ORIGINAL PAPER

Disciplined quasiconvex programming

Akshay Agrawal[1] · Stephen Boyd[1]

Abstract

We present a composition rule involving quasiconvex functions that generalizes the classical composition rule for convex functions. This rule complements well-known rules for the curvature of quasiconvex functions under increasing functions and pointwise maximums. We refer to the class of optimization problems generated by these rules, along with a base set of quasiconvex and quasiconcave functions, as *disciplined quasiconvex programs*. Disciplined quasiconvex programming generalizes disciplined convex programming, the class of optimization problems targeted by most modern domain-specific languages for convex optimization. We describe an implementation of disciplined quasiconvex programming that makes it possible to specify and solve quasiconvex programs in CVXPY 1.0.

Keywords Quasiconvex programming · Convex optimization · Domain-specific languages

1 Introduction

A real-valued function f is *quasiconvex* if its domain C is convex, and for any $\alpha \in \mathbf{R}$, its α-sublevel sets $\{ x \in C \mid f(x) \leq \alpha \}$ are convex [5, §3.4]. A function f is quasiconcave if $-f$ is quasiconvex, and it is quasilinear if it is both quasiconvex and quasiconcave. A *quasiconvex program* (QCP) is a mathematical optimization problem in which the objective is to minimize a quasiconvex function over a convex set. Because every convex function is also quasiconvex, QCPs generalize convex programs. Though QCPs are in general nonconvex, many can nonetheless be solved efficiently by a bisection method that involves solving a sequence of convex programs [5, §4.2.5], or by subgradient methods [21,22].

✉ Akshay Agrawal
  akshayka@cs.stanford.edu

  Stephen Boyd
  boyd@stanford.edu

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

The study of quasiconvex functions is several decades old [10,24,25]. Quasiconvexity has been of particular interest in economics, where it arose in the study of competitive equilibria and the modeling of utility functions [3,16]. More recently, quasiconvex programming has been applied to control [6,15,28], model order reduction [29], computer vision [19,20], computational geometry [9], and machine learning [17]. While QCPs have many applications, it remains difficult for non-experts to specify and solve them in practice. The point of this paper is to close that gap.

Domain-specific languages (DSLs) have made convex optimization widely accessible. DSLs let users specify their programs in natural mathematical notation, abstracting away the process of canonicalizing problems to standard forms for numerical solvers. The syntax of most DSLs for convex optimization, including CVX [12], CVXPY [1,8], Convex.jl [30], and CVXR [11], is determined by a grammar known as *disciplined convex programming* (DCP) [13]. DCP includes a set of functions with known curvature (affine, convex, or concave) and monotonicity, and a composition rule for combining the functions to produce expressions that are also convex or concave. Some software does exist for solving quasiconvex problems (e.g., YALMIP [23]), but no DSLs exist for specifying them in a way that guarantees quasiconvexity.

In this paper, we introduce *disciplined quasiconvex programming* (DQCP), an analog of DCP for quasiconvex optimization. Like DCP, DQCP is a grammar that consists of a set of functions and rules for combining them. A contribution of this paper is the development of a theorem for the composition of a quasiconvex function with convex (and concave) functions that guarantees quasiconvexity of the composition. This rule includes as a special case the composition rule for convex functions upon which DCP is based. The class of programs producible by DQCP is a subset of QCPs (and depends on the function library), and a superset of the class corresponding to DCP.

In Sect. 2, we review properties of quasiconvex functions, state our composition theorem, and provide several examples of quasiconvex functions. In Sect. 3, we describe a bisection method for solving QCPs. In Sect. 4, we present DQCP, and in Sect. 5, we describe an implementation of DQCP in CVXPY 1.0.

2 Quasiconvexity

2.1 Properties

In this section, we review basic properties of quasiconvex functions, many of which are parallels of properties of convex functions; see [14] for many more.

Jensen's inequality. Quasiconvex functions are characterized by a kind of Jensen's inequality: a function f mapping a set C into $\mathbf{R}$ is quasiconvex if and only if C is convex and, for any $x, y \in C$ and $\theta \in [0, 1]$,

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

Similarly, f is quasiconcave if and only if C is convex and $f(\theta x + (1 - \theta)y) \geq \min\{f(x), f(y)\}$, for all $x, y \in C$ and $\theta \in [0, 1]$.

Functions on the real line. For $f : C \subseteq \mathbf{R} \to \mathbf{R}$, quasiconvexity can be described in simple terms: f is quasiconvex if it is nondecreasing, nonincreasing, or nonincreasing over $C \cap (\infty, t]$ and nondecreasing over $[t, \infty) \cap C$, for some $t \in C$.

