Symbolic Computation of Fenchel Conjugates

In honour of Alfred Auslender

Jonathan Borwein, Chris Hamilton
{jborwein,chamilton}@cs.dal.ca

30 November 2005

Abstract

Convex optimization is a branch of mathematics dealing with non-linear optimization problems with additional geometric structure. This area has been the focus of considerable recent research due to the fact that convex optimization problems are scalable and can be efficiently solved by interior-point methods. Over the last ten years or so, convex optimization has found new applications in many areas including control theory, signal processing, communications and networks, circuit design, data analysis and finance.

Of key importance in convex optimization is the notion of duality, and in particular that of Fenchel duality. This work explores algorithms for calculating symbolic Fenchel conjugates of a class of real-valued functions defined on $\mathbb{R}^n$.

1 Motivation

To make the development available to a wide variety of practitioners we include the following discussion.

1.1 Definition and Basic Results

Suppose f is a function defined on $\mathbb{R}^n$ that takes on values in $(-\infty, \infty] = \mathbb{R} \cup \{\infty\} = \bar{\mathbb{R}}$. Recall that f is convex if

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2),$$

for every $x_1, x_2 \in \mathbb{R}^n$, and all $\lambda \in [0,1]$. Recall also that the *effective domain* of f, dom f, is the set of all points where f is finite-valued. Convex functions lie at the heart of of convex, functional and real analysis, as well convex optimization. Several excellent overviews of the subject are available, ranging from Rockafeller's [13] and Luenberger's [12] classics, to more modern treatments by Boyd and Vandenberghe [7], and by Borwein and Lewis [5].

Basic calculus teaches that a minimizer $\bar{x}$ of a differentiable function f is necessarily a critical point: $\nabla f(\bar{x}) = 0$. Since many interesting convex functions are not everywhere differentiable, this technique is not available. Instead, one defines a generalized differential, the *subdifferential* of f at x by

$$\partial f(x) = \{y \in \mathbb{R}^n : \langle y, x' - x\rangle \leq f(x') - f(x),\ \forall x' \in \mathbb{R}^n\}.$$

Members of this subdifferential are called *subgradients*, and have a clear geometric interpretation as being slopes of tangential hyperplanes that minorize f at the point x. The importance of subgradients in convex optimization stems from the calculus-like fact that $\bar{x}$ is a global minimizer of f if and only if $0 \in \partial f(\bar{x})$.

Theorem 1.1 (Differentiability and the Subdifferential) *Consider a convex function $f : \mathbb{R}^n \mapsto \mathbb{R}$. The subdifferential generalizes differentiability: f is differentiable at x if and only if $\partial f(x)$ is a singleton, in which case $\partial f(x) = \{\nabla f(x)\}$.*

Proof: Elementary. See any of [12], [13] or [8] for details. ■

Theorem 1.2 (Subdifferential on $\mathbb{R}$) *If f is convex and defined on $\mathbb{R}^n$, then the left (f'_-) and right (f'_+) derivatives exist at every point in dom f. Moreover, the subdifferential at every $x \in$ dom f is a closed interval or singleton completely described by the directional derivatives:*

$$\partial f(x) = [f'_-, f'_+].$$

Proof: Elementary. See any of [12], [13] or [8] for details. ■

The *Fenchel conjugate*, or *Fenchel-Legendre transform*, of f, denoted f^*, is defined by

$$f^*(y) = \sup_{x \in \mathbb{R}^n} \{[y, x] - f(x)\}, \quad \forall y \in \mathbb{R}^n.$$

The fenchel conjugate is always a convex and lower semi-continuous function on $\mathbb{R}^n$. The role of the Fenchel conjugate in convex analysis is extremely important, and draws many parallels with the role of the Fourier analysis in harmonic analysis. Assuming lower semi-continuity and properness of f, the bi-conjugate of f recovers the original function: $f = f^{**}$. In fact, the converse is also true, leading to:

$$f \text{ is convex and lower semi-continuous} \Leftrightarrow f = f^{**}.$$

An immediate consequence of the definition of the Fenchel conjugate is the well-known *Fenchel-Young Inequality*:

$$f(x) + f^*(y) \geq \langle x, y\rangle, \quad \forall x, y \in \mathbb{R}^n. \tag{1}$$

Sufficient and necessary conditions for this to hold with equality may be formulated in terms of subgradients:

$$f(x) + f^*(y) = \langle x, y\rangle \Leftrightarrow y \in \partial f(x) \Leftrightarrow x \in \partial f^*(y).$$

1.2 Optimization and Duality Results

1.2.1 Convex Optimization Duality

Convex optimization deals with primal problems of the type

$$p = \inf_{x \in \mathbb{R}^n} \{f(x) + g(Ax)\},$$

where $A \in \mathbb{R}^{m \times n}$, f is convex and lsc on $\mathbb{R}^n$, and g is convex and lsc on $\mathbb{R}^m$. Fenchel conjugation leads to a natrual dual representation of the problem as

$$d = \sup_{z \in \mathbb{R}^m} \{f^*(-A^*z) + g^*(z)\},$$

where A^* is simply the transpose of A. There are several fundamental results relating these dual formulations, a few of which we will present here.

Theorem 1.3 (Fenchel's Duality Theorem) *Suppose $x \in \mathbb{R}^n$, $z \in \mathbb{R}^m$ and f, g and A are as above. Then the following hold:*

1. *Weak Duality: $p \geq d$.*
2. *Strong Duality: If $A(dom\ f) \cap int\ dom\ g \neq \emptyset$, then $p = d$ and the infimum defining d is attained.*
3. *Primal Solutions: If z is a solution to the dual, then the solutions to the primal are equal to the (possibly empty) set*

$$A^{-1}\partial g^*(z) \cap \partial f^*(-A^*z).$$

Proof: Refer to [5]. ■

Fenchel's Duality Theorem highlights the importance of subdifferentials and Fenchel conjugates. Additionally, it is worth noting that Fenchel Duality is general enough to encompass both Linear Programming Duality and the well known Min-Max Theorem from game theory. For further details refer to [12].

