Universidade Federal do Rio de Janeiro
Escola Politécnica
Engenharia Eletrônica e de Computação
Circuitos Elétricos II
Alunos: Marcelle de Souza Campos
Pedro Angelo Medeiros Fonini

Programa de análise de circuitos no tempo para estudo dos Métodos de Gear

Nosso trabalho foi desenvolvido na linguagem C++, sendo ele composto por 4 arquivos: `Makefile`, `trabalhoCircuitosEletricos.cpp`, `myFuntions.cpp` e `circuitAnalysis.h`. O arquivo `trabalhoCircuitosEletricos.cpp` contém a rotina principal do programa; os outros são auxiliares, contendo definições e implementações das funções.

Analisamos o circuito da seguinte forma:

1-) O conteúdo do arquivo com a netlist é lido para a memória, e todos os dados são guardados numa lista de elementos, uma instância da classe elementsList, que guarda os elementos indexados pelo nome.

2-) Fazemos uma análise inicial com ordem de GEAR igual a 1 e passo igual ao passo especificado dividido por 1E+9 para descobrir a solução em t=0. O "chute" inicial que fazemos para o método de Newton-Raphson é tomar todas as tensões como sendo zero. Os estados dos elementos reativos (capacitores e indutores) são os especificados na netlist, ou então zero caso não haja condições iniciais. Esta solução encontrada para o instante t=0 fica sendo também, caso seja necessário, a solução para t<0.

3-) Para cada instante de tempo, fazemos novamente o método de newton-raphson, mas desta vem com chute inicial igual às tensões no instante anterior. Dessa maneira, o método converge mais rapidamente, e evita que sistemas com mais de uma solução apresentem "pulos" inesperados.

Observações:

- Na presença de cada curto, fonte de tensão, amplificador de tensão, transresistor ou indutor, uma variável j (de corrente) a mais é acrescentada às matrizes da análise modificada;

- Na presença de indutores e capacitores, para que ocorra a montagem de estampas, a função gearMethod é chamada. Esta função terá como responsabilidade apresentar o modelo destes elementos, que será constituido por uma resistência e uma fonte de corrente/tensão, de acordo com as respectivas especificações fornecidas ao programa, tais como a ordem do Método de Gear e o passo interno e o passo; Usamos fontes de tensão para o indutor (com o objetivo de medir a corrente que passa nele) e fontes de corrente para o capacitor.

6-) Terminado este processo, o arquivo finalmente será fechado e estará pronto para analise pelo usuário. Ele pode ser executado diretamente pelo MATLAB para gerar variáveis de nomes `t`, `e1`, `e2`, `e3`, ..., `j`<nome1>, `j`<nome2>, etc. Elas podem ser plotadas, por exemplo, com

```
plot(t,e1, t,e2, t,jVcc);
```

Observações:

