THE PIECEWISE LINEAR-QUADRATIC MODEL FOR COMPUTATIONAL CONVEX ANALYSIS

YVES LUCET, HEINZ H. BAUSCHKE, AND MIKE TRIENIS

Abstract. A new computational framework for computer-aided convex analysis is proposed and investigated. Existing computational frameworks are reviewed and their limitations pointed out. The class of piecewise linear-quadratic functions is introduced to improve convergence and stability. A stable convex calculus is achieved using symbolic-numeric algorithms to compute all fundamental transforms of convex analysis. Our main result states the existence of efficient (linear time) algorithms for the class of piecewise linear-quadratic functions. We also recall that such class is closed under convex transforms. We illustrates the results with numerical examples, and validate numerically the resulting computational framework.

1. Introduction

Computational convex analysis focuses on developing efficient tools to compute fundamental transforms arising in convex analysis. Symbolic computation tools have been developed [4, 5, 14], and have allowed more insight into the calculation of the Legendre-Fenchel conjugate (also called Fenchel conjugate, convex conjugate, or —in the context of convex analysis— conjugate) and related transforms. When such tools are not applicable *e.g.* when there is no closed form, fast transform algorithms perform numerical computation efficiently.

Although the early idea of efficient numerical computation of convex transforms can be traced back to [29], the development of efficient algorithms started with the note [6] on the Fast Legendre Transform (FLT), which was subsequently investigated in [7, 23]. The FLT algorithm was also independently presented in [32, 36]. Its log-linear worst-case time complexity was

Date: Reviewed March 1, 2007, Version 1.2; Submitted July 17, 2006, Version 1.1.
2000 *Mathematics Subject Classification.* 52B55,52A41,65Y20.
Key words and phrases. Computational Convex Analysis, Proximal Average, Legendre-Fenchel Transform, Fenchel conjugate, Moreau Envelope, Moreau-Yosida Approximate, Convex Analysis.
Yves Lucet was partially supported by the Natural Sciences and Engineering Research Council of Canada. Heinz Bauschke was partially supported by the Natural Sciences and Engineering Research Council of Canada and the Canada Research Chair Program. Mike Trienis was partially supported by the Pacific Institute for the Mathematical Sciences, and UBC Okanagan Internal Grant program.

subsequently improved with the Linear-time Legendre transform (LLT) algorithm [24] (see also [25] for the description of a numerical implementation of the LLT algorithm).

Recently, several new linear fast algorithms have been investigated [18, 27]. They take advantage of the equivalence between the computation of the conjugate and of the Moreau envelope [35, Example 11.26c] also called the Moreau–Yosida approximate, Yosida Approximate or Moreau–Yosida regularization. The Moreau envelope goes back to the work of Yosida [39] on maximal monotone operators, and its behaviour is well known in the fields of convex analysis [28, 29, 30, 33] and variational analysis [35, Chapter 12]. Its associated transform, the proximal mapping, has been extensively studied through the analysis of the proximal point algorithm [34], whose convergence properties are well known [1, 13, 22].

The fast algorithms are not limited to computing the conjugate or the Moreau envelope. For example, Moreau [29] already noted that the set of proximal mappings is convex, although an explicit formula for the convex function whose proximal mapping is the convex combination of two proximal mappings was only recently presented [3]. The resulting transform, named the proximal average in [2], combines several fundamental operations of convex analysis: addition, scalar multiplication, conjugation, and regularization with the Moreau envelope. Other transforms related to the Moreau envelope include the Lasry-Lions double envelope, the proximal hull, the inf-convolution of convex functions, and the deconvolution of convex functions [17, 35]. They are all computable by combining fast algorithms.

Beyond computer-aided convex analysis, the FLT algorithm and the faster LLT algorithm have been used in efficient numerical simulations of the Burger's equation, see for example [11, 31]. The LLT has also found applications in robotics [20], network communication [19], pattern recognition [26], numerical simulation of multiphasic flows [15], and analysis of the distribution of chemical compounds in the atmosphere [21]. The Moreau envelope is an extension of the distance transform encountered in image processing, and several authors have investigated fast algorithms in that context [8, 9, 10, 12, 37] (see also [26] for the explicit application of the LLT algorithm to the computation of the distance transform). The inf-convolution and addition of convex functions are also related to the Linear Cost Network Flow problem on Series-Parallel Networks [38].

We summarize the intrinsic framework of fast algorithms, and point out their limitations in Section 2. We then consider the parametric framework introduced in [18], which implicitly model the domain of the conjugate in Section 3. The underlying models are explicited in Section 4, and the class of piecewise linear-quadratic (PLQ) functions is introduced in Section 5 to remedy shortcomings in the implicit and parametric frameworks. In contrast to these frameworks, the PLQ functions are closed under standard convex operations, and can be manipulated with linear time algorithms. We present numerical examples in Section 6 while Section 7 introduces our numerical package, and validates the algorithms numerically. We conclude the paper in Section 8.

Unless otherwise stated, we restrict our framework to lower semicontinuous (lsc) proper extended-valued convex functions on the real line. (Future extensions to functions of several variables will be considered in Section 8.) Note that all algorithms for univariate functions extend straightforwardly to separable functions. So checking that $(\|\cdot\|^2/2)^* = \|\cdot\|^2/2$ can be performed in our framework. We denote I_S the indicator function of a set S: $I_S(x) = 0$ when $x \in S$, and $I_S(x) = +\infty$ otherwise.

2. Discrete convex Transform

In this section, we recall the computational framework for fast algorithms and point out their limitations for computing composition of convex transforms. We recall that the domain of a lower semicontinuous (lsc) proper extended-valued convex function $f : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ is defined by $\operatorname{Dom} f := \{x \in \mathbb{R} | f(x) < +\infty\}$. (A function is proper if its domain is nonempty.)

Computing fundamental transforms of convex analysis is reduced to a discrete optimization problem. For example, the computation of the conjugate or Legendre-Fenchel transform

$$f^*(s) := \sup_{x\in\mathbb{R}}[sx - f(x)] \tag{1}$$

is approximated with the discrete Legendre transform

$$f_n^*(s_j) := \max_i [s_j x_i - f(x_i)] \tag{2}$$

where the function f is only evaluated at the points x_i, $i = 1 \dots n$, and the conjugate is approximated at the slopes s_j, $j = 1 \dots m$. The key point we are interested in is *evaluating the transform on a grid*, and not at a single point. By evaluating the transform on the full grid, we can take advantage of evaluations at other points to speed up computation.

We focus on the efficient computation of (2) in terms of worst-case time complexity. While a brute force computation has an $O(nm)$ complexity (evaluate a maximum over n points for m slopes), an efficient (linear-time) algorithm like the LLT algorithm [24] evaluates (2) in $O(n + m)$.

Convergence results [7, 23] allow us to approximate (1) from (2) as follow. Indeed, when the function or its transform have an unbounded domain, we enlarge the grid to obtain a more accurate numerical approximation of its domain *i.e.* we consider intervals $[a, b]$ with $a \to -\infty$ and $b \to +\infty$. Then we fix a and b, and increase the number of points in the grid by decreasing the grid stepsize.

For the sake of completeness we recall the main convergence facts.

