Techniques for circuit simulation

M. B. Patil
www.ee.iitb.ac.in/~sequel

Department of Electrical Engineering
Indian Institute of Technology Bombay

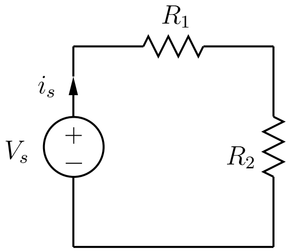
Outline

* Circuit simulation: introduction
* Nodal analysis
* Modified nodal analysis
* Sparse tableau approach
* Nonlinear circuits
* Transient (dynamic) analysis

Circuit simulation

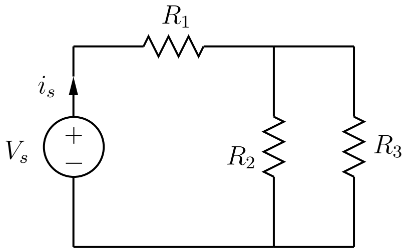
* DC analysis
* transient (time-domain) analysis
* AC (frequency-domain) analysis
* logic-level simulation
* mixed-signal simulation
* noise computation
* periodic steady state computation
* sensitivity analysis

Why do we need circuit simulation?



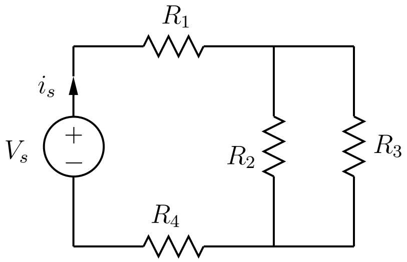
Example 1

Why do we need circuit simulation?



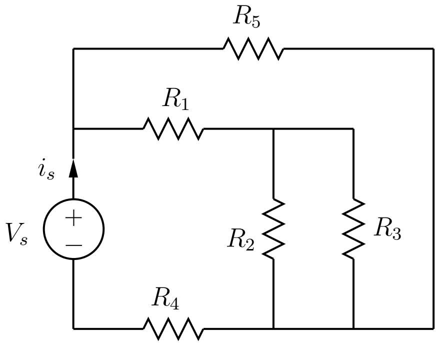
Example 1

Why do we need circuit simulation?



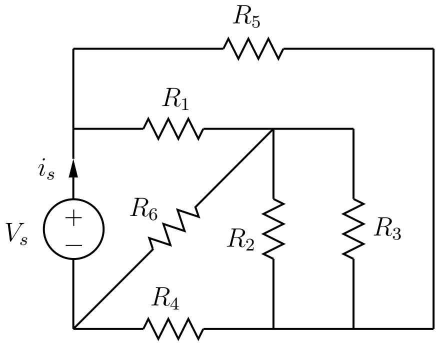
Example 1

Why do we need circuit simulation?



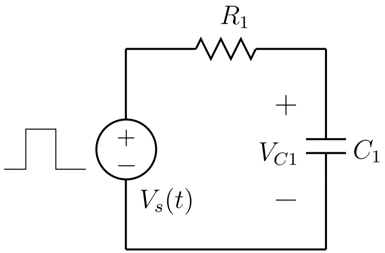
Example 1

Why do we need circuit simulation?



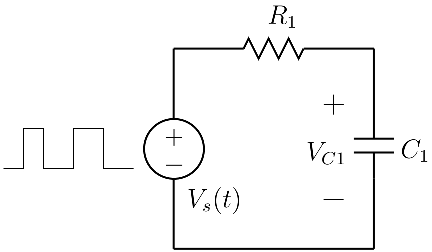
Example 1

Why do we need circuit simulation?



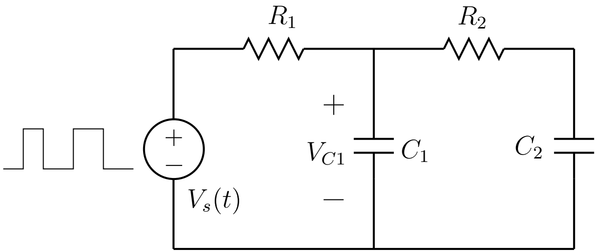
Example 2

Why do we need circuit simulation?



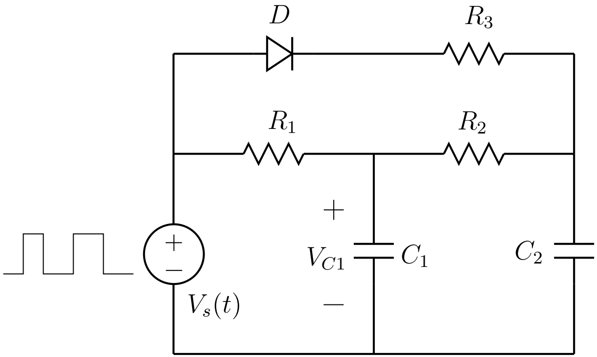
Example 2

Why do we need circuit simulation?



Example 2

Why do we need circuit simulation?



Example 2

Circuit simulation on a computer

* Must be efficient in terms of CPU time (especially for large circuits).

Circuit simulation on a computer

* Must be efficient in terms of CPU time (especially for large circuits).
* Must make good use of the memory available. If a matrix is sparse, it should not be stored in the $a(i,j)$ form.

Circuit simulation on a computer

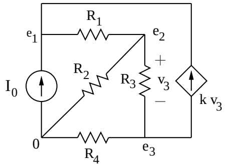
* Must be efficient in terms of CPU time (especially for large circuits).
* Must make good use of the memory available. If a matrix is sparse, it should not be stored in the $a(i, j)$ form.
* The approach must be systematic. "Tricks" such as resistors in series or parallel, star-to-delta conversion, etc. will work in special cases. What we need is a *general-purpose* method that will work for *all* circuits.

Outline

* Circuit simulation: introduction
* Nodal analysis
* Modified nodal analysis
* Sparse tableau approach
* Nonlinear circuits
* Transient (dynamic) analysis

Nodal Analysis of a linear circuit

* Take some node as the "reference node" and denote the node voltages of the remaining nodes by e_1, e_2, etc.

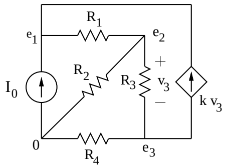


SPICE file

```
I0 1 0 1m
R1 1 2 1k
R2 2 0 1.2k
R3 2 3 200
R4 0 3 1k
VCCS1 1 3 2 3 0.5m
```

Nodal Analysis of a linear circuit

* Take some node as the "reference node" and denote the node voltages of the remaining nodes by e_1, e_2, etc.
* Write KCL at each node in terms of the node voltages. Follow a fixed convention, e.g., current *leaving* a node is *positive*.

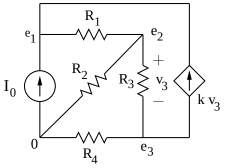


SPICE file

```
I0 1 0 1m
R1 1 2 1k
R2 2 0 1.2k
R3 2 3 200
R4 0 3 1k
VCCS1 1 3 2 3 0.5m
```

Nodal Analysis of a linear circuit

* Take some node as the "reference node" and denote the node voltages of the remaining nodes by e_1, e_2, etc.
* Write KCL at each node in terms of the node voltages. Follow a fixed convention, e.g., current *leaving* a node is *positive*.
* When all KCL equations are treated, we have the "admittance matrix" and the RHS vector.

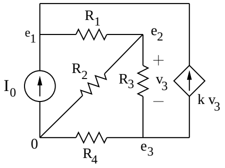


SPICE file

```
I0 1 0 1m
R1 1 2 1k
R2 2 0 1.2k
R3 2 3 200
R4 0 3 1k
VCCS1 1 3 2 3 0.5m
```

Nodal Analysis of a linear circuit

- Take some node as the "reference node" and denote the node voltages of the remaining nodes by e_1, e_2, etc.
- Write KCL at each node in terms of the node voltages. Follow a fixed convention, e.g., current *leaving* a node is *positive*.
- When all KCL equations are treated, we have the "admittance matrix" and the RHS vector.
- Solve the resulting linear system of equations, $\mathbf{Ye} = \mathbf{I}_s$ for the node voltages.

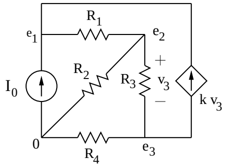


SPICE file

```
I0 1 0 1m
R1 1 2 1k
R2 2 0 1.2k
R3 2 3 200
R4 0 3 1k
VCCS1 1 3 2 3 0.5m
```

Nodal Analysis of a linear circuit

- Take some node as the "reference node" and denote the node voltages of the remaining nodes by e_1, e_2, etc.
- Write KCL at each node in terms of the node voltages. Follow a fixed convention, e.g., current *leaving* a node is *positive*.
- When all KCL equations are treated, we have the "admittance matrix" and the RHS vector.
- Solve the resulting linear system of equations, $\mathbf{Ye} = \mathbf{I}_s$ for the node voltages.
- The equation assembly (also called "parsing") can be done element-by-element, i.e., by considering one line of the circuit file at a time.

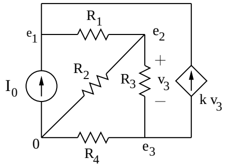


SPICE file

```
I0 1 0 1m
R1 1 2 1k
R2 2 0 1.2k
R3 2 3 200
R4 0 3 1k
VCCS1 1 3 2 3 0.5m
```

Nodal Analysis of a linear circuit

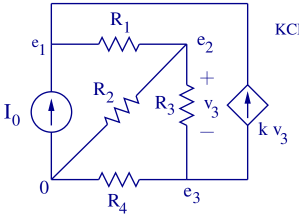
- Take some node as the "reference node" and denote the node voltages of the remaining nodes by e_1, e_2, etc.
- Write KCL at each node in terms of the node voltages. Follow a fixed convention, e.g., current *leaving* a node is *positive*.
- When all KCL equations are treated, we have the "admittance matrix" and the RHS vector.
- Solve the resulting linear system of equations, $\mathbf{Ye} = \mathbf{I}_s$ for the node voltages.
- The equation assembly (also called "parsing") can be done element-by-element, i.e., by considering one line of the circuit file at a time.
- The computer cannot *see* the entire circuit; it can, however, go through the circuit file line by line.



SPICE file

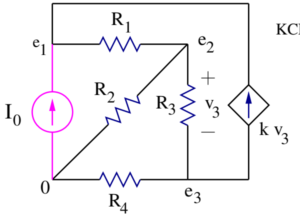
```
I0 1 0 1m
R1 1 2 1k
R2 2 0 1.2k
R3 2 3 200
R4 0 3 1k
VCCS1 1 3 2 3 0.5m
```

Step 1: Initialize



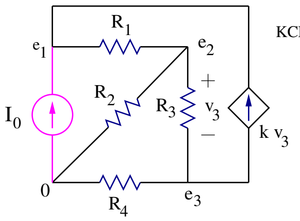
$$\begin{array}{r} \text{KCL at 1} \\ 2 \\ 3 \end{array} \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} e_1 \\ \hline e_2 \\ \hline e_3 \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline 0 \\ \hline 0 \end{array}\right]$$

Step 2



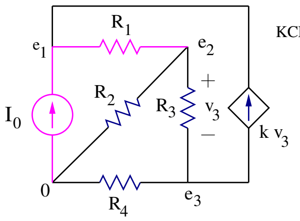
$$\begin{array}{r} \text{KCL at 1} \\ 2 \\ 3 \end{array} \left[\begin{array}{c:c:c} 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} e_1 \\ \hdashline e_2 \\ \hdashline e_3 \end{array}\right] = \left[\begin{array}{c} 0 \\ \hdashline 0 \\ \hdashline 0 \end{array}\right]$$

Step 2



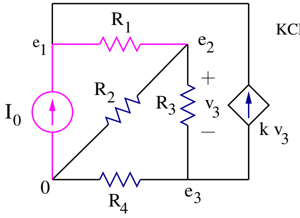
$$\begin{array}{r} \text{KCL at 1} \\ 2 \\ 3 \end{array} \left[\begin{array}{c:c:c} 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} e_1 \\ \hdashline e_2 \\ \hdashline e_3 \end{array}\right] = \left[\begin{array}{c} I_0 \\ \hdashline 0 \\ \hdashline 0 \end{array}\right]$$

Step 3



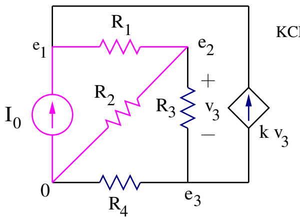
$$
\begin{array}{r}
\text{KCL at 1} \\ 2 \\ 3
\end{array}
\left[\begin{array}{c:c:c}
0 & 0 & 0 \\ \hdashline
0 & 0 & 0 \\ \hdashline
0 & 0 & 0
\end{array}\right]
\left[\begin{array}{c}
e_1 \\ \hdashline e_2 \\ \hdashline e_3
\end{array}\right]
=
\left[\begin{array}{c}
I_0 \\ \hdashline 0 \\ \hdashline 0
\end{array}\right]
$$

Step 3



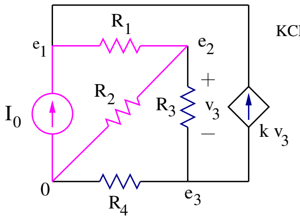
$$\begin{array}{r} \text{KCL at 1} \\ 2 \\ 3 \end{array} \left[\begin{array}{c:c:c} G_1 & -G_1 & 0 \\ \hdashline -G_1 & G_1 & 0 \\ \hdashline 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} e_1 \\ \hdashline e_2 \\ \hdashline e_3 \end{array}\right] = \left[\begin{array}{c} I_0 \\ \hdashline 0 \\ \hdashline 0 \end{array}\right]$$

Step 4



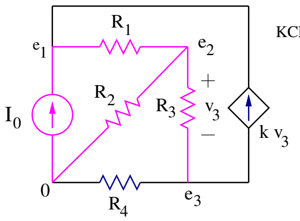
$$\begin{array}{l} \text{KCL at 1} \\ 2 \\ 3 \end{array} \left[\begin{array}{c|c|c} G_1 & -G_1 & 0 \\ \hline -G_1 & G_1 & 0 \\ \hline 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} e_1 \\ \hline e_2 \\ \hline e_3 \end{array}\right] = \left[\begin{array}{c} I_0 \\ \hline 0 \\ \hline 0 \end{array}\right]$$

Step 4



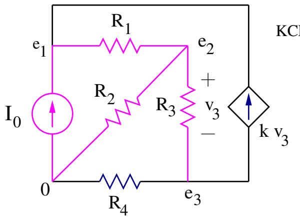
$$\begin{array}{r} \text{KCL at 1} \\ 2 \\ 3 \end{array} \left[\begin{array}{c:c:c} G_1 & -G_1 & 0 \\ \hdashline -G_1 & G_1+G_2 & 0 \\ \hdashline 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} e_1 \\ \hdashline e_2 \\ \hdashline e_3 \end{array}\right] = \left[\begin{array}{c} I_0 \\ \hdashline 0 \\ \hdashline 0 \end{array}\right]$$

Step 5



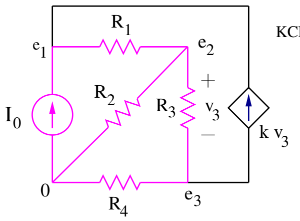
$$
\begin{array}{r}
\text{KCL at 1} \\ 2 \\ 3
\end{array}
\left[\begin{array}{c:c:c}
G_1 & -G_1 & 0 \\ \hdashline
-G_1 & G_1+G_2 & 0 \\ \hdashline
0 & 0 & 0
\end{array}\right]
\left[\begin{array}{c}
e_1 \\ \hdashline e_2 \\ \hdashline e_3
\end{array}\right]
=
\left[\begin{array}{c}
I_0 \\ \hdashline 0 \\ \hdashline 0
\end{array}\right]
$$

