1

Optimal Design of a CMOS Op-Amp via Geometric Programming

Maria del Mar Hershenson, Stephen P. Boyd, and Thomas H. Lee

ABSTRACT We describe a new method for determining component values and transistor dimensions for CMOS operational amplifiers (op-amps). We observe that a wide variety of design objectives and constraints have a special form, i.e.*, they are* posynomial *functions of the design variables. As a result the amplifier design problem can be expressed as a special form of optimization problem called* geometric programming, *for which very efficient* global *optimization methods have been developed. As a consequence we can efficiently determine* globally optimal *amplifier designs, or globally optimal trade-offs among competing performance measures such as power, open-loop gain, and bandwidth. Our method therefore yields completely automated synthesis of (globally) optimal CMOS amplifiers, directly from specifications.*
In this paper we apply this method to a specific, widely used operational amplifier architecture, showing in detail how to formulate the design problem as a geometric program. We compute globally optimal trade-off curves relating performance measures such as power dissipation, unity-gain bandwidth, and open-loop gain. We show how the method can be used to synthesize robust designs, i.e.*, designs guaranteed to meet the specifications for a variety of process conditions and parameters.*

1.1 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 this paper we introduce a new method for determining the component values and transistor dimensions for CMOS op-amps. The method handles a very wide variety of specifications and constraints, is *extremely fast*, and results in *globally optimal* designs.

The performance of an op-amp is characterized by a number of performance measures such as open-loop voltage gain, quiescent power, input-referred noise, output voltage swing, unity-gain bandwidth, input offset voltage, common-mode rejection ratio, slew rate, die area, and so on. These performance measures are determined by the design parameters, *e.g.*, transistor dimensions, bias currents, and other component values. The CMOS amplifier design problem we consider in this paper is to determine values of the design parameters that optimize an objective measure while satisfying specifications or constraints on the other performance measures. This design problem can be approached in several ways, for example by hand or a variety of computer-aided design methods, *e.g.*, classical optimization methods, knowledge-based methods, or simulated annealing. (These methods are described more fully below).

In this paper, we introduce a new method that has a number of important advantages over current methods. We formulate the CMOS op-amp design problem as a very special type of optimization problem called a *geometric program*. The most important feature of geometric programs is that they can be reformulated as *convex optimization problems*, and therefore *globally optimal* solutions can be computed with *great efficiency*, even for problems with hundreds of variables and thousands of constraints, using recently developed interior-point algorithms. Thus, even challenging amplifier design problems with many variables and constraints can be (globally) solved.

The fact that geometric programs (and hence, CMOS op-amp design problems cast as geometric programs) can be globally solved has a number of important practical consequences. The first is that sets of infeasible specifications are unambiguously recognized: the algorithms either produce a feasible point or a proof that the set of specifications is infeasible. Indeed, the choice of initial design for the optimization procedure is completely irrelevant (and can even be infeasible); it has no effect on the final design obtained. Since the global optimum is found, the op-amps obtained are not just the best our method can design, but in fact the best *any* method can design (with the same specifications). In particular, our method computes the *absolute limit of performance* for a given amplifier and technology parameters.

The fact that geometric programs can be solved very efficiently has a number of practical consequences. For example, the method can be used to simultaneously optimize the design of a large number of op-amps in a single large mixed mode integrated circuit. In this case the designs of the individual op-amps are coupled by constraints on total power and area, and by various parameters that affect the amplifier coupling such as input capacitance, output resistance, etc. Another application is to use the efficiency to obtain *robust designs*, *i.e.*, designs that are guaranteed to meet a set of specifications over a variety of processes or technology parameter values. This is done by simply replicating the specifications with a (possibly large) number of representative process parameters, which is practical only because geometric programs with thousands of constraints are readily solved.

All the advantages mentioned (convergence to a global solution, unambiguous detection of infeasibility, sensitivity analysis ...) are due to the formulation of the design problem as a *convex* optimization problem. Geometric programming (when reformulated as described in §1.2.1) is just a special type of convex optimization problem. Although general convex problems can be solved efficiently, the special structure of geometric programming can be exploited to obtain an even more efficient solution algorithm.

The method we present can be applied to a wide variety of amplifier architectures, but in this paper we apply the method to a specific two-stage CMOS op-amp. The authors show how the method extends to other architectures in [50, 49].

The work reported in this paper originated as Mar Hershenson's project for the course EE364, *Introduction to Convex Optimization with Engineering Applications*, given at Stanford in Winter quarter 1996-7. Related work has been reported in several conference publications, *e.g.*, [47, 49, 50]; a preliminary version of this paper has been available at URL

`http://www.stanford.edu/~boyd/opamp.html`

since November 1997.

1.1.1 The two-stage amplifier

The specific two-stage CMOS op-amp we consider is shown in Figure 1.1. The circuit consists of an input differential stage with active load followed by a common-source stage also with active load. An output buffer is not used; this amplifier is assumed to be part of a VLSI system and is only required to drive a fixed on-chip capacitive load of a few picofarads. This op-amp architecture has many advantages: high open-loop voltage gain, rail-to-rail output swing, large common-mode input range, only one frequency compensation capacitor, and a small number of transistors. Its main drawback is the nondominant pole formed by the load capacitance and the output impedance of the second stage, which reduces the achievable bandwidth. Another potential disadvantage is the right half plane zero that arises from the feedforward signal path through the compensating capacitor. Fortunately the zero is easily removed by a suitable choice for the compensation resistor R_c (see [2]).

This op-amp is a widely used general purpose op-amp [87]; it finds applications for example in switched capacitor filters [23], analog to digital converters [71, 59], and sensing circuits [84].

There are 18 design parameters for the two-stage op-amp:

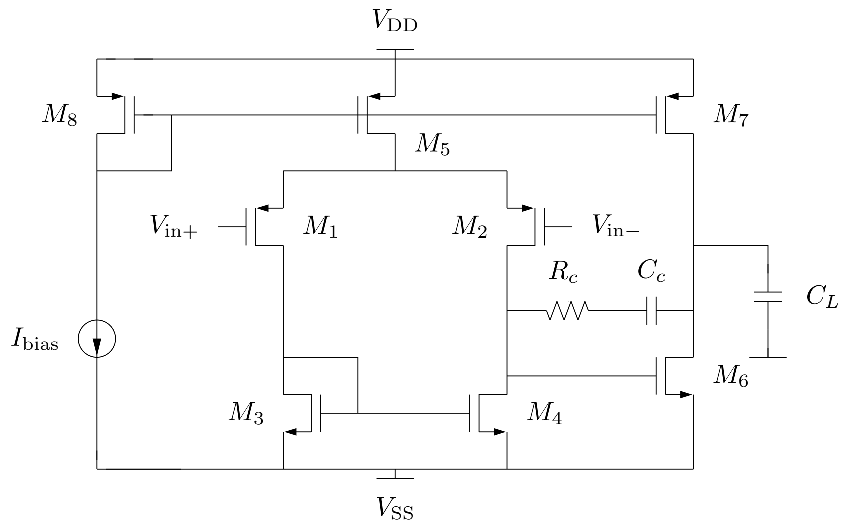

FIGURE 1.1. Two stage op-amp considered in this paper.

- The widths and lengths of all transistors, *i.e.*, $W_1, \ldots, W_8$ and $L_1, \ldots, L_8$.
- The bias current $I_{\rm bias}$.
- The value of the compensation capacitor C_c.

The compensation resistor R_c is chosen in a specific way that is dependent on the design parameters listed above (and described in §1.5). There are also a number of parameters that we consider fixed, *e.g.*, the supply voltages $V_{\rm DD}$ and $V_{\rm SS}$, the capacitive load C_L, and the various process and technology parameters associated with the MOS models. To simplify some of the equations we assume (without any loss of generality) that $V_{\rm SS} = 0$.

1.1.2 Other approaches

There is a huge literature, which goes back more than twenty years, on computer-aided design of analog circuits. A good survey of early research can be found in the survey [11]; more recent papers on analog circuit CAD tools include, *e.g.*, [13, 4, 12]. The problem we consider in this paper, *i.e.*, selection of component values and transistor dimensions, is only a part of a complete analog circuit CAD tool. Other parts, that we do not consider here, include topology selection (see, *e.g.*, [65]) and actual circuit layout (see, *e.g.*, ILAC [26], KOAN/ANAGRAM II [15]). The part of the CAD process that we consider lies between these two tasks; the remainder of the discussion is restricted to methods dealing with component and transistor sizing.

Classical optimization methods

General purpose classical optimization methods, such as steepest descent, sequential quadratic programming and Lagrange multiplier methods, have been widely used in analog circuit CAD. These methods can be traced back to the survey paper [11]. The widely used general purpose optimization codes NPSOL [38] and MINOS [70] are used in, *e.g.*, [63, 66, 24]. LANCELOT [16], another general purpose optimizer, is used in [22]. Other CAD approaches based on classical optimization methods, and extensions such as a minimax formulation, include the one described in [62, 46, 60], OAC [77], OPASYN [55], CADICS [53], WATOPT [30], and STAIC [44]. The classical methods can be used with more complicated circuit models, including even full SPICE simulations in each iteration, as in DELIGHT.SPICE [74] (which uses the general purpose optimizer DELIGHT [75]) and ECSTASY [85].

The main advantage of these methods is the wide variety of problems they can handle; the only requirement is that the performance measures, along with one or more derivatives, can be computed. The main disadvantage of the classical optimization methods is they only find *locally optimal* designs. This means that the design is at least as good as neighboring designs, *i.e.*, small variations of any of the design parameters results in a worse (or infeasible) design. Unfortunately this does not mean the design is the best that can be achieved, *i.e.*, globally optimal; it is possible (and often happens) that some other set of design parameters, far away from the one found, is better. The same problem arises in determining feasibility: a classical (local) optimization method can fail to find a feasible design, even though one exists. Roughly speaking, classical methods can get stuck at local minima. This shortcoming is so well known that it is often not even mentioned in papers; it is taken as understood.

The problem of nonglobal solutions from classical optimization methods can be treated in several ways. The usual approach is to start the minimization method from many different initial designs, and to take the best final design found. Of course there are no guarantees that the globally optimal design has been found; this method merely increases the likelihood of finding the globally optimal design. This method 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. This method also requires human intervention (to give "good" initial designs), which makes the method less automated.

The classical methods become slow if complex models are used, as in DELIGHT.SPICE, which requires more than a complete SPICE run at each iteration ("more than" since, at the least, gradients must also be computed).

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 SEAS [73], DARWIN [57, 99]; systems based on fuzzy logic like FASY [91] and [45]; special heuristics based systems like IDAC [28, 29], OASYS [43], BLADES [21], and KANSYS [42].

One advantage of these methods is that there are few limitations on the types of problems, specifications, and performance measures that can be considered. Indeed, there are even fewer limitations than for classical optimization methods since many of these methods do not require the computation of derivatives.

These methods have several disadvantages. They find a locally optimal design (or, even just a "good" or "reasonable" design) instead of a globally optimal design. The final design depends on the initial design chosen and the algorithm parameters. As with classical optimization methods, infeasibility is not unambiguously detected; the method simply fails to find a feasible design (even when one may exist). These methods require substantial human intervention either during the design process, or during the training process.

Global optimization methods

Optimization methods that are guaranteed to find the globally optimal design have also been used in analog circuit design. The most widely known global optimization methods are branch and bound [103] and simulated annealing [100, 93].

A branch and bound method is used, for example, in [65]. Branch and bound methods unambiguously determine the globally optimal design: at each iteration they maintain a suboptimal feasible design and also a lower bound on the achievable performance. This enables the algorithm to terminate nonheuristically, *i.e.*, with complete confidence that the global design has been found within a given tolerance. The disadvantage of branch and bound methods is that they are extremely slow, with computation growing exponentially with problem size. Even problems with ten variables can be extremely challenging.

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 implementations there is no guarantee at all, since, for example, the cooling schedules called for in the theoretical treatments are not used in practice. Moreover, no real-time lower bound is available, so termination is heuristic. Like classical and knowledge-based methods, SA allows a very wide variety of performance measures and objectives to be handled. Indeed, SA is extremely effective for problems involving continuous variables and discrete variables, as in, *e.g.*, simultaneous amplifier topology and sizing problems. Simulated annealing has been used in several tools such as ASTR/OBLX [76], OPTIMAN [37] , FRIDGE [67], SAMM [105] and [14].

The main advantages of SA are that it handles discrete variables well, and greatly reduces the chances of finding a nonglobally optimal design. (Practical implementations do not reduce the chance to zero, however.) The main disadvantage is that it can be very slow, and cannot (in practice) guarantee a globally optimal solution.

Convex optimization and geometric programming methods

In this section we describe the general optimization method we employ in this paper: convex optimization. These are special optimization problems in which the objective and constraint functions are all convex.

While the theoretical properties of convex optimization problems have been appreciated for many years, the advantages in practice are only beginning to be appreciated now. The main reason is the development of extremely powerful interior-point methods for general convex optimization problems in the last five years (*e.g.*, [72, 102]). These methods can solve large problems, with thousands of variables and tens of thousands of constraints, very efficiently (in minutes on a small workstation). Problems involving tens of variables and hundreds of constraints (such as the ones we encounter in this paper) are considered small, and can be solved on a small current workstation in less than one second. The extreme efficiency of these methods is one of their great advantages.

The other main advantage is that the methods are truly global, *i.e.*, the global solution is *always* found, regardless of the starting point (which, indeed, need not be feasible). Infeasibility is unambiguously detected, *i.e.*, if the methods do not produce a feasible point they produce a certificate that proves the problem is infeasible. Also, the stopping criteria are completely nonheuristic: at each iteration a lower bound on the achievable performance is given.

One of the disadvantages is that the types of problems, performance specifications, and objectives that can be handled are far more restricted than any of the methods described above. This is the price that is paid for the advantages of extreme efficiency and global solutions. (For more on convex optimization, and the implications for engineering design, see [10].)

The contribution of this paper is to show how to formulate the analog amplifier design problem as a certain type of convex problem called geometric programming. The advantages, compared to the approaches described above, are extreme efficiency and global optimality. The disadvantage is less flexibility in the types of constraints we can handle, and the types of circuit models we can employ.

As far as we know, the only other application of geometric programming to circuit design is in transistor and wire sizing for Elmore delay minimization in digital circuits, as in TILOS [35] and other programs [86, 80, 81].

Their use of geometric programming can be distinguished from ours in several ways. First of all, the geometric programs that arise in Elmore delay minimization are very specialized (the only exponents that arise are 0 and ± 1). Second, the problems they encounter in practice are extremely large, involving up to hundreds of thousands of variables. Third, their representation of the problem as a geometric program is only approximate (since the actual circuits are nonlinear, and the threshold delay, not Elmore delay, is the true objective).

Convex optimization is mentioned in several papers on analog circuit CAD. The advantages of convex optimization are mentioned in [65, 64]. In [24, 25] the authors use a supporting hyperplane method, which they point out provides the global optimum if the feasible set is convex. In [88] the authors optimize a few design variables in an op-amp using a Lagrange multiplier method, which yields the global optimum since the small subproblems considered are convex. In [94, 95], convex optimization is used to optimize area, power, and dominant time constant in digital circuit wire and transistor sizing.

1.1.3 Outline of paper

In §1.2, we briefly describe geometric programming, the special type of optimization problem at the heart of the method, and show how it can be cast as a convex optimization problem. In §1.3–1.6 we describe a variety of constraints and performance measures, and show that they have the special form required for geometric programming. In §1.7 we give numerical examples of the design method, showing globally optimal trade-off curves among various performance measures such as bandwidth, power, and area. We also verify some of our designs using high fidelity SPICE models, and briefly discuss how our method can be extended to handle short-channel effects. In §1.9 we discuss robust design, *i.e.*, how to use the methods to ensure proper circuit operation under various processing conditions. In §1.10 we give our concluding remarks.

1.2 Geometric programming

Let $x_1, \ldots, x_n$ be n real, positive variables. We will denote the vector $(x_1, \ldots, x_n)$ of these variables as x. 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 $\alpha_{ij} \in \mathbf{R}$. Note that the coefficients c_k must be nonnegative, but the exponents α_{ij} can be any real numbers, including negative or fractional. When there is exactly one nonzero term in the sum, *i.e.*, $t = 1$ and $c_1 > 0$, we call f a *monomial* function. (This terminology is not consistent with the standard definition of a monomial in algebra, but it should not cause any confusion.) Thus, for example, $0.7 + 2x_1/x_3^2 + x_2^{0.3}$ is posynomial (but not monomial); $2.3(x_1/x_2)^{1.5}$ is a monomial (and, therefore, also a posynomial); while $2x_1/x_3^2 - x_2^{0.3}$ is neither. 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

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 1, \quad i = 1, \ldots, m, \\ & g_i(x) = 1, \quad i = 1, \ldots, p, \\ & x_i > 0, \quad i = 1, \ldots, n, \end{array} \tag{1.1}$$

where $f_1, \ldots, f_m$ are posynomial functions and $g_1, \ldots, g_p$ are monomial functions.

