The MVS Nanotransistor Model: A Case Study in Compact Modeling

Shaloo Rakheja and Dimitri Antoniadis

Microsystems Technology Laboratories, MIT

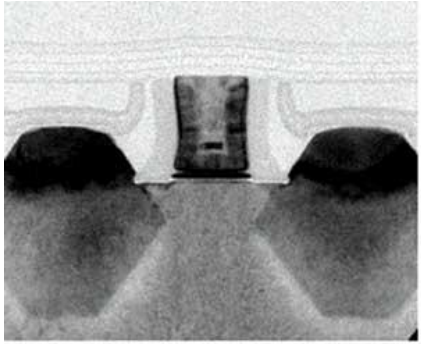
November 13, 2014

Thanks to Dr. Geoffrey Coram and Prof. Jaijeet Roychowdhury

Massachusetts Institute of Technology

shaloo@mit.edu

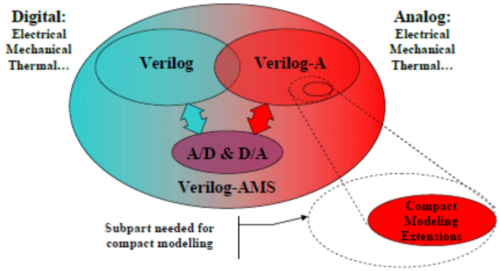
This presentation focuses on



I. MVS model

- Basic model formulation
- Mathematical issues

35 min



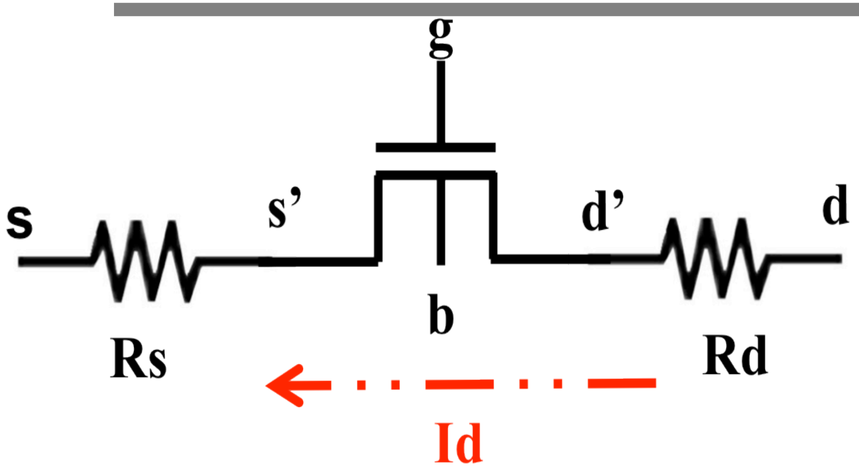
II. Model implementation in Verilog-A

- Performance-limiting constructs
- Examples from MVS

15 min

PART I
MVS MODEL FORMULATION

What is MVS model?



MIT **V**irtual **S**ource (MVS) nanotransistor model gives *currents* and *charges* as functions of terminal voltages.

Currents

Id = f(Vg,Vd,Vs,Vb)

Ig = Ib = 0

MVS is a **source-referenced** model.

Charges

Qs = f1(Vg,Vd,Vs,Vb)

Qd = f2(Vg,Vd,Vs,Vb)

Qb = f3(Vg,Vd,Vs,Vb)

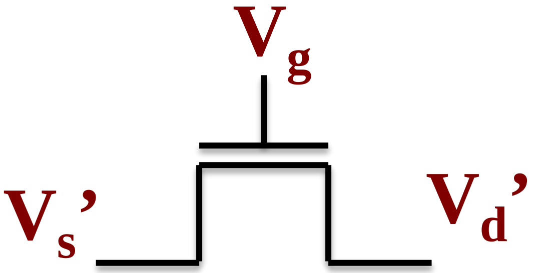
Qg = -(Qs+Qd+Qb)

DC Model

$$\frac{I_D}{W} = Q_{x0} v_{x0} F_{sat} \longrightarrow \text{Empirical function}$$

Charge at VS

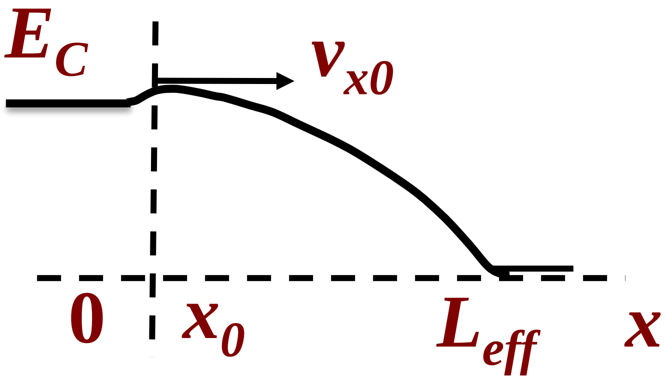
Velocity at VS



→ 10 fitting parameters.

→ most of the parameters are physical and can easily be obtained through device characterization.

→ describes quasi-ballistic **silicon, III-V and graphene devices.**



Dynamic MVS model

- Valid in **quasi-static** conditions in the channel.
- At low V_{ds}, transport can be modeled as **drift-diffusion** with no velocity saturation (**DD-NVSAT**).
- Quasi-ballistic and DD-NVSAT charges are blended w/ F_{sat}^2.

Ward-Dutton charge partitioning scheme

$$Q_S = \int_0^{L_g} \left(1 - \frac{x}{L_g}\right) Q_e(x)\,dx$$

$$Q_D = \int_0^{L_g} \left(\frac{x}{L_g}\right) Q_e(x)\,dx$$

Quasi-ballistic charges

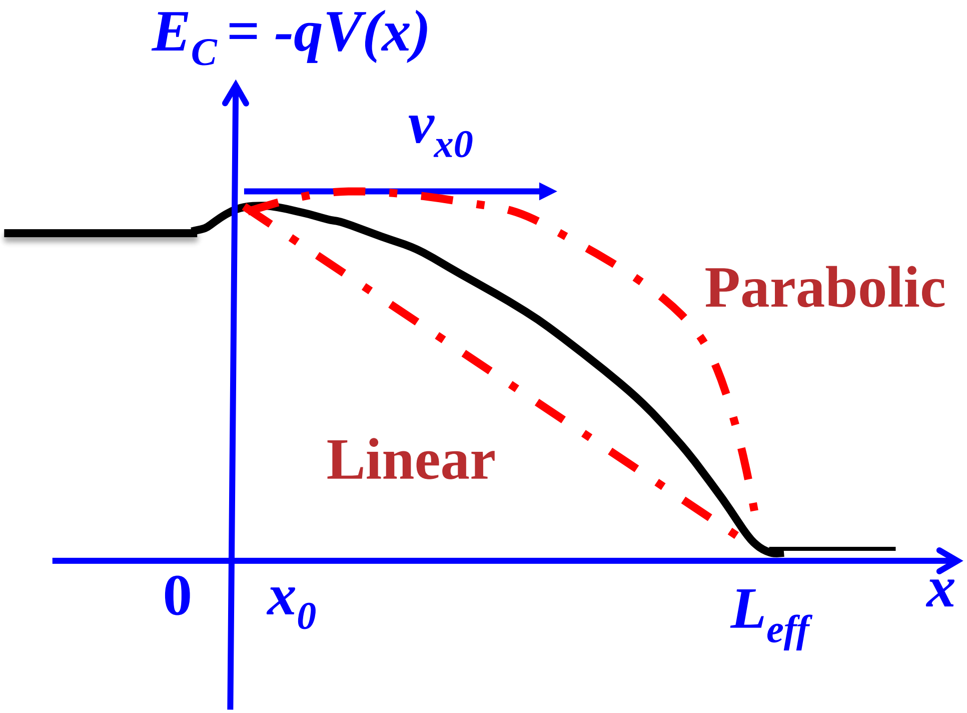
Current continuity

$$Q_{x0} v_{x0} = Q_e(x) v_x(x)$$

Energy balance

$$\frac{1}{2} m^* v_{x0}^2 + qV(x)\zeta = \frac{1}{2} m^* v_x(x)^2$$

$0 \leq \zeta \leq 1$: Fraction of V_{ds} energy gained by carriers.



Dynamic MVS model

$$Q_S = Q_{S,ballistic}F_{satq}^2 + Q_{S,DD}\left(1 - F_{satq}^2\right) + Q_{S,ov} + Q_{S,if}$$

$$Q_D = Q_{D,ballistic}F_{satq}^2 + Q_{D,DD}\left(1 - F_{satq}^2\right) + \underbrace{Q_{D,ov} + Q_{D,if}}$$

$$Q_G = -\left(Q_S + Q_D + Q_B\right)$$

Parasitic fringing charges

- **Option** to choose between only the **DD-NVSAT** charge model or **blended QB** charge model.
- Body charge, Q_B, is calculated using approx. surface potential formulation [check Tsividis].

Dynamic MVS model

$$Q_S = Q_{S,ballistic}F_{satq}^2 + Q_{S,DD}\left(1 - F_{satq}^2\right) + Q_{S,ov} + Q_{S,if}$$

$$Q_D = Q_{D,ballistic}F_{satq}^2 + Q_{D,DD}\left(1 - F_{satq}^2\right) + Q_{D,ov} + Q_{D,if}$$

$$Q_G = -\left(Q_S + Q_D + Q_B\right)$$

- Capacitance is the slope of charges with respect to voltages.

$$C_{ij} = -\frac{\partial Q_i}{\partial V_j}(i \neq j)$$

$$C_{jj} = \frac{\partial Q_j}{\partial V_j}$$

Charge Smoothness issues ??

References for MVS model equations

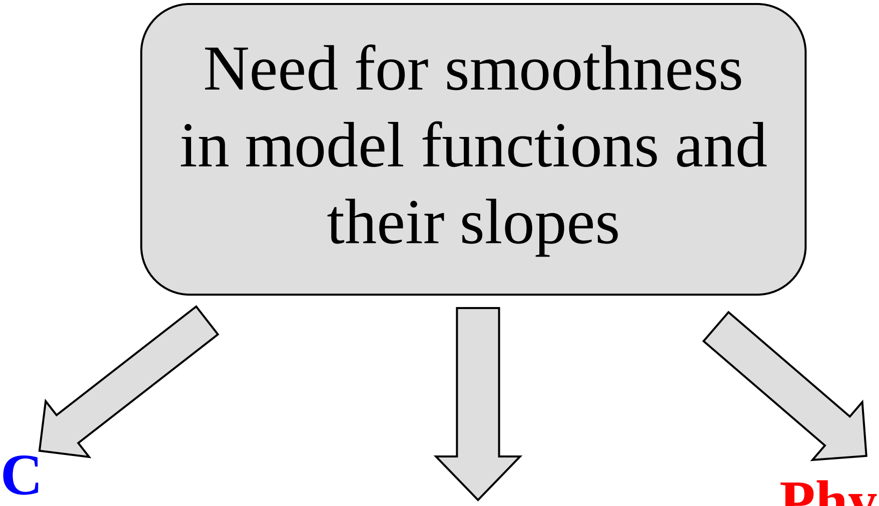
1. A. Khakifirooz et al., "A simple semi-empirical short-channel MOSFET current-voltage model continuous across all regions of operation and employing only physical parameters," IEEE Trans. Electron Devices, vol. 56, no. 8, **July 2009**.

