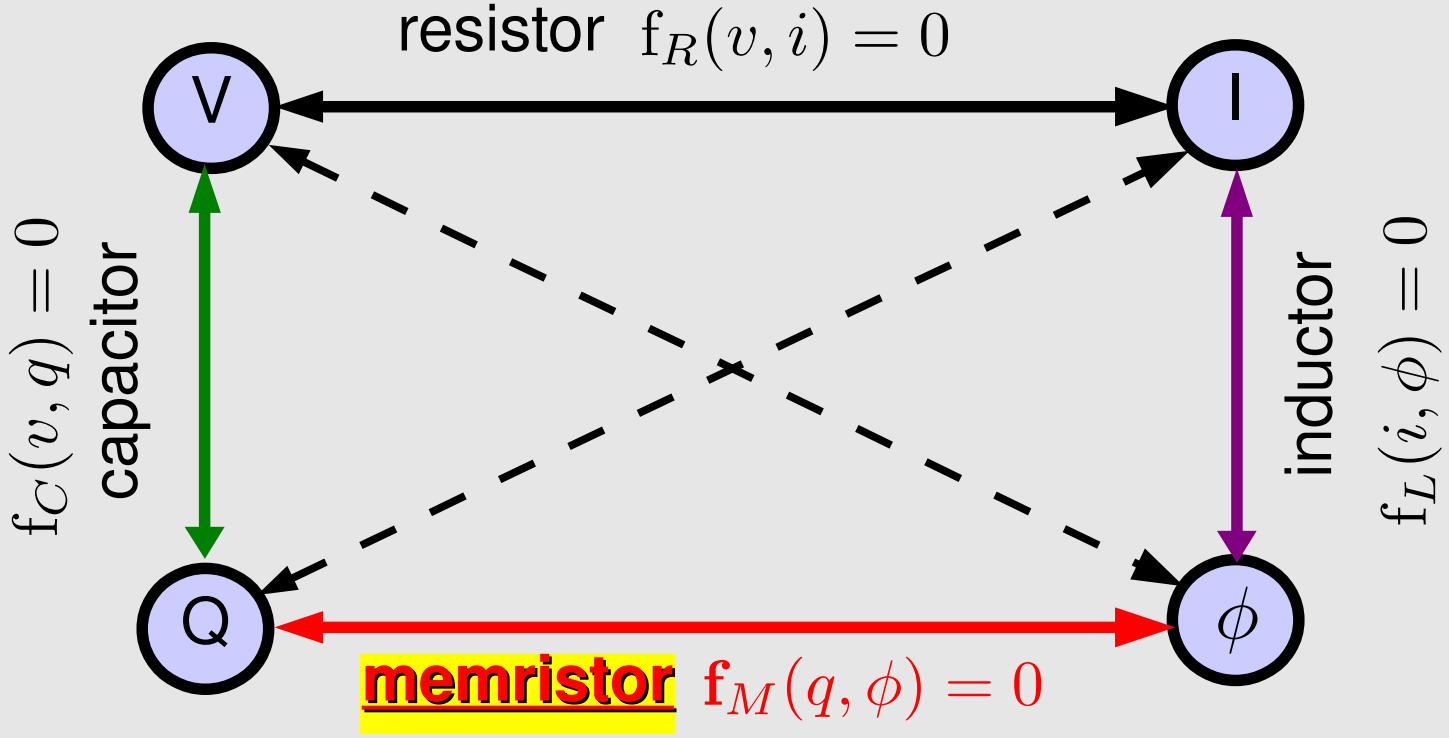
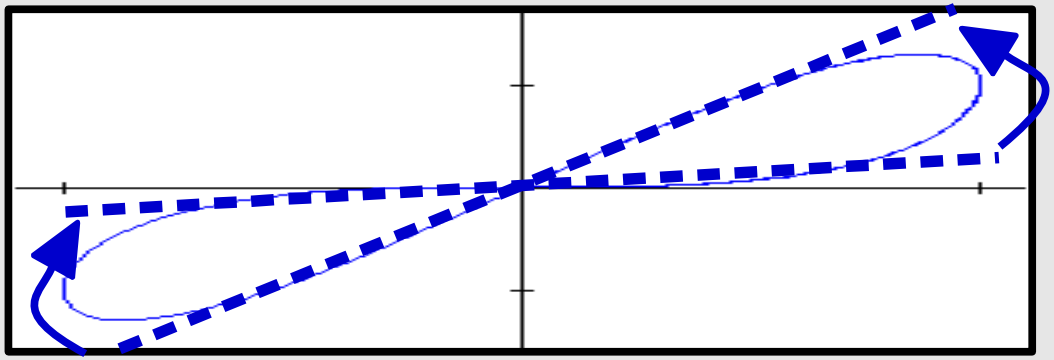
Well-Posed Models of Memristive Devices

Tianshi Wang

Department of EECS, University of California, Berkeley

Memristor: the missing element





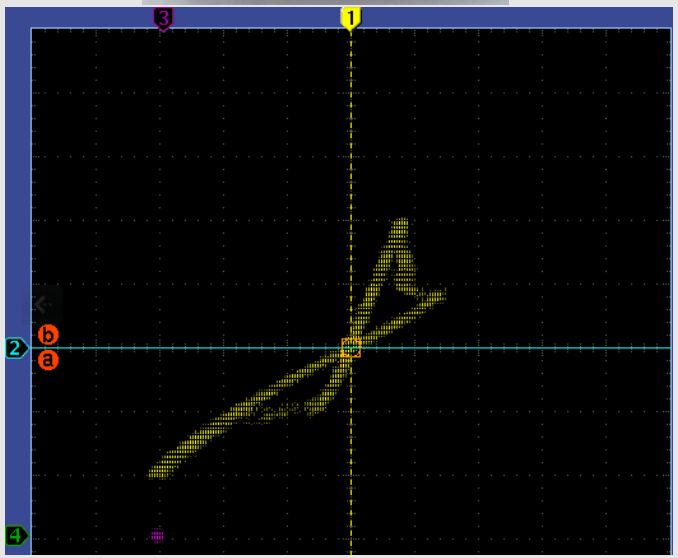
- 1971: postulated by Leon Chua
- 2008: “invented” by Stan Williams et al, HP Labs

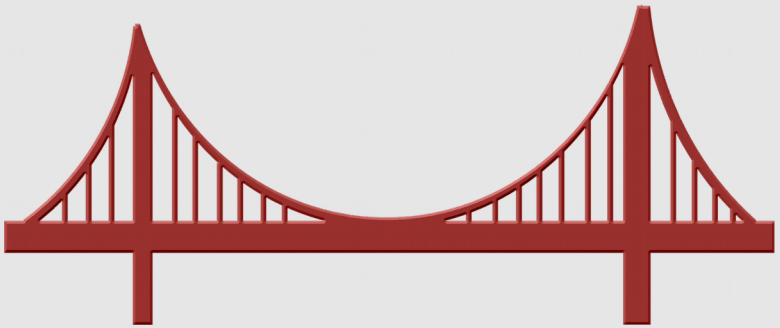
Memristive Devices & Applications

devices

UMich, Stanford,
HP, HRL Labs,
Micron, Crossbar,
Samsung, ...