Several extensions are readily handled. 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 $f(x)/g(x) \leq 1$ (since f/g is posynomial). For example, we can handle constraints of the form $f(x) \leq a$, where f is posynomial and $a > 0$. 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 $g_1(x)/g_2(x) = 1$ (since g_1/g_2 is monomial).

We will also encounter functions whose reciprocals are posynomials. We say h is *inverse posynomial* if $1/h$ is a posynomial. If h is an inverse posynomial and f is a posynomial, then geometric programming can handle the constraint $f(x) \leq h(x)$ by writing it as $f(x)(1/h(x)) \leq 1$. As another example, if h is an inverse posynomial, then we can maximize it, by minimizing (the posynomial) $1/h$.

Geometric programming has been known and used since the late 1960s, in various fields. There were two early books on geometric programming, by Duffin, Peterson, and Zener [18] and Zener [106], which include the basic theory, some electrical engineering applications (*e.g.*, optimal transformer design), but not much on numerical solution methods. Another book appeared in 1976 [9]. The 1980 survey paper by Ecker [19] has many references on applications and methods, including numerical solution methods used at that time. Geometric programming is briefly described in some surveys of optimization, *e.g.*, [20, p326-328] or [98, Ch.4]. While geometric programming is certainly known, it is nowhere near as widely known as, say, linear programming. In addition, advances in general purpose nonlinear constrained optimization algorithms and codes (such as the ones described above) have contributed to decreased use (and knowledge) of geometric programming in recent years.

1.2.1 Geometric programming in convex form

A geometric program can be reformulated as a *convex optimization problem*, *i.e.*, the problem of minimizing a convex function subject to convex inequality constraints and linear equality constraints. This is the key to our ability to globally and efficiently solve geometric programs. We define new variables $y_i = \log x_i$, and take the logarithm of a posynomial f to get

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

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 new variable y: for all $y, z \in \mathbf{R}^n$ and $0 \leq \lambda \leq 1$ we have

$$h(\lambda y + (1-\lambda)z) \leq \lambda h(y) + (1-\lambda)h(z).$$

Note that if the posynomial f is a monomial, then the transformed function h is affine, *i.e.*, a linear function plus a constant.

We can convert the standard geometric program (1.1) into a convex program by expressing it as

$$\begin{array}{ll} \text{minimize} & \log f_0(e^{y_1}, \ldots, e^{y_n}) \\ \text{subject to} & \log f_i(e^{y_1}, \ldots, e^{y_n}) \leq 0, \quad i = 1, \ldots, m \\ & \log g_i(e^{y_1}, \ldots, e^{y_n}) = 0, \quad i = 1, \ldots, p. \end{array} \tag{1.2}$$

This is the so-called *convex form* of the geometric program (1.1). Convexity of the convex form geometric program (1.2) has several important implications: we can use efficient interior-point methods to solve them, and there is a complete and useful duality, or sensitivity theory for them; see, *e.g.*, [10].

1.2.2 Solving geometric programs

Since Ecker's survey paper there have been several important developments, related to solving geometric programming in the convex form. A huge improvement in computational efficiency was achieved in 1994, when Nesterov and Nemirovsky developed efficient interior-point algorithms to solve a variety of nonlinear optimization problems, including geometric programs [72]. Recently, Kortanek et al. have shown how the most sophisticated primal-dual interior-point methods used in linear programming can be extended to geometric programming, resulting in an algorithm approaching the efficiency of current interior-point linear programming solvers [56]. The algorithm they describe has the desirable feature of exploiting *sparsity* in the problem, *i.e.*, efficiently handling problems in which each variable appears in only a few constraints. Other methods developed specifically for geometric programs include those described by Avriel et al. [7] and Rajpogal and Bricker [79], which require solving a sequence of linear programs (for which very efficient algorithms are known).

The algorithms described above are specially tailored for the geometric program (in convex form). It is also possible to solve the convex form problem using general purpose optimization codes that handle smooth objectives and constraint functions, *e.g.*, LANCELOT [16], MINOS [70], LOQO [96], or LINGO-NL [82]. These codes will (in principle) find a globally optimal solution, since the convex form problem is convex. They will also determine the optimal dual variables (sensitivities) as a by-product of solving the problem. In an unpublished report [104], Xu compares the performance of the sophisticated primal-dual interior-point method developed by Kortanek, Xu, and Ye (XGP, [56]) with two general purpose optimizers, MINOS and LINGO-NL, on a suite of standard geometric programming problems (in convex form). The general purpose codes fail to solve some of the problems, and in all cases take substantially longer to obtain the solution.

For our purposes, the most important feature of geometric programs is that they can be *globally* solved with *great* efficiency. Problems with hundreds of variables and thousands of constraints are readily handled, on a small workstation, in minutes; the problems we encounter in this paper, which have on the order of 10 variables and 100 constraints, are easily solved in under one second.

To carry out the designs in this paper, we implemented, in MATLAB, a simple and crude primal barrier method for solving the convex form problem. Roughly speaking, this method consists of applying a modified Newton's method to minimizing the smooth convex function

$$t \log f_0(e^{y_1}, \ldots, e^{y_n}) + \sum_{i=1}^{m} \log\left(-\log f_i(e^{y_1}, \ldots, e^{y_n})\right)$$

subject to the affine (linear equality) constraints $\log g_i(e^{y_1}, \ldots, e^{y_n}) = 0$, $i = 1, \ldots, p$, for an increasing sequence of values of t, starting from the optimal y found for the last value of t. It can be shown that when $t \geq m/\epsilon$, the optimal solution of this problem is no more than ϵ suboptimal for the original convex form GP. For much more detail, see, *e.g.*, [10, 34].

Despite the simplicity of the algorithm (*i.e.*, primal only, with no sparsity exploited) and the overhead of an interpreted language, the geometric programs arising in this paper were all solved in approximately one or two seconds on an ULTRA SPARC1 running at 170 MHz. Since our simple interior-point method is already extremely fast on the relatively small problems we encounter in this paper, we feel that the choice of algorithm is not critical. When the method is applied to large-scale problems, such as the ones obtained for a robust design problem (see §1.9), the choice may well become critical.

1.2.3 Sensitivity analysis

Suppose we modify the right-hand sides of the constraints in the geometric program (1.1) as follows:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \le e^{u_i}, \quad i = 1, \dots, m, \\ & g_i(x) = e^{v_i}, \quad i = 1, \dots, p, \\ & x_i > 0, \quad i = 1, \dots, n. \end{array} \tag{1.3}$$

If all of the u_i and v_i are zero, this modified geometric program coincides with the original one. If $u_i < 0$, then the constraint $f_i(x) \le e_i^u$ represents a *tightened* version of the original ith constraint $f_i(x) \le 1$; conversely if $u_i > 0$, it represents a *loosening* of the constraint. Note that u_i gives a logarithmic or fractional measure of the change in the specification: $u_i = 0.0953$ means that the ith constraint is loosened 10%, whereas $u_i = -0.0953$ means that the ith constraint is tightened 10%.

Let $f_0^*(u, v)$ denote the optimal objective value of the modified geometric program (1.3), as a function of the parameters $u = (u_1, \dots, u_m)$ and $v = (v_1, \dots, v_p)$, so the original objective value is $f_0^*(0, 0)$. In *sensitivity analysis*, we study the variation of f_0^* as a function of u and v, for small u and v. To express the change in optimal objective function in a fractional form, we use the *logarithmic sensitivities*

$$S_i = \frac{\partial \log f_0^*}{\partial u_i}, \quad T_i = \frac{\partial \log f_0^*}{\partial v_i}, \tag{1.4}$$

evaluated at $u = 0$, $v = 0$. These sensitivity numbers are dimensionless, since they express fractional changes per fractional change.

For simplicity we are assuming here that the original geometric program is feasible, and remains so for small changes in the right-hand sides of the constraints, and also that the optimal objective value is differentiable as a function of u_i and v_i. More complete descriptions of sensitivity analysis in other cases can be found in the references cited above, or in a general context in [10]. The surprising part is that the sensitivity numbers $S_1, \dots, S_m$ and $T_1, \dots, T_p$ come for free, when the problem is solved using an interior-point or Lagrangian-based method (from the solution of the dual problem; see [10]).

We start with some simple observations. If at the optimal solution x^* of the original problem, the ith inequality constraint is not active, *i.e.*, $f_i(x^*)$ is strictly less than one, then $S_i = 0$ (since we can slightly tighten or loosen the ith constraint with no effect). We always have $S_i \le 0$, since increasing u_i slightly loosens the constraints, and hence lowers the optimal objective value. The sign of T_i tells us whether increasing the right-hand side side of the equality constraint $g_i = 1$ increases or decreases the optimal objective value.

The sensitivity numbers are extremely useful in practice, and give tremendous insight to the designer. Suppose, for example, that the objective f_0 is power dissipation, $f_1(x) \leq 1$ represents the constraint that the bandwidth is at least 30MHz, and $g_1(x) = 1$ represents the constraint that the open-loop gain is 10^5V/V. Then $S_1 = -3$, say, tells us that a small fractional increase in required bandwidth will translate into a three times larger fractional increase in power dissipation. $T_1 = 0.1$ tells us that a small fractional increase in required open-loop gain will translate into a fractional increase in power dissipation only one-tenth as big. Although both constraints are active, the sensitivities tell us that the design is, roughly speaking, more tightly constrained by the bandwidth constraint than the open-loop gain constraint. The sensitivity information from the example above might lead the designer to reduce the required bandwidth (to reduce power), or perhaps increase the open-loop gain (since it won't cost much). We give an example of sensitivity analysis in §1.7.4.

1.3 Dimension constraints

We start by considering some very basic constraints involving the device dimensions, *e.g.*, symmetry, matching, minimum or maximum dimensions, and area limits.

1.3.1 Symmetry and matching

For the intended operation of the input differential pair, transistors M_1 and M_2 must be identical and transistors M_3 and M_4 must also be identical. These conditions translate into the four equality constraints

$$W_1 = W_2, \quad L_1 = L_2, \quad W_3 = W_4, \quad L_3 = L_4. \tag{1.5}$$

The biasing transistors M_5, M_7, and M_8 must match, *i.e.*, have the same length:

$$L_5 = L_7 = L_8. \tag{1.6}$$

The six equality constraints in (1.5) and (1.6) have monomial expressions on the left and right hand sides, hence are readily handled in geometric programming (by expressing them as monomial equality constraints such as $W_1/W_2 = 1$).

Note that equations (1.5) and (1.6) effectively reduce the number of variables from 18 to 12. We can, for example, eliminate the variables L_7 and L_8 by substituting L_5 wherever they appear. For clarity we will continue to use the variables L_7 and L_8 in our discussion; for computational purposes, however, they can be replaced by L_5. (In any case the number of variables and constraints is so small for a geometric program, that there is almost no computational penalty in keeping the extra variables and equality constraints.)

1.3.2 Limits on device sizes

Lithography limitations and layout rules impose minimum (and possibly maximum) sizes on the transistors:

$$L_{\min} \leq L_i \leq L_{\max}, \quad W_{\min} \leq W_i \leq W_{\max}, \quad i = 1, \ldots, 8. \tag{1.7}$$

These 32 constraints can be expressed as posynomial constraints such as $L_{\min}/L_1 \leq 1$, etc. Since L_i and W_i are variables (hence, monomials), we can also fix certain devices sizes, *i.e.*, impose equality constraints.

We should note that a constraint limiting device dimensions to a finite number of allowed values, or to an integer multiple of some fixed small value, *cannot* be (directly) handled by geometric programming. Such constraints can be approximately handled by simple rounding to an allowed value, or using more sophisticated integer programming methods.

1.3.3 Area

The op-amp die area A can be approximated as a constant plus the sum of transistor and capacitor area as

$$A = \alpha_0 + \alpha_1 C_c + \alpha_2 \sum_{i=1}^{8} W_i L_i. \tag{1.8}$$

Here $\alpha_0 \geq 0$ gives the fixed area, $\alpha_1 > 1$ is the ratio of capacitor area to capacitance, and the constant $\alpha_2 > 1$ (if it is not one) can take into account wiring in the drain and source area. This expression for the area is a posynomial function of the design parameters, so we can impose an upper bound on the area, *i.e.*, $A \leq A_{\max}$, or use the area as the objective to be minimized. More accurate posynomial formulas for the amplifier die area could be developed, if needed.

1.3.4 Systematic input offset voltage

To reduce input offset voltage, the drain voltages of M_3 and M_4 must be equal, ensuring that the current I_5 is split equally between transistors M_1 and M_2. This happens when the current densities of M_3, M_4, and M_6 are equal, *i.e.*,

$$\frac{W_3/L_3}{W_6/L_6} = \frac{W_4/L_4}{W_6/L_6} = \frac{1}{2}\frac{W_5/L_5}{W_7/L_7}. \tag{1.9}$$

These two conditions are equality constraints between monomials, and are therefore readily handled by geometric programming.

1.4 Bias conditions, signal swing, and power constraints

In this section we consider constraints involving bias conditions, including the effects of common-mode input voltage and output signal swing. We also consider the quiescent power of the op-amp (which is determined, of course, by the bias conditions). In deriving these constraints, we assume that the symmetry and matching conditions (1.5) and (1.6) hold. To derive the equations we use a standard long channel, square-law model for the MOS transistors, which is described in detail in §1.A. We refer to this model as the GP0 model; the same analysis also applies to the more accurate GP1 model, also described in the appendix.

In order to simplify the equations, it is convenient to define the bias currents I_1, I_5, and I_7 through transistors M_1, M_5 and M_7, respectively. Transistors M_5 and M_7 form a current mirror with transistor M_8. Their currents are given by

$$I_5 = \frac{W_5 L_8}{L_5 W_8} I_{\text{bias}}, \quad I_7 = \frac{W_7 L_8}{L_7 W_8} I_{\text{bias}}. \tag{1.10}$$

Thus I_5 and I_7 are monomials in the design variables. The current through transistor M_5 is split equally between transistor M_1 and M_2. Thus we have

$$I_1 = \frac{I_5}{2} = \frac{W_5 L_8}{2 L_5 W_8} I_{\text{bias}}, \tag{1.11}$$

which is another monomial.

Since these bias currents are monomials, we can include lower or upper bounds on them, or even equality constraints, if we wish. We will use I_1, I_5, and I_7 in order to express other constraints, remembering that these bias currents can simply be eliminated (*i.e.*, expressed directly in terms of the design variables) using (1.10) and (1.11).

1.4.1 Bias conditions

The setup for deriving the bias conditions is as follows. The input terminals are at the same DC potential, the common-mode input voltage V_{cm}. We assume that the common-mode input voltage is allowed to range between a minimum value $V_{\text{cm,min}}$ and a maximum value $V_{\text{cm,max}}$, which are given. Similarly, we assume that the output voltage is allowed to swing between a minimum value $V_{\text{out,min}}$ and a maximum value $V_{\text{out,max}}$ (which takes into account large signal swings in the output).

The bias conditions are that each transistor $M_1, \ldots, M_8$ should remain in saturation for all possible values of the input common-mode voltage and the output voltage. The derivation of the bias constraints given below can be found in §1.B. The important point here is that the constraints are each posynomial inequalities on the design variables, and hence can be handled by geometric programming.

- **Transistor** M_1**.** The lowest common-mode input voltage, $V_{\rm cm,min}$, imposes the toughest constraint on transistor M_1 remaining in saturation. The condition is:

$$\sqrt{\frac{I_1 L_3}{\mu_{\rm n} C_{\rm ox}/2W_3}} \le V_{\rm cm,min} - V_{\rm ss} - V_{\rm TP} - V_{\rm TN}. \tag{1.12}$$

  Note that if the righthand side of equation (1.12) were negative, *i.e.*, if $V_{\rm cm,min} \le V_{\rm ss} + V_{\rm TP} + V_{\rm TN}$, then the design is immediately known to be infeasible (since the lefthand side is, of course, positive).

- **Transistor** M_2**.** The systematic offset condition (1.9) makes the drain voltage of M_1 equal to the drain voltage of M_2. Therefore, the condition for M_2 being saturated is the same as the condition for M_1 being saturated, *i.e.*, (1.12). Note that the minimum allowable value of $V_{\rm cm,min}$ is determined by M_1 and M_2 entering the linear region.

- **Transistor** M_3**.** Since $V_{\rm gd,3} = 0$ transistor M_3 is always in saturation and no additional constraint is necessary.

