UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

SIMULADOR TEMPORAL DE CIRCUITOS LINEARES

ADOLFO BAUCHSPIESS

Orientador: FRANCISCO ASSIS DE OLIVEIRA NASCIMENTO

Dissertação apresentada ao Departamento de Engenharia Elétrica da Universidade de Brasília, como requisito parcial à obtenção do grau de Mestre em Engenharia Elétrica.

BRASÍLIA, DF - BRASIL
DEZEMBRO DE 1990

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

SIMULADOR TEMPORAL DE CIRCUITOS LINEARES

ADOLFO BAUCHSPIESS

Orientador: FRANCISCO ASSIS DE OLIVEIRA NASCIMENTO

Dissertação apresentada ao Departamento de Engenharia Elétrica da Universidade de Brasília, como requisito parcial à obtenção do grau de Mestre em Engenharia Elétrica.

BRASÍLIA, DF - BRASIL
DEZEMBRO DE 1990

Tese aprovada por:

Francisco Assis de Oliveira Nascimento - D.Sc.
Orientador / ENE-CEPESC

Antonio Carneiro de Mesquita Filho - Dr.d'Etat
COPPE - UFRJ

Evandro Emílio de Souza Lima - Dr. Ing.
ENE - UnB

Luís Geraldo P. Meloni - Dr.
ENE - UnB

José Camargo da Costa - Dr.
ENE - UnB

Vista e permitida a impressão
Brasília, 3 de dezembro de 1990.

Coordenador de Pós-Graduação do departamento de Engenharia Elétrica da Universidade de Brasília.

BAUCHSPIESS, ADOLFO

Simulador Temporal de Circuitos Lineares (Brasília) 1990
XI, 265 pag. (Departamento de Engenharia Elétrica -UnB,
M.Sc. Engenharia Elétrica, 1990)

Tese - Universidade de Brasília, UnB

1. Simulador Temporal de Circuitos Lineares

I. ENE/UnB II. Título (Série)

AGRADECIMENTOS

. Ao Prof. Francisco Assis de Oliveira Nascimento, pela orientação segura, pela prestatividade e entusiasmo em todos os momentos.

. Aos professores do Departamento de Engenharia Elétrica da UnB, e em especial aos da Pós-graduação pelos conhecimentos transmitidos e pelo incentivo constante.

. Aos colegas do curso de mestrado, que enriqueceram este trabalho com discussões e sugestões.

. Ao CDT, pelo apoio que vem prestando ao Grupo de Processamento Digital de Sinais da UnB.

A meus pais, Alfred e Elli, pelo exemplo e dedicação sempre presentes em minha vida acadêmica.

A minha querida noiva, Enir, pela paciência e compreensão, renovando minhas forças durante a elaboração deste trabalho.

RESUMO

As ferramentas computacionais para simulação são consideradas atualmente como indispensáveis em qualquer projeto profissional de circuitos. A categoria de simuladores no domínio do tempo (simuladores elétricos) permite observar o comportamento temporal das tensões e correntes nos pontos de acesso de um circuito submetido a sinais de entrada. A simulação, juntamente com técnicas de otimização, fornece os subsídios para o projeto automatizado de circuitos assistido por computador.

Este trabalho se propõe a apresentar um Simulador Temporal de Circuitos Lineares, em que as fontes de sinal são descritas pelas suas amostras contidas em um arquivo de computador digital. Os sinais podem ser as amostras de um conversor analógico digital, sinais de teste padrão (senóides, ondas quadradas), sinais estocásticos, sinais gerados por algum codificador, enfim, a simulação pode aceitar qualquer sinal (inteiro ou real) discretizado em intervalos de tempo uniformes.

No processo de representação computacional é utilizada a formulação nodal modificada, possibilitando uma representação flexível e eficiente de circuitos. A resolução do sistema nodal modificado é obtida via algoritmo de bifatoração da matriz de circuito e técnicas de tratamento da esparsidade intrínseca a essa matriz. O processo de iteração temporal é implementado através de diversos modelos discretizados de elementos de circuito.

ABSTRACT

Computational tools for simulation are today accepted as indispensable in any professional circuit design. Time-domain simulators (electrical simulators) give insight on the behavior of voltages and currents in the access points of one circuit, subjected to input signals. The simulation, together with optimization techniques, provides the background to computer-aided automated circuit design.

The purpose of this work is to present a Time-domain Simulator for Linear Circuits, where the signal sources are described by their samples in digital computer files. The signals may be the samples of an analog to digital converter output, standard test signals (sine waves, square waves), stochastic signals, a coded signal, that is, the simulation can accept any signal (integer or real) discretized in uniform time steps.

The modified nodal approach is used in the computational description, providing flexible and efficient circuit representation. The solution of the modified nodal system is obtained by bifactorization of the circuit matrix, and by techniques that exploit the intrinsic sparsity of this matrix. The time iteration process is implemented by several discretized circuit device models.

ÍNDICE

1 - INTRODUÇÃO 1

1.1 - Revisão Bibliográfica 4

1.2 - Resumo das Pesquisas 7

1.3 - Síntese do Trabalho 10

2 - FORMULAÇÃO DAS EQUAÇÕES DE CIRCUITO 12

2.1 - Formulação Nodal Modificada 15

2.2 - Formulação Nodal Modificada de Alguns Componentes Lineares 18

2.3 - Exemplos 29

3 - DISCRETIZAÇÃO DOS MODELOS ASSOCIADOS À FORMULAÇÃO NODAL MODIFICADA 34

3.1 - Definição de Modelo Discreto Associado 36

3.2 - Algoritmo de Euler Regressivo 37

3.2.1 - Modelo Discreto Associado ao Capacitor para o Algoritmo de Euler Regressivo 37

3.2.2 - Relações Constitutivas Associadas ao Algoritmo de Euler Regressivo para o Capacitor 40

3.2.3 - Modelo Discreto Associado ao Indutor para o Algoritmo de Euler Regressivo 42

3.2.4 - Relações Constitutivas Associadas ao Algoritmo de Euler Regressivo para o Indutor 44

3.3 - Algoritmo Trapezoidal 45

3.3.1 - Modelo Discreto Associado ao Capacitor para o Algoritmo Trapezoidal 46

3.3.2 - Relações Constitutivas Associadas ao Algoritmo Trapezoidal para o Capacitor 47

3.3.3 - Modelo Discreto Associado ao Indutor para o Algoritmo Trapezoidal 48

3.3.4 - Relações Constitutivas Associadas ao Algoritmo Trapezoidal para o Indutor 49
3.4 - Algoritmos de Gear de Segunda a Quarta Ordem 51
3.4.1 - Modelos Discretos Associados ao Capacitor para os Algoritmos de Gear até Quarta Ordem 52
3.4.2 - Estampas Associadas aos Algoritmos de Gear de Segunda a Quarta Ordem para o Capacitor . 53
3.4.3 - Modelos Discretos Associados ao Indutor para os Algoritmos de Gear até Quarta Ordem 60
3.4.4 - Estampas Associadas aos Algoritmos de Gear de Segunda a Quarta Ordem para o Indutor ... 61
3.5 - Exemplos 65
3.6 - Considerações Computacionais 70
3.7 - Generalização da Formulação Nodal Modificada para Modelos Discretos Associados 74

4 - SOLUÇÃO DO SISTEMA DE EQUAÇÕES NODAIS MODIFICADAS .. 79
4.1 - Técnicas de Matrizes Esparsas para Solução do Sistema de Equações Nodais Modificadas .. 81
4.1.1 - Estruturas de Dados Dinâmicas para o Sistema de Equações Nodais Modificadas 83
4.2 - Esparsidade e Solução de Sistemas de Equações Através de Métodos de Fatoração ... 89
4.2.1 - Solução de um Sistema de Equações pelo Método da Bifatoração 91
4.2.2 - Comparação entre os Métodos de Triangularização e de Bifatoração 97
4.2.3 - Estruturas de Dados Dinâmicas para as Matrizes Fator do Método da Bifatoração 99
4.2.4 - Reordenação de Linhas Durante o Processo de Bifatoração 103

4.3 - Efeito da Ordenação do Sistema de Equações sobre a esparsidade das Matrizes Fator 107
4.3.1 - Esquema de Ordenação Proposto 112

5 - DESCRIÇÃO DOS ALGORITMOS IMPLEMENTADOS NO SIMULADOR TEMPORAL DE CIRCUITOS LINEARES 116
5.1 - Descrição Geral do Simulador Temporal de Circuitos 118
5.2 - Algoritmo de Leitura de Circuito 121
5.2.1 - Estruturas de Dados para Representação Interna do Circuito 127
5.2.2 - Algoritmo de Inserção de Componente em Lista 135
5.3 - Algoritmo de Expansão de Componente de Biblioteca 137
5.3.1 - Algoritmo para Salvar a Expansão da Biblioteca 143
5.4 - Algoritmo para Construção do Sistema de Equações Nodais Modificadas 145
5.4.1 - Determinação das Variáveis Nodais Modificadas 145
5.4.2 - Construção Direta das Matrizes para o Sistema de Equações Nodais Modificadas 148
5.5 - Algoritmo de Leitura dos Sinais de Entrada . 149
5.6 - Algoritmo de Escrita das Variáveis de Saída 151

6 - RESULTADOS DE SIMULAÇÕES 153
6.1 - Exemplo 1 - Circuito VRC 155
6.2 - Exemplo 2 - Circuito VRLC 163
6.3 - Exemplo 3 - Linha de Transmissão 168
6.4 - Exemplo 4 - Circuito VR3L..... 173
6.5 - Exemplo 5 - Circuito JC2RL..... 175
6.6 - Exemplo 6 - Filtro Rejeita-Faixa de Segunda Ordem 175
6.7 - Exemplo 7 - Amplificador Diferencial 180
6.8 - Exemplo 8 - Amplificador com Realimentação Série de Tensão 185
6.9 - Exemplo 9 - Filtro Chebyshev de Sétima Ordem 190
6.10 - Exemplo 10 - Conexão em Cascata de Três Filtros Chebyshev de Sétima Ordem 194
6.11 - Exemplo 11 - Simulação Simultânea de Quatro

Filtros Chebyshev de Sétima Ordem 199
6.12 - Comparação dos Resultados Obtidos com o SITECI e com o PSPICE 202

7 - CONCLUSÕES 206

BIBLIOGRAFIA 212

Apêndice A
Soluções Numéricas de Problemas de Valor Inicial 216
A.1 - Solução Numérica por Aproximações Polinomiais 218
A.2 - Consistência 221
A.3 - Famílias de Algoritmos de Integração Numérica por Aproximações Polinomiais..... 225
A.3.1 - Algoritmos de Adams-Bashforth 225
A.3.2 - Algoritmos de Adams-Moulton 226
A.3.3 - Algoritmos de Gear 228
A.4 - Considerações de Erro 230
A.4.1 - Erro de Truncamento dos Algoritmos de Integração Multipassos 232
A.5 - Estabilidade dos Algoritmos de Integração Multipassos 233
A.6 - Convergência 237

Apêndice B - Exemplos de Soluções de Sistemas
B.1 - Exemplo de Solução de um Sistema de Quarta Ordem 240
B.2 - Exemplo do Efeito da Ordenação de Linhas no Esforço Computacional da Solução de Sistemas Esparsos 241
B.3 - Matrizes Fator da Matriz Tridiagonal 244
B.4 - Bifatoração de um Sistema de Quarta Ordem .. 245

1 - INTRODUÇÃO

O desenvolvimento acelerado da eletrônica nos últimos anos tem exigido constante atualização das técnicas de projeto, adequadas à complexidade dos circuitos que a tecnologia permite produzir. Antes da disponibilidade dos computadores o projeto de circuitos eletrônicos costumava ser feito por um engenheiro que iniciava com papel e lápis, utilizando tabelas, folhas de dados, fórmulas e diagramas. O projeto era baseado em grande parte na sua intuição, experiência anterior e capacidade de fazer aproximações razoáveis. Depois do projeto preliminar passava-se à fase de protótipos, onde os resultados do projeto preliminar eram confirmados, e eventualmente melhorados, pelo ajuste de componentes em um processo empírico.

Com a evolução da tecnologia estas metodologias manuais de projeto se tornaram obsoletas. Grandes circuitos discretos e os circuitos integrados, então recém desenvolvidos,

passaram a exigir, em função da complexidade, formas automatizadas ou semi-automatizadas de projeto. A utilização de papel e lápis não era mais adequada considerando-se a precisão e o tempo necessários para completar o projeto.

Por outro lado, o custo de produção de uma máscara para circuito integrado é muito alto, não dando margem a "tentativas e erros". Além das considerações de custo, um estudo de tolerância ou de pior caso sem o auxílio de uma ferramenta computacional seria extremamente cansativo.

É neste ambiente que o computador surge como uma importante ferramenta de projeto. Assim sendo, antes de se implementar o protótipo do circuito, este pode ser simulado em computador de forma a se obter um projeto otimizado. Uma ferramenta desta natureza é o primeiro passo em direção ao projeto automático de circuitos. O outro aspecto complementar é uma eficiente técnica de otimização [1]. Atualmente os programas de simulação são considerados como indispensáveis em qualquer projeto profissional de circuitos.

Quase todos os problemas de análise de circuito são resolvidos em dois passos. O primeiro considera a formulação das equações de equilíbrio do circuito em uma forma apropriada, utilizando as duas leis de Kirchhoff e as características dos componentes. O segundo passo considera a solução destas equações, utilizando técnicas analíticas ou numéricas. Antes da disponibilidade dos computadores, as equações eram resolvidas analiticamente. Esta abordagem impunha severas restrições quanto ao tamanho e ao tipo de circuitos que podiam ser analisados em tempo razoável. Grandes circuitos lineares (contendo, por exemplo, mais de 50 elementos) ou mesmo pequenos circuitos não lineares raramente

tinham análise exata. Como opção os engenheiros baseavam-se em sua intuição e métodos "manuais" para obter uma análise aproximada de tais circuitos. Invariavelmente a análise final era obtida através da montagem de protótipo do circuito e da medida das variáveis de interesse.

Um outro aspecto pelo qual protótipos com componentes discretos são inadequados para o caso de circuitos integrados é que os efeitos parasitas de acoplamento entre os componentes internos não pode ser representado adequadamente com componentes discretos. Em circuitos integrados também não é possível realizar medidas de tensões ou correntes de nós internos, tornando imprescindível o uso de uma ferramenta computacional.

O computador como ferramenta de auxílio ao projeto permitiu grande aumento da complexidade dos circuitos que poderiam ser implementados, em forma integrada ou não. Atualmente existem diversas ferramentas de simulação que permitem auxiliar diferentes etapas do projeto. Particularmente, os simuladores de circuito permitem ao projetista observar o comportamento de um circuito sem a prévia construção de protótipos. Para uma dada entrada, as formas de onda e espectro de freqüência dos sinais de tensão e corrente em qualquer ponto do circuito também são passíveis de serem observadas.

As equações que descrevem o circuito podem ser resolvidas analiticamente ou utilizando-se técnicas numéricas. Neste trabalho serão abordadas com maior ênfase as soluções numéricas, uma vez que a concepção e implementação de um simulador temporal de circuitos lineares em computador constitui-se no objetivo principal.

1.1 - Revisão Bibliográfica

Alguns trabalhos desenvolvidos na área de simulação foram tomados como referência, entre os quais o PASOR (Programa de Análise, Cálculo de Sensibilidade e Otimização de Redes), desenvolvido na COPPE em 1985 [29]. Esta ferramenta de projeto e trabalho utiliza a formulação nodal modificada na implementação das equações de equilíbrio do circuito. Como no PASOR a simulação é feita no domínio da freqüência é necessário construir e resolver o sistema de equações para cada valor individual de freqüência. Os conceitos e modelos utilizados no PASOR para construção da matriz nodal modificada foram adaptados no presente trabalho para a simulação temporal.

O simulador SPICE2 da Universidade da Califórnia, Berkeley, desenvolvido em meados dos anos 70, tornou-se um padrão aceito mundialmente para a simulação de circuitos analógicos. A versão para computadores pessoais padrão IBM da MicroSim Corporation (PSPICE2) foi uma referência constante durante as pesquisas para elaboração deste trabalho [5].

A aplicação do PSPICE2 para processamento de sinais apresenta no entanto algumas limitações. Os sinais de entrada descritos por amostras digitalizadas contidas em arquivo de computador não podem ser utilizadas diretamente para obter uma simulação. O que mais aproximadamente poderia ser feito para utilizar o PSPICE neste caso seria descrever os sinais como uma fonte do tipo PWL (Piecewise Linear Waveform), ou seja, linear por partes. Este processo torna-se extremamente lento quando temos um sinal de voz com dezenas de milhares de

amostras a serem simuladas. Um outro aspecto é que o PSPICE utiliza algoritmos que adaptam o passo de integração. Nos pontos de maior derivada do sinal de entrada são utilizadas mais iterações. Isto não seria possível para um sinal que só é definido em alguns instantes discretos.

Ainda outra limitação do PSPICE é o fato de as formas de onda obtidas pela simulação não terem a opção de serem armazanadas em arquivo de sinal. Esta opção permitiria utilizá-las como sinais de entrada para a simulação de um outro estágio do circuito. Com esta opção poder-se-ia então simular um circuito grande por partes.

Quanto à formulação das equações de circuito a literatura acessível sobre o assunto consagra o método nodal [1, 6]. Este método foi largamente utilizado na formulação das equações de circuito em ferramentas de análise e projeto de circuitos auxiliado por computador. Diversos programas (CANCER [21], ECAP [24], BIAS-3 [25]) foram implementados utilizando a formulação nodal básica.

A formulação nodal possui no entanto, em sua forma básica, algumas limitações. Em 1975 foi publicado um artigo "The Modified Nodal Approach to Network Analysis" (Chung-Wen Ho et allii, 1975 [16]) que propôs modificações no método nodal básico que permitiam superar todas as suas limitações. O que os autores deste artigo propuseram foi um método híbrido que permitia tratar de forma adequada as limitações da formulação nodal básica em processar fontes de tensão e outros dispositivos não representáveis sob a forma de admitância. O novo método também tinha vantagens computacionais sobre outros métodos híbridos que naquela época dispunham de relativa popularidade, como por exemplo a

formulação TABLEAU [23].

Quando se utilizam técnicas de matrizes esparsas para a solução do sistema de equações nodais modificadas existe a possibilidade de um elemento nulo aparecer na diagonal. Um cuidado especial deve ser tomado para que este elemento não seja utilizado como pivô. O artigo "Avoiding Zero pivots in the Modified Nodal Approach" (Hajj et allii, 1981 [17]) apresenta esta situação e propõe um esquema de reordenação das equações nodais modificadas no sentido de evitar que o processo de solução falhe por uma ordem inadequada da eliminação das variáveis.

A redução do tempo de processamento é uma das características mais desejadas em simulação de circuitos. O ponto mais crítico no que concerne à simulação temporal é a solução do sistema de equações proveniente da formulação das equações de equilíbrio do circuito. A solução dos sistemas de equações lineares pelo método da bifatoração foi introduzido por Zollenkopf [15] em 1975 como alternativa para preservar a esparsidade original do sistema durante a sua solução. Este método é particularmente eficiente quando aplicado a sistemas simétricos ou com simetria estrutural (elementos não nulos, mesmo que de valores distintos, em posições simétricas em relação à diagonal principal). A preservação da esparsidade contribui de maneira decisiva no tempo de processamento da simulação temporal.

1.2 - Resumo das Pesquisas

O processo de pesquisa na literatura acessível iniciou-se com uma busca das formulações de circuito mais adaptáveis para implementação em um computador digital. Buscou-se também as soluções dos sistemas através de algoritmos eficientes computacionalmente.

Buscando a flexibilidade de tratar processos digitalizados, as interfaces de entrada/saída operam com sinais geralmente descritos como um conjunto de amostras contidas em arquivos. Desta forma, os sinais são conhecidos em apenas alguns instantes de tempo discretos e uniformemente distribuídos. Esta característica do problema estabelece restrições quanto aos tipos de algoritmo que podem ser utilizados eficientemente. Os algoritmos que utilizam os valores dos sinais em pontos intermediários das amostras para a simulação foram preteridos, pois necessitam que o sinal original seja interpolado, como é o caso dos algoritmos de Runge-Kutta[1].

Um estudo sobre topologia de circuitos mostrou-se útil para dar uma boa compreensão das leis de Kirchhoff generalizadas (equações em termos de matrizes de incidência, de corte e de laços) [1]. As equações topológicas juntamente com as relações constitutivas são suficientes para descrever os circuitos lineares. Existem porém, diversas alternativas

---

[1] O PSPICE, por exemplo, utiliza sinais de entrada definidos para todos os instantes de tempo, permitindo assim uma adaptação dinâmica do passo de solução numérica [5].

para a formulação destas equações [1, 6, 7, 13, 14, 16, 17].

A formulação das equações de circuito pelas matrizes topológicas mostrou-se bastante ineficiente, já que as equações de equilíbrio podem ser construídas diretamente, sem necessidade das matrizes topológicas [1, 16, 17].

Quando as equações são escritas apenas em função das tensões nodais obtemos a formulação por matriz de admitância nodal. No entanto, esta formulação não foi adotada por apresentar algumas restrições computacionais, como por exemplo, descrição de fontes de tensão (independentes ou controladas) e fontes controladas por corrente, que não podem ser representadas como admitância.

A formulação nodal modificada foi por fim adotada para a implementação das equações de equilíbrio por ser mais genérica que as outras estudadas e permitir a representação dos componentes mais conhecidos. Esta formulação fornece um sistema de equações diferenciais ordinárias de primeira ordem, onde as equações podem ser resolvidas no domínio do tempo através de algoritmos de integração.

Diferentes métodos de integração numérica foram avaliados quanto a figuras de mérito como por exemplo: esforço computacional, erro acumulativo e estabilidade. Deste estudo surgiu a proposta de uma metodologia para a implementação dos algoritmos de integração numérica destinados à solução do sistema de equações geradas pela formulação nodal modificada. Cinco métodos de integração (Euler regressivo, Trapezoidal e Gear de segunda, terceira e quarta ordem [1, 6, 10, 12]) foram implementados, pelas suas características de estabilidade, complexidade de

implementação e esforço computacional.

A construção da Matriz Nodal Modificada fornece um sistema de equações lineares, que devido a natureza dos circuitos eletrônicos é geralmente bastante esparsa. A solução deste sistema de equações lineares esparso é o ponto mais crítico, em termos computacionais. Estudou-se então, diversos métodos de solução. Numa fase inicial do trabalho foi implementada a solução através da inversão pura e simples da matriz nodal modificada, com o propósito de verificar o funcionamento dos algoritmos de iteração temporal. Posteriormente, levando-se em conta as matrizes geradas pelos circuitos elétricos escolheu-se o método da bifatoração [4, 15] para a solução do sistema de equaçãoes.

A grande esparsidade das matrizes envolvidas sugeriu um levantamento das técnicas de matrizes esparsas disponíveis na literatura [2,3,4]. Da análise dos métodos de armazenamento das matrizes esparsas chegou-se a uma forma eficiente de armazenamento que reduz o esforço computacional para o método da bifatoração.

Alguns circuitos utilizados comercialmente foram simulados e os resultados obtidos comparados com o PSPICE visando avaliar a precisão dos algoritmos utilizados. A comparação não pode ser feita diretamente pois o PSPICE utiliza o modelo completo para o transistor e o algoritmo que foi implementado utiliza o modelo para pequenos sinais. Desta forma os modelos utilizados no PSPICE também foram linearizados para permitir uma melhor avaliação.

Encerrando este primeiro capítulo, é apresentada logo a seguir uma pequena síntese do trabalho.

1.3 - Síntese do Trabalho

Este trabalho se propõe a apresentar um Simulador Temporal de Circuitos Lineares, em que as fontes de sinal são descritas pelas suas amostras contidas em arquivo. Os sinais podem ser as amostras digitalizadas de um conversor analógico digital, sinais de teste padrão (senóides, ondas quadradas), sinais estocásticos, sinais gerados por algum codificador, enfim, a simulação pode aceitar qualquer sinal (inteiro ou real) discretizado em intervalos de tempo uniformes.

A seguir, no Capítulo 2, é apresentada a formulação das equações de circuito para o método nodal modificado. As vantagens deste método em relação a outros é brevemente discutida.

Os capacitores e indutores são os componentes que inserem a característica dinâmica nas equações de circuito. Assim no Capítulo 3 aplicam-se métodos numéricos para obtenção de modelos discretos associados a capacitores e indutores. Para cada método de integração numérica pode-se derivar um modelo discreto associado. São discutidos os modelos discretos associados aos métodos de integração implementados no Simulador Temporal de Circuitos. Também são mostradas as leis constitutivas [16] que permitem construir diretamente, por inspeção da topologia do circuito, as matrizes do sistema de equações.

O Capítulo 4 apresenta algoritmos computacionalmente eficientes para a solução do sistema de equações obtidas no Capítulo 3. É apresentado o método da bifatoração, que aliado

a estruturas de dados apropriadas, permite explorar a característica esparsa do sistema de equações associado a circuitos elétricos.

O capítulo 5 apresenta uma descrição a nível de algoritmos do simulador de circuitos implementados.

No capítulo 6 são apresentados resultados da simulação de diversos circuitos e comparação com outros programas, demonstrando a eficiência e validade dos algoritmos de simulação propostos.

Finalmente, o Capítulo 7 apresenta as principais conclusões deste trabalho e sugestões para continuidade das pesquisas. As referências bibliográficas vêm a seguir. Os apêndices apresentam tópicos que ilustram melhor os temas discutidos no corpo da tese.

2 - Formulação das Equações de Circuito

O objetivo da teoria de circuitos é o de descrever fenômenos que ocorrem em uma determinada parte do mundo físico através do estabelecimento de um modelo matemático. O propósito de um modelo é prever o desenvolvimento do fenômeno físico. No caso dos circuitos elétricos, o modelo é bem estabelecido no que concerne aos circuitos lineares.

Por circuito elétrico entende-se uma interconexão de componentes e dispositivos elétricos formando uma estrutura com pontos de acesso onde os sinais podem ser observados.

Mais particularmente, no caso dos circuitos lineares, supõe-se que os elementos ou dispositivos constituintes são representados por modelos ou elementos hipotéticos cujas relações tensão-corrente são equações lineares que podem ser:

i) algébricas - domínio de freqüências.

ii) diferenciais ordinárias - domínio do tempo, parâmetros concentrados.

iii) diferenciais parciais - domínio do tempo, parâmetros distribuidos.

A representação no domínio do tempo a parâmetros concentrados constitui o principal motivo de pesquisa deste trabalho, onde se procurou desenvolver uma ferramenta de projeto semi-automático de circuitos. As formulações no domínio de freqüências e por parâmetros distribuídos podem ser encontradas nas referências bibliográficas [1, 6, 7, 18, 19].

As propriedades de um circuito podem ser atribuídas a dois fatores principais:

i) à maneira como são interligados os componentes de onde derivam propriedades topológicas do circuito e

ii) à natureza dos componentes, de onde derivam as propriedades de processamento de sinal do circuito (a topologia pode impor restrições adicionais a estas propriedades do processamento de sinais, mas em princípio o fator preponderante é a natureza do componente) [1].

Os sinais, ou variáveis em termos das quais o comportamento de um circuito é descrito, são a tensão e a corrente. Estas variáveis são funções do tempo e como tal são representadas por v(t) e i(t). As leis fundamentais sobre as quais se baseia a teoria de circuitos exprimem relações entre tensões e correntes em vários pontos do circuito.

Existem diferentes métodos para construir o sistema de

incógnitas, i.e., b correntes de ramos, b tensões de ramos, e $n - 1$ tensões com referência à terra [1, pp 671-674]. A matriz gerada é desta forma muito maior que a fornecida por qualquer outra formulação, e não há em geral interesse em todas estas incógnitas como variáveis de saída.

Tendo como objetivo remover as limitações da formulação nodal, mantendo ainda simplicidade e eficiência computacional foi proposto e publicado em 1975 o artigo "The Modified Nodal Approach to Network Analysis" (Ho, Ruehli & Brennan 1975) [16]. Esta formulação nodal *modificada* mostrou-se então, ser muito eficiente para formulação das equações de circuito e sua respectiva implementação em um computador digital [17]. Esta formulação é atualmente utilizada nos algoritmos do programa SPICE2 [17], que se tornou um padrão em termos de simulação analógica.

A seguir apresenta-se a formulação Nodal Modificada. Este método de formulação é genérico e aplica-se também ao domínio de freqüências. Entretanto, devido ao propósito de implementar um simulador no domínio do tempo, a abordagem será feita neste domínio.

2.1 - Formulação Nodal Modificada

A formulação das equações de circuito utilizando a abordagem nodal modificada toma inicialmente as tensões dos nós em relação a terra como variáveis, de forma semelhante ao método base da matriz nodal. A lei das correntes de Kirchhoff é aplicada a cada nó, com exceção do nó terra, de tal forma que a soma das correntes que deixam qualquer nó é zero. Para o caso simples de um circuito contendo apenas condutâncias lineares e fontes de corrente independentes, o sistema de equações nodais modificadas se reduz ao sistema de equações

nodais tradicional,

$$Y\,V = J \qquad (2.1)$$

Onde Y representa a matriz de admitâncias nodais, V as tensões nodais, e J o vetor de fontes de corrente.

Para os circuitos que contêm fontes de tensão e outros componentes cujas correntes são as variáveis de controle, as variáveis nodais são acrescidas destas correntes e as relações constitutivas dos ramos são incluídas no conjunto das equações nodais. Estas correntes de ramo tornam-se desta forma disponíveis como variáveis de saída.

O sistema de equações nodais modificadas pode ser escrito na forma geral:

$$\begin{bmatrix} Yr & \vdots & B \\ \cdots & \vdots & \cdots \\ C & \vdots & D \end{bmatrix} \begin{bmatrix} V \\ \cdots \\ I \end{bmatrix} = \begin{bmatrix} J \\ \cdots \\ F \end{bmatrix} \qquad (2.2)$$

Onde Yr é uma forma reduzida da matriz nodal que exclui as contribuições devido às fontes de tensão e componentes não representáveis sob forma de admitância. A matriz B é uma matriz topológica que considera o efeito das correntes incluídas como variáveis nas equações nodais. As matrizes C e D contêm as relações constitutivas dos ramos cujas correntes são variáveis. Os vetores J e F são as excitações.

A formulação nodal modificada permite a construção direta (por inspeção da topologia do circuito) do sistema de equações mostrado em (2.2). Para tal, cada componente de

circuito possui uma *estampa*[2] que descreve a sua contribuição na matriz nodal modificada. Com a utilização destas estampas todo o sistema de equações pode ser construído inserindo as contribuições individuais de cada um dos componentes do circuito em questão. Alguns componentes têm representação tanto por admitância como por impedância e assim admitem duas formas alternativas de estampas. Caso a corrente no componente não seja solicitada como variável de saída é mais vantajoso utilizar a representação por admitância, pois assim é reduzida a ordem do sistema. A seguir são desenvolvidas as estampas de alguns componentes básicos.

---

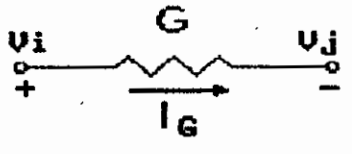
[2] "Estampa" é uma forma gráfica utilizada neste trabalho para apresentar a contribuição de um elemento à matriz nodal madificada, quando a construção desta é feita por inspeção da topologia do circuito. Na estampa estão inseridas as leis constitutivas do ramo.

2.2 - Formulação Nodal Modificada de alguns Componentes Lineares

Nesta seção desenvolvem-se as estampas que permitem construir por inspeção o sistema de equações nodais modificadas de um circuito linear. Os principais componentes lineares são descritos sucintamente. A equação constitutiva de cada componente linear é utilizada na obtenção da respectiva estampa.

Resistor Linear:

Um resistor linear conectado aos nós i e j apresenta corrente I_G proporcional à diferença de tensão em seus terminais $(V_i - V_j)$. A constante de proporcionalidade é conhecida por condutância G (medida em siemens), e é o inverso da resistência (medida em ohms, Ω).



$$I_G = G(V_i - V_j)$$

Fig. 2.1 - Resistor Linear

A contribuição do resistor à matriz nodal modificada pode ser incluída de duas maneiras. Caso a corrente no resistor não seja solicitada como variável de saída então tem-se a corrente do resistor deixando o nó i e entrando no nó j. Para o nó i a contribuição é $G(V_i - V_j)$ e para o nó j a contribuição é $-G(V_i - V_j)$, conforme visto na estampa seguinte:

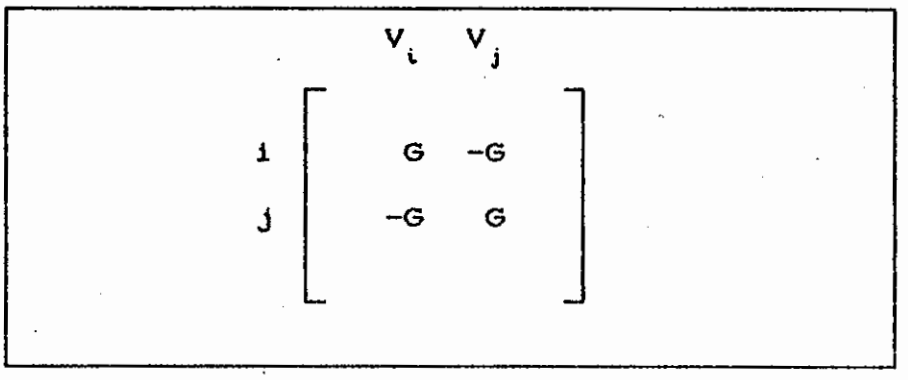


Fig. 2.2 - Estampa do Resistor Linear na Parte Nodal

Caso a corrente no resistor I_G seja solicitada como variável de saída então a incluímos no conjunto de variáveis. A matriz nodal modificada passa a ser acrescida de uma linha e uma coluna correspondentes à corrente I_G. A contribuição do resistor é então $+I_G$ à corrente do nó i e $-I_G$ à corrente do nó j. A relação constitutiva do resistor $G(V_i - V_j) - I_G = 0$ é acrescentada à matriz nodal modificada como equação constitutiva do resistor (matrizes C e D da equação 2.2), conforme visto na estampa da Fig. 2.3 abaixo.

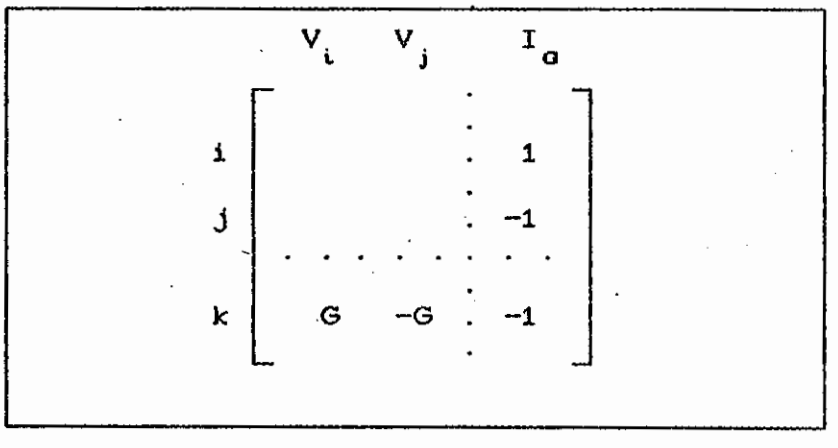
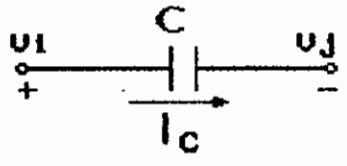


Fig. 2.3 - Estampa do Resistor na Parte Híbrida

Capacitor Linear:

Um capacitor linear conectado aos nós i e j apresenta corrente I_c proporcional à derivada em relação ao tempo da diferença de tensão em seus terminais $(V_i - V_j)$. A constante de proporcionalidade é conhecida por capacitância C (medida em farads).



$$I_c = C \frac{d}{dt}(U_i - U_j)$$

Fig. 2.4 - Capacitor Linear

De forma semelhante ao resistor linear têm-se duas formas de representar a contribuição do capacitor à matriz nodal modificada, caso a corrente seja ou não solicitada como variável de saída. Se a corrente não for solicitada tem-se uma corrente $C \frac{d}{dt}(V_i - V_j)$ deixando o nó i e $-C \frac{d}{dt}(V_i - V_j)$ entrando no nó j, conforme visto na estampa abaixo:

$$\begin{array}{c} \\ i \\ j \end{array} \begin{array}{c} \begin{array}{cc} V_i & V_j \end{array} \\ \begin{bmatrix} C\frac{d}{dt} & -C\frac{d}{dt} \\ -C\frac{d}{dt} & C\frac{d}{dt} \end{bmatrix} \end{array}$$

Fig. 2.5 - Estampa do Capacitor Linear na Parte Nodal

Caso a corrente no capacitor seja solicitada como variável de saída aplica-se a seguinte estampa:

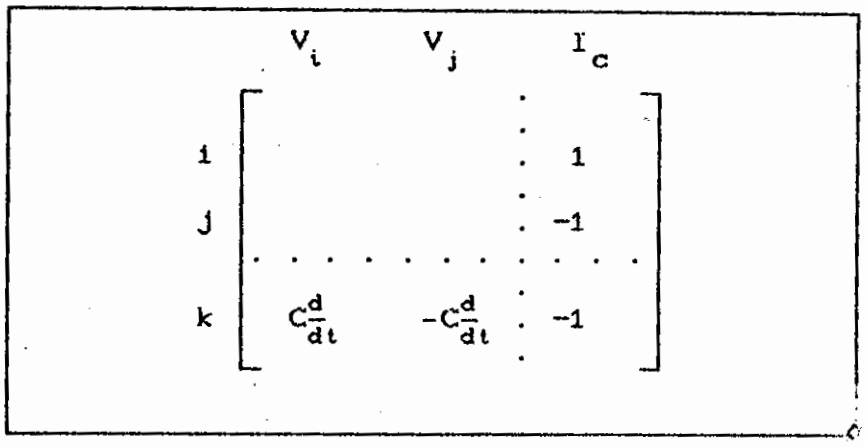
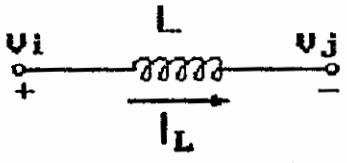


Fig. 2.6 - Estampa do Capacitor na Parte Híbrida

Indutor Linear:

Um indutor linear conectado aos nós i e j apresenta diferença de tensão em seus terminais $(V_i - V_j)$ proporcional à derivada da corrente em relação ao tempo $\frac{d}{dt} I_L$. A constante de proporcionalidade é conhecida por indutância L (medida em henries).



$$(V_i - V_j) = L \frac{d}{dt} I_L$$

Fig. 2.7 - Indutor Linear

Como o indutor é um componente cuja corrente é proporcional à integral da tensão, não é conveniente utilizar a representação por admitância. Caso isto fosse feito apareceria uma contribuição $\int (V_i - V_j)$ na matriz, gerando após derivação temporal de todas as equações um sistema de equações diferenciais ordinárias de segunda ordem. Este sistema teria de ser decomposto em equações diferenciais

ordinárias de primeira ordem para ser solucionado pelos métodos numéricos convencionais. Em suma, a representação do indutor por admitância torna mais complexo o sistema de equações, tendo em vista que a representação do indutor na parte híbrida da matriz nodal modificada produz um sistema de equações ordinárias de primeira ordem. Assim, escrevendo as leis constitutivas do ramo $(V_i - V_j) - L \frac{d}{dt} I_L = 0$, o mesmo pode ser colocado na parte híbrida sem perda de generalidade.

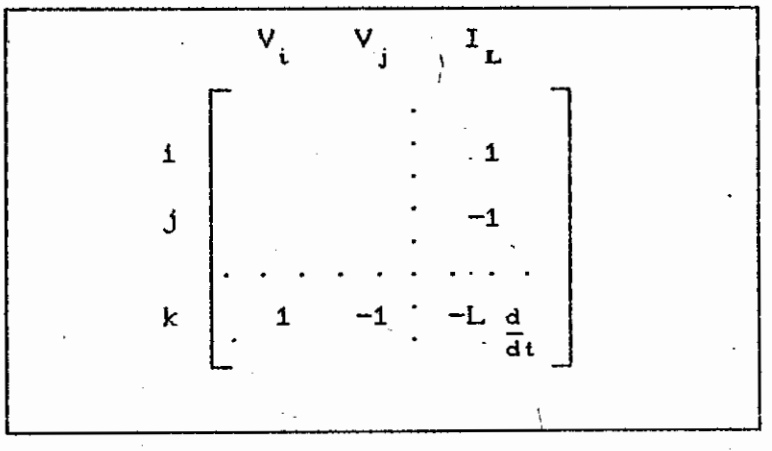
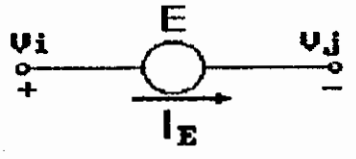


Fig. 2.8 - Estampa do Indutor Linear

Fonte Independente de Tensão:

A fonte de tensão deve ser representada na parte híbrida da matriz nodal modificada pois a sua corrente I_E geralmente não é conhecida. A equação $(V_i - V_j) = E$ é representada na matriz nodal modificada conforme a estampa abaixo:



$$(V_i - V_j) = E$$

Fig. 2.9 - Fonte Independente de Tensão

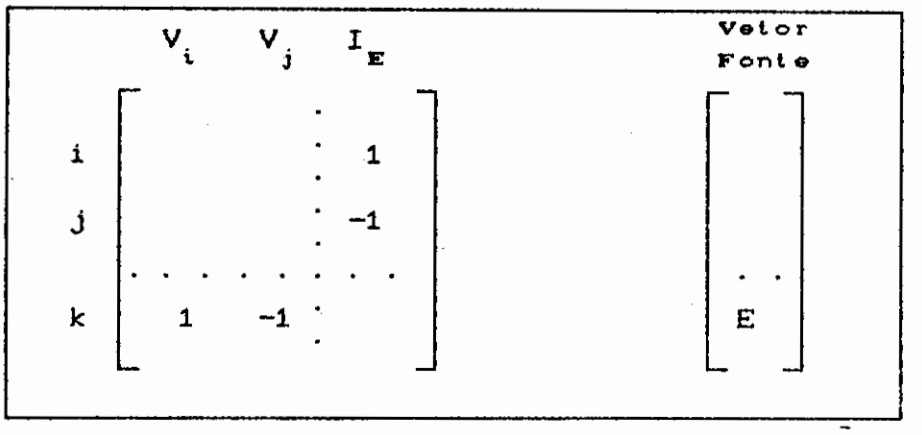


Fig. 2.10 - Estampa da Fonte Independente de Tensão

Fonte Independente de Corrente:

As fontes independentes de corrente são representadas sem que a corrente esteja incluída na saída (o seu valor já é conhecido à priori). A tensão sobre a fonte de corrente pode ser determinada a partir das tensões dos nós aos quais a mesma está ligada.

Fig. 2.11 - Fonte Independente de Corrente

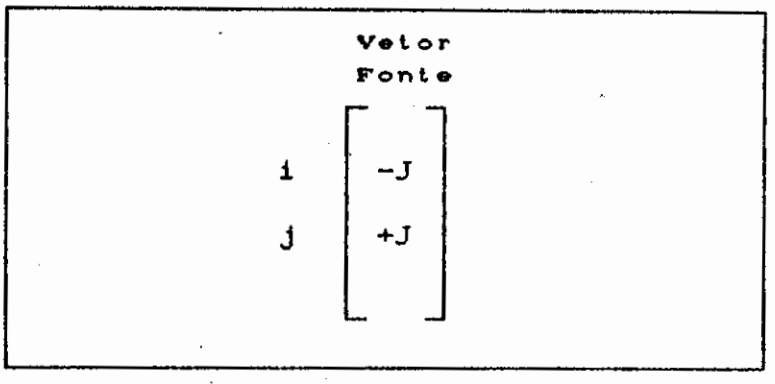
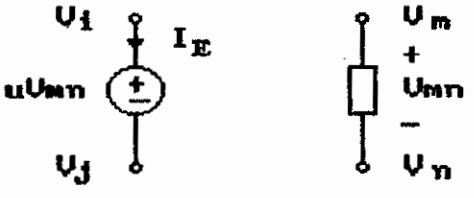


Fig. 2.12 - Estampa da Fonte Independente de Corrente

Fonte de Tensão Controlada por Tensão:

Uma fonte de tensão controlada por tensão ligada aos nós i e j tem a sua tensão proporcional à diferença de tensão entre os nós m e n. A constante de proporcionalidade "μ" é definida como o ganho de tensão da fonte controlada.



$$(V_i - V_j) = u\,(V_m - V_n)$$

Fig. 2.13 - Fonte de Tensão Controlada por Tensão

Pelo fato do parâmetro "ganho de tensão" possuir natureza adimensional, a fonte de tensão controlada por tensão não pode ser representada como admitância. Assim sendo, é necessário construir as relações constitutivas do ramo $V_i - V_j = \mu(V_m - V_n)$ na parte híbrida da Matriz Nodal Modificada (linha k da estampa). A corrente I_E que percorre esta fonte é acrescida ao nó i e subtraida do nó j.

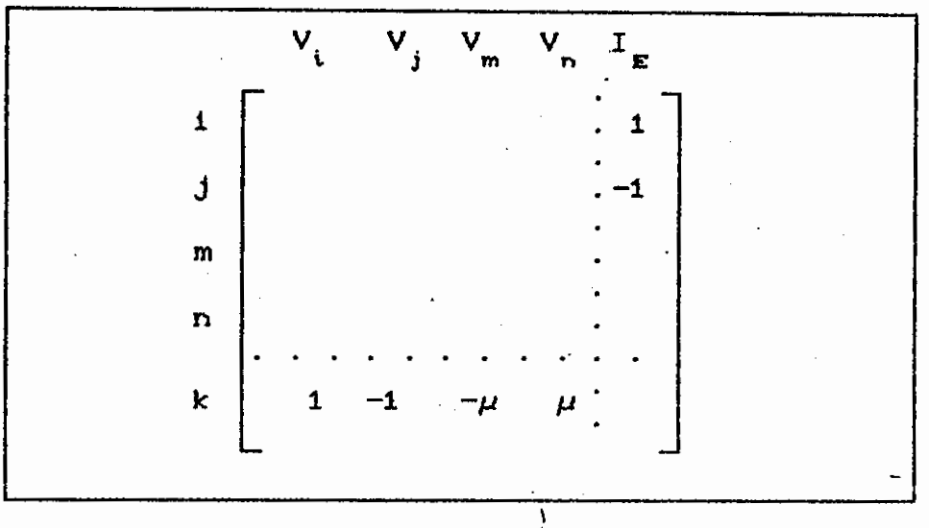


Fig. 2.14 - Estampa da Fonte de Tensão Controlada por Tensão

Fonte de Tensão Controlada por Corrente:

Uma fonte de tensão controlada por corrente ligada aos nós i e j tem sua tensão proporcional à corrente de um ramo I_r. A constante de proporcionalidade "r" é definida como a transresistência da fonte por ser dimensionalmente a razão de uma tensão por uma corrente.

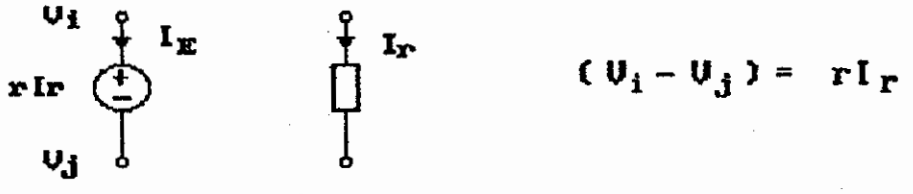


Fig. 2.15 - Fonte de Tensão Controlada por Corrente

Como as demais fontes de tensão, a representação da fonte de tensão controlada por corrente é feita na parte híbrida da matriz nodal modificada (Corrente I_E incluída no conjunto das variáveis). Assim sendo a corrente I_E que percorre esta fonte é acrescida ao nó i e subtraída do nó j. A relação constitutiva desta fonte ($V_i - V_j = rI_r$) é acrescida ao conjunto de equações (linha "k" da estampa). O ramo

controlador deve ter sua corrente I_r incluída no conjunto das variáveis nodais modificadas. A estampa correspondente é mostrada logo a seguir.

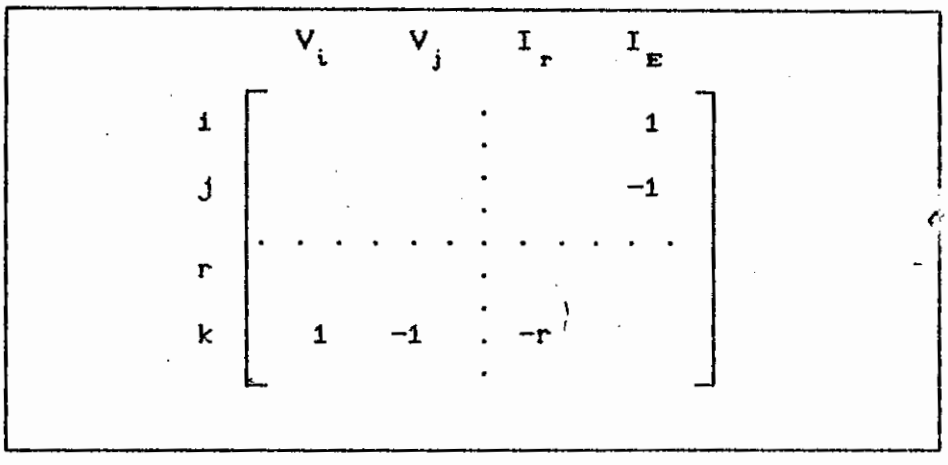


Fig. 2.16 - Estampa da Fonte de Tensão Controlada por Corrente

As fontes de corrente tanto controladas por tensão quanto por corrente podem ser representadas em duas formas interessantes, apresentadas a seguir.

Fonte de Corrente Controlada por Tensão:

Uma fonte de corrente controlada por tensão ligada aos nós *i* e *j* tem sua corrente proporcional à tensão entre os nós *m* e *n*. A constante de proporcionalidade "gm" é definida como a transcondutância da fonte por ser dimensionalmente a razão de uma corrente por uma tensão.

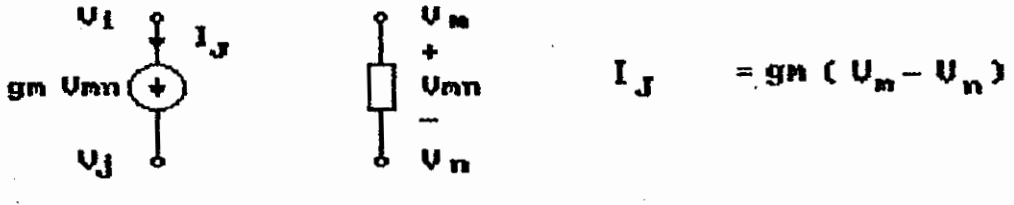


Fig. 2.17 - Fonte de Corrente Controlada por Tensão

Caso a corrente da fonte de corrente controlada por tensão não seja incluída como variável de saída pode-se descrever sua contribuição por uma corrente $g_m(V_m - V_n)$ deixando o nó i e uma corrente $-g_m(V_m - V_n)$ deixando o nó j, conforme visto na estampa abaixo:

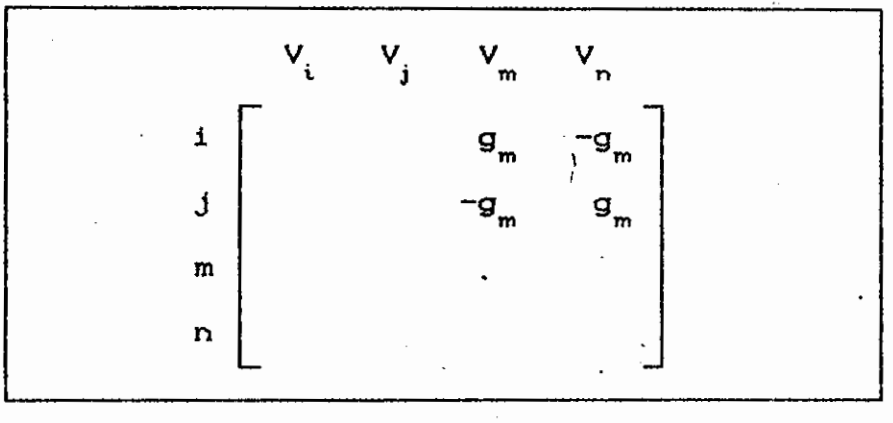


Fig. 2.18 - Estampa da Fonte de Corrente Controlada por Tensão na Parte Nodal

No caso em que a corrente é solicitada como variável de saída acrescenta-se a relação constitutiva da fonte na parte híbrida da matriz nodal modificada (linha "k"), a corrente I_J é somada ao nó i e subtraída do nó j, conforme visto na estampa abaixo:

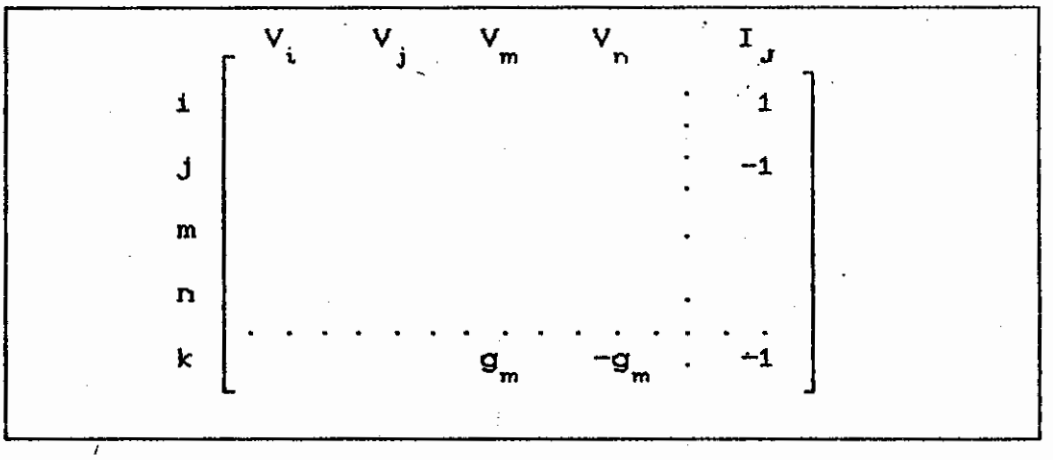


Fig. 2.19 - Estampa da Fonte de Corrente Controlada por Tensão na Parte Híbrida

Fonte de Corrente Controlada por Corrente:

Uma fonte de corrente controlada por corrente ligada aos nós i e j tem sua corrente proporcional à corrente de um ramo I_r. A constante de proporcionalidade "β" é definida como o ganho de corrente da fonte.

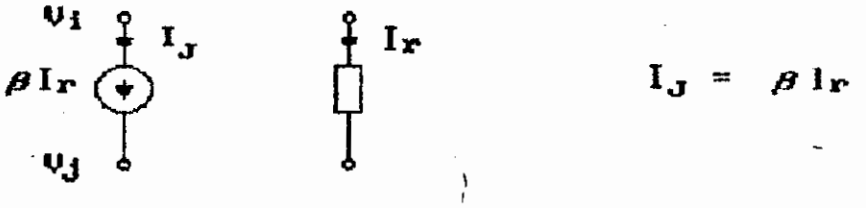


Fig. 2.20 - Fonte de Corrente Controlada por Corrente

Caso a corrente não seja solicitada como variável de saída a contribuição à matriz nodal modificada é bastante simples, acrescentando-se βI_r ao nó i e $-\beta I_r$ ao nó j. A corrente no ramo controlador deve ser incluída no conjunto das variáveis. A estampa correspondente é vista abaixo:

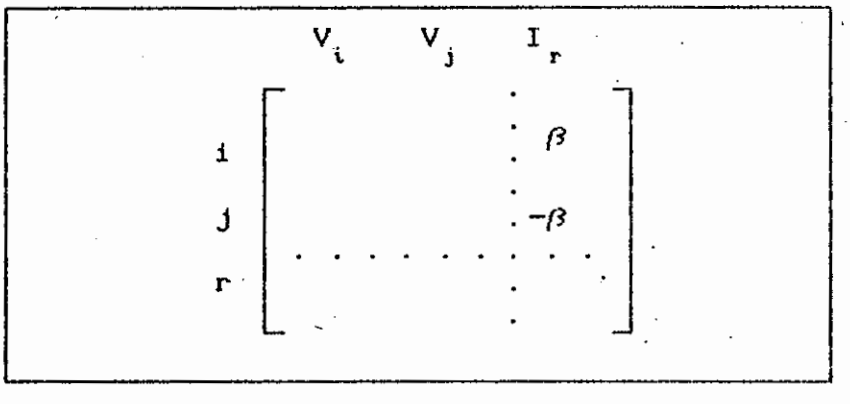


Fig 2.21 - Estampa da Fonte de Corrente Controlada por Corrente na parte Nodal

Caso a corrente da fonte seja solicitada como variável de saída inclui-se a relação constitutiva da fonte no conjunto de equações (linha "k"), conforme visto a seguir:

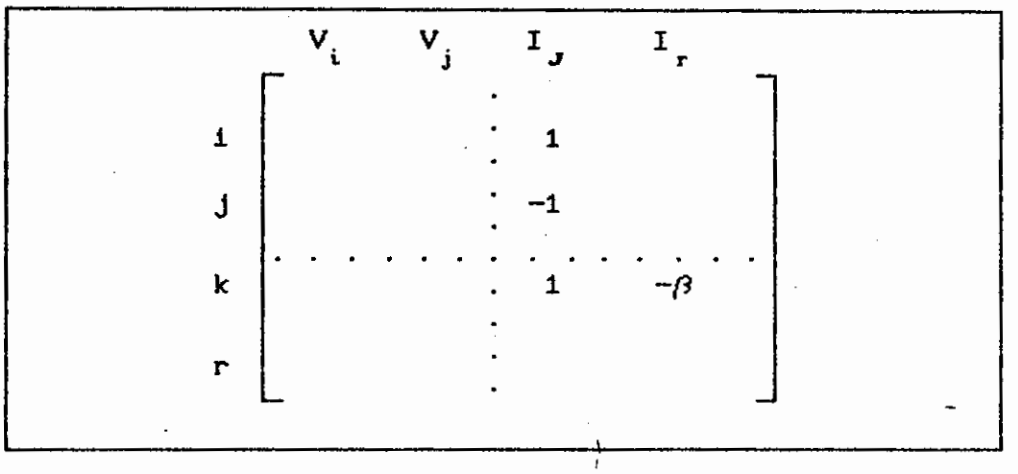


Fig. 2.22 - Estampa da Fonte de Corrente Controlada por Corrente na parte Híbrida

2.3 - EXEMPLOS

Para ilustrar a formulação de circuitos proposta e tornar mais claro o tema abordado são apresentados dois exemplos a seguir. O primeiro circuito (Fig. 2.23) constitui-se de cinco componentes, entre os quais uma fonte de tensão e um indutor. A construção direta da equação matricial (2.2) foi feita pela inspeção da topologia do circuito, aplicando-se as estampas correspondentes a cada ramo. Conforme desenvolvimento anterior nota-se que a fonte de tensão e o indutor aparecem na parte híbrida da matriz, suas correntes estando disponíveis como saídas. As linhas pontilhadas na eq. (2.2) permitem identificar as matrizes Yr, B, C e D.

equações de circuito. Uma vez que o computador digital seja escolhido como ferramenta de auxílio ao projeto de circuitos eletrônicos é necessário que o modelo matemático utilizado para representar os circuitos seja adequado à ferramenta. O método pelo qual as equações de circuito são formuladas afeta significativamente o tempo de pré-processamento ("set-up time"), o esforço computacional, a quantidade de memória necessária e o tempo de execução das ferramentas de auxílio à análise e síntese de circuitos.

O método escolhido deve ser flexível, computacionalmente eficiente e econômico em relação ao uso de memória. Durante a década de 70, um dos métodos mais utilizados consistia na formulação NODAL (Chua & Lin, 1975 pp 166-196 [1]), que além de atender as necessidades comentadas, fornecia uma diagonal numericamente bem condicionada. Este método apresentava entretanto, algumas limitações em sua forma básica, que, por exemplo, não permitiam tratar de maneira eficiente fontes de tensão. A representação de circuitos contendo indutores e capacitores pela formulação nodal no domínio do tempo também era problemática pois levava a um sistema de equações diferenciais ordinárias de segunda ordem. Outra desvantagem do método nodal é que correntes de ramos não eram obtidas de forma conveniente como saída.

Algumas formulações foram propostas antes de 1975 no sentido de generalizar o método nodal. Por exemplo, no programa CANCER [21], cada fonte independente de tensão era substituída pela fonte de corrente equivalente de Norton. Calahan [22] utilizou giradores para converter indutores para capacitores no domínio do tempo. Algumas formulações híbridas como a TABLEAU [23] foram propostas, superando todas as limitações citadas, entretanto computacionalmente de forma bastante ineficiente [16]. A formulação TABLEAU constrói um sistema com $2b + n - 1$ equações lineares com $2b + n - 1$

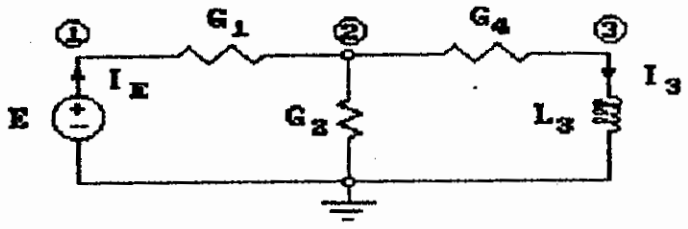


Fig. 2.23 - Circuito contendo fonte de tensão e Indutor.

$$\left[\begin{array}{ccc:cc} G_1 & -G_1 & 0 & -1 & 0 \\ -G_1 & G_1+G_2+G_4 & -G_4 & 0 & 0 \\ 0 & -G_4 & G_4 & 0 & 1 \\ \hdashline 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -L_3\frac{d}{dt} \end{array}\right] \left[\begin{array}{c} V_1 \\ V_2 \\ V_3 \\ \hdashline I_E \\ I_3 \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ \hdashline E \\ 0 \end{array}\right] \qquad (2.3)$$

O segundo exemplo (fig. 2.24) ilustra a formulação nodal modificada para um circuito contendo algumas fontes controladas e um capacitor.

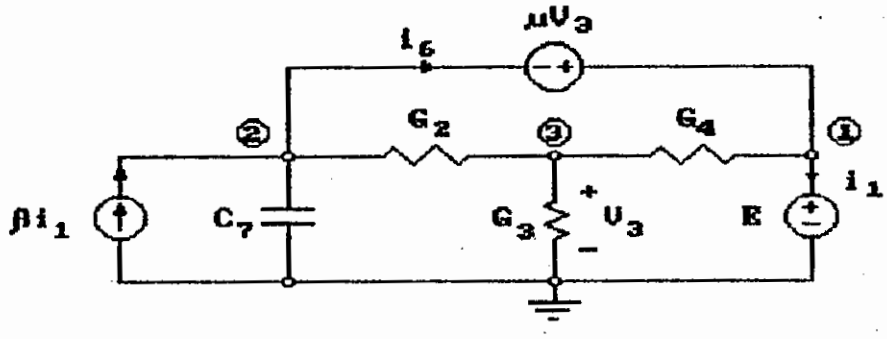


Fig. 2.24 - Circuito com fontes controladas e um capacitor.

$$\left[\begin{array}{ccc:cc} G_4 & 0 & -G_4 & 1 & -1 \\ 0 & G_2+C_7\frac{d}{dt} & -G_2 & -\beta & 1 \\ G_4 & -G_2 & G_4+G_3+G_2 & 0 & 0 \\ \hdashline 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & -\mu & 0 & 0 \end{array}\right]\left[\begin{array}{c} V_1 \\ V_2 \\ V_3 \\ \hdashline I_1 \\ I_6 \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ \hdashline E \\ 0 \end{array}\right] \qquad (2.4)$$

Concluindo este capítulo são apresentadas em tabelas as estampas das contribuições de cada componente à matriz nodal modificada. Estas tabelas podem ser utilizadas para uma consulta rápida ao se construir a matriz nodal modificada de forma direta.

O sistema de equações obtido não está ainda em uma forma adequada para implementação em computador digital. No capítulo 3, a seguir, discute-se o modelo discreto associado aos métodos de solução numérica das equações diferenciais que permitem a sua implementação em computador digital.

COMPONENTE	Corrente nao e' variável de saída	Corrente incluída como variável de saída
G	$\begin{array}{c} \\ i \\ j \end{array}\begin{array}{c} \begin{array}{cc} Vi & Vj \end{array} \\ \begin{bmatrix} G & -G \\ -G & G \end{bmatrix} \end{array}$	$\begin{array}{c} \\ i \\ j \\ k \end{array}\begin{array}{c} \begin{array}{ccc} Vi & Vj & Ir \end{array} \\ \left[\begin{array}{cc:c} & & 1 \\ & & -1 \\ \hdashline G & -G & -1 \end{array}\right] \end{array}$
C	$\begin{array}{c} \\ i \\ j \end{array}\begin{array}{c} \begin{array}{cc} Vi & Vj \end{array} \\ \begin{bmatrix} C\frac{d}{dt} & -C\frac{d}{dt} \\ -C\frac{d}{dt} & C\frac{d}{dt} \end{bmatrix} \end{array}$	$\begin{array}{c} \\ i \\ j \\ k \end{array}\begin{array}{c} \begin{array}{ccc} Vi & Vj & Ic \end{array} \\ \left[\begin{array}{cc:c} & & 1 \\ & & -1 \\ \hdashline C\frac{d}{dt} & -C\frac{d}{dt} & -1 \end{array}\right] \end{array}$
L		$\begin{array}{c} \\ i \\ j \\ k \end{array}\begin{array}{c} \begin{array}{ccc} Vi & Vj & Il \end{array} \\ \left[\begin{array}{cc:c} & & 1 \\ & & -1 \\ \hdashline 1 & -1 & -L\frac{d}{dt} \end{array}\right] \end{array}$
V		$\begin{array}{c} \\ i \\ j \\ k \end{array}\begin{array}{c} \begin{array}{ccc} Vi & Vj & Ie \end{array} \\ \left[\begin{array}{cc:c} & & 1 \\ & & -1 \\ \hdashline 1 & -1 & \end{array}\right] \end{array} \quad \begin{array}{c} \text{VETOR} \\ \text{FONTE} \\ \\ \left[\begin{array}{c} \\ \\ \hdashline E \end{array}\right] \end{array}$
I	$\begin{array}{c} \text{VETOR} \\ \text{FONTE} \\ \\ i \\ j \end{array}\begin{bmatrix} \\ -I \\ +I \end{bmatrix}$	

Tabela 2.1 - Contribuição de alguns componentes lineares à matriz nodal-modificada. G - Condutância, C - Capacitor, L - Indutor, V - Fonte de Tensão Independente, I - Fonte de Corrente Independente

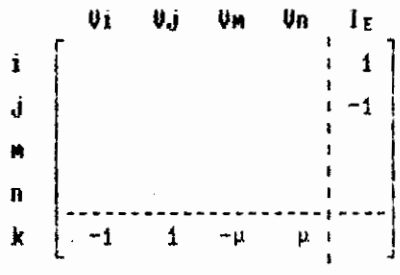
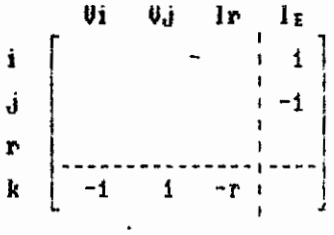
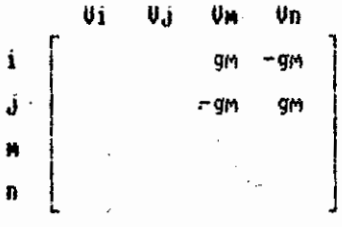
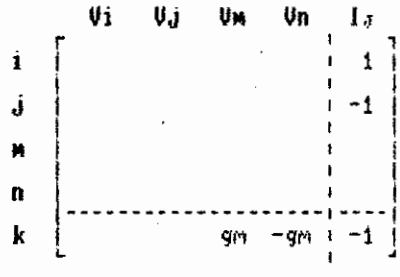
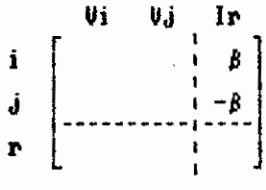
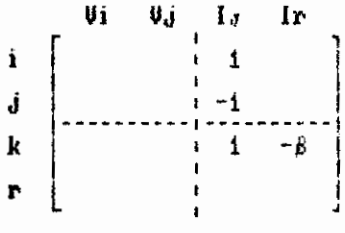
COMPONENTE	Corrente não e' variável de saída	Corrente incluída como variável de saída
E		
H		
G		
F		

Tabela 2.2 - Contribuição das Fontes Controladas à matriz nodal-modificada

m, n - Nós da tensão controladora
r - Ramo da corrente controladora

E - Fonte de Tensão Controlada por Tensão
H - Fonte de Tensão Controlada por Corrente
G - Fonte de Corrente Controlada por Tensão
F - Fonte de Corrente Controlada por Corrente

μ - Ganho de tensão
r - Transresistência
gm - Transcondutância
β - Ganho de Corrente

3 - DISCRETIZAÇÃO DOS MODELOS ASSOCIADOS A FORMULAÇÃO NODAL MODIFICADA

Conforme visto no capítulo anterior a formulação das equações de equilíbrio de um certo circuito pela abordagem nodal modificada fornece um sistema de equações diferenciais ordinárias de primeira ordem. Os componentes que introduzem a característica dinâmica nas equações são os indutores e os capacitores. A liberdade de expressar o indutor na parte híbrida pela abordagem nodal modificada permite que o sistema seja apenas de equações diferenciais de primeira ordem.

Após a construção do sistema de equações de circuito e o estabelecimento dos sinais de entrada (funções forçantes), a solução temporal do sistema de equações fornece a descrição do comportamento das tensões e correntes dos componentes de circuito. O resultado destas soluções é, naturalmente, função do modelo adotado na descrição do circuito. Se o modelo é simplificado, visando reduzir o esforço computacional ou meramente pelo desconhecimento das características reais dos

componentes, deve-se esperar uma resposta apenas aproximada da realidade. Entretanto, a prática mostra que modelagens simplificadas atendem à grande maioria das aplicações lineares.

Para que a simulação de circuitos possa ser feita para um circuito arbitrariamente grande é necessário que seja utilizada uma poderosa ferramenta de cálculo, geralmente um computador digital. Por outro lado, a utilização do computador para a solução do sistema de equações requer uma metodologia que permita resolver numericamente o problema.

A ênfase na simulação de circuitos de uma forma adequada à utilização em processamento de sinais, isto é, circuitos excitados por sinais descritos por suas amostras armazenadas em arquivo, impõe condições adicionais aos metódos numéricos aplicáveis.

Serão utilizados neste capítulo os resultados conhecidos nas referências bibliográficas como soluções numéricas de problemas de valor inicial. Duas abordagens são geralmente empregadas, a aproximação por série de Taylor e a aproximação polinomial [1]. Devido à característica dos sinais de entrada serem amostrados, a forma de resolver as equações diferenciais de primeira ordem por algoritmos numéricos que envolvem aproximações polinomiais foi adotada [1, 6, 7, 10, 12]. A solução numérica permite reescrever as equações nodais modificadas de uma forma apropriada para sua representação e solução em um computador digital. Maiores informações sobre as soluções numéricas de problemas de valor inicial são discutidas no apêndice A. A seguir introduzir-se-á o conceito de modelo discreto associado a capacitores e indutores.