Fact 2.1. *Assume* $f : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ *is proper.*

Convergence on a bounded domain: *(see* [7, 23, 24]*).*

(i) *If f is upper semi-continuous on $[a, b]$, then f_n^* converges pointwise to $f^*_{[a,b]}(s) := \sup_{x\in[a,b]}[sx - f(x)]$.*

(ii) *If f is continuously twice differentiable on an open interval containing $[a, b]$, then*

$$\max_{[a,b]} |f^*_{[a,b]} - f_n^*| \le \frac{1}{2}\frac{(b-a)^2}{n^2} \max_{[a,b]} f''.$$

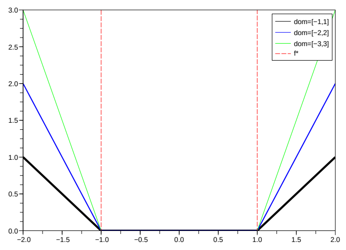
Convergence on unbounded domains: *(see* [16, 24]*)*

The following equivalence holds for any $s \in \mathbb{R}$, and any $a > 0$

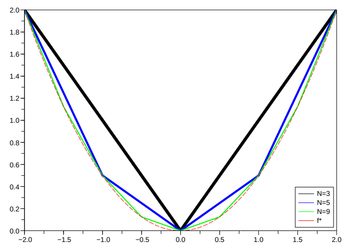
$$\partial f^*(s) \cap [-a, a] \ne \emptyset \Leftrightarrow f^*_{[-a,a]}(s) = f^*(s),$$

where ∂f is the subdifferential in the sense of convex analysis

$$\partial f(x) := \{s \in \mathbb{R} \;:\; f(y) \ge f(x) + s(y - x) \textit{ for all } y\}.$$



(a) Convergence by enlarging the domain: the conjugate of $f(x) := |x|$ converges to $I_{[-1,1]}(x)$ the indicator function of the interval $[-1, 1]$



(b) Convergence by decreasing the grid stepsize: the discrete conjugate of the function $f(x) := x^2/2$ converges to $f^* = f$.

FIGURE 1. Convergence of the discrete Legendre transform.

Figure 1(a) illustrates an approximation of the domain: increasing the size of the grid for x_i gives a better approximation of the conjugate (the slopes converge to the two vertical lines), which is the indicator function of the interval $[-1, 1]$. Figure 1(b) shows an approximation of the function within the domain: reducing the grid stepsize gives a better approximation (the approximation is the piecewise linear function; it converges to the exact conjugate, which is the quadratic function).

We will see in Section 5 how the approximation of the domain can be avoided altogether by using a more general class of functions than the piecewise linear functions. The convergence on the domain will also be improved by using piecewise quadratic functions instead of piecewise linear functions.

Similarly, the computation of the Moreau envelope

$$M_\lambda(s) := \inf_{x\in\mathbb{R}} \left[\frac{(s-x)^2}{2\lambda} + f(x)\right]$$

for all values of s in an interval $[a, b]$ ($\lambda > 0$ is a fixed parameter) is approximated with the discrete Moreau envelope

$$M_{n,\lambda}(s_j) := \min_i \left[\frac{(s_j - x_i)^2}{2\lambda} + f(x_i) \right]$$

where $i = 1 \dots n$, and $j = 1 \dots m$. Note that the Moreau envelope is a special case of the inf-convolution operator

$$f \Box g(x) := \inf_y [f(x - y) + g(y)].$$

Throughout the paper we will use the following facts.

Fact 2.2. [17, X.2.1.3]
Assume f and g are both lsc convex proper functions and $\operatorname{Dom} f^* \cap \operatorname{Dom} g^*$ *is nonempty. Then* $(f \Box g)^* = (f^* + g^*)^*$

Fact 2.3. [35, 11.26c] *and* [27, Proposition 3]
The following formula holds for any function f

$$M_\lambda(s) = \frac{s^2}{2\lambda} - \frac{1}{\lambda} \left(\lambda f + \frac{|\cdot|^2}{2} \right)^* (s). \tag{3}$$

Formula 3 implies that as far as computational algorithms are concerned, computing the conjugate is equivalent to computing the Moreau envelope. So any efficient (linear-time) algorithm to compute one transform can be used to compute the other.

The framework is the same for all the fast transform algorithms previously considered [27]: the computation is restricted to a grid of points, and convergence results are employed to obtain a numerical approximation of the transform under consideration.

Unfortunately, the fast algorithm approach becomes awkward when we consider transforms like the proximal average

$$\mathcal{P}(f_0, \lambda, f_1) := \left[(1 - \lambda) \left(f_0 + \frac{1}{2}|\cdot|^2 \right)^* + \lambda \left(f_1 + \frac{1}{2}|\cdot|^2 \right)^* \right]^* - \frac{1}{2}|\cdot|^2.$$

While it decomposes in several transforms, functions resulting from intermediate computations have unbounded domains. In other words, even when the input function requires only few grid points and we only want a coarse approximation of the transform, we may have to compute on large grids with a prohibitive number of points during intermediate computations.

Consider the following example.

Example 2.4. *Let $f_0(x) := -x$, $f_1(x) := x$, and compute their proximal average $f := \mathcal{P}(f_0, 1/2, f_1)$. The conjugate of f_0 (resp. f_1) is $f_0^*(s) = I_{\{-1\}}(s)$ (resp. $f_1^*(s) = I_{\{1\}}(s)$). The Moreau envelope of f_0^* (resp. f_1^*) is $f_0^* \Box 1/2|\cdot|^2 = (x+1)^2/2$ (resp. $f_1^* \Box 1/2|\cdot|^2 = (x-1)^2/2$). Since the domain of both Moreau envelopes is $\mathbb{R}$, computing their average requires modeling the domain: taking a large number of points to numerically approximate the* *domain. Failing to do that results in a poor numerical approximation of the sum, which propagates to a poor approximation of the conjugate, and eventually to a poor approximation of the proximal average.*

In fact, if we only consider the convergence required due to unbounded domains and ignore the convergence required by decreasing the grid stepsize, the combination of fast discrete algorithms to compute the proximal average — add the energy $x^2/2$, apply the LLT, build the convex combination, apply the LLT, and subtract the energy — results in the following function

$$\begin{cases} +\infty & \text{if } |x| > b, \\ (2\lambda-1)x - 2\lambda(1-\lambda) & \text{if } 2(1-\lambda) - b \le x \le b - 2\lambda, \\ \frac{\lambda}{2(1-\lambda)}x^2 + \frac{\lambda+\lambda b-1}{1-\lambda}x + \frac{\lambda b(4\lambda+b-4)}{2(1-\lambda)} & \text{if } -b \le x \le 2(1-\lambda) - b, \\ \frac{1-\lambda}{2\lambda}x^2 + \frac{\lambda-b+\lambda b}{\lambda}x + \frac{b(4\lambda^2+b-\lambda b-4\lambda)}{2\lambda} & \text{if } b - 2\lambda \le x \le b \end{cases}$$

where f_0 (resp. f_1) is approximated with $-x+I_{[-b,b]}(x)$ (resp. $x+I_{[-b,b]}(x)$), and $b > 1$.

Even though the functions f_0 and f_1 are linear, the fast algorithm framework requires an (unnecessary) technical knowledge: in addition to guessing the domain on which to calculate the convex combination for each of the intermediate operations, we now need to increase b to $+\infty$ to guarantee accurate approximations. When b is chosen large enough, we finally have to decrease the grid stepsize to achieve our desired approximation.

To use the fast numerical algorithms to compute the proximal average, we have to keep track of four sets: The domain of f_0, the domain of f_1, the set that approximates the domain of $(1-\lambda)(f_0+|\cdot|^2/2)^* + \lambda(f_1+|\cdot|^2/2)^*$, and the domain on which we want to compute $\mathcal{P}(f_0,\lambda,f_1)$. Section 5 presents a new model that does not require such tracking. It thus greatly simplify the computational framework, and the technical knowledge to compute compositions of convex operator such as the proximal average.

