Geometric Programming for Circuit Design

Stephen Boyd Seung Jean Kim

EPFL 6/16/06

Outline

- Basic approach
- Geometric programming & generalized geometric programming
- Digital circuit design applications
- Analog and RF circuit design applications
- Monomial and posynomial fitting
- Software and modeling systems
- Conclusions

Basic Approach

Basic approach

1. formulate circuit design problem as **geometric program** (GP), an optimization problem with special form

2. solve GP using specialized, tailored method

- this tutorial focuses on step 1 (a.k.a. **GP modeling**)
- step 2 is **technology**

Why?

- we can solve even large GPs very effectively, using recently developed methods
- so once we have a GP formulation, we can solve circuit design problem effectively

we will see that

- GP is especially good at handling a large number of concurrent constraints
- GP formulation is useful even when it is approximate

Trade-offs in optimization

- general trade-off between **generality** and **effectiveness**
- generality
  - number of problems that can be handled
  - accuracy of formulation
  - ease of formulation
- effectiveness
  - speed of solution, scale of problems that can be handled
  - global vs. local solutions
  - reliability, baby-sitting, starting point

Example: least-squares vs. simulated annealing

least-squares

- large problems reliably (globally) solved quickly
- no initial point, no algorithm parameter tuning
- solves very restricted problem form
- with tricks and extensions, basis of vast number of methods that work (control, filtering, regression, . . .)

simulated annealing

- can be applied to any problem (more or less)
- slow, needs tuning, babysitting; not global in practice
- method of choice for some problems you can't handle any other way

Where GP fits in

somewhere in between, closer to least-squares . . .

- like least-squares, large problems can be solved reliably (globally), no starting point, tuning, . . .
- solves a class of problems broader than least-squares, less general than simulated annealing
- **formulation takes effort, but is fun and has high payoff**

Geometric Programming & Generalized Geometric Programming

Monomial & posynomial functions

$x = (x_1, \ldots, x_n)$: vector of positive optimization variables

- function g of form
$$g(x) = c x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n},$$
with $c > 0$, $\alpha_i \in \mathbf{R}$, is called **monomial**

- sum of monomials, *i.e.*, function f of form
$$f(x) = \sum_{k=1}^{t} c_k x_1^{\alpha_{1k}} x_2^{\alpha_{2k}} \cdots x_n^{\alpha_{nk}},$$
with $c_k > 0$, $\alpha_{ik} \in \mathbf{R}$, is called **posynomial**

Examples

with x, y, z variables,

- 0.23, $2z\sqrt{x/y}$, $3x^2y^{-.12}z$ are monomials (hence also posynomials)
- $0.23 + x/y$, $2(1 + xy)^3$, $2x + 3y + 2z$ are posynomials
- $2x + 3y - 2z$, $x^2 + \tan x$ are neither

Generalized posynomials

f is a **generalized posynomial** if it can be formed using addition, multiplication, positive power, and maximum, starting from posynomials

examples:

- $\max\left\{1+x_1, 2x_1+x_2^{0.2}x_3^{-3.9}\right\}$
- $\left(0.1x_1x_3^{-0.5}+x_2^{1.7}x_3^{0.7}\right)^{1.5}$
- $\left(\max\left\{1+x_1, 2x_1+x_2^{0.2}x_3^{-3.9}\right\}\right)^{1.7}+x_2^{1.1}x_3^{3.7}$

Composition rules

- **monomials** closed under product, division, positive scaling, power, inverse
- **posynomials** closed under sum, product, positive scaling, division by monomial, positive integer power
- **generalized posynomials** closed under sum, product, max, positive scaling, division by monomial, positive power

Generalized geometric program (GGP)

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 1, \quad i = 1, \ldots, m \\ & g_i(x) = 1, \quad i = 1, \ldots, p \end{array}$$

f_i are **generalized posynomials**, g_i are monomials

- called **geometric program (GP)** when f_i are **posynomials**
- a highly nonlinear constrained optimization problem

GP example

- maximize volume of box with width w, height h, depth d
- subject to limits on wall and floor areas, aspect ratios h/w, d/w

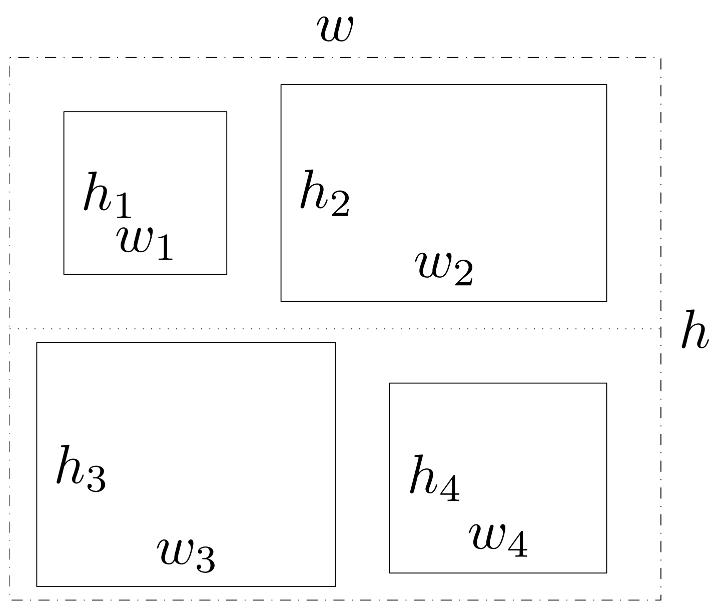
$$\begin{array}{ll} \text{maximize} & hwd \\ \text{subject to} & 2(hw+hd) \leq A_{\text{wall}}, \quad wd \leq A_{\text{flr}} \\ & \alpha \leq h/w \leq \beta, \quad \gamma \leq d/w \leq \delta \end{array}$$

in standard GP form:

$$\begin{array}{ll} \text{minimize} & h^{-1}w^{-1}d^{-1} \\ \text{subject to} & (2/A_{\text{wall}})hw + (2/A_{\text{wall}})hd \leq 1, \quad (1/A_{\text{flr}})wd \leq 1 \\ & \alpha h^{-1}w \leq 1, \quad (1/\beta)hw^{-1} \leq 1 \\ & \gamma wd^{-1} \leq 1, \quad (1/\delta)w^{-1}d \leq 1 \end{array}$$

GGP example: Floor planning

- choose cell widths, heights
- fixed cell areas
- (1 left of 2) above (3 left of 4)
- aspect ratio constraints
- minimize bounding box area



$$\begin{array}{ll} \text{minimize} & hw \\ \text{subject to} & h_i w_i = A_i, \quad 1/\alpha_{\max} \le h_i/w_i \le \alpha_{\max}, \\ & \max\{h_1, h_2\} + \max\{h_3, h_4\} \le h, \\ & \max\{w_1 + w_2, w_3 + w_4\} \le w \end{array}$$

. . . a GGP

Trade-off analysis

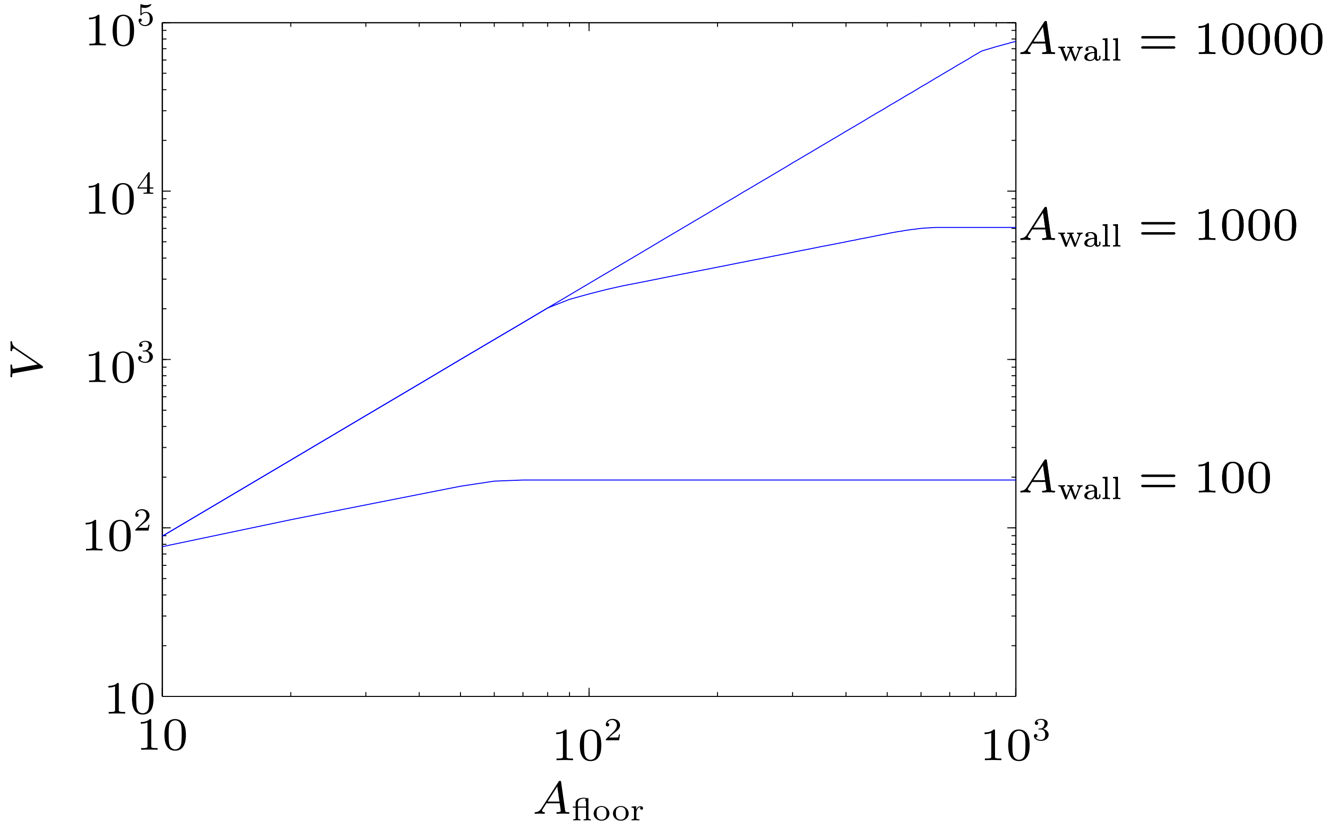
(no equality constraints, for simplicity)

- form perturbed version of original GP, with changed righthand sides:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq u_i, \quad i = 1, \ldots, m \end{array}$$

- $u_i > 1$ ($u_i < 1$) means ith constraint is relaxed (tightened)
- let $p(u)$ be optimal value of perturbed problem
- plot of p vs. u is (globally) **optimal trade-off surface** (of objective against constraints)

Trade-off curves for maximum volume box example



- maximum volume V vs. $A_{\rm flr}$, for $A_{\rm wall} = 100,\ 1000,\ 10000$
- h/w, d/w aspect ratio limits 0.5, 2

Sensitivity analysis

- optimal sensitivity of ith constraint is

$$S_i = \left. \frac{\partial p/p}{\partial u_i/u_i} \right|_{u=1}$$

- S_i predicts fractional change in optimal objective value if ith constraint is (slightly) relaxed or tightened

- very useful in practice; give quantitative measure of how tight a binding constraint is

- when we solve a GP **we get all optimal sensitivities at no extra cost**

Example

- minimize circuit delay, subject to power, area constraints (details later)

$$\begin{array}{ll} \text{minimize} & D(x) \\ \text{subject to} & P(x) \leq P^{\max}, \quad A(x) \leq A^{\max} \end{array}$$

- both constraints tight at optimal $x^\star$: $P(x^\star) = P^{\max}$, $A(x^\star) = A^{\max}$
- suppose optimal sensitivities are $S^{\rm pwr} = -2.1$, $S^{\rm area} = -0.3$
- we predict:
  - for 1% increase in allowed power, optimal delay decreases 2.1%
  - for 1% increase in allowed area, optimal delay decreases 0.3%

GP and GGP attributes

- after log transform of variables/constraints, they become **convex problems**

- can convert GGP to GP, $e.g.$, $f(x) + \max\{g(x), h(x)\} \leq 1$ becomes

$$f(x) + t \leq 1, \quad g(x)/t \leq 1, \quad h(x)/t \leq 1$$

  where t is new (dummy) variable

- **conversion tricks can be automated**
  - parser scans problem description, forms GP
  - efficient GP solver solves GP
  - solution transformed back (dummy variables eliminated)

How GPs are solved

the practical answer: **none of your business**

more politely: **you don't need to know**

it's **technology:**

- good algorithms are known
- good software implementations are available

How GPs are solved

- work with log of variables: $y_i = \log x_i$
- take log of monomials/posynomials to get

$$\begin{array}{ll} \text{minimize} & \log f_0(e^y) \\ \text{subject to} & \log f_i(e^y) \leq 0, \quad i = 1, \ldots, m \\ & \log g_i(e^y) = 0, \quad i = 1, \ldots, p \end{array}$$

- $\log f_i(e^y)$ are (smooth) **convex** functions
- $\log g_i(e^y)$ are affine functions, *i.e.*, linear plus a constant
- solve (nonlinear) **convex optimization problem** above using interior-point method

Current state of the art

- basic interior-point method that exploits sparsity, generic GP structure
- approaching efficiency of linear programming solver
  - sparse 1000 vbles, 10000 monomial terms: few seconds
  - sparse 10000 vbles, 100000 monomial terms: minute
  - sparse 10^6 vbles, 10^7 monomial terms: hour

  (these are order-of-magnitude estimates, on simple PC)

History

- GP (and term ‘posynomial’) introduced in 1967 by Duffin, Peterson, Zener

- engineering applications from the very beginning

  - early applications in chemical, mechanical, power engineering
  - digital circuit transistor and wire sizing with Elmore delay since 1984 (Fishburn & Dunlap’s TILOS)
  - analog circuit design since 1997 (Hershenson, Boyd, Lee)
  - other applications in finance, wireless power control, statistics, . . .

