Computational Convex Analysis: From Continuous Deformation to Finite Convex Integration

by

Michael Joseph Trienis

B.Sc., The University of British Columbia Okanagan, 2006

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

College of Graduate Studies

(Interdisciplinary)

The University Of British Columbia Okanagan

January 1st, 2007

Abstract

After introducing concepts from convex analysis, we study how to continuously transform one convex function into another. A natural choice is the arithmetic average, as it is pointwise continuous; however, this choice fails to average functions with different domains. On the contrary, the proximal average is not only continuous (in the epi-topology) but can actually average functions with disjoint domains. In fact, the proximal average not only inherits strict convexity (like the arithmetic average) but also inherits smoothness and differentiability (unlike the arithmetic average).

Then we introduce a computational framework for computer-aided convex analysis. Motivated by the proximal average, we notice that the class of piecewise linear-quadratic (PLQ) functions is closed under (positive) scalar multiplication, addition, Fenchel conjugation, and Moreau envelope. As a result, the PLQ framework gives rise to linear-time and linear-space algorithms for convex PLQ functions. We extend this framework to nonconvex PLQ functions and present an explicit convex hull algorithm.

Finally, we discuss a method to find primal-dual symmetric antiderivatives from cyclically monotone operators. As these antiderivatives depend on the minimal and maximal Rockafellar functions [5, Theorem 3.5, Corollary 3.10], it turns out that the minimal and maximal function in [12, p.132,p.136] are indeed the same functions. Algorithms used to compute these antiderivatives can be formulated as shortest path problems.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my supervisor, Dr. Yves Lucet, for giving me the opportunity to pursue research in the area of optimization and convex analysis. His infinite patience and guidance were vital in every aspect of this work. As well, I sincerely thank Dr. Donovan Hare for being the first professor to spark my interest in optimization from a real world perspective, and to Dr. Heinz Bauschke, whose passion for mathematics inspires us all. I also give thanks to Liangjin Yao and Melisa Lavallee, who helped in revising my thesis. Everyone who attended the OCANA seminar provided invaluable knowledge and constructive criticism.

This thesis was partially supported by the Pacific Institute for the Mathematical Sciences, the National Science and Engineering Research Council of Canada, and UBC Okanagan Internal Grant program.

Introduction

The main content of this thesis is derived from the papers [4], [5], and [19]. Contributions not contained in the previous papers include the extension to nonconvex PLQ functions, and the link with the all-pairs shortest path problem. We now summarize the contributions of each research paper.

Firstly, article [4] states that the proximal average allows a parametric family of convex functions to continuously transform one convex function into another, even when the domains of the two functions do not intersect. The proximal average operator is also shown to be an homotopy with respect to the epi-topology. Moreover, the paper also shows that the parametric family inherits desirable properties such as differentiability and strict convexity.

Next, article [19] presents a new computational framework for computer-aided convex analysis, and states that the class of piecewise linear-quadratic functions improves convergence and stability with respect to current models. A stable convex calculus is achieved by using symbolic-numeric algorithms to compute all fundamental convex transforms. The main result states the existence of efficient (linear-time) algorithms for the class of PLQ functions. These results are extended in the present thesis to nonconvex functions.

Finally, article [5] discloses a new method which always produces primal-dual symmetric antiderivatives using Fitzpatrick functions and the proximal midpoint average. A link with the all-pairs shortest path algorithms is given at the end of this thesis.

We first introduce fundamental convex notations in Chapter 1, then summarize the main results of the paper [4] in Chapter 2. In Chapter 3, we sum up and expand on several contributions in [19], which include explicitly defining the (linear-time) PLQ algorithms and extending the framework for those algorithms to nonconvex functions. Chapter 4 provides some notes on primal-dual symmetric methods, and links the minimal and maximal antiderivatives in [5, Theorem 3.5, Corollary 3.10] with [12, p.132, p.136]. The last chapter concludes the thesis with a brief summary and suggestions for future research.

Chapter 1

Preliminaries

In this section we recall basic notions from convex analysis.

Throughout, we assume the reader has a basic knowledge in set theory as well as fundamental results of calculus. We name $X = X^* = \mathbb{R}^d$, and $\bar{\mathbb{R}} = \mathbb{R} \cup \{+\infty\}$ with inner product $< \cdot | \cdot >$, and norm $\| \cdot \|$. We extend any proper function $f : \operatorname{dom} f \to \mathbb{R}$ with

$$\bar{f}(x) = \begin{cases} f(x) & \text{if } x \in \operatorname{dom} f, \\ \infty & \text{if } x \notin \operatorname{dom} f. \end{cases}$$

We can recover the domain of the original function f from $\bar{f}$ using the set

$$\operatorname{dom} f := \{x : \bar{f}(x) < +\infty\}.$$

As our framework involves functions with infinite values, we need to acquire a convention for algebra that involves infinite values. Table 1.1 summarizes the convention which will be used throughout. For the sake of simplicity, we define positive and negative reals $\mathbb{R}_{--} :=]-\infty, 0[$ and $\mathbb{R}_{++} :=]0, +\infty[$.

Definition 1.1. *A function $f : X \to \bar{\mathbb{R}}$ is proper if there is an $x \in X$ such that $f(x) < \infty$.*

Table 1.1: Let $\alpha \in \mathbb{R}_{++}$ and $\beta \in \mathbb{R}_{--}$. Then we have the following extended-valued convention.

Arithmetic	Multiplication
$\alpha + \infty = +\infty$	$\alpha(+\infty) = +\infty$
$\alpha - \infty = -\infty$	$\alpha(-\infty) = -\infty$
$\beta + \infty = +\infty$	$\beta(+\infty) = -\infty$
$\beta - \infty = -\infty$	$\beta(-\infty) = +\infty$
$+\infty + \infty = +\infty$	$0(+\infty) = 0$
	$0(-\infty) = 0$

1.1 Convex Sets

Let $x_1, x_2 \in X$. Then the point x is on the line segment $[x_1, x_2]$ if x is a convex combination of x_1 and x_2 *i.e.*

$$[x_1, x_2] = \{x \ : \ \lambda_1 + \lambda_2 = 1, \lambda_1, \lambda_2 \geq 0 \text{ and } x = \lambda_1 x_1 + \lambda_2 x_2\}.$$

See Figure 1.1.

Definition 1.2. *A set C is convex if for any $x_1, x_2 \in C \subset X$,*

$$x_1, x_2 \in C \Rightarrow [x_1, x_2] \subseteq C$$

holds.

The epigraph is a notion which relates convex sets to convex functions.

Definition 1.3. *The epigraph of a function f is the set of all points on or above the graph of f. That is,*

$$\operatorname{epi} f = \{(x, r) \ : \ x \in X, r \in \mathbb{R}, r \geq f(x)\}.$$

The above definition is illustrated on Figure 1.2.

We can also describe a convex set C as the convex combination of all points in C.

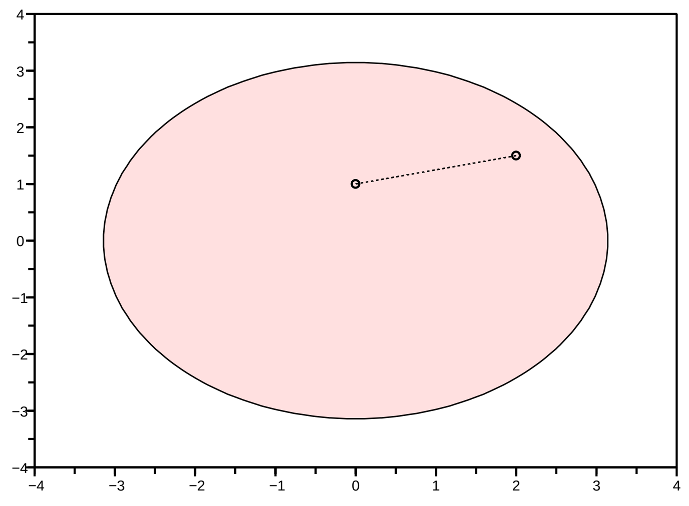


Figure 1.1: If the points $x_1, x_2 \in C \subset \mathbb{R}^2$, and $\lambda \in [0,1]$ then $\lambda x_1 + (1-\lambda)x_2 \in C$. So C is a convex set.

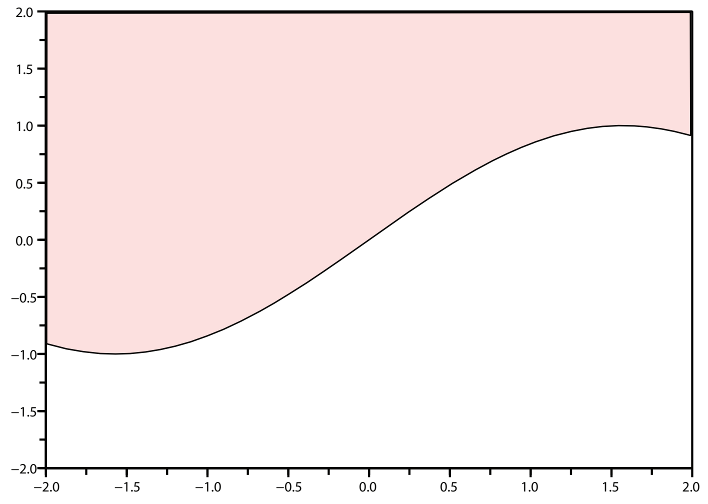


Figure 1.2: The epigraph of $x \mapsto \sin(x)$ is the set of all points on and above the graph.

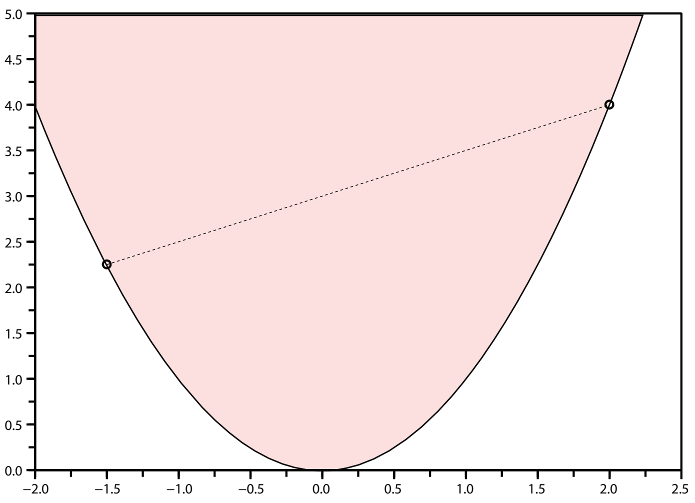


Figure 1.3: The segment $[(x, f(x)), (y, f(y))]$ is always above the graph of f.

Fact 1.4. *[20, Theorem 2.2] A set C is convex if and only if*

$$\forall n \in \mathbb{N}\ x_1, \cdots, x_n \in C \textit{ we have } \sum_{i=1}^{n} \lambda_i x_i \in C \textit{ for all } \lambda_i \geq 0 \textit{ where } \sum_{i=1}^{n} \lambda_i = 1.$$

1.2 Convex Functions

We now define convex functions.

Definition 1.5. *A function $f : X \to \bar{\mathbb{R}}$ is convex if $\operatorname{dom} f$ is a convex set and*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) \quad \big(\forall\ x, y \in \operatorname{dom} f\big),$$

where $\lambda \in [0, 1]$.

Geometrically, the inequality can be described as the line segment connecting $(x, f(x))$ and $(y, f(y))$ being always on or above the graph of the function f, as seen on Figure 1.3. The inequality becomes an equality when considering affine functions; therefore, affine functions are also convex. The notion of strict convexity is almost identical to the previous definition except that the inequality is strict.

Definition 1.6. *The function f is strictly convex if $\operatorname{dom} f$ is convex and*

$$f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda) f(y) \quad \big(\forall\ x, y \in \operatorname{dom} f\big),$$

whenever $x \neq y$, and $\lambda \in \,]0,1[\,$.

Remark 1.7. *Strict convexity is a stronger notion than convexity as all strictly convex functions are convex but the converse is false. A strictly convex function is a convex function without any linear parts. It is an important notion as strictly convex functions have at most one minimizer (see Figure 1.3).*

Notation 1.8. *The set of all convex functions on X is denoted by Conv X.*

Notation 1.9. *Any vector $x \in X = \mathbb{R}^d$ is identified as a column vector; the transpose, of $x \in \mathbb{R}^{d\times 1}$, is a new vector in $\mathbb{R}^{1\times d}$ denoted by x^T. Finally, we denote $\langle \cdot, \cdot \rangle$ as the standard dot product $\langle x, y\rangle = y^T x$.*

Definition 1.10. *The $n \times n$ matrix M is positive-semidefinite if $x^T M x \geq 0$ for all $x \neq 0$. Furthermore, a matrix M is positive-definite if $x^T M x > 0$ for all $x \neq 0$.*

Example 1.11. *Quadratic functions $f(x) = \langle Ax, x\rangle + \langle b, x\rangle + c$ with A positive-semidefinite, are convex. If A is positive definite, then f is strictly convex.*

Example 1.12. *The exponential function $f(x) = \exp(x)$ for $x \in \mathbb{R}$ is strictly convex with no minimizer.*

Fact 1.13. *[20, Theorem 4.1] The function f is convex, if and only if $\operatorname{epi} f$ is a convex set.*

Fact 1.14. *[7, p.639] A function $f : X \to \mathbb{R}$ is said to be closed if and only if* $\operatorname{epi} f$ *is closed.*

1.3 Fenchel Conjugate

The convex conjugate, also known as the Fenchel conjugate (we will refer to it simply as the conjugate) is an important operation in convex analysis. It is used as an intermediate transform for other more advanced operations like the Moreau envelope and the proximal average.

Definition 1.15. *Let $f : X \to \bar{\mathbb{R}}$. The function f^* is defined as*

$$f^*(y) = \sup_{x \in \operatorname{dom} f} \{\langle y, x\rangle - f(x)\}.$$

We notice that f^* is a convex function as the supremum of affine (convex) functions is a convex function.

Definition 1.16. *A set is closed if every limit point of the set is a point in the set.*

Proposition 1.17. *[7, p.91] The function f^* is always convex and lsc, (see Definition 1.40) even if f is not convex.*

Proof. The conjugate is the pointwise supremum of a family of affine functions of y, therefore the epigraph of the conjugate is an intersection of a family of closed half spaces (closed convex sets). Since the intersection of closed convex sets is a closed convex set, the epigraph is a closed convex set. So the conjugate is lsc and convex. □

Example 1.18. *The conjugate of a quadratic function $f(x) = a_0x^2 + b_0x + c_0$, the conjugate is*

$$f^*(y) = \begin{cases} \frac{1}{4a_0}(y - b_0)^2 - c_0 & \text{if } a_0 > 0, \\ I_{\{b_0\}}(y) - c_0 & \text{if } a_0 = 0, \\ \infty & \text{if } a_0 < 0. \end{cases}$$

The conjugate function has nice duality properties with respect to convexity.

Fact 1.19. *[20, Corollary 12.2] The biconjugate $f^{**} = f$, if and only if f is proper, convex and lsc.*

Example 1.20. *Let $f(x) = b_0x + c_0$. Then $f^*(y) = I_{\{b_0\}}(y) - c_0$. Taking the conjugate again we get $f^{**}(x) = b_0x + c_0 = f(x)$.*

Proposition 1.21. *[20, p.105] Given a function $f : X \to \bar{\mathbb{R}}$ and $x \in \operatorname{dom} f$, Fenchel's inequality holds:*

$$\langle p, x\rangle \leq f(x) + f^*(p) \quad \big(\forall\ p, x \in X\big).$$

Proof. Directly from the definition of the Fenchel conjugate $f^*(p) := \sup_x\{\langle p, x\rangle - f(x)\}$, gives $f^*(p) \geq \langle p, x\rangle - f(x)$. □

The energy function is the only function whose conjugate is itself.

Fact 1.22. *[20, p.106] The only self conjugate function is the energy function $\frac{1}{2}\|\cdot\|^2$.*

Notation 1.23. *The set of all closed convex functions on X is denoted by $\overline{Conv}\ X$.*

As the conjugate function is always convex, taking the double conjugate yields the closed convex hull.

Fact 1.24. *[20, p.36] Let f be a proper function. Then $\bar{co}\ f = f^{**}$.*

Proposition 1.25. *[11, Corollary 1.4.4] Let $f \in \overline{Conv}\ X$ and $x \in \operatorname{dom} f$. Then*

$$\langle s, x\rangle = f(x) + f^*(s) \Leftrightarrow s \in \partial f(x) \Leftrightarrow x \in \partial f^*(s). \tag{1.1}$$

Proof. By definition of the conjugate we have

$$-f^*(s) = \inf_{x\in X} [f(x) - \langle s, x\rangle]. \tag{1.2}$$

So the lsc proper convex function g: $x \mapsto f(x) - \langle s, x\rangle$, achieves its infimum at $\hat{x}$ if and only if $0 \in \partial g(\hat{x}) = \partial f(\hat{x}) - s$. That is,

$$-f^*(s) = [f(\hat{x}) - \langle s, x\rangle] \text{ if and only if } s \in \partial f(\hat{x}).$$

Applying this same result to f^*, we obtain

$$x \in \partial f^*(s) \text{ if and only if } \langle s, x\rangle = f^*(s) + f^{**}(x),$$

which is again Equation (1.1) since $f^{**} = f$. □

1.4 Moreau Envelope

Let us first define an operation which is used in the Moreau envelope.

Definition 1.26. *Let f_1 and f_2 be two functions from X to $\bar{\mathbb{R}}$. Their infimal convolutions is defined* *by*

$$\begin{aligned} f_1 \Box f_2(x) &:= \inf\{f_1(x) + f_2(x_2) \;:\; x_1 + x_2 = x\}, \\ &= \inf_{y \in X} [f_1(y) + f_2(x-y)]. \end{aligned}$$

Definition 1.27. *[20, Theorem 31.5.] Let $\lambda \in \mathbb{R}_{++}$ and $s \in X$. Then the Moreau envelope, also called the Moreau-Yosida regularization is defined as*

$$M_\lambda(s) = f \Box \frac{1}{2\lambda}\|\cdot\|^2(s) = \inf_{x \in X} f(x) + \frac{\|s-x\|^2}{2\lambda}. \tag{1.3}$$

We summarize some of its key properties.

Fact 1.28. *(i.) [21, Theorem 1.25] M_λ converges pointwise to f as λ decreases to 0.*

(ii.) [21, Theorem 13.37] The Moreau envelope is smooth and continuous.

(iii.) [18, p.2] The functions $M_\lambda(x)$ and $f(x)$ share the same critical points.

It is also important due to its regularization properties, in particular, with nondifferentiable functions as seen in Figure 1.4.

Using Equation (1.3) and expanding $\|\cdot\|^2$ we obtain

$$M_\lambda(s) = \frac{\|s\|^2}{2\lambda} - \frac{1}{\lambda} g_\lambda^*(s), \tag{1.4}$$

where $g_\lambda^*(s) = \sup\{\langle s, x\rangle - g_\lambda(x)\}$ and $g_\lambda(x) = \frac{\|x\|^2}{2} + \lambda f(x)$ (see [17]).

