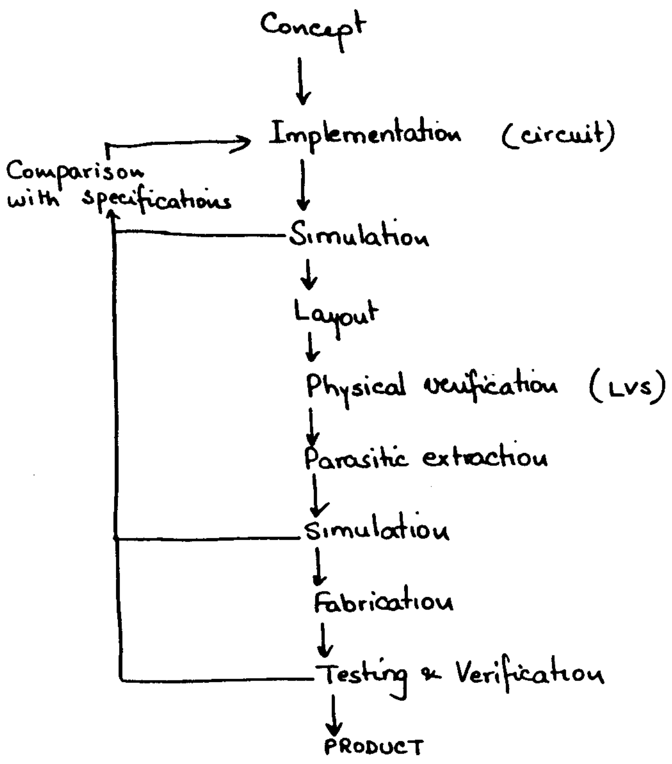
ECE 521

Analog Circuit Simulation

To understand what is simulation let us consider a very simple circuit design flow



Simulation is a means of verifying the design

Why verify?

Verification is necessary to see if the specifications are met

i.e. Is the design that I have what I wanted?

How can one verify a circuit design?

- Prototyping or breadboarding
  - implement and see if the design works
  - tweak component values to ensure proper operation
- Simulation
  - start from mathematical descriptions (models) of components
  - formulate equations based on physical laws
  - specify input test patterns and determine the output as a function of the input
  - equations solved by use of a computer

Why solve the equations on a computer?

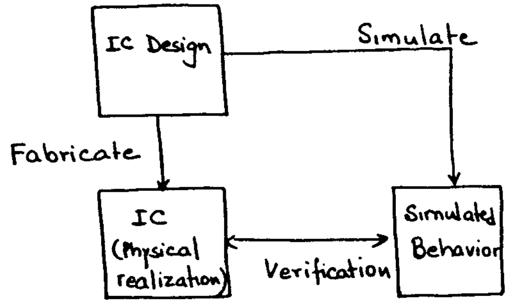
- problem has no closed form solution, so use numerical techniques
- problems may be very large so cannot be solved by hand
- A closed-form solution may exist but only in specific cases. Cannot be used for general situations

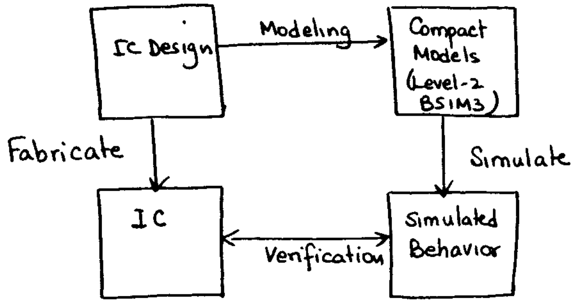
In the IC world simulation is a necessity

- Cannot breadboard ICs
- Fabrication for design iterations is an expensive alternative

- Simulation is the first step in ensuring successful first-pass silicon
- Can do an early design even before the complete process exists.

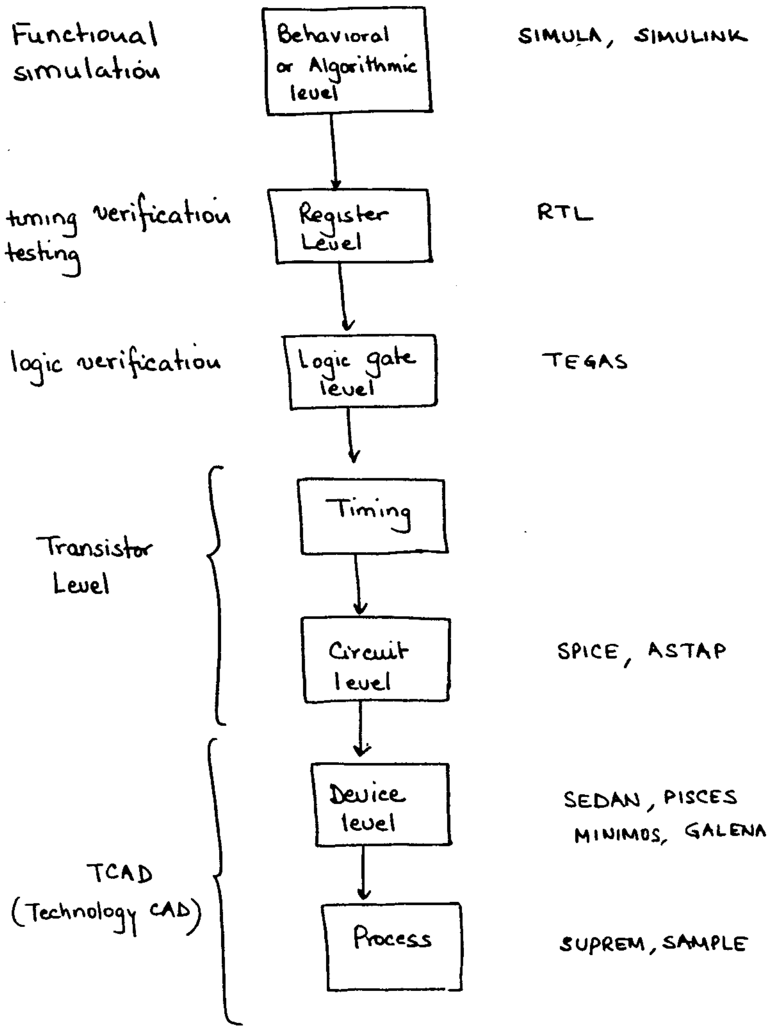
The role of simulation & modeling





Models are extremely important for analog simulation!

The various levels of simulation



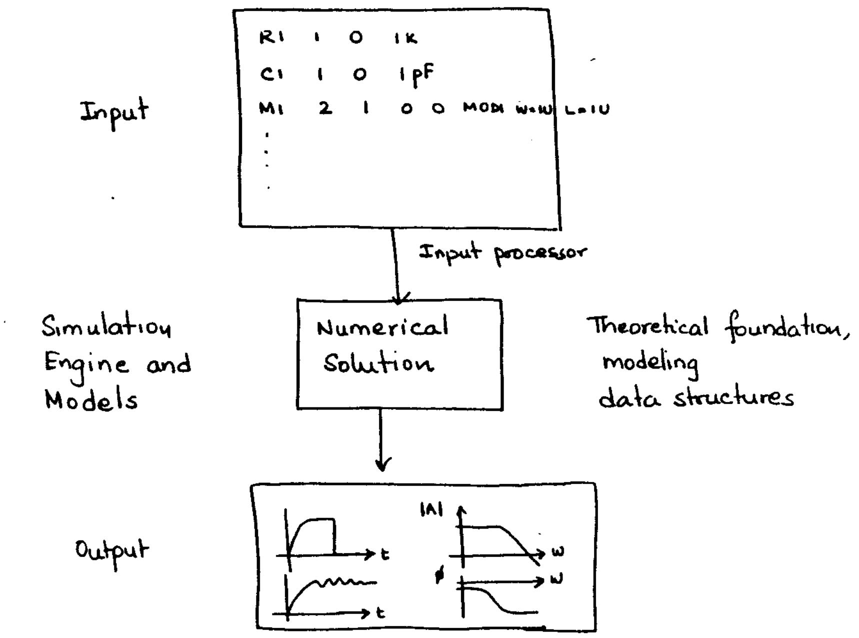
Our focus is on circuit level simulation ala SPICE

Why analog circuit-level simulation

- Most difficult and challenging
- Analog behavior specified in terms of complex functio-
  - time-domain waveforms (settling time, slewing)
  - frequency response (ac analysis, power spectra)
  - distortion (HD, IMD)
  - noise (due to devices: thermal, flicker, ..
  - matching
- Require very accurate component models. Digital is rather forgiving in terms of models.

Challenges in analog circuit simulation
- accurate models;
  - low frequency
  - high frequency
  - noise
  - distortion generation
  - statistical variations
- faster simulation techniques
  - power supplies
  - ΣΔ modulators
  - RF oscillators, mixers
  - Phase-locked loops
  - phase noise, noise mixing.
  - accurate distortion calculation

What is this course about?



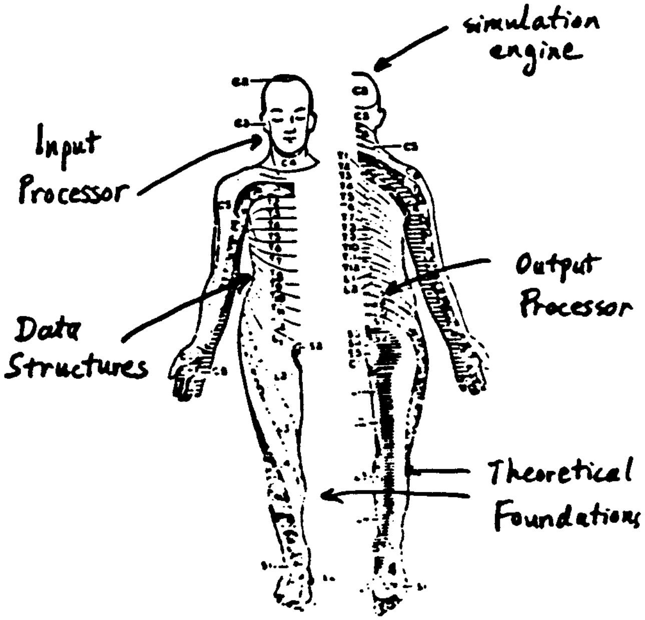
This course will focus on the underlying principles of a SPICE-like simulator

Who can benefit from the course?

Circuit designers: be an informed consumer of the simulation tools. An understanding of the simulator can help in identifying simulation related problems.

Model developers: Models go in simulators. A tight coupling between models & algorithms

THE ANATOMY OF

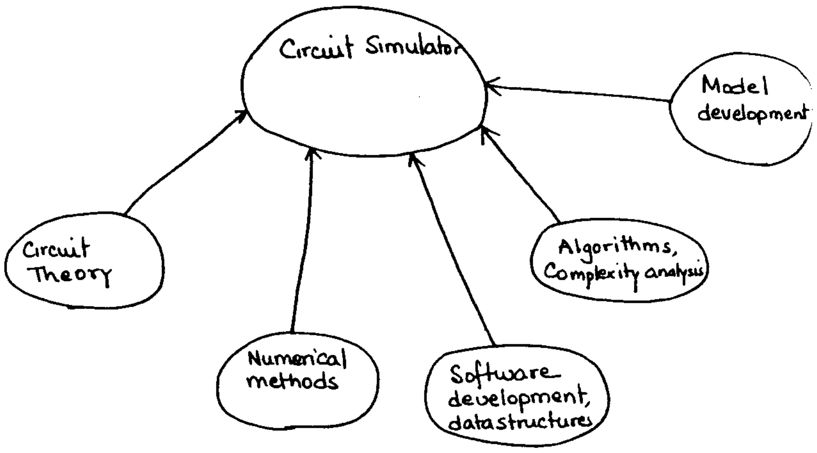


A CIRCUIT SIMULATOR

FROM DR. RES SALEH

CAD tool developer: simulators are an important part of the CAD framework. Develop new simulators and algorithms.

What are the basic skills required


Circuit Simulator
Model development
Circuit Theory
Numerical methods
Software development, datastructures
Algorithms, Complexity analysis

The approach in this course

- Linear resistive networks (R, I, V, controlled sources)
  - equation formulation
  - solution techniques (linear)
- Nonlinear resistive networks
  - equation formulation
  - nonlinear equation solution
- Nonlinear dynamic networks (L, C)

Circuit Simulation

Simulators are the most often used CAD tools.
Circuit simulation ~ 90%

- 60,000 runs/month at company A
- several weeks on a Sparc 20 for simulating PLLs, A/D converters
- several thousand copies of SPICE & internal SPICE-like simulators in use around the world
- several companies selling alphabet SPICE (seems to be a good business!)
- several companies have entire supercomputers dedicated to simulation.

Simulation is an expensive task so if you come up with a new technique to improve simulation speed or solve difficult problems ⇒ Famous & Rich!

A Brief Overview of SPICE (Simulation Program with IC Emphasis)

1969-70	The CANCER Project (Computer Analysis of Nonlinear Circuits Excluding Radiation) Ron Rohrer's class project	
1970-72	CANCER Program (Rohrer and Larry Nagel)	
1972	SPICE 1 released as a public-domain tool	
1975	SPICE 2A, 2C	
1976	SPICE 2D	New MOS Models
1979	SPICE 2E	Device levels
1980	SPICE 2F	Portable SPICE, MOSFET charge models
1982	SPICE 2G	
1985	SPICE 3C	

The Alphabet SPICEs
HSPICE, ISPICE, GSPICE, OSPICE, PSPICE, ...

Internal SPICE
HP-SPICE, TI-SPICE, TEK-SPICE, $\underbrace{\text{M}}_{\text{Motorola}}$ SPICE, $\underbrace{\text{AD}}_{\text{Analog Devices}}$ICE
$\underbrace{\text{ADVICE}}_{\text{ATT}}$, ...

Others
SmartSPICE, B^2SPICE, IsSPICE, SPICE-It, Dr. SPICE, ...

Why has SPICE been so successful?

- proper choice of algorithms and software system
- "friendly" input description language
- public domain software
- developed by circuit designers

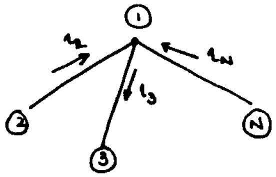
What is the present?

- SPICE 3 from Berkeley in support mode
- Next generation company proprietary simulators; Motorola, Lucent, TI, IBM
- Commercial simulators: SPECTRE / SPECTRE-RF
  HP EESof / LIBRA
  SABER

Formulation of Circuit Equations

Kirchhoff's Current Law (KCL)

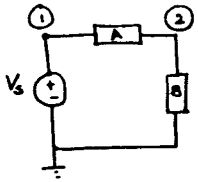
- The sum of currents leaving a node is zero.



$$-i_2 + i_3 - i_N = 0$$

Kirchhoff's Voltage Law (KVL)

- The sum of voltage drops around a loop is zero.



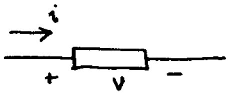
$$(V_1 - V_2) + V_2 - V_1 = 0$$
$$V_A + V_B - V_S = 0$$

Note $V_A = V_1 - V_2$ or $V_A - (V_1 - V_2) = 0$
i.e. a relation between branch and node voltages

Branch Constitutive Relations (BCRs) or Equations (BCEs)

These are the mathematical models of the circuit components. Relate currents, voltages, charges and fluxes.

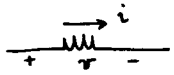
Define a reference direction



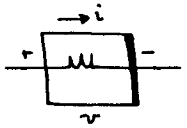
Ideal Two-terminal elements

1) Resistor

linear



non linear

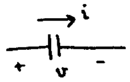


	linear	non linear
Voltage controlled	$i = 1/R\ v = Gv$	$i = i(v)$
Current controlled	$v = Ri$	$v = v(i)$

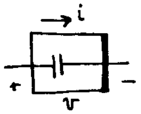
Can also have an implicit relationship $f(i,v) = 0$

2) Capacitor

linear



non linear



linear	non linear
$i = dq/dt$	$i = dq/dt$
$q = Cv$	$q = q(v)$
$i = C\dfrac{dv}{dt}$	$i = \dfrac{\partial q}{\partial v}\dfrac{dv}{dt}$

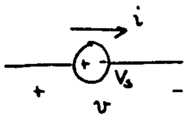
Implicit form $f(q,v) = 0$.

Independent sources

1) Voltage source

$v = V_s$ (dc, tran, ac)

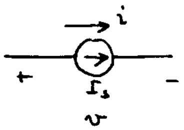
i = any value



2) Current source

$i = I_s$ (dc, tran, ac)

v = any value



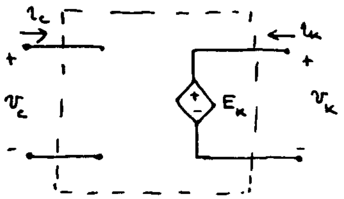
Dependent (Controlled) sources

1) VCVS (voltage controlled voltage source)

linear

$v_k = E_k v_c$

$i_c = 0$; i_k = any value



nonlinear

$v_k = v_k(v_c)$

$i_c = 0$; i_k = any value

2) CCCS (current controlled current source)

linear

$i_k = F_k i_c$

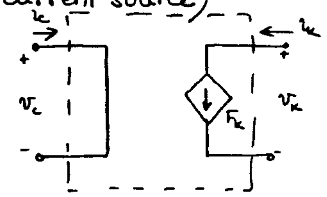
$v_c = 0$

nonlinear

$i_k = i_k(i_c)$

$v_c = 0$

v_k = any value



3) VCCS (Voltage controlled current source)

linear:

$i_k = G_k v_c$

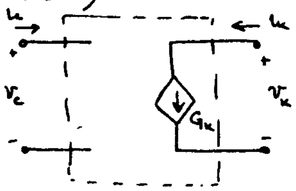
$i_c = 0$

nonlinear

$i_k = i_k(v_c)$

$i_c = 0$

v_k can be any value



4) CCVS (current controlled voltage source)

linear

$v_k = H_k i_c$

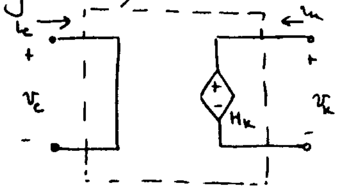
$v_c = 0$

nonlinear

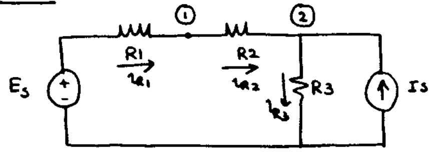
$v_k = v_k(i_c)$

$v_c = 0$

i_k can be any value.



Nodal Analysis



KCL @ ① : $-i_{R1} + i_{R2} = 0$

KCL @ ② : $-i_{R2} + i_{R3} - I_s = 0$

KVL and BCR give : $i_{R1} = (E_s - V_1)\frac{1}{R_1} = G_1(E_s - V_1)$

$i_{R2} = G_2(V_1 - V_2)$

$i_{R3} = G_3 V_2$

Thus,

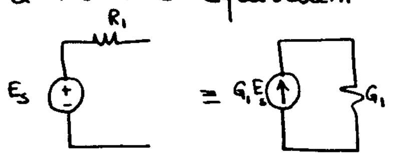
$$+G_1(V_1-E_s) + G_2(V_1-V_2) = 0$$

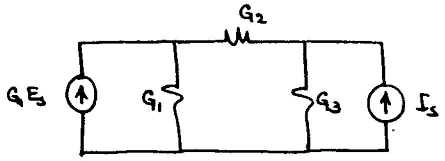
$$-G_2(V_1-V_2) + G_3 V_2 - I_s = 0$$

or

$$\begin{bmatrix} G_1+G_2 & -G_2 \\ -G_2 & G_2+G_3 \end{bmatrix}\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} G_1E_s \\ I_s \end{bmatrix}$$

These equations can be formulated by inspection if one uses a Norton's equivalent



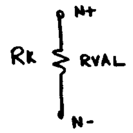


In Matrix form we have

$$GV = I_s$$

where G & I_s can be assembled by inspection. The contribution of each element can be represented as a "template" or "stamp"

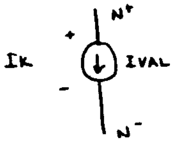
Resistor   SPICE input : RK N+ N- RVAL



$$\underbrace{\left[\begin{array}{c|cc} & N+ \cdots & N- \\ \hline N+ & +1/RVAL & -1/RVAL \\ \vdots & & \\ N- & -1/RVAL & +1/RVAL \end{array}\right]}_{\text{Resistor stamp}} \qquad \overset{RHS}{\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}}$$

Current Source (independent)

SPICE input : IK N+ N- IVAL



$$\begin{array}{c} N+ \\ \vdots \\ N- \end{array}\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} = \overset{RHS}{\begin{bmatrix} -IVAL \\ \\ +IVAL \end{bmatrix}}$$

Voltage source (independent)

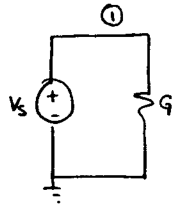
There is no nodal formulation. Nodal analysis requires elements that are voltage controlled. So what do we do?

0) Transform into Norton's Equivalent if R in series

1) Grounded voltage sources:

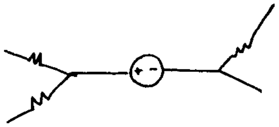
Node voltage is known so do not write any equation

i.e. have to check node connection for each element !



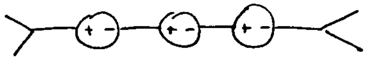
2) Floating voltage source

No "stamp". Difficult if not impossible to treat floating voltage sources in NA



3) Stacked voltage sources

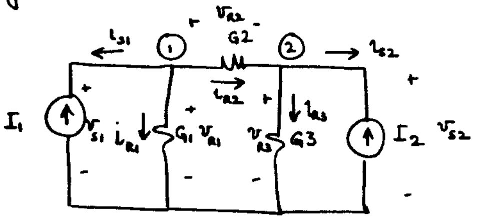
- this again poses a difficulty for NA



Summary of Nodal Analysis

- Circuit equations can be assembled by inspection
- G is sparse (depends on interconnection between nodes)
- G has non-zero diagonal entries and is diagonally dominant
- Cannot handle voltage sources
- Requires voltage controlled elements ($i = f(v)$)
    - cannot incorporate VCVS, CCCS, CCVS
    - VCCS is OK because $i = Gv$.

Let us revisit our example with a Norton's equivalent for the voltage source in series



Our method:

KCL at nodes ① & ②

$$\begin{aligned} i_{S1} + i_{R1} + i_{R2} &= 0 \\ - i_{R2} + i_{R3} + i_{S2} &= 0 \end{aligned} \Rightarrow \underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 \end{bmatrix}}_{A} \begin{bmatrix} i_{S1} \\ i_{R1} \\ i_{R2} \\ i_{R3} \\ i_{S2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

General form $\quad A\, I_b = 0$

Branch Constitutive Relations (BCR)

$$\begin{aligned} i_{S1} - 0\, v_{S1} &= -I_1 \\ i_{R1} - G_1 v_{R1} &= 0 \\ i_{R2} - G_2 v_{R2} &= 0 \\ i_{R3} - G_3 v_{R3} &= 0 \\ i_{S2} - 0\, v_{S2} &= -I_2 \end{aligned} \Rightarrow \begin{bmatrix} i_{S1} \\ i_{R1} \\ i_{R2} \\ i_{R3} \\ i_{S2} \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & +G_1 & 0 & 0 & 0 \\ 0 & 0 & +G_2 & 0 & 0 \\ 0 & 0 & 0 & G_3 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_{S1} \\ v_{R1} \\ v_{R2} \\ v_{R3} \\ v_{S2} \end{bmatrix} = \begin{bmatrix} -I_1 \\ 0 \\ 0 \\ 0 \\ -I_2 \end{bmatrix}$$

General form: $\quad I_b - G_b V_b = S$

Node to branch relations (KVL)

$$\begin{aligned} v_{S1} &= v_1 \\ v_{R1} &= v_1 \\ v_{R2} &= v_1 - v_2 \\ v_{R3} &= v_2 \\ v_{S2} &= v_2 \end{aligned} \Rightarrow \begin{bmatrix} v_{S1} \\ v_{R1} \\ v_{R2} \\ v_{R3} \\ v_{S2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & -1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

General form $\quad V_b = A^T V_n$

Remark

- The matrix A is called the incidence matrix

Summary of Equations

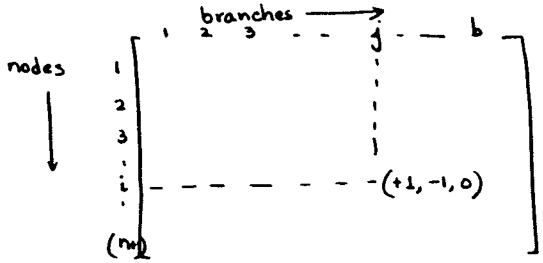
$$\begin{aligned} A\, I_b &= 0 && (\text{KCL}) \\ I_b - G_b V_b &= S && (\text{BCR}) \\ V_b - A^T V_n &= 0 && (\text{node-to-branch relation}) \end{aligned}$$

$$I_b - G_b A^T V_n = S$$

$$\cancel{A I_b}^{\,0} - A G_b A^T V_n = A S$$

$$\underbrace{A G_b A^T}_{G} V_n = \underbrace{-AS}_{I_s}$$

Incidence Matrix



$(1+n) \times b$ matrix
n nodes, 1 ground

A_{ij} = +1 if node i is + terminal of branch j
= −1 " " " " − " " "
0 if node i is not connected to branch j

Properties

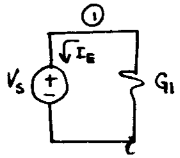
- A is <u>unimodular</u>, i.e., determinant of any square submatrix of A is +1, −1, or 0.
- A has 2 nonzero entries in any column
- All columns sum to zero (+1 + −1 = 0)
- Can delete a row of the complete incidence matrix i.e. choose a ground or datum node

Let us now revisit the problem of voltage sources in nodal analysis. How do we handle voltage sources?

1) crude solution: add a "small" resistance in series with each voltage source

2) Reformulate equations ⇒ Modified Nodal Analysis or MNA.

At ① we have $V_1 = V_S$
and $G_1 V_1 + I_E = 0$



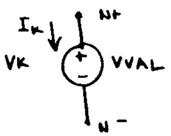
We now have 2 eqns in 2 unknowns V_1 & I_E

$$\begin{bmatrix} 1 & 0 \\ G_1 & 1 \end{bmatrix} \begin{bmatrix} V_1 \\ I_E \end{bmatrix} = \begin{bmatrix} V_S \\ 0 \end{bmatrix}$$

So we can now define a stamp for the voltage source in MNA

Floating voltage source

SPICE VK N+ N− VVAL



$$\begin{array}{c|ccc} & N+ \cdots & N- \cdots & I_K \\ \hline N+ & 0 & 0 & 1 \\ \vdots & & & \\ N- & 0 & 0 & -1 \\ \text{branch } k & 1 & -1 & 0 \end{array} \qquad \begin{array}{c} \text{RHS} \\ \hline 0 \\ \vdots \\ 0 \\ VVAL \end{array}$$

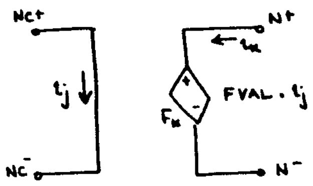
At node N+ : $\cdots + I_K = 0$
" " N− : $\cdots - I_K = 0$
BCR (branch k): $V_{N+} - V_{N-} = VVAL$

Remarks

- MNA matrix and RHS can be assembled <u>directly from the input</u>. The stamps provide a convenient way of assembling the equations
- MNA can be applied to any circuit
- Sometimes there may be zeros on the main diagonal
- MNA matrix is "close" to the NA matrix. It includes NA + additional equations & unknowns.

MNA stamp for CCVS

SPICE FK N+ N- NC+ NC- FVAL



KCL:

$$\begin{array}{lllll}
N+ & \ldots & + i_k & = 0 \\
N^- & \ldots & - i_k & = 0 \\
NC+ & \ldots & + i_j & = 0 \\
NC- & \ldots & - i_j & = 0
\end{array}$$

BCR:

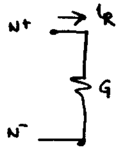
$$V_{N+} - V_{N-} - FVAL * i_j = 0$$

$$V_{NC+} - V_{NC-} = 0$$

$\Rightarrow$ stamp

$$\begin{array}{c|cccccc|}
 & N+ & N- & NC+ & NC- & i_k & i_j \\
\hline
N+ & & & & & +1 & \\
N- & & & & & -1 & \\
NC+ & & & & & & +1 \\
NC- & & & & & & -1 \\
\text{branch } k & 1 & -1 & & & & FVAL \\
\text{branch } j & & & 1 & -1 & & \\
\end{array}$$

What is the MNA representation for a resistor?



$$\begin{array}{c|ccc|}
 & N+ & N- & i_R \\
\hline
N+ & & & 1 \\
N- & & & -1 \\
\text{branch} & 1 & -1 & -R \\
\end{array}$$

or $(G \quad -G \quad -1)$

When would we need such a stamp?

- when the current through R is requested as an output
- some circuit element depends on i_R

MNA Rules

1) A branch current is always used as an additional variable for a
   - voltage source
   - inductor
   - controlled voltage source

2) For R, C, I the branch current is taken as an additional variable only when
   - the branch current is an output
   - some circuit element depends on the branch current.

General Formulation - The Sparse Tableau Analysis (STA)

(Brayton, Gustavson, Hachtel 1969-71)

KCL: $A I_b = 0$

(n equations)

$$\underbrace{\overset{\longleftarrow b \text{ branches} \longrightarrow}{\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}}}_{} \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_b \end{bmatrix} = 0$$

n nodes (exclude ground)

KVL: $V_b - A^T V_n = 0$

(b equations)

$$\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_b \end{bmatrix} - \left[A^T\right]_{b \times n} \begin{bmatrix} V_{n1} \\ V_{n2} \\ \vdots \\ V_{nn} \end{bmatrix} = 0$$

BCR $k_i I_b + k_v V_b = S$

(b equations)

Examples

Resistor $I_b - GV_b = 0$

Voltage source $0 \cdot I_b - V_b = V_s$

Current source $I_b - 0V_b = I_s$

$\therefore$ in general write $K_i I_b + K_v V_b = S$

Collect all these equations and we have the sparse tableau

$$\begin{array}{c|c}
 & \begin{array}{ccc} \longleftarrow b \longrightarrow & \longleftarrow b \longrightarrow & \longleftarrow n \longrightarrow \end{array} \\
\begin{array}{c} n \\ b \\ b \end{array} & \left[\begin{array}{c:c:c} A & 0 & 0 \\ \hdashline 0 & I & -A^T \\ \hdashline K_i & K_v & 0 \end{array}\right] \left[\begin{array}{c} I_{b1} \\ \vdots \\ I_{bb} \\ \hdashline V_{b1} \\ \vdots \\ V_{bb} \\ \hdashline V_{n1} \\ \vdots \\ V_{nn} \end{array}\right] = \left[\begin{array}{c} 0 \\ \hdashline 0 \\ \hdashline S \end{array}\right]
\end{array}$$

Observations

- $n + 2b$ equations
- $2b + n$ unknowns
- A has at most $2b$ nonzeros
- A^T " " " $2b$ "
- I has b nonzeros
- K_i, K_v have at most b nonzeros

$\therefore$ maximum # of nonzeros: $2b + 2b + b + b + b = 7b$

Matrix size $(n + 2b) \times (n + 2b)$

$\Rightarrow$ matrix is very sparse

Summary of STA

- Can be applied to any circuit
- Equations can be assembled by inspection
- Sparse tableau matrix is very sparse
- There are several +1, -1 entries (A, A^T, I) which can be processed once
  $\Rightarrow$ sphisticate programming techniques + data structures for efficiency.

Remark

NA and MNA can be derived from STA

Example (NA)

$$A I_b = 0 \qquad \text{KCL}$$
$$V_b - A^T V_n = 0 \qquad \text{KVL}$$
$$K_i I_b + K_v V_b = S \qquad \text{BCR}$$

Substitute for V_b in BCR from KVL, i.e. $V_b = A^T V_n$

$$\therefore \quad A I_b = 0$$
$$K_i I_b + K_v A^T V_n =$$

Assume K_i is invertible

$$A I_b = 0 \qquad ①$$
$$I_b = -K_i^{-1} K_v A^T V_n + K_i^{-1} S \qquad ②$$

Multiply ② by A: $A I_b = -A K_i^{-1} K_v A^T V_n + A K_i^{-1} S$
$= 0$ from ①

$$\therefore \quad \underbrace{\left(A K_i^{-1} K_v A^T\right)}_{\substack{\text{NA} \\ \text{matrix}}} \underset{\substack{\uparrow \\ \text{node voltages}}}{V_n} = A K_i^{-1} \underset{\substack{\nwarrow \\ \text{Source}}}{S}$$

So far we have seen how to assemble circuit equations using NA, MNA, or STA.

The next step is to solve these equations given S, the source vector. So we now look at the solution of Linear algebraic equations

Solution of Linear Algebraic Equations

Given a matrix A, a source vector b, solve for the unknown x

$$A x = b$$

A is $n \times n$ and non-singular (real)

x is $n \times 1$, b is $n \times 1$

Methods

- Direct Methods
  - Gaussian Elimination
  - LU Decomposition
    - Crout
    - Doolittle
- Indirect Methods (Iterative methods)
  - Gauss-Jacobi
  - Gauss-Seidel
  - Successive Over Relaxation (SOR)
  - Krylov subspace methods
    - conjugate gradient
    - Generalized minimum residuals (GMRES)
    - QMR

Gaussian Elimination (GE)

$$A x = b$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Solve 1st equation for x_1 and eliminate x_1 from other equations

multipliers

$$\text{row}'2 = \text{row}2 - \frac{a_{21}}{a_{11}}\,\text{row}1$$

$$\vdots$$

$$\text{row}n = \text{row}n - \frac{a_{n1}}{a_{11}}\,\text{row}1$$

This elimination step results in:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} - \frac{a_{21}}{a_{11}} a_{12} & \cdots & a_{2n} - \frac{a_{21}}{a_{11}} a_{1n} \\ \vdots & & & \\ 0 & a_{n2} - \frac{a_{n1}}{a_{11}} a_{12} & \cdots & a_{nn} - \frac{a_{n1}}{a_{11}} a_{1n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 - \frac{a_{21}}{a_{11}} b_1 \\ \vdots \\ b_n - \frac{a_{n1}}{a_{11}} b_1 \end{bmatrix}$$

Repeat process to obtain

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ 0 & 0 & \dots & a_{3n}^{(3)} \\ \vdots & \vdots & & \\ 0 & 0 & \dots & a_{nn}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2^{(2)} \\ \vdots \\ b_n^{(n)} \end{bmatrix}$$

This is an upper triangular system from which one can solve $x_n, x_{n-1}, \dots x_1$ using <u>backward</u> <u>substitution</u>

$$x_n = \frac{b_n^{(n)}}{a_{nn}^{(n)}}$$

$$x_{n-1} = \frac{b_{n-1}^{(n-1)} - a_{n-1,n}^{(n-1)} x_n}{a_{n-1,n-1}^{(n-1)}}$$

$$\vdots$$

$$x_1 = \Big(b_1 - \sum_{j=2}^{n} a_{1j} x_j\Big) / a_{11}$$

Summarize the procedure

- triangularization

```
For $i = 1$ to $n-1$ {                    (each row)
    For $j = i+1$ to $n$ {                 (each row below row $i$)
        For $k = i+1$ to $n$ {               (traverse row $j$)
            $A[j][k] = A[j][k] - \frac{A[j][i]}{A[i][i]} A[i][k]$
        }
        $b[j] = b[j] - \frac{A[j][i]}{A[i][i]} b[i]$
    }
}
```

At step i

Matrix

- 1 division $1/A[i][i]$
- $(n-i)$ multiplications $A[j][i] * \left(\frac{1}{A[i][i]}\right)$
- $(n-i)(n-i)$ " $\left(\frac{A[j][i]}{A[i][i]}\right) * A[i][k]$
- $(n-i)(n-i)$ additions $A[j][k] \; -$

RHS

- $(n-i)$ multiplications $\left(\frac{A[j][i]}{A[i][i]}\right) * b[i]$
- $(n-i)$ additions $b[j] \; -$

Total # of multiplication/ division

$$= 1 + (n-i) + (n-i)(n-i) + \underbrace{(n-i)}_{\text{RHS modification}}$$

$$= (n-i)(n-i+1) + \underbrace{(n-i)}_{\text{RHS}} + \underbrace{1}_{1/A[i][i]}$$

Sum these for i going from 1 to $n-1$

$$\sum_{i=1}^{n-1} (n-i)(n-i+1) = \frac{n^3-n}{3} \cong \frac{n^3}{3}$$

$$\sum_{i=1}^{n-1} (n-i) = \frac{n^2-n}{2} \cong \frac{n^2}{2}$$

<u>Remark</u>

Gaussian elimination requires $\sim \frac{n^3}{3}$ operations.

- Back Substitution

```
For i = n-1 to 1 {
    For j = i+1 to n {
        b[i] = b[i] - A[i][j] * x[j]
    }
    x[i] = b[i]/A[i][i]
}
```

∴ Total # of operations

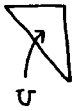
n divisions

$\sum_{i=1}^{n-1}(n-i)$ multiplications , additions $= \frac{n^2-n}{2} \sim \frac{n^2}{2}$

Remark

Back substitution requires $\sim \frac{n^2}{2}$ operations

Pictorally

$Ax = b \rightarrow$  $x = \hat{b}$

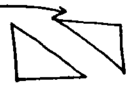
Problem

- For each new b we need to re triangularize the matrix. This can be expensive! Can we do something about it? YES

Save the multipliers a_{ij}/a_{jj} during triangularizatiou by modifying the elimination process

⇒ LU factorization

$Ax = LUx = b$



LU Decomposition or Factorization

$A = LU$

To solve $Ax = b \Rightarrow L(\underbrace{Ux}_{y}) = b$

First solve $Ly = b$ forward substitution

Then " $Ux = y$ back "

Let us consider how the RHS vector is updated during the GE process

$$b_1^{(1)} = b_1$$

$$b_2^{(2)} = b_2 - \frac{a_{21}}{a_{11}} b_1^{(1)} = b_2 - l_{21} b_1^{(1)}$$

$$\vdots$$

$$b_n^{(n)} = b_n - \frac{a_{n1}}{a_{11}} b_1^{(1)} - \frac{a_{n2}}{a_{22}} b_2^{(2)} \cdots - \frac{a_{n,n-1}}{a_{n-1,n-1}} b_{n-1}^{(n-1)}$$

$$= b_n - l_{n1} b_1^{(1)} - l_{n2} b_2^{(2)} \cdots - l_{n,n-1} b_{n-1}^{(n-1)}$$

or rearranging we have

$$b_1^{(1)} = b_1$$

$$l_{21} b_1^{(1)} + b_2^{(2)} = b_2$$

$$\ddots$$

$$l_{n1} b_1^{(1)} + l_{n2} b_2^{(2)} + \cdots + b_n^{(n)} = b_n$$

i.e. $L = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & & & \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix}$

Alternatively, one can start with the L & U matrices

$$\begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 \cdots & 0 \\ \vdots & & & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & & & \\ 0 & \cdots & 0 & u_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

# of unknowns in L $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

" " " in U $= \frac{n(n+1)}{2}$

$\Rightarrow$ Total # of unknowns $= n(n+1) = n^2+n$

However, we have n^2 equations

$\Rightarrow$ n degrees of freedom

The equations to be solved are:

$$l_{11} u_{11} = a_{11}$$
$$l_{11} u_{12} = a_{12}$$
$$\vdots$$
$$l_{n1} u_{1n} + l_{n2} u_{2n} + \cdots + l_{nn} u_{nn} = a_{nn}$$

Doolittle Set $l_{ii} = 1$, $i = 1, \ldots, n$ (same as in GE)

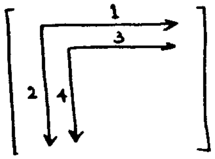
Grout Set $u_{ii} = 1$, $i = 1, \ldots, n$

Doolittle's Algorithm

$$l_{ii} = 1$$

The equations are solved in a particular sequence:

1) Solve first row of U
2) Solve first column of L
⋮



For row i:

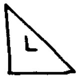
$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \qquad j = i, i+1, \ldots n$$

For column j:

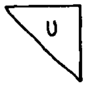
$$l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}\right) / u_{jj} \qquad i = j+1, \ldots, n$$

Operation Counts

1) LU factorization $\sim \frac{n^3}{3}$

2) Forward substitution 

$y = b$ $\sim \frac{n^2}{2}$ operations

3) Back substitution 

$x = y$ $\sim \frac{n^2}{2}$ operations

Remarks

- Operation count is the same as in GE
- In GE modify the remaining matrix whereas this is not the case in LU. Touch row/column entries
- When multiple RHS are used, LU factorization is done only once. The solution is obtained

by forward/back solve on each RHS. Thus the # of operations required is $\sim n^2/2$. With GE would requre $\sim n^3/3$ operations

- LU decomposition can be performed <u>in place</u> i.e., with the original matrix. No additional memory is required. Once an element in U or L is computed the corresponding element of A is not requred.