Representation via a family of convex functions. The sublevel sets of a quasiconvex function can be represented as inequalities of convex functions. In this sense, every quasiconvex function can be represented by a family of convex functions. If $f : C \to \mathbf{R}$ is quasiconvex, then there exists a family of convex functions $\phi_t : C \to \mathbf{R}$, indexed by $t \in \mathbf{R}$, such that

$$f(x) \leq t \iff \phi_t(x) \leq 0.$$

The indicator functions for the sublevel sets of f,

$$\phi_t(x) = \begin{cases} 0 & f(x) \leq t \\ \infty & \text{otherwise,} \end{cases}$$

generate one such family. As another example, if the sublevel sets of f are closed, a suitable family is $\phi_t(x) = \inf_{z \in \{z \mid f(z) \leq t\}} \|x - z\|$. We are typically interested in finding families that possess nice properties. For the purpose of DQCP, we seek functions ϕ_t whose 0-sublevel sets can be represented by convex cones over which optimization is tractable.

Partial minimization. Minimizing a quasiconvex function over a convex set with respect to some of its variables yields another quasiconvex function.

Supremum of quasiconvex functions. The supremum of a family of quasiconvex functions is quasiconvex, as can be easily verified [5, §3.4.4]; similarly, the infimum of quasiconcave functions is quasiconcave.

Composition with monotone functions. If $g : C \to \mathbf{R}$ is quasiconvex and h is a nondecreasing real-valued function on the real line, then $f = h \circ g$ is quasiconvex. This can be seen by observing that for any $\alpha \in \mathbf{R}$, a point x (belonging to the domain of f) is in the α-sublevel set of f if and only if

$$g(x) \leq \sup\{ y \mid h(y) \leq \alpha \}.$$

Because g is quasiconvex, this shows that the sublevel sets of f are convex. Similarly, a nonincreasing function of a quasiconvex function is is quasiconcave, a nondecreasing function of a quasiconcave function is quasiconcave, and a nonincreasing function of a quasiconcave function is quasiconvex.

2.2 Composition theorem

A basic result from convex analysis is that a nondecreasing convex function of a convex function is convex; DCP is based on a generalization of this result. The composition rule for convex functions admits a partial extension for quasiconvex functions, which

we state below as a theorem. Though the theorem is straightforward, we are unaware of any references to it in the literature. Of course, the analog of the theorem for quasiconcave functions also holds.

In the statement of the theorem, when considering a function f mapping a subset C of $\mathbf{R}^n$ into $\mathbf{R}^k$, we use $f_1, f_2, \ldots, f_k$ to denote the components of f, i.e., the real functions defined by

$$f(x) = (f_1(x), f_2(x), \ldots, f_k(x)) \quad (x \in C).$$

Theorem 1 *Suppose h is a quasiconvex mapping of a subset C of $\mathbf{R}^k$ into $\mathbf{R} \cup \infty$, and $\{I_1, I_2, I_3\}$ is a partition of $\{1, 2, \ldots, k\}$ such that h is nondecreasing in the arguments indexed by I_1 and nonincreasing in the arguments indexed by I_2, and g maps a subset of $\mathbf{R}^n$ into $\mathbf{R}^k$ in such a way that its components g_i are convex for $i \in I_1$, concave for $i \in I_2$, and affine for $i \in I_3$. Then the composition*

$$f = h \circ g$$

is quasiconvex. If additionally h is convex, then f is convex as well.

The final statement of the theorem is just the well-known composition rule for convex functions.

We provide two proofs of this result. The first proof directly verifies that the domain of f is convex and that f satisfies the modified Jensen's inequality. This proof is almost identical to a proof of the composition theorem for convex functions. The only difference is that an application of Jensen's inequality for convex functions is replaced with its variant for quasiconvex functions. The second proof just applies the composition theorem for convex functions to the representation of a quasiconvex function via a family of convex functions.

Proof (*Proof via Jensen's inequality*) Assume x, y are in the domain of f, and $\theta \in [0, 1]$. Since the components of g are convex or concave (or affine), the convex combination $\theta x + (1 - \theta)y$ is in the domain of g. For $i \in I_1$, the components are convex, so

$$g_i(\theta x + (1 - \theta)y) \leq \theta g_i(x) + (1 - \theta)g_i(y).$$

For $i \in I_2$, the inequality is reversed, and for $i \in I_3$, it is an equality. Since x and y are in the domain of f, $g(x)$ and $g(y)$ are in the domain C of h, and $\theta g(x) + (1 - \theta)g(y) \in C$. Let e_i denote the ith standard basis vector of $\mathbf{R}^k$. Since h is an extended-value function and in light of its per-argument monotonicities, C extends infinitely in the directions $-e_i$ for $i \in I_1$ and e_i for $i \in I_2$. This fact, combined with the inequalities involving the components of g and the fact that $\theta g(x) + (1 - \theta)g(y) \in C$, shows that $g(\theta x + (1 - \theta)y) \in C$. Hence the domain of f is convex.