1.2.2 Semi-Definite Optimization

In semi-definite optimization one minimizes a linear function subject to the constraint that an affine combination of symmetric matrices remains positive semi-definite. Such a constraint is inherently non-linear and non-smooth, but convex. Semi-definite optimization is a generalization and a unification of several standard problems (linear, quadratic and convex optimization) and as such finds many applications. Although much more general than linear programming, they may aren't much harder to solve.

In semi-definite optimization the primal problem may be stated as

$$\inf_{x \in \mathbb{R}^n} \{c^T x : B(x) \preceq 0\},$$

where $B(x) = B_0 + \sum_{i=1}^{m} x_i B_i$, $c \in \mathbb{R}^m$ and $B_i \in \mathbb{R}^{n \times n}$. General duality results for semi-definite programs are weaker than for linear programs and no straight-forward simplex methods exist for solving them. Generally, they are solved using path-following interior point methods, which instead optimize a convergent sequence of smooth approximations to the original problem. Under this framework, a smoothing convex *barrier function* H is introduced leading to the approximate primal problem

$$\inf_{x \in \mathbb{R}^n} \{c^T x + r^{-1} H(rB(x))\}$$

which converges to the original as $r \to \infty$. The dual of the approximate problem is given as

$$\sup_{Z \succeq 0} \{\mathrm{tr}(B_0 Z) - rH^*(Z) : -\mathrm{tr}(B_i Z) = c_i\}.$$

As can be seen, the Fenchel conjugate plays an important role in the dual formulation. For much more detail, refer to [5, 2, 7].

2 Algorithmic Approach

2.1 One Dimension

Most of the ideas presented in this section closely follow work by Heinz Bauschke and Martin von Mohrenschildt in [3, 4].

2.1.1 A Good Class of Functions

Computer algebra systems are naturally suited to working with functions defined over the real numbers that are finite in representation. It is useful to characterize what is meant by having a finite representation, and to formalize the space of admissible functions.

Let $\mathcal{F}$ be the class of all functions f satisfying the following conditions:

(i) f is a function from $\mathbb{R}$ to $\overline{\mathbb{R}}$;
(ii) f is a closed convex function;
(iii) f is continuous on its effective domain; and,
(iv) there are finitely many points x_i such that

$$x_0 = -\infty < x_1 < \cdots < x_n < x_{n+1} = \infty$$

and each $f|_i$ (f restricted to the open interval (x_i, x_{i+1})) is one of the following:

(a) identically equal to ∞; or,
(b) differentiable.

Since we restrict ourselves to functions on the real line, condition (iii) is equivalent to the lower semi-continuity of f. Additionally, if $f_1, f_2 \in \mathcal{F}$ and $\alpha_1, \alpha_2 \geq 0$, then $\alpha_1 f_1 + \alpha_2 f_2$ and f^* are in $\mathcal{F}$. The class $\mathcal{F}$ is thus well-suited to our purpose.

2.1.2 Subdifferentiation

A convex lower semi-continuous function function on the real line is very well behaved; for instance, it is subdifferentiable on the interior of its domain. It is a straight-forward application of Theorems 1.1 and 1.2 to calculate the subdifferential. Over each open-interval the subdifferential is calculated as

$$\partial f|_i = \begin{cases} \emptyset, & \text{if } f|_i = \infty, \\ \{f|_i{}'\}, & \text{otherwise.} \end{cases}$$

For each point $x_i \notin \text{dom } f$ the subdifferential is empty, while for each point $x_i \in \text{int dom } f$ it may be calculated as the (possibly singleton) closed interval

$$\partial f(x_i) = \left[\lim_{x \uparrow x_i} f|_{i-1}{}'(x),\ \lim_{x \downarrow x_i} f|_i{}'(x) \right].$$

We are left with calculating the subdifferential over the (zero, one or two) points $x_i \in \text{bd dom } f$ as

$$\partial f(x_i) = \begin{cases} (-\infty, \infty), & \text{if } f|_{i-1} = \infty = f|_i, \\ \left(-\infty, \lim_{x \downarrow x_i} f|_i{}'(x)\right], & \text{if } f|_{i-1} = \infty \neq f|_i, \text{ or} \\ \left[\lim_{x \uparrow x_i} f|_{i-1}{}'(x), \infty\right), & \text{if } f|_{i-1} \neq \infty = f|_i. \end{cases}$$

2.1.3 Fenchel Conjugation

Given the subdifferential of f the conjugate of a point y may be calculated in two steps: firstly, solve $y \in \partial f(x)$ for x – the key step – and let $\bar{x}$ be such a solution (if none can be found then $y \notin \text{dom } f^*$, and $f^*(y) = \infty$. Secondly, use the Fenchel-Young equality (Equation 1) to obtain $f^*(y) = \langle \bar{x}, y \rangle - f(\bar{x})$.

To calculate the conjugate over the whole real line, we simply invert each $f|_i{}'$ and insert it into the Fenchel-Young inequality, which in turn defines the conjugate for all $y \in (\lim_{x \downarrow x_i} f|_i{}'(x), \lim_{x \uparrow x_{i+1}} f|_i{}'(x))$. Similarly, each point x_i with a non-singleton differential $\partial f(x_i) = (a, b)$ yields $f^*(y) = x_i y - f(x_i)$ for $y \in (a, b)$. Continuity then determines the value of the Fenchel conjugate at boundary points, with the function taking the value ∞ outside of its domain.