- extremely efficient solution methods since 1994 or so (Nesterov & Nemirovsky)

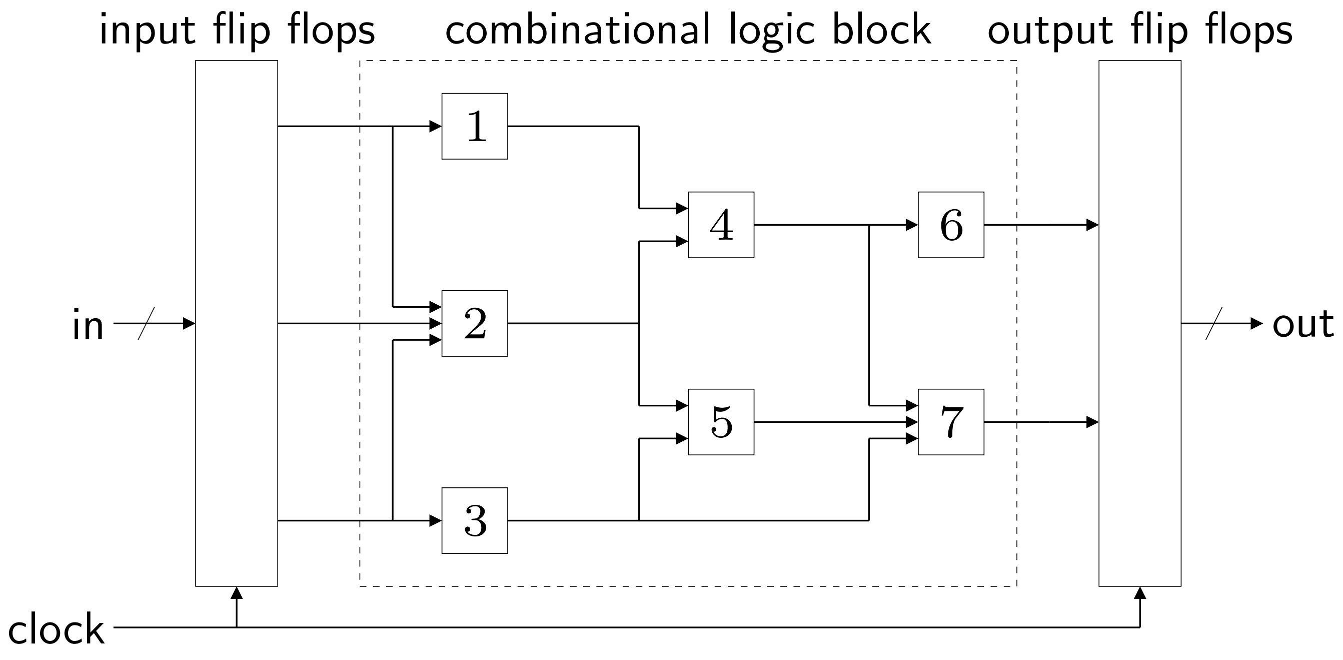
Mixed-integer geometric program

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 1, \quad i = 1, \ldots, m \\ & g_i(x) = 1, \quad i = 1, \ldots, p \\ & x_i \in \mathcal{D}_i, \quad i = 1, \ldots, k \end{array}$$

- f_i are generalized posynomials, g_i are monomials
- $\mathcal{D}_i$ are discrete sets, *e.g.*, $\{1, 2, 3, 4, \ldots\}$ or $\{1, 2, 4, 8 \ldots\}$
- **very hard** to solve exactly; all methods make some compromise (compared to methods for GP)
- **heuristic methods** attempt to find good approximate solutions quickly, but cannot guarantee optimality
- **global methods** always find the global solution, but can be extremely slow

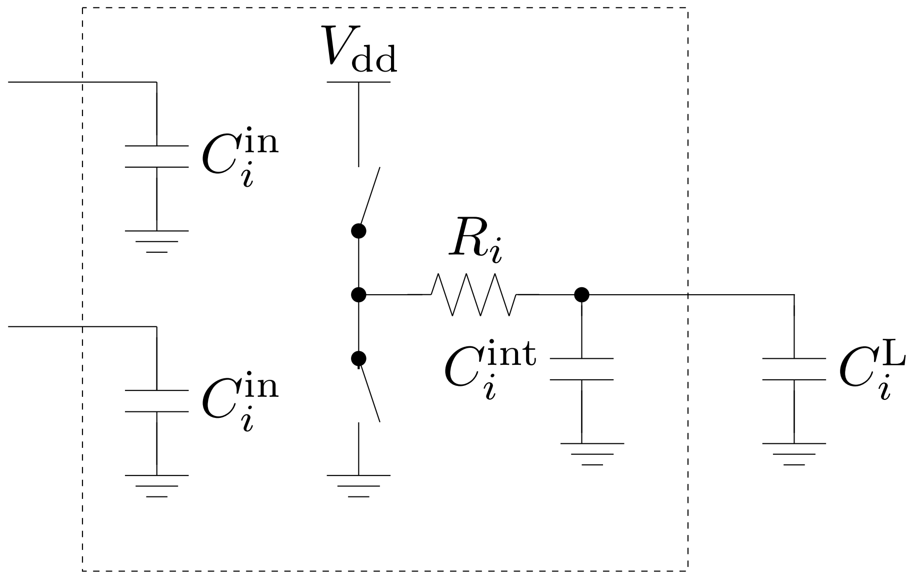
Digital Circuit Design Applications

Gate scaling



- combinational logic; circuit topology & gate types given
- gate sizes (scale factors $x_i \geq 1$) to be determined
- scale factors affect total circuit area, power and delay

RC gate delay model



- input & intrinsic capacitances, driving resistance, load capacitance

$$C_i^{\mathrm{in}} = \bar{C}_i^{\mathrm{in}} x_i, \qquad C_i^{\mathrm{int}} = \bar{C}_i^{\mathrm{int}} x_i, \qquad R_i = \bar{R}_i / x_i, \qquad C_i^{\mathrm{L}} = \sum_{j \in \mathrm{FO}(i)} C_j^{\mathrm{in}}$$

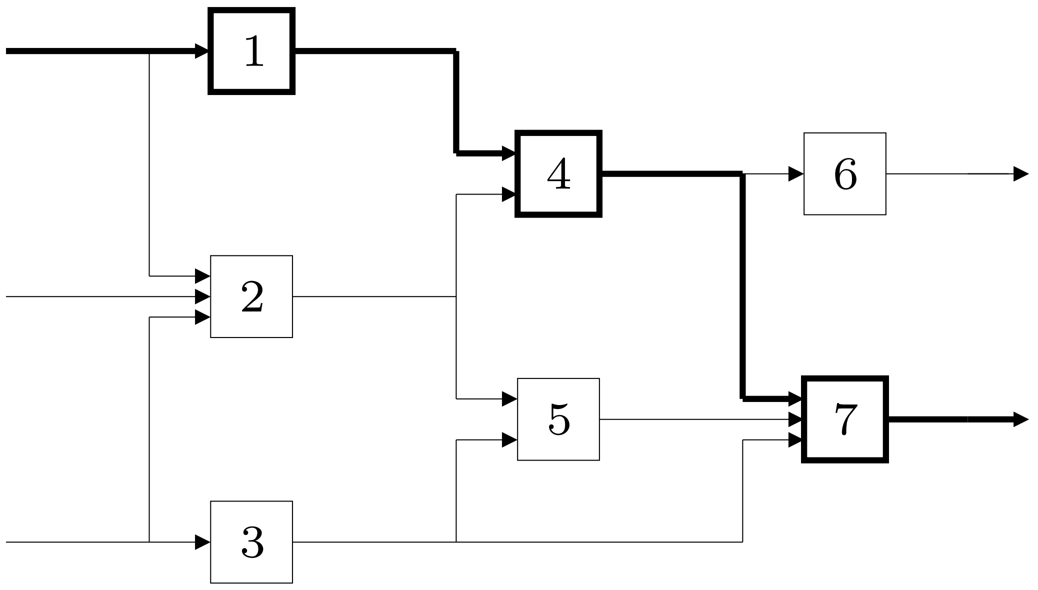
RC gate model

- RC gate delay:

$$D_i = 0.69 R_i (C_i^{\mathrm{L}} + C_i^{\mathrm{int}}) = 0.69 \left(\bar{R}_i \bar{C}_i^{\mathrm{in}} + (\bar{R}_i / x_i) \sum_{j \in \mathrm{FO}(i)} \bar{C}_j^{\mathrm{in}} x_j \right)$$

- D_i are **posynomials** (of scale factors)

Path and circuit delay



- delay of a path: sum of delays of gates on path
  . . . **posynomial**

- circuit delay: maximum delay over all paths
  . . . **generalized posynomial**

Area & power

- total circuit area: $A = x_1 \bar{A}_1 + \cdots + x_n \bar{A}_n$

- total power is $P = P_{\mathrm{dyn}} + P_{\mathrm{stat}}$

  – dynamic power $P_{\mathrm{dyn}} = \displaystyle\sum_{i=1}^{n} f_i (C_i^{\mathrm{L}} + C_i^{\mathrm{int}}) V_{\mathrm{dd}}^2$

    f_i is gate switching frequency

  – static power $P_{\mathrm{stat}} = \displaystyle\sum_{i=1}^{n} x_i \bar{I}_i^{\mathrm{leak}} V_{\mathrm{dd}}$

    $\bar{I}_i^{\mathrm{leak}}$ is leakage current (average over input states) of unit scaled gate

- A and P are linear functions of x, with positive coefficients, hence posynomials

Basic gate scaling problem

$$\begin{array}{ll} \text{minimize} & D \\ \text{subject to} & P \le P^{\max}, \quad A \le A^{\max} \\ & 1 \le x_i, \quad i = 1, \ldots, n \end{array}$$

. . . a **GGP**

extensions/variations:

- minimize area, power, or some combination
- maximize clock frequency subject to area, power limits
- add other constraints
- optimal trade-off of area, power, delay

Clock frequency maximization

- f_{clk} is variable
- timing requirement: $D \leq 0.8/f_{\mathrm{clk}}$
  (20% margin for flip-flop delay, setup time, clock skew . . .)
- P is posynomial of scalings and f_{clk}, assuming f_i scale with f_{clk}

$$\begin{array}{ll} \text{maximize} & f_{\mathrm{clk}} \\ \text{subject to} & P \leq P^{\max}, \quad A \leq A^{\max}, \quad (1/0.8) D f_{\mathrm{clk}} \leq 1, \\ & 1 \leq x_i, \quad i = 1, \ldots, n \end{array}$$

. . . a **GGP**

Example: 32-bit Ladner-Fisher adder

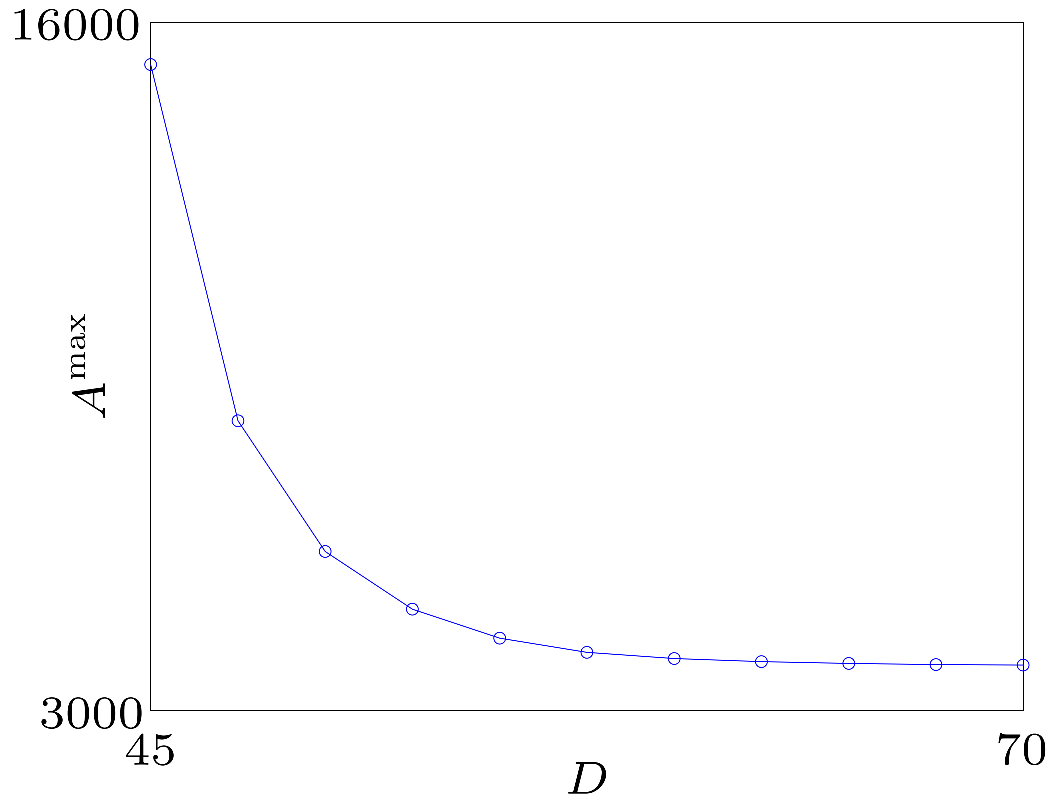
- 451 gates (scale factors), 5 gate types, 64 inputs, 32 outputs
- logical effort gate delay model parameters:

gate type	$\bar{C}^{\mathrm{in}}$	$\bar{C}^{\mathrm{int}}$	$\bar{R}$	$\bar{A}$	$\bar{I}^{\mathrm{leak}}$
INV	3	3	0.48	3	0.006
NAND2	4	6	0.48	8	0.007
NOR2	5	6	0.48	10	0.009
AOI21	6	7	0.48	17	0.003
OAI21	6	7	0.48	16	0.003