By the monotonicity of h and Jensen's inequality applied to the components of g,

$$h(g(\theta x + (1 - \theta)y)) \leq h(\theta g(x) + (1 - \theta)g(y)).$$

Because h is quasiconvex,

$$h(\theta g(x) + (1 - \theta)g(y)) \leq \max\{h(g(x)), h(g(y))\}.$$

Hence f is quasiconvex. $\qquad\square$

Proof (*Proof via representation by convex functions*) Let $\phi_t : C \to \mathbf{R}$ be a member of a family of convex functions, indexed by t, such that $\phi_t(x) \leq 0$ if and only if $h(x) \leq t$. Assume without loss of generality that the per-argument monotonicities of ϕ_t match those of h (*e.g.*, take ϕ_t to be the indicator function for the t-sublevel set of h). Then $f(x) = h(g(x)) \leq t$ if and only if $\phi_t(g(x)) \leq 0$. By the composition theorem for convex functions, $\phi_t \circ g$ is convex. We therefore conclude that the sublevel sets of f are convex, *i.e.*, f is quasiconvex. $\qquad\square$

2.3 Examples

Product. The scalar product $f(x, y) = xy$ is quasiconcave when restricted to either $\mathbf{R}_+^2$ or $\mathbf{R}_-^2$, where $\mathbf{R}_+^n$ denotes the set of nonnegative real n-vectors and $\mathbf{R}_-^n$ the set of nonpositive real n-vectors. The product is quasiconvex when one variable is nonnegative and the other is nonpositive. From this fact and the composition rule, one can deduce that the product of two nonnegative concave functions is quasiconcave (see also [4,18]), and the product of a nonnegative concave function with a nonpositive convex function is quasiconvex.

Ratio. The ratio $f(x, y) = x/y$ is quasilinear on $\mathbf{R} \times \mathbf{R}_{++}$, as well as on $\mathbf{R} \times \mathbf{R}_{--}$ (but not on $\mathbf{R}^2$), where $\mathbf{R}_{++}^n$ and $\mathbf{R}_{--}^n$ denote the sets of positive and negative real n-vectors, respectively. When $x \geq 0$ and $y > 0$, f is increasing in x and decreasing in y. Hence the ratio of a nonnegative convex function and a positive concave function is quasiconvex, and the ratio of a nonnegative concave function and a positive convex function is quasiconcave. The problem of maximizing the ratio of a nonnegative concave function and a positive convex function is known as concave-fractional programming [26,27].

Linear-fractional function. The function

$$f(x) = \frac{a^T x + b}{c^T x + d}$$

is quasilinear when the denominator is positive. This can be seen by the composition rule, since the ratio x/y is quasilinear when $y > 0$. It is also quasilinear when restricted to negative denominators. The problem of minimizing a linear-fractional function over a polyhedron is known as linear-fractional programming. Though linear-fractional programming is often described as a generalization of linear programming, linear-fractional programs can be reduced to linear programs [7].

Distance ratio function. The function

$$f(x) = \frac{\|x - a\|_2}{\|x - b\|_2}$$

is quasiconvex on the halfspace $\{ x \in \mathbf{R}^n \mid \|x - a\|_2 \leq \|x - b\|_2 \}$. This result cannot be derived by applying the composition rule to the ratio function, but it is simple to show that its sublevel sets are Euclidean balls [5, §3.4].

Monotone functions on the real line. Monotone functions whose domains are convex subsets of $\mathbf{R}$ are quasilinear; examples include the exponential function, logarithm, square root, and positive odd powers.

Generalized eigenvalue. The maximum eigenvalue of a symmetric matrix is convex, since it can be written as the supremum of a family of linear functions. Analogously, the maximum generalized eigenvalue $\lambda_{\max}(A, B)$ of a pair of symmetric matrices (A, B) (with B positive definite) is quasiconvex, since

$$\lambda_{\max}(A, B) = \sup_{x \neq 0} \frac{x^T A x}{x^T B x}$$

is the supremum of a family of linear-fractional functions [5, §3.4]. Another way to see this is to note that the inequality

$$\lambda_{\max}(A, B) = \sup\{\lambda \in \mathbf{R} \mid Ax = \lambda Bx\} \leq t$$

is satisfied if and only if $tB - A$ is positive semidefinite. Similarly, the minimum generalized eigenvector is quasiconcave in A and B.

2.3.1 Integer-valued functions

Ceiling and floor. The functions $\lceil x \rceil = \inf\{ z \in \mathbf{Z} \mid z \geq x \}$ and $\lfloor x \rfloor = \sup\{ z \in \mathbf{Z} \mid z \leq x \}$ are quasilinear, because they are monotone functions on the real line.

Sign. The function mapping a real number to -1 if it is negative and $+1$ otherwise is quasilinear.

