GPCAD: A Tool for CMOS Op-Amp Synthesis

Maria del Mar Hershenson, Stephen P. Boyd, Thomas H. Lee
Electrical Engineering Department, Stanford University, Stanford CA 94305
marita@smirc.stanford.edu, boyd@ee.stanford.edu, tomlee@smirc.stanford.edu

To appear in International Conference on Computer-Aided Design, November 1998

Abstract

We present a method for optimizing and automating component and transistor sizing for CMOS operational amplifiers. We observe that a wide variety of performance measures can be formulated as *posynomial* functions of the design variables. As a result, amplifier design problems can be formulated as a *geometric program*, a special type of convex optimization problem for which very efficient *global* optimization methods have recently been developed. The synthesis method is therefore fast, and determines the globally optimal design; in particular the final solution is completely independent of the starting point (which can even be infeasible), and infeasible specifications are unambiguously detected.

After briefly introducing the method, which is described in more detail in [1], we show how the method can be applied to six common op-amp architectures, and give several example designs.

1 Introduction

As the demand for mixed mode integrated circuits increases, the design of analog circuits such as operational amplifiers (op-amps) in CMOS technology becomes more critical. Many authors have noted the disproportionately large design time devoted to the analog circuitry in mixed mode integrated circuits. In [1] we introduced a new method for determining the component values and transistor dimensions for CMOS op-amps. The method handles a wide variety of specifications and constraints, is *extremely fast*, and results in *globally optimal* designs. We have developed a simple op-amp synthesis tool, called GPCAD, based on our method.

We have formulated the op-amp design problem as a special type of convex optimization problem called a *geometric program* (GP). Methods to solve convex optimization problems have several advantages when compared to general purpose optimization methods: they find the globally optimal solution; the solution can be computed extremely fast even for large problems; and, if a solution exists, convergence is guaranteed. The disadvantage of convex optimization methods is that they apply to a more restricted class of problems than the general methods. The contribution of this paper is to demonstrate the surprising result that a wide variety of op-amp design problems can be formulated with considerable accuracy as convex programming problems.

In this paper we describe how the method applies to six different types of op-amps: two simple op-amps (OTAs), a two-stage op-amp, a two-stage cascoded op-amp, a folded-cascode op-amp, and a telescopic op-amp. We also give some design examples.

In §2, we give a brief overview of previous approaches to op-amp synthesis. In §3, we describe geometric programming, the optimization problem which is the basis of the method. In §4, we briefly describe the transistor model used, which we call the GP1 model. This simple model gives reasonable agreement with sophisticated BSIM1 models over a range of lengths, widths, and bias currents, and moreover, is compatible with the GP method. In §5, we describe the six op-amp architectures we consider. In §6, we show how a variety of performance measures can be cast in the GP framework. To simplify the discussion (and also due to lack of space) we concentrate our discussion on a single typical op-amp, and use a simple transistor model based on a classical long channel square law. The same ideas and methods, however, can be used to formulate the GP for the other five architectures we consider, and the more accurate GP1 MOS model. In §7, we give design examples for the different op-amps. More details on geometric programming, transistor models and how the method applies, can be found in [1, 2].

2 Other approaches

We can classify previous methods for analog circuit CAD into four groups.

Classical optimization methods

Classical optimization methods, such as steepest descent, sequential quadratic programming and Lagrange multiplier methods, have been widely used in analog circuit CAD. The general purpose optimization codes NPSOL [3] and MINOS are used in, *e.g.*, [4, 5]. Other CAD methods based on classical optimization methods, and extensions such as a minimax formulation, include OPASYN [6], OAC [7], and STAIC [8]. These classical methods can be used with complicated circuit models, including full SPICE simulations in each iteration, as in DELIGHT.SPICE [9].

The main disadvantage of classical methods is that they only find *locally optimal* designs. This means that it is possible that some other set of design parameters, far away from the one found, results in a better design. The same problem arises in determining feasibility: they can fail to find a feasible design, even if one exists. In order to avoid local solutions, the minimization method is carried out from many different initial designs. This increases the likelihood of finding the globally optimal design but it also destroys one of the advantages of classical methods, *i.e.*, speed, since the computation effort is multiplied by the number of different initials designs that are tried. It also requires human intervention (to give "good" initial designs), which makes the method less automated. Also, these methods become slow if complex models are used, as in DELIGHT.SPICE.

Knowledge-based methods

Knowledge-based and expert-systems methods have also been widely used in analog circuit CAD. Examples include genetic algorithms or evolution systems like DARWIN [10] and special heuristics based systems like IDAC [11] and OASYS [12].

These methods have few limitations on the types of problems, specifications, and performance measures that can be considered but they have several disadvantages. They find a locally optimal design (or, even just a "good" or "reasonable" design). The final design depends on the initial design chosen and the algorithm parameters. These methods require substantial human intervention either during the design process, or during the training process.

Global optimization methods

Global optimization methods such as branch and bound and simulated annealing have also been used, *e.g.*, in [13]. Branch and bound unambiguously determines the global optimal design but it is *extremely* slow, with computation growing exponentially with problem size. Simulated annealing (SA) is another very popular method that can avoid becoming trapped in a locally optimal design. In *principle* it can compute the globally optimal solution, but in practical implementations there is no guarantee at all; termination is heuristic. Like classical and knowledge-based methods, SA allows a very wide variety of performance measures and objectives to be handled. Simulated annealing has been used in several tools such as ASTR/OBLX [14] and OPTIMAN [15]. The main disadvantages of SA are that it can be very slow, and in practice it cannot guarantee a global optimal solution.

