HW # 1 Due today

HW # 2 Assigned and due Wed. Oct 26
- Interfacing to Sparse 1.4
- Solution of linear circuits
- Setup of Newton loop with linear circuit elements.

Sparse Matrix Techniques

Independent voltage source

\[ \begin{bmatrix} A & I_{br} \\ B & 0 \end{bmatrix} \begin{bmatrix} V_A \\ V_B \end{bmatrix} = \begin{bmatrix} +1 \\ -1 \end{bmatrix} + \begin{bmatrix} \mathbf{d} \\ \mathbf{v}_s \end{bmatrix} \]

Damped Newton Method

\[ f(x) \]

\[ J_F \text{ is bounded} \]
Diode (pn junction) Limiting

Ensure that

\[ V_{im}^{K+1} \leq V_{im} \leq V_{im}^{K+1} = V_{im}^{K+1} + 10V_{th} \tan\left(\frac{V_{im}^{K+1} - V_{im}^{K}}{10V_{th}}\right) \]

SPICE PN Junction limiting (PNJLIM)

\[
\begin{align*}
\text{For the diode:} & \\
i &= I_s \left( e^{V_{th}/V_m} - 1 \right) \\
\frac{\partial i}{\partial V} &= I_s \frac{V_{th}/V_m}{e^{V_{th}/V_m}} \\
\text{Linearization at iteration } k+1 & \\
\hat{i} &= I_s \left( e^{V_{th}/V_m} - 1 \right) + \frac{I_s}{V_m} V_{th} \left( V_{im}^{K+1} - V_{im}^{K} \right) \\
\hat{i} &= I_s \left( e^{V_{th}/V_m} - 1 \right) + \frac{I_s}{V_m} V_{th} \ln \left( \frac{\hat{i}}{I_s} + 1 \right) \\
&= V_m \ln \left[ e^{V_{th}/V_m} + \frac{e^{V_{th}/V_m} (V_{im}^{K+1} - V_{im}^{K})}{V_m} \right]
\end{align*}
\]
\[ V_{th} \ln \left[ e^{v/kT} \left( 1 + \frac{v^{k+1} - v^K}{v_{th}} \right) \right] \]

\[ V_{lim}^{k+1} = V^K + V_{th} \ln \left[ \frac{v^{k+1} - v^K}{v_{th}} + 1 \right] \]

This limiting formula is applied only when \[ v^{k+1} > V_{init} \text{, and } v^{k+1} > v^K \]

\[ V_{th} \ln \left( \frac{v_{th}}{2J_3} \right) \leftarrow \text{maximum curvature} \]

What happens if \( v^K \) are negative?

\[ \frac{di}{dv} = \frac{I_S}{v_{th}} e^{v/v_{th}} \approx 0 \]

\[ i = I_S \left( e^{v/v_{th}} - 1 \right) + G_{min} v \quad G_{min} = 10^{-12} \]

\[ \frac{di}{dv} = \frac{I_S}{v_{th}} e^{v/v_{th}} + G_{min} \]

In SPICE 2 \( v < -5v_{th} \)

\[ i = -I_S + G_{min} v \]

\[ \frac{di}{dv} = G_{min} \]

**Other limiting schemes in SPICE**

- **FETLIM** (limits \( V_{gs} \))
- **LIMVDS** (limits \( V_{ds} \))

**Usage for Sparse 1.4:**

```
/sparse mato
```

on unix:

```
mato
   -- mat4
```
Nonlinear equation solution

1) Newton's method
2) Optimization methods
3) Continuation methods

Optimization-based methods

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

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

$\Rightarrow f(x) = 0$

Steepest descent method

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

Newton's method  

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

Continuation methods

1) Source stepping

When sources are at zero: $x = 0$

Ramp up source voltage by a small amount (say $\delta_i$) then $x(\delta_i)$ is close to $x(0)$

Recall "Close enough"

$\Rightarrow$ Expect Newton's method to converge

Procedure:

Source vector $S$

Define a parameter $\lambda$: $0 \leq \lambda \leq 1$

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

Vary $\lambda$ from 0 to 1 and solve circuit for each $\lambda$

$\lambda = 1$ is the solution for the original problem
\( \chi(\lambda) \) i.e. the solution for source \( \lambda S \) should be sufficiently smooth.

