SmartSmooth: A linear time convexity preserving smoothing algorithm for numerically convex data with application to VLSI design

Sanghamitra Roy
Charlie Chung-Ping Chen
VLSI-EDA GROUP, UW-Madison

National Taiwan University

Acknowledgements

- We are grateful to
  - Intel
  - TSMC
  - UMC
  - Faraday
  - SpringSoft
  - NSF

Outline

- Motivation
- ConvexFit: basic idea
- Smoothing
- SmartSmooth: problem formulation
- Experimental Results

Motivation

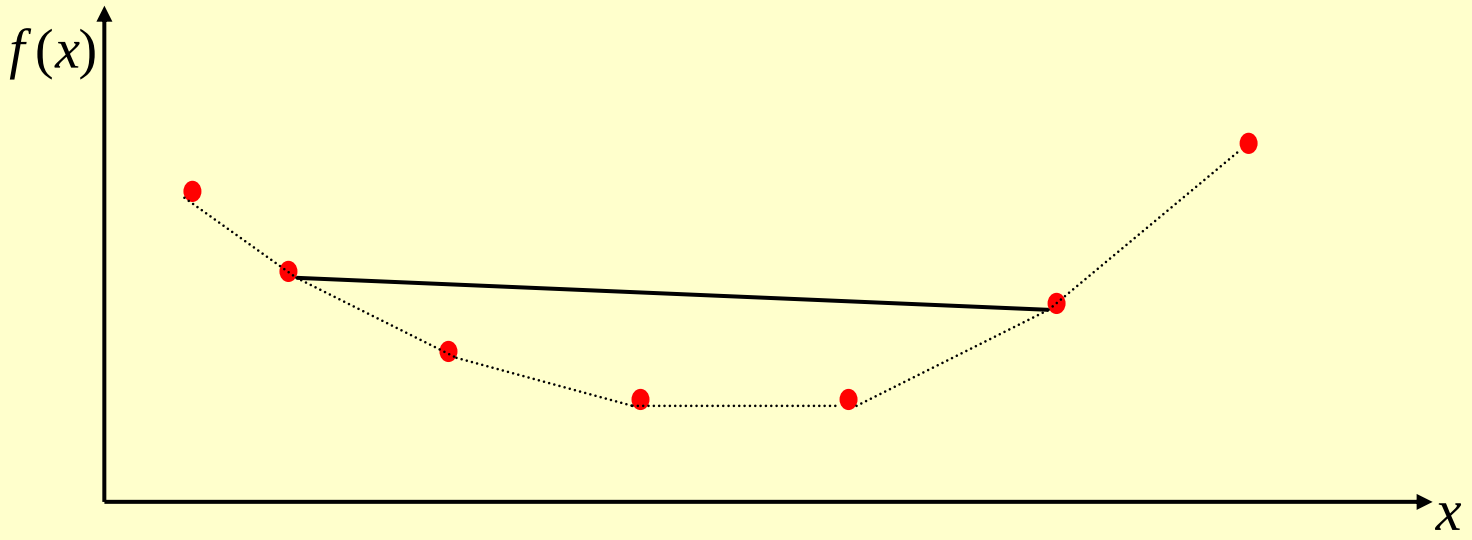
- VLSI design optimization: tabular data
  - example: gate delay, slew, and power
- Convex optimization very popular
  - + can reach **global optimal point** fast
  - - data needed in convex form
  - - data traditionally fitted to analytically convex form: posynomials
  - - huge fitting errors: fitting problem not convex