Remark 1.29. *Formula (1.4) is important, as any algorithm used to compute the conjugate can also be used to compute the Moreau envelope. As the Moreau envelope is decomposed into conjugation, addition, and (positive) scalar multiplication, all algorithms and models need only to accommodate*

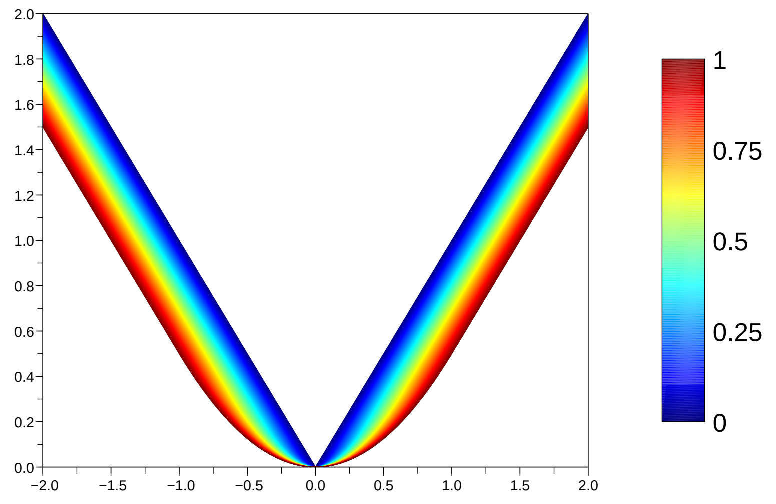


Figure 1.4: The Moreau envelope of $f(x) = |x|$ is a smooth PLQ function.

these operations. In other words, the Moreau envelope (regularization) depends entirely on the three operations in Equation (1.4).

1.5 Convex Optimization

Duality theory in optimization associates another optimization problem (the dual) to a given problem (the primal). The primal problem is the initial model which typically consists of an objective function and a series of constraints. The conjugate plays a critical role in convex duality because it gives a dual representation (D) of the primal (P). Let h and k be proper, lsc and convex on $\mathbb{R}^n$

and $\mathbb{R}^m$ respectively. Then

$$\begin{aligned} \text{(P)}\ \inf\ \{\phi(x)\ :\ \phi(x) &:= \langle c,x\rangle + k(x) + h(b-Ax),\ \text{and } x\in\mathbb{R}^n\}, &(1.5)\\ \text{(D)}\ \sup\ \{\psi(y)\ :\ \psi(y) &:= \langle b,y\rangle - h^*(y) - k^*(A^*y-c),\ \text{and } y\in\mathbb{R}^m\}. &(1.6)\end{aligned}$$

The dual problem clearly shows the importance of the conjugate function as Equation (1.6) relies directly on the conjugate of h and k.

Remark 1.30. *A version of Fenchel Duality Theorem involving the primal and dual problems, can be found in [20, Theorem 31.1].*

The notion of duality has had a huge impact as some dual problems are much easier to solve than their primal counterparts. The primal and the dual are intrinsically linked to each other such that the solution to the dual provides insight into the solution of the primal. Under some conditions, the optimal value of both problems is the same.

The indicator function of a convex set $C \subset X$ is the bridge between functions and sets. It is defined as

$$x \mapsto I_C(x) = \begin{cases} 0 & \text{if } x\in C,\\ +\infty & \text{otherwise.}\end{cases}$$

The indicator function will allow us to translate results on convex sets to convex functions.

Example 1.31. *In optimization we are concerned with the following problem*

$$\textit{minimize } f \textit{ over } C.$$

This problem can be equivalently rewritten as the unconstrained optimization problem

$$\textit{minimize } f + I_C \textit{ over } X,$$

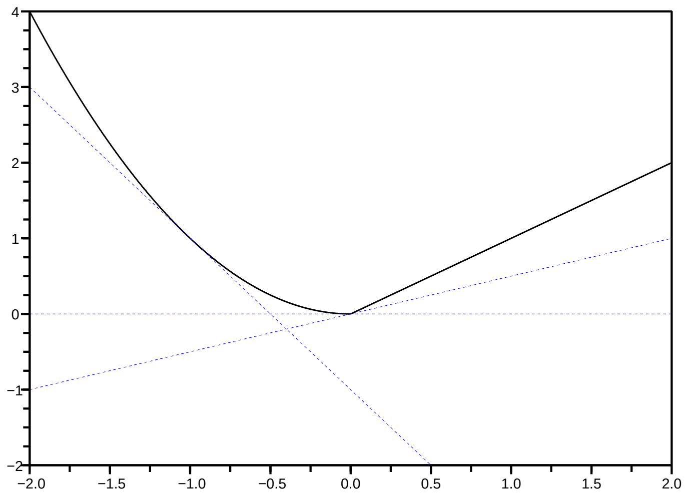


Figure 1.5: A set of subtangents for smooth and non-smooth parts.

as indicator functions act as infinite penalization.

Definition 1.32. *We say that a function is smooth if it is differentiable everywhere.*

In optimization we use critical points to locate minimum or maximum values.

Definition 1.33. *Let f be a differentiable function then a critical point is a point in the interior of the domain of the function where the derivative equals zero.*

As the gradient is undefined for nonsmooth functions, we require some notion of generalized differentiability; this is known as the subdifferential operator.

1.6 Subdifferential

In order to properly describe the construction of the subdifferential operator, we need to define some other notions. Given a proper convex function f, we can construct a hyperplane $f(x_0)+\langle s, x-x_0\rangle$ which passes through $(x_0, f(x_0))$ that always lies on or below the function f (seen in Figure 1.5). The slope of this supporting hyperplane is called a subgradient.

Definition 1.34. *The vector s is a subgradient of f at $x_0 \in \operatorname{dom} f$ if and only if*

$$f(x) \geq f(x_0) + \langle s, x - x_0\rangle \quad \big(\forall\ x \in X\big).$$

In the case where the function f has a corner at x_0, there may exist infinitely many supporting hyperplanes, and thus many subgradients. Figure 1.5 illustrates multiple subgradients at the corner $(0,0)$ and only a single subgradient otherwise. The subdifferential is simply the set of all possible subgradients at all points.

Definition 1.35. *The subdifferential of a proper function f at x is*

$$\partial f(x) = \{s \in X \ :\ f(y) \geq f(x) + \langle s, y - x\rangle,\ \forall\ y \in X\}\,.$$

It is constructed from hyperplanes which support $\operatorname{epi} f$ at $(x, f(x))$; therefore the subdifferential may be empty for nonconvex functions. Recall that the subdifferential is a generalization of the derivative and in the case when f is convex and smooth, $\{\nabla f(x)\} = \partial f(x)$.

Definition 1.36. *We say that $\bar{x}$ is a critical point of f if $0 \in \partial f(\bar{x})$.*

Remark 1.37. *When considering the class of conjugate functions which have closed form solutions, one strategy is to determine if $\langle y, x\rangle - f(x)$ is concave (in term of x). If this is the case then the critical point will maximize $\langle y, x\rangle - f(x)$. This is the approach taken by the symbolic convex*

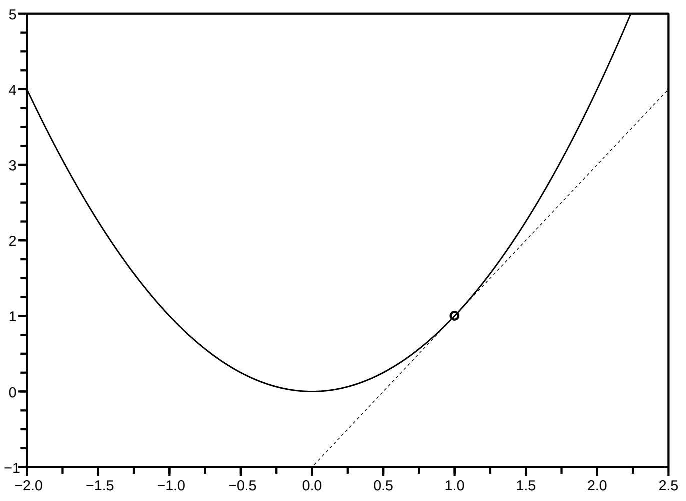


Figure 1.6: First-order convexity characterization.

analysis toolkit (SCAT) package [9, p.72]. However, when no closed form exists we require numerical methods to find a solution.

Convexity is characterized by the following first order condition.

Observation 1.38. *Assume f is differentiable at each point in* $\operatorname{dom} f$*, and that* $\operatorname{dom} f$ *is open. Then the function f is convex, if and only if,* $\operatorname{dom} f$ *is convex and*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle \quad \big(\forall\ x, y \in \operatorname{dom} f\big).$$

We can see from Figure 1.6 that the open circle is the point where the subtangent $f(x) + f'(x)(y-x)$ passes through $(x, f(x))$, and the constraint $f(y) \geq f(x) + f'(x)(y-x)$ means that the subtangent must lie on or below the function $f(y)$.

The following is a second-order characterization of convexity.

Fact 1.39. *[20, Theorem 4.5] Assume f is proper, twice continuously differentiable at each point in dom f, where the dom f is open and convex. Then the function f is convex on dom f, if and only if,*

$$\textit{for all } x \in \operatorname{dom} f, \nabla^2 f(x) \textit{ is positive semi-definite.}$$

If $\nabla^2 f(x)$ is positive-definite for all $x \in \operatorname{dom} f$, then f is strictly convex.

1.7 Continuity Notions

Definition 1.40. *A function f is lower semicontinuous (lsc) at $x_0 \in \operatorname{dom} f$ if for every $\varepsilon > 0$, there exists a neighbourhood U of x_0 such that $f(x) > f(x_0) - \varepsilon$, for all x in U.*

This definition can be equivalently expressed as

$$\liminf_{x \to x_0} f(x) \geq f(x_0).$$

Lower semicontinuity is a weaker notion than continuity in the sense that continuity implies lower semicontinuity. In other words, if a function f is continuous at a point x_0 then it is also lsc at x_0 but the converse is not true, as seen in Figure 1.7. If a function f is lsc at every point in its domain then f is a lsc function.

The notion of upper semicontinuous (usc) is analogous to that of lsc.

Definition 1.41. *The function f is usc at $x_0 \in \operatorname{dom} f$ if for every $\varepsilon > 0$, there exists a neighborhood U of x_0 such that $f(x) < f(x_0) + \varepsilon$ for all x in U.*

This definition can be equivalently expressed as

$$\limsup_{x \to x_0} f(x) \leq f(x_0).$$

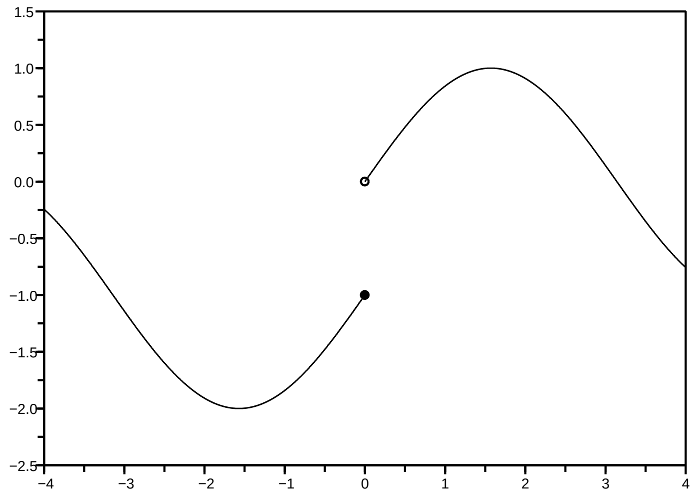


Figure 1.7: The function is lsc at $x_0 = 0$, and continuous otherwise therefore it is lsc everywhere.

A function f is lsc if and only if $-f$ is usc. Moreover, the notion of being continuous at x_0 is equivalent to being both lsc and usc at x_0.

Fact 1.42. *[20, Theorem 7.1] A function f is lsc if and only if the epigraph of f is closed.*

Definition 1.43 (continuous function)**.** *Assume that x_n, x_0 belong to* $\operatorname{dom} f$. *Then a real function f is continuous if for any sequence (x_n) such that*

$$\lim_{n\to\infty} x_n = x_0,$$

it holds that

$$\lim_{n\to\infty} f(x_n) = f(x_0).$$

Equivalently, we can express the above definition as the following.

Notation 1.44. *For all $\epsilon > 0$, there exists an $\sigma > 0$ such that*

$$|x - x_0| < \sigma \Rightarrow |f(x) - f(x_0)| < \epsilon.$$

Fact 1.45. *[20, p.51] A function f is lsc and usc at x_0 if and only if f is continuous at x_0.*

1.8 Convex Hull

Given a nonconvex function g, the convex hull (denoted by co) is found by taking the supremum of all convex functions which minorize g.

Definition 1.46. *If two functions f and g from X to $\mathbb{R}$ satisfy $f(x) \leq g(x)$ for all $x \in X$, we say that f minorizes g (on X).*

Fact 1.47. *[11, Proposition 2.5.1, Proposition 2.5.2] Let $g : X \to \bar{\mathbb{R}}$, not identically $+\infty$, be minorized by an affine function: for some $(s, b) \in X \times \mathbb{R}$ $g(x) \geq \langle s, x\rangle - b \quad \big(\forall\ x \in X\big)$. Then*

$$\operatorname{co} g(x) := \sup\{h(x) \ : \ h \in \ \mathit{Conv}\ X, h \leq g\}. \tag{1.7}$$

Similarly, the closed convex hull of g is

$$\overline{\operatorname{co}}\, g(x) := \sup\{h(x) \ : \ h \in \overline{\mathit{Conv}} X, h \leq g\}. \tag{1.8}$$

The above definition of a closed convex hull is illustrated on Figure 1.8.

The concept of convex functions is intrinsically linked to convex sets using the epigraph; therefore, we also define the convex hull for sets. If we are given any set S, not necessarily convex, we can construct a convex set by taking the intersection of all convex sets that contain S.

Fact 1.48. *[11, Proposition 1.3.4, Definition 1.4.1] The convex hull is the intersection of all convex sets which are supersets of S:*

$$\operatorname{co} S := \cap\{C \ : \ C \textit{ is convex and contains } S\ \}. \tag{1.9}$$

The closed convex hull of a nonempty set $S \subset X$ is the intersection of all closed convex sets containing S. It is also the intersection of all closed half-spaces containing S.

We can also find the convex hull of a set of discrete points by finding the set of all convex combinations.

Definition 1.49. *A convex combination of $x_1, \cdots, x_k$ is the point $x = \lambda_1 x_1 + \lambda_2 x_2 + \cdots + \lambda_k x_k$ such that $\sum_{i=1}^{k} \lambda_i = 1$, and $\lambda_i \geq 0$ for $i = 1, \cdots, k$.*

Fact 1.50. *[7, p.34] The convex hull of a finite set of points $\{x_1, \cdots, x_k\}$ is*

$$\operatorname{co}\{x_1, \cdots, x_k\} = \{\lambda_1 x_1 + \cdots + \lambda_k x_k \ : \ \lambda_1 + \cdots + \lambda_k = 1, \lambda_i \geq 0 \textit{ for } i = 1, \cdots, k\}.$$

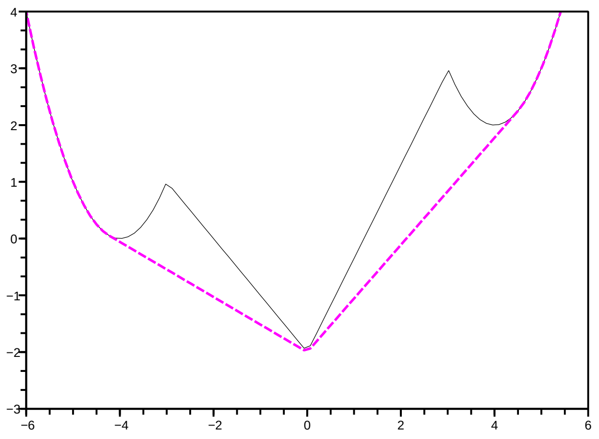


Figure 1.8: The closed convex hull.

Chapter 2

Proximal Average

Throughout this chapter we will assume f_0, and f_1 belong to $\overline{\text{Conv}}\ X$.

2.1 Arithmetic Average

The main motivation for the proximal average is to determine how to continuously transform f_0 into f_1. The first and most natural way is the arithmetic average function

$$x \mapsto (1-\lambda)f_0(x) + \lambda f_1(x) \quad \big(\forall \lambda \in \,]0,1[\,\big).$$

The arithmetic average is the convex combination of f_0 and f_1 taken pointwise. In fact, when f_0 and f_1 are finite-valued the arithmetic average is pointwise continuous.

Remark 2.1. *We say that a function is finite if it does not take the values $+\infty$ and $-\infty$.*

Modern convex analysis uses extended-valued functions for many valid reasons. Indeed, many constrained optimization problems are often reformulated as unconstrained optimization problems by introducing indicator functions in the objective (see Example 1.31). As many functions (including indicator functions) consider infinite values, we encounter our first shortcoming: when we take the arithmetic average of two functions which do not have identical domains, the domain of the average may be empty.

Example 2.2. *Let $f_0(x) := x\ln(x) - x$ and $f_1(x) := \exp(x)$. Then the arithmetic average function is*

$$a_\lambda(x) := (1-\lambda)f_0(x) + \lambda f_1(x) \quad \big(\forall \lambda \in \,]0,1[\,\big).$$

As illustrated by Figure 4.2(a), $x \mapsto a_\lambda(x)$ is only continuous on the interval $\operatorname{dom} f_0 \cap \operatorname{dom} f_1 = [0,+\infty[$. *When* $\operatorname{dom} f_0 \cap \operatorname{dom} f_1 = \emptyset$ *the arithmetic average is not even defined anywhere and will be exactly $+\infty$ everywhere while the proximal average always has full domain.*

In contrast to the arithmetic average, let us consider the proximal average. The proximal average originates from the convex combination of two proximal maps, (which turns out to be a proximal map)

$$\operatorname{Prox}(f_\lambda) = (1-\lambda)\operatorname{Prox}(f_0) + \lambda \operatorname{Prox}(f_1),$$

where the proximal mapping of $f \in \overline{\operatorname{Conv}}\, X$ defined by

$$\operatorname{Prox}(f)(x) = \operatorname{Argmin} f(y) + \frac{\|x-y\|^2}{2\lambda}, \tag{2.1}$$

is the set of minimizers of the Moreau envelope (see Equation (1.3)).

Definition 2.3. *The proximal average operator $\mathcal{P}$ is defined as*

$$\begin{aligned} \mathcal{P}\colon\ & \overline{\operatorname{Conv}} \times [0,1] \times \overline{\operatorname{Conv}} \to \overline{\operatorname{Conv}} \\ & (f_0,\lambda,f_1) \mapsto \big((1-\lambda)(f_0 + \tfrac{1}{2}\|\cdot\|^2)^* + \lambda(f_1 + \tfrac{1}{2}\|\cdot\|^2)^*\big)^* - \tfrac{1}{2}\|\cdot\|^2. \end{aligned}$$

The arithmetic average is convex as it is composed of operations which are convexity preserving. But it is unclear from the definition, whether the proximal average is convex, as the difference of convex functions is not always convex.