3.1 - Definição de Modelo Discreto Associado

Do ponto de vista da integração numérica, as equações diferenciais que caracterizam o capacitor e o indutor podem ser resolvidas utilizando *circuitos resistivos associados* com o algoritmo de integração. Desta forma tem-se então *modelos de circuito discretos associados*. O adjetivo "*associado*" é utilizado para enfatizar que os modelos associados a diferentes algoritmos de integração são diferentes. O adjetivo "*discreto*" é utilizado para destacar que os parâmetros do modelo são discretos por natureza; i. é., eles diferem de um instante de integração para outro. O modelo discreto aproxima equações diferenciais por equações a diferenças. A técnica do modelo discreto associado reduz a análise transitória de um circuito dinâmico a uma análise DC de um circuito resistivo.

Um *modelo discreto associado* pode ser derivado a partir dos métodos implícitos de integração numérica [1]. No presente trabalho foram implementados os algoritmos de integração implícitos de Euler regressivo, Trapezoidal e de Gear de segunda a quarta ordem [1]. Algoritmos de ordem mais elevada não foram implementados pois apresentam maiores restrições de estabilidade numérica condicional, como também oneram com maior esforço computacional [1] o processo de solução.

A seguir são apresentados os algoritmos de integração utilizados neste trabalho e os modelos discretos associados para o capacitor e indutor para cada um dos algoritmos.

3.2 - Algoritmo de Euler Regressivo

O algoritmo de *Euler Regressivo* para a solução de uma equação diferencial de primeira ordem, da forma $x' = f(x)$ com um passo h é dado por

$$x_{n+1} = x_n + h\, f(x_{n+1})$$

$$= x_n + h\, x'_{n+1} \qquad (3.1)$$

onde o índice subscrito (n+1, n) indica o instante em que a variável é considerada; o apóstrofo denota diferenciação em relação ao tempo. Mais detalhes podem ser encontrados no apêndice A.

Como este algoritmo utiliza $f(x_{n+1}) = x'_{n+1}$, a derivada da variável independente em $t = t_{n+1}$, o mesmo é dito um algoritmo de integração implícito. Em contraposição, o algoritmo de Euler progressivo utiliza a derivada da variável independente em $t = t_n$ e é dito um algoritmo de integração explícito. Devido às melhores características de estabilidade do algoritmo implícito, adotou-se neste trabalho apenas a versão implícita [1] (As regiões de estabilidade de diversos algoritmos de integração numérica podem ser vistas no final do apêndice A) .

Para este algoritmo serão apresentados a seguir os modelos discretos associados ao capacitor e ao indutor.

3.2.1 - Modelo Discreto Associado ao Capacitor para o Algoritmo de Euler Regressivo

O algoritmo de Euler Regressivo é o método mais simples

que pode ser utilizado para a discretização das equações diferenciais que descrevem um circuito. O erro de truncamento associado é da ordem de h^2 (onde h é o passo de integração, [1]). Este é o maior erro dentre os algoritmos implementados, porém, devido às suas excelentes características de estabilidade numérica, o algoritmo de Euler Regressivo é o mais indicado para a simulação de circuitos críticos.

A corrente que circula em um capacitor linear em função do tempo pode ser expressa pela equação

$$i(t) = C \frac{dv}{dt} = C\, v'(t)$$

Para um instante genérico t_{n+1} tem-se

$$v'(t_{n+1}) = \frac{1}{C}\, i(t_{n+1}) \qquad (3.2)$$

Observa-se que (3.2) é uma relação exata enquanto (3.1) é uma solução aproximada. Se aproximarmos as soluções exatas $v'(t_{n+1})$ e $i(t_{n+1})$ por v'_{n+1} e i_{n+1}, respectivamente, então (3.2) torna-se

$$v'_{n+1} = \frac{1}{C}\, i_{n+1} \qquad (3.3)$$

Substituindo (3.3) em (3.1) e resolvendo para i_{n+1}, obtém-se

$$i_{n+1} = \frac{C}{h}\, v_{n+1} - \frac{C}{h}\, v_n \qquad (3.4)$$

A expressão (3.4) pode ser realizada pela *porta linear equivalente* da Fig. 3.1, com corrente de porta i_{n+1} e tensão de porta v_{n+1}.

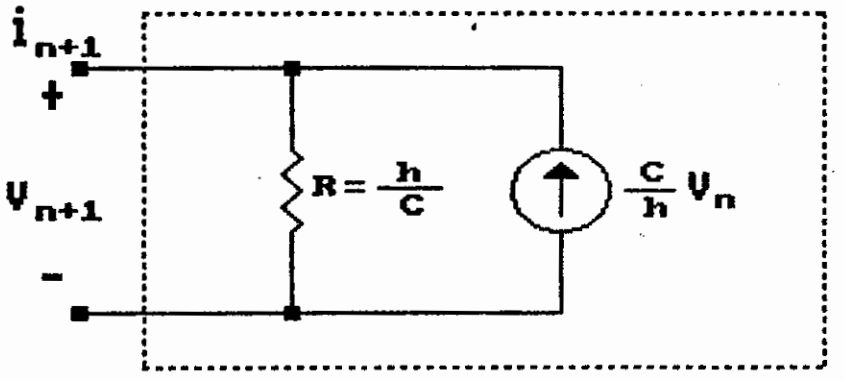


Fig 3.1 Modelo Discreto Associado com o algoritmo de Euler Regressivo para um capacitor linear.

É interessante observar na Fig. 3.1 que o resistor da porta linear equivalente é obtido em virtude do algoritmo de integração ser implícito, havendo uma relação entre i_{n+1} e v_{n+1}. Os algoritmos de integração explícitos não permitem uma discretização tão eficiente.

Enquanto o valor do resistor R permanece fixo (assumindo um passo de integração constante), a fonte de corrente depende da tensão v_n, que é computada previamente. Obviamente, se os únicos elementos armazenadores de energia no circuito forem capacitores lineares, e cada capacitor for substituido pelo seu Modelo Discreto Associado da Fig. 3.1, então o circuito resultante será *puramente resistivo*.

Atribuindo o valor $(C_j / h)v_o^{(j)}$ à fonte de corrente associada ao capacitor C_j, com $v_o^{(j)}$ sendo a tensão inicial sobre o capacitor C_j, podemos obter a tensão $v_1^{(j)}$ sobre cada capacitor aplicando um dos métodos de solução de circuitos disponíveis. No caso particular deste trabalho a solução é obtida pela formulação nodal modificada, por razões já comentadas no capítulo 2. A tensão $v_1^{(j)}$ é então a tensão sobre o capacitor C_j em $t = t_1 \stackrel{\Delta}{=} h$ calculada pelo algoritmo de Euler Regressivo [1].

A cada passo de integração do algoritmo o modelo discreto de circuito é *atualizado*. Assim sendo, o valor da fonte de corrente no modelo para C_j é alterado de $(C_j / h)v_o^{(j)}$ para $(C_j / h)v_1^{(j)}$. Este circuito resistivo atualizado pode então ser resolvido para fornecer $v_2^{(j)}$, que é a tensão sobre C_j em $t = t_2 \triangleq t_1 + h$. Este procedimento deve então ser repetido quantas vezes for necessário, ou seja, até que os sinais de entrada sejam totalmente processados.

3.2.2 - Relações Constitutivas Associadas ao Algoritmo de Euler Regressivo para o Capacitor

Para o modelo discreto associado ao capacitor pelo algoritmo de integração de Euler Regressivo tem-se que a corrente no instante de integração pode ser calculada por $i_{n+1} = \frac{C}{h} v_{n+1} - \frac{C}{h} v_n$. Conforme visto na seção 3.2.1 a porta linear equivalente corresponde a um resistor (de valor h/C) em paralelo com uma fonte de corrente independente, cujo valor é determinado pela tensão sobre o capacitor no instante anterior (com ganho C/h). Desta forma tem-se uma corrente $C/h\ (V_i - V_j)_{n+1}$ deixando o nó i e uma corrente $-C/h\ (V_i - V_j)_{n+1}$ deixando o nó j no lado das incógnitas. A esta contribuição soma-se uma corrente $C/h\ (V_i - V_j)_n$ deixando o nó i e $-C/h\ (V_i - V_j)_n$ deixando o nó j, pelo lado das variáveis independentes, pois o valor de $(V_i - V_j)_n$ já é conhecido a priori.

As considerações acima podem ser apresentadas de forma compacta por meio de uma estampa. A Fig. 3.2 corresponde à contribuição do modelo discreto associado ao capacitor à matriz nodal modificada:

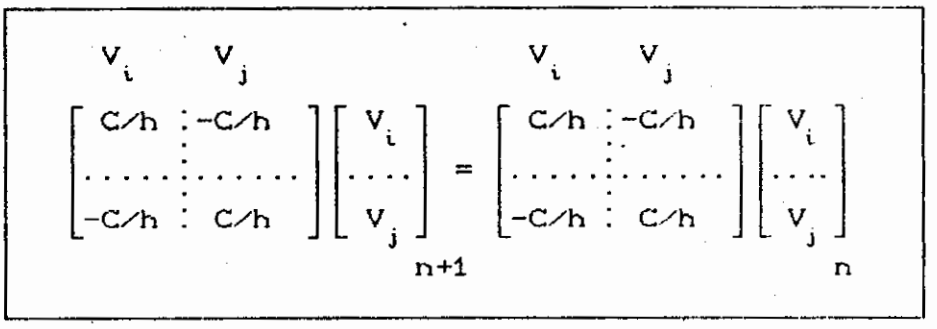


Fig. 3.2 - Estampa do modelo discreto associado ao capacitor para o algoritmo de Euler Regressivo na parte nodal

Onde o subscrito "n+1" na figura acima indica as variáveis no instante $t = t_{n+1}$, e "n" indica as variáveis no instante $t = t_n$.

Esta estampa corresponde a um capacitor ocupando a parte nodal da matriz nodal modificada (Yr em 2.2). Desta forma a corrente do capacitor não está disponível no conjunto das variáveis. Caso a corrente do capacitor seja solicitada como uma das variáveis de saída haverá acréscimo de uma linha e uma coluna na matriz. A relação constitutiva passa a ocupar uma posição híbrida no conjunto de equações. A estampa que considera a corrente do capacitor como variável de saída é obtida por um procedimento similar ao descrito acima, fornecendo:

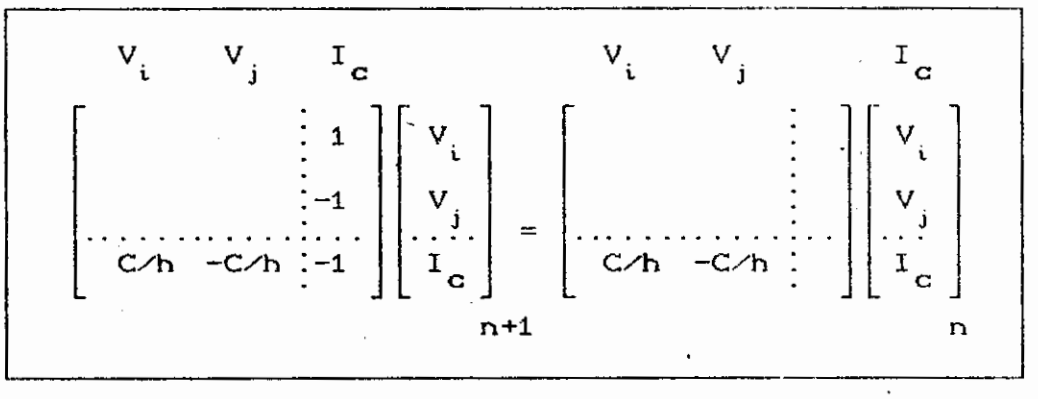


Fig. 3.3 - Estampa do modelo discreto associado ao capacitor para o algoritmo de Euler Regressivo na parte hibrida

As estampas das figuras 3.2 e 3.3 permitem construir por inspeção as equações de equilíbrio de um circuito contendo capacitores discretizados pelo algoritmo de Euler Regressivo. Nestas estampas estão embutidas as relações constitutivas do modelo discretizado associado ao capacitor.

As estampas acima consideram o capacitor conectado aos nós *i* e *j*. Caso o capacitor esteja aterrado, i.é., um dos nós (*i* ou *j*) é o nó de referência então a linha e a coluna correspondentes não devem constar da estampa. Neste caso as duas "matrizes" da Fig. 3.2 conterão apenas um elemento. Na Fig 3.3, a "matriz" do lado esquerdo conterá três elementos e a "matriz" do lado direito apenas um elemento.

Na seção seguinte apresenta-se o desenvolvimento para obtenção da estampa associada ao indutor linear.

3.2.3 - Modelo Discreto Associado ao Indutor para o Algoritmo de Euler Regressivo

A obtenção do modelo para o indutor linear é semelhante à do capacitor, pelo fato de ser o seu dual elétrico. O desenvolvimento seguinte leva a modelos de indutor como resistores em paralelo com fontes de corrente. A formulação nodal modificada contém a corrente de cada indutor como variável híbrida, aumentando em uma unidade a ordem da matriz nodal modificada. Isto é necessário para que as equações diferenciais do circuito original sejam apenas de primeira ordem.

No caso do indutor o algoritmo de *Euler regressivo* para a solução da equação diferencial de primeira ordem $i' = f(i)$ com passo de integração h é dado por

$$i_{n+1} = i_n + h\, f(i_{n+1})$$

$$= i_n + h\, i'_{n+1} \qquad (3.5)$$

A relação exata $v(t) = Li'(t)$ para o indutor linear pode ser aproximada por

$$i'_{n+1} = \frac{1}{L} v_{n+1} \qquad (3.6)$$

Substituindo (3.6) em (3.5) obtém-se:

$$i_{n+1} = \frac{h}{L} v_{n+1} + i_n \qquad (3.7)$$

A porta linear equivalente representada pela equação (3.7) é vista na Fig. 3.4.

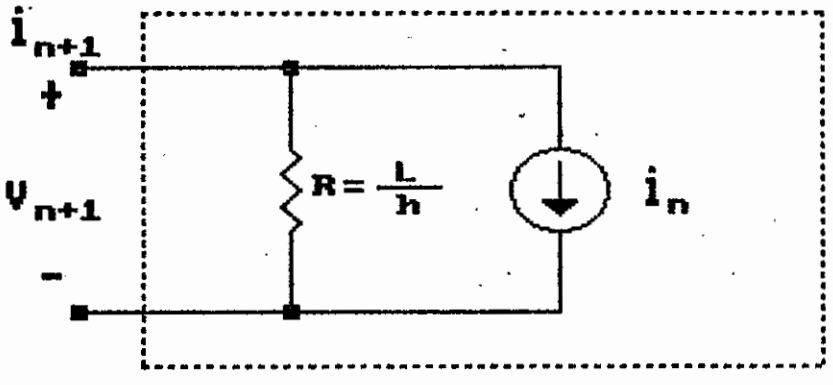


Fig. 3.4 – Modelo Discreto Associado com o algoritmo de Euler regressivo para o Indutor Linear.

Este modelo discreto associado permite que cada indutor do circuito a ser simulado seja substituído por um resistor e uma fonte de corrente. Como o valor da fonte é determinado pela corrente no instante anterior, torna-se necessário incluir a corrente do indutor no conjunto das variáveis nodais modificadas. A topologia do circuito não é alterada, no sentido de que não são criados nós adicionais.

3.2.4 - Relações Constitutivas Associadas ao Algoritmo de Euler Regressivo para o Indutor

A obtenção da estampa que descreve a contribuição do modelo discreto associado ao indutor para o algoritmo de Euler Regressivo é semelhante ao procedimento do capacitor. Nota-se que apenas uma estampa é mostrada em virtude de que o indutor aparece na parte híbrida da matriz.

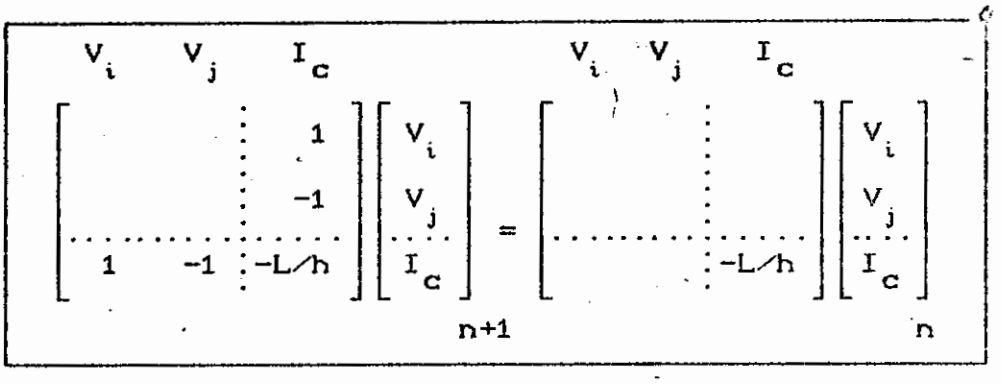


Fig. 3.5 - Estampa do modelo discreto associado ao indutor para o algoritmo de Euler Regressivo na parte híbrida

Com as estampas das figuras 3.2, 3.3 e 3.5 pode ser construído o sistema de equações nodais modificadas de um circuito em que os capacitores e indutores tenham sido discretizados pelo algoritmo de Euler Regressivo. Os demais componentes básicos de um circuito linear têm estampas idênticas às apresentadas no capítulo 2.

A seguir apresenta-se o algoritmo Trapezoidal como outra alternativa de algoritmo numérico para a discretização das equações diferenciais que descrevem o capacitor e o indutor.

3.3 - ALGORITMO TRAPEZOIDAL

Dentre os diversos algoritmos de integração multipassos encontrados na literatura o algoritmo trapezoidal é um caso particular da família de algoritmos conhecidos como de Adams-Moulton [1]. É considerado um algoritmo implícito de segunda ordem por fornecer a solução exata para polinômios de segunda ordem.

Pelo método de integração numérica trapezoidal a solução de $x' = f(x)$, é dada por:

$$x_{n+1} = x_n + \frac{h}{2}\left[f(x_{n+1}) + f(x_n) \right]$$

$$= x_n + \frac{h}{2} x'_{n+1} + \frac{h}{2} x'_n \qquad (3.8)$$

Este algoritmo utiliza x_n e uma média ponderada das derivadas da variável independente para obter o novo valor x_{n+1}. No método trapezoidal tem-se um esforço computacional maior e uma região de estabilidade mais restrita que a região de estabilidade do método de Euler (Apêndice A, [1]). O benefício da utilização deste método reside em que o erro entre cada iteração é menor. Isto é, a aproximação da equação diferencial é mais precisa. O erro de truncamento[3] dos algoritmos de Euler é da ordem de h^2 e o erro de truncamento

---

[3] O erro cometido quando se adota uma solução numérica, é composto por um erro de truncamento, devido às restrições do algoritmo, e um erro de arredondamento, devido às restrições de comprimento de palavra do computador utilizado (Ver apêndice A-4).

do algoritmo trapezoidal é da ordem de h^3.

3.3.1 - Modelo Discreto Associado ao Capacitor para o Algoritmo Trapezoidal

Apesar da relação v-i exata para o capacitor em $t = t_{n+1}$ ser a equação (3.2) e que em $t = t_n$

$$v'(t_n) = \frac{1}{C}\, i(t_n) \qquad (3.9)$$

pode-se aproximar estas duas relações por

$$v'_{n+1} = \frac{1}{C}\, i_{n+1} \qquad (3.10)$$

$$v'_n = \frac{1}{C}\, i_n \qquad (3.11)$$

Substituindo (3.11) e (3.10) em (3.8) para resolver i_{n+1}, obtém-se:

$$i_{n+1} = \frac{2C}{h}\, v_{n+1} - \left(\frac{2C}{h}\, v_n + i_n \right) \qquad (3.12)$$

A equação (3.12) pode ser representada pela *porta linear equivalente* da Fig. 3.6.

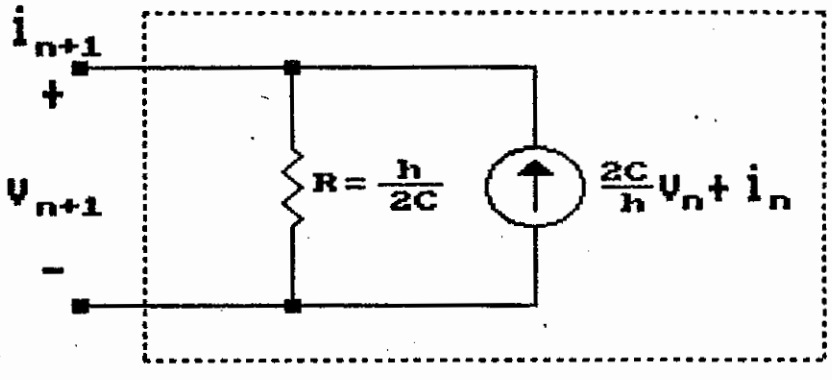


Fig. 3.6 Modelo Discreto Associado do algoritmo Trapezoidal para um capacitor linear.

O modelo discreto associado da Fig. 3.6 mostra que o capacitor é substituído por um resistor e uma fonte de corrente controlada quando o algoritmo trapezoidal é utilizado na discretização da equação que descreve o capacitor. Comparando-se as figuras 3.1 e 3.6 nota-se que a estrutura das portas lineares equivalentes são semelhantes, diferindo no valor do resistor e principalmente na fonte de corrente, que no caso trapezoidal é controlada tanto pela tensão como pela corrente no instante anterior. No processo de discretização todos os capacitores são substituídos pela porta linear da Fig. 3.5, sem alterar a quantidade de nós que descrevem o circuito.

3.3.2 - Relações Constitutivas Associadas ao Algoritmo Trapezoidal para o Capacitor

O algoritmo de integração trapezoidal utiliza a tensão e a corrente anteriores para atualizar a fonte de corrente do modelo associado do capacitor ($i_{n+1} = \frac{2C}{h} v_{n+1} - \frac{2C}{h} v_n - i_n$), assim é conveniente que o capacitor seja introduzido na parte híbrida da matriz nodal modificada de tal forma que a corrente esteja disponível para atualizar a fonte de corrente do modelo associado. A estampa correspondente é mostrada a seguir:

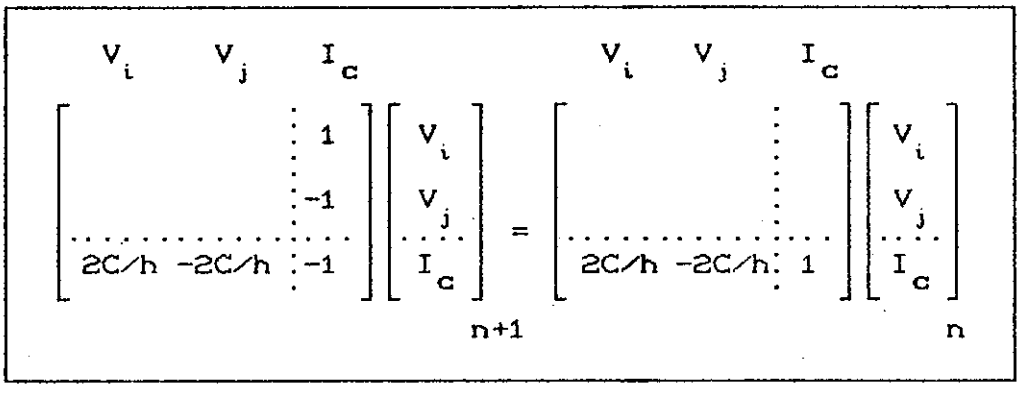


Fig. 3.7 - Estampa do modelo discreto associado ao capacitor para o algoritmo Trapezoidal

Esta estampa pode ser utilizada na construção direta da matriz nodal modificada, quando o capacitor é discretizado pelo algoritmo de integreção trapezoidal. Da mesma forma que para o algoritmo de Euler Regressivo, caso o capacitor esteja aterrado, a linha e a coluna correspondentes ao nó de referência não aparecem na estampa.

3.3.3 - Modelo Discreto Associado ao Indutor para o Algoritmo Trapezoidal

Para o caso do indutor utilizando-se o *algoritmo trapezoidal* para i' = f(i) obtém-se

$$i_{n+1} = i_n + \frac{h}{2}\left[f(i_{n+1}) + f(i_n) \right]$$

$$= i_n + \frac{h}{2} i'_{n+1} + \frac{h}{2} i'_n \qquad (3.13)$$

De acordo com os procedimentos anteriores, substituimos $i'_{n+1} = \frac{1}{L} v_{n+1}$ e $i'_n = \frac{1}{L} v_n$ para obter

$$i_{n+1} = \frac{h}{2L} v_{n+1} + \left(\frac{h}{2L} v_n + i_n \right) \qquad (3.14)$$

A *porta linear equivalente* representada pela equação (3.14) é vista na Fig. 3.8.

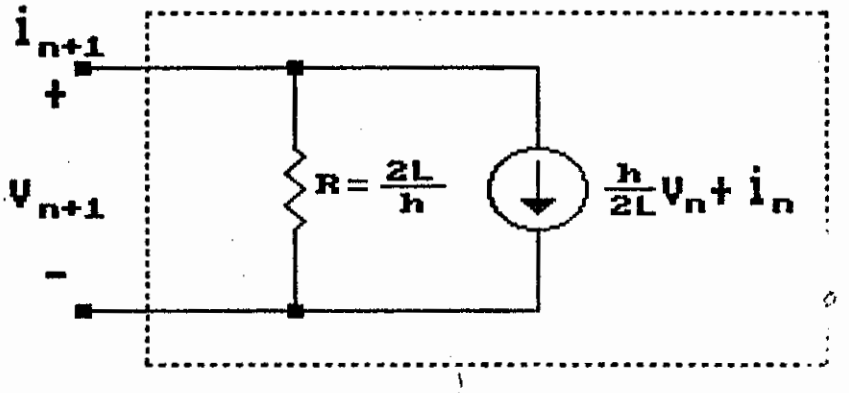


Fig. 3.8 Modelo Discreto Associado com o algoritmo Trapezoidal para o indutor linear.

A Fig. 3.8 mostra que o indutor discretizado pelo algoritmo trapezoidal é substituído por um resistor e uma fonte de corrente controlada. A fonte de corrente é controlada pela tensão e pela corrente do indutor no instante de integração anterior.

3.3.4 - Relações Constitutivas Associadas ao Algoritmo Trapezoidal para o Indutor

O algoritmo de integração trapezoidal utiliza a tensão e a corrente anteriores para atualizar a fonte de corrente do modelo associado do indutor, assim é conveniente que o indutor seja introduzido na parte híbrida da matriz nodal modificada. A estampa correspondente é mostrada a seguir:

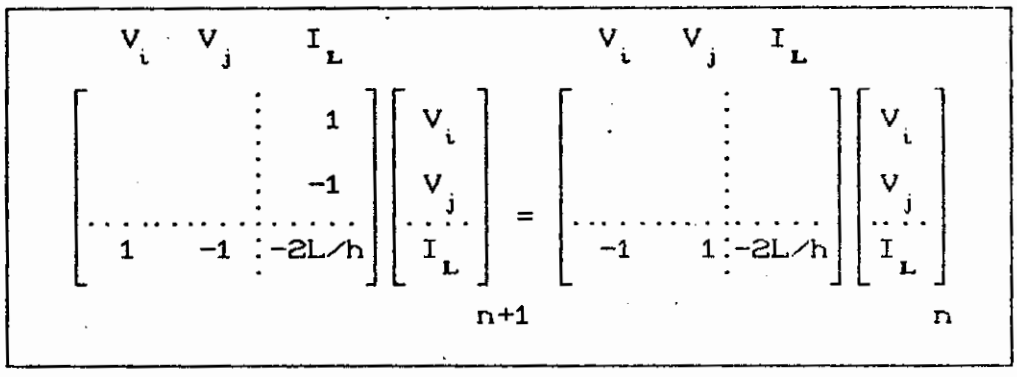


Fig.3.9 - Estampa do modelo discreto associado ao indutor para o algoritmo Trapezoidal

Com as estampas das figuras 3.7 e 3.9, as matrizes de equilíbrio de um circuito podem ser construidas por inspeção da topologia, quando o algoritmo trapezoidal é utilizado na aproximação das equações diferenciais que descrevem o capacitor e o indutor. Nota-se que para este método de discretização todos os capacitores e indutores aparecem na parte híbrida da matriz nodal modificada. É verdade que os indutores aparecem na parte híbrida para qualquer método de discretização, no entanto o capacitor pode permanecer na parte nodal utilizando-se outros algoritmos de integração.

A seguir são apresentados os algoritmos de integração de Gear, para os quais a fonte de corrente dos modelos discretos associados ao capacitor é controlada apenas pelas tensões em instantes anteriores. Isto permite manter os capacitores na parte nodal da matriz nodal modificada. Para um circuito contendo grande quantidade de capacitores haverá diferença significativa na ordem do sistema de equações.

3.4 - Algoritmos de Gear de Segunda a Quarta Ordem

Os algoritmos de Gear apresentam melhores características de estabilidade que outros algoritmos de integração multipassos (Vide em [1], ou no apêndice A, a descrição dos algoritmos de integração multipassos de Adams-Moulton e de Adams-Bashforth) e por este motivo foram escolhidos para a implementação dos modelos discretos de ordem mais elevada do capacitor e do indutor. A solução dos modelos discretos associados segue procedimento semelhante aos algoritmos de Euler Regressivo e trapezoidal e por este motivo é apresentada de forma sucinta.

Os algoritmos de Gear de primeira a quarta ordem (apêndice A, [1]) para a solução de $\dot{x} = f(x)$ são dados por:

1	$x_{n+1} = x_n + h\, f(x_{n+1}, t_{n+1})$
2	$x_{n+1} = \frac{4}{3} x_n - \frac{1}{3} x_{n-1} + h\, \frac{2}{3} f(x_{n+1}, t_{n+1})$
3	$x_{n+1} = \frac{18}{11} x_n - \frac{9}{11} x_{n-1} + \frac{2}{11} x_{n-2} + h\, \frac{6}{11} f(x_{n+1}, t_{n+1})$
4	$x_{n+1} = \frac{48}{25} x_n - \frac{36}{25} x_{n-1} + \frac{16}{25} x_{n-2} - \frac{3}{25} x_{n-3} + h\, \frac{12}{25} f(x_{n+1}, t_{n+1})$

Tabela 3.1 - Algoritmos de Gear até quarta ordem para solução de equações diferenciais ordinárias de 1.a ordem

O algoritmo de Gear de primeira ordem é semelhante ao algoritmo de Euler Regressivo. Para os algoritmos de ordem superior vê-se que a solução considera uma soma ponderada da derivada em x_{n+1} com soluções em pontos anteriores (x_n, x_{n-1}, x_{n-2}, x_{n-3}).

3.4.1 - Modelos Discretos Associados ao Capacitor para os Algoritmos de Gear até Quarta Ordem

Utilizando-se os algoritmos de Gear até quarta ordem para a solução da equação diferencial que descreve o capacitor obtém-se a seguinte tabela de relações constitutivas:

1	$i_{n+1} = \frac{C}{h} v_{n+1} - \frac{C}{h} \left(v_n \right)$
2	$i_{n+1} = \frac{3C}{2h} v_{n+1} - \frac{C}{h} \left(2v_n - \frac{1}{2} v_{n-1} \right)$
3	$i_{n+1} = \frac{11C}{6h} v_{n+1} - \frac{C}{h} \left(3v_n - \frac{3}{2} v_{n-1} + \frac{1}{3} v_{n-2} \right)$
4	$i_{n+1} = \frac{25C}{12h} v_{n+1} - \frac{C}{h} \left(4v_n - 3v_{n-1} + \frac{4}{3} v_{n-2} - \frac{1}{4} v_{n-3} \right)$

Tabela 3.2 - Relações Constitutivas do Capacitor discretizado pelos algoritmos de Gear até quarta ordem

As relações acima podem ser representadas por uma *porta linear equivalente*, de forma similar aos algoritmos já apresentados. No caso particular do algoritmo de Gear de segunda ordem tem-se a *porta linear equivalente* da Fig. 3.10.

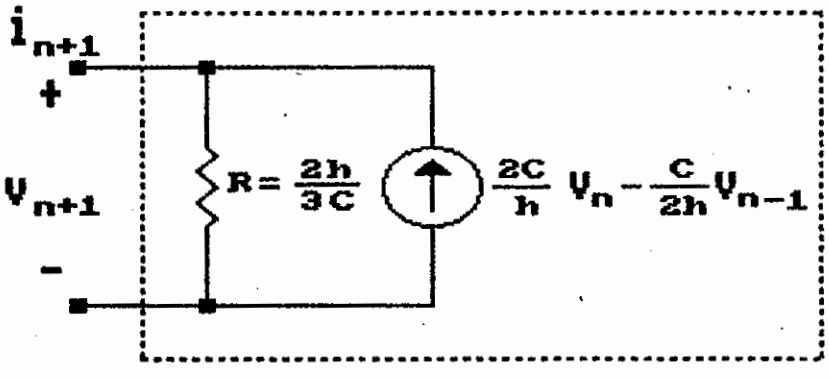


Fig. 3.10 Modelo Discreto Associado com o algoritmo de Gear de segunda ordem para o capacitor linear.

A contrário dos algoritmos de Adams-Bashforth e de Adams-Moulton [1], os algoritmos de Gear permitem obter uma discretização do capacitor computacionalmente mais eficiente. Esta característica deriva do fato de que as portas lineares equivalentes obtidas pelos algoritmos de Gear utilizam apenas as tensões em instantes anteriores na atualização das fontes de corrente associadas, permitindo assim que o capacitor permaneça na parte nodal da matriz nodal modificada.

Visando a construção por inspeção da matriz nodal modificada apresentam-se a seguir as estampas associadas ao capacitor discretizado pelos algoritmos de Gear.

3.4.2 - Estampas associadas aos Algoritmos de Gear de Segunda a Quarta Ordem para o Capacitor

Conforme a tabela 3.2, os modelos discretos associados ao capacitor para os algoritmos de Gear são atualizados pelas tensões já computadas em instantes anteriores. A contribuição destas tensões anteriores aparece no modelo discreto associado como uma fonte de corrente. A ordem do algoritmo estabelece quantas tensões anteriores são utilizadas na atualização do modelo. O sistema de equações nodais modificadas deve então ser estendido de tal forma que todas as atualizações de modelos possam ser feitas metodicamente.

No caso dos algoritmos de Euler Regressivo e Trapezoidal utilizou-se uma matriz auxiliar para permitir a atualização metódica das fontes de corrente associadas em cada passo de solução temporal. Apenas as variáveis no instante anterior eram necessárias para atualizar os modelos. Para os algoritmos de Gear pode-se estender esta metodologia e

criar-se quantas matrizes auxiliares sejam necessárias para atualizar as fontes de corrente dos modelos associados. Esta metologia ficará mais clara após apresentar-se as estampas para os algoritmos de Gear. Ao final do capítulo a metodologia é generalizada.

As estampas correspondentes aos modelos discretos associados ao capacitor para os algoritmos de Gear de segunda a quarta ordem são apresentadas a seguir.

GEAR DE SEGUNDA ORDEM

O algoritmo de Gear de segunda ordem aplicado ao capacitor fornece, de acordo com a tabela 3.2,

$$i_{n+1} = \frac{3C}{2h} v_{n+1} - \frac{C}{h}\left(2v_n - \frac{1}{2}v_{n-1}\right) \qquad (3.15)$$

Para simplificar a notação faz-se a seguinte substituição:

$$\frac{3C}{2h} = g_a; \qquad \frac{2C}{h} = g_0; \qquad \frac{C}{2h} = g_1. \qquad (3.16)$$

Para um capacitor conectado aos nós "i" e "j" tem-se uma corrente

$$g_a (V_i - V_j)_{n+1} - g_0(V_i - V_j)v_n + g_1(V_i - V_j)_{n-1} \qquad (3.17)$$

deixando o nó "i" e

$$-g_a (V_i - V_j)_{n+1} + g_0(V_i - V_j)v_n - g_1(V_i - V_j)_{n-1} \qquad (3.18)$$

deixando o nó "j". V_i e V_j são as tensões nos nós "i" e "j", respectivamente.

As equações 3.17 e 3.18 permitem apresentar a seguinte estampa de um capacitor na parte nodal da matriz nodal modificada:

$$\begin{matrix} & V_i & V_j \end{matrix}\begin{bmatrix} g_0 & -g_0 \\ -g_0 & g_0 \end{bmatrix}\begin{bmatrix} V_i \\ V_j \end{bmatrix}_{n+1} = \begin{matrix} & V_i & V_j \end{matrix}\begin{bmatrix} g_0 & -g_0 \\ -g_0 & g_0 \end{bmatrix}\begin{bmatrix} V_i \\ V_j \end{bmatrix}_{n} - \begin{matrix} & V_i & V_j \end{matrix}\begin{bmatrix} g_1 & -g_1 \\ -g_1 & g_1 \end{bmatrix}\begin{bmatrix} V_i \\ V_j \end{bmatrix}_{n-1}$$

Fig. 3.11 - Estampa associada ao modelo discreto do capacitor para o algoritmo de Gear de Segunda Ordem parte nodal

Naturalmente a estampa do capacitor pode ser obtida incluindo sua corrente no conjunto das variáveis, conforme a seguir:

$$\begin{matrix} V_i & V_j & I_c \end{matrix}\begin{bmatrix} & & 1 \\ & & -1 \\ g_0 & -g_0 & -1 \end{bmatrix}\begin{bmatrix} V_i \\ V_j \\ I_c \end{bmatrix}_{n+1} = \begin{matrix} V_i & V_j & I_c \end{matrix}\begin{bmatrix} & & \\ & & \\ g_0 & -g_0 & \end{bmatrix}\begin{bmatrix} V_i \\ V_j \\ I_c \end{bmatrix}_{n} - \begin{matrix} V_i & V_j & I_c \end{matrix}\begin{bmatrix} & & \\ & & \\ g_1 & -g_1 & \end{bmatrix}\begin{bmatrix} V_i \\ V_j \\ I_c \end{bmatrix}_{n-1}$$

Fig. 3.12 - Estampa associada ao modelo discreto do capacitor para o algoritmo de Gear de Segunda Ordem na parte híbrida

onde g_0, g_1 e g_2 tem os mesmos valores declarados acima.

Estas estampas permitem construir o sistema de equações nodais modificadas por inspeção da topologia do circuito, em que os capacitores são discretizados pelo algoritmo de Gear de segunda ordem.

Para os demais algoritmos de Gear as estampas são apenas

apresentadas, pois sua dedução segue procedimento similar ao descrito para o algoritmo de segunda ordem. Novamente, visando simplificar a notação serão adotadas constantes auxiliares.

GEAR DE TERCEIRA ORDEM

O algoritmo de Gear de terceira ordem considera as tensões sobre o capacitor em três instantes anteriores para atualizar a fonte de corrente controlada do modelo associado. Caso a corrente no capacitor não seja solicitada obtém-se a seguinte estampa para o algoritmo de Gear de terceira ordem:

$$\begin{matrix} & V_i & V_j \end{matrix}$$
$$\begin{bmatrix} g_a & -g_a \\ -g_a & g_a \end{bmatrix} \begin{bmatrix} V_i \\ V_j \end{bmatrix}_{n+1} = \begin{bmatrix} g_o & -g_o \\ -g_o & g_o \end{bmatrix} \begin{bmatrix} V_i \\ V_j \end{bmatrix}_{n} - \begin{bmatrix} g_1 & -g_1 \\ -g_1 & g_1 \end{bmatrix} \begin{bmatrix} V_i \\ V_j \end{bmatrix}_{n-1}$$
$$+ \begin{bmatrix} g_2 & -g_2 \\ -g_2 & g_2 \end{bmatrix} \begin{bmatrix} V_i \\ V_j \end{bmatrix}_{n-2}$$

Fig.3.13 - Estampa associada ao modelo discreto do capacitor para o algoritmo de Gear de Terceira Ordem na parte nodal

caso a corrente seja solicitada como variável de saída, obtém-se:

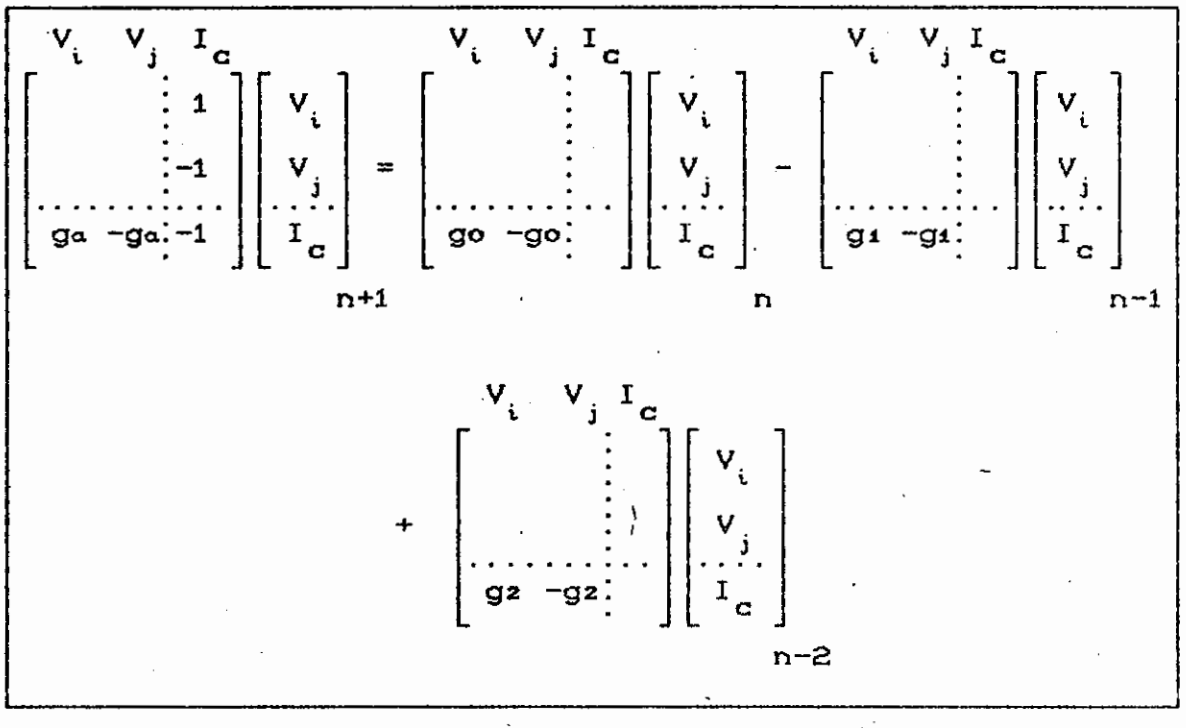


Fig. 3.14 - Estampa associada ao modelo discreto do capacitor para o algoritmo de Gear de Terceira Ordem na parte híbrida

$$\text{onde} \quad g_a = \frac{11C}{6h} \; ; \quad g_o = \frac{3C}{h} \; ; \quad g_1 = \frac{3C}{2h} \; ; \quad g_2 = \frac{C}{3h} \qquad (3.19)$$

As estampas das figuras 3.13 e 3.14 permitem construir por inspeção as matrizes da formulação nodal modificada de um circuito em que as equações diferenciais que descrevem o capacitor e o indutor tenham sido discretizadas pelo algoritmo de Gear de terceira ordem. Como nos casos anteriores, se o capacitor estiver aterrado, as linhas correspondentes ao nó de referência não aparecem nas estampas.

O algoritmo de Gear de terceira ordem tem uma região de estabilidade numérica mais restrita que os algoritmos de segunda ordem (Trapezoidal e Gear). Esta situação foi verificada experimentalmente para algumas simulações de circuitos simples, formados por um indutor, um capacitor e um

resistor. Quando estas simulações utilizaram o algoritmo de Gear de terceira ordem, excitados em freqüências próximas à freqüência natural, resultaram em resposta instável. Por sua vez o mesmo circuito discretizado por algoritmos de ordem menor apresentou a resposta esperada, conforme será visto mais adiante, no Capítulo 6.

GEAR DE QUARTA ORDEM

O algoritmo de Gear de quarta ordem para a discretização das equações que descrevem o capacitor fornecem (sem corrente solicitada):

$$\begin{matrix} V_i & V_j \end{matrix}$$
$$\begin{bmatrix} g_a & -g_a \\ -g_a & g_a \end{bmatrix}\begin{bmatrix} V_i \\ V_j \end{bmatrix}_{n+1} = \begin{matrix} V_i & V_j \end{matrix}\begin{bmatrix} g_0 & -g_0 \\ -g_0 & g_0 \end{bmatrix}\begin{bmatrix} V_i \\ V_j \end{bmatrix}_{n} - \begin{matrix} V_i & V_j \end{matrix}\begin{bmatrix} g_1 & -g_1 \\ -g_1 & g_1 \end{bmatrix}\begin{bmatrix} V_i \\ V_j \end{bmatrix}_{n-1}$$
$$+ \begin{matrix} V_i & V_j \end{matrix}\begin{bmatrix} g_2 & -g_2 \\ -g_2 & g_2 \end{bmatrix}\begin{bmatrix} V_i \\ V_j \end{bmatrix}_{n-2} - \begin{matrix} V_i & V_j \end{matrix}\begin{bmatrix} g_3 & -g_3 \\ -g_3 & g_3 \end{bmatrix}\begin{bmatrix} V_i \\ V_j \end{bmatrix}_{n-3}$$

Fig.3.15 - Estampa associada ao modelo discreto do capacitor para o algoritmo de Gear de Quarta Ordem na parte nodal

se a corrente do capacitor é solicitada como variável de saída, a relação constitutiva do capacitor discretizado aparece na parte híbrida da matriz nodal modificada:

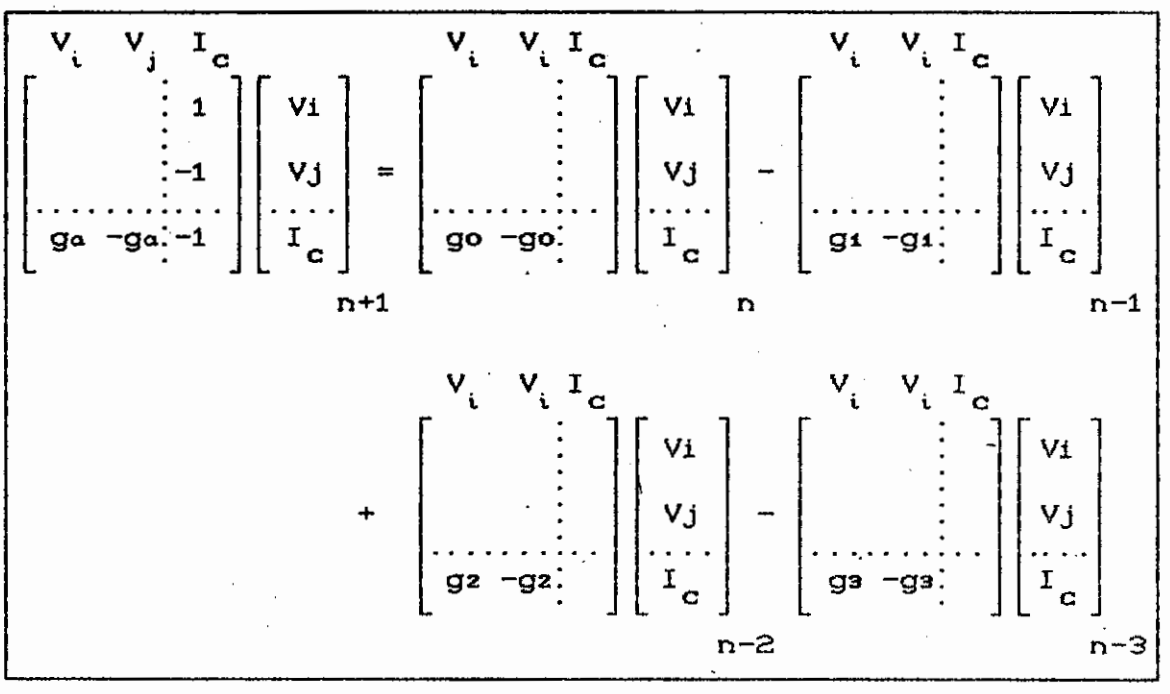


Fig.3.16 - Estampa associada ao modelo discreto do capacitor para o algoritmo de Gear de Quarta Ordem na parte híbrida

onde $g_a = \frac{25C}{12h}$; $g_o = \frac{4C}{h}$; $g_1 = \frac{3C}{h}$; $g_2 = \frac{4C}{3h}$; $g_3 = \frac{C}{4h}$

(3.20)

Nota-se que para o algoritmo de Gear de quarta ordem são necessárias quatro matrizes auxiliares para descrever a contribuição da fonte de corrente controlada pelas tensões em instantes anteriores.

O algoritmo de Gear de quarta ordem possue um erro da ordem de h^5, apresenta entretanto a menor região de estabilidade dentre os algoritmos de integração apresentados [1].

3.4.3 - Modelos Discretos Associados ao Indutor para os Algoritmos de Gear de Segunda a Quarta Ordem

A obtenção dos modelos discretos associados ao indutor para os algoritmos de Gear segue procedimento similar aos apresentados anteriormente. Aplicando-se os algoritmos de Gear (tabela 3.1) à equação diferencial que descreve o indutor linear obtém-se a seguinte tabela de relações constitutivas:

1	$i_{n+1} = \frac{h}{L} v_{n+1} + \left(i_n \right)$
2	$i_{n+1} = \frac{2h}{3L} v_{n+1} + \left(\frac{4}{3} i_n - \frac{1}{3} i_{n-1} \right)$
3	$i_{n+1} = \frac{6h}{11L} v_{n+1} + \left(\frac{18}{11} i_n - \frac{9}{11} i_{n-1} + \frac{2}{11} i_{n-2} \right)$
4	$i_{n+1} = \frac{12h}{25L} v_{n+1} + \left(\frac{48}{25} i_n - \frac{36}{25} i_{n-1} + \frac{16}{25} i_{n-2} - \frac{3}{25} i_{n-3} \right)$

Tabela 3.3 - Relações Constitutivas do Capacitor discretizado pelos algoritmos de Gear até quarta ordem

Para cada relação constitutiva pode ser deduzido uma *porta linear equivalente*. Na fig. 3.17 é vista a *porta linear equivalente* para a discretização do indutor pelo algoritmo de Gear de segunda ordem.

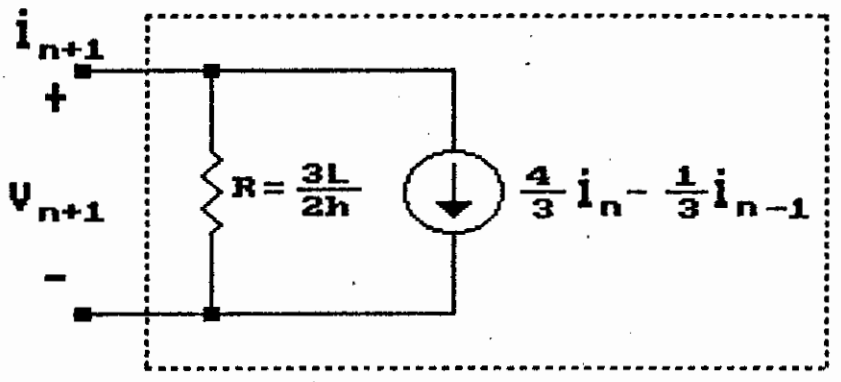


Fig. 3.17 Modelo Discreto Associado com o algoritmo de Gear de Segunda Ordem para o Indutor Linear.

Este modelo discreto permite substituir cada indutor de um circuito por um resistor e uma fonte de corrente, sem alterar a topologia do circuito, em termos de nós. O circuito resultante é puramente resistivo e reduz a análise transitória de um circuito à análise DC de um circuito resistivo.

3.4.4 - Estampas associadas aos Algoritmos de Gear de Segunda a Quarta Ordem para o Indutor

A seguir apresentam-se as estampas correspondentes aos modelos discretos associados ao indutor para os algoritmos de Gear de segunda a quarta ordem. Nota-se que apenas estampas que contém a corrente do indutor estão disponíveis por este entrar na parte híbrida da matriz nodal modificada.

Gear de Segunda Ordem

O algoritmo de Gear de segunda ordem aplicado ao indutor fornece a seguinte estampa:

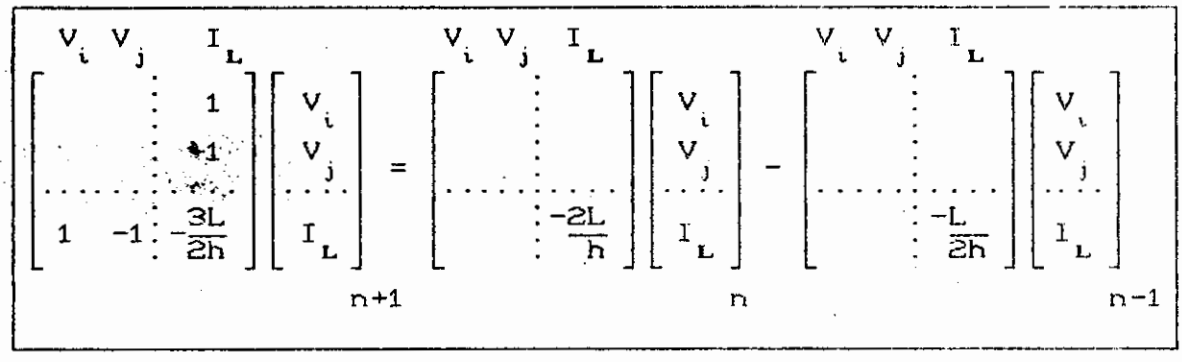


Fig. 3.18 - Estampa associada ao modelo discreto do indutor para o algoritmo de Gear de Segunda Ordem

A Fig. 3.18 mostra que as "matrizes" auxiliares da estampa que descreve a contribuição do indutor à formulação nodal modificada contém apenas um elemento. A primeira "matriz" desta estampa terá apenas três elementos se o indutor estiver ligado ao nó de referência.

GEAR DE TERCEIRA ORDEM

O algoritmo de Gear de terceira ordem aplicado ao indutor fornece a seguinte estampa:

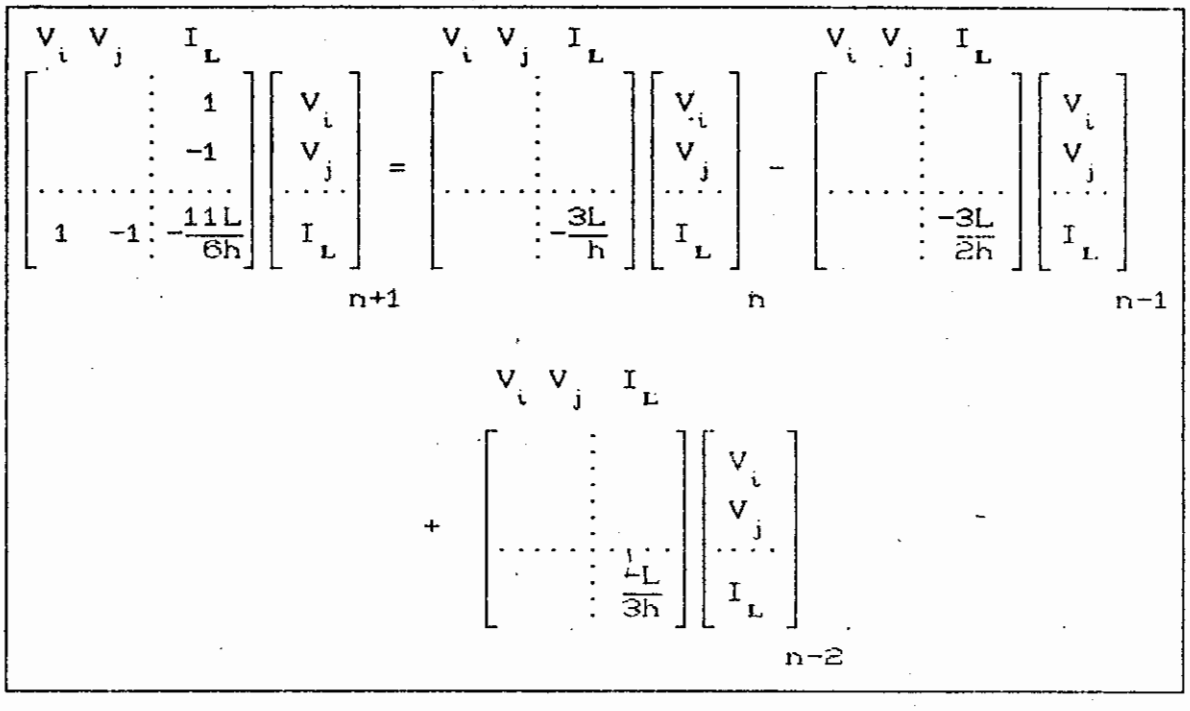


Fig.3.19 - Estampa associada ao modelo discreto do indutor para o algoritmo de Gear de Terceira Ordem

A Fig. 3.19 mostra que são necessárias três matrizes auxiliares para que as fontes de corrente dos modelos discretos associados ao indutor para o algoritmo de Gear de terceira ordem sejam atualizadas.

GEAR DE QUARTA ORDEM

O algoritmo de Gear de quarta ordem aplicado ao indutor fornece a estampa apresentada na Fig. 3.20. Nota-se que são necessárias quatro matrizes auxiliares para representar a atualização da fonte de corrente do modelo associado.

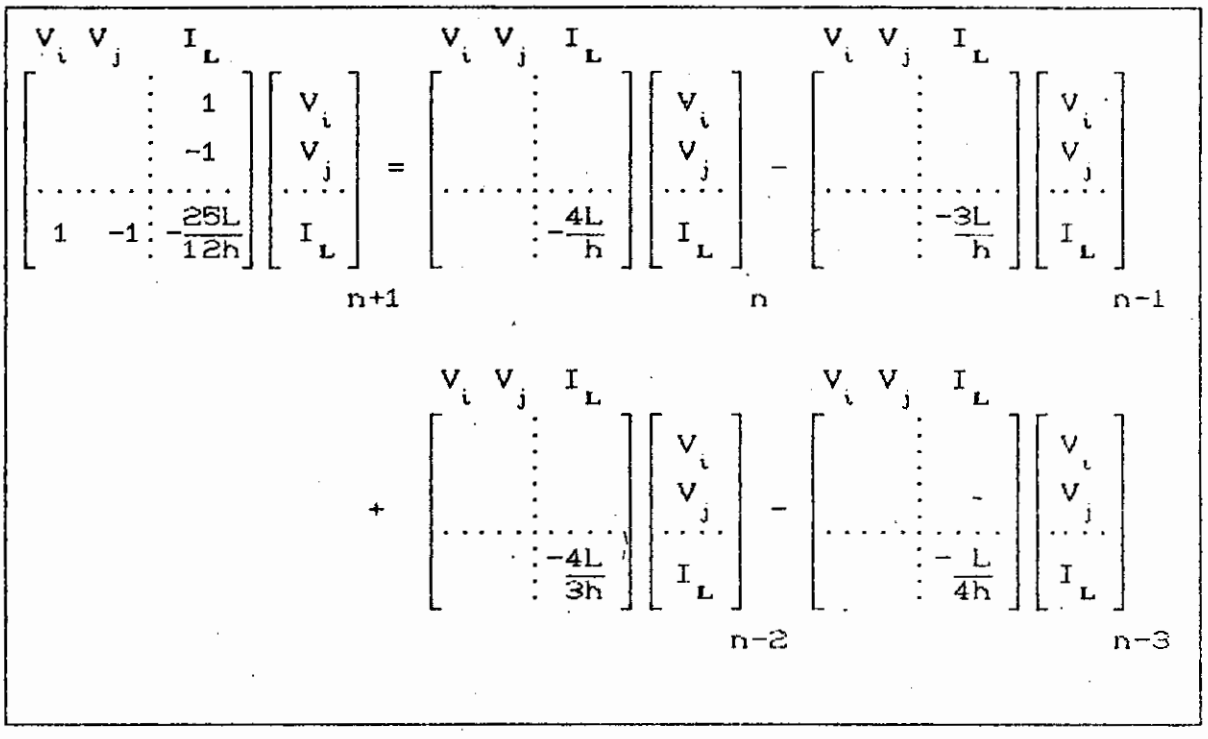


Fig.3.20 - Estampa associada ao modelo discreto do indutor para o algoritmo de Gear de Quarta Ordem

Conforme pode ser visto por todas as estampas apresentadas até aqui, a forma como as estampas são construídas obedece um procedimento simples. A construção de estampas para algoritmos de integração de ordem mais elevada pode ser feita acrescentando-se "matrizes" auxiliares às estampas, e atribuindo-se valores convenientes aos elementos das "matrizes".

Com o intuito de esclarecer a aplicação das estampas dos modelos discretizados, apresentam-se a seguir alguns exemplos de circuitos com as equações matriciais que os descrevem para propósito de simulação.

Ao final deste capítulo tem-se tabelas que resumem as estampas das contribuições dos modelos associados ao capacitor e indutor. Estas tabelas podem ser úteis para uma referência rápida.

3.5 -EXEMPLOS

Nesta seção apresentam-se alguns exemplos da formulação de circuitos nodal modificada em que os capacitores e indutores são substituídos pelos seus modelos discretos associados. É ilustrada a utilização das estampas para a construção direta do sistema de equações.

Inicialmente vê-se a equação matricial que descreve o exemplo da fig. 2.23, repetido aqui por conveniência na Fig. 3.21, discretizado pelo algoritmo de Euler Regressivo.

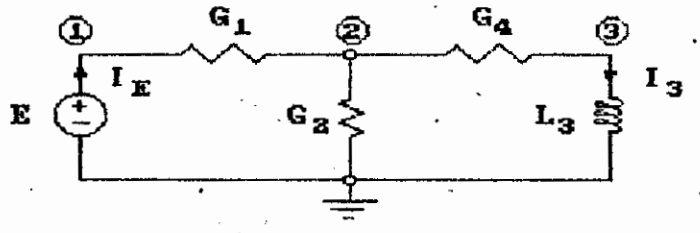


Fig. 3.21 - Circuito exemplo contendo uma fonte de tensão e um indutor

$$\begin{bmatrix} G_1 & -G_1 & 0 & -1 & 0 \\ -G_1 & G_1+G_2+G_4 & -G_4 & 0 & 0 \\ 0 & -G_4 & G_4 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \frac{-L_3}{h} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ I_E \\ I_3 \end{bmatrix}_{n+1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{-L_3}{h} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ I_E \\ I_3 \end{bmatrix}_{n} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ E \\ 0 \end{bmatrix}_{n+1} \quad (3.21)$$

A equação matricial 3.21 mostra que o circuito com três nós é descrito por um sistema de quinta ordem. A fonte independente de tensão e o indutor aparecem na parte híbrida. Conforme pode ser visto, estes dois componentes por estarem conectados ao nó de referência, aparecem em 3.21 com uma estampa simplificada, os elementos correspondentes ao nó de referência não estão presentes.

O segundo exemplo (correspondente à Fig. 2.24 - repetido na Fig. 3.22) ilustra a formulação nodal modificada para um circuito contendo algumas fontes controladas e um capacitor discretizado pelo algoritmo de Euler Regressivo.

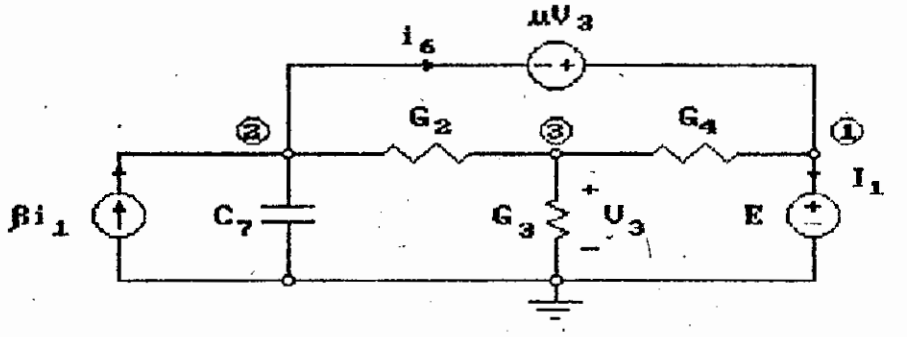


Fig. 3.22 - Circuito exemplo contendo fontes controladas e um capacitor

$$\begin{bmatrix} G_4 & 0 & -G_4 & 1 & -1 \\ 0 & G_2+\frac{C_7}{h} & -G_2 & -\beta & 1 \\ G_4 & -G_2 & G_4+G_3+G_2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & -\mu & 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ I_1 \\ I_6 \end{bmatrix}_{n+1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{C_7}{h} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ I_1 \\ I_6 \end{bmatrix}_{n} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ E \\ 0 \end{bmatrix}_{n+1}$$

(3.22)

Os dois primeiros exemplos mostram que para circuitos que contêm poucos componentes dinâmicos a matriz que descreve a contribuição da fonte de corrente do modelo discreto associado é extremamente esparsa[4]. No entanto, a matriz de contribuição pode inclusive ser bastante densa, como no exemplo fig. 3.23 a seguir:

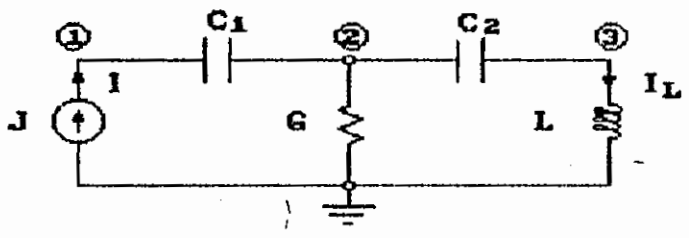


Fig. 3.23 - Circuito exemplo com dois capacitores e e um indutor

Quando o circuito exemplo da Fig. 3.23 é discretizado pelo algoritmo de Euler Regressivo obtém-se:

$$\begin{bmatrix} \frac{C_1}{h} & -\frac{C_1}{h} & 0 & 0 \\ -\frac{C_1}{h} & \frac{C_1}{h}+\frac{C_2}{h}+G & -\frac{C_2}{h} & 0 \\ 0 & -\frac{C_2}{h} & \frac{C_2}{h} & 1 \\ 0 & 0 & 1 & -\frac{L}{h} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ I_L \end{bmatrix}_{n+1} = \begin{bmatrix} \frac{C_1}{h} & -\frac{C_1}{h} & 0 & 0 \\ -\frac{C_1}{h} & \frac{C_1}{h}+\frac{C_2}{h} & -\frac{C_2}{h} & 0 \\ 0 & -\frac{C_2}{h} & \frac{C_2}{h} & 0 \\ 0 & 0 & 0 & -\frac{L}{h} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ I_L \end{bmatrix}_{n} + \begin{bmatrix} I \\ 0 \\ 0 \\ 0 \end{bmatrix}_{n+1} \quad (3.23)$$

---

[4] Considera-se uma matriz esparsa quando apenas uma pequena quantidade de seus elementos (digamos menos de 90%) é diferente de zero.

A fonte de corrente deste circuito, por estar conectada ao nó de referência, aparece em apenas uma posição no vetor de fontes. O seu valor contribui para a primeira equação do sistema de equações 3.23, que corresponde à aplicação da lei das correntes de Kirchhoff ao nó "1".

A equação 3.23 mostra que para o algoritmo de Euler Regressivo apenas o indutor aparece na parte híbrida. A seguir (equação 3.24) mostra-se o sistema de equações obtido quando o algoritmo trapezoidal é aplicado ao circuito exemplo da Fig. 3.23. Nota-se que para este método de discretização das equações diferenciais que descrevem os capacitores e o indutor, tanto o indutor como os capacitores aparecem na parte híbrida do sistema de equações.

$$\left[\begin{array}{ccc:ccc} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & G & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ \hdashline \frac{2C_1}{h} & \frac{-2C_1}{h} & 0 & -1 & 0 & 0 \\ 0 & \frac{2C_2}{h} & \frac{-2C_2}{h} & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & \frac{-2L}{h} \end{array}\right] \left[\begin{array}{c} V_1 \\ V_2 \\ V_3 \\ \hdashline I_1 \\ I_2 \\ I_L \end{array}\right]_{n+1} = \left[\begin{array}{ccc:ccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline \frac{2C_1}{h} & \frac{-2C_1}{h} & 0 & 1 & 0 & 0 \\ 0 & \frac{2C_2}{h} & \frac{-2C_2}{h} & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & \frac{-2L}{h} \end{array}\right] \left[\begin{array}{c} V_1 \\ V_2 \\ V_3 \\ \hdashline I_1 \\ I_2 \\ I_L \end{array}\right]_{n} + \left[\begin{array}{c} I \\ 0 \\ 0 \\ \hdashline 0 \\ 0 \\ 0 \end{array}\right]_{n+1}$$

(3.24)

Para o mesmo circuito da Fig. 3.23 discretizado pelo algoritmo de Gear de Segunda Ordem obtém-se a equação matricial 3.25:

$$\begin{bmatrix} \frac{3C_1}{2h} & \frac{-3C_1}{2h} & 0 & 0 \\ \frac{-3C_1}{2h} & \frac{3C_1}{2h}+\frac{3C_2}{2h}+G & \frac{-3C_2}{2h} & 0 \\ 0 & \frac{-3C_2}{2h} & \frac{3C_2}{2h} & 1 \\ 0 & 0 & 1 & \frac{-3L}{2h} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ I_L \end{bmatrix}_{n+1} = \begin{bmatrix} \frac{2C_1}{h} & \frac{-2C_1}{h} & 0 & 0 \\ \frac{-2C_1}{h} & \frac{2C_1}{h}+\frac{2C_2}{h} & \frac{-2C_2}{h} & 0 \\ 0 & \frac{-2C_2}{h} & \frac{2C_2}{h} & 0 \\ 0 & 0 & 0 & \frac{-2L}{h} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ I_L \end{bmatrix}_{n}$$

$$- \begin{bmatrix} \frac{C_1}{2h} & \frac{-C_1}{2h} & 0 & 0 \\ \frac{-C_1}{2h} & \frac{C_1}{2h}+\frac{C_2}{2h} & \frac{-C_2}{2h} & 0 \\ 0 & \frac{-C_2}{2h} & \frac{C_2}{2h} & 0 \\ 0 & 0 & 0 & \frac{-L}{2h} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ I_L \end{bmatrix}_{n-1} + \begin{bmatrix} I \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{n+1} \qquad (3.25)$$

Da eq. 3.25 nota-se que o algoritmo de Gear de segunda ordem fornece uma equação matricial com duas matrizes auxiliares. É necessário, no entanto apenas um sistema com quatro equações. A seguir far-se-á uma comparação, com relação ao esforço computacional dos dois métodos de segunda ordem apresentados.

3.6 - CONSIDERAÇÕES COMPUTACIONAIS

Observando-se os modelos de circuitos discretos associados ao capacitor e ao indutor vê-se que a substituição destes pelo seu modelo não altera a topologia do circuito. Apenas os parâmetros do circuito variam. Será visto adiante que quando a análise nodal modificada é utilizada na solução da rede resistiva, os modelos associados com os métodos de Gear de segunda ordem têm algumas vantagens computacionais sobre o método associado com o algoritmo trapezoidal. Isto principalmente porque a fonte de corrente da figura 3.4 depende apenas das *tensões* nos instantes anteriores, e no caso da Fig. 3.3 precisa-se tanto da tensão como da corrente no instante anterior.

