Computer Engineering Department
Faculty of Engineering
Cairo University

Modeling & Simulation **Fall 2020**

Lab Requirement #3

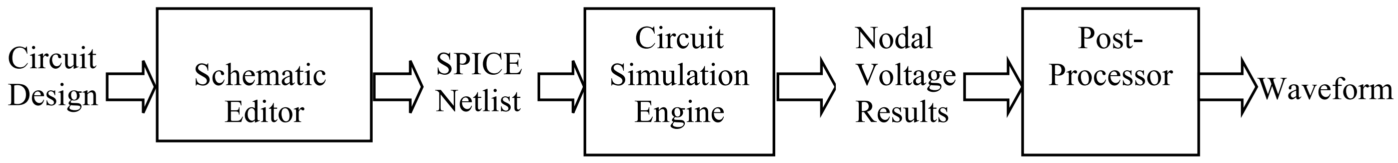
Introduction

SPICE Circuit simulator is a well-known modern circuit simulator that uses direct methods to simulate analog circuits at the differential equation level. It takes an analog circuit schematic, simulates, calculates and plots the voltage at each node, and the current produced by each voltage source.

Requirements:

You're required to develop a simulator tool like SPICE simulator using any general purpose programming language (python/C/C++/C# or Java). Your simulator should be able to simulate circuits composed of: Resistor, Capacitor, Inductor, Voltage Source and Current Source.

System Architecture:



i. Schematic Editor:

- designer should be able to use the GUI to specify the design.
- check for valid circuit (floating wires, unconnected nodes and missing ground).
- generate a Netlist file to be passed to the simulator engine.

Netlist Format:
Simulation_time
# of iterations
-Component 1
-Component2
…
-1 (indicates end of input)
Each component has the following format:

Component_Type | Node1 | Node2 | Value | Initial_Value

- Component_Type stands for the type of the component. It can be one of the following:

- Voltage Source "Vsrc".
- Current Source "Isrc".
- Resistance "R".
- Inductor "I".
- Capacitor "C".

- Node1 and Node2 stands for the nodes" numbers to which the component is connected to. (for uni-polar components as Vsrc, Node1 is the +ve port, Node2 is the –ve port)
- Value is the physical value of the component.
- Initial_Value is the initial current or voltage that is observed on the component at time=0.

ii. Circuit Simulation Engine:

- The simulation engine is responsible for performing all the numerical calculations necessary to compute the various circuit parameters (i.e. nodal voltages and voltage source currents).
- You can use the Modified Node Analysis algorithm with Backward Euler method. A short review on this algorithm will be presented during the tutorial.

iii. Post-Processor:

- The module is responsible for displaying the results of the simulator in both tabular form and waveform.

Delivery and Logistics:

- You're required to develop the **Circuit Simulation Engine** only. No need for GUI (Schematic Editor and Post Processor).
- This week, we'll explain how to handle Voltage sources, Current sources, and Resistances.
- The following week, we'll also handle Inductors and Capacitors.
- Your deliverable will eventually handle all of these components. (Though it is encouraged to build it gradually in parallel with the labs).
- Due date: **Thursday, December 10**
- **Groups of two**
- You should submit a single file containing your code (.ipynb/.py). The file name is the group members' names (e.g. OsamaNabih_and_SalmaAbdelMonem).
- Some test cases and their respective outputs are provided for validation.

NOTE:

1- You can use any matrix library (for matrix determinant, adjunct or inversion)
2- Matlab is not allowed

Appendix 1: Matrix Operations

You will need to remember some matrix operations.

1. **Matrix Transpose**
   The transpose of a nxm matrix is the swapping of rows & columns to produce a mxn matrix
   $A_{ij} = A_{ji}$

2. **Determinant of Matrix**
   The determinant of a nxn matrix could be computed by:
   $$\det(A) = \sum_{j=1}^{n} (-1)^{i+j} a_{i,j} M_{i,j}$$
   Where $M_{i,j}$ is the determinant of a subgroup of the larger matrix A, after deleting a row i & col j from A

3. **Adjunct of Matrix**
   Adjunct of Matrix A is the transpose of cofactor Matrix C
   $$\operatorname{adj}(\mathbf{A}) = \mathbf{C}^{\mathsf{T}}$$
   Where each element of C is $M_{i,j}$ the determinant of subgroup of the larger matrix A, after deleting a row i & col j from A
   $$\mathbf{C}_{ij} = (-1)^{i+j} \mathbf{M}_{ij}$$

4. **Matrix Cross Multiplication (x)**
   2 Matrices could be multiplied if matrix A is [n x m] & matrix B is [m x p], so the output matrix C will be [n x p], where
   $C_{ij} = \mathrm{Sum}_m(A_{im} * B_{mj})$
   $$\mathbf{AB} = \begin{pmatrix} a & b & c \\ p & q & r \\ u & v & w \end{pmatrix} \begin{pmatrix} \alpha & \beta & \gamma \\ \lambda & \mu & \nu \\ \rho & \sigma & \tau \end{pmatrix} = \begin{pmatrix} a\alpha + b\lambda + c\rho & a\beta + b\mu + c\sigma & a\gamma + b\nu + c\tau \\ p\alpha + q\lambda + r\rho & p\beta + q\mu + r\sigma & p\gamma + q\nu + r\tau \\ u\alpha + v\lambda + w\rho & u\beta + v\mu + w\sigma & u\gamma + v\nu + w\tau \end{pmatrix}$$

5. **Matrix Dot Multiplication (.)**
   2 Matrices could be dot multiplied by multiplying each matrix element
   $C_{ij} = A_{ij} * B_{ij}$

6. **Matrix Inverse**
   Matrix Inverse is calculated using the determinant & adjunct of the matrix
   $$A^{-1} = \frac{1}{det\,(A)}\; adj\,(A)$$

Appendix 2: Coding Advices

1. Recursive Functions

When writing your matrix class, remember that you will need to use recursion. For example: when coding determinant of A “Det_Mat(A,n)”,
Det_A = Det_A + ((-1)^(i+j) * A[i,j] * Det_Mat(Sub_Mat(A,i,j),n-1));

2. Incremental Development

After you finish implementing a function (as determinant), test your function using the results from any online calculator as;
https://matrix.reshish.com/
http://onlinemschool.com/math/assistance/matrix/