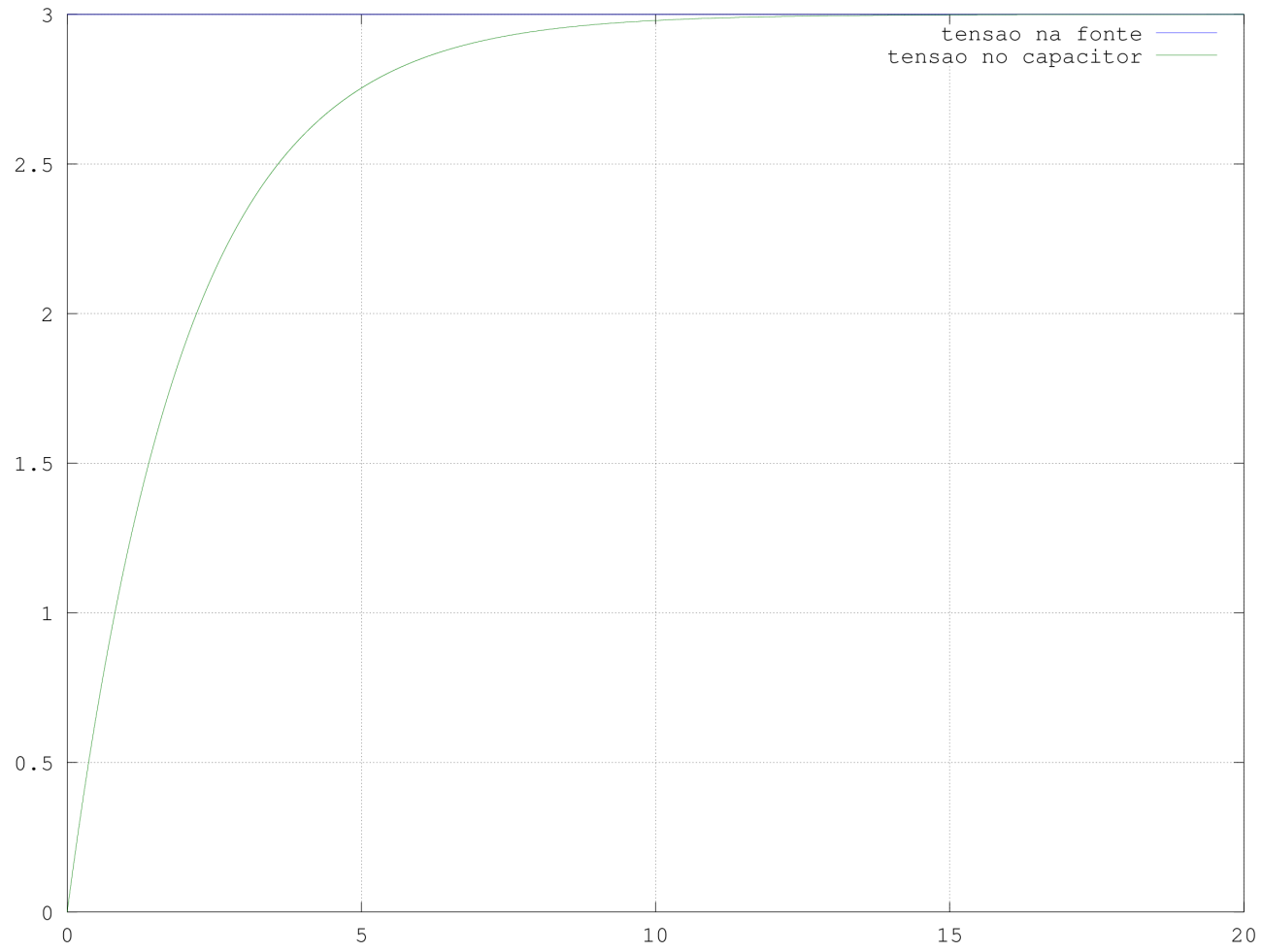
- Se for desejado que a saída seja diretamente legível pelo MATLAB, a constante `OUTPUT_MATLAB` deve ser definida, ou pelo comando de compilação (`-DOUTPUT_MATLAB`, que é o método usado no `Makefile`) ou então através de um comando `#define` no arquivo `circuitAnalysis.h` . Se essa macro não for definida, a saída será um arquivo `.txt` contendo somente uma tabela, sem nenhum comando para o matlab.
- Alguns sistemas não implementam a função `strtold()` do header `stdlib.h` . Se houver algum problema durante a compilação relacionado a essa função, deve-se acionar a macro `CHANGE_STRTOLD` (ou via linha de comando, ou via `#define`, como explicado acima)

Exemplos:

capacitor descarregando:
circuito simples com um capacitor e um resistor, o capacitor com condição inicial.