Os dois métodos de segunda ordem podem ser implementados de maneira simples pela formulação nodal modificada. É pertinente então uma análise do esforço computacional associado à iteração de cada um dos métodos.

Na implementação do método trapezoidal para a formulação nodal modificada a corrente do capacitor é incluída na parte híbrida da matriz nodal modificada. Se não são utilizadas técnicas de matrizes esparsas existe uma grande desvantagem para o método trapezoidal, pois cada corrente de capacitor aumenta uma linha e uma coluna na matriz nodal modificada. Como o esforço computacional para resolução do sistema de equações é aproximadamente da ordem de $[n^3]$ vemos que uma grande quantidade de capacitores aumenta bastante a dimensão da matriz nodal modificada. A comparação deve portanto ser feita para o caso em que se utilizam técnicas de matrizes esparsas, onde o esforço computacional é em grande parte função do número de elementos não nulos da matriz.

Para o método de Gear de segunda ordem a atualização das

fontes de corrente é feita adicionando dois vetores de mesma dimensão da Matriz Nodal Modificada. Estes vetores são obtidos do produto de duas matrizes auxiliares[5] pelas tensões em t_n e t_{n-1} (conforme derivado nas seções anteriores).

Vê-se abaixo um esboço da topologia da Matriz Nodal Modificada para estes dois métodos de integração.

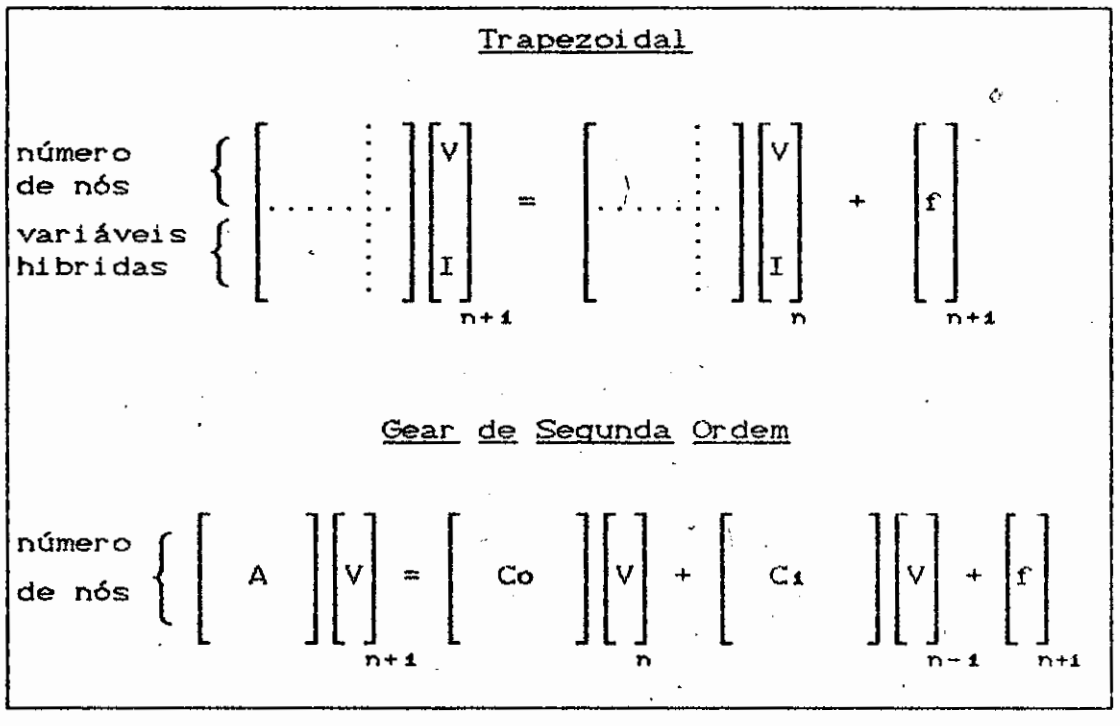


Fig. 3.24 - Esboço da topologia das equações nodais modificadas obtidas pelos métodos trapezoidal e de Gear de segunda ordem

A Fig. 3.24 mostra de maneira simplificada a topologia do sistema de equações, de um circuito que contêm apenas capacitores sem corrente solicitada, como componentes dinâmicos. O objetivo é mostrar que para o algoritmo trapezoidal tem-se uma matriz nodal modificada maior, no

---

[5] Entende-se por Matrizes Auxiliares as matrizes do lado direito do sistema de equações nodais modificadas, que permitem atualizar as fontes de corrente dos modelos discretos associados.

entanto apenas uma matriz auxiliar, e que para o algoritmo de Gear segunda ordem, a matriz nodal modificada é menor, requerendo, no entanto duas matrizes auxiliares.

Como a matriz do método trapezoidal é maior, a expectativa é de que eventualmente técnicas de matrizes esparsas poderiam fornecer um esforço computacional semelhante ao método de Gear de segunda ordem no processo de solução.

No entanto, o método trapezoidal tem desvantagem quanto ao número de elementos não nulos na Matriz Nodal Modificada em relação ao método de Gear de segunda ordem. São consideradas a seguir as duas configurações que o capacitor pode assumir:

- O capacitor com um terminal ligado ao nó terra -

Para o método de Gear de segunda ordem, quando a corrente do capacitor é solicitada, tem-se a inclusão de três elementos não nulos na Matriz Nodal Modificada. Se a corrente não precisa ser calculada tem-se apenas um elemento não nulo, com a vantagem adicional deste valor ser somado às outras condutâncias ligadas a este nó. Isto é, se houver várias condutâncias ligadas a um nó, a soma destas condutâncias aparece como apenas um elemento na matriz nodal modificada.

Para o método trapezoidal tem-se três elementos adicionados à matriz nodal modificada, e não existe a possibilidade de combinação de condutâncias.

- O capacitor flutuante[6] -

Para o método de Gear de segunda ordem tem-se cinco elementos não nulos quando a corrente do capacitor é

---

[6] Nenhum terminal ligado ao nó terra.

solicitada e quatro ou menos elementos não nulos se a corrente não é solicitada. Haverá menos de quatro elementos não nulos quando houver combinação de condutâncias ligadas a um mesmo nó.

Para o método de Trapezoidal tem-se cinco elementos não nulos, mesmo que a corrente no capacitor não seja solicitada.

Portanto, em termos de elementos não nulos da Matriz Nodal Modificada, o método trapezoidal é menos eficiente pois não propicia a combinação de admitâncias.

A desvantagem do método de Gear de segunda ordem consiste na necessidade de termos duas matrizes de atualização das fontes de corrente. Cada matriz é quadrada e tem a mesma dimensão da Matriz Nodal Modificada. Sem a utilização de técnicas de matrizes esparsas teríamos para cada passo de iteração n^2 produtos a mais. A topologia das matrizes auxiliares é no entanto idêntica e através de um artifício de programação pode-se tornar a atualização das fontes de corrente bastante eficiente.

A comparação do tempo de processamento para os dois algoritmos mostrou que de fato o algoritmo de Gear é mais rápido (aproximadamente 50 %) para circuitos contendo muitos capacitores. O fator predominante nesta diferença de velocidades é provavelmente a ordem do sistema de equações, sempre maior para o algoritmo trapezoidal.

3.7 - Generalização da Formulação Nodal Modificada para Modelos Discretos Associados

Os modelos discretos associados aos capacitores e indutores obtidos em seções anteriores podem ser obtidos para

qualquer método de integração implícito. Apresentar-se-á agora uma proposta de generalização desta técnica. O objetivo é estabelecer a forma das estampas que permitem a construção direta da matriz para um certo algoritmo de integração. Apenas capacitores e indutores contribuem com termos da forma $\frac{d}{dt}$ à matriz. Deste modo serão os únicos a terem sua estampa alterada em relação ao obtido no capítulo 2.

De acordo com a família do algoritmo de integração e a ordem deste a fonte de corrente do modelo associado é função de um certo número de variáveis em instantes de tempo anteriores. De uma forma genérica a estampa de um modelo discreto associado a um algoritmo de integração de ordem p+1 pode ser representada da seguinte forma:

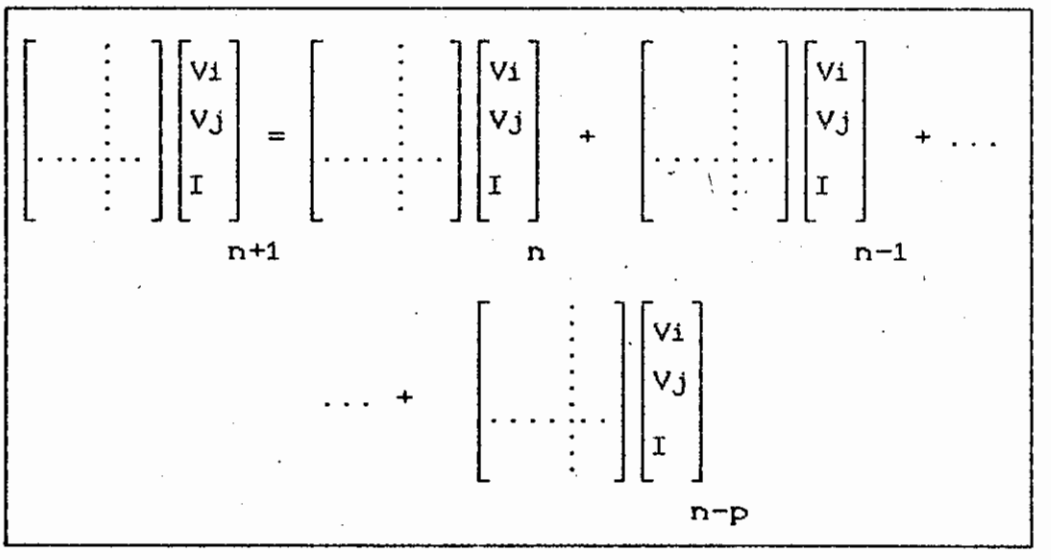


Fig. 3.25- Forma da estampa de um componente dinâmico discretizado por um algoritmo de integração multipassos implícito

onde na fig. 3.25:

Vi e Vj - tensões nos nós aos quais o componente está ligado

I - corrente no componente.

n+1 - índice do tempo para o qual as variáveis nodais modificadas serão resolvidas.

n, n-1,... n-p - índices do tempo para o qual as variáveis nodais modificadas já estão disponíveis (foram calculadas previamente).

Quando estampas desta forma são aplicadas a todos os componentes do circuito obter-se-á um sistema de equações nodais modificadas associado ao algoritmo de integração utilizado:

$$A.V_{n+1} = C_0.V_n + C_1.V_{n-1} + \ldots C_p.V_{n-p} + f \qquad (3.26)$$

onde

A - Matriz nodal modificada com componentes Discretizados

C_0, C_1, ... C_p - Matrizes auxiliares que apresentam as contribuições das fontes de corrente dos componentes discretizados para os instantes de tempo n, n-1, ... n-p.

V_{n+1} - Vetor das variáveis nodais modificadas a ser calculado (em $t = t_{n+1}$).

V_n, V_{n-1}, ... V_{n-p} - Vetor das variáveis nodais modificadas já calculadas (em t = n, n-1, ..., n-p).

A equação 3.26 apresenta a forma geral do sistema de equações nodal modificado discretizado pelos métodos de integração de passo multiplo implícitos. Este resultado permite visualizar a forma do sistema de equações quando outros algoritmos de integração são utilizados na discretização das equações diferenciais que descrevem o capacitor e o indutor linear.

Concluindo este capítulo, as tabelas 3.4 a 3.6 a seguir apresentam as estampas dos modelos discretos associados aos algoritmos de integração implementados.

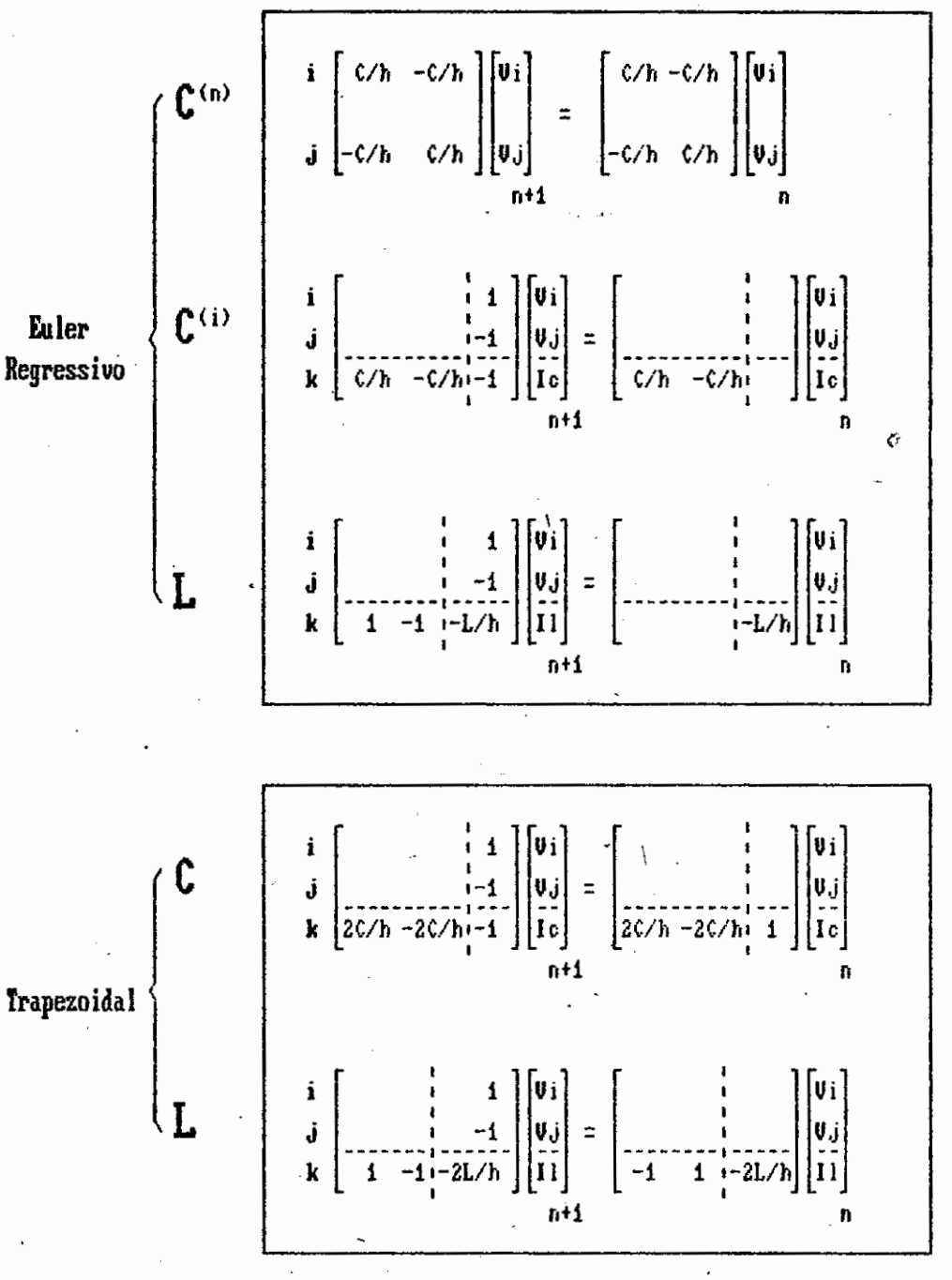


Tabela 3.4 - Estampas dos modelos discretos associados aos algoritmos de Euler Regressivo e Trapezoidal para o Capacitor e o Indutor

i , j - Linhas correspondentes aos nós aos quais o componente esta' ligado

k - Linha da corrente do componente

(n) Estampa para o capacitor sem que a corrente esteja disponível entre as variáveis nodais modificadas

(i) Estampa para o capacitor com a corrente incluída entre as variáveis nodais modificadas

Gear de Segunda Ordem

$$C^{(n)}\quad \begin{bmatrix} ga & -ga \\ -ga & ga \end{bmatrix}\begin{bmatrix} Vi \\ Vj \end{bmatrix}_{n+1} = \begin{bmatrix} g_0 & -g_0 \\ -g_0 & g_0 \end{bmatrix}\begin{bmatrix} Vi \\ Vj \end{bmatrix}_{n} - \begin{bmatrix} g_1 & -g_1 \\ -g_1 & g_1 \end{bmatrix}\begin{bmatrix} Vi \\ Vj \end{bmatrix}_{n-1}$$

$$C^{(i)}\quad \left[\begin{array}{cc:c} & & 1 \\ & & -1 \\ \hdashline ga & -ga & -1 \end{array}\right]\begin{bmatrix} Vi \\ Vj \\ Ic \end{bmatrix}_{n+1} = \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline g_0 & -g_0 & \end{array}\right]\begin{bmatrix} Vi \\ Vj \\ Ic \end{bmatrix}_{n} - \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline g_1 & -g_1 & \end{array}\right]\begin{bmatrix} Vi \\ Vj \\ Ic \end{bmatrix}_{n-1}$$

$$ga = \frac{3C}{2h}, \qquad g_0 = \frac{2C}{h}, \qquad g_1 = \frac{C}{2h}$$

$$L\quad \left[\begin{array}{cc:c} & & 1 \\ & & -1 \\ \hdashline 1 & -1 & -za \end{array}\right]\begin{bmatrix} Vi \\ Vj \\ Il \end{bmatrix}_{n+1} = \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline & & -z_0 \end{array}\right]\begin{bmatrix} Vi \\ Vj \\ Il \end{bmatrix}_{n} - \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline & & -z_1 \end{array}\right]\begin{bmatrix} Vi \\ Vj \\ Il \end{bmatrix}_{n-1}$$

$$za = \frac{3L}{2h}, \quad z_0 = \frac{2L}{h}, \quad z_1 = \frac{L}{2h}$$

Gear de Terceira Ordem

$$C^{(n)}\quad \begin{bmatrix} ga & -ga \\ -ga & ga \end{bmatrix}\begin{bmatrix} Vi \\ Vj \end{bmatrix}_{n+1} = \begin{bmatrix} g_0 & -g_0 \\ -g_0 & g_0 \end{bmatrix}\begin{bmatrix} Vi \\ Vj \end{bmatrix}_{n} - \begin{bmatrix} g_1 & -g_1 \\ -g_1 & g_1 \end{bmatrix}\begin{bmatrix} Vi \\ Vj \end{bmatrix}_{n-1} + \begin{bmatrix} g_2 & -g_2 \\ -g_2 & g_2 \end{bmatrix}\begin{bmatrix} Vi \\ Vj \end{bmatrix}_{n-2}$$

$$C^{(i)}\quad \left[\begin{array}{cc:c} & & 1 \\ & & -1 \\ \hdashline ga & -ga & -1 \end{array}\right]\begin{bmatrix} Vi \\ Vj \\ Ic \end{bmatrix}_{n+1} = \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline g_0 & -g_0 & \end{array}\right]\begin{bmatrix} Vi \\ Vj \\ Ic \end{bmatrix}_{n} - \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline g_1 & -g_1 & \end{array}\right]\begin{bmatrix} Vi \\ Vj \\ Ic \end{bmatrix}_{n-1} + \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline g_2 & -g_2 & \end{array}\right]\begin{bmatrix} Vi \\ Vj \\ Ic \end{bmatrix}_{n-2}$$

$$ga = \frac{11C}{6h}, \quad g_0 = \frac{3C}{h}, \quad g_1 = \frac{3C}{2h}, \quad g_2 = \frac{C}{3h}$$

$$L\quad \left[\begin{array}{cc:c} & & 1 \\ & & -1 \\ \hdashline 1 & -1 & -za \end{array}\right]\begin{bmatrix} Vi \\ Vj \\ Il \end{bmatrix}_{n+1} = \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline & & -z_0 \end{array}\right]\begin{bmatrix} Vi \\ Vj \\ Il \end{bmatrix}_{n} - \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline & & -z_1 \end{array}\right]\begin{bmatrix} Vi \\ Vj \\ Il \end{bmatrix}_{n-1} + \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline & & -z_2 \end{array}\right]\begin{bmatrix} Vi \\ Vj \\ Il \end{bmatrix}_{n-1}$$

$$za = \frac{11L}{6h}, \quad z_0 = \frac{3L}{h}, \quad z_1 = \frac{3L}{2h}, \quad z_2 = \frac{L}{3h}$$

Tabela 3.5 - Estampas dos modelos discretos associados aos algoritmos de Gear de segunda e terceira ordem

(n) - Nao inclui a corrente nas variáveis nodais modificadas

(i) - Inclui a corrente nas variáveis nodais modificadas

C$^{(n)}$

$$\begin{bmatrix} g_a & -g_a \\ -g_a & g_a \end{bmatrix}\begin{bmatrix} V_i \\ V_j \end{bmatrix}_{n+1} = \begin{bmatrix} g_0 & -g_0 \\ -g_0 & g_0 \end{bmatrix}\begin{bmatrix} V_i \\ V_j \end{bmatrix}_{n} - \begin{bmatrix} g_1 & -g_1 \\ -g_1 & g_1 \end{bmatrix}\begin{bmatrix} V_i \\ V_j \end{bmatrix}_{n-1} + \begin{bmatrix} g_2 & -g_2 \\ -g_2 & g_2 \end{bmatrix}\begin{bmatrix} V_i \\ V_j \end{bmatrix}_{n-2} - \begin{bmatrix} g_3 & -g_3 \\ -g_3 & g_3 \end{bmatrix}\begin{bmatrix} V_i \\ V_j \end{bmatrix}_{n-3}$$

C$^{(i)}$

$$\left[\begin{array}{cc:c} & & 1 \\ & & -1 \\ \hdashline g_a & -g_a & -1 \end{array}\right]\begin{bmatrix} V_i \\ V_j \\ I_c \end{bmatrix}_{n+1} = \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline g_0 & -g_0 & \end{array}\right]\begin{bmatrix} V_i \\ V_j \\ I_c \end{bmatrix}_{n} - \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline g_1 & -g_1 & \end{array}\right]\begin{bmatrix} V_i \\ V_j \\ I_c \end{bmatrix}_{n-1} + \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline g_2 & -g_2 & \end{array}\right]\begin{bmatrix} V_i \\ V_j \\ I_c \end{bmatrix}_{n-2} - \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline g_3 & -g_3 & \end{array}\right]\begin{bmatrix} V_i \\ V_j \\ I_c \end{bmatrix}_{n-3}$$

L

$$\left[\begin{array}{cc:c} & & 1 \\ & & -1 \\ \hdashline 1 & -1 & -z_a \end{array}\right]\begin{bmatrix} V_i \\ V_j \\ I_l \end{bmatrix}_{n+1} = \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline & & -z_0 \end{array}\right]\begin{bmatrix} V_i \\ V_j \\ I_l \end{bmatrix}_{n} - \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline & & -z_1 \end{array}\right]\begin{bmatrix} V_i \\ V_j \\ I_l \end{bmatrix}_{n-1} + \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline & & -z_2 \end{array}\right]\begin{bmatrix} V_i \\ V_j \\ I_l \end{bmatrix}_{n-1} - \left[\begin{array}{cc:c} & & \\ & & \\ \hdashline & & -z_3 \end{array}\right]\begin{bmatrix} V_i \\ V_j \\ I_l \end{bmatrix}_{n-3}$$

Para o Capacitor: $g_a = \frac{25C}{12h}$, $g_0 = \frac{4C}{h}$, $g_1 = \frac{3C}{h}$, $g_2 = \frac{4C}{3h}$, $g_3 = \frac{C}{4h}$

Para o Indutor: $z_a = \frac{25L}{12h}$, $z_0 = \frac{4L}{h}$, $z_1 = \frac{3L}{h}$, $z_2 = \frac{4L}{3h}$, $z_3 = \frac{L}{4h}$

Tabela 3.6 - Estampas dos modelos discretos associados aos algoritmos de Gear de quarta ordem

(n) - Não inclui a corrente nas variáveis nodais modificadas

(i) - Inclui a corrente nas variáveis nodais modificadas

4 - SOLUÇÃO DO SISTEMA DE EQUAÇÕES NODAIS MODIFICADAS

Neste capítulo procura-se apresentar uma forma computacionalmente eficiente para a solução do sistema de equações nodais modificadas obtido no capítulo anterior. Assim sendo, tecem-se algumas considerações sobre o sistema de equações obtido para os circuitos elétricos e da necessidade de utilização de técnicas de matrizes esparsas para a solução deste. O método da bifatoração é utilizado para a solução do sistema de equações. Também se discutem outras possíveis alternativas de solução.

O sistema de equações obtido pela formulação nodal modificada aplicada aos modelos discretos associados pode ser escrito matricialmente na forma

$$A v_{n+1} = b \qquad (4.1)$$

onde A - Matriz Nodal Modificada

v_{n+1} - variáveis nodais modificadas a obter (em t_{n+1})

b - vetor independente

Conforme visto no Capítulo 3 o vetor independente b é formado pela soma de alguns fatores, dependentes do método de discretização:

$$b = C_o . v_n + C_1 . v_{n-1} + \ldots + C_p . v_{n-p} + f \qquad (4.2)$$

onde C_o, C_1, ... C_p - Matrizes auxiliares de atualização dos modelos discretos

v_n, v_{n-1},... v_{n-p} - Variáveis nodais em instantes (n-m), m = 0, 1, ...,p, de integração anteriores

f - Vetor das fontes independentes

A equação 4.2 aplica-se aos métodos de discretização baseados em algoritmos de integração multipassos. O número de matrizes auxiliares de atualização necessárias depende da ordem do algoritmo de integração. O método de discretização baseado no algoritmo de integração de Euler regressivo, como foi visto, utiliza apenas a primeira matriz, C_o. As matrizes C_o, C_1, ... C_p permitem que as atualizações dos modelos discretos associados possam ser feitas a cada passo da iteração. Estas matrizes são em geral bastante esparsas, tornando a aplicação de técnicas de matrizes esparsas para sua representação muito importante no sentido de reduzir o esforço computacional e quantidade de memória utilizada.

A análise de circuitos utilizando simulação no domínio do tempo requer, portanto, a solução do conjunto de equações (4.1) para cada amostra dos sinais de entrada. Cada solução utiliza a mesma matriz de coeficientes, variando-se

entretanto, o vetor independente b. Tais sistemas podem ser resolvidos através de diversos métodos elementares (Eliminação Gaussiana [11], método de Crout[11], etc.). Outra possibilidade é a inversão direta da matriz A. Este procedimento requer no entanto n^2 posições de memória para armazenar a matriz e aproximadamente n^3 operações aritméticas para a solução de n equações lineares simultâneas. Mesmo que a matriz A seja bastante esparsa é possível que sua inversa seja uma matriz não esparsa, como é comum em circuitos. Desta forma este método é geralmente ineficiente para a solução de um sistema de ordem elevada [4].

A seguir introduzem-se conceitos, características e formas de implementações aplicadas à representação de circuitos por matrizes esparsas.

4.1 - Técnicas de Matrizes Esparsas para Solução do Sistema de Equações Nodais Modificadas

Em redes de transmissão de energia elétrica e projetos de circuitos integrados, é comum se trabalhar com circuitos modelados com centenas e até milhares de nós. As matrizes que descrevem matematicamente estes sistemas são bastante esparsas. Se nenhuma técnica é utilizada no sentido de aproveitar esta esparsidade tem-se severas limitações computacionais. Além do excessivo tempo de processamento, o armazenamento dos elementos da matriz A como uma matriz bidimensional n x n em memória principal passa a ser problemático em computadores de porte médio. Mesmo utilizando computadores de alto desempenho, é claro que a aplicação direta de métodos de "*matriz densa*"[7] para a solução deste

---

[7] Utiliza-se o termo "matriz densa" para descrever uma matriz em que todos os seus elementos são armazenados no computador, independente de seus valores, conforme Morozovsky Filho, [3].

sistema de equações simultâneas é pouco eficiente, quando não torna a solução impossível.

O fato é que os sistemas de equações que são obtidos na análise de circuitos (sistemas de potência e circuitos eletrônicos) são usualmente caracterizados por equações simultâneas tendo uma matriz de coeficientes esparsa. Uma matriz quadrada A é considerada *esparsa* se apenas uma pequena porcentagem (supõe-se menos de 30%) dos seus elementos é não nula [1]. Em circuitos eletrônicos, a matriz de admitâncias nodal **Yn** para um circuito típico de 10 nós geralmente contêm menos de 50% de elementos não nulos. Para um circuito típico de 100 nós a matriz **Yn** contêm aproximadamente 5% de elementos não nulos. Nestes casos, o número de operações para obter a solução pode ser reduzido em muito do valor original para matrizes cheias (o qual é $n^3/3$), se um esquema apropriado é utilizado.

No apêndice B pode ser encontrado um exemplo comparativo do esforço computacional para a solução de dois sistemas, um de matriz densa e outro de matriz esparsa. Mostra-se que a esparsidade da matriz é explorada reduzindo-se em muito o esforço computacional.

A idéia básica por trás de toda técnica de matrizes esparsas é *reduzir o tempo de computação omitindo operações triviais*, que envolvam *elementos nulos*. O ponto central é a habilidade de evitar as operações triviais *completamente*. Quando se resolve um sistema manualmente apenas são verificados os elementos não nulos e não se efetuam operações que envolvam tais elementos. No caso de computador com arquitetura von Neumann a verificação, de um elemento ser ou não nulo, implica no acesso a uma posição de memória e na transferência de seu conteúdo para um dos registradores internos do processador. Esta operação consome tempo de

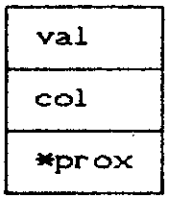
processador (aproximadamente o mesmo tempo de uma soma). Isto significa que a melhor técnica é aquela que evita completamente as operações de acesso a memória e comparação, para os elementos nulos. Desta forma uma técnica de matriz esparsa para implementação em computador deve ter as duas características seguintes:

1. Apenas elementos não nulos e a indexação necessária são armazenados.
2. Operações com elementos nulos não são efetuadas.

Apesar da redução do esforço computacional se constituir no objetivo principal, é importante ressaltar que o limite de memória de massa disponível pode ser o fator mandatório para a escolha da técnica de matriz esparsa a ser adotada para a solução de (4.1).

4.1.1 - Estruturas de Dados Dinâmicas para o Sistema de Equações Nodais Modificadas

Conforme mencionado as matrizes esparsas são convenientemente armazenadas guardando-se apenas os valores não nulos e informações de indexação. Alguns esquemas são propostos na literatura para armazenar matrizes esparsas (Morozowski Filho, 1981 [3]; Pissanetsky, 1984 [2]; Brameller et allii, 1976 [4]). Implementou-se neste trabalho uma estrutura de dados baseada em listas encadeadas que permite grande eficiência computacional. Como em alguns esquemas de armazenamento encontrados na literatura, a célula básica que armazena um elemento não nulo da matriz é da seguinte forma:



Onde val - valor do elemento não nulo
col - coluna que o elemento ocupa na matriz
*prox - ponteiro[8] para o próximo elemento não nulo

Alguns esquemas encontrados na literatura que utilizam esta célula propõem criar uma única lista que armazena todos os elementos não nulos da matriz e utilizar uma lista adicional com a informação de onde começa cada linha. Devido aos algoritmos de simulação propostos operarem intensamente sobre linhas propõe-se uma estrutura alternativa que apresenta grande simplicidade para o acesso às linhas da matriz esparsa:

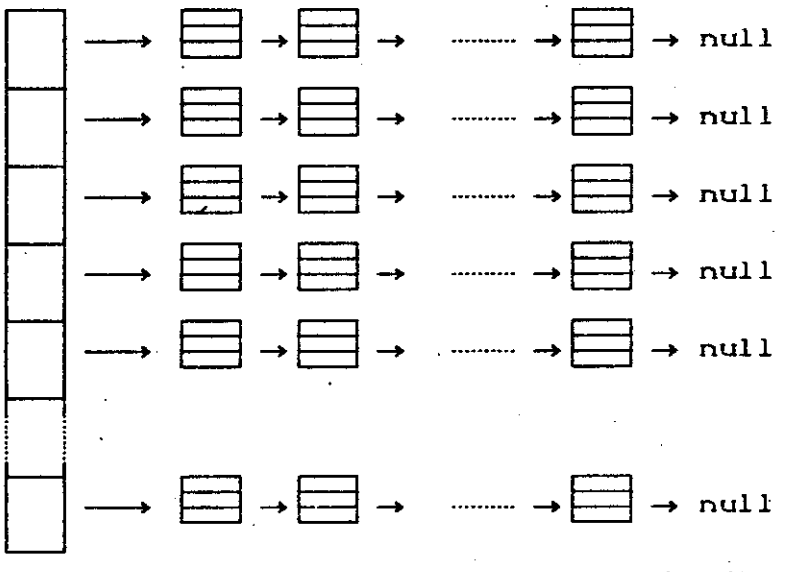


Fig. 4.1 - Estrutura de dados tipo "vetor de listas" para armazenar matrizes esparsas

[8] Ponteiro é um endereço de memória, com o qual o computador pode acessar uma estrutura de dados, geralmente alocada dinamicamente.

Na Fig. 4.1 um vetor armazena ponteiros (indicado pela seqüência de retângulos, à esquerda na figura), onde cada ponteiro aponta para uma lista de elementos não nulos. O primeiro ponteiro aponta para a lista de elementos não nulos da primeira linha, o segundo ponteiro aponta para a lista de elementos não nulos da segunda linha e assim por diante. Cada uma das listas é terminada pela constante pré-definida "null". Esta estrutura tem algumas vantagens em relação à lista única, pois a permutação de duas linhas consiste na simples permutação dos ponteiros no vetor de ponteiros. O acesso a qualquer linha é imediato, sem necessidade de consulta à listas auxiliares.

A seguir, na Fig. 4.3., ilustra-se a topologia da estrutura de dados correspondente à matriz nodal modificada do circuito da Fig. 3.23 (repetido na Fig. 4.2), discretizada pelo método trapezoidal (vide Capítulo 3). A matriz que aparece à esquerda na figura é apresentada apenas com os elementos não nulos assinalados por um "x". A cada elemento não nulo corresponde uma célula básica, conforme indicado na estrutura de dados esquematizada à direita na figura. Aparecem seis listas encadeadas, correspondentes às seis linhas da matriz.

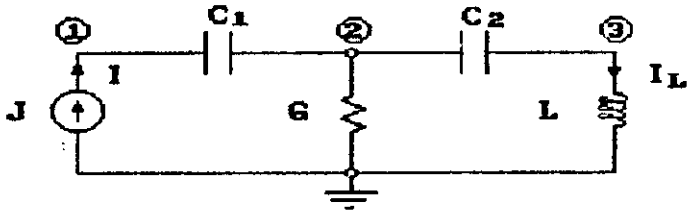


Fig. 4.2 - Circuito exemplo com capacitores e indutor

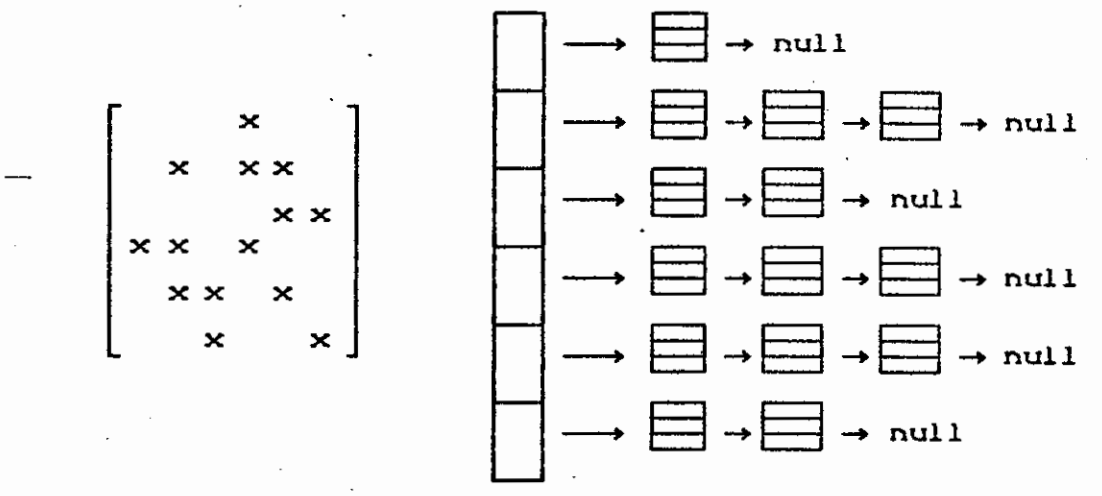


Fig. 4.3 - Exemplo de representação de uma matriz esparsa pela estrutura de vetor de listas.

Naturalmente a topologia da estrutura de dados para o circuito da Fig. 4.2 discretizado por outro algoritmo de integração seria diferente da Fig. 4.3, refletindo a posição de seus elementos não nulos.

As matrizes C_0, C_1,... C_p da equação (4.2) possuem topologias idênticas (elementos não nulos nas mesmas posições), por descreverem a atualização dos modelos discretos associados. Assim uma estrutura de dados semelhante à utilizada para a matriz nodal modificada A pode ser utilizada para armazenar as matrizes C_0, C_1,... C_p. Neste trabalho a discretização dos componentes dinâmicos utiliza algoritmos de integração multipassos que consideram no máximo até quatro instantes de integração anteriores (Gear de quarta ordem). Desta forma a estrutura foi limitada a quatro matrizes de atualização (p = 3), conforme pode ser visto na fig. 4.4 a seguir:

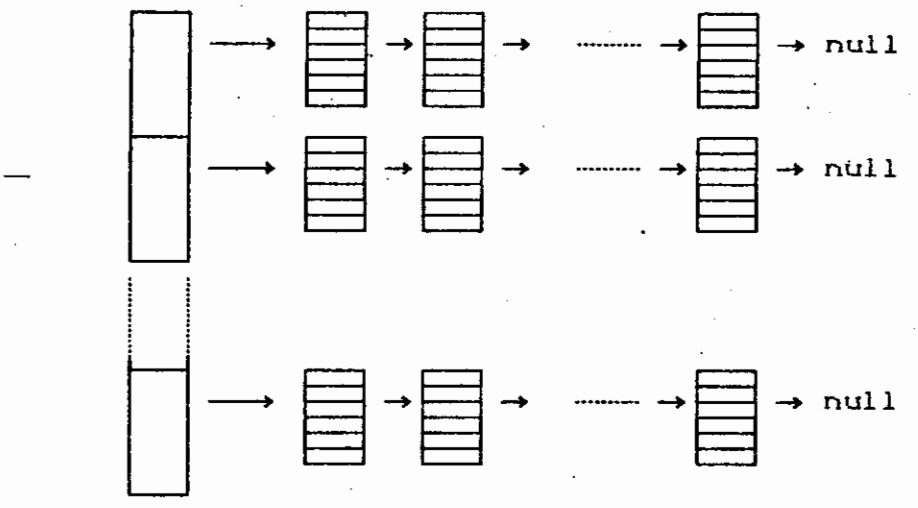
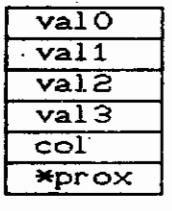


Fig 4.4 -Estrutura de dados para as matrizes Co a C3 por vetor de listas

Cada célula básica das listas das matrizes C contêm os seguintes dados



Onde val0 a val3 - Valores dos elementos não nulos em posições idênticas nas matrizes C_0 a C_3.

col - coluna que os elementos ocupam nas matrizes

*prox - ponteiro para o próximo elemento

A célula básica para as matrizes C é proposta com quatro campos para valores numéricos, atendendo assim a todos os algoritmos de discretização implementados (Euler Regressivo, Trapezoidal e Gear até quarta ordem).

A seguir (Fig. 4.5) ilustra-se para o circuito exemplo da Fig. 4.2 a forma com que o sistema de equações discretizado pelo algoritmo de Gear de terceira ordem é armazenado:

$$\begin{bmatrix} x & x & & \\ x & x & x & \\ & x & x & x \\ & & x & x \end{bmatrix}\begin{bmatrix}\\\\\\\end{bmatrix} = \begin{bmatrix} x & x & & \\ x & x & x & \\ & x & x & \\ & & & x \end{bmatrix}\begin{bmatrix}\\\\\\\end{bmatrix} + \begin{bmatrix} x & x & & \\ x & x & x & \\ & x & x & \\ & & & x \end{bmatrix}\begin{bmatrix}\\\\\\\end{bmatrix} + \begin{bmatrix} x & x & & \\ x & x & x & \\ & x & x & \\ & & & x \end{bmatrix}\begin{bmatrix}\\\\\\\end{bmatrix} + f$$

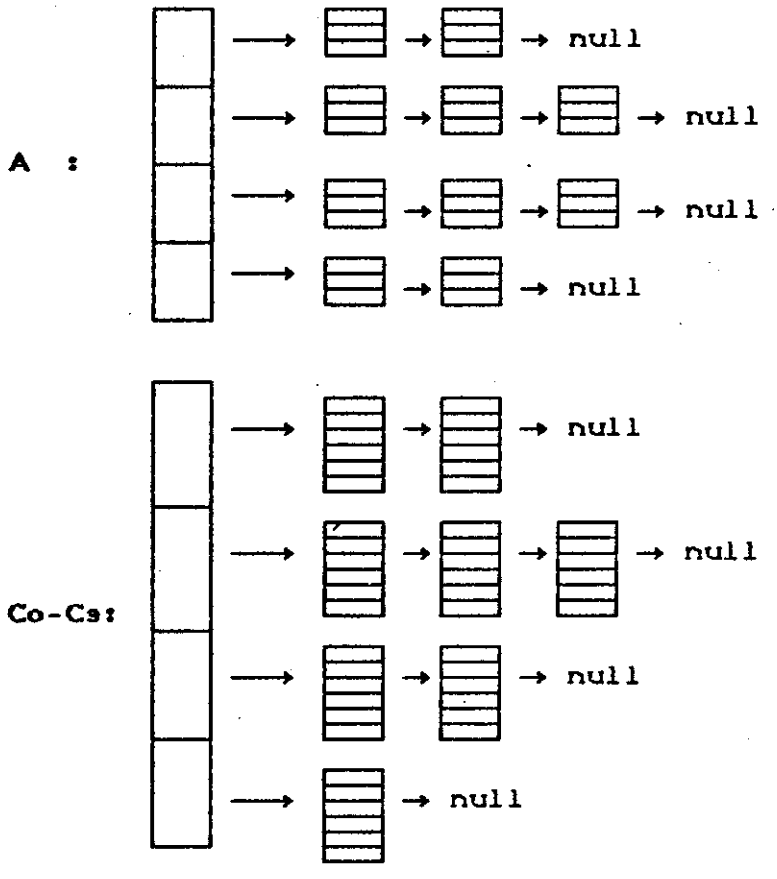


Fig. 4.5 - Armazenamento dinâmico do sistema de equações diferenciais discretizadas da Fig. 4.2

A topologia da estrutura de dados que armazena C_o-C_3 é devida aos dois capacitores "flutuantes" e ao indutor que

está aterrado. Nota-se que as matrizes C beneficiam-se sobremaneira das técnicas de matrizes esparsas.

4.2 - Esparsidade e Solução de Sistemas de Equações Através de Métodos de Fatoração.

Existem diferentes métodos para resolver um sistema de equações lineares. A eliminação Gaussiana e o método de Gauss-Jordan estão entre os mais populares no caso de matrizes densas. Por outro lado, os métodos de fatoração são melhores, no caso de matrizes esparsas, pois preservam a esparsidade da matriz original [4]. Esta característica é ainda mais importante quando a solução para diferentes valores do vetor independente é necessária.

Se uma equação do tipo **Ax = b** deve ser resolvida uma única vez, tanto o método da eliminação Gaussiana como os métodos de fatoração requerem $(n^3 + 3n^2 - n)/3$ multiplicações-divisões. A obtenção da matriz A^{-1} pela eliminação Gaussiana requer n^3 operações [1, pg 635], enquanto a obtenção das matrizes fator requerem $(n^3 - n)/3$ operações [2, pg 63]. Para se obter uma nova solução x para um novo b precisam-se n^2 operações, tanto utilizando $x = A^{-1}b$ como calculando x pelos métodos de fatoração [1, pg 635]. Desta forma, para o caso de *matrizes densas* não existe vantagem decisiva da fatoração sobre a eliminação Gaussiana, com exceção do fato de que a obtenção por A^{-1} requer mais operações do que a obtenção das matrizes fator.

Esta situação é muito diferente no caso de matrizes esparsas. Frequentemente *a inversa de uma matriz esparsa é uma matriz densa*, enquanto *as matrizes fator*[9] *preservam a esparsidade* [4]. Desta forma se um sistema de equações precisa ser resolvido diversas vezes para diferentes valores de b a solução por $x = A^{-1}b$ requer muito mais operações do que pelos métodos de fatoração.

A importante classe de circuitos na forma escada produz uma matriz de admitância nodal na forma tridiagonal, bastante esparsa. A topologia das matrizes inversa e das matrizes fator é mostrada no apêndice B, enfatizando o fato de que a solução pelos métodos de fatoração requer menor esforço computacional.

Verifica-se que o método da eliminação Gaussiana utiliza aproximadamente $n^3/3$ operações aritméticas para obter a solução de (4.1), requerendo entretanto um número indeterminado de posições de memória, que pode sér menor ou igual à quantidade exigida pela inversão direta. Esse método é portanto significativamente melhor que a inversão direta, necessitando porém uma forma sistematizada para permitir uma implementação eficiente em computador digital.

A sistematização pode ser conseguida através de métodos que consistem essencialmente na modificação da eliminação Gaussiana básica. Tais métodos geralmente são conhecidos como métodos de matriz fatorada. Estes utilizam a eliminação Gaussiana para obter a inversa da matriz A implicitamente, como o produto de diversas matrizes fator. Não reduzem em si a quantidade de memória necessária ou o número de operações

---

[9] Os métodos de fatoração fornecem "matrizes fator" com as quais é obtida a solução do sistema.

utilizadas na eliminação Gaussiana, entretanto, devido à sua forma sistematizada, levam a técnicas numéricas que, juntamente com estruturas de dados baseadas em listas encadeadas dinâmicas , permitem reduzir drasticamente tanto a quantidade de memória utilizada quanto o número de operações.

Existem diversos métodos e variantes de fatoração possíveis [4]. É importante notar que embora estes métodos pareçam complicados, a sua implementação é bastante simples, pois são apenas a *extensão sistemática da eliminação Gaussiana básica*. Na seção seguinte apresenta-se o método adotado neste trabalho.

4.2.1 - Solução de um sistema de equações pelo método da Bifatoração

Além do método da bifatoração utilizado neste trabalho, são conhecidos dois outros métodos de fatoração bem estabelecidos para a solução de grandes sistemas de equações lineares, que são a inversa em forma de produto e a fatoração triangular [4]. Apesar dos dois métodos utilizarem a fatoração, a forma de expressar os fatores é consideravelmente diferente. Zollenkopf combinou estas duas técnicas em 1971 e denominou o novo método de Bifatoração [15].

Este método é particularmente adequado para matrizes esparsas que tenham diagonal não nula e dominante, e que sejam simétricas ou então possuam simetria estrutural (elementos não nulos distintos em posições simétricas em relação à diagonal). Este é o caso típico de circuitos

elétricos, análise estrutural e sistemas de fluxo em fenômenos de transporte.

O método consiste na obtenção de 2n matrizes fator para um sistema de ordem n, de tal forma que o produto destas matrizes fator satisfaça a condição:

$$L^{(n)}L^{(n-1)}\ldots L^{(2)}L^{(1)}A\,R^{(1)}R^{(2)}\ldots R^{(n-1)}R^{(n)} = U \qquad (4.3)$$

Onde A = matriz original
L = matrizes fator do lado esquerdo (*Left*)
R = matrizes fator do lado direito (*Right*)
U = matriz Identidade de ordem n

Os fatores obtidos pela condição (4.3) permitem obter a inversa A^{-1} de forma implícita, em termos dos fatores, como será visto a seguir.

Pré-multiplicando (4.3) pelas inversas de $L^{(n)}$, $L^{(n-1)}\ldots L^{(2)}$ e $L^{(1)}$ consecutivamente obtém-se:

$$A\,R^{(1)}R^{(2)}\ldots R^{(n-1)}R^{(n)} = (L^{(1)})^{-1}(L^{(2)})^{-1}\ldots(L^{(n-1)})^{-1}(L^{(n)})^{-1} \qquad (4.4)$$

Pós-multiplicando (4.4) por $L^{(n)}, L^{(n-1)}\ldots L^{(2)}$ e $L^{(1)}$ consecutivamente resulta em:

$$A\,R^{(1)}R^{(2)}\ldots R^{(n-1)}R^{(n)}\,L^{(n)}L^{(n-1)}\ldots L^{(2)}\,L^{(1)} = U \qquad (4.5)$$

Finalmente pré-multiplicando (4.5) por A^{-1} obtém-se:

$$R^{(1)}R^{(2)}\ldots R^{(n-1)}R^{(n)}\,L^{(n)}L^{(n-1)}\ldots L^{(2)}\,L^{(1)} = A^{-1} \qquad (4.6)$$

A solução para o sistema de equações originais pode ser calculada da seguinte forma:

$$x = A^{-1}b$$

$$A^{-1} = R^{(1)}R^{(2)}\ldots R^{(n-1)}R^{(n)}\ L^{(n)}L^{(n-1)}\ldots L^{(2)}\ L^{(1)}$$

portanto:

$$x = R^{(1)}R^{(2)}\ldots R^{(n-1)}R^{(n)}\ L^{(n)}L^{(n-1)}\ldots L^{(2)}\ L^{(1)}\ b \qquad (4.7)$$

A obtenção de x pela equação (4.7) deve ser feita de trás para frente, isto é, o vetor coluna b é sucessivamente multiplicado pelas matrizes fator. A forma das matrizes R e L permite que a quantidade de multiplicações-divisões deste método seja menor que na alternativa de obter a solução multiplicando A^{-1} por b.

Apesar desta forma parecer bem mais complicada do que a aplicação direta de outros métodos, observa-se que a implementação da equação (4.7) é bastante simplificada pela utilização de técnicas de matrizes esparsas. Porém, antes serão introduzidos os algoritmos de cálculo das matrizes fator.

Para obter-se as matrizes fator L e R considera-se as seguintes matrizes intermediárias [4] :

$$
\begin{aligned}
A &= A^{(0)} \\
A^{(1)} &= L^{(1)}A^{(0)}R^{(1)} \\
A^{(2)} &= L^{(2)}A^{(1)}R^{(2)} \\
&\vdots \\
A^{(k)} &= L^{(k)}A^{(k-1)}R^{(k)} \\
&\vdots \\
A^{(n)} &= L^{(n)}A^{(n-1)}R^{(n)}
\end{aligned}
$$

Os sucessivos produtos internos triplos $L^{(k)}A^{(k-1)}R^{(k)}$ devem transformar a matriz original $A = A^{(0)}$ na matriz unitária U.

A produção das matrizes fator $L^{(k)}$ e $R^{(k)}$ e a redução da matriz de coeficientes é ilustrada no apêndice B, para um sistema de quarta ordem. As matrizes $L^{(k)}$ e $R^{(k)}$ no k-ésimo passo de redução que transformam $A^{(k-1)}$ em $A^{(k)}$ são dadas por [4]:

$$
L^{(k)} = \begin{bmatrix}
1 & & & & \\
& \ddots & & & \\
& & 1 & & \\
& & L_{kk}^{(k)} & & \\
& & L_{ik}^{(k)} & 1 & \\
0 & & \vdots & & \ddots \\
& & L_{nk}^{(k)} & & & 1
\end{bmatrix} \qquad (4.8)
$$

$$R^{(k)} = \begin{bmatrix} 1 & & & 0 \\ & \ddots & & \\ & & 1 & \\ & & & R_{kj}^{(k)} \cdots R_{kn}^{(k)} \\ & & & 1 \\ & 0 & & \ddots \\ & & & & 1 \end{bmatrix} \qquad (4.9)$$

$$A^{(k)} = \begin{bmatrix} 1 & & & 0 \\ & \ddots & & \\ & & 1 & \\ & & a_{k+1,k+1}^{(k)} & a_{k+1,k+2}^{(k)} \cdots a_{k+1,n}^{(k)} \\ & 0 & a_{k+2,k+1}^{(k)} & a_{k+2,k+2}^{(k)} \cdots a_{k+2,n}^{(k)} \\ & & \vdots & \\ & & a_{n,k+1}^{(k)} & a_{n,k+2}^{(k)} \cdots a_{n,n}^{(k)} \end{bmatrix} \qquad (4.10)$$

onde:

$$L_{kk}^{(k)} = \frac{1}{a_{kk}^{(k-1)}},$$

$$L_{ik}^{(k)} = -\frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}, \qquad (i = k+1, \ldots, n)$$

$$R_{kj}^{(k)} = -\frac{a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}}, \qquad (j = k+1,\ldots,n)$$

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - \frac{a_{ik}^{(k-1)}\, a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}}, \qquad \begin{matrix}(i = k+1,\ldots,n)\\ (j = k+1,\ldots,n)\end{matrix}$$

Caso a matriz A seja simétrica tem-se

$$a_{ik}^{(k-1)} = a_{ki}^{(k-1)}$$

Assim,

$$R_{ik}^{(k)} = L_{ki}^{(k)} \qquad (4.11)$$

A equação (4.11) indica que para o caso de matrizes simétricas, com exceção da diagonal, os elementos da k-ésima linha de $R^{(k)}$ são idênticos aos elementos da k-ésima coluna de $L^{(k)}$. Além disso, como os elementos da diagonal de $R^{(k)}$ são todos iguais a 1, e, como são conhecidos implicitamente, somente é necessário calcular os elementos de $L^{(k)}$. Desta forma o número de operações e a quantidade de memória necessária são reduzidos à metade. Este não é, em geral, o caso do sistema de equações nodais modificadas no qual não se pode garantir a simetria.

A obtenção do vetor x é feita multiplicando-se o vetor b seqüencialmente por 2n matrizes fator. No entanto, com exceção de uma linha e uma coluna, os fatores são todos matrizes identidade, e desta forma cada multiplicação individual torna-se bastante simples. A implementação em

programa só precisa considerar os elementos não nulos, pois estes já têm as suas posições conhecidas *a priori*.

A seguir são comentadas as diferenças entre os métodos de triangularização e bifatoração, justificando-se desta forma a escolha do último na implementação do simulador temporal de circuitos.

4.2.2 - COMPARAÇÃO ENTRE OS MÉTODOS DE TRIANGULARIZAÇÃO E DE BIFATORAÇÃO.

Os processos numéricos envolvidos na decomposição triangular e na bifatoração são análogos, pois são variantes da eliminação Gaussiana básica. A diferença está apenas na implementação específica da eliminação Gaussiana em cada caso. Desta forma, a quantidade de operações aritméticas necessárias para a fatoração de matrizes, considerando a mesma seqüência de eliminação de variáveis, é a mesma para os dois métodos. As matrizes fator de cada um dos métodos podem ser sobrepostas, se a diagonal principal unitária da matriz triangular superior (U) e das matrizes fator do lado direito (R) forem ignoradas, sendo conhecidas implicitamente.

Uma dificuldade na decomposição triangular é que o produto das matrizes fator produz a matriz de coeficientes original, e não a sua inversa. Por outro lado o produto das matrizes fator geradas pela bifatoração produzem a inversa. Consequentemente, a solução para um vetor de variáveis independentes é mais difícil utilizando decomposição triangular do que bifatoração.

Apesar destas diferenças, ambos os métodos são bem superiores à inversão direta da matriz. Para torná-los eficientes, no entanto, é necessária uma implementação cuidadosa, com técnicas de matrizes esparsas que permitam reduzir efetivamente o tempo de solução do sistema de equações. Este assunto é abordado logo a seguir.

4.2.3 - Estruturas de Dados Dinâmicas para as Matrizes Fator do Método da Bifatoração

Uma vez que a matriz nodal modificada esteja construída o procedimento seguinte consiste em obter as matrizes fator conforme descrito na seção 4.2.1. Para tornar mais claro o assunto repetem-se a seguir as equações 4.1 e 4.7

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{x} = \mathbf{R}^1\mathbf{R}^2 \ldots \mathbf{R}^{n-1}\mathbf{L}^n\mathbf{L}^{n-1} \ldots \mathbf{L}^2\mathbf{L}^1\mathbf{b}$$

Onde o produto do vetor **b** pelas matrizes fator **L** e **R** obtém a solução **x** de forma mais eficiente que a solução pela inversa. A topologia das matrizes **L** e **R** permite que estas sejam armazenadas de forma bastante compacta. Uma matriz L^k genérica tem a seguinte topologia (repetição da eq. 4.8):

$$L^k = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & L_{k,k} & & \\ & & L_{k+1,k} & 1 & \\ & 0 & \vdots & & \ddots \\ & & L_{n,k} & & & 1 \end{bmatrix} \qquad (4.12)$$

Durante o processo de solução do sistema de equações o vetor b^{k-1} (resultante após k-1 multiplicações matriciais) é multiplicado pela matriz L^k resultando em um novo vetor b^k :

$$\underline{b}^k = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & L_{k,k} & & \\ & 0 & L_{k+1,k} & 1 & \\ & & \vdots & & \ddots \\ & & L_{n,k} & & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_k \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_k \cdot L_{k,k} \\ b_k \cdot L_{k+1,k} + b_{k+1} \\ \vdots \\ b_k \cdot L_{n,k} + b_n \end{bmatrix} \quad (4.13)$$

Nota-se que na obtenção do vetor b^k os elementos da coluna k de L^k são utilizados seqüencialmente. Portanto uma estrutura de dados dinâmica que permita uma eficiente solução do sistema de equações pelas matrizes fator armazena as matrizes L^k por colunas.

Como cada matriz L^k possui apenas uma coluna com elementos não nulos à exceção da diagonal unitária podemos sobrepor todas as matrizes L^k formando uma única matriz. Assim sendo a estrutura apresentada na Fig. 4.6 permite armazenar todas as matrizes L^k, de maneira a propiciar uma eficiente solução do sistema de equações.

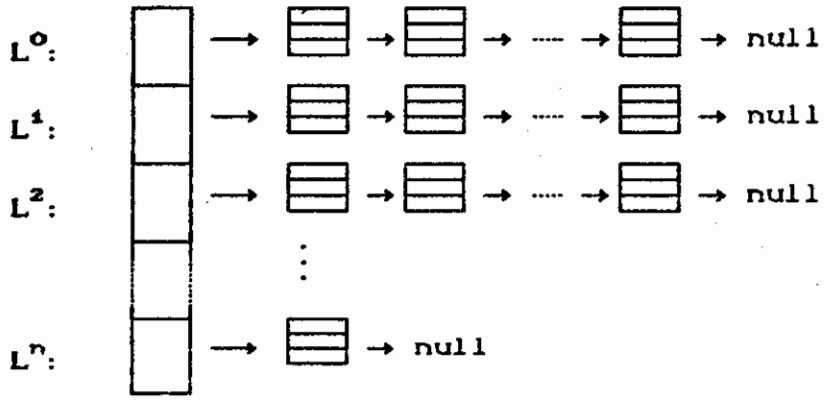


Fig. 4.6 - Estrutura de dados para armazenar as matrizes L por vetor de listas

A estrutura de dados da Fig. 4.6 é semelhante à da Fig. 4.1. No entanto a da Fig. 4.1 é utilizada para armazenar uma única matriz de ordem "n" (nodal modificada), enquanto a da Fig 4.6 armazena "n" matrizes (matrizes fator L). Denominou-se ambas estruturas de "vetor de listas", a interpretação do seu conteúdo sendo porém distinta. A última lista da estrutura acima foi desenhada com apenas um elemento pois a matriz L^n possui apenas uma diagonal unitária com um elemento geralmente diferente de "1" na última posição. As demais listas possuem um número variável de células básicas, dependendo da esparsidade da matriz.

Após a multiplicação de **b** pelas matrizes L^k o processo de solução continua multiplicando o vetor resultante pelas matrizes R^k, que tem a forma geral (repetição da eq. 4.9):

$$R^k = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & 0 & & \\ & & 1 & & & & \\ & & & 1 & R_{k,k+1} & \cdots & R_{k,n} \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & 0 & & & & 1 \end{bmatrix} \qquad (4.14)$$

O vetor b^k que se obtém pela multiplicação de R^k por b^{k-1} é dado por

$$\underline{b}^k = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & 0 & & \\ & & 1 & & & & \\ & & & 1 & R_{k,k+1} & \cdots & R_{k,n} \\ & & & & 1 & & \\ & & & 0 & & \ddots & \\ & & & & & & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_k \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_k + b_{k+1}R_{k,k+1} + \ldots + b_nR_{k,n} \\ b_{k+1} \\ \vdots \\ b_n \end{bmatrix}$$

(4.15)

Conforme se vê, apenas os valores da linha k na matriz $\mathbf{R}^k$ são acessados seqüencialmente na obtenção de $\mathbf{R}^k.\mathbf{b}^{k-1}$. A diagonal unitária não precisa ser armazenada, pois valor e posição são conhecidos. Na realidade durante a solução os produtos pelos elementos da diagonal nem precisam ser efetuados. Pode-se então guardar todos as matrizes $\mathbf{R}^k$ em uma única matriz, conforme foi feito com as matrizes $\mathbf{L}^k$. A Fig. 4.7 apresenta a estrutura de dados para armazenar as matrizes fator R:

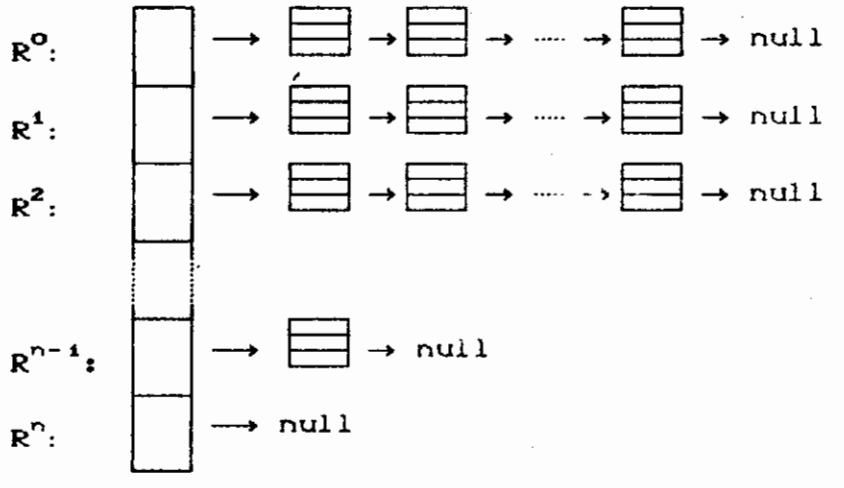


Fig 4.7 - Estrutura de dados para armazenar as matrizes R por vetor de listas

Novamente a estrutura "vetor de listas" da Fig. 4.7 armazena um conjunto de "n" matrizes. É interessante observar que a ultima lista está vazia pois a última matriz R^n é uma matriz unitária.

Uma vez que o algoritmo de bifatoração tenha sido aplicado à matriz nodal modificada A e as matrizes fator tenham sido obtidas, o espaço de memória ocupado por A pode então ser liberado. A representação interna do circuito passa a ser feita pelas matrizes fator (L e R), pelas matrizes C e pelo vetor das fontes independentes. Desde que se mantenha a mesma taxa de amostragem, esta representação interna do circuito pode ser utilizada para se obter a simulação para qualquer conjunto de sinais de entrada.

A seguir apresentam-se considerações sobre a necessidade e conveniência de reordenar o sistema de equações durante o processo de bifatoração.

4.2.4 - Reordenação de Linhas durante o Processo de Bifatoração

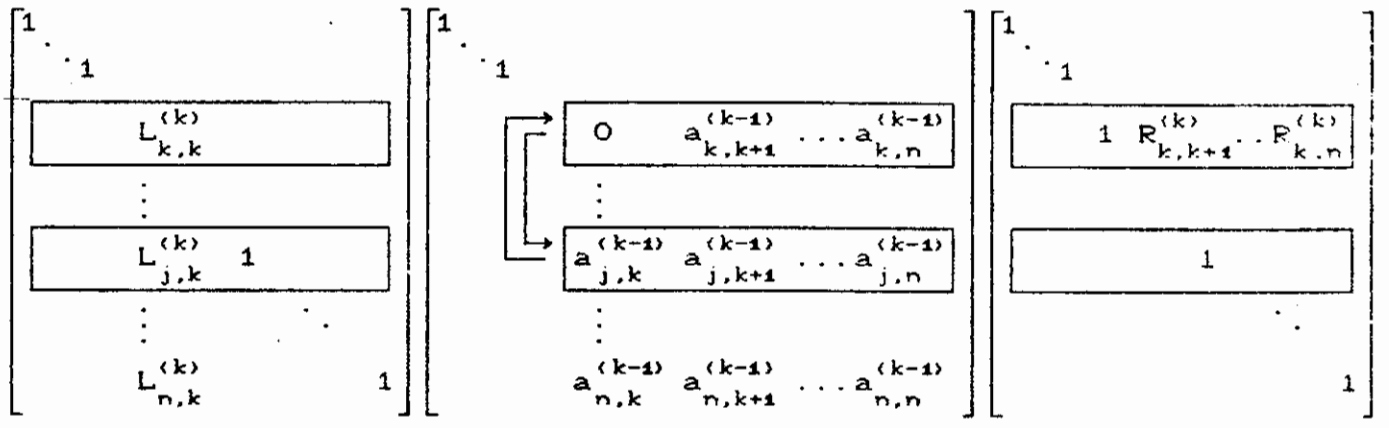
O algoritmo da bifatoração foi proposto numa época em que as formulações das equações de circuitos mais utilizadas eram a matriz de admitâncias nodal e formas híbridas que oneravam a ordem do sistema de equações. A formulação nodal simples fornece uma matriz bem condicionada com diagonal dominante. Não existe, portanto, grande possibilidade do procedimento de bifatoração falhar por encontrar um elemento nulo na diagonal [4]. As formas híbridas por sua vez, não fornecem em geral uma matriz com diagonal dominante, tornando necessário um maior cuidado na aplicação do método da bifatoração.

A matriz de admitância nodal modificada não permite, em geral, aplicação direta do processo de bifatoração. Fontes de tensão e dispositivos não representáveis na forma de admitância podem gerar mais freqüentemente linhas e colunas com elementos nulos na diagonal. O processo de bifatoração básico utiliza em cada passo de redução o elemento da diagonal da matriz A reduzida como pivô. Caso este elemento seja nulo, o procedimento, conforme apresentado, não teria como prosseguir. Esta situação pode ocorrer quando a matriz original é singular, devido a algum erro na topologia do circuito, ou então simplesmente devido a uma ordem inadequada de eliminação das variáveis.

Caso a matriz seja singular não há realmente como prosseguir e o algoritmo deve ser interrompido. Para a situação em que a ordem das variáveis obtida pela formulação nodal modificada produz uma ordem inadequada para a eliminação das variáveis constatou-se que o procedimento de bifatoração pode prosseguir pela permutação de linhas da matriz A reduzida (eq. 4.10). A permutação pode ser feita com qualquer linha abaixo da linha corrente que tenha um elemento não nulo na coluna correspondente (ver-se-á mais adiante que a propagação do erro de arredondamento impõe restrições adicionais). Para que o sistema de equações originais seja preservado faz-se necessário também permutar linhas do vetor independente e certas linhas das matrizes fator.

Para ilustrar o tratamento de elemento nulo na diagonal, suponha-se uma situação hipotética, tal que após o passo de redução "k-1" a nova matriz A^k reduzida seria obtida por:

$$A^k = L^k \; A^{k-1} \; R^k =$$



(4.16)

No entanto, para que as matrizes fator L^k e R^k possam ser geradas, quando o elemento da matriz A reduzida $a^{(k-1)}_{k,k} = 0$, torna-se necessário permutar as linhas "k" e "j" da matriz A^{k-1} reduzida, conforme indicado pelas setas nas matrizes acima. A linha "j" deve ter, obviamente, um elemento não nulo na coluna "k".

O vetor independente é formado pela soma algébrica de alguns fatores (repetição da eq. 4.2):

$$b = C_0 . v_n + C_1 . v_{n-1} + \ldots C_p . v_{n-p} + f \qquad (4.17)$$

Assim sendo, para manter a consistência do sistema de equações originais, as mesmas linhas que foram permutadas na matriz A reduzida devem ser permutadas nas matrizes C_0, C_1, ... C_p. Com a estrutura de dados proposta para armazenar as matrizes C (fig. 4.4), a permutação de duas linhas de todas a matrizes C resume-se à permutação de dois ponteiros. Além da permutação das linhas das matrizes C, as fontes que ocupam as linhas correspondentes no vetor f também devem ser permutadas

entre si. Com isso tem-se o vetor independente **b** consistente com as matrizes fator geradas.

As matrizes fator geradas antes da permutação de linhas também são afetadas, necessitando ajustes. Se o procedimento estava no passo "k", apenas as matrizes $L^{(1)}\ L^{(2)}\ \ldots\ L^{(k-1)}$ precisam ser ajustadas para manter consistente o sistema de equações (repetição da eq. 4.7):

$$x = R^{(1)}\ R^{(2)} \ldots\ R^{(n-1)}\ R^{(n)}\ L^{(n)}\ L^{(n-1)} \ldots\ L^{(2)}\ L^{(1)}\ b \qquad (4.18)$$

Isto porque as matrizes $R^{(1)}\ R^{(2)}\ \ldots\ R^{(k-1)}$ são matrizes com a diagonal principal unitária e elementos diferentes de zero nas linhas (1), (2), ... (k-1) respectivamente. Portanto não são alteradas pela permutação da linha "k" por uma linha abaixo de "k". Entretanto, as matrizes $L^{(1)}\ L^{(2)}\ \ldots\ L^{(k-1)}$ são matrizes com a diagonal principal unitária e elementos diferentes de zero nas colunas (1), (2), ... (k-1) respectivamente. A permutação da linha "k" por uma linha abaixo de "k" na matriz **A** reduzida, implica desta forma na permutação das linhas correspondentes nas matrizes $L^{(1)}\ L^{(2)}\ \ldots\ L^{(k-1)}$. Isto está ilustrado na eq. 4.16 através de retângulos envolvendo as linhas permutadas.

Com isso mantém-se a consistência do sistema de equações e as matrizes fator podem ser utilizadas na solução do sistema de equações lineares que descrevem o circuito.

Na seção seguinte estudam-se os efeitos das permutações de linhas sobre a esparsidade das matrizes fator.

4.3 - Efeito da Ordenação do Sistema de Equações sobre a Esparsidade das Matrizes Fator

Esta seção procura abordar aspectos da ordenação eficiente dos sistemas esparsos gerados pela formulação nodal modificada. Freqüentemente, a ordem em que as variáveis são eliminadas desempenha um papel importante no esforço computacional dos métodos de matrizes esparsas. Desta forma deseja-se, para uma dada matriz esparsa A, encontrar uma reordenação adequada de linhas e colunas de tal forma que as matrizes fator do sistema permutado mantenham o máximo de esparsidade possível. O apêndice B ilustra com um exemplo o efeito causado pela permutação de equações no esforço computacional necessário à solução de sistemas esparsos.