2.1.4 Inversion

Inverting the component derivatives is the biggest challenge to symbolically computing the Fenchel conjugate. We rely on the *Maple* function `solve`, which by its nature has to deal with branch cuts and hence does not always return a unique closed form inverse.

A typical example of this behaviour is in conjugating $f = x^4/4$. In order to proceed we need to invert $y = f'(x) = x^3$. Since *Maple* implicitly works in the complex plane an inverse calculation yields the three cubic roots of y. But none of these expressions is the real root for *all* real values of y.

Definition 2.1 (Branch point) For our purposes, a *branch point* is a point at which a branch cut of an analytic multi-valued function intersects the real line. □

If *Maple* had internal representations of elementary functions (and their inverses) as functions from $\mathbb{R}$ to $\mathbb{R}$, this problem would largely disappear. However, we are left with having to explicitly find the branch points and determine which branch of the inverse is applicable over which sub-interval. Conveniently, classical complex analysis tells us exactly where these branch points may be located when the function we are inverting is analytic.

Theorem 2.2 ([1], Chapter 3, Theorem 11) *Suppose that $f(z)$ is analytic at z_0, $f(z_0) = w_0$, and that $f(z) - w_0$ has a zero of order n at z_0. If $\epsilon > 0$ is sufficiently small, there exists a corresponding $\delta > 0$ such that for all a with $|a - w_0| < \delta$ the equation $f(z) = a$ has exactly n roots in the disk $|z - z_0| < \epsilon$.*

Corollary 2.3 (Location of branch points) *Suppose that f is as in Theorem 2.2. Suppose furthermore that $f(z)$ is analytic on the entire neighborhood $|z - z_0| < \epsilon$, and let $g_1(a), \ldots, g_n(a)$ represent the n roots of $f(z) = a$ on the neighborhood $|a - w_0| < \delta$. Then $g_1(w_0) = \cdots = g_n(w_0) = z_0$.*

Proof: Due to the n^{th} order zero of $f(z)$ at z_0, it follows that $f(z)$ may be expressed as $f(z) - w_0 = (z - z_0)^n g(z)$, where $g(z) \neq 0$, for all z with $|z - z_0| < \epsilon$. Due to the analyticity of $f(z)$ and the existence of exactly n roots by Theorem 2.2, for any a with $|a - w_0| < \delta$ we can write $f(z) - a = (z - g_1(a)) \cdots (z - g_n(a))h(z)$, for some $h(z) \neq 0$. Since $\lim_{a \to z_0} f(z) - a = f(z) - w_0$, it follows that $\lim_{a \to z_0} (z - g_1(a)) \cdots (z - g_n(a))h(z) = (z - z_0)^n g(z)$, and therefore $(z - g_1(w_0)) \cdots (z - g_n(w_0))h(z) = (z - z_0)^n g(z)$. Suppose $g_i(w_0) \neq z_0$ for some i. Then, since $h(z_0) \neq 0$, it follows that the left hand side of the equation has at most $n - 1$ roots at w_0, a contradiction. Thus, it must be that $g_1(w_0) = \cdots = g_n(w_0) = z_0$. ■

As an immediate result of Corollary 2.3 we can infer that branch points of a function f may only occur at zeroes of the first derivative of f. Assuming we are able to find these zeroes, and there are finitely many of them within our interval of interest, we may then test each of the candidate inverses over each implied sub-interval and determine which is the correct branch. Finding these zeroes is left to *Maple*'s routine `solve`, which may not always succeed.

The original algorithms presented in [3, 4] have no facilities to deal with branch selection. The above extension to these original algorithms greatly enlarges the space of functions over which the algorithm may calculate conjugates.

2.2 Many Dimensions

For functions defined over $\mathbb{R}^n$ the Fenchel conjugate may be rewritten as:

$$\begin{aligned} f^*(\mathbf{y}) &= \sup_{\mathbf{x}} \{\langle \mathbf{x}, \mathbf{y} \rangle - f(\mathbf{x})\} \\ &= \sup_{x_1, \ldots, x_n} \left\{ \sum_{i=1}^{n} x_i y_i - f(\mathbf{x}) \right\} \\ &= \sup_{x_1} \left\{ x_1 y_1 + \sup_{x_2} \left\{ x_2 y_2 + \cdots + \sup_{x_n} \{x_n y_n - f(\mathbf{x})\} \cdots \right\} \right\}. \end{aligned}$$

We introduce the concept of a *partial conjugate*. Consider an n-dimensional function that has had a one-dimensional conjugate calculated with respect to the variable x_i. The notation f^{x_i} then represents this partial conjugate of f with respect to x_i. The above may be rewritten as

$$f^* = (-(\cdots - (f^{x_n} \cdots)^{x_2})^{x_1}.$$

This implies that an n-dimensional conjugate can be seen as an iterated sequence of n one-dimensional conjugation and $n-1$ negation operations.

2.2.1 A Good Class of Functions

The natural space to work in is the recursive extension to $\mathcal{F}$. An n-dimensional function f is in $\mathcal{F}^n$ if:

(i) $f(x_1, \ldots, x_n)$ is a function from $\mathbb{R}^n$ to $\overline{\mathbb{R}}$;
(ii) $f(x_1, \ldots, x_n)$ is a closed convex function;
(iii) $f(x_1, \ldots, x_n)$ is continuous on its effective domain; and,
(iv) there are finitely many points a_i such that $a_0 = -\infty < a_1 < \cdots < a_{m-1} < a_m = \infty$ and f restricted to each open interval (a_i, a_{i+1}) is in $\mathcal{F}^{n-1}$ (where $\mathcal{F}^1 = \mathcal{F}$) with respect to the variables $x_2, \ldots, x_n$.

