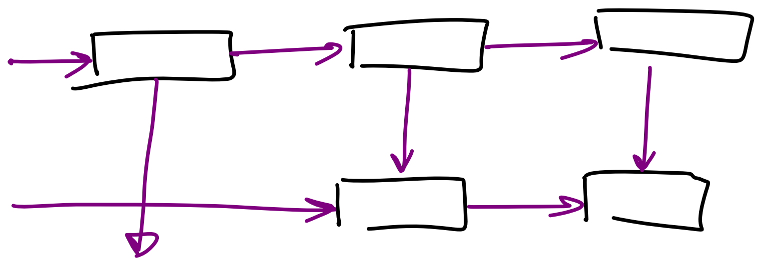
HW # 1 Due today

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

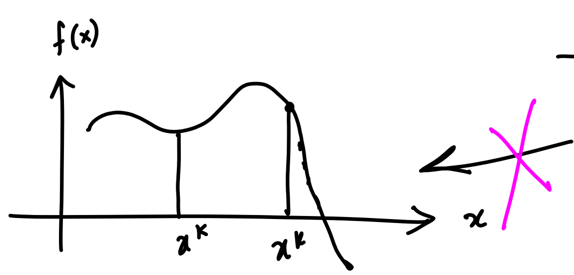


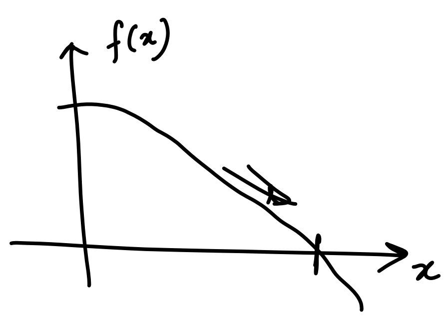
Independent voltage source

spGetElement (Matrix, A, Ibr)

$$\begin{array}{c|ccc} & V_A & V_B & I_{br} \\ \hline A & & & \boxed{+1} \\ B & & & \boxed{-1} \\ BCR & \boxed{1} & \boxed{-1} & \end{array} \qquad \begin{bmatrix} 0 \\ 0 \\ V_s \end{bmatrix}$$

Damped Newton Method





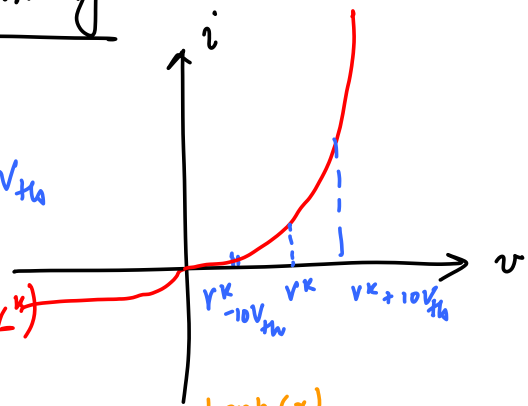
J_F^{-1} is bounded

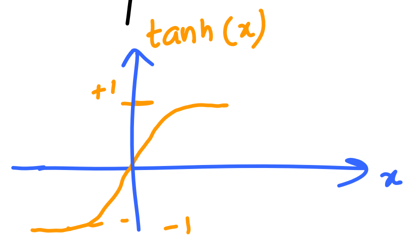
Diode (pn junction) Limiting

Ensure that

$$V^k - 10V_{th} \leq V_{lim}^{k+1} \leq V^k + 10V_{th}$$

$$V^{k+1} = V^k + 10V_{th} \tanh\left(\frac{V^{k+1} - V^k}{10V_{th}}\right)$$



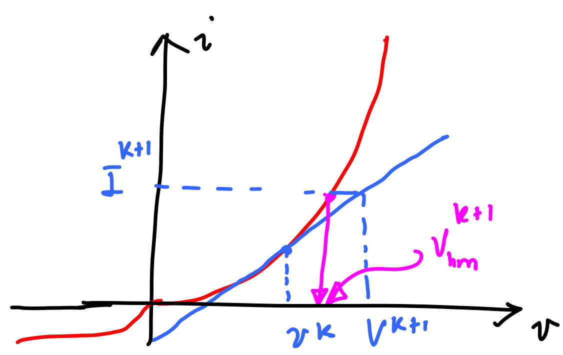


SPICE PN Junction limiting (PNJLIM)

For the diode:

$$i = I_s\left(e^{v/V_{th}} - 1\right)$$

$$\frac{\partial i}{\partial v} = \frac{I_s}{V_{th}} e^{v/V_{th}}$$



Linearization at iteration k+1

$$\hat{I} = I^{k+1} = \underbrace{I_s\left(e^{v^k/V_{th}} - 1\right)} + \frac{I_s}{V_{th}} e^{v^k/V_{th}} \left(V^{k+1} - V^k\right)$$

