CAD and Circuit Simulators

劉 尙 大 敎授

전 자 공 학 부

Kyungpook National University

Computer Aided Design (CAD)

- ❑ **Design** = key word
- ❑ A **working chip** NOT just a working program
- ❑ **Evolutionary** not revolutionary
- ❑ Close coupling with designers
- ❑ 20 lines of bug-free code per day

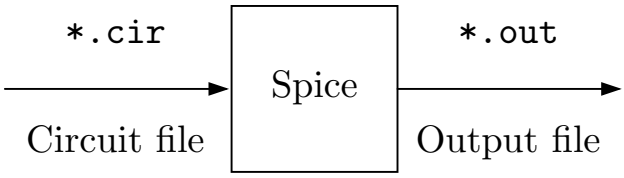
CAD Areas for Integrated Circuits

- Design verification
  - Simulation: circuit, logic, mixed
  - Functional verification
  - Timing verification
  - Physical verification
- Circuit synthesis
  - High-level synthesis
  - Combinational logic optimization
  - Sequential logic optimization
  - Layout synthesis
  - Analog circuit synthesis

- Circuit testing
  - Fault simulation
  - Automatic test pattern generation
  - Sequential test
  - Buil-in self test (BIST)
  - Analog test
- Design representation
  - Domain: behavioral, structural, physical
  - Digital HDL: VHDL, ELLA, Verilog HDL, Hardware C
  - Analog HDL: VHDL-AMS, AnaVHDL, AHDL, Verilog-A
  - Design database

Circuit Simulators

- Spice: Simulation Program with Integrated Circuit Emphasis
  - a general-purpose circuit simulator
  - *de facto* industrial standard for computer-aided circuit analysis
  - `Spice2g6` (1980), `Spice3f5` (1994), `PSpice` (1984), `HSpice`
- Simulation flow



- Output postprocessing using `probe`

Analysis Types of Spice

- DC analysis
  - operating point (`.op`)
  - dc sweep (`.dc`)
  - small-signal transfer function (`.tf`)
  - small-signal sensitivity (`.sens`)
- Transient analysis
  - time domain response (`.tran`)
  - Fourier analysis (`.four`)
- AC analysis
  - small-signal frequency response (`.ac`)
  - noise analysis (`.noise`)

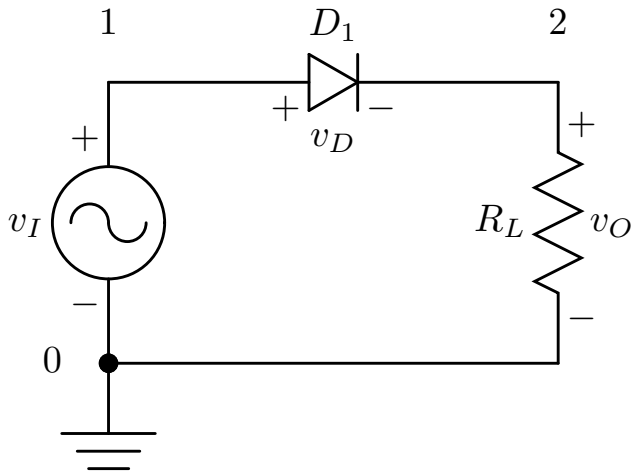
Spice Input File

❑ Circuit file format

```
Title statement
    Analysis requests
    Output requests
    Circuit description
        Power supplies
        Signal sources
        Element descriptions
        Model statements
+   Continued line
*   Comment line
.end
```


A Diode Circuit

❑ Circuit diagram



❑ Node number: 0 for the ground node, 1 to 9999

Example of Circuit File

❑ diode.cir

```
A diode circuit
.options reltol=0.001, abstol=1pA, vntol=1uV
.dc vI 0 1v 0.01v
.print dc v(2) (.probe)
*      step  stop start max
*.tran 0.01m 4m   0m    0.001m
vI 1 0 dc 1v ac 1v 0
+      sin (0v 5v 1kHz 0s 0)
D1 1 2 D1N4184
RL 2 0 1k
.model D1N4184 D (Is=0.1p Rs=16 Cjo=2p Tt=12n Bv=100)
.end
```