Knowm





applications

- nonvolatile memories
- FPAAs
- neuromorphic circuits
- oscillators

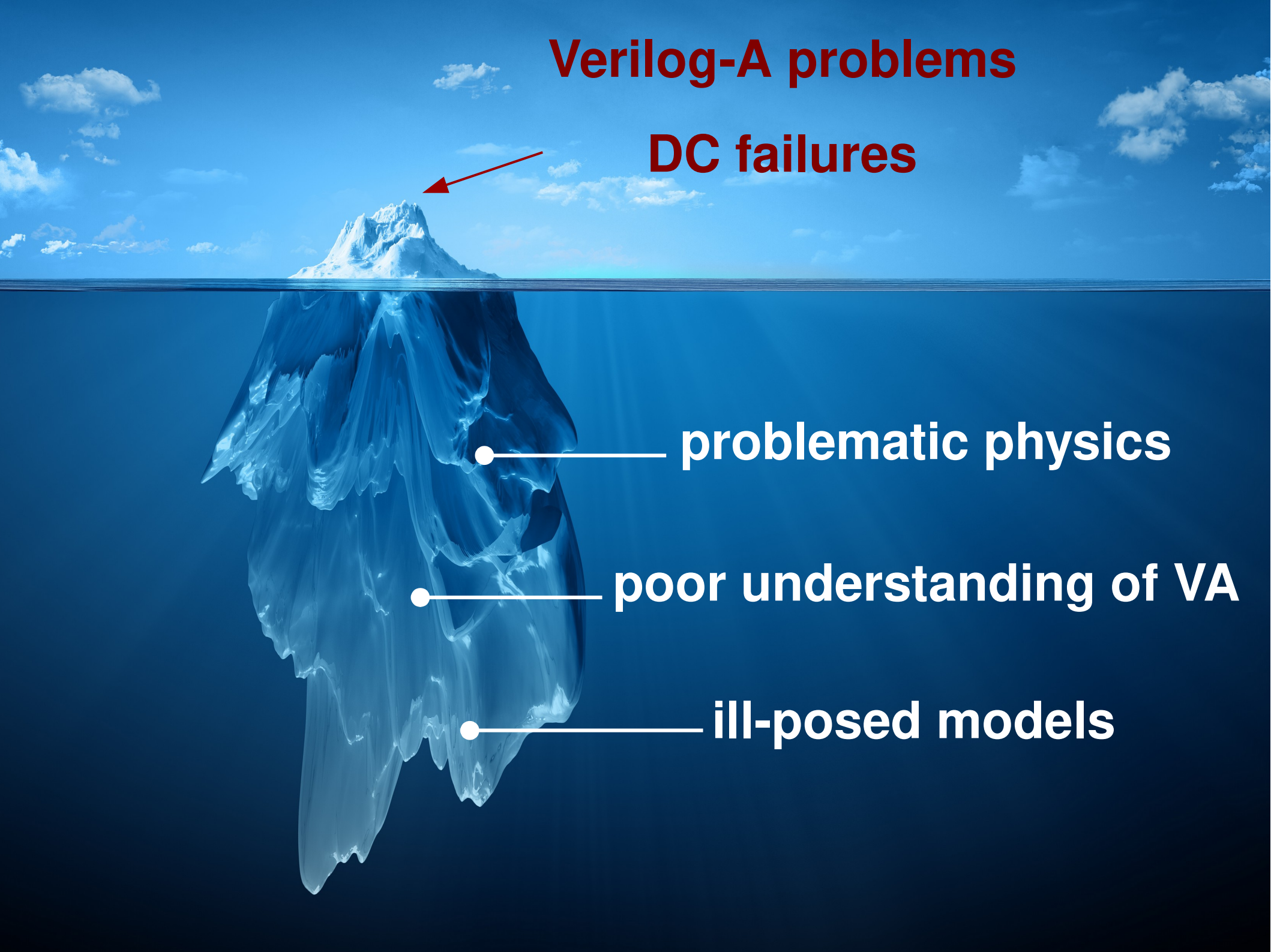
Compact Models

- Linear/nonlinear ion drift models
  Biolek (2009), Joglekar (2009),
  Prodromakis (2011), etc.
- UMich RRAM model (2011)
- TEAM model (2012)
- Simmons tunneling barrier model (2013)
- Yakopcic model (2013)
- Stanford/ASU RRAM model (2014)
- Knowm "probabilistic" model (2015)

none works in DC

Verilog-A problems

idt(), $bound_step,
$abstime, @initial_step,
$rdist_normal, ...


Verilog-A problems
DC failures
problematic physics
poor understanding of VA
ill-posed models

Good Compact Models

- "Simulation-ready"
  - run in all analyses (DC, AC, TRAN, homotopy, PSS, …)
  - run in all simulators

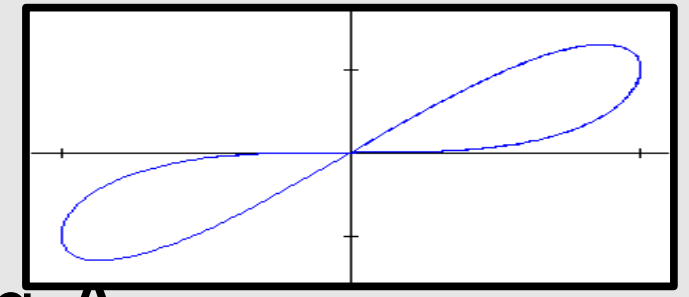
consistently

~~analysis-specific code~~

- Well-posed
  - a solution exists
  - the solution is unique
  - the solution's behavior changes continuously with the initial conditions.

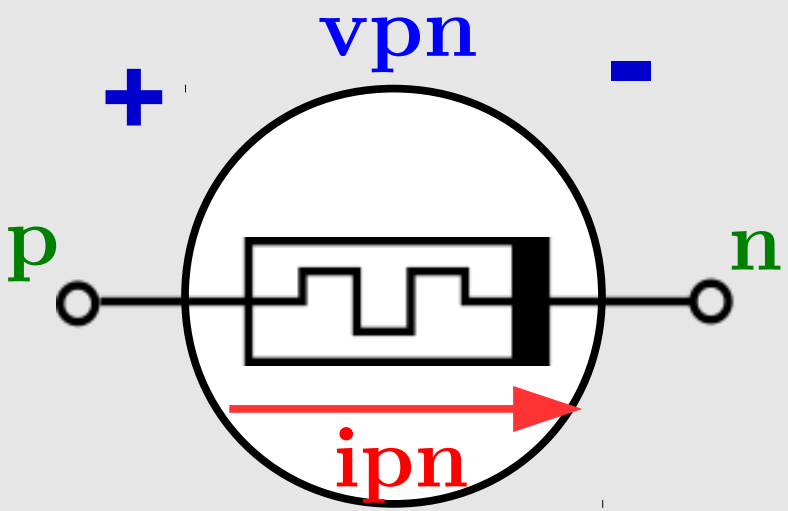
https://en.wikipedia.org/wiki/Well-posed_problem

Challenges in Memristor Modelling



- hysteresis
  - internal state variable
- model internal unks in Verilog-A
  - use potentials/flows
- upper/lower bounds of internal unks
  - physical distance
  - clipping functions
- smoothness, continuity, finite precision issues, ...

How to Model Hysteresis Properly

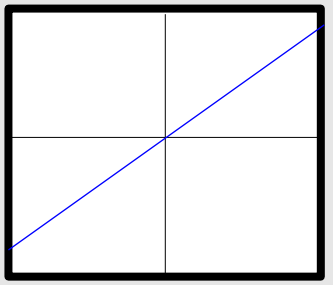


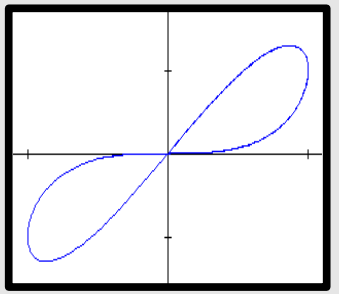
$$\mathbf{ipn} = f\,(\mathbf{vpn})$$

$$\Updownarrow$$

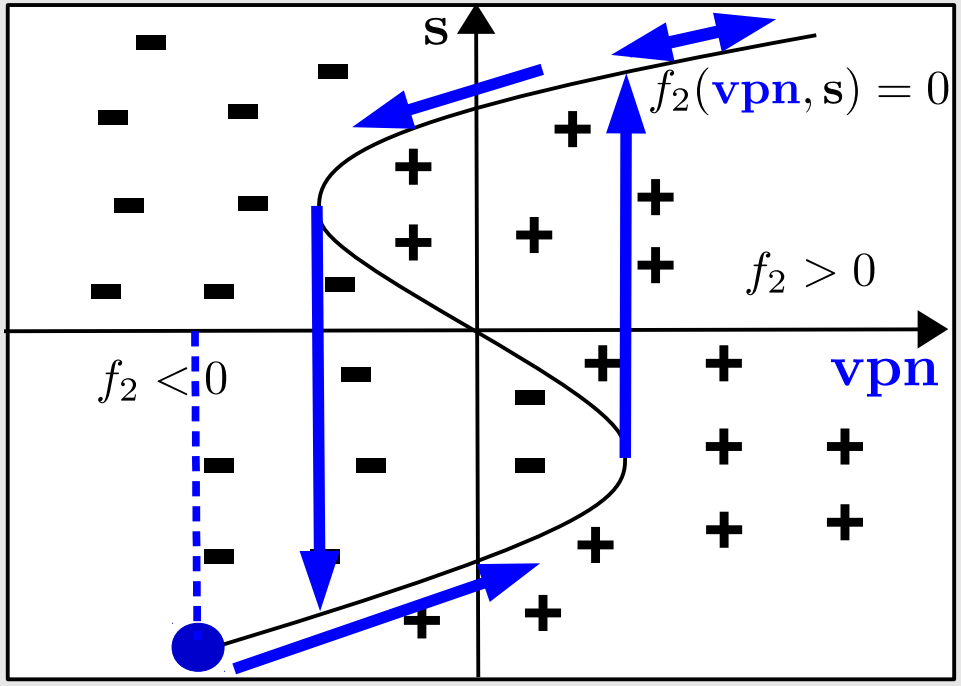
$$\mathbf{ipn} = f_1\,(\mathbf{vpn},\ \mathbf{s})$$

$$\frac{d}{dt}\mathbf{s} = f_2\,(\mathbf{vpn},\ \mathbf{s})$$





internal state variable
"memory"



Example:

$$f_1\,(\mathbf{vpn},\ \mathbf{s}) = \frac{\mathbf{vpn}}{R} \cdot (1 + \tanh(\mathbf{s}))$$

$$f_2\,(\mathbf{vpn},\ \mathbf{s}) = \mathbf{vpn} - \mathbf{s}^3 + \mathbf{s}$$

multiple stability and
abrupt change in DC sols

How to Model Hysteresis Properly

Template:

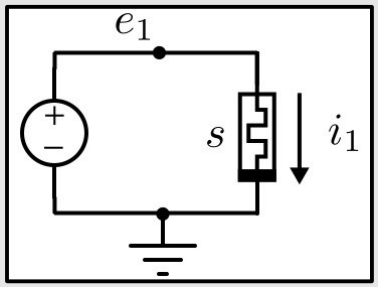
$$\mathbf{ipn} = f_1\left(\mathbf{vpn},\ \mathbf{s}\right)$$