Our observation

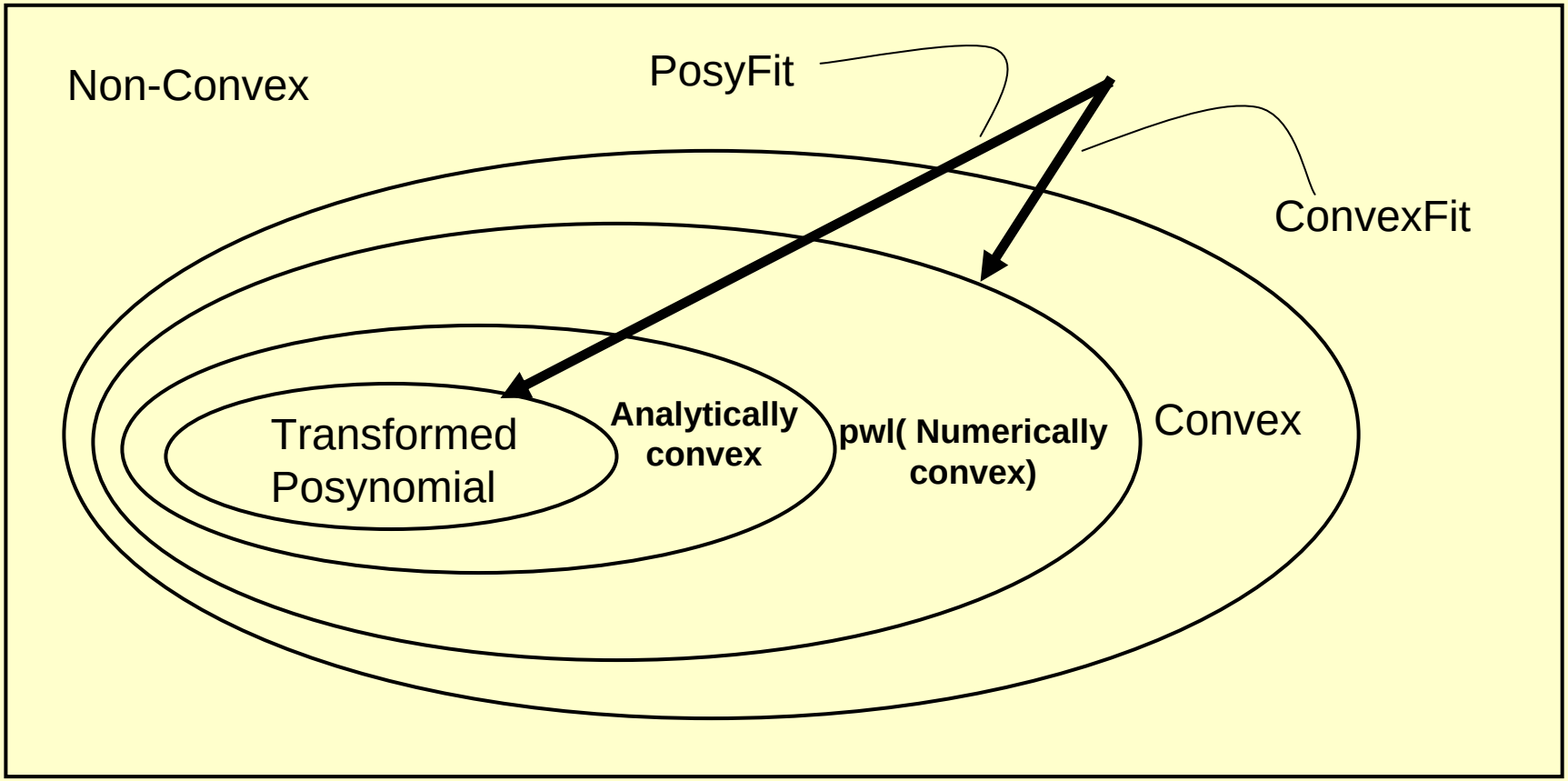
- Delay: theoretically a convex function of gate size, load capacitance: $D(slew, inputcap, loadcap)$
  - Non-convexity introduced by simulation/measurement
- Why not directly make the table convex?
  - Saves huge effort in fitting to analytical form
  - Drastically reduces fitting errors
- Find optimal solution to fitting problem
- Propose: **Numerically convex tables**