**General Case**

\[ f(x) = 0 \quad \text{original problem} \]

Solve \( \tilde{f}(\chi(\lambda), \lambda) = 0 \)

Where:
1. \( \tilde{f}(\chi(1), 1) = f(x) \)
2. \( \tilde{f}(\chi(0), 0) = 0 \) is easy to solve
3. \( \chi(\lambda) \) is sufficiently smooth

**Simple scheme:**

Embedding function:

\[ \tilde{f}(\chi(\lambda), \lambda) = \lambda f(x(\lambda)) + (1-\lambda) x(\lambda) \]

For \( \lambda = 0 \):

\[ \tilde{f}(\chi(0), 0) = x(0) = 0 \]

For \( \lambda = 1 \):

\[ \tilde{f}(\chi(1), 1) = f(x(1)) = 0 \]

Application of Newton's method

\[ \frac{\partial \tilde{f}}{\partial \chi} = \lambda \frac{\partial f}{\partial \chi} + (1-\lambda) I \]

**Circuit interpretation**

\[ \frac{\partial \tilde{f}}{\partial \lambda} = f(x(\lambda)) - \chi(\lambda) \]
\[\tilde{f}(x(\lambda), \lambda) = 0\]

\[
\tilde{\frac{\partial f}{\partial x}} \frac{\partial x}{\partial \lambda} + \tilde{\frac{\partial f}{\partial \lambda}} = 0
\]

\[
\frac{\partial x}{\partial \lambda} = \left[\tilde{\frac{\partial f}{\partial x}}\right]^{-1} \left(\tilde{\frac{\partial f}{\partial x}}\right)
\]

\[s_0 = \sqrt{(\delta x)^2 + (\delta \lambda)^2}\]

Parameterize \(x, \lambda\) in terms of the arc length

\[x(\sigma), \lambda(\sigma)\]

\[0 \leq \sigma \leq \sigma_0\]

\[\int_0^{\sigma} (x(\sigma), \lambda(\sigma)) = 0\]

\[\|\delta x\|^2 + (\delta \lambda)^2 = s_0^2\]

\[
\begin{bmatrix}
\tilde{\frac{\partial f}{\partial x}} & \tilde{\frac{\partial f}{\partial \lambda}} \\
2(\lambda - \lambda_{prev}) & 2(\lambda - \lambda_{prev})
\end{bmatrix}
\begin{bmatrix}
x^{k+1} - x^k \\
\lambda^{k+1} - \lambda^k
\end{bmatrix}
= \begin{bmatrix}
-\tilde{f}(x(\lambda), \lambda^k)
\end{bmatrix}
\]

**Exact arc length**

\[\|\dot{x}(\sigma)\|^2 + |\dot{\lambda}(\sigma)|^2 = 1\]

HOMPACK homotopy method
Calling Sparse 1.4

/*Declaration of cktMatrix*/
char * cktMatrix;

/* setup circuit matrix*/
cktMatrix = spCreate( NumEqns, 0, &error );
if( error IS spNO_MEMORY ) {
    printf( "\n: --- NO MEMORY ---" );
    exit( -1 );
}

/* compute DC solution*/
/* first Factor the matrix and then Forward/Back solve*/
error = spFactor( cktMatrix ); /* LU factorization; Pivoting*/
if( foundError(error) ) {
    exit( -1 );
}

spSolve(cktMatrix, Rhs, Sol ); /* Forward/Back Solve*/

Res.h

typedef struct resistor{
    char *name;  /* pointer to character string naming this instance*/
    int pNode; /* number of positive node of resistor*/
    int nNode; /* number of negative node of resistor*/
    double value; /* resistance*/
    double conduct; /* conductance*/
    double *pn1n1;   /* pointer to sparse-matrix location (pNode, pNode)*/
    double *pn1n2;   /* pointer to sparse-matrix location (pNode, nNode)*/
    double *pn2n2;   /* pointer to sparse-matrix location (nNode, nNode)*/
    double *pn2n1;   /* pointer to sparse-matrix location (nNode, pNode)*/
} resistor;

Setup

void setupRes(Matrix, Res, numRes)
char *Matrix;
resistor *Res[];
int numRes;
{
    int i, n1, n2;
    resistor *inst;
    for(i = 1; i <= numRes; i++) {
        inst = Res[i];
        inst->conduct = 1.0/inst->value;
        n1 = inst->pNode;
        n2 = inst->nNode;
        /* setup matrix and pointers*/
        inst->pn1n1 = spGetElement(Matrix, n1, n1);
        inst->pn1n2 = spGetElement(Matrix, n1, n2);
        inst->pn2n2 = spGetElement(Matrix, n2, n2);
        inst->pn2n1 = spGetElement(Matrix, n2, n1);
    }
}

Load

void loadRes(Matrix, Rhs, Res, numRes)
char *Matrix;
double *Rhs;
resistor *Res[];
int numRes;
{
    int i;
    resistor *inst;
    double conduct;
    for(i = 1; i <= numRes; i++) {
        inst = Res[i];
        conduct = inst->conduct;
        /* load matrix*/
        *(inst->pn1n1) += conduct;
        *(inst->pn1n2) -= conduct;
        *(inst->pn2n2) += conduct;
        *(inst->pn2n1) -= conduct;
    }
}
**Multidimensional Newton Method Algorithm**

