ORIGINAL PAPER

Disciplined geometric programming

Akshay Agrawal[1] · Steven Diamond[2] · Stephen Boyd[1]

© Springer-Verlag GmbH Germany, part of Springer Nature 2019

Abstract

We introduce log-log convex programs, which are optimization problems with positive variables that become convex when the variables, objective functions, and constraint functions are replaced with their logs, which we refer to as a log-log transformation. This class of problems generalizes traditional geometric programming and generalized geometric programming, and it includes interesting problems involving nonnegative matrices. We give examples of log-log convex functions, some well-known and some less so, and we develop an analog of disciplined convex programming, which we call disciplined geometric programming. Disciplined geometric programming is a subclass of log-log convex programming generated by a composition rule and a set of functions with known curvature under the log-log transformation. Finally, we describe an implementation of disciplined geometric programming as a reduction in CVXPY 1.0.

1 Introduction

1.1 Geometric and generalized geometric programs

A *geometric program* (GP) is a nonlinear mathematical optimization problem in which all the variables are positive and the objective and constraint functions are either

---

✉ Akshay Agrawal
  akshayka@cs.stanford.edu

Steven Diamond
diamond@cs.stanford.edu

Stephen Boyd
boyd@stanford.edu

[1] Department of Electrical Engineering, Stanford University, 450 Serra Mall, Stanford, CA 94305, USA

[2] Department of Computer Science, Stanford University, 450 Serra Mall, Stanford, CA 94305, USA

monomial functions or *posynomial functions*. A monomial is any real-valued function given by $x \mapsto c x_1^{a_1} x_2^{a_2} \ldots x_n^{a_n}$, where $x = (x_1, x_2, \ldots, x_n)$ is a vector of positive real variables, the coefficient c is positive, and the exponents a_i are real; a posynomial function is any sum of monomial functions. A GP is an optimization problem of the form

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

where the functions f_i are posynomials, the functions g_i are monomials, and $x \in \mathbf{R}_{++}^n$ is the decision variable. ($\mathbf{R}_{++}$ denotes the set of positive reals.)

The problem (1) is not convex, but it can be transformed to a convex optimization problem by a well-known transformation. We can make the change of variables $u = \log x$ (meant elementwise) and take the logarithm of the objective and constraint functions to obtain the equivalent problem

$$
\begin{aligned}
\text{minimize} \quad & \log f_0(e^u) \\
\text{subject to} \quad & \log f_i(e^u) \le 0, \quad i = 1, \ldots, m \\
& \log g_i(e^u) = 0, \quad i = 1, \ldots, p,
\end{aligned}
\tag{2}
$$

which can be verified to be convex [7, §4.5.3]. (The exponential e^u is meant elementwise.) Because GPs are reducible to convex programs, they can be solved efficiently and reliably using any algorithm for convex optimization, such as interior-point methods [36] or first-order methods [6]. When all f_i are monomials, the problem (2) reduces to a general linear program (LP), so GP is a generalization of LP.

Since its introduction four decades ago [16], geometric programming has found application in chemical engineering [13], environment quality control [22], digital circuit design [4], analog and RF circuit design [23,30,46], transformer design [26], communication systems [11,12,28], biotechnology [32,45], epidemiology [41], optimal gas flow [33], tree-water-network control [40], and aircraft design [8,24,42]. This list is far from exhaustive; for many other examples, see §10.3 of [5].

Evidently monomials and posynomials are closed under various operations. For example, monomials are closed under multiplication, division, and taking powers, while posynomials are closed under addition, multiplication, and division by monomials. A *generalized posynomial* is defined as a function formed from monomials using the operations addition, multiplication, positive power, and maximum. Generalized posynomials, which include posynomials, are also convex under a logarithmic change of variable, after taking the log of the function. It follows that a *generalized geometric program* (GGP), *i.e.*, a problem of the form (1), with f_i generalized posynomials and g_i monomials, transforms to a convex problem in (2) [5, §5], and therefore is tractable.

1.2 Log-log convex programs

For a function $f : D \to \mathbf{R}_{++}$, with $D \subseteq \mathbf{R}_{++}^n$, we refer to the function $F(u) = \log f(e^u)$, with domain $\{u \mid e^u \in D\}$, as its *log-log transformation*. We refer to a

Fig. 1 Hierarchy of optimization problems
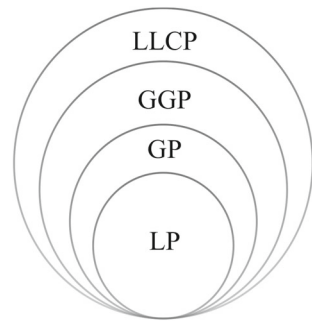
function f as *log-log convex* if F is convex, *log-log concave* if F is concave, and *log-log affine* if F is affine. As in convex analysis, we can consider the analog of extended-value extensions [7, §3.1.2]: we allow a log-log convex function to take the value $+\infty$, and a log-log concave function to take the value zero, which corresponds to F taking the value $-\infty$. A function is log-log affine if and only if it is a monomial; posynomials and generalized posynomials are log-log convex, but there are log-log convex functions that are not generalized posynomials (examples are given in §2.3 and §2.4).

An optimization problem of the form (1), with f_i log-log convex and g_i log-log affine, is called a *log-log convex program* (LLCP). The set of LLCPs is a strict superset of GGPs. The hierarchy of LPs, GPs, GGPs, and LLCPs is shown in Fig. 1.

