento de cada indivíduo foi realizado levando-se em consideração esta exigência do campo "multiplicador", desta forma, cada indivíduo se encaixa de forma ideal como solução do problema de otimização proposto.

Uma vez ligado, o simulador devolverá como resposta a força simulada pelo bloco

"motor de indução linear", e que será comparada com a força desejada, definindo assim a aptidão de cada indivíduo.

Com todos os indivíduos avaliados, o programa de algoritmo genético segue o seu curso realizando as operações de *crossover* e mutação.

Logo que se atinge o valor da força desejada, o *software* de AG finaliza a sua execução e devolve como resposta os valores das constantes, Ka Kb e Kc, que vão causar o desbalanceamento no sistema de tensões trifásicas que alimenta o motor.

4.2 Simulador Digital

O Simulador Digital é composto do "Gerador de Harmônicos"e do "Motor de Indução Linear".

4.2.1 Gerador de Harmônicos

Para configurar o bloco "Gerador de Harmônicos", utilizou-se o método de controle desenvolvido por [GONTIJO 2011] para definir os valores das amplitudes e ângulos das tensões harmônicas que geram o perfil de força eletromagnética desejada. O perfil da força a ser adotada, como exemplo, neste trabalho, é trapezoidal, com duração 0,5 segundo e amplitude 25N, como mostra a Figura 4.1.

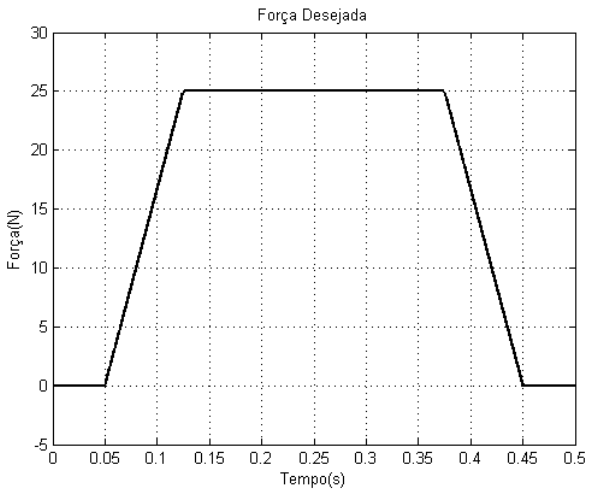

Figura 4.1: Perfil da força trapezoidal requerida.

Para obter as componentes harmônicas que formam as tensões de alimentação do motor, a referência [GONTIJO 2011] desenvolve uma metodologia, que por razões didáticas de apresentação deste trabalho passa a ser explicada de forma bem sintética. A questão fundamental é expressar os componentes harmônicos das tensões em função do conjugado.

Tradicionalmente é conhecida a equação do conjugado em função das correntes no motor. Ela é algébrica, e genericamente não linear pois alguns coeficientes são variáveis, representando as indutâncias do motor. No caso em questão, como o motor está sempre bloqueado, as suas indutâncias são invariáveis no tempo e portanto a referida equação se torna linear. Isto possibilita uma inversão algébrica colocando-se a corrente em função do conjugado. Outra função tradicionalmente conhecida relaciona a tensão de fase com as correntes do motor. Esta é uma equação diferencial e também genericamente não linear. Porém, com o motor bloqueado, pela mesma razão anterior, a equação se torna linear. Esta condição permite resolver a equação diferencial através da transformada de Laplace, tornando-a uma equação algébrica linear. Desta forma, utilizando-se as equações algébricas lineares "corrente em função do conjugado" e "tensão em função da corrente" é possível, por substituição, obter a "tensão em função do conjugado". Fazendo um tratamento isolado por componente harmônica é possível obter, para um dado perfil de conjugado, o perfil da tensão de cada fase do motor. Este perfil é formado pela composição das componentes harmônicas obtidas.

A Tabela II mostra as amplitudes e ângulos das tensões harmônicas inseridas no bloco "Gerador de Harmônicos", para obter o perfil de conjugado da Figura 4.1, valores estes extraídos da referência [GONTIJO 2011].

Tabela II: Tensões harmônicas para o motor linear(Trapezoidal - período de 0,5 segundos)- Extraido referência [GONTIJO 2011]

	Harmônicas							
	1^a	2^a	4^a	5^a	7^a	8^a	10^a	11^a
V (Volts)	8,3834	-3,7227	-1,5534	-1,7714	-25,7653	-7,3991	20,3366	1,4822
θ (Graus)	55,4134	70,5291	-297,1818	67,2709	12,1582	-266,3280	-26,3496	0

Fazendo a composição harmônica com os valores apresentados na tabela II obtem-se o perfil das tensões de cada fase que alimentam o motor, Figura 4.2.

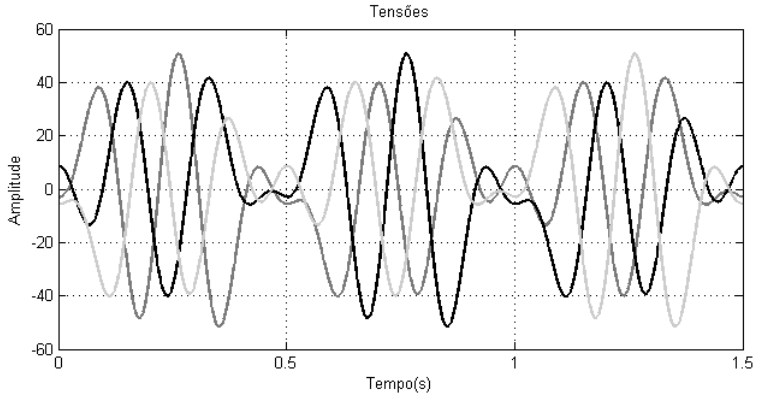

Figura 4.2: Perfil das tensões de entrada do MIL.

4.2.2 Motor de Indução Linear

Este subitem fornece os parâmetros do protótipo do motor de indução linear utilizados no bloco "Gerador de Harmônicos". O método para obtenção destes parâmetros foi utilizado por [GONTIJO 2011] com base em [PONTES 2003].

O protótipo do motor de indução linear é o mesmo de [PONTES 2003], Figura 4.3, e que se trata de uma guilhotina impulsionada pelo motor linear.

O núcleo magnético do primário foi construído de material ferromagnético enquanto o secundário é uma chapa de alumínio.