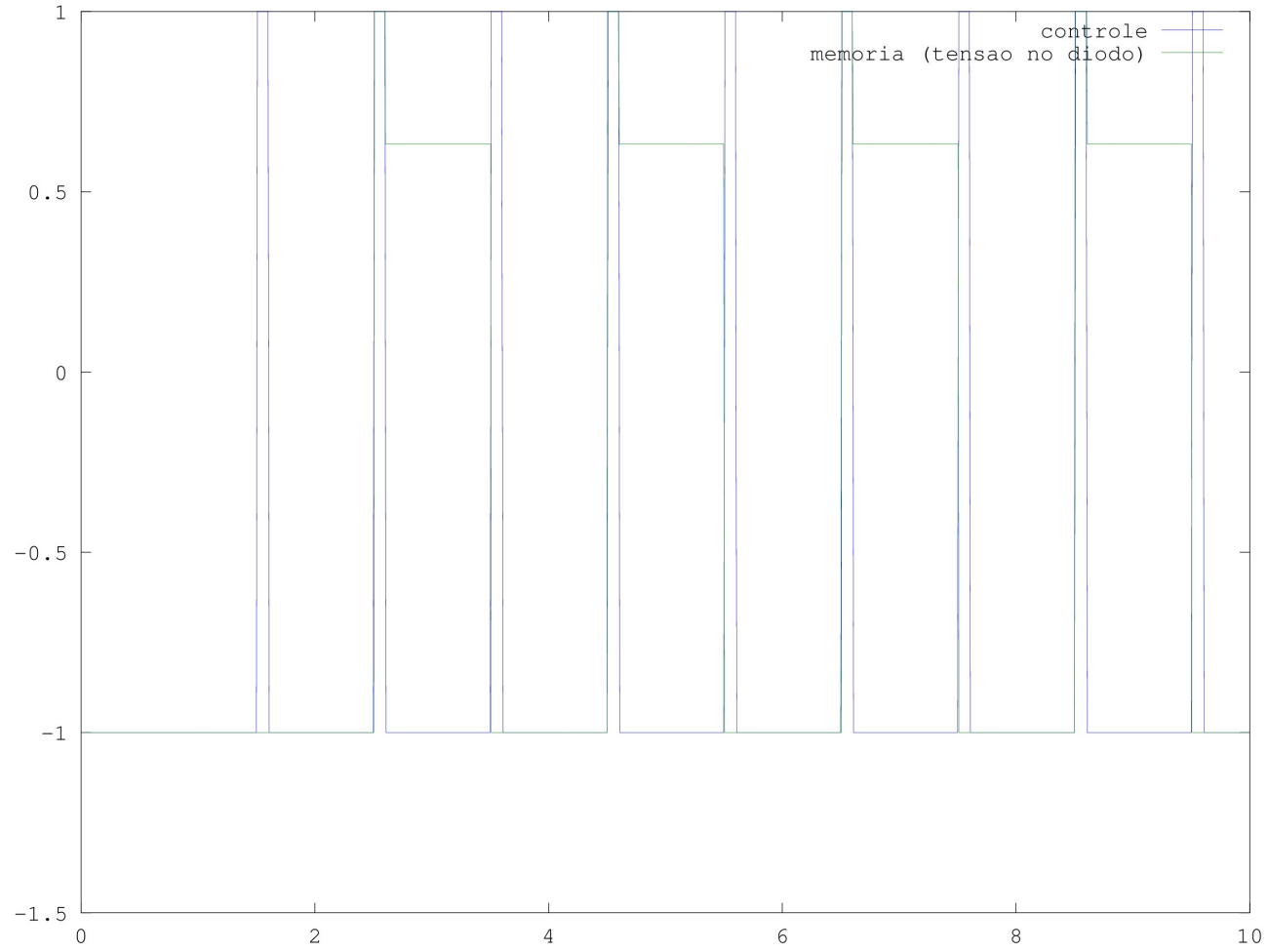
capacitor carregando:
Idem, mas sem condição inicial, e com uma fonte de tensão

```
V1 1 0 DC 3
R1 1 2 2
C1 2 0 1

.TRAN 20 .001 GEAR4 1
```



diodo memoria:

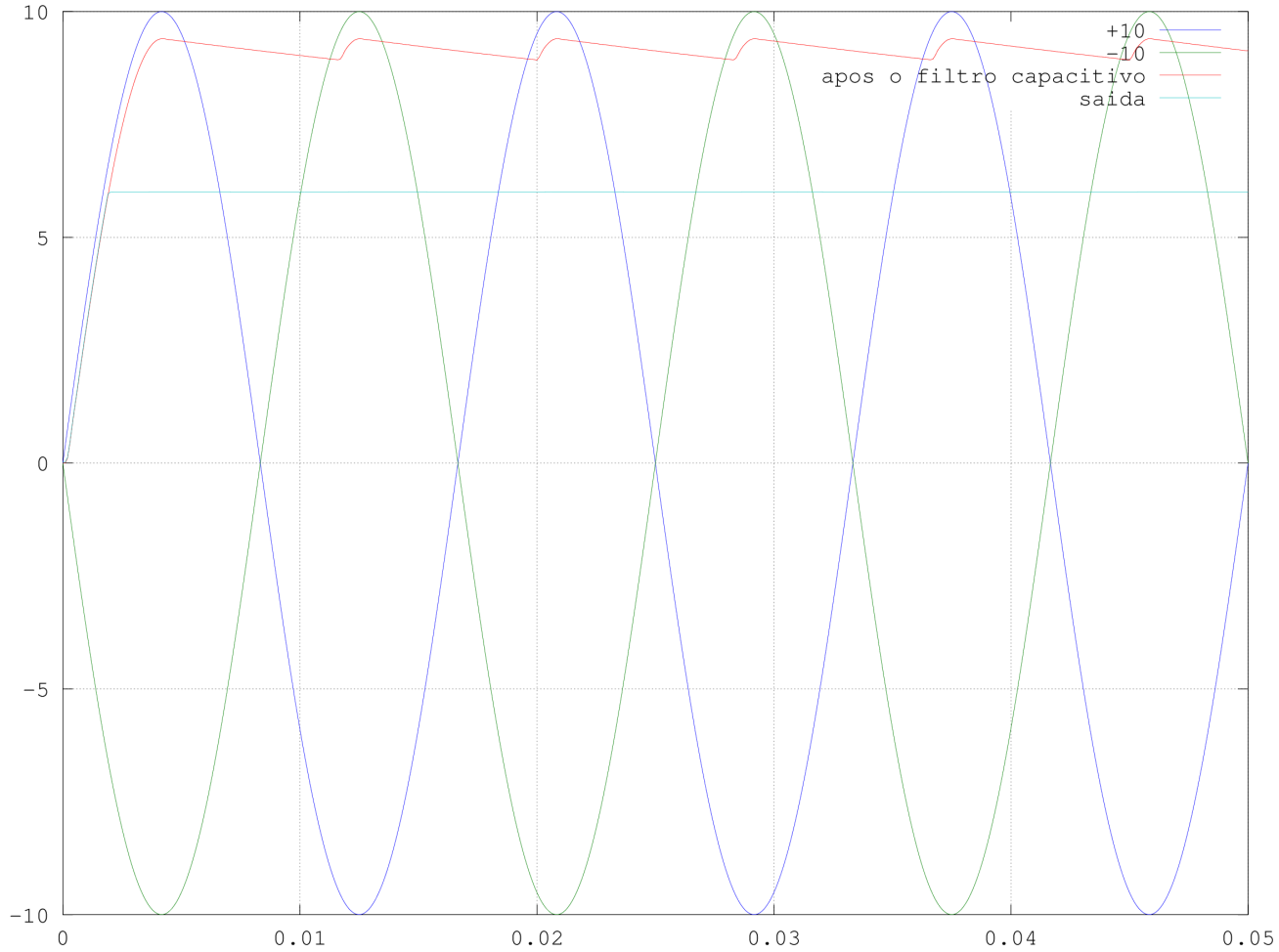
```
# A interseção da curva do diodo com a curva de um thevenin com resistor
# negativo tem duas soluções. Este circuito usa esse fato para implementar uma
# memória. Se a chave estiver aberta, a tensão no diodo é constante (é a
# memória; pode ser -1 ou 0.6). Quando a chave fecha, o valor da tensão Vpulso
# é "carregado" no diodo, que mantém esse valor após a chave abrir de novo.

# esta fonte é desconexa do resto do circuito
Vctrl 1 0 PULSE -1 1 1.5 .01 .01 .09 1 1000

V1 2 0 DC -1
R1 2 3 -2
N1 3 0 -10 -1.001e-6 -6e-3 -1e-6 0.6 .1e-3 1 10

$1 3 4 1 0 1e10 1e-10
Vpulso 4 0 PULSE -1 1 0 .1 .1 .9 2 1000

.TRAN 10 .005 GEAR3 1
```



filtro zener:

Fonte de tensão regulada com diodo zener

```
# transformador 120 -> 10
V1 1 0 SIN 0 10 60 0 0 0 100
V2 2 0 SIN 0 10 60 0 0 180 100

# diodos retificadores
N1 1 3 -10 -1.001e-6 -6e-3 -1e-6 0.6 .1e-3 1 10
N2 2 3 -10 -1.001e-6 -6e-3 -1e-6 0.6 .1e-3 1 10

# filtro capacitivo
Cfiltro 3 0 500e-6
R1 3 4 100

# zener regulador
Nzener 0 4 -7 -10 -6 1e-3 .6 .1e-3 1 10

# carga
RL 4 0 10e3

.TRAN .05 .0001 GEAR4 1
```

