Dealing with Common Numerical Issues in Compact Models

Excerpts from Numerical Simulation and Modeling
EECS 219A Fall 2013

J. Roychowdhury
EECS, U.C. Berkeley

NEEDS
needs.nanoHUB.org

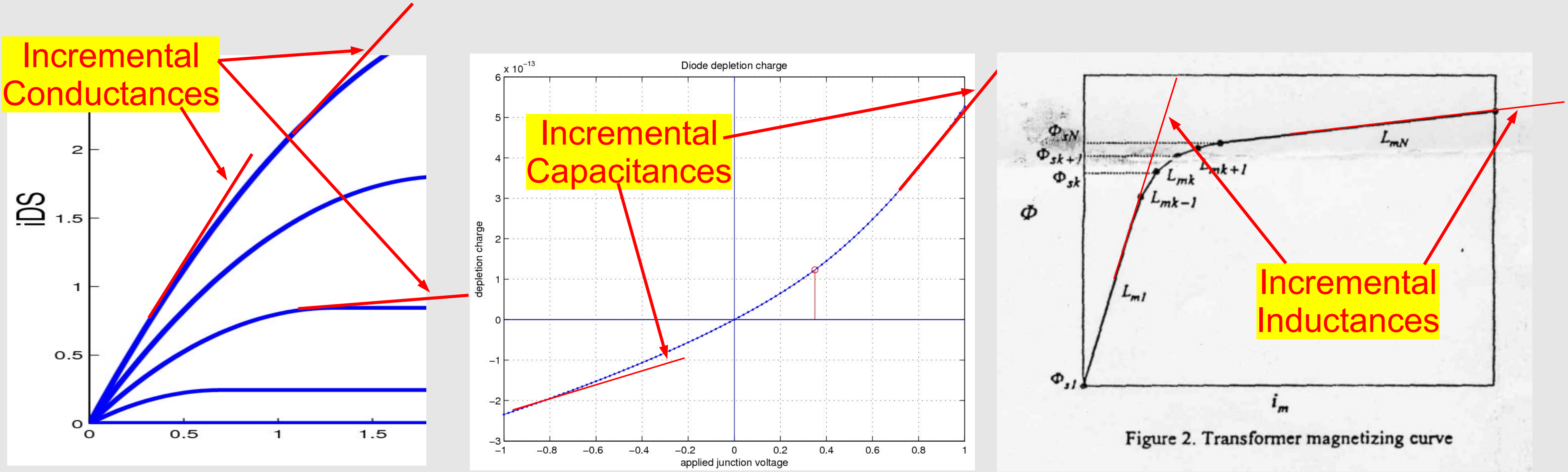
introduction

This presentation is a few excepts presented in a lecture on Electronic Device Models, which is part of a course on numerical modeling and simulation.

These excerpts deal with:

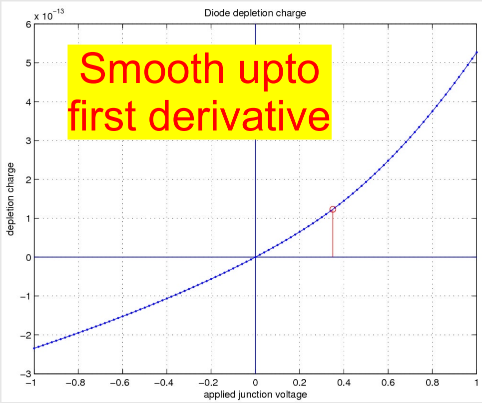
-The need for smoothness in models
-Smoothing discontinuous models
-Domain and overflow problems

Slopes of Nonlinear Elements



- slopes (first derivatives) of model equations important in simulation
  - crucial in DC/transient/AC etc. algorithms
  - circuit design insights centered around slopes
    - ➔ small-signal resistance/capacitance/inductance
  - important to capture accurately during modelling

Model "Smoothness"