Fact 2.4. *[6] Set $f_\lambda = \mathcal{P}(f_0, \lambda, f_1)$. Then*

$$\big(\mathcal{P}(f_0, \lambda, f_1)\big)^* = \mathcal{P}(f_0^*, \lambda, f_1^*).$$

Using the Biconjugate Theorem with Fact 2.4 and conjugating the proximal average twice, gives

$$f_\lambda^{**} = \big(\mathcal{P}(f_0, \lambda, f_1)\big)^{**} = \big(\mathcal{P}(f_0^*, \lambda, f_1^*)\big)^* = \mathcal{P}(f_0^{**}, \lambda, f_1^{**}) = f_\lambda.$$

Thus, $f_\lambda \in \overline{\text{Conv}}\ X$.

Fact 2.5. *[4, Proposition 2.2] The following properties always hold for the proximal average operator:*

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

(ii.) $\mathcal{P}(f_0, 0, f_1) = f_0$,

(iii.) $\mathcal{P}(f_0, 1, f_1) = f_1$.

Proposition 2.6. *[4, Proposition 2.8] Let $f \in \overline{\mathit{Conv}}\ X$, then $\mathcal{P}(f, \frac{1}{2}, f^*) = \frac{1}{2}\|\cdot\|^2$.*

Proof. In fact,

$$\big(\mathcal{P}(f, \tfrac{1}{2}, f^*)\big)^* = \mathcal{P}(f^*, \tfrac{1}{2}, f^{**}) = \mathcal{P}(f^*, 1-\tfrac{1}{2}, f) = \mathcal{P}(f, \tfrac{1}{2}, f^*).$$

As the conjugate of $\mathcal{P}(f, \frac{1}{2}, f^*)$ is itself, by Fact 1.22 the proximal midpoint average $\mathcal{P}(f, \frac{1}{2}, \frac{1}{2}, f^*)$ is $\frac{1}{2}\|\cdot\|^2$. □

2.2 Continuity and Homotopy

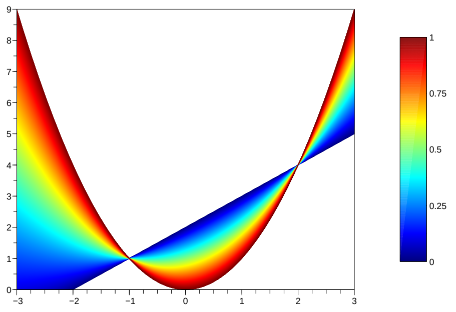
In contrast to the arithmetic average the proximal average is not pointwise continuous.

Example 2.7. *Let* $f_0(x) := I_{\{b_0\}}(x)$*, and* $f_1 := I_{\{b_1\}}(x)$*. Then*

$$f_\lambda(x) = I_{\{(1-\lambda)b_0+\lambda b_1\}}(x) + \frac{(1-\lambda)\lambda}{2}\|b_0 - b_1\|^2.$$

Let $\lambda_0 > 0$ *and set* $x_0 = (1-\lambda_0)b_0 + \lambda_0 b_1$*. Then for* $\lambda \neq \lambda_0$ *we have* $f_\lambda(x_0) = +\infty$*, but for* $\lambda = \lambda_0$ *we obtain* $f_\lambda(x_0) = \frac{(1-\lambda)\lambda}{2}\|b_0 - b_1\|^2$*. So the proximal average operator* $\lambda \mapsto \mathcal{P}(f_0, \lambda, f_1)(x)$ *is not pointwise continuous.*

Figure 2.2(b) illustrates that there may exist some type of continuity with the proximal average.


(a) From f_0 to f_1: Arithmetic Average

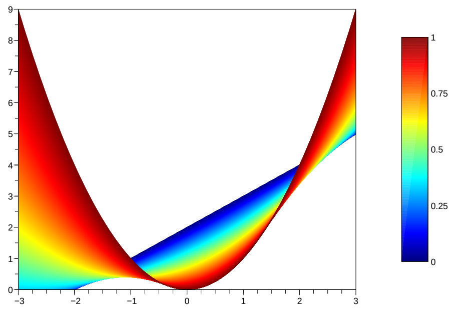

(b) From f_0 to f_1: Proximal Average

Figure 2.1: Averages of the linear function $f_0(x) = x + 2$ and the quadratic function $f_1(x) = x^2$.

Next, we quote several results from [4] that detail the setting in which the proximal average is continuous.

Definition 2.8 (epi-convergence and epi-topology)**.** *Let* g *and* $(g_n)_{n\in\mathbb{N}}$ *be functions from* X *to* $]-\infty, +\infty]$*. Then* $(g_n)_{n\in\mathbb{N}}$ *epi-converges to* g*, if the following properties hold for every* $x \in X$*:*

$$\text{for every sequence } (x_n)_{n\in\mathbb{N}} \text{ in } X \text{ converging to } x, \text{ one has } g(x) \leq \liminf_{n\to+\infty} g_n(x_n),$$

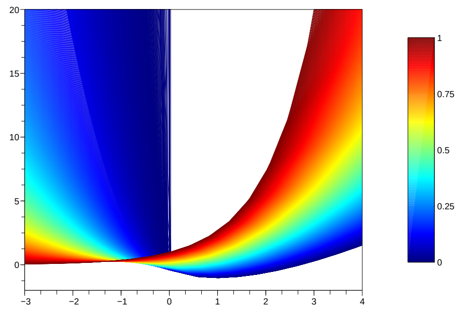
$$\text{there exists a sequence } (x_n)_{n\in\mathbb{N}} \text{ in } X \text{ converging to } x \text{ such that } \limsup_{n\to+\infty} g_n(x_n) \leq g(x).$$

The epi-topology is the topology induced by epi-convergence.

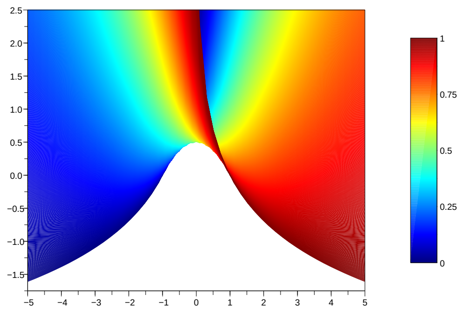
Fact 2.9. *[4, Theorem 4.2] If $\overline{Conv}\ X$ is equipped with the epi-topology, then the proximal average operator*

$$\mathcal{P}\colon \overline{Conv}\ X \times [0,1] \times \overline{Conv}\ X \to \overline{Conv}\ X$$

is continuous.



(a) Deforming $f_0(x) = x\ln(x) - x$ into $f_1(x) = \exp(x)$.



(b) Deforming $f_0(x) = -\ln(-x)$ into $f_1(x) = -\ln(x)$.

Figure 2.2: The proximal averaging of two functions with different domains.

Fact 2.10. *[4, Corollary 3.3] Let $f_0 \in \overline{Conv}\ X$ and $f_1 \in \overline{Conv}\ X$. Then*

$$\mathcal{P}(f_0, \cdot, f_1)\colon [0,1] \to \overline{Conv}\ X$$

provides a homotopy between f_0 and f_1. In fact, all functions in $\overline{Conv}\ X$ are homotopic for the epi-topology.

Next, we define the domain of the proximal average operator.

Fact 2.11. *[6] Let C_0 and C_1 be nonempty closed convex subsets of X, and $\lambda \in\ ]0,1[$. Set $f_0 := I_{C_0}$,*

$f_1 := I_{C_1}$, $f_\lambda := \mathcal{P}(f_0, \lambda, f_1)$. *Then for every* $x \in X$,

$$f_\lambda(x) = (1-\lambda)\lambda \inf\left\{\tfrac{1}{2}\|c_0 - c_1\|^2 \;:\; (1-\lambda)c_0 + \lambda c_1 = x, c_i \in C_i\right\}.$$

Consequently, $\operatorname{dom} f_\lambda = (1-\lambda)C_0 + \lambda C_1$.

Definition 2.12. *The set* $\overline{C} \subseteq X$ *is the set of all* x *such that there exists a sequence* $x_n \in C$ *such that* $x_n \to x$.

Fact 2.13. *[4, Theorem 2.13] For any* $f_0 \in \overline{Conv}\, X$ *and* $f_1 \in \overline{Conv}\, X$,

$$\begin{aligned} \overline{\operatorname{dom} f_\lambda} &= \overline{(1-\lambda)\operatorname{dom} f_0 + \lambda \operatorname{dom} f_1}, \\ \mathsf{int}\operatorname{dom} f_\lambda &= \mathsf{int}\big((1-\lambda)\operatorname{dom} f_0 + \lambda \operatorname{dom} f_1\big). \end{aligned}$$

The result was later refined in [3, Theorem 4.6] as the domain is the convex combination of the domains. As a consequence of Fact 2.13 the proximal average remedies the shortcomings of Example 2.2. This is illustrated on Figure 2.2.

2.3 Strict Convexity and Smoothness

The proximal average inherits nice properties from the two functions it averages.

Notation 2.14. *We say a function has full domain if* $\operatorname{dom} f = X = \mathbb{R}^d$ *is the whole space.*

Fact 2.15. *[4, Theorem 3.2] Let* $f_0 \in \overline{Conv}\, X$, $f_1 \in \overline{Conv}\, X$, $\lambda \in \,]0,1[$, $f_\lambda := \mathcal{P}(f_0, \lambda, f_1)$. *Suppose that* f_0 *or* f_1 *has full domain, and that* f_0^* *or* f_1^* *has full domain. Then the following hold:*

(i.) Both f_λ *and* f_λ^* *have full domain.*

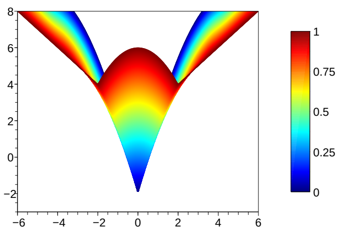
(ii.) If f_0 *or* f_1 *is smooth (i.e. differentiable everywhere), then so is* f_λ.

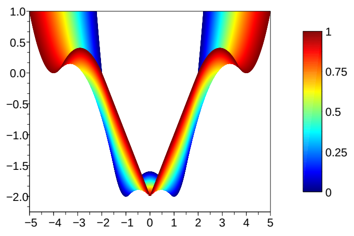
(iii.) If f_0 or f_1 is strictly convex and its Fenchel conjugate has full domain, then f_λ is strictly convex.

The arithmetic average preserves strict convexity if one function is strictly convex; however, if one function is not smooth then the arithmetic average will not be smooth.

Remark 2.16. *It is possible to find the proximal average of nonconvex functions; however, in order to avoid the degenerate cases of $+\infty$ or $-\infty$, each nonconvex function must be bounded below by at least one affine function.*

As a consequence of Section 3.3.2 we can now compute the proximal average of nonconvex functions. Figure 2.16 illustrates the proximal average between convex and nonconvex functions.


(a)


(b)

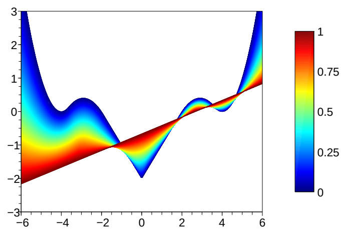

(c)

Figure 2.3: Proximal averages of nonconvex functions

Chapter 3

PLQ Model

Definition 3.1. *Let $\mathcal{PLQ}$ be the set of extended valued functions which are convex, lsc and proper with a piecewise linear subdifferential mapping $\partial f : \mathbb{R} \to 2^{\mathbb{R}}$.*

The set $\mathcal{PLQ}$ contains piecewise linear functions, piecewise quadratic functions and the sum of any such function with an indicator function.

There are several standard operations which arise frequently in convex analysis.

Definition 3.2. *The standard convex operations are (positive) scalar multiplication, addition, conjugation, and regularization (Moreau envelope).*

Throughout we consider only these operations as benchmarks for algorithms and model accuracy. These operations provide a natural set of tools for convex problems, as each operation preserve convexity.

Proposition 3.3. *The following operations are convexity preserving: (positive) scalar multiplication, addition, conjugation, and regularization.*

Proof. (Positive) scalar multiplication and addition clearly preserve convexity. The conjugate is convexity preserving as it is always convex by Proposition 1.17. The Moreau envelope of a convex function f is convex by Formula (1.3). $\square$

3.1 Numerical Methods for Convex Transforms

In convex analysis many transforms are difficult to compute or do not have a closed form.

Example 3.4. *[15, p.46] Let W (the Lambert function) be the inverse function of $x \mapsto xe^x$ defined on the interval $]0, +\infty[$. Then the conjugate of*

$$f(x) = x\log(x) + \frac{x^2}{2} - x$$

is

$$f^*(y) = \frac{1}{2}\left[W(e^y)\right]^2 + W(e^y).$$

As the conjugate of f cannot be written with standard functions (i.e. a combination of trigonometric or polynomial functions) , we use W to represent such functions (it is well known that W does not admit a closed form).

Notation 3.5. *Let $I := \{0, \cdots, n\}$, be a finite set of ordered integers.*

Fast algorithms which efficiently approximate these transforms, depend on discretization of a continuous model. Discretization is the process of converting a continuous model into discrete counterparts.

Example 3.6. *The continuous model for the Fenchel conjugate is*

$$f^*(y) = \sup_{x\in\mathbb{R}}[yx - f(x)], \tag{3.1}$$

and the discretization of this model is

$$f_n^*(y_j) = \max_{i\in I}\{y_j x_i - f(x_i)\}, j \in I. \tag{3.2}$$

The main idea in approximating the continuous model is to evaluate the discrete model over a grid and not on a single point. Many fast algorithms utilize the fact that the domain to approximate the operation and the domain on which the function is defined, is known prior to computation. This prior knowledge allows for efficient computation and can lead to a decrease in time complexity.

Remark 3.7. *Computing grids for fast algorithms requires prior knowledge of not only the domain of f but also the domain of the resulting function. For example, the function $f(x) = -\log(x)$ is defined only on the positive reals.*

The framework is the same for all fast algorithms. First, we discretize our model and compute the transform using an efficient algorithm. Convergence results from Section 3.4 allow us to recover the continuous model.

3.2 Function Approximation

Given a discrete set of points $\{(x_i, f(x_i))\}_{i\in I}$ sampled from a function f (typically the function is not known) such that $x_0 < x_1 < x_2 < \cdots < x_n$, what is the closest approximating function $\tilde{f}$ (from a well defined class of functions) that passes through our discrete set of points? The process used to solve this problem is known as interpolation.

When considering the class of piecewise polynomial functions there are various orders of approximation which correspond to the order of differentiability used in the model (approximation).

Example 3.8. *The following implications describe the orders of approximation:*

$$\begin{aligned}
\textit{Zeroth-order Model} \quad &\Rightarrow \quad f(x_i) = \tilde{f}(x_i), \\
\textit{First-order Model} \quad &\Rightarrow \quad f(x_i) = \tilde{f}(x_i) \textit{ and } f'(x_i) = \tilde{f}'(x_i), \\
\textit{Second-order Model} \quad &\Rightarrow \quad f(x_i) = \tilde{f}(x_i),\ f'(x_i) = \tilde{f}'(x_i),\ \textit{and } f''(x_i) = \tilde{f}''(x_i).
\end{aligned}$$

Remark 3.9. *The word model and approximation can be used interchangeably.*

3.2.1 The Class of PL Functions

The class of piecewise linear (PL) functions accommodates a variety of models of finite order. The following models take a discrete set of points and output a continuous piecewise linear approximation of the function f.

Consider the zeroth-order model

$$\tilde{f}_0(x) := \max_i [f_i + s_i(x - x_i)], \tag{3.3}$$

where $s_i = \frac{f_{i+1} - f_i}{x_{i+1} - x_i}$ and $f_i = f(x_i) = \tilde{f}_0(x_i)$. The model $\tilde{f}$ is convex, lsc, and piecewise linear as Equation (3.3) takes the maximum of a finite set of affine functions. Moreover, $\tilde{f}_0 \geq f$ as it is an inner approximation of the epigraph (see Figure 3.1).

Lemma 3.10. *Let* $f \in \mathcal{PLQ}$ *such that* $\operatorname{dom} f := \{x \ : \ x_0 \leq x \leq x_n\}$. *Then, for* $x \in \operatorname{dom} f$

$$f(x) \leq \max_{i \in \{0, \cdots, n-1\}} \big[f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i) \big]. \tag{3.4}$$

Proof. Let

$$h := \max_{i \in \{0, \cdots, n-1\}} \big[f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i) \big].$$

First we show that $f(x_i) = h(x_i)$ for $i = 0, \ldots, n$. Since f is convex we have

$$\frac{f(x_1) - f(x_0)}{x_1 - x_0} \leq \frac{f(x_2) - f(x_1)}{x_2 - x_1} \leq \cdots \leq \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}. \tag{3.5}$$

Then denote

$$l_i(x) := f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i) \quad \big(\forall \ i \in I\big).$$

Now we have $l_i(x_{i+1}) = f(x_{i+1}) = l_{i+1}(x_i)$. Using Equation (3.5) and

$$\begin{aligned} l_k(x_{i+1}) \leq l_i(x_{i+1}) \quad &\text{if} \quad k \leq i, \\ l_k(x_{i+1} \leq l_i(x_{i+1}) \quad &\text{if} \quad k \geq i, \end{aligned}$$

we obtain

$$h(x_{i+1}) = \max_{0\leq k\leq n} l_k(x_{i+1}) = f(x_{i+1}).$$

By Definition 1.5 all segments which join any two points on a convex function f must lay on or above f. Hence, there exists an index i_0 such that $x \in [x_{i_0}, x_{i_0+1}]$, as $x_i < x_{i+1}$. There also exists λ such that $x = \lambda x_{i_0} + (1-\lambda)x_{i_0+1}$ where $\lambda \in [0,1]$. Then

$$\begin{aligned} f(x_{i_0}) + \frac{f(x_{i_0+1}) - f(x_{i_0})}{x_{i_0+1} - x_{i_0}}(x - x_{i_0}) &= f(x_{i_0}) + \frac{f(x_{i_0+1}) - f(x_{i_0})}{x_{i_0+1} - x_{i_0}}(\lambda x_{i_0} + (1-\lambda)x_{i_{0+1}} - x_{i_0}), \\ &= f(x_{i_0}) + \frac{f(x_{i_0+1}) - f(x_{i_0})}{x_{i_0+1} - x_{i_0}}(1-\lambda)(x_{i_0+1} - x_{i_0}), \\ &= f(x_{i_0}) + (1-\lambda)[f(x_{i_0+1}) - f(x_{i_0})], \\ &= \lambda f(x_{i_0}) + (1-\lambda)f(x_{i_0+1}) \geq f(\lambda x_{i_0} + (1-\lambda)x_{i_0+1}) = f(x). \end{aligned}$$

□

Consider the first-order model

$$\tilde{f}_1(x) := \max_i [f_i + g_i(x - x_i)], \tag{3.6}$$

where $g_i \in \partial f(x_i)$. This model is also convex, continuous, and piecewise linear as Equation (3.6) is

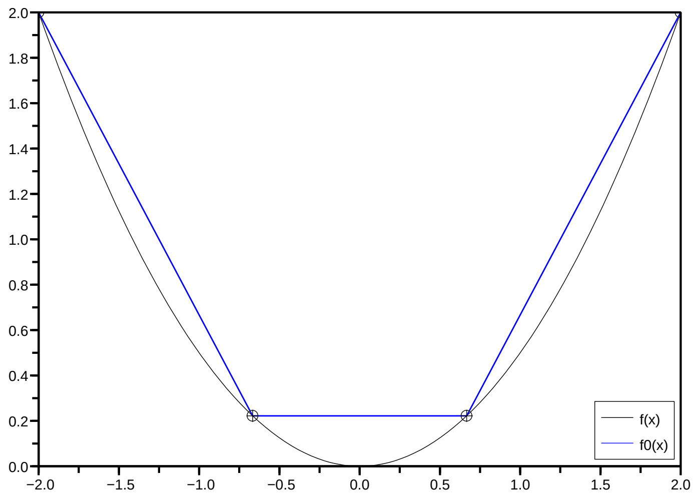


Figure 3.1: The zeroth-order model takes the maximum of a set of affine functions.

taking the maximum of a finite set of linear functions as seen on Figure 3.2. The main difference between $\tilde{f}_0$ and $\tilde{f}_1$ is that this model is an outer approximation of the epigraph with $\tilde{f}_1 \leq f$.

