



Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Programa de Pós-graduação em Engenharia Elétrica

Um Ambiente Computacional para Modelagem Simbólica de Sistemas Físicos Lineares

Gilbert Azevedo da Silva

Natal / RN - Brasil
Janeiro 2005

Gilbert Azevedo da Silva

Um Ambiente Computacional para Modelagem Simbólica de Sistemas Físicos Lineares

Orientador:

Prof. D. Sc. André Laurindo Maitelli

Tese submetida ao Programa de Pós-graduação
em Engenharia Elétrica da Universidade Federal do
Rio Grande do Norte como parte dos requisitos
necessários para obtenção do título de Doutor em
Ciências

Natal / RN - Brasil

Janeiro 2005

Gilbert Azevedo da Silva

Um Ambiente Computacional para Modelagem Simbólica de Sistemas Físicos Lineares

Banca Examinadora:

Prof. **André Laurindo Maitelli**, D. Sc. - UFRN (Orientador)

Prof. **Eduardo Bráulio Wanderley Netto**, D. Sc. - CEFET/RN

Prof. **Roberto Célio Limão de Oliveira**, D. Sc. - UFPA

Prof. **Dario José Aloise**, D. Sc. - UFRN

Prof. **Luiz Affonso Henderson Guedes de Oliveira**, D. Sc. - UFRN

Natal / RN - Brasil

Janeiro 2005

Índice de Assuntos

LISTA DE FIGURAS	VI
LISTA DE TABELAS	X
LISTA DE SÍMBOLOS	XI
RESUMO	XIV
ABSTRACT	XV
AGRADECIMENTOS	XVI
DEDICATÓRIA	XVII
1. INTRODUÇÃO	1
1.1. Motivação	1
1.2. Educação em Controle	2
1.3. Modelagem de Sistemas Físicos	5
1.4. Objetivo	7
1.5. Trabalhos Correlatos e Contribuição	10
1.6. Organização do Trabalho	12
2. FUNDAMENTOS TEÓRICOS	13
2.1. Introdução	13
2.2. Grafos	13
2.2.1. Conceito	13
2.2.2. Definições	14
2.2.2.1. Grafos Simples	14
2.2.2.2. Vértices e Ramos Adjacentes	15

2.2.2.3. Subgrafo	15
2.2.2.4. Caminho.....	15
2.2.2.5. Ciclo	16
2.2.2.6. Grafo Conexo, Conexidade e Componentes Conexas	16
2.2.2.7. Árvore e Co-Árvore	17
2.2.2.8. Floresta e Co-Floresta	17
2.2.2.9. Grafo Direcionado e Valorado	18
2.2.2.10. Circuito ou Laço	19
2.3. Modelagem de Sistemas Físicos	19
2.3.1. Variáveis Generalizadas	19
2.3.2. Representação do Sistema.....	21
2.3.3. Generalização dos Elementos do Sistema	22
2.3.4. Relações Constitutivas.....	23
2.3.4.1. Propriedades Constitutivas das Fontes de Energia	23
2.3.4.2. Propriedades Constitutivas dos Armazenadores de Energia	24
2.3.4.3. Propriedades Constitutivas dos Dissipadores de Energia.....	26
2.3.4.4. Propriedades Constitutivas dos Moduladores de Uma Porta	27
2.3.4.5. Propriedades Constitutivas dos Elementos de Duas Portas	28
2.3.4.6. Exemplos de Propriedades Constitutivas.....	29
2.3.5. Grafo de Ligação	29
2.3.6. Relações de Interconectividade	34
2.4. Diagramas de Fluxo de Sinal	34
2.4.1. Conceito.....	34
2.4.2. Equações do DFS.....	35
2.4.3. Definições	35
2.4.4. Álgebra do DFS	37
2.5. Regra de Mason.....	37
2.5.1. Fórmula do Ganho de Mason	38
2.5.2. DFS Fechado	40
2.6. Funções de Sistema na Forma Simbólica.....	42
2.6.1. Funções de Variáveis Generalizadas.....	42
2.6.2. Funções Simbólicas	43
2.6.3. Aplicações das Funções Simbólicas	44
2.6.3.1. Introspecção	44
2.6.3.2. Melhoria na Precisão de Cálculos.....	44
2.6.3.3. Análise de Sensibilidade.....	45
2.6.3.4. Sensibilidade em Larga Escala.....	46

3. MODELAGEM DE SOFTWARE	47
3.1. Introdução	47
3.2. Paradigmas de Programação.....	47
3.2.1. Programação Estruturada	47
3.2.2. Programação Modular.....	48
3.2.3. Programação Orientada ao Objeto	49
3.2.3.1. Classificação, Abstração e Instanciação.....	49
3.2.3.2. Classes e Objetos.....	50
3.2.3.3. Atributos ou Propriedades	50
3.2.3.4. Métodos ou Comportamentos.....	51
3.2.3.5. Visibilidade.....	51
3.2.3.6. Herança e Polimorfismo.....	52
3.3. Linguagem de Modelagem Unificada.....	55
3.3.1. Diagrama de Casos de Uso	55
3.3.2. Diagrama de Classes.....	58
3.3.3. Diagrama de Estruturas Compostas	60
3.3.4. Diagrama de Colaboração	61
3.3.5. Diagrama de Atividades	62
 4. MODELAGEM DO MODSYM	 64
4.1. Introdução	64
4.2. Objetivos e Funções do Software	64
4.3. Diagrama de Casos de Uso	65
4.4. Diagrama de Estruturas Compostas	68
4.5. Diagramas de Classes	70
4.5.1. Interface Gráfica de Modelagem.....	70
4.5.2. Estruturas de Dados	78
4.5.2.1. Estrutura de Dados do Desenho Gráfico	78
4.5.2.2. Estrutura de Dados do Grafo de Ligação e DFS	81
4.5.2.3. Estrutura de Dados do Algoritmo Grafo de Ligação – DFS	84
4.5.2.4. Estrutura de Dados da Regra de Mason.....	84
4.5.2.5. Estrutura de Dados da Função de Transferência	86
4.6. Diagramas de Colaborações.....	87
4.6.1. Obtenção do Grafo de Ligação do Sistema.....	87
4.6.2. Cálculo da Função de Transferência do Sistema	87

4.6.3. Obtenção do DFS a partir do Grafo de Ligação.....	89
4.6.4. Cálculo da Função de Transferência do Grafo de Ligação.....	89
4.6.5. Cálculo da Função de Transferência do DFS	90
 5. AMBIENTE COMPUTACIONAL PROPOSTO	 92
5.1. Introdução	92
5.2. Interface Gráfica de Modelagem.....	92
5.2.1. Objetivo.....	92
5.2.2. Área de Montagem de Desenhos Gráficos	93
5.2.2.1. Vértices, Conexões e Conectores.....	94
5.2.2.2. Modelagem de Sistemas Elétricos	96
5.2.2.3. Modelagem de Sistemas Mecânicos Translacionais.....	98
5.2.2.4. Modelagem de Sistemas Mecânicos Rotacionais	100
5.2.2.5. Modelagem de Sistemas Hidráulicos	102
5.2.2.6. Modelagem de Sistemas Físicos com Acopladores.....	103
5.2.2.7. Modelagem de Sistemas Físicos em Malha Fechada	107
5.2.3. Área de Montagem de Grafos de Ligação	108
5.2.4. Área de Montagem de Diagramas de Fluxo de Sinal.....	111
5.3. Interface Gráfica de Resultados.....	112
5.3.1. Grafo de Ligação do Sistema	113
5.3.2. Diagrama de Fluxo de Sinal do Sistema	114
5.3.3. Funções de Transferência do Sistema	116
5.3.4. Resultados da Regra de Mason.....	119
5.3.5. Função de Sensibilidade Paramétrica.....	119
 6. ALGORITMOS.....	 121
6.1. Introdução	121
6.2. Algoritmo Sistema – Grafo	121
6.2.1. Objetivo.....	121
6.2.2. Etapas do Algoritmo.....	122
6.2.2.1. Definição da Estrutura de Dados do Sistema	123
6.2.2.2. Remoção de Vértices Redundantes	125
6.2.2.3. Adição de Vértices Ocultos	126
6.2.2.4. Numeração dos Vértices Elétricos	127
6.2.2.5. Numeração dos Vértices Mecânicos Translacionais.....	128
6.2.2.6. Numeração dos Vértices Mecânicos Rotacionais	133

6.2.2.7. Numeração dos Vértices Hidráulicos	133
6.2.2.8. Associações em Série e Paralelo	134
6.2.2.9. Definição da Estrutura de Dados do Grafo de Ligação.....	134
6.3. Algoritmo Grafo – DFS	136
6.3.1. Objetivo.....	136
6.3.2. Etapas do Algoritmo.....	137
6.3.2.1. Definição da Estrutura de Dados do Grafo	137
6.3.2.2. Obtenção do Grafo Auxiliar Não-Orientado	139
6.3.2.3. Verificação dos Pré-Requisitos	140
6.3.2.4. Obtenção das Componentes Conexas do Grafo Auxiliar.....	141
6.3.2.5. Determinação das Variáveis do DFS	141
6.3.2.6. Obtenção das Matrizes de Equações	144
6.3.2.7. Algoritmo de Busca.....	152
6.3.2.8. Inexistência de Solução	154
6.3.2.9. Determinação das Transmitâncias do DFS.....	155
7. RESULTADOS	157
7.1. Introdução	157
7.2. Exemplos de Grafos de Ligação e Funções de Transferência	157
7.2.1. Sistema Elétrico	157
7.2.2. Motor DC com Controlador PID.....	158
7.2.3. Sistema Mecânico Translacional.....	159
7.3. Exemplos de DFS e Funções de Transferência.....	160
7.3.1. Sistema Elétrico com Duas Fontes de Energia	160
7.3.2. Sistema de Controle de Nível.....	162
7.4. Exemplo Final	166
8. CONCLUSÕES	167
REFERÊNCIAS BIBLIOGRÁFICAS.....	170

Lista de Figuras

1.1. Etapas de estudo de um sistema de controle	3
1.2. Diagramas gráficos de um circuito elétrico	6
1.3. Fluxograma de execução do <i>ModSym</i>	8
2.1. Uma representação gráfica para o Grafo G_1	14
2.2. Grafo G_2	14
2.3. Grafo G_3 – Subgrafo de G_2	15
2.4. Grafo G_4 – Grafo não conexo.	16
2.5. Grafo G_5 – Árvore geradora de G_2	17
2.6. Grafo G_6 – Co-árvore de G_2	17
2.7. Grafo G_7 – Floresta geradora de G_4	18
2.8. Grafo G_8 – Co-floresta de G_4	18
2.9. Grafo G_9 – Grafo direcionado e valorado.....	19
2.10. Representação do sistema físico.	21
2.11. Fonte de esforço: (a) Símbolo, (b) Relação constitutiva.	23
2.12. Fonte de fluxo: (a) Símbolo, (b) Relação constitutiva.	24
2.13. Armazenador de esforço: (a) Símbolo, (b) Relação constitutiva.	25
2.14. Armazenador de fluxo: (a) Símbolo, (b) Relação constitutiva.	26
2.15. Dissipador de energia: (a) Símbolo, (b) Relação constitutiva.	27
2.16. Elemento de duas portas.....	28
2.17. Convenção para elementos passivos: (a) Elemento, (b) Orientação do ramo.	31
2.18. Convenção para fontes de esforço: (a) Elemento, (b) Orientação do ramo.....	31
2.19. Convenção para fontes de fluxo: (a) Elemento, (b) Orientação do ramo.....	31
2.20. Convenção para elementos de duas portas: (a) Elemento, (b) Ramos.	32
2.21. Grafo de ligação de um sistema elétrico.....	32
2.22. Grafo de ligação de um sistema mecânico.	33
2.23. Exemplo de DFS.....	35
2.24. Exemplo de DFS.....	36
2.25. Exemplo de DFS fechado.....	40

2.26. Exemplo de sistema físico.	42
2.27. Introspecção de um sistema elétrico.....	44
3.1. Decomposição de um problema.	48
3.2. Composição de um problema.	49
3.3. Representação gráfica de uma classe.....	50
3.4. Exemplo de classe com atributos.	51
3.5. Exemplo de classe com atributos e métodos.....	51
3.6. Exemplo de visibilidade.	52
3.7. Exemplo de herança e polimorfismo.....	53
3.8. Diagrama de Casos de Uso.....	56
3.9. Diagrama de Classes.....	59
3.10. Diagrama de Estruturas Compostas.	60
3.11. Agrupamento de Colaborações.	60
3.12. Diagrama de Colaboração.	62
3.13. Diagrama de Atividades.....	63
4.1. Diagrama de Casos de Uso do <i>ModSym</i>	66
4.2. Diagrama de Estruturas Compostas do <i>ModSym</i>	69
4.3. Diagrama de Classes da interface gráfica de modelagem.....	71
4.4. Especializações da classe <i>TVisualObject</i>	74
4.5. Especializações da classe <i>TVOEleComp</i>	75
4.6. Especializações da classe <i>TVOHidComp</i>	75
4.7. Especializações da classe <i>TVOMctComp</i>	76
4.8. Especializações da classe <i>TVOAcp</i>	76
4.9. Especializações da classe <i>TVOGrfComp</i>	77
4.10. Estrutura de dados do desenho gráfico.	79
4.11. Estrutura de dados dos grafos de ligação e diagramas de fluxo de sinal.....	82
4.12. Estrutura de dados do algoritmo grafo de ligação – DFS.	84
4.13. Estrutura de dados da regra de Mason.....	85
4.14. Estrutura de dados da função de transferência.	86
4.15. Diagrama de colaboração: Caso de uso Obter o Grafo de Ligação.	87
4.16. Diagrama de colaboração: Caso de uso FT Simbólica do Sistema.	88
4.17. Diagrama de colaboração: Caso de uso Obter o DFS.....	89
4.18. Diagrama de colaboração: Caso de uso FT Simbólica do Grafo.	90
4.19. Diagrama de colaboração: Caso de uso FT Simbólica do DFS.	90
5.1. Área de montagem de desenhos gráficos.	94
5.2. Exemplo de sistema elétrico.	97

5.3. Exemplo de sistema mecânico translacional.	100
5.4. Exemplo de sistema mecânico rotacional.	101
5.5. Desenho gráfico de sistema hidráulico.	103
5.6. Exemplo de sistema hidráulico.	103
5.7. Modelo de Motor DC controlado pela armadura.	104
5.8. Exemplo de sistema físico com acoplador.	104
5.9. Modelo de sistema de controle de nível em malha aberta.	105
5.10. Exemplo de sistema físico com acoplador e fonte controlada.	106
5.11. Exemplo de sistema físico em malha fechada.	107
5.12. Área de montagem de grafos de ligação.	108
5.13. Exemplo de grafo de ligação.	110
5.14. Área de montagem de diagramas de fluxo de sinal.	111
5.15. Motor DC em malha fechada com controlador Proporcional.	113
5.16. Grafo de ligação do Motor DC.	113
5.17. Vértices do grafo de ligação do Motor DC.	114
5.18. Ramos do grafo de ligação do Motor DC.	114
5.19. DFS do Motor DC.	115
5.20. Matriz de equações do Motor DC.	116
5.21. Seleção das variáveis da função de transferência.	116
5.22. Função de transferência simbólica do motor DC.	117
5.23. Função de transferência simbólica fatorada do motor DC.	117
5.24. Lista de símbolos da função de transferência do motor DC.	117
5.25. Função de transferência numérica do motor DC.	118
5.26. Simulação do motor DC realizada no SINCON.	118
5.27. Lista de laços de ordem superior da regra de Mason.	119
5.28. Sensibilidade da FT do motor DC em relação a K_p	119
6.1. Diagrama de atividades do algoritmo Sistema-Grafo.	122
6.2. Motor DC em malha fechada com controlador Proporcional.	123
6.3. Motor DC em malha fechada com controlador Proporcional.	127
6.4. Ilustração da condição C1.	131
6.5. Ilustração da condição C2.	131
6.6. Ilustração da condição C3.	132
6.7. Ilustração da condição C4.	132
6.8. Numeração dos vértices do grafo.	134
6.9. Associações de massas em série e paralelo.	134
6.10. Grafo de ligação do Motor DC.	135

6.11. Diagrama de atividades do algoritmo Grafo-DFS.	137
6.12. Componentes conexas do grafo de ligação do Motor DC.....	141
6.13. Matriz de impedâncias para o exemplo do Motor DC.	146
6.14. Matriz de fontes controladas para o exemplo do Motor DC.	147
6.15. Matriz de transformação e rotação para o exemplo do Motor DC.....	148
6.16. Matriz de compatibilidade de esforço para o exemplo do Motor DC.....	149
6.17. Matriz de continuidade de fluxo para o exemplo do Motor DC.....	151
6.18. Exemplos de grafos de ligação sem DFS.....	155
7.1. Exemplo de Sistema elétrico.....	157
7.2. Função de transferência do sistema elétrico.	158
7.3. Motor DC com controlador PID.	158
7.4. Grafo de ligação do motor DC com controlador PID.....	159
7.5. Função de transferência do Motor DC com controlador PID.....	159
7.6. Exemplo de sistema mecânico translacional.	160
7.7. Função de transferência do sistema mecânico translacional.....	160
7.8. Exemplo de sistema elétrico com duas fontes de energia.	161
7.9. DFS do sistema elétrico com duas fontes de energia.....	161
7.10. Função de transferência entre a tensão em R1 e a tensão em v1.	161
7.11. Função de transferência entre a corrente em R1 e a corrente em i1.....	162
7.12. Sistema de controle de nível.....	162
7.13. Função de transferência do sistema de controle de nível.....	163
7.14. DFS do sistema de controle de nível.	164
7.15. Laços no DFS fechado do sistema de controle de nível.....	164
7.16. Laços de ordem superior no DFS fechado do sistema de controle de nível.....	165
7.17. Sistema elétrico com 12 malhas resistivas.	166
7.18. Função de transferência entre a corrente e tensão na fonte v1.	166

Lista de Tabelas

2.1. Variáveis de esforço e fluxo.....	20
2.2. Elementos básicos de uma porta.....	22
2.3. Conversores de energia.....	23
2.4. Relações constitutivas.	29
2.5. Laços do DFS.	38
2.6. Laços introduzidos no DFS.....	41
4.1. Métodos da classe TSistema.	80
4.2. Atributos da classe TElemento.	80
4.3. Atributos da classe TConexao.	80
4.4. Métodos da classe TGrafo.....	83
4.5. Atributos da classe TVertice.	83
4.6. Atributos da classe TRamo.....	83
4.7. Atributos da classe TGanho.....	83
5.1. Elementos elétricos.....	96
5.2. Elementos mecânicos translacionais.	98
5.3. Elementos mecânicos rotacionais.....	101
5.4. Elementos hidráulicos.....	102
5.5. Elementos acopladores.	103
5.6. Elementos do grafo de ligação.	109
5.7. Elementos do diagrama de fluxo de sinal.	112
6.1. Lista de elementos do sistema.	124
6.2. Lista de conexões do sistema.....	124
6.3. Lista de vértices do grafo.....	138
6.4. Lista de ramos do grafo.	139
6.5. Variáveis do algoritmo Grafo-DFS.	142
6.6. Número de equações das matrizes.	152

Lista de Símbolos

Δ	Determinante do DFS
Δ_C	Determinante do DFS fechado
Δ_K	K-ésimo cofator do DFS
λ	Fluxo magnético concatenado
Γ_f	Momento do sistema fluido
A	Matriz de equações do grafo de ligação
A₁	Matriz de impedâncias
A₂	Matriz de fontes controladas
A₃	Matriz de transformação e giração
A₄	Matriz de compatibilidade de esforço
A₅	Matriz de continuidade de fluxo
A_{5N}	Matriz de continuidade de fluxo resumida
b	Número de ramos do grafo
B	Amortecimento
c	Conexidade do grafo
C	Capacitância
C_f	Capacitância fluida
e	Variável generalizada de esforço
E(elemento)	Esforço no elemento
e₁	Esforço na primeira porta de um elemento
e₂	Esforço na segunda porta de um elemento
e_a	Esforço acumulado
f	Variável generalizada de fluxo
F	Força
F(elemento)	Fluxo no elemento
f₁	Fluxo na primeira porta de um elemento
f₂	Fluxo na segunda porta de um elemento
f_a	Fluxo acumulado
fc	Número de fontes controladas

f_e	Número de fontes de energia
G	Relação de giração
G_{DFS}	Ganho do DFS
$G(S)$	Função de transferência
G_k	Ganho do k-ésimo caminho direto do DFS
i	Corrente elétrica
J	Inércia
K	Elasticidade
K_e	Ganho de esforço
K_F	Ganho de força
K_f	Ganho de fluxo
K_i	Ganho de corrente
K_p	Ganho de pressão
K_q	Ganho de vazão
K_t	Ganho de torque
K_v	Ganho de tensão
K_V	Ganho de velocidade linear
K_w	Ganho de velocidade angular
L	Indutância
M	Massa
n	Número de vértices do grafo
N	Relação de transformação
$n_{AcpEleMcr}$	ID da classe do acoplador elétrico-mecânico rotacional
n_{DFS}	Número de variáveis do DFS
n_E	Número de variáveis de entrada do DFS
n_{Ele}	ID do sistema elétrico
n_{EleEsf}	ID da fonte de tensão
$n_{EleEsfC}$	ID da fonte controlada de tensão
n_{EleInd}	ID do indutor
n_{EleMcr}	ID do sistema elétrico-mecânico rotacional
n_{EleRes}	ID do resistor
n_{EleVrt}	ID do vértice elétrico
$n_{GrfDpsP1}$	ID da primeira porta do elemento genérico de duas portas
$n_{GrfDpsP2}$	ID da segunda porta do elemento genérico de duas portas
n_{GrfEsf}	ID da fonte de esforço
$n_{GrfEsfC}$	ID da fonte controlada de esforço
n_{GrfFlx}	ID da fonte de fluxo
$n_{GrfFlxC}$	ID da fonte controlada de fluxo

$n_{GrfGrdP1}$	ID da primeira porta do elemento girador
$n_{GrfGrdP2}$	ID da segunda porta do elemento girador
n_{GrfImp}	ID da impedância
$n_{GrfTrfP1}$	ID da primeira porta do elemento transformador
$n_{GrfTrfP2}$	ID da segunda porta do elemento transformador
n_{Mcr}	ID do sistema mecânico rotacional
$n_{McrCapRes}$	ID do elemento Inércia + Atrito
n_{McrRef}	ID da referência mecânica rotacional
n_{MctCap}	ID do elemento Massa
$n_{MctCapRes}$	ID do elemento Massa + Atrito
n_{MctRef}	ID da referência mecânica translacional
n_{MctVrt}	ID do vértice mecânico translacional
n_s	Número de variáveis de saída do DFS
P	Pressão
P_{mt}	Momento do sistema mecânico translacional
q	Carga elétrica
Q	Vazão
R	Resistência
R_f	Resistência fluida
ρ	Variável generalizada de esforço ou fluxo
T	Torque
T_{pi}	Função de sensibilidade paramétrica
t	Número de elementos de duas portas
$U(S)$	Entrada da função de transferência
v	Tensão elétrica
V	Velocidade linear
V_f	Volume
VS	Vetor solução do algoritmo de busca
W	Velocidade angular
x	Deslocamento
X	Vetor de variáveis do DFS
Xe	Variáveis de entrada do DFS
Xs	Variáveis de saída do DFS
Y	Vetor de variáveis do DFS com dados dos vértices terminais
$Y(S)$	Saída da função de transferência
Z	Impedância
z	Número de impedâncias

Resumo

Este trabalho propõe um ambiente computacional aplicado ao ensino de sistemas de controle, denominado de *ModSym*. O software implementa uma interface gráfica para a modelagem de sistemas físicos lineares e mostra, passo a passo, o processamento necessário à obtenção de modelos matemáticos para esses sistemas.

Um sistema físico pode ser representado, no software, de três formas diferentes. O sistema pode ser representado por um diagrama gráfico a partir de elementos dos domínios elétrico, mecânico translacional, mecânico rotacional e hidráulico. Pode também ser representado a partir de grafos de ligação ou de diagramas de fluxo de sinal.

Uma vez representado o sistema, o *ModSym* possibilita o cálculo de funções de transferência do sistema na forma simbólica, utilizando a regra de Mason. O software calcula também funções de transferência na forma numérica e funções de sensibilidade paramétrica.

O trabalho propõe ainda um algoritmo para obter o diagrama de fluxo de sinal de um sistema físico baseado no seu grafo de ligação. Este algoritmo e a metodologia de análise de sistemas conhecida por *Network Method* permitiram a utilização da regra de Mason no cálculo de funções de transferência dos sistemas modelados no software.

Abstract

This work proposes a computational environment for teaching of control systems, called *ModSym*. The software implements a graphical interface for linear physical systems modeling and shows, step by step, the required process to obtain mathematical models for these systems.

A physical system can be represented at the software in three different ways. The physical system can be represented in a graphical diagram using elements of electrical, mechanical translational, mechanical rotational and fluid domains. The systems can be defined using linear graphs and signal-flow graphs, too.

Once represented the system, *ModSym* calculates the system transfer functions in symbolic form, using Mason's rule. The software also calculates transfer functions in numeric form and parametric sensibility functions.

The work also proposes an algorithm to obtain a signal-flow graph of a physical system based on its linear graph. This algorithm and the systems analyses methodology know as *Network Method* allowed the use of Mason's rule to calculate transfer functions of systems modeled in the software.

Agradecimentos

Ao professor André Laurindo Maitelli, pela maneira paciente, amigável e estimuladora com que conduziu o desenvolvimento deste trabalho.

Aos professores Dario José, Luiz Affonso, e Roberto Limão pelas sugestões dadas no exame de qualificação.

Ao professor Agostinho Júnior, grande amigo, pelas idéias trocadas durante o desenvolvimento do trabalho.

Aos professores da UFRN, pelos conhecimentos e experiências repassados.

Ao CEFET-RN, pela liberação parcial concedida para a realização do curso.

Aos meus pais Olívio Borges da Silva e Aynar Azevedo da Silva e aos meus irmãos Jefferson Azevedo da Silva e George Azevedo da Silva pelo carinho e incentivo que sempre me oferecem.

À minha esposa Adriana pelo carinho, amor e compreensão sempre presentes.

Dedico este trabalho à minha esposa
Adriana e à minha filha Mariana.

1. Introdução

1.1. Motivação

A Engenharia de Controle, [1-3], tem contribuído significativamente para o progresso tecnológico da humanidade. Os benefícios decorrentes das aplicações do controle automático estão presentes na maioria dos processos industriais modernos, [2], onde as técnicas de controle são largamente empregadas. Grande parte das inovações tecnológicas consumidas pelo homem moderno é produto direto ou indireto das aplicações da Engenharia de Controle.

Sistemas de controle de temperatura, pressão, umidade, fluxo e velocidade, por exemplo, permitem que a fabricação de produtos em indústrias petroquímicas, têxteis e alimentícias ocorra em um ambiente de produção mais próximo do ideal, evitando desperdícios e melhorando a qualidade dos produtos. Em várias indústrias, os sistemas de controle automático e os robôs substituem o homem nas tarefas realizadas em ambientes insalubres, salvaguardando vidas humanas. A automação também substitui o homem nas tarefas repetitivas, aumentando a escala de produção industrial e, conseqüentemente, diminuindo o custo dos produtos.

Aliada ao progresso das telecomunicações, a Engenharia de Controle tem permitido à humanidade a exploração do universo. O lançamento de sondas meteorológicas, por exemplo, fundamental para o entendimento do clima terrestre, não seria possível sem a tecnologia desenvolvida pela Engenharia de Controle. A manipulação remota de robôs, [2], como os *Spirit* e *Opportunity* que exploram atualmente a superfície de Marte, seria inimaginável sem os conhecimentos adquiridos com o estudo dos sistemas de controle.

No nosso cotidiano, os sistemas de controle automático são utilizados cada

vez mais. Aplicações do controle estão presentes, por exemplo, em sistema de navegação aérea, sistemas de injeção eletrônica de combustível e em equipamentos aplicados à área médico-hospitalar, como órgãos artificiais, [2]. Algumas destas aplicações são de vital importância para o homem moderno; outras, como os órgãos artificiais, são indispensáveis para a manutenção da vida de muitas pessoas.

É fundamental salientar também que a Engenharia de Controle tem sido fortemente impulsionada pelos avanços da tecnologia digital, assim como ocorre em diversas outras áreas da ciência. Aliada aos avanços tecnológicos, a computação se consolida como uma ciência indispensável ao desenvolvimento dos sistemas de controle automático e se aplica de forma bastante enfática ao ensino de disciplinas desta engenharia.

1.2. Educação em Controle

A Engenharia de Controle está centrada no estudo da dinâmica de sistemas físicos. Dentre as principais atividades desenvolvidas por esta engenharia, estão a modelagem, a análise, a simulação e o projeto de sistemas de controle, [2, 3].

O diagrama da figura 1.1 mostra um fluxo de realização destas atividades na análise e síntese de um sistema de controle. Conforme pode ser visto, a etapa de modelagem é especialmente importante para este processo. Nesta etapa, um modelo matemático para o sistema físico é formulado, de forma que seu comportamento possa ser descrito a partir de um conjunto de equações. É baseado neste modelo que as demais etapas do estudo são realizadas. A partir do modelo matemático, os engenheiros de controle podem analisar e simular o comportamento do sistema físico em um computador digital e verificar vários aspectos de sua dinâmica, como resposta a estímulos, estabilidade, robustez a perturbações e a variações de parâmetros, sem a necessidade de sua implementação física. Com base na análise e simulação do modelo, várias técnicas de controle podem ser aplicadas de forma a obter um comportamento específico desejado para o sistema

físico. Isto justifica a necessidade de definição de um modelo matemático que represente com a maior precisão possível a dinâmica do sistema.

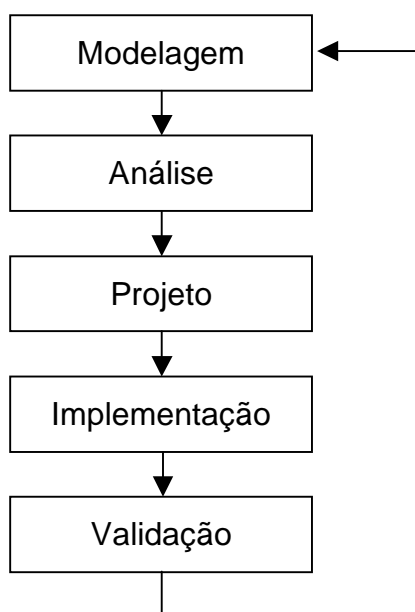


Figura 1.1. Etapas de estudo de um sistema de controle.

A tarefa de modelagem de sistemas demanda normalmente um grande esforço dos estudantes desta engenharia. A obtenção de modelos matemáticos para um sistema envolve normalmente o estudo da dinâmica de cada um dos seus elementos e das interações resultantes de suas interconexões. Em geral, o estudante necessita compreender um conjunto razoável de leis da física, como mecânica, eletricidade e magnetismo, por exemplo, bem como dominar ferramentas de cálculo avançado e de álgebra linear que lhe permitam formular e solucionar as equações resultantes do processo de modelagem.

A experiência em sala de aula tem comprovado uma problemática: grande parte dos estudantes dos cursos de controle apresenta dificuldades na sistemática utilizada na obtenção de modelos matemáticos de um sistema físico. Esta realidade é influenciada também por algumas deficiências oriundas de cursos básicos de matemática e de física, as quais têm contribuído para a formação de uma base científica pobre que dificulta o desenvolvimento das competências e habilidades requeridas para o engenheiro de controle.

Uma solução que vem sendo constantemente adotada na prática pedagógica

de cursos de controle tem sido o uso de ferramentas computacionais que auxiliem o processo de ensino e aprendizagem. Aliás, as tecnologias da informação e comunicação têm sido amplamente discutidas na comunidade. Seja no âmbito pedagógico, [4], ou na engenharia, [5, 6], a utilização do computador na prática de ensino tem sido constantemente debatida e analisada. Um grande número de educadores em controle tem avaliado o rumo da educação frente às crescentes inovações tecnológicas e, em particular, ao uso do computador e das ferramentas de projeto direcionadas a esta área, [7]. Parece ser um consenso que o uso da tecnologia da informação traz uma série de benefícios para o processo de ensino, facilitando as etapas de aprendizado do estudante e despertando-lhe interesses para a ciência.

Esta nova visão de ensino tem trazido à tona também toda uma discussão a respeito de estruturas curriculares de cursos de controle. Para muitos educadores envolvidos no processo, é fundamental e inevitável a formulação de novos currículos onde a ferramenta computacional seja mais evidenciada, [8], contrapondo-se às estruturas curriculares clássicas que enfatizam principalmente as ferramentas matemáticas e a teoria de controle.

Entretanto, apesar dos recentes avanços na área de computação, é importante ressaltar que grande parte dos softwares relacionados à Engenharia de Controle é deficiente na parte educacional. Uma análise realizada em grande parte das ferramentas disponíveis, [9], tem mostrado que, em geral, elas apresentam uma interface homem-computador pouco amigável. Essas interfaces, normalmente baseadas em interação de comandos, dificultam o processo educacional e tornam tedioso e enfadonho o uso da ferramenta computacional.

Este cenário, a vivência no ensino de disciplinas nas áreas de controle e informática e a experiência no desenvolvimento de ferramentas computacionais voltadas para o ensino em controle, [10, 11], motivaram o desenvolvimento de um ambiente computacional direcionado à área de modelagem de sistemas físicos com o objetivo de facilitar o processo de aprendizagem de estudantes nesta atividade específica da Engenharia de Controle.

1.3. Modelagem de Sistemas Físicos

Na atividade de modelagem, os engenheiros de controle costumam utilizar diagramas gráficos para representar os sistemas físicos. Dentre os tipos de diagramas mais comumente usados, destacam-se os desenhos gráficos dos sistemas, os diagramas de blocos, [2], os grafos de ligação, [12], e os diagramas de fluxo de sinal (DFS), [2].

A figura 1.2 mostra quatro possíveis representações gráficas para o circuito elétrico conhecido por divisor de tensão. O desenho gráfico do sistema, visto na figura 1.2a, é a forma mais natural e intuitiva de representação de um sistema físico. Neste desenho, os elementos físicos são representados por ícones gráficos que simbolizam cada um dos elementos que o compõem e as interações entre os elementos são representadas por segmentos de reta interligando os componentes. Os desenhos gráficos são normalmente utilizados para definir o sistema físico a ser estudado.

Os diagramas de blocos, os grafos de ligação e os diagramas de fluxo de sinal são utilizados para auxiliar o engenheiro de controle na formulação do modelo matemático do sistema. Nos diagramas de blocos, cada bloco quantifica uma relação existente entre variáveis do sistema físico, representando normalmente a função de transferência de um subsistema. A interconexão dos subsistemas, baseada nas relações existentes entre eles, produz o diagrama que descreve a dinâmica do sistema físico como um todo. Um diagrama de blocos para o circuito divisor de tensão pode ser visto na figura 1.2b.

O grafo de ligação do sistema elétrico, figura 1.2c, é um dígrafo, [13, 14], cujos vértices representam os pontos de tensões iguais no circuito e os ramos, cada um dos componentes elétricos. Em conjunto com as leis das malhas e dos nós de Kirchhoff e a teoria de circuitos, [12], os grafos de ligação são amplamente usados na análise e projeto de sistemas elétricos. Finalmente, o circuito elétrico pode ser representado pelo DFS visto na figura 1.2d. Neste dígrafo, os vértices representam as variáveis do sistema e os ramos as relações entre estas variáveis.

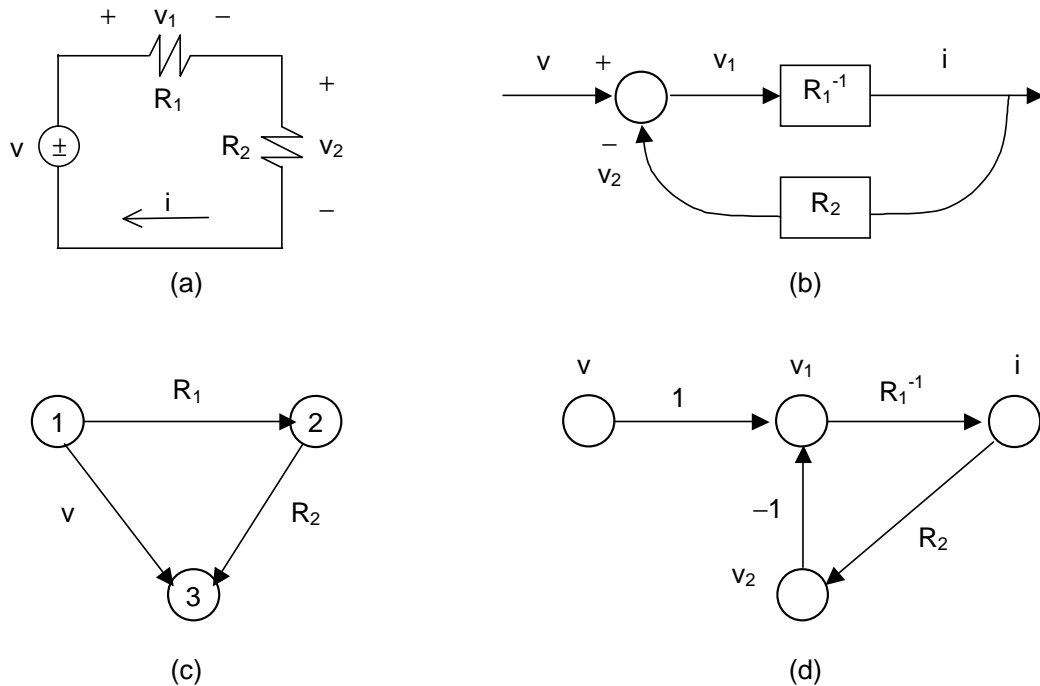


Figura 1.2. Diagramas gráficos de um circuito elétrico.

Apesar das particularidades inerentes a cada um destes diagramas, todos objetivam facilitar a obtenção de modelos matemáticos que descrevam o comportamento do sistema físico. Os tipos mais comuns de representações matemáticas para sistemas físicos lineares são: o sistema de equações diferenciais, que descreve o comportamento das variáveis de saída do sistema físico no domínio do tempo; a representação por variáveis de estados, que descreve a dinâmica de um conjunto de variáveis do sistema físico, também no domínio do tempo, e a função de transferência, que quantifica uma relação entre duas variáveis do sistema no domínio de Laplace, [2].

A função de transferência, em particular, é de grande importância para o estudo de sistemas de controle. Na forma numérica, esta função possibilita a observação do desempenho do sistema através de simulações e na forma simbólica, permite a realização de diversas análises de desempenho do sistema, como análises de estabilidade e sensibilidade paramétrica, [15].

Dos diagramas apresentados anteriormente, o diagrama de fluxo de sinal é o mais adequado para o cálculo de funções de transferência na forma simbólica. Algoritmos encontrados na literatura, como a regra de Mason, [16, 17], são

comprovadamente eficientes no cálculo computacional desta função. Embora seja possível calcular a função de transferência de sistemas físicos a partir dos diagramas de blocos e dos grafos de ligação, o cálculo de tais funções através destes diagramas torna-se excessivamente trabalhoso à medida que a complexidade do sistema físico aumenta, por requerer uma redução gradual desses diagramas.

Motivado por este aspecto, o presente trabalho propõe um conjunto de algoritmos para a obtenção de diagramas de fluxo de sinal de um sistema físico baseado no seu desenho gráfico. A implementação destes algoritmos permite o cálculo de funções de transferência na forma simbólica de sistemas físicos, partindo do seu desenho gráfico, através da regra de Mason.

1.4. Objetivo

É objetivo deste trabalho propor um ambiente computacional para a área de modelagem de sistemas físicos lineares. O software apresentado no trabalho, denominado de *ModSym* (Modelagem Simbólica de Sistemas Físicos), [18-20], permite realizar as seguintes atividades relacionadas à modelagem de um sistema de controle:

- Modelar um sistema físico com base no seu desenho gráfico realizado a partir de um conjunto de elementos dos sistemas elétrico, mecânico translacional, mecânico rotacional e hidráulico;
 - Obter o grafo de ligação deste sistema físico;
 - Obter o diagrama de fluxo de sinal deste sistema;
 - Modelar sistemas físicos a partir de grafos de ligação e de diagrama de fluxo de sinal;
 - Calcular funções de transferência entre variáveis do sistema físico na forma simbólica;
 - Calcular a sensibilidade da função de transferência em relação a algum parâmetro do sistema.
-

O diagrama da figura 1.3 mostra um fluxograma de execução do ambiente.

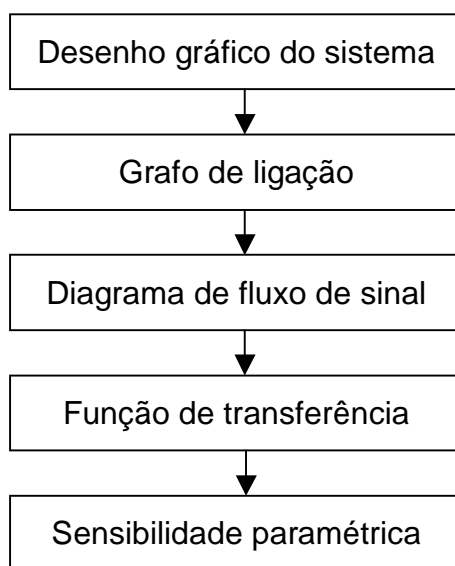


Figura 1.3. Fluxograma de execução do *ModSym*.

Na etapa de desenho gráfico do sistema, o *ModSym* disponibiliza ao usuário uma interface gráfica bastante amigável, que permite modelar um sistema conectando ícones gráficos que representam os elementos físicos utilizando basicamente o *mouse*. São disponibilizados ao usuário 48 elementos físicos dos sistemas elétricos, mecânicos e hidráulicos, além de acopladores que possibilitam interconectar elementos de sistemas físicos diferentes.

A partir do desenho gráfico do sistema, o software implementa um algoritmo para construir o seu grafo de ligação. Apesar de tais diagramas serem comumente utilizados na análise circuitos elétricos, a utilização destes grafos pode ser estendida e aplicada ao estudo do comportamento de sistemas de outros domínios da física. Uma generalização de variáveis e elementos físicos possibilita também a obtenção de modelos matemáticos para sistemas mecânicos, hidráulicos, térmicos e magnéticos utilizando a mesma metodologia empregada na solução de circuitos elétricos: o Método de Rede, [12].

O *ModSym* permite também que o usuário modele o sistema físico definindo diretamente o seu grafo de ligação. Para isto, disponibiliza uma segunda interface gráfica, com a mesma facilidade de uso da interface de desenho do sistema, onde elementos generalizados como fontes de energia, dissipadores, acumuladores,

transformadores e acopladores podem ser interconectados na forma de um grafo. A partir deste grafo, podem ser realizadas as etapas subseqüentes apresentadas no fluxograma da figura 1.3.

Uma vez obtido o grafo de ligação do sistema, o *ModSym* propõe e implementa um algoritmo para a construção de diagramas de fluxo de sinal a partir deste grafo, [21]. Com isso, o software alia a capacidade do Método de Rede em representar graficamente sistemas de diversos domínios da física com a capacidade computacional dos diagramas de fluxo de sinal no cálculo de funções de transferência de sistemas na forma simbólica. A implementação deste algoritmo, que não foi encontrado na pesquisa bibliográfica realizada, foi de fundamental importância para o desenvolvimento deste trabalho. O *ModSym* permite também que o usuário defina o DFS de sistema físico através de uma terceira interface gráfica. A partir do DFS, também podem ser realizadas as etapas subseqüentes apresentadas no fluxograma da figura 1.3.

Para calcular funções de transferência do sistema físico na forma simbólica, o *ModSym* implementa a regra de Mason, que é um algoritmo clássico e comprovadamente eficiente no cálculo de tais funções. Em seguida, o software possibilita o cálculo da sensibilidade paramétrica normalizada da função de transferência em relação a qualquer um dos parâmetros da função.

O *ModSym* calcula ainda funções de transferência na forma numérica, possibilitando exportar tais funções para o software SINCON, [10-11], onde podem ser realizadas as atividades de análise e síntese do sistema de controle.

Para reforçar o caráter educacional do *ModSym*, o software disponibiliza ao usuário os resultados produzidos por cada um dos algoritmos implementados de forma que o estudante possa acompanhar, passo a passo, todas as etapas do processamento, desde o desenho gráfico do sistema até a obtenção da função de transferência. Esse processamento possui uma complexidade relevante e, por envolver normalmente um grande número de manipulações matemáticas e de manipulações em grafos, é bastante susceptível a erros quando realizado manualmente. Portanto, é bastante recomendável ao estudante a utilização de

alguma ferramenta auxiliar de cálculo na realização de tais tarefas.

O ambiente computacional do *ModSym* é implementado em linguagem *Object Pascal*, [22, 23], para as plataformas *Windows* e *Linux*. Compatível com as ferramentas de desenvolvimento de aplicativos *Borland Delphi* e *Borland Kylix*, o software desenvolvido possui o código fonte aberto e disponível para estudantes e pesquisadores incorporarem novas funcionalidades. A implementação do código fonte do software é realizada segundo os paradigmas da linguagem de modelagem unificada de objetos (UML), [24, 25], visando facilitar o seu entendimento e a implementação de novas características por outros pesquisadores.

1.5. Trabalhos Correlatos e Contribuição

A pesquisa bibliográfica realizada para avaliar a viabilidade de desenvolvimento deste trabalho pode ser didaticamente resumida em três grupos.

O primeiro grupo está relacionado com a pesquisa voltada para a educação em controle. O objetivo era identificar os anseios da comunidade científica no que se refere à utilização do computador como ferramenta de ensino. Conforme comentado na Seção 1.2, grande parte dos educadores em controle acredita que o uso do computador no processo de aprendizagem traz benefícios à atividade pedagógica. Uma das vantagens desta abordagem é dispor ao aluno uma gama muito maior de problemas em controle para estudo, independente da implementação física ou não desses sistemas. Outro aspecto relevante que foi verificado está relacionado à necessidade de disponibilizar ao aluno uma interface homem-máquina amigável, [9], com o objetivo de facilitar sua interatividade com a ferramenta computacional. Isto, de certa forma, motivou-nos a construir de uma interface gráfica bem elaborada para o ambiente computacional proposto.

O segundo grupo está relacionado aos softwares desenvolvidos para a área de controle. Grande parte dos softwares científicos encontrados está voltada para a tarefa de simulação de sistemas físicos e utilizam sempre expressões na forma numérica. Neste grupo, podemos listar: *CODAS (Control System Design and*

Simulation), [26], EDCON (*Educational Control System Analysis and Design Program*), [27], PCS (*Process Control Simulation*), [26] e SINCON, [28].

Alguns softwares comerciais foram também avaliados. O software *Electronics Workbench*, [29], tem grande funcionalidade na área de circuitos elétricos, apresenta uma interface bastante amigável e semelhante à interface implementada no *ModSym*. Entretanto, este software trabalha apenas com sistemas elétricos e eletrônicos e na forma numérica. Os softwares *20 Sim*, [30], e *Simplorer*, [31], apresentam grande funcionalidade na atividade de modelagem de sistemas, permitindo a definição de sistemas físicos a partir de seu desenho gráfico, onde disponibilizam componentes dos sistemas elétricos e mecânicos, principalmente, de diagramas de blocos e de *bond graphs*, [12]. Estes softwares entretanto não utilizam grafos de ligação e diagramas de fluxo de sinal, não calculam a função de transferência do físico na forma simbólica e estão mais voltados para a atividade de simulação dos sistemas.

Dentre os softwares encontrados na pesquisa, os que apresentaram maior semelhança com o *ModSym* foram o *Lgraph*, [32], o *MASD*, [33], e o *SAPWIN*, [34]. Os dois primeiros utilizam o conceito de grafos de ligação para obter a matriz de estados do sistema na forma simbólica. O *Lgraph* é desenvolvido para estações de trabalho *Unix* e o *MASD* implementa uma interface de integração com o *Mathematica* para processamento simbólico das equações de estados. Esses softwares entretanto não apresentam uma interface que modele o sistema físico a partir do seu desenho gráfico e não calculam a função de transferência do sistema. Dos três, o *SAPWIN* é o único que calcula funções de transferência nas formas simbólica e numérica a partir de diagramas de desenho gráfico; entretanto, ele utiliza apenas componentes do sistema elétrico, não permitindo a modelagem de sistemas mecânicos, hidráulicos e, logicamente, de sistemas com componentes de mais de um domínio físico como a modelagem realizada pelo *ModSym*. O *SAPWIN* também não permite a modelagem de sistemas através de grafos de ligação e de diagramas de fluxo de sinal.

Finalmente, a pesquisa foi direcionada aos algoritmos usados na

manipulação dos diagramas gráficos dos sistemas físicos. São encontrados na literatura, algoritmos para conversão de diagramas de blocos em diagramas de fluxo de sinal, [3], algoritmos para redução de diagramas de fluxo de sinal, [35], e para cálculo do ganho destes diagramas, [36]. Entretanto, não foi encontrado na pesquisa nenhum algoritmo para obter o diagrama de fluxo de sinal de um sistema físico a partir do seu grafo de ligação.

Em resumo, acreditamos que o presente trabalho apresenta duas contribuições para a área de controle: o ambiente computacional *ModSym* que implementa o cálculo de funções de transferência na forma simbólica de sistemas físicos lineares baseado no desenho gráficos destes sistemas, e o algoritmo de conversão de grafos de ligação em diagramas de fluxo de sinal.

1.6. Organização do Trabalho

Este trabalho está dividido em 8 capítulos. Este capítulo serve como apresentação de todo trabalho desenvolvido, enfocando a motivação, o objetivo e a contribuição do trabalho. O Capítulo 2 apresenta os principais fundamentos teóricos utilizados no seu desenvolvimento referentes à Teoria de Grafos e à Engenharia de Controle. O Capítulo 3 mostra aspectos relevantes ao trabalho na área de Engenharia de Software. O Capítulo 4 apresenta os fundamentos mais importantes da modelagem do ambiente computacional proposto. O Capítulo 5 apresenta o ambiente computacional, sua funcionalidade e aspectos essenciais de sua implementação. O Capítulo 6 mostra os principais algoritmos implementados no software, relacionados à modelagem de sistemas físicos. O Capítulo 7 mostra exemplos e resultados obtidos. O Capítulo 8 conclui o trabalho, apresentando algumas sugestões para futuros estudos.

2. Fundamentos Teóricos

2.1. Introdução

Este capítulo apresenta os fundamentos teóricos mais importantes para o desenvolvimento deste trabalho. São apresentados conceitos básicos sobre grafos [13, 14], diagramas de fluxo de sinal, [2, 3], regra de Mason, [15, 16], e funções de sistemas de controle na forma simbólica, [2, 15], destacando-se os conceitos aplicados no trabalho.

O capítulo apresenta ainda a abordagem utilizada na modelagem dos sistemas físicos, [12], mostrando as suas principais características e delimitando o universo de aplicação do ambiente computacional proposto: a modelagem de sistemas físicos lineares.

2.2. Grafos

Os grafos são aplicados neste trabalho na representação de sistemas físicos sob a forma de grafos de ligação, [12], e diagramas de fluxo de sinal. Algumas definições da teoria de grafos relevantes ao trabalho são apresentadas em seguida.

2.2.1. Conceito

Um grafo $G(V, R)$ é definido pelo par de conjuntos V e R , onde V é um conjunto não vazio de elementos - denominados vértices ou nós - e R é um conjunto de pares (v, w) , com v e $w \in V$ - denominados de ramos ou arestas.

Os símbolos $v_1, v_2, v_3, \dots, v_n$ serão usados para representar os n vértices do grafo e os símbolos $r_1, r_2, r_3, \dots, r_b$, para representar os b ramos. A ordem do grafo, identificada por n , é definida exatamente pelo seu número de vértices. Os vértices v_i

e v_j associados ao ramo r_k são chamados de vértices terminais do ramo $r_k = (v_i, v_j)$.

Na forma gráfica, os vértices são representados por circunferências identificadas pelo índice do vértice e os ramos, por segmentos de reta, interligando seus vértices terminais, identificados pelos índices dos ramos. A figura 2.1 mostra uma representação para o grafo G_1 , de ordem 3, definido por:

$$V = \{ v_1, v_2, v_3 \}$$

$$R = \{ r_1 = (v_1, v_2), r_2 = (v_2, v_3), r_3 = (v_3, v_1) \}$$

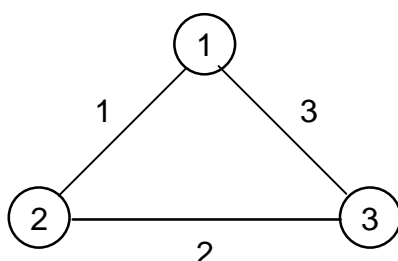


Figura 2.1. Uma representação gráfica para o Grafo G_1 .

2.2.2. Definições

2.2.2.1. Grafos Simples

Dois ou mais ramos do grafo são ditos paralelos quando possuem o mesmo par de vértices terminais. Além disso, quando os vértices terminais de um ramo são idênticos, ou seja $r_k = (v_i, v_i)$, este ramo forma um laço próprio do vértice v_i . Os grafos são denominados de grafos simples quando não possuem ramos paralelos ou laços próprios.

O grafo G_1 , visto na figura 2.1, é um grafo simples. No grafo G_2 , visto na figura 2.2, os ramos r_3 e r_4 são paralelos; portanto, ele não é um grafo simples.

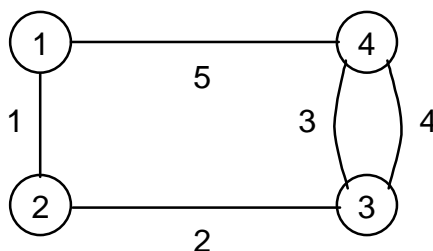


Figura 2.2. Grafo G_2 .

2.2.2.2. Vértices e Ramos Adjacentes

Dois vértices são ditos adjacentes quando são os vértices terminais de algum ramo. Dois ramos são adjacentes quando possuem um vértice terminal comum.

No grafo G_2 , visto na figura 2.2, os vértices v_2 e v_4 são adjacentes a v_1 e os ramos r_2 e r_5 são adjacentes a r_1 .

2.2.2.3. Subgrafo

Um grafo $G'(V', R')$ é um subgrafo de $G(V, R)$ se as condições a seguir são válidas:

- V' é subconjunto de V .
- R' é subconjunto de R .
- Qualquer ramo $r'_k = (v_i, v_j) \in R'$ implica em $v_i, v_j \in V'$.

O grafo G_3 , visto na figura 2.3, é um exemplo de subgrafo do grafo G_2 .

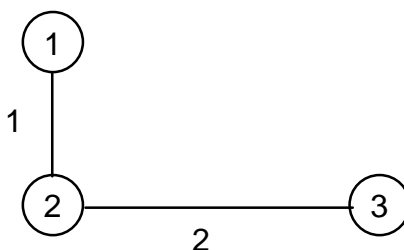


Figura 2.3. Grafo G_3 – Subgrafo de G_2 .

2.2.2.4. Caminho

O conjunto de ramos $R' = (r_1, r_2, \dots, r_b)$ de um grafo $G(V, R)$ forma um caminho entre os vértices v_i, v_j do grafo se e somente se as seguintes condições forem satisfeitas:

- Ramos consecutivos r_i e r_{i+1} de R' possuem sempre um vértice terminal comum;
- Nenhum vértice do grafo é terminal para mais de dois ramos de R' ;
- Cada um dos vértices v_i, v_j do grafo é terminal para somente um ramo em R' .