Se A consiste da matriz nodal modificada descritiva de um circuito linear, o problema também pode ser colocado na forma: Dado um circuito linear, encontrar a melhor ordenação de nós e equações constitutivas, de tal modo que a matriz nodal modificada A tenha fatores L e R os mais esparsos possível.

Um esquema de ordenação é considerado ótimo quando as matrizes obtidas no processo de fatoração preservam ao máximo a esparsidade da matriz original. Este problema geralmente não é simples e requer algoritmos elaborados e um grande esforço computacional. Entretanto, este esforço computacional inicial pode ser compensado se o sistema de equações for resolvido inúmeras vezes para diferentes vetores independentes, e a redução do número de "*fill-ins*"[10] obtida com a reordenação ótima for significativa.

---

[10] "Fill-in" é definido como um elemento não nulo que é acrescentado às matrizes fator durante o processo de solução além do número de elementos não nulos da matriz original.

A avaliação do problema colocado acima pode ser complexa pois depende da topologia do circuito a ser simulado e de quantas iterações da simulação serão feitas com a mesma matriz bifatorada.

Um método eficiente de obtenção da ordenação ótima é a simulação dinâmica dos *fill-ins* gerados durante o processo de fatoração. A simulação é feita de uma forma simbólica, sem que as matrizes fatoradas sejam calculadas efetivamente (sem operações de ponto flutuante). O objetivo de determinar a posição dos *fill-ins* é alcançado apenas operando sobre os índices que indicam a posição dos elementos não nulos da matriz esparsa [1].

Por outro lado, na maioria dos casos um esquema de ordenação sub-ótimo é suficiente para obter-se um compromisso aceitável entre a preservação da esparsidade e a complexidade do algoritmo de ordenação.

Um outro aspecto pelo qual a ordenação ótima não pode ser implementada no caso do sistema de equações nodais modificadas é devido ao acúmulo de erro de arredondamento. Durante os trabalhos de pesquisa implementou-se um algoritmo de ordenação sub-ótimo, sem pivotamento, e o erro propagou-se de tal forma que o processo de bifatoração, para matrizes de ordem elevada, muitas vezes não chegava ao fim. Outras vezes as matrizes fator obtidas faziam com que a solução temporal divergisse. O efeito do erro de arredondamento tornou-se crítico para sistemas com mais de 50 nós. A simulação de um circuito com 61 nós, por exemplo, só pôde ser efetuada quando o algoritmo de ordenação passou a levar em consideração o pivotamento como forma de reduzir a propagação do erro de arredondamento.

Pissanetsky [2] desenvolve considerações sobre erro numérico em processos de eliminação Gaussiana. Os resultados são ali apresentados considerando-se a fatoração triangular (LU). Os resultados fornecidos por Pissanetsky podem, no entanto, ser aplicados para a bifatoração, uma vez que os processos numéricos envolvidos são semelhantes. Um resultado fundamental é o que define o erro acumulado por um elemento das matrizes fator:

$$|E_{ij}| \leq 3.01\ \varepsilon_M \alpha_{ij} n_{ij} \qquad (4.19)$$

onde ε_M - precisão da máquina utilizada. Para computadores digitais que utilizam aritmética binária e representação em ponto flutuante constituída por uma mantissa de t bits e um expoente, $\varepsilon_M = 2^{-t}$.

$\alpha_{ij} = \max_k |a_{ij}^{(k)}|$, ou seja α_{ij} é o maior valor absoluto que aparece na posição (i,j) das matrizes reduzidas $A^{(k)}$.

n_{ij} - número de operações em ponto flutuante efetuadas sobre o elemento a_{ij} até a obtenção do respectivo valor nas matrizes fator (L_{ij} ou R_{ij}).

Com este erro acumulado o produto das matrizes fator fornece

$$R^{(1)}R^{(2)}\ldots R^{(n-1)}R^{(n)}L^{(n)}L^{(n-1)}\ldots L^{(2)}L^{(1)} = A^{-1} + E \qquad (4.20)$$

onde cada elemento da matriz de erro E em (4.20) é definido por (4.19).

Na equação (4.20) A^{-1} é o valor exato da inversa de A, obtido com precisão infinita. A matriz de erro E é acrescida

como conseqüência da precisão finita do computador.

A equação (4.19) fornece limites para os elementos de E em termos dos parâmetros ε_M, α_{ij} e n_{ij}. Atuando adequadamente sobre estes parâmetros pode ser melhorada a precisão da solução. Por outro lado, uma má escolha pode produzir resultados imprecisos ou mesmo uma completa perda de significado de A^{-1}.

Para que o parâmetro α_{ij} seja mantido pequeno é necessário evitar um excessivo crescimento dos elementos das sucessivas matrizes $A^{(k)}$. Isto é conseguido geralmente pela escolha de pivôs apropriados durante o processo de eliminação. Durante a redução da matriz A os pivôs são utilizados como divisores (4.8, 4.9 e 4.10). Desta forma, quanto maior o valor do pivô, menor será o crescimento dos elementos das matrizes reduzidas $A^{(k)}$.

O pivô pode ser escolhido como o maior elemento da matriz reduzida ativa[11]. Este método conhecido por pivotamento total é muito restritivo pois a sua aplicação pode destruir completamente a esparsidade original. A prática mostra também que o pivotamento parcial, por linha ou coluna, fornece resultados plenamente satisfatórios em termos de erro numérico [2]. O pivotamento parcial também é por demais restritivo não levando em consideração a preservação da esparsidade. Um critério mais flexível para a escolha do pivô, sem perda significativa de precisão, o *pivotamento por limiar*, foi introduzido por Reid [26]. No *pivotamento por limiar* todos os elementos não nulos da matriz reduzida ativa que satisfaçam a condição de *limiar* são candidatos primários à pivô. Uma tolerância u é escolhida na faixa $0 < u \leq 1$.

---

[11] Matriz reduzida ativa é a parte de $A^{(k)}$ ainda não diagonalizada.

Assim, os candidatos primários a pivô devem satisfazer uma das seguintes *condições de limiar*:

$$|a_{ij}^{(k)}| \geq \upsilon \max_{k \leq p \leq n} |a_{pj}^{(k)}| \qquad (4.21a)$$

$$|a_{ij}^{(k)}| \geq \upsilon \max_{k \leq q \leq n} |a_{iq}^{(k)}| \qquad (4.21b)$$

O pivô pode assim ser escolhido por critérios que preservem a esparsidade. A escolha de $\upsilon = 1$ corresponde ao pivotamento parcial. A prática mostrou que o valor de υ não é muito crítico [2] e valores de $\upsilon = 0.01$ permitem boa retenção tanto da esparsidade como da precisão.

O parâmetro ε_M, em (4.19) é função do comprimento de palavra do computador utilizado, e não há em geral disponibilidade para alterar o seu valor. É interessante registrar que para máquinas IBM, quando t é aumentado de 24 para 56 bits, correspondendo a precisão simples e dupla, o erro decresce de um fator $2^{32} \simeq 4 \times 10^9$.

Por último, em (4.19) o parâmetro n_{ij}, que corresponde ao número de operações realizadas sobre as matrizes $A^{(k)}$ reduzidas, revela que os esforços em preservar a esparsidade reduzem o erro, além é claro, de outras vantagens obtidas pela exploração da esparsidade. Se um certo elemento da matriz reduzida só é atualizado poucas vezes então o erro numérico acumulado será reduzido. Por sua vez, um elemento que é atualizado diversas vezes terá maior erro numérico acumulado. O maior valor possível para n_{ij} é a ordem da matriz A.

Estas considerações práticas levaram à proposta do esquema de ordenação adotado, apresentado na seção seguinte.

4.3.1 - Esquema de Ordenação Proposto

O esquema de ordenação das equações adotado neste trabalho está dividido em duas partes. Inicialmente o sistema de equações é pré-ordenado, visando aproveitar linhas com um ou dois elementos não nulos. A segunda ordenação ocorre durante o processo de fatoração, como conseqüência do pivotamento por limiar.

A matriz nodal modificada consiste de duas partes. A primeira é a matriz de admitâncias nodais convencional, que inclui as contribuições de todos os componentes de circuito com representação por admitância. A segunda parte da matriz nodal modificada é gerada por componentes que não possuem representação por admitância. Tipicamente esta parte da matriz possui diversos zeros na diagonal, não tornando possível a aplicação do pivotamento diagonal. No entanto, a reordenação do sistema de equações nodais modificadas permite em geral remover os zeros da diagonal.

Conforme visto nos capítulos 2 e 3, as estampas que geram zeros na diagonal também introduzem elementos de valor ± 1 na linha e na coluna correspondentes ao elemento diagonal nulo. Estes elementos são conhecidos como *uns estruturais*. Na maioria das estampas os *uns estruturais* são colocados em posições simétricas na matriz.

É desejável ordenar as equações de tal forma que os pivôs com melhores características de preservação de esparsidade sejam eliminados primeiro. Uma linha ou coluna que possua apenas um elemento, quando é eliminada não gera nenhum "fill-in". Fontes de tensão com um terminal aterrado geram linhas e colunas com esta característica e são os componentes mais desejáveis para serem utilizados como pivôs iniciais. No entanto, se a busca pelo pivô é restrita aos

elementos da diagonal, estes elementos são perdidos. Torna-se pois necessário permutar o sistema de equações para que estes elementos sejam aproveitados. Um outro aspecto retratado em [17] é o cancelamento completo de elementos da diagonal quando o circuito inclui um conjunto de corte contendo apenas componentes não representados por admitância. A permutação das linhas contendo *uns estruturais* é uma forma de evitar o cancelamento completo.

As considerações anteriores levaram à adoção do seguinte procedimento de pré-ordenação fornecido por K. S. Kundert em [28, pg 306] :

Dado: A, uma matriz nodal modificada nxn
0: k ← 0.

Linhas com um único elemento não nulo ("singleton"):
1: Incremente k. Se k > n, então vá ao passo 7, não há mais "singletons" a serem reordenados.
2: Se $a_{kk} \neq 0$ volte ao passo 1.
3: Se a_{kk} não é o único elemento não nulo da linha, então volte ao passo 1; a_{kk} não é "singleton".
4: j ← 0.
5: Incremente k. Se j > n, então volte ao passo 1. O zero em a_{kk} não deve ser removido da diagonal.
6: Se $|a_{kj}| = 1$ e $a_{kj} = a_{jk}$, então permute as linhas j e k e retorne ao passo 1, caso contrário volte ao passo 5.

Demais linhas
7: k ← 0.
8: Incremente k. Se k > n, então pare, a matriz não precisa mais ser reordenada.
9: Se $a_{kk} \neq 0$ então volte ao passo 8.
10: j ← 0.
11: Incremente j. Se j > n, então retorne ao passo 8. O

zero em a_{kk} não deve ser removido da diagonal.
12: Se $|a_{kj}| = 1$ e $a_{kj} = a_{jk}$, então permute as linhas j e k e retorne ao passo 8, caso contrário volte ao passo 11.

Depois que o sistema de equações estiver pré-ordenado passa-se à obtenção das matrizes fator. Durante a obtenção das matrizes fator ocorrem novas permutações de equações no sentido de reduzir a propagação de erro numérico. Neste passo a ordenação proposta considera que a permutação é feita na seguinte condição:

Critério de permutação:

Se durante o processo de bifatoração o valor absoluto do elemento da diagonal for menor que um certo "ε", a linha correspondente deve ser permutada com uma linha abaixo, que satisfaça tanto considerações de esparsidade como de propagação de erro.

Se "ε" é grande provocam-se muitas permutações perdendo-se a estrutura banda original e conseqüentemente a esparsidade. No entanto se "ε" é muito pequeno, o erro numérico pode propagar-se significativamente. Após a simulação de diversos circuitos com diferentes valores observaram-se bons resultados tanto em termos de preservação de esparsidade como propagação de erro para valores de $\varepsilon \cong 10^{-6}$ até $\varepsilon \cong 10^{-10}$. O valor implementado foi de $\varepsilon = 10^{-8}$.

Caso seja constatada a necessidade de permutar linhas adota-se uma variante do *pivotamento por limiar* citado na seção anterior:

1 - Para o passo de redução de $A^{(k)}$ é feita a busca do maior elemento da coluna "k". O *limiar de pivotamento* é estabelecido como $0.001 \max_{k \le p \le n} |a_{pk}^{(k)}|$.

2 - Determina-se o número de elementos não nulos em cada uma das linhas da matriz $A^{(k)}$ ativa.

3 - A coluna "k" da matriz $A^{(k)}$ é percorrida na busca de uma linha que tenha o mínimo de elementos não nulos e que o elemento da coluna "k" satisfaça o *limiar de pivotamento*. Caso uma linha "j" preencha as duas condições a busca chega ao fim e as linhas "k" e "j" do sistema de equações são permutadas.

4 - Caso o passo 3 não encontre nenhuma linha que satisfaça as duas condições, de esparsidade e de propagação de erro, então é relaxada a condição de esparsidade. Passam a ser consideradas linhas com maior número de elementos não nulos até que a condição de pivotamento limiar seja satisfeita.

O passo 1 estabelece o *limiar de pivotamento* utilizando os elementos da coluna "k". O passo 2 associa a cada linha da matriz $A^{(k)}$ reduzida um número que pode ser utilizado como critério de preservação de esparsidade. Quanto menor o número de elementos não nulos menor será a produção de "fill-ins". Os passos 3 e 4 buscam uma nova linha para ocupar a posição "k" no sistema de equações. A linha que tenha menor número de elementos não nulos e que satisfaça a condição de limiar será escolhida. Caso mais de uma linha satisfaça as condições, a primeira encontrada será a escolhida.

Com o material apresentado até aqui tem-se os fundamentos teóricos que permitem a implementação de um simulador temporal de circuitos lineares. No capítulo seguinte são descritos os demais algoritmos implementados no simulador temporal de circuitos lineares.

5 - Descrição dos Algoritmos Implementados no Simulador Temporal de Circuitos Lineares

O assundo abordado nos capítulos anteriores fornece a base teórica necessária ao entendimento dos modelos de circuito adotados. À esta base teórica é preciso acrescentar a descrição de algoritmos de tal forma a tornar factível a implementação do simulador em ambiente computacional. Desta forma, no presénte capítulo apresentam-se os algoritmos utilizados na construção de uma ferramenta de auxílio ao projeto de circuitos lineares.

O algoritmo geral para a simulação de circuitos lineares no domínio do tempo pode ser visto no fluxograma da figura 5.1. Este fluxograma mostra os módulos que compõem o simulador. A seguir na seção 5.1 procurar-se-á discutir com mais detalhes o fluxograma apresentado na Fig. 5.1.

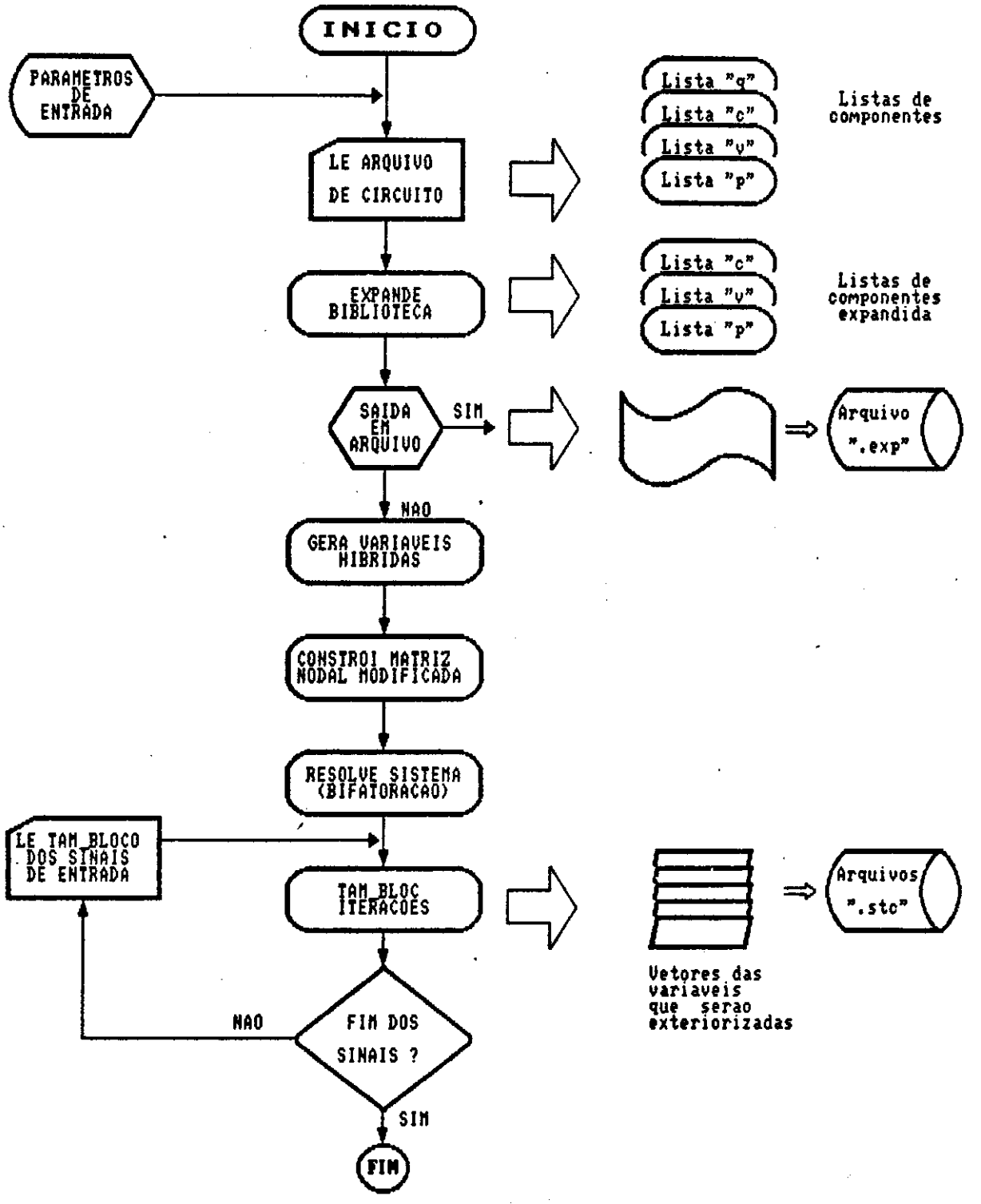


Fig. 5.1 - Fluxograma Geral do Simulador Temporal de Circuitos proposto.

5.1 - Descrição Geral do Simulador Temporal de Circuitos

A simulação tem início com a especificação, na entrada de dados, do circuito a ser simulado e do algoritmo de discretização a ser utilizado. O simulador recebe a descrição do circuito a ser simulado através de um arquivo texto[12]. A descrição do circuito deve ser feita de uma forma simples e completa. Uma descrição completa implica em conter tanto a topologia como o valor de cada um dos componentes do circuito.

A descrição do circuito através de um arquivo texto é apropriada para o usuário, mas não é uma representação conveniente para o computador. Desta forma os componentes são separados em categorias e armazenados em listas alocadas dinamicamente. Quatro listas são criadas ("q", "c", "v" e "p") em função da quantidade de parâmetros necessários para a descrição completa dos componentes. As categorias de componentes que estas listas armazenam serão apresentadas mais adiante.

A lista "q" tem um tratamento diferenciado das demais. Essa lista armazena dispositivos com três terminais de acesso (transistores e amplificadores operacionais) que não possuem uma estampa que permita a construção direta da matriz nodal modificada. Desta maneira é necessário expandir estes dispositivos de três terminais em termos de componentes mais simples que os modelem. Diferentes modelos podem ser utilizados para representar um transistor, dependendo da aplicação e do nível de detalhe que seja necessário. Desta

---

[12] Um arquivo texto contém caracteres que podem ser lidos e manipulados pelo usuário, através de um editor de texto.

forma uma biblioteca de modelos é adequada para conter as diferentes representações dos transistores. Diferentes transistores estão disponíveis na biblioteca com diferentes valores dos parâmetros de um modelo.

A expansão de todos os dispositivos da lista "q" em termos dos componentes das outras listas fornece uma nova representação, interna ao computador, do circuito que está sendo simulado. Esta nova representação aparece como "Listas de componentes expandida" no fluxograma. Existe neste ponto a possibilidade de obter-se um arquivo de saída contendo a descrição expandida do circuito. Este arquivo é útil para verificação do circuito que está sendo simulado, bem como pode ser utilizado como arquivo de entrada para uma outra simulação. Neste último caso poderiam ser verificadas formas de ondas de nós internos a um certo modelo de transistor ou de algum modelo de amplificador operacional.

Uma vez que todos os dispositivos do circuito estão representados por componentes que possuem estampas para a formulação nodal modificada é preciso determinar quais serão as variáveis da matriz nodal modificada e que posição estas variáveis irão ocupar na matriz. Este módulo especificado no fluxograma por "Gera variáveis hibridas" deve percorrer as listas de componentes e gerar uma lista de componentes cujas correntes precisam aparecer na parte híbrida da matriz nodal modificada.

Após este passo a matriz nodal modificada pode ser construída pela utilização da estampa de cada componente. O sistema de equações é armazenado através de técnicas de matrizes esparsas, reduzindo-se o esforço computacional e quantidade de memória necessária.

Uma vez que a matriz nodal modificada esteja disponível

procede-se com a solução do sistema de equações. O processo de solução por bifatoração obtém inicialmente as matrizes fator. As mesmas matrizes fator são utilizadas em todos os passos de solução numérica. Apenas o vetor independente é alterado em cada iteração. As matrizes fator são armazenadas utilizando a estrutura de dados apresentada no capítulo 4 (Fig. 4.4). Esta estrutura de dados é bem adequada ao processo de iteração que a utiliza, possibilitando grande eficiência computacional.

O vetor independente é atualizado em cada passo de solução, pelas fontes controladas dos modelos discretos associados e pelos sinais de entrada. Para cada amostra dos sinais de entrada obtém-se a solução das variáveis nodais modificadas utilizando-se as matrizes fator obtidas pela bifatoração. Visando a simulação de circuitos contendo uma grande quantidade de nós submetidos a sinais com um grande número de amostras, o processamento das amostras é feito por blocos. Esta metodologia permite que apenas um bloco por vez ocupe a memória principal do computador. Cada bloco é processado e as variáveis que tenham sido solicitadas como saída do simulador são armazenadas em arquivo. O procedimento se repete até que todos os sinais tenham sido processados.

O circuito simulado pode ser excitado por diversas fontes independentes. Cada fonte é considerada um sinal de entrada. Para que a simulação tenha sentido todos os sinais de entrada devem possuir a mesma taxa de amostragem e o mesmo número de amostras. Além disso o vetor fonte (eq. 4.2) deve ser formado pelos valores das fontes em instantes de amostragem correspondentes. O retângulo "Le TAM_BLOCO dos sinais de entrada" na Fig. 5.1, mostra que a simulação por blocos lê blocos correspondentes de todos os sinais de entrada. Os sinais de entrada podem tanto ser inteiros com reais; no entanto, internamente eles serão convertidos para

reais.

Em geral, nem todas as variáveis nodais modificadas são de interesse para o usuário. Desta forma ele deve especificar na entrada de dados quais tensões e/ou correntes serão exteriorizadas na forma de arquivo de sinal. Ao final da simulação, para cada variável solicitada terá sido criado um arquivo de sinal. Estes arquivos estão indicados na Fig. 5.1 por "Arquivos '.stc'". A extensão "stc" para os arquivos de saída vem das iniciais de "Simulador Temporal de Circuitos". Os arquivos contendo os sinais de saída possuem o mesmo padrão dos sinais de entrada, facilitando a sua visualização e possível utilização como entradas para outros circuitos.

5.2 - Algoritmo de Leitura de Circuito

A fig. 5.2 mostra o fluxograma do algoritmo de leitura de circuito. O circuito é descrito por meio de um arquivo texto. A leitura do circuito de um arquivo texto pressupõe um padrão para a definição de cada componente. Um esquema adequado é considerar cada componente definido em uma linha de texto.

Diferentes tipos de componentes tem, no entanto, uma quantidade distinta de parâmetros necessários à sua definição completa. Assim sendo os componentes são agrupados em categorias. O fluxograma da Fig. 5.2 indica esta abordagem pelas cinco opções que o algoritmo pode percorrer de acordo com o tipo que está sendo lido de uma certa linha.

Os tipos de componentes definidos são os seguintes (agrupados por categorias):

R,L,C - Resistor, Indutor e Capacitor.
V,I - Fontes independentes de Tensão e de Corrente.
E,G - Fontes de Tensão e de Corrente controladas por tensão.
H,F - Fontes de Tensão e de Corrente controladas por corrente.
Q,O - Transistor e Amplificador Operacional.

Cada categoria tem a sua própria rotina de leitura, que acrescenta os parâmetros que definem o componente à sua representação interna, em forma de lista encadeada.

Todas as linhas do arquivo de entrada são lidas e verificadas quanto à erros de sintaxe. Linhas iniciadas por "*" são consideradas comentários e não afetam a descrição do circuito.

A rotina de leitura do circuito também inclui a leitura de uma linha especial que contém a especificação das variáveis que serão exteriorizadas. Esta linha especial inicia-se com a letra "A". O restante desta linha contém a especificação das tensões e correntes de interesse, que deverão ser escritas em arquivo. As tensões de saída são especificadas peía letra "V" seguidas pelo número do nó. As correntes de saída são especificadas pela letra "I" seguidas pelo nome do ramo cuja corrente é solicitada. A leitura desta linha especial não aparece no fluxograma da Fig. 5.2., visando não obscurecer o algoritmo de leitura principal.

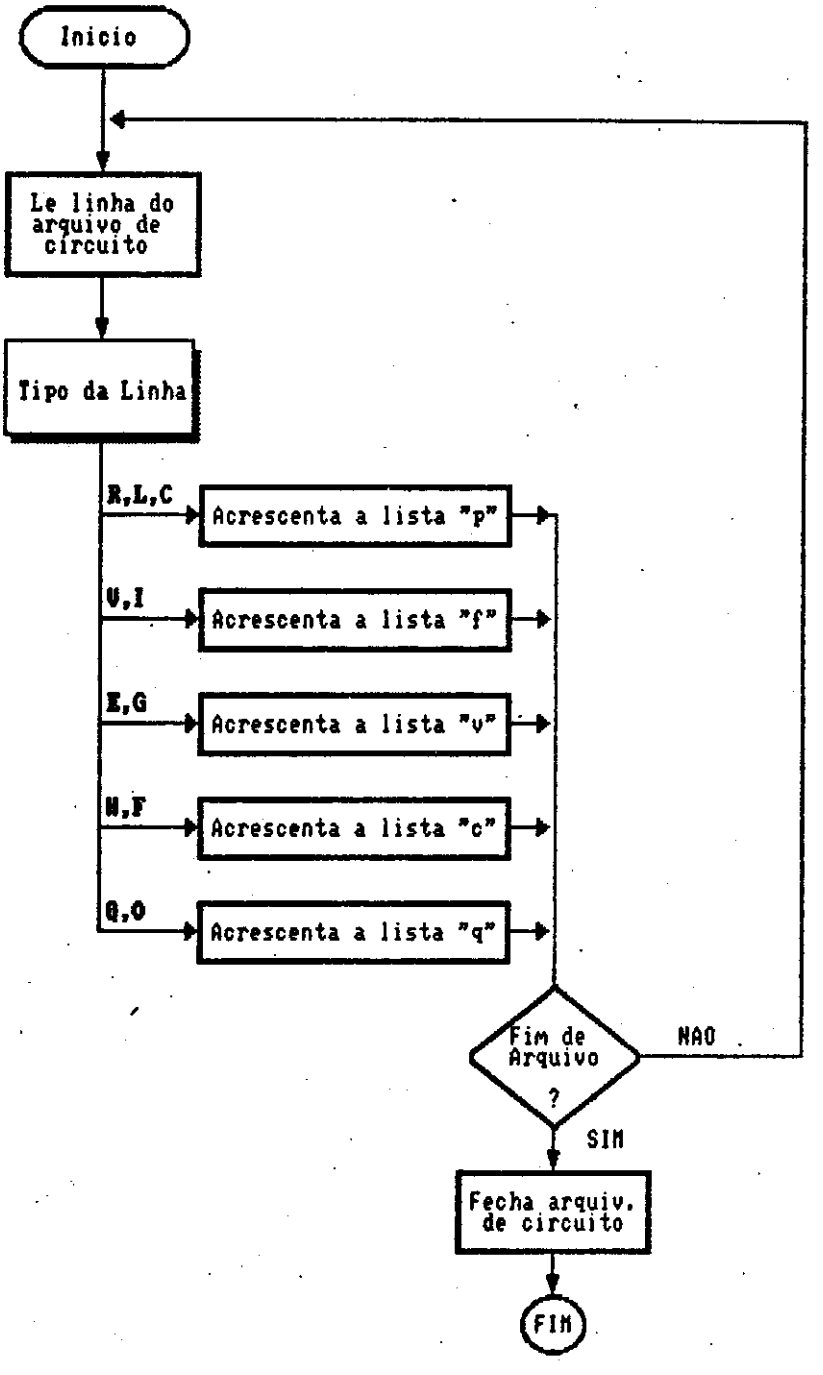


Fig 5.2 - Algoritmo de Leitura de Circuito

A seguir apresentam-se dois arquivos texto que ilustram a sintaxe utilizada na descrição do circuito para o simulador implementado. O primeiro exemplo, Fig. 5.3, é o mesmo da Fig. 2.23. O arquivo texto que o descreve aparece no quadro da Fig. 5.4.

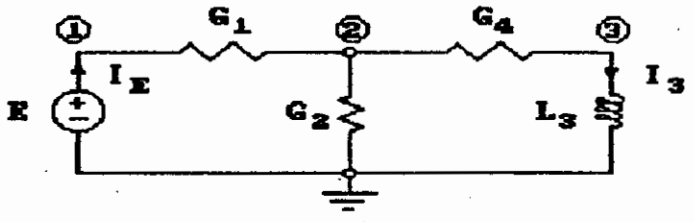


Fig. 5.3 - Circuito contendo Fonte de Tensão e Indutor

```
* Circuito exemplo Fig.5.3
*
* Sinais a serem exteriorizados
A  V2  V3  Ientrada  IL3
*
* Sinal de entrada
Ventrada  1  0  0
*
* Componentes do circuito
R1  1  2  100
R2  2  0  1e3
R4  2  3  100
L3  3  0  10e-3
```

Fig. 5.4 - Arquivo Texto que descreve o circuito da Fig. 5.3

No quadro da Fig. 5.4 as linhas que se iniciam com "*" são comentários. A linha que se inicia com "A" descreve as variáveis de saída. Serão criados quatro arquivos de saída: "V2.STC", "V3.STC", "ENTRADA.STC" e "L3.STC". Os dois

primeiros arquivos conterão as tensões nos nós *2* e *3*, respectivamente. Os dois últimos conterão a corrente na fonte de tensão "entrada" e no indutor "L3".

Na especificação da fonte de tensão o último parâmetro "0" indica que as amostras que definem esta fonte serão lidas do arquivo "entrada". Caso houvesse um valor não nulo, este seria considerado o valor, fixo durante a simulação, de uma fonte DC.

Os resistores e o indutor que completam a descrição deste circuito têm os parâmetros "nome", "nop", "noc" e "valor", conforme pode ser visto na Fig. 5.4.

O segundo exemplo de arquivo texto, Fig. 5.6, corresponde à descrição do circuito Rejeita-faixa de segunda ordem mostrado na Fig. 5.5.

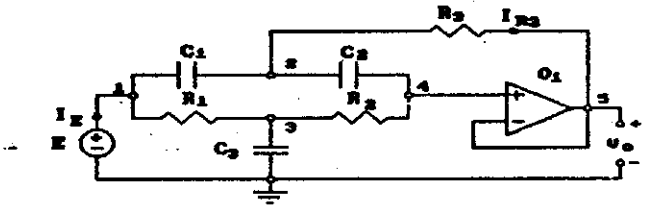


Fig. 5.5 - Circuito Rejeita-faixa de segunda ordem

A descrição de circuito da Fig. 5.6 mostra que, em função da linha "A", serão criados dois arquivos de saída: "V5.STC" e "R3.STC". O primeiro conterá a tensão no nó "5" e o segundo a corrente no resistor "R3" obtidas com a simulação.

```
* Circuito Rejeita-faixa de segunda ordem
A  V5  IR3
Vvarr.ger  1  0  0
*  sai  n_inv  inv  modelo  comp
O1   5     4     5      3     QA741
C1   1     2     2e-7
C2   2     4     2e-7
C3   3     0     4e-7
R1   1     3     1e3
R2   3     4     1e3
R3   2     5     0.5e3
```

Fig. 5.6 - Arquivo Texto que descreve o circuito da Fig. 5.5

Na fig. 5.6 a linha de comentário antes da descrição do amplificador operacional "O1" auxilia a compreensão de cada um de seus parâmetros. "sai" indica o nó de saída a que está conectado o amplificador operacional. "n_inv" e "inv" indicam a entrada não inversora e inversora de "O1". O parâmetro "modelo" = "3" indica que o modelo pi-híbrido completo será utilizado para os transistores de entrada de "O1". O parâmetro "comp" = "QA741" indica os valores numéricos que serão utilizados para os transistores de entrada de "O1".

Uma descrição mais rigorosa dos parâmetros de cada um dos componentes de circuito implementados será vista mais adiante. O objetivo das Figs. 5.4 e 5.6 é apenas ilustrar a forma do arquivo texto utilizado para descrição de circuitos para o simulador implementado.

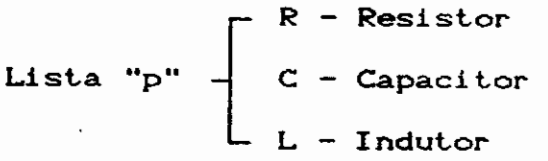
Na seção seguinte são apresentadas as estruturas de dados com as quais os componentes de circuito são representados para o simulador temporal de circuitos.

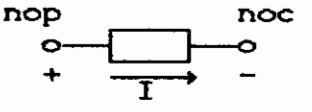
5.2.1 - Estruturas de Dados para Representação Interna do Circuito

Conforme citado na seção anterior os componentes que estão disponíveis para definição do circuito estão agrupados em categorias quanto ao número de parâmetros que os especificam. Cada categoria tem, desta maneira uma estrutura de dados distinta para armazenar estes parâmetros.

É interessante que o simulador temporal de circuitos tenha um certo nível de independência da configuração do computador utilizado. Desta forma é conveniente que as estruturas de dados sejam armazenadas em listas alocadas dinamicamente. Esta característica permite que o tamanho máximo de circuito que pode ser simulado seja determinado durante a simulação (função da memória de massa disponível em cada configuração). Além disso, após construção da matriz nodal modificada, a memória que foi alocada dinamicamente para representação interna do circuito pode ser devolvida, permitindo-se a sua utilização por outras estruturas de dados.

A estrutura de dados da lista "p", que armazena os parâmetros de 'Resistores, Capacitores e Indutores, é apresentada na fig. 5.7. Ponteiros auxiliares que indicam o começo (ramo_p_inic) e o fim (ultimo_p) da lista estão disponíveis, de tal forma a facilitar o acesso à estrutura de dados.





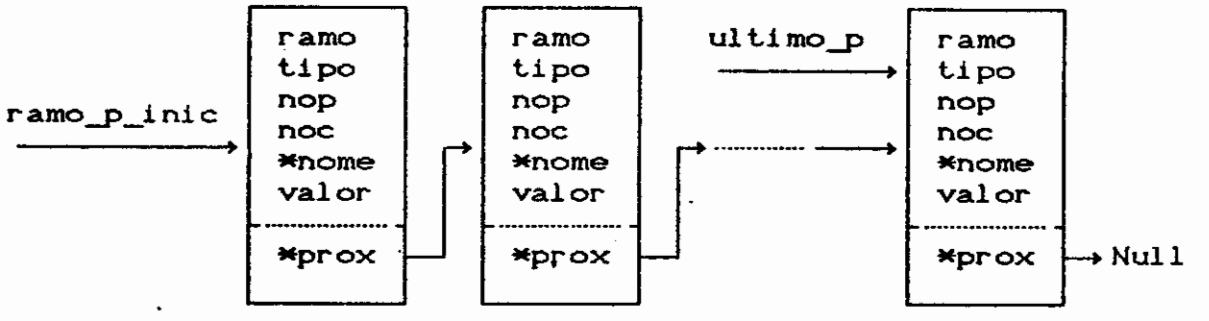


Fig. 5.7 - Estrutura de dados da lista "p".
(Lista encadeada alocada dinamicamente)

A seguir é apresentada uma breve descrição dos campos da estrutura de dados da lista "p". Esta descrição tem a finalidade de mostrar a necessidade de cada um dos campos da estrutura de dados de maneira a descrever completamente a topologia e a relação constitutiva do componente dentro do circuito.

ramo - identificação do ramo, permite referenciar a corrente deste ramo como controlador de uma fonte controlada por corrente. Permite ainda identificar a corrente do ramo como uma das variáveis de saída.

tipo - identifica o tipo do componente (R, L ou C).

nop - nó de partida ao qual o componente está conectado, permite identificar a polaridade da corrente que percorre o componente.

noc - nó de chegada ao qual o componente está conectado.

*nome - ponteiro que dá acesso ao nome do componente. Esta informação pode ser utilizada quando a corrente do componente é solicitada como variável a ser exteriorizada.

valor - valor em Ohms da resistência, em Farads da capacitância e em Henrys da indutância.

*prox - ponteiro que indica o próximo elemento da lista encadeada. O último elemento da lista aponta para "null" (constante pré-definida que indica fim da lista).

A lista "f", vista na fig. 5.8 armazena as fontes independentes de tensão e de corrente.

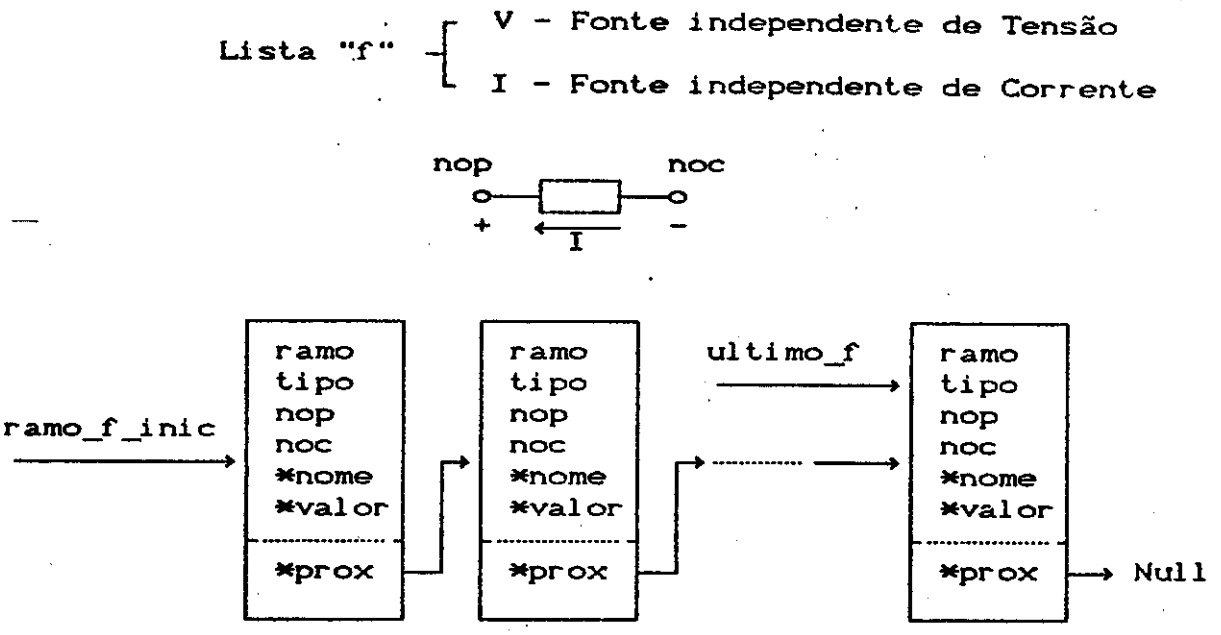
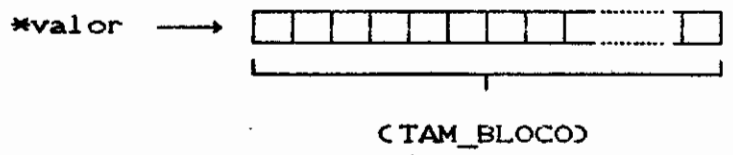


Fig. 5.8 - Estrutura de dados da lista "f".
(Lista encadeada alocada dinamicamente)

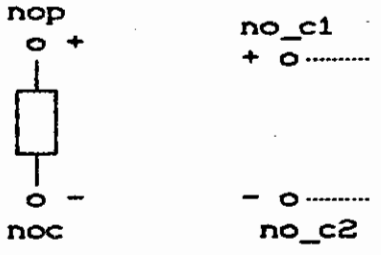
À exceção do campo *valor, todos os outros campos da estrutura de dados que compõem a lista "f" tem função semelhante ao da lista "p". Naturalmente o campo tipo do componente assumirá o valor V ou I.

O campo *valor é um ponteiro que aponta para um vetor de tamanho TAM_BLOCO (constante pré-definida). Este vetor será preenchido durante a simulação com os valores das amostras que descrevem a fonte correspondente. Se a fonte for DC os valores das amostras são preenchidos apenas uma vez, não tendo alteração de seus valores durante a simulação. No caso genérico, antes da simulação de um novo conjunto de amostras, TAM_BLOCO amostras são lidas do arquivo de sinal que define a fonte e colocadas neste vetor. As fontes independentes de tensão têm suas amostras lidas em Volts e as fontes de corrente tem suas amostras lidas em Ampères.



Na figura seguinte apresenta-se a estrutura de dados que armazena as fontes de tensão controladas por tensão e as fontes de corrente controladas por tensão. Estas duas fontes têm em comum o fato de terem a tensão entre dois nós controlando o valor da fonte. São portanto necessários mais dois parâmetros para a completa especificação destas fontes.

Lista "v" — E - Fonte de Tensão controlada por Tensão
Lista "v" — G - Fonte de Corrente controlada por Tensão



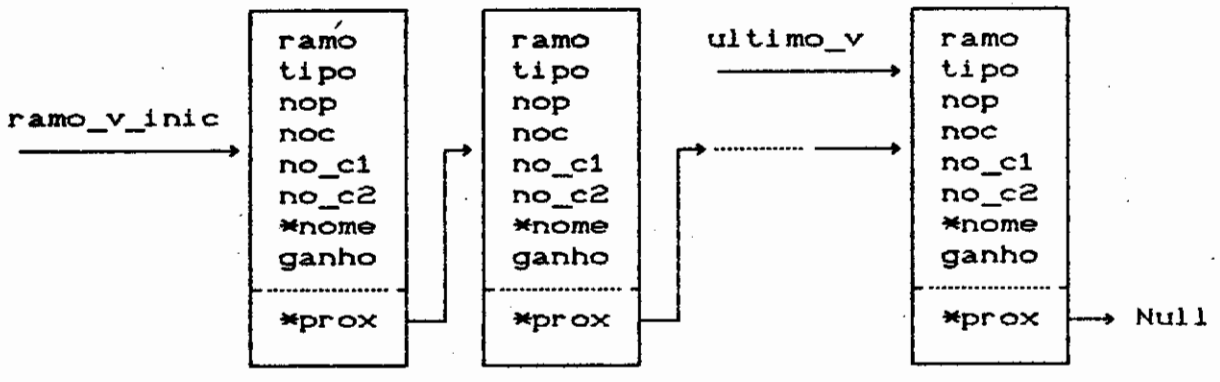


Fig. 5.9 - Estrutura de dados da lista "v".
(Lista encadeada alocada dinamicamente)

Os campos no_c1 e no_c2 da estrutura de dados das listas "v" especificam os dois nós controladores das fontes controladas por tensão e por corrente. O campo ganho é de natureza adimensional no caso da fonte de tensão controlada por tensão e tem dimensão de corrente por tensão no caso da fonte de corrente controlada por tensão. Os demais campos desta estrutura de dados têm as mesmas funções descritas anteriormente para o caso da lista "p".

A lista "c" contém as fontes controladas por corrente, de acordo com a fig. 5.10.

Lista "c" — H - Fonte de Tensão controlada por Corrente
F - Fonte de Corrente controlada por Corrente

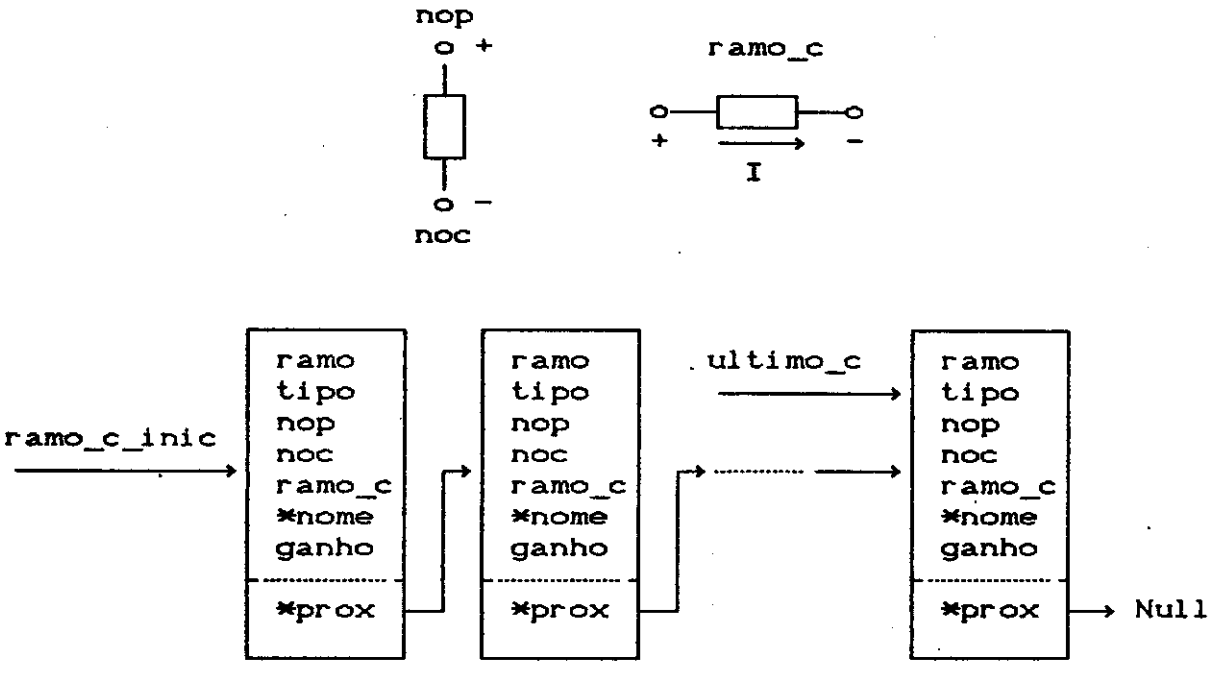


Fig. 5.10 - Estrutura de dados da lista "c".
(Lista encadeada alocada dinamicamente)

Nesta lista aparece o campo "ramo_c" que fornece o ramo cuja corrente controla as fontes controladas por corrente. A polaridade da corrente será a especificada na definição do ramo controlador, em outra linha do arquivo de circuito. O campo ganho tem a dimensão de transresistência para a fonte de tensão controlada por corrente e é adimensional para a fonte de corrente controlada por corrente. Os demais campos da estrutura de dados têm mesma função que nas listas já descritas.

A quantidade de componentes que estão definidos em uma ferramenta de simulação profissional é geralmente suficiente para descrever a maioria dos circuitos utilizados convencionalmente. No presente trabalho, tendo em vista a finalidade acadêmica, estão definidos apenas alguns componentes além dos já apresentados. Alguns modelos de transistores e um modelo para o amplificador operacional são propostos, que permitem simplificar a simulação de alguns circuitos lineares mais complexos.

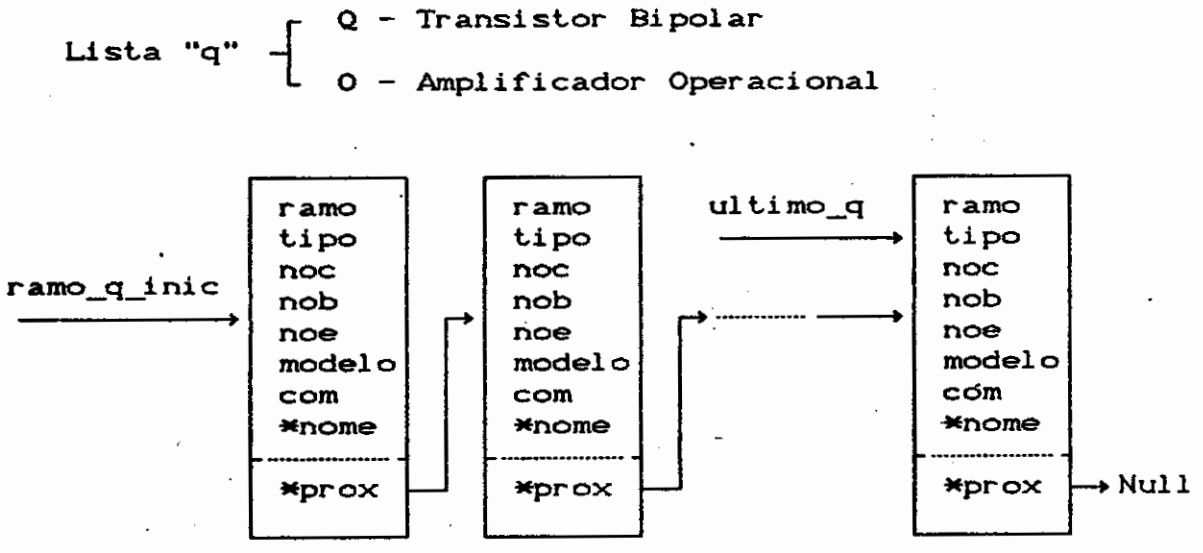


Fig. 5.11 - Estrutura de dados da lista "q".
(Lista encadeada alocada dinamicamente)

A especificação de um transistor é feita pelos seguintes campos:

noc - nó do coletor

nob - nó da base

noe - nó do emissor

modelo - este campo define o modelo que será utilizado para descrever o transistor. Os modelos híbrido simplificado, híbrido completo, pi-híbrido em baixa freqüência e pi-híbrido estão definidos.

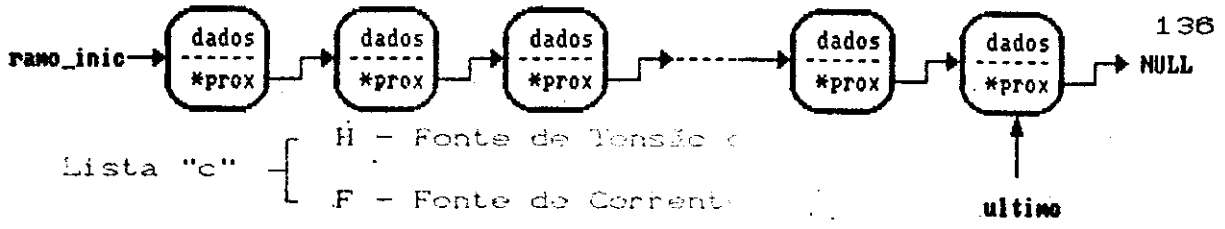
Lista "c"

com - este campo permite selecionar os valores dos componentes do modelo utilizado. Assim alguns componentes comerciais podem ser simulados.

5.2.2 - Algoritmo de Inserção de Componente em Lista

As estruturas de dados apresentadas na seção anterior são utilizadas pelo algoritmo de leitura de circuito da figura 5.2. Como são estruturas de dados alocadas dinamicamente é necessário haver durante o tempo de execução do algoritmo um gerenciamento da memória disponível. Todas as listas que armazenam o circuito internamente ao computador utilizam procedimentos idênticos para inserir um componente novo em lista. Assim sendo a fig. 5.12 apresenta o algoritmo de inserção de componente em lista que se aplica a todas as listas de componentes da seção anterior.

No início da figura vê-se uma representação de uma lista encadeada como blocos do qual partem setas. Os blocos representam os dados armazenados por cada elo da lista. As setas indicam que cada bloco dispõe da informação necessária para acessar o próximo elemento da lista.



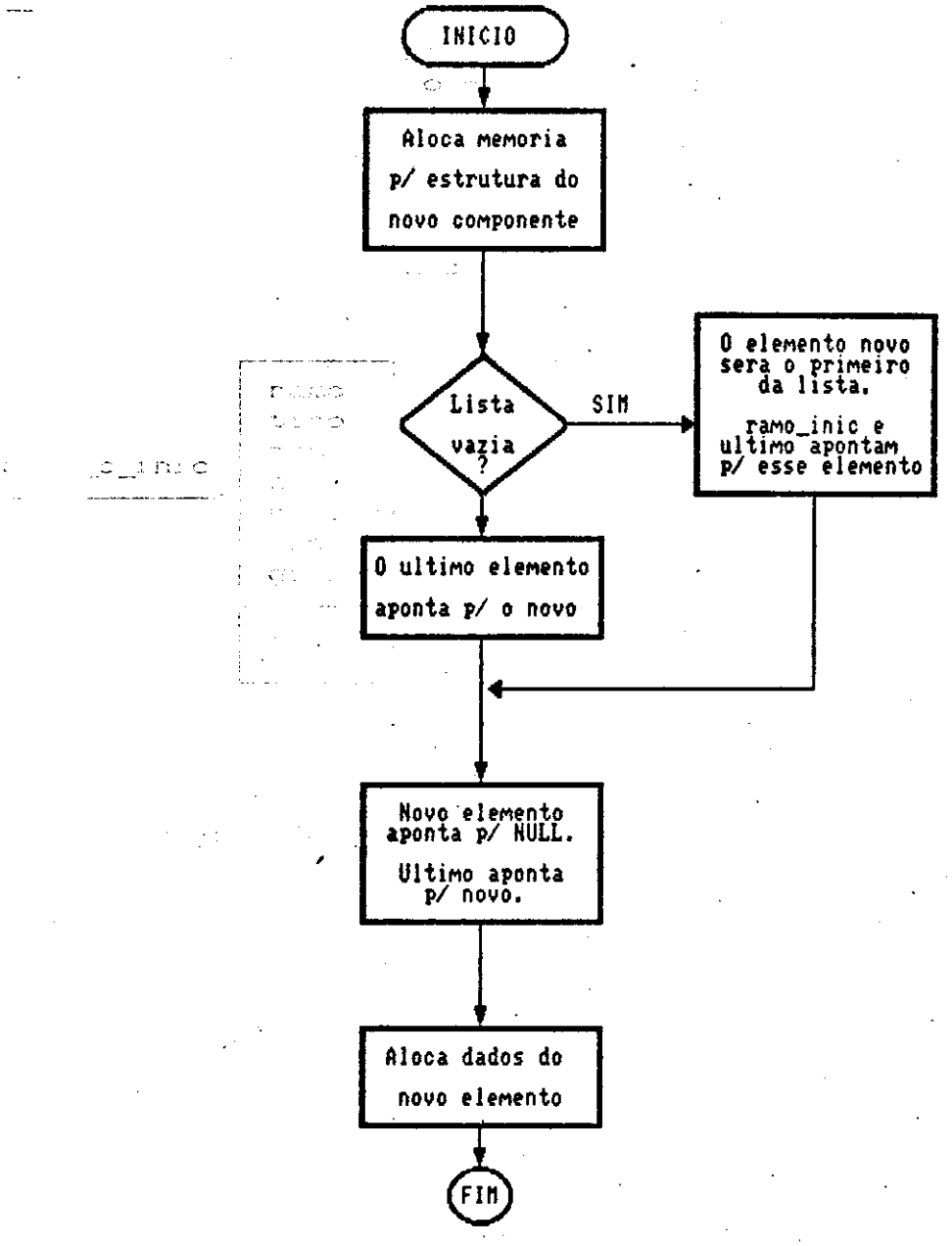


Fig 5.12 - Algoritmo de Insercao de Componente em Lista

A inserção de um novo componente em uma das listas de dados têm início com a alocação de memória para a estrutura do novo componente do circuito. Caso a lista esteja vazia, ou seja, o componente que acabou de ser lido será o primeiro da lista, então os ponteiros auxiliares "ramo_inic" e "ultimo" apontam para o novo elemento alocado. Caso a lista já contenha componentes então o componente novo é colocado no final da lista. O elemento novo aponta para NULL e o ponteiro auxiliar "ultimo" passa a apontá-lo. Após este ajuste dos ponteiros os parâmetros do novo elemento são armazenados nas posições correspondentes.

5.3 - Algoritmo de Expansão de Componente de Biblioteca

Com a finalidade de facilitar a simulação de circuitos com grande número de componentes utiliza-se em geral uma biblioteca de componentes que contém os dispositivos básicos com os quais são construídos os circuitos. A construção do circuito é feita pela especificação da interconexão dos blocos básicos. A ferramenta de simulação executa o tedioso processo de expandir os blocos básicos em componentes elementares cuja "estampa" para construção direta da matriz nodal modificada já é conhecida.

Foi implementado um sistema de biblioteca bastante simples, contendo o transistor bipolar e o amplificador operacional, conforme a figura 5.13. Considera-se que estes dispositivos estejam corretamente polarizados e um modelo linear é adotado, correspondendo à operação na vizinhança do ponto de operação.

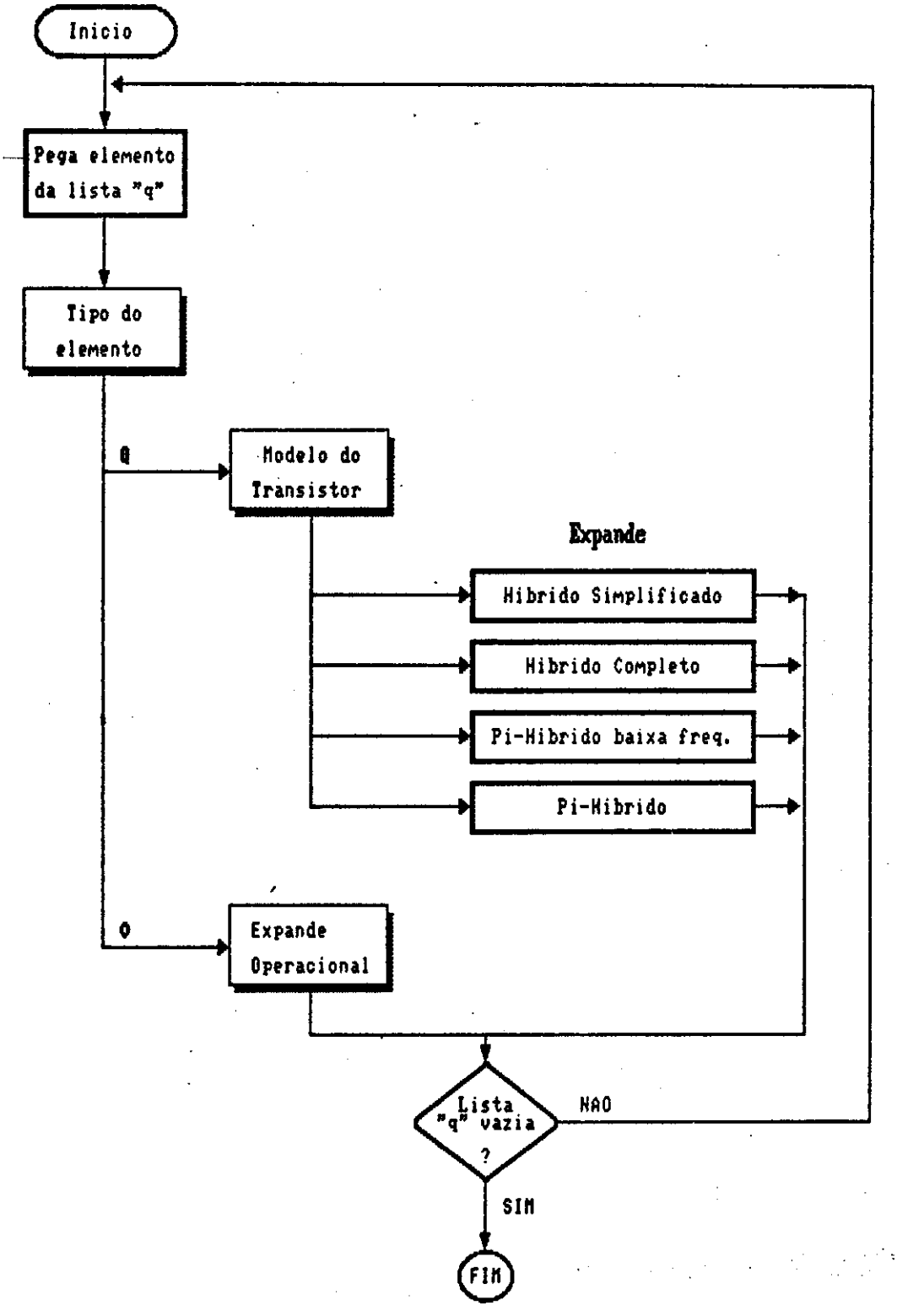


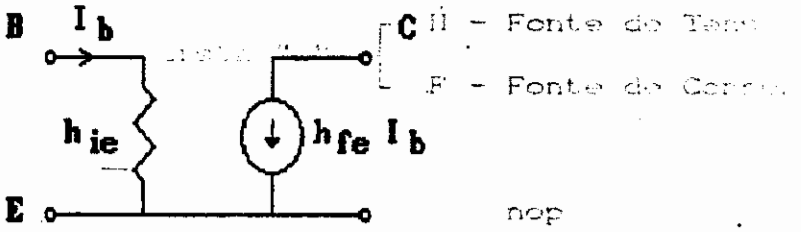
Fig 5.13 - Algoritmo de expansao de componente em biblioteca.

O caso que considera o modelo completo do transistor é bem mais complexo e o tratamento deve considerar a natureza não linear do transistor. Em geral teriam que ser atualizados os parâmetros do transistor em cada iteração da simulação, em função de um novo ponto de operação. Isto forçaria a construção e solução de uma nova matriz nodal modificada. No entanto, a prática mostra que para circuitos corretamente polarizados o modelo linear ora adotado fornece resultados satisfatórios.

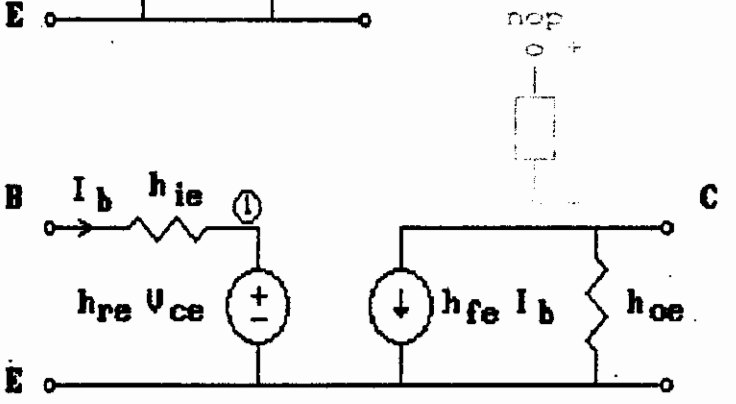
O algoritmo de expansão de componentes da biblioteca toma os dispositivos contidos na lista "q" e os expande conforme o modelo para o caso dos transistores bipolares. O amplificador operacional que foi implementado só dispõe de um modelo para expansão. Este procedimento é repetido para todos os componentes da lista "q".

O modelo do transistor a ser utilizado na expansão é definido no arquivo de entrada. Estão disponíveis quatro modelos. Os valores dos componentes são função do parâmetro "com", conforme descrição da fig. 5.11 da seção de estruturas de dados. A seguir (fig. 5.14) apresenta-se a topologia dos modelos do transistor, omitindo-se os valores dos componentes expandidos por serem função de cada transistor específico.

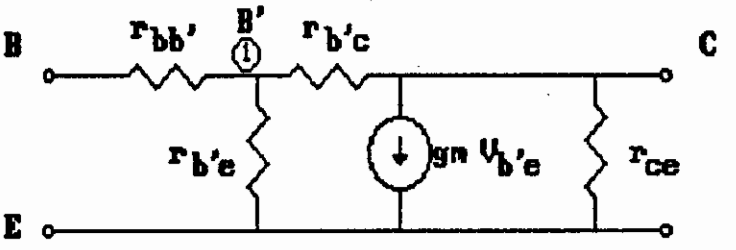
A expansão do modelo híbrido simplificado (fig 5.14 - a) substitui o transistor por um resistor (hie) entre o nó da base e o nó de emissor, e uma fonte de corrente controlada pela corrente de base (hfe) entre o nó de coletor e o nó de emissor. Este modelo simples não acrescenta nenhum nó ao circuito. Os demais modelos do transistor acrescentam um nó interno (B') à topologia do circuito, o que acarreta o aumento da ordem da matriz nodal modificada.



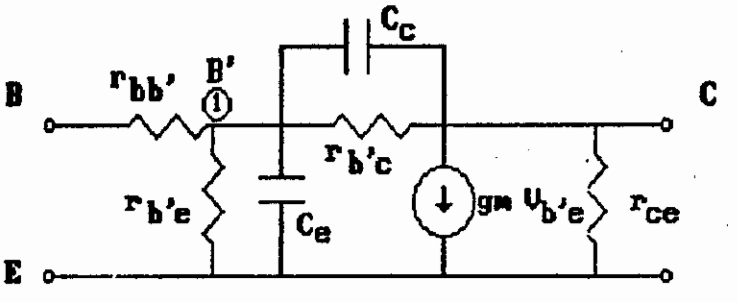
a) Hibrido Simplificado



b) Hibrido Completo



c) Pi-Hibrido Simplificado



d) Pi-Hibrido Completo

Fig. 5.14 - Modelos do Transistor

A expansão do modelo híbrido completo (fig 5.14 - b) soma uma fonte de tensão controlada pela tensão Vce e um resistor hoe ao modelo híbrido simplificado, sendo uma melhor descrição do modelo de pequenos sinais de um transistor.

O modelo pi-híbrido simplificado (fig. 5.14 - c) utiliza uma topologia contendo quatro resistores e uma fonte de corrente controlada pela tensão entre B' e E.

O modelo pi-híbrido completo (fig. 5.14 - d) acrescenta dois capacitores ao modelo pi-híbrido simplificado sendo uma descrição apropriada para um modelo de pequenos sinais em alta freqüência para um transistor bipolar.

Quanto ao esforço computacional destes quatro modelos nota-se que o modelo híbrido completo, além de acrescentar um nó à topologia original, também exige que a corrente no resistor "hie" seja incluída no conjunto das variáveis nodais modificadas. Pelo fato dos modelos pi-híbridos conterem uma fonte de corrente controlada por tensão não é necessário acrescentar-se nenhuma corrente ao conjunto de variáveis nodais modificadas.

Logo a seguir (fig. 5.15) mostra-se o modelo do amplificador operacional implementado. Este modelo é uma versão linearizada do modelo que é utilizado pelo PSPICE para o uA741. Os números que aparecem próximos aos nós deste circuito são os nós que são acrescentados à topologia original do circuito pelo algoritmo de expansão do amplificador operacional. Os transistores Q1 e Q2 deste modelo são internamente expandidos pelo modelo pi-híbrido completo da fig. 5.14 - d.

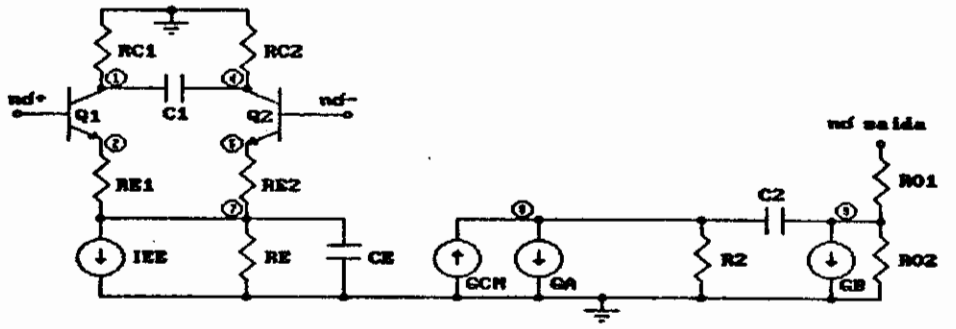


Fig. 5.15 - Modelo do Amplificador Operacional

Devido aos nós internos dos transistores Q1 e Q2 os nós 3 e 6 estão omitidos no modelo do amplificador operacional. A fonte I_{EE} é uma fonte independente de corrente. A fonte de corrente GCM é controlada pela tensão entre o nó interno 7 e a referência (terra). A fonte de corrente GA é controlada pela tensão entre os coletores de Q2 e Q1 (nós 4 e 1). A fonte de corrente GB é controlada pela tensão entre o nó 8 e a referência. O capacitor de valor 30 pF utilizado para compensação interna do uA741 é modelado pelo capacitor C2.

Conforme visto por este modelo a especificação de um amplificador operacional contido em um circuito é feita por três nós (nó+, nó- e nó saída). Por sua vez o algoritmo de expansão produz nove nós internos e vinte e nove componentes elementares (que têm estampa definida) ligados a eles. Isto ilustra a grande simplificação obtida na especificação do circuito a simular quando é disponível um algoritmo de gerenciamento de biblioteca.

5.3.1 - Algoritmo para Salvar a Expansão da Biblioteca

O circuito interno gerado pela expansão dos componentes da biblioteca pode ser salvo em arquivo texto pelo algoritmo descrito por fluxograma da figura seguinte (fig. 5.16). O arquivo de texto pode ser utilizado para verificação ou para alteração de algum parâmetro interno de um modelo. Esta última utilização pode ser útil no sentido de simular um circuito contendo um transistor ou amplificador operacional que não esteja entre os pré-definidos na biblioteca.

O algoritmo toma cada uma das listas de componentes e salva cada componente com a mesma sintaxe utilizada durante a leitura. Este algoritmo tem a característica de agrupar os componentes da mesma categoria. O arquivo original pode ter os componentes declarados em qualquer ordem, mas durante o processo de leitura os componentes são colocados em suas respectivas listas. A ordem em que os componentes aparecem no arquivo de texto com a expansão da biblioteca segue a ordem mostrada no algoritmo. Assim as fontes independentes aparecem primeiro, seguidas dos componentes das listas "p", lista "v" e finalmente lista "c".

