Circuitos Elétricos II – 1º semestre de 2015 - Trabalho

Prof. Antônio Carlos Moreirão de Queiroz

Fazer um programa que analise circuitos lineares invariantes no tempo, encontrando respostas em transformada de Laplace por interpolação.

O programa deverá analisar um circuito composto pelos elementos:

- Fontes de corrente e de tensão independentes.
- Resistores, capacitores, indutores e transformadores.
- As quatro fontes controladas.
- Amplificadores operacionais ideais, de 4 terminais.

O programa deve ler um netlist, e realizar uma série de análises nodais modificadas em transformada de Laplace, para valores numéricos de "s". A partir dos valores obtidos para as variáveis $E_i(s)$ e para o determinante da matriz $[Y_n(s)]$, deve interpolar polinômios de forma a encontrar as transformadas de Laplace das tensões nodais e correntes acrescentadas.
As fontes podem ser em impulso ou em degrau. Capacitores e indutores podem ter condições iniciais, que geram outras fontes em impulso.

As saídas geradas devem ser salvas em arquivos, de forma a que possam ser processadas por algum outro programa que plote respostas em frequência ou inverta as transformadas para obter resposta no tempo.
É conveniente deixar os polinômios com coeficiente de grau mais alto 1, colocando constantes multiplicando os polinômios. A constante multiplicando o denominador deve ser ajustada para 1, com apenas os numeradores ficando com constantes diferentes de 1.
O formato dos arquivos de saída, que contém polinômios, é:
Primeira linha: Grau
Linhas seguintes, coeficientes, começando pelo de grau 0.
Penúltima linha: Constante multiplicando o polinômio.
Última linha: Fator de normalização em frequência.
Os nomes dos arquivos devem ser:
<nome>.d para denominador.
<nome>.n<nó> para numeradores. <nó> um valor entre 0 e 99.

Os programas laplacemna e lapelim, de trabalhos passados, fazem a mesma análise por outros métodos, e podem ser usados para comparação.

http://www.coe.ufrj.br/~acmq/cursos/laplacemna.zip
http://www.coe.ufrj.br/~acmq/cursos/lapelim.zip

O programa pode ser baseado no programa exemplo MNA1, que implementa a análise nodal modificada para um circuito resistivo linear. Uma versão gráfica (para o Borland C++ builder 6) está também disponível:

http://www.coe.ufrj.br/~acmq/cursos/mna1.zip
http://www.coe.ufrj.br/~acmq/cursos/mna1gr.zip

Formato para o netlist:
O netlist pode ser gerado pelo programa Edfil. Veja em http://www.coe.ufrj.br/~acmq/programs.

Primeira linha: Comentário, ignorar (o editor Edfil coloca o número de nós nesta linha).
Linhas seguintes: Descrição do circuito, com um elemento por linha. A primeira letra determina o tipo de elemento.

Resistor: R<nome> <nó1> <nó2> <Resistência>
Indutor: L<nome> <nó1> <nó2> <Indutância> [IC=<corrente inicial>]
Acoplamento entre indutores: K<nome> <La> <Lb> <k> (La e Lb já declarados.)
Capacitor: C<nome> <nó1> <nó2> <Capacitância> [IC=<tensão inicial>]
Fonte de tensão controlada a tensão: E<nome> <nóV+> <nóV-> <nóv+> <nóv-> <Av>

Atualizado em 17/4/2015

Fonte de corrente controlada a corrente: F<nome> <nóI+> <nóI-> <nói+> <nói-> <Ai>
Fonte de corrente controlada a tensão: G<nome> <nóI+> <nóI-> <nóv+> <nóv-> <Gm>
Fonte de tensão controlada a corrente: H<nome> <nóV+> <nóV-> <nói+> <nói-> <Rm>
Fonte de corrente: I<nome> <nó+> <nó-> <parâmetros>
Fonte de tensão: V<nome> <nó+> <nó-> <parâmetros>
Amplificador operacional ideal: O<nome> <nó saída+> <nó saída-> <nó entrada+> <nó entrada->
Comentário: *<comentário>

(Notar que <xxx> significa colocar o valor xxx sem <>. [xxx] significa opcional.)

Os parâmetros para as fontes devem ser:
<parâmetros> = <valor> <IMPULSO ou DEGRAU>
Note que a presença do degrau vai exigir um tratamento especial, pois aumenta o grau do denominador das transformadas. Pode ser necessário evitar análise em s=0.

Os programas exemplo permitem nomes nos nós. O programa feito pode continuar permitindo usando o mesmo algoritmo (código no programa MNA1), ou admitir apenas números. Neste caso a primeira linha gerada pelo editor Edfil pode ser usada.

As direções para fontes são de acordo com a ordem dos nós e as direções convencionais associadas, sendo o primeiro nó o positivo. Os ramos de entrada de fontes controladas a corrente são curto-circuitos.

O programa deve ler as instruções de como tratar o netlist de uma linha de comando no próprio netlist, no formato abaixo. Não deve ser necessário fornecer qualquer outro parâmetro ao programa além do arquivo de entrada.

.INT <modo> <raio> <normalização>

O modo pode ser LIN ou CIR. Se for LIN, os valores de "s" devem ser reais, distribuídos uniformemente entre ±raio. Se o modo for CIR, os valores são complexos, distribuídos uniformemente sobre um círculo com o raio dado. É interessante usar um número par de pontos, em pares complexos conjugados. Assim basta uma análise por cada par, pois os resultados são também complexos conjugados. Para distribuição linear também, para evitar análise em s=0. O número de valores de "s" a usar deve ser, em princípio, igual ao número de elementos reativos no circuito mais um, ou mais dois para ser par. A normalização é um fator que deve multiplicar capacitâncias e indutâncias, de forma a normalizar o circuito em frequência para melhorar a precisão numérica do cálculo e permitir uso de raio=1 ou próximo. É interessante examinar os polinômios obtidos, e ajustar seu grau se os coeficientes de ordem mais alta obtidos forem desprezíveis.

Opcionalmente, o próprio programa pode plotar gráficos de resposta em frequência (fácil) ou voltar as transformadas para o tempo (difícil).
É interessante incluir também outros componentes, como um transistor bipolar com modelo de pequenos sinais:
No netlist: Q<nome> <coletor> <base> <emissor> <hfe> <hie> <hre> <hoe> [<Cbe> <Cbc>]

O programa deve ser escrito preferencialmente em uma linguagem compilada como C, C++ ou Pascal. O programa deve preferencialmente rodar em ambiente gráfico Windows, 32 bits. Um arquivo .zip com tudo o que for necessário para o programa, inclusive fontes, arquivo executável, documentação, bibliotecas (dll) e exemplos não deve ter mais de 5 Mbytes. Evite sistemas de desenvolvimento que requeiram extensas bibliotecas instaladas.

Sugere-se partir do programa exemplo MNA1, que já tem o algorítmo completo da análise nodal modificada, implementar a análise com variáveis complexas, implementar a interpolação e a geração dos arquivos de saída, e então implementar os componentes reativos, sempre testando.

Grupos de 3 alunos, no máximo. O programa deverá ser apresentado e demonstrado, completamente funcional, por todo o grupo, e entregue com um relatório com comentários e exemplos significativos e originais verificados, até (entenda-se antes de) duas semanas antes da segunda prova. Programas incompletos ou incorretos serão devolvidos para correção até que estejam corretos.

Note-se que o trabalho é bastante extenso, e deve ser começado imediatamente.

Atualizado em 17/4/2015