❑ PSpice: File->Open, File->Run Probe, Trace->Add

Basic Concepts of Circuit Simulators

❑ Reliable device models

❑ Systematic formulation

❑ Solution by Newton-Raphson method

❑ Sparse matrix techniques

❑ Circuit analysis methods

- Sparse Tableau Analysis (STA): ASTAP
- Modified Nodal Analysis (MNA): SPICE

❑ Stable integration formulae

❑ Automatic stepsize control

Sparse Tableau Analysis

❏ KCL equations

$$\begin{aligned} \mathbf{Ai} &= \mathbf{0}, \quad \mathbf{A} = \text{incidence matrix} \\ a_{ij} &= +1 \text{ if node } i \text{ is the positive node of branch } j \\ \mathbf{i} &= \text{branch current vector} \end{aligned}$$

❏ KVL equations

$$\begin{aligned} \mathbf{e} - \mathbf{A}^t\mathbf{v} &= \mathbf{0} \\ \mathbf{e} &= \text{branch voltage vector} \\ \mathbf{v} &= \text{node voltage vector} \end{aligned}$$

❏ Branch equations

$$\mathbf{Z}_b\mathbf{i} - \mathbf{Y}_b\mathbf{e} = \mathbf{S} = \text{source vector}$$

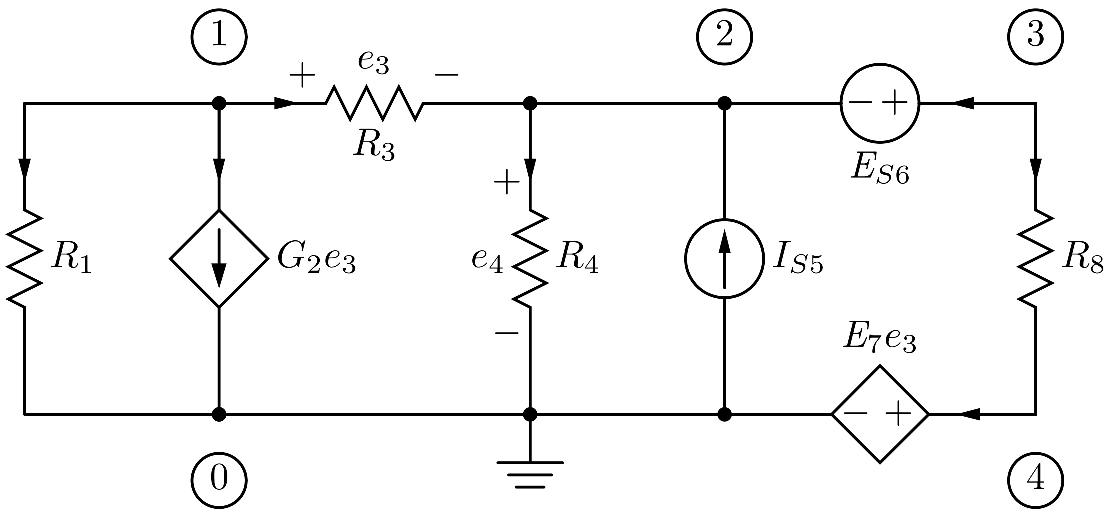
❏ STA equations

$$\begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & -\mathbf{A}^t \\ \mathbf{Z}_b & -\mathbf{Y}_b & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{i} \\ \mathbf{e} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{S} \end{bmatrix}$$

❏ Features

- Application to **any** circuit
- Assembly by **inspection**
- **Sparse** coefficient matrix
- Sophisticated programing
- Solutions to all node voltages, all branch voltages, and all branch currents → large matrix size