Os conjuntos $R_1' = \{ r_1, r_2, r_3 \}$, $R_2' = \{ r_1, r_2, r_4 \}$ e $R_3' = \{ r_5 \}$ constituem todos os possíveis caminhos entre os vértices v_1 e v_4 do grafo G_2 , visto na figura 2.2.

2.2.2.5. Ciclo

Obtém-se um ciclo quando um conjunto de ramos $R' = (r_1, r_2, \dots, r_b)$ forma um caminho cujos vértices inicial e final são idênticos, ou seja, $v_i = v_j$. No grafo G_2 , visto na figura 2.2, por exemplo, os conjuntos $R_4' = \{ r_1, r_2, r_3, r_5 \}$ e $R_5' = \{ r_1, r_2, r_4, r_5 \}$ formam todos os possíveis ciclos passando pelo vértices v_1 .

2.2.2.6. Grafo Conexo, Conexidade e Componentes Conexas

Um grafo é conexo se existir pelo menos um caminho entre todos os possíveis pares de vértices do grafo. O grafo G_2 visto figura 2.2, por exemplo, é conexo. Ou seja, partindo-se de qualquer vértice de G_2 é possível encontrar pelo menos um caminho para os demais vértices.

A figura 2.4, apresentada a seguir, mostra um grafo não conexo. Os vértices v_5 e v_6 de G_4 estão “isolados” dos demais vértices, pois, nenhum caminho pode ser estabelecido entre eles e os outros vértices do grafo.

Graficamente, é fácil perceber que o grafo G_4 é composto por duas partes separadas. A primeira é formada pelos conjuntos $V_1' = \{ v_1, v_2, v_3, v_4 \}$ e $R_1' = \{ r_1, r_2, r_3, r_4, r_5 \}$; a segunda, por $V_2' = \{ v_5, v_6 \}$ e $R_2' = \{ r_6 \}$. Se analisadas separadamente, cada uma dessas partes de G_4 é um grafo conexo. Portanto, é correto afirmar que o grafo G_4 possui duas componentes conexas, onde cada componente é, na verdade, um subgrafo de G_4 .

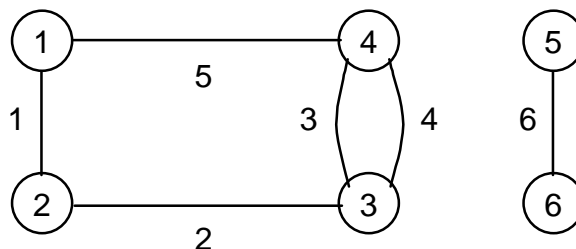


Figura 2.4. Grafo G_4 – Grafo não conexo.

O número de componentes conexas de um grafo, indicado por c , é chamado de conexidade. Para grafos conexos, a conexidade é sempre igual a 1.

2.2.2.7. Árvore e Co-Árvore

Árvore é definida como um grafo conexo e sem ciclos. O grafo G_5 , visto na figura 2.5, apresenta uma árvore de ordem 4. Neste tipo de grafo, o número de ramos é sempre igual ao número de vértices menos um, ou seja, $b = n - 1$.



Figura 2.5. Grafo G_5 – Árvore geradora de G_2 .

Um grafo G' é denominado árvore geradora de um grafo conexo G se:

- G' for uma árvore;
- G' contiver todos os vértices de G .

O grafo G_5 visto na figura 2.5, por exemplo, é uma árvore geradora do grafo G_2 , visto na figura 2.2.

O grafo G'' , formado a partir dos vértices de G e dos ramos de G não incluídos em G' , é chamado de co-árvore. A figura 2.6 mostra uma co-árvore do grafo G_2 , visto na figura 2.2.

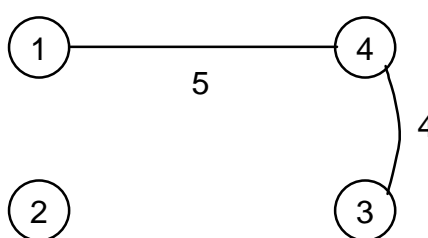


Figura 2.6. Grafo G_6 – Co-árvore de G_2 .

Os ramos da co-árvore são comumente chamados de cordas. As cordas são importantes para a obtenção dos ciclos de um grafo; pois, ao incluir uma corda da co-árvore na árvore geradora, encontra-se um ciclo do grafo original.

2.2.2.8. Floresta e Co-Floresta

Floresta é definida como um conjunto de árvores. Neste tipo de grafo, o

número de ramos é sempre igual ao número de vértices menos a sua conexidade, ou seja, $b = n - c$. A figura 2.7, apresentada a seguir, mostra uma floresta de conexidade igual a 2.

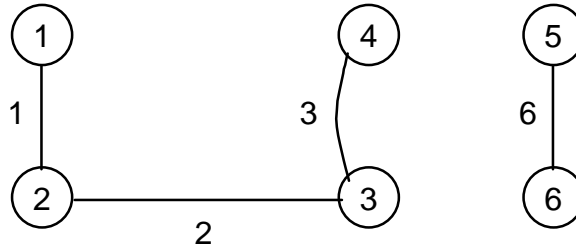


Figura 2.7. Grafo G_7 – Floresta geradora de G_4 .

O termo floresta geradora é utilizado para denominar a árvore geradora de um grafo não conexo. Quando o grafo é não conexo, cada uma das componentes do grafo possui uma árvore geradora resultando, portanto, em uma floresta. O grafo G_7 visto na figura 2.7 é um exemplo de floresta geradora para o grafo G_4 visto na figura 2.4. A co-floresta pode ser obtida a partir dos vértices do grafo e dos ramos não incluídos na floresta. A co-floresta possui a mesma finalidade da co-árvore: encontrar os ciclos do grafo. A figura 2.8 mostra uma co-floresta do grafo G_4 .

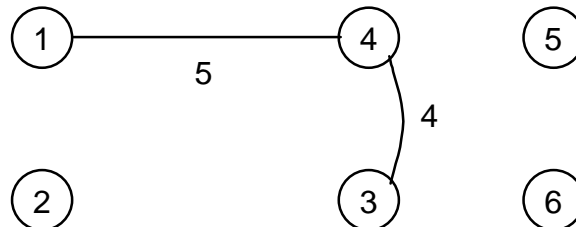


Figura 2.8. Grafo G_8 – Co-floresta de G_4 .

2.2.2.9. Grafo Direcionado e Valorado

O grafo direcionado, ou dígrafo, $G_D(V, R)$ é um tipo de grafo onde cada ramo $r_k = (v_i, v_j)$, com $r_k \in R$, v_i e $v_j \in V$, é associado com um par de vértices ordenados. O grafo G_D é dito valorado se existirem funções ou ganhos associados aos conjuntos V ou R .

Os vértices terminais v_i e v_j associados ao ramo r_k são chamados de vértices inicial e final do ramo, respectivamente.

Na forma gráfica, os vértices são representados por circunferências identificadas pelo índice do vértice. Os ramos são representados por segmentos de reta orientados, interligando o vértice inicial ao vértice final. Os ramos são identificados pelo seu número ou pelo ganho associado a ele.

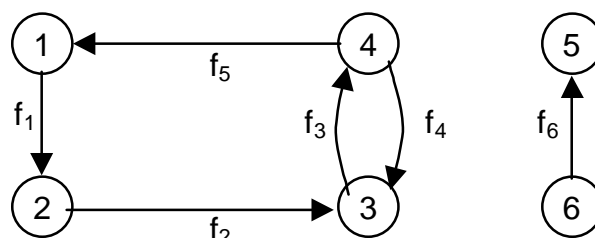


Figura 2.9. Grafo G_9 – Grafo direcionado e valorado.

2.2.2.10. Circuito ou Laço

O conceito de ciclo se aplica aos grafos direcionados mas deve satisfazer a orientação dos ramos. No grafo G_9 , visto na figura 2.9, o conjunto $R_1 = \{ r_1, r_2, r_3, r_5 \}$ forma um circuito, mas o conjunto $R_2 = \{ r_1, r_2, r_4, r_5 \}$ não forma. No escopo deste trabalho, os circuitos serão chamados de laços.

2.3. Modelagem de Sistemas Físicos

Esta seção apresenta tópicos relacionados à modelagem de sistemas físicos. Neste trabalho, o conceito de energia, [12], empregado na modelagem desses sistemas, permite estabelecer uma abordagem unificada no estudo de dispositivos de vários domínios da física, com o objetivo de formular um modelo matemático para o sistema físico com um todo. Conceitos relevantes à modelagem de sistemas sob esta visão são apresentados a seguir.

2.3.1. Variáveis Generalizadas

A abordagem unificada no estudo de sistemas de vários domínios da física pressupõe a definição de duas variáveis generalizadas denominadas de esforço e fluxo. O objetivo destas variáveis é identificar grandezas físicas equivalentes nos diversos domínios em estudo, sistematizando o processo de modelagem dos

sistemas. Dentre os sistemas estudados neste trabalho podemos citar: sistemas elétricos, mecânicos translacionais, mecânicos rotacionais e fluídos (ou hidráulicos).

A variável generalizada de esforço representa uma grandeza física mensurável entre dois pontos distintos de um sistema físico. Nos sistemas elétricos, a variável esforço é representada pela tensão elétrica que é aferida entre dois pontos quaisquer de um circuito. Velocidade linear, velocidade angular e pressão são as variáveis de esforço definidas para os sistemas mecânicos translacionais, rotacionais e fluidos, respectivamente.

A variável generalizada de fluxo representa uma grandeza mensurável através de um ponto do sistema. Nos sistemas elétricos, a variável fluxo é a corrente elétrica a qual flui através de um dispositivo. Nos sistemas mecânicos translacionais, rotacionais e fluídos, os fluxos são força, torque e vazão, respectivamente.

As variáveis de esforço são também conhecidas como variáveis “entre” por aferir uma grandeza entre dois pontos do sistema e as variáveis de fluxo são conhecidas como variáveis “através” por aferir grandezas que atravessam um ponto do sistema físico. A tabela 2.1 mostra as variáveis de esforço e fluxo nos domínios físicos em estudo.

Tabela 2.1. Variáveis de esforço e fluxo.

Variável	Mecânico Translacional	Mecânico Rotacional	Elétrico	Fluido
Esforço	Velocidade linear	Velocidade angular	Tensão	Pressão
Fluxo	Força	Torque	Corrente	Vazão

A definição de variáveis generalizadas e o conceito de energia são fundamentais para o estabelecimento de uma abordagem unificada na modelagem de sistemas físicos, possibilitando, por conseguinte, a generalização dos dispositivos físicos constituintes dos sistemas. Esta generalização facilita o estudo da dinâmica dos sistemas e a definição de modelos com dispositivos físicos de diferentes domínios. Um sistema composto por um motor DC alimentado por uma fonte de tensão é um exemplo deste fato, uma vez que envolve os sistemas elétrico

e mecânico.

2.3.2. Representação do Sistema

Sob a visão energética, os sistemas físicos são tratados como manipuladores de energia. Esta energia é processada de forma diferente por cada dispositivo do sistema de acordo com as características físicas intrínsecas a cada um. Nos sistemas elétricos, por exemplo, as fontes de tensão podem ser vistas como fontes de energia e os resistores, como dissipadores.

A figura 2.10 apresenta o modelo de um sistema físico, mostrando o mecanismo de transferência de energia entre uma fonte e o restante do sistema. A energia fornecida pela fonte é transmitida através uma conexão conhecida como “porta de energia”, na qual podemos identificar as variáveis generalizadas de esforço e fluxo.

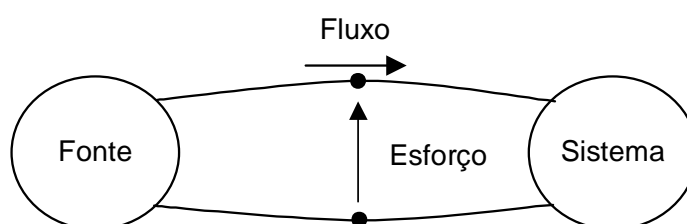


Figura 2.10. Representação do sistema físico.

O produto entre esforço e fluxo é a potência instantânea fornecida pela fonte de energia ao sistema através da porta. Por conseguinte, a energia transmitida ao sistema pode ser obtida pela equação 2.1:

$$E = \int_0^t e.f.dt \quad (2.1)$$

onde: E = energia transmitida pela fonte,

e = esforço entre os terminais da porta,

f = fluxo através da porta.

Esta equação é fundamental para a análise em questão porque ela representa a troca de energia que ocorre entre todos os dispositivos constituintes

de um sistema de qualquer domínio físico.

2.3.3. Generalização dos Elementos do Sistema

Os dispositivos físicos constituintes dos sistemas são classificados quanto ao número de portas e ao processamento da energia. Em relação ao número de portas, os dispositivos são classificados como elementos de uma ou duas portas de energia. Quanto ao processamento de energia, são classificados como fontes, armazenadores, dissipadores, conversores, acopladores e moduladores de energia.

Os elementos básicos de uma porta interagem com o restante do sistema através de uma única conexão de energia e são classificados em: fontes de energia: fontes de esforço e de fluxo; armazenadores de energia: armazenadores de esforço e fluxo; dissipadores e moduladores.

Os elementos básicos de duas portas, que interagem com o sistema através de duas conexões, podem ser classificados em: conversores de energia, dispositivos que manipulam a energia em um único domínio físico, e acopladores de energia, dispositivos que transformam a energia de um domínio físico para outro.

A tabela 2.2 apresenta exemplos de elementos de uma porta para os sistemas mecânicos translacionais, elétricos e fluidos.

Tabela 2.2. Elementos básicos de uma porta.

Domínio	Mec. Translacional	Elétrico	Fluido
Fonte de Esforço	Fonte de Velocidade	Fonte de Tensão	Bomba com Pressão Constante
Fonte de Fluxo	Fonte de Força	Fonte de Corrente	Bomba com Vazão Constante
Acumulador de Esforço	Mola	Indutor	Tubulação
Acumulador de Fluxo	Massa	Capacitor	Reservatório
Dissipador de Energia	Amortecedor	Resistor	Atrito Fluido

A tabela 2.3 apresenta exemplos de conversores de energia para os sistemas

mecânicos rotacionais, elétricos e fluidos.

Tabela 2.3. Conversores de energia.

Domínio	Mecânico Rotacional	Elétrico	Fluido
Conversor de Energia	Conjunto de Polias	Transformador Elétrico	Macaco Hidráulico

Dentre os dispositivos acopladores de energia, podemos citar: o virabrequim, que transforma um movimento linear em rotacional, ou seja, é acoplador mecânico e o motor DC, que transforma energia elétrica em movimento, ou seja, é um acoplador de energia entre os domínios elétrico e mecânico rotacional.

2.3.4. Relações Constitutivas

Outro passo importante na modelagem dos sistemas físicos é a definição das relações constitutivas dos elementos dos sistemas em termos das variáveis generalizadas de esforço e fluxo. Essas relações, obtidas a partir de leis da física ou de ensaios em laboratório, descrevem as características físicas dos dispositivos e são usadas na obtenção do modelo matemático dos sistemas.

2.3.4.1. Propriedades Constitutivas das Fontes de Energia

As fontes de energia em estudo neste trabalho são fontes ideais de esforço ou fluxo. As fontes ideais de esforço fornecem uma quantidade específica de esforço dada pela sua relação constitutiva. A figura 2.11a mostra o símbolo e a figura 2.11b, a relação constitutiva para este tipo de fonte. Na prática, esta relação pode ser uma função temporal, como em fontes de tensão alternada de sistemas elétricos, mas independente do fluxo aferido através da fonte.

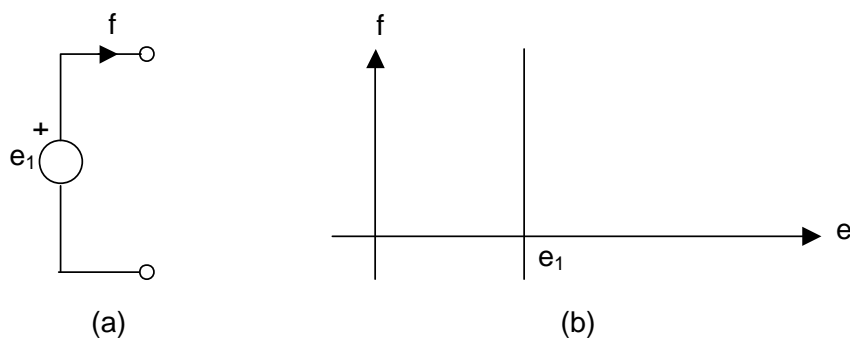


Figura 2.11. Fonte de esforço: (a) Símbolo, (b) Relação constitutiva.

De forma análoga, as fontes ideais de fluxo fornecem uma quantidade de fluxo determinada pela sua relação constitutiva. A figura 2.12 mostra o símbolo e a relação constitutiva para essas fontes. Esta relação pode ser também uma função temporal, como em fontes de corrente alternada de sistemas elétricos, mas obrigatoriamente independente do esforço aferido entre seus terminais.

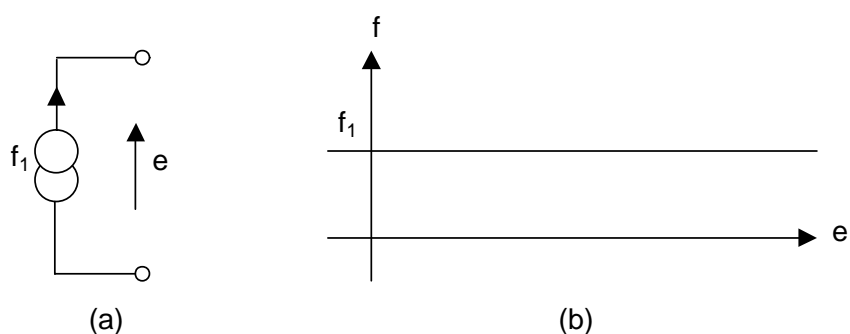


Figura 2.12. Fonte de fluxo: (a) Símbolo, (b) Relação constitutiva.

2.3.4.2. Propriedades Constitutivas dos Armazenadores de Energia

Os armazenadores de energia são classificados em armazenadores de esforço e de fluxo. Os armazenadores de esforço armazenam energia sob a forma de esforço acumulado (e_a). Esta variável é obtida pela integral temporal do esforço, conforme mostra a equação 2.2.

$$e_a = \int_0^t e \cdot dt \quad (2.2)$$

A relação constitutiva dos armazenadores de esforço é dada pela equação 2.3, mostrada abaixo, que relaciona o esforço acumulado com o fluxo no

armazenador de energia.

$$e_a = \varphi(f) \quad (2.3)$$

Nos sistemas elétricos, por exemplo, o indutor é o armazenador de esforço. Nestes dispositivos, o esforço (tensão elétrica) é armazenado sob a forma de fluxo magnético concatenado, o qual é função da variável generalizada fluxo (corrente elétrica), conforme mostram as equações 2.4 e 2.5, apresentadas a seguir:

$$\lambda = \int_0^t v \cdot dt \quad (2.4)$$

$$\lambda = L \cdot i \quad (2.5)$$

onde: λ = fluxo magnético concatenado – esforço acumulado (e_a),

v = tensão elétrica – variável generalizada de esforço (e),

i = corrente elétrica – variável generalizada de fluxo (f),

L = indutância.

O símbolo e a relação constitutiva para os armazenadores de esforço são mostrados na figura 2.13. A relação apresentada na figura refere-se ao caso linear onde o esforço armazenado é linearmente proporcional ao fluxo: $e_a = L \cdot f$.

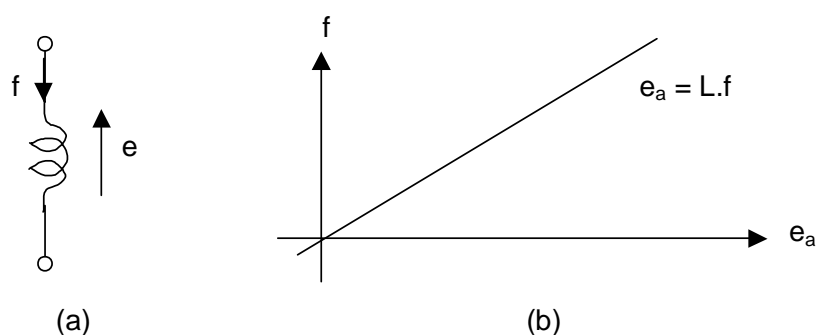


Figura 2.13. Armazenador de esforço: (a) Símbolo, (b) Relação constitutiva.

Os armazenadores de fluxo armazenam energia sob a forma de fluxo acumulado (f_a), o qual pode ser obtido pela integral do fluxo no tempo, conforme mostra a equação 2.6.

$$f_a = \int_0^t f \cdot dt \quad (2.6)$$

A equação 2.7 apresenta a relação constitutiva dos armazenadores de fluxo. Nestes dispositivos, o fluxo acumulado é função do esforço entre seus terminais.

$$f_a = \varphi(e) \quad (2.7)$$

Nos sistemas elétricos, por exemplo, o capacitor é o armazenador de fluxo. A carga elétrica é o fluxo armazenado, resultante da integral temporal da corrente (fluxo) e proporcional à tensão (esforço); conforme visto nas equações 2.8 e 2.9:

$$q = \int_0^t i \cdot dt \quad (2.8)$$

$$q = C \cdot v \quad (2.9)$$

onde: q = carga elétrica – fluxo acumulado (f_a),

v = tensão elétrica – variável generalizada de esforço (e),

i = corrente elétrica – variável generalizada de fluxo (f),

C = capacitância.

O símbolo e a relação constitutiva para os armazenadores de fluxo são mostrados na figura 2.14. A relação apresentada na figura refere-se ao caso linear onde o fluxo armazenado é linearmente proporcional ao esforço: $f_a = C \cdot e$.

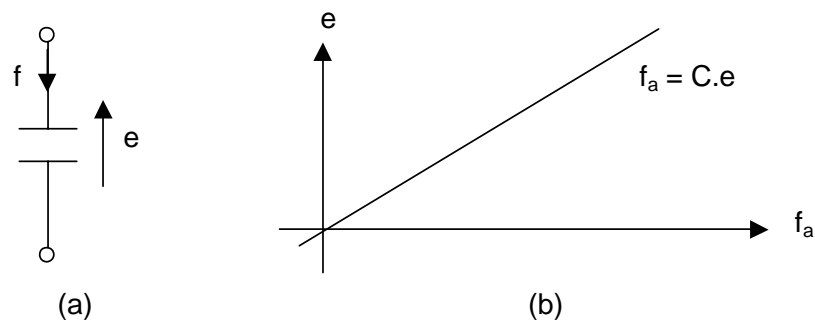


Figura 2.14. Armazenador de fluxo: (a) Símbolo, (b) Relação constitutiva.

2.3.4.3. Propriedades Constitutivas dos Dissipadores de Energia

Os dissipadores de energia são os dispositivos responsáveis pela conversão

da energia do sistema em um tipo de energia, normalmente térmico, irrecuperável pelo sistema. Nos sistemas elétricos, esses dispositivos são representados pelos resistores.

A relação constitutiva destes elementos é dada pela equação 2.10 que relaciona o esforço com o fluxo no elemento.

$$e = \varphi(f) \quad (2.10)$$

O símbolo e a relação constitutiva para os dissipadores são mostrados na figura 2.15, que apresenta essa relação para o caso linear, onde o esforço é proporcional ao fluxo: $e = R.f$.

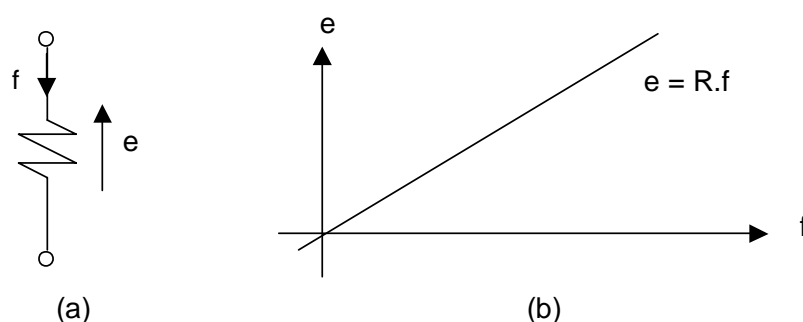


Figura 2.15. Dissipador de energia: (a) Símbolo, (b) Relação constitutiva.

2.3.4.4. Propriedades Constitutivas dos Moduladores de Uma Porta

Alguns elementos físicos apresentam uma relação constitutiva dependente de uma variável do sistema do qual fazem parte e são denominados de moduladores. Exemplos reais para tais dispositivos são amplificadores elétricos, motores e dinamômetros.

No presente trabalho, são considerados dois tipos específicos de moduladores: a fonte de esforço controlada, capaz de fornecer uma quantidade de esforço dependente de alguma variável generalizada do sistema, e a fonte de fluxo controlada, que, de forma análoga, fornece uma quantidade de fluxo dependente de outras variáveis do sistema físico.

A relação constitutiva para as fontes de esforço controladas é dada pela equação 2.11 que relaciona o esforço no elemento com a variável auxiliar. A

equação 2.12 mostra a relação constitutiva para as fontes de fluxo controladas.

$$e = \varphi(\rho) \quad (2.11)$$

$$f = \varphi(\rho) \quad (2.12)$$

Em ambos casos, ρ é uma variável generalizada de esforço ou fluxo em outro elemento do sistema.

2.3.4.5. Propriedades Constitutivas dos Elementos de Duas Portas

Os elementos de duas portas de energia, conversores ou acopladores, podem ser representados genericamente pelo diagrama mostrado na figura 2.16. Nestes elementos, a energia recebida pela porta 1 (e_1 , f_1) é transformada e transmitida à porta 2 (e_2 , f_2).

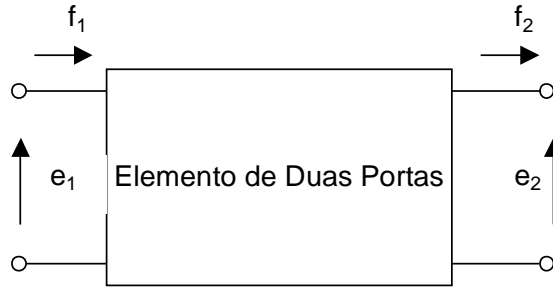


Figura 2.16. Elemento de duas portas.

Neste trabalho, são considerados os dispositivos de duas portas ideais e conservadores de energia, ou seja, que transmitem integralmente a potência recebida na porta 1 para a porta 2, sem geração, armazenamento ou dissipação de energia. Para o caso linear, a relação constitutiva destes dispositivos é dada pela equação 2.13.

$$\begin{bmatrix} e_2 \\ f_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} e_1 \\ f_1 \end{bmatrix} \quad (2.13)$$

A relação constitutiva para o transformador ideal é dada pela equação 2.14.

$$\begin{bmatrix} e_2 \\ f_2 \end{bmatrix} = \begin{bmatrix} N^{-1} & 0 \\ 0 & N \end{bmatrix} \begin{bmatrix} e_1 \\ f_1 \end{bmatrix} \quad (2.14)$$

A equação 2.15 apresenta a relação constitutiva para o girador ideal.

$$\begin{bmatrix} e_2 \\ f_2 \end{bmatrix} = \begin{bmatrix} 0 & G^{-1} \\ G & 0 \end{bmatrix} \begin{bmatrix} e_1 \\ f_1 \end{bmatrix} \quad (2.15)$$

Os elementos de duas portas são amplamente usados na modelagem de sistemas para representar dispositivos como transformadores elétricos e motores DC, por exemplo.

2.3.4.6. Exemplos de Propriedades Constitutivas

A tabela 2.4 apresenta exemplos de relações constitutivas para elementos de uma porta dos sistemas mecânicos translacionais, elétricos e fluidos.

Tabela 2.4. Relações constitutivas.

Domínio	Mecânico Translacional	Elétrico	Fluido
Esforço	Velocidade Linear (V)	Tensão (v)	Pressão (P)
Fluxo	Força (F)	Corrente (i)	Vazão (Q)
Esforço Acumulado	Deslocamento (x)	Fluxo Magnético (λ)	Momento Fluido (Γ_f)
Fluxo Acumulado	Momento (P_{mt})	Carga Elétrica (q)	Volume (V_f)
Acumulador de Esforço	Mola (K) $x = K^{-1}F$	Indutor (L) $\lambda = Li$	Tubulação (L_f) $\Gamma_f = L_f Q$
Acumulador de Fluxo	Massa (M) $P_{mt} = MV$	Capacitor (C) $q = Cv$	Reservatório (C_f) $V_f = C_f Q$
Dissipador de Energia	Amortecedor (B) $F = BV$	Resistor (R) $v = Ri$	Atrito Fluido (R_f) $P = R_f Q$

2.3.5. Grafo de Ligação

Uma vez estudados e definidos os elementos básicos dos sistemas e suas relações constitutivas, o passo seguinte na atividade de modelagem consiste na determinação do método utilizado para obtenção do modelo matemático do sistema

do físico.

Dentre os métodos de modelagem encontrados na literatura, [2, 3, 12], podemos citar: o método variacional, que realiza uma análise das alterações infinitesimais nas variáveis generalizadas do sistema; o método do *bond graph*, que consiste no estudo do sistema através de uma representação gráfica das interações energéticas entre seus elementos, e o Método de Rede (*Network Method*), que estuda as relações de inter-conectividade entre os dispositivos de um sistema representado como um dígrafo, denominado de grafo de ligação.

Dentre os três métodos citados, o Método de Rede apresentou-se como o de maior viabilidade para o desenvolvimento do ambiente computacional proposto neste trabalho. Isto porque a representação do sistema como grafo de ligação é a que mais se aproxima do desenho gráfico do sistema. Além disso, engenheiros de controle e eletricitas estão amplamente habituados a utilizar esta representação gráfica, visto que os grafos são a forma natural de representar circuitos elétricos em disciplinas relacionadas ao estudo de sistemas de controle.

O Método de Rede sistematiza a modelagem de um sistema físico como um grafo direcionado e valorado (dígrafo). Os vértices deste grafo representam os pontos do sistema com o mesmo valor para a variável generalizada de esforço. Nos sistemas elétricos, por exemplo, os vértices são os pontos do circuito com tensões elétricas iguais. Para sistemas mecânicos, os vértices do dígrafo são os pontos do sistema com velocidades iguais.

Os ramos do grafo de ligação representam os elementos do sistema. Elementos como fontes, acumuladores e dissipadores, que somente possuem uma porta de energia, são representados por um único ramo. Os elementos com duas portas, conversores e acopladores, são representados por dois ramos.

O ganho dos ramos é oriundo das relações constitutivas dos elementos. Para as fontes de energia, o ganho é o valor do esforço ou fluxo fornecido pela fonte. Para armazenadores e dissipadores, o ganho é a “impedância” generalizada, ou seja, a relação entre o esforço e o fluxo no elemento. Os elementos de duas portas normalmente apresentam ganho idêntico às relações de transformação ou giração.

Neste trabalho, os ganhos são expressos no domínio de Laplace, [3], objetivando o cálculo da função de transferência entre variáveis dos sistemas.

Uma padronização importante na representação do sistema como grafo de ligação consiste na definição da orientação dos ramos. A fim de estabelecer um referencial, os ramos são sempre orientados no sentido positivo do fluxo e decrescente do esforço, isto é, na direção positiva do fluxo de potência. Esta é a convenção normalmente utilizada na modelagem de sistemas elétricos. Ressalvas são feitas para as fontes de energia onde ocorre inversão no sentido da variável esforço, para fontes de fluxo, e fluxo, para fontes de esforço.

A figura 2.17 mostra a convenção utilizada para os elementos passivos do sistema: acumuladores de esforço e fluxo e dissipadores.

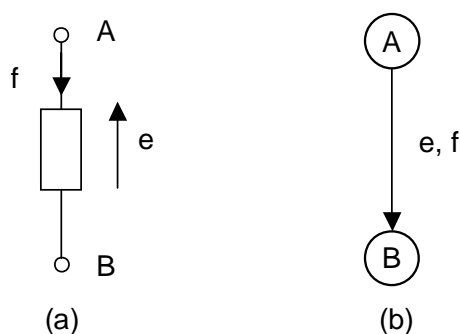


Figura 2.17. Convenção para elementos passivos: (a) Elemento, (b) Orientação do ramo.

A figura 2.18 mostra a convenção utilizada para as fontes de esforço. Nestes elementos, a variável fluxo é invertida em relação à orientação do ramo.

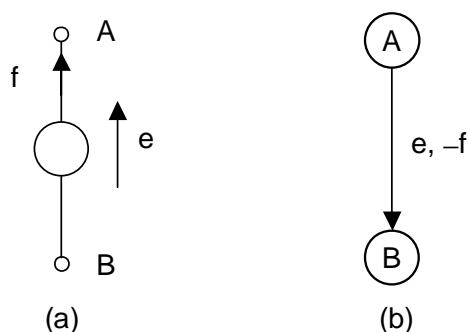


Figura 2.18. Convenção para fontes de esforço: (a) Elemento, (b) Orientação do ramo.

A convenção utilizada para as fontes de fluxo é vista na figura 2.19. Nestes elementos, a variável esforço é invertida em relação à orientação do ramo.

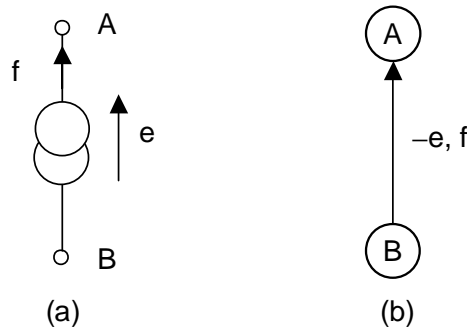


Figura 2.19. Convenção para fontes de fluxo: (a) Elemento, (b) Orientação do ramo.

Finalmente, a figura 2.20 mostra a convenção para os elementos de duas portas. Na segunda porta destes elementos, a variável fluxo é invertida em relação à orientação do ramo.

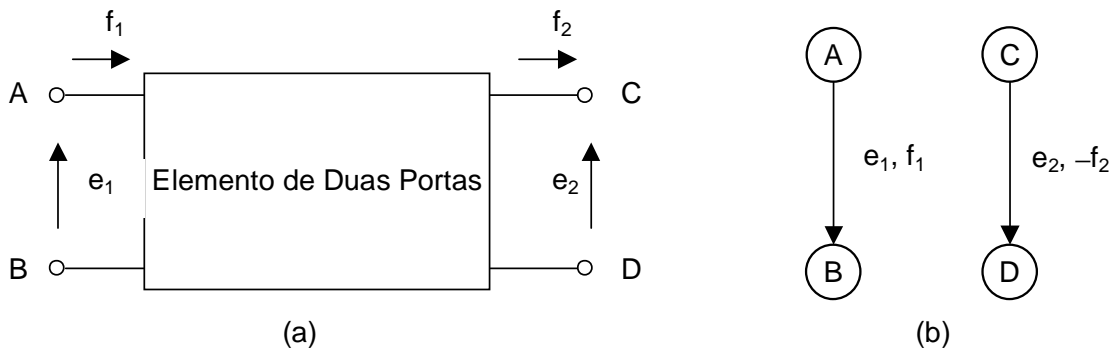


Figura 2.20. Convenção para elementos de duas portas: (a) Elemento, (b) Ramos.

Devido à inversão na orientação do fluxo na segunda porta, a relação constitutiva dos elementos de duas portas, dada pela equação 2.13, é reescrita conforme visto na equação 2.16.

$$\begin{bmatrix} e_2 \\ -f_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} e_1 \\ f_1 \end{bmatrix} \quad (2.16)$$

A figura 2.21 apresenta um circuito elétrico juntamente com seu grafo de ligação. Os pontos de tensões iguais, indicados pelos algarismos de 1 a 4 no circuito, originam os vértices do grafo. Cada componente passivo é representado por um ramo orientado no sentido decrescente do esforço. Esta orientação coincide com o sentido convencional da corrente elétrica. Os ganhos nestes ramos são as impedâncias dos elementos. De acordo com a convenção adotada, o sentido do fluxo na fonte de tensão é invertido em relação ao sentido convencional da corrente.

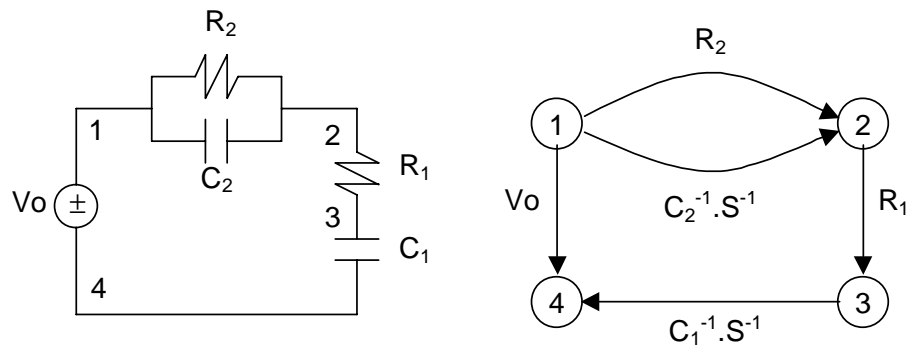


Figura 2.21. Grafo de ligação de um sistema elétrico.

A figura 2.22 mostra um sistema mecânico e seu grafo de ligação. A referência e as massas do sistema definem os pontos de velocidades iguais, ou seja, de esforços iguais. Tais pontos, indicados pelos algarismos de 1 a 3, originam os vértices do grafo, sendo o vértice v_3 o de maior esforço e o v_1 , o de menor.

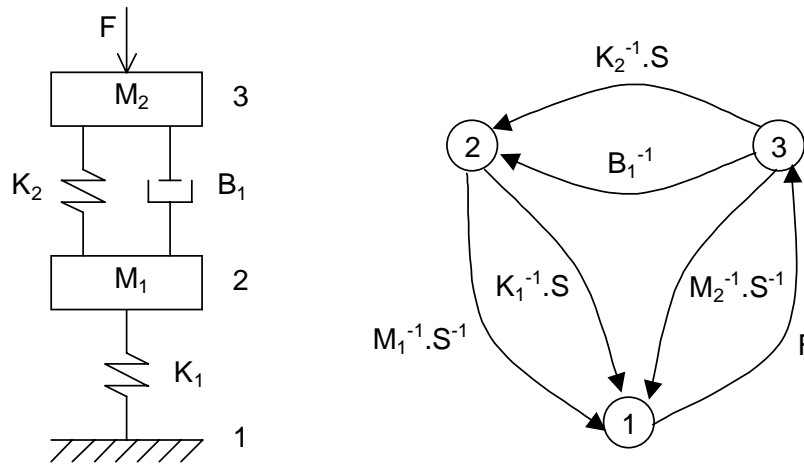


Figura 2.22. Grafo de ligação de um sistema mecânico.

Os elementos passivos dos sistemas mecânicos, isto é, massas, molas e amortecedores, originam ramos orientados no sentido decrescente da variável generalizada de esforço. No exemplo, a mola K_2 e o amortecedor B_1 estão submetidos à diferença de velocidade entre os pontos 3 e 2, sendo representados por ramos orientados do vértice v_3 ao vértice v_2 , ou seja, no sentido decrescente da velocidade. A mola K_1 , sujeita à velocidade entre os pontos 2 e 1, é modelada por um ramo com vértice inicial v_2 e final v_1 . Conforme mostra a figura 2.2, os ganhos destes ramos são definidos pela impedância generalizada de cada elemento no domínio de Laplace.

As massas e forças dos sistemas mecânicos estão sempre conectadas à referência do sistema. Isto porque a velocidade e a aceleração aferida nestes

elementos são tomadas em relação ao referencial estabelecido. Por isso, no exemplo, as massas M_1 e M_2 são representadas por ramos com vértice inicial v_2 e v_3 , respectivamente, e vértice final v_1 . A força aplicada à massa M_2 é modelada por uma fonte de fluxo. Conforme citado anteriormente, nestes elementos o sentido da variável esforço é invertido, implicando na orientação do ramo que representa esta fonte no sentido crescente do esforço. Portanto, o ramo da fonte possui como vértice inicial v_1 , referência do sistema, e como vértice final v_3 , esforço em M_2 .

2.3.6. Relações de Interconectividade

A partir do grafo de ligação, o passo seguinte na busca de um modelo matemático para o sistema é a aplicação das relações de interconectividade do grafo. Essas relações são expressas algebricamente por duas matrizes: a matriz de compatibilidade de esforço e a matriz de continuidade de fluxo, [12], resultantes de uma generalização das leis das malhas e dos nós de Kirchoff aplicadas às variáveis de esforço e fluxo. Essas matrizes expressam as interações que ocorrem entre os elementos dos sistemas e são de fundamental importância no processo de estudo da sua dinâmica.

Neste trabalho, as matrizes de compatibilidade de esforço e continuidade de fluxo foram utilizadas na implementação do algoritmo que obtém o diagrama de fluxo de sinal do sistema a partir do seu grafo de ligação. O Capítulo 6 apresenta mais detalhes sobre as matrizes supracitadas e mostra os passos do algoritmo proposto para a obtenção do DFS.

2.4. Diagramas de Fluxo de Sinal

Esta seção apresenta o conceito de diagrama de fluxo de sinal (DFS), destacando alguns tópicos relacionados ao desenvolvimento deste trabalho.

2.4.1. Conceito

O diagrama de fluxo de sinal é uma representação gráfica de um sistema equações algébricas lineares, podendo representar o modelo matemático de um

sistema físico, linear, contínuo ou discreto e invariante no tempo. Este diagrama é modelado por um dígrafo, com os vértices representando as variáveis do sistema e os ramos, as relações entre estas variáveis.

No contexto deste trabalho, os vértices do DFS são variáveis generalizadas de esforço e fluxo e os ramos podem representar tanto as relações constitutivas dos elementos do sistema quanto às relações de interconectividade entre eles. Desta forma, tais diagramas representam o modelo de sistemas físicos em qualquer um dos domínios em estudo. A representação dos sistemas através do DFS objetiva o cálculo da função de transferência do sistema com a aplicação da regra de Mason ao diagrama.

2.4.2. Equações do DFS

A figura 2.23 apresenta um exemplo de DFS com 6 variáveis. O conjunto de equações que o diagrama representa pode ser obtido com a aplicação das seguintes regras:

- Os vértices v_1, v_2, \dots, v_n representam as variáveis x_1, x_2, \dots, x_n do sistema,
- Os ganhos dos ramos representam coeficientes que relacionam as variáveis,
- Para todo vértice com ramo de entrada (convergente), obtém-se uma equação onde a variável deste vértice é igual ao somatório dos produtos entre o ganho de cada ramo convergente e a variável do vértice inicial deste ramo.

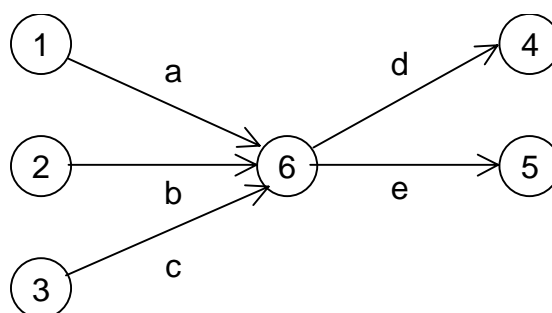


Figura 2.23. Exemplo de DFS.

A equação 2.17 mostra o sistema de equações representado por este diagrama de fluxo de sinal:

$$\begin{aligned}
 x_4 &= d.x_6 \\
 x_5 &= e.x_6 \\
 x_6 &= a.x_1 + b.x_2 + c.x_3
 \end{aligned}
 \tag{2.17}$$

2.4.3. Definições

Algumas definições referentes aos diagramas de fluxo de sinal e necessárias à implementação da regra de Mason serão apresentadas em seguida. Os exemplos apresentados em cada definição referem-se ao DFS mostrado na figura 2.24.

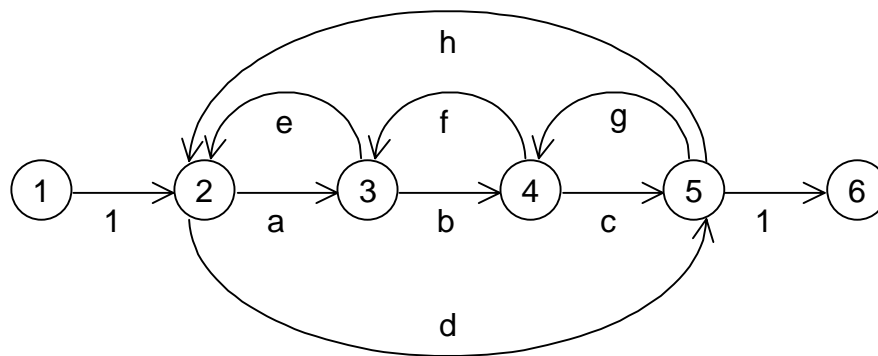


Figura 2.24. Exemplo de DFS.

Nó de entrada é definido como o vértice que não possui ramos convergentes. No presente contexto, corresponde a uma variável de esforço ou de fluxo referente a uma fonte de energia. O vértice v_1 , por exemplo, é um nó de entrada.

Nó de saída é o vértice que possui somente ramos convergentes. O vértice v_6 do DFS é um nó de saída.

Nó misto é o vértice que possui tanto ramos convergentes como divergentes. Os vértices v_2 , v_3 , v_4 e v_5 são exemplos de nós mistos. Vértices deste tipo representam, normalmente, as variáveis generalizadas de esforço e fluxo em cada elemento do sistema.

Transmitância é definida como o ganho entre dois nós. A transmitância entre os vértices v_1 e v_2 , por exemplo, é igual a 1.

Ganho de um caminho é o produto das transmitâncias dos ramos do caminho. O ganho do caminho formado pelos vértices v_2 , v_3 , v_4 e v_5 , por exemplo, é igual a “a.b.c”.

Ganho de um laço é o produto das transmitâncias dos ramos do laço. O ganho do laço formado pelos vértices v_2 , v_3 , e v_2 , por exemplo, é igual a “a.e”.

Dois laços de um DFS que não possuem nenhum vértice em comum são definidos como laços que não se tocam, laços disjuntos ou laços de segunda ordem. No DFS da figura 2.24, os laços $L_1 = \{ v_2, v_3, \text{ e } v_2 \}$ e $L_2 = \{ v_4, v_5, \text{ e } v_4 \}$, por exemplo, são laços de segunda ordem. Quando um conjunto formado por n laços é tal que nenhum dos laços toca outro laço do conjunto, ou seja, não possuem vértices em comum, esses laços são ditos laços de n -ésima ordem.

2.4.4. Álgebra do DFS

A obtenção da função de transferência do sistema a partir do DFS consiste no cálculo da transmitância entre o vértice do DFS que representa a saída do sistema físico e o vértice que representa a entrada ou excitação. Este cálculo pode ser realizado a partir de um conjunto básico de operações denominado de álgebra do diagrama de fluxo de sinal.

A álgebra do DFS consiste em um conjunto de transformações na estrutura do DFS que analisa a disposição dos ramos do grafo. Ramos em série ou em paralelo, por exemplo, podem ser substituídos por ramos equivalentes. Outros tipos de transformações envolvem normalmente a eliminação de nós mistos e de realimentações.

Para diagramas com estruturas complexas, entretanto, a utilização de transformações elementares é muito demorada e, normalmente, inviável. Por isso, a solução implementada neste trabalho consiste na utilização da regra de Mason para o cálculo da transmitância entre nós mistos ou de saída e nós de entrada. O algoritmo proposto por Mason para o cálculo de ganhos em diagramas de fluxo de sinal é especialmente útil na solução de diagramas complexos e não requer a aplicação de reduções ao diagrama. Além disso, o algoritmo é adequado para implementação computacional.

2.5. Regra de Mason

O cálculo da função de transferência de um sistema físico linear envolve a solução de um sistema de equações também lineares que descreve a dinâmica do sistema. A solução deste sistema de equações pode ser obtida de várias formas diferentes, utilizando, por exemplo, algoritmos para inversão de matrizes.

Um algoritmo proposto por Mason, em 1956, conhecido como fórmula do ganho de Mason ou regra de Mason, demonstrou-se eficiente na solução deste problema. A regra sistematiza o cálculo do ganho de um diagrama de fluxo de sinal, que pode representar a função de transferência de um sistema físico. Esta regra estuda combinações entre os diversos laços do diagrama.. Em geral, os ganhos dos laços e das combinações entre eles, que estão relacionados aos ganhos dos componentes do sistema, são termos da função de transferência.

2.5.1. Fórmula do Ganho de Mason

A fórmula do ganho de Mason é apresentada pela equação 2.18:

$$G_{DFS} = \frac{\sum_k G_k \Delta_k}{\Delta} \quad (2.18)$$

onde: G_{DFS} = ganho do DFS,

Δ = determinante do DFS,

G_k = ganho do k-ésimo caminho direto entre os nós de entrada e de saída,

Δ_k = cofator do k-ésimo caminho direto.

Cada termo da fórmula e os passos realizados para o cálculo do ganho serão apresentados a seguir, considerando o DFS apresentado na figura 2.24.

O primeiro passo é a obtenção de todos os laços do DFS. A tabela 2.5 mostra os laços simples (primeira ordem) e os laços que não se tocam dois a dois (segunda ordem). O DFS, neste exemplo, não apresenta laços de ordem superior a dois.

Tabela 2.5. Laços do DFS.

Laço	Vértices / Laços	Ganho	Ordem
L1	{ v ₂ , v ₃ , v ₄ , v ₅ , v ₂ }	a.b.c.h	1
L2	{ v ₂ , v ₃ , v ₂ }	a.e	1
L3	{ v ₂ , v ₅ , v ₂ }	d.h	1
L4	{ v ₂ , v ₅ , v ₄ , v ₃ , v ₂ }	d.g.f.e	1
L5	{ v ₃ , v ₄ , v ₃ }	b.f	1
L6	{ v ₄ , v ₅ , v ₄ }	c.g	1
L7	{ L2, L6 }	a.e.c.g	2
L8	{ L3, L5 }	d.h.b.f	2

O passo seguinte é o cálculo do determinante Δ do DFS, que é calculado pela equação 2.19:

$$\Delta = 1 - \sum (\text{Ganho dos Laços de 1ª Ordem}) + \sum (\text{Ganho dos Laços de 2ª Ordem}) - \sum (\text{Ganho dos Laços de 3ª Ordem}) + \dots \quad (2.19)$$

Aplicando os dados da tabela 2.5 à equação 2.19, obtemos o seguinte valor para o determinante Δ :

$$\Delta = 1 - (a.b.c.h + a.e + d.h + d.g.f.e + b.f + c.g) + (a.e.c.g + d.h.b.f) \quad (2.20)$$

O terceiro passo é o cálculo dos ganhos G_k . Conforme pode ser observado na figura 2.24, o DFS possui dois caminhos diretos entre os nós de entrada e saída. O primeiro caminho apresenta o seguinte conjunto de vértices: { v₁, v₂, v₃, v₄, v₅, v₆ }, cujo ganho é dado pela equação 2.21.

$$G_1 = a.b.c \quad (2.21)$$

O segundo caminho atravessa os vértices { v₁, v₂, v₅, v₆ } e possui um ganho dado pela equação 2.22.

$$G_2 = d \quad (2.22)$$

Finalmente, os cofatores Δ_k dos caminhos são calculados. Para obter o valor

de Δ_k , elimina-se de Δ os laços que tocam o k-ésimo caminho direto. Os valores de Δ_1 e Δ_2 são dados pelas equações 2.23 e 2.24.

$$\Delta_1 = 1 \quad (2.23)$$

$$\Delta_2 = 1 - b.f \quad (2.24)$$

Substituindo as equações 2.20 a 2.24 em 2.18, obtemos o seguinte valor para o ganho do DFS:

$$G_{\text{DFS}} = \frac{a.b.c + d - b.d.f}{1 - (a.b.c.h + a.e + d.h + d.g.f.e + b.f + c.g) + (a.e.c.g + d.h.b.f)} \quad (2.25)$$

Conforme citado, o valor do ganho do DFS representa a função de transferência entre variáveis de um sistema físico. As variáveis de saída e entrada da função de transferência são representadas, respectivamente, pelos nós de saída e entrada do diagrama. Desta forma, a equação 2.25 é a função de transferência entre as variáveis x_6 e x_1 do sistema representado pelo DFS da figura 2.24.

2.5.2. DFS Fechado

O cálculo do ganho de um diagrama de fluxo de sinal pode ser facilitado com a introdução do conceito de DFS fechado em que o determinante e os cofatores são obtidos da mesma forma. Este DFS é obtido a partir da inclusão de um ramo no diagrama original, interligando o nó de saída ao nó de entrada. A figura 2.25 mostra o DFS fechado do diagrama apresentado na figura 2.24.

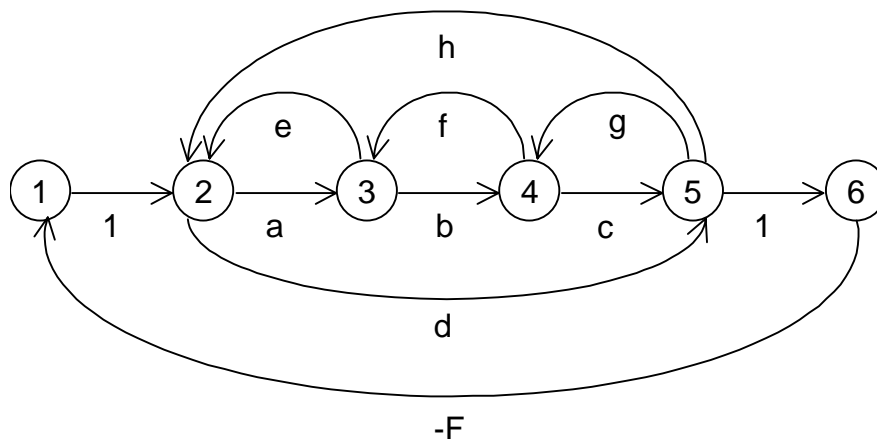


Figura 2.25. Exemplo de DFS fechado.

Para o cálculo do ganho do DFS original, o ramo introduzido no fechamento do diagrama deve possuir um valor simbólico diferente de todos os outros ganhos do grafo. Esta restrição deve-se ao fato deste ganho necessitar ser manipulado simbolicamente, de forma que seja possível separar os termos que possuem o novo símbolo daqueles que não o possuem. No diagrama da figura 2.25, por exemplo, o ramo inserido interligando a saída à entrada do DFS possui ganho igual a “-F”.

O determinante Δ_c do DFS fechado pode ser expresso em função dos ganhos diretos, cofatores e determinante do DFS original e, ainda, do ganho “F” referente ao ramo utilizado no fechamento do diagrama. Esta relação é apresentada pela equação 2.26, [15].

$$\Delta_c = \Delta + F \cdot \left(\sum_k G_k \Delta_k \right) \quad (2.26)$$

Comparando as equações 2.18 e 2.26 é fácil perceber que o ganho do DFS original pode ser obtido a partir do determinante do DFS fechado, bastando-se para isto isolar os termos de Δ_c que possuem o símbolo “F” introduzido ao diagrama. Os termos que não possuem “F” compõem o denominador da equação 2.18, ou seja, o determinante do diagrama original. Os termos que o possuem, compõem o numerador desta equação.

A tabela 2.6, apresentada a seguir, mostra os laços que foram introduzidos no diagrama com a inclusão do ramo de ganho “-F”.

Tabela 2.6. Laços introduzidos no DFS.

