UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA QUÍMICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

DEPURAÇÃO PARA SIMULADORES DE PROCESSOS BASEADOS EM EQUAÇÕES

TESE DE DOUTORADO

RAFAEL DE PELEGRINI SOARES

PORTO ALEGRE, RS
2007

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA QUÍMICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

DEPURAÇÃO PARA SIMULADORES DE PROCESSOS BASEADOS EM EQUAÇÕES

RAFAEL DE PELEGRINI SOARES

Tese de Doutorado apresentada como requisito parcial para obtenção do título de Doutor em Engenharia.

Área de Concentração: Pesquisa e Desenvolvimento de Processos

Orientador:
Prof. Argimiro Resende Secchi, D.Sc.

PORTO ALEGRE, RS
2007

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA QUÍMICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

A Comissão Examinadora, abaixo assinada, aprova a Tese *Depuração para Simuladores de Processos Baseados em Equações*, elaborada por Rafael de Pelegrini Soares como requisito parcial para obtenção do Grau de Doutor em Engenharia.

Comissão Examinadora:

Prof. Evaristo Chalbaud Biscaia Jr., D.Sc.

Prof. Jorge Otávio Trierweiler, D.Sc.

Prof. Vilmar Trevisan, Phd

Uma longa jornada começa com um único passo.

(Lao-Tsé)

Agradecimentos

Aos colegas que formei no Departamento de Engenharia Química da UFRGS meus sinceros agradecimentos, muitos já o deixaram, novos chegaram, mas todos foram grandes companheiros de trabalho e acima de tudo amigos.

Agradeço ao Prof. Dr. Argimiro R. Secchi para o qual o adjetivo de orientador foi perfeito, trazendo sempre uma opinião de bom senso e visão global sem esquecer dos detalhes.

Agradeço em especial à Paulinha e à minha família que forneceram todo o suporte e apoio, tornando a execução deste trabalho muito mais fácil.

Nada mais justo que agradecer àqueles que tornaram este trabalho possível, nada mais injusto que não citar todos os nomes que, de alguma forma, contribuíram para este objetivo... a todos estes ficam minhas sinceras desculpas.

Resumo

Na área de simulação de processos, existe uma visível tendência da migração das ferramentas seqüenciais modulares, que hoje são as mais amplamente utilizadas, para as baseadas em equações. Uma das principais vantagens do paradigma baseado em equações ou simultâneo é que este se mostra eficiente na solução de problemas de simulação, otimização, estimação de parâmetros e reconciliação de dados, todos baseados em um mesmo conjunto de modelos, evitando retrabalho de modelagem. Porém, a tecnologia simultânea também apresenta algumas deficiências, onde destacam-se os problemas de robustez tanto na modelagem quanto na obtenção de resultados numéricos. Este trabalho tem como objetivo reunir e desenvolver técnicas que permitam reduzir estas deficiências. Para tanto, as técnicas conhecidas para depuração de sistemas de equações que representam problemas estacionários e dinâmicos foram estudadas em detalhe. Pôde-se observar que para o caso estático os métodos disponíveis para depuração de modelos, já se apresentam em um nível bem desenvolvido. Já para o caso dinâmico, onde há uma maior complexidade, as técnicas conhecidas encontram-se em um nível de desenvolvimento muito menor. Neste ponto encontram-se as principais contribuições deste trabalho.

Palavras chave: Análise Estrutural, Depuração, NLA, DAE, Índice, Inicialização Consistente

Abstract

In the field of process simulation the movement from the sequential modular tools, which are currently the most widely used, to the equation based approach is clear. One of the key advantages of the equation based or simulatneous approach is that using a single model one can solve simulation, optimization, parameter estimation, and optimization problems. This fact avoids modeling rework for each application. However, the simultaneous technology has problems regarding modeling and solving robustness. This work aims to group and develop methods capable of minimize these deficiencies. In order to achieve this goal, available debugging approaches for both steady-state and dynamic system of equations were studied in detail. For the steady-state case well stablished debugging techniques are known. For dynamic models, where the complexity is higher, the analysis and debugging methods are much less mature. This was the source for the major contributions of this work.

Keywords: Structural Analysis, Debug, NLA, DAE, Index, Consistent Initialization

Sumário

Lista de Figuras **xix**

Lista de Tabelas **xxi**

Lista de Símbolos **xxiii**

1 Introdução **1**

1.1 Histórico 1

1.2 Motivação 3

1.3 Estrutura da Tese 6

2 Depuração de Sistemas Não-Lineares **9**

2.1 Introdução 9

2.2 Teoria de Grafos 11

2.2.1 Definições Básicas 11

2.2.2 Grafos Bipartidos 12

2.2.3 Máxima Cardinalidade 13

2.3 Grafos Bipartidos e Sistemas de Equações 17

2.4 Detecção de Singularidades 18

2.5 Decomposição Dulmage-Mendelshon 19

2.5.1 Sub-sistemas sobre-determinados 21

2.5.2 Sub-sistemas sub-determinados 22
2.5.3 Conjunto Determinado 22
2.6 Depuração 25
2.6.1 Depuração de Sistemas Sub-Determinados 25
2.6.1.1 Depuração Fina do Conjunto Sub-determinado 28
2.6.2 Depuração de Sistemas Sobre-determinados 31
2.7 Análise Numérica 31

3 Sistemas Algébrico-Diferenciais 35

3.1 Introdução 35
3.2 Sistemas DAE *versus* ODEs 37
3.2.1 Condições Iniciais 40
3.3 O índice de sistemas DAE 41
3.4 Métodos para solução de sistemas DAE 43
3.4.1 Métodos de Passos Múltiplos 43
3.4.2 Métodos de Passo Simples 45
3.5 Análise Estrutural de Sistemas DAE 46
3.5.1 Sistemas DAE na forma de Grafos 48
3.5.1.1 Derivada Estrutural 49
3.6 Algoritmo de Pantelides 50
3.6.1 Exemplo de Aplicação 53
3.6.2 Falha na Detecção de Singularidades 56
3.6.3 Inicialização 57
3.6.4 Condições Iniciais Válidas 59
3.6.5 Outras Deficiências da Análise Estrutural 60
3.7 Os Estados de Sistemas DAE 62

3.7.1 Formulação de Espaço de Estado 63
3.7.2 Caso Geral de Sistemas DAE 64

4 Depuração de Sistemas Algébrico-Diferenciais 67

4.1 Introdução 67
4.2 Novo Algoritmo 68
4.2.1 Exemplo Ilustrativo 69
4.2.2 Modificação dos Algoritmos 71
4.2.3 Detecção de Singularidades 73
4.2.4 Possíveis Condições Iniciais 75
4.2.5 Unicidade da Solução 76
4.2.6 Sistemas Híbridos Contínuo-Discreto 77

5 Aplicações e Discussões 85

5.1 Sistemas NLA 85
5.2 Desempenho do Novo Algoritmo 88
5.3 Inicialização de sistemas DAE 89
5.3.1 Inicialização de sistemas de baixo índice 93
5.3.2 Inicialização de sistemas de índice elevado 96
5.3.3 Inicialização sem derivação simbólica 97
5.3.3.1 Inicialização Numérica 98
5.3.3.2 Inicialização Numérica Modificada 100
5.4 Reinicialização de DAEs 102
5.5 Solução de sistemas DAE de índice elevado 105
5.5.1 Solução com Redução de Índice 106
5.5.2 Códigos para sistemas de índice elevado 107

5.5.3 Problemas onde o índice estrutural supera o índice 108
5.5.4 Eventos Discretos em Sistemas de Índice Elevado 110

6 Conclusões **115**

A Algoritmos em C++ **119**

A.1 Bibliotecas para Grafos 119
A.2 RPS Graph 120
A.2.1 Classes para Grafos Bipartidos 120
A.2.1.1 Criação dos Grafos 122
A.2.2 Algoritmos 124
A.2.2.1 Associação Ótima 124
A.2.2.2 Pantelides 125
A.2.2.3 Novo Algoritmo para Análise de DAEs 125
A.2.3 Licensa de Uso 129

B Listagem de Modelos **131**

B.1 Processo de Síntese de Amônia 131
B.2 Processos de Destilação 134

Referências Bibliográficas **145**

Lista de Figuras

2.1 Representação gráfica do grafo $G(V, E)$, com o conjunto de vértices $V = \{1 \ldots 8\}$ e arestas $E = \{\{1, 5\}, \{3, 6\}, \{3, 7\}, \{4, 7\}\}$. 11

2.2 Aumentando a cardinalidade de M utilizando o caminho alternativo P. 14

2.3 Representação do sistema de equações (2.3) na forma de um grafo bipartido. 17

2.4 Associação de máxima cardinalidade para o sistema de equações (2.3) com os nós não cobertos grifados. 18

2.5 Uma segunda associação de máxima cardinalidade para o sistema de equações (2.3). 19

2.6 Conversão de uma associação ótima em um grafo direcionado para detecção da parte sobre-determinada. 21

2.7 Conversão de uma associação ótima para um grafo direcionado para detecção da parte sub-determinada. 22

2.8 Decomposição de Dulmage-Mendelsohn ao sistema de equações (2.3). . 23

2.9 Decomposição DM para o sistema de equações (2.3) considerando-se a relação adicional $f_3 - x_3$. 24

2.10 Sistema de troca térmica com um *bypass*. 27

2.11 Grafo direcionado para o sistema de equações (2.4) para determinação da parte sub-determinada. 27

2.12 Subgrafos induzidos por nós de variáveis livres para o sistema (2.5) . . . 28

2.13 Associação ótima quando adiciona-se ao sistema (2.4) uma equação para x_2 e outra para x_4. 29

2.14 Uma associação ótima explorando a simetria do sistema de equações (2.4). 30

2.15 Grafo bipartido representando o sistema de equações (2.5) e suas possíveis associações ótimas. 32

3.1 Tanque misturador com aquecimento. 37

3.2 Grafo para o sistema algébrico-diferencial (3.10). 49

3.3 Grafo para o sistema algébrico-diferencial (3.10) com a equação f_1 derivada estruturalmente. 50

3.4 Grafo para o modelo de um reator exotérmico com controle ótimo. . . . 54

3.5 Grafo da Figura 3.4 com o conjunto x removido (a) e uma das duas associações ótimas possíveis (b). 54

3.6 Primeira derivação no modelo de um reator exotérmico sugerida pelo algoritmo de Pantelides (a) e passo final (b). 55

3.7 Grafo para o sistema algébrico-diferencial (3.10) com a variável x removida (a) e uma das duas associações ótimas admitidas (b). 56

3.8 Grafos da evolução do Algoritmo 3 para o sistema (3.10). 57

3.9 Seqüência de grafos da aplicação do Algoritmo 3 ao sistema (3.15). . . . 60

3.10 Circuito elétrico linear, onde o índice estrutural é maior que o índice. . . 61

3.11 Seqüência de grafos da aplicação do Algoritmo 3 ao sistema (3.17). . . . 62

3.12 Definição de *estado* para diagramas de blocos. 63

4.1 Comparação entre grafos para o sistema (3.15) para o método de Pantelides (a) e o proposto (b). 70

4.2 Evolução do algoritmo proposto para o sistema de equações (3.15). . . . 71

4.3 Evolução do Algoritmo 5 para o sistema de equações (3.10). 74

4.4 Representação do sistema do pêndulo oscilante. 81

4.5 Evolução do Algoritmo 5 para o sistema de índice elevado (4.3). 82

5.1 Processo de síntese de amônia. 86

5.2 Grafo bipartido para o processo de síntese de amônia apresentado na Figura 5.1. 87

5.3 Erro apresentado pelo ASPEN Dynamics na solução do problema (3.15). 92

5.4 Evolução do Algoritmo 5 para o sistema galvanostático (5.1). 94

5.5 Aplicação do Algoritmo 5 ao sistema de equações (5.9). 103

5.6 Solução do sistema descontínuo (5.9). 104

5.7 Simulação de uma partida de uma coluna de destilação para visualização de eventos discretos. 105

5.8 Distorção na solução do sistema de índice 3 (4.3) com redução de índice por diferenciação e utilizando o método MEBDF. 107

5.9 Aplicação do Algoritmo 5 ao sistema de equações (3.17). 109

5.10 Solução numérica do sistema contínuo-discreto de índice elevado (5.11). 113

5.11 Detalhe de descontinuidade de derivadas na solução numérica do sistema contínuo-discreto de índice elevado (5.11). 114

A.1 Diagrama de classes de um nó (`Node`) e grafo (`Graph`). 121

Lista de Tabelas

5.1 Tempo computacional para a análise com o Algoritmo 5 de um modelo dinâmico de uma coluna de destilação para diferentes números de pratos. 88

5.2 Faixas de convergência de diferentes códigos quando da inicialização de (5.1). 95

5.3 Determinação de condições iniciais consistentes para o sistema de índice elevado (4.3) com os valores em negrito arbitrados. 97

5.4 Faixas de convergência de diferentes códigos quando da inicialização de (5.1), incluindo as técnicas por perturbação numérica. 100

Lista de Símbolos

E	Conjunto de arestas de um grafo
$F(y)$	Sistema de equações não-lineares
F_y	Matriz Jacobiana do sistema de equações não-lineares F com relação à y
F'_y	Gradiente do sistema de equações F com relação à y'
G	Grafo G com vértices V e arestas E
G^+	Grafo do sub-sistema sobre-determinado
G^-	Grafo do sub-sistema sub-determinado
G^w	Grafo do sub-sistema bem determinado
M^{max}	Associação máxima (ótima) em um grafo bipartido
t	Variável independente, usualmente o tempo
V	Conjunto de vértices de um grafo
V_e	Conjunto dos vértices que representam as equações em um grafo
v_e	Vértice de equação, faz parte do conjunto V_e
V_v	Conjunto dos vértices que representam as variáveis em um grafo
v_v	Vértice de variável, faz parte do conjunto V_v
y	Conjunto de variáveis dependentes
y'	Conjunto de derivadas das variáveis dependentes com relação à t
y^*	Solução para as variáveis dependentes
y_0	Estimativa inicial para as variáveis dependentes

Capítulo 1

Introdução

Na área de simulação de processos, existe uma visível tendência da migração das ferramentas seqüenciais modulares, que hoje são as mais amplamente utilizadas, para as baseadas em equações. Porém, isto só será uma realidade se algumas das deficiências do paradigma baseado em equações forem ultrapassadas, contribuir para isto é o objetivo deste trabalho.

1.1 Histórico

A simulação dinâmica, que tem emergido recentemente como uma tecnologia estratégica, surgiu antes mesmo da simulação estacionária. Os primeiros simuladores dinâmicos eram analógicos. A idéia era modelar um sistema na forma de equações diferenciais ordinárias e então construir um dispositivo físico que obedecesse tais equações. O sistema físico era *inicializado* com os valores desejados e o seu comportamento ao desenrolar do tempo imitava a solução das equações (ÅSTRöM et al., 1998). Um grande salto foi dado com a publicação do trabalho de Ragazzini et al. (1947), onde foi demonstrado que uma simulação poderia ser executada *eletronicamente*, onde as variáveis eram representadas como as voltagens em circuitos eletrônicos.

As ferramentas computacionais para simulação de processos, assim como as conhecemos hoje, já acumulam uma história de mais de 50 anos. Porém, estas têm se tornado populares apenas nos últimos 20 anos, juntamente com os PCs.

O conceito de operação unitária parece ter surgido em torno de 1915. Nos anos 50, foram desenvolvidos os primeiros modelos de operações unitárias os quais já eram executados em computadores (WESTERBERG, 1998). No final da década, Kesler e Kessler (1958) apresentaram o sistema *Flexible Flow*. Este programa computacional calculava de forma seqüencial os equipamentos, passando o resultado da saída de um como entrada do outro e iterando para a solução de reciclos. Esta metodologia tornou-se conhecida como modular seqüencial (*sequential modular*).

Na década de 60 houve um grande esforço no sentido do desenvolvimento de ferramentas do tipo modular seqüencial. Praticamente toda grande companhia do ramo químico ou petroquímico possuía um sistema próprio, chegando-se a mais de 200 pacotes diferentes (WESTERBERG, 1998). Este desenvolvimento interno e independente despendia de grandes esforços e com o passar do tempo foi sendo abandonado. De forma paralela, diversos pesquisadores do meio acadêmico iniciaram o desenvolvimento dos conceitos e métodos para o que chamamos hoje de sistemas baseados em equações (*EO, equation oriented*).

A metodologia EO é baseada em um conceito simples, porém a sua implementação recai na complexidade. Basicamente: ao invés de resolver as operações unitárias em seqüência e iterar para a solução de reciclos, um simulador EO constrói um único sistema de equações através da concatenação das equações descritivas de todos os equipamentos do processo e o resolve com auxílio de algum código apropriado. Além disto, as ferramentas baseadas em equações utilizam-se de modelos abertos e livres para a visualização e edição por parte do usuário. Nas ferramentas modulares, o conhecimento incorporado nos modelos dos equipamentos assim como sua estruturação são essen-

cialmente fixos e muitas vezes fechados (tipo *caixa preta*).

Atualmente existem no mercado empresas especializadas que desenvolvem e distribuem ferramentas para simulação de processos. Nestas ferramentas, ainda existe a divisão entre o paradigma modular e o baseado em equações mas as modulares são as mais utilizadas. Porém, nota-se um movimento no sentido das ferramentas baseadas em equações, principalmente para a utilização em simulação dinâmica e otimização, onde a utilização do paradigma modular não é conveniente por razões de eficiência computacional e flexibilidade para o cálculo de variáveis e especificações. Principalmente porque atualmente não é difícil se deparar com problemas altamente acoplados que envolvem de 10.000 a 100.000 equações e, freqüentemente, mais do que isto (BIEGLER et al., 1997).

1.2 Motivação

O primeiro traço de uma ferramenta baseada em equações parece ser o *Exxon Flowsheeting System*, onde Goldstein e Stanfield (1970) propuseram a solução simultânea de diagramas de processos inteiros, utilizando um método tipo Newton. Por este motivo a tecnologia EO também é conhecida como *Solução Simultânea*.

Mais detalhadamente, uma simulação baseada na tecnologia EO é composta pelos seguintes passos:

- Os equipamentos dos processos são descritos individualmente através de suas variáveis e equações. Assim, os modelos ainda são descritos de uma forma modular embora esta modularidade muitas vezes não seja devidamente explorada;
- Os equipamentos são conectados uns aos outros e especificações adicionais

são fornecidas para contemplar os graus de liberdade do sistema;

- Estas equações, variáveis, conexões e especificações são concatenadas para formar um único sistema de equações;
- Finalmente o sistema de equações resultante é resolvido pela utilização de métodos apropriados.

Uma das principais vantagens do paradigma baseado em equações é que este se mostra eficiente na solução de problemas de simulação, otimização, estimação de parâmetros e reconciliação de dados, todos baseados em um mesmo conjunto de modelos (RICO-RAMIREZ, 1998). Claramente a tecnologia EO é muito mais elegante e apresenta uma série de vantagens, algumas delas são:

- A utilização de ferramentas modulares em processos de grande dimensão e que apresentem um grande número de reciclos ou especificações de variáveis de saída de equipamentos é inviável devido à necessidade da repetida solução do processo como um todo;
- Modelos de sistemas EO podem ser muito mais facilmente modificados ou estendidos, principalmente se o sistema apresenta uma linguagem de modelagem orientada a objetos;
- Diversos tipos de solução podem ser aplicados ao mesmo sistema de equações gerado pelas ferramentas EO, tais como: soluções estacionárias, dinâmicas, otimizações, estimações de parâmetros, reconciliação de dados e geração de modelos lineares.

Porém, a tecnologia EO também apresenta algumas deficiências quando comparada com a tecnologia modular:

- Em geral, os códigos para solução do sistema de equações não são tão robustos quanto os códigos desenvolvidos para a solução de equipamentos

específicos. Em verdade, na solução de problemas complexos com a tecnologia EO, muitas vezes é necessário o conhecimento de uma boa aproximação da solução para que esta seja obtida;

- Processos que não contém reciclos apresentam uma potencial vantagem para as arquiteturas modulares. Uma vez que os equipamentos do diagrama de processos podem ser resolvidos de forma independente na ordem em que as correntes são conectadas. Contudo, isto pode ser tratado na tecnologia EO com o uso de técnicas de decomposição de blocos (PANTELIDES, 1988a);

- Normalmente, a simulação de processos utilizando a tecnologia EO requer mais recursos computacionais que a solução modular. Porém, os avanços em hardware agregados aos códigos de álgebra esparsa disponíveis hoje praticamente eliminaram esta deficiência.

Uma discussão detalhada sobre a comparação entre a metodologia modular e a baseada em equações é apresentada por Marquardt (1996). O reconhecimento dos benefícios da tecnologia baseada em equações levou ao desenvolvimento de diversas ferramentas, exemplos são: SpeedUp (PANTELIDES, 1988b), OMOLA (MATTSSON; ANDERSSON, 1993), gPROMS (OH; PANTELIDES, 1996) e ASCEND (ALLAN, 1997).

Em 2001 foi iniciado o desenvolvimento de um novo simulador de processos baseado em equações (SOARES; SECCHI, 2003). Esta ferramenta, chamada EMSO (*Environment for Modelling Simulation and Optimization*), hoje contém uma rica biblioteca aberta de modelos. Esta biblioteca contempla os principais equipamentos da indústria química e petroquímica, tornando fácil até mesmo a modelagem de processos topologicamente complexos. Porém, devido à própria natureza do EMSO (uma ferramenta baseada em equações), existem algumas deficiências que tendem a afastar usuários menos experientes com a simulação de processos. Contextos típicos onde estas defi-

ciências aparecem são:

- Para um diagrama subdeterminado, quais são os conjuntos válidos de variáveis que devem ser especificadas?
- Para um diagrama sobredeterminado, quais especificações e/ou equações devem ser removidas?
- Como detectar se um modelo está mal posto e reportar esta singularidade de forma que a mesma possa ser facilmente corrigida?
- Em uma simulação dinâmica, quais são os possíveis conjuntos de condições iniciais que devem ser fornecidos?
- Como tratar os modelos dinâmicos quando há problema de índice diferencial elevado?
- No caso de uma falha na solução numérica, que ação corretiva deve ser tomada?

Este trabalho objetiva reunir, aplicar e desenvolver técnicas que auxiliem na solução destas perguntas de forma eficiente computacionalmente.

1.3 Estrutura da Tese

Esta tese está dividida em seis capítulos, arranjados da seguinte forma:

O Capítulo 1 (este capítulo) trata da introdução, motivação e objetivos do trabalho.

A simulação estacionária em ferramentas baseadas em equações requer a solução de sistemas de equações não-lineares. No Capítulo 2 é apresentada como a teoria de

grafos se aplica tipicamente à análise estrutural de problemas desta categoria. Neste capítulo aparecem as primeiras contribuições deste trabalho, onde uma nova forma de reportagem de singularidades estruturais é proposta.

No Capítulo 3 é visto porque a modelagem de sistemas dinâmicos gera naturalmente sistemas de equações algébrico-diferenciais. Neste capítulo também é apresentada a teoria conhecida para solução desta categoria de problemas e quais são as principais dificuldades quando comparados aos bem conhecidos sistemas de equações diferenciais ordinárias. Além disto, o histórico sobre a análise de resolubilidade para sistemas algébrico-diferenciais é apresentado. A técnica estrutural mais utilizada para a análise e redução de índice é estudada em detalhe, revelando as suas principais deficiências.

Historicamente, a análise de resolubilidade de sistemas algébrico-diferenciais se detém apenas ao problema da determinação do número de graus de liberdade dinâmicos e ao problema de índice. Porém, todo o conhecimento sobre sistemas não-lineares, apresentado no Capítulo 2, pode ser adaptado para o caso dinâmico. No Capítulo 4 é apresentado como esta adaptação pode ser feita. Para tanto, modificações à técnica clássica, estudada no Capítulo 3, são necessárias. Tais modificações são as principais contribuições deste trabalho.

No Capítulo 5 é apresentado como as técnicas, já bem conhecidas na literatura, revisadas no Capítulo 2, podem ser aplicadas. A solução de problemas típicos envolvendo a as técnicas desenvolvidas no Capítulo 4 também são apresentadas. Além disto, alguns casos particulares são analisados e discutidos.

No Capítulo 6 são sumarizadas as principais questões abordadas na tese. Neste capítulo também são reforçados os pontos de contribuição do trabalho.

Capítulo 2

Depuração de Sistemas Não-Lineares

A análise estrutural de sistemas lineares e não-lineares é bem desenvolvida. Neste capítulo são apresentadas técnicas, baseadas na teoria de grafos, que permitem uma fina depuração de sistemas não-lineares.

2.1 Introdução

Um sistema de equações não-lineares (*Nonlinear Algebraic - NLA*) genérico pode ser representado por:

$$F(y) = 0 \tag{2.1}$$

onde, $F : \Re^n \rightarrow \Re^m$ e $y \in \Re^n$. O sistema (2.1) define um sistema de m equações em termos de n variáveis. No caso mais simples, onde as funções são apenas lineares (*Linear Algebraic - LA*), o sistema (2.1) se reduz a:

$$Ay = b \tag{2.2}$$

Em simuladores de processos baseados em equações, sistemas como (2.1) surgem

naturalmente da modelagem estacionária. Para a solução de (2.1), tipicamente são utilizados métodos tipo Newton (SOARES, 2003). Estes métodos resolvem iterativamente problemas lineares como (2.2), onde a matriz A é alguma aproximação da matriz Jacobiana de F com relação às variáveis y, F_y. Na prática, para que a solução do sistema seja obtida é necessário que F_y seja inversível em todo o caminho $y = y_0 \rightarrow y^*$, onde y_0 é uma estimativa inicial e y^* é a solução. Por este motivo a análise de sistemas não-lineares está intimamente ligada à análise de sistemas lineares.

A maioria dos códigos para inversão de matrizes é capaz de detectar singularidades numéricas em matrizes, porém estes não geram informações detalhadas sobre a causa da singularidade. Códigos robustos de álgebra linear aplicam, antes de executar a solução numérica, uma análise estrutural que explora a esparsidade da matriz verificando se o problema está bem posto (DUFF et al., 1986; AMESTOY et al., 2004). Onde, uma propriedade de uma matriz é dita *estrutural* se esta independe dos valores dos elementos não nulos da matriz. Entretanto, a reportagem de informações precisas, que levem à fonte da singularidade, é muito importante para o sucesso de uma ferramenta baseada em equações.

A análise estrutural e diagnóstico de problemas em sistemas lineares foi o tema de diversos trabalhos disponíveis na literatura. Por exemplo, Sridhar et al. (1996) apresentam uma revisão interessante dos trabalhos disponíveis até o momento.

Nas seções subseqüentes são apresentadas técnicas estruturais (baseadas na teoria de grafos) capazes de reportar a fonte de singularidades em sistemas NLA. Para tanto, primeiramente alguns conceitos básicos da teoria de grafos são apresentados na Seção 2.2.

2.2 Teoria de Grafos

Os *grafos* têm se tornado ferramentas cada vez mais poderosas na engenharia, comunicações, gerenciamento industrial, informática, economia, marketing e muitas outras áreas. Este crescimento se dá porque os problemas práticos que se enquadram no modelo de um grafo e que podem ser resolvidos com a teoria de grafos são muitos (KREYSZIG, 2000).

Exceto quando referenciado de forma diferente, os conceitos e definições particulares à teoria de grafos apresentados nesta seção foram retirados de Diestel (2000).

2.2.1 Definições Básicas

Um *grafo* consiste em um par de conjuntos V e E que satisfazem $E \subseteq [V]^2$. Desta forma, os elementos de E são pares dos elementos de V. Os elementos de V são os *vértices* (*nós*, *pontos* ou *vertex*) do grafo $G = (V, E)$ e os elementos de E são as *arestas* (*relações*, *linhas* ou *edges*). Na Figura 2.1 é apresentada a representação gráfica de um grafo.

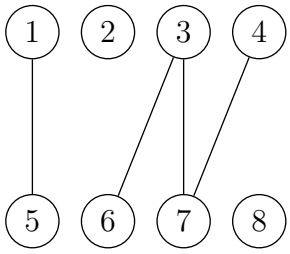


Figura 2.1: Representação gráfica do grafo $G(V, E)$, com o conjunto de vértices $V = \{1 \dots 8\}$ e arestas $E = \{\{1, 5\}, \{3, 6\}, \{3, 7\}, \{4, 7\}\}$.

Um grafo com o conjunto de vértices V é dito um grafo *em* V. O conjunto de vértices de um grafo G é referenciado por $V(G)$ e seu conjunto de arestas como $E(G)$.

Um vértice v é *incidente* em uma aresta e se $v \in e$. Os dois vértices incidentes em uma aresta são ditos *adjacentes*.

No grafo da Figura 2.1, o subconjunto de arestas $P = \{\{3,6\},\{3,7\},\{4,7\}\}$ formam um *caminho*. Neste caso, o caminho $\{6-3-7-4\}$ *conecta* os nós 6 e 4 por P. O número de arestas de um caminho P é o seu *comprimento*, $|P|$. Da mesma forma, $|V|$ e $|E|$ são o número de vértices e arestas, respectivamente, de um grafo $G = (V, E)$.

2.2.2 Grafos Bipartidos

Uma importante classe de problema de otimização combinatorial são os problemas de associação. Exemplos destes problemas são a associação de trabalhadores para tarefas, tarefas para máquinas, turmas para salas de aula, provas para períodos e assim por diante. Dentre estas aplicações, inclui-se a análise estrutural de sistemas de equações, onde é analisada a relação equação-variável. Em problemas de associação entre dois grupos é conveniente trabalhar com o conceito de grafos *bipartidos*.

Um grafo $G = (V = V_e \cup V_v, E)$ pode ser bipartido se V admite uma partição em duas classes tal que toda a aresta incide em ambas as classes. Esta condição impõe que nós pertencentes à mesma classe de partição não possam ser adjacentes. Neste trabalho, as classes de partição de V são providencialmente chamados de V_e e V_v.

O grafo da Figura 2.1 apresenta as propriedades de um grafo bipartido, com a possível partição em $V_e = \{1 \dots 4\}$ e $V_v = \{5 \dots 8\}$.

2.2.3 Máxima Cardinalidade

Um dos problemas mais básicos abordados na teoria de grafos, mas não menos importante, é o problema de associação de máxima cardinalidade ou associação ótima em grafos bipartidos. Neste problema é considerado um grafo bipartido e deseja-se encontrar o maior conjunto $M \in E$, tal que todos os elementos de M são independentes, ou seja, nenhuma aresta de M compartilha um mesmo vértice.

Uma aresta que pertence à uma associação M é dita *coberta* ou *não exposta* pela associação, o mesmo vale para os vértices. A *cardinalidade* da associação M é definida como o número de arestas $E(G)$ cobertas por M.

A forma clássica da solução deste problema é considerar uma associação arbitrária inicial de M em G (podendo ser inclusive uma associação nula). Considerando o grafo da Figura 2.1, uma associação arbitrária $M = \{\{1,5\},\{3,7\}\}$ é apresentada na Figura 2.2 (arestas destacadas). O caminho $P = \{6-3-7-4\}$ alterna arestas cobertas e não cobertas por M e tem seus extremos em nós não cobertos por M. Um caminho que alterne arestas cobertas e não com respeito à uma associação M é conhecido como *caminho alternante* (*alternating path*) em M. Se um caminho alternante tem seus extremos não cobertos por M, este é conhecido como um *caminho alternativo* com respeito à M. Diestel (2000) apresenta as definições de *alternating path* e *augmenting path*, a segunda definição foi traduzida neste trabalho como *caminho alternativo*.

Uma característica importante de um caminho alternativo P é que, se cada aresta de P coberta por M for removida de M e cada aresta não coberta for adicionada, a cardinalidade da associação M é aumentada em uma unidade. Esta propriedade é facilmente visualizada na Figura 2.2, pois aplicando-se este procedimento ao caminho P com relação à M (cardinalidade 2) obtém-se a associação $M' = \{\{1,5\},\{3,6\},\{4,7\}\}$

(cardinalidade 3).

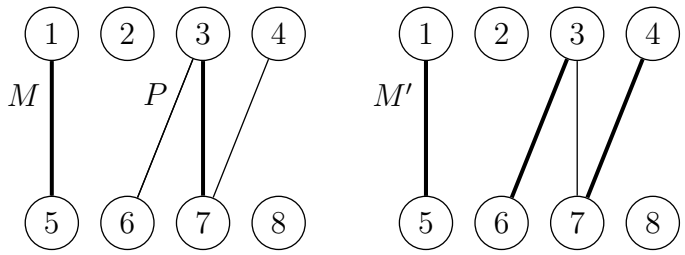


Figura 2.2: Aumentando a cardinalidade de M utilizando o caminho alternativo P.

Caminhos alternativos são muito importantes na procura de uma associação ótima M^{max}. Uma vez que, se há algum caminho alternativo, então uma associação de cardinalidade maior é viável. Esta é a base do teorema desenvolvido por Berge (apud KREYSZIG, 2000).

Teorema 2.1 (BERGE, 1957) *Uma associação M em um grafo bipartido G é máxima (ótima) se e somente se G não contém nenhum caminho alternativo à M.*

Assim, uma solução prática para a obtenção de uma associação de máxima cardinalidade é a procura por caminhos alternativos até que não haja mais caminhos alternativos viáveis. Quando uma associação ótima inclui todos os nós do grafo esta é dita uma associação *perfeita* ou *completa*.

Considerando um grafo bipartido, uma busca em profundidade (*depth-first search - DFS*) pode ser utilizada para encontrar caminhos alternativos à M conforme apresentado no Algoritmo 1.

Sobre o Algoritmo 1 é importante notar alguns pontos:

- O algoritmo retorna com sucesso se foi possível aumentar a cardinalidade da associação atual pela inclusão do vértice v_e.

Algoritmo 1 Pseudocódigo para aumentar a cardinalidade de uma associação M com relação à $G = (V_e \cup V_v, E)$, iniciando no vértice de equação v_e.