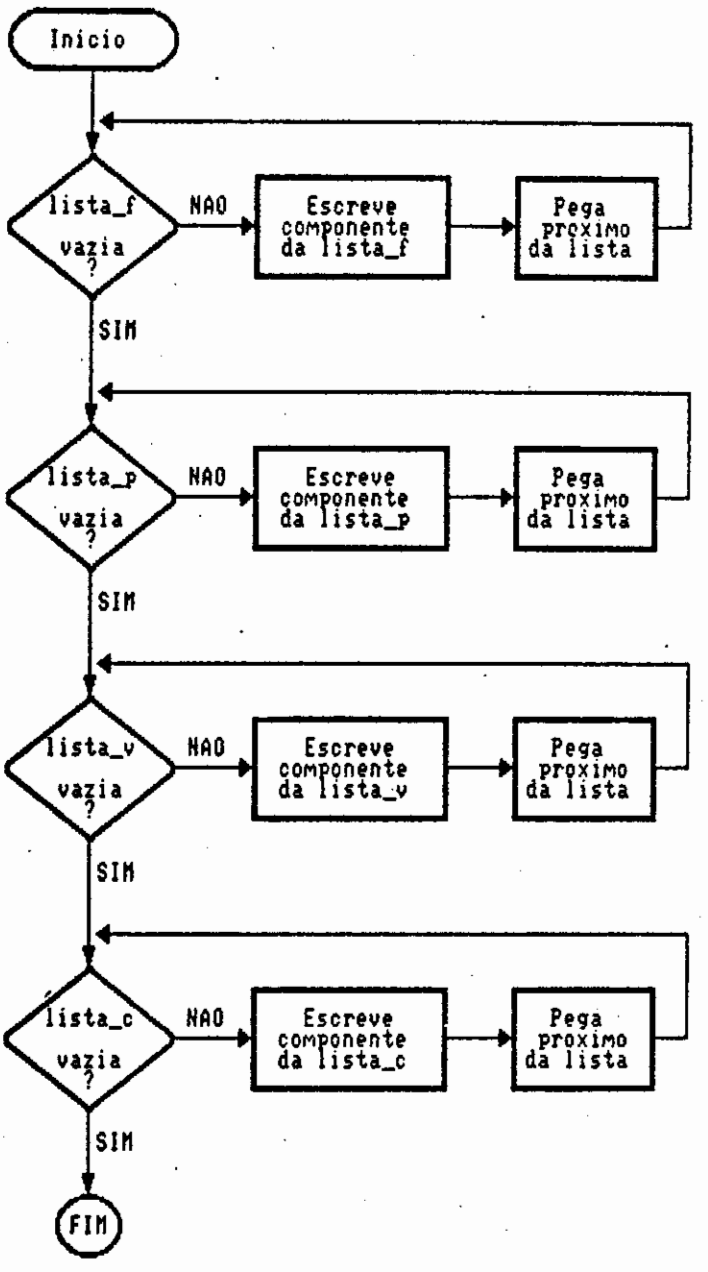


Fig 5.16 - Algoritmo para salvar expansao em Biblioteca

5.4 - Algoritmo para a Construção do Sistema de Equações Nodais Modificadas

A construção direta da matriz nodal modificada de um circuito (Fig. 5.17) pode ser feita em duas etapas principais. A primeira consiste em determinar a dimensão da matriz nodal modificada e a posição que as equações nodais e as equações constitutivas ocupam na matriz. A segunda etapa toma a contribuição de cada componente do circuito e a adiciona nas devidas posições da matriz.

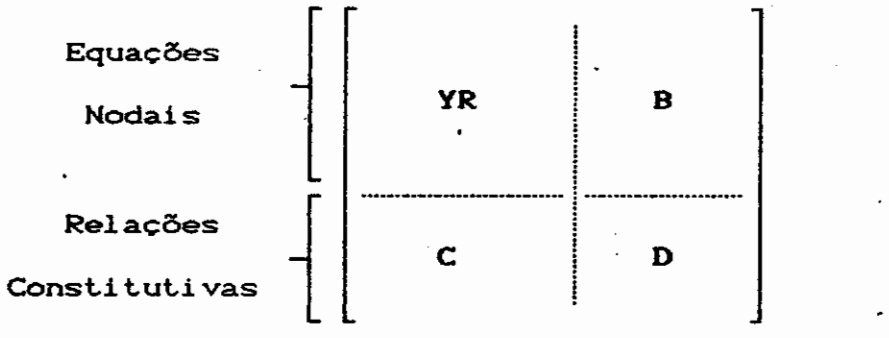


Fig. 5.17 - Equações que compõem a matriz nodal modificada.

5.4.1 - Determinação das Variáveis Nodais Modificadas

A dimensão da matriz é a soma do número de equações nodais com o número de relações constitutivas. Cada nó do circuito contribui com uma equação nodal, à exceção do nó de referência (pois é combinação linear das demais). O número de relações constitutivas necessárias à completa descrição do circuito pode ser determinado construindo-se uma lista contendo os seguintes ramos:

1) Ramos cuja corrente seja solicitada como saída
2) Ramos indutivos

3) Ramos capacitivos com discretização trapezoidal
4) Ramos de fontes independentes de tensão
5) Ramos de fontes de tensão controladas
6) Ramos cuja corrente controla uma fonte controlada.

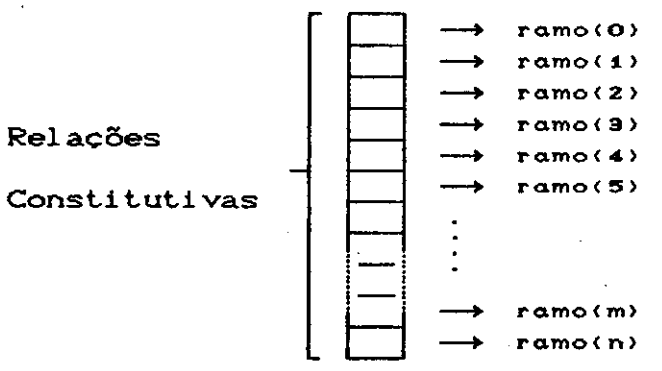


Fig. 5.18 - Lista de ramos cuja corrente será calculada: (O índice em parênteses indica apenas a ordem em que os ramos são acrescentados à lista)

A obtenção dos ramos que têm sua corrente incluída na matriz nodal modificada pode ser feita tomando-se cada ramo do circuito e verificando se o mesmo satisfaz uma das seis condições acima. Cada ramo que satisfaz uma das condições é colocado na lista, conforme ilustrado na Fig. 5.18. No entanto, como alguns componentes podem satisfazer mais de uma condição (como um indutor cuja corrente controla uma fonte controlada) é preciso remover os elementos duplicados da lista de ramos.

Após estes procedimentos ficam estabelecidas as variáveis da matriz nodal modificada (Fig. 5.19). Durante a simulação as variáveis de interesse são armazenadas em arquivo. A especificação destas variáveis de interesse a serem exteriorizadas em arquivo é feita pelo arquivo de entrada. A especificação é feita pelos nós sobre os quais aparecem tensões de interesse e, no caso de correntes, pela

especificação dos ramos percorridos por correntes de interesse. Uma vez especificadas as variáveis de interesse, o algoritmo de construção da matriz nodal modificada a constrói com a menor dimensão possível. Por este motivo, diferentes conjuntos de variáveis de saída para um mesmo circuito podem produzir diferentes matrizes nodais modificadas.

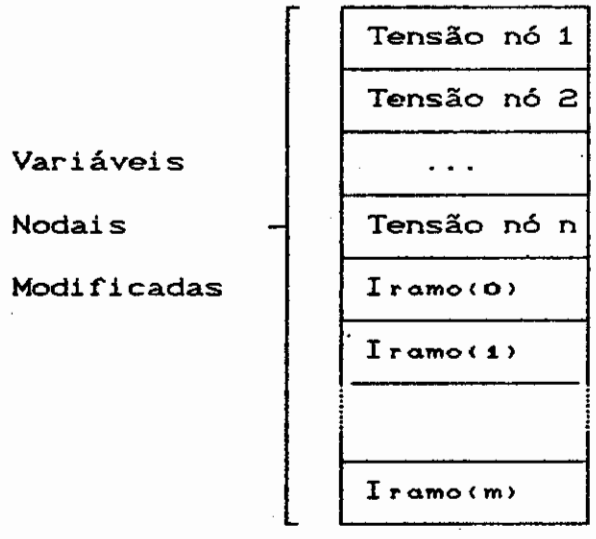


Fig. 5.15 - Variáveis nodais modificadas

Para que as variáveis de interesse sejam armazenadas precisa-se das posições na matriz nodal modificada ocupadas por estas variáveis. Para esse fim pode ser criado um vetor com as posições na matriz que contêm as variáveis que serão exteriorizadas. Por outro lado, durante o processo de solução pelo método da bifatoração ocorrem permutações de equações (linhas da matriz) quer no sentido de preservar a esparsidade como para evitar pivotamento nulo. Assim, o vetor de posições das variáveis de interesse deve acompanhar as permutações para refletir corretamente a posição dessas variáveis.

5.4.2 - Construção Direta das Matrizes para o Sistema de Equações Nodais Modificadas

A segunda etapa, de construção da matriz nodal modificada propriamente dita, será considerada agora. Conforme já apresentado, a discretização dos modelos para a formulação nodal modificada leva a um sistema de equações da forma (repetição das eq. 4.1 e 4.2):

$$A.v_{n+1} = b \qquad (5.1)$$

$$b = C_0.v_n + C_1.v_{n-1} + C_2.v_{n-2} + C_3.v_{n-3} + f \qquad (5.2)$$

A construção direta das matrizes para o sistema de equações nodais modificadas é feita percorrendo-se cada lista de componentes e acrescentando-se a estampa correspondente ao elemento às matrizes A e C_0 a C_3. Para o capacitor e o indutor a estampa é função do algoritmo de discretização utilizado. Um tratamento diferenciado é dado às estampas se um dos nós aos quais o componente estiver ligado for o nó terra. Neste caso alguns elementos da estampa estarão ausentes da matriz, conforme já mencionado na descrição das estampas no capítulo 3.

Conforme visto na seção sobre leitura do circuito, as fontes são colocadas em uma lista encadeada. Os campos *nop* (nó de partida) e *noc* (nó de chegada) identificam as posições na matriz nodal modificada original. Se durante o processo de bifatoração for necessário permutar algumas linhas, alterando a ordem do sistema de equações, então a posição em que as fontes têm a sua contribuição deve sofrer a mesma permutação. Com isto o sistema de equações permanece consistente.

As fontes de tensão independentes sempre ocupam a

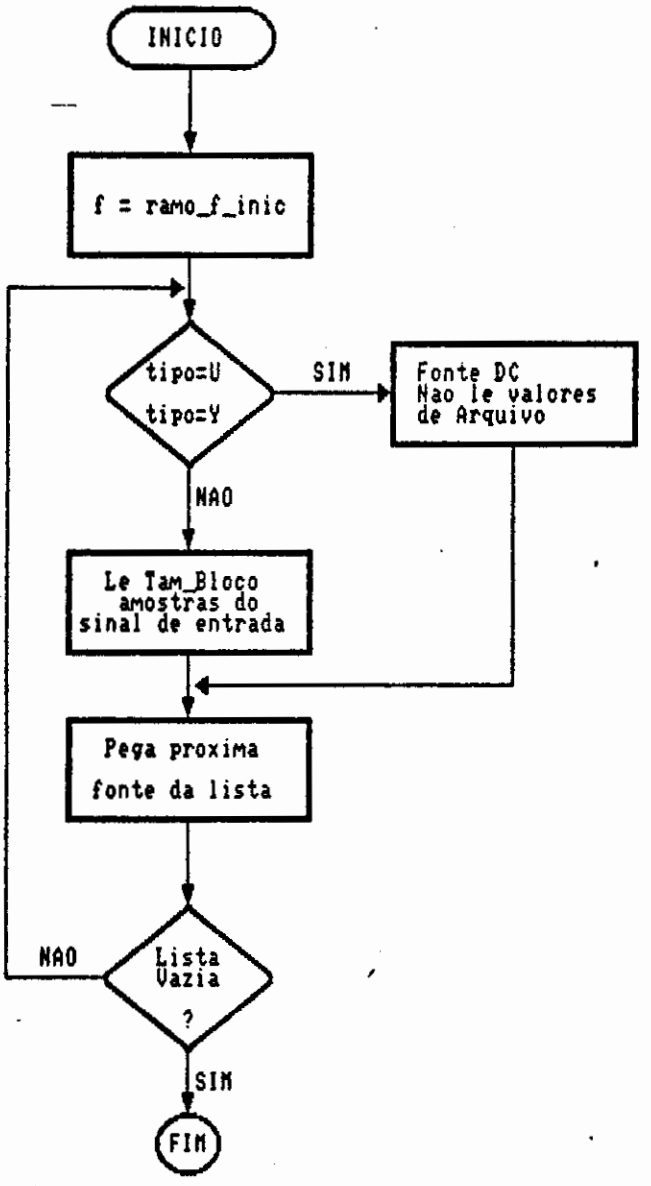
posição híbrida da matriz nodal modificada, e a descrição por relação constitutiva faz com que uma única equação receba a contribuição da fonte de tensão. Por outro lado a contribuição das fontes de corrente aparece na parte nodal das equações. Todas as fontes de corrente que estão ligadas a um nó são somadas ou subtraídas (de acordo com a polaridade da fonte) da equação nodal correspondente.

5.5 - Algoritmo de Leitura dos Sinais de Entrada

Esse algoritmo é mostrado por fluxograma na figura 5.16. Na mesma figura aparece um desenho que mostra a representação interna do arquivo de sinal para o algoritmo de simulação.

Antes que a simulação tenha início é verificada a consistência dos arquivos especificados como fontes de sinais. Um "HEADER" - cabeçalho com informações de controle - é verificado em cada arquivo. Todos os arquivos devem conter o mesmo número de amostras e a freqüência de amostragem deve ser a mesma. Podem ser aceitos arquivos de sinais inteiros ou reais. Internamente todos os sinais são tratados como reais. Conforme visto na figura o número total de amostras contidas no arquivo (NRO_AMOSTRAS) é dividido em blocos de tamanho TAM_BLOCO (constante, pré-definida). Esta estratégia permite que fontes com um grande número de amostras sejam simuladas sem ocupar uma quantidade significativa de posições de memória. Eventualmente o último bloco terá uma quantidade menor de amostras caso o número de amostras total não tenha divisão exata pelo número de amostras de cada bloco.

Todos os arquivos de sinais ficam abertos simultaneamente durante a simulação, e blocos correspondentes são lidos antes do início da simulação de cada bloco, conforme indicado pelos blocos hachurados. A simulação toma



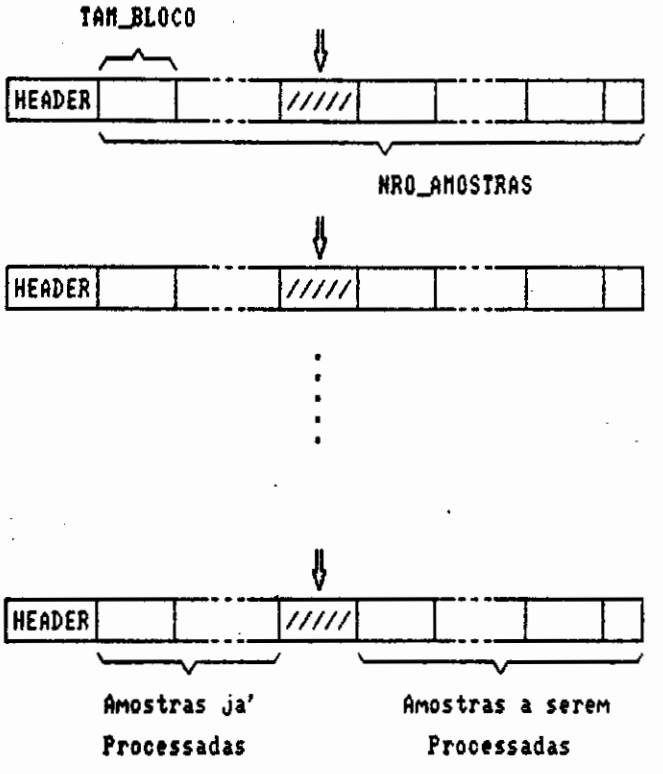


Fig 5.20 - Algoritmo de Leitura dos sinais de entrada

cada uma das amostras de entrada num certo instante e calcula um novo conjunto de variáveis nodais modificadas.

O fluxograma da fig. 5.20 indica que a leitura dos sinais utiliza a lista de fontes (descrita na seção 5.1). A fonte inicial é indicada pelo ponteiro "ramo_f_inic". Uma fonte independente DC tem um tratamento diferente das demais fontes. Durante a inicialização o valor DC é atribuído a todos os campos do vetor de TAM_BLOCO que representa internamente a fonte. O algoritmo de leitura do circuito altera o tipo da fonte de V para U e de I para Y quando a fonte for DC. Isto permite identificar as fontes que não precisam ser lidas de arquivo. A lista de fontes é percorrida até que um novo bloco de amostras de todas as fontes estejam disponíveis.

5.6 - Algoritmo de Escrita das Variáveis de Saída

A figura 5.21 seguinte apresenta um fluxograma simplificado do processo de escrita das variáveis de saída. A lista que contem as variáveis de saída é percorrida e cada variável é escrita no arquivo de saída correspondente.

Conforme indicado no desenho da figura 5.21, para cada passo de solução temporal ($t=t_k$) é obtido um conjunto de variáveis nodais modificadas. Algumas destas variáveis (marcadas com x) são solicitadas como saídas da simulação. Para cada saída é criado um vetor auxiliar que armazena as variáveis de interesse na medida em que vão sendo calculadas. Depois que TAM_BLOCO amostras tenham sido processadas os vetores auxiliares são escritos nos arquivos de saída correspondentes.

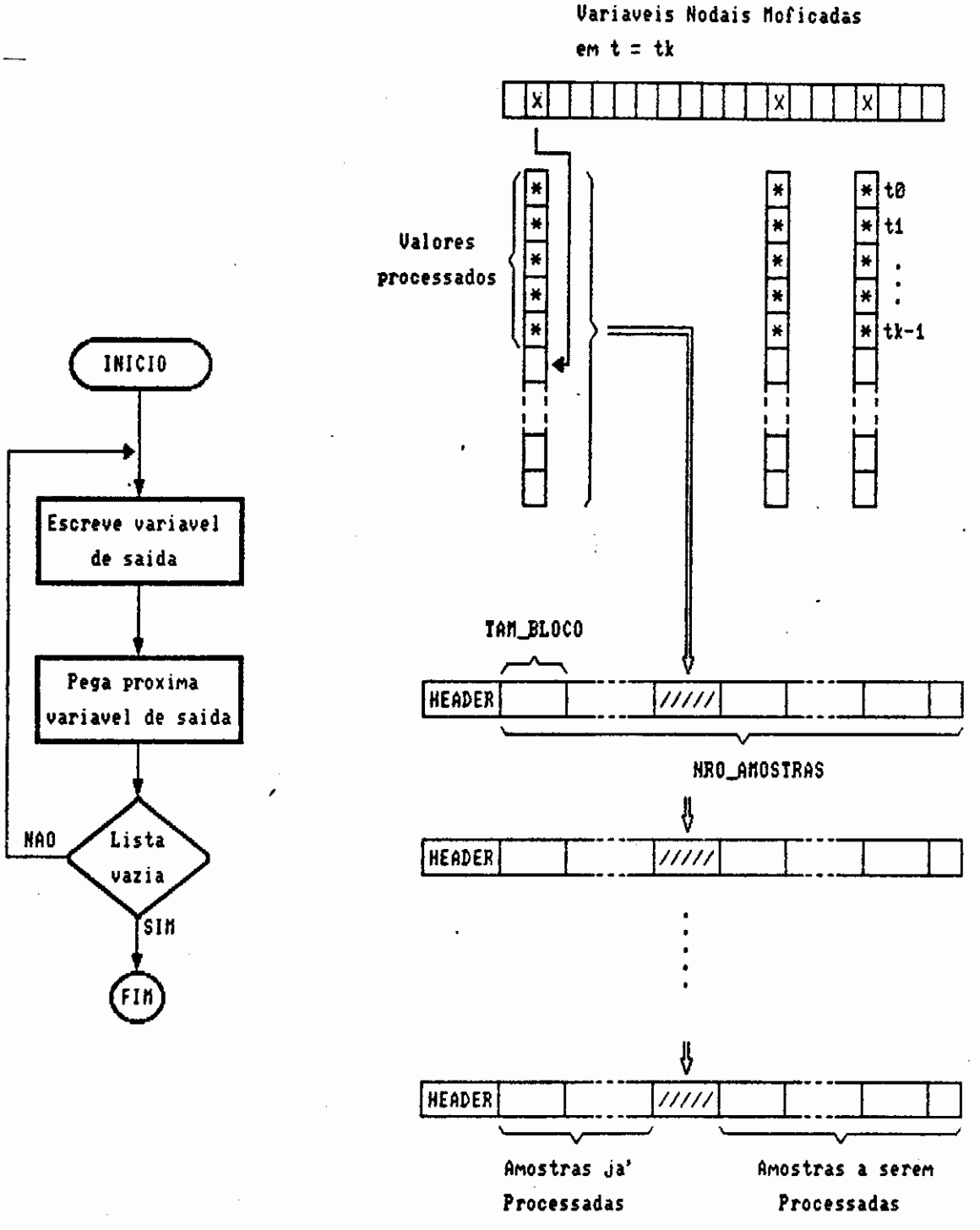


Fig 5.21 - Algoritmo de escrita das variaveis de saida.

6 - Resultados de simulações

A simulação no domínio do tempo procura descrever a operação de um circuito ao longo do tempo submetido à variações dos sinais de entrada. Esta análise de circuitos é uma das mais problemáticas, devido ao compromisso que deve haver entre (i) utilizar passos de integração pequenos para melhorar a precisão (exigindo um grande tempo para completar a simulação), e (ii) utilizar passos de integração grandes com precisão reduzida (e eventualmente perder alguma característica importante na resposta do circuito).

No presente trabalho, o passo de integração é uniforme e definido pelos arquivos dos sinais de entrada. Esta característica torna o Simulador Temporal de Circuitos Lineares adequado a algumas aplicações e restringe a sua aplicação em outras. A simulação de um circuito analógico conectado a sistemas de processamento digital de sinais é um exemplo típico em que se obtém bons resultados. Este é o caso, por exemplo, do projeto de MODEM, no qual se deseja

obter o sinal analógico que será transmitido via canal telefônico. São transmitidos arquivos com dezenas ou até centenas de milhares de amostras. A taxa de amostragem uniforme geralmente adotada em sistemas de processamento digital torna os sinais disponíveis apenas em instantes discretos. O interesse do projetista é pela resposta do circuito analógico a estes sinais.

As características do problema proposto permitem obter soluções bem adequadas ao mesmo. Neste sentido, o passo de integração uniforme condicionou a escolha dos algoritmos implementados nos capítulos anteriores. No Capítulo 3, a discretização das equações pelos algoritmos de integração multipassos permitiu que valores previamente calculados pudessem ser aproveitados. Além disso, devido à amostragem uniforme, só foi necessário contruir (Cap. 2) e fatorar (Cap. 4) a matriz nodal modificada uma única vez durante toda a simulação.

Caso a taxa de amostragem não seja satisfatória para uma certa simulação podem ser adotadas algumas providências, de acordo com a fonte dos sinais de entrada e a finalidade da simulação. Se a simulação utiliza sinais de teste gerados por uma ferramenta computacional (é o caso da grande maioria dos exemplos apresentados neste capítulo) então basta gerar os sinais a uma taxa de amostragem maior. Se os sinais são oriundos da digitalização de um sinal de voz, por exemplo, então freqüentemente não existe interesse em frequências muito elevadas pois a faixa audível é limitada. Eventuais variações entre os instantes de amostragem são perdidas na reprodução ou em amostragens subseqüêntes.

Visando ilustrar a utilização do simulador implementado, bem como comprovar a eficiência e precisão, apresentam-se neste capítulo resultados de simulações de circuitos típicos.

Os resultados são comparados com o PSPICE, que é reconhecido como um padrão em simulação analógica. Teve-se o cuidado em adotar, na descrição do circuito para o PSPICE, os mesmos modelos lineares utilizados no SITECI (abreviatura que será utilizada para denominar o SImulador TEmporal de CIrcuitos lineares implementado)

A comparação do desempenho e precisão com o PSPICE está sujeita a algumas restrições, devido a algumas características distintas dos dois simuladores:

- O PSPICE adapta o passo de integração temporal, enquanto o SITECI utiliza um passo de integração uniforme.

- Os sinais de entrada são gerados internamente pelo PSPICE, enquanto o SITECI utiliza arquivos externos para definição dos sinais.

Todos os exemplos de simulações consideram condições iniciais nulas, i.é, ao iniciar-se a simulação a corrente nos indutores e a tensão nos capacitores é zero. As simulações têm início para $t = 0$, o que não restringe os resultados, uma vez que os circuitos simulados são invariantes com o tempo. Os gráficos mostram "tempo" na horizontal e tensão ou corrente na vertical, conforme indicado em cada caso.

6.1 - Exemplo 1 - Circuito VRC

O primeiro exemplo consiste em um circuito formado por três componentes: uma fonte independente de tensão (V), um resistor (R) e um capacitor (C). Na Fig. 6.1 apresenta-se o esquemático deste circuito, denominado VRC.

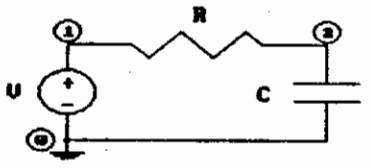


Fig. 6.1 - Esquemático do circuito VRC

Os valores adotados para os componentes foram:

R = 100 Ω
C = 1 μF

Aplicando o algoritmo de Euler Regressivo para discretização das equações do circuito VRC, obtém-se o circuito da Fig. 6.2:

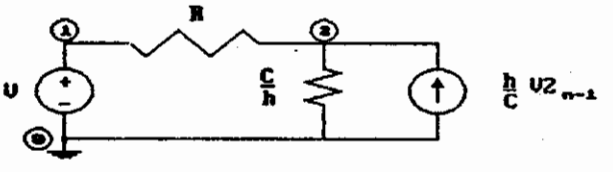


Fig. 6.2 - Circuito VRC discretizado pelo algoritmo de Euler Regressivo

A resposta deste circuito, quando excitado por uma onda quadrada, de freqüência 1 KHz, amplitude 4Vpp, amostrada em 250KHz pode ser vísta na figura 6.3. A fig. 6.3-a apresenta o sinal de entrada, ou seja, a tensão no nó 1. A fig. 6.3-b apresenta a tensão no nó 2, enquanto a fig. 6.3-c apresenta a corrente no capacitor. Os resultados apresentados foram obtidos com o algoritmo de Gear de segunda ordem. Os outros algoritmos de integração implementados fornecem resultados idênticos, em virtude da alta taxa de amostragem em relação à constante de tempo do circuito. Na figura 6.4 têm-se os resultados da simulação do circuito do primeiro exemplo para uma forma de onda dente de serra.

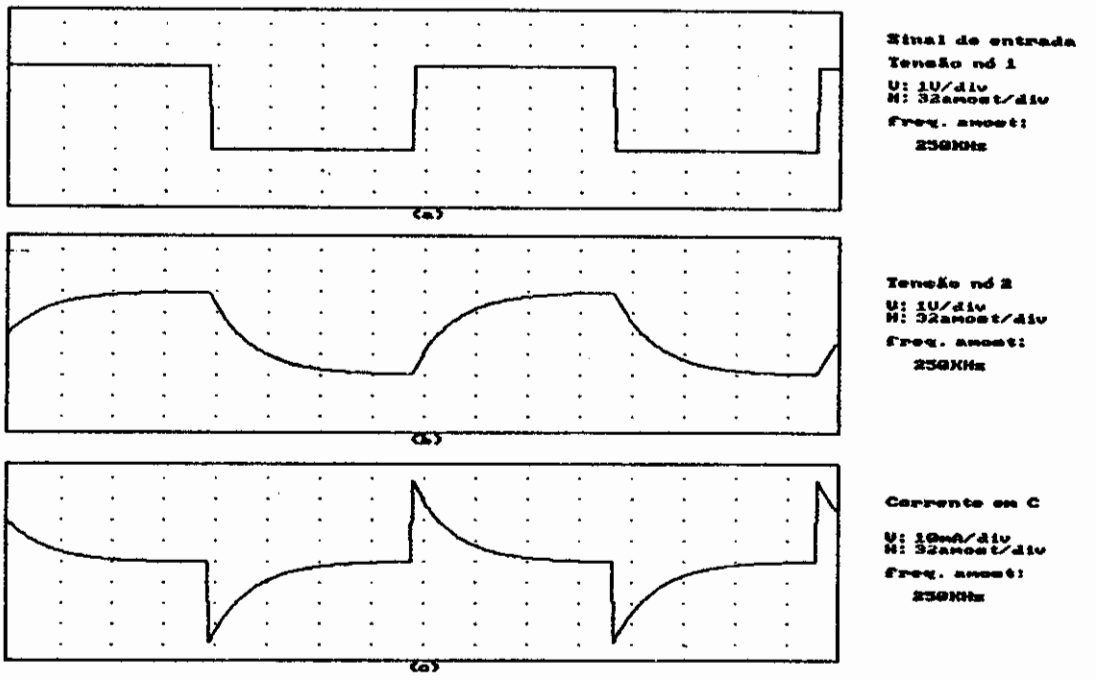


Fig. 6.3 - Simulação do circuito "URC" - Exemplo 1, obtida pelo SITECI.
a) Sinal de entrada, tensão no nó 1
b) Tensao no nó 2
c) Corrente em C

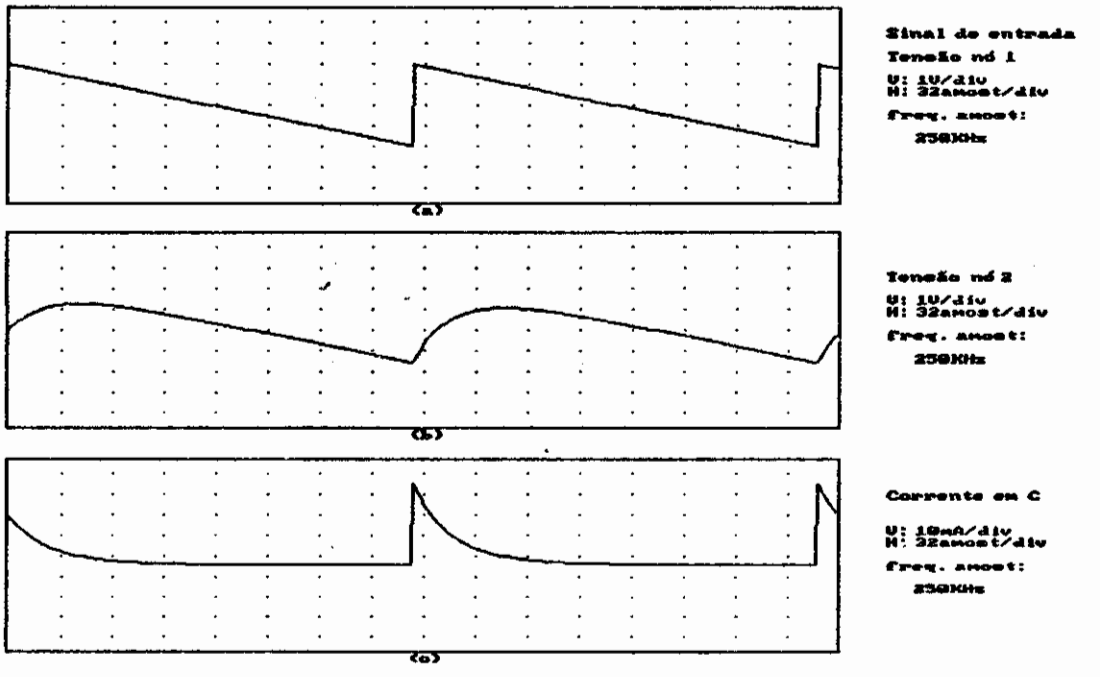


Fig. 6.4 - Simulação do circuito "URC" - Exemplo 1, obtida pelo SITECI.
a) Sinal de entrada, tensão no nó 1
b) Tensao no nó 2
c) Corrente em C

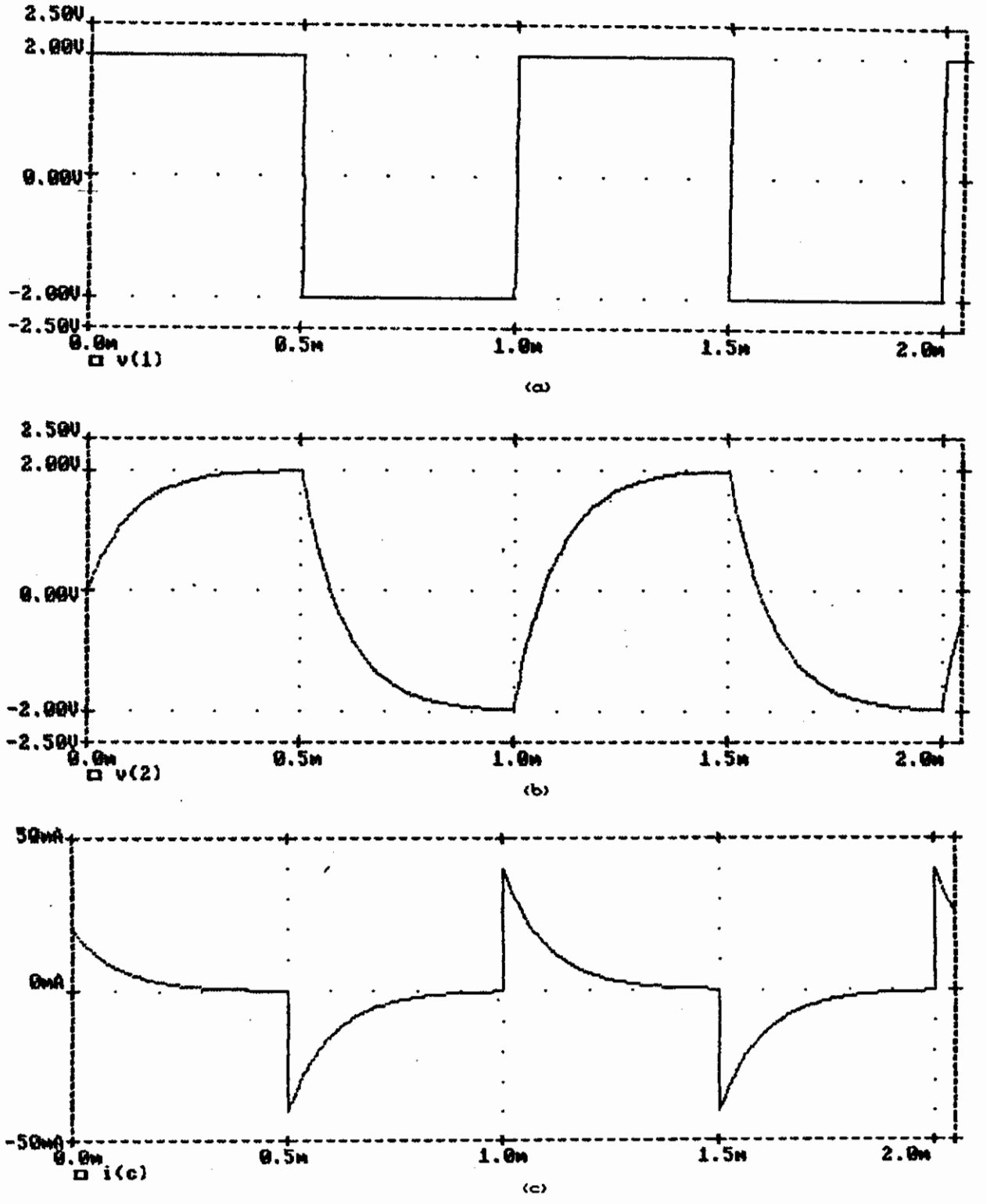


Fig. 6.5 - Simulação do circuito "VRC", obtida pelo PSPICE
a) Sinal de entrada, onda quadrada em 1KHz
b) Tensão no nó 2
c) Corrente em C

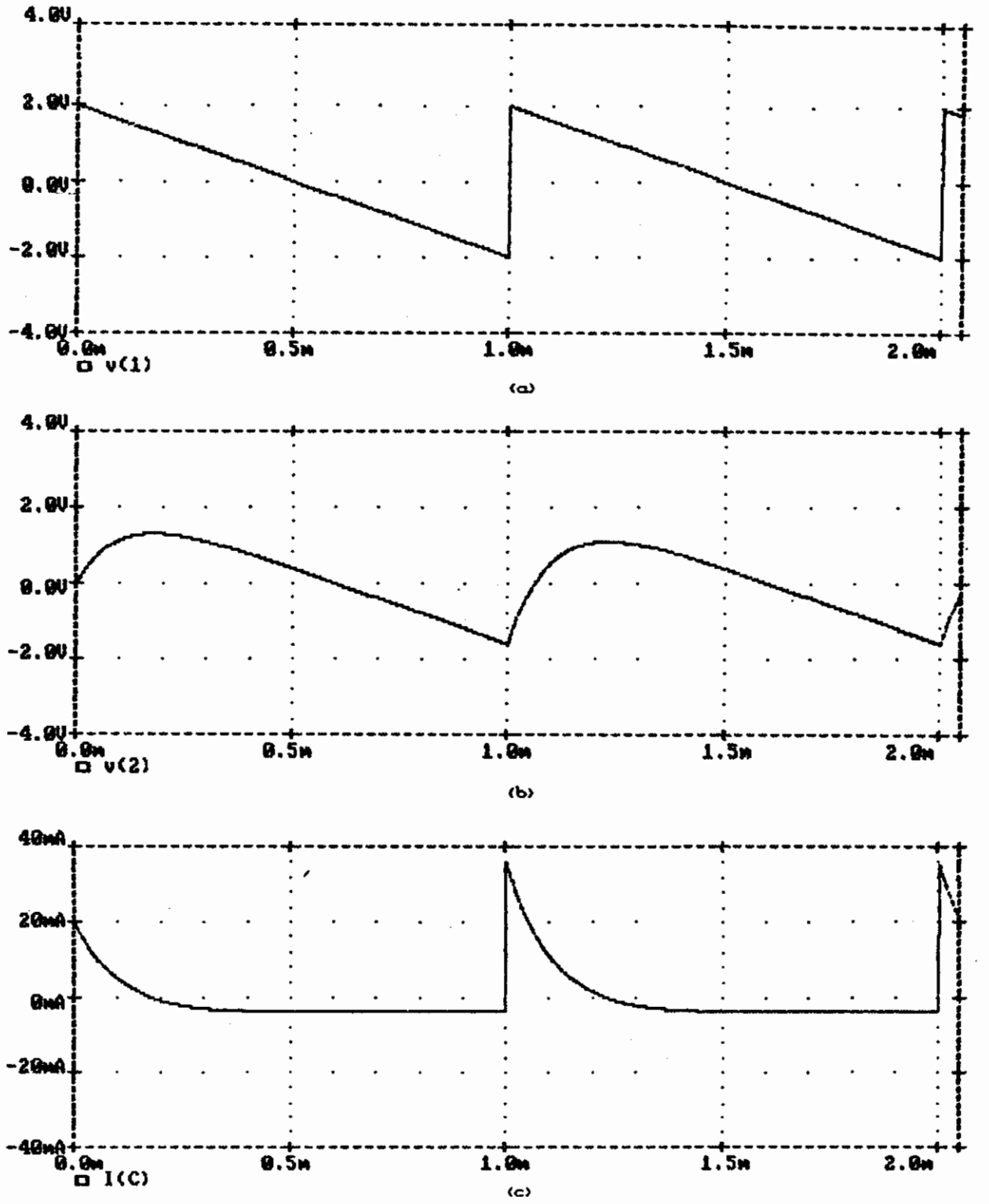


Fig. 6.6 - Simulação do circuito "VRC", obtida pelo PSPICE
a) Sinal de entrada, dente de serra em 1KHz
b) Tensão no nó 2
c) Corrente em C

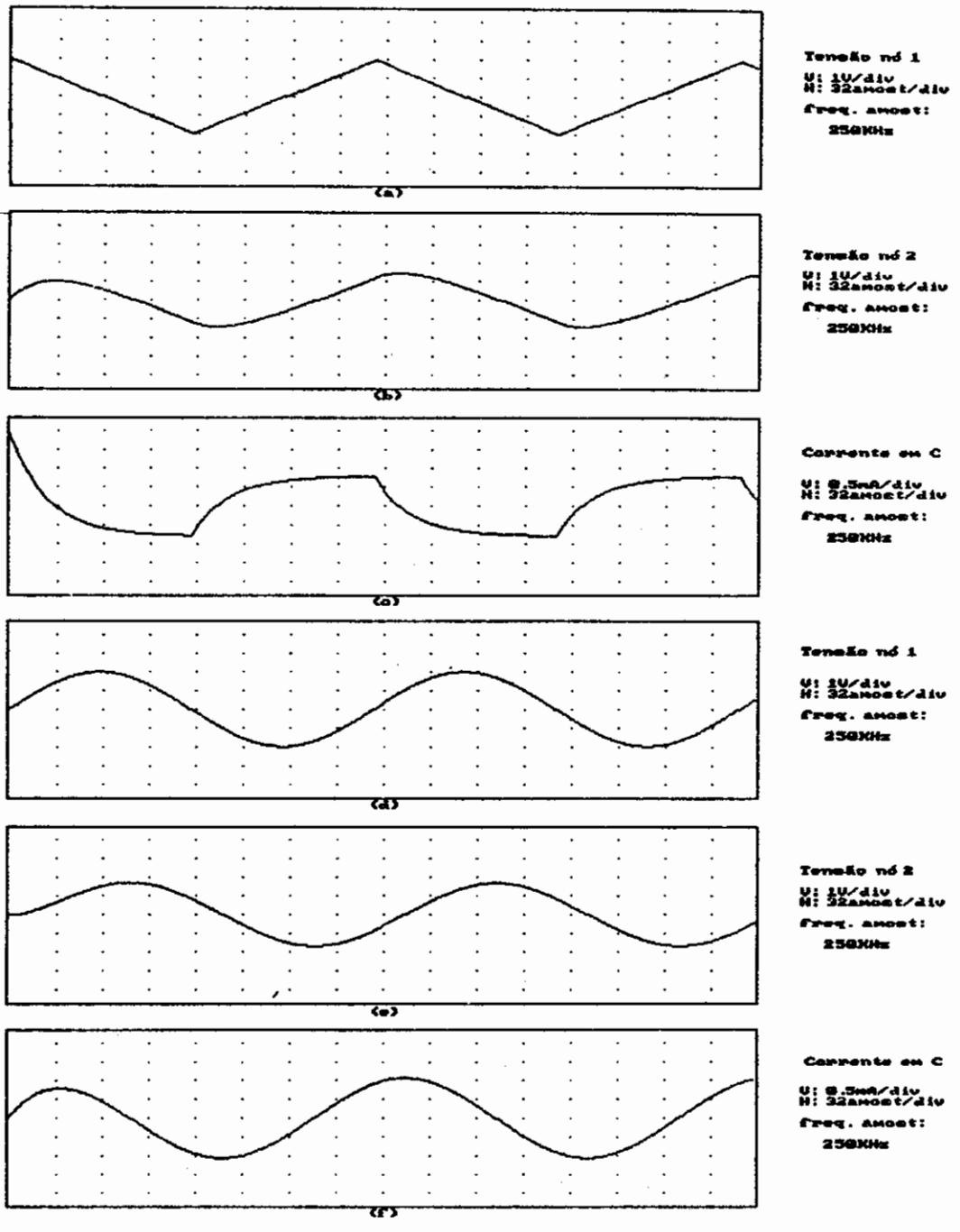
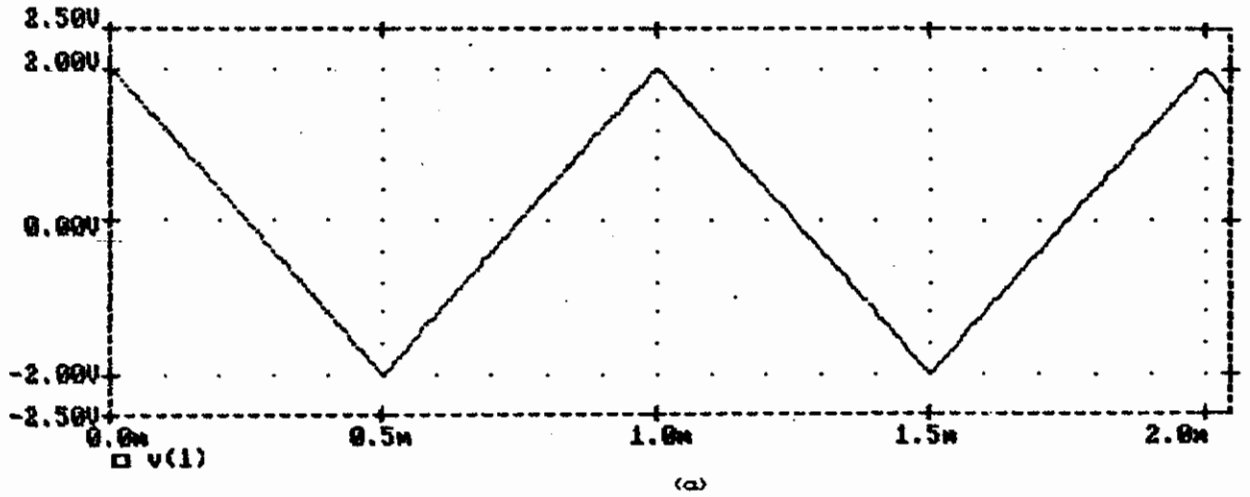
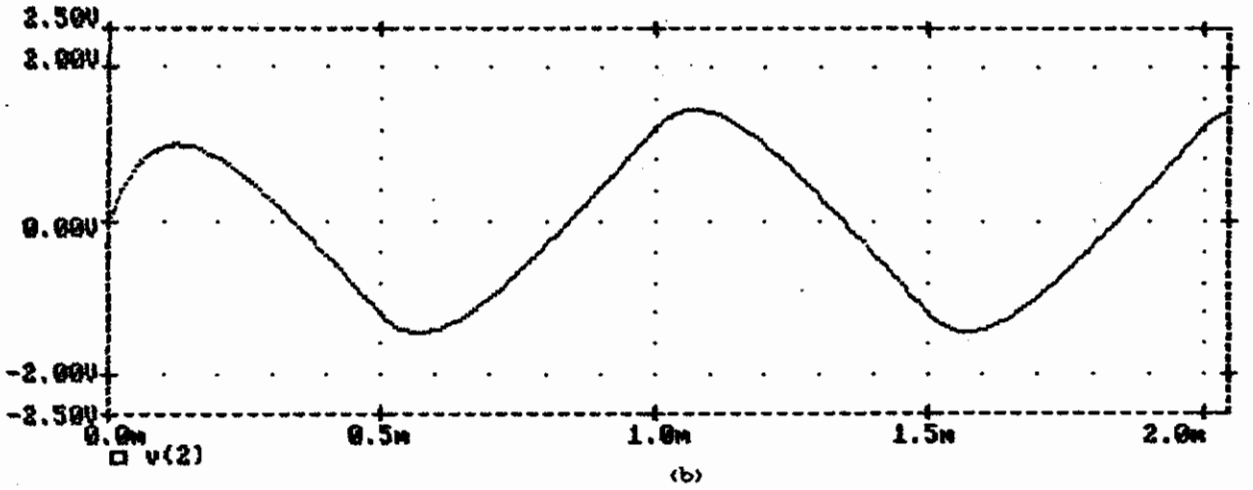


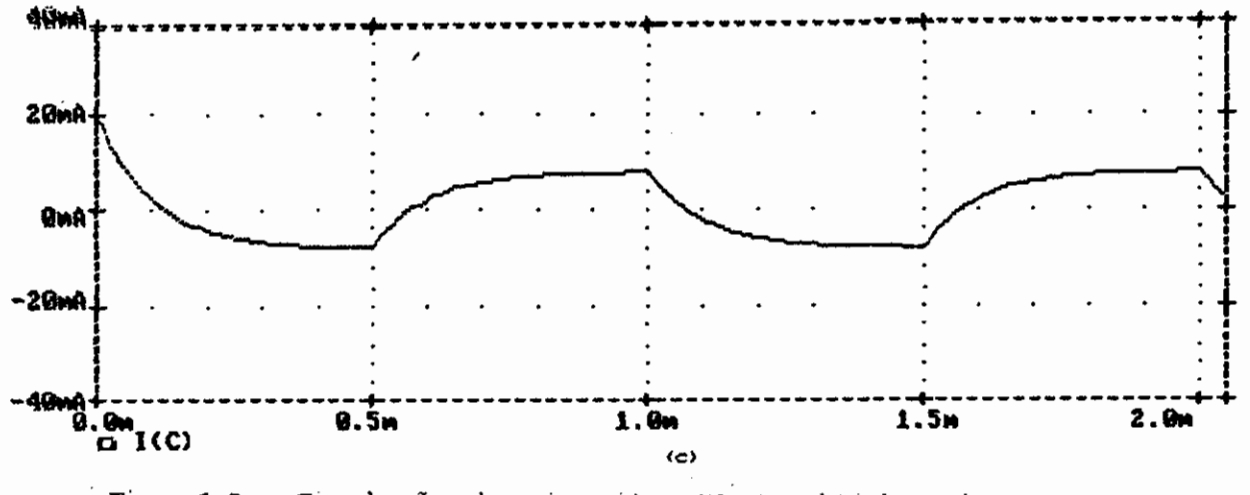
Fig. 6.7 - Simulação do circuito "URC" - Exemplo 1
a) Sinal de entrada triangular
b) Tensão no nó 2, em resposta ao sinal em a)
c) Corrente em C, em resposta ao sinal em a)
d) Sinal de entrada senoidal
e) Tensão no nó 2, em resposta ao sinal em d)
f) Corrente em C, em resposta ao sinal em d)



(a)

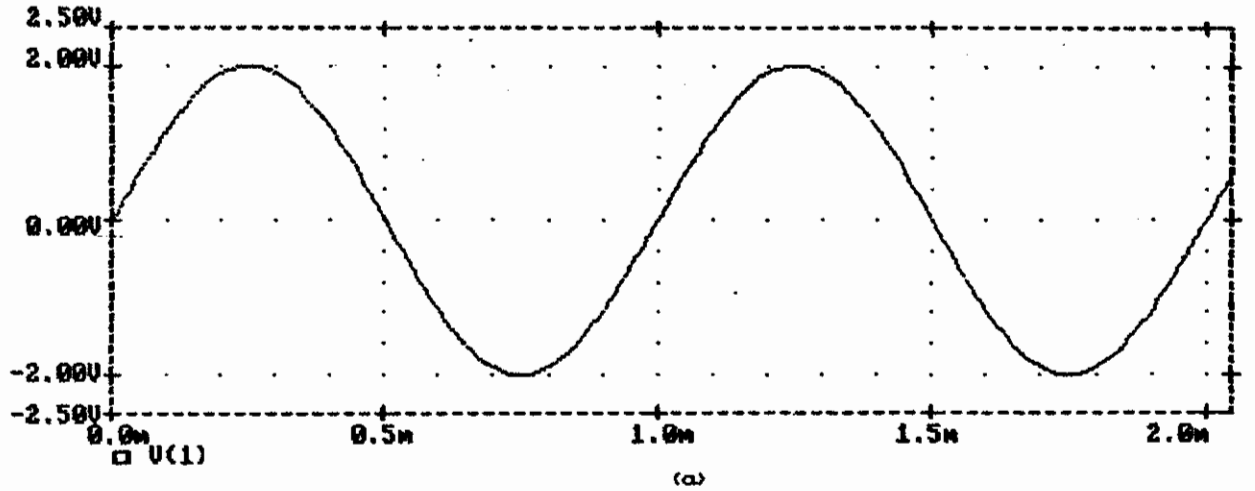


(b)

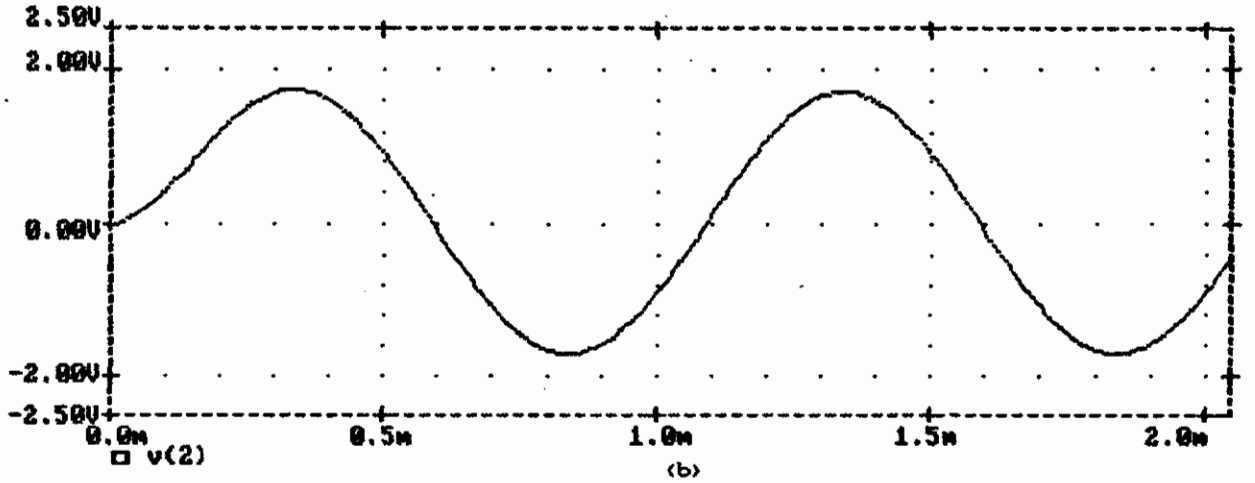


(c)

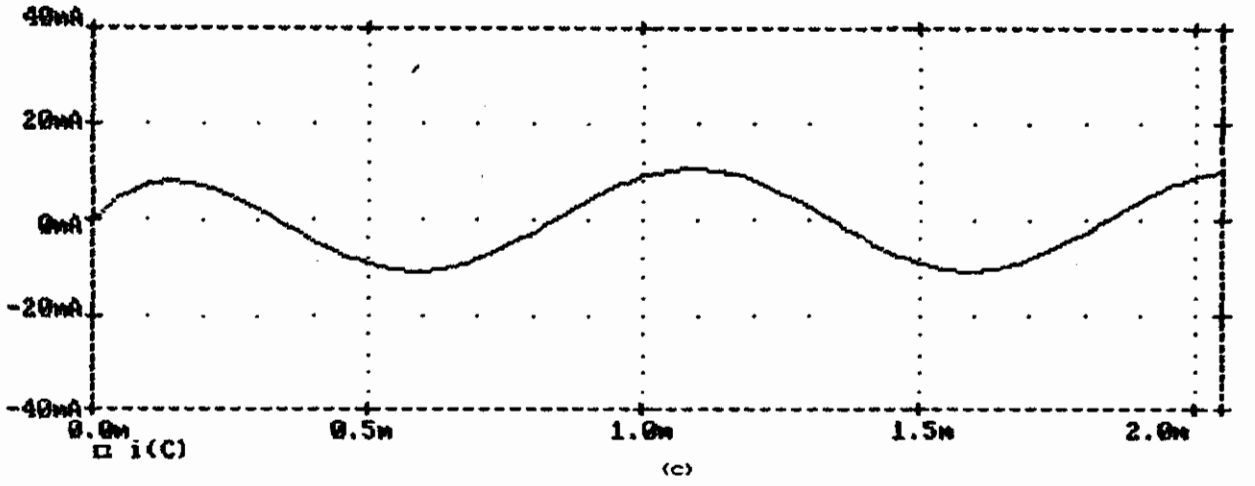
Fig. 6.8 - Simulação do circuito "VRC", obtida pelo PSPICE
a) Sinal de entrada, triangula em 1KHz
b) Tensão no nó 2
c) Corrente em C



(a)



(b)



(c)

Fig. 6.9 - Simulação do circuito "VRC", obtida pelo PSPICE
a) Sinal de entrada, senoidal em 1KHz
b) Tensão no nó 2
c) Corrente em C

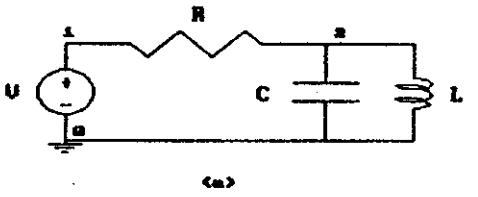
As figuras 6.5 e 6.6 apresentam os resultados da simulação do circuito do primeiro exemplo, obtidos pelo simulador PSPICE, para as formas de onda quadrada e dente de serra. Solicitou-se a saída em arquivo da tensão no nó 2 e da corrente no capacitor a cada 4 μs (mesma taxa fornecida pelo SITECI para sinais amostrados a 250 KHz). Além disso, limitou-se o passo temporal à 8 μs, obtendo-se a resposta em 566 iterações. Utilizando-se um mesmo computador[1], a simulação das 512 amostras com o SITECI consumiu aproximadamente 2 segundos, enquanto o PSPICE, para 566 iterações, consumiu aproximadamente 9 segundos. Conforme mencionado na introdução deste capítulo, as características peculiares dos dois simuladores não permite uma comparação direta. Se a simulação com o PSPICE é feita em seu modo "padrão", então são efetuadas apenas 166 iterações, consumindo aproximadamente 7 segundos. Grande parte do tempo consumido pelo PSPICE é ocupado pela saída de dados.

As figuras 6.7-b e 6.7-c apresentam o resultado da simulação quando o sinal de entrada é uma onda triangular (fig. 6.7-a). A resposta para entrada senoidal (fig. 6.7-d) é vista nas figuras 6.7-e e 6.7-f. A resposta a estas formas de onda obtidas pelo PSPICE aparecem na fig. 6.8 e 6.9, respectivamente.

6.2 - Exemplo 2 - Circuito VRLC

O segundo exemplo consiste em uma fonte de tensão, um resistor (100KΩ), um indutor (10mH) e um capacitor (100nF), conectados conforme a figura 6.10-a. Quando este circuito é excitado por uma onda quadrada de 1KHz (fig. 6.10-b) têm-se os sinais apresentados nas figuras 6.10-c a 6.10-f.

[1] Utilizou-se um computador compatível com IBM, processador 80386SX à 16MHz, e coprocessador 80387SX.



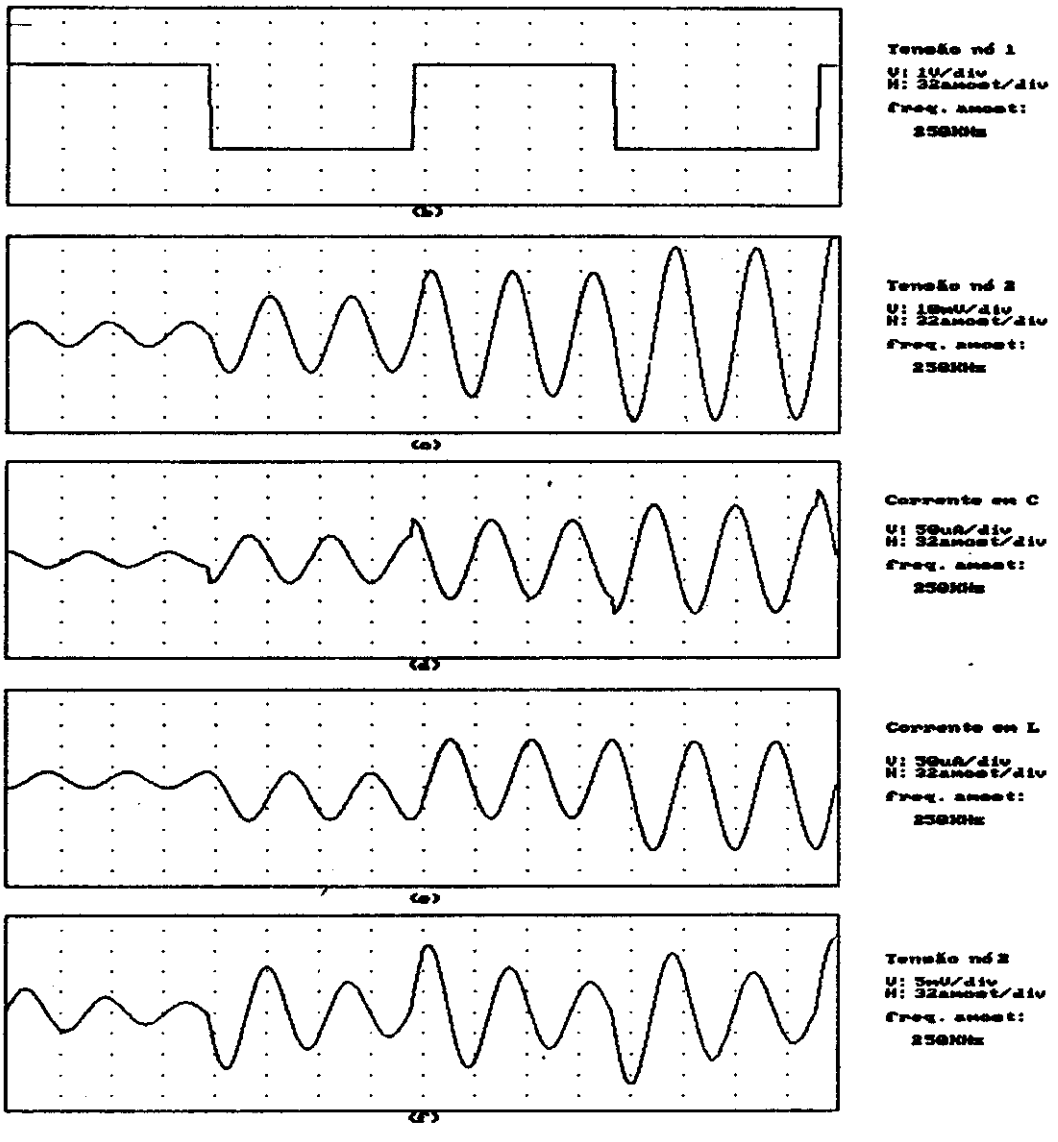
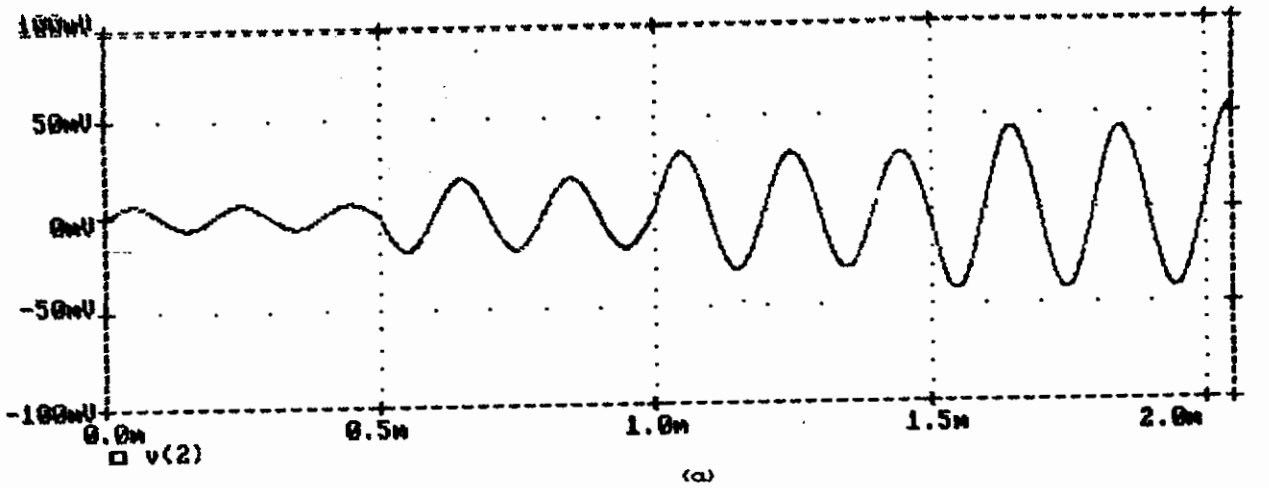
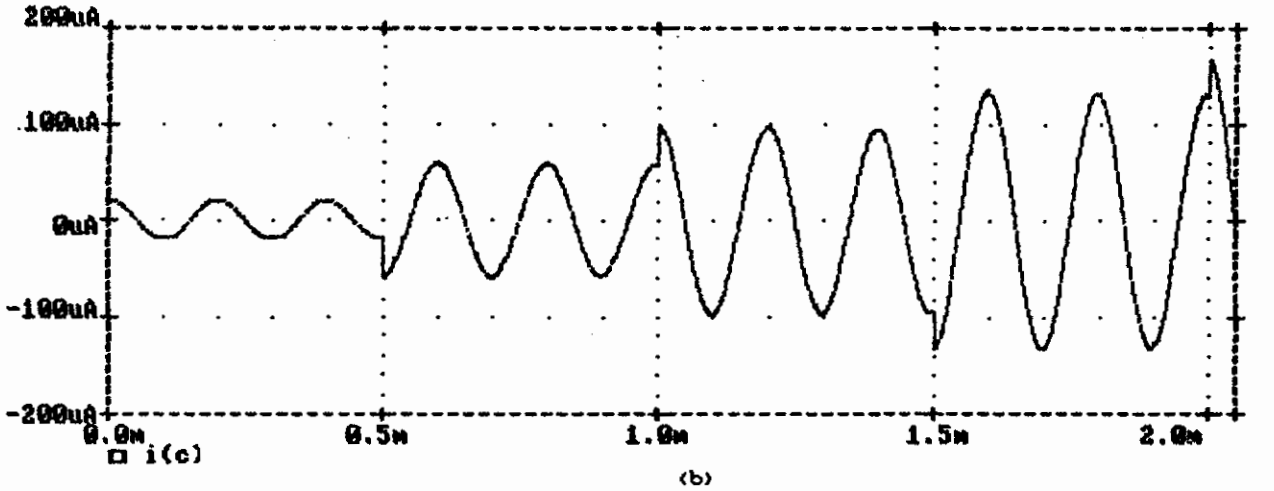


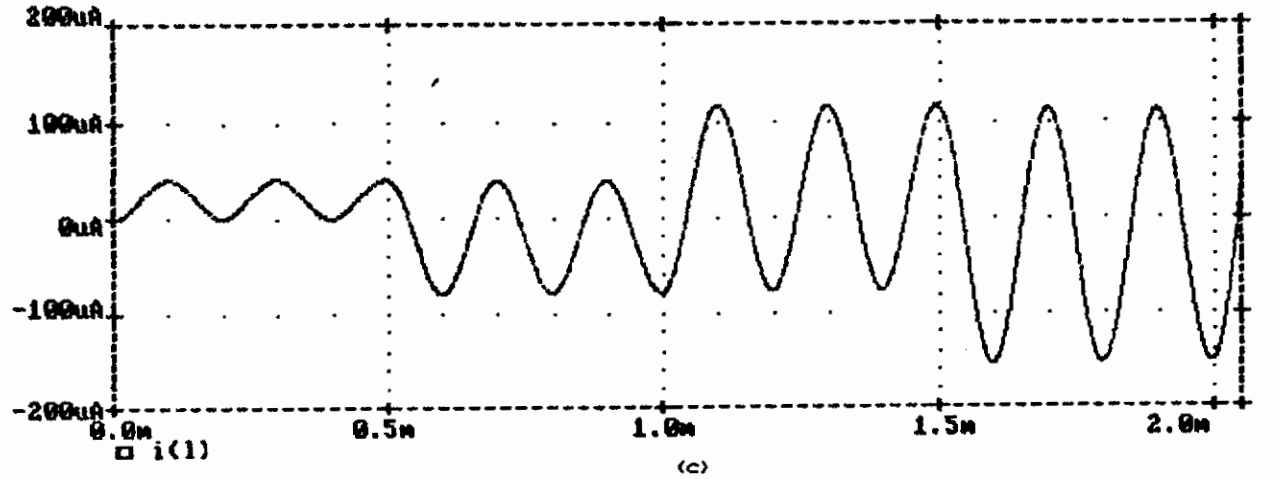
Fig. 6.10 – Simulação do circuito "URLC" – Exemplo 2
a) Circuito URLC
b) Sinal de entrada, onda quadrada
c) Tensão no nó 2
d) Corrente no Capacitor C
e) Corrente no Indutor L
f) Tensão no nó 2, pelo algoritmo de Euler Regressivo



(a)



(b)



(c)

Fig. 6.11- Simulação do circuito "VRLC", obtida pelo PSPICE
Resposta à onda quadrada de 1KHz
a) Tensão no nó 2
b) Corrente em C
c) Corrente em L

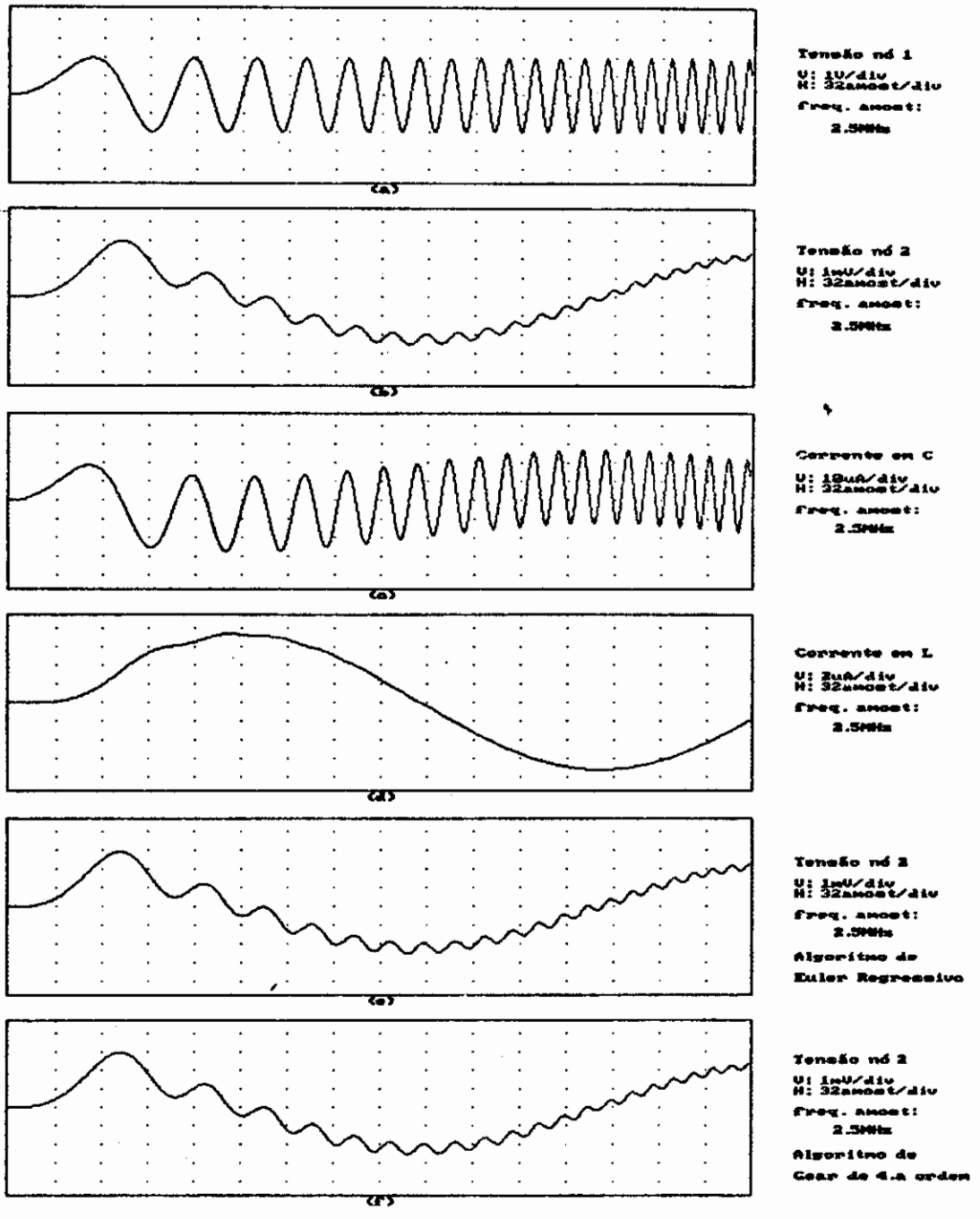


Fig. 6.12 - Simulação do circuito "VRLC" - Exemplo 2
a) Sinal de entrada, varredura de freqüências
b) Tensão no nó 2
c) Corrente no Capacitor C
d) Corrente no Indutor L
e) Tensão no nó 2, pelo algoritmo de Euler Regressivo
f) Tensão no nó 2, pelo algoritmo de Gear de 4a. ordem

A fig. 6.11 apresenta a simulação do circuito do segundo exemplo obtida pelo PSPICE. O sinal de entrada é uma onda quadrada de 1KHz. A simulação é feita de 0 a 2.048ms. Observa-se que as figuras 6.11-*a*,*b* e *c* correspondem aos resultados obtidos pelo SITECI nas figuras 6.10-*c*,*d* e *e*.