Lemma 3.11. *Let $f \in \mathcal{PLQ}$ and $g_i \in \partial f(x_i)$. Then*

$$f(x) \geq \max_{i \in \{0,\cdots,n\}} \left[f(x_i) + g_i(x - x_i)\right] \quad \big(\forall\ x \in \mathbb{R}\big). \tag{3.7}$$

Proof. From Definition 1.34, we have $f(x) \geq f(x_i) + g_i(x - x_i)$. Therefore the supremum of these minorizing functions will also minorize f. □

Theorem 3.12. *Let $f \in \mathcal{PLQ}$, then $\tilde{f}_1(x) \leq f(x) \leq \tilde{f}_0(x)$.*

Proof. This is a direct consequence of Lemma 3.10, and Lemma 3.11. □

The problem with the zeroth-order and first-order models is that the class of piecewise linear

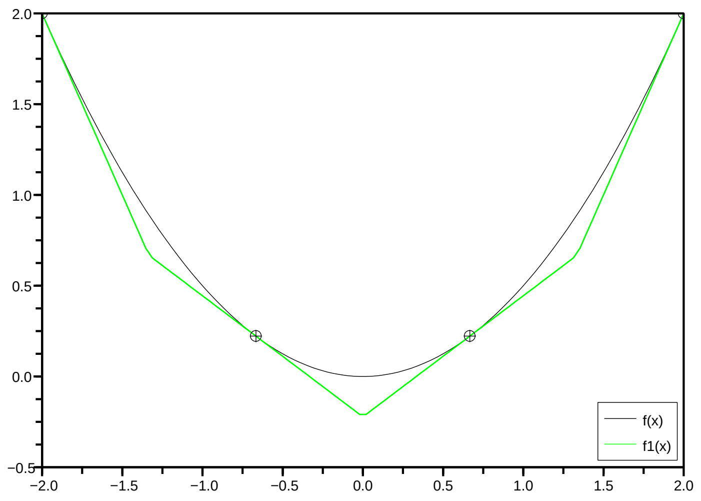


Figure 3.2: The first-order model takes the maximum of a set of subtangents.

(PL) functions is not closed under standard convex operations.

Example 3.13. *The following example is motivated by the proximal average. Let $f_1(x) = x$. Then*

$$f_1^*(y) = I_{\{1\}}(y) \implies \big(f_1^* \Box \frac{1}{2}\|\cdot\|^2\big)(y) = \frac{(y-1)^2}{2}.$$

So a simple combination of standard convex operations results in a function which is quadratic.

Therefore all quadratic pieces cannot be computed exactly, we have to rely on convergence of piecewise linear approximations.

3.2.2 The Class of PLQ Functions

The class of piecewise linear-quadratic (PLQ) functions remedies the shortcomings of Example 3.13. In fact, the class of PLQ functions is closed under all the standard convex operations.

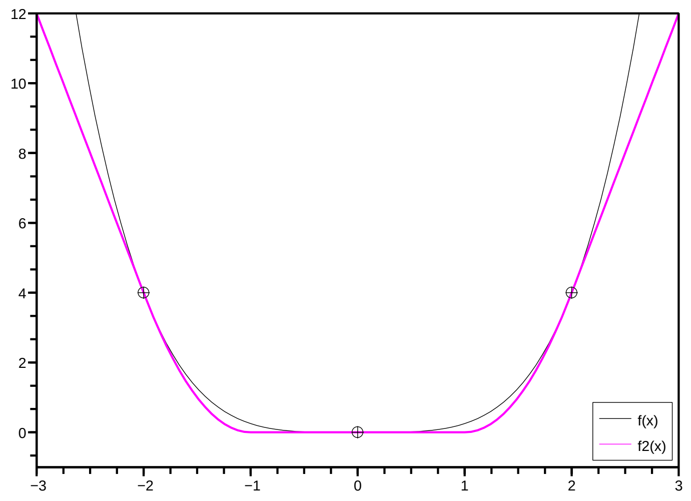


Figure 3.3: Second-order model of the function $f(x) = \frac{x^4}{4}$.

Proposition 3.14. *The class of PLQ functions is closed under (positive) scalar multiplication, addition, conjugation, and regularization.*

Proof. It is clear that the class of PLQ functions is closed under (positive) scalar multiplication and addition.

The conjugate of a PLQ function f is a PLQ function. We notice that the graph of ∂f^* is piecewise linear, and is symmetric (with respect to the line $y = x$) to the graph of ∂f. Therefore f^* is PLQ [21, p.484].

As a consequence of Equation (1.4) and Remark 1.29, the class of PLQ functions is closed under the Moreau envelope operation. □

The PLQ class is robust enough to approximate any convex function using a zeroth-order, first-order or even second-order model. An example of a second order model is seen in Figure 3.3.

Remark 3.15. *It is important that we use a higher-order model because as the order increases, the approximating function deviates less from the actual function. In fact, as the degree of the Taylor series rises, the approximating function is closer to the correct function in a neighborhood of the given discrete set of points (see Subsection 3.4).*

3.3 Algorithms

The objective of this section is to provide a detailed comparison between Fast and PLQ algorithms (see [16–19]). Moreover, we intend to discuss the advantages and drawbacks of each algorithm with respect to the standard convex operations. We will also provide examples and detailed steps which will emphasize the main idea behind each algorithm.

In order to compare algorithms we need a measure of efficiency. Let us consider time-complexity and space-complexity as the main benchmarks for practical use. The time and space complexity of an algorithm can be determined by counting the number of times we look at each point and its corresponding memory allocation. Moreover, we can verify the complexity analysis by increasing the size of input and checking the rate at which the CPU time or memory allocation increases. From here we can categorize an algorithm into certain classes of efficiency.

3.3.1 PLQ Algorithms

Not only does the class of PLQ functions provide closure for the standard convex operations but the class also gives rise to linear-time and linear-space algorithms. Moreover, these algorithms do not require prior knowledge of the domain (unlike fast algorithms). They provide exact solutions when performing standard convex operations on PLQ functions.

An efficient way to store the PLQ function is a PLQ matrix. To illustrate, the unbounded PLQ function

$$f(x) := \begin{cases} a_0x^2 + b_0x + c_0 & \text{if } x \le x_0, \\ a_1x^2 + b_1x + c_1 & \text{if } x_0 < x \le x_1, \\ \vdots & \\ a_{n-1}x^2 + b_{n-1}x + c_{n-1} & \text{if } x_{n-2} < x \le x_{n-1}, \\ a_nx^2 + b_nx + c_n & \text{otherwise.} \end{cases} \tag{3.8}$$

is stored as a matrix

$$\texttt{plqf} := \begin{bmatrix} x_0 & a_0 & b_0 & c_0 \\ x_1 & a_1 & b_1 & c_1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{n-1} & a_{n-1} & b_{n-1} & c_{n-1} \\ +\infty & a_n & b_n & c_n \end{bmatrix}. \tag{3.9}$$

The bounded-domain version of $f(x)$ is stored as

$$\texttt{plqf} := \begin{bmatrix} x_0 & 0 & 0 & +\infty \\ x_1 & a_1 & b_1 & c_1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{n-1} & a_{n-1} & b_{n-1} & c_{n-1} \\ +\infty & 0 & 0 & +\infty \end{bmatrix}.$$

Half-bounded functions are stored similarly.

Remark 3.16. *The indicator function $f(x) := I_{\{x_0\}}(x)$ of a single point x_0 is a special case which is stored as the row vector **plqf** $:= [x_0 \quad 0 \quad 0 \quad +\infty]$.*

Definition 3.17. *Let $\mathcal{PLQE}$ denote the set of all lsc extended-valued functions which are PLQ and* *continuous on the interior of their domains.*

Remark 3.18. *The class of functions in $\mathcal{PLQE}$ is just the set of functions in $\mathcal{PLQ}$ extended to nonconvex functions.*

A PLQ matrix can only store functions in $\mathcal{PLQE}$ due to the following limitations.

Example 3.19. *The usc function*

$$f(x) = \begin{cases} x^2 & \text{if } x < 0, \\ x^2 + 1 & \text{if } x \geq 0, \end{cases}$$

cannot be stored as a PLQ matrix as

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ +\infty & 1 & 0 & 1 \end{bmatrix} \Leftrightarrow \begin{cases} x^2 & \text{if } x \leq 0, \\ x^2 + 1 & \text{if } x > 0. \end{cases}$$

This limitation is due to the fact that the data structure is not robust enough to represent all combinations of inequalities and strict inequalities.

Similarly, functions which are neither lsc nor usc cannot be stored as a PLQ matrix.

Example 3.20. *A discontinuous function*

$$f(x) = \begin{cases} x^2 & \text{if } x < 0, \\ x^2 + 1 & \text{if } x > 0, \end{cases}$$

does not fit into our model as the quadratic, x^2 for $x < 0$ is defined by a strict inequality.

Another limitation of the model is that we cannot explicitly store vertical lines.

Example 3.21. *The PLQ operator*

$$f(x) = \begin{cases} \{-1\} & \text{if } x < 0, \\ [-1, 1] & \text{if } x = 0, \\ \{1\} & \text{if } x > 0, \end{cases}$$

cannot be represented by PLQ matrix due to the vertical line at $x = 0$.

Remark 3.22. *The limitations of the PLQ matrix in Example 3.19 and 3.20 could be remedied by adding a point between each interval. Also, in Example 3.21 we could store vertical lines implicitly: store all nonvertical lines, and complete the graph by continuity/maximality. We will not consider such models in the present thesis.*

Next we present a linear-time algorithm for each standard convex operation in Definition 3.2, where each algorithm only considers functions from the set $\mathcal{PLQ}$.

(Positive) Scalar Multiplication (PLQ Algorithm)

Scalar multiplication of a PLQ function amounts to multiplying each piece of a PLQ function by a scalar (see Figure 3.4). As multiplying each piece takes constant time, multiplying n pieces will take n steps; hence the algorithm runs in linear-time.

Let $\lambda \in [0, +\infty[$. Then

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

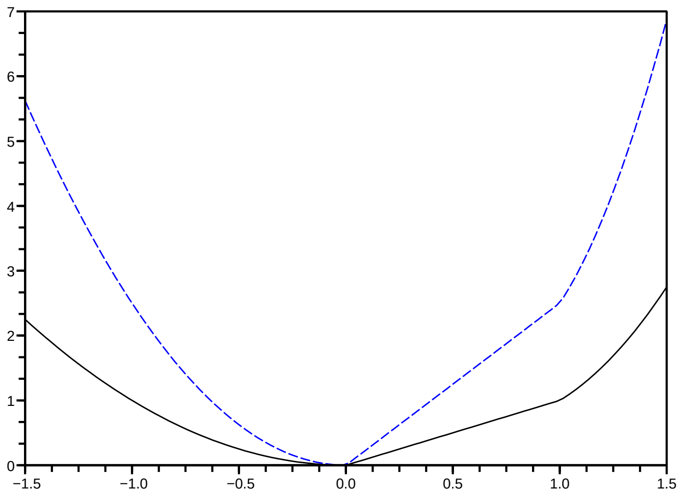


Figure 3.4: The dashed line is the function resulting from multiplying a PLQ function by a scalar.

Notation 3.23. *The symbol $\otimes$ denotes PLQ multiplication, and is not the usual matrix multiplication.*

Addition (PLQ Algorithm)

Now we will present an algorithm to compute the sum of two PLQ functions (which could be extended to the sum of a finite number of PLQ functions).

Definition 3.24. *The partition points are the x_i of the first column in the PLQ matrix (3.9). Moreover, the set of partition points is the whole column vector.*

Algorithm 3.25. *The algorithm sorts all the partition points on the x-axis (visualize an x-y plane) where along each interval $[x_i, x_{i+1}]$ we compute the sum of two quadratic (or linear) pieces which are active in $[x_i, x_{i+1}]$.*

The addition of two PLQ functions is not trivial unless both functions have identical sets of partition points as in Example 3.26.

Example 3.26.

$$\begin{bmatrix} x_0 & a_0 & b_0 & c_0 \\ \vdots & \vdots & \vdots & \vdots \\ x_{n-1} & a_{n-1} & b_{n-1} & c_{n-1} \\ +\infty & a_n & b_n & c_n \end{bmatrix} \oplus \begin{bmatrix} x_0 & d_0 & e_0 & f_0 \\ \vdots & \vdots & \vdots & \vdots \\ x_{n-1} & d_{n-1} & e_{n-1} & f_{n-1} \\ +\infty & d_n & e_n & f_n \end{bmatrix} = \begin{bmatrix} x_0 & a_0+d_0 & b_0+e_0 & c_0+f_0 \\ \vdots & \vdots & \vdots & \vdots \\ x_{n-1} & a_{n-1}+d_{n-1} & b_{n-1}+e_{n-1} & c_{n-1}+f_{n-1} \\ +\infty & a_n+d_n & b_n+e_n & c_n+f_n \end{bmatrix}.$$

Remark 3.27. *We use the symbol $\oplus$ to denote the addition of two PLQ functions, which is not the usual matrix addition.*

In the extreme case, if both PLQ matrices (or functions) have disjoint sets of partitioning points, the matrix resulting from the addition operation will have the union of both sets as partition points (see Example 3.28).

Example 3.28. *The addition of the following PLQ matrices (or PLQ functions) is shown in Figure 3.5.*

$$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ +\infty & 1 & 0 & 0 \end{bmatrix} \oplus \begin{bmatrix} 0 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ +\infty & 1 & 0 & -2 \end{bmatrix} = \begin{bmatrix} -1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 2 & 1 & 1 & 0 \\ +\infty & 2 & 0 & -2 \end{bmatrix}.$$

So the general case requires us to keep track of each partitioned interval and determine which quadratic (or linear) pieces are added together.

As we are adding at most $n+m$ quadratic (or linear) pieces together the running time is

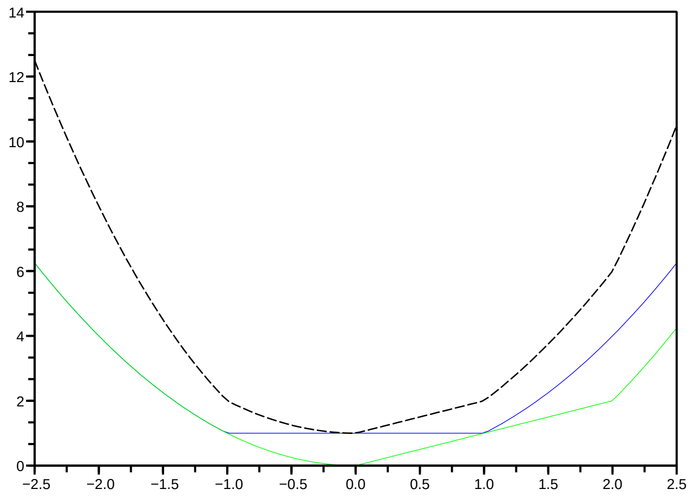


Figure 3.5: The addition of two PLQ matrices (or functions) with disjoint sets of partitioning points results in the ”dashed” function.

$O(n+m)$, where n, (resp. m) is the size of the sets of partitioning points for the first (resp. second) function.

Conjugation (PLQ Algorithm)

Next we present an algorithm which finds the conjugate of a function $f(x) \in \mathcal{PLQ}$. Visualizing $f(x)$ as Equation (3.8) we consider a pair of quadratic functions from $f(x)$:

$$\bar{f}(x) := \begin{cases} \varphi_i(x) := a_i x^2 + b_i x + c_i & \text{if } x_{i-1} < x \leq x_i, \\ \varphi_{i+1}(x) := a_{i+1} x^2 + b_{i+1} x + c_{i+1} & \text{if } x_i < x \leq x_{i+1}. \end{cases}$$

It is convex if and only if

$$\begin{cases} \varphi_i(x_i) = \varphi_{i+1}(x_i) & \text{(continuity criterion)}, \\ \varphi_i'(x_i) \leq \varphi_{i+1}'(x_i) & \text{(increasing "slopes" criterion)}, \\ a_i, a_{i+1} \geq 0 & \text{(convexity criterion)}. \end{cases}$$

From these assumptions, we can compute the conjugate directly as

$$\bar{f}^*(y) = \begin{cases} \varphi_i^*(y) & \text{if } \varphi_i'(x_{i-1}) < y \leq \varphi_i'(x_i), \\ \bar{y}(y - \varphi_i'(x_i)) + \varphi_i^*(\varphi_i'(x_i)) & \text{if } \varphi_i'(x_i) \leq y \leq \varphi_{i+1}'(x_i), \\ \varphi_{i+1}^*(y) & \text{if } \varphi_{i+1}'(x_i) < y \leq \varphi_{i+1}'(x_{i+1}), \end{cases} \tag{3.10}$$

where

$$\bar{y} := \frac{\varphi_{i+1}'(x_i)x_i - \varphi_{i+1}(x_i) - \varphi_i'(x_i)x_i + \varphi_i(x_i))}{\varphi_{i+1}'(x_i) - \varphi_i'(x_i)}.$$

The middle case (which only arises when the function $\bar{f}(x)$ is nonsmooth at x_i) is an affine function bridging the graph of $\varphi_i^*(y)$ with $\varphi_{i+1}^*(y)$. Computing $\bar{f}^*(y)$ can be done in linear-time and space since the computation of $\varphi_i^*(y)$ and $\varphi_{i+1}^*(y)$ can be performed directly (see Example 1.18).