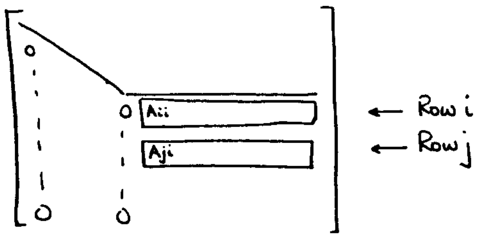
Summary of LU Factorization

To solve $Ax = b$
decompose $A \rightarrow LU$

Then
Solve $Ly = b$ for y (FS)
Solve $Ux = y$ for x, the solution (BS)

Problems with Gaussian Elimination

At step i



Suppose $A_{ii} = 0$
Cannot perform division by A_{ii} to get $\frac{A_{ji}}{A_{ii}}$

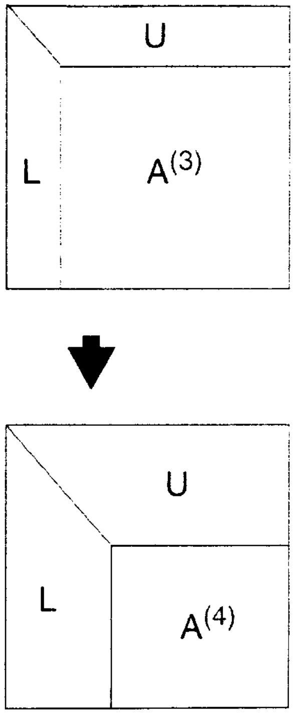
So how does one proceed from here

Pivoting for Accuracy

<u>Example 1:</u> After two steps of G.E. MNA matrix becomes:

Pivots

$$\begin{bmatrix} x & x & 0 & 0 & 0 & 0 \\ 0 & x & 0 & 0 & x & 0 \\ 0 & 0 & \frac{1}{R} & -\frac{1}{R} & 1 & 0 \\ 0 & 0 & -\frac{1}{R} & \frac{1}{R} & 0 & 1 \\ 0 & 0 & x & 0 & x & 0 \\ 0 & 0 & 0 & x & x & 0 \end{bmatrix}$$



$$\begin{bmatrix} x & x & 0 & 0 & 0 & 0 \\ 0 & x & 0 & 0 & x & 0 \\ 0 & 0 & \frac{1}{R} & -\frac{1}{R} & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & x & x & 0 \\ 0 & 0 & 0 & x & x & 0 \end{bmatrix}$$

pivot = 0

$$l_{i4} = \frac{a_{i4}^{(4)}}{a_{44}^{(4)}} = \infty \text{ !!!}$$

0

From ASV

Solution:

Interchange rows and/or columns to bring non-zero element into position (k,k):

$$\begin{bmatrix} 0 & 1 & 1 \\ x & x & 0 \\ x & x & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} x & x & 0 \\ 0 & 1 & 1 \\ x & x & 0 \end{bmatrix}$$

Example 2:

$$\begin{bmatrix} 1.25 \times 10^{-4} & 1.25 \\ 12.5 & 12.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6.25 \\ 75 \end{bmatrix}$$

solution to 5 digit accuracy

$x_1 = 1.0001$

$x_2 = 5.0000$

From ASV

Assumption:

Finite Arithmetic: 3 digit floating point

Gaussian Elimination produces

$x_1 = 0, x_2 = 5$

However, using row interchange

$x_1 = 1, x_2 = 5$

$$\begin{bmatrix} 1.25 \times 10^{-4} & 1.25 \\ 12.5 & 12.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6.25 \\ 75 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 1.25 \times 10^{-4} & 1.25 \\ 0 & -1.25 \times 10^{5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6.25 \\ -6.25 \times 10^{5} \end{bmatrix}$$

$$\Downarrow$$

$$x_2 = 5$$

$$(1.25 \times 10^{-4})\, x_1 + (1.25)\, x_2 = 6.25$$

$$\Downarrow$$

$$x_1 = 0$$

WHY ?

a_{11} is too small relative to the other numbers

From ASV

Pivoting Strategies

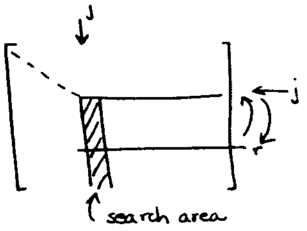
Partial pivoting (row interchange only)

Select r such that

$$|\hat{a}_{rj}| = \max_{i=j\dots n} |\hat{a}_{ij}|$$

and interchange rows r and j

i.e. find the maximum in the column.

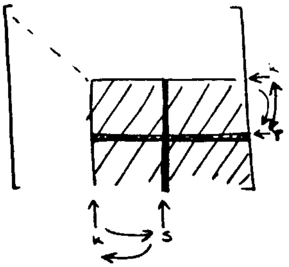


Complete pivoting (row and column interchange)

Select r and s such that

$$|\hat{a}_{rs}| = \max_{\substack{i=k,\dots n \\ j=k,\dots n}} |\hat{a}_{ij}|$$

and interchange rows r and k and columns s and k.



i.e. find the maximum in the submatrix.
There is some gain but this is an expensive process.

Threshold pivoting

- apply partial pivoting only if
  $|\hat{a}_{kk}| < \epsilon_p |\hat{a}_{rk}|$

- apply complete pivoting only if
  $|\hat{a}_{kk}| < \epsilon_p |\hat{a}_{rs}|$

where ϵ_p is a relative pivot tolerance

The next question we must ask is how accurately can we solve $Ax = b$?

$\hat{x}$ = computed solution
x = Exact "

The error is $e = \hat{x} - x$

Error mechanisms

- ill conditioning i.e. almost singular matrix. In this case we are out of luck

[rounding
numerical stability of method

Review of Linear Algebra

Consider m vectors $x_1, x_2, \dots x_m \in \mathbb{R}^n$
scalar $\alpha_i \in \mathbb{R}$

Span set of all $y \in \mathbb{R}^n$ such that $y = \sum_{i=1}^{m} \alpha_i x_i$

Basis If span $\{x_1, \dots x_n\} = \mathbb{R}^n$ then $x_1, \dots x_n$ is a basis for $\mathbb{R}^n$

Linear Independence (LI) vectors $x_1, \dots x_m$ are LI if

$$\sum_{i=1}^{m} \alpha_i x_i = 0 \quad \Rightarrow \alpha_i = 0 \quad \text{for all } i$$

Example

$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ are LI

$$\alpha_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0 \quad \Rightarrow \alpha_1 = 0, \ \alpha_2 = 0$$

Linear System of Equations

$$Ax = b \quad , \quad x, b \in \mathbb{R}^n$$

$$\begin{bmatrix} \bar{c}_1 & \cdots & \bar{c}_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = b$$

A is nonsingular if $\forall\, b \in \mathbb{R}^n$ there exists an $x \in \mathbb{R}^n$ s.t. $Ax = b$

OR

$$\text{Span}\{\bar{c}_1, \cdots \bar{c}_n\} = \mathbb{R}^n$$

OR

$\bar{c}_1, \bar{c}_2, \cdots \bar{c}_n$ are LI

OR

If $Ax = b$ then x is unique. i.e., there is no other $\tilde{x}$ for which $A\tilde{x} = b$

Norms

Vector Norms

L_1 : $\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|$

L_2 : $\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$

L_∞ : $\|x\|_\infty = \max\limits_{1 \le i \le n} |x_i|$

Properties of Norms

$$\|x\| = 0 \quad \text{iff} \quad x = 0$$
$$\|\alpha x\| = |\alpha| \, \|x\|$$
$$\|x + y\| \le \|x\| + \|y\|$$

Induced Matrix Norms

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}$$

L_1 : $\|A\|_1 = \max\limits_{1 \le j \le n} \sum\limits_{i=1}^{n} |a_{ij}|$ (max column sum)

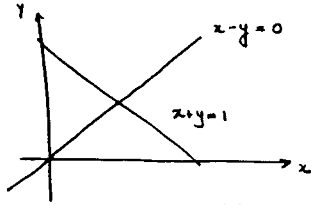
L_2 : $\|A\|_2 = \sqrt{\text{largest eigenvalue of } AA^T}$

L_∞ : $\|A\|_\infty = \max\limits_{1 \le i \le n} \sum\limits_{j=1}^{n} |a_{ij}|$ (max row sum)

For matrices $\|AB\| \le \|A\| \, \|B\|$.

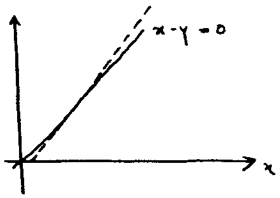
Ill conditioning

1) $x + y = 1$
$x - y = 0$



Well-conditioned system.
Slight perturbations do not change the solution significantly

2) $x - y = 0$
$x - 1.01y = +0.01$



Ill-condition system.
A perturbation in one of the lines can result in a large change in solution.

Remarks

- When ill-conditioning is a problem, an attempt must be made to improve the system of equations.
- Extended precision arithmetic will provide a solution of better quality.

Perturbation Analysis

$A \rightarrow A + \delta A$ due to round-off in GE

$x \rightarrow x + \delta x$ the change in solution due to the uncertainty in A

So we are solving

$$(A+\delta A)(x+\delta x) = b$$

$$\cancel{Ax} + A\delta x + \delta A\, x + \delta A\,\delta x = \cancel{b}$$

$$\Rightarrow A\,\delta x + \delta A\,(x+\delta x) = 0$$

or $$\delta x = -A^{-1}\,\delta A\,(x+\delta x)$$

whereby $$\|\delta x\| \le \|A^{-1}\|\,\|\delta A\|\,\|x+\delta x\|$$

∴ relative error $$\frac{\|\delta x\|}{\|x+\delta x\|} \le \underbrace{\|A\|\,\|A^{-1}\|}_{\kappa(A)}\,\frac{\|\delta A\|}{\|A\|}$$

$\kappa(A) = \|A\|\,\|A^{-1}\|$ is called the condition number of A.

A large condition number is bad ⇒ ill conditioned matrix

Example

$$\begin{bmatrix} 1 & 0 \\ 0 & 10^{-6} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \qquad \|A\|_\infty\,\|A^{-1}\|_\infty = 10^6$$

This problem is not hard to solve: a diagonal scaling works

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ 10^6 b_2 \end{bmatrix} \qquad \|A\|_\infty\,\|A^{-1}\|_\infty = 1$$

Suppose there is a perturbation in b

$b \rightarrow b + \delta b$

$x \rightarrow x + \delta x$

What we are solving is

$$A(x+\delta x) = b + \delta b$$

$$\cancel{Ax} + A\,\delta x = \cancel{b} + \delta b$$

$$\therefore\ \delta x = A^{-1}\,\delta b$$

or $$\|\delta x\| \le \|A^{-1}\|\,\|\delta b\|$$

$$\therefore\ \frac{\|\delta x\|}{\|x\|} \le \|A\|\,\|A^{-1}\|\,\frac{\|\delta b\|}{\|b\|} = \kappa(A)\,\frac{\|\delta b\|}{\|b\|}$$

(because $\dfrac{1}{\|x\|} \le \dfrac{\|A\|}{\|b\|}$ from $\|b\| \le \|A\|\,\|x\|$)

Remark

When $\kappa(A)$ is large, a small relative perturbation in A and b produces large relative perturbations in x.

Rounding

Let ϵ be the machine precision or resolution for a floating point number

e.g. single precision $\epsilon \cong 10^{-8}$
double " $\epsilon \cong 10^{-16}$

$\Rightarrow$ $1 + \epsilon = 1$ whereby ϵ is the machine precision

i.e. $1 + 10^{-16} = 1$ but $1 + 10^{-15} \neq 1$.

Thus for any floating point number a, $\bar{a}$ is its machine representation and

$$|a - \bar{a}| \le \epsilon\,|a|$$

Thus $\|\delta A\|_\infty = \|\bar{A} - A\|_\infty \leq \epsilon \|A\|_\infty$

$\|\delta b\|_\infty = \|\bar{b} - b\|_\infty \leq \epsilon \|b\|_\infty$

and $\|\delta x\|_\infty \leq \|\delta x_A\|_\infty + \|\delta x_b\|_\infty \leq 2\epsilon\, \kappa(A)\, \|x\|_\infty$

Remark

Even if the algorithm is perfect we still have an error in the solution on the computer.

Growth during solution

$$A = \begin{bmatrix} 0.1 & & & & 1 \\ 1 & 0.1 & & & 1 \\ & 1 & 0.1 & & 1 \\ & & 1 & 0.1 & 1 \\ & & & 1 & 0.1 \end{bmatrix}$$

Step 1 of GE

$$A^{(1)} = \begin{bmatrix} 0.1 & & & & 1 \\ 0 & 0.1 & & & -9 \\ & 1 & 0.1 & & 1 \\ & & 1 & 0.1 & 1 \\ & & & 1 & 0.1 \end{bmatrix} \leftarrow (1-10)$$

Step 2 of GE

$$A^{(2)} = \begin{bmatrix} 0.1 & & & & 1 \\ 0 & 0.1 & & & -9 \\ & 0 & 0.1 & & 1+90 \\ & & 1 & 0.1 & 1 \\ & & & 1 & 0.1 \end{bmatrix}$$

We have a problem here due to round off in the subtraction process

1 − a large number

Let us now look at partial pivoting

swap row i & j if $|a_{ij}| > |a_{ii}|$

Step 0

$$\begin{bmatrix} 1 & 0.1 & & & 1 \\ 0.1 & 0 & & & 1 \\ & 1 & 0.1 & & 1 \\ & & 1 & 0.1 & 1 \\ & & & 1 & 0.1 \end{bmatrix}$$

Step 1.

$$\begin{bmatrix} 1 & 0.1 & & & 1 \\ 0 & -0.01 & & & 0.9 \\ & 1 & 0.1 & & 1 \\ & & 1 & 0.1 & 1 \\ & & & 1 & 0.1 \end{bmatrix}$$

Reorder

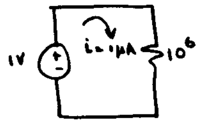
$$\begin{bmatrix} 1 & 0.1 & & & 1 \\ 0 & 1 & 0.1 & & 1 \\ 0 & -0.01 & 0 & & 0.9 \\ & & 1 & 0.1 & 1 \\ & & & 1 & 0.1 \end{bmatrix}$$

Step 2.

$$\begin{bmatrix} 1 & 0.1 & & & 1 \\ 0 & 1 & 0.1 & & 1 \\ 0 & 0 & 0.001 & & 0.91 \\ & & 1 & 0.1 & 1 \\ & & & 1 & 0.1 \end{bmatrix}$$

Scaling and Equilibration

Consider



$$\begin{bmatrix} 10^{-6} & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} V_1 \\ I_E \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

There are different physical quantities V & I in the above system of equations (Volts & Amps)
$\Rightarrow$ widely varying element sizes.

In the above example suppose current is solved in μA
i.e. $\hat{I}_E = I_E / 10^{-6}$ or $I_E = 10^{-6}\hat{I}_E$
Then the above matrix becomes

$$\begin{bmatrix} 10^{-6} & 10^{-6} \\ 1 & 0 \end{bmatrix}\begin{bmatrix} V_1 \\ \hat{I}_E \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Here we have scaled an unknown $- I_E$.

In general

multiplying every element in column j by a constant α_j is equivalent to scaling the jth unknown
i.e. $\hat{x}_j = \frac{x_j}{\alpha_j}$

Similarly, multiplying every element in row i by β_i scales the ith equation i.e. $\hat{b}_i = \beta_i b_i$.

Thus, we obtain a transformed system
$Ax = b \longrightarrow \hat{A}\hat{x} = \hat{b}$

i.e. $D_2 A D_1 x = D_2 b$
$D_1 = \text{diag}(\alpha_1, \cdots \alpha_n)$ and $D_2 = \text{diag}(\beta_1 \cdots \beta_n)$

Example

$$\begin{bmatrix} 1 & 10^4 \\ 1 & 10^{-4} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 10^4 \\ 1 \end{bmatrix}$$

Solution $x_1 = x_2 = 0.9999$ to four decimals.

Using 3 digit accuracy, the computed solution is
(a_{11} pivot) $x_2 = 1$, $x_1 = 0$ which is incorrect

Now multiply first equation by 10^{-4}

$$\Rightarrow \begin{bmatrix} 10^{-4} & 1 \\ 1 & 10^{-4} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Using partial pivoting a_{21} is the pivot
$\Rightarrow x_2 = 1$, $x_1 = 1$ which is a good solution

Remarks

- The effect of scaling on the accuracy of a solution not well understood
- A more accurate answer will result if before pivoting the matrix is scaled such that all elements in the matrix are about the same size $\Rightarrow$ equilibration

Row equilibration $\max_{1 \le j \le n} |a_{ij}| \cong 1$

Column equilibration $\max_{1 \le i \le n} |a_{ij}| \cong 1$

Matrix is equilibrated if it is both row & column equilibrated.

Example

$$A = \begin{bmatrix} \epsilon & -1 & 1 \\ -1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad A^{-1} = \frac{1}{4}\begin{bmatrix} 0 & -2 & 2 \\ -2 & 1-\epsilon & 1+\epsilon \\ 2 & 1+\epsilon & 1-\epsilon \end{bmatrix} \qquad |\epsilon| \ll 1$$

$\kappa(A) = 3 \Rightarrow$ GE with partial pivoting will give an accurate solution

Choice of $a_{11} = \epsilon$ as pivot will have a disastrous effect on the accuracy of the computed solution.

Now scale $\hat{x}_2 = \frac{x_2}{\epsilon}$, $\hat{x}_3 = \frac{x_3}{\epsilon}$

$$\hat{A} = \begin{bmatrix} \epsilon & -\epsilon & \epsilon \\ -1 & \epsilon & \epsilon \\ 1 & \epsilon & \epsilon \end{bmatrix} \xrightarrow{\text{equilibrate}} \begin{bmatrix} 1 & -1 & 1 \\ -1 & \epsilon & \epsilon \\ 1 & \epsilon & \epsilon \end{bmatrix}$$

$\kappa(\hat{A}) = \frac{3}{\epsilon}$ i.e. large $\Rightarrow$ inaccurate solution

Does row-scaling reduce growth if pivot order is unchanged?

No.

$$Ax = b \xrightarrow[\text{scaling}]{\text{row}} DAx = Db$$

Consider row 2 of matrix

$$\text{scaled row 2} = \beta_2 \left[a_{21} \quad a_{22} \quad \cdots \quad a_{2n} \right]$$

$$\hat{a}_{22} = \beta_2 a_{22} - \frac{\beta_2 a_{22}}{\cancel{\beta_1} a_{11}} \cancel{\beta_1} a_{12}$$

$$= \underset{\uparrow \text{ just a scaling}}{\beta_2} \left(a_{22} - \frac{a_{22}}{a_{11}} a_{12} \right)$$

There is no roundoff reduction and hence growth is not affected

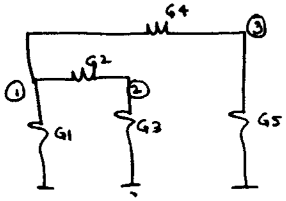
Pivoting for sparsity

First we define two terms structural zero and fill-in

Structural zero in a matrix is a matrix element that is zero regardless of the element values.

Example.

$$\begin{bmatrix} G_1+G_2+G_4 & -G_2 & -G_4 \\ -G_2 & G_2+G_3 & 0 \\ -G_4 & 0 & G_4+G_5 \end{bmatrix}$$



These are structural zeros because there is no connection between nodes ① & ③

Fill-in A structural zero that becomes nonzero during factorization

$$\begin{bmatrix} x & x & x \\ x & x & 0 \\ x & 0 & x \end{bmatrix} \xRightarrow[\text{of LU}]{\text{first step}} \begin{bmatrix} x & x & x \\ x & x & \textcircled{x} \\ x & 0 & x \end{bmatrix} \leftarrow \text{fill-in}$$

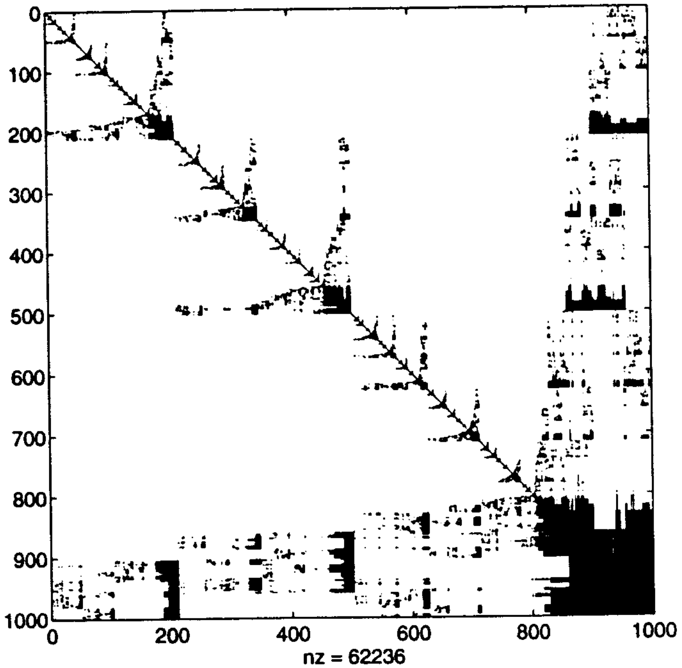
If the above matrix was reordered as

$$\begin{bmatrix} x & & x \\ & x & x \\ x & x & x \end{bmatrix} \xRightarrow[\text{LU}]{\text{first step}} \begin{bmatrix} x & & x \\ & x & x \\ x & x & x \end{bmatrix} \quad \text{No fill-in}$$

Fill-ins can be very bad in terms of sparsity because they can create additional fills

Reordering can reduce the number of fill ins.

MATRIX PATTERN
(LU decomposed with fill-ins)



Key Idea

Where can a fill-in occur?

$$\begin{matrix} x & x & x \\ x & \text{here} & \text{here} \\ x & \text{here} & \text{here} \end{matrix}$$

An estimate for fill-ins is (# in row − 1) (# in col − 1)

excludes diagonal

The Markowitz Criterion is based on minimizing this product

Example

$$A = \begin{array}{c|c} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} & \begin{matrix} 4 \\ 2 \\ 2 \\ 3 \end{matrix} \\ \hline \begin{matrix} 2 & 4 & 3 & 2 \end{matrix} & \end{array} \Rightarrow \underbrace{\begin{bmatrix} 3 & 9 & 6 & 3 \\ 1 & 3 & - & - \\ - & 3 & 2 & - \\ - & 6 & 4 & 2 \end{bmatrix}}_{\text{Markowitz products}}$$

Therefore, one would choose a_{21} as the pivot

Typical Markowitz ordering on circuit matrices uses diagonal pivoting

⇒ decrease cost of Markowitz counts

⇒ preserves matrix numerical properties

- NA matrix is diagonally dominant
- MNA is "close" to NA

Markowitz Reordering

```
for i = 1, N
    1) Find  $j \geq i$  to minimize Markowitz product
           (# of nonzero in row -1)(# of nonzero in col -1)

    2) Swap rows $j$ & $i$ and columns $j$ and $i$ so
       that $a_{jj}$ is the pivot
    3) Determine fill-ins and update Markowitz
       products
```

Other Criterion is the minimum fill-in criterion. Here the next pivot is the element that produces the minimum number of fill-ins.

The actual number of fill-ins is computed for each potential pivot

Remarks

- This reordering can give quite a different ordering than Markowitz
- Experimentally Markowitz is fast and works well.

What happens when there is a tie for pivots?

Use tie breaking rules

NZUR = # non zeros in an upper triangular row including diagonal

NZLC = # non-zeros in a lower triangular col including diagonal

Now let's look at various ordering & tie breaking strategies

Markowitz:
1) min (NZUR-1) * (NZLC-1)
2) min NZLC (avoid divisions for determining multipliers)

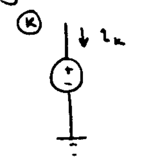
Berry:
1) min fill-in
2) max NZUR
3) max NZLC

Get rid of as many non-zero elements as possible

Hsieh:
1) Row singletons (NZUR = 1)
2) Col singletons (NZLC = 1)
3) (NZUR, NZLC) = (2,2), (2,3), (2,4), (4,2), (3,3)
4) min (NZUR-1) * (NZLC-1)
5) max NZLC

Nakhla:
1) min fill-in
2) min NZUR
3) min NZLC

Exploiting MNA structure



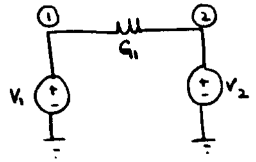
$$\begin{array}{rcc} & k & i_k \\ k & 0 & 1 \\ \text{branch} & \multicolumn{2}{c}{\underbrace{1 \qquad 0}_{\text{singleton}}} \end{array}$$

Have a singleton in a row ⇒ excellent pivot choice (Markowitz count = 0)

However, not on the main diagonal

⇒ preprocess the matrix by pre ordering for MNA

Example



$$\begin{bmatrix} G_1 & -G_1 & 1 & 0 \\ -G_1 & G_1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

swap rows 1 & 3 and 2 & 4

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ G_1 & -G_1 & 1 & 0 \\ -G_1 & G_1 & 0 & 1 \end{bmatrix}$$

So by preprocessing the row singletons have been made pivots !

SPICE2G6 pivoting strategy

1) Choose Markowitz element on main diagonal
2) Check with largest element in column $|a_{max}|$

   if $\quad < \epsilon_a + \epsilon_r |a_{max}|$ $\qquad \underset{\text{PIVTOL}}{\epsilon_a} = 10^{-13}$, $\underset{\text{PIVREL}}{\epsilon_r} = 10^{-3}$

   reject

   Note $\epsilon_a = 10^{-13}$ is tied to GMIN.

   If GMIN $< 10^{-12}$ PIVTOL should be appropriately adjusted
3) If rejected go back to 1).

   If all elements on diagonal fail test, select pivot outside the diagonal.

Now that we know how to solve linear algebraic equations let us revisit the issue of matrix sparsity.

Example

$n = 1000$ equations

dense matrix ⇒ storage for 1000 x 1000 numbers
i.e. 10^6 numbers

a double precision number requires 8 bytes of storage
⇒ 8 M bytes storage

Gaussian Elimination $\sim n^3$ operations
$\sim 10^9$ flops

Why should we do dense matrix solution when the circuit matrices are sparse ?

If we exploit sparsity

- MNA $\sim$ 3 nonzeros/row ⇒ 3000 nonzeros
  storage for 3000 NZ + some overhead
  $\cong$ 6000 double precision numbers
  $=$ 48 KB storage
- GE $\sim n^{1.1} - n^{1.5}$ or 2k − 32k flops.

We need to understand how to handle sparse matrices for reduced storage and computational effort
⇒ sparse matrix technology (SMT)

Issues in SMT

- Do not store zeros. Use specialized data structures
- Avoid trivial operations i.e. $0x = 0$, $0+x = x$,
- Avoid loosing sparsity ⇒ pivoting for sparsity

Sparse Matrix Data Structures

Example $$A = \begin{bmatrix} 5 & 3 & 0 & 1 \\ 7 & 1 & 0 & 0 \\ 0 & 2 & 3 & 0 \\ -1 & 0 & 5 & 2 \end{bmatrix}$$

Simplest way to store the sparse matrix is the use of 3 arrays A_{ij}, I, J which record the nonzero pattern

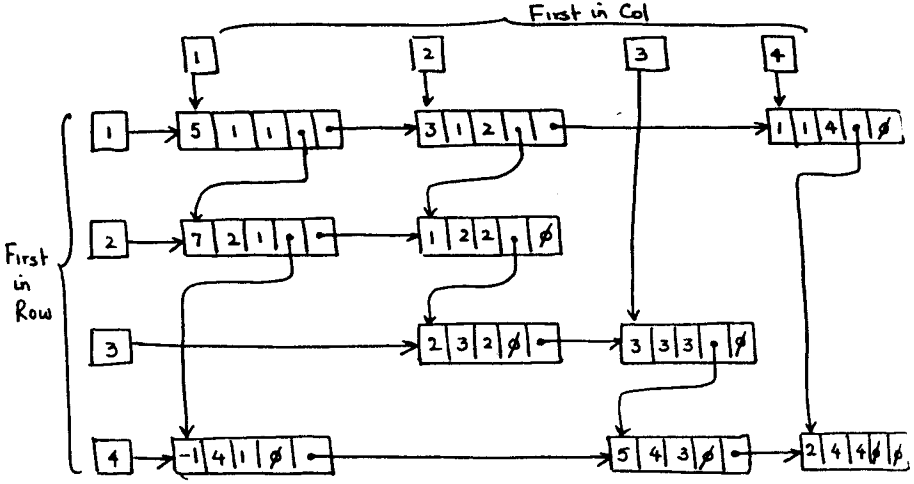
$$\underset{A_{ij}}{\begin{bmatrix}5\\3\\1\\7\\1\\2\\3\\-1\\5\\2\end{bmatrix}} \qquad \underset{I}{\begin{bmatrix}1\\1\\1\\2\\2\\3\\3\\4\\4\\4\end{bmatrix}} \qquad \underset{J}{\begin{bmatrix}1\\2\\4\\1\\2\\2\\3\\1\\3\\4\end{bmatrix}}$$

With this arrangement only need to store the nonzeros. However this is not an efficient arrangement when one wants to do GE or LU factorization on the matrix

Requirements of storage technique

- storage requirements
- processing flexibity
- " efficiency

The Bidirectional Threaded List



Also have an array Diag with pointers to the diagonal elements.

Diag [i] = diagonal in row i i.e. a_{ii}

Storage Requirements

2 n for row & column pointers (First In Row/Col)
3 NZ for data and row/column indices
2 NZ for pointers next In Row/Col

Remark

- The list structure is better because it allows for easy addition and deletion of elements

Now let us look at some functions that will facilitate working with sparse matrices using the above data structure

createElement (Matrix, Row, Col)
- creates and splices a new element in the matri

getElement (Matrix, Row, Col)
- returns element if it exists otherwise calls createElement

crateFillin (Matrix, Row, Col)
- creates a fillin

ExchangeRowAndCol (Matrix, Row1, Row2, Col1, Col2)
- this has to be done with care because there may be missing elements in a row/col.

Remarks

- In a list an element can be accessed only by traversing the list. ⇒ search.
- This can be a problem when stamping into the matrix because the appropriate matrix element has to be found
- To overcome this overhead use direct pointers to the matrix elements during the construction phase

Example

R n1 n2 val

$$\begin{array}{c|cc} & n1 & n2 \\ \hline n1 & G & -G \\ n2 & -G & G \end{array}$$

⇒ matrix locations (n1,n1), (n1, n2)
(n2,n1), (n2, n2)

Have an array of pointers to each of these locations
⇒ storage of 4 pointers for a resistor!

Implementation of Threaded Lists for Sparse Matrices

C (or a structure based language)

```
Element = struct {
                   double          value
                   int             row
                   int             col
                   (ptr) Element   * nextInCol
                     Element       * nextInRow
                 }
```

FORTRAN.

$$\begin{bmatrix} 5 & 3 & 0 & 1 \\ 7 & 1 & 0 & 0 \\ 0 & 2 & 3 & 0 \\ -1 & 0 & 5 & 2 \end{bmatrix}$$

	VALUE	IROW	JCOL	IPT	JPT
(1)	5	1	1	2	4
(2)	3	1	2	3	5
(3)	1	1	4	0	10
(4)	7	2	1	5	8
(5)	1	2	2	0	6
(6)	2	3	2	7	0
(7)	3	3	3	0	9
(8)	-1	4	1	9	0
(9)	5	4	3	10	0
(10)	2	4	4	0	0

(next elem in Row) → IPT

JPT ↑ (next elem in Col)

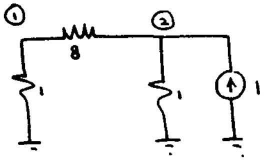
IIPT = (row starts) $\begin{bmatrix} 1 \\ 4 \\ 6 \\ 8 \end{bmatrix}$

JJPT = (col starts) $\begin{bmatrix} 1 \\ 2 \\ 7 \\ 3 \end{bmatrix}$

Iterative solution methods

Relaxation Methods

$$\begin{bmatrix} 9/8 & -1/8 \\ -1/8 & 9/8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$



Solution Using GE is

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$$

Relaxation idea

Guess x_2^0, solve for x_1^1
Use x_1^1 (or x_1^0) and solve for x_2^1
Repeat the process

Example

$x_1^0 = x_2^0 = 0;$

$$x_1^1 = 0 \quad ; \quad x_1^k = 1/9\, x_2^{k-1}$$

$$x_2^1 = \frac{8}{9} = 0.8889 \quad \left(x_2^k = \frac{1}{9}(8 + x_1^k)\right)$$

$$x_1^2 = \frac{\cancel{8}}{9}\left(\frac{1}{8} x_2^1\right) = \frac{8}{81} = 0.0988$$

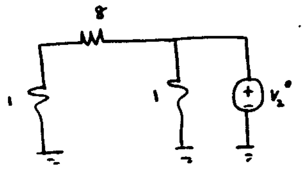
$$x_2^2 = \frac{8}{9}\left(1 + \frac{1}{8} x_1^2\right) = 0.8999$$

$$x_1^3 = 0.09999$$

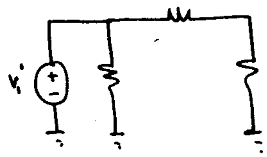
$$x_2^3 \cong 0.9$$

So we see that the iterative process converges to the correct solution.

Circuit interpretation



Calculation of v_1^1



Calculation of v_2^1

Suppose $A = \begin{bmatrix} 1/8 & -9/8 \\ -9/8 & 1/8 \end{bmatrix}$

The solution using GE $= \begin{bmatrix} -0.9 \\ -0.1 \end{bmatrix}$

Let us consider the relaxation process

$$\begin{bmatrix} 1/8 & -9/8 \\ -9/8 & 1/8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The iteration formula is

$$x_1^{k+1} = 9x_2^k$$
$$x_2^{k+1} = 8 + 9x_1^{k+1}$$

Start with $x_2^0 = 0 \Rightarrow$

$$x_1^1 = 0$$
$$x_2^1 = 8$$
$$x_1^2 = 72$$
$$x_2^2 = 656$$

The solution quickly diverges!

Questions

- Under what conditions does the scheme converge?
- How fast does the scheme converge? Should not take forever!
- If the scheme converges to some $\hat{x}$, does $\hat{x}$ solve the original problem?

What do we mean by convergence?

Let x^* be the solution of $Ax = b$
Let $x^{(i)}$ be the i-th iterate of the iterative meth.

Define $e^{(i)} = x^{(i)} - x^*$ as the error (ith iteration

Convergence if $\|e^{(i)}\|_2 \to 0$ as i increases independent of $e^{(0)}$ (or the initial guess $x^{(0)}$).

Consider the linear system $Ax = b$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Gauss Jacobi Method

```
For k = 1 to convergence {
    For i = 1 to n {
        $a_{ii} x_i^{(k)} = \left[ b_i - \sum_{j \neq i} a_{ij} x_j^{(k-1)} \right]$
    }
}
```

$$\sum_{j \neq i} a_{ij} x_j^{(k-1)} = \sum_{j=1}^{i-1} a_{ij} x_j^{(k-1)} + \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)}$$

Gauss-Seidel

```
For k = 1 to convergence {
    For i = 1 to n {
        $a_{ii} x_i^{(k)} = \left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} \right)$
```

Matrix Description for these matrices

$$A \equiv L + D + U$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 0 & & & \\ a_{21} & 0 & & \\ a_{31} & a_{22} & 0 & \\ \vdots & & \ddots & \\ a_{n1} & a_{n2} & \cdots\ a_{n,n-1} & 0 \end{bmatrix} + \begin{bmatrix} a_{n} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{bmatrix} + \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & a_{23} \cdots & a_{2n} \\ & & \ddots & a_{n-1,n} \\ & & & 0 \end{bmatrix}$$

Gauss-Jacobi

$$D x^{(k)} = b - (L+U) x^{(k-1)}$$

$$\therefore\ x^{(k)} = -D^{-1}(L+U) x^{(k-1)} + D^{-1} b$$

Gauss-Seidel

$$(L+D) x^{(k)} = b - U x^{(k-1)}$$

$$\therefore\ x^{(k)} = -(L+D)^{-1} U x^{(k-1)} + (L+D)^{-1} b$$

Compute the errors from iteration k-1 to k.

G-J
$$x^{(k)}-x^{(k-1)} = -D^{-1}(L+U)\,x^{(k-1)} + D^{-1}b - \underbrace{x^{(k-1)}}_{\left(-D^{-1}(L+U)\,x^{(k-2)}+D^{-1}b\right)}$$
$$= \underbrace{-D^{-1}(L+U)}_{M_{GJ}}\left(x^{(k-1)}-x^{(k-2)}\right)$$

G-S
$$x^{(k)}-x^{(k-1)} = -(L+D)^{-1}U\,x^{(k-1)} + (L+D)^{-1}b$$
$$= \underbrace{-(L+D)^{-1}U}_{M_{GS}}\left(x^{(k-1)}-x^{(k-2)}\right)$$

Now $x^{(k)}$ can be written as

$$x^{(k)} = \left(x^{(k)}-x^{(k-1)}\right) + \left(x^{(k-1)}-x^{(k-2)}\right) + \cdots + \left(x^{(1)}-x^{(0)}\right) + x^{(0)}$$
$$= \sum_{i=0}^{k-1} M^{i}\left(x^{(1)}-x^{(0)}\right) + x^{(0)}.$$

Under what conditions on M can we ensure that the above series converges

Scalar case $\quad \sum \alpha^{i}$ is finite iff $|\alpha|<1$

Matrix/vector case $\quad \sum M^{i}\left(x^{(1)}-x^{(0)}\right)$ is finite iff $\max_i |\lambda_i| < 1$ where λ is an eigenvalue of M.

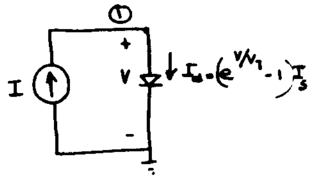
Remarks

- If A is strictly diagonally dominant, G-J and G-S converge
- If G-J converges then G-S converges faster than G-J
  " " diverges " " diverges " " "

ECE 521

Nonlinear circuits

$$I_d = I_s\left(e^{V/V_T}-1\right)$$



where $V_T = \dfrac{kT}{q} \cong 0.0258\ \text{V}$

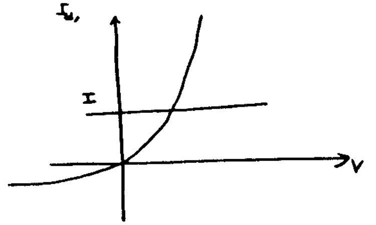
Given a value of I how do we determine V?

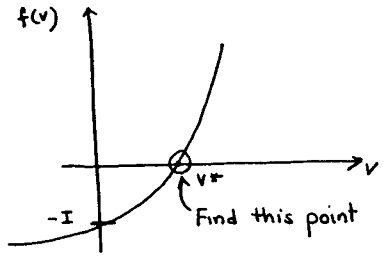
In this case one can come up with the analytical solution
$$V = V_T \ln\left(\frac{I_d}{I_s}+1\right) = V_T \ln\left(\frac{I}{I_s}+1\right)$$

For $I_s = 10^{-16}$, $I = 10^{-3}$ $\quad V = 0.772\,\text{V}$

How do we solve this problem numerically?