A fig. 6.10-f apresenta a tensão no nó 2 obtida pelo SITECI utilizando-se o algoritmo de Euler Regressivo. Nota-se uma grande diferença deste resultado para o obtido pelo PSPICE (fig. 6.11-a) e pelo SITECI utilizando o algoritmo de Gear de segunda ordem (fig. 6.10-b). Esta diferença é devida ao erro de truncamento do algoritmo de Euler, que na taxa de amostragem de 250KHz é bastante significativo.

O circuito "VRLC" também foi simulado pelo SITECI utilizando-se os algoritmos Trapezoidal, Gear de terceira ordem e Gear de quarta ordem. Os resultados foram semelhantes aos obtidos pelo PSPICE e pelo SITECI com algoritmo de Gear de segunda ordem. Desta forma as formas de onda correspondentes foram omitidas.

Os resultados das figuras 6.10 e 6.11 apresentam a simulação de 0 até 2.048ms. A oscilação crescente do nó 2 corresponde à resposta transitória, da condição de repouso até o regime permanente. A simulação para dezenas de ciclos do sinal de entrada mostrou que, em regime permanente, a oscilação tem amplitude constante. Este resultado foi verificado tanto com o PSPICE, como com o SITECI.

A fig. 6.12 apresenta a simulação do segundo exemplo para um sinal de entrada varredura senoidal (freq. inicial = 10Hz, freq. final = 100KHz), amostrada a 2.5MHz (fig.6.12-a). As figuras 6.12-*b*, *c* e *d* foram obtidas com o algoritmo de Gear de segunda ordem. As figuras 6.12-*e* e *f* foram obtidas com os algoritmos de Euler Regressivo e Gear de quarta ordem,

respectivamente. A comparação de 6.12-b,e e f mostra que para este sinal de entrada a ordem do algoritmo não é crítica.

A fig. 6.12-c mostra que, para as freqüências envolvidas, a corrente do capacitor praticamente acompanha as variações do sinal de entrada. Por outro lado, a corrente do indutor (fig. 6.12-d) não responde a estas freqüências, variando lentamente durante a simulação.

A simulação com o SITECI para a varredura senoidal é bastante simples, uma vez que este sinal esteja disponível em arquivo. Por sua vez, a simulação para este sinal de entrada pelo PSPICE não é simples, uma vez que não está disponível entre seus geradores de sinais.

O sinal varredura senoidal é interessante para testes no domínio do tempo, pois fornece uma indicação do comportamento em freqüência. Esta característica será explorada mais adiante em outros exemplos de simulação.

6.3 - Exemplo 3 - Linha de Transmissão

No terceiro exemplo é apresentada a simulação de uma linha de transmissão de 300Km (fig. 6.14-a). A linha foi modelada por 10 elementos "RLC", com os seguintes parâmetros:

R = 0.025 Ω/Km	⟶	R1=R2=... R10 = 0.75 Ω
ωL = 0.327 Ω/Km	⟶	L1=L2=... L10 = 0.026 H
ωC = 5.022 μsiemens/Km	⟶	C1=C2=... C10 = 0.360 pF
RL = 10 KΩ		

Fig. 6.13 - Valores dos parâmetros da linha de transmissão

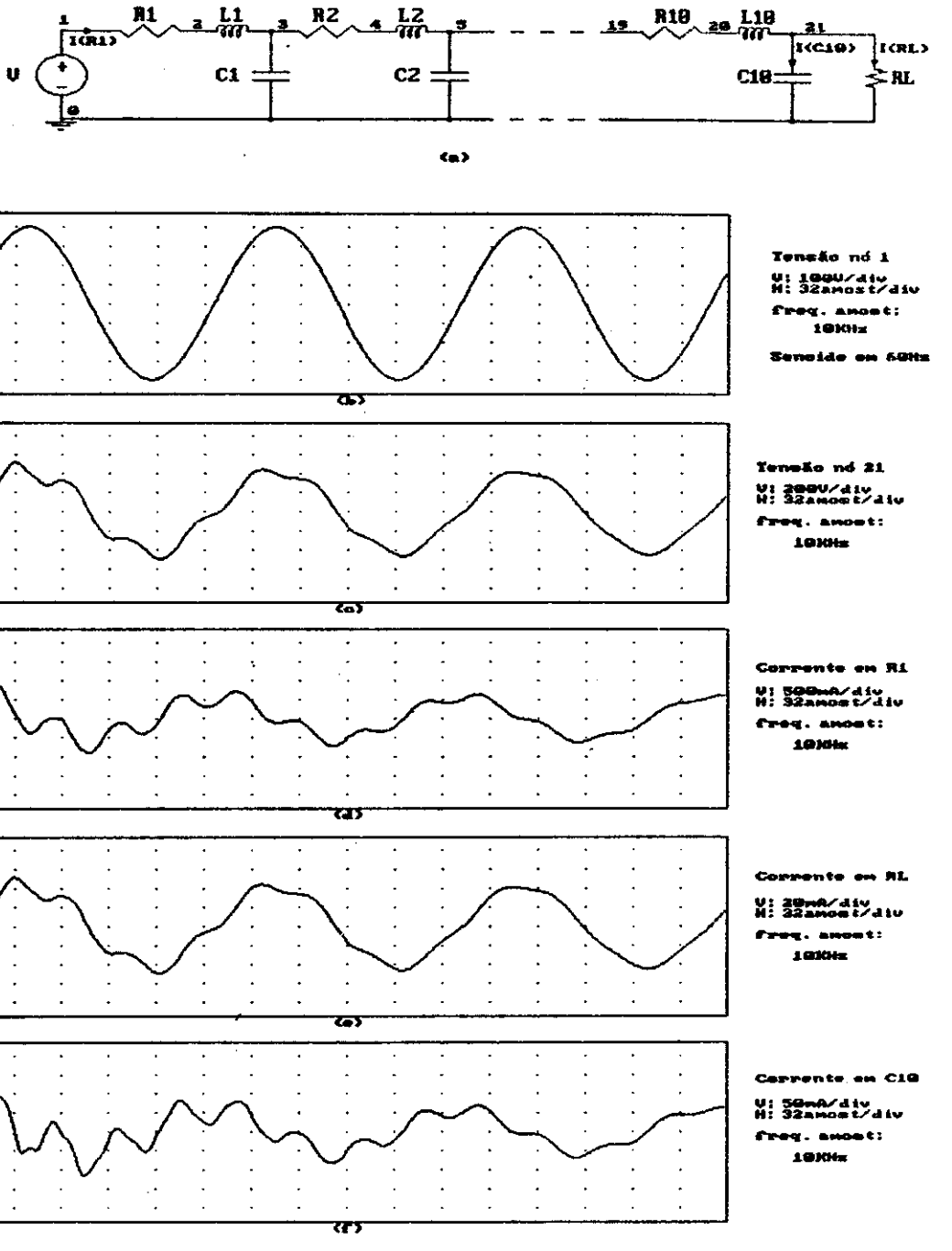
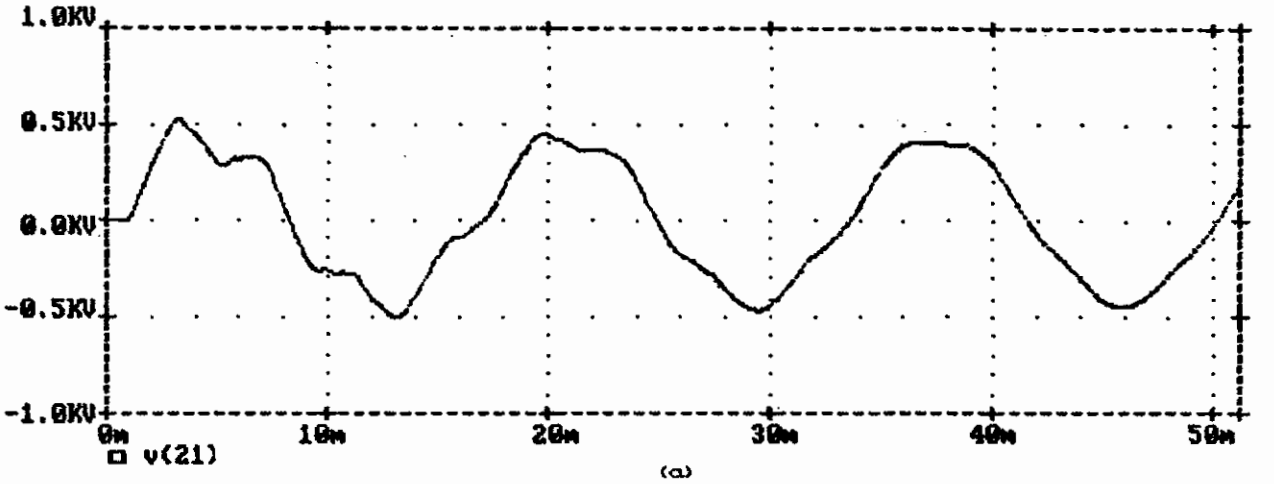
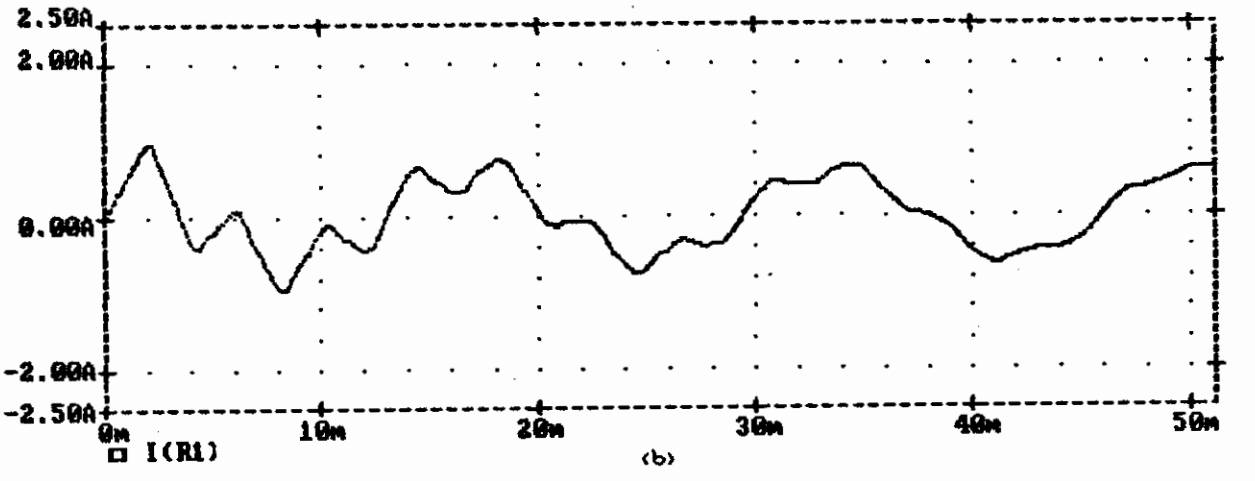


Fig. 6.14 – Simulação do circuito "TRANSMIS" – Exemplo 3
a) Circuito "TRANSMIS" – Linha de Transmissão de 300KM
b) Sinal de entrada, senóide em 60Hz, 800Vpp
c) Tensão no nó 21
d) Corrente em R1
e) Corrente em RL
f) Corrente em C10





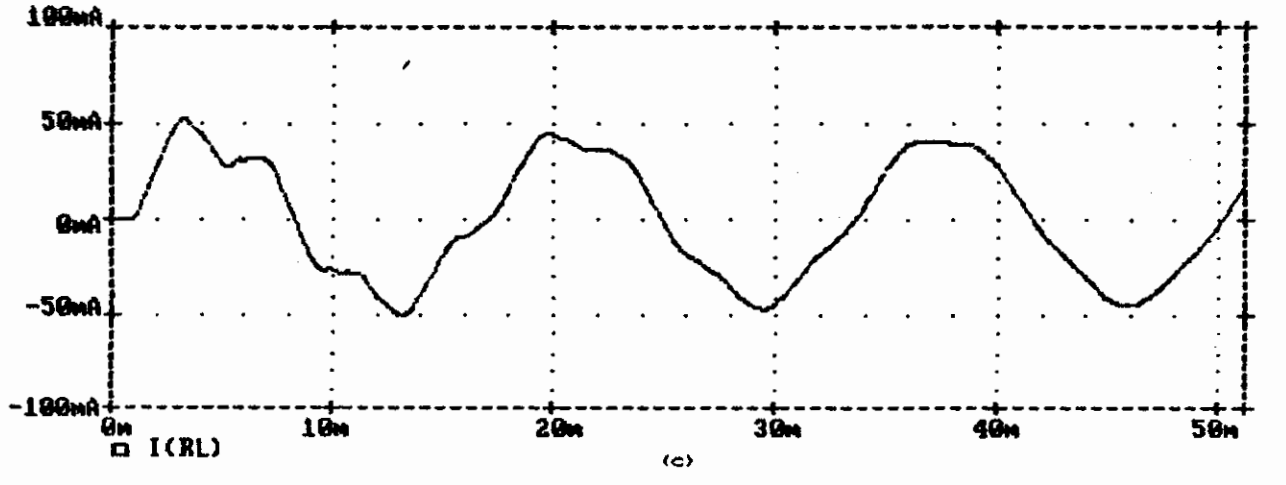


Fig. 6.15- Simulação do circuito "TRANSMIS", obtida pelo PSPICE
Entrada: senoide 6KHz, 800Vpp. Duração 0s - 51.2ms
a) Tensão no nó 21
b) Corrente em R1
c) Corrente em RL

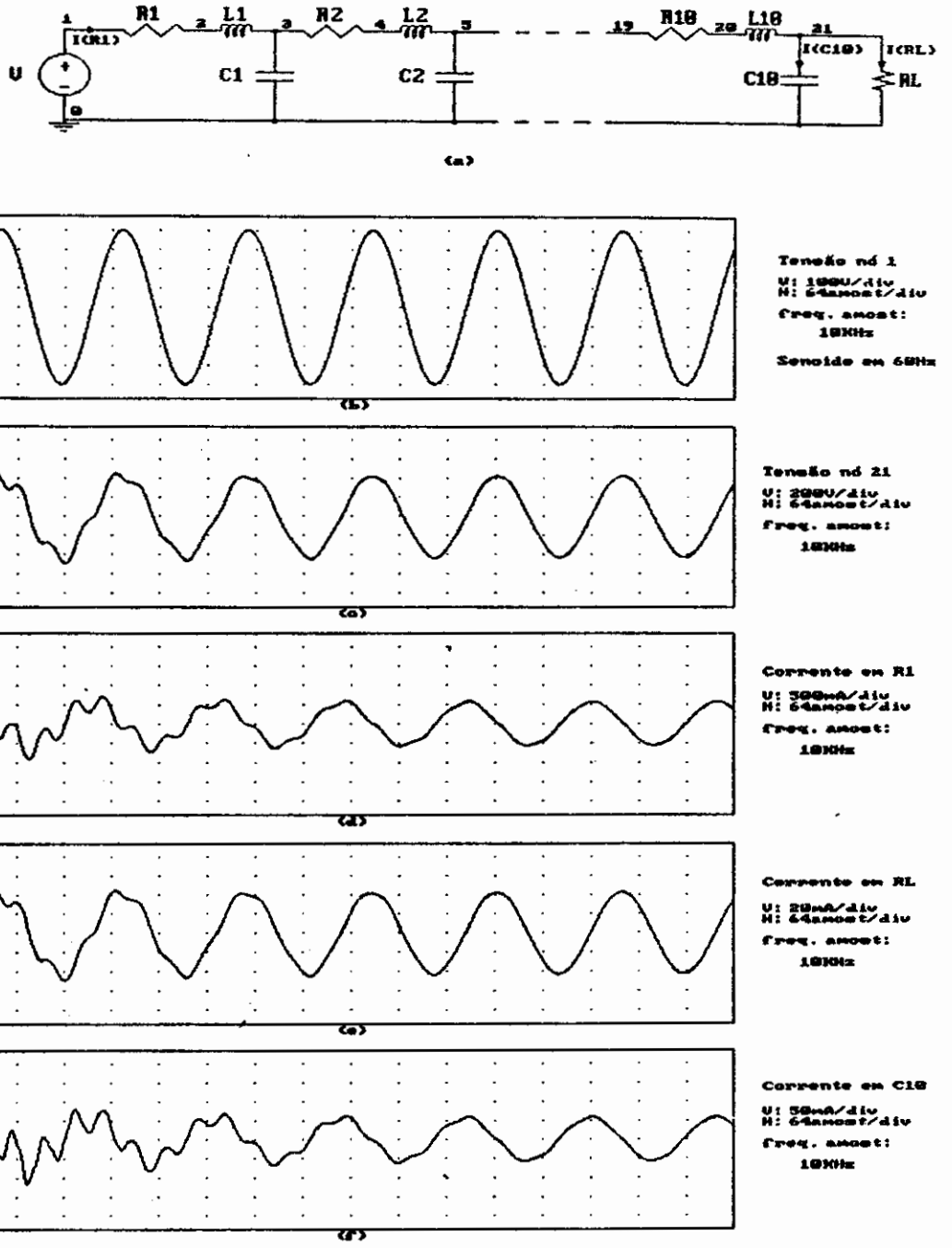


Fig. 6.16 - Simulação do circuito "TRANSMIS" - Exemplo 3
a) Circuito "TRANSMIS" - Linha de Transmissão de 300KM
b) Sinal de entrada, senóide em 60Hz, 800Vpp
c) Tensao no nó 21
d) Corrente em R1
e) Corrente em RL
f) Corrente em C10

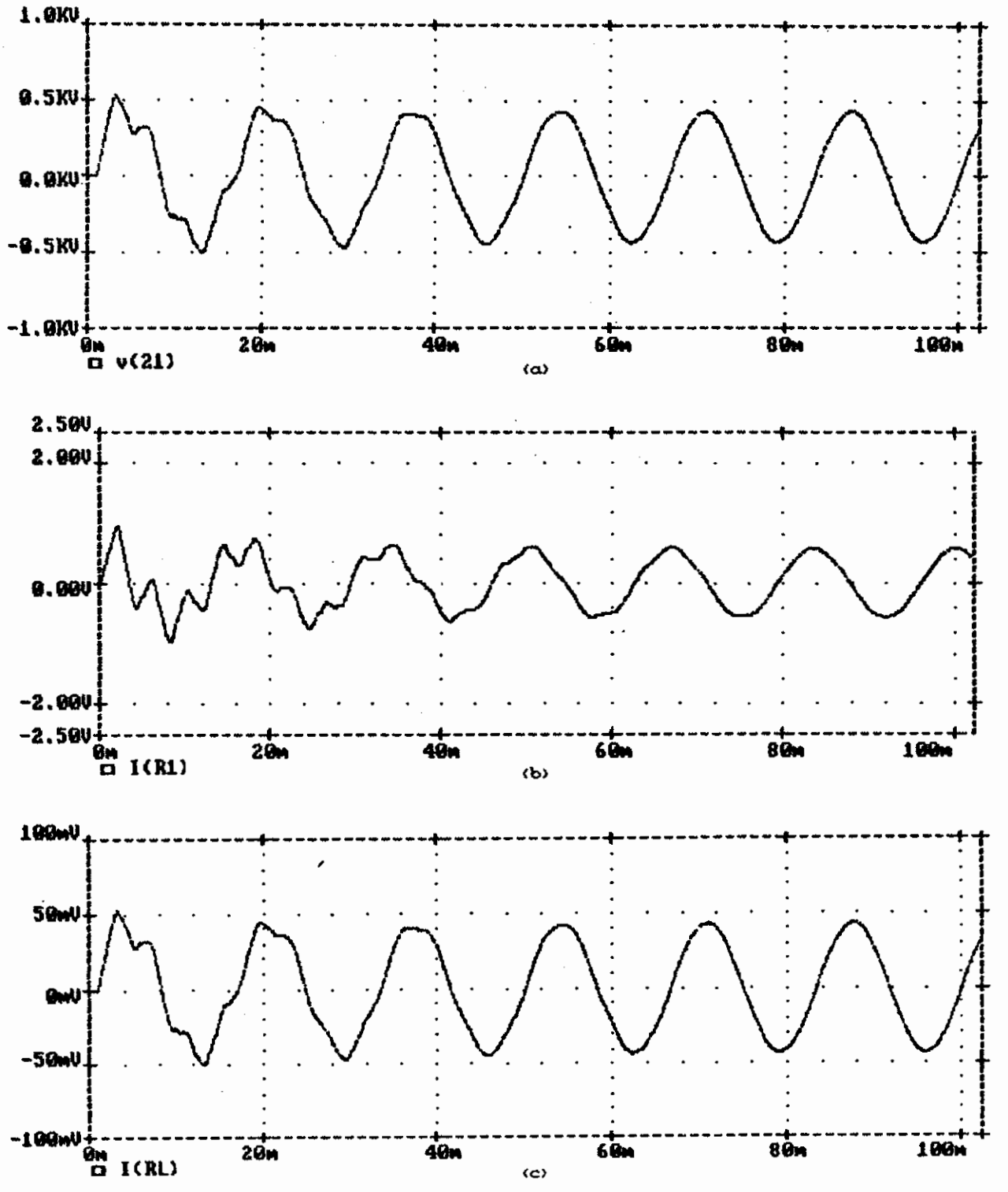


Fig. 6.17- Simulação do circuito "TRANSMIS", obtida pelo PSPICE
Entrada: senoide 60Hz, 800Vpp. Duração 0s - 102.4ms
a) Tensão no nó 21
b) Corrente em R1
c) Corrente em RL

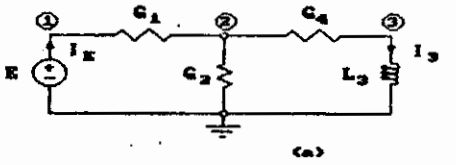
A simulação da linha de transmissão do terceiro exemplo aparece nas figuras 6.14 e 6.16. Resultados equivalentes obtidos pelo PSPICE podem ser vistos nas figuras 6.15 e 6.17. O sinal de entrada, conectado ao nó 1, consiste de uma senóide de amplitude 400V em 60Hz, conforme visto em 6.14-a e 6.16-a. As figuras 6.14 e 6.15 abrangem de 0 a 51.2ms, focalizando o transitório da linha. Já as figuras 6.16 e 6.17, cobrem de 0 a 102.4ms, permitindo visualizar a estabilização da linha em regime permanente.

A simulação começa com todos os capacitores descarregados e todos os indutores sem corrente. A partir do instante em que a alimentação de 60Hz é ligada ocorre um transitório na linha de transmissão que dura aproximadamente quatro ciclos. Pode ser visto nas figuras 6.14-c e 6.15-a que a tensão no nó 21, a ponta da linha, só aparece após um atraso de aproximadamente 1ms. A tensão no nó 21 atinge mais de 500V em $\simeq$ 3ms.

As simulações das figuras 6.14 e 6.16 utilizaram o algoritmo de Gear de segunda ordem. As simulações com algoritmos de ordem mais elevada forneceram resultados semelhantes.

6.4 - Exemplo 4 - Circuito VR3L

O quarto exemplo de simulação aparece na figura 6.18-a. Este circuito foi apresentado no capítulo 2 (fig. 2.23) ilustrando a formulação nodal modificada. No capítulo 3 (fig. 3.21), o mesmo circuito exemplifica a discretização do sistema de equações pelo algoritmo de Euler Regressivo. Quando este circuito é excitado por um trem de pulsos a 100KHz (fig. 6.18-b) obtém-se, pelo algoritmos de Euler Regressivo, os sinais das figuras 6.18-c a *f*.



G1 = 1e-3 ℧
G2 = 1e-3 ℧
G4 = 1e-3 ℧
L = 1e-3 H

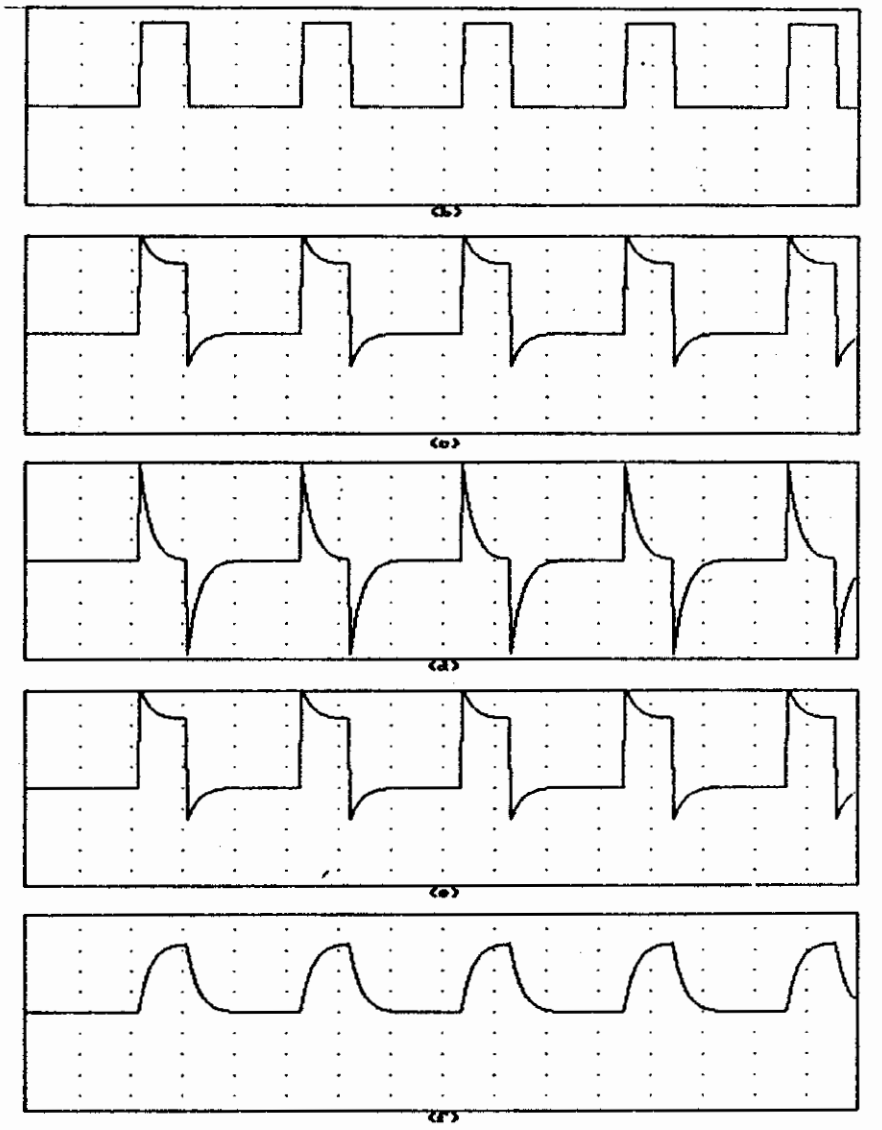


Fig. 6.18 – Simulação do circuito "VR3L" – Exemplo 4
a) Circuito "VR3L"
b) Sinal de entrada, trem de pulsos a 100KHz
c) Tensão no nó 2
d) Tensão no nó 3
e) Corrente em G2
f) Corrente em L

6.5 - Exemplo 5 - Circuito JC2RL

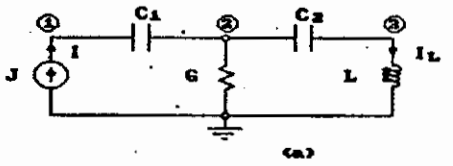
O circuito da fig. 6.19-a é o quinto exemplo simulado. Este circuito foi apresentado no capítulo 3 (fig. 3.23) exemplificando a discretização do sistema de equações nodais modificadas pelos algoritmos de Euler Regressivo, Trapezoidal e Gear de segunda ordem (equações 3.23, 3.24 e 3.25). Na presente seção este circuito é simulado por uma fonte de corrente que gera uma varredura senoidal (fig. 6.19-b). A freqüência inicial é 0.1Hz e a freqüência final é 1KHz. A taxa de amostragem é de 25KHz.

Da fig. 6.19-c, que mostra a tensão no nó 2, pode ser visto o comportamento "rejeita-faixa" deste circuito. Por sua vez, a tensão no nó 3 (fig. 6.19-d) tem amplitude crescente com o aumento da freqüência. Esta característica era esperada, em virtude do aumento da impedância do indutor com o aumento da freqüência. As simulações foram obtidas com o algoritmo de Gear de segunda ordem.

6.6 - Exemplo 6 - Filtro Rejeita-Faixa de Segunda Ordem

O exemplo 6 consiste do circuito rejeita-faixa de segunda ordem da figura 6.20-a. Dois sinais de entrada são utilizados, a varredura senoidal da fig. 6.20-b e a onda quadrada da fig. 6.20-e.

O sinal da fig. 6.20-c, correspondente à tensão no nó 5, caracteriza bem o circuito rejeita-faixa. A freqüência central deste filtro está em 796Hz. Na figura 6.21-a pode ser vista a corrente em R3 para a entrada 6.20-e, obtida pelo SITECI. As figuras 6.21-b e 6.21-c apresentam a resposta à onda quadrada obtidas pelo PSPICE.



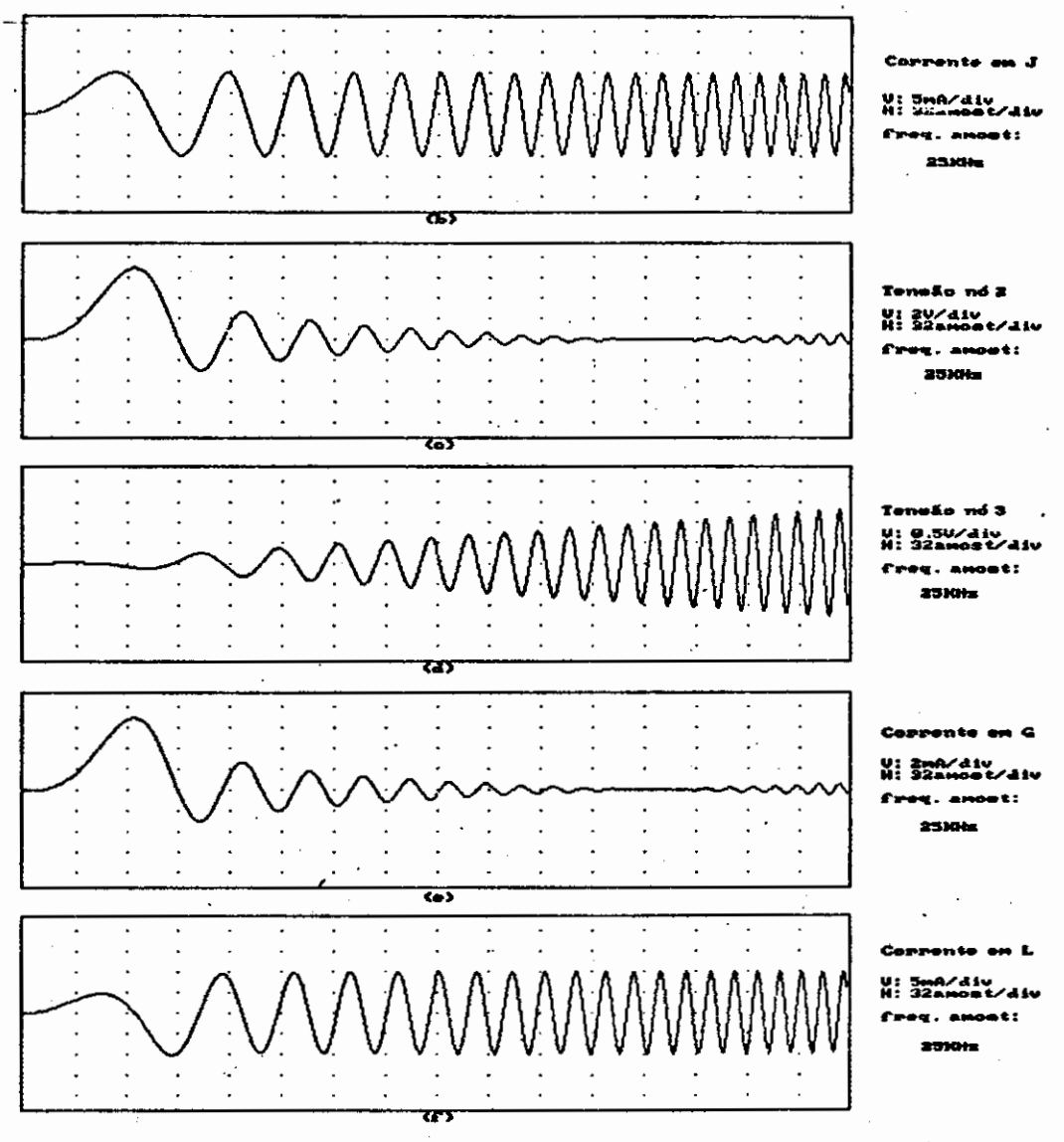
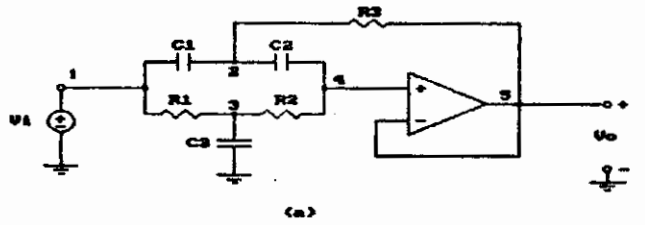
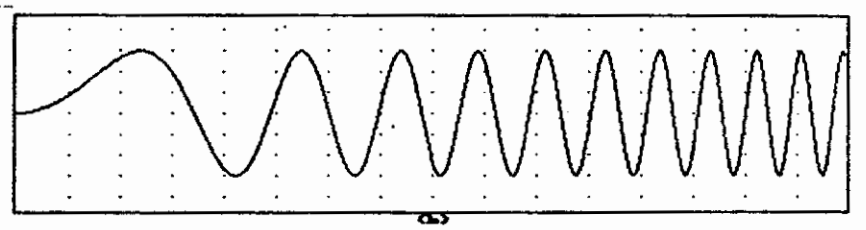


Fig. 6.19 – Simulação do circuito "JC2RL" – Exemplo 5
a) Circuito "JC2RL"
b) Sinal de entrada, varredura senoidal
c) Tensão no nó 2
d) Tensão no nó 3
e) Corrente em G
f) Corrente em L



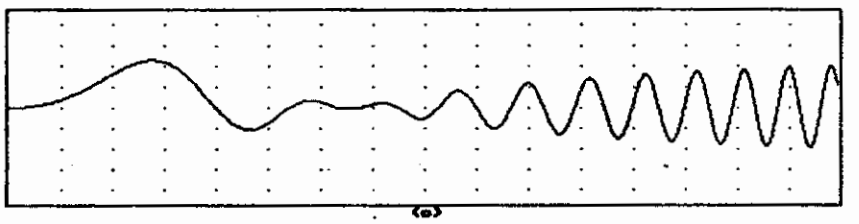
(a)

C1 = C2 = 200nF
C3 = 400nF
R1 = R2 = 1Kohm
R3 = 500ohm



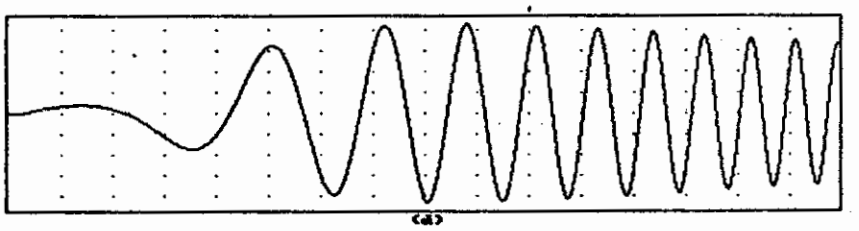
(b)

Tensão nó 1
V: 1V/div
H: 32amost/div
freq. amost:
500KHz



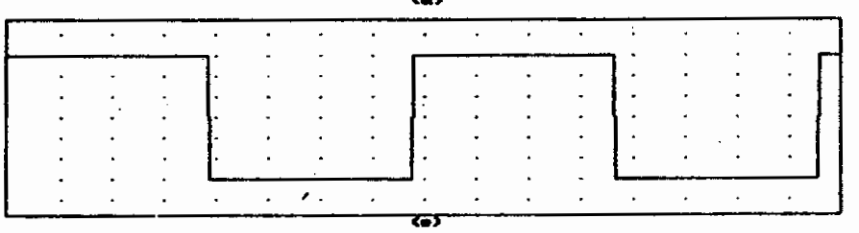
(c)

Tensão nó 5
V: 1V/div
H: 32amost/div
freq. amost:
500KHz



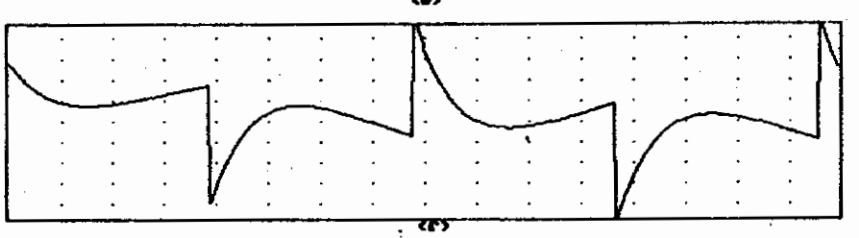
(d)

Corrente em R3
V: 0.5mA/div
H: 32amost/div
freq. amost:
500KHz



(e)

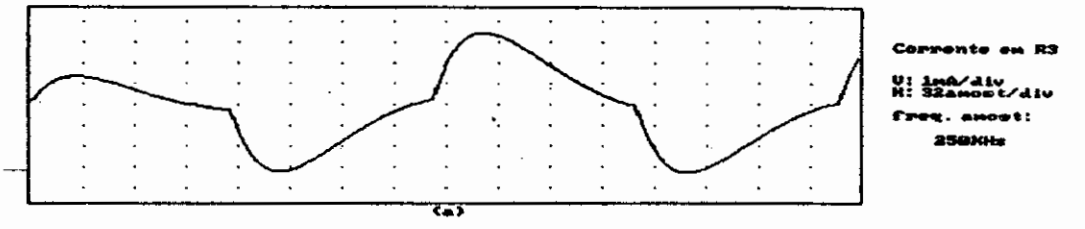
Tensão nó 1
V: 1V/div
H: 32amost/div
freq. amost:
250KHz

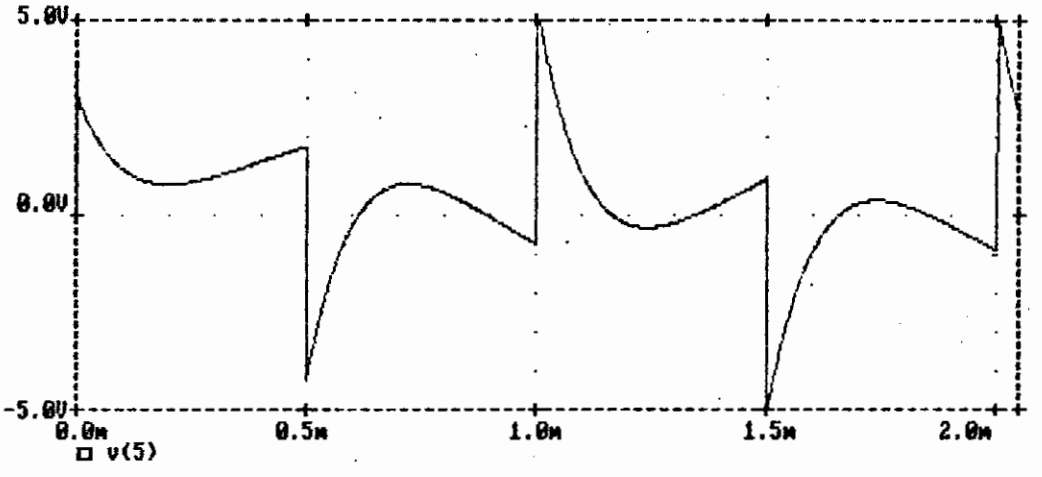


(f)

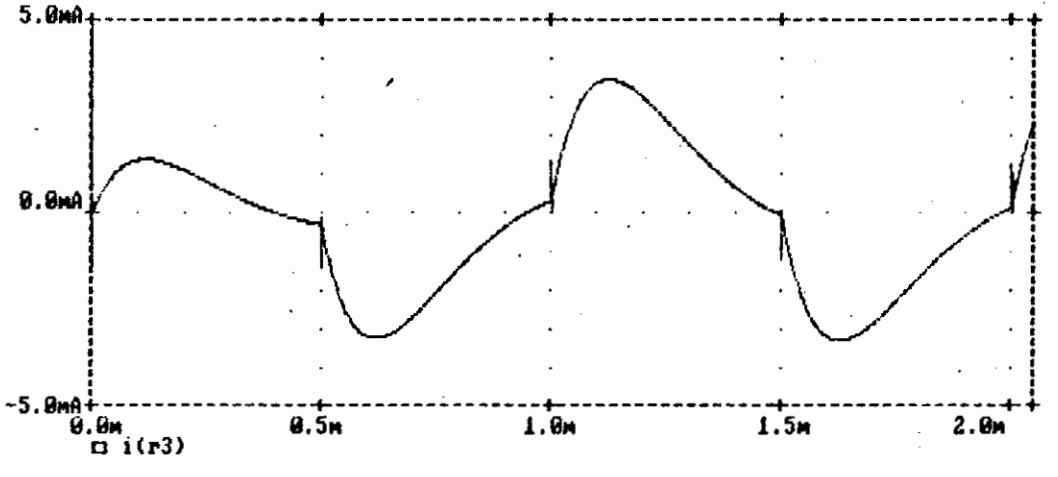
Tensão nó 5
V: 1V/div
H: 32amost/div
freq. amost:
250KHz

Fig. 6.28 – Simulação do circuito "REJEITAF" – Exemplo 6
a) Circuito "REJEITAF"
b) Sinal de entrada, varredura senoidal
c) Tensão no nó 5
d) Corrente em R3
e) Sinal de entrada, onda quadrada
f) Tensão no nó 5



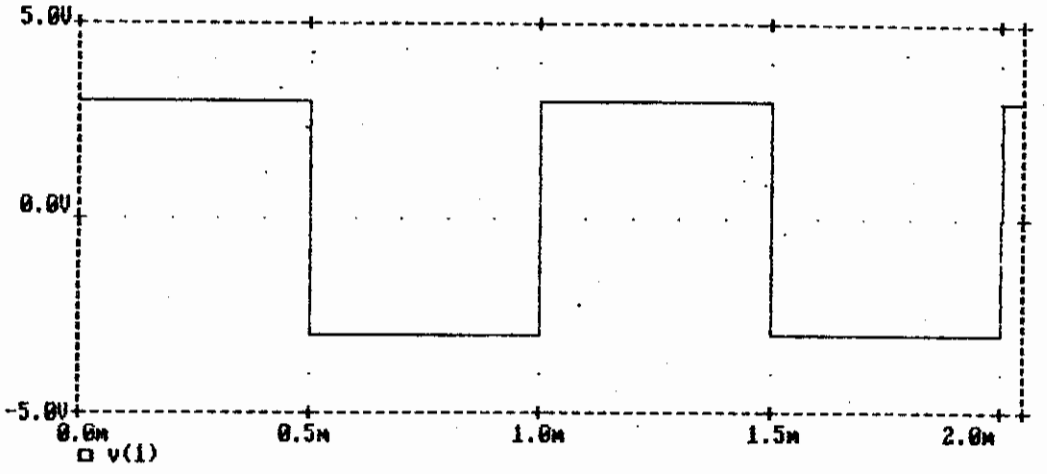


(b)

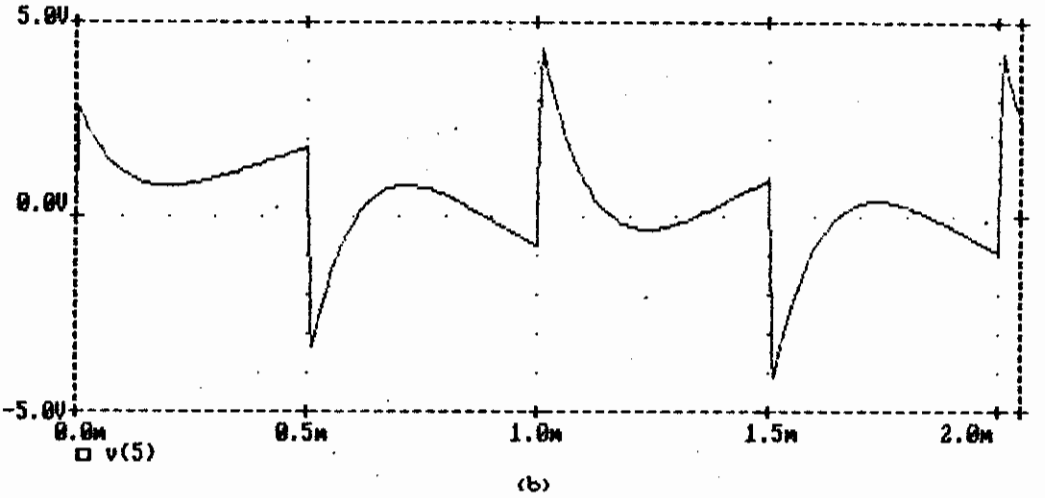


(c)

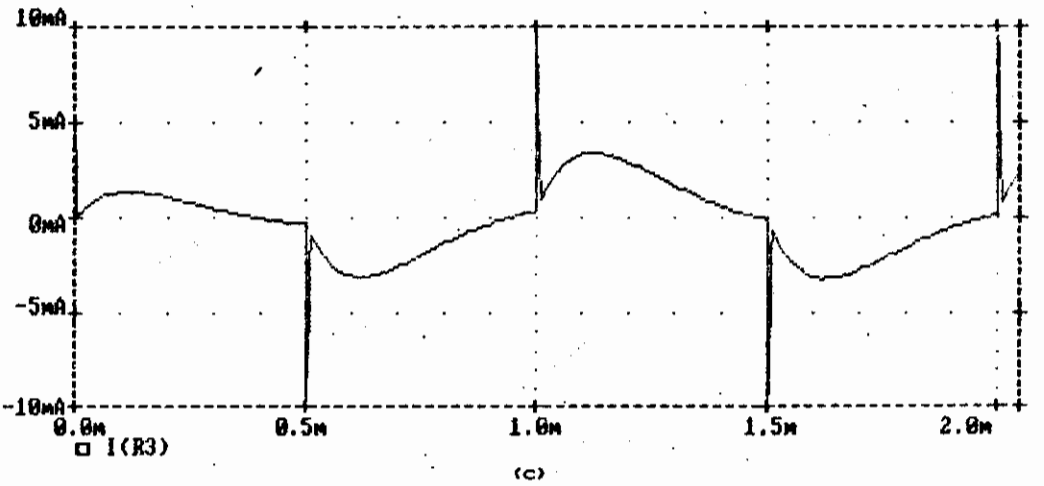
Fig. 6.21- Simulação do circuito "REJEITAF"
Entrada: Onda quadrada em 1KHz
a) Corrente em R3 obtida pelo SITECI
b) Tensão no nó 5 obtida pelo PSPICE
c) Corrente em R3 obtida pelo PSPICE



(a)



(b)



(c)

Fig. 6.22- Simulação do circuito "REJEITAF", obtida pelo PSPICE
Amplificador Operacional: Modelo completo do UA741
a) Sinal de entrada, tensão no nó 1
b) Tensão no nó 5
c) Corrente em R3

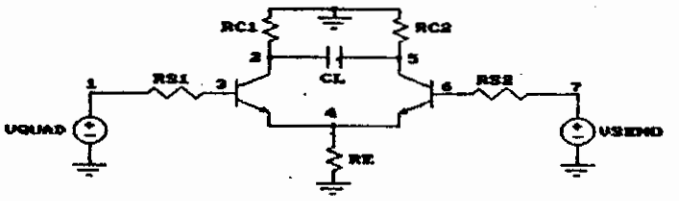
As figuras 6.21-b e c foram obtidas pelo PSPICE utilizando-se um modelo linearizado do amplificador operacional UA741. Este modelo linearizado é mesmo utilizado pela biblioteca do SITECI, que pode ser visto na fig. 5.15 (pag. 142). Nota-se que a saída do filtro obtida pelo PSPICE (nó 5, fig.6.21-b) coincide com a saída obtida pelo SITECI (fig. 6.20-f).

A fig. 6.22 apresenta os resultados da simulação do circuito do sexto exemplo obtidos pelo PSPICE utilizando-se o modelo completo do UA741. Neste caso o modelo do operacional inclui o modelo completo dos dois transistores do amplificador diferencial de entrada. Também estão incluídos dois diodos do estágio de saída. Estes quatro componentes não lineares permitem explicar a diferença entre o sinal de corrente no resistor R3 (figuras 6.21-a, 6.21-c e 6.22-c). Para o modelo completo a transição do sinal de entrada provoca uma grande variação na corrente em R3. Para o modelo linearizado esta variação é bastante atenuada.

6.7 - Exemplo 7 - Amplificador Diferencial

O sétimo exemplo consiste de um amplificador diferencial formado por dois transitores (fig. 6.23). A simulação considera um modelo linear em torno do ponto de operação. Duas fontes de sinal excitam o circuito, uma fonte senoidal em 4KHz (fig. 6.24-a)e uma fonte de onda quadrada em 1KHz (6.24-b), ambas amostradas em 250KHz e amplitude 2mVpp.

As formas de onda obtidas pela simulação com o SITECI podem ser vistas nas figuras 6.24-c a 6.24-e. O modelo adotado para os transistores foi o pi-híbrido completo, conforme descrito na seção 5.3. A simulação do mesmo circuito obtida pelo PSPICE, pode ser visto na figura 6.25.



RS1 = RS2 = 100 ohm
RC1 = RC2 = 1 Kohm
RE = 100 ohm
CL = 1 nF

Fig. 6.23 - Amplificador Diferencial do Exemplo 7

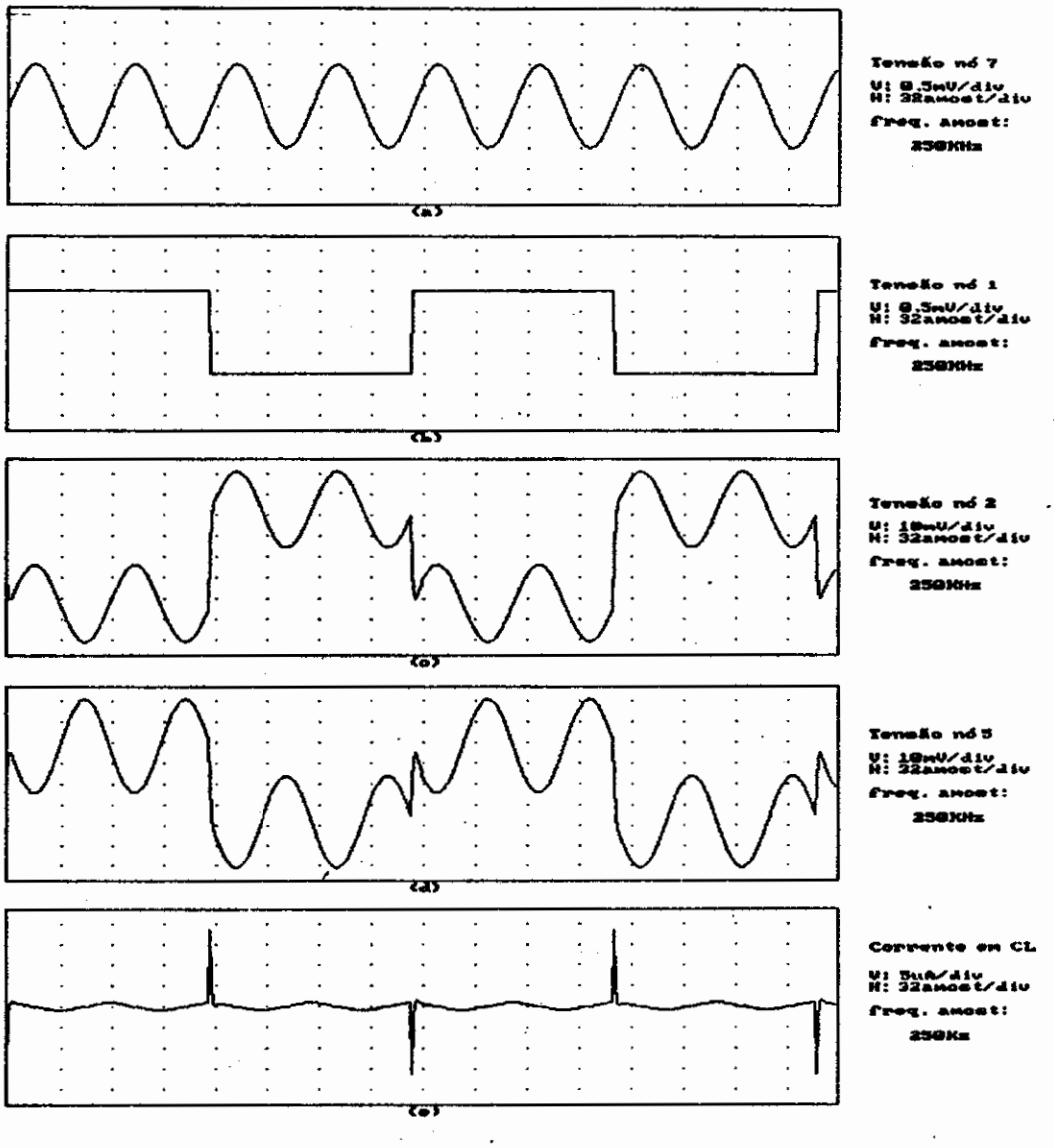


Fig. 6.24 - Simulação do circuito Amplificador Diferencial, exemplo 7, obtida pelo SITECI
a) Sinal de entrada, nó 7, senóide em 4KHz
b) Sinal de entrada, nó 1, quadrada em 1KHz
c) Tensão no nó 2
d) Tensão no nó 5 } Pelo Alg. de Gear de segunda ordem
e) Corrente em CL

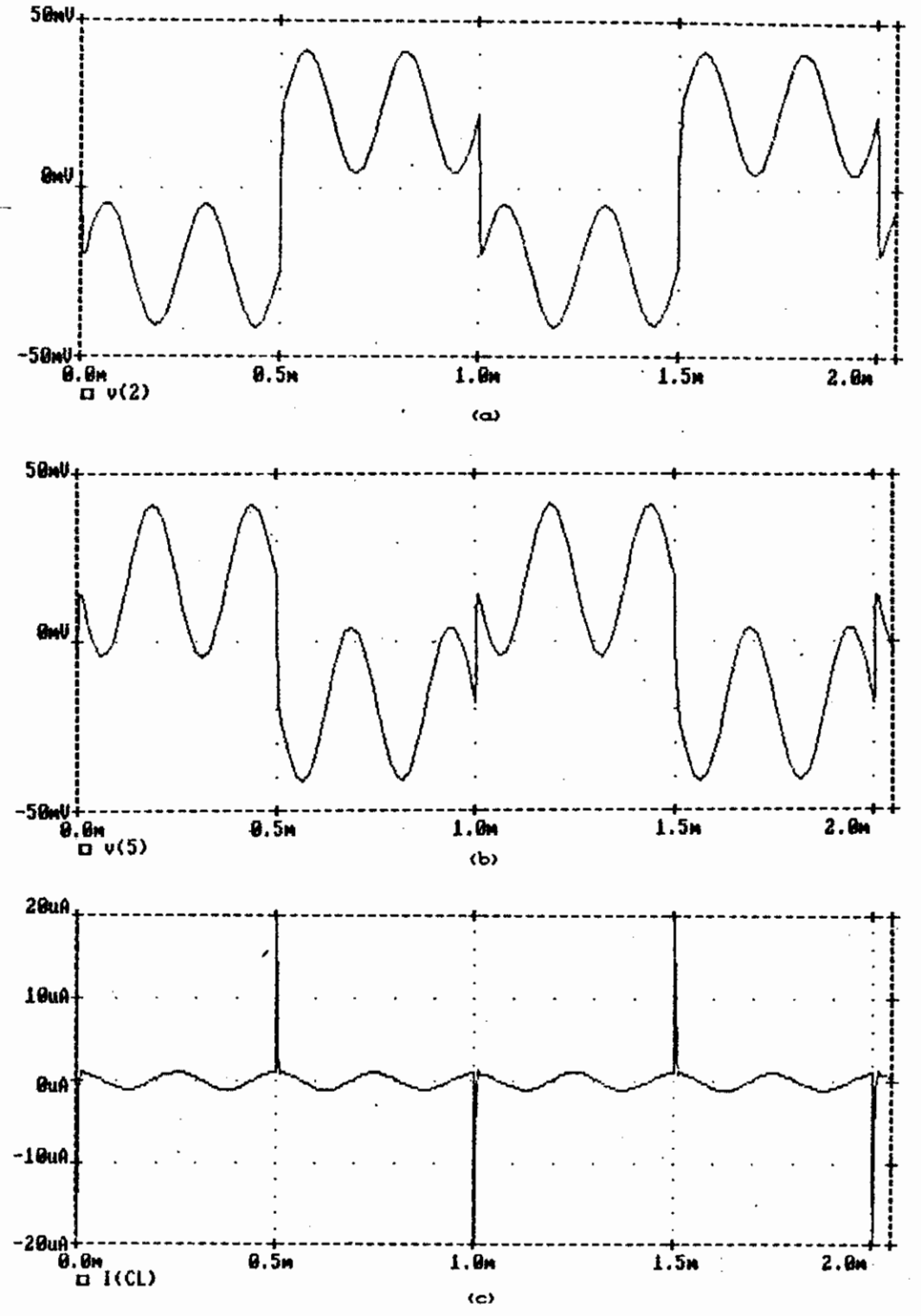


Fig. 6.25 - Simulação Exemplo 7, obtida pelo PSPICE
a) Tensão no nó 2
b) Tensão no nó 5
c) Corrente em CL

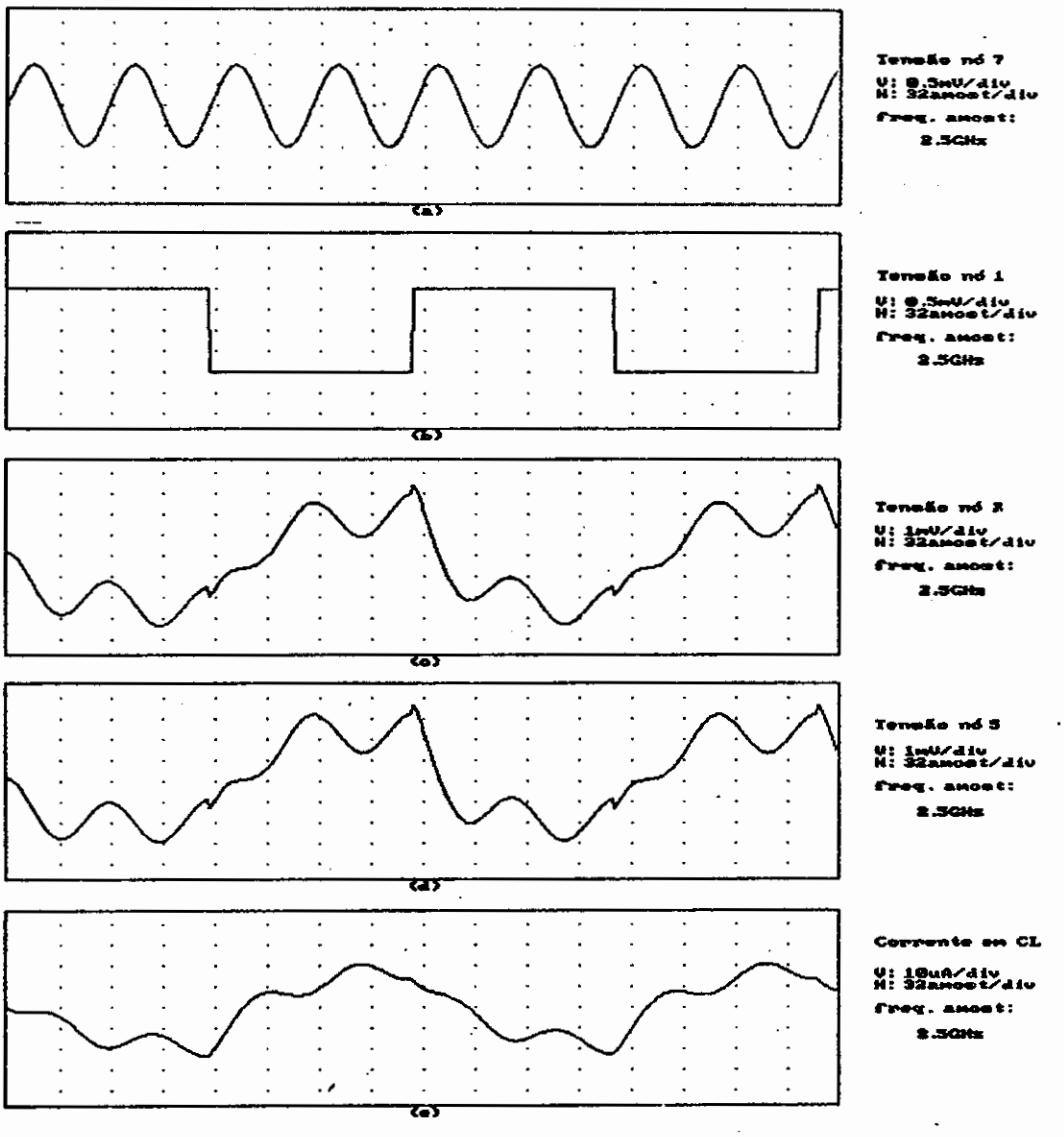


Fig. 6.26 – Simulação do circuito Amplificador Diferencial, exemplo 7, obtida pelo SITECI
a) Sinal de entrada, nó 7, senóide em 40 MHz
b) Sinal de entrada, nó 1, quadrada em 10 MHz
c) Tensao no nó 2 ┐
d) Tensao no nó 5 ├ Pelo Alg. de Gear de segunda ordem
e) Corrente em CL ┘

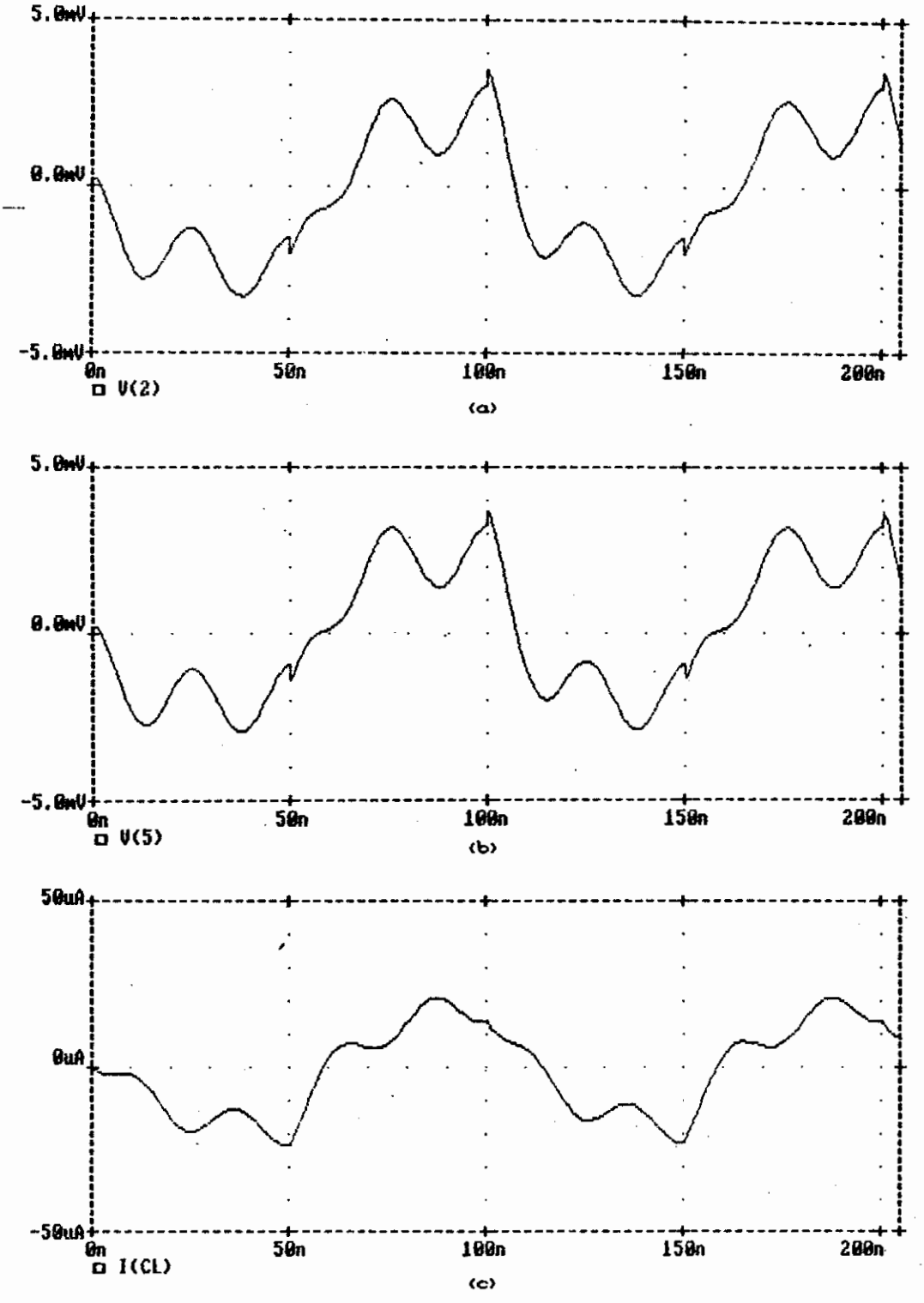


Fig. 6.27 - Simulação Exemplo 7, obtida pelo PSPICE
Sinal de entrada nó 7: Senóide em 40MHz
Sinal de entrada nó 1: Onda quadrada em10MHz
a) Tensão no nó 2
b) Tensão no nó 5
c) Corrente em CL

As figuras 6.26 e 6.27 consideram a simulação do sétimo exemplo quando os sinais de entrada são uma onda quadrada em 10MHz e uma senóide em 40MHz, ambas de amplitude 2mVpp amostradas a 2.5GHz. Com estas simulações pode ser visto o efeito das capacitâncias do modelo pi-híbrido em altas freqüências.

Os parâmetros do modelo pi-híbrido dos transistores para estas simulações foram os seguintes:

rbb' = 100Ω
rb'e = 1000Ω
rb'c = 4MΩ
rce = 81.6KΩ
gm = 50mA/V
Cc = 3pF
Ce = 100pF

A conexão destes componentes na configuração pi-híbrida completa pode ser vista na fig. 5.14.

As figuras 6.24 e 6.25 mostram que em baixas freqüências (1KHz e 4KHz dos sinais de entrada) as tensões nos nós 2 e 5 são praticamente simétricas. Por outro lado, para altas freqüências dos sinais de entrada, as figuras 6.26 e 6.27 mostram que o capacitor CL é praticamente um curto circuito. A tensão no nó 2 é muito próxima da tensão no nó 5.

6.8 - Exemplo 8 - Amplificador com Realimentação Série de Tensão

Neste exemplo considera-se um amplificador em dois estágios a transistor com realimentação série de tensão. A realimentação é feita do coletor do segundo transistor para o emissor do primeiro transistor, conforme visto na figura 6.28.