Cada pacote de bobinas do primário possui 40cm de comprimento, 5cm de largura, 10cm de profundidade e 15 ranhuras. Cada ranhura tem 10mm de largura e 50 mm de profundidade, a distância entre as ranhuras é de 12mm. As extremidades são mais largas para atenuar os efeitos de extremidade. O motor possui 4 polos, cada um é composto por

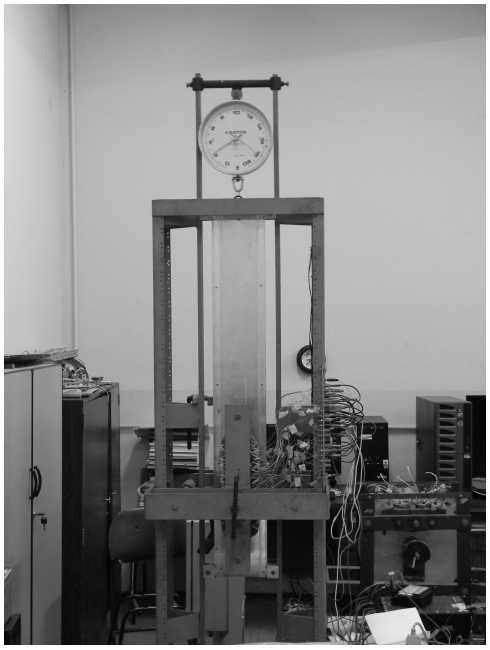

Figura 4.3: Foto do protótipo do MIL.

uma bobina de 150 espiras de fio 19AWG [1]. O passo polar tem 6,5cm e o seu comprimento, excluindo-se as bordas laterais, é de 32cm, a Figura 4.4 mostra a geometria do enrolamento do primário. A Tabela III mostra os parâmetros da máquina linear obtidos através de ensaios.

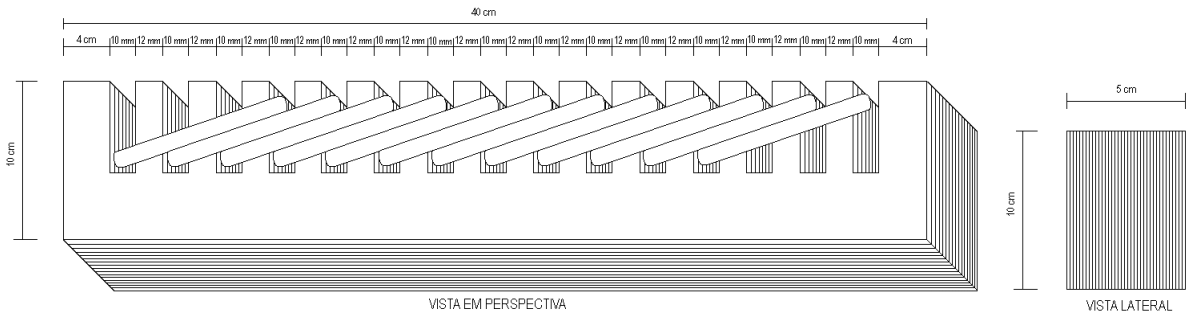

Figura 4.4: Geometria do Enrolamento do Primário.

[1] American Wire Gauge(AWG), é um sistema numérico de tamanho de fios. AWG também é conhecida como Brown & Sharp Gage

[2] Corresponde aos valores de Xab Xbc Xca.

Tabela III: Valores dos parâmetros do motor de indução linear

Parâmetros do MIL		
Parâmetro	Descrição	Valor
Resistências do Estator	Ra	3,8958 Ω
	Rb	4,2171 Ω
	Rc	4,2231 Ω
Resistências do Rotor	RA	7,3021 Ω
	RB	7,3000 Ω
	RC	8,2140 Ω
Reatância de Dispersão	Xda	13,9686 Ω
	Xdb	14,1650 Ω
	Xdc	14,0563 Ω
	XdA	1,5630 Ω
	XdB	-0,0240 Ω
	XdC	1,8010 Ω
Reatância Própria	Xaa	9,5552 Ω
	Xbb	9,1173 Ω
	Xcc	9,7362 Ω
	XAA	9,5552 Ω
	XBB	9,1173 Ω
	XCC	9,7362 Ω
Reatância Mútua	Xss [2]	9,1014 Ω

4.3 Avaliação do Simulador Digital

Uma vez que a assimetria da máquina linear pode ser compensada através do desbalanceamento das tensões de fase que alimentam a máquina, [GONTIJO 2011] realizou uma série de ensaios com o referido protótipo. Afim de avaliar o comportamento do simulador digital construido neste trabalho, foram realizados testes de forma análoga aos ensaios da referência citada. Os parâmetros utilizados no simulador estão expressos no item 4.2. Neste primeiro momento, nenhum desbalanceamento foi aplicado ao sistema de tensões que alimentam o simulador do motor linear.

A Figura 4.5 mostra a comparação entre o perfil da força simulada e a desejada, notando-se que a força simulada não atinge o valor desejado de 25N, conseguindo apenas um valor médio de 19N. Este fato comprova a existência da componente de força negativa, já mencionada anteriormente.

Para minimizar esta componente, utiliza-se o artifício de desbalancear as tensões de fase que alimentam o motor. Matematicamente este desbalanceamento é representado por fatores multiplicativos(Ka Kb Kc) nas tensões balanceadas de fase. A referência [GONTIJO 2011], por um processo manual de tentativas experimentou alguns valores de Ka, Kb e Kc, obtendo-se para cada trio de valores a força do motor. A Tabela IV mostra os resultados obtidos ao se refazer este teste para o simulador digital.

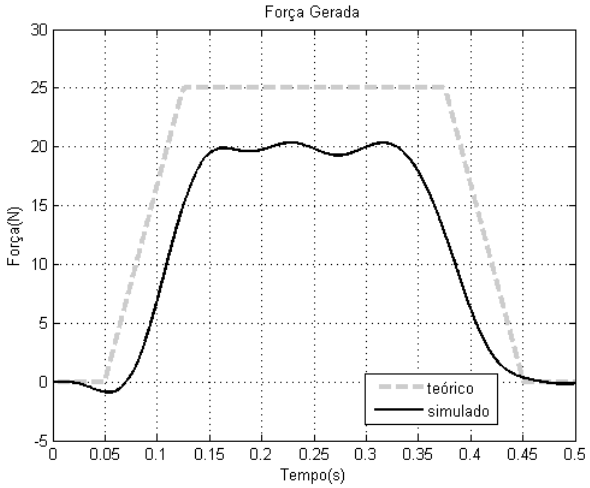

Figura 4.5: Comparação entre as forças simulada e requerida do motor linear.

