Initialization and Limiting for NR in MAPP

Tianshi Wang

University of California, Berkeley

tianshi@berkeley.edu

December 17, 2014

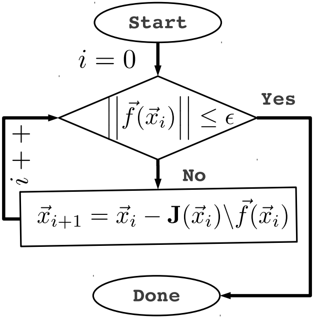
Overview

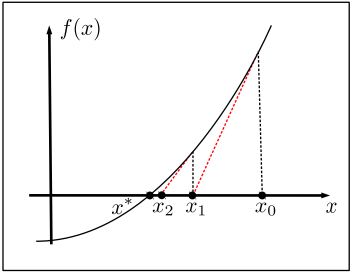
1. Background
   - the Newton Raphson method
     - the need for init/limiting
   - previous implementations of init/limiting
     - in MAPP
     - in SPICE
     - in Xyce
2. Mathematics behind init/limiting
   - algebraic function formula with init/limiting
   - equivalence with SPICE
   - equivalence with Xyce
3. Implementation in MAPP
   - ModSpec level
   - DAE level
   - Analysis level
   - TODOs
4. Conclusions

the Newton Raphson method

NR: iterative numerical algorithm to solve $\vec{f}(\vec{x}) = \vec{0}$

- Start with $\vec{x}_0$, update $\vec{x}_i$ with derivative information
- $\vec{dx}_i = \vec{x}_{i+1} - \vec{x}_i = -\mathbf{J}(\vec{x}_i)\backslash\vec{f}(\vec{x}_i)$



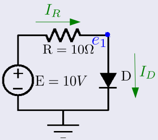


NR: no guarantee to converge!

the need for init/limiting

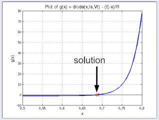
In fact, it is very easy to break NR

simple diode example

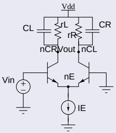


circuit equation: $I_D - I_R = 0$

$$\Longrightarrow \quad f(e_1) \triangleq I_S\left(e^{\frac{e_1}{V_T}} - 1\right) - \frac{\mathrm{E} - e_1}{\mathrm{R}} = 0$$



BJT differential pair example



won't converge unless given very good initial guess

```
x0 = [5; 3; 3; -0.7; 0.1; -2e-3; 0];
...
qss = feval(qss.solve, x0, qss);
sol = feval(qss.getsolution, qss); % sol close to x0
```

and many more $\cdots$

diode mixer (4 diodes), 741 op-amp (∼20 BJTs), $\cdots$

Previous Implementation in MAPP

initialization

set $\vec{x}_0$ to "good" value

e.g.

x0=[5; 3; 3; -0.7; 0.1; -2e-3; 0];

or

x0=DAE.QSSinitguess(DAE);

limiting

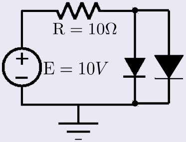
directly limit NR update $\vec{dx}_i$

$$\begin{aligned} \vec{dx}_i &= -\mathbf{J}(\vec{x}_i)\backslash\vec{f}(\vec{x}_i) \\ \vec{dx}_i &= \text{limiting}(\vec{dx}_i, \vec{x}_i) \\ \vec{x}_{i+1} &= \vec{x}_i + \vec{dx}_i \end{aligned}$$

implemented in DAE.NRlimiting

problems

- difficult to write at system-level
  - requires knowing all unk indices/names
  - subject to change with inputs
- how to get it from devices?
- init/limiting for different devices may conflict
  - e.g. two diodes in parallel



init/limiting in SPICE

NR in SPICE

SPICE uses RHS — circuit (current) right hand side

$$\begin{aligned} \vec{x}_{i+1} &= \mathbf{J}(\vec{x}_i)\backslash\mathrm{RHS}(\vec{x}_i) \\ \mathrm{RHS}(\vec{x}_i) &\approx -\vec{f}(\vec{x}_i) + \mathbf{J}(\vec{x}_i)\cdot\vec{x}_i \end{aligned}$$