Log-log convexity is also known as *geometric convexity* or *multiplicative convexity*, since it is equivalent to convexity with respect to the geometric mean (see §2.1). Montel [34] studied the class of log-log convex functions many decades ago, in the context of subharmonic functions. More recently, Niculescu [37] developed a theory of inequalities derived from log-log convexity, parallel to the theory of convex functions, Förster and Nagy [17] studied the log-log convexity of certain operator polynomials, and Baricz [3] examined the log-log concavity of various univariate probability distributions. See also [15,27,39] for related work.

Many functions can be well-approximated by log-log convex functions [5,10,25], but the lack of a coherent modeling framework for LLCPs has hindered their use in practical applications. The point of this paper is to close that gap.

1.3 Domain-specific languages for convex optimization

Disciplined convex programming (DCP) describes a subset of convex programs generated by a single rule and a set of *atoms*, functions with known curvature (convex, concave, or affine) and monotonicity [21]. DCP is a natural starting point for building a domain-specific language (DSL) for convex optimization, *i.e.*, a programming language that parses convex optimization problems expressed in a human-readable form, rewrites them into canonical forms, and supplies the lowered representations to numerical solvers. By abstracting away solvers, DSLs make optimization accessible to researchers and engineers who are not experts in the details of optimization algorithms. Most DSLs for convex optimization have DCP as their foundation; examples
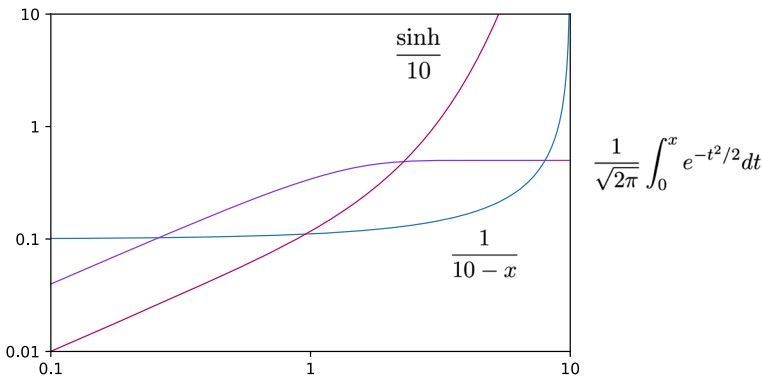
Fig. 2 Two log-log convex functions and one log-log concave function

include CVX [20], CVXPY [1,14], Convex.jl [44], and CVXR [18]. For a survey of DSLs for convex optimization, see [1, §1]. Some DSLs, like CVX and Yalmip [31], can also parse GPs and GGPs. There also exist DSLs specifically for GPs, including GPKit [9] and GGPLAB [35]. These software packages parse and rewrite GPs and GGPs.

In this paper, we introduce the analog of DCP for log-log convex problems. We refer to our analog of DCP as *disciplined geometric programming* (DGP). Like DCP, every disciplined geometric program is generated by a single rule and a library of atoms. The class of disciplined geometric problems is a subclass of log-log convex problems (and of course depends on the library of atoms), and, with a sensible atom library, a strict superset of both geometric programming and generalized geometric programming. In §2, we characterize log-log convexity and give many examples of log-log convex functions, some obvious and some less so; when appropriate, we also supply graph implementations [19]. In §3, we present DGP, along with a verification procedure that we articulate in terms of mathematical expression trees. We close in §4 by describing an implementation of DGP as a reduction to disciplined convex programs in CVXPY 1.0.

2 Log-log convexity

2.1 Properties

Convexity with respect to the geometric mean Log-log convex functions obey a variant of Jensen's inequality: a function f is log-log convex if and only if for all x, y in the domain of f, and for each $\theta \in [0, 1]$,

$$f(x^\theta \circ y^{1-\theta}) \leq f(x)^\theta f(y)^{1-\theta},$$

where $\circ$ is the Hadamard (elementwise) product and the powers are meant elementwise.

Scalar log-log convex functions A scalar function $f : D \to \mathbf{R}_{++}$, $D \subseteq \mathbf{R}_{++}$, is log-log convex if its graph has positive curvature on a log-log plot, as shown in Fig. 2.

If f is additionally twice-differentiable, then it is log-log convex if and only if for all $x \in D$,

$$f''(x) + \frac{f'(x)}{x} \geq \frac{f'(x)^2}{f(x)}. \tag{3}$$

Epigraph If the set $\{u \mid e^u \in D\}$ is convex, $D \subseteq \mathbf{R}_{++}^n$, we say that D is a *log-convex* set. The domain of a log-log convex function f is of course a log-convex set. Its epigraph

$$\mathbf{epi} \, f = \{ (x, t) \mid f(x) \leq t \}$$

is also a log-convex set. The converse is true as well: if the epigraph of a function is a log-convex set, then the function is log-log convex. These facts follow from the similar rules for convex functions and epigraphs [7, §3.1.7].

Relationship to log-convexity Log-log convex functions are related to log-convex functions, which are real-valued functions f for which $\log f$ is convex [7, §3.5]. If f is log-convex and nondecreasing in each of its arguments, then its log-log transformation $F(u) = \log f(e^u)$ is log-log convex, as can be seen via the vector composition rule for convex functions [7, §3.2.4]. Similarly, if f is log-concave and nonincreasing in its arguments, then its log-log transformation is log-log concave. Since every positive concave function is log-concave, it follows that every positive concave function that is nonincreasing in its arguments is also log-log concave.