$AugmentMatching(G = (V_e \cup V_v, E), M, v_e)$

```
1:  colour v_e
2:  if exists {v_e, v_v} ∈ E and {v_e, v_v} ∉ M then
3:     M ← M ∪ {v_e, v_v}
4:     return true
5:  end if
6:  for all {v_e, v_v} ∈ E do
7:     if exists {v_e2, v_v} ∈ M and v_e2 not colored then
8:        if AugmentMatching(G = (V_e ∪ V_v, E), M, v_e2) then
9:           M ← M ∪ {v_e, v_v}
10:          return true
11:       end if
12:    end if
13: end for
14: return false
```

- A cardinalidade da associação pode ser incrementada quando a equação contém uma variável ainda não associada (linha 3) ou através de um caminho alternativo (linha 9).
- Note-se que o caminho alternativo pode ter qualquer comprimento, uma vez que o algoritmo é recursivo (linha 8).
- Na linha 1 o nó v_e é *colorido*, esta ação, juntamente com o teste na linha 7, faz com que um nó do conjunto V_e não seja visitado duas vezes, evitando uma recursividade infinita.
- Quando o algoritmo falha em aumentar a cardinalidade, todos os nós de equação tocados por caminhos alternantes estarão *coloridos*.

O Algoritmo 2 utiliza seqüencialmente o Algoritmo 1 (linha 4) para obter uma associação de máxima cardinalidade. Além disto, o algoritmo retorna verdadeiro se a associação é perfeita com relação à V_e, isto é detectado pela linha 5. A linha 7 é necessária para que vértices *coloridos* devido à falha na associação anterior sejam *descoloridos* antes do início de uma nova fase.

Algoritmo 2 Pseudocódigo para obtenção da associação de máxima cardinalidade M^{max} para um dado $G = (V_e \cup V_v, E)$.

$MaxMatching(G = (V_e \cup V_v, E), M)$

1: $M \leftarrow \emptyset$
2: $flag \leftarrow$ **true**
3: **for all** $v_e \in V_e$ **do**
4: **if not** $AugmentMatching(G = (V_e \cup V_v, E), M, v_e)$ **then**
5: $flag \leftarrow$ **false**
6: **end if**
7: *uncolour* V_e
8: **end for**
9: **return** $flag$

O Algoritmo 2 também é conhecido como algoritmo clássico ou rudimentar (Saip e Lucchesi (1993) o referenciam como *primitive algorithm*). Este algoritmo apresenta uma complexidade $O(mn)$, onde n é o número de vértices e m o número de arestas de G. A literatura apresenta diversas modificações do algoritmo clássico objetivando uma redução de complexidade. Por exemplo, no trabalho de Hopcroft e Karp (1973) os autores desenvolveram uma técnica capaz de aumentar a cardinalidade da associação em mais do que uma unidade por fase. Esta modificação faz com que a complexidade do algoritmo fique limitada a $O(m\sqrt{n})$. Considerando um grafo não denso, este é o algoritmo seqüencial mais eficiente disponível (SAIP; LUCCHESI, 1993). Já para grafos densos, o trabalho de Alt et al. (1991) apresenta uma técnica de complexidade $O(n^{3/2}\sqrt{m/\log n})$. Além destes algoritmos, Saip e Lucchesi (1993) também apresentam uma série de algoritmos paralelos para a obtenção de associações ótimas.

Por motivos de clareza, os desenvolvimentos apresentados neste trabalho utilizam como base o algoritmo clássico. Isto não impede que implementações futuras sejam baseadas em versões mais eficientes.

2.3 Grafos Bipartidos e Sistemas de Equações

Sistemas lineares (2.2), assim como não-lineares (2.1), podem ser representados na forma de grafos bipartidos, como definidos na Subseção 2.2.2. Onde as equações são os vértices V_e, as variáveis são os vértices V_v e as relações equação-variável são as arestas E.

Utilizando esta notação, o sistema de equações não-lineares (2.3) pode ser representado pelo grafo da Figura 2.3 (BUNUS, 2002).

$$\begin{aligned} &f_1(x_1) = 0 \\ &f_2(x_1, x_2) = 0 \\ &f_3(x_1, x_2) = 0 \\ &f_4(x_2, x_3, x_4) = 0 \\ &f_5(x_4, x_5) = 0 \\ &f_6(x_3, x_4, x_5) = 0 \\ &f_7(x_5, x_6, x_7) = 0 \end{aligned} \tag{2.3}$$

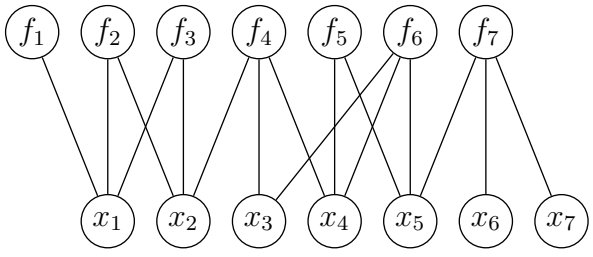


Figura 2.3: Representação do sistema de equações (2.3) na forma de um grafo bipartido.

Como pode ser observado na relação entre o sistema (2.3) e a Figura 2.3, o grafo não considera os valores ou formato das expressões, apenas é relevante a relação equação-variável. Esta consideração é a essência da análise ou depuração estrutural.

2.4 Detecção de Singularidades

Utilizando a representação de sistemas de equações na forma de grafos, como apresentado na Seção 2.3, a singularidade estrutural de um sistema NLA pode ser verificada de uma forma muito eficiente através da solução de um problema de máxima cardinalidade (Subseção 2.2.3).

Se a associação ótima do problema for também perfeita (cobrir todos os nós), então este é dito estruturalmente não-singular. Caso contrário, uma singularidade estrutural é detectada. Um sistema estruturalmente singular é garantidamente numericamente singular, porém o inverso não é verdadeiro.

Como exemplo, considere-se o sistema de equações (2.3), o qual pode ser representado pelo grafo da Figura 2.3. Uma associação de máxima cardinalidade para este sistema é apresentada na Figura 2.4. Como pode ser visto nesta figura, a associação não é perfeita, os nós f_1 e x_7 (grifados) não foram cobertos pela associação final. Isto significa que o sistema é estruturalmente singular, portanto este sistema não possui solução numérica ou esta não é única.

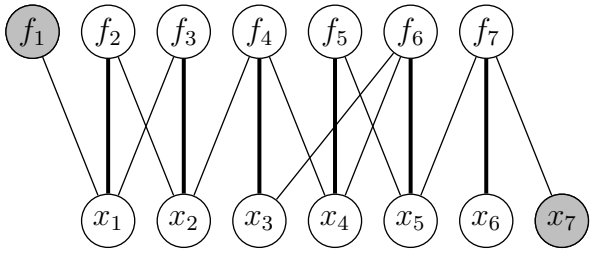


Figura 2.4: Associação de máxima cardinalidade para o sistema de equações (2.3) com os nós não cobertos grifados.

É importante notar que uma associação de máxima cardinalidade não é única. Ou seja, é possível que uma associação envolvendo um conjunto diferente de arestas resulte na mesma cardinalidade. Neste sentido, diz-se que uma associação de máxima cardinalidade é apenas ótima, mas não única.

Por exemplo, o sistema de equações (2.3) admite diversas soluções além da apresentada na Figura 2.4. Uma segunda solução para o problema de associação ótima para o sistema (2.3) pode ser vista na Figura 2.5.

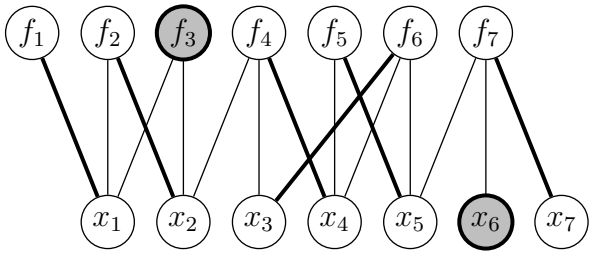


Figura 2.5: Uma segunda associação de máxima cardinalidade para o sistema de equações (2.3).

2.5 Decomposição Dulmage-Mendelshon

Apenas a detecção de uma singularidade, como apresentado na Seção 2.4, não é o suficiente para depurar um problema singular. Primeiramente, porque os casos que podem gerar uma singularidade são muitos:

1. O número de variáveis n não é igual ao número de equações m;
2. A existência de uma equação que não envolve nenhuma variável;
3. A existência de uma variável que não aparece em nenhuma equação;
4. Existe um subsistema de equações sobre-determinado;
5. Existe um subsistema de equações subdeterminado;
6. Existem equações ou variáveis que são linearmente dependentes.

Além disto, apenas detectar qual ou quais dos problemas acima citados estão presentes não é a informação ideal para que o usuário de um simulador baseado em equações alcance a solução. Porém, utilizando as informações estruturais presentes no sistema de equações (2.3) é possível dizer que:

- O sub-conjunto de equações $\{f_1, f_2, f_3\}$ sobre-determina as variáveis x_1 e x_2, portanto uma das equações deve ser removida;

- O sub-conjunto de variáveis $\{x_6, x_7\}$ está subdeterminado, é necessário adicionar uma equação envolvendo uma destas variáveis.

Informações como estas podem ser obtidas aplicando a decomposição canônica desenvolvida por Dulmage e Mendelsohn (apud ASHCRAFT; LIU, 1998), sendo mais tarde conhecida como decomposição Dulmage-Mendelsohn (DM). A decomposição DM para um grafo G parte de uma associação ótima M^{max}, como apresentada na Subseção 2.2.3, e é capaz de particionar o sistema em três: subsistema sobre-determinado G^+, subsistema bem determinado G^w e subsistema sub-determinado G^-.

De forma sucinta, a decomposição DM de um grafo G parte de uma associação ótima M^{max} qualquer e decompõe G em três, da seguinte forma:

- G^+ conjunto de todos os nós tocados por caminhos alternantes em G com respeito à M^{max} partindo-se de nós de equações v_e expostos por M^{max};

- G^- conjunto de todos os nós tocados por caminhos alternantes em G com respeito à M^{max} partindo-se de nós de variáveis v_v expostos por M^{max};

- $G^w = G \setminus (G^+ \cup G^-)$.

Nas seções 2.5.1, 2.5.2 e 2.5.3, grafos direcionados são utilizados para fazer a partição de sistema de equações de uma forma mais didática do que a presente nas definições acima.

2.5.1 Sub-sistemas sobre-determinados

Para isolar a parte sobre-determinada de um sistema de equações, se esta existe, parte-se de uma associação ótima M^{max} de um grafo bipartido G e converte-se o mesmo em um *grafo direcionado*, da seguinte forma:

- Todas as arestas de G não cobertas por M^{max} são substituídas por arestas direcionadas no sentido das equações V_e para as variáveis V_v;

- As arestas de G pertencentes à M^{max} são substituidas por arestas direcionadas no sentido das variáveis V_v para as equações V_e.

Aplicando-se este procedimento à associação ótima apresentada na Figura 2.4, obtém-se o grafo direcionado da Figura 2.6.

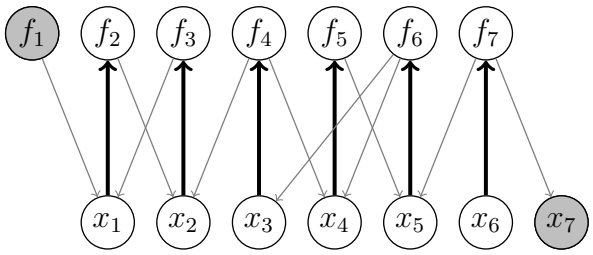


Figura 2.6: Conversão de uma associação ótima em um grafo direcionado para detecção da parte sobre-determinada.

Utilizando-se o grafo direcionado e partindo-se dos nós de equações v_e não cobertos por M^{max}, o conjunto de todos os nós que podem ser tocados pelos caminhos direcionados forma a partição sobre-determinada. Para o caso da Figura 2.6, o nó f_1 é o único nó de equação livre. Partindo-se deste, os nós que podem ser tocados pelos caminhos direcionados são, na ordem, x_1, f_2, x_2 e f_3. Disto conclui-se que conjunto de variáveis $\{x_1, x_2\}$ é sobre-determinado pelo conjunto de equações $\{f_1, f_2, f_3\}$.

2.5.2 Sub-sistemas sub-determinados

De forma análoga ao procedimento apresentado na Subseção 2.5.1, é possível revelar a parte sub-determinada de um sistema de equações. Na verdade, o problema a ser resolvido é simétrico ao anterior, portanto o sentido das arestas precisa ser invertido.

Com as arestas direcionadas invertidas, o grafo da Figura 2.6 fica como apresentado na Figura 2.7. Iniciando-se pelos nós de variáveis v_v não cobertos por M^{max}, o conjunto de todos os nós que podem ser tocados pelos caminhos direcionados forma a partição sub-determinada.

Para o caso da Figura 2.7, o nó x_7 é o único nó de variável livre. Iniciando-se a procura por este nó, a única variável que pode ser tocada é x_6 e a única equação incluída é f_7. Assim, o conjunto sub-determinado é formado por $\{x_6, x_7\}$ e $\{f_7\}$.

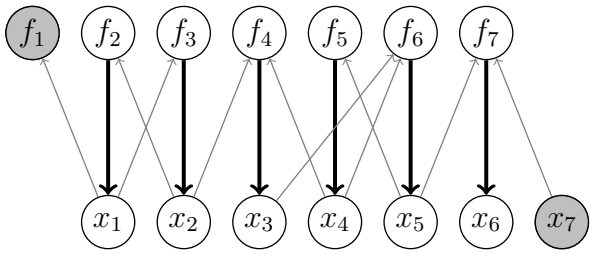


Figura 2.7: Conversão de uma associação ótima para um grafo direcionado para detecção da parte sub-determinada.

2.5.3 Conjunto Determinado

Por fim, o conjunto de equações e variáveis que não incluem-se nas partes sub e sobre-determinadas formam a parte *determinada*. Bunus (2002) e Bliek et al. (1998) chamam esta partição de *well-constrained*.

Assim, o grafo resultante da decomposição DM aplicada ao sistema de equações (2.3)

aparece na Figura 2.8. Note-se que a partição determinada é quadrada, envolvendo equações e variáveis em igual número e que contém uma associação perfeita ou completa.

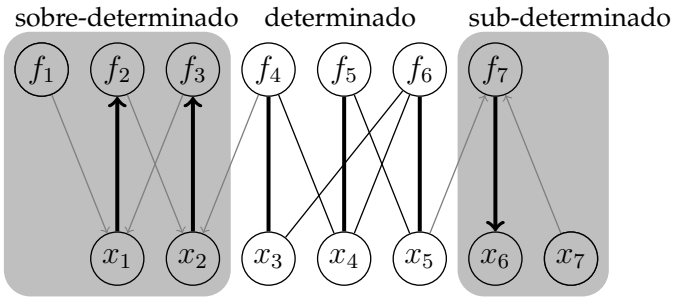


Figura 2.8: Decomposição de Dulmage-Mendelsohn ao sistema de equações (2.3).

Sobre a partição determinada cabem algumas considerações:

- A partição determinada não envolve nenhuma variável da partição sub-determinada. Isto significa que a parte determinada é independente da parte sub-determinada, mas o inverso não é verdadeiro;
- A partição determinada *pode* depender da partição sobre-determinada.

Das considerações acima, pode-se dizer que a partição bem determinada, embora contenha uma associação completa, não é necessariamente estruturalmente bem posta, diferentemente do afirmado por Bunus (2002). Como conseqüência, esta partição não pode ser resolvida independentemente das demais equações e variáveis do sistema em todos os casos. Este fato pode ser observado no grafo da Figura 2.8, onde a equação f_4 depende da variável x_2, variável esta que não faz parte do conjunto determinado.

O fato da partição determinada não poder ser resolvida independentemente para todos os casos gera uma discussão interessante sobre a nomenclatura *partição determinada* ou *well-constrained*. Esta nomenclatura é amplamente utilizada na literatura (BLIEK et al., 1998; BUNUS, 2002), mas talvez a forma mais adequada seja *partição*

restante como utilizado por Ashcraft e Liu (1998). Entretanto, se a partição sobre-determinada é nula, então do ponto de vista estrutural a partição determinada pode ser resolvida de forma independente da partição sub-determinada.

É importante notar que o resultado da decomposição DM é único, embora utilize como ponto de partida uma associação de máxima cardinalidade que é apenas ótima mas não única (DULMAGE; MENDELSOHN, 1958 apud ASHCRAFT; LIU, 1998).

Também é interessante observar o quão sensível pode ser a estrutura de um sistema de equações. Por exemplo, se a equação f_3 do grafo apresentado na Figura 2.8 envolvesse a variável x_3, a parte bem determinada seria nula. O resultado da decomposição DM para este caso pode ser visto no grafo da Figura 2.9.

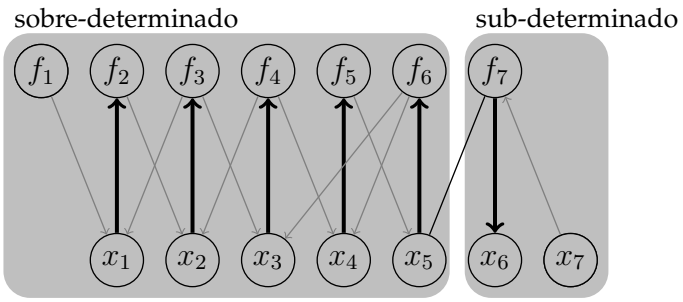


Figura 2.9: Decomposição DM para o sistema de equações (2.3) considerando-se a relação adicional $f_3 - x_3$.

O conjunto bem determinado pode ainda ser particionado em conjuntos menores, conhecidos como *componentes fortes*. O algoritmo mais conhecido para a determinação dos componentes fortes é o algoritmo de Tarjan (DUFF; REID, 1978). Quando o grafo é representado na forma matricial, a decomposição em componentes fortes é interpretada como uma permutação que torna a matriz bloco triangular inferior. Tal permutação é muito interessante, pois permite que, na solução numérica, cada componente forte seja resolvido separadamente. Esta técnica também é conhecida como *decomposição em blocos*.

2.6 Depuração

Como citado no Capítulo 1, a simulação de processos utilizando simuladores baseados em equações pode apresentar diversos problemas. Um exemplo simples é a determinação de quais variáveis podem ou não se fixadas para contemplar os graus de liberdade. Técnicas que auxiliem na solução deste tipo de problema foram chamadas de *depuração*.

Como ilustrado na Figura 2.8, a decomposição DM particiona qualquer sistema de equações em três seções distintas. Cada uma destas partes pode ser nula ou até mesmo englobar o sistema como um todo. Por exemplo, sistemas bem postos contêm apenas a parte bem determinada.

Problemas de simulação de processos podem envolver milhares de equações e, dependendo da estrutura do problema, as partes sub ou sobre-determinadas também podem envolver um número muito elevado de variáveis. Neste contexto fica claro que um particionamento mais detalhado das seções sub e sobre-determinadas é necessário para que a singularidade possa ser mais facilmente corrigida pelo usuário.

Por outro lado, mesmo para o caso de um número pequeno de variáveis, alguma das partições pode acabar englobando o sistema inteiro, fazendo com que as informações provenientes da decomposição DM não sejam tão úteis.

2.6.1 Depuração de Sistemas Sub-Determinados

Para virtualmente todos os modelos na área da engenharia o número de equações é menor que o número de variáveis (RICO-RAMIREZ; WESTERBERG, 2002). Nestes

casos a decomposição DM sempre fornecerá uma parte sub-determinada e, para que se tenha uma solução, existem algumas opções:

1. Desconsiderar a parte sub-determinada e resolver apenas a parte bem determinada;
2. Remover variáveis da parte sub-determinada;
3. Fornecer equações adicionais.

O caso 1 é a solução trivial porém não é usual, uma vez que o conjunto bem determinado pode ser muito pequeno ou até mesmo nulo. No trabalho de Bunus (2002) a possibilidade da remoção de variáveis (caso 2) para solucionar um problema sub-determinado também é considerada. Porém, do ponto de vista de modelagem, não faz sentido algum remover uma variável de uma equação. No presente trabalho a opção de remover variáveis de um modelo é desconsiderada, excetuando-se o caso em que a variável é isolada no grafo (não aparece em nenhuma equação). Assim, a alternativa considerada para a correção de um sistema sub-determinado é a adição de equações, restando como incógnita quais as variáveis devem ser envolvidas nestas restrições adicionais.

Como exemplo considere-se um sistema de troca térmica com um *bypass*, como apresentado na Figura 2.10. Neste exemplo apenas o fluxo de massa é considerado, gerando o seguinte sistema de equações:

$$\begin{aligned} f_1 &: x_1 = x_2 + x_3 \\ f_2 &: x_2 = x_4 \\ f_3 &: x_3 = x_5 \\ f_4 &: x_6 = x_4 + x_5 \end{aligned} \tag{2.4}$$

onde as variáveis $x_1 \ldots x_6$ são as vazões.

O grafo direcionado para a determinação da partição sub-determinada do sistema de equações (2.4) é apresentado na Figura 2.11. Como pode ser visto, os caminhos no

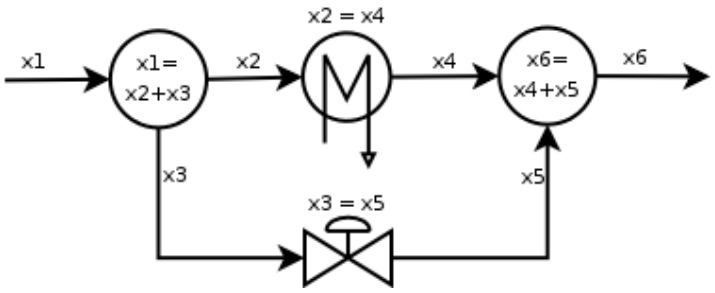


Figura 2.10: Sistema de troca térmica com um *bypass*.

sentido das arestas direcionadas que partem dos nós de variáveis expostos, x_5 e x_6, acabam por tocar todos os nós do grafo.

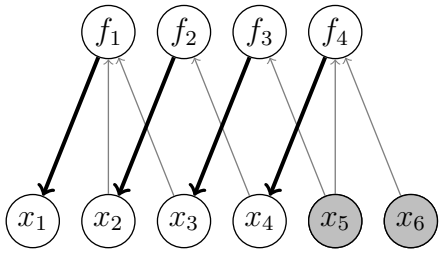


Figura 2.11: Grafo direcionado para o sistema de equações (2.4) para determinação da parte sub-determinada.

Neste caso, a decomposição DM não trouxe informações úteis, mas a reportagem da singularidade pode ser feita de uma forma mais detalhada. Pois, cada nó de G não coberto pela associação ótima M^{max} induz um subgrafo por caminhos alternantes com relação à M. Sendo que estes subgrafos, que juntos formam o conjunto sub-determinado, podem ou não conter uma intersecção.

O grafo da Figura 2.8, analisado na Seção 2.5, contém apenas uma variável livre, neste caso apenas um sub-grafo pode ser formado, o que não trará informações adicionais. Já no caso da Figura 2.11, onde há dois nós expostos pela associação ótima considerada, a reportagem utilizando os subgrafos com raiz em cada um destes nós é mais interessante:

- Iniciando-se em x_5 pode-se chegar em $\{x_1, x_2, x_3, x_4\}$;
- Iniciando-se em x_6 pode-se chegar em $\{x_1, x_2, x_4\}$.

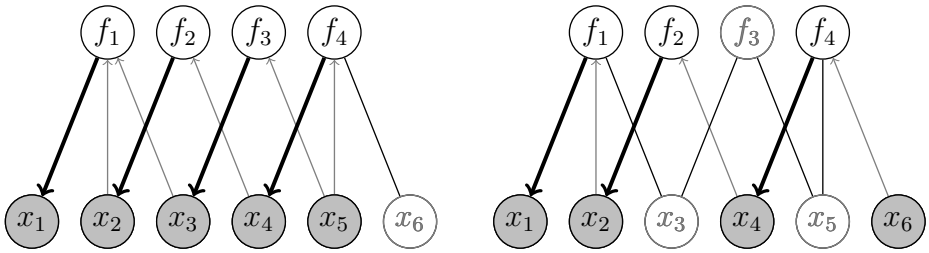


Figura 2.12: Subgrafos induzidos por nós de variáveis livres para o sistema (2.5) .

Estes subgrafos estão representados por arestas direcionadas na Figura 2.12, as variáveis que podem ser tocadas pelos caminhos alternativos estão grifadas. Assim, para a solução do problema, é necessário adicionar duas equações ao sistema:

- Uma envolvendo pelo menos uma das variáveis de $\{x_1, x_2, x_3, x_4, x_5\}$;
- Outra com pelo menos uma das variáveis de $\{x_1, x_2, x_4, x_6\}$.

Reportar os subgrafos individualmente ao invés da parte sub-determinada como um todo é uma forma muito mais conveniente para o usuário de um sistema baseado em equações. Neste trabalho vamos chamar este procedimento de decomposição DM *detalhada*. Esta idéia foi explorada no trabalho de Bunus (2002), que trata da depuração estrutural de sistemas de equações.

2.6.1.1 Depuração Fina do Conjunto Sub-determinado

A decomposição DM detalhada já é um avanço quando comparada com a decomposição usual. Porém, as informações adicionais que ela traz são apenas uma nova forma de reportar algo já revelado pela forma usual.

Na seção anterior, foi apresentado como resultado da decomposição detalhada para o exemplo do trocador de calor (Figura 2.10) a necessidade da adição de duas novas equações ao sistema. Porém, os conjuntos de variáveis elegíveis para estas

equações ainda se apresentou muito próximo ao número total de variáveis do sistema. Outro fato que contribui negativamente é que algumas combinações entre os dois conjuntos possíveis de variáveis levam à sistemas estruturalmente singulares, para o problema da Figura 2.10 este é o caso quando:

1. São adicionadas duas equações envolvendo apenas a mesma variável, uma vez que a combinação dos conjuntos admite isto para o caso de x_1, x_2 e x_4;

2. Uma equação envolvendo x_2 e outra envolvendo x_4.

O caso 1 é de solução muito simples, bastaria remover a intersecção dos conjuntos. O caso 2 está representado na Figura 2.13, onde pode ser visto que a adição da equação f_5 envolvendo x_2 e da equação f_6 envolvendo x_4 não é capaz de gerar uma associação perfeita. Como pode ser visto na Figura 2.13, os nós grifados f_6 e x_6, estão expostos pela associação apresentada, a qual é ótima. Neste caso, se torna clara a necessidade de um procedimento que remova condições como estas do conjunto de especificações viáveis.

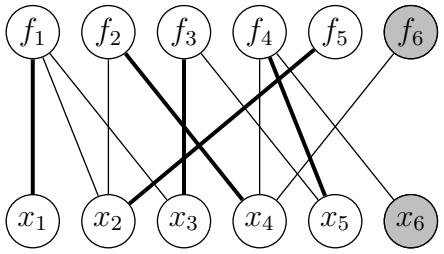


Figura 2.13: Associação ótima quando adiciona-se ao sistema (2.4) uma equação para x_2 e outra para x_4.

Do ponto de vista de modelagem fica fácil observar que especificar simultaneamente x_2 e x_4 ou x_3 e x_5 para o sistema (2.4) não faz sentido. Mais uma vez se torna clara a necessidade de um procedimento automático que detecte estas situações.

Note-se que o par x_3 e x_5 já não faz parte das combinações possíveis, mas isto é apenas uma coincidência, uma vez que o par x_2 e x_4, que faz parte do conjunto de

combinações elegíveis, apresenta um problema análogo do ponto de vista de modelagem. Na Figura 2.14 a simetria do problema se torna mais clara.

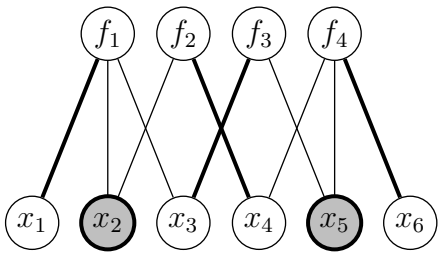


Figura 2.14: Uma associação ótima explorando a simetria do sistema de equações (2.4).

Não foi encontrado na literatura nenhum trabalho que tivesse identificado este problema, e conseqüentemente, delineado alguma solução. A alternativa mais simples seria verificar se cada uma das combinações possíveis admite uma associação perfeita. Porém, a complexidade na solução deste problema seria muito elevada, pois para cada uma das possibilidades seria necessário resolver um problema de associação ótima que apresenta uma complexidade $O(nm)$.

Analisando-se o problema de outra forma, para os casos em que a associação ótima é única, então o conjunto viável de especificações é único e já está determinado pelo conjunto de variáveis expostas pela associação. Quando a associação ótima não é única, cada uma das associações ótimas gera um conjunto de especificações viáveis. Então, para se obter todos os conjuntos viáveis, basta enumerar todas as associações ótimas admitidas pelo grafo do sistema. Uno (2001) apresenta uma forma muito eficiente para enumerar todas as associações perfeitas de um grafo quando uma associação perfeita já é conhecida, tendo uma complexidade de apenas $O(\log n)$ por associação.

A adaptação do algoritmo proposto por Uno (2001) para trabalhar com associações ótimas que não são perfeitas parece ser uma alternativa interessante para a enumeração de todas as associações ótimas de um grafo de forma eficiente. A verificação da viabilidade e a implementação de tais modificações não foram executadas neste tra-

balho e ficam como sugestão para trabalhos futuros.

2.6.2 Depuração de Sistemas Sobre-determinados

A maioria dos modelos para a representação de processos físicos apresenta uma parte sub-determinada, como apresentado na Subseção 2.6.1, uma vez que o número de variáveis usualmente é maior que o número de equações. Entretanto, um problema típico no processo de desenvolvimento de modelos é a formação de subsistemas sobre-determinados.

A presença de uma seção sobre-determinada inviabiliza a utilização do modelo. Já no caso sub-determinado, as informações pendentes podem ser adicionadas em níveis superiores na hierarquia da modelagem.

Para a correção de uma sobre-determinação a solução é a remoção de equações da parte sobre-determinada G^+, revelada pela decomposição DM (Subseção 2.5.1). O número de equações que deve ser removido é igual ao número de equações não cobertas pela associação ótima inicial M^{max}. Na verdade, a decomposição detalhada e decomposição fina, apresentados na Subseção 2.6.1 para o caso sub-determinado, podem ser utilizadas para a correção de sobre-determinações.

2.7 Análise Numérica

Infelizmente, a análise fundamentada apenas na informação estrutural nada pode ajudar no caso de singularidades de origem puramente numérica. Como exemplo con-

sidere o seguinte sistema de equações não-lineares:

$$\begin{aligned} f_1(x_1, x_2) &= 0 \\ f_2(x_1, x_2) &= 0 \\ f_3(x_2) &= \epsilon x_3 \end{aligned} \tag{2.5}$$

O grafo bipartido para o sistema (2.5), assim como as duas associações ótimas admitidas por ele, podem ser vistos na Figura 2.15.

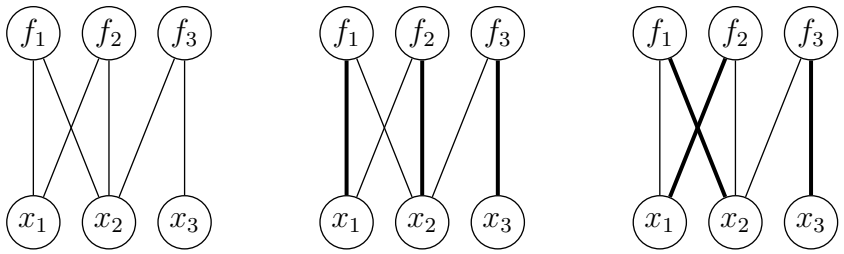


Figura 2.15: Grafo bipartido representando o sistema de equações (2.5) e suas possíveis associações ótimas.

Uma vez que uma associação ótima para o sistema (2.5) também é perfeita, este é estruturalmente não-singular. Porém um sistema estruturalmente não-singular ainda pode ser numericamente singular.

A parametrização de modelos é uma técnica comum para a construção de modelos genéricos, sendo que alguns parâmetros são nulos para a simulação de problemas mais simples. Por exemplo, se o parâmetro ϵ do sistema (2.5) for nulo, o sistema não admitirá uma solução, apresentando tanto uma parte sobre quanto sub-determinada. Outro caso que pode gerar uma singularidade numérica no sistema (2.5) é se as equações f_1 e f_2 forem linearmente dependentes. Em ambos os casos, a singularidade numérica pode ser detectada pela deficiência de posto da matriz Jacobiana.

Como pôde ser observado ao longo da Seção 2.5, as informações que a decomposição DM revela são de grande valia. É desejável obter informações semelhantes quando a singularidade é apenas numérica. Isto pode ser feito através da verificação

se as arestas consideradas no grafo são realmente elementos não nulos da matriz Jacobiana F_y. Claramente esta verificação depende do ponto atual y da solução.

Capítulo 3

Sistemas Algébrico-Diferenciais

Sistemas algébrico-diferenciais surgem naturalmente na modelagem dinâmica de processos, principalmente se estes são gerados por uma ferramenta baseada em equações. Entretanto, estes sistemas apresentam dificuldades adicionais no seu tratamento, quando comparados aos bem conhecidos sistemas de equações diferenciais ordinárias. Neste capítulo, algumas diferenças e semelhanças entre estas duas categorias de sistemas de equações são apresentadas. Além disto, uma introdução à teoria envolvida na solução de sistemas algébrico-diferenciais é apresentada.

3.1 Introdução

Um sistema algébrico-diferencial (*Differential-Algebraic Equations - DAE*) genérico pode ser representado por:

$$F(t, y, y') = 0 \tag{3.1}$$

onde t é a variável independente (usualmente o tempo), $F \in \Re^n$, y e $y' \in \Re^n$ são o vetor de variáveis dependentes e suas derivadas com relação à t, respectivamente.

Se (3.1) pode ser escrito na forma de um sistema de equações diferenciais ordinárias

(*Ordinary Differential Equations - ODE*):

$$y' = f(t, y) \tag{3.2}$$

com os mesmos estados y, então (3.1) na verdade é um ODE implícito. A condição necessária e suficiente para que (3.1) possa ser convertido na forma (3.2) é que a matriz Jacobiana $F_{y'}$ seja não-singular.

Neste trabalho o interesse está voltado para sistemas onde esta conversão não é possível, e mesmo que possível, não é desejável. Existem diversas razões para tratar (3.1) diretamente em favor de uma conversão para a forma (3.2) (BRENAN et al., 1989).