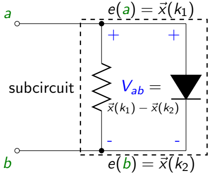
When there is no init/limiting:

$$\begin{aligned} \mathbf{J}(\vec{x}_i)\cdot\vec{x}_{i+1} &= \mathrm{RHS}(\vec{x}_i) = -\vec{f}(\vec{x}_i) + \mathbf{J}(\vec{x}_i)\cdot\vec{x}_i \\ \vec{x}_{i+1} - \vec{x}_i &= -\mathbf{J}(\vec{x}_i)\backslash\vec{f}(\vec{x}_i) \end{aligned}$$

$\mathbf{J}$ and RHS evaluation in SPICE

- each device contributes to $\mathbf{J}/\mathrm{RHS}$ (based on its bias)
  - $\vec{x}$ contains node voltages, but devices use branch voltages
- when using init/limiting, device directly/“secretly” changes its bias voltages for its $\mathbf{J}/\mathrm{RHS}$ evaluation
- system-level inconsistency
  - $\vec{x}$ values are not changed uniformly in device evaluation

init/limiting in SPICE: example

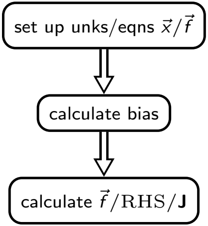


$$I_R = V_{ab} \cdot G$$

$$\hat{V}_{ab} = \mathrm{pnjlim}(V_{ab}, V_{ab}^{(old)})$$

$$I_D = \mathrm{Diode\_Id}(\hat{V}_{ab})$$

$$G_D = \mathrm{dDiode\_Id}(\hat{V}_{ab})$$



$$\vec{x} \qquad \begin{array}{c} \\ k_1 \\ k_2 \\ \\ \end{array} \begin{bmatrix} \vdots \\ e(a) \\ e(b) \\ \vdots \end{bmatrix}$$

$$\vec{f} \qquad \begin{array}{c} \\ k_1 \\ k_2 \\ \\ \end{array} \begin{bmatrix} \vdots \\ \mathrm{KCL}(a) + I_R + I_D \\ \mathrm{KCL}(b) - I_R - I_D \\ \vdots \end{bmatrix}$$

$$\mathbf{J}$$

$$\begin{array}{c} \\ \\ k_1 \\ k_2 \\ \\ \end{array} \begin{array}{c} \quad k_1 \qquad\qquad\qquad\qquad k_2 \\ \begin{bmatrix} & \vdots & \vdots & \\ \cdots & \frac{d\mathrm{KCL}(a)}{de(a)} + G + G_D & \frac{d\mathrm{KCL}(a)}{de(b)} - G - G_D & \cdots \\ \cdots & \frac{d\mathrm{KCL}(b)}{de(a)} - G - G_D & \frac{d\mathrm{KCL}(b)}{de(b)} + G + G_D & \cdots \\ & \vdots & \vdots & \end{bmatrix} \end{array}$$

$$\mathrm{RHS} \approx \mathbf{J}\vec{x} - \vec{f}$$

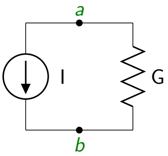
I_R uses V_{ab}

I_D and G_D use $\hat{V}_{ab}$

$$\mathrm{RHS} \qquad \begin{array}{c} \\ k_1 \\ k_2 \\ \\ \end{array} \begin{bmatrix} \vdots \\ \mathrm{RHS}(a) + G_D \cdot \hat{V}_{ab} - I_D \\ \mathrm{RHS}(b) - G_D \cdot \hat{V}_{ab} + I_D \\ \vdots \end{bmatrix}$$

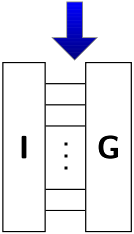
What is RHS ?

Linear Circuit:



$$\vec{x} = \begin{bmatrix} e(a) \\ e(b) \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} +G & -G \\ -G & +G \end{bmatrix}$$

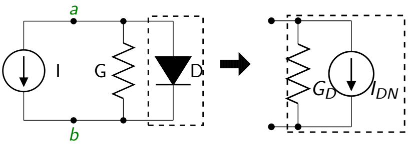
$$\mathbf{G} \cdot \vec{x}^* = \begin{bmatrix} +I \\ -I \end{bmatrix} = \mathbf{I}$$



$\mathbf{G} \cdot \vec{x}^* = \mathbf{I}$

current sources

Nonlinear Circuit:



use Norton's Theorem!

$$\mathbf{J} = \begin{bmatrix} +G + G_D & -G - G_D \\ -G - G_D & +G + G_D \end{bmatrix}$$

$$\mathbf{J} \cdot \vec{x}^* = \begin{bmatrix} +I + I_{DN} \\ -I - I_{DN} \end{bmatrix} = \mathrm{RHS}$$