In some cases, log-log convexity implies log-convexity. A function f is log-convex if and only if for all x and y in its domain and for each $\theta \in [0, 1]$,

$$f(\theta x + (1 - \theta)y) \leq f(x)^\theta f(y)^{1-\theta}.$$

In light of this fact and the AM-GM inequality, every nonincreasing log-log convex function is also log-convex, and every nondecreasing log-log concave function is also log-concave.

Partial minimization If f is log-log convex in the variables x and y, and if D is a log-convex set, then the function

$$g(x) = \inf_{y \in D} f(x, y)$$

is also log-log convex. A similar result holds for log-log concave functions: if $f(x, y)$ is log-log concave and D is a log-convex set, then $g(x) = \sup_{y \in D} f(x, y)$ is log-log concave. These results are translations of identical results for convex functions [7, §3.2.5].

Integration If $f : [0, a) \to [0, \infty)$ is continuous and log-log convex (log-log concave) on $(0, a)$, then

$$x \mapsto \int_0^x f(t)dt$$

is also log-log convex (log-log concave) on $(0, a)$ [34,37]. As an example, if X is a real-valued random variable with a continuous log-log concave density f defined on $[0, a)$, then the probability that X lies between 0 and some $x \in (0, a)$ is a log-log concave function of x. Several common distributions, including the Gaussian, Gibrat, and the Student's t, have log-log concave densities [3, §5].

2.2 Composition rule

A basic result of convex analysis is that a nondecreasing convex function of a convex function is convex. (Similarly, a nonincreasing convex function of a concave function is convex.) These results, along with similar ones for concave functions, are special cases of just one result on the curvature of function compositions, and it is on this single result that DCP is based [21, §6.4]. An analogous composition rule holds for log-log convex functions, which we provide in full generality below. Its proof is an elementary exercise in convex analysis.

Suppose $h : D \to \mathbf{R}_{++} \cup \{\infty\}$, $D \subseteq \mathbf{R}_{++}^k$, is log-log convex, nondecreasing in its ith argument for each i in an index set $I \subseteq \{1, 2, \ldots, k\}$, and nonincreasing in the arguments indexed by I^c. For $i = 1, 2, \ldots, k$, let $g_i : D_i \subseteq \mathbf{R}_{++}^n \to \mathbf{R}_{++}$. Let $f : \bigcap D_i \to \mathbf{R}_{++} \cup \{\infty\}$ be given by

$$f(x) = h(g_1(x), g_2(x), \ldots, g_k(x)).$$

If g_i is log-log convex for $i \in I$ and log-log concave for $i \in I^c$, then the function f is log-log convex.

A symmetric result holds when $h : D \to \mathbf{R}_+$, $D \subseteq \mathbf{R}_+^k$, is log-log concave: If g_i is log-log concave for $i \in I$ and log-log convex for $i \in I^c$, then $f(x) = h(g_1(x), \ldots, g_k(x))$ is log-log concave.

2.3 Some simple examples

We have already seen that monomials are log-log affine and that posynomials and generalized posynomials are log-log convex. In this section we provide several other examples of log-log convex and log-log concave functions.

Product The product $f(x_1, x_2) = x_1 x_2$ is log-log affine, since $F(u) = \log(e^{u_1} e^{u_2}) = u_1 + u_2$ is affine. (This is also clear since f is a monomial.) It follows that the product of log-log affine functions is log-log affine, and (since the product is monotone increasing) the product of log-log convex functions is log-log convex, and the product of log-log concave functions is log-log concave.

Ratio The ratio $f(x_1, x_2) = x_1/x_2$ is log-log affine (since it is a monomial), increasing in its first argument and decreasing in its second argument. It follows that the ratio of a log-log convex and a log-log concave function is log-log convex, and that the ratio of log-log concave and a log-log convex function is log-log concave.

Power For $a \in \mathbf{R}$, the function given by x^a is log-log affine in x, since $\log(e^{ax}) = ax$. It follows that a power of a log-log affine function is log-log affine. For $a \geq 0$, the power

of a log-log convex function is log-log convex, and the power of a log-log concave function is log-log concave. For $a < 0$, the power of a log-log convex function is log-log concave, and the power of a log-log concave function is log-log convex.

Sum The function $f(x_1, x_2) = x_1 + x_2$ is log-log convex since $F(u) = \log(e^{u_1} + e^{u_2})$ is convex. It follows that the sum of log-log convex functions is log-log convex. Log-log concavity is *not* in general preserved under addition.

Max and min The function $f(x) = \max_i x_i$ is log-log convex, and the function $f(x) = \min_i x_i$ is log-log concave. Since both are nondecreasing, it follows that the max of log-log convex functions is log-log convex, and the min of log-log concave functions is log-log concave.

Sum largest For $x \in \mathbf{R}^n_{++}$, the sum of the r largest elements in x is log-log convex, since it can be represented as $\max\{ x_{i_1} + x_{i_2} + \cdots + x_{i_r} \mid i_1 < i_2 < \cdots < i_r \}$, which is the max of a finite number of log-log convex functions.

One-minus The function $f(x) = 1 - x$ with domain $(0, 1)$ is log-log concave, as can be seen by noting that f is concave and decreasing in x, or by the fact that the second derivative of its log-log transformation is negative. It is also decreasing in x, so we conclude that if g is log-log convex, $f(g(x)) = 1 - g(x)$ is log-log concave (with domain $\{ x \mid g(x) < 1 \}$).

Difference The function $f(x) = x_1 - x_2$, with domain $\{x > 0 \mid x_1 - x_2 > 0\}$, is log-log concave, increasing in its first argument and decreasing in its second. It follows that the difference of a log-log concave function and a log-log convex function (with obvious domain) is log-log concave.