KCL @ ① $\Rightarrow \quad -I + I_d = -I + I_s\left(e^{V/V_T}-1\right) = f(V) = 0$





The problem is: Given f(V) = 0, solve V

The idea is to use an iterative process that generates a sequence $\{V^{(k)}\}$ such that $V^{(k)} \to V^*$, the solution, starting from an initial guess $V^{(0)}$.

Suppose v^0 is close to the solution v^*, then a Taylor's series expansion gives

$$0 = f(v^*) = f(v^0) + \frac{\partial f}{\partial v}\Big|_{v^0}(v^*-v^0) + \frac{1}{2}\frac{\partial^2 f}{\partial v^2}\Big|_{\hat{v}}(v^*-v^0)^2$$

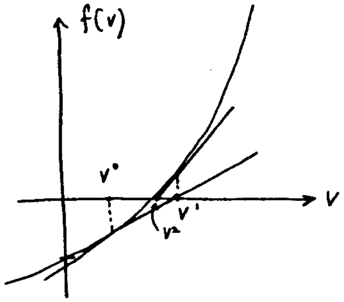
$$\text{where } \hat{v} \in [v^0, v^*]$$

Since v^0 is close to v^* we have

$$f(v^0) + \frac{\partial f}{\partial v}\Big|_{v^0}(v^*-v^0) \simeq 0$$

$$\text{or} \quad \frac{\partial f}{\partial v}\Big|_{v^0}(v^*-v^0) = -f(v^0)$$

Newton's method is based on the above idea. The iterates are generated by using a linearization of the function at each iteration



At iteration k+1

$$f'(v^{(k)}) = \frac{-f(v^{(k)})}{v^{(k+1)}-v^{(k)}}$$

This is seen from the following

$$f(v) \doteqdot f(v^{(k)}) + f'(v^{(k)})(v-v^{(k)})$$

Solve $f(v)=0 \Rightarrow v^{(k+1)}$ i.e. $f(v^{(k)}) + f'(v^{(k)})(v^{(k+1)}-v^{(k)}) = 0$

$$\text{or} \quad v^{(k+1)} = v^{(k)} - f(v^{(k)})/f'(v^{(k)})$$

The sequence $v^0, v^{(1)}, v^{(2)} \dots v^*$ is such that $f(v^*)=0$

Examples

1) Diode Example : $f(v) = -I + I_s(e^{v/v_T}-1)$ $(v^* = 0.772)$

$$f'(v) = \frac{I_s}{V_T}e^{v/v_T}$$

Newton's Method $\Rightarrow$ $v^{k+1} = v^k - \dfrac{[-I + I_s(e^{v^k/v_T}-1)]}{\frac{I_s}{V_T}e^{v_k/v_T}}$

2) Linear example: $f(x) = x-1$ $(x^* = 1)$

$f'(x) = 1$

Newton's Method $\Rightarrow$ $x^{k+1} = x^k - \dfrac{(x^k-1)}{1} = 1$

i.e. converges to the exact solution regardless of the initial guess

3) Slightly nonlinear example: $f(x) = x^2-1$ $(x^* = 1)$

$f'(x) = 2x$

Newton's method $\Rightarrow$ $x^{k+1} = x^k - \dfrac{((x^k)^2-1)}{2x^k}$

$$= \frac{x^k}{2} + \frac{1}{2x^k}$$

$x^0 = 0.5$: $x^1 = \dfrac{0.5}{2} + \dfrac{1}{2\times 0.5} = 1.25$

$$x^2 = \frac{1.25}{2} + \frac{1}{2\times 1.25} = 1.025$$

$$x^3 = \frac{1.025}{2} + \frac{1}{2\times 1.025} = 1.0003$$

$$x^4 = 1.00$$

$|x^1-1| = 0.25$, $|x^2-1| = 0.025$, $|x^3-1| = 0.0003$

i.e. $|x^{k+1}-x^*| \leqslant \alpha\, |x^k-x^*|^2$

Quadratic convergence

$$\| x^{(k+1)} - x^* \| < \alpha \| x^{(k)} - x^* \|^2 \quad \text{for all } k \geqslant N$$

Linear convergence

$$\| x^{(k+1)} - x^* \| \leqslant \gamma \| x^{(k)} - x^* \| \qquad \gamma < 1 .$$

Newton's method has quadratic convergence

$$0 = f(x^*) = f(x^k) + \frac{\partial f}{\partial x}\Big|_{x^k} (x^* - x^k) + \frac{1}{2} \frac{\partial^2 f}{\partial x^2}\Big|_{\hat{x}} (x^* - x^k)^2 \qquad ①$$

Newton iteration is given by

$$0 = f(x^k) + \frac{\partial f}{\partial x}\Big|_{x^k} (x^{k+1} - x^k) \qquad ②$$

Subtract ① from ②

$$0 = \frac{\partial f}{\partial x}\Big|_{x^k} (x^{k+1} - x^*) - \frac{1}{2} \frac{\partial^2 f}{\partial x^2}\Big|_{\hat{x}} (x^* - x^k)^2$$

$$\Rightarrow (x^{k+1} - x^*) = \left(\frac{\partial f}{\partial x}\Big|_{x^k}\right)^{-1} \frac{1}{2} \frac{\partial^2 f}{\partial x^2}\Big|_{\hat{x}} (x^* - x^k)^2$$

$$= \left(\frac{\partial f}{\partial x}\Big|_{x^k}\right)^{-1} \frac{1}{2} \frac{\partial^2 f}{\partial x^2}\Big|_{\hat{x}} (x^k - x^*)^2$$

Suppose $\left(\frac{\partial f}{\partial x}\Big|_{x^k}\right)^{-1} \left(\frac{\partial^2 f(\hat{x})}{\partial x^2}\right)$ is bounded by α independent of x, k

Then $\| x^{k+1} - x^* \| \leqslant \alpha \| x^k - x^* \|^2$

What does the condition $\left\| \frac{\partial f}{\partial x}(x^k)^{-1} \frac{\partial^2 f}{\partial x^2}(\hat{x}) \right\| < \alpha$ mean?

- $\frac{\partial^2 f}{\partial x^2}$ is bounded
- $\frac{\partial f}{\partial x}$ is bounded away from zero.

Examples:

1) $f(x) = x^2 - 1 = 0 ; \qquad (x^* = 1)$

$\frac{\partial f}{\partial x} = 2x \quad ; \quad \frac{\partial^2 f}{\partial x^2} = 2$

$\therefore$ Newton's Method $\Rightarrow$

$$x^{k+1} = x^k - \frac{[(x^k)^2 - 1]}{2x^k}$$

$$= x^k - \frac{1}{2} x^k + \frac{1}{2x^k}$$

$$= \frac{1}{2} x^k + \frac{1}{2x^k}$$

This example has quadratic convergence because

$\frac{\partial^2 f}{\partial x^2} = 2$ is bounded

$\frac{\partial f}{\partial x} = 2x$ is bounded.

$$\| x^{k+1} - 1 \| \leqslant \| x^k - 1 \|^2 / 2|x^k|$$

2) $f(x) = x^2 = 0 \qquad (x^* = 0)$

$\frac{\partial f}{\partial x} = 2x \quad ; \quad \frac{\partial^2 f}{\partial x^2} = 2$

Newton's Method: $x^{k+1} = x^k - \frac{(x^k)^2}{2x^k} = \frac{1}{2} x^k$

This method has linear convergence

$$x^{k+1} - 0 = \frac{1}{2} x^k - 0 = +\frac{1}{2} x^k - \frac{1}{2} 0$$

$$x^{k+1} - x^* = +\frac{1}{2} x^k - \frac{1}{2} x^* = \frac{1}{2}(x^k - x^*)$$

$$\therefore \| x^{k+1} - x^* \| \leqslant \frac{1}{2} \| x^k - x^* \|$$

This is so because $\frac{\partial f}{\partial x} = 0$ at $x = 0$

$\Rightarrow \left(\frac{\partial f}{\partial x}\right)^{-1}$ is not bounded at the solution

Note If $|\alpha(x^0-x^*)| \le r < 1$

then the sequence x^k converges to x^*

This can be easily shown by considering:

$$\begin{aligned} |x^1-x^*| &\le \alpha|x^0-x^*|^2 \\ &\le \alpha|x^0-x^*|\,|x^0-x^*| \\ &\le r\,|x^0-x^*| \end{aligned}$$

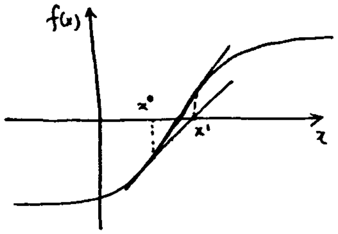
Now $$\begin{aligned} |x^2-x^*| &\le \alpha|x^1-x^*|^2 = \alpha|x^1-x^*|\,|x^1-x^*| \\ &\le \alpha r\,|x^0-x^*|\,|x^1-x^*| \\ &\le r^2\,|x^1-x^*| \\ &\le r^3\,|x^0-x^*| \end{aligned}$$

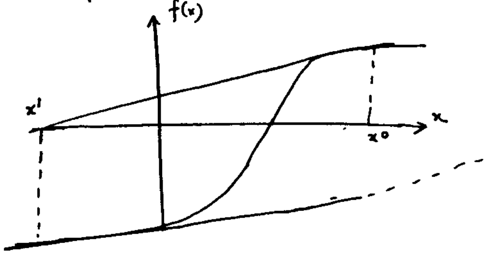
$\vdots$

Thus if α is bounded (i.e. $\frac{\partial f}{\partial x}$ is bounded away from zero, and $\frac{\partial^2 f}{\partial x^2}$ is bounded)

then Newton's method will converge given an initial guess that is sufficiently close to the solution.

What do we mean by sufficiently close?



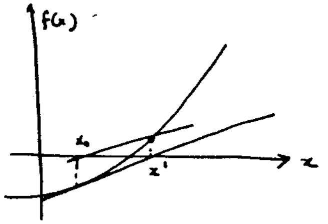


Potential Problems

1) x^0 is not close enough to x^*

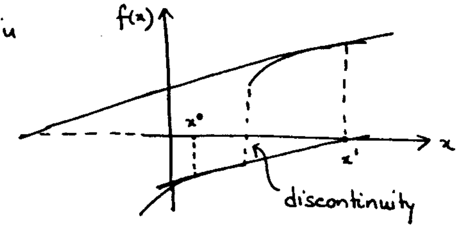
2) Error in derivative

In this particular case one gets oscillations!



3) Discontinuity in function

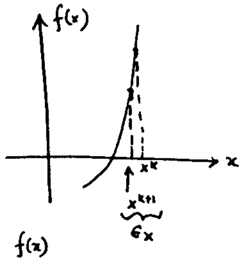
Because of the discontinuity there will be oscillations.



Convergence Checks

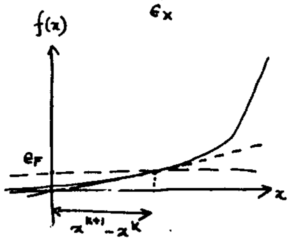
1) $\|x_{k+1}-x_k\| < \epsilon_x$

However, this is not sufficient as in this example where $f(x)$ is not close to zero



2) $\|f(x^{k+1})\| < \epsilon_F$

This in itself is also a problem because $|x^{k+1}-x^k|$ is large



So for convergence use both criterions

Multidimensional Case

In general $f(x)$ and x are n-vectors

$$f : \mathbb{R}^n \to \mathbb{R}^n$$

$$f(x^*) = f(x^\circ) + \frac{\partial f}{\partial x}(x^\circ)\,(x^* - x^\circ) + \text{higher-order terms}$$

As before solve for x from

$$f(x^\circ) + \frac{\partial f}{\partial x}(x_0)\,(x - x^\circ) = 0$$

The iterative procedure is then

$$\left.\frac{\partial f}{\partial x}\right|_{x^k} (x^{k+1} - x^k) = -f(x^k)$$

or $\left.\frac{\partial f}{\partial x}\right|_{x^k} x^{k+1} = -f(x^k) + \left.\frac{\partial f}{\partial x}\right|_{x^k} x^k$

This is a system of linear equations $Ax = b$ where

$$A = \left.\frac{\partial f}{\partial x}\right|_{x^k} \; ; \quad b = -f(x^k) + \left.\frac{\partial f}{\partial x}\right|_{x^k} x^k$$

$\left.\frac{\partial f}{\partial x}\right|_{x^k}$ is called the Jacobian of f evaluated at x^k. Also denoted by $J(x^k)$.

<u>Note</u> $J(x) = \frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & & & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$

where $f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix}$ and $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

Quadratic convergence of Newton's method (Newton-Raphson)

$$x^{k+1} = x^k - J^{-1}(x^k)\, f(x^k).$$

<u>Suppose</u>

1) $\|J^{-1}(x)\| \le \beta$ bounded inverse

2) $J(x)$ is Lipschitz continuous
$$\|J(x) - J(y)\| \le L\,\|x - y\| \quad \forall\, x, y$$

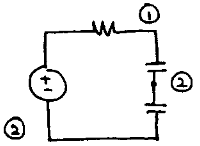
<u>Then</u>

- $x^{(k)}$ converges to x^* (the solution) provided $x^{(0)}$ is sufficiently close to x^*
- $\|x^{(k+1)} - x^*\| \le \alpha\, \|x^{(k)} - x^*\|^2$
  i.e. quadratic convergence

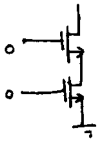
What does this all mean?

1) Device model equations must be continuous with continuous derivatives. Derivative calculations must be accurate
2) Provide a good initial guess $x^{(0)}$.
3) There should be no floating nodes.
   A floating node ⇒ singular Jacobian.

- Node ② is a floating node.
  SPICE will issue an error:
  No DC path to ground from Node ②



- Two off MOSFETs in series result in "floating" nodes



Computational Procedure

1) Evaluate $f(x^{(k)})$ and $J(x^{(k)})$

2) Assemble circuit equations
$$J(x^k)\,(x^{k+1}-x^k) = -f(x^k)$$
or
$$J(x^k)\,x^{k+1} = -f(x^k) + J(x^k)\,x^k$$

3) Do not compute the inverse of J, instead solve linear equations with sparse-matrix techniques.

4) Solution gives x^{k+1} or $\Delta x^{k+1} \Rightarrow x^{k+1} = x^k + \Delta x^{k+1}$
Repeat procedure until convergence or Maximum Iterations

Remarks

- Since the device equations are known, the derivatives can be computed explicitly
- Automatic differentiation is required for computing accurate derivatives for complex models
- The significant costs are in calculating
  $f(x^k)$, $J(x^k)$ i.e. model evaluation
  Solve $Ax = b$ i.e. linear eqn. solution
- Depending on the cost of model evaluation one of these becomes the dominant cost. For very complex models, model evaluation dominates
- To improve performance the model evaluation is done once every few iterations
  $\Rightarrow$ savings in computing $J(x)$, $f(x)$
  This is known as Bypass in SPICE.
- If only $J(x)$ calculation is skipped, then also there are significant savings.

Newton-Raphson applied to linear equations

$$f(x) = x_1 + x_2 + x_3 = 0$$

Linearize per NR method:
$$\Rightarrow \left.\frac{\partial f}{\partial x}\right|_{x^k}(x^{k+1}-x^k) = -f(x^k)$$

$$\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial x_2} = \frac{\partial f}{\partial x_3} = 1$$

$$\Rightarrow 1(x_1^{k+1}-x_1^k) + 1(x_2^{k+1}-x_2^k) + 1(x_3^{k+1}-x_3^k) = -(x_1^k+x_2^k+x_3^k$$

$$\Rightarrow x_1^{k+1} + x_2^{k+1} + x_3^{k+1} = 0$$

i.e. Linearization of linear equations results in the same linear equations
$\Rightarrow$ no change to linear elements for stamping

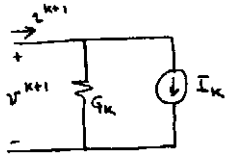
What about the nonlinear elements?

Consider a nonlinear resistor $i = i(v)$

Linearize: $$i^{k+1} = \underbrace{\left.\frac{\partial i}{\partial v}\right|_{v^k}}_{G_k}(v^{k+1}-v^k) + i(v^k)$$

$$\text{or } i^{k+1} = G_k v^{k+1} + \underbrace{i(v^k) - G_k v^k}_{I_k}$$

$$\therefore i^{k+1} = G_k v^{k+1} + I_k \quad \Rightarrow$$

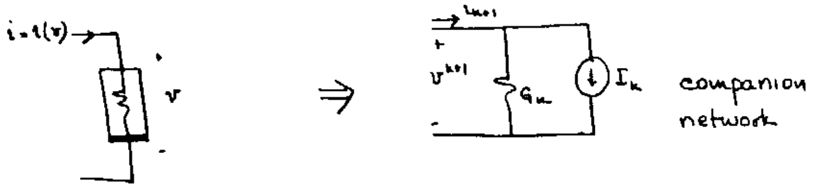


where $G_k = \left.\dfrac{\partial i}{\partial v}\right|_{v^k}$

$I_k = i(v^k) - G_k v^k$

The equivalent circuit representation is known as the companion network.

Once we have the companion network "stamping" is easy.



Stamp

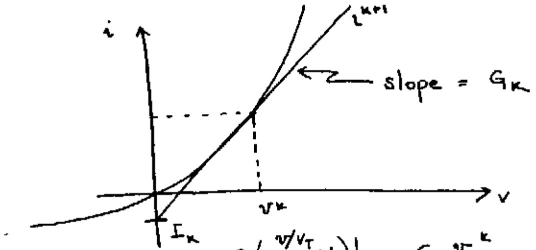
$$\begin{array}{c|cc} & v+ & v- \\ \hline + & +G_K & -G_K \\ - & -G_K & +G_K \end{array} \qquad \begin{array}{c} \text{RHS} \\ \begin{bmatrix} -I_K \\ +I_K \end{bmatrix} \end{array}$$

For a diode

$$i = I_s(e^{v/V_T} - 1)$$

$$\therefore \left.\frac{\partial i}{\partial v}\right|_{v^k} = \left.\frac{I_s}{V_T} e^{v/V_T}\right|_{v^k} = G_K$$

and $I_K = I_s(e^{v/V_T} - 1)\Big|_{v^k} - G_K v^k$

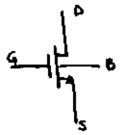


$$\simeq I_s(e^{v/V_T} - 1)\Big|_{v^k} - G_K v^k$$

MOSFET

Let us consider the SPICE Level 1 model

In saturation



$$I_{DS} = \frac{K'}{2}\frac{W}{L}(V_{GS}-V_T)^2(1+\lambda V_{DS})$$

i.e. $I_{DS} = f(V_{GS}, V_{DS})$ assuming V_T is independent of V_{BS} (for simplicity only).

Note: The source node has been chosen as the reference

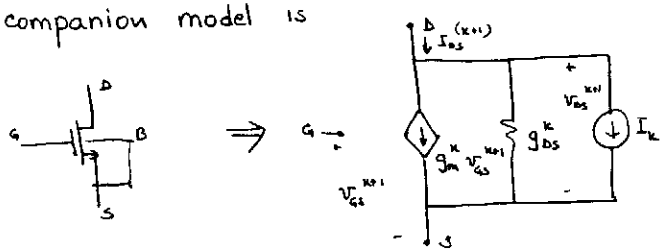
For NR we linearize

$$\Rightarrow I_{DS}^{K+1} = I_{DS}^{K} + \left.\frac{\partial I_{DS}}{\partial V_{GS}}\right|_{V_{GS}^K, V_{DS}^K}(V_{GS}^{K+1} - V_{GS}^{K})$$
$$+ \left.\frac{\partial I_{DS}}{\partial V_{DS}}\right|_{V_{GS}^K, V_{DS}^K}(V_{DS}^{K+1} - V_{DS}^{K})$$

where $g_m^K = \left.\frac{\partial I_{DS}}{\partial V_{GS}}\right|_{V_{GS}^K, V_{DS}^K} = \frac{K'W}{L}(V_{GS}^K - V_T)(1+\lambda V_{DS}^K)$

$$g_{DS}^K = \left.\frac{\partial I_{DS}}{\partial V_{DS}}\right|_{V_{GS}^K, V_{DS}^K} = \frac{K'}{2}\frac{W}{L}(V_{GS}^K - V_T)^2\lambda$$

The companion model is

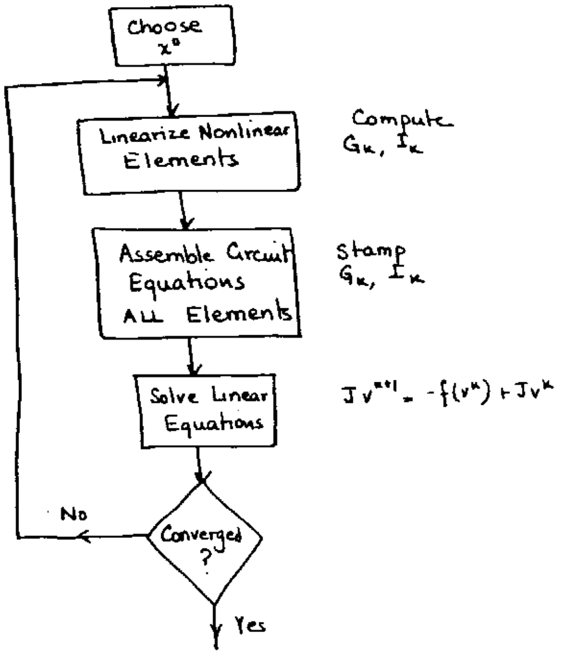


where $I_K = I_{DS}^K - g_m^K V_{GS}^K - g_{DS}^K V_{DS}^K$

Based on the companion model we have the stamp

$$\begin{array}{c|ccc|c} & V_D & V_S & V_G & \text{RHS} \\ D & g_{DS}^k & -g_{DS}^k - g_m^k & +g_m^k & -I_k \\ S & -g_{DS}^k & +g_{DS}^k + g_m^k & -g_m^k & +I_k \\ G & 0 & 0 & 0 & 0 \end{array}$$

SPICE DC Analysis


Choose x^0
Linearize Nonlinear Elements
Compute G_k, I_k
Assemble Circuit Equations ALL Elements
Stamp G_k, I_k
Solve Linear Equations
$J v^{k+1} = -f(v^k) + J v^k$
Converged ?
No
Yes

Convergence Check

For each node

$$|v^{k+1} - v^k| \leq \underset{\substack{\uparrow \\ \text{VNTOL}}}{\epsilon_A} + \underset{\substack{\uparrow \\ \text{RELTOL}}}{\epsilon_R} \text{ MAX}(|v^{k+1}|, |v^k|)$$

SPICE has no convergence check on $f(x)$. Instead it does the following test

$$|I_k - \hat{I}_k| \leq \underset{\substack{\uparrow \\ \text{ABSTOL}}}{\epsilon_A} + \epsilon_R \text{ MAX}(|I_k|, |\hat{I}_k|)$$

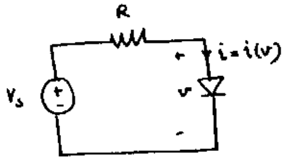
where $\hat{I}_k$ is the linearized terminal current for v^k;
and I_k is the actual current for v^k.

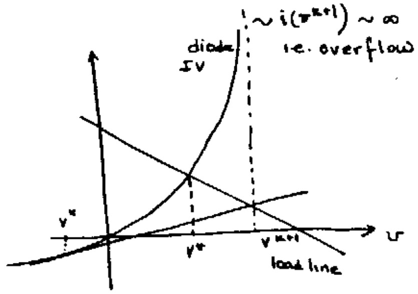
Remark

The default tolerances in SPICE are
VNTOL = 1 μV, ABSTOL = 1 pA, RELTOL = 10^{-3}

More Problems with NR-Method

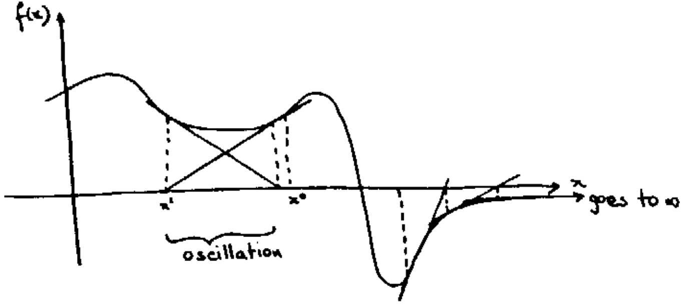
1) Numerical Overflow :





There are various ways to fix this problem
1) limit the voltage step for model evaluation
2) modify device characteristics such that there is no overflow
   e.g. linearize exponential around some $\hat{v}$ which is large but small enough to avoid overflow
3) Use a damped - Newton method
   $v^{k+1} = v^k + \lambda \Delta v^{k+1} \qquad 0 < \lambda \leq 1$

2) Nonconvergence



First let us look at limiting considering the diode example where the problem is the exponential nonlinearity

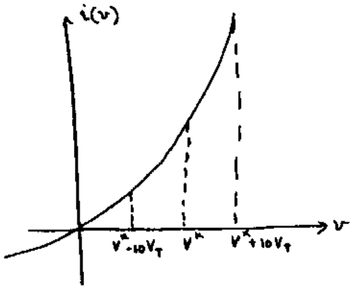
Example 1

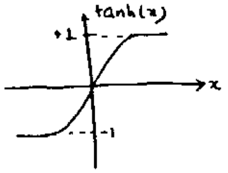
Ensure that

$$V^k - 10V_T \leqslant V_{lim}^{k+1} \leqslant V^k + 10V_T$$

Use a tah limitng

i.e. $V_{lim}^{k+1} = V^k + 10V_T \tanh\left(\frac{V^{k+1} - V^k}{10V_T}\right)$





For x large $\tanh(x) \cong \pm 1.0$

$\therefore V_{lim}^{k+1} = V^k + 10V_T$ or $V^k - 10V_T$

For x small $\tanh(x) \cong x$

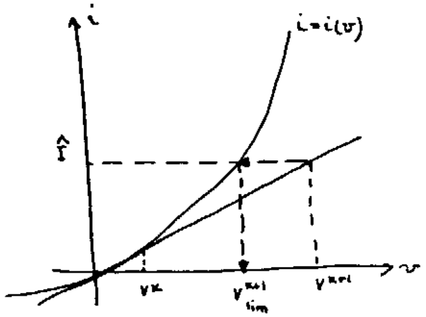
$\therefore V_{lim}^{k+1} \cong V^k + 10V_T \left(\frac{V^{k+1} - V^k}{10V_T}\right) = V^{k+1}$ i.e. no limitng performed

SPICE PN Junction Limitng (PNJLIM)

For the diode

$$i = I_s\left(e^{V/V_T} - 1\right)$$

$$\frac{di}{dv} = \frac{I_s}{V_T} e^{V/V_T}$$



The linearization at iterathou k+1 gives

$$i^{k+1} = I_s\left(e^{v^k/V_T} - 1\right) + \frac{I_s}{V_T}\left(e^{v^k/V_T}\right)\left(v^{k+1} - v^k\right) = \hat{I}$$

Now use $\hat{I}$ to calculate V_{lim}^{k+1} as

$$\hat{I} = I_s\left(e^{v_{lim}^{k+1}/V_T} - 1\right)$$

$$\Rightarrow \quad v_{lim}^{k+1} = V_T \ln\left(\frac{\hat{I}}{I_s} + 1\right)$$

$$= V_T \ln\left[\frac{I_s}{V_T} e^{v^k/V_T}\left(v^{k+1} - v^k\right) + I_s e^{v^k/V_T}\right]$$

$$= v^k + V_T \ln\left[\frac{v^{k+1} - v^k}{V_T} + 1\right]$$

This limitng formula is applied only when $v^{k+1} > V_{crit}$ and $v^{k+1} > v^k$

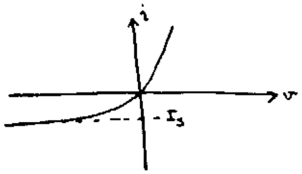
where $V_{crit} = V_T \ln\left(V_T / \sqrt{2} I_s\right)$

Remark

V_{crit} corresponds to the point of maximum curvature

What happens if v^k are negative ?

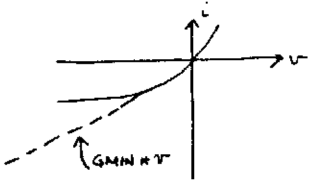
$$\frac{di}{dv} = \frac{I_s}{V_T} e^{v/V_T} \simeq 0$$



This could also be a problem. So a limiting is used for negative voltages as well.

In addition, a resistor called GMIN is added in parallel to the device. This helps in linear equation solution.

So we use a modified set of equations for the diode



$$i = I_s\left(e^{v/V_T} - 1\right) + GMIN * v$$

$$\frac{di}{dv} = \frac{I_s}{V_T} e^{v/V_T} + GMIN$$

SPICE2 also uses another check

$$\text{if } v < -5.0 * V_T \text{ then } \quad i = -I_s + GMIN * v$$
$$\frac{di}{dv} = -\frac{I_s}{v} + GMIN$$

We can see some problems here

- discontinuity in i & $\frac{di}{dv}$

SPICE3 uses a linearized extension at $-3 * V_T$. So the function is modified but continuous.

Remark.

- We would like to limit even if overflow is not a problem.
- A smaller Δv ensures better convergence

SPICE: FETLIM (limit VGS), LIMVDS (limit VDS)

Step-Limited Approach

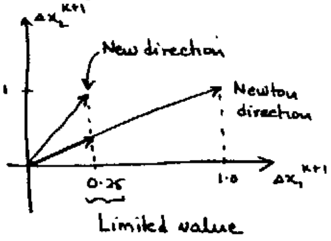
Newton's method: $J \Delta x^{k+1} = -f(x^k)$

$$x^{k+1} = x^k + \underbrace{\Delta x^{k+1}}_{\text{use a limited value here}}$$

i.e. $\Delta\tilde{x}^{k+1}$

One choice $\quad \Delta\tilde{x}_i^{k+1} = \begin{cases} \Delta x_i^{k+1} & \text{if } |\Delta x_i^{k+1}| < 0.25 \\ 0.25 \text{ sign}(\Delta x_i^{k+1}) & \text{otherwise} \end{cases}$

This approach corrupts the Newton direction



How can we ensure that the Newton direction is maintained ?

Take $\Delta\tilde{x}^{k+1} = \alpha \Delta x^{k+1}$

where $\alpha = \dfrac{0.25}{\max_i |\Delta x_i^{k+1}|}$

i.e. each element is scaled by the same amount. Here the iteration proceeds in the Newton direction, although with a smaller step size.

Damped Newton Schemes

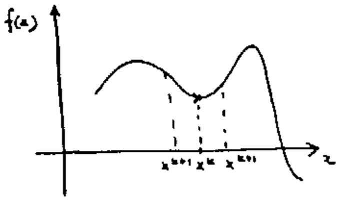
$$J \underbrace{\Delta x^{k+1}}_{\text{Newton direction}} = -f(x^k)$$

$$x^{k+1} = x^k + \lambda^k \Delta x^{k+1}$$

where λ^k is unity close to the solution so that a standard Newton iteration is performed.

How does one pick λ^k ?

Take λ^k such that $\| f(x^k)\|_2^2$ is reduced at each iteration. The idea being that $\| f(x^k)\|_2$ reduction implies closeness to the solution. Is this always going to work? No



Here $\|f(x^{k+1})\|$ will increase regardless of the value of λ^k !

The problem is that the derivative is zero at x^k i.e. there is a local minima

When can we be sure that the method will converge?

If

$$\|J^{-1}(x)\| \le \beta$$

$$\| J(x) - J(y)\| \le L \|x-y\| \qquad \forall\ x, y$$

Then there exists a set of γ^k (not all equal) such that

$$\| f(x^{k+1})\| \le \alpha \| f(x^k)\| \qquad \alpha < 1$$

One simple algorithm to pick γ^k is the following procedure

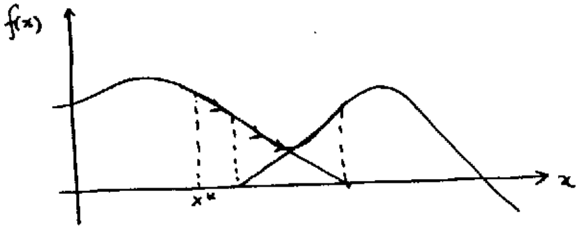
for $j = 0$ until $\| f(x^k + \gamma_j^k \Delta x^{k+1})\| < \| f(x^k)\|$

$$\gamma_j^k = \frac{1}{2^j} \qquad (1, 1/2, 1/4, \text{----})$$

Ofcourse one may not find a γ_j^k that satisfies the above condition i.e. a local minima. Then out of luck

Remarks

- if J^{-1} is bounded, damping will always work and the method becomes a regular Newton method near the solution. (ADVANTAGE)
- if J is singular somewhere then damped Newton will push you there (DISADVANTAGE)



When at a local minima there is no way of getting out of it when damping is used. With a regular Newton there could be some hope!

How can we improve convergence?

- Improve models i.e. model and derivative continuity, bounded functions. This can be a cause for lot of problems
- Provide better initial guesses. This is a problem for analog circuits. Digital circuits can be initialized by logic simulation ...
- Improve algorithms, i.e. have something that works better than the Newton method.

New algorithms

- Optimization based
- Continuation - source stepping, pseudotransient
  - Homotopy

Optimization-based methods

Solve $f(x) = 0$ by solving $\min \| f(x) \|_2^2$

If $f(x)$ has a zero then $\min \| f(x) \|_2^2 = 0 \Rightarrow f(x) = 0$

No matter where one starts from, a downhill direction will lead us to a minima. To move downhill be need information of the derivative of the function

$$\text{Suppose} \quad F(x) = \| f(x) \|_2^2 = \sum_{i=1}^{n} f_i^2(x)$$

then the gradient of the function $\frac{\partial F}{\partial x}$ (or ∇F) points in the direction of increasing function values

$\Rightarrow -\nabla F$ is the downhill direction

Steepest descent $\Rightarrow$ proceed along the negative of the gradient to achieve fastest decrease in the function

$$\frac{\partial F}{\partial x} = \sum 2 f_i(x) \frac{\partial f_i}{\partial x} = 2 \sum f_i(x) \begin{bmatrix} \frac{\partial f_i}{\partial x_1} \\ \frac{\partial f_i}{\partial x_2} \\ \vdots \\ \frac{\partial f_i}{\partial x_n} \end{bmatrix} = 2 J^T f$$

The updak rule is:

$$x^{k+1} = x^k + \alpha \left(-2 J^T f \right)$$

Note: Newton's method $\Rightarrow$ $x^{k+1} = x^k + \lambda \left(-J^{-1} f \right)$

Remark.

- A steepest descent method can lead to a local minima if the function is not convex.

Continuation Methods

Source Stepping

Solution is known when the sources are at 0. $x = 0$. Ramp up the source voltage by a small amount, say δ_1. Then $x(\delta_1)$ is close to $x(0)$, and $x(0)$ is used as an initial guess for Newton iteration (Recall "close enough" for Newton convergence). Step to δ_2 where δ_2 is close to δ_1 and use $x(\delta_1)$ as an initial guess. Repeat until $\delta_n = 1$.

Procedure

source vector S. Take a parameter λ, $0 \leq \lambda \leq 1$

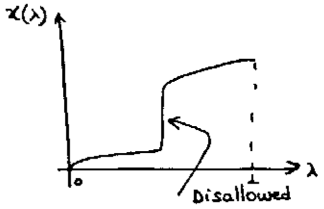
Define $S(\lambda) = \lambda S$.

Vary λ from 0 to 1 and solve at each λ.

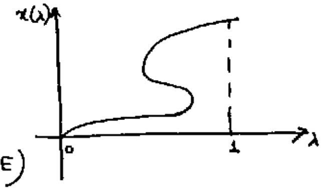
Obtain circuit solution when $\lambda = 1$

Assumption

$x(\lambda)$ i.e. solution for source $= \lambda S$ is sufficiently smooth (continuous in λ), hence the name continuation method.



If this assumption is satisfied then one can choose λ small enough and get a solution, except in the case shown.



Other schemes

GMIN stepping. (SPICE3, HSPICE)

Let us look at the general case.

Solve $\quad \tilde{f}(x(\lambda), \lambda) = 0$

Where

1) $\tilde{f}(x(0), 0) = 0$ is easy to solve

2) $\tilde{f}(x(1), 1) = f(x)$

3) $x(\lambda)$ is sufficiently smooth. This may be hard to ensure!

Simple scheme

$$\tilde{f}(x(\lambda), \lambda) = \lambda f(x(\lambda)) + (1-\lambda)\, x(\lambda)$$

For $\lambda = 0$; $\quad \tilde{f}(x(0), 0) = x(0)$

$\lambda = 1$; $\quad \tilde{f}(x(1), 1) = f(x(1))$

For $\lambda = 0$, the problem being solved is

$$\tilde{f}(x(0), 0) = x(0) = 0 \quad \text{, simple to solve}$$

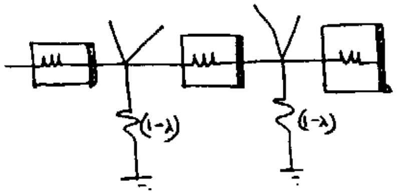
For any λ: $\quad \tilde{f}(x(\lambda), \lambda) = \lambda f(x(\lambda)) + (1-\lambda)\, x(\lambda) = 0$

If one applies Newton's method then

$$\frac{\partial \tilde{f}}{\partial x} = \lambda \frac{\partial f}{\partial x} + (1-\lambda) I$$

so the Jacobian is modified by λ and a $(1-\lambda)$ diagonal term is added.

Circuit interpretation



Why is this any better? – we have an improvement

Consider $\tilde{f}(x(\lambda+\cancelto{0}{\delta\lambda), \lambda}+\delta\lambda) \cong \tilde{f}(\cancelto{0}{x(\lambda)}, \lambda) + \frac{\partial \tilde{f}}{\partial x}(x(\lambda), \lambda)\left(x(\lambda+\delta\lambda) - x(\lambda)\right)$

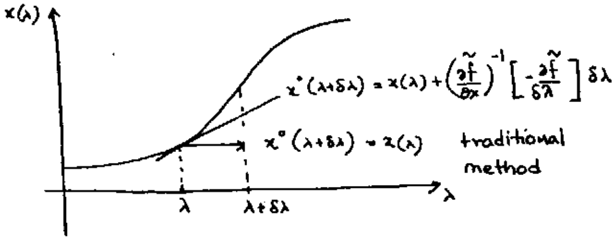
$$+ \frac{\partial \tilde{f}}{\partial \lambda}(x(\lambda), \lambda)\, \delta\lambda$$

$$\therefore \quad \underbrace{\frac{\partial \tilde{f}}{\partial x}(x(\lambda), \lambda)}_{\text{This is available from Newton's method}} \delta x = -\underbrace{\frac{\partial \tilde{f}}{\partial \lambda}\ \delta\lambda}_{\text{What is this}}$$

For $\quad \tilde{f}(x(\lambda), \lambda) - \lambda f(x) + (1-\lambda)\, x$

$$\frac{\partial \tilde{f}}{\partial \lambda} = f(x) + (0-1)\, x = f(x) - x$$

and this can be easily determined.