Step 5



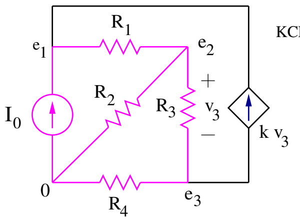
$$
\begin{array}{r}
\text{KCL at 1} \\ 2 \\ 3
\end{array}
\left[\begin{array}{c:c:c}
G_1 & -G_1 & 0 \\ \hdashline
-G_1 & \begin{array}{c}G_1+G_2\\ G_3\end{array} & -G_3 \\ \hdashline
0 & -G_3 & G_3
\end{array}\right]
\left[\begin{array}{c}
e_1 \\ \hdashline e_2 \\ \hdashline e_3
\end{array}\right]
=
\left[\begin{array}{c}
I_0 \\ \hdashline 0 \\ \hdashline 0
\end{array}\right]
$$

Step 6



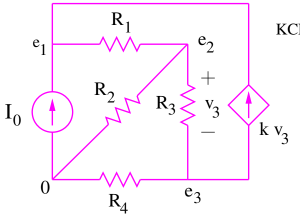
$$\begin{array}{r} \text{KCL at 1} \\ 2 \\ 3 \end{array} \left[\begin{array}{c:c:c} G_1 & -G_1 & 0 \\ \hdashline -G_1 & \begin{matrix} G_1{+}G_2 \\ G_3 \end{matrix} & -G_3 \\ \hdashline 0 & -G_3 & G_3 \end{array}\right] \left[\begin{array}{c} e_1 \\ \hdashline e_2 \\ \hdashline e_3 \end{array}\right] = \left[\begin{array}{c} I_0 \\ \hdashline 0 \\ \hdashline 0 \end{array}\right]$$

Step 6



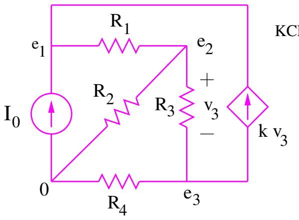
$$\begin{array}{r} \text{KCL at 1} \\ 2 \\ 3 \end{array} \left[\begin{array}{c:c:c} G_1 & -G_1 & 0 \\ \hdashline -G_1 & \begin{array}{c} G_1+G_2 \\ G_3 \end{array} & -G_3 \\ \hdashline 0 & -G_3 & G_3+G_4 \end{array}\right] \left[\begin{array}{c} e_1 \\ \hdashline e_2 \\ \hdashline e_3 \end{array}\right] = \left[\begin{array}{c} I_0 \\ \hdashline 0 \\ \hdashline 0 \end{array}\right]$$

Step 7



$$\begin{array}{r} \text{KCL at 1} \\ 2 \\ 3 \end{array} \left[\begin{array}{c:c:c} G_1 & -G_1 & 0 \\ \hdashline -G_1 & \begin{matrix} G_1+G_2 \\ G_3 \end{matrix} & -G_3 \\ \hdashline 0 & -G_3 & G_3+G_4 \end{array}\right] \left[\begin{array}{c} e_1 \\ \hdashline e_2 \\ \hdashline e_3 \end{array}\right] = \left[\begin{array}{c} I_0 \\ \hdashline 0 \\ \hdashline 0 \end{array}\right]$$

Step 7

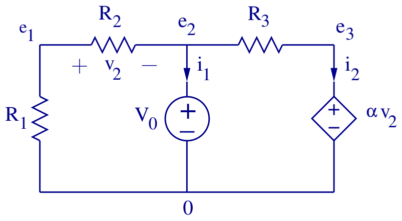


$$
\begin{array}{r}
\text{KCL at 1} \\ 2 \\ 3
\end{array}
\left[\begin{array}{c|c|c}
G_1 & -G_1-k & k \\
\hline
-G_1 & \begin{array}{c}G_1+G_2\\ G_3\end{array} & -G_3 \\
\hline
0 & -G_3+k & \begin{array}{c}G_3+G_4\\ -k\end{array}
\end{array}\right]
\left[\begin{array}{c}
e_1 \\ \hline e_2 \\ \hline e_3
\end{array}\right]
=
\left[\begin{array}{c}
I_0 \\ \hline 0 \\ \hline 0
\end{array}\right]
$$

Outline

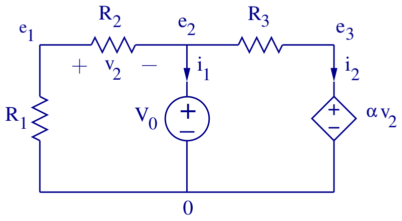
* Circuit simulation: introduction
* Nodal analysis
* Modified nodal analysis
* Sparse tableau approach
* Nonlinear circuits
* Transient (dynamic) analysis

Modified Nodal Analysis (MNA) of a linear circuit



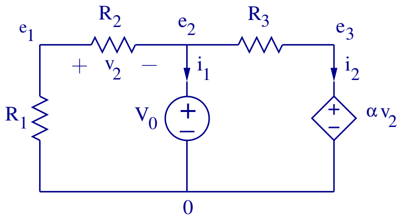
* When a voltage source is involved, we cannot write its current in terms of node voltages (e_1, e_2, etc.). The NA approach has to be modified $\Rightarrow$ MNA.

Modified Nodal Analysis (MNA) of a linear circuit



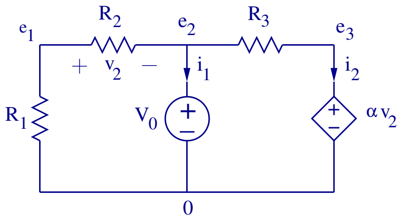
* When a voltage source is involved, we cannot write its current in terms of node voltages (e_1, e_2, etc.). The NA approach has to be modified $\Rightarrow$ MNA.
* Treat the current through the voltage source as an additional unknown.

Modified Nodal Analysis (MNA) of a linear circuit



* When a voltage source is involved, we cannot write its current in terms of node voltages (e_1, e_2, etc.). The NA approach has to be modified $\Rightarrow$ MNA.
* Treat the current through the voltage source as an additional unknown.
* We also need to get an additional equation since the number of unknowns has gone up by 1. This equation is provided by the branch equation of the voltage source.

Modified Nodal Analysis (MNA) of a linear circuit



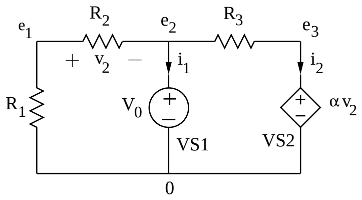
* When a voltage source is involved, we cannot write its current in terms of node voltages (e_1, e_2, etc.). The NA approach has to be modified $\Rightarrow$ MNA.

* Treat the current through the voltage source as an additional unknown.

* We also need to get an additional equation since the number of unknowns has gone up by 1. This equation is provided by the branch equation of the voltage source.

* The "solution vector" now contains the voltage source currents in addition to the node voltages.

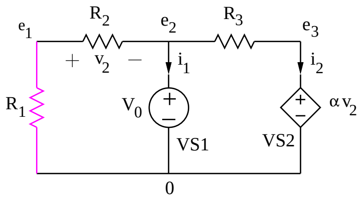
Modified Nodal Analysis of a linear circuit



$$
\begin{array}{l}
\text{KCL at 1} \\
\text{KCL at 2} \\
\text{KCL at 3} \\
\text{BCE for VS1} \\
\text{BCE for VS2}
\end{array}
\left[\begin{array}{c:c:c:c:c}
 & & & & \\ \hdashline
 & & & & \\ \hdashline
 & & & & \\ \hdashline
 & & & & \\ \hdashline
 & & & &
\end{array}\right]
\left[\begin{array}{c}
e_1 \\ \hdashline
e_2 \\ \hdashline
e_3 \\ \hdashline
i_1 \\ \hdashline
i_2
\end{array}\right]
=
\left[\begin{array}{c}
 \\ \hdashline
 \\ \hdashline
 \\ \hdashline
 \\ \hdashline

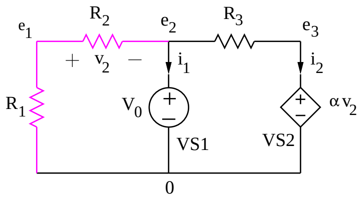
\end{array}\right]
$$

Modified Nodal Analysis of a linear circuit



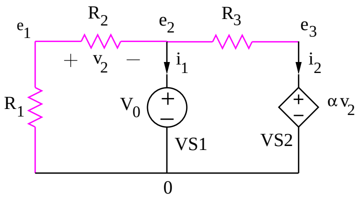
$$\begin{array}{l} \text{KCL at 1} \\ \text{KCL at 2} \\ \text{KCL at 3} \\ \text{BCE for VS1} \\ \text{BCE for VS2} \end{array} \left[\begin{array}{c:c:c:c:c} G_1 & & & & \\ \hdashline & & & & \\ \hdashline & & & & \\ \hdashline & & & & \\ \hdashline & & & & \end{array}\right] \left[\begin{array}{c} e_1 \\ \hdashline e_2 \\ \hdashline e_3 \\ \hdashline i_1 \\ \hdashline i_2 \end{array}\right] = \left[\begin{array}{c} \\ \hdashline \\ \hdashline \\ \hdashline \\ \hdashline \\ \end{array}\right]$$

Modified Nodal Analysis of a linear circuit



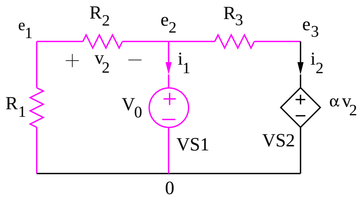
$$\begin{array}{l}\text{KCL at 1}\\ \\ \text{KCL at 2}\\ \\ \text{KCL at 3}\\ \\ \text{BCE for VS1}\\ \\ \text{BCE for VS2}\end{array}
\left[\begin{array}{c:c:c:c:c}
G_1+G_2 & -G_2 & & & \\ \hdashline
-G_2 & G_2 & & & \\ \hdashline
 & & & & \\ \hdashline
 & & & & \\ \hdashline
 & & & &
\end{array}\right]
\left[\begin{array}{c}
e_1\\ \hdashline e_2\\ \hdashline e_3\\ \hdashline i_1\\ \hdashline i_2
\end{array}\right]
=
\left[\begin{array}{c}
 \\ \hdashline \\ \hdashline \\ \hdashline \\ \hdashline \\
\end{array}\right]$$

Modified Nodal Analysis of a linear circuit



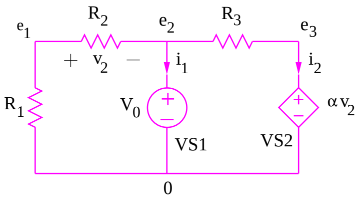
$$\begin{array}{r} \text{KCL at 1} \\ \text{KCL at 2} \\ \text{KCL at 3} \\ \text{BCE for VS1} \\ \text{BCE for VS2} \end{array} \left[\begin{array}{c:c:c:c:c} G_1+G_2 & -G_2 & & & \\ \hdashline -G_2 & G_2+G_3 & -G_3 & & \\ \hdashline & -G_3 & G_3 & & \\ \hdashline & & & & \\ \hdashline & & & & \end{array}\right] \left[\begin{array}{c} e_1 \\ \hdashline e_2 \\ \hdashline e_3 \\ \hdashline i_1 \\ \hdashline i_2 \end{array}\right] = \left[\begin{array}{c} \\ \hdashline \\ \hdashline \\ \hdashline \\ \hdashline \\ \end{array}\right]$$

Modified Nodal Analysis of a linear circuit



$$
\begin{array}{l}
\text{KCL at 1} \\
\text{KCL at 2} \\
\text{KCL at 3} \\
\text{BCE for VS1} \\
\text{BCE for VS2}
\end{array}
\left[\begin{array}{c:c:c:c:c}
G_1+G_2 & -G_2 & & & \\ \hdashline
-G_2 & G_2+G_3 & -G_3 & 1 & \\ \hdashline
 & -G_3 & G_3 & & \\ \hdashline
 & 1 & & & \\ \hdashline
 & & & &
\end{array}\right]
\left[\begin{array}{c}
e_1 \\ \hdashline e_2 \\ \hdashline e_3 \\ \hdashline i_1 \\ \hdashline i_2
\end{array}\right]
=
\left[\begin{array}{c}
\\ \hdashline \\ \hdashline \\ \hdashline V_0 \\ \hdashline \\
\end{array}\right]
$$

Modified Nodal Analysis of a linear circuit



$$
\begin{array}{l}
\text{KCL at 1} \\
\text{KCL at 2} \\
\text{KCL at 3} \\
\text{BCE for VS1} \\
\text{BCE for VS2}
\end{array}
\left[\begin{array}{c:c:c:c:c}
G_1+G_2 & -G_2 & & & \\ \hdashline
-G_2 & G_2+G_3 & -G_3 & 1 & \\ \hdashline
 & -G_3 & G_3 & & 1 \\ \hdashline
 & 1 & & & \\ \hdashline
-\alpha & \alpha & 1 & &
\end{array}\right]
\left[\begin{array}{c}
e_1 \\ \hdashline e_2 \\ \hdashline e_3 \\ \hdashline i_1 \\ \hdashline i_2
\end{array}\right]
=
\left[\begin{array}{c}
\\ \hdashline \\ \hdashline \\ \hdashline V_0 \\ \hdashline \\
\end{array}\right]
$$

Outline

* Circuit simulation: introduction
* Nodal analysis
* Modified nodal analysis
* Sparse tableau approach
* Nonlinear circuits
* Transient (dynamic) analysis

Sparse Tableau Analysis (STA)

* Variables: node voltages, branch currents, and branch voltages

Sparse Tableau Analysis (STA)

* Variables: node voltages, branch currents, and branch voltages
* No need for special treatment of voltage sources or any other elements

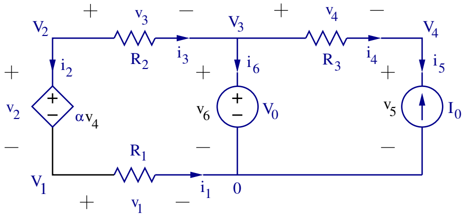
Sparse Tableau Analysis (STA)

* Variables: node voltages, branch currents, and branch voltages
* No need for special treatment of voltage sources or any other elements
* Circuit topology and element equations are decoupled.

