**

**ECE 4960: Computational and Software Engineering**

**Spring 2018**

**Programming Assignment 4: Compact SPICE for ODE Solution**

**1. Goal**

1. Learn to compose the transient simulation of circuits into ODE
2. Understand the numerical techniques behind ODE solvers
3. Use the error estimator to learn validation and adaptivity in physical simulations

**2 Background preparation**

**2.1 Introduction of SPICE**

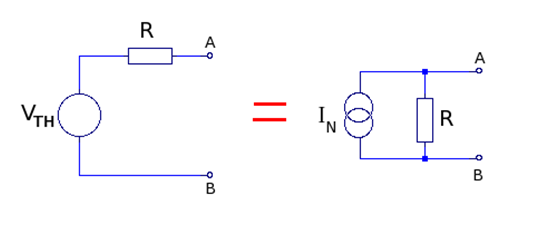
You probably have used SPICE for circuit simulation since your sophomore year in your introductory circuit class. SPICE is also embedded in the numerical engine under your Cadence or Synopsis CAD platform. SPICE stands for Simulation Program with Integrated Circuits Emphasis. The integrated circuits (IC) in the naming means that all components in SPICE are much smaller than the shortest wavelength in consideration, where the wavelength is defined as *c/freq* with *c* being the speed of light and *freq* being the highest frequency of the circuit functions.

The regular SPICE commands contain three major parts: the netlist (describing the circuits), the control (parameters for the solution method) and the output (print and plot of the nodal solutions). In our compact SPICE exercise, we will only implement the transient part of the SPICE simulation with some restriction on the netlist to ease up the implementation. However, most of the essence in the SPICE transient analysis is included, so SPICE is demystefied for you after using it for so long.

**2.2 Circuit elements in nodal analysis**

One of the ways to transform a topological description of circuits to the ODE formulation is the nodal analysis, where the voltage on each circuit node is the dependent variable *x* in the ODE *dx/dt = f(x, t)*, and the function *f(x, t)* is derived from KCL. All nodal voltage is referenced to a ground node 0. We will implement only a few element types in the conventional nodal analysis: resistor *R*, capacitor *C*, current source *i(t)* and NMOS transistors with the EKV large-signal model in Program Assignment 3. The nodal voltages will be the unknown dependent variables, and the current on each element branch will be assembled to the ordinary differential equations to be solved. Furthermore, we will assume that every node will have a capacitance to ground. This is the reality in integrated circuits, as no nodal voltage can respond instantaneously. However, the SPICE is also used to teach circuits, so not every node has a grounded capacitor in the regular SPICE. The selection and constraint are meant to make your ODE assembly easier, but without losing the general traits in the transient nodal analysis. For *R, C, i(t)* and NMOS, the current on the branch can be readily expressed. For example, the currents on *R* and *C* will be: *(V1 – V2)/R* and *C⋅d(V1 – V2)/dt*.

We do not include ideal voltage source here. Any realistic voltage source will have a source resistance, and then we can transfer the voltage source to a current source with the Norton equivalent circuits, as shown in Fig. 1. We can readily use the Thevenin or the Norton functional forms in the ODE with the source resistance. For ideal voltage sources, as the connected nodal voltage will be known but the branch current is unknown, a substitution of variables is needed. This will increase your implementation complexity, although it is not a difficult thing to program.



**Fig. 1.** The Thevenin and Norton equivalent circuits where *IN* (Norton) = *VTH/R* (Thevenin). Notice that the nodal voltage solutions of nodes A and B are identical in these two circuits, but not the power to be provided by the independent sources.

*R, C* and *i(t)* are two terminal devices, so they are not difficult in the assembly of KCL assuming the nodal voltage is the unknown depedent variables in the ODE. However, the transistor will be a three-terminal device (no further simplification possible in general), and we will use the following circuits in Fig. 2 for the transistor connection. The current source *ID, EKV* will be implemented as in Program Assignment 3. We will assume there is no direct DC current to the gate node (i.e., G will only have the *RC* displacement current).