Laço	Vértices / Laços	Ganho	Ordem
L9	{ v ₁ , v ₂ , v ₃ , v ₄ , v ₅ , v ₆ , v ₁ }	- a.b.c.F	1
L10	{ v ₁ , v ₂ , v ₅ , v ₆ , v ₁ }	- d.F	1
L11	{ L5, L10 }	- b.f.d.F	2

Aplicando os dados das tabela 2.5 e 2.6 à equação 2.19, obtemos o seguinte valor para o determinante Δ_c do diagrama fechado:

$$\begin{aligned} \Delta_c = & 1 - (a.b.c.h + a.e + d.h + d.g.f.e + b.f + c.g - a.b.c.F - d.F) + \\ & + (a.e.c.g + d.h.b.f - b.f.d.F) \end{aligned} \quad (2.27)$$

Isolando os termos em “F” e substituindo o valor de Δ dado pela equação 2.20, podemos escrever a equação 2.27 na seguinte forma:

$$\Delta_c = \Delta + F(a.b.c + d - b.d.f) \quad (2.28)$$

onde Δ é o determinante do diagrama original e os termos multiplicados por “F” são o somatório dos produtos entre os ganhos diretos e os cofatores, referentes também a este DFS, conforme mostra a equação 2.25.

A utilização da técnica do DFS fechado diminui o esforço computacional do cálculo do ganho do diagrama porque elimina a necessidade de implementação de algoritmos para cálculo dos cofatores e determinação dos caminhos diretos entre nós de entrada e saída do DFS. Ou seja, a função de transferência do sistema físico, representado pelo DFS, é obtida com o cálculo e a manipulação simbólica do determinante Δ_c do diagrama fechado.

2.6. Funções de Sistema na Forma Simbólica

Esta seção apresenta o conceito de funções de sistema na forma simbólica, destacando alguns tópicos relacionados ao desenvolvimento deste trabalho.

2.6.1. Funções de Variáveis Generalizadas

As funções de transferência de sistemas lineares podem ser definidas como sendo a razão entre a transformada de Laplace da resposta e da entrada do sistema, com condições iniciais nulas. No contexto deste trabalho, tais funções podem ser classificadas quanto aos tipos de variáveis envolvidas nesta razão. A figura 2.26 mostra o esquema de um sistema físico, destacando-se dois elementos: uma fonte de energia e um dissipador. As variáveis generalizadas de esforço e fluxo (e_1 , e_2 , f_1 e f_2) serão as variáveis envolvidas no cálculo da função de transferência do sistema.

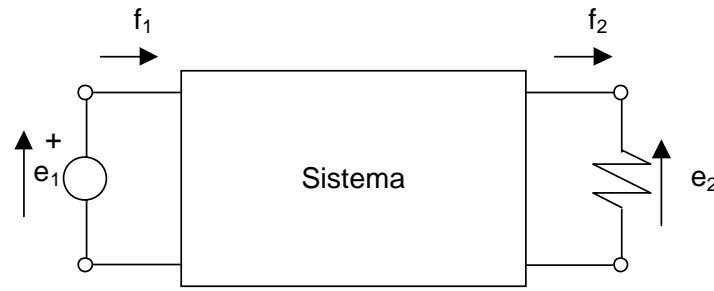


Figura 2.26. Exemplo de sistema físico.

Considerando $E_1(S)$, $E_2(S)$, $F_1(S)$ e $F_2(S)$ como a transformada de Laplace das variáveis generalizadas e_1 , e_2 , f_1 e f_2 , respectivamente, quatro tipos de funções de sistema poder ser obtidas:

$$\frac{E_2(S)}{F_1(S)} = \text{Impedância de Transferência} \quad (2.29)$$

$$\frac{F_2(S)}{E_1(S)} = \text{Admitância de Transferência} \quad (2.30)$$

$$\frac{E_2(S)}{E_1(S)} = \text{Ganho de Esforço} \quad (2.31)$$

$$\frac{F_2(S)}{F_1(S)} = \text{Ganho de Fluxo} \quad (2.32)$$

2.6.2. Funções Simbólicas

As funções de sistema podem também ser classificadas quanto ao número de parâmetros simbólicos expressos na função. As funções são denominadas de totalmente simbólicas quando todos os parâmetros do sistema são representados por símbolos, como mostra a equação 2.33.

$$\frac{E_2(S)}{E_1(S)} = \frac{1}{C.L.S^2 + C.R.S + 1} \quad (2.33)$$

As funções são parcialmente simbólicas quando apenas alguns parâmetros do sistema são expressos na forma de símbolos como na equação 2.34.

$$\frac{E_2(S)}{E_1(S)} = \frac{1}{10^{-6}.L.S^2 + 10^{-2}.S + 1} \quad (2.34)$$

Por fim, as funções podem ser não simbólicas, ou numéricas, quando nenhum parâmetro do sistema é simbólico. Nestes casos, são funções racionais em S com coeficientes numéricos, como na equação 2.35.

$$\frac{E_2(S)}{E_1(S)} = \frac{1}{2 \cdot 10^{-9} \cdot S^2 + 10^{-2} \cdot S + 1} \quad (2.35)$$

A grande maioria dos programas computacionais utilizados na modelagem e simulação de sistemas físicos utiliza funções numéricas na representação dos sistemas. Em tais programas, os parâmetros dos dispositivos do sistema têm de ser especificados por números reais ou complexos, implicando em limitações na análise de seu desempenho. O tratamento simbólico da função de transferência do sistema permite que uma melhor análise do sistema seja efetuada. Algumas vantagens desta manipulação são apresentadas a seguir.

2.6.3. Aplicações das Funções Simbólicas

2.6.3.1. Introspecção

A introspecção consiste no estudo do comportamento do sistema através de uma análise macroscópica da sua função de transferência. Considere, por exemplo, o sistema elétrico apresentado pela figura 2.27.

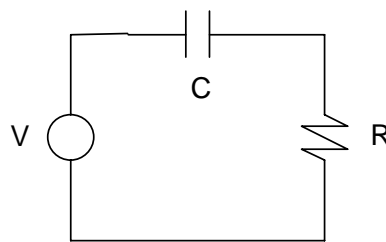


Figura 2.27. Introspecção de um sistema elétrico.

A função de transferência entre a tensão no resistor e a tensão da fonte é dada pela equação 2.36:

$$\frac{V_R(S)}{V(S)} = \frac{R}{R + \frac{1}{C \cdot S}} \quad (2.36)$$

Analisando o sistema no domínio da frequência, onde $S = j\omega$, podemos

concluir que se a frequência da fonte tende a infinito, a tensão no resistor tende a tensão da fonte. E, se a frequência se aproxima de zero, a tensão no resistor tende a zero. Considerando uma frequência fixa, podemos observar que a capacitância pode ser utilizada para ajustar o nível de tensão no resistor. A análise destas situações não poderia ser realizada tão facilmente se os parâmetros R e C do sistema estivessem na forma numérica o que dificultaria concluir que o sistema apresentado é um filtro elétrico conhecido como “passa alta” e que a frequência de corte, relacionada com a tensão no resistor, pode ser regulada com alteração da capacitância do circuito.

2.6.3.2. Melhoria na Precisão de Cálculos

Outro aspecto importante da representação simbólica diz respeito aos erros numéricos. Como a representação de grandezas reais no computador digital está sujeita a erros de arredondamento, devido ao tamanho finito da palavra da máquina, a manipulação numérica dos parâmetros dos sistemas está também submetida a estes erros. É importante lembrar, ainda, que quanto maior a quantidade de operações matemáticas realizadas com parâmetros numéricos, maior será o erro obtido no final das operações.

Considere, por exemplo, a avaliação de um determinado valor dado simbolicamente pela equação 2.37:

$$D = p + q - p - q \quad (2.37)$$

Se os valores de p e q são manipulados simbolicamente, o resultado da equação é exatamente o esperado, ou seja, zero.

Se atribuirmos valores numéricos para os símbolos p e q, o valor da operação pode não ser o esperado. Considerando $p=1,33333$ e $q=0,0133333$, uma precisão de máquina de 6 dígitos significativos e a seqüência de cálculo da esquerda para a direita na equação 2.37, obtemos o seguinte valor para D:

$$D = -0,0000033 \quad (2.38)$$

Então, é fácil concluir que o erro de aproximação pode ser significativo

dependendo da função do sistema e da diferença de grandeza entre os valores dos parâmetros numéricos.

2.6.3.3. Análise de Sensibilidade

Outro benefício da manipulação simbólica trata da análise de sensibilidade do sistema. Esta análise mostra quanto a variação dos parâmetros do sistema pode influenciar na sua resposta.

Considerando T_p como uma função de interesse e $\{ p_1, p_2, \dots p_n \}$, o conjunto de parâmetros do sistema, a sensibilidade normalizada de T_p , com respeito a um parâmetro p_i , é definida pela equação 2.39, [15].

$$T_{pi} = \frac{\partial T_p}{\partial p_i} \cdot \frac{p_i}{T_p} \quad (2.39)$$

Esta sensibilidade é importante na especificação da tolerância de componentes, que deve ser tanto menor quanto maior for a sensibilidade do sistema ao mesmo. Além disso, a análise de sensibilidade é importante na previsão de funcionamento de sistema em ambientes variáveis e no projeto automatizado de sistemas. Tal análise é realizada de forma mais conveniente com a função de sistema na forma simbólica. O software proposto encontra a sensibilidade normalizada da função de transferência com respeito a qualquer um dos parâmetros do sistema.

2.6.3.4. Sensibilidade em Larga Escala

A sensibilidade em larga escala consiste no cálculo da função do sistema para diferentes valores de um parâmetro; como, por exemplo, o cálculo do ganho de tensão de um amplificador para diferentes valores da resistência de carga R_L . Um método de força bruta consiste em analisar o amplificador para os diversos valores requeridos de R_L . Neste caso, o conjunto de equações do sistema é resolvido para cada um dos valores da carga, resultando numa repetição desnecessária de trabalho. Entretanto, se a função de transferência entre o ganho do amplificador e a carga é obtida na forma simbólica, a sensibilidade em larga escala é realizada pela

simples substituição do símbolo R_L pelos diferentes valores da carga em estudo. Esta operação é muito mais racional porque o conjunto de equações do sistema é resolvido uma única vez.

3. Modelagem de Software

3.1. Introdução

Este capítulo apresenta os tópicos mais importantes relacionados ao trabalho na área de modelagem e desenvolvimento de softwares. São apresentados os conceitos de básicos do Paradigma de Orientação a Objetos e da Linguagem de Modelagem Unificada (UML), [24, 25].

3.2. Paradigmas de Programação

As atividades de modelagem e desenvolvimento de softwares estão relacionadas a uma área da ciência bastante moderna: a Ciência da Computação. Os primeiros compiladores utilizados para desempenhar estas tarefas foram escritos a partir da década de 1950 e as pessoas que os escreviam tinham de “ajustá-los” à medida que trabalhavam com ele, devido às constantes falhas.

Com o passar do tempo e o aprimoramento das pesquisas, algumas metodologias e estilos são reconhecidos como sendo melhores que outras, de forma que, a ciência da Engenharia de Software continua seu avanço.

3.2.1. Programação Estruturada

Uma das primeiras grandes revoluções no desenvolvimento de softwares foi o advento da Programação Estruturada. Neste paradigma de programação, freqüentemente referido como *design top-down*, o processo utilizado na solução de um grande problema consiste na subdivisão sucessiva deste problema em problemas menores, conforme mostra o diagrama da figura 3.1. Desta forma, atinge-se o ponto onde os subproblemas são pequenos o suficiente para serem

implementados em uma rotina. A subdivisão sucessiva do problema a ser resolvido é conhecida como decomposição. Em síntese, o programador decompõe o problema em problemas menores que são mais facilmente resolvidos.

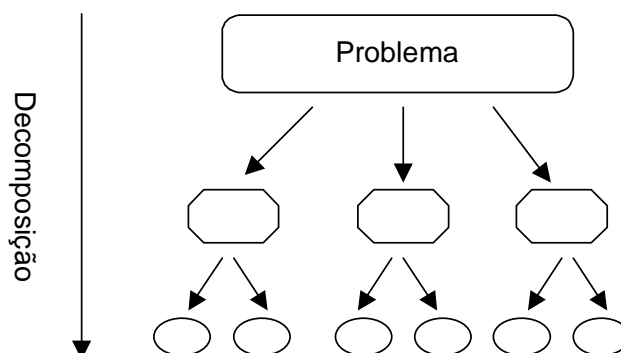


Figura 3.1. Decomposição de um problema.

A Programação Estruturada apresentou grande melhoria sobre os paradigmas de programação existentes, estabelecendo uma seqüência lógica ao fluxo de execução de um programa. Entretanto, este paradigma apresenta limitações. Um dos principais problemas com a Programação Estruturada é que, embora possibilite realizar um grande nível de decomposição, não oferece muita recomposição. Ou seja, apesar de facilitar a criação de um programa quebrando-o diversas vezes, dificulta a reutilização das rotinas desenvolvidas em outros programas. Como foram criadas dentro de uma hierarquia, as rotinas em geral são dependentes do problema no qual foram aplicadas.

3.2.2. Programação Modular

A Programação Modular, freqüentemente chamada de *design bottom-up*, surgiu com o intuito de resolver as deficiências da Programação Estruturada. Ao invés de iniciar com o problema principal e decompô-lo ao nível de rotinas, a Programação Modular encoraja a criação de pequenos módulos de software independentes e componíveis. Desta forma, a solução de um determinado problema pode ser resolvida juntando-se os pequenos módulos de software, de forma a obter pedaços cada vez maiores até que o problema seja resolvido, conforme mostra o diagrama da figura 3.2.

A essência da Programação Modular consiste no fato de se resolver um

problema da forma mais genérica possível, permitindo que a solução dada a um problema seja mais facilmente empregada em outros problemas, possibilitando com isso um maior grau de reutilização do código escrito.

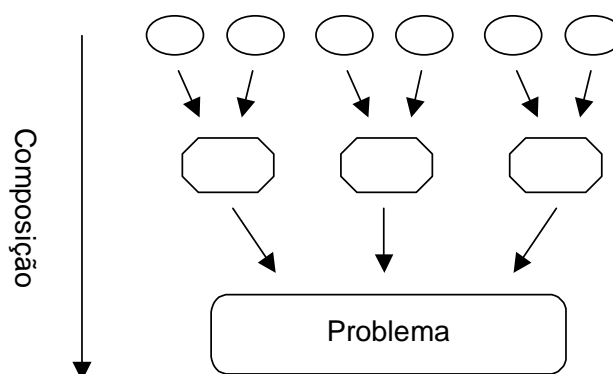


Figura 3.2. Composição de um problema.

3.2.3. Programação Orientada ao Objeto

O paradigma da Programação Orientada ao Objeto baseia-se nas idéias de Programação Modular e cria mecanismos para sistematizar o desenvolvimento dos pequenos módulos de software independentes e componíveis. Os principais conceitos atrelados a este paradigma são apresentados em seguida.

3.2.3.1. Classificação, Abstração e Instanciação

Classificar e abstrair são atividades bastante relacionadas ao pensamento humano. Quando crianças, aprendemos conceitos simples como pessoa, carro e casa, por exemplo, e, ao fazermos isso, definimos classes, ou seja, grupos de objetos que possuem características e comportamentos semelhantes. A partir daí, classificamos como sendo uma casa qualquer construção onde pessoas entram, e qualquer peça de metal com quatro rodas recebe a denominação de carro. Aos poucos, abstraímos a idéia de que o termo “carro” se refere a muitos objetos, apesar dos diferentes formatos e cores. Cada “objeto-carro”, entretanto, apresenta características semelhantes, como rodas, portas e direção, e realiza as mesmas tarefas, como transportar pessoas de um lugar para outro, [25].

No momento em que compreendemos esse conceito, percebemos que carro está associado a um grupo de objetos com características semelhantes, abstraindo

assim o conceito de classe. A partir de então, sempre que nos deparmos com um objeto com as características de um carro, concluímos que este é exemplo do grupo carro, ou seja, uma instância da classe “Carro”. Quando instanciamos objetos de uma classe, criamos um novo item do conjunto representado por ela, com as mesmas características e comportamentos dos outros objetos da classe.

É importante perceber, entretanto, que objetos de uma classe não são exatamente iguais, podendo apresentar “valores” diferentes para uma mesma característica. Um carro azul e outro vermelho, por exemplo, apresentam diferentes valores para a característica “cor”.

3.2.3.2. Classes e Objetos

O termo classe, conforme dito, é usado para representar uma categoria e os objetos são os exemplos desta categoria, [25]. Graficamente, as classes são representadas por retângulos, podendo possuir até três divisões. A primeira divisão apresenta o nome que identifica a classe, a segunda lista seus atributos e a terceira, seus métodos. Uma representação para a classe Carro é vista na figura 3.3.

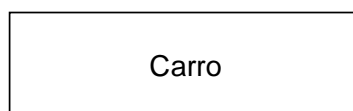


Figura 3.3. Representação gráfica de uma classe.

3.2.3.3. Atributos ou Propriedades

Os atributos ou propriedades de uma classe representam suas características, ou seja, as peculiaridades que costumam variar de objeto para objeto, [25].

A classe Carro pode, por exemplo, possuir os atributos de cor e número de portas, os quais podem assumir valores diferentes em cada objeto da classe. Desta forma, é perfeitamente possível termos um carro azul de duas portas e outro branco de quatro portas. É importante perceber que todas as instâncias de uma classe possuem exatamente os mesmos atributos, que são os atributos definidos na

classe. Um diagrama da classe Carro, com os atributos cor e número de portas, pode ser visto na figura 3.4.

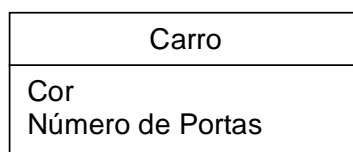


Figura 3.4. Exemplo de classe com atributos.

3.2.3.4. Métodos ou Comportamentos

Métodos ou comportamentos representam as atividades que um objeto pode executar. Fazendo um paralelo com a Programação Estruturada, podemos visualizar os métodos como rotinas especializadas para executar tarefas em um tipo específico de dado. Ou seja, os métodos representam conjuntos de instruções que são executadas quando eles são invocados. Os dados acessíveis pelos métodos de um objeto restringem-se aos atributos do próprio objeto.

A figura 3.5 mostra um diagrama da classe Carro com a inclusão do método TransportarPessoas. Desta forma, todas as instâncias dessa classe possuem agora a capacidade realizar esta tarefa.

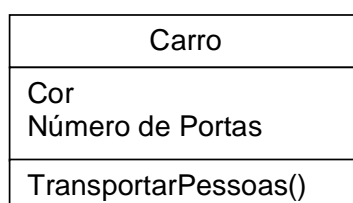


Figura 3.5. Exemplo de classe com atributos e métodos.

3.2.3.5. Visibilidade

A visibilidade é utilizada para indicar o nível de acessibilidade de atributos e métodos, [25]. Existem basicamente três tipos de visibilidade: pública, protegida e privada. Quando a visibilidade é pública, representada pelo símbolo “+”, o atributo ou método de um objeto pode ser acessado por outro objeto. Por sua vez, os atributos ou métodos protegidos de um objeto, simbolizados pelo caractere “#”, podem ser acessados somente pelo próprio objeto ou por descendentes deste.

Finalmente, os atributos ou métodos privados de um objeto, representados pelo símbolo “–”, são acessíveis somente pelo próprio objeto.

A figura 3.6 mostra o diagrama da classe CDPlayer. Nesta classe, são definidos os métodos Play e LigarMotor. Como o método Play é público, o usuário externo de um objeto CDPlayer, possivelmente um objeto da classe SerHumano, pode invocar o método Play para iniciar a execução de um disco. O método LigarMotor, entretanto, só pode ser invocado pelo próprio objeto CDPlayer, o que deverá ser feito quando a execução do disco for iniciada. Neste caso, é impossível um objeto SerHumano ligar diretamente o motor do CDPlayer porque ele simplesmente não consegue “ver” que o CDPlayer pode executar esta tarefa. Com esse mecanismo, a visibilidade controla os atributos e operações que um objeto disponibiliza aos demais objetos de um sistema. A visibilidade protegida está intrinsecamente relacionada à herança e será exemplificada em seguida.

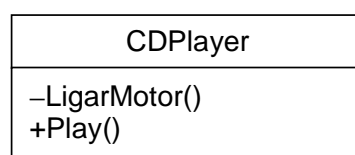


Figura 3.6. Exemplo de visibilidade.

3.2.3.6. Herança e Polimorfismo

A herança é um dos conceitos mais importantes do paradigma de Orientação a Objetos. Ela refere-se à habilidade de uma classe herdar as características de uma outra classe, permitindo o aproveitamento de todos seus atributos e métodos. A classe que herda as características de uma outra recebe normalmente as seguintes denominações: subclasse, classe-filha ou classe descendente. A outra classe, a que “cede” as características, é conhecida por superclasse, classe-mãe ou classe ancestral.

A utilização do conceito de herança na modelagem de sistemas permite otimizar o tempo de desenvolvimento, diminuir o número total de linhas de código e facilitar a realização de futuras manutenções. Toda esta funcionalidade está relacionada ao fato de um determinado comportamento do sistema, modelado em

uma única classe, poder ser “copiado” para outras classes do sistema sem reescrita de código. Se o comportamento apresenta alguma deficiência, é fácil identificar e solucionar o problema visto que existe uma única implementação dele.

O diagrama da figura 3.7 mostra em exemplo de herança. A classe ContaComum modela o funcionamento básico de uma conta bancária. Um objeto desta classe armazena os dados de número da conta, código do cliente e saldo. Ele também executa as seguintes operações: abertura e encerramento da conta, saque, depósito e verificação do saldo.

As classes ContaEspecial e ContaPoupança são descendentes da classe ContaComum. Isto significa que todos os atributos e operações definidos na classe ancestral são herdados por elas. Desta forma, contas especiais e poupanças possuem também os atributos de número da conta, código do cliente e saldo e realizam as todas as operações associadas a uma conta comum. Na definição da classe ContaEspecial, entretanto, foi incluído o atributo Limite, associando ao valor extra de saque que esta conta suporta. De forma análoga, a conta poupança define o atributo DiaRendimento, associado ao dia de crédito do rendimento nesta conta.

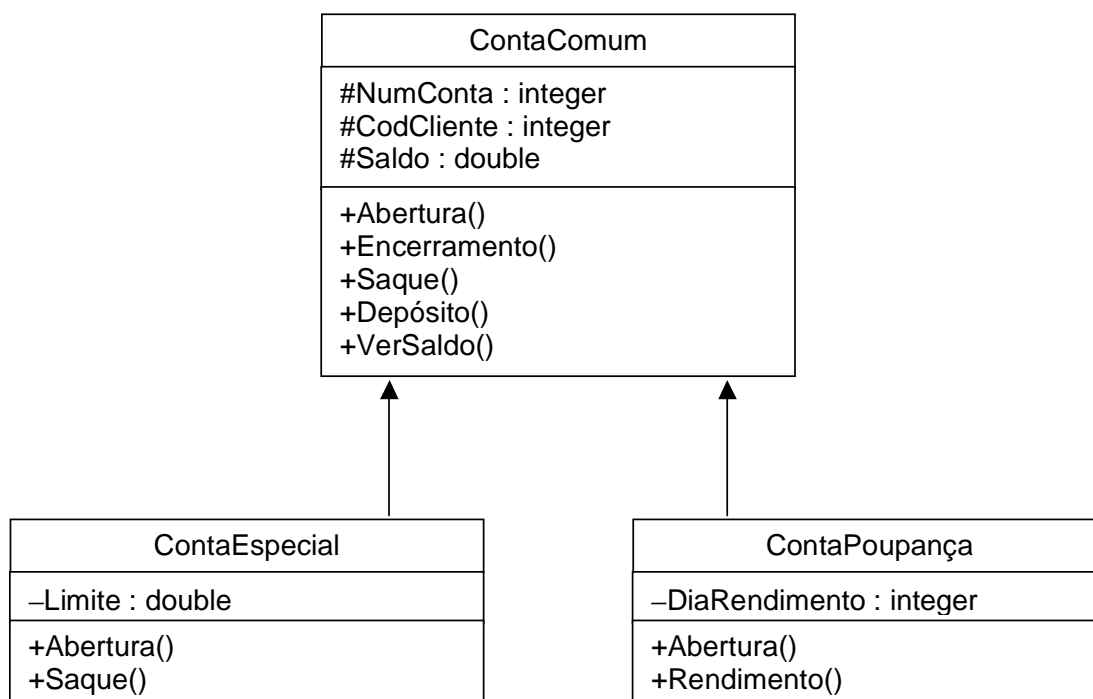


Figura 3.7. Exemplo de herança e polimorfismo.

Outro conceito importante do paradigma de Orientação a Objetos é o

polimorfismo, que está relacionado com a re-declaração de métodos previamente herdados por uma classe. Algumas vezes, os métodos herdados por uma classe descendente precisam ser refeitos visto que seu comportamento, de alguma forma, é diferente do comportamento definido na classe ancestral.

No diagrama da figura 3.7, percebe-se que os métodos Abertura e Saque da classe ContaEspecial foram declarados novamente. De fato, ao abrir uma conta especial é necessário informar o limite extra de saque e, ao realizar a operação de saque, é possível atribuir um valor negativo ao saldo da conta limitado a este valor. Desta forma, apesar das operações de Abertura e Saque terem sido herdadas da classe ContaComum, as contas especiais necessitam redefinir tais operações, garantindo um funcionamento correto. O mesmo acontece com as contas de poupança onde o dia de rendimento deve ser informado na operação de Abertura, possibilitando que operação Rendimento seja também realizada corretamente.

Outra informação importante pode ser visualizada no diagrama da figura 3.7. Os atributos da classe ContaComum possuem visibilidade protegida. Desta forma, é impossível a um objeto externo, por exemplo um objeto SerHumano, acessar diretamente o saldo de uma conta. Alterações no saldo são necessariamente realizadas através das operações de Saldo e Depósito. Entretanto, as classes descendentes ContaEspecial e ContaPoupança precisam manipular os atributos herdados da classe ContaComum para definir corretamente seus comportamentos. Esta é a aplicação da visibilidade protegida: permitir que objetos de classes descendentes acessem determinadas funcionalidades definidas na classe ancestral mas evitando o acesso a objetos do meio exterior.

Uma característica bastante relevante do conceito de herança está no fato de permitir que classes descendentes sejam definidas a partir do código compilado de uma classe ancestral. Ou seja, é possível especializar a funcionalidade de uma classe sem dispor de seu código fonte. Esta característica evidencia a grande capacidade que o paradigma da Orientação a Objetos traz a atividade de modelagem e desenvolvimento de softwares e, de certa forma, justifica a grande aceitação deste paradigma na comunidade científica e tecnológica.

3.3. Linguagem de Modelagem Unificada

A Linguagem de Modelagem Unificada (*Unified Modeling Language* – UML) é uma linguagem visual utilizada para modelar sistemas computacionais baseados no paradigma de Orientação a Objetos. A UML, como é normalmente conhecida, tornou-se, nos últimos anos, a linguagem padrão de modelagem de sistemas adotada pelos profissionais da área de Engenharia de Software, [25].

O objetivo da UML é auxiliar os engenheiros de software a definir as características de um sistema computacional, tais como seus requisitos, seu comportamento, sua estrutura lógica, a dinâmica de seus processos e até mesmo as necessidades físicas relacionadas com os equipamentos necessários ao seu funcionamento. Para isto, a UML define uma série de diagramas gráficos cujo objetivo é fornecer múltiplas visões do sistema computacional a ser projetado, analisando-o e modelando-o sob diversos aspectos, de forma que o processo de modelagem como um todo atinja sua plenitude. Cada diagrama da linguagem analisa o funcionamento do sistema sob uma determinada ótica, partindo de uma visão externa ou macro do sistema e atingido as camadas mais profundas do software, onde um enfoque mais técnico do sistema é visualizado. A utilização de diversos diagramas permite que falhas sejam descobertas, diminuindo a possibilidade de ocorrência de falhas no futuro.

Os principais diagramas definidos pela UML e relevantes a modelagem do ambiente computacional proposto são apresentados em seguida.

3.3.1. Diagrama de Casos de Uso

O Diagrama de Casos de Uso procura, por meio de uma linguagem simples, possibilitar a compreensão do comportamento externo do sistema por qualquer pessoa, apresentando a idéia geral de seu funcionamento. Este diagrama normalmente é utilizado no início do processo de modelagem e objetiva apresentar uma visão das funções e serviços que o sistema deve oferecer aos usuários, sem se

preocupar com os detalhes de implementação nem com questões relacionadas à temporalidade dessas funções, ou seja, em que momento elas podem ocorrer, [25].

O Diagrama de Casos de Uso concentra-se em dois itens principais: Atores e Casos de Uso. Os atores representam os papéis desempenhados pelos usuários do sistema na utilização de suas funções ou serviços. Eventualmente, os atores podem representar também um hardware ou outro software que interaja com o sistema que está sendo modelado. Os casos de uso referem-se aos serviços ou funções que podem ser utilizadas pelos usuários do sistema, como emitir um relatório ou realizar algum cadastro. Os casos de uso são utilizados para expressar e documentar os comportamentos pretendidos para o sistema computacional.

A figura 3.8 mostra um diagrama de casos de uso simplificado para um sistema bancário. Os atores do sistema são identificados por “bonecos magros”, contendo uma breve descrição do seu papel logo abaixo do símbolo. No exemplo, são definidos como atores os clientes do banco e a instituição bancária, a qual gerencia o funcionamento das contas dos clientes.

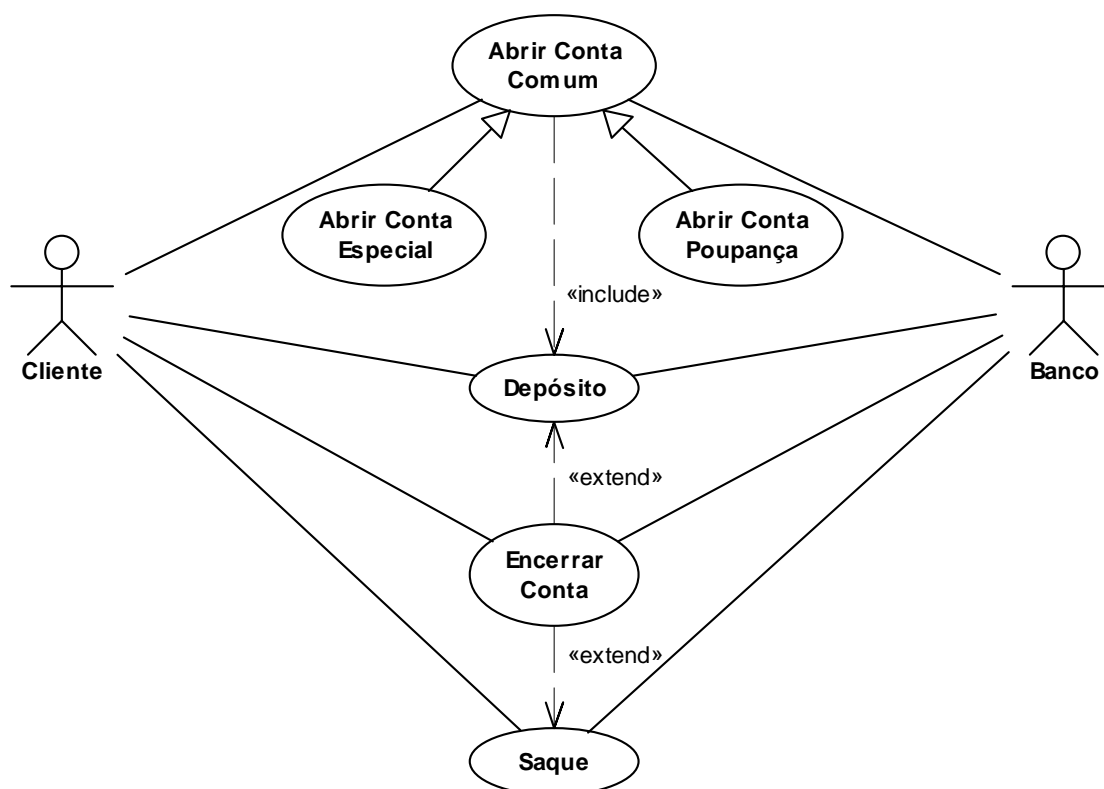


Figura 3.8. Diagrama de Casos de Uso.

Os casos de uso são identificados por elipses contendo uma descrição da

função ou serviço do sistema ao qual o caso de uso se refere. Os segmentos de reta interligando atores e casos de uso são denominados de associações. As associações demonstram que um ator utiliza-se da função do sistema representada pelo caso de uso, seja requisitando a execução da função ou recebendo o resultado produzido por ela a pedido de outro ator. O diagrama do exemplo mostra que um cliente pode requisitar abertura e fechamento de contas, bem como realizar depósitos e saques em uma conta. A instituição bancária interage com esses serviços processando as requisições dos clientes.

Relacionamentos entre casos de uso recebem nomes especiais como Inclusão, Extensão e Generalização-Especialização. O relacionamento de generalização-especialização ocorre quando casos de uso diferentes possuem características em comum. No exemplo dado, os casos de uso Abrir Conta Especial e Abrir Conta Poupança, por exemplo, são especializações do caso de uso Abrir Conta Comum, que especifica a funcionalidade comum aos três. Esses relacionamentos são representados por setas partindo do caso de uso especializado para o caso de uso geral.

As inclusões costumam ocorrer quando existe um serviço comum a mais de um caso de uso. Os relacionamentos de inclusão indicam uma obrigatoriedade, ou seja, quando um determinado caso de uso possui um relacionamento de inclusão com outro, a execução do primeiro obriga também a execução do segundo. No exemplo dado, observa-se a obrigatoriedade de realizar um depósito quando alguma conta for aberta pelo cliente. As inclusões são representadas por setas tracejadas partindo do caso de uso principal para os casos de uso incluídos e são identificadas pelo texto “<<include>>”.

As extensões são utilizadas para descrever cenários opcionais de um caso de uso. Os casos de uso estendidos normalmente descrevem situações que poderão ou não ocorrer. Os relacionamentos de extensão representam eventos que não acontecem sempre, o que não significa que sejam incomuns. No exemplo dado, observa-se que, ao encerrar uma conta, o cliente pode necessitar realizar um depósito, caso o saldo esteja negativo; um saque, caso este esteja positivo; ou

simplesmente nenhum dos dois. As extensões são representadas por setas tracejadas partindo do caso de uso estendido para o outro caso de uso e identificadas pelo texto “<<extend>>”.

3.3.2. Diagrama de Classes

O Diagrama de Classes é o diagrama mais importante e utilizado da UML. Seu principal enfoque está em permitir a visualização das classes do sistema computacional que está sendo modelado. O diagrama apresenta também uma visão estática da organização das classes, de seus relacionamentos e associações, possibilitando uma análise geral da estrutura lógica do sistema, [25].

Como visto anteriormente, as classes costumam possuir atributos, que armazenam os dados dos objetos da classe, e métodos que são as operações que as instâncias podem realizar. Os valores dos atributos podem variar de instância para instância, permitindo identificar um a um cada objeto da classe, enquanto que os métodos de uma classe são normalmente idênticos para todos seus objetos.

Apesar de listar os atributos e métodos das classes de um sistema, os diagramas de classe não se preocupam em definir a seqüência de execução do sistema, sendo esta tarefa atribuída a outros diagramas da linguagem. De fato, além da definição dos atributos e métodos das classes, as principais informações apresentadas por estes diagramas referem-se ao conceito de generalização-especialização e aos relacionamentos de agregação entre as classes.

O conceito de generalização-especialização, explorado anteriormente, refere-se à capacidade de uma classe herdar características de uma outra, sendo representado no diagrama de classes por uma seta partindo da classe descendente para a ascendente.

As agregações, por sua vez, determinam a forma com que as instâncias de uma classe estão ligadas a instâncias de outras classes e em geral são usadas para demonstrar que as informações de um objeto, conhecido por objeto-todo, precisam ser complementadas por informações contidas em um ou mais objetos de outra classe, chamados objetos-parte. No escopo deste trabalho, as agregações que

associam um objeto-todo a um único objeto-parte são representadas por uma seta com um losango na extremidade da classe do objeto-todo. As agregações que associam um objeto-todo a vários objetos-parte apresentam um retângulo com o texto “*index*” junto ao losango.

A figura 3.9 mostra um diagrama de classes simplificado para o modelo de grafos utilizado pelo *ModSym*, no qual foram omitidos os atributos e métodos das classes. A classe *TGrafo*, que representa um grafo, possui relacionamentos de agregação com as classes *TListVertice* e *TListRamo*, as quais representam as listas de vértices e ramos do grafo, respectivamente. Estas agregações são de um para um, visto que um grafo, ou seja, um objeto da classe *TGrafo*, possui apenas uma lista de vértices e uma lista de ramos.

A classe *TListVertice* possui uma agregação de um para vários com a classe *TVertice*. De fato, uma lista de vértices está associada a vários vértices, que são objetos da classe *TVertice*. O mesmo acontece para a classe *TListRamo* em relação à classe *TRamo*. Finalmente, a classe *TRamo* possui duas agregações de um para um com a classe *TVertice*, onde são associados os vértices terminais de cada ramo, denominados no exemplo de vértices origem e destino.

O diagrama mostra ainda que as classes *TListVertice* e *TListRamo* são especializações da classe *TList*. A classe *TList* faz parte da biblioteca de classes do *Borland Delphi/Kylix* e implementa toda a funcionalidade necessária para manipular listas de objetos genéricos. Especializações de *TList* são comumente utilizadas para manipular lista de objetos de uma classe específica. No exemplo, especializações de *TList* são utilizadas para manipular os vértices e os ramos de um grafo.

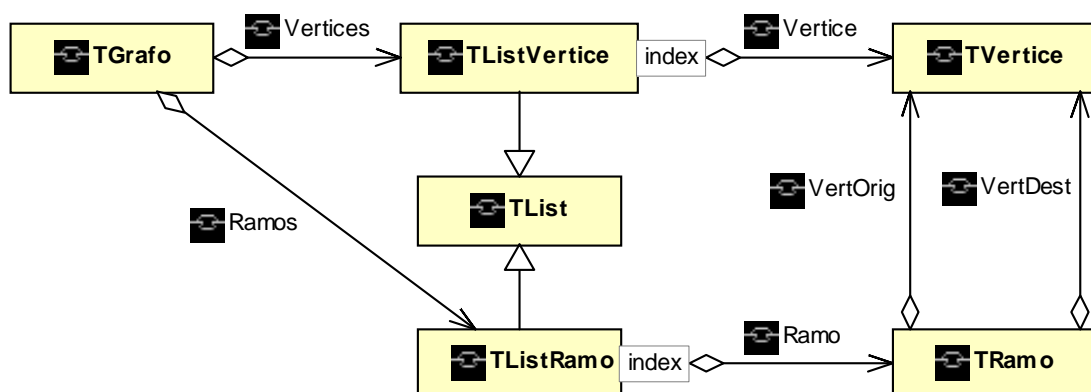


Figura 3.9. Diagrama de Classes.

3.3.3. Diagrama de Estruturas Compostas

O Diagrama de Estruturas Compostas é utilizado para modelar Colaborações. Uma colaboração descreve a visão de um conjunto de instâncias que cooperam entre si para executar uma tarefa ou um grupo específico de tarefas, [25].

As colaborações são representadas por elipses tracejadas, contendo uma descrição da tarefa a ser executada, e engloba ou interliga os objetos que cooperam entre si para realizá-la. A figura 3.10 apresenta um diagrama de estruturas compostas onde dois objetos, um cliente e o gerente de um banco, cooperam para a realização da atividade Realizar Empréstimo.

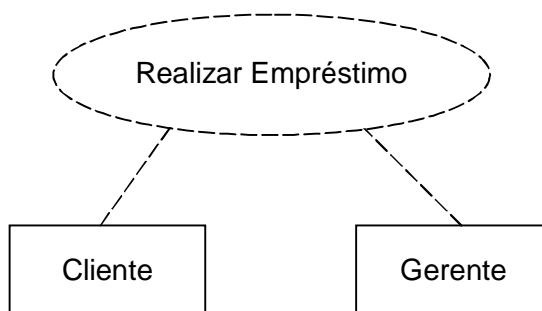


Figura 3.10. Diagrama de Estruturas Compostas.

Uma colaboração pode também agrupar um conjunto de colaborações reunindo colaborações relacionadas entre si. A figura 3.11 apresenta o diagrama da colaboração Comprar Imóvel que agrupa os objetos: Cliente, Gerente e Vendedor e as colaborações: Realizar Empréstimo e Liberar Pagamento. Neste exemplo, o Gerente e o Vendedor negociam a liberação do pagamento para a compra do imóvel realizada pelo Cliente.

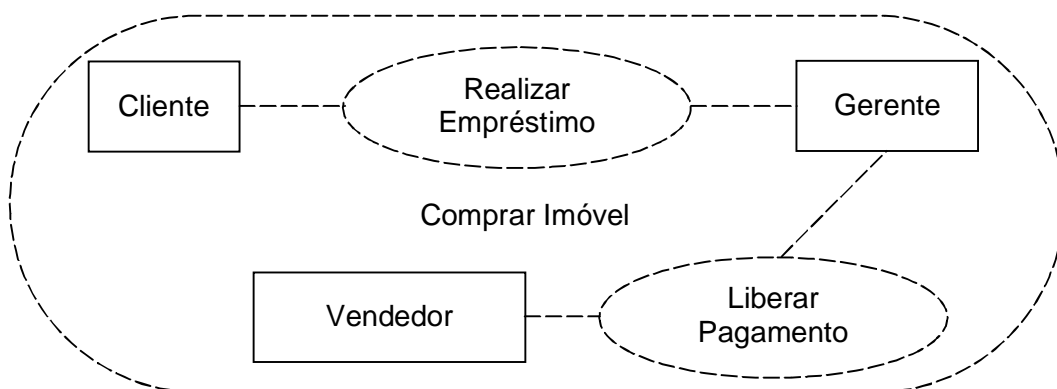


Figura 3.11. Agrupamento de Colaborações.

3.3.4. Diagrama de Colaboração

O Diagrama de Colaboração preocupa-se com os vínculos existentes entre os diversos objetos do sistema e, em particular, nas mensagens trocadas entre os objetos envolvidos na realização de uma determinada tarefa. Em geral, baseia-se nos diagramas de casos de uso e de classes para determinar os objetos relacionados na realização de um processo específico. O diagrama pode ser utilizado para detalhar as colaborações identificadas no diagrama de estruturas compostas.

Os diagramas de colaboração são constituídos por atores, objetos, vínculos e mensagens. Os atores, quando presentes, representam as entidades externas que interagem com o sistema e solicitam os serviços, gerando os eventos que iniciam um processo. São representados pelo mesmo símbolo usado no diagrama de casos de uso.

Os objetos representam as instâncias das classes envolvidas no processo, sendo simbolizados por um retângulo contendo um identificador para o objeto e, em seguida, o nome de sua classe. O relacionamento entre os objetos é representado por um segmento de reta que vincula um objeto a outro.

Finalmente, as mensagens demonstram a ocorrência dos eventos relacionados ao processo e normalmente identificam a invocação de um método. As mensagens são representadas por setas entre dois componentes do diagrama, atores ou objetos, orientadas na direção do receptor da mensagem. As mensagens são numeradas, dando uma noção temporal da realização de uma tarefa.

A figura 3.12 mostra um diagrama de colaboração que detalha a atividade Realizar Empréstimo. O ator banco dispara a mensagem ConsultarCPF para verificar as informações de um cliente. Se o cadastro do cliente estiver desatualizado, o método Atualizar é invocado para atualizar seus dados. Em seguida, a mensagem NovoEmprestimo inicia o processo de cadastro do empréstimo do cliente. Quando este cadastro é realizado, a mensagem Depósito é disparada para a conta do cliente, aumentando o seu saldo.

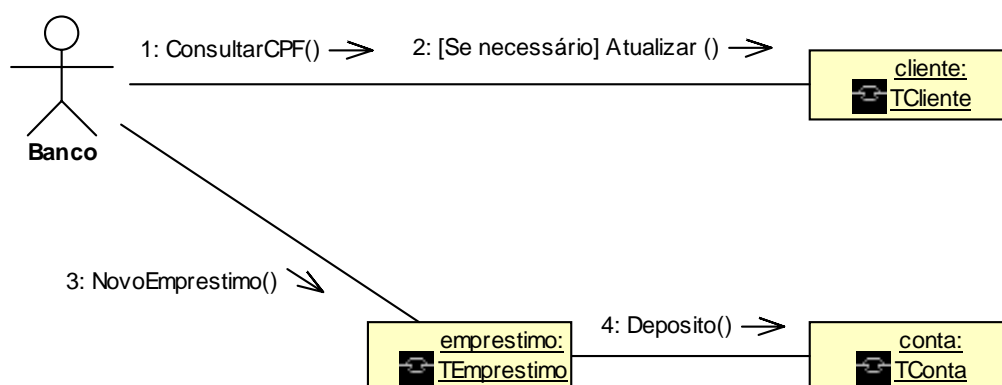


Figura 3.12. Diagrama de Colaboração.

3.3.5. Diagrama de Atividades

O Diagrama de Atividades é o diagrama com maior ênfase ao nível de algoritmo da UML, apresentando muita semelhança com os antigos fluxogramas utilizados para desenvolver a lógica de programação. O diagrama determina o fluxo de execução de um algoritmo, permitindo inclusive a utilização de pseudocódigo ou de códigos de linguagens de programação como Pascal e C, [25].

O Diagrama de Atividades procura descrever os passos percorridos para a conclusão de uma atividade específica, muitas vezes representada por um método com um certo grau de complexidade. Em alguns casos, entretanto, pode ser utilizado para representar um processo completo, assumindo uma função semelhante à função do diagrama de colaboração.

Os principais componentes dos diagramas de atividades são o estado de ação, o ponto de decisão e os estados inicial e final. Os estados de ação representam a realização de uma ação dentro de um fluxo de controle, sendo simbolizados por um retângulo com as bordas arredondadas. Em geral, os estados de ação são atômicos, ou seja, não podem ser decompostos em outros estados.

O ponto de decisão representa um ponto do fluxo onde deve ser realizado um teste, ou seja, uma tomada de decisão. De acordo com esta decisão, o fluxo optará por seguir um caminho ou outro dentro do diagrama. O ponto de decisão é representado por um símbolo de losango, de onde partem pelo menos duas transições, indicando os fluxos alternativos a serem executados. Os pontos de

decisão são também usados para unir fluxos de controle divididos anteriormente por outros pontos de decisão.

Finalmente, os estados inicial e final determinam, respectivamente, o início e o fim do diagrama de atividades. Os símbolos utilizados para representar esses estados podem ser vistos no diagrama a seguir.

A figura 3.13 mostra um diagrama de atividades que representa o método ConsultarCPF do exemplo apresentado no diagrama da figura 3.12. O diagrama mostras as etapas de execução da consulta de clientes, iniciando com o recebimento do CPF do cliente que se deseja consultar. Em seguida, tenta-se encontrar o cliente identificado por este número. Caso a busca obtenha sucesso, os dados do cliente são retornados; se não, é retornada a mensagem “Cliente não encontrado”.

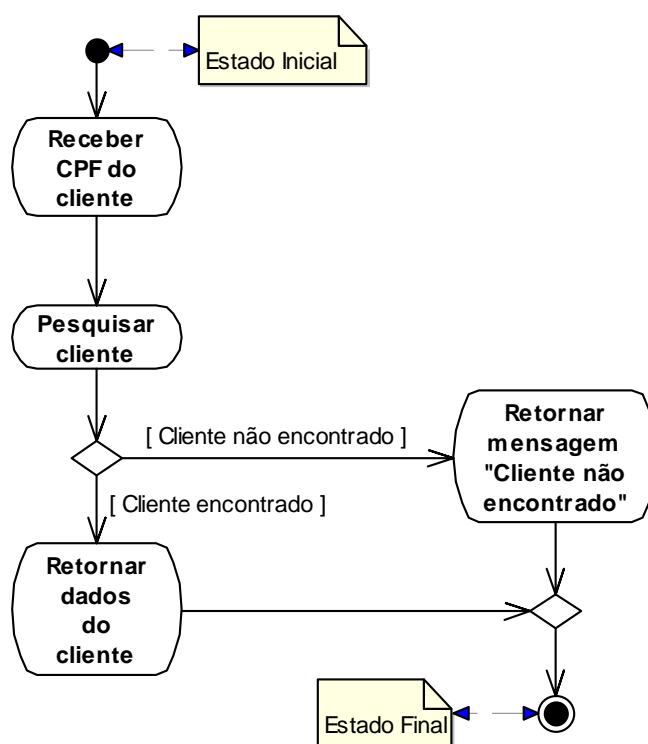


Figura 3.13. Diagrama de Atividades.

É importante perceber que os Diagramas de Estruturas Compostas, de Colaboração e de Atividade apresentam a visão completa de um sistema computacional partindo de uma análise geral do seu funcionamento até os detalhes de implementação de cada das tarefas ou serviços fornecidos aos usuários.

4. Modelagem do ModSym

4.1. Introdução

Este capítulo apresenta os fundamentos mais importantes da modelagem do ambiente computacional proposto: o *ModSym*. São apresentados os aspectos mais relevantes do projeto do software relacionados à sua funcionalidade, ao desenvolvimento da interface gráfica de interação com o usuário e às estruturas de dados que dão suporte à implementação dos algoritmos requeridos pelo ambiente computacional.

4.2. Objetivos e Funções do Software

De uma forma geral, as primeiras etapas no desenvolvimento de um software são denominadas de Levantamento e Análise de Requisitos, [25], as quais objetivam determinar a funcionalidade do sistema de computação que esta sendo modelado.

Conforme exposto na introdução do trabalho, o objetivo do ambiente computacional proposto é auxiliar o processo de ensino e aprendizagem na área de Engenharia de Controle, especificamente na atividade de modelagem de sistemas físicos lineares. A avaliação contextual desta atividade no âmbito educacional, bem como das ferramentas e algoritmos computacionais relacionados a ela, também discutidos anteriormente, definiram os requisitos de funcionamento do ambiente computacional. Em resumo, o *ModSym* possibilita a um estudante da Engenharia de Controle a realizar as seguintes tarefas:

- Modelar um sistema físico com base no seu desenho gráfico realizado a partir de
-

um conjunto de elementos dos sistemas elétrico, mecânico translacional, mecânico rotacional e hidráulico;

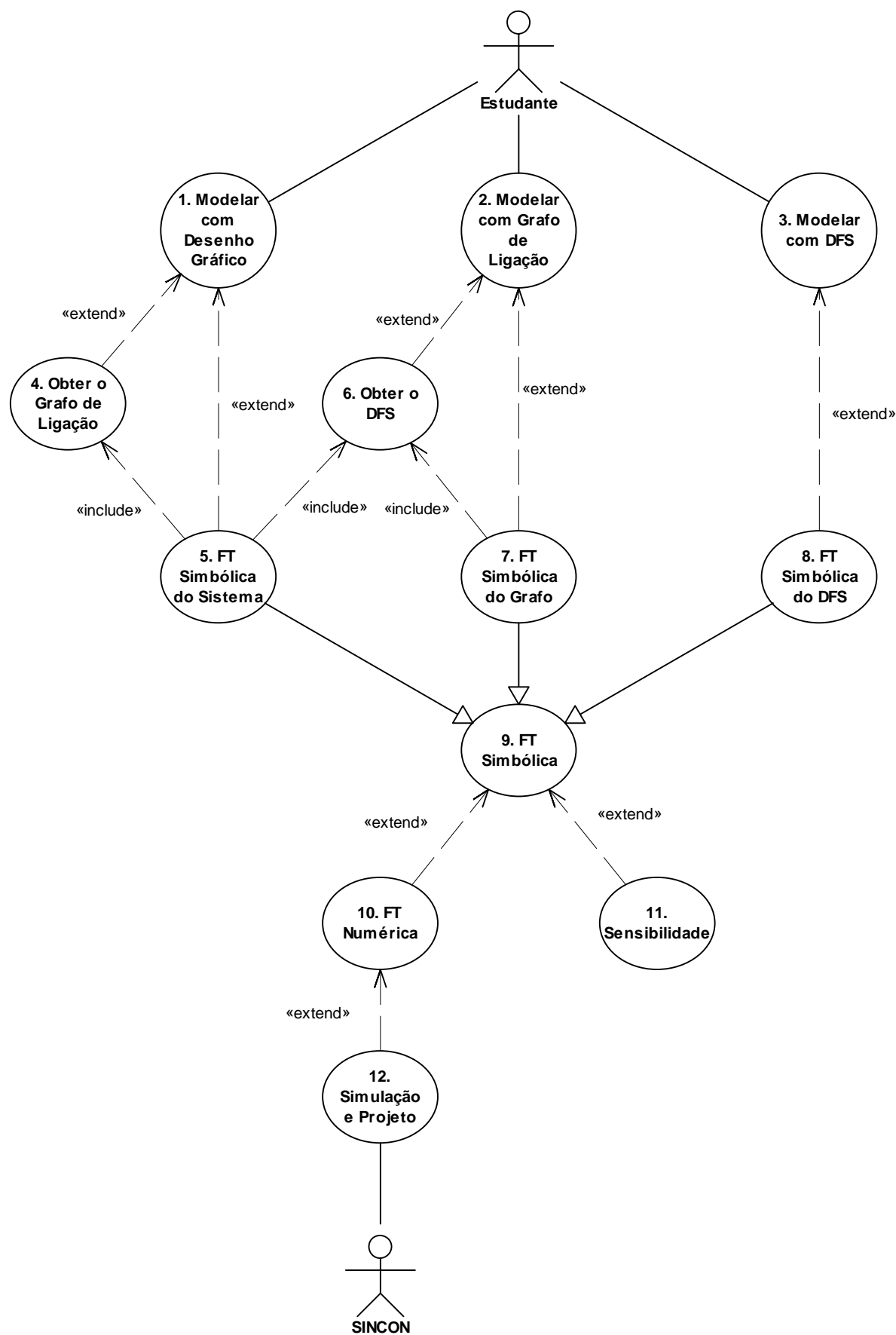
- Modelar um sistema físico com base no seu grafo de ligação realizado a partir de um conjunto de elementos generalizados como fontes, acumuladores, dissipadores, modeladores, transformadores e acopladores de energia;
- Modelar um sistema físico a partir de um diagrama de fluxo de sinal;
- Obter o grafo de ligação de um sistema físico;
- Obter o diagrama de fluxo de sinal de um sistema físico;
- Calcular funções de transferência de um sistema físico na forma simbólica;
- Calcular funções de transferência na forma numérica e exportá-las para o SINCON, [10, 11], possibilitando a análise e a síntese de sistemas de controle;
- Calcular a sensibilidade da função de transferência simbólica em relação a algum parâmetro do sistema.

4.3. Diagrama de Casos de Uso

Uma vez definidos os objetivos do software, sua funcionalidade pôde, então, ser sistematicamente modelada em um diagrama de casos de uso, que identificou os atores e as funções requeridas pelo sistema. A figura 4.1 mostra um diagrama de casos de uso simplificado para o *ModSym*, evidenciando os atores e as tarefas pertinentes ao ambiente computacional. Os casos de uso foram numerados para facilitar a referência a cada um deles em particular.

Conforme visto no diagrama, o ator principal do sistema é um estudante que pode utilizar inicialmente os casos de uso numerados de 1 a 3. Estes casos de uso se referem respectivamente a: modelar um sistema físico a partir do seu desenho gráfico, modelar um sistema físico a partir do seu grafo de ligação e modelar um sistema físico a partir do diagrama de fluxo de sinal (DFS).