Sparse Tableau Analysis (STA)

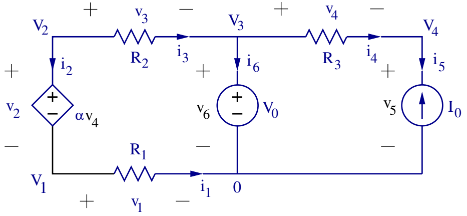
* Variables: node voltages, branch currents, and branch voltages
* No need for special treatment of voltage sources or any other elements
* Circuit topology and element equations are decoupled.
* Easier to implement as compared to MNA

STA example



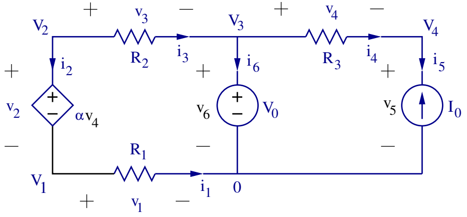
$$
\left[\begin{array}{cccccc|cccccc|cccc}
1 & -1 & & & & & & & & & & & & & & \\
 & 1 & 1 & & & & & & & & & 0 & & & & \\
 & & -1 & 1 & & 1 & & & & & & & & & & \\
 & & & -1 & 1 & & & & & & & & & & & \\
\hline
 & & & & & & 1 & & & & & & -1 & & & \\
 & & & & & & & 1 & & & & & 1 & -1 & & \\
 & & 0 & & & & & & 1 & & & & & -1 & 1 & \\
 & & & & & & & & & 1 & & & & & -1 & 1 \\
 & & & & & & & & & & 1 & & & & & -1 \\
 & & & & & & & & & & & 1 & & & -1 & \\
\hline
-R_1 & & & & & & 1 & & & & & & & & & \\
 & & & & & & & 1 & & -\alpha & & & & & & \\
 & & -R_2 & & & & & & 1 & & & & & & & \\
 & & & -R_3 & & & & & & 1 & & & & 0 & & \\
 & & & & 1 & & & & & & & & & & & \\
 & & & & & & & & & & & 1 & & & &
\end{array}\right]
\left[\begin{array}{c}
i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \\ i_6 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ V_1 \\ V_2 \\ V_3 \\ V_4
\end{array}\right]
=
\left[\begin{array}{c}
\cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ -I_0 \\ V_0
\end{array}\right]
$$

STA example



$$
\begin{array}{c}
\\
\text{KCL} \\
\\
\\
\\
\\
\\
\\
\\
\\
\\
\\
\\
\\
\\
\\
\end{array}
\left[\begin{array}{cccccc|cccccc|cccc}
1 & -1 & & & & & & & & & & & & & & \\
 & 1 & 1 & & & & & & & & 0 & & & & & \\
 & & -1 & 1 & & 1 & & & & & & & & & & \\
 & & & -1 & 1 & & & & & & & & & & & \\
\hline
 & & & & & & 1 & & & & & & -1 & & & \\
 & & & & & & & 1 & & & & & 1 & -1 & & \\
 & & & & & & & & 1 & & & & & -1 & 1 & \\
 & & 0 & & & & & & & 1 & & & & & -1 & 1 \\
 & & & & & & & & & & 1 & & & & & -1 \\
 & & & & & & & & & & & 1 & & & -1 & \\
\hline
-R_1 & & & & & & 1 & & & & & & & & & \\
 & & & & & & & 1 & & -\alpha & & & & & & \\
 & & -R_2 & & & & & & 1 & & & & & & & \\
 & & & -R_3 & & & & & & 1 & & & & 0 & & \\
 & & & & 1 & & & & & & & & & & & \\
 & & & & & & & & & & & 1 & & & &
\end{array}\right]
\left[\begin{array}{c}
i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \\ i_6 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ V_1 \\ V_2 \\ V_3 \\ V_4
\end{array}\right]
=
\left[\begin{array}{c}
\\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ -I_0 \\ V_0
\end{array}\right]
$$

STA example

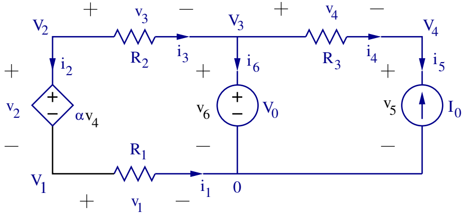


KCL

Branch Voltage Definition

$$
\left[\begin{array}{cccccc|cccccc|cccc}
1 & -1 & & & & & & & & & & & & & & \\
 & 1 & 1 & & & & & & & & 0 & & & & & \\
 & & -1 & 1 & & 1 & & & & & & & & & & \\
 & & & -1 & 1 & & & & & & & & & & & \\
\hline
 & & & & & & 1 & & & & & & -1 & & & \\
 & & & & & & & 1 & & & & & 1 & -1 & & \\
 & & & 0 & & & & & 1 & & & & & -1 & 1 & \\
 & & & & & & & & & 1 & & & & & -1 & 1 \\
 & & & & & & & & & & 1 & & & & & -1 \\
 & & & & & & & & & & & 1 & & & -1 & \\
\hline
-R_1 & & & & & & 1 & & & & & & & & & \\
 & & & & & & & 1 & & -\alpha & & & & & & \\
 & & -R_2 & & & & & & 1 & & & & & & & \\
 & & & -R_3 & & & & & & 1 & & & & & 0 & \\
 & & & & 1 & & & & & & & & & & & \\
 & & & & & & & & & & & 1 & & & &
\end{array}\right]
\left[\begin{array}{c}
i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \\ i_6 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ V_1 \\ V_2 \\ V_3 \\ V_4
\end{array}\right]
=
\left[\begin{array}{c}
 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ -I_0 \\ V_0
\end{array}\right]
$$

STA example



$$
\begin{array}{l}
\text{KCL} \\ \\ \\ \\ \text{Branch Voltage Definition} \\ \\ \\ \\ \text{Branch Equations}
\end{array}
\quad
\left[\begin{array}{cccccc|cccccc|cccc}
1 & -1 & & & & & & & & & & & & & & \\
 & 1 & 1 & & & & & & & & 0 & & & & & \\
 & & -1 & 1 & & 1 & & & & & & & & & & \\
 & & & -1 & 1 & & & & & & & & & & & \\
\hline
 & & & & & & 1 & & & & & & -1 & & & \\
 & & & & & & & 1 & & & & & 1 & -1 & & \\
 & & & & & & & & 1 & & & & & -1 & 1 & \\
 & & 0 & & & & & & & 1 & & & & & -1 & 1 \\
 & & & & & & & & & & 1 & & & & & -1 \\
 & & & & & & & & & & & 1 & & & -1 & \\
\hline
-R_1 & & & & & & 1 & & & & & & & & & \\
 & & & & & & & 1 & & -\alpha & & & & & & \\
 & & -R_2 & & & & & & 1 & & & & & & & \\
 & & & -R_3 & & & & & & 1 & & & & 0 & & \\
 & & & & 1 & & & & & & & & & & & \\
 & & & & & & & & & & & 1 & & & &
\end{array}\right]
\begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \\ i_6 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix}
=
\begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ -I_0 \\ V_0 \end{bmatrix}
$$

Comparison of MNA and STA

* STA matrix is larger, but more sparse.

Comparison of MNA and STA

* STA matrix is larger, but more sparse.
* If $\mathbf{A}$ is an $N \times N$ matrix, the CPU time to solve $\mathbf{Ax} = \mathbf{b}$ is proportional to N^{α}, where α is 3 for a dense matrix and typically 1.5 to 2 for a sparse matrix.

Comparison of MNA and STA

* STA matrix is larger, but more sparse.
* If $\mathbf{A}$ is an $N \times N$ matrix, the CPU time to solve $\mathbf{Ax} = \mathbf{b}$ is proportional to N^{α}, where α is 3 for a dense matrix and typically 1.5 to 2 for a sparse matrix.
* STA is generally slower than MNA, but this is not a concern for relatively small problems (including many problems in power electronics).

Comparison of MNA and STA

* STA matrix is larger, but more sparse.
* If **A** is an $N \times N$ matrix, the CPU time to solve $\mathbf{Ax} = \mathbf{b}$ is proportional to N^{α}, where α is 3 for a dense matrix and typically 1.5 to 2 for a sparse matrix.
* STA is generally slower than MNA, but this is not a concern for relatively small problems (including many problems in power electronics).
* Historically, STA was the first systematic approach used for circuit simulation (ASTAP by IBM). SPICE, based on MNA, was developed subsequently at UC Berkeley.

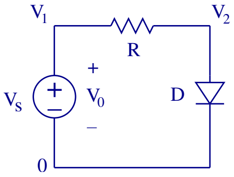
Comparison of MNA and STA

* STA matrix is larger, but more sparse.
* If $\mathbf{A}$ is an $N \times N$ matrix, the CPU time to solve $\mathbf{Ax} = \mathbf{b}$ is proportional to N^{α}, where α is 3 for a dense matrix and typically 1.5 to 2 for a sparse matrix.
* STA is generally slower than MNA, but this is not a concern for relatively small problems (including many problems in power electronics).
* Historically, STA was the first systematic approach used for circuit simulation (ASTAP by IBM). SPICE, based on MNA, was developed subsequently at UC Berkeley.
* Most of the circuit simulation programs available today are based on MNA, and many of them make use of SPICE.

Outline

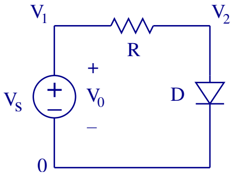
* Circuit simulation: introduction
* Nodal analysis
* Modified nodal analysis
* Sparse tableau approach
* Nonlinear circuits
* Transient (dynamic) analysis

Nonlinear circuits: Newton-Raphson method



[1]Note that a circuit simulator such as SPICE will use a combination of MNA and N-R to solve this problem. Here, we will reduce it to the form $f(x) = 0$ for simplicity.

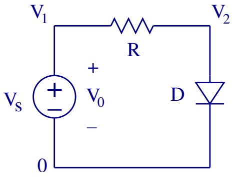
Nonlinear circuits: Newton-Raphson method



$$\frac{V_0 - V_2}{R} = I_s\,\left[\exp\left(V_2/V_T\right) - 1\right]$$

[1]Note that a circuit simulator such as SPICE will use a combination of MNA and N-R to solve this problem. Here, we will reduce it to the form $f(x) = 0$ for simplicity.

Nonlinear circuits: Newton-Raphson method

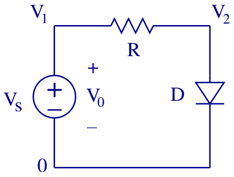


$$\frac{V_0 - V_2}{R} = I_s\,[\exp\,(V_2/V_T) - 1]$$

$$\frac{V_0 - V_2}{R} - I_s\,[\exp\,(V_2/V_T) - 1] = 0$$

[1]Note that a circuit simulator such as SPICE will use a combination of MNA and N-R to solve this problem. Here, we will reduce it to the form $f(x) = 0$ for simplicity.

Nonlinear circuits: Newton-Raphson method



$$\frac{V_0 - V_2}{R} = I_s\,[\exp{(V_2/V_T)} - 1]$$

$$\frac{V_0 - V_2}{R} - I_s\,[\exp{(V_2/V_T)} - 1] = 0$$

Rewrite[1] as $f(V_2) = 0$. In general, consider $f(x) = 0$. Expand around an initial guess x_0.

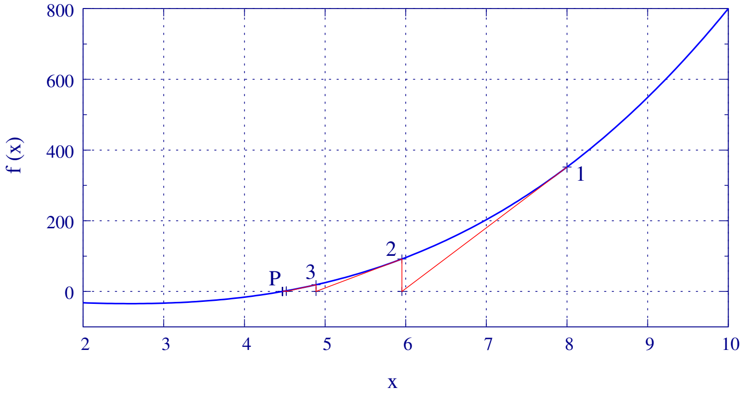
$$f(x_0 + \Delta x) = f(x_0) + \Delta x\, f'(x_0) + \cdots$$

We want Δx such that $f(x_0 + \Delta x) = 0$.

$$\Delta x = -\frac{f(x_0)}{f'(x_0)}$$

[1]Note that a circuit simulator such as SPICE will use a combination of MNA and N-R to solve this problem. Here, we will reduce it to the form $f(x) = 0$ for simplicity.

Newton-Raphson method: graphical interpretation of $\Delta x = -\frac{f(x_0)}{f'(x_0)}$



Solution of $x^3 - 20\,x = 0$, with $x = 8$ as the initial guess.

Newton-Raphson method: convergence

i	$x^{(i)}$	$f(x^{(i)})$	$\Delta x^{(i)}$
1	0.800000×10^{1}	0.352×10^{3}	-0.204×10^{1}
2	0.595349×10^{1}	0.919×10^{2}	-0.106×10^{1}
3	0.488846×10^{1}	0.190×10^{2}	-0.368
4	0.451992×10^{1}	0.194×10^{1}	-0.470×10^{-1}
5	0.447288×10^{1}	0.298×10^{-1}	-0.746×10^{-3}
6	0.447214×10^{1}	0.748×10^{-5}	-0.187×10^{-6}
7	0.447214×10^{1}	0.470×10^{-12}	-0.117×10^{-13}

Solution of $f(x) = x^3 - 20\,x = 0$, with $x = 8$ as the initial guess.

Convergence of Newton-Raphson method

Consider solving $f(x) = 0$ with the N-R method. Define

$$g(x) = x - \frac{f(x)}{f'(x)} \; . \tag{1}$$

Convergence of Newton-Raphson method

Consider solving $f(x) = 0$ with the N-R method. Define

$$g(x) = x - \frac{f(x)}{f'(x)} \; . \tag{1}$$

The N-R iteration can be written as [8],

$$x^{(n+1)} = x^{(n)} + \Delta x^{(n)} = g(x^{(n)}) \; . \tag{2}$$

Convergence of Newton-Raphson method

Consider solving $f(x) = 0$ with the N-R method. Define

$$g(x) = x - \frac{f(x)}{f'(x)} \; . \tag{1}$$

The N-R iteration can be written as [8],

$$x^{(n+1)} = x^{(n)} + \Delta x^{(n)} = g(x^{(n)}) \; . \tag{2}$$

Application of Taylor's theorem to Eq. 1 yields,

$$g(x) = g(r) + g'(r)(x-r) + \frac{g''(\xi)}{2}(x-r)^2, \tag{3}$$

where ξ lies between x and r.

Convergence of Newton-Raphson method

Consider solving $f(x) = 0$ with the N-R method. Define

$$g(x) = x - \frac{f(x)}{f'(x)} \; . \tag{1}$$

The N-R iteration can be written as [8],

$$x^{(n+1)} = x^{(n)} + \Delta x^{(n)} = g(x^{(n)}) \; . \tag{2}$$

Application of Taylor's theorem to Eq. 1 yields,

$$g(x) = g(r) + g'(r)(x-r) + \frac{g''(\xi)}{2}(x-r)^2, \tag{3}$$

where ξ lies between x and r.