2. L. Wei et al., " Virtual-source-based self-consistent current and charge FET models: from ballistic to drift-diffusion velocity-saturation operation," IEEE Trans. Electron Devices, vol. 59, no. 5, **May 2012**.

3. S. Rakheja and D. Antoniadis, "MVS 1.0.1 Nanotransistor Model (Silicon)," https://nanohub.org/resources/19684 (**Nov. 2013**)

MATHEMATICAL ISSUES IN MVS MODEL

“Smoothness” is key in compact modeling



Need for smoothness in model functions and their slopes

DC/transient/AC analysis of circuits

Small-signal resistance/capacitance/inductance

Physical systems are smooth at fine enough resolution

“A quick circuit simulation primer” https://nanohub.org/resources/20610

Fundamentals: continuity

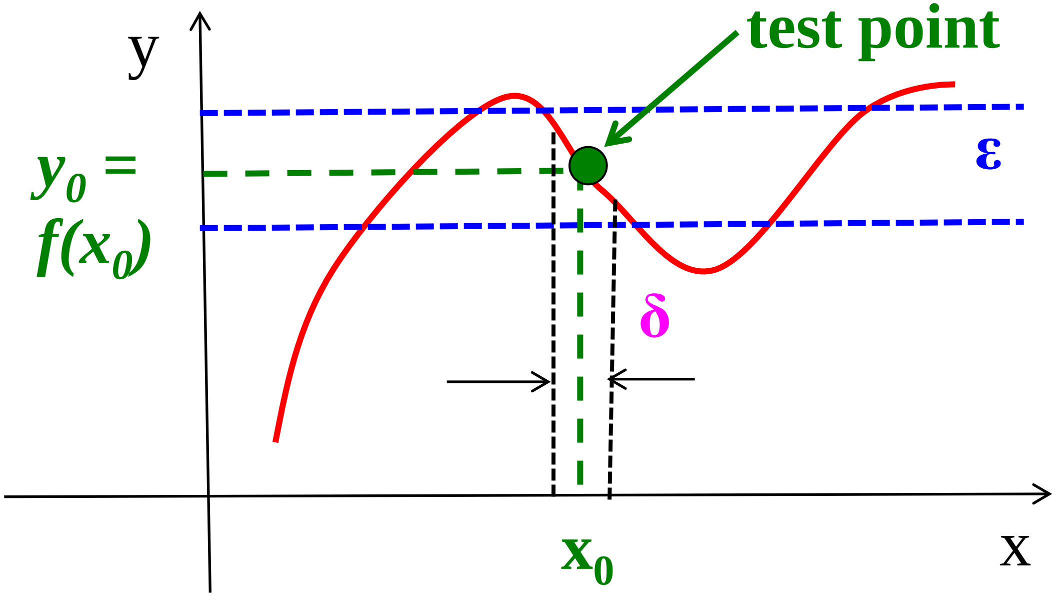
***f(x)* is continuous at x_0 if:**

given any ***ε > 0***

we can always find ***δ > 0*** such that:

$|f(x)-f(x_0)| < \varepsilon$

for all x satisfying $|x-x_0| < \delta$



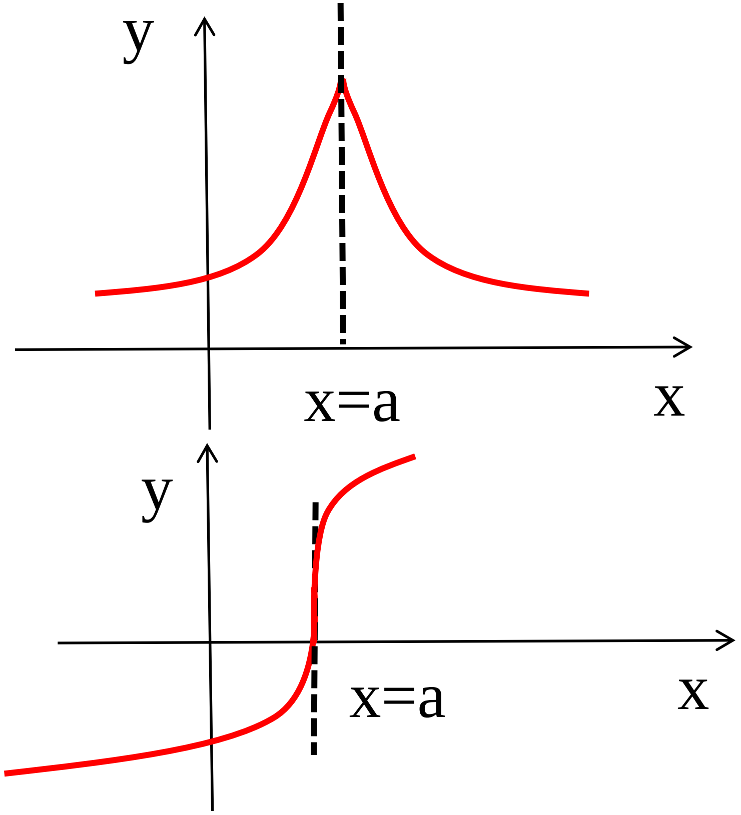
Fundamentals: differentiability

Derivative:

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

Function *f(x)* is differentiable if:
f'(x) exists at all *x* and is continuous

A function can fail to be differentiable at a point if either there is a ***cusp*** in the graph or a ***point of vertical tangency***.



Causes of non-smoothness in models

- Idealization
  - Look out for “if” conditions
- Beware of constructs that blow up
  - Ex: **y=1/(x+a)**has a problem at **x=-a**
  - Ex: **y=log(x); dy/dx = 1/x** has a problem at x=0
- Examples of non-smooth functions:
  - **sign, abs, max, min**
- Empirical functions to stitch various regions of operation often lead to non-differentiability.

“Dealing with common numerical issues in compact models”
https://nanohub.org/resources/21262

Causes of non-smoothness in models

- Idealization
  - Look out for “if” conditions
- **Beware of constructs that blow up**
  - **Ex: y=1/(x+a)has a problem at x=-a**
  - **Ex: y=log(x); dy/dx = 1/x has a problem at x=0**
- Examples of non-smooth functions:
  - **sign, abs, max, min**
- Empirical functions to stitch various regions of operation often lead to non-differentiability.

Example from MVS

Source terminal charge in quasi-ballistic case in MVS

$$Q_{SB}=Q_{inv}\frac{(4k+4)\sqrt{k+1}-(6k+4)}{3k^2}$$

$$k=\frac{2q}{m^*}\frac{V_{ds}}{v_{x0}^2}$$

At V_{ds} = 0V, Q_{sb} will not exist → clearly a problem.

How can this be fixed?

Example from MVS

Taking limits

$$\lim_{V_{ds}\to 0} Q_{SB}\left(V_{ds}\right)=Q_{inv}\left(0.5-\frac{k}{24}+\frac{k^2}{80}\right)$$

```
if($V_{ds}<1e-3$)
    $Q_{sb}=Q_{inv}\left(0.5-\frac{k}{24}+\frac{k^2}{80}\right)$
else
    $Q_{sb}=Q_{inv}\frac{\left(4k+4\right)\sqrt{k+1}-\left(6k+4\right)}{3k^2}$
end
```

From MVS implementation

Causes of non-smoothness in models

- Idealization
  - Look out for "if" conditions
- Beware of constructs that blow up
  - Ex: **y=1/(x+a)**has a problem at **x=-a**
  - Ex: **y=log(x); dy/dx = 1/x** has a problem at x=0
- **Examples of non-smooth functions:**
  - **sign, abs, max, min**
- Empirical functions to stitch various regions of operation often lead to non-differentiability.

Voltage definitions in MVS model use non-smooth functions

$$V_{ds}=\mathrm{abs}\left(V_d-V_s\right)$$

$$V_{gs}=\max\left(\mathrm{type}\times\left(V_g-V_s\right),\mathrm{type}\times\left(V_g-V_d\right)\right)$$

$$V_{bs}=\max\left(\mathrm{type}\times\left(V_b-V_s\right),\mathrm{type}\times\left(V_b-V_d\right)\right)$$

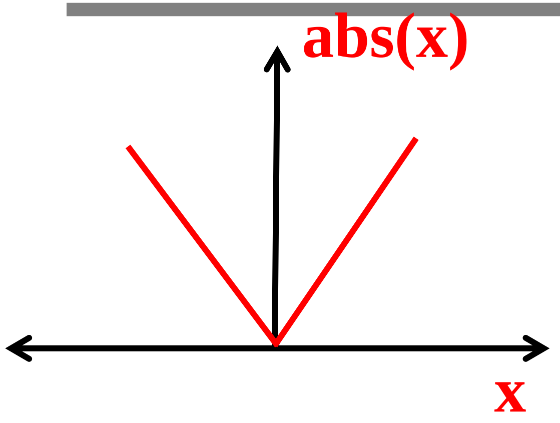
type =

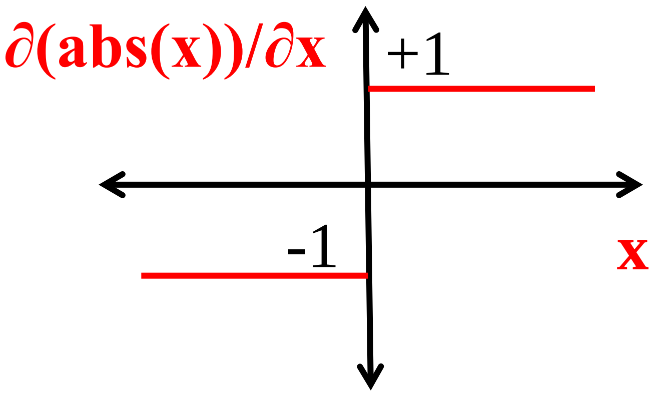
+1 for n-FET

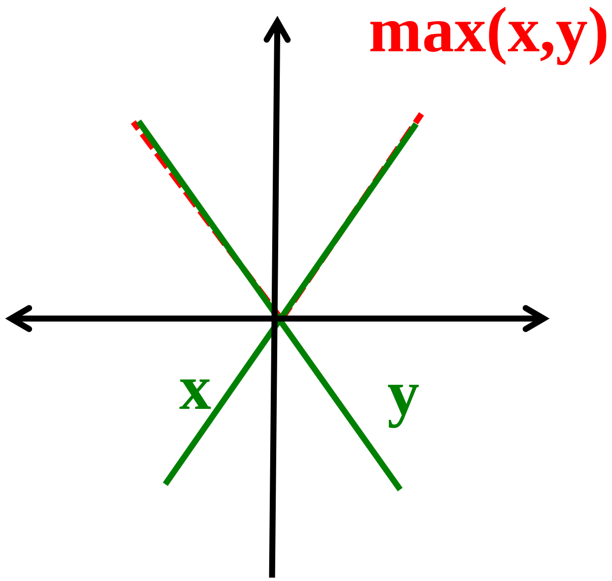
-1 for p-FET

MVS uses source-drain swapping feature forcing the model to be symmetric.

