The Legendre–Fenchel Conjugate: Numerical Computation

Yves Lucet *

Abstract: This paper describes a numerical implementation in Maple V R. 5 of an algorithm to compute the Legendre–Fenchel conjugate, namely the Linear-time Legendre Transform algorithm. After a brief motivation on the importance of the Legendre–Fenchel transform, we illustrate the information the conjugate gives (how to test for convexity or compacity, and how to smooth a convex function), with several examples (including solving a Hamilton-Jacobi equation). The last section shows the convergence behavior. The package is available from the Computational Convex Analysis web page at http://www.cecm.sfu.ca/projects/CCA or directly at http://www.cecm.sfu.ca/projects/CCA/LLT.

Introduction

Symbolic computation of the Legendre–Fenchel conjugate has been studied recently in [1]. However, it can only be limited to special classes of functions. Therefore several authors studied fast algorithms for numerical computation [2, 4, 7], so called fast Legendre transform algorithms which have a log-linear worst-case computation time. Later, a linear algorithm, the Linear-time Legendre Transform algorithm, was studied in [5, 6]. We describe a package implementing it in Maple V.5.

The Legendre–Fenchel conjugate

Given a function \( f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\} \), its Legendre–Fenchel conjugate (also called conjugate, convex conjugate, or Fenchel conjugate) is defined by:

\[
 f^*(s) := \sup_{x \in \mathbb{R}^n} [\langle s, x \rangle - f(x)]
\]

(1)

where \( \langle s, x \rangle \) denotes the scalar product. It is fruitful to allow the function to take the value \( \infty \) as it allows to tackle constrained and unconstrained minimization problems simultaneously. The domain of the function: \( \text{Dom } f := \{ x \in \mathbb{R}^n : f(x) < \infty \} \) is then a natural object to consider (see [3, 8] for more on convex analysis).

Before describing the package, let me emphasize the importance of the Fenchel conjugate in duality theory.

Consider \( f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\} \), \( g : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\} \), a linear operator \( A : \mathbb{R}^n \to \mathbb{R}^m \), and the following problems:

\[
 p := \inf [f(x) + g(Ax)] ,
\]

(2)

\[
 d := - \inf [f^*(-A^T z) + g^*(z)] ,
\]

(3)

where \( A^T \) denotes the transpose of \( A \). Problem (2) is usually called the primal problem and (3) the dual problem. The basic idea of duality theory is that the dual problem gives information on the primal problem and may be easier to solve.

More precisely, weak duality always hold: \( d \leq p \). In addition, under some constraint qualifications (for example: \( A(\text{Dom } f) \cap \text{int (Dom } g) \neq \emptyset \)) strong duality holds: \( p = d \) with the infimum attained in (3).

Finally, Fenchel’s duality theorem characterizes when both problems attain their infima:

\[
 x \text{ solves (2) and } z \text{ solves (3)}
\]

if and only if

\[
 Ax \in \partial g^*(z) \text{ and } -A^T z \in \partial f(x).
\]

The notation \( \partial \) denotes the convex subdifferential:

\[
 \partial f(x) := \{ s \in \mathbb{R}^n : f(\cdot) \geq f(x) + \langle s, \cdot - x \rangle \}
\]

Fenchel’s theorem has far reaching applications. Indeed, linear programming is only one particular case of it.

Consequently, obtaining a good intuition of the shape of the conjugate may give a good feeling of why and where a problem is difficult. Our aim is to compute the

---

*CECM, Dept. Math. and Stat., Simon Fraser University, Burnaby BC, V5A 1S6 Canada, lucet@cecm.sfu.ca
A tour of the LLT package

Let us give two simple examples.

First, we compute the conjugate of a quadratic bivariate function.

```maple
A:=matrix(2,2,[[2,1],[1,3]]);
B:=inverse(A);
f:=proc(x,y) local v;v:=array([x,y]);
1/2*multiply(transpose(v),
multiply(A,v));end;
g:=proc(x,y) local v;v:=array([x,y]);
1/2*multiply(transpose(v),
multiply(B,v));end;
P:=Conjugate(f,[-50..50,-50..50],[20,20],
[-9..9,-12..12],[10,10],
output=plot,outcompress=false):
Q:=plot3d(g(x,y),x=-9..9,y=-12..12):
display({P,Q});
```

The function \( g \) is the conjugate of the function \( f \).

Fig. 1 shows the graph of the conjugate \( g \) (stored in the variable \( P \)) and the graph of the numerical approximation computed by the algorithm (stored in \( P \)). The graphs being close illustrates the accuracy of the algorithm.

