Circuitos Elétricos II – 1º semestre de 2018 - Trabalho

Prof. Antônio Carlos Moreirão de Queiroz

Escrever um programa que analise circuitos no domínio do tempo, contendo elementos lineares e não lineares, usando análise nodal compacta usando modelos baseados em amplificadores operacionais e o "método θ" de integração junto com o método de Newton-Raphson.

O programa deverá analisar um circuito composto pelos elementos:

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

O programa deve ler um "netlist" descrevendo o circuito, inicializar a análise no domínio do tempo com cálculo de ponto de operação, e então fazer a análise com o tempo total e o tamanho do passo, fixo, dados. Os resultados devem ser salvos em uma tabela em arquivo, de forma que possam ser lidos por outro programa que plote as curvas, como o MNAE. Uma linha de comando no "netlist" define os parâmetros necessários. A primeira linha desta tabela deve listar os nomes de todas as variáveis na tabela, iniciando pelo tempo "t", com correntes citadas como "j" seguidas do nome do elemento onde estão. Ex: t 1 2 3 jH1 jF2.

O programa pode ser baseado no programa exemplo MNA1AMP, que implementa a análise pedida para um circuito resistivo linear. Uma versão gráfica (para o Borland C++ builder 6) está também disponível:

http://www.coe.ufrj.br/~acmq/cursos/mna1amp.zip
http://www.coe.ufrj.br/~acmq/cursos/mna1ampgr.zip

Formato para o "netlist":
O "netlist" pode ser gerado pelo programa EDFIL, a partir do diagrama esquemático. Veja os programas em http://www.coe.ufrj.br/~acmq/cursos

Primeira linha: Comentário (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>
Capacitor: C<nome> <nó1> <nó2> <Capacitância>
Fonte de tensão controlada a tensão: E<nome> <nóV+> <nóV-> <nóv+> <nóv-> <A_v>
Fonte de corrente controlada a corrente: F<nome> <nóI+> <nóI-> <nói+> <nói-> <A_i>
Fonte de corrente controlada a tensão: G<nome> <nóI+> <nóI-> <nóv+> <nóv-> <G_m>
Fonte de tensão controlada a corrente: H<nome> <nóV+> <nóV-> <nói+> <nói-> <R_m>
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->
Diodo: D<nome> <nó+> <nó-> <I_s> <nV_T>
Transistor bipolar: Q<nome> <nóc> <nób> <nóe> <tipo> <α> <α_r> <I_sbe> <nV_Tbe> <I_sbc> <nV_Tbc>
Transformador ideal: K<nome> <nó a> <nó b> <nó c> <nó d> <n>
Comentário: *<comentário>

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

Os parâmetros para as fontes são de acordo com o formato do SPICE, como implementado no programa MNAE. Devem ser suportadas fontes contínuas, senoidais e em pulsos.

Atualizado em 27/4/2018

Fonte contínua: DC <valor>
Fonte senoidal: SIN <nível contínuo> <amplitude> <frequência em Hz> <atraso> <amortecimento> <defasagem em graus> <número de ciclos>
Fonte pulsada: PULSE <amplitude 1> <amplitude 2> <atraso> <tempo de subida> <tempo de descida> <tempo ligada> <período> <número de ciclos>

A fonte senoidal vale:

$$x(t) = A_0 + Ae^{-\alpha(t-t_a)}\,\mathrm{sen}\left(2\pi f\,(t-t_a) + \frac{\pi}{180}\varphi\right)$$

onde A_0 é o nível contínuo, A a amplitude, f a frequência, t_a o atraso, α o amortecimento e φ a defasagem. Antes de $t = t_a$ ou após o número de ciclos, tem o valor inicial ou final respectivamente, de forma a não criar descontinuidades. (Há casos em que é mais útil que volte ao valor contínuo. O programa pode permitir esse modo de operação.) A fonte pulsada começa na amplitude 1, e fica aí até o fim do tempo de atraso. Então muda para a amplitude 2 variando linearmente dentro do tempo de subida, fica na amplitude 2 durante o tempo ligada, volta à amplitude 1 dentro do tempo de descida, e repete tudo com o período e o número de ciclos especificados. Termina na amplitude 1. Os tempos de subida e de descida podem ser nulos. O programa pode usar o tempo do passo então.

