Computer - Aided Circuit Design and Analysis 2022 Spring

Project 3

A Nonlinear Circuit Simulator

Deadline: 2022/6/17 23:59

Project Introduction

Please reference to the description in textbook p.198 (problem 4.8) and p.302 (problem 5.13) for details.

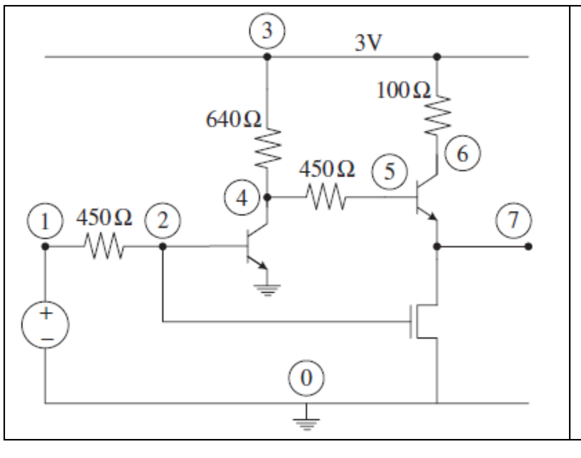
1. Nonlinear Algebraic Circuit

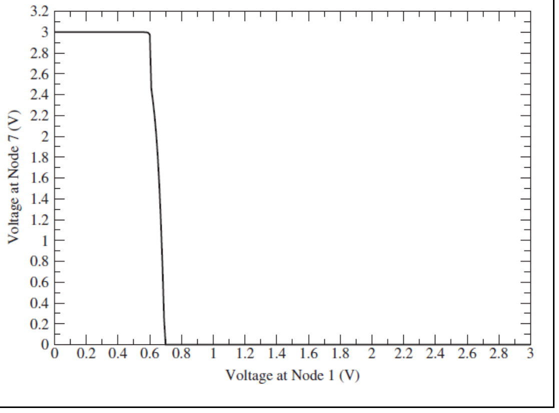
Based on the linear solver that was developed previously in Project2, write a C or C++ implementation of a DC solver for nonlinear resistive circuits, using Newton's method based on the use of companion models and element stamps. Your implementation should be general, in the sense that it should accept any linear or nonlinear circuit description consisting of any combination of linear resistors, independent voltage and current sources, diodes, BJTs, and MOSFETs. You will model diodes, BJTs, and MOSFETs using the simple models given in the text, with no series resistance. Thus, the diode has the standard exponential diode model (4.74), the BJT has the Ebers-Moll model we saw earlier in Fig. 4.20, and MOSFETs have the simplest (quadratic) model we saw above in (4.172), which is suitable for long-channel devices. As a first step in your solution, you should scan the element list and create the matrix G that is the contribution to the Jacobian by the linear elements, the vector s that is the contribution to the RHS vector by the linear elements, and the vector $g(x)$ of nonlinear functions that correspond to each nonlinear CVS or CCS in the network. When using Newton's method, you should build the Jacobian $J_f\left(x^{(k)}\right) \triangleq G + HJ_g(x^{(k)})$ by using element stamps, instead of by using the defining expression $G + HJ_g(x^{(k)})$ directly. In each Newton iteration, you should build the system $J_f\left(x^{(k)}\right)x^{(k+1)} = s^{(k)}$ by creating fresh copies of G and s and then adding the stamps due to the nonlinear elements.

In order to determine when to stop the Newton iterations, you should check both the size (norm) of the steps in x and the value (norm) of the function $f(x) = Gx + Hg(x) - s$, using a relative tolerance of 0.1% and an absolute tolerance of 1mV (for voltages) and $1\mu A$ (for currents). To evaluate the function $f(x)$, you need to know the matrix H, which you can infer from the incidence relations in the circuit, and you need to evaluate $g(x)$. In order to improve the chances of convergence in Newton's method, you should use generalized damping with $\gamma = 1.3$ and $k = 16$.

Use your code to perform a DC Sweep of the circuit shown in Fig. 4.31, based on the following parameters. For the MOSFET, $V_t = 0.6V$, $\lambda = 0.01/\text{V}$, and $\beta = 0.5\text{mA}/V^2$. For the BJT, $\alpha_F = 0.99$, $\alpha_R = 0.02$, $I_{es} = 2 \times 10^{-14}\text{A}$, $I_{cs} = 99 \times 10^{-14}\text{A}$, and $V_{Tc} = V_{Te} = 26mV$. Generate a plot of the output DC voltage (at node 7) versus the input DC voltage (at node 1), as it is swept from 0V to 3 V, in steps of 0.01 V. The correct solution is shown in Fig. 4.32.