Convex optimization methods

In a convex optimization problem we minimize a convex objective function subject to linear equality constraints, and inequality constraints that are expressed as upper bounds on convex functions. The great practical advantages of convex optimization are beginning to be widely appreciated, mostly due to the development of extremely powerful interior-point methods for general convex optimization problems in the last five years (*e.g.*, [16, 17]). These methods can solve large problems, with thousands of variables and tens of thousands of constraints, very efficiently (*e.g.*, in minutes on a small workstation). Problems involving tens of variables and hundreds of constraints are considered small, and can be solved on a small current workstation in less than one second.

One very great advantage of convex optimization, compared to general purpose optimization methods, is that the global solution is *always* found, regardless of the starting point. Infeasibility is unambiguously detected, *i.e.*, if the methods do not produce a feasible solution, they produce a certificate that proves that the problem is infeasible.

3 Geometric Programming

Geometric programming (GP) is a special type of convex optimization problem. It has been known and used since the late 1960s (see [18]); more recently it has been widely used in transistor and wire sizing for Elmore delay minimization in digital circuits, as in TILOS [19]. As far as we know, it has not been used before in analog amplifier design.

Let x be a vector $(x_1, \ldots, x_n)$ of n real, positive variables. A function f is called a *posynomial* function of x if it has the form

$$f(x_1, \ldots, x_n) = \sum_{k=1}^{t} c_k x_1^{\alpha_{1k}} x_2^{\alpha_{2k}} \cdots x_n^{\alpha_{nk}}$$

where $c_j \geq 0$ and $\alpha_{ij} \in \mathbf{R}$. When there is only one term in the sum, *i.e.*, $t = 1$, we call f a *monomial* function. Note that posynomials are closed under addition, multiplication, and nonnegative scaling. Monomials are closed under multiplication and division.

A *geometric program* is an optimization problem of the form

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 1, \quad i = 1, \ldots, m, \\ & g_i(x) = 1, \quad i = 1, \ldots, p, \\ & x_i > 0, \quad i = 1, \ldots, n, \end{array} \tag{1}$$

where $f_0, \ldots, f_m$ are posynomial functions and $g_1, \ldots, g_p$ are monomial functions.

If f is a posynomial and g is a monomial, then the constraint $f(x) \leq g(x)$ can be handled by expressing it as $f(x)/g(x) \leq 1$. In a similar way if g_1 and g_2 are both monomial functions, then we can handle the equality constraint $g_1(x) = g_2(x)$ by expressing it as $g_1(x)/g_2(x) = 1$.

We say that a function h is *inverse-posynomial* if $1/h$ is a posynomial function. If h is an inverse-posynomial and f is a posynomial, then GP can handle the constraint $f(x) \leq h(x)$ by expressing it as $f(x)(1/h(x)) \leq 1$. If h is an inverse-posynomial, then we can maximize it, by minimizing $1/h$. Inverse-posynomials are closed under products, and also parallel combinations: if h and $\tilde{h}$ are inverse-posynomial then so is $h \| \tilde{h} = 1/(1/h + 1/\tilde{h})$.

Geometric programming in convex form

A geometric program can be reformulated as a convex optimization problem, by changing variables and considering the logs of the functions involved. We define new variables $y_i = \log x_i$, and take the logarithm of a posynomial f to get

$$h(y) = \log f\left(e^{y_1}, \ldots, e^{y_n}\right) = \log \left(\sum_{k=1}^{t} e^{a_k^T y + b_k} \right)$$

where $a_k^T = [\alpha_{1k} \cdots \alpha_{nk}]$ and $b_k = \log c_k$. It can be shown that h is a *convex* function of the variable y.

Thus, we can convert the standard geometric program (1) into a convex program by expressing it as

$$\begin{array}{ll} \text{minimize} & h_0(y) = \log f_0(e^{y_1}, \ldots, e^{y_n}) \\ \text{subject to} & h_i(y) = \log f_i(e^{y_1}, \ldots, e^{y_n}) \leq 0, \\ & v_i(y) = \log g_i(e^{y_1}, \ldots, e^{y_n}) = 0. \end{array} \tag{2}$$

This is the so-called *exponential form* of the geometric program (1) (see, *e.g.*, [20]).

There are several methods for solving geometric programs. One option is to solve the exponential form of the geometric program using a general purpose optimization code such as NPSOL or MINOS. These general purpose codes will in principle find the globally optimal solution, but codes specifically designed for solving geometric programs offer greater computational efficiency. Recently, Kortanek et al. have shown how the most sophisticated primal-dual interior-point methods used in linear programming can be extended to GP, resulting in an algorithm with efficiency approaching that of current interior-point linear programming solvers [21].

For use in GPCAD we have implemented, in MATLAB, a very simple primal barrier method, which is described in [1] and [20]. Despite the simplicity of our algorithm and the overhead of an interpreted language, the GPs arising in this paper are solved in less than two seconds on a SUN1 ULTRA-SPARC 1 (170MHz) workstation. A more efficient algorithm and implementation would make it considerably faster.

4 Transistor modeling

The transistor model we use has the following form, which we call a *GP1 model.*

- The overdrive voltage $V_{\rm gs} - V_{\rm TH}$ is a monomial function of transistor length L, transistor width W and transistor drain current I.
- The transconductance $g_{\rm m}$ is a monomial function in L, W, and I.
- The output conductance $g_{\rm o}$ is given by $\alpha g_{\rm o,m}$ where $g_{\rm o,m}$ is monomial in L, W, and I, and α is a constant. We use two different values of α, depending on whether the transistor in question typically operates with large or small $V_{\rm ds}$.
- Capacitances between the terminals and bulk are posynomial in L, W, and I.