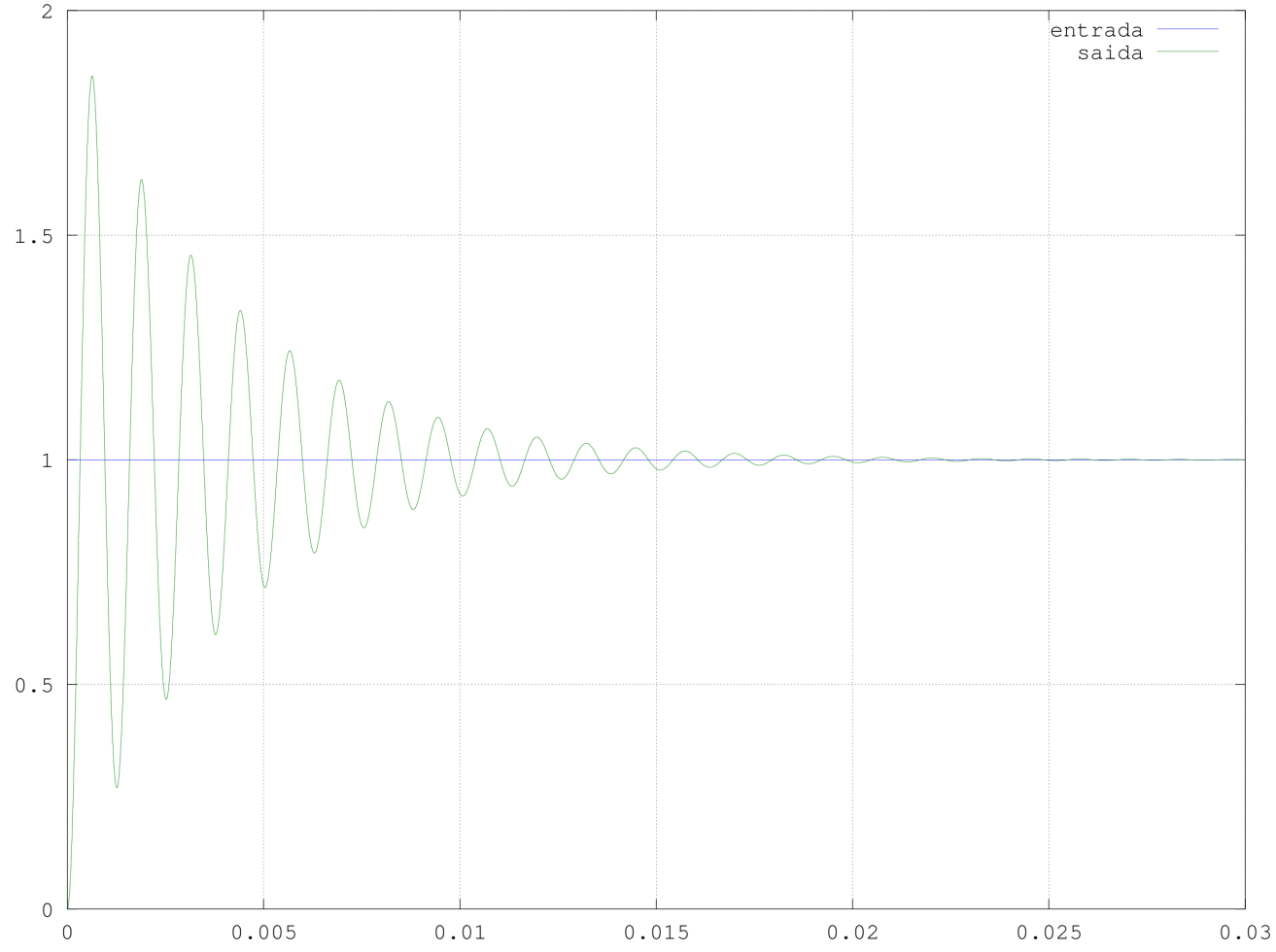


oscilador:
Um capacitor e um indutor, com condição inicial.

Passa altas:
Um filtro com topologia sallen-key passa altas

Passa baixas:
Idêntico, mas passa baixas

```
# passa-baixas sallen-key com Q=10, e freq=(1/2pi)5kHz

V1 1 0 DC 1
R1 1 2 10e3
R2 2 3 10e3
C1 2 4 400e-9
C2 3 0 1e-9
O1 4 0 3 4

.TRAN 30e-3 10e-6 GEAR4 1
```



resistencia negativa:
igual ao oscilador, mas com uma resistência negativa em paralelo. A tensão cresce exponencialmente.

resistivo:
Circuito com 7 nós, e várias casos degenerados, para testar a estabilidade numérica. Foi constatado, por exemplo, que uma resistência de 10e+9 ou mais ligada a um nó flutuando (ou seja, sem corrente) apresenta uma certa queda de potencial.

teste_sin:
Somente um nó e uma fonte de tensão com atraso, número de ciclos, atenuação, etc, etc.