A utilização de sistemas ODE para a predição do comportamento dinâmico de sistemas tem uma longa história, iniciando-se com Newton em 1671 (HAIRER; WANNER, 1996). Porém, apenas em meados de 1950 Curtiss e Hirschfelder (1958), estudando problemas de cinética química, publicaram os primeiros artigos que identificaram claramente as dificuldades na solução de ODEs *rígidos*. Cerca de 10 anos depois foi dito, nas palavras de G. Dahlquist, "na década de 60 todos tomaram ciência de que o mundo está cheio de problemas rígidos" (HAIRER; WANNER, 1996).

A teoria a cerca da solução de sistemas ODE é bem conhecida, entretanto para o caso de sistemas rígidos o problema permanece nebuloso, uma vez que sequer é claro como definir exatamente o termo *rigidez*. Por exemplo, o respeitado livro de Hairer e Wanner (1996) deliberadamente desiste de fornecer uma definição rigorosa para este termo. O único fato que é claro é que métodos para a solução de sistemas rígidos devem satisfazer restrições de estabilidade muito mais estreitas do que métodos adequados para sistemas não-rígidos (CASH, 2000).

As dificuldades na solução de problemas DAE iniciam-se pela sua grande semelhança com os sistemas ODE rígidos. Além disso, as restrições puramente algébricas

presentes nos sistemas DAE acarretam em ainda mais dificuldades.

3.2 Sistemas DAE *versus* ODEs

A maioria dos sistemas físicos são modelados mais *elegantemente* na forma de um DAE do que um ODE. As equações diferenciais se originam das leis de conservação, por exemplo, os balanços de massa, energia e momento. Equações constitutivas, restrições geométricas, forças motrizes, restrições de equilíbrio e outras contribuem com equações puramente algébricas.

Como exemplo considere-se o processo de um tanque misturador com aquecimento como representado na Figura 3.1, o qual pode ser modelado com o seguinte conjunto de equações:

$$\begin{array}{ll} M' = F - L & (a) \\ U' = Fh_f - Lh_l + Q & (b) \\ U = Mu & (c) \\ h = f_1(T, P) & (d) \\ h_f = f_2(T_f, P_f) & (e) \\ u = f_3(T, P) & (f) \end{array} \tag{3.3}$$

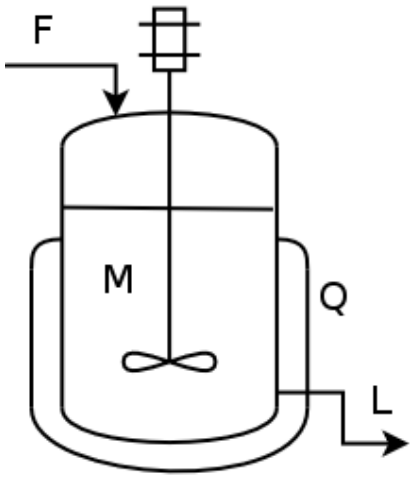


Figura 3.1: Tanque misturador com aquecimento.

O sistema de equações (3.3) envolve tanto equações diferenciais quanto algébricas. As equações (a) e (b) contribuem com a parte diferencial e as equações (c–f) com as restrições mais rígidas, as equações puramente algébricas. Claramente, para este sistema em particular, é possível fazer manipulações algébricas para eliminar as equações algébricas (c–f), gerando um sistema ODE. Porém, do ponto de vista de modelagem, esta conversão não é conveniente, pois na forma de um DAE:

- As equações são mais simples;
- As variáveis têm significado físico;
- Cálculos termodinâmicos, propriedades físicas, modelos cinéticos entre outros são mais facilmente incluídos e/ou substituidos quando necessário;
- Para o caso de modelos não-lineares a conversão para um ODE pode não ser possível;
- A conversão para um ODE pode destruir a esparsidade do problema original.

A modelagem do sistema de equações (3.3) na linguagem do simulador dinâmico de processos EMSO (SOARES; SECCHI, 2003) pode ser vista no Código 3.1. Como pode ser observado, a clareza deste código é extremamente beneficiada pela presença das equações algébricas.

Mas a preferência na utilização de sistemas DAE na substituição dos ODEs não é só uma questão de conveniência de modelagem. Murata (1996) apresenta um estudo indicando que é mais eficiente computacionalmente utilizar a formulação DAE em problemas típicos da engenharia química (onde o número de equações diferenciais é essencialmente menor que o número de equações algébricas).

Código 3.1: Modelo de um tanque misturador com aquecimento na linguagem do EMSO.

```
Model Tank
      PARAMETERS
outer   PP  as Plugin(Brief="Physical Properties");

      VARIABLES
in    F   as flow_mol(Brief="Inlet flow");
in    hf  as enth_mol(Brief="Inlet enthalpy");
in    Tf  as temperature(Brief="Inlet temperature");
in    Pf  as pressure(Brief="Inlet pressure");
out   L   as flow_mol(Brief="Outlet flow");
      U   as energy(Brief="Energy holdup");
      M   as mol(Brief="Molar holdup");
      T   as temperature(Brief="Tank temperature");
      P   as pressure(Brief="Tank pressure");
      h   as enth_mol(Brief="Specific enthalpy");
      u   as enth_mol(Brief="Specific internal engergy");

      EQUATIONS
      diff(M) = F - L;
      diff(U) = F * hf - L * h + Q;
      U = M * u;
      h = PP.LiquidEnthalpy(T, P);
      hf = PP.LiquidEnthalpy(Tf, Pf);
      u = PP.LiquidEnergy(T, P);
end
```

3.2.1 Condições Iniciais

Classicamente a simulação dinâmica consiste em: dado um estado y_0 em um ponto conhecido da variável independente t_0, utilizar um modelo para determinar o valor dos estados em um determinado horizonte de t. Esta técnica é conhecida como solução de *problema de valor inicial*.

Na solução do problema de valor inicial (*Initial Value Problem - IVP*) de um ODE (3.2), partindo-se de um conjunto de valores $y(t_0) = y_0$ arbitrários em um determinado ponto t_0, é possível obter uma descrição contínua de todas as variáveis y em t. Na verdade, para que isto seja verdade é necessário que $f(t, y)$ respeite algumas condições (HAIRER; WANNER, 1996), omitidas aqui por simplicidade.

A primeira grande diferença entre um ODE e um DAE é que, para o caso de um DAE, as equações algébricas impõem restrições que não permitem que o valor de $y(t_0)$ seja arbitrário. Portanto, para a solução de um IVP de um DAE é necessário encontrar um conjunto de valores que respeite as restrições algébricas. Uma *condição inicial consistente* respeita as restrições algébricas e o procedimento para encontrar tal conjunto é conhecido como *inicialização*.

O problema da inicialização de sistemas DAE pode ser formulado como segue. No sentido matemático, fornecida a informação suficiente sobre o estado inicial (condições iniciais) que especifique unicamente a solução do DAE, determinar completamente os valores iniciais $y(t_0)$ e $y'(t_0)$ (BRENAN et al., 1989). A informação sobre a condição inicial que é suficiente, no sentido matemático, representa os *graus de liberdade dinâmicos* ou o número de *estados*.

Em um ODE todas as variáveis do sistema são *estados*, uma vez que, para qualquer

ponto em t, qualquer conjunto de valores y é capaz de gerar uma solução que respeite o conjunto de equações (3.2). Para os casos mais simples de sistemas DAE, o número de estados é igual ao número de equações diferenciais presentes no sistema (SOARES, 2003). Mas, para o caso geral, a determinação do número de estados de um sistema DAE já é um problema complexo.

3.3 O índice de sistemas DAE

Na Seção 3.2 foram discutidas algumas semelhanças e diferenças entre os sistemas DAE e ODE. A definição de pontos chave como o *índice* e a resolubilidade de DAEs evoluiu durante muitos anos. Grosseiramente, a propriedade conhecida como *índice* é utilizada para medir a *distância* de um DAE para um ODE.

O índice é de grande importância na classificação do comportamento numérico de um DAE (BRENAN et al., 1989). A título de ilustração considere-se o seguinte sistema DAE semi-explícito:

$$\begin{aligned} x' &= f(x, y, t) \\ g(x, y, t) &= 0 \end{aligned} \tag{3.4}$$

Se o subsistema de equações g for diferenciado com respeito à variável independente t, pode-se formar o novo sistema de equações:

$$\begin{aligned} x' &= f(x, y, t) \\ g'(x, y, t) &= g_x(x, y, t)x' + g_y(x, y, t)y' + g_t(x, y, t) = 0 \end{aligned} \tag{3.5}$$

onde, g_x, g_y e g_t são matrizes Jacobianas.

Note-se que se g_y for não-singular o sistema (3.5) será um ODE implícito. Neste caso, o sistema (3.4) é dito de índice um, uma vez que o derivando apenas uma vez foi possível obter um ODE. No caso em que g_y é singular, pode-se dizer que o sistema (3.4)

é de índice elevado (maior que a unidade) e serão necessárias manipulações algébricas e novas diferenciações para que se obtenha um ODE (BRENAN et al., 1989). A formalização da metodologia exemplificada neste procedimento é apresentada na definição abaixo:

Definição 3.1 *(BRENAN et al., 1989) O índice diferencial ν_d é o número mínimo de vezes que todo ou um subgrupo de equações de um sistema DAE $F(y, y', t) = 0$ precisa ser diferenciado, em relação à variável independente t, até ser transformado em um ODE.*

Obviamente, um ODE (3.2) tem índice diferencial igual a zero. Sistemas DAE com índice igual a dois ou superior são conhecidos como sistemas de índice elevado.

A literatura apresenta outras definições de índice, que representam a mesma essência, mas cada definição é baseada em um critério diferente de resolubilidade. Referências para diferentes definições de índices de sistemas algébrico-diferenciais podem ser encontradas em Unger et al. (1995).

O índice de um sistema DAE está intimamente ligado com a dificuldade na sua solução numérica. Mas, como pôde ser observado no caso acima, qualquer sistema de índice elevado, que seja passível de solução, pode ser representado em outra forma de índice inferior. Porém é importante ressaltar que a conversão através da diferenciação gera um sistema que admite muito mais soluções que o sistema original, o que pode gerar anomalias na solução como apresentado na Subseção 5.5.1. Além disso, a necessidade de derivações para a formação de um ODE sugere que a solução de um DAE pode requerer a diferenciabilidade de algumas de suas equações.

3.4 Métodos para solução de sistemas DAE

Nenhuma das técnicas numéricas conhecidas é adequada para todos os tipos de sistemas DAE (BRENAN et al., 1989). Sistemas DAE de baixo índice são tratáveis com códigos projetados inicialmente para a solução de sistemas ODE, porém a inicialização apresenta dificuldades que não estão presentes nos ODEs (PANTELIDES, 1988a).

Para a solução numérica de sistemas DAE são utilizados basicamente duas categorias de métodos: passos múltiplos e passo simples (Runge-Kutta implícito). Entre estas duas categorias existe um dilema, os métodos de passos múltiplos geram implementações extremamente eficientes, mas sofrem de uma severa degradação de estabilidade com o aumento da ordem. Os métodos Runge-Kutta implícitos apresentam excelentes propriedades de estabilidade mas a eficiência na implementação é um grande desafio (CASH, 2000).

3.4.1 Métodos de Passos Múltiplos

Os métodos baseados em BDF (*backward differentiation formulas*) se tornaram os mais populares para a solução de sistemas DAE e, conseqüentemente, a classe de método de passos múltiplos mais conhecida (BRENAN et al., 1989). O método BDF mais simples é o método de Euler implícito, o qual consiste na substituição da derivada presente no sistema (3.1) por uma diferença retroativa ou para trás:

$$F(t_n, y_n, \frac{y_n - y_{n-1}}{h}) = 0 \tag{3.6}$$

onde, $h = t_n - t_{n-1}$.

O sistema (3.6) é um sistema de equações não-lineares em y_n para cada passo n, o qual é usualmente resolvido por um método tipo Newton. Como ilustração, considere-

se o seguinte sistema de índice 3:

$$\begin{aligned} x_1' &= x_2 \\ x_2' &= x_3 \\ x_1 &= g(t) \end{aligned} \tag{3.7}$$

A solução analítica deste problema é $x_1 = g(t)$, $x_2 = g'(t)$, $x_3 = g''(t)$. Pode-se notar que, para a solução de (3.7), a força motriz $g(t)$ precisa ser duas vezes diferenciável. Discretizando-se o sistema (3.7) segundo o método de Euler Implícito obtém-se:

$$\begin{aligned} x_{1,n} &= g(t_n) \\ x_{2,n} &= (x_{1,n} - x_{1,n-1})/h \\ x_{3,n} &= (x_{2,n} - x_{2,n-1})/h \end{aligned} \tag{3.8}$$

Utilizando-se (3.8), x_1 pode ser determinado exatamente em todos os passos, até mesmo para uma condição inicial $x_{1,0}$ não consistente ($x_{1,0} \neq g(t_0)$). Porém, para uma condição inicial não consistente, $x_{2,1}$ e $x_{3,1}$, obtidos após o primeiro passo discreto, são incorretos. Depois de dois passos, $x_{2,2}$ terá uma precisão $O(h)$, mas $x_{3,2}$ ainda será incorreto. No terceiro passo $x_{3,3}$ será $O(h)$ preciso.

O exemplo (3.8) ilustra bem a relação entre o índice e a solução numérica dos sistemas DAE. Pode-se notar que para sistemas de índice elevado ($\nu > 1$), a solução numérica converge em um intervalo *distante* do tempo inicial (BRENAN et al., 1989).

O método BDF em k-passos consiste em substituir y' pela derivada de um polinômio que interpola a solução em $k+1$ pontos $t_n, t_{n-1}, \ldots, t_{n-k}$, gerando:

$$F(t_n, y_n, \frac{1}{h}\sum_{i=0}^{k} \alpha_i y_{n-i}) = 0 \tag{3.9}$$

onde, α_i são os coeficientes do método. O método BDF em k-passos é estável para $k < 7$.

Seguindo o trabalho de Gear (1971) os primeiros códigos BDF foram implementados. Na década de 80 a segunda geração de códigos BDF surgiram, principalmente

pela identificação da importância dos sistemas DAE em muitos campos científicos e da engenharia (BRENAN et al., 1989). Mais especificamente, estes códigos são DASSL (PETZOLD, 1983), LSODI (HINDMARSH, 1980) e LIMEX (DEUFLHARD et al., 1987). DASSL atualmente é um dos códigos mais difundidos, por sua reputação de bom desempenho e por não impor restrições na estrutura do DAE, tratando problemas da forma totalmente implícita (3.1). Os códigos, LSODI e LIMEX, resolvem apenas problemas na forma *linearmente implícita* $A(t, y)y' = f(t, y)$. Uma implementação mais atual, inspirada no código do DASSL, esta presente no pacote SUNDIALS (HINDMARSH et al., 2005).

Um dos principais motivos de sucesso dos métodos BDF em passos múltiplos é que esta categoria não sofre nenhuma redução de ordem para problemas com $\nu \leq 1$. Isto significa que para sistemas DAE de baixo índice tal método alcança a mesma ordem de convergência obtida na solução de ODEs.

Entretanto, os códigos citados iniciam seu processo de integração utilizando uma fórmula de primeira ordem (3.6). Como uma implicação disto, tais códigos falham na integração até mesmo de um sistema com coeficientes constantes, desde que este tenha $\nu > 1$. Claro que, se o índice é conhecido *a priori*, pode-se projetar um código que inicia com uma fórmula de maior ordem, o que seria mais custoso em termos de memória e eficiência. Um exemplo de implementação de código tipo BDF capaz de resolver problemas com $\nu \leq 3$ é o código MEBDF (CASH, 2000).

3.4.2 Métodos de Passo Simples

Métodos Runge-Kutta implícitos (IRK, *implicit Runge-Kutta*) podem ser utilizados diretamente para a solução de sistemas DAE (3.1). Para o caso de problemas com índice elevado, o método pode sofrer redução de ordem. Neste caso métodos com ordem

elevada, por exemplo RADAUIIA, são preferidos. Exemplos de implementação são os códigos RADAU5 (HAIRER; WANNER, 1999) e PSIDE (LIOEN et al., 1998).

Nos anos 80 os métodos IRK foram muito utilizados para a solução de problemas ODE rígidos. Segundo Brenan et al. (1989), é provável que estes métodos se tornem muito importantes na solução numérica de DAEs. Por exemplo, para problemas onde as descontinuidades são freqüentes, os métodos IRK tem uma potencial vantagem sobre os métodos de passos múltiplos. Devido a sua natureza de passo único, mesmo quando acontece uma descontinuidade, passos de alta ordem podem ser executados, enquanto que nos métodos de passos múltiplos cada descontinuidade significa reduzir a ordem e o tamanho do passo.

3.5 Análise Estrutural de Sistemas DAE

No Capítulo 2, as técnicas disponíveis na literatura, juntamente com alguns novos desenvolvimentos, para a depuração de sistemas não-lineares foram apresentados. Para o caso de sistemas algébrico-diferenciais, a literatura apresenta vários trabalhos que abordaram o problema da resolubilidade através da caracterização e redução de índice.

No trabalho de Duff e Gear (1986) os autores sugerem uma análise estrutural capaz de identificar se um sistema semi-explícito é de índice 0, 1 ou 2. Gear (1988) propôs um algoritmo simbólico para a redução de índice e, baseado nesta idéia, Bachmann et al. (1990) apresentaram um algoritmo para redução de índice de sistemas DAE lineares. Pantelides (1988a) introduziu um algoritmo baseado na teoria de grafos que aborda o problema da inicialização consistente de sistemas DAE genéricos. A idéia básica deste algoritmo é a detecção de sub-conjuntos de equações estruturalmente singulares e sua remoção através de derivação destes sub-conjuntos. Estendendo o trabalho de Bach-

mann et al. (1990), Unger et al. (1995) desenvolveram um algoritmo capaz de caracterizar e reduzir o índice de sistemas DAE genéricos. Esse algoritmo, além de determinar o número de graus de liberdade dinâmicos, verifica se um conjunto de valores iniciais para as variáveis permite uma inicialização consistente.

Murata (1996) desenvolveu um código utilizando o *software* de computação simbólica MAPLE para a caracterização de sistemas DAE. O sistema também é capaz de gerar as funções de resíduos e matrizes de iteração na forma de códigos otimizados em FORTRAN para utilização com os integradores DASSL e RADAU5.

No trabalho de Costa Jr. (2003), o algoritmo de Pantelides foi aplicado para a resolução direta de sistemas de índice elevado. Neste tabalho, as diferenciações sugeridas pelo algoritmo de Pantelides são executadas utilizando-se o código de derivação automática ADOLC (GRIEWANK et al., 1996) e a integração é feita com o DASSLC (SECCHI, 2005).

Murota (2000) demonstrou como determinar o índice estrutural de sistemas DAE utilizando associações de custo ótimo (*maximum-weight matching*). Esta técnica, diferentemente das demais, é capaz de tratar diretamente sistemas que contenham derivadas de ordem maior que 1 e se mostra muito eficiente para determinar o índice. Entretanto, o método não revela o número de graus de liberdade dinâmicos nem como reduzir o índice do problema. Leitold e Hangos (2001) também abordam os temas de resolubilidade, determinação e redução do índice estrutural com base na teoria de grafos.

Poulsen (2001) também estuda a análise estrutural de sistemas DAE com uma aplicação do algoritmo de Pantelides. Neste trabalho, uma notação matricial, em substituição dos grafos, é utilizada. Para a solução de DAEs de índice elevado é proposta a formação de um ODE equivalente e os efeitos desta metodologia são investigados.

Como pôde ser observado, a maioria dos trabalhos é baseada na análise estrutural, desconsiderando os valores numéricos, exatamente como tratado na Seção 2.3. Desta consideração tem origem a propriedade conhecida como índice estrutural ν_s.

Definição 3.2 *O índice estrutural ν_s é o número mínimo de vezes que todo ou um subgrupo de equações de um sistema DAE $F(y, y', t) = 0$ precisa ser diferenciado, em relação à variável independente t, até que $F_{y'}$ seja estruturalmente não-singular.*

Também fica claro que, historicamente, quando se trata de sistemas DAE, a análise de resolubilidade se confunde muito com o conceito de índice (Seção 3.3) e da consistência de condições iniciais (Subseção 3.2.1). Hoje, a técnica de caracterização proposta por Pantelides é a mais difundida. Entretanto, diferentemente do tratamento considerado pelos autores citados na revisão aqui apresentada, a depuração para sistemas DAE não se restringe à análise da consistência de condições iniciais e ao problema de índice. No Capítulo 4, é apresentado como as técnicas desenvolvidas para a depuração de sistemas não-lineares (Capítulo 2) podem ser utilizadas para produzir informações importantes na depuração de sistemas DAE.

Nas seções que seguem, o algoritmo desenvolvido por Pantelides (1988a) é apresentado. No Capítulo 4, uma adaptação desta técnica é proposta a qual remove uma deficiência presente no método original e faz com que o procedimento revele mais informações sobre o sistema de equações.

3.5.1 Sistemas DAE na forma de Grafos

Na Seção 2.3 foi apresentado como um sistema de equações não-lineares pode ser representado como um grafo bipartido. De forma análoga, sistemas algébrico-diferenciais

também podem ser representados como grafos bipartidos.

Como exemplo, considere-se o seguinte sistema de equações (PANTELIDES, 1988a):

$$\begin{aligned} f_1(x, u_1, u_2) &= 0 \\ f_2(x, x', y_1) &= 0 \\ f_3(x, y_2) &= 0 \end{aligned} \tag{3.10}$$

onde, em um contexto de controle ótimo de processos, y_1 e y_2 são as saídas desejadas (conhecidas), u_1 e u_2 são as entradas do sistema e x é uma variável dependente.

Uma vez que as variáveis y_1 e y_2 são conhecidas, por simplicidade, estas não serão consideradas na análise. O mesmo resultado poderia ser obtido adicionando-se equações que envolvessem estas variáveis ao sistema (3.10). Baseado nestas considerações, o grafo presente da Figura 3.2 representa o sistema (3.10).

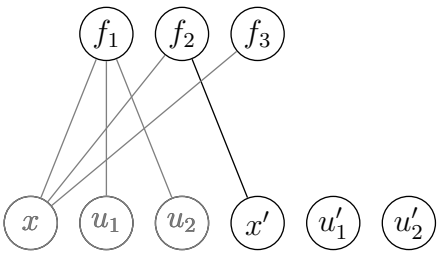


Figura 3.2: Grafo para o sistema algébrico-diferencial (3.10).

O grafo apresentado na Figura 3.2 é análogo aos utilizados no Capítulo 2. Porém, além dos nós para as variáveis também são considerados os nós para as derivadas das variáveis em relação a t.

3.5.1.1 Derivada Estrutural

Os métodos de análise estrutural estudados neste trabalho necessitam muitas vezes derivar equações. Para isto define-se a *derivada estrutural* da seguinte forma: a derivada estrutural de uma equação envolve todas e somente as derivadas das variáveis envolvi-

das pela equação original. Por exemplo, derivando-se estruturalmente a função f_1 do grafo apresentado na Figura 3.2 obtém-se o grafo apresentado na Figura 3.3.

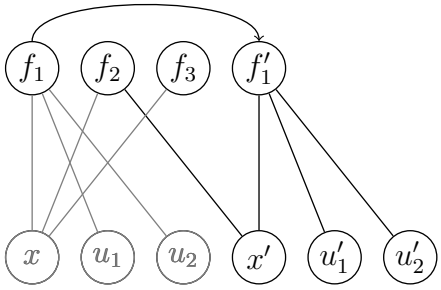


Figura 3.3: Grafo para o sistema algébrico-diferencial (3.10) com a equação f_1 derivada estruturalmente.

Como pode ser observado, a equação original f_1 envolvia as variáveis $\{x, u_1, u_2\}$, então sua derivada estrutural f_1' envolverá $\{x', u_1', u_2'\}$. Note-se que a derivada estrutural implica em uma simplificação do problema real. Dependendo da forma de f_1, a equação derivada f_1' poderia envolver alguma das variáveis $\{x, u_1, u_2\}$.

3.6 Algoritmo de Pantelides

A técnica estrutural mais difundida atualmente para a caracterização e redução de índice foi proposta no trabalho de Pantelides (1988a). O objetivo inicial deste trabalho era a obtenção de condições iniciais consistentes para sistemas DAE genéricos. Mais tarde, muitos trabalhos utilizaram-se desta técnica para a redução de índice (POULSEN, 2001; Costa Jr., 2003). A redução de índice permite, em princípio, que códigos projetados para sistemas de índice até 1, sejam utilizados para a solução de problemas de índice elevado.

Diferentemente da forma preferida neste trabalho para a representação de um DAE genérico (3.1), Pantelides utiliza a seguinte forma:

$$F(t, x, x', y) = 0 \tag{3.11}$$

onde, x e $x' \in \Re^n$, $y \in \Re^m$ e $F \in \Re^{n+m}$.

Note-se que em (3.11), diferentemente de (3.1), existe uma distinção entre as variáveis. Para as variáveis x a derivada aparece explicitamente no sistema (variáveis diferenciais), já as variáveis y aparecem apenas na forma algébrica (variáveis algébricas).

A idéia de Pantelides parte de uma particularidade dos sistemas DAE. A solução de um DAE deve respeitar não somente as equações originais do sistema mas também as derivadas deste. Para um ODE, derivadas das suas equações nunca vão impor restrições adicionais à solução, mas para o caso de DAEs isto pode ocorrer. Quando as derivadas das equações de um DAE impõem restrições adicionais, então diz-se que o sistema continha restrições *escondidas*.

Do ponto de vista estrutural, se há um subconjunto das variáveis $\{x', y\}$ que seja sobre-determinado por um subsistema de equações, então a derivada deste subsistema impõem restrições adicionais. Pois, se há tal subsistema, a derivação deste gerará mais equações do que variáveis novas (derivadas de x' ou y). Desta forma, o algoritmo de Pantelides é baseado na determinação de subsistemas estruturalmente singulares (sobre-determinados) e suas derivações.

O texto do artigo onde Pantelides propõe seu método para a determinação de condições iniciais consistentes é de complicada compreensão. Porém, utilizando-se o Algoritmo 1 apresentado no Capítulo 2, o algoritmo de Pantelides pode ser descrito como apresentado no Algoritmo 3.

Comparando-se o Algoritmo 2 (utilizado para obter uma associação ótima) com o Algoritmo 3, pode-se notar que estes diferem apenas pelas linhas 3 e 5. Entre estas semelhanças e diferenças é importante notar os seguintes pontos:

Algoritmo 3 Pseudocódigo para o algoritmo de Pantelides.

$Pantelides(G = (V_e, V_v, E), M)$

1: $M \leftarrow \emptyset$
2: **for** $v_e \in V_e$ **do**
3: $V_v \leftarrow V_v \setminus x$, $E \leftarrow E \setminus \{*, x\}$
4: **if not** $AugmentMatching(G = (V_e, V_v, E), M, v_e)$ **then**
5: **diff all** *colored* $v_k \in V_e$
6: **end if**
7: *uncolour* V_e
8: **end for**

- O algoritmo de Pantelides também é baseado na procura por caminhos alternativos em profundidade (linha 4), conseqüentemente a associação final não é única;

- O algoritmo só termina quando todos os vértices de equações foram analisados (linha 2). Uma vez que na linha 5 podem ser adicionadas novas equações, dependendo da estrutura do sistema de equações, o algoritmo executará continuamente sem fim;

- Quando não é possível uma associação na linha 4, o conjunto de todas as equações que podem ser tocadas por caminhos alternantes que partem da equação atual v_e foi *colorido* pelo Algoritmo 1;

- Na linha 5 devem ser adicionadas derivadas das equações. Quando uma equação é diferenciada, considera-se que ela envolve as derivadas das variáveis da equação original. Neste procedimento é possível que equações envolvam variáveis que não apareciam originalmente no sistema, neste caso variáveis também devem ser adicionadas. Uma vez que variáveis são adicionadas, é necessário atualizar o conjunto x para removê-lo do grafo na linha 3.

3.6.1 Exemplo de Aplicação

Com o intuito de tornar mais claro o funcionamento do Algoritmo 3, nesta seção o algoritmo é aplicado para a análise de um problema simples.

Considere-se um modelo descrevendo o comportamento de um reator químico contínuo onde ocorre uma reação exotérmica de primeira ordem e o calor gerado é removido por um sistema de resfriamento. As equação relevantes são (PANTELIDES, 1988a):

$$\begin{aligned} f_1 &: C' = K_1(C_0 - C) - R \\ f_2 &: T' = K_1(T_0 - T) + K_2R - K_3(T - T_c) \\ f_3 &: R = -K_4 e^{-K_5/T} C \end{aligned} \tag{3.12}$$

onde, C_0 e T_0 são a concentração e temperatura da corrente de alimentação do reator (conhecidos), C e T são a concentração e temperatura da corrente de produto, R é a taxa de reação, T_c é a temperatura do fluido refrigerante, K_1, K_2, K_3, K_4 e K_5 são constantes.

Se a temperatura T_c é conhecida, tanto a inicialização quanto a solução do sistema (3.12) é simples, típica de um DAE de índice 1. Mas o problema de controle ótimo, onde deseja-se determinar qual o perfil de temperatura T_c é capaz de gerar um dado perfil de concentração duas vezes diferenciável $u(t)$ da corrente de produto, C, é mais interessante:

$$f_4 : C = u(t) \tag{3.13}$$

O grafo bipartido para estas condições pode ser visto na Figura 3.4, onde tem-se $x' = \{C', T'\}$ e $y = \{R, T_c\}$.

No algoritmo de Pantelides a singularidade estrutural é testada com relação ao conjunto $\{x', y\}$. Então, conforme a linha 3 do Algoritmo 3, o conjunto de variáveis

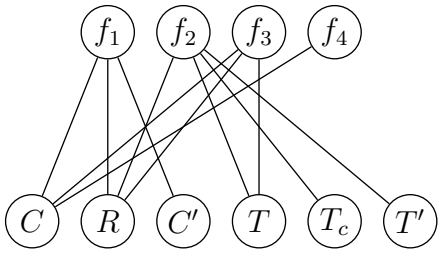


Figura 3.4: Grafo para o modelo de um reator exotérmico com controle ótimo.

x deve ser removido do grafo. O grafo da Figura 3.4 com o conjunto de variáveis x removido pode ser visto na Figura 3.5(a).

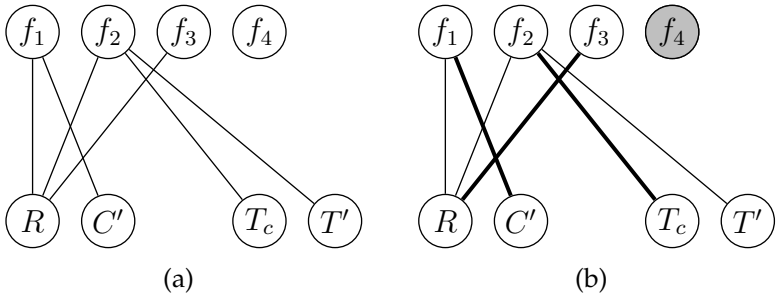


Figura 3.5: Grafo da Figura 3.4 com o conjunto x removido (a) e uma das duas associações ótimas possíveis (b).

Seguindo os passos do Algoritmo 3, a linha 4 retorna verdadeiro para as três primeiras equações. Partindo-se da equação f_4 não é possível incrementar a associação atual, como pode ser visto na Figura 3.5(b). Então todos os nós *coloridos* devem ser diferenciados para formar um novo sistema. Neste caso em particular, a equação f_4 está isolada, portanto nenhum nó além do inicial foi *colorido*. Os nós *coloridos* (que devem ser diferenciados) são destacados graficamente, como apresentado na Figura 3.5(b). Adicionando a equação f_4' ao sistema tem-se o grafo apresentado na Figura 3.6(a).

Neste ponto a equação atual é f_4', mais uma vez pode-se observar que não é possível aumentar a cardinalidade da associação partindo-se desta equação. Iniciando-se do nó f_4', passando por caminhos alternantes, é possível tocar os nós f_1 e f_3 (a linha 4 do Algoritmo 3 *colore* f_4', f_1 e f_3). Então, na linha 5 são adicionadas as equações f_4'', f_1' e f_3'. Juntamente com as derivadas das equações é necessário adicionar as variáveis R' e C''

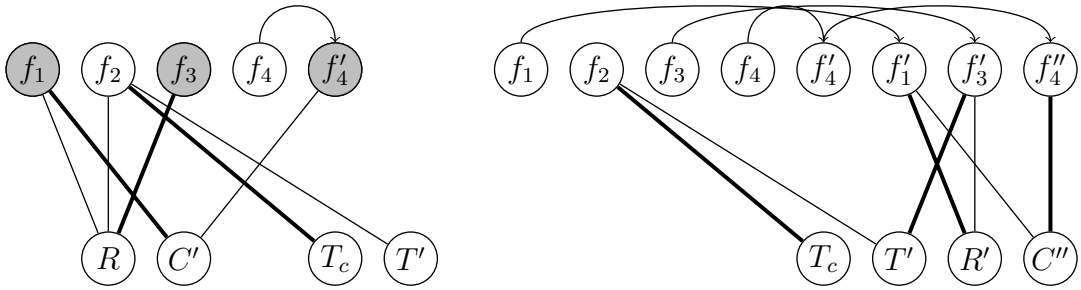


Figura 3.6: Primeira derivação no modelo de um reator exotérmico sugerida pelo algoritmo de Pantelides (a) e passo final (b).

e, conseqüentemente, remover os nós de variáveis R e C' (linha 3). Com o novo grafo formado, a linha 4 retorna verdadeiro para todas as equações restantes chegando-se ao fim do procedimento. A associação final pode ser vista na Figura 3.6(b)

Como pode ser visto na Figura 3.6(b), o sistema final contém 8 equações e também 8 variáveis $\{C, T, R, T_c, C', T', R', C''\}$. Portanto, embora haja duas equações diferenciais no sistema original (3.12), as restrições *escondidas* são tais que não é possível arbitrar o valor de nenhuma das variáveis na condição inicial.