2.2.2 Fenchel Conjugation

Functions in $\mathcal{F}^n$ have an implicit variable order due to their structure. A function defined over the variable order $x_1, \ldots, x_n$ may only be partially conjugated along the last variable. In order to calculate a partial conjugate with respect to another variable x_j, the function must be rewritten such that x_j is the last variable in its representation. We illustrate with an example in $\mathcal{F}^2$.

Example 2.4 (Product of roots) Consider the two-dimensional function

$$f(x_1, x_2) = \begin{cases} \{\ \infty, & \forall x_2 \ , & x_1 < 0 \\ \begin{cases} \infty, & x_2 < 0 \\ 0, & x_2 = 0 \ , & x_1 = 0 \\ 0, & 0 < x_2 \end{cases} & & \\ \begin{cases} \infty, & x_2 < 0 \\ 0, & x_2 = 0 \ , & 0 < x_1 \\ -\sqrt{x_1 x_2}, & 0 < x_2 \end{cases} & & \end{cases}.$$

Calculating the partial conjugate with respect to the x_2 axis involves calculating two one-dimensional partial conjugates; one along the line $x_1 = 0$ and the other over the half-plane $0 < x_1$. Calculating these conjugates (and negating the

results) yields:

$$f^{x_2}(x_1,y_2) = \begin{cases} \begin{cases} \infty, & \forall y_2 \quad , \\ \end{cases} & x_1 < 0 \\ \begin{cases} 0, & y_2 < 0 \\ 0, & y_2 = 0 \\ \infty, & 0 < y_2 \end{cases} & x_1 = 0 \\ \begin{cases} \frac{x_1}{4y_2}, & y_2 < 0 \\ \infty, & y_2 = 0 \\ \infty, & 0 < y_2 \end{cases} & 0 < x_1 \end{cases}$$

We now wish to calculate the partial conjugate along the x_1 variable in order to complete the two-dimensional conjugation. However, in order to do this, we must first reorder the variables to (y_2, x_1). In this example this is easily done through inspection, resulting in:

$$f^{x_2}(y_2,x_1) = \begin{cases} \begin{cases} \infty, & x_1 < 0 \\ 0, & x_1 = 0 \\ \frac{x_1}{4y_2}, & 0 < x_1 \end{cases} & y_2 < 0 \\ \begin{cases} \infty, & \forall x_1 \quad , \\ \end{cases} & y_2 = 0 \\ \begin{cases} \infty, & \forall x_1 \quad , \\ \end{cases} & 0 < y_2 \end{cases}$$

We may now proceed to calculate the complete conjugate by partially conjugating along the x_1 axis. There are two distinct one-dimensional conjugates to be calculated along the line $y_2 = 0$ and the half-plane $y_2 < 0$. This yields:

$$f^{*}(y_2,y_1) = \begin{cases} \begin{cases} 0, & y_1 < \frac{1}{4y_2} \\ 0, & y_1 = \frac{1}{4y_2} \\ \infty, & \frac{1}{4y_2} < y_1 \end{cases} & y_2 < 0 \\ \begin{cases} \infty, & \forall y_1 \quad , \\ \end{cases} & y_2 = 0 \\ \begin{cases} \infty, & \forall y_1 \quad , \\ \end{cases} & 0 < y_2 \end{cases}$$

It is desirable to have the conjugated function in the same variable order as the original function. This involves yet another variable reordering to (y_1, y_2). The result of this operation is the final conjugate:

$$f^{*}(y_1,y_2) = \begin{cases} \begin{cases} 0, & y_2 < \frac{1}{4y_1} \\ 0, & y_2 = \frac{1}{4y_1} \\ \infty, & \frac{1}{4y_1} < y_2 \end{cases} & y_1 < 0 \\ \begin{cases} \infty, & \forall y_2 \quad , \\ \end{cases} & y_1 = 0 \\ \begin{cases} \infty, & \forall y_2 \quad , \\ \end{cases} & 0 < y_1 \end{cases}$$

□

2.2.3 Variable Reordering

A function $f \in \mathcal{F}^n$ can be thought of as a union of functions f_i defined over disjoint sets $S_i \subset \mathbb{R}^n$. Given the variable order $x_1, \ldots, x_n$, each S_i is naturally

represented as $S_i = \{\mathbf{x} : x_1 \in X_1, x_2 \in X_2(x_1), \ldots, x_n \in X_n(x_1, \ldots, x_{n-1})\}$. The operation of changing variable orders to (for example) $x_n, x_1, \ldots, x_{n-1}$ is equivalent to finding $\bar{X}_i$ such that $S_i = \{\mathbf{x} : x_n \in \bar{X}_n, x_1 \in \bar{X}_1(x_n), \ldots, x_{n-1} \in \bar{X}_{n-1}(x_n, x_1, \ldots, x_{n-2})\}$. This is completely equivalent to the problem of changing the order of variables in a multiple-integral (assuming the integrand is sufficiently well behaved). Consider the integral

$$\int_{S_i} f(\mathbf{x}) d\mathbf{x} = \int_{X_1} \cdots \int_{X_n} f(\mathbf{x}) dx_n \cdots dx_1.$$

To change the order of the variables to $x_n, x_1, \ldots, x_{n-1}$ we wish to find $\bar{X}_i$ such that:

$$\int_{S_i} f(\mathbf{x}) d\mathbf{x} = \int_{\bar{X}_n} \int_{\bar{X}_1} \cdots \int_{\bar{X}_{n-1}} f(\mathbf{x}) dx_{n-1} \cdots dx_1 dx_n.$$

In the general case (where none of the dimensions describing S_i are separable) this problem is extremely hard. However, in the non-separable two-dimensional case the problem may be fully broken down to a set of 23 distinct sub-problems (which may be further reduced to 12 due to symmetry), each of which may be solved assuming appropriate zeroes and inverses may be found ([8]). This allows us the associated symbolic Fenchel conjugation algorithm to deal quite robustly with non-separable two-dimensional objects, and higher dimensional ones in certain cases.