$\mathbf{J} \cdot \vec{x}^* = \mathrm{RHS}$

current right hand side

init/limiting in Xyce

limiting correction in Xyce

$$
\begin{aligned}
\vec{x}_{k+1} - \vec{x}_k = \Delta\vec{x}_k^{(total)} &= \Delta\vec{x}_k^{(newton)} + \Delta\vec{x}_k^{(correction)} \\
&= -\mathbf{J}(\vec{x}_k)\backslash\vec{f}(\vec{x}_k) + \Delta\vec{x}_k^{(correction)} \\
&= -\mathbf{J}(\vec{x}_k)\backslash\left[\vec{f}(\vec{x}_k) + \vec{f}_k^{(correction)}\right]
\end{aligned}
$$

```
bool Instance::loadDAEFVector ()
{
// 3f5 compatible currents
// Including derivation of Vd_diff and
// Limiting Correction
...
// load the voltage limiter vector.
if( getDeviceOptions().voltageLimiterFlag )
{
    double Vd_diff = Vd - Vd_orig;
    double Gd_Jdxp = 0.0;
    Gd_Jdxp = -( Gd ) * Vd_diff;
    // Load the dFdxdVp vector
    (extData.dFdxdVpVectorRawPtr)[li_Neg] += Gd_Jdxp;
    (extData.dFdxdVpVectorRawPtr)[li_Pri] -= Gd_Jdxp;
}
...
return true;
}
```

Merits

- SPICE compatible
- can be turned off!

Problems

- physical meaning?
- new device? how?

Challenges with init/limiting Implementations

- figure out what they really are
- get NR to converge
- easy to explain to device/DAE developers
- easy for users to write
  - avoid system level limiting
- SPICE compatible, but more than SPICE
  - diode-res ModSpec example, limits only junction voltage
- backward compatibility
  - some devices/DAEs don't provide init/limiting information
  - init/limiting can be turned off

Mathematical Expression of init/limiting

- contains "dangerous" functions (sensitive to their inputs)
  - e.g. $\exp()$

$$\vec{f}\,(\vec{x}) = \vec{0} \longrightarrow \hat{\vec{f}}\,(\vec{x},\ \overrightarrow{xlim}\,) = \vec{0}$$

- $\overrightarrow{xlim}$: inputs to "dangerous" functions
  - subset of $\vec{x}$ or linear combination of $\vec{x}$
- $\hat{\vec{f}}$: new function, slightly different from $\vec{f}$
  - easy to write from $\vec{f}$
- unknown space enlarged

Mathematical Expression of init/limiting

$$\hat{\vec{f}}(\vec{x}, \overrightarrow{xlim}) = \vec{0}$$

What is $\overrightarrow{xlim}$

- without init/limiting:
  - subset of $\vec{x}$ or linear combination of $\vec{x}$

$$\overrightarrow{xlim} = \mathbf{xTOxlimMatrix} \cdot \vec{x}$$

$$\hat{\vec{f}}(\vec{x}, \mathbf{xTOxlimMatrix} \cdot \vec{x}) = \vec{f}(\vec{x})$$

- with initialization:
  - directly set to good values

$$\overrightarrow{xlim} = \overrightarrow{\mathrm{initGuess}}$$

- with limiting:
  - limited based on old values (last successful iteration)

$$\overrightarrow{xlim} = \mathrm{limiting}(\vec{x}, \overrightarrow{xlimOld})$$

Mathematical Expression of init/limiting

$$\hat{\vec{f}}(\vec{x}, \overrightarrow{xlim}) = \vec{0}$$

- We now know $\vec{x} \rightarrow \overrightarrow{xlim}$
- rewrite $\hat{\vec{f}}$ as
$$\tilde{\vec{f}}(\vec{x}) = \hat{\vec{f}}(\vec{x}, \overrightarrow{xlim}(\vec{x}))$$
- without init/limiting:
$$\tilde{\vec{f}}(\vec{x}) = \vec{f}(\vec{x})$$
- with init/limiting:
  - e.g. with limiting
$$\hat{\vec{f}}(\vec{x}, \mathrm{limiting}(\vec{x}, \overrightarrow{xlimOld})) = \tilde{\vec{f}}(\vec{x}, \overrightarrow{xlimOld})$$
  - $\tilde{\vec{f}}$: a function of $\vec{x}$ and parameters $\overrightarrow{xlimOld}/\overrightarrow{initGuess}$
  - converges to the same solution
  - has better numerical properties