Diversos valores de desbalanceamento foram testados com o objetivo de mostrar, de forma quantitativa, os efeitos na força eletromagnética gerada pelo simulador do motor linear, e também para comprovar o funcionamento do simulador construido neste trabalho. Os resultados dos testes podem ser visualizados na Tabela IV.

Tabela IV: Resultados obtidos no desbalanceamento das tensões

Controles			Amplitude da Pulso de Força		
Ka	Kb	Kc	Requerida(N)	Simulado(N)	Erro(%)
1,0000	1,0000	1,0000	25	19,8570	20,5722
1,0000	1,0500	1,1000	25	21,8557	12,5771
1,0000	1,1000	1,1000	25	22,5696	9,7216
1,0100	1,1110	1,1110	25	23,0233	7,9070
0,9600	1,0610	1,0610	25	20,9337	16,2651
0,9120	1,1140	1,1140	25	21,6621	13,3516
0,9120	1,1140	1,2250	25	23,1078	7,5687
0,9120	1,1140	1,3000	25	24,0825	3,6700
0,9120	1,1140	1,3500	25	24,7321	1,0717

O simulador se comportou da forma esperada, obtendo-se assim um valor de força próximo ao desejado. Desta forma, o simulador está pronto para desempenhar a função de avaliação do *software* do algoritmo genético.

4.4 Verificação do Conjunto Simulador-AG

Este item apresenta os principais testes realizados com o programa do algoritmo genético, utilizando o simulador digital como função de avaliação. Sua finalidade é comprovar a eficiência do conjunto e mostrar sua diferença de desempenho perante distintos parâme-

tros de entrada. Os dois primeiros testes utilizaram um erro percentual máximo de 1%, enquanto o terceiro utilizou um erro de 0,1%. Os resultados estão apresentados a seguir.

4.4.1 Teste 1

```
Parâmetros de Entrada do AG
n      = 16 ;  %número máximo de individuos.
lim    = 2  ;  %limite numérico dos numeros da população
f      = 25 ;  %força requerida do MIL.
z      = 2  ;  %tamanho do torneio.
Tcruza = 0.75; %taxa de crossover.
Tmuta  = 0.02; %taxa de mutação.
erro   = 0.01; %erro entre as curvas(1%).

-------------Resultados-----------------
Numero de Gerações: 27
Melhor indivíduo: [1,7001 0,7670 1,0091]
Ka = 1,7001
Kb = 0,7670
Kc = 1,0091
Força Simulada: 25,0028 N
Erro: 0,0112 %
```

A Figura 4.6 mostra um gráfico que relaciona a força eletromagnética com a geração em que foi produzida, devido aos parâmetros utilizados o algoritmo levou 27 geração para encontrar o resultado desejado, ou seja, o valor da força com erro menor ou igual a 1% do valor desejado.

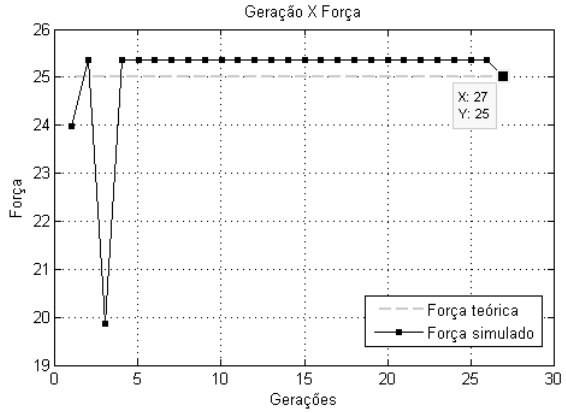

Figura 4.6: Gráfico do teste 1 Geração X Força

4.4.2 Teste 2

```
Parâmetros de Entrada do AG
n      = 20 ;  %número máximo de individuos.
lim    = 2  ;  %limite numérico dos numeros da população
f      = 25 ;  %força requerida do MIL.
z      = 2  ;  %tamanho do torneio.
Tcruza = 0.75; %taxa de crossover.
Tmuta  = 0.02; %taxa de mutação.
Gmax   = 100 ; %número máximo de geração.
erro   = 0.01; %erro entre as curvas(1%).

-------------Resultados-----------------
Número de Gerações: 5
Melhor indivíduo: [1,3703 1,2425 0,7969]
Ka = 1,3703
Kb = 1,2425
Kc = 0,7969
Força Simulada: 25,2491 N
Erro: 0,9964 %
```

O segundo teste apresenta uma alteração no parâmetro "n", referente ao tamanho da população. Nota-se que esta pequena alteração resultou em uma resposta mais rápida, obtida em apenas 5 gerações. Os dois primeiros testes serviram de exemplo para mostrar que a configuração dos parâmetros de entrada geram uma mudança de desempenho, é claro que qualquer um dos outros parâmetros poderiam ter sido utilizados como exemplo, mas ficaria entediante mostrar cada um deles uma vez que o objetivo de demonstrar a diferença de desempenho em cada um dos testes foi cumprida.

O desempenho do teste 2 é apresentado na Figura 4.7. Apesar do resultado do teste 1 ter ficado mais preciso, o teste 2 cumpriu o critério de parada imposto pelo parâmetro "erro".