The package has no trouble computing with extended valued functions which are common in convex analysis and optimization. To illustrate that point, we compute the biconjugate of the indicator function of an elliptic set. By definition, the indicator function is \( 0 \) if the point belongs to the set and \( +\infty \) otherwise.

```maple
f:=(x,y)->if type(x,numeric) and
    type(y,numeric) then
    if x^2+2*y^2<=1 then 0
    else infinity;fi;
else 'f'(x,y);fi:
g:=(x,y)->if type(x,numeric) and
    type(y,numeric) then
    if x^2+2*y^2<=1 then 0 else 10000;
    fi; else 'g'(x,y);fi:
P:=plot3d(g,-2..2,-3..3,color=red):
fs:=Conjugate(f,[-1..1,-2..2],
[30,30],[-10..10,-10..10],[30,30],
convex=false,outcompress=false):
Q1:=Conjugate(fs[2],fs[1],
[-2..2,-1..1],[20,20],output=plot,
convex=false,outcompress=false):
fs:=Conjugate(f,[-1..1,-2..2],
[20,20],[-9..9,-12..12],[10,10],
output=plot,outcompress=false):
Q2:=Conjugate(fs[2],fs[1],
[-2..2,-1..1],[20,20],output=plot,
convex=false,outcompress=false):
display({P,Q1,Q2},axes=framed,
view=-1..20,orientation=[51,54]);
```

We have to define a function \( F \) to plot the graph of \( f \) because Maple has trouble to plot functions taking the value \( +\infty \) (at least in Maple V R. 5). Fig. 2 shows...
The Legendre-Fenchel Conjugate: Numerical Computation

Figure 3: The biconjugate of a fourth degree polynomial

the two approximation of the biconjugate we computed wrap around the function. In other words, the larger the dual domain, the tighter the wrapping. That convergence behavior is illustrated again in the last section.

Getting information from the conjugate

Convexity

If you take the conjugate of \( f \) twice, you obtain the biconjugate \( f^{**} \) which is also the closed convex hull: it is the largest lower semi-continuous convex function smaller than \( f \). So computing the conjugate is a way of convexifying a function.

Here is an example with a simple polynomial.

```maple
> f:=x->(x^2-1)^2;
f := x -> (x^2 - 1)^2
> Domain(f,-infinity..infinity);
-∞..∞
```

Fig. 3 shows the computation is very accurate. Note that the conjugate can be computed explicitly for a fourth degree polynomial but the formula is complicated (it uses Cardan’s formula for obtaining the real root of a third degree polynomial).

Another example is shown in Fig. 2.

Compacity

Compacity can also be checked with the conjugate. Remember that compacity is very important in optimization problems since it is the usual way of proving the existence of an optimum: the infimum of a lower semi-continuous function on a compact set is always attained in that set. A nice property is then inf-compactness: all level sets

\[ \{ x \in \mathbb{R}^n : f(x) \leq r \} \]

are compact for all \( r \in \mathbb{R} \). Clearly if a function is inf-compact, it admits a minimum on all \( \mathbb{R}^n \). Using the conjugate, we can easily check for inf-compactness: \( f \) is inf-compact if and only if \( 0 \in \text{int} (\text{Dom} f) \).

Here are two examples of using the package to check compacity. In the first case, the function is not inf-compact while it is in the second case.

```maple
> g:=y->-y*ln(-y)+1+y:
g := y -> -y*ln(-y) + 1 + y
> Domain(g,-infinity..0);
-∞..0
```

The Domain procedure works only for univariate functions.

Smoothing/Regularizing

Finally, the conjugate allows to compute regularizations of a nonsmooth convex function. Define the infimal convolution of two functions \( f \) and \( g \) by

\[ f \Box g(x) := \inf_{y \in \mathbb{R}^n} [f(y) + g(x - y)]. \]

Then the Moreau–Yosida approximate of a convex lower semi-continuous function \( f \) is \( f \Box \| \cdot \|/2 \) and the Lipschitz regularization is \( f \Box \| \cdot \| \). The former is always convex continuously differentiable with Lipschitz gradient while the later is always Lipschitz. Both are useful to smooth \( f \) and extend it to the whole space.

First, consider smoothing the absolute value function.

```maple
> f:=x->abs(x):
f := x -> abs(x)
> P:=Moreau(f,[-1..1],[100],[-1..1],
> [100],outcompress=false,output=plot,
> color=red,thickness=3):
P := Moreau(f, -1..1, 100, -1..1, 100)
> Q:=plot(abs,-2..2,color=blue,
> style=line):
```

1998
The Legendre-Fenchel Conjugate: Numerical Computation

Figure 4: The Moreau regularization of the absolute value

```maple
plots[display]({P, Q}, axes=framed, view=[-2..2, 0..2]);
```

Fig. 4 shows the result.