Voltage definitions- abs and max functions





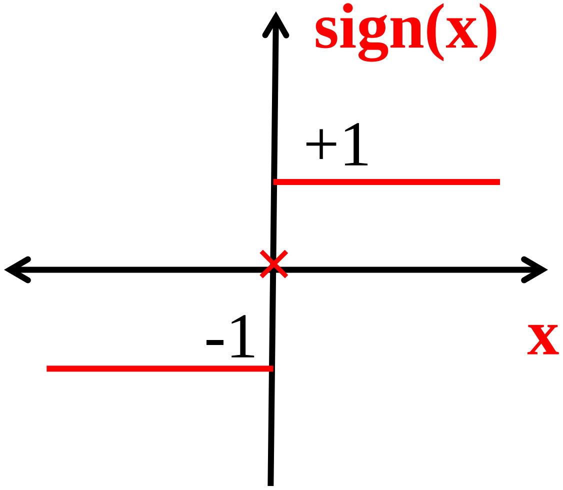


Issue 1: abs(.) & max(.) functions continuity and differentiability ?

Current definition

$$I_d = \text{type} \times \text{dir} \times \left(Q_{x0} v_{x0} F_{sat}\right)$$

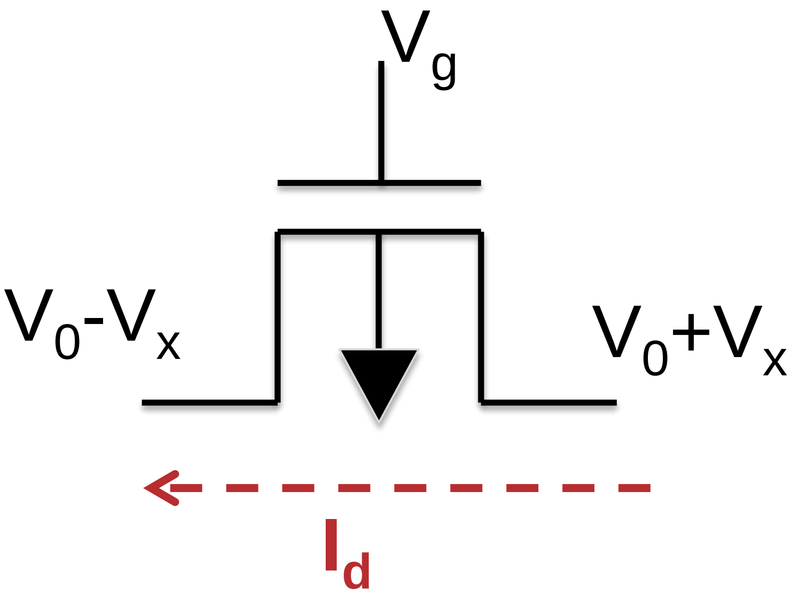
$$\text{dir} = \text{type} \times \text{sign}\left(V_d - V_s\right)$$



Issue 2: sign(.) function continuity and differentiability ?

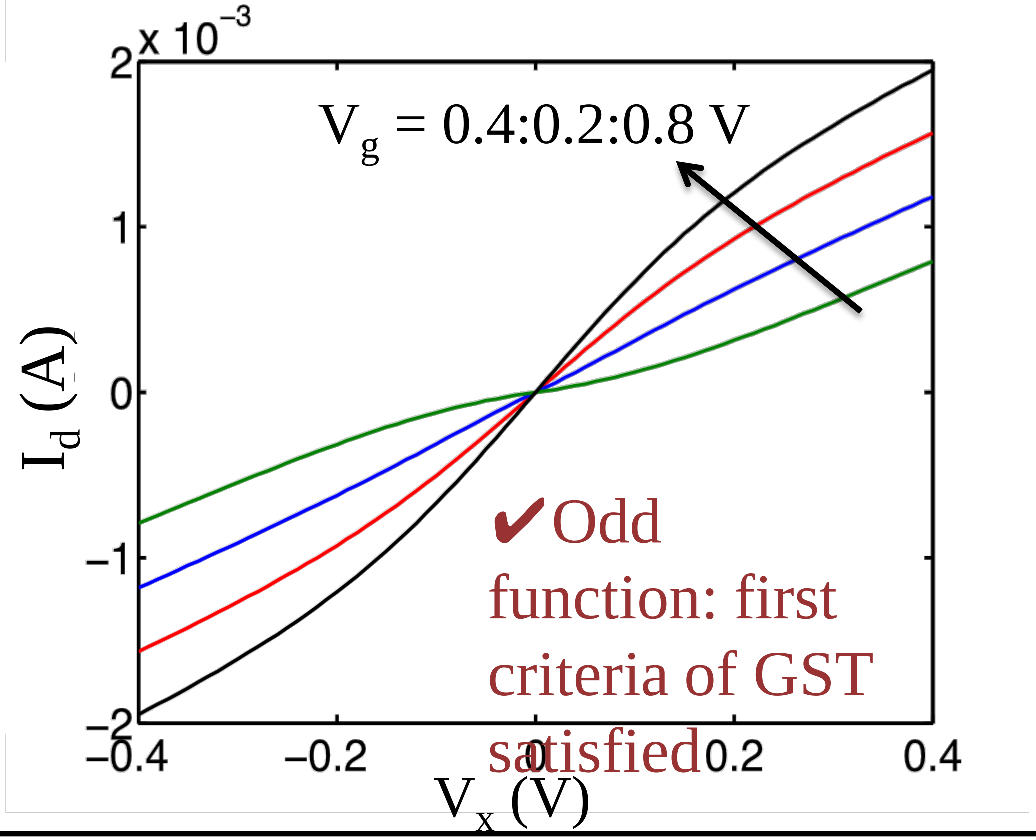
Gummel Symmetry Test (GST)

Test circuit



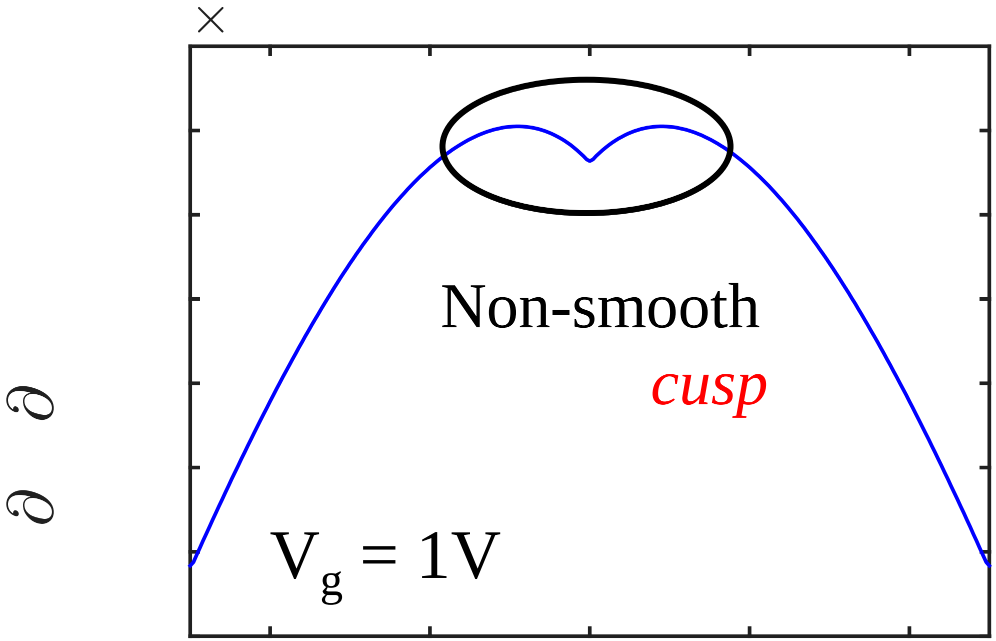
- Benchmark test in compact models and important for RF/analog.
- Odd function **$I_d(V_{ds})=-I_d(-V_{ds})$.**
- **Odd-order derivative of I_d** should be continuous at V_x = 0V.
- **Even-order derivative of I_d** should exist and be equal to 0 at V_x= 0V.