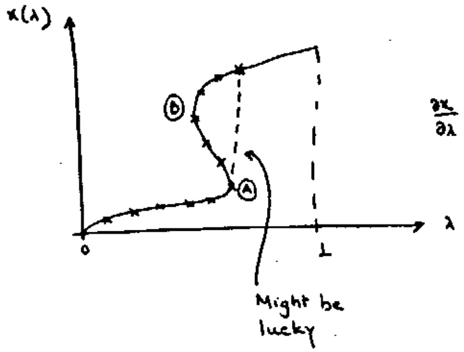
Newton-Continuation Procedure

```
Solve  $\tilde{f}(x(0),0) = 0$
    $x(\lambda_{prev}) = x(0)$
    $\delta\lambda = 0.1$
    $\lambda = \delta\lambda$

While  $\lambda < 1$  {
    $x^0(\lambda) = x(\lambda_{prev}) + \frac{\partial\tilde{f}}{\partial x}^{-1}(x(\lambda_{prev}),\lambda_{prev})\left(-\frac{\partial\tilde{f}}{\partial\lambda}\delta\lambda\right)$

        Solve the nonlinear equation
            $\tilde{f}(x(\lambda),\lambda) = 0$

        i.e.  $\frac{\partial\tilde{f}}{\partial x}(x^k(\lambda),\lambda)\left(x^{k+1}(\lambda) - x^k(\lambda)\right) = -\tilde{f}(x^k(\lambda))$

            until converged
        If converged {
            $x(\lambda_{prev}) = x^{k+1}(\lambda)$
            $\lambda = \lambda + \delta\lambda$
            $\delta\lambda = 2\delta\lambda$            /* use a larger $\delta\lambda$ */
        } else {
            $\delta\lambda = \frac{\delta\lambda}{2}$            /* reduce $\delta\lambda$ and
                                          retry */
            $\lambda = \lambda_{prev} + \delta\lambda$
        }
}
```

This method would work except when $\frac{\partial x}{\partial\lambda} \to \infty$

$$\text{i.e. } \frac{\partial x}{\partial\lambda} = \frac{\partial\tilde{f}}{\partial x}^{-1} \cdot \frac{\partial\tilde{f}}{\partial\lambda} \longrightarrow \infty$$

i.e. $\frac{\partial\tilde{f}}{\partial x}$ is singular

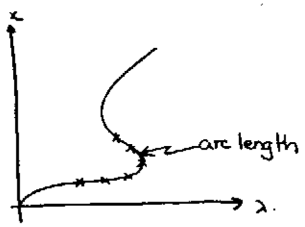


$\frac{\partial x}{\partial\lambda} \to \infty$ at A & B

The only way to progress here is by decreasing λ.

How can we make λ decrease when such turning points are encountered?

We parameterize the curve! i.e. try to move along the curve by a certain amount at each step. The arc-length will always increase



$$\text{arc length} \cong \delta\sigma = \sqrt{(\delta x)^2+(\delta\lambda)^2}$$

Now x & λ are parameterized in terms of the approx arc length. σ

i.e. $\begin{bmatrix} x(\sigma) \\ \lambda(\sigma) \end{bmatrix}$ $\quad 0 \le \sigma \le \sigma_L$

↑total length of the curve.

So we now have a modified system to solve

$$\tilde{f}(x(\sigma), \lambda(\sigma)) = 0$$
$$\|\delta x\|^2 + |\delta\lambda|^2 = \delta\sigma^2$$

i.e. we have an extra equation which is given by

$$\sum \left(x(\lambda) - x(\lambda_{prev})\right)^2 + (\lambda - \lambda_{prev})^2 = \delta\sigma^2$$

Solve this system of equations by Newton's method. The unknowns are x & λ.

$$\begin{bmatrix} \frac{\partial \tilde{f}}{\partial x} & \frac{\partial \tilde{f}}{\partial \lambda} \\ 2(x - x_{prev})^T & 2(\lambda - \lambda_{prev}) \end{bmatrix} \begin{bmatrix} x^{k+1}(\lambda) - x^k(\lambda) \\ \lambda^{k+1} - \lambda^k \end{bmatrix}$$

$$= - \begin{bmatrix} \tilde{f}(x^k(\lambda), \lambda^k) \\ (\lambda^k - \lambda_{prev})^2 + \sum \left(x(\lambda^k) - x_{prev}\right)^2 - \delta\sigma^2 \end{bmatrix}$$

This procedure works in terms of starting out with $\sigma = 0$ and then incrementing σ as

$$\sigma = \sigma_{old} + \delta\sigma$$

where $\underline{\delta\sigma > 0}$ is the increment in arc-length.

Remark

If $\frac{\partial \tilde{f}}{\partial x}$ is singular, the new matrix is not

e.g.

$$\left[\begin{array}{cc|c} 1 & 0 & x \\ 0 & 0 & x \\ x & x & x \end{array}\right]$$

augmented $\quad \nwarrow \frac{\partial \tilde{f}}{\partial \lambda}$

The above scheme for solving the nonlinear equations is called a <u>homotopy</u> technique

Remarks

- The strict formulation is based on the actual arc-length s.

  If x & λ are parameterized by s then the arc-length condition is

$$\|\dot{x}(s)\|_2^2 + |\dot{\lambda}(s)|^2 = 1$$

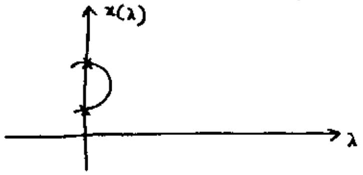
- HOMPACK is a package that solves the above equation as an ordinary differential equation.

Issues

- choice of embedding function

  i.e. $\tilde{f}(x(\lambda), \lambda)$

  Convergence depends on a proper choice of $\tilde{f}$
- $\tilde{f}(x(0), 0)$ must have a unique solution



This avoids the situation shown above

- The solution $x(\lambda)$ must be bounded. This ensures that the track doesn't go to infinity

Remarks

- several nonlinear systems have to be solved for different λ

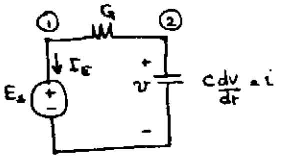
  i.e. $\tilde{f}(x(\lambda), \lambda) = 0$ instead of only $f(x) = 0$
- Only a few steps of Newton's method are needed for each λ
- Convergence problems are solved!

Small-signal AC analysis

- Frequency-domain analysis of linear systems
- provides information on magnitude and phase response.
  - ⇒ phase margin, stability of filters, amplifers

Let us look at a linear circuit

Let us write the MNA equations for this circuit



$$G(V_1 - V_2) + I_E = 0$$
$$G(V_2 - V_1) + C\frac{dV_2}{dt} = 0$$
$$V_1 = E_s$$

Assume the response is of the form $x = x_0 + x_{ac}e^{j\omega t}$ then we obtain the following system of equations

$$G(V_{1ac} - V_{2ac}) + I_{Eac} = 0$$
$$G(V_{2ac} - V_{1ac}) + j\omega C V_{2ac} = 0$$
$$V_{1ac} = E_{sac}$$

or in matrix form
$$\begin{bmatrix} G & -G & 1 \\ -G & G+j\omega C & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_{1ac} \\ V_{2ac} \\ I_{Eac} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ E_{sac} \end{bmatrix}$$

Note AC stamp for capacitor is

$$\begin{bmatrix} j\omega C & -j\omega C \\ -j\omega C & j\omega C \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

We now have a system of complex linear equations. Sparse-matrix solvers such as Sparse1.3 solve a sparse linear complex system so these equations can be solved.

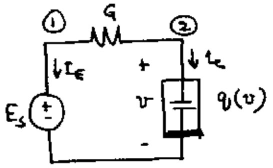
Procedure

```
for  $\omega = \omega_{init}$ to $\omega_{final}$ {
    assemble circuit equations for each $\omega$
    (Note there is a frequency dependent
     part and a frequency independent part)

    Solve  $A(j\omega)\, x = b$
}
```

⇒ x is a function of frequency.

Now consider a nonlinear capacitor

i.e. $q = q(v)$
$i_c = \frac{dq}{dt}$



The equations are

$$G(V_1 - V_2) + I_E = 0$$
$$G(V_2 - V_1) + \frac{dq}{dt} = 0$$
$$V_1 = E_s$$

Linearize $q(v(t))$ around the dc operating point

$$\Rightarrow q = q(v_0) + \underbrace{\left.\frac{\partial q}{\partial v}\right|_{v_0}}_{C(v_0)} (v - v_0)$$

Again take $x = x_0 + x_{ac}e^{j\omega t}$

Then obtain the following equations:

$$\begin{bmatrix} G & -G & 1 \\ -G & G+j\omega C(v_0) & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_{1ac} \\ V_{2ac} \\ I_{Eac} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ E_{ac} \end{bmatrix}$$

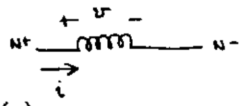
Note the stamp for the nonlinear capacitor is

$$\begin{bmatrix} j\omega C(v_0) & -j\omega C(v_0) \\ -j\omega C(v_0) & j\omega C(v_0) \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

where $C(v_0) = \left.\frac{\partial q}{\partial v}\right|_{v_0}$ i.e. the incremental capacitance evaluated at the dc operating point.

What about a Linear Inductor?

$$v = L\frac{di}{dt}$$



Since this is not of the form $i = i(v)$ we will use an MNA description, whereby I_L is an unknown.

For small-signal ac analysis $v = j\omega L i$

Thus, the stamp is

$$\begin{array}{c|c} & \begin{array}{ccc} V_{N+} & V_{N-} & I_L \end{array} \\ \begin{array}{c} N+ \\ N- \\ \text{branch} \end{array} & \begin{bmatrix} & & 1 \\ & & -1 \\ 1 & -1 & -j\omega L \end{bmatrix} \end{array} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

For dc, $\omega = 0$

$$\begin{bmatrix} & & 1 \\ & & -1 \\ 1 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Remarks

- For ac analysis $V_{N+} - V_{N-} - j\omega L I_L = 0$ can be expressed as $I_L = \frac{V_{N+} - V_{N-}}{j\omega L}$

  So, a NA formulation can be written. The stamp is

$$\begin{bmatrix} -j/\omega L & j/\omega L \\ +j/\omega L & -j/\omega L \end{bmatrix}$$

- This stamp cannot be used for "DC" analysis i.e. analysis at $\omega = 0$. The MNA formulation degrades nicely.
- With an NA formulation say $\omega = 1$, $L = 1nH$

  $\Rightarrow \frac{1}{\omega L} = 10^9$

  For a C of 1pF $\omega C = 10^{-12}$ i.e. the dynamic range of entries is huge $[10^{-12}, 10^9]$. This can result in numerical problems.
- For MNA formulation $\omega L = 10^{-9}$, $\omega c = 10^{-12}$ so the dynamic range of matrix elements is $[10^{-12}, 1]$ which is much better than for NA.

Summary

- For small-signal ac analysis non-linear capacitors replaced by incremental capacitances at the dc operating point
- nonlinear conductance elements $i = i(v)$ are replaced by the linearized conductances
- An MNA formulation used for inductors
- Sparse complex equations solved by a sparse matrix package

  $\Rightarrow$ ac response

ECE 521

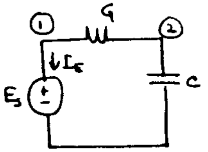
Transient Analysis

- large-signal time-domain analysis of nonlinear circuits
- provides node voltage waveforms for various input sources: pulse, sin, pwl, ...
- used as the basis for Fourier analysis

Consider a linear circuit

MNA:

$$
\begin{aligned}
G(V_1 - V_2) &+ I_E &= 0 \\
G(V_2 - V_1) &+ C\frac{dV_2}{dt} &= 0 \\
V_1 & &= E_s
\end{aligned}
$$



Given $E_s(t)$ we want to determine V_1 & V_2 as functions of time.

What do we do about $\frac{dV_2}{dt}$?

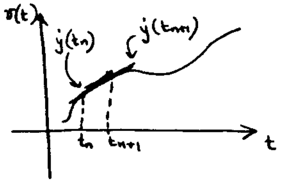
Think of $\frac{dV_2}{dt}$ as another unknown in the above system.

Then to solve these equations we need another equation.

Suppose we can write $\frac{dV_2}{dt} = \alpha V_2 + \beta$ then we have the additional equation.

Simple idea

$$\frac{dv}{dt} = \frac{v(t_{n+1}) - v(t_n)}{t_{n+1} - t_n}$$



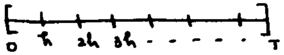
This is the Backward Euler discretization

i.e. $\frac{dv}{dt}$ is approximated by the difference relationship

So we will numerically solve the equations by discretizing time

The time derivative is then approximated by

$$\frac{d}{dt}v(t) = \frac{v(t+h) - v(t)}{h}$$



Two choices here

$$\left.\frac{d}{dt}v(t)\right|_{t+h} = \frac{v(t+h) - v(t)}{h} = \frac{v(mh) - v((m-1)h)}{h}$$

⇒ Backward Euler

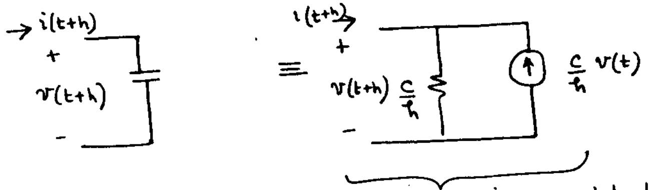
$$\left.\frac{d}{dt}v(t)\right|_{t} = \frac{v(t+h) - v(t)}{h} \Rightarrow \text{Forward Euler.}$$

What are the issues?

- what should 'h' be? Error control
- which integration method should be used?
  ⇒ understand properties of the methods
- should 'h' be uniform or nonuniform? i.e. variable time steps.

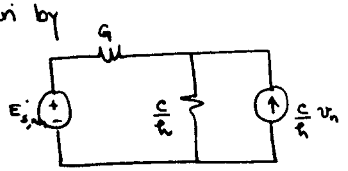
Let us revisit our example:

$$C\frac{dV_2}{dt} = C\,\frac{V_2(t+h) - v(t)}{h} = \left.C\frac{dV_2}{dt}\right|_{t+h} = i_c(t+h)$$



Companion model of the capacitor

So the original network is given by at time $t_{n+1} = t_n + h$

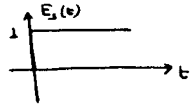


Given v_0, this network can be solved for $v_1, v_2, \dots v_N$.

Example

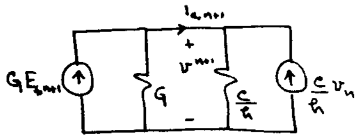
$R = 1\Omega$, $C = 1F$, $E_s(t) = 1V$ step, $h = 0.1$ sec

$v_c(0) = 0$. (Initial condition)



We use a nodal formulation here:
Then

$$v_{n+1} = \frac{1}{G + \frac{C}{h}} \left[G E_{s,n+1} + \frac{C}{h} v_n \right]$$



$$= \frac{1}{1 + 1/0.1} \left[1 + 10 v_n \right]$$

$$= \frac{1}{11} \left[1 + 10 v_n \right]$$

Now calculate values

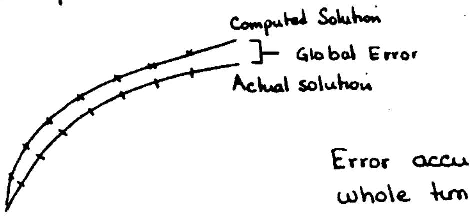
	Numerical	Exact
$v(0.1)$	0.0909	0.095
$v(0.2)$	0.174	0.181
$v(0.3)$	0.249	0.259
$v(0.4)$	0.317	0.33

Note

The exact solution is $1 - e^{-t}$

So we see that the numerical method gives us the solution but with some error. We need to understand and quantify these errors

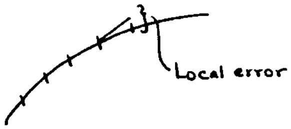
Sources of Error



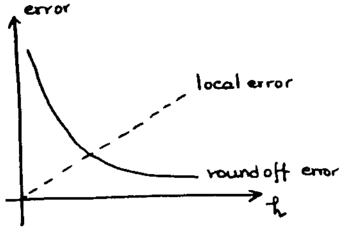
Error accumulated over the whole time interval.

The Global error is due to local error and round off.

What is local error? This is the error due to a finite h at time t, i.e. one-step error
Naturally as $h \to 0$ this error should go to zero.



For very small h the round off error becomes important!



Let us now formalize and generalize

$$\dot{x} = \frac{dx}{dt} = f(x) \quad , \quad x(0) = x_0$$

Convergent

An integration method is said to be convergent if

$$\lim_{h \to 0} \max_{0 \le m \le M} \| \hat{x}(t_m) - x(t_m) \| \to 0$$

where: $\hat{x}(t_m)$ is the computed solution and $x(t_m)$ is the true solution. $t_m = mh$ & $M = \frac{T}{h}$

We have two conditions: 1) local
2) global

Consistency

One step error is small i.e. local condition

A one-step method is consistent if given $\hat{x}(t_{m-1}) = x(t_{m-1})$

$e_L(h) = \hat{x}(t_m) - x(t_m)$ satisfies

$$\lim_{h\to 0} \frac{\|e_L(h)\|}{h} = 0$$

Why do we need this condition?

$$\text{Global Error} \cong \sum_{i=1}^{M} e_{L_i}(h) \quad \text{where } M = \frac{T}{h}$$

$$\leq M \max e_L(h)$$

$$\leq \frac{T}{h} \max e_L(h)$$

$$\lim_{h\to 0} \frac{T}{h} \max e_L(h) = T \lim_{h\to 0} \max \frac{e_L(h)}{h} = 0$$

i.e. if we want the Global error →0 as $h\to 0$ we need $\frac{e_L(h)}{h} \to 0$.

Stability

The single step errors don't grow too fast. i.e. a global condition

Classical Theorem

Consistency + Stability ⇔ Convergence

Remark

Convergent means that the Global Error →0 as $h\to 0$

Examples $\dot{x} = f(x)$

1) B.E.

$$\frac{x(mh) - x((m-1)h)}{h} = f(x(mh))$$

or $x(mh) = x((m-1)h) + h f(x(mh))$

i.e. an implicit method.

Consider $x(0)$ & $x(h)$.

Computed solution is $\hat{x}$, actual solution x

① $\hat{x}(h) = x(0) + h f(\hat{x}(h))$

② $x(h) = x(0) + h \underset{f(x(h))}{\overset{}{\dot{x}(h)}} - \frac{h^2}{2}\ddot{x}(\tau) \qquad \tau \in [0,h]$

Taylor series of $x(0)$ around $x($

Subtract ② from ①

$$\hat{x}(h) - x(h) = h\left[f(\hat{x}(h)) - f(x(h))\right] - \frac{h^2}{2}\ddot{x}(\tau)$$

$$\therefore \|\hat{x}(h) - x(h)\| \leq \underbrace{h\,\|f(\hat{x}(h)) - f(x(h))\|}_{\leq hL\|\hat{x}(h) - x(h)\|} + \frac{h^2}{2}\|\ddot{x}(c)\|$$

by Lipschitz continuity

$$\therefore (1 - hL)\|\hat{x}(h) - x(h)\| \leq \frac{h^2}{2}\|\ddot{x}(\tau)\|$$

$$\Rightarrow \|\hat{x}(h) - x(h)\| \leq \frac{1}{2}\frac{h^2}{1-hL}\|\ddot{x}(\tau)\|$$

Whereby $\lim_{h\to 0} \frac{\|\hat{x}(h) - x(h)\|}{h} = 0$ i.e. consistent

2) F.E.

$$\frac{x(mh) - x((m-1)h)}{h} = f(x((m-1)h))$$

or $x(mh) = x((m-1)h) + h f(x((m-1)h))$

This is an explicit method

$$\hat{x}(h) = x(0) + h f(x(0))$$

$$x(h) = x(0) + h f(x(0)) + \frac{h^2}{2}\ddot{x}(\tau), \quad \tau\in[0,h]$$

$$\therefore \quad \hat{x}(h) - x(h) = \frac{h^2}{2}\dot{x}(\tau)$$

$$\Rightarrow \|\hat{x}(h) - x(h)\| \leqslant \frac{h^2}{2}\|\ddot{x}(\tau)\|$$

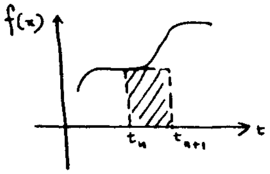
and $\displaystyle\lim_{h\to 0} \frac{\|\hat{x}(h) - x(h)\|}{h} = 0$ i.e. consistent.

Summary

$$\dot{x} = f(x)$$

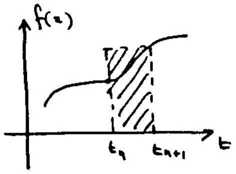
FE $\qquad \dot{x}_n = \dfrac{x_{n+1} - x_n}{h}$

or $\quad x_{n+1} = x_n + h\dot{x}_n = x_n + hf(x_n)$



BE $\qquad \dot{x}_{n+1} = \dfrac{x_{n+1} - x_n}{h}$

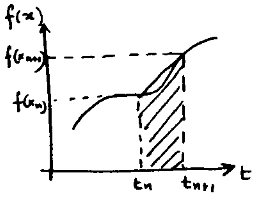
$$\Rightarrow x_{n+1} = x_n + hf(x_{n+1})$$



TR $\qquad \dfrac{\dot{x}_n + \dot{x}_{n+1}}{2} = \dfrac{x_{n+1} - x_n}{h}$

$$\Rightarrow x_{n+1} = x_n + \frac{h}{2}\left(\dot{x}_n + \dot{x}_{n+1}\right)$$

$$= x_n + \frac{h}{2}\Big(f(x_n) + f(x_{n+1})\Big)$$



Remarks

- FE, BE and TR are all one-step methods, i.e. they use only the information at the previous timepoint
- FE is an explicit method i.e. x_{n+1} can be easily computed
- BE, TR are implicit methods

Linear Multistep Methods

$$\dot{x} = \frac{dx}{dt} = f(x)$$

$\displaystyle\sum_{i=0}^{p} \alpha_i x_{n-i} + h\sum_{i=0}^{p} \beta_i \dot{x}_{n-i} = 0$ is a p-step method

Examples

1) FE $\qquad x_n = x_{n-1} + h\dot{x}_{n-1}$

$$\Rightarrow \alpha_0 = 1, \quad \alpha_1 = -1, \quad \beta_0 = 0, \quad \beta_1 = -1$$

2) BE $\qquad x_n = x_{n-1} + h\dot{x}_n$

$$\Rightarrow \alpha_0 = 1, \quad \alpha_1 = -1, \quad \beta_0 = -1, \quad \beta_1 = 0$$

3) TR $\qquad x_n = x_{n-1} + \frac{h}{2}\left(\dot{x}_{n-1} + \dot{x}_n\right)$

$$\Rightarrow \alpha_0 = 1, \quad \alpha_1 = -1, \quad \beta_0 = -1/2, \quad \beta_1 = -1/2$$

For an explicit method $\beta_0 = 0$; for an implicit method $\beta_0 \neq 0$

Revisit Errors

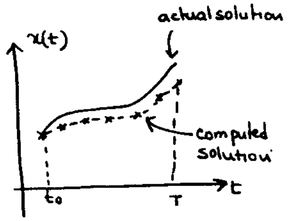
Let $x(t)$ be the solution of

$$\dot{x}(t) = f(x(t))$$

Denote by x_n, the points computed by the numerical integration method.

Global Error at t_n

$$\epsilon_n = \|x(t_n) - x_n\|$$



The error has two components

1) truncation error - due to time discretization
2) round-off error - due to computation

Note previous errors propogate

Local Truncation Error (LTE)

$$LTE_n \triangleq x(t_n) - x_n$$

assuming no round-off error and that no previous error has been made, i.e.,

$$x_i = x(t_i) \qquad i = 0, 1, \dots n-1$$

This is the error made in one step assuming past data is exact

Local Error (LE)

$$LE_n \triangleq \sum_{i=0}^{p} \alpha_i x(t_{n-i}) + h\sum_{i=0}^{p} \beta_i \dot{x}(t_{n-i})$$

it is the amount by which the linear multi-step formula is wrong. i.e. the error by plugging in the exact solution in the LMS formula.

What is the relationship between LTE_n and LE_n?

$$LTE_n = x(t_n) - x_n$$

From the LTE definition x_n is obtained as a solution of

$$x_n + \sum_{i=1}^{p} \alpha_i x(t_{n-i}) + h\beta_0 \dot{x}_n + h\sum_{i=1}^{p} \beta_i \dot{x}(t_{n-i}) = 0$$

i.e. $$x_n + \sum_{i=0}^{p} \alpha_i x(t_{n-i}) - x(t_n) + h\beta_0 \dot{x}_n + h\sum_{i=0}^{p} \beta_i \dot{x}(t_{n-i}) - h\beta_0 \dot{x}(t_n) = 0$$

$$\therefore \underbrace{x(t_n) - x_n}_{LTE_n} = \underbrace{\sum_{i=0}^{p} \alpha_i x(t_{n-i}) + h\sum_{i=0}^{p} \beta_i \dot{x}(t_{n-i})}_{LE_n} - h\beta_0 \left(\dot{x}(t_n) - \dot{x}_n\right)$$

$$LTE_n = x(t_n) - x_n = LE_n - h\beta_0 \left(f(x(t_n) - f(x_n)) \right)$$

$$\Rightarrow \quad |LTE_n| \leq |LE_n| + |h\beta_0| \left| f(x(t_n) - f(x_n) \right|$$

$$\leq |LE_n| + |h\beta_0| L \, |\underbrace{x(t_n) - x_n}_{|LTE_n|}|$$

$$\therefore \quad |LTE_n| \leq \frac{|LE_n|}{1 - |h\beta_0| L}$$

Remarks

- $LTE_{n+1} = LE_{n+1}$ if $\beta_0 = 0$ i.e. for explicit method.
- LE bounds LTE and we will use LE to estimate the error.

Now we are interested in seeing how the error varies with h. Given a LMS method only the timestep, i.e., h can be controlled so we need to understand how h affects the error!

$$E[x(t), h] = LE_n = \sum_{i=0}^{p} \alpha_i x(t_{n-i}) + h\beta_i \dot{x}(t_{n-i})$$

We will expand this as a Taylor's series in h

$$E[x,h] = E[x,0] + E^{(1)}[x,0]h + E^{(2)}[x,0]\frac{h^2}{2} + \cdots$$
$$+ E^{(k+1)}[x,0]\frac{h^{k+1}}{(k+1)!} + O(h^{k+2})$$

Let us see what $E^{(1)}[x,0]$ is

$$E^{(1)}[x,h] = \sum_{i=0}^{p} \alpha_i\, x^{(1)}(t_n - ih)(-i) + \sum_{i=0}^{p} \beta_i x^{(1)}(t_n - ih)$$
$$+ h\sum_{i=0}^{p} \beta_i\, x^{(2)}(t_n - ih)(-i)$$

<u>Definition</u>

A multi-step method is said to be a <u>k-th order</u> method if it is exact for polynomials of degree k.

i.e. $E^{(i)}[x,0] = 0 \qquad$ for $0 \leq i \leq k$

<u>Note</u> $\qquad x^{(k+1)}(t) = 0 \qquad$ for a polynomial of degree k

<u>For a uniform step size</u>

$$0 = E[x,0] \Rightarrow \sum_{i=0}^{p} \alpha_i = 0$$
$$0 = E^{(1)}[x,0] \Rightarrow \sum_{i=0}^{p} \alpha_i(-i) + \beta_i = 0$$
$$\vdots$$
$$0 = E^{(k)}[x,0] \Rightarrow \sum_{i=0}^{p} \alpha_i(-i)^k + k\,\beta_i(-i)^{k-1} = 0$$

This set of equations is called <u>Exactness Constraints</u>

<u>Examples</u>

<u>BE</u> $\qquad x_n = x_{n-1} + h\dot{x}_n$

$$\alpha_0 = 1, \quad \alpha_1 = -1, \quad \beta_0 = -1, \quad \beta_1 = 0$$

Lets examine the exactness constraints

$$\alpha_0 + \alpha_1 = \sum \alpha_i = 0 \qquad\qquad E[x,0] = 0$$

$$-\alpha_1 + \beta_0 + \beta_1 = -(-1) + (-1) + 0 = 0 \qquad\qquad E^{(1)}[x,0] = 0$$

$$\alpha_1(-1)^2 + 2\beta_1(-1)^1 = \alpha_1 - 2\beta_1 = (-1) - 0 \neq 0 \qquad E^{(2)}[x,0] \neq 0$$

Therefore, BE is a <u>first-order</u> method.

<u>TR</u> $\qquad x_n = x_{n-1} + \frac{h}{2}(\dot{x}_n + \dot{x}_{n-1})$

$$\alpha_0 = 1, \quad \alpha_1 = -1, \quad \beta_0 = -\frac{1}{2}, \quad \beta_1 = -1/2$$

$$\alpha_0 + \alpha_1 = 1 + (-1) = 0 \qquad\qquad E[x,0] = 0$$
$$-\alpha_1 + \beta_0 + \beta_1 = -(-1) + (-\tfrac{1}{2}) + (-\tfrac{1}{2}) = 0 \qquad\qquad E^{(1)}[x,0] = 0$$
$$\alpha_1 - 2\beta_1 = (-1) - 2(-1/2) = 0 \qquad\qquad E^{(2)}[x,0] = 0$$

$$\alpha_1(-1)^3 + 3\beta_1(-1)^2 = -\alpha_1 + 3\beta_1 \neq 0 \qquad\qquad E^{(3)}[x,0] \neq 0$$

Therefore, TR is a <u>second-order</u> method

<u>Recall</u> $\quad$ Both BE & TR are one-step methods.

<u>Remark</u>

Order indicates how accurate a method is. Therefore, TR is more accurate than BE since it is a higher order method.

Synthesis of LMS Methods

Suppose we have a p-step method of order k. Take $\alpha_0 = 1$.

Exactness constraints $\Rightarrow$ k+1 equations
p-step method $\Rightarrow$ $(p+1) + (p+1) - 1 = 2p+1$ unknowns

Thus we need $2p+1 \geq k+1$ or $p \geq \frac{k}{2}$

Algorithm for choosing coefficients

1) Choose order of method – k (i.e. accuracy)
2) Choose p, the number of steps $p \geq \frac{k}{2}$
3) Write the k+1 exactness constraints
4) For $p > \frac{k}{2}$, $2p+1-(k+1) = 2p-k$ coefficients may be assigned arbitrarily, or other constraints may be used.

Example

Suppose $p = k$ then k coefficients need additional constraints

Take $\beta_1 = \beta_2 \cdots = \beta_p = 0$

i.e. $$\sum_{i=0}^{k} \alpha_i x_{n-i} + h\beta_0 \dot{x}_n = 0$$

This is the <u>kth-order</u> backward differentiation formula (BDF).

For a 2nd-order BDF method $\alpha_2, \alpha_2, \beta_0$ determined by exactness constraints.

Exactness Constraints

$$\alpha_0 + \alpha_1 + \alpha_2 = 1 + \alpha_1 + \alpha_2 = 0$$
$$\alpha_1(-1) + \alpha_2(-2) + \beta_0 = 0$$
$$\alpha_1(-1)^2 + \alpha_2(-2)^2 + 2\beta_0(0) = 0$$

i.e. 3 equations in 3 unknowns

$$\left.\begin{aligned} \alpha_1 + \alpha_2 &= -1 \\ -\alpha_1 - 2\alpha_2 + \beta_0 &= 0 \\ \alpha_1 + 4\alpha_2 &= 0 \end{aligned}\right] \Rightarrow \quad \alpha_1 = -4/3, \ \alpha_2 = 1/3, \ \beta_0 = -2/3$$

$\therefore$ the integration method is

$$x_n - \frac{4}{3}x_{n-1} + \frac{1}{3}x_{n-2} + h\left(-\frac{2}{3}\right)\dot{x}_n = 0$$

or $$x_n = \frac{4}{3}x_{n-1} - \frac{1}{3}x_{n-2} + \frac{h2}{3}\dot{x}_n$$

or $$\dot{x}_n = \frac{1}{h}\left[\frac{3}{2}x_n - 2x_{n-1} + \frac{1}{2}x_{n-2}\right]$$

Local Error for a kth-order Method

If the method is of order k then
$$E[x,0] = E^{(1)}[x,0] = \cdots = E^{(k)}[x,0] = 0$$

$$\Rightarrow \quad LE = \frac{E^{(k+1)}[x,0]}{(k+1)!}h^{k+1} + O(h^{k+2})$$

$$\cong \underbrace{\frac{1}{(k+1)!}\left[\sum_{i=0}^{p} \alpha_i(-i)^{k+1} + (k+1)\beta_i(-i)^{k}\right]}_{C_{k+1}} h^{k+1} x^{(k+1)}(t_n)$$

$$\cong C_{k+1} h^{k+1} x^{(k+1)}(t_n)$$

More on Local Error

For LMS Method : $\sum_{i=0}^{p} \alpha_i x_{n-i} + h \sum_{i=0}^{p} \beta_i \dot{x}_{n-i} = 0$ —— ①

we have the Local Error as

$$LE \cong \frac{E^{(k+1)}[x,0]}{(k+1)!} h^{k+1} \qquad \text{for kth-order method}$$

$$\Rightarrow \quad LE = \frac{1}{(k+1)!}\left[\sum_{i=0}^{p} \alpha_i (-i)^{k+1} + (k+1)\sum_{i=0}^{p} \beta_i (-i)^{k}\right] h^{k+1} x^{(k+1)}(t_n)$$

If ① is scaled such that the LMS formula is now given by

$$\sum_{i=0}^{p} \alpha_i' x_{n-1} + h\sum_{i=0}^{p} \beta_i' \dot{x}_{n-i} = 0 \qquad \text{—— ②}$$

where $\alpha_i' = \frac{\alpha_i}{\eta}$ and $\beta_i' = \frac{\beta_i}{\eta}$

This scaling suggests that the LE will also be reduced. However, the values of x computed from ① or ② are the same ⇒ accuracy of the method doesn't change

∴ need a normalization such that LE is the same if either ① or ② is used.

The normalization is $\sum_{i=0}^{p} \beta_i = -1$

Example

BE $\quad x_n - x_{n-1} - h\dot{x}_n = 0 \quad \Rightarrow \sum \beta_i = -1$

TR $\quad x_n - x_{n-1} - \frac{h}{2}(\dot{x}_n + \dot{x}_{n-1}) = 0 \Rightarrow \sum \beta_i = -\frac{1}{2} + (-\frac{1}{2}) = -1$

Examples

FE : $\alpha_0 = 1,\ \alpha_1 = -1,\ \beta_0 = 0,\ \beta_1 = -1,\quad p = 1,\ k = 1$

$$c_{k+1} = \frac{1}{2}\left[\alpha_1(-1)^2 + 2\beta_1(-1)\right] = \frac{1}{2}(-1+2) = \frac{1}{2}$$

BE $\quad \alpha_0 = 1,\ \alpha_1 = -1,\ \beta_0 = 1,\ \beta_1 = 0,\ p = 1,\ k = 1$

$$c_2 = \frac{1}{2}\left[\alpha_1(-1)^2 + 2\beta_1(-1)\right] = -\frac{1}{2}$$

TR $\quad \alpha_0 = 1,\ \alpha_1 = -1,\ \beta_0 = -1/2,\ \beta_1 = -1/2,\ p = 1,\ k = 2$

$$c_3 = \frac{1}{3!}\left[\alpha_1(-1)^3 + 3\beta_1(-1)^2\right] = \frac{1}{6}\left[(-1)(-1) + 3(-1/2)\right]$$

$$= -\frac{1}{12}$$

i.e. $\quad LE \cong -\frac{1}{12} h^3 x^{(3)}(t_n)$

Remark

- Two kth order methods $c_{k+1,1}$, $c_{k+1,2}$. The method that has a smaller c_{k+1} has better accuracy.

Summary

- Linear Multi-Step Methods (LMS)
  $\sum_{i=0}^{p} \alpha_i x_{n-i} + h\beta_i \dot{x}_{n-i} = 0 \qquad$ p-step method
- kth order method ⇒ exact for all polynomials of degree k or less. This gives exactness constraints
- LE is the error by which the exact solution fails to satisfy the LMS formula
  $LE = C_{k+1} h^{(k+1)} x^{(k+1)}(t_n)$.

Implicit integration Methods

$$\dot{x} = f(x)$$

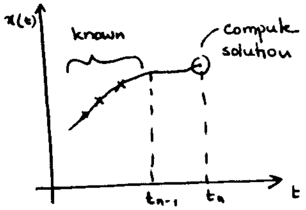
Integration method: $\sum_{i=0}^{p} \alpha_i x_{n-i} + h\beta \dot{x}_{n-i} = 0$; $\beta_0 \neq 0$

$$\rightarrow \quad x_n + \sum_{i=1}^{p} \alpha_i x_{n-i} + h\beta_i \dot{x}_{n-i} + h\beta_0 \dot{x}_n = 0$$

or $\underbrace{x_n + h\beta_0 f(x_n)}_{\text{These are computed at the current time point}} + \underbrace{\sum \alpha_i x_{ni} + h\beta_i \dot{x}_{n-i}}_{\text{These have been computed the previous time points}} = 0$

Thus we have

$x_n + h\beta_0 f(x_n) = \underset{b}{\text{known quantity}}$



i.e. $F(x_n) = x_n + h\beta_0 f(x_n) - b = 0$

is a nonlinear equation in x_n and we want to solve x_n

How do we determine x_n?

Solve the nonlinear equation using Newton's method

i.e. $\left.\frac{\partial F}{\partial x_n}\right|_k \left(x_n^{(k+1)} - x_n^{(k)}\right) = -F(x_n^{(k)})$

Newton's method will converge provided $x_n^{(0)}$ is close to x_n - the actual solution.

Take $x_n^{(0)} = x_{n-1}$, then for h small enough $x_n^{(0)}$ is a good initial guess

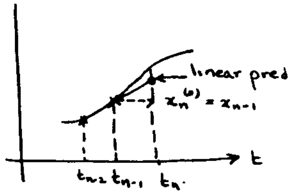
Note

- A similar idea was used in the Newton continuation method.

Idea

Instead of $x_n^{(0)} = x_{n-1}$, use the previous timepoint values to predict a better initial guess.

For a first-order method a linear prediction can be used



In general use extrapolation to predict $x_n^{(0)}$. Fit the past data with a kth degree polynomial for a kth order method.

$P_k(t_n) \rightarrow$ polynomial of degree k through the points $x_{n-1}, x_{n-2}, \ldots x_{n-k-1}$

Examples

1) Backward Euler k=1 i.e. 1st order prediction

$$P_1(t) = x_{n-1} + \dot{x}_{n-1}(t - t_{n-1}) \qquad \text{(FE method)}$$
$$= x_{n-1} + \frac{(x_{n-1} - x_{n-2})}{(t_{n-1} - t_{n-2})}(t - t_{n-1})$$

2) TR k = 2 i.e. use 2nd order prediction

$$P_2(t) = x_{n-1} + \dot{x}_{n-1}(t - t_{n-1}) + \frac{\ddot{x}_{n-1}}{2}(t - t_{n-1})^2$$

$$\dot{x}_{n-1} = \frac{x_{n-1} - x_{n-2}}{t_{n-1} - t_{n-2}}$$

$$\ddot{x}_{n-1} = \frac{\dot{x}_{n-1} - \dot{x}_{n-2}}{t_{n-1} - t_{n-2}} = \frac{\frac{x_{n-1} - x_{n-2}}{t_{n-1} - t_{n-2}} - \frac{x_{n-2} - x_{n-3}}{t_{n-2} - t_{n-3}}}{t_{n-1} - t_{n-2}}$$

The derivatives are expressed in terms of divided differences.

Remark

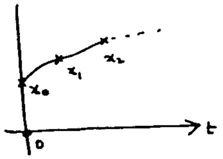
- Methods that use a predictor are called predictor-corrector methods. The "corrector" is the solution determined by the application of the integration formula & Newton's method.

Note

A p-step method requires values from p-previous timepoints

Questions

How do we start a multi-step method i.e. what do we use for x_1 since the only known quantities are at $t=0$



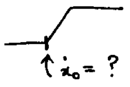
$\Rightarrow$ Use a single-step method

what is the value of x_0?

This is the dc operating point value i.e. $t=0$ value. It is for this reason that SPICE and other simulators perform a dc operating point analysis before starting the transient analysis.

TR is a one-step method that requires the value of $\dot{x}_0$ to start. An $\dot{x}_0 = 0$ is typically used. However, this can be a problem with ideal waveforms for the sources

Remark



It is important to have a correct set of initial conditions. If the initial conditions are not correct $\Rightarrow$ inconsistent initial conditions. Switching circuits will result in inconsistent ICs.

Remark

A kth-order integration method satisfies ($k \geq 1$)

$$\lim_{h \to 0} \frac{LTE}{h} = 0$$

Therefore, the method is consistent. Alternatively, a method is consistent if $k \geq 1$.