Rectangle. The rectangle function $f : \mathbf{R} \to \mathbf{R}$ given by

$$f(x) = \begin{cases} 0 & |x| > \frac{1}{2} \\ 1 & |x| \leq \frac{1}{2} \end{cases}$$

is quasiconcave.

Length of a vector. The length of a vector in $\mathbf{R}^n$ is defined as the largest index corresponding to a nonzero component:

$$\mathrm{len}(x) = \max\{i \mid x_i \neq 0\}.$$

This function is quasiconvex on $\mathbf{R}^n$ because its sublevel sets are subspaces. The inequality $f(x) \leq \alpha$ implies $x_i = 0$ for $i = \lfloor \alpha \rfloor + 1, \ldots, n$.

Cardinality of a nonnegative vector. The function $\mathrm{card}(x)$, which gives the number of nonzero components in the vector x, is quasiconcave on $\mathbf{R}^n_+$: $\mathrm{card}(x + y) \geq \min\{\mathrm{card}(x), \mathrm{card}(y)\}$ for nonnegative x and y.

Matrix rank. The matrix rank is quasiconcave on the set of positive semidefinite matrices, since the rank of a sum of positive semidefinite matrices is at least the minimum of the ranks of the matrices.

3 Solution method

The problem of minimizing a quasiconvex function $f : C \rightarrow \mathbf{R}$ can be solved in many ways [17,21,22]. Here, we describe a simple method that reduces a QCP to a sequence of convex feasibility problems [5, §4.2.5]. Suppose the interval $[\alpha, \beta]$ is known to contain the optimal value $p^\star$. Put $t = (\alpha + \beta)/2$, and let $\phi_t : C \rightarrow \mathbf{R}$ be a family of convex functions indexed by $t \in \mathbf{R}$ such that $f(x) \leq t$ if and only if $\phi_t(x) \leq 0$. Consider the convex feasibility problem

$$
\begin{aligned}
&\text{find} \quad x \\
&\text{subject to } \phi_t(x) \leq 0.
\end{aligned}
\tag{1}
$$

If this problem yields a feasible point x, then $p^\star \leq t$ and in particular $p^\star \in [\alpha, f(x)]$; otherwise, $p^\star \in [t, \beta]$. In either case, solving the feasibility problem yields an interval containing the optimal value, with width half as large as the original interval. To obtain an ϵ-suboptimal solution to the QCP, we repeat this process until the width of the interval is at most ϵ, which requires at most $\lceil \log_2(\beta - \alpha)/\epsilon \rceil$ iterations. Of course, to apply our method, one must first find a suitable one-parameter family of convex functions ϕ_t to represent the sublevel sets of the objective function f.

Finding an initial interval for bisection. The optimal value $p^\star$ is usually not known before solving a QCP. In such cases, a simple heuristic can be employed to find an interval containing it, assuming that the QCP is feasible (which can be checked by solving a single convex feasibility problem). Start with a candidate interval $[\alpha, \beta]$, where $\alpha < 0$ and $\beta > 0$. If the problem (1) is feasible for $t = \beta$ and infeasible for $t = \alpha$, then $p^\star \in [\alpha, \beta]$. Otherwise, if the problem is infeasible for $t = \beta$, put $\alpha := \beta$ and $\beta := 2\beta$. If on the other hand the problem is feasible for $t = \alpha$, put $\beta := \alpha$ and $\alpha := 2\alpha$. Repeating this process will eventually produce an interval containing $p^\star$, provided that the QCP is not unbounded.

4 Disciplined quasiconvex programming

DQCP is a grammar for constructing QCPs from a set of functions, or *atoms*, with known curvature (affine, convex, concave, quasiconvex, or quasiconcave) and per-argument monotonicities. A program produced using DQCP is called a disciplined quasiconvex program; we say that such programs are DQCP-compliant, or just DQCP, for short. DQCP guarantees that every function appearing in a disciplined quasiconvex program is affine, convex, concave, quasiconvex, or quasiconcave.

A disciplined quasiconvex program is an optimization problem of the form

$$
\begin{aligned}
\text{minimize} \quad & f_0(x) \\
\text{subject to} \quad & f_i(x) \le \alpha_i, \quad i = 1, \ldots, m_1 \\
& \beta_i \le g_i(x), \quad i = 1, \ldots, m_2 \\
& \tilde{f}_i(x) \le \tilde{g}_i(x), \quad i = 1, \ldots, m_3 \\
& h_i(x) = \tilde{h}_i(x), \quad i = 1, \ldots, p.
\end{aligned}
\tag{2}
$$

The functions f_i must be quasiconvex, g_i must be quasiconcave, $\tilde{f}_i$ must be convex, $\tilde{g}_i$ must be concave, and h_i, $\tilde{h}_i$ must be affine; α_i and β_i must be constants. All of the functions appearing in (2) must be produced from atoms, using only the composition rule from Theorem 1 and the rules governing the maximum of quasiconvex functions, the minimum of quasiconcave functions, and composition with monotone functions (see Sect. 2.1). Because Theorem 1 includes the composition rule for convex functions as a special case, DQCP is a modest extension of DCP.