The next section considers another existing computational framework which has a better modeling of the domain.

3. Parametric convex transforms

We summarize the recent introduction of a parametric algorithm [18] to compute the conjugate (which using (3) can also be used to compute the Moreau envelope), generalize the approach to a full convex calculus, and point out the limitations of such a framework.

Recently, the numerical computation of the conjugate was investigated using the parametrization

$$\begin{cases} s \in \partial f(x), \\ f^*(s) = sx - f(x); \end{cases}$$

for $x \in \mathbb{R}$. The term parametrization comes from the description of the conjugate: instead of returning a function f^*, the algorithm returns a parametric description of the planar curve $(s, f^*(s))$. Note that we no longer have access to $f^*(s)$ at any slope s, we only obtain f^* on the range of ∂f. The idea relies on the geometric characterization of the epigraph of f^* whose extreme points are obtained using the parametrization, then the epigraph is completed by affine parts to recover the full graph of f^*. Consider the following example.

Example 3.1. *Take $f(x) := |x|$. Then $\partial f(x) = \{-1\}$ when $x < 0$, $\partial f(x) = [-1, 1]$ when $x = 0$, and $\partial f(x) = \{1\}$ when $x > 0$. So the parametrization above gives*

- *for $x < 0$, $s = -1$ and $f^*(s) = sx - f(x) = -x - |x| = 0$.*
- *for $x = 0$, $s \in [-1, 1]$ and $f^*(s) = 0$.*
- *for $x > 0$, $s = 1$ and $f^*(s) = x - |x| - 0$.*

Consequently, $f^*(s) = 0$ for $s \in [-1, 1]$. For any s not in the range of ∂f, which is $[-1, 1]$, $f^*(s) = +\infty$. We obtain the right answer: $f^* = I_{[-1,1]}$.

While the framework is different from Section 2, the fast algorithms can be formalized in the parametric framework as follow. Their computation of the conjugate amounts to computing the function

$$(x_i, f_i, s_j) \mapsto (s_j, f_j^*),$$

where x_i is a primal grid point, f_i is an approximation of $f(x_i)$, s_j is a dual grid point, and f_j^* is an approximation of $f^*(s_j)$. Note that the algorithms really return f_j^* since s_j is part of the input, but we write the pair (s_j, f_j^*) to emphasize the comparison with the parametric framework of the current section. The points (s_j, f_j^*) are an approximation of the graph of f^* in the plane.

To simplify our presentation, we assume the function f is twice differentiable for the remainder of this section only. Note that contrary to second order results which do not hold when f is not twice differenciable, the first order results still hold by replacing the derivative f' with the subgradient ∂f when f is not differentiable.

In the parametric framework, given a discretization $f_i \approx f(x_i)$ of a function f and of its derivatives $g_i \approx f'(x_i)$, $h_i \approx f''(x_i)$ on a grid x_i, we define the discretization of the conjugate and its derivatives by

$$\mathrm{Conj} : (x_i, f_i, g_i, h_i) \mapsto (g_i, x_i g_i - f_i, x_i, \frac{1}{h_i}). \tag{4}$$

Proposition 3.2. *The function* Conj *provides an approximation of the conjugate and of its first two derivatives:*

$$\mathrm{Conj}(x_i, f_i, g_i, h_i) \approx (g_i, f^*(g_i), (f^*)'(g_i), (f^*)''(g_i)).$$

Proof. The proof that Conj is an approximation of the conjugate is contained in [18] except for the second order approximation, which is proven in [17, X.4.2.9] and also in [35, 13.21 and p. 605] (we are using the convention $1/0 := +\infty$). □

Remark 3.3. *Note that Formula* (4) *implies the discrete equivalent of the Biconjugation Theorem, which states that $f^{**} = f$. Indeed, apply* Conj *twice, to obtain*

$$\mathrm{Conj}(\mathrm{Conj}(x_i, f_i, g_i, h_i)) = \mathrm{Conj}(g_i, x_i g_i - f_i, x_i, \frac{1}{h_i}) = (x_i, f_i, g_i, h_i).$$

So we recover our original discrete data set (x_i, f_i, g_i, h_i).

In addition to providing approximations to the derivatives of the conjugate, the advantage of the parametrization is two-fold: the implementation is very simple in matrix-like languages like Matlab, Scilab, or GNU Octave, and the domain of the conjugate is automatically obtained without any a priori knowledge of the original function. In effect, the domain of the conjugate is implicitly modeled using the range of ∂f.

The first price we pay for such advantages is that we only obtain vertices on the convex envelope of the graph of the conjugate, not the full graph. We have to complete the result by linear interpolation. Consider the following example.

Example 3.4. *Consider Example 3.1 with $x_1 := -1$, $x_2 := 0$, and $x_3 := 1$. Then* $\mathrm{Conj}(x_1, f_1, g_1) = \mathrm{Conj}(-1, 1, -1) = (-1, 0, -1)$, $\mathrm{Conj}(x_2, f_2, g_2) = \mathrm{Conj}(0, 0, 0) = (0, 0, 0)$, *and* $\mathrm{Conj}(x_3, f_3, g_3) = \mathrm{Conj}(1, 1, 1) = (1, 1, 1)$. *The values of f^* between the grid values are recovered by linear interpolation. The values outside the grid must be computed by extrapolation (see* [18] *for more details).*

See Example 3.6 below for the equivalent effect on computing the Moreau envelope (in that case quadratic interpolation is needed).

The second price we pay is we require more than just a black box returning the values of the function, we also need an approximation of the derivative of f or its subgradients when such derivative does not exist.

To obtain a similar parametrization for the Moreau envelope, we use Formula (3) to deduce (see [18] for details)

$$\begin{cases} z & \in x + \lambda \partial f(x), \\ M_\lambda(z) & = f(x) + \frac{(z-x)^2}{2\lambda}. \end{cases}$$

The equivalent to the discrete parametrization Conj is then the discrete parametrization Me defined by

$$\mathrm{Me} : (x_i, f_i, g_i, h_i) \mapsto (x_i + \lambda g_i, f_i + \frac{\lambda}{2} g_i^2, g_i, \frac{h_i}{1 + \lambda h_i}), \tag{5}$$

where, as for Conj, the appropriate modifications are made when f is not twice differentiable.

Proposition 3.5. *The function* Me *provides an approximation of the Moreau envelope and, when they exist, of its first two derivatives.*

Proof. The discretization (5) can be obtained either from Equation (4) and Equation (3) or in our present convex framework from Equation (4) and using $M_\lambda = \big(f^* + \frac{\lambda}{2}|\cdot|^2\big)^*$. The result follows using Proposition 3.2. □

The next example from [18] illustrates the fact that while the graph of the conjugate has to be completed by linear interpolation, the graph of the Moreau envelope has to be completed by quadratic interpolation.