The derivative $g'(x)$ can be obtained from Eq. 1 as,

$$g'(x) = 1 - \frac{[f'(x)]^2 - f(x)f''(x)}{[f'(x)]^2} \; . \tag{4}$$

Convergence of Newton-Raphson method

Since $f(r) = 0$, we get $g(r) = r$ from Eq. 1 and $g'(r) = 0$ from Eq. 4. Substituting for $g(r)$ and $g'(r)$ in Eq. 3, we get,

$$g(x) = r + \frac{g''(\xi)}{2}(x - r)^2 \ . \tag{5}$$

Convergence of Newton-Raphson method

Since $f(r) = 0$, we get $g(r) = r$ from Eq. 1 and $g'(r) = 0$ from Eq. 4. Substituting for $g(r)$ and $g'(r)$ in Eq. 3, we get,

$$g(x) = r + \frac{g''(\xi)}{2}(x - r)^2 \ . \tag{5}$$

Replace x by $x^{(n)}$ and use the fact that $g(x^{(n)})$ is the same as $x^{(n+1)}$ in the N-R procedure, to get

$$\left(x^{(n+1)} - r\right) = \frac{g''(\xi)}{2}\left(x^{(n)} - r\right)^2 \ . \tag{6}$$

Convergence of Newton-Raphson method

Since $f(r)=0$, we get $g(r)=r$ from Eq. 1 and $g'(r)=0$ from Eq. 4. Substituting for $g(r)$ and $g'(r)$ in Eq. 3, we get,

$$g(x) = r + \frac{g''(\xi)}{2}(x-r)^2 \ . \tag{5}$$

Replace x by $x^{(n)}$ and use the fact that $g(x^{(n)})$ is the same as $x^{(n+1)}$ in the N-R procedure, to get

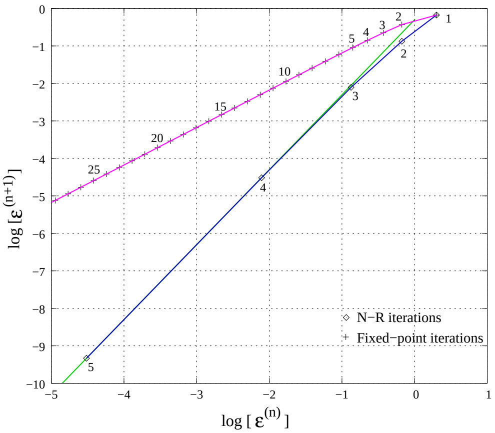
$$\left(x^{(n+1)} - r\right) = \frac{g''(\xi)}{2}\left(x^{(n)} - r\right)^2 \ . \tag{6}$$

As $x^{(n)}$ converges to r, so does ξ; and we can replace $g''(\xi)$ by $g''(r)$, a constant. Further, if we define $\epsilon^{(n)} \equiv x^{(n)} - r$ (the "error" at the n^{th} N-R iteration), we can write Eq. 6 as

$$\epsilon^{(n+1)} = k\,[\epsilon^{(n)}]^2 \, , \tag{7}$$

where $k = g''(r)/2$. Eq. 7 describes the well-known feature of "quadratic convergence" of the N-R method, i.e., the error goes down quadratically as $x^{(n)} \rightarrow \ r$.

Convergence of Newton-Raphson method



$\log(\epsilon^{(n+1)})$ versus $\log(\epsilon^{(n)})$ with the N-R scheme and the fixed-point iteration method for $f(x) = x^2 - 6x + 8 = 0$, with $x = 0$ as the initial guess. The green line represents $\epsilon^{(n+1)} = \dfrac{g''(r)}{2}(\epsilon^{(n)})^2$. The iteration numbers are also shown for each scheme. Note the quadratic convergence of the N-R method. (Both schemes were found to converge to $r = 2$ for the specified initial guess.)

Newton-Raphson method for N equations

Consider a system of N ODEs:

$$
\begin{aligned}
f_1(x_1, x_2, \ . \ . \ , x_N) &= 0 \ , \\
f_2(x_1, x_2, \ . \ . \ , x_N) &= 0 \ , \\
. \ . \ . \ . \qquad & \qquad , \\
f_N(x_1, x_2, \ . \ . \ , x_N) &= 0 \ .
\end{aligned}
$$

The correction vector $\Delta\mathbf{x}$ can be obtained by solving

$$\mathbf{J}^{(i)}\,\Delta\mathbf{x}^{(i)} = -\mathbf{f}^{(i)}\,,$$

where i is the iteration number, $\mathbf{J}$ is the Jacobian matrix, and $\mathbf{f}$ is the function vector.

$$\mathbf{f} = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ .. \\ f_N(\mathbf{x}) \end{bmatrix},$$

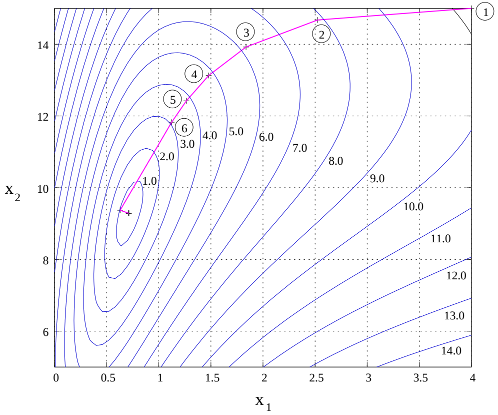
$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} & .. & \dfrac{\partial f_1}{\partial x_N} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} & .. & \dfrac{\partial f_2}{\partial x_N} \\ .. & .. & .. & \\ \dfrac{\partial f_N}{\partial x_1} & \dfrac{\partial f_N}{\partial x_2} & .. & \dfrac{\partial f_N}{\partial x_N} \end{bmatrix}.$$

N-R method: example with two variables

i	$x_1^{(i)}$	$x_2^{(i)}$	$\lVert f \rVert_2$	$\Delta x_1^{(i)}$	$\Delta x_2^{(i)}$
1	0.40000×10^1	0.15000×10^2	0.10241×10^2	-0.73776×10^1	-0.16223×10^1
2	0.25244×10^1	0.14675×10^2	0.78909×10^1	-0.34368×10^1	-0.37631×10^1
3	0.18371×10^1	0.13922×10^2	0.61523×10^1	-0.17887×10^1	-0.39712×10^1
4	0.14793×10^1	0.13128×10^2	0.48512×10^1	-0.10737×10^1	-0.35342×10^1
5	0.12646×10^1	0.12421×10^2	0.38481×10^1	-0.70747	-0.29789×10^1
6	0.11231×10^1	0.11826×10^2	0.30620×10^1	-0.49427	-0.24548×10^1
7	0.62883	0.93711×10^1	0.95091	0.80932×10^{-1}	-0.80932×10^{-1}
8	0.70976	0.92902×10^1	0.31487×10^{-1}	0.28690×10^{-2}	-0.28690×10^{-2}
9	0.71263	0.92873×10^1	0.38735×10^{-4}	0.35381×10^{-5}	-0.35381×10^{-5}
10	0.71263	0.92873×10^1	0.58855×10^{-10}	0.53759×10^{-11}	-0.53753×10^{-11}

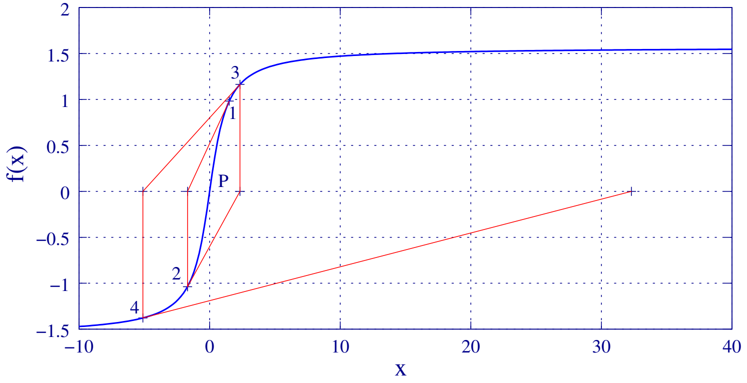
Application of the N-R method to a system of two equations, with $f_1 \equiv x_1 + x_2 - 10 = 0$, and $f_2 \equiv x_2 - 15 \tan^{-1}(x_1) = 0$. (damping was used for the first 5 iterations.)

N-R method: example with two variables



Application of the N-R method to a system of two equations, with $f_1 \equiv x_1 + x_2 - 10 = 0$, and $f_2 \equiv x_2 - 15\tan^{-1}(x_1) = 0$. The contours are labelled by the 2-norm, $\|\, f \,\|_2$. Circled integers represent the iteration numbers. (damping was used for the first 5 iterations.)

Newton-Raphson method: convergence issues



Application of the N-R method to $f(x) = \tan^{-1} x = 0$, with $x = 1.5$ as the initial guess.

Newton-Raphson method: use of damping

Instead of

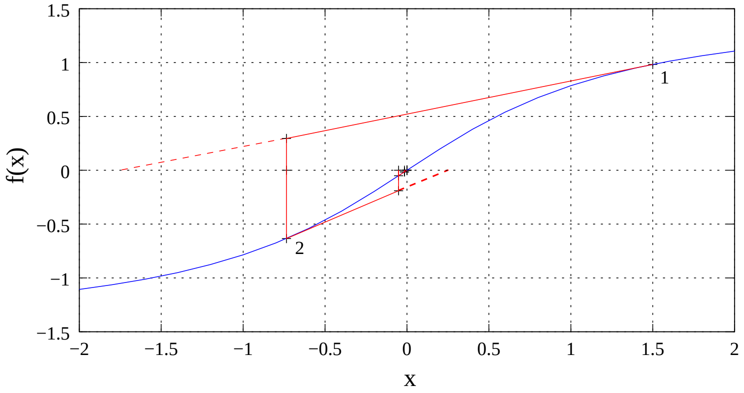
$$x^{(n+1)} = x^{(n)} + \Delta x^{(n)} \, ,$$

as in the standard N-R algorithm, we use

$$\begin{aligned} x^{(n+1)} &= x^{(n)} + k \, \Delta x^{(n)} \\ &= x^{(n)} + k \left\{ -[f'(x^{(n)})]^{-1} \, f(x^{(n)}) \right\} \, , \end{aligned}$$

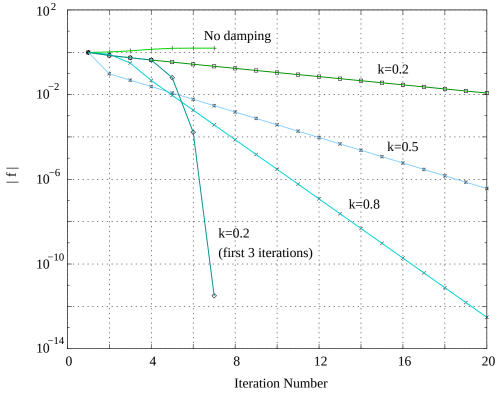
where k (< 1) is the "damping factor."

Newton-Raphson method: use of damping



Application of the N-R method to $f(x) = \tan^{-1} x = 0$, with $x = 1.5$ as the initial guess and a damping factor $k = 0.7$.

Newton-Raphson method: use of damping



Application of the N-R method to $f(x) = \tan^{-1} x = 0$, with $x = 1.5$ as the initial guess and different damping factors. (For the case with no damping, N-R iterations stopped due to $\dfrac{df}{dx}$ becoming too small.)

Newton-Raphson method: use of damping

∗ Damping improves chances of convergence.

Newton-Raphson method: use of damping

* Damping improves chances of convergence.
* However, it makes convergence slower as compared to the standard N-R method.

Newton-Raphson method: use of damping

* Damping improves chances of convergence.
* However, it makes convergence slower as compared to the standard N-R method.
* Damping should be used only when the standard N-R method fails to converge.

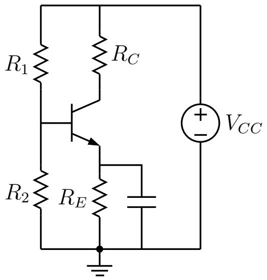
Newton-Raphson method: use of damping

* Damping improves chances of convergence.
* However, it makes convergence slower as compared to the standard N-R method.
* Damping should be used only when the standard N-R method fails to converge.
* Damping is very useful in power electronic circuits since they are highly non-linear (due to switches).

Newton-Raphson method: use of damping

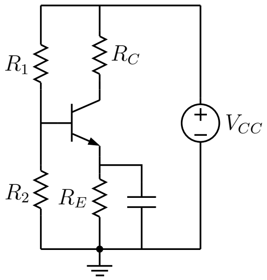
* Damping improves chances of convergence.
* However, it makes convergence slower as compared to the standard N-R method.
* Damping should be used only when the standard N-R method fails to converge.
* Damping is very useful in power electronic circuits since they are highly non-linear (due to switches).
* For transient simulation, in addition to damping, reducing the time step may also help in convergence.

Convergence of N-R iterations



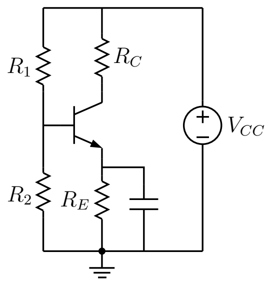
∗ We are interested in obtaining the DC (“bias”) solution for a circuit with highly non-linear elements (e.g., BJTs).

Convergence of N-R iterations



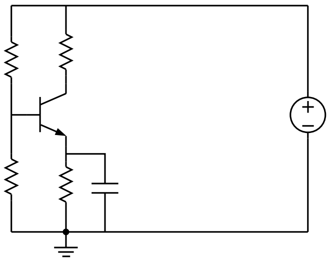
* We are interested in obtaining the DC ("bias") solution for a circuit with highly non-linear elements (e.g., BJTs).
* N-R iterations, starting from the zero solution (i.e., all node voltages equal to 0 V), may fail to converge in this case.

Convergence of N-R iterations

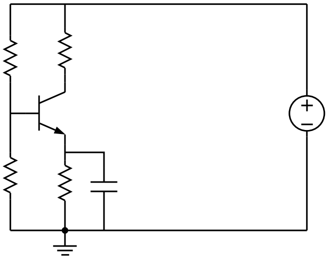


- We are interested in obtaining the DC ("bias") solution for a circuit with highly non-linear elements (e.g., BJTs).
- N-R iterations, starting from the zero solution (i.e., all node voltages equal to 0 V), may fail to converge in this case.
- Two tricks: (a) g_{min} stepping, (b) V_{CC} stepping.

$g_{\min}$ stepping

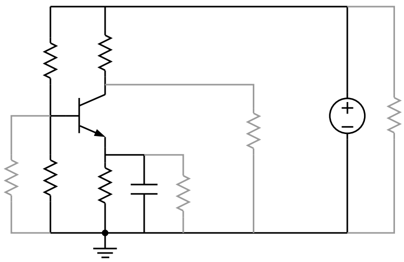


g_{min} stepping



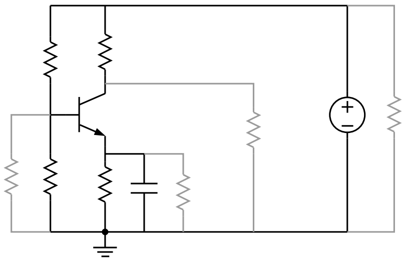
* Connect $R = 1/g$ between each node and ground.

$g_{\min}$ stepping



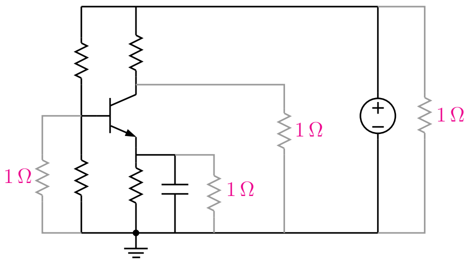
* Connect $R = 1/g$ between each node and ground.