The traditional long channel MOS transistor model (level 1 in HSPICE) fits exactly the format of the GP1 model (see [1]). By simple data fitting techniques [20], we have obtained GP1 models that have reasonable agreement with HSPICE higher order models (BSIM1 models), over large ranges of length, width, and bias currents. To model submicron devices, we are now developing more sophisticated and accurate models that are still compatible with geometric programming based design. These models, which we call GP2, are described in [2].

5 Op-amp architectures

We will apply the method to the six different op-amp architectures, or topologies, shown in figures 1–6. These are:

- a two-stage op-amp (figure 1)
- a simple OTA op-amp (figure 2)
- an OTA op-amp (figure 3)
- a two-stage cascoded op-amp (figure 4)
- a folded-cascode op-amp (figure 5)
- a telescopic op-amp (figure 6)

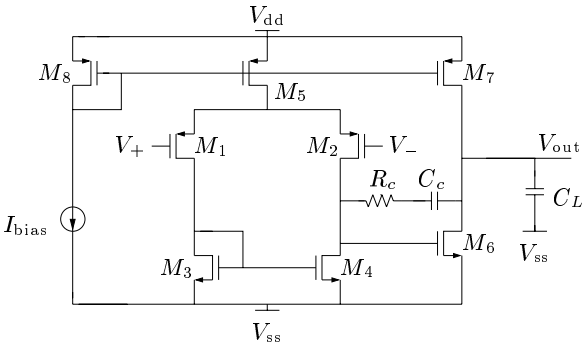

Figure 1: Two stage op-amp.

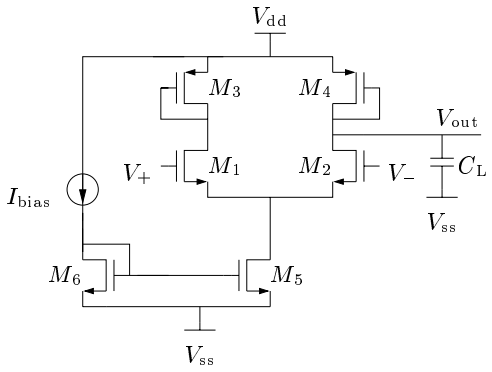

Figure 2: Simple OTA op-amp.

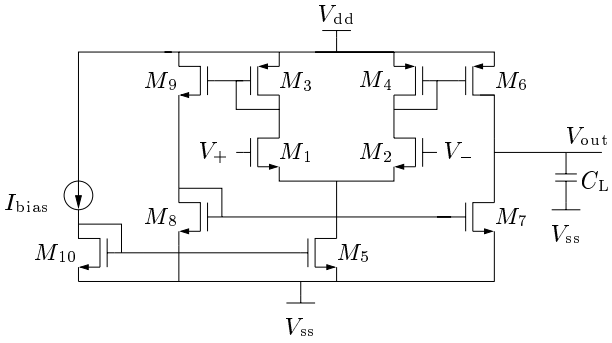

Figure 3: OTA op-amp.

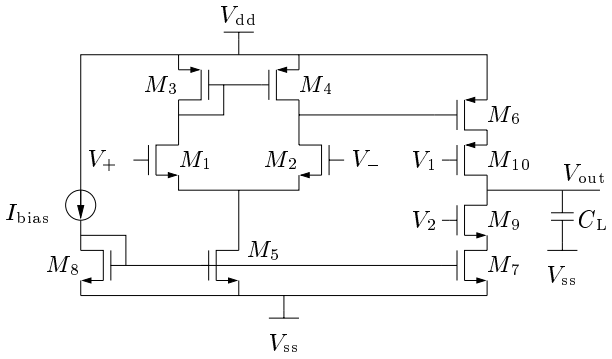

Figure 4: Two stage cascoded op-amp.

6 Performance specifications/constraints

In this section we show how a variety of performance measures and constraints can be formulated using geometric programming. To simplify the discussion (as well as to save space) we concentrate on one op-amp topology, the two-stage op-amp of figure 1, and assume a simple MOS model; more details can be found in [1]. In each section we give short comments about how the method extends to the other five op-amp architectures, as well as more sophisticated MOS models.

The design variables are the transistor sizes (width and length), the value of the passive components (capacitors and resistors), and the value of bias currents and bias voltages. For this particular two-stage op-amp, there are nineteen design variables.

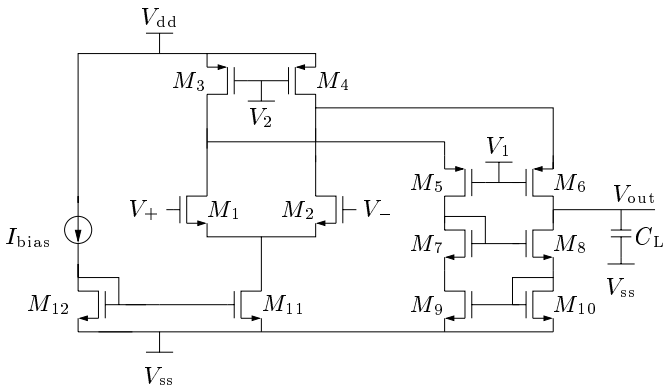

Figure 5: Folded-cascode op-amp.