- time unit is τ, delay of min-size inverter $(0.69 \cdot 0.48 \cdot 3 = 1)$
- area (total width) unit is width of NMOS in min-size inverter

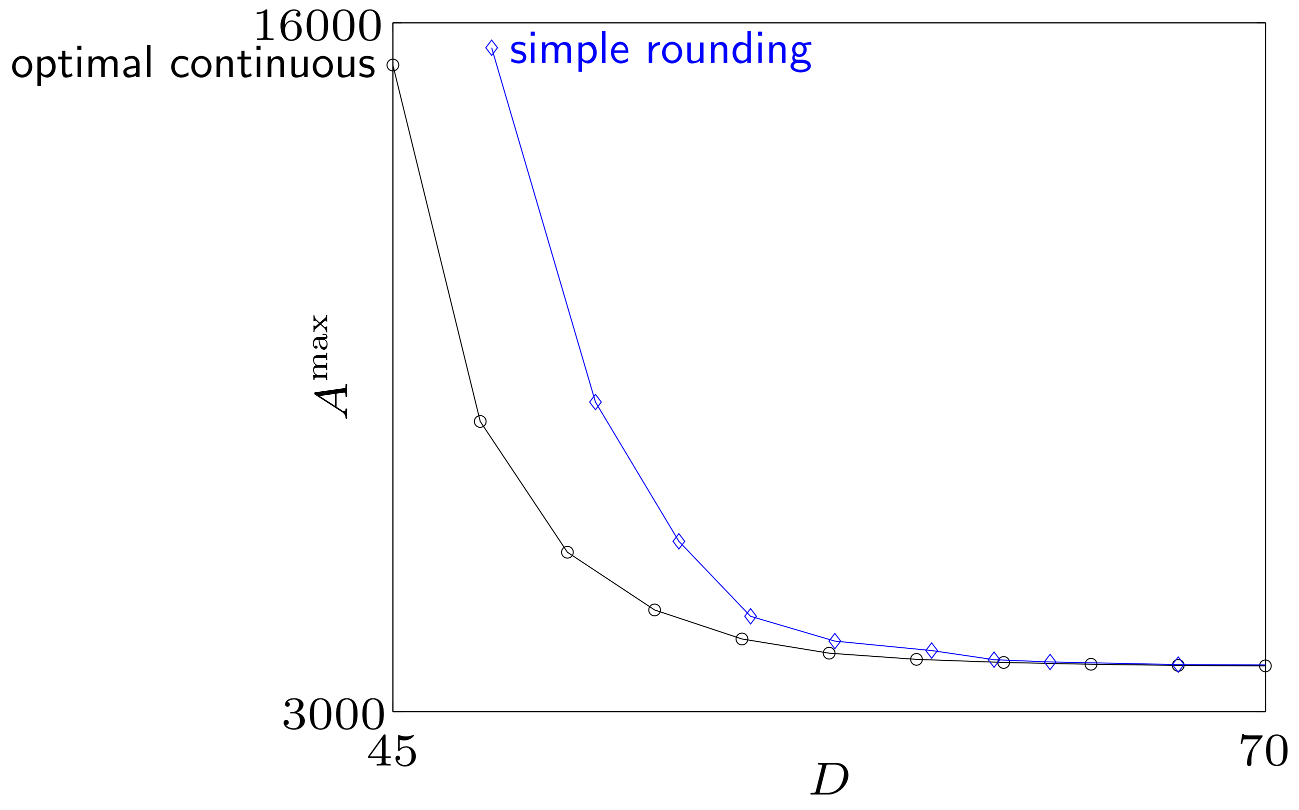
Example: 32-bit Ladner-Fisher adder

- typical optimization time: few seconds on PC



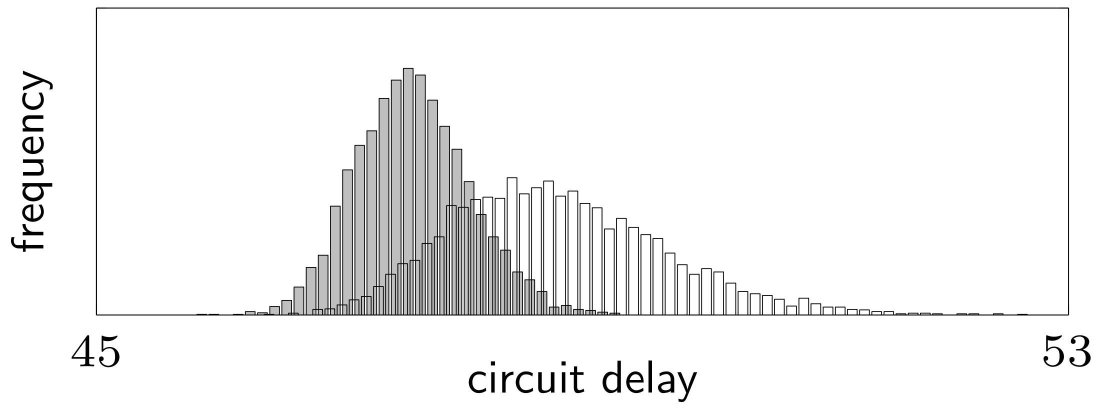
32-bit Ladner-Fisher adder with discrete scale factors

- add constraints $x_i \in \{1, 2, 4, 8, 16, \ldots\}$
- simple rounding of optimal continuous scalings



Statistical parameter variation

- circuit peformance depends on random device and process parameters
- hence, performance measures like P, D are random variables $\mathbf{P}$, $\mathbf{D}$
- delay **D** is max of many random variables; often skewed to right
- **distributions** of $\mathbf{P}$, $\mathbf{D}$ depend on gate scalings x_i



- related to (parametric) yield, DFM, DFY . . .

Statistical design

- measure random performance measures by 95% quantile (say)

$$\begin{array}{ll} \text{minimize} & \mathbf{Q}^{.95}(\mathbf{D}) \\ \text{subject to} & \mathbf{Q}^{.95}(\mathbf{P}) \leq P^{\max}, \qquad A \leq A^{\max} \\ & 1 \leq x_i, \quad i = 1, \ldots, n \end{array}$$

- **extremely difficult** stochastic optimization problem; almost no analytic/exact results

- but, (GP-compatible) heuristic method works well

Statistical model

- for simplicity consider $V_{\rm th}$ variation only
- Pelgrom's model: $\sigma_{V_{\rm th}} = \bar{\sigma}_{V_{\rm th}} x^{-1/2}$
- alpha-power law model: $D \propto V_{\rm dd}/(V_{\rm dd} - V_{\rm th})^{\alpha}$, with $\alpha \approx 1.3$
- for small variation in $V_{\rm th}$,

$$\sigma_D \approx \left|\frac{\partial D}{\partial V_{\rm th}}\right| \sigma_{V_{\rm th}} = \alpha (V_{\rm dd} - V_{\rm th})^{-1} \bar{\sigma}_{V_{\rm th}} x^{-0.5} D$$

- σ_D is posynomial
- get similar (posynomial) models for σ_D with more complex gate delay statistical models

Heuristic for statistical design

- assume generalized posynomial models for gate delay mean $D_i(x)$ and variance $\sigma_i(x)^2$
- optimize using **surrogate gate delays**

$$\tilde{D}_i(x) = D_i(x) + \kappa_i \sigma_i(x)$$

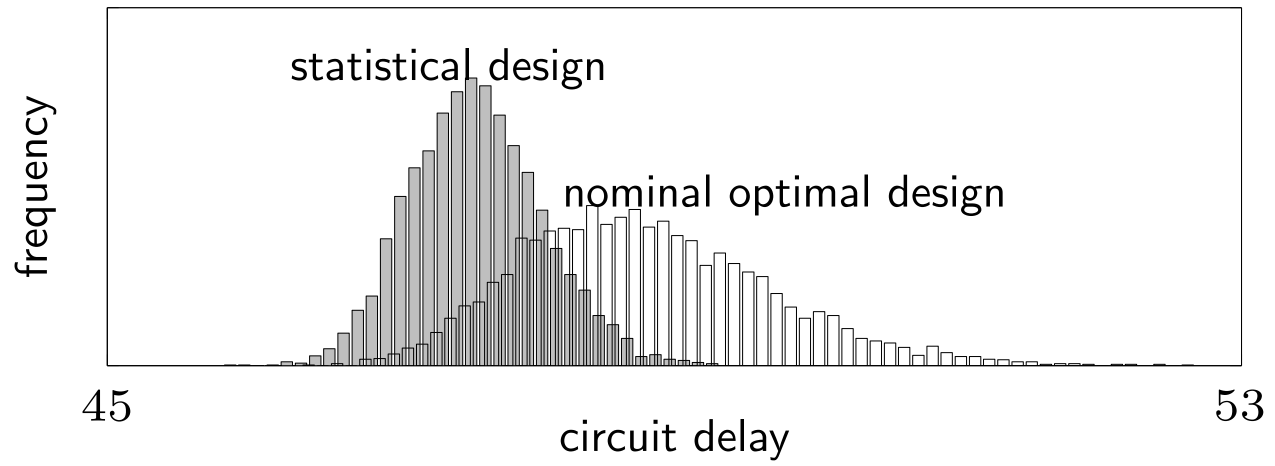
  $\kappa_i \sigma_i(x)$ are **margins** on gate delays (κ_i is typically 2 or 3)

- verify statistical performance via Monte Carlo analysis (can update κ_i's and repeat)

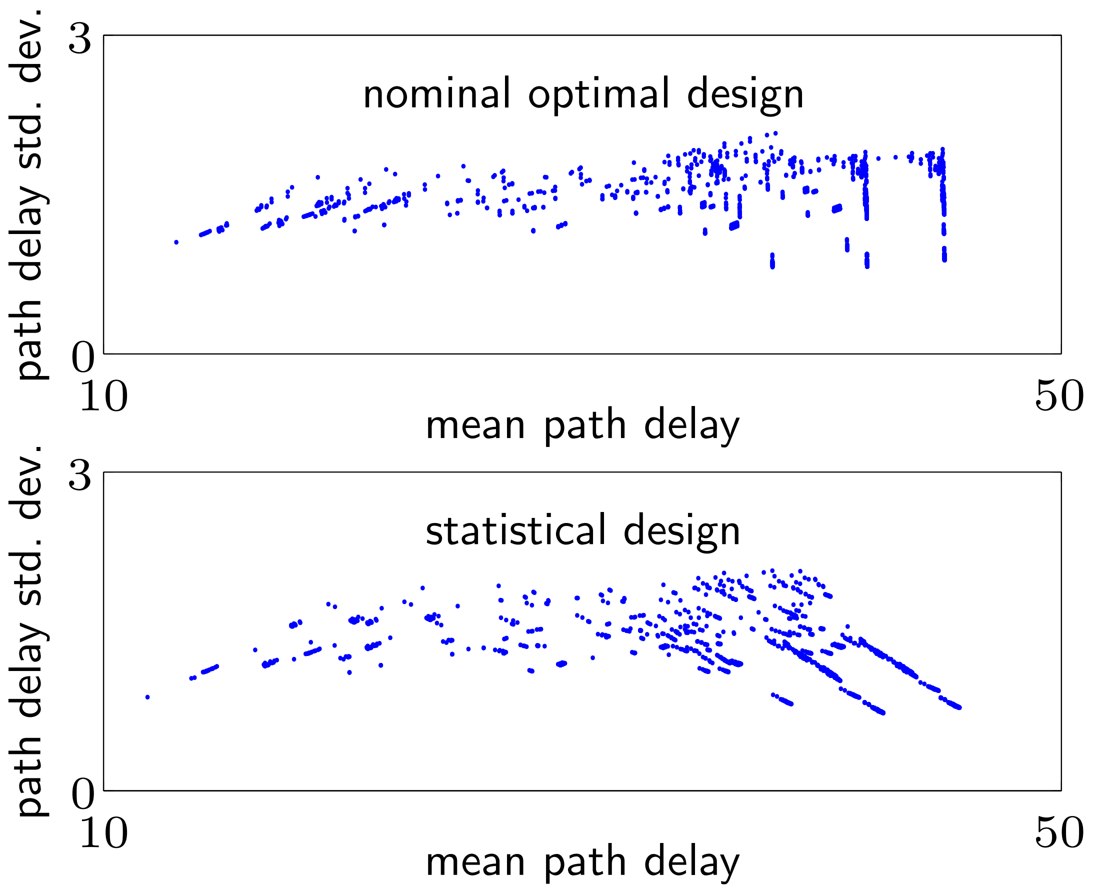
Heuristic for statistical design

heuristic statistical design

- often far superior to design obtained ignoring statistical variation
- not very sensitive to details of process variation statistics (distribution shape, correlations, . . .)
- below: 32-bit Ladner-Fisher adder, Pelgrom variance model



Path delay mean/std. dev. scatter plots



Joint size and supply/threshold voltage optimization

- **goal:** jointly optimize gate size, supply and threshold voltages via GGP
- **need to:** model delay, power as generalized posynomial functions of gate size, supply and threshold voltages