Numerically Convex Function

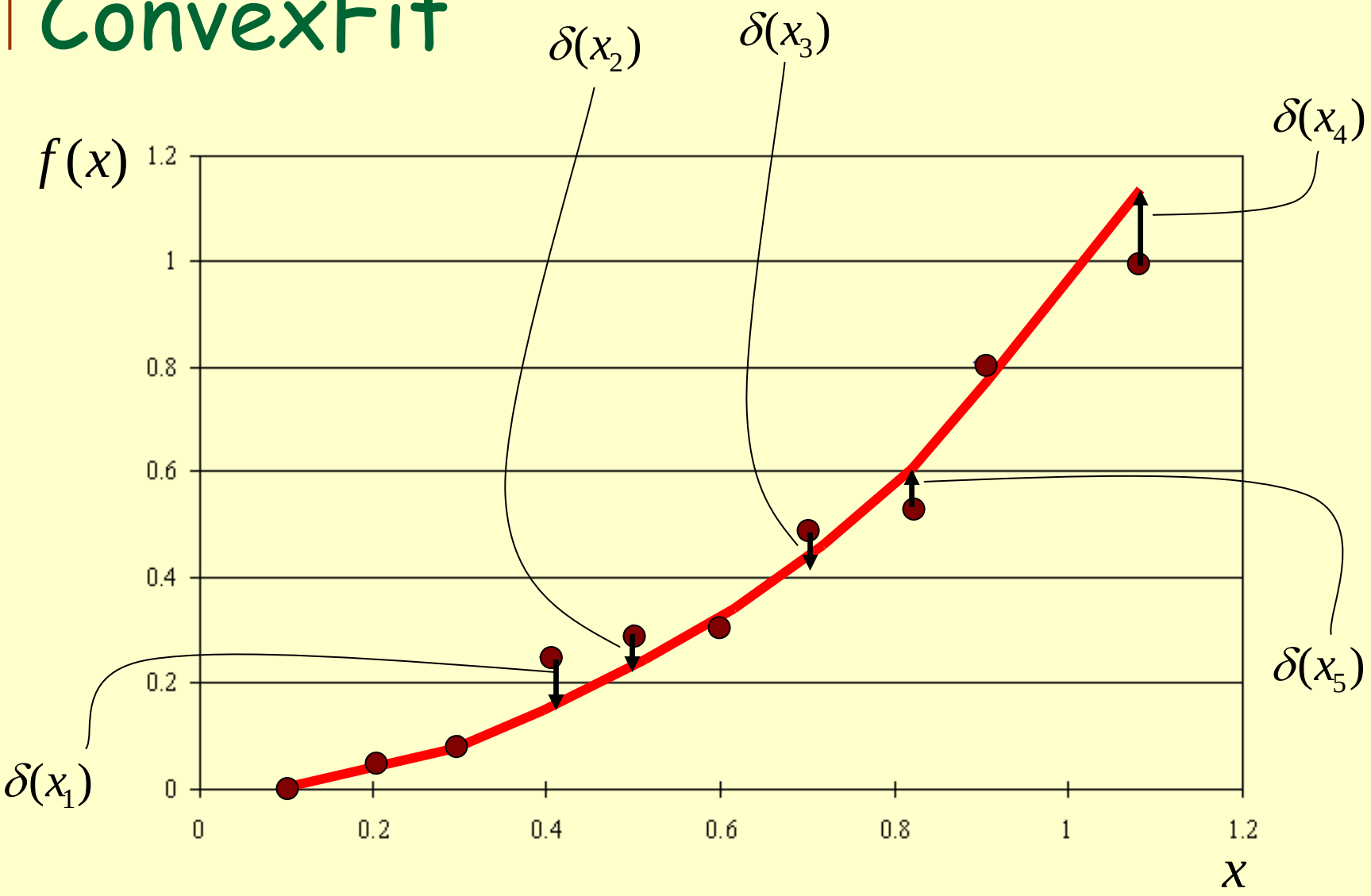
- Numerically convex: straight line joining any two points lies on or above the function
- A data set is numerically convex iff there exists a convex **piecewise linear interpolant** to the data
- Can approximate Hessian using finite difference



Set of Convex functions



ConvexFit


$\delta(x_2)$
$\delta(x_3)$
$\delta(x_4)$
$f(x)$
1.2
1
0.8
0.6
0.4
0.2
0
$\delta(x_5)$
$\delta(x_1)$
0
0.2
0.4
0.6
0.8
1
1.2
x

ConvexFit (Roy'05)