Quando o caso de uso Modelar com DFS é realizado, o ator pode utilizar o caso de uso 8 que se refere ao cálculo da função de transferência (FT) simbólica do sistema físico a partir deste diagrama.

Figura 4.1. Diagrama de Casos de Uso do *ModSym*.

Se o caso de uso Modelar com Grafo de Ligação é utilizado, o estudante pode realizar em seguida duas tarefas: obter o DFS do sistema a partir deste grafo, dado pelo caso de uso 6, ou calcular a função de transferência simbólica do sistema, utilizando o caso de uso 7. Observe que o cálculo da FT a partir do grafo de ligação implica necessariamente na obtenção do DFS do sistema; entretanto, o ator pode realizar o cálculo desta função sem obter previamente o diagrama de fluxo de sinal. Nestes casos, o software realiza o caso de uso de obtenção do DFS internamente de forma transparente ao usuário.

O caso de uso Modelar com Desenho Gráfico trata da modelagem do sistema físico a partir da interligação dos ícones gráficos que representam seus elementos físicos. Após definir o sistema utilizando este caso de uso, o ator pode obter o seu grafo de ligação ou calcular a função de transferência simbólica, realizando os casos de uso 4 ou 5, respectivamente. No cálculo da FT, os casos de uso Obter o Grafo de Ligação e Obter o DFS são realizados internamente pelo software.

O diagrama da figura 4.1 mostra ainda que os casos de uso 5, 7 e 8, relacionados ao cálculo da função de transferência na forma simbólica, são especializações do caso de uso 9, que generaliza todas as possibilidades de realização desta tarefa. Após a obtenção da FT na forma simbólica, o estudante pode utilizar os casos de uso FT Numérica, para calcular a FT na forma numérica, e Sensibilidade, para calcular a sensibilidade normalizada da FT em relação a algum parâmetro do sistema. Finalmente, após calcular a FT na forma numérica, o estudante pode salvar um arquivo com esta função no formato do SINCON e realizar a análise e a síntese do sistema de controle.

Os casos de uso costumam ser documentados, normalmente por meio de linguagens simples, ou seja, sem formalidade técnica, onde descreve-se em linhas gerais os atores que interagem com o caso de uso, as etapas executadas pelos atores e pelo sistema, as informações e condições necessárias a sua execução, e finalmente as restrições a qual estão submetidos. Entretanto, para não tornar o trabalho demasiadamente extenso, uma vez que os casos de uso já foram comentados um a um, é fundamental ressaltar que o cálculo da função de

transferência dos sistemas de controle é garantido apenas aos sistemas que possuam implementação física. Casos esdrúxulos, tais como a aplicação de velocidades diferentes a um mesmo corpo ou de correntes diferentes a um dispositivo elétrico, por exemplo, são restrições à realização dos casos de uso utilizados para calcular a função de transferência do sistema físico.

No diagrama da figura 4.1 também foram omitidos os casos de uso mais simples, como abrir, salvar e iniciar modelos; inserir, excluir e conectar elementos em um modelo. Em síntese, foram apresentados os casos de uso mais importantes para o funcionamento global do sistema computacional.

4.4. Diagrama de Estruturas Compostas

Uma vez definida a funcionalidade básica do sistema em termos de casos de uso, o passo seguinte foi visualizar o comportamento geral do sistema computacional em um ambiente orientado ao objeto. Esta análise teve por objetivo identificar as principais classes que deveriam compor o software e as principais colaborações realizadas por elas.

O diagrama da figura 4.2 mostra as principais colaborações do ambiente computacional. Neste diagrama em particular, a maioria das colaborações foi representada por setas tracejadas, ao invés de elipses, procurando melhorar a visualização do comportamento global do software. Além disso, as colaborações foram também numeradas para facilitar a referência a cada uma delas.

Conforme visto no diagrama, a modelagem do software foi dividida em dois grandes grupos. O primeiro grupo está relacionado à construção da interface gráfica de modelagem, que permite aos usuários modelar os sistemas físicos através dos três diagramas gráficos já citados anteriormente: desenho gráfico, grafo de ligação e DFS. Esta interface é composta por um conjunto de classes que representa tanto os diagramas gráficos do sistema físico quanto os elementos físicos que os compõem. A colaboração 1, portanto, é responsável pela implementação das funções associadas aos casos de uso relacionados à modelagem do sistema físico.

O segundo grupo está relacionado à definição das estruturas de dados utilizadas para representar cada um dos diagramas gráficos na memória do computador, englobando ainda a implementação dos algoritmos necessários ao cálculo da função de transferência do sistema físico. O conjunto de classes relacionadas à colaboração 2 implementa as funções associadas aos demais casos de uso do ambiente computacional.

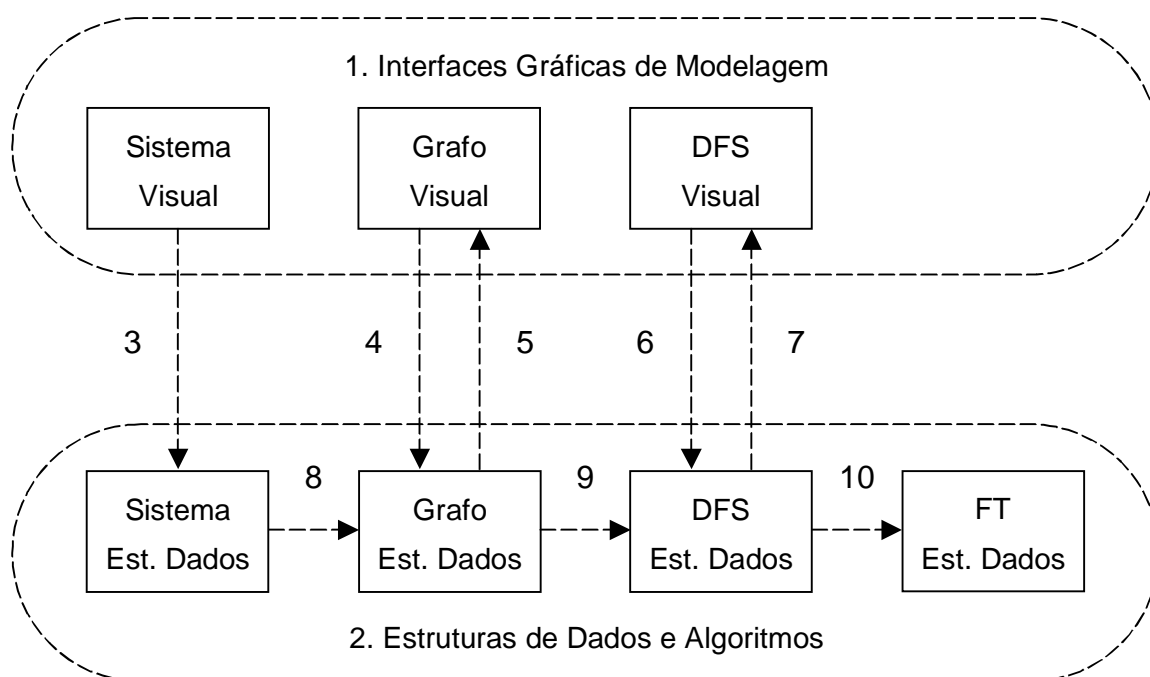


Figura 4.2. Diagrama de Estruturas Compostas do *ModSym*.

As colaborações de 3 a 7 estão relacionadas à conversão da representação visual de um diagrama gráfico do sistema físico para uma estrutura de dados equivalente, e vice-versa. O termo visual refere-se, aqui, à representação de cada diagrama na interface gráfica de modelagem, ou seja, a imagem do diagrama formada a partir de ícones de seus elementos e interconexões. O termo estrutura de dados, conforme citado anteriormente, refere-se à forma sistemática de representação dos diagramas na memória. Em síntese, a colaboração 3 mapeia a conversão do desenho gráfico do sistema físico na estrutura de dados equivalente; as colaborações 4 e 5 mapeiam as conversões entre o grafo de ligação visual e sua estrutura de dados e as colaborações 6 e 7, as conversões relacionadas ao diagrama de fluxo de sinal.

Finalmente, as colaborações 8, 9 e 10 estão associadas aos principais algoritmos necessários ao ambiente computacional, estando relacionadas respectivamente: à obtenção do grafo de ligação a partir do desenho gráfico do sistema, à obtenção do DFS a partir do grafo de ligação e à implementação da regra de Mason que calcula a função de transferência na forma simbólica.

4.5. Diagramas de Classes

A definição do comportamento de um sistema computacional em termos de colaborações auxilia a identificação das classes que devem compor o software. Desta forma, o passo seguinte na modelagem do sistema computacional proposto foi definir as classes constituintes dos dois blocos básicos do sistema: a interface gráfica de modelagem e as estruturas de dados.

4.5.1. Interface Gráfica de Modelagem

O primeiro passo na definição da interface gráfica consistiu em analisar a funcionalidade requerida por ela. A interface deveria permitir ao usuário do software construir, de forma bastante amigável, os três tipos de diagramas gráficos que representam um sistema físico: o desenho gráfico, o grafo de ligação e o diagrama de fluxo de sinal. Os diagramas deveriam ser constituídos por um conjunto de ícones interligados entre si, onde cada ícone representaria um elemento do diagrama. Para os desenhos gráficos, por exemplo, os ícones representariam os elementos físicos como resistores, amortecedores e reservatórios, dentre outros. Nos grafos, os ícones representariam os vértices e os elementos generalizados e nos DFS's, vértices e ramos.

Além disso, para facilitar a tarefa de modelagem, seria conveniente que a interface apresentasse os diversos ícones gráficos disponíveis para a montagem dos sistemas físicos, permitindo que eles fossem incluídos, excluídos e manipulados dentro de cada diagrama. A interface deveria também manipular três tipos de diagramas diferentes, além de possibilitar que vários deles fossem visualizados simultaneamente, permitindo ao estudante observar ao mesmo tempo o desenho

gráfico, o grafo de ligação e o DFS de um sistema físico.

Foi fácil perceber que a funcionalidade requerida pela interface de modelagem do ambiente computacional proposto era bastante similar àquela apresentada pelos ambientes integrados de desenvolvimento de softwares, como por exemplo o Delphi, onde um conjunto de objetos é disponibilizado para a construção de aplicativos computacionais. Conseqüentemente, a idéia do funcionamento desses ambientes de programação serviu de modelo para a definição da interface gráfica de modelagem do *ModSym*.

O diagrama de classes, apresentado na figura 4.3, mapeia as classes que foram necessárias à implementação da interface de modelagem do software, baseada no modelo citado anteriormente. A classe *TFormMain* implementa a funcionalidade da janela principal do ambiente computacional, listando os ícones gráficos disponíveis para a construção dos diagramas gráficos. Ela mantém uma lista de todos os diagramas abertos, controlando as operações relacionadas a arquivo, como iniciar, fechar, abrir e salvar diagramas no disco. A lista de diagramas gráficos é representada pela classe *TListVisualForm*.

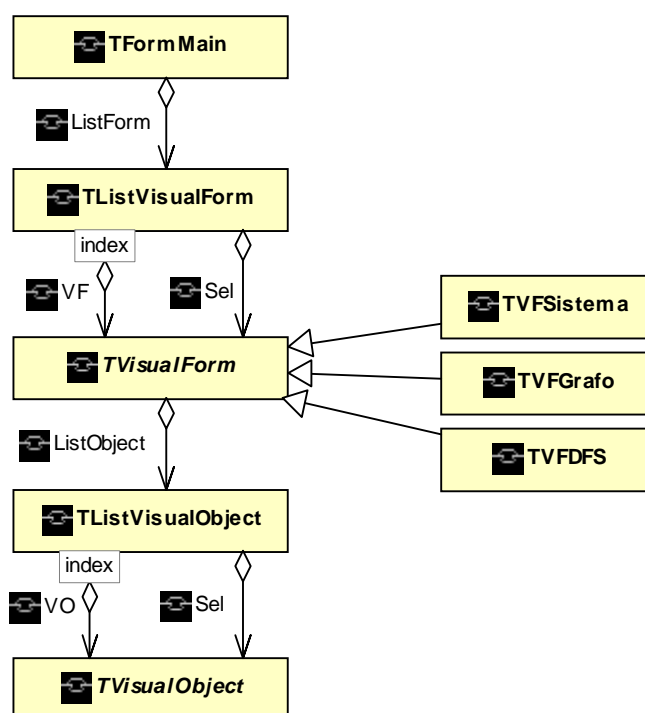


Figura 4.3. Diagrama de Classes da interface gráfica de modelagem.

A classe *TVisualForm*, que representa um diagrama gráfico, implementa a

funcionalidade básica comum aos três tipos de diagramas em estudo, sendo especializada pelas classes TVFSistema, TVFGrafo e TVFDFS, associadas respectivamente aos desenhos gráficos, grafos e DFS's visuais dos sistemas físicos. A classe TVisualForm e suas descendentes mantêm uma lista de todos ícones gráficos incluídos no diagrama. Esta lista é representada pela classe TListVisualObject e seus itens, os ícones gráficos, pela classe TVisualObject.

Em resumo, a interface gráfica de modelagem mantém uma lista de diagramas e cada diagrama, uma lista dos objetos gráficos pertencentes a ele.

Uma vez definida a estrutura básica de funcionamento da interface, o passo seguinte foi definir os tipos de ícones gráficos que deveriam compor os diagramas. É importante lembrar esses diagramas são constituídos por um conjunto de ícones interligados entre si. Isto é, não bastava apenas representar graficamente os elementos do diagrama, as conexões entre eles deveriam ser também visualizadas.

Desta forma, os diagramas gráficos passaram a ser constituídos por dois tipos de objetos: os elementos, representando evidentemente cada elemento do diagrama e as conexões, representando a interligação entre eles. Entretanto, uma análise mais aprofundada a respeito dos diversos tipos de elementos necessários para a montagem dos diagramas, como por exemplo, elementos elétricos, mecânicos, hidráulicos, elementos generalizados, dentre outros, e das conexões entre esses elementos, acarretou na adoção do seguinte modelo:

- Os elementos foram divididos em dois grupos, denominados de vértices e componentes.
 - Os vértices possuem a capacidade de realizar um número qualquer de conexões entre os diversos elementos do diagrama.
 - Os componentes realizam um número determinado de conexões, onde cada conexão é realizada a partir de um conector, que identifica os terminais físicos do componente.
 - Vértices, conexões e conectores estão associados a um domínio físico, impedindo a interligação entre elementos de domínios físicos diferentes sem a utilização de um acoplador.
-

Em resumo, o modelo adotado causou a divisão dos elementos dos diagramas em dois grandes grupos: componentes e vértices. Desta forma, os diagramas passaram a ser constituídos por três tipos de básicos de objetos: componentes, vértices e conexões. Os componentes e vértices representam os elementos do sistema e as conexões, a interligação entre eles. Além disso, a associação de um domínio físico a cada grupo acarretou com a divisão deles em subgrupos associados a cada um dos domínios físicos em estudo.

O diagrama da figura 4.4 mostra as especializações da classe TVisualObject, acarretadas pelos agrupamentos definidos no modelo. O primeiro nível de especialização de TVisualObject mostra que um objeto do diagrama pode ser um elemento, representado pela classe Tvoelemento, ou uma conexão, representada pela classe Tvoconexao. Os elementos são então especializados em componentes (Tvocomp) e vértices (Tvovertice), conforme citado anteriormente.

Os três principais grupos: componentes, vértices e conexões são então especializados para cada um dos domínios físicos em estudo: elétrico (Ele), hidráulico (Hid), mecânico rotacional (Mcr) e mecânico translacional (Mct), além dos “domínios” definidos para os grafos de ligação (Grf) e diagramas de fluxo (Dfs), conforme mostra os níveis subseqüentes de especialização no diagrama. Finalmente, um componente do sistema físico pode ser classificado como acoplador, com a função de interligar elementos de dois domínios físicos. Esses componentes são representados pela classe Tvoacp.

Na verdade, a razão de tantos níveis de especialização está em identificar atributos e comportamentos comuns a determinados grupos de objetos. Assim, a classe TVisualObject, por exemplo, implementa todas as rotinas relacionadas à movimentação e seleção dos objetos em um diagrama, bem como as funções referentes a arquivos. A classe Tvoelemento representa os objetos que realizam conexões; a classe Tvocomp mapeia os objetos que possuem um ganho; a classe Tvosiscomp agrupa os objetos cujo ícone gráfico é constituído por um bitmap e que podem ser girados dentro do diagrama. Enfim, as classes implementam, passo a passo, o comportamento requerido para cada grupo de objetos definidos no modelo.

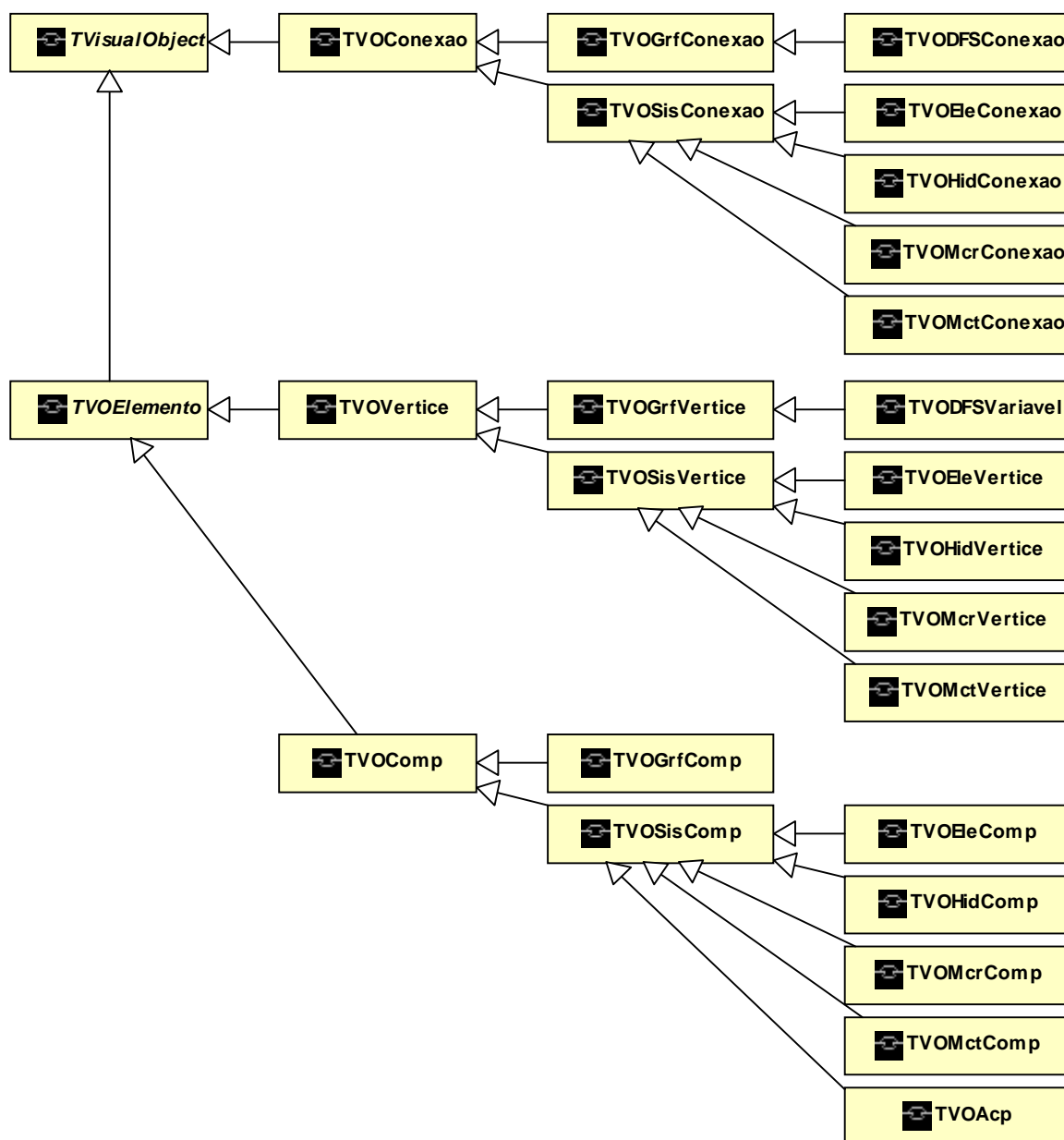


Figura 4.4. Especializações da classe TVisualObject.

Finalmente, os componentes dos diagramas foram especializados até atingir o nível de representação de um elemento físico. Neste nível de especialização, a classe, praticamente, define apenas o ícone gráfico do componente físico; toda sua funcionalidade, atributos e comportamentos, são definidos nas classes ancestrais.

O diagrama da figura 4.5 mostra as especializações da classe Tvoelecomp, que representa os componentes elétricos. Conforme pode ser visto, os dispositivos elétricos que podem ser incluídos em um diagrama são: fontes de tensão e de corrente, resistores, indutores, capacitores, transformadores e fontes controladas (ou moduladas) de tensão e corrente.

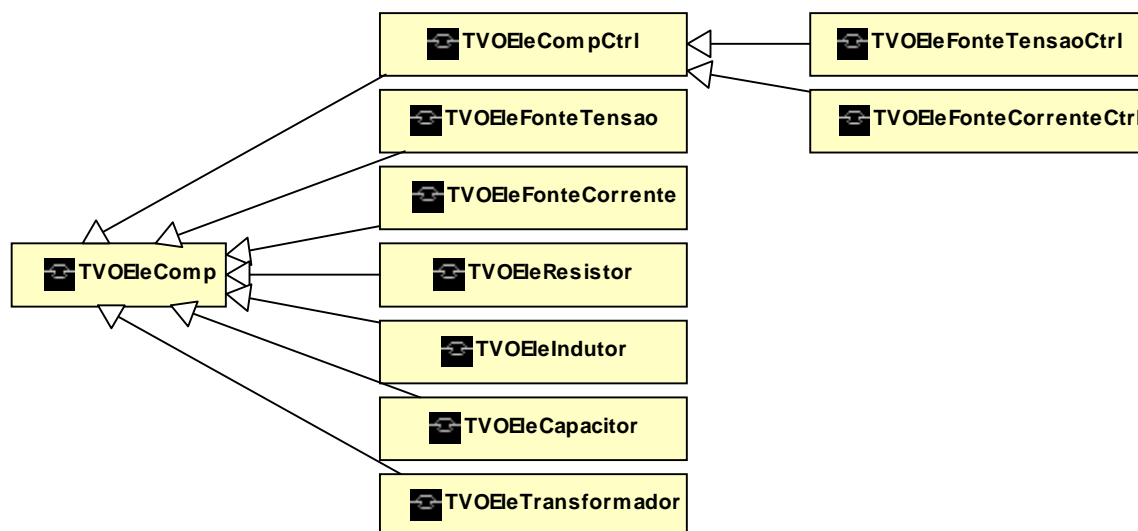


Figura 4.5. Especializações da classe TVOEleComp.

O diagrama da figura 4.6 mostra as especializações da classe TVOHidComp, representando os componentes hidráulicos (ou fluidos), dentre os quais: fontes de pressão e de vazão, tubulações com resistência, tubulações com indutância, reservatórios, transformadores, referências e fontes controladas.

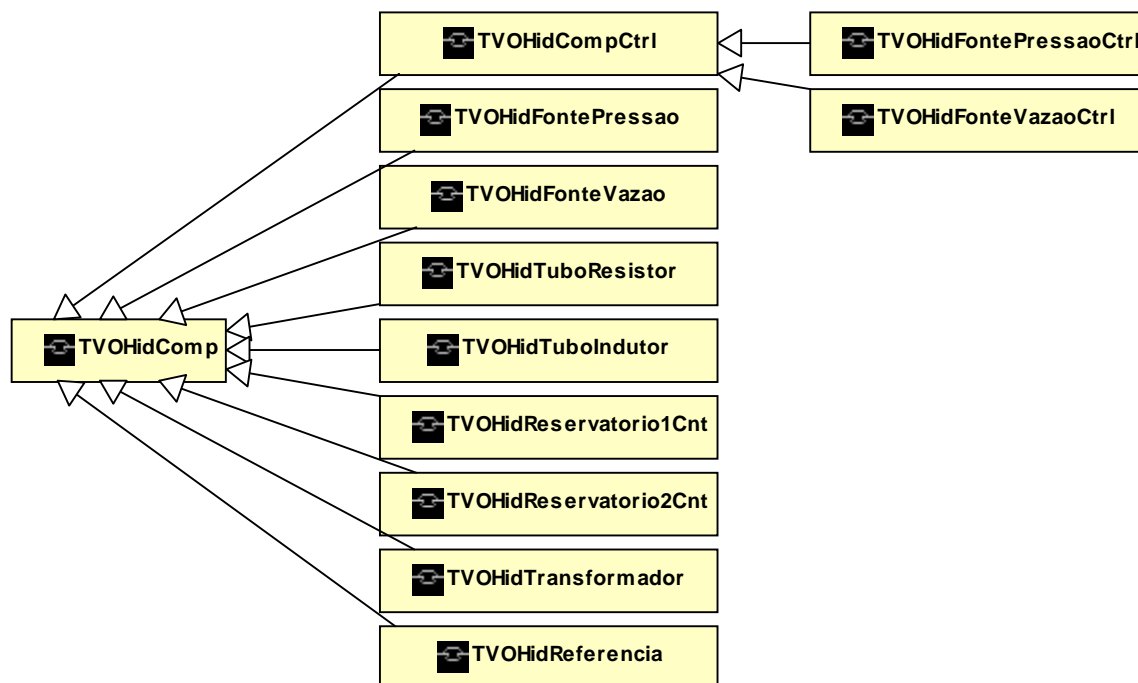


Figura 4.6. Especializações da classe TVOHidComp.

O diagrama da figura 4.7 mostra as especializações da classe Tvomctcomp, representando os componentes mecânicos translacionais, dentre os quais: fontes de velocidade e de força, amortecedores, molas, massa, transformadores, referências e fontes controladas.

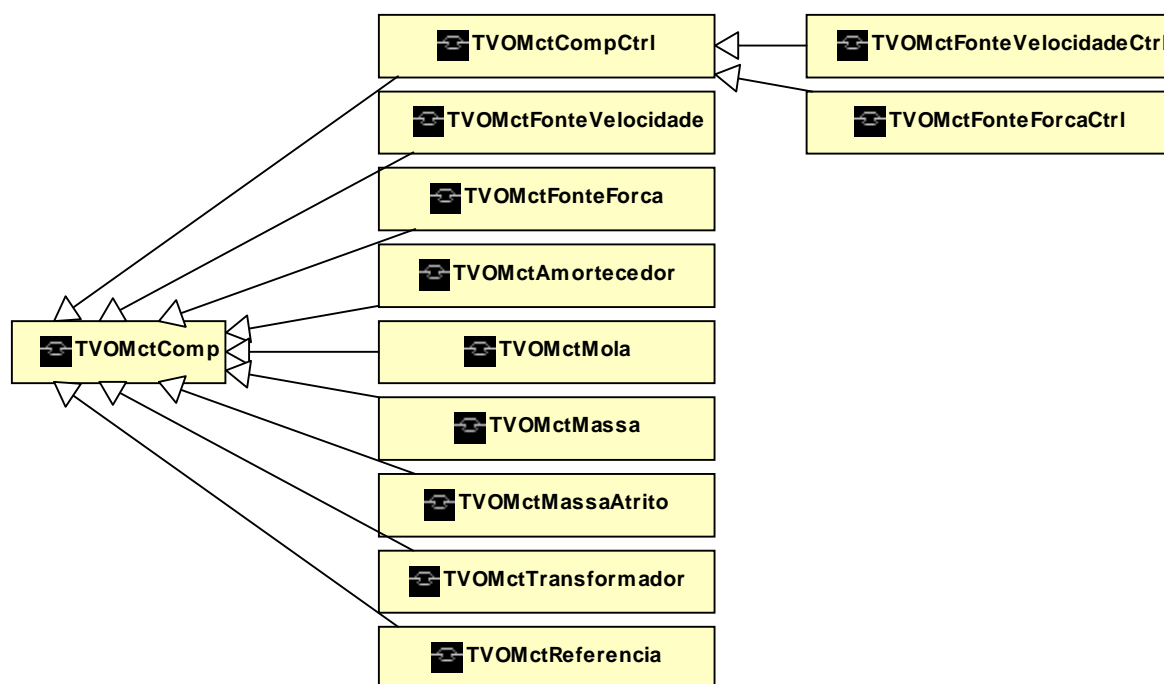


Figura 4.7. Especializações da classe TVOMctComp.

Os componentes do sistema mecânico rotacional são praticamente idênticos aos componentes do mecânico translacional, excetuando-se as fontes de energia que são representadas por fontes de velocidade angular e de torque.

O diagrama da figura 4.8 mostra as especializações da classe TVOacp, que representa os acopladores utilizados para interligar um domínio físico com outro.

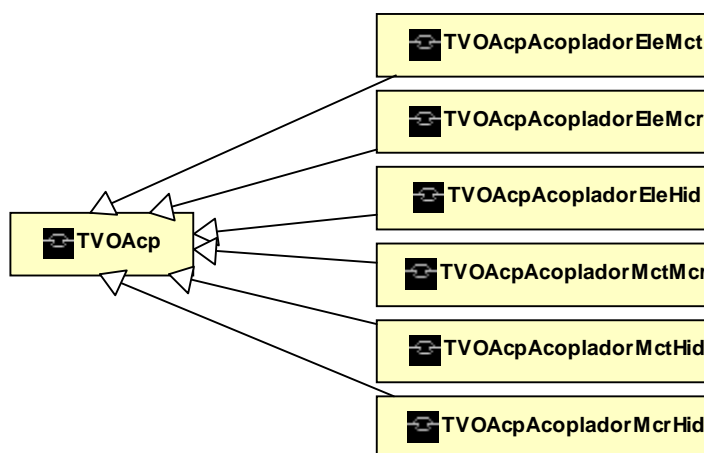


Figura 4.8. Especializações da classe TVOacp.

O modelo adotado definiu a utilização de acopladores genéricos, ou seja, que possam representar qualquer dispositivo físico capaz de interligar dois componentes de domínios físicos distintos. Desta forma, os acopladores mapeiam o

funcionamento de dispositivos como motores, geradores, virabrequins, entre outros. Como mostra o diagrama, foram definidos componentes para todas as combinações possíveis de acoplamento: elétrico – mecânico translacional, elétrico - mecânico rotacional, elétrico - hidráulico, mecânico translacional - rotacional, mecânico translacional - hidráulico e mecânico rotacional - hidráulico.

É importante perceber que a tarefa de inclusão de novos componentes físicos ao sistema computacional tornou-se uma tarefa bastante simples devido ao modelo desenvolvido. Conforme citado, no último nível de especialização basicamente é necessário definir o ícone do elemento físico. Evidentemente, que a inclusão de novos componentes está restringida à metodologia de modelagem de sistemas físicos utilizada pelo ambiente computacional, a qual mapeia os componentes como elementos generalizados em função de suas propriedades constitutivas associadas as variáveis de esforço e fluxo, comentadas no Capítulo 2.

Finalmente, o diagrama da figura 4.9 mostra as especializações da classe *Tvogrcomp*, que representa os ícones gráficos constituintes dos grafos de ligação e dos diagramas de fluxo de sinal.

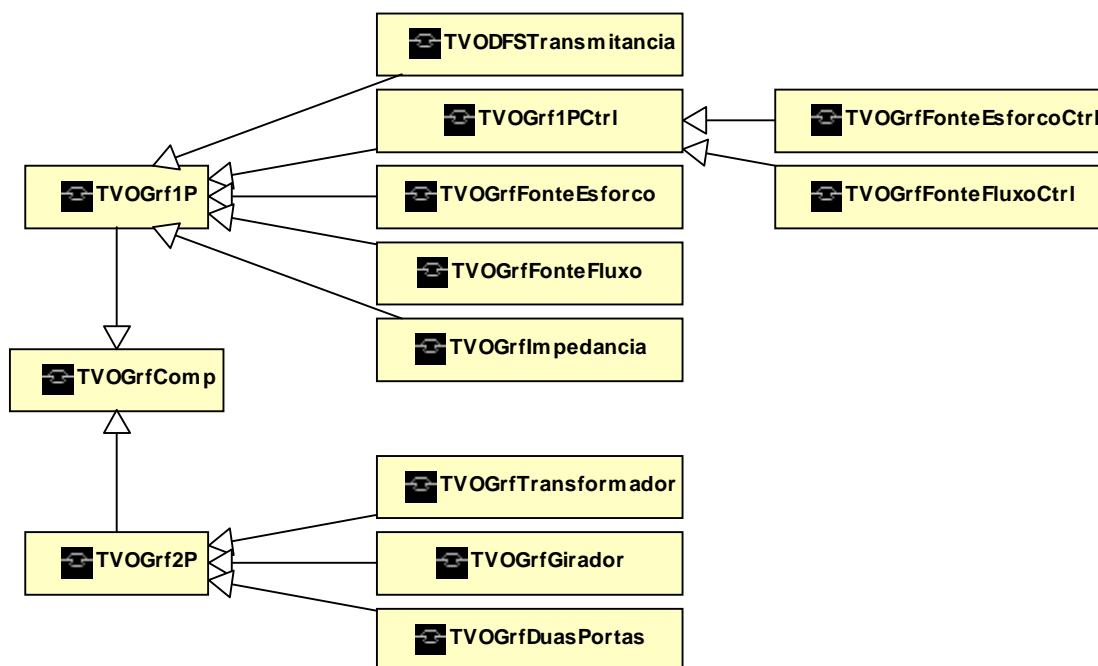


Figura 4.9. Especializações da classe *TVOGrfComp*.

Como mostra o diagrama, a classe *Tvogrcomp* é primeiramente

especializada para representar os elementos generalizados com uma ou duas portas de energia. Em seguida, são especializadas até o nível de elemento generalizado: fonte de esforço ou fluxo, impedância, transformador, girador, elementos genéricos de duas portas e fontes controladas de esforço e fluxo. Todos esses elementos representam, na verdade, ramos de um grafo. Elementos de uma porta representam um ramo, interligando dois vértices. Este comportamento é exatamente o mesmo requerido para as transmitâncias em um DFS. Por este motivo, a classe `Tvodefstransmitancia` aparece como uma especialização da classe `Tvogrf1p`, que representa os elementos de uma porta.

Apesar de não apresentar detalhes de implementação da interface gráfica, os diagramas de classe apresentados dão uma idéia geral do seu funcionamento e de sua organização lógica. O ambiente computacional possui ainda outras classes relacionadas à interface de modelagem. Essas classes estão associadas às janelas de propriedades, vistas no próximo capítulo, as quais permitem aos usuários alterar as propriedades dos objetos gráficos, como ganho e descrição, por exemplo.

4.5.2. Estruturas de Dados

O segundo grupo de classes que compõe o ambiente computacional está relacionado à definição das estruturas de dados utilizadas para representar os diagramas gráficos na memória e as funções de transferência nas formas simbólica e numérica. O grupo inclui ainda, as estruturas requeridas pelos principais algoritmos, em particular, a obtenção do DFS do sistema e a regra de Mason.

Ao contrário dos diagramas da seção anterior, que mostravam basicamente apenas o nome das classes, os diagramas de classe vistos nesta seção mostram os atributos e métodos indispensáveis ao entendimento de cada estrutura e dos algoritmos apresentados no Capítulo 6.

4.5.2.1. Estrutura de Dados do Desenho Gráfico

A primeira estrutura de dados modelada é utilizada para mapear o desenho gráfico do sistema físico e implementar o algoritmo de obtenção do seu grafo de ligação. O diagrama da figura 4.10 mostra as classes que compõem essa estrutura.

A classe TSistema representa o desenho gráfico do sistema físico como um todo. Ela mantém uma lista de elementos, representando os vértices e os componentes do diagrama gráfico, e uma lista de conexões, representando as interligações entre os elementos. As listas de elementos e conexões são instâncias das classes TListElemento e TListConexao, respectivamente.

A classe TElemento representa um elemento do desenho gráfico. Cada elemento é identificado por um ID, que é diferente para todos elementos e conexões do sistema. A classe possui uma auto-associação que mostra os elementos interligados a cada um dos seus conectores.

Finalmente, a classe TConexao representa uma conexão entre dois elementos do sistema, possuindo portanto duas associações com a classe TElemento, conforme visto no diagrama.

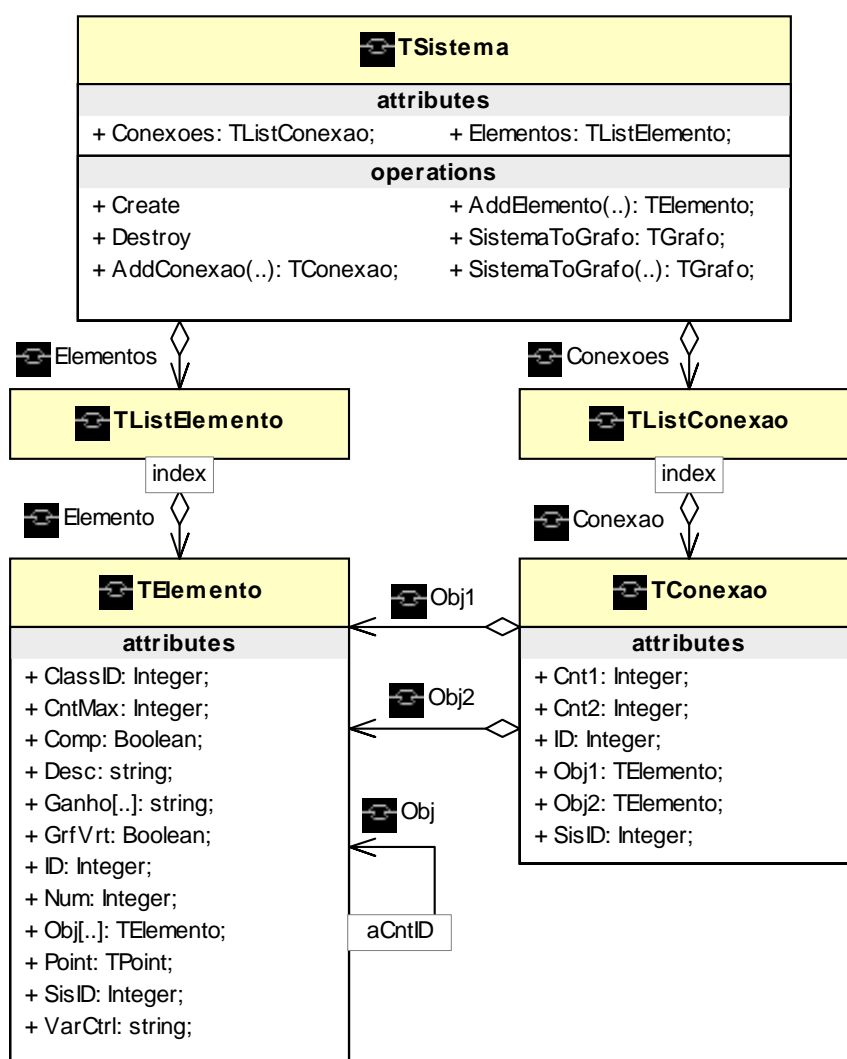


Figura 4.10. Estrutura de dados do desenho gráfico.

A tabela 4.1 descreve a funcionalidade dos métodos da classe TSistema. Conforme visto no diagrama, apenas os métodos públicos da classe foram listados.

Tabela 4.1. Métodos da classe TSistema.

Método	Descrição
AddConexao	Adiciona uma conexão no sistema.
AddElemento	Adiciona um elemento no sistema.
SistemaToGrafo	Executa o algoritmo de obtenção do grafo do sistema.

As tabelas 4.2 e 4.3 descrevem respectivamente os principais atributos das classes TElemento e TConexao.

Tabela 4.2. Atributos da classe TElemento.

Atributo	Descrição
ClassID	Identificador da classe do objeto. Ex: resistor, amortecedor, etc.
CntMax	Número de conectores do objeto.
Comp	Flag para componente ou vértice.
Desc	Nome do objeto. Ex: resistor1, resistor2, etc.
Ganho	Vetor de ganhos.
GrfVrt	Flag que identifica se o vértice no sistema também é vértice no grafo.
Num	ID do vértice no grafo.
Obj	Vetor de objetos conectados ao elemento.
Point	Posição gráfica do elemento no diagrama.
SisID	Identificador do domínio físico.
VarCtrl	Variável de controle de fontes moduladas.

Tabela 4.3. Atributos da classe TConexao.

Atributo	Descrição
Cnt1	Identificador do conector usado no primeiro objeto da conexão.
Cnt2	Identificador do conector usado no segundo objeto da conexão.
ID	Identificador da conexão.
Obj1	Primeiro objeto da conexão.
Obj2	Segundo objeto da conexão.

SisID	Identificador do domínio físico.
-------	----------------------------------

4.5.2.2. Estrutura de Dados do Grafo de Ligação e DFS

A segunda estrutura de dados modelada é utilizada para mapear tanto os grafos de ligação quanto os diagramas de fluxo de sinal, uma vez que ambos os diagramas são dígrafos. A estrutura também é utilizada pelos algoritmos de obtenção do DFS de um grafo de ligação e pela regra de Mason.

O diagrama da figura 4.11 mostra as classes que compõem essa estrutura. A classe TGrafo representa tanto um grafo não orientado, quanto um dígrafo, conforme indicado pelo atributo Direcional. Um objeto da classe TGrafo mantém três listas: uma lista de vértices, uma de ramos e outra de símbolos.

A lista de vértices, que é um objeto da classe TListVertice, representa todos os vértices do grafo; a lista de ramos, que é uma instância da classe TListRamo, mapeia os seus ramos; e a lista de símbolos, que é um objeto da classe TListSimboloVal, lista todos os símbolos que deverão aparecer na função de transferência simbólica do sistema físico, assim como, os valores numéricos atribuídos a cada um.

A classe TVertice representa um vértice do grafo. Cada vértice, que é identificado por um ID único, mantém uma lista de vértices adjacentes, relacionando todos os vértices do grafo adjacentes a ele. As listas de adjacências são objetos da classe TListAdjacente e cada um dos itens da lista é um objeto da classe TAdjacente. Apesar de redundante, a informação armazenada na lista de adjacentes facilita a execução de alguns algoritmos implementados no software, em especial, os algoritmos de busca de caminho e conexidade, os quais utilizam bastante o conceito de lista de adjacência.

A classe TRamo representa um ramo do grafo. Cada ramo, que também possui um ID único, está associado a dois vértices, seus vértices terminais, e a uma instância da classe TGanho, a qual representa o seu ganho. Os ganhos são compostos pelo produto entre uma constante, a variável de Laplace elevada a um expoente e um símbolo elevado a um expoente.

Finalmente, a classe TListGrafo mantém uma lista de grafo, sendo utilizada pelo algoritmo que encontra as componentes conexas de um grafo não conexo. Cada item de uma lista de grafos, que é uma instância da classe TGrafo, representa um subgrafo conexo do grafo original.

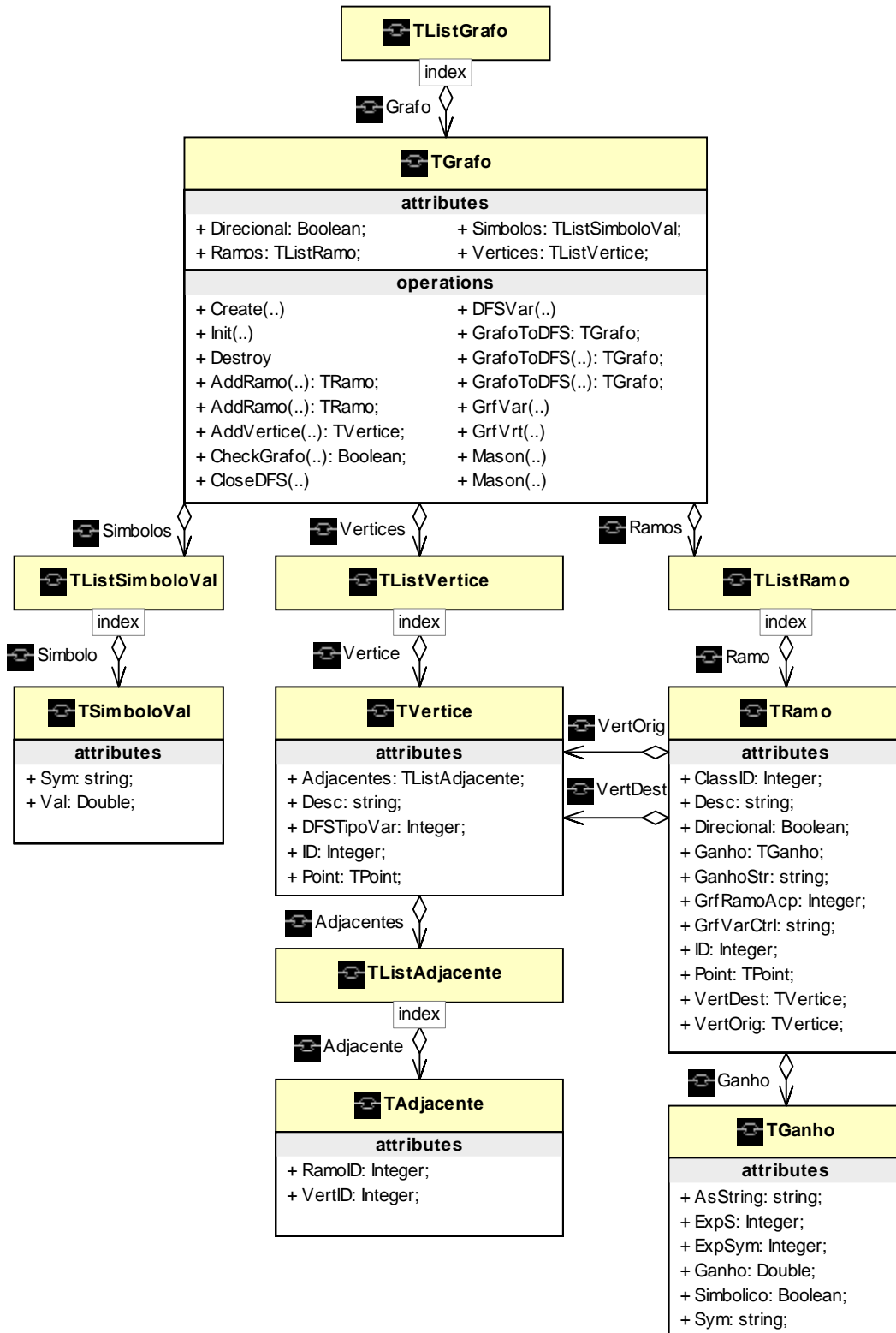


Figura 4.11. Estrutura de dados dos grafos de ligação e diagramas de fluxo de sinal.

A tabela 4.4 descreve os principais métodos da classe TGrafo.

Tabela 4.4. Métodos da classe TGrafo.

Método	Descrição
AddRamo	Adiciona um ramo no grafo.
AddVertice	Adiciona um vértice no grafo.
CheckGrafo	Verifica a estrutura do grafo para execução do algoritmo que obtém o DFS a partir do grafo de ligação.
CloseDFS	Fecha o DFS para execução da regra de Mason.
GrafoToDFS	Executa o algoritmo que obtém o DFS a partir do grafo de ligação.
Mason	Executa a regra de Mason.

As tabelas 4.5, 4.6 e 4.7 descrevem os principais atributos das classes TVertice, TRamo e TGainho.

Tabela 4.5. Atributos da classe TVertice.

Atributo	Descrição
Desc	Nome do vértice.
DFSTipoVar	Tipo da variável no diagrama de fluxo de sinal: entrada ou saída.
Point	Posição do vértice no diagrama gráfico.

Tabela 4.6. Atributos da classe TRamo.

Atributo	Descrição
ClassID	Identificador da classe do elemento generalizado.
Desc	Nome associado ao ramo.
Ganho	Ganho do ramo.
GrfRamoAcp	Identificador do ramo acoplado em elementos de duas portas.
GrfVarCtrl	Variável de controle de fontes moduladas.
VertDest	Vértice terminal (final) do ramo.
VertOrig	Vértice terminal (inicial) do ramo.

Tabela 4.7. Atributos da classe TGainho.

Atributo	Descrição
ExpS	Expoente da variável de Laplace.

ExpSym	Expoente do símbolo.
Ganho	Ganho numérico.
Sym	Símbolo do ganho.

4.5.2.3. Estrutura de Dados do Algoritmo Grafo de Ligação – DFS

A terceira estrutura de dados modelada é utilizada pelo algoritmo que obtém o DFS de um sistema físico a partir do seu grafo de ligação. Basicamente, são necessárias apenas duas estruturas adicionais, visto que a maior parte do processamento do algoritmo é realizada utilizando a própria estrutura do grafo.

O diagrama da figura 4.12 mostra as classes que compõem a estrutura. A classe TMatrizStr define uma matriz dinâmica de valores simbólicos, sendo utilizada para representar as matrizes de equações obtidas a partir do grafo de ligação do sistema físico. As classes TArvore, TListNo e TNo implementam uma estrutura de árvore que é utilizada no algoritmo de busca necessário à obtenção do DFS.

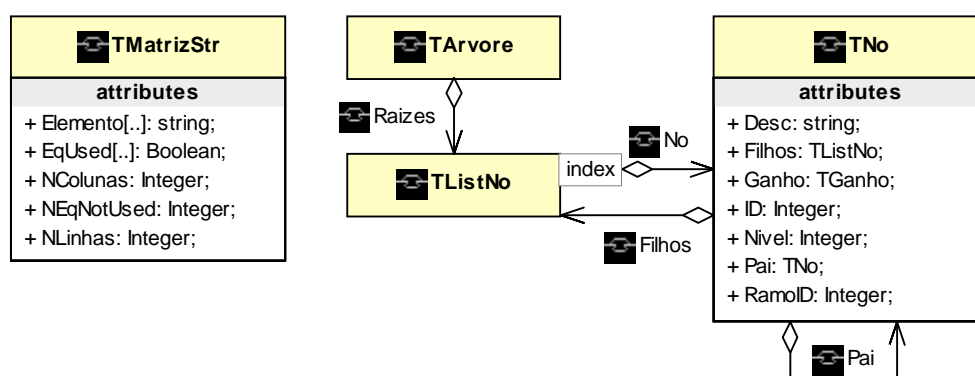


Figura 4.12. Estrutura de dados do algoritmo grafo de ligação – diagrama de fluxo de sinal.

4.5.2.4. Estrutura de Dados da Regra de Mason

A quarta estrutura de dados modelada é utilizada pela regra de Mason para o cálculo da função de transferência do sistema físico.

O diagrama da figura 4.13 mostra as classes que compõem a estrutura. A classe TLaco representa um laço do DFS. Um objeto desta classe mantém três listas: as listas de vértices e ramos do grafo que originam o laço e a lista de símbolos que compõe o seu ganho. As listas de vértices e ramos, dadas pelos atributos ListV e ListR respectivamente, são objetos da classe TListInt, que representa uma lista de inteiros e armazena apenas os IDs dos elementos. O ganho

do laço é obtido a partir de uma constante, da variável de Laplace elevada a um expoente e de uma lista de símbolos elevados a seus respectivos expoentes, os quais estão associados aos atributos: Ganho, ExpS e ListS, respectivamente. A lista de símbolos é uma instância da classe TListSimboloExp e cada item desta lista é um objeto da classe TSimboloExp.

A classe TListListInt é utilizada para representar a lista de laços disjuntos do grafo, isto é, os laços que não possuem qualquer vértice em comum. Um objeto desta classe mantém uma lista de objetos da classe TListInt, ou seja, cada objeto manipula uma lista de listas de inteiros.

A classe TLacoOS representa um laço de ordem superior da regra de Mason, que é formado por uma lista de laços do grafo. O atributo ListL da classe representa a lista com o ID dos laços que compõe o laço de ordem superior. Cada objeto da classe TLacoOS possui também um ganho, um expoente da variável S e uma lista de símbolos elevados a expoentes, dados pelos atributos Ganho, ExpS e ListS.

A classe TListLacoOS mantém uma lista com todos os laços de superior de uma determinada ordem. Finalmente, a classe TListListLacoOS mantém uma lista com as lista de laços de ordem superior de todas as ordens. Objetos desta classe são usados para calcular a função de transferência do sistema físico.

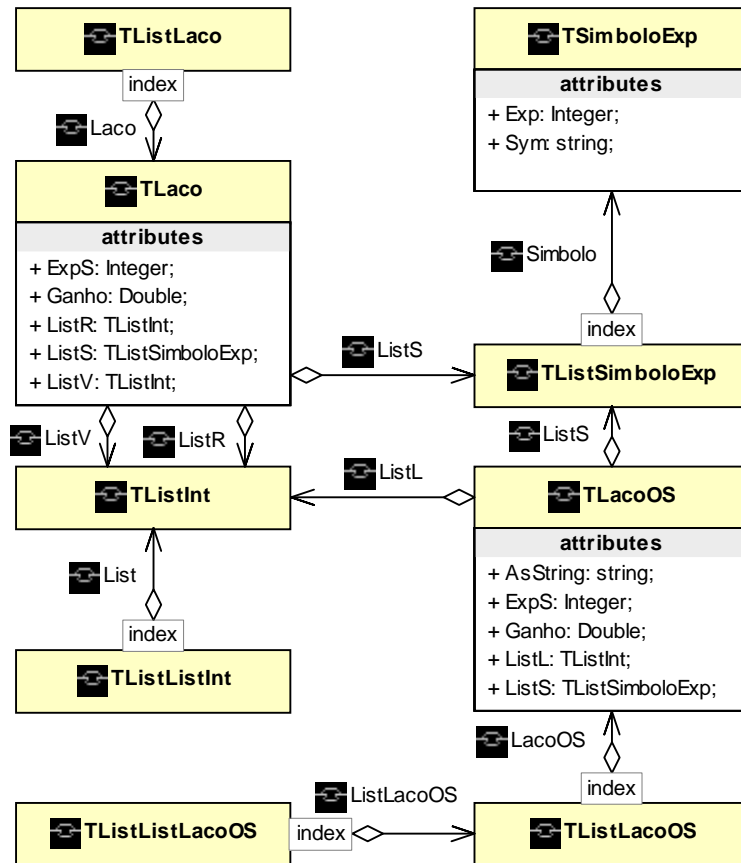


Figura 4.13. Estrutura de dados da regra de Mason.

4.5.2.5. Estrutura de Dados da Função de Transferência

A última estrutura de dados modelada de maior relevância ao ambiente computacional é utilizada para representar uma função de transferência nas formas simbólica e numérica. As classes da estrutura implementam algoritmos para calcular a FT simbólica a partir das listas de laços de ordem superior, a FT numérica a partir dos valores atribuídos aos símbolos e a função de sensibilidade paramétrica.

O diagrama da figura 4.14 mostra as classes que compõem a estrutura. A classe TFuncao representa uma função de transferência. Um objeto desta classe mantém duas listas de fatores: uma para o numerador e outra para o denominador da função. As listas são objetos da classe TListFator. Cada fator da lista, que é um objeto da classe TFator, possui também um ganho numérico, um expoente em S e uma lista de símbolos elevados a expoentes, dados por Ganho, ExpS e ListS.

Quando a função numérica é calculada, cada uma das listas de fatores passa a ser constituída por um ganho numérico e por um vetor de coeficientes de um

polinômio em S, dados pelos atributos FGanho e FGExpS, respectivamente.