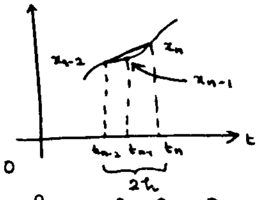
We have considered local errors previously. Now we must address the question

How do local errors accumate? We now consider stability of the integration method.

Example Mid-point Method

$$x_n = x_{n-2} + 2h\dot{x}_{n-1}$$

$$\text{i.e.} \quad \dot{x}_{n-1} = \frac{x_n - x_{n-2}}{2h}$$



Note: This is an explicit method $\beta_0 = 0$

$\alpha_0 = 1, \quad \alpha_1 = 0, \quad \alpha_2 = -1, \quad \beta_0 = 0, \quad \beta_1 = -2, \quad \beta_2 = 0$

$\Rightarrow$ 2-step method

Exactness constraints:

$$\alpha_0 + \alpha_1 + \alpha_2 = 1 + 0 - 1 = 0$$
$$-\alpha_1 - 2\alpha_2 + \beta_0 + \beta_1 + \beta_2 = -0 + 2 + 0 - 2 + 0 = 0$$
$$\alpha_1(-1)^2 + \alpha_2(-2)^2 + 2\beta_1(-1)^1 + 2\beta_2(-2)^1 \Rightarrow$$
$$0 + 4(-1) + 2(-2)(-1) = 0$$
$$\alpha_1(-1)^3 + \alpha_2(-2)^3 + 3\beta_1(-1)^2 + 3\beta_2(-2)^2$$
$$0 + (-1)(-8) + 3(-2)(1) \neq 0$$

$\Rightarrow$ This is a second-order method.
$\Rightarrow$ the method is consistent (local behavior OK!).

Now consider solving the equation $\dot{x} = -x$, $x(0) = 1$ using this method

<u>Exact Solution</u> $x(t) = e^{-t}$

Assume a uniform step size h.

$$\dot{x}(0) = -x(0) = -1$$

<u>Note</u> $\dot{x} = -x \Rightarrow \dot{x}_n = -x_n$

Take $h=1$: $\dot{x}_1 = -0.368$, $x_1 = 0.368$

$$\begin{aligned}
x_2 &= x_0 + 2h\dot{x}_1 = 1 + 2(1)(-.368) = 0.264 \\
\dot{x}_2 &= -x_2 = -0.264 \\
x_3 &= x_1 + 2h\dot{x}_2 = 0.368 + 2(1)(-.264) = -0.16 \\
x_4 &= x_2 + 2h\dot{x}_3 = 0.264 + 2(1)(+0.16) = 0.584 \\
x_5 &= -0.16 + 2(1)(-0.584) = -1.328 \\
x_6 &= 0.584 + 2(1)(+1.328) = 3.24 \\
x_7 &= -1.328 + 2(1)(-3.24) = -7.808
\end{aligned}$$

and the solution blows up!

Try $h = 0.1$ ... $x_{9.9} = 44.03$, $x_{10} = -48.65$,
$h = 0.01$ .. $x_{12} = 12124.17$

No matter how small h is the solution blows up.
$\Rightarrow$ the method is unstable for any h.

What about the methods we have seen so far?

1) FE : $x_n = x_{n-1} + h\dot{x}_{n-1}$

$h=1$:
$$\begin{aligned}
x_1 &= x_0 + \dot{x}_0 = 1 + (-1) = 0 \\
x_2 &= x_1 + \dot{x}_1 = 0 + (-0) = 0 \\
&\vdots \\
x_n &= 0
\end{aligned}$$

so the solution doesn't grow, although we don't get the correct solution

Now take $h = 3$

$$\begin{aligned}
x_n &= x_{n-1} + 3\dot{x}_{n-1} \\
x_1 &= 1 + 3(-1) = -2 \\
x_2 &= -2 + 3(2) = 4 \\
x_3 &= 4 + 3(-4) = -8 \\
x_4 &= -8 + 3(8) = 16
\end{aligned}$$

i.e. there are oscillations that grow!!

<u>Check</u> BE & TR will not have growing oscillations, whatever timestep is used.

So we see that there are three types of methods

1) "Good" methods - stable regardless of h (BE, TR,
2) semi-Good methods - stable for some values of h, unstable for others (FE)
3) Bad methods - unstable for all values of h (explicit mid point formula)

We will now formalize these ideas by using a test problem

<u>Test problem</u> $\dot{x} = +\lambda x$, $x(0) = 1$, λ complex.

<u>Why this problem?</u>
- simple enough to study behavior
- local behavior of nonlinear systems can be approximated by
$$\delta\dot{x} = A(t)\,\delta x$$
i.e. linearization of nonlinear problem

General Multistep Method

$$\Rightarrow \quad \sum_{i=0}^{p} \alpha_i x_{n-i} + h\beta_i \dot{x}_{n-i} = 0$$

Test equation : $\dot{x} = \lambda x, \; x(0) = 1$

solution $x(t) = e^{\lambda t}$

The difference eqn gives

$$\sum_{i=0}^{p} \alpha_i x_{n-i} + h\beta_i \lambda x_{n-i} = \sum_{i=0}^{p} (\alpha_i + h\lambda\beta_i) x_{n-i} = 0$$

$$= \sum_{i=0}^{p} (\alpha_i + \sigma\beta_i) x_{n-i} = 0 \quad , \text{where } \sigma = h\lambda.$$

What the integration method computes is the solution of this difference equation.

Try a solution of the form $x_n = cr^n$

$$\Rightarrow \quad (\alpha_0 + \sigma\beta_0) cr^n + (\alpha_1 + \sigma\beta_1) cr^{n-1} + \cdots + (\alpha_p + \sigma\beta_p) cr^{n-p} = 0$$

or $(\alpha_0 + \sigma\beta_0) r^p + (\alpha_1 + \sigma\beta_1) r^{p-1} + \cdots + (\alpha_p + \sigma\beta_p) = 0$

i.e. we have a solution if r is a root of the polynomial

$$(\alpha_0 + \sigma\beta_0) z^p + (\alpha_1 + \sigma\beta_1) z^{p-1} + \cdots + (\alpha_p + \sigma\beta_p)$$

This polynomial will have p roots - in general complex

Distinct roots $\Rightarrow \quad x_n = c_1 r_1^n + c_2 r_2^n + \cdots + c_p r_p^n$

Multiple roots ; multiplicity of r_i is m_i

$$\Rightarrow \quad x_n = (c_{i0} + c_{i1} n + \cdots + c_{im_i} n^{m_i - 1}) r_i^n$$

Thus x_n is <u>stable</u> under the following conditions

1) $|r_i| < 1$ for all i
2) $|r_i| = 1$ and $m_i = 1$, other $|r_j| < 1$

<u>Unstable</u> when

1) $|r_i| > 1$ for some i
2) $|r_i| = 1$ and $m_i > 1$

<u>Examples</u>

1) <u>Explicit Midpoint</u>

$$x_n = x_{n-2} + 2h\dot{x}_{n-1}$$

$$= x_{n-2} + 2h\lambda x_{n-1}$$

$$\Rightarrow \quad x_n - x_{n-2} - 2\underbrace{h\lambda}_{\sigma} x_{n-1} = 0$$

Now the polynomial is :

$$z^2 - 2\sigma z - 1 = 0$$

The roots are $z_{1,2} = \sigma \pm \sqrt{\sigma^2 + 1} \quad \Rightarrow |z_1| > 1$

$\therefore$ the method is unstable for all h

2) <u>BE</u>

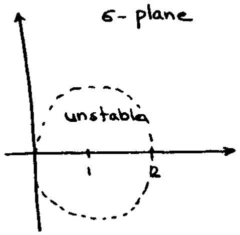
$$x_n = x_{n-1} + h\dot{x}_n$$

$$\Rightarrow \quad x_n - (x_{n-1} + h\dot{x}_n) = 0$$

$$z - 1 - \sigma z = 0$$

$$\Rightarrow \quad z = \frac{1}{1-\sigma}$$

$$|z| \leq 1 \quad \Rightarrow \quad \frac{1}{|1-\sigma|} \leq 1$$



i.e. for all $\text{Re}(\sigma) < 0$ the method is stable for all h $\sigma = \lambda h$, $h > 0$ $\Rightarrow \text{Re}(\lambda) < 0$. i.e. for a stable system $\dot{x} = \lambda x$ the numerical solution is also <u>stable</u>

Remark

For $\text{Re}\,\lambda > 0$, the differential eqn. is unstable. However, the numerical solution may be stable. This is a problem when simulating oscillators. BE damps out the oscillations, and is not a desireable method for simulating oscillators.

Region of absolute stability

for a LMS method is the set of $\sigma = \lambda h$ (complex) such that all solutions of the difference equation

$$\sum_{i=0}^{p} (\alpha_i + \sigma\beta_i)\, x_{n-i} = 0$$

remain bounded as $n \to \infty$

Stable method

if all the solutions of the associated difference equation obtained by setting $\sigma = 0$ remain bounded as $n \to \infty$ i.e. the stability region contains the point $\sigma = 0$.

Theorem

The solutions of $\sum_{i=0}^{p} (\alpha_i + \sigma\beta_i)\, x_{n-i} = 0$ are bounded

if and only if the roots of the associated polynomial

$$\sum_{i=0}^{p} (\alpha_i + \sigma\beta_i)\, z^{p-i} = 0$$

$z_1, z_2, \ldots z_r$ (r = # of distinct roots) are inside or on the complex unit circle $|z| \leq 1$ and the roots for which $|z| = 1$ are of multiplicity one.

(multiplicity 2 $\Rightarrow$ $c_1 + c_2 n$)

Remark

The roots of the polynomial are functions of σ. Therefore, the region of absolute stability is the set of values of σ for which the necessary and sufficient conditions are satisfied.

$\Rightarrow$ bound on timestep that can be used!

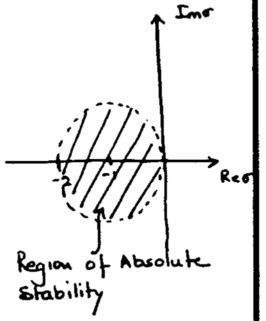
Examples

1) FE

$$x_n = x_{n-1} + h\dot{x}_{n-1}$$
$$= x_{n-1} + \sigma x_{n-1}$$

or $z - (1+\sigma) = 0 \Rightarrow z = 1+\sigma$

$\therefore |z| \leq 1 \iff |1+\sigma| \leq 1$



2) TR

$$x_n = x_{n-1} + \frac{h}{2}\left(\dot{x}_{n-1} + \dot{x}_n\right)$$

$$\Rightarrow \quad x_n - x_{n-1} - \frac{h}{2}\left(\lambda x_{n-1} + \lambda x_n\right) = 0$$

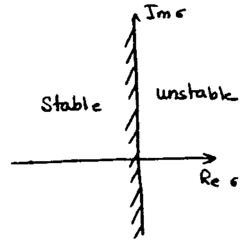
$$x_n - \frac{\sigma}{2} x_n - x_{n-1} - \frac{\sigma}{2} x_{n-1} = 0$$

$\therefore$ the polynomial is $\left(1 - \frac{\sigma}{2}\right) z - \left(1 + \frac{\sigma}{2}\right) = 0$

whereby $z = \dfrac{1+\sigma/2}{1-\sigma/2}$

and $|z| \leq 1 \iff \left|\dfrac{1+\sigma/2}{1-\sigma/2}\right| \leq 1$

For $\text{Re}(\sigma) \leq 0$, this condition is always satisfied.



Remark

TR is stable exactly when $\dot{x} = \lambda x$ is stable i.e. it is an ideal method.

A - stability

A method is A-stable if its region of absolute stability includes the open LHP.

Remark

TR is an A-stable method.

Consider the TR method

$\mathrm{Re}(\lambda) < 0$

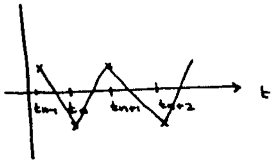
$$x_n\left(1-\frac{\lambda h}{2}\right) - \left(1+\frac{\lambda h}{2}\right) x_{n-1} = 0$$

$$\Rightarrow x_n = \left(\frac{1+\lambda h/2}{1-\lambda h/2}\right) x_{n-1}$$

Take $h \to \infty$ i.e. a large value then

$$x_n = -x_{n-1}$$

$\Rightarrow$ ringing in the trapezoidal method!



L - Stability

An A-stable method is L-stable if $\mathrm{Re}(\lambda) < 0$ implies

$$\lim_{h\to\infty} x_n = 0 \qquad \text{for all } x_{n-1}$$

Remark

TR is not an L-stable method, so there is a potential for ringing unless the stepsize h is carefully controlled.

Determining the Region of Absolute Stability

Recall $\quad (1+\sigma\beta_0)z^p + (\alpha_1+\sigma\beta_1)z^{p-1} + \cdots + (\alpha_p+\sigma\beta_p) = 0$

We want to find the values of σ for which all p roots of the above polynomial satisfy the stability conditions.

Methods

1) Choose σ, compute the roots, test
   repeat for all σ
   This method is not very practical

2) Solve for $\sigma = -\frac{P_N(z)}{P_D(z)}$ from the equation

$$\underbrace{\left(z^p + \alpha_1 z^{p-1} + \cdots + \alpha_p\right)}_{P_N(z)} + \sigma\underbrace{\left(\beta_0 z^p + \beta_1 z^{p-1} + \cdots \beta_p\right)}_{P_D(z)} = 0$$

Now look at the set

$$S = \left\{ \sigma \;\middle|\; \sigma = -\frac{P_N(z)}{P_D(z)}, \quad |z| \leq 1 \right\}.$$

i.e. let z vary in $|z| \leq 1$ and note values of σ

There is a problem here

may get same σ values for two or more different z's. One with $|z| \leq 1$ and one with $|z| > 1$

$\Rightarrow$ S is a superset of the absolute stability region

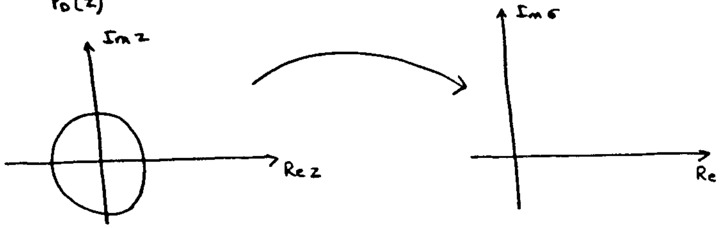
So what do we do now?

Define

The boundary locus Γ_σ as the contour $-\frac{P_N(z)}{P_D(z)}$ when $|z|=1$

Observations

- $-\frac{P_N(z)}{P_D(z)}$ defines a map from the z-plane to the σ-plane



- Ensure that for a certain σ all the roots of $P_N(z) + \sigma P_D(z)$ are inside the unit circle.

Consider $|z| = 1$, in polar coordinates $z = e^{j\theta}$ $0 \leq \theta \leq 2\pi$

Then Γ_σ is the locus described by

$$\sigma(\theta) = -\frac{P_D(e^{j\theta})}{P_N(e^{j\theta})}$$

Basic facts on Γ_σ

1) Γ_σ separates the σ-plane into sets such that the # of roots outside the unit circle has a constant value in each set.
2) The boundary of the stability region is a subset of Γ_σ
3) The mapping from the unit circle to the boundary locus is conformal

Similar to Nyquist stability theory

Implication

Move counterclockwise along Γ_σ then you see one more root outside the unit circle for those σ to the right than for those σ to the left.

Examples

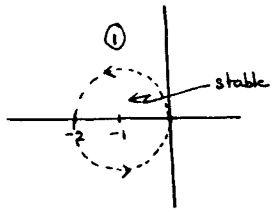
1) FE: $x_n = x_{n-1} + h\dot{x}_{n-1}$

$$x_n - x_{n-1} - \sigma x_{n-1} = 0$$

$$z - (1+\sigma) = 0 \quad \text{or} \quad \sigma = z-1$$

Take $z = e^{j\theta} \Rightarrow \sigma = e^{j\theta} - 1$

Vary θ: $0 \leq \theta \leq 2\pi$



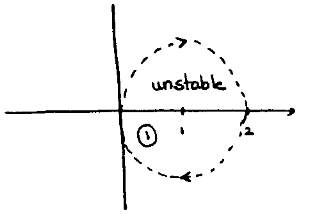
2) BE $x_n = x_{n-1} + h\dot{x}_n$

$$x_n - x_{n-1} - \sigma x_n = 0$$

$$z - 1 - \sigma z = 0$$

$$\Rightarrow \sigma = \frac{z-1}{z} = 1 - \frac{1}{z}$$

$z = e^{j\theta} \Rightarrow \sigma = 1 - \frac{1}{e^{j\theta}} = 1 - e^{-j\theta}$

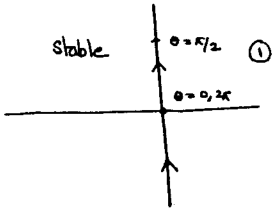


3) TR $x_n = x_{n-1} + \frac{h}{2}(\dot{x}_{n-1} + \dot{x}_n)$

$$x_n - x_{n-1} - \frac{h}{2}\lambda x_{n-1} - \frac{h\lambda}{2} x_n = 0$$

$$z - 1 - \frac{\sigma}{2} - \frac{\sigma}{2} z = 0$$

$$\sigma = \frac{2(z-1)}{z+1} = \frac{2(e^{j\theta}-1)}{e^{j\theta}+1}$$



Recall

A method is A-stable if the region of absolute stability includes the open LHP

Dahlquist's Theorem

An A-stable LMS method cannot exceed order 2. The most accurate A-stable method is TR, i.e., it has the smallest truncation error.

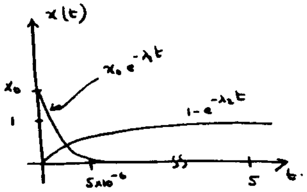
Stiff Problems

Consider $\dot{x} = -\lambda_1 (x - s(t)) + \frac{ds}{dt}$ $\quad x(0) = x_0$

$s(t) = 1 - e^{-\lambda_2 t}$

Where $\lambda_1 = 10^6$, $\lambda_2 = 1$.

The exact solution is $x(t) = x_0 e^{-\lambda_1 t} + 1 - e^{-\lambda_2 t}$



For $t > 5 \times 10^{-6}$ $\quad x_0 e^{-\lambda_1 t} \cong 0$

$t > 5$ $\quad 1 - e^{-\lambda_2 t} \simeq 1$

So the problem spans an interval of [0, 5] wherein the first term of the solution decays to zero in 5×10^{-6}.

Suppose a uniform step size was being used to solve the problem

$h < 10^{-6}$ for accuracy & resolving the first term

$\Rightarrow$ 5×10^6 time steps for the complete solution

This is a stiff system one that is characterized by widely varying time constants

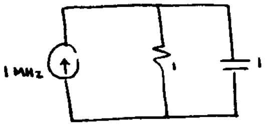
A better solution is to use variable steps. Small steps initially and then much larger steps for the example problem.

What are stiff problems?

- widely separated time constants i.e. different λ's
- input time constant. The circuit has 1 time constant and the source a widely different one

source 1MHz i.e. $T = 10^{-6}$s

circuit $\tau = 1$s



- simulation interval e.g. [0, 5s].

For this reason

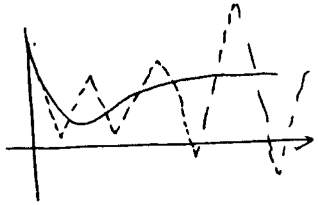
- Require variable time steps for numerical integration
- Integration methods should have large regions of stability

  FE cannot be used since it requires $h\lambda \leq 2$

  i.e. $\lambda = 10^6 \Rightarrow h \leq 2 \times 10^{-6}$

  $\Rightarrow$ cannot use large time steps.
- Integration method should be stable for variable timesteps. Our stability analysis was for uniform h.

Remarks



- If FE was used with variable h, 5 steps of 10^{-6} during the initial transient and then 5 steps of 1 s, the result is growing oscillations. We cannot use $h > 2 \times 10^{-6}$!
- In general, explicit methods cannot be used to solve stiff problems

Construction of a Stiffly Stable Method

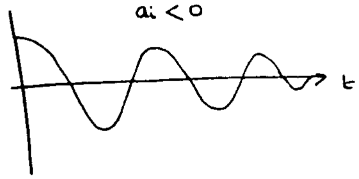
Test Problem $\dot{x}_i = \lambda_i x_i$, $i = 1, 2, \ldots n$, $\lambda_i = a_i + jb_i$

Exact solution $x_i(t) = k_i e^{a_i t} e^{jb_i t}$

$\sigma = \lambda h$

Since complex numbers appear in conjugate pairs

$\Rightarrow x_i(t) = c_i e^{a_i t} \cos b_i t$

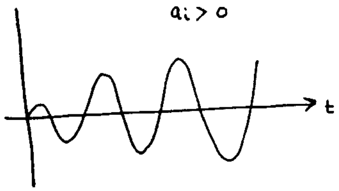


Observations

- For accuracy want at least 8 points per cycle

  $\Rightarrow h \leq \frac{1}{8}\left(\frac{2\pi}{b_i}\right)$

  or $hb_i \leq \frac{\pi}{4} \Rightarrow Im(\sigma) \leq \frac{\pi}{4}$

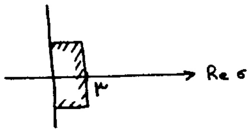


- Want a region of absolute stability which gives a stable algorithm for the initial transient

  i.e. $a_i h > 0$ $a_i h < \mu$

  $\sigma = \lambda h \Rightarrow Re(\sigma) = a_i h$

  require $0 \leq Re(\sigma) \leq \mu$

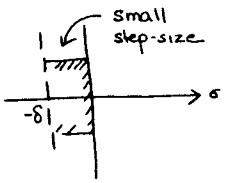


- Require a small step size to capture the fast transient

  Suppose the fast component is given by

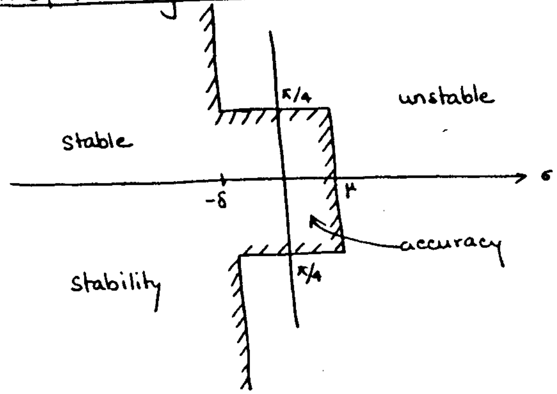
  $\lambda_{fast} = a_{fast} + j b_{fast}$

  Take $\delta > \frac{5}{a_{fast}}$, $a_{fast} < 0$



- Want larger steps for $t > |a_i|$ for all λ.

Putting all of this together



Remarks

- There is a region for $Re(\sigma) < -\delta$ that is absolutely stable
- For $0 < Im(\sigma) < \pi/4$ the region is of absolute stability and the algorithm is accurate
- For $0 < Re(\sigma) < \mu$ the region is stable and the algorithm is accurate

Can we synthesize such a method?

Recall
$$\underbrace{(z^p + \alpha_1 z^{p-1} + \cdots + \alpha_p)}_{P_N(z)} + \sigma(\underbrace{\beta_0 z^p + \beta_1 z^{p-1} + \cdots}_{P_D(z)} + \beta_p) = 0$$

$$\sigma = -\frac{P_N(z)}{P_D(z)}$$

h large $\Rightarrow \sigma \to -\infty$

$\Rightarrow P_D(z) \to 0$

$\Rightarrow \sum_{i=0}^{p} \beta_i z^{p-i} = 0$

For $\sigma \to -\infty$ the char polynomial is $\beta_0 z^p + \beta_1 z^{p-1} + \cdots + \beta_p = 0$

What if we set $\beta_1 = \beta_2 = \cdots = \beta_p = 0$ then the characteristic polynomial is

$$\lim_{r \to \infty} \beta_0 z^p = 0$$

$\therefore$ the roots are $z = 0$ with multiplicity p.

Note

There are p roots and all of them are at the origin
$\Rightarrow$ stability region includes $-\infty$!

This is the basic idea behind Gear's Methods

Gear's Methods (Backward Differentiation Formula (BDF))

order of accuracy $k = p$

$\beta_0 \neq 0, \ \beta_1 = \beta_2 \cdots = \beta_k = 0$

Kth order BDF or Gear: $\displaystyle\sum_{i=0}^{k} \alpha_i x_{n-i} + h\beta_0 \dot{x}_n = 0$

$$\dot{x}_n = -\frac{1}{h\beta_0} \sum_{i=0}^{k} \alpha_i x_{n-i} \quad \text{hence backward differentiation}$$

Note $k=1 \Rightarrow$ BE. $x_n - x_{n-1} - h\dot{x}_n = 0$

Determining the coefficients

There are coeff: $\alpha_1, \alpha_2, \cdots \alpha_k$; $\alpha_0 = 1$
β_0

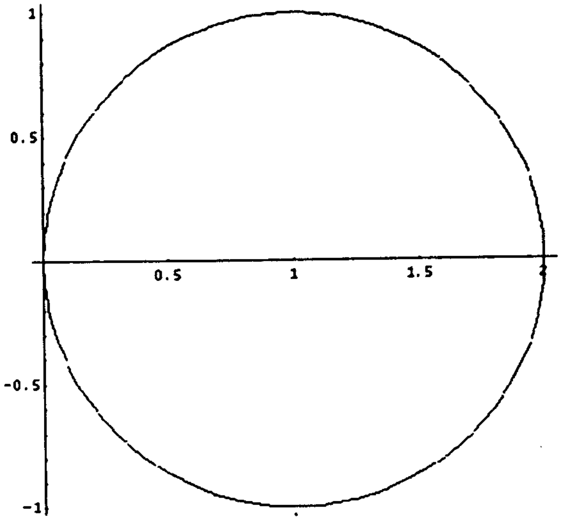
i.e. $k+1$ coeff.

For a kth order method we have $k+1$ exactness constraints

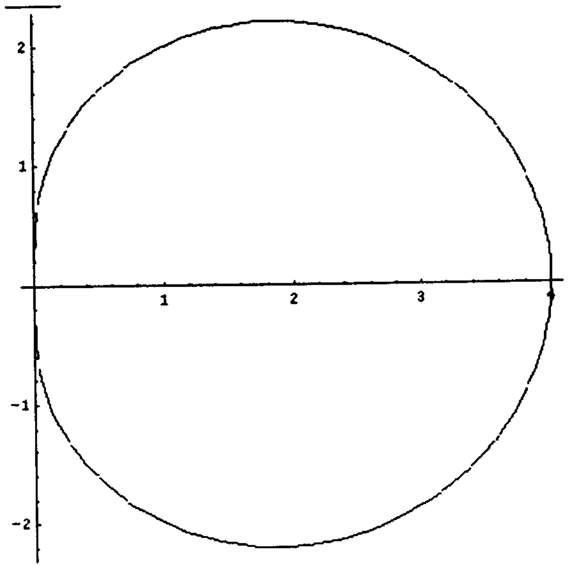
$\Rightarrow$ solve for $(k+1)$ coeff from $(k+1)$ eqns

Region of Stability
for BDF
(Region outside
the closed curve)

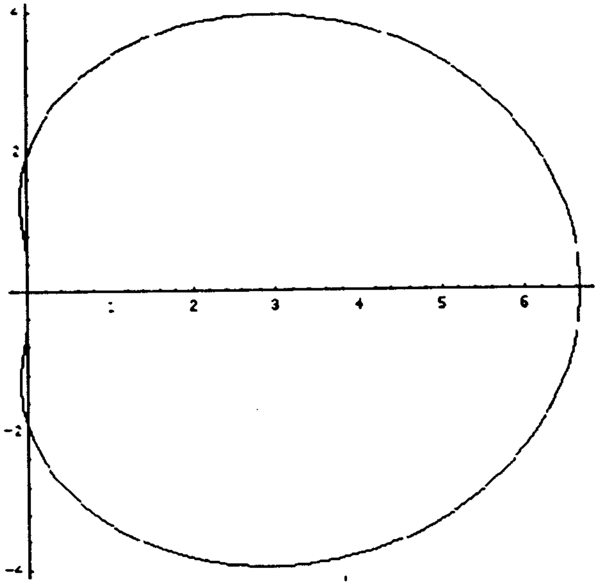
k = 1



k = 2



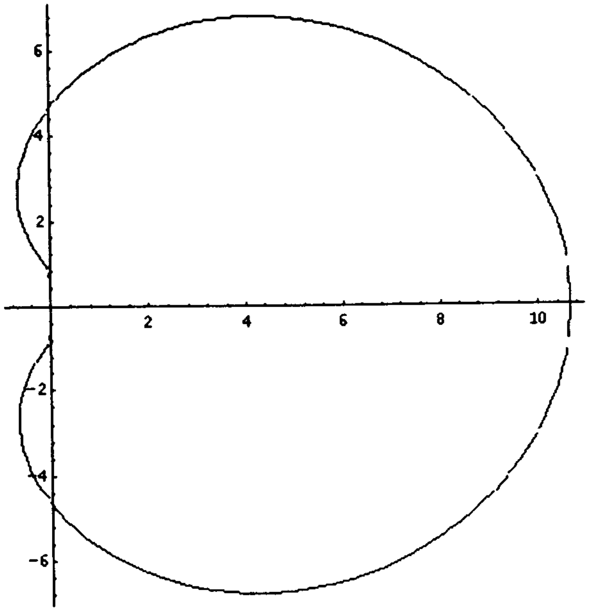
From ASV



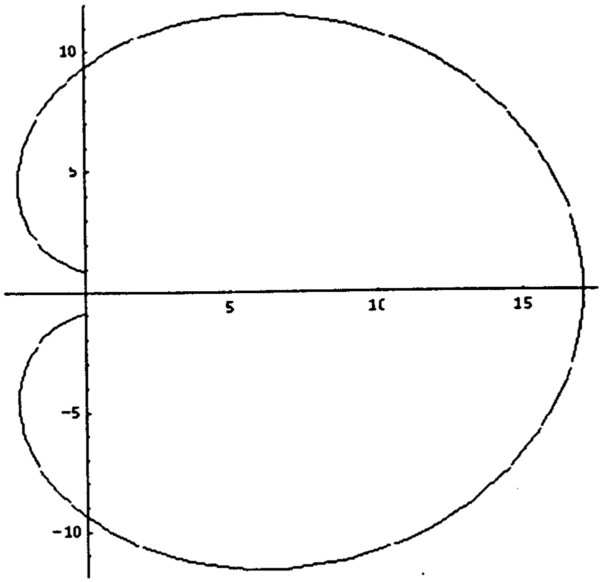
Region of Stability for BDF (Region outside the closed curve)

k = 3

k = 4



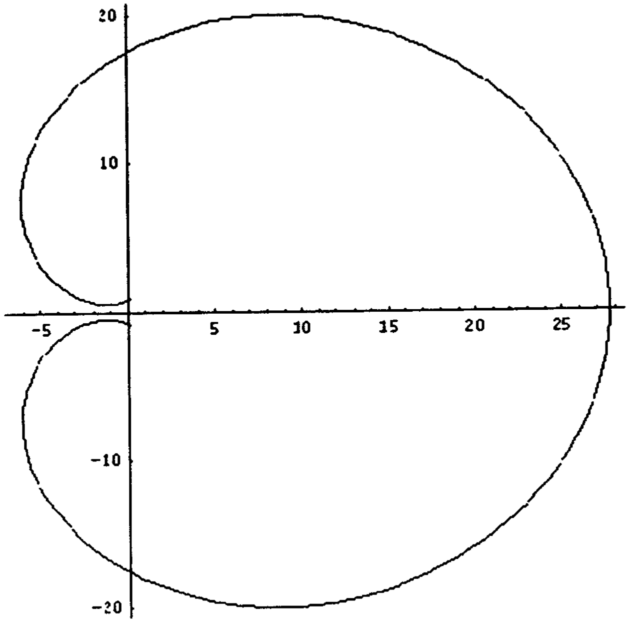
From ASV



Region of Stability for BDF (Region outside the closed curve)

k = 5

k = 6



From ASV

Variable time steps

What to do when timesteps are variable, ie, a fixed h is not used. In this case coefficients depend on past time steps.

Remark

We will always write the LMS in the following form

$$\sum_{i=0}^{p} \alpha_i x_{n-i} + h_n \sum_{i=0}^{p} \beta_i \dot{x}_{n-i} = 0$$

where $\sum_{i=0}^{p} \beta_i = -1$. This form results in an accurate expression for LE.

Example 2nd order BDF (Gear)

$$\sum_{i=0}^{2} \alpha_i x_{n-i} + h_n \underset{\overset{\downarrow}{-1}}{(\beta_0)} \dot{x}_n = 0$$

$$\Rightarrow \quad \alpha_0 x_n + \alpha_1 x_{n-1} + \alpha_2 x_{n-2} - h_n \dot{x}_n = 0$$

For this method to be order k it must be exact for polynomials of degree k = 0, 1 and 2.
Take

$$P(t) = (t-t_n)^k \qquad k = 0, 1, 2$$

$$\therefore \text{LE} = \alpha_0 x(t_n) + \alpha_1 x(t_{n-1}) + \alpha_2 x(t_{n-2}) - h_n \dot{x}(t_n)$$

For 2nd-order method we have

$$x(t) = (t-t_n)^k, \quad k = 1, 2; \qquad x(t) = 1 \quad k = 0$$

$$\dot{x}(t) = k(t-t_n)^{k-1}, \quad k = 1, 2; \qquad \dot{x}(t) = 0$$

$$\dot{x}(t_n) = 1 \quad \text{when } k = 1;$$

$$\therefore \quad 0 = \alpha_0 (t_n - t_n)^k + \alpha_1 (t_{n-1} - t_n)^k + \alpha_2 (t_{n-2} - t_n)^k - h_n \beta_0 k (t-t_n)^{k-1}$$

for $k > 1$

$\underline{k=0}$: $x(t) = 1$

$$0 = \alpha_0 + \alpha_1 + \alpha_2$$

$\underline{k=1}$: $x(t) = (t - t_n)$, $\dot{x} = 1$

$$0 = \alpha_0 (0) + \alpha_1 (-h_n) + \alpha_2 (-h_n - h_{n-1}) - h_n$$

$\underline{k=2}$: $x(t) = (t-t_n)^2$, $\dot{x}(t) = 2(t-t_n)$

$$0 = \alpha_0 . 0 + \alpha_1 (-h_n)^2 + \alpha_2 (-h_n - h_{n-1})^2 - h_n . 0$$

The equations are

$$\alpha_0 + \alpha_1 + \alpha_2 = 0 \qquad \text{——} ①$$

$$h_n \alpha_1 + (h_n + h_{n-1}) \alpha_2 = -h_n \qquad \text{——} ②$$

$$h_n^2 \alpha_1 + (h_n + h_{n-1})^2 \alpha_2 = 0 \qquad \text{——} ③$$

$$② \Rightarrow \quad \alpha_1 = -1 - \left(\frac{h_n + h_{n-1}}{h_n} \right) \alpha_2$$

Substitute in ③

$$\Rightarrow \quad -h_n^2 - h_n (h_n + h_{n-1}) \alpha_2 + (h_n + h_{n-1})^2 \alpha_2 = 0$$

$$\therefore \alpha_2 = \frac{h_n^2}{h_{n-1}(h_n + h_{n-1})}$$

$$\Rightarrow \quad \alpha_1 = -1 - \left(\frac{h_n + \cancel{h_{n-1}}}{\cancel{h_n}} \right) \frac{h_n^{\cancel{2}}}{h_{n-1}(h_n + \cancel{h_{n-1}})}$$

$$= -1 - \frac{h_n}{h_{n-1}} = - \left(\frac{h_n + h_{n-1}}{h_{n-1}} \right)$$

$$\alpha_0 = -\alpha_1 - \alpha_2 = -\left(-1 - \frac{h_n + h_{n-1}}{h_n} \alpha_2\right) - \alpha_2$$

$$= 1 + \left(\frac{\cancel{h_n} + h_{n-1} - \cancel{h_n}}{h_n} \right) \alpha_2$$

$$= 1 + \frac{\cancel{h_{n-1}}}{\cancel{h_n}} \cdot \frac{h_n^{\cancel{2}}}{\cancel{h_{n-1}} (h_n + h_{n-1})} = 1 + \frac{h_n}{h_n + h_{n-1}}$$

$$= h_n \left(\frac{1}{h_n} + \frac{1}{h_n + h_{n-1}} \right)$$

In General k-th BDF $(1 \le \text{order} \le k)$

$$\sum_{i=0}^{k} \alpha_0 x_{n-i} - h_n \dot{x}_n = 0$$

$$\alpha_0 = h_n \sum_{i=1}^{k} \frac{1}{(t_n - t_{n-i})}$$

$$1 \le j \le k: \quad \alpha_j = \frac{-h_n}{(t_n - t_{n-j})} \prod_{\substack{i=1 \\ i \ne j}}^{k} \frac{(t_n - t_{n-i})}{(t_{n-j} - t_{n-i})}$$

Recall $\quad LE = C_{k+1} h^{k+1} x^{(k+1)}(t_n)$

For kth-order BDF

$$C_{k+1} = \frac{1}{(k+1)!} \frac{1}{h_n^k} \prod_{i=1}^{k} (t_n - t_{n-i})$$

Example 2nd-order BDF

$$\alpha_0 = h_n \left(\frac{1}{h_n} + \frac{1}{h_n + h_{n-1}} \right)$$

$$\alpha_1 = - \frac{h_n + h_{n-1}}{h_{n-1}}$$

$$\alpha_2 = - \frac{h_n}{(h_n + h_{n-1})} \frac{h_n}{-h_{n-1}} = \frac{h_n^2}{h_{n-1}(h_n + h_{n-1})}$$

and $\quad C_{k+1} = \frac{1}{6} \frac{1}{h_n^2} \cdot (h_n)(h_n + h_{n-1}) = \frac{1}{6} \left(\frac{h_n + h_{n-1}}{h_n} \right)$

Remark

For uniform step size $\quad \alpha_0 = 3/2, \ \alpha_1 = -2, \ \alpha_2 = 1/2$

$$C_{k+1} = \frac{1}{6} \cdot 2 = 1/3$$

$$\Rightarrow LE = \frac{h^3}{3} \frac{d^3 x(t_n)}{dt^3}$$

Let us consider TR

Uniform h: $\quad x_n - x_{n-1} - \frac{h}{2}(\dot{x}_n + \dot{x}_{n-1}) = 0 \qquad \sum \beta_i = -1$

Non uniform h:

$$\alpha_0 x_n + \alpha_1 x_{n-1} + h_n \beta_0 \dot{x}_n + h_n \beta_1 \dot{x}_{n-1} = 0$$

Also require $\quad \beta_0 + \beta_1 = -1 \quad \Rightarrow \beta_1 = -1 - \beta_0$

$$\therefore \quad \alpha_0 x_n + \alpha_1 x_{n-1} + h_n \beta_0 \dot{x}_n - h_n (1 + \beta_0) \dot{x}_{n-1} = 0$$

2nd order method ⇒ exact for $P(t) = (t - t_n)^k \qquad k = 0, 1, 2$

$k=0 \quad x(t) = 1; \quad \dot{x}(t) = 0$

$\qquad \alpha_0 + \alpha_1 = 0$

$k=1 \quad x(t) = (t - t_n); \quad \dot{x}(t) = 1$

$\qquad \alpha_0 \cdot 0 + (-h_n)\alpha_1 + h_n \cancel{\beta_0} - h_n(1 + \cancel{\beta_0}) = 0$

$k=2 \quad x(t) = (t - t_n)^2; \quad \dot{x}(t) = 2(t - t_n)$

$\qquad (-h_n)^2 \alpha_1 - h_n (1 + \beta_0) 2(-h_n) = 0$

Thus: $\quad \alpha_0 + \alpha_1 = 0 \quad \Rightarrow \alpha_0 = -\alpha_1 = 1$

$\qquad \alpha_1 = -1$

$\qquad \cancel{h_n^2} \alpha_1 + 2\cancel{h_n^2}(1 + \beta_0) = 0 \Rightarrow \beta_0 = -1/2 \Rightarrow \beta_1 = -1/2$

$\therefore$ the LMS is $\quad x_n - x_{n-1} - \frac{h_n}{2} \dot{x}_n - \frac{h_n}{2} \dot{x}_{n-1} = 0$

Note:

This is the same as TR with a uniform timestep. The integration formula doesn't change for a nonuniform h.