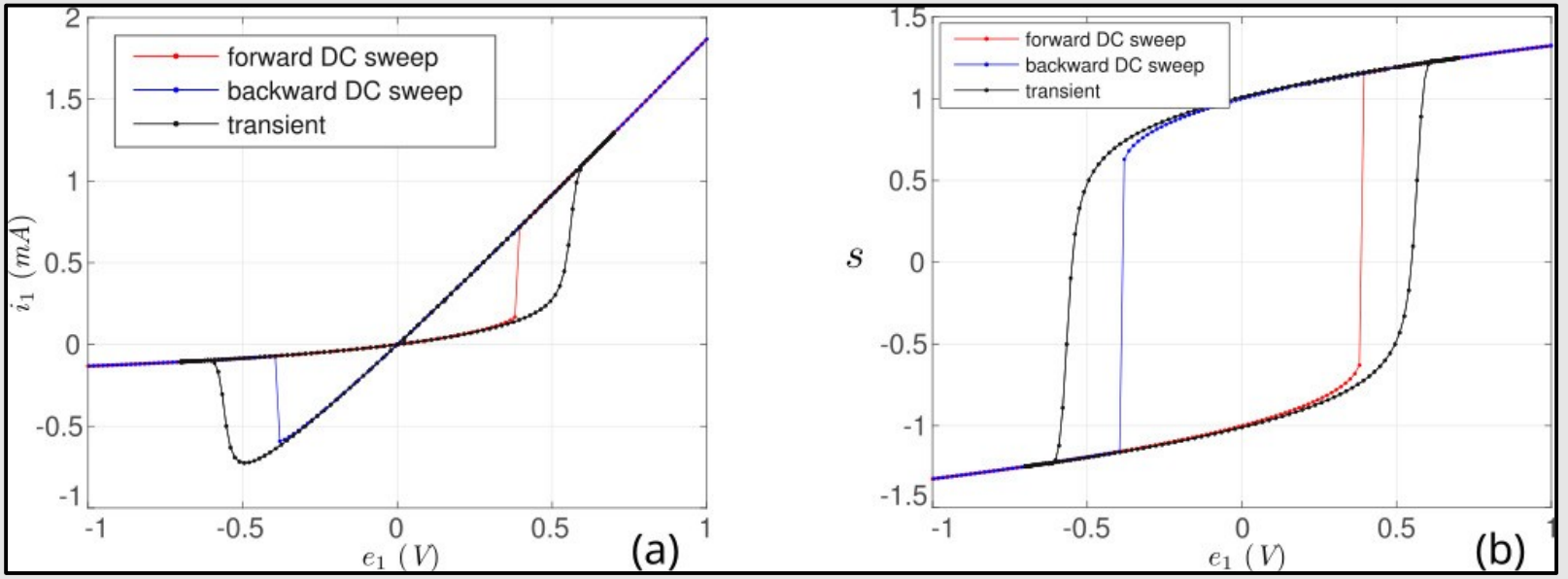
$$\frac{d}{dt}\mathbf{s} = f_2\left(\mathbf{vpn},\ \mathbf{s}\right)$$

ModSpec:

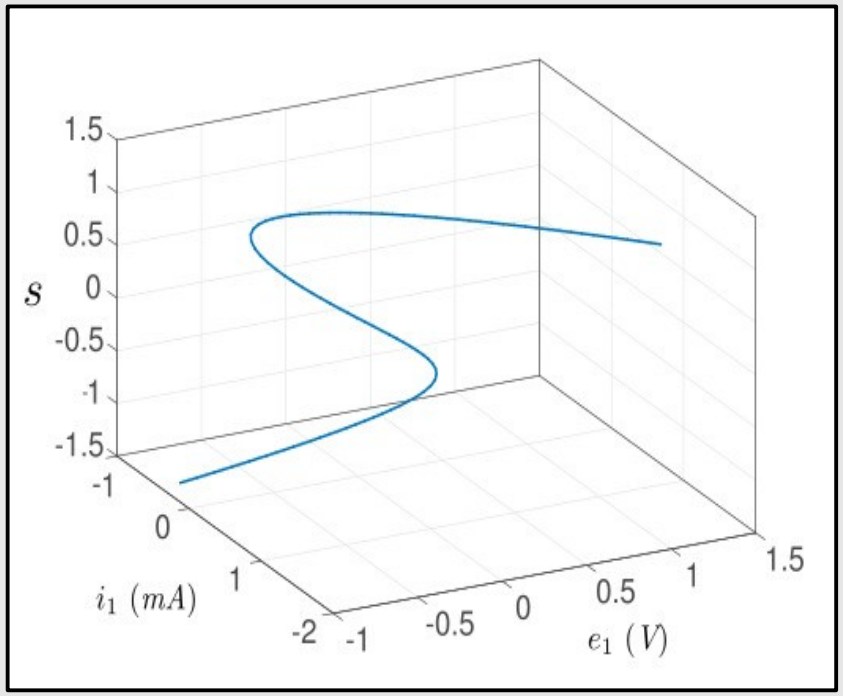
$$\mathbf{ipn} = \frac{d}{dt}\underbrace{q_e\left(\mathbf{vpn},\ \mathbf{s}\right)}_{\mathbf{0}} + \underbrace{f_e\left(\mathbf{vpn},\ \mathbf{s}\right)}_{f_1}$$

$$0 = \frac{d}{dt}\underbrace{q_i\left(\mathbf{vpn},\ \mathbf{s}\right)}_{-\mathbf{s}} + \underbrace{f_i\left(\mathbf{vpn},\ \mathbf{s}\right)}_{f_2}$$



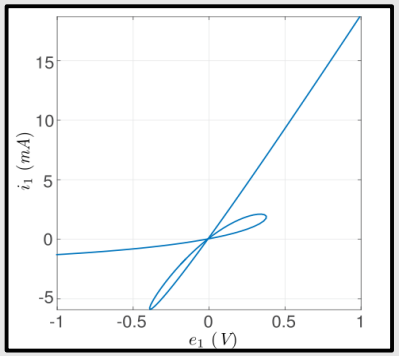


How to Model Hysteresis Properly

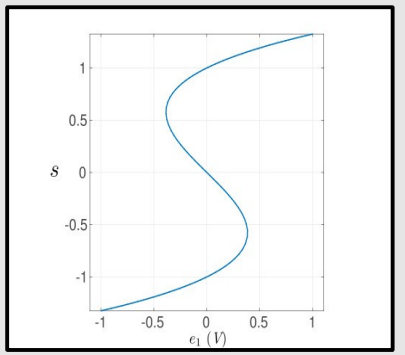


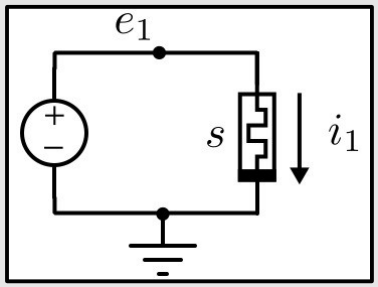
homotopy

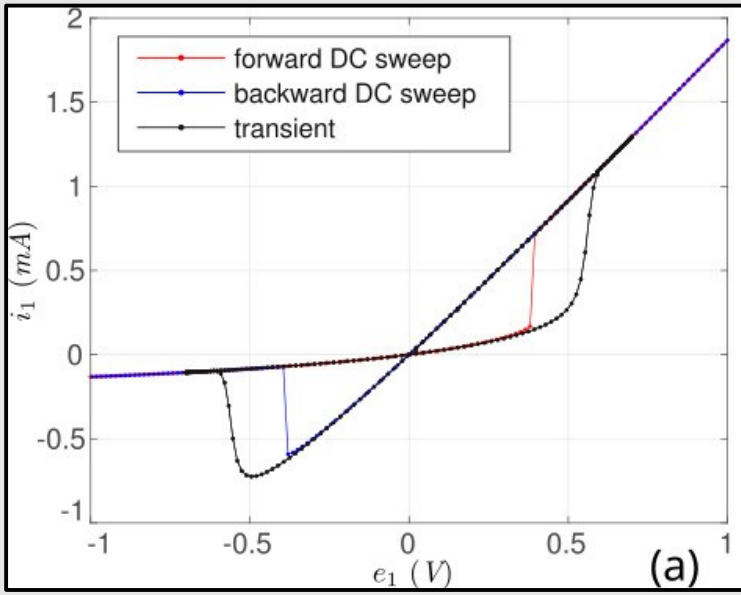
top

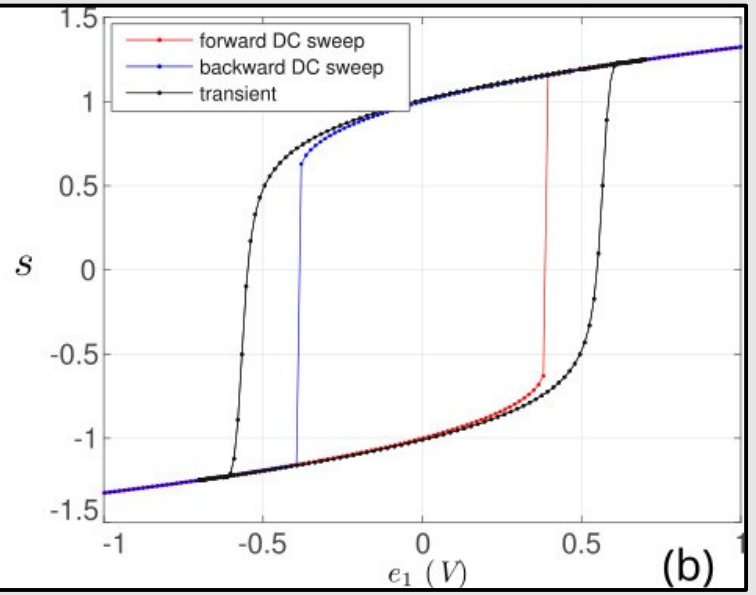


side









Internal Unknowns in Verilog-A

Template:

$$\mathbf{ipn} = f_1(\mathbf{vpn}, \mathbf{s})$$

$$\frac{d}{dt}\mathbf{s} = f_2(\mathbf{vpn}, \mathbf{s})$$

?