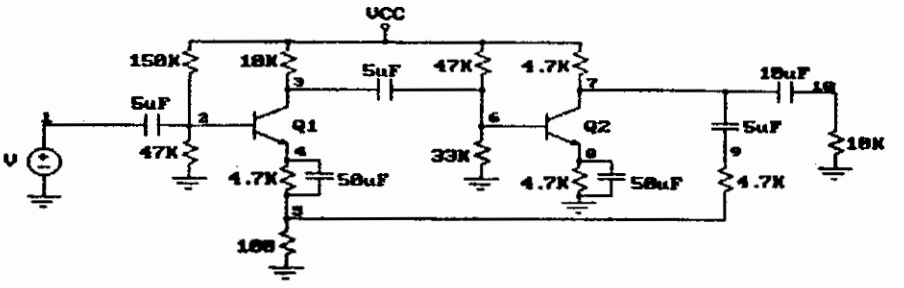


Fig. 6.28 – Amplificador com Realimentação Série de Tensão – Exemplo 8

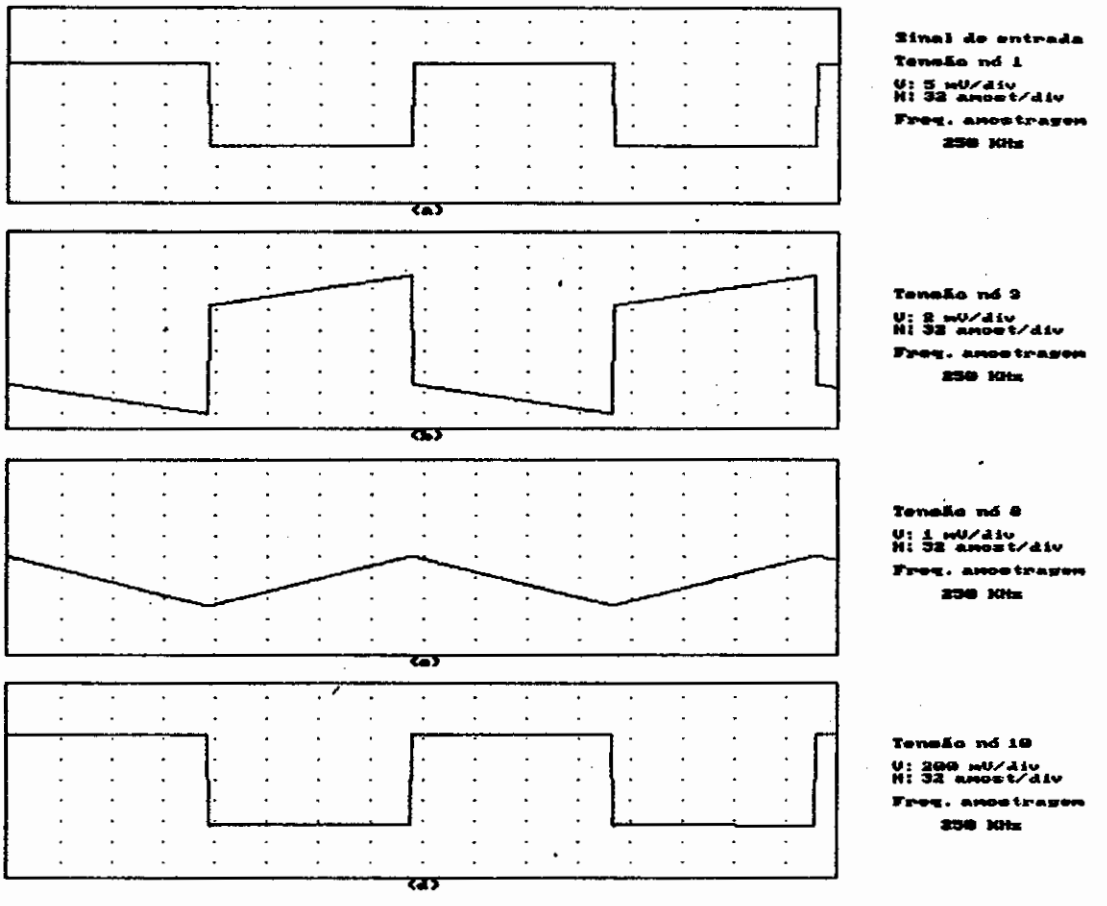


Fig. 6.29 – Simulação do Exemplo 8, obtida com o SITECI
a) Sinal de entrada, onda quadrada em 1KHz
b) Tensão no nó 3
c) Tensão no nó 8
d) Tensão no nó 10

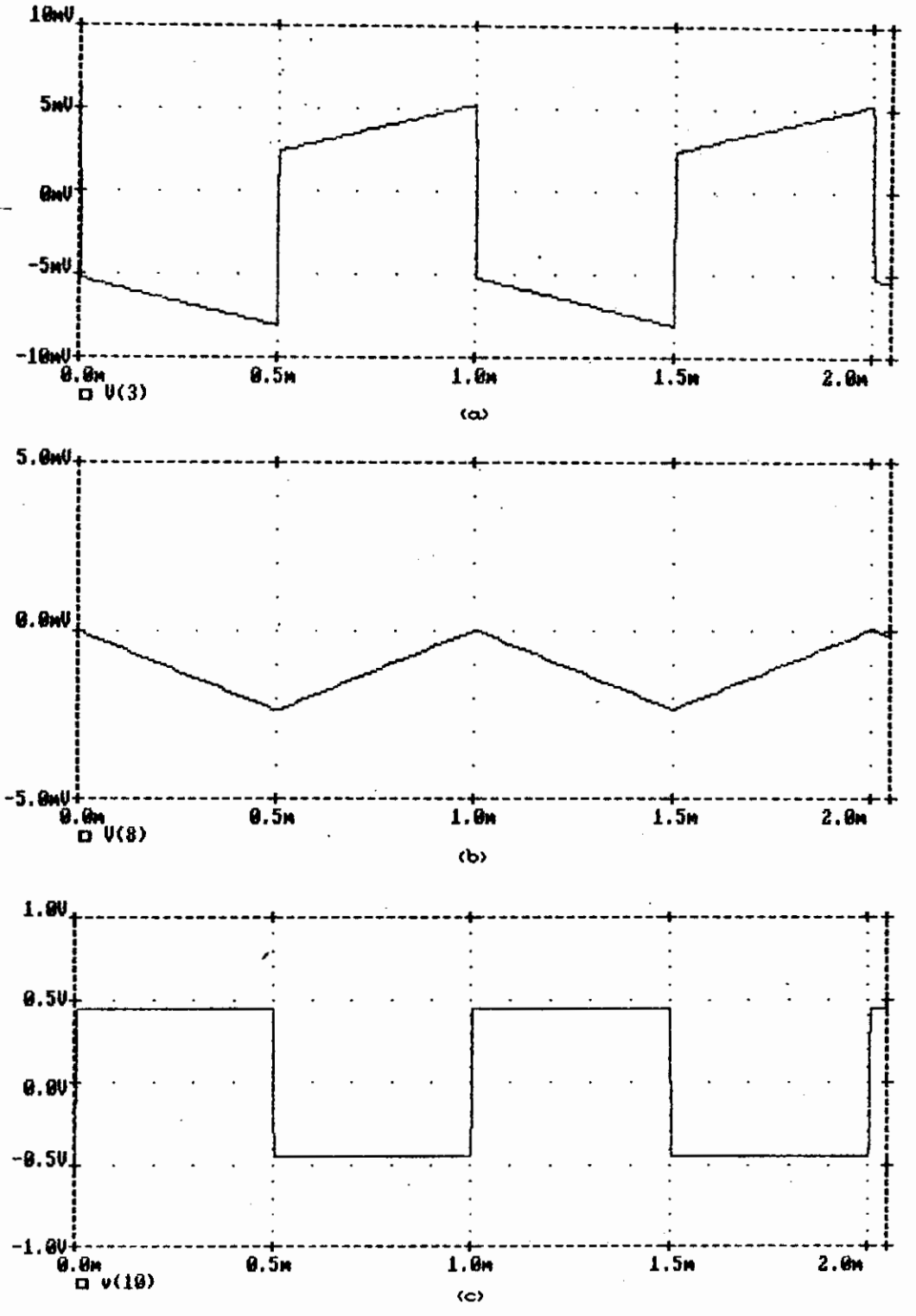


Fig. 6.30 - Simulação do Exemplo 8, obtida pelo PSPICE
Sinal de entrada nó 1: Onda quadrada em 1KHz
a) Tensão no nó 3
b) Tensão no nó 8
c) Tensão no nó 10, saída do amplificador.

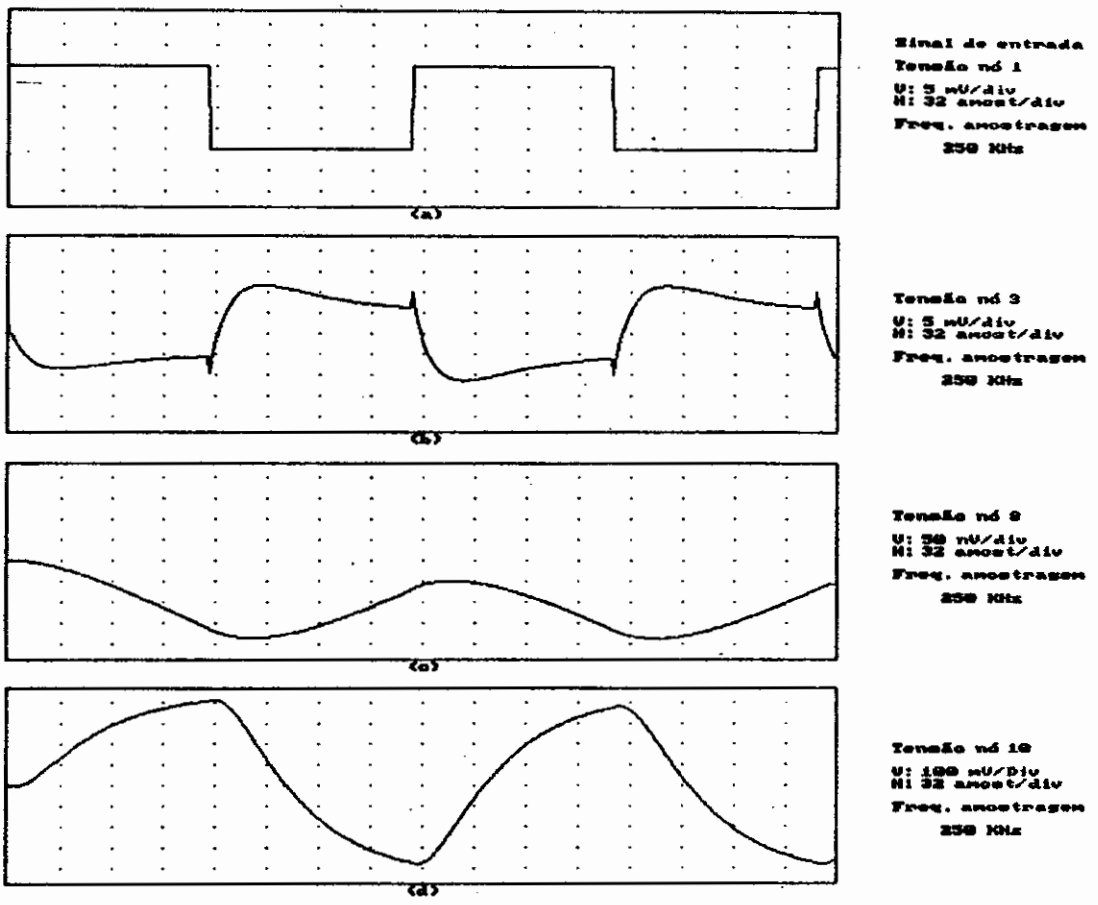


Fig. 6.31 – Simulação do Exemplo 8, obtida com o SITECI
a) Sinal de entrada, onda quadrada em 10 MHz
b) Tensão no nó 3
c) Tensão no nó 8
d) Tensão no nó 10

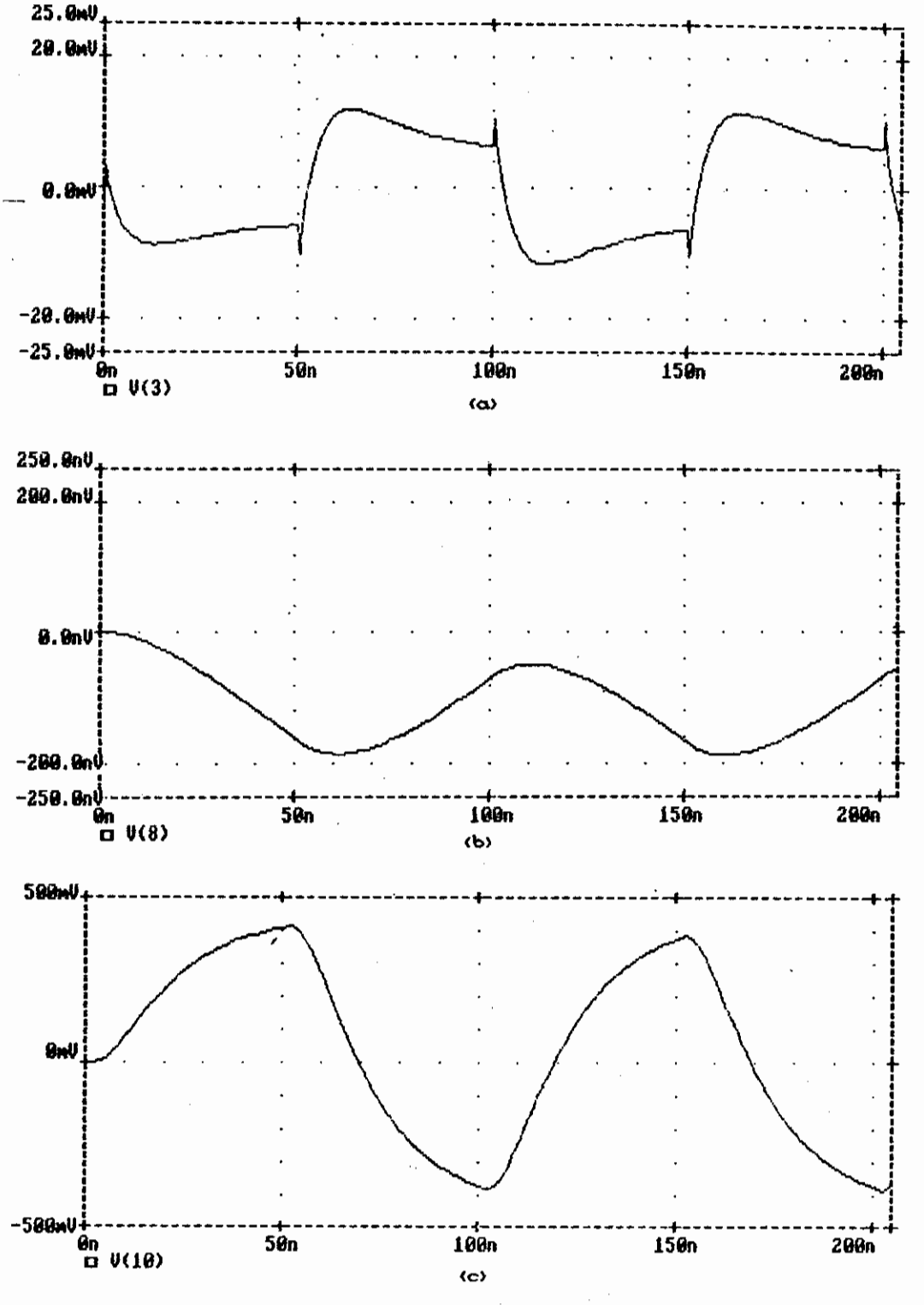


Fig. 6.32 - Simulação do Exemplo 8, obtida pelo PSPICE
Sinal de entrada nó 1: Onda quadrada em 10MHz
a) Tensão no nó 9
b) Tensão no nó 8
c) Tensão no nó 10, saída do amplificador.

A fig. 6.29 apresenta os resultados da simulação do oitavo exemplo obtida pelo SITECI. A entrada conectada ao nó 1 é uma onda quadrada de 1KHz, amplitude 20mVpp amostrada em 250KHz. Os transistores foram representados pelo seu modelo pi-híbrido completo para pequenos sinais. Os parâmetros adotados pelo modelo são os mesmos da seção anterior:

rbb' = 100Ω
rb'e = 1000Ω
rb'c = 4MΩ
rce = 81.6KΩ
gm = 50mA/V
Cc = 3pF
Ce = 100pF

É interessante observar que neste exemplo a simulação considera o efeito dos capacitores de desacoplamento DC.

A simulação do oitavo exemplo obtida com o PSPICE pode ser vista na fig. 6.30. Tendo em vista a coerência da comparação, os transistores foram substituidos pelo mesmo modelo adotado no SITECI.

Nas figuras 6.31 e 6.32 são apresentados os resultados das simulações do oitavo exemplo para um sinal de entrada de onda quadrada de 10MHz, amplitude 20mVpp amostrado em 2.5GHz. Nota-se que para esta freqüência, as capacitâncias internas do modelo pi-híbrido produzem grande atenuação do sinal de saída.

6.9 - Exemplo 9 - Filtro Chebyshev de Sétima Ordem

Este exemplo considera a simulação de um filtro ativo passa-baixa Chebyshev de sétima ordem. Quatro amplificadores operacionais são utilizados na implementação, conforme pode ser visto na fig. 6.33

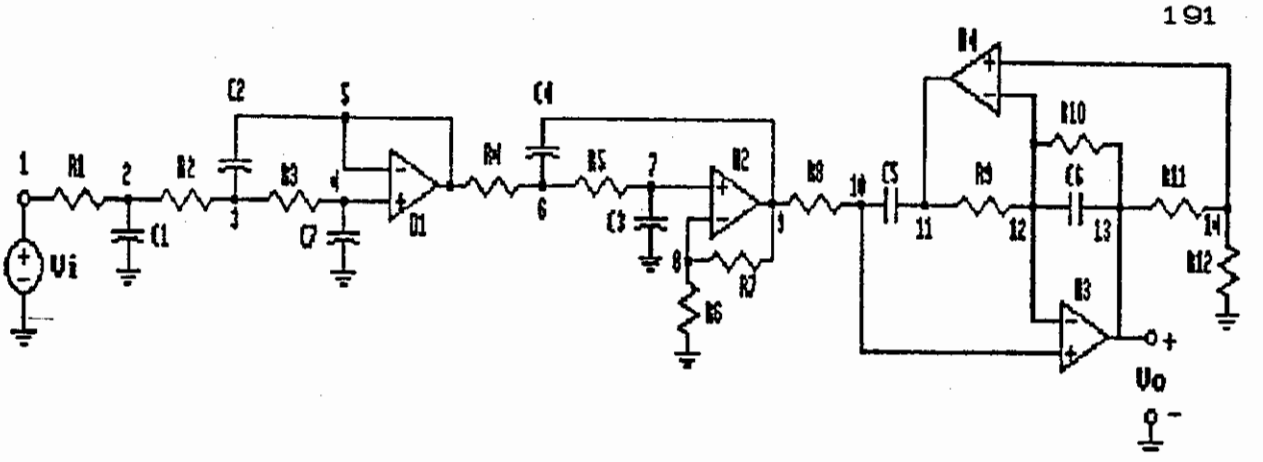
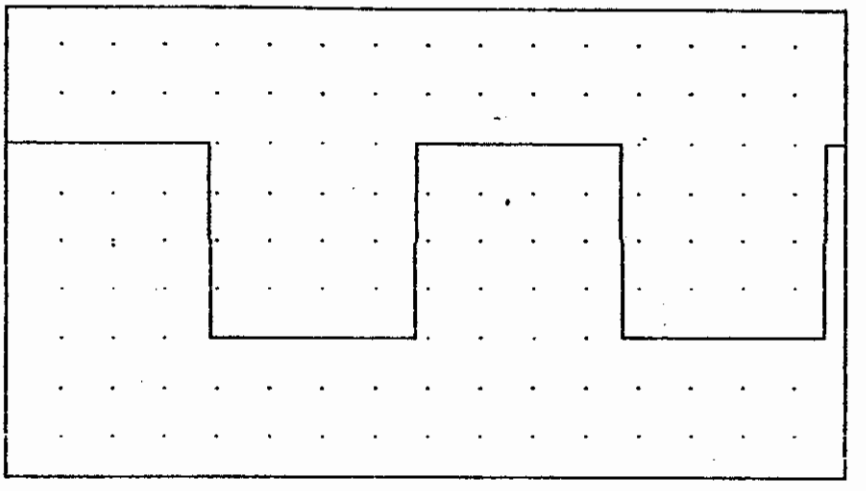
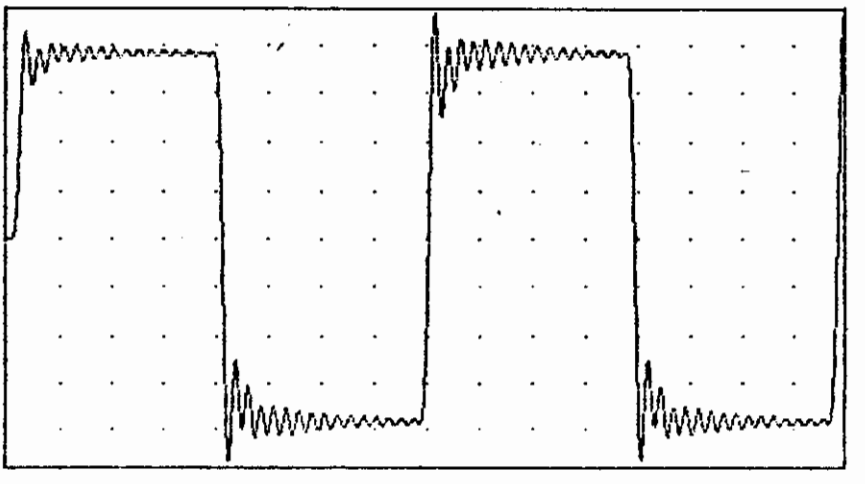


Fig. 6.33 - Filtro Passa-baixa Chebyshev de Sétima Ordem



Sinal de entrada
Tensão nó 1
V: 1 V/div
H: 64 amost/div
Freq. amostragem
500 KHz

(a)



Tensão nó 13
V: 1 V/div
H: 64 amost/div
Freq. amostragem
500 KHz

(b)

Fig. 6.34 - Simulação do Exemplo 9, obtida pelo SITECI
Algoritmo de Discretização: Gear de QuartaOrdem
a) Sinal de entrada nó 1: Onda quadrada em100 Hz
b) Sinal de saída, tensão no nó 13

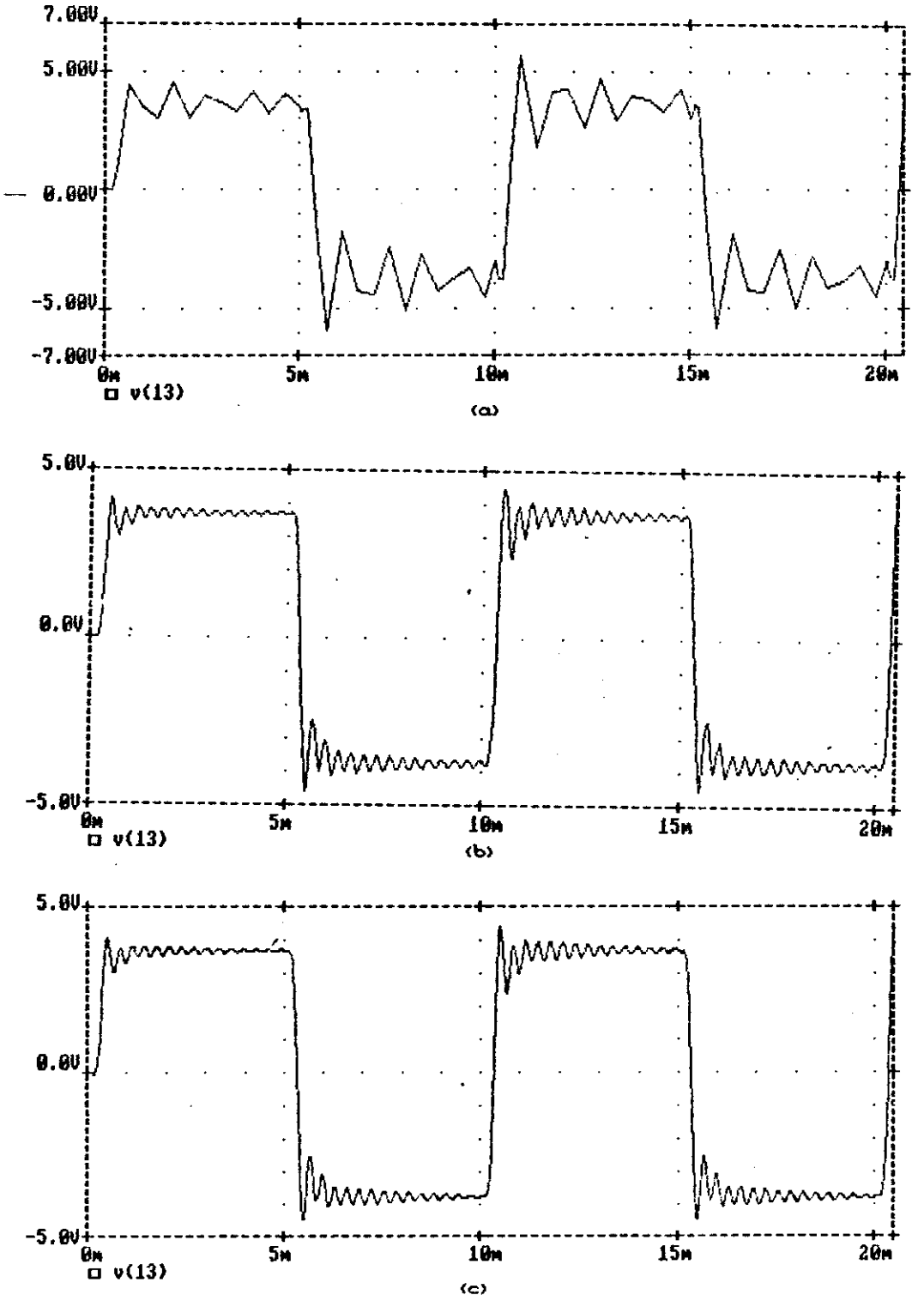


Fig. 6.35 - Simulação do Exemplo 9, obtida pelo PSPICE
Sinal de entrada nó 1: Onda quadrada de 100 Hz
a) Tensão nó 13, passo de iteração livre.
b) Tensão nó 13, passo máximo limitado em 0.04ms.
c) Tensão nó 13, passo máximo limitado em 0.016ms.

Os valores dos componentes do filtro passa-baixa Chebyshev da fig. 6.33 são os seguintes:

R1 = R2 = 57.6 KΩ
R3 = 26.1 KΩ
R4 = 23.7 KΩ
R5 = 24.3 KΩ
R6 = 191 KΩ
R7 = 1 KΩ
R8 = R9 = R11 = R12 = 33.2 KΩ
R10 = 453 KΩ
C1 = 4.42 nF
C2 = C4 = 15 nF
C3 = C7 = 442 pF
C5 = C6 = 1.5 nF

O simulador substitui cada amplificador operacional pelo modelo linear do UA741 descrito na seção 5.3, fig. 5.15. Com estas substituições a matriz nodal modificada que representa este circuito tem dimensão 51. Nota-se que a descrição original do circuito possui apenas 14 nós. Desta forma ilustra-se a versatilidade da representação do amplificador operacional em biblioteca.

O resultado da simulação deste circuito pelo SITECI, quando excitado por uma onda quadrada de 100Hz (fig. 6.34-a), pode ser visto na figura 6.34-b. Este resultado foi obtido com a discretização das equações feita pelo algoritmo de Gear de Quarta ordem.

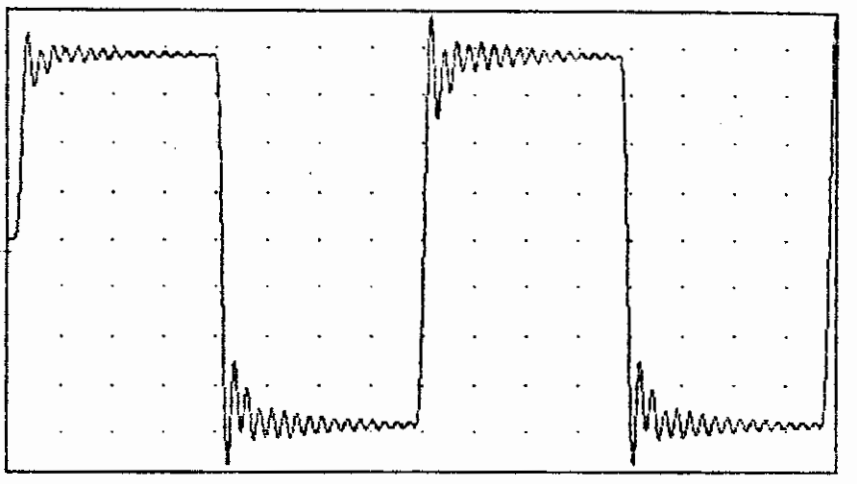
A figura 6.35 apresenta a simulação do nono exemplo obtida pelo PSPICE. Apesar do PSPICE possuir em biblioteca o modelo completo do amplificador operacional UA741, utilizou-se o mesmo modelo simplificado adotado pelo SITECI. Isto permite que seja feita a comparação tanto em termos de forma de onda de saída, como em termos de tempo de processamento. Na fig.6.35 podem ser vistos três resultados fornecidos pelo PSPICE. A fig. 6.35-a foi obtida deixando-se livre a adaptação do passo de iteração. Neste caso o passo

máximo utilizado pelo PSPICE foi 0.4096ms. Obviamente esta forma de onda apresenta um erro excessivo e não corresponde à realidade. A fig. 6.35-b foi obtida limitando-se o passo de iteração máximo em 0.04ms, enquanto na fig. 6.35-c este limite foi de 0.016ms. Nota-se que não existe um ganho significativo de precisão quando o passo máximo é limitado em menos de 0.04ms. A conclusão é que para este circuito a adaptação do passo de iteração não pode ser deixada a critério do PSPICE. Os tempos de processamento deste e dos outros circuitos exemplo serão apresentados em uma seção a parte, no final deste capítulo.

6.10 - Exemplo 10 - Conexão em Cascata de Três Filtros Chebyshev de Sétima Ordem.

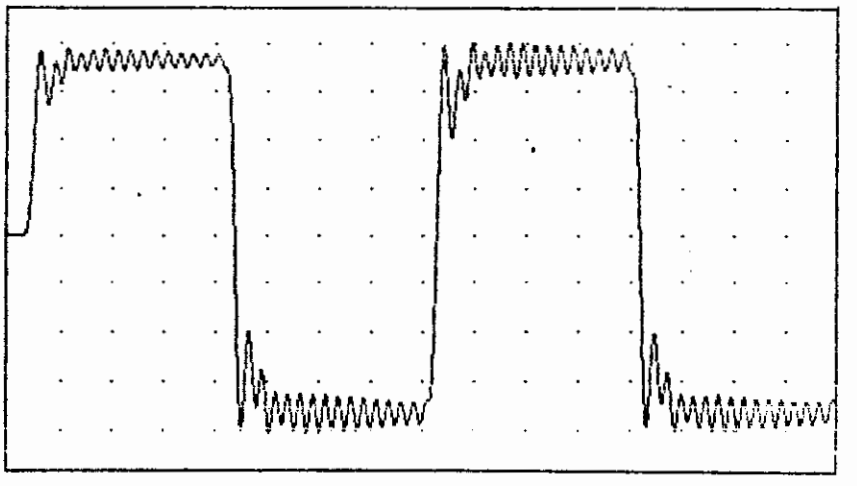
Neste exemplo apresenta-se a resposta no domínio do tempo da conexão em cascata de três estágios do filtro Chebyshev visto na seção anterior. O filtro obtido com esta associação terá cada um de seus polos triplicado.

A simulação do exemplo 10 obtida pelo SITECI para um sinal de entrada de onda quadrada de 100 Hz, amplitude 4Vpp amostrada em 500 KHz (mesma entrada da seção anterior, fig. 6.34-a) pode ser vista na figura 6.36. Na fig. 6.36-a vê-se a tensão no nó 13, sinal de saída do primeiro estágio. Nota-se que esta forma de onda é semelhante à fig. 6.34-b, da seção anterior, demonstrando não haver carregamento do segundo estágio sobre o primeiro. As saídas do segundo estágio, nó 26, e do terceiro estágio, nó 39, podem ser vistas nas figuras 6.36-b e 6.36-c, respectivamente. Simulações equivalentes obtidas pelo PSPICE podem ser vistas nas figuras 6.37. A adaptação do passo de integração do PSPICE foi limitada ao máximo de 0.04ms, pelas razões citadas na seção anterior.



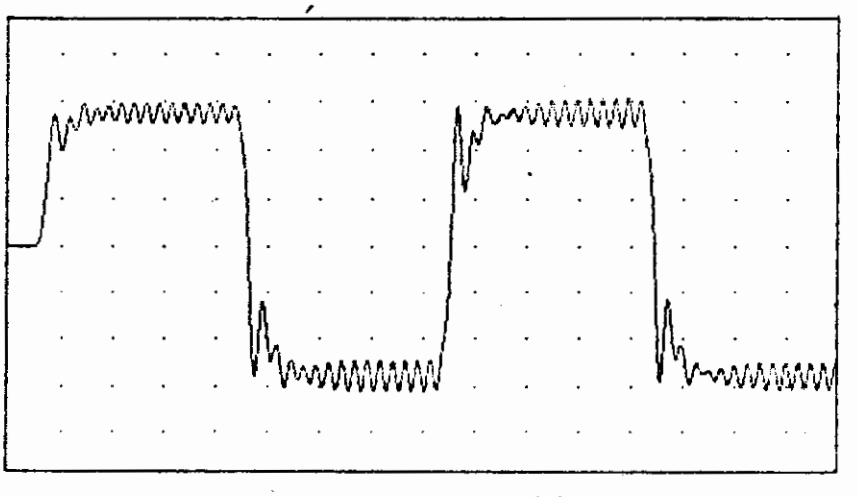
Tensão nó 13
V: 1 V/div
H: 64 amost/div
Freq. amostragem
500 KHz

(a)



Tensão nó 26
V: 2 V/div
H: 64 amost/div
Freq. amostragem
500 KHz

(b)



Tensão nó 39
V: 5 V/div
H: 64 amost/div
Freq. amostragem
500 KHz

(a)

Fig. 6.36 - Simulação do Exemplo 10, obtida pelo SITECI
Algoritmo de Discretização: Gear de Quarta Ordem
a) Tensão no nó 13, saída do primeiro estágio.
b) Tensão no nó 26, saída do segundo estágio.
c) Tensão no nó 39, saída do terceiro estágio.

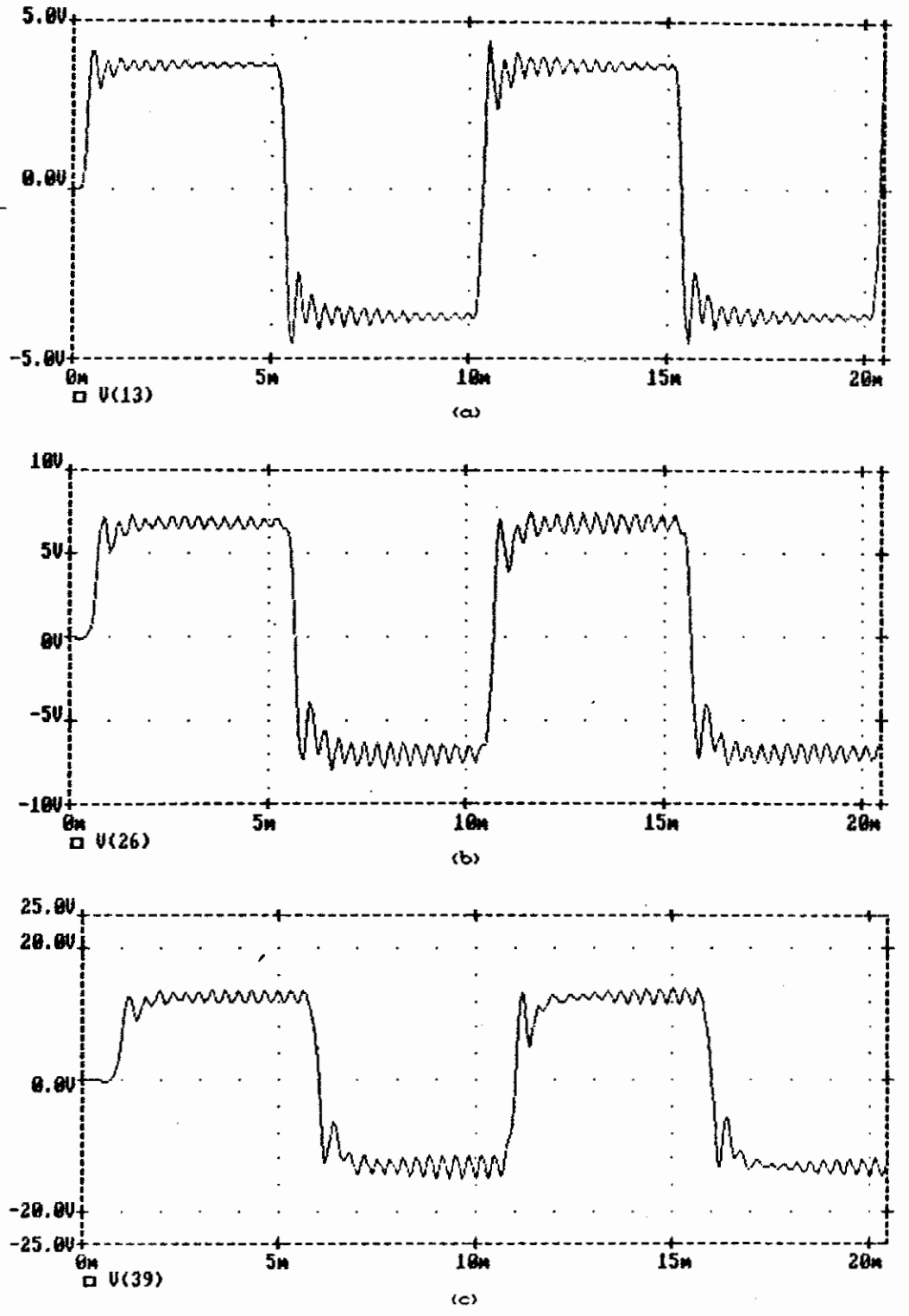
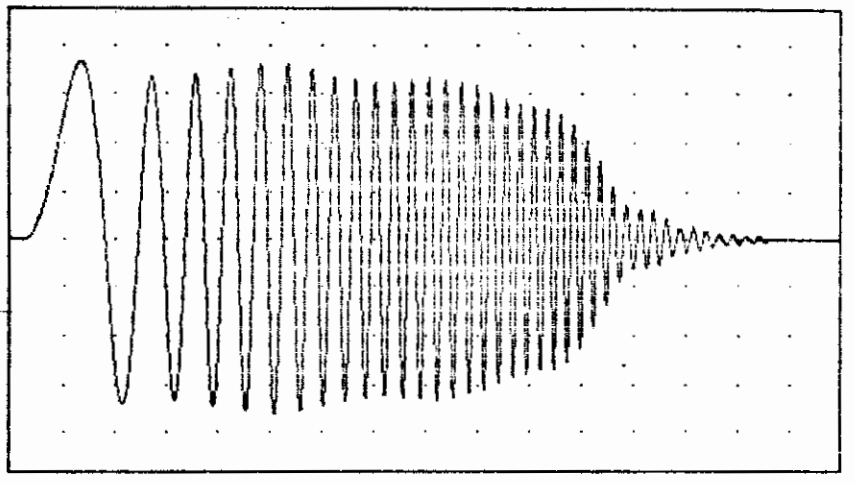
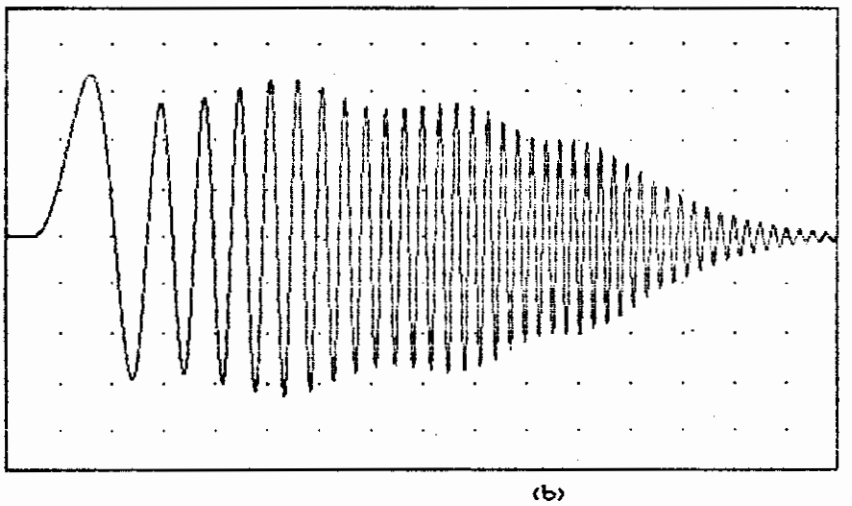


Fig. 6.37 - Simulação do Exemplo 10, obtida pelo PSPICE
Passo de iteração máximo limitado em 0.04ms.
a) Tensão no nó 13, saída do primeiro estágio.
b) Tensão no nó 26, saída do segundo estágio.
c) Tensão no nó 39, saída do terceiro estágio.



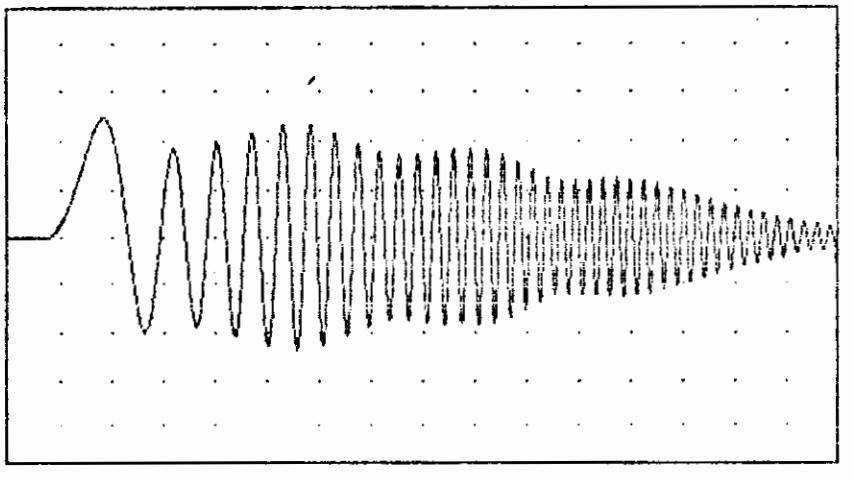
Tensão nó 13
V: 1 V/div
H: 64 amost/div
Freq. amostragem
50 KHz

(a)



Tensão nó 26
V: 2 V/div
H: 64 amost/div
Freq. amostragem
50 KHz

(b)



Tensão nó 39
V: 5 V/div
H: 64 amost/div
Freq. amostragem
50 KHz

(c)

Fig. 6.38 - Simulação do Exemplo 10, obtida pelo SITECI
Sinal de entrada: varrendura senoidal de amplitude 4Vpp
Freq. inicial = 0.025 Hz, Freq. Final = 2.5KHz
a) Tensão no nó 13, saída do primeiro estágio.
b) Tensão no nó 26, saída do segundo estágio.
c) Tensão no nó 39, saída do terceiro estágio.

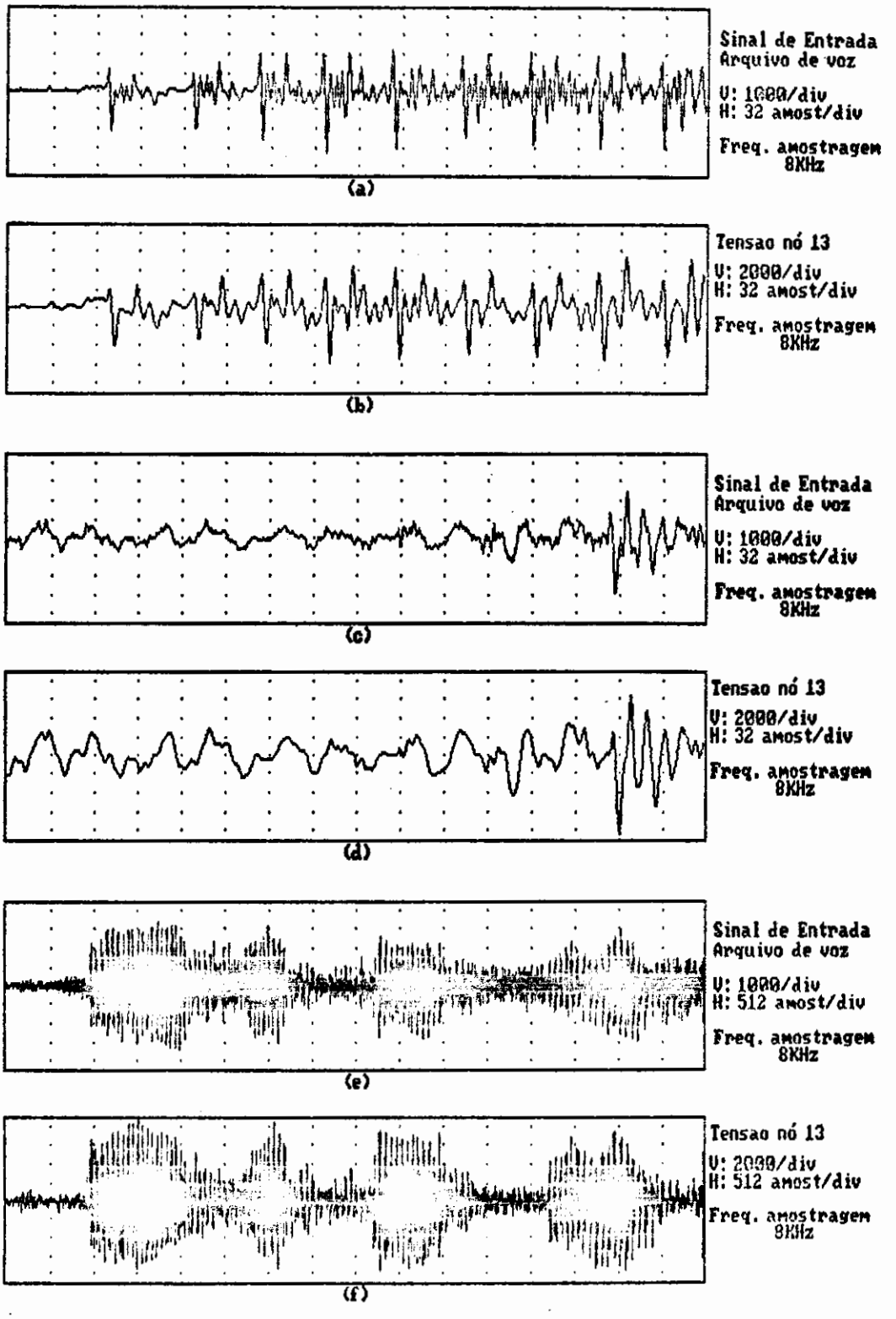


Fig. 6.39 - Simulação do Filtro Chebyshev de Sétima Ordem para sinal de Voz
Algoritmo de discretização das equações: Gear de Segunda Ordem
(a), (c) e (e) - Trechos do sinal de entrada
(b), (d) e (f) - Trechos correspondentes do sinal de saída

Na fig. 6.38 pode ser vista a resposta do exemplo 10 a um sinal de entrada de varredura senóidal. O algoritmo de discretização utilizado foi o de Gear de Segunda Ordem. É interessante notar que a tensão no nó 13 (fig. 6.38-a), saída do primeiro estágio, permite "visualizar", com as devidas ressalvas, a resposta em freqüência do filtro Chebyshev. As típicas oscilações na banda passante da resposta em freqüência podem ser notadas nesta figura. Nas figuras 6.38-b e 6.38-c são apresentados os sinais nos nós 26 e 39, correspondendo às respostas do segundo e terceiro estágios.

A fig. 6.39 apresenta a simulação do filtro de Chebyshev de sétima ordem (fig. 6.33) para um sinal de voz contido em arquivo. Este sinal foi obtido através de digitalização à taxa de 8 KHz, produzindo um arquivo de números inteiros com um total de 23552 amostras. Apenas alguns trechos da simulação são apresentados na fig. 6.39. O par de figuras 6.39-a e 6.39-b corresponde a um trecho do sinal de entrada e a respectiva saída do filtro (nó 13). Da mesma forma as figuras (c) e (e) são trechos do sinal de entrada e as figuras (d) e (f) são os trechos correspondentes do sinal de saída. Pelas quatro primeiras figuras pode ser vista a rejeição de altas freqüências efetuada pelo filtro. Para este sinal de entrada não é apresentada simulação pelo PSPICE pois este não permite descrever as fontes de sinal por arquivo externo.

6.11 - Exemplo 11 - Simulação Simultânea de Quatro Filtros Chebyshev de Sétima Ordem

Encerrando os exemplos deste capítulo apresenta-se (fig. 6.40) a simulação simultânea de quatro filtros Chebyshev de sétima ordem (cada um descrito pela fig. 6.33).

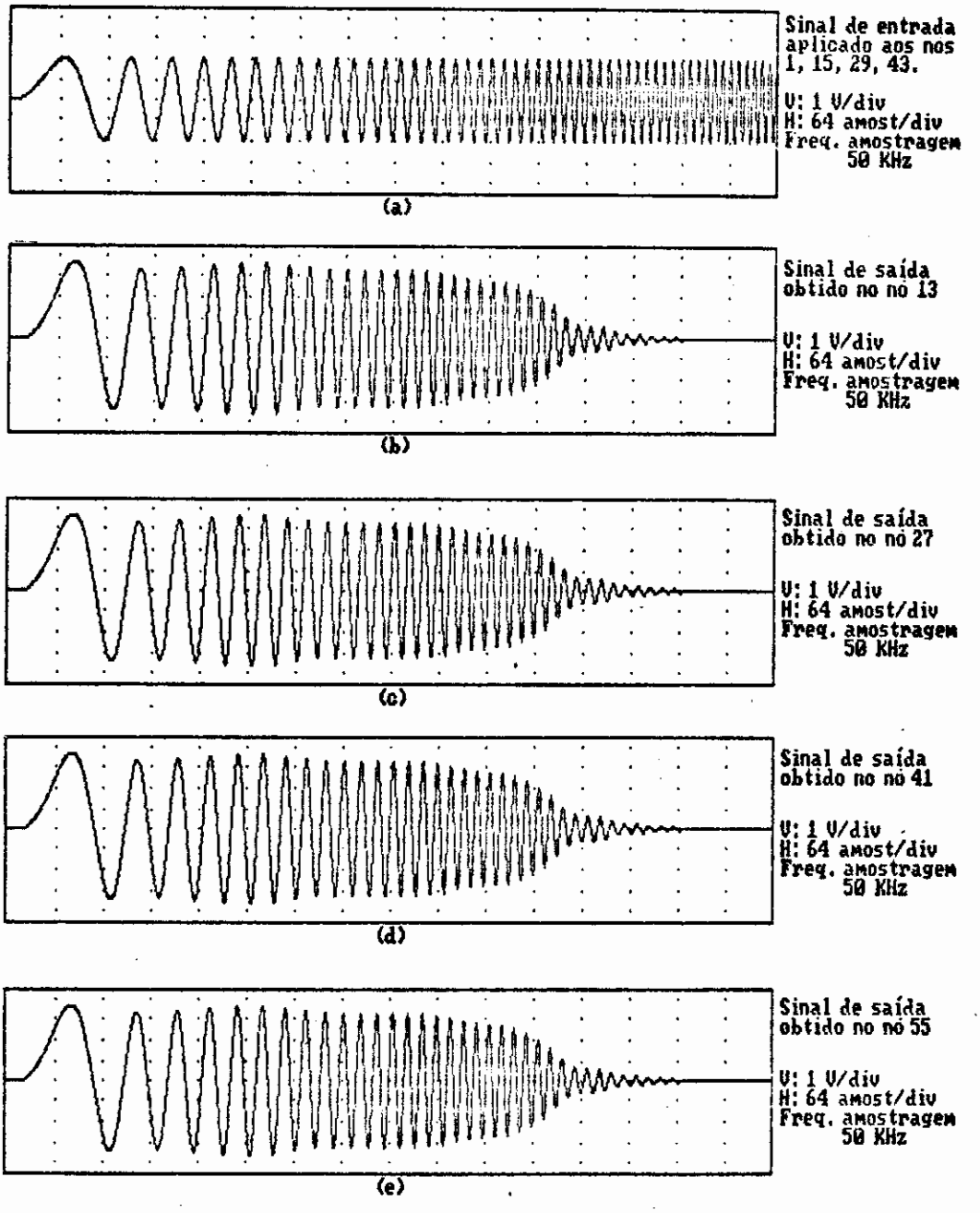


Fig. 6.40 - Simulação do Exemplo 11- Quatro Filtros Chebyshev em Paralelo.
Algoritmo de Discretização: Gear de Segunda Ordem
a) Sinal aplicado aos terminais de entrada
b), c), d), e) Sinais obtidos nos terminais de saída

O circuito do exemplo 11 é constituído de quatro filtros Chebyshev, cada qual excitado por sua fonte de tensão. Não existe conexão física entre os filtros. O objetivo desta simulação é verificar se, para uma matriz nodal modificada de ordem elevada, a propagação do erro numérico é significativa. Apesar de não existir conexão entre os filtros, a solução simultânea das equações que descrevem os quatro filtros produz um certo acoplamento entre as variáveis. Este acoplamento é tanto mais significativo quanto maior o sistema de equações e o número de operações em ponto flutuante efetuadas.

Cada um dos filtros teve um sinal de varredura senóidal aplicado à sua entrada (nós 1, 15, 29 e 43). Os sinais nos terminais de saída de cada um dos filtros (nós 13, 27, 41, 55), obtidos pelo algoritmo de Gear de Segunda Ordem, são apresentados nas figuras 6.40-(b) a 6.40-(e). Conforme pode ser visto, para este exemplo, o erro de propagação numérica não é significativo. As saídas dos quatro filtros são visualmente idênticas.

A simulação deste circuito não pôde ser efetuada pelo PSPICE, pois este acusou falta de memória. A simulação pelo SITECI utilizando-se o algoritmo Trapezoidal também apresentou a saída dos quatro filtros indênticas, no entanto a um custo computacional maior, conforme visto a seguir:

	Gear 2.a Ordem	Trapezoidal
Ordem Matriz Nodal Modificada	204	928
Não Nulos na Matriz Nodal Mod.	744 (1.79%)	1268 (1.18%)
Não Nulos nas matrizes fator	2999 (5.62%)	8395 (7.75%)
Tempo de Processamento (seg)	114	947

Nota-se que o algoritmo Trapezoidal é três vezes mais lento, para este exemplo em particular.

6.12 - Comparação dos Resultados Obtidos com o SITECI e com o PSPICE

Neste capítulo foram apresentados diversos exemplos de simulações efetuadas com o SITECI. Grande parte das simulações também foi obtida com o PSPICE. Nesta seção apresentam-se, em forma de tabela, dados que permitem comparar o desempenho dos dois simuladores, no que concerne aos circuitos lineares.

Exemplo:	1	2	3	4	5	6
SITECI						
Simulação	fig.6.4	fig.6.10	fig.6.16	fig.6.18	fig.6.19	fig.6.20
Ordem da MNM	4	5	35	6	5	16
Espars. MNM (%)	56.25	48.00	8.08	41.67	52.00	21.88
Espars. M. Fator(%)	75.00	75.00	14.61	77.78	72.00	38.67
Nro. Amostras	512	512	1024	512	512	512
Tempo de Proc.(seg)	1	1	10	2	2	3
PSPICE						
Simulação	fig.6.6	fig.6.11	fig.6.17			fig.6.21
Passo de Iter.max	livre	8us	0.4ms			16us
Iterações	166	600	536			648
Análise Transit(s)	2.47	9.17	20.76			25.70
Tempo Total(seg)	7.14	14.34	26.20			30.37

Tabela 6.1 - Dados referêntes aos exemplos 1 a 6.

Exemplo:	7	8	9	9	10	11
SITECI						
Simulação	fig.6.24	fig.6.29	fig.6.34	fig.6.39	fig.6.36	fig.6.40
Ordem da MNM	12	15	41	41	149	204
Espars. MNM (%)	23.39	20.89	7.15	7.15	2.49	1.79
Espars. M. Fator(%)	45.83	35.11	22.49	22.49	17.62	5.62
Nro. Amostras	512	512	1024	23552	1024	1024
Tempo de Proc. (seg)	3	4	29	555	164	114
PSPICE						
Simulação	fig.6.25	fig.6.11	fig.6.35 (b)		fig.6.37	
Passo de Iter. max	16us	16us	0.04ms		0.04ms	
Iterações	518	552	1697		1647	
Análise Transil(s)	10.00	15.00	241.23		931.26	
Tempo Total(seg)	10.03	17.52	258.65		984.93	

Tabela 6.2 - Dados referêntes aos exemplos 7 a 11.

A tabela 6.1 apresenta os dados relativos às simulações dos circuitos exemplos de 1 a 6, e na tabela 6.2 têm-se os dados das simulações dos exemplos 7 a 11. A cada exemplo de simulação corresponde uma coluna de dados, inicialmente os dados obtidos pelo SITECI e em seguida os obtidos pelo PSPICE. Juntamente com os dados indica-se a figura em que a simulação foi apresentada.

Os dados apresentados para a simulação com o SITECI são: a ordem da matriz nodal modificada, a esparsidade da matriz nodal modificada (porcentagem de elementos não nulos), a esparsidade das matrizes fator, o número de amostras contidas nos arquivos de entrada e o tempo total de processamento.

Para o PSPICE apresentam-se os seguintes dados: limite máximo para adaptação do passo de iteração, número de iterações efetuadas, tempo consumido pela análise transitória e tempo total da simulação. O estabelecimento do limite para a adaptação do passo de iteração foi devido aos resultados insatisfatórios obtidos quando a adaptação era deixada a cargo do PSPICE. O tempo total da simulação inclui além do tempo consumido pela análise transitória, os tempos de entrada e saída de dados e de solução do sistema de equações, entre outros.

Todos os dados apresentados foram obtidos utilizando-se um computador pessoal compatível com a linha IBM, com processador 80386SX à 16MHz, coprocessador matemático 80387SX e 736 Kbytes de memória principal.

Conforme pode ser visto, os tempos de processamento dos dois simuladores apresentam uma diferença significativa em favor do SITECI. O SITECI é de 3 a aproximadamente 8 vezes mais rápido, considerando-se resultados de precisão equivalente. Um aspecto que provavelmente contribui em muito para aumentar o tempo de processamento do PSPICE é a adaptação do passo de integração. Este procedimento torna necessário atualizar a matriz do sistema implicando em novo procedimento de fatoração.

Os dados relativos à fig. 6.39, filtro de Chebyshev de sétima ordem (exemplo 9) excitado por sinal de voz, só estão disponíveis para o SITECI, porque o PSPICE não possui

recursos adequados para entrada de sinal por arquivo. Por sua vez, o exemplo 11 só pôde ser simulado com o SITECI porque no microcomputador utilizado o PSPICE não conseguiu acomodar o circuito na memória principal disponível.

Os tempos de processamento dos exemplos 10 e 11 mostram que a preservação da esparsidade das matrizes fator é muito importante para o desempenho do simulador. Apesar da matriz do exemplo 11 ser maior, a grande esparsidade de suas matrizes fator (5.62% versus 17.62% do exemplo 10) permite que o seu tempo de processamento seja menor.

Com os exemplos de simulação vistos neste capítulo encerra-se a apresentação do Simulador Temporal de Circuitos Lineares. No capítulo seguinte ainda se apresentam as principais conclusões, bem como algumas sugestões para a continuidade das pesquisas.

7 - Conclusões

Ferramentas de auxílio como a desenvolvida neste trabalho contribuem para a automatização do procedimento de projeto de circuitos elétricos. Esta automatização torna-se cada vez mais importante face à complexidade dos circuitos que a tecnologia permite produzir.

O simulador temporal de circuitos lineares implementado, por ter os sinais de entrada/saída descritos por arquivo externo, mostrou-se adequado à utilização em sistemas de processamento de sinais. A possibilidade de utilizar-se arquivos de sinal provenientes de fenômenos físicos, como sinais de voz, com um número arbitrário de amostras (limitado apenas pela memória secundária disponível) o diferenciou de outros simuladores.

A simulação temporal de circuitos consiste

essencialmente na construção do sistema de equações de equilíbrio do circuito e sua solução ao longo do tempo. Quanto à construção das equações de equilíbrio do circuito, a formulação nodal modificada mostrou-se concisa e completa, resultando em um sistema de equações diferenciais ordinárias de primeira ordem. A discretização destas equações por algoritmos de integração multipassos permitiu sua solução eficiente em computador.

A esparsidade do sistema de equações de equilíbrio, que caracteriza os circuitos elétricos, pôde ser aproveitada no sentido de redução do tempo de processamento e memória de massa necessária pela utilização de listas encadeadas alocadas dinâmicamente. O método da bifatoração adotado para a solução do sistema de equações lineares permitiu preservar a esparsidade original das equações nas matrizes fator.

A função de biblioteca mostrou-se muito útil durante a simulação de circuitos contendo transistores e amplificadores operacionais, permitindo uma descrição concisa e menos sujeita à erros de digitação.

Ao final deste trabalho ainda se encontram dois apêndices. O apêndice "A" descreve a solução numérica de problemas de valor inicial. Ali são apresentados os algoritmos de integração que foram implementados no Capítulo 3, juntamente com considerações de erro e convergência além de regiões de estabilidade. Por sua vez, o apêndice "B" ilustra alguns exemplos relativos à solução do sistema de equações esparso que caracteriza os circuitos elétricos.

Como principais contribuições deste trabalho pode-se citar:

- Descrição das entradas por arquivos de sinais, podendo ser geradas digitalmente, provenientes de outras simulações (por exemplo, projeto de MODEM, no qual se deseja obter o sinal analógico que será transmitido via canal telefônico) ou mesmo sinais digitalizados de processos físicos (por exemplo, voz humana).

-Pelo fato dos arquivos que contém os sinais de saída serem gerados no mesmo padrão dos arquivos com os sinais de entrada, estes podem ser utilizados como entradas de outras simulações. Esta característica abre a possibilidade de um circuito ser simulado por blocos. O sinal de saída de um estágio pode ser utilizado como entrada do estágio seguinte, desde que o carregamento entre estágios seja devidamente modelado.

- Adaptação do método da bifatoração para resolução do sistema de equações nodal modificado. A eventual ocorrência de um elemento nulo na diagonal principal (freqüentemente na parte híbrida da matriz) pode levar a uma situação de singularidade. Este problema foi resolvido procurando-se manter a esparsidade das matrizes fator, bem como, minimização da propagação de erro numérico. Isto foi conseguido através de pivotamento por limiar e conseqüênte permutação de linhas da matriz nodal modificada e atualização das matrizes fator.

- Estruturação dinâmica dos dados através de listas encadeadas que permitiram eficiente aplicação do

algoritmo de bifatoração.

Como resultado das pesquisas e da comparação de exemplos simulados com o SITECI destacam-se as seguintes conclusões:

- A aplicação de técnicas de matrizes esparsas foi essencial para que os circuitos elétricos pudessem ser analisados eficientemente. A solução do sistemas de equações pelos métodos de fatoração permitiu que a esparsidade original fosse preservada.

- O pivotamento por limiar permitiu, nos exemplos executados, tanto manter o erro numérico em patamares aceitáveis como ainda preservar a esparsidade do sistema de equações original nas matrizes fator.

- Dentre os algoritmos de integração numérica analisados para a discretização das equações, para o caso de um passo de integração uniforme, os algoritmos de Gear permitiram uma implementação mais eficiente que o algoritmo trapezoidal. Isto devido à menor ordem da matriz nodal modificada obtida pela aplicação dos algoritmos de Gear.

- O tempo de processamento obtido pelo SITECI para os diferentes algoritmos de Gear é muito próximo. Desta forma a escolha da ordem do algoritmo de Gear deve ser feita por considerações de erro e de estabilidade.

- Provavelmente devido ao passo de integração uniforme, que permitiu que o sistema de equações seja fatorado apenas uma vez, e as mesmas matrizes fator sejam

utilizadas em todos os passos de integração, o tempo de processamento do SITECI foi de 3 a 8 vezes menor que o PSPICE. No caso do PSPICE o sistema de equações precisa ser atualizado e resolvido cada vez que o passo de integração é alterado.

A seguir algumas sugestões para uma eventual continuação das pesquisas iniciadas com este trabalho:

- Quanto aos componentes básicos, com os quais os circuitos são descritos, poder-se-ia acrescentar a indutância mútua, com a qual seriam representados os transformadores lineares. A biblioteca de componentes poderia ser expandida acrescentado-se parâmetros de diversos transistores e amplificadores operacionais. Outros circuitos lineares poderiam ser incluídos na biblioteca.

- A introdução da estimativa de erro poderia ser utilizada como critério para adaptação dinâmica da ordem do algoritmo de integração. Considerando-se um passo de integração uniforme seria necessário armazenar as variáveis em tantos instantes anteriores quantos sejam necessários ao algoritmo de ordem mais elevada. Poderiam ser armazenadas as matrizes fator decorrentes de cada um dos algoritmos de integração uma vez que estas não se alteram para um passo de integração uniforme.

- A estimativa de erro também poderia ser utilizada como critério para a adaptação do passo de integração. Uma vez que os sinais de entrada são definidos por arquivos de sinal externos, com passo de amostragem uniforme, haveria necessidade de interpolá-los em pontos de

grande derivada, e/ou decimá-los onde a derivada fosse pequena.

- A inclusão de componentes não-lineares é uma extensão natural do simulador linear implementado. O tratamento destes componentes levaria, no entanto à necessidade da atualização do sistema de equações a cada alteração do ponto de operação dos componentes não lineares. A atualização dos parâmetros que descrevem os componentes não lineares poderia ser feita pelo algoritmo de Newton-Raphson. Haveria desta forma, necessidade de se gerar novas matrizes fator para solução do sistema de equações nodais modificadas a cada alteração do ponto de operação.

- A análise do condicionamento da matriz nodal modificada poderia ser incluída, fornecendo subsídios adicionais para a escolha do passo de integração e/ou do algoritmo de integração a ser utilizado para a simulação de um circuito em particular.

BIBLIOGRAFIA

[1] Chua, L. O. and Lin, P. - "Computer-Aided Analysis of Electronic Circuits". Englewood Cliffs, NJ, Prentice-Hall, 1975.

[2] Pissanetsky, S. - "Sparse Matrix Technology". London, Academic Press, 1984.

[3] Morozowski. F., M. - "Matrizes Esparsas em Redes de Potência". Rio de Janeiro, LTC/ELETROBRÁS/FEESC, 1981.

[4] Brameller, A.; Allan, R. N. and Hamam, Y. M. - "Sparsity". London, Pitman Publishing, 1976.

[5] Tuinenga, P.W. - "SPICE: A guide to circuit simulation and analysis using PSpice". Englewood Cliffs, NJ, Prentice-Hall, 1988.

[6] Director, S. W. - "Circuitos Elétricos". Rio de Janeiro, Livros Técnicos e Científicos Editora S.A., 1980.

[7] Spence, R. and Burgess, J. P. - "Circuit Analysis by Computer - from algorithms to package". Englewood Cliffs, NJ, Prentice-Hall, 1986.

[8] Gray, P. E. and Searle, C. L. - "Electronic Principles - Physics, Models, and Circuits". New York, John Wiley & Sons, 1969.

[9] Caldas, R. F. - "Simulação Lógica com Temporização para Circuitos Integrados MOS". Tese de Mestrado, Universidade de Brasília, Brasília, 1987.

[10] Pacitti, T. e Atkinson, C. P. - "Programação e Métodos Computacionais" - Volume 2. Rio de Janeiro, Livros Técnicos e Científicos Editora S.A., 1977, 2a. Edição.

[11] Forsythe, G. and Moler, C. B. - "Computer Solution of Linear Algebraic Systems". Englewood Cliffs, NJ, Prentice-Hall, 1967.

[12] Demidovich, B. P. and Maron, I. A. - "Computational Mathematics". Moscou, MIR Publishers, 1987.

[13] Blume, W. - "Computer Circuit Simulation". BYTE, July 1986, pp 165-170.

[14] McNeill, D. - "Analog Circuit Analysis - An analog circuit modeling and simulator program for the Commodore 64". BYTE, July, 1986, pp 170-178.

[15] Zollenkopf, K. - "Bi-factorisation: basic computational algorithm and programming techniques. In *Large Sparse Sets of Linear Equations*". London, Academic Press, 1971, pp. 75-96.

[16] Ho, C. W.; Ruehli, A. E. and Brennan, P. A. - "The Modified Nodal Approach to Network Analysis". IEEE Transactions on Circuits and Systems, VOL CAS-22, No. 6, June 1975, pp 504-509.

[17] Hajj, I. N.; Yang, P. and Trick, T. N. - "Avoiding Zero Pivots in the Modified Nodal Approach". IEEE Transactions on Circuits and Systems, VOL CAS-28, No. 4, April 1981, pp 271-279.

[18] Appel, G. - "A Nodal Network Analysis Prgram". RF Design, November 1989, pp 82-83.

[19] Webb, R. C. - "SANA: Nodal Network Signal and Noise Analysis and Optimization". RF Design, November 1989, pp 29-31.

[20] Willoughby, R. A. - "Sparse Matrix Algorithms and Their Relation to Problem Classes and Computer Architecture. In *Large Sparse Sets of Linear Equations*". London, Academic Press, 1971, pp. 75-96.

[21] Nagel, L. and Rohrer, R. - "Computer analysis of nonlinear circuits excluding radiation (CANCER)". IEEE J. Solid-State Circuits, vol. SC-6, pp. 166-182, Aug. 1971.

[22] Calahan, D. A. - "Computer-Aided Network Design". New York, McGraw-Hill,' 1972.

[23] Weeks, W. T. et alii. - "Network analysis using a sparse tableau with tree selection to increase sparseness". IEEE Proc. Inst. Symp. Circuit Theory, Toronto, Ont., Canada, pp. 165-168, Apr. 1973.

[24] ———. - "1620 Electronic Circuit Analysis Program (ECAP) [1620-EE-02X2] user's manual". IBM Application Program File H20-0170-1, 1965.

[25] McCalla, W. J. and Howard, W. G. Jr. - "BIAS-3: A program for the nonlinear dc analysis of bipolar transistor circuits". IEEE J. Solid-State Circuits, vol. SC-6, pp. 14-19, Feb. 1971.

[26] Reid, J. K. - "A note on the stability of Gaussian elimination". J. Inst. Math. Appl. 8, pp. 374-375, 1971.

[27] Millman, J e Halkias, C. C. - "Eletrônica: dispositivos e circuitos, vol. 1 e 2". São Paulo, McGraw-Hill do Brasil, 1981.

[28] Ruehli, A. E., (Editor) - "Circuit Analysis, Simulation and Design". Amsterdam, North-Holland, 1986.

[29] Nascimento, F. A. O. et allii - "PASOR - Programa de Análise, Cálculo de Sensibilidade e Otimização de Redes" - COPPE - RJ- 1985 (Não publicado).

APÊNDICE A

A - SOLUÇÕES NUMÉRICAS DE PROBLEMAS DE VALOR INICIAL

A solução de um sistema de equações diferenciais ordinárias pode ser obtida quando estas estão sujeitas a um conjunto de restrições impostas à função e suas derivadas, para um particular valor de x ou para alguns particulares valores de x. Se todas as restrições, ou condições, são referidas a um mesmo valor de x, definido como valor inicial x_o, tem-se a formulação conhecida como "problema de valor inicial". Caso as restrições correspondam a diversos valores da variável independente, denomina-se "problema de valor de contorno" [10]. De uma forma genérica considera-se um problema de valor inicial como um conjunto de equações lineares do tipo

$$\dot{x} = f(x,t), \qquad (A.1)$$

onde

$\dot{x}$- derivada de x em relação ao tempo

f - uma função linear de x e t

x - variável dependente

t - variável independente (em geral representa tempo)

um vetor de condições iniciais x_o e um intervalo de tempo T.

Um vetor de funções do tempo $x = \hat{x}(t)$ é dita uma solução deste problema de valor inicial se $\hat{x}(t_o) = x_o$ e $\hat{x}(t) = f(\hat{x}(t),t)$ para todo $t \in [t_o, t_o+T]$.

Para se obter a solução numericamente para $\hat{x}(t)$, deve-se dividir o intervalo de tempo T em pequenos incrementos de tempo, onde cada incremento de tempo é conhecido como passo de integração [1], definido como

$$h_i \triangleq (\Delta t)_i,$$

onde

h_i - "i-ésimo" passo de integração

$(\Delta t)_i$ - incremento de tempo

O objetivo é encontrar $\hat{x}(t)$ em

$$t_k \triangleq t_o + \sum_{i=1}^{k} h_i \quad , \quad k = 1,2,\ldots,N, \text{ onde } t_N = t_o + T.$$

Apesar de existirem diversos algoritmos para a solução de problemas de valor inicial, a maioria deles é tratada tomando duas abordagens básicas: A abordagem por expansão em série de Taylor e a abordagem por aproximação polinomial [Chua & Lin, 1975]. Algoritmos baseados na expansão em série de Taylor são usualmente conhecidos por *algoritmos de Runge-Kutta*, enquando os baseados na aproximação polinomial são usualmente conhecidos por *algoritmos de integração numérica multipassos*.

Neste trabalho pelo fato dos sinais de entrada serem amostrados a uma taxa constante adota-se um passo de

integração uniforme $h_i = h$ na formulação dos algoritmos, desta forma, o passo temporal é simplificado para

$$t_n = t_o + nh \; ; \qquad n = 1,2,3,\ldots N$$

No presente trabalho os algoritmos de Runge-Kutta não são interessantes pois o valor da função $f(x_n,t_n)$ precisa ser avaliada p[1] vezes dentro de cada passo e os sinais são conhecidos apenas nos instantes de integração. Além disso, estes valores intermediários da função não são utilizados em nenhum passo computacional subsequente. Desta forma, computacionalmente, os algoritmos de Runge-Kutta não são tão eficientes como os algoritmos multipassos, que fazem proveito dos valores previamente calculados.