Example 3.6. *Consider the function f defined as the convex envelope of $x \mapsto ||x| - 1|$. We have $f(x) = -x - 1$ when $x \le -1$, $f(x) = 0$ when $-1 \le x \le 1$, and $f(x) = x - 1$ when $1 < x$. Its Moreau envelope is*

$$M_\lambda(x) = \begin{cases} -x - 1 - \frac{\lambda}{2} & \text{when } x \le -\lambda - 1, \\ \frac{(x+1)^2}{2\lambda} & \text{when } -\lambda - 1 \le x \le -1, \\ 0 & \text{when } -1 \le x \le 1, \\ \frac{(x-1)^2}{2\lambda} & \text{when } 1 \le x \le \lambda + 1, \\ x - 1 - \frac{\lambda}{2} & \text{when } 1 + \lambda < x. \end{cases}$$

However, the parametrization formula gives

$$\begin{cases} z = x - \lambda \text{ and } M_\lambda(z) = -x - 1 + \frac{\lambda}{2} & \text{when } x < -1, \\ z = x \text{ and } M_\lambda(z) = 0 & \text{when } -1 < x < 1, \\ z = x + \lambda \text{ and } M_\lambda(z) = x - 1 + \frac{\lambda}{2} & \text{when } 1 < x. \end{cases}$$

So we deduce

$$M_\lambda(z) = \begin{cases} -z - \frac{\lambda}{2} - 1 & \text{when } z < -\lambda - 1, \\ 0 & \text{when } -1 < z < 1, \\ z - \frac{\lambda}{2} - 1 & \text{when } \lambda + 1 < z. \end{cases}$$

The two remaining segments when x is in $[-\lambda - 1, -1]$ and $[1, \lambda + 1]$, have to be recovered by quadratic interpolation.

We finish the section by pointing out that in addition to conjugation, two further elementary operations are needed to compute transforms such the proximal average: scalar multiplication and addition.

The multiplication by a positive scalar λ (the result is true for $\lambda \in \mathbb{R}$ although convexity is preserved only when $\lambda \ge 0$) trivially gives the following transform

$$\text{scalar} : (x_i, f_i, g_i, h_i) \mapsto (x_i, \lambda f_i, \lambda g_i, \lambda h_i).$$

So scalar multiplication integrates perfectly in our parametric framework.

Unfortunately, the addition operator is not so easy since adding two discretizations is not straightforward when the functions do not have identical grids. In the extreme case when the functions have no grid point in common, the pointwise addition results in the function that is identically equal to $+\infty$. So the underlying functions have to be modeled between grid points, *e.g.* as piecewise linear functions, to compute their sum. The next section considers several such models

4. The zeroth- and first-order models

In this section, we consider several models to recover a (convex) continuous function from a discrete set of points. We aim to provide an answer to the previous section in term of an addition operator for the parametric framework. In addition, our results motivate the introduction of the class of piecewise linear-quadratic functions in Section 5.

Given a discretization of a function f on a grid x_i and possibly of its derivatives, we say that the function $\check{f}$ is *a model of the function* f if it interpolates the function on the grid *i.e.* $f(x_i) = \check{f}(x_i)$. It is a zeroth-order model if only the value of the function is taken into account, a *first-order model* if both the value of the function and of its first derivative are interpolated *i.e.* $f(x_i) = \check{f}(x_i)$ and $f'(x_i) = \check{f}'(x_i)$, and a second order model if in addition $f''(x_i) = \check{f}''(x_i)$. Again we do not assume f is differentiable. At any point where it is not, substitute ∂f for f', make the appropriate modifications, and discard the statement for f''. We only write f' to simplify the exposition and clarify the main ideas.

A natural zeroth-order model to consider for convex transforms is the piecewise linear model

$$\check{f}_0(x) := \max_i [f_i + s_i(x - x_i)],$$

where s_i corresponds to increasing slopes in $\mathbb{R}$ (*i.e.* $s_i < s_{i+1}$), not necessarily to the derivative of f at x_i. Since the function f is convex, under a reasonable choice of s_i, we have $\check{f}_0(x_i) = f_i = f(x_i)$, and the function $\check{f}_0$ is convex as the maximum of convex functions. Additionally, its conjugate is also piecewise linear [35, 11.14 (a)] with bounded domain. This is the implicit model used by the fast transform algorithms. In particular, the LLT algorithm relies explicitly on the finite difference slopes $s_i := (f_{i+1} - f_i)/(x_{i+1} - x_i)$.

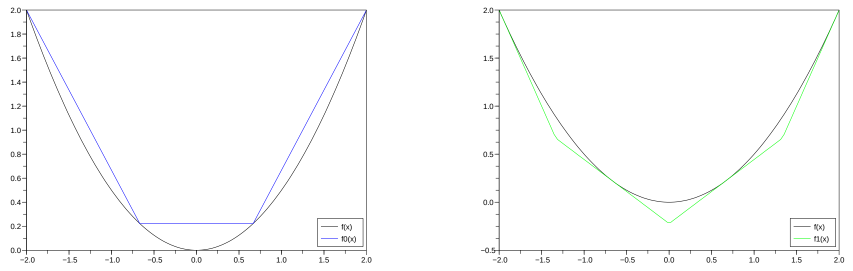
The main drawback of this model is its absence of any modeling of the domain of the function f (see Example 2.4). Even if we only consider piecewise linear functions, their conjugate is piecewise linear but with a bounded domain. So its domain has to be approximated using convergence results. Moreover, the lack of second order term means that any quadratic function is approximated by piecewise linear functions. Unfortunately, the Moreau envelope gives rise to quadratic functions for the simplest convex functions: indicators of a point.

We now turn to the parametric framework of Section 3. The PLT algorithm uses the following first order model

$$\check{f}_1(x) := \max_i [f_i + g_i(x - x_i)]$$

with $g_i := f'(x_i)$ (or $g_i \in \partial f(x_i)$ when f is not differentiable at x_i). We have $\check{f}_1(x_i) = f(x_i)$, and $\check{f}_1'(x_i) = g_i = f'(x_i)$. Hence $\check{f}_1$ is a first order model. Additionally, Formula (4) provides a nice duality for the class of piecewise linear functions.

However, $\check{f}_1$ suffers from the same drawbacks as $\check{f}_0$: the class of piecewise linear functions is not closed under standard convex operators. For example, computing the conjugate of a linear function results in an indicator function whose Moreau envelope is a quadratic function. Consequently, convergence has to be used to obtain a good numerical approximation even for linear, indicator, or quadratic functions.



(a) Zeroth order interpolation as used in the LLT algorithm (b) First order interpolation as used in the PLT algorithm

Figure 2. Interpolation schemes for discrete transforms.

Figure 2(a) illustrates the zeroth order model $\check{f}_0$ to approximate a quadratic function, while Figure 2(b) shows the first order model $\check{f}_1$ to also approximate a quadratic. Taylor approximation justifies the better fitting of the first order model.

Both models can be used to build a discrete addition operator for the parametric framework of the previous section as follow. Consider two functions to be added on two potentially disjoint grids. Consider the union of both grids, and compute any missing value for each function by linear interpolation. Then just add the values at each grid point. While the idea is simple and provides a solution to the parametric framework, it is an ad hoc solution that does not solve the intrinsic issues of a piecewise linear model namely the class of piecewise linear functions is not closed under the standard convex operators.

To remedy this shortcoming we consider second order models in the next section using the class of piecewise linear-quadratic functions.

5. Piecewise linear-quadratic (PLQ) functions

Our objective is to work with a class of functions that is closed under the standard convex operators (conjugation, addition, regularization, scalar multiplication). The class should be rich enough to approximate any convex function, and offer efficient numerical algorithms for all fundamental convex transforms namely addition, scalar multiplication, conjugation, regularization (by taking the Moreau envelope), and combinations.

Let $\mathcal{F}$ be the class of all convex lower-semicontinuous proper extended-valued functions with a piecewise linear subdifferential mapping $\partial f : \mathbb{R} \to 2^{\mathbb{R}}$. The set $\mathcal{F}$ contains the piecewise linear functions, the piecewise quadratic functions, and the sum of any such function with an indicator function. In [35, 10.20 p. 440], $\mathcal{F}$ is defined as the set of convex lower-semicontinuous proper extended-valued functions with piecewise linear domain for which the function is either linear or quadratic on each piece of its domain. A function in $\mathcal{F}$ is called a piecewise linear-quadratic (PLQ) function. In the present paper, we keep the same name even though for functions of one variable the class is usually called convex piecewise quadratic functions (the classes differ when considering functions of more than one variable). (Note that in the present paper, a PLQ function is always convex.) The PLQ functions provide a natural class of functions for convex calculus.