O algoritmo de Pantelides permite uma redução até índice 1 e determina o índice como sendo 1 mais o máximo número de vezes que alguma equação foi diferenciada. Analisando-se o grafo apresentado na Figura 3.6(b), nota-se que a equação f_4 foi diferenciada duas vezes, ficando determinado como 3 o índice estrutural do sistema. Por exemplo, no trabalho de Costa Jr. (2003), as derivadas das equações e variáveis sugeridas pelo algoritmo são utilizadas para formar um sistema de baixo índice. Este procedimento, conhecido como *redução de índice*, permite, em princípio, a solução numérica do problema com códigos projetados para sistemas de baixo índice (Subseção 5.5.1). No simulador de processos EMSO, um procedimento de redução de índice que utiliza derivação simbólica é utilizado para a solução de problemas de índice elevado (SOARES; SECCHI, 2003).

3.6.2 Falha na Detecção de Singularidades

Nesta seção uma grave falha do Algoritmo 3 é apresentada através da sua utilização em um exemplo simples. Considere-se o sistema de equações (3.10), utilizado na Subseção 3.5.1 para introduzir a notação de grafos para sistemas DAE. Para este sistema, as variáveis puramente algébricas y são $\{u_1, u_2\}$ e a única diferencial é x. Desta forma, o grafo resultante da remoção das variáveis, de acordo com a linha 3 do Algoritmo 3, é apresentado na Figura 3.7(a).

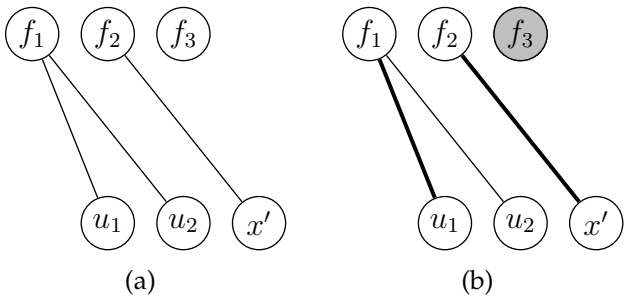


Figura 3.7: Grafo para o sistema algébrico-diferencial (3.10) com a variável x removida (a) e uma das duas associações ótimas admitidas (b).

Seguindo os passos do Algoritmo 3, a linha 4 retorna verdadeiro para as duas primeiras equações. Partindo-se da equação f_3 não é possível incrementar a cardinalidade da associação atual, como pode ser visto na Figura 3.7(b). Então, a linha 5 instrui a adicionar a equação f_3' ao sistema, como apresentado na Figura 3.8(a).

Passando para a próxima equação (f_3'), a linha 4 falha novamente. Neste ponto as equações f_3' e f_2 foram *coloridas*, uma vez que há o caminho $\{f_3' - x' - f_2\}$ na Figura 3.8(a). Adicionando-se as derivadas destas equações juntamente com a variável x'' e conseqüente remoção do nó x' forma-se o grafo apresentado na Figura 3.8(b).

Seguindo o algoritmo, a equação f_2' pode ser associada, mas a linha 4 falha novamente para a equação f_3'', *colorindo* f_2' e f_3''. Entretanto, o último subgrafo *colorido* $\{f_2' - x'' - f_3''\}$, apresentado na Figura 3.8(b), é essencialmente o mesmo subgrafo

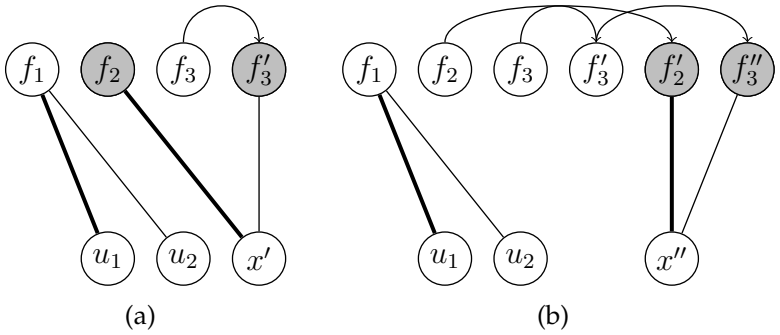


Figura 3.8: Grafos da evolução do Algoritmo 3 para o sistema (3.10).

$\{f_2 - x' - f_3'\}$, presente na Figura 3.8(a). Como uma consequência disto, o algoritmo irá continuar diferenciando este mesmo subgrafo infinitamente revelando uma falha grave do método.

O trabalho de Murata (1996) cita a verificação do posto estrutural de $F_y + F_{y'}$ como uma alternativa para evitar este problema de recursão infinita do Algoritmo 3. De fato, o problema (3.10), utilizado para exemplificar esta deficiência do algoritmo, não apresenta posto estrutural cheio para a matriz $F_y + F_{y'}$.

Desta forma, para evitar a recursão infinita no Algoritmo 3, uma verificação prévia do posto da matriz $F_y + F_{y'}$ precisa ser executada. O algoritmo proposto neste trabalho no Capítulo 4 faz esta verificação como parte de sua lógica.

3.6.3 Inicialização

O objetivo original do algoritmo de Pantelides é a inicialização consistente para sistemas DAE. Quando o algoritmo termina com sucesso, as restrições *escondidas* são reveladas juntamente com o número de graus de liberdade dinâmicos. Assim, adicionando-se equações (condições inicias) em um número igual ao de graus de liberdade dinâmicos, um sistema quadrado de equações não-lineares pode ser formado.

A título de demonstração, considere-se o sistema DAE semi-explícito (3.4):

$$\begin{aligned} x' &= f(x, y, t) \\ g(x, y, t) &= 0 \end{aligned}$$

Segundo a notação utilizada por Pantelides, têm-se n variáveis diferencias (x) e m variáveis que aparecem apenas na forma algébrica (y). Porém, para que o problema de inicialização seja resolvido, tem-se $2n + m$ incógnitas no sistema $\{x_0, x'_0, y_0\}$ e apenas $n+m$ equações. O algoritmo poderá ainda revelar m' restrições adicionais, dependendo da estrutura do problema. Assim, o número de graus de liberdade para a inicialização ou graus de liberdade dinâmicos é $n - m'$.

Claramente, para que o sistema tenha solução, m' está limitado por $m' \leq n$. Quando n e m' são iguais, como no caso apresentado na Subseção 3.6.1, nenhuma condição inicial pode ser fornecida. Entretanto, códigos típicos para a solução de problemas DAE necessitam de uma condição inicial onde são conhecidos não apenas $\{x_0, x'_0, y_0\}$, mas também y'_0. Por exemplo, DASSL (PETZOLD, 1983) resolve problemas na forma:

$$\begin{aligned} F(t, y, y') &= 0 \\ y(t_0) &= y_0 \\ y'(t_0) &= y'_0 \end{aligned} \tag{3.14}$$

onde, y_0 e y'_0 são valores conhecidos. Esta notação não diferencia as variáveis que aparecem apenas na forma algébrica, diferentemente da forma apresentada no sistema (3.11).

Para a determinação de y'_0, contendo variáveis que não aparecem explicitamente no sistema de equações original, tipicamente são utilizados métodos baseados em perturbações. Para o caso particular do DASSL, dado y_0 o código pode estimar y'_0 executando um passo suficientemente pequeno de um método tipo Euler implícito. Entretanto, se o vetor y_0 não for consistente esta metodologia falha. Neste sentido o algoritmo de Pantelides se mostra muito conveniente pois, utilizando as equações reveladas pelo método, juntamente com condições iniciais suficientes para determinar o sistema, y_0 pode ser determinado exatamente.

Se o algoritmo pudesse fazer a redução até índice zero, mais equações seriam adicionadas para que y_0' pudesse ser determinado diretamente. Neste caso, todas as variáveis poderiam ser determinadas diretamente pela solução de um problema não-linear. No Capítulo 4 é proposta uma modificação ao Algoritmo 3. Esta alteração permite fazer a redução até índice zero, revelando todas as equações necessárias para a determinação de todas as variáveis e suas derivadas diretamente.

3.6.4 Condições Iniciais Válidas

Como foi apresentado na Subseção 3.6.3, para um DAE semi-explícito (3.4) é necessário arbitrar $n - m'$ condições iniciais. Onde, n é o número de equações diferenciais e m' é o número de restrições adicionais reveladas pelo Algoritmo 3.

Fica claro que o algoritmo de Pantelides é muito adequado para a determinação do número de graus de liberdade dinâmicos. Porém, além do número de condições iniciais, é importante conhecer quais variáveis podem ter seus valores arbitrados e quais não podem ser envolvidas na condição inicial. Infelizmente, a associação resultante da aplicação do algoritmo de Pantelides não traz consigo estas informações.

Como exemplo, considere-se o seguinte sistema de equações:

$$\begin{aligned} x_1' - x_2' &= a(t) \\ x_2 &= b(t) \end{aligned} \tag{3.15}$$

A aplicação do Algoritmo 3 para o sistema (3.15) é apresentada na Figura 3.9. A associação final produzida pelo algoritmo é apresentada na Figura 3.9(d).

Como pode ser observado, a derivada da segunda equação ($x_2' = b'(t)$) é revelada como uma restrição adicional ao sistema. É interessante notar que o subsistema de

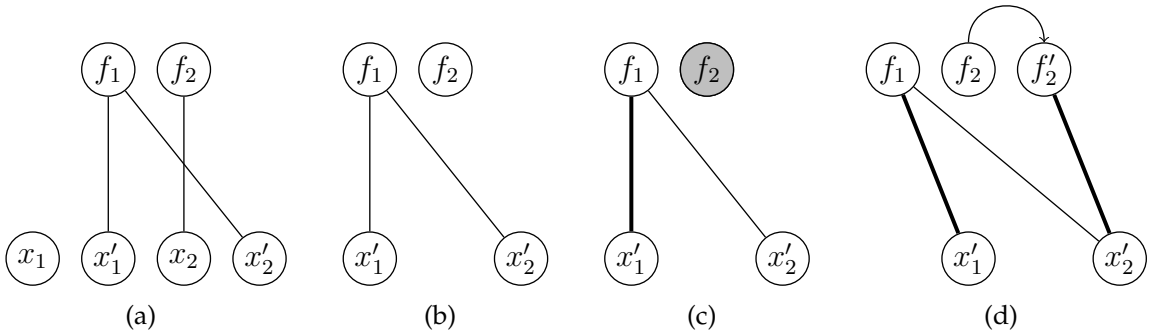


Figura 3.9: Seqüência de grafos da aplicação do Algoritmo 3 ao sistema (3.15).

equações $\{f_1, f_2'\}$ nas variáveis $\{x_1', x_2'\}$ é bem determinado. Ou seja, nenhuma das variáveis $\{x_1', x_2'\}$ pode ter seu valor arbitrado como condição inicial. Na verdade, para o sistema (3.15) a única variável que pode ter seu valor arbitrado como condição inicial é x_1. Informações como esta são de grande valia para o usuário de um sistema baseado em equações, infelizmente não é possível obter este detalhamento em nenhum dos grafos da Figura 3.9. No Capítulo 4 uma modificação ao Algoritmo 3 é proposta, tal modificação permite determinar exatamente quais variáveis podem participar da condição inicial.

3.6.5 Outras Deficiências da Análise Estrutural

Na Seção 2.7 já ficou claro que qualquer análise baseada apenas na estrutura tem limitações. Na caracterização e determinação do índice estrutural não poderia ser diferente. O fato de que o índice estrutural ν_s pode ser menor do que o índice diferencial ν é bem conhecido (UNGER et al., 1995; REISSIG et al., 2000).

Porém, até pouco tempo afirmava-se na literatura que o índice estrutural seria limitado por $\nu_s \leq \nu$ (UNGER et al., 1995). Contrariando o que se imaginava correto, Reißig et al. (2000) provaram que o algoritmo de Pantelides (Algoritmo 3) aplicado a um sistema DAE de índice 1 pode indicar um índice estrutural muito maior que a unidade

para uma determinada categoria de problema.

Um exemplo simples onde este tipo de problema aparece é o modelo para um circuito linear, como o apresentado na Figura 3.10. Este circuito consiste em uma fonte de voltagem $v(t)$, um resistor de resistência R e um capacitor de capacitância C.

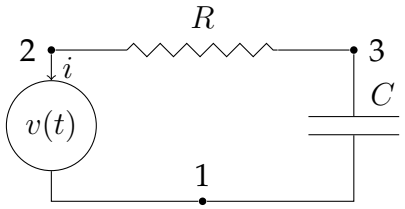


Figura 3.10: Circuito elétrico linear, onde o índice estrutural é maior que o índice.

Se o nó 2 é escolhido como o *terra*, a análise nodal modificada (*modified nodal analysis* - MNA) (CHUA et al., 1987 apud REISSIG et al., 2000) fornece o seguinte sistema de equações na forma matricial:

$$\begin{pmatrix} C & -C & 0 \\ -C & C & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} v_1' \\ v_3' \\ i' \end{pmatrix} + \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1/R & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_3 \\ i \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ v(t) \end{pmatrix} \tag{3.16}$$

onde, i é a corrente que flui no circuito e v_1 e v_3 são os potenciais nos nós 1 e 3, respectivamente.

O mesmo sistema também pode ser reescrito da seguinte forma:

$$\begin{aligned} &C(v_1' - v_3') = i \\ &C(v_1' - v_3') = v_3/R \\ &v_1 = v(t) \end{aligned} \tag{3.17}$$

O sistema (3.17) é de índice 1, isto é facilmente verificado se a expressão $C(v_1' - v_3')$ é substituída pela variável i na segunda equação. Feita esta substituição, se a segunda e terceira equações são diferenciadas 1 vez o sistema resultante será um ODE, portanto índice 1. A análise estrutural segundo o algoritmo de Pantelides para o sistema (3.16), pode ser observada na seqüência da Figura 3.11.

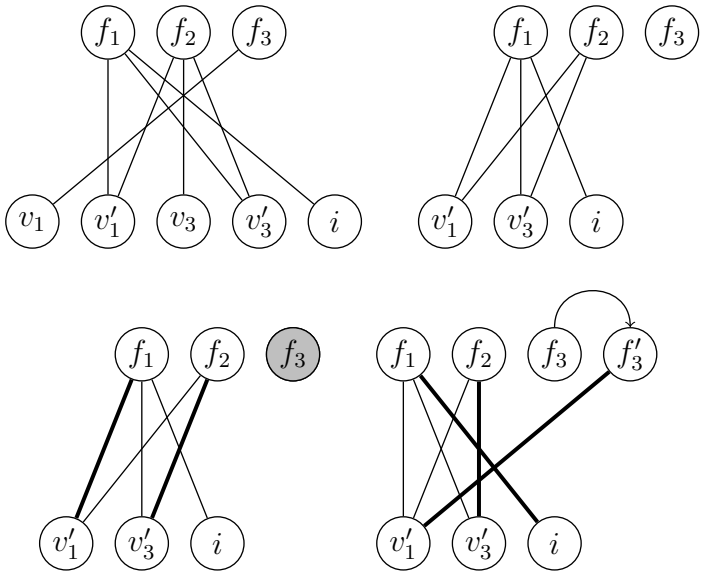


Figura 3.11: Seqüência de grafos da aplicação do Algoritmo 3 ao sistema (3.17).

Como pode ser visto na Figura 3.11, o algoritmo revela a necessidade de uma diferenciação para atingir uma formulação de índice estrutural 1, portanto o sistema é de índice estrutural 2. Este exemplo de fato prova a existência de problemas onde o índice estrutural supera em número o índice diferencial. Uma discussão sobre os impactos deste fenômeno na solução numérica desta categoria de problemas é apresentada na Subseção 5.5.3.

3.7 Os Estados de Sistemas DAE

O termo *estado* parece ser tão fundamental na modelagem e simulação de sistemas dinâmicos que são poucos os livros texto que apresentam uma definição formal para o seu significado.

Quando se trata de sistemas ODE, todas as variáveis são estados independentes do sistema, em termos de sua condição inicial. Entretanto, para o caso geral de sistemas

DAE, a presença de restrições algébricas faz com que nem todas as variáveis sejam independentes. Neste contexto é preciso definir o conceito de *estado* para o caso geral de sistemas DAE.

3.7.1 Formulação de Espaço de Estado

Na literatura voltada para o controle de processos, onde os modelos dinâmicos são fundamentais, as variáveis de um modelo ou processo são usualmente caracterizadas como sendo de três tipos (OGUNNAIKE; RAY, 1994):

Variáveis de entrada são as variáveis que, independentemente, estimulam o processo e podem, portanto, induzir modificações nas condições internas do processo;

Variáveis de saída são as variáveis que fornecem informações sobre as condições internas do processo;

Estados é o conjunto mínimo de variáveis para que as condições internas do processo sejam totalmente determinadas.

Utilizando-se um diagrama de blocos como o da Figura 3.12, também muito freqüente na representação de sistemas de controle, as definições acima são representadas. Os valores atuais das variáveis de saída y são função de valores anteriores dos estados x. Modificações nas variáveis de entrada u podem induzir modificações nos estados x e, por conseqüência, nas variáveis de saída y.

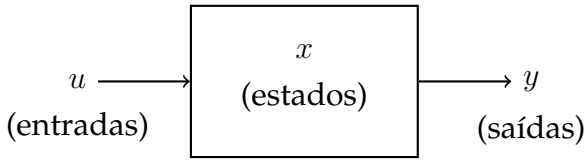


Figura 3.12: Definição de *estado* para diagramas de blocos.

Usualmente os conceitos acima apresentados estão associados a representação dos sistemas de equações na forma de *espaço de estado*:

$$\underset{\text{(linear)}}{\begin{cases} x' = Ax + Bu \\ y = Cx + Du \end{cases}} \quad \text{ou} \quad \underset{\text{(não-linear)}}{\begin{cases} x' = f(x,u) \\ y = h(x,u) \end{cases}} \tag{3.18}$$

nesta formulação, x é o vetor de estados, y e u são as saídas e entradas, respectivamente.

3.7.2 Caso Geral de Sistemas DAE

Entretanto, a formulação (3.18) é um caso muito especial, onde os valores iniciais de todos os elementos de x são independentes e x' e y estão isolados. Boa parte dos sistemas DAE não podem ser convertidos para a forma (3.18). A necessidade da extensão do conceito de estado, entradas e saídas para o caso geral de problemas DAE acabou por gerar diversas distorções dos conceitos. Por exemplo, no trabalho de (FEEHEY; BARTON, 1998), a seguinte forma para a definição de sistemas DAE é utilizada:

$$F(t, x, x', y, u) = 0 \tag{3.19}$$

onde os autores chamam x de variáveis de *estado diferenciais* e y são as variáveis de *estado algébricas*.

Para alguns casos, a forma (3.19) se assemelha muito com a representação de blocos da Figura 3.12 e com a formulação (3.18). Porém, as variáveis x' não aparecem na forma explícita como em (3.18) e todas as variáveis são chamadas de *variáveis de estado* sendo ainda categorizadas como algébricas ou diferenciais. Na forma utilizada por Pantelides (1988a), apresentada anteriormente como (3.11) e aqui reproduzida:

$$F(t, x, x', y) = 0$$

as variáveis x são conhecidas como variáveis *diferenciais* ou *dinâmicas*[1] e as variáveis y

[1]Não faz sentido chamar x de variáveis *dinâmicas*, uma vez que y também pode variar com o tempo.

chamadas de variáveis *algébricas*. Em tal trabalho o conceito de estado é abandonado em favor de um outro tipo de categorização – variáveis algébricas e diferenciais. Este fato levanta a questão da real viabilidade de estender o conceito de estado para o caso geral de sistemas DAE.

Na forma geral de sistemas DAE (3.1), preferida neste trabalho, não há distinção entre as variáveis que aparecem na forma diferenciada e as que aparecem apenas na forma algébrica:

$$F(t, y, y') = 0$$

Considerando-se as formas (3.11) e (3.19), para alguns casos mais simples, um conjunto qualquer de valores $x = x_0$ possibilita uma inicialização consistente do sistema. Isto pode levar à conclusão errônea de que x são os estados do sistema, o que não é verdade para o caso geral. Este equívoco pode ser ainda reforçado pela semelhança entre a representação de sistemas DAE nas formas (3.11) e (3.19) e a Figura 3.12.

A diversidade de nomenclaturas e formulações presentes na literatura para o tratamento dos *estados* indicam a necessidade de uma definição universal do conceito. Esta definição deve ser aplicável a sistemas DAE genéricos porém deve recair na definição atual quando se trata de sistemas ODE e dos sistemas na forma de espaço de estado.

Neste trabalho, propõem-se a manutenção da definição de *estado* exatamente como ela é aceita atualmente:

Definição 3.3 *Os* estados *de um sistema são o conjunto mínimo de variáveis que determinam completamente o sistema.*

Porém, diferentemente das tentativas anteriores, não se propõem uma formulação

na qual os estados apareçam como um conjunto distinto das demais variáveis. Mas sim, levanta-se a necessidade de, dado um sistema DAE qualquer, determinar quantos e quem são os estados do problema.

Este problema pode ser parcialmente resolvido com a técnica de análise estrutural de sistemas DAE mais conhecida. O Algoritmo 3, desenvolvido por Pantelides e apresentado na Seção 3.6, é capaz de determinar o número de condições iniciais necessárias para uma inicialização consistente de sistemas DAE. As condições iniciais necessárias para a inicialização consistente de um sistema DAE contém toda a informação necessária para determinar completamente o sistema. Portanto, estas devem ser igual, em número, aos estados. Os quais também coincidem com o número de graus de liberdade dinâmicos.

Entretanto, não é necessário conhecer apenas o número de estados de um determinado sistema ou modelo, mas sim, dentro do conjunto de todas as variáveis, quais são as possíveis combinações elegíveis para esta condição. Esta informação não é revelada com as técnicas disponíveis atualmente. No Capítulo 4 são propostos novos algoritmos que tornam possível determinar estas informações.

Capítulo 4

Depuração de Sistemas Algébrico-Diferenciais

Historicamente, a análise de resolubilidade de sistemas algébrico-diferenciais se ateve ao problema da determinação do número de graus de liberdade dinâmicos e ao problema de índice. Porém, todo o conhecimento sobre sistemas não lineares, apresentado no Capítulo 2, pode ser adaptado para o caso dinâmico. Neste capítulo é apresentado como esta adaptação pode ser feita. Para tanto, modificações à técnica clássica, estudada no Capítulo 3, são necessárias.

4.1 Introdução

No Capítulo 3 a análise estrutural para sistemas algébrico-diferenciais conhecida na literatura foi apresentada. Além disto, a técnica estrutural mais difundida, o algoritmo de Pantelides, foi estudada em detalhes e suas deficiências identificadas. Também foi visto que, historicamente, a análise de resolubilidade de sistemas DAE se ateve ao problema da determinação do número de graus de liberdade dinâmicos e ao problema de índice.

Se comparadas as técnicas apresentadas no Capítulo 2 (sistemas NLA) e no Capítulo 3 (sistemas DAE), pode-se notar que a depuração para sistemas não-lineares está muito mais avançada que a para o caso dinâmico. Talvez a análise estrutural de DAEs não tenha evoluído desde a apresentação do trabalho de Pantelides (1988a), pela própria forma com que este artigo foi escrito. Utilizando-se os pseudo-códigos apresentados no trabalho é muito simples implementar a técnica em duas funções com poucas linhas de código. Entretanto, o texto não faz questão de conectar o desenvolvimento com as bem conhecidas técnicas para sistemas não-lineares. No Capítulo 3, o algoritmo de Pantelides foi propositadamente apresentado com um enfoque diferente do original, deixando clara a sua ligação com as técnicas para sistemas lineares e não-lineares.

4.2 Novo Algoritmo

No Capítulo 3, as deficiências do algoritmo de Pantelides foram analisadas, e são sumarizadas abaixo:

1. O método não termina para alguns tipos de sistemas singulares, executando infinitamente (Subseção 3.6.2);

2. Para os casos onde o algoritmo finaliza com sucesso, nem todas as variáveis necessárias para a integração com os códigos clássicos podem ser determinadas diretamente (Subseção 3.6.3);

3. Embora o método seja extremamente conveniente para a determinação do número de graus de liberdade dinâmicos, este não revela quais são as variáveis que podem e não podem ser envolvidas na condição inicial (Subseção 3.6.4);

4. O método apresenta algumas outras deficiências provenientes da sua na-

tureza estrutural (Subseção 3.6.5).

Neste capítulo são propostas modificações ao algoritmo de Pantelides que permitem adaptar todo o conhecimento sobre sistemas não-lineares (Capítulo 2) para o caso dinâmico. Estas modificações trazem uma contribuição importante, pois removem as deficiências 1–3.

No trabalho de Soares (2003) a deficiência citada no item 3 já havia sido identificada. Em tal trabalho foi proposta uma nova técnica capaz de revelar todas as equações necessárias para a determinação de todas as variáveis envolvidas na inicialização (redução até índice zero). Entretanto, esta técnica é baseada em heurísticas e não foi possível provar uma limitação para a sua complexidade. Na prática, esta técnica se mostrou inadequada em termos de eficiência para problemas de índice elevado e de dimensão moderada a elevada (mais do que 1000 variáveis). Assim sendo, outras alternativas continuaram sendo estudadas até a convergência em um novo método, o qual pode ser considerado como a maior contribuição deste trabalho.

4.2.1 Exemplo Ilustrativo

A introdução do novo algoritmo, proposto neste trabalho, é mais interessante através de um exemplo ilustrativo. Considere-se novamente o problema (3.15):

$$\begin{aligned} x_1' - x_2' &= a(t) \\ x_2 &= b(t) \end{aligned}$$

No algoritmo de Pantelides as variáveis que apresentam a sua derivada no sistema devem ser removidas do grafo. Utilizando esta consideração, o sistema (3.15) fica como apresentado na Figura 4.1(a). A grosso modo, a remoção destas variáveis significa ignorar parte das informações presentes nas restrições algébricas.

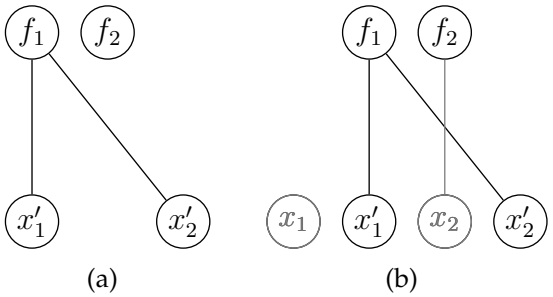


Figura 4.1: Comparação entre grafos para o sistema (3.15) para o método de Pantelides (a) e o proposto (b).

No método proposto neste trabalho, o sistema (3.15) é representado como presente na Figura 4.1(b). Nesta representação, todas as variáveis são consideradas, há apenas uma classificação entre variáveis algébricas (em cinza) e diferenciais (em preto). Nesta classificação, a presença ou não da variável no sistema é irrelevante. Por exemplo, a variável x_1 não aparece explicitamente no sistema de equações (3.15) mas é representada no grafo da Figura 4.1(b). Nestes casos, o nó da variável estará isolado no grafo.

Então, utilizando-se esta convenção considere-se o seguinte procedimento:

1. Tentar associar cada equação com uma variável *diferencial* única;

2. Se todas as equações foram associadas, o procedimento terminou, caso contrário as equações não associadas são *marcadas*;

3. Tentar associar cada equação com uma variável única, agora incluindo também as variáveis *algébricas*;

4. Derivar as equações *marcadas* e retornar para 1.

Aplicando-se este procedimento para o sistema (3.15), a equação f_1 pode ser conectada com x'_1 mas f_2 não pode ser conectada com variáveis *diferenciais*, então esta equação é *marcada*. Esta situação é representada na Figura 4.2(a).

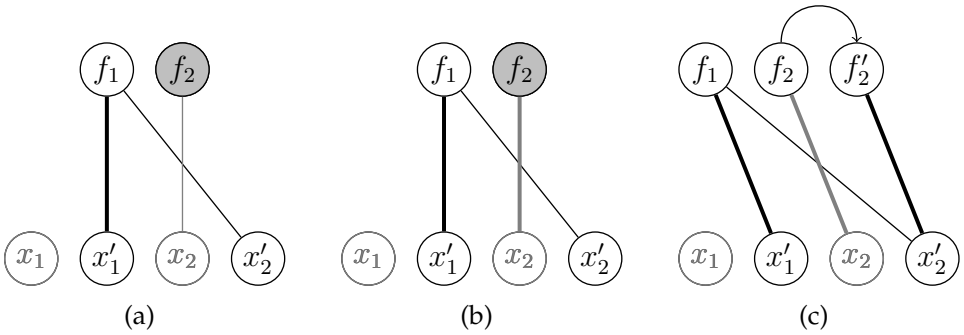


Figura 4.2: Evolução do algoritmo proposto para o sistema de equações (3.15).

Considerando-se também as variáveis *algébricas*, a conexão $f_2 - x_2$ é possível e então executada, como apresentado na Figura 4.2(b). Após, f_2 é derivada para formar a equação f_2' e então adicionada ao sistema. Na seqüência, f_2' pode ser conectada com a variável diferencial x_2' finalizando o procedimento, como apresentado na Figura 4.2(c).

Neste ponto é muito interessante comparar as Figuras 4.2 e 3.9 que são o resultado gerado pelo procedimento sendo proposto e o Algoritmo 3, respectivamente. A primeira figura se parece muito com os grafos analisados no Capítulo 2. Nesta figura, todas as equações estão associadas no final do procedimento, o que não acontece no caso da Figura 3.9. E o mais interessante, na Figura 4.2(c) existe um nó de variável não coberto (x_1), coincidindo exatamente com o número de graus de liberdade dinâmicos do sistema estudado. Além disto, x_1 está isolada no grafo sugerindo que esta é a única condição inicial viável, mais uma vez coincidindo com a discussão apresentada na Subseção 3.6.4.

4.2.2 Modificação dos Algoritmos

Para a implementação do método delineado na seção anterior o algoritmo para a procura por caminhos alternativos (Algoritmo 1) precisa ser modificado. Uma vez que existem duas possibilidades de procura: apenas variáveis *diferenciais* ou todas as vari-

áveis. A modificação necessária está presente no Algoritmo 4.

Algoritmo 4 Modificação do Algoritmo 1 para o novo método.

$AugmentMatching2(G = (V_e \cup V_v, E), M, v_e, alg)$

1: *colour* v_e
2: **if exists** $\{v_e, v_v\} \in E$ **and** $\{v_e, v_v\} \notin M$ **and** v_v *is elegible* **then**
3: $M \leftarrow M \cup \{v_e, v_v\}$
4: **return true**
5: **end if**
6: **for all** $\{v_e, v_v\} \in E$ **do**
7: **if exists** $\{v_{e2}, v_v\} \in M$ **and** v_{e2} **not** *colored* **and** v_v *is elegible* **then**
8: **if** $AugmentMatching2(G = (V_e \cup V_v, E), M, v_{e2}, alg)$ **then**
9: $M \leftarrow M \cup \{v_e, v_v\}$
10: **return true**
11: **end if**
12: **end if**
13: **end for**
14: **return false**

Os algoritmos 1 e 4 diferem em poucos pontos. O Algoritmo 4 tem o argumento adicional alg que é *verdadeiro* quando as variáveis algébricas podem ser envolvidas, no caso contrário *falso*. Existem verificações adicionais nas linhas 2 e 7, onde é feito o teste se a variável v_v é elegível para a associação. Estes testes adicionais (*is eligible*) funcionam da seguinte forma:

- Se alg é verdadeiro o teste é sempre verdadeiro
- Se alg é falso, o teste retorna verdadeiro apenas para variáveis diferenciais

Utilizando-se como base o Algoritmo 4, o procedimento apresentado no Algoritmo 5 pode ser utilizado para executar a metodologia ilustrada na Subseção 4.2.1.

O Algoritmo 5, proposto neste trabalho, é muito semelhante ao Algoritmo 3 (Pantelides). As principais diferenças podem ser resumidas nos seguintes pontos:

- A linha 3 do Algoritmo 3, onde algumas variáveis são removidas da análise, não existe no Algoritmo 5;

Algoritmo 5 Pseudocódigo do novo algoritmo estrutural para análise de sistemas algébrico-diferenciais.

$DAESystems(G = (V_e, V_v, E), M)$

```
1:  M ← ∅
2:  for v_e ∈ V_e do
3:    if not AugmentMatching2(G = (V_e, V_v, E), M, v_e, false) then
4:      mark all colored v_k ∈ V_e
5:      uncolour V_e
6:      if not AugmentMatching2(G = (V_e, V_v, E), M, v_e, true) then
7:        return false
8:      end if
9:      diff all marked v_k ∈ V_e
10:   else
11:     uncolour V_e
12:   end if
13: end for
14: return true
```

- A linha 7 do Algoritmo 5 detecta singularidades, evitando a recursão infinita presente no Algoritmo 3;

- No Algoritmo 5 existem dois pontos de recursão, a linha 3 representa a análise das equações diferenciais e a linha 6 é relativa à análise das restrições algébricas, a qual não existe no Algoritmo 3.

Estas diferenças de implementação atacam diretamente as deficiências presentes no algoritmo de Pantelides, identificadas na Seção 3.6. As seções que seguem ilustram como o novo algoritmo trata cada uma destas deficiências.

4.2.3 Detecção de Singularidades

Para ilustrar a capacidade na detecção de singularidades do novo algoritmo (Algoritmo 5), para os casos onde o algoritmo de Pantelides (Algoritmo 3) executa indefinida-

mente, considere-se novamente o sistema de equações (3.10):

$$f_1(x, u_1, u_2) = 0$$
$$f_2(x, x', y_1) = 0$$
$$f_3(x, y_2) = 0$$

O sistema (3.10) foi analisado pelo Algoritmo 3 na Subseção 3.6.2 quando pôde ser observado que o algoritmo executa indefinidamente para este caso.

A execução do Algoritmo 5 para o sistema de equações (3.10) é apresentada na Figura 4.3.

