Geometric Programming for Circuit Design

Stephen Boyd

Seung Jean Kim
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

\[ x = (x_1, \ldots, x_n) : \text{vector of positive optimization variables} \]

• function \( g \) of form

\[ g(x) = cx_1^{\alpha_1}x_2^{\alpha_2} \cdots x_n^{\alpha_n}, \]

with \( c > 0, \alpha_i \in \mathbb{R} \), is called monomial

• sum of monomials, \( i.e. \), function \( f \) of form

\[ f(x) = \sum_{k=1}^{t} c_kx_1^{\alpha_{1k}}x_2^{\alpha_{2k}} \cdots x_n^{\alpha_{nk}}, \]

with \( c_k > 0, \alpha_{ik} \in \mathbb{R} \), 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:**

- \( \max \{1 + x_1, 2x_1 + x_2^{0.2}x_3^{-3.9}\} \)

- \( (0.1x_1x_3^{-0.5} + x_2^{1.7}x_3^{0.7})^{1.5} \)

- \( (\max \{1 + x_1, 2x_1 + x_2^{0.2}x_3^{-3.9}\})^{1.7} + x_2^{1.1}x_3^{3.7} \)
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)

\[
\begin{align*}
\text{minimize} & \quad f_0(x) \\
\text{subject to} & \quad f_i(x) \leq 1, \quad i = 1, \ldots, m \\
& \quad g_i(x) = 1, \quad i = 1, \ldots, p
\end{align*}
\]

\( f_i \) 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 \( h/w \), \( d/w \)

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

in standard GP form:

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

EPFL 6/16/06
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

\[
\begin{align*}
\text{minimize} & \quad hw \\
\text{subject to} & \quad h_iw_i = A_i, \quad 1/\alpha_{\text{max}} \leq h_i/w_i \leq \alpha_{\text{max}}, \\
& \quad \max\{h_1, h_2\} + \max\{h_3, h_4\} \leq h, \\
& \quad \max\{w_1 + w_2, w_3 + w_4\} \leq w
\end{align*}
\]

... a GGP
Trade-off analysis

(no equality constraints, for simplicity)

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

\[
\begin{align*}
\text{minimize} & \quad f_0(x) \\
\text{subject to} & \quad f_i(x) \leq u_i, \quad i = 1, \ldots, m
\end{align*}
\]

• \(u_i > 1\) (\(u_i < 1\)) means \(i\)th 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. $A_{flr}$, for $A_{wall} = 100, 1000, 10000$

• $h/w, d/w$ aspect ratio limits 0.5, 2
Sensitivity analysis

• optimal sensitivity of $i$th constraint is

$$S_i = \frac{\partial p/p}{\partial u_i/u_i} \bigg|_{u=1}$$

• $S_i$ predicts fractional change in optimal objective value if $i$th 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)

\[
\begin{align*}
    \text{minimize} & \quad D(x) \\
    \text{subject to} & \quad P(x) \leq P^{\text{max}}, \quad A(x) \leq A^{\text{max}}
\end{align*}
\]

• both constraints tight at optimal \( x^* \): \( P(x^*) = P^{\text{max}}, A(x^*) = A^{\text{max}} \)

• suppose optimal sensitivities are \( S^{\text{pwr}} = -2.1, S^{\text{area}} = -0.3 \)

• 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

\[
\begin{align*}
f(x) + t & \leq 1, \\
g(x)/t & \leq 1, \\
h(x)/t & \leq 1
\end{align*}
\]

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

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

- $\log f_i(e^y)$ are (smooth) \textbf{convex} functions

- $\log g_i(e^y)$ are affine functions, \textit{i.e.}, linear plus a constant

- solve (nonlinear) \textbf{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 variables, 10000 monomial terms: few seconds
  - sparse 10000 variables, 100000 monomial terms: minute
  - sparse $10^6$ variables, $10^7$ 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

\[
\begin{align*}
&\text{minimize} & f_0(x) \\
&\text{subject to} & f_i(x) \leq 1, & i = 1, \ldots, m \\
& & g_i(x) = 1, & i = 1, \ldots, p \\
& & x_i \in D_i, & i = 1, \ldots, k
\end{align*}
\]