In MVS model, current is an odd function of V_x

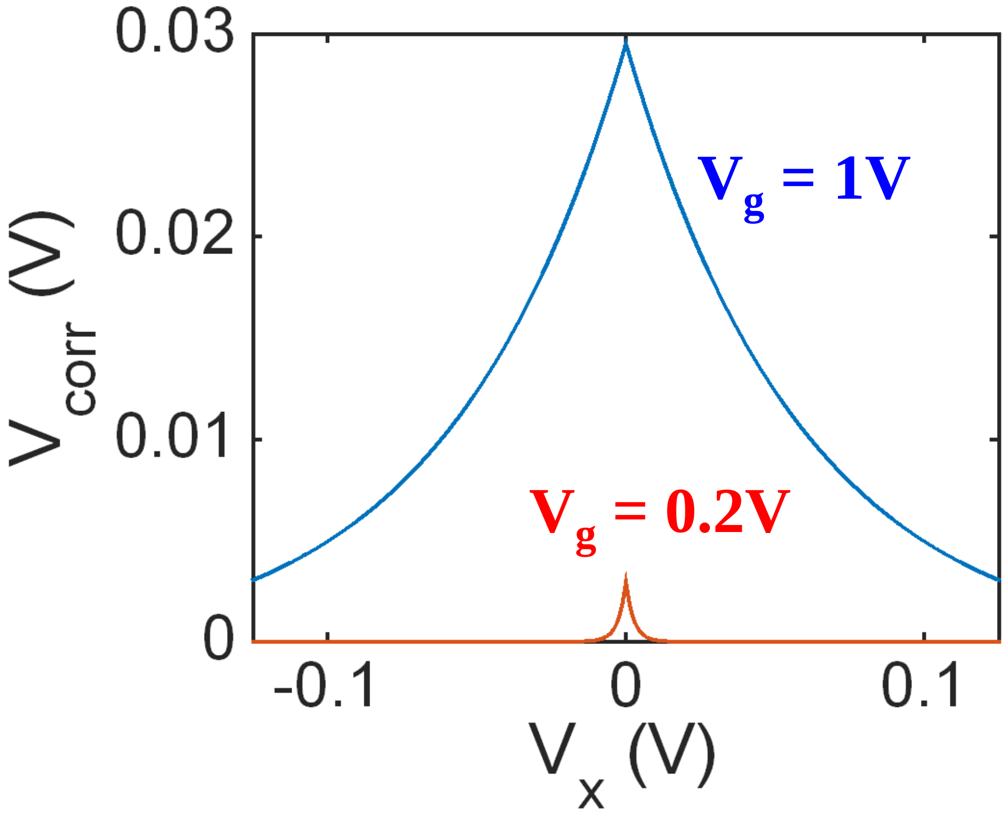


✔Odd function: first criteria of GST satisfied

First derivative of current wrt V_x



Adding a correction term in V_{gs} and V_{bs}



$$V'_{gs} = V_{gs} - \Delta V_{gs} + V_{corr}$$

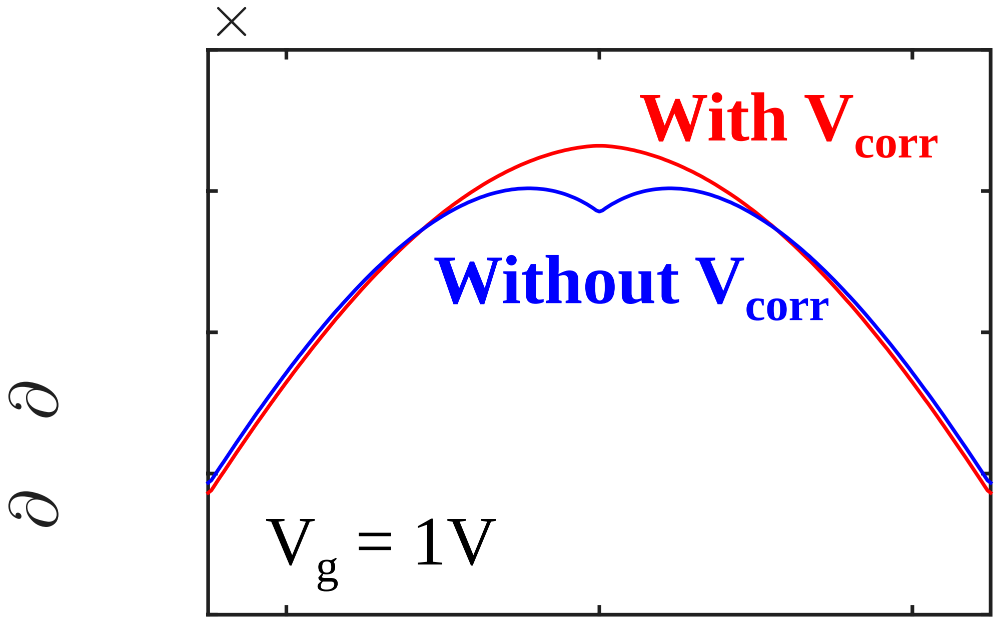
$$V'_{bs} = V_{bs} - \Delta V_{bs} + V_{corr}$$

$$V_{corr} = (1+2\delta)\frac{ab}{2}\exp\left(\frac{-V'_{ds}}{ab}\right)$$

$$ab = 2(1-0.99FF)\phi_t$$

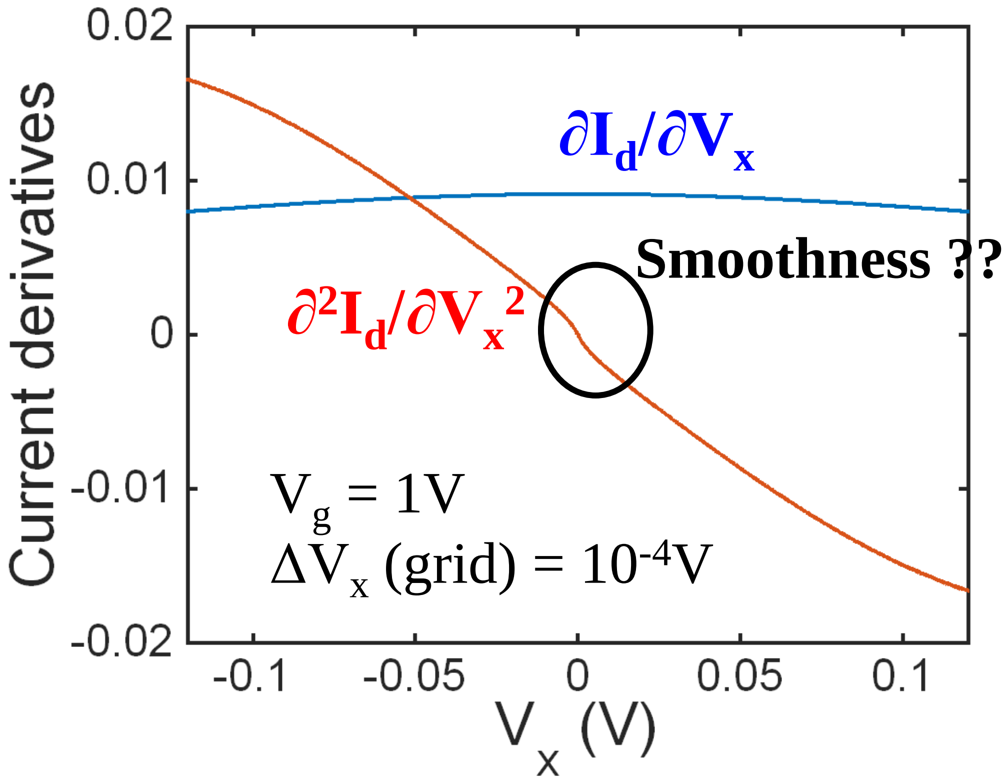
$$FF = \frac{1}{1+\exp\left(\frac{V_{gs}-V_{th}}{1.5\alpha\phi_t}\right)}$$

First derivative of current wrt V_x



shaloo@mit.edu

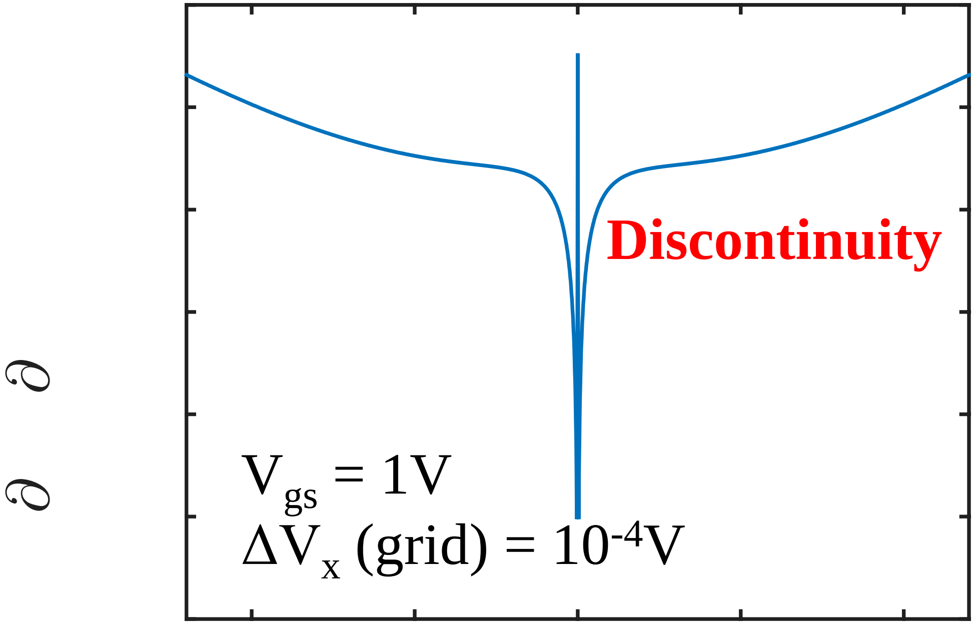
First and second derivatives of current with respect to V_x (with V_{corr})