Generalized posynomial delay model

- alpha-power law model predicts variation in gate delay with $V_{\rm dd}$, $V_{\rm th}$:

$$D_i = \frac{V_{\mathrm{dd},i}}{(V_{\mathrm{dd},i} - V_{\mathrm{th},i})^\alpha} \tilde{D}_i(x)$$

  $\tilde{D}_i$ is generalized posynomial gate delay model, function of scalings x

- generalized posynomial approximation

$$\widehat{D}_i = V_{\mathrm{dd},i}^{1-\alpha} (1 + V_{\mathrm{th},i}/V_{\mathrm{dd},i} + \cdots + (V_{\mathrm{th},i}/V_{\mathrm{dd},i})^5)^\alpha \tilde{D}_i(x)$$

  error under 1% for $V_{\mathrm{dd},i} \geq 2V_{\mathrm{th},i}$, $1.3 \leq \alpha \leq 2$

Generalized posynomial power model

- gate dynamic power: $P_{\mathrm{dyn}} = \sum_{i=1}^{n} f_i (C_i^{\mathrm{L}} + C_i^{\mathrm{int}}) V_{\mathrm{dd},i}^2$

- simple static power model:

$$P_{\mathrm{stat}} = \sum_{i=1}^{n} x_i \bar{I}_i^{\mathrm{leak}} V_{\mathrm{dd},i}, \qquad \bar{I}_i^{\mathrm{leak}} \propto e^{-(V_{\mathrm{th},i} - \gamma V_{\mathrm{dd},i})/V_0}$$

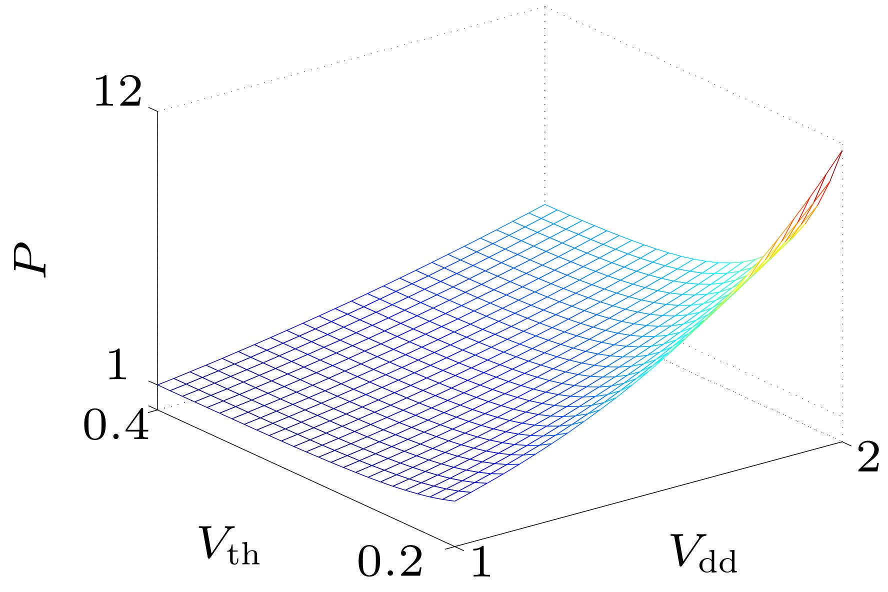
  γ, V_0 are (process) constants

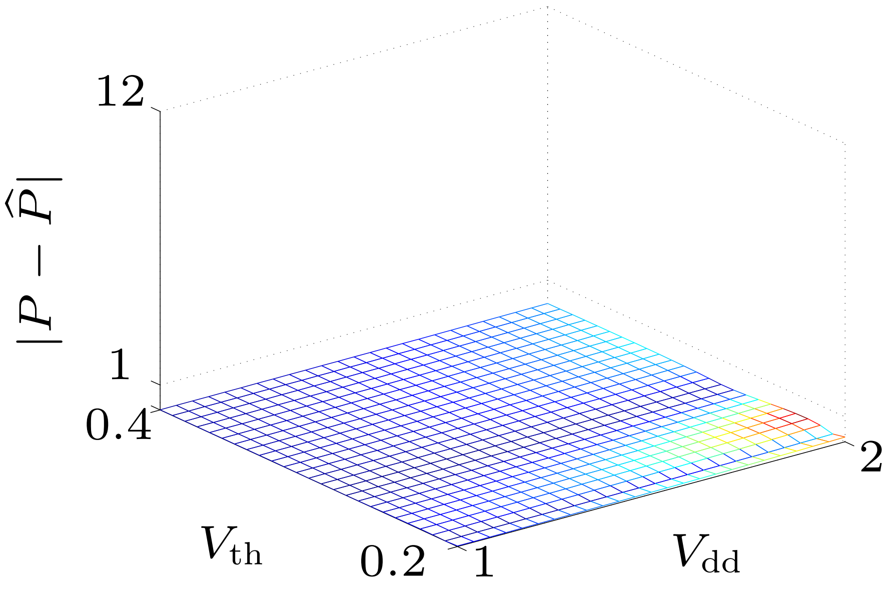
- P_{stat} (by itself) **cannot** be approximated well by a generalized posynomial over large range of V_{dd}, V_{th}

- but, total power $P = P_{\mathrm{dyn}} + P_{\mathrm{stat}}$ **can** be approximated well by a generalized posynomial

Generalized posynomial power model example

total power $P = V_{\rm dd}^2 + 30V_{\rm dd}e^{-(V_{\rm th}-0.06V_{\rm dd})/0.039}$ (up to scaling)





- generalized posynomial approximation
  $\widehat{P} = V_{\rm dd}^2 + 0.06V_{\rm dd}(1+0.0031V_{\rm dd})^{500}(V_{\rm th}/0.039)^{-6.16}$

- error under 3% (well under accuracy of model!)

Joint optimization of gate sizes, $V_{\rm dd}$, & $V_{\rm th}$

basic problem, with variables: x_i, $V_{{\rm th},i}$, $V_{{\rm dd},i}$

$$\begin{array}{ll} \mbox{minimize} & D \\ \mbox{subject to} & P \le P^{\max}, \qquad A \le A^{\max} \\ & V_{\rm th}^{\min} \le V_{{\rm th},i} \le V_{\rm th}^{\max}, \quad i = 1, \ldots, n \\ & V_{\rm dd}^{\min} \le V_{{\rm dd},i} \le V_{\rm dd}^{\max}, \quad i = 1, \ldots, n \\ & \mbox{other constraints} \ldots \end{array}$$

(. . . **a GGP**)

discrete allowed $V_{\rm dd}$, $V_{\rm th}$ values yields MIGP

Extensions/variations

- clustering, with single $V_{\rm dd}$, $V_{\rm th}$ per cluster:

$$V_{{\rm dd},i} = V_{{\rm dd},j}, \quad V_{{\rm th},i} = V_{{\rm th},j} \quad \text{for } i, j \text{ in same cluster}$$

  . . . monomial (equality) constraints

- clustered voltage scaling (CVS): low $V_{\rm dd}$ cells cannot drive high $V_{\rm dd}$ cells

$$V_{{\rm dd},j} \leq V_{{\rm dd},i} \quad \text{for } j \in {\rm FO}(i)$$

  . . . monomial (inequality) constraints

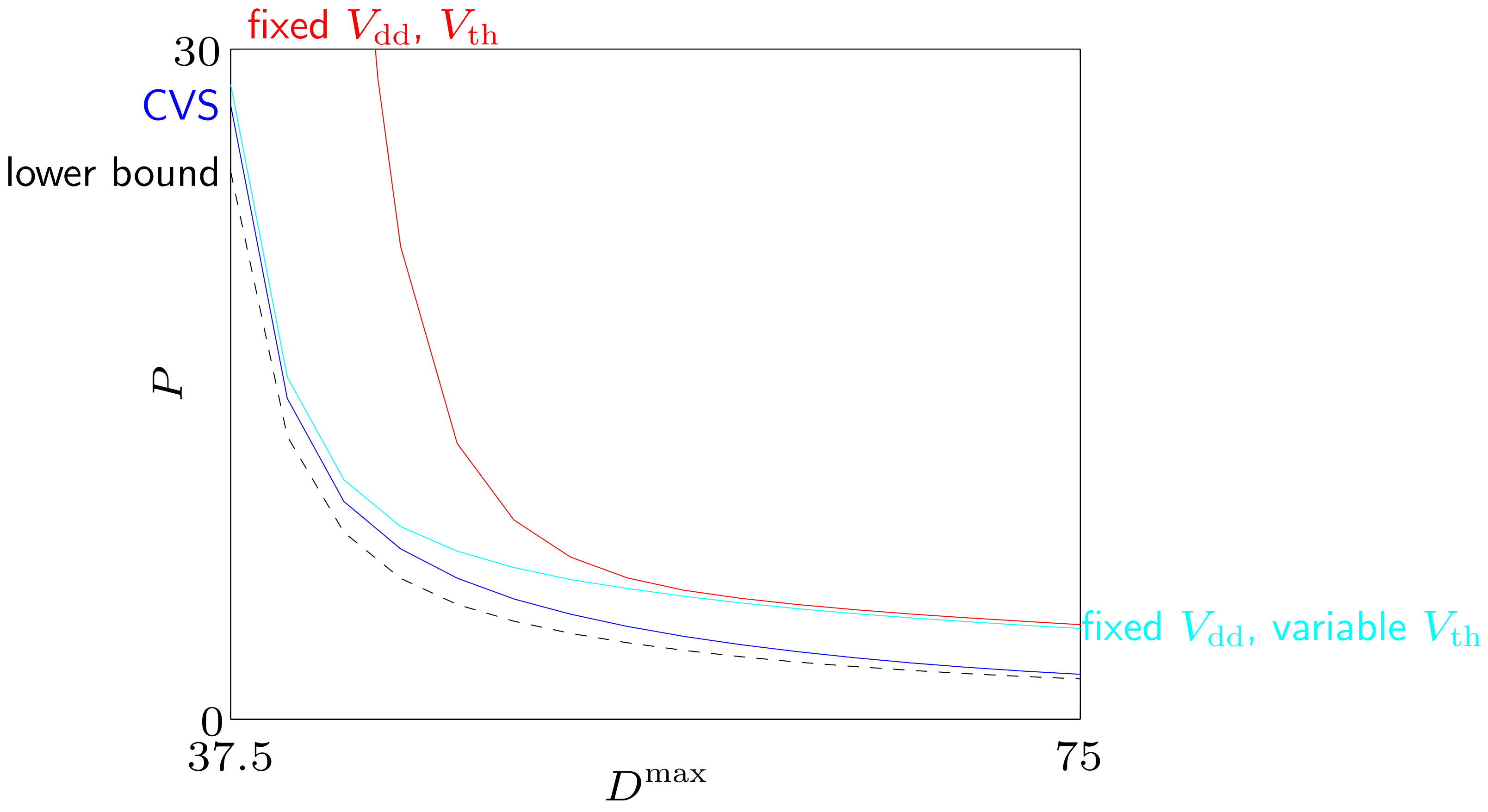
- multimode design: choose single set of gate scalings, different $V_{\rm dd}^{(k)}$, $V_{\rm th}^{(k)}$ for each scenario $k = 1, \ldots, K$

  related to **dynamic voltage scaling**, **adaptive bulk biasing**, . . .

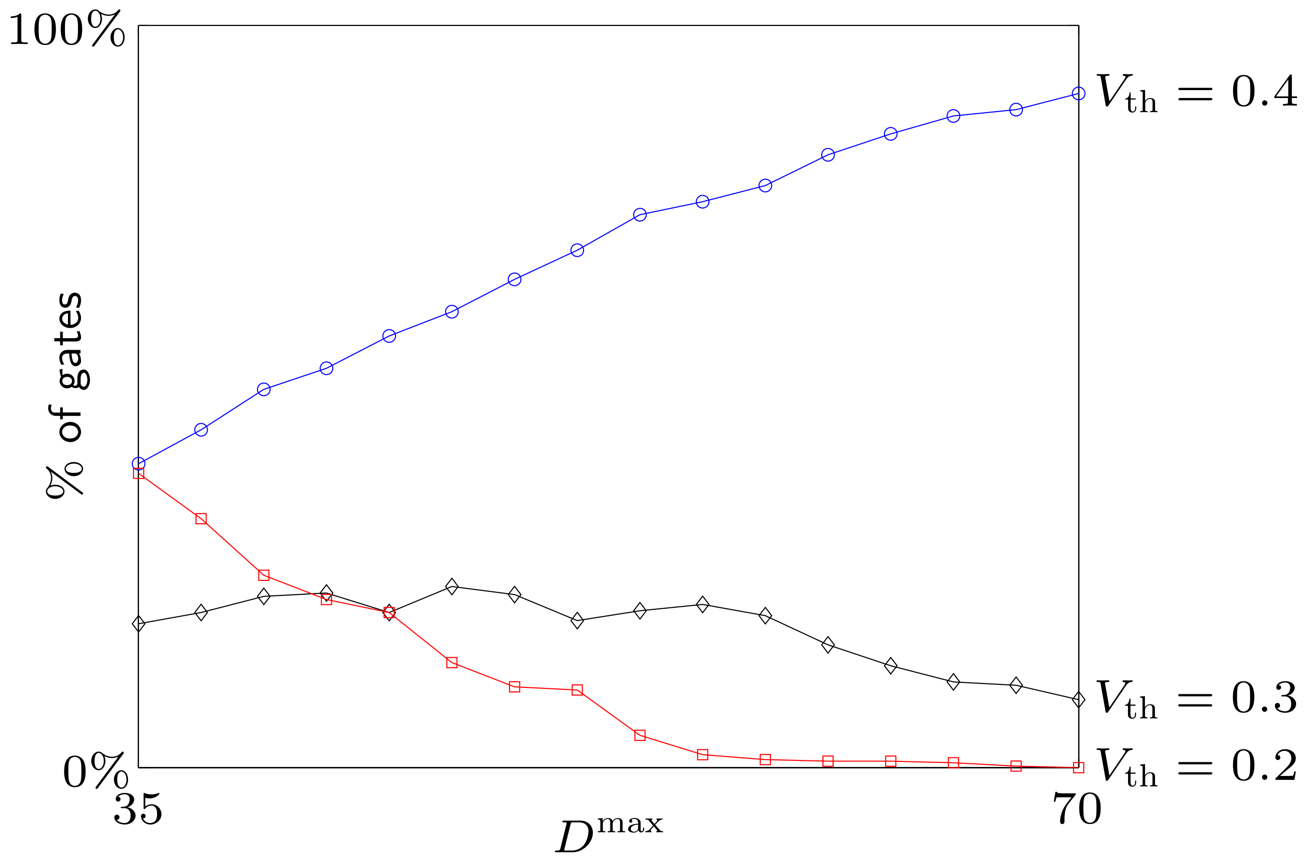
Joint optimization examples

- Ladner-Fisher adder
- variables: gate scalings x_i, supply voltages $V_{\mathrm{dd},i}$, threshold voltages $V_{\mathrm{th},i}$
- four delay-power trade-off curves:
  - fixed $V_{\mathrm{dd},i} = 1.0$, fixed $V_{\mathrm{th},i} = 0.3$
  - fixed $V_{\mathrm{dd},i} = 1.0$, variable $V_{\mathrm{th},i} \in \{0.2, 0.3, 0.4\}$
  - CVS with $V_{\mathrm{dd},i} \in \{0.6, 1.0\}$, $V_{\mathrm{th},i} \in \{0.2, 0.3, 0.4\}$
  - variable continuous V_{dd}, V_{th}, $0.6 \leq V_{\mathrm{dd},i} \leq 1.0$, $0.2 \leq V_{\mathrm{th},i} \leq 0.4$ (not practical, but serves as lower bound)