Geometric mean The geometric mean $f(x) = \left(\prod_{i=1}^n x_i\right)^{1/n}$ is log-log affine, *i.e.*, a monomial. The geometric mean of log-log convex functions is log-log convex, and likewise for log-log concave functions.

Harmonic mean The harmonic mean $f(x) = n(1/x_1 + 1/x_2 + \cdots + 1/x_n)^{-1}$ is log-log concave, since it is the reciprocal of a log-log convex function.

ℓ_p -norm The ℓ_p-norm $\|x\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{1/p}$, $p \geq 1$, is log-log convex for $x \in \mathbf{R}^n_{++}$, since $\|x\|_p$ with the absolute values removed is a posynomial raised to $1/p$.

Exponential and logarithm The function f given by $f(x) = e^x$ for $x > 0$ is log-log convex, since $F(u) = \log f(e^u) = e^u$, which is convex. Similarly, the logarithm function restricted to $(1, \infty)$ is log-log concave.

Entropy The function $f(x) = -x \log x$ with domain $(0, 1)$ is log-log concave, as can be seen via the composition rule.

Functions with positive Taylor expansions Suppose $f : \mathbf{R} \to \mathbf{R}$ is given by a power series $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots$, with $a_i \geq 0$ and radius of convergence R. We restrict f to the domain $(0, R)$. Then f is log-log convex. This is readily shown by noting that the partial sums are posynomials, so f is the pointwise limit of log-log convex functions. As examples, the functions sinh and cosh restricted to $(0, \infty)$, tan,

sec, and csc restricted to $(0, \pi/2)$, arcsin restricted to $(0, 1]$, and $\log((1+x)/(1-x))$ restricted to $(0, 1)$ are all log-log convex.

Complementary CDF of a log-concave density The complementary cumulative distribution function (CCDF) of a log-concave density is log-log concave. This follows from the fact that the CCDF of a log-concave density is log-concave [7, §3.5.2] and nonincreasing. As an example, the CCDF of a Gaussian

$$x \mapsto \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2} dt$$

is log-log concave on $(0, \infty)$. The densities of many common distributions, including the uniform, exponential, chi-squared, and beta distributions, are log-concave. For several other examples, see [2, Table 1].

Gamma function The Gamma function

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$$

is log-convex and nondecreasing for $x \geq 1$ [7, §3.5]. Hence, the restriction $\Gamma|_{[1,\infty)}$ is log-log convex.

2.4 Functions of positive matrices

In the following exposition, all inequalities should be interpreted elementwise. For any two vectors x, y in $\mathbf{R}^n$, $x \leq y$ if and only if the entries of $y - x$ are all nonnegative, and for any two matrices $A, B \in \mathbf{R}^{m \times n}$, we write $A \leq B$ to mean that the entries of $B - A$ are nonnegative. Similarly, $x < y$ means that the entries of $y - x$ are positive, and likewise for $A < B$. If $A > 0$, we will say that A is a positive matrix.

Let $\mathbf{R}_{++}^{m \times n}$ denote the set of positive m-by-n matrices. The log-log transformation of a function $f : D \subseteq \mathbf{R}_{++}^{m \times n} \to \mathbf{R}_{++}^{p \times q}$ is $F(U) = \log f(e^U)$, defined on $\{ U \mid e^U \in D \}$, where the logarithm and exponential are meant elementwise. We say that f is log-log convex if F is convex with respect to $\leq$, *i.e.*, if for any U, V in the domain of F, $\theta \in [0, 1]$

$$F(\theta U + (1 - \theta)V) \leq \theta F(U) + (1 - \theta)F(V).$$

Equivalently, f is log-log convex if for any $X, Y \in D, \theta \in [0, 1]$,

$$f(X^\theta \circ Y^{1-\theta}) \leq f(X)^\theta f(Y)^{1-\theta},$$

where $\circ$ denotes the Hadamard product and the powers are meant elementwise. Informally, we say that f is log-log convex if $f(X)$ has log-log convex entries for each $X \in D$.

Of course, the trace of a positive matrix and the product of positive matrices are both log-log convex functions. More interesting is the link between log-log convexity and

the Perron-Frobenius theorem, which states, among other things, that every positive square matrix has a positive eigenvalue equal to its spectral radius. We provide a few examples below.

Spectral radius Let $X \in \mathbf{R}^{n \times n}$ have positive entries. The Perron-Frobenius theorem states that X has a positive real eigenvalue λ_{pf} equal to its spectral radius, *i.e.*, the magnitude of its largest eigenvalue. It turns out that λ_{pf} is a log-log convex function of X. This can be seen by the fact that

$$\lambda_{\mathrm{pf}} = \min\{ \lambda \mid Xv \le \lambda v \quad \text{for some } v > 0 \},$$

where the inequalities are elementwise, which implies that $\lambda_{\mathrm{pf}} \le \lambda$ if and only if

$$\sum_{j=1}^{n} X_{ij} v_j / \lambda v_i \le 1, \quad i = 1, \ldots, n.$$

The lefthand side of the above inequality is a posynomial in X_{ij}, v_i, and λ, hence the epigraph of λ_{pf} is log convex. This result is described in more detail in [7, §4.5.4]. For related material, see [17,29,38,43].