As correntes nos indutores devem ser calculadas. As nos capacitores não, mas são necessárias internamente no tratamento do método θ, sendo calculadas ao fim de cada passo de tempo (ou no início do passo seguinte antes da montagem das estampas dos capacitores). Opcionalmente podem ser colocadas nas tabelas também.

O transformador ideal deve implementar $v_{cd} = nv_{ab}$ e $j_{ab} = -nj_{cd}$. A corrente j_{cd} deve ser calculada. O transformador é sempre ideal, funcionando mesmo com sinais contínuos, inclusive na análise do ponto de operação. Opcionalmente, o transformador real pode também ser implementado, como no programa MNAE.

Os diodos são exponenciais, com a curva $j = I_s(e^{v/(nV_T)} - 1)$. Não tem capacitâncias.
Os transistores bipolares usam o modelo de Ebers-Moll com fontes de corrente controladas pelas correntes nos diodos. Não tem efeito Early nem capacitâncias. A implementação de todos os diodos deve estender as curvas como retas acima de 0.7 V. É interessante que haja forma de alterar este limite no programa.

O método de integração é o "método θ", que permite escolher continuamente entre os métodos "backward" (θ=1) e "forward" de Euler (θ=0), passando pelo método dos trapézios (θ=0.5) (Se for usado θ=0, melhor limitar a um valor baixo como 0.001 para continuar tendo modelos com resistor e fonte para capacitores e indutores em todos os casos):

$$y(t_0 + \Delta t) = y(t_0) + \int_{t_0}^{t_0+\Delta t} x(t)dt$$

$$y(t_0 + \Delta t) \approx y(t_0) + \Delta t(\theta x(t_0 + \Delta t) + (1-\theta)x(t_0))$$

O programa exemplo permite nomes nos nós. O programa feito pode continuar permitindo usando o mesmo algoritmo (código no programa MNA1AMP), embora os "netlists" criados no editor EDFIL tenham apenas números.

As direções para fontes são de acordo com a ordem dos nós e as direções convencionais associadas, sendo o primeiro nó o positivo.

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, embora o programa possa ter outros meios de configuração, fora dos parâmetros normais, por exemplo para "debug".

.TRAN <tempo final> <passo> TETA <θ> <passos por ponto na tabela>

No método de Newton-Raphson, caso não ocorra convergência em um número razoável de iterações (20-50), use a técnica de estimar uma nova solução com valores randômicos para as variáveis que não convergem. Conte quantas vezes o ciclo de Newton-Raphson é executado para determinar se a randomização é necessária, e quantas vezes a randomização foi usada, desistindo por não convergência se este limite for atingido. Como sugestão, durante a comparação da solução nova $\mathbf{e}(t_0+\Delta t)$ com a anterior $\mathbf{e}(t_0)$, compare também com $\mathbf{e}(t_0-\Delta t)$ para ver se a solução está oscilando entre dois valores, o que determina que uma randomização é necessária antes do fim da contagem máxima de ciclos.

Opcionalmente, o próprio programa pode plotar seus gráficos. O programa MNAE pode ser usado para plotar os gráficos a partir das tabelas. Este programa faz a mesma análise nos casos dos métodos "backward" e trapézios, mas com análise nodal modificada. Uma versão como pedido será preparada em breve.
http://www.coe.ufrj.br/~acmq/programs/mnae.zip

O programa deve ser escrito preferencialmente em uma linguagem compilada como C, C++ ou Pascal. O programa deve preferencialmente rodar em ambiente gráfico Windows. Evite usar apenas uma interface de console. Um arquivo .zip com tudo o que for necessário para o programa, inclusive fontes, arquivo executável, documentação, bibliotecas e exemplos não deve ter mais de 4 Mbytes. Evite sistemas de desenvolvimento que requeiram extensas bibliotecas instaladas.

Sugere-se partir do programa exemplo MNA1AMP, que já tem o algoritmo completo da análise compactada em C, implementar a análise no tempo com as fontes de sinal, e então implementar os elementos reativos. Por fim implementar os não lineares com o ciclo de Newton-Raphson.

Grupos de 3 alunos, no máximo. O programa deverá ser apresentado e demonstrado, completamente funcional, por todo o grupo, e entregue com um relatório (em arquivo, não impresso) com comentários e exemplos significativos e originais verificados, até (entenda-se antes de) duas semanas antes da segunda prova.

Note-se que o trabalho é bastante extenso, e deve ser começado imediatamente. Trabalhos incompletos serão devolvidos para serem completados, só sendo aceitos completos.