Newton Algorithm for Solving $F(x) = 0$

$x^0 = \text{Initial Guess, } k = 0$

Repeat 

1. Compute $F(x^k), J_F(x^k)$
2. Solve $J_F(x^k)(x^{k+1} - x^k) = -F(x^k)$ for $x^{k+1}$
3. $k = k + 1$

Until $\|x^{k+1} - x^k\|, \|f(x^{k+1})\|$ small enough

From: A. Nardi

**Multidimensional Newton Method Convergence**

Local Convergence Theorem

If

1) Inverse is bounded

$$\|J^{-1}_F(x^k)\| \leq \beta$$  (Inverse is bounded)

2) Derivative is Lipschitz Continuous

$$\|J_F(x) - J_F(y)\| \leq \ell \|x - y\|$$  (Derivative is Lipschitz Cont)

Then Newton’s method converges given a sufficiently close initial guess (and convergence is quadratic)

From: A. Nardi

**The Newton Loop in DC Analysis**

```
choose x^0

Linearize Nonlinear Elements

Assemble Circuit Equations of Complete Network

Solve

No

Converged ?

Yes

J(x^{(k+1)})

J(x^{(k)})x^{(k+1)} = f(x^{(k)}) - J(x^{(k)})x^{(k)}

Solve linear equations (spFactor, spSolve)

\[|x^{(k+1)} - x^k| \leq \varepsilon_A + \varepsilon_R \min(|x^{(k+1)}|, |x^{(k)}|)\]

On a per unknown basis
```

**Practical Considerations**

- Device model equations must be continuous with continuous derivatives
  - Not all models have this property
    - Recent shift to $C^\infty$ models – correct by construction
  - Be cautious of designer/user-supplied models

- Check for floating nodes
  - If a node is disconnected, then $J(x)$ is singular

- Provide a good initial guess for $x^{(0)}$
  - This is easy for digital circuits but difficult for analog circuits
Application of NR to Circuit Equations
Companion Network – MNA Stamp

\[ \begin{align*}
+ & \quad e^+ \quad e^- \quad \text{RHS} \\
+ & \quad G_0 \quad -G_0 \quad -I_d \\
- & \quad -G_0 \quad G_0 \quad +I_d
\end{align*} \]

\( G_0 \) and \( I_d \) depend on the iteration number \( k \)
\( \Rightarrow G_0 = G_0(k) \) and \( I_d = I_d(k) \)

MOS Level 1 MOSFET Equations
Body Effect Ignored

Choose source as reference node
\[ \frac{\partial I_{ds}}{\partial V_{gs}} = g_m \]
\[ \frac{\partial I_{ds}}{\partial V_{ds}} = g_{ds} \]

Linearized Model (Companion Model)
\[ I = I_{ds}^k - g_m V_{gs} \]

Problems with Newton Method
Non Convergence

Convergence Depends on a Good Initial Guess

Example:
oscillation
converges to \( \infty \)

Convergence Problems With Newton Method

Must limit the changes in \( X \)
Convergence Problems

Overflow

Must limit the changes in $X$

Newton Method with Limiting

Newton Algorithm for Solving $F(x) = 0$

$x^0 = \text{Initial Guess, } k = 0$

Repeat

1. Compute $F(x^k), J_F(x^k)$
2. Solve $J_F(x^k) \Delta x^{k+1} = -F(x^k)$ for $\Delta x^{k+1}$
3. $x^{k+1} = x^k + \text{limited}(\Delta x^{k+1})$
4. $k = k + 1$

Until $\|\Delta x^{k+1}\|, \|F(x^{k+1})\|$ small enough

Newton Method with Limiting

Limiting Schemes

- Direction Corrupting

limited$(\Delta x^{k+1})_i = \Delta x^{k+1}_i$ if $|\Delta x^{k+1}_i| < \gamma$

$\gamma \text{sign}(\Delta x^{k+1}_i)$ otherwise

- Non Direction Corrupting

limited$(\Delta x^{k+1}) = \alpha \Delta x^{k+1}$

$\alpha = \min \left\{ 1, \frac{\gamma}{\|\Delta x^{k+1}\|} \right\}$

Heuristics, No Guarantee of Convergence

Newton Method with Limiting

Damped Newton Scheme

General Damping Scheme

Solve $J_F(x^k) \Delta x^{k+1} = -F(x^k)$ for $\Delta x^{k+1}$

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

Key Idea: Line Search

Pick $\alpha^k$ to minimize $\|F(x^k + \alpha^k \Delta x^{k+1})\|_2^2$

Perform a one-dimensional search in Newton Direction