❑ Example: a linear resistive circuit



❑ STA equations

$$
\left[\begin{array}{cccccccccccccccccccc}
1&1&1\\
&&-1&1&-1&-1\\
&&&&&1&0&1\\
&&&&&&1&-1\\
&&&&&&&&1&&&&&&&&-1\\
&&&&&&&&&1&&&&&&&-1\\
&&&&&&&&&&1&&&&&&-1&1\\
&&&&&&&&&&&1&&&&&&-1\\
&&&&&&&&&&&&1&&&&&1\\
&&&&&&&&&&&&&1&&&&1&-1\\
&&&&&&&&&&&&&&1&&&&0&-1\\
&&&&&&&&&&&&&&&1&&&-1&1\\
R_1&&&&&&&&-1\\
&1&&&&&&&&&-G_2\\
&&R_3&&&&&&&&-1\\
&&&R_4&&&&&&&&-1\\
&&&&1\\
&&&&&&&&&&&&&1\\
&&&&&&&&&&E_7&&&&-1\\
&&&&&&&R_8&&&&&&&&-1
\end{array}\right]
\left[\begin{array}{c} i_1\\ i_2\\ i_3\\ i_4\\ i_5\\ i_6\\ i_7\\ i_8\\ e_1\\ e_2\\ e_3\\ e_4\\ e_5\\ e_6\\ e_7\\ e_8\\ v_1\\ v_2\\ v_3\\ v_4 \end{array}\right]
=
\left[\begin{array}{c} 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ I_{S5}\\ E_{S6}\\ 0\\ 0 \end{array}\right]
$$

Modified Nodal Analysis

❑ KCL equations: node equations

$$\begin{aligned} \mathbf{Yv} &= \mathbf{I}_s = \text{current source vector} \\ \mathbf{Y} &= \text{node admittance matrix} \\ \mathbf{v} &= \text{node voltage vector} \end{aligned}$$

❑ KVL equations for voltage sources

❑ MNA equations

$$\begin{bmatrix} \mathbf{Y} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{i} \end{bmatrix} = \begin{bmatrix} \mathbf{J} \\ \mathbf{E} \end{bmatrix}$$

❑ Features

- Application to **any** circuit
- Assembly by **inspection**
- **Sparse** coefficient matrix
- **Closeness** to **Y** of nodal analysis: most nonzero diagonal entries
- Solutions to all node voltages and currents of voltage sources

❑ Example: the linear resistive circuit

1. Write KCL at the nodes

   ①: $i_1 + i_2 + i_3 = 0$, ②: $-i_3 + i_4 - i_5 - i_6 = 0$

   ③: $i_6 + i_8 = 0$, ④: $i_7 - i_8 = 0$

2. Substitute branch currents and voltages with node voltages

①: $v_1/R_1 + G_2(v_1 - v_2) + (v_1 - v_2)/R_3 = 0$

②: $-(v_1 - v_2)/R_3 + v_2/R_4 - i_6 = I_{S5}$

③: $i_6 + (v_3 - v_4)/R_8 = 0$, ④: $i_7 - (v_3 - v_4)/R_8 = 0$

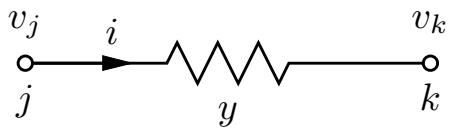
3. Append the branch equations of branch voltage sources
   - Branch 6: $v_3 - v_2 = E_{S6}$
   - Branch 7: $v_4 - E_7(v_1 - v_2) = 0$
4. MNA equations

$$\begin{bmatrix} \frac{1}{R_1} + G_2 + \frac{1}{R_3} & -G_2 - \frac{1}{R_3} & 0 & 0 & 0 & 0 \\ -\frac{1}{R_3} & \frac{1}{R_3} + \frac{1}{R_4} & 0 & 0 & -1 & 0 \\ 0 & 0 & \frac{1}{R_8} & -\frac{1}{R_8} & 1 & 0 \\ 0 & 0 & -\frac{1}{R_8} & \frac{1}{R_8} & 0 & 1 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ -E_7 & E_7 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ i_6 \\ i_7 \end{bmatrix} = \begin{bmatrix} 0 \\ I_{S5} \\ 0 \\ 0 \\ E_{S6} \\ 0 \end{bmatrix}$$

MN Formulation by Inspection

❑ Node-by-node (person), element-by-element (computer) basis

❑ KCL equations for a conductance element

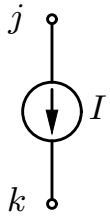
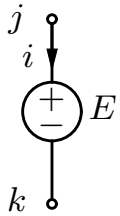
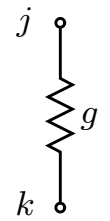