Example:

$$f_1(\mathbf{vpn}, \mathbf{s}) = \frac{\mathbf{vpn}}{R} \cdot (1 + \tanh(\mathbf{s}))$$

$$f_2(\mathbf{vpn}, \mathbf{s}) = \mathbf{vpn} - \mathbf{s}^3 + \mathbf{s}$$

DO NOT

- declare internal unks as "`real`" variables
- code time integration inside model
  - with `$abstime, @initial_step` and memory states
- use `idt()`
- use implicit contributions
  - unless you know what you are doing

Internal Unknowns in Verilog-A

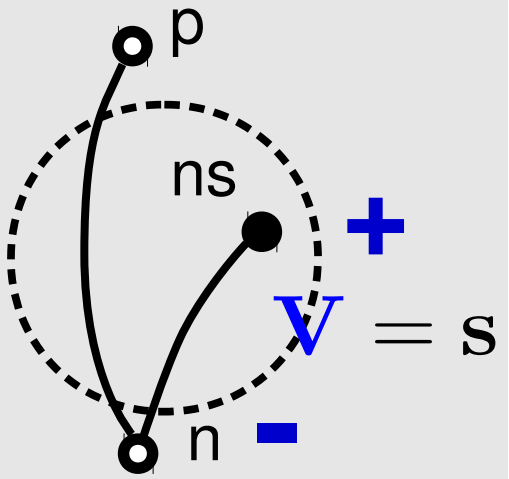
$$\mathbf{ipn} = \frac{\mathbf{vpn}}{R} \cdot (1 + \tanh(\mathbf{s}))$$

$$\frac{d}{dt}(\tau \cdot \mathbf{s}) = \mathbf{vpn} - \mathbf{s^3} + \mathbf{s}$$

use a potential or flow

```
 1 `include "disciplines.vams"
 2 module hys(p, n);
 3     inout p, n;
 4     electrical p, n, ns;
 5     parameter real R = 1e3 from (0:inf);
 6     parameter real k = 1 from (0:inf);
 7     parameter real tau = 1e-5 from (0:inf);
 8     real s;
 9
10     analog begin
11         s = V(ns, n);
12         I(p, n)  <+ V(p, n)/R * (1+tanh(k*s));
13         I(ns, n) <+ V(p, n) - pow(s, 3) + s;
14         I(ns, n) <+ ddt(-tau*s);
15     end
16 endmodule
```

internal node



internal unknown

implicit differential equation

RRAM Model

Template:

$$\mathbf{ipn} = f_1\,(\mathbf{vpn},\ \mathbf{s})$$

$$\frac{d}{dt}\mathbf{s} = f_2\,(\mathbf{vpn},\ \mathbf{s})$$

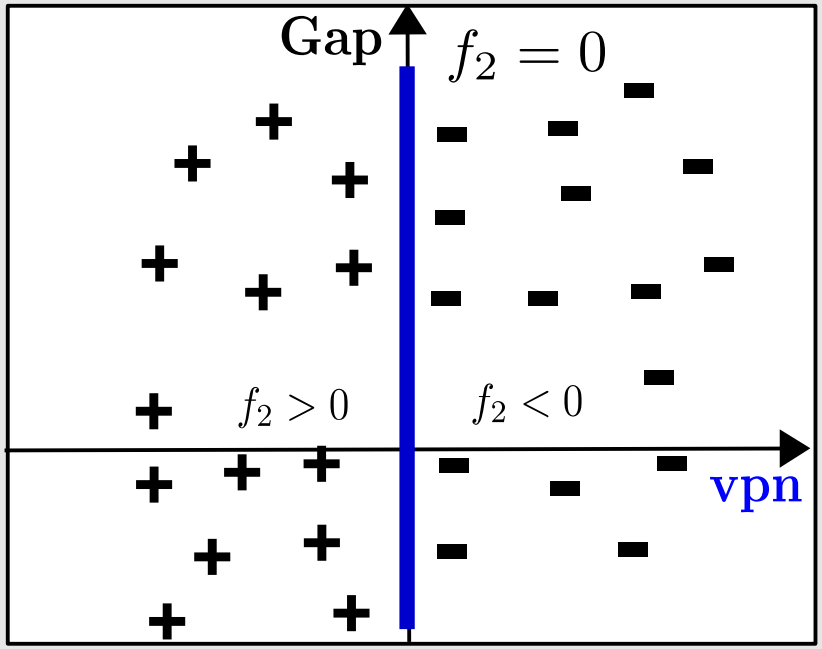
RRAM:

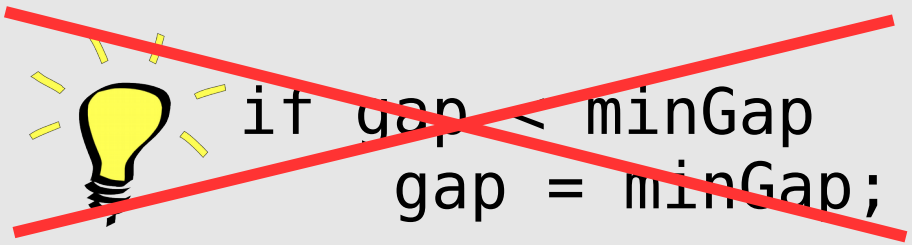
$$f_1\,(\mathbf{vpn},\ \mathbf{Gap}) = I_0 \cdot e^{-\mathbf{Gap}/g0} \cdot \sinh(\mathbf{vpn}/V_0)$$

$$f_2\,(\mathbf{vpn},\ \mathbf{Gap}) = -v_0 \cdot \exp(-\frac{E_a}{V_T}) \cdot \sinh(\frac{\mathbf{vpn} \cdot \gamma \cdot a_0}{t_{ox} \cdot V_T})$$

Jiang, Z., Wong, H. (2014). Stanford University Resistive-Switching Random Access Memory (RRAM) Verilog-A Model. nanoHUB.