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

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

$$= V_{th} \ln\left[\underline{e^{v^k/V_{th}}} + \frac{e^{v^k/V_{th}}}{V_{th}}\left(V^{k+1} - V^k\right)\right]$$

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

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

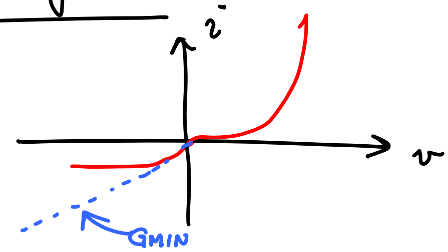
This limiting formula is applied only when

$$v^{k+1} > V_{crit}, \quad \text{and} \quad v^{k+1} > v^k$$

$$V_{crit} = V_{th} \ln\left(\frac{V_{th}}{\sqrt{2} I_s}\right) \leftarrow \text{maximum curvature}$$

What happens if v^k are negative?

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



$$i = I_s\left(e^{v/V_{th}} - 1\right) + G_{min}\, v \qquad G_{min} = 10^{-12}$$

$$\frac{\partial i}{\partial v} = \frac{I_s}{V_{th}} e^{v/V_{th}} + G_{min}$$

In SPICE2 $\quad v < -5 * V_{th} \quad i = -I_s + G_{min}\, v$

$$\frac{di}{dv} = G_{min}$$

Other limiting schemes in SPICE

FETLIM (limits V_{GS})

LIMVDS (limits V_{DS})

MOSFETs

Usage for Sparse 1.4: on unix

./sparse mat0
mat1
-- 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

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

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

Continuation methods

1) Source stepping

when sources are at zero: $x = 0$
ramp up source voltage by a small amount
(say δ_1) then $x(\delta_1)$ is close to $x(0)$

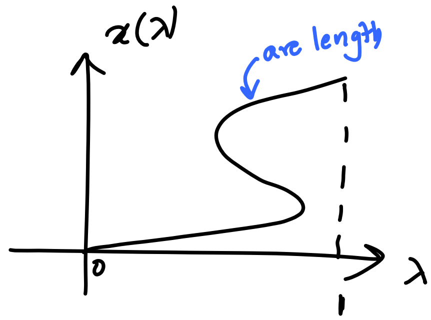
Recall "close enough"

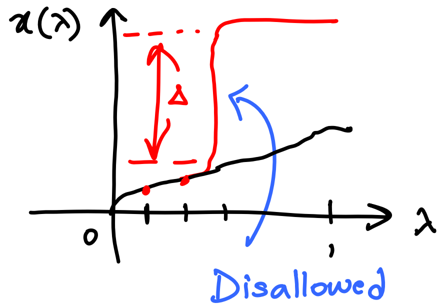
$\Rightarrow$ expect Newton's method to converge

Procedure:

Source vector S
Define a parameter λ: $0 \leq \lambda \leq 1$
Define $S(\lambda) = \lambda S$
Vary λ from 0 to 1 and solve circuit for each λ
$\lambda = 1$ is the solution for the original problem

$x(\lambda)$ i.e. the solution for source λS should be sufficiently smooth





General Case

$f(x) = 0$ original problem

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

where:

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

Simple scheme:

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

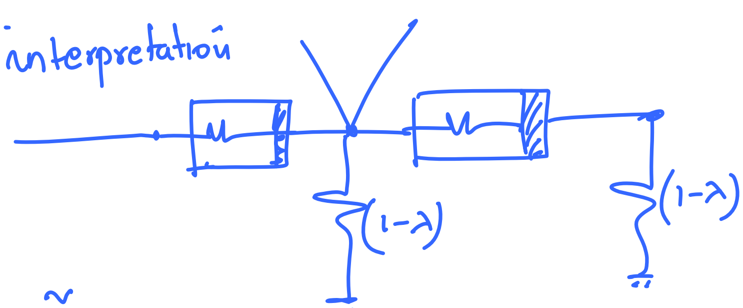
For $\lambda = 0$: $\tilde{f}(x(0), 0) = x(0) = 0$

$\lambda = 1$: $\tilde{f}(x(1), 1) = f(x(1)) = 0$

Application of Newton's method

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

Circuit interpretation



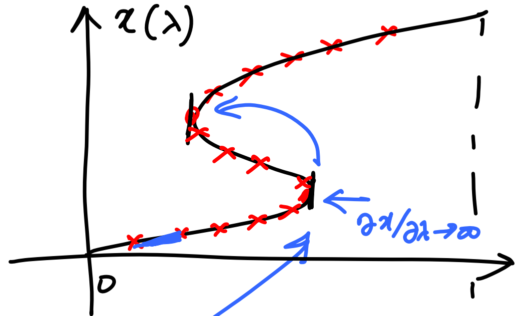
$$\frac{\partial \tilde{f}}{\partial \lambda} = f(x(\lambda)) - x(\lambda)$$