- \(f_i\) are generalized posynomials, \(g_i\) are monomials
- \(D_i\) 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

\[
C_{i}^{\text{in}} = \bar{C}_{i}^{\text{in}} x_i, \quad C_{i}^{\text{int}} = \bar{C}_{i}^{\text{int}} x_i, \quad R_{i} = \bar{R}_{i} / x_i, \quad C_{i}^{L} = \sum_{j \in \text{FO}(i)} C_{j}^{\text{in}}
\]
RC gate model

- RC gate delay:

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

- \( D_i \) are posynomials (of scale factors)
Path and circuit delay

- delay of a path: sum of delays of gates on path
  \[ \ldots \text{posynomial} \]

- circuit delay: maximum delay over all paths
  \[ \ldots \text{generalized posynomial} \]
Area & power

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

- total power is \( P = P_{\text{dyn}} + P_{\text{stat}} \)
  
  - dynamic power \( P_{\text{dyn}} = \sum_{i=1}^{n} f_i (C_i^L + C_i^{\text{int}}) V_{dd}^2 \)
    
    \( f_i \) is gate switching frequency
  
  - static power \( P_{\text{stat}} = \sum_{i=1}^{n} x_i \bar{I}_i^{\text{leak}} V_{dd} \)
    
    \( \bar{I}_i^{\text{leak}} \) 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

minimize \( D \)
subject to \( P \leq P^{\text{max}}, \ A \leq A^{\text{max}} \)
\( 1 \leq x_i, \ i = 1, \ldots, n \)

... 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

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

\[
\begin{align*}
\text{maximize} & \quad f_{\text{clk}} \\
\text{subject to} & \quad P \leq P^{\max}, \quad A \leq A^{\max}, \quad (1/0.8)Df_{\text{clk}} \leq 1,
& \quad 1 \leq x_i, \quad i = 1, \ldots, n
\end{align*}
\]

... 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:

<table>
<thead>
<tr>
<th>gate type</th>
<th>$C^{\text{in}}$</th>
<th>$C^{\text{int}}$</th>
<th>$\bar{R}$</th>
<th>$\bar{A}$</th>
<th>$\bar{I}^{\text{leak}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>INV</td>
<td>3</td>
<td>3</td>
<td>0.48</td>
<td>3</td>
<td>0.006</td>
</tr>
<tr>
<td>NAND2</td>
<td>4</td>
<td>6</td>
<td>0.48</td>
<td>8</td>
<td>0.007</td>
</tr>
<tr>
<td>NOR2</td>
<td>5</td>
<td>6</td>
<td>0.48</td>
<td>10</td>
<td>0.009</td>
</tr>
<tr>
<td>AOI21</td>
<td>6</td>
<td>7</td>
<td>0.48</td>
<td>17</td>
<td>0.003</td>
</tr>
<tr>
<td>OAI21</td>
<td>6</td>
<td>7</td>
<td>0.48</td>
<td>16</td>
<td>0.003</td>
</tr>
</tbody>
</table>

- time unit is $\tau$, 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 performance depends on random device and process parameters
- hence, performance measures like $P$, $D$ are random variables $P$, $D$
- delay $D$ is max of many random variables; often skewed to right
- distributions of $P$, $D$ depend on gate scalings $x_i$

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

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

\[
\begin{align*}
\text{minimize} & \quad Q^{.95}(D) \\
\text{subject to} & \quad Q^{.95}(P) \leq P^{\text{max}}, \quad A \leq A^{\text{max}} \\
& \quad 1 \leq x_i, \quad i = 1, \ldots, n
\end{align*}
\]

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

• but, (GP-compatible) heuristic method works well
**Statistical model**

- for simplicity consider $V_{th}$ variation only

- Pelgrom’s model: $\sigma_{V_{th}} = \bar{\sigma}_{V_{th}} x^{-1/2}$

- alpha-power law model: $D \propto V_{dd}/(V_{dd} - V_{th})^\alpha$, with $\alpha \approx 1.3$

- for small variation in $V_{th}$,

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

- $\sigma_D$ is posynomial

- get similar (posynomial) models for $\sigma_D$ 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 ($\kappa_i$ is typically 2 or 3)