- ConvexFit Problem Formulation

Sufficient for convexity

$$\text{minimize} \quad \sum_{\mathbf{x_m} \in DOMf} \delta'(\mathbf{x_m})$$

$$\text{subject to} \quad \nabla^2 (f(\mathbf{x_m}) + \delta(\mathbf{x_m})) \succeq 0$$

$$-\delta'(\mathbf{x_m}) \le \delta(\mathbf{x_m}) \le \delta'(\mathbf{x_m})$$

$$\delta'(\mathbf{x_m}) \ge 0$$

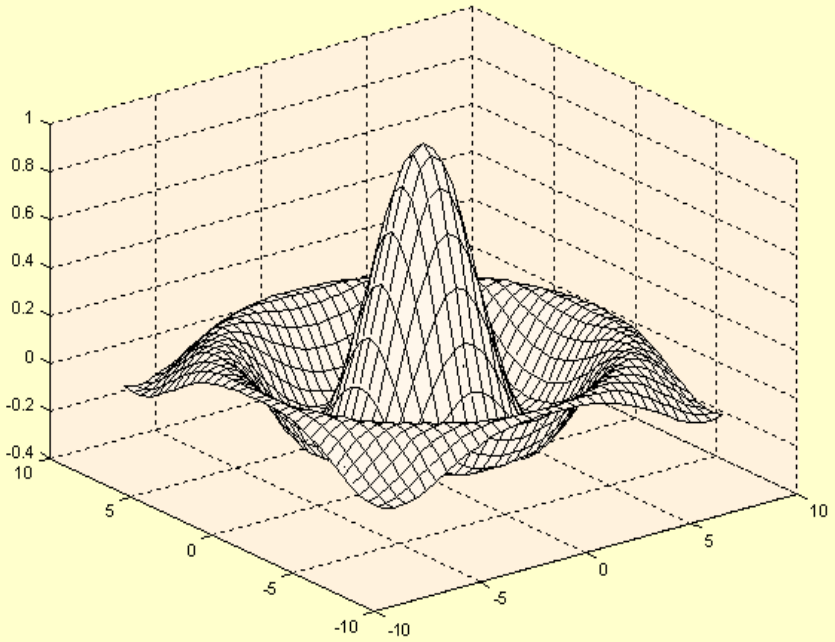
$$\mathbf{x_m} \in \mathrm{DOM}f$$

- Can be easily fitted into the semidefinite programming dual form : ***SDP***

Smooth function

- $for\ all\ \varepsilon > 0, there\ exists\ \delta > 0$

  $such\ that\ |y - x| < \delta \Rightarrow |\nabla f(y) - \nabla f(x)| < \varepsilon$



Why Smoothing?

- Smoothness necessary for quick convergence to optimal point
- In a numerical function smoothing helps in
  - Finding interpolated values between existing points
  - Improving accuracy of gradient and Hessian
- ConvexSmooth: simultaneous convex fitting and smoothing (Roy'06)
  - Very high execution time

Adjacent points in a grid

- Defined for particular direction *i*
- Adjacent points in i_{th} direction if
  - Identical coordinates for (n-1) directions
  - Adjacent coordinates for i_{th} direction

$$(x_1(k),\ldots.,\ x_i(j),\ldots.,\ x_n(l))$$

$$(x_1(k),\ldots.,\ x_i(j+1),\ldots.,\ x_n(l))$$

Non-smoothness numerical measure

- Non-smoothness index: maximum difference between gradients of adjacent points in the table

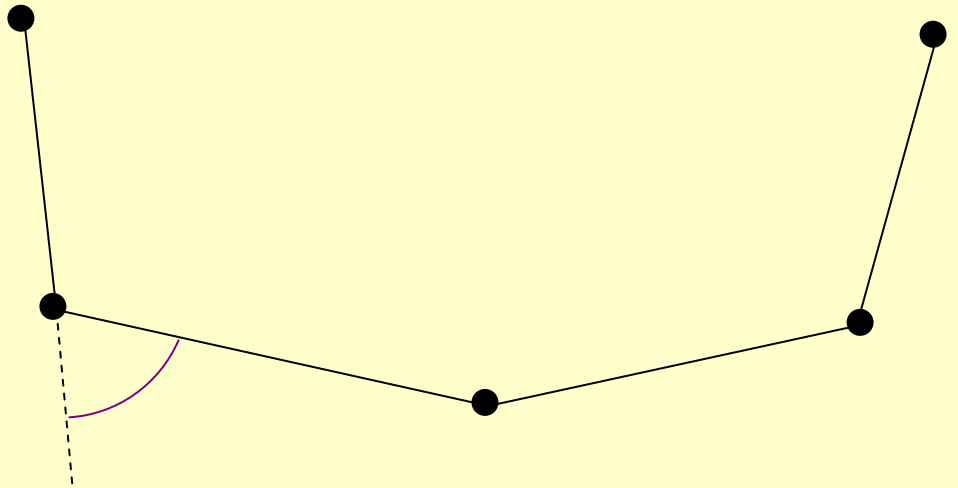
$$NSI(f) = \max\left|\nabla_i f(x) - \nabla_i f(y)\right|,$$