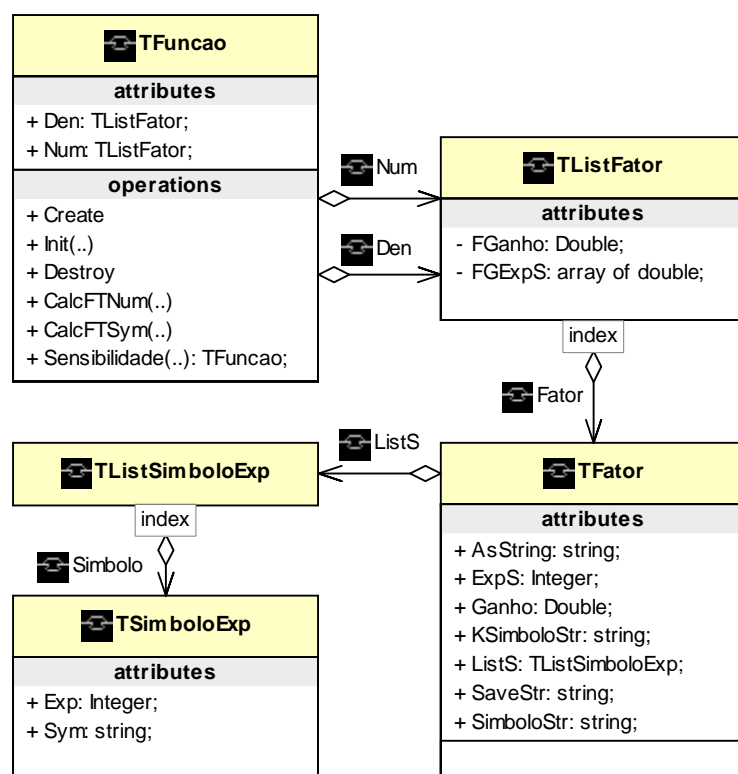


Figura 4.14. Estrutura de dados da função de transferência.

4.6. Diagramas de Colaborações

Uma vez definidas as classes que compõem o software, os diagramas de colaboração permitem visualizar os vínculos estabelecidos entre os objetos do sistema e as mensagens trocadas entre eles, objetivando a realização dos casos de uso identificados para o ambiente computacional.

4.6.1. Obtenção do Grafo de Ligação do Sistema

O diagrama da figura 4.15 mostra as colaborações estabelecidas para a obtenção do grafo de ligação de um sistema físico a partir do seu desenho gráfico, relacionadas ao caso de uso Obter o Grafo de Ligação.

Após a definição do desenho gráfico pelo ator, o objeto VFSistema que representa este diagrama, chama os métodos AddElemento e AddConexão do objeto Sistema para definir sua estrutura de dados. O método SistemaToGrafo, executado em seguida, implementa o algoritmo que encontra o grafo de ligação do

sistema. Os métodos `AddVertice` e `AddRamo` do objeto `Grafo` são chamados para definir a estrutura do grafo definido pelo algoritmo. A estrutura de dados do grafo é então retorna e, em seguida, passada como parâmetro do método `InitVFGrafo` do objeto `VFGrafo`. O objeto `VFGrafo` constrói então o diagrama gráfico do grafo, a partir de sua estrutura de dados, apresentando-o ao ator.

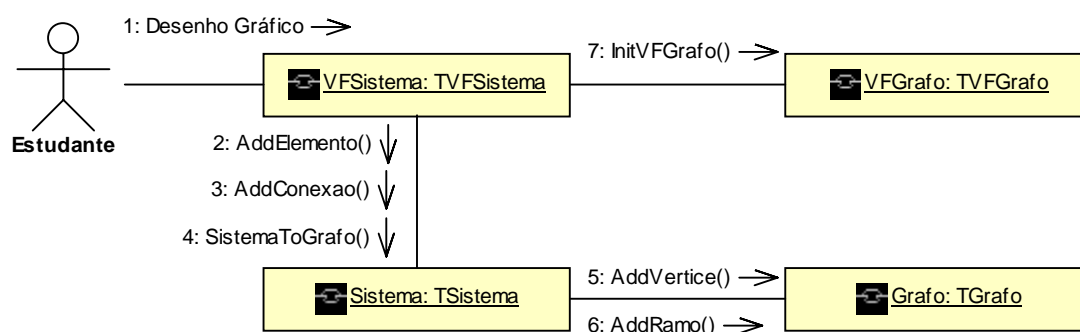


Figura 4.15. Diagrama de colaboração referente ao caso de uso Obter o Grafo de Ligação.

4.6.2. Cálculo da Função de Transferência do Sistema

O diagrama da figura 4.16 mostra as colaborações relacionadas ao cálculo da função de transferência simbólica do sistema físico realizado a partir do desenho gráfico do sistema, referindo-se portanto ao caso de uso FT Simbólica do Sistema.

A seqüência inicial da colaboração é idêntica àquela apresenta no exemplo anterior. Após a definição do desenho gráfico pelo ator, o objeto `VFSistema` executa os métodos `AddElemento` e `AddConexão` do objeto `Sistema` para definir sua estrutura de dados e o método `SistemaToGrafo` para obter o grafo de ligação do sistema. Os métodos `AddVertice` e `AddRamo` são executados para definir a estrutura do grafo de ligação que é então retornado para o objeto `VFSistema`.

Em seguida, o objeto `VFSistema` chama o método `CheckGrafo` do grafo de ligação para verificar se a estrutura deste grafo é adequada à execução do algoritmo que obtém o DFS do grafo. Caso positivo, o método `GrafoToDFS`, executado em seguida, implementa este algoritmo. Os métodos `AddVertice` e `AddRamo` do objeto `DFS` são então chamados para definir a estrutura do DFS obtido a partir do grafo de ligação. A estrutura de dados do DFS é então retornada ao objeto `VFSistema`.

No passo seguinte, o método CloseDFS é executado para obter o DFS fechado deste grafo. Em seguida, o método Mason é executado para iniciar o processo de cálculo da FT simbólica do sistema. O método retorna as listas de laços de ordem superior que é passada como parâmetro pelo método CalcFTSym do objeto Funcao. A FT é então calculada por este objeto e apresentada ao ator.

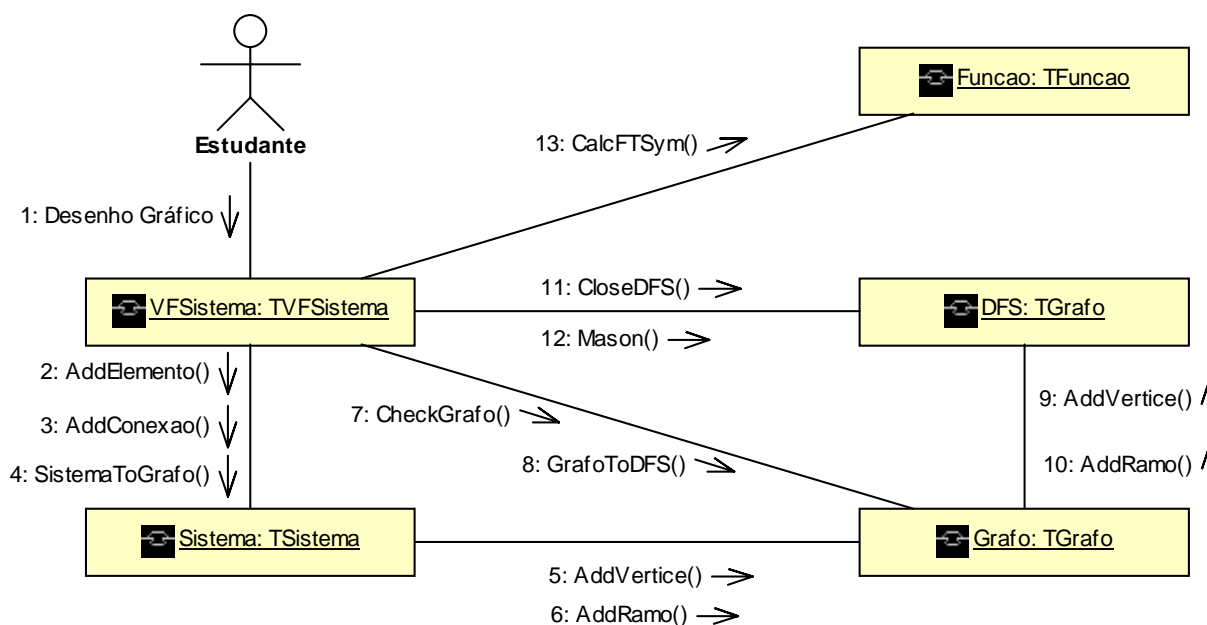


Figura 4.16. Diagrama de colaboração referente ao caso de uso FT Simbólica do Sistema.

4.6.3. Obtenção do DFS a partir do Grafo de Ligação

O diagrama da figura 4.17 mostra as colaborações estabelecidas para a obtenção do DFS de um sistema físico a partir do seu grafo de ligação, relacionadas portanto ao caso de uso Obter o DFS.

Após o ator definir o modelo do sistema através do grafo de ligação, o objeto VFGrafo chama os métodos AddVertice e AddRamo para montar a estrutura de dados do grafo. O método CheckGrafo, chamado em seguida, verifica a adequação da estrutura do grafo para a execução do método GrafoToDFS que obtém o DFS do grafo. Então, os métodos AddVertice e AddRamo do objeto DFS são chamados para definir a sua estrutura que é retornada e passada como parâmetro pelo método InitVFDFS do objeto VFDFS. O objeto VFGrafo constrói o diagrama gráfico do DFS, a partir de sua estrutura de dados, apresentando-o ao ator.

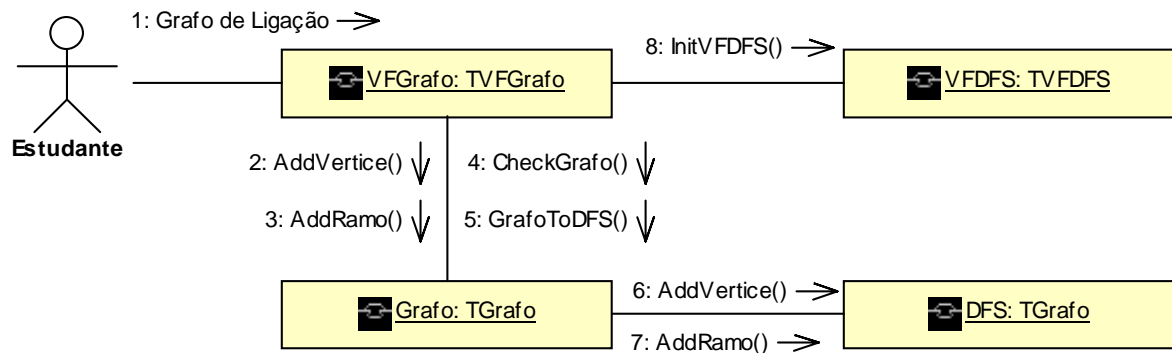


Figura 4.17. Diagrama de colaboração relacionado ao caso de uso Obter o DFS.

4.6.4. Cálculo da Função de Transferência do Grafo de Ligação

O diagrama da figura 4.18 mostra as colaborações relacionadas ao cálculo da função de transferência realizado a partir do grafo de ligação do sistema, referindo-se portanto ao caso de uso FT Simbólica do Grafo.

A parte inicial da colaboração é idêntica a apresentada no exemplo anterior e a parte final, a apresentada no diagrama da figura 4.16. Após a definição do grafo de ligação pelo ator, a sua estrutura de dados é montada e, em seguida, o método GrafoToDFS calcula o DFS do grafo. O DFS é então fechado e a regra de Mason é executada, retornando as listas de laços de ordem superior que é passada ao objeto Funcao que finaliza o cálculo da função de transferência simbólica.

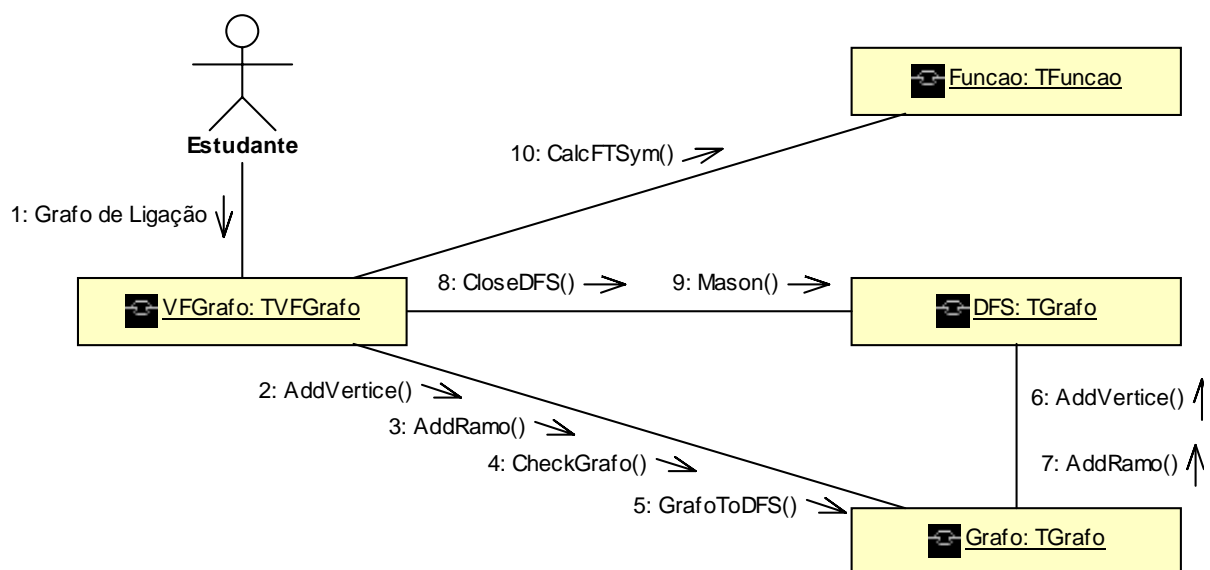


Figura 4.18. Diagrama de colaboração relacionado ao caso de uso FT Simbólica do Grafo.

4.6.5. Cálculo da Função de Transferência do DFS

Finalmente, o diagrama da figura 4.19 mostra as colaborações relacionadas ao cálculo da função de transferência realizado a partir de um diagrama de fluxo de sinal, referindo-se portanto ao caso de uso FT Simbólica do DFS.

Após a definição do diagrama gráfico pelo ator, o objeto VFDFS define a estrutura de dados do DFS, através dos métodos AddVertice e AddRamo. Em seguida o DFS é fechado pelo método CloseDFS e a regra de Mason é executada, através do método Mason. De forma análoga aos exemplos anteriores, a lista de laços retornada pelo método Mason é passada para o objeto Função pelo método CalcFTSym que calcula a função de transferência do sistema.

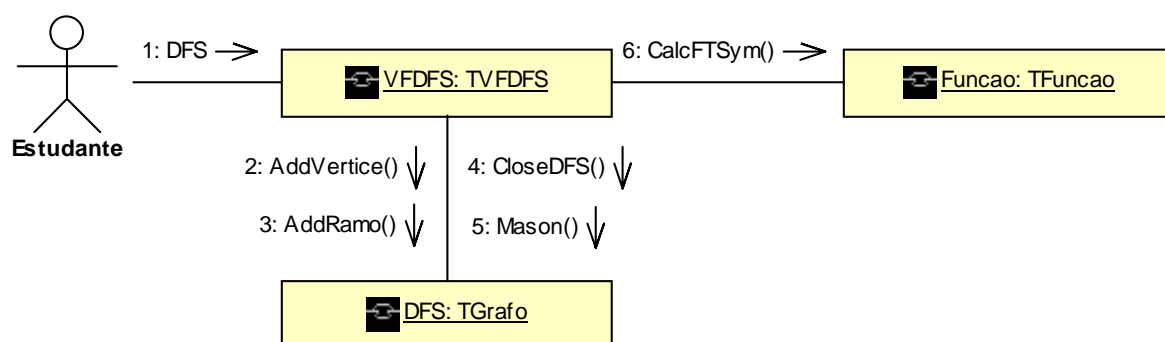


Figura 4.19. Diagrama de colaboração relacionado ao caso de uso FT Simbólica do DFS.

Os demais casos de uso definidos pelo ambiente computacional e relacionados às de estruturas de dados, isto é, os casos de uso FT Numérica, Sensibilidade e Simulação e Projeto, são implementados pelos métodos CalcFTNum, Sensibilidade e Sincon da classe TFuncao, apresentada no diagrama da figura 4.14.

Uma vez definidas as classes e as principais colaborações entre elas, o processo de modelagem do software é direcionado para a etapa de implementação. No Capítulo 5, o ambiente computacional proposto é apresentado, destacando a funcionalidade da interface gráfica de modelagem desenvolvida para a realização das funções atribuídas ao software.

No Capítulo 6, os métodos SistemaToGrafo e GrafoToDFS da Classe TGrafo, apresentados nos diagramas de colaboração desta seção, são detalhados em nível

de algoritmo, apresentando todos os passos necessários à obtenção do grafo de ligação e do diagrama de fluxo de sinal de um sistema físico modelado a partir de seu desenho gráfico.

Finalmente, no Capítulo 7 são apresentados alguns resultados obtidos pelo ambiente computacional proposto.

5. Ambiente Computacional Proposto

5.1. Introdução

Este capítulo apresenta o ambiente computacional proposto, destacando suas principais características, mostrando os recursos disponíveis para a modelagem de sistemas físicos lineares e delimitando o seu universo de aplicação. O capítulo mostra também detalhes da implementação do software que não foram apresentados no capítulo anterior, como por exemplo, a interface que apresenta os resultados produzidos pelos algoritmos.

5.2. Interface Gráfica de Modelagem

Como citado anteriormente, o ambiente computacional proposto foi denominado de *ModSym*, que é uma “abreviação” para Modelagem Simbólica de Sistemas Físicos. O *ModSym* tem como finalidade auxiliar os estudantes da Engenharia de Controle na realização da atividade de modelagem de sistemas físicos lineares, a qual é fundamental ao estudo de sistemas de controle.

5.2.1. Objetivo

Os objetivos principais do software já foram amplamente discutidos nos capítulos anteriores. Mas, em resumo, a idéia do software é permitir que estudantes modelem no computador sistemas físicos a partir de diagramas gráficos e obtenham modelos matemáticos que descrevam seu comportamento; em particular, funções de transferência nas formas simbólica e numérica e funções de sensibilidade paramétrica.

A interface gráfica de modelagem, implementada no software, facilita a realização da tarefa de modelagem de sistemas físicos e permite a construção dos diagramas gráficos que representam um sistema utilizando basicamente o *mouse*. Didaticamente, a interface gráfica pode ser dividida em três partes: a área de montagem de desenhos gráficos, a área de montagem de grafos de ligação e a área de montagem de diagramas de fluxo de sinal. Evidentemente, cada área de montagem está relacionada a um tipo de diagrama gráfico suportado pelo software.

5.2.2. Área de Montagem de Desenhos Gráficos

A área de montagem de desenhos gráficos permite a modelagem de sistemas a partir de um conjunto de elementos físicos dos domínios elétrico, mecânico translacional, mecânico rotacional e hidráulico, além de acopladores utilizados para interligar elementos de dois domínios físicos diferentes.

Os diagramas gráficos realizados nesta área de montagem assemelham-se bastante aos desenhos de sistemas físicos realizados à mão. Entretanto, algumas convenções necessárias à sistematização do processo de modelagem, apresentadas no decorrer do capítulo, acarretaram em pequenas diferenças na forma habitual de representação do desenho gráfico do sistema físico. As alterações, entretanto, não são relevantes e são absorvidas sem muito esforço.

A figura 5.1 apresenta a interface gráfica de modelagem do *ModSym*. A janela com o título “ModSym – Modelagem Simbólica de Sistemas Físicos” é a janela principal do aplicativo. Esta janela é composta pelo menu de comandos, que dá acesso às diversas funções do software; pela barra de ferramentas, localizada abaixo do menu, que permite acessar rapidamente suas principais funções; e pela paleta de componentes, que apresenta os ícones gráficos disponíveis à modelagem dos sistemas físicos referentes ao tipo de diagrama selecionado. Conforme pode ser visto, os elementos disponíveis para a montagem de desenhos gráficos estão agrupados de acordo com seus respectivos domínios físicos.

A janela “Sistema1”, vista abaixo da janela principal, representa a área de montagem de um diagrama de desenho gráfico, onde os elementos podem ser

inseridos e conectados de forma a definir a estrutura de um diagrama em particular. Como citado anteriormente, o ambiente computacional gerencia uma lista de diagramas, permitindo que vários deles sejam montados simultaneamente.

Finalmente, a janela “Propriedades”, vista ao lado do diagrama, mostra as características de um objeto do diagrama em particular ou do próprio diagrama. A janela permite que o usuário altere rápida e facilmente os atributos de um objeto, como, por exemplo, a sua descrição ou o seu ganho, dentre outros.

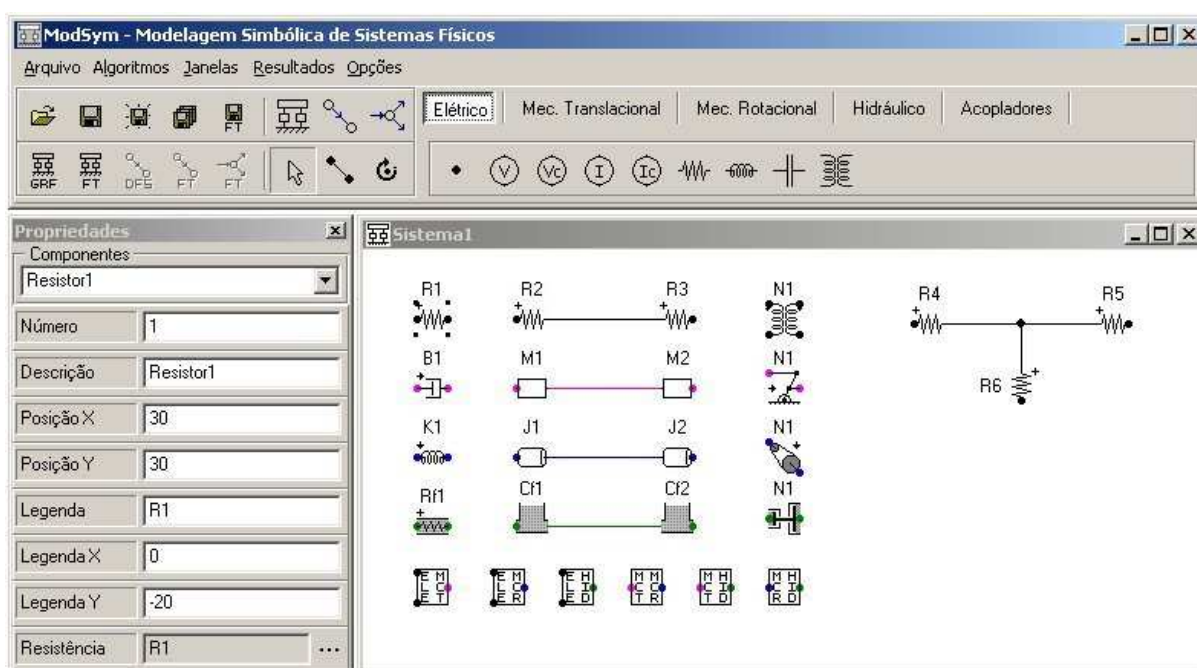


Figura 5.1. Área de montagem de desenhos gráficos.

5.2.2.1. Vértices, Conexões e Conectores

Antes de explorar os elementos dos diversos domínios físicos, é importante entender os conceitos de vértices, conexões e conectores, associados ao desenho dos diagramas.

No diagrama da figura 5.1, por exemplo, foram inseridos alguns elementos de diferentes sistemas físicos. Todos eles apresentam pequenos círculos sólidos em cores distintas representando os seus conectores. É a partir desses conectores que as interligações entre os elementos de um diagrama, denominadas de conexões, são estabelecidas. As conexões são representadas por segmentos de reta interligando um conector de um objeto a um conector de outro objeto. Quando uma

conexão é estabelecida, os conectores interligados por ela são ocultados, conforme mostra a interligação entre os resistores R2 e R3. Os conectores tornam-se novamente visíveis se a conexão for removida.

As cores dos conectores estão associadas aos domínios físicos em estudo e restringem a realização de conexões entre os elementos. Em síntese, apenas conectores de cores iguais podem ser interligados. As conexões realizadas utilizam também o mesmo padrão de cor.

Alguns conectores apresentam ainda um pequeno símbolo “+” ou “→”, indicando que ele é considerado o conector positivo do elemento. Isto significa que a variável generalizada de esforço, associada ao domínio físico do componente, é medida convencionando este conector como referencial positivo. Nos elementos que não apresentam este referencial, normalmente a variável de esforço é medida entre o próprio elemento e a referência. A velocidade das massas e a pressão dos reservatórios são exemplos destes casos.

Todas as conexões realizadas entre elementos dos vários domínios físicos são consideradas ideais. Isto significa que a variável generalizada de esforço assume valor nulo entre os dois conectores interligados pela conexão. Desta forma, pode-se afirmar que não existe diferença de tensão entre os resistores R2 e R3, vistos no diagrama da figura 5.1, considerando evidentemente os conectores interligados. Esta convenção tem uma implicação importante para os sistemas mecânicos e hidráulicos: as massas M1 e M2, vistas no mesmo exemplo, estão submetidas à mesma velocidade linear; as inércias J1 e J2, à mesma velocidade angular; e os reservatórios Cf1 e Cf2, à mesma pressão.

Finalmente, todos os domínios físicos possuem um elemento especial denominado de vértice, o qual permite a conexão de vários componentes entre si. Ainda no mesmo exemplo, os resistores R4, R5 e R6 foram interligados com o auxílio de um vértice elétrico. O ícone gráfico dos vértices é idêntico ao ícone dos conectores, simbolizando a possibilidade de realização de conexões. Os vértices também estão associados a um domínio físico e utilizam a mesma padronização de cores dos conectores; mas, diferentemente deles, suportam várias conexões.

5.2.2.2. Modelagem de Sistemas Elétricos

A modelagem de sistemas elétricos pode ser realizada no *ModSym* através de nove elementos. Cada elemento possui um ícone gráfico e é normalmente associado a um ganho simbólico, o qual representa uma grandeza física. Na tabela 5.1 são listados os elementos elétricos disponíveis no software e suas relações constitutivas no domínio de Laplace, vistas na Seção 2.3.4. Aspectos relevantes a alguns desses elementos serão comentados a seguir. É válido lembrar ainda que para os sistemas elétricos, as variáveis generalizadas de esforço, indicadas por e , e de fluxo, indicadas por f , são respectivamente a tensão e a corrente elétrica. No software, tensões e correntes são simbolizadas pelas letras v e i em minúsculo, respectivamente.

As fontes de tensão disponíveis no sistema são fontes ideais e fornecem uma quantidade específica de esforço, dada pela sua relação constitutiva, independente da corrente através dela. De forma análoga, as fontes de corrente são ideais e suprem uma determinada corrente, independente da tensão entre seus terminais.

Tabela 5.1. Elementos elétricos.

Ícone	Elemento	Ganho	Grandeza	Relação constitutiva
•	Vértice elétrico			
	Fonte de tensão	v	Tensão	$e = v$
	F. de tensão controlada	K_v	Ganho de tensão	$e = K_v \cdot p$
	Fonte de corrente	i	Corrente	$f = i$
	F. de corrente controlada	K_i	Ganho de corrente	$f = K_i \cdot p$
	Resistor	R	Resistência	$e = R \cdot f$
	Indutor	L	Indutância	$e = L \cdot S \cdot f$
	Capacitor	C	Capacitância	$e = C^{-1} S^{-1} \cdot f$
	Transformador elétrico	N	Transformação	$e_2 = N^{-1} \cdot e_1, f_2 = N \cdot f_1$

As fontes de tensão controlada representam dispositivos físicos que possuem a tensão elétrica proporcional a alguma variável generalizada de esforço ou fluxo de

outro componente constituinte do sistema físico. A variável que controla esta tensão é representada por p na sua relação constitutiva. O mesmo acontece com as fontes controladas de corrente em relação a corrente elétrica fornecida por ela. As fontes controladas, ou moduladas, são utilizadas para representar genericamente uma série de dispositivos e dão grande poder de modelagem ao sistema computacional. Aplicações destes componentes serão vistas decorrer no trabalho.

Nos transformadores elétricos, a variável generalizada de fluxo na segunda porta de energia (f_2) é dada pelo produto entre a relação de transformação (N) e a variável de fluxo na primeira porta (f_1); a variável de esforço na segunda porta (e_2) é obtida pelo inverso desta relação vezes o esforço na primeira porta (e_1), conforme mostra a equação 2.14. Esta convenção é, na verdade, adotada para os transformadores de todos os domínios físicos.

Finalmente, a figura 5.2 apresenta um sistema elétrico, [3], modelado na área de montagem de desenhos gráficos. A figura à esquerda mostra os elementos elétricos antes de serem interligados. Por conseguinte, todos os conectores são visíveis. Na figura à direita, os elementos aparecem conectados; conseqüentemente, os conectores ficam ocultos e sem a capacidade de realizar novas conexões. Observe ainda que foram necessários dois vértices para realizar todas as conexões necessárias ao circuito.

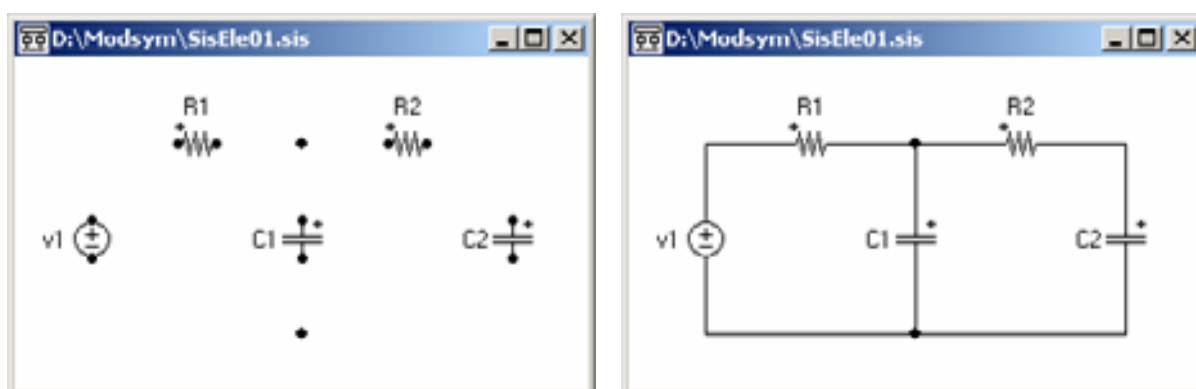


Figura 5.2. Exemplo de sistema elétrico.

O processo de construção dos diagramas é bastante simples. Para incluir os elementos, basta selecionar o ícone desejado na paleta de componentes e clicar no diagrama. Para realizar uma conexão, é só clicar com o botão direito do *mouse* nos





dois conectores que devem ser interligados. O usuário pode ainda movimentar e girar os elementos dentro do diagrama para melhorar sua aparência visual. Além disso, não é necessário se preocupar com o uso indiscriminado de vértices, pois os algoritmos eliminam os vértices supérfluos aos sistemas, quando necessário.

5.2.2.3. Modelagem de Sistemas Mecânicos Translacionais

A modelagem de sistemas mecânicos translacionais pode ser realizada através de onze elementos. A tabela 5.2 mostra os elementos disponíveis e suas relações constitutivas no domínio de Laplace. Aspectos relevantes a alguns desses elementos serão também comentados em seguida. É importante ressaltar que a modelagem dos sistemas mecânicos no ambiente computacional é suportada apenas para as direções horizontal e vertical.

Nos sistemas mecânicos translacionais, as variáveis generalizadas de esforço e de fluxo são respectivamente a velocidade linear e a força. No software, velocidade e força são simbolizadas pelas letras V e F em maiúsculo.

Tabela 5.2. Elementos mecânicos translacionais.

Ícone	Elemento	Ganho	Grandeza	Relação constitutiva
•	Vértice mecânico			
	Referência mecânica			
	Fonte de velocidade	V	Velocidade linear	$e = V$
	Fonte de velocidade controlada	KV	Ganho de velocidade	$e = KV.p$
	Fonte de força	F	Força	$f = F$
	Fonte de força controlada	KF	Ganho de força	$f = KF.p$
	Amortecedor	B	Amortecimento	$e = B^{-1}.f$
	Mola	K	Elasticidade	$e = K^{-1}S.f$
	Massa	M	Massa	$e = M^{-1}S^{-1}.f$
	Massa + Atrito	$M + B$		
	Transformador mecânico	N	Transformação	$e_2 = N^{-1}.e_1, f_2 = N.f_1$

Diferentemente dos sistemas elétricos, onde as fontes de energia definem um referencial, nos sistemas mecânicos o elemento referência mecânica deve ser utilizado com a finalidade de definir a velocidade de referência para os demais elementos do sistema. Por conseguinte, as fontes de energia realizam apenas uma conexão, estabelecendo a velocidade ou a força aplicada a um determinado ponto do sistema mecânico. O outro conector das fontes de energia está implicitamente conectado a referência. O mesmo acontece com o elemento transformador mecânico, onde a variável de esforço em ambas as portas de energia é medida em relação à referência. Desta forma, o transformador realiza apenas duas conexões: uma na primeira porta e outra na segunda.

O elemento Massa + Atrito representa uma massa do sistema mecânico com amortecimento proporcional à sua velocidade. Desta forma, o elemento é equivalente a uma massa e um amortecedor ligados à referência, sendo representado no grafo de ligação por dois elementos generalizados: o primeiro, com relação constitutiva idêntica a da massa, e o segundo, com relação constitutiva idêntica a do amortecedor.

A figura 5.3 mostra um sistema mecânico que modela a suspensão de um automóvel, [32]. O desenho gráfico habitual deste sistema pode ser visto na figura 2.22. Comparando os desenhos, é fácil perceber que existe uma diferença na representação das conexões entre a mola K2, o amortecedor B1 e as massas M1 e M2. De fato, sempre que for necessário conectar mais de um elemento a outro, as conexões terão que ser realizadas através de um vértice, conforme exposto anteriormente. Entretanto, como as conexões são ideais, não existe alteração no comportamento do sistema como um todo.

No exemplo da figura 5.3, portanto, foi necessário incluir dois vértices: um para conectar a mola K2 e o amortecedor B1 à massa M1, e outro para conectar esses mesmos componentes a massa M2. O importante é que ambos os elementos estão submetidos à diferença de velocidade entre as massas, conforme é indicado também no diagrama da figura 2.22. Esta é alteração visual mais relevante referente

Tabela 5.3. Elementos mecânicos rotacionais.

Ícone	Elemento	Ganho	Grandeza	Relação constitutiva
•	Vértice mecânico			
	Referência mecânica			
	Fonte de velocidade	W	Velocidade angular	$e = W$
	Fonte de velocidade controlada	K_w	Ganho de velocidade	$e = K_w \cdot p$
	Fonte de torque	T	Torque	$f = T$
	Fonte de torque controlada	K_t	Ganho de torque	$f = K_t \cdot p$
	Amortecedor	B	Amortecimento	$e = B^{-1} \cdot f$
	Mola	K	Elasticidade	$e = K^{-1} \cdot S \cdot f$
	Inércia	J	Inércia	$e = J^{-1} S^{-1} \cdot f$
	Inércia + Atrito	$J + B$		
	Transformador mecânico	N	Transformação	$e_2 = N^{-1} \cdot e_1, f_2 = N \cdot f_1$

A figura 5.4 mostra um sistema mecânico rotacional, [12], onde um torque é aplicado a duas inércias acopladas através de molas e amortecedores. A inércia J_1 é interligada a inércia J_2 através da mola K_1 e do amortecedor B_1 e inércia J_2 é conectada a referência através da mola K_2 . A fonte de torque aplica um torque T_1 ao sistema que gira em torno do eixo X no sentido horário visto da esquerda para a direita no eixo de rotação.

Percebe-se ainda no exemplo que nenhum vértice foi utilizado para conectar os elementos, apesar de existir implicitamente um "vértice" entre os elementos K_1 e B_1 , definindo uma velocidade angular neste ponto.

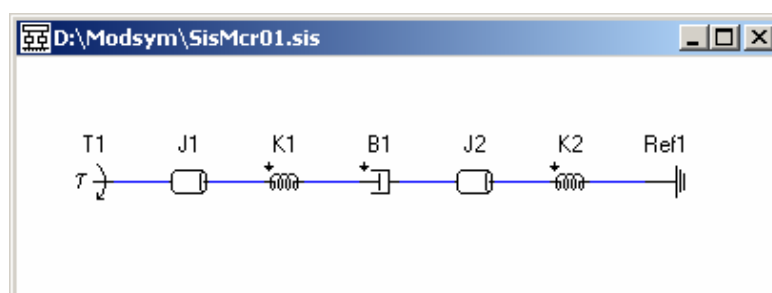












Figura 5.4. Exemplo de sistema mecânico rotacional.

5.2.2.5. Modelagem de Sistemas Hidráulicos

A tabela 5.4 mostra os elementos hidráulicos ou fluidos disponíveis no ambiente computacional para a modelagem de sistemas hidráulicos. Nestes sistemas, as variáveis generalizadas de esforço e de fluxo são a pressão e a vazão, sendo simbolizadas no software, pelas letras P e Q em maiúsculo, respectivamente.

Tabela 5.4. Elementos hidráulicos.

Ícone	Elemento	Ganho	Grandeza	Relação constitutiva
•	Vértice hidráulico			
	Referência hidráulica			
	Fonte de pressão	P	Pressão	$e = P$
	F. de pressão controlada	Kp	Ganho de pressão	$e = K_p.p$
	Fonte de vazão	Q	Vazão	$f = Q$
	F. de vazão controlada	Kq	Ganho de vazão	$f = K_q.p$
	Tubo resistor	Rf	Resistência fluida	$e = R_f.f$
	Tubo indutor	Lf	Indutância fluida	$e = L_f.S.f$
	Reservatório	Cf	Capacitância fluida	$e = C_f^{-1}S^{-1}.f$
	Reservatório	Cf	Capacitância fluida	$e = C_f^{-1}S^{-1}.f$
	Transformador hidráulico	N	Transformação	$e_2 = N^{-1}.e_1, f_2 = N.f_1$

A figura 5.5 mostra o desenho gráfico habitual de um sistema hidráulico, [12], onde uma fonte de vazão Q bombeia o fluido para dois reservatórios de capacitância Cf1 e Cf2 interligados por um tubo que possui indutância fluida Lf1 e resistência fluida Rf1. O segundo reservatório do sistema possui ainda uma abertura para o meio exterior com resistência fluida Rf2. O desenho gráfico deste sistema na área de montagem do ambiente computacional é visto na figura 5.6. A principal diferença entre ambos está na representação das interligações entre os elementos. No software, as tubulações são representadas por segmentos de reta; mas, de uma forma geral, a aparência visual do sistema é a mesma. Assim como nos sistemas mecânicos é necessário utilizar um elemento referência hidráulica

para definir a pressão de referência para os demais elementos do sistema.

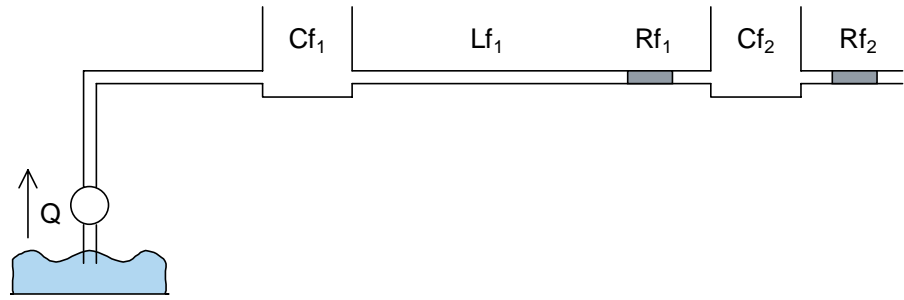


Figura 5.5. Desenho gráfico de sistema hidráulico.

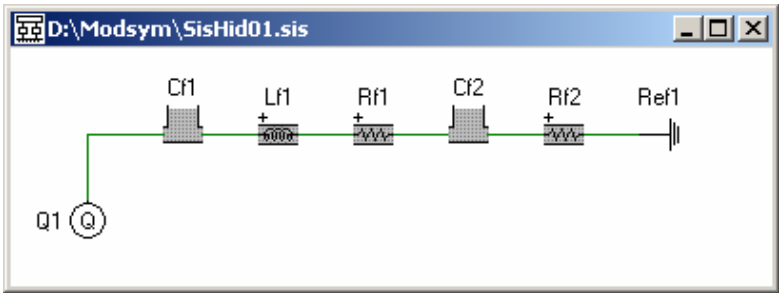


Figura 5.6. Exemplo de sistema hidráulico.

5.2.2.6. Modelagem de Sistemas Físicos com Acopladores

Finalmente, a tabela 5.5 mostra os componentes acopladores disponíveis no ambiente computacional que permitem interligar elementos de sistemas físicos diferentes. Conforme citado no Capítulo 4, o software implementa acopladores genéricos que podem representar qualquer dispositivo físico capaz de interligar dois domínios físicos distintos, simulando, desta forma, o funcionamento de dispositivos como motores, geradores, virabrequins, entre outros.

Tabela 5.5. Elementos acopladores.

Ícone	Elemento	Sistema da Porta 1	Sistema da Porta 2
	Acoplador Ele – Mct	Elétrico	Mecânico translacional
	Acoplador Ele – Mcr	Elétrico	Mecânico rotacional
	Acoplador Ele – Hid	Elétrico	Hidráulico
	Acoplador Mct – Mcr	Mecânico translacional	Mecânico rotacional
	Acoplador Mct – Hid	Mecânico translacional	Hidráulico
	Acoplador Mcr – Hid	Mecânico rotacional	Hidráulico

A relação constitutiva dos acopladores é dada pela equação 2.16, possibilitando qualquer combinação linear entre as variáveis generalizadas de esforço e fluxo em ambas as portas do dispositivo.

A figura 5.7 mostra o modelo para um motor DC controlado pela corrente de armadura, [2]. O resistor R_a e o indutor L_a representam a resistência e a indutância da armadura do motor, respectivamente. O torque desenvolvido pelo motor (T_m) é proporcional a corrente na sua armadura (i_a), de acordo com a equação 5.1:

$$T_m(S) = K_m \cdot i_a(S) \quad (5.1)$$

A força contra-eletromotriz (V_b) é proporcional à velocidade angular do motor (W), de acordo com a equação 5.2:

$$V_b(S) = K_b \cdot W(S) \quad (5.2)$$

O torque produzido pelo motor é aplicado a um corpo com inércia J_1 e amortecimento B_1 , conforme mostra a figura 5.7.

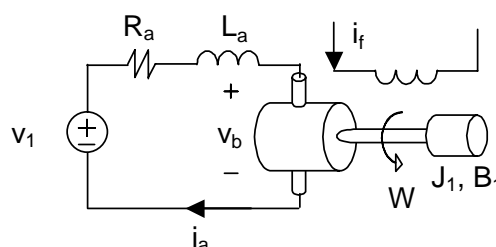


Figura 5.7. Modelo de Motor DC controlado pela armadura.

A figura 5.8 mostra o motor DC modelado no ambiente computacional. O acoplador elétrico – mecânico rotacional é utilizado para representar as conversões existentes entre corrente elétrica e torque, e entre tensão elétrica e velocidade angular, dadas pelas equações 5.1 e 5.2.

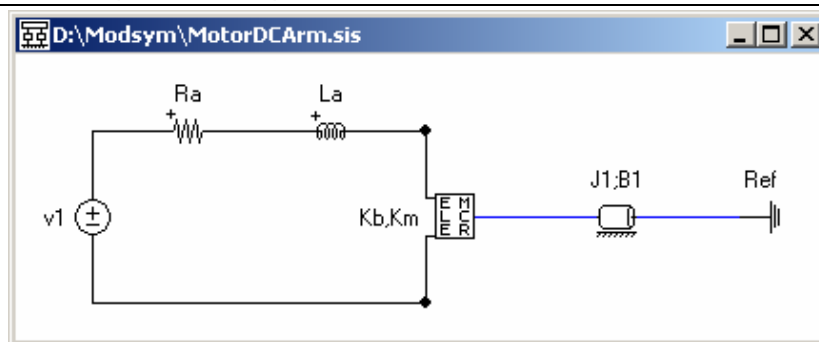


Figura 5.8. Exemplo de sistema físico com acoplador.

Ainda na figura 5.8, é possível observar que o acoplador realiza duas conexões na porta elétrica e uma na porta mecânica. De fato, o referencial elétrico, fornecido pela fonte de tensão, é conectado ao conector elétrico inferior do acoplador. Na porta mecânica, entretanto, subentende-se que o referencial mecânico encontra-se conectado a referência Ref1, portanto apenas uma conexão é realizada. Os símbolos K_b e K_m representam as constantes do motor e são utilizados pelo acoplador para associar as variáveis de esforço e fluxo na primeira porta de energia com essas mesmas variáveis na segunda porta, de acordo com as equações 5.1 e 5.2. Por fim, o corpo acoplado ao eixo do motor é representado por um elemento Inércia + Atrito com inércia J_1 e amortecimento B_1 .

Sistemas físicos distintos podem ser acoplados também através de fontes controladas de energia. A figura 5.9 apresenta um sistema de nível em malha aberta composto por um gerador, um motor, um trem de engrenagem, um reservatório e uma válvula, [20]. O objetivo do sistema, quando em malha fechada, é controlar a altura (H) do fluido no reservatório e seu funcionamento é descrito a seguir.

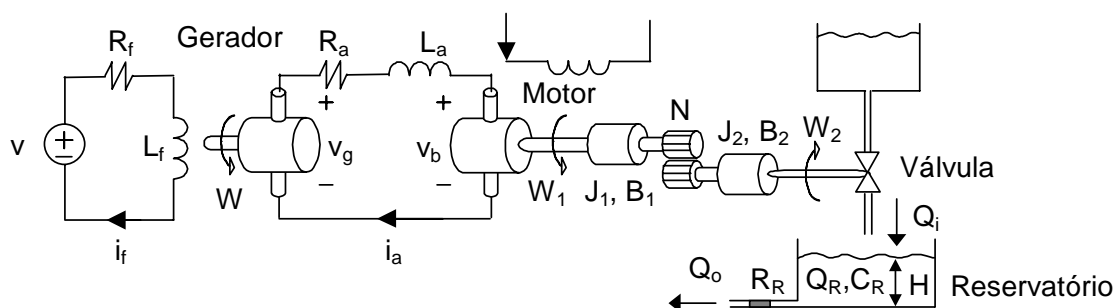


Figura 5.9. Modelo de sistema de controle de nível em malha aberta.

A tensão de referência (v) aplicada ao gerador submete ao motor DC uma tensão (v_g) proporcional a corrente no gerador (i_f), dada pela seguinte equação:

$$v_g(S) = K_g \cdot i_f(S) \quad (5.3)$$

O motor DC, controlado pela armadura, tem o comportamento descrito pelas equações 5.1 e 5.2. O trem de engrenagem ideal, com relação de transformação dada por N , é usado para controlar a velocidade do eixo que aciona a válvula. A relação entre as velocidades W_1 e W_2 é dada pela equação 5.4:

$$W_1(S) = N \cdot W_2(S) \quad (5.4)$$

A vazão na válvula (Q_i), proporcional à posição do eixo acoplado ao trem de engrenagem (θ_2), é dada pela seguinte equação :

$$Q_i(S) = K_i \cdot S^{-1} \cdot W_2(S) \quad (5.5)$$

A vazão Q_i determina o volume de fluido que entra no reservatório. Uma parte do fluido é acumulada no tanque que possui capacitância fluida igual a C_R . A outra é drenada através de uma tubulação com resistência fluida igual a R_R . Em resumo, a tensão aplicada ao gerador aciona o motor que controla a abertura da válvula e, conseqüentemente, varia a altura do fluido no reservatório.

A figura 5.10 mostra o sistema de nível modelado no ambiente computacional. O gerador é representado por uma fonte de tensão controlada com ganho K_g . A variável de controle ρ desta fonte é o fluxo no indutor L_f . Desta forma, a tensão na fonte satisfaz a equação 5.3.

O motor DC é representado pelo elemento acoplador e seu funcionamento é idêntico ao exemplo anterior, vista na figura 5.8. O transformador mecânico rotacional representa o trem de engrenagem e possui relação de transformação N , de acordo com a equação 5.4. O transformador interliga os dois corpos acoplados ao motor.

Finalmente, a válvula é representada por uma fonte de vazão controlada com ganho $K_i \cdot S^{-1}$ e variável de controle ρ dada pelo esforço no corpo $J_2; B_2$, satisfazendo assim a equação 5.5.

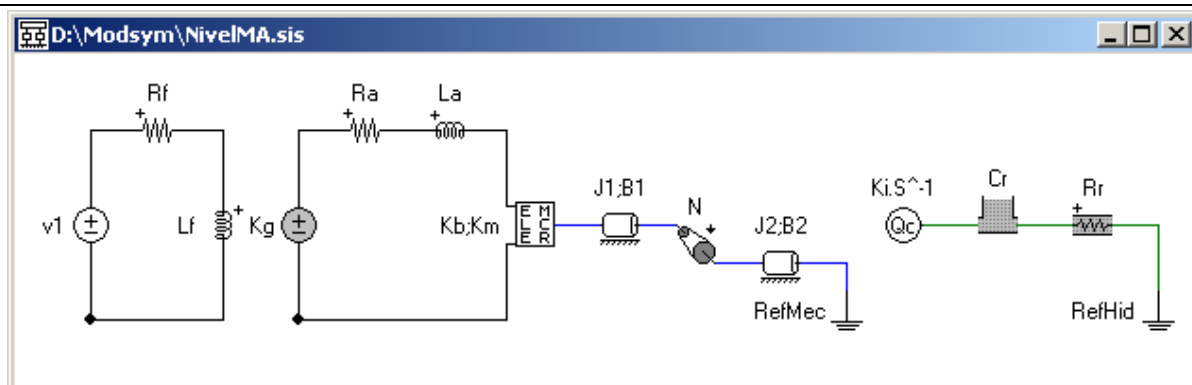


Figura 5.10. Exemplo de sistema físico com acoplador e fonte controlada.

5.2.2.7. Modelagem de Sistemas Físicos em Malha Fechada

As fontes controladas podem ser utilizadas também para modelar sistemas físicos em malha fechada. Como o ganho desses componentes é dado pelo produto entre um ganho numérico, um símbolo e uma potência da variável S , é possível utilizá-los na representação de controladores PID, [2].

A figura 5.11 mostra um sistema de controle de velocidade de um motor DC utilizando um controlador deste tipo. A referência do sistema é dada pela fonte de tensão $vref$. A realimentação é realizada por uma fonte de tensão controlada com ganho Ks , que representa o sensor de velocidade. A variável de controle ρ dessa fonte é o esforço da carga, ou seja, a velocidade W do motor.

O controlador PID do sistema é representado por três fontes de tensão controladas: a primeira fonte tem ganho Kp e representa a parte proporcional do controlador; a segunda, com ganho $Ki.S^{-1}$, constitui a parte integrativa; a terceira, com ganho $Kd.S$, constitui a parte derivativa. A variável de controle ρ dessas fontes é dada pela tensão no resistor R , que representa o sinal de erro.

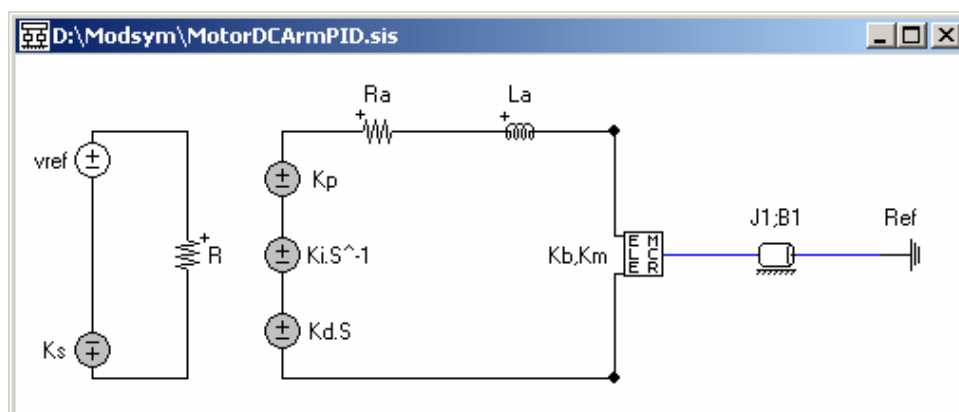


Figura 5.11. Exemplo de sistema físico em malha fechada.

As três fontes controladas de tensão dispostas em série produzem a tensão v_m dada pela equação 5.6, que representa o sinal de controle aplicado ao motor:

$$v_m = (K_p + K_i \cdot S^{-1} + K_d \cdot S)(v_{ref} - K_s \cdot W) \quad (5.6)$$

Enfim, os exemplos vistos mostram que o ambiente computacional permite a construção de uma quantidade razoável de sistemas de físicos, evidenciando a sua importância para o processo de ensino e aprendizagem na área de controle.

5.2.3. Área de Montagem de Grafos de Ligação

A segunda área de montagem do ambiente computacional permite a modelagem de sistemas físicos a partir de grafos de ligação. Conforme citado na Seção 2.3.5, os grafos de ligação são dígrafos cujos vértices representam os pontos do sistema físico com o mesmo valor para a variável generalizada de esforço. Os ramos deste grafo representam os elementos físicos do sistema que podem ser classificados em fontes, acumuladores, dissipadores, modeladores, transformadores e acopladores de energia.

A figura 5.12 apresenta a interface gráfica de modelagem do ambiente computacional. Quando o diagrama gráfico selecionado é um grafo de ligação, a paleta de componentes, vista na janela principal do software, apresenta os elementos generalizados que podem ser utilizados para modelar o sistema físico. A janela abaixo da principal representa a área de montagem de grafos de ligação, onde os elementos do grafo podem ser inseridos e conectados de forma a definir a estrutura de um grafo em particular.

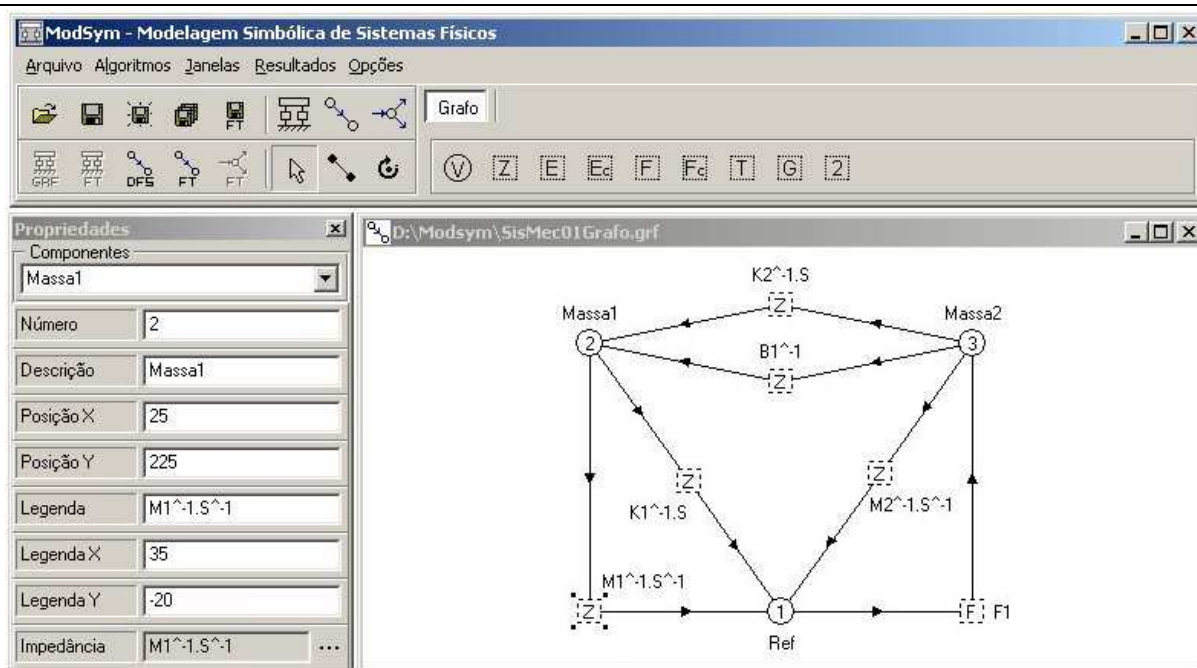







Figura 5.12. Área de montagem de grafos de ligação.