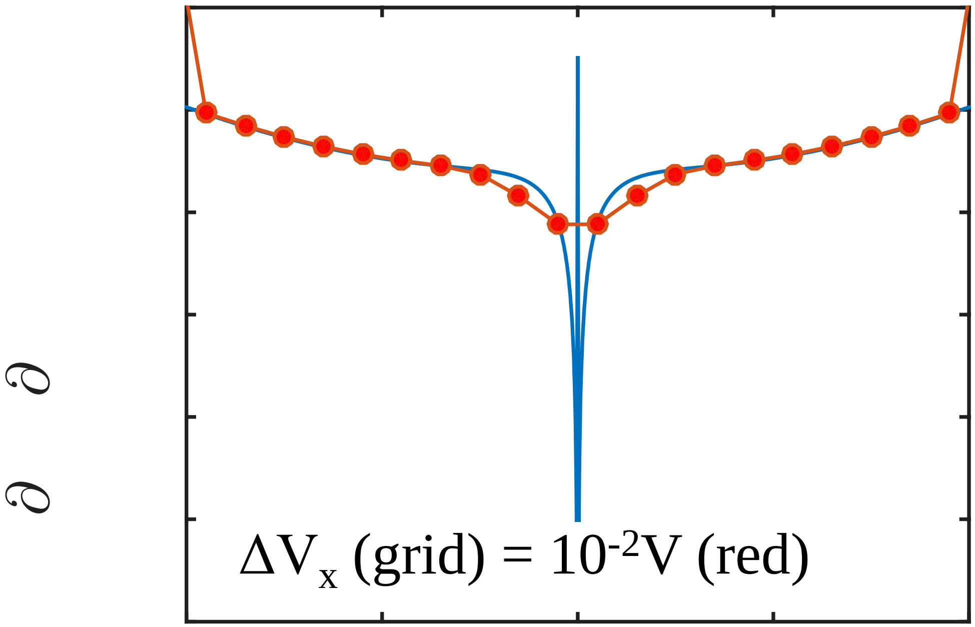
Massachusetts Institute of Technology

shaloo@mit.edu

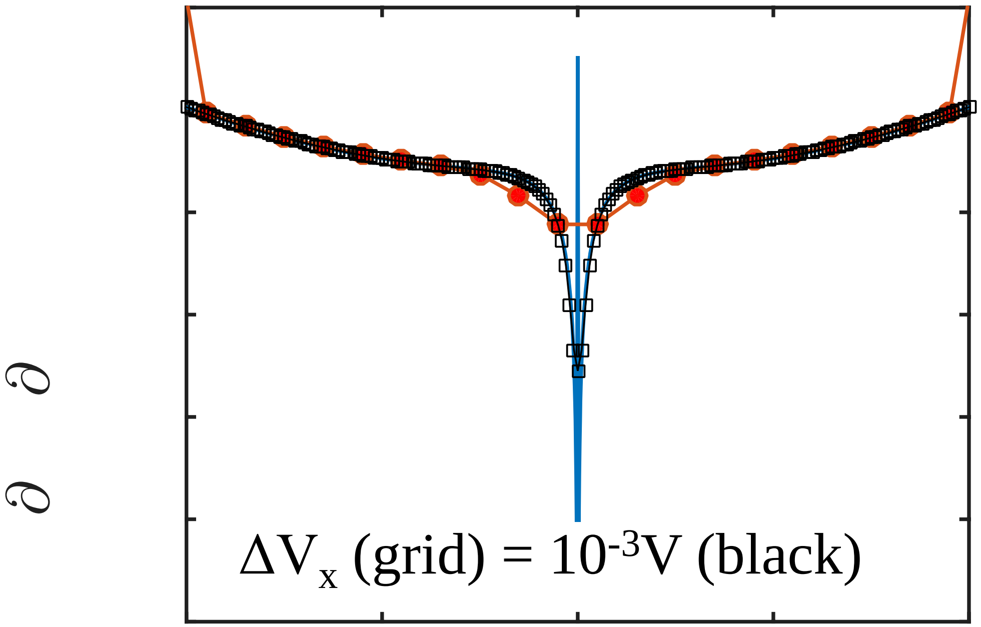
Third derivative of current with respect to V_x



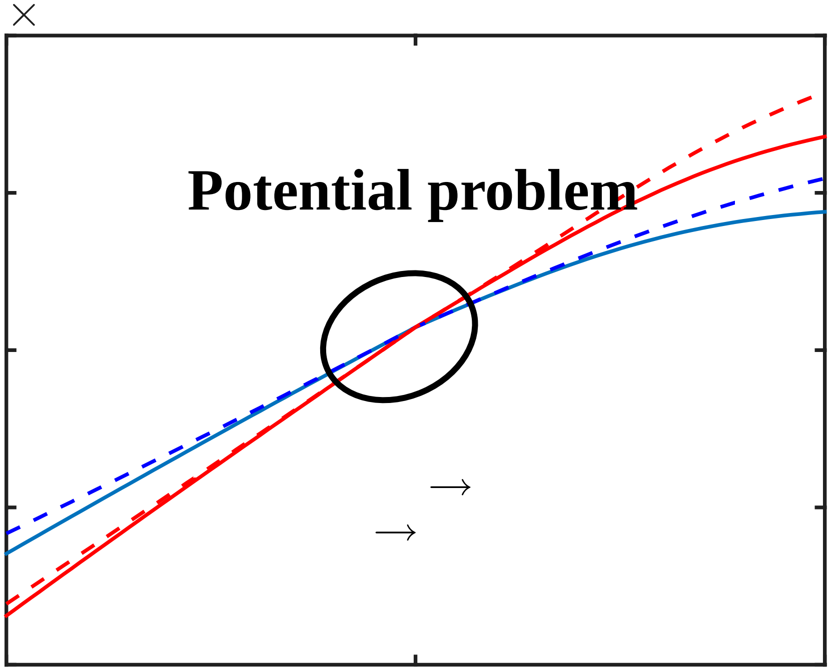
Third derivative of current with respect to V_x



Third derivative of current with respect to V_x

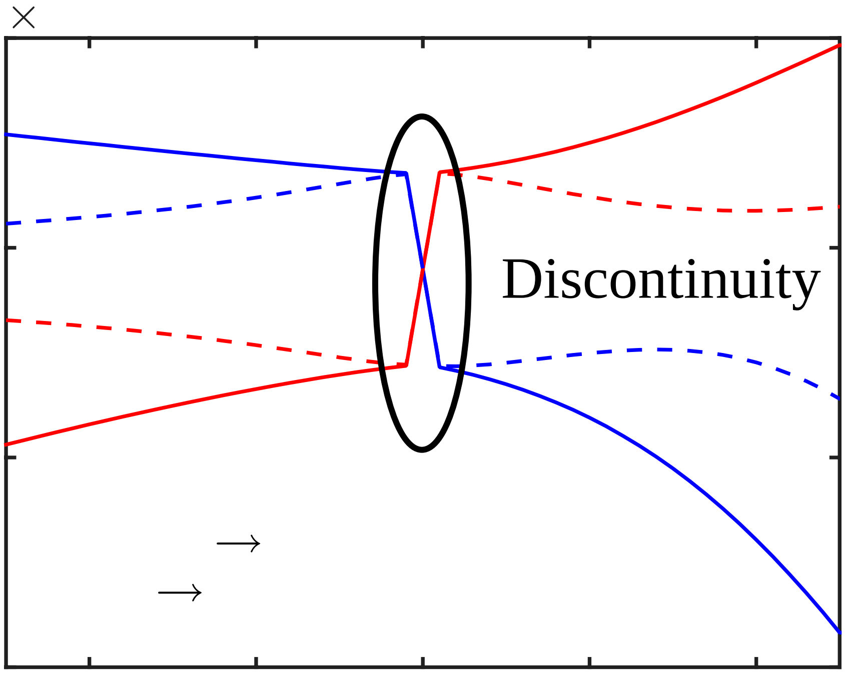


Partitioned charges in MVS model



Models converge for low-V_{ds} as expected.

C_{gs} & C_{gd} versus Vds Above threshold (V_{gs} = 1V)



Summary

- MVS is a source-referenced model.
- To ensure model symmetry for GST, source/drain swapping is implemented.
- Source/drain swapping leads to non-differentiable higher-order derivatives of currents and charges at V_{ds} = 0V.
- Discontinuity in C_{gg} @ V_{ds} = 0V is much less than the discontinuity in C_{gs} and C_{gd}.
- Discontinuities also exist in C_{ds} and C_{dd}.
- Adding body charge worsens the discontinuity in capacitance.