Remark

$$LE = -\frac{h_n^3}{12} \frac{d^3 x(t_n)}{dt^3}$$

Timestep Control

How do we determine the stepsize?

Goal: Minimize computing time given an error bound.

⇒ minimize # of timepoints

⇒ choose as large a timestep as possible consistent with the error bound.

Assume E_n, a bound on the absolute value of LE_n is given for a kth-order method

i.e. $|LE_n| \le E_n$

Recall $LE_n = c_{k+1} h_n^{k+1} x^{(k+1)}(t_n)$

$$\therefore \quad |LE_n| = |c_{k+1} h_n^{k+1} x^{(k+1)}(t_n)| = h_n^{k+1} |c_{k+1} x^{(k+1)}(t_n)| \le E_n$$

Solve for h_n

$$\Rightarrow \quad h_n \le \left[\frac{E_n}{|c_{k+1} x^{(k+1)}(t_n)|}\right]^{1/(k+1)}$$

E_n is known, c_{k+1} can be computed. What about $x^{(k+1)}(t_n)$?

In SPICE divided differences are used

i.e.

$$DD_1 = \frac{x_n - x_{n-1}}{h_n} \cong \dot{x}_n$$

$$DD_2 = \frac{DD_1(t_n) - DD_1(t_{n-1})}{h_n + h_{n-1}} \cong \frac{x_n^{(2)}}{2!}$$

$$\vdots$$

$$DD_{k+1} = \frac{DD_k(t_n) - DD_k(t_{n-1})}{\sum_{i=0}^{k} h_{n-i}} \cong \frac{x_n^{(k+1)}}{(k+1)!}$$

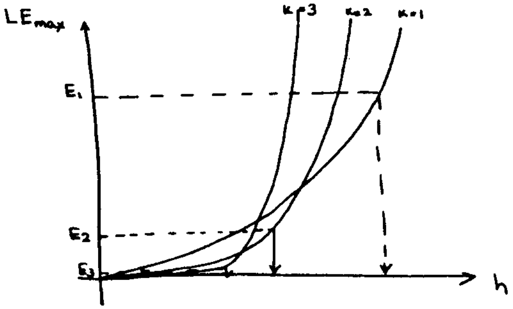
$$\Rightarrow \quad h_n \le \left[\frac{E_n}{|c_{k+1}|\,(k+1)!\,|DD_{k+1}|}\right]^{1/(k+1)}$$

Order Control

If we have many methods with different order k, e.g. BDF $1 \le k \le 6$, we choose a method which gives the largest step size.

i.e. we compute h_n for each of the methods and select $\hat{h}_n$ and k such that

$$\hat{h}_n = \max_{h_n, k} h_n$$



Remark

SPICE uses TR by default. Gear's or BDF can be used by specifying METHOD = GEAR, and MAXORD = n $(2 \le n \le 6)$

Heuristic rules for stepsize control

- Do not change stepsize too often.
- Change order only if improvement is worthwhile i.e. at least 2h
- Attempt change of stepsize and order only if
  - LE < E for k+1 steps after last change
  - error is large

In SPICE

$$E_n = \underset{\uparrow}{\epsilon_A} + \epsilon_R \, \text{MAX}\left(|x_n|, |x_{n-1}|\right)$$

ABSTOL (currents)
VNTOL (voltages)
CHGTOL (charges)

Remark

In SPICE, the LE estimate is divided by TRTOL. Default value is 7. This relaxes the time step control. Equivalent to saying allowed error is TRTOL $*$ E_n.

Timestep control in SPICE

.tran TSTEP TSTOP TSTART TMAX

$$\text{DELMAX} = \text{MIN}\left(\frac{\text{TSTOP}}{50}, \text{TMAX}, \underset{\uparrow}{\text{TSTEP}}\right)$$

used only when there are no energy storage elements

$$\text{DELMIN} = 10^{-9} * \text{DELMAX}$$

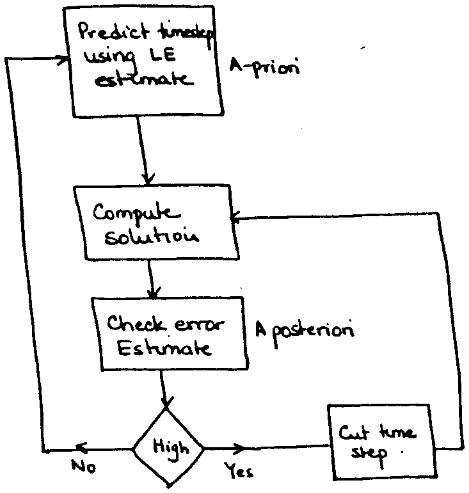
A new time step is determined from the LE estimate say h. The new step that is used is given by

$$h_n = \text{MIN}\left(2*h_{n-1}, h_n, \text{DELMAX}\right)$$

Remark

- The LE estimate in SPICE uses coefficients for uniform time steps. This is OK for TR but incorrect for GEAR's method!

Timestep control Strategy

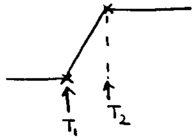


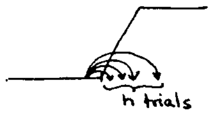
Remark

- The time step is cut by whatever the posteriori error check suggest or by a factor of 8 if the nonlinear equations do not converge in 10 iteration.

Sharp input transitions

Time points T_1 and T_2 are called breakpoints. If these are not specified apriori then the timestep algorithm will "hunt" for a suitable time step.





So in SPICE the breakpoints are specified and a solution timepoint is forced at each break point.

In addition, a first-order integration is used after the break point since past timepoint data is no longer valio in predicting a solution.

Iteration count time step control

Two iteration limits in SPICE 2

ITL3 (default 4)

ITL4 (default 10)

If Newton's method does not converge in ITL4 iteration cut time step by a factor of 8. If it converges in less than ITL3 iterations increase timestep by a factor of 2.

Remarks

- LE is not computed and controlled. This could be a problem in some circuits. Classic example linear circuits: NR converges in 2 iterations ⇒ double timestep at each timepoint!
- Does reduce work in NR iterations because a timestep is selected based on the NR effort.
- SPICE2:

  LVLTIM = 2, LTE timestep control (default)

  LVLTIM = 1, iteration count control

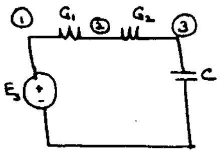
Other options for transient

ITL5	total iteration limit for transient analysis
LIMPTS	total number of points that can be specified for transient analysis

Application of LMS Algorithms to Circuits

Circuit equations are not ordinary Differential Eqns (ODEs) i.e.

$$\dot{x} = f(x)$$



$$G_1(v_1 - v_2) \quad + I_E \quad = 0 \qquad (A)$$

$$G_1(V_2 - V_1) \quad + G_2(V_2 - V_3) = 0 \qquad (A)$$

$$G_2(V_3 - V_2) + C\frac{dV_3}{dt} \quad = 0 \qquad (D)$$

$$V_1 \quad = E_s(t) \qquad (A)$$

In general, $F(x, \dot{x}, t) = 0$ are called DAEs

In the context of circuits $\dot{x}$ are due to capacitor and inductors

Consider linear/nonlinear capacitors

$$C\frac{dv}{dt} - i = 0 \qquad \text{or} \qquad \frac{dq(v)}{dt} - i = 0$$

The time derivatives are expressed using the LMS formula

$$\sum_{i=0}^{p} \alpha_i x_{n-i} + h_n \sum_{i=0}^{p} \beta_i \dot{x}_{n-i} = 0$$

$$\Rightarrow \quad \dot{x}_n = -\frac{1}{h_n} \sum_{i=0}^{p} \alpha_i x_{n-i} - \sum_{i=1}^{p} \beta_i \dot{x}_{n-i}$$

$\therefore \quad F(x, \dot{x}, t) = 0$ becomes.

at t_n

$$F(x_n, \dot{x}_n(x_n), t_n) = 0$$

This is a nonlinear function of the unknowns x_n. Nonlinear equations solved by Newton Raphson

Remark

- Resistive elements are not affected.

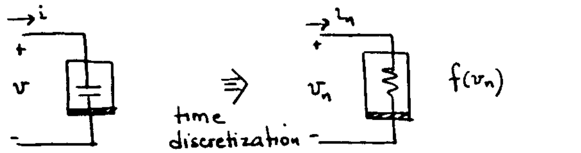
Example (Nonlinear C)

$$q(v) = q_0 \sqrt{v}$$

$$\therefore\ i = \frac{dq}{dt} = \frac{dq}{dv}\cdot\frac{dv}{dt} = C(v)\,\dot{v}$$

Using BE we have $i_n = \frac{q_0}{2\sqrt{v_n}}\left(\frac{v_n - v_{n-1}}{h_n}\right) = f(v_n)$

i.e. it is a nonlinear voltage controlled resistor



Application of Newton's method gives

$$i_n^{k+1} = i_n^k + g(v_n^k)\,(v_n^{k+1} - v_n^k) \qquad \text{where } g(v_n^k) = \left.\frac{\partial f(v_n)}{\partial v_n}\right|_{v_n^k}$$

$$= \frac{q_0}{2\sqrt{v_n^k}}\left(\frac{v_n^k - v_{n-1}}{h_n}\right) + \left[\frac{-q_0}{4(\sqrt{v_n^k})^3}\left(\frac{v_n^k - v_{n-1}}{h_n}\right) + \frac{q_0}{2\sqrt{v_n}}\left(\frac{1}{h_n}\right)\right](v_n^{k+1} - v_n^k)$$

$$= G_k v_n^{k+1} + I_k$$

where $G_k = \left[\frac{-q_0}{4(v_n^k)^{3/2}}\left(\frac{v_n^k - v_{n-1}}{h_n}\right) + \frac{q_0}{2\sqrt{v_n}}\left(\frac{1}{h_n}\right)\right]$

This looks quite clumsy! We are taking the derivative of a capacitance i.e. $\frac{dc}{dv}$.

An elegant solution is in terms of the charge formulation

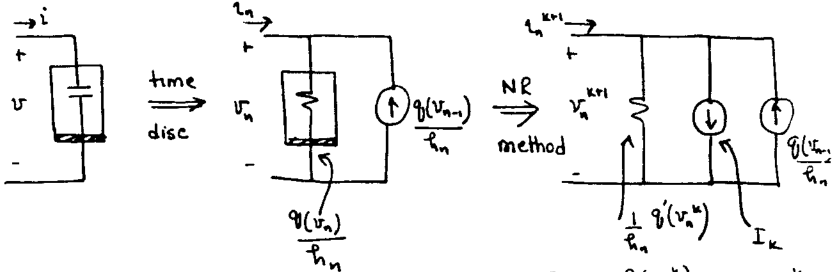
$$i_n = \frac{q_n - q_{n-1}}{h_n} \qquad \text{if BE is used.}$$

$$= \frac{q(v_n) - q(v_{n-1})}{h_n}$$

Apply NR method

$$i_n^{k+1} = i_n^k + \left.\frac{di_q}{dv_n}\right|_{v_n^k}(v_n^{k+1} - v_n^k)$$

$$= \frac{q(v_n^k) - q(v_{n-1})}{h_n} + \underbrace{\frac{1}{h_n}\,q'(v_n^k)}_{G_k}\,(v_n^{k+1} - v_n^k)$$



$$I_k = \frac{q(v_n^k)}{h_n} - G_k v_n^k$$

Charge Conservation

We have seen that the nonlinear charge equations can be written in two forms

① $\frac{d}{dt}\,q(v(t)) = f(v(t))$

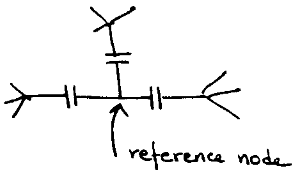
② $C(v(t))\,\frac{d}{dt}v(t) = f(v(t))$

where $C(v) = \frac{dq}{\partial v}$. The two forms are equivalent

Consider the (n+1) nodal equations with capacitors from every node to ground

Charge conservation requires

$$\sum_{i=1}^{n+1} q_i(v(t)) = K$$



Also note that $\sum_{i=1}^{n+1} f_i(v(t)) = 0$

Let us now apply an integration method to ① & ② and observe what happens

Example FE applied to the above equations

①
$$\frac{q_n - q_{n-1}}{h_n} = f(v_{n-1}) \Rightarrow q(v_n) - q(v_{n-1}) = h_n f(v_{n-1})$$

$$\therefore \sum_{i=1}^{n+1} q_i(v_n) - \sum_{i=1}^{n+1} q_i(v_{n-1}) = h_n \sum_{i=1}^{n+1} \cancelto{0}{f_i(v_{n-1})}$$

$$\therefore \sum_{i=1}^{n+1} q_i(v_n) = \sum_{i=1}^{n+1} q_i(v_{n-1}) \overset{= 0}{} = K$$

i.e. charge is conserved.

②
$$C(v_{n-1})\,(v_n - v_{n-1}) = h_n f(v_{n-1})$$

$$q(v_n) = q(v_{n-1}) + \overbrace{\frac{\partial q}{\partial v}(v_{n-1})}^{C(v_{n-1})}(v_n - v_{n-1})$$

$$+ \frac{1}{2}\frac{\partial^2 q}{\partial v^2}(\hat{v})\,(v_n - v_{n-1})^2$$

$$= q(v_{n-1}) + \underbrace{(v_n - v_{n-1})C(v_{n-1})}_{h_n\ f(v_{n-1})} + \frac{1}{2}\frac{\partial^2 q}{\partial v^2}(\hat{v})(v_n - v_{n-1})^2$$

$$\therefore q(v_n) = q(v_{n-1}) + h_n f(v_{n-1}) + \frac{1}{2}\frac{\partial^2}{\partial v^2} q(\hat{v})\,(v_n - v_{n-1})^2$$

Thus
$$\sum_{i=1}^{n+1} q(v_n) = \sum_{i=1}^{n+1} q(v_{n-1}) + h_n \cancelto{0}{\sum_{i=1}^{n} f(v_{n-1})} + O(h_n^2)$$

$$= K + 0 + O(h_n^2)$$

$\Rightarrow$ Charge is not conserved

Remarks

- A small timestep applied to ② may not exhibit charge nonconservation visually. However, that doesn't mean charge is conserved
- Any consistent multistep method conserves charge when charge is used as the unknown.

Observations on BDF

$$\dot{x}_n = \frac{-1}{h_n}\sum_{i=0}^{k} \alpha_i x_{n-i} \qquad \text{kth order BDF}$$

We use a kth-order predictor for estimating x_n^0. The predictor order is the same as that of the integration method

$$x_n^0 = \sum_{i=1}^{k+1} \gamma_i x_{n-i} = x_n^p \quad \text{(predicted)}$$

i.e. extrapolate thru previous (k+1) points for a k degree polynomial

It can be then shown that

$$LE_n = \frac{h_n}{t_n - t_{n-k-1}}\,(\underset{\uparrow \text{corrected}}{x_n^c} - \overset{\text{predicted}}{x_n^p})$$

Remark

- LE calculation is simple and accurate. x_n^p has to be computed for prediction $\Rightarrow$ LE calculation is cheap

Example (2nd-order method)

A second-order predictor is used i.e.

$$x_n^p = \gamma_1 x_{n-1} + \gamma_2 x_{n-2} + \gamma_3 x_{n-3}$$

where γ_1, γ_2, and γ_3 are determined by exactness constraints

Remark

- It can be shown that

$$x_n - x_n^p \propto \frac{d^3 x(t_n)}{dt^3}$$

Hence the relation between LE $\propto (x_n^c - x_n^p)$.

Stability with variable time steps

BE & TR coefficients do not change when variable time steps are used. So the stability requirements remain the same.

However, for BDF the coefficients are a function of the step size, so stability cannot be easily determined.

Remark

For BDF-2 it can be shown that the method is stable with variable time steps if

$$\frac{h_n}{h_{n-1}} \leq 1.2.$$

A Note on Global Error

Consider FE $\qquad \dot{x} = f(x)$

$$x_n = x_{n-1} + h\dot{x}_{n-1} = x_{n-1} + hf(x_{n-1})$$

$$x(t_n) = x(t_{n-1}) + h\dot{x}(t_{n-1}) + \frac{h^2}{2}\ddot{x}(\tau) \qquad \tau \in [0,h]$$

The local error is $e_L(t_{n-1})$ is $O(h^2)$.

The Global error $E_n = x(t_n) - x_n$

Subtracting we have

$$E_n = E_{n-1} + h\left[f(x(t_{n-1})) - f(x_{n-1})\right] + e_L$$

$$\Rightarrow |E_n| \leq |E_{n-1}| + h\left|f(x(t_{n-1})) - f(x_{n-1})\right| + |e_L|$$

$$\leq |E_{n-1}| + hL\left|x(t_{n-1}) - x_{n-1}\right| + |e_L|$$

$$\leq (1+hL)|E_{n-1}| + |e_L|$$

$$\leq (1+hL)|E_{n-1}| + e_{LMAX} \quad ; \; |e_L| \leq e_{LMAX}$$

$$\Rightarrow |E_n| \leq (1+hL)\left[(1+hL)|E_{n-2}| + e_{LMAX}\right] + e_{LMAX}$$

$$\vdots$$

$$\leq (1+hL)^n \overset{0}{\cancel{|E_0|}} + \left[\sum_{i=0}^{n-1}(1+hL)^i\right] e_{LMAX}$$

$$\leq e_{LMAX}\frac{1-(1+hL)^n}{1-(1+hL)} = e_{LMAX}\frac{(1+hL)^n - 1}{hL}$$

$$\leq e_{LMAX}\frac{e^{nhL}-1}{hL} \leq e_{LMAX}\frac{e^{TL}-1}{hL} \qquad \text{since } nh \leq T$$

$$\leq \frac{e_{LMAX}}{h}\,\frac{e^{TL}-1}{L} = \frac{1}{2}\max_{t\in[0,T]}|\ddot{x}(t)|\, h\,\frac{e^{TL}-1}{L}$$

Observations

- $\lim_{h\to 0} |E_n| = 0$ $\qquad$ i.e. convergent
- Global error is one order lower in h than local error

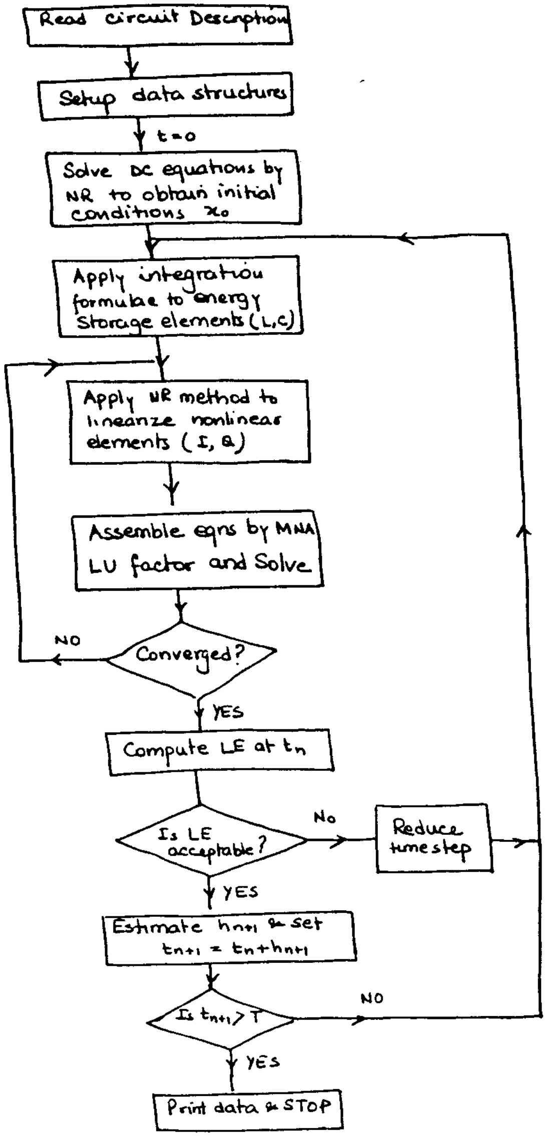
Inside a Circuit Simulator - SPICE 3

SPICE 2 is outdated and several problems
- 20K lines of FORTRAN
- 10 years of changes (1985)
- basic algorithms, devices, data structures distributed throughout the code
- Difficult to add new models, analyses,
- Difficult to maintain

SPICE3 is the current version of SPICE
- faster
- more robust
  - model problems/discontinuities removed
- a flexible framework for circuit simulation
  - new models and analyses can be easily added
- new and improved algorithms
  - GMIN and source stepping
  - predictor-corrector integration (BDF)
- new device models
  - GaAs MESFET, BSIM1, BSIM3
  - Voltage/current controlled switches
  - Arbitrary controlled sources. Useful for macromodeling
  - uniform distributed R-C lines
- clean simulator interface
  - clearly defined functions independent of front end
  - table driven

In summary, modular and extensible !

Flow chart for a circuit simulator (transient analysis)


Read circuit Description
Setup data structures
t=0
Solve DC equations by NR to obtain initial conditions x0
Apply integration formulae to energy storage elements (L,C)
Apply NR method to linearize nonlinear elements (I, Q)
Assemble eqns by MNA
LU factor and Solve
Converged?
NO
YES
Compute LE at tn
Is LE acceptable?
No
Reduce timestep
YES
Estimate hn+1 & set tn+1 = tn+hn+1
Is tn+1 > T
NO
YES
Print data & STOP

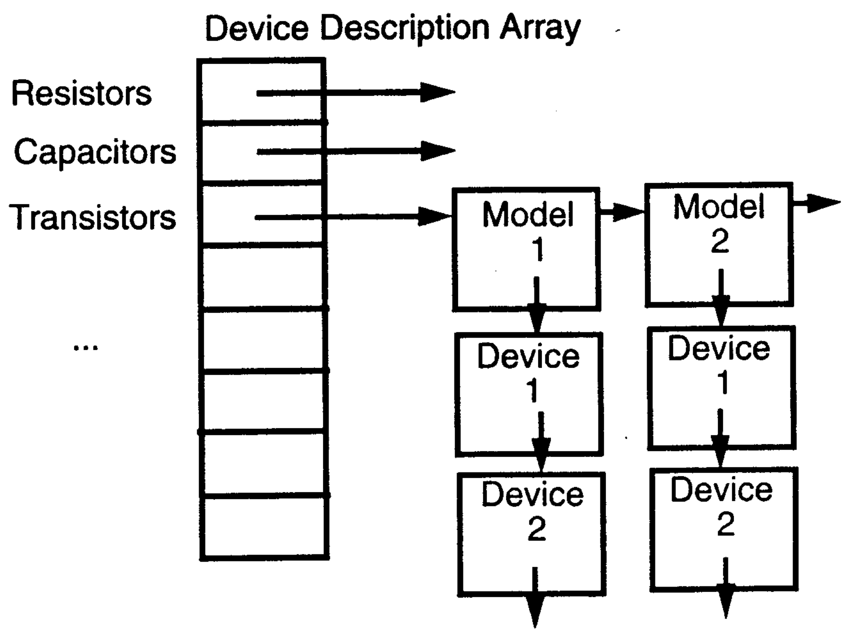
In summary the required pieces are

- Input processor
- Data Structures
- Theoretical foundations & formulation
  - linear equation solution
  - solution of nonlinear eqns : Convergence
  - numerical solution of differential eqns
    - accuracy
    - stability
- Simulation engine
  - numerical algorithms
  - models for devices
  - control loops for analyses
  - analyses
    - DC operating point analysis
    - DC transfer curves
    - Transient analysis
    - small-signal ac analysis
    - Fourier analysis
    - Noise analysis.
- Output processor
  - save solution
  - provide postprocessing capabilities
  - plotting

SPICE3

- NR for nonlinear equation solution
- Sparse 1.3 for linear eqn. solution
- integration methods TR & BDF
- Postprocessor - Nutmeg.

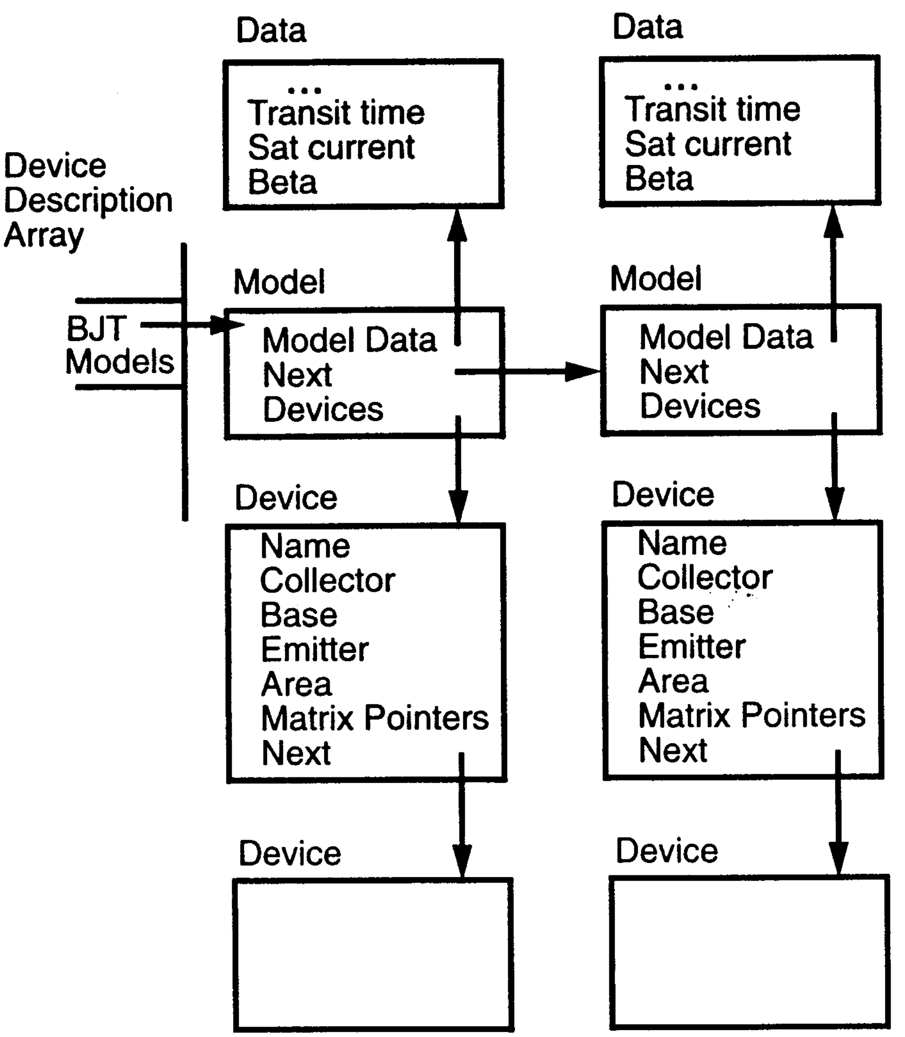
Circuit Data Structures (SPICE)



- can store generic device info in model rather than in device
- can preprocess model and then the devices which depend on that model
- can easily skip devices

From ASY

Device Data Structures (SPICE)



Device Description Array

BJT Models

Data

...
Transit time
Sat current
Beta

Model

Model Data
Next
Devices

Device

Name
Collector
Base
Emitter
Area
Matrix Pointers
Next

Device

Data

...
Transit time
Sat current
Beta

Model

Model Data
Next
Devices

Device

Name
Collector
Base
Emitter
Area
Matrix Pointers
Next

Device

From ASV

```
/* information used to describe a single instance */

typedef struct sRESinstance {
    struct sRESmodel *RESmodPtr;    /* backpointer to model */
    struct sRESinstance *RESnextInstance;   /* pointer to next instance of
                                             * current model*/

    IFuid RESname;  /* pointer to character string naming this instance */
    int RESstate; /* not used */
    int RESposNode; /* number of positive node of resistor */
    int RESnegNode; /* number of negative node of resistor */

    double REStemp;     /* temperature at which this resistor operates */
    double RESconduct;  /* conductance at current analysis temperature */
    double RESresist;   /* resistance at temperature Tnom */
    double RESwidth;    /* width of the resistor */
    double RESlength;   /* length of the resistor */
    double *RESposPosptr;    /* pointer to sparse matrix diagonal at
                              * (positive,positive) */
    double *RESnegNegptr;    /* pointer to sparse matrix diagonal at
                              * (negative,negative) */
    double *RESposNegptr;    /* pointer to sparse matrix offdiagonal at
                              * (positive,negative) */
    double *RESnegPosptr;    /* pointer to sparse matrix offdiagonal at
                              * (negative,positive) */
    unsigned RESresGiven : 1;   /* flag to indicate resistance was specified */
    unsigned RESwidthGiven  : 1;    /* flag to indicate width given */
    unsigned RESlengthGiven : 1;    /* flag to indicate length given */
    unsigned REStempGiven   : 1;    /* indicates temperature specified */
    int    RESsenParmNo;          /* parameter # for sensitivity use;
                 set equal to  0 if not a design parameter*/
#ifndef NONOISE
    double RESnVar[NSTATVARS];
#else /* NONOISE */
        double *RESnVar;
#endif /* NONOISE */

} RESinstance ;


/* per model data */

typedef struct sRESmodel {       /* model structure for a resistor */
    int RESmodType; /* type index of this device type */
    struct sRESmodel *RESnextModel; /* pointer to next possible model in
                                     * linked list */
    RESinstance * RESinstances; /* pointer to list of instances that have this
                                 * model */
    IFuid RESmodName;       /* pointer to character string naming this model */

    double REStnom;         /* temperature at which resistance measured */
    double REStempCoeff1;   /* first temperature coefficient of resistors */
    double REStempCoeff2;   /* second temperature coefficient of resistors */
    double RESsheetRes;     /* sheet resistance of devices in ohms/square */
    double RESdefWidth;     /* default width of a resistor */
    double RESnarrow;       /* amount by which device is narrower than drawn */
    unsigned REStnomGiven: 1;   /* flag to indicate nominal temp. was given */
    unsigned REStc1Given : 1;   /* flag to indicate tc1 was specified */
    unsigned REStc2Given : 1;   /* flag to indicate tc2 was specified */
    unsigned RESsheetResGiven   :1; /* flag to indicate sheet resistance given*/
    unsigned RESdefWidthGiven   :1; /* flag to indicate default width given */
    unsigned RESnarrowGiven     :1; /* flag to indicate narrow effect given */
} RESmodel;
```

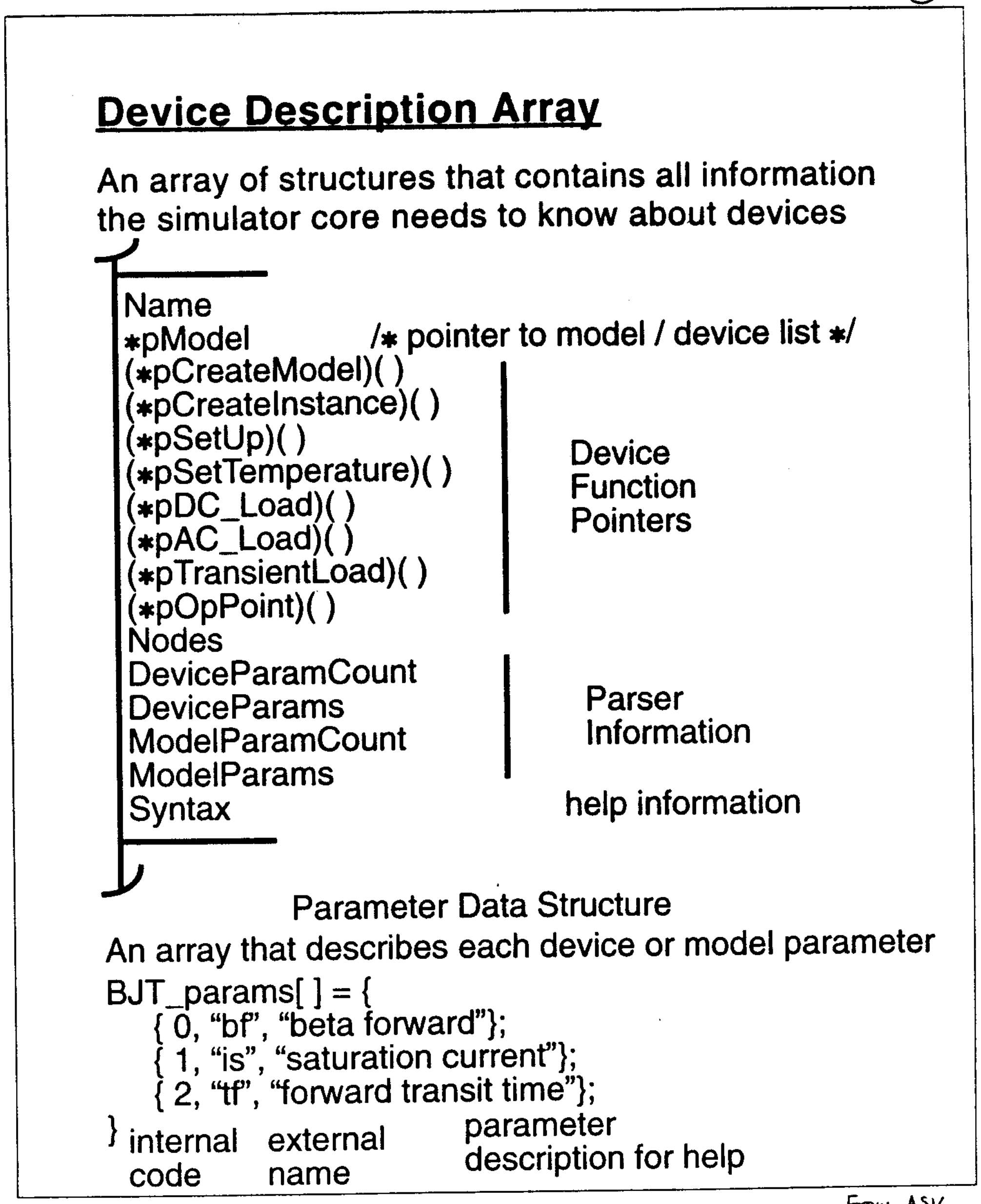
```
/**********
Copyright 1990 Regents of the University of California.  All rights reserved.
Author: 1985 Thomas L. Quarles
**********/
/*
 */

#include "spice.h"
#include <stdio.h>
#include "util.h"
#include "smpdefs.h"
#include "resdefs.h"
#include "sperror.h"
#include "suffix.h"


/* ARGSUSED */
int
RESsetup(matrix,inModel,ckt,state)
    register SMPmatrix *matrix;
    GENmodel *inModel;
    CKTcircuit*ckt;
    int *state;
        /* load the resistor structure with those pointers needed later
         * for fast matrix loading
         */
{
    register RESmodel *model = (RESmodel *)inModel;
    register RESinstance *here;

    /*  loop through all the resistor models */
    for( ; model != NULL; model = model->RESnextModel ) {

        /* loop through all the instances of the model */
        for (here = model->RESinstances; here != NULL ;
                here=here->RESnextInstance) {

/* macro to make elements with built in test for out of memory */
#define TSTALLOC(ptr,first,second) \
if((here->ptr = SMPmakeElt(matrix,here->first,here->second))==(double *)NULL){\
    return(E_NOMEM);\
}

            TSTALLOC(RESposPosptr, RESposNode, RESposNode);
            TSTALLOC(RESnegNegptr, RESnegNode, RESnegNode);
            TSTALLOC(RESposNegptr, RESposNode, RESnegNode);
            TSTALLOC(RESnegPosptr, RESnegNode, RESposNode);
        }
    }
    return(OK);
}
```

```
/**********
Copyright 1990 Regents of the University of California.  All rights reserved.
Author: 1985 Thomas L. Quarles
**********/
/*
 */

#include "spice.h"
#include <stdio.h>
#include "cktdefs.h"
#include "resdefs.h"
#include "sperror.h"
#include "suffix.h"


/*ARGSUSED*/
int
RESload(inModel,ckt)
    GENmodel *inModel;
    CKTcircuit *ckt;
        /* actually load the current resistance value into the
         * sparse matrix previously provided
         */
{
    register RESmodel *model = (RESmodel *)inModel;
    register RESinstance *here;

    /*  loop through all the resistor models */
    for( ; model != NULL; model = model->RESnextModel ) {

        /* loop through all the instances of the model */
        for (here = model->RESinstances; here != NULL ;
                here=here->RESnextInstance) {

            *(here->RESposPosptr) += here->RESconduct;
            *(here->RESnegNegptr) += here->RESconduct;
            *(here->RESposNegptr) -= here->RESconduct;
            *(here->RESnegPosptr) -= here->RESconduct;
        }
    }
    return(OK);
}
```

```
/* information needed for each instance */

typedef struct sVSRCinstance {
    struct sVSRCmodel *VSRCmodPtr;  /* backpointer to model */
    struct sVSRCinstance *VSRCnextInstance;  /* pointer to next instance of
                                              *current model */
    IFuid VSRCname; /* pointer to character string naming this instance */
    int VSRCstate;      /* not used */

    int VSRCposNode;    /* number of positive node of resistor */
    int VSRCnegNode;    /* number of negative node of resistor */

    int VSRCbranch; /* equation number of branch equation added for source */

    int VSRCfunctionType;   /* code number of function type for source */
    int VSRCfunctionOrder;  /* order of the function for the source */
    double *VSRCcoeffs; /* pointer to array of coefficients */

    double VSRCdcValue; /* DC and TRANSIENT value of source */

    double VSRCacPhase; /* AC phase angle */
    double VSRCacMag; /* AC magnitude */

    double VSRCacReal; /* AC real component */
    double VSRCacImag; /* AC imaginary component */

    double VSRCdF1mag; /* distortion f1 magnitude */
    double VSRCdF2mag; /* distortion f2 magnitude */
    double VSRCdF1phase; /* distortion f1 phase */
    double VSRCdF2phase; /* distortion f2 phase */

    double *VSRCposIbrptr;  /* pointer to sparse matrix element at
                             * (positive node, branch equation) */
    double *VSRCnegIbrptr;  /* pointer to sparse matrix element at
                             * (negative node, branch equation) */
    double *VSRCibrPosptr;  /* pointer to sparse matrix element at
                             * (branch equation, positive node) */
    double *VSRCibrNegptr;  /* pointer to sparse matrix element at
                             * (branch equation, negative node) */
    double *VSRCibrIbrptr;  /* pointer to sparse matrix element at
                             * (branch equation, branch equation) */
    unsigned VSRCdcGiven     :1 ;   /* flag to indicate dc value given */
    unsigned VSRCacGiven     :1 ;   /* flag to indicate ac keyword given */
    unsigned VSRCacMGiven    :1 ;   /* flag to indicate ac magnitude given */
    unsigned VSRCacPGiven    :1 ;   /* flag to indicate ac phase given */
    unsigned VSRCfuncTGiven  :1 ;   /* flag to indicate function type given */
    unsigned VSRCcoeffsGiven :1 ;   /* flag to indicate function coeffs given */
    unsigned VSRCdGiven  :1 ; /* flag to indicate source is a disto input */
    unsigned VSRCdF1given :1; /* flag to indicate source is an f1 dist input */
    unsigned VSRCdF2given :1; /* flag to indicate source is an f2 dist input */
} VSRCinstance ;


/* per model data */

typedef struct sVSRCmodel {
    int VSRCmodType;    /* type index of this device type */
    struct sVSRCmodel *VSRCnextModel;    /* pointer to next possible model
                                          *in linked list */
    VSRCinstance * VSRCinstances;    /* pointer to list of instances
                                      * that have this model */
    IFuid VSRCmodName;       /* pointer to character string naming this model */
} VSRCmodel;
```

```
/**********
Copyright 1990 Regents of the University of California.  All rights reserved.
Author: 1985 Thomas L. Quarles
**********/

#include "spice.h"
#include <stdio.h>
#include "util.h"
#include "smpdefs.h"
#include "cktdefs.h"
#include "vsrcdefs.h"
#include "sperror.h"
#include "suffix.h"

/* ARGSUSED */
int
VSRCsetup(matrix,inModel,ckt,state)
    register SMPmatrix *matrix;
    GENmodel *inModel;
    register CKTcircuit *ckt;
    int *state;
        /* load the voltage source structure with those pointers needed later
         * for fast matrix loading
         */
{
    register VSRCmodel *model = (VSRCmodel *)inModel;
    register VSRCinstance *here;
    CKTnode *tmp;
    int error;

    /*  loop through all the voltage source models */
    for( ; model != NULL; model = model->VSRCnextModel ) {

        /* loop through all the instances of the model */
        for (here = model->VSRCinstances; here != NULL ;
                here=here->VSRCnextInstance) {

            if(here->VSRCbranch == 0) {
                error = CKTmkCur(ckt,&tmp,here->VSRCname,"branch");
                if(error) return(error);
                here->VSRCbranch = tmp->number;
            }

/* macro to make elements with built in test for out of memory */
#define TSTALLOC(ptr,first,second) \
if((here->ptr = SMPmakeElt(matrix,here->first,here->second))==(double *)NULL){\
    return(E_NOMEM);\
}

            TSTALLOC(VSRCposIbrptr, VSRCposNode, VSRCbranch)
            TSTALLOC(VSRCnegIbrptr, VSRCnegNode, VSRCbranch)
            TSTALLOC(VSRCibrNegptr, VSRCbranch, VSRCnegNode)
            TSTALLOC(VSRCibrPosptr, VSRCbranch, VSRCposNode)
        }
    }
    return(OK);
}
```

```
/**********
Copyright 1990 Regents of the University of California.  All rights reserved.
Author: 1985 Thomas L. Quarles
**********/

#include "spice.h"
#include <stdio.h>
#include "util.h"
#include "cktdefs.h"
#include "vsrcdefs.h"
#include "trandefs.h"
#include "sperror.h"
#include "suffix.h"

int
VSRCload(inModel,ckt)
    GENmodel *inModel;
    register CKTcircuit *ckt;
        /* actually load the current voltage value into the
         * sparse matrix previously provided
         */
{
    register VSRCmodel *model = (VSRCmodel *)inModel;
    register VSRCinstance *here;
    double time;

    /*  loop through all the voltage source models */
    for( ; model != NULL; model = model->VSRCnextModel ) {

        /* loop through all the instances of the model */
        for (here = model->VSRCinstances; here != NULL ;
                here=here->VSRCnextInstance) {

            *(here->VSRCposIbrptr) += 1.0 ;
            *(here->VSRCnegIbrptr) -= 1.0 ;
            *(here->VSRCibrPosptr) += 1.0 ;
            *(here->VSRCibrNegptr) -= 1.0 ;
            if( (ckt->CKTmode & (MODEDCOP | MODEDCTRANCURVE)) &&
                    here->VSRCdcGiven ) {
                /* grab dc value */
                *(ckt->CKTrhs + (here->VSRCbranch)) += ckt->CKTsrcFact *
                        here->VSRCdcValue;
            } else {
                if(ckt->CKTmode & (MODEDC)) {
                    time = 0;
                } else {
                    time = ckt->CKTtime;
                }
                /* use the transient functions */
                switch(here->VSRCfunctionType) {
                default: { /* no function specified:   use the DC value */
                    *(ckt->CKTrhs + (here->VSRCbranch)) += here->VSRCdcValue;
                    break;
                }

                case PULSE: {
                        .
                        .
                        .
                break;
                case SINE: {
                }
                break;
                case PWL: {
```