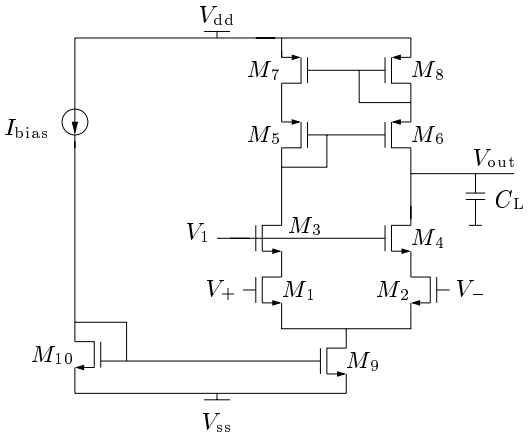

Figure 6: Telescopic op-amp.

Symmetry and matching

In many op-amps there are some symmetry and matching constraints that must be met. For example, the input transistors should be identical and the bias transistors should have the same length to improve the matching among them. These conditions can be expressed as monomial constraints of the form

$$W_i = W_j, \quad L_i = L_j. \tag{3}$$

(Equivalently we can simply work with a smaller number of variables, *e.g.*, using W_i to represent the widths of several transistors. For the small problems we encounter, however, keeping the extra variables and equality constraints results in a very small computational penalty.)

Device sizes

Lithography limitations and layout rules impose lower limits on the device sizes:

$$L_{\min} \leq L_i, \quad W_{\min} \leq W_i, \quad i = 1, \ldots, n. \tag{4}$$

These constraints are monomial and can be handled by GP.

Another common constraint that *cannot* be handled by GP is that the lengths and widths can only assume discrete values, *e.g.*, multiples of some small dimension. In practice this is not a problem since many transistors end up being minimum length, and transistor widths are often much larger than the minimum grid resolution making the rounding error for device width small.

Area

An approximate expression for the active device op-amp die area A is given by the sum of the sum of transistor and capacitor ares,

$$A = \alpha_0 + \alpha_1 C_c + \alpha_2 \sum_{i=1}^{n} W_i L_i, \tag{5}$$

where the α_i are positive constants. This expression is a posynomial function of the design parameters, so we can impose an upper bound on the area, or use the area as the objective to be minimized.

Current equations

Biasing is typically accomplished by mirroring a reference current to the different stages of the op-amp. In this case the bias currents become monomials of the design parameters. In the two-stage op-amp, for example, if we define the bias currents I_5 and I_7 through transistors M_5 and M_7, respectively, we have,

$$I_5 = \frac{W_5 L_8}{L_5 W_8} I_{\rm bias}, \quad I_7 = \frac{W_7 L_8}{L_7 W_8} I_{\rm bias}. \tag{6}$$

Since these expressions are monomial we can use I_5 and I_7 as part of the design variables, considering (6) as an equality constraint. This reduces the complexity of the other design equations, at some (negligible) computational cost. Note that the bias reference current, $I_{\rm bias}$ can be fixed (by the biasing scheme of the IC) or can be a design variable.

Bias conditions

For the correct operation of most op-amps the transistors need to remain in the saturation region for the specified input common mode range ($[V_{\rm cm,min}, V_{\rm cm,max}]$) and the specified output voltage swing ($[V_{\rm out,min}, V_{\rm out,max}]$). These bias, input common mode, and output swing constraints turn out to have posynomial form. For example, using the long channel square-law MOS transistor model in the two-stage op-amp, the condition for M_5 remaining in saturation is

$$\sqrt{\frac{I_1 L_1}{K_{\rm p} W_1}} + \sqrt{\frac{I_5 L_5}{K_{\rm p} W_5}} \leq V_{\rm dd} - V_{\rm cm,max} + V_{\rm TP}, \tag{7}$$

which is a posynomial constraint.

Quiescent power

For the two-stage op-amp the quiescent power has the form

$$P = (V_{\rm dd} - V_{\rm ss})\,(I_{\rm bias} + I_5 + I_7)\,, \tag{8}$$

which is a posynomial function of the design parameters, so we can impose an upper bound on, or minimize, quiescent power. Similar equations hold for the other architectures we consider.

Open-loop DC gain

For the two-stage op-amp, the open-loop voltage gain is given by

$$A_v = \left(\frac{g_{\rm m2}}{g_{\rm o2} + g_{\rm o4}}\right)\left(\frac{g_{\rm m6}}{g_{\rm o6} + g_{\rm o7}}\right), \tag{9}$$

which is an inverse-posynomial function of the design parameters. We can therefore impose a minimum required open-loop voltage gain or we can maximize the gain.

The other five op-amp topologies have different expressions for the open-loop voltage gain, but in each case the gain is inverse-posynomial. This can be explained as follows. The total op-amp

gain is the product of the gain of the stages. the gain of each stage has the typical form $G_i = g_{\rm m}/g_{\rm out}$ where $g_{\rm m}$ is the transconductance of the stage input transistor and $g_{\rm out}$ is the output conductance seen by the transistor. Since $g_{\rm out}$ is a sum of several load conductances, each of which is monomial in the design variables, the stage gain is inverse-posynomial. Therefore the overall gain is inverse-posynomial.

Unity-gain bandwidth

The op-amps we consider are designed to have a dominant pole. For the two-stage op-amp the dominant pole is given by

$$p_1 = g_{\rm m1}/(A_v C_c), \tag{10}$$

and the unity gain bandwidth is given by the expression

$$\omega_c = g_{\rm m1}/C_c, \tag{11}$$

which is monomial. For the other architectures, the unity-gain bandwidth is given by the ratio of the transconductance of some device to the compensation (or output) capacitance, and is either monomial or inverse-posynomial. Therefore we can impose lower bounds on, or maximize, the unity-gain bandwidth.