$$\mathbf{minGap} \leq \mathbf{Gap} \leq \mathbf{maxGap}$$





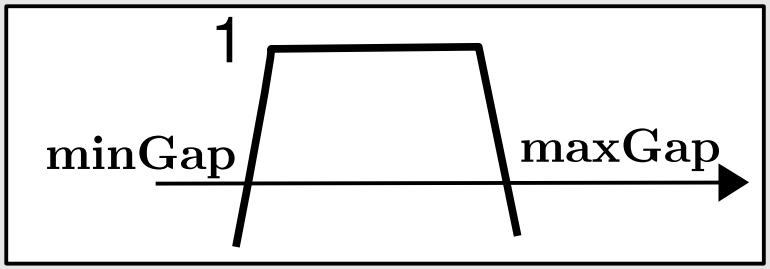
hybrid model

RRAM Model

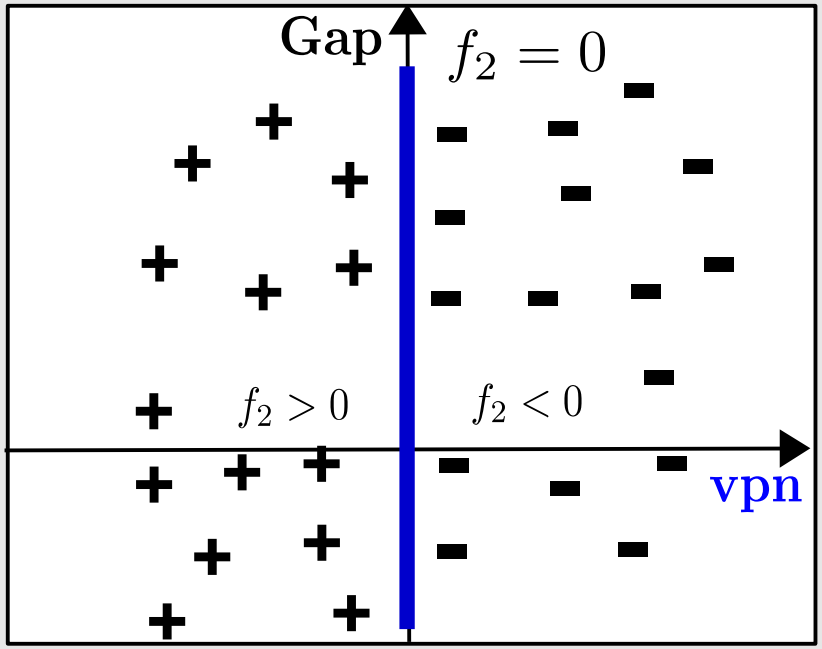
Template: **RRAM:**

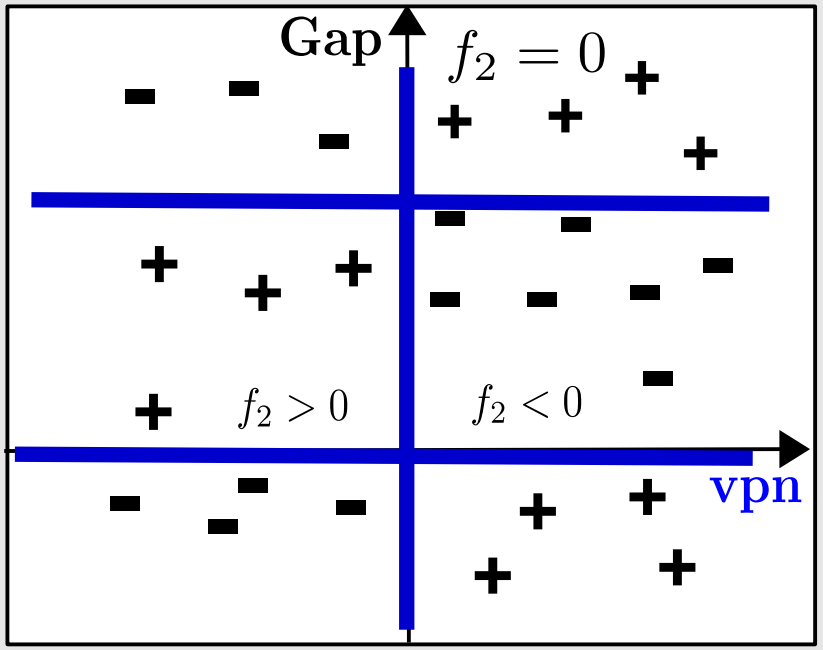
$$\mathbf{ipn} = f_1(\mathbf{vpn}, \mathbf{s}) \qquad f_1(\mathbf{vpn}, \mathbf{Gap}) = I_0 \cdot e^{-\mathbf{Gap}/g0} \cdot \sinh(\mathbf{vpn}/V_0)$$

$$\frac{d}{dt}\mathbf{s} = f_2(\mathbf{vpn}, \mathbf{s}) \qquad f_2(\mathbf{vpn}, \mathbf{Gap}) = -v_0 \cdot \exp(-\frac{E_a}{V_T}) \cdot \sinh(\frac{\mathbf{vpn} \cdot \gamma \cdot a_0}{t_{ox} \cdot V_T}) \times F_{window}(\mathbf{Gap})$$

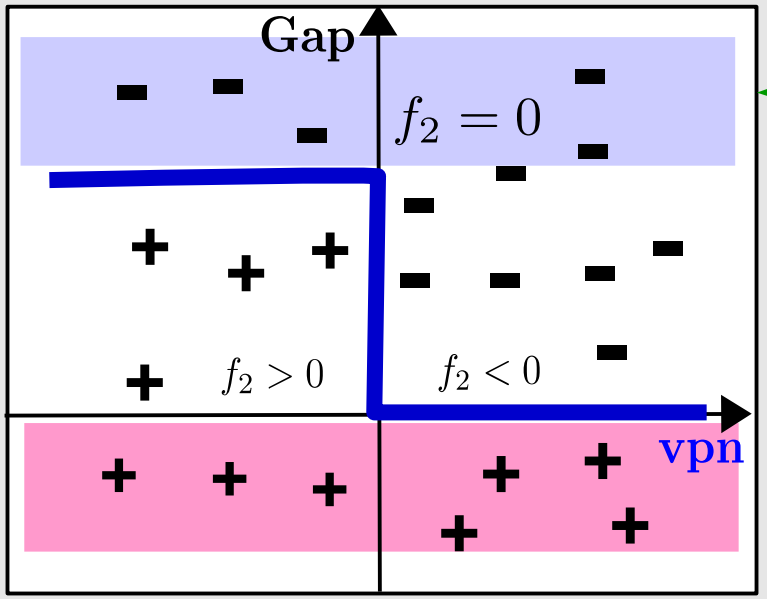


Biolek, Jogelkar, Prodromakis, UMich, TEAM/VTEAM, Yakopcic, etc.





RRAM Model

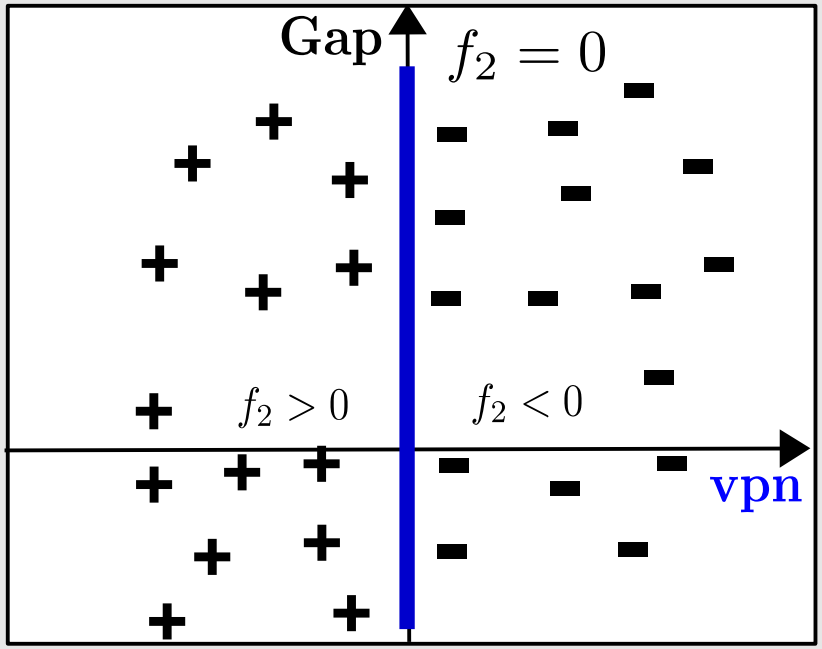


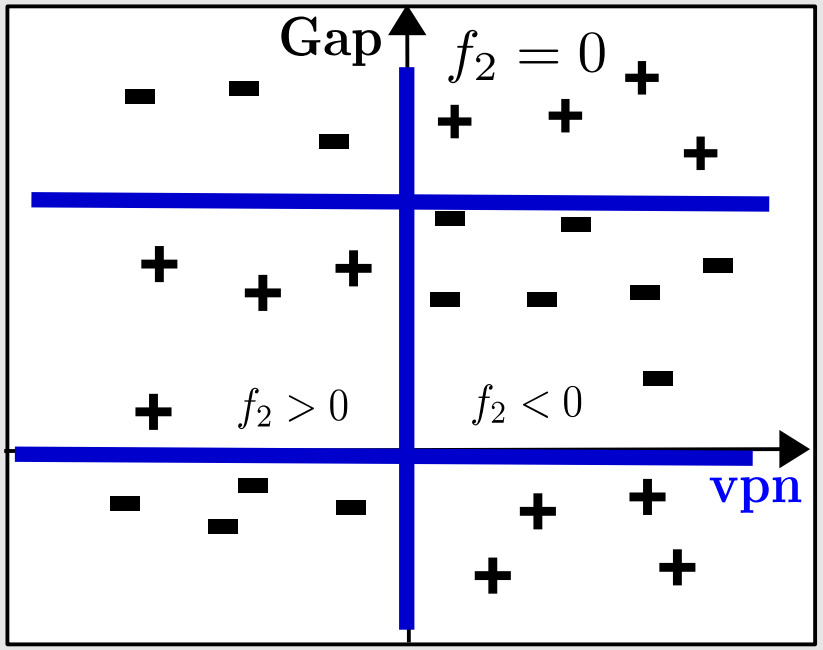
clipping functions

Analogy: MEMS switch
Zener diode voltage regulator

Guan X, Yu S, Wong H S. A SPICE compact model of metal oxide resistive switching memory with variations[J]. IEEE electron device letters, 2012.

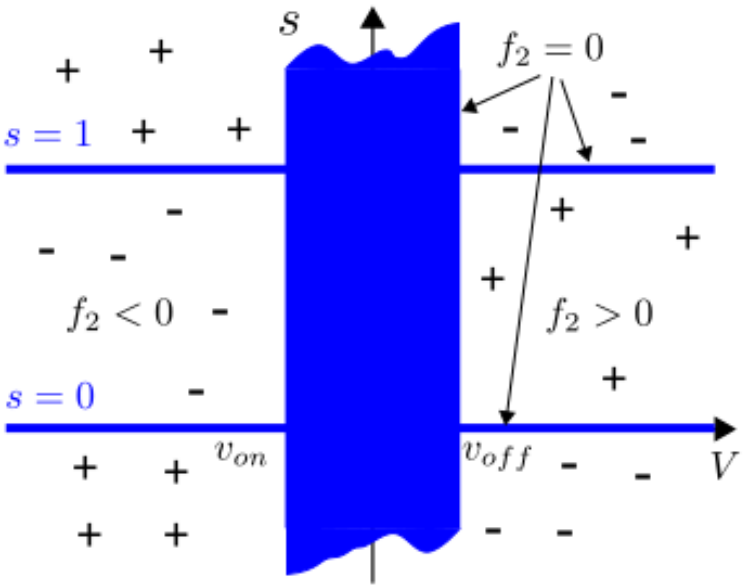
Vourkas I, Sirakoulis G C. Memristor-Based Nanoelectronic Computing Circuits and Architectures[M]. Springer, 2015.