- **Transistor** M_4**.** The systematic offset condition also implies that the drain voltage of M_4 is equal to the drain voltage of M_3. Thus M_4 will be saturated as well.

- **Transistor** M_5**.** The highest common-mode input voltage, $V_{\rm cm,max}$, imposes the tightest constraint on transistor M_5 being in saturation. The condition is:

$$\sqrt{\frac{I_1 L_1}{\mu_{\rm p} C_{\rm ox}/2W_1}} + \sqrt{\frac{I_5 L_5}{\mu_p C_{\rm ox}/2W_5}} \le V_{\rm dd} - V_{\rm cm,max} + V_{\rm TP}. \tag{1.13}$$

  Thus, the maximum allowable value of $V_{\rm cm,min}$ is determined by M_5 entering the linear region. As explained above, if the righthand side of equation (1.13) is negative, *i.e.*, $V_{\rm cm,max} \ge V_{\rm dd} + V_{\rm TP}$, then the design is obviously infeasible.

- **Transistor** M_6**.** The most stringent condition occurs when the output voltage is at its minimum value $V_{\rm out,min}$:

$$\sqrt{\frac{I_7 L_6}{\mu_{\rm n} C_{\rm ox}/2W_6}} \le V_{\rm out,min} - V_{\rm ss}. \tag{1.14}$$

  In this case the righthand side of equation (1.14) will not be negative if we assume the minimum output voltage is above the negative supply voltage.

- **Transistor** M_7**.** For M_7, the most stringent condition occurs when the output voltage is at its maximum value $V_{\mathrm{out,max}}$:

$$\sqrt{\frac{I_7 L_7}{\mu_{\mathrm{p}} C_{\mathrm{ox}}/2W_7}} \leq V_{\mathrm{dd}} - V_{\mathrm{out,max}}. \tag{1.15}$$

  Here too the right hand side of equation (1.15) will be positive assuming the maximum output voltage is below the positive supply voltage.

- **Transistor** M_8**.** Since $V_{\mathrm{gd},8} = 0$, transistor M_8 is always in saturation; no additional constraint is necessary.

In summary, the requirement that all transistors remain in saturation for all values of common-mode input voltage between $V_{\mathrm{cm,min}}$ and $V_{\mathrm{cm,max}}$, and all values of output voltage between $V_{\mathrm{out,min}}$ and $V_{\mathrm{out,max}}$, is given by the four inequalities (1.12), (1.13), (1.14), and (1.15). These are complicated, but *posynomial* constraints on the design parameters.

1.4.2 Gate overdrive

It is sometimes desirable to operate the transistors with a minimum gate overdrive voltage. This ensures that they operate away from the subthreshold region, and also improves matching between transistors. For any given transistor this constraint can be expressed as

$$V_{\mathrm{gs}} - V_{\mathrm{T}} = \sqrt{\frac{I_D L}{\mu C_{\mathrm{ox}}/2\ W}} \geq V_{\mathrm{overdrive,min}}. \tag{1.16}$$

The expression on the left is a monomial, so we can also impose an upper bound on it, or an equality constraint, if we wish. (We will see in §1.9 that robustness to process variations can be dealt with in a more direct way.)

1.4.3 Quiescent power

The quiescent power of the op-amp is given by

$$P = (V_{\mathrm{dd}} - V_{\mathrm{ss}})\,(I_{\mathrm{bias}} + I_5 + I_7)\,, \tag{1.17}$$

which is a posynomial function of the design parameters. Hence we can impose an upper bound on P, or use it as the objective to be minimized.

1.5 Small signal transfer function constraints

1.5.1 Small signal transfer function

We now assume that the symmetry, matching, and bias constraints are satisfied, and consider the (small signal) transfer function H from a differential input source to the output. To derive the transfer function H, we use a standard small signal model for the transistors, which is described in §1.A.2. The standard value of the compensation resistor is used, *i.e.*,

$$R_c = 1/g_{\mathrm{m6}} \tag{1.18}$$

(see [2]).

The transfer function can be well approximated by a four pole form

$$H(s) = A_v \frac{1}{(1+s/p_1)(1+s/p_2)(1+s/p_3)(1+s/p_4)}. \tag{1.19}$$

Here A_v is the open-loop voltage gain, $-p_1$ is the dominant pole, $-p_2$ is the output pole, $-p_3$ is the mirror pole, and $-p_4$ is the pole arising from the compensation circuit. In order to simplify the discussion in the sequel, we will refer to $p_1 \ldots, p_4$, which are positive, as the poles (whereas precisely speaking, the poles are $-p_1 \ldots, -p_4$).

We now give the expressions for the gain and poles. The two-stage op-amp has been previously analyzed by many authors [87, 31, 52]. The compensation scheme has also been analyzed previously [2, 101]. A complete derivation of the next results can be found in [101].

- The open-loop voltage gain is

$$A_v = \left(\frac{g_{\mathrm{m2}}}{g_{\mathrm{o2}}+g_{\mathrm{o4}}}\right)\left(\frac{g_{\mathrm{m6}}}{g_{\mathrm{o6}}+g_{\mathrm{o7}}}\right) = \frac{2\mathrm{C_{ox}}}{(\lambda_{\mathrm{n}}+\lambda_{\mathrm{p}})^2}\sqrt{\mu_{\mathrm{n}}\mu_{\mathrm{p}}\frac{W_2W_6}{L_2L_6I_1I_7}}, \tag{1.20}$$

  which is monomial function of the design parameters.

- The dominant pole is accurately given by

$$p_1 = \frac{g_{\mathrm{m1}}}{A_vC_c}. \tag{1.21}$$

  Since A_v and g_{m1} are monomials, and C_c is a design variable, p_1 is a monomial function of the design variables.

- The output pole p_2 is given by

$$p_2 = \frac{g_{\mathrm{m6}}C_c}{C_1C_c + C_1C_{\mathrm{TL}} + C_cC_{\mathrm{TL}}}, \tag{1.22}$$

where C_1, the capacitance at the gate of M_6, can be expressed as

$$C_1 = C_{\rm gs6} + C_{\rm db2} + C_{\rm db4} + C_{\rm gd2} + C_{\rm gd4}. \tag{1.23}$$

and $C_{\rm L}$, the total capacitance at the output node, can be expressed as

$$C_{\rm TL} = C_{\rm L} + C_{\rm db6} + C_{\rm db7} + C_{\rm gd6} + C_{\rm gd7}. \tag{1.24}$$

The meanings of these parameters, and their dependence on the design variables, is given in the appendix, in §1.A.2. The important point here is that p_2 is an inverse posynomial function of the design parameters (*i.e.*, $1/p_2$ is a posynomial).

- The mirror pole p_3 is given by

$$p_3 = \frac{g_{\rm m3}}{C_2}, \tag{1.25}$$

where C_2, the capacitance at the gate of M_3, can be expressed as

$$C_2 = C_{\rm gs3} + C_{\rm gs4} + C_{\rm db1} + C_{\rm db3} + C_{\rm gd1}. \tag{1.26}$$

Thus, p_3 is also an inverse posynomial.

- The compensation pole is

$$p_4 = \frac{g_{\rm m6}}{C_1}, \tag{1.27}$$

which is also inverse posynomial.

In summary: the open-loop gain A_v and the dominant pole p_1 are monomial, and the parasitic poles p_2, p_3, and p_4 are all inverse posynomials. Now we turn to various design constraints and specifications that involve the transfer function.

1.5.2 Open-loop gain constraints

Since the open-loop gain A_v is a monomial, we can constrain it to equal some desired value $A_{\rm des}$. We could also impose upper or lower bounds on the gain, as in

$$A_{\rm min} \le A_v \le A_{\rm max}, \tag{1.28}$$

where $A_{\rm min}$ and $A_{\rm max}$ are given lower and upper limits on acceptable open-loop gain.

1.5.3 Minimum gain at a frequency

The magnitude squared of the transfer function at a frequency ω_0 is given by

$$|H(j\omega_0)|^2 = \frac{A_v^2}{\prod_{i=1}^{4}(1+\omega_0^2/p_i^2)}.$$

Since p_i are all inverse posynomial, the expressions ω_0^2/p_i^2 are posynomial. Hence the whole denominator is posynomial. The numerator is monomial, so we conclude that the squared magnitude of the transfer function, $|H(j\omega_0)|^2$, is inverse posynomial. (Indeed, it is inverse posynomial in the design variables and ω_0 as well.) We can therefore impose any constraint of the form

$$|H(j\omega_0)| \geq a$$

using geometric programming (by expressing it as $a^2/|H(j\omega_0)|^2 \leq 1$).

The transfer function magnitude $|H(j\omega)|$ decreases as ω increases (since it has only poles), so $|H(j\omega_0)| \geq a$ is equivalent to

$$|H(j\omega)| \geq a \quad \text{for} \quad \omega \leq \omega_0. \tag{1.29}$$

We will see below that this allows us to specify a minimum bandwidth or crossover frequency.

1.5.4 3dB bandwidth

The 3dB bandwidth $\omega_{3\text{dB}}$ is the frequency at which the gain drops 3dB below the DC open-loop gain, *i.e.*, $|H(j\omega_{3\text{dB}})| = A_v/\sqrt{2}$. To specify that the 3dB bandwidth is at least some minimum value $\omega_{3\text{dB,min}}$, *i.e.*, $\omega_{3\text{dB}} \geq \omega_{3\text{dB,min}}$, is equivalent to specifying that $|H(\omega_{3\text{dB,min}})| \geq A_v/\sqrt{2}$. This is turn can be expressed as

$$A_v/|H(\omega_{3\text{dB,min}})|^2 \leq 2, \tag{1.30}$$

which is a posynomial inequality.

In almost all designs p_1 will be the dominant pole, (see below) so the 3dB bandwidth is very accurately given by

$$\omega_{3\text{dB}} = p_1 = \frac{g_{\text{m1}}}{A_v C_c}, \tag{1.31}$$

which is a monomial. Using this (extremely accurate) approximation, we can constrain the 3dB bandwidth to equal some required value. Using the constraint (1.30), which is exact but inverse posynomial, we can constrain the 3dB bandwidth to exceed a given minimum value.

1.5.5 Dominant pole conditions

The amplifier is intended to operate with p_1 as the dominant pole, *i.e.*, p_1 much smaller than p_2, p_3, and p_4. These conditions can be expressed as

$$\frac{p_1}{p_2} \leq 0.1, \quad \frac{p_1}{p_3} \leq 0.1, \quad \frac{p_1}{p_4} \leq 0.1, \tag{1.32}$$

where we (arbitrarily) use one decade, *i.e.*, a factor of 10 in frequency, as the condition for dominance. These dominant pole conditions are readily handled by geometric programming, since p_1 is monomial and p_2, p_3, and p_4 are all inverse posynomial. In fact these dominant pole conditions usually do not need to be included explicitly since the phase margin conditions described below are generally more strict, and describe the real design constraint. Nevertheless it is common practice to impose a minimum ratio between the dominant and non-dominant poles; see, for example, [41].

1.5.6 Unity-gain bandwidth and phase margin

We define the unity-gain bandwidth ω_c as the frequency at which $|H(j\omega_c)| = 1$. The phase margin is defined in terms of the phase of the transfer function at the unity-gain bandwidth:

$$\mathrm{PM} = \pi - \angle H(j\omega_c) = \pi - \sum_{i=1}^{4} \arctan\left(\frac{\omega_c}{p_i}\right).$$

A phase margin constraint specifies a lower bound on the phase margin, typically between 30° and 60°.

The unity-gain bandwidth and phase margin are related to the closed-loop bandwidth and stability of the amplifier with unity-gain feedback, *i.e.*, when its output is connected to the inverting input. If the op-amp is to be used in some other specific closed-loop configuration, then a different frequency will be of more interest but the analysis is the same. For example, if the op-amp is to be used in a feedback configuration with closed-loop gain +20dB, then the critical frequency is the 20dB crossover point, *i.e.*, the frequency at which the open-loop gain drops to 20dB, and the phase margin is defined at that frequency. All of the analysis below is readily adapted with minimal changes to such a situation. For simplicity, we continue the discussion for the unity-gain bandwidth.

We start by considering a constraint that the unity-gain bandwidth should exceed a given minimum frequency, *i.e.*,

$$\omega_c \geq \omega_{c,\min}. \tag{1.33}$$

This constraint is just a minimum gain constraint at the frequency $\omega_{c,\min}$ (as in (1.29)), and so can be handled exactly by geometric programming as a posynomial inequality.

Here too we can develop an approximate expression for the unity-gain bandwidth which is monomial. If we assume the parasitic poles p_2, p_3, and p_4 are at least a bit (say, an octave) above the unity-gain bandwidth, then the unity-gain bandwidth can be approximated as the open-loop gain times the 3dB bandwidth, *i.e.*,

$$\omega_{c,\text{approx}} = \frac{g_{\text{m1}}}{C_c}, \tag{1.34}$$

which is a monomial. If we use this approximate expression for the unity-gain bandwidth, we can fix the unity-gain bandwidth at a desired value. The approximation (1.34) ignores the decrease in gain due to the parasitic poles, and consequently overestimates the actual unity-gain bandwidth (*i.e.*, the gain drops to 0dB at a frequency slightly less than $\omega_{c,\text{approx}}$).

We now turn to the phase margin constraint, for which we can give a very accurate posynomial approximation. Assuming the open-loop gain exceeds 10 or so, the phase contributed by the dominant pole at the unity-gain bandwidth, *i.e.*, $\arctan(\omega_c/p_1)$, will be very nearly 90°. Therefore the phase margin constraint can be expressed as

$$\sum_{i=2}^{4} \arctan\left(\frac{\omega_c}{p_i}\right) \leq \frac{\pi}{2} - \text{PM}, \tag{1.35}$$

i.e., the nondominant poles cannot contribute more than $90° - \text{PM}$ total phase shift.

The phase margin constraint (1.35) cannot be exactly handled by geometric programming, so we use two reasonable approximations to form a posynomial approximation. The first is an approximate unity-gain bandwidth $\omega_{c,\text{approx}}$ (from (1.34)) instead of the exact unity-gain bandwidth ω_c as the frequency at which we will constrain the phase of H. As mentioned above, we have $\omega_{c,\text{approx}} \leq \omega_c$, so our specification is a bit stronger than the exact phase margin specification (since we are constraining the phase at a frequency slightly above the actual unity gain bandwidth). We will also approximate $\arctan(x)$ as a monomial. A simple approximation is given by $\arctan(x) \approx x$, which is quite accurate for $\arctan(x)$ less than 25°. Thus, assuming that each of the parasitic poles contributes no more than about 25° of phase shift, we can approximate the phase margin constraint accurately as

$$\sum_{i=2}^{4} \frac{\omega_{c,\text{approx}}}{p_i} \leq \frac{\pi}{2} - \text{PM}_{\text{min}}, \tag{1.36}$$

which is a posynomial inequality in the design variables (since $\omega_{c,\text{approx}}$ is monomial). The approximation error involved here is almost always very small for the following reasons. The constraint (1.36) makes sure none of the nondominant poles is too near ω_c. This, in turn, validates our approximation $\omega_{c,\mathrm{approx}} \approx \omega_c$. It also ensures that our approximation that the phase contributed by the nondominant poles is $\sum_{i=2}^{4} \omega_c/p_i$ is good.

Finally we note that it is possible to obtain a more accurate monomial approximation of $\arctan(x)$ that has less error over a wider range, *e.g.*, $\arctan(x) \leq 60°$. For example the approximation $\arctan(x) \approx 0.75x^{0.7}$ gives a fit around $\pm 3°$ for angles between 0 and $60°$, as shown in Figure 1.2.

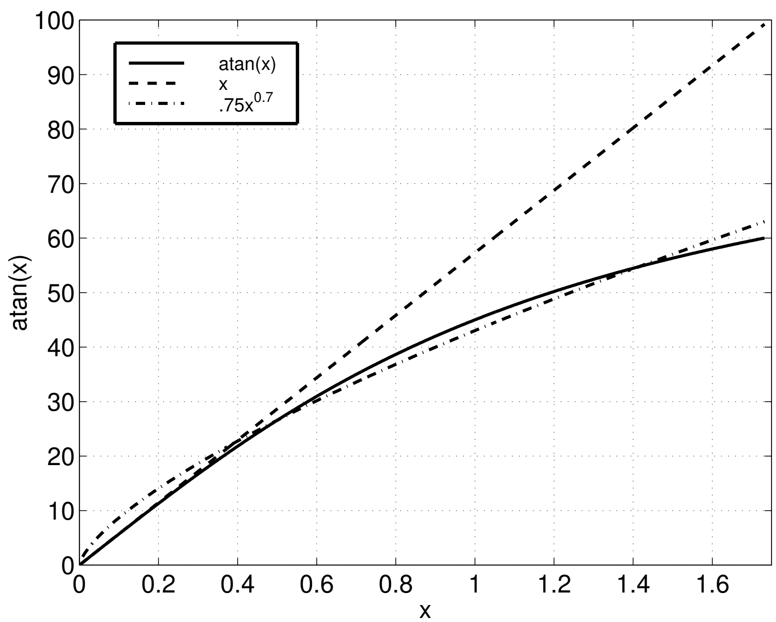

FIGURE 1.2. Approximations of $\arctan(x)$

1.6 Other constraints

In this section we collect several other important constraints.