$$i \in [1,2,\ldots,n],$$

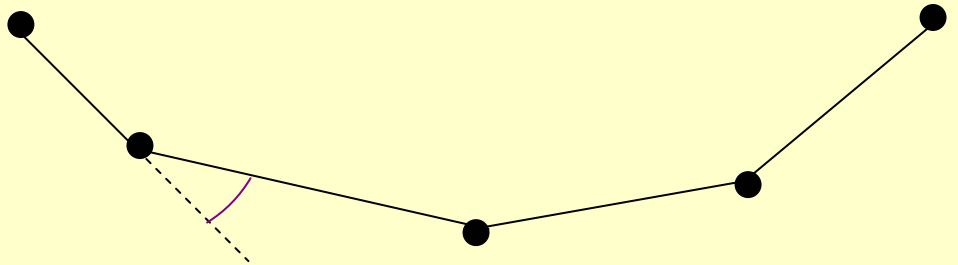
$$x, y \in \text{adjacent points in } DOMf$$

- **Smaller NSI(f), greater smoothness**
- Maximum allowable non-smoothness ε

Non-smoothness index



function a



function b

SmartSmooth: linear time convexity preserving smoothing algorithm

- Add data points: quadratic interpolation
- Divide domain of function into hypercubes
- Local optimization for each hypercube
  - Perturb newly added points
  - Enforce positive semidefinite Hessian
  - Enforce non-smoothness < ε
- Linear complexity

SmartSmooth problem formulation

Convexity constraint

minimize $\sum_{x \in DOMq_{hi}} \left| \phi'_{hi}(x) \right|$

subject to $\nabla^2 (g_{hi}(x'_{hi}) + \phi'_{hi}(x'_{hi})) \succeq 0,$

$-\varepsilon < \nabla_j f_{hi}(x'_{hi}) - \nabla_j f_{hi}(y'_{hi}) < \varepsilon,$

Smoothness constraint

$j \in [1,2,\ldots..,n],$

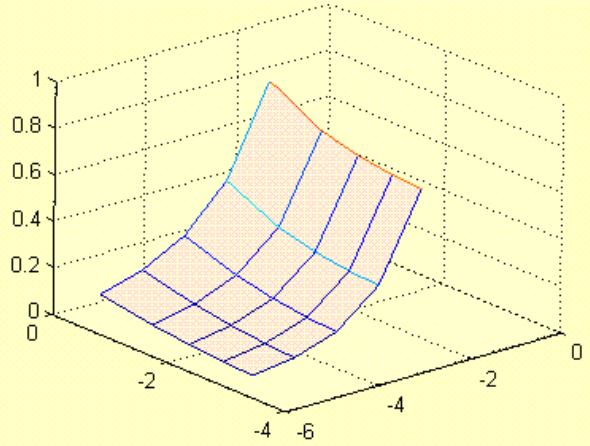
$\forall x'_{hi}, y'_{hi} \in$ adjacent points in $DOMf_{hi}$,

for $\varepsilon > 0$

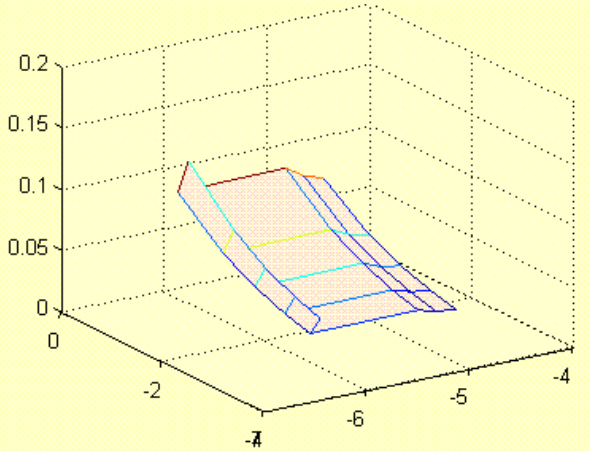
$g_{hi}(x'_{hi})$ is the subfunction in the i^{th} hypercube x'_{hi}