Non-dominant pole conditions

With the choice of the compensation resistor $R_c = 1/g_{\rm m6}$, the two-stage op-amp has three non-dominant poles given by

$$\begin{aligned} p_2 &= g_{\rm m6}C_c/(C_1C_c + C_1C_{\rm TL} + C_cC_{\rm TL}) && (12)\\ p_3 &= g_{\rm m3}/C_2 && (13)\\ p_4 &= g_{\rm m6}/C_1, && (14)\end{aligned}$$

where C_1, $C_{\rm L}$ and C_2 can be expressed as posynomial expressions of the transistors widths and lengths,

$$\begin{aligned} C_1 &= C_{\rm gs6} + C_{\rm db2} + C_{\rm db4} + C_{\rm gd2} + C_{\rm gd4}\\ C_{\rm TL} &= C_{\rm L} + C_{\rm db6} + C_{\rm db7} + C_{\rm gd6} + C_{\rm gd7}\\ C_2 &= C_{\rm gs3} + C_{\rm gs4} + C_{\rm db1} + C_{\rm db3} + C_{\rm gd1}. \end{aligned}$$

The important point is that the non-dominant poles p_2, p_3 and p_4 are given by inverse-posynomial functions of the design parameters. We can therefore impose a lower bound on the non-dominant poles (*e.g.*, limiting them to be a decade above the unity-gain bandwidth). For the other architectures as well, the non-dominant poles are given by inverse-posynomial expressions.

Phase margin

For large gains, the phase due to the dominant pole at the unity-gain frequency will be very nearly 90°. For small phase shifts (less than 25°), we have $\arctan(x) \approx x$, so the phase margin constraint can be approximated as

$$\sum_{i=2}^{4} \frac{\omega_{c,\rm approx}}{p_i} \leq \frac{\pi}{2} - {\rm PM}_{\rm min}, \tag{15}$$

which is a posynomial inequality in the design variables. The approximation error is very small since the phase contributed by each non-dominant pole is small (less than 25°).

In the general case, the phase margin depends on the sum of phase shifts, at the unity-gain frequency, contributed by the non-dominant poles and zeros. Left half plane poles and right half plane zeros contribute phase shifts that can be approximated, using $\arctan(x) \approx x$, as posynomial expressions. For the six op-amps considered, the resulting phase margin constraints are posynomial.

Slew Rate

The slew rate is typically determined by the amount of current that can be sourced or sinked into the compensation (or output) capacitance. For the two-stage op-amp the conditions to ensure a minimum slew rate $\rm SR_{min}$ are

$$C_c/(2I_1) \leq 1/{\rm SR}_{\rm min}, \quad ({\rm C_c} + {\rm C_{TL}})/{\rm I_7} \leq 1/{\rm SR}_{\rm min}, \tag{16}$$

which are posynomial inequalities. A similar situation holds for the other architectures.

CMRR

For the two-stage op-amp, the common mode rejection ratio is given by

$${\rm CMRR} = \frac{2{\rm g_{m1}g_{m3}}}{({\rm g_{o3}} + {\rm g_{o1}})\,{\rm g_{o5}}}, \tag{17}$$

which is inverse-posynomial, so we can impose a minimum CMRR. For the other five op-amp topologies, the expression for the CMRR is also inverse-posynomial. In simple op-amps the CMRR is just the ratio of the differential gain of the first stage to the common mode gain of the first stage and is proportional to the output resistance of the biasing network.

There are several other specifications that can be handled using geometric programming, but are omitted here for space considerations. These include, for example, minimum gate overdrive voltage, minimum 3dB bandwidth, and limits on input-referred spot noise, or total RMS noise in a band (see [1]).

7 Design examples

In this section we provide some sample designs for the two-stage op-amp using a standard 1.2μm CMOS process (described in more detail in [1]). The load capacitance is 5pF and the supply voltages are $V_{\rm dd} = 5$V and $V_{\rm ss} = 0$V.

A simple design example

Table 1 describes the sample design problem, and shows the performance of the design obtained by GPCAD using GP1 models, and the simulated performance with BSIM1 models (HSPICE level 13). The objective was to maximize the unity gain bandwidth subject to the other given constraints. The first column in table 1 identifies the performance measure (and its units); the second gives the specification (showing whether it is an upper or lower bound). The third column, labeled GPCAD, shows the performance of the design obtained, according to the GP1 model of GPCAD. The fourth column, labeled HSPICE, shows the value of the specification as simulated by BSIM1 from our design.

We can see that there is close agreement between the predicted results from GPCAD and the HSPICE simulations. Recall that the GPCAD values are based on simple posynomial expressions, as well as a number of approximations; the HSPICE simulations are based on sophisticated MOS models, and no approximations (*e.g.*, of transfer functions). The close agreement between the two shows that our posynomial models and approximations, though simple, are adequate for real design.

The computer time required for the design is approximately two seconds, using an inefficient MATLAB implementation of an interior-point method. (An efficient implementation would be far faster.)

The optimal design parameters found are shown in table 2. It is interesting to note that only device M_6 is minimum length. This can be easily explained. BSIM1 models and our GP1 models take

into account the fact that the output conductance of the device decreases with increasing transistor length. Since the gain requirement is high, longer devices are needed to obtain 80dB of open-loop gain. The output conductances of M_1, M_3, M_6 and M_7 affect the final gain but only device M_6 is minimum length. The reason is that the second non-dominant pole position is proportional to the transconductance of M_6. Since the phase margin requirement is stringent, GPCAD decides to increase the transconductance of M_6 by making it as short as possible. If the gain requirement is dropped to 60dB, all transistors are minimum length and the unity-gain frequency is 25MHz. One may also think that transistor M_1 should be minimum length so its transconductance is maximum. The problem is that for larger unity-gain bandwidths it is very hard to meet the phase margin specification, and GPCAD opts for a lower M_1 transconductance.