1.6.1 Slew rate

The slew rate can be expressed [78] as

$$\mathrm{SR} = \min\{2I_1/C_c,\ I_7/(C_c + C_{\mathrm{TL}})\}.$$

In order to ensure a minimum slew rate $\mathrm{SR}_{\min}$ we can impose the two constraints

$$\frac{C_c}{2I_1} \leq \frac{1}{\mathrm{SR}_{\min}}, \quad \frac{C_c + C_{\mathrm{TL}}}{I_7} \leq \frac{1}{\mathrm{SR}_{\min}}. \tag{1.37}$$

These two constraints are posynomial.

1.6.2 Common-mode rejection ratio

The common-mode rejection ratio (CMRR) can be approximated as

$$\mathrm{CMRR} = \frac{2g_{m1}g_{m3}}{(g_{o3} + g_{o1})\, g_{o5}} = \frac{2\mathrm{C}_{\mathrm{ox}}}{(\lambda_{\mathrm{n}} + \lambda_{\mathrm{p}})\, \lambda_{\mathrm{p}}} \sqrt{\mu_{\mathrm{n}}\mu_{\mathrm{p}} \frac{W_1 W_3}{L_1 L_3 I_5^2}}, \tag{1.38}$$

which is a monomial (see [101]). In particular, we can specify a minimum acceptable value of CMRR.

1.6.3 Power supply rejection ratio

Negative power supply rejection ratio

The negative power supply rejection ratio (PSRR) is given by (see [58, 51])

$$\frac{g_{m2}g_{m6}}{(g_{o2} + g_{o4})\, g_{o6}} \frac{1}{(1 + s/p_1)(1 + s/p_2)}. \tag{1.39}$$

Thus, the low-frequency negative PSRR is given by the inverse posynomial expression

$$\frac{g_{m2}g_{m6}}{(g_{o2} + g_{o4})\, g_{o6}}, \tag{1.40}$$

which, therefore, can be lower bounded.

The high-frequency PSRR characteristics are generally more critical than the low-frequency PSRR characteristics since noise in mixed-mode chips (clock noise, switching regulator noise, etc.) is typically high frequency. One can see that the expression for the magnitude squared of the negative PSRR at a frequency ω_0 has the form

$$|\mathrm{PSRR}(j\omega_0)|^2 = \frac{A_p^2}{(1 + \omega_0^2/p_1^2)(1 + \omega_0^2/p_2^2)},$$

where A_p, p_1 and p_2 are given by inverse posynomial expressions. As in §1.5.3, we can impose a lower bound on the negative PSRR at frequencies smaller than the unity-gain bandwidth by imposing posynomial constraints of the form

$$|\mathrm{PSRR}(j\omega_0)| \geq a. \tag{1.41}$$

Positive power supply rejection ratio

The low-frequency positive power supply rejection ratio is given by

$$\mathrm{PSRR} = \frac{2g_{m2}g_{m3}g_{m6}}{(g_{o2}+g_{o4})\,(2g_{m3}g_{o7}-g_{m6}g_{o5})} \tag{1.42}$$

which is neither posynomial nor inverse-posynomial, so it follows that constraints on the positive power supply rejection cannot be handled by geometric programming. However, this op-amp suffers from much worse negative PSRR characteristics than positive PSRR characteristics, both at low and high frequencies (see [39, 41]). Therefore, not constraining the positive PSRR is not critical.

At the least, we must check the positive PSRR of any design carried out by the method described in this paper. (It is more than adequate in every design we have carried out.) However, if the positive PSRR specification becomes critical, it can be approximated (conservatively) by a posynomial inequality, for example, using Duffin linearization [17, 7].

1.6.4 Noise performance

The equivalent input-referred noise power spectral density $S_{\mathrm{in}}(f)^2$ (in V^2/Hz, at frequency f assumed smaller than the 3dB bandwidth), can be expressed as

$$S_{\mathrm{in}}^2 = S_1^2 + S_2^2 + \left(\frac{g_{\mathrm{m3}}}{g_{\mathrm{m1}}}\right)^2 \left(S_3^2+S_4^2\right),$$

where S_k^2 is the input-referred noise power spectral density of transistor M_k. These spectral densities consist of the input-referred thermal noise and a $1/f$ noise:

$$S_k(f)^2 = \left(\frac{2}{3}\right)\frac{4kT}{g_{\mathrm{m,k}}} + \frac{K_{\mathrm{f}}}{C_{\mathrm{ox}}W_kL_kf}.$$

Thus the input-referred noise spectral density can be expressed as

$$S_{\mathrm{in}}(f)^2 = \alpha/f + \beta,$$

where

$$\alpha = \frac{2K_p}{C_{\mathrm{ox}}W_1L_1}\left(1+\frac{K_n\mu_{\mathrm{n}}L_1^2}{K_p\mu_{\mathrm{p}}L_3^2}\right),$$

$$\beta = \frac{16KT}{3\sqrt{2\mu_{\mathrm{p}}C_{\mathrm{ox}}(W/L)_1I_1}}\left(1+\sqrt{\frac{\mu_{\mathrm{n}}(W/L)_3}{\mu_{\mathrm{p}}(W/L)_1}}\right).$$

Note that α and β are (complicated) posynomial functions of the design parameters.

We can therefore impose spot noise constraints, *i.e.*, require that

$$S_{\mathrm{in}}(f)^2 \leq S_{\mathrm{max}}^2, \tag{1.43}$$

for a certain f, as a posynomial inequality. (We can impose multiple spot noise constraints, at different frequencies, as multiple posynomial inequalities.)

The total RMS noise level V_{noise} over a frequency band $[f_0, f_1]$ (where f_1 is below the equivalent noise bandwidth of the circuit) can be found by integrating the noise spectral density:

$$V_{\mathrm{noise}}^2 = \int_{f_0}^{f_1} S_{\mathrm{in}}(f)^2 \, df = \alpha \log(f_1/f_0) + \beta(f_1 - f_0).$$

Therefore imposing a maximum total RMS noise voltage over the band $[f_0, f_1]$ is the posynomial constraint

$$\alpha \log(f_1/f_0) + \beta(f_N - f_0) \leq V_{\mathrm{max}}^2 \tag{1.44}$$

(since f_1 and f_0 are fixed, and α and β are posynomials in the design variables).

1.7 Optimal design problems and examples

1.7.1 Summary of constraints and specifications

The many performance specifications and constraints described in the previous sections are summarized in table 1.1. Note that with only one exception (the positive supply rejection ratio) the specifications and constraints can be handled via geometric programming.

Since all the op-amp performance measures and constraints shown above can be expressed as posynomial functions and posynomial constraints, we can solve a wide variety of op-amp design problems via geometric programming. We can, for example, maximize the bandwidth subject to given (upper) limits on op-amp power, area, and input offset voltage, and given (lower) limits on transistor lengths and widths, and voltage gain, CMRR, slew rate, phase margin, and output voltage swing. The resulting optimization problem is a geometric programming problem. The problem may appear to be very complex, involving many complicated inequality and equality constraints, but in fact is readily solved.

1.7.2 Example

In this section, we describe a simple design example. A 0.8μm CMOS technology was used; see Appendix §1.C for the technology parameters. The

Specification/constraint	Type	Equation(s)
Symmetry and matching	Mono.	1.5, 1.6
Device sizes	Mono.	1.7
Area	Posy.	1.8
Systematic offset voltage	Mono.	1.9
Bias conditions		
Common-mode input range	Posy.	1.12, 1.13
Output voltage swing	Posy.	1.14, 1.15
Gate overdrive	Mono.	1.16
Quiescent power	Posy.	1.17
Open loop gain	Mono.	1.20
Dominant pole conditions	Posy.	1.32
3dB bandwidth	Mono.	1.31
Unity-gain bandwidth	Mono.	1.34
Phase margin	Posy.	1.36
Slew rate	Posy.	1.37
CMRR	Mono.	1.38
Neg. PSRR	Inv. Posy.	1.40, 1.41
Pos. PSRR	Neither	1.42
Input-referred spot noise	Posy.	1.43
Input-referred total noise	Posy.	1.44

TABLE 1.1. Design constraints and specifications for the two-stage op-amp.

positive supply voltage was set at 5V and the negative supply voltage was set at 0V. The load capacitance was 3pF.

The objective is to maximize unity-gain bandwidth subject to the requirements shown in Table 1.2. The resulting geometric program has eighteen variables, seven (monomial) equality constraints, and twenty-eight (posynomial) inequality constraints. The total number of monomial terms appearing in the objective and all constraints is 68. Our simple MATLAB program solves this problem in roughly one to two seconds real-time. The optimal design obtained is shown in Table 1.3.

Constraint	Specification
Device length	$\geq 0.8\mu$m
Device width	$\geq 2\mu$m
Area	$\leq 10000\mu\text{m}^2$
Common-mode input voltage	fixed at $V_{\text{DD}}/2$
Output voltage range	$[0.1, 0.9]V_{\text{DD}}$
Quiescent power	$\leq$ 5mW
Open-loop gain	$\geq$ 80dB
Unity-gain bandwidth	Maximize
Phase margin	$\geq 60°$
Slew rate	$\geq 10\text{V}/\mu$s
Common-mode rejection ratio	$\geq$ 60dB
Neg. power supply rejection ratio	$\geq$ 80dB
Pos. power supply rejection ratio	$\geq$ 80dB
Input-referred spot noise, 1kHz	$300\text{nV}/\sqrt{\text{Hz}}$

TABLE 1.2. Specifications and constraints for design example.

The performance achieved by this design, as predicted by the program, is summarized in Table 1.4. The design achieves an 86MHz unity-gain bandwidth. Note that some constraints are tight (minimum device length, minimum device width, maximum output voltage, quiescent power, phase margin and input-referred spot noise) while some constraints are not tight (area, minimum output voltage open-loop gain, common-mode rejection ratio and slew rate).

1.7.3 Trade-off analyses

By repeatedly solving optimal design problems as we sweep over values of some of the constraint limits, we can sweep out globally optimal trade-off curves for the op-amp. For example, we can fix all other constraints, and repeatedly minimize power as we vary a minimum required unity-gain bandwidth. The resulting curve shows the globally optimal trade-off between unity-gain bandwidth and power (for the values of the other limits).

Variable	Value
$W_1 = W_2$	232.8μm
$W_3 = W_4$	143.6μm
W_5	64.6μm
W_6	588.8μm
W_7	132.6μm
W_8	2.0μm
$L_1 = L_2$	0.8μm
$L_3 = L_4$	0.8μm
L_5	0.8μm
L_6	0.8μm
L_7	0.8μm
L_8	0.8μm
C_c	3.5pF
I_{bias}	10μA

TABLE 1.3. Optimal design for design example.

Specification/Constraint	Performance
Minimum device length	0.8μm
Minimum device width	2μm
Area	8200μm^2
Output voltage range	$[0.03, 0.9]V_{\text{DD}}$
Minimum gate overdrive	130mV
Quiescent power	5mW
Open-loop gain	89.2dB
Unity-gain bandwidth	86MHz
Phase margin	60°
Slew rate	88V/μs
Common-mode rejection ratio	92.5dB
Negative power supply rejection ratio	98.4dB
Positive power supply rejection ratio	116dB
Maximum input-referred spot noise, 1kHz	300nV/$\sqrt{\text{Hz}}$

TABLE 1.4. Performance of optimal design for design example.

In this section we show several optimal trade-off curves for the operational amplifier. We do this by fixing all the specifications at the default values shown in Table 1.2 except two that we vary to see the effect on a circuit performance measure. When the optimization objective is not bandwidth we use a default value of minimum unity-gain bandwidth of 30MHz.

We first obtain the globally optimal trade-off curve of unity-gain bandwidth versus power for different supply voltages. The results can be seen in Figure 1.3. Obviously the more power we allocate to the amplifier, the larger the bandwidth obtained; the plots, however, show exactly how much more bandwidth we can obtain with different power budgets. We can see, for example, that the benefits of allocating more power to the op-amp disappear above 5mW for a supply voltage of 2.5V, whereas for a 5V supply the bandwidth continues to increase with increasing power. Note also that each of the supply voltages gives the largest unity-gain bandwidth over some range of powers.

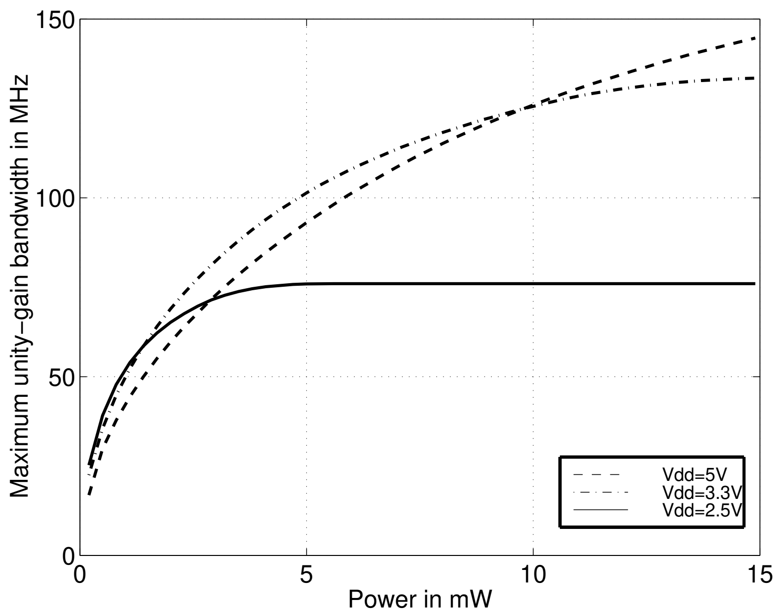

FIGURE 1.3. Maximum unity-gain bandwidth versus power for different supply voltages.

We next determine the influence of the output voltage range on the maximum unity-gain bandwidth versus power relation. The output voltage range is specified to be 98%, 95%, and 90% of the supply around mid-supply. Figure 1.4 shows the results. Note that when the output voltage range is 98% of the supply the maximum achievable frequency is 55MHz. However when the output range constraint is loosened to 90% of the supply the maximum unity-gain bandwidth is almost 150MHz.

In Figure 1.5 we plot the globally optimal trade-off curve of open-loop gain versus unity-gain frequency for different phase margins. Note that for a large unity-gain bandwidth requirement only small gains are achievable. Also, we can see that for a tighter phase margin constraint the gain bandwidth product is lower.

Figure 1.6 shows the minimum input-referred spectral density at 1kHz versus power, for different unity-gain frequency requirements. Note that when the power specification is tight, increasing the power greatly helps to

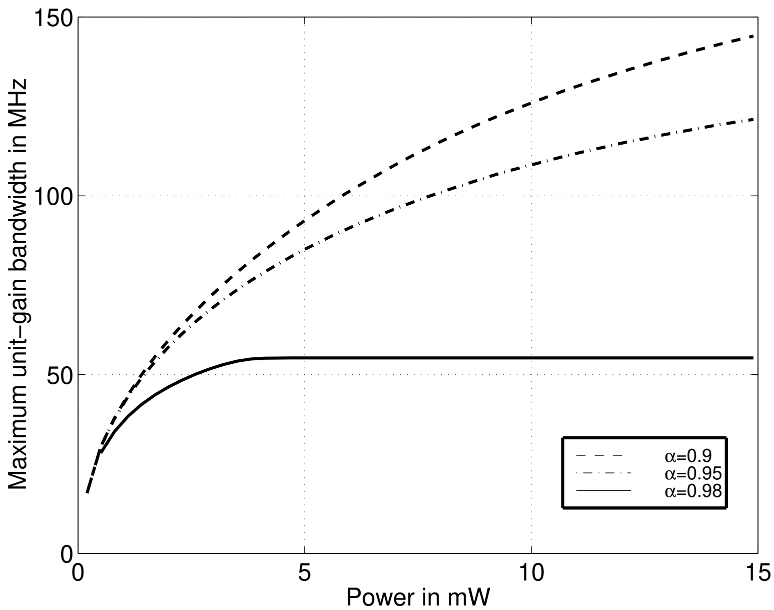

FIGURE 1.4. Maximum unity-gain bandwidth versus power for different output voltage ranges.

decrease the input-referred noise spectral density.