A utilização da área de montagem de grafos é análoga ao apresentado anteriormente para a modelagem de sistemas físicos baseada nos diagramas de desenhos gráficos. Nesta área de montagem, os grafos de ligação podem ser modelados a partir de nove ícones gráficos, representando os seus vértices e os elementos generalizados do sistema físico, conforme mostra a tabela 5.6. Cada elemento generalizado é normalmente associado a um ganho simbólico e possui também uma relação constitutiva, conforme discutido no Capítulo 2.

Os vértices do grafo de ligação são representados graficamente por uma circunferência numerada, enquanto que os ramos são representados por um elemento generalizado e dois segmentos de reta orientados que interligam o elemento aos seus vértices terminais, como mostra a figura 5.12.

Tabela 5.6. Elementos do grafo de ligação.

Ícone	Elemento	Ganho	Grandeza	Relação constitutiva
⓪	Vértice			
[Z]	Impedância	Z	Impedância	$e = Z.f$
[E]	Fonte de esforço	E	Esforço	$e = E$
[E _c]	Fonte de esforço controlada	Ke	Ganho de esforço	$e = Ke.p$

	Fonte de fluxo	F	Fluxo	$f = F$
	Fonte de fluxo controlada	Kf	Ganho de fluxo	$f = Kf.\rho$
	Transformador	N	Transformação	$e_2 = N^{-1}.e_1, f_2 = N.f_1$
	Girador	G	Giração	$e_2 = G^{-1}.f_1, f_2 = G.e_1$
	Elemento genérico de 2 portas			Equação 2.13

Os elementos generalizados podem ser divididos em dois grupos. No primeiro grupo estão: as impedâncias, que representam os dissipadores e acumuladores de energia do sistema; as fontes de esforço e fluxo, representando as fontes de energia; e as fontes controladas, representando os dispositivos cujo esforço ou fluxo é proporcional à outra variável do sistema. Esses elementos têm uma única porta de energia e, por conseguinte, originam um único ramo no grafo.

No segundo grupo estão os transformadores, giradores e elementos genéricos de duas portas, que representam os transformadores e acopladores do sistema físico, apresentados anteriormente. Todos esses elementos originam dois ramos no grafo; um para cada porta de energia.

O grafo de ligação apresentado na figura 5.12 refere-se ao sistema mecânico visto nas figuras 2.22 e 5.3. Este grafo foi obtido pelo algoritmo implementado no software. Conforme visto na figura, o algoritmo nomeia os vértices do grafo de acordo com o elemento físico do sistema que o origina. Isto facilita ao estudante analisar a representação de um sistema físico na forma de grafo de ligação. Ainda nesta figura, é possível verificar que o ganho dos ramos é obtido a partir das relações constitutivas dos elementos, dadas pela tabela 5.2, para este exemplo.

A figura 5.13, vista abaixo, mostra o grafo de ligação do sistema mecânico apresentado na figura 5.4. Na figura à esquerda, as conexões dos elementos foram removidas. Desta forma, é fácil perceber que os elementos generalizados possuem também conectores, os quais são utilizados para interligar os elementos aos vértices. Diferentemente dos desenhos gráficos, não é possível interligar um elemento a outro nem um vértice a outro. Cada elemento possui dois conectores por porta de energia que devem ser interligados obrigatoriamente aos seus vértices

terminais. Na figura à direita, percebe-se novamente que os vértices foram nomeados de acordo com o elemento do desenho gráfico. O vértice 4, neste caso, não estava explícito no desenho do sistema. É importante salientar também que o algoritmo que obtém o grafo não organiza seus vértices no plano. Mas, a área de montagem permite ao usuário manipular e alterar o grafo gerado pelo algoritmo.

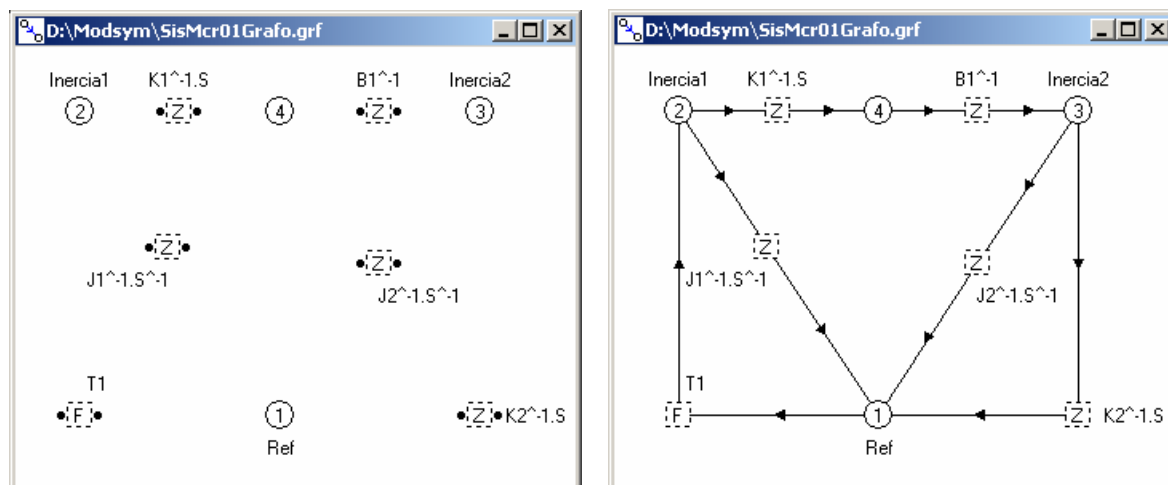


Figura 5.13. Exemplo de grafo de ligação.

5.2.4. Área de Montagem de Diagramas de Fluxo de Sinal

A terceira área de montagem do ambiente computacional permite a modelagem de sistemas físicos a partir de diagramas de fluxo de sinal. Conforme citado na Seção 2.4, os diagramas de fluxo de sinal são dígrafos cujos vértices representam as variáveis de um sistema de equações lineares e os ramos, as relações entre estas variáveis. Desta forma, um DFS representa graficamente um conjunto de equações que pode descrever o comportamento de sistemas físicos, lineares e invariantes no tempo, tais como os sistemas abordados neste trabalho.

A figura 5.14 apresenta a interface gráfica de modelagem do software, quando um DFS é selecionado. A paleta de componentes, vista na janela principal, mostra apenas os elementos utilizados para modelar tais diagramas. A janela logo abaixo representa a área de montagem de DFS's, onde os elementos de um diagrama em particular podem ser inseridos e conectados.

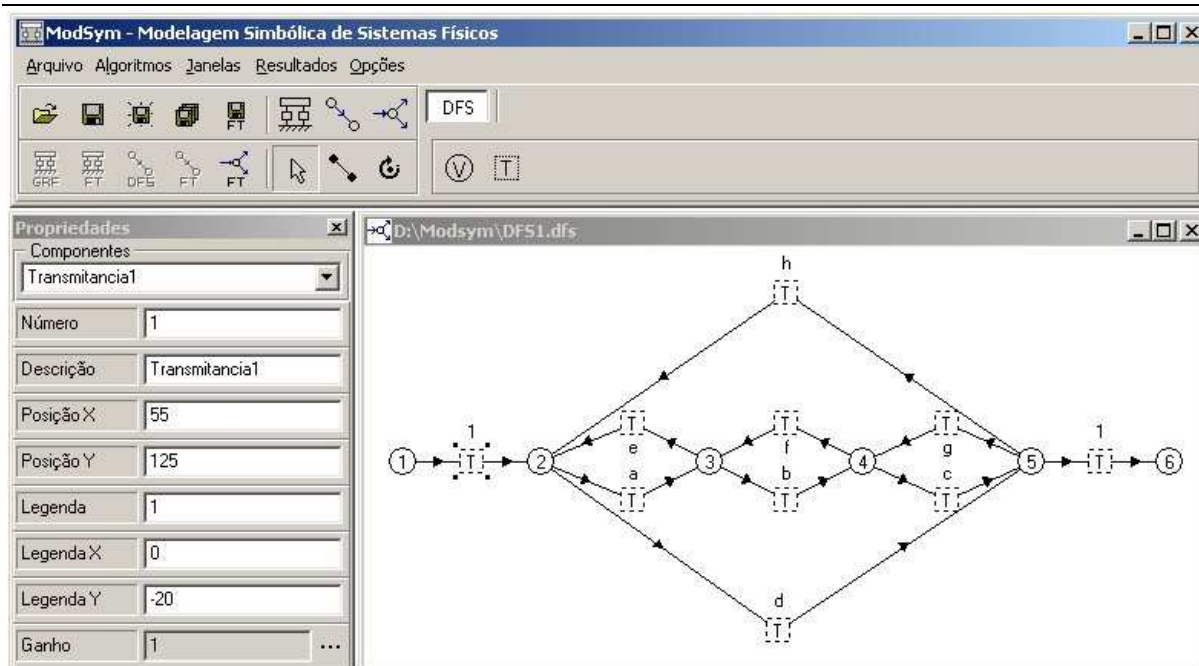


Figura 5.14. Área de montagem de diagramas de fluxo de sinal.

A utilização desta área de montagem é idêntica à apresentada anteriormente para a área de montagem de grafos de ligação. Entretanto, são disponíveis apenas dois ícones gráficos, representado as variáveis e as transmitâncias do diagrama, conforme mostra a tabela 5.7.

Tabela 5.7. Elementos do diagrama de fluxo de sinal.

Ícone	Elemento
	Vértice
	Transmitância

Assim como os grafos de ligação, os diagramas de fluxo de sinal visualizados no ambiente computacional podem ser definidos diretamente pelo usuário ou produzidos pela execução de algum algoritmo. Neste caso, pelo algoritmo que obtém o DFS de um sistema a partir do grafo de ligação. O DFS exibido na figura 5.14, por exemplo, foi definido diretamente na área de montagem e representa o diagrama de fluxo de sinal visto anteriormente na figura 2.24.

5.3. Interface Gráfica de Resultados

Uma vez conhecida a funcionalidade da interface gráfica de modelagem, é

importante visualizar os resultados que podem ser obtidos a partir dos sistemas modelados no ambiente computacional.

As principais funções implementadas no software, discutidas no Capítulo 4, podem ser didaticamente resumidas em: obtenção do grafo de ligação a partir do desenho gráfico do sistema, obtenção do DFS a partir do grafo de ligação, cálculo de funções de transferência nas formas simbólica e numérica, cálculo de funções de sensibilidade paramétrica e geração do arquivo de simulação no formato do SINCON.

Todos os exemplos apresentados nesta seção serão baseados no desenho gráfico do sistema visto na figura 5.15, que modela um motor DC controlado pela corrente de armadura e em malha fechada com um controlador Proporcional. Apesar do software permitir o cálculo da função de transferência diretamente a partir do desenho gráfico, este cálculo será realizado passo a passo, de forma a explorar os resultados gerados por cada um dos algoritmos implementados no ambiente computacional. Desta forma, primeiro será obtido o grafo do sistema, em seguida o seu DFS e finalmente, a função de transferência através da regra de Mason.

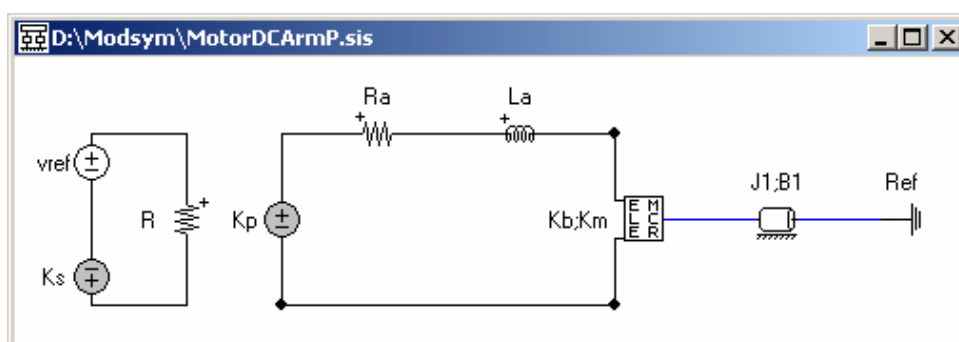


Figura 5.15. Motor DC em malha fechada com controlador Proporcional.

5.3.1. Grafo de Ligação do Sistema

O grafo de ligação para o motor DC obtido pelo ambiente computacional pode ser visto na figura 5.16. Basicamente, a tensão de referência é representada por uma fonte de esforço; o sensor de velocidade e o controlador Proporcional, por fontes de esforço controladas; o acoplador por um elemento generalizado de duas portas e os demais elementos elétricos e mecânicos, por impedâncias

generalizadas. As relações constitutivas dos elementos também podem ser visualizadas na figura.

Como dito anteriormente, é importante ressaltar que o software não calcula a posição ideal dos vértices e elementos do grafo. Na figura 5.16, a disposição dos elementos foi alterada pelo usuário para melhorar sua apresentação visual.

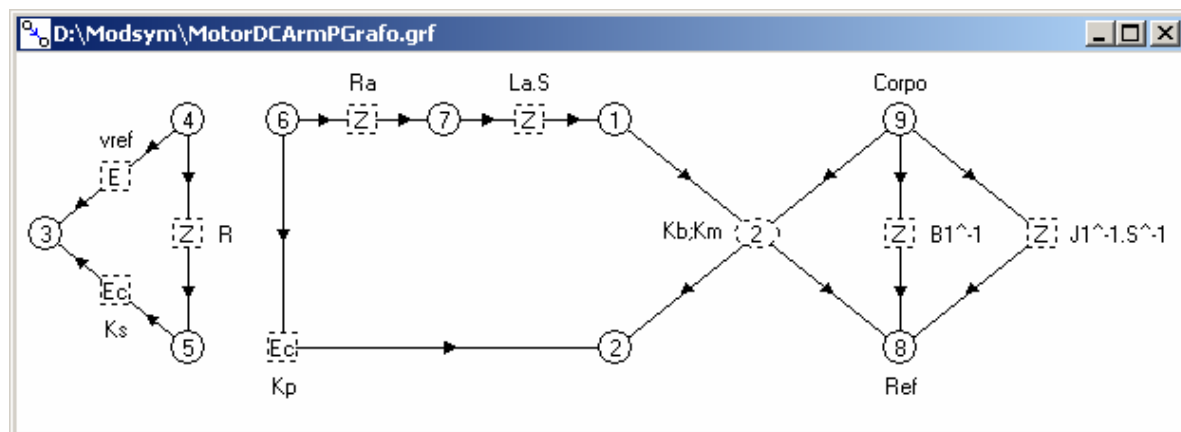
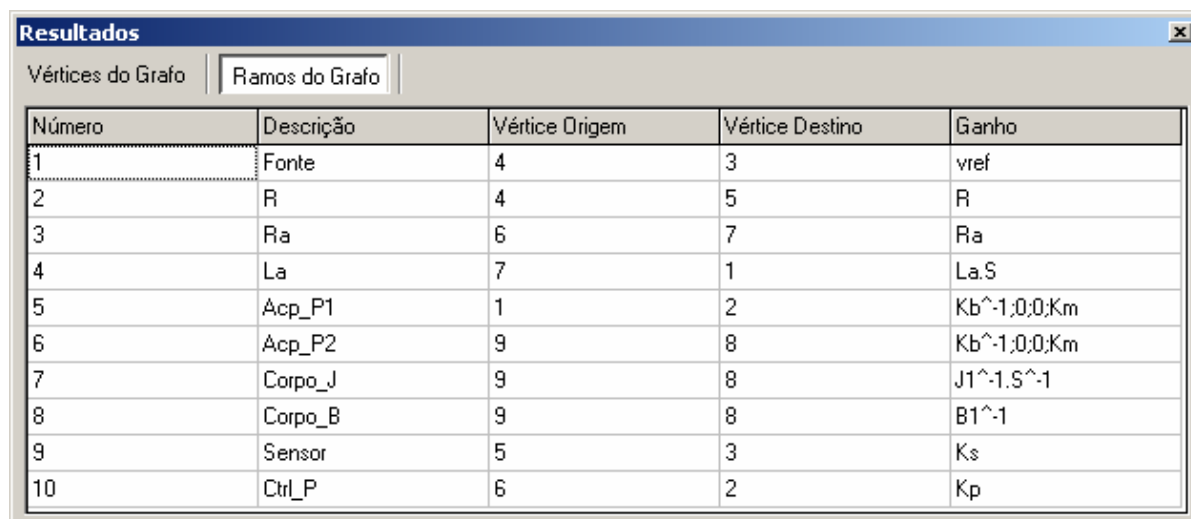


Figura 5.16. Grafo de ligação do Motor DC.

O algoritmo implementado para converter a representação do desenho gráfico do sistema em grafo de ligação é apresentado no Capítulo 6. Esse algoritmo não produz resultados intermediários relevantes ao usuário, de forma que, além do próprio grafo de ligação na forma visual, o software apresenta apenas as listas de vértices e ramos do grafo, vistas nas figuras 5.17 e 5.18, respectivamente. Na verdade, o resultado mais importante gerado pelo algoritmo é o próprio grafo. De fato, é fundamental que o estudante da Engenharia de Controle compreenda a sistemática implementada para a construção do grafo de ligação que é o primeiro passo realizado na obtenção do modelo matemático do sistema físico.

Resultados	
Vértices do Grafo	Ramos do Grafo
Número	Descrição
1	Vertice1
2	Vertice2
3	_Vertice1
4	_Vertice2
5	_Vertice3
6	_Vertice4
7	_Vertice5
8	Ref
9	Corpo

Figura 5.17. Vértices do grafo de ligação do Motor DC.



Número	Descrição	Vértice Origem	Vértice Destino	Ganho
1	Fonte	4	3	vref
2	R	4	5	R
3	Ra	6	7	Ra
4	La	7	1	La.S
5	Acp_P1	1	2	$Kb^{-1};0;0;Km$
6	Acp_P2	9	8	$Kb^{-1};0;0;Km$
7	Corpo_J	9	8	$J1^{-1}.S^{-1}$
8	Corpo_B	9	8	$B1^{-1}$
9	Sensor	5	3	Ks
10	Ctrl_P	6	2	Kp

Figura 5.18. Ramos do grafo de ligação do Motor DC.

5.3.2. Diagrama de Fluxo de Sinal do Sistema

O diagrama de fluxo de sinal para o motor DC obtido pelo ambiente computacional pode ser visto na figura 5.19. O algoritmo implementado para realizar esta tarefa, uma das contribuições deste trabalho, é apresentado no Capítulo 6.

Como mostra a figura, o algoritmo inclui no DFS as variáveis generalizadas de esforço e fluxo em todos os elementos do sistema físico, possibilitando o cálculo de funções de transferência entre qualquer uma dessas variáveis e a excitação do sistema. As variáveis de esforço são indicadas no diagrama por $E(x)$, onde x é um elemento do sistema; de forma análoga, as variáveis de fluxo são indicadas por $F(x)$. As transmitâncias representam as relações entre as variáveis, que são obtidas a partir das relações constitutivas e de interconexão entre os elementos.

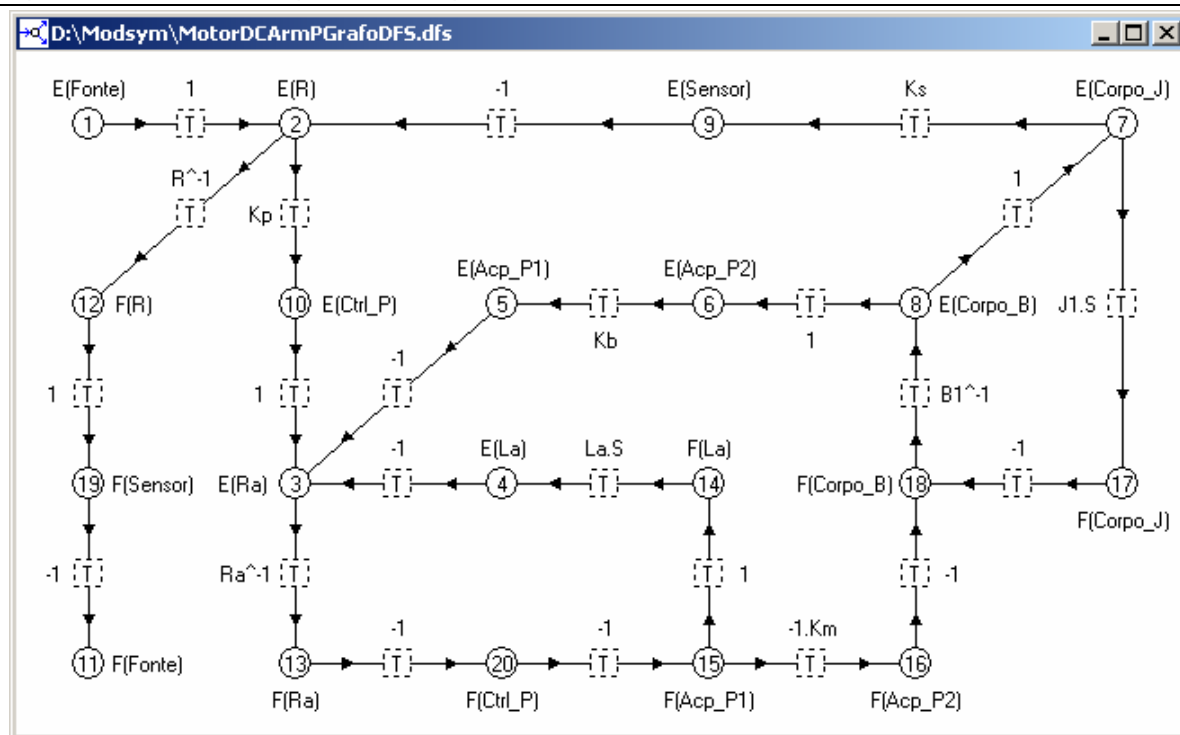


Figura 5.19. DFS do Motor DC.

Na realização desta tarefa, o ambiente computacional disponibiliza uma série de resultados intermediários produzidos pelo algoritmo, os quais são importantes para o estudante da Engenharia de Controle. A figura 5.20, por exemplo, apresenta as equações que são obtidas a partir do grafo de ligação do sistema utilizadas na construção do DFS. É na obtenção dessas equações que os estudantes apresentam, normalmente, grandes dificuldades.

As outras informações disponíveis estão relacionadas ao processamento utilizado na obtenção dessas equações e ao algoritmo de busca implementado para calcular as transmitâncias do diagrama. Todo este processamento é abordado no próximo capítulo.

Resultados																					
Variáveis	A1	A2	A3	A4	A5	A	Algoritmo de Busca										Solução				
Matriz de Equações																					
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
1	0	0	-1	-1	-1	0	0	0	0	1	0	0	0	0	0	0		0	0	0	
2	-1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0		0	0	0	
3	0	0	0	0	0	-1	0	1	0	0	0	0	0	0	0	0		0	0	0	
4	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0		0	0	0	
5	0	-1	0	0	0	0	0	0	0	0	0	R	0	0	0	0		0	0	0	
6	0	0	-1	0	0	0	0	0	0	0	0	0	Ra	0	0	0		0	0	0	
7	0	0	0	-1	0	0	0	0	0	0	0	0	0	La.S	0	0		0	0	0	
8	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	J1 ⁻¹ .S ⁻¹	0	0	0	
9	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0		B1 ⁻¹	0	0	
10	0	0	0	0	0	0	Ks	0	-1	0	0	0	0	0	0	0		0	0	0	
11	0	Kp	0	0	0	0	0	0	0	-1	0	0	0	0	0	0		0	0	0	
12	0	0	0	0	Kb ⁻¹	-1	0	0	0	0	0	0	0	0	0	0		0	0	0	
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Km	1		0	0	0	
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0		0	0	-1	
15	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0		0	0	1	
16	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0		0	0	0	
17	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0		0	1	0	
18	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0		0	-1	0	
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1		1	0	0

Figura 5.20. Matriz de equações do Motor DC.

5.3.3. Funções de Transferência do Sistema

Uma vez definido o sistema físico, seja através do desenho gráfico, do grafo de ligação ou do DFS, o cálculo da função de transferência pode ser realizado. Para isto, basta selecionar as variáveis de entrada e saída da função de transferência. A interface de seleção das variáveis é apresentada na figura 5.21.

Figura 5.21. Seleção das variáveis da função de transferência.

A figura 5.22 mostra a função de transferência na forma simbólica entre o esforço na inércia e o esforço na fonte de tensão, ou seja, a função de transferência entre a velocidade do motor e a tensão de referência.

Função de Transferência

Variáveis | Símbolos | **FT Simbólica** | FT Simbólica | Sensibilidade Paramétrica | FT Numérica

Numerador
Km.Kp

Denominador
 $J1.La.S^2 + B1.La.S + J1.Ra.S + B1.Ra + Kb.Km + Km.Kp.Ks$

Figura 5.22. Função de transferência simbólica do motor DC.

A função de transferência também pode ser visualizada na forma fatorada, conforme mostra a figura 5.23. Em ambos casos, os fatores da função estão ordenados pela potência da variável S e, em seguida, alfabeticamente.

Função de Transferência

Variáveis | Símbolos | FT Simbólica | **FT Simbólica** | Sensibilidade Paramétrica | FT Numérica

Numerador
S⁰ + Km.Kp

Denominador

S ²	+ J1.La		
S ¹	+ B1.La	+ J1.Ra	
S ⁰	+ B1.Ra	+ Kb.Km	+ Km.Kp.Ks

Figura 5.23. Função de transferência simbólica fatorada do motor DC.

Função de Transferência

Variáveis | **Símbolos** | FT Simbólica | FT Simbólica | Sensibilidade Paramétrica | FT Numérica

✓ Calcular FT Numérica

Símbolo	Valor
B1	1
J1	1
Kb	1
Km	1
Kp	10
Ks	1
La	1
R	1
Ra	1

Figura 5.24. Lista de símbolos da função de transferência do motor DC.

A figura 5.24 apresenta a lista de símbolos presentes na função de

transferência simbólica. Conforme visto, a interface permite que o estudante defina valores para cada um dos símbolos e calcule a função na forma numérica. No exemplo, o ganho K_p do controlador recebe valor numérico igual a 10. Os demais símbolos recebem valor unitário. A figura 5.25 apresenta a função de transferência numérica obtida a partir dos valores atribuídos aos símbolos.

Função de Transferência

Variáveis | Símbolos | FT Simbólica | FT Simbólica | Sensibilidade Paramétrica | **FT Numérica**

✓ Salvar Arquivo do Sincon

Ganho
10

Numerador
1

Denominador
 $s^2 + 2.5s + 12$

Figura 5.25. Função de transferência numérica do motor DC.

Uma vez obtida a FT numérica é possível gerar o arquivo no formato do SINCON, o que permite realizar outras tarefas relacionadas à Engenharia de Controle. A figura 5.26, por exemplo, mostra a simulação da FT no SINCON.

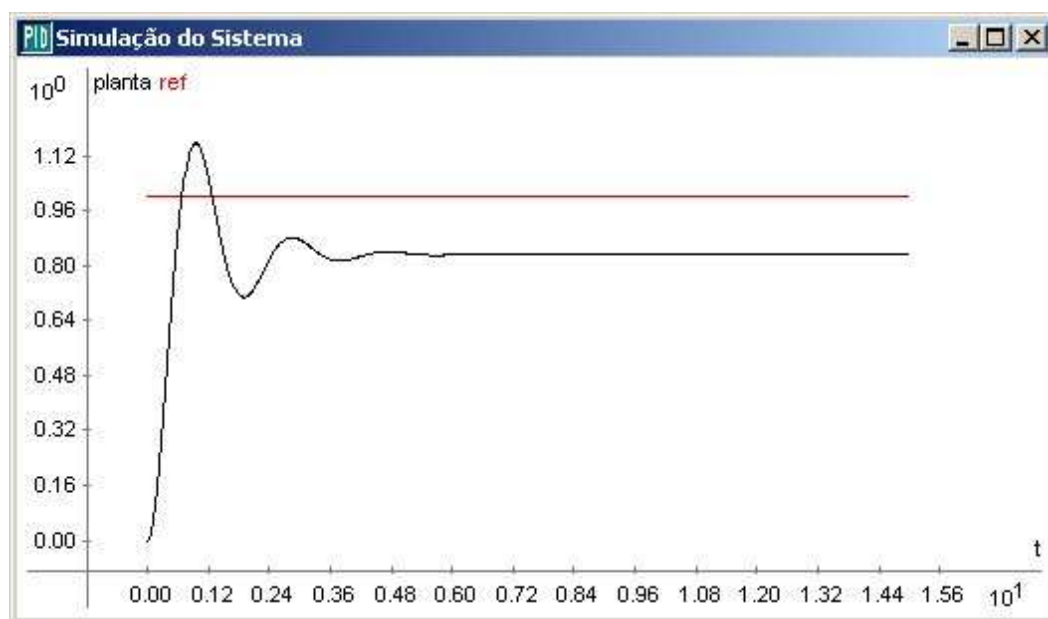
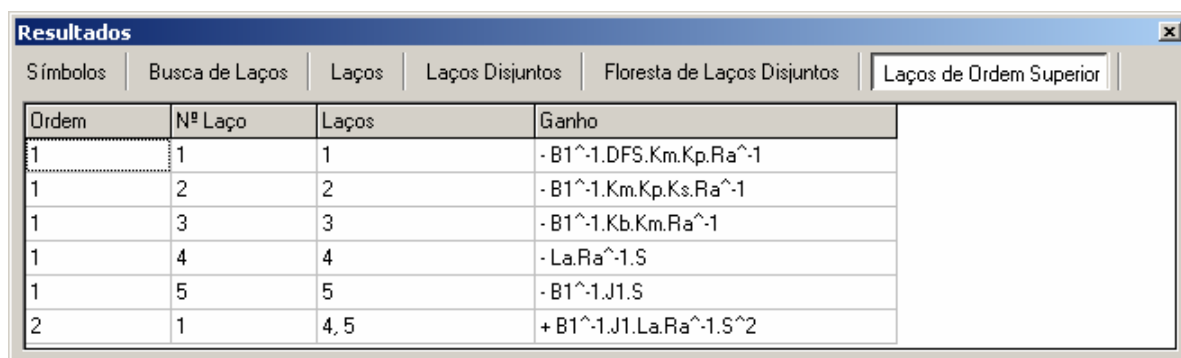


Figura 5.26. Simulação do motor DC realizada no SINCON.

5.3.4. Resultados da Regra de Mason

No cálculo da função de transferência simbólica, que é realizada pela regra de Mason, o ambiente computacional disponibiliza todos os resultados relevantes a esta regra, mostrando desde o algoritmo de Lin-Alderson, [15], que realiza a busca dos laços do diagrama, até as lista de laços de ordem superior que são utilizadas no cálculo da FT simbólica. A figura 5.27, por exemplo, mostra as lista de laços de ordem superior obtidas para o DFS do motor DC. A verificação destes resultados é de grande relevância para os estudantes da Engenharia de Controle.

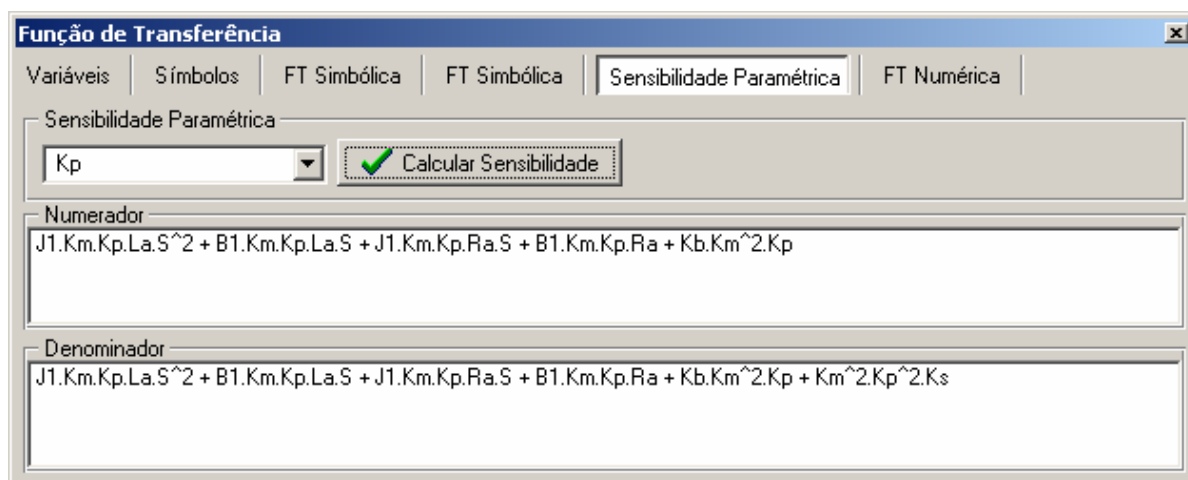


Ordem	Nº Laço	Laços	Ganho
1	1	1	$-B1^{-1}.DFS.Km.Kp.Ra^{-1}$
1	2	2	$-B1^{-1}.Km.Kp.Ks.Ra^{-1}$
1	3	3	$-B1^{-1}.Kb.Km.Ra^{-1}$
1	4	4	$-La.Ra^{-1}.S$
1	5	5	$-B1^{-1}.J1.S$
2	1	4, 5	$+B1^{-1}.J1.La.Ra^{-1}.S^2$

Figura 5.27. Lista de laços de ordem superior da regra de Mason.

5.3.5. Função de Sensibilidade Paramétrica

Finalmente, o ambiente computacional realiza o cálculo da sensibilidade normalizada de uma de função de transferência, dada pela equação 2.39, em relação a algum parâmetro do sistema. A figura 5.28, por exemplo, mostra a sensibilidade da FT do motor DC em relação ao ganho Kp do controlador.



Função de Transferência

Variáveis | Símbolos | FT Simbólica | FT Simbólica | **Sensibilidade Paramétrica** | FT Numérica

Sensibilidade Paramétrica

Kp

Numerador

$$J1.Km.Kp.La.S^2 + B1.Km.Kp.La.S + J1.Km.Kp.Ra.S + B1.Km.Kp.Ra + Kb.Km^2.Kp$$

Denominador

$$J1.Km.Kp.La.S^2 + B1.Km.Kp.La.S + J1.Km.Kp.Ra.S + B1.Km.Kp.Ra + Kb.Km^2.Kp + Km^2.Kp^2.Ks$$

Figura 5.28. Sensibilidade da FT do motor DC em relação a Kp.

Em suma, o ambiente computacional proposto apresenta uma boa funcionalidade na atividade de modelagem de sistemas físicos lineares. A utilização de elementos básicos dos domínios elétricos, mecânicos e hidráulicos e, em particular, as fontes controladas e os dispositivos acopladores permitem boa parte dos sistemas lineares de controle seja modelada no ambiente computacional, evidenciando a sua funcionalidade na atividade educacional.

Além disso, a metodologia adotada pelo software para o cálculo de funções de transferência possibilita que um estudante de Engenharia de Controle acompanhe, passo a passo, os procedimentos necessários à obtenção de tais funções. Neste processamento, são apresentados ao estudante temas importantes para a teoria de controle, como por exemplo: os grafos de ligação, que mostram o sistema elétrico equivalente de um sistema físico, os diagramas de fluxo de sinal, que apresentam graficamente o fluxo de energia no sistema e a regra de Mason. Conforme visto neste capítulo, o software apresenta ainda os resultados intermediários mais importantes de cada um algoritmos empregados no cálculo da função de transferência, os quais são relevantes no estudo de um sistema de controle. A obtenção das equações que descrevem o comportamento do sistema físico, por exemplo, auxiliada pelo ambiente computacional, é um dos maiores obstáculos para os estudantes na formulação do modelo matemático de um sistema físico.

No próximo capítulo, são apresentados dois dos principais algoritmos implementados no ambiente computacional: o algoritmo que constrói o grafo de ligação do sistema e o algoritmo que obtém o diagrama de fluxo de sinal. Com a regra de Mason, esses algoritmos permitiram o cálculo de funções de transferência de sistemas de vários domínios físicos sem resolver algebricamente as equações que descrevem a dinâmica do sistema físico.

6. Algoritmos

6.1. Introdução

Este capítulo apresenta dois dos principais algoritmos implementados no ambiente computacional proposto. O primeiro algoritmo está relacionado à obtenção do grafo de ligação de um sistema físico a partir do seu desenho gráfico, [12], e o segundo é o responsável pela obtenção do diagrama de fluxo de sinal do sistema a partir deste grafo, [21]. Conforme dito anteriormente, este último algoritmo é uma das contribuições do trabalho. O terceiro algoritmo de maior importância para o ambiente computacional é a regra de Mason, apresentada na Seção 2.5. Detalhes de implementação deste algoritmo podem ser vistos em [15].

Os algoritmos apresentados em seguida estão intimamente relacionados às estruturas de dados vistas no Capítulo 4 onde foram abordados os aspectos gerais da modelagem do ambiente computacional. Conforme visto, os diagramas construídos na interface gráfica são mapeados em estruturas de dados utilizadas para representá-los na memória. É com base nestas estruturas que os algoritmos citados anteriormente são implementados.

6.2. Algoritmo Sistema – Grafo

O algoritmo que obtém o grafo de ligação de um sistema físico a partir do seu desenho gráfico é apresentado em seguida. Para facilitar a referência a ele, será utilizada a denominação algoritmo Sistema-Grafo.

6.2.1. Objetivo

O objetivo do algoritmo Sistema-Grafo é montar a estrutura de dados do grafo

de ligação do sistema físico baseado na estrutura de dados do diagrama de desenho gráfico deste sistema.

O diagrama de colaboração apresentado na figura 4.15 mostra o contexto no qual o algoritmo Sistema-Grafo está inserido. No primeiro momento, o usuário define o diagrama de desenho gráfico do sistema que é representado por um objeto da classe TVFSistema. O diagrama é então mapeado em uma estrutura de dados modelada pela classe TSistema. Em seguida, o método SistemaToGrafo desta classe é executado para definir a estrutura de dados do grafo de ligação, que é um objeto da classe TGrafo. Em síntese, este método controla, passo a passo, a “conversão” entre as estruturas do sistema e do grafo.

6.2.2. Etapas do Algoritmo

O diagrama de atividades da figura 6.1 mostra, passo a passo, as etapas do algoritmo Sistema-Grafo, partindo desde a definição da estrutura de dados do sistema até a obtenção da estrutura de dados do grafo de ligação.

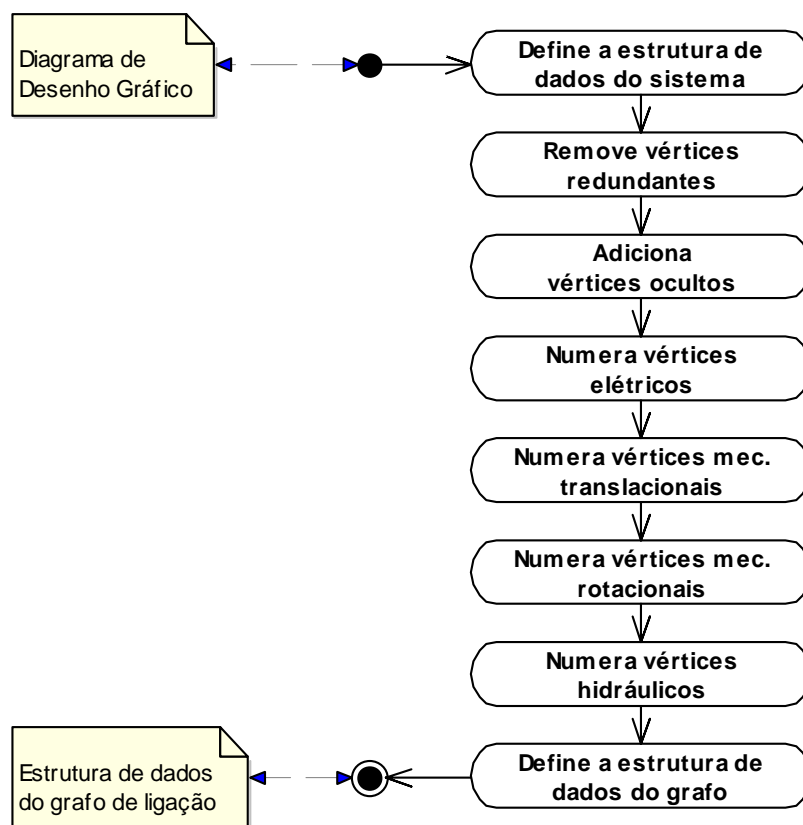


Figura 6.1. Diagrama de atividades do algoritmo Sistema-Grafo.

Cada etapa do algoritmo é explorada em seguida e o processamento é exemplificado com base no desenho gráfico do sistema apresentado na figura 6.2, que representa o motor DC em malha fechada com controlador Proporcional.

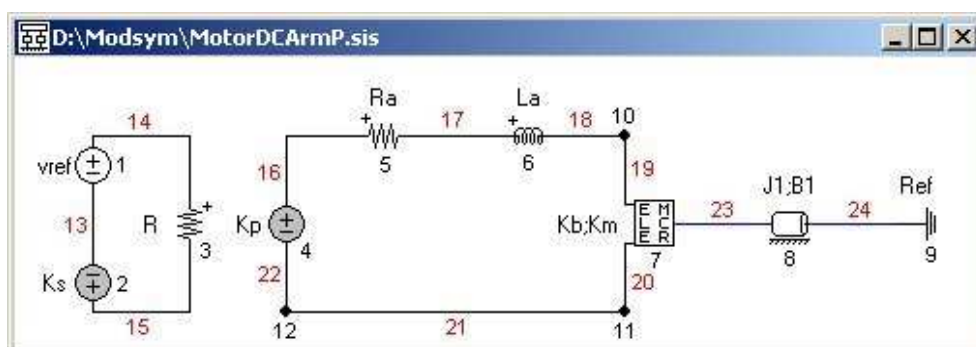


Figura 6.2. Motor DC em malha fechada com controlador Proporcional.

6.2.2.1. Definição da Estrutura de Dados do Sistema

Na primeira etapa do algoritmo é definida a estrutura de dados do sistema com base no desenho realizado na área de montagem. Conforme visto no diagrama de classes da figura 4.3, um diagrama gráfico mantém a lista de todos os objetos incluídos nele. Esta lista é utilizada para gerar a estrutura de dados do sistema vista no diagrama da figura 4.10. Com isto, a informação contida no diagrama gráfico passa a ser manipulada por um objeto da classe TSistema, que mantém as listas dos elementos e das conexões do sistema físico.

Os objetos inseridos em um diagrama são sempre identificados por um número inteiro único, denominado de ID. A figura 6.2 apresenta o ID dos objetos que modelam o motor DC e as tabelas 6.1 e 6.2 apresentam as listas de elementos e conexões geradas a partir do diagrama do motor. Cada coluna das tabelas está associada a um dos atributos dos objetos, descritos nas tabelas 4.2 e 4.3.

Na tabela 6.1, por exemplo, os principais atributos dos elementos podem ser verificados. O atributo ClassID está associado à classe do objeto e é fundamental para identificar que tipo elemento generalizado o representa no grafo de ligação. O atributo Comp indica se o elemento é um componente ou um vértice. Os atributos Desc e Ganho armazenam a descrição e o ganho do elemento, respectivamente. O atributo Num é usado para numerar os vértices do grafo de ligação. Finalmente, os

atributos SisID e VarCtrl armazenam respectivamente dados sobre o sistema físico do elemento e sobre a variável que controla seu ganho.

Tabela 6.1. Lista de elementos do sistema.

ID	ClassID	Comp	Desc	Ganho	Num	SisID	VarCtrl
1	nEleEsf	N	FonteTensao	vref	0	nEle	
2	nEleEsfC	N	Sensor	Ks	0	nEle	W(Corpo)
3	nEleRes	N	R	R	0	nEle	
4	nEleEsfC	N	Ctrl_P	Kp	0	nEle	v(R)
5	nEleRes	N	Ra	Ra	0	nEle	
6	nEleInd	N	La	La	0	nEle	
7	nAcpEleMcr	N	Acoplador	Kb;Km	0	nEleMcr	
8	nMcrCapRes	N	Corpo	J1;B1	0	nMcr	
9	nMcrRef	N	Ref		0	nMcr	
10	nEleVrt	S	Vertice1		0	nEle	
11	nEleVrt	S	Vertice2		0	nEle	
12	nEleVrt	S	Vertice3		0	nEle	

Tabela 6.2. Lista de conexões do sistema.

ID	Cnt1	Obj1 (ID)	Cnt2	Obj2 (ID)	SisID
13	2	1	2	2	nEle
14	1	1	1	3	nEle
15	2	3	1	2	nEle
16	1	4	1	5	nEle
17	2	5	1	6	nEle
18	2	6	1	10	nEle
19	1	10	1	7	nEle
20	2	7	1	11	nEle
21	1	11	1	12	nEle
22	1	12	2	4	nEle
23	3	7	1	8	nMcr
24	2	8	1	9	nMcr

A tabela 6.2 mostra os principais atributos das conexões. A conexão com ID

15, por exemplo, interliga o conector 2 do objeto com ID 3 ao conector 1 do objeto com ID 2. Ou seja, essa conexão interliga o conector negativo do resistor R ao conector positivo do Sensor. Os conectores ímpares se referem normalmente aos referenciais positivos e os pares, aos negativos.

A definição da estrutura de dados do sistema é realizada pelos métodos `AddElemento` e `AddConexao` da classe `TSistema`, conforme mostra o diagrama de colaboração da figura 4.15. Para o sistema da figura 6.2, a estrutura de dados definida é representada por um objeto da classe `TSistema`, que sempre agrega um objeto da classe `TListElemento` e outro da classe `TListConexao`, conforme mostra o diagrama da figura 4.10. Os atributos `Elementos` e `Conexoes` do objeto `TSistema` agregam neste exemplo doze objetos da classe `TElemento`, cujos atributos são listados na tabela 6.1, e doze objetos da classe `TConexao`, cujos atributos estão na tabela 6.2, respectivamente.

As etapas seguintes do algoritmo, controladas pelo método `SistemaToGrafo` da classe `TSistema`, são baseadas na estrutura de dados definida para o diagrama.

6.2.2.2. Remoção de Vértices Redundantes

Inicialmente, a estrutura de dados do sistema passa por duas modificações para padronizar sua representação. Na primeira, são removidos todos os vértices redundantes, ou seja, os vértices do sistema interligados a outros vértices. O pseudocódigo abaixo ilustra esta operação:

```

i ← 0;
Enquanto i < Conexoes.Count faça
    Cnx ← Conexoes[i];
    Obj1 ← Cnx.Obj1;    Obj2 ← Cnx.Obj2;
    Cnt1 ← Cnx.Cnt1;    Cnt2 ← Cnx.Cnt2;
    Se not(Obj1.Comp) e not(Obj2.Comp) então
        Conexoes.SubstituaElemento(Obj2, Cnt2, Obj1, Cnt1);
        Elementos.DeleteElemento(Obj2);
        Conexoes.DeleteConexao(Cnx);
    Senão Incrementa(i);
Fim Se
Fim Enquanto

```

Em síntese, é estabelecido um laço para varrer sequencialmente a lista de conexões do sistema, dada por *Conexoes*. O atributo *Count* retorna o número de itens desta lista que é indexada a partir de 0. Para cada conexão do sistema, dada por *Cnx*, são recuperados os objetos interligados por ela (*Obj1* e *Obj2*) e seus respectivos conectores (*Cnt1* e *Cnt2*). Se ambos objetos forem vértices, o método *SubstituaElemento* da lista de conexões é executado. Este método varre a lista e redireciona todas as conexões feitas ao conector *Cnt2* do objeto *Obj2* para o conector *Cnt1* do objeto *Obj1*. Em seguida, o objeto *Obj2* e a conexão *Cnx* são excluídos, através dos métodos *DeleteElemento* e *DeleteConexao*, respectivamente. Aplicando a modificação ao exemplo da figura 6.2, a conexão 22 passa a interligar os objetos 4 e 11 e são removidos o vértice 12 e a conexão 21.

6.2.2.3. Adição de Vértices Ocultos

A segunda modificação da estrutura de dados é realizada para que todas as conexões conectem um componente a um vértice. Ou seja, a interligação entre dois dispositivos físicos deve ser realizada por um vértice e duas conexões: a primeira conexão interliga o primeiro dispositivo ao vértice e a segunda, o vértice ao segundo dispositivo. O pseudocódigo abaixo ilustra esta operação:

```

i ← 0;
MaxID ← RetorneMaiorID() + 1;
Enquanto i < Conexoes.Count faça
    Cnx ← Conexoes[i];
    Obj1 ← Cnx.Obj1;    Obj2 ← Cnx.Obj2;
    Se Obj1.Comp e Obj2.Comp então
        Cnt1 ← Cnx.Cnt1;    Cnt2 ← Cnx.Cnt2;    SisID ← Cnx.SisID;
        Obj3 ← Elementos.AdicioneVertice(MaxID, SisID);
        Conexoes.DeleteConexao(Cnx);
        Conexoes.AdicioneConexao(MaxID+1, Obj1, Cnt1, Obj3, 1);
        Conexoes.AdicioneConexao(MaxID+2, Obj2, Cnt2, Obj3, 1);
        Incremente(MaxID, 3);
    Senão Incremente(i);
Fim Se
Fim Enquanto

```

Em resumo, o maior ID do sistema, dado por *MaxID*, é obtido. Logo após,

cada conexão do sistema, dada por Cnx, é analisada. Se os objetos Obj1 e Obj2 interligados pela conexão forem componentes, um novo vértice (Obj3) do mesmo sistema físico da conexão (SisID) é adicionado ao sistema. Então, a conexão Cnx é excluída e duas novas conexões são adicionadas, interligando os objetos Obj1 e Obj2 ao novo vértice. A variável MaxID controla o ID dos novos elementos.

A figura 6.3 mostra o desenho gráfico equivalente à estrutura de dados obtida após a realização das duas modificações apresentadas, tomando como exemplo o diagrama do motor DC da figura 6.2.

Neste ponto, as conexões do sistema estão padronizadas de forma que uma conexão sempre conecta um componente a um vértice. A primeira consequência desta padronização é que todos os vértices elétricos passam a representar pontos do sistema com tensões diferentes. Por conseguinte, todos os vértices elétricos são vértices do grafo de ligação. Os vértices mecânicos (e hidráulicos em outros exemplos) incluídos entre os dispositivos físicos, conforme visto na figura 6.3, vão ser utilizados no algoritmo que determina os demais vértices deste grafo.

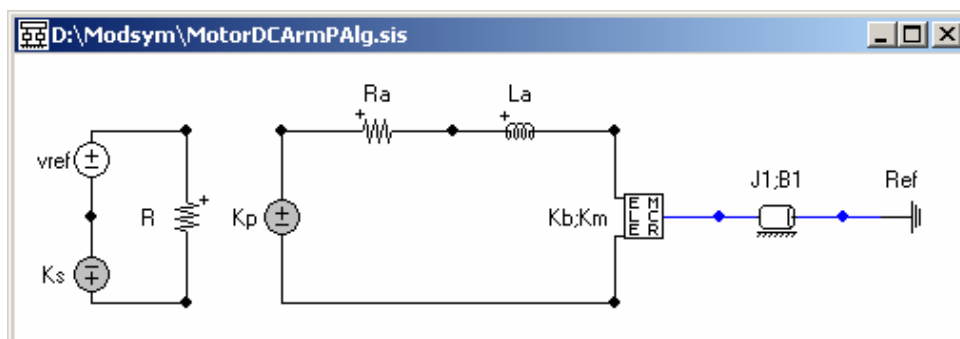


Figura 6.3. Motor DC em malha fechada com controlador Proporcional.

6.2.2.4. Numeração dos Vértices Elétricos

Com a estrutura de dados padronizada, o algoritmo Sistema-Grafo inicia a determinação dos vértices que devem compor o grafo de ligação. No primeiro instante, os vértices são identificados e numerados seqüencialmente, antes da estrutura do grafo de ligação ser definida. Conforme dito, todos os vértices elétricos incluídos na estrutura do sistema originam vértices no grafo de ligação.

O pseudocódigo a seguir ilustra a etapa de numeração dos vértices elétricos. Em síntese, é estabelecido um laço para varrer a lista de elementos do sistema que

possui Count itens indexados a partir de 0. O identificador da classe, dado pelo atributo ClassID, de cada objeto da lista (Obj) é testado. Se o elemento for um vértice elétrico (ClassID = nEleVrt), o atributo Num recebe o valor de n, que é utilizado para numerar os vértices do grafo de ligação. Após a conclusão do laço, o número de vértices elétricos numerados para o grafo é armazenado no atributo FNEleVrt do objeto TSistema. No exemplo da figura 6.3, são identificados e numerados os sete primeiros vértices do grafo.

```

n ← 0;
Para i de 0 até Elementos.Count-1 faça
    Obj ← Elementos[i];
    Se Obj.ClassID = nEleVrt então
        Incremente(n);
        Obj.Num ← n;
    Fim Se
Fim Para
FNEleVrt ← n;

```

6.2.2.5. Numeração dos Vértices Mecânicos Translacionais

Em seguida, o algoritmo analisa os elementos mecânicos translacionais para identificar os vértices do grafo de ligação referentes a este sistema físico. O processamento para este domínio é mais complexo porque tanto as massas, quanto as referências e ainda os vértices mecânicos podem originar vértices no grafo. O processamento desta etapa pode ser didaticamente dividido em três passos. O pseudocódigo a seguir ilustra o primeiro passo da operação:

```

Ref ← Falso; n ← 1;
base ← FNEleVrt;
FMctRef ← base + n;
Para i de 0 até Elementos.Count-1 faça
    Obj ← Elementos[i];
    Se Obj.ClassID = nMctRef então
        Obj.Num ← FMctRef;
        Vrt1 ← Obj.Obj[1];
        Vrt1.Num ← FMctRef;
        Ref ← Verdadeiro;
    Fim Se
Fim Para

```

```

Se not(Ref) então
    Incremente(MaxID);
    Obj ← Elementos.AdicioneElemento(MaxID, nMct, nMctRef);
    Obj.Num ← FMctRef;
Fim Se

```

Em síntese, é feita uma busca pelo elemento que representa a referência mecânica translacional do sistema. O flag Ref identifica se a busca obteve ou não sucesso. A variável n conta o número de vértices mecânicos incluídos no grafo; a variável base possui o número de vértices incluídos no passo anterior. O atributo FMctRef identifica o número do vértice do grafo de ligação que representa a referência mecânica.