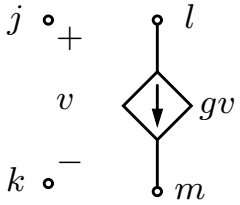
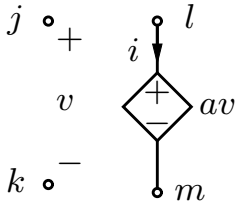
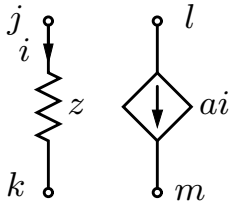
node j : $\quad \cdots + i \cdots = \cdots + yv_j - yv_k \cdots = \cdots$

node k : $\quad \cdots - i \cdots = \cdots - yv_j + yv_k \cdots = \cdots$

❑ Element stamps: row (node current), column (node voltage)

$$\begin{matrix} j \\ k \end{matrix} \begin{bmatrix} y & -y \\ -y & y \end{bmatrix}$$

Element	Symbol	Matrix	Equation
Current source		$\begin{matrix} j \\ k \end{matrix}\begin{bmatrix} -I \\ I \end{bmatrix}$ Source vector	$i_j = I$ $i_k = -I$
Voltage source		$\begin{matrix} & j & k & l \\ j \\ k \\ m \end{matrix}\begin{bmatrix} & & 1 \\ & & -1 \\ 1 & -1 & \end{bmatrix}\begin{bmatrix} \\ \\ E \end{bmatrix}$ Source vector	$i_j = i$ $i_k = -i$ $v_j - v_k = E$
Admittance		$\begin{matrix} & j & k \\ j \\ k \end{matrix}\begin{bmatrix} g & -g \\ -g & g \end{bmatrix}$	$i_j = gv_{jk}$ $i_k = -gv_{jk}$

Element	Symbol	Matrix	Equation
VCCS		$\begin{matrix} & j & k \\ l \\ m \end{matrix}\begin{bmatrix} g & -g \\ -g & g \end{bmatrix}$	$i_l = gv_{jk}$ $i_m = -gv_{jk}$
VCVS		$\begin{matrix} & j & k & l & m & n \\ l \\ m \\ p \end{matrix}\begin{bmatrix} & & & & 1 \\ & & & & -1 \\ -a & a & 1 & -1 & \end{bmatrix}$	$i_l = i$ $i_m = -i$ $v_{lm} = av_{jk}$
CCCS		$\begin{matrix} & j & k & l & m & n \\ j \\ k \\ l \\ m \\ p \end{matrix}\begin{bmatrix} & & & & 1 \\ & & & & -1 \\ & & & & a \\ & & & & -a \\ 1 & -1 & & & -z \end{bmatrix}$	$i_j = -i_k = i$ $i_l = -i_m = ai$ $v_j - v_k = zi$

Algorithms for Solving Linear Equations

❑ Equation

$$\mathbf{Ax} = \mathbf{b}$$

❑ Methods

- Iterative methods: Gauss-Jacobi, Gauss-Seidel, SOR
- Direct methods: Gaussian elimination, LU decomposition

❑ Gauss-Jacobi method

$$\mathbf{Ax} = (\mathbf{L} + \mathbf{D} + \mathbf{U})\mathbf{x} = \mathbf{b}$$

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

❑ Example for solving a matrix equation

$$\begin{bmatrix} 4 & 1 & 2 \\ 1 & 3 & 1 \\ 1 & 2 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 16 \\ 10 \\ 12 \end{bmatrix} \tag{1}$$

$$\begin{aligned} \mathbf{x}^0 &= [\,0 \;\; 0 \;\; 0\,]^t \\ \mathbf{x}^1 &= [\,4 \;\; 10/3 \;\; 12/5\,]^t \\ \mathbf{x}^2 &= [\,59/30 \;\; 18/15 \;\; 4/15\,]^t \\ \mathbf{x}^3 &= [\,107/30 \;\; 233/90 \;\; 229/150\,]^t \end{aligned}$$