In Figure 1.7 we show the optimal trade-off curve of unity gain bandwidth versus power for different load capacitances. We can see that a smaller load capacitance provides larger unity-gain bandwidth. For our specifications, a change in load capacitance from 9pF to 1pF doubles the achievable unity-gain bandwidth at large power budgets.

In Figure 1.8 we show the optimal trade-off curve of unity-gain bandwidth versus area for different different power budgets. We can see that when the area constraint is tight, increasing the available area translates into a greater unity bandwidth. After some point, other constraints become more stringent and increasing the available area does not improve the maximum achievable unity-gain bandwidth.

The phase margin constraint given by (1.35) assumes that the closed-loop gain is one. If the closed-loop gain is greater than one, then we specify the crossover frequency and phase margin at the appropriate closed-loop gain level. In Figure 1.9 we show the optimal trade-off curve of crossover bandwidth versus power for different closed-loop gains. Note the large improvement obtained when the closed-loop gain is greater than one. This is easily understood: when the closed-loop gain increases, the parasitic poles frequencies can be much lower.

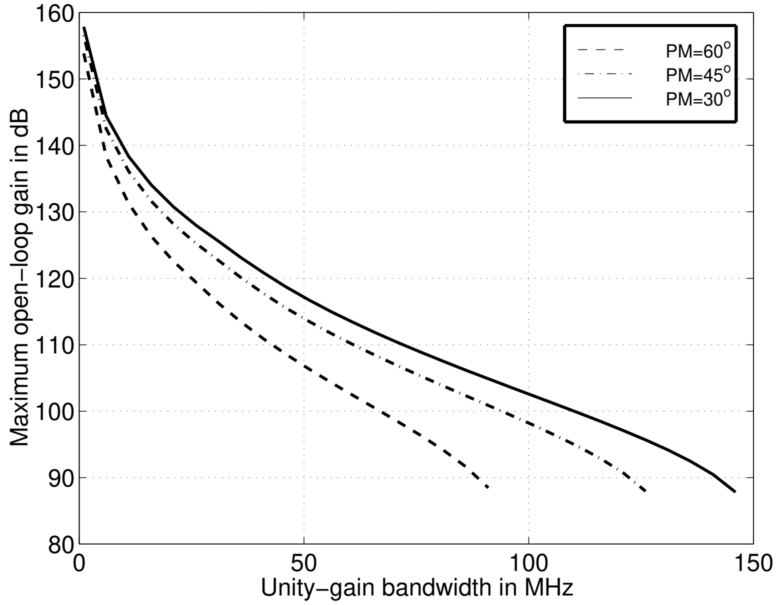

FIGURE 1.5. Maximum open-loop gain versus unity-gain bandwidth for different phase margins.

1.7.4 Sensitivity analysis example

In this section we analyze the information provided by the sensitivity analysis of the first design problem in §1.7.2 (maximize the unity-gain bandwidth when the rest of specifications/constraints are set to the values shown in Table 1.2). The results of this sensitivity analysis are shown in Table 1.5. The column labeled Sensitivity (numerical) is obtained by tightening and loosening the constraint in question by 5% and re-solving the problem. (The average from the two is taken.) The column labeled Sensitivity comes (essentially for free) from solving the original problem. Note that it gives an excellent prediction of the numerically obtained sensitivities.

There are six active constraints: minimum device length, minimum device width, maximum output voltage, quiescent power, phase margin, and input-referred spot noise at 1kHz. All of these constraints limit the maximum unity-gain bandwidth. The sensitivities indicate which of these constraints are more critical (more limiting). For example a 10% increase in the allowable input-referred noise at 1kHz will produce a design with (approximately) 2.4% improvement in unity-gain bandwidth. However a 10% decrease in the maximum phase margin at the unity gain bandwidth will produce a design with (approximately) 17.6% improvement in unity-gain bandwidth. It is very interesting to analyze the sensitivity to the minimum device width constraint. A 10% decrease in the minimum device width pro-

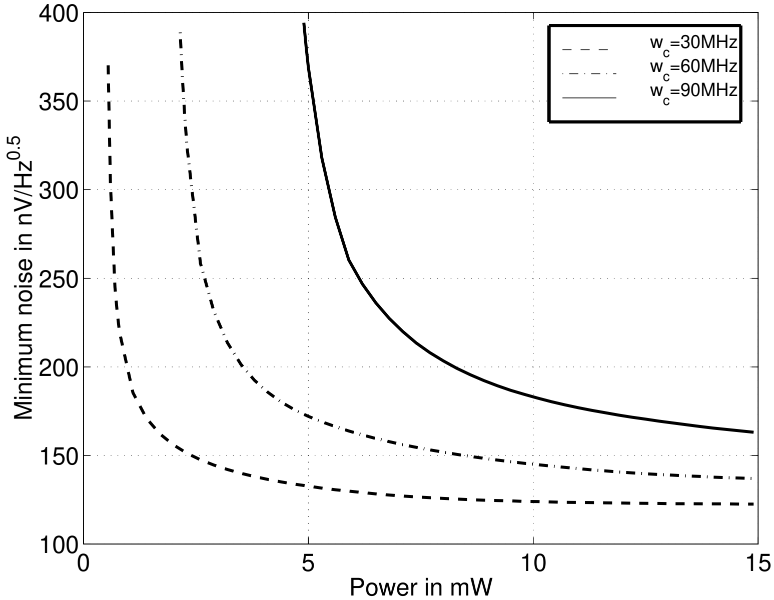

FIGURE 1.6. Minimum noise density at 1kHz versus power for different unity-gain bandwidths.

Constraint	Spec.	Program	Sensitivity (numerical)	Sensitivity
Min. device length	$\geq 0.8\mu$m	$.8\mu$m	0.299	0.309
Min. device width	$\geq 2\mu$m	2.0μm	0.0049	0.0048
Area	$\leq 10000\mu\text{m}^2$	$8200\mu\text{m}^2$	0	0
Max. output voltage	4.5V	4.5V	-0.365	-0.349
Min. output voltage	0.5V	0.13V	0	0
Quiescent power	$\leq$ 5mW	4.99mW	-0.482	-0.483
Open-loop gain	$\geq$ 80dB	89.2dB	0	0
Phase margin	$\geq 60^\circ$	60°	-1.758	-1.757
Slew rate	$\geq 10\text{V}/\mu$s	$88\text{V}/\mu$s	0	0
CMRR	$\geq$ 60dB	92.5dB	0	0
Neg. PSRR	$\geq$ 80dB	98.4dB	0	0
Pos. PSRR	$\geq$ 80dB	116dB	0	0
Spot noise, 1kHz	$\leq 300\text{nV}/\sqrt{\text{Hz}}$	$300\text{nV}/\sqrt{\text{Hz}}$	0.24	0.241

TABLE 1.5. Sensitivity analysis for the design example.

duces a design with only a 0.05% improvement in unity-gain bandwidth. This can be interpreted as follows: even though the minimum device width constraint is binding, it can be considered not binding in a practical sense since tightening (or loosening) it will barely change the objective.

The program classifies the given constraints in order of importance from

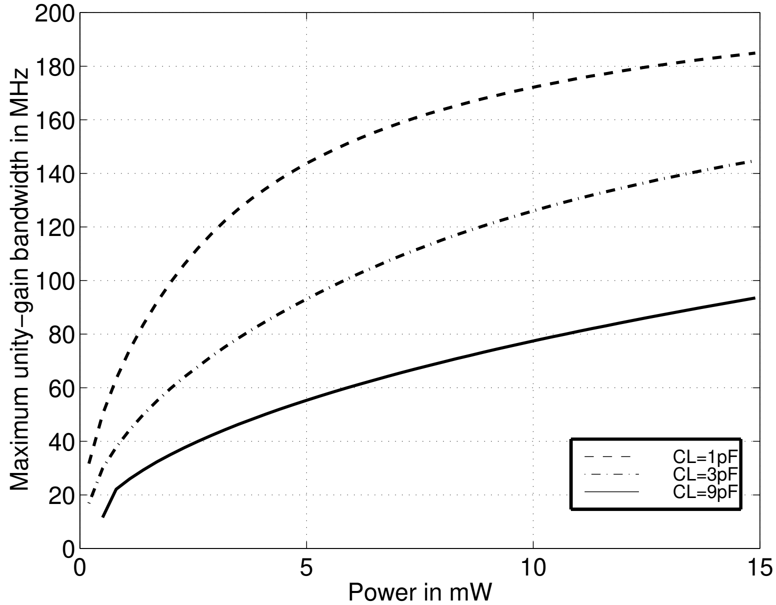

FIGURE 1.7. Maximum unity-gain bandwidth versus power for different load capacitances.

most limiting to least limiting. For this design the order is: phase margin, maximum output voltage, minimum device length, quiescent power, input-referred noise at 1kHz, and minimum device width. The program also tells the designer which constraints are *not critical* (the ones whose sensitivities are zero or small). A small relaxation of these constraints will not improve the objective function, so any effort to loosen them will not be rewarded.

1.8 Design verification

Our optimization method is based on the simple square-law device models described in appendix §1.A. Our model does not include several potentially important factors such as body effect, channel length modulation in the bias equations, and the dependence of junction capacitances on junction voltages. Moreover we make several approximations in the circuit analysis used to formulate the constraints. For example, we approximate the transfer function with the four pole form (1.19); the actual transfer function, even based on the simple model, is more complicated. As another example, we approximate $\arctan(a) \approx a$ in our simple version of the phase margin constraint.

While all of these approximations are reasonable (at least when channel lengths are not too short), it is important to *verify* the designs obtained

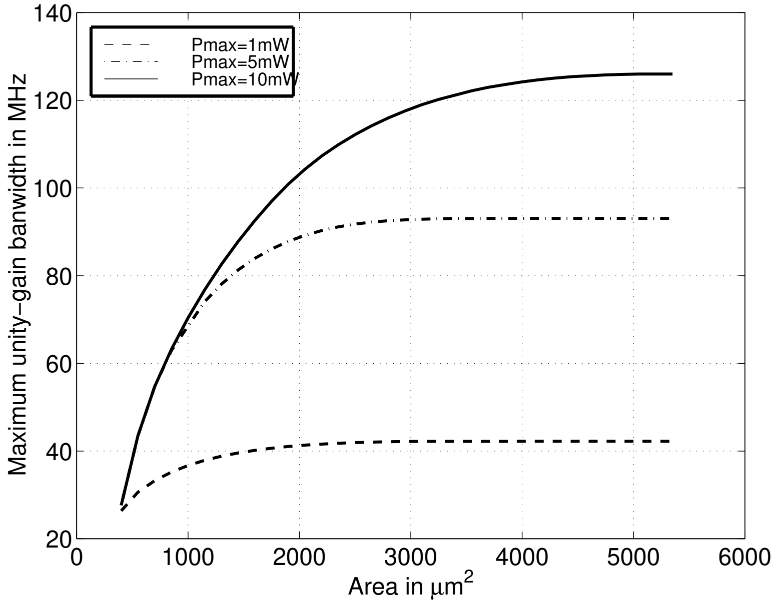

FIGURE 1.8. Maximum unity-gain bandwidth versus area for different power budgets.

using a higher fidelity (presumably nonposynomial) model.

1.8.1 HSPICE level 1 verification with GP0 models

We first verify the designs generated by our geometric programming method (using GP0, or long channel, models) with HSPICE using a long channel model (HSPICE level 1 model). We take the design found by the geometric programming method, and then use HSPICE level 1 model to check the various performance measures. The level 1 HSPICE model is substantially more accurate (and complicated) than our simple posynomial models. It includes, for example, body effect, channel-length modulation, junction capacitance that depends on bias conditions, and a far more complex transfer function that includes many other parasitic capacitances. The unity gain bandwidth and phase margin are computed by solving the complete small signal model of the op-amp. The results of such verification always show excellent agreement between our posynomial models and the more complex (and nonposynomial) HSPICE level 1 model. As an example, Table 1.6 summarizes the results for the standard problem described above in §1.7.4. Note that the values of the performance specifications from the posynomial model (in the column labeled Program) and the values according to HSPICE level 1 (in the right-hand column) are in close agreement. Moreover the deviations between the two are readily understood. The unity-gain

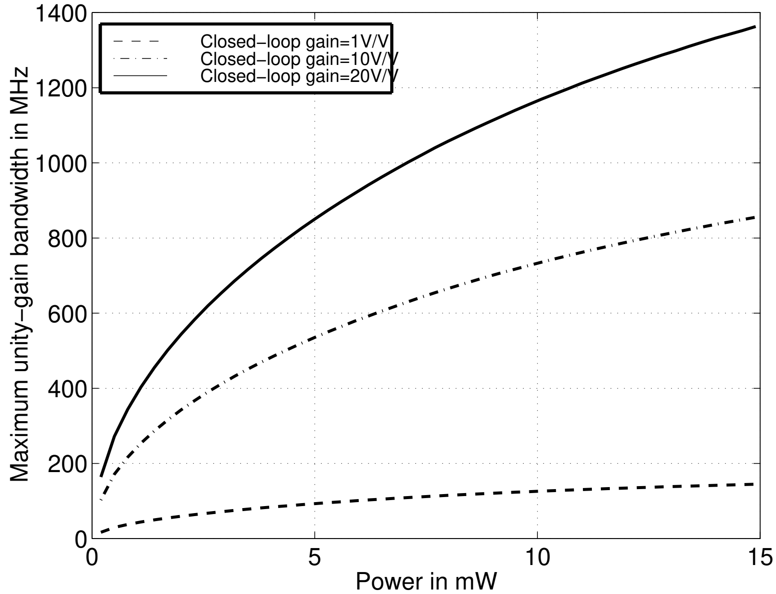

FIGURE 1.9. Maximum crossover bandwidth versus power for different closed-loop gains.

frequency is slightly overestimated and the phase margin is slightly underestimated because we use the approximate expression (1.34), which ignores the effect of the parasitic poles on the crossover frequency. The noise is overestimated 7% because the open loop gain has decreased 7% already at 1kHz; this gain reduction translates into a reduction in the input-referred noise.

Constraint	Spec.	Program	HSPICE 1
Max. output voltage	$\geq$ 4.5V	4.5V	4.5V
Min. output voltage	$\leq$ 0.5V	0.13V	0.13V
Quiescent power	$\leq$ 5mW	4.99mW	4.95mW
Open-loop gain	$\geq$ 80dB	89.2dB	89.4dB
Unity-gain bandwidth	maximize	86MHz	81MHz
Phase margin	$\geq 60°$	60°	64°
Slew rate	$\geq$ 10V/μs	88V/μs	92.5V/μs
CMRR	$\geq$ 60dB	92.5dB	94dB
Neg. PSRR	$\geq$ 80dB	98.4dB	98.1dB
Pos. PSRR	$\geq$ 80dB	116dB	114dB
Spot noise, 1kHz	$\leq 300\mathrm{nV}/\sqrt{\mathrm{Hz}}$	$300\mathrm{nV}/\sqrt{\mathrm{Hz}}$	$280\mathrm{nV}/\sqrt{\mathrm{Hz}}$

TABLE 1.6. Design verification with HSPICE level 1. The performance measures obtained by the program are compared with those found by HSPICE level 1 simulation.

We have verified the geometric program results with the HSPICE level 1 model simulations for a wide variety of designs (with a wide variety of power, bandwidth, gain, etc.). The results are always in close agreement. Thus our simple posynomial models are reasonably good approximations of the HSPICE level 1 models.

1.8.2 HSPICE BSIM model verification with GP1 models

In this section we show how the geometric programming method performs well even for short-channel devices, when more sophisticated GP1 transistor models are used, by verifying designs against sophisticated HSPICE level 39 (BSIM3v1) simulations. The GP1 model is described in appendix §1.A.3; the only difference is that we use an empirically found monomial expression for the output conductance of a MOS transistor instead of the standard long channel formula. Using these GP1 models, all of the constraints described above are still compatible with geometric programming.

Table 1.7 shows the comparison between the results of geometric programming design, using GP1 models, and HSPICE level 39 simulation, for the standard problem described in §1.7.4. The predicted values are very close to the simulated values. The agreement holds for a wide variety of designs.

Constraint	Spec.	Program	HSPICE 39
Max. output voltage	≥ 4.5V	4.55V	4.4V
Min. output voltage	≤ 0.5V	240mV	200mV
Quiescent power	≤ 5mW	4.99mW	5.2mW
Open-loop gain	≥ 80dB	80dB	83dB
Unity-gain bandwidth	maximize	75MHz	73MHz
Phase margin	$\geq 60°$	60°	62°
Slew rate	$\geq 10\text{V}/\mu\text{s}$	$97\text{V}/\mu\text{s}$	$95\text{V}/\mu\text{s}$
CMRR	≥ 60dB	86dB	88dB
Neg. PSRR	≥ 80dB	84dB	86dB
Pos. PSRR	≥ 80dB	93dB	92dB
Spot noise, 1kHz	$\leq 300\text{nV}/\sqrt{\text{Hz}}$	$300\text{nV}/\sqrt{\text{Hz}}$	$285\text{nV}/\sqrt{\text{Hz}}$

TABLE 1.7. Design verification with HSPICE level 39. Geometric programming is used to solve the standard problem, using the more sophisticated GP1 models. The results are compared with HSPICE level 39 simulation.