Mathematical Expression of init/limiting

$$\tilde{\vec{f}}(\vec{x}) = \hat{\vec{f}}(\vec{x}, \overrightarrow{xlim}) = \vec{0}$$

This is the equation we are actually solving

What's **J**

- $\hat{\vec{f}}$ has multiple derivatives: $\frac{\partial \hat{\vec{f}}}{\partial \vec{x}}$ and $\frac{\partial \hat{\vec{f}}}{\partial \overrightarrow{xlim}}$.
- without init/limiting

$$\frac{d\tilde{\vec{f}}}{d\vec{x}} = \frac{\partial \hat{\vec{f}}}{\partial \vec{x}} + \frac{\partial \hat{\vec{f}}}{\partial \overrightarrow{xlim}} \cdot \mathbf{xTOxlimMatrix} \quad = \frac{d\vec{f}}{d\vec{x}}$$

- with init/limiting
  - what's **J** depends on your perspective
  - treat $\overrightarrow{xlim}$ as a function of $\vec{x}$

$$\frac{d\tilde{\vec{f}}}{d\vec{x}} = \frac{\partial \hat{\vec{f}}}{\partial \vec{x}} + \frac{\partial \hat{\vec{f}}}{\partial \overrightarrow{xlim}} \cdot \frac{d\text{limiting}}{d\vec{x}}$$

  - treat $\overrightarrow{xlim}$ as extra variables ?
    - same as no limiting scenario   but $\neq \frac{d\vec{f}}{d\vec{x}}$
    - SPICE/Xyce's **J**

Compatibility with SPICE

- don't want to bother device/DAE users with RHS
- construct RHS from $\hat{\vec{f}}(\vec{x}, \overrightarrow{xlim})$

$$\mathrm{RHS} = (\frac{\partial \hat{\vec{f}}}{\partial \vec{x}} \cdot \vec{x} + \frac{\partial \hat{\vec{f}}}{\partial \overrightarrow{xlim}} \cdot \overrightarrow{xlim}) - \hat{\vec{f}}$$

- calculate $\mathbf{J}$ from $\hat{\vec{f}}(\vec{x}, \overrightarrow{xlim})$

$$\mathbf{J} = \frac{\partial \hat{\vec{f}}}{\partial \vec{x}} + \frac{\partial \hat{\vec{f}}}{\partial \overrightarrow{xlim}} \cdot \mathbf{xTOxlimMatrix}$$

- RHS moved from devices to analyses
  - no MOD.RHS or DAE.RHS in MAPP
  - analyses can construct SPICE-compatible RHS when needed

Compatibility with Xyce

$\mathbf{J}$ in Xyce

$$\mathbf{J} = \frac{\partial \hat{\vec{f}}}{\partial \vec{x}} + \frac{\partial \hat{\vec{f}}}{\partial \overrightarrow{xlim}} \cdot \mathbf{xTOxlimMatrix}$$

same as in SPICE

limiting correction in Xyce

$$\begin{aligned}
\vec{x}_{k+1} - \vec{x}_k &= \Delta\vec{x}_k^{(total)} = \Delta\vec{x}_k^{(newton)} + \Delta\vec{x}_k^{(correction)} \\
&= -\mathbf{J}(\vec{x}_k, \overrightarrow{xlim}_k)\backslash\hat{\vec{f}}(\vec{x}_k, \overrightarrow{xlim}_k) + \mathbf{J}(\vec{x}_k)\backslash\left[\left.\frac{\partial \hat{\vec{f}}}{\partial \overrightarrow{xlim}}\right|_{\vec{x}_k, \overrightarrow{xlim}_k} \cdot (\overrightarrow{xlim}_k - \mathbf{xTOxlimMatrix} \cdot \vec{x}_k)\right] \\
&= -\mathbf{J}(\vec{x}_k, \overrightarrow{xlim}_k)\backslash\left[\hat{\vec{f}}(\vec{x}_k, \overrightarrow{xlim}_k) + \left.\frac{\partial \hat{\vec{f}}}{\partial \overrightarrow{xlim}}\right|_{\vec{x}_k, \overrightarrow{xlim}_k} \cdot (\overrightarrow{xlim}_k - \mathbf{xTOxlimMatrix} \cdot \vec{x}_k)\right]
\end{aligned}$$

Evidence of Limiting Correction in Other Simulators

"If limiting occurs, the simulator may need to apply a so-called limiting correction to correct for the fact that currents were not computed at the voltages requested by the simulator. (SPICE itself does not use the currents directly because it solves for the new voltage vector instead of the voltage increment; the so-called SPICE right-hand side does not need a limiting correction.) When correcting a current for limiting, one adds to the current a term composed of the derivative of the current with respect to the limited voltage multiplied by the difference between the voltage requested by the simulator and that used in the equations." [1]

"The proposed syntax was tested in the internal circuit simulator at Analog Devices, $\cdots$"

---

[1] Lemaitre, Laurent, et al. "Extensions to Verilog-A to support compact device modeling." Behavioral Modeling and Simulation, 2003. BMAS 2003. Proceedings of the 2003 International Workshop on. IEEE, 2003.