Device Description Array

An array of structures that contains all information the simulator core needs to know about devices

Name
pModel / pointer to model / device list */
(*pCreateModel)( )
(*pCreateInstance)( )
(*pSetUp)( )
(*pSetTemperature)( )
(*pDC_Load)( )
(*pAC_Load)( )
(*pTransientLoad)( )
(*pOpPoint)( )
Nodes
DeviceParamCount
DeviceParams
ModelParamCount
ModelParams
Syntax

Device Function Pointers

Parser Information

help information

Parameter Data Structure

An array that describes each device or model parameter

```
BJT_params[ ] = {
    { 0, "bf", "beta forward"};
    { 1, "is", "saturation current"};
    { 2, "tf", "forward transit time"};
}
```

internal code | external name | parameter description for help

From ASV

Input Phase (SPICE)

Information read

1. devices in circuit (R, C, M, ...)
2. models for devices (NMOS, PMOS)
3. analysis requests (DC, TRAN)
4. analysis parameters:
    - stop time - TSTOP
    - relative tolerance - RELTOL
    - absolute tolerance - ABSTOL
5. output requests (plot nodes)

From ASV

Set up Phase (SPICE)

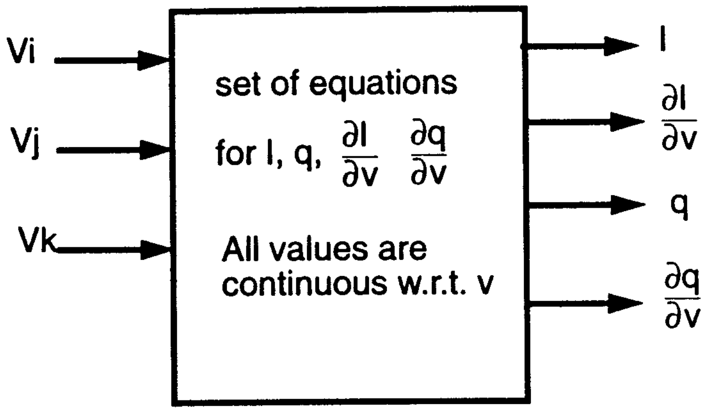
- preprocess all models
    - load default values
    - check validity of user values
    - calculate useful quantities
- preprocess all devices
    - allocate matrix entries
    - calculate useful quantities
- topology checks

After set up phase:

- circuit is topologically correct
- model and device parameters are reasonable
- matrix has been allocated
- circuit data structures allocated

From ASV

Device Evaluation Routines (SPICE)


Vi
Vj
Vk
I
$\frac{\partial I}{\partial v}$
q
$\frac{\partial q}{\partial v}$

set of equations for I, q, $\frac{\partial I}{\partial v}$ $\frac{\partial q}{\partial v}$

All values are continuous w.r.t. v

Matrix Load: template

X	X	X	
X	X	X	
			X

X
X
X

From ASV

Initial guess/conditions in SPICE

Two commands

- . NODESET
- . IC

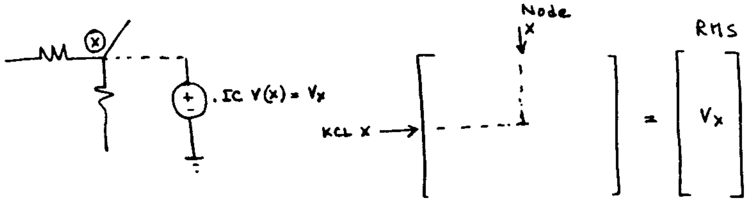
. NODESET is a suggested guess for node voltages in dc analy

. IC holds a node (force) at a prescribed value for dc analysis. The hold is released for transient analysis

Remarks

- NODESET is an aid for dc convergence
- IC is useful when starting transient conditions are different from the dc operating point, e.g. oscillators

One can think of .IC in the following manner



The solution will be V_x for node ⊗.

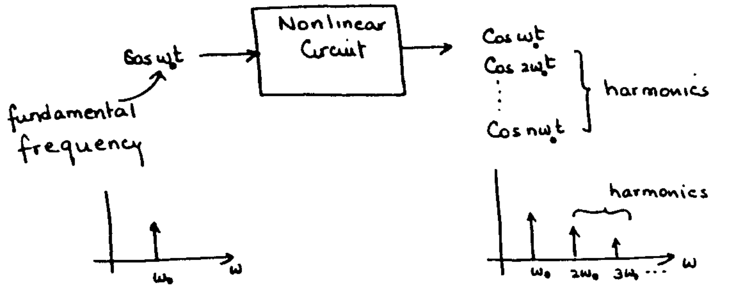
. NODESET is implemented as a .IC in the first pass. When convergence is obtained the forced nodes are released and another operating point analysis is performed

Fourier Analysis

Used to measure distortion in circuits.

Harmonic distortion (HD)

measure of distortion in wide-band circuits



input spectrum   output spectrum

For an output $x(t) = x_0 + x_1 \cos w_0 t + x_2 \cos 2w_0 t + \dots + x_k \cos k w_0 t$

$$HD_2 = \frac{|x_2|}{|x_1|}$$

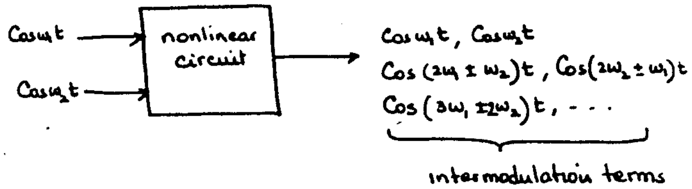
$$\vdots$$

$$HD_k = \frac{|x_k|}{|x_1|}$$

and $$THD = \frac{1}{|x_1|}\sqrt{x_2^2 + x_3^2 + \dots + x_k^2}$$

Intermodulation Distortion (IMD)

Used to measure distortion in narrow-band circuits. Here the harmonic distortion terms fall outside the bandwidth of the circuit.



$\cos w_1 t$, $\cos w_2 t$
$\cos(2w_1 \pm w_2)t$, $\cos(2w_2 \pm w_1)t$
$\cos(3w_1 \pm 2w_2)t$, ...

intermodulation terms

For narrow-band circuits the intermod terms fall within the bandwidth of the circuit.

Note

We are considering circuits in which the inputs and outputs are periodic with some fundamental frequency.

Fourier series

A periodic signal $x(t)$, period T, $w = \frac{2\pi}{T}$, can be expressed by a Fourier series expansion

$$x(t) = \sum_{k=0}^{\infty} a_k \cos kwt + b_k \sin kwt$$

where

$$a_0 = \frac{1}{T}\int_0^T x(t)\, dt$$

$$a_k = \frac{2}{T}\int_0^T x(t) \cos kwt\, dt$$

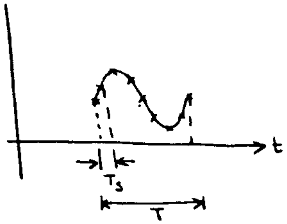
$$b_k = \frac{2}{T}\int_0^T x(t) \sin kwt\, dt$$

The above definitions apply to a continuous waveform. What do we do when we have discrete-time sampled waveforms? -as is the case with waveforms produced by a circuit simulator.

We now consider sampled waveforms

Assume a uniform sampling time of Ts and N samples then

$$T = T_s N$$



Discrete Fourier Transform (DFT)

- used for computing Fourier coefficients of sampled waveforms.

Given a time period T with N sample points we can determine at most $2k-1 = N$ coefficients for the Fourier series, i.e.

$$x_n = \sum_{i=0}^{k-1} a_i \cos 2\pi \frac{in}{N} + b_i \sin 2\pi i \frac{n}{N}$$

(Recall: $x(t) = \sum_{i=0}^{\infty} a_i \cos 2\pi i \frac{t}{T} + b_i \sin 2\pi i \frac{t}{T}$)

where
$$a_0 = \frac{1}{N} \sum_{n=0}^{N-1} x_n$$

$$a_i = \frac{2}{N} \sum_{n=0}^{N-1} x_n \cos 2\pi i \frac{n}{N}$$

$$b_i = \frac{2}{N} \sum_{n=0}^{N-1} x_n \sin 2\pi i \frac{n}{N}$$

where the coefficients are computed using numerical integration

Remarks

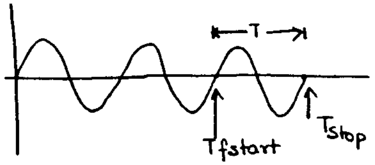
- Sampling results in aliasing. So if there are several higher harmonics in the signal then there will be an error in harmonic distortion computation
- To avoid aliasing a sufficient number of samples must be used.

SPICE's Fourier Analysis

- unequally spaced data from transient analysis due to the use of a variable timestep integration method.
- DFT requires a uniform grid ⇒ interpolate transient data onto a uniform grid of N points. Linear interpolation used!
- $N = \frac{Tstop - Tstart}{Tstep} + 1$

  where the transient command is

  .tran Tstep Tstop Tstart
- Fourier analysis applied on one period of the fundamental frequency



$$T_{fstart} = Tstop - \frac{1}{freq}$$

- DFT applied on the N interpolated points to compute 9 Harmonics in SPICE2
- The number of samples N depends on Tstep and there should be sufficient points in the Fourier interval to avoid aliasing.

- 100 timepoints in the Fourier analysis interval is a good choice. However, if K is the largest harmonic that is computed then $T_s \leq \frac{T}{10K}$. i.e., there are 10 points for the period of the largest harmonic.

SPICE 3

- default 200 grid points and linear interpolation for the DFT
- Can change number of grid points by setting fourgridsize.
- Can change the interpolation error by setting polydegree
- Allows calculation of more than 9 harmonics by setting nfreq.

Number of operations

Recall $$x_n = \sum_{i=0}^{K-1} a_i \cos 2\pi i \frac{n}{N} + b_i \sin 2\pi i \frac{n}{N}$$

$$a_0 = \frac{1}{N} \sum_{n=0}^{N-1} x_n$$

$$a_i = \frac{2}{N} \sum_{n=0}^{N-1} \underbrace{x_n \cos 2\pi i \frac{n}{N}}_{N \text{ multiplications}}, \quad b_i = \frac{2}{N} \sum_{i=0}^{N-1} \underbrace{x_n \sin 2\pi i \frac{n}{N}}_{N \text{ multiplications}}.$$

$$\left.\begin{array}{ll} \# \text{ of } a_i \text{ to be computed} & K \\ \# \text{ " } b_i \text{ " " "} & K-1 \end{array}\right\} = \underbrace{2K-1}_{N}$$

$\therefore$ Total # of multiplications $N^2 = N.K + N(K-1)$
$= N(2K-1) = N.N$

Fast Fourier Transform (FFT)

An efficient technique to calculate the N Fourier coefficients. # of multiplications $N \log_2 N$

To demonstrate its efficiency consider the complex form of the discrete Fourier transform

$$X(n) = \sum_{k=0}^{N-1} x_k e^{-j 2\pi nk/N}$$

where $X(n)$ are the Fourier coefficients obtained from the time samples $x_0, x_1, \ldots x_{N-1}$

$$\text{Let} \quad W = e^{-j2\pi/N} \Rightarrow e^{-j2\pi nk/N} = \left(e^{-j\frac{2\pi}{N}}\right)^{nk} = W^{nk}$$

Consider $N = 2^2 = 4$ then we have

$$X(0) = x_0 W^0 + x_1 W^0 + x_2 W^0 + x_3 W^0$$
$$X(1) = x_0 W^0 + x_1 W^1 + x_2 W^2 + x_3 W^3$$
$$X(2) = x_0 W^0 + x_1 W^2 + x_2 W^4 + x_3 W^6$$
$$X(3) = x_0 W^0 + x_1 W^3 + x_2 W^6 + x_3 W^9$$

$$\Rightarrow \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Note Computing the DFT as above requires 16 multiplication ($= 4^2 = N^2$).

Simplify $$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 \\ 1 & W^2 & W^0 & W^2 \\ 1 & W^3 & W^2 & W^1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

where we have used the fact that

$$W^4 = W^0, \quad W^6 = W^2, \quad W^9 = W^1$$

$$W^6 = e^{-j\frac{2\pi}{4}6} = e^{-j3\pi} = e^{-j\pi} = e^{-j\frac{2\pi}{4}\cdot 2} = W^2$$

Next factor the matrix in the following manner

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Note rows 1 & 2 have been interchanged

Let

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_0 + W^0 x_2 \\ x_1 + W^0 x_3 \\ x_0 + W^2 x_2 \\ x_1 + W^2 x_3 \end{bmatrix}$$

Since $W^2 = e^{-j\frac{2\pi}{4}\cdot 2} = e^{-j\pi} = -1 = -e^{j0} = -W^0$ we have

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x_0 + W^0 x_2 \\ x_1 + W^0 x_3 \\ x_0 - W^0 x_2 \\ x_1 - W^0 x_3 \end{bmatrix}$$

⇒ only 2 multiplications $W^0 x_2$ & $W^0 x_3$

Then

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} y_0 + W^0 y_1 \\ y_0 + W^2 y_1 \\ y_2 + W^1 y_3 \\ y_2 + W^3 y_3 \end{bmatrix}$$

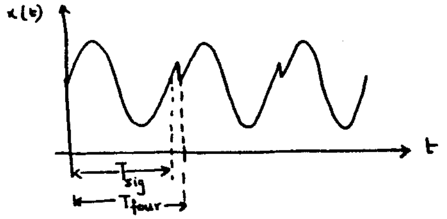
This again requires 2 multiplications

⇒ Total 4 multiplications

Error Mechanisms in Fourier Analysis

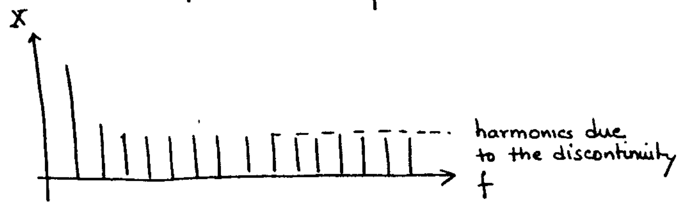
Incorrect period

- the assumption for Fourier analysis is that the waveform is T-periodic where T is the Fourier analysis interval
  i.e. $x(t+T) = x(t)$
- Suppose the period of the signal doesn't match the Fourier analysis interval



⇒ a discontinuity is generated in the waveform

⇒ a broad frequency spectrum that contaminates the results of Fourier analysis



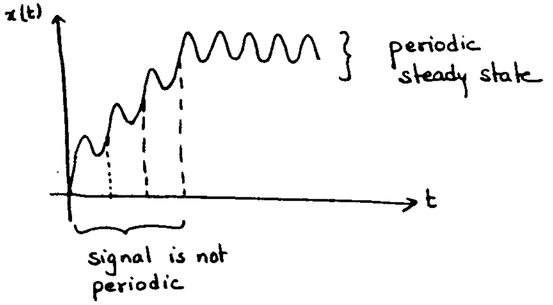
- Thus small signals cannot be resolved when an incorrect period is used.

Remark

- For oscillators require two transient analyses
  1) to determine T
  2) to accurately do the DFT

Error due to transients

- The signal should be periodic. However, if the signal has not settled into its periodic steady-state ⇒ discontinuity
  ⇒ spectral contamination



- Transient simulation should be run for a sufficiently long time to ensure that all transients have decayed
- One way to check this is to check whether the signal is periodic or not.

Errors due to aliasing

- DFT computes Fourier coefficients at a finite number of frequencies
  - Aliasing occurs when energy from the higher-order harmonics that are not computed translates as energy of the computed harmonics.
  - When sampled data points are used the sampled points can represent many signals

Example

T = 1 sec : 7th harmonic $\cos 2\pi 7t$

9th harmonic $\cos 2\pi 9t$

Suppose 16 points are used to sample the waveform

i.e. $T_s = \frac{1}{16}$

t	$\cos 2\pi 7t$	$\cos 2\pi 9t$
1/16	-0.924	-0.924
2/16	0.707	0.707
⋮		
8/16	-1	-1
⋮		
16/16	1	1

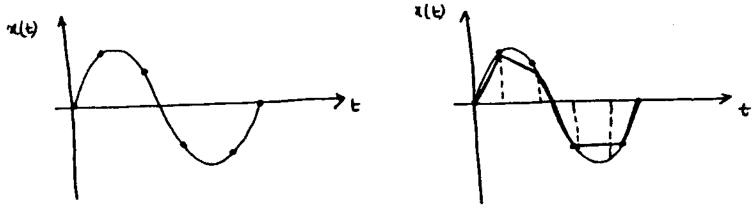
i.e. the values of the two waveforms are identical at the sample points

⇒ aliasing

- To avoid aliasing the signal must be sampled at a rate faster than the Nyquist frequency. Here the Nyquist frequency is twice the highest frequency that is present in the signal being sampled.

Interpolation Errors

- The DFT requires equally spaced data points
- Circuit simulators naturally produce unequally spaced points
  ⇒ interpolate data onto a uniform grid.



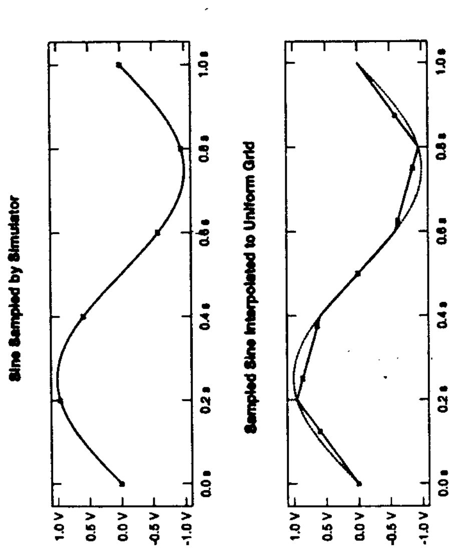


Figure 5.11: The top waveform is sampled directly at 6 equally spaced points. To illustrate the errors caused by interpolating, the 6 equally spaced sample points are linearly interpolated and sampled a 8 equally spaced points, as shown in the top waveform. The 8 sample points do not actually fall on the sine wave and so are in error.

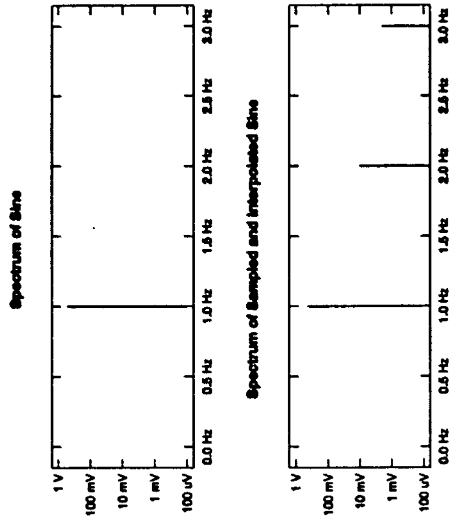


Figure 5.12: The spectrum of the 6 point sampled waveform from the previous figure is shown at the top. It exhibits no distortion. The spectrum of the 8 point interpolated and sampled waveform is shown on the bottom. It shows considerable distortion.

173-A

173-B

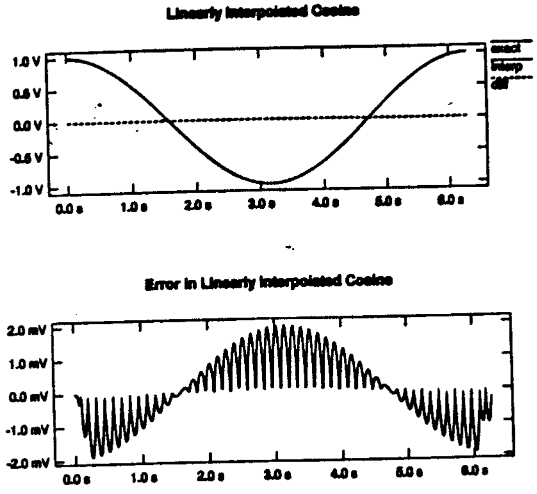


Figure 5.13: A cosine interpolated to 55 roughly equally-spaced points. The difference between the exact cosine and the interpolated cosine is shown in the lower graph.

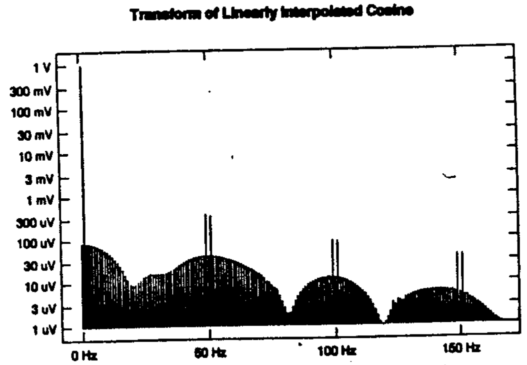


Figure 5.14: The spectrum of the interpolated cosine of Figure 5.13 on the preceding page.

- Interpolation errors can cause problems when resolving small signals

Errors due to the Simulator

- both Newton convergence and the numerical integration algorithm generate errors.
  ⇒ error in Fourier analysis
- Can reduce these errors by tightening reltol

The Fourier Integral Method

Recall $$a_k = \frac{2}{T}\int_0^T x(t)\,\cos\frac{2\pi k t}{T}\,dt$$

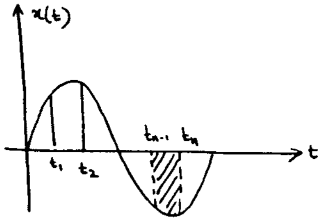
Now consider how the circuit simulator performs the transient analysis

discretize simulation interval at $t_0, t_1, t_2, --- t_N$.
solve problem at each timepoint

$$\therefore\ a_k = \frac{2}{T}\sum_{n=1}^{N}\int_{t_{n-1}}^{t_n} x(t)\,\cos\frac{2\pi k t}{T}\,dt$$

where $T = t_N - t_0$

Note the simulator approximates $x(t)$ by a low degree polynomial over the interval $[t_{n-1}, t_n]$. This is due to the integration method



$$\Rightarrow\ x(t) \simeq \sum_{i=0}^{p} c_{in} t^i \qquad \text{for} \quad t_{n-1} \le t \le t_n$$

Use this approximation to write

$$a_k \approx \frac{2}{T}\sum_{n=1}^{N}\int_{t_{n-1}}^{t_n}\sum_{i=0}^{p} c_{in} t^i \cos\frac{2\pi k t}{T}\,dt$$

$$= \frac{2}{T}\sum_{n=1}^{N}\sum_{i=0}^{p} c_{in}\underbrace{\int_{t_{n-1}}^{t_n} t^i \cos 2\pi k\frac{t}{T}\,dt}_{\text{can be integrated as a closed form expression}}$$

$$\int t\cos k\omega t\,dt = \frac{1}{(k\omega)^2}\cos k\omega t + \frac{t}{k\omega}\sin k\omega t$$

$\vdots$

Advantages

- This method is not subject to aliasing
- Can compute only the desired harmonics. Don't require calculation of higher order harmonics for improved accuracy
- Not subject to external interpolation errors
- Works with the nonuniform timesteps used by the simulator
- Gives accuracy similar to the integration method that is used.

Remarks

- Accuracy can be improved by tightening RELTOL
  ⇒ smaller timesteps
- Fourier integral method availabe in Spectre
- Stability in computing with higher order integration methods questionable

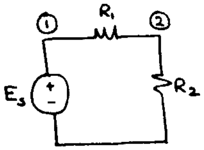
Network Sensitivity Calculations

Why sensitivities ?

- how do variations of a parameter affect the circuit
- compare quality of circuits having same nominal response
- Useful in optimization applications

Example

$$V_2 = \frac{R_2}{R_1+R_2} E_s$$



How does V_2 change with R_2

$$\frac{\partial V_2}{\partial R_2} = \frac{R_1}{(R_1+R_2)^2} E_s$$

Define a normalized sensitivity

$$S^{V_2}_{R_2} = \frac{\partial V_2}{\partial R_2} \Big/ (V_2/R_2) = \frac{R_2}{V_2} \cdot \frac{\partial V_2}{\partial R_2} = \frac{\partial V_2/V_2}{\partial R_2/R_2}$$

$$\Rightarrow \quad S^{V_2}_{R_2} = \frac{R_1}{(R_1+R_2)}$$

$$\therefore \quad \frac{\Delta V_2}{V_2} = S^{V_2}_{R_2} \frac{\Delta R_2}{R_2}$$ for small variations

This information can be used for tolerance analysis i.e. given V_2 must be accurate to a given percentage what is the required tolerance on R_2.

In General we have a function F that depends on p. then the simple sensitivity is

$$S = \frac{\partial F}{\partial p}$$

The normalized sensitivity is

$$S^F_p = \frac{p}{F}\frac{\partial F}{\partial p} = \frac{\partial \ln F}{\partial \ln p}$$

So how do we determine $\frac{\partial F}{\partial p}$ using the computer ?

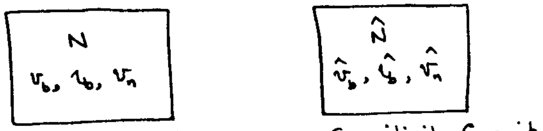
Let us consider a circuit consisting of R, I & V.

$$v_R = R i_R \Rightarrow \frac{\partial v_R}{\partial p} = R \frac{\partial i_R}{\partial p} + \frac{\partial R}{\partial p} i_R$$

$$i_I = I_s \Rightarrow \frac{\partial i_I}{\partial p} = \frac{\partial I_s}{\partial p}$$

$$v_V = V_s \Rightarrow \frac{\partial v_V}{\partial p} = \frac{\partial V_s}{\partial p}$$

Now construct a new circuit which we will call the sensitivity circuit. This circuit has the same topology as the original circuit



Original Circuit

Sensitivity Circuit

Goal the currents and voltages of the sensitivity circuit are the desired sensitivities

Thus

Resistors: $\hat{v}_R = R \hat{i}_R + \frac{\partial R}{\partial p} i_R$

Indep. I $\hat{i}_I = \frac{\partial I_s}{\partial p}$

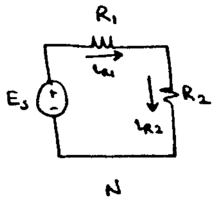
Indep V $\hat{v}_V = \frac{\partial V_s}{\partial p}$

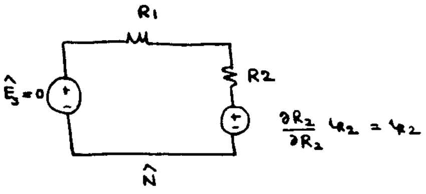
Thus each branch voltage in the sensitivity network is the sensitivity of the same branch voltage. Similarly currents.

Remarks

- Each resistor is replaced by a resistance of the same value in series with a voltage source
- Each independent source replaced by its partial derivative w.r.t. p.

Example





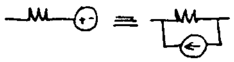
$$i_{R2} = \frac{E_s}{R_1 + R_2} \qquad \text{from } N.$$

$$\hat{v}_{R2} = i_{R2} - i_{R2}\left(\frac{R_2}{R_1+R_2}\right) \qquad \text{from } \hat{N}$$

$$= \frac{R_1}{R_1+R_2}\, i_{R2} = \frac{R_1}{(R_1+R_2)^2} E_s.$$

which is the expression we had derived earlier.

Observations

- The sensitivity circuit has the same topology and the same nodal admittance matrix Y. (Convert Thevenin's to Norton )
- The source vector J is different
- LU factors of the original circuit can be used to determine the solution of the sensitivity circuit by forward and back substitutions
- All sensitivities of a parameter p determined by forward/ back substitution.

- If sensitivities are required for multiple parameters $p_1, p_2, \dots p_m$, then the sensitivity circuit is solved with different source vectors ⇒ m forward/ back substitution

General Form for linear algebraic systems

Consider linear circuits, dc or small-signal ac analysis

$$Ax = b$$

where A & b may be real or complex and depend on some parameters p

The solution of $Ax = b$ is $x = A^{-1}b$ where no explicit calculation of A^{-1} is done. LU factorization is used

Differentiate w.r.t. p

$$A\frac{\partial x}{\partial p} + \frac{\partial A}{\partial p} x = \frac{\partial b}{\partial p}$$

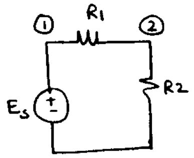
$$\Rightarrow A\frac{\partial x}{\partial p} = -\left(\frac{\partial A}{\partial p} x - \frac{\partial b}{\partial p}\right)$$

Given A, b we can determine $\frac{\partial A}{\partial p}$, $\frac{\partial b}{\partial p}$ and thus the RHS can be constructed

$\frac{\partial x}{\partial p}$ solved by forward/back substitutions since A is available in its LU factors.

Example

$$\begin{bmatrix} 1/R_1 & -1/R_1 & 1 \\ -1/R_1 & 1/R_1 + 1/R_2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ I_E \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ E_s \end{bmatrix}$$



Rewrite as

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ -1/R_1 & \frac{1}{R_1}+1/R_2 & 0 \\ \frac{1}{R_1} & -\frac{1}{R_1} & 1 \end{bmatrix}}_{A} \begin{bmatrix} v_1 \\ v_2 \\ I_E \end{bmatrix} = \underbrace{\begin{bmatrix} E_s \\ 0 \\ 0 \end{bmatrix}}_{b} \Rightarrow x = \begin{bmatrix} E_s \\ \frac{R_2}{R_1+R_2} E_s \\ -\frac{E_s}{R_1+R_2} \end{bmatrix}$$

<u>Note</u> A is in a triangularized form

Suppose $p = R_2$ then $\frac{\partial A}{\partial R_2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1/R_2^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$, $\frac{\partial b}{\partial R_2} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

$$\therefore \text{RHS} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1/R_2^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} E_s \\ \frac{R_2}{(R_1+R_2)} E_s \\ -\frac{E_s}{R_1+R_2} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{-E_s}{R_2(R_1+R_2)} \\ 0 \end{bmatrix}$$

To determine $\frac{\partial x}{\partial p}$, Solve $\begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{R_1} & \frac{1}{R_1}+\frac{1}{R_2} & 0 \\ +1/R_1 & -1/R_1 & 1 \end{bmatrix} \begin{bmatrix} \partial v_1/\partial R_2 \\ \partial v_2/\partial R_2 \\ \partial I_E/\partial R_2 \end{bmatrix} = - \begin{bmatrix} 0 \\ -\frac{E_s}{R_2(R_1+R_2)} \\ 0 \end{bmatrix}$

$$\therefore \quad \frac{\partial v_1}{\partial R_2} = 0$$

$$\left(\frac{1}{R_1} + \frac{1}{R_2}\right) \frac{\partial v_2}{\partial R_2} = \frac{E_s}{R_2(R_1+R_2)} \Rightarrow \frac{\partial v_2}{\partial R_2} = \frac{R_1 E_s}{(R_1+R_2)^2}$$

and

$$-\frac{1}{R_1} \frac{\partial v_2}{\partial R_2} + \frac{\partial I_E}{\partial R_2} = 0$$

$$\therefore \frac{\partial I_E}{\partial R_2} = \frac{1}{R_1} \frac{\partial v_2}{\partial R_2} = \frac{E_s}{(R_1+R_2)^2}$$

<u>Note</u> $\frac{\partial v_2}{\partial R_2} = \frac{R_1 E_s}{(R_1+R_2)^2}$ as before

<u>Remark</u>

The sensitivity of all components of x is seldom required. Normally we have some output ϕ that is related to x and we are interested in $\partial\phi/\partial p_i$.

How can this be done in a computationally efficient manner?

$$Ax = b \qquad \text{———} \ \textcircled{0}$$

<u>Recall</u>

$$A \frac{\partial x}{\partial p} = -\left(\frac{\partial A}{\partial p} x - \frac{\partial b}{\partial p} \right) \qquad \text{———} \ \textcircled{1}$$

Now let $\phi = c^T x$

$$\Rightarrow \frac{\partial \phi}{\partial p} = c^T \frac{\partial x}{\partial p}$$

But $\frac{\partial x}{\partial p} = -A^{-1} \left(\frac{\partial A}{\partial p} x - \frac{\partial b}{\partial p} \right)$

which results in

$$\frac{\partial \phi}{\partial p} = -c^T A^{-1} \left(\frac{\partial A}{\partial p} x - \frac{\partial b}{\partial p} \right) \qquad \text{———} \ \textcircled{2}$$

Next define an <u>adjoint vector</u> x_a such that

$$x_a^T = -c^T A^{-1}$$

$$\Rightarrow x_a^T A = -c^T$$

$$\Rightarrow A^T x_a = -c \qquad \text{———} \ \textcircled{3}$$

② can be expressed as.

$$\frac{\partial \phi}{\partial p} = x_a^T \left(\frac{\partial A}{\partial p} x - \frac{\partial b}{\partial p} \right)$$

Once x_a is determined from a solution of ③ the sensitivity $\frac{\partial \phi}{\partial p}$ can be readily computed

<u>Note</u> Only two linear equations ⓪ & ③ need to be solved irrespective of the number of parameters p_i

Computational Procedure

1) Solve $A x = b$

A decomposed into LU factors

$$LU x = b$$

a) Solve $Ly = b$

b) Solve $Ux = y$

2) Solve the adjoint system

$$A^T x_a = -c$$

$$\Rightarrow (LU)^T x_a = -c$$

$$\Rightarrow U^T L^T x_a = -c$$

a) Solve $U^T z = -c$

b) Solve $L^T x_a = z$

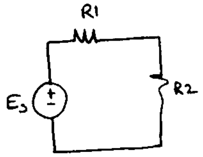
Note only forward and back solves are required with the same triangularized matrices

3) For each parameter p_i, form $\partial A/\partial p_i$ and $\partial b/\partial p_i$ then obtain

$$\frac{\partial \phi}{\partial p_i} = (x_a)^T \left(\frac{\partial A}{\partial p_i} x - \frac{\partial b}{\partial p_i} \right)$$

Example

$$\begin{array}{ccc} V_1 & V_2 & I_E \\ \hline 1 & 0 & 0 \\ -1/R_1 & \frac{1}{R_1}+\frac{1}{R_2} & 0 \\ +\frac{1}{R_1} & -\frac{1}{R_1} & 1 \end{array} \qquad \begin{bmatrix} E_s \\ 0 \\ 0 \end{bmatrix}$$



Note matrix is already in a triangular form.

$$x = \begin{bmatrix} E_s \\ \frac{R_2}{R_1+R_2} E_s \\ -\frac{E_s}{R_1+R_2} \end{bmatrix}$$

Adjoint system is:

$$A^T = \begin{bmatrix} 1 & -1/R_1 & 1/R_1 \\ 0 & \frac{1}{R_1}+\frac{1}{R_2} & -1/R_1 \\ 0 & 0 & 1 \end{bmatrix}$$

The output is $v_2 = \phi = \underbrace{\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}}_{c^T} \begin{bmatrix} v_1 \\ v_2 \\ I_E \end{bmatrix}$

Solve $A^T x_a = -c \Rightarrow$

$$\begin{bmatrix} 1 & -1/R_1 & 1/R_1 \\ 0 & 1/R_1 + 1/R_2 & -1/R_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_{1a} \\ v_{2a} \\ I_{Ea} \end{bmatrix} = - \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

The solution is $x_a = \begin{bmatrix} -R_2/(R_1+R_2) \\ -R_1 R_2/(R_1+R_2) \\ 0 \end{bmatrix}$

If we want $\frac{\partial \phi}{\partial R_2}$ we compute $\frac{\partial A}{\partial R_2}$ & $\frac{\partial b}{\partial R_2}$

$$\Rightarrow \frac{\partial \phi}{\partial R_2} = \begin{bmatrix} \frac{-R_2}{R_1+R_2} & -\frac{R_1 R_2}{R_1+R_2} & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\frac{1}{R_2^2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} E_s \\ \frac{R_2}{R_1+R_2} E_s \\ -\frac{E_s}{R_1+R_2} \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{R_2}{R_1+R_2} & -\frac{R_1 R_2}{R_1+R_2} & 0 \end{bmatrix} \begin{bmatrix} 0 \\ \frac{-1}{R_2(R_1+R_2)} E_s \\ 0 \end{bmatrix}$$

$$= \frac{R_1}{(R_1+R_2)^2} E_s$$

Suppose we wanted $\frac{\partial v_2}{\partial E_s}$

$$\Rightarrow \frac{\partial \phi}{\partial E_s} = \begin{bmatrix} \frac{-R_2}{R_1+R_2} & -\frac{R_1R_2}{R_1+R_2} & 0 \end{bmatrix} \left(- \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right)$$

$$= \frac{R_2}{R_1+R_2}$$

i.e. no additional matrix solve is required.

Remarks

- If we want the sensitivity of a few outputs w.r.t. a large number of parameters then use the adjoint method
- However, if we want sensitivity for a large number of outputs w.r.t. a few parameters then use the direct or sensitivity circuit approach.