1.9 Design for process robustness

So far we have assumed that parameters such as transistor threshold voltages, mobilities, oxide parameters, channel modulation parameters, supply voltages, and load capacitance are all known and fixed. In this section we show to how to use the methods of this paper to develop designs that are *robust* with respect to variations in these parameters, *i.e.*, designs that meet a set of specifications for a set of values of these parameters. Such designs can dramatically increase yield.

There are many approaches to the problem of robustness and yield optimization; see, *e.g.*, [5], [27]. The robust design problem can be formulated as a so-called semi-infinite programming problem, in which the constraints must hold for all values of some parameter that ranges over an interval, as in [74], which used DELIGHT.SPICE to do robust designs, or more recently, Mukherjee et al. ([68] ref), who use ASTRX/OBLX. These methods often involve very considerable run times, ranging from minutes to hours.

We also formulate the problem as a sampled version of a semi-infinite program. The method is practical only because geometric programming can readily handle problems with many hundreds, or even thousands, or constraints; the computational effort grows approximately linearly with the number of constraints.

The basic idea in our approach is to list a set of possible parameters, and to replicate the design constraints for all possible parameter values. Let $\alpha \in \mathbf{R}^k$ denote a vector of parameters that may vary. Then the objective and constraint functions can be expressed as functions of x (the design parameters) and α (which we will call the *process parameters*, even if some components, *e.g.*, the load capacitance, are not really process parameters):

$$f_0(x, \alpha), \qquad f_i(x, \alpha), \qquad g_i(x, \alpha).$$

The functions f_i are all posynomial functions of x, for each α, and the functions g_i are all monomial functions of x, for each α. Let $\mathcal{A} = \{\alpha_1, \dots \alpha_N\}$ be a (finite) set of possible parameter values. Our goal is to determine a design (*i.e.*, x) that works well for all possible parameter values (*i.e.*, $\alpha_1, \dots, \alpha_N$).

First we describe several ways the set $\mathcal{A}$ might be constructed. As a simple example, suppose there are 6 parameters, which vary independently over intervals $[\alpha_{\min,i}, \alpha_{\max,i}]$. We might sample each interval with 3 values (*e.g.*, the midpoint and extreme values), and then form every possible combinations of parameter values, which results in $N = 3^6$.

We do not have to give every possible combination of parameter values, but only the ones likely to actually occur. For example if it is unlikely that the oxide capacitance parameter is at its maximum value while the n-threshold voltage is maximum, then we delete these combinations from our set $\mathcal{A}$. In this way we can model interdependencies among the parameter values.

We can also construct $\mathcal{A}$ in a straightforward way. Suppose we require a design that works, without modification, on several processes, or several variations of processes. Then $\mathcal{A}$ is simply a list of the process parameters for each of the processes.

The robust design is achieved by solving the problem

$$\begin{array}{ll} \text{minimize} & \max_{\alpha \in \mathcal{A}} f_0(x, \alpha) \\ \text{subject to} & f_i(x, \alpha) \leq 1, \quad i = 1, \ldots, m, \quad \text{for all } \alpha \in \mathcal{A}, \\ & g_i(x, \alpha) = 1, \quad i = 1, \ldots, p, \quad \text{for all } \alpha \in \mathcal{A}, \\ & x_i > 0, \quad i = 1, \ldots, n. \end{array} \tag{1.45}$$

This problem can be reformulated as a geometric program with N times the number of constraints, and an additional scalar variable γ (see [22]):

$$\begin{array}{ll} \text{minimize} & \gamma \\ \text{subject to} & f_0(x, \alpha_j) \leq \gamma, \quad j = 1, \ldots, N, \\ & f_i(x, \alpha_j) \leq 1, \quad i = 1, \ldots, m, \quad j = 1, \ldots, N, \\ & g_i(x, \alpha_j) = 1, \quad i = 1, \ldots, p, \quad j = 1, \ldots, N, \\ & x_i > 0, \quad i = 1, \ldots, n. \end{array} \tag{1.46}$$

The solution of (1.45) (which is the same as the solution of (1.46)) satisfies the specifications for all possible values of the process parameters. The optimal objective value gives the (globally) optimal minimax design. (It is also possible to take an average value of the objective over process parameters, instead of a worst-case value.)

Equality constraints have to be handled carefully. Provided the transistor lengths and widths are not subject to variation, equality constraints among them (*e.g.*, matching and symmetry) are likely not to depend on the process parameter α. Other equality constraints, however, can depend on α. When we enforce an equality constraint for each value of α, the result is (usually) an infeasible problem. For example suppose we specify that the open-loop gain equal exactly 80dB. Process variation will change the open-loop gain, making it impossible to achieve a design that has an open-loop gain of *exactly* 80dB for more than a few process parameter values. The solution to this problem is to convert such specifications into inequalities. We might, for example, change our specification to require that the open-loop gain exceed 80dB, or require it to be between 80dB and 85dB. Either way the robust problem now has at least a chance of being feasible.

It's important to contrast a robust design for a set of process parameters $\mathcal{A} = \{\alpha_1, \ldots, \alpha_N\}$ with the optimal designs for each process parameter. The objective value for the robust design is worse (or no better) than the optimal design for each parameter value. This disadvantage is offset by the advantage that the design works for all the process parameter values. As a simple example, suppose we seek a design that can be run on two processes (α_1 and α_2). We can compare the robust design to the two optimal designs. If the objective achieved by the robust is not much worse than the two optimal designs, then we have the advantage of a single design that works on two processes. On the other hand if the robust design is much worse (or even infeasible) we may elect to have two versions of the amplifier design, each one optimized for the particular process.

So far we have considered the case in which the set $\mathcal{A}$ is finite. But in most real cases it is infinite; for example, individual parameters lie in ranges. We have already indicated above that such situations can be modeled or approximated by sampling the interval. While we believe this will always work in practice, it gives no guarantee, in general, that the design works for *all* values of the parameter in the given range; it only guarantees performance for the sampled values of the parameters.

There are many cases, however, when we *can* guarantee the performance for a parameter value in an interval. Suppose that the function $f_i(x, \alpha)$ is posynomial not just in x, but in x and α as well, and that α lies in the interval $[\alpha_{\min}, \alpha_{\max}]$. (We take α scalar here for simplicity.) Then it suffices to impose the constraint at the endpoints of the interval, *i.e.*:

$$f_i(x, \alpha_{\min}) \leq 1, \quad f_i(x, \alpha_{\max}) \leq 1$$

$$\Rightarrow \quad f_i(x, \alpha) \leq 1 \quad \text{for all } \alpha \in [\alpha_{\min}, \alpha_{\max}].$$

This is easily proved using convexity of the $\log f_i$ in the transformed variables.

The reader can verify that the constraints described above are posynomial in the parameters C_{ox}, μ_{n}, μ_{p}, λ_{n}, λ_{p}, and the parasitic capacitances. Thus, for these parameters at least, we can handle ranges with no approximation or sampling, by specifying the constraints only at the endpoints.

The requirement of robustness is a real practical constraint, and is currently dealt with by many methods. For example, a minimum gate overdrive constraint is sometimes imposed because designs with small gate overdrive tend to be nonrobust. The point of this section is that robustness can be achieved in a more methodical way, which takes into account a more detailed description of the possible uncertainties or parameter variations. The result will be a better design than an *adhoc* method for achieving robustness.

Finally we demonstrate the method with a simple example. In Table 1.8 we show how a robust design compares to a nonrobust design. We take three process parameters: the bias current error factor, the positive power supply error factor, and oxide capacitance. The bias current error factor is the ratio of the actual bias current to our design value, so when it is one, the true bias current is what we specify it to be, and when it is 1.1, the true bias current is 10% larger than we specify it to be. Similarly the positive power supply error factor is the ratio of the actual bias current to our design value. The bias current error factor varies between 0.9 and 1.1, the positive power supply error factor varies between 0.9 and 1.1, and the oxide capacitance varies ±10% around its nominal value. The three parameters are assumed independent, and we sample each with three values (midpoint and extreme values) so all together we have $N = 3^3$, *i.e.*, 27 different process parameter vectors. In the third column we show the performance of the robust design. For each specification we determine the worst performance over all 27 process parameters. In the fourth column we show the performance of the nonrobust design. Again, only the worst case performance over all 27 process parameters is indicated for each specification. The resulting geometric program involves eighteen variables, seven monomial equality constraints (*i.e.*, symmetry and matching) and 756 posynomial inequality constraints.

Constraint	Spec.	Rob. design	Std. design
Quiescent power	$\leq$ 5mW	4.99mW	5.75mW
Open-loop gain	$\geq$ 80dB	89dB	87dB
Unity-gain bandwidth	maximize	72MHz	77MHz
Phase margin	$\geq 60°$	60°	55°
CMRR	$\geq$ 60dB	93dB	90dB
Neg. PSRR	$\geq$ 80dB	94dB	93dB
Pos. PSRR	$\geq$ 80dB	110dB	109dB
Spot noise, 1kHz	$\leq 300\text{nV}/\sqrt{\text{Hz}}$	$300\text{nV}/\sqrt{\text{Hz}}$	$316\text{nV}/\sqrt{\text{Hz}}$

TABLE 1.8. Robust design.

The new design obtains a unity-gain bandwidth of 72MHz. The design in §1.7.2 obtains a worst-case unity gain bandwidth of 77MHz but since it was specified only for nominal conditions, it fails to meet some constraints when tested over all conditions. For example, the power consumption increases by 15%, the open-loop gain decreases by 20%, the input-referred spot noise at 1kHz increases by 5% and the phase margin decreases by 5°. The robust design, on the other hand, meets *all* specifications for all 27 sets of process parameters.

1.10 Discussions and conclusions

We have shown how geometric programming can be used to design and optimize a common CMOS amplifier. The method yields globally optimal designs, is extremely efficient, and handles a very wide variety of practical constraints.

Since no human intervention is required (*e.g.*, to provide an initial 'good' design, or to interactively guide the optimization process), the method yields completely automated synthesis of (globally) optimal CMOS amplifiers, directly from specifications. This implies that the circuit designer can spend more time doing real design, *i.e.*, carefully analyzing the optimal trade-offs between competing objectives, and less time doing parameter tuning, or wondering whether a certain set of specifications can be achieved. The method could be used, for example, to do full custom design for each op-amp in a complex mixed-signal integrated circuit; each amplifier is optimized for its load capacitance, required bandwidth, closed-loop gain, etc.

In fact, the method can handle problems with constraints or coupling between the different op-amps in an integrated circuit. As simple examples, suppose we have 100 op-amps, each with a set of specifications. We can minimize the *total* area or power by solving a (large) geometric program. In this case, we are solving (exactly) the power/area allocation problem for the 100 op-amps on the integrated circuit. We can also handle direct coupling between the op-amps, *i.e.*, when component values in one op-amp (*e.g.*, input transistor widths) affect another (*e.g.*, as load capacitance). The resulting geometric program will have perhaps hundreds of variables, thousands of constraints, and be quite sparse, so it is well within the capabilities of current interior-point methods.

For example, switched capacitor filters (see [41]) are complex systems where the performance (maximum clock frequency, area, power, etc.) is influenced by both the op-amp and the capacitors. Current CAD tools for switched capacitor filters size the capacitors but use the same op-amp for all integrators (see, *e.g.*, FIDES [92],[8, 6]). In contrast, we can custom design each op-amp in the switched capacitor filter. Designing optimal op-amps for filters in oversampled converters has also been addressed in [97], but little work has been done to fully automate the design. Limiting amplifiers for FM communication systems (see [54]) require the design of multiple amplifiers in cascade. Typically the same amplifier is used in all stages because it takes too long and it is too difficult to design each stage separately.

Our ability to handle much larger problems than arise from a single op-amp design can be used to develop robust designs. This could increase yield, or result in designs with a longer lifetime (since they work with several different processes).

The method unambiguously determines feasibility of a set of specifications: it either produces a design that meets the specifications, or it provides a proof that the specifications cannot be achieved. In either case it also provides, at essentially no additional cost, the sensitivities with respect to every constraint. This gives a very useful quantitative measure of how tight each constraint is, or how much it affects the objective.

In this paper we considered only one op-amp circuit, but the general method is applicable to many other circuits, as will be reported in papers currently under preparation [48]. For the op-amp considered here, the analytical expressions for the constraints and specifications were derived by hand, but in a more general setting this step could be automated by the use of symbolic circuit simulators like ISAAC [36], SYNAP [83] and ASAP [32]. A CAD tool for optimization of analog op-amps could be developed. It would consist of a symbolic analyzer ([33]), a GP code solver, and a user interface. It could be linked to an automatic layout program (such as ILAC [26] or KOAN/ANAGRAM [15], so the resulting tool could generate mask designs directly from amplifier specifications.

The main disadvantage of the method we have described is that it handles only certain types of constraints and specifications, *i.e.*, monomial equality constraints and posynomial inequality constraints. The main contribution of this paper is to point out that despite this restriction, we can handle a very wide variety of practical amplifier specifications.

The method we propose can be effectively combined with a (local) optimization method that uses nonposynomial but more accurate model equations, or even circuit simulation. Thus, the geometric programming method we propose is used to get close to the (presumably global) optimum, and the final design is tuned using the more accurate (but nonposynomial) model or direct circuit simulation. In the many hundreds of designs we have carried out, verification with an accurate simulator has shown that our posynomial models are very accurate, so, presumably, tuning our designs with a more accurate (*e.g.*, simulation-based) method would require only an iteration or two to converge.

We close our discussion on the topic of circuit models. This paper introduces a new quality of a circuit model, *i.e.*, whether it results in posynomial specifications. The traditional trade-off in circuit modeling is between fidelity and complexity, *e.g.*, simple but not too accurate models for hand analysis and design, versus complex, high fidelity models for design verification. Evidently we have a third quality for a model: whether or not it results in posynomial specifications. Thus we have a trade-off between monomial/posynomial models (for design via GP) and fidelity or accuracy. Note that complexity doesn't matter: a very complex, but posynomial, model is readily handled by GP. Developing accurate, but posynomial, circuit models is a new area for research.

Acknowledgments: We thank Edo Waks, who wrote the geometric programming solver used for the numerical experiments shown in this paper. We are grateful to Boyd Fowler, Abbas El Gamal, Ali Hajimiri, as well as several anonymous reviewers, for very useful comments and suggestions.

References

[1] M. A. Aguirre, J. Chávez, , A. Torralba, and L. G. Franquelo. Analog design optimization by means of a tabu search approach. In *Proceedings IEEE International Symposium on Circuits and Systems*, volume 1, pages 375–378, 1994.

[2] B. K. Ahuja. An improved frequency compensation technique for CMOS operational amplifiers. *IEEE Journal of Solid-State Circuits*, 18:629–633, December 1983.

[3] P. E. Allen and D. R. Holberg. *CMOS analog circuit design*. Oxford Univeristy Press, 1st. edition, 1987.

[4] B. A. A. Antao. Trends in CAD of analog ICs. *IEEE Circuits and Devices Magazine*, 12(5):31–41, September 1996.

[5] K. J. Antreich and H. E. Graeb. Circuit optimization driven by worst-case distances. In *Proceedings IEEE International Conference on Computer-Aided Design*, pages 166–169, 1991.

[6] J. Assael, P. Senn, and M. S. Tawfik. A switched-capapcitor filter silicon compiler. *IEEE Journal of Solid-State Circuits*, 23:166–174, February 1988.

[7] M. Avriel, R. Dembo, and U. Passy. Solution of generalized geometric programs. *International Journal for Numerical Methods in Engineering*, 9:149–168, 1975.

[8] A. Barlow, K. Takasuka, Y. Nambu, T. Adachi, and J. Konno. An integrated switched-capacitor filter design system. In *Proceedings IEEE Custom Integrated Circuit Conference*, pages 4.5.1–4.5.5, 1989.

[9] C. S. Beightler and D. T. Phillips. *Applied Geometric Programming*. John Willey & Sons, 1976.

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

[11] R. K. Brayton, G. D. Hachtel, and A. Sangiovanni-Vincentelli. A survey of optimization techniques for integrated-circuit design. *Proceedings of the IEEE*, 69(10):1334–1362, October 1981.