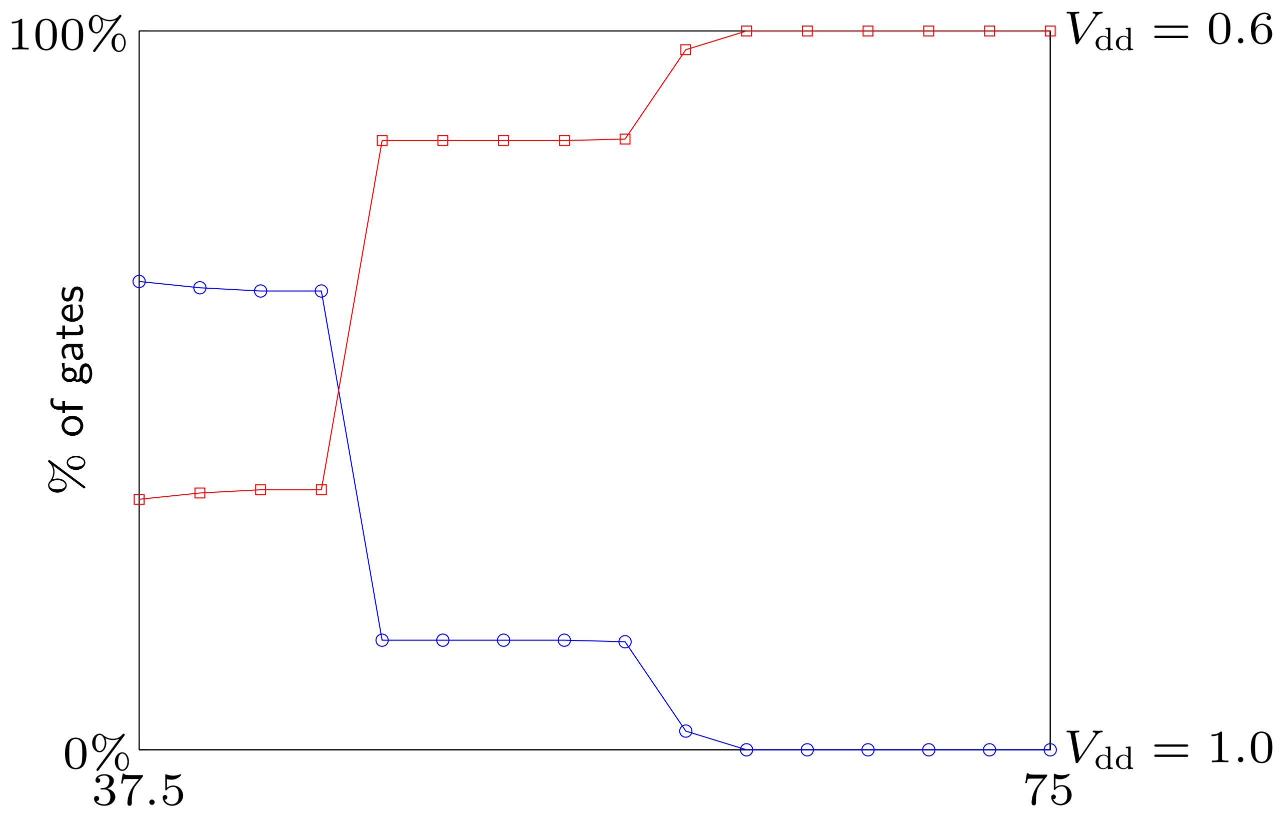
Trade-off curve analysis



Design with multiple threshold voltages

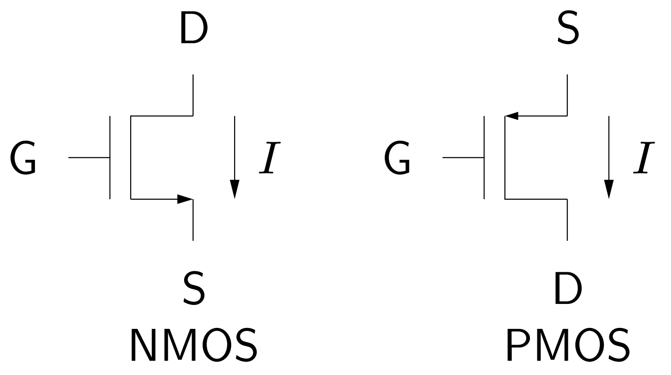


Clustered voltage scaling



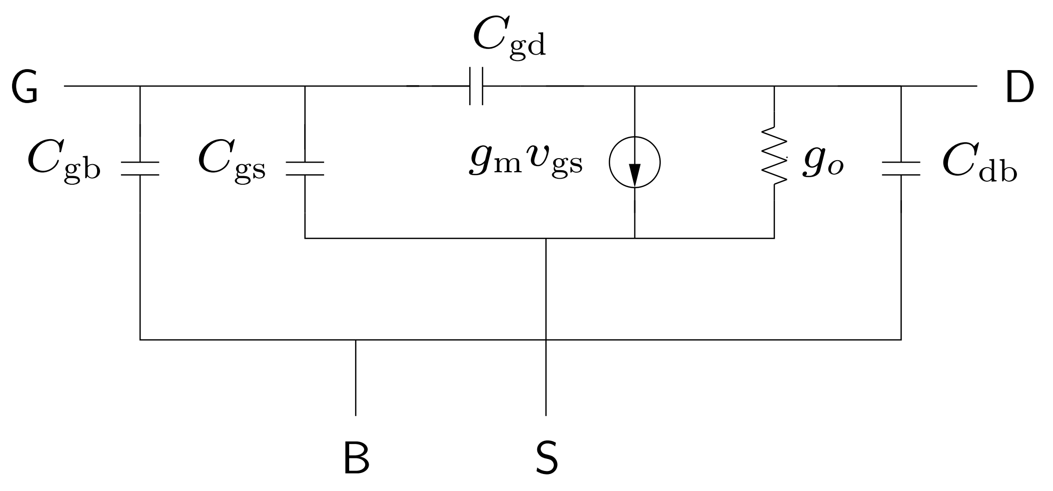
Analog Circuit Design Applications

Large signal MOS model



- gate overdrive voltage $V_{\text{gov}} = V_{\text{gs}} - V_{\text{th}}$
- saturation condition: $V_{\text{ds}} \geq V_{\text{dsat}} = V_{\text{gov}}$ (V_{dsat} is minimum drain-source voltage for device to operate in saturation)
- square-law model $I = 0.5\mu C_{\text{ox}}(W/L)V_{\text{gov}}^2$
- GP model variables: $I,\ L,\ W$
- $V_{\text{gov}} = (\mu C_{\text{ox}}/2)^{-1/2} I^{1/2} L^{1/2} W^{-1/2}$ is monomial
- $V_{\text{gs}} = V_{\text{gov}} + V_{\text{th}}$ is posynomial

Small signal dynamic MOS model



- transconductance $g_{\mathrm{m}} = (2\mu C_{\mathrm{ox}})^{1/2} I^{1/2} L^{-1/2} W^{1/2}$ is monomial
- output conductance $g_{\mathrm{o}} = \lambda I$ is monomial
- all capacitances are (approximately) posynomial in I, L, W
- better (GP-compatible) models can be obtained by fitting data from accurate models or measurements

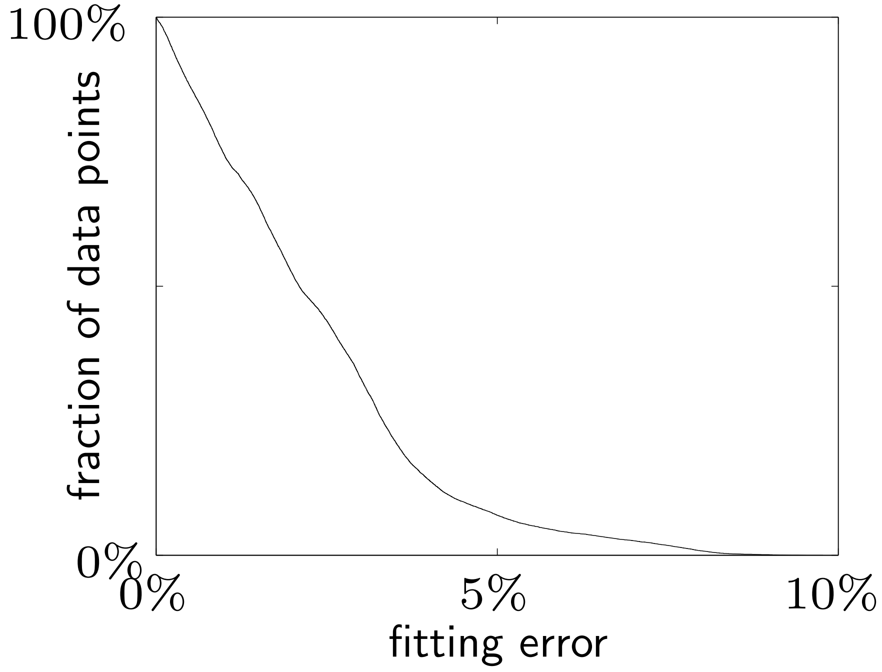
Example: monomial g_m model

- monomial model of g_m for I/O NMOS device in a 0.13μm technology
- 11000 data points (from BSIM3) over ranges
  - $0.3\mu\text{m} \leq L \leq 3\mu\text{m}$, $2\mu\text{m} \leq W \leq 20\mu\text{m}$
  - $0.7\text{V} \leq V_\mathrm{gs} \leq 1.7\text{V}$, $V_\mathrm{dsat} \leq V_\mathrm{ds} \leq 1.5V_\mathrm{gs}$
- V_ds appears in data set, but not in g_m model
- monomial fit (using simple log-regression, SI units):

$$g_\mathrm{m} = 0.0278 I^{0.4798} L^{-0.511} W^{0.5632}$$

Example: monomial g_{m} model

- fitting (relative) error cumulative distribution plot:

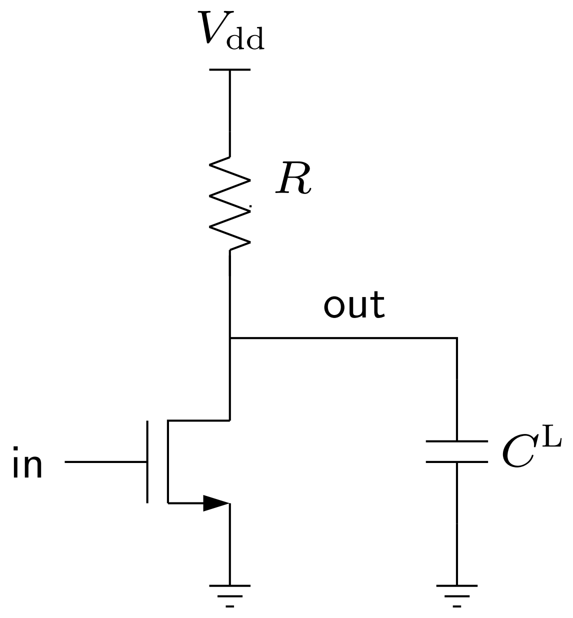


- for 90% of points, fit is better than 4%

Single transistor common source amplifier

- variables: I, L, W, R
- saturation: $V_{\mathrm{dsat}} + IR \leq V_{\mathrm{dd}}$
- gain $G = g_{\mathrm{m}}/(1/R + g_{\mathrm{o}})$
- power $P = V_{\mathrm{dd}} I$
- (unity gain) bandwidth $B = g_{\mathrm{m}}/C^{\mathrm{L}}$
- design problem:

$$\begin{array}{ll} \text{minimize} & P \\ \text{subject to} & B \geq B^{\min}, \quad G \geq G^{\min} \\ & \text{saturation} \end{array}$$