$g_{\min}$ stepping



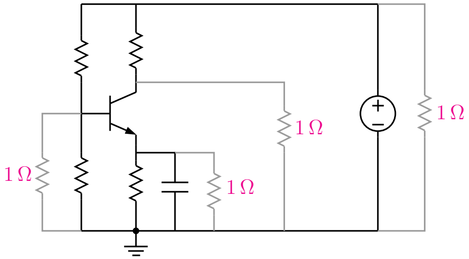
- Connect $R = 1/g$ between each node and ground.
- Assign a small value (say, $1\,\Omega$) to each resistance, i.e., a large value to g ($1\,\mho$).

g_{min} stepping



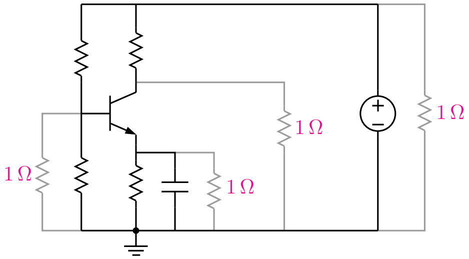
- Connect $R = 1/g$ between each node and ground.
- Assign a small value (say, 1 Ω) to each resistance, i.e., a large value to g (1 ℧).

g_{min} stepping

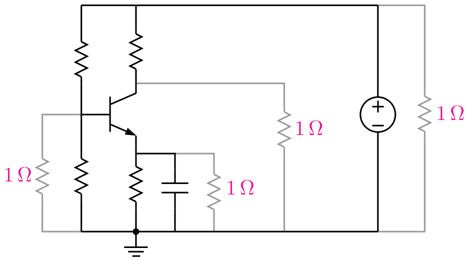


* Connect $R = 1/g$ between each node and ground.
* Assign a small value (say, $1\,\Omega$) to each resistance, i.e., a large value to g ($1\,\mho$).
  $\rightarrow$ easy convergence since the non-linear elements got bypassed.

$g_{\min}$ stepping

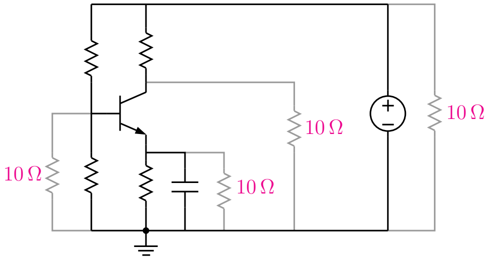


g_{min} stepping



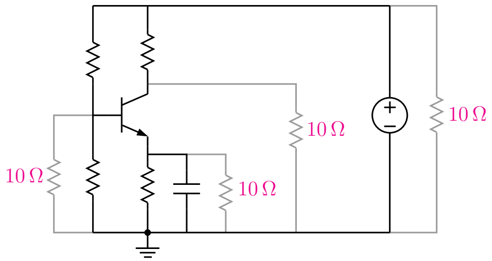
∗ Increase R from, say, $1\,\Omega$ to $10\,\Omega$, i.e., decrease g from $1\,\mho$ to $0.1\,\mho$.

$g_{\min}$ stepping



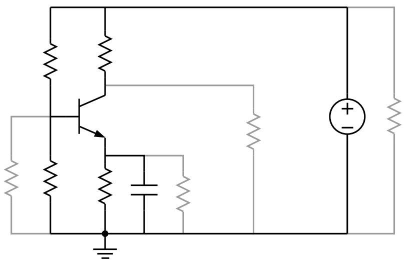
∗ Increase R from, say, $1\,\Omega$ to $10\,\Omega$, i.e., decrease g from $1\,\mho$ to $0.1\,\mho$.

g_{min} stepping

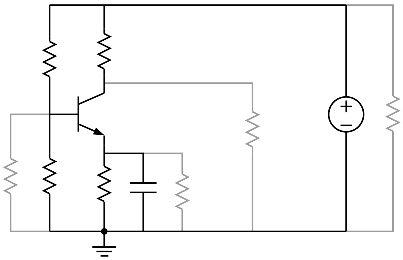


* Increase R from, say, $1\,\Omega$ to $10\,\Omega$, i.e., decrease g from $1\,\mho$ to $0.1\,\mho$.
* Convergence is easy since the previous solution serves as a good initial guess.

$g_{\min}$ stepping

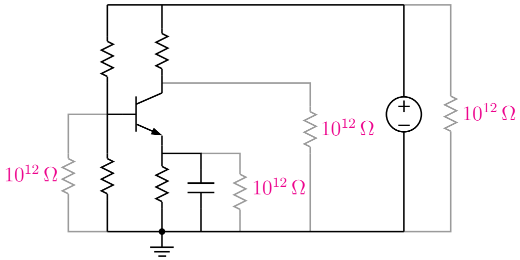


$g_{\min}$ stepping



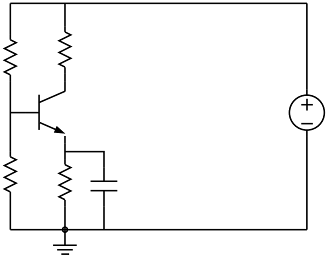
* Keep increasing R (i.e., decreasing g) and solve every time.

g_{min} stepping



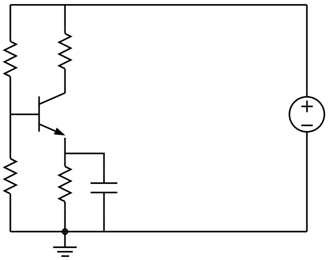
* Keep increasing R (i.e., decreasing g) and solve every time.
* When $g = 10^{-12}\,\mho$, for example, $R = 10^{12}\,\Omega$, which is as good as an open circuit.

g_{min} stepping



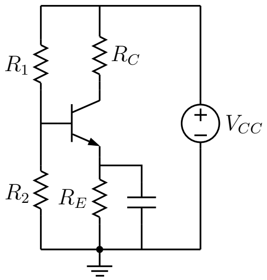
* Keep increasing R (i.e., decreasing g) and solve every time.
* When $g = 10^{-12}\,℧$, for example, $R = 10^{12}\,\Omega$, which is as good as an open circuit.

g_{min} stepping



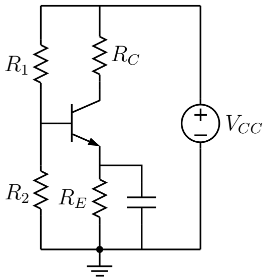
- Keep increasing R (i.e., decreasing g) and solve every time.
- When $g = 10^{-12}\,\mho$, for example, $R = 10^{12}\,\Omega$, which is as good as an open circuit.
- We have now got the DC solution for the original circuit.

Voltage supply stepping



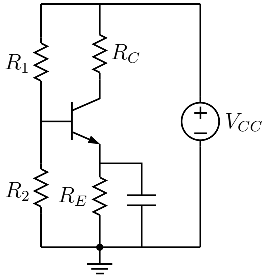
∗ When $V_{CC} = 0\,\text{V}$, the zero initial solution (all node voltages equal to $0\,\text{V}$) is valid.

Voltage supply stepping



* When $V_{CC} = 0\,\text{V}$, the zero initial solution (all node voltages equal to $0\,\text{V}$) is valid.
* Treating that as the initial guess, solve for a small value of V_{CC} (say, $0.1\,\text{V}$). The N-R iterations are likely to converge since $V_{CC} = 0.1\,\text{V}$ is a small change from $V_{CC} = 0\,\text{V}$.

Voltage supply stepping



* When $V_{CC} = 0\,\text{V}$, the zero initial solution (all node voltages equal to $0\,\text{V}$) is valid.
* Treating that as the initial guess, solve for a small value of V_{CC} (say, $0.1\,\text{V}$). The N-R iterations are likely to converge since $V_{CC} = 0.1\,\text{V}$ is a small change from $V_{CC} = 0\,\text{V}$.
* Repeat. V_{CC} : $0\,\text{V} \rightarrow 0.1\,\text{V} \rightarrow 0.2\,\text{V} \rightarrow \cdots \rightarrow 5\,\text{V}$

N-R method: effect of scaling and precision

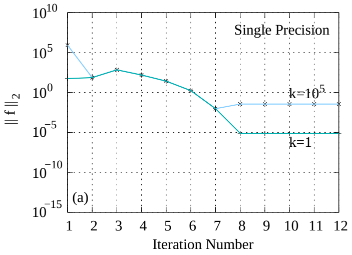
Consider the system of equations,

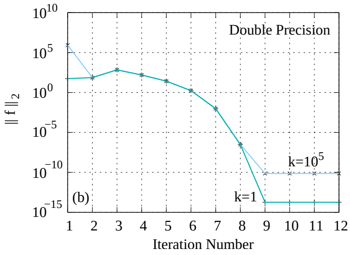
$$\begin{aligned} f_1(x_1, x_2) &\equiv k\,(x_1 + x_2 - 6\sqrt{3}) = 0\ , \\ f_2(x_1, x_2) &\equiv 10x_1^2 - x_2^2 + 45 = 0\ . \end{aligned} \qquad (8)$$

N-R method: effect of scaling and precision

Consider the system of equations,

$$\begin{aligned} f_1(x_1,x_2) &\equiv k\,(x_1+x_2-6\sqrt{3})=0\ , \\ f_2(x_1,x_2) &\equiv 10x_1^2-x_2^2+45=0\ . \end{aligned} \tag{8}$$



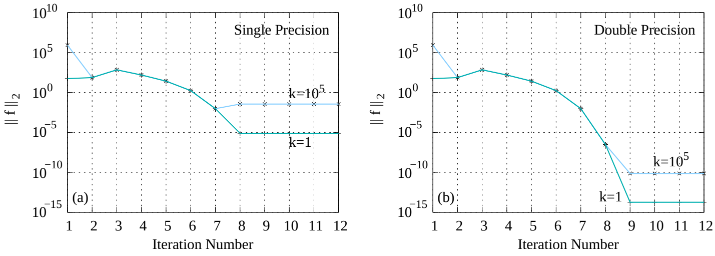


$||\,f\,||_2$ versus N-R iteration number for Eq. 8, with $x_1=x_2=1$ as the initial guess, (a) Single precision arithmetic, (b) Double precision arithmetic.

N-R method: effect of scaling and precision

Consider the system of equations,

$$\begin{aligned} f_1(x_1,x_2) &\equiv k\,(x_1+x_2-6\sqrt{3})=0\ , \\ f_2(x_1,x_2) &\equiv 10x_1^2-x_2^2+45=0\ . \end{aligned} \tag{8}$$



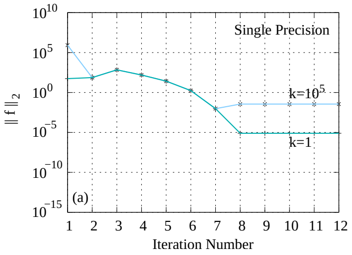
$||\ f\ ||_2$ versus N-R iteration number for Eq. 8, with $x_1 = x_2 = 1$ as the initial guess, (a) Single precision arithmetic, (b) Double precision arithmetic.

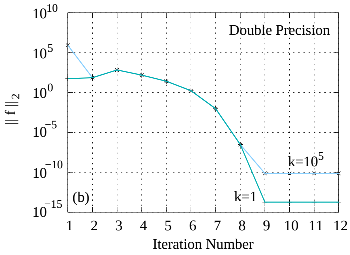
$*$ If k is made larger, the norm saturates at a higher value.

N-R method: effect of scaling and precision

Consider the system of equations,

$$\begin{aligned} f_1(x_1,x_2) &\equiv k\,(x_1+x_2-6\sqrt{3})=0\ , \\ f_2(x_1,x_2) &\equiv 10x_1^2-x_2^2+45=0\ . \end{aligned} \qquad (8)$$

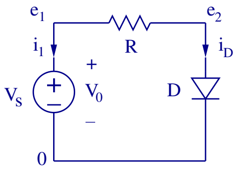




$||\,f\,||_2$ versus N-R iteration number for Eq. 8, with $x_1=x_2=1$ as the initial guess, (a) Single precision arithmetic, (b) Double precision arithmetic.

* If k is made larger, the norm saturates at a higher value.
* Precision has a significant effect on the lowest achievable norm.

Non-linear circuit analysis



MNA equations:

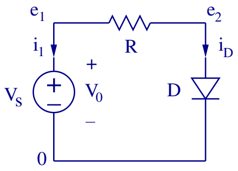
$$
\begin{aligned}
i_1 + G(e_1 - e_2) &= 0\,,\\
G(e_2 - e_1) + i_D(e_2) &= 0\,,\\
e_1 &= V_0\,,
\end{aligned}
$$

where

$$i_D(e_2) = I_{s0}\,\left[\exp\left(e_2/V_T\right) - 1\right]\,.$$

* The circuit equations can be assembled using the MNA or STA approach.

Non-linear circuit analysis



MNA equations:

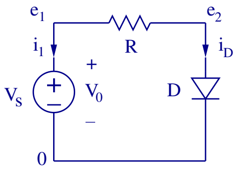
$$\begin{aligned} i_1 + G(e_1 - e_2) &= 0\,, \\ G(e_2 - e_1) + i_D(e_2) &= 0\,, \\ e_1 &= V_0\,, \end{aligned}$$

where

$$i_D(e_2) = I_{s0}\,\left[\exp\left(e_2/V_T\right) - 1\right]\,.$$

* The circuit equations can be assembled using the MNA or STA approach.
* Since the equations are non-linear, the N-R method is used to solve them.

Non-linear circuit analysis



MNA equations:

$$\begin{aligned} i_1 + G(e_1 - e_2) &= 0\,, \\ G(e_2 - e_1) + i_D(e_2) &= 0\,, \\ e_1 &= V_0\,, \end{aligned}$$

where

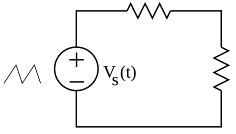
$$i_D(e_2) = I_{s0}\,[\exp(e_2/V_T) - 1]\,.$$

- The circuit equations can be assembled using the MNA or STA approach.
- Since the equations are non-linear, the N-R method is used to solve them.
- More expensive than a linear circuit of the same size, since several (typically 3 to 5) N-R iterations are involved, each requiring the solution of $\mathbf{J}\boldsymbol{\Delta}\mathbf{x} = -\mathbf{f}$.

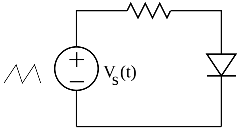
Outline

* Circuit simulation: introduction
* Nodal analysis
* Modified nodal analysis
* Sparse tableau approach
* Nonlinear circuits
* Transient (dynamic) analysis

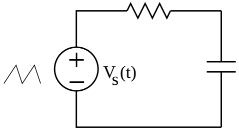
Transient (dynamic) analysis



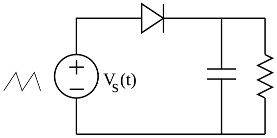
(a)



(b)

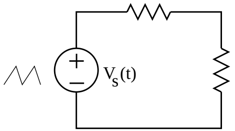


(c)

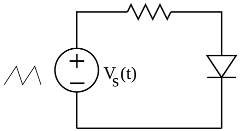


(d)

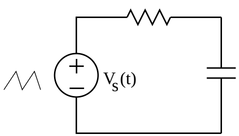
Transient (dynamic) analysis



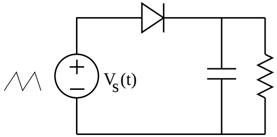
(a)



(b)



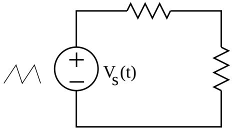
(c)



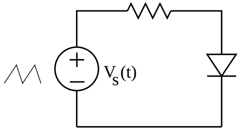
(d)

- In (a) and (b), we can use the techniques seen earlier. At a given time t, we simply need to replace the source with a DC source with voltage = $V_s(t)$.