A mathematical expression is verifiably quasiconvex under DQCP if it is

- a convex expression;
- a quasiconvex atom, applied to a variable or constant;
- the max of quasiconvex expressions;
- a nondecreasing function of a quasiconvex expression, or a nonincreasing function of a quasiconcave expression;
- the composition of a quasiconvex atom with convex, concave, and affine expressions that satisfies the hypotheses of Theorem 1.

These rules are applied recursively, with the recursion bottoming out at variables and constants. For example, if $\exp(\cdot)$ and the generalized eigenvalue $\lambda_{\max}(\cdot, \cdot)$ are atoms, and X and Y are matrix variables, then the expressions

$$
\lambda_{\max}(X, Y), \quad \exp(\lambda_{\max}(X, Y)), \quad \text{and} \quad \exp(\exp(\lambda_{\max}(X, Y)))
$$

are all verifiably quasiconvex under DQCP, since $\exp(\cdot)$ is increasing and $\lambda_{\max}(\cdot, \cdot)$ is quasiconvex. Likewise, an expression is quasiconcave under DQCP if it is a concave expression, a quasiconcave atom applied to a variable or constant, the min of quasiconcave functions, a nondecreasing function of a quasiconcave function, a nonincreasing function of a quasiconvex function, or a valid composition of a quasiconcave function with convex, concave, and affine functions. Whether an expression is convex, concave, or affine under DQCP is precisely the same as under DCP.

A DQCP program is naturally represented as a collection of expression trees, one for the objective and one for each constraint. Verifying whether a program is DCQP amounts to recursively verifying that each expression tree is DQCP. For example, the program

$$
\begin{aligned}
\text{minimize} \quad & -\sqrt{x}/y \\
\text{subject to} \quad & \exp(x) \le y
\end{aligned}
\tag{3}
$$

can be represented by the trees shown in Fig. 1. This program is DQCP when y is known to be positive, because the ratio of a nonnegative concave functions and a positive convex function is quasiconcave, and the negation of a quasiconcave function is quasiconvex. The atoms in this program are the functions $\exp(\cdot)$, $\sqrt{\cdot}$, and $\cdot/\cdot$.
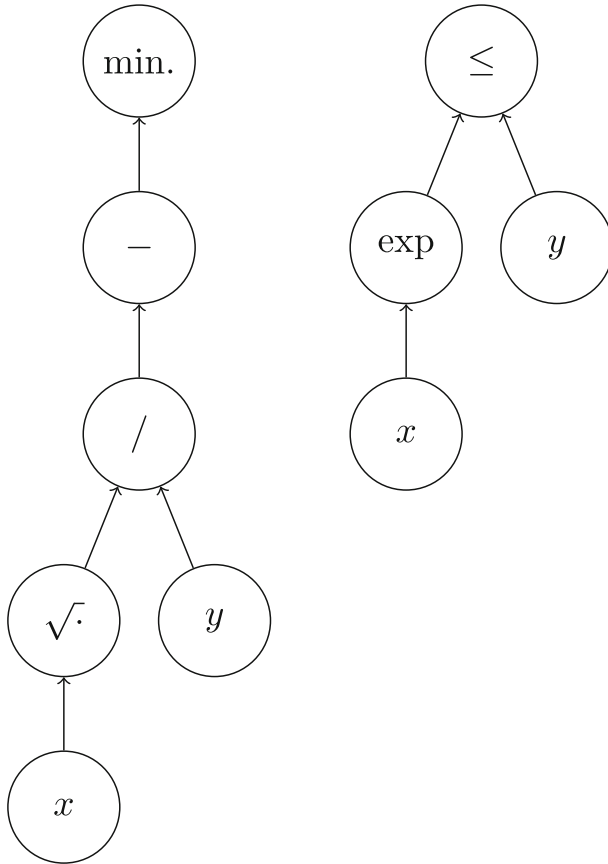
Fig. 1 Expression trees representing the program (3)

The ability of DQCP to express a particular QCP is limited by the atoms available in the library. This issue is a surmountable because the atom library is extensible.