*ID, EKV(VGS, VDS)*

*VGS*

***G***

***S***

***D***

*+*

*−*

**Fig. 2**. The MOSFET large-signal π model.

*VDS*

*+*

*−*

(1)



The nodal voltages here are *VG* and *VD* for nodes G and D, with the node S assumed to be at ground (node 0). Surely there are transistor circuits when S is not grounded (say common-gate amplifiers and source followers), but the additional transformation is again not difficult to modify in your program.

**2.3 From SPICE to ODE: Assembly of ODE from nodal analysis by KCL**

We will use a simple RC circuit to illustrate the ODE assembly from KCL. In Fig. 3, we have two nodes, three resistors, 2 grounded capacitors on the two nodes, and one time-dependent current source.

*R1*

*R2*

*i(t)*

**1**

*R3*

**2**

**0**

*C1*

*C2*

**Fig. 3**. Simple RC circuits to illustrate the ODE assembly from the circuit schematic.

KCL on node 1: ;



KCL on node 2: ;



To cast these two KCL equations in the form of :



We will give another example of a common-source (CS) amplifier with gate resistance in Fig. 4. The CS amplifier is a fundamental building block for many analog circuit, which provides a very large voltage gain in the small-signal range of *vin(t)*. We will now try to transform the circuits into an ODE description.

*RG*

*RL*

**1**

**2**

**0**

**Fig. 4**. A common-source amplifier

*vout*

*vin(t)*

*+*

*−*

*+*

*−*

From the equivalent circuits in Fig. 5(a) as transformed from Fig. 2,

KCL on node 1: ;



KCL on node 2: ;



To cast these two KCL equations in the form of : (notice that due to the nonlinear *ID,EKV* term, we cannot express the system in the array form).



;



;



*iin(t)*

*RL*

*C1*

*C2*

**Fig. 5**. Two possible equivalent circuits for the common-source amplifier: (a) Unilateral; (b) Bilateral coupling due to the Miller capacitance *CGD*.

*ID, EKV*

*RG*

*iDD*

**1**

**0**

**2**

*RL*

*C1*

*C2*

*ID, EKV*

*RG*

*iDD*

**1**

**0**

**2**

*CGD*

**(a)**

**(b)**

*iin(t)*

**2.4 Cross capacitor assembly**

Notice that the equivalent circuits in Fig. 5(a) is unilateral, so *f1(V1, V2)* does not depend on *V2*, as the CS amplifier can have input mostly isolated from output in low frequencies. This is called the unilateral property. However, if we consider the Miller capacitance *CGD*, we will have one more current component for the nodal equations on 1 and 2:, which will add to both KCL before formulating . Now the two KCL equations will become:



KCL on node 1: ;



KCL on node 2: ;



After some reassembly, we can achieve the two ODE with coupled time derivative of *V1* and *V2*.

,



where



Now we can transform the KCL to the usual ODE form with:



We notice that the matrix inverse is always valid as long as one of *C1* and *C2* is non-trivial. The Miller effect is closely related to this matrix. We also notice even if there are many other nodes and variables, we can always use this 2×2 matrix to handle the cross capacitance assembly.

**3 Tasks**

1. Construct the top-down functions to solve the generic ODE in the form of . The rank of the solution vector should be flexible. You may need larger rank to solve in Program Assignment 5. As we will perform only one-step methods in the following, you will create a general time stepping scheme of *xi+1 = xi + φh*. Different methods will call for the corresponding increment function *φ*.