Remark 3.29. *The conjugate in Equation 3.10 was computed directly using the Maple package "symbolic convex analysis toolkit" (SCAT) [9]. As $\bar{f}(x)$ is convex we can compute the conjugate $\sup_y \langle x, y\rangle - \bar{f}(x)$ by solving for the critical point of the inner function $\langle y, x\rangle - \bar{f}(x)$ and substituting it back into the conjugate equation.*

Algorithm 3.30. *The general strategy is to iteratively take pairs of quadratic (or linear) functions from $f(x)$ (from left to right, consecutively) and apply the explicit closed form of the conjugate operation from Equation (3.10) to $\bar{f}(x)$.*

One drawback of Equation (3.10) is that we assumed $\bar{f}(x)$ is convex. Therefore the PLQ conjugate and Moreau envelope algorithms are restricted to convex functions. The next section removes the convexity assumption.

3.3.2 Extending the PLQ Algorithms

We recall the fact that the PLQ conjugate, and Moreau envelope algorithms are restricted to convex functions. From Formula (1.4), and Remark 1.29 we need only to extend the conjugate algorithm to nonconvex functions. In fact, the relationship between the Fenchel conjugate and the convex hull provides some insight into conjugating a nonconvex PLQ function.

Remark 3.31. *The class of functions in $\mathcal{PLQE}$ will be use the same data structure as in the set $\mathcal{PLQ}$.*

Fact 3.32. *[11, p.44] For any function f minorized by at least one affine function, the biconjugate of f is the closed convex hull: $f^{**} = \overline{\operatorname{co}}\, f$.*

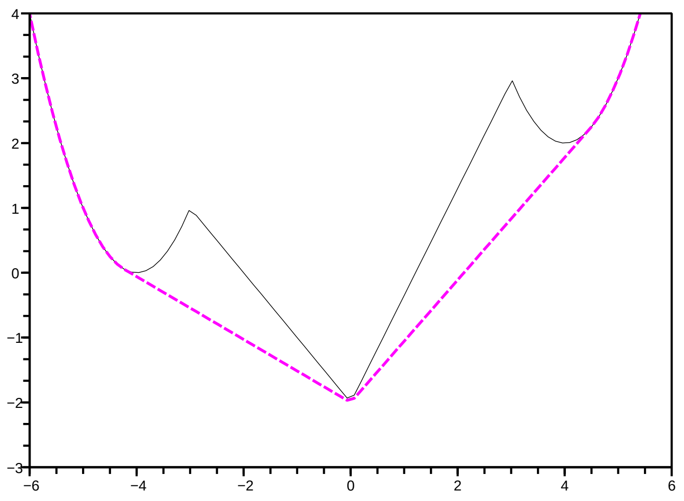


Figure 3.6: The biconjugate (f^{**}) yields the closed convex hull.

As the Fenchel conjugate and convex hull depend on the pointwise supremum of a family of affine functions, the Fenchel conjugate is exactly the conjugate of the convex hull (see Facts 3.32 and 1.47).

Theorem 3.33. *For any proper function f, we have $f^* = (\overline{\text{co}}\ f)^*$ and $f^{***} = f^*$.*

Proof. The function f^* is lsc and convex so we have $f^{***} = f^*$. The conjugate of Fact 3.32, yields $f^{***} = (\overline{\text{co}}\, f)^*$. Since $f^{***} = f^*$ we have $f^* = (\overline{\text{co}}\ f)^*$. ☐

So regardless whether we take the conjugate of the convex hull or the conjugate itself, the conjugate function will be the same. Therefore, the PLQ conjugate algorithm can be extended to nonconvex functions by first taking the convex hull.

Convex Hull

Now we present an algorithm which finds the convex hull of a function $f(x)$ in $\mathcal{PLQE}$. Let $f(x)$ be defined as Equation (3.8). Then iteratively take pairs of PLQ functions

$$\bar{f}(x) := \begin{cases} \varphi_i(x) = a_i x^2 + b_i x + c_i & \text{if } x_{i-1} < x \leq x_i, \\ \varphi_{i+1}(x) = a_{i+1} x^2 + b_{i+1} x + c_{i+1} & \text{if } x_i < x \leq x_{i+1}, \end{cases}$$

from $f(x)$ (i.e. for $i = 1, \ldots, n$), and determine if they are nonconvex. The result below is important, but we omit its proof as it is trivial.

Theorem 3.34. *The PLQ function $\bar{f}(x)$ is nonconvex if at least one of the following conditions hold.*

I. $a_i < 0$ *(Concavity Criterion),*

II. $a_{i+1} < 0$ *(Concavity Criterion),*

III. $\varphi_i(x_i) < \varphi_{i+1}(x_i)$ *(Discontinuity Criterion),*

IV. $\varphi_i(x_i) > \varphi_{i+1}(x_i)$ *(Discontinuity Criterion),*

V. $\varphi_i'(x_i) > \varphi_{i+1}'(x_i)$ *(Decreasing "Slope" Criterion).*

Remark 3.35. *Although we have restricted our framework to the class of functions which are continuous (see the set $\mathcal{PLQE}$). We still consider cases III. and IV. as a more general setting for the PLQ convex hull algorithm.*

First, we address several special cases.

Table 3.1: The following special cases arise when taking the convex hull of $f(x)$.

Case 1.	$a_0 < 0$ or $a_n < 0$	co $f(x) = -\infty$
Case 2.	$a_0 \geq 0$ and $a_n \geq 0$	
Case 2.a.	$a_0 \geq 0$ and $a_n > 0$	co $f(x)$ is PLQ
Case 2.b.	$a_0 > 0$ and $a_n \geq 0$	co $f(x)$ is PLQ
Case 2.c.	$a_0 = 0$ and $a_n > 0$	co $f(x)$ is PLQ
Case 2.d.	$a_0 > 0$ and $a_n = 0$	co $f(x)$ is PLQ
Case 2.e	$a_0 = a_n = 0$ and $b_0 > b_n$	co $f(x) = -\infty$
Case 2.f	$a_0 = a_n = 0$ and $b_0 \leq b_n$	co $f(x)$ is PLQ

Remark 3.36. *The above special cases arise from the fact that the convex hull requires at least one supporting affine function which is finite. If we cannot find at least one, then the convex hull is exactly minus infinity.*

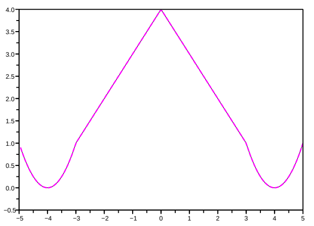
Theorem 3.37. *Let $f(x) \in \mathcal{PLQE}$, and be defined as Equation (3.8). Then the convex hull has several special cases*

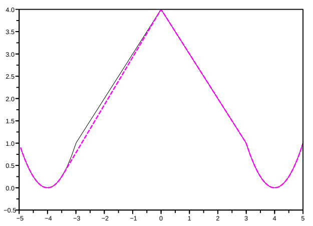
$$\operatorname{co} f(x) = \begin{cases} -\infty & \text{if } a_0 < 0, \\ -\infty & \text{if } a_n < 0, \\ -\infty & \text{if } a_0 = a_n = 0 \text{ and } \varphi_0'(x_0) > \varphi_n'(x_{n-1}), \\ PLQ & \text{otherwise.} \end{cases}$$

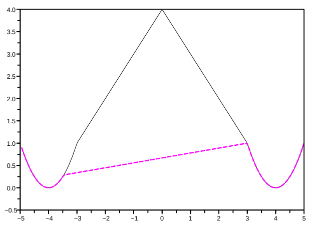
Proof. Table 3.1 considers all possible cases concerning the first and last quadratics of f, which determine if $f(x)$ can be supported by an affine function. □

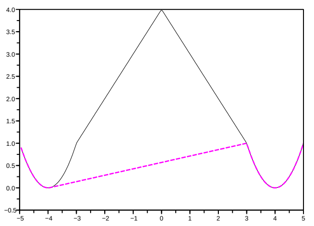
Algorithm 3.38. *If $f(x)$ falls under one of the special cases in Theorem 3.37 we stop with co $f(x) = -\infty$. As the unbounded concave functions are noted in Theorem 3.37 we address the case of bounded concave functions.*

If $f(x)$ contains any bounded concave functions (i.e. satisfying I or II in Theorem 3.34 with x_{i-1}, and x_{i+1} finite) then replace each concave part by an affine function interpolating the endpoints (See Equation (3.11) in Solution 3.43). Next, iteratively grab pairs of functions $\bar{f}(x)$ from $f(x)$ and determine if they are "quadratic-quadratic", "quadratic-linear", "linear-quadratic" or "linear-linear". Then proceed to the appropriate Problem (resp. 3.39, 3.41, 3.40, or 3.43). If the pair turns out to be nonconvex by one of the criteria in Theorem 3.34, then we backtrack by decrementing the index i and repeating the above process. The algorithm is complete when there no longer exists any nonconvex pairs by Theorem 3.34.


(a)


(b)


(c)


(d)

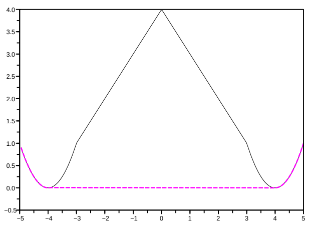

(e)

Figure 3.7: The above sequence of images, demonstrate the required steps in computing the convex hull of a PLQ function. We see from Figure (c) that it is necessary to back-track at most once per iteration to ensure a convex function. Indeed a single pass may not produce a convex function, and as a result the algorithm looks at each partition point at most twice. Hence, the algorithm takes at most $2n$ steps and runs in linear time.

Problem 3.39 (quadratic-quadratic)**.** *Given two quadratic (or linear) functions φ_i and φ_{i+1} with domain $[x_{i-1}, x_{i+1}]$, partitioned at x_i, solve the following system for x_α and x_β.*

$$\begin{aligned}
\varphi_i'(x_\alpha) &= \varphi_{i+1}'(x_\beta) \\
\varphi_i'(x_\alpha)(x - x_\alpha) + \varphi_{i+1}(x_\alpha) &= \varphi_{i+1}'(x_\beta)(x - x_\beta) + \varphi_{i+1}(x_\beta) \quad \forall\ x \in \mathbb{R} \\
x_{i-1} \leq \quad x_\alpha &\leq x_i \\
x_i \leq \quad x_\beta &\leq x_{i+1}.
\end{aligned}$$

If there exists a solution, then the convex hull is

$$\operatorname{co} \bar{f}(x) = \begin{cases} \varphi_i(x) & \text{if } x_{i-1} \leq x < x_\alpha, \\ \varphi_i'(x_\alpha)(x - x_\beta) + \varphi_{i+1}(x_\beta) & \text{if } x_\alpha \leq x \leq x_\beta, \\ \varphi_{i+1}(x) & \text{if } x_\beta < x < x_{i+1}. \end{cases}$$

If the above system of equations has no solution, we solve Problem 3.41.

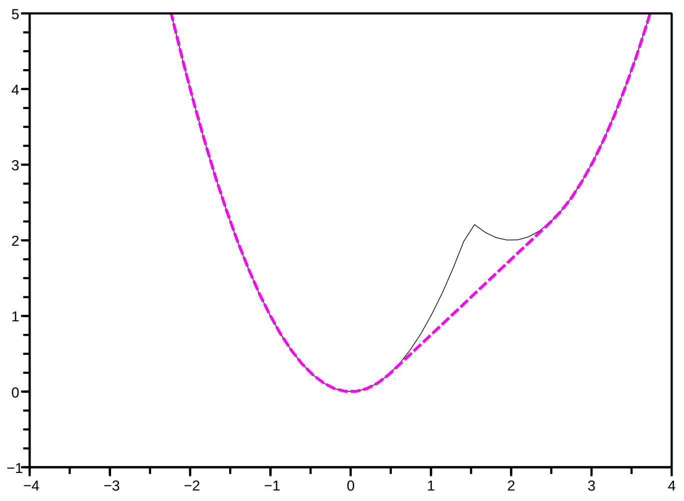


Figure 3.8: The "dashed" function is an instance of $\operatorname{co}\bar{f}(x)$, and the partition points x_α and x_β are where the affine function $\varphi_i'(x_\alpha)(x - x_\beta) + \varphi_{i+1}(x_\beta)$ bridges the gap between the left quadratic $\varphi_i(x)$, and the right quadratic $\varphi_{i+1}(x)$. Note that we do not need to consider the bounded and unbounded cases as we can always find an affine function minorizing $\bar{f}(x)$ and supporting both quadratic pieces.

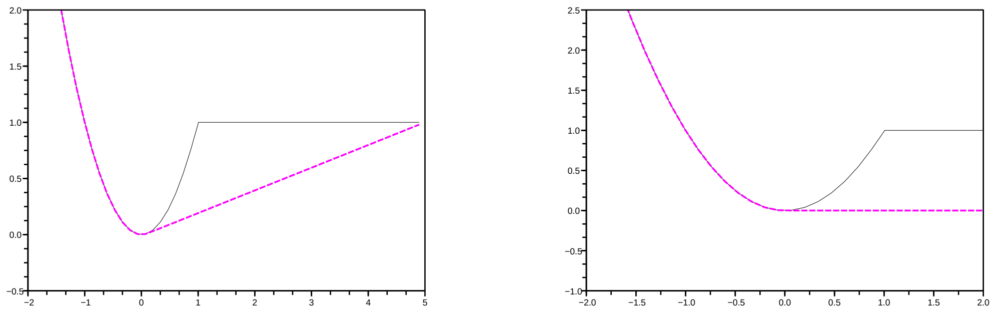
Problem 3.40 (quadratic-linear)**.** *Given two quadratic (or linear) functions φ_i and φ_{i+1} joined at x_i with domain $[x_{i-1}, x_{i+1}]$, find an x_α such that*

$$\begin{aligned} \varphi_i'(x_\alpha)(x - x_\alpha) + \varphi_i(x_\alpha) &= \varphi_i'(x_\alpha)(x - x_{i+1}) + \varphi_i(x_{i+1}), \quad \forall\, x \in \mathbb{R} \\ x_{i-1} \leq \quad x_\alpha &\leq x_i. \end{aligned}$$

If a solution exists then the convex hull is

$$\operatorname{co}\bar{f}(x) = \begin{cases} \varphi_i(x) & \textit{if } x_{i-1} \le x < x_\alpha, \\ \varphi_i'(x_\alpha)(x - x_{i+1}) + \varphi_i(x_{i+1}) & \textit{if } x_\alpha \le x \le x_{i+1}. \end{cases}$$

If the above system of equations has no solution, we solve Problem 3.40.



(a) (b)

Figure 3.9: The "dashed" function is an instanceco $\bar{f}(x)$ and the partition point x_α is where the affine functions $\varphi_{i+1}'(x_\alpha)(x - x_{i-1}) + \varphi_i(x_{i-1})$ on the right joins the quadratic $\varphi_{i+1}(x)$ on the left. The two graphs (a) and (b) represent the difference between a bounded and unbounded convex hull. If x_{i+1} is finite then our graph will look like (a) but if $x_{i+1} = +\infty$ then it will look like (b).

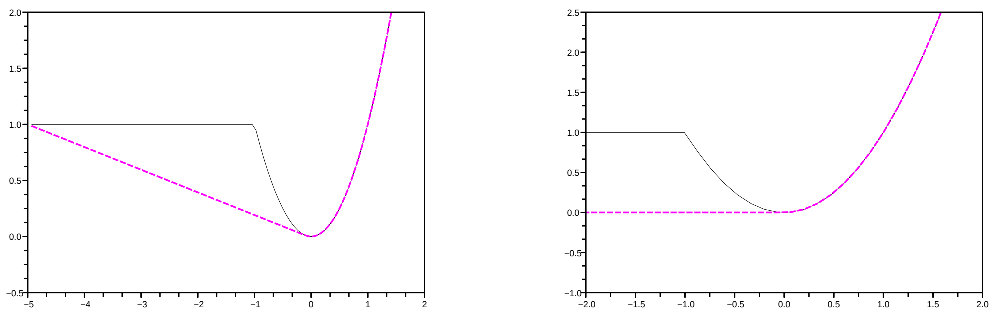
Problem 3.41 (linear-quadratic). *Given two quadratic (or linear) functions φ_i and φ_{i+1} defined on the interval $[x_{i-1}, x_{i+1}]$ and joined at x_i, find an x_β such that*

$$\begin{aligned} \varphi_i'(x_\beta)(x - x_{i-1}) + \varphi_i(x_{i-1}) &= \varphi_{i+1}'(x_\beta)(x - x_\beta) + \varphi_{i+1}(x_\beta) \quad \forall\, x \in \mathbb{R}, \\ x_i \le \quad x_\beta & \le x_{i+1}. \end{aligned}$$

If we find a solution to the above equations the convex hull is

$$\operatorname{co}\bar{f}(x) = \begin{cases} \varphi'_{i+1}(x_\beta)(x - x_{i-1}) + \varphi_i(x_{i-1}) & \text{if } x_{i-1} \le x \le x_\beta, \\ \varphi_{i+1}(x) & \text{if } x_\beta < x \le x_{i+1}. \end{cases}$$

If the above system of equations has no solution, we solve Problem 3.43.



(a) (b)

Figure 3.10: The "dashed" function is an instance $\operatorname{co}\bar{f}(x)$, and the partition point at x_β is where the affine function $\varphi'_i(x_\beta)(x - x_{i+1}) + \varphi_{i+1}(x_{i+1})$ on the left joins the quadratic $\varphi_i(x)$ on the right. The two graphs (a) and (b) represent the difference between a bounded and unbounded convex hull. If x_{i-1} is finite then our graph will look like (a) but if $x_{i-1} = -\infty$ then it will look like (b).