ADDRESSING THE ISSUE OF SMOOTHNESS IN MVS

Smoothing functions

$$smoothabs = @(x)\sqrt{x^2+\varepsilon^2}-\varepsilon$$

$$V_{ds} = smoothabs\left(V_d - V_s\right)$$

$$smooth\max = @(x,y)0.5\left(x+y+smoothabs(x-y)\right)$$

$$V_{gs} = smooth\max(V_g - V_s, V_g - V_d)$$

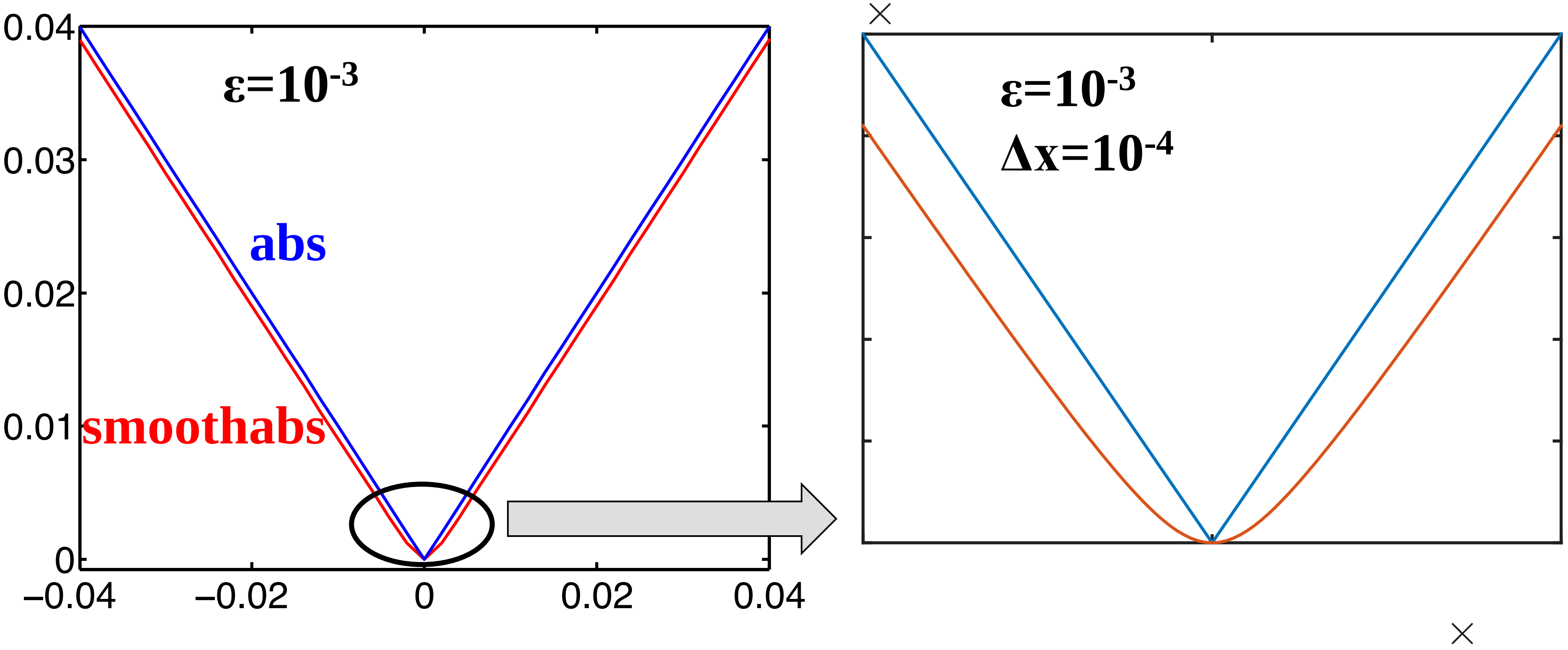
Derivative of smoothabs

$$smoothsign(x) = \frac{x}{\sqrt{x^2+\varepsilon_2^2}}$$

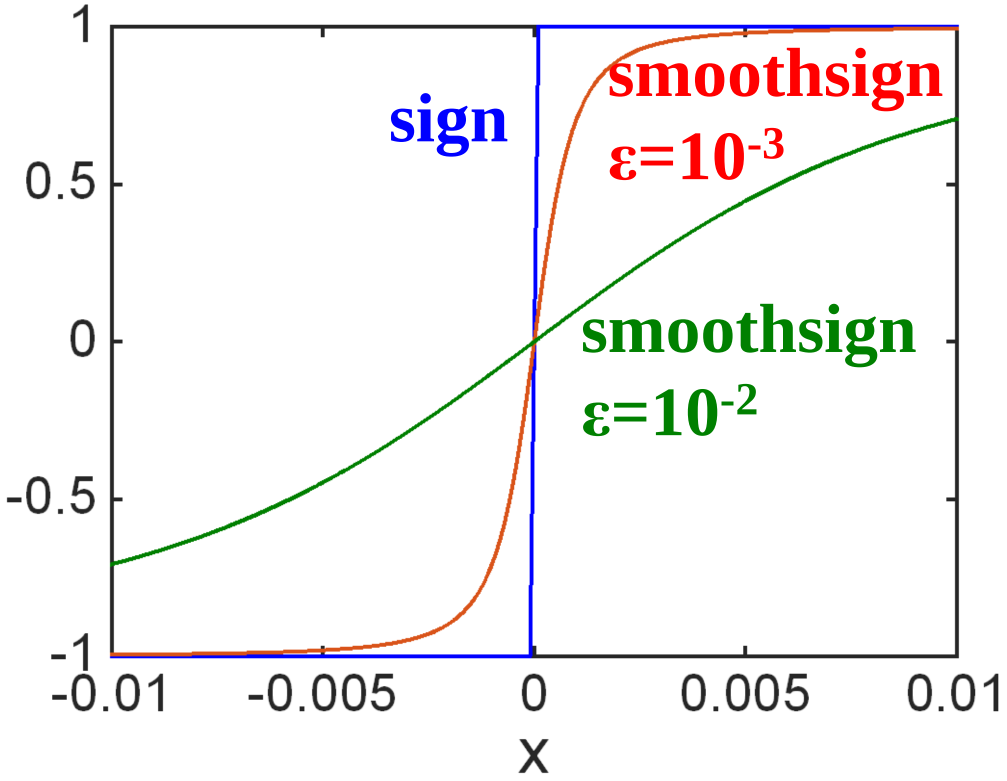
smoothsign function is used in place of the variable ***dir*** in the code.

Use two different values of correction: ε and ε_2

Smoothabs


0.04
0.03
0.02
0.01
0
–0.04
–0.02
0
0.02
0.04
ε=10^-3
abs
smoothabs
ε=10^-3
Δx=10^-4

Smoothsign

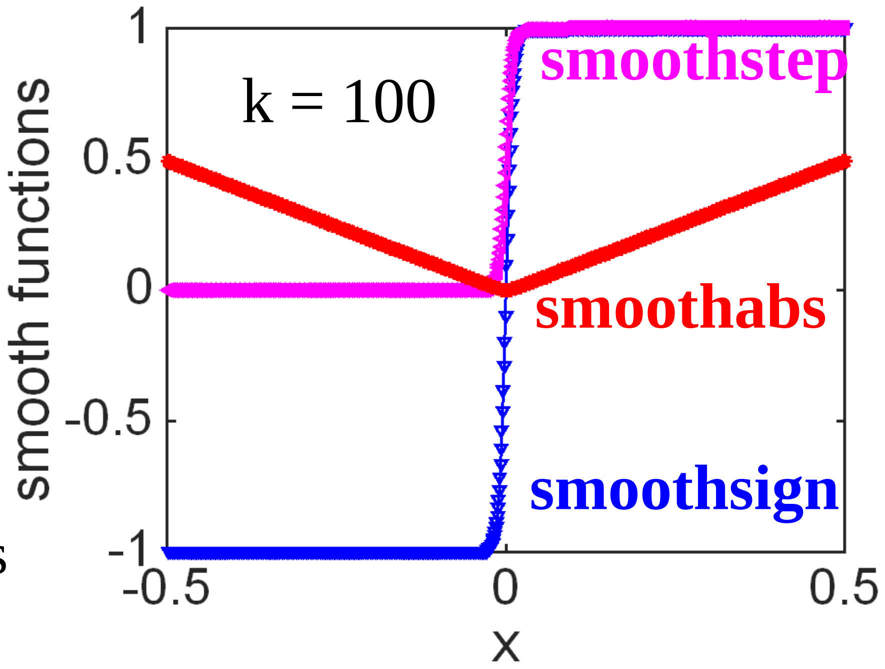

1
0.5
0
-0.5
-1
-0.01
-0.005
0
0.005
0.01
X
sign
smoothsign
ε=10^-3
smoothsign
ε=10^-2

Other possible implementations of smoothing functions

sign(x) → tanh(k*x)

step(x) → 0.5*(1+smoothsign(x))

abs(x) → $2\int_0^x$ smoothstep(y)dy –x



k is the smoothing parameter & governs the width of the transition region.

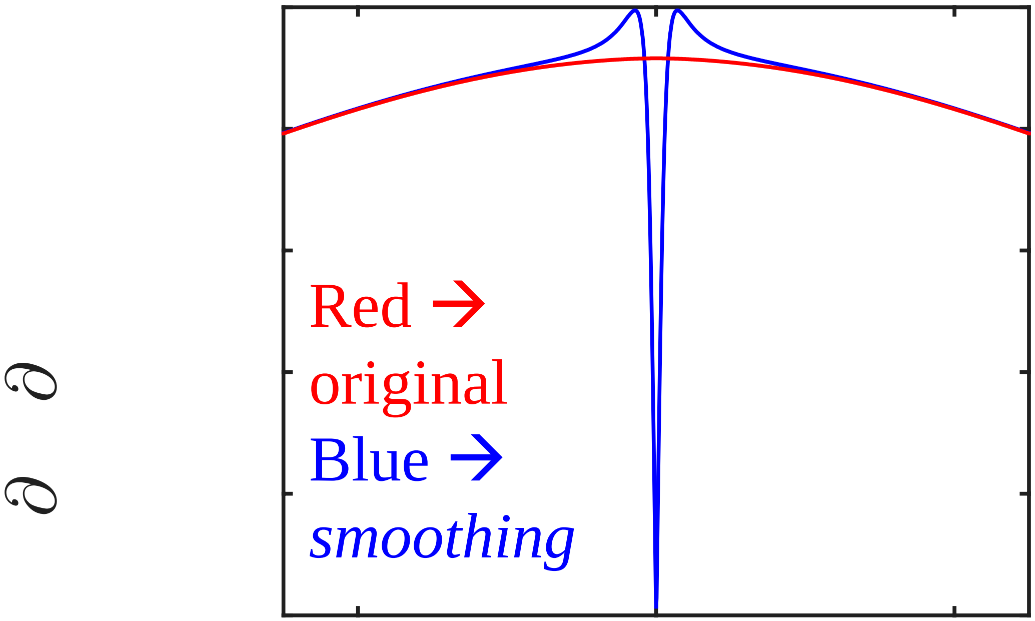
Reference: Prof. Roychowdhury’s lecture notes *https://nanohub.org/resources/21262*

Smoothing

- Several different implementations of ***smoothabs()***, ***smoothsign()*** etc. exist.
- The value of **smoothing parameters** must be carefully chosen for a device as these values **depend on device parameters**.
- The **discretization** in voltage vector is important since derivatives are being computed numerically.
- Finally, the smoothing parameters may also **depend on the terminal voltage V_{gs}** in the transistor.

What problems do you foresee in the MVS transistor model by using these smoothing functions?

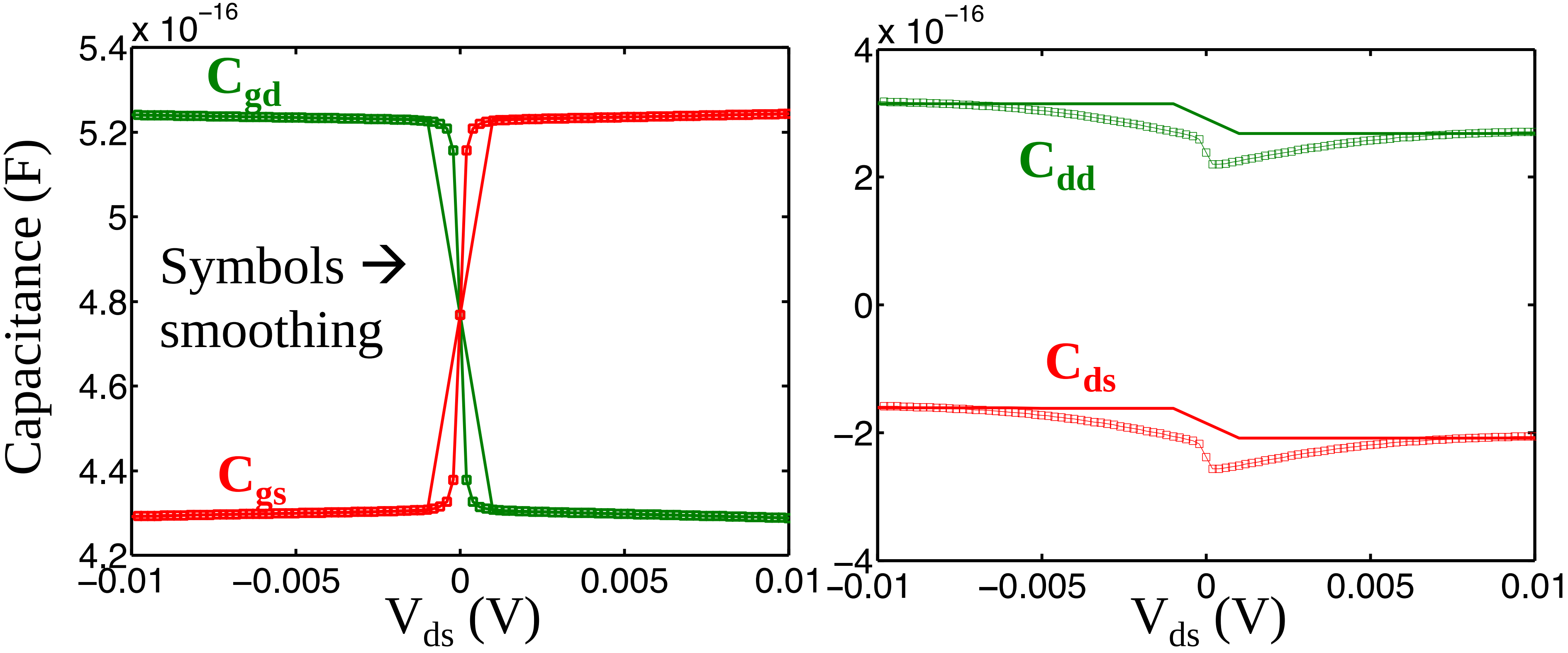
Problem in first derivative of current



Smoothing may not always capture the correct physical picture !