- verify statistical performance via Monte Carlo analysis (can update $\kappa_i$’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

nominal optimal design

statistical design
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 $V_{dd}$, $V_{th}$:

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

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

- generalized posynomial approximation

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

error under 1% for $V_{dd,i} \geq 2V_{th,i}$, $1.3 \leq \alpha \leq 2$
Generalized posynomial power model

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

- simple static power model:

\[
P_{\text{stat}} = \sum_{i=1}^{n} x_i I_{\text{leak}} V_{dd,i}, \quad I_{\text{leak}} \propto e^{-(V_{th,i} - \gamma V_{dd,i})/V_0}
\]

\( \gamma, V_0 \) are (process) constants

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

- but, total power \( P = P_{\text{dyn}} + P_{\text{stat}} \) **can** be approximated well by a generalized posynomial
Generalized posynomial power model example

Total power \( P = V_{dd}^2 + 30V_{dd}e^{-\left(V_{th}-0.06V_{dd}\right)/0.039} \) (up to scaling)

- Generalized posynomial approximation
  \( \hat{P} = V_{dd}^2 + 0.06V_{dd}(1 + 0.0031V_{dd})^{500}(V_{th}/0.039)^{-6.16} \)

- Error under 3% (well under accuracy of model!)
Joint optimization of gate sizes, $V_{dd}$, & $V_{th}$

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

minimize $D$

subject to $P \leq P_{max}$, $A \leq A_{max}$

$V_{th_{min}} \leq V_{th,i} \leq V_{th_{max}}$, $i = 1, \ldots, n$

$V_{dd_{min}} \leq V_{dd,i} \leq V_{dd_{max}}$, $i = 1, \ldots, n$

other constraints . . .

(. . . a GGP)

discrete allowed $V_{dd}$, $V_{th}$ values yields MIGP
Extensions/variations

• clustering, with single $V_{dd}$, $V_{th}$ per cluster:

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

... monomial (equality) constraints

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

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

... monomial (inequality) constraints

• multimode design: choose single set of gate scalings, different $V_{dd}^{(k)}$, $V_{th}^{(k)}$ 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 $V_{dd,i}$, threshold voltages $V_{th,i}$

- four delay-power trade-off curves:
  - fixed $V_{dd,i} = 1.0$, fixed $V_{th,i} = 0.3$
  - fixed $V_{dd,i} = 1.0$, variable $V_{th,i} \in \{0.2, 0.3, 0.4\}$
  - CVS with $V_{dd,i} \in \{0.6, 1.0\}$, $V_{th,i} \in \{0.2, 0.3, 0.4\}$
  - variable continuous $V_{dd}$, $V_{th}$, $0.6 \leq V_{dd,i} \leq 1.0$, $0.2 \leq V_{th,i} \leq 0.4$
    (not practical, but serves as lower bound)
Trade-off curve analysis

- Fixed $V_{dd}$, $V_{th}$
- Fixed $V_{dd}$, variable $V_{th}$

$P$ vs $D_{max}$

CVS lower bound
Design with multiple threshold voltages

\[
\begin{align*}
V_{th} = 0.4 & \quad 100\% \\
V_{th} = 0.3 & \quad 35\% \\
V_{th} = 0.2 & \quad 0\%
\end{align*}
\]
Clustered voltage scaling

% of gates

V_{dd} = 0.6

V_{dd} = 1.0

EPFL 6/16/06
Analog Circuit Design Applications
Large signal MOS model

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

- transconductance $g_m = (2\mu C_{ox})^{1/2} I^{1/2} L^{-1/2} W^{1/2}$ is monomial
- output conductance $g_o = \lambda I$ 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
  - $0.3\mu m \leq L \leq 3\mu m$, $2\mu m \leq W \leq 20\mu m$
  - $0.7V \leq V_{gs} \leq 1.7V$, $V_{dsat} \leq V_{ds} \leq 1.5V_{gs}$