3 Examples

3.1 The *Maple* Package `SCAT`

Earlier work by Bauschke and Mohrenschildt [3, 4] focussed on symbolically calculating exact subdifferentials and conjugates for one-dimensional real-valued functions on $\mathbb{R}$, and separable multi-dimensional function on $\mathbb{R}^n$. Their work led to the development of the *Maple* package `fenchel`. The *Maple* package `SCAT` is the result of refining that work and extending it to the non-separable many-dimensional case. It also serves to unite the complementary approach of numerically computing subdifferentials and conjugates, using approaches such as those developed in [10, 11] when symbolic approaches break down.

The *Maple* package `SCAT` (Symbolic Convex Analysis Toolkit) introduces several new constructs and commands to *Maple*: the objects `PWF` and `SD` for representing convex functions and subdifferentials; the function `SCAT[Plot]` for exploring them graphically; the function `SCAT[Eval]` for evaluating them at points, or taking lower dimensional slices; the functions `SCAT[SubDiff]` and `SCAT[Int]` for calculating subdifferentials from convex functions and vice-versa; and the functions `SCAT[Conj]` and `SCAT[InfConv]` for calculating Fenchel conjugates and infimal convolutions. Additionally, the toolkit is well integrated with *Maple*, tying in with *Maple*'s conversion (`convert`), evaluation (`eval`, `evalf`, etc), pretty printing (`print`) and simplification (`simplify`) functionality.

The latest version of this software, along with extensive documentation and usage guides, are available at:

`http://ddrive.cs.dal.ca/projects/scat/`

3.2 Classic Examples

We explore the functionality and capabilities of `SCAT` using several classic examples from the literature.

Example 3.1 (Absolute value) One of the simplest examples of a convex function that is not everywhere differentiable is the absolute value function $f : x \mapsto |x|$. Its derivative at the origin fails to exist since $f'_-(0) = -1 < 1 = f'_+(0)$. The notion of the subgradient is able to capture this behaviour and accordingly it is seen that $\partial f(0) = [-1, 1]$. In order to explore this function we first represent it in a form that `SCAT` understands; the `PWF` (piecewise function) format:

```
> f1 := convert(abs(x),PWF);
```

$$f1 := \begin{cases} -x, & x < 0 \\ 0, & x = 0 \\ x, & x > 0 \end{cases}$$

We may easily calculate the subdifferential of `f1`:

```
> sdf1 := SubDiff(f1);
```

$$sdf1 := \begin{cases} \{-1\}, & x < 0 \\ [-1, 1], & x = 0 \\ \{1\}, & x > 0 \end{cases}$$

We may also calculate the conjugate, yielding:

```
> g1 := Conj(f1,y);
```

$$g1 := \begin{cases} \infty, & y < -1 \\ 0, & y = -1 \\ 0, & (-1 < y) \textbf{ and } (y < 1) \\ 0, & y = 1 \\ \infty, & 1 < y \end{cases}$$

□

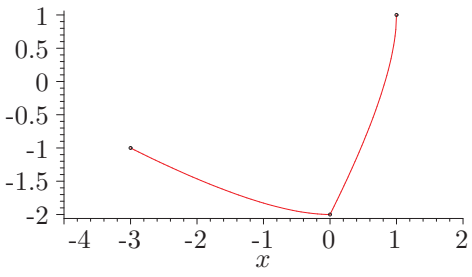

Figure 1: Plot of `f5` from Example 3.2

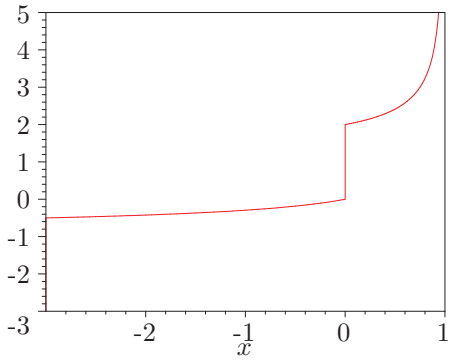

Figure 2: Plot of `sdf5` from Example 3.2

Example 3.2 (An example from Rockafeller) The following function can be found on page 229 of Rockafeller's text [13]. The function is easily constructed using `piecewise` and converted to the `PWF` format:

```
> piecewise(-3<=x and x<=1,abs(x)-2*sqrt(1-x),infinity):
  f5 := convert(%,PWF);
```

$$f5 := \begin{cases} \infty, & x < -3 \\ 1, & x = -3 \\ -2\sqrt{1-x} - x, & (-3 < x) \textbf{ and } (x < 0) \\ -2, & x = 0 \\ -2\sqrt{1-x} + x, & (0 < x) \textbf{ and } (x < 1) \\ 1, & x = 1 \\ \infty, & 1 < x \end{cases}$$