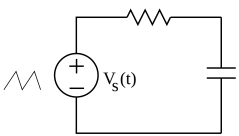
Transient (dynamic) analysis



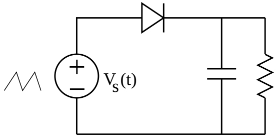
(a)



(b)



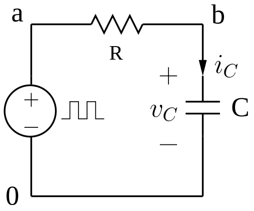
(c)



(d)

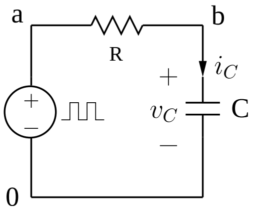
* In (a) and (b), we can use the techniques seen earlier. At a given time t, we simply need to replace the source with a DC source with voltage $= V_s(t)$.
* In (c) and (d), the situation is very different due to the presence of a capacitor which involves time derivatives.

Transient analysis



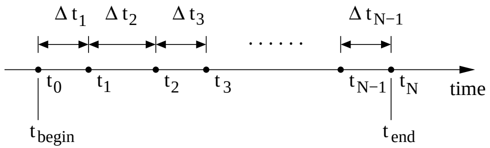
* The capacitor current, $i_C = C\,\dfrac{dv_C}{dt}$, cannot be written in terms of the instantaneous node voltages or branch voltages since its value depends on the past behaviour of v_C.

Transient analysis



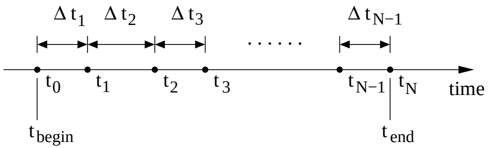
* The capacitor current, $i_C = C\,\dfrac{dv_C}{dt}$, cannot be written in terms of the instantaneous node voltages or branch voltages since its value depends on the past behaviour of v_C.
* We need some way of approximating the derivative in terms of the past behaviour of v_C.

Discretization of time



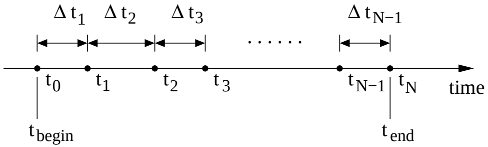
* Discretization of time is required since numerical solution can only be obtained at a finite number of points.

Discretization of time



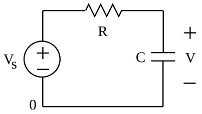
* Discretization of time is required since numerical solution can only be obtained at a finite number of points.
* The time steps (Δt_i) may not be uniform.

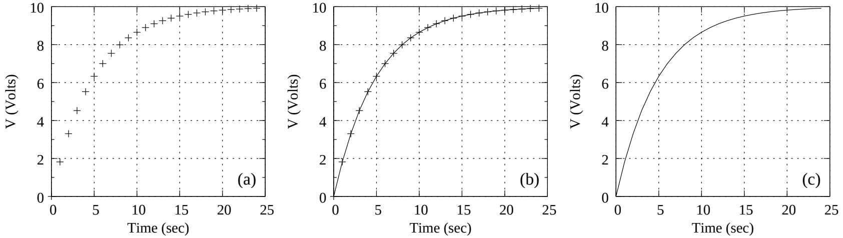
Discretization of time



* Discretization of time is required since numerical solution can only be obtained at a finite number of points.
* The time steps (Δt_i) may not be uniform.
* Generally, the time steps are computed *dynamically*, not *a priori*.

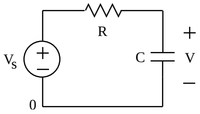
Discretization of time

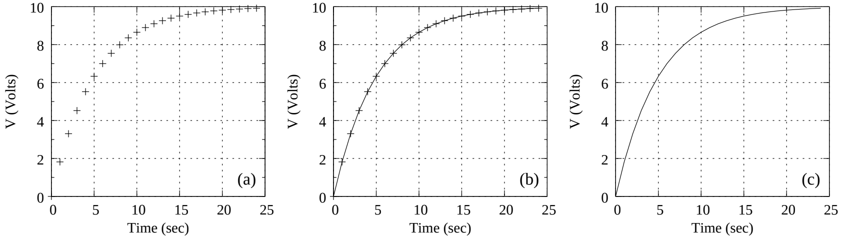




(a) Typical simulator output.

Discretization of time

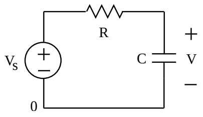


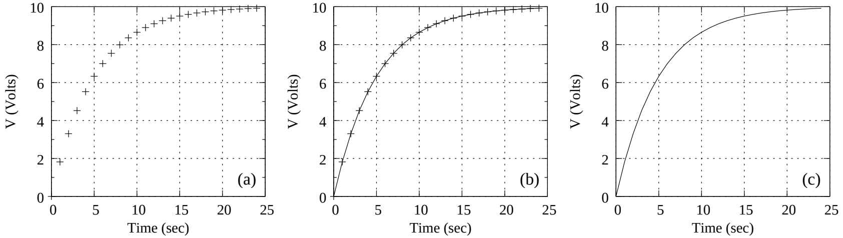


(a) Typical simulator output.

(b) After connecting the output points with line segments.

Discretization of time



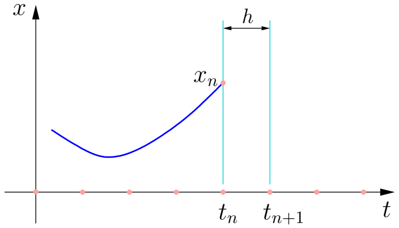


(a) Typical simulator output.

(b) After connecting the output points with line segments.

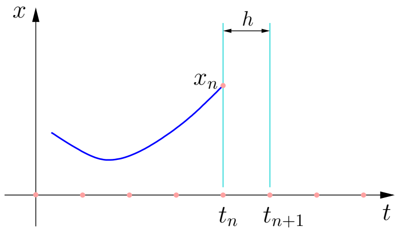
(c) After removing the output points (but retaining the segments), the waveform looks continuous, but this is an illusion!

Transient simulation: Forward Euler method



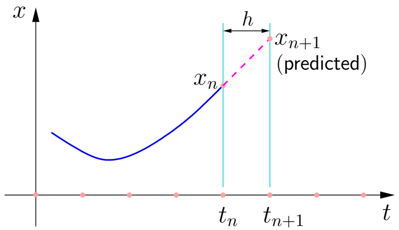
* Consider $\dfrac{dx}{dt} = f(t, x)$. We have the solution at t_n and want to obtain $x(t_{n+1})$.

Transient simulation: Forward Euler method



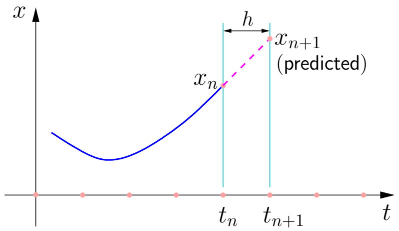
- Consider $\dfrac{dx}{dt} = f(t, x)$. We have the solution at t_n and want to obtain $x(t_{n+1})$.
- Compute the slope at t_n: $\left.\dfrac{dx}{dt}\right|_{t=t_n} = f(t_n, x_n)$.

Transient simulation: Forward Euler method



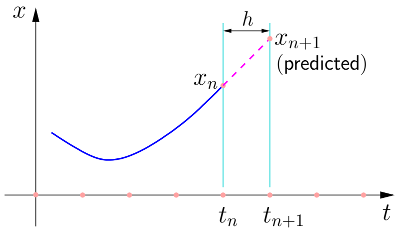
- Consider $\dfrac{dx}{dt} = f(t,x)$. We have the solution at t_n and want to obtain $x(t_{n+1})$.
- Compute the slope at t_n: $\left.\dfrac{dx}{dt}\right|_{t=t_n} = f(t_n, x_n)$.

Transient simulation: Forward Euler method



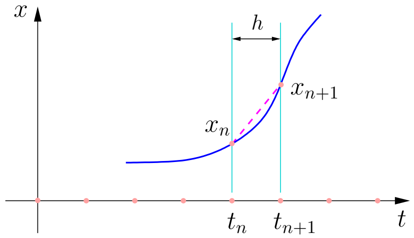
* Consider $\dfrac{dx}{dt} = f(t, x)$. We have the solution at t_n and want to obtain $x(t_{n+1})$.
* Compute the slope at t_n: $\left.\dfrac{dx}{dt}\right|_{t=t_n} = f(t_n, x_n)$.
* $\dfrac{x_{n+1} - x_n}{t_{n+1} - t_n} \approx f(t_n, x_n)$

Transient simulation: Forward Euler method



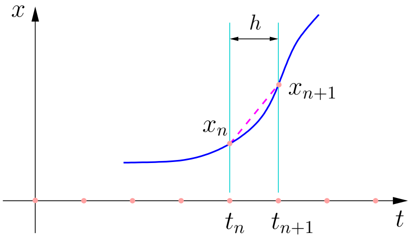
* Consider $\dfrac{dx}{dt} = f(t, x)$. We have the solution at t_n and want to obtain $x(t_{n+1})$.
* Compute the slope at t_n: $\left.\dfrac{dx}{dt}\right|_{t=t_n} = f(t_n, x_n)$.
* $\dfrac{x_{n+1} - x_n}{t_{n+1} - t_n} \approx f(t_n, x_n) \rightarrow x_{n+1} = x_n + h\, f(t_n, x_n)$.

Transient analysis: a quick look



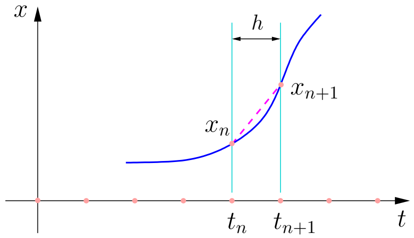
Method	Approximation for $\dfrac{dx}{dt} = f(t, x)$
Forward Euler	$\dfrac{x_{n+1} - x_n}{h} = f(t_n, x_n)$

Transient analysis: a quick look



Method	Approximation for $\frac{dx}{dt} = f(t,x)$
Forward Euler	$\frac{x_{n+1} - x_n}{h} = f(t_n, x_n)$
Backward Euler	$\frac{x_{n+1} - x_n}{h} = f(t_{n+1}, x_{n+1})$

Transient analysis: a quick look



Method	Approximation for $\dfrac{dx}{dt} = f(t,x)$
Forward Euler	$\dfrac{x_{n+1}-x_n}{h} = f(t_n, x_n)$
Backward Euler	$\dfrac{x_{n+1}-x_n}{h} = f(t_{n+1}, x_{n+1})$
Trapezoidal	$\dfrac{x_{n+1}-x_n}{h} = \dfrac{1}{2}\left[f(t_n, x_n) + f(t_{n+1}, x_{n+1})\right]$

Application to $\dot{x} = -x$, with $x(0) = 1$

$$\begin{aligned}
FE: &\quad \frac{x_{n+1} - x_n}{h} = f(t_n, x_n) && = -x_n \\
BE: &\quad \frac{x_{n+1} - x_n}{h} = f(t_{n+1}, x_{n+1}) && = -x_{n+1} \\
TRZ: &\quad \frac{x_{n+1} - x_n}{h} = \frac{1}{2}\left[f(t_n, x_n) + f(t_{n+1}, x_{n+1})\right] && = -\frac{1}{2}(x_n + x_{n+1})
\end{aligned}$$

Simple manipulation yields the following approximations:

$$\begin{aligned}
FE: &\quad x_{n+1} = x_n\,(1 - h) \\
BE: &\quad x_{n+1} = x_n\,\frac{1}{1+h} \\
TRZ: &\quad x_{n+1} = x_n\,\frac{1 - h/2}{1 + h/2}
\end{aligned}$$

Application to $\dot{x} = -x$, with $x(0) = 1$

The exact solution is $\hat{x}(t) = e^{-t}$. Expanding around t_n, we get,

$$\hat{x}_{n+1} = \hat{x}_n + h\,\frac{d\hat{x}}{dt} + \cdots = \hat{x}_n + h(-e^{-t_n}) + \cdots = \hat{x}_n(1 - h + h^2/2 - h^3/6 + \cdots)\,.$$

Application to $\dot{x} = -x$, with $x(0) = 1$

The exact solution is $\hat{x}(t) = e^{-t}$. Expanding around t_n, we get,

$$\hat{x}_{n+1} = \hat{x}_n + h\,\frac{d\hat{x}}{dt} + \cdots = \hat{x}_n + h(-e^{-t_n}) + \cdots = \hat{x}_n(1 - h + h^2/2 - h^3/6 + \cdots)\,.$$

Compare with

$$FE: \qquad x_{n+1} = x_n\,(1 - h)$$

Application to $\dot{x} = -x$, with $x(0) = 1$

The exact solution is $\hat{x}(t) = e^{-t}$. Expanding around t_n, we get,

$$\hat{x}_{n+1} = \hat{x}_n + h\,\frac{d\hat{x}}{dt} + \cdots = \hat{x}_n + h(-e^{-t_n}) + \cdots = \hat{x}_n(1 - h + h^2/2 - h^3/6 + \cdots)\,.$$

Compare with

$$\begin{aligned} FE: \quad x_{n+1} &= x_n\,(1-h) \\ BE: \quad x_{n+1} &= x_n\,\frac{1}{1+h} = x_n\,(1 - h + h^2 + \cdots) \end{aligned}$$

Application to $\dot{x} = -x$, with $x(0) = 1$

The exact solution is $\hat{x}(t) = e^{-t}$. Expanding around t_n, we get,

$$\hat{x}_{n+1} = \hat{x}_n + h\,\frac{d\hat{x}}{dt} + \cdots = \hat{x}_n + h(-e^{-t_n}) + \cdots = \hat{x}_n(1 - h + h^2/2 - h^3/6 + \cdots)\,.$$

Compare with

$$\begin{aligned}
FE: \quad & x_{n+1} &=& \; x_n\,(1-h) \\
BE: \quad & x_{n+1} &=& \; x_n\,\frac{1}{1+h} = x_n\,(1 - h + h^2 + \cdots) \\
TRZ: \quad & x_{n+1} &=& \; x_n\,\frac{1-h/2}{1+h/2} = x_n\,(1 - h + h^2/2 - h^3/4 + \cdots)
\end{aligned}$$

Application to $\dot{x} = -x$, with $x(0) = 1$

The exact solution is $\hat{x}(t) = e^{-t}$. Expanding around t_n, we get,

$$\hat{x}_{n+1} = \hat{x}_n + h\,\frac{d\hat{x}}{dt} + \cdots = \hat{x}_n + h(-e^{-t_n}) + \cdots = \hat{x}_n(1 - h + h^2/2 - h^3/6 + \cdots)\,.$$

Compare with

$$\begin{aligned} FE: &\quad x_{n+1} &=&\quad x_n\,(1-h) \\ BE: &\quad x_{n+1} &=&\quad x_n\,\frac{1}{1+h} = x_n\,(1 - h + h^2 + \cdots) \\ TRZ: &\quad x_{n+1} &=&\quad x_n\,\frac{1-h/2}{1+h/2} = x_n\,(1 - h + h^2/2 - h^3/4 + \cdots) \end{aligned}$$

$*$ If h $\ll 1$, the three approximations are equivalent, as we would expect.