❑ Gauss-Seidel method

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

❏ Gaussian elimination: backward substitution

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

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_n \end{bmatrix}$$

❏ LU decomposition: forward, backward substitution

$$\mathbf{Ax} = \mathbf{LUx} \equiv \mathbf{Ly} = \mathbf{b}$$

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

❏ Pivoting: interchanging rows and/or columns to bring a nonzero element in diagonal element for solution accuracy

❏ Exploiting sparsity

$$\begin{aligned} \text{CPU time}: \quad & O(n^3) \rightarrow O(n^{1.2}) \\ \text{Memory}: \quad & O(n^2) \rightarrow O(n) \end{aligned}$$

Sparse Matrix Package

```
/*
 *  sparse.c
 *  Test routine for the sparse matrix package
 *
 *  Author:
 *     Prof. Yu Sang Dae
 *     Kyungpook National University
 *
 *  Emphasis is on demonstrating how to use the basic capability of
 *  the matrix routines.
 *
 */

#include <stdio.h>
#include <math.h>
#include <sys/types.h>
#include "spconfig.h"
```



```
#include "spmatrix.h"
#include "spdefs.h"
#include "sperror.h"

main( argc, argv )
int   argc;
char *argv[];
{
    char *Matrix;
    int  i, last, size, complex, error;
    double RHS[10], Solution[10], iRHS[10], iSolution[10];
    RealNumber *pElement;

    size    = 3;
    complex = 1;

    Matrix = spCreate( size, complex, &error );
    spErrorMessage( Matrix, stdout, NULL );
    spClear( Matrix );
```


```
    error = spAddComplexElement( Matrix, 1, 1, 1.0, 1.0 );
    error = spAddComplexElement( Matrix, 2, 2, 1.0, 2.0 );
    error = spAddComplexElement( Matrix, 3, 3, 2.0, 2.0 );
    spErrorMessage( Matrix, stdout, NULL );

    /* spPrint( Matrix, PrintReordered, Data, Header ) */

    spPrint( Matrix, 0, 1, 0 );
    error = spFactor( Matrix );

    /* spPrint( Matrix, 1, 1, 0 ); */

    spErrorMessage( Matrix, stdout, NULL );

    RHS[1]  = 1.0; RHS[2]  = 1.0; RHS[3]  = 1.0;
    iRHS[1] = 0.0; iRHS[2] = 0.0; iRHS[3] = 0.0;

    spSolve( Matrix, RHS, Solution, iRHS, iSolution );
```



```
    printf( "Solution\n" );
    if ( complex ) for (i = 1; i <= size; i++)
        printf( "%-16.9g   %-16.9gj\n", Solution[i], iSolution[i]);
    else for (i = 1; i <= size; i++)
        printf( "%-16.9g\n", Solution[i]);
    spDestroy( Matrix );
}

spAddComplexElement( Matrix, Row, Col, Real, Imag )
char *Matrix;
int Row, Col;
double Real, Imag;
{
    register RealNumber *pElement;
    pElement = spGetElement( Matrix, Row, Col );
    *pElement       += Real;
    *(pElement + 1) += Imag;
    return spError( Matrix );
}
```


Solution of Nonlinear Algebraic Equations

❏ Equation

$$f(x) = 0$$

❏ Taylor expansion

$$f(x^*) = 0 = f(x_k) + f'(x_k)(x^* - x_k) + R(x^* - x_k)$$

❏ Newton method: Newton-Raphson method

$$f'(x_k)\Delta x_k = -f(x_k)$$

$$x_{k+1} = x_k + \Delta x_k = x_k - f(x_k)/f'(x_k)$$

❏ Convergency: if x_k is sufficiently close to the solution, the NR method has quadratic convergence.

$$\epsilon_{k+1} = |x_{k+1} - x^*| \leq \beta\epsilon_k^2$$

❏ Convergence criteria: relative error tolerence E_R, absolute error tolerence E_A

$$|\Delta x_k| = |x_{k+1} - x_k| < \epsilon = E_R|x_{k+1}| + E_A$$

❏ Global methods: convergence from almost any starting point

- Controlled NR method

$$x_{k+1} = x_k + \alpha_k \Delta x_k, \quad 0 < \alpha \leq 1$$

- Charge-up method: capacitor $\|$ nonlinear current source, inductor + nonlinear voltage source