- $V_{ds}$ appears in data set, but not in $g_m$ model

- monomial fit (using simple log-regression, SI units):
  \[
g_m = 0.0278I^{0.4798}L^{-0.511}W^{0.5632}
  \]
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: $V_{dsat} + IR \leq V_{dd}$
- gain $G = g_m/(1/R + g_o)$
- power $P = V_{dd}I$
- (unity gain) bandwidth $B = g_m/C_L$
- design problem:

  minimize $P$

  subject to $B \geq B^{\text{min}}$, $G \geq G^{\text{min}}$

  saturation
Common source amplifier design via GP

- rewrite as

\[
\begin{align*}
\text{minimize} & \quad P \\
\text{subject to} & \quad B^{-1} \leq 1/B^\text{min}, \quad G^{-1} \leq 1/G^\text{min} \\
& \quad V_{\text{dsat}} + IR \leq V_{\text{dd}} \\
\end{align*}
\]

- ... a GP, since \( P \) and \( B \) are monomials, and

\[
G^{-1} = \frac{1/R + g_o}{g_m}
\]

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

minimize \( P \)
subject to \( B \geq B_{\text{min}}, \quad G \geq G_{\text{min}}, \quad A \leq A_{\text{max}} \)
other constraints . . .

• 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: \( V_{\text{dd}}, C_L, I_{\text{ref}}, \) common-mode voltage \( V_{\text{cm}} \)

• we’ll formulate as GP
Power, bandwidth, gain, & area

• power: \( P = V_{dd}(I_8 + I_5 + I_7 + I_{10}) \) \ldots \) posynomial

• bandwidth: \( B = \frac{g_{m,2}g_{m,6}}{(g_{m,4}C_L)} \) \ldots \) monomial

• area: \( A = W_1L_1 + \cdots + W_{10}L_{10} \) \ldots \) posynomial

• gain: \( G = \frac{g_{m,2}g_{m,6}}{g_{m,4}(g_{o,6} + g_{o,7})} \)

\ldots \) \( G^{-1} \) is posynomial, so \( G \geq G^{\text{min}} \) can be written as \( G^{-1} \leq 1/G^{\text{min}} \)
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 \((M_1, M_2 \text{ and } M_3, M_4 \text{ matched})\):

\[
W_1 = W_2, \quad L_1 = L_2, \quad I_1 = I_2,
W_3 = W_4, \quad L_3 = L_4, \quad I_3 = I_4,
\]

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

\[
L_5 = L_8, \quad L_{10} = L_7, \quad L_3 = L_9, \quad L_4 = L_6
\]
\[
V_{gov,5} = V_{gov,8}, \quad V_{gov,10} = V_{gov,7}, \quad V_{gov,3} = V_{gov,9}, \quad V_{gov,4} = V_{gov,6}
\]

- current relations:

\[
I_1 = I_3 = I_5/2, \quad I_8 = I_{\text{ref}}, \quad I_6 = I_7, \quad I_9 = I_{10}
\]
Saturation constraints

• diode connected devices ($M_3, M_4, M_8, M_{10}$) automatically in saturation

• others must have $V_{ds} \geq V_{dsat}$:
  - $M_7$: $V_{dsat,7} \leq V_{cm}$
  - $M_6$: $V_{dsat,6} + V_{cm} \leq V_{dd}$
  - $M_9$: $V_{dsat,9} + V_{gs,10} \leq V_{dd}$
  - $M_5$: $V_{ds,5} + V_{gs,1} \leq V_{cm}$
  - $M_1$ & $M_2$: $V_{cm} + V_{gs,3} \leq V_{dd} + V_{th}$

• . . . all are posynomial inequalities
Node capacitances and non-dominant poles

- Capacitances at nodes are posynomials, e.g.,

\[ C_{\text{out}} = C_{\text{gd},6} + C_{\text{db},6} + C_{\text{gd},7} + C_{\text{db},7} + C_L \]

- Non-dominant time constants are posynomials:

\[ \tau_1 = \frac{C_{\text{d1}}}{g_{m,3}}, \quad \tau_2 = \frac{C_{\text{d2}}}{g_{m,4}}, \quad \tau_9 = \frac{C_{\text{d9}}}{g_{m,10}} \]

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

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

\[ \tau_1 + \tau_2 + \tau_9 \leq \tau_{\text{dom}} = \frac{C_L}{g_m} \]