Application to $\dot{x} = -x$, with $x(0) = 1$

The exact solution is $\hat{x}(t) = e^{-t}$. Expanding around t_n, we get,

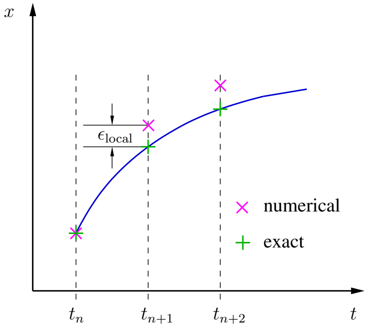
$$\hat{x}_{n+1} = \hat{x}_n + h\,\frac{d\hat{x}}{dt} + \cdots = \hat{x}_n + h(-e^{-t_n}) + \cdots = \hat{x}_n(1 - h + h^2/2 - h^3/6 + \cdots)\,.$$

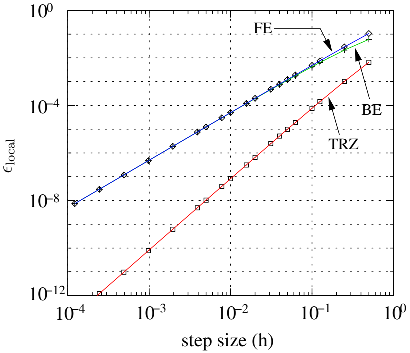
Compare with

$$\begin{aligned}
FE:&\quad x_{n+1} = x_n\,(1-h)\\
BE:&\quad x_{n+1} = x_n\,\frac{1}{1+h} = x_n\,(1 - h + h^2 + \cdots)\\
TRZ:&\quad x_{n+1} = x_n\,\frac{1-h/2}{1+h/2} = x_n\,(1 - h + h^2/2 - h^3/4 + \cdots)
\end{aligned}$$

- If h $\ll$ 1, the three approximations are equivalent, as we would expect.
- If the starting point $x(t_n)$ is the same, the "error" (difference between the exact and numerical solutions) is $O(h^2)$ for FE and BE, and $O(h^3)$ for TRZ.

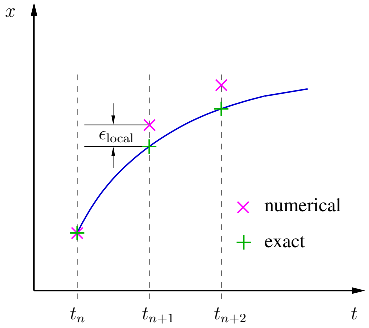
Application to $\dot{x} = -x$, with $x(0) = 1$

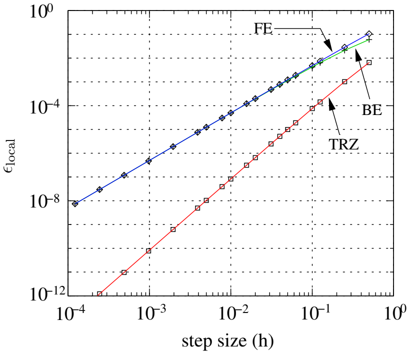




* The *local* error is the error made in a *single* step, assuming that the starting point is exact. In this case, starting from the exact value, $x(0) = 1$, the difference $|x(h) - \hat{x}(h)|$ has been computed.

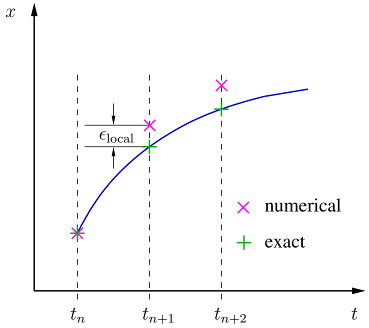
Application to $\dot{x} = -x$, with $x(0) = 1$

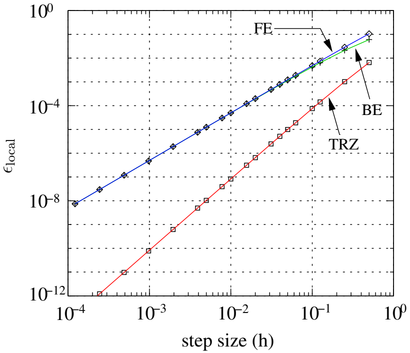




* The *local* error is the error made in a *single* step, assuming that the starting point is exact. In this case, starting from the exact value, $x(0) = 1$, the difference $|x(h) - \hat{x}(h)|$ has been computed.
* If $h \to h/10$, the error decreases by a factor of 10^2 for the FE and BE methods, and by 10^3 for the TRZ method.

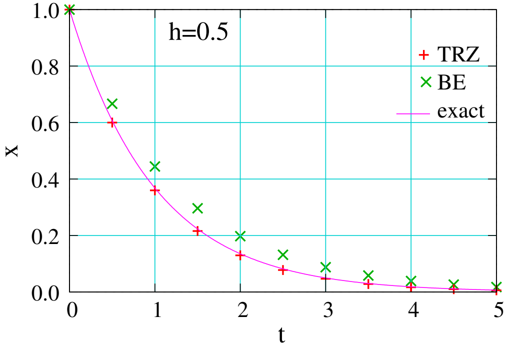
Application to $\dot{x} = -x$, with $x(0) = 1$



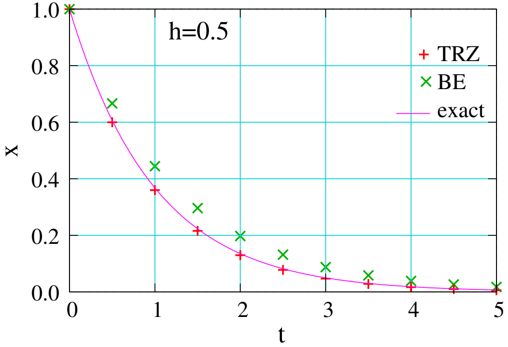


* The *local* error is the error made in a *single* step, assuming that the starting point is exact. In this case, starting from the exact value, $x(0) = 1$, the difference $|x(h) - \hat{x}(h)|$ has been computed.
* If $h \rightarrow h/10$, the error decreases by a factor of 10^2 for the FE and BE methods, and by 10^3 for the TRZ method.
* The TRZ method is therefore said to be more *accurate* than FE or BE.

Application to $\dot{x} = -x$, with $x(0) = 1$

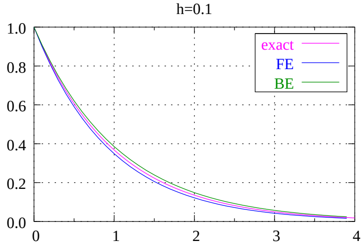


Application to $\dot{x} = -x$, with $x(0) = 1$

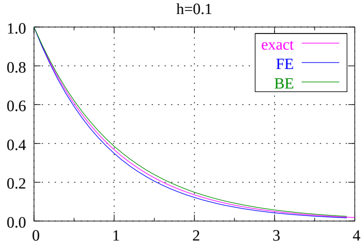


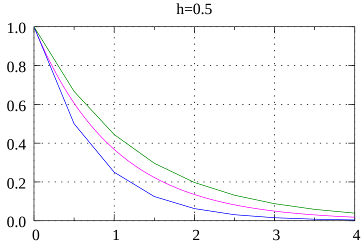
* The higher accuracy of the TRZ method allows larger time steps.

Comparison of FE and BE for $\dot{x} = -x, \quad x(0) = 1$

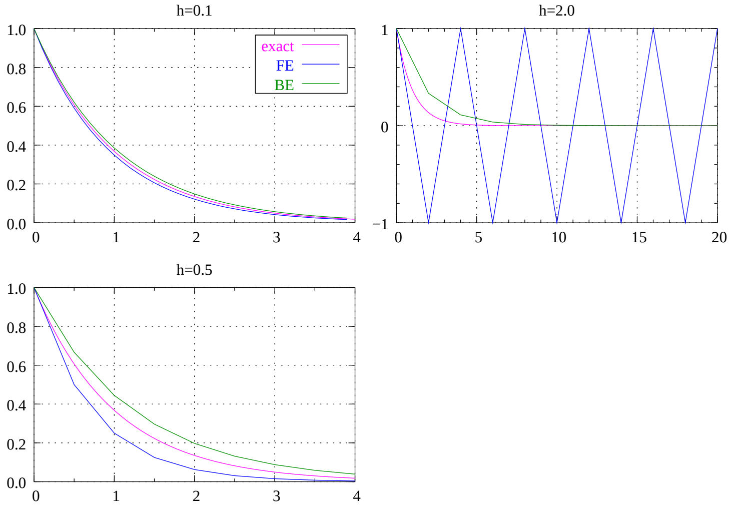


Comparison of FE and BE for $\dot{x} = -x, \quad x(0) = 1$

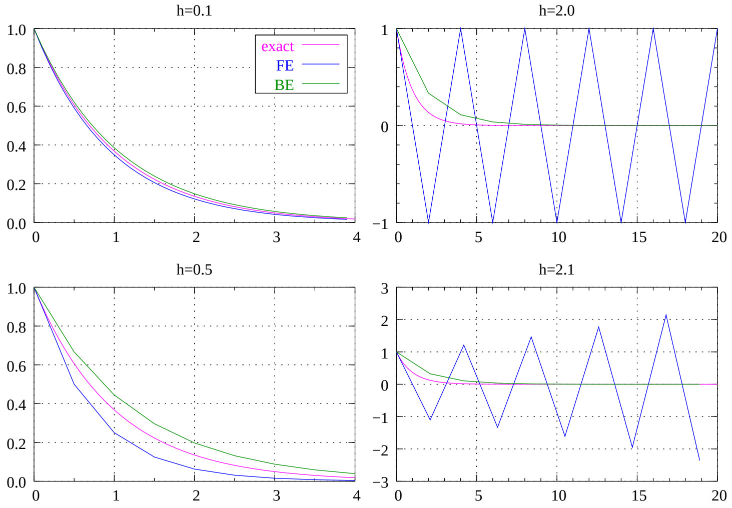




Comparison of FE and BE for $\dot{x} = -x, \quad x(0) = 1$



Comparison of FE and BE for $\dot{x} = -x, \quad x(0) = 1$



Comparison of FE and BE for $\dot{x} = -x, \quad x(0) = 1$

* Although the FE and BE methods are comparable in accuracy, the FE method is *unstable* and therefore not useful for circuit simulation.

Comparison of FE and BE for $\dot{x} = -x, \quad x(0) = 1$

* Although the FE and BE methods are comparable in accuracy, the FE method is *unstable* and therefore not useful for circuit simulation.
* Can we not use a smaller time step and avoid the instability problem?

Comparison of FE and BE for $\dot{x} = -x, \quad x(0) = 1$

* Although the FE and BE methods are comparable in accuracy, the FE method is *unstable* and therefore not useful for circuit simulation.
* Can we not use a smaller time step and avoid the instability problem? Yes, but it increases the simulation time, and in some cases (stiff circuits), by orders of magnitude!

Comparison of FE and BE for $\dot{x} = -x, \quad x(0) = 1$

* Although the FE and BE methods are comparable in accuracy, the FE method is *unstable* and therefore not useful for circuit simulation.
* Can we not use a smaller time step and avoid the instability problem? Yes, but it increases the simulation time, and in some cases (stiff circuits), by orders of magnitude!
* The issue of stability rules out many other methods as well.

Explicit and implicit methods

Consider the ODE,

$$\frac{dx}{dt} = -x^2, \quad \text{with } 1 \leq t \leq 5,\ x(1) = 1\,.$$

Explicit and implicit methods

Consider the ODE,

$$\frac{dx}{dt} = -x^2, \quad \text{with } 1 \leq t \leq 5,\ x(1) = 1\,.$$

Application of the FE, BE, and TRZ formulas yields,

$$\begin{aligned} x_{n+1} &= x_n + h\,(-x_n^2) && (FE)\,, \\ x_{n+1} &= x_n + h\,(-x_{n+1}^2) && (BE)\,, \\ x_{n+1} &= x_n + \frac{h}{2}\,(-x_n^2 - x_{n+1}^2) && (TRZ)\,. \end{aligned}$$

Explicit and implicit methods

Consider the ODE,

$$\frac{dx}{dt} = -x^2, \quad \text{with } 1 \leq t \leq 5, \; x(1) = 1\,.$$

Application of the FE, BE, and TRZ formulas yields,

$$\begin{aligned} x_{n+1} &= x_n + h\,(-x_n^2) && (FE)\,, \\ x_{n+1} &= x_n + h\,(-x_{n+1}^2) && (BE)\,, \\ x_{n+1} &= x_n + \frac{h}{2}\,(-x_n^2 - x_{n+1}^2) && (TRZ)\,. \end{aligned}$$

* In the FE formula, x_{n+1} can be *explicitly* evaluated in terms of x_n.

Explicit and implicit methods

Consider the ODE,

$$\frac{dx}{dt} = -x^2, \quad \text{with } 1 \leq t \leq 5,\ x(1) = 1\,.$$

Application of the FE, BE, and TRZ formulas yields,

$$\begin{aligned} x_{n+1} &= x_n + h\,(-x_n^2) && (FE)\,, \\ x_{n+1} &= x_n + h\,(-x_{n+1}^2) && (BE)\,, \\ x_{n+1} &= x_n + \frac{h}{2}\,(-x_n^2 - x_{n+1}^2) && (TRZ)\,. \end{aligned}$$

* In the FE formula, x_{n+1} can be *explicitly* evaluated in terms of x_n.
* The BE and TRZ formulas result in equations which must be *solved* for x_{n+1}. This is much more work, and it gets worse when there are many equations involved.

Explicit and implicit methods

Consider the ODE,

$$\frac{dx}{dt} = -x^2, \quad \text{with } 1 \leq t \leq 5,\ x(1) = 1\,.$$

Application of the FE, BE, and TRZ formulas yields,

$$\begin{aligned} x_{n+1} &= x_n + h\,(-x_n^2) && (FE)\,, \\ x_{n+1} &= x_n + h\,(-x_{n+1}^2) && (BE)\,, \\ x_{n+1} &= x_n + \frac{h}{2}\,(-x_n^2 - x_{n+1}^2) && (TRZ)\,. \end{aligned}$$

* In the FE formula, x_{n+1} can be *explicitly* evaluated in terms of x_n.
* The BE and TRZ formulas result in equations which must be *solved* for x_{n+1}. This is much more work, and it gets worse when there are many equations involved.
* However, the FE method is not useful because it can be unstable in some cases.

Choice of method for transient simulation

∗ Two major concerns: accuracy (order) and stability

Choice of method for transient simulation

* Two major concerns: accuracy (order) and stability
* A method with a higher accuracy (order) is more efficient as it allows a larger time step $\Rightarrow$ fewer time points $\Rightarrow$ faster simulation.

Choice of method for transient simulation

- Two major concerns: accuracy (order) and stability
- A method with a higher accuracy (order) is more efficient as it allows a larger time step $\Rightarrow$ fewer time points $\Rightarrow$ faster simulation.
- However, high-order methods are *conditionally* stable, i.e., if the time step is large (compared to the smallest time constant in the circuit), the solution grows indefinitely, as in the FE example.