We now use the command `Plot(f5,x=-4..2,scaling=constrained,axes=framed)` to plot the function, yielding Figure 1. Next, to calculate and plot the subdifferential we use the commands `sdf5 := SubDiff(f5)`, and

```
Plot(sdf5,-3..1,view=[-3..1,-3..5],axes=none),
```

yielding

$$sdf5 := \begin{cases} \{\}, & x < -3 \\ [-\infty, -\frac{1}{2}], & x = -3 \\ \{\frac{(-1+\sqrt{1-x})\sqrt{1-x}}{x-1}\}, & (-3 < x) \textbf{ and } (x < 0) \\ [0, 2], & x = 0 \\ \{-\frac{(1+\sqrt{1-x})\sqrt{1-x}}{x-1}\}, & (0 < x) \textbf{ and } (x < 1) \\ \{\}, & x = 1 \\ \{\}, & 1 < x \end{cases}$$

and the plot in Figure 2. Finally, we find the conjugate, the biconjugate and manually verify the convexity of `f5` with the following commands:

```
> g5 := Conj(f5,y);
```

$$g5 := \begin{cases} -3y+1, & y < -\frac{1}{2} \\ \frac{5}{2}, & y = \frac{-1}{2} \\ \frac{y^2+2y+2}{1+y}, & (\frac{-1}{2} < y) \text{ and } (y < 0) \\ 2, & y = 0 \\ 2, & (0 < y) \text{ and } (y < 2) \\ 2, & y = 2 \\ \frac{y^2-2y+2}{-1+y}, & 2 < y \end{cases}$$

```
> F5 := Conj(g5,x):
  Equal(f5,F5);
```

$$true$$

□

Example 3.3 (Young's Inequality) Suppose $1 < p < \infty$ and let q be such that $\frac{1}{p} + \frac{1}{q} = 1$. The equality

$$\frac{1}{p}a^p + \frac{1}{q}b^q \geq ab, \quad \forall a, b \geq 0,$$

is known as *Young's Inequality*. As we are about to see, since $(\frac{1}{p}|\cdot|^p)^* = \frac{1}{q}|\cdot|^q$ this is actually a special case of the stronger Fenchel-Young inequality from Equation 1. In this example we show and confirm the above conjugate pair.

This example elaborates on a similar example provided in [4]. The algorithms developed in this work are able to handle p as a free parameter while those in [4] were restricted to fixed values of p. The general pair of conjugate functions is easily derived using the following commands:

```
> f7 := convert(abs(x)^p/p,PWF,x,{p>1});
  g7 := Conj(f7,y):
  g7 := Subs(p=1/(1-1/q),g7);
```

$$f7 := \begin{cases} \frac{(-x)^p}{p}, & x < 0 \\ 0, & x = 0 \\ \frac{x^p}{p}, & 0 < x \end{cases}$$

$$g7 := \begin{cases} \frac{(-\frac{1}{y})^{(-q)}}{q}, & y < 0 \\ 0, & y = 0 \\ \frac{y^q}{q}, & 0 < y \end{cases}$$

In creating `f7`, notice that we passed additional parameters consisting of a set of assumptions. In this example, if we do not provide the information that $p > 1$ then the process will fail, producing the following output:

```
> f := convert( abs(x)^p/p, PWF, x );
Error, (in EvalRel) unable to evaluate relation:  1/p*limit(x^p,x =
0,right) = 1/p*limit((-x)^p,x = 0,left)
```

□

3.3 Barrier Functions

We turn now to the problem of calculating conjugates of some common barrier functions in semi-definite optimization. Many such functions may be generated

from appropriate one-dimensional convex functions.

Theorem 3.4 (Barrier Function Construction) *Let $\theta : \mathbb{R} \to \bar{\mathbb{R}}$ be an lsc, proper convex and non-decreasing function with dom $\theta = (-\infty, b)$, $b \in [0, \infty)$, $\lim_{x \to -\infty} \theta(x) = 0$ and $\lim_{x \to b} \theta(x) = \infty$. Let $\lambda(D)$ be the vector of eigenvalues of the symmetric matrix $D \in S_n$ in non-decreasing order. Then*

$$H(D) = \sum_{i=1}^{n} \theta(\lambda_i(D))$$

is a smooth barrier function such that $\lim_{r \to \infty} r^{-1} H(rD) = \delta_{S_n^-}(D)$, for all $D \in S_n$.

Proof: Refer to Propositions 2.6.1, 2.8.1 and Theorem 2.7.2 from [2]. ■

A few interesting choices for θ include the following functions taken from page 74 of [2]:

$$\begin{array}{ll}
\theta_1 = \exp(u), & \text{dom } \theta_1 = \mathbb{R}; \\
\theta_2 = -\log(1-u), & \text{dom } \theta_2 = (-\infty, 1); \\
\theta_3 = \frac{u}{1-u}, & \text{dom } \theta_3 = (-\infty, 1); \\
\theta_4 = -\log(-u), & \text{dom } \theta_4 = (-\infty, 0); \text{ and} \\
\theta_5 = -u^{-1}, & \text{dom } \theta_5 = (-\infty, 0).
\end{array}$$

These lead directly to the following barrier functions:

$$\begin{array}{ll}
H_1(D) = \text{tr}(\exp(D)); & \\
H_2(D) = -\log(\det(I - D)), & \text{for } D \prec I, \ \infty \text{ otherwise ;} \\
H_3(D) = \text{tr}((I - D)^{-1} D), & \text{for } D \prec I, \ \infty \text{ otherwise ;} \\
H_4(D) = -\log(\det(-D)), & \text{for } D \prec 0, \ \infty \text{ otherwise ; and} \\
H_5(D) = \text{tr}(-D^{-1}), & \text{for } D \prec 0, \ \infty \text{ otherwise .}
\end{array}$$