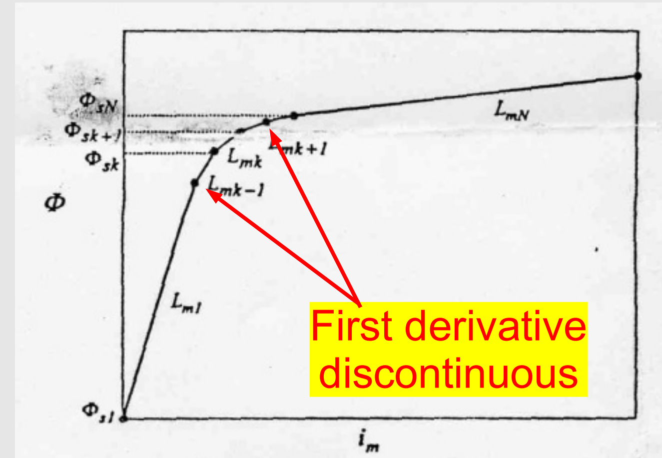


Figure 2. Transformer magnetizing curve

- **"Smoothness" is key**
  - physical systems typically smooth
    - at fine enough resolution
  - important for model/system "robustness"
    - expectation: small tweaks shouldn't lead to drastic changes
    - (but can happen even in perfectly smooth systems)
  - important for numerical methods to work properly
    - Newton-Raphson method, transient solution, etc.
- **Causes of non-smoothness in models**
  - idealization (look out for if conditions)
  - lack of awareness of importance of smoothness
- **Examples to ponder:** diode; MOS (Schichman Hodges)

Schichman-Hodges MOS Model

- Two-port (3-terminal) model
  - expresses I_G, I_D in terms of V_{GS}, V_{DS}
  - **not one-piece: if/then statements**
  - (nonlinear)

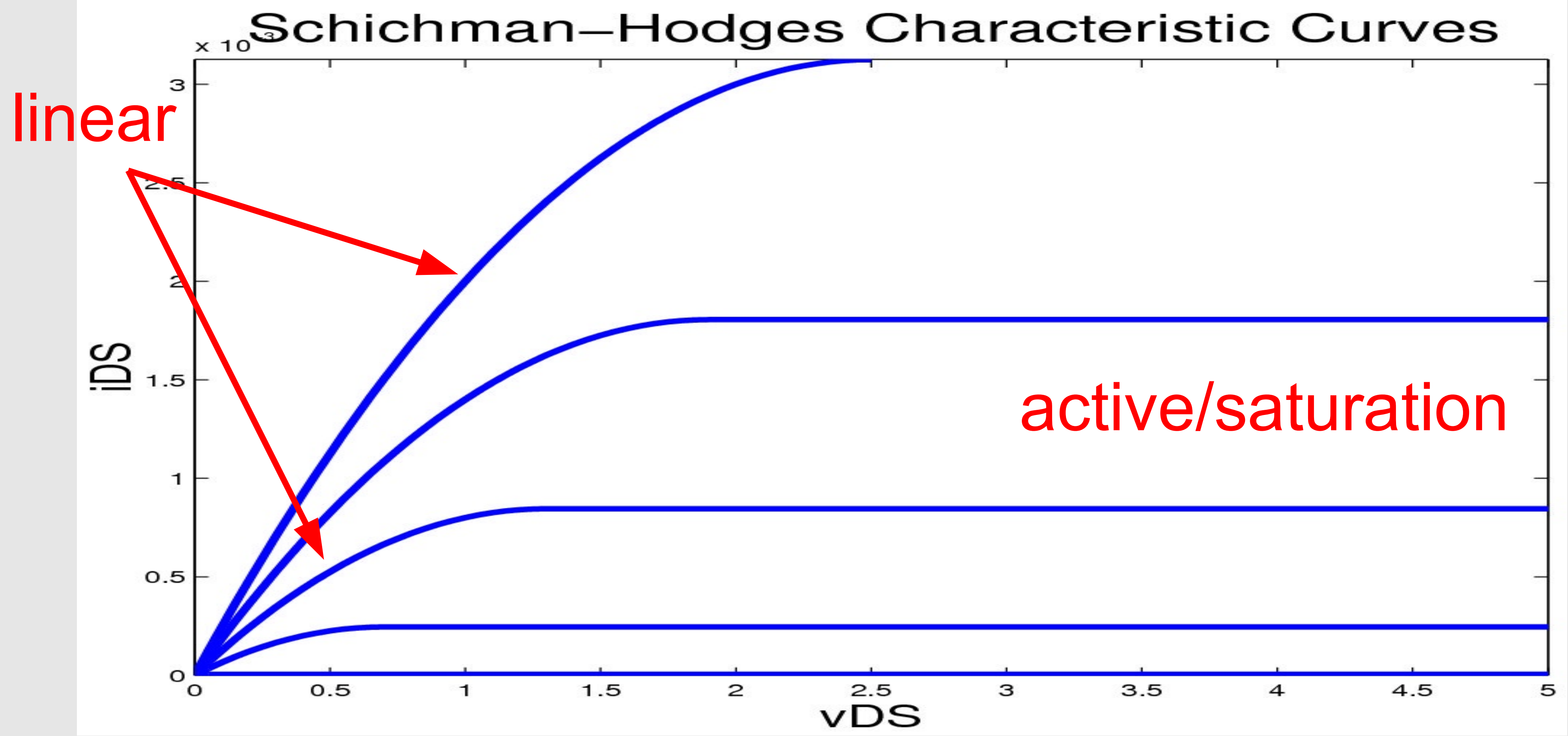
$$I_D = f_{N+}(V_{GS}, V_{DS}) = \begin{cases} \beta \left[(V_{GS} - V_T) - \frac{V_{DS}}{2} \right] V_{DS}, & \text{if } V_{DS} < V_{GS} - V_T \\ \frac{\beta}{2} (V_{GS} - V_T)^2, & \text{if } V_{DS} \geq V_{GS} - V_T \\ 0 & \text{if } V_{GS} < V_T \end{cases}$$