[12] L. R. Carley, G. G. E. Gielen, R. A. Rutenbar, and W. M. C. Sansen. Synthesis tools for mixed-signal ICs: Progress on front-end and back-end strategies. In *Proceedings of the 33rd Annual Design Automation Conference*, pages 298–303, 1996.

[13] L. R. Carley and R. A. Rutenbar. How to automate analog IC designs. *IEEE Spectrum*, 25:26–30, August 1988.

[14] J. Chávez, M. A. Aguirre, and A. Torralba. Analog design optimization: a case study. In *Proceedings IEEE International Symposium on Circuits and Systems*, volume 3, pages 2083–2085, 1993.

[15] J. M. Cohn, D. J. Garrod, R. A. Rutenbar, and L. R. Carley. KOAN/ANAGRAM II: New tools for device-placement and routing. *IEEE Journal of Solid-State Circuits*, 26, March 1991.

[16] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *LANCELOT: A Fortran package for large-scale nonlinear optimization (Release A)*, volume 17 of *Springer Series in Computational Mathematics*. Springer-Verlag, 1992.

[17] R. J. Duffin. Linearizing geometric programs. *SIAM Review*, 12:211–227, 1970.

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

[19] J. Ecker. Geometric programming: methods, computations and applications. *SIAM Rev.*, 22(3):338–362, 1980.

[20] J. Ecker and M. Kupferschmid. *Introduction to Operations Research*. Krieger, 1991.

[21] F. El-Turky and E. E. Perry. BLADES: An artificial intelligence approach to analog circuit design. *IEEE Transactions on Computer-Aided Design*, 8:680–692, June 1989.

[22] A. R. Conn et al. Optimization of custom CMOS circuits by transistor sizing. In *Proceedings IEEE International Conference on Computer-Aided Design*, pages 174–180, 1996.

[23] D. G. Marsh et al. A single-chip CMOS PCM codec with filters. *IEEE Journal of Solid-State Circuits*, 16:308–315, August 1981.

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

[25] I. Vassiliou et al. A video-driver system designed using a top-down, constraint-driven, methodoly. In *Proceedings of the 33rd Annual Design Automation Conference*, 1996.

[26] J. Rijmenants et al. ILAC: An automated layout tool for analog CMOS circuits. *IEEE Journal of Solid-State Circuits*, 24:417–425, April 1989.

[27] J. W. Bandler et al. Robustizing circuit optimization using huber functions. In *IEEE MTT-S International Microwave Symposium digest.*, pages 1009–1012, 1993.

[28] M. G. R. Degrauwe et al. IDAC: An interactive design tool for analog CMOS circuits. *IEEE Journal of Solid-State Circuits*, 22:1106–115, December 1987.

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

[30] R. Chadha et al. WATOPT: An optimizer for circuit applications. *IEEE Transactions on Computer-Aided Design*, 6:472–479, May 1987.

[31] W.C. Black et al. A high performance low power CMOS channel filter. *IEEE Journal of Solid-State Circuits*, 15:929–938, December 1980.

[32] F. V. Fernández, A. Rodríguez-Vázquez, and J. L. Huertas. Interactive ac modeling and characterization of analog circuits via symbolic analysis. *Analog Integrated Design and Signal Processing*, 1:183–208, November 1991.

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

[34] A. Fiacco and G. McCormick. *Nonlinear programming: sequential unconstrained minimization techniques*. Wiley, 1968. Reprinted 1990 in the SIAM Classics in Applied Mathematics series.

[35] J. P. Fishburn and A. E. Dunlop. TILOS: a posynomial programming approach to transistor sizing. In *Proceedings ICCAD'85*, pages 326–328, 1985.

[36] G. G. E. Gielen, H. C. C. Walscharts, and W. M. C. Sansen. ISAAC: A symbolic simulator for analog integrated circuits. *IEEE Journal of Solid-State Circuits*, 24:1587–1597, December 1989.

[37] 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.

[38] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. 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.

[39] P. R. Gray and R. G. Meyer. MOS operational amplifier design-a tutorial overview. *IEEE Journal of Solid-State Circuits*, 17:969–982, December 1982.

[40] P. R. Gray and R. G. Meyer. *Analysis and Design of Analog Integrated Circuits*. John Wiley & Sons, 3rd. edition, 1993.

[41] R. Gregorian and G. C. Temes. *Analog MOS integrated circuits for signal processing*. John Wiley & Sons, 1st. edition, 1986.

[42] S. K. Gupta and M. M. Hasan. KANSYS: A CAD tool for analog circuit synthesis. In *Proceedings of the 9th International Conference on VLSI Design*, pages 333–334, 1996.

[43] 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.

[44] 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.

[45] M. Hashizume, H. Y. Kawai, K. Nii, and T. Tamesada. Design automation system for analog circuits based on fuzzy logic. In *Proceedings IEEE Custom Integrated Circuit Conference*, pages 4.6.1–4.6.4, 1989.

[46] P. Heikkilã, M. Valtonen, and K. Mannersalo. CMOS op-amp dimensioning using multiphase optimization. In *Proc. IEEE Int. Symp. Circuits and Systems*, pages 167–170, 1988.

[47] M. Hershenson, S. Boyd, and T. H. Lee. CMOS operational amplifier design and optimization via geometric programming. In *Proceedings of the First International Workshop on Design of Mixed-mode Integrated Circuits and Applications*, pages 15–18, Cancun, 1997.

[48] M. Hershenson, S. Boyd, and T. H. Lee. Optimal design of a CMOS op-amp via geometric programming. *Submitted to IEEE Transactions on Computer-Aided Design*, November 1997.

[49] M. Hershenson, S. Boyd, and T. H. Lee. Automated design of folded-cascode op-amps with sensitivity analysis. In *5th IEEE International Conference on Electronics, Circuits and Systems*, Lisbon, September 1998.

[50] M. Hershenson, S. Boyd, and T. H. Lee. GPCAD: A tool for CMOS op-amp synthesis. In *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, San Jose, CA, November 1998.

[51] L. P. Huelsman and G. E. Gielen. *Symbolic Analysis of Analog Circuits: Techniques and Applications*. Kluwer Academic Publishers, 1993.

[52] D. A. Johns and K. Martin. *Analog integrated circuit design*. John Wiley & Sons, 1st. edition, 1997.

[53] G. Jusuf, P. R. Gray, and A. Sangiovanni-Vincentelli. CADICS-cyclic analog-to-digital converter synthesis. In *Proceedings IEEE International Conference on Computer-Aided Design*, pages 286–289, 1990.

[54] S. Khorram, A. Rofougaran, and A. A. Abidi. A CMOS limiting amplifier and signal-strength indicator. In *Symposium on VLSI Circuits Deigest of Technical Papers*, pages 95–96, 1995.

[55] 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.

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

[57] 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.

[58] K. R Laker and W. M. C. Sansen. *Design of Analog integrated circuits and systems*. McGraw-Hill, 1st. edition, 1994.

[59] G. F. Landsburg. A charge-balancing monolithic A/D converter. *IEEE Journal of Solid-State Circuits*, 12:662–673, December 1977.

[60] F. Leyn, W. Daems, G. Gielen, and W. Sansen. Analog circuit sizing with constraint programming modeling and minimax optimization. In *Proceedings IEEE International Symposium on Circuits and Systems*, volume 3, pages 1500–1503, 1997.

[61] S. P. Boyd M. Hershenson, S. S. Mohan and T. H. Lee. Optimization of inductor circuits via geometric programming. In *Design Automation Conference, session 54.3*, pages 994–998, June 1999.

[62] K. Madsen, O. Niedseln, H. Schjaer-Jakobsen, and H. Tharne. Efficient minimax design of networks without using derivatives. *IEEE Transactions on Microwave Theory and Techniques*, 23:803–809, October 1975.

[63] P. C. Maulik and L. R. Carley. High-performance analog module generation using nonlinear optimization. In *Proceedings Fourth Annual IEEE International ASIC Conference Exhibit*, pages T13–5.1–T13–5.2, 1991.

[64] P. C. Maulik, L. R. Carley, and D. J. Allstot. Sizing of cell-level analog circuits using constrained optimization techniques. *IEEE Journal of Solid-State Circuits*, 28:233–241, March 1993.

[65] 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.

[66] P. C. Maulik, M. J. Flynn, D. J. Allstot, and L. R. Carley. Rapid redesign of analog standard cells using constrained optimization techniques. In *Proceedings IEEE Custom Integrated Circuit Conference*, pages 8.1.1–8.1.3, 1992.

[67] F. Medeiro, F.V. Fernández, R. Domínguez-Castro, and A.Rodríguez-Vázquez. A statistical optimization-based approach for automated sizing of analog cells. In *Proceedings of the 31st Annual Design Automation Conference*, pages 594–597, 1994.

[68] T. Mukherjee, L. R. Carley, and R. A. Rutenbar. Synthesis of manufacturable analog circuits. In *Proceedings of the 31st Annual Design Automation Conference*, pages 586–593, 1994.

[69] R. S. Muller and T. H. Kamins. *Device Electronics for Integrated Circuits*. John Wiley & Sons, 2nd. edition, 1986.

[70] B. A. Murtagh and M. A. Saunders. Minos 5.4 user's guide. Technical Report SOL 83-20R, Systems Optimization Laboratory, Stanford University, Stanford, California 94305, December 1983.

[71] F. H. Musa and R. C. Huntington. A CMOS monolithic 3.5 digit A/D converter. In *International Solid-State Conference*, pages 144–145, 1976.

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

[73] Z. Ning, T. Mouthaan, and H. Wallinga. SEAS: A simulated evolution approach for analog circuit synthesis. In *Proceedings IEEE Custom Integrated Circuit Conference*, pages 5.2.1–5.2.4, 1991.

[74] W. Nye, D. C. Riley, A. Sangiovanni-Vincentelli, and A. L. Tits. DELIGHT.SPICE: An optimization-based system for the design of integrated circuits. *IEEE Transactions on Computer-Aided Design*, 7:501–518, April 1988.

[75] W. T. Nye, E. Polak, and A. Sangiovanni-Vincentelli. DELIGHT: An optimization-based computer-aided design system. In *Proceedings IEEE International Symposium on Circuits and Systems*, pages 851–855, 1981.

[76] 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.

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

[78] S. Rabii and B. A. Wooley. A 1.8-V digital-audio sigma-delta modulator in 0.8μm CMOS. *IEEE Journal of Solid-State Circuits*, 32:783–795, June 1997.

[79] J. Rajpogal and D. L. Bricker. Posynomial geometric programming as a special case of semi-infinite linear programming. *Journal of Optimization Theory and Applications*, 66(3):455–475, September 1990.

[80] S. Sapatnekar, V. B. Rao, P. Vaidya, and S.-M. Kang. An exact solution to the transistor sizing problem for CMOS circuits using convex optimization. *IEEE Transactions on Computer-Aided Design*, 12:1621–1634, 1993.

[81] S. S. Sapatnekar. Wire sizing as a convex optimization problem: exploring the area-delay tradeoff. *IEEE Transactions on Computer-Aided Design*, 15(8):1001–1011, 1996.

[82] L. E. Schrage. *User's Manual for linear, integer and quadratic porgramming with LINDO.* Scientific Press, Redwood City, California, 3rd edition, 1987.

[83] S. J. Seda, M. G. R. Degrauwe, and W. Fichtner. A symbolic tool for analog circuit design automation. In *Digest of Technical Papers IEEE International Conference on Computer-Aided Design*, pages 488–491, 1988.

[84] J. Shieh, M. Patil, and B. J. Sheu. Measurement and analysis of charge injection in MOS analog switches. *IEEE Journal of Solid-State Circuits*, 22:277–281, April 1987.

[85] J. Shyu and A. Sangiovanni-Vincentelli. ECSTASY: A new environment for IC design optimization. In *Digest of Technical Papers IEEE International Conference on Computer-Aided Design*, pages 484–487, 1988.

[86] J.-M. Shyu, A. Sangiovanni-Vincentelli, J. P. Fishburn, and A. E. Dunlop. Optimization-based transistor sizing. *IEEE Journal of Solid-State Circuits*, 23(2):400–409, 1988.

[87] J. E. Solomon. The monolithic op amp: a tutorial study. *IEEE Journal of Solid-State Circuits*, 9:969–982, December 1974.

[88] H. Su, C. Michael, and M. Ismail. Statistical constrained optimization of analog MOS circuits using empirical performance tools. In *Proceedings IEEE International Symposium on Circuits and Systems*, volume 1, pages 133–136, 1994.

[89] K. Swings, G. Gielen, and W. Sansen. An intelligent analog IC design system based on manipulation of design equations. In *Proceedings IEEE Custom Integrated Circuit Conference*, pages 8.6.1–8.6.4, 1990.

[90] S. M. Sze. *Physics of Semiconductor Devices*. John Wiley & Sons, 2nd. edition, 1981.

[91] A. Torralba and J. Chávez andl L. G. Franquelo. FASY: A fuzzy-logic based tool for analog synthesis. *IEEE Transactions on Computer-Aided Design*, 15:705–715, July 1996.

[92] L. Trontelj, J. Trontelj, T. Slivnik, R. Sosic, and D. Strle. Analog silicon compiler for switched capacitor filters. In *Proceedings IEEE International Conference on Computer-Aided Design*, pages 506–509, 1987.

[93] P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. D. Reidel, 1987.

[94] L. Vandenberghe, S. Boyd, and A. El Gamal. Optimal wire and transistor sizing for circuits with non-tree topology. In *Proceedings of the 1997 IEEE/ACM International Conference on Computer Aided Design*, pages 252–259, 1997.

[95] L. Vandenberghe, S. Boyd, and A. El Gamal. Optimizing dominant time constant in RC circuits. *IEEE Transactions on Computer-Aided Design*, 2(2):110–125, February 1998.

[96] R. J. Vanderbei. *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers, 1997.

[97] F. Wang and R. Harjani. Optimal design of opamps for oversampled converters. In *Proceedings IEEE Custom Integrated Circuit Conference*, pages 15.5.1–15.5.4, 1996.

[98] D. Wilde and C. Beightler. *Foundations of Optimization*. Prentice-Hall, Englewood Cliffs, NJ, 1967.

[99] M. Wójcikowski, J. Glinianowicz, and M. Bialko. System for optimization of electronic circuits using genetic algorithm. In *Proceedings IEEE International Conference on Electronics, Circuits and Systems*, pages 247–250, 1996.

[100] D. F. Wong, H. W. Leong, and C. L. Liu. *Simulated Annealing for VLSI design*. Kluwer, 1988.

[101] B. Wooley. EE315 course notes. 1996.

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

[103] C. Xinghao and M. L. Bushnell. *Efficient branch and bound search with application to computer-aided design*. Kluwer Academic Publishers, 1996.

[104] X. Xu. Xgp-an optimizer for geometric programming. `ftp://col.biz.uiowa.edu/dist/xu/doc/home.html`, May 1995.

[105] H. Z. Yang, C. Z. Fan, H. Wang, and R. S. Liu. Simulated annealing algorithm with multi-molecule: an approach to analog synthesis. In *Proceedings of the 1996 European Design & Test Conference*, pages 571–575, 1996.

[106] C. Zener. *Engineering Design by Geometric Programming*. Wiley, 1971.

1.A GP0 MOSFET models

In this section we describe the MOSFET large and small signal models used in our method. The model, which we refer to as GP0, is essentially the standard long channel square law model described in, *e.g.*, [40, 3]. This model can be inadequate for short channel transistors (see, *e.g.*, [69, 90]), in which case better models can be developed that still allow optimization via geometric programming; see §1.A.3.

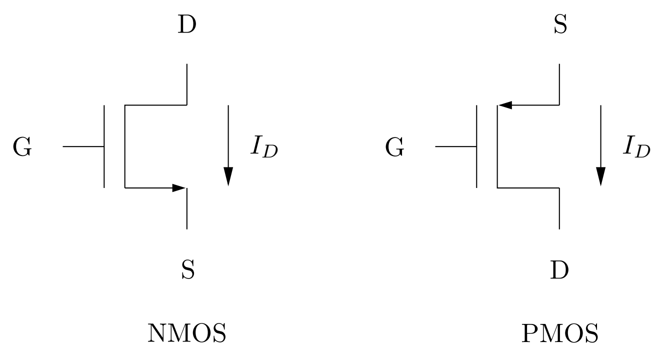

FIGURE 1.10. Transistor symbols

1.A.1 Large signal models

Correct operation of the op-amp requires all transistors to be in saturation. For an NMOS transistor this means

$$V_{\mathrm{DS}} \geq V_{\mathrm{GS}} - V_{\mathrm{TN}}. \tag{1.47}$$

When the NMOS transistor is saturated, *i.e.*, (1.47) holds, the drain current can be expressed as

$$I_{\mathrm{D}} = \frac{1}{2}\mu_{\mathrm{n}}\mathrm{C}_{\mathrm{ox}}\frac{W}{L}\left(V_{\mathrm{GS}} - V_{\mathrm{TN}}\right)^2 (1 + \lambda_n V_{\mathrm{DS}}),$$

where L is the transistor channel length, W is the transistor width, μ_{n} is the electron mobility, C_{ox} is the oxide capacitance per unit area, V_{TN} is the NMOS threshold voltage and λ_n is the channel-length modulation parameter.