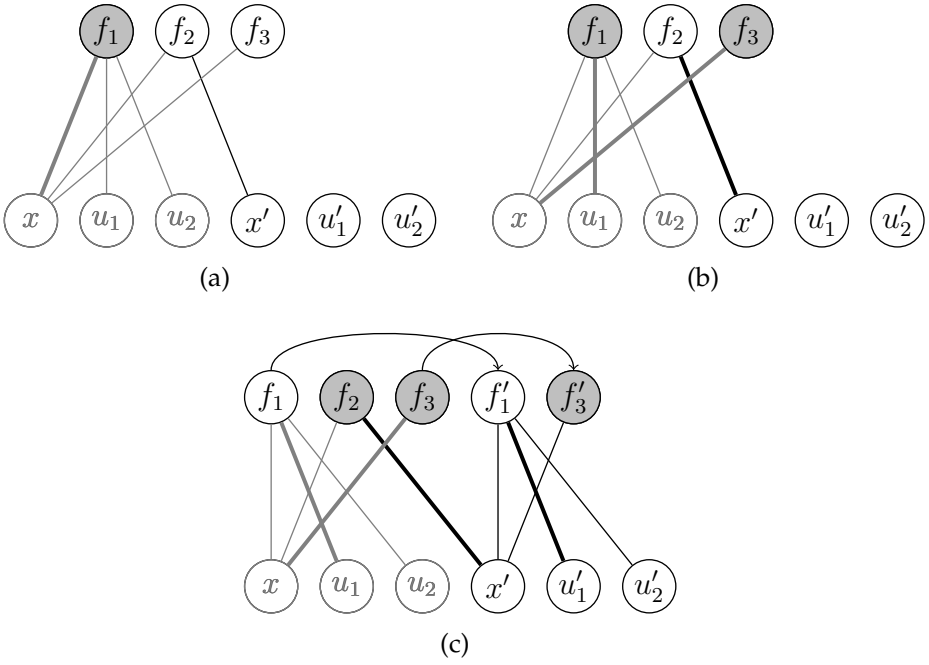


Figura 4.3: Evolução do Algoritmo 5 para o sistema de equações (3.10).

Como pode ser visto na Figura 4.3(a), uma conexão para a equação f_1 não é possível com as variáveis diferenciais, então esta é *marcada* para derivação. Já quando a conexão com as variáveis algébricas é permitida (linha 6) uma conexão com x é formada. Para a equação f_2, uma conexão com a variável x' é adicionada pela linha 3, como pode ser visto na Figura 4.3(b). Para a equação f_3 a linha 3 também falha e esta é *marcada*. Após, a linha 6 encontra o caminho alternativo $\{f_3 - x - f_1 - u_1\}$ sendo este adicionado como apresentado na Figura 4.3(b). Na seqüência, a equação f_1' é conectada diretamente com

u_1' (Figura 4.3(c)). Já para a equação f_3' não só a conexão com variáveis diferenciais (linha 3) falha mas também quando são incluídas as variáveis algébricas (linha 6). Isto faz com que a linha 7 seja alcançada e o procedimento terminado prematuramente.

Como pode ser observado, o Algoritmo 5 é capaz de detectar singularidades estruturais evitando a recursão infinita presente no Algoritmo 3.

4.2.4 Possíveis Condições Iniciais

Como discutido na Subseção 3.6.4, além de determinar o número de graus de liberdade dinâmicos é interessante saber quais variáveis podem ter seus valores arbitrados como condição inicial. Também já foi apresentado que o Algoritmo 3 não revela este tipo de informação. No trabalho de Pantelides (1988a), o autor comenta a possibilidade de aplicar um algoritmo de triangularização como o apresentado em Duff e Reid (1978) como uma forma de tornar visível o grupo de variáveis que pode fazer parte da condição inicial.

O exemplo apresentado na Subseção 4.2.1 para a introdução do novo método de análise estrutural para sistemas DAE já sugeriu que é possível determinar qual o conjunto de variáveis pode fazer parte da condição inicial. Mais precisamente, a combinação da decomposição Dulmage-Mendelshon (Seção 2.5) com o Algoritmo 5 é capaz de trazer esta informação.

Se o Algoritmo 5 retorna com sucesso para um dado sistema algébrico-diferencial, este não contém sobre-determinação. Assim, a aplicação da decomposição DM resultará na partição do sistema em apenas dois: uma parte sub-determinada e uma parte bem determinada. Os conjuntos de equações adicionais capazes de determinar a parte sub-determinada representam condições iniciais estruturalmente consistentes.

Deve-se observar que o conjunto de variáveis necessárias para a formação de uma condição inicial consistente não é único. Entretanto, estas devem, obrigatoriamente, fazer parte da partição sub-determinada (formada por todos os nós que podem ser tocados por caminhos alternantes que partem de variáveis não cobertas pela associação final).

Uma vez que a parte sub-determinada do sistema foi revelada, todo o conhecimento desenvolvido para depuração de sistemas NLA, apresentado na Subseção 2.6.1, pode ser utilizado para reportar os possíveis conjuntos de condições iniciais. Ou seja, a decomposição DM e as técnicas de depuração disponíveis para os modelos estáticos agora podem ser também utilizadas para o caso de sistemas dinâmicos representados por modelos algébrico-diferenciais. Este fato representa um avanço importante na depuração desta categoria de sistemas, deixando-os no mesmo patamar da depuração para os sistemas não-lineares.

4.2.5 Unicidade da Solução

Tanto o algoritmo clássico (Algoritmo 3) quando o novo algoritmo proposto (Algoritmo 5) compartilham parte do método de procura por associações ótimas. É bem sabido que métodos deste tipo não apresentam uma solução única, apenas ótima. Como não poderia ser diferente, a solução (associação final) do algoritmo proposto também não é única.

Entretanto, embora a associação final seja apenas ótima, os resultados de depuração que podem ser obtidos desta são únicos. Isto é consequência direta da unicidade do resultado da decomposição DM. Pois, em tal decomposição chega-se em uma solução única partindo-se de uma associação que não é única, apenas ótima (DULMAGE; MENDELSOHN, 1958 apud ASHCRAFT; LIU, 1998).

Este fato também reforça que é possível apenas determinar o número de estados de um sistema (número de graus de liberdade dinâmicos) e as variáveis que são elegíveis a estados. Mas, não é possível dizer, para o caso geral, quais variáveis exatamente são os estados. Apenas é possível determinar os conjuntos de variáveis que podem ser utilizados como estados do problema.

4.2.6 Sistemas Híbridos Contínuo-Discreto

Em vários momentos neste trabalho, foi citado que a modelagem de processos na engenharia química gera naturalmente sistemas DAE. Neste contexto, muitos são os casos em que as relações algébricas ou forças motrizes de um sistema são descontínuas. Modelos com este tipo de comportamento geram modelos híbridos contínuo-discreto. Basicamente, pode-se diferenciar dois tipos de descontinuidades (MAJER et al., 1995):

- Relacionadas com a operação, como por exemplo, a troca de uma posição de válvula;
- Inerentes ao modelo, como por exemplo, troca de fase, rompimento de uma válvula de segurança e correlações descontínuas.

Os principais códigos para solução numérica de sistemas DAE (Seção 3.4) são capazes de tratar apenas sistemas de equações contínuos. Assim, para a obtenção prática da solução de sistemas híbridos, é necessário construir diferentes sistemas contínuos, um para cada região contínua do modelo. Assim, o tratamento correto de uma descontinuidade requer a localização precisa do ponto onde esta ocorre, a troca do modelo e posterior determinação de uma nova condição consistente para que a integração possa ser continuada (MAO; PETZOLD, 2002). Existem alguns trabalhos que investigam detalhadamente a questão da localização do ponto de descontinuidade (ESPOSITO et al., 2001; SHAMPINE; THOMPSON, 2000; PARK; BARTON, 1996).

Quando há uma descontinuidade em uma ou mais equações de um sistema DAE, esta vai se refletir na descontinuidade no valor de *algumas* variáveis do sistema, mas não em todas. Da mesma forma como na inicialização, em uma reinicialização é necessário conhecer o valor de determinadas variáveis em um número igual ao número de graus de liberdade dinâmicos. Entretanto, como na inicialização, nem todas as combinações de variáveis podem ser utilizadas para o fechamento dos graus de liberdade dinâmicos.

Na determinação da nova condição consistente após uma descontinuidade, algumas variáveis terão seus valores alterados bruscamente enquanto outras permanecerão contínuas. Como ilustração, considere-se novamente o sistema DAE semi-explícito de índice 1 (3.4), aqui reproduzido:

$$\begin{aligned} x' &= f(x, y, t) \\ g(x, y, t) &= 0 \end{aligned}$$

onde, $x \in \Re^n$, y e $g \in \Re^m$ e g_y é não-singular.

Para o sistema (3.4), x são os estados do problema e o número de graus de liberdade dinâmicos é n (ver Seção 3.7). Então, para uma inicialização, é necessária a adição de equações em um número igual a n, por exemplo, $x(t_0) = x_0$. Para o caso de uma descontinuidade em $f(x, y, t)$ ou $g(x, y, t)$, os valores de y e x' poderão sofrer uma alteração brusca, enquanto os x permanecem contínuos. Isto pode ser facilmente visualizado pela forma integral de (3.4):

$$x(t) = x(t^-) + \int_{t^-}^{t} f(x, y, t)dt \tag{4.1}$$

onde, y é determinado implicitamente por g, uma vez que g_y é não-singular.

Ou seja, se ocorre uma descontinuidade em f ou g no tempo t: $x(t) = x(t^-)$, uma vez que a integral em (4.1) se anula quando $t \to t^-$. Então, para sistemas como (3.4), utiliza-se a seguinte equação adicional para contemplar os graus de liberdade dinâmi-

cos em uma reinicialização:

$$x = x^- \tag{4.2}$$

onde, x^- são os valores dos estados imediatamente antes do evento descontínuo. A equação (4.2) também é conhecida como equação de consistência (MAJER et al., 1995).

Para o caso geral (3.1), ou até mesmo quando g_y é singular em (3.4), a equação (4.1) não pode ser utilizada e a descontinuidade também pode afetar alguns, ou até mesmo todos, os x. Este problema se pronuncia quando o sistema apresenta restrições escondidas, as quais reduzem o número de graus de liberdade dinâmicos. Os casos onde não há restrições escondidas são bem estudados na literatura (MAJER et al., 1995; VIEIRA, 1998). Mas, quando há tais restrições adicionais, é necessário determinar para quais variáveis é permitido utilizar equações de consistência. No trabalho de Majer et al. (1995), este problema é identificado e soluções são propostas para sistemas DAE linearmente implícitos. Na verdade, diversos casos particulares dos DAEs linearmente implícitos são estudados, sendo que para cada caso é proposto um tratamento especial. No mesmo trabalho, os autores argumentam que a continuidade dos *estados* nem sempre pode ser considerada.

Nesse trabalho um enfoque diferente é proposto: a continuidade dos *estados* sempre é verdade, a questão é determinar quem *realmente* são os estados do problema. Isto acaba por gerar uma definição paralela de *estado* para sistemas DAE, proposta da seguinte forma:

Definição 4.1 *Para o caso geral de um sistema DAE $F(y, y', t) = 0$, os estados são o subconjunto de elementos de y que admitem continuidade em eventos de descontinuidade em quaisquer funções de F.*

Segundo a definição acima, nos casos de ODEs (3.2) e DAEs semi-explícitos de índice 1 (3.4), os estados coincidem com o vetor x. Esta coincidência pode levar a conclusão equivocada de que, para o caso geral (3.1), as variáveis diferenciais do sistema são os seus estados. Por exemplo, considerando a definição proposta, a argumentação de Majer et al. (1995) de que a continuidade dos estados nem sempre é verdade se torna incoerente. Além disso, para a reinicialização consistente de um sistema DAE geral (3.1) é necessário primeiramente determinar quem são os estados do problema. Uma vez determinados os estados, a continuidade destes na forma (4.2) é assumida.

O algoritmo de Pantelides (Algoritmo 3) não revela quais variáveis podem ser fornecidas como condições iniciais, portanto este também não fornece informações sobre quais são os estados do problema. O Algoritmo 5, proposto neste trabalho para a depuração de sistemas DAE, tem a vantagem de indicar os possíveis conjuntos de condições iniciais viáveis (Subseção 4.2.4).

Como exemplo, considere-se a aplicação do Algoritmo 5 ao sistema (3.15), apresentada na Figura 4.2. Para este caso, a única variável não coberta pela associação final é x_1 e esta é a única variável que permaneceria contínua no caso de um evento descontínuo em qualquer das suas funções. A extensão desta observação em um exemplo simples, gera a seguinte propriedade:

Os estados de um DAE são iguais em número ao número de graus de liberdade dinâmicos e os conjuntos de variáveis que podem ser estados são determinados pelos caminhos alternantes iniciados em variáveis não cobertas pela associação final do Algoritmo 5.

Em casos como o do problema (3.15), os estados são exatamente as variáveis não cobertas pela associação final, mas este não é o caso geral. O exemplo clássico do sistema do pêndulo oscilante de índice 3, largamente citado na literatura (PANTELIDES, 1988a; UNGER et al., 1995; MATTSSON et al., 2000), é um caso mais interessante.

Este sistema pode ser representado como ilustrado na Figura 4.4, e modelado conforme (4.3).

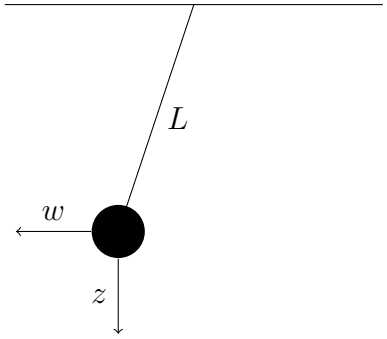


Figura 4.4: Representação do sistema do pêndulo oscilante.

$$\begin{aligned} w &= x' \\ z &= y' \\ Tx &= w' \\ Ty - g &= z' \\ x^2 + y^2 &= L^2 \end{aligned} \tag{4.3}$$

onde, x e y representam as posições horizontal e vertical, w e z são as correspondentes velocidades nestas direções, T e L são a tensão e o comprimento da haste e g a aceleração da gravidade.

A aplicação do método de análise estrutural proposto nesta seção ao sistema (4.3) é ilustrada na Figura 4.5. No grafo final, Figura 4.5(e), apenas dois nós de variável estão livres de associação (y e z). Estes nós representam os graus de liberdade dinâmicos e qualquer variável que possa ser tocada por um caminho alternante que parte destes nós livres pode fazer parte de uma condição inicial consistente. Iniciando-se em y, pode-se chegar, por caminhos alternantes, em $\{x, w', T, z', x'', y''\}$. Partindo-se de z, os nós tocados por caminhos alternantes são $\{w, x', y'\}$. Como estes conjuntos não se interceptam, qualquer combinação de duas variáveis destes conjuntos formará uma condição inicial estruturalmente consistente. Na Subseção 5.3.2 a inicialização numérica envolvendo algumas destas combinações é demonstrada.

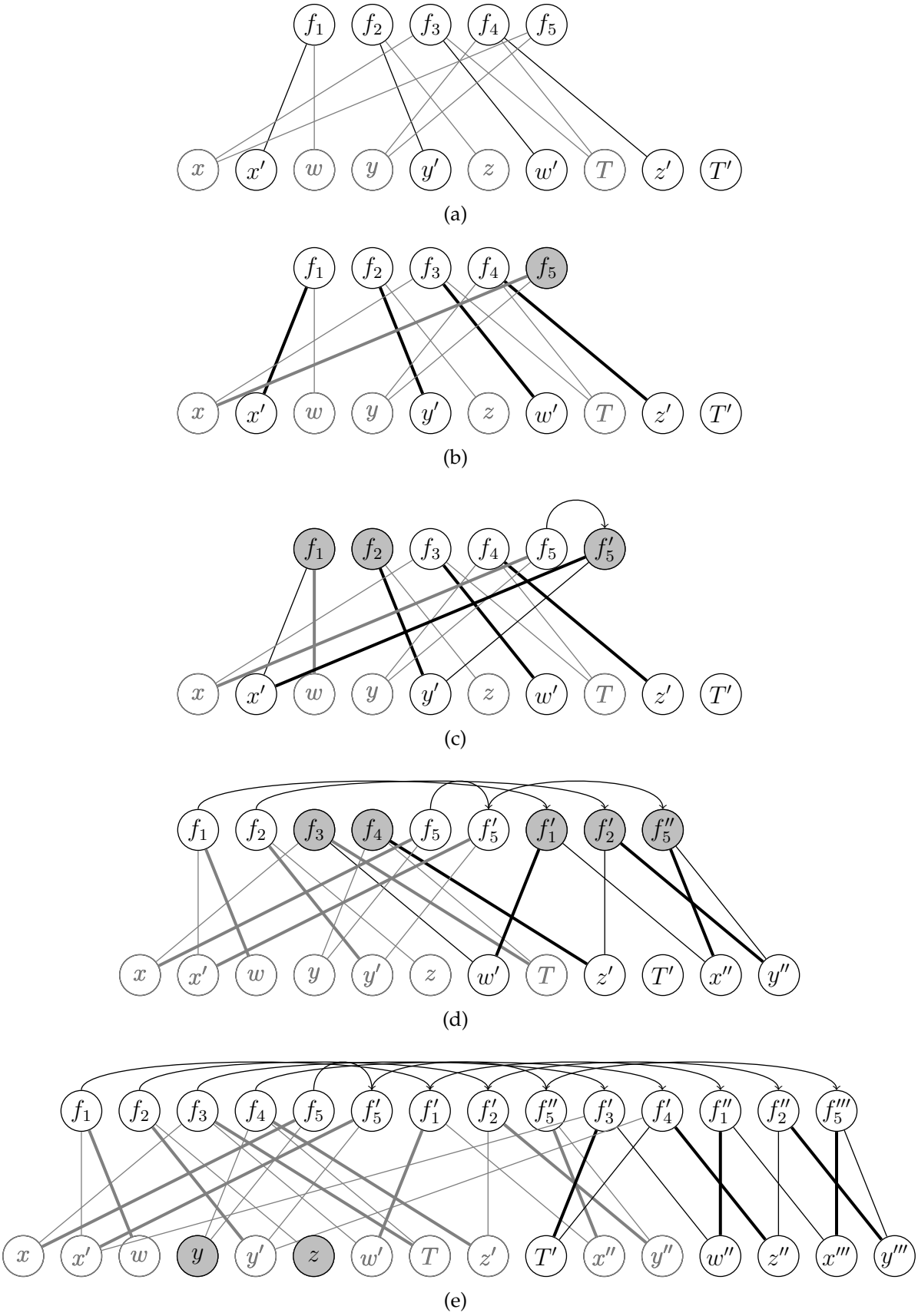


Figura 4.5: Evolução do Algoritmo 5 para o sistema de índice elevado (4.3).

Entretanto, para a determinação das variáveis que podem ser estados, ou seja, que serão contínuas mesmo no evento de qualquer descontinuidade nas funções, é necessário impor restrições adicionais ao conjunto sub-determinado de variáveis. Seguindo-se os desenvolvimentos apresentados nesse capítulo, os conjuntos de variáveis elegíveis a estados para o sistema (4.3) são $\{x, y\}$ e $\{z, w\}$. Pois, de todas as variáveis que podem ser tocadas por caminhos alternantes iniciados nos nós expostos, estas são as únicas variáveis que aparecem na forma diferencial no sistema original. Na Subseção 5.5.4 a simulação deste problema para o caso de um evento de descontinuidade é apresentada. Nesta seção também são apresentadas observações interessantes sobre a continuidade das variáveis elegíveis a estados.

Capítulo 5

Aplicações e Discussões

Neste capítulo é apresentado como os métodos, já bem conhecidos na literatura e revisados no Capítulo 2, podem ser aplicados. As soluções de problemas típicos envolvendo as técnicas propostas no Capítulo 4 também são apresentadas. Além disto, alguns casos particulares são analisados e discutidos.

5.1 Sistemas NLA

No Capítulo 2 os métodos disponíveis na literatura para a inspeção de problemas estruturais em sistemas NLA foram revisados. Entretanto, os exemplos utilizados até o momento foram de pequeno porte. A fim de exemplificar a aplicação destes algoritmos em problemas reais de simulação de processos, considere-se o processo de síntese de amônia apresentado na Figura 5.1 (BIEGLER et al., 1997).

Um modelo estático, consistindo em 134 variáveis, foi construído para o processo da Figura 5.1. Este modelo, na linguagem do simulador EMSO, pode ser encontrado no Apêndice B, Código B.4.

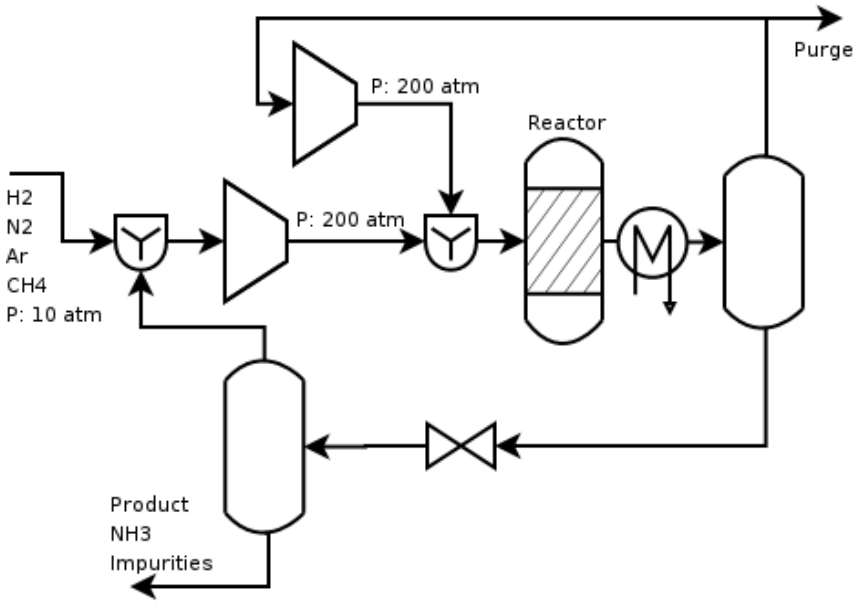


Figura 5.1: Processo de síntese de amônia.

Se especificações são fornecidas corretamente, em um número igual ao número de graus de liberdade do sistema, então o algoritmo de máxima cardinalidade (Algoritmo 2) termina com uma associação perfeita ou completa. Mas, se tivermos uma especificação faltante, por exemplo a vazão de alimentação do processo, então uma associação ótima completa não será possível. Para este caso, a decomposição DM irá revelar o conjunto das variáveis que estão sub-determinadas. A especificação adicional de qualquer uma das variáveis da partição sub-determinada é capaz de gerar novamente um sistema estruturalmente bem posto. Para o caso do modelo de síntese de amônia sem a especificação da vazão de alimentação, a partição sub-determinada envolve 96 variáveis, ou seja, a partição determinada cobre apenas 30% das variáveis.

Infelizmente um comportamento similar é observado para a maioria dos modelos de processos químicos e petroquímicos: no caso de uma singularidade, o número de opções de correção é da ordem do próprio tamanho do sistema. Isto ocorre devido ao alto grau de interação entre as variáveis do modelo. Por exemplo, na Figura 5.2 é apresentado o grafo bipartido para o modelo do processo de síntese de amônia. Como pode ser visto nesta figura, há uma grande complexidade nas relações entre as equações e

variáveis e um alto grau de integração. Note-se que, embora o grafo apresentado na Figura 5.2 seja um grafo bipartido, a partição entre equações e variáveis não é clara. Entretanto, por questões de espaço, não é possível a construção de um grafo bipartido como os apresentados anteriormente.

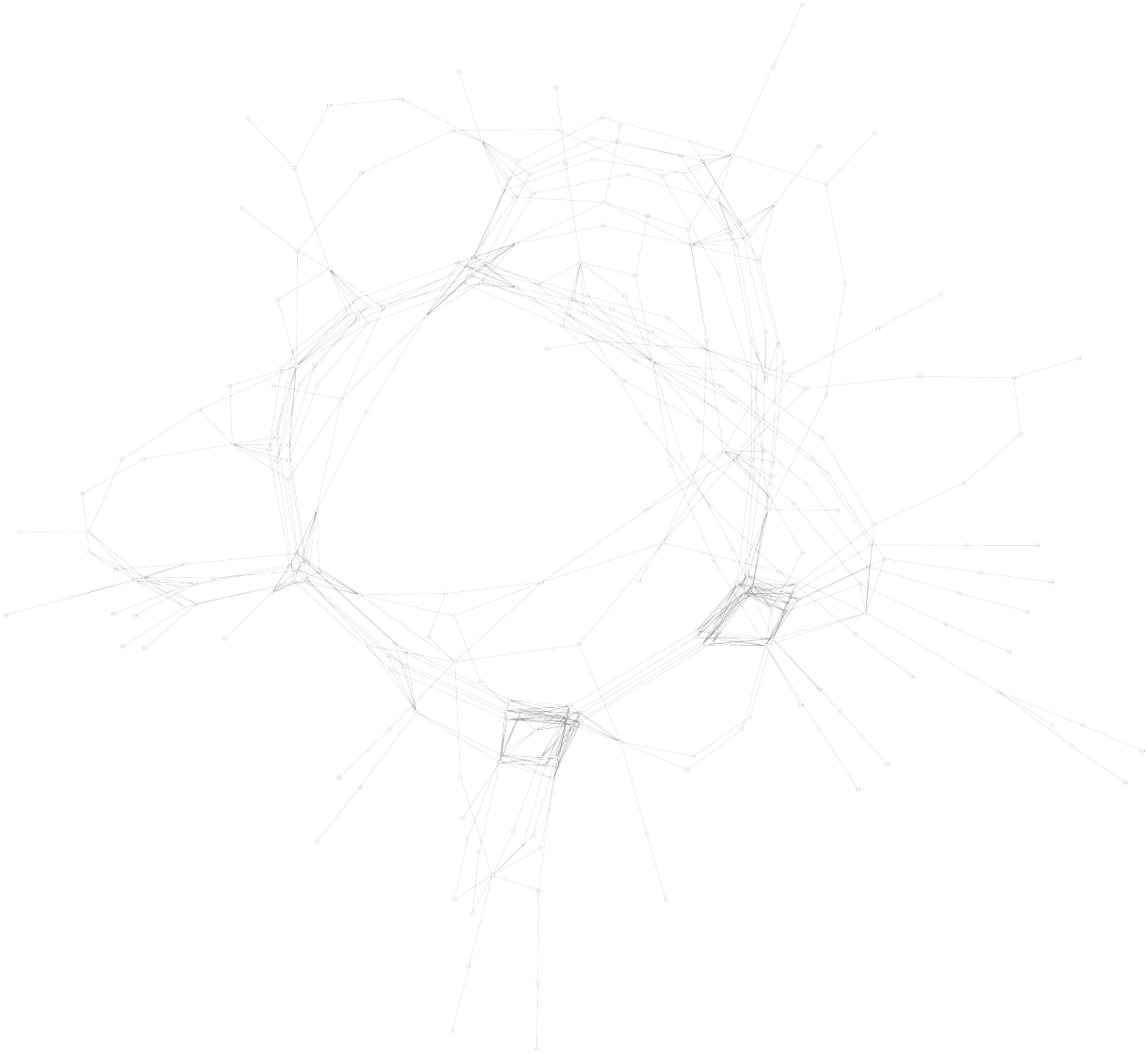


Figura 5.2: Grafo bipartido para o processo de síntese de amônia apresentado na Figura 5.1.

Sem dúvida o grande número de opções de correção de um modelo mal posto é um ponto fraco das técnicas de depuração hoje conhecidas. Para a atenuação desta deficiência, as opções de correção poderiam ser classificadas, utilizando regras heurísticas. Assim, as opções seriam apresentadas ao usuário em uma determinada ordem,

facilitando a correção do problema sem uma sobrecarga de informações. O estudo e implementação destas técnicas não fazem parte do escopo deste trabalho e ficam como sugestão para trabalhos futuros.

5.2 Desempenho do Novo Algoritmo

Para a verificação do desempenho do algoritmo proposto neste trabalho (Algoritmo 5) para a análise estrutural de sistemas DAE, um modelo dinâmico para processos de destilação foi analisado. Este modelo contém balanços de massa e energia para cada prato, além de equações para a predição de propriedades termodinâmicas e correlações para a hidrodinâmica dos pratos. No Apêndice B é encontrada a descrição deste modelo na linguagem do simulador EMSO.

Para o caso de uma separação de isobutano de uma mistura contendo 13 componentes em uma coluna de destilação com 40 pratos, o número de variáveis é da ordem de 4000. O tempo computacional necessário para analisar este modelo com o Algoritmo 5, considerando-se diferentes números de pratos, pode ser visto na Tabela 5.1.

Tabela 5.1: Tempo computacional para a análise com o Algoritmo 5 de um modelo dinâmico de uma coluna de destilação para diferentes números de pratos.

Número de Pratos	Número de Variables - N	Tempo (s)	Tempo $/N^2$ (s $\cdot 10^9$)
20	2157	0,04	9,46
40	3877	0,14	9,58
80	7317	0,52	9,79

Os resultados apresentados na Tabela 5.1 foram obtidos em um Pentium M 1.70 GHz com 2 MB de memória cache, rodando um Ubuntu Linux versão 6.06. Todos os casos analisados são problemas bem postos - este teste não foi utilizado para verificar o poder de depuração do algoritmo, apenas a sua eficiência e complexidade. Como pode ser observado na última coluna da tabela, o desempenho é aproximadamente quadrática,

assim como a maioria dos métodos utilizados para a solução numérica desta categoria de problema.

Outro bom resultado é que o tempo computacional necessário para a análise é aceitável em termos de interação com o usuário. Com as máquinas disponíveis atualmente, o tempo necessário para a análise de sistemas com 10.000 variáveis é da ordem de 1 segundo. Além disso, o algoritmo pode ser aplicado incrementalmente, adicionando novas variáveis e equações à medida que o usuário adiciona os equipamentos para formar o diagrama de processos. Isto pode fazer com que o *software* responda ainda mais rapidamente ao usuário, disponibilizando as informações de depuração dentro do tempo de interação com o usuário.

5.3 Inicialização de sistemas DAE

Diversos aspectos da inicialização de sistemas DAE (3.1) foram abordados nos capítulos 3 e 4. Para a determinação numérica de uma condição inicial, alguns problemas precisam ser resolvidos:

1. Determinação do número de graus de liberdade dinâmicos;

2. Verificação se as condições iniciais fornecidas são coerentes com o modelo;

3. Determinação dos valores de todas as variáveis (y e y') no tempo inicial.

Utilizando-se o algoritmo de Pantelides (Seção 3.6) o item 1 é atualmente bem resolvido. Muitos softwares comerciais executam tal algoritmo antes da solução numérica, porém alguns abortam a solução se um problema com índice maior que 1 é encontrado.

O trabalho de Unger et al. (1995) trata parcialmente do item 2, onde o autor utiliza rotinas de triangularização da álgebra linear para verificar se uma dada condição inicial é viável. Entretanto se esta não for viável, as informações geradas não são de grande valia para uma depuração do problema. As demais implementações existentes simplesmente fazem a tentativa da solução numérica. Mais uma vez, se esta falhar não há um mecanismo para determinar se a fonte da falha é uma incoerência das condições iniciais ou um problema puramente numérico. Na Subseção 4.2.4, como uma contribuição importante deste trabalho, uma solução de depuração estrutural para este problema foi apresentada.

Quanto à terceira etapa de uma inicialização, a determinação dos valores numéricos de todas as variáveis (y e y') no tempo inicial, iniciou-se uma discussão na Subseção 3.6.3 e esta seguiu na Subseção 4.2.4.

Ainda sobre a inicialização de sistemas DAE, é interessante considerar novamente o problema (3.15):

$$\begin{aligned} x_1' - x_2' &= a(t) \\ x_2 &= b(t) \end{aligned}$$

Este é um problema de índice 1, contém apenas 1 grau de liberdade dinâmico e, como apresentado na Subseção 4.2.4, x_1 é a única variável que pode ser fornecida como condição inicial. A solução analítica para este problema é:

$$\begin{aligned} x_1 &= x_1(0) + \int a(t) + b(t) \\ x_2 &= b(t) \end{aligned}$$

Embora seja um problema simples, ele não é tratado atualmente pelos simuladores de processo comerciais. Na última versão disponível do gPROMS (OH; PANTELIDES, 1996), obtém-se o seguinte resultado:

```
Set up of simulation
All 2 variables will be monitored during this simulation!
The number of initial conditions (1) does not match the
```

```
    number of states (2)
Building mathematical problem description took 0 seconds.
```

Como pode ser observado no texto de saída acima, o sistema considera que há 2 estados no modelo e espera, então, 2 condições iniciais. Fica claro que há uma falha, pois o sistema requer apenas uma condição inicial para que seja completamente determinado, uma vez que $x_2(0) = b(0)$. Se são fornecidas 2 condições iniciais, como requerido pelo *software*, condição esta que não é correta, o sistema falha novamente:

```
 Performing initialization calculation at time: 0
  Variables
       Known            : 0
       Unknown          : 2
          Differential  : 2
          Algebraic     : 0
   Model equations      : 2
   Initial conditions   : 2
Checking index of differential-algebraic equations (DAEs)...
   ERROR: Your problem is a DAE system of index greater than 1.
          Your differential variables ("states") are not independent
```

Observa-se também na mensagem de erro acima que o *software* reporta um índice maior do que 1 para este problema, o que não é correto. Na verdade, este é um caso onde o índice singular (UNGER et al., 1995) é maior que o índice diferencial. Sendo então muito provável que este *software* tenha utilizado o conceito de índice singular em sua análise estrutural, mostrando não ser uma boa alternativa.

Resultado semelhante é obtido na tentativa da solução deste problema no simulador ASPEN Dynamics. Como pode ser visto na Figura 5.3, o *software* identifica equivocadamente o sistema como sendo de índice elevado, provavelmente pelo mesmo motivo do *software* anterior.