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

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

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

↑ singular $\frac{\partial \tilde{f}}{\partial x}$



$$\delta\sigma = \sqrt{(\delta x)^2 + (\delta\lambda)^2}$$

Parameterize x & λ in terms of the arc length

$x(\sigma)$ $\qquad 0 \leq \sigma \leq \sigma_L$

$\lambda(\sigma)$

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

$$\|\delta x\|^2 + |\delta\lambda|^2 = \delta\sigma^2$$

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

<u>Exact arc length :</u> $\qquad \|\dot{x}(s)\|^2 + |\dot{\lambda}(s)|^2 = 1$

↑ exact arc length

HOMPACK $\qquad$ 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 $F(x)=0$

$x^0 =$ Initial Guess, $k=0$

Repeat {

Compute $F(x^k), J_F(x^k)$

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

$k=k+1$

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

From: A. Nardi

Multidimensional Newton Method
Convergence

Local Convergence Theorem

If

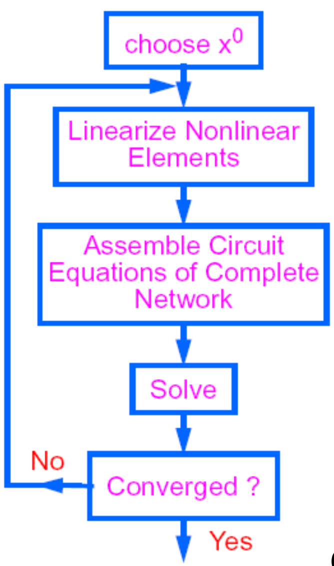
$a)\ \|J_F^{-1}(x^k)\| \le \beta$ (Inverse is bounded)

$b)\ \|J_F(x)-J_F(y)\| \le \ell\|x-y\|$ (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



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

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

Solve linear equations (spFactor, spSolve)

$$\left|x^{(k+1)} - x^k\right| \le \varepsilon_A + \varepsilon_R \min\left(\left|x^{(k+1)}\right|, \left|x^{(k)}\right|\right)$$

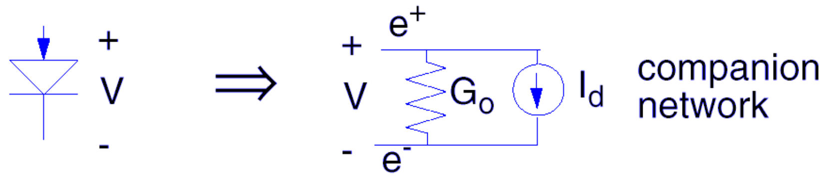
On a per unknown basis

Practical Considerations

- **Device model equations must be continuous with continuous derivatives**
  - Not all models have this property
    - **Recent shift to C∞ 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



	e^+	e^-	RHS
+	G_o	$-G_o$	$-I_d$
-	$-G_o$	G_o	$+I_d$

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

From: A. Nardi

MOS Level 1 MOSFET Equations

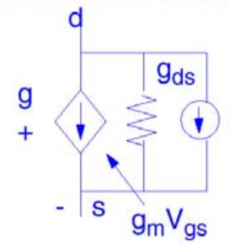
Body Effect Ignored

Choose source as reference node

$$\frac{\partial I_{ds}}{\partial V_{gs}} \equiv g_m$$

$$\frac{\partial I_{ds}}{\partial V_{ds}} \equiv g_{ds}$$

Linearized Model (Companion Model)



$$I = I^k_{ds} - g_m v^k_{gs} - v^k_{ds}$$

	e_d	e_s	e_g	RHS
d	g_{ds}	$-g_{ds}-g_m$	g_m	$-I$
s	$-g_{ds}$	$g_{ds}+g_m$	$-g_m$	I
g				