$$I_G = 0$$

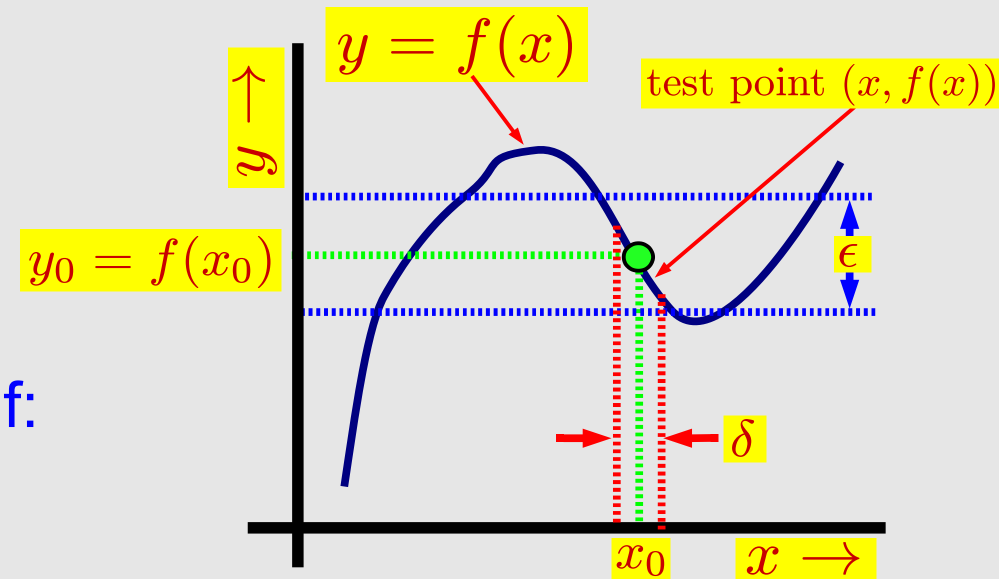
- above is the basic model, only valid for $V_{DS} \geq 0$
- full model must incorporate:
  - source/drain interchange (i.e., when $V_{DS} < 0$)
  - P/N type modelling

S-H MOSFET Characteristic Curves

- Plot drain current I_D vs V_{DS}
  - using V_{GS} as a parameter to get a family of curves