Within that framework, we have the following properties.

Proposition 5.1 (Closedness under convex transforms)**.** *The class of PLQ functions is closed under positive scalar multiplication, addition, conjugation, and taking the Moreau envelope.*

Proof. It is clear that the class is closed under (positive) scalar multiplication and addition. Using Formula (3), it will be closed by taking Moreau envelope as soon as it is closed for conjugation.

So we only need to prove that the conjugate of a PLQ function f is a PLQ function. The fact that f^* is PLQ comes from [35, 11.14 p. 484]. Alternatively, one can notice that the graph of ∂f^* is piecewise linear as it is the symmetric with respect to the line $y = x$ of the graph of f. So f^* is PLQ.

An alternate proof is to note that the proposition is contained in [35, 10.22, 11.14, 11.32, 11.33]. □

Proposition 5.2 (Efficient Algorithms)**.** *All fundamental convex operators, addition, scalar multiplication, conjugation, and taking the Moreau envelope, can be computed in linear time and space within the class of PLQ functions.*

Proof. Assume f, f_0, f_1 are PLQ functions.

First for scalar multiplication, take any $\alpha \in \mathbb{R}$. The function αf is defined on the same grid as the function f (only its values are multiplied by α). So we can compute it in linear time and space by multiplying each coefficients of f by α.

Next we consider the addition of f_0 and f_1. The function $f_0 + f_1$ is PLQ and can be computed in linear time and space *i.e.* in $O(n + m)$, where n and m are the number of intervals partitioning the domain of f_0 and of

f_1, respectively. Indeed, elementary computations show that the function $f_0 + f_1$ is PLQ on the grid $\{x_i\} \cup \{y_j\}$, where x_i (resp. y_j) is the grid of f_0 (resp. f_1).

For the conjugation operation, we first provide the details for a representative special case. Consider the PLQ function f defined by

$$f(x) := \begin{cases} \varphi_0(x) := a_0x^2 + b_0x + c_0 & \text{if } x \le x_0, \\ \varphi_1(x) := a_1x^2 + b_1x + c_1 & \text{otherwise.} \end{cases}$$

It is convex if, and only if,

$$\begin{cases} \varphi_0(x_0) = \varphi_1(x_0) & \text{(continuity condition),} \\ \varphi_0'(x_0) \le \varphi_1'(x_0) & \text{(convexity condition).} \end{cases}$$

Then its conjugate is directly computed as

$$f^*(s) = \begin{cases} \varphi_0^*(s) & \text{if } s \le \varphi_0'(x_0), \\ \bar{s}(s - \varphi_0'(x_0)) + \varphi_0^*(\varphi_0'(x_0)) & \text{if } \varphi_0'(x_0) \le s \le \varphi_1'(x_0), \\ \varphi_1^*(s) & \text{if } s \ge \varphi_1'(x_0); \end{cases}$$

where

$$\bar{s} := \frac{\varphi_1'(x_0)x_0 - \varphi_1(x_0) - \varphi_0'(x_0)x_0 + \varphi_0(x_0))}{\varphi_1'(x_0) - \varphi_0'(x_0)}.$$

In fact, the middle case, which only arises when the function f' is not continuous at x_0, is an affine function bridging the graph of φ_0^* with φ_1^*. Now computing f^* can be done in constant time and space since the computation of φ_0^* and φ_1^* can be performed explicitly directly.

The result follows by considering a general (convex) PLQ function as a sequence of quadratic (or linear) functions from left to right, taking boundary points of the domain into account. Applying variants of the above special case to each interval in the domain of f individually, we can compute the coefficients of that part in constant time and space. Hence, computing f^* takes linear time and space in the number of intervals that partition its domain.

Finally, the existence of a linear time and space algorithm for the Moreau envelope is a direct consequence of Formula (3) and of the above algorithms. □

Corollary 5.3. *The proximal average of two convex PLQ functions is a convex PLQ function that can be computed in linear time and space.*

Proof. The proximal average decomposes into conjugation, addition, and scalar multiplication, so it is a closed operation on the class of PLQ functions and can be computed in linear time and space. □

The key results in the present section are not only that the class of convex PLQ functions is closed under the operations of addition, positive scalar multiplication, conjugation, and regularization, but also that for each operation we can explicitly compute the resulting transform in linear time in the size of the grid; in other words, each transform has an associated efficient algorithmic implementation.

One can view the PLQ algorithms as hybrid symbolic-numeric algorithms: they provide exact results on the class of PLQ functions. Hence, computing the transform of any convex function decomposes into two step: a numerical approximation step which returns a PLQ function, and a symbolic computation step which returns the transform of that PLQ function. The later step is exact up to floating point arithmetic while the former relies on convergence.

Note also that the domain of a PLQ function is explicitly modeled, and that PLQ functions form an explicit second order model, which resolve the issues pointed out in Section 4. Consider again Example 2.4. The PLQ algorithms compute the (exact) proximal average

$$\mathcal{P}(f_0, \lambda, f_1) = (2\lambda - 1)x - 2\lambda(1-\lambda).$$

Compare the simplicity of that answer with the one provided by fast algorithms in Example 2.4: no convergence is necessary to approximate an intermediate function, or its domain.

6. Numerical Examples

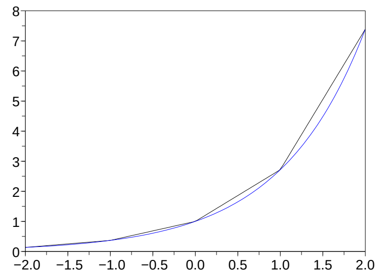
Several examples are presented here to illustrates the powerful convex calculus provided through the PLQ class.

The transform of any function belonging to the PLQ class is computed symbolically. So we immediately obtain the conjugate of the absolute value $(|\cdot|)^* = I_{[-1,1]}$, the Moreau envelope of the absolute value, and that the energy function is self-conjugate: $(0.5|\cdot|^2)^* = 0.5|\cdot|^2$. We also obtain that for any linear function $f(x) := ax$, its conjugate is the indicator function $f^* = I_{\{a\}}$. Hence, we do not need to resort to symbolic computation packages like [14] to build simple examples with piecewise quadratic functions.

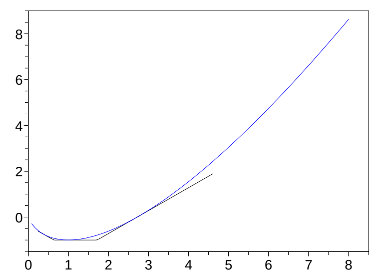
To illustrate Section 4, consider Figure 3(a), which shows a zeroth order model approximating the exponential function. Figure 3(b) shows the corresponding conjugate: $f^*(x) = x\ln(x) - x$. Compare with Figure 3(c) (resp. Figure 3(d)) which shows a first-order model approximating the exponential (resp. which shows the conjugate of the first-order model).

To compare the PLQ framework of Section 5 with the Fast Algorithm framework of Section 2, consider Figure 4. The addition of two quadratic functions on disjoint grids (15 points equispaced for function f_1 but 5 points equispaced for function f_2) results in a visible error for the fast algorithms but no visible difference for PLQ functions. (Recall that addition is one of the missing operators for the parametric framework of Section 3, which motivated the models of Section 4.)