$$\frac{dx_{n+1}}{dt} = \frac{x_{n+1} - x_n}{\Delta t}$$

- Optimization-based method

$$\text{Minimize } f = \frac{1}{2}\mathbf{F} \cdot \mathbf{F}$$

- Damping: avoiding overflow due to exponential dependency

$$\Delta x = \text{sign}(\Delta x)\frac{\ln(1 + a|\Delta x|)}{a}$$

- Source stepping algorithm
- $G_{\min}$ stepping algorithm

❏ Multidimensional NR method: Jacobian matrix $\mathbf{J}$

$$\mathbf{J}\Delta\mathbf{x} = -\mathbf{F}, \quad J_{ij} \equiv \frac{\partial F_i}{\partial x_j}$$

❏ Convergency of MNR method

- $\mathbf{J}$ is nonsingular
- $\mathbf{J}$ is Lipschitz continuous

$$\| \mathbf{J}(\mathbf{x}^*)^{-1}[\mathbf{J}(\mathbf{x}) - \mathbf{J}(\mathbf{x}^*)] \| \leq \beta \| \mathbf{x} - \mathbf{x}^* \|$$

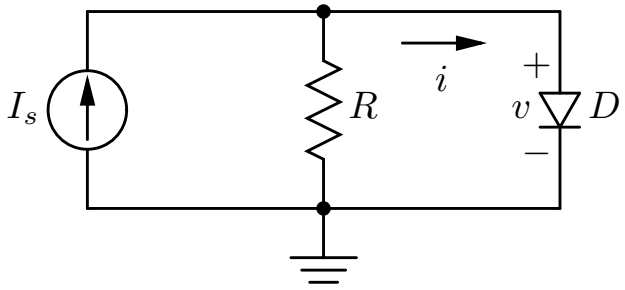
Stamps of Nonlinear Elements

❏ Δx : x formulation

$$\mathbf{J}\Delta\mathbf{x} = -\mathbf{F} \quad : \quad \mathbf{J}_k\mathbf{x}_{k+1} = \mathbf{J}_k\mathbf{x}_k - \mathbf{F}_k$$

❏ Companion models: linearized equivalent circuits for nonlinear and charge-storage elements

$$f_1(i,\, v) = i - i_D(v) = 0, \quad f_2(i,\, v) = i + \frac{v}{R} - I_s = 0$$

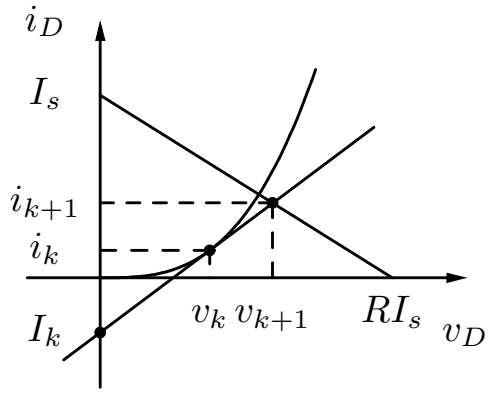


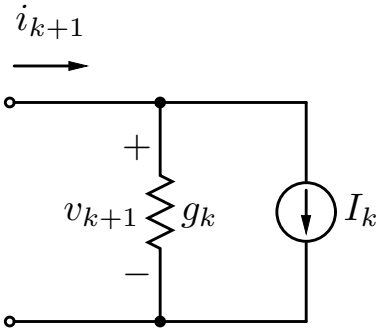
Companion Model of Diodes

❑ A linearized circuit for the nonlinear diode equation

$$\begin{bmatrix} 1 & -g_k \\ 1 & 1/R \end{bmatrix} \begin{bmatrix} i_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & -g_k \\ 1 & 1/R \end{bmatrix} \begin{bmatrix} i_k \\ v_k \end{bmatrix} - \begin{bmatrix} f_{1k} \\ f_{2k} \end{bmatrix}$$

$$i_{k+1} = g_k v_{k+1} + i_k - g_k v_k - f_{1k} \equiv g_k v_{k+1} + I_k = -v_{k+1}/R + I_s$$