Xyce's Compatibility with SPICE

Why limiting correction is equivalent to SPICE's RHS?
limiting correction:

$$\vec{x}_{k+1} - \vec{x}_k = \Delta\vec{x}_k^{(total)} = \Delta\vec{x}_k^{(newton)} + \Delta\vec{x}_k^{(correction)}$$

$$= -\mathbf{J}(\vec{x}_k, \overrightarrow{xlim_k}) \backslash \left[\hat{\vec{f}}(\vec{x}_k, \overrightarrow{xlim_k}) + \left.\frac{\partial \hat{\vec{f}}}{\partial \overrightarrow{xlim}}\right|_{\vec{x}_k, \overrightarrow{xlim_k}} \cdot (\overrightarrow{xlim_k} - \mathbf{xTOxlimMatrix} \cdot \vec{x}_k) \right]$$

$$\Longrightarrow \quad \mathbf{J}(\vec{x}_k, \overrightarrow{xlim_k}) \cdot \vec{x}_{k+1} - \left(\left.\frac{\partial \hat{\vec{f}}}{\partial \vec{x}}\right|_{\vec{x}_k, \overrightarrow{xlim_k}} + \left.\frac{\partial \hat{\vec{f}}}{\partial \overrightarrow{xlim}}\right|_{\vec{x}_k, \overrightarrow{xlim_k}} \cdot \mathbf{xTOxlimMatrix} \right) \cdot \vec{x}_k$$

$$= \mathbf{J}(\vec{x}_k, \overrightarrow{xlim_k}) \cdot (\vec{x}_{k+1} - \vec{x}_k)$$

$$= -\vec{f}(\vec{x}_k, \hat{\overrightarrow{xlim_k}}) + \left[\left.\frac{\partial \hat{\vec{f}}}{\partial \overrightarrow{xlim}}\right|_{\vec{x}_k, \overrightarrow{xlim_k}} \cdot (\overrightarrow{xlim_k} - \mathbf{xTOxlimMatrix} \cdot \vec{x}_k) \right]$$

$$\Longrightarrow \mathbf{J}(\vec{x}_k, \overrightarrow{xlim_k}) \cdot \vec{x}_{k+1} = \left.\frac{\partial \hat{\vec{f}}}{\partial \vec{x}}\right|_{\vec{x}_k, \overrightarrow{xlim_k}} \cdot \vec{x}_k + \left.\frac{\partial \hat{\vec{f}}}{\partial \overrightarrow{xlim}}\right|_{\vec{x}_k, \overrightarrow{xlim_k}} \cdot \overrightarrow{xlim_k} - \hat{\vec{f}}(\vec{x}_k, \overrightarrow{xlim_k})$$

$$= \mathrm{RHS}(\vec{x}_k, \overrightarrow{xlim_k})$$

Implementation in MAPP: ModSpec level

demo

go to init-limiting-notes for contents

Implementation in MAPP: DAE level

demo

go to init-limiting-notes for contents

Implementation in MAPP: Analysis level

demo

go to init-limiting-notes for contents

Implementation in MAPP: TODOs

- clean up new versions of ModSpec/DAEAPI and documentation
- xTOxlim or xuTOxlim?
  - debate over the role of u especially in hand-coded DAEs e.g. allow q(x, u)?
- efficiency considerations:
  - e.g. df = dfdx + dfdxlim * dxlimdx
    potentially calls vecvalder 3 times
    (not always, when xlim = [ ], vecvalder is by-passed)
- more thoughts on AFobj
  - why we have it in the first place

Conclusions

- previous implementations of init/limiting
  - in MAPP, SPICE and Xyce
  - their drawbacks
- mathematical formula for init/limiting
  - how to replicate SPICE/Xyce's implementation with it
- implementation in MAPP
  - at ModSpec/DAE/Analyses level
  - how to update your existing ModSpec/DAE/Analyses with init/limiting