1. Indicate what has been inherited and revised from your previous program assignments.
2. Perform a validation of your ODE solver by the test problem of , with the exact solution for *x(0)* = 2 as: . You have done the exact problem in your Hacker Practice. We will implement the forward Euler method and the RK34 methods without time adaptation and with time adaptation. In RK34 time adaptation, choose one of the error estimation for your time stepping. Validate your results by |*εx*| in different methods for the above test problem. Use plot program to generate normalized error vs time plot for different solvers. In the SPICE language, this transient analysis is:



.TRAN x START=0s STOP=5s STEP=1s

1. For the circuits in Fig. 3, use *R1 = R2 = R3* = 10kΩ, and *C1 = C2* = 1pF. *V1(0) = V2(0)* = 0. The current source is described by Fig. 6 as a large-signal transient. You should simulate from 0ns to 100ns with 0.2ns and 1ns step size (500 and 100 steps for methods without adaptation). Output *V1(t)* and *V2(t)* to be used in the plot program of your choice. The RC circuits will be like a low-pass filter, and the time constant given here is about 10ns. Compare the results from all three one-step methods.

.TRAN V1 V2 START=0s STOP=100ns STEP=0.2ns

*1ns*

*10ns*

**Fig. 6**. Transient *i(t)* for the circuits in Fig. 3.

*9ns*

*11ns*

*9ns*

*20ns*

*21ns*

*30ns*

*31ns*

*0.1mA*

1. For the circuits in Fig. 5(a), use *IS* = 5μA, *κ* = 0.7, and *Vth* = 1.0V for the EKV model. *VDD* = 5V. *RG = RL* = 10kΩ. *C1 = C2* = 1pF.  *V1(0) = V2(0)* = 2.5V. Use the same *iin(t)* as in Fig. 6, and simulate the same period. Output *V1(t)* and *V2(t)* to be used in the plot program of your choice. The CS amplifier will amplify *ΔV1* to *ΔV2*, but *V2*(t) will be bounded between 0 and 5V (voltage undershoot and overshoot possible with *CCG*, which is however not included here). Compare the transient results from all three one-step methods.

**4 Execution**

* This project can be done by groups of 2 students, but each student needs to know how to run the program as it would be used in later assignments.
* I expect the time of your programming to be similar, but may be shorter than, Program Assignment 3.

**5 Due dates and grading**

* Upload your code and reports to Github, and send the link by email before 4/22 end of day.
* Grading distribution will be very similar to Program Assignment 3:
  + The design of solver, testing and validation codes, following the top down programming practice and code organization. Keep solver code separate from test code. Maintain proper header files. Indicate inherited codes from previous projects in report. (30%).
  + Testing of ODE solvers, that includes task 3, as well as of other helper functions, and report test results. (25%)
  + Your understanding and implementation of the ODE solvers to complete all tasks, and reporting of the output solver results and plots for different time stamps, with comment on the observation. (35%)
  + The in-code documentation and readability of code. (10%)

**6. Additional circuit element of inductors**

We will illustrate here how to assemble an inductor into the ODE of the circuits. Other elements such as diodes, BJT and transformers can also be added, but that will leave to your own curiosity.

Consider the following resonance circuits with only one nodal voltage *V1* to solve. However, we know from our circuit class that this will become a second-order differential equation with both L and C elements. Following our principle of adding variables to decrease the order of ODE, we can opt to use *iL* as a basic variable. We can formulate the KCL equation and the inductor equation:

Node 1:



Inductor:



*iin(t)*

*R*

*L*

*C*

*V1*

*+*

*−*

**Fig. 7.** An RF resonator with RLC in parallel.

We can assemble our common ODE form by using *V1* and *iL* as the basic variables:



**7. Some comments in SPICE linearization for forming ODE**

In SPICE version 2 and before, the circuits are linearized before “integration” (the increment function accumulates the time steps to *x*, and therefore is equivalent to integration). When the circuits are linear, we can see that the nodal analysis will assemble to the conductance matrix as in the circuits of Fig. 3. If the circuits are linearized, we can view the capacitance as a current source in parallel with a resistance by the following transformation:

;



An integration method needs to be chosen here. The trapezoidal rule is used as an example:

;



Although this is a convenient view, the linear circuits depend on the numerical scheme, which makes the implementation of different discretizations more difficult. Later SPICE has changed to more generic ODE solution hierarchy.