**Relatório do programa AlyTSPICE – Simulador de Circuitos**

Matheus Silva de Lima

&

高詰ありさデオリベイラ

*Disciplina de Circuitos Elétricos II – Trabalho 2017.2*

***Resumo* - Este relatório tem por objetivo demonstrar o programa desenvolvido durante as aulas da disciplina de Circuitos Elétricos II, ministradas na Universidade Federal do Rio de Janeiro.**

**I** – **Introdução**

Durante a disciplina de Circuitos Elétricos II, estuda-se a fundo, dentre outras coisas, diversos métodos de análise de circuitos, incluindo a modelagem para resolução dos mesmos em sistemas de computação, operando sobre suas formas matriciais. Desta forma, o programa desenvolvido durante a disciplina aplica as técnicas de modelagem de circuitos no domínio do tempo para simulação de elementos lineares, reativos, não lineares e outros componentes ideais, utilizando o método de integração por trapézios, o método de convergência para elementos não lineares de Newton-Raphson, a partir de uma análise de ponto de operação e aplicando a técnica de *gminsteping*, e algoritmo de resolução de sistemas lineares de Gauss-Jordan com Condensação Pivotal.

**II** – **Aspecto Geral**

Neste contexto, este trabalho tem por objetivo o desenvolvimento de um programa, em linguagem C++, que seja capaz de ler um *netlist* gerado pelo programa *Edfilw*(1), analisa-lo e simula-lo, gerando por fim um arquivo de extensão *.tab* que possa ser plotado, por exemplo, pelo programa *mnae(2)* de simulação de circuitos, ou por um programa nativo à interface.

Os elementos que são aceitos pelo programa desenvolvido são como se seguem:

* Resistor, Capacitor e Indutor.
* Fontes de Tensão e Corrente Independentes, dos tipos DC, PULSO & SENO.
* Todas as Fontes Dependentes.
* Amplificador Operacional Ideal (quatro terminais), Transformador Ideal.
* Chaves Resistivas & Resistor Linear por Partes.

**III** – **Desenvolvimento do Programa**

A primeira etapa do desenvolvimento do trabalho consistiu na implementação da leitura correta do netlist,

Em seguida, o desenvolvimento seguiu com a implementação das fontes dependentes, independentes, resistores, amplificadores operacionais e transformador ideal. Toda esta parte inicial foi baseada no programa exemplo mna1.net, disponibilizado na página oficial da disciplina. Como esses elementos, com exceção das fontes independentes, possuem *estampas* invariantes no tempo, a solução no tempo dada pelo programa nesses casos é exata, a menos de erros numéricos.

Com as fontes implementadas, segue-se para o código dos elementos reativos, capacitores e indutores. Para os mesmos, foi utilizada a aproximação integral dos trapézios, como especificado. Desta forma, as estampas desses elementos dependem não apenas do valor dos elementos, mas também do “passo de integração”, este lido diretamente do arquivo *netlist.*

|  |
| --- |
|  |
| *Figura 1 - Modelagem dos Capacitores e Indutores* |

Em conjunto com os capacitores e indutores, cria-se as funções responsáveis pela análise do ponto de operação, e por atualizar as memórias de tensão nos capacitores e corrente nos indutores. Para a análise do ponto de operação, considera-se os capacitores como resistores de valor alto (108 Ω) e os indutores como resistores de valores baixos (10-8 Ω), e fazemos uma simples de circuito resistivo.

Como o decorrer do trabalho, foram programados os elementos não lineares utilizando a o algoritmo de Newton-Raphson. As estampas desses elementos variam de acordo com o a tensão calculada na interação anterior do algoritmo. Em especial, foi feito um tratamento no resistor linear por partes para o caso da resistência ser zero (G0->0), onde atribuímos apenas uma constante de valor alto. O Algoritmo implementado considera que o mesmo convergiu quando a variação de tensão em todos os nós do circuito é menor que um dado valor especificado, a partir do qual a análise no tempo pode prosseguir como normalmente. Caso o algoritmo não convirja em um dado número de passos de interação, é iniciado o procedimento de *gminstteping*.

O procedimento de *gminsteping* consiste em adicionar um resistor de resistência baixa em paralelo com cada nó que não esteja convergindo, de maneira a aumentar a linearidade do sistema e ajudar a forçar o Newton-Raphson a convergir. Caso o Newton-Raphson convirja, se aumenta o valor desta resistência gradativamente (multiplicando-a por fatores de 10), até ela assumir um valor tão alto quanto suficiente para podermos despreza-la, ponto este em que o algoritmo converge. Caso o algoritmo não convirja, diminui-se novamente o valor do resistor, e aumentamo-lo a passos menores. Vale notar que além deste resistor, também adicionamos uma fonte de corrente em paralelo com estes nós, de maneira que forçamos com que a reta equivalente resistor-fonte não necessariamente convirja apenas próximo do zero, generalizando a solução para uma variedade maior de resistores não lineares.

Durante o desenvolvimento do trabalho, desenvolvemos também um script em linguagem Python capaz de plotar os gráficos das varáveis ao final da análise.

**IV** – **Resultados & Exemplos**

A seguir, estão listados resultados de algumas simulações, tanto de circuitos próprios, quanto de exemplos disponibilizados no site da disciplina:

|  |
| --- |
|  |
|  |
| Figura 2 - Filtro Sallen-Key Passa-Baixa Em destaque: Entrada & Saída |

|  |
| --- |
|  |
|  |
| Figura 3 - Dobrador de Tensão com Carga na Saída Em destaque: Entrada & Saída, com Ripple |

|  |
| --- |
|  |
|  |
| Figura 4 - Oscilador Astável com modelo de Ebber-Molls Em Destaque: Tensão de Saída de oscilação |

|  |
| --- |
|  |
|  |
| Figura 5 - Amplificador com transistor NPN, em modelo de Ebber-Molls, configuração emissor comum  Em destaque, sinais de Entrada & Saída |

|  |
| --- |
|  |
|  |
| Figura 6 – Fonte de alimentação Simétrica Em destaque: Sinais de alimentação Positiva e negativa, com Rípple |

**V – Conclusões**

O simulador desenvolvido apresenta convergência correta nos circuitos. O trabalho cumpre a proposta de desenvolvimento do simulador, e aplica o conhecimento adquirido durante as aula ministradas da disciplina.