Utilizando-se o simulador EMSO, o qual implementa a técnica desenvolvida na Subseção 4.2.4, o problema pode ser resolvido sem dificuldades. O texto de saída,

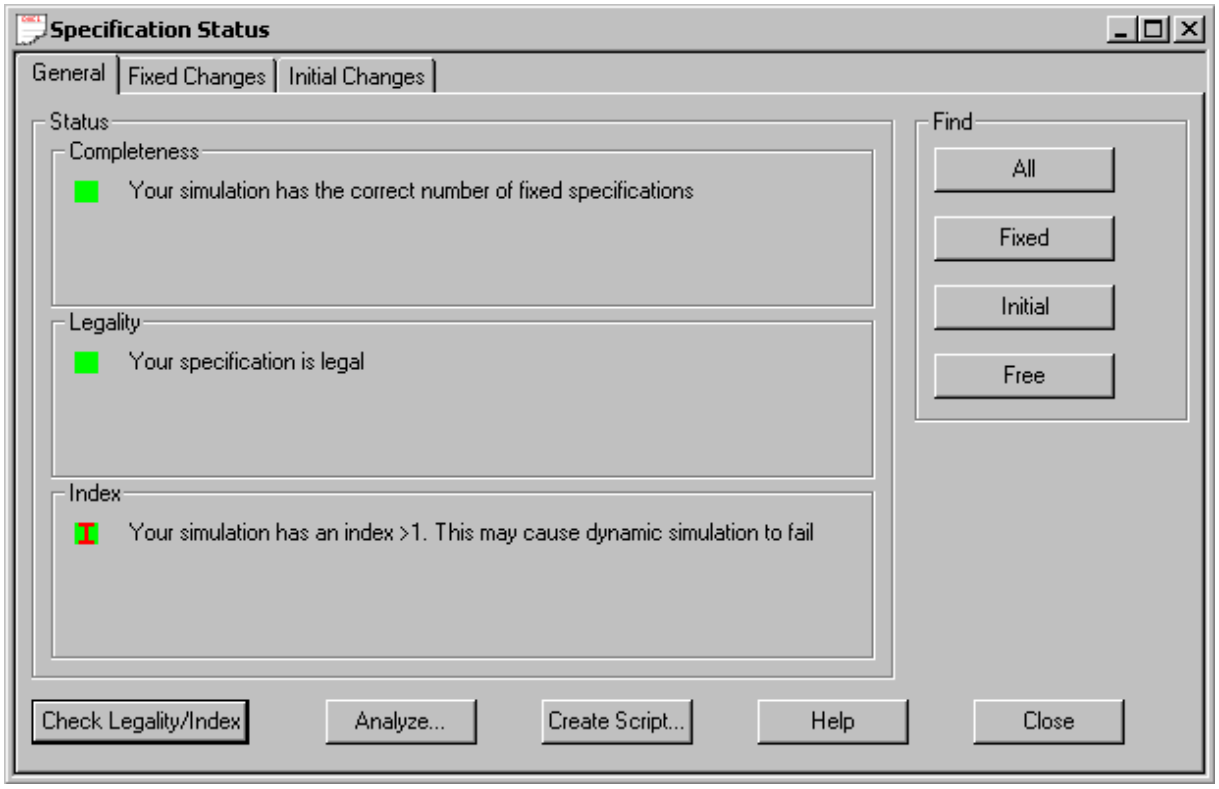


Figura 5.3: Erro apresentado pelo ASPEN Dynamics na solução do problema (3.15).

produzido pela análise estrutural do sistema, segue:

```
Number of variables:  2
Number of equations:  2
Number of specifications:  0
Degrees of freedom: 0
Structural differential index: 1
Extra Equations:    1
Extra Variables:    0
Dynamic degrees of freedom (states):  1
Number of initial Conditions: 1
```

Como pode ser observado, o EMSO identifica corretamente que há apenas 1 estado dentre as 2 variáveis do sistema e consegue proceder a solução numérica sem problemas.

Na Subseção 3.6.3 foi apresentada uma metodologia para determinação de uma condição inicial consistente através da resolução de um problema não-linear. Este

método utiliza-se das informações geradas pelo algoritmo de Pantelides para construir um problema não-linear composto pelo modelo original, derivadas de algumas de suas equações e um conjunto de condições iniciais. Entretanto, como discutido na Subseção 3.6.3, tal método não é capaz de determinar o valor de todas as variáveis necessárias para partir uma integração com códigos clássicos, como por exemplo o DASSL. Assim, as variáveis não determinadas diretamente precisam ser obtidas com algum método aproximado, usualmente um método tipo Euler de primeira ordem.

Na Seção 4.2 foi proposto um novo método para análise estrutural que é capaz de revelar todas as restrições relevantes. Assim, utilizando-se procedimento semelhante ao apresentado na Subseção 3.6.3, porém baseado no novo algoritmo, o valor de todas as variáveis é determinado diretamente. Também deve ser notado que esta metodologia pode ser utilizada para determinar uma condição inicial consistente para problemas de qualquer índice. Além disso, o problema pode ser não-linear nas derivadas, caso onde as técnicas baseadas em fórmulas de primeira ordem usualmente falham (PETZOLD, 1983).

A aplicação desta metodologia, comparando-a com as técnicas usuais quando possível, para sistemas de índice baixo e elevado seguem nas próximas seções.

5.3.1 Inicialização de sistemas de baixo índice

A maior parte das falhas na solução numérica de sistemas algébrico-diferenciais ocorre durante a sua inicialização (VIEIRA, 1998; WU; WHITE, 2001). Com intuito de comparação entre a metodologia proposta para a inicialização de sistemas DAE genéricos, considere o sistema galvanostático proposto por Wu e White (2001):

$$\begin{aligned} \frac{\rho V}{W} y_1' &= \frac{j_1(y_1, y_2)}{F} \\ j_1(y_1, y_2) + j_2(y_2) &= i_{app} \end{aligned} \tag{5.1}$$

onde, y_1 e y_2 são as variáveis do problema e as demais são constantes ou funções destas variáveis, omitidas aqui por simplicidade.

O sistema (5.1) representa a modelagem de um processo galvanostático de um filme fino de hidróxido de níquel, onde y_1 é a concentração de NiOOH e y_2 a diferença de potencial na interface sólido-líquido. Uma análise estrutural deste sistema (algoritmos 3 ou 5) revela que o sistema é de índice 1 e tem apenas 1 grau de liberdade dinâmico. Assim, arbitrado o valor para uma das variáveis em um ponto t_0 as demais (incluindo y_2' que não aparece explicitamente no sistema) devem ser determinadas.

No trabalho de Wu e White (2001) o sistema (5.1) foi utilizado para uma comparação de robustez numérica. Na Figura 5.4 é apresentada a evolução do Algoritmo 5 quando aplicado ao sistema (5.1)

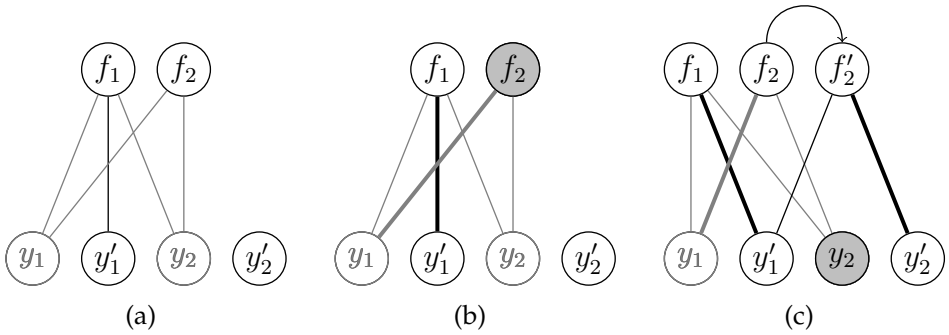


Figura 5.4: Evolução do Algoritmo 5 para o sistema galvanostático (5.1).

Como pode ser observado na Figura 5.4(c), o sistema tem 1 grau de liberdade dinâmico (variável y_2 não coberta pela associação final). Assim sendo, para a obtenção de uma condição inicial consistente, 1 variável pode ser arbitrada e as demais devem ser determinadas. Entretanto, como todo método numérico, as variáveis incógnitas necessitam de uma estimativa inicial para a solução.

Então, a comparação de robustez apresentada por Wu e White (2001) se deu através de uma análise da faixa de estimativa inicial para a qual se obtém convergência de uma

variável quando a outra é fornecida como condição inicial. Os resultados da comparação são apresentados na Tabela 5.2. Os códigos utilizados na comparação baseiam-se em diferentes métodos de inicialização: DASSL utiliza a fórmula de diferenças finitas retroativa (BRENAN et al., 1989); RADAU5 utiliza o método de Runge-Kutta implícito (HAIRER; WANNER, 1996); LIMEX utiliza um método de extrapolação baseado em Euler implícito (DEUFLHARD et al., 1987); DAEIS utiliza um método tipo Newton (WU; WHITE, 2001); e o método proposto na Seção 5.3.

Tabela 5.2: Faixas de convergência de diferentes códigos quando da inicialização de (5.1).

Código	**Faixa de convergência de** $y_2(t_0)$**, dado** $y_1(t_0)$	**Faixa de convergência de** $y_1(t_0)$**, dado** $y_2(t_0)$
DASSL*	$0,321 \leq y_2(t_0) \leq 0,370$	$0,071 \leq y_1(t_0) \leq 0,352$
LIMEX*	$0,318 \leq y_2(t_0) \leq 0,377$	$0,056 \leq y_1(t_0) \leq 0,418$
RADAU5*	$0,348 \leq y_2(t_0) \leq 0,352$	$0,143 \leq y_1(t_0) \leq 0,190$
DAEIS*	$-0,974 \leq y_2(t_0) \leq 1,663$	$0,0 \leq y_1(t_0) \leq 1,0$
Proposto	$-2,70 \leq y_2(t_0) \leq 2,66$	$-\infty \leq y_1(t_0) \leq \infty$
Valor consistente	$y_2(t_0) = 0,35024$	$y_1(t_0) = 0,15513$

*Fonte Wu e White (2001)

Como pode ser observado na Tabela 5.2 o método proposto, o qual utiliza o sistema de índice zero para a inicialização mostrou-se mais robusto. A faixa de convergência foi muito superior à obtida com pacotes computacionais largamente utilizados.

Entretanto, para obter os resultados com o método proposto, é necessária a utilização de derivação simbólica. Deste fato surgem duas questões: o esforço computacional na execução de derivações simbólicas e a derivação de rotinas que não são descritas na forma de equações dentro do pacote de simulação (usualmente o caso dos cálculos termodinâmicos).

Quanto ao primeiro ponto, a derivação simbólica com relação à variável independente está implementada de forma muito eficiente no simulador de processos EMSO (SOARES; SECCHI, 2003), permitindo a utilização desta técnica até mesmo para pro-

blemas de grande escala. Para tratar dos casos onde não é possível executar a diferenciação simbólica, existem alternativas, algumas delas são apresentada na Subseção 5.3.3.

5.3.2 Inicialização de sistemas de índice elevado

A maioria dos códigos utilizados para a inicialização de sistemas DAE não consegue tratar sistemas com índice superior a um. Entretanto, falhas e dificuldades de convergência, como as apresentadas na Tabela 5.2, podem surgir até mesmo para sistemas de índice 1. A metodologia aplicada na Subseção 5.3.1, a qual executa uma redução de índice até zero para posterior inicialização pela solução de um sistema não-linear, independe do índice do sistema original.

Na Subseção 4.2.6, o exemplo do pêndulo oscilante de índice 3 foi introduzido. O sistema pode ser ilustrado pela Figura 4.4 e modelado conforme o sistema (4.3), aqui reproduzido por conveniência:

$$
\begin{aligned}
w &= x' \\
z &= y' \\
Tx &= w' \\
Ty - g &= z' \\
x^2 + y^2 &= L^2
\end{aligned}
$$

Como discutido na Subseção 4.2.6, o sistema tem 2 graus de liberdade dinâmicos. Estes graus de liberdade podem ser contemplados por uma variável de $\{w, x', y'\}$ e outra de $\{x, w', T, z', x'', y''\}$. Também foi discutido que, dentre estas variáveis, os estados do sistema podem ser compostos por uma variável de $\{y, x\}$ e outra de $\{z, w\}$. Para fins de teste, a inicialização numérica do sistema utilizando todas as combinações dos estados do sistema foi executada. A metodologia é a mesma apresentada na Subseção 4.2.4, que neste caso significa utilizar o sistema de equações representado na Figura 4.5(e) concatenado com duas equações adicionais. Estas equações adicionais são valores arbitrados para duas variáveis. A Tabela 5.3 apresenta o resultado destes

testes.

Tabela 5.3: Determinação de condições iniciais consistentes para o sistema de índice elevado (4.3) com os valores em negrito arbitrados.

Caso	x	y	w	z	T
1	**0,5**	0,87	**0,0**	0,0	8,49
2	**0,5**	0,87	1,74	**-1,0**	4,49
3	0,0	**1,0**	**2,0**	0.0	5,80
4	0,87	**0,5**	0,58	**-1,0**	3,56

Como pode ser observado na Tabela 5.3, obteve-se uma solução consistente para todas as combinações envolvendo as variáveis elegíveis a estados (as variáveis arbitradas como condições iniciais aparecem em negrito na tabela). Embora neste estudo tenham sido utilizados apenas valores arbitrados para as variáveis, cada condição inicial pode ser uma expressão, desde que envolvam um conjunto viável de variáveis. Um conjunto de condições iniciais estruturalmente viável é qualquer conjunto capaz de determinar a parte sub-determinada do sistema. Uma implementação que suporta expressões como condições iniciais está disponível no simulador dinâmico EMSO (SOARES; SECCHI, 2003).

5.3.3 Inicialização sem derivação simbólica

A técnica utilizada para inicialização de sistemas de baixo índice (Subseção 5.3.1) e elevado (Subseção 5.3.2) utilizam derivação simbólica das equações do modelo. Mais detalhadamente, o seguinte sistema de equações é resolvido para a determinação da condição inicial:

$$\begin{aligned} &F(t_0, y_0, y_0') = 0 \\ &I(t_0, y_0, y_0') = 0 \\ &F^n(t_0, y_0, y_0') = 0 \end{aligned} \tag{5.2}$$

onde, F é o conjunto original de equações do modelo, I são as condições arbitradas (em número igual aos graus de liberdade dinâmicos) e F^n são derivadas de ordem até n de alguns dos elementos de F (determinadas pelo Algoritmo 5).

Porém, nem sempre tem-se acesso à forma explícita das equações para que as derivações simbólicas presentes em F^n possam ser executadas. Estes são os casos quando são utilizadas rotinas de terceiros que encontram-se *compiladas*, o que é típico em cálculos de predição de propriedades termo-físicas.

Uma forma de contornar este problema é uma adaptação do método estudado por Costa Jr. (2003), determinando F' por derivadas parciais aplicando a regra da cadeia:

$$f' = f_y y' + f_{y'} y'' + f_t \tag{5.3}$$

5.3.3.1 Inicialização Numérica

Para problemas típicos de baixo índice, as equações a derivar são equações algébricas (não envolvem y') e também não envolvem explicitamente a variável independente t. Para estes casos, a equação (5.3) pode ser reescrita simplesmente como:

$$f' = f_y y' \tag{5.4}$$

É importante notar que a utilização da equação (5.4) não causa complicação adicional alguma na implementação, uma vez que a derivada f_y já precisa ser calculada para a obtenção da solução numérica por métodos usuais. Porém, a solução do sistema (5.2) por um método tipo Newton requer além do cálculo de F', alguma aproximação da Jacobiana:

$$J = \begin{bmatrix} F_y & F_{y'} \\ I_y & I_{y'} \\ F'_y & F'_{y'} \end{bmatrix} \tag{5.5}$$

Ou seja, para a solução do problema não-linear resultante, também é necessário conhecer as derivadas parciais f'_y e $f'_{y'}$. Utilizando-se a equação (5.4), estas derivadas

são respectivamente:

$$\begin{aligned} f'_y &= f_{yy}y' \\ f'_{y'} &= f_{yy'}y' + f_y \end{aligned} \tag{5.6}$$

Lembrando da condição previamente considerada de que as equações a derivar não envolvem y' pode-se concluir que o termo $f_{yy'}$, na equação (5.6), é nulo. Portanto, a equação (5.6) pode ser reescrita como:

$$\begin{aligned} f'_y &= f_{yy}y' \\ f'_{y'} &= f_y \end{aligned} \tag{5.7}$$

Infelizmente a utilização da equação (5.7) para a determinação da Jacobiana não é tão direta, uma vez que a derivada segunda f_{yy} não está usualmente disponível nas ferramentas de simulação. Na maioria dos casos, quando não se tem acesso à forma da equação, f_y já é determinado por perturbações. Desta forma, a determinação de f_{yy} por um outro nível de perturbações gera resultados muito imprecisos e que certamente vão impactar na solução. Poderia ser argumentado que a derivada segunda f_{yy} é facilmente obtida com códigos de derivação automática, porém deve ser lembrado que esta metodologia está sendo desenvolvida justamente para os casos em que não há acesso ao formato ou código da função, apenas seu resíduo.

Apesar dos inconvenientes destacados acima, como uma metodologia alternativa à derivação simbólica, pode-se utilizar a equação (5.4) e as equações (5.7) para o cálculo do Jacobiano das equações derivadas. Esta técnica foi aplicada ao sistema (5.1), apresentado na Subseção 5.3.1 para a comparação entre diferentes métodos de inicialização. Os resultados para esta técnica, a qual calcula as derivadas numericamente por perturbação, estão presentes na Tabela 5.4, indicados como *Proposto (Num.)*. Nesta tabela os resultados já apresentados na Tabela 5.2 são reproduzidos para uma melhor comparação.

Como seria de se esperar, os resultados da Tabela 5.4 demonstram que a utiliza-

Tabela 5.4: Faixas de convergência de diferentes códigos quando da inicialização de (5.1), incluindo as técnicas por perturbação numérica.

Código	**Faixa de convergência de** $y_2(t_0)$**, dado** $y_1(t_0)$	**Faixa de convergência de** $y_1(t_0)$**, dado** $y_2(t_0)$
DASSL*	$0,321 \leq y_2(t_0) \leq 0,370$	$0,071 \leq y_1(t_0) \leq 0,352$
LIMEX*	$0,318 \leq y_2(t_0) \leq 0,377$	$0,056 \leq y_1(t_0) \leq 0,418$
RADAU5*	$0,348 \leq y_2(t_0) \leq 0,352$	$0,143 \leq y_1(t_0) \leq 0,190$
DAEIS*	$-0,974 \leq y_2(t_0) \leq 1,663$	$0,0 \leq y_1(t_0) \leq 1,0$
Proposto	$-2,70 \leq y_2(t_0) \leq 2,66$	$-\infty \leq y_1(t_0) \leq \infty$
Proposto (Num.)	$-0,64 \leq y_2(t_0) \leq 1,22$	$-1190 \leq y_1(t_0) \leq 1190$
Proposto (Num., $f'_y = 0$**)**	$-4,74 \leq y_2(t_0) \geq 5,37$	$-\infty \leq y_1(t_0) \leq \infty$
Valor consistente	$y_2(t_0) = 0,35024$	$y_1(t_0) = 0,15513$

*Fonte Wu e White (2001)

ção de perturbações em substituição das derivações simbólicas geram uma perda de qualidade. Por outro lado, quando são considerados os códigos clássicos, o método proposto baseado em perturbações representa um ganho significativo de robustez.

Note-se que a metodologia apresentada nesta seção está limitada por suas considerações restritivas. Entretanto, como apresentado em Costa Jr. (2003), é possível determinar derivadas de mais alta ordem para a equação (5.3). Isto permite a inicialização de sistemas de qualquer índice, mesmo quando derivadas simbólicas não estão disponíveis.

5.3.3.2 Inicialização Numérica Modificada

A necessidade do cálculo adicional de derivadas de segunda ordem em (5.7) e a visível perda de qualidade quando não é possível executar derivações simbólicas (Tabela 5.4) são desvantagens notórias. Felizmente, para o caso de problemas típicos de baixo índice, um artifício que remove estas duas desvantagens de uma só vez foi encontrado.

Como já citado anteriormente, para sistemas típicos de baixo índice, as equações a serem derivadas (determinadas pelo Algoritmo 5) são equações algébricas. Também deve ser notado que, para estes casos, as equações derivadas terminam associadas com variáveis do conjunto y' (por exemplo Figura 5.4). Isto significa, como era de se esperar, que o objetivo da adição das equações derivadas é permitir que variáveis y' que não aparecem explicitamente no sistema sejam determinadas.

Refletindo sobre esta discussão foi proposta uma outra aproximação para o Jacobiano (5.5):

$$J = \begin{bmatrix} F_y & F_{y'} \\ I_y & I_{y'} \\ 0 & F'_{y'} \end{bmatrix} \tag{5.8}$$

Nesta aproximação, o termo F'_y presente em (5.5) é considerado nulo, com a justificativa de que F' é introduzido para determinar elementos de y' e não de y. Esta consideração traz consigo uma grande simplificação na implementação uma vez que, segundo (5.7), os elementos de F'_y são iguais aos elementos que lhe deram origem presentes em F_y.

O resultado para a utilização desta metodologia com o problema teste (5.1) também está presente na Tabela 5.4, indicado por *Num.*, $f'_y = 0$. Surpreendentemente, os resultados foram ainda melhores do que os quando a derivação simbólica foi utilizada. Entretanto, esta metodologia não pode ser aplicada para sistemas de índice elevado.

O método como descrito nesta seção está implementado no simulador dinâmico EMSO (SOARES; SECCHI, 2003) e tem sido utilizado com sucesso na inicialização de problemas complexos, como por exemplo colunas de destilação onde as propriedades são calculadas com o pacote termodinâmico comercial VRTherm (VRTECH, 2006).

5.4 Reinicialização de DAEs

Na Subseção 4.2.6 o tratamento de sistemas DAE híbridos foi discutida e a questão da reinicialização de sistemas DAE abordada. Basicamente, para os casos mais simples os estados são as variáveis diferenciais e para o caso geral um sub-conjunto destas. Também foi apresentado na Subseção 4.2.6 como revelar os estados de um sistema DAE utilizando-se o resultado do Algoritmo 5.

Uma vez determinados os estados do problema, são utilizadas equações de consistência (4.2) em substituição às condições iniciais. Assim, a reinicialização pode utilizar-se de todos os desenvolvimentos apresentados anteriormente.

Para fins de demonstração, considere-se o seguinte sistema de equações (MAJER et al., 1995):

$$\begin{aligned} x' &= y_1 - 1 \\ 0 &= x - y_2 + y_1^2 + u \\ 0 &= 2y_1^{\frac{1}{3}} + \sqrt{y_2} - 4 \\ u &= \begin{cases} 0, t < 50 \\ u_1, t \geq 50 \end{cases} \end{aligned} \tag{5.9}$$

A aplicação do Algoritmo 5 ao sistema (5.9) é ilustrada na Figura 5.5.

Como pode ser visto na Figura 5.5(c), o sistema (5.9) apresenta o número de estados ou graus de liberdade dinâmicos igual a 1. O nó de variável livre na associação final é y_2, porém esta não pode ser um estado do modelo, uma vez que não é uma variável diferencial no sistema (5.9). Os caminhos alternantes que partem de y_2 podem tocar ainda y_1 e x. Dentre as possíveis variáveis do conjunto $\{x, y_1, y_2\}$, x é o estado, uma vez que esta é a única variável diferencial em (5.9).

A solução de (5.9) com $u_1 = 2$ e uma condição inicial de $x(t_0) = 2$, utilizando-se

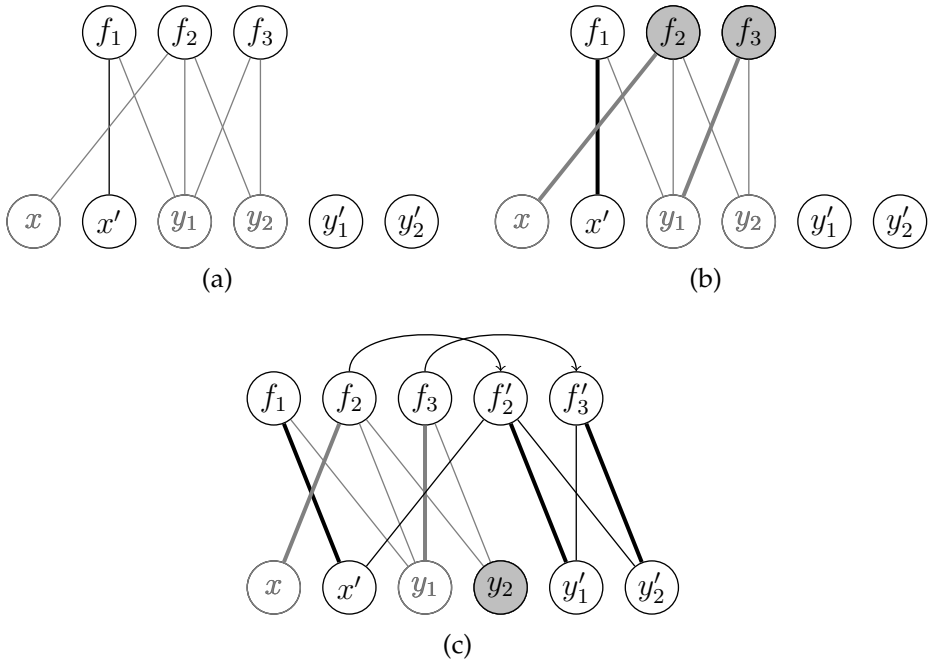


Figura 5.5: Aplicação do Algoritmo 5 ao sistema de equações (5.9).

o simulador EMSO, que dispõe de uma implementação como descrita neste trabalho, pode ser vista na Figura 5.6. Como pode ser observado, com exceção de x, todas as variáveis apresentam descontinuidades em $t = 50$, inclusive x'.

Embora a solução ilustrada na Figura 5.6 seja fácil de obter com a metodologia proposta, este mesmo problema pode ser utilizado para apresentar a fragilidade dos métodos tipo Newton. Para isto, considere-se a situação onde $u_1 = 5$ em (5.9). Neste caso, o método presente no EMSO falha no ponto onde ocorre o evento discreto, exatamente como descrito em Majer et al. (1995). Para estes casos fica clara a necessidade da utilização de métodos mais robustos na solução do problema não-linear ou a substituição do evento discreto por funções de regularização como descrito, por exemplo, em Vieira (1998).

Entretanto, a experiência tem mostrado que problemas como este não são comuns em aplicações da engenharia química e que métodos tipo Newton se aplicam muito

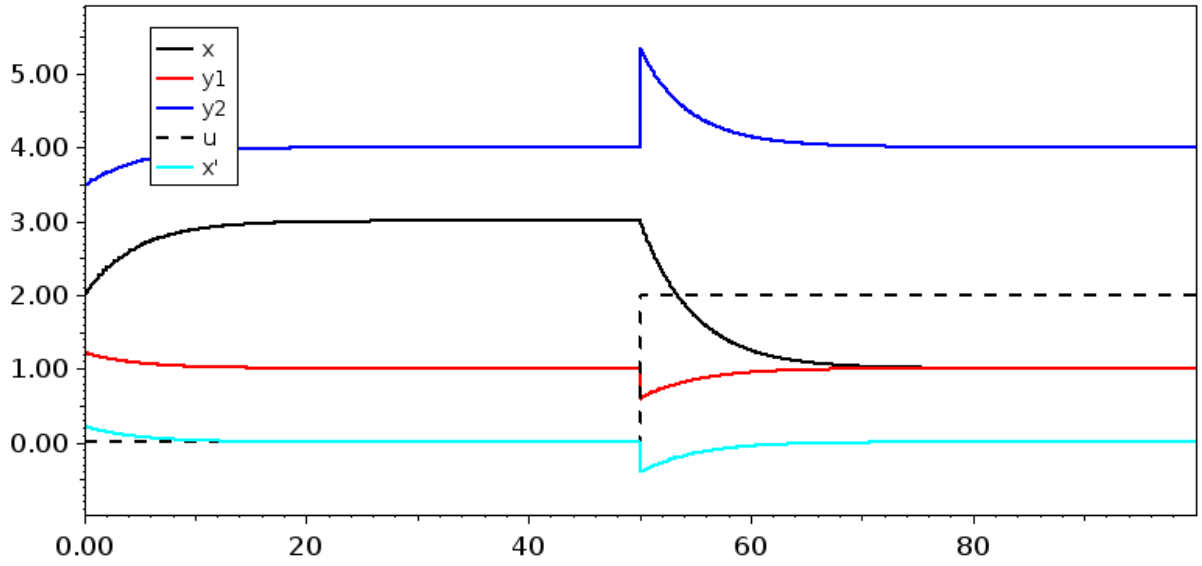


Figura 5.6: Solução do sistema descontínuo (5.9).

bem para a reinicialização. Isto se deve principalmente ao fato de que, para a maioria dos problemas, a solução antes do evento é uma estimativa muito boa para a solução após o evento discreto.

A mesma metodologia pode perfeitamente ser utilizada para a solução de problemas de grande escala. Por exemplo, na Figura 5.7 é apresentado o perfil de vazão de líquido nos pratos de uma coluna de destilação quando executado um procedimento de partida. A simulação de tal procedimento envolve diversos eventos discretos, tratados com o método proposto sem grandes dificuldades. O modelo utilizado para obter os resultados da Figura 5.7 é apresentado no Apêndice B e mais detalhes sobre o processo e o procedimento de partida utilizados para obter os perfis apresentados podem ser encontrados em Staudt et al. (2006).

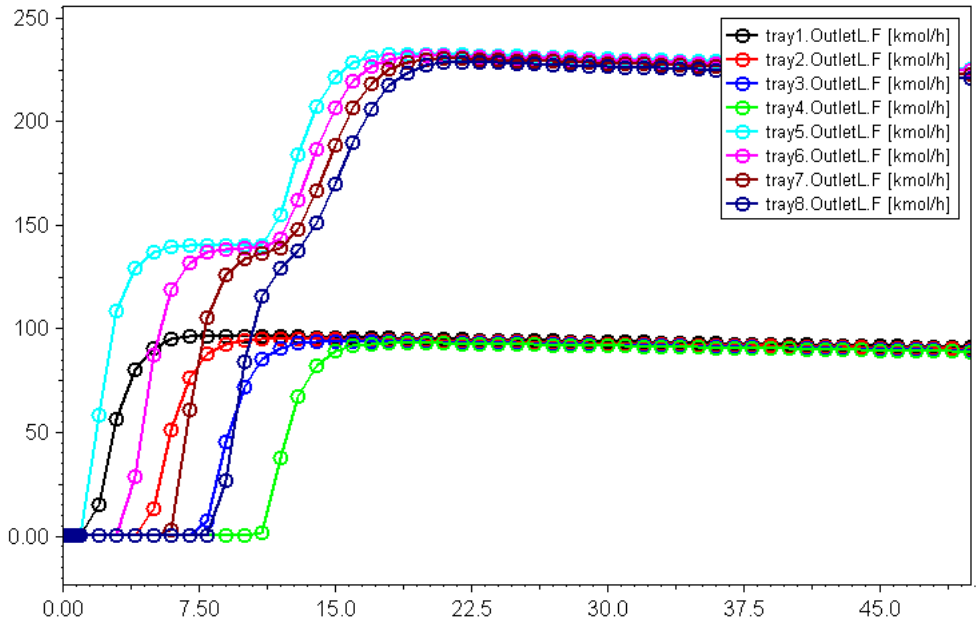


Figura 5.7: Simulação de uma partida de uma coluna de destilação para visualização de eventos discretos.

5.5 Solução de sistemas DAE de índice elevado

A solução de sistemas DAE (3.1) de baixo índice não acarreta em dificuldades adicionais quando comparada à solução de problemas ODE (3.2), mas a inicialização destes sistemas ainda pode ser problemática, como tratado na Seção 5.3. O objeto desta seção é a evolução na variável independente e o termo *solução* será utilizado como seu sinônimo.

Embora já tenham se passado 15 anos, nada mudou desde a afirmativa de Brenan et al. (1989) de que não existem métodos capazes de tratar da solução de qualquer tipo de sistema DAE. Para a solução de problemas de índice elevado restam como opções a redução de índice ou a utilização de códigos especialmente projetados para determinadas categorias de problemas. Estas duas abordagens são apresentadas nas seções que seguem.

5.5.1 Solução com Redução de Índice

A técnica de redução de índice por diferenciação é bem conhecida, assim como os problemas que esta acarreta (HAIRER; WANNER, 1996). Manualmente pode-se diferenciar as restrições algébricas e fazer algumas substituições até que se obtenha uma formulação de índice 1 ou 0 e então resolver o problema resultante com um código para sistemas DAE de baixo índice. Uma forma automática deste procedimento, utilizando o algoritmo de Pantelides, é tratada em vários trabalhos (MATTSSON et al., 2000; POULSEN, 2001; Costa Jr., 2003).

Como pode ser observado, a redução de índice por diferenciação insere novas equações diferenciais em substituição de equações algébricas. Esta substituição gera um sistema que tem muito mais soluções que o sistema original. Isto não acarretaria nenhum problema para uma solução analítica, mas para uma solução numérica o problema de *drift-off* aparece.

Como exemplo, considere-se o problema do pêndulo (4.3) de índice 3. A redução de índice para este problema pode ser visualizada na Figura 4.5. A integração do sistema resultante com o DASSL é possível, porém a solução não é correta. Na Figura 5.8 a distorção da solução ($x^2 + y^2 = L^2$) quando da integração do sistema com redução de índice por derivação é apresentada.

Como pode ser visto, a solução apresentada não pode ser considerada uma solução do problema original, onde $x^2+y^2-L^2 = 0$. Note-se que para a solução deste problema utilizou-se uma tolerância de 10^{-6} e os erros foram da ordem de 1. Certamente erros desta magnitude são inadmissíveis, entretanto a solução por redução de índice tem a potencial vantagem de não impor limitações à estrutura ou índice do problema.

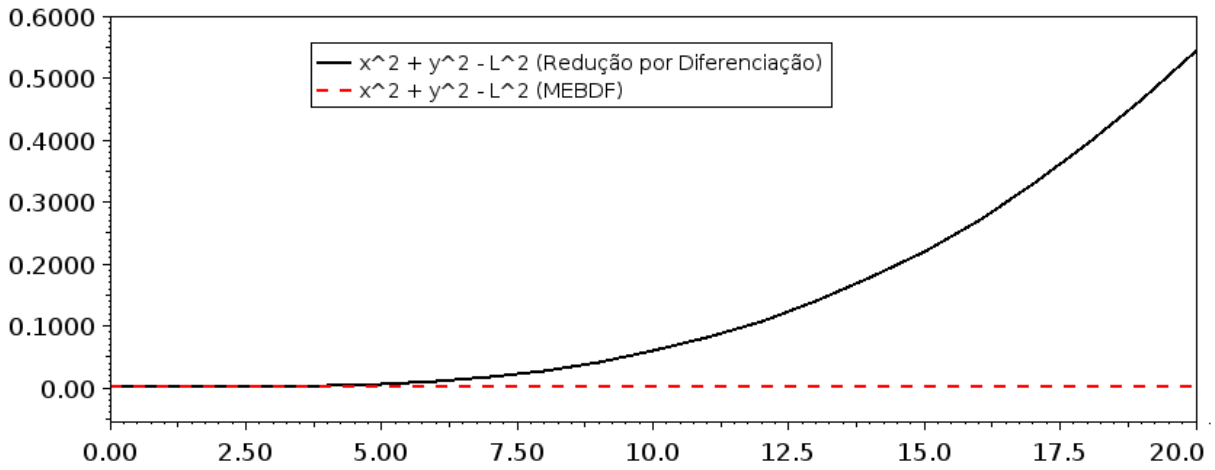


Figura 5.8: Distorção na solução do sistema de índice 3 (4.3) com redução de índice por diferenciação e utilizando o método MEBDF.