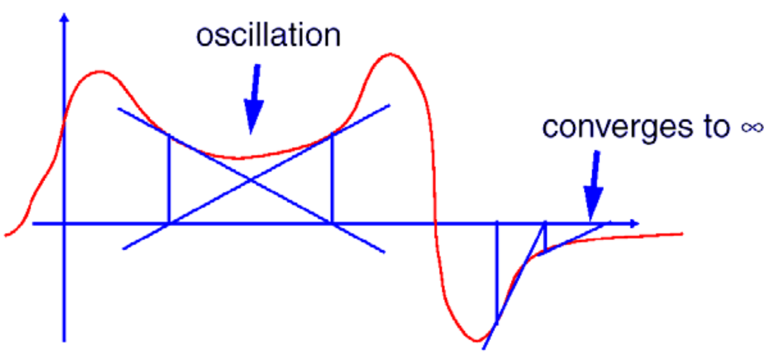
From: A. Nardi

Problems with Newton Method

Non Convergence

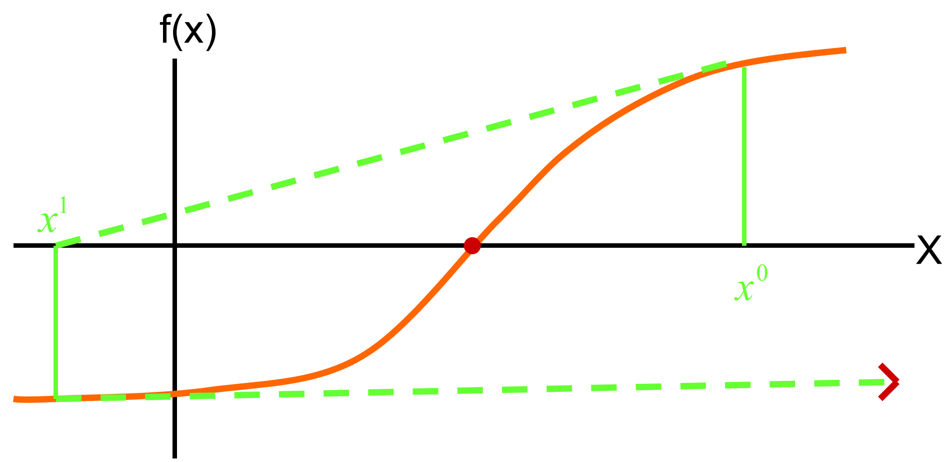
Convergence Depends on a Good Initial Guess

Example:



From: A. Nardi

Convergence Problems With Newton Method

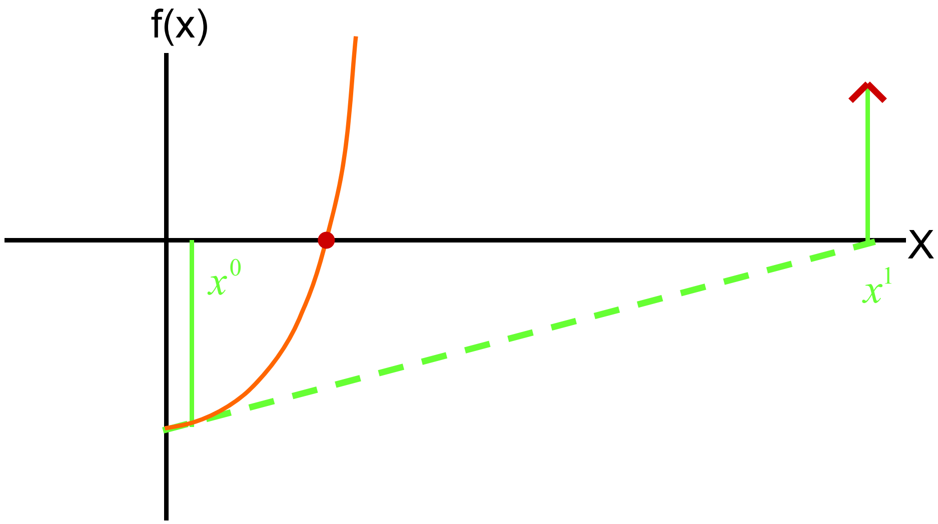


Must limit the changes in X

From: A. Nardi

Convergence Problems

Overflow



Must limit the changes in X

From: A. Nardi

Newton Method with Limiting

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

$x^0 =$ Initial Guess, $k = 0$

Repeat {

Compute $F(x^k), J_F(x^k)$

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

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

$k = k+1$

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

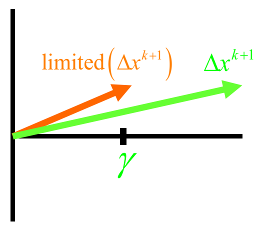
From: A. Nardi

Newton Method with Limiting

Limiting Schemes

- Direction Corrupting

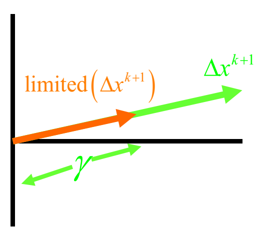
$$\text{limited}(\Delta x^{k+1})_i = \begin{cases} \Delta x_i^{k+1} & \text{if } |\Delta x_i^{k+1}| < \gamma \\ \gamma\, sign(\Delta x_i^{k+1}) & \text{otherwise} \end{cases}$$



- Non Direction Corrupting

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

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



Heuristics, No Guarantee of Convergence

From: A. Nardi

Newton Method with Limiting

Damped Newton Scheme

General Damping Scheme

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

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

Key Idea: Line Search

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

Perform a one-dimensional search in Newton Direction

From: A. Nardi