The grammar. Table 1 specifies the DQCP grammar, in the programming languages sense [2, §4]. In the specification, S denotes the start symbol. The symbols

$$\text{AFF,} \quad \text{CVX,} \quad \text{CCV,} \quad \text{QCVX,} \quad \text{QCCV}$$

are nonterminals used to represent affine, convex, concave, quasiconvex, and quasiconcave expressions producible by DQCP. Their lowercase counterparts represent atoms, *e.g.*, cvx stands for a convex atom. Atoms can have multiple curvatures. For example, every affine atom is also a convex atom and a concave atom. The symbols

$$\text{incr,} \quad \text{decr}$$

denote nondecreasing and nonincreasing functions, respectively,

$$\text{constant,} \quad \text{variable}$$

Table 1 The DQCP grammar, which extends DCP

S	→ QCVX
S	→ QCCV
LEAF	→ constant
LEAF	→ variable
AFF	→ LEAF
AFF	→ aff(AFF, ..., AFF)
CVX	→ AFF
CVX	→ cvx(CVX, ..., CVX, CCV, ..., CCV, AFF, ..., AFF)
CCV	→ AFF
CCV	→ ccv(CCV, ..., CCV, CVX, ..., CVX, AFF, ..., AFF)
QCVX	→ CVX
QCVX	→ qcvx(CVX, ..., CVX, CCV, ..., CCV, AFF, ..., AFF)
QCVX	→ incr(QCVX)
QCVX	→ decr(QCCV)
QCVX	→ max{QCVX, ..., QCVX}
QCCV	→ CCV
QCCV	→ qccv(CCV, ..., CCV, CVX, ..., CVX, AFF, ..., AFF)
QCCV	→ incr(QCCV)
QCCV	→ decr(QCVX)
QCCV	→ min{QCCV, ..., QCCV}

The rules for compositions with convex, concave, and affine expressions denote compositions satisfying the hypotheses of Theorem 1

denote numerical constants and optimization variables, and

$$\text{cvx}(\text{CVX}, \dots, \text{CVX}, \text{CCV}, \dots, \text{CCV}, \text{AFF}, \dots, \text{AFF})$$

denotes a composition of a convex atom with convex, concave, and affine expressions that can be certified as convex via Theorem 1. Because DQCP is a grammar for QCPs, it can be used to define the syntax of a DSL for quasiconvex optimization.

5 Implementation

We have implemented DQCP in CVXPY 1.0, a Python-embedded DSL for convex optimization [1,8]. Our implementation, which is available at

https://www.cvxpy.org,

makes CVXPY the first DSL for quasiconvex optimization. Because DQCP is a generalization of DCP, it fits seamlessly into CVXPY, which parses problems using DCP by default. Our atom library includes many of the functions presented in Sect. 2.3. We have also implemented the bisection method described in Sect. 3.

5.1 Canonicalization

The process of rewriting a problem to an equivalent standard form is called canonicalization. In CVXPY 1.0, canonicalization is facilitated by `Reduction` objects, which rewrite problems of one form into equivalent problems of another form.

We have implemented a reduction called `Dqcp2Dcp` that canonicalizes DQCP problems by converting them into an equivalent one-parameter family of DCP feasibility problems. When applied to a DQCP problem, this reduction first introduces a scalar parameter and constrains the problem's objective to be no greater than the parameter. It recursively processes this constraint and every other constraint, representing the sublevel sets of quasiconvex expressions and superlevel sets of quasiconcave expressions in DCP-compliant ways. The reduction then emits a parameterized DCP problem. The constraints of the emitted problem are the canonicalized constraints of the original problem, and the goal is to find an assignment of the variables that satisfies the constraints. A solution to the original problem can be obtained by running bisection on the emitted problem.

5.2 Bisection

We have implemented the bisection routine described in Sect. 3. Our method first checks whether the original problem is feasible by solving a convex feasibility problem. If the problem is feasible, our routine automatically finds an interval containing the optimal value and then runs bisection. Our bisection routine tightens the boundaries of the bisection interval depending on the values of the original problem's objective function. For example, when the objective is integer-valued, our implementation will tighten a lower bound α to $\lceil \alpha \rceil$, and an upper bound β to $\lfloor \beta \rfloor$.

5.3 Examples

Hello, world. Below is an example of how to use CVXPY 1.0 to specify and solve the problem (3), meant to highlight the syntax of our modeling language. More interesting examples are subsequently presented.

```
1  import cvxpy as cp
2
3  x = cp.Variable()
4  y = cp.Variable(pos=True)
5  objective_fn = -cp.sqrt(x)/y
6  objective = cp.Minimize(objective_fn)
7  constraint = cp.exp(x) <= y
8  problem = cp.Problem(objective, [constraint])
9  problem.solve(qcp=True)
10 print("Optimal value: ", problem.value)
11 print("x: ", x.value)
12 print("y: ", y.value)
```