Após as atribuições iniciais, é estabelecido um laço para varrer seqüencialmente a lista de elementos do sistema. O identificador da classe de cada elemento é novamente testado. Se o elemento for uma referência mecânica translacional (ClassID = nMctRef), seu atributo Num recebe o valor FMctRef. Este valor também é passado para o atributo Num do vértice conectado à referência, dado por Vrt1. No fim do laço, todas as referências mecânicas translacionais, se existirem, assim como os vértices conectados a elas, estão associadas ao mesmo vértice do grafo de ligação. Caso nenhuma referência seja encontrada, o método AdicioneElemento é executado para inserir uma referência mecânica translacional e seu atributo Num recebe o valor FMctRef.

No segundo passo da numeração dos vértices mecânicos translacionais, todas as massas do sistema são analisadas. Em geral, elas definem os outros pontos de diferentes velocidades em um sistema mecânico, mas é necessário analisar possíveis associações entre estes elementos no sistema. Como as conexões são ideais, a associação entre duas massas, por exemplo, define apenas uma velocidade.

O pseudocódigo apresentado a seguir ilustra a numeração das massas. Primeiramente, é estabelecido um laço para percorrer a lista de elementos do sistema. A condição inicial avaliada refere-se ao identificador da classe de cada elemento. Conforme visto na tabela 5.2, as massas do sistema podem ser representadas por elementos Massa e Massa + Atrito, cujos atributos ClassID são

respectivamente $nMctCap$ e $nMctCapRes$. Os vértices $Vrt1$ e $Vrt2$ conectados a cada massa (Obj) são então recuperados e quatro condições, identificadas por $C1$, $C2$, $C3$ e $C4$, são analisadas em seguida.

Para i de 0 até $Elementos.Count-1$ faça

$Obj \leftarrow Elementos[i]$;

Se $Obj.ClassID$ em $[nMctCap, nMctCapRes]$ então {Condição inicial}

$Vrt1 \leftarrow Obj.Obj[1]$;

$Vrt2 \leftarrow Obj.Obj[2]$;

Se $((Vrt1.Num = 0) \text{ e } (Vrt2.Num = 0))$ ou

$((Vrt1.Num = 0) \text{ e } (Vrt2.Num = FMctRef))$ ou

$((Vrt2.Num = 0) \text{ e } (Vrt1.Num = FMctRef))$ então { $C1$ }

$Incrementa(n)$;

$Obj.Num \leftarrow base + n$;

Se $Vrt1.Num = 0$ então $Vrt1.Num \leftarrow Obj.Num$; Fim Se;

Se $Vrt2.Num = 0$ então $Vrt2.Num \leftarrow Obj.Num$; Fim Se;

Senão

Se $((Vrt1.Num \text{ em } [0, FMctRef]) \text{ e } (Vrt2.Num \neq FMctRef))$ ou

$((Vrt2.Num \text{ em } [0, FMctRef]) \text{ e } (Vrt1.Num \neq FMctRef))$ então { $C2$ }

Se $Vrt1.Num \text{ em } [0, FMctRef]$ então $numx \leftarrow Vrt2.Num$

Senão $numx \leftarrow Vrt1.Num$;

Fim Se;

$Obj.Num \leftarrow numx$;

Se $Vrt1.Num = 0$ então $Vrt1.Num \leftarrow numx$; Fim Se;

Se $Vrt2.Num = 0$ então $Vrt2.Num \leftarrow numx$; Fim Se;

Senão

Se $Vrt1.Num = Vrt2.Num$ então { $C3$ }

$Obj.Num \leftarrow Vrt1.Num$;

Senão { $C4$ }

Se $Vrt1.Num > Vrt2.Num$ então

$numx \leftarrow Vrt1.Num$;

$numy \leftarrow Vrt2.Num$;

Senão

$numx \leftarrow Vrt2.Num$;

$numy \leftarrow Vrt1.Num$;

Fim Se

$Obj.Num \leftarrow numy$;

Para j de 0 até $Elementos.Count-1$ faça

Se $Elementos[j].Num = numx$ então

$Elementos[j].Num \leftarrow numy$;

Fim Se

Fim Para

Fim Se { $C3$ }

Fim Se { C2 }
Fim Se { C1 }
Fim Se { Condição inicial }
Fim Para

Na condição C1, um dos vértices conectados à massa possui o atributo Num igual a zero e o outro, igual a zero ou ao número da referência. Isto significa que a velocidade desta massa não foi definida ainda; logo, um novo vértice do grafo de ligação é numerado. Os atributos Num da massa e dos vértices não numerados, ou seja, com Num igual a zero, recebem o número do novo vértice do grafo.

A figura 6.4 ilustra esta condição. No lado esquerdo da figura, são apresentados os possíveis valores para os atributos Num das massas e dos vértices, antes da execução do algoritmo. À direita, são vistos os atributos após a numeração.

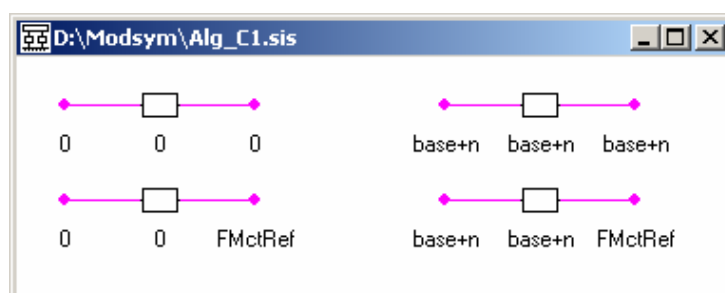


Figura 6.4. Ilustração da condição C1.

Na condição C2, um dos vértices conectados à massa possui o atributo Num igual a zero ou ao número da referência. O outro possui atributo Num diferente da referência e dado por numx. Isto significa que a velocidade desta massa é igual à velocidade de outra massa do sistema, pois a numeração de um dos vértices conectados à massa indica a velocidade a qual ela está submetida, excetuando-se a referência. Neste caso, os atributos Num da massa e do vértice não numerado recebem o atributo Num do outro vértice. A figura 6.5 ilustra esta condição.

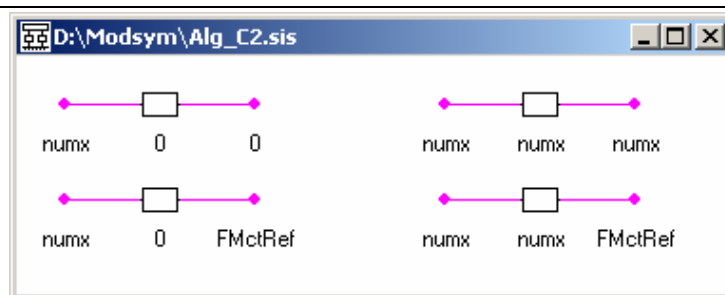


Figura 6.5. Ilustração da condição C2.

Na condição C3, os vértices conectados à massa possuem o mesmo valor para o atributo Num. Neste caso, a velocidade da massa também já está definida e seu atributo Num recebe este valor. A figura 6.6 ilustra esta condição.

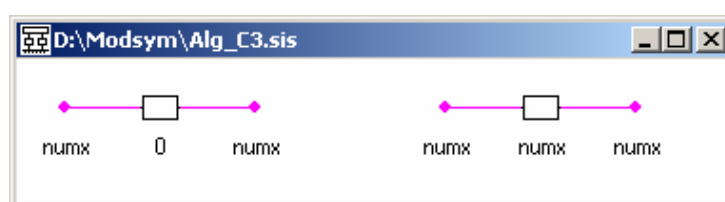


Figura 6.6. Ilustração da condição C3.

Na condição C4, os vértices conectados à massa possuem valores diferentes para o atributo Num, indicados por numx e numy. Isto significa que duas velocidades estão atribuídas à massa, o que evidentemente não é possível. Nestes casos, o atributo Num da massa recebe o valor numy, assim como todos os outros elementos que possuem Num igual a numx. A figura 6.7 ilustra esta condição.

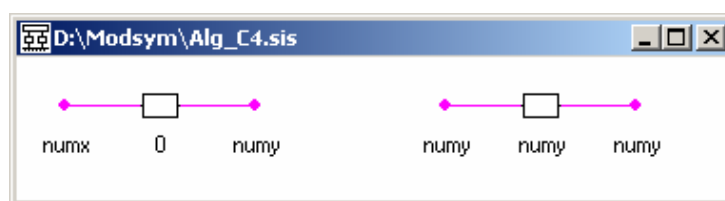


Figura 6.7. Ilustração da condição C4.

No terceiro e último passo da numeração dos vértices mecânicos, são analisados os vértices que interligam dois dispositivos mecânicos quaisquer, desde que não representem uma massa do sistema. Nestes casos, o vértice também define um ponto de velocidade e deve ser incluído no grafo de ligação. A identificação deles é bastante simples, uma vez que todos os vértices conectados a massas já estão numerados. O pseudocódigo apresentado a seguir ilustra essa operação.

```

Para  $i$  de 0 até  $\text{Elementos.Count}-1$  faça
     $\text{Obj} \leftarrow \text{Elementos}[i];$ 
    Se  $(\text{Obj.ClassID} = \text{nMctVrt})$  e  $(\text{Obj.Num} = 0)$  então
         $\text{Incremente}(n);$ 
         $\text{Obj.Num} \leftarrow \text{base} + n;$ 
         $\text{Obj.GrVrt} \leftarrow \text{True};$ 
    Fim Se
Fim Para
 $\text{FNMctVrt} \leftarrow n;$ 

```

Neste passo, a lista de elementos é percorrida outra vez e cada elemento é testado. Se ele for um vértice mecânico translacional ($\text{ClassID} = \text{nMctVrt}$) e não estiver numerado, o atributo Num recebe um novo valor de n , utilizado para numerar os vértices do grafo. O atributo GrVrt recebe o valor verdadeiro, indicando que o vértice deve ser incluído no grafo. Após a conclusão do laço, o número de vértices mecânicos translacionais numerados é armazenado no atributo FNMctVrt do objeto TSistema. Ele será utilizado para definir a base de numeração para os outros domínios físicos: mecânico rotacional e hidráulico.

6.2.2.6. Numeração dos Vértices Mecânicos Rotacionais

Na etapa seguinte, o algoritmo analisa os elementos mecânicos rotacionais. O processamento é idêntico ao realizado para o domínio mecânico translacional, considerando evidentemente os componentes e vértices rotacionais. Logo, três passos são realizados; o primeiro numera a referência mecânica rotacional, o segundo numera os componentes Inércia e Inércia + Atrito e o terceiro, os vértices rotacionais ainda não numerados. A variável base de numeração neste domínio é dada pela soma entre os números de vértices elétricos e mecânicos translacionais. Devido à analogia, o pseudocódigo desta etapa não será apresentado.

6.2.2.7. Numeração dos Vértices Hidráulicos

Na última etapa de numeração, o algoritmo analisa os elementos hidráulicos. O processamento é novamente idêntico, considerando os componentes deste domínio. Três passos são realizados: a numeração da referência hidráulica, dos reservatórios e dos demais vértices. A variável base da numeração para este

domínio é dada pela soma entre os vértices elétricos e mecânicos em geral, que totaliza a quantidade de vértices numerados até o momento. Devido à analogia, o pseudocódigo desta etapa também não será apresentado.

Finalmente, a figura 6.8 mostra o resultado final da numeração dos vértices para o exemplo do motor DC. É importante salientar que as etapas de numeração dos vértices de cada um dos domínios físicos somente são efetuadas se o sistema possuir algum elemento daquele domínio físico. Para o atual exemplo, apenas as numerações dos domínios elétrico e mecânico rotacional foram efetuadas.

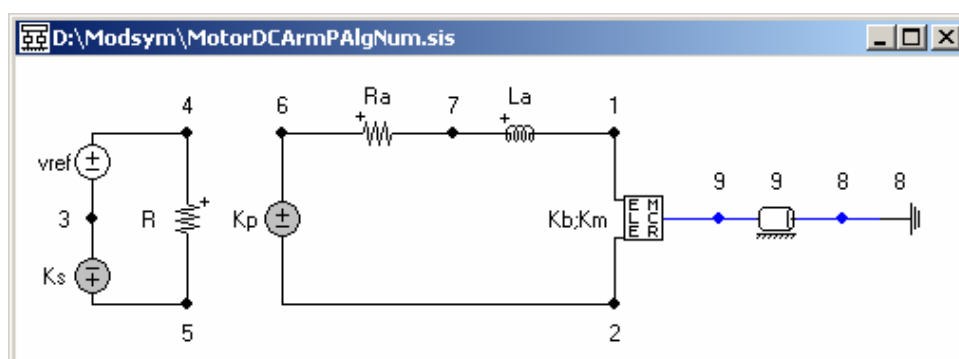


Figura 6.8. Numeração dos vértices do grafo.

6.2.2.8. Associações em Série e Paralelo

É fundamental verificar que associações em série e em paralelo de massas, inércias e reservatórios produzem o mesmo efeito físico. O algoritmo de numeração parte do pressuposto que as conexões são ideais, de forma que, ao conectar duas ou mais massas, submete-se os corpos à mesma velocidade, já que o esforço entre os componentes é nulo. O mesmo acontece para as inércias e os reservatórios, considerando, respectivamente, a velocidade angular e a pressão. A figura 6.9, mostra o resultado produzido pela numeração de massas em série e paralelo. Em ambos os casos, os corpos estarão conectados aos mesmos vértices no grafo.

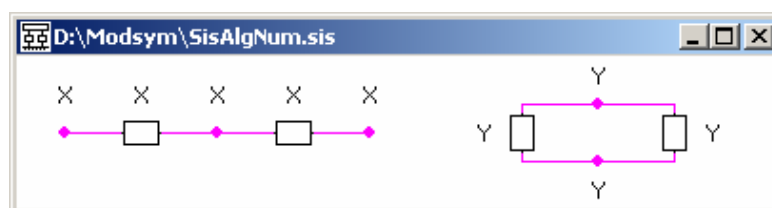


Figura 6.9. Associações de massas em série e paralelo.

6.2.2.9. Definição da Estrutura de Dados do Grafo de Ligação

Finalmente, após a etapa de numeração, o algoritmo Sistema-Grafo conclui o processo de obtenção do grafo de ligação do sistema. Nesta última etapa, é definida a estrutura de dados vista no diagrama de classes da figura 4.11, de forma que o sistema físico passa a ser representado também por um objeto da classe TGrafo. É nesta classe que o algoritmo que obtém o DFS do grafo é implementado.

Os vértices do grafo de ligação são obtidos a partir da numeração estabelecida nas etapas anteriores. Basicamente, eles são dados pelos vértices elétricos; pelas referências mecânicas e hidráulicas; pelas massas, inércias e reservatórios e pelos vértices com atributo GrfVrt verdadeiro. Como visto nas figuras 6.8 e 6.9, é possível que vários elementos do sistema recebam o mesmo valor para o atributo Num. A numeração realizada controla, então, a inserção dos vértices no grafo, estabelecendo que apenas um dentre os elementos com números idênticos origina um vértice no grafo de ligação.

Os ramos do grafo são obtidos a partir dos elementos físicos. Para as massas, inércias e reservatórios, os vértices terminais do ramo que representa o elemento são dados pelo seu atributo Num e pela referência do sistema físico ao qual o elemento pertence. Para os demais elementos, os vértices terminais são dados pelo atributo Num dos vértices do diagrama de desenho gráfico aos quais o elemento físico está conectado. É importante lembrar que as portas mecânicas e hidráulicas dos transformadores e acopladores possuem uma conexão implícita com a referência. A orientação e o ganho desses ramos são discutidos em [12].

A figura 6.10 mostra o diagrama equivalente à estrutura de dados obtida para exemplo do motor DC. Conforme visto na figura, todos os vértices elétricos originaram vértices no grafo. A referência mecânica definiu o vértice com ID 8 e o corpo, o vértice com ID 9. Por fim, os ramos do grafo foram representados por elementos generalizados, vistos na tabela 5.6. É fácil perceber que os vértices terminais dos elementos possuem o ID idêntico ao atributo Num do vértice equivalente no diagrama visto na figura 6.8. O ganho dos ramos está relacionado às relações constitutivas dos elementos apresentadas nas tabelas 5.1 a 5.6.

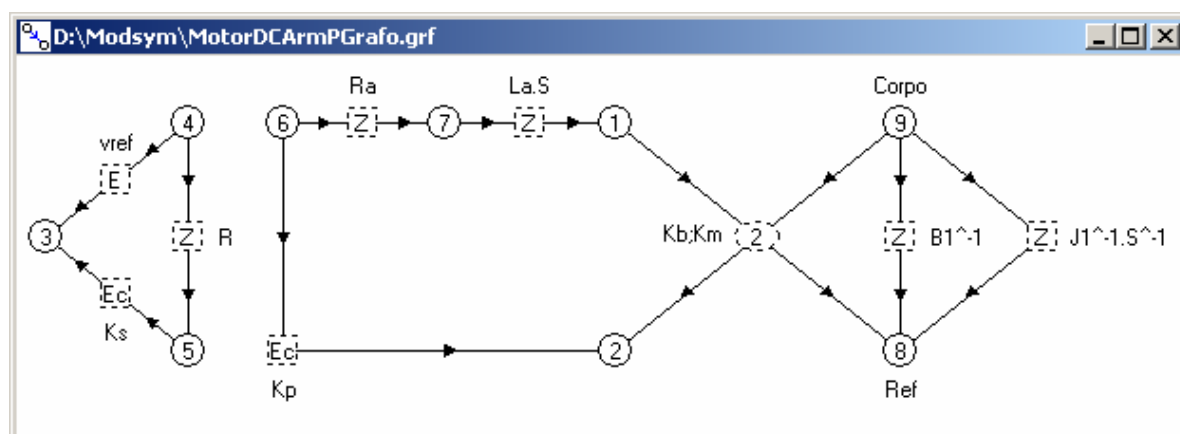


Figura 6.10. Grafo de ligação do Motor DC.

Apesar do algoritmo que obtém o grafo de ligação de um sistema ser encontrado na literatura, [12], o processo de identificação dos vértices deste grafo precisou ser sistematizado para o ambiente computacional proposto. De fato, como o usuário do software tem a possibilidade de modelar uma gama muito grande de sistemas físicos, tornou-se necessário estabelecer uma lógica que definisse corretamente os vértices do diagrama, principalmente no tocante as associações entre os elementos massa, inércia e reservatório, discutidos anteriormente.

6.3. Algoritmo Grafo – DFS

O algoritmo responsável pela obtenção do diagrama de fluxo de sinal do sistema físico a partir do grafo de ligação é apresentado em seguida. Para este algoritmo, será utilizada a denominação algoritmo Grafo-DFS.

6.3.1. Objetivo

O objetivo do algoritmo Grafo-DFS é definir a estrutura de dados do diagrama de fluxo de sinal do sistema físico baseado na estrutura de dados do grafo de ligação do sistema. Ambas as estruturas manipuladas no algoritmo são objetos da classe TGrafo, vista no diagrama de classes da figura 4.11.

É válido lembrar também que a estrutura do grafo de ligação pode ser definida de duas formas. Primeiro, ela pode ser obtida pelo algoritmo Sistema-Grafo, apresentado anteriormente. Neste caso, o usuário modela o sistema a partir

do seu desenho gráfico e a estrutura de dados é obtida durante o processo de cálculo da função de transferência, conforme mostra o diagrama de colaboração da figura 4.16. Segundo, a estrutura de dados do grafo pode ser obtida a partir de um diagrama de grafo de ligação definido pelo usuário, como mostra a colaboração da figura 4.17.

Em ambos os casos, o algoritmo Grafo-DFS executa previamente uma checagem da estrutura do grafo, realizada pelo método CheckGrafo, para verificar os pré-requisitos necessários à sua execução. O processo de obtenção do DFS é então executado pelo método GrafoToDFS, que controla, passo a passo, a “conversão” entre as duas estruturas. Os contextos nos quais o algoritmo está inserido são vistos nos diagramas 4.16 a 4.18.

6.3.2. Etapas do Algoritmo

O diagrama de atividades da figura 6.11 mostra as etapas do algoritmo Grafo-DFS, partindo desde a definição da estrutura de dados do grafo até a obtenção da estrutura de dados do diagrama de fluxo de sinal.

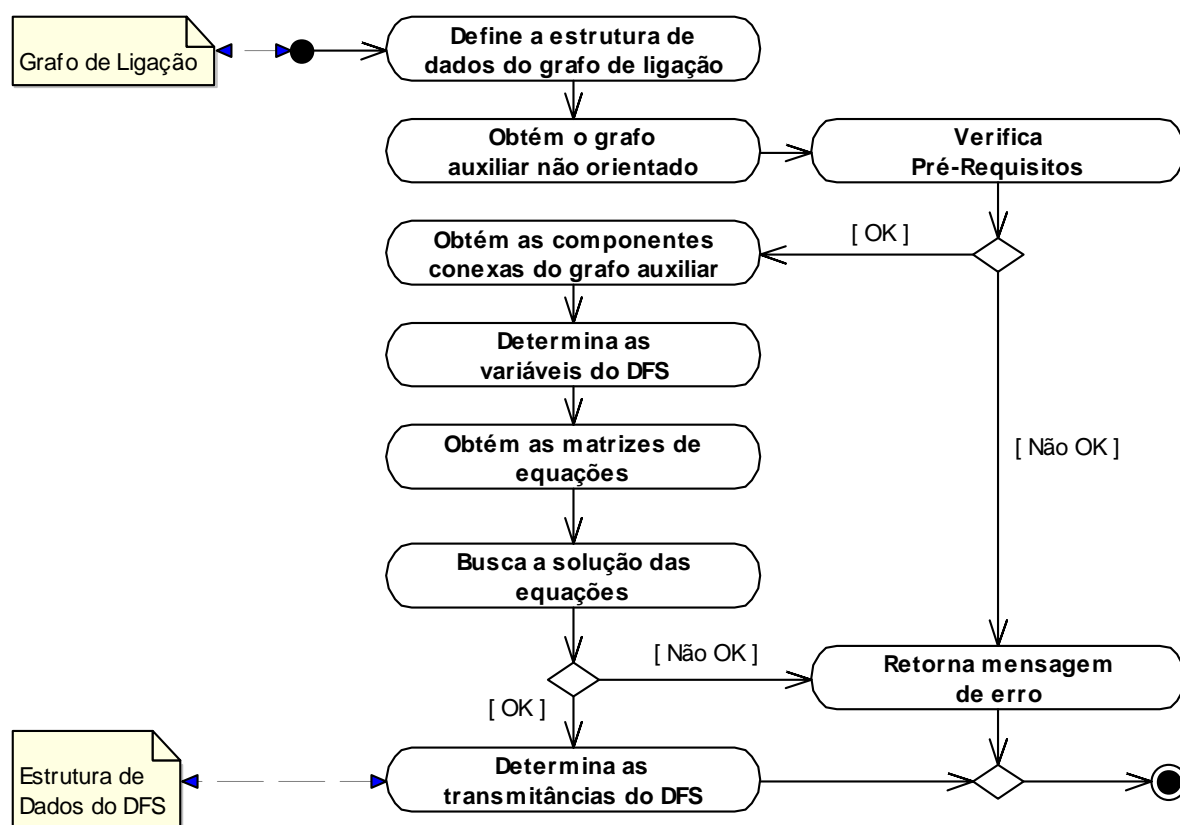


Figura 6.11. Diagrama de atividades do algoritmo Grafo-DFS.

Cada etapa do algoritmo é explorada em seguida e o processamento é exemplificado com base no grafo de ligação do Motor DC visto na figura 6.10.

6.3.2.1. Definição da Estrutura de Dados do Grafo

Conforme visto no diagrama de classes da figura 4.11, os grafos são mapeados em estruturas de dados representadas pela classe TGrafo. Um objeto TGrafo mantém pelo menos duas listas de objetos: a lista de vértices e a lista de ramos, utilizadas para representar os elementos do grafo. Uma terceira lista de objetos é eventualmente usada na representação da lista de símbolos utilizada pela regra de Mason. Os vértices e os ramos do grafo são objetos da classe TVertice e TRamo, respectivamente, cujos atributos relevantes ao algoritmo Grafo-DFS podem ser vistos nas tabelas 4.5 e 4.6.

De forma análoga à estrutura de dados do sistema, todos os vértices e ramos do grafo são identificados por um ID que é único em cada uma das listas. Na tabela 6.3, por exemplo, os atributos dos vértices que compõem o grafo de ligação do motor DC, visto na figura 6.10, podem ser verificados. A descrição dos vértices não está relacionada ao ID e é irrelevante ao algoritmo. A lista de vértices para o exemplo pode ser vista também na figura 5.17.

Tabela 6.3. Lista de vértices do grafo.

ID	Desc
1	Vertice1
2	Vertice2
3	_Vertice1
4	_Vertice2
5	_Vertice3
6	_Vertice4
7	_Vertice5
8	Ref
9	Corpo

A tabela 6.4 mostra os ramos do grafo do motor DC. O atributo ClassID está

associado à classe do elemento generalizado no grafo de ligação e o atributo Desc contém uma descrição para ele. Os atributos VertOrig e VertDest definem os vértices terminais do ramo. Os atributos Ganho e GrfVarCtrl armazenam, respectivamente, o ganho do elemento e a variável de esforço ou fluxo que o controla, se houver. Finalmente, o atributo GrfRamoAcp é utilizado para acoplar dois ramos do grafo. O acoplamento ocorre nos elementos de duas portas, onde é necessário que a primeira porta do elemento identifique a segunda e vice-versa. A lista de ramos para este grafo pode ser vista também na figura 5.18.

As listas de vértices e ramos de um grafo são sempre definidas através dos métodos AddVertice e AddRamo da classe TGrafo.

Tabela 6.4. Lista de ramos do grafo.

ID	ClassID	Desc	Vert Orig(ID)	Vert Dest(ID)	Ganho	Grf VarCtrl	Grf RamoAcp
1	nGrfEsf	Fonte	4	3	vref		
2	nGrfImp	R	4	5	R		
3	nGrfImp	Ra	6	7	Ra		
4	nGrfImp	La	7	1	La		
5	nGrfDpsP1	Acp_P1	1	2	$Kb^{-1};0;0;Km$		6
6	nGrfDpsP2	Acp_P2	9	8	$Kb^{-1};0;0;Km$		5
7	nGrfImp	Corpo_J	9	8	$J1^{-1}.S^{-1}$		
8	nGrfImp	Corpo_B	9	8	$B1^{-1}$		
9	nGrfEsfC	Sensor	5	3	Ks	E(Corpo_J)	
10	nGrfEsfC	Ctrl_P	6	2	Kp	E(R)	

6.3.2.2. Obtenção do Grafo Auxiliar Não-Orientado

Antes de iniciar o processo de obtenção do DFS, o algoritmo faz uma checagem na estrutura do grafo de ligação para verificar os pré-requisitos iniciais a sua execução. A verificação é realizada em um grafo auxiliar não orientado formado a partir de todos os vértices e ramos do grafo de ligação.

O pseudocódigo a seguir ilustra a obtenção do grafo auxiliar. Inicialmente,

um novo objeto TGrafo é alocado. O parâmetro False indica que o novo grafo, dado por Aux, é não orientado. Os dois laços, em seguida, inserem no grafo Aux todos os vértices e ramos do grafo principal. Os atributos Vertices e Ramos referem-se sempre à estrutura de dados do grafo de ligação quando não forem explicitamente referenciados.

```
Aux ← Novo TGrafo (False);
Para i de 0 até Vertices.Count-1 faça
    Aux.AddVertice(Vertices[i].ID, Vertices[i].Desc);
Fim Para
Para i de 0 até Ramos.Count-1 faça
    Aux.AddRamo(Ramos[i].ID, Ramos[i].ClassID, Ramos[i].Desc, Ramos[i].VertOrig.ID,
    Ramos[i].VertDest.ID);
Fim Para
```

6.3.2.3. Verificação dos Pré-Requisitos

Os pré-requisitos para execução do algoritmo são: primeiro, cada vértice do grafo auxiliar deve originar pelo menos um ciclo; segundo, o grafo auxiliar não deve conter autociclos, ou seja, ciclos formados por apenas um ramo. Estas condições são testadas para verificar se os elementos que compõem o sistema estão devidamente interligados. Em geral, a ausência de ciclos indica que existem componentes não conectados e a presença de autociclos indica a existência de elementos “em curto”. O pseudocódigo abaixo ilustra a checagem do grafo auxiliar.

```
Ok ← True;
Para i de 0 até Aux.Vertices.Count-1 faça
    V ← Aux.Vertices[i];
    Aux._LinAlderson(V.ID, V.ID, ListVC, ListRC, True);
    Se ListVC.Count = 0 então Ok ← False; Fim Se
Fim Para
Para i de 0 até Aux.Ramos.Count-1 do
    R ← Aux.Ramos[i];
    Orig ← R.VertOrig.ID;
    Dest ← R.VertDest.ID;
    Se Orig = Dest então Ok ← False; Fim Se
Fim Para
Retorne Ok;
```

Inicialmente, a variável Ok recebe o valor verdadeiro. Em seguida, a lista de

vértices do grafo auxiliar é percorrida. Para cada vértice V , é realizada uma busca de ciclos através do método `_LinAlderson`. O método obtém a lista de vértices de um ciclo iniciando em V , dada por `ListVC`. Se nenhum ciclo é encontrado para algum vértice, a quantidade de itens na lista `ListVC` é nula e a variável `Ok` recebe falso. O método `_LinAlderson` implementa o algoritmo de Lin-Alderson, apresentado em [15], que encontra todos os possíveis caminhos entre dois vértices em um grafo. Neste caso, o algoritmo é utilizado para encontrar, no máximo, um ciclo para cada vértice. Em seguida, a lista de ramos do grafo auxiliar é percorrida. Se os vértices terminais de algum ramo R são idênticos, o elemento físico representado pelo ramo está em curto e a variável `Ok` recebe falso. O resultado da checagem, dado pela variável `Ok`, é então retornado.

6.3.2.4. Obtenção das Componentes Conexas do Grafo Auxiliar

Se a checagem do grafo for bem sucedida, o processo de obtenção do DFS é de fato iniciado. No primeiro momento, o grafo auxiliar é dividido em subgrafos conexos, que são utilizados na obtenção das equações do sistema. O algoritmo que encontra as componentes conexas de um grafo é encontrado em [13].

No ambiente computacional, as componentes conexas são representadas por um objeto da classe `TListGrafo`, vista no diagrama da figura 4.11. O objeto mantém uma lista onde cada item é um grafo. Para o exemplo da figura 6.10, são obtidos três subgrafos conexos dados por G_1 , G_2 e G_3 , conforme visto na figura 6.12.

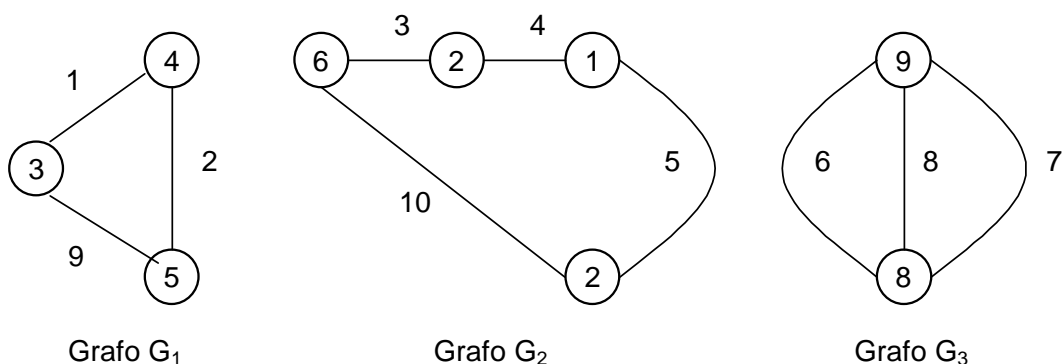


Figura 6.12. Componentes conexas do grafo de ligação do Motor DC.

6.3.2.5. Determinação das Variáveis do DFS

Após a obtenção das componentes conexas, a estrutura de dados do DFS começa a ser definida. É importante lembrar que o diagrama de fluxo de sinal obtido pelo algoritmo Grafo-DFS objetiva o cálculo de funções de transferência de um sistema físico, dadas pela equação 6.1:

$$G(S) = \frac{Y(S)}{U(S)} \quad (6.1)$$

onde: $Y(S)$ é a saída do sistema,

$U(S)$ é a entrada ou excitação.

Como a representação de um sistema físico por um DFS não é única, o algoritmo proposto constrói um diagrama de fluxo de sinal de modo que a função de transferência do sistema obtida a partir dele tenha as seguintes características:

- A entrada $U(S)$ da função deve ser dada pela variável generalizada de fluxo em fontes de fluxo ou pela variável generalizada de esforço em fontes de esforço;
- A saída $Y(S)$ da função deve ser dada pela variável generalizada de fluxo ou de esforço em qualquer elemento do sistema físico.

O estabelecimento destas características implica na construção de um DFS com as variáveis esforço e fluxo de todos os elementos constituintes do sistema físico. Desta forma, cada elemento físico origina duas ou quatro variáveis no diagrama, de acordo com o seu número de portas de energia. Ou seja, cada ramo no grafo de ligação, que representa uma porta, está relacionado a duas variáveis: o esforço entre os vértices terminais do ramo e o fluxo através dele.

A tabela 6.5 mostra algumas variáveis do algoritmo Grafo-DFS calculadas nesta etapa do processamento. Os valores apresentados referem-se ao exemplo do motor DC.

Tabela 6.5. Variáveis do algoritmo Grafo-DFS.

Variável	Descrição	Valor
n	Nº de vértices do grafo de ligação	9
b	Nº de ramos do grafo de ligação	10
c	Nº de componentes conexas	3

fe	Nº de fontes de energia	1
fc	Nº de fontes controladas	2
z	Nº de impedâncias generalizadas	5
t	Nº de elementos de duas portas	1

O pseudocódigo a seguir mostra a implementação realizada para obter as variáveis do diagrama e do algoritmo. Inicialmente, as variáveis *n*, *b* e *c* são obtidas a partir das listas de vértices, ramos e de componentes conexas, dada por *ccAux*. Logo após, a variável *DFS*, que representa o diagrama, é alocada como um grafo orientado. Os vetores de string **X**, **Y**, **Xe**, **Xs** são iniciados. **X** e **Y** armazenam as descrições das variáveis do *DFS*. **X** mantém apenas a descrição da variável, enquanto que **Y** atrela à descrição os vértices terminais do ramo. As variáveis **Xe** e **Xs** representam, respectivamente, as variáveis de entrada e de saída do *DFS*.

```

n ← Vertices.Count;  b ← Ramos.Count;  c ← ccAux.Count;
DFS ← Novo TGrafo (True);
X ← [ ];  Y ← [ ];
Xe ← [ ];  Xs ← [ ];
Para i de 0 até b−1 faça
    R ← Ramos[i];
    varID ← i + 1;
    varX ← varEsforco(R.Desc);
    varY ← varEsforco(R.Desc, R.VertOrig.ID, R.VertDest.ID);
    X.Add(varX);
    Y.Add(varY);
    DFS.AddVertice(varID, varX);
    Se R.ClassID = nGrfEsf então Xe.Add(varX) Senão Xs.Add(varX); Fim Se
    Caso R.ClassID seja
        nGrfEsf, nGrfFlx :  incremente (fe);
        nGrfEsfC, nGrfFlxC :  incremente (fc);
        nGrfImp :  incremente (z);
        nGrfGrdP1, nGrfTrfP1, nGrfDpsP1 : incremente (t);
    Fim Caso
Fim Para
Para i de 0 até b−1 faça
    R ← Ramos[i];
    varID ← b + i + 1;
    varX ← varFluxo(R.Desc);
    varY ← varFluxo(R.Desc, R.VertOrig.ID, R.VertDest.ID);
    X.Add(varX);

```

```

    Y.Add(varY);
    DFS.AddVertice(varID, varX);
    Se Ramos[I].ClassID = nGrfFlx então Xe.Add(varX) Senão Xs.Add(varX); Fim Se
Fim Para

```

Em seguida, a lista de ramos do grafo de ligação é percorrida para identificar as variáveis de esforço do DFS. O ID das variáveis é obtido a partir da numeração dos ramos do grafo e a descrição é gerada pela função varEsforco, que adiciona ou não dados dos vértices terminais do ramo. O método Add adiciona as variáveis nos vetores e o método AddVertice, no DFS. Logo após, o atributo ClassID associado a cada ramo é testado para identificar as variáveis de entrada e de saída. Finalmente, o laço contabiliza as variáveis fe, fc, z e t, que classificam os elementos generalizados do grafo de ligação. A lista de ramos é então percorrida pela segunda vez. Neste laço, são identificadas as variáveis de fluxo do DFS de forma análoga ao processamento realizado para as variáveis de esforço.

O número de ramos do grafo de ligação pode ser escrito em função do número de elementos generalizados, conforme mostra a equação 6.2:

$$b = fe + fc + z + 2.t \quad (6.2)$$

Conseqüentemente, o número de variáveis do DFS, dado por n_{DFS} , pode também ser obtido em função do número de elementos, conforme a equação 6.3:

$$n_{DFS} = 2.b = 2(fe + fc + z + 2.t) \quad (6.3)$$

Por fim, o número de variáveis de entrada e de saída, dados respectivamente por n_E e n_S , são dados pelas equações 6.4 e 6.5:

$$n_E = fe \quad (6.4)$$

$$n_S = fe + 2.fc + 2.z + 4.t \quad (6.5)$$

A equação 6.6 mostra o vetor de variáveis do DFS, dado por \mathbf{X} , para o exemplo do motor DC. A variável E(Fonte) que representa a tensão de referência aplicada pela fonte de esforço é a única variável de entrada para o exemplo. Todas as outras são variáveis de saída.

$$\begin{aligned}
\mathbf{X} = [& E(\text{Fonte}) \ E(\text{R}) \ E(\text{Ra}) \ E(\text{La}) \ E(\text{Acp_P1}) \ E(\text{Acp_P2}) \\
& E(\text{Corpo_J}) \ E(\text{Corpo_B}) \ E(\text{Sensor}) \ E(\text{Ctrl_P}) \\
& F(\text{Fonte}) \ F(\text{R}) \ E(\text{Ra}) \ F(\text{La}) \ F(\text{Acp_P1}) \ F(\text{Acp_P2}) \\
& F(\text{Corpo_J}) \ F(\text{Corpo_B}) \ F(\text{Sensor}) \ F(\text{Ctrl_P})]^T
\end{aligned} \tag{6.6}$$

onde: $E(\text{elemento})$ é a variável generalizada de esforço no elemento;

$F(\text{elemento})$ é a variável generalizada de fluxo no elemento.

6.3.2.6. Obtenção das Matrizes de Equações

Uma vez definidas as variáveis do DFS, a tarefa seguinte do algoritmo é obter as transmitâncias do diagrama, que representam as relações entre as variáveis do sistema físico. Este relacionamento é descrito por um sistema de equações lineares que expressa algebricamente as propriedades constitutivas dos elementos físicos e as relações de interconexão entre eles. O sistema de equações que descreve o comportamento do sistema físico representado pelo grafo de ligação pode ser sistematicamente obtido a partir de cinco matrizes:

- \mathbf{A}_1 = Matriz de impedâncias
- \mathbf{A}_2 = Matriz de fontes controladas
- \mathbf{A}_3 = Matriz de transformação e giração
- \mathbf{A}_4 = Matriz de compatibilidade de esforço
- \mathbf{A}_5 = Matriz de continuidade de fluxo

Todas essas matrizes satisfazem a relação matricial dada pela equação 6.7, com i variando no intervalo $[1, 5]$:

$$\mathbf{A}_i \cdot \mathbf{X} = 0 \tag{6.7}$$

Os passos necessários à obtenção de cada uma das matrizes, implementado pelo algoritmo Grafo-DFS, são apresentados em seguida.

A matriz de impedâncias \mathbf{A}_1 relaciona as variáveis de esforço e fluxo nas impedâncias generalizadas do grafo de ligação. O pseudocódigo a seguir ilustra a obtenção desta matriz:

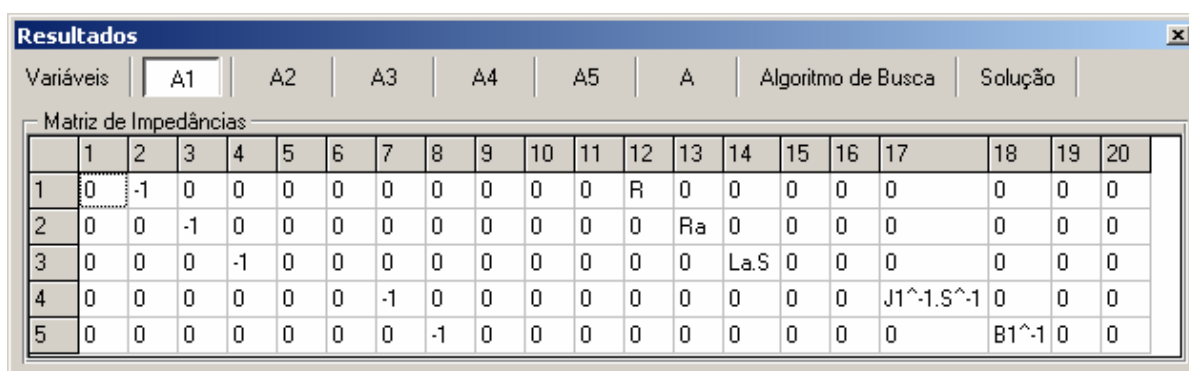

```

 $k \leftarrow 0$ ;
Para  $i$  de 0 até  $b-1$  faça
     $R \leftarrow \text{Ramos}[i]$ ;
    Se  $R.\text{ClassID} = n\text{GrfImp}$  então
         $VE \leftarrow \text{varEsforco}(R.\text{Desc})$ ;
         $VF \leftarrow \text{varFluxo}(R.\text{Desc})$ ;
         $jVE \leftarrow X.\text{IndexOf}(VE)$ ;
         $jVF \leftarrow X.\text{IndexOf}(VF)$ ;
         $A1[k, jVE] \leftarrow '-1'$ ;
         $A1[k, jVF] \leftarrow R.\text{Ganho}$ ;
         $\text{Incremente}(k)$ ;
    Fim Se
Fim Para

```

Basicamente, a lista de ramos é percorrida e o atributo ClassID de cada ramo do grafo é testado. Se elemento for uma impedância, a variável VE recebe a variável de esforço e VF, a variável de fluxo. As posições de VE e VF no vetor **X**, retornadas pelo método IndexOf, determinam as colunas da matriz **A**₁ que se referem às variáveis, dadas por jVE e jVF. Por fim, os elementos de cada linha da matriz, indexada pela variável k, são definidos com base na relação constitutiva do elemento, dada pelo ganho do ramo.

A figura 6.13 mostra a matriz de impedâncias para o exemplo do motor DC. Aplicando a matriz à equação 6.7, é fácil verificar que, uma a uma, as linhas relacionam o esforço e o fluxo em todas as impedâncias do grafo de ligação.



Variáveis	A1	A2	A3	A4	A5	A	Algoritmo de Busca	Solução
Matriz de Impedâncias								
1	0	-1	0	0	0	0	0	0
2	0	0	-1	0	0	0	0	0
3	0	0	0	-1	0	0	0	0
4	0	0	0	0	0	-1	0	0
5	0	0	0	0	0	0	-1	0

Figura 6.13. Matriz de impedâncias para o exemplo do Motor DC.

A matriz de fontes controladas **A**₂ relaciona as variáveis de esforço ou fluxo das fontes controladas de energia com as outras variáveis do sistema usadas para controlar a fonte. O pseudocódigo a seguir ilustra a desta matriz:

```

k ← 0;
Para i de 0 até b−1 faça
    R ← Ramos[i];
    Se R.ClassID em [nGrfEsfC, nGrfFlxC] então
        Se R.ClassID = nGrfEsfC então Vfc ← varEsforco(R.Desc);
        Senão Vfc ← varFluxo(R.Desc); Fim Se
        Vctrl ← R.GrVarCtrl;
        jVfc ← X.IndexOf(Vfc);
        jVctrl ← X.IndexOf(Vctrl);
        A2[k, jVfc] ← '−1';
        A2[k, jVctrl] ← R.Ganho;
        Incremente(k);
    Fim Se
Fim Para

```

O processamento é bastante análogo ao passo anterior. Em resumo, a lista de ramos é percorrida em busca das fontes controladas de energia. A variável de cada um das fontes, dada por *Vfc*, que é o esforço ou o fluxo na fonte controlada é relacionada com a variável que a controla, dada por *Vctrl*. A figura 6.14 mostra a matriz de fontes controladas para o exemplo do motor DC.

Resultados

Variáveis	A1	A2	A3	A4	A5	A	Algoritmo de Busca	Solução												
Matriz de Fontes Controladas																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	0	0	0	0	Ks	0	-1	0	0	0	0	0	0	0	0	0	0	0
2	0	Kp	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0

Figura 6.14. Matriz de fontes controladas para o exemplo do Motor DC.

A matriz de transformação e rotação **A₃** relaciona as variáveis de esforço e fluxo nos elementos de duas portas de energia. A obtenção desta matriz é ilustrada pelo pseudocódigo a seguir.

```

k ← 0;
Para i de 0 até b−1 faça
    R1 ← Ramos[i];
    Se R1.ClassID em [nGrfGrdP1, nGrfTrfP1, nGrfDpsP1] então
        R2 ← Ramos.GetRamo(R1.GrRamoAcp);
        jVE1 ← X.IndexOf( varEsforco(R1.Desc) );
        jVE2 ← X.IndexOf( varEsforco(R2.Desc) );
        jVF1 ← X.IndexOf( varFluxo(R1.Desc) );
        jVF2 ← X.IndexOf( varFluxo(R2.Desc) );

```

Caso R1.ClassID seja

```

nGrfGrdP1:  A3[2*k, jVE2] ← '-1';
              A3[2*k, jVF1] ← R1.Ganho;
              A3[2*k+1, jVE1] ← '1';
              A3[2*k+1, jVF2] ← R1.Ganho;

nGrfTrfP1:  A3[2*k, jVE1] ← '-1';
              A3[2*k, jVE2] ← R1.Ganho;
              A3[2*k+1, jVF1] ← R1.Ganho;
              A3[2*k+1, jVF2] ← '1';

nGrfDpsP1:  A3[2*k, jVE1] ← R1.Ganho[1];
              A3[2*k, jVE2] ← '-1';
              A3[2*k, jVF1] ← R1.Ganho[2];
              A3[2*k+1, jVE1] ← R1.Ganho[3];
              A3[2*k+1, jVF1] ← R1.Ganho[4];
              A3[2*k+1, jVF2] ← '1';

```

Fim Caso

Incremente(k);

Fim Se

Fim Para

O processamento é também análogo aos casos anteriores; entretanto, a lista de ramos é percorrida para encontrar apenas a primeira porta de energia dos elementos de duas portas. A variável R1 está relacionada à primeira porta de energia e R2, obtida a partir do atributo GrfRamoAcp de R1, à segunda. A relação entre as variáveis de esforço e fluxo nas portas é dada pela equação 2.16 e varia de acordo com o tipo do elemento: girador, transformador ou elemento genérico de duas portas. A figura 6.15 mostra a matriz A_3 para o exemplo do motor DC.

Resultados

Variáveis	A1	A2	A3	A4	A5	A	Algoritmo de Busca										Solução					
Matriz de Transformação e Giração																						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20		
1	0	0	0	0	Kb^-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Km	1	0	0	0	0		

Figura 6.15. Matriz de transformação e giração para o exemplo do Motor DC.

A matriz A_4 representa as restrições de compatibilidade de esforço do grafo, [12]. Essas restrições são generalizações da lei das malhas de Kirchhoff que mostra que o somatório dos esforços em qualquer laço do grafo de ligação é igual a zero. A obtenção da matriz é realizada por algoritmos que utilizam os conceitos de árvore

geradora, co-árvore e laço. Os laço do grafo são encontrados quando as cordas de uma co-árvore são incluídas, uma a uma, em uma árvore geradora, [13].

O pseudocódigo a seguir ilustra a obtenção da matriz. Inicialmente, o algoritmo que encontra a árvore geradora de um grafo, descrito em [13], é executado para o grafo auxiliar. O algoritmo retorna uma árvore geradora para este grafo, dada por *ArvGer*, e a lista de ramos removidos dele para obtenção da árvore, dada por *Cordas*. A lista *Cordas* é então percorrida. Cada corda é incluída no grafo *Aux2*, que é uma cópia de *ArvGer*, gerando um laço neste grafo. O algoritmo de Lin-Alderson retorna as listas de vértices e ramos deste laço, dadas por *ListVC* e *ListRC*, que são utilizadas para gerar os vetores **YP** e **YN**. **YP** lista as variáveis de esforço associadas ao laço no sentido positivo e **YN**, no sentido negativo. O laço final busca cada variável de **YP** e **YN** no vetor **Y**, que mantém a lista das variáveis de esforço do grafo de ligação, determinando se cada ramo do laço foi percorrido no sentido positivo ou negativo da variável de esforço.

```

ArvGer ← Aux._ArvoreGeradora(Cordas);
Para i de 0 até Cordas.Count-1 faça
    R ← _GetRamo(Cordas[i]);
    Aux2 ← TGrafo.Init(ArvGer);
    Aux2.AddRamo(R.ID, R.VertOrig.ID, R.VertDest.ID);
    ViID ← R.VertOrig.ID;
    Aux2._LinAlderson(ViID, ViID, ListVC, ListRC, True);
    YP ← [ ];
    YN ← [ ];
    Para j de 0 até ListVC.Count-2 faça
        Desc ← Aux2._GetRamo(ListRC[j]).Desc;
        YP.Add( varEsforco(Desc, ListVC[j], ListVC[j+1]) );
        YN.Add( varEsforco(Desc, ListVC[j+1], ListVC[j]) );
    Fim Para
    Para k de 0 até YP.Count-1 faça
        j ← Y.IndexOf(YP[k]);
        Se j <> -1 então A4[i, j] ← '1';
        Senão j ← Y.IndexOf(YN[k]);
        A4[i, j] ← '-1';
    Fim Se
Fim Para
Fim Para

```

A figura 6.16 mostra a matriz A_4 para o exemplo do motor DC.

Resultados

Variáveis	A1	A2	A3	A4	A5	A	Algoritmo de Busca	Solução												
Matriz de Compatibilidade de Esforço																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	-1	-1	-1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
2	-1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0

Figura 6.16. Matriz de compatibilidade de esforço para o exemplo do Motor DC.

A matriz A_5 representa as restrições de continuidade de fluxo do grafo de ligação, [12]. Essas restrições são generalizações da lei dos nós de Kirchoff que mostra que o somatório dos fluxos em qualquer vértice do grafo é igual a zero. A matriz A_5 é obtida a partir dos subgrafos conexos do grafo de ligação, analisando os ramos adjacentes em cada vértice dos subgrafo. Para garantir a independência linear das equações, um vértice em cada componente conexa é descartado. O pseudocódigo abaixo ilustra a obtenção da matriz.

```

kA5 ← 0;
Para k de 0 até ccAux.Count-1 faça
    kccAux ← ccAux[k];
    kn ← kccAux.Vertices.Count;
    Para i de 0 to kn-1 faça
        ViID ← kccAux.Vertices[i].ID;
        Para j de 0 até b-1 faça
            R ← Ramos[j];
            Se R.VertOrig.ID = ViID então A5[i+kA5, b+j] ← '1'; Fim Se
            Se R.VertDest.ID = ViID então A5[i+kA5, b+j] ← '-1'; Fim Se
        Fim Para
    Fim Para
imax ← 0;
max ← 0;
Para j de 0 até b-1 faça
    Se A5[kA5, b+j] <> '0' então incremente(max); Fim Se
Fim Para
Para i de 1 até kn-1 faça
    qtd ← 0;
    Para j de 0 até b-1 faça
        Se A5[i+kA5, b+j] <> '0' então Incremente(qtd); Fim Se
    Fim Para

```

```

    Se  $qtd > max$  então
         $max \leftarrow qtd$ ;
         $imax \leftarrow i$ ;
    Fim Se
Fim Para
A5.EqUsed[ $imax+kA5$ ]  $\leftarrow True$ ;
 $kA5 \leftarrow kA5 + kn$ ;
Fim Para

```

O primeiro laço estabelece uma repetição para cada componente conexa do grafo auxiliar, dada por $kccAux$. A variável kn recebe então o número de vértices da k -ésima componente; o ID de cada um deles, dado por $ViID$, é recuperado. A lista de ramos do grafo é percorrida em seguida; se o ramo for convergente no vértice $ViID$, o elemento da matriz referente a k -ésima componente, ao i -ésimo vértice a ao j -ésimo ramo recebe “1”; se for divergente, o elemento recebe “-1”.

Após analisar todos os vértices de uma componente, o algoritmo identifica a linha da matriz A_5 com o maior número de elementos não nulos. Esta linha é excluída logicamente através do atributo $EqUsed$ que recebe o valor verdadeiro. Conforme dito, um vértice em cada componente deve ser eliminado para garantir a independência linear da matriz. A eliminação é realizada para o vértice com maior número de ramos associados. O processo então se repete para as demais componentes do grafo.

A figura 6.17 mostra a matriz A_5 para o exemplo do motor DC.

Resultados

Variáveis

A1

A2

A3

A4

A5

A

Algoritmo de Busca

Solução

Matriz de Continuidade de Fluxo

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	-1
3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	1	0
7	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	-1	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	-1	-1	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0

Figura 6.17. Matriz de continuidade de fluxo para o exemplo do Motor DC.

Por fim, as matrizes são agrupadas na matriz **A** de forma que o sistema de equações lineares do sistema físico pode ser escrito conforme a equação 6.8:

$$\mathbf{A} \cdot \mathbf{X} = \begin{bmatrix} \mathbf{A}_4 \\ \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \\ \mathbf{A}_{5N} \end{bmatrix} \cdot \mathbf{X} = 0 \quad (6.8)$$

onde: **A**_{5N} é a matriz obtida a partir das linhas linearmente independentes de **A**₅.

A seqüência de agrupamento definida pela equação 6.8 foi determinada empiricamente de forma a gerar o menor esforço computacional pelo o algoritmo de busca apresentado em seguida. A figura 5.20 mostra a matriz de equações obtida para o exemplo do motor DC.

O número de equações da matriz **A**, dado por n_{EQ} na equação 6.9, pode ser obtido a partir do número de linhas das matrizes **A**₁, **A**₂, **A**₃, **A**₄ e **A**_{5N}, apresentado na tabela 6.6. Os valores listados na tabela referem-se ao exemplo do motor DC.

Tabela 6.6. Número de equações das matrizes.

Matriz	Número de Equações	Valor
A ₁	z	5
A ₂	fc	2
A ₃	2.t	2
A ₄	b–n+c	4
A _{5N}	n–c	6

$$n_{EQ} = (z) + (fc) + (2.t) + (b - n + c) + (n - c) \quad (6.9)$$

Substituindo o valor de b, dado pela equação 6.2, na equação 6.9, mostra-se que o número de equações n_{EQ} corresponde ao número de variáveis de saída do DFS, dado pela equação 6.5. Por isso, se o sistema de equações tiver solução é possível determinar os valores de cada uma das variáveis de saída do diagrama de fluxo de sinal, que são as incógnitas do sistema.

Normalmente, a obtenção de funções de transferência do sistema físico a

partir de um sistema de equações diferenciais envolve uma série de manipulações matriciais do sistema de equações e resulta no cálculo da inversa de uma matriz na forma simbólica. Além de complexo, o cálculo da inversa de matrizes simbólicas é pouco ilustrativo para o estudante da Engenharia de Controle, porque representa apenas uma manipulação matemática realizada nas equações do sistema.