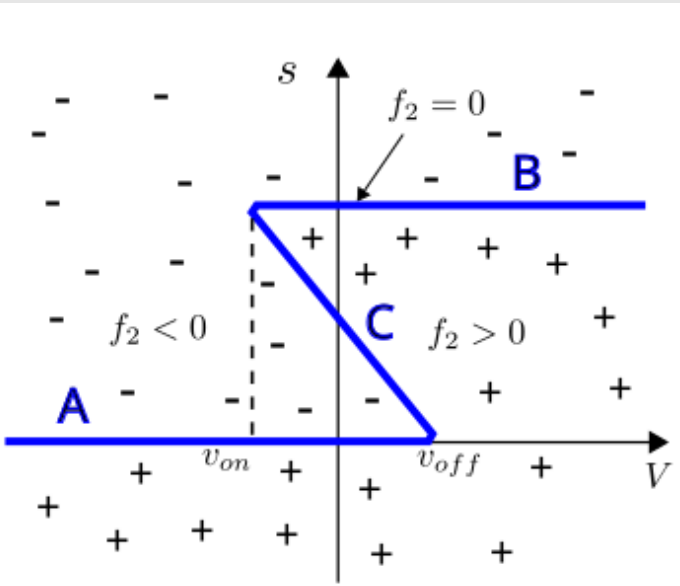


Memristor Models

Another (deeper) problem with f_2



TEAM/VTEAM, Yakopcic models



fix flat f_2 region

$$f_2 = \begin{cases} k_{off} \cdot \left(\frac{vpn}{v_{off}} - 1\right)^{\alpha_{off}}, & \text{if } vpn > v_{off} \\ k_{on} \cdot \left(\frac{vpn}{v_{on}} - 1\right)^{\alpha_{on}}, & \text{if } vpn < v_{on} \\ 0, & \text{otherwise} \end{cases}$$

$$f_2 = \begin{cases} k_{off} \cdot \left(\frac{vpn - v^*}{v_{off}}\right)^{\alpha_{off}}, & \text{if } vpn > v^* \\ k_{on} \cdot \left(\frac{vpn - v^*}{v_{on}}\right)^{\alpha_{on}}, & \text{otherwise,} \end{cases}$$

where

$$v^* = (1 - s) \cdot v_{off} + s \cdot v_{on}$$

Memristor Models

$$\frac{d}{dt}\mathbf{s} = f_2\left(\mathbf{vpn},\ \mathbf{s}\right)$$

Available f2:

1. linear ion drift

$$f_2 = \mu_v \cdot R_{on} \cdot f_1(\mathbf{vpn},\ s)$$

2. nonlinear ion drift

$$f_2 = a \cdot \mathbf{vpn}^m$$

3. Simmons tunnelling barrier

$$f_2 = \begin{cases} c_{off} \cdot \sinh(\frac{i}{i_{off}}) \cdot \exp(-\exp(\frac{s-a_{off}}{w_c} - \frac{i}{b}) - \frac{s}{w_c}), & \text{if } i \geq 0 \\ c_{on} \cdot \sinh(\frac{i}{i_{on}}) \cdot \exp(-\exp(\frac{a_{on}-s}{w_c} + \frac{i}{b}) - \frac{s}{w_c}), & \text{otherwise,} \end{cases}$$

4. TEAM model

5. Yakopcic model

6. Stanford/ASU

$$f_2 = -v_0 \cdot \exp(-\frac{E_a}{V_T}) \cdot \sinh(\frac{\mathbf{vpn} \cdot \gamma \cdot a_0}{t_{ox} \cdot V_T})$$

$$\mathbf{ipn} = f_1\left(\mathbf{vpn},\ \mathbf{s}\right)$$

Available f1:

1. $f_1 = (R_{on} \cdot s + R_{off} \cdot (1-s))^{-1} \cdot \mathbf{vpn}$

2. $f_1 = \frac{1}{R_{on}} \cdot e^{-\lambda \cdot (1-s)} \cdot \mathbf{vpn}$

3. $f_1 = s^n \cdot \beta \cdot \sinh(\alpha \cdot \mathbf{vpn}) + \chi \cdot (\exp(\gamma \cdot) - 1)$

4. $\mathrm{f}_1 = \begin{cases} A_1 \cdot s \cdot \sinh(B \cdot \mathbf{vpn}), & \text{if } \mathbf{vpn} \geq \mathbf{0} \\ A_2 \cdot s \cdot \sinh(B \cdot \mathbf{vpn}), & \text{otherwise.} \end{cases}$

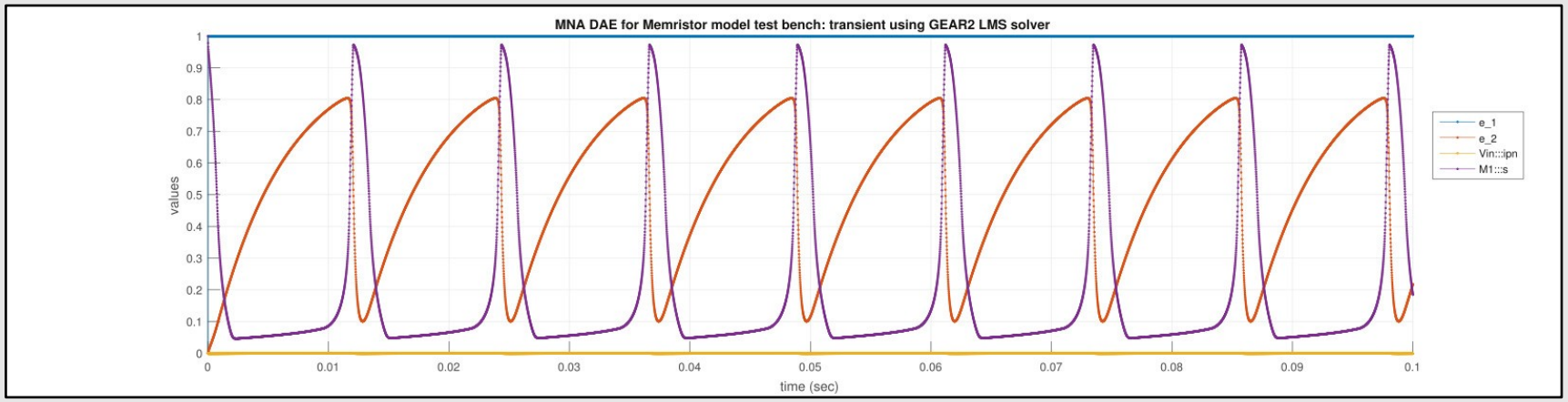
5. $f_1 = I_0 \cdot e^{-\mathbf{Gap}/g0} \cdot \sinh(\mathbf{vpn}/V_0)$
   $\mathbf{Gap} = s \cdot minGap + (1-s) \cdot maxGap.$

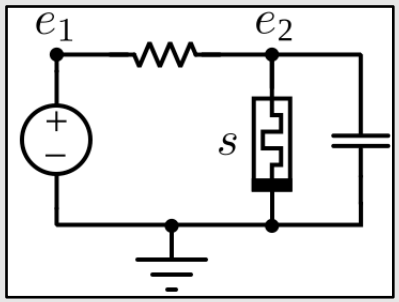
- **set up boundary**
- **fix f2 flat regions**
- **smooth, safe funcs, scaling, etc.**

Memristor Models

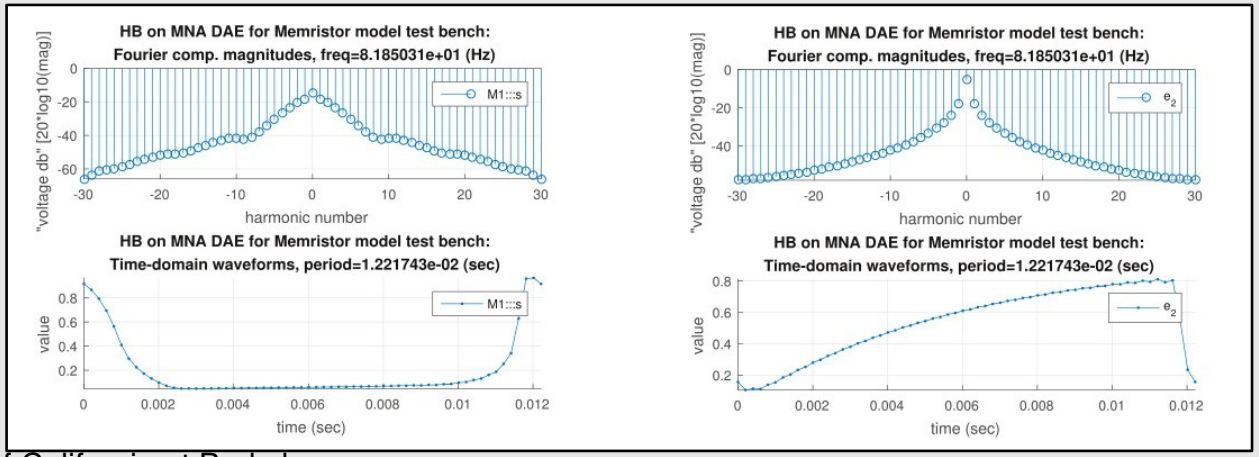
A collection of 30 models:

- all smooth, all well posed
- not just RRAM, but general memristive devices
- not just bipolar, but unipolar
- not just DC, AC, TRAN, but homotopy, PSS, ...