ConvexFit + SmartSmooth: cell INV

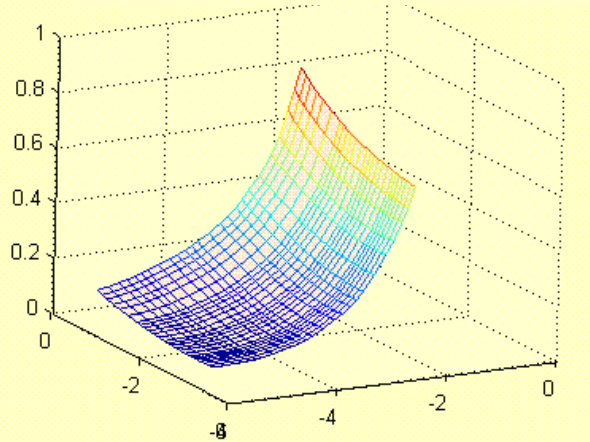
cell-rise Vs slew, load



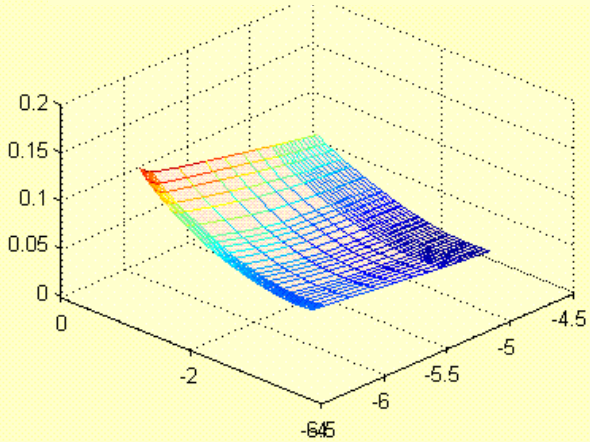
cell-rise Vs slew, input cap



ConvexFit + SmartSmooth



ConvexFit + SmartSmooth



Experimental setup

- $0.13\mu m$ Standard cell library: 415 core cells 53 I/O cells
- 67 combinational cells used for experiments
- DSDP5.7 solver in C for semidefinite optimization ConvexFit and SmartSmooth
- PosyFit implemented in C++ using CFSQP solver
- Four lookup tables for each cell : cell-rise, cell-fall, rise-transition and fall-transition: 49 - 392 data points

Fitting error comparison

Cell Name	PosyFit		ConvexFit + SSmooth	
	SE $(ns)^2$	AE (ns)	SE $(ns)^2$	AE (ns)
AN2	0.883	0.035	0.287	0.016
BUF	3.861	0.047	1.412	0.026
INV	9.427	0.087	0.052	0.004
MUX4	5.9	0.108	0.032	0.004
ND3	1.774	0.068	0.168	0.014
OR3B2	4.009	0.087	0.074	0.004
XOR2	2.537	0.064	0	0.0002

SE: total square error AE: average absolute error

Execution time: average time per cell

Mode	data points							
	196	196	1183	3610	8125	1183	3610	8125
	PFit	CFit	ConvexSmooth			CFit+SSmooth		
cell-rise	17s	2s	116s	489s	3024s	5s	11s	46s
cell-fall	19s	1s	117s	560s	2470s	4s	10s	44s
rise-transition	18s	2s	112s	403s	2551s	5s	10s	44s
fall-transition	18s	2s	117s	461s	3237s	5s	11s	46s

Tuning time for gate sizing

circuit	ConvexFit linear (sec)	ConvexFit SSmooth (sec)
C432	11	4
C499	3	1
C880	8	15
C1355	14	9
C1908	5	3
C2670	53	26
C3540	66	64
C5315	55	39
C6288	1952	347
C7552	169	110

1.76X speedup

Conclusion

- Convex optimization popular in VLSI design
- SmartSmooth: Linear time convexity preserving smoothing algorithm using semidefinite optimization
- 1.76X speedup in tuning time over linear interpolation
- Desired smoothness by setting *NSI* tolerance

THANKS