The optimization problem `problem` has two scalar variables, x and y. Notice that y is declared as positive in line 3, with `pos=True`. The objective is to minimize the ratio of $-\sqrt{x}$ and y, which is quasiconvex since the ratio is quasiconcave when the numerator is a nonnegative concave expression and the denominator is a positive convex expression. Line 6 constructs the objective of the problem. In line 7, `exp(x)` is constrained to be no larger than y via the relational operator `<=`. Line 8 constructs `problem`, which represents the optimization problem as two expression trees, one for `objective_fn` and one for `constraint`. The internal nodes in these expression trees are the atoms `sqrt`, `exp`, ratio (`/`), and negation (`-`). The problem is DQCP, which can be verified by asserting `problem.is_dqcp()`. Line 9 canonicalizes `problem`, parsing it as a DQCP (`qcp=True`), and then solves it by bisection. The optimal value of the problem and the values of the variables are printed in lines 10-12, yielding the following output.

```
1  Optimal value:  -0.4288821220397949
2  x:  0.49999737143004713
3  y:  1.648717724845007
```

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

Generalized eigenvalue matrix completion. We have implemented the maximum generalized eigenvalue as an atom. As an example, we can use CVXPY 1.0 to formulate and solve a generalized eigenvalue matrix completion problem. In this problem, we are given *some* entries of two symmetric matrices A and B, and the goal is to choose the missing entries so as to minimize the maximum generalized eigenvalue $\lambda_{\max}(A, B)$. Letting Ω denote the set of indices (i, j) for which A_{ij} and B_{ij} are known, the optimization problem is

$$\text{minimize } \lambda_{\max}(X, Y)$$
$$\text{subject to } X_{ij} = A_{ij}, \ (i, j) \in \Omega,$$
$$Y_{ij} = B_{ij}, \ (i, j) \in \Omega,$$

which is a QCP. Below is an implementation of this problem, with specific problem data

$$A = \begin{bmatrix} 1.0 & ? & 1.9 \\ ? & 0.8 & ? \\ ? & ? & ? \end{bmatrix}, \quad B = \begin{bmatrix} 3.4 & ? & 1.4 \\ ? & 0.2 & ? \\ ? & ? & ? \end{bmatrix}.$$

(The question marks denote the missing entries.)

```
1  import cvxpy as cp
2
3  X = cp.Variable((3, 3))
4  Y = cp.Variable((3, 3))
5  gen_lambda_max = cp.gen_lambda_max(X, Y)
```

```
6   omega = tuple(zip(*[[0, 0], [0, 2], [1, 1]]))
7   constraints = [
8     X[omega] == [1.0, 1.9, 0.8],
9     Y[omega] == [3.0, 1.4, 0.2],
10  ]
11  problem = cp.Problem(cp.Minimize(gen_lambda_max), constraints)
12  problem.solve(qcp=True)
13  print("Generalized eigenvalue: ", gen_lambda_max.value)
14  print("X: ", X.value)
15  print("Y: ", Y.value)
```

Executing the above code prints the below output.

```
Objective:   4.000002716411653
X:   [[9.99999767e-01 9.86154616e-16 1.89999959e+00]
 [9.86154616e-16 7.99999761e-01 5.19126535e-15]
 [1.89999911e+00 5.19126535e-15 1.25733692e+00]]
Y:   [[ 2.99999980e+00 -2.78810135e-16  1.40000015e+00]
 [-2.78810135e-16  1.99999804e-01  2.14473098e-16]
 [ 1.40000015e+00  2.14473098e-16  1.14038551e+00]]
```

Notice that the `gen_lambda_max` atom automatically enforced the symmetry and positive definiteness constraints on X and Y.

Minimum length least squares. Our atom library includes several integer-valued functions, including the length function. As an example, the following QCP finds a minimum-length vector $x \in \mathbf{R}^n$ that has small mean-square error for a particular least squares problem:

$$\text{minimize } \operatorname{len}(x)$$
$$\text{subject to } 1/n\|Ax - b\|_2^2 \leq \epsilon.$$

The problem data are $A \in \mathbf{R}^{n \times n}$, $b \in \mathbf{R}^n$, and $\epsilon \in \mathbf{R}$. Below is an implementation of this problem in CVXPY.

```
1   import cvxpy as cp
2   import numpy as np
3   np.set_printoptions(precision=2)
4
5   n = 10
6   np.random.seed(1)
7   A = np.random.randn(n, n)
8   x_star = np.random.randn(n)
9   b = A @ x_star
10  epsilon = 1e-2
11
12  x = cp.Variable(n)
13  mse = cp.sum_squares(A @ x - b)/n
14  problem = cp.Problem(cp.Minimize(cp.length(x)), [mse <= epsilon])
15  problem.solve(qcp=True)
16  print("Length of x: ", problem.value)
17  print("MSE: ", mse.value)
18  print("x: ", x.value)
19  print("x_star: ", x_star)
```

Running the code produces the following output.

```
Length of x:  8.0
MSE:  0.00926009328775564
x: [-0.26  1.38  0.21  0.94 -1.15  0.15  0.66 -1.16 -0.   -0. ]
x_star: [-0.45  1.22  0.4   0.59 -1.09  0.17  0.74 -0.95 -0.27  0.03]
```