PSS using HB



Numerical Explosion

RRAM:

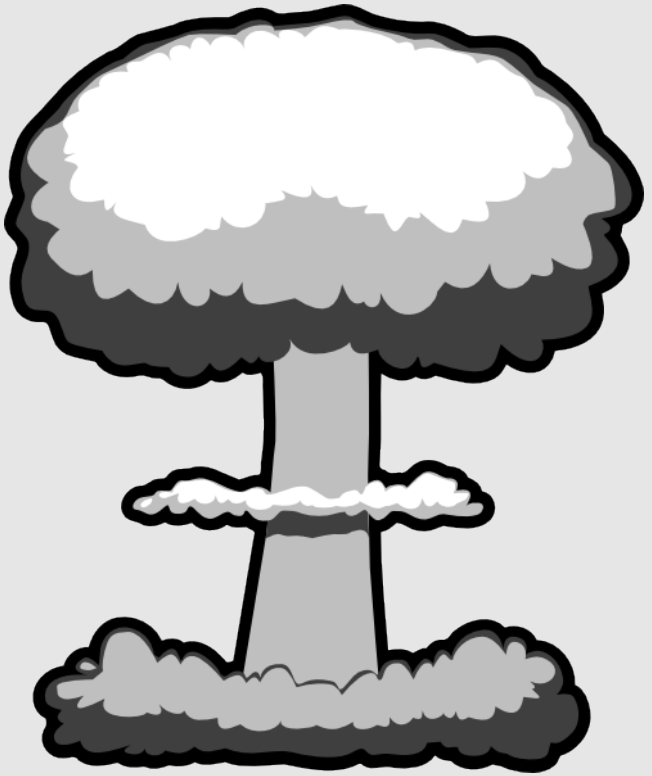
$$f_1\,(\mathbf{vpn},\ \mathbf{Gap}) = I_0 \cdot e^{-\mathbf{Gap}/g0} \cdot \sinh(\mathbf{vpn}/V_0)$$

$$f_1\,(0.25V,\ 0) = 1mA.$$

$$f_1\,(1V,\ 0) = 27.3mA.$$

$$f_1\,(10V,\ 0) = 1.17 \times 10^{14}A.$$

$$f_1\,(100V,\ 0) = 2.61 \times 10^{170}A.$$



similar to the diode equation:

$$diode(Vd) = I_S \cdot (\exp(Vd/V_T) - 1).$$

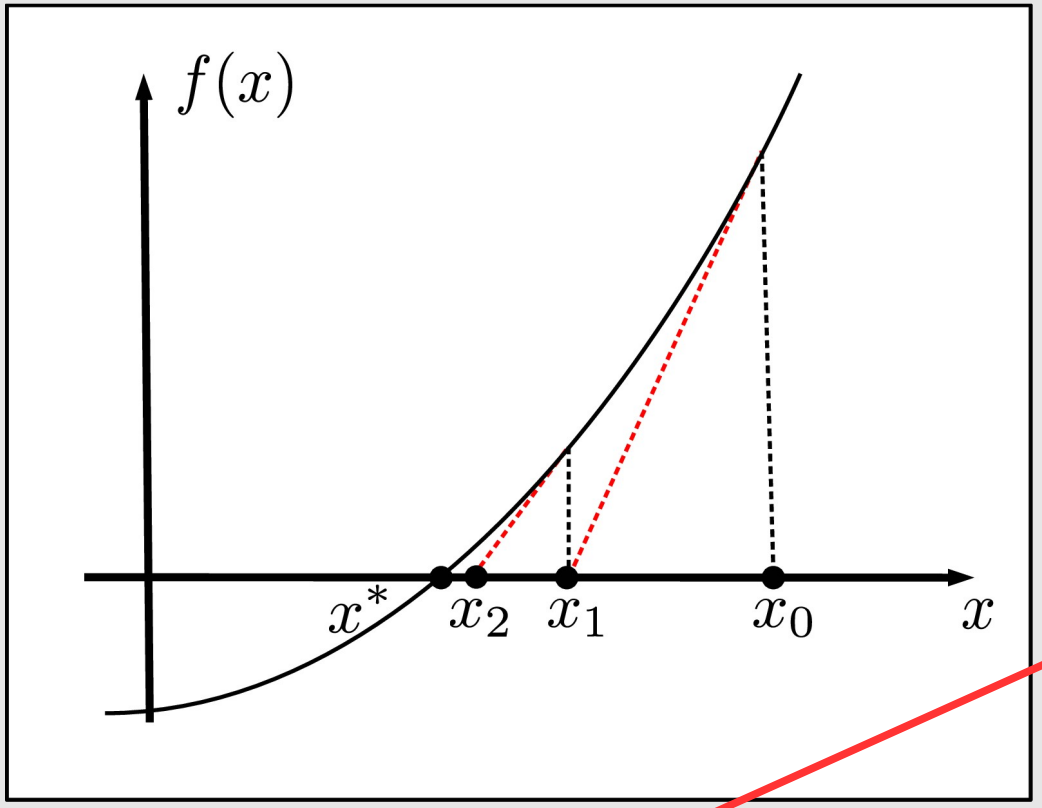
$$diode(0.7) \approx 0.5A.$$

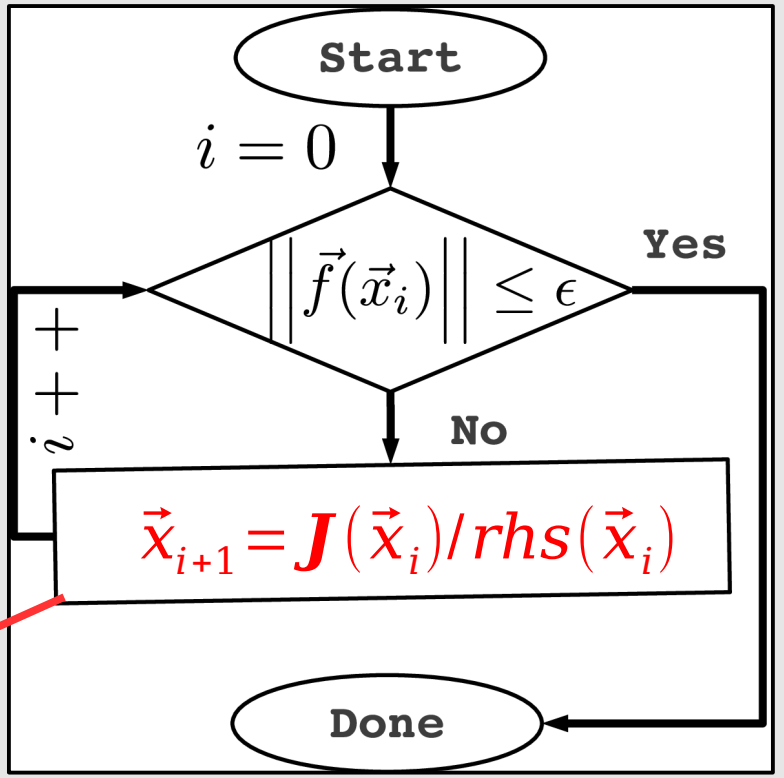
$$diode(1) \approx 50000A.$$

$$diode(10) \approx 10^{155}A.$$

$limexp()?

What does SPICE do?





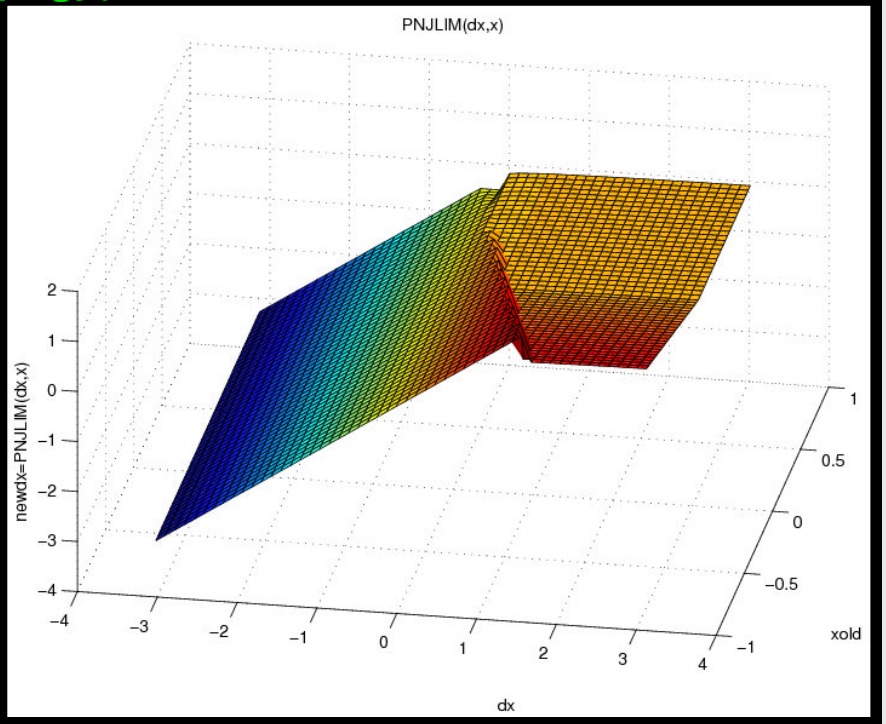
change Vd "arbitrarily"

{ if (there is IC)
if (first NR step)
if (Vd too large)
...