Common source amplifier design via GP

- rewrite as

$$\begin{array}{ll} \text{minimize} & P \\ \text{subject to} & B^{-1} \leq 1/B^{\min}, \quad G^{-1} \leq 1/G^{\min} \\ & V_{\rm dsat} + IR \leq V_{\rm dd} \end{array}$$

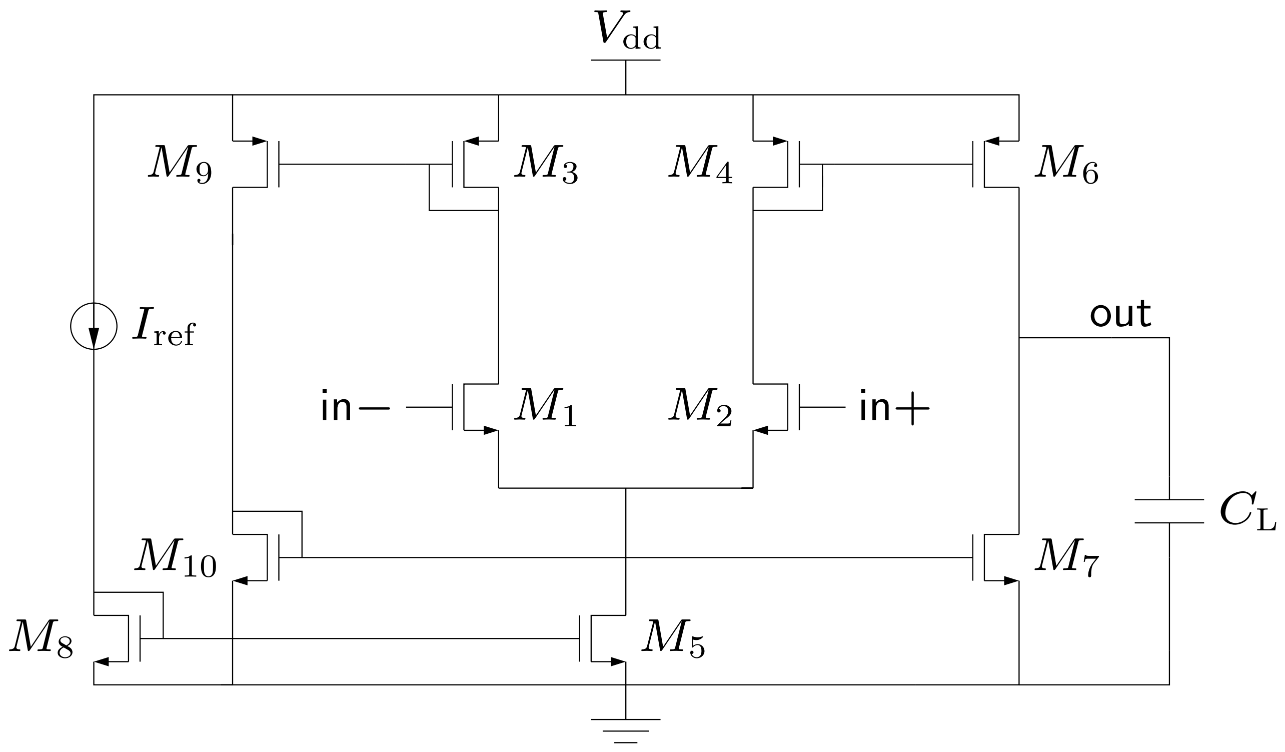
- . . . a **GP**, since P and B are monomials, and

$$G^{-1} = \frac{1/R + g_{\rm o}}{g_{\rm m}}$$

  is posynomial

- this is a simple problem; don't need GP sledgehammer . . .

Current mirror opamp



- M_1, M_2 and M_3, M_4 matched pairs
- four current mirrors: M_8, M_5; M_{10}, M_7; M_9, M_3; M_4, M_6

Design problem

$$\begin{array}{ll} \text{minimize} & P \\ \text{subject to} & B \geq B^{\min}, \quad G \geq G^{\min}, \quad A \leq A^{\max} \\ & \text{other constraints } \ldots \end{array}$$

- objective & specifications:
  - P is power dissipation
  - B is unity gain bandwidth
  - G is DC gain
  - A is (active) area
- design variables: $L_1, \ldots, L_{10}$, $W_1, \ldots, W_{10}$
- given: $V_{\rm dd}$, $C_{\rm L}$, $I_{\rm ref}$, common-mode voltage $V_{\rm cm}$
- we'll formulate as GP

Power, bandwidth, gain, & area

- power: $P = V_{\mathrm{dd}}(I_8 + I_5 + I_7 + I_{10})$ . . . posynomial
- bandwidth: $B = g_{\mathrm{m},2} g_{\mathrm{m},6} / (g_{\mathrm{m},4} C_{\mathrm{L}})$ . . . monomial
- area: $A = W_1 L_1 + \cdots + W_{10} L_{10}$ . . . posynomial
- gain: $G = \dfrac{g_{\mathrm{m},2} g_{\mathrm{m},6}}{g_{\mathrm{m},4}(g_{\mathrm{o},6} + g_{\mathrm{o},7})}$

  . . . G^{-1} is posynomial, so $G \geq G^{\mathrm{min}}$ can be written as $G^{-1} \leq 1/G^{\mathrm{min}}$

Dimension, matching, and current constraints

- limits on device sizes: $L_{\min} \leq L_i \leq L_{\max}$, $W_{\min} \leq W_i$, $i = 1, \ldots, 10$
- differential symmetry constraints (M_1, M_2 and M_3, M_4 matched):

$$\begin{array}{lll} W_1 = W_2, & L_1 = L_2, & I_1 = I_2, \\ W_3 = W_4, & L_3 = L_4, & I_3 = I_4, \end{array}$$

- length & gate overdrive voltage matched for current mirror pairs:

$$\begin{array}{llll} L_5 = L_8, & L_{10} = L_7, & L_3 = L_9, & L_4 = L_6 \\ V_{\text{gov},5} = V_{\text{gov},8}, & V_{\text{gov},10} = V_{\text{gov},7}, & V_{\text{gov},3} = V_{\text{gov},9}, & V_{\text{gov},4} = V_{\text{gov},6} \end{array}$$

- current relations:

$$I_1 = I_3 = I_5/2, \qquad I_8 = I_{\text{ref}}, \qquad I_6 = I_7, \qquad I_9 = I_{10}$$

Saturation constraints

- diode connected devices (M_3, M_4, M_8, M_{10}) automatically in saturation
- others must have $V_{\mathrm{ds}} \geq V_{\mathrm{dsat}}$:
  - M_7: $V_{\mathrm{dsat},7} \leq V_{\mathrm{cm}}$
  - M_6: $V_{\mathrm{dsat},6} + V_{\mathrm{cm}} \leq V_{\mathrm{dd}}$
  - M_9: $V_{\mathrm{dsat},9} + V_{\mathrm{gs},10} \leq V_{\mathrm{dd}}$
  - M_5: $V_{\mathrm{ds},5} + V_{\mathrm{gs},1} \leq V_{\mathrm{cm}}$
  - M_1 & M_2: $V_{\mathrm{cm}} + V_{\mathrm{gs},3} \leq V_{\mathrm{dd}} + V_{\mathrm{th}}$
- . . . all are posynomial inequalities

Node capacitances and non-dominant poles

- capacitances at nodes are posynomials, *e.g.*,

$$C^{\mathrm{out}} = C_{\mathrm{gd},6} + C_{\mathrm{db},6} + C_{\mathrm{gd},7} + C_{\mathrm{db},7} + C_{\mathrm{L}}$$

- non-dominant time constants are posynomials:

$$\tau_1 = \frac{C_{\mathrm{d1}}}{g_{\mathrm{m},3}}, \qquad \tau_2 = \frac{C_{\mathrm{d2}}}{g_{\mathrm{m},4}}, \qquad \tau_9 = \frac{C_{\mathrm{d9}}}{g_{\mathrm{m},10}}$$

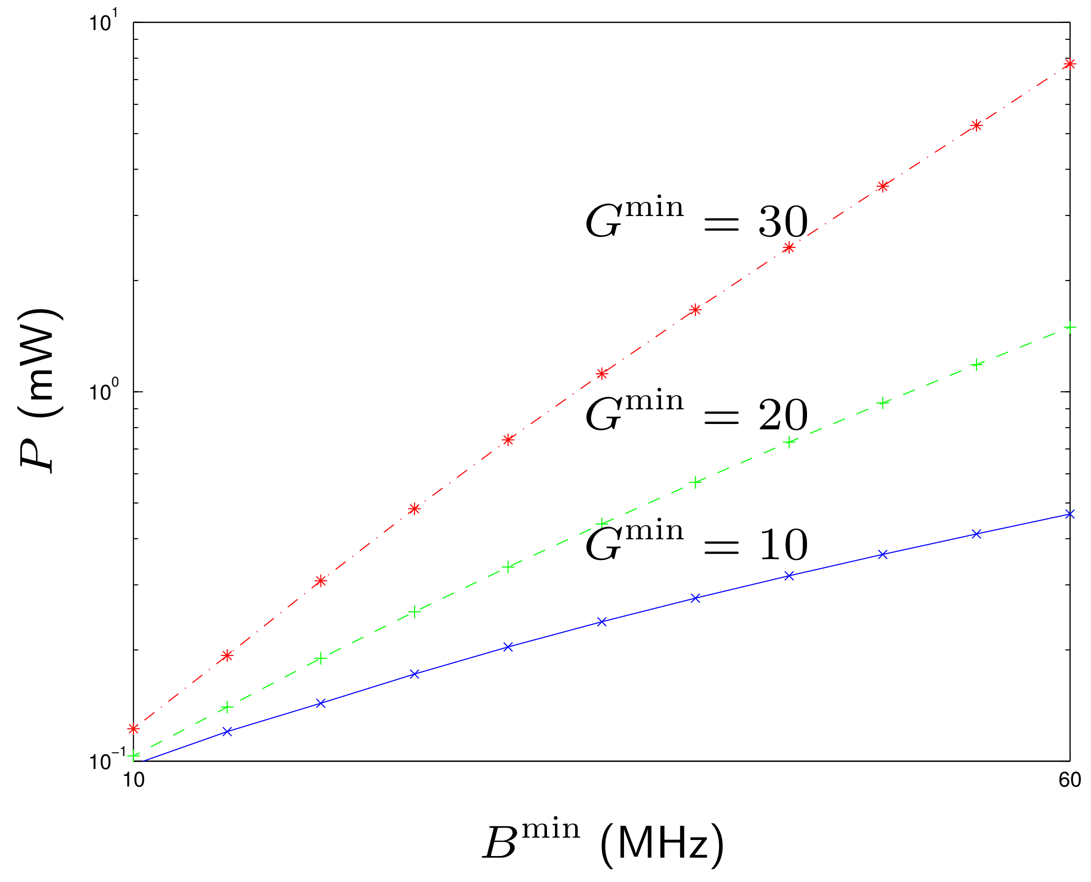
  ($C_{\mathrm{d1}}, C_{\mathrm{d2}}, C_{\mathrm{d9}}$ are node capacitances at drains of M_1, M_2, M_9)

- to limit effect of non-dominant poles, make sum smaller than dominant time constant:

$$\tau_1 + \tau_2 + \tau_9 \leq \tau_{\mathrm{dom}} = C_{\mathrm{L}}/g_{\mathrm{m}}$$