Além disso, a literatura relata pesquisas em que os métodos de Runge-Kutta e métodos preditor-corretor são considerados ineficientes quando as equações diferenciais que descrevem o circuito são "stiff" (i.e. quando é grande o espectro de constantes de tempo do sistema). [20]

A.1 - Solução Numérica por Aproximações Polinomiais

Se a solução exata de um problema de valor inicial $\dot{x} = f(x,t)$, $x(t_o) = x_o$, é dada por um polinômio de grau k:

$$\hat{x}(t) = x_o + x_1 t + x_2 t^2 + \ldots + x_k t^k \qquad (A.2)$$

onde $x_o, x_1, \ldots x_k$ são constantes e se for conhecido o valor exato de $\hat{x}(t)$ e de sua derivada primeira $\hat{x}'(t)$ em $t = t_n$, $t_{n-1}, t_{n-2}, \ldots, t_{n-p}$; i.e., $\hat{x}(t_n)$, $\hat{x}(t_{n-1}), \ldots \hat{x}(t_{n-p})$, e $\hat{x}'$

---

[1] Onde p é a ordem do algoritmo.

(t_n), $\hat{x}'(t_{n-1}), \ldots \hat{x}'(t_{n-p})$ [Chua & Lin, 1] estabelecem a seguinte definição:

Definição *Uma fórmula de integração numérica de ordem k é um algoritmo capaz de calcular o valor exato* $\hat{x}(t_{n+1})$ *para um problema de valor inicial com solução na forma de um polinômio de grau k.*

A fórmula genérica de um algoritmo de integração por aproximações polinomiais é dada por

$$x_{n+1} = a_o x_n + a_1 x_{n-1} + \ldots + a_p x_{n-p} +$$

$$h\left[b_{-1} f(x_{n+1}, t_{n+1}) + b_o f(x_n, t_n) + \ldots + b_p f(x_{n-p}, t_{n-p})\right]$$

$$\boxed{x_{n+1} = \sum_{i=o}^{p} a_i x_{n-i} + h \sum_{i=-1}^{p} b_i f(x_{n-i}, t_{n-i})} \qquad (A.3)$$

onde $a_o, a_1, \ldots a_p, b_{-1}, b_o, b_1, \ldots$ e b_p são $2p + 3$ coeficientes a determinar, de tal modo que se a solução é um polinômio e se os valores previamente calculados são exatos, então a eq.(3.2) fornece o valor exato $x_{n+1} = \hat{x}(t_{n+1})$.

É claro que se a solução exata *não* é um polinômio, uma fórmula de integração numérica fornecerá apenas um valor aproximado x_{n+1} e não o valor exato $\hat{x}(t_{n+1})$. No entanto, tendo em vista o clássico *teorema da aproximação de Weierstrass* [2] que afirma que *toda função contínua pode ser*

---

[2] Hille, E. Analysis, Vols I and II Lexington, Mass.: Xerox College Publishing, 1966.

aproximada com erro arbitrariamente pequeno dentro de qualquer intervalo fechado por um polinômio de ordem suficientemente grande, é claro que se utilizamos uma fórmula de integração de ordem suficientemente grande, podemos em tese calcular $\hat{x}(t_{n+1})$ com qualquer precisão desejada. Na prática entretanto, o esforço computacional e o erro de arredondamento crescem com a ordem da fórmula de integração, e apenas valores de k < 10 são de interesse prático [1].

Ao contrário dos algoritmos baseados em séries de Taylor, as informações de passos anteriores são utilizadas na maioria das fórmulas de integração numéricas para o cálculo de x_{n+1}, como ilustrado na figura A.1

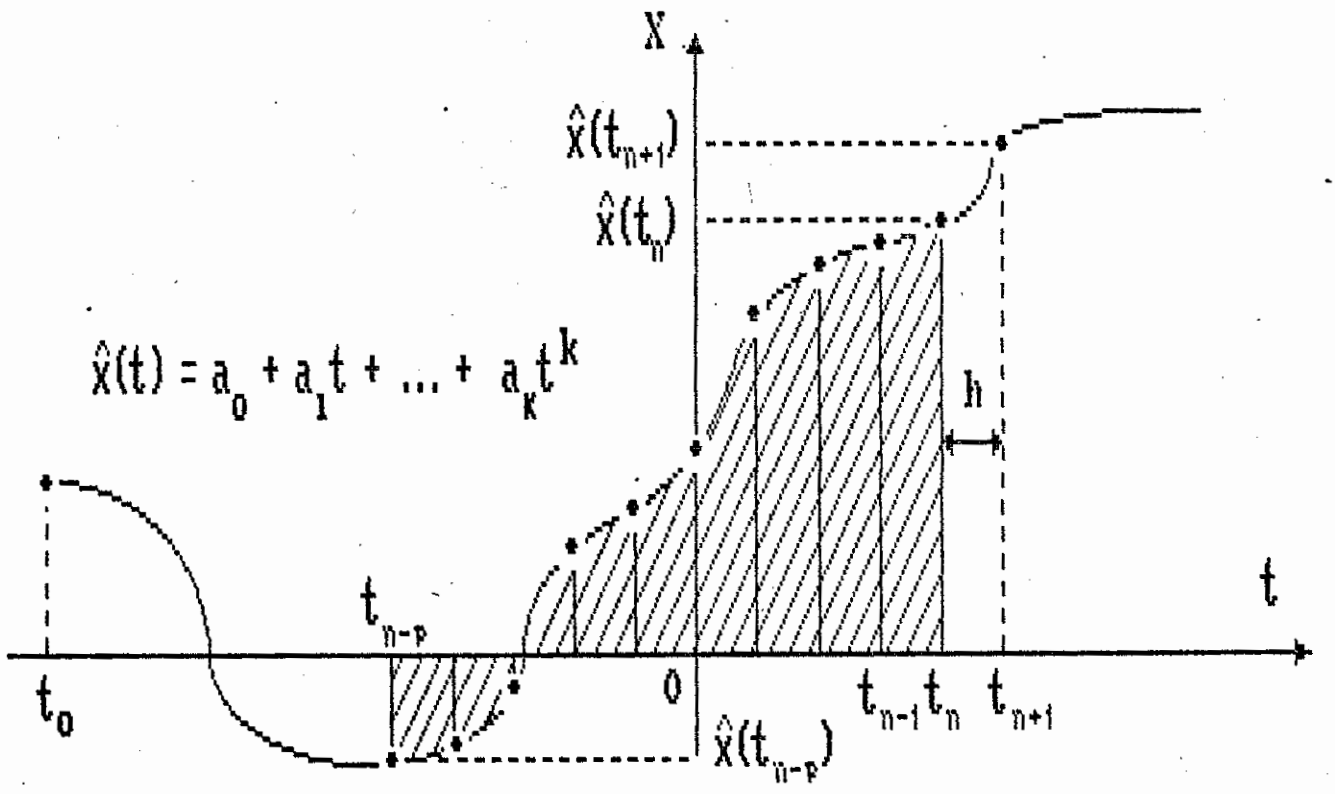


Fig. A.1 Interpretação Geométrica de um algoritmo multipassos

Por isso os algoritmos de integração por aproximações polinomiais com p > 1 são chamados de algoritmos multipassos, em contraste com os algoritmos de Taylor, que são algoritmos de passo único.

A. 2 - CONSISTÊNCIA

A determinação dos 2p + 3 coeficientes correspondentes à solução da equação A.3 de tal forma que A.3 seja exata para todas as soluções polinomiais de grau menor ou igual a k, é abordada considerando-se os seguintes casos

1. $k = 0$, $x = \hat{x}(t) = \alpha_o$ (α_o constante)

A classe de problemas de valor inicial com solução $\hat{x}(t) = \alpha_o$ é $\dot{x} = f(x,t) = 0$. Desta forma as restrições para uma solução exata são

$$x_{n+1} = \alpha_o, \quad x_{n-1} = \alpha_o, \quad f(x_{n-1}, t_{n-1}) = 0 \qquad (A.4)$$

Substituindo (A.4) em (A.3) obtem-se

$$\alpha_o = \sum_{i=0}^{p} a_i \alpha_o \qquad (A.5)$$

Desta forma as restrições para as soluções polinomiais de ordem 0 (zero) serem exatas são simplesmente

$$\sum_{i=0}^{p} a_i = 1 \qquad (A.6)$$

2. $k = 1$, $x = \hat{x}(t) = \alpha_o + \alpha_1 t$ (α_o e α_1 constantes)

A classe de problemas de valor inicial com solução

$\hat{x}(t) = \alpha_0 + \alpha_1 t$ é $\dot{x} = f(x,t) = \alpha_1$.

Por conveniência, escolhe-se $t_n = 0$; desta forma, $t_{n+1} = h$, $t_{n-1} = -h$, $t_{n-2} = -2h$, ..., $t_{n-p} = -ph$. As restrições para solução exata são então

$$x_{n+1} = \alpha_0 + \alpha_1 h, \quad x_{n-1} = \alpha_0 - \alpha_1(ih), \quad f(x_{n-1}, t_{n-1}) = \alpha_1 \quad (A.7)$$

Substituindo (A.7) em (A.1) obtem-se

$$\alpha_0 + \alpha_1 h = \sum_{i=0}^{p} a_i [\alpha_0 - \alpha_1(ih)] + h \sum_{i=-1}^{p} b_i \alpha_1 \quad (A.8)$$

Substituindo (A.6) em (A.8) obtem-se

$$\alpha_1 h = \sum_{i=0}^{p} [-a_i \alpha_1(ih)] + h \sum_{i=-1}^{p} b_i \alpha_1 \quad (A.9)$$

. Dividindo os dois lados por $\alpha_1 h$ obtem-se as seguintes restrições para solução exata de polinômios de grau 1.

$$\sum_{i=0}^{p} (-i)\, a_i + \sum_{i=-1}^{p} b_i = 1 \quad (A.10)$$

3. $k = 2$, $x = \hat{x}(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2$ (α_0, α_1 e α_2 constantes)

Através do mesmo desenvolvimento obtem-se

$$\sum_{i=1}^{p} i^2 a_i + 2 \sum_{i=-1}^{p} (-i) b_i = 1 \quad (A.11)$$

4. No caso genérico, $x = \hat{x}(t) = \alpha_o + \alpha_1 t + \alpha_2 t^2 + \ldots \alpha_k t^k$ obtem-se

$$\sum_{i=o}^{p} a_i = 1$$

$$\sum_{i=1}^{p} (-i)^j a_i + j \sum_{i=-1}^{p} (-i)^{j-1} b_i = 1, \qquad j = 1,2,\ldots k \tag{A.12}$$

Condição de solução exata para um polinômio de ordem k

A primeira equação em (A.12) pode ser obtida da segunda substituindo-se o índice do somatório de i = 1 por i = 0 e variando-se j = 0,1,2,...,k.

A equação (A.12) fornece as relações que devem ser satisfeitas pelos 2p + 3 coeficientes da Equação (A.3) para que este algoritmo forneça a solução exata para x_{n+1} sempre que a solução $\dot{x}=f(x,t)$ é um polinômio de grau menor ou igual a k. Obviamente poder-se-ia obter as restrições dos coeficientes para que o algoritmo de integração seja exato sempre que as soluções pertencerem a uma certa classe de funções - como exponenciais - em vez de polinômios. No entanto, a maioria dos algoritmos multipassos utilizados hoje em dia são baseados no critério de solução exata para polinômios. A ordem do algoritmo é definida como o grau do polinômio de maior grau para o qual a solução é exata. Diferentes escolhas dos coeficientes levam a diferentes famílias de algoritmos de integração.

Um algoritmo multipassos descrito por

$$x_{n+1} = \sum_{i=0}^{p} a_i x_{n-i} + h \sum_{i=-1}^{p} b_i f(x_{n-i}, t_{n-i})$$

é dito consistente se as duas condições seguintes são satisfeitas

$$\sum_{i=0}^{p} a_i = 1$$

$$\sum_{i=0}^{p} (-i)\ a_i + \sum_{i=-1}^{p} b_i = 1$$

Ou seja, um algoritmo multipassos consistente é um algoritmo que fornece a solução numérica exata para um problema de valor inicial $\dot{x} = f(x,t)$ tendo uma solução polinomial <u>linear</u> $\hat{x}(t) = \alpha_0 + \alpha_1 t$. Desta forma, todos os algoritmos multipassos de ordem $k \geq 1$ são consistentes.

A.3. Famílias de Algoritmos de Integração Numérica por Aproximações Polinomiais

Estabelecendo-se restrições particulares aos coeficientes da fórmula de integração genérica (A.3) obtem-se famílias de algoritmos de integração multipassos. Existem três famílias mais utilizados na prática, os algoritmos de Adams-Bashforth, de Adams-Moulton e os de Gear. O método de integração de Euler progressivo é um caso particular dos algoritmos de Adams-Bashforth, os métodos de Euler regressivo e trapezoidal são casos particulares dos algoritmos de Adams-Moulton.

Será visto mais adiante que as regiões de estabilidade dos algoritmos de Adams-Bashforth são bastante restritas, desta forma este trabalho concentrou esforços nos algoritmos de Adams-Moulton e Gear por terem as melhores características de estabilidade.

A.3.1 - Algoritmos de Adams-Bashforth

O algoritmo de ordem k de Adams-Bashforth é um algoritmo de integração explícito obtido por:

$$p = k - 1, \quad a_1 = a_2 = \ldots a_{k-1} = 0, \quad b_{-1} = 0 \qquad (A.13)$$

O algoritmo de Euler progressivo é obtido para k = 1, de onde obtem-se

$$x_{n+1} = x_n + hf(x_n, t_n) \qquad (A.14)$$

Conforme pode ser visto em (A.14) o novo valor x_{n+1} depende apenas de valores já disponíveis.

A.3.2 - ALGORITMOS DE ADAMS-MOULTON

O algoritmo de ordem k de Adams-Moulton é um algoritmo de integração <u>implícito</u> obtido por

$$p = k - 2, \quad a_1 = a_2 = \ldots = a_{k-2} = 0 \qquad (A.15)$$

Substituindo na equação (A.3) obtem-se

$$x_{n+1} = a_0 x_n + h \left\{ b_{-1} f(x_{n+1}, t_{n+1}) + b_0 f(x_n, t_n) + b_1 (f x_{n-1}, t_{n-1}) \right.$$

$$\left. + \ldots + b_{k-2} f(x_{n-k+2}, t_{n-k+2}) \right\} \qquad (A.16)$$

Os $k + 1$ coeficientes a_0, b_{-1}, b_0, b_1 ... b_{k-2} são determinados de tal forma que a equação (A.16) seja a solução exata para todos os polinômios de ordem k. Observa-se que $k + 1$ é o número exato de coeficientes desconhecidos que precisam ser determinados para um polinômio de ordem k utilizando-se (A.16). Da equação (A.6) vê-se imediatamente que $a_0 = 1$.

Os coeficientes b_i dependem da ordem do algoritmo, desta forma pode-se relacionar $b_i(k)$ para enfatizar esta dependência.

$$x_{n+1} = x_n + h \sum_{i=-1}^{k-2} b_i(k) \, f(x_{n-i}, t_{n-i}) \qquad (A.17)$$

Substituindo os coeficientes de (A.17) na condição de solução exata (A.12) obtem-se

$$\sum_{i=-1}^{k-2} (-i)^{j-1} b_i(k) = \frac{1}{j} \quad , \quad j = 1,2,\ldots,k \qquad (A.18)$$

Que quando expandido fornece um sistema de k equações lineares com k incógnitas $b_{-1}(k)$, $b_0(k)$, ... $b_{k-2}(k)$:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & \ldots\ldots & 1 \\ 1 & 0 & -1 & -2 & -3 & \ldots\ldots & -(k-2) \\ 1 & 0 & 1 & 4 & 9 & \ldots\ldots & +(k-2)^2 \\ 1 & 0 & -1 & -8 & -27 & \ldots\ldots & -(k-2)^3 \\ \vdots & & & & & & \\ 1 & 0 & (-1)^{k-1} & (-2)^{k-1} & (-3)^{k-1} & \ldots\ldots & [-(k-2)]^{k-1} \end{bmatrix} \begin{bmatrix} b_{-1}(k) \\ b_0(k) \\ b_1(k) \\ b_2(k) \\ \\ b_{k-2}(k) \end{bmatrix} = \begin{bmatrix} 1 \\ 1/2 \\ 1/3 \\ 1/4 \\ \\ 1/k \end{bmatrix}$$

(A.19)

A solução de (A.19) fornece então a solução única dos demais coeficientes do algoritmo de Adams-Moulton de ordem k. Na tabela A.1 têm-se os algoritmos de Adams-Moulton para k = 1 até k = 4.

Ordem	
1	$x_{n+1} = x_n + hf(x_{n+1}, t_{n+1})$
2	$x_{n+1} = x_n + h \left\{ \frac{1}{2} f(x_{n+1}, t_{n+1}) + \frac{1}{2} f(x_n, t_n) \right\}$
3	$x_{n+1} = x_n + h \left\{ \frac{5}{12} f(x_{n+1}, t_{n+1}) + \frac{8}{12} f(x_n, t_n) - \frac{1}{12} f(x_{n-1}, t_{n-1}) \right\}$
4	$x_{n+1} = x_n + h \left\{ \frac{9}{24} f(x_{n+1}, t_{n+1}) + \frac{19}{25} f(x_n, t_n) - \frac{5}{24} f(x_{n-1}, t_{n-1}) + \frac{1}{24} f(x_{n-2}, t_{n-2}) \right\}$

Tabela A.1 - Algoritmos de Adams-Moulton

A.3.3 - ALGORITMOS DE GEAR

O algoritmo de Gear de ordem k é um método de integração implícito obtido por

$$p = k-1, \qquad b_0 = b_1 = b_2 = \ldots = b_{k-1} = 0 \qquad (A.20)$$

Substituindo na equação (A.3) obtem-se

$$x_{n+1} = a_0(k)x_n + a_1(k)x_{n-1} + a_2(k)x_{n-2} + \ldots + a_{k-1}(k)x_{n-k+1} + h\left[b_{-1}(k)\, f\,(x_{n+1}, t_{n+1}) \right] \qquad (A.21)$$

Onde a_i é escrito como $a_i(k)$ para enfatizar a sua dependência da ordem k do método. Os k + 1 coeficientes $a_0(k)$, $a_1(k)$, $a_2(k)$, ... $a_{k-1}(k)$, e $b_{-1}(k)$ são determinados de tal forma que (A.21) seja a solução exata para todos os

polinômios de grau menor ou igual a k.

Utilizando a condição (A.12) obtem-se

$$\sum_{i=0}^{k-1} (-i)^j a_i(k) + j b_{-1}(k) = 1, \quad j = 0,1,2,\ldots k-1 \qquad (3.22)$$

A equação (A.22) expandida consiste de um sistema linear de k + 1 equações em k + 1 incógnitas, $a_0(k)$, ... $a_{k-1}(k)$ e $b_{-1}(k)$:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & \ldots\ldots & 1 & 0 \\ 0 & -1 & -2 & -3 & \ldots\ldots & -(k-1) & 1 \\ 0 & 1 & 4 & 9 & \ldots\ldots & (k-1)^2 & 2 \\ 0 & -1 & -8 & -27 & \ldots\ldots & [-(k-1)]^3 & 3 \\ \vdots & & & & & & \\ 0 & (-1)^k & (-2)^k & (-3)^k & \ldots\ldots & [-(k-1)]^k & k \end{bmatrix} \begin{bmatrix} a_0(k) \\ a_1(k) \\ a_2(k) \\ a_3(k) \\ \\ b_{-1}(k) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

... (A.23)

A solução de (A.23) fornece então a solução única dos demais coeficientes do algoritmo de Gear. Na tabela A.2 tem-se os algoritmos de Gear para k = 1 a k = 4

Ordem

1	$x_{n+1} = x_n + h\, f(x_{n+1}, t_{n+1})$
2	$x_{n+1} = \frac{4}{3} x_n - \frac{1}{3} x_{n-1} + h\, \frac{2}{3} f(x_{n+1}, t_{n+1})$
3	$x_{n+1} = \frac{18}{11} x_n - \frac{9}{11} x_{n-1} + \frac{2}{11} x_{n-2} + h\, \frac{6}{11} f(x_{n+1}, t_{n+1})$
4	$x_{n+1} = \frac{48}{25} x_n - \frac{36}{25} x_{n-1} + \frac{16}{25} x_{n-2} - \frac{3}{25} x_{n-3} + h\, \frac{12}{25} f(x_{n+1}, t_{n+1})$

Tabela A.2 - Algoritmos de Gear

A.4 - Considerações de Erro

Desde que nenhum método numérico é capaz de encontrar $\hat{x}(t_k)$ exatamente, denomina-se $\hat{x}_k$ a solução calculada em $t = t_k$, e define-se

$$\hat{\varepsilon}_k \triangleq || \hat{x}(t_k) - \hat{x}_k || \qquad (A.24)$$

como o erro total em $t = t_k$. O erro total sempre pode ser analisado como sendo composto de duas componentes: um *erro de truncamento* $\hat{\varepsilon}_{tk}$ e um *erro de arredondamento* $\hat{\varepsilon}_{rk}$. O erro de truncamento é o erro que ocorre quando uma aritmética de precisão infinita é utilizada. Este erro pode ser classificado como um erro algorítmico, desde que depende da natureza do método numérico utilizado no cálculo de $\hat{x}_k$. O erro de arredondamento é o erro adicional devido ao comprimento finito da palavra do computador. Este erro pode ser classificado como um erro da máquina, pois depende apenas da precisão aritmética do computador.

A figura A.2 ilustra estas duas situações

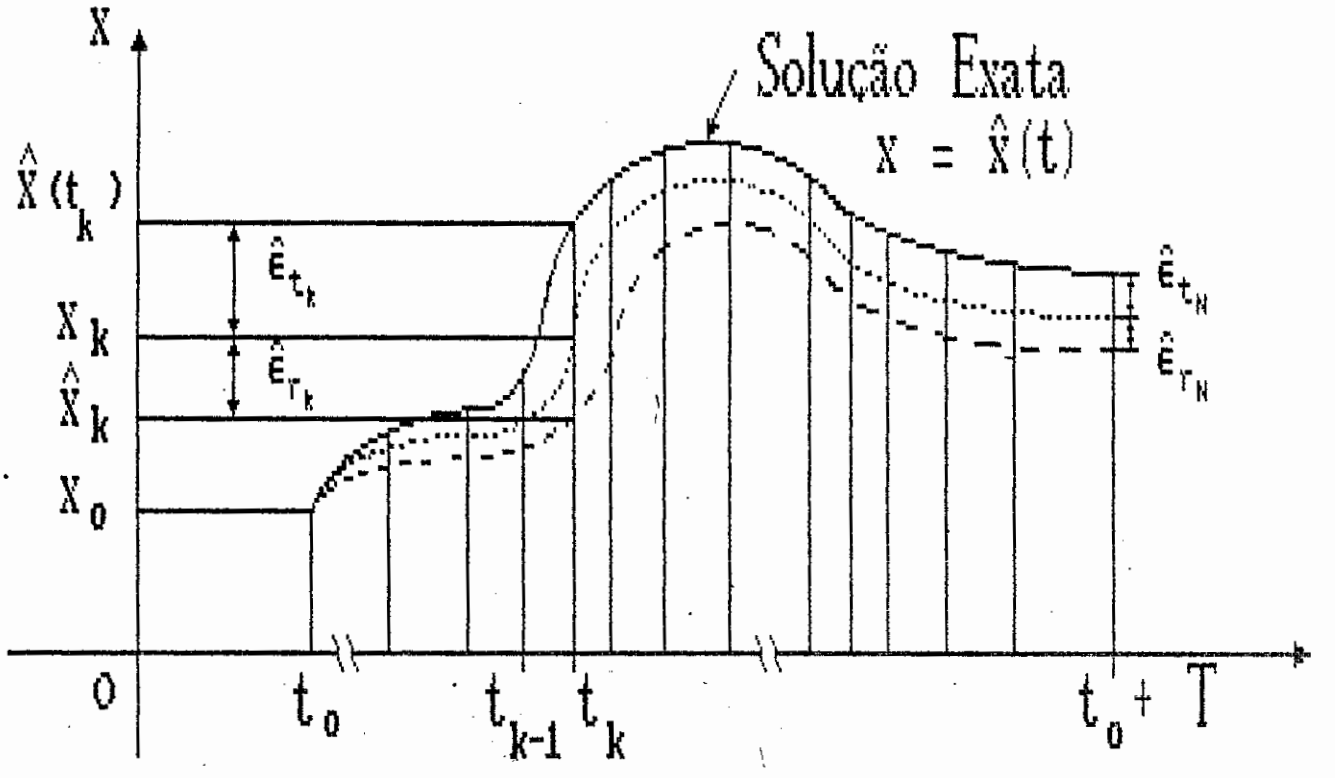


Fig. A.2 Diferença entre a solução exata $x = \hat{x}(t)$ e a solução calculada $\hat{x}(t_N)$ sem erro de arredondamento. $x(t_N)$ é a solução tendo erro de truncamento e de arredondamento.

Através da Fig. A.2 é claro que tanto o erro de truncamento como o erro de arredondamento tendem a acumular-se com o aumento do número de passos de integração. Desta forma para comparar a precisão entre dois algoritmos, é necessário compará-los para o mesmo tempo $t = t_k$ e as mesmas condições iniciais. Para propósitos de comparação, no entanto, basta considerar o *erro local* em $t = t_1$, onde não há erro anterior, i é:

$$\varepsilon_1 \triangleq \| \hat{x}(t_1) - \hat{x}_1 \| \qquad (A.25)$$

Desde que o erro local de arredondamento depende essencialmente do tipo de dados da linguagem de programação e do computador utilizado, ele não pode ser reduzido uma vez que um certo ambiente de computação é escolhido. Entretanto, diferentes algoritmos reagem de modo diferente com respeito a quão rápido o erro de arredondamento é propagado. É importante reconhecer que o erro *total* de arredondamento em $t = t_k$ não é apenas a soma dos erros locais de arredondamento que ocorrem a cada iteração, isto porque o erro de arredondamento local pode tanto aumentar ou diminuir durante o processo computacional. Um algoritmo com a propriedade desejável de que o erro local de arredondamento caia com o aumento do número de passos é dito *numericamente estável*. Caso contrário este algoritmo é numericamente instável e não tem interesse prático [1].

A.4.1 - Erro de Truncamento dos Algoritmos de Integração Multipassos

Pode ser visto em [1] - pg 458, que o erro local de truncamento ε_T de um algoritmo de integração numérica de ordem k é dado por

$$\varepsilon_T = C_k \, \hat{x}^{(k+1)}(\hat{\gamma}) h^{k+1} = O(h^{k+1}) \qquad (A.26)$$

onde

$$-ph < \hat{\gamma} < h$$

e

$$C_k \triangleq \frac{1}{(k+1)!} \left\{ (p+1)^{k+1} - \left[\sum_{i=0}^{p-1} a_i (p-i)^{k+1} + (k+1) \sum_{i=-1}^{p-1} b_i (p-i)^k \right] \right\}$$

... (A.27)

A equação (A.26) mostra que o *Erro de Truncamento* exato é função do valor de $\hat{x}^{(k+1)}$ $(\hat{\gamma})$. Porém $\hat{\gamma}$ não pode ser determinado exatamente, é algum valor de "t" entre -ph e h. O resultado importante é que o erro de truncamento é da ordem de $O(h^{k+1})$, o que fornece a estimativa do erro quando a reposta exata não é um polinômio de ordem "k".

A.5 - Estabilidade de Algoritmos de Integração Multipassos

A comparação e a análise das propriedades de estabilidade dos algoritmos de integração multipassos é geralmente feita com referência ao seguinte problema de valor inicial

$$\dot{x} = -\lambda x, \quad x(0) = 1 \qquad (A.28)$$

onde λ é uma constante *complexa*. A necessidade de permitir que λ seja complexo vem das considerações de estabilidade da solução de um sistema de equações lineares

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} \qquad (A.29)$$

onde A é uma matriz de constantes reais, n x n.

Assumindo que a matriz A é diagonalizável [26] na forma

$$\Lambda = \mathbf{M}\mathbf{A}\mathbf{M}^{-1} \qquad (A.30)$$

onde Λ é uma matriz diagonal n x n cujos valores são os autovalores de A, então definindo $\mathbf{y} = \mathbf{M}\mathbf{x}$, obtem-se o seguinte sistema equivalente de equações lineares desacopladas

$$\dot{y} = \Lambda y \qquad (A.31)$$

cada equação em (A.31) é da forma

$$\dot{y}_i = \lambda_i y_i \qquad (A.32)$$

onde λ_i é em geral um autovalor complexo de A. Desta forma, um algoritmo de integração multipassos para ser genericamente aplicável deve ser estável para a equação (A.32).

Há diversas razões para escolher (A.28) como a equação de teste padrão:

a) É a equação mais simples geralmente encontrada na prática, um algoritmo que falhe na solução deste problema será inútil.

b) Como é uma equação linear, podem ser obtidos critérios de estabilidade em uma forma explícita.

c) Mesmo nos casos genéricos de sistemas de equações não lineares

$$\dot{x} = f(x)$$

o seu comportamento local em torno de alguns pontos iniciais x_o pode ser aproximado pela equação variacional linear

$$\delta \dot{x} = A \, \delta x \qquad (A.33)$$

onde

$$A \triangleq \frac{\partial f(x)}{\partial x} \Big|_{x = x_o} \qquad (A.34)$$

é a matriz Jacobiana n x n de f(x) avaliada no ponto inicil x_o.

Desta forma, um algoritmo que funcione bem para a equação de teste padrão terá boas possibilidades de funcionar com a maioria das equações não lineares também.

Aplicando a equação genérica dos algoritmos de integração multipassos à equação (A.28) obtem-se

$$x_{n+1} = \sum_{i=0}^{p} a_i \, x_{n-i} + h \sum_{i=-1}^{p} b_i \, (-\lambda \, x_{n-i}) \qquad (A.35)$$

Desta equação far-se-á a análise da estabilidade dos algoritmos multipassos.

Pode-se reescrever a eq. (A.35) na seguinte forma equivalente de equações diferença lineares:

$$(1 + \sigma b_{-1})x_{n+1} - (a_o - \sigma b_o)x_n - (a_1 - \sigma b_1) \, x_{n-1}$$

$$- \ldots - (a_p - \sigma b_p)x_{n-1} = 0 \qquad (A.36)$$

onde

$$\sigma \triangleq h\lambda \qquad (A.37)$$

A solução completa da equação (A.36) tomando sua transformada Z é dada por [1] - p 488 :

$$x_n = C_1 Z_1^n + C_2 Z_2^n + \ldots + C_{p+1} Z_{p+1}^n \qquad (A.37)$$

onde Z_1, Z_2, ..., Z_{p+1} são as **p + 1** raízes da equação polinomial

$$P(Z) \triangleq (1 - \sigma b_{-1})Z^{p+1} - (a_o - \sigma b_o)Z^p - (a_1 - \sigma b_1)Z^{p-1}$$

$$- \ldots - (a_p - \sigma b_p) = 0 \qquad (A.38)$$

desde que as raizes sejam todas distintas.

As constantes arbitrárias c_1, c_2, ..., c_{p+1} são determinadas a partir das condições iniciais x_{n-p}, x_{n-p+1}, ... x_{n-1}, x_n.

Se a eq. (A.38) tiver alguma raiz múltipla $Z = Z_i$ de multiplicidade m_i, então o termo correspondente em (A.37) terá a forma

$$(c_{io} + c_{i1}n + c_{i2}n^2 + \ldots + c_{im_{i-1}} n^{m_{i-1}}) Z_i^n \qquad (A.39)$$

Observa-se que se $|Z_i| = 1$ tal termo fará com que $|x_n| \longrightarrow \infty$ para $n \longrightarrow \infty$. Se $|Z_i| > 1$, então $|x_n| \longrightarrow \infty$ mesmo se Z_i for uma raiz não múltipla.

As constantes arbitrárias c_1, c_2, ..., c_{p+1} são determinadas a partir dos valores iniciais x_{n-p}, x_{n-p+1}, ..., x_{n-1}, x_n.

Proposição :

Uma condição necessária para a estabilidade de um algoritmo multipassos é que todas as raizes de $P(z) = 0$ de magnitude 1 sejam simples.

Teorema

(Critério de estabilidade numérica para Algoritmos de Integração multipassos)

Um algoritmo multipassos consistente que satisfaça a seguinte condição

$$b_{-1} + b_0 + b_1 + \ldots + b_p \neq 0$$

é estável no sentido de que o erro de truncamento e o erro de arredondamento permaneçam limitados para um passo h suficientemente pequeno se, e somente se, todas as raizes da

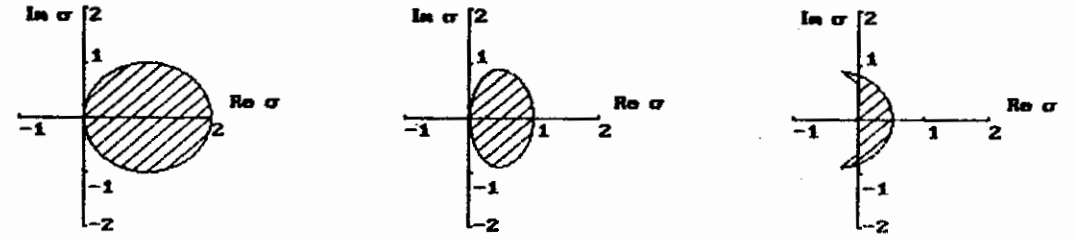
equação polinomial P(z) definidaa em (A.38) estejam estritamente no interior do círculo unitário.

Corolário : Os algoritmos de Adams-Bashforth, Adams-Moulton e Gear são estáveis segundo o teorema da estabilidade numérica.

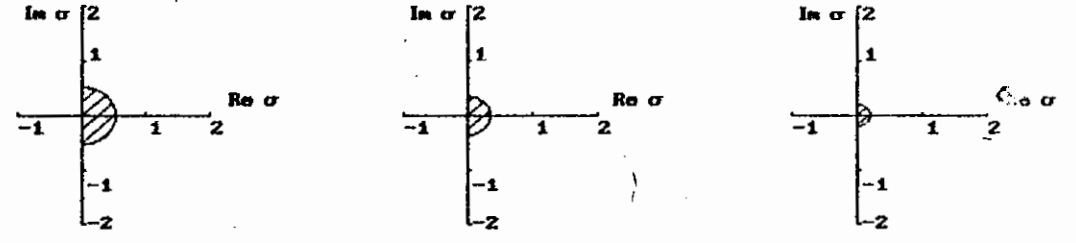
Nas figuras A.3, A.4 e A.5 mostra-se as regiões de estabilidade absoluta destas três famílias de integração numérica para alguns valores de k

A.6 - CONVERGÊNCIA

Um algoritmo multipassos estável apenas garante que os erros local de truncamento e de arredondamento não serão amplificados para um passo h suficientemente pequeno. Com o aumento do passo h, mesmo um algoritmo estável, pode gerar um erro local de truncamento oscilatório, desta forma gerando uma solução numérica oscilatória ("ringing") errônea. O algoritmo além de ser estável deve ser *convergente*. Esta propriedade indesejável é uma das razões pelas quais o algoritmo de Euler regressivo é algumas vezes preferido sobre o algoritmo trapezoidal, apesar da maior precisão deste último.

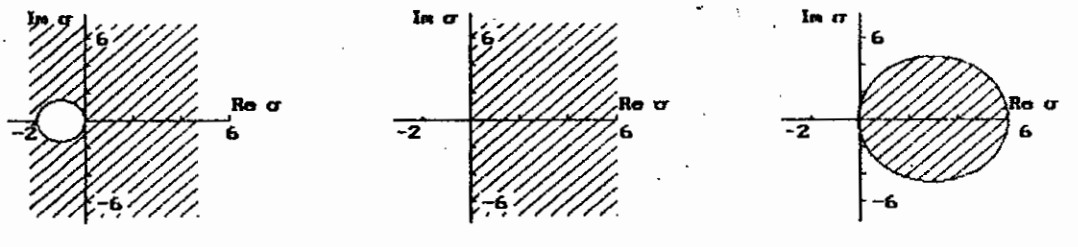


(a) 1a. Ordem (Euler Progressivo) (b) 2a. Ordem (c) 3a. Ordem

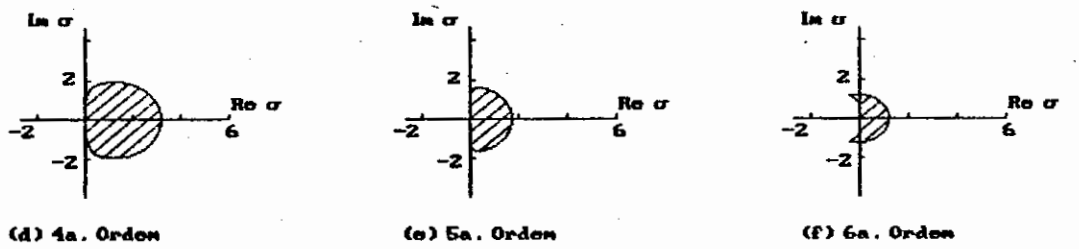


(d) 4a. Ordem (e) 5a. Ordem (f) 6a. Ordem

Fig A.3 Regiões de Estabilidade Absoluta dos Algoritmos de Integração de Adams-Bashforth para k = 1, 2, ... 6



(a) 1a. Ordem (Euler Regressivo) (b) 2a. Ordem (Trapezoidal) (c) 3a. Ordem



(d) 4a. Ordem (e) 5a. Ordem (f) 6a. Ordem

Fig A.4 Regiões de Estabilidade Absoluta dos Algoritmos de Integração de Adams-Moulton para k = 1, 2, ... 6

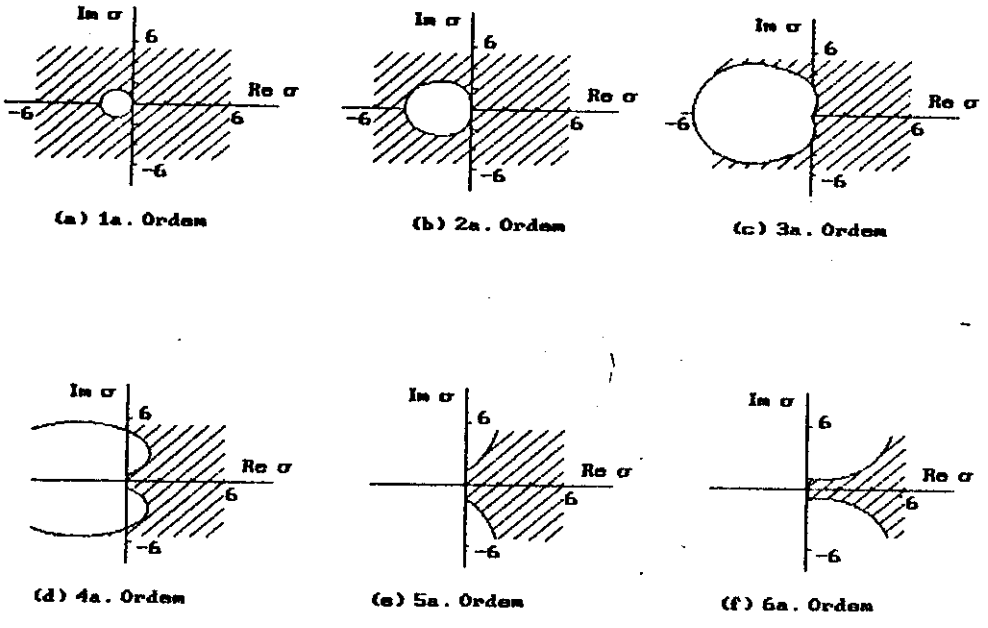


Fig A.5 Regiões de Estabilidade Absoluta dos Algoritmos de Integração de Gear para k = 1, 2, ... 6

APÊNDICE B - EXEMPLOS DE SOLUÇÕES DE SISTEMAS

B.1 - EXEMPLO DE SOLUÇÃO DE UM SISTEMA DE QUARTA ORDEM

Para ilustrar o benefício das técnicas de matrizes esparsas sobre o esforço computacional, apresentar-se-á a solução de dois sistemas, um de matriz densa e outro de matriz esparsa. Inicialmente o considera-se o seguinte sistema, em que a matriz é densa:

$$\begin{bmatrix} 2 & 3 & -1 & -1 \\ 1 & 3 & 2 & 2 \\ 2 & -2 & 4 & 1 \\ 3 & -1 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 3 \\ 8 \end{bmatrix} \qquad (B.1)$$

Utilizando o método da eliminação Gaussiana ou o método da fatoração LU a solução obtida é:

$$x = \begin{bmatrix} 2 & -1 & -1 & 1 \end{bmatrix}^t$$

Em qualquer caso o número exato de operações (multiplicações e divisões) é dado por

$$\frac{n^3}{3} + n^2 - \frac{n}{3} = \frac{4^3}{3} + 4^2 - \frac{4}{3} = 36$$

Considerando-se em seguida a seguinte matriz de coeficientes, contendo apenas 50% de elementos não nulos

$$\begin{bmatrix} 2 & 0 & 0 & 6 \\ 0 & 4 & 0 & 8 \\ 0 & 0 & 2 & 4 \\ 4 & 0 & 0 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \\ 8 \\ 10 \end{bmatrix} \qquad (B.2)$$

Utilizando-se um algoritmo computacional que não leve em consideração a característica esparsa desta matriz (isto é opere sobre elementos nulos) então o número de operações necessárias para obter a solução também seria 36. O tempo que o computador levaria para solucionar (B.1) seria o mesmo tempo para solucionar (B.2) independente do fato de que (B.2) é esparsa. Desta forma observa-se que a esparsidade da matriz de coeficientes é de pouca ajuda se esta é armazenada com técnicas de *matriz densa*, que não aproveitam os elementos nulos.

B.2 - Exemplo do Efeito da Ordenação de Linhas no Esforço Computacional da Solução de Sistemas Esparsos

Conforme visto no apêndice anterior, a característica esparsa da matriz **A** pode reduzir o esforço computacional necessário à solução de **Ax = b**. Ver-se-á agora que a ordem em que as variáveis são eliminadas nos processos baseados na

eliminação gaussiana também desempenha um papel importante no esforço computacional.

Considerando a equação

$$\begin{bmatrix} 2 & 4 & -2 & -6 \\ 3 & 9 & 0 & 0 \\ -2 & 0 & 8 & 0 \\ 2 & 0 & 0 & -12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 6 \\ -6 \\ 14 \\ 26 \end{bmatrix} \qquad (B.3)$$

se o método da eliminação Gaussiana for aplicado diretamente, o passo da eliminação Gaussinana progressiva requer 30 multiplicações e divisões para fornecer

$$\begin{bmatrix} 1 & 2 & -1 & -3 \\ 0 & 1 & 1 & 3 \\ 0 & 0 & 1 & -9 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 3 \\ -5 \\ 20 \\ -2 \end{bmatrix}, \qquad (B.4)$$

o passo da substituição regressiva requer outras 6 operações (assumindo que os dois primeiros elementos da coluna 3 são tratados como quaisquer elementos não nulos e não como + 1 e -1) para fornecer

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 2 \\ -2 \end{bmatrix}. \qquad (B.5)$$

Desta forma foram utilizadas as mesmas 36 operações necessárias para resolver uma *matriz densa*. A esparsidade da matriz A com os elementos nulos distribuidos como em (B.3)

não produziu nenhum benefício em termos de números de multiplicações e divisões.

Por outro lado se a equação (B.3) é escrita em outra ordem, como em seguida:

$$\begin{bmatrix} -12 & 0 & 0 & 2 \\ 0 & 9 & 0 & 3 \\ 0 & 0 & 8 & -2 \\ -6 & 4 & -2 & 2 \end{bmatrix} \begin{bmatrix} x_4 \\ x_2 \\ x_3 \\ x_1 \end{bmatrix} = \begin{bmatrix} 26 \\ -6 \\ 14 \\ 6 \end{bmatrix} \qquad (B.6)$$

Se agora as variáveis são eliminadas na ordem x4, x2 e x3 no passo de eliminação progressiva precisaremos de apenas 13 operações para obter

$$\begin{bmatrix} 1 & 0 & 0 & -1/6 \\ 0 & 1 & 0 & 1/3 \\ 0 & 0 & 1 & -1/4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_4 \\ x_2 \\ x_3 \\ x_1 \end{bmatrix} = \begin{bmatrix} -13/6 \\ -2/3 \\ 7/4 \\ 1 \end{bmatrix} \qquad (B.7)$$

O passo de substituição regressiva requer apenas 3 operações para obter a solução

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_4 \\ x_2 \\ x_3 \\ x_1 \end{bmatrix} = \begin{bmatrix} -2 \\ -1 \\ 2 \\ 1 \end{bmatrix} \qquad (B.8)$$

Desta forma, a simples reordenação da equação (B.3) reduziu o número de operações de 36 para 16. O impacto seria ainda maior no caso de matrizes esparsas de ordem elevada. Desta forma torna-se desejável encontrar a melhor ordem de eliminação no sentido de minimizar o esforço computacional. Este problema é conhecido como *ordenação ótima* [1, 2]. É

claro que não é necessário efetivamente reescrever as equações como foi feito na equação (B.6). O mesmo esforço computacional para reduzir a matriz poderia ser feito sobre a matriz original (B.3) porém escolhendo os pivos na seguinte ordem: a_{44} , $a_{22}^{(2)}$ $a_{33}^{(3)}$, $a_{11}^{(4)}$. Desta forma o problema do ordenamento ótimo pode ser visto como o problema da escolha dos pivos (não necessáriamente restritos aos elementos da diagonal).

B.3 - Matrizes Fator da Matriz Tridiagonal

A estrutura tridiagonal é importante pois a matriz de admitância nodal (e também a matriz de admitância nodal modificada) de redes RLC escada possuem esta forma. Neste apêndice mostra-se a forma da matriz inversa e das matrizes fator para enfatizar a vantagem dos métodos de fatoração em preservar a esparsidade original.

Como exemplo considerar-se-á a matriz tridiagonal 6x6 (x· simboliza elemento não nulo)

$$A = \begin{bmatrix} x & x & & & & \\ x & x & x & & & \\ & x & x & x & & \\ & & x & x & x & \\ & & & x & x & x \\ & & & & x & x \end{bmatrix} \qquad (B.9)$$

É facil mostrar [1] que

$$A^{-1} = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \end{bmatrix} \qquad (B.10)$$

$$L = \begin{bmatrix} x & & & & & \\ x & x & & & & \\ & x & x & & & \\ & & x & x & & \\ & & & x & x & \\ & & & & x & x \end{bmatrix} \qquad (B.11)$$

$$U = \begin{bmatrix} 1 & x & & & & \\ & 1 & x & & & \\ & & 1 & x & & \\ & & & 1 & x & \\ & & & & 1 & x \\ & & & & & 1 \end{bmatrix} \qquad (B.12)$$

Nota-se que A^{-1} é *densa*, enquanto L e U são esparsas. As matrizes L e R do método da bifatoração [4, 15]possuem a mesma quantidade de elementos não nulos revelantes (sem considerar a diagonal unitária). A fatoração LU foi utilizada neste exemplo, para torna-lo mais conciso.

B.4 - Bifatoração de um Sistema de Quarta Ordem

Com o propósito de ilustrar a obtenção das matrizes fator no procedimento de solução de um sistema de equações lineares pelo método da bifatoração apresenta-se aqui o desenvolvimento para um problema de quarta ordem. Considerando a seguinte matriz inicial:

$$A = A^{(0)} = \begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & a_{13}^{(0)} & a_{14}^{(0)} \\ a_{21}^{(0)} & a_{22}^{(0)} & a_{23}^{(0)} & a_{24}^{(0)} \\ a_{31}^{(0)} & a_{32}^{(0)} & a_{33}^{(0)} & a_{34}^{(0)} \\ a_{41}^{(0)} & a_{42}^{(0)} & a_{43}^{(0)} & a_{44}^{(0)} \end{bmatrix} \qquad (B.13)$$

O primeiro passo de redução $A^{(1)} = L^{(1)} A^{(0)} R^{(1)}$ fornece:

$$\begin{bmatrix} L_{11}^{(1)} & & & \\ L_{21}^{(1)} & 1 & & \\ L_{31}^{(1)} & & 1 & \\ L_{41}^{(1)} & & & 1 \end{bmatrix} \begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & a_{13}^{(0)} & a_{14}^{(0)} \\ a_{21}^{(0)} & a_{22}^{(0)} & a_{23}^{(0)} & a_{24}^{(0)} \\ a_{31}^{(0)} & a_{32}^{(0)} & a_{33}^{(0)} & a_{34}^{(0)} \\ a_{41}^{(0)} & a_{42}^{(0)} & a_{43}^{(0)} & a_{44}^{(0)} \end{bmatrix} \times$$

$$\begin{bmatrix} 1 & R_{12}^{(1)} & R_{13}^{(1)} & R_{14}^{(1)} \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & a_{34}^{(1)} \\ 0 & a_{42}^{(1)} & a_{43}^{(1)} & a_{44}^{(1)} \end{bmatrix}$$

(B.14)

onde:

$$L_{11}^{(1)} = \frac{1}{a_{11}^{(0)}}$$

$$L_{i1}^{(1)} = -\frac{a_{i1}^{(0)}}{a_{11}^{(0)}} \quad , \qquad R_{1i}^{(1)} = -\frac{a_{1i}^{(0)}}{a_{11}^{(0)}} \qquad (i = 2,3,4)$$

$$a_{ij}^{(1)} = a_{ij}^{(0)} - \frac{a_{i1}^{(0)}\, a_{1j}^{(0)}}{a_{11}^{(0)}} \qquad (i = 2,3,4, \quad j = 2,3,4)$$

O segundo passo de redução $A^{(2)} = L^{(2)} A^{(1)} R^{(2)}$ fornece:

$$\begin{bmatrix} 1 & 0 & & \\ & L_{22}^{(2)} & & \\ & L_{32}^{(2)} & 1 & \\ & L_{42}^{(2)} & & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & a_{34}^{(1)} \\ 0 & a_{42}^{(1)} & a_{43}^{(1)} & a_{44}^{(1)} \end{bmatrix} \times$$

$$\begin{bmatrix} 1 & & & \\ 0 & 1 & R_{23}^{(2)} & R_{24}^{(2)} \\ & & 1 & \\ & & & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & a_{33}^{(2)} & a_{34}^{(2)} \\ & & a_{43}^{(2)} & a_{44}^{(2)} \end{bmatrix}$$

(B.15)

onde:

$$L_{22}^{(2)} = \frac{1}{a_{22}^{(1)}}$$

$$L_{i2}^{(2)} = -\frac{a_{i2}^{(1)}}{a_{22}^{(1)}}, \qquad R_{2i}^{(2)} = -\frac{a_{2i}^{(1)}}{a_{22}^{(1)}} \qquad (i = 3,4)$$

$$a_{ij}^{(2)} = a_{ij}^{(1)} - \frac{a_{i2}^{(1)} a_{2j}^{(1)}}{a_{22}^{(1)}} \qquad (i = 3,4, \quad j = 3,4)$$

O quarto e ultimo passo de redução $A^{(4)} = U = L^{(4)}A^{(3)}R^{(4)}$ fornece:

$$\begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \\ & & & L_{44}^{(4)} \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & a_{44}^{(1)} \end{bmatrix} \times$$

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

(B.16)

onde:

$$L_{44}^{(4)} = \frac{1}{a_{44}^{(3)}}$$

APÊNDICE C

UTILIZAÇÃO DO PROGRAMA SITECI

Este apêndice apresenta uma descrição a nível de usuário do programa SITECI, Simulador Temporal de Circuitos Lineares. O programa foi escrito utilizando-se a liguagem TURBO-C versão 2.0 da Borland International, em computador pessoal da linha IBM-PC, com MS-DOS 4.01. O SITECI roda com ou sem co-processador matemático. Caso este esteja presente será utilizado, aumentando em muito a velocidade de processamento.

O apêndice C está organizado nas seguintes seções:

- Opções da linha de Comando
- Ajuda
- Padrão do Arquivo de Entrada
- Biblioteca
- Limitações

C.1 - Opções da Linha de Comando

O programa SITECI aceita alguns dados pela linha de comando, que definem as seguintes condições de simulação:

SITECI nome-arq. alg-integração tip:arq-saída

nome-arq :

"caminho" que dá acesso ao arquivo, de acordo com o padrão do sistema operacional MS-DOS.

alg-integração:

Opções: 0, 1, 2, 3, 4.

Este parâmetro define o algoritmo que será utilizado para discretizar o sistema de equações.

0 - Euler Regressivo

1 - Trapezoidal

2 - Gear de segunda ordem

3 - Gear de terceira ordem

4 - Gear de quarta ordem

tip:arq-saída

Opções: "tip:int" , "tip:float".

Caso este argumento seja "tip:int" então os arquivos de sinal de saída serão do tipo "inteiro". Caso este parâmetro esteja ausente ou seja "tip:float", então os os arquivos de sinal de saída serão do tipo "float". Internamente os sinais são sempre armazenados como "float" e os cálculos são todos feitos em "double".

C.2 - AJUDA

Estão disponíveis duas telas de auxílio.

SITECI ou SITECI ajuda

Quando não é fornecido nenhum parâmetro ou apenas "ajuda", então será apresentada uma tela contendo informações de auxílio quanto a forma de utilização do programa.

SITECI tabela

Quando é fornecido o parâmetro tabela, então apresenta-se uma tabela contendo os componentes implementados.

C.3 - PADRÃO DO ARQUIVO DE ENTRADA

Os arquivos de definição de circuito devem ter extensão ".cir" para serem reconhecidos pela função de leitura. Toda linha que inicie por "*" é considerada como sendo de comentário e é simplesmente copiada para o arquivo de saída de circuito ".exp". Na atual implementação este arquivo é sempre criado, e contém a expansão de todos os componentes de biblioteca em componentes básicos.

As variáveis que serão exteriorizadas, isto é, tensões nodais ou correntes de ramos que serão escritas nos arquivos de sinais de saída, são especificadas numa linha iniciada pela letra "A". As tensões de saída são especificadas pela letra "V" seguida pelo número do nó correspondente na descrição de circuito. O arquivo contendo o sinal de saída terá este mesmo nome, acrescido da extensão ".stc". A inclusão de uma corrente de ramo nas variáveis de saída é feita pelo nome do ramo precedido pela letra "I". O arquivo

contendo o sinal de saída terá este mesmo nome acrescido da extensão ".stc".

O seguinte exemplo ilustra as opções:

```
A  V1  V5  V7  IR2  ICLOAD
```

Fig. C.1 - Exemplo da especificação das variáveis a a serem exteriorizadas

Para esta linha de especificação das variáveis de saída serão gerados os seguintes arquivos com sinais de saída:

V1.stc, V5.stc, V7.stc, R2.stc e CLOAD.stc

Onde os três primeiros arquivos conterão as tensões dos nós 1, 5 e 7 em relação ao nó de referência (terra). Os dois últimos arquivos conterão a corrente nos ramos R2 e CLOAD, respectivamente. A ordem em que estes argumentos aparecem não é relevante. Se o circuito em questão não tiver um número de nó de uma tensão especificada ou não houver ramo com um nome solicitado, será apresentada uma mensagem de erro. Em ambos os casos o processamento é interrompido. No caso de correntes de ramos é feita distinção entre letras maiúsculas e minúsculas.

Cada ramo do circuito é descrito por uma linha do arquivo de definição de circuito. De acordo com o tipo de ramo existem outros parâmetros na linha. Os primeiros parâmetros são sempre o nome do ramo, iniciado por uma letra reservada e os nós aos quais o ramo está ligado. O ramo de referência é sempre identificado por Ø. A numeração dos nós do circuito inicia com 1 e deve ser seqüêncial, isto é, se houver "n" nós além do Ø , estes deverão estar numerados de 1

a n. Esta é uma forma mais restritiva do que a utilizada pelo PSPICE, em que os nós podem ter qualquer numeração. Esta restrição permitiu simplificar a representação interna dos nós.

As seguintes letras são aceitas como identificadoras de ramo:

R L C V I F G H E Q A O

Uma linha que inicie com outra letra provocará a interrupção do processamento, com apresentação de mensagem explicativa. Estas letras identificam os seguintes ramos:

R - Resistor linear
L - Indutor linear
C - Capacitor linear
V - Fonte de tensão independente
I - Fonte de corrente independente
F - Fonte de corrente controlada por corrente
G - Fonte de corrente controlada por tensão
H - Fonte de tensão controlda por corrente
E - Fonte de tensão controlda por tensão
Q - Transistor de junção bipolar
A - Especificação das variáveis de saída
O - Amplificador Operacional

Pelas características comuns de especificação os componentes foram agrupados da seguinte forma:

Componentes padrão: R, L e C
Fontes independentes: V e I
Fontes controladas por tensão: G e E
Fontes controladas por corrente: H e F
Transistores e Amplificadores operacionais: Q e O

COMPONENTES PADRÃO : R L C

São descritos por:

X nome nop noc valor

Onde X representa R, L ou C. nop e noc são os nós de partida e chegada em que estes ramos estão ligados. A ordem destes parâmetros especifica a polaridade da corrente no ramo.

O valor destes componentes é especificado através de um número, sem dimensão. Os resistores são lidos com dimensão em ohms, os capacitores em farads e os indutores em henrys. A seguir vêm-se alguns exemplos:

```
R1        1    2    1E3
CR12     12    0    1E - 9
Lmútua    5    6    1E - 6
```

Fig. C.2 - Exemplos da especificação de componentes padrão

A primeira linha no exemplo acima identifica um resistor ligado entre os nós 1 e 2 de valor 1 Kohm. O nome do ramo é "R1". A segunda linha identifica um capacitor ligado do nó 12 e ao nó de referência, de valor 1 nF. O nome do ramo é "CR12". A terceira linha identifica um indutor "Lmutua" de 1μH conectado aos nós 5 e 6.

FONTES INDEPENDENTES: V, I

As fontes independentes são identificadas pela primeira letra "V" para fontes de tensão em "I" para fontes de corrente. Após a primeira letra têm-se o nome do arquivo que contém o sinal da fonte. O nome pode conter o caminho dos diretórios onde está o arquivo de sinal. A extensão do arquivo de sinal deve ser incluída no nome. Abaixo alguns exemplos

```
Vseno.ger          1   0   Ø
Iger\quad.stc      2   3   Ø
Vconst             3   Ø   2.5
```

Fig. C.3.- Exemplos de fontes independentes

Após o nome da fonte têm-se "nop" e "noc", nó de partida e nó de chegada, respectivamente. A ordem destes nós define a polaridade da fonte. O quarto parâmetro quando diferente de zero especifica o valor de uma fonte DC e os valores das amostras que definem a fonte não serão lidos de arquivo. Quando o último parâmetro é "Ø" então os valores das amostras serão lidos de arquivo de sinal. O "cabeçalho" dos arquivos fonte contém as informações de tipo do sinal, número de amostras, comentários, etc. Internamente todas as fontes de sinal são armazenadas como "float" (ponto flutuante). No caso do arquivo ser do tipo "int" (inteiro) ocorre a conversão automática para o tipo "float". Quando existirem mais de uma fonte de sinal o número de amostras de todos os arquivos deve ser igual, caso contrário o processamento é interrompido. A dimensão das fontes é **volt** para as fontes de tensão e **ampere** para as fontes de corrente.

FONTES CONTROLADAS POR CORRENTE: H, F

As fontes controladas por corrente são descritas por:

Xnome nop noc ramo_controle ganho

onde X representa H ou F

"H" inicia a descrição de uma fonte de tensão controlada por corrente e "F" inicia a descrição de uma fonte de corrente controlada por corrente. O parâmetro "ramo_controle"é um inteiro que corresponde ao número do ramo de controle da fonte. Todos os ramos do circuito são numerados sequencialmente a partir de Ø . É preciso cuidado ao inserir novos ramos na descrição de um circuito para que o ramo de controle corresponda a nova posição deste. Abaixo vê-se um exemplo:

```
Vsen.ger   1   0   0
R1         1   2   10e3
H          2   0   1    10
```

Fig. C.4- Exemplo de fonte tensão controlada por corrente

No exemplo da fig. C.4 tem-se a descrição de um circuito com três componentes, entre eles uma fonte de tensão controlada por corrente. Neste caso a numeração dos ramos é : 0 → Vsen, 1 → R1, 2 → G. A terceira linha especifica uma fonte de tensão entre os nós 2 e 0 controlada pela corrente de R1 (ramo 1) o ganho de transresistência desta fonte é dado pelo último parâmetro (10).

FONTES CONTROLADAS POR TENSÃO: E , G

As fontes controladas por tensão são descritas por:

Xnome nop noc noc-1 noc-2 ganho

Onde "E" inicia a descrição de uma fonte de tensão controlada por tensão e "G" inicia a descrição de uma fonte de corrente controlada por tensão. "nop" e "noc" são os ramos em que a fonte está ligada e "noc-1" e "noc-2" são os nós cuja diferença de tensões controla a fonte. O último parâmetro "ganho" fornece o ganho de tensão para fontes de tensão controladas por tensão ("E") ou o ganho de transcondutância para as fontes de corrente controladas por tensão ("G").

```
Vsen.ger   1   0   0
R1         1   2   10e3
Gtrans     2   0   1   2   0.050
```

Fig. C.5- Exemplo de fonte tensão controlada por corrente

No exemplo acima têm-se uma fonte de corrente de nome "Gtrans" conectada aos nós 2 e 0 controlada pela tensão entre os nós 1 e 2, de ganho 0.050.

TRANSISTORES: Q

Qnome noc noe nob modelo tipo

A especificação de transistores têm início com a letra "Q". Os parâmetros "noc", "noe" e "nob" especificam os nó do

coletor, nó do emissor e o nó da base, respectivamente. O parâmetro modelo pode assumir os valores de 0 a 3, correspondendo aos seguintes modelos de transistor

0 : Híbrido simplificado
1 : Híbrido completo
2 : Pi - híbrido em baixa frequência
3 : Pi - híbrido

O parâmetro tipo é uma variável do tipo "string"[1] que identifica um conjunto de valores para os componentes do modelo. Atualmente estão disponíveis dez conjuntos de valores, associados aos seguintes nomes:

BC457, BC458, 2N2222, TIP31, QA741,
QB741, BC247, BC248, QA709, QB709.

O exemplo abaixo ilustra a especificação de um transistor para o SITECI:

*	noc	nob	noe	modelo	tipo
Q1	4	3	2	3	2n2222

Fig. C.6 - Exemplo da especificação de um Transistor

Este exemplo identifica um transistor ligado entre os nós 4, 3, e 2 com modelo pi-híbrido para altas frequências com valores dos parâmetros do transistor 2n2222.

Os modelos e os valores numéricos dos tipos implementados são apresentados no tópico que descreve as funções de biblioteca.

---

[1] "string" - Conjunto de caracteres.

AMPLIFICADOR OPERACIONAL: O

A letra "O" inicia a descrição de um amplificador operacional para o SITECI. Os seguintes parâmetros são necessários para uma descrição completa:

Onome nó_saída não_inv inv modelo tipo

Onde "nó_saída", "não_inv" e "inv" identificam o nó que está ligado à saída. à entrada não inversora e à entrada inversora do amplificador operacional, respectivamente. O parâmetro "modelo" permitiria utilizar os diferentes modelos para o amplificador operacional. Atualmente apenas o modelo linearizado do UA741 apresentado no Capítulo 5, pg. 142 (Fig. 5.15) esta disponível. O parâmetro tipo permitiria indentificar qual conjunto de valores seriam utilizados pelos componentes do modelo. Na atual implementação só estão disponíveis os valores do UA741 linearizado, que serão vistos na seção referente à biblioteca. O exemplo seguinte apresenta a descrição de um amplificador operacional para o SITECI.

```
*    sai   n_inv  inv   modelo   tipo
01    2      7     8      3     QA741
```

Fig. C.8 - Exemplo de especificação de amplificador operacional

A primeira linha é um comentário que ajuda a identificar os nós do amplificador operacional. A segunda linha descreve um amplificador operacional com saída ligada ao nó 2 e entradas inversora e não inversora ligadas aos nós 8 e 7. Os valores dos parâmetros são os do amplificador operacional μA741.

C.4 - BIBLIOTECA

O módulo de Biblioteca visa simplificar a descrição de dispositivos no arquivo de descrição de circuito. Um amplificador operacional é representado linearmente por diversos nós e ramos internos. A topologia interna e o valor dos componentes são sempre os mesmos para um dado operacional. Se não houvesse o módulo de biblioteca, cada amplificador operacional precisaria ser descrito por seus componentes constitutivos, num processo tedioso e sujeito a erros de digitação. Com o módulo de biblioteca entram-se apenas os nós externos do dispositivo e a função de biblioteca o expande em termos dos componentes básicos (cuja estampa está definida).

Os dispositivos que foram implementados no módulo biblioteca são:

Transistor Bipolar (Q)
Amplificador Operacional (O)

O módulo de biblioteca do simulador temporal de circuitos não teve como objetivo ser completo ou ter os valores reais dos componentes. Isto é, apenas alguns dispositivos foram incluídos na biblioteca com a finalidade de testar os algoritmos desenvolvidos. Dez tipos de transistores e um amplificador operacional foram implementados. A expansão dos tipos de transistores pode ser feita acrescentando-se novos valores de parâmetros ao arquivo texto "Biblio.DEF".

O algoritmo de Biblioteca utiliza uma lista dinâmica de transistores e amplificadores operacionais fornecida pela

função de leitura (fig. 5.11, pg. 134). Esta lista é percorrida e os componentes são expandidos de acordo com o modelo utilizando-se os valores do tipo.

Após o nome do transistor e os nós dos três terminais tem-se o modelo do transistor. De acordo com o modelo são acrescentados ramos às listas padrão e às listas de fontes controladas. No caso dos modelos 1 a 3 existe a criação de um nó interno.

Estão disponíveis quatro modelos para o transistor bipolar:

modelo Ø : híbrido simplificado

Apenas dois componentes são utilizados para modelar o transistor, "hie" e "hfe", conforme visto na fig. 5.14-a, pg. 140. Na fig. C.10 seguinte apresenta-se um exemplo de descrição de um transistor pelo modelo híbrido simplificado.

```
*      noC     noB     noE
Q1     1       2       3       Ø       BC457
```

Fig. C.10 - Exemplo de Transistor - modelo híbrido simplificado

Esta descrição do transistor seria expandida pela função biblioteca em um resistor de 1.1 KΩ entre os nós 2 e 3 e uma fonte de corrente controlada por corrente enter os nós 1 e 3 de ganho 100.

modelo 1 : híbrido completo

Para o modelo híbrido completo são utilizados quatro componentes básicos para modelar o transistor, "hie", "hre", "hfe" e "hoe", conforme visto na fig. 5.14-b, pg. 140.

modelo 2 : pi-híbrido em baixa freqüência

Para o modelo pi-híbrido em baixa freqüência são utilizados cinco componentes básicos para modelar o transistor, "rbbl", "rble", "rblc", "rce" e "gm", conforme visto na fig. 5.14-c, pg. 140.

modelo 3 : pi-híbrido completo

Para o modelo pi-híbrido completo são utilizados sete componentes básicos para modelar o transistor, "rbbl", "rble", "rblc", "rce", "Cc", "Ce" e "gm", conforme visto na fig. 5.14-d, pg. 140.

Os valores dos componentes básicos utilizados na expansão dos modelos do transistor foram obtidos de [27] como valores típicos:

```
hie  = 1100
hfe  = 100
hre  = 1e-4
hoe  = 100
rbbl = 100
rble = 1000
rblc = 4e6
rce  = 80e3
gm   = 0.05
Ce   = 3e-12
Cc   = 12e-12
```

Todos os transistores têm os componentes básicos correspondentes expandidos com os mesmos nomes no arquivo ".exp". Isto não causa maiores problemas pois não é possível solicitar a corrente de um ramo interno a um transistor como variável exteriorizada.

Além dos transistores bipolares foi implementado um modelo para o amplificador operacional. Este modelo é o mesmo utilizado pelo PSPICE para o μA741, com algumas simplificações. Do modelo original utilizado no PSPICE foram retirados os diodos, componentes não-lineares não disponíveis no SITECI. O modelo linearizado do amplificador operacional pode ser visto na fig. 5.15, pg. 142. Os valores dos componentes básicos utilizados na expansão do amplificador operacional foram obtidos da biblioteca do PSPICE [5], versão estudante:

```
rbb1_741   =  100
gm1_741    =  91.66667
gm2_741    =  117.8571
rble_741   =  1e3
rblc_741   =  4e6
rce_741    =  80e3
Cc_741     =  3e-12
Ce_741     =  100e-12
cop1_741   =  5.459553e-12
iee_741    =  1.666000e-5
ropC_741   =  5.305165e3
ropEm_741  =  2.151297e3
ropE_741   =  1.200480e7
copE_741   =  3e-12
gopGM_741  =  5.960075e-9
gopA_741   =  1.884955e-4
rop2_741   =  1e-5
cop2_741   =  3e-11
gopB_741   =  2.357851e2
ropO1_741  =  30
ropO2_741  =  45
```

É interessante notar neste modelo o capacitor cop2_741 = 30pF que fornece a compensação interna do μA741, estabelecendo o produto ganho-banda passante em 1 MHz.

O modelo do μA741 após expandido cria 9 nós internos e 28 ramos. A numeração dos nós e ramos é feita automaticamente pela rotina de expansão de biblioteca. Em um circuito com, por exemplo, 5 operacionais pode-se ver a simplicidade e a clareza que a função de biblioteca fornece. Em vez de 5 x 28 = 140 ramos, entram-se com apenas 5 ramos!

O modelo do μA741 considera que o operacional está corretamente polarizado e que os transistores diferenciais de entrada estão operando em sua faixa linear. Esta simplificação é bastante razoável para a maioria dos casos de interesse prático.

Outros tipos de amplificadores operacionais podem ser simulados acrescentando-se valores à tabela de componentes do arquivo de definição da biblioteca "BIBLIO.DEF". Na atual implemetação há necessidade de recompilar o programa para que os novos componentes sejam reconhecidos.

C.5 - LIMITAÇÕES

O programa SITECI, em sua atual implementação não adapta a ordem do algoritmo de integração, sendo deixado a cargo do usuário a avaliação dos resultados fornecidos pelo simulador. Em uma simulação típica a instabilidade de um algoritmo de integração pode ser verificada facilmente quando os sinais de saída oscilam com amplitude crescente, levando inclusive algumas vezes à erro de "overflow". Neste caso deve ser utilizado um algoritmo de menor ordem, ou no caso dos sinais de entrada serem gerados pelo usuário, utilizar-se uma freqüência de amostragem mais elevada. A ausência de uma estimativa do erro na atual implementação, torna necessário a busca de uma taxa de amostragem adequada através de um processo empírico.

Alguns outros limites estabelecidos nesta implementação do SITECI são:

Número máximo de ramos = 800
Máxima ordem da matriz nodal modificada = 400
Extensão padrão dos arquivos de saída = ".stc"
Número máximo de variáveis de saída = 16
Número máximo de sinais de entrada = 16
Tamanho do bloco para processamento = 128 amostras

A maioria dos dados são armazenados internamente em estruturas de dados alocadas dinamicamente, tornando possível utilizar toda a memória principal disponível.