Observation

Suppose we want to evaluate a scalar output function for multiple RHS vectors, then the adjoint method can be used to advantage

$$A x_i = b_i$$
$$\phi_i = c^T x_i$$

$$\phi_i = c^T x_i = c^T A^{-1} b_i = -(-c^T A^{-1}) b_i$$
$$= -(x_a)^T b_i$$

i.e. all ϕ_i can be evaluated with a single analysis of the adjoint system.

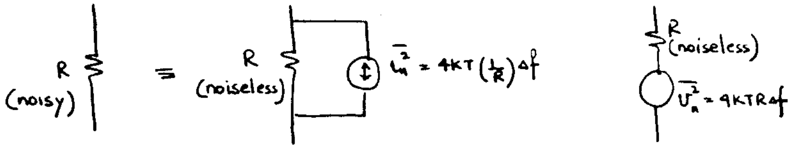
This is the basic idea in noise analysis.

Noise Analysis

semiconductor devices (BJT, D, MOS, R) introduce noise in a circuit. Noise is present because charge is not continuous but is carried in discrete amounts ($q = 1.6 \times 10^{-16} C$)

Various noise sources

- Thermal noise : associated with the random thermal motion of electrons within the physical body of a resistor


R
(noisy)
≡
R
(noiseless)
$\overline{i_n^2} = 4KT(\frac{1}{R})\Delta f$
R
(noiseless)
$\overline{v_n^2} = 4KTR\Delta f$

- Shot noise : associated with flow of carriers across a barrier (pn junctions)

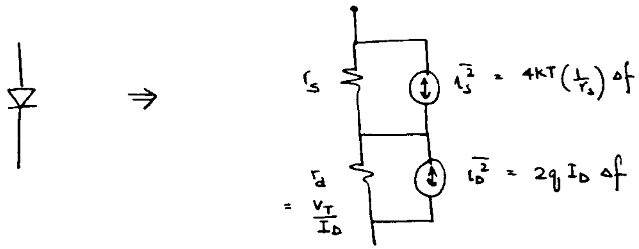
$$\overline{i^2} = 2q I_{DC} \Delta f$$

- Flicker noise (1/f noise) : associated with surface states or traps

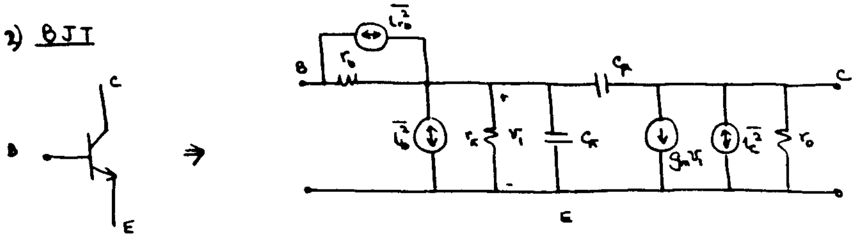
$$\overline{i^2} = K_1 \frac{I^a}{f^b} \Delta f$$

Noise models for semiconductor devices

1) Diode


⇒
r_s
$\overline{i_s^2} = 4KT(\frac{1}{r_s})\Delta f$
$r_d = \frac{V_T}{I_D}$
$\overline{i_D^2} = 2q I_D \Delta f$

2) BJT

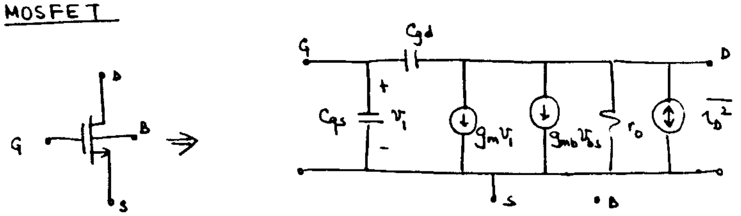


$$\overline{i_{rb}^2} = 4KT\left(\frac{1}{r_b}\right)\Delta f$$

$$\overline{i_b^2} = \underbrace{2q I_B \Delta f}_{\text{shot noise}} + \underbrace{K_1 \frac{I_B^a}{f} \Delta f}_{\text{flicker noise}}$$

$$\overline{i_c^2} = 2q I_c \Delta f$$

3) MOSFET



$$\overline{i_D^2} = 4KT\gamma g_{do} + \frac{K}{f}\,\frac{g_m^2}{WLC_{ox}}$$

where $g_{do} = g_{ds}\big|_{V_{DS}=0}$

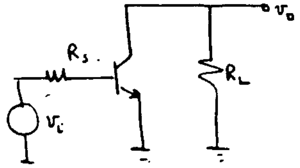
$\gamma = 2/3$ for longchannel devices in saturation

For a long channel MOSFET $g_{ds}\big|_{V_{DS}=0} = g_{msat}$

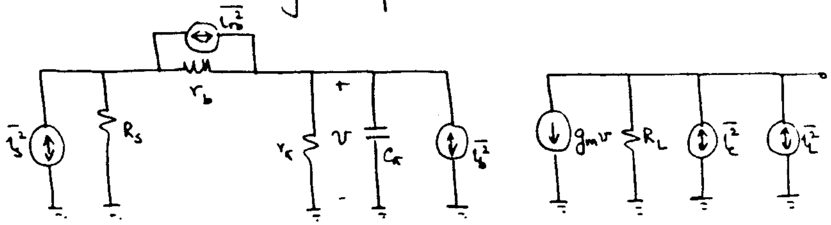
$$\therefore 4KT\gamma g_{do} = 4KT\left(\frac{2}{3}\right) g_m$$

Example of Noise Analysis

$C_\mu = 0, \quad r_o = \infty$



Look at the small-signal equivalent circuit



Note There are 5 noise sources in the circuit and all of these are uncorrelated

To analyze this circuit we proceed in the following manner
For each noise source $\overline{i_n^2}$

1) replace $\overline{i_n^2}$ with a sinusoidal deterministic value i_n
2) calculate $v_{on}(f) = i_n\,|H(jf)|$
3) mean square output voltage is
$$\overline{v_{on}^2} = \overline{i_n^2}\,|H(jf)|^2$$

The total output noise is then

$$\overline{v_{o\,Total}^2} = \overline{v_{on1}^2} + \overline{v_{o2}^2} + \cdots$$

Observation

- For each noise source we are solving the same circuit with a different RHS or source vector i.e. we have the following situation

$$\left.\begin{aligned} Y v_i &= J_i \\ \phi_i &= c^T v_i \end{aligned}\right\} \quad i = 1, 2, \dots m$$

where J_i are the various source (RHS) vectors and ϕ_i the corresponding outputs

$$\phi_i = c^T Y^{-1} J_i = -(-c^T Y^{-1}) J_i$$
$$= -(v_a)^T J_i$$

where v_a is obtained by solving $Y^T v_a = -c$

Note ϕ_i can be evaluated with a single analysis of the adjoint system.

This is an efficient way for doing noise analysis.

Remark
J_i contains at most two nonzero entries of the form $\pm i_n$.

Output noise calculation

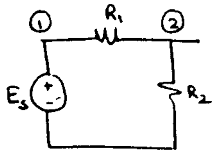
For each noise source 'i' determine ϕ_i. Then

$$v_{o\,\text{noise}} = \left(\sum_{i=1}^{m} |\phi_i|^2 \right)^{1/2}$$

Calculation of each ϕ_i requires $-(v_a)^T J_i$ which involves 1 multiplication and a subtraction

$$\begin{bmatrix} v_{a1} & v_{a2} & \cdots & v_{an} \end{bmatrix} \begin{bmatrix} \vdots \\ -i_n \\ \vdots \\ +i_n \end{bmatrix} \begin{matrix} \\ \leftarrow i \\ \\ \leftarrow j \end{matrix} \qquad i_n(-v_{ai} + v_{aj})$$

Example
Calculate $\overline{v_o^2}$



$$A = \begin{bmatrix} 1 & 0 & 0 \\ -1/R_1 & 1/R_1 + 1/R_2 & 0 \\ 1/R_1 & -1/R_1 & 1 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 1 & -1/R_1 & 1/R_1 \\ 0 & 1/R_1 + 1/R_2 & -1/R_1 \\ 0 & 0 & 1 \end{bmatrix}$$

We have $x_a = \begin{bmatrix} -R_2/(R_1+R_2) \\ -R_1R_2/(R_1+R_2) \\ 0 \end{bmatrix}$

Compute noise output due to R_1: $b_1 = \begin{bmatrix} 0 \\ -i_{R1} \\ i_{R1} \end{bmatrix}$

$$\therefore v_{o1} = -i_{R1} \begin{bmatrix} \frac{-R_2}{R_1+R_2} & -\frac{R_1R_2}{R_1+R_2} & 0 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$$

$$= -i_{R1} \left[\frac{R_1R_2}{R_1+R_2} \right] = i_{R1} \frac{R_1R_2}{R_1+R_2}$$

$$v_{o2} = -i_{R2} \begin{bmatrix} \frac{-R_2}{R_1+R_2} & -\frac{R_1R_2}{R_1+R_2} & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = i_{R2} \frac{R_1R_2}{R_1+R_2}$$

$$\therefore \overline{V_o^2} = \left(\frac{R_1R_2}{R_1+R_2} \right)^2 \left[\overline{i_{R1}^2} + \overline{i_{R2}^2} \right]$$

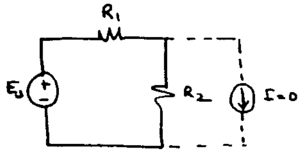
Network Interpretation of the Adjoint method

The adjoint vector is obtained as a solution of

$$A^T x_a = -c$$

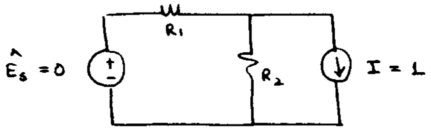
Consider the resistive example

$$A = \begin{bmatrix} 1 & 0 & 0 \\ -1/R_1 & 1/R_1 + 1/R_2 & 0 \\ 1/R_1 & -1/R_1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} E_s \\ 0 \\ 0 \end{bmatrix}$$



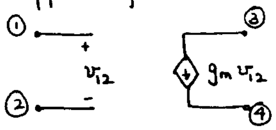
$$A^T = \begin{bmatrix} 1 & -1/R_1 & 1/R_1 \\ 0 & 1/R_1 + 1/R_2 & -1/R_1 \\ 0 & 0 & 1 \end{bmatrix} \qquad c = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

From this the adjoint network is

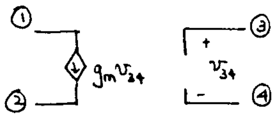


For R, L, C the adjoint network is the same as the original network.

What happens for controlled sources?



N



$\hat{N}$

Sensitivities for Nonlinear circuits

1) DC Sensitivity

We solve $f(x) = 0$ nonlinear equations

$\Rightarrow x^*$ is the solution

Suppose we are interested in sensitivities w.r.t. parameter p.

i.e. $f(x, p) = 0$

differentiate w.r.t. p

$$\frac{\partial f}{\partial x}\Big|_{x^*} \frac{\partial x}{\partial p} + \frac{\partial f}{\partial p} = 0$$

i.e.

$$\frac{\partial f}{\partial x}\Big|_{x^*} \frac{\partial x}{\partial p} = -\frac{\partial f}{\partial p}$$

Notes:
- $\frac{\partial f}{\partial x}\Big|_{x^*}$ is the Jacobian matrix at the dc operating point
- It is available in factored form at the convergence of Newton's method used to solve $f(x) = 0$

$\Rightarrow$ sensitivity $\frac{\partial x}{\partial p}$ can be obtained by forward/back substitution for RHS $= -\frac{\partial f}{\partial p}$.

- The derivatives $\frac{\partial f}{\partial p}$ are required for computing the sensitivity.

Remarks
- The above approach is the direct sensitivity approach.
- As with the linear case an adjoint formulation can also be constructed.

2) AC Sensitivity

small-signal circuit is a linear circuit and the linear sensitivity analysis can be used.

This becomes tricky if a parameter affects the dc operating point!

3) Transient Sensitivity

Circuit equations are differential algebraic equations (DAEs) that are expressed as

$$f(x, \dot{x}, p) = 0 \qquad \text{———} \quad ①$$

where p is a parameter.

At the dc operating point we have

$$f(x) = 0 \qquad \text{———} \quad ②$$

and

$$\left.\frac{\partial f}{\partial x}\right|_{x_0} \left[\frac{\partial x}{\partial p}\right]_0 = -\frac{\partial f}{\partial p}. \qquad \text{———} \quad ③$$

Differentiate ① w.r.t. p

$$\Rightarrow \frac{\partial f}{\partial x}\frac{\partial x}{\partial p} + \frac{\partial f}{\partial \dot{x}}\frac{\partial \dot{x}}{\partial p} + \frac{\partial f}{\partial p} = 0$$

Let $\frac{\partial x}{\partial p} = z \qquad \Rightarrow \quad \frac{\partial \dot{x}}{\partial p} = \frac{d}{dt}\frac{\partial x}{\partial p} = \dot{z}$

$$\Rightarrow \frac{\partial f}{\partial x} z + \frac{\partial f}{\partial \dot{x}} \dot{z} = -\frac{\partial f}{\partial p} \qquad \text{———} \quad ④$$

where z is the sensitivity vector

Note. ④ is a linear time varying DAE which is solved with the initial condition $z_0 = -\left(\frac{\partial f}{\partial x}\right)_{x_0}^{-1} \frac{\partial f}{\partial p}$ from ③

Pole/zero Analysis

Consider small-signal AC analysis where we determine the output of some transfer function as a function of frequency.

Let us represent the transfer function as

$$T(s) = \frac{N(s)}{D(s)} \qquad \text{———} \quad ①$$

We would like to know T(s) in the following form

$$T(s) = \frac{k \prod_{i=1}^{n} (s - z_i)}{\prod_{i=1}^{m} (s - p_i)} \qquad \text{———} \quad ②$$

where z_i are the zeros and p_i the poles of the transfer function.

Why this form?

1) The small-signal ac response can be obtained cheaply from ②. Otherwise require F/B solves for each frequency when doing the conventional ac analysis

2) The time-domain response can be determined by using inverse Laplace transforms

3) For closed-loop systems a knowledge of the poles allows determining the stability of the system. Right-half-plane poles ⇒ unstable system.

So how do we determine the poles and zeros of network transfer functions?

Note This is a small-signal analysis for nonlinear circuits

For small-signal conditions the circuit equations can be expressed as

$$A(s)\,x = b \qquad \text{——} \ ①$$

where $A(s) = G + sC$

Suppose the output of interest is

$$y = \ell^T x \qquad \text{——} \ ②$$

Then we are interested in determining the poles/zeros of y.

$$y = \ell^T A(s)^{-1} b = \ell^T (G+sC)^{-1} b$$

$$= \ell^T \frac{\text{adj}\,(G+sC)}{\det\,(G+sC)}\, b$$

Note. For all network transfer functions the determinant of $(G+sC)$ is the denominator

⇒ poles of the transfer function are given by
$$\det\,(G+sC) = 0$$

What about zeros?

Write ① & ② as a complete system of equations

$$A(s)\,x = b$$
$$-\ell^T x + y = 0$$

$$\Rightarrow \begin{bmatrix} A(s) & 0 \\ -\ell^T & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

From this the output can be determined by use of Cramer's rule

i.e.
$$y = \frac{\det\begin{bmatrix} A(s) & b \\ -\ell^T & 0 \end{bmatrix}}{\det\begin{bmatrix} A(s) & 0 \\ -\ell^T & 1 \end{bmatrix}} = \frac{\det\begin{bmatrix} A(s) & b \\ -\ell^T & 0 \end{bmatrix}}{\det\,(A(s))}.$$

So the zeros can be found from the determinant of an augmented matrix $A_a(s)$

additional column

$$A_a(s) = \begin{bmatrix} A(s) & b \\ -\ell^T & 0 \end{bmatrix} \Rightarrow \left[\begin{array}{c|c} A(s) & b \\ \hline -\ell^T & 0 \end{array}\right]$$

additional row

The zeros are given by $\det\big(A_a(s)\big) = 0$.

Remarks

- Both the poles and zeros can be obtained from the roots of two determinants.
- The determinants are polynomials in s.

How do we determine the roots of these determinants?

We will look at two methods: i) Muller's method and the QZ-Algorithm.

Muller's Method

This is an iterative method for finding the roots of a polynomial

$$f(s) = a_0 + a_1 s + a_2 s^2 + \cdots + a_n s^n$$

First, the function is approximated by a quadratic, i.e.

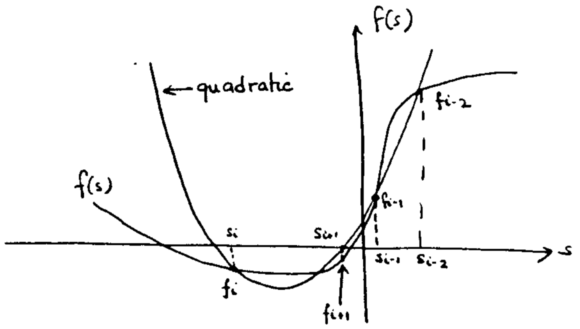
$$b_0 + b_1 s + b_2 s^2$$

Procedure

- start with initial values of s, s_i, s_{i-1}, s_{i-2} and the corresponding values of f f_i, f_{i-1}, f_{i-2}. Solve linear equations to detemine b_0 b_1, b_2
- From the quadratic $b_0+b_1 s+b_2 s^2$ detemine the root of smallest magnitude $\Rightarrow s_{i+1}$
- Repeat process to termination i.e. $f(s) < \sim 10^{-20}$
- Once a root of $f(s)$ has been determined its effect is factored out of the function $f(s)$ and Muller's method applied to the reduced polynomial

i.e. $f^k(s) = \dfrac{f(s)}{\prod_{j=1}^{k}(s-s_j)}$

after k roots of $f(s)$ have been detemined



How is this method implemented in a circuit simulator?

Note the process relies on calculating $f(s) = \det(A(s))$ efficiently for a given value of s, say $s_i = \sigma_i + j\omega_i$

How do we calculate the determinant?

For $A = LU$ we can compute the detemninant in the following manner

$$\det(A) = \det(LU) = \det(L)\det(U)$$

$$L = \begin{bmatrix} l_{11} & & & \\ l_{12} & l_{22} & & 0 \\ \vdots & & \ddots & \\ l_{1n} & \cdots & \cdots & l_{nn} \end{bmatrix} \Rightarrow \det(L) = l_{11} l_{22} \ldots l_{nn}$$

$$U = \begin{bmatrix} 1 & u_{12} & \cdots & u_{1n} \\ & 1 & u_{23} \cdots & u_{2n} \\ & 0 & \ddots & u_{n-1\,n} \\ & & & 1 \end{bmatrix} \Rightarrow \det(U) = 1$$

$$\therefore \quad \det(A) = l_{11} l_{22} \ldots l_{nn}.$$

Thus, given $s = s_i$ the matrix load $\Rightarrow G + s_i C$

LU factorization $\Rightarrow \det(G + s_i C)$

This idea is used with Muller's method.

Remarks

- SPICE 2 doesn't have .PZ analysis
- SPICE3 / HSPICE support .PZ analysis using Muller's method
- Muller's method is not very reliable. Check the # of options in HSPICE for .PZ analysis
- The next method QZ-algorithm is a robust technique

2) QZ-algorithm

- A method to solve the generalized eigen value problem

  i.e. $Ax = \lambda Bx$

  where both A & B are matrices
- The solution is obtained by solving $\det(A - \lambda B) = 0$

Note This is exactly the form that we have for A(s) and Aa(s)

$$\det(A(s)) = \det(G + sC) \Rightarrow G \to A, \quad C \to B,$$

Remark

The QZ algorithm is available as a subroutine package (rgg in eispack from netlib)

Computing poles & zeros

QZ algorithm returns the determinant in the following form:

$$\det(G + sC) = \prod(\alpha_i + \beta_i s)$$

where α_i are complex & β_i are real. Either α_i or β_i may be zero.

Poles use QZ with $A(s) = G + sC$

Zeros use QZ with $A_a(s) = G_a + sC_a$.

$$G_a = \begin{bmatrix} G & b \\ -\ell^T & 0 \end{bmatrix} \qquad C_a = \begin{bmatrix} C & 0 \\ 0 & 0 \end{bmatrix}$$

Remark

- QZ-algorithm results in dense matrices (n×n). Hence it cannot be applied to circuits with > 500 nodes

What do we do for larger circuits?

Question 1 Do we really need to know all of the poles of a system?

e.g. a linear network of 10,000 or more nodes?

Question 2 What would one do even if one could generate the complete set of poles?

Observation

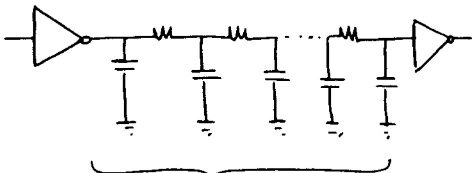
Some of the poles make an insignificant contribution to the circuit performance

⇒ obtain dominant poles that can accurately represent the circuit behavior.

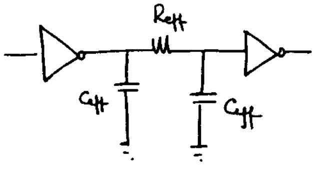
When is such an approach useful?

- Analog designer's look at only the dominant poles.
- For IC interconnect simulation would like to have a simple delay estimate

So what we would like to have is an efficient technique for obtaining the dominant poles of a circuit



long interconnect



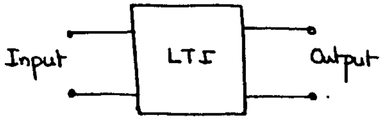
approximate representation

Efficient techniques to obtain a dominant pole representation are

AWE (Asymptotic Waveform Evaluation)
PVL (Pade' Via Lanczos)

The underlying idea is that of Pade' approximation

Consider a linear TI network



The output / input relationship is described by a transfer function

$$H(s) = \frac{a_0' + a_1' s + a_2' s^2 + \ldots a_m' s^m}{b_0' + b_1' s + b_2' s^2 + --- + b_n' s^n}$$

This TF is approximated by

$$H_{p,q}(s) = \frac{a_0 + a_1 s + a_2 s^2 + -- + a_p s^p}{1 + b_1 s + b_2 s^2 + --- + b_q s^q}$$

where $p < m$ and $q < n$

The above approximation $H_{p,q}(s)$ is the Pade' approximation of H(s). of type $[p/q]$.

Defn A Pade' approximant of F(s) is denoted by

$$[p/q] = \frac{N_p(s)}{D_q(s)}$$

Why would the approximation work ?

Consider H(s) the exact TF. Suppose the poles are $p_1, p_2, -- p_n$ then we have

$$H(s) = \sum_{i=1}^{n} \frac{k_i}{s - p_i}$$

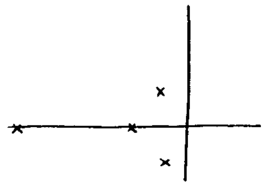
$$\text{and} \quad h(t) = \sum_{i=1}^{n} k_i e^{-p_i t}$$

We approximate the response as a reduced-order approximation

$$H_{p,q}(s) = \sum_{i=1}^{q} \frac{k_i'}{s - p_i'}$$

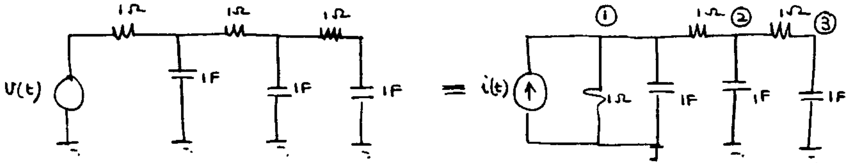
$$\tilde{h}(t) = \sum k_i' e^{-p_i t}$$

This is possible by considering only the dominant poles. Poles far away from the origin have a negligible contribution to the overall response



In the transient the non dominant pole terms would decay rapidly !

Example



$$\begin{bmatrix} 2+s & -1 & 0 \\ -1 & 2+s & -1 \\ 0 & -1 & 1+s \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} i(s) \\ 0 \\ 0 \end{bmatrix}$$

$$V_3(s) = \frac{V(s)}{s^3+5s^2+6s+1} \Rightarrow \frac{V_3(s)}{V(s)} = \frac{1}{s^3+5s^2+6s+1}$$

For $V(s) = \frac{1}{s}$ i.e. a step function

$$V_3(s) = \frac{1}{s(s^3+5s^2+6s+1)}$$

$$= \frac{K_1}{s} + \frac{K_2}{s+3.247} + \frac{K_3}{s+1.555} + \frac{K_4}{s+0.1981}$$

$$= \frac{1}{s} - \frac{0.0597}{s+3.247} + \frac{0.2801}{s+1.555} - \frac{1.2204}{s+0.1981}$$

$$\Rightarrow v(t) = 1 - 0.0597\,e^{-3.247t} + 0.2801\,e^{-1.555t} - 1.2204\,e^{-.1981t}$$

Pade approximations

$$[0/1] = \frac{a_0}{1+b_1 s} = \frac{1}{1+6s+5s^2+s^3}$$

$$\Rightarrow a_0 = 1; \quad b_1 = 6a_0 = 6$$

$$\therefore H_{0,1}(s) = \frac{1}{1+6s}$$

$$[1/2] = \frac{a_0+a_1 s}{1+b_1 s+b_2 s^2} = \frac{1}{1+6s+5s^2+s^3}$$

$$a_0 + (6a_0+a_1)s + (5a_0+6a_1)s^2 + (a_0+5a_1)s^3 = 1+b_1 s+b_2 s^2$$

$$\Rightarrow \begin{aligned} a_0 &= 1 \\ 6a_0+a_1 &= b_1 \\ 5a_0+6a_1 &= b_2 \\ a_0+5a_1 &= 0 \quad \Rightarrow a_1 = \frac{-a_0}{5} = -\frac{1}{5} \end{aligned}$$

$$\Rightarrow b_1 = 6 - \frac{1}{5} = 29/5, \quad b_2 = 5 - \frac{6}{5} = \frac{19}{5}$$

$$\therefore H_{1,2}(s) = \frac{1 - \frac{1}{5}s}{1 + \frac{29}{5}s + \frac{19}{5}s^2}$$

$$\Rightarrow V_3(s) = \frac{K_1}{s} + \frac{K_2}{s+1.3282} + \frac{K_3}{s+0.1981} \qquad \text{(step input)}$$

$$= \frac{1}{s} + \frac{0.2219}{s+1.3282} - \frac{1.2219}{s+0.1981}$$

$$v(t) = 1 + 0.2219\,e^{-1.3282t} - 1.2219\,e^{-0.1981t}$$

Note From the exact transfer function the dominant pole is -0.1981. This is the same as from the approximate equation

Recall

A Pade' approximant of F(s) is

$$[p/q] = \frac{N_p(s)}{D_q(s)}$$

Where there are several choices of p, q. Thus can build up an array of approximants called the Pade' table

q \ p	0	1	2	- - -
0	[0/0]	[1/0]	[2/0]	
1	[0/1]	[1/1]	[2/1]	
2	[0/2]	[1/2]	[2/2]	

Example For $e^s = 1+s+\frac{s^2}{2!}$ the Pade' Table is

q \ p	0	1	2
0	$\frac{1}{1}$	$\frac{1+s}{1}$	$\frac{2+2s+s^2}{2}$
1	$\frac{1}{1-s}$	$\frac{2+s}{2-s}$	$\frac{6+4s+s^2}{6-2s}$
2	$\frac{2}{2-2s+s^2}$	$\frac{6+2s}{6-4s-s^2}$	$\frac{12+6s+s^2}{12-6s+s^2}$ . . .

How good are these approximants?

In practice work well for interconnect circuits

Problems

- may introduce RHP poles in a stable system
- may introduce small parasitic poles close to the origin – but with small residues

What is AWE?

Consider McLaurin series expansion

$$F(s) = m_0 + m_1 s + m_2 s^2 + \cdots = \sum_{i=0}^{\infty} m_i s^i$$

where m_i is called the ith moment

Recall $\quad F(s) = \int_0^\infty f(t)\, e^{-st}\, dt \qquad$ Laplace Transform

$$= \int_0^\infty f(t)\left[1 - st + \frac{(st)^2}{2!} - \frac{(st)^3}{3!} + \cdots\right] dt$$

$$= \int_0^\infty f(t)\,dt - s\int_0^\infty t\, f(t)\,dt + s^2\int_0^\infty \frac{t^2}{2!} f(t)\,dt$$

$$- s^3 \int_0^\infty \frac{t^3}{3!} f(t)\,dt + \cdots$$

$$= m_0 + m_1 s + m_2 s^2 + \cdots$$

where $\quad m_i = \frac{(-1)^i}{i!}\int_0^\infty t^i f(t)\, dt$

Note $\quad \int_0^\infty t^i f(t)\, dt$ is the ith moment

Suppose $m_0, m_1, m_2, \ldots$ are known. We will shortly see how this can be done then $a_0, a_1, \ldots a_p$ and $b_1, b_2, \ldots b_q$ can be obtained in the following manner

$$m_0 + m_1 s + m_2 s^2 + \cdots\cdots m_{2q-1} s^{2q-1} = \frac{a_0 + a_1 s + \cdots a_{q-1} s^{q-1}}{1 + b_1 s + \cdots b_q s^q}$$

where we are taking a $[q\text{-}1/q]$ approximant.

$$\Rightarrow \left(m_0 + m_1 s + m_2 s^2 + \cdots + m_{2q-1} s^{2q-1}\right)\left(1 + b_1 s + b_2 s^2 + \cdots + b_q s^q\right)$$

$$= a_0 + a_1 s + \cdots + a_{q-1} s^{q-1}$$

$$\begin{array}{lll}
s^0: & a_0 = & m_0 \\
s^1: & a_1 = & m_1 + m_0 b_1 \\
s^2: & a_2 = & m_2 + m_1 b_1 + m_0 b_2 \\
 & \vdots & \\
s^{q-1}: & & \\
 & & \\
s^q: & 0 = & m_q + m_{q-1} b_1 + \cdots m_1 b_{q-1} + m_0 b_q \\
s^{q+1}: & 0 = & m_{q+1} + m_q b_1 + \cdots + m_1 b_q \\
s^{q+2}: & 0 = & m_{q+2} + m_{q+1} b_1 + \cdots + m_2 b_q \\
\vdots & & \\
s^{2q-1} & &
\end{array}$$

$$\Rightarrow \begin{bmatrix} m_0 & m_1 & m_2 & \cdots & m_{q-1} \\ m_1 & m_2 & m_3 & \cdots & m_q \\ m_2 & m_3 & m_4 & \cdots & m_{q+1} \\ \vdots & & & & \\ m_{q-1} & m_q & m_{q+1} & \cdots & m_{2q-2} \end{bmatrix} \begin{bmatrix} b_q \\ b_{q-1} \\ b_{q-2} \\ \vdots \\ b_1 \end{bmatrix} = - \begin{bmatrix} m_q \\ m_{q+1} \\ m_{q+2} \\ \vdots \\ m_{2q-1} \end{bmatrix}$$

Can now solve for $b_1, \dots b_q$ and then $a_0, \dots a_{q-1}$.

Note $2q-1$ moments are required to compute a $[q-1/q]$ approximant

Next we can find the roots to determine poles. Alternatively, consider the following

$$F(s) = \sum_{i=1}^{q} \frac{k_i}{(s-p_i)} \qquad \text{for } p_i \text{ distinct}$$

$$= \sum_{i=1}^{q} \frac{-k_i/p_i}{(1-s/p_i)}$$

$$= -\sum_{i=1}^{q} \frac{k_i}{p_i}\left[1 + \frac{s}{p_i} + \frac{s^2}{p_i^2} + \cdots\right]$$

$$= \sum_{i=1} m_i s^{2q-1}$$

$$\therefore \quad m_i = -\sum_{j=1}^{q} \frac{k_j}{p_j^{i+1}}$$

$$\Rightarrow \quad -m_0 = \frac{k_1}{p_1} + \frac{k_2}{p_2} + \cdots + \frac{k_q}{p_q}$$

$$-m_1 = \frac{k_1}{p_1^2} + \frac{k_2}{p_2^2} + \cdots + \frac{k_q}{p_q^2}$$

$$\vdots$$

$$-m_{q-1} = \frac{k_1}{p_1^q} + \frac{k_2}{p_2^q} + \cdots + \frac{k_q}{p_q^q}$$

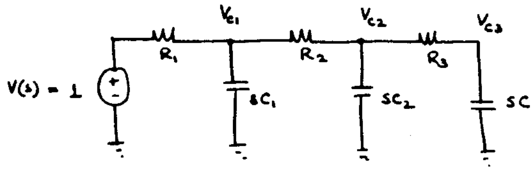
$$\begin{bmatrix} 1/p_1 & 1/p_2 & \cdots & 1/p_q \\ 1/p_1^2 & 1/p_2^2 & \cdots & 1/p_q^2 \\ \vdots & & & \\ 1/p_1^q & 1/p_2^q & \cdots & 1/p_q^q \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ \vdots \\ k_q \end{bmatrix} = -\begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_{q-1} \end{bmatrix}$$

So how do we compute the moments?

Recall These are moments of the impulse response

$v(t) = \delta(t) \rightarrow V(s) = 1$.

Consider the RC Example



Express the capacitor voltages as an infinite power series in s

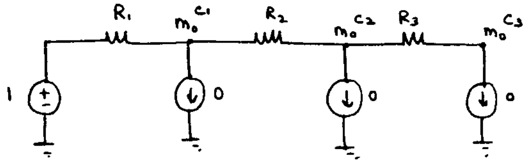
i.e.
$$V^{c1} = m_0^{c1} + m_1^{c1}s + m_2^{c1}s^2 + \cdots$$
$$V^{c2} = m_0^{c2} + m_1^{c2}s + m_2^{c2}s^2 + \cdots$$
$$V^{c3} = m_0^{c3} + m_1^{c3}s + m_2^{c3}s^2 + \cdots$$

The currents through the capacitors are

$$I^{ci} = sC_i\left(V^{ci}\right)$$
$$= sC_i\left(m_0^{ci} + m_1^{ci}s + m_2^{ci}s^2 + \cdots\right)$$

To obtain m_0 we set $s = 0$

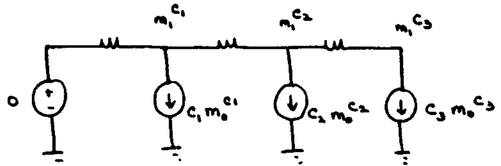
$$\Rightarrow V^{ci} = m_0^{ci}$$
$$I^{ci} = 0$$



Solution of the above circuit gives

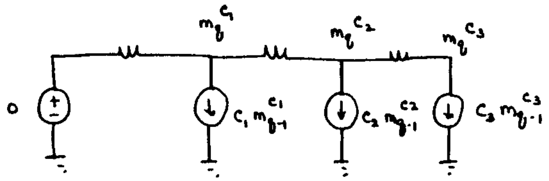
$$m_0^{c_1} = m_0^{c_2} = m_0^{c_3} = 1$$

Next solve for s^1 coefficients i.e. $m_1^{c_i}$.



Since $m_0^{c_i}$ are known, the current source values are known and one can solve for $m_1^{c_i}$ as the node voltages of the above circuit

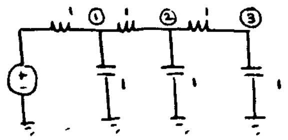
One can proceed in a similar manner to obtain the higher-order moments



Remarks

- The moments are obtained from a dc solution of the network with capacitors replaced by current sources (inductors replaced by voltage sources)
- For each moment calculation the circuit topology remains the same. Only source vector changes
- LU factor circuit matrix and solve using forward / back substitutions with various RHS.
- Once the moments have been calculated, the poles and residues / zeros can be computed as described before.

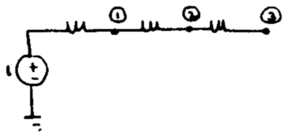
RC Example



m_0

$I^{c_i} = 0$

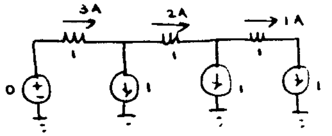
$m_0(1) = 1$
$m_0(2) = 1$
$m_0(3) = 1$



m_1

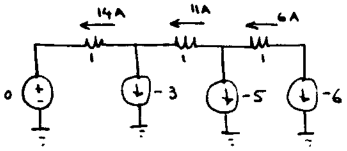
$I^{c_i} = m_0^{c_i} C_i = 1 . 1 = 1$

$m_1(1) = -3$
$m_1(2) = -5$
$m_1(3) = -6$



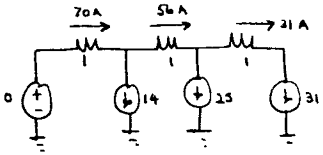
m_2

$m_2(1) = 14$
$m_2(2) = 25$
$m_2(3) = 31$



m_3

$m_3(1) = -70$
$m_3(2) = -126$
$m_3(3) = -157$



Thus we have $V_3(s) = 1 - 6s + 31s^2 - 157s^3$
Now let us obtain the Pade' approximants

[0/1]

$$1 - 6s = \frac{a_0}{1 + b_1 s} \Rightarrow a_0 = 1, \quad b_1 = 6$$

$$\therefore \quad H_{0,1}(s) = \frac{1}{1 + 6s} \quad \text{as before}$$

[1/2] $$1 - 6s + 31s^2 - 157s^3 = \frac{a_0 + a_1 s}{1 + b_1 s + b_2 s^2}$$

$$\Rightarrow 1 + (b_1 - 6)s + (31 + b_2 - 6b_1)s^2 + (-157 + 31b_1 - 6b_2)s^3 = a_0 + a_1 s$$

$$\left.\begin{aligned} a_0 &= 1 \\ a_1 &= b_1 - 6 \\ 0 &= 31 - 6b_1 + b_2 \\ 0 &= -157 + 31b_1 - 6b_2 \end{aligned}\right] \Rightarrow \begin{aligned} b_1 &= 29/5 \\ b_2 &= 19/5 \end{aligned} \Rightarrow a_1 = -\frac{1}{5}$$

Note: These are the same results as we obtained on page 202.

General Method

We have $$(G + sC)x = b$$ $$y = l^T x$$

Let $\underline{x} = \underline{m_0} + \underline{m_1} s + \underline{m_2} s^2 + ----$

For impulse response $b = \underline{1}$

$$\therefore (G + sC)(m_0 + m_1 s + m_2 s^2 + \cdots) = b$$

$$\Rightarrow \quad G m_0 = b$$
$$(G m_1 + C m_0) = 0$$
$$\vdots$$
$$(G m_i + C m_{i-1}) = 0$$

⇒ Solve $G m_0 = b$ for m_0 by LU factoring G and Forward/Back Solves

Solve $G m_i = -C m_{i-1}$ by Forward/Back solves

⇒ moments m_i can be determined

Observations

$$m_0 = G^{-1} b$$
$$m_1 = G^{-1} C G^{-1} b$$
$$m_2 = G^{-1} C G^{-1} C G^{-1} b$$
$$\vdots$$
$$m_i = (G^{-1} C)^i G^{-1} b$$

Remarks

- $(G^{-1})^i$ can become numerically ill-conditioned when G has a large spread in eigenvalues
- In practice this approach (AWE) is limited to about 10 dominant poles with reasonable precision.

What do you do if you need more than 10 dominant poles?

Use a method that computes the Padé approximation by another method. i.e. do not explicitly compute the moments

⇒ Padé Via Lanczos (PVL)

(Read Paper by Feldmann & Freund, IEEE Trans. CAD, 1995)

Problem with Padé approximation (unstable reduced systems)

Suppose $$H(s) = \frac{k_1}{s - p_1} + \frac{k_2}{s - p_2}$$

and $$H_{0,1}(s) = \frac{k}{s - p}$$

$$\Rightarrow \frac{k}{p} = \frac{k_1}{p_1} + \frac{k_2}{p_2} \quad \text{and} \quad \frac{k}{p^2} = \frac{k_1}{p_1^2} + \frac{k_2}{p_2^2}$$

$$\therefore p = \frac{k_1/p_1 + k_2/p_2}{k_1/p_1^2 + k_2/p_2^2}$$

Suppose $$\left.\begin{aligned} k_1 &= 3, & k_2 &= -8 \\ p_1 &= -1, & p_2 &= -2 \end{aligned}\right\} \Rightarrow p = \frac{3/{-1} + {-8}/{-2}}{3/1 + {-8}/4} = 1$$

i.e. a RHP pole even though the original system was stable