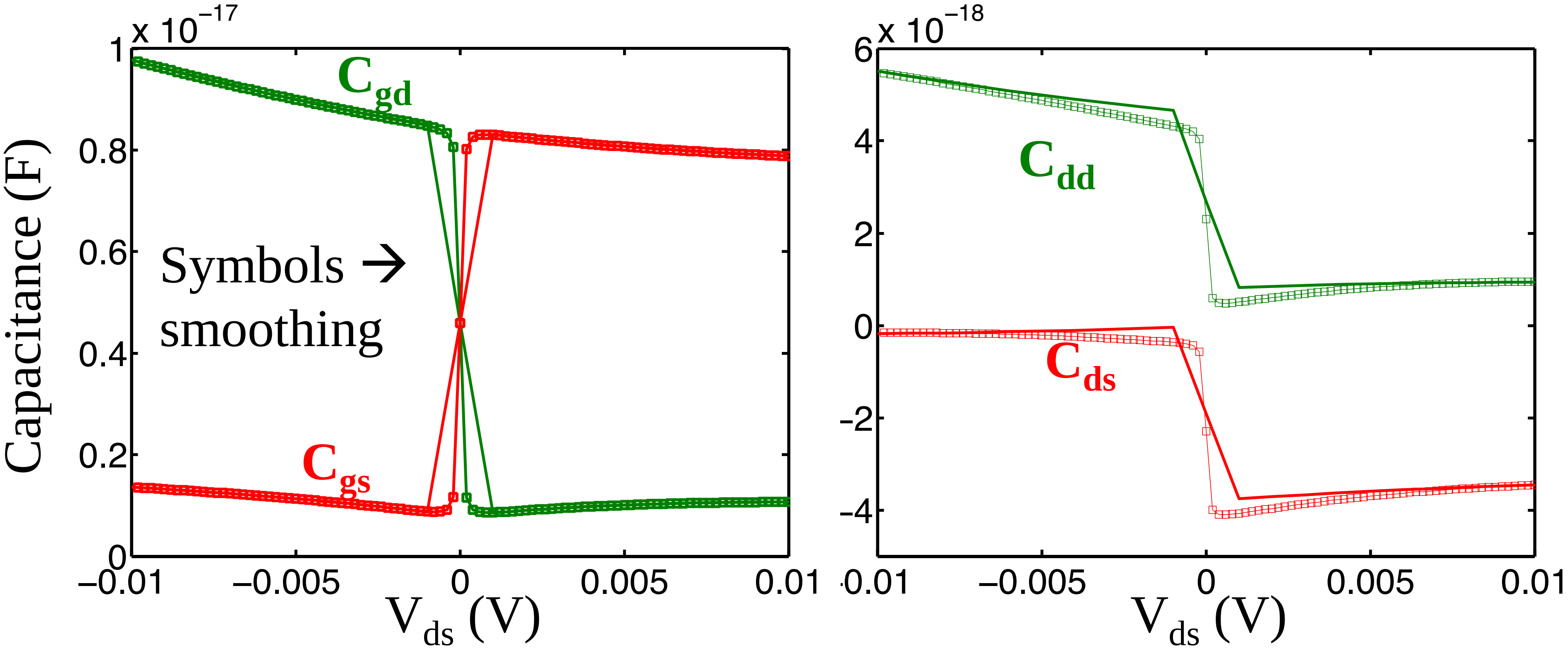
45 nm device, $\varepsilon=10^{-4}$, $\varepsilon_2=10^{-2}$

$\Delta V_{ds} = 2\varepsilon$; $V_{gs} = 1V$



45 nm device, $\varepsilon=10^{-4}$, $\varepsilon_2=10^{-2}$

$\Delta V_{ds} = 2\varepsilon$, $V_{gs} = 0.2V$



Summary: smoothing capacitances

- With smoothing the abs, sign, and max functions only for charge calculations, capacitances can be smoothened.
- Smooth capacitances achieved for both below and above threshold voltages.
- Even with finite body charge, the capacitances remain smooth.
- As a next step, ***vecvalder*** will be tried.

OVERFLOW PROBLEMS

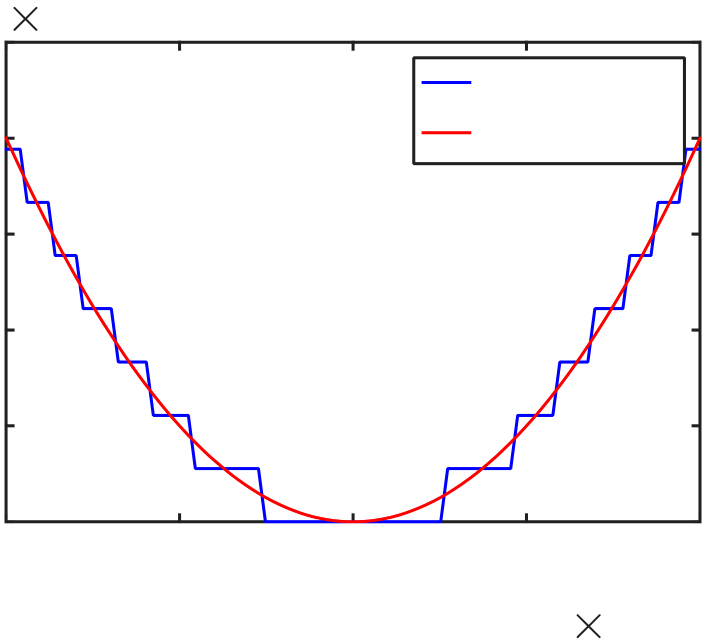
Overflow problems

- Watch out for fast growing functions like **exponentials**
  - **trap IEEE FP errors early on; design your model to avoid them**
  - **Note: $e^{709} = 10^{308}$ is the largest double precision number**
  - **Be careful when subtracting two large numbers:**
    - **Try in MATLAB: (exp(x)+x)-exp(x) for x = 40**

NEEDS

Know the right way to calculate stuff- 1/2

- Use **$2*\sin^2(x/2)$** instead of **(1-cos(x))** when x is tiny
  - 1-cos(x) catastrophically loses precision for tiny x.



Know the right way to calculate stuff- 2/2

Function	Better implementation
$\sqrt{1+x}-1$	$\dfrac{x}{\sqrt{1+x}+1}$
$\left(1+x\right)^{2}-1$	$x(2+x)$
$\mathbf{ln}(1+x)$	$2\times a\,\mathbf{tanh}\left(\dfrac{x}{x+2}\right)$
$\mathbf{exp}(x)-1$	$\mathbf{tanh}\left(x/2\right)\left(\mathbf{exp}(x)+1\right)$

Plot both lhs and rhs functions for x between (-1e-15 to 1e-15) and notice the difference !!

PART II
PERFORMANCE-INHIBITING CONSTRUCTS IN VERILOG-A

Avoid

1. Unused variables
2. Floating nodes
3. Use of events → initial_step, final_step,cross
4. Use of block-level modeling features → transition, slew, last_crossing, absdelay
5. Use of loops
6. log() versus ln() [Verilog-A uses log() as base-10 logarithm unlike MATLAB.]

Also avoid

7. Superfluous assignments
8. Memory states
9. Discontinuity → ***if*** *clauses; functions such as* ***abs***
10. Numerical hazards → *division by zero, exponential growth, domain & overflow problems*
11. Constructs that are inhibit performance

Example of 1-6 are given in the talk: https://nanohub.org/resources/18621

Avoid superfluous assignments

```
(1)  x = V(a,b)/R;
(2)  if (type == 1)
(3)        x = V(a,b)/R1;
(4)  else
(5)        x = V(b,a)/R2;
```

→ Superfluous

Diagnostic message from compiler:

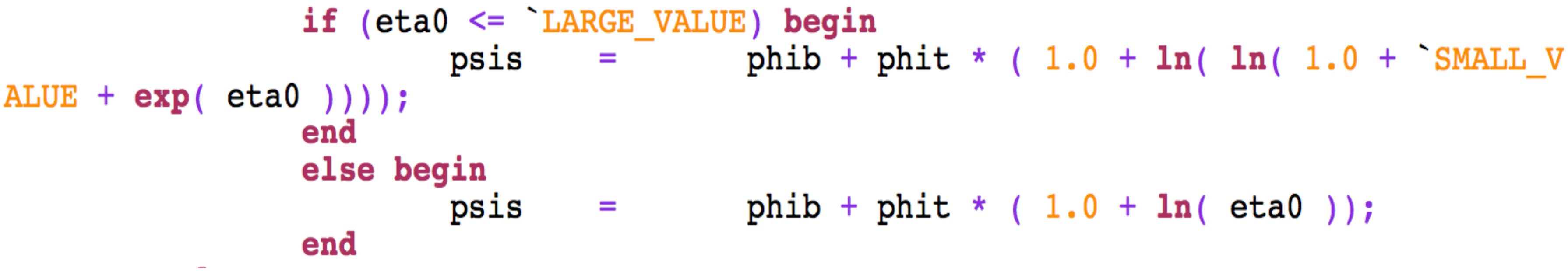
```
Warning: Assignment to ‘x’ may be superfluous.
      [ filename.va, line 1 ]
```

Memory states

1. Also known as ***hidden states***.
2. Variables are initialized to zero on first call to module.
3. Simulator will retain the value of the previous iteration if the variable is not assigned before it is used.
4. Memory states cause ***unexpected behavior***.
5. These states are not typically identified in DC/TRAN simulations.