Eye-minus-inverse Let D be the set of positive matrices in $\mathbf{R}^{n \times n}$ with spectral radius $\rho(X)$ less than 1. The function $f : D \to \mathbf{R}^{n \times n}$ given by

$$f(X) = (I - X)^{-1}$$

is log-log convex in X, *i.e.*, $f(X)$ has log-log convex entries. The function f is well-defined: for any square matrix $X \in D$, the power series $I + X + X^2 + \cdots$ converges to $(I - X)^{-1}$. One intuitive way to see that f is log-log convex is to note that every partial sum $s_n(X) = \sum_{i=0}^{n} X^i$ with $n \ge 1$ has posynomial entries, and therefore is log-log convex. Because $s_n \to f$, we obtain that f is log-log convex.

We can also prove that the function f is log-log convex by studying its epigraph. Let $X > 0$ and T be matrices. Then

$$(I - X)^{-1} \le T \tag{4}$$

if and only if there exists a matrix $Y \ge 0$ such that

$$I \le Y - YX, \quad Y \le T. \tag{5}$$

The equivalence between (4) and (5) shows that the epigraph of f is log convex: the set of matrices X, Y, and T satisfying (5) is log convex, and the epigraph of f is the projection of this set onto its first and third (matrix) coordinates. It is clear that (4) implies (5), for if X and T satisfy (4), then X, $Y = (I - X)^{-1}$, and T satisfy (5). For the other direction, assume that the matrices $X > 0$, $Y \ge 0$, and T satisfy (5). Let $\lambda_{\mathrm{pf}} = \rho(X) > 0$ be the Perron-Frobenius eigenvalue of X and let $v > 0$ be a corresponding right eigenvector. Multiplying both sides of (5) by v, we obtain that $v \le (1 - \lambda_{\mathrm{pf}})Tv$. This necessitates that $\lambda_{\mathrm{pf}} = \rho(X) < 1$, which together with the fact

that $X > 0$ implies that $(I - X)^{-1}$ exists and is positive. Multiplying both sides of (5) by $(I - X)^{-1}$ yields (4).

Resolvent For any square matrix X and any scalar $s > 0$ such that s is not an eigenvalue of X, the matrix $(sI - X)^{-1}$ is called the *resolvent* of X. The function $(X, s) \mapsto (sI - X)^{-1}$ is log-log convex in both s and X whenever X has positive entries and $\rho(X) < s$. This can be seen by writing $(sI - X)^{-1}$ as $s^{-1}(I - X/s)^{-1}$.

3 Disciplined geometric programming

While it is intractable to determine whether an arbitrary mathematical program is log-log convex, it is easy to check if a composition of *atoms* (functions with known log-log curvature and monotonicity) satisfies the composition rule given in §2.2. This fact motivates *disciplined geometric programming* (DGP), a methodology for constructing log-log convex programs from a set of atoms. A problem constructed via disciplined geometric programming is called a *disciplined geometric program*. If a problem is a disciplined geometric program, we colloquially say that the problem is DGP.

Like DCP [21], DGP has two key components: an atom library and a grammar for composing atoms. Every function appearing in a disciplined geometric program must be either an atom or a grammatical composition of atoms; a composition is grammatical if it satisfies the rule from §2.2. Concretely, a disciplined geometric program is an optimization problem of the form

$$
\begin{aligned}
\text{minimize} \quad & f_0(x) \\
\text{subject to} \quad & f_i(x) \leq \tilde{f}_i(x), \quad i = 1, \ldots, m \\
& g_i(x) = \tilde{g}_i(x), \quad i = 1, \ldots, p,
\end{aligned}
\tag{6}
$$

where the functions f_i are log-log convex, the functions $\tilde{f}_i$ are log-log concave, the functions g_i and $\tilde{g}_i$ are log-log affine, $x \in \mathbf{R}_{++}^n$ is the decision variable, and all the functions are grammatical compositions of atoms. (A problem where the objective is to maximize a log-log concave function and the constraints are as in (6) is also a disciplined geometric program.) Clearly, every disciplined geometric program is an LLCP, but the converse is not true. This is not a limitation in practice because atom libraries are extensible (*i.e.*, the class of DGP is parameterized by the atom library), and because invalid compositions of atoms can often be appropriately re-expressed.

DGP offers an easy-to-understand prescription for constructing a large class of log-log convex problems. If the product, power, sum, and max functions are taken as atoms, then DGP is equivalent to generalized geometric programming. If other functions from §2.3 and §2.4 are also included, then the set of disciplined geometric programs becomes a strict superset of the set of GGPs. As we shall see in §4, DGP is easily supported in a DCP-based DSL for optimization. For these reasons, it seems sensible to suggest that DGP might replace GPs in the optimization modeling toolbelt.

Verifying whether an optimization problem is DGP involves representing the problem as a collection of mathematical expression trees (one for the objective and one for each constraint), and recursively verifying each expression tree. For example, the problem
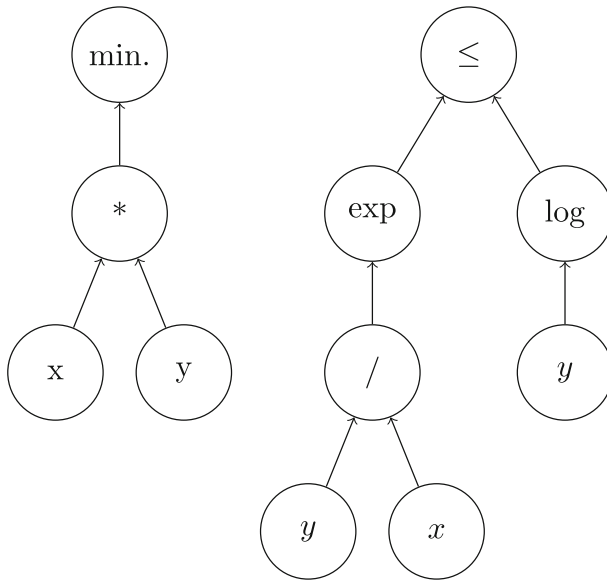
Fig. 3 Expression trees representing the optimization problem (7)

$$
\begin{aligned}
\text{minimize} \quad & xy \\
\text{subject to} \quad & e^{y/x} \leq \log y
\end{aligned}
\tag{7}
$$

can be represented by the expression trees shown in Fig. 3; assuming that the variables x and y are positive, this problem is an LLCP, but it is neither a GP nor a GGP.