Stamps of Nonlinear Current Sources

❑ i is not part of $\mathbf{x}$: $f_j = \cdots + i(\mathbf{x}) \cdots$, $f_k = \cdots - i(\mathbf{x}) \cdots$

	x_1	x_2	$\cdots$	x_n	$\cdots$	RHS
j	$\partial i/\partial x_1$	$\partial i/\partial x_2$	$\cdots$	$\partial i/\partial x_n$	$\cdots$	$-i(\mathbf{x})$
k	$-\partial i/\partial x_1$	$-\partial i/\partial x_2$	$\cdots$	$-\partial i/\partial x_n$	$\cdots$	$+i(\mathbf{x})$

❑ i is part of $\mathbf{x}$: $f_j = \cdots + i_l(\mathbf{x})$, $f_k = \cdots - i_l(\mathbf{x})$, $f_l(\mathbf{x}) = i(\mathbf{x}) - i_l = 0$

	x_1	x_2	$\cdots$	x_n	$\cdots$	i_l	$\cdots$	RHS
j						$+1$		
k						-1		
l	$\partial i/\partial x_1$	$\partial i/\partial x_2$	$\cdots$	$\partial i/\partial x_n$	$\cdots$	-1	$\cdots$	$-f_l(\mathbf{x})$

❑ AC analysis

$$\mathbf{YV} = (\mathbf{G} + s\mathbf{C})\mathbf{V} = \mathbf{I}$$

```
/*
 *      main.c
 *         iCADs
 *         Integrated Circuit Analysis and Design by Simulation
 *
 *         Written by Prof. Yu Sang Dae
 *         Copyright (C) 1996 Kyungpook National University.
 *         1996. 04. 17. - 1996. 08. 18 initial version.
 */

main( argc, argv )
int argc;
char *argv[];
{
    Circuit *Ckt;
    if ( argc < 2 ) {
        printf( "Usage: icads file\n" );
        exit( 1 );
    }
```



```
    Ckt = Malloc( Circuit );
    InitSystem( Ckt );
    ReadFile( argv[1], Ckt );
    ParseCommand( Ckt );

    MakeModelTable( Ckt );
    MakeSubcktTable( Ckt );
    MakeFlatCircuit( Ckt );

    ParseDevice( Ckt );
    SetupMatrix( Ckt );
    Analysis( Ckt );

    FreeCircuit( Ckt );
    Free( Ckt );
    return( 0 );
}
```


Homework

(1) Write MNA equations for the linear resistive circuit using element stamps.

(2) Solve the matrix equation (1) using the given sparse matrix package.

(3) Derive a damping equation for the diode current $i = I_s(e^{v/V_T} - 1)$.

$$\Delta v = V_T \ln(1 + \Delta v/V_T) \qquad (\Delta v > 0)$$

(4) Find the element stamp of nMOS transistors.

References

[1] F. H. Branin et al., "ECAP II – A new electronic circuit analysis program", *IEEE J. Solid-State Circuits*, vol. SC-6, no. 4, pp. 146–166, 1971.

[2] L. W. Nagel and R. A. Rohrer, "Computer analysis of nonlinear circuits excluding radiation (CANCER)", *IEEE J. Solid-State Circuits*, vol. SC-6, no. 4, pp. 166–182, 1971.

[3] G. D. Hachtel et al., "The sparse tableau approach to network analysis and design", *IEEE Trans. on Circuit Theory*, vol. CT-18, pp. 101–108, 1971.

[4] C. W. Ho et al., "The modified nodal approach to network analysis", *IEEE Trans. on Circuits and Systems*, vol. CAS-22, No. 6, pp. 504–509, 1975.

[5] D. A. Zein, "Solution of a set of nonlinear algebraic equations for general-purpose CAD programs", *IEEE Circuits and Devices Magazine*, vol. 1, pp. 7–20, 1985.

[6] E. Ngoya, J. Rousset, and J. J. Obregon, "Newton-Raphson iteration speed-up algorithm for the solution of nonlinear circuit equations in general-purpose CAD programs", *IEEE Tran. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, pp. 638–644, 1997.

[7] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*, Van Nostrand Reinhold, 1983.

[8] W. J. McCalla, *Fundamentals of Computer-Aided Circuit Simulation*, Kluwer Academic Publishers, 1988.