O algoritmo Grafo-DFS propõe então a construção de um diagrama de fluxo de sinal a partir do sistema de equações do sistema físico de forma que o cálculo de funções de transferência do sistema pode ser realizado através da regra de Mason. Tanto a representação do sistema através de diagramas de fluxo quanto a regra de Mason são temas importantes para a formação educacional dos estudantes de sistemas de controle.

6.3.2.7. Algoritmo de Busca

No DFS construído pelo algoritmo, as variáveis de entrada, dadas pelo vetor **Xe** que lista as excitações do sistema físico, são representadas por nós de entrada (ver Seção 2.4.3). Nestas variáveis nenhum ramo pode ser convergente. As outras variáveis, dadas pelo vetor **Xs**, são representadas por nós mistos ou de saída, devendo haver pelo menos um ramo convergente em cada uma. A tarefa do algoritmo é encontrar para cada variável em **Xs** a relação entre ela e as outras variáveis do DFS. Cada linha da matriz **A**, que representa as equações do sistema, deve fornecer a relação para uma variável distinta do diagrama.

Com o objetivo de encontrar uma solução para o DFS, o algoritmo Grafo-DFS implementa uma busca recursiva e em profundidade testando todas as equações da matriz **A** e todas as variáveis referenciadas em cada equação. A finalidade da busca é encontrar qual a variável que deve ser calculada por cada uma das equações, sendo finalizada ao se encontrar a primeira solução viável para o problema.

O pseudocódigo a seguir ilustra o algoritmo de busca. No primeiro passo, as variáveis que controlam o processo são iniciadas. O vetor **VS**, inicialmente vazio, é utilizado para armazenar a seqüência de cálculo das variáveis do diagrama. A variável ArvB armazena a árvore de busca gerada pelo algoritmo e a variável FS

controla a finalização do processamento quando uma solução é encontrada.

```
// Passo 1: Início da busca
VS ← [ ];
FS ← False;
Para i de 0 até Xe.Count-1 faça
    Se i = 0 então N ← ArvB.AddNoRaiz (Xe[i])
    Senão N ← ArvB.AddNoFilho(N, Xe[i]);
    Fim Se
Fim Para
```

No segundo passo, a matriz **A** é analisada. O primeiro laço é utilizado para percorrer todas as linhas de **A**. Se a equação da i-ésima linha não foi utilizada até o momento, o segundo laço monta o vetor **LIndex**, armazenando o índice das colunas não nulas da linha. O terceiro laço, então, define o vetor **LV** que lista as variáveis calculáveis pela i-ésima linha. Não são incluídas neste vetor, as variáveis de entrada e as variáveis associadas a outras equações da matriz. Se nenhuma variável pode ser obtida pela i-ésima equação, o caminho estabelecido pela busca falhou e um *Back-Tracking* é realizado. Caso contrário, uma variável de **LV** é selecionada e incluída no vetor solução **VS**. Se a busca não foi concluída, o processamento é recursivamente executado e uma nova linha da matriz é analisada. A busca termina quando uma solução é encontrada ou quando todas as possibilidades de solução são esgotadas. Se algoritmo obtiver êxito, a variável FS recebe o valor verdadeiro.

```
// Passo 2: Busca recursiva
Para i de 0 até A.NLinhas-1 faça
    Se not(A.EqUsed[i]) então
        LIndex ← [ ];
        LV ← [ ];
        Para j de 0 até A.NColunas-1 faça
            Se A[i,j] <> '0' então LIndex.Add(j); Fim Se
        Fim Para
        Para j de 0 até LIndex.Count-1 faça
            V ← X[ LIndex[j] ];
            Se (Xe.IndexOf(V) = -1) e (VS.IndexOf(V) = -1) então
                LV.Add(V);
            Fim Se
        Se LV.Count = 0 então Realizar_Back_Tracking;
```

```

    Para  $j$  de 0 até  $LV.Count-1$  faça
         $V \leftarrow LV[j]$ ;
         $VS.Add(V)$ ;
         $A.EqUsed[i] \leftarrow True$ ;
         $N \leftarrow ArvB.AddNoFilho(N, V)$ ;
         $FS \leftarrow VS.Count = A.NLinhas$ ;
        Se not (FS) então Executar_Recursivamente_Passo2; Fim Se
        Se FS then Busca_Finalizada
        Senão
             $VS.Delete(VS.Count-1)$ ;
             $A.EqUsed[i] \leftarrow False$ ;
        Fim Se
    Fim Para
Fim Se
Abortar_Laço;
Fim Para

```

6.3.2.8. Inexistência de Solução

Conforme dito anteriormente, o número de equações da matriz **A** é idêntico ao número de variáveis de saída do DFS, de forma que em geral o algoritmo de busca obtém êxito no processamento. Entretanto, uma análise mais aprofundada no sistema de equações mostra que as associações de fontes de esforço em paralelo e de fontes de fluxo em série tornam o sistema impossível para fins de cálculo da função de transferência.

Os grafos da figura 6.18 ilustram essas situações. No grafo 1, à esquerda, ao calcular uma função de transferência em relação ao esforço na fonte E1, o esforço na fonte E2 deve receber o valor nulo, de acordo com o teorema da superposição, [2]. Como as fontes são ideais, o esforço em Z1 passa a ser simultaneamente dado por E1 e por zero, tornando o sistema de equações impossível. O mesmo acontece no grafo 2, à direita, em relação a variável generalizada de fluxo. Ao zerar o fluxo na fonte F2, por exemplo, o fluxo na impedância Z1 passa a ser simultaneamente F1 e zero, tornando impossível a solução do sistema de equações.

Nestes casos, as matrizes **A₄** e **A₅**, obtidas pelo algoritmo Grafo-DFS, contém equações que impossibilitam a construção de um diagrama de fluxo de sinal para o sistema físico.

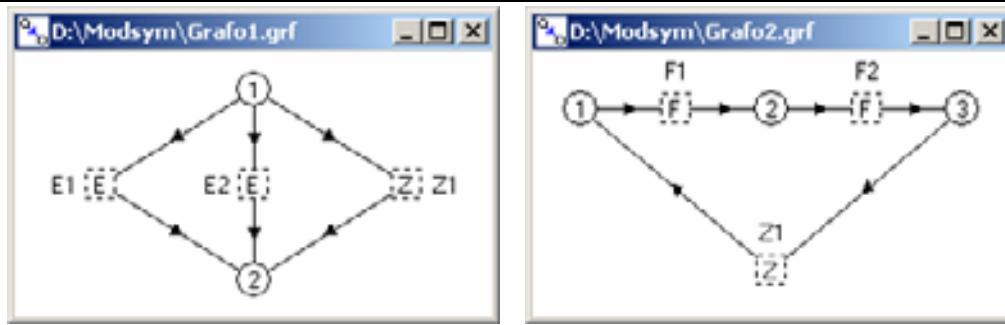


Figura 6.18. Exemplos de grafos de ligação sem DFS.

6.3.2.9. Determinação das Transmitâncias do DFS

Por fim, após a realização do algoritmo de busca, o vetor solução **VS** gerado por este algoritmo é processado de forma a definir as transmitâncias do DFS.

O pseudocódigo a seguir ilustra a operação. Inicialmente, o número de transmitâncias do diagrama, dado por TID, é zerado. Em seguida, cada equação da matriz **A** é processada. A variável jVS recebe então o índice da variável de saída calculada pela primeira linha da matriz, cujo ID no DFS é dado por jVSID. Em seguida, todas as colunas dessa linha são analisadas. As colunas com elementos não nulos e que se referem a variáveis diferentes da variável de saída originam transmitâncias no diagrama, cujo ganho é dado pela razão entre o elemento de **A** que se refere a tais variáveis e o elemento de **A** que se refere a variável de saída. O processamento é então repetido para as demais linhas da matriz, finalizando a definição da estrutura de dados do diagrama de fluxo de sinal. É com base nesta estrutura de dados que a regra de Mason calcula as funções de transferência do sistema físico.

```

TID ← 0;
Para i de 0 até A.NLinhas-1 faça
    jVS ← X.IndexOf(VS[i]);
    jVSID ← jVS + 1;
    Para j de 0 até A.NColunas-1 faça
        Se (A[i, j] <> '0') e (j <> jVS) então
            jVOID ← j + 1;
            Ganho = - A[i, j] / A[i, jVS];
            Inc(TID);
            DFS.AddRamo(TID, Ganho, jVOID, jVSID);
        Fim Se
    Fim Para

```

Fim Para

A figura 5.19 mostra o diagrama gráfico equivalente à estrutura de dados do diagrama de fluxo de sinal obtido pelo algoritmo Grafo-DFS para o exemplo do motor DC. Conforme visto, o diagrama apresenta as variáveis de esforço e fluxo de todos os elementos físicos, o que possibilita calcular a função de transferência entre qualquer uma dessas variáveis e a variável de entrada.

O algoritmo proposto para obtenção do diagrama de fluxo de sinal de um sistema físico demonstrou-se eficiente na tarefa pertinente a ele e foi aplicado com sucesso no processamento necessário ao cálculo de funções de transferência de sistemas físicos modelados no ambiente computacional. No próximo capítulo são apresentados mais resultados produzidos pelo ambiente computacional, obtidos com a utilização dos algoritmos descritos neste capítulo.

7. Resultados

7.1. Introdução

Este capítulo apresenta resultados obtidos pelo ambiente computacional proposto. Basicamente, são apresentados modelos de sistemas físicos dos diversos domínios em estudo e funções de transferência na forma simbólica obtidas entre variáveis destes sistemas. Grafos de ligação, diagramas de fluxo de sinal e resultados produzidos pela regra de Mason são também vistos. Todos os resultados obtidos e apresentados podem ser comprovados analiticamente.

7.2. Exemplos de Grafos de Ligação e Funções de Transferência

7.2.1. Sistema Elétrico

A figura 7.1 apresenta um circuito elétrico simples formado por dois capacitores e dois resistores, dado em [3]. Na figura à esquerda, é visto o diagrama de desenho gráfico do circuito e na figura à direita, o grafo de ligação obtido pelo ambiente computacional.

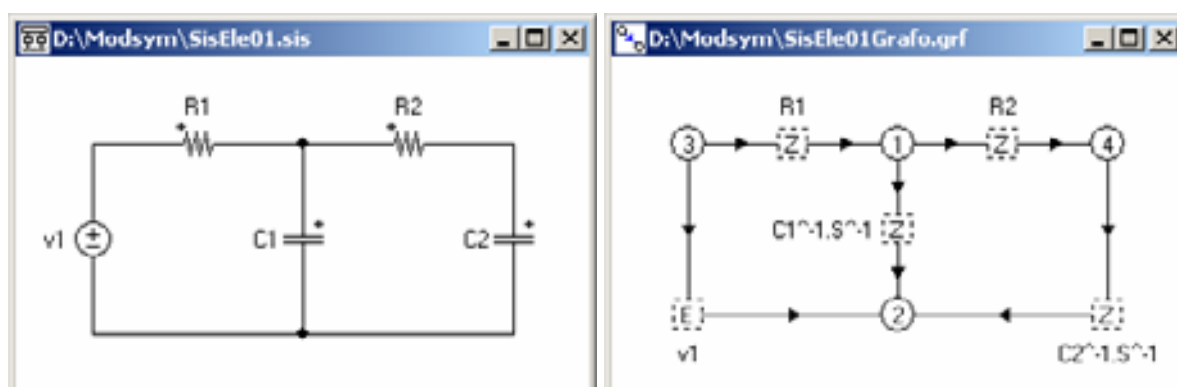


Figura 7.1. Exemplo de sistema elétrico.

A função de transferência entre a tensão no capacitor C2 e tensão na fonte, calculado pelo *ModSym*, pode ser vista na figura 7.2. Sistemas de segunda ordem, como este, são normalmente abordados na literatura de controle e servem como exemplos para ilustrar a metodologia de cálculo de funções de transferência.

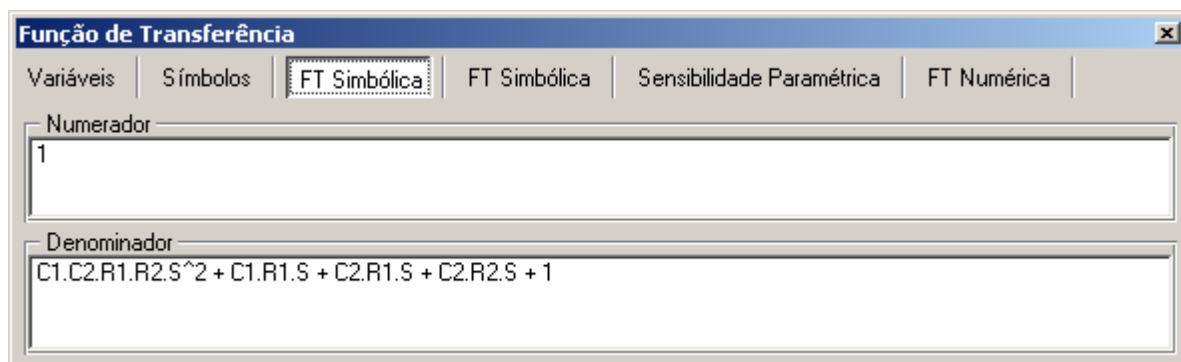


Figura 7.2. Função de transferência do sistema elétrico.

7.2.2. Motor DC com Controlador PID

O diagrama da figura 7.3 modela um sistema de controle de velocidade de um motor DC utilizando um controlador PID. A descrição do funcionamento do sistema é vista na Seção 5.2.2.7.

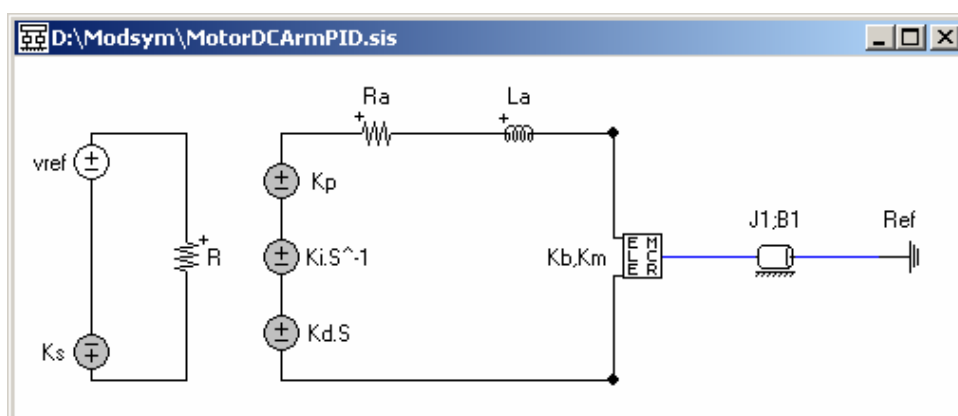


Figura 7.3. Motor DC com controlador PID.

O grafo de ligação para o sistema é apresentado na figura 7.4. A tensão de referência é fornecida por uma fonte de esforço com ganho v_{ref} . O sensor de velocidade e o controlador PID são representados por fontes de esforço controladas. A variável que controla o sensor é o esforço no corpo. As fontes do PID são controladas pelo esforço no resistor R utilizado para gerar o sinal de erro. O acoplamento entre os sistemas elétrico e mecânico é realizado por um elemento

generalizado de duas portas de energia.

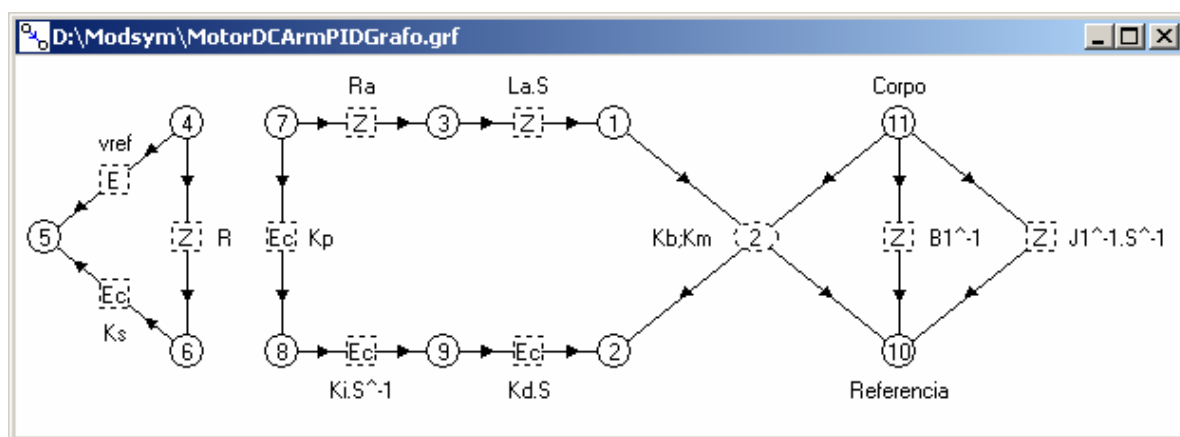


Figura 7.4. Grafo de ligação do motor DC com controlador PID.

A função de transferência calculada pelo ambiente computacional, entre a velocidade no corpo e a tensão de referência, é apresentada na figura 7.5. Conforme visto, esta função é de terceira ordem. A solução analítica de funções como esta já demanda um número razoável de manipulações matemáticas.

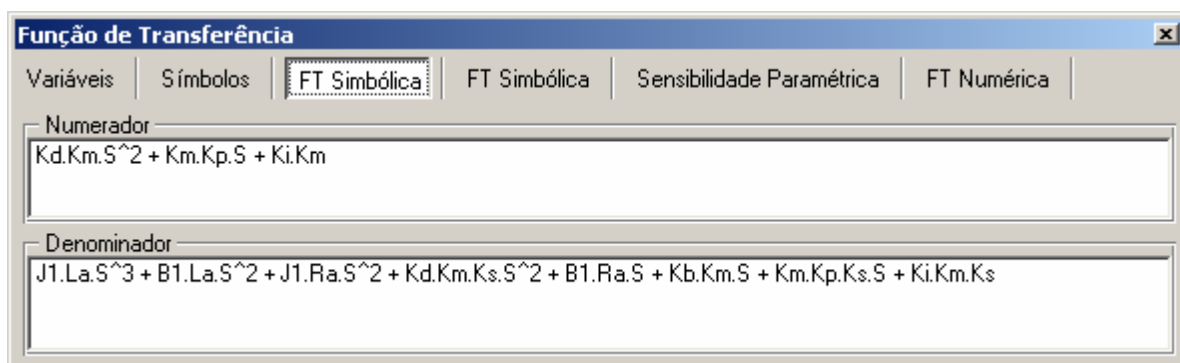


Figura 7.5. Função de transferência do Motor DC com controlador PID.

7.2.3. Sistema Mecânico Translacional

A figura 7.6 mostra um sistema mecânico que modela a suspensão de um automóvel, [32]. Neste modelo, o corpo M1 representa a massa da roda e de outros componentes entre a mola espiral do veículo e a estrada. Esta massa é acoplada a estrada através da mola K1 que representa a elasticidade do pneu. O corpo M2 representa ¼ da massa do chassi, do motor e dos demais componentes do carro. A massa M2 é acoplada a M1 através da mola espiral K2 e do amortecedor B1 que constituem a suspensão do veículo, propriamente dita. O grafo de ligação para este sistema pode ser visto na figura 5.12.

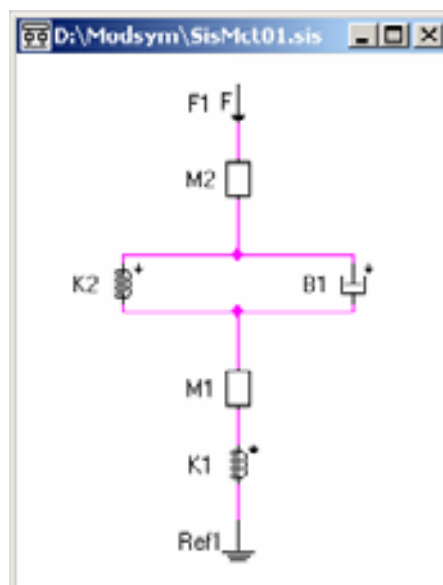


Figura 7.6. Exemplo de sistema mecânico translacional.

A função de transferência entre a velocidade na massa M2 e a força aplicada ao sistema pode ser vista na figura 7.7. Neste exemplo, a função apresentada é de quarta ordem. Funções com esta ordem de grandeza são encontradas com pouca frequência em uma abordagem educacional de cursos de controle.

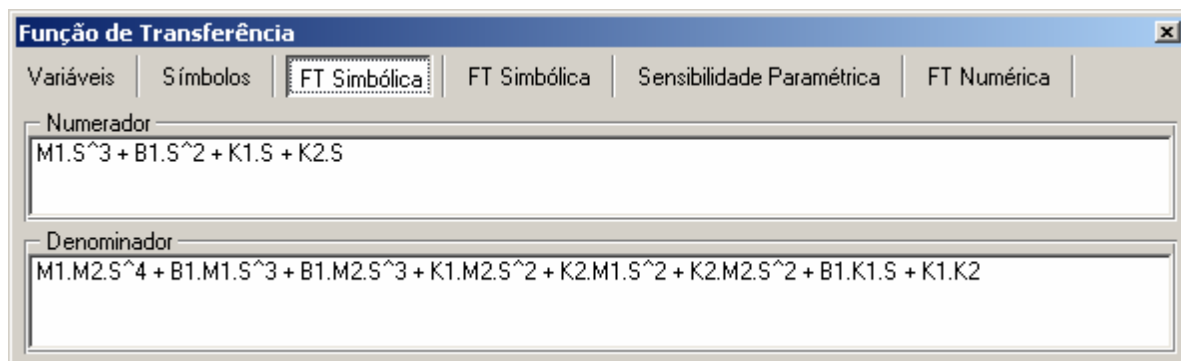


Figura 7.7. Função de transferência do sistema mecânico translacional.

7.3. Exemplos de DFS e Funções de Transferência

7.3.1. Sistema Elétrico com Duas Fontes de Energia

A figura 7.8 mostra um circuito elétrico com duas fontes de energia. Quando uma função de transferência é calculada tomando a tensão $v1$ como entrada, a corrente $i1$ é zerada e o circuito se comporta como um divisor de tensão. De forma análoga, ao estabelecer como entrada a corrente $i1$, a tensão $v1$ é zerada e o

circuito se comporta como um divisor de corrente.

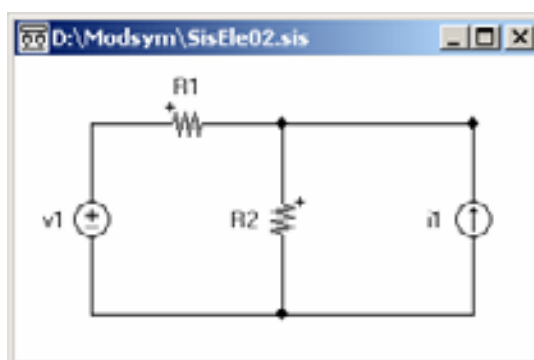


Figura 7.8. Exemplo de sistema elétrico com duas fontes de energia.

A figura 7.9 mostra o diagrama de fluxo de sinal para o circuito obtido pelo algoritmo proposto no trabalho. Conforme visto, o algoritmo inclui no DFS duas variáveis de entrada, dadas pelo esforço na fonte de tensão e o fluxo na fonte de corrente, o que permite calcular funções de transferências para o circuito considerando uma ou outra excitação.

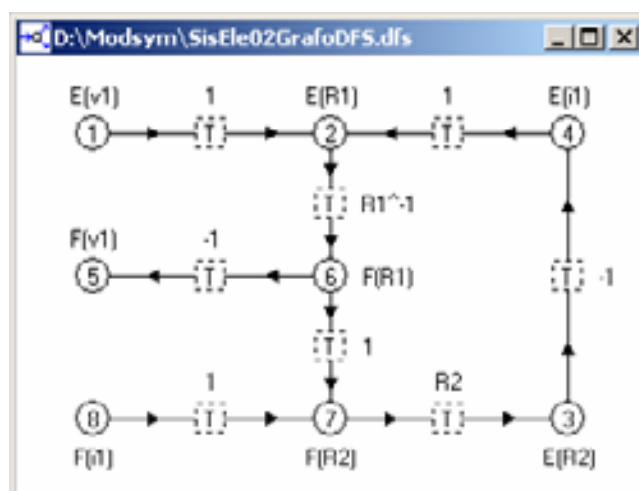


Figura 7.9. DFS do sistema elétrico com duas fontes de energia.

A função de transferência entre a tensão no resistor $R1$ e a tensão na fonte $v1$, calculada pelo software, pode ser vista na figura 7.10.

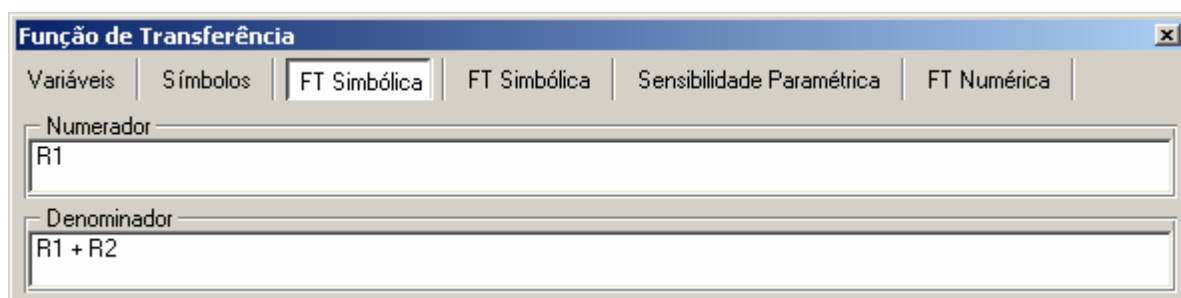


Figura 7.10. Função de transferência entre a tensão em $R1$ e a tensão em $v1$.

A função de transferência entre a corrente no resistor R1 e a corrente na fonte i1, pode ser vista na figura 7.11. A corrente no resistor é negativa porque a ela entra no elemento pelo referencial negativo.

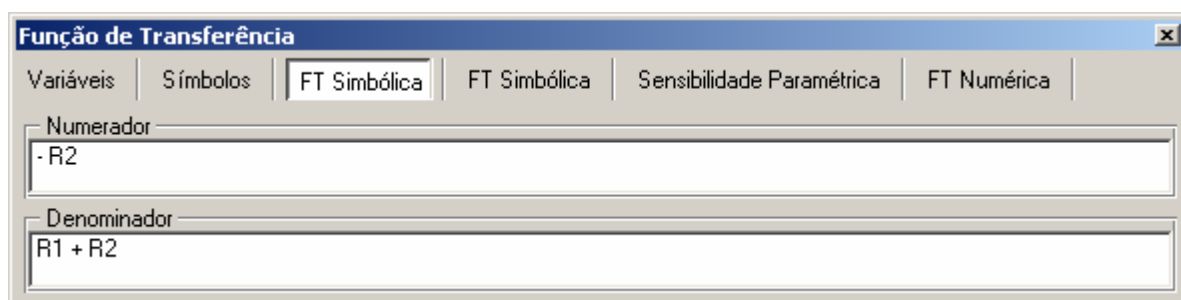


Figura 7.11. Função de transferência entre a corrente em R1 e a corrente em i1.

Sistemas maiores com mais de uma fonte de energia foram testados e obtiveram êxito. A intenção do exemplo é, entretanto, apresentar a funcionalidade do algoritmo em um exemplo que pode ser imediatamente analisado.

7.3.2. Sistema de Controle de Nível

A figura 7.12 mostra o modelo para um sistema de controle de nível em malha aberta, [20], visto na Seção 5.2.2.6. O objetivo do sistema, quando em malha fechada, é controlar a altura do fluido, dada por H, no reservatório Cr. O funcionamento do sistema é descrito na referida seção.

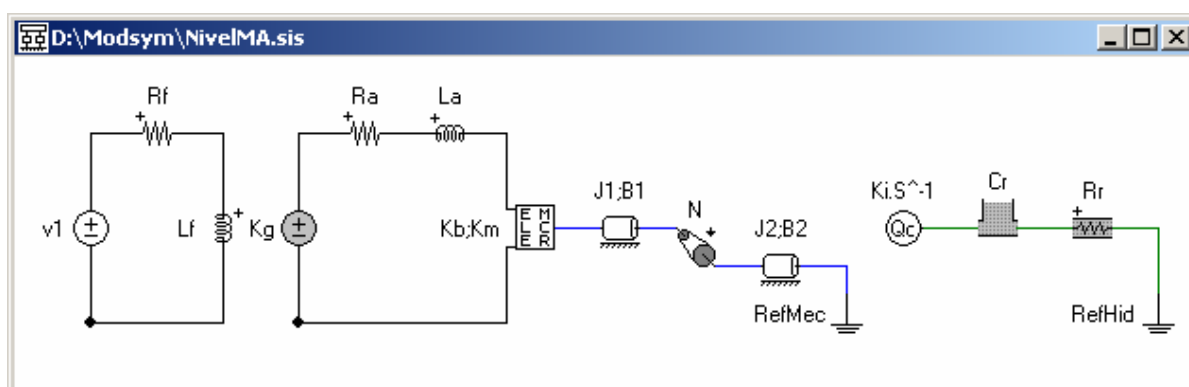


Figura 7.12. Sistema de controle de nível.

Para controlar a altura do fluido no reservatório, a função de transferência $G(S)$ entre a altura H do fluido e a tensão de referência v1 deve ser obtida:

$$G(S) = \frac{H(S)}{v1(S)} \quad (7.1)$$

Considerando constante a área da seção transversal do tanque, a equação pode ser reescrita na seguinte forma:

$$G(S) = \frac{1}{A.S} \frac{Qr(S)}{v1(S)} \quad (7.2)$$

onde: A é área da seção transversal do reservatório;

Q_r é a vazão no reservatório.

O ambiente computacional proposto pode ser usado para calcular a seguinte função de transferência:

$$G_q(s) = \frac{Qr(s)}{v1(s)} \quad (7.3)$$

onde: Q_r é a variável generalizada de fluxo no reservatório;

A figura 7.13 mostra a função de transferência $G_Q(S)$ calculada pelo software. Como pode ser visto, a função de transferência para este exemplo possui uma complexidade razoável. Devido ao grande número de termos da função, a solução analítica desta equação, sem a utilização de uma ferramenta computacional auxiliar, é bastante susceptível a erros.

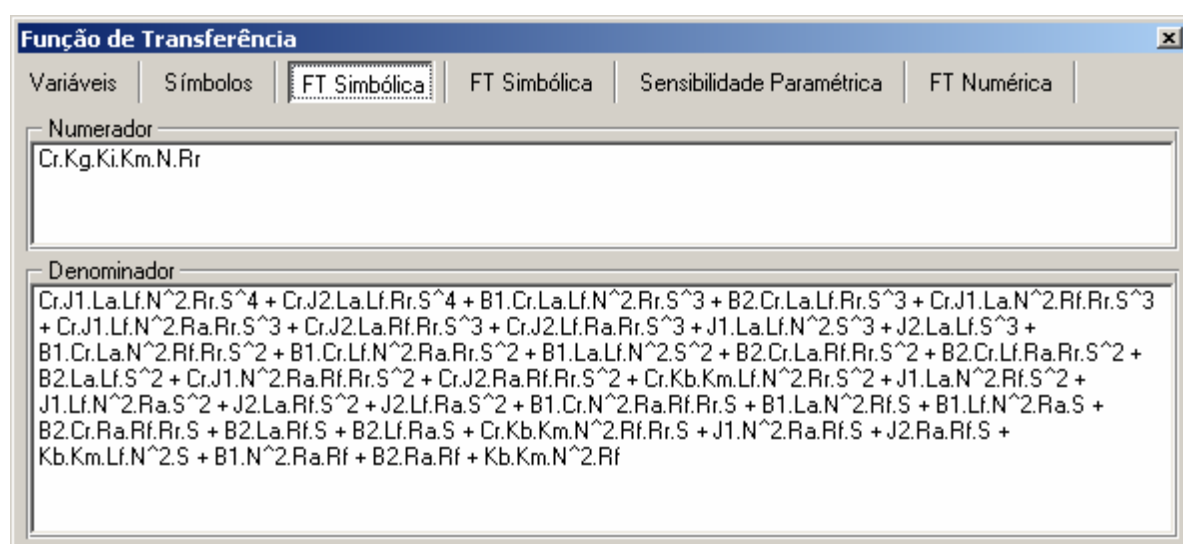


Figura 7.13. Função de transferência do sistema de controle de nível.

O diagrama de fluxo de sinal para o exemplo pode ser visto na figura 7.14. É interessante para o estudante de controle perceber no diagrama o acoplamento entre os subsistemas que compõem o controle de nível. O ramo cujos vértices

terminais são 20 e 16 acopla os dois subsistemas elétricos; os ramos com vértices terminais (7, 6) e (23, 24) acoplam os subsistemas elétrico e mecânico; por fim, o ramo cujos vértices terminais são 12 e 34 acopla o sistema mecânico ao hidráulico.

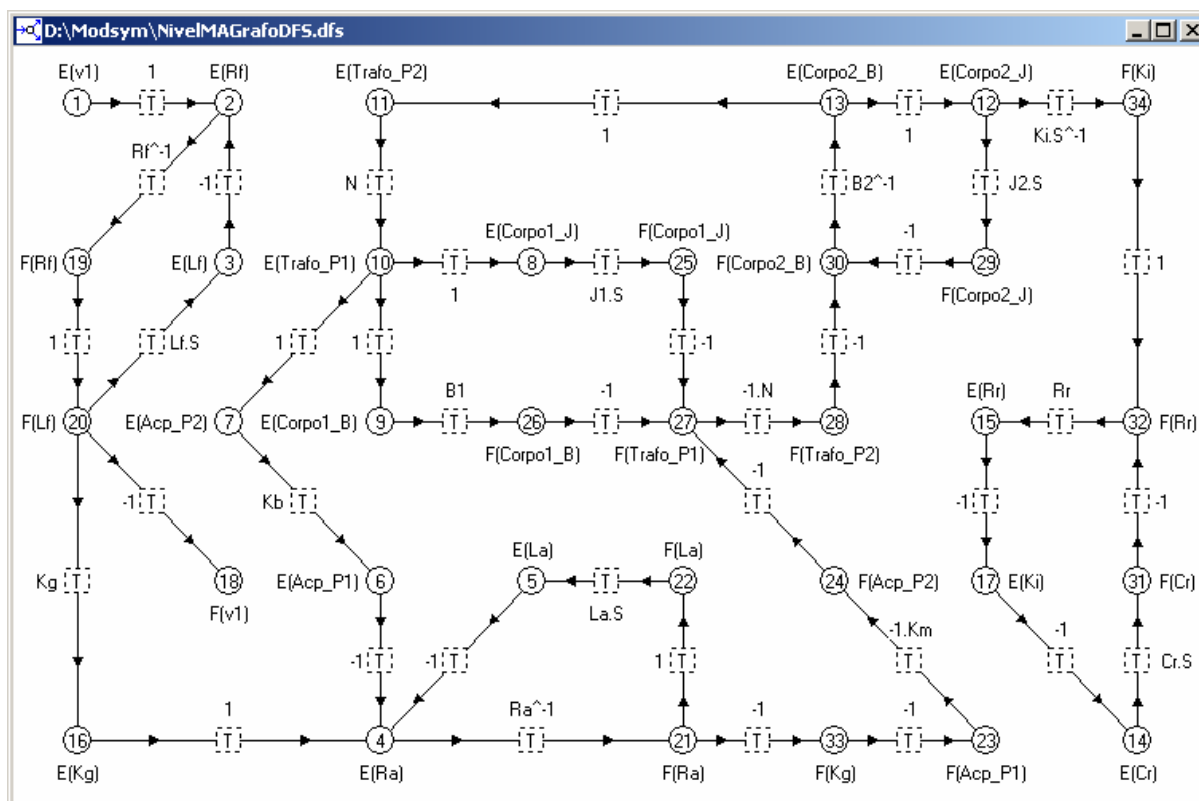


Figura 7.14. DFS do sistema de controle de nível.

Além dessa análise, os resultados produzidos pelo algoritmo de Mason são também de grande valia para o estudo do sistema. A figura 7.15, por exemplo, mostra os laços encontrados no DFS pelo algoritmo de Lin-Alderson, que foram utilizados no cálculo da função de transferência realizado anteriormente.

Resultados				
Símbolos	Busca de Laços	Laços	Laços Disjuntos	Floresta de Laços Disjuntos
				Laços de Ordem
Nº Laço	Vértices			
1	1, 2, 19, 20, 16, 4, 21, 33, 23, 24, 27, 28, 30, 13, 12, 34, 32, 15, 17, 14, 31, 1			
2	2, 19, 20, 3, 2			
3	4, 21, 33, 23, 24, 27, 28, 30, 13, 11, 10, 7, 6, 4			
4	4, 21, 22, 5, 4			
5	8, 25, 27, 28, 30, 13, 11, 10, 8			
6	9, 26, 27, 28, 30, 13, 11, 10, 9			
7	12, 29, 30, 13, 12			
8	14, 31, 32, 15, 17, 14			

Figura 7.15. Laços no DFS fechado do sistema de controle de nível.

A figura 7.16 mostra os laços de ordem superior para o exemplo. Como o ganho de cada laço se refere normalmente a um fator do numerador ou do denominador da função de transferência, dá para ter uma idéia do esforço necessário para se obter analiticamente esta função. Isto reforça a importância da ferramenta computacional proposta para o ensino em controle.

Resultados			
Laços Laços Disjuntos Floresta de Laços Disjuntos Laços de Ordem Superior			
Ordem	Nº Laço	Laços	Ganho
1	1	1	$-B2^{-1}.Cr.DFS.Kg.Ki.Km.N.Ra^{-1}.Rf^{-1}.Rr$
1	2	2	$-Lf.Rf^{-1}.S$
1	3	3	$-B2^{-1}.Kb.Km.N^2.Ra^{-1}$
1	4	4	$-La.Ra^{-1}.S$
1	5	5	$-B2^{-1}.J1.N^2.S$
1	6	6	$-B1.B2^{-1}.N^2$
1	7	7	$-B2^{-1}.J2.S$
1	8	8	$-Cr.Rr.S$
2	1	2, 3	$+B2^{-1}.Kb.Km.Lf.N^2.Ra^{-1}.Rf^{-1}.S$
2	2	2, 4	$+La.Lf.Ra^{-1}.Rf^{-1}.S^2$
2	3	2, 5	$+B2^{-1}.J1.Lf.N^2.Rf^{-1}.S^2$
2	4	2, 6	$+B1.B2^{-1}.Lf.N^2.Rf^{-1}.S$
2	5	2, 7	$+B2^{-1}.J2.Lf.Rf^{-1}.S^2$
2	6	2, 8	$+Cr.Lf.Rf^{-1}.Rr.S^2$
2	7	3, 8	$+B2^{-1}.Cr.Kb.Km.N^2.Ra^{-1}.Rr.S$
2	8	4, 5	$+B2^{-1}.J1.La.N^2.Ra^{-1}.S^2$
2	9	4, 6	$+B1.B2^{-1}.La.N^2.Ra^{-1}.S$
2	10	4, 7	$+B2^{-1}.J2.La.Ra^{-1}.S^2$
2	11	4, 8	$+Cr.La.Ra^{-1}.Rr.S^2$
2	12	5, 8	$+B2^{-1}.Cr.J1.N^2.Rr.S^2$
2	13	6, 8	$+B1.B2^{-1}.Cr.N^2.Rr.S$
2	14	7, 8	$+B2^{-1}.Cr.J2.Rr.S^2$
3	1	2, 3, 8	$-B2^{-1}.Cr.Kb.Km.Lf.N^2.Ra^{-1}.Rf^{-1}.Rr.S^2$
3	2	2, 4, 5	$-B2^{-1}.J1.La.Lf.N^2.Ra^{-1}.Rf^{-1}.S^3$
3	3	2, 4, 6	$-B1.B2^{-1}.La.Lf.N^2.Ra^{-1}.Rf^{-1}.S^2$
3	4	2, 4, 7	$-B2^{-1}.J2.La.Lf.Ra^{-1}.Rf^{-1}.S^3$
3	5	2, 4, 8	$-Cr.La.Lf.Ra^{-1}.Rf^{-1}.Rr.S^3$
3	6	2, 5, 8	$-B2^{-1}.Cr.J1.Lf.N^2.Rf^{-1}.Rr.S^3$
3	7	2, 6, 8	$-B1.B2^{-1}.Cr.Lf.N^2.Rf^{-1}.Rr.S^2$
3	8	2, 7, 8	$-B2^{-1}.Cr.J2.Lf.Rf^{-1}.Rr.S^3$
3	9	4, 5, 8	$-B2^{-1}.Cr.J1.La.N^2.Ra^{-1}.Rr.S^3$
3	10	4, 6, 8	$-B1.B2^{-1}.Cr.La.N^2.Ra^{-1}.Rr.S^2$
3	11	4, 7, 8	$-B2^{-1}.Cr.J2.La.Ra^{-1}.Rr.S^3$
4	1	2, 4, 5, 8	$+B2^{-1}.Cr.J1.La.Lf.N^2.Ra^{-1}.Rf^{-1}.Rr.S^4$
4	2	2, 4, 6, 8	$+B1.B2^{-1}.Cr.La.Lf.N^2.Ra^{-1}.Rf^{-1}.Rr.S^3$
4	3	2, 4, 7, 8	$+B2^{-1}.Cr.J2.La.Lf.Ra^{-1}.Rf^{-1}.Rr.S^4$

Figura 7.16. Laços de ordem superior no DFS fechado do sistema de controle de nível.

7.4. Exemplo Final

O diagrama de desenho gráfico da figura 7.17 mostra um dos maiores sistemas físicos testado para o ambiente computacional. Como pode ser visto, o sistema é um circuito elétrico composto por doze malhas resistivas. Sistemas análogos a este dificilmente são encontrados em uma abordagem educacional, visto que a solução analítica para uma função de transferência genérica, ou seja, com ganhos diferentes para todos os elementos, levaria o estudante à exaustão.

Para comprovar o resultado produzido pelo software, todos os resistores do circuito receberam resistências proporcionais a R , de forma que uma função de transferência pode ser obtida, para este caso em particular, através de reduções sucessivas na malha resistiva. De fato, a resistência equivalente do circuito é $2R$.

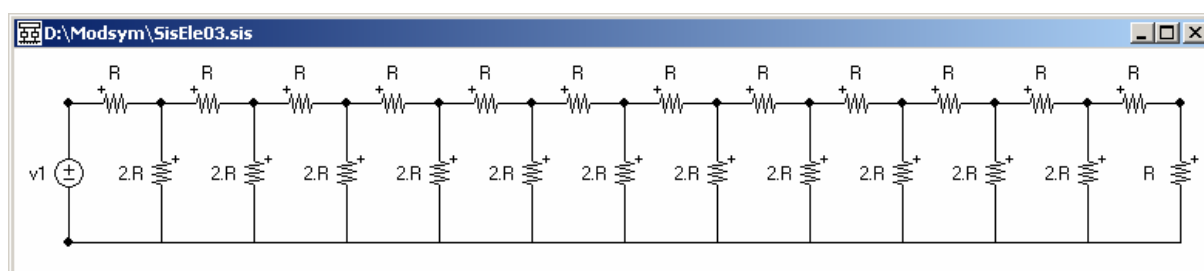


Figura 7.17. Sistema elétrico com 12 malhas resistivas.

A função de transferência entre a corrente e tensão na fonte $v1$ pode ser vista na figura 7.18. A corrente aparece com sinal invertido devido à convenção adotada pelo software. Conforme visto ainda, as funções de transferência também não são simplificadas. O DFS obtido para este circuito possui 102 laços de primeira ordem e 121.381 laços de ordem superior que chegam até a 11ª ordem. Isto mostra que o exemplo é muito mais complexo computacionalmente que o sistema de nível da figura 7.12. O cálculo da FT levou cerca de 2 segundos em um Pentium IV, 2GHz.

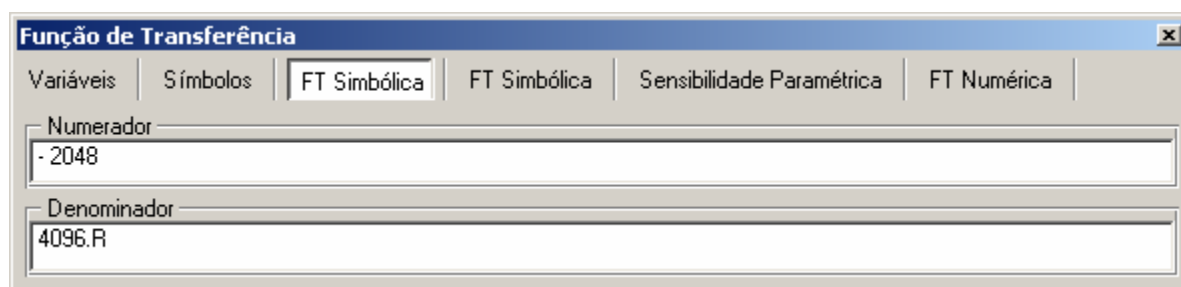


Figura 7.18. Função de transferência entre a corrente e tensão na fonte v1.

8. Conclusões

Apresentamos neste trabalho um ambiente computacional para a modelagem simbólica de sistemas físicos lineares. Acreditamos que o software, denominado *ModSym*, é uma ferramenta computacional bastante versátil para a área de controle, podendo ser utilizado tanto em trabalhos práticos como no processo de ensino e aprendizagem.

O ambiente computacional pode ser empregado na modelagem de sistemas físicos, auxiliando pesquisadores no projeto de sistemas e na obtenção de modelos matemáticos. O cálculo de funções de transferência nas formas simbólica e numérica, por exemplo, essencial para a realização de diversos experimentos, pode ser efetuado, precisa e rapidamente, com o auxílio do software.

No processo de ensino e aprendizagem, o *ModSym* pode ser utilizado em disciplinas relacionadas às áreas de controle, de física e de circuitos elétricos, entre outras. Os recursos disponíveis no software para a montagem de sistemas a partir de componentes básicos dos domínios elétricos, mecânicos e hidráulicos, a partir de grafos de ligação e de diagramas de fluxo de sinal podem ser utilizados para realizar explicações ricas em ilustrações e conteúdo que despertem o interesse dos alunos para as disciplinas correlatas ao software e para a ciência de um modo geral.

Na área educacional, o software pode ainda ser utilizado por estudantes para comprovar os resultados obtidos na realização de listas de exercícios e similares. A modelagem e análise dos sistemas físicos através da ferramenta permitem que estudos sejam efetuados com uma maior dinâmica, evitando ou auxiliando a realização de cálculos repetitivos e susceptíveis a erros.

Acreditamos também o trabalho apresenta contribuições científicas

importantes. Conforme dito anteriormente, pelo menos duas contribuições são relevantes à área de controle: primeiro, o ambiente computacional como um todo que abrange tópicos importantes da atividade de modelagem de sistema de controle não encontrados em outros softwares e segundo, o algoritmo de conversão da representação de um sistema físico como grafo de ligação em um diagrama de fluxo de sinal, que foi fundamental para a obtenção dos resultados apresentados e da funcionalidade requerida à ferramenta computacional.

Além disso, o ambiente computacional implementa uma quantidade razoável de algoritmos que podem ser utilizados em trabalhos de pesquisa correlatos. No total, o software possui aproximadamente 25000 linhas de código utilizadas na programação de mais de 130 classes requeridas à sua implementação. Como a modelagem do sistema obedece ao paradigma da Programação Orientada ao Objeto, o aproveitamento, a manutenção e o aperfeiçoamento das estruturas de dados e das rotinas implementadas no ambiente computacional podem ser realizados mais facilmente e adequados a outros trabalhos.

Além dos algoritmos apresentados, responsáveis pela obtenção do grafo de ligação e do diagrama de fluxo de sinal de um sistema físico, são codificados no software vários algoritmos relacionados à Teoria dos Grafos, como os algoritmos de busca de caminhos, de divisão de um grafo não conexo em subgrafos conexos e de obtenção da árvore geradora de um grafo. Em relação à Engenharia de Controle, são implementados algoritmos importantes como a regra de Mason e o cálculo de funções de sensibilidade paramétrica. Por este motivo, achamos de fundamental importância disponibilizar o código fonte do ambiente computacional proposto para que estudante e pesquisadores utilizem as classes, as estruturas e os algoritmos implementados no software em seus próprios trabalhos, o que certamente abre várias perspectivas para o aperfeiçoamento do próprio ambiente computacional.

Dentre as várias sugestões para trabalhos futuros, baseados no ambiente computacional, podemos citar: a extensão dos domínios físicos abordados pelo software com a inclusão, por exemplo, de sistemas térmicos e magnéticos; a definição de novos tipos de componentes como somadores e comparadores; a

inclusão de novos tipos de diagramas gráficos, como diagramas de blocos e *bond graphs*; o aperfeiçoamento da interface gráfica para suportar operações de zoom, por exemplo; e finalmente, a implementação de rotinas para organizar graficamente os grafos de ligação e os diagramas de fluxo de sinal obtidos pelos algoritmos implementados no ambiente computacional.

Enfim, esses são os principais aspectos que ressaltam a importância do ambiente computacional proposto e as principais sugestões para a continuidade do trabalho. Hoje, a integração que existe entre a Engenharia de Controle e a Engenharia de Computação, exemplificada aqui neste trabalho, tem sido de grande relevância para o desenvolvimento da ciência e da tecnologia, bem como para processo educacional da área de controle.

Referências Bibliográficas

- [1] Chen, C.T. (1984). *Linear System Theory and Design*. CBS College Publishing, New York.
 - [2] Dorf, R.C. & R.H. Bishop. (1995). *Modern Control Systems*. 7th Edition. Addison-Wesley Publishing Company, New York.
 - [3] Ogata, K. (2003). *Engenharia de Controle Moderno*. Quarta Edição. Editora Prentice-Hall do Brasil, São Paulo.
 - [4] Quartiero, E.M. (1999). As Tecnologias da Informação e Comunicação e a Educação. *Revista Brasileira de Informática na Educação*, Nº. 4, pp. 69-74.
 - [5] Zhu, J.J. (1996). Control Education: A World Showcase. *IEEE Control Systems Magazine*, Vol. 16(2), pp. 8-10.
 - [6] Dorf, R.C. & R.H. Bishop. (1999). Teaching Modern Control System Design. *Proc. of the 38th IEEE Conference on Decision and Control*, Vol. 1 , pp. 364 -369.
 - [7] Heck, B.S. (1999). Future Directions in Control Education. *IEEE Control Systems Magazine*, Vol. 19(5), pp. 16-17.
 - [8] Bissell, C.C. (1999). Control Education: Time for Radical Change?. *IEEE Control Systems Magazine*, Vol. 19(5), pp. 44-49.
 - [9] Kroumov, V. & H. Inoue. (2001). Enhancing Education in Automatic Control via Interactive Learning Tools. *Proc. of the 40th SICE Annual Conference*, pp. 220-225.
-

-
- [10] Silva, G.A. (1997). Um Ambiente para Síntese de Sistemas de Controle Empregando Otimização. Dissertação de Mestrado, UFRN, Natal-RN.
- [11] Silva, G.A., A.L. Maitelli & A.D. Araújo. (1998). Um Ambiente para Projeto de Controladores Clássicos Empregando Técnicas de Otimização. *Proc. of XII Brazilian Automatic Control Conference*, Vol. 6, pp. 1911-1916.
- [12] Wellstead, P.E. (1979). *Introduction to Physical System Modelling*. Academic Press.
- [13] Rabuske, M.A. (1992). *Introdução à Teoria dos Grafos*. Editora da UFSC.
- [14] Boaventura Netto, P.O. (1996). *Grafos: Teoria, Modelos, Algoritmos*. Editora Edgard Blücher Ltda, São Paulo, Brasil.
- [15] Maitelli, A.L. (1988). Geração Computacional de Funções de Sistema na Forma Simbólica. Dissertação de Mestrado. UnB. Brasília-DF.
- [16] Mason, S.J. (1953). Feedback Theory – Some Properties of Signal-Flow Graphs. *Proc. IRE*, 41, pp. 1144-1156.
- [17] Mason, S.J. (1956). Feedback Theory – Further Properties of Signal-Flow Graphs”. *Proc. IRE*, 44, pp. 920-926.
- [18] Maitelli, A.L. & G.A. Silva. (2004). Um Ambiente Computacional para Modelagem Simbólica de Sistemas Físicos Lineares Utilizando Grafos. *Revista Brasileira de Informática na Educação*, Nº. 12, pp. 9-23.
- [19] Maitelli, A.L. & G.A. Silva. (2004). Interactive Software for Symbolic Modeling of Physical Systems Using Graphs. *Proc. of First International Conference on Informatics in Control, Automation and Robotics*, Vol 2, pp. 488-491.
- [20] Maitelli, A.L. & G.A. Silva. (2004). Software Interativo para Modelagem Simbólica de Sistemas Físicos Utilizando Grafos. *Proc. of XV Brazilian Automatic Control Conference*, pp. 1-6.
-

-
- [21] Maitelli, A.L. & G.A. Silva. (2004). Um Algoritmo para Construção de Diagramas de Fluxo de Sinal de Sistemas Físicos. *Proc. of XV Brazilian Automatic Control Conference*, pp. 1-6.
- [22] Calvert, C. (1999). *Delphi 4 Unleashed*. Sams Publishing.
- [23] Cantù, M. (2003). *Dominando o Delphi 7 – A Bíblia*. Makron Books.
- [24] Booch, G., J. Rumbaugh & I. Jacobson. (2000). *UML – Guia do Usuário*. Primeira Edição. Editora Campus.
- [25] Guedes, G.T.A. (2004). *UML – Uma Abordagem Prática*. Novatec Editora.
- [26] Golten, J.W. & A.A. Verwer. (2002). Windows Packages for Control Education: PCS and CODAS. *IEEE: Engineering Education 2002: Professional Engineering Scenarios*, Vol. 2, pp. 49/1-49/6.
- [27] Prendergast, D.P. & A.M. Eydgahi. (1993). EDCON: An Educational Control System Analysis and Design Program. *IEEE Transactions on Education*, Vol. 36, pp. 42 -44.
- [28] Araújo, A.D. & G.A. Silva. (1996). Projeto de Controladores Assistido por Computador. *Anais do 7º Congresso Latino Americano de Controle Automático*, Vol. 2, pp. 684-689.
- [29] www.electronicworkbench.com
- [30] www.20sim.com
- [31] www.ansoft.com
- [32] Durfee, W.K., M.B. Wall, D. Rowell & F.K. Abbott. (1991). Interactive Software for Dynamic System Modeling Using Linear Graphs. *IEEE Control Systems*, pp. 60-66.
-

-
- [33] Adade Filho, A. & J.B. Gonçalves. (1999). Automatic Modelling of Lumped Parameters Dynamic Systems. *Revista Controle e Automação*, Vol. 10, pp. 1-12.
- [34] Luchetta, A., S. Manetti & A. Reatti. (2001). SAPWIN - A Symbolic Simulator as a Support in Electrical Engineering Education. *IEEE Transactions on Education*, Vol. 44.
- [35] Szymkat, M. (1996). Algorithms for Symbolic Reduction of Signal-Flow Graphs. *Mathematics and Computers in Simulation*, 42, pp. 675-684.
- [36] El-Hajj, A. & K.Y. Kabalan. (1995). A Transfer Function Computational Algorithm for Linear Control Systems. *IEEE Control Systems*, Vol. 15(2), pp. 114-118.
-