5.5.2 Códigos para sistemas de índice elevado

Na Subseção 3.4.1 foi ilustrada a dificuldade dos métodos de passos múltiplos no tratamento de problemas de índice elevado. Os métodos IRK (Subseção 3.4.2) também apresentam problemas quando o índice é maior que 1, pois nestes casos pode haver redução de ordem tornando necessária a utilização de métodos de alta ordem (HAIRER; WANNER, 1996).

De uma forma limitada, é possível utilizar alguns artifícios para tratar dos problemas relativos a solução de problemas DAE de índice elevado. Estas implementações dividem-se basicamente em dois grupos de códigos:

- Os projetados para problemas de uma dada estrutura (semi-explícito ou Hessemberg);
- Os aplicáveis para qualquer estrutura mas para índice no máximo igual a 3.

O primeiro grupo de códigos não é adequado para ferramentas orientadas a equa-

ções, uma vez que não se pode predizer a estrutura do problema. O segundo grupo é mais adequado, uma vez que problemas com índice superior a 3 não são tão comuns. Representando esta abordagem (aplicável a qualquer sistema com índice até 3) existe um código de passos múltiplos chamado MEBDF (CASH, 2000) e um tipo IRK chamado PSIDE (LIOEN et al., 1998).

A distorção na solução do problema (4.3) utilizando o método MEBDF também é apresentada na Figura 5.8. Como pode ser visto, não há distorção perceptível para este caso. Disto conclui-se que a melhor alternativa para a solução de problemas de índice elevado é a utilização de métodos especialmente projetados, entretanto estes estão limitado para problemas com índice até 3. Ambos os métodos, redução de índice por diferenciação e MEBDF, estão disponíveis no simulador de processos EMSO.

5.5.3 Problemas onde o índice estrutural supera o índice

Reißig et al. (2000) identificaram alguns problemas onde o índice estrutural supera em número o índice. Nesta seção os impactos no algoritmo de análise estrutural proposto neste trabalho e na solução numérica desta categoria de problema são analisados. Para tal, considere novamente o sistema (3.17), aqui reproduzido:

$$\begin{aligned} &C(v_1' - v_3') = i \\ &C(v_1' - v_3') = v_3/R \\ &v_1 = v(t) \end{aligned}$$

A análise deste sistema com o Algoritmo 5, gera a seqüência de grafos apresentada na Figura 5.9. Assim como o algoritmo de Pantelides, o novo algoritmo determina um índice estrutural igual a 2, pois a equação f_3 é diferenciada 2 vezes. Mais uma vez observa-se que, de fato, o índice estrutural pode superar em número o índice, uma vez que é bem sabido que o sistema (3.17) é de índice 1.

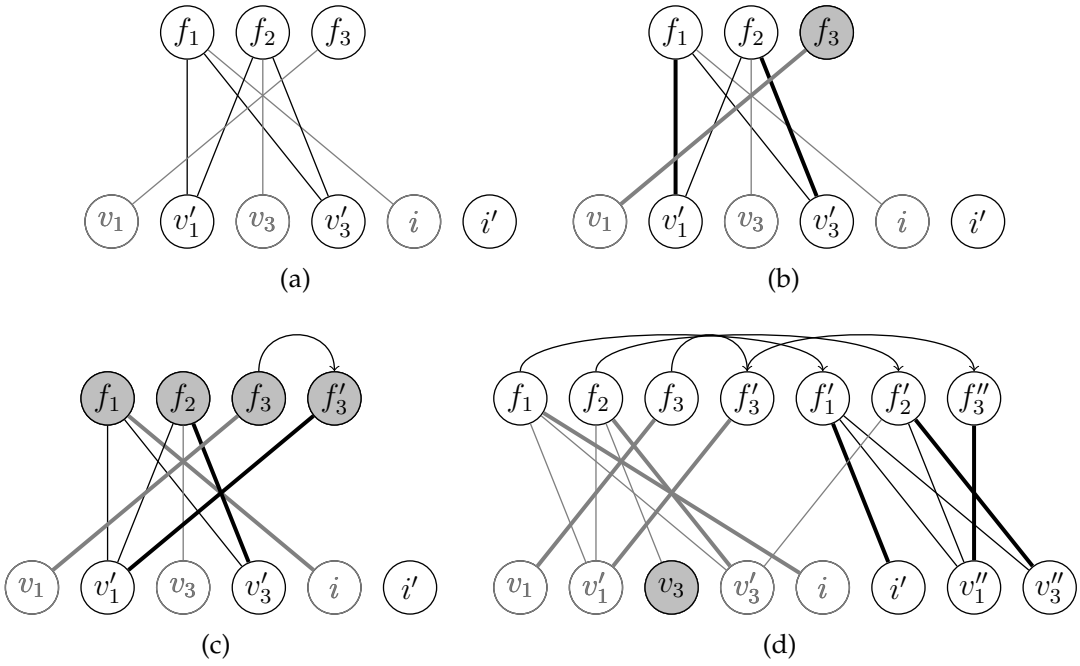


Figura 5.9: Aplicação do Algoritmo 5 ao sistema de equações (3.17).

Analisando-se a associação final apresentada na Figura 5.9(d), nota-se que o sistema contém apenas um grau de liberdade dinâmico e que v_3 é uma variável elegível para estado. Utilizando-se os caminhos alternantes iniciados em v_3, conclui-se que as variáveis que podem ser fornecidas como condição inicial são v_3 e i, pois como era de se esperar, a análise identifica corretamente que a variável v_1 é bem determinada pela equação $v_1 = v(t)$.

A saída da análise estrutural disponível no simulador EMSO para o problema (3.17) segue:

```
Number of variables:  3
Number of equations:  3
Number of specifications:  0
Degrees of freedom: 0
Structural differential index: 2
Extra Equations:    4
Extra Variables:    2
Dynamic degrees of freedom:  1
Number of initial Conditions: 1
```

Se a substituição da expressão $C(v_1' - v_3') = i$ é aplicada ao problema (3.17), uma versão mais simples do sistema é gerada:

$$\begin{aligned} &C(v_1' - v_3') = i \\ &i = v_3/R \\ &v_1 = v(t) \end{aligned} \tag{5.10}$$

Para o sistema (5.10) o índice estrutural coincide com o índice, ambos são iguais a 1. Uma observação interessante é que, embora os sistemas (3.17) e (5.10) apresentem índices estruturais diferentes, eles são indistinguíveis em termos de dificuldade de obtenção da solução numérica. Nos testes realizados, ambos sistemas requerem exatamente o mesmo número de avaliações dos resíduos e atualizações da Jacobiana. Esta é uma observação importante, uma vez que Reißig et al. (2000) sugeriram que a solução numérica de sistemas onde o índice estrutural supera o índice poderia ser mais difícil do que o índice sugere.

Como conclusão, os casos onde o índice estrutural supera o índice não impactam no poder de análise do algoritmo proposto neste trabalho (Algoritmo 5). Pois, mesmo nestes casos, o algoritmo ainda revela corretamente o número de graus de liberdade dinâmicos e as variáveis elegíveis a estados. Além disto, tal categoria de problema não apresenta dificuldades numéricas adicionais, contrariando a suposição de Reißig et al. (2000).

5.5.4 Eventos Discretos em Sistemas de Índice Elevado

Na Seção 5.4 foi abordada a questão da reinicialização de sistemas DAE quando ocorrem descontinuidades em suas funções propondo uma metodologia de reinicialização. Embora a técnica proposta não imponha nenhuma limitação quanto à forma do sistema DAE nem ao seu índice, o exemplo apresentado foi um sistema simples de índice 1.

Para a demonstração da eficácia da técnica quando aplicada a sistemas de índice elevado considere novamente o sistema do pêndulo (4.3), para o caso onde a aceleração gravidade sofre uma descontinuidade:

$$\begin{aligned} & w = x' \\ & z = y' \\ & Tx = w' \\ & Ty - g = z' \\ & x^2 + y^2 = L^2 \\ & g = \begin{cases} g_0, t < t_1 \\ g_0 - \Delta g, t \geq t_1 \end{cases} \end{aligned} \tag{5.11}$$

A situação hipotética apresentada no conjunto de equações (5.11) é equivalente a, para o tempo igual a t_1, acelerar o sistema na mesma direção da gravidade com uma aceleração constante e igual a Δg. O sistema (5.11), assim como o sistema sem a descontinuidade (4.3), apresenta dois graus de liberdade dinâmicos. Assim, no caso de uma descontinuidade, a equação da continuidade (4.2) deverá ser utilizada para apenas duas variáveis, satisfazendo exatamente os graus de liberdade dinâmicos.

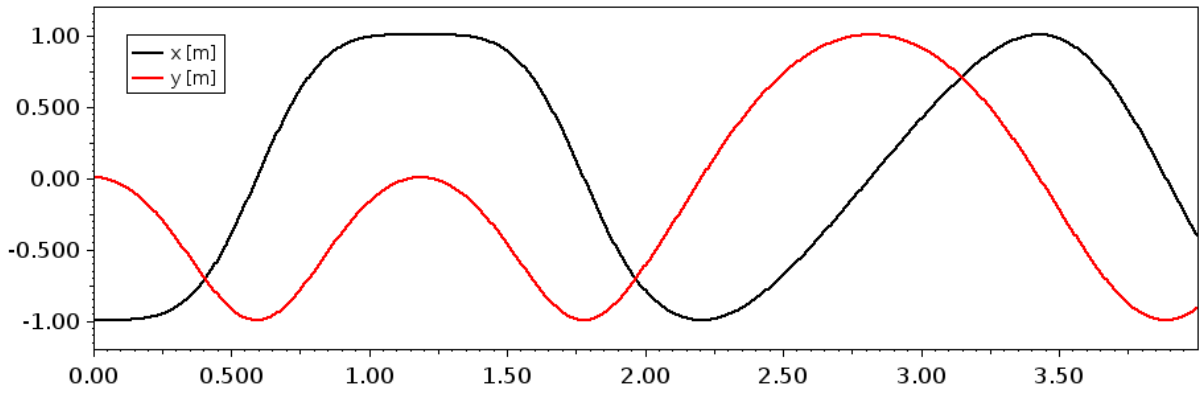
Na Subseção 4.2.6 os estados do sistema (4.3) foram estudados. Os conjuntos de variáveis que formam os estados podem ser qualquer combinação de $\{x, y\}$ e $\{z, w\}$. Assim, qualquer combinação de duas variáveis destes conjuntos poderá ser utilizada na equação da continuidade (4.2). Se esta afirmação é correta, então pode-se gerar o seguinte teorema:

Teorema 5.1 *Todas as variáveis determinadas como elegíveis a estados pelo Algoritmo 5 devem permanecer constantes em um evento de descontinuidade em qualquer das funções, mesmo estas variáveis sejam, em número, maiores que o número de graus de liberdade dinâmicos.*

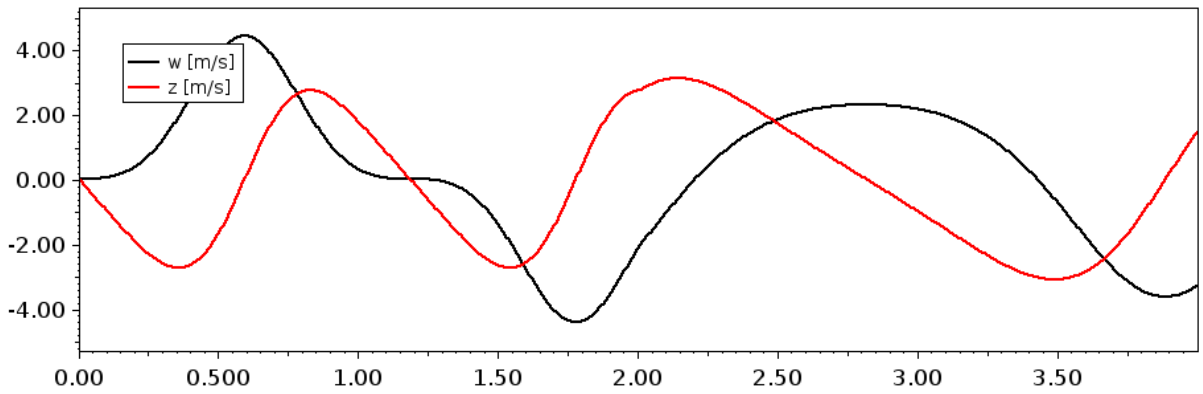
O 5.1 é fundamentado no fato de que a seleção de diferentes variáveis para a utilização na equação da continuidade deve fornecer uma única solução. Em cada seleção diferente de estados a continuidade destes é garantida e, como todas as soluções

devem ser idênticas, só se pode concluir que todas as variáveis que podem ser selecionadas como estados devem ser contínuas em um evento de descontinuidade nas funções. Também é possível utilizar o sistema (5.11) para a verificação do 5.1. A solução numérica deste problema para $t_1 = 2$, $g_0 = 9,8$ e $\Delta g = 7,8$ pode ser vista na Figura 5.10. Para o obtenção dos resultados presentes nesta figura, a continuidade foi assumida para as variáveis y e z, mas como pode ser observado as outras duas variáveis elegíveis a estado x e w também apresentaram continuidade.

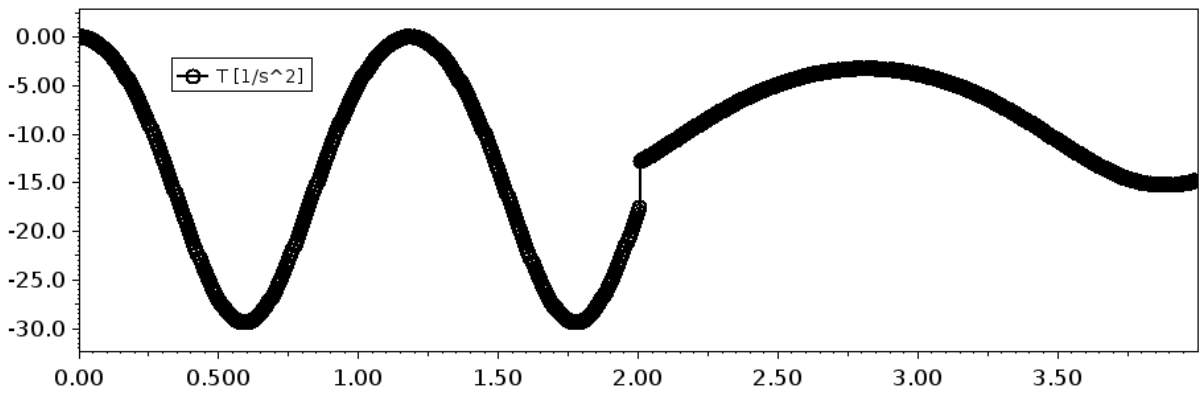
Como pode ser visto em Figura 5.10(c), o resultado não foi o mesmo para a variável T. A continuidade desta variável não estava garantida, uma vez que ela não faz parte do conjunto de possíveis estados. A sutil mudança de direção de z em t_1, presente na Figura 5.10(b), também indica que z' é descontínuo no evento. O mesmo acontece com w', este fato é quase imperceptível na Figura 5.10(b), mas fica claro na Figura 5.11.



(a)



(b)



(c)

Figura 5.10: Solução numérica do sistema contínuo-discreto de índice elevado (5.11).

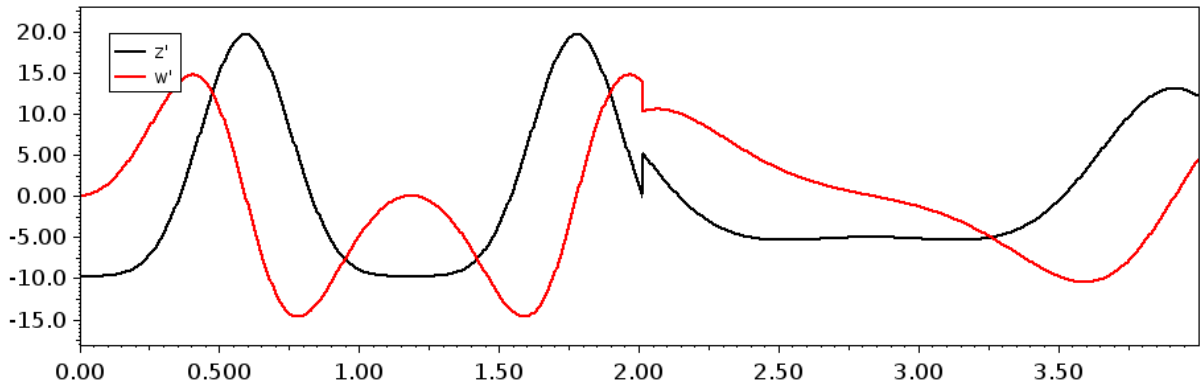


Figura 5.11: Detalhe de descontinuidade de derivadas na solução numérica do sistema contínuo-discreto de índice elevado (5.11).

Capítulo 6

Conclusões

O ponto central desta tese é a análise estrutural de sistemas de equações, com um enfoque na depuração de problemas de modelagem e simulação. Para o caso de sistemas estáticos a literatura é rica, mas para o caso dinâmico as técnicas disponíveis são muito limitadas. A principal contribuição deste trabalho foi estender as técnicas antes aplicáveis apenas em modelos estáticos para o caso dinâmico. Este capítulo sumaria os passos necessários para a realização desta tarefa.

- Na Seção 2.6 foi observado que, quando a decomposição DM revela uma partição sub-determinada, nem todas as combinações de especificações de variáveis pertencentes à esta partição são capazes de gerar um sistema estruturalmente não-singular. Entretanto, utilizando-se a associação ótima final e os caminhos alternantes induzidos pelos nós de variáveis expostos é possível determinar os conjuntos de especificações viáveis, este método foi chamado de *depuração fina*. Isto pode ser obtido pela enumeração de todas as associações ótimas possíveis. Uma implementação baseada no trabalho de (UNO, 2001) foi sugerida como trabalho para investigações futuras. O mesmo método pode ser utilizado para depurar problemas sobre-especificados, onde o usuário pode ser aconselhado a remover determina-

dos conjuntos de equações, mas não qualquer combinação das equações da partição sobre-determinada.

- Na Seção 3.5 foi apresentada uma ampla revisão dos métodos disponíveis na literatura para análise estrutural de sistemas DAE. Pôde-se concluir que, historicamente, quando se trata de sistemas DAE, a análise de resolubilidade se confunde muito com o conceito de índice e a determinação do número de graus de liberdade dinâmicos. Entretanto, para permitir uma depuração dos sistemas DAE, a análise estrutural deve ir além da determinação do índice e dos graus de liberdade.

- Na Seção 3.6 o algoritmo mais utilizado para análise estrutural de DAEs (PANTELIDES, 1988a) foi estruturado de uma forma diferente da original. Nesta forma, foi possível perceber as suas semelhanças com os métodos para análise de sistemas não-lineares. As deficiências do método como uma ferramenta de depuração foram estudadas em detalhe, ficando claro que a análise estrutural para sistemas DAE se encontrava em um patamar muito inferior quando comparado aos sistemas NLA.

- Na Seção 4.2 um novo método para análise estrutural de sistemas DAE foi proposto. Este método remove todas as deficiências de natureza estrutural do algoritmo de Pantelides, identificadas no Capítulo 3. Também foi apresentada a conveniência da utilização deste novo método para fins de depuração, deixando a análise estrutural de sistemas dinâmicos em um patamar equivalente aos sistemas estáticos.

- Na Subseção 3.7.2 o conceito de *estado* para sistemas DAE genéricos foi estudado. Utilizando o algoritmo de análise estrutural proposto, foi apresentado um método para determinação do número de estados e dos conjuntos de variáveis elegíveis a estados. Após, foi demonstrado como a determinação dos *estados* é útil na simulação de sistemas híbridos contínuo-discreto.

Nesta linha, conclusões interessantes sobre a continuidade de todas as variáveis elegíveis a estados foram apresentadas na Subseção 5.5.1.

- Diversas aplicações das técnicas apresentadas neste trabalho foram apresentadas no Capítulo 5. Estas aplicações reforçam a contribuição do novo algoritmo proposto para a análise estrutural de sistemas DAE. Neste capítulo também foi sugerido, como trabalho futuro, a investigação de métodos que reduzam o efeito de *drift-off* na solução numérica de sistemas DAE baseados em redução de índice por diferenciação.

- Os casos onde o índice estrutural supera em número o índice diferencial também foram estudados neste trabalho. Na Subseção 5.5.3 foi visto que esta particularidade não impacta no poder de análise do algoritmo proposto neste trabalho (Algoritmo 5). Pois, mesmo nestes casos, o algoritmo ainda revela corretamente o número de graus de liberdade dinâmicos e as variáveis elegíveis a estados. Além disto, testes numéricos mostraram que esta categoria de problema não apresenta dificuldades numéricas adicionais, contrariando a suposição de Reißig et al. (2000).

Apêndice A

Algoritmos em C++

Para teste e execução dos algoritmos propostos neste trabalho, uma biblioteca para a representação de grafos foi implementada em C++. Esta biblioteca contém algumas classes e os algoritmos apresentados no trabalho. Além disto estão presentes rotinas para importação de grafos de arquivos de texto assim como a exportação para o formato das figuras apresentadas neste trabalho.

A.1 Bibliotecas para Grafos

Existem algumas bibliotecas que implementam classes e algoritmos da teoria de grafos. A biblioteca mais completa parece ser Leda (MEHLHORN et al., 1997), entretanto esta não é livre. No trabalho de (LEE et al., 1999) uma pesquisa sobre as biblitecas e algoritmos disponíveis é feita e o desenvolvimento de uma nova biblioteca é iniciado. No decorrer deste desenvolvimento a BGL (*Boost Graph Library*) foi construída (SIEK et al., 2002). Esta biblioteca é distribuida sob uma licensa de código livre e utiliza programação genérica (*tempates* em C++). Esta característica torna a biblioteca extremamente flexível, porém em detrimento da sua usabilidade. A BGL comtempla uma ampla

gama de algoritmos da teoria de grafos, exemplos são:

- *Shortest Paths*
- *Minimum Spanning Tree*
- *Connected Components*
- *Maximum Flow*
- *Sparse Matrix Ordering*
- *Layout*
- *Clustering*

Entretanto o enfoque da BGL não está muito voltado para os grafos bipartidos os quais são utilizados neste trabalho. Uma vez que a versão atual da BGL (1.33) ainda não contém sequer o algoritmo rudimentar para associações ótimas em grafos bipartidos.

Por estes motivos, além da reaproveitar implementações anteriores, no escopo deste trabalho foi desenvolvida uma biblioteca em C++ específica para a análise de sistemas de equações.

A.2 RPS Graph

A.2.1 Classes para Grafos Bipartidos

A Figura A.1 apresenta os diagramas de colaboração para as duas classes básicas da biblioteca. A classe `Node` representada os nós ou vértices e a classe `Graph` representa

um grafo bipartido em V_e e V_v. Como pode ser visto, os conjuntos V_e e V_v são listas de `Nodes`. Estas listas representam, respectivamente, as equações e variáveis do problema em estudo.

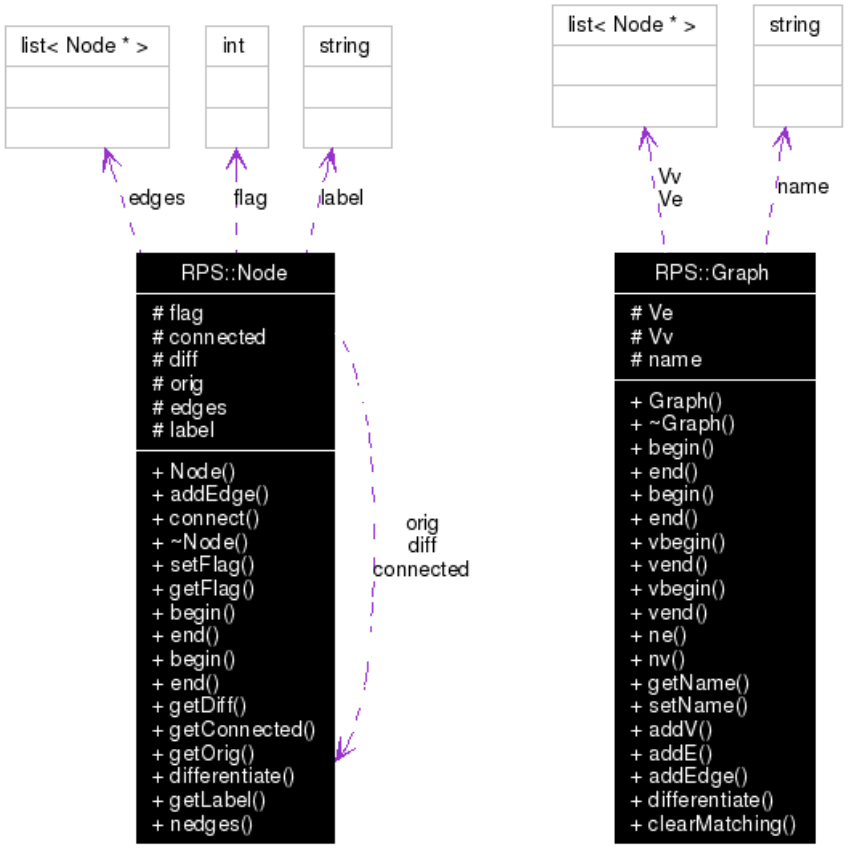


Figura A.1: Diagrama de classes de um nó (`Node`) e grafo (`Graph`).

Cada `Node` também contém uma lista de nós, que são as arestas (`edges`) do grafo. Assim, não exite uma classe para as arestas, estas estão embutidas nos vérices. Esta forma de representação se mostrou muito eficiente, uma vez que é comum nos algoritmos a necessidade de iterar por todos os nós adjacentes a um dado nó. Em contrapartida cada aresta $\{v_e - v_v\}$ precisa ser salva duas vezes, pois v_v está na lista de `edges` de v_e e *vice-versa*. Mas o gasto de memória acarretado é irrelevante, pois trata-se apenas de um *ponteiro* adicional por aresta.

Utilizando-se as classes da Figura A.1 a utilização da biblioteca se resume nos passos:

- Criar um grafo representado o sistema a ser analisado
- Executar um ou mais algoritmos ao grafo

Os algoritmos desenvolvidos são tratados na Subseção A.2.2. Para a construção do grafo existem algumas opções, tratadas a seguir.

A.2.1.1 Criação dos Grafos

Uma opção para a criação de um grafo é especificar diretamente quais arestas fazem parte do mesmo. Este procedimento está exemplificado no Código A.1.

Código A.1: Criação de um grafo indicando as arestas.

```
#include "RPSgraph.h"

// The graph object
RPS::Graph G("reactor");

// Add the edges automatically create the nodes
G.addEdge("f_1", "C"); G.addEdge("f_1", "R"); G.addEdge("f_1", "C'");
G.addEdge("f_2", "T"); G.addEdge("f_2", "R"); G.addEdge("f_2", "T_c");
G.addEdge("f_2", "T'");
G.addEdge("f_3", "C"); G.addEdge("f_3", "T"); G.addEdge("f_3", "R");
G.addEdge("f_4", "C");
```

Utilizando-se o procedimento apresentado no Código A.1 fica claro que não podem ser incluídos nós isolados ao grafo. Note-se também que nós de variável terminados com um apóstrofe, como por exemplo `C'`, são detectados automaticamente como a versão derivada de um nó, no caso `C`.

A criação de um grafo também pode ser executada através da criação dos nós explicitamente, como apresentado no Código A.2.

Uma terceira via, e talvez a mais conveniente, é a importação do grafo de um

Código A.2: Criação de um grafo indicando as arestas.

```
#include "RPSgraph.h"

// The graph object
RPS::Graph G("reactor");
RPS::Node *e, *v;

// Add the nodes
G.addE(e = new RPS::Node("f_1"));
G.addV(v = new RPS::Node("C"));
e->addEdge(v);
G.addV(v = new RPS::Node("R"));
e->addEdge(v);
```

arquivo de textos previamente construído. Este procedimento está demonstrado no Código A.3.

Código A.3: Criação de um grafo através da importação de um arquivo de texto.

```
#include "RPSgraph.h"
#include <fstream>

// The graph object
RPS::Graph G("unnamed");

// Import the contents
std::ifstream file("reactor.dot");
if(RPS::ImportGraphViz(file, G))
      return 1;
```

A função `ImpotGraphViz` é capaz de interpretar um subconjunto da linguagem DOT (GANSNER; NORTH, 2000). Esta linguagem é utilizada pela ferramenta *GraphViz*, a qual gera representações visuais da relação entre objetos, incluindo os grafos. Um exemplo de arquivo na linguagem DOT é apresentado no Código A.4.

Código A.4: Grafo para o sistema (3.12) na linguagem DOT.

```
graph Reactor{
 f_1--C  f_1--R  f_1--C'
 f_2--T  f_2--R  f_2--T_c f_2--T'
 f_3--C  f_3--T  f_3--R
 f_4--C
}
```

Como pode ser visto no Código A.4, o arquivo DOT contém o nome do grafo e as relações entre os nós. Para cada comando do tipo `f_1--C` o primeiro nó é considerado a equação e o segundo a variável. Arquivos como este, para todos os exemplos tratados neste trabalho, podem ser obtidos com o autor.

A.2.2 Algoritmos

A biblioteca de grafos introduzida nesta seção está dividida em classes e algoritmos. As classes (Subseção A.2.1) representam os dados do problema e os algoritmos são as técnicas aplicadas aos dados para produzir resultados. Nesta seção são apresentados os algoritmos de interesse neste trabalho.

A.2.2.1 Associação Ótima

Como apresentado na Subseção 2.2.3 uma associação ótima em um grafo bipartido pode ser obtida pela procura por caminhos alternativos com relação a uma associação qualquer. Uma implementação em C++ para o Algoritmo 1, utilizado para aumentar a cardinalidade de uma dada associação, é apresentada no Código A.5.

O Código A.6 (que representa o Algoritmo 2) pode ser utilizado para determinar uma associação ótima de um grafo bipartido. Este código executa o Código A.5 para cada nó de equação e retorna *verdadeiro* se a associação é perfeita com relação às equações.

Código A.5: Código em C++ para aumento da cardinalidade de uma associação.

```
int RPS::AugmentMatching(RPS::Graph &G, RPS::Node *ve){
      std::list<Node*>::iterator vv;
      // colour ve
      ve->setFlag(ve->getFlag() | COLORED);
      // try a direct connection
      for(vv = ve->begin(); vv!=ve->end(); ++vv){
            if(!(*vv)->getConnected() && !((*vv)->getFlag()&DELETED)){
                  ve->connect(*vv);
                  ve->setFlag(ve->getFlag()&~COLORED);
                  return SUCESS;
            }
      }
      // try an alternating path
      for(vv = ve->begin(); vv!=ve->end(); ++vv){
            Node *ve2 = (*vv)->getConnected();
            if(!((*vv)->getFlag()&DELETED) && ve2 && !(ve2->getFlag()&
               COLORED)
                  && AugmentMatching(G, ve2)){
                  ve->connect(*vv);
                  ve->setFlag(ve->getFlag()&~COLORED);
                  return SUCESS;
            }
      }
      return FAILED;
}
```

A.2.2.2 Pantelides

Uma implementação em C++ do algoritmo de Pantelides (Algoritmo 3 tratado na Seção 3.6) é apresentada no Código A.7. Este código é muito semelhante ao código para obtenção de uma associação de máxima cardinalidade (Código A.6).

A.2.2.3 Novo Algoritmo para Análise de DAEs

O novo algoritmo para análise de sistemas DAE, proposto na Seção 4.2, é apresentado no Código A.8. Este algoritmo necessita de uma versão modificada do método para a procura por caminhos alternativos. Esta modificação consiste no Algoritmo 4 e uma implementação em C++ é apresentada no Código A.9.