In all cases, we are able to symbolically calculate the conjugates of the underlying functions θ_i, yielding:

$$\begin{array}{ll}
\theta_1^* = v(\log(v) - 1), & \text{dom } \theta_1^* = (0, \infty); \\
\theta_2^* = v - \log(v) - 1, & \text{dom } \theta_2^* = (0, \infty); \\
\theta_3^* = (\sqrt{v} - 1)^2, & \text{dom } \theta_3^* = [0, \infty); \\
\theta_4^* = -\log(v) - 1, & \text{dom } \theta_4^* = (0, \infty); \text{ and} \\
\theta_5^* = -2\sqrt{v}, & \text{dom } \theta_5^* = [0, \infty).
\end{array}$$

Example 3.5 (Barrier Function H_1) Considering the two-dimensional case and letting $D = \begin{bmatrix} d_1 & d_2 \\ d_2 & d_3 \end{bmatrix}$ we find that H_1 reduces to the separable two-dimensional function

$$H_1 = \exp d_1 + \exp d_3,$$

for all $d_1, d_3 \in \mathbb{R}$. The conjugate of this is calculated to be

$$H_1^*(Z) = z_1(\log(z_1) - 1) + z_3(\log(z_3) - 1),$$

for $Z = \begin{bmatrix} z_1 & z_2 \\ z_2 & z_3 \end{bmatrix} \succeq 0$. □

Although not powerful enough to calculate conjugates of all the barrier functions H_i for arbitrary dimensions, they do let us explore them in certain lower dimensional cases. We consider a restricted two-dimensional space of symmetric matrices (a slice across the three-dimensional space of matrices S_2)

$$\mathcal{S} = \left\{ \begin{bmatrix} s_1 & s_2 \\ s_2 & s_1 \end{bmatrix} \ : \ s_1, s_2 \in \mathbb{R} \right\}.$$

Example 3.6 (Barrier Function H_2) The function H_2 is defined for $D \prec I$. Further restricting ourselves to $\mathcal{S}$ this leads to the domain being the set of matrices $\mathcal{D} = \{D \ : \ d_1 < 1, \ d_1 - 1 \leq d_2 \leq 1 - d_1\}$. Thus, we may define H_2 as

$$H_2 = -\log(1 - 2d_1 + d_1^2 - d_2^2), \text{ for } D \in \mathcal{D}.$$

The conjugate is calculated by `SCAT` as

$$H_2^* := \begin{cases} \{\infty, & \text{all}(z_2), & z_1 < 0 \\ \{\infty, & \text{all}(z_2), & z_1 = 0 \\ \begin{cases} \infty, & z_2 < -z_1 \\ \infty, & z_2 = -z_1 \\ z_1 - \log(z_1 - z_2) - \log(z_1 + z_2) + 2\log(2) - 2, & (-z_1 < z_2) \text{ and } (z_2 < 0) \\ z_1 - 2\log(z_1) + 2\log(2) - 2, & z_2 = 0 \\ z_1 - \log(z_1 - z_2) - \log(z_1 + z_2) + 2\log(2) - 2, & (0 < z_2) \text{ and } (z_2 < z_1) \\ \infty, & z_2 = z_1 \\ \infty, & z_1 < z_2 \end{cases}, & 0 < z_1 \end{cases}$$

With a little massaging we can simplify this to

$$H_2^*(Z) = z_1 - \log(z_1 - z_2) - \log(z_1 + z_2) + 2\log(2) - 2,$$

for $Z \in \mathcal{S}$ and $Z \succeq 0$. □

3.4 NMR Imaging

The method of maximum entropy reconstruction has been applied in nuclear magnetic resonance spectroscopy (NMR) to the problem of estimating complex spectra. It has proven to be a useful approach as it has the ability to estimate high-resolution spectra from short data records, to deconvolve spectra without enhancing noise and to estimate spectra from non-uniformly sampled time series [6, 9]. By maximizing entropy a spectral estimate is found with the least amount of 'false information', thus, such an optimized spectra is the 'most informative' of all possible estimates.

The choice of the most appropriate information entropy to use has been considered [9]. The Hoch and Stern information measure for complex spectra has been shown to be consistent with the underlying statistical mechanical entropy governing the physical system. In addition to this, it was shown by Borwein et al in [6] that this entropy has a very natural dual representation, and the corresponding dual optimization problem is more efficient to solve with greater numerical accuracy.

Let $D = [-\Delta/2, \Delta/2]$ be the support of the unknown spectrum, modelled by a function $\phi(s) \in L^2_{\mathbb{C}}(D)$, and let $\psi(t) \in L^2_{\mathbb{C}}(D)$ represent the time signal. The problem is that of recovering ϕ from a noisy discrete measurement of ψ,

$Y = \psi(\mathbb{J}\delta t) \in \mathbb{C}^m$, where $\mathbb{J} = \{0, \ldots, m-1\}$ and δt represents the sampling interval. The Hoch and Stern entropy is defined by $H(X) = \sum_{j=1}^{n} h(X_j)$, where h is the convex function on $\mathbb{C}$ given by

$$h(z) = f\left(\frac{|z|}{b}\right) \text{ and } f(u) = u \log\left(u + \sqrt{1+u^2}\right) - \sqrt{1+u^2},$$

with $f(|z|)$ plotted in Figure 3. With these definitions one version of the primal problem may be stated as

$$(P_{NMR}) \qquad \inf\left\{H(X) \ : \ X \in \mathbb{C}^n, \ \|Y - AX\| \leq \varepsilon\right\},$$

with the matrix A representing the discrete Fourier transform operator. In [6] the dual of this problem is derived as

$$(D_{NMR}) \qquad \sup\left\{\mathrm{Re}[\langle Y, \Lambda\rangle] - \varepsilon\|\Lambda\| - H^*(A^*\Lambda) \ : \ \Lambda \in \mathbb{C}^m\right\}.$$