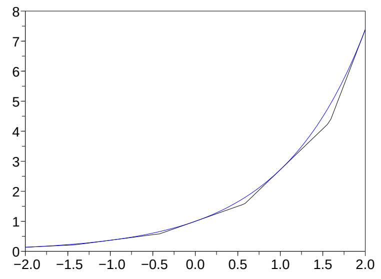
Finally, Figure 5(a) shows the proximal average for Example 2.4 computed using the PLQ algorithm. The colour scheme corresponds to the value of the parameter λ in $\mathcal{P}(f_0, \lambda, f_1)$. Figure 5(b) illustrates another example for which the PLQ algorithm is exact: the proximal average of the indicator function of a single point with the indicator function of (another) single



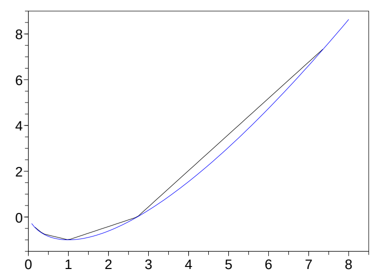
(a) Zeroth order approximation.



(b) Conjugate of the zeroth-order model

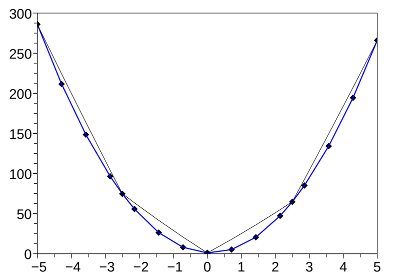


(c) First order approximation.

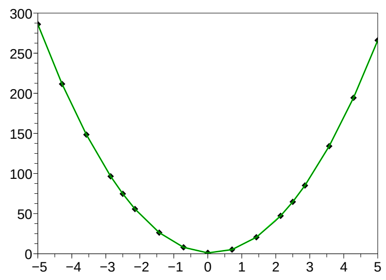


(d) Conjugate of the first-order model

FIGURE 3. Approximation of the exponential by a zeroth- and first-order models with the corresponding conjugates.



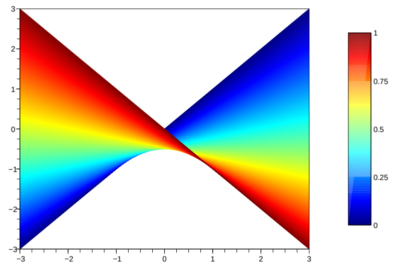
(a) Fast algorithms (using piecewise linear functions).



(b) PLQ (using piecewise quadratic functions).

FIGURE 4. Comparing the addition of functions $f_1(x) := (x-1)^2$ and $f_2(x) := x^2$.

point. (The intermediate functions are indicator functions, see [2] for the explicit formula).


(a) Proximal average of $f_0(x) := -x$ with $f_1(x) := x$.

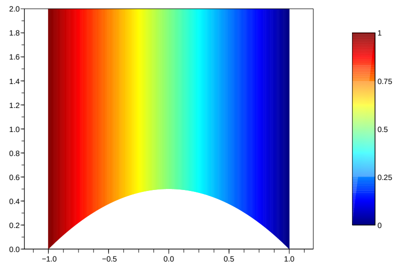

(b) Proximal average of $f_0 := I_{\{-1\}}$ with $f_1 := I_{\{1\}}$

FIGURE 5. Proximal averages computed exactly by the PLQ algorithm.

7. THE PLQ TOOLBOX

The PLQ toolbox is a collection of functions to manipulate PLQ functions implemented in Scilab v4.0. A PLQ function is stored as an $n \times 4$ matrix *e.g.* the function

$$f(x) := \begin{cases} a_0x^2 + b_0x + c_0 & \text{if } x \le x_0, \\ a_1x^2 + b_1x + c_1 & \text{if } x_0 < x \le x_1, \\ \vdots & \vdots \\ a_{n-1}x^2 + b_{n-1}x + c_{n-1} & \text{if } x_{n-1} < x \le x_{n-1}, \\ a_nx^2 + b_nx + c_n & \text{otherwise.} \end{cases}$$

is stored as the matrix

$$\texttt{plqf} := \begin{bmatrix} x_0 & a_0 & b_0 & c_0 \\ x_1 & a_1 & b_1 & c_1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{n-1} & a_{n-1} & b_{n-1} & c_{n-1} \\ +\infty & a_n & b_n & c_n \end{bmatrix},$$

with bounded domains stored by $a_0 := b_0 := 0$, $c_0 := +\infty$, and $a_n := b_n := 0$, $c_n := +\infty$. The indicator function $f(x) := I_{\{x_0\}}(x)$ of a single point x_0 is stored, as a special case, as the row vector $\texttt{plqf} := [x_0 \quad 0 \quad 0 \quad +\infty]$.

The toolbox provides the functions listed on Table 1. For example, $\texttt{plq\_lft}([+\infty, 1/2, 0, 0]) = [+\infty, 1/2, 0, 0]$ means the conjugate of the energy $x^2/2$ is the energy. Similarly, the fact the conjugate of the absolute

Function Name	Description
`plq_eval(plqf,X)`	Evaluate a PLQ function on the grid X.
`plq_build(x,f,df)`	Build a first-order model of a function.
`plq_add(plqf1,plqf2)`	Addition.
`plq_me(plqf,`λ`)`	Moreau envelope.
`plq_scalar(plqf,`λ`)`	Scalar multiplication.
`plq_pa(plqf1,plqf2,`λ`)`	Proximal average.
`plq_lft(plqf)`	Conjugation.

TABLE 1. Functions provided in the PLQ package.

LISTING 1. Computing the conjugate of the exponential

```
1 plqf = plq_build([-2,-1,0,0.5],exp,exp)
2 plqfstar=plq_lft(plqf)
3 x=linspace(-2,2)';
4 y=plq_eval(plqf,x);
5 s=linspace(0,2)';s=s(2:$);
6 ystar=plq_eval(plqfstar,s);
7 scf(0);clf();scf(1);clf();
8 plot2d(x,y);plot2d(x,exp(x),style=2);
9 plot2d(s,ystar);plot2d(s,s.*log(s)-s,style=2)
```

value is the indicator function of $[-1,1]$ is written as

$$\texttt{plq\_lft}\left(\begin{bmatrix} 0 & 0 & -1 & 0 \\ +\infty & 0 & 1 & 0 \end{bmatrix}\right) = \begin{bmatrix} -1 & 0 & 0 & +\infty \\ 1 & 0 & 0 & 0 \\ +\infty & 0 & 0 & +\infty \end{bmatrix},$$

and the Moreau envelope of the absolute value is computed by

$$\texttt{plq\_me}\left(\begin{bmatrix} 0 & 0 & -1 & 0 \\ +\infty & 0 & 1 & 0 \end{bmatrix}, \frac{1}{2}\right) = \begin{bmatrix} -\frac{1}{2} & 0 & -1 & -\frac{1}{4} \\ \frac{1}{2} & 1 & 0 & 0 \\ +\infty & 0 & 1 & -\frac{1}{4} \end{bmatrix}.$$

Note that all these computations do not involve numerical approximation or convergence — they are exact!

Listing 1 illustrates how one can quickly investigate the conjugate of a given function. Line 1 computes a PLQ approximation of the exponential function, and Line 2 computes the conjugate of the exponential: $s\ln(s) - s$. The remaining lines plot the PLQ function along with their exact counterparts. Line 3 builds a uniform grid x of 100 points between -2 and 2. Line 4 evaluates the PLQ function on the grid x, and Line 8 plots both the exponential function and its PLQ approximation. Line 5 builds a grid for the dual space (the singular value 0 is removed), and the conjugate function is plotted along its PLQ approximation on Line 9.