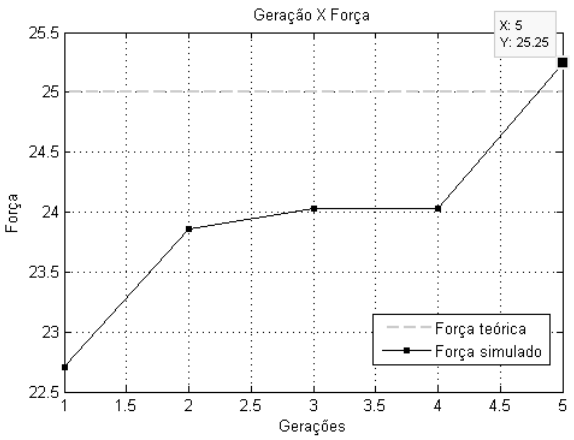

Figura 4.7: Gráfico do teste 2 Geração X Força

4.4.3 Teste 3

```
Parâmetros de Entrada do AG
n      = 30 ;  %número máximo de individuos.
lim    = 2  ;  %limite numérico dos numeros da população
f      = 25 ;  %força requerida do MIL.
z      = 3  ;  %tamanho do torneio.
Tcruza = 0.75; %taxa de crossover.
Tmuta  = 0.02; %taxa de mutação.
Gmax   = 100 ; %número máximo de geração.
erro   = 0.001; %erro entre as curvas(0,1%).

-------------Resultados-----------------
Número de Gerações: 7
Melhor indivíduo: [1,4750 0,3430 1,8602]
Ka = 1,4750
Kb = 0,3430
Kc = 1,8602
Força Simulada: 24,9992 N
Erro: 0,0032 %
```

O terceiro teste mostra que o programa desenvolvido neste trabalho pode encontrar valores de força muito próximos do valor ideal. Alguns dos parâmetros de entrada do algoritmo genético foram alterados, o tamanho da população foi aumentado para poder melhorar as buscas dentro do universo de respostas, e o valor de "z" alterado para melhorar a seleção dos melhores indivíduos.

A Figura 4.8 mostra que com apenas 7 gerações o *software* conseguiu encontrar uma resposta com um valor de erro menor que 0,1%, comprovando assim a eficiência do programa.

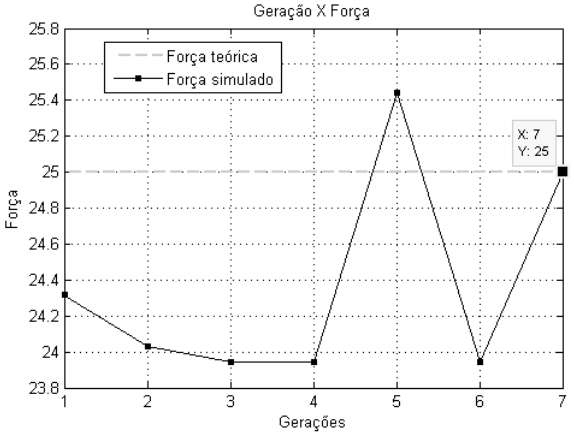

Figura 4.8: Gráfico do teste 3 Geração X Força

As Figuras 4.9 e 4.10 mostram o perfil das tensões de entrada desbalanceadas e a força eletromagnética gerada, respectivamente. Estas figuras foram geradas utilizando-se as constantes de desequilíbrio encontradas no terceiro teste.

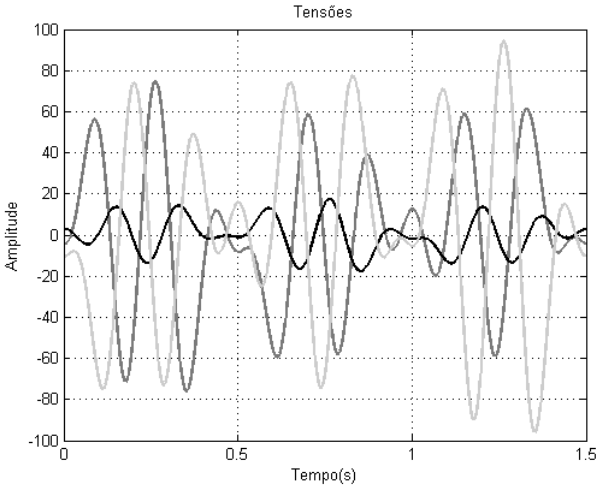

Figura 4.9: Perfil das tensões de entrada desbalancedas do MIL

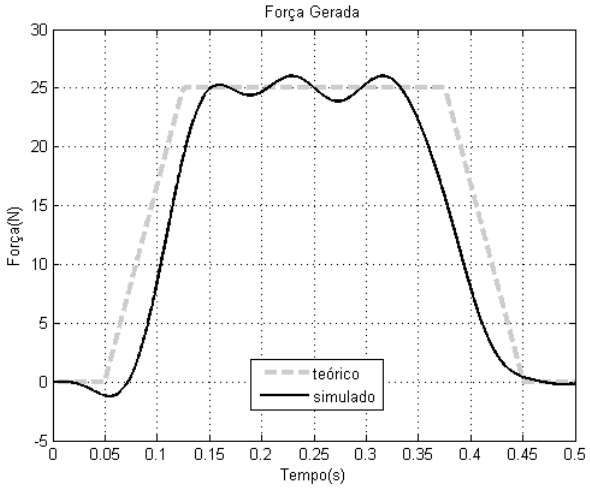

Figura 4.10: Comparação entre a força simulada e a força teórica

4.5 Análise dos Resultados

No item anterior foram apresentados os resultados dos três principais testes realizados com o programa do algoritmo genético.

Estes testes visaram comprovar a funcionalidade do simulador em emular o protótipo do motor de indução linear e comprovar a eficiência e desempenho do *software* do algoritmo genético.

Os resultados apresentados para os testes do programa do algoritmo genético comprovam que o *software* conseguiu encontrar valores para as constantes Ka, Kb e Kc que possibilitaram o motor gerar a força desejada. A Figura 4.10 deixa claro a pequena diferença entre o valor teórico e o valor simulado da força eletromagnética.

Ao analisar o valor do menor erro percentual encontrado por [GONTIJO 2011] e o menor erro encontrado pelo programa de AG, notou-se que houve uma melhora significativa pois o erro que era de 1,8130% passou para 0,0032%

Capítulo 5

Conclusões e Sugestões

Os objetivos propostos neste trabalho foram alcançados, e seus resultados satisfatórios confirmam a viabilidade prática de se aplicar a metodologia desenvolvida pela referência [GONTIJO 2011] para os motores de indução lineares.

A modelagem matemática apresentada neste trabalho para o motor de indução linear, considerando a sua inerente assimetria de ordem construtiva, foi validada através dos resultados finais que comprovam a viabilidade de se alimentar o motor com tensões não senoidais desbalanceadas, com o objetivo de se obter um determinado conjugado desejado. De fato, fica comprovado, neste trabalho, que certos desbalanceamentos de tensões podem, com grande precisão neutralizar o efeito da assimetria construtiva do motor linear.

O outro objetivo do trabalho, que também foi alcançado com êxito, se refere a aplicação de técnicas computacionais de AG na obtenção dos desbalanceamentos de tensão que produzem no motor um determinado conjugado, minimizando ao máximo a diferença entre os valores obtido e desejado.