In developing our bias constraints, we use the simplified large signal equation

$$I_{\mathrm{D}} = \frac{1}{2}\mu_{\mathrm{n}}\mathrm{C}_{\mathrm{ox}}\frac{W}{L}\left(V_{\mathrm{GS}} - V_{\mathrm{TN}}\right)^2, \tag{1.48}$$

i.e., we ignore channel-length modulation. This introduces only a small error.

For a PMOS transistor, the saturation condition is

$$V_{\mathrm{DS}} \leq V_{\mathrm{GS}} - V_{\mathrm{TP}}. \tag{1.49}$$

The drain current is then given by

$$I_{\mathrm{D}} = \frac{1}{2}\mu_{\mathrm{p}}\mathrm{C}_{\mathrm{ox}}\frac{W}{L}\left(V_{\mathrm{GS}} - V_{\mathrm{TP}}\right)^2\left(1 + \lambda_p V_{\mathrm{DS}}\right),$$

where μ_{p} is the hole mobility, V_{TP} is the PMOS threshold voltage, and λ_p is the channel-length modulation parameter. Here too we ignore the channel modulation effects and use the simplified expression

$$I_{\mathrm{D}} = \frac{1}{2}\mu_{\mathrm{p}}\mathrm{C}_{\mathrm{ox}}\frac{W}{L}\left(V_{\mathrm{GS}} - V_{\mathrm{TP}}\right)^2. \tag{1.50}$$

1.A.2 Small signal models

Figure 1.11 shows the small signal model around the operating point for a MOSFET transistor in saturation. The derivation of this model can also be found in [40]. The values of the various elements and parameters are described below.

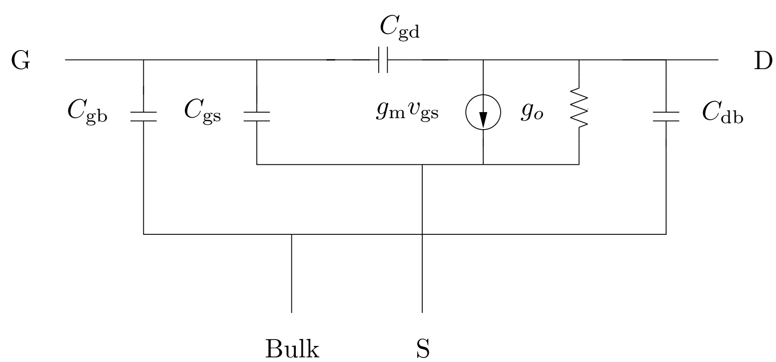

FIGURE 1.11. Small signal model for a MOSFET

The transconductance g_m is given by

$$g_m = \frac{\partial I_{\mathrm{D}}}{\partial V_{\mathrm{GS}}} = \sqrt{2\mu\mathrm{C}_{\mathrm{ox}}I_D\frac{W}{L}} \tag{1.51}$$

(where we ignore, with only small error, channel-length modulation effects). The output conductance g_o is given by

$$g_o = \frac{\partial I_{\mathrm{D}}}{\partial V_{\mathrm{DS}}} = \lambda I_D. \tag{1.52}$$

Note that we ignore channel-length modulation in our transconductance expression, but must include it in the output conductance expression (which would otherwise be zero).

The gate to source capacitance is given by the sum of the gate oxide capacitance and the overlap capacitance:

$$C_{\mathrm{gs}} = \frac{2}{3} W L \mathrm{C}_{\mathrm{ox}} + W L_{\mathrm{D}}, \mathrm{C}_{\mathrm{ox}} \tag{1.53}$$

where L_{D} is the source/drain lateral diffusion length.

The source to bulk capacitance is a junction capacitance and can be expressed as

$$C_{\mathrm{sb}} = \frac{C_{\mathrm{sb0}}}{\left(1 + \frac{V_{\mathrm{SB}}}{\psi_o}\right)^{\frac{1}{2}}}, \tag{1.54}$$

where

$$C_{\mathrm{sb0}} = C_{\mathrm{j}} L_{\mathrm{s}} W + C_{\mathrm{jsw}}(2L_{\mathrm{s}} + W), \tag{1.55}$$

ψ_o is the junction built-in potential, and L_{s} is the source diffusion length.

The drain to bulk capacitance is also a junction capacitance given by

$$C_{\mathrm{db}} = \frac{C_{\mathrm{db0}}}{\left(1 + \frac{V_{\mathrm{DB}}}{\psi_o}\right)^{\frac{1}{2}}}, \tag{1.56}$$

where $C_{\mathrm{db0}} = C_{\mathrm{sb0}}$ for equal source and drain diffusions.

The gate to drain capacitance is due to the overlap capacitance and is given by

$$C_{\mathrm{gd}} = \mathrm{C}_{\mathrm{ox}} W L_{\mathrm{D}}. \tag{1.57}$$

Equations (1.53), (1.55), and (1.57) are posynomial in the design variables, and therefore are readily handled. The expressions for the junction capacitances (1.54) and (1.56) are not posynomial, except in the special case where V_{SB} and V_{DB} do not depend on the design variables. We can take two approaches to approximating these capacitances. One simple method is to take a worst-case analysis, and use the maximum values (which decreases bandwidth, slew rate, phase margin, etc.) This corresponds to the approximation $V_{\mathrm{SB}} = 0$ or $V_{\mathrm{DB}} = 0$. It is also possible to estimate the various junction voltages as constant, so (1.54) and (1.56) are constant.

In our op-amp circuit, the only junction capacitances that appear in the design equations (see §1.5) are the drain to bulk capacitances of M_1, M_2, M_3, M_4, M_6 and M_7. We have estimated the drain to bulk voltages of transistors M_1, M_2, M_3, M_4, M_6 and M_7, and use these estimated voltages for calculating the junction capacitances.

The bulk terminal of the PMOS transistors is connected to the positive supply, V_{DD}, and that of the NMOS transistors is connected to the negative supply, $V_{\mathrm{SS}} = 0$. The drain voltages of M_1, M_2, M_3 and M_4 are the same as the gate voltage of M_6, namely $V_{\mathrm{G},6}$. In most designs, $V_{\mathrm{G},6}$ is a few hundred millivolts above V_{TN} (recalling that we assume $V_{\mathrm{SS}} = 0$). Thus we can write $V_{\mathrm{G},6}$ as

$$V_{\mathrm{G},6} = V_{\mathrm{TN}} + \Delta V_o \tag{1.58}$$

where we use a typical overdrive voltage of $\Delta V_o = 200\mathrm{mV}$. The drain to bulk capacitances of M_1, M_2, M_3 and M_4 are then given by the expressions

$$\begin{aligned} C_{\mathrm{db},1} &= C_{\mathrm{db},2} &= \frac{C_{\mathrm{dbo},1}}{\left(1 + \frac{V_{\mathrm{DD}} - V_{\mathrm{TN}} - \Delta V_o}{\psi_o}\right)^{\frac{1}{2}}} \\ C_{\mathrm{db},3} &= C_{\mathrm{db},4} &= \frac{C_{\mathrm{dbo},3}}{\left(1 + \frac{V_{\mathrm{TN}} + \Delta V_o}{\psi_o}\right)^{\frac{1}{2}}}. \end{aligned}$$

The drain voltage of M_6 and M_7 is the output voltage of the amplifier. The quiescent output voltage is at mid-supply for an op-amp with small offset. Then, we can write $V_{\mathrm{D},6}$ as

$$V_{\mathrm{D},6} = \frac{V_{\mathrm{DD}}}{2}$$

and we obtain constant expressions for C_{db6} and C_{db7}

$$C_{\mathrm{db},6} = \frac{C_{\mathrm{dbo},6}}{\left(1 + \frac{V_{\mathrm{DD}}}{2\psi_o}\right)^{\frac{1}{2}}} \qquad C_{\mathrm{db},7} = \frac{C_{\mathrm{dbo},7}}{\left(1 + \frac{V_{\mathrm{DD}}}{2\psi_o}\right)^{\frac{1}{2}}}.$$

These approximations can be validated in several ways. First, we have observed that changing these typical voltages has very little effect on the final designs. And second, SPICE simulation (which includes the junction capacitances) reveals that we incur only small errors.

1.A.3 GP1 models

The GP0 models described above are essentially the same as the standard long channel device models. It is also possible to derive device models that are more accurate than the long channel models, but at the same time are compatible with geometric programming based design.

Analysis of the errors incurred by the GP0 model shows that most of the modeling error comes from the expressions for transconductance and output conductance. By fitting monomial expressions to empirical data, or data obtained from a high fidelity SPICE simulation, we obtain transistor models that are still compatible with geometric programming based design. We refer to these models as GP1.

We found that the following simple models work very well. For NMOS devices, we use the monomial expression

$$g_{\mathrm{d,NMOS}} = 3.1 \cdot 10^{-2} W^{0.18} L^{-1.14} I_{\mathrm{D}}^{0.82},$$

where the output conductance is given in millisiemens, the bias current is in milliamps, and the width and length are in μm. This simple model provides a very good fit over a wide range of transistor width, length, and bias current. For PMOS devices we find it useful to use two models, one model ($g_{\mathrm{d1,PMOS}}$) for devices operating at low drain-to-source voltage (M_5 and M_8) and another one ($g_{\mathrm{d2,PMOS}}$) for devices operating at high drain-to-source voltage (M_1, M_2 and M_7):

$$\begin{aligned} g_{\mathrm{d1,PMOS}} &= 4.5 \cdot 10^{-1} W^{0} L^{-1.58} I_{\mathrm{D}}^{1.04} \\ g_{\mathrm{d2,PMOS}} &= 8.9 \cdot 10^{-2} W^{0.13} L^{-1.97} I_{\mathrm{D}}^{0.87}, \end{aligned}$$

where again the output conductance is given in millisiemens, the bias current is in milliamps, and the width and length are in μm.

For all other circuit parameters, we used the GP0 model described above (although we could easily have improved the models using empirical fits to monomials and posynomials).

1.B Derivation of bias conditions

In this section we derive the bias conditions given in section §1.4.1. We start by noting that the bias current I_{bias} determines the gate to source voltage of the current source devices M_8, M_5, and M_7. Since these three transistors have the same gate to source voltage their currents are simply related by their dimensions:

$$I_5 = \frac{W_5 L_8}{L_5 W_8} I_{\mathrm{bias}}, \qquad I_7 = \frac{W_7 L_8}{L_7 W_8} I_{\mathrm{bias}}.$$

Since the pair M_1 and M_2 is symmetric, the current I_5 splits equally between them, so $I_1 = \frac{I_5}{2}$.

We now derive the bias condition for transistors M_1, M_5, M_6, and M_7. (The conditions for the others are simple, and given directly in §1.4.1.)

- **Transistor** M_1**.** Transistor M_1 is in saturation when $V_{\mathrm{D},1} \le V_{\mathrm{G},1} - V_{\mathrm{TP}}$. The gate voltage of M_3, $V_{\mathrm{G},3}$, is the same as the drain voltage of transistor M_1, $V_{\mathrm{D},1}$. This voltage can be expressed as

$$V_{\mathrm{G},3} = V_{\mathrm{G},4} = V_{\mathrm{D},1} = \sqrt{\frac{I_1 L_3}{\mu_{\mathrm{n}} C_{\mathrm{ox}}/2W_3}} + V_{\mathrm{TN}} + V_{\mathrm{ss}}.$$

Then, the condition for M_1 being in saturation is

$$\sqrt{\frac{I_1 L_3}{\mu_{\rm n} C_{\rm ox}/2W_3}} \le V_{\rm cm,min} + V_{\rm ss} - V_{\rm TP} - V_{\rm TN}.$$

where $V_{\rm cm,min}$, the lowest value of common-mode input voltage represents the worst case for keeping transistor M_1 in saturation.

- **Transistor M_5.** Transistor M_5 is in saturation when $V_{\rm D,5} \le V_{\rm G,5} - V_{\rm TP}$. The drain voltage of M_5 is equal to the source voltage of transistors M_1 and M_2 and can be expressed as

$$V_{\rm D,5} = V_{\rm S,1} = V_{\rm cm} - V_{\rm TP} + \sqrt{\frac{I_1 L_1}{\mu_{\rm p} C_{\rm ox}/2W_1}}.$$

The gate voltage of M_5 can be expressed as

$$V_{\rm G,5} = V_{\rm DD} + V_{\rm TP} - \sqrt{\frac{I_5 L_5}{\mu_{\rm p} C_{\rm ox}/2W_5}}.$$

Then, the condition for M_5 being in saturation can be written as

$$\sqrt{\frac{I_1 L_1}{\mu_{\rm p} C_{\rm ox}/2W_1}} + \sqrt{\frac{I_5 L_5}{\mu_p C_{\rm ox}/2W_5}} \le V_{\rm dd} - V_{\rm cm,max} + V_{\rm TP}$$

where $V_{\rm cm,max}$, the highest value of common-mode voltage, has been used since it represents the worst case for keeping transistor M_5 in saturation.

- **Transistor M_6.** Transistor M_6 is in saturation when $V_{\rm D,6} \ge V_{\rm G,6} - V_{\rm TN}$. The gate voltage of transistor M_6 is determined by the dimensions of transistor M_6 and by its drain current,

$$V_{\rm G,6} = \sqrt{\frac{I_7 L_6}{\mu_{\rm n} C_{\rm ox}/2W_6}} + V_{\rm TN}.$$

The minimum output voltage, $V_{\rm out,min}$, is the lowest value for the drain voltage of transistor M_6 and therefore it determines when transistor M_6 enters the linear region. The condition for M_6 being in saturation is then

$$\sqrt{\frac{I_7 L_6}{\mu_{\rm n} C_{\rm ox}/2W_6}} \le V_{\rm out,min} - V_{\rm ss}$$

- **Transistor M_7.** Transistor M_7 is in saturation when $V_{\mathrm{D},7} \leq V_{\mathrm{G},7} - V_{\mathrm{TP}}$. The gate voltage of transistor M_7 is also determined by its dimensions and its drain current,

$$V_{\mathrm{G},7} = -\sqrt{\frac{I_7 L_7}{\mu_{\mathrm{p}} C_{\mathrm{ox}}/2W_7}} + V_{\mathrm{TP}} + V_{\mathrm{dd}}.$$

The maximum output voltage, $V_{\mathrm{out,max}}$, is the highest value for the drain voltage of transistor M_7 and therefore it determines when transistor M_7 enters the linear region. The condition for M_7 being in saturation is then

$$\sqrt{\frac{I_7 L_7}{\mu_{\mathrm{p}} C_{\mathrm{ox}}/2W_7}} \leq V_{\mathrm{dd}} - V_{\mathrm{out,max}}.$$

1.C Technology parameters

Table 1.9 shows the HSPICE Level 1 parameters for the technology used.

Parameter	Value
Oxide thickness	20nm
Lateral diffusion	0.2μm
NMOS	
Threshold voltage	0.7V
Channel length modulation	0.03^{-1}
Low-field electron mobility	$600\text{cm}^2/(\text{V sec})$
Body effect factor	$0.298\text{V}^{0.5}$
Surface inversion potential	0.688V
Diffusion sheet resistance	$40\Omega/\square$
Zero-bias bulk junction capacitance	$271\mu\text{F/m}^2$
Zero-bias sidewall bulk junction capacitance	600μF/m
Bulk junction grading coefficient	0.5
Bulk junction contact potential	0.904V
Gate-drain overlap capacitance	346μF/m
Gate-source overlap capacitance	346μF/m
Flicker noise coefficient	$4e^{-24}\text{V}^2\text{F}$
PMOS	
Threshold voltage	-0.9V
Channel length modulation	0.06^{-1}
Low-field electron mobility	$200\text{cm}^2/(\text{V sec})$
Body effect factor	$0.471\text{V}^{0.5}$
Surface inversion potential	0.730V
Diffusion sheet resistance	$60\Omega/\square$
Zero-bias bulk junction capacitance	$423\mu\text{F/m}^2$
Zero-bias sidewall bulk junction capacitance	1.2nF/m
Bulk junction grading coefficient	0.5
Bulk junction contact potential	0.928V
Gate-drain overlap capacitance	346μF/m
Gate-source overlap capacitance	346μF/m
Flicker noise coefficient	$2e^{-24}\text{V}^2\text{F}$

TABLE 1.9. HSPICE Level 1 technology parameters