As for numerical performance, the PLQ package is compared with previous computational convex analysis algorithms in Table 2. The Transform column indicates which transform (Moreau or Conjugate) the core algorithm computes, the Scales column indicates whether the algorithm scales immediately to higher dimension, and the Order column indicates up to what derivative the algorithm takes into account (zeroth order for incorporating function values, first order for using also the values of the first derivative, etc.). The algorithms are fully detailed in [27]. Briefly, `Brute` and `Direct` are provided for comparison only (`Brute` stands for brute force while `Direct` uses the separability of the dot product to speed up the computations in higher dimensions). The `FLT` is historically the first algorithms considered and runs in log-linear time (provided for comparison only, see [23]). The `LLT`, `NEP`, and `PE` belong to the fast algorithm framework of Section 2 and are presented in [24], [10], and [27] respectively. The `PLT` belongs to the parametric framework of Section 3 and is detailed in [18]. The `PLQ` column stands for algorithms in the PLQ framework. The current state-of-the-art as summarized in the table implies that one should use the PLQ framework to compute with univariate functions, and the fast algorithms for functions of more than one variable.

Algorithm	Transform	Convex Only	Scales	Order	Complexity ($\mathbb{R}^d$)
`Brute`	Both	No	Yes	0	N^2
`Direct`	Both	No	Yes	0	$N^{1+1/d}$
`FLT`	Conjugate	No	Yes	0	$N^d \log N$
`LLT`	Conjugate	No	Yes	0	dN
`PE`	Moreau	No	Yes	0	dN
`NEP`	Moreau	Yes	Yes	0	dN
`PLT`	Conjugate	Yes	No	1	N
`PLQ`	Conjugate	Yes	No	2	N

Table 2. Computational Convex Analysis Algorithm Comparison.

Table 3 provides a running time comparison of the algorithms. The test were run on an IBM Thinkpad T60 with Intel dual core CPU at 1.8GHz under Linux Mandriva 2007 running Scilab v4.0. The table confirms the linear running time of our implementations.

Table 3 columns `PLQ`, `PLTc`, `PLTi`, `LLTc`, `LLTi` correspond to the PLQ algorithm for computing the conjugate (using interpreted Scilab syntax), the PLT algorithm (Parametric Legendre Transform) using vectorized Scilab syntax, the PLT algorithm using interpreted Scilab syntax, the LLT algorithm (Linear-time Legendre Transform) using Scilab vectorized syntax, and the LLT algorithm using interpreted Scilab syntax respectively. Note that the c in `LLTc` and `PLTc` stands for compiled since these implementations are almost as efficient as compiled functions.

Although the ranking favours `PLTc` then `LLTc`, the comparison with `PLTi` and `LLTi` shows that most of the difference comes from taking advantage of Scilab optimized matrix operations. In contrast, the PLQ algorithm for computing conjugates cannot be straightforwardly put into matrix operations. So it should really be compared with `PLTi` and `LLTi`. In that context, the PLQ algorithm is very competitive, achieving an order of magnitude similar to matrix-optimized codes like `LLTc` and `PLTc`. In addition, keep in mind that the PLT needs to be completed by linear interpolation, and the LLT algorithm requires both a priori knowledge of the domain of the conjugate, and convergence properties to achieve similar results as the PLQ algorithm. In other words, the PLQ algorithm has many significant structural advantages, while enjoying similar computational efficiency as the LLT and PLQ algorithms.

Remark 7.1. *If one compares Table 3 with* [18, Table 1], *one needs to take into account that we computed the conjugate instead of the Moreau envelope, we also assumed the data is convex and did not include the computation of the convex hull with the timing of the LLT algorithm, and finally we used vectorized versions of all three algorithms. Vectorization accounts for the difference between our timings of the LLT and the timings reported in* [18].

n	`PLQ`	`PLTc`	`PLTi`	`LLTc`	`LLTi`
10,000	0.04	0.00	0.29	0.02	2.20
20,000	0.10	0.01	0.57	0.03	8.61
30,000	0.15	0.01	0.86	0.05	18.75
40,000	0.20	0.02	1.20	0.06	33.16
50,000	0.29	0.02	1.50	0.09	49.71
60,000	0.31	0.03	1.79	0.10	79.84
70,000	0.41	0.03	2.21	0.13	106.56
80,000	0.49	0.05	2.54	0.17	146.57
90,000	0.54	0.05	2.92	0.20	174.68
100,000	0.61	0.06	3.22	0.23	223.60
110,000	0.66	0.06	3.57	0.21	270.26
120,000	0.73	0.08	3.73	0.21	330.63

TABLE 3. Numerical comparison of the fast algorithms, parametric and PLQ framework for computing the conjugate of the function $f(x) := x^2/2$ on the interval $[-n/2, n/2]$. The results are in seconds.

Table 4 shows computational results for the Moreau envelope. Note that the `LLTi` implementation was not included since it takes a prohibitive amount of time to run for these values of n. The conclusions are the same as for computing the conjugate: the PLQ algorithm for the Moreau envelope is competitive with compiled versions of the PLT and LLT algorithms (in this case, `PLQ` even outperforms `LLTc`) while providing significant structural advantages.

n	`PLQ`	`PLTc`	`PLTi`	`LLTc`
10,000	0.05	0.01	0.58	0.03
20,000	0.13	0.02	1.32	0.06
30,000	0.19	0.02	1.99	0.15
40,000	0.26	0.03	2.76	0.23
50,000	0.33	0.04	3.28	0.33
60,000	0.38	0.04	3.91	0.45
70,000	0.46	0.06	4.69	0.59
80,000	0.45	0.05	5.19	0.75
90,000	0.50	0.07	6.01	0.83
100,000	0.67	0.08	6.66	0.98
110,000	0.74	0.09	7.28	1.19
120,000	0.80	0.08	8.03	1.57

Table 4. Numerical comparison of the fast algorithms, parametric and PLQ framework for computing the Moreau envelope of the function $f(x) := x^2/2$ on the interval $[-n/2, n/2]$. The results are in seconds.

8. Conclusion

To conclude, we reviewed the two existing frameworks in computer-aided convex analysis: fast algorithms, and parametric algorithms. We pointed out their intrinsic limitations, and presented a new framework, based on PLQ functions, that allows for hybrid symbolic-numeric algorithms, and enjoys the following advantages:

- The class of PLQ functions is closed under the standard operations of convex analysis. As such it offers a natural framework for investigation.
- The PLQ functions give rise to natural algorithms with a linear worst-case time (and space) complexity. Moreover, the algorithms are easy to implement: they require only a single loop and no advanced data structures.
- The algorithms do not require a priori knowledge of the domain of the conjugate. Furthermore, there is no need to enlarge the domain of the function, which would be unavoidable when relying on convergence results. Both properties simplify significantly the manipulation of functions when one computes advanced convex operators like the proximal average.

- The PLQ functions offer a flexible framework in which a function can be initially numerically approximated using a zeroth-order model, a first-order model, or a second-order model.

While the parametric framework of Section 3 is limited to functions of one variable, fast algorithms extend to functions of several variables using the key fact that computing multivariate conjugates amounts to computing several univariate conjugates. This key property named factorization applies to all convex operators, and has been used in all fast algorithms [23, 24, 27].

The existence of efficient algorithms for the PLQ framework for multivariate functions remains a challenging topic of future research. For example, bivariate PLQ functions have a domain which is the intersection of linear functions, in other words it is a triangulation of the plane.