Apesar de não utilizado, o programa de algoritmo genético está apto para encontrar valores de desbalanceamento para forças com perfis diferentes do trapezoidal, uma vez que, o referido perfil não influi na construção do algoritmo, sendo apenas um parâmetro de comparação. Para a utilização do *software* com outros perfis de força o usuário necessita somente ajustar os parâmetros de entrada do algoritmo genético.

Com base na experiência adquirida no desenvolvimento deste trabalho, tem-se como

sugestão para futuros trabalhos:

1. Desenvolvimento de um modelo de máquina linear que não se restrinja a baixas velocidades, ou seja, o modelo que considera a velocidade do rotor. Desta forma a metodologia se estenderia para qualquer tipo de funcionamento do motor linear.

2. Neste trabalho o erro encontrado entre os valores calculado e desejado para o conjugado é muito pequeno do ponto de vista prático, na ordem de 0,001%. Talvez trabalhando-se com um erro mais realista, ou seja, bem maior que 0,001%, haveria a possibilidade de se encontrar desbalanceamentos menores, podendo simplificar e até ser menos dispendioso o equipamento eletrônico que gera as tensões não senoidais e desbalanceadas que alimentam o motor linear.

3. Incluir outros elementos na codificação do indivíduo no algoritmo genético. Estes novos elementos podem ser parâmetros físicos da máquina linear, permitindo assim que o AG ajude a projetar um protótipo de máquina linear.

Bibliografia

[BARBI 2013] BARBI, I. (2013). Introdução a Teoria de Conversão Eletromecânica de Energia. Disponível em: http://http://www.ivobarbi.com/PDF/livros/cap1.pdf , Acessado em: 08/02/2013.

[BOX 1957] BOX, G. E. P. (1957). "Evolutionary Operation: A Method for Increasing Industrial Productivity". Journal of the Royal Statistical Society,C,6(2),81-101.

[FITZGERALD et al. 2006] FITZGERALD, A. E.; KINGSLEY, C. J.; UMANS, S. D.; (2006). "Máquinas Elétricas". [ISBN: 978-85-60031-04-7], [ISBN: 007-3660094] 6ł ed, Bookman, 648p.

[GOLDBERG 1989] GOLDBERG, D. E.; (2005). "Genetic Algorithms in search, optimization, and machine learning". [ISBN: 0-201-15767-5], 27th Printing, editora ADDISON-WESLEY., 412p.

[GONTIJO 2011] GONTIJO, A. L. (2011). "Modelagem Do Motor De Indução Trifásico Alimentado De Forma Não Tradicional-Conversor Eletromecânico de Conjugado Para Baixas Velocidades". Universidade Federal de Uberlândia,2011.

[GONTIJO et al. 2012] GONTIJO, A. L.; NETO, L. M.; WU, M.; CALIXTO, W.P. (2012). "Linear Induction Machine Fed on a Non Traditional Manner to Generate Force Pulses at Low Speeds". Science Direct - Electric Power Systems Research,2012.

[HOLLAND 1975] HOLLAND, J. H.; (1975). "Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence". [ISBN: 9780472084609], University of Michigan Press.

[LINDEN 2012] LINDEN, R.; (2012). "Algoritmos Genéticos". [ISBN: 978-85-399-0195-1], 3ł ed, editora CIÊNCIA MODERNA Ltda., 475p.

[PONTES 2003] PONTES, R. S. T. (2003). "Modelagem do motor de indução linear baseando-se na equivalência com o motor rotativo desbalanceado". Universidade Federal de Uberlândi,2003.

[PONTES et al. 2000] PONTES, R. S. T.; NETO, L. M.; CAMACHO, J. R.; SILVA, R. V. R.. (2000). "Linear Induction Motor Applied to a Guillotine". MagLev 2000 - The 16th International Conference on Magnetically Levitated Systems and Linear Drives Brazil,2000, p7.

[SIVANANDAM 2010] SIVANANDAM, S. N.; (2010). "Introduction to Genetic Algorithms". [ISBN: 978-3-642-09224-4], [e-ISBN: 978-3-540-73190-0], 1ł ed, editora SPRINGER., 442p.

APÊNDICE A

Programa do bloco "Gerador de Harmônicos"

Código A.1: Código do bloco "Gerador de Harmônicos"

```
%|-CÓDIGO DO BLOCO GERADOR HARMÔNICOS-------|
%| t -- tempo                               |
%| V -- amplitude das tensões               |
%| O -- ângulo das tensões                  |
%| f -- frequencia                          |
%| k -- constantes de desbalanço            |
%|------------------------------------------|

function [va,vb,vc] = Tensoes(t,V,O,f,mult,K)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.
    O  = O*pi/180;
    alpha = 120*pi/180;
    vaa= V(1) *cos(1* (2*pi*f*t + O(1) )) ...
        +V(2) *cos(2* (2*pi*f*t + O(2) )) ...
        +V(3) *cos(3* (2*pi*f*t + O(3) )) ...
        +V(4) *cos(4* (2*pi*f*t + O(4) )) ...
        +V(5) *cos(5* (2*pi*f*t + O(5) )) ...
        +V(6) *cos(6* (2*pi*f*t + O(6) )) ...
        +V(7) *cos(7* (2*pi*f*t + O(7) )) ...
        +V(8) *cos(8* (2*pi*f*t + O(8) )) ...
        +V(9) *cos(9* (2*pi*f*t + O(9) )) ...
        +V(10)*cos(10*(2*pi*f*t + O(10))) ...
        +V(11)*cos(11*(2*pi*f*t + O(11)));
    vbb= V(1) *cos(1 *(2*pi*f*t - alpha + O(1) )) ...
        +V(2) *cos(2 *(2*pi*f*t - alpha + O(2) )) ...
        +V(3) *cos(3 *(2*pi*f*t - alpha + O(3) )) ...
        +V(4) *cos(4 *(2*pi*f*t - alpha + O(4) )) ...
        +V(5) *cos(5 *(2*pi*f*t - alpha + O(5) )) ...
        +V(6) *cos(6 *(2*pi*f*t - alpha + O(6) )) ...
        +V(7) *cos(7 *(2*pi*f*t - alpha + O(7) )) ...
```

```
32          +V(8) *cos(8 *(2*pi*f*t - alpha + O(8) )) ...
33          +V(9) *cos(9 *(2*pi*f*t - alpha + O(9) )) ...
34          +V(10)*cos(10*(2*pi*f*t - alpha + O(10))) ...
35          +V(11)*cos(11*(2*pi*f*t - alpha + O(11)));
36      vcc= V(1) *cos(1 *(2*pi*f*t + alpha + O(1) )) ...
37          +V(2) *cos(2 *(2*pi*f*t + alpha + O(2) )) ...
38          +V(3) *cos(3 *(2*pi*f*t + alpha + O(3) )) ...
39          +V(4) *cos(4 *(2*pi*f*t + alpha + O(4) )) ...
40          +V(5) *cos(5 *(2*pi*f*t + alpha + O(5) )) ...
41          +V(6) *cos(6 *(2*pi*f*t + alpha + O(6) )) ...
42          +V(7) *cos(7 *(2*pi*f*t + alpha + O(7) )) ...
43          +V(8) *cos(8 *(2*pi*f*t + alpha + O(8) )) ...
44          +V(9) *cos(9 *(2*pi*f*t + alpha + O(9) )) ...
45          +V(10)*cos(10*(2*pi*f*t + alpha + O(10))) ...
46          +V(11)*cos(11*(2*pi*f*t + alpha + O(11)));
47  if (mult==0),
48      va = vaa;
49      vb = vbb;
50      vc = vcc;
51  else
52      %k(1) é a constante ka
53      %k(2) é a constante kb
54      %k(3) é a constante kc
55
56      va = K(1) * vaa;
57      vb = K(2) * vbb;
58      vc = K(3) * vcc;
59
60  end
```