Declare and initialize variables before use

Avoid memory/hidden states



```
            if (eta0 <= `LARGE_VALUE) begin
                    psis    =       phib + phit * ( 1.0 + ln( ln( 1.0 + `SMALL_V
ALUE + exp( eta0 ))));
            end
            else begin
                    psis    =       phib + phit * ( 1.0 + ln( eta0 ));
            end
```

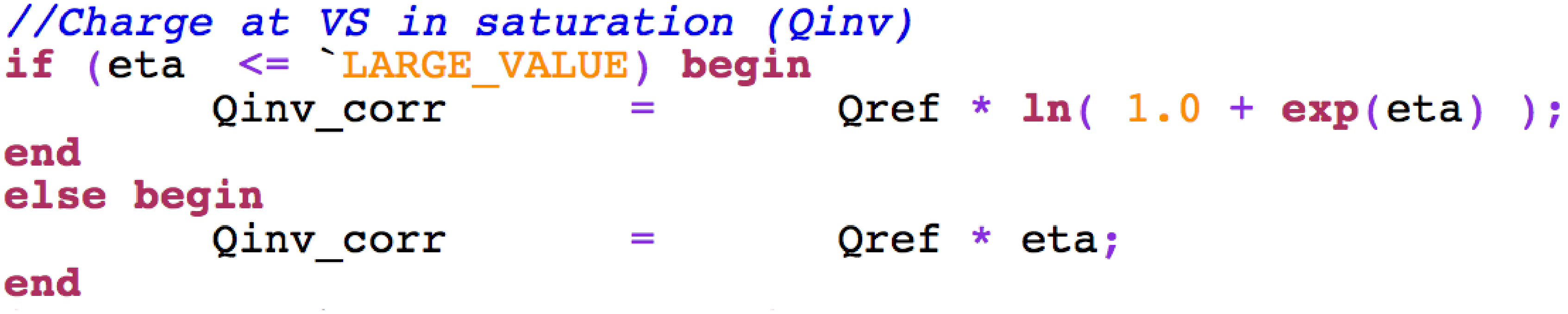
The variable **psis** must always be assigned a value.

Simulation error due to hidden state in MVS 1.0.0 (fixed in 1.0.1)

Discovered through periodic steady state (PSS) analysis

Evaluating $exp()

Explicitly linearize $exp()above a break-point



```
//Charge at VS in saturation (Qinv)
if (eta  <= `LARGE_VALUE) begin
        Qinv_corr       =       Qref * ln( 1.0 + exp(eta) );
end
else begin
        Qinv_corr       =       Qref * eta;
end
```

Recommended practice

Evaluating $ln()

```
psis  =  ( 1.0 + ln( ln( 1.0 +exp( eta0 ))));
```

eta0 → large negative, exp(eta0) = 0 → ln(0) can’t be evaluated

Adding a small correction `SMALL_VALUE fixed the problem

```
psis  =  ( 1.0 + ln( ln( 1.0 +`SMALL_VALUE+ exp( eta0 ))));
```

Defined as 1e-10

Avoid extra state variables → use current contributions

- Try to formulate contributions as currents
  - **I(a,b) <+ …**
  - Use existing state variables & no increase in matrix size
- Implement a nonlinear capacitance as
  - **I(a,b) <+ f(V(a,b));**
- But voltage contributions are better for tiny resistances (convergence)
  - **V(a,b) <+ I(a,b) * Rab;**

Avoid extra state variables → use voltage contributions ONLY when needed

- Truly voltage controlled elements must be implemented with voltage contributions.
- Inductances in Verilog-A will add an additional state variable
  - V(a,b) <+ L * ddt(I(a,b)); ✓
  - I(a,b) <+ idt(V(a,b))/L;

The ddt() form translates to

$-X_a + X_b + ddt(L*I_{ab}) = 0$

Recall: MNA

Avoid extra state variables → branches from conditionals

- When variables that depend on **ddt()** are used in conditionals, the compiler must create extra branch equations
  - Do not place the function **ddt()** within conditionals
  - Place the arguments of **ddt()** within conditionals

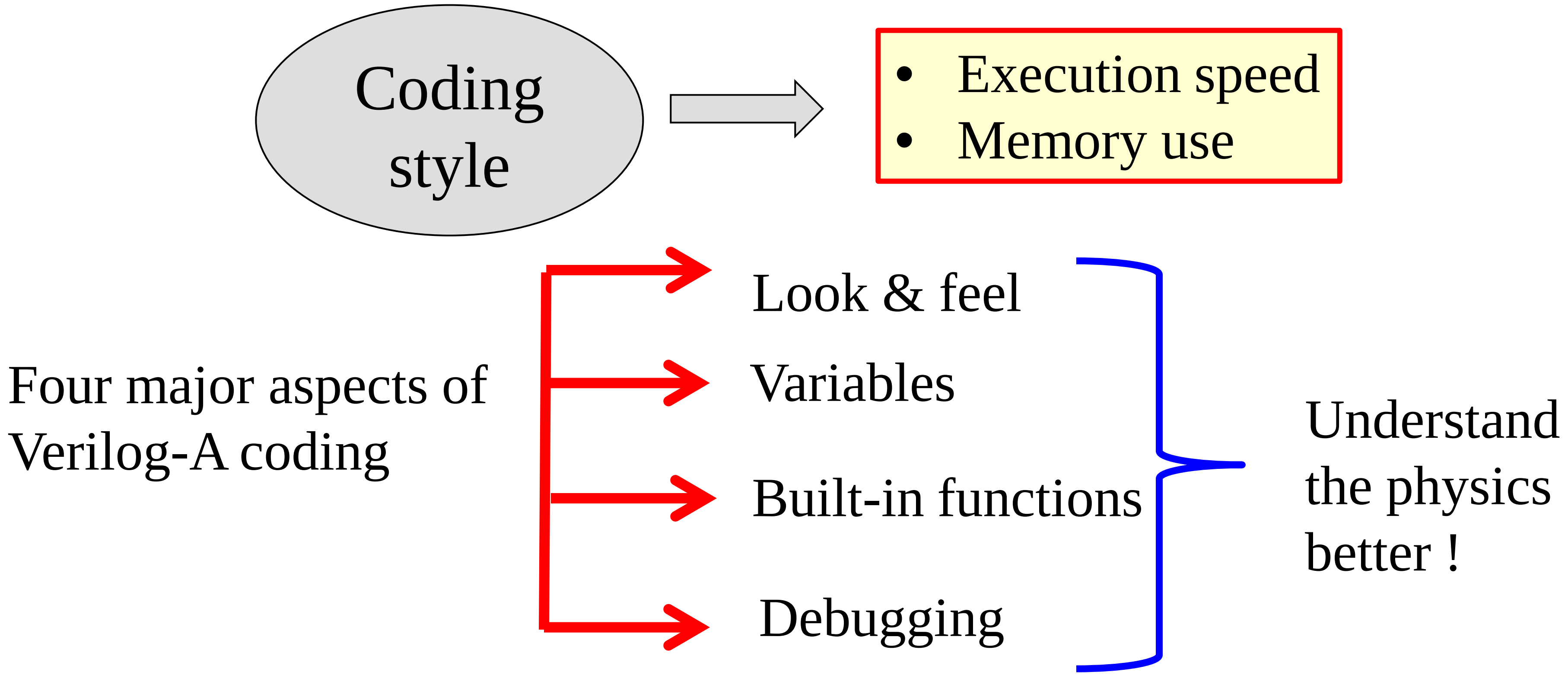
Avoid extra state variables → branches from conditionals

```
Qbd_ddt = ddt(Qbd);
Qbs_ddt = ddt(Qbs);

if (Mode == 1) begin
        t0 = TYPE*Ibd + Qbd_ddt;
        t1 = TYPE*Ibs + Qbs_ddt;
end
else begin
        t1 = TYPE*Ibd + Qbd_ddt;
        t0 = TYPE*Ibs + Qbs_ddt;
end
I(b,di) <+ t0;
I(b,si) <+ t1;
```

```
if (Mode == 1) begin
        t0 = TYPE*Ibd;                 ✔
        arg0 = Qbd;
        t1 = TYPE*Ibs;
        arg1 = Qbs;
end
else begin
        t1 = TYPE*Ibd;
        arg1 = Qbd;
        t0 = TYPE*Ibs;
        arg0 = Qbs;
end
I(b,di) <+ t0 + ddt(arg0);
I(b,si) <+ t1 + ddt(arg1);
```

Summary



Coding style

- Execution speed
- Memory use

Four major aspects of Verilog-A coding

Look & feel

Variables

Built-in functions

Debugging

Understand the physics better !

References

1. http://www.mos-ak.org/baltimore/talks/11_Mierzwinski_MOS-AK_Baltimore.pdf

2. www.**mos-ak**.org/sanfrancisco/.../01_McAndrew_**MOS-AK**_SF08.ppt

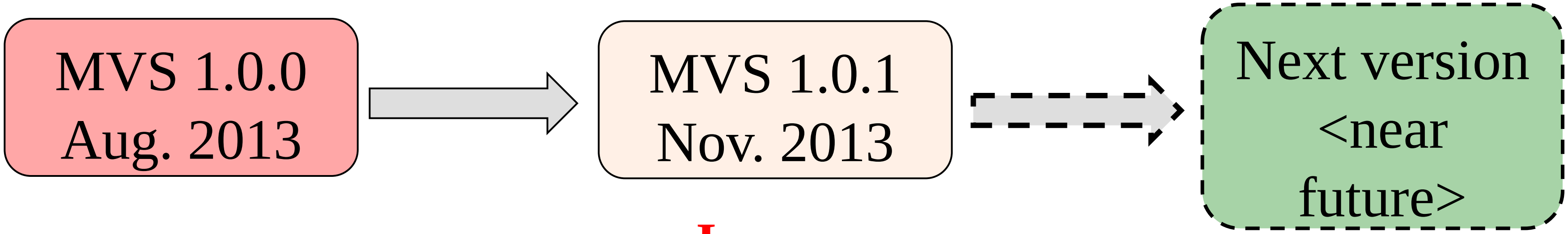
3. www.mos-ak.org/montreux/papers/06_Coram_MOS-AK06.ppt

4. G. Coram, “How to (and how not not) write a compact model in Verilog-A”, BMAS 2004.

5. Tianshi Wang; Jaijeet Roychowdhury (2013), "Guidelines for Writing NEEDS-certified Verilog-A Compact Models," https://nanohub.org/resources/18621

6. G. Coram, “Verilog-A present status and guidelines,” https://nanohub.org/resources/18557

Evolution of MVS



MVS 1.0.0
Aug. 2013

MVS 1.0.1
Nov. 2013

Next version
<near
future>

Issues:

- Unused variables
- Hidden states
- Parameter range
- Indentation

Issues:

- **Capacitance discontinuity**
- Better ways needed to fix some other numerical issues in VA

- Can we address the non-differentiability of higher-order current derivatives?