Fundamentals: Continuity, Differentiability



- Function f(x) continuous at x_0 if:
  - [first find $y_0 = f(x_0)$]
    - given *any* ε > 0
    - can *always* find δ > 0 such that:
      - $|f(x) - f(x_0)| < \varepsilon$ *for all x* satisfying $|x - x_0| < \delta$
- **Function f(x) continuous if it is continuous at all x_0**
- **Function f(x) differentiable if:**
  - f'(x) exists at all x, and is continuous
  - (similarly for higher derivatives)

Nonlinear Inductor Example

- Nonlinear Transformer Iron Core

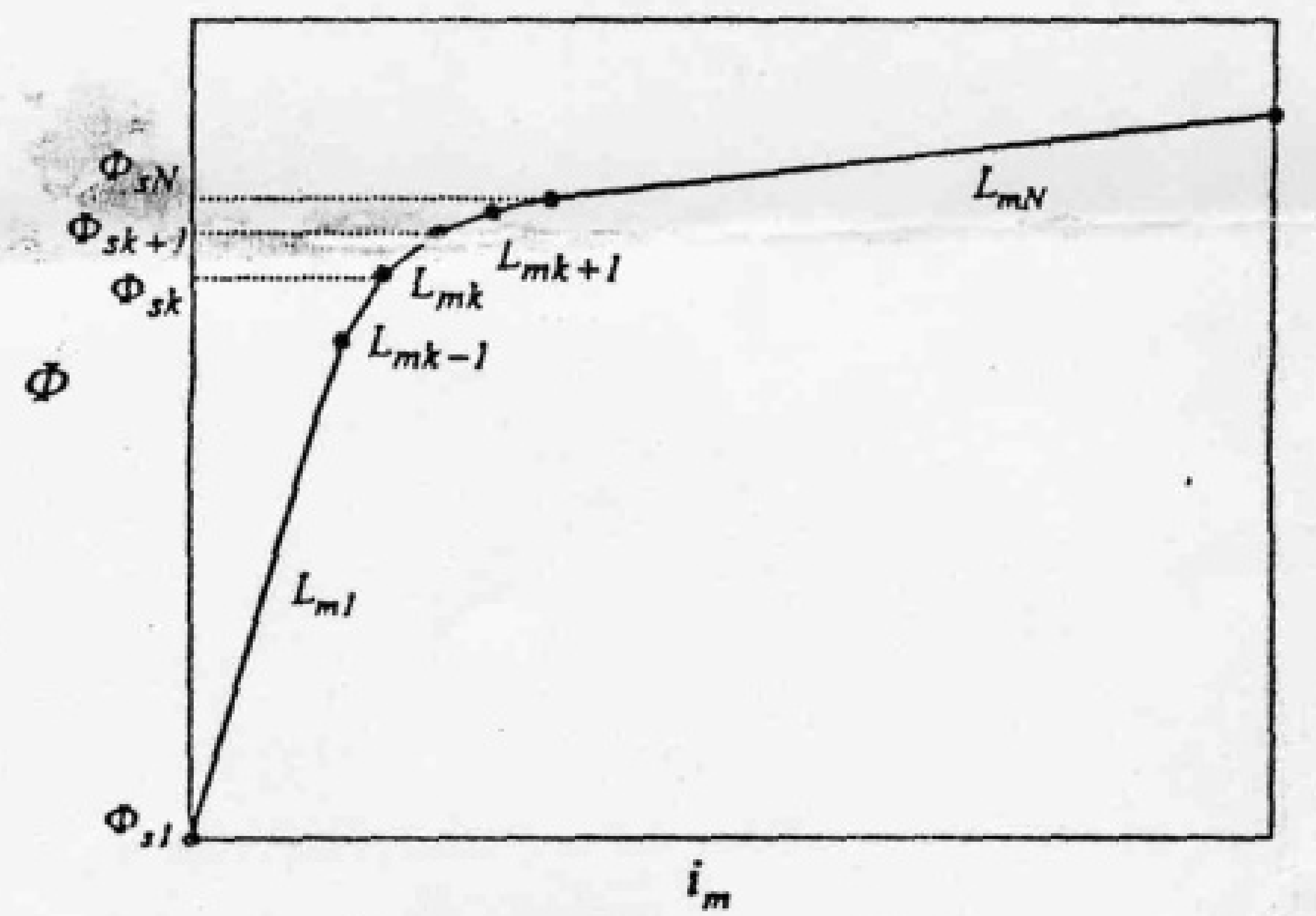
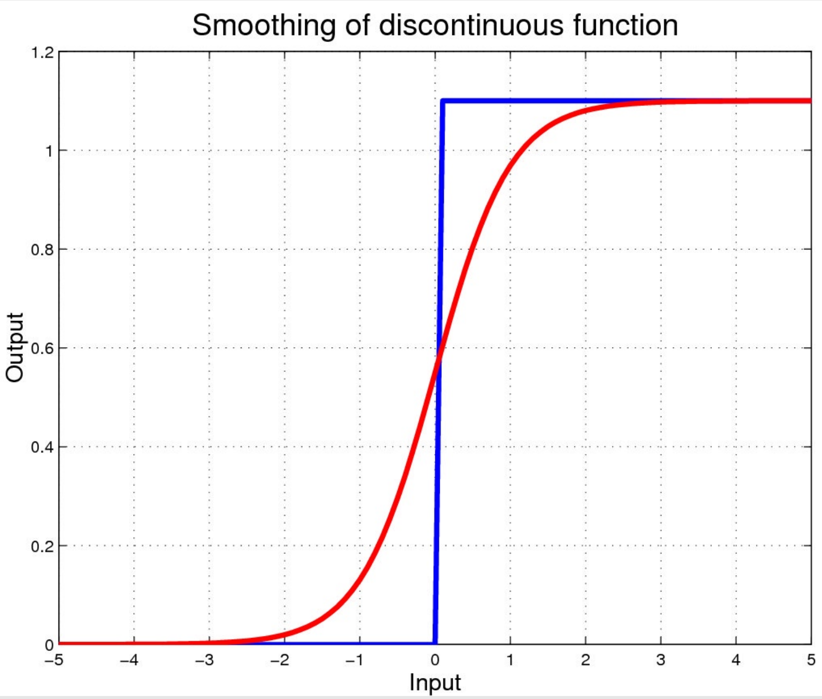