APÊNDICE B

S-function do bloco "MIL"

Código B.1: Código da *S-function* do bloco "MIL"

```
% |--SIMULADOR DO MOTOR DE INDUÇÃO LINEAR-----|
% |     Universidade Federal de Uberlândia     |
% |      Faculdade de Engenharia Elétrica      |
% |        autor: Matheus Garcia Soares        |
% |--------------------------------------------|

function [sys,y0,str,ts] = MIB(t,y,u,flag,R,Lda,Ldb,Ldc,LdA,...
    LdB,LdC,P,Laa,Lbb,Lcc,LAA,LBB,LCC,Lab,Lac,Lbc,LAB,LAC,...
    LBC,LaA,LaB,LaC,LbA,LbB,LbC,LcA,LcB,LcC,Pos)

    kaa=1;
switch flag,
  case 0          % condições iniciais
    [sys,y0,str,ts,u] = ValoresIniciais();

  case 1          % calcula derivadas
    sys = Derivadas(t,y,u,R,Lda,Ldb,Ldc,LdA,...
        LdB,LdC,P,Laa,Lbb,Lcc,LAA,LBB,LCC,Lab,...
        Lac,Lbc,LAB,LAC,LBC,LaA,LaB,LaC,LbA,LbB,LbC,LcA,LcB,LcC,Pos);
    % Calculo da derivada

  case 2          % calcula valores discretos
    sys = VariaveisDiscretas(t,y,u,R,Lda,Ldb,Ldc,LdA,...
        LdB,LdC,P,Laa,Lbb,Lcc,LAA,LBB,LCC,Lab,Lac,Lbc,...
        LAB,LAC,LBC,LaA,LaB,LaC,LbA,LbB,LbC,LcA,LcB,LcC,Pos);

    % variavéis discretas

  case 3          % determina a saída
    sys = Saida(t,y,u);
  case {4, 9}   % valores do flag não usados
    sys = [];

  otherwise
    error(['Valor do flag = ',num2str(flag)]); % Erro
```

```
36 end
37 end
38 % fim.
39 %
40 %=============================================================
41 % Inicializa as variáveis
42 % Returna sizes, condições iniciais, e sample times para a
43 % S-function.
44 %=============================================================
45 function [sys,y0,str,ts,u] = ValoresIniciais()
46 sizes = simsizes;
47 sizes.NumContStates  = 6;
48 sizes.NumDiscStates  = 1;
49 sizes.NumOutputs     = 7;
50 sizes.NumInputs      = 6;
51 sizes.DirFeedthrough = 0;
52 sizes.NumSampleTimes = 0;
53 sys = simsizes(sizes);
54 y0=[0 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0];
55 u(6)=0;
56 %u(10)=0;
57 str = [];
58 ts  = [];
59 end
60 %fim ValoresIniciais
61 %
62 %=============================================================
63 % Derivadas
64 % Calcula as derivadas para variáveis contínuas.
65 %=============================================================
66 %
67 function [sys] = Derivadas(t,y,u,R,Lda,Ldb,Ldc,LdA,LdB,LdC,...
68     P,Laa,Lbb,Lcc,LAA,LBB,LCC,Lab,Lac,Lbc,LAB,LAC,LBC,LaA,...
69     LaB,LaC,LbA,LbB,LbC,LcA,LcB,LcC,Pos)
70
71 %Inicializa valores
72     beta = 2*pi/3;
73     ia   = y(1);
74     ib   = y(2);
75     ic   = y(3);
76     iA   = y(4);
77     iB   = y(5);
78     iC   = y(6);
79
80 %Atualiza Vetores
81     %calcula dl/dt
82
83     dlt = zeros(6,6);
84
85    %calcula l
86     l(1,1) = Lda+ Laa;                    %Laa
87     l(1,2) = Lab*cos(beta);               %Lab
88     l(1,3) = Lac*cos(beta);               %Lac
89     l(1,4) = LaA*cos(Pos);                %LaA
90     l(1,5) = LaB*cos(Pos+beta);           %LaB
91     l(1,6) = LaC*cos(Pos-beta);           %LaC
```

```
92
93      l(2,1) = Lab*cos(beta);                 %Lba
94      l(2,2) = Ldb+Lbb;                       %Lbb
95      l(2,3) = Lbc*cos(2*beta);               %Lbc
96      l(2,4) = LbA*cos(Pos-beta);             %LbA
97      l(2,5) = LbB*cos(Pos);                  %LbB
98      l(2,6) = LbC*cos(Pos-(2*beta));         %LbC
99
100     l(3,1) = Lac*cos(beta);                 %Lca
101     l(3,2) = Lbc*cos(2*beta);               %Lcb
102     l(3,3) = Ldc+Lcc;                       %Lcc
103     l(3,4) = LcA*cos(Pos+beta);             %LcA
104     l(3,5) = LcB*cos(Pos+(2*beta));         %LcB
105     l(3,6) = LcC*cos(Pos);                  %LcC
106
107     l(4,1) = LaA*cos(Pos);                  %LAa
108     l(4,2) = LbA*cos(Pos-beta);             %LAb
109     l(4,3) = LcA*cos(Pos+beta);             %LAc
110     l(4,4) = LdA+LAA;                       %LAA
111     l(4,5) = LAB*cos(beta);                 %LAB
112     l(4,6) = LAC*cos(beta);                 %LAC
113
114     l(5,1) = LaB*cos(Pos+beta);             %LBa
115     l(5,2) = LbB*cos(Pos);                  %LBb
116     l(5,3) = LcB*cos(-Pos-(2*beta));        %LBc
117     l(5,4) = LAB*cos(beta);                 %LBA
118     l(5,5) = LdB+LBB;                       %LBB
119     l(5,6) = LBC*cos(2*beta);               %LBC
120
121     l(6,1) = LaC*cos(Pos-beta);             %LCa
122     l(6,2) = LbC*cos(Pos-(2*beta));         %LCb
123     l(6,3) = LcC*cos(-Pos);                 %LCc