Compliance with ethical standards

Conflict of interest The authors have no conflicts 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. Aho, A., Lam, M., Sethi, R., Ullman, J.: Compilers: Principles, Techniques, and Tools, 2nd edn. Addison-Wesley Longman Publishing Co., Inc, Boston (2006)
3. Arrow, K., Debreu, G.: Existence of an equilibrium for a competitive economy. Econometrica **22**(3), 265–290 (1954)
4. Bector, C.: Programming problems with convex fractional functions. Oper. Res. **16**, 383–391 (1968). https://doi.org/10.1287/opre.16.2.383
5. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, New York (2004)
6. Bullo, F., Liberzon, D.: Quantized control via locational optimization. IEEE Trans. Autom. Control **51**(1), 2–13 (2006)
7. Charnes, A., Cooper, W.: Programming with linear fractional functionals. Naval Res. Logist Q. **9**, 181–186 (1962). https://doi.org/10.1002/nav.3800090303
8. Diamond, S., Boyd, S.: CVXPY: a python-embedded modeling language for convex optimization. J. Mach. Learn. Res. **17**(83), 1–5 (2016)
9. Eppstein, D.: Quasiconvex programming: combinatorial and computational. Geometry **52**(287–331), 3 (2005)
10. Fenchel, W.: Convex cones, sets, and functions. Lecture Notes (1953)
11. Fu, A., Narasimhan, B., Boyd, S.: CVXR: An R package for disciplined convex optimization. arXiv (2017)
12. Grant, M., Boyd, S.: CVX: MATLAB software for disciplined convex programming, version 2.1. http://cvxr.com/cvx (2014)
13. Grant, M., Boyd, S., Ye, Y.: Disciplined convex programming. In: Global Optimization, vol. 84, pp. 155–210. Springer, New York (2006). https://doi.org/10.1007/0-387-30528-9_7
14. Greenberg, H., Pierskalla, W.: A review of quasi-convex functions. Oper. Res. **19**(7), 1553–1570 (1971)
15. Gu, K.: Designing stabilizing control of uncertain systems by quasiconvex optimization. IEEE Trans. Autom. Control **39**(1), 127–131 (1994)
16. Guerraggio, A., Molho, E.: The origins of quasi-concavity: a development between mathematics and economics. Hist. Math. **31**(1), 62–75 (2004)
17. Hazan, E., Levy, K., Shalev-Shwartz, S.: Beyond convexity: Stochastic quasi-convex optimization. In: Advances in Neural Information Processing Systems 28 (NeurIPS '15), pp. 1594–1602 (2015)
18. Kantrowitz, R., Neumann, M.: Optimization for products of concave functions. Rendiconti del Circolo Matematico di Palermo. Serie II **54**(2), 291–302 (2005). https://doi.org/10.1007/BF02874642
19. Ke, Q., Kanade, T.: Uncertainty models in quasiconvex optimization for geometric reconstruction. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), vol. 1, pp. 1199–1205. IEEE (2006)
20. Ke, Q., Kanade, T.: Quasiconvex optimization for robust geometric reconstruction. IEEE Trans. Pattern Anal. Mach. Intell. **29**(10), 1834–1847 (2007)

21. Kiwiel, K.: Convergence and efficiency of subgradient methods for quasiconvex minimization. Math. Program. **90**(1), 1–25 (2001)
22. Konnov, I.: On convergence properties of a subgradient method. Optim. Methods Softw. **18**(1), 53–62 (2003). https://doi.org/10.1080/1055678031000111236
23. Löfberg, J.: YALMIP: a toolbox for modeling and optimization in MATLAB. In: Proceedings of the CACSD Conference. Taipei, Taiwan (2004)
24. Luenberger, D.: Quasi-convex programming. SIAM J. Appl. Math. **16**, 1090–1095 (1968). https://doi.org/10.1137/0116088
25. Nikaidô, H.: On von Neumann's minimax theorem. Pac. J. Math. **4**, 65–72 (1954)
26. Schaible, S.: Analyse und Anwendungen von Quotientenprogrammen, Mathematical Systems in Economics, vol. 42. Verlag Anton Hain, Königstein/Ts. (1978). Ein Beitrag zur Planung mit Hilfe der nichtlinearen Programmierung
27. Schaible, S.: Fractional programming: applications and algorithms. Eur. J. Oper. Res. **7**(2), 111–120 (1981). https://doi.org/10.1016/0377-2217(81)90272-1
28. Seiler, P., Balas, G.: Quasiconvex sum-of-squares programming. In: 49th IEEE Conference on Decision and Control (CDC), pp. 3337–3342 (2010)
29. Sou, K., Megretski, A., Daniel, L.: A quasi-convex optimization approach to parameterized model order reduction. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **27**(3), 456–469 (2008)
30. 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)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.