Código A.6: Código em C++ para determinar a associação ótima de um grafo bipartido.

```
bool RPS::MaximumMatching(Graph &G){
      std::list<Node*>::iterator ve;
      bool isPerfect = true;
      // augment the matching one by one
      for(ve = G.begin(); ve!=G.end(); ++ve){
            if(!(*ve)->getConnected() && !AugmentMatching(G, *ve)){
                  isPerfect = false;
                  uncolour(G.begin(), G.end());
            }
      }
      return isPerfect;
}
```

Código A.7: Código em C++ para o algoritmo de Pantelides.

```
int RPS::Pantelides(Graph &G){
      std::list<Node*>::iterator ve, ve2, vv;
      bool shouldReturn = false;

      // remove the pure algebraic variables (x)
      for(vv = G.vbegin(); vv!= G.vend(); ++vv){
            if((*vv)->getDiff() && (*vv)->getDiff()->nedges() && !((*vv)->
                getFlag()&DELETED)){
                   (*vv)->setFlag((*vv)->getFlag() | DELETED);
                   shouldReturn = true;
            }
      }
      // augment the matching one by one
      for(ve = G.begin(); ve!=G.end(); ++ve){
            if(!(*ve)->getDiff() && !(*ve)->getConnected() && !
                AugmentMatching(G, *ve)){
                   // diff all COLORED equations
                   for(ve2 = G.begin(); ve2!=G.end(); ++ve2){
                         if((*ve2)->getFlag() & COLORED){
                               G.differentiate(*ve2);
                               (*ve2)->setFlag((*ve2)->getFlag()&~COLORED);
                               ve2 = G.begin();
                         }
                   }
            }
      }
      return true;
}
```

Código A.8: Código em C++ para o novo algoritmo para análise de sistemas DAE.

```
int RPS::DAEAnalysis(Graph &G){
      std::list<Node*>::iterator ve;
      bool singular, needDiff;

      while(true){
            singular = needDiff = false;
            // augment the matching one by one
            for(ve = G.begin(); ve!=G.end(); ++ve){
                  if(!(*ve)->getConnected() && !AugmentMatching2(G, *ve,
                     false)){
                        needDiff = true;
                        markColored(G.begin(), G.end());
                        if(!AugmentMatching2(G, *ve, true))
                              singular = true;
                  }
            }
            if(singular)
                  return FAILED;
            if(!needDiff)
                  return SUCESS;
            // diff all COLORED equations
            for(ve = G.begin(); ve!=G.end(); ++ve){
                  if((*ve)->getFlag() & MARKED){
                        G.differentiate(*ve);
                        (*ve)->setFlag((*ve)->getFlag()&~MARKED);
                        ve = G.begin();
                  }
            }
      }
}
```

Código A.9: Código em C++ para a versão modificada do método para aumento da cardinalidade de uma associação.

```
int RPS::AugmentMatching2(RPS::Graph &G, RPS::Node *ve, bool alg){
      std::list<Node*>::iterator vv;
      // colour ve
      ve->setFlag(ve->getFlag() | COLORED);
      // try a direct connection
      for(vv = ve->begin(); vv!=ve->end(); ++vv){
            if(!(*vv)->getConnected() && !((*vv)->getFlag()&DELETED) &&
                        isElegible(*vv, alg)){
                  ve->connect(*vv);
                  ve->setFlag(ve->getFlag()&~COLORED);
                  return SUCESS;
            }
      }
      // try an alternating path
      for(vv = ve->begin(); vv!=ve->end(); ++vv){
            Node *ve2 = (*vv)->getConnected();
            if(!((*vv)->getFlag()&DELETED) && isElegible(*vv, alg) &&
                        ve2 && !(ve2->getFlag()&COLORED) &&
                           AugmentMatching2(G, ve2, alg)){
                  ve->connect(*vv);
                  ve->setFlag(ve->getFlag()&~COLORED);
                  return SUCESS;
            }
      }
      return FAILED;
}

bool isElegible(RPS::Node *vv, bool alg){
      if(alg)
            return true;
      else
            return vv->getDiff() == NULL;
}
```

A.2.3 Licensa de Uso

rafael@rps.eng.br – `http://www.rps.eng.br`

Apêndice B

Listagem de Modelos

Neste apêndice são apresentados os códigos dos modelos utilizados nas aplicações das técnicas estudadas e desenvolvidas neste trabalho. Estes códigos estão escritos na linguagem de modelagem do simulador EMSO (SOARES; SECCHI, 2003).

B.1 Processo de Síntese de Amônia

Código B.1: Modelo de um compressor ideal.

```
Model Compressor
      PARAMETERS
outer PP as Plugin(Brief = "External Physical Properties", Type="PP");
outer NComp as Integer;

      VARIABLES
in    Inlet as stream;
out   Outlet as streamPH;

      EQUATIONS
      "Isentropic expansion"
      PP.VapourEntropy(Outlet.T, Outlet.P, Outlet.z) =
            PP.VapourEntropy(Inlet.T, Inlet.P, Inlet.z);

      "Global Molar Balance"
      Inlet.F = Outlet.F;
      "Component Molar Balance"
      Inlet.z = Outlet.z;
end
```

Código B.2: Modelo de um misturador simples com 2 entradas.

```
Model Mixer
      PARAMETERS
outer PP as Plugin(Brief = "External Physical Properties", Type="PP");
outer NComp as Integer;

      VARIABLES
in    Inlet1 as stream;
in    Inlet2 as stream;
out   Outlet as streamPH;

      EQUATIONS
      "Energy Balance"
      Outlet.F * Outlet.h = Inlet1.F * Inlet1.h + Inlet2.F * Inlet2.h;

      Inlet1.P = Outlet.P;

      "Global Molar Balance"
      Inlet1.F + Inlet2.F = Outlet.F;
      "Component Molar Balance"
      Inlet1.z*Inlet1.F + Inlet2.z*Inlet2.F = Outlet.F * Outlet.z;
end
```

Código B.3: Modelo de um reator simplificado - baseado na conversão.

```
Model Reactor
      PARAMETERS
outer PP as Plugin(Brief = "External Physical Properties", Type="PP");
outer NComp as Integer;
      NReac as Integer(Default=1);
      stoic(NComp, NReac) as Real (Brief = "Stoichiometric Matrix");
      comp(NReac) as Integer(Brief = "Key Component of the reaction");

      VARIABLES
in    Inlet as stream;
out   Outlet as streamPH;
      Outletz(NComp) as fraction;
      X(NReac) as fraction(Brief="Convertion of the key component");

      EQUATIONS
      "Energy Balance"
      Outlet.F * Outlet.h = Inlet.F * Inlet.h;

      "Global Molar Balance"
      Outlet.F = Inlet.F * (1 - sum(Outletz));

      for i in [1:NComp]
            "Component Molar Balance"
            Outletz(i) = Inlet.z(i) + sum(stoic(i,:)*X*Inlet.z(comp));
      end

      "Normalize the outlet composition"
      Outlet.z * sum(Outletz) = Outletz;

      Outlet.P = Inlet.P;
```

```
end
```

Código B.4: Modelo do processo de síntese de amônia.

```
FlowSheet Ammonia
      PARAMETERS
      PP    as Plugin(Brief="Physical Properties", Type="PP",
            Components = ["hydrogen", "nitrogen", "argon",
                  "methane", "ammonia"],
            LiquidModel = "APR",
            VapourModel = "APR");
      NComp as Integer;
      SET

      NComp = PP.NumberOfComponents;

      DEVICES
      FEED   as source;
      C101   as Compressor;
      R101   as Reactor;
      F101   as flash_steady;
      F102   as flash_steady;
      S101   as splitter;
      M101   as Mixer;
      M102   as Mixer;
      C102   as Compressor;

      VARIABLES
      purity as fraction(Brief="Purity of the product");
      production as flow_mol(DisplayUnit = 'lbmol/h', Brief="Ammonia in the
          product");
      loose as flow_mol(DisplayUnit = 'lbmol/h', Brief="Ammonia in the
         purge");
      Q1 as heat_rate;
      Q2 as heat_rate;

      CONNECTIONS
      FEED.Outlet    to   M101.Inlet1;
      M101.Outlet    to   C101.Inlet;
      C101.Outlet    to   M102.Inlet1;
      M102.Outlet    to   R101.Inlet;
      R101.Outlet    to   F101.Inlet;
      F101.OutletL   to   F102.Inlet;
      F102.OutletV   to   M101.Inlet2;
      F101.OutletV   to   S101.Inlet;
      S101.Outlet1   to   C102.Inlet;
      C102.Outlet    to   M102.Inlet2;

      Q1             to   F101.Q;
      Q2             to   F102.Q;

      SET
      R101.comp = 2; # Key component of the reaction
      R101.stoic = [-3, -1, 0, 0, 2]; # Stoichiometry of the reaction
```

```
      SPECIFY
      FEED.Outlet.F = 2000 * 'lbmol/h';
      FEED.Outlet.T = (27 + 273.15) * 'K';
      FEED.Outlet.P = 10 * 'atm';
      FEED.Outlet.z = [0.74, 0.24, 0.01, 0.01, 0.0];

      C101.Outlet.P = 200 * 'atm';
      C102.Outlet.P = 200 * 'atm';

      R101.X = 0.4; # Convertion of the reactor

      F101.OutletV.P = 199 * 'atm';
      F101.OutletV.T = (-34 + 273.15) * 'K';

      F102.OutletV.P = 10 * 'atm';
      F102.Q = 0 * 'kJ/h';

      # We can choose between one of the following specs
      S101.frac = 0.78; # Recycle fraction
      #loose = 1 * 'lbmol/h'; # Ammonia in the purge

      EQUATIONS
      production = purity * F102.OutletL.F;
      purity = F102.OutletL.z(5);
      loose  = S101.Outlet2.F * S101.Outlet2.z(5);

      OPTIONS
      Dynamic = false;
end
```

B.2 Processos de Destilação

Código B.5: Modelo básico de um prato de uma coluna de destilação - sem hidrodinâmica.

```
Model trayBasic
      PARAMETERS
outer PP as Plugin(Brief = "External Physical Properties", Type="PP");
outer NComp as Integer;
      V as volume(Brief="Total Volume of the tray");
      Q as heat_rate (Brief="Rate of heat supply");
      Ap as area (Brief="Plate area = Atray - Adowncomer");

      VARIABLES
in    Inlet as stream;
in    InletL as stream;
in    InletV as stream;
out   OutletL as liquid_stream;
out   OutletV as vapour_stream;

      M(NComp) as mol (Brief="Molar Holdup in the tray");
```

```
    ML as mol (Brief="Molar liquid holdup");
    MV as mol (Brief="Molar vapour holdup");
    E as energy (Brief="Total Energy Holdup on tray");
    vL as volume_mol (Brief="Liquid Molar Volume");
    vV as volume_mol (Brief="Vapour Molar volume");
    Level as length (Brief="Height of clear liquid on plate");
    yideal(NComp) as fraction;
    Emv as Real (Brief = "Murphree efficiency");

    EQUATIONS
    "Component Molar Balance"
    diff(M)=Inlet.F*Inlet.z + InletL.F*InletL.z + InletV.F*InletV.z
          - OutletL.F*OutletL.z - OutletV.F*OutletV.z;

    "Energy Balance"
    diff(E) = ( Inlet.F*Inlet.h + InletL.F*InletL.h + InletV.F*InletV.h
          - OutletL.F*OutletL.h - OutletV.F*OutletV.h + Q );

    "Molar Holdup"
    M = ML*OutletL.z + MV*OutletV.z;

    "Energy Holdup"
    E = ML*OutletL.h + MV*OutletV.h - OutletL.P*V;

    "Mol fraction normalisation"
    sum(OutletL.z)= 1.0;
    sum(OutletL.z)= sum(OutletV.z);

    "Liquid Volume"
    vL = PP.LiquidVolume(OutletL.T, OutletL.P, OutletL.z);
    "Vapour Volume"
    vV = PP.VapourVolume(OutletV.T, OutletV.P, OutletV.z);

    "Chemical Equilibrium"
    PP.LiquidFugacityCoefficient(OutletL.T, OutletL.P, OutletL.z)*OutletL
        .z =
          PP.VapourFugacityCoefficient(OutletV.T, OutletV.P, yideal)*
              yideal;

    "Murphree Efficiency"
    OutletV.z = Emv * (yideal - InletV.z) + InletV.z;

    "Thermal Equilibrium"
    OutletV.T = OutletL.T;

    "Mechanical Equilibrium"
    OutletV.P = OutletL.P;

    "Geometry Constraint"
    V = ML* vL + MV*vV;

    "Level of clear liquid over the weir"
    Level = ML*vL/Ap;
end
```

Código B.6: Modelo de prato com hidrodinâmica tipo Francis.

```
Model tray as trayBasic
      PARAMETERS
      Ah as area (Brief="Total holes area");
      lw as length (Brief="Weir length");
      g as acceleration (Default=9.81);
      hw as length (Brief="Weir height");
      beta as fraction (Brief="Aeration fraction");
      alfa as fraction (Brief="Dry pressure drop coefficient");

      VARIABLES
      rhoL as dens_mass;
      rhoV as dens_mass;

      EQUATIONS
      "Liquid Density"
      rhoL = PP.LiquidDensity(OutletL.T, OutletL.P, OutletL.z);
      "Vapour Density"
      rhoV = PP.VapourDensity(InletV.T, InletV.P, InletV.z);

      if Level > (beta * hw) then
            "Francis Equation"
            OutletL.F = 1.84*'1/s'*lw*((Level-(beta*hw))/(beta))^2/vL;
      else
            "Low level"
            OutletL.F = 0 * 'mol/h';
      end

end
```

Código B.7: Modelo de um condensador.

```
Model condenser
      PARAMETERS
      outer PP as Plugin(Brief = "External Physical Properties", Type="PP")
          ;
      outer NComp as Integer;
      V as volume (Brief="Condenser total volume");
      Across as area (Brief="Cross Section Area of reboiler");

      VARIABLES
in    InletV as stream(Brief="Vapour inlet stream");
out   OutletL as liquid_stream(Brief="Liquid outlet stream");
out   OutletV as vapour_stream(Brief="Vapour outlet stream");
in    Q as heat_rate (Brief="Heat supplied");

      M(NComp) as mol (Brief="Molar Holdup in the tray");
      ML as mol (Brief="Molar liquid holdup");
      MV as mol (Brief="Molar vapour holdup");
      E as energy (Brief="Total Energy Holdup on tray");
      vL as volume_mol (Brief="Liquid Molar Volume");
      vV as volume_mol (Brief="Vapour Molar volume");
      Level as length (Brief="Level of liquid phase");

      EQUATIONS
```

```
      "Component Molar Balance"
      diff(M) = InletV.F*InletV.z - OutletL.F*OutletL.z
                       - OutletV.F*OutletV.z;

      "Energy Balance"
      diff(E) = InletV.F*InletV.h - OutletL.F*OutletL.h
                       - OutletV.F*OutletV.h + Q;

      "Molar Holdup"
      M = ML*OutletL.z + MV*OutletV.z;

      "Energy Holdup"
      E = ML*OutletL.h + MV*OutletV.h - OutletV.P*V;

      "Mol fraction normalisation"
      sum(OutletL.z)=1.0;
      sum(OutletL.z)=sum(OutletV.z);

      "Liquid Volume"
      vL = PP.LiquidVolume(OutletL.T, OutletL.P, OutletL.z);
      "Vapour Volume"
      vV = PP.VapourVolume(OutletV.T, OutletV.P, OutletV.z);

      "Chemical Equilibrium"
      PP.LiquidFugacityCoefficient(OutletL.T, OutletL.P, OutletL.z)*OutletL
          .z =
             PP.VapourFugacityCoefficient(OutletV.T, OutletV.P, OutletV.z)*
                 OutletV.z;

      "Thermal Equilibrium"
      OutletL.T = OutletV.T;

      "Mechanical Equilibrium"
      OutletV.P = OutletL.P;

      "Geometry Constraint"
      V = ML*vL + MV*vV;

      "Level of liquid phase"
      Level = ML*vL/Across;
end
```

Código B.8: Modelo de um refervedor.

```
Model reboiler
      PARAMETERS
      outer PP as Plugin(Brief = "External Physical Properties", Type="PP")
          ;
      outer NComp as Integer;
      Across as area (Brief="Cross Section Area of reboiler");
      V as volume (Brief="Total volume of reboiler");

      VARIABLES
in    Inlet as stream(Brief="Feed Stream");
in    InletL as stream(Brief="Liquid inlet stream");
```

```
out    OutletL as liquid_stream(Brief="Liquid outlet stream");
out    OutletV as vapour_stream(Brief="Vapour outlet stream");
in     Q as heat_rate (Brief="Heat supplied");

       M(NComp) as mol (Brief="Molar Holdup in the tray");
       ML as mol (Brief="Molar liquid holdup");
       MV as mol (Brief="Molar vapour holdup");
       E as energy (Brief="Total Energy Holdup on tray");
       vL as volume_mol (Brief="Liquid Molar Volume");
       vV as volume_mol (Brief="Vapour Molar volume");
       Level as length (Brief="Level of liquid phase");
       rhoV as dens_mass (Brief="Vapour Density");

       EQUATIONS
       "Component Molar Balance"
       diff(M)= Inlet.F*Inlet.z + InletL.F*InletL.z
             - OutletL.F*OutletL.z - OutletV.F*OutletV.z;

       "Energy Balance"
       diff(E) = Inlet.F*Inlet.h + InletL.F*InletL.h
             - OutletL.F*OutletL.h - OutletV.F*OutletV.h + Q;

       "Molar Holdup"
       M = ML*OutletL.z + MV*OutletV.z;

       "Energy Holdup"
       E = ML*OutletL.h + MV*OutletV.h - OutletL.P*V;

       "Mol fraction normalisation"
       sum(OutletL.z)=1.0;
       sum(OutletL.z)=sum(OutletV.z);

       "Vapour Density"
       rhoV = PP.VapourDensity(OutletV.T, OutletV.P, OutletV.z);

       "Liquid Volume"
       vL = PP.LiquidVolume(OutletL.T, OutletL.P, OutletL.z);

       "Vapour Volume"
       vV = PP.VapourVolume(OutletV.T, OutletV.P, OutletV.z);

       "Chemical Equilibrium"
       PP.LiquidFugacityCoefficient(OutletL.T, OutletL.P, OutletL.z)*OutletL
          .z =
             PP.VapourFugacityCoefficient(OutletV.T, OutletV.P, OutletV.z)*
                OutletV.z;

       "Mechanical Equilibrium"
       OutletL.P = OutletV.P;

       "Thermal Equilibrium"
       OutletL.T = OutletV.T;

       "Geometry Constraint"
       V = ML*vL + MV*vV;
```

```
      "Level of liquid phase"
      Level = ML*vL/Across;
end
```

Código B.9: Modelo de uma coluna de destilação com condensador e refervedor dinâmicos.

```
Model Distillation_kettle_cond
      PARAMETERS
      outer PP as Plugin(Brief = "External Physical Properties", Type="PP")
          ;
      outer NComp as Integer;
      NTrays as Integer(Brief="Number of trays", Default=2);
      topdown as Integer(Brief="Trays counting (1=top-down, -1=bottom-up)",
           Default=1);
      top as Integer(Brief="Number of top tray");
      bot as Integer(Brief="Number of bottom tray");

      SET
      top = (NTrays-1)*(1-topdown)/2+1;
      bot = NTrays/top;

      VARIABLES
      trays(NTrays) as tray;
      cond as condenser;
      reb as reboiler;
      sptop as splitter;
      pump1 as pump;

      EQUATIONS
      if (reb.OutletV.P > reb.InletL.P) then
            "Pressure Drop through the reboiler"
            reb.OutletV.F = trays(bot).Ah/reb.vV * sqrt((reb.OutletV.P -
               trays(bot).OutletL.P)
                        / (trays(bot).beta*reb.rhoV) );
      else
            "No flow in reboiler"
            reb.OutletV.F = 0.0 * 'mol/s';
      end

      "Pressure Drop through the tray"
      trays(top).OutletV.F = (1 + tanh(1 * (trays(top).OutletV.P - cond.
         OutletL.P)/'Pa'))/2 *
            trays(top).Ah/trays(top).vV * sqrt(2*(trays(top).OutletV.P -
            cond.OutletL.P + 1e-8 * 'atm') / (trays(top).alfa*trays(top).
               rhoV));

      trays([top+topdown:topdown:bot]).OutletV.F = (1 + tanh(1 *
          (trays([top+topdown:topdown:bot]).OutletV.P -
          trays([top+topdown:topdown:bot]).InletL.P)/'Pa'))/2 *
          trays([top+topdown:topdown:bot]).Ah/trays([top+topdown:topdown:
             bot]).vV *
          sqrt(2*(trays([top+topdown:topdown:bot]).OutletV.P -
            trays([top+topdown:topdown:bot]).InletL.P + 1e-8 * 'atm') /
```

```
        (trays([top+topdown:topdown:bot]).alfa*trays([top+topdown:topdown
            :bot]).rhoV));

    CONNECTIONS
    #vapor
    reb.OutletV to trays(bot).InletV;
    trays([top+topdown:topdown:bot]).OutletV to trays([top:topdown:bot-
        topdown]).InletV;
    trays(top).OutletV to cond.InletV;

    #liquid
    cond.OutletL to sptop.Inlet;
    sptop.Outlet2 to pump1.Inlet;
    pump1.Outlet to trays(top).InletL;
    trays([top:topdown:bot-topdown]).OutletL to trays([top+topdown:
        topdown:bot]).InletL;
    trays(bot).OutletL to reb.InletL;
end
```

Referências Bibliográficas

ALLAN, B. A. *A More Reusable Modeling System*. Tese (Doutorado) — Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA, 1997.

ALT, H.; BLUM, N.; MEHLHORN, K.; PAUL, M. Computing a maximum cardinality matching in a bipartite graph in time $O(n^{1.5}\sqrt{m/\log n})$. *Information Processing Letters*, North-Holland, Amsterdam, The Netherlands, v. 37, n. 4, p. 237–240, 1991.

AMESTOY, P.; ENSEEIHT-IRIT; DAVIS, T. A.; DUFF, I. S. Algorithm 837: Amd, an approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, v. 30, n. 3, p. 381–388, 2004.

ASHCRAFT, C.; LIU, J. W. H. Applications of the dulmage–mendelsohn decomposition and network flow to graph bisection improvement. *SIAM J. Matrix Anal. Appl.*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, v. 19, n. 2, p. 325–354, 1998. ISSN 0895-4798.

BACHMANN, R.; BRÜLL, L.; MRZIGLOD, T.; PALLASKE, U. On methods for reducing the index of differential algebraic equations. *Computers and Chemical Engineering*, v. 14, n. 11, p. 1271–1273, 1990.

BERGE, C. Two theorems in graph theory. *Proc. Nat. Acad. Sci.*, v. 43, p. 842–844, 1957.

BIEGLER, L. T.; GROSSMANN, I. E.; WESTERBERG, A. W. *Systematic Methods of Chemical Process Design*. [S.l.]: Prentice Hall International Series in Phisical and Chemical Engineering Sciences, 1997.

BLIEK, C.; NEVEU, B.; TROMBETTONI, G. Using graph decomposition for solving continuous csps. In: *CP '98: Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming*. London, UK: Springer-Verlag, 1998. p. 102–116. ISBN 3-540-65224-8.

BRENAN, K. E.; CAMPBELL, S. L.; PETZOLD, L. R. *Numerical Solution of Initial-Value Problem in Differential-Algebraic Equations*. New York: North-Holland, 1989.

BUNUS, P. *Debugging and Structural Analysis of Declarative Equation-Based Languages*. Tese (Doutorado) — Department of Computer and Information, Science Linköpings Universitet, Linköping, Sweden, 2002.

CASH, J. R. Modified extended backward differentiation formulae for the numerical solution of stiff initial value problems in odes and daes. *Journal of Computational and Applied Mathmatics*, v. 125, p. 117–130, 2000.

CHUA, L. O.; DESOER, C. A.; KUH, E. S. *Linear and Nonlinear Circuits*. [S.l.]: McGraw HillMcGraw Hill, New York, 1987.

Costa Jr., E. da. *Resolução automática de equações algébrico-diferenciais de índice superior*. Tese (Doutorado) — COPPE/UFRJ, Rio de Janeiro, Brasil, mar. 2003.

CURTISS, C.; HIRSCHFELDER, J. Integration of stiff equations. *Proc. Nat. Acad. Sci.*, v. 38, p. 235–243, 1958.

DEUFLHARD, P.; HAIRER, E.; ZUGCK, J. One step and extrapolation methods for differential-algebraic systems. *Numerische Mathematik*, v. 51, p. 501–516, 1987.

DIESTEL, R. *Graph theory*. 2. ed. New York: Springer-Verlag, 2000. 312 p. Disponível em: <http://www.math.uni-hamburg.de/home/diestel/books/graph.theory-/GraphTheoryII.pdf>.

DUFF, I. S.; ERISMAN, A. M.; REID, J. K. *Direct methods for sparse matrices*. New York, NY, USA: Oxford University Press, Inc., 1986. ISBN 0-198-53408-6.

DUFF, I. S.; GEAR, C. W. Computing the structural index. *SIAM Journal on Algebraic and Discrete Methods*, v. 7, n. 4, p. 594–603, 1986.

DUFF, I. S.; REID, J. K. An implementation of Tarjan's algorithm for the block triangularization of a matrix. *ACM Transactions on Mathematics and Software*, v. 4, n. 2, p. 137–147, 1978.

DULMAGE, A. L.; MENDELSOHN, N. S. Coverings of bipartite graphs. *Canad. J. Math.*, n. 10, p. 517–534, 1958.

ESPOSITO, J. M.; KUMAR, V.; PAPPAS, G. J. Accurate event detection for simulating hybrid systems. *Lecture Notes in Computer Science*, v. 2034, p. 204–217, 2001. Disponível em: <http://citeseer.ist.psu.edu/esposito01accurate.html>.

FEEHEY, W. F.; BARTON, P. I. Dynamic optimization with state variables path constraints. *CCE*, v. 22, p. 1241 – 1256, 1998.

GANSNER, E. R.; NORTH, S. C. An open graph visualization system and its applications to software engineering. *Software — Practice and Experience*, v. 30, n. 11, p. 1203–1233, 2000. Disponível em: <http://citeseer.ist.psu.edu/gansner99open%-.html>.

GEAR, C. W. The simultaneous numerical solution of differential-algebraic equations. *IEEE Transactions on Circuit Theory*, CT-18, p. 89–95, 1971.

GEAR, C. W. Differential-algebraic equation index transformation. *SIAM J. Sci. Stat. Comp.*, v. 9, n. 1, p. 39–47, jan. 1988.

GOLDSTEIN; STANFIELD. Exxon flowsheeting system. *I&EC Proc. Design*, v. 9, n. 1, p. 78, 1970.

GRIEWANK, A.; JUEDES, D. W.; UTKE, J. Algorithm 755: Adol-c: A package for the automatic differentiation of algorithms written in c/c++. *ACM Trans. Math. Softw.*, v. 22, n. 2, p. 131–167, 1996.

HAIRER, E.; WANNER, G. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Berlin: Springer-Verlag, 1996.

HAIRER, E.; WANNER, G. Stiff differential equations solved by radau methods. *Journal of Computational and Applied Mathmatics*, v. 111, p. 93–111, 1999.

HINDMARSH, A. C. LSODE and LSODI, two new initial value ordinary differential equation solvers. *ACM-SIGNUM Newsletters*, v. 15, p. 10–11, 1980.

HINDMARSH, A. C.; BROWN, P. N.; GRANT, K. E.; LEE, S. L.; SERBAN, R.; SHUMAKER, D. E.; WOODWARD, C. S. Sundials: Suite of nonlinear and differential/algebraic equation solvers. *ACM Trans. Math. Softw.*, ACM, New York, NY, USA, v. 31, n. 3, p. 363–396, 2005. ISSN 0098-3500.

HOPCROFT, J.; KARP, R. An $n^{5/2}$ algorithm for maximum matching in bipartite graphs. *SIAM Journal on Computing*, v. 2, p. 225–231, 1973.

KESLER, M. G.; KESSLER, M. M. Flexible flow. *World Pet.*, v. 19, p. 60, 1958.

KREYSZIG, E. *Advanced Engineering Mathematics*. New York, NY, USA: John Wiley & Sons, Inc., 2000.

LEE, L.-Q.; SIEK, J. G.; LUMSDAINE, A. The generic graph component library. In: *Conference on Object-Oriented*. [s.n.], 1999. p. 399–414. Disponível em: <http://citeseer-.ist.psu.edu/siek00generic% -.html>.

LEITOLD, A.; HANGOS, K. M. Structural solvability analysis of dynamic process models. *Computers and Chemical Engineering*, v. 25, p. 1633–1646, 2001.

LIOEN, W. M.; SWART, J. J. B. de; VEEN, W. A. van der. *PSIDE User's Guide*. [S.l.], dec 1998.

MAJER, C.; MARQUARDT, W.; GILLES, E. D. Reinitialization of dae's after discontinuities. *Computers & Chemical Engineering*, v. 19, n. Supplement 1, European Symposium on Computer Aided Process Engineering 3-5, p. 507–512, June 1995.

MAO, G.; PETZOLD, L. R. Efficient integration over discontinuities for differential-algebraic systems. *Computers & Mathematics with Applications*, v. 43, n. 1-2, p. 65–79, January 2002.

MARQUARDT, W. Trends in computer-aided process modeling. *Computers and Chemical Engineering*, v. 20, n. 6, p. 591–609, 1996.

MATTSSON, S. E.; ANDERSSON, M. Omola – an object oriented modeling language. *Recent Advances in Computer Aided Control Engineering*, v. 9, p. 291–310, 1993.

MATTSSON, S. E.; OLSSON, H.; ELMQVIST, H. Dynamic selection of states in dymola. In: *Modelica Workshop*. [S.l.: s.n.], 2000. p. 61–67.

MEHLHORN, K.; NAHER, S.; UHRIG, C. The LEDA platform of combinatorial and geometric computing. In: *Automata, Languages and Programming*. [s.n.], 1997. p. 7–16. Disponível em: <http://citeseer.csail.mit.edu/article/mehlhorn95leda.html>.

MURATA, V. *Caracterização simbólica de Equações Algébrico-Diferenciais por um Sistema de Álgebra Computacional com Aplicações na Engenharia Química*. Tese (Doutorado) — COPPE/UFRJ, Rio de Janeiro, Brasil, 1996.

MUROTA, K. *Matrices and Matroids for Systems Analysis*. [S.l.]: Springer-Verlag, 2000.

OGUNNAIKE, B. A.; RAY, W. H. *Process Dynamics, Modeling and Control*. [S.l.]: Oxford University Press, 1994.

OH, M.; PANTELIDES, C. C. A modelling and simulation language for combined lumped and distributed parameter systems. *Computers and Chemical Engineering*, v. 20, p. 611–633, 1996.

PANTELIDES, C. C. The consistent initialization of differential-algebraic systems. *SIAM J. Sci. Stat. Comp.*, v. 9, n. 2, p. 213–231, mar. 1988.

PANTELIDES, C. C. Speedup-recent advances in process simulation. *Computers and Chemical Engineering*, v. 12, n. 7, p. 745–755, 1988.

PARK, T.; BARTON, P. I. State event location in differential-algebraic models. *ACM Trans. Model. Comput. Simul.*, v. 6, n. 2, p. 137–165, 1996.

PETZOLD, L. R. *DASSL: Differential Algebraic System Solver*. [S.l.], 1983.

POULSEN, M. Z. *Structural analysis of DAEs*. Tese (Doutorado) — Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2001. Disponível em: <http://www2.imm.dtu.dk/pubdb/p.php?1190>.

RAGAZZINI, J. R.; RANDALL, R. H.; RUSSELL, F. A. Analysis of problems in dynamics by electronic circuits. *Proc. IRE*, v. 35, p. 444–452, 1947.

REISSIG, G.; MARTINSON, W. S.; BARTON, P. I. Differential-algebraic equations of index 1 may have an arbitrarily high structural index. *SIAM J. Sci. Comput.*, v. 21, n. 6, p. 1987–1990 (electronic), 2000.

RICO-RAMIREZ, V. *Representation,analysis And Solution Of Conditional Models In An Equation-Based Environment*. Tese (Doutorado) — Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA, 1998.

RICO-RAMIREZ, V.; WESTERBERG, A. W. Structural analysis of conditional models. *Computers and Chemical Engineering*, v. 26, n. 3, p. 359–373, 2002.

SAIP, H.; LUCCHESI, C. *Matching algorithms for bipartite graph*. Campinas, Brazil, 1993. Disponível em: <http://citeseer.ist.psu.edu/baiersaip93matching.html>.

SECCHI, A. R. *DASSLC*. 2005. Disponível em: <http://www.enq.ufrgs.br/enqlib-/numeric/numeric.html>.

SHAMPINE, L.; THOMPSON, S. Event location for ordinary differential equations. *Comp. & Maths. with Appls.*, v. 39, p. 43–54, 2000.

SIEK, J. G.; LEE, L.-Q.; LUMSDAINE, A. *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley, 2002. Disponível em: <http://www-.awprofessional.com/title/0201729148>.

SOARES, R. P. *Desenvolvimento de um Simulador Genérico de Processos Dinâmicos*. Dissertação (Mestrado) — DEQUI/UFRGS, Porto Alegre, Brasil, fev. 2003.

SOARES, R. P.; SECCHI, A. R. EMSO: A new environment for modelling, simulation, and optimisation. In: *ESCAPE 13th*. [S.l.]: Elsevier Science Publishers, 2003. v. 1, p. 947–952.

SRIDHAR, N.; AGRAWAL, R.; KINZEL, G. L. Algorithms for the structural diagnosis and decomposition of sparse, underconstrained design systems. *Computer-Aided Design*, v. 28, p. 237–249, 1996.

STAUDT, P. B.; SOARES, R. P.; SECCHI, A. R. Simulação dinâmica de colunas de destilação reativa para a predição de comportamento em partidas. In: *XVI COBEQ*. [S.l.: s.n.], 2006.

UNGER, J.; KRONER, A.; MARQUARDT, W. Structural analysis of differential-algebraic equation systems–theory and applications. *Computers & Chemical Engineering*, v. 19, n. 8, p. 867–882, ago. 1995. Disponível em: <http://www.sciencedirect.com-/science/article/B6TFT-3YB50WR-5M/2/15610e9d42a3e15c2c880e7aaee8a7e4>.

UNO, T. A fast algorithm for enumerating bipartite perfect matchings. In: *ISAAC '01: Proceedings of the 12th International Symposium on Algorithms and Computation*. London, UK: Springer-Verlag, 2001. p. 367–379. ISBN 3-540-42985-9.

VIEIRA, R. C. *Métodos diretos para Iniciação de Sistemas Algébrico-Diferenciais*. Dissertação (Mestrado) — COPPE/UFRJ, Rio de Janeiro, Brasil, Nov 1998.

VRTECH. *VRTherm User's Manual*. [S.l.], 2006. Disponível em: <http://www.vrtech-.com.br>.

WESTERBERG, A. W. *Process Engineering: Part II - A Systems View*. 1998. Disponível em: <http://citeseer.ist.psu.edu/248833.html>.

WU, B.; WHITE, R. E. An initialization subroutine for daes solvers: Daeis. *Computers & Chemical Engineering*, v. 25, n. 2-3, p. 301–311, mar. 2001. Disponível em: <http://www.sciencedirect.com/science/article/B6TFT-42RVP6J-8/2-/e4e6ea42de014ea3b5cdf669ea55932a>.

ÅSTRöM, K. J.; ELMQVIST, H.; MATTSON, S. E. Evolution of continuous-time modeling and simulation. In: ZOBEL, R. N.; MÖLLER, D. P. F. (Ed.). *ESM*. [S.l.]: SCS Europe, 1998. p. 9–18. ISBN 1-56555-148-6.