Notation 3.42. *Assume we have a bounded PLQ function $\bar{f}(x)$ discontinuous at x_i. Then* $\operatorname{cont}\bar{f}(x)$ *returns the continuous version of $\bar{f}(x)$. The function $\bar{f}(x)$ is discontinuous by case III or IV of Theorem 3.34. These cases are addressed below.*

Problem 3.43 (linear-linear)**.** *If $\bar{f}(x)$ satisfies III of Theorem 3.34, then the function* $\operatorname{cont}\bar{f}(x)$

is the line interpolating $(x_i, \varphi_i(x_i))$ *and* $(x_{i+1}, \varphi_{i+1}(x_{i+1}))$. *Hence,*

$$\operatorname{cont} \bar{f}(x) = \begin{cases} \varphi_i(x) & \text{if } x_{i-1} \le x \le x_i, \\ \frac{\varphi_{i+1}(x_{i+1}) - \varphi_i(x_i)}{x_{i+1} - x_i}(x - x_i) + \varphi_i(x_i) & \text{if } x_i < x \le x_{i+1}. \end{cases}$$

If $\bar{f}(x)$ *satisfies IV of Theorem 3.34, then the function* $\operatorname{cont} \bar{f}(x)$ *is the line interpolating* $(x_{i-1}, \varphi_i(x_{i-1}))$ *and* $(x_i, \varphi_{i+1}(x_i))$. *Hence,*

$$\operatorname{cont} \bar{f}(x) = \begin{cases} \frac{\varphi_{i+1}(x_i) - \varphi_i(x_{i-1})}{x_{i+1} - x_i}(x - x_i) + \varphi_i(x_i) & \text{if } x_{i-1} < x \le x_i, \\ \varphi_{i+1}(x) & \text{if } x_i < x \le x_{i+1}. \end{cases}$$

If $\bar{f}(x)$ *satisfies V of Theorem 3.34, then the function* $\operatorname{co} \bar{f}(x)$ *is the line interpolating* $(x_{i-1}, \varphi(x_{i-1}))$ *and* $(x_{i+1}, \varphi(x_{i+1}))$. *Hence,*

$$\operatorname{co} \bar{f}(x) = \frac{\varphi_{i+1}(x_{i+1}) - \varphi_i(x_{i-1})}{x_{i+1} - x_{i-1}}(x - x_{i-1}) + \varphi_i(x_{i-1}) \quad \text{if } x_{i-1} \le x \le x_{i+1}. \tag{3.11}$$

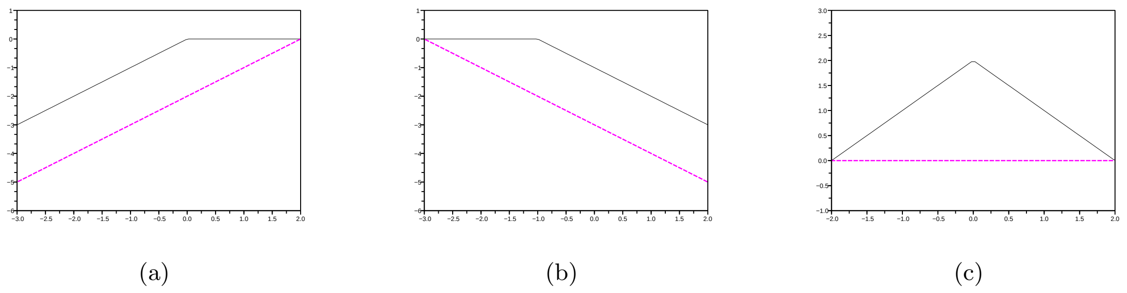


Figure 3.11: The "dashed" curves represent different instances of the function $\operatorname{co} \bar{f}(x)$. Figure (a) is the convex hull when the function $\bar{f}(x)$ is bounded from the right. Figure (b) is the convex hull when $\bar{f}(x)$ is bounded from the left, and Figure (c) is the convex hull when $\bar{f}(x)$ is bounded from both sides.

3.3.3 Fast Algorithms for Convex Transforms

Let us recall fast algorithms, and compare them with PLQ algorithms using the same standard convex operations as in Definition 3.2.

(Positive) Scalar Multiplication (Fast Algorithm)

Multiplying a function by a (positive) scalar, amounts to evaluating a function on a grid and multiplying each sample value by the scalar value.

Notation 3.44. *The value $X[i]$ is the i^{th} term in the grid X. For example if $X := (5, 8, 9, 2)$ then $X[1] = 5$. Moreover, if we are given a function $f(x) := x^2$ for $x \in \mathbb{R}$, then $f(X[1]) = 25$, and $f(X) = (X)^2 = (25, 64, 81, 4)$.*

Remark 3.45. *In the following Fast algorithms, we do not require a uniform grid and use it for the sake of simplicity.*

Algorithm 3.46. *Let $X := \{a + i\frac{b-a}{n} \ : \ i = 1, 2, \cdots, n\}$ be a discrete grid on the interval $[a, b]$ with n points. Then scalar multiplication amounts to multiplying each value $f(X[i])$ by a scalar λ.*

Fast scalar multiplication runs in linear-time and is an equivalent algorithm to PLQ scalar multiplication in the sense that they both use the same set of steps.

Addition (Fast Algorithm)

Adding two continuous functions numerically calls for the evaluation of each function, and computing the sum of their values. The addition of two grids is not necessarily straightforward. In order to add two functions evaluated on different grids, we need to model our discrete set of points and compute the sum along interpolated values.

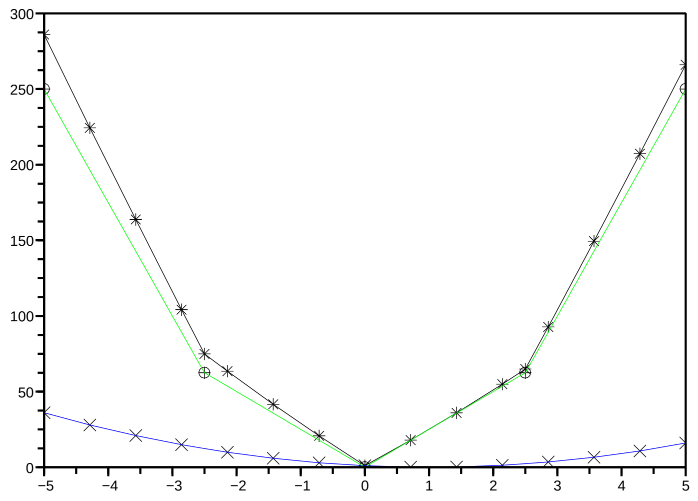


Figure 3.12: The limitations of discrete addition.

The addition of two functions on disjoint grids is shown in Figure 3.12. The function marked by "crosses" is $f_1(X_1) = (X_1)^2 - 2(X_1) + 1$, and the function marked by "circle-crosses" is $f_2(X_2) = 10(X_2)^2$. They are evaluated on the grids $X_2 := \{-5 + i\frac{5+5}{5} \ : \ i = 1, 2, \cdots, 5\}$, and $X_1 := \{-5 + i\frac{5+5}{15} \ : \ i = 1, 2, \cdots, 15\}$ respectively. The addition of $f_1(X_1)$ with $f_2(X_2)$ is the curve marked by "stars".

When considering needle functions, any grid points which do not match up, sum to positive infinity. The extreme case is when the two grids are disjoint; then the resulting function is exactly positive infinity.

Conjugation (Fast Algorithm)

One of the first fast algorithms used to approximate the conjugate was the Fast Legendre Transform (FLT) algorithm [8, 13]. It was introduced to numerically approximate the conjugate of a function with a log-linear worst-case complexity. A more efficient algorithm, conveniently named the Linear-time Legendre transform (LLT) improved the complexity to linear-time [14].

The Linear-Time Legendre Transform (LLT) is a fast algorithm which takes as input the sets X, Y, S where $Y[i]$ is an approximation of $f(X[i])$. The input is a set of points and slopes in $\mathbb{R}$. The output is a set Z_m where $Z[j]$ is an approximation of $f^*(S[j])$. The LLT algorithm can be considered a black box which computes the conjugate at various points in space. Computing the approximation $f^*(S[j])$ involves two different steps depending on the convexity of f.

Step 1. Assuming f is nonconvex and $X[i]$ is increasing, we use the Beneath-Beyond algorithm (see [2]) to compute the convex hull of the planar set $(X[i], Y[i])$ in linear-time. After obtaining the convex hull, we discard all points inside the convex hull and proceed to Step 2.

Step 2. Assuming f is convex (otherwise complete Step 1), we compute the slopes

$$C := \left(c_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \quad (\forall\ i \in I)\right).$$

Finding a solution for Equation (3.2) is equivalent to finding the supremum of all affine functions that support f. As we are given a set of slopes to approximate the Fenchel conjugate, we need only to find the support point (or where the function is maximized). Using the fact that f is convex and therefore has increasing slopes, we only need to find index i such that

$$C[i] \leq S[j] \leq C[i+1],$$

as $c_1 \leq c_2 \leq \cdots \leq c_n$. Thus, $X[i]$ is the supporting point and maximizes $i \mapsto S[j]X[i] - Y[i]$ for slope $S[j]$. As a result we have an approximation of the conjugate

$$Z[j] = S[j]X[i] - Y[i] \approx f^*(S(j)),$$

at $S(j)$. For further information see [15, p.27].

Remark 3.47. *The LLT algorithm utilizes the fact that the supremum of Formula (3.1) is attained if f is convex and $x \in \operatorname{dom} f$ and $s \in \partial f(x)$. See Proposition 1.25.*

3.4 Convergence

We recall the definition of pointwise convergence.

Definition 3.48. *Suppose $\{f_n\}$ is a sequence of functions with common domain. If*

$$\lim_{n\to\infty} f_n(x) = f(x)$$

holds for all $x \in \operatorname{dom} f$ then the sequence $\{f_n\}$ converges pointwise to f.

3.4.1 Fast Algorithms for Convex Transforms (Convergence)

All fast algorithms depend on discretizing a continuous model in order to approximate a convex operation. Since fast algorithms output a discrete set of data points, all fast algorithms implicitly rely on a zeroth order model. In order for these computational algorithms to converge, we need to either increase the number of points or enlarge the domain. For example, when we consider the discrete Legendre transform in Equation (3.2), we have the following convergence results [19, p.3]:

Notation 3.49. *For the sake of simplicity we consider equidistant points:*

$$X := (a + i\frac{b-a}{n}\ :\ i = 1, 2, \cdots, n).$$

We define the discrete approximation of f as f_X with linear interpolation defining f_X at points not in X.

Notation 3.50. *The function f_n is the function f evaluated on a grid $X := \{a + i\frac{(b-a)}{n} \ : \ i = 1, 2, \ldots, n\}$ where the n corresponds to the size of the grid X. Moreover, f_n is the $n-th$ term in a sequence of functions (f_n).*

Fact 3.51. *Assume $f : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ is proper.*

Convergence on a bounded domain *(see [8, 13, 14]).*

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

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

Convergence on unbounded domains *(see [10, 14])*

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

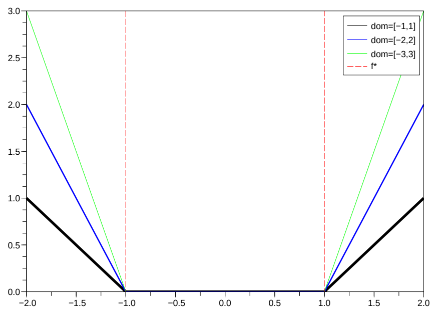
$$\partial f^*(s) \cap [-a, a] \neq \emptyset \Leftrightarrow f^*_{[-a,a]}(s) = f^*(s).$$

3.4.2 PLQ Algorithms (Convergence)

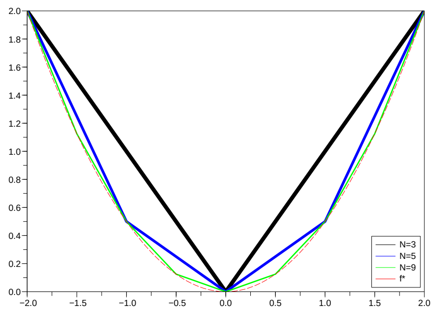
Let us recall the fact that PLQ algorithms provide exact solutions within the class of PLQ functions, and that functions outside this class must be modeled via PL or PLQ approximation.

Theorem 3.52. *Assume f is proper, modeled by a PL approximation $\tilde{f}_n$, and satisfies condition (i) or (ii) of Fact 3.51. Then $\tilde{f}_n^*$ converges pointwise to f^*.*

Proof. Assume $\tilde{f}_n$ is modeled via PL approximation, then it falls under the same framework as the LLT algorithm. As consequence of Fact 3.51, $\tilde{f}_n$ converges pointwise to f^* on both bounded and unbounded domain. $\square$



(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 spacing: the discrete conjugate of the function $f(x) := x^2/2$ converges to $f^* = f$.

Figure 3.13: Convergence of the discrete Legendre transform.

Chapter 4

Finite Convex Integration

This chapter provides additional examples and details for [12]. Throughout we will also describe and illustrate the relationship between the two papers, [12] and [5]. As notations differ, our purpose is to explain both styles in an effort to provide additional perspectives and to aid understanding.

4.1 Introduction

Notation 4.1. *Define* $I := \{0, 1, \ldots, n\}$.

We are concerned with the following problem.

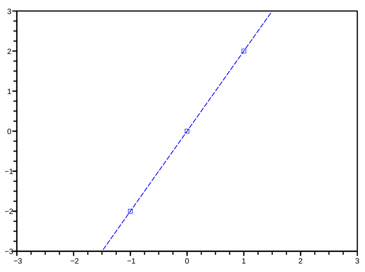
Problem 4.2. *Given a finite family* $\{(x_i, x_i^*)\}_{i \in I} \subset X \times X^*$. *find a lsc convex function*

$$f : X \to \mathbb{R} \text{ such that } x_i^* \in \partial f(x_i) \ \forall i \in I$$

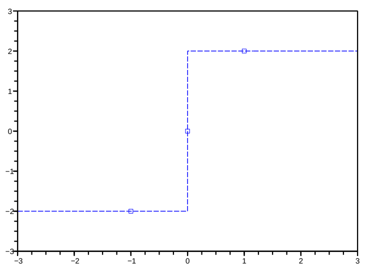
or determine no such function exists.

Consider an operator A where $\operatorname{gra} A := \{(x, x^*) \in X \times X^* \ : \ x^* \in Ax\}$. The finite family $\{(x_i, x_i^*)\}_{i \in I}$ is interpreted as the graph of a finite operator. Finite operators which have certain properties can admit antiderivatives.

Definition 4.3. *The function* f *is an antiderivative of* A *if* $\operatorname{gra} A \subseteq \operatorname{gra} \partial f$.



(a) $\partial f_1(x)$



(b) $\partial f_2(x)$

Figure 4.1: Finite convex integration is not unique up to a constant.

The continuous integration problem is well known to be unique up to a constant. For example, the antiderivative of the continuous function $f(x) = x$ is the function $F(x) = \frac{1}{2}x^2 + K$, where $K \in \mathbb{R}$. However, the finite integration problem may have many solutions.

Example 4.4. *Let* $\operatorname{gra} A = \{(-1,-2),(0,0),(1,2)\}$ *and* $f(0) = 0$ *(initial condition). Then both* $f_1(x) = x^2$ *and* $f_2(x) = 2|x|$ *are convex, lsc antiderivatives as they satisfy gra* $A \subseteq \operatorname{gra}\partial f_1(x)$ *and* $\operatorname{gra} A \subseteq \operatorname{gra}\partial f_2(x)$ *(see Problem 4.2 and Figure 4.2).*

Hence, given λ_0, we define the solution set $\mathfrak{F}$ as the set of all lsc convex functions that satisfy

$$f(x_0) = \lambda_0 \text{ and } x_i^* \in \partial f(x_i)\ \forall i \in I.$$

Altogether,

$$f \in \mathfrak{F} \Leftrightarrow \begin{cases} f \text{ is lsc, convex,} \\ f(x_0) = \lambda_0, \text{ and} \\ x_i^* \in \partial f(x_i)\ \forall i \in I. \end{cases}$$

Next we discuss the relationship between cyclic monotonicity and the existence of antiderivatives.

4.2 Cyclic monotonicity

Finite operators can have a special property called cyclic monotonicity (CM), which is necessary for an operator to admit a convex antiderivative.

Definition 4.5. *The family $\{(x_i, x_i^*)\}_{i \in I}$ is said to be CM if for all $j_0, j_1, \ldots, j_k, j_{k+1} \in I$ with $j_0 = j_{k+1}$, the inequality*

$$\sum_{l=0,\ldots,k} \langle x_{j_l}^*, x_{j_{l+1}} - x_{j_l} \rangle \leq 0$$

holds.

Proposition 4.6. *For every convex lsc function, the subdifferential operator ∂f is CM.*

Proof. Take $x_i^* \in \partial f(x_i)$ for all $i \in I$. We have

$$\begin{array}{rcl} f(x_{j_1}) - f(x_{j_0}) & \geq & \langle x_{j_0}^*, x_{j_1} - x_{j_0} \rangle, \\ f(x_{j_2}) - f(x_{j_1}) & \geq & \langle x_{j_1}^*, x_{j_2} - x_{j_1} \rangle, \\ & \vdots & \\ f(x_{j_k}) - f(x_{j_{k-1}}) & \geq & \langle x_{j_{k-1}}^*, x_{j_k} - x_{j_{k-1}} \rangle, \\ f(x_{j_{k+1}}) - f(x_{j_k}) & \geq & \langle x_{j_k}^*, x_{j_{k+1}} - x_{j_k} \rangle. \end{array}$$

Adding up all inequalities and setting $x_{j_{k+1}} = x_{j_0}$, the LHS becomes

$$[f(x_{j_1}) - f(x_{j_0})] + [f(x_{j_2}) - f(x_{j_1})] + \cdots + [f(x_{j_k}) - f(x_{j_{k-1}})] + [f(x_{j_0}) - f(x_{j_k})] = 0.$$

As a result, we get

$$\sum_{l=0,\ldots,k} \langle x_{j_l}^*, x_{j_{l+1}} - x_{j_l} \rangle \leq 0.$$

□

Definition 4.7. *A multivalued mapping ρ from X to X is said to be monotone if*

$$\langle x_1 - x_0, x_1^* - x_0^* \rangle \ \geq 0$$

holds for every (x_0, x_0^*) and (x_1, x_1^*) in the graph of ρ.

This definition corresponds to the case where $k = 1$ from the definition of CM. Thus, if ρ is a CM mapping then ρ is a monotone mapping. Geometrically, monotonicity describes a function which is "non-decreasing". This means that the graph of the function is either "increasing" or "flat". However, when the dimension of X is greater than one, there exists monotone mappings which are not cyclically monotone (for an example, see [20, p.240]).

Remark 4.8. *The most efficient way to verify that a finite operator is cyclically monotone, is to use an algorithm (such as [12, p.140]) to compute an antiderivative. If the algorithm cannot find an antiderivative then the given operator is not cyclically monotone. However, if it does find a convex antiderivative, this implies that the operator is cyclically monotone. Note that the algorithm stated in [12, p.140] runs in $O(n^3)$ time.*

4.3 Antiderivatives and their properties

A CM operator may admit many antiderivatives using a variety of different methods.

Notation 4.9. *We say that $\mathfrak{m}_A$ is a method $\mathfrak{m} : A \to \operatorname{Conv} X$ applied to an operator A, and $\mathfrak{m}_A$ is the resulting antiderivative.*

These antiderivatives may have a variety of different properties.

Definition 4.10. *Let A be a finite CM operator with $\operatorname{gra} A$ containing the points $(a_i, a_i^*)_{i \in I}$. The* *method $\mathfrak{m}$ is primal-dual symmetric if its resulting antiderivative $\mathfrak{m}_A$ satisfies*

$$\mathfrak{m}_A^* = \mathfrak{m}_{A^{-1}}.$$

Definition 4.11. *[5, Theorem 3.5] Let A be a CM operator with $\operatorname{gra} A$ containing n points (a_i, a_i^*). Then the Rockafellar function is defined by*

$$R_{A,(a_1,a_1^*)}(x) = \max_{\substack{(a_2,a_2^*)\in\operatorname{gra}A,\\ \vdots \\ (a_n,a_n^*)\in\operatorname{gra}A}} \langle a_1^*, a_2 - a_1\rangle + \cdots + \langle a_{n-1}^*, a_n - a_{n-1}\rangle + \langle a_n^*, x - a_n\rangle. \tag{4.1}$$

Remark 4.12. *We refine the above definition to be more precise. Let*

$$\sigma := \sum_{i=1}^{n+1} \langle a_{i+1} - a_i, a_i^*\rangle \leq 0, \ \textit{where } a_{n+2} = a_1$$

Then $\operatorname{gra} A$ contains at most n points, and there exists integers k and l such that $a_k = a_l$ with $1 \leq k < l \leq n+1$ and $\sigma := \sigma_1 + \sigma_2$ with

$$\begin{aligned} \sigma_1 &:= \sum_{i=k}^{l-1} \langle a_{i+1} - a_i, a_i^*\rangle, \ \textit{and} \\ \sigma_2 &:= \sum_{i=l}^{n+1} \langle a_{i+1} - a_i, a_i^*\rangle + \sum_{i=1}^{k-l} \langle a_{i+1} - a_i, a_i^*\rangle. \end{aligned}$$

Since $\sigma_1 \leq 0$ we have $\sigma \leq \sigma_2$. If we repeat the above argument k times we obtain $\sigma \leq \sigma_2 \leq \cdots \leq \sigma_k$.