124     l(6,4) = LAC*cos(beta);                 %LCA
125     l(6,5) = LBC*cos(2*beta);               %LCB
126     l(6,6) = LdC+LCC;                       %LCC
127
128     %calcula V
129     V = u(1:6);
130     V = (V'/l)';
131     %calcula R+dl/dt
132     aux = (((R + dlt))'/l)';
133     %coloca V em u e divide Tc por J
134     u(1:6) = V;
135     %coloca R+dl/dt em A
136     A=aux;
137     %faz o calculo
138     U = u(1:6);
139     yy = y(1:6);
140     sys = U - (A*yy);
141 end
142 % fim da rotina Derivadas.
143 %
144 %=========================================================
145 % Dados para o controle do motor
146 % Dados de estado discretos - calculo do torque de saída
147 %=========================================================
```

```
148 %
149 function sys = VariaveisDiscretas(t,y,u,R,Lda,Ldb,Ldc,LdA,LdB,...
150     LdC,P,Laa,Lbb,Lcc,LAA,LBB,LCC,Lab,Lac,Lbc,LAB,LAC,LBC,LaA,...
151     LaB,LaC,LbA,LbB,LbC,LcA,LcB,LcC,Pos)
152
153     %Inicializa valores
154         beta = 2*pi/3;
155         ia   = y(1);
156         ib   = y(2);
157         ic   = y(3);
158         iA   = y(4);
159         iB   = y(5);
160         iC   = y(6);
161     %calcula dl/dt
162         dlt(1,1) = 0;
163         dlt(1,2) = 0;
164         dlt(1,3) = 0;
165         dlt(1,4) = -LaA*sin(Pos);
166         dlt(1,5) = -LaB*sin(Pos+beta);
167         dlt(1,6) = -LaC*sin(Pos-beta);
168
169         dlt(2,1) = 0;
170         dlt(2,2) = 0;
171         dlt(2,3) = 0;
172         dlt(2,4) = -LbA*sin(Pos-beta);
173         dlt(2,5) = -LbB*sin(Pos);
174         dlt(2,6) = -LbC*sin(Pos-(2*beta));
175
176         dlt(3,1) = 0;
177         dlt(3,2) = 0;
178         dlt(3,3) = 0;
179         dlt(3,4) = -LcA*sin(Pos+beta);
180         dlt(3,5) = -LcB*sin(Pos+(2*beta));
181         dlt(3,6) = -LcC*sin(Pos);
182
183         dlt(4,1) = -LaA*sin(Pos);
184         dlt(4,2) = -LbA*sin(Pos-beta);
185         dlt(4,3) = -LcA*sin(Pos+beta);
186         dlt(4,4) = 0;
187         dlt(4,5) = 0;
188         dlt(4,6) = 0;
189
190         dlt(5,1) = -LaB*sin(Pos+beta);
191         dlt(5,2) = -LbB*sin(Pos);
192         dlt(5,3) = -LcB*sin(Pos+(2*beta));
193         dlt(5,4) = 0;
194         dlt(5,5) = 0;
195         dlt(5,6) = 0;
196
197         dlt(6,1) = -LaC*sin(Pos-beta);
198         dlt(6,2) = -LbC*sin(Pos-(2*beta));
199         dlt(6,3) = -LcC*sin(Pos);
200         dlt(6,4) = 0;
201         dlt(6,5) = 0;
202         dlt(6,6) = 0;
203
```

```
204         aux = [ia;ib;ic;iA;iB;iC];
205         %sys = (P/4)*aux'*dlt*aux;
206
207         sys = (P/4)*((2*pi)/0.4)*aux'*dlt*aux;
208     end
209     % fim da rotina VariaveisDiscretas.
210     %
211     %=============================================================
212     % Saída
213     % Retorna o vetor de saída da S-Function
214     %=============================================================
215     %
216     function sys = Saida(t,y,u)
217     sys = y;
218     end
219     % fim da rotina Saida
```

APÊNDICE C

Programa do Algortimo Genético

Código C.1: Programa principal do algoritmo genético

```
% |-ALGORITMO GENÉTICO - OTIMIZAÇÃO DO MOTOR DE INDUÇÃO LINEAR----------|
% | UNIVERSIDADE FEDERAL DE UBERLÂDIA - FACULDADE DE ENGENHARIA ELÉTRICA |
% | AUTOR -  Matheus Garcia Soares                                       |
% |----------------------------------------------------------------------|

clc; clear all; warning off;

%******* PARÂMETROS DO AG ***************************
n       = 20 ;        %número máximo de individuos.
lim     = 1.5;        %limite numérico dos numeros da população
f       = 25 ;        %força requerida do MIL.
z       = 2  ;        %tamnho do torneio.
Tcruza  = 0.75;       %taxa de cruzamento.
Tmuta   = 0.10;       %taxa de mutação.
Gmax    = 100 ;       %número máximo de geração.
erro    = 0.01;       %erro entre as curvas.

%**************gera a população inicial**************
[pop] = popula(n , lim);

%**************variáveis de inicialização************
G         = 0 ; %contador de gerações
MaMeInd   = [];
erroP     = 1 ;

while (erroP > erro) %condição de parada do algoritmo é o erro
    G = G + 1;

[fit ,Mind,Mmed,error] = fitness(pop,f); %aptidão da população

[winners]= selection(pop,fit ,k);          % seleção por torneio
[cross] = simplecross(winners,Tcruza);     %crossover
[mutantes] = xgene(cross,Tmuta);           %mutação