We remind the reader that the design found by GPCAD is *globally* optimal, and not merely (as with local methods) a locally optimal design. Please note also that our discussion above is offered only as an explanation for what GPCAD obtained; GPCAD obtained the design by solving a geometric program, and not using any circuit design reasoning!

Performance measure	Spec	GPCAD	HSPICE
Area (μm^2)	≤ 2.5k	2.5k	2.5k
Max. output(V)	≥ 4.5	4.5	4.5
Min. output (V)	≤ 0.5	0.1	0.1
Power (mW)	≤ 5	0.5	0.6
DC gain (dB)	≥ 80	80	83
Unity-gain BW (MHz)	Max.	9	8
Phase margin ($^\circ$)	≥ 60	60	66
Slew rate (V/μs)	≥ 10	10	11
CMRR (dB)	≥ 60	80	84
PSRRn (dB)	≥ 80	91	94
PSRRp (dB)	≥ 80	95	97
Noise, 1kHz(nV/$\sqrt{\text{Hz}}$)	≤ 600	600	590
$L_{\min}$ (μm)	≥ 1.2	1.2	1.2
$W_{\min}$(μm)	≥ 2	2.3	2.3

Table 1: Design specifications for two-stage op-amp.

Variable	Value
$W_1 = W_2$	41.3μm
$W_3 = W_4$	18.2μm
W_5	5.4μm
W_6	520μm
W_7	109μm
W_8	2.3μm
$L_1 = L_2$	3.3μm
$L_3 = L_4$	1.7μm
L_5	2.6μm
L_6	1.2μm
L_7	2.6μm
L_8	2.6μm
C_c	0.58pF
I_{bias}	2.6μA

Table 2: Optimal design for design example.

A globally optimal trade-off curve

By repeatedly solving the design problem while varying one of the specifications, we can sweep out the *globally optimal* trade-off curve between competing objectives (with the others fixed). In figure 7 we plot a trade-off curve for the two-stage op-amp, obtained by maximizing unity-gain bandwidth, while varying the maximum allowed power dissipation, for three different supply voltages. The rest of specifications are set to the values given in table 2, except the area, which is relaxed to $10^4\mu$m^2. It is interesting to note that the curves cross, with different power supply voltages being optimal for different maximum powers (or unity gain bandwidths). We see that for low power designs we obtain a higher bandwidth with lower supply voltages but when power is not a limit we are better off with a higher supply voltage. The reason is that for a given power, the low voltage design can use more current than the high voltage design.

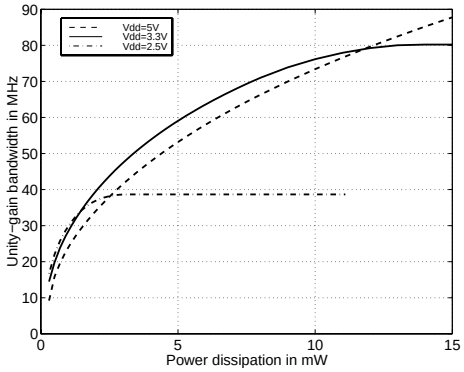

Figure 7: Globally optimal trade-off curve between maximum unity-gain bandwidth and maximum power, for different supply voltages.

Each trade-off curve was computed in under two minutes, by solving each problem from scratch, *i.e.*, "cold start". This would be far faster with an efficient implementation, using "warm start", *i.e.*, starting from the previous design found.

Designs for the other architectures

In this section we give design examples for the other op-amp topologies studied: simple OTA op-amp (figure 2, table 3), OTA op-amp (figure 3, table 4), two-stage cascoded op-amp (figure 4, table 5), folded-cascode op-amp (figure 5, table 6), and telescopic op-amp (figure 6, table 7).

For each problem the objective was to minimize area. The other specifications are given in the table as the first entry; the next two entries give the value obtained by GPCAD, using the GP1 model, and the simulated value from HSPICE BSIM1. The specifications are similar, but not the same, since the op-amps differ considerably in achievable gain, bandwidth, etc. As in the example design above for the two-stage op-amp, the results obtained from GPCAD agree quite closely with HSPICE simulation.

Trade-off curves for telescopic and folded-cascode op-amps

In figure 8 we show the globally optimal trade-off curves between DC gain and unity-gain bandwidth, for the telescopic op-amp and the folded-cascode op-amp, each subject to the same specifications. The curves show that the telescopic op-amp achieves more gain at higher unity-gain bandwidths. The reason is that for the same quiescent power, the input transistors in the telescopic op-amp carry approximately twice the current of the input transistors in the folded-cascode op-amp. In this example, the output voltage swing constraint was set to only 2V. If this last requirement is tougher the folded-cascode outperforms the telescopic op-amp.

Performance measure	Spec	GPCAD	HSPICE
C_{L} (pF)	1	1	1
DC gain (dB)	≥ 40	40	39
Unity-gain BW(MHz)	≥ 50	50	50
Phase margin ($^\circ$)	≥ 60	65	70
Power (mW)	≤ 1	0.4	0.4
PSRRp (dB)	≥ 20	65	70
PSRRn (dB)	≥ 20	40	40
Output swing (V)	≥ 2.3	3.5	3.6
Slew Rate (V/μs)	≥ 10	37	40
Area(μm^2)	Min.	400	400

Table 3: Simple OTA design.