As a result, we now have

$$R_{A,(a_1,a_1^*)}(x) = \max_{\substack{(a_2,a_2^*)\in \operatorname{gra} A,\\ \vdots \\ (a_n,a_n^*)\in \operatorname{gra} A}} \langle a_1^*, a_2 - a_1\rangle + \cdots + \langle a_{n-1}^*, a_n - a_{n-1}\rangle + \langle a_n^*, x - a_n\rangle. \tag{4.2}$$

such that $(a_r, a_r^*) \neq (a_s, a_s^*)$ for $r \neq s$.

Definition 4.13. *Let A be a CM operator, and let $a \in \operatorname{dom} A$. Then we set*

$$R_{A,a} := R_{A,(a,a^*)}$$

where a^* is an arbitrary point in Aa.

Definition 4.14. *Let A be CM. We say that a function $f\colon X \to \bar{\mathbb{R}}$ is an intrinsic antiderivative if* $\operatorname{gra} A \subseteq \operatorname{gra} \partial f$ *and f depends only on* $\operatorname{gra} A$.

The following example is not an intrinsic method (i.e. does not always produce intrinsic antiderivatives).

Example 4.15. *Let $e \in X$ be such that $\|e\| = 1$ and define A via* $\operatorname{gra} A := \{(-e,-e),(e,e)\}$. *Then for every $x \in X$, we have*

$$R_{A,-e}(x) = \max\big\{ -\langle x|e\rangle - 1, \langle x|e\rangle - 3\big\} = -2 + |\langle x|e\rangle - 1| \tag{4.3}$$

and

$$R_{A,e}(x) = \max\big\{ \langle x|e\rangle - 1, -\langle x|e\rangle - 3\big\} = -2 + |\langle x|e\rangle + 1|. \tag{4.4}$$

Consequently, $R_{A,e} \not\geq R_{A,-e}$ and $R_{A,e} \not\leq R_{A,-e}$.

There are several types of intrinsic methods when the graph of A is finite.

Example 4.16. *Let A be cyclically monotone such that $\operatorname{gra} A$ is finite. Then the two methods*

$$\max_{(a,a^*)\in \operatorname{gra} A} R_{A,(a,a^*)},$$

and

$$\sum_{(a,a^*)\in \operatorname{gra} A} \frac{1}{n_A} R_{A,(a,a^*)}$$

produce intrinsic methods of A (n_A is the number of points in $\operatorname{gra} A$).

Let us consider a process to build a primal-dual symmetric method from any intrinsic antiderivative.

Definition 4.17. *Let $f_0, f_1 \in \mathcal{F}$, then the proximal midpoint average of f_0 and f_1 is the function*

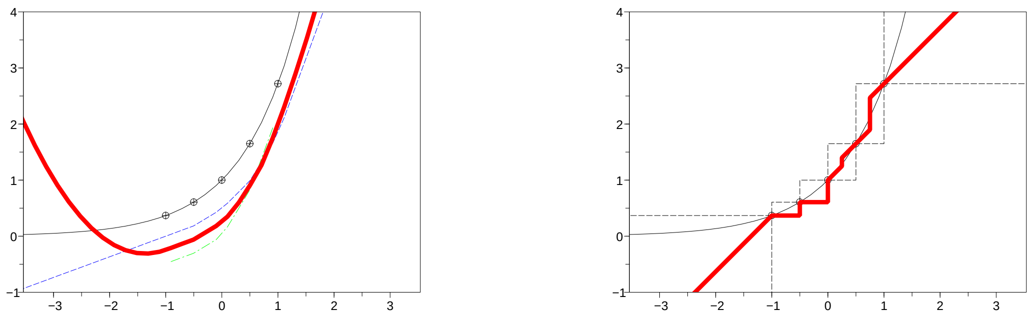
$$\mathcal{P}(f_0, \frac{1}{2}, f_1) := \Big(\tfrac{1}{2}\big(f_0 + \tfrac{1}{2}\|\cdot\|^2\big)^* + \tfrac{1}{2}\big(f_1 + \tfrac{1}{2}\|\cdot\|^2\big)^*\Big)^* - \tfrac{1}{2}\|\cdot\|^2. \tag{4.5}$$

Fact 4.18. *[5, Corollary 4.17] Let $\mathfrak{m}$ be a method which produces an intrinsic method with full domain. Then $\mathcal{P}\big(m_A, \frac{1}{2}, m^*_{A^{-1}}\big)$ and $\mathcal{P}\big(m_{A^{-1}}, \frac{1}{2}, m^*_A\big)$ are both primal-dual symmetric antiderivatives for A and A^{-1} respectively. Moreover, they have full domain.*

The importance of primal-dual symmetric methods can be illustrated by the following diagram.

$$\begin{tikzcd} A \arrow[r, "*"] \arrow[d, "\mathfrak{m}"'] & A^{-1} \arrow[d, "\mathfrak{m}"'] \\ \mathfrak{m}_A = \mathcal{P}\big(m_A, \frac{1}{2}, m^*_{A^{-1}}\big) \arrow[r, "*"] & \mathfrak{m}_{A^{-1}} = \mathcal{P}\big(m_{A^{-1}}, \frac{1}{2}, m^*_A\big) \end{tikzcd}$$

In general (within our framework), we always have primal-dual symmetry at the discrete level: $x_i^* \in A(x_i) \Leftrightarrow x_i \in A^{-1}(x_i^*)$; however, there is no reason to expect this type of symmetry at the continuous level, i.e. $\mathfrak{m}_A^* = \mathfrak{m}_{A^{-1}}$. It turns out that when the antiderivatives m_A, and $m_{A^{-1}}^*$ are intrinsic methods, the proximal midpoint average $\mathcal{P}\big(m_A, \frac{1}{2}, m_{A^{-1}}^*\big)$ produces an antiderivative which is primal-dual symmetric at the continuous level [5, Example 4.19 and Example 4.20].



(a) Convex antiderivatives.

(b) The subdifferential.

Figure 4.2: Constructing a primal-dual symmetric method.

In Figure 4.2(a), the family $\operatorname{gra} A := \big\{(a, \exp(a)) \;:\; a \in \{-1, -\frac{1}{2}, 0, \frac{1}{2}, 1\}\big\}$ is denoted by the "circle-cross" marks on the function $f(x) = \exp(x)$. The "dashed" function and the "dashed-dotted" function are two antiderivatives m_A, and $m_{A^{-1}}^*$, respectively. As we are interested in finding an antiderivative between these two functions, we take the proximal midpoint average $\mathcal{P}\big(m_A, \frac{1}{2}, m_{A^{-1}}^*\big)$ represented by the "thick" function. This function is primal-dual symmetric.

We see from Figure 4.2(b) that the "thick" function is $\partial\mathcal{P}\big(m_A, \frac{1}{2}, m_{A^{-1}}^*\big)$ and is an antiderivative (as $\operatorname{gra} A \subset \operatorname{gra} \partial\mathcal{P}$). Any convex antiderivative will have a monotone subdifferential, hence $\partial\mathcal{P}\big(m_A, \frac{1}{2}, m_{A^{-1}}^*\big)$ lies within the "dashed" rectangles.

4.4 Relationship between [12] and [5]

4.4.1 Minimal antiderivative in higher dimension

Fact 4.19. *[12, p.132] Let Γ be the set of all ordered subsets $J = \{j_0, \cdots, j_k\}$ of I with $j_0 = 0$, and for all $r \neq s$, $j_r \neq j_s$. Then, for each $J \in \Gamma$, we define the function*

$$f_J(x) := \langle x^*_{j_0}, x_{j_1} - x_{j_0}\rangle + \cdots + \langle x^*_{j_{k-1}}, x_k - x_{j_{k-1}}\rangle + \langle x^*_{j_k}, x - x_{j_k}\rangle \tag{4.6}$$

on X. Then for a given λ_0 and x_0, the minimal antiderivative f^- is defined as

$$f^-(x) := \lambda_0 + \max_{J\in\Gamma} f_J(x). \tag{4.7}$$

The Rockafellar function $R_{A,(a,a^*)}$ is a special method that produces minimal antiderivatives.

Remark 4.20. *We say that an antiderivative is minimal if it is the greatest piecewise linear function which bounds all other antiderivatives from below. Similarly, the maximal antiderivative is the smallest piecewise linear function which bounds all other antiderivatives from above.*

Fact 4.21. *[5, Theorem 3.5] Let A be CM and $a \in \operatorname{dom} A$. Then*

$$R_{A\cdot(a,a^*)} = \min\{f \in \mathcal{F} \,:\, f \textit{ is an antiderivative of } A \textit{ and } f(a) = 0\}.$$

As both f^- and $R_{A,(a,a^*)}$ are minimal antiderivatives, they must be equal.

Transformation 4.22. *Let Γ be the collection of all subsets $J = \{j_0, \cdots, j_k\}$ of I such that* *and*$j_0 = 0$*, and for all* $r \neq s$*,* $j_r \neq j_s$*. Let us rename the following indexes and variables:*

$$\begin{aligned} J = \{j_0, j_1, \cdots, j_k\} &:= \{1, 2, \cdots, n\}, \\ x_i &:= a_i \ \ \textit{for } i = 1, \cdots, n-1, \\ x_i^* &:= a_i^* \ \ \textit{for } i = 1, \cdots, n-1, \\ x_0 &:= a_1, \ \textit{and} \\ x_0^* &:= a_1^*. \end{aligned}$$

Theorem 4.23. *Assume that* A *is a CM operator and that* $\lambda_0 = 0 = f^-(a)$*. Then*

$$R_{A,(a,a^*)} = f^-,$$

where $f^- \in \mathcal{F}$ *and* $(a, a^*) \in \operatorname{gra} A$*.*

Proof. Recall, that

$$f^-(x) := \lambda_0 + \max_{J \in \Gamma} f_J(x),$$

where $\lambda_0 = 0$. For each $J \in \Gamma$ we have

$$f_J(x) := \langle x_{j_0}^*, x_{j_1} - x_{j_0} \rangle + \cdots + \langle x_{j_{k-1}}^*, x_{j_k} - x_{j_{k-1}} \rangle + \langle x_{j_k}^*, x - x_{j_k} \rangle. \tag{4.8}$$

After redefining Equation (4.8) by Transformation 4.22, we get

$$f_J(x) = \langle a^*, a_2 - a \rangle + \cdots + \langle a_{n-1}^*, a_n - a_{n-1} \rangle + \langle a_n^*, x - a_n \rangle.$$

Taking the maximum gives the announced result. □

Similarly, the maximal antiderivatives are the same.

Fact 4.24. *[12, p.137] Let* $\mathcal{H} := \{h \in X \ : \ h(x_0^*) = \lambda_0^*, \lambda_0^* = \langle x_0, x_0^* \rangle - \lambda_0, x_i \in \partial h^-(x_i^*) \ \forall i \in I\}$

Then, by [12, Theorem 3.4] there exists at most one minimal antiderivative h^- which is piecewise linear with full domain.

Remark 4.25. *Note that* h^- *is the minimal antiderivative in the dual space* $\mathcal{H}$*. It is unique as it is the greatest piecewise linear function which bounds all other antiderivatives (in* $\mathcal{H}$*) from below. If we wish to compute* $f^+ := (h^-)^*$*, it is completely characterized in [12, Theorem 4.3].*

Observation 4.26. *Since* $h^- \in \mathcal{H}$ *we have the following constraints:*

i. $h^-(x_0^*) = \lambda_0^* \ : \ \lambda_0^* = \langle x_0, x_0^* \rangle - \lambda_0$,

ii. $x_i \in \partial h^-(x_i^*) \ \forall i \in I$.

Taking the conjugate of the minimal antiderivative in the dual space will yield the maximal antiderivative in the primal space. Hence $(h^-)^*$ is maximal in $\mathcal{F}$.

Notation 4.27. *Throughout we will denote the maximal antiderivative* $(h^-)^*$ *as* f^+*.*

Fact 4.28. *[5, Corollary 3.10] Let* A *be CM and* $a \in \operatorname{dom} A$*,*

$$R^*_{A^{-1},(a^*,a)} - \langle a, a^* \rangle = \max\{f \in \mathcal{F} \ : \ f \textit{ is an antiderivative of } A \textit{ and } f(a) = 0\}.$$

Theorem 4.29. *Assume* A *is a CM operator and that* $f^+(a) = \lambda_0^* = 0$*. Then*

$$R^*_{A^{-1},(a^*,a)} - \langle a, a^* \rangle = f^+,$$

where $f^+ := (h^-)^*$*,* $h^- \in \mathcal{H}$*, and* $(a, a^*) \in \operatorname{gra} A$*.*

Proof. It suffices to show

$$h^- = R_{A^{-1},(a^*,a)} + \langle a, a^* \rangle.$$

Recall,

$$h^- \in \mathcal{H} \Leftrightarrow \begin{cases} h^-(x_0^*) = \lambda_0^* \ : \ \lambda_0^* = \langle x_0, x_0^* \rangle - \lambda_0, \\ x_i \in \partial h^-(x_i^*) \ \forall i \in I. \end{cases}$$

From $\lambda_0 = \lambda_0^* - \langle x_0, x_0^* \rangle$ and $\lambda_0^* = 0$ we have

$$\begin{aligned} h^-(x) &= \lambda_0 + \max_{J \in \Gamma} h_J(x) && (4.9) \\ &= \lambda_0^* - \langle x_0, x_0^* \rangle + \max_{J \in \Gamma} h_J(x) && (4.10) \\ &= \max_{J \in \Gamma} h_J(x) - \langle x_0, x_0^* \rangle && (4.11) \end{aligned}$$

where for each $J \in \Gamma$ we have

$$h_J(x) \ := \ \langle x_{j_0}, x_{j_1}^* - x_{j_0}^* \rangle + \cdots + \langle x_{j_{k-1}}, x_k^* - x_{j_{k-1}}^* \rangle + \langle x_{j_k}, x - x_{j_k}^* \rangle.$$

After redefining Equation (4.11) by Transformation 4.22, we get

$$h^-(x) = \max_{J \in \Gamma} h_J(x) - \langle a, a^* \rangle$$

where for each $J \in \Gamma$ we have

$$h_J(x) = \langle a, a_2^* - a^* \rangle + \cdots + \langle a_{n-1}, a_n^* - a_{n-1}^* \rangle + \langle a_n, x - a_n^* \rangle.$$

□

4.4.2 Minimal antiderivative in one dimension

In one dimension Problem 4.2 has a closed form for the minimal antiderivative.

Fact 4.30. *[5, Theorem 3.14] Let A have a finite graph and suppose that the graph of B : conv(Ax) is*

$$\cup_{i=1}^{n}(\{a_i\} \times [b_i^-, b_i^+]),$$

where $n \in \{1, 2, \cdots\}$, $a_1 < a_2 < \cdots < a_n$, and $b_1^- \leq b_1^+ \leq b_2^- \leq \cdots \leq b_n^- \leq b_n^+$. Set $a_0 := -\infty$ and $a_{n+1} := +\infty$. Suppose that $k \in \{1, \cdots, n\}$. Then $R_{A,(a_k,a_k^*)}$ is given by

$$x \mapsto \begin{cases} (x-a_i)b_i^- + \sum_{j=i+1}^{k}(a_{j-1}-a_j)b_j^-, & \textit{if } a_{i-1} < x \leq a_i \leq a_k; \\ (x-a_i)b_i^+ + \sum_{j=k}^{i-1}(a_{j+1}-a_j)b_j^+, & \textit{if } a_k \leq a_i \leq x < a_{i+1}; \end{cases}$$

and $\partial R_{A,(a_k,a_k^*)}$ is given by

$$x \mapsto \begin{cases} \{b_i^-\}, & \textit{if } a_{i-1} < x \leq a_i \leq a_k; \\ [b_i^-, b_{i+1}^-], & \textit{if } x = a_i < a_k; \\ [b_k^-, b_k^+], & \textit{if } x = a_k; \\ [b_{i-1}^+, b_i^+], & \textit{if } a_k < x = a_i; \\ \{b_i^+\}, & \textit{if } a_k \leq a_i < x < a_{i+1}; \end{cases}$$

Remark 4.31. *In one dimension, the closed form of the minimal antiderivative in [12, p.145] is exactly the same closed form of the Rockafellar function in [5, Theorem 3.14] since both closed forms have full domain and are piecewise linear minimal antiderivatives.*

In one dimension we have a closed form of the minimal antiderivative. Therefore the time complexity is linear.

4.5 Computational Algorithms for Antiderivatives

The most efficient algorithms used to compute minimal and maximal antiderivatives are described in [12, p.139-140]. They are specialized algorithms driven by the following LP formulation.

Fact 4.32. *[12, Proposition 3.5] The optimal solutions of*

$$\min_{\lambda}\{ \sum_{i\in I, i\neq 0} \lambda_i \ : \ \lambda_j - \lambda_i \geq \langle x_i^*, x_j - x_i \rangle \quad \big(\forall\ i,j \in I\big)\} \tag{4.12}$$

correspond to the values $f^-(x_i)$ (i.e. $\lambda_i = f^-(x_i)$ for $i = 1,\ldots,n$), which completely determines the minimal antiderivative $R_{A,(a,a^*)}$.

Remark 4.33. *We only need to discuss the minimal antiderivative, as the maximal antiderivative is determined through conjugate duality, see [12, p.137] and [5, Corollary 3.10].*

Definition 4.34. *Considering the set of linear inequalities $Ax \leq b$, we say that the system $Ax \leq b$ is a difference of constraints when the constraint matrix A contains one $+1$ and one -1 in each row; all other entries are zero, for all rows.*

As x_i, x_i^*, x_j are given, the constraint matrix in Equation (4.12) is a system of difference constraints. It turns out that we can find a feasible solution (an antiderivative) to the Linear program in Equation (4.12) by finding a solution to the shortest path problem in the corresponding graph formulation (see [1, p.103-105]).