... a posynomial constraint
Power versus bandwidth trade-off

\[ P \text{ (mW)} \]

\[ B^{\text{min}} \text{ (MHz)} \]

\[ G_{\text{min}} = 10 \]
\[ G_{\text{min}} = 20 \]
\[ G_{\text{min}} = 30 \]
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

\[
\begin{align*}
    w_a &= w_b + w_c, & h_a &= \max\{h_b, h_c\}
\end{align*}
\]

• for each horizontal slice with parent \( a \) and children \( b, c \) add constraints

\[
\begin{align*}
    w_a &= \max\{w_b, w_c\}, & h_a &= h_b + h_c
\end{align*}
\]

• 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 \((I_i, L_i, W_i, g_{m,i} \ldots)\)
  – physical variables, constraints \((w_i, h_i, w_{bbox}, h_{bbox}, \ldots)\)
  – coupling constraints \((w_i h_i \geq 4W_i L_i, \ldots)\)

• 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

The transfer function is

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

\( g_i \) 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

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

- design variables are $u_1, u_2, v_1, v_2$
Optimal filter implementation problem

• filter is Butterworth with frequency \( \omega_c \):

\[
t_1 = \frac{\sqrt{2}}{\omega_c}, \quad t_2 = \frac{1}{\sqrt{2}}/\omega_c
\]

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

\[
\text{minimize} \quad P(u_1) + P(u_2)
\]

subject to

\[
t_1 = \frac{\sqrt{2}}{\omega_c}, \quad t_2 = \frac{1}{\sqrt{2}}/\omega_c
\]

\[
A^{\text{amp}}(u_1) + A^{\text{amp}}(u_2) + A^{\text{cap}}(v_1) + A^{\text{cap}}(v_2) \leq A^{\text{max}}
\]

\[
M = \left(\frac{\omega_c}{4\sqrt{2}}\right)(N_1^2 + 2N_2^2)^{1/2} \leq M^{\text{max}}
\]

• a **GGP** in the variables \( u_1, u_2, v_1, v_2 \)
Example

- Butterworth filter with $\omega_c = 10^8 \text{rad/s}$

- private variables in amplifiers: (equivalent) $L, W$

- amplifier model:
  \[
  A^{\text{amp}} = WL, \quad P = 2.5 \cdot 10^{-4} W/L, \\
  g = 4 \cdot 10^{-5} W/L, \quad N = \sqrt{7.5 \cdot 10^{-16} L/W}
  \]
  (based on simple model with $V_{dd} = 2.5$, $V_{gov} = 0.2$)

- private variable in capacitors is area $A^{\text{cap}}$; $C = 10^{-4} A^{\text{cap}}$

- $A^{\text{max}} = 4 \cdot 10^{-6}$
Power versus noise trade-off

![Graph showing the relationship between power $P$ (mW) and maximum noise $M_{\text{max}}$ ($\mu$V RMS). The graph indicates a trade-off where higher power results in lower maximum noise.](image)
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 \( \phi \) 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 $\phi$ 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
- \( \frac{0.5}{(1.5 - x)} \) can be fit by a generalized posynomial
- \( \frac{2}{\sqrt{\pi}} \int_{x}^{\infty} e^{-t^2} dt \) cannot be fit very well by a generalized posynomial
What problems can be approximated by GGPs?

minimize \( f_0(x) \)
subject to \( f_i(x) \leq 1, \quad i = 1, \ldots, m \)
\( g_i(x) = 1, \quad i = 1, \ldots, p \)

- 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

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

• rewrite as

$$\log f(x^{(i)}) = \log c + a_1 \log x_1^{(i)} + \cdots + a_n \log x_n^{(i)}$$

$$\approx \log f^{(i)}, \quad i = 1, \ldots, N$$

• 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

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

• 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](http://www.mosek.com)  
  (commercial; C with Matlab interface)

- **GPCVX, GPPOSY**: [www.stanford.edu/~boyd/ggplab/](http://www.stanford.edu/~boyd/ggplab/)  
  (open source; Matlab)

- **CVXOPT**: [www.ee.ucla.edu/~vandenbe/cvxopt/](http://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](control.ee.ethz.ch/~joloef/yalmip.msql)
  - Matlab; supports multiple solvers
  - Part of much larger optimization modeling system

- **GGPLAB**: [www.stanford.edu/~boyd/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/](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](http://www.stanford.edu/~boyd/research.html)