```

```
35 pop = mutantes;                          %nova populção
36 erroP = error;                           %verificação do erro
37
38 MaMeInd(end+1,:) = Mind;                  %melhores indivíduos
39
40 end
```

Código C.2: Função Gera População

```
1 %|-FUNÇÃO GERA POPULAÇÃO-----------------------|
2 %| Cria uma população para o AG               |
3 %| npop -- é o tamanho da população           |
4 %| lim  -- é o limite numérico da população   |
5 %|-------------------------------------------|
6
7 function [pop] = popula(npop , lim)
8   pop = 0 + (lim-0).*rand(npop,3);        %gera a população
9 end
```

Código C.3: Função Fitness

```
1  % |-FUNÇÃO FITNESS ------------------------------------|
2  % | Testa a aptidão de cada indivíduo da população |
3  % | povo -- população                              |
4  % | force -- força a ser atingida                  |
5  % |-----------------------------------------------|
6
7  function [fit ,Mind,Mmed,error] = fitness(povo,force)
8  fitlst = zeros(size(povo),1);       % cria lista de apitidões(vazia)
9
10 % declara funções que não fazem parte do bloco EMBEDDED
11 eml.extrinsic('load_system', 'set_param','sim', 'get_param','get');
12
13 mdl = 'modelo/SIMMXMIB22';           % caminho para o modelo
14 mdl_g = 'SIMMXMIB22/gerador';        % caminho para o bloco gerador
15 load_system(mdl);                    % carrega o simulador MIL em Simulink
16 best = 0;
17 error = 1;
18 for i=1:1:(size(povo))
19
20  % muda o valor dos parâmetros ka kb kc dentro do Simulink
21  set_param(mdl_g,'K',mat2str(povo(i,:),4));
22
23  % define a saida do simulink
24  simOut = sim(mdl,'SaveOutput','on','OutputSaveName','Te');
25
26  yout = simOut.get('Te');            % atribui o vetor Te para yout
27  media = mean(yout(1700:3200)) ;     % realiza a média do patamar central
28  apt = 1 / ((force - media)^2) ;     % calculo da aptidão(erro absoluto)
29  fitlst(i) = apt;                    % preenche lista de aptidões
30
31  if best < apt
32       best = apt;                    %melhor individuo
33       Mind = povo(i,:);
```

```
34        Mmed = media;
35        error = abs((force - media)/force); %erro percentual
36    end
37  end
38
39  fit=fitlst;                          %lista de aptidões
40
41  end
```

Código C.4: Função de Seleção

```
1   %|-FUNÇÃO DE SELEÇÃO VIA TORNEIO ------------------------------|
2   %| Seleciona indivíduos para o crossover                       |
3   %| glads -- populão que será aplicada a seleção                |
4   %| power -- aptidão dos componentes da população               |
5   %| numk  -- número de indivíduos por partida do torneio        |
6   %|-------------------------------------------------------------|
7
8   function  [winners]= selection(glads,power,numk)
9   tamanho = length(glads);                 %tamanho do vetor da população
10  winlst = zeros(tamanho,3);               %vetor de vencedores
11
12  for i=1:1:tamanho
13        win=0;                             %vencedor da rodada
14        wpow=0;                            %apitidão do vencedor da rodada
15    for g=1:1:numk                         %aplica a luta do torneio
16        lutador = randi(tamanho);
17
18        if win == 0
19            win = glads(lutador,:);
20            wpow = power(lutador);
21        else
22            if wpow > power(lutador)
23                win;
24                wpow;
25            else
26                win = glads(lutador,:);
27                wpow = power(lutador);
28            end
29        end
30  end
31  winlst(i,:) = win;
32
33  end
34
35  winners = winlst;                        %lista de vencedores
```

Código C.5: Função de Crossover

```
%|-FUNÇÃO DE CROSSOVER - CROSSOVER SIMPLES ----------|
%| crossover simples com apenas um ponto de corte.   |
%| population --  população a ser cruzada.           |
%| rate  -------  taxa de crossover.                 |
%|---------------------------------------------------|

function [cross] = simplecross(population,rate)

for i=1:2:length(population)     %aplica o crossover em toda a população
    probe = randi(1000)/1000;

    if rate >= probe             %probabilidade de ocorrer o crossover
        [pai1,pai2] = popcross(population(i,:),population(i+1,:));
        population(i,:) = pai1;
        population(i+1,:) =pai2;
    else                         %caso o crossover não ocorra
        population(i,:) = population(i,:);
        population(i+1,:) = population(i+1,:);
    end
   cross = population;           %retorna a população após o crossover
end

%os argumentos de entrada são dois individuos da população
%retorna dois filhos,onde o ponto de corte é definido aleatoriamente.

% função que realiza o crossover.
function [pai1,pai2] = popcross(ind1,ind2)
pontoc = randi(length(ind1));
pai1 = ind1;
pai2 = ind2;

 if pontoc == 3
     pai1(1,3) = ind2(1,3);
     pai2(1,3) = ind1(1,3);
 else
     pai1(1,1:pontoc) = ind2(1,1:pontoc);
     pai2(1,1:pontoc) = ind1(1,1:pontoc);

end
```

Código C.6: Função de Mutação

```
%|-FUNÇÃO DE MUTAÇÃO - MUTAÇÃO SIMPLES ------|
%| Mutação simples utilizando numeros reais  |
%| personas -- população que sofrerá mutação |
%| xrate    -- taxa de mutação               |
%|-------------------------------------------|

function [mutantes] = xgene(personas,xrate)
tam = length(personas);          % tamanho da população
genes = length(personas(1,:));  % tamanho dos genes de cada indivíduo

    for i = 1:1:tam
        for j = 1:1:genes
            mut = randi(1000)/1000; %taxa aleatória de mutação
            if xrate >= mut          %ocorre mutação
                personas(i,j) = 0 + (1.5-0).*rand(1,1);

            else                     %não ocorre mutação
                personas(i,j) = personas(i,j);
            end
        end
    end

    mutantes = personas;     %nova população depois da mutação
end
```