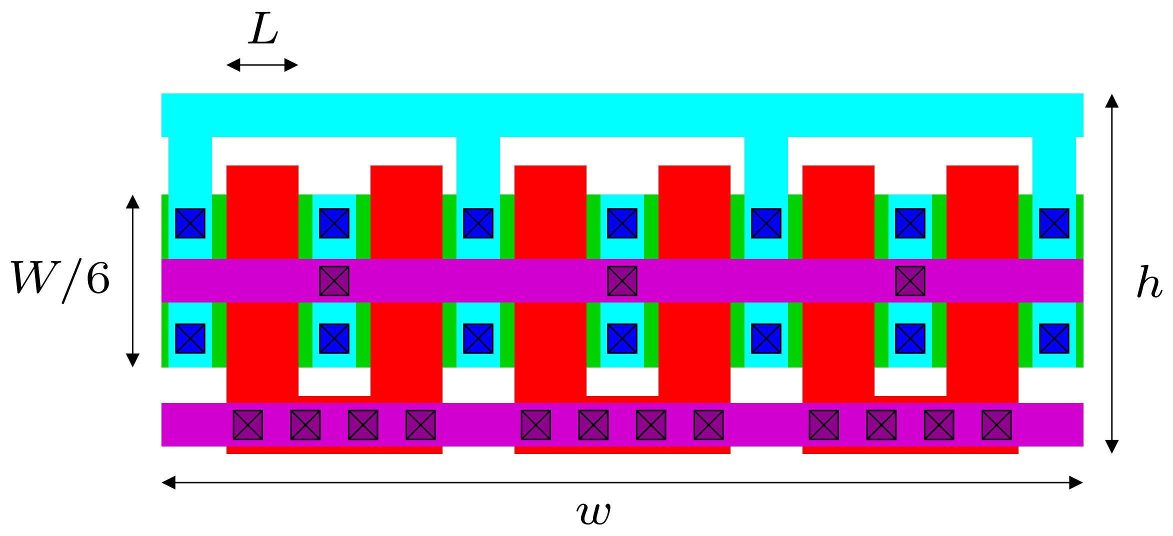
  . . . a posynomial constraint

Power versus bandwidth trade-off



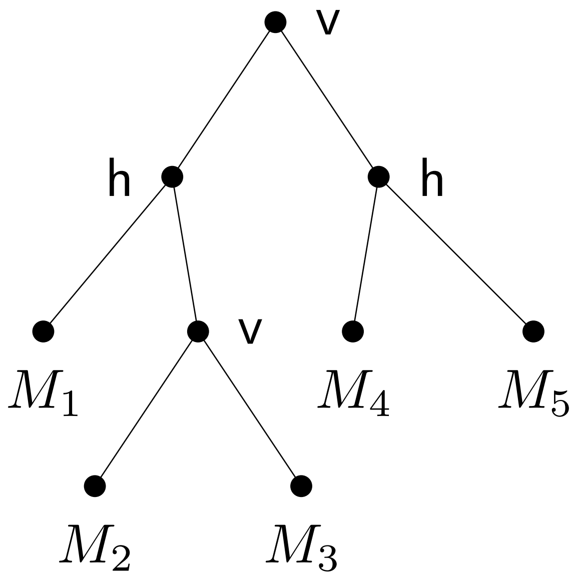
Joint electrical/physical design

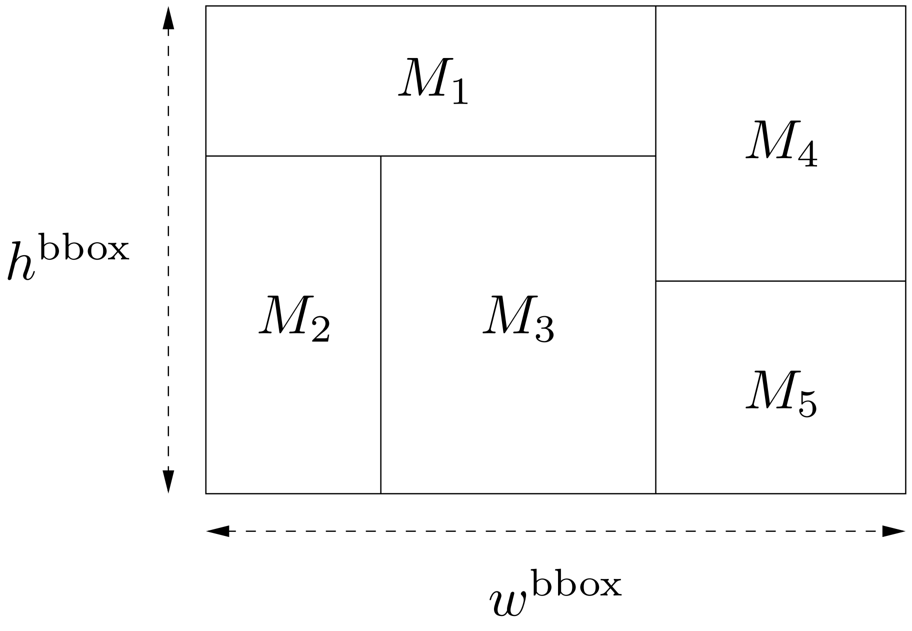
- each device has a (physical) cell width w and height h for floor planning
- devices are folded into multiple fingers
- (approximate) posynomial or monomial relations link electrical variables (I, L, W) and physical variables (w, h), *e.g.*,
  - cell area is at least $4\times$ active area: $wh \geq 4WL$
  - cell aspect ratio limited to 5:1: $1/5 \leq w/h \leq 5$



Slicing tree layout scheme

- vertical and horizontal slices fix relative placement of device cells
- leaves are device cells; root is bounding box





Slicing tree constraints

- introduce width, height for each node in slicing tree
- for each vertical slice with parent a and children b, c add constraints

$$w_a = w_b + w_c, \qquad h_a = \max\{h_b, h_c\}$$

- for each horizontal slice with parent a and children b, c add constraints

$$w_a = \max\{w_b, w_c\}, \qquad h_a = h_b + h_c$$

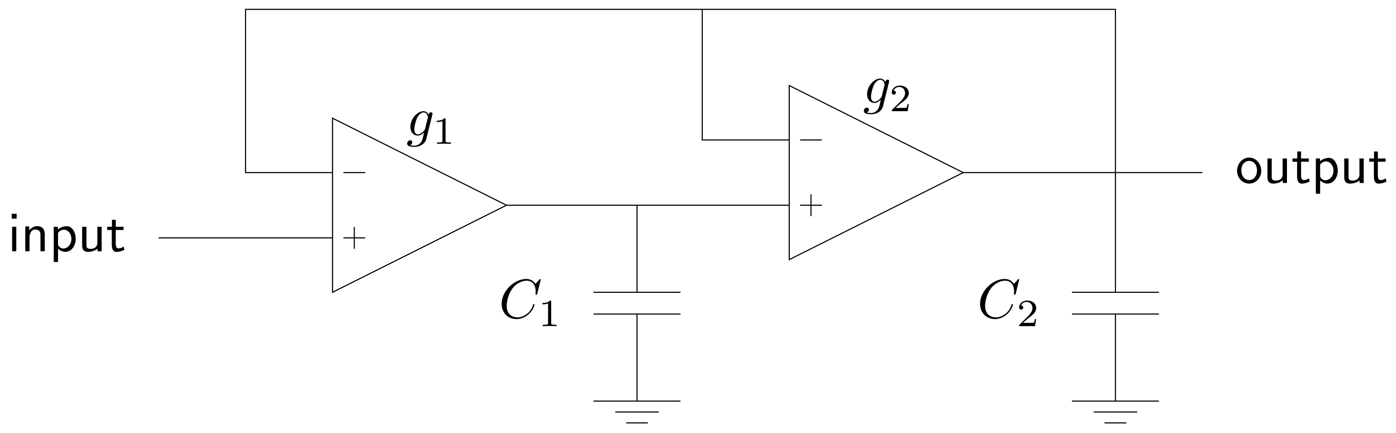
- shows width and height of bounding box and each node is generalized posynomial of device cell widths, heights
- resulting GP formulation is very sparse

Joint electrical/physical design via GP

- form **one** GP that includes
  - electrical variables, constraints $(I_i, L_i, W_i, g_{\mathrm{m},i} \ldots)$
  - physical variables, constraints $(w_i, h_i, w^{\mathrm{bbox}}, h^{\mathrm{bbox}}, \ldots)$
  - coupling constraints $(w_i h_i \geq 4 W_i L_i$, . . . $)$
- solve it all together
- extensions: can add
  - parasitic estimates
  - more accurate expressions for device cell dimensions
  - channels for routing

Optimal filter implementation

simple Gm-C two-pole lowpass filter



transfer function is

$$H(s) = \frac{1}{1 + t_1 s + t_1 t_2 s^2}, \qquad t_1 = C_1/g_1, \quad t_2 = C_2/g_2$$

g_i is amplifier transconductance

Noise analysis

- N_i is input referred (white) amplifier input-referred voltage density
- spectral density of output noise is

$$N(\omega)^2 = \frac{N_1^2 + \omega^2 N_2^2}{(1 - t_1 t_2 \omega^2)^2 + t_1^2 \omega^2}$$

- root-mean-square output noise voltage is

$$M = \left(\int_0^\infty N(\omega)^2 \, d\omega \right)^{1/2} = \left(\alpha N_1^2 + \beta N_2^2 \right)^{1/2}$$

Amplifier and capacitor implementation models

- each amplifier has **private variables** u ($e.g.$, device lengths & widths) and constraints

- transconductance g is monomial in u; area A^{amp}, power P, input-referred noise density N are posynomial in u

- each capacitor has private variables v ($e.g.$, physical dimensions) and constraints

- capacitance C is monomial in v; area A^{cap} is posynomial

- design variables are u_1, u_2, v_1, v_2

Optimal filter implementation problem

- filter is Butterworth with frequency ω_{c}:

$$t_1 = \sqrt{2}/\omega_c, \qquad t_2 = (1/\sqrt{2})/\omega_c$$

- minimize total power of implementation, subject to area, output noise limits:

$$\begin{array}{ll} \text{minimize} & P(u_1) + P(u_2) \\ \text{subject to} & t_1 = \sqrt{2}/\omega_c, \qquad t_2 = (1/\sqrt{2})/\omega_c \\ & A^{\mathrm{amp}}(u_1) + A^{\mathrm{amp}}(u_2) + A^{\mathrm{cap}}(v_1) + A^{\mathrm{cap}}(v_2) \leq A^{\mathrm{max}} \\ & M = (\omega_c/4\sqrt{2})(N_1^2 + 2N_2^2)^{1/2} \leq M^{\mathrm{max}} \end{array}$$

- a **GGP** in the variables u_1, u_2, v_1, v_2

Example

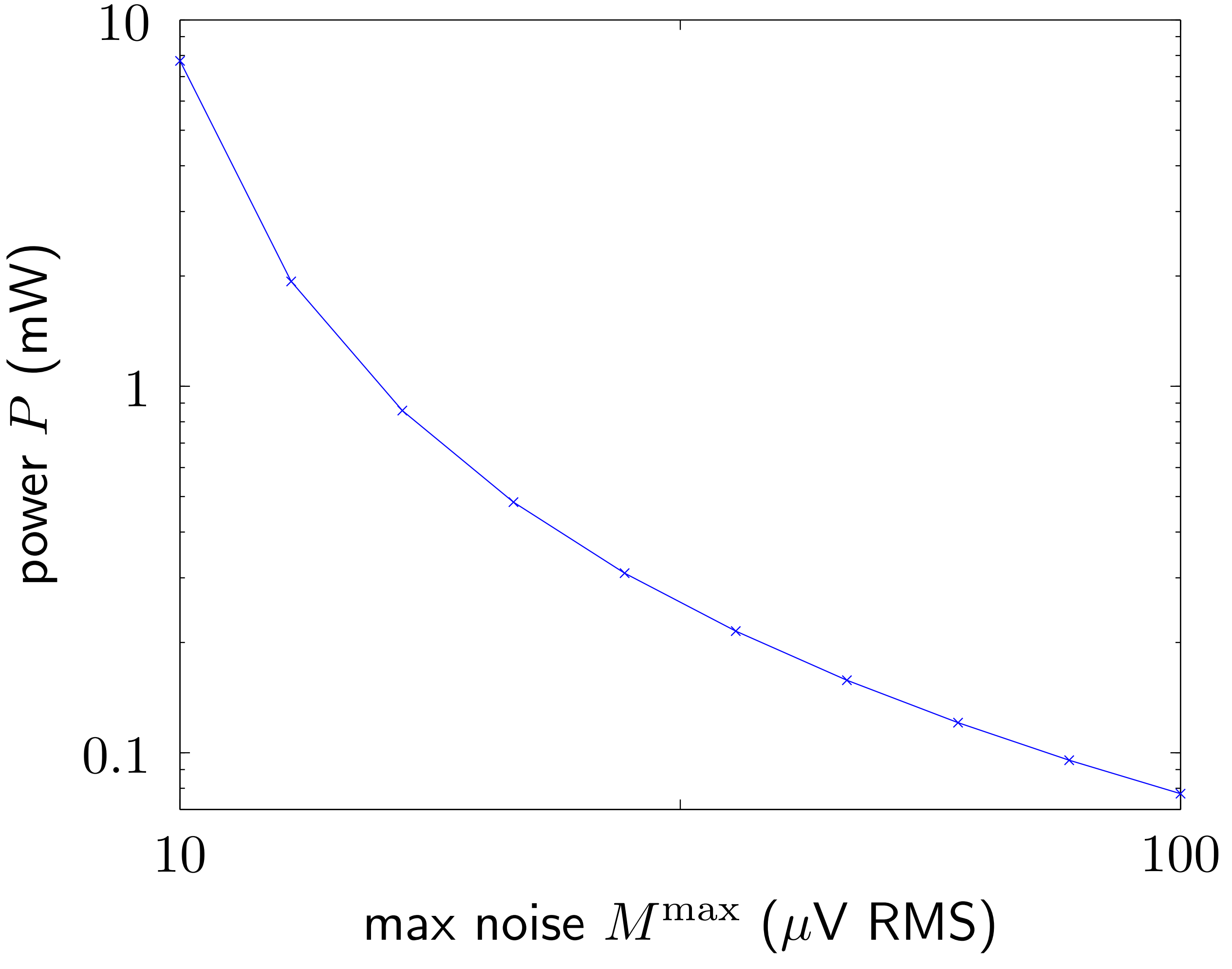
- Butterworth filter with $\omega_c = 10^8 \mathrm{rad/s}$
- private variables in amplifiers: (equivalent) L, W
- amplifier model:

$$A^{\mathrm{amp}} = WL, \qquad P = 2.5\cdot 10^{-4} W/L,$$
$$g = 4\cdot 10^{-5} W/L, \qquad N = \sqrt{7.5\cdot 10^{-16} L/W}$$

  (based on simple model with $V_{\mathrm{dd}} = 2.5$, $V_{\mathrm{gov}} = 0.2$)

- private variable in capacitors is area A^{cap}; $C = 10^{-4} A^{\mathrm{cap}}$
- $A^{\mathrm{max}} = 4\cdot 10^{-6}$

Power versus noise trade-off



Monomial and Posynomial Fitting

A basic property of posynomials

- if f is a monomial, then $\log f(e^y)$ is **affine** (linear plus constant)
- if f is a posynomial, then $\log f(e^y)$ is **convex**
- roughly speaking, a posynomial is convex when plotted on log-log plot
- midpoint rule for posynomial f:
  - let z be elementwise geometric mean of x, y, *i.e.*, $z_i = \sqrt{x_i y_i}$
  - then $f(z) \leq \sqrt{f(x)f(y)}$
- a converse: if $\log \phi(e^y)$ is convex, then ϕ can be approximated as well as you like by a posynomial