The motivation to pursue the dual problem came from the casual observation that $f^*(u^*) = \cosh u^*$ (plotted in Figure 4), which was calculated by Borwein using `SCAT`'s direct ancestor, `fenchel`. This leads immediately to $H^*(X^*) = \sum_{j=1}^{n} \cosh(b|X_j^*|)$. Given a solution $\bar{\Lambda}$ to the dual problem, a solution to the primal is found as

$$\bar{X} = b \exp\left[i \arg A^*\bar{\Lambda}\right] \sinh(b|A^*\bar{\Lambda}|),$$

and a solution to the original spectra as

$$\bar{\phi}(s) = b \exp\left[i \arg\left([A^*\bar{\Lambda}](s)\right)\right] \sinh(b|[A^*\bar{\Lambda}](s)|).$$

As can be seen the dual problem is an even hence smooth unconstrained maximization. Furthermore, the characterization of the dual of the Hoch and Stern entropy showed that it is directly related to the dual of the classical Shannon entropy, being the even part. These insights have led to the development of more efficient and numerically stable algorithms for reconstructing NMR images, and were partially facilitated by the ease and low-cost of thought experiments using tools like `SCAT`.

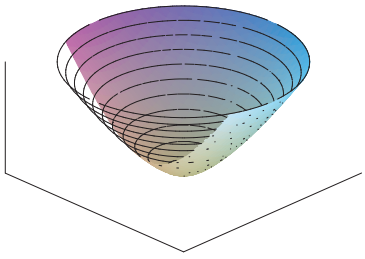

Figure 3: Plot of $f(|z|)$ from §3.4

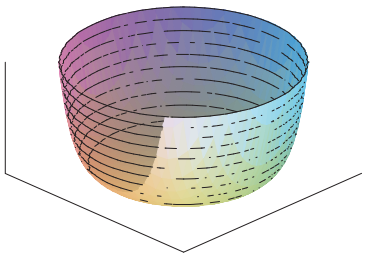

Figure 4: Plot of $f^*(|z^*|)$ from §3.4

4 Limitations and Future Work

In one dimension, the biggest challenge to symbolically computing Fenchel conjugates is in inverting the subdifferential. Although improved compared to earlier versions of the algorithm, we rely on the *Maple* function `solve` to identify branch points and applicable inverses for each branch. This often prevents us from computing a closed form even when such a solution exists. As *Maple* (or *Mathematica*, in which one could certainly also implement these ideas) makes improvements in the underlying machinery, the space of functions on which `SCAT` can successfully compute conjugates will only grow.

In two-dimensions the same difficulty in computing inverses often prevents us from changing the variable order of the partial conjugate, preventing the completion of the conjugate operation. In higher dimensions this problem is even harder, and we currently have no mechanism to even approach it.

Finally, we are limited by the internal representation of functions. Functions must be input in rectangular coordinates, making it very awkward to manipulate some otherwise very simple and natural functions (for example, the indicator function of the unit ball in $\mathbb{R}^n$).

5 Concluding Remarks

In the *Maple* package `SCAT` we have implemented algorithms for conjugation, subdifferentiation and infimal convolution of convex functions defined on $\mathbb{R}$. We have extended previous algorithms to allow conjugation of non-separable functions defined on $\mathbb{R}^2$, and in certain cases for higher dimensions. We have provided examples and applications and commented on the limitations of the algorithms. One especially useful application is that practitioners can produce automated code for symbolic components of larger computational projects. It is our hope that `SCAT` will be useful to researchers, instructors and students of convex analysis and optimization.

References

[1] L. Ahlfors. *Complex Analysis.* McGraw-Hill, New York, 1966.

[2] A. Auslender and M. Teboulle. *Asymptotic Cones and Functions in Optimization and Variational Inequalities.* Springer-Verlag, 2003.

[3] H.H. Bauschke and M. v. Mohrenschildt. Fenchel conjugates and subdifferentiation in Maple. Technical Report CORR 97-23, Department of Combinatorics and Optimization, University of Waterloo, 1997.

[4] H.H. Bauschke and M. v. Mohrenschildt. Symbolic computation of Fenchel conjugates. To appear in ACM SIGSAM bulletin, 2005.

[5] J.M. Borwein and A.S. Lewis. *Convex Analysis and Nonlinear Optimization.* CMS Books in Mathematics. Springer-Verlag, New York, 2000.

[6] J.M. Borwein, P. Maréchal, and D. Naugler. A convex dual approach to the computation of NMR complex spectra, 1998.

[7] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, Cambridge, UK, 2004.

[8] C.H. Hamilton. Symbolic convex analysis. Master's thesis, Department of Mathematics and Statistics, Simon Fraser University, April 2005.

[9] J.C. Hoch, A.S. Stern, D.L. Donoho, and I.M Johnstone. Maximum entropy reconstruction of complex (phase-sensitive) spectra. *Journal of Magnetic Resonance*, 86:236–246, 1990.

[10] Y. Lucet. A fast computational algorithm for the Legendre-Fenchel transform. *Computational Optimization and Applications*, 6(1):27–57, 1996.

[11] Y. Lucet. Faster than the fast Legendre transform, the linear-time Legendre transform. *Numerical Algorithms*, 16:171–185, 1997.

[12] D. Luenberger. *Optimization by Vector Space Methods.* Series in Decision and Control. Wiley, New York, 1969.

[13] R.T. Rockafeller. *Convex Analysis.* Princeton University Press, Princeton, NJ, 1970.