References

[1] H. H. Bauschke, J. V. Burke, F. R. Deutsch, H. S. Hundal, and J. D. Vanderwerff, *A new proximal point iteration that converges weakly but not in norm*, Proc. Amer. Math. Soc., 133 (2005), pp. 1829–1835 (electronic).

[2] H. H. Bauschke, Y. Lucet, and M. Trienis, *How to transform one convex function continuously into another*, tech. rep., University of British Columbia, July 2006.

[3] H. H. Bauschke, E. Matoušková, and S. Reich, *Projection and proximal point methods: Convergence results and counterexamples*, Nonlinear Anal., 56 (2004), pp. 715–738.

[4] H. H. Bauschke and M. von Mohrenschildt, *Symbolic computation of Fenchel conjugates*, SIGSAM Bull., (2006).

[5] J. M. Borwein and C. H. Hamilton, *Symbolic computation of multidimensional Fenchel conjugate*. Accepted for publication in Mathematical Programming, 2006.

[6] Y. Brenier, *Un algorithme rapide pour le calcul de transformées de Legendre–Fenchel discrètes*, C. R. Acad. Sci. Paris Sér. I Math., 308 (1989), pp. 587–589.

[7] L. Corrias, *Fast Legendre–Fenchel transform and applications to Hamilton–Jacobi equations and conservation laws*, SIAM J. Numer. Anal., 33 (1996), pp. 1534–1558.

[8] L. Deniau, *Proposition d'un opérateur géométrique pour l'analyse et l'identification de signaux et images*, PhD thesis, Université de Paris-Sud, Centre d'Orsay, Dec. 1997.

[9] L. Deniau and J. Blanc-Talon, *Fractal analysis with Hausdorff distance under affine transformations*, tech. rep., ETCA-CREA-SP, 1995.

[10] P. F. Felzenszwalb and D. P. Huttenlocher, *Distance transforms of sampled functions*, Tech. Rep. TR2004-1963, Cornell Computing and Information Science, Sept. 2004.

[11] U. Frisch and J. Bec, *Burgulence*, in Les Houches 2000: New Trends in Turbulence, A. M. Lesieur and e. F. David, eds., Springer EDP-Sciences, 2001, pp. 341–383.

[12] M. Gavrilova and M. H. Alsuwaiyel, *Two algorithms for computing the Euclidean distance transform*, Tech. Rep. 2000-661-13, Computer Science Technical Reports, University of Calgary, 2000.

[13] O. Güler, *On the convergence of the proximal point algorithm for convex minimization*, SIAM J. Control Optim., 29 (1991), pp. 403–419.

[14] C. H. Hamilton, *Symbolic convex analysis*, Master's thesis, Simon Fraser University, 2005.

[15] P. Helluy, *Simulation numérique des écoulements multiphasiques : De la théorie aux applications*, PhD thesis, Institut des Sciences de l'Ingenieur de Toulon et du

Var, Laboratoire Modélisation Numérique et Couplages, BP 56, 83162 La Valette CEDEX, France, Jan. 2005. Habilitation à Diriger des Recherches.

[16] J.-B. Hiriart-Urruty, *Lipschitz r-continuity of the approximate subdifferential of a convex function*, Math. Scand., 47 (1980), pp. 123–134.

[17] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms*, vol. 305–306 of Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], Springer-Verlag, Berlin, 1993. Vol I: Fundamentals, Vol II: Advanced theory and bundle methods.

[18] J.-B. Hiriart-Urruty and Y. Lucet, *Parametric computation of the Legendre–Fenchel conjugate*, tech. rep., University of British Columbia, 2005.

[19] T. Hisakado, K. Okumura, V. Vukadinovic, and L. Trajkovic, *Characterization of a simple communication network using Legendre transform*, in Proc. IEEE Int. Symp. Circuits and Systems, vol. 3, May 2003, pp. 738–741.

[20] B. Koopen, *Contact of bodies in 2D-space: Implementing the Discrete Legendre Transform*, AI Master's thesis, Intelligent Autonomous Systems Group, University of Amsterdam, Feb. 2002.

[21] B. Legras, I. Pisso, G. Berthet, and F. Lefvre, *Variability of the Lagrangian turbulent diffusion in the lower stratosphere*, Atmospheric Chemistry and Physics, 5 (2005), pp. 1605–1622.

[22] B. Lemaire, *The proximal algorithm*, in New methods in Optimization and their industrial uses (Pau/Paris, 1987), vol. 87 of Internat. Schriftenreihe Numer. Math., Birkhäuser, Basel, 1989, pp. 73–87.

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

[24] ———, *Faster than the Fast Legendre Transform, the Linear-time Legendre Transform*, Numer. Algorithms, 16 (1997), pp. 171–185.

[25] ———, *The Legendre-Fenchel conjugate: Numerical computation*, tech. rep., CECM, 1998.

[26] ———, *A linear Euclidean distance transform algorithm based on the Linear-time Legendre Transform*, in Proceedings of the Second Canadian Conference on Computer and Robot Vision (CRV 2005), Victoria BC, May 2005, IEEE Computer Society Press.

[27] ———, *Fast Moreau envelope computation I: Numerical algorithms*, Numerical Algorithms, 43 (2006), pp. 235–249. DOI - 10.1007/s11075-006-9056-0.

[28] J.-J. Moreau, *Propriétés des applications "prox"*, C. R. Acad. Sci. Paris, 256 (1963), pp. 1069–1071.

[29] ———, *Proximité et dualité dans un espace Hilbertien*, Bull. Soc. Math. France, 93 (1965), pp. 273–299.

[30] ———, *Convexity and duality*, in Functional Analysis and Optimization, Academic Press, New York, 1966, pp. 145–169.

[31] A. Noullez, S. N. Gurbatov, E. Aurell, and S. I. Simdyankin, *The global picture of self-similar and not self-similar decay in Burgers turbulence*, Tech. Rep. nlin.CD/0409022, arXiv.org eprint archive, Sept. 2004.

[32] A. Noullez and M. Vergassola, *A fast Legendre transform algorithm and applications to the adhesion model*, Journal of Scientific Computing, 9 (1994), pp. 259–281.

[33] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, New york, 1970.

[34] ———, *Monotone operators and the proximal point algorithm*, SIAM J. Control Optimization, 14 (1976), pp. 877–898.

[35] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*, Springer-Verlag, Berlin, 1998.

[36] Z.-S. She, E. Aurell, and U. Frisch, *The inviscid Burgers equation with initial data of brownian type*, Comm. Math. Phys., 148 (1992), pp. 623–641.

[37] F. Y. Shih and Y.-T. Wu, *Fast Euclidean distance transformation in two scans using a* 3×3 *neighborhood*, Comput. Vis. Image Underst., 93 (2004), pp. 195–205.

[38] P. Tseng and Z.-Q. Luo, *On computing the nested sums and infimal convolutions of convex piecewise-linear functions*, Journal of Algorithms, 21 (1996), pp. 240–266.

[39] K. Yosida, *Functional analysis*, Classics in Mathematics, Springer-Verlag, Berlin, 1995. Reprint of the sixth (1980) edition.

E-mail address: yves.lucet@ubc.ca
URL: http://people.ok.ubc.ca/ylucet/

E-mail address: heinz.bauschke@ubc.ca
URL: http://people.ok.ubc.ca/bauschke/

E-mail address: mjtrieni@interchange.ubc.ca

University of British Columbia - Okanagan, 3333 University Way, Kelowna BC V1V 1V7