Example 4.35. *Let* $\operatorname{gra} A := \{(0,0),(-1,-2),(1,2)\}$. *Then by Equation (4.12) we have the fol-*

lowing constraints:

$$\begin{aligned}
\lambda_0 - \lambda_1 &\leq 0,\\
\lambda_0 - \lambda_2 &\leq 0,\\
\lambda_1 - \lambda_0 &\leq 2,\\
\lambda_1 - \lambda_2 &\leq 4,\\
\lambda_2 - \lambda_0 &\leq 2,\\
\lambda_2 - \lambda_1 &\leq 4.
\end{aligned}$$

Each system of difference constraints is associated with a graph whose nodes are λ_i *and each constraint* $\lambda_i - \lambda_j \leq \langle x_i^*, x_i - x_j\rangle$ *represents the arc* (i, j) *with the weighed value* $\langle x_i^*, x_j - x_i\rangle$*. Hence, we obtain the following graph.*

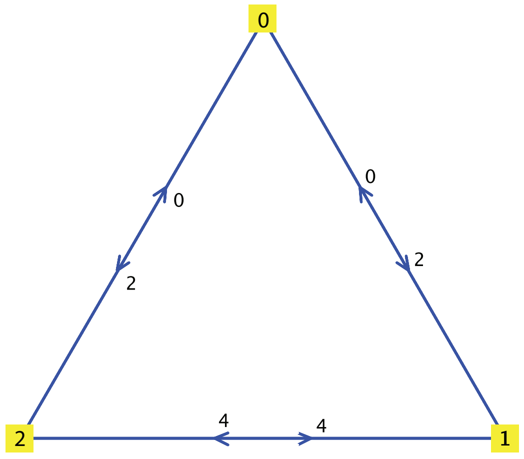


Figure 4.3: Graph associated with a system of difference constraints.

In this specific example we notice that all weights are nonnegative, therefore we can use Dijkstra's algorithm to obtain the shortest path distances from a particular node to every other node. Thus the following table represents the shortest path distances from node i to node j.

$d(i,j)$	*Node* 0	*Node* 1	*Node* 2
Node 0	0	2	2
Node 1	0	0	2
Node 2	0	2	0

By [1, p.104] and Fact 4.32 we have $d(i,j) = \lambda_j = f(x_j)$. Hence,

i. $f_1(x_0 = 0) = \lambda_0 = 0,\ f_1(x_1 = -1) = \lambda_1 = 2,\ f_1(x_2 = 1) = \lambda_2 = 2,$

ii. $f_2(x_0 = 0) = \lambda_0 = 0,\ f_2(x_1 = -1) = \lambda_1 = 0,\ f_2(x_2 = 1) = \lambda_2 = 2,$

iii. $f_3(x_0 = 0) = \lambda_0 = 0,\ f_3(x_1 = -1) = \lambda_1 = 2,\ f_3(x_2 = 1) = \lambda_2 = 0.$

We recall that $\lambda_j = f(x_j)$ for $j = 0, \ldots, n$ completely determines an antiderivative f. Therefore the above list determines exactly three antiderivatives (f_1, f_2, and f_3). However, not one of these antiderivatives is the minimal antiderivative as the minimal antiderivative corresponds to the case $\lambda_i = f(x_i)$ as illustrated on Figure 4.4.

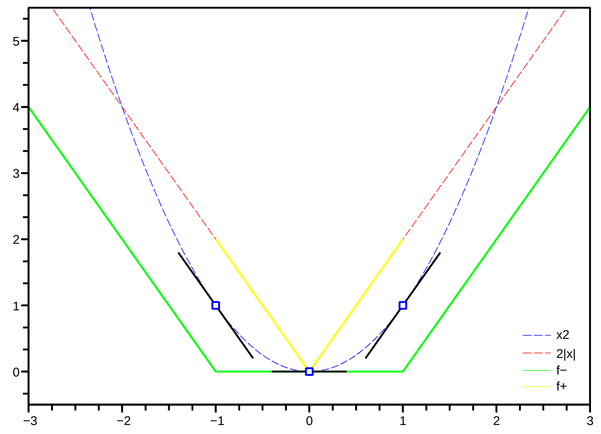


Figure 4.4: Minimal and maximal antiderivatives.

Therefore, generally we cannot find a minimal antiderivative by solving its corresponding shortest path problem, but we can find several antiderivatives. In fact, for each node in the graph G we can find at most one antiderivative. Moreover, if the weights $\langle x_i^*, x_j - x_i\rangle$ are all nonnegative we can find an antiderivative in $O(n^2)$ time using Dijkstra's algorithm. On the other hand, if we remove the nonnegative weight assumption, we can find several antiderivatives in $O(n^3)$ using Floyd-Warshall algorithm which is described in [1, p.147]. Note that the graph of G is always full (i.e. n^2 arcs), so labeling algorithms [1, p.155] (whose complexity depends on the number of arcs) will not help us compute an antiderivative more efficiently.

Remark 4.36. *As many convex transforms rely on the subgradient constraint in Equation (4.12), they may also be linked to the shortest path problem.*

Chapter 5

Conclusion

We visited the proximal average which is an advanced convex transform. It inherits a much richer set of properties (continuity, differentiability, full domain, and strict convexity properties) than the arithmetic average. Due to the complicated nature of the proximal average, existing computational algorithms failed to accurately approximate nested transforms (without complications). This shortcoming is remedied by the PLQ model and its corresponding algorithms. As these algorithms depend on convexity, we extended their framework to nonconvex functions using the convex hull. We also explicitly defined each algorithm and stated its general strategy.

We showed that a method used to produce primal-dual symmetric antiderivatives depends on the proximal midpoint average and Rockafellar's function. The latter is a minimal antiderivative, and through conjugate duality, it also provides a maximal antiderivative. Therefore, we showed that the minimal and maximal antiderivatives in [5, Theorem 3.5, Corollary 3.10] are the same minimal and maximal antiderivatives as in [12, p.132, p.136]. So the algorithms in [12, p.139, p.140] can be used to compute primal-dual symmetric antiderivatives. Moreover, these minimal and maximal antiderivatives can be computed by solving a shortest path problem. In fact, the specialized algorithms used to solve the all-pairs shortest path problem [1, p.144] can be used to find many antiderivatives (including Rockafellar's functions).

Future work may focus on the following extensions.

- Developing a second-order model to accommodate the PLQ algorithms.

- Analyzing convergence for a second-order model.
- Extending the PLQ model to higher dimension.
- Generalizing the PLQ matrix to represent a more general class of functions.
- Expanding the PLQ model to include other convex operations which have the same closure properties.
- Exploring the link between convex transforms and shortest path problems.

Bibliography

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows*, Prentice Hall Inc., Englewood Cliffs, NJ, 1993. Theory, algorithms, and applications.

[2] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, *The quickhull algorithm for convex hulls*, ACM Transactions on Mathematical Software, 22 (1996), pp. 469–483.

[3] H. H. Bauschke, R. Goebel, Y. Lucet, and X. Wang, *The proximal average: Basic properties*, tech. report, University of British Columbia, 2007.

[4] H. H. Bauschke, Y. Lucet, and M. Trienis, *How to transform one convex function continuously into another*, tech. report, University of British Columbia, July 2006. Accepted for publication in SIAM Review.

[5] H. H. Bauschke, Y. Lucet, and X. Wang, *Primal-dual symmetric intrinsic methods for finding antiderivatives of cyclically monotone operators*, SIAM J. Control Optim., 46 (2007), pp. 2031–2051.

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

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

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

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

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

[11] 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.

[12] D. Lambert, J.-P. Crouzeix, V. H. Nguyen, and J.-J. Strodiot, *Finite convex integration*, J. Convex Anal., 11 (2004), pp. 131–146.

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

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

[15] ——, *La Transformée de Legendre–Fenchel et la convexifiée d'une fonction : algorithmes rapides de calcul, analyse et régularité du second ordre*, PhD thesis, Laboratoire Approximation et Optimisation, UFR MIG, Université Paul Sabatier, Feb. 1997.

[16] ——, *Fast Moreau Envelope computation II: Applications*, tech. report, University of British Columbia, 2005.

[17] ——, *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.

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

[19] Y. Lucet, H. H. Bauschke, and M. Trienis, *The piecewise linear-quadratic model for computational convex analysis*, Comput. Optim. Appl., (2006). Accepted for publication.

[20] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, New York, 1970.

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

Appendix A

```
//PROBLEM 1. (1) f1'(x2) = f2'(x4),
//           (2) m1(x-x2) + f1(x2) = m2(x-x4) + f2(x4), where
//           (3) x1 < x2 < x3
//           (4) x3 < x4 < x5
//
//PROBLEM 2. (1) m1(x-x2) + f1(x2) = m1(x-x5) + f2(x5), where
//           (2) x1 < x2 < x3
//           (3) x3 < x4 < x5, and
//           (4) m1=f1'(x2)
//
//PROBLEM 3. (1) m2(x-x1) + f1(x1) = m2(x-x4) + f2(x4), where
//           (2) x1 < x2 < x3
//           (3) x3 < x4 < x5, and
//           (4) m2=f2'(x4)
//
//PROBLEM 4. (1) m1(x-x1) + f1(x1) = m1(x-x5) + f2(x5), where
//           (2) x1 < x2 < x3
//           (3) x3 < x4 < x5
```

```
function plqco=plq_co(plqf)
    EPS=1E-15;
    plqf_original=plqf; plqfLHS=[];plqfRHS=[];
    //Bounded plq functions will have an infinity value as a coefficient
    //in either the first row or the last row, simply remove these pieces
    //and add these pieces after the convex hull has been computed.
    if or(plqf(1,2:$) == %inf) then
      plqfLHS = plqf(1,:); plqf=plqf(2:$,:);
    end;
    if or(plqf($,2:$) == %inf) then
      plqfRHS = plqf($,:); plqf=plqf(1:$-1,:);
    end;
    n = size(plqf,1);
    x=plqf(:,1);a=plqf(:,2);b=plqf(:,3);c=plqf(:,4); //update the coefficients
    if a(1) < 0  | a(n) < 0 | (a(1)==0 & a(n) == 0 & b(n) < b(1) ) then
    plqco=[%inf,0,0,-%inf]; return; end; //special case
    i=1;
    while i <= n-1 & n >= 2,
        a=clean(a(:,1)); b=clean(b(:,1)); c=clean(c(:,1));
        //Considering a PLQ function with two pieces, it is nonconvex
        //if the following condition holds.
        if (2*a(i)*x(i)+b(i)) - (2*a(i+1)*x(i)+b(i+1)) > EPS |  a(i) < 0 | a(i+1) < 0 then
            if n >=3 & i >= 2 then x1 = plqf(i-1,1); else x1 = -%inf; end;
            f1=plqf(i,:); f2=plqf(i+1,:);x3=f1(1,1); x5=f2(1,1);
            qplot(plqf_original,plqf,-%inf,%inf);
            plqf = [plqf(1:(i-1),:);plq_conv_on_interval(f1,f2,x1,x3,x5);plqf((i+2):$,:)];
            plqf = plq_clean(plqf);
            if n <= 1 then plqco=[plqf]; break; end;
            if i > 1 then i=i-1; end; //backtrack
          else i=i+1; end;
          n = size(plqf,1); //update size of the PLQ function
          x=plqf(:,1);a=plqf(:,2);b=plqf(:,3);c=plqf(:,4); //update the coefficients
    end;
    plqco=[plqfLHS;plqf;plqfRHS];
    qplot(plqf_original,plqco,-%inf,%inf);
    return;
endfunction
```

```
//We assume x(1) < x(2) < x(3) < x(4) < x(5) are ordered and finite
//(no infinity values). In creating the PLQ hull we may need to create
//points x(2) and x(4). Note that x(1) is the LEFTBOUND, x(3) is the
//partition point, and x(5) is the RIGHTBOUND in our PLQ interval.
//plq_conv_on_interval finds the convex hull on the interval [x1,x5].
function [plqco] = plq_conv_on_interval(f1,f2,x1,x3,x5)
  //Assume f1 is to the left of f2
  x(1) = x1; a(1)=f1(1,2); a(2)=f2(1,2);
  x(3) = x3; b(1)=f1(1,3); b(2)=f2(1,3);
  x(5) = x5; c(1)=f1(1,4); c(2)=f2(1,4);
  ieee(2); //allow infinity i.e. 1/0
  plqco=[];
  if a(1) < 0 then //concave pieces are replaced by a linear function.
      f1=_plq_conv_buildl(x,a,b,c,1,1,1,3);
      f2=[x(5),a(2),b(2),c(2)];
      plqco=plq_conv_on_interval(f1,f2,x(1),x(3),x(5));  return;
  elseif a(2) < 0 then
      f1=[x(3),a(1),b(1),c(1)];
      f2=_plq_conv_buildl(x,a,b,c,2,5,2,3);
      plqco=plq_conv_on_interval(f1,f2,x(1),x(3),x(5)); return;
  elseif 2*a(1)*x(3)+b(1) <= 2*a(2)*x(3)+b(2) then //triveral case
      plqco=[x(3),a(1),b(1),c(1);x(5),a(2),b(2),c(2)]; return;
  elseif a(1) == 0 & a(2) == 0 then //(LINEAR-LINEAR) f1 is linear and f2 is linear.
```

```
    [plqco]=_plq_conv_buildl(x,a,b,c,1,1,2,5); return;
elseif a(1) == 0 & a(2) <> 0 then
    //(LINEAR-QUDARTIC) if f1 is linear and f2 is quadratic then x(4) is determined.
    plqco=[_conv_interval_routine(x,a,b,c,x1,x3,x5)]; return;
elseif a(1) <> 0 & a(2) == 0 then
    //(QUADRATIC-LINEAR) if f1 is quadratic and f2 is linear then x(2) is determined.
    plqco=[_conv_interval_routine(x,a,b,c,x1,x3,x5)]; return;
elseif a(1) <> 0 & a(2) <> 0 then
    //(QUADRATIC-QUADRATIC) if f1 is quadratic and f2 is quadratic
    //we need to find solutions to A*x(4)^2 + B*x(4) + C.
    //The following code solves [PROBLEM 1.],
    A=(1/4)*(-4*a(2)^2+4*a(2)*a(1))/a(1);
    B=(1/4)*(-4*a(2)*b(2)+4*b(1)*a(2))/a(1);
    C=(1/4)*(-b(2)^2-b(1)^2+4*c(1)*a(1)+2*b(1)*b(2)-4*c(2)*a(1))/a(1);
    D = B^2 - 4*A*C; //Discriminant.
    if A == 0 then //Linear case, so we have Bx+C=0 with one zero.
        x(4) = (-C/B);
        x(2) = (-1/2)*(b(1)-2*a(2)*x(4)-b(2))/a(1);
        if x(2) < x(1) | x(3) < x(2) then x(2)=%inf; end;
        if x(4) < x(3) | x(4) > x(5) then x(4)=%inf; end;
    //if x(2) or x(4) are outside the interval, send the x(i) to infinity.
    //In otherwords these solutions provide no information and thus we
    //should just ignore them. Similarly if D < 0, just ignore x(2) and x(4)
    //as they don't help with the hull.
    else // A <> 0, Quadratic case, so we have Ax^2+Bx+C=0, with one or two zeros.
```

```
          //Solving for x(4).
           if D>0 then
               nr = (-B - sqrt(D))/(2*A); //negative root
               pr = (-B + sqrt(D))/(2*A); //positive root
               if D==0 then nr=pr; end;
               if x(3) < pr & pr < x(5) then x(4) = pr;
               elseif x(3) < nr & nr < x(5) then x(4) = nr;
               else x(4)=%inf; end;
               x(2) = (-1/2)*(b(1)-2*a(2)*x(4)-b(2))/a(1);
               if x(1) > x(2) | x(2) > x(3) then x(2)=%inf; end;
           else x(2)=%inf; x(4)=%inf;
           end
     end
     if x(2) < %inf & x(4) < %inf then
         plqco=[x(2),a(1),b(1),c(1);
         _plq_conv_buildl(x,a,b,c,1,2,2,4);x(5),a(2),b(2),c(2)]; return;
     else //x(2) == %inf | x(4) == %inf,
        //Here we solve the SECOND PROBLEM: as x(2) or x(4) is infeasible in [PROBLEM 1.]
         plqco=[_conv_interval_routine(x,a,b,c,x1,x3,x5)]; return;
     end;
 else printf("Error: Case does not exist");
 end
endfunction
```

```
//if we can't find a solution to [PROBLEM 1.] then we need to find a solution to [PROBLEM 2.].
//The following routine solves [PROBLEM 2.].
function [plqco] = _conv_interval_routine(x,a,b,c,x1,x3,x5),
        EPS=1E-15; x(1)=x1;x(3)=x3;x(5)=x5;
        //solving the quadratic for x(2)
        if a(1) <> 0 & x(5) < %inf then
            A1 = -a(1);
            B1 = 2*a(1)*x5;
            C1 = c(1)+b(1)*x5-a(2)*x5^2-b(2)*x5-c(2);
            D1 = B1^2 - 4*A1*C1;
            pr1 = (-B1 + sqrt(D1))/(2*A1); //solution is x2
            nr1 = (-B1 - sqrt(D1))/(2*A1); //negative root for the LINEAR-QUADRATIC CASE.
            if x(1) <= pr1 & pr1 <= x(3) then x(2) = pr1;
              plqco=[x(2),a(1),b(1),c(1);_plq_conv_buildl(x,a,b,c,1,2,2,5)]; return;
            elseif x(3) <= pr1 & pr1 <= x(5) then x(4)=pr1;
              plqco=[_plq_conv_buildl(x,a,b,c,1,1,2,4);x(5),a(2),b(2),c(2)];  return;
            elseif x(1) <= nr1 & nr1 <= x(3) then x(2) =pr1;
              plqco=[x(2),a(1),b(1),c(1);_plq_conv_buildl(x,a,b,c,1,2,2,5)]; return;
            elseif x(3) <= nr1 & nr1 <= x(5) then x(4)=pr1;
              plqco=[_plq_conv_buildl(x,a,b,c,1,1,2,4);x(5),a(2),b(2),c(2)];  return;
            else plqco=_plq_conv_buildl(x,a,b,c,1,1,2,5); return;
            end;
        //solving the quadratic for x(4)
        elseif a(2) <> 0 & x(1) > -%inf then
```

```
            A2 = a(2);
            B2 = -2*a(2)*x(1);
            C2 = -b(2)*x(1)+a(1)*x(1)^2+b(1)*x(1)+c(1)-c(2);
            D2 = B2^2 - 4*A2*C2;
            pr2 = (-B2 + sqrt(D2))/(2*A2);
            nr2 = (-B2 - sqrt(D2))/(2*A2);
            if x(3) <= pr2 & pr2 <= x(5)  then x(4)=pr2;
              plqco=[_plq_conv_buildl(x,a,b,c,1,1,2,4);x(5),a(2),b(2),c(2)]; return;
            elseif x(1) <= pr2 & pr2 <= x(3)  then x(2)=pr2;
              plqco=[x(2),a(1),b(1),c(1);_plq_conv_buildl(x,a,b,c,1,2,2,5)]; return;
            elseif x(3) <= nr2 & nr2 <= x(5)  then x(4)=nr2;
              plqco=[_plq_conv_buildl(x,a,b,c,1,1,2,4);x(5),a(2),b(2),c(2)];  return;
            elseif x(1) <= nr2 & nr2 <= x(3)  then x(2)=nr2;
              plqco=[x(2),a(1),b(1),c(1);_plq_conv_buildl(x,a,b,c,1,2,2,5)]; return;
            else plqco=_plq_conv_buildl(x,a,b,c,1,1,2,5); return;
            end;
        else plqco=_plq_conv_buildl(x,a,b,c,1,1,2,5); return;
        end;
endfunction
```