An expression tree for an objective is valid if its root is the minimize (maximize) operator and the subtree rooted at its child is a valid log-log convex (log-log concave) composition of atoms. A tree rooted at an atom is valid if the subtrees rooted at its children are valid compositions of atoms, and if the composition of the root with the subtrees of its children is grammatical. Likewise, a tree for an inequality constraint is valid if the left subtree is a valid log-log convex composition of atoms, and the right subtree is a valid log-log concave composition of atoms. A tree for an equality constraint is valid if both subtrees are log-log affine compositions of atoms. The recursion bottoms out at the leaves of each tree, which are variables or constants. Leaves are log-log affine provided that they are positive.

4 Implementation

We have implemented DGP in CVXPY 1.0, a Python-embedded, object-oriented DSL for convex optimization [1]. Our implementation, which is available at

https://www.cvxpy.org,

makes CVXPY 1.0 the first DSL for log-log convex programming.

Our atom library includes a number of the functions presented in §2.3 and §2.4, and our implementation of DGP is a strict superset of generalized geometric programming. CVXPY 1.0 can canonicalize any DGP problem and furnish a solution to it, along with the optimal dual values; it does this by reducing every DGP problem to a DCP problem, canonicalizing and solving the DCP problem, and retrieving a solution to the original problem.

4.1 Canonicalization

In CVXPY 1.0, canonicalization is facilitated by `Reduction` objects, which rewrite problems of one form into equivalent problems of another form and record how to retrieve a solution to the source problem from a solution to the reduced-to problem. Canonicalizing DGP problems in CVXPY 1.0 is simple: we first reduce each DGP problem to a DCP problem, after which we apply the DCP canonicalization procedure.

We have added a class `Dgp2Dcp` that subclasses `Reduction`. `Dgp2Dcp` accepts exactly those problems that are DGP. When applied to a problem, the `Dgp2Dcp` reduction recursively replaces subexpressions with DCP log-log transformations or graph implementations. For example, constants are replaced with their logarithms, positive variables are replaced with unconstrained variables, products of two expressions are replaced with sums of the log-log transformations of those expressions, and sums of expressions are replaced with the `log_sum_exp` of their canonicalized expressions. This procedure makes sense because the log-log transformation of $f = h \circ g$ is equal to the composition of the log-log transformations of h and g.

Atoms like `eye_minus_inv` whose log-log transformations are not DCP are replaced by their graph implementations (a graph implementation of `eye_minus_inv` is given in §2.4). For example, the expression `trace(eye_minus_inv(X))` would be canonicalized to `trace(Y)`, together with the log-log transformation of the constraint `Y U + I <= Y`, where `U` is a variable representing `log X`.

4.2 Solution retrieval

When a DGP problem $\mathcal{P}_1$ is reduced to a DCP problem $\mathcal{P}_2$, for each variable in $\mathcal{P}_1$, a variable representing its logarithm is instantiated in $\mathcal{P}_2$. Given a solution to $\mathcal{P}_2$, *i.e.*, an assignment of numeric values to variables, we recover a solution to $\mathcal{P}_1$ by exponentiating the values of the variables in $\mathcal{P}_2$ and assigning the results to the corresponding variables in $\mathcal{P}_1$. When $\mathcal{P}_2$ is unbounded, $\mathcal{P}_1$ is unbounded as well, in which case the optimal value of the optimization problem is 0 (if $\mathcal{P}_1$ is a minimization problem) or $+\infty$ (if $\mathcal{P}_1$ is a maximization problem). Similarly, $\mathcal{P}_1$ is infeasible when $\mathcal{P}_2$ is infeasible.

The optimal dual values of $\mathcal{P}_1$ are the same as those of $\mathcal{P}_2$. Under certain assumptions, the optimal dual values of $\mathcal{P}_2$ represent fractional changes in the optimal objective given fractional changes in the constraints [5, §3.3].

4.3 Examples

Hello, World. Below is a example of how to use CVXPY 1.0 to specify and solve the DGP problem (7), meant to highlight the syntax of our modeling language. A more interesting example is subsequently presented.

```
1   import cvxpy as cp
2
3   x = cp.Variable(pos=True)
4   y = cp.Variable(pos=True)
5   objective_fn = x * y
6   objective = cp.Minimize(objective_fn)
7   constraints = [cp.exp(y/x) <= cp.log(y)]
8   problem = cp.Problem(objective, constraints)
9   problem.solve(gp=True)
10  print("Optimal value: ", problem.value)
11  print("x: ", x.value)
12  print("y: ", y.value)
13  print("Dual value: ", constraints[0].dual_value)
```