Performance measure	Spec	GPCAD	HSPICE
C_{L} (pF)	1	1	1
DC gain (dB)	≥ 40	40	44
Unity-gain BW(MHz)	≥ 25	25	25
Phase margin ($^\circ$)	≥ 45	53	58
Power (mW)	≤ 1	0.75	0.75
PSRRp (dB)	≥ 40	40	42
PSRRn (dB)	≥ 40	49	52
Output swing (V)	≥ 2.5	3	2.8
Slew Rate (V/μs)	≥ 35	48	50
Area(μm^2)	Min.	300	300

Table 4: OTA design.

Performance measure	Spec	GPCAD	HSPICE
C_{L} (pF)	1	1	1
DC gain (dB)	≥ 60	92	95
Unity-gain BW(MHz)	≥ 10	10	10
Phase margin ($^\circ$)	≥ 60	60	59
Power (mW)	≤ 1	0.14	0.14
PSRRp (dB)	≥ 20	86	88
PSRRn (dB)	≥ 40	70	72
Output swing (V)	≥ 2	3.5	3.5
Slew Rate (V/μs)	≥ 2	7	6
Area(μm^2)	Min.	1900	1900

Table 5: Two stage cascoded op-amp design.

Performance measure	Spec	GPCAD	HSPICE
C_{L} (pF)	1.25	1.25	1.25
DC gain (dB)	≥ 70	70	70
Unity-gain BW(MHz)	≥ 50	50	50
Phase margin ($^\circ$)	≥ 60	60	74
Power (mW)	≤ 1	0.94	0.9
PSRRp (dB)	≥ 70	77	79
PSRRn (dB)	≥ 70	70	69
Output swing (V)	≥ 1	2.1	2.4
Slew Rate (V/μs)	≥ 50	50	52
Area(μm^2)	Min.	720	720

Table 6: Folded-cascode op-amp design.

This example points out a very important feature of GPCAD. By itself, GPCAD does not design circuit topology or architecture; it merely designs component values and transistor dimensions for a given architecture. By solving the same problem (*i.e.*, identical constraints and objective) for several different architectures, GPCAD can decide which architecture is best. Here again the fact that GPCAD finds globally optimal solutions for each architecture, and not just locally optimal solutions, is critical. It allows us to say with certainty that, for a given set of specifications, one architecture is better than another.

Performance measure	Spec	GPCAD	HSPICE
C_{L} (pF)	1	1	1
DC gain (dB)	≥ 80	80	78
Unity-gain BW(MHz)	≥ 20	33	33
Phase margin ($^\circ$)	≥ 60	60	70
Power (mW)	≤ 1	0.1	0.13
PSRRp (dB)	≥ 70	74	78
PSRRn (dB)	≥ 70	75	76
Output swing (V)	≥ 2	2	2.2
Slew Rate (V/μs)	≥ 20	20	22
Area(μm^2)	Min.	440	440

Table 7: Telescopic op-amp design.

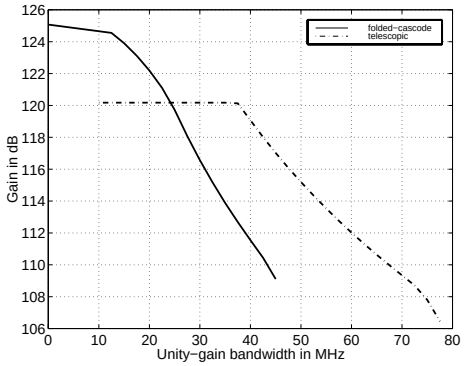

Figure 8: Maximum achievable gain versus unity-gain bandwidth for folded-cascode and telescopic op-amp architectures.

8 Conclusions and extensions

We have shown how geometric programming can be used to design CMOS op-amps. The method is very efficient, can handle a wide variety of constraints and provides globally optimal designs. Even with relatively simple transistor models, we achieve good agreement with SPICE simulations based on sophisticated models. We are currently developing more sophisticated and accurate models, that are compatible with geometric programming design, and are very accurate even for submicron, short channel designs [2].

GPCAD can also be effectively combined with a (local) optimization method that uses more accurate model equations, or even circuit simulation (such as DELIGHT.SPICE). Thus, GPCAD is used to get close to the (presumably global) optimum, and the final design is tuned using the more accurate model or direct circuit simulation. This would reduce the search time of the local optimizer considerably while still preserving extreme accuracy.

We mention here several important features that space limitations did not allow us to mention above. When the geometric program is solved, we get a complete sensitivity analysis of the problem, without any additional computational effort. These sensitivity numbers provide extremely useful information to the designer; it shows which constraints are 'most' binding; which constraints can be relaxed to obtain a great improvement in the objective function, and which constraints can be tightened without much cost. See [1] for a complete discussion of this topic (as well as the sensitivities for some of the designs described in this paper).

Another useful feature of GPCAD is its ability to develop *robust* designs, *i.e.*, designs that guarantee a set of specifications are met for a variety of different processes and technology parameter values. This is done by replicating the design constraints for the different operating conditions, which is practical only because the computational effort for solving geometric programs grows approximately linearly with the number of constraints.