The test circuit and a reference result are shown as follow:
(An example of circuit description is shown in **testCircuit1.txt**)



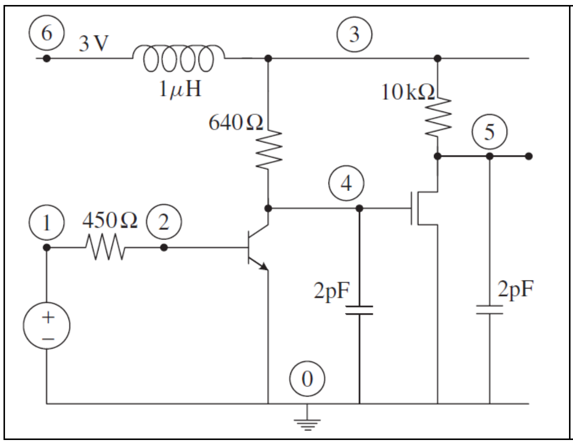


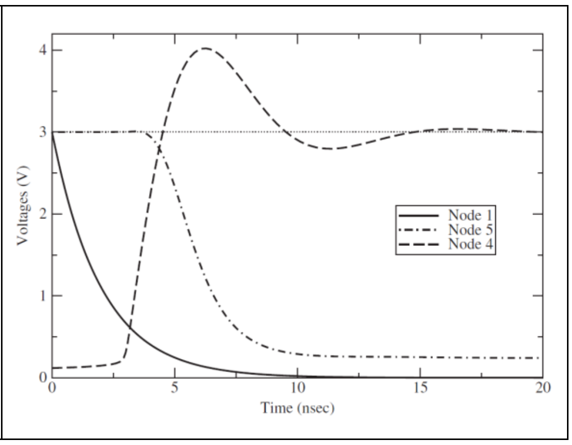
2. Differential Circuit Equations

Based on the nonlinear solver that was developed previously in part1, write a C or C++ implementation of a time-domain circuit simulator, based on the trapezoidal rule. Your implementation should be general, in the sense that it should accept any circuit description consisting of any combination of linear resistors, independent voltage and current sources, diodes, BJTs, MOSFETs, and linear capacitors and inductors. As in part1, you should use the simple DC models for the semiconductor devices given in chapter 4.

Use your code to perform a time-domain simulation of the circuit in Fig. 5.26, based on $v_1(t) = 3e^{-t/\tau}\,V$, where $\tau = 2ns$, over the interval $[0, 20ns]$, and using the following parameters. For the MOSFET, $V_t = 0.6V$, $\lambda = 0.01/\text{V}$, and $\beta = 0.5\text{mA}/V^2$. For the BJT, $\alpha_F = 0.99$, $\alpha_R = 0.02$, $I_{es} = 2 \times 10^{-14}\text{A}$, $I_{cs} = 99 \times 10^{-14}\text{A}$, and $V_{Tc} = V_{Te} = 26mV$. For the Newton stopping criteria, use a relative tolerance of 0.1% and an absolute tolerance of 1mV (for voltages) and 1μA (for currents). The overall flow of your solution should be as shown in Fig. 5.25, based on a minimum allowable time-step of 1psec. For the time-step control scheme in Fig. 5.25, you should use the following thresholds for the PLTE vector. To check if the PLTE is too large, use (5.413) based on a relative tolerance of 0.1% and an absolute tolerance of 1mV (for voltages) and 1μA (for currents). To check if the PLTE is too small, use a relative tolerance of 0.001% and an absolute tolerance of 10μV (for voltages) and 10nA (for currents). Generate a plot of the voltage waveforms at nodes 1, 4 and 5. The correct solution is shown in Fig. 5.27.

The test circuit and a reference result are shown as follow:
(An example of circuit description is shown in **testCircuit2.txt**)





Output Format

<**Mode 1**> The output format should be output as following:

```
<Node1_voltage>   <Node7_voltage >
.
.
.
```

<**Mode 2**> The output format should be output as following:

```
<Time sample_nsec>   <Node1_voltage>   <Node4_voltage >   <Node5_voltage>
.
.
.
```

Language

C/C++

Platform

Linux (Please make sure your code is available on linux server).

Warning

Several circuit description files will be put into your program for testing and be scored.

Naming, Programming, and Command rules

You should output the file of the results as the format shown in above examples.
Your program should take the command-line arguments as follows:
``**./Project3 [mode] [input] [output]**"
(example: **./Project3 1 testCircuit1.txt output1.txt**)
(example: **./ Project3 2 testCircuit2.txt output2.txt**)

Mode:
1: DC solver
2: AC solver (time-domain circuit simulator)

Submission

Please upload the following materials in a .zip file (e.g. **Project_3_Student_ID.zip**) to E3 by the deadline, specifying your student ID in the subject field.
(1) S Source code (.cpp/.c, .h, or Makefile).
(2) Executable binary.
(3) A Readme file (Information of how to compile/execute your codes/program.)

Grading Policy

(1) **Plagiarism is not allowed.**
(2) **Correctness: 90%, Runtime: 10%.** If the output files are correct, you will get at least 90 points. Note that several hidden cases will be tested on your program.