The optimization problem `problem` has two scalar variables, `x` and `y`. For a problem to be DGP, every optimization variable must be declared as positive, as done here with `pos=True`. The objective is to minimize the product of x and y, which is neither convex nor concave but is log-log affine, since the product atom is log-log affine. Every atom is an `Expression` object, which may in turn have references to other `Expressions`; *i.e.*, each `Expression` represents a mathematical expression tree. In line 7, the `Expressions` are represented using three atoms: ratio (`/`), exp, and `log`. Also in line 7, `exp(y/x)` is constrained to be no larger than `log(y)` via the relational operator `<=`, which constructs a `Constraint` object linking two `Expressions`. Line 8 constructs but does not solve `problem`, which encapsulates the expression trees for the objective and constraints. The problem is DGP (which can be verified by asserting `problem.is_dgp()`), but it is not DCP (which can be verified by asserting `not problem.is_dcp()`). Line 9 canonicalizes and solves `problem`. The optimal value of the problem, the values of the variables, and the optimal dual value are printed in lines `10-13`, yielding the following output.

```
Optimal value:  48.81026898447343
x:  11.780089932635645
y:  4.143454698868564
Dual value:  2.843059917747706
```

As this code example makes clear, users do not need to know how canonicalization works. All they need to know is how to construct DGP problems. Calling the `solve` method on a `Problem` instance with the keyword argument `gp=True` canonicalizes the problem and retrieves a solution. If the user forgets to type `gp=True` when her problem is DGP (and not DCP), a helpful error message is raised to alert her of the omission.

Perron-Frobenius matrix completion We have implemented several functions of positive matrices as atoms, including the trace, product, sum, Perron-Frobenius eigenvalue, and eye-minus-inverse. As an example, we can use CVXPY 1.0 to formulate and solve a Perron-Frobenius matrix completion problem. In this problem, we are given *some* entries of an elementwise positive matrix A, and the goal is to choose the missing entries so as to minimize the Perron-Frobenius eigenvalue or spectral radius. Letting Ω denote the set of indices (i, j) for which A_{ij} is known, the optimization problem is

$$
\begin{aligned}
& \text{minimize } \lambda_{\text{pf}}(X) \\
& \text{subject to } \textstyle\prod_{(i,j)\notin\Omega} X_{ij} = 1 \\
& \qquad\qquad X_{ij} = A_{ij}, \ (i, j) \in \Omega,
\end{aligned}
\tag{8}
$$

which is an LLCP. Below is an implementation of the problem (8), with specific problem data

$$
A = \begin{bmatrix} 1.0 & ? & 1.9 \\ ? & 0.8 & ? \\ 3.2 & 5.9 & ? \end{bmatrix},
\tag{9}
$$

where the question marks denote the missing entries.

```
1  import cvxpy as cp
2
3  n = 3
4  known_value_indices = tuple(
5     zip(*[[0, 0], [0, 2], [1, 1], [2, 0], [2, 1]]))
6  known_values = [1.0, 1.9, 0.8, 3.2, 5.9]
7  X = cp.Variable((n, n), pos=True)
8  objective_fn = cp.pf_eigenvalue(X)
9  constraints = [
10    X[known_value_indices] == known_values,
11    X[0, 1] * X[1, 0] * X[1, 2] * X[2, 2] == 1.0,
12 ]
13 problem = cp.Problem(cp.Minimize(objective_fn), constraints)
14 problem.solve(gp=True)
15 print("Optimal value: ", problem.value)
16 print("X:\n", X.value)
```

Executing the above code prints the below output.

```
Optimal value:  4.702374203221535
X:
[[1.          4.63616907 1.9        ]
 [0.49991744 0.8        0.37774148]
 [3.2        5.9        1.14221476]]
```

Compliance with ethical standards

Conflict of interest The authors have no conflict of interest to disclose.

References

1. Agrawal, A., Verschueren, R., Diamond, S., Boyd, S.: A rewriting system for convex optimization problems. J. Control Decis. **5**(1), 42–60 (2018)
2. Bagnoli, M., Bergstrom, T.: Log-concave probability and its applications. Econ. Theory **26**(2), 445–469 (2005)
3. Baricz, Á.: Geometrically concave univariate distributions. J. Math. Anal. Appl. **363**(1), 182–196 (2010)
4. Boyd, S., Kim, S.J., Patil, D., Horowitz, M.: Digital circuit optimization via geometric programming. Op. Res. **53**(6), 899–932 (2005)
5. Boyd, S., Kim, S.J., Vandenberghe, L., Hassibi, A.: A tutorial on geometric programming. Optim. Eng. **8**(1), 67 (2007)
6. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. learn. **3**(1), 1–122 (2011)
7. Boyd, S., Vandenberghe, L.: Convex Optim. Cambridge University Press, New York (2004)
8. Brown, A., Harris, W.: A vehicle design and optimization model for on-demand aviation. In: AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference (2018)
9. Burnell, E., Hoburg, W.: GPkit software for geometric programming. https://github.com/convexengineering/gpkit (2018). Version 0.7.0
10. Calafiore, G., Gaubert, S., Possieri, C.: Log-sum-exp neural networks and posynomial models for convex and log-log-convex data. arXiv (2018)
11. Chiang, M.: Geometric programming for communication systems. Commun. Inf. Theory **2**(1/2), 1–154 (2005)
12. Chiang, M., Tan, C.W., Palomar, D., O'neill, D., Julian, D.: Power control by geometric programming. IEEE Trans. Wirel. Commun. **6**(7), 2640–2651 (2007)
13. Clasen, R.: The solution of the chemical equilibrium programming problem with generalized benders decomposition. Op. Res. **32**(1), 70–79 (1984)
14. Diamond, S., Boyd, S.: CVXPY: a python-embedded modeling language for convex optimization. J. Mach. Learn. Res. **17**(83), 1–5 (2016)
15. Doyle, P., Reeds, J.: The knee-jerk mapping. arXiv (2006)
16. Duffin, R., Peterson, E., Zener, C.: Geometric Programming—Theory and Application. Wiley, Hoboken (1967)
17. Förster, K.H., Nagy, B.: Spectral properties of operator polynomials with nonnegative coefficients. In: Berlin, B.B. (ed.) Operator Theory and Indefinite Inner Product Spaces, pp. 147–162. Springer, Berlin (2005)
18. Fu, A., Narasimhan, B., Boyd, S.: CVXR: An R package for disciplined convex optimization. arXiv (2017)
19. Grant, M., Boyd, S.: Graph implementations for nonsmooth convex programs. In: Blondel, V., Boyd, S., Kimura, H. (eds.) Recent Advances in Learning and Control Lecture Notes in Control and Information Sciences, pp. 95–110. Springer, Berlin (2008)
20. Grant, M., Boyd, S.: CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx (2014)
21. Grant, M., Boyd, S., Ye, Y.: Disciplined convex programming. In: Grant, M. (ed.) Global Optimization, pp. 155–210. Springer, Berlin (2006)
22. Greenberg, H.: Mathematical programming models for environmental quality control. Op. Res. **43**(4), 578–622 (1995)
23. Hershenson, M., Boyd, S., Lee, T.: Optimal design of a CMOS op-amp via geometric programming. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **20**(1), 1–21 (2001)
24. Hoburg, W., Abbeel, P.: Geometric programming for aircraft design optimization. AIAA J. **52**(11), 2414–2426 (2014)