Figure 2. Transformer magnetizing curve

Smoothing Discontinuous Models



- Useful if given “bad” equations for device
  - ➔ or if making a model from measurement data
- Basic idea: replace jumps by smooth functions
  - ➔ e.g., sign_function(x) by tanh(kx)
    - k: smoothing parameter

Common Nonsmooth Primitives

- Common primitives w/ smoothness issues
  - sgn(x), step(x)
  - abs(x)
  - clip(x)
  - if-then-else conditions
    - if (x > a) {y=f1(x)} else {y=f2(x)};
- A recipe for smoothing problematic functions
  - sgn(x): replace by smoothsign(x,k)=tanh(k*x)
  - step(x) = 0.5*(1+sgn(x));
  - clip(x) = $\int_0^x \mathrm{step}(y)\, dy$
  - abs(x) = clip(x) + clip(-x); % or 2*clip(x) - x;
  - replace: if (*cond(x) > 0*) {*y=f1(x)*} else {*y=f2(x)*} by:
    - yesno = smoothstep(cond(x),k);
    - y = f1(x)*yesno + f2(x)*[1-yesno]

Domain, Overflow Problems

- Common primitives that have domain problems
  - sqrt(x): try sqrt(smoothclip(x,k)) instead
  - log(x): log(smoothclip(x,k))
  - 1/x: fixes are very situation specific
- Overflow problems
  - watch out for fast-growing functions (like exponentials)
    - trap IEEE FP errors; **design your model to avoid them *a priori***
  - note: $e^{709} = 10^{308}$ (largest double precision number)
    - diode: exp(40*v), ie, at v=20V
  - remember: $e^x + x$ **exactly equals** e^x for x > ~40
    - catastrophic loss of numerical precision (double: ~16 digits)
      - try (exp(x)+x)-exp(x) in MATLAB
    - do not subtract 2 large numbers
  - useful recipe for exponentials
    - define safeexp(x, threshold)
      - make growth linear (polynomial) after some point
      - choose transition point wisely (situation specific)
  - others: factorial(), gamma functions