Choice of method for transient simulation

* Two major concerns: accuracy (order) and stability
* A method with a higher accuracy (order) is more efficient as it allows a larger time step $\Rightarrow$ fewer time points $\Rightarrow$ faster simulation.
* However, high-order methods are *conditionally* stable, i.e., if the time step is large (compared to the smallest time constant in the circuit), the solution grows indefinitely, as in the FE example.
* Power electronic circuits are usually stiff (i.e., they involve time constants which are vastly different), and one cannot afford to make h smaller than the smallest τ because
  (a) such a high resolution is not required,
  (b) it would dramatically increase the simulation time.

Choice of method for transient simulation

* Two major concerns: accuracy (order) and stability
* A method with a higher accuracy (order) is more efficient as it allows a larger time step $\Rightarrow$ fewer time points $\Rightarrow$ faster simulation.
* However, high-order methods are *conditionally* stable, i.e., if the time step is large (compared to the smallest time constant in the circuit), the solution grows indefinitely, as in the FE example.
* Power electronic circuits are usually stiff (i.e., they involve time constants which are vastly different), and one cannot afford to make h smaller than the smallest τ because
  (a) such a high resolution is not required,
  (b) it would dramatically increase the simulation time.
* The stability constraints significantly reduce the choices available for circuit simulation. BE, Gear (order 2), and Trapezoidal methods are commonly used.

Equivalent circuit for a capacitor

With Backward Euler method, we get

$$\frac{v_C^{n+1} - v_C^n}{h} = \frac{1}{C}\, i_C^{n+1}\,.$$

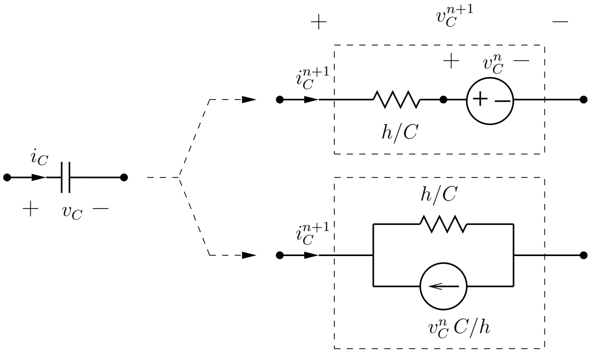
$$\text{i.e.,}\quad v_C^{n+1} = \frac{h}{C}\, i_C^{n+1} + v_C^n \qquad \text{OR} \qquad i_C^{n+1} = \frac{C}{h}\, v_C^{n+1} - \frac{C}{h}\, v_C^n\,.$$

Equivalent circuit for a capacitor

With Backward Euler method, we get

$$\frac{v_C^{n+1} - v_C^n}{h} = \frac{1}{C}\, i_C^{n+1}\,.$$

i.e., $v_C^{n+1} = \frac{h}{C}\, i_C^{n+1} + v_C^n$ OR $i_C^{n+1} = \frac{C}{h}\, v_C^{n+1} - \frac{C}{h}\, v_C^n\,.$



Time step selection

* In some circuits, a constant Δt is appropriate; in others, especially with many switching events, automatic time step selection is more effective.

Time step selection

* In some circuits, a constant Δt is appropriate; in others, especially with many switching events, automatic time step selection is more effective.
* Automatic time step selection is based on (a) estimate of the local truncation error at a given time step, (b) convergence behaviour of N-R iterations.

Time step selection

* In some circuits, a constant Δt is appropriate; in others, especially with many switching events, automatic time step selection is more effective.
* Automatic time step selection is based on (a) estimate of the local truncation error at a given time step, (b) convergence behaviour of N-R iterations.
* Power electronic circuits are generally nonlinear; time step has a significant impact on convergence of N-R iterations. Option (b) is therefore very effective.

Time step selection

* In some circuits, a constant Δt is appropriate; in others, especially with many switching events, automatic time step selection is more effective.
* Automatic time step selection is based on (a) estimate of the local truncation error at a given time step, (b) convergence behaviour of N-R iterations.
* Power electronic circuits are generally nonlinear; time step has a significant impact on convergence of N-R iterations. Option (b) is therefore very effective.
* N-R convergence

Time step selection

* In some circuits, a constant Δt is appropriate; in others, especially with many switching events, automatic time step selection is more effective.
* Automatic time step selection is based on (a) estimate of the local truncation error at a given time step, (b) convergence behaviour of N-R iterations.
* Power electronic circuits are generally nonlinear; time step has a significant impact on convergence of N-R iterations. Option (b) is therefore very effective.
* N-R convergence
  - The solution obtained at t_i serves as the "initial guess" at t_{i+1}.

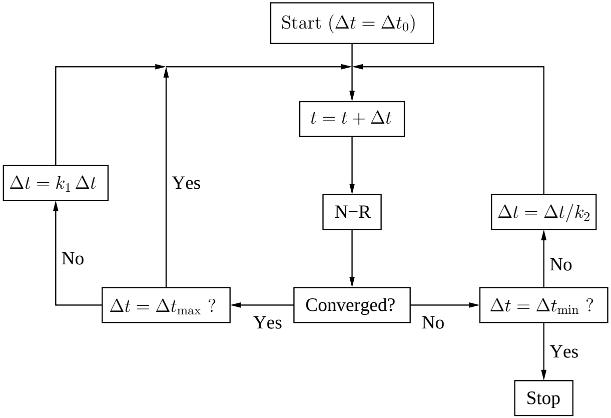
Time step selection

* In some circuits, a constant Δt is appropriate; in others, especially with many switching events, automatic time step selection is more effective.
* Automatic time step selection is based on (a) estimate of the local truncation error at a given time step, (b) convergence behaviour of N-R iterations.
* Power electronic circuits are generally nonlinear; time step has a significant impact on convergence of N-R iterations. Option (b) is therefore very effective.
* N-R convergence
  - The solution obtained at t_i serves as the "initial guess" at t_{i+1}.
  - Δt too large $\Rightarrow$ N-R iterations may not converge.

Time step selection

- In some circuits, a constant Δt is appropriate; in others, especially with many switching events, automatic time step selection is more effective.
- Automatic time step selection is based on (a) estimate of the local truncation error at a given time step, (b) convergence behaviour of N-R iterations.
- Power electronic circuits are generally nonlinear; time step has a significant impact on convergence of N-R iterations. Option (b) is therefore very effective.
- N-R convergence
  - The solution obtained at t_i serves as the "initial guess" at t_{i+1}.
  - Δt too large $\Rightarrow$ N-R iterations may not converge.
  - Δt too small $\Rightarrow$ large simulation time.

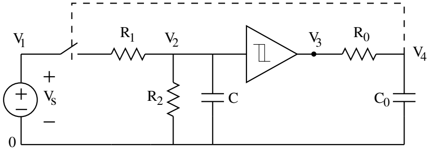
Automatic time step selection

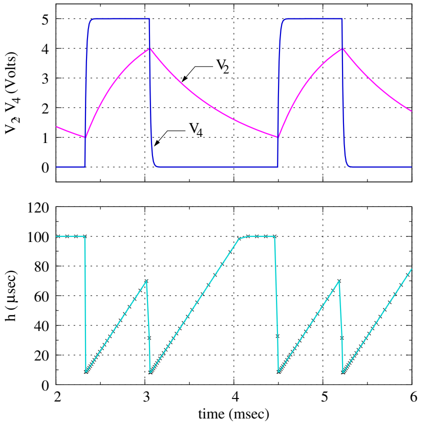


Start ($\Delta t = \Delta t_0$)
$t = t + \Delta t$
N−R
Converged?
Yes
No
$\Delta t = \Delta t_{\max}$?
$\Delta t = \Delta t_{\min}$?
Yes
No
$\Delta t = k_1\,\Delta t$
No
Yes
$\Delta t = \Delta t / k_2$
Stop

Automatic time step selection: example

Automatic time step selection based on convergence of N–R iterations





Transient simulation: are the results accurate?

* very difficult to judge except for simple problems

Transient simulation: are the results accurate?

* very difficult to judge except for simple problems
* In practice, reduce Δt by a factor of 2 and see if the results are different.

Transient simulation: are the results accurate?

* very difficult to judge except for simple problems
* In practice, reduce Δt by a factor of 2 and see if the results are different.
* Usually, the user would have some idea of the time scale, For example,
  (a) Buck converter: $\Delta t = T_c/50$ may be appropriate.
  (b) Half-wave rectifier: $\Delta t = T/50$ may be appropriate.
  Such a rule of thumb provides a good starting point.

Steady-State Waveform (SSW) Analysis

* In many periodic systems, only the steady-state behaviour is of interest (and not how it is attained). e.g., power electronic circuits, rf circuits

Steady-State Waveform (SSW) Analysis

* In many periodic systems, only the steady-state behaviour is of interest (and not how it is attained). e.g., power electronic circuits, rf circuits
* Transient simulation (from some initial condition to the steady state) may involve thousands of cycles; this is very expensive.

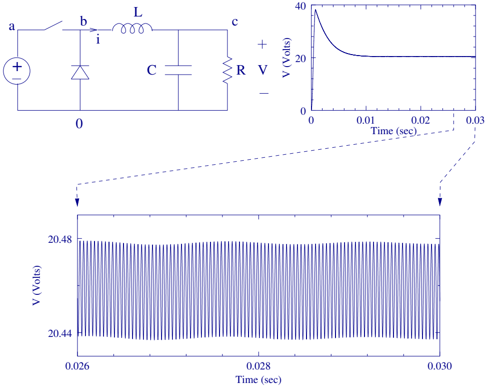
Steady-State Waveform (SSW) Analysis

* In many periodic systems, only the steady-state behaviour is of interest (and not how it is attained). e.g., power electronic circuits, rf circuits
* Transient simulation (from some initial condition to the steady state) may involve thousands of cycles; this is very expensive.
* Total time for which transient simulation needs to be performed to reach the steady state is not known *a priori*; need to rely on a trial-and-error approach.

Steady-State Waveform (SSW) Analysis

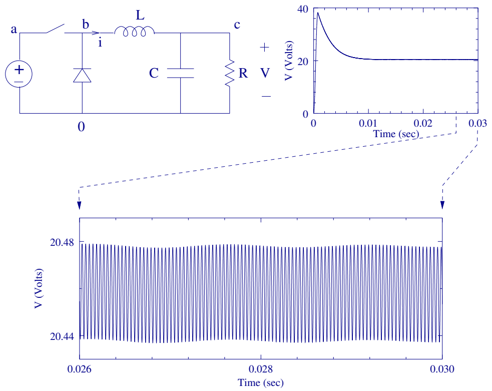
- In many periodic systems, only the steady-state behaviour is of interest (and not how it is attained). e.g., power electronic circuits, rf circuits
- Transient simulation (from some initial condition to the steady state) may involve thousands of cycles; this is very expensive.
- Total time for which transient simulation needs to be performed to reach the steady state is not known *a priori*; need to rely on a trial-and-error approach.
- It is much faster to obtain the steady-state information *directly* where a nonlinear problem in the state variables is solved.

SSW Analysis: Buck Converter



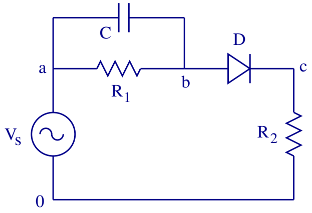
∗ A large number of cycles are required if transient simulation is used. (Note that, for this example, the steady state is not quite reached as indicated by the small amplitude variation.)

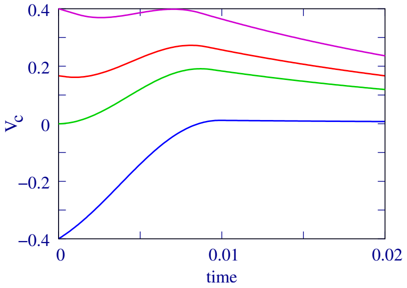
SSW Analysis: Buck Converter



* A large number of cycles are required if transient simulation is used. (Note that, for this example, the steady state is not quite reached as indicated by the small amplitude variation.)
* If a component value (L or C) is changed, we would not know how long to simulate to attain steady state. This is cumbersome.

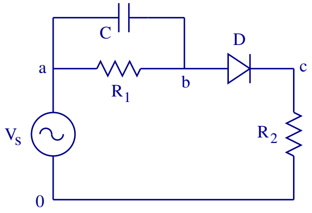
SSW Analysis: Basic idea

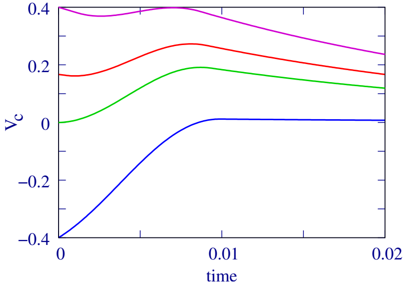




* Start with an initial guess for the state variable(s) (the capacitor voltage here).

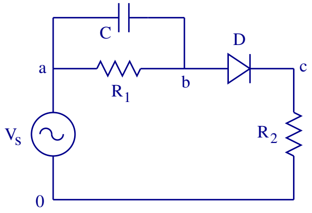
SSW Analysis: Basic idea

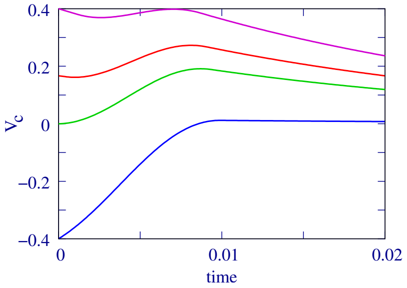




* Start with an initial guess for the state variable(s) (the capacitor voltage here).
* Integrate for one cycle. Is $V_c(T) = V_c(0)$?

SSW Analysis: Basic idea





* Start with an initial guess for the state variable(s) (the capacitor voltage here).
* Integrate for one cycle. Is $V_c(T) = V_c(0)$?
* If yes (red curve), we have obtained the SSW solution; if not, we need to compute a better initial guess (in an *outer* Newton-Raphson loop) and repeat [7].

SSW: Examples

Example	N_{trns}	N_{ssw}
Buck Converter	750	4
Boost Converter	625	3
Cúk Converter	1250	3
1-ϕ half-wave rectifier	150	3
1-ϕ half-controlled bridge converter	110	4
3-ϕ diode bridge rectifier	200	4
Induction motor	125	17

* Note the dramatic reduction in computational effort for the SSW method as compared to transient analysis.

References

[1] L. O. Chua and P. M. Lin, *Computer-Aided Analysis of Electronic Circuits*, Englewood Cliffs: Prentice-Hall, 1976.

[2] W. J. McCalla, *Fundamentals of Computer-Aided Circuit Simulation*, Boston: Kluwer Academic Publishers, 1987.

[3] R. Raghuram, *Computer Simulation of Electronic Circuits*, New Delhi: Wiley Eastern, 1989.

[4] K. S. Kundert, *The Designer's Guide to SPICE and SPECTRE*, Boston: Kluwer Academic Publishers, 1995.

[5] M. B. Patil, V. Ramanarayanan, and V. T. Ranganathan, *Simulation of Power Electronic Circuits,* to be published.

[6] C. D. Hachtel and R. K. Brayton and F. G. Gustavson, "The sparse tableau approach to network analysis and design," *IEEE Trans. CT*, vol. 18, pp. 101-113, 1971.

[7] F. R. Colon and T. N. Trick, "Fast periodic steady-state analysis for large-signal electronic circuits," *IEEE J. Solid-State Circuits*, vol. 8, pp. 260-269, 1973.

[8] C. F. Gerald and P. O. Whitley, *Applied Numerical Analysis*, Delhi: Pearson Education India, 1999.