Although GPCAD does not directly design op-amp topology, it can be used to choose the best op-amp topology out of a number of topologies. This would work as follows: first, a library of topologies is developed; for each one we have a method for translating the op-amp specifications into an associated geometric program. (Thus, in this paper we consider a small library of six topologies.) Then for a specific set of design specifications, one can perform the design for each op-amp architecture in the library. This evidently identifies the best architecture in the library, for the given specifications. Note that (as in the example above) which architecture is best depends on the specifications. Since each design is very fast (*i.e.*, 2 seconds with our current, inefficient implementation), a large library (with, say, hundreds of architectures) can be scanned quickly. This process can be speeded up in several ways. For example, the optimization process for a given architecture can be terminated once it is clear that the optimal value is more than the best optimal value found so far. Again, this is only practical because geometric programs can be solved so efficiently.

We end our discussion by explaining what needs to be done to apply geometric programming to the design of a specific circuit, *i.e.*, to develop a library entry for a given circuit topology that has not yet been analyzed. The task is to express the design constraints and performance specifications in a form appropriate for geometric programming. At this point, we have no method for automating this step, although we envision using some type of symbolic analyzer (*e.g.*, [22]) to aid in this process. Of course, this step is done only once for each topology; once the library entry exists, specific design problems involving that topology are solved within a few of seconds, since only geometric programming is involved.

In a similar way, new GP1 models have to be developed for each new process technology. This step, which is based on fitting monomials to given tabulated data (which come from a sophisticated model, or even from empirical device measurements) is completely automated; it only takes a few minutes (assuming the data are already available). This is discussed in depth in [2].

References

[1] M. Hershenson, S.Boyd, and T. Lee. Optimal design of a CMOS op-amp via geometric programming. http://www-isl.stanford.edu/people/boyd (submitted to IEEE Transactions on Computer-Aided Design).

[2] M. Hershenson, S. Boyd, and T. H. Lee. MOS models for geometric programming. Technical report, Stanford University, 1998. http://www-isl.stanford.edu/people/boyd.

[3] P. E. Gill et al.˙ User's guide for NPSOL (Version 4.0): A FORTRAN package for nonlinear programming. Technical Report SOL 86-2, Operations Research Dept., Stanford University, Stanford, California 94305, January 1986.

[4] P. C. Maulik et al.˙ Rapid redesign of analog standard cells using constrained optimization techniques. In *IEEE Custom Integrated Circuit Conference*, pages 8.1.1–8.1.3, 1992.

[5] H. Chang et al. A top-down constraint-driven design methodology for analog integrated circuits. In *IEEE Custom Integrated Circuit Conference*, pages 8.4.1–8.4.6, 1992.

[6] H. Y. Koh, C. H. Séquin, and P. R. Gray. OPASYN: A compiler for CMOS operational amplifiers. *IEEE Transactions on Computer-Aided Design*, 9:113–125, February 1990.

[7] H. Onodera, H. Kanbara, and K. Tamaru. Operational amplifier compilation with performance optimization. *IEEE Journal of Solid-State Circuits*, 25:466–473, April 1990.

[8] J. P. Harvey, M. I. Elmasry, and B. Leung. STAIC: An interactive framework for synthesizing CMOS and BiCMOS analog circuits. *IEEE Transactions on Computer-Aided Design*, 11:1402–1417, November 1992.

[9] W. Nye et al.˙ DELIGTH.SPICE: An optimization-based system for the design of integrated circuits. *IEEE Transactions on Computer-Aided Design*, 7:501–518, April 1988.

[10] W. Kruiskamp and D. Leenaerts. DARWIN: CMOS op amp synthesis by means of a genetic algorithm. In *Proceedings of the 32nd Annual Design Automation Conference*, pages 433–438, 1995.

[11] M. G. R. Degrauwe et al. Towards an analog system design environment. *IEEE Journal of Solid-State Circuits*, 24:1587–1597, December 1989.

[12] R. Harjani, R. A. Rutenbar, and L. R. Carley. OASYS: A framework for analog circuit synthesis. *IEEE Transactions on Computer-Aided Design*, 8:1247–1265, December 1989.

[13] P. C. Maulik, L. R. Carley, and R. A. Rutenbar. Integer programming based topology selection of cell-level analog circuits. *IEEE Transactions on Computer-Aided Design*, 14:401–412, April 1995.

[14] E. S. Ochotta, R. A. Rutenbar, and L. R. Carley. Synthesis of high-performance analog circuits in ASTRX/OBLX. *IEEE Transactions on Computer-Aided Design*, 15:273–293, March 1996.

[15] G. G. E. Gielen, H .C. C. Walscharts, and W. M. C. Sansen. Analog circuit design optimization based on symbolic simulation and simulated annealing. *IEEE Journal of Solid-State Circuits*, 25:707–713, June 1990.

[16] Y. Nesterov and A. Nemirovsky. *Interior-point polynomial methods in convex programming*, volume 13 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, 1994.

[17] S. J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, Philadelphia, 1997.

[18] R. J. Duffin, E. L. Peterson, and C. Zener. *Geometric Programming — Theory and Applications*. Wiley, 1967.

[19] J. P. Fishburn and A. E. Dunlop. TILOS: A polynomial programming approach to transistor sizing. In *IEEE International Conference on Computer-Aided Design*, pages 326–328, 1985.

[20] S. Boyd and L. Vandenberghe. Introduction to convex optimization with engineering applications. Course Notes, 1997. http://www-leland.stanford.edu/class/ee364/.

[21] K. O. Kortanek, X. Xu, and Y. Ye. An infeasible interior-point algorithm for solving primal and dual geometric programs. *Math Programming*, 76:155–181, 1996.

[22] F. V. Fernández, A. Rodríguez-Vázquez, J. L. Huertas, and G. G. R. Gielen. *Symbolic Analysis Techniques*. IEEE Press, 1998.