based on Vd_old

xlim = pnjlim(xnew, xold)

```
199 double DeviceSupport::pnjlim (
200     double vnew,
201     double vold,
202     double vt,
203     double vcrit,
204     int *icheck
205   )
206 {
207   double arg;
208   if((vnew > vcrit) && (fabs(vnew - vold) > (vt + vt)))
209   {
210     if(vold > 0)
211     {
212       arg = 1 + (vnew - vold) / vt;
213       if(arg > 0)
214       {
215         vnew = vold + vt * log(arg);
216       }
217       else
218       {
219         vnew = vcrit;
220       }
221     }
222     else
223     {
224       vnew = vt *log(vnew/vt);
225     }
226     *icheck = 1;
227   }
228   else
229   {
230     *icheck = 0;
231   }
232   return(vnew);
233 }
```



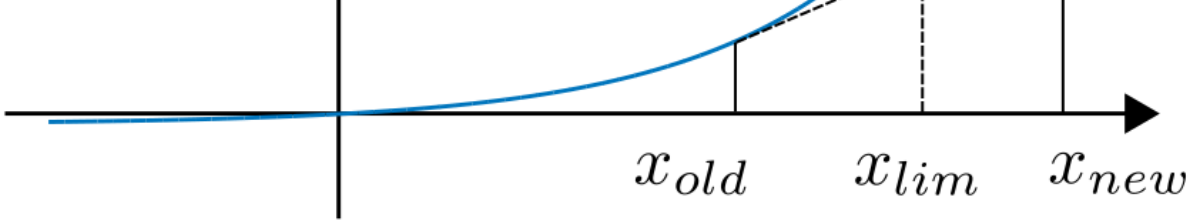
xlim = limvds(xnew, xold)

```
172 double DeviceSupport::limvds ( double vnew, double vold)
173 {
174
175   if(vold >= 3.5)
176   {
177     if(vnew > vold) vnew = std::min(vnew,(3.0 * vold) + 2.0);
178     else
179     {
180       if (vnew < 3.5) vnew = std::max(vnew,2.0);
181     }
182   }
183   else
184   {
185     if(vnew > vold) vnew = std::min(vnew, 4.0);
186     else            vnew = std::max(vnew,-0.5);
187   }
188   return(vnew);
189 }
```

xlim = pnjlim(xnew, xold)

$$f(x) = I_S \cdot (e^{\frac{x}{V_T}} - 1)$$

$$x_{lim} = \text{pnjlim\_core}(x_{new},\ x_{old},\ V_T) = x_{old} + V_T \cdot \ln\left(1 + \frac{x_{new} - x_{old}}{V_T}\right).$$

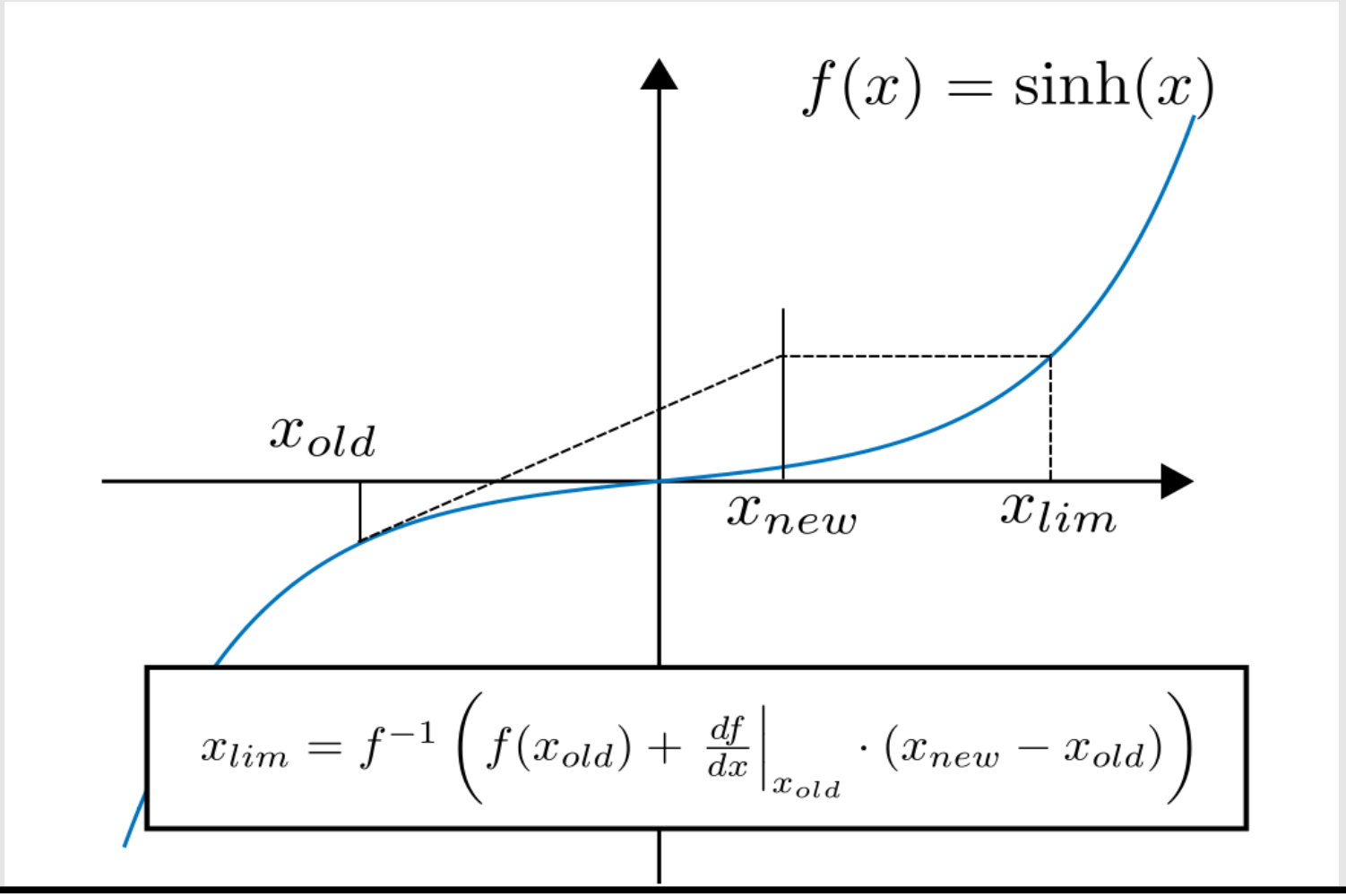


Hierarchical Newton? limvds(), fetlim() are similar

concave function, magic numbers inside

(are they outdated?)

xlim = sinhlim(xnew, xold)

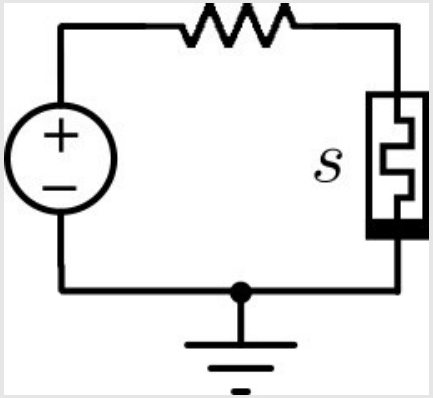


$$x_{lim} = f^{-1}\left(f(x_{old}) + \left.\frac{df}{dx}\right|_{x_{old}} \cdot (x_{new} - x_{old})\right)$$

$$\sinh(x_{lim}) = y_{lim} = \sinh(k \cdot x_{old}) + k \cdot \cosh(k \cdot x_{old}) \cdot (x_{new} - x_{old}),$$

$$x_{lim} = \text{sinhlim}(x_{new},\ x_{old},\ k) = \frac{1}{k} \cdot \ln\left(y_{lim} + \sqrt{1 + y_{lim}^2}\right).$$

xlim = sinhlim(xnew, xold)



$\longleftarrow I = \sinh(V).$

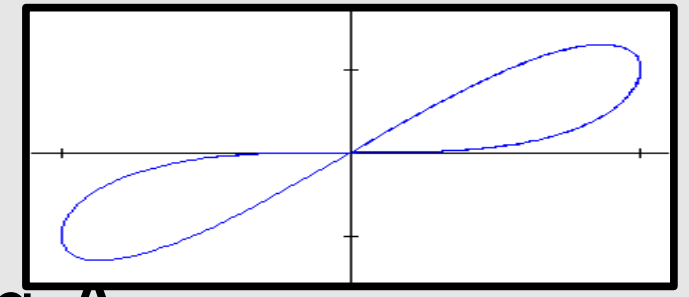
Supply Voltage (V)	with `sinhlim` (niters)	without limiting (niters)
1	4	4
10	4	9
100	4	50
1000	4	non-convergence within 100 iters

actual implementation slightly more complicated

$$f_1\,(\mathbf{vpn},\ \mathbf{Gap}) = I_0 \cdot e^{-\mathbf{Gap}/g0} \cdot \sinh(\mathbf{vpn}/V_0)$$

$$f_2\,(\mathbf{vpn},\ \mathbf{Gap}) = -v_0 \cdot \exp(-\frac{E_a}{V_T}) \cdot \sinh(\frac{\mathbf{vpn} \cdot \gamma \cdot a_0}{t_{ox} \cdot V_T})$$

Challenges in Memristor Modelling



- hysteresis
  - internal state variable
- model internal unks in Verilog-A
  - use potentials/flows
- upper/lower bounds of internal unks
  - filament length, tunneling tap size
  - clipping functions
- smoothness, continuity, finite precision issues, ...
  - use smooth functions, safe functions
  - GMIN
  - scaling of unks/eqns
  - SPICE-compatible limiting function (the only smooth one)