25. Hoburg, W., Kirschen, P., Abbeel, P.: Data fitting with geometric-programming-compatible softmax functions. Optim. Eng. **17**(4), 897–918 (2016)
26. Jabr, R.A.: Application of geometric programming to transformer design. IEEE Trans. Magn. **41**(11), 4261–4269 (2005)
27. Jarczyk, W., Matkowski, J.: On Mulholland's inequality. Proc. Am. Math. Soc. **130**(11), 3243–3247 (2002)
28. Kandukuri, S., Boyd, S.: Optimal power control in interference-limited fading wireless channels with outage-probability specifications. Trans. Wirel. Commun. **1**(1), 46–55 (2002)
29. Kingman, J.: A convexity property of positive matrices. Q. J. Math. **12**(1), 283–284 (1961)
30. Li, X., Gopalakrishnan, P., Xu, Y., Pileggi, L.: Robust analog/RF circuit design with projection-based posynomial modeling. In: Proceedings of the 2004 IEEE/ACM International Conference on Computer-aided Design, ICCAD '04, pp. 855–862. IEEE Computer Society, Washington, DC (2004)
31. Löfberg, J.: YALMIP: A toolbox for modeling and optimization in MATLAB. In: Proceedings of the CACSD Conference. Taipei, Taiwan (2004)
32. Marin-Sanguino, A., Voit, E., Gonzalez-Alcon, C., Torres, N.: Optimization of biotechnological systems through geometric programming. Theor. Biol. Med. Model. **4**(1), 38 (2007)
33. Misra, S., Fisher, M., Backhaus, S., Bent, R., Chertkov, M., Pan, F.: Optimal compression in natural gas networks: a geometric programming approach. IEEE Trans. Control Netw. Syst. **2**(1), 47–56 (2015)
34. Montel, P.: Sur les fonctions convexes et les fonctions sousharmoniques. J. Math. Pures Appl. **9**(7), 29–60 (1928)
35. Mutapcic, A., Koh, K., Kim, S., Boyd, S.: GGPLAB: a matlab toolbox for geometric programming. Available from https://web.stanford.edu/~boyd/ggplab/ (2006)
36. Nesterov, Y., Nemirovski, A.: Interior-point Polynomial Algorithms in Convex Programming. Society for Industrial and Applied Mathematics (1994)
37. Niculescu, C.: Convexity according to the geometric mean. Math. Inequal. Appl. **3**(2), 155–167 (2000)
38. Nussbaum, R.: Convexity and log convexity for the spectral radius. Linear Algebra Appl. **73**, 59–122 (1986)
39. Özdemir, M.E., Yildiz, Ç., Gürbüz, M.: A note on geometrically convex functions. J. Inequal. Appl. **2014**(1), 180 (2014)
40. Perelman, L.S., Amin, S.: Control of tree water networks: a geometric programming approach. Water Resour. Res. **51**(10), 8409–8430 (2015)
41. Preciado, V., Zargham, M., Enyioha, C., Jadbabaie, A., Pappas, G.: Optimal resource allocation for network protection: a geometric programming approach. IEEE Trans. Control Netw. Syst. **1**(1), 99–108 (2014)
42. Saab, A., Burnell, E., Hoburg, W.: Robust designs via geometric programming. arXiv (2018)
43. Tan, C.W.: Wireless network optimization by Perron–Frobenius theory. Found. Trends Netw. **9**(2–3), 107–218 (2015)
44. Udell, M., Mohan, K., Zeng, D., Hong, J., Diamond, S., Boyd, S.: Convex optimization in Julia. In: SC14 Workshop on High Performance Technical Computing in Dynamic Languages (2014)
45. Vera, J., González-Alcón, C., Marín-Sanguino, A., Torres, N.: Optimization of biochemical systems through mathematical programming: methods and applications. Comput. Op. Res. **37**(8), 1427–1438 (2010)
46. Xu, Y., Pileggi, L., Boyd, S.: ORACLE: optimization with recourse of analog circuits including layout extraction. In: Proceedings of the 41st Annual Design Automation Conference, DAC '04, pp. 151–154. ACM, New York, USA (2004)