Convexity in circuit design context

- consider circuit with design variables $W_1, \ldots, W_n$ (say) & performance measure $\phi(W_1, \ldots, W_n)$ ($e.g.$, power, delay, area)

- two designs: $W_i^{(a)}$ & $W_i^{(b)}$, with performance $\phi^{(a)}$ & $\phi^{(b)}$

- form **geometric mean** compromise design with $W_i^{(c)} = \sqrt{W_i^{(a)} W_i^{(b)}}$, performance $\phi^{(c)}$

- if ϕ is generalized posynomial, then we have $\phi^{(c)} \leq \sqrt{\phi^{(a)} \phi^{(b)}}$

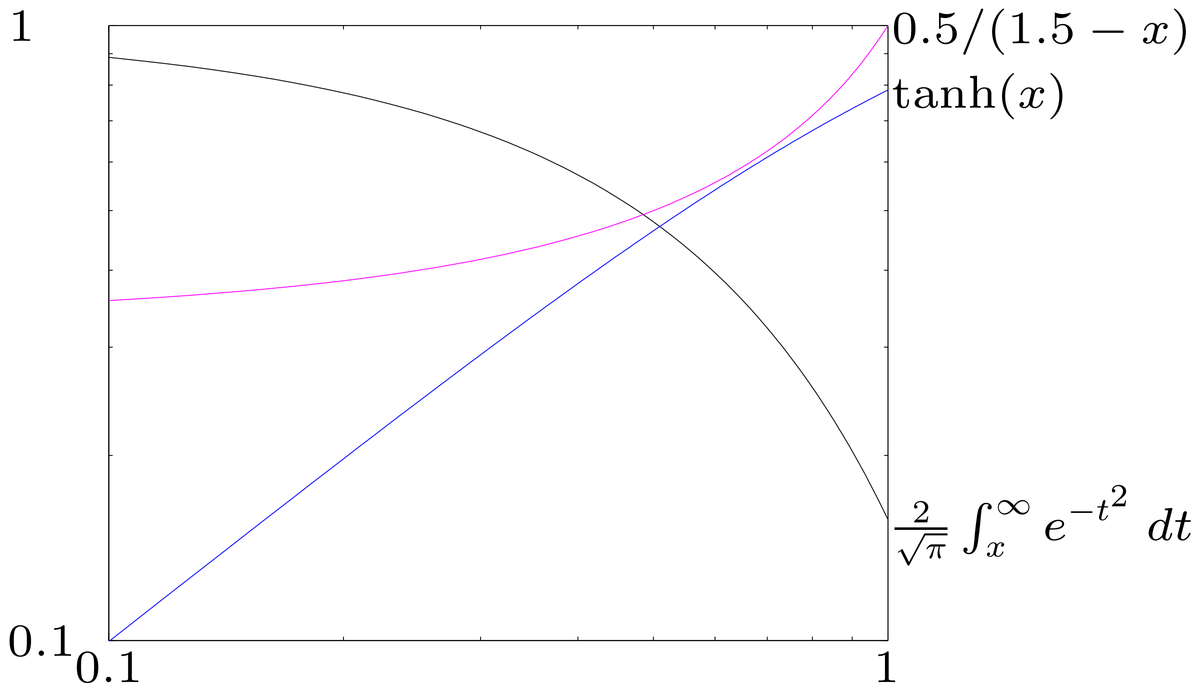
- this is **not obvious**

Monomial/posynomial approximation: Theory

when can a function f be approximated by a monomial or generalized posynomial?

- form function $F(y) = \log f(e^y)$
- f can be approximated by a monomial if and only if F is nearly affine (linear plus constant)
- f can be approximated by a generalized posynomial if and only if F is nearly convex

Examples



- $\tanh(x)$ can be reasonably well fit by a monomial
- $0.5/(1.5-x)$ can be fit by a generalized posynomial
- $(2/\sqrt{\pi})\int_x^\infty e^{-t^2}\,dt$ cannot be fit very well by a generalized posynomial

What problems can be approximated by GGPs?

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 1, \quad i = 1, \ldots, m \\ & g_i(x) = 1, \quad i = 1, \ldots, p \end{array}$$

- transformed objective and inequality constraint functions $F_i(y) = \log f_i(e^y)$ must be nearly convex
- transformed equality constraint functions $G_i(y) = \log G_i(e^y)$ must be nearly affine

Monomial fitting via log-regression

find coefficient $c > 0$ and exponents $a_1, \ldots, a_n$ of monomial f so that

$$f(x^{(i)}) \approx f^{(i)}, \qquad i = 1, \ldots, N$$

- rewrite as

$$\begin{aligned} \log f(x^{(i)}) &= \log c + a_1 \log x_1^{(i)} + \cdots + a_n \log x_n^{(i)} \\ &\approx \log f^{(i)}, \qquad i = 1, \ldots, N \end{aligned}$$

- use least-squares (regression) to find $\log c$, $a_1, \ldots, a_n$ that minimize

$$\sum_{i=1}^{N} \left(\log c + a_1 \log x_1^{(i)} + \cdots + a_n \log x_n^{(i)} - \log f^{(i)} \right)^2$$

Posynomial fitting via Gauss-Newton

find coefficients and exponents of posynomial f so that

$$f(x^{(i)}) \approx f^{(i)}, \qquad i = 1, \ldots, N$$

- minimize sum of squared fractional errors

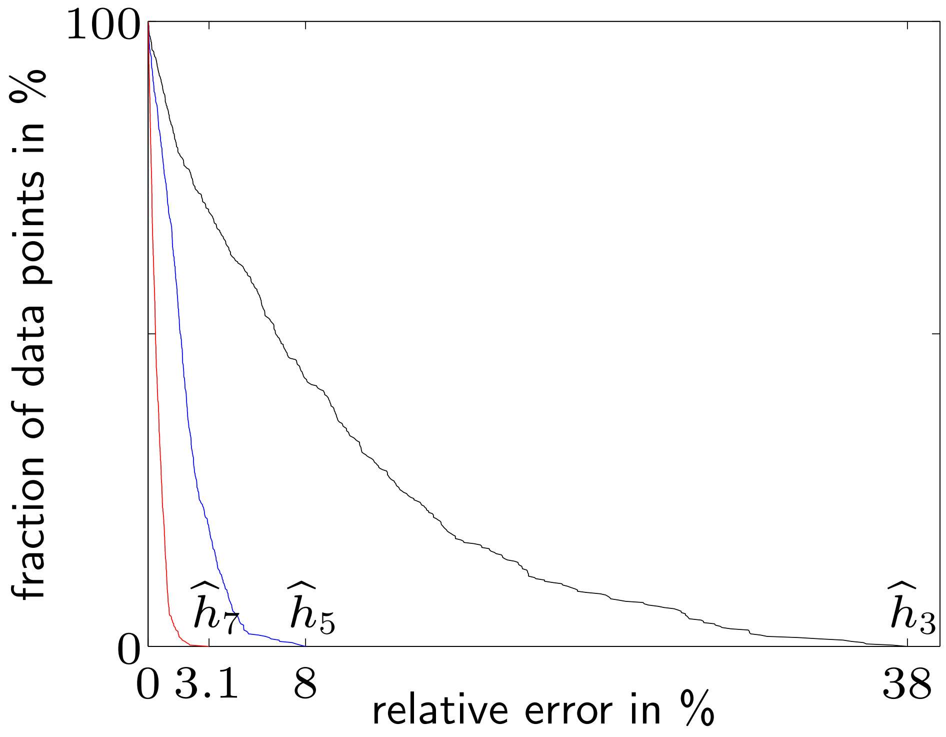
$$\sum_{i=1}^{N} \left(\frac{f^{(i)} - f(x^{(i)})}{f^{(i)}} \right)^2$$

  can be (locally) solved by Gauss-Newton method

- needs starting guess for coefficients, exponents

Posynomial fitting example

- 1000 data points from $f(x) = e^{(\log x_1)^2 + (\log x_2)^2}$ over $0.1 \leq x_i \leq 1$
- cumulative error distribution for 3-, 5-, and 7-term posynomial fits



A simple max-monomial fitting method

fit **max-monomial**

$$f(x) = \max_{k=1,\ldots,K} f_k(x)$$

($f_1, \ldots, f_k$ monomials) to data $x^{(i)}, f^{(i)}$, $i = 1, \ldots, N$

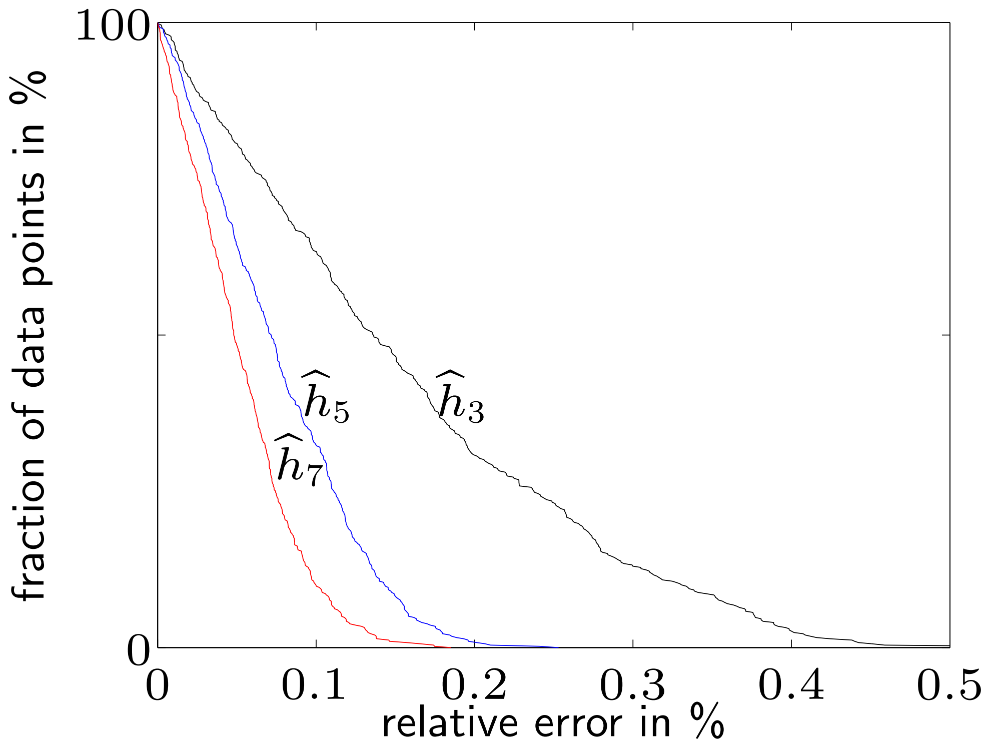
simple algorithm:

```
repeat
for $k = 1, \ldots, K$
1. find all data points $x^{(j)}$ for which $f_k(x^{(j)}) = f(x^{(j)})$
   ($i.e.$, data points at which $f_k$ is the largest of the monomials)

2. update $f_k$ by carrying out monomial fit to these data
```

Max-monomial fitting example

- same 1000 data points as previous example
- cumulative error distribution for 3-, 5-, and 7-term max-monomial fits



Software & Modeling Systems

GP solvers

GP solvers (primal-dual, interior-point, exploit sparsity):

- MOSEK: `www.mosek.com`
  (commercial; C with Matlab interface)
- GPCVX, GPPOSY: `www.stanford.edu/~boyd/ggplab/`
  (open source; Matlab)
- CVXOPT: `www.ee.ucla.edu/~vandenbe/cvxopt/`
  (open source; Python/C)

GP/GGP modeling systems

- allow simple specification of GPs and GGPs in natural form
  - declare optimization variables
  - form monomial, posynomial, generalized posynomial expressions
  - specify objective and constraints
- automatically transform to standard GP, call solver, transform back
- built using object-oriented methods and/or compiler-compilers

Example (ggplab)

```
gpvar x y z                 % create three scalar GP variables
m1 = 3.4*x^-0.33/z          % form a monomial
p1 = z*sqrt(m1)+0.1/m1      % form a posynomial
gp1 = max(1,x+y,p1)         % form a generalized posynomial

% form an array of constraints
constrs = [ m1==x, p1<=m1, 1<=y, gp1+p1<=5/y ]

% solve generalized GP
[obj_value, solution, status] = gpsolve(x+y+z,constrs)
```

Current GP/GGP modeling systems

- YALMIP: `control.ee.ethz.ch/~joloef/yalmip.msql`
  - Matlab; supports multiple solvers
  - part of much larger optimization modeling system
- GGPLAB: `www.stanford.edu/~boyd/ggplab/`
  - open source; Matlab
  - simple system for GP/GGP only; meant for tutorial purposes
- CVX: `www.stanford.edu/~boyd/cvx/`
  - open source; Matlab/C
  - part of larger convex optimization modeling system

Conclusions

Conclusions

(generalized) geometric programming

- comes up in a variety of circuit sizing contexts
- can be used to formulate a variety of problems
- admits fast, reliable solution of large-scale problems
- is good at concurrently balancing lots of coupled constraints and objectives
- is useful even when problem has discrete constraints

Approach

- most problems don't come naturally in GP form; be prepared to reformulate and/or approximate
- GP modeling is not a "try my software" method; it requires thinking
- our approach:
  - start with simple analytical models (RC, square-law, Pelgrom, . . .) to verify GP might apply
  - then fit GP-compatible models to simulation or measured data
  - for highest accuracy, revert to local method for final polishing

References

- *A tutorial on geometric programming*
  (Optimization and Engineering 2006)

- *Digital circuit sizing via geometric programming*
  (Operations Research 2005)

- *Convex optimization*, Cambridge Univ. Press 2004

(these include hundreds of references)

available at `www.stanford.edu/~boyd/research.html`