Here is the same example in two dimensions:

```maple
f:=(x, y) -> abs(x)+abs(y):
P:=Moreau(f, [-1..1, -1..1], [40, 40], [-1..1, -1..1], [20, 20], output=plot, color=red):
Q:=plot3d(f, -2..2, -2..2, color=red, outcompress=false):
g:=(x, y) -> if type(x, numeric) and type(y, numeric) then if abs(x)<=1 then if abs(y)<=1 then 1/2*x^2+1/2*y^2 else 1/2*x^2+abs(y)-1/2;fi; else if abs(y)<=1 then abs(x)-1/2+1/2*y^2 else abs(x)-1/2+abs(y)-1/2;fi;fi:
Q1:=plot3d(g, -3..3, -3..3, color=green):
```

The result is shown in Fig. 5. The graph Q1 of the exact formula g of the Moreau-Yosida regularization is very close to the numerical approximation P.

The package deals as easily with Lipschitz regularization.

```maple
f:=(x, y) -> 1/2*x^2+1/2*y^2:
Q:=plot3d(f, -3..3, -3..3):
```

Figure 5: The Moreau regularization of a bivariate function

```maple
P:=Lipschitz(f, [-3..3, -3..3], [30, 30], [-1..1, -1..1], [20, 20], output=plot, color=red, outcompress=false):
g:=(x, y) -> if type(x, numeric) and type(y, numeric) then if abs(x)<=1 then if abs(y)<=1 then 1/2*x^2+1/2*y^2 else 1/2*x^2+abs(y)-1/2;fi; else if abs(y)<=1 then abs(x)-1/2+1/2*y^2 else abs(x)-1/2+abs(y)-1/2;fi;fi:
Q1:=plot3d(g, -3..3, -3..3, color=green):
```

The result is presented in Fig. 6. The regularization is sandwiched between the function Q in grey and the exact result Q1 in black.

For example, solving Hamilton–Jacobi equations can be done by using the Moreau procedure. Define

\[
\theta(x) = \begin{cases} 
1 - \sqrt{1-x^2} & \text{if } |x| \leq 1, \\
+\infty & \text{otherwise}.
\end{cases}
\]

\[
f(x) = \begin{cases} 
(x^2 - 1)^2 & \text{if } |x| > 1, \\
0 & \text{otherwise}.
\end{cases}
\]
The Legendre-Fenchel Conjugate: Numerical Computation

Figure 6: The Lipschitz regularization of a bivariate function

\[
F(x,t) = \begin{cases} (f \circ t \theta(t))(x) & \text{if } t > 0, \\ f(x) & \text{if } t = 0, \\ +\infty & \text{otherwise.} \end{cases}
\]

Then the function \( F \) (displayed in Fig. 7) satisfies the Hamilton–Jacobi equation:

\[
\frac{\partial F}{\partial t} + \theta^*(\frac{\partial F}{\partial x}) = 0, \quad \text{with } \lim_{t \to 0^+} F(\cdot, t) = f.
\]

So it can be computed by applying several times our algorithm. Fig. 7 shows the graph of \( F \).

A word on the algorithm

The goal of the algorithm is to compute a numerical approximation of the conjugate at a lot of points. First, the multidimensional case is reduced to several one-dimensional computations, next a convex hull algorithm is used as a preprocessing step, finally the conjugate is obtained in linear time by using the convex structure. The key idea to obtain a linear algorithm is to realize the planar convex hull can be computed in linear time as long as the coordinates of the points are sorted along one direction.

The algorithm works with discrete data i.e. the function is first sampled at \( N \) points, then the conjugate is computed at \( M \) slopes. The linear time property holds with respect to \( N + M \).

The improvement can easily be measured (it depends on the speed of your workstation).

Figure 7: The solution of a Hamilton–Jacobi equation

\[
> f:=x->\exp(x); \\
> t:=time():\text{Conjugate}(f,[-10..10],[100],\text{method=direct}):time()-t;
\]

158.058

\[
> t:=time():\text{Conjugate}(f,[-10..10],[100],\text{method=sequential}):time()-t;
\]

2.706

To obtain more accurate approximations, we should be aware of the two ways convergence occurs. First, we can fix the domains and increase the number of points. We obtain convergence as soon as the function is upper semi-continuous [2, 4]. The smoother the function, the faster the convergence.

Here is an example to illustrate that kind of convergence, the result is shown in Fig. 8.

\[
> f:=x->x^2/2;P1:=\text{Conjugate}(f,[-2..2],[10],\text{output=plot},\text{color=yellow}); \\
> P2:=\text{Conjugate}(f,[-2..2],[20],\text{output=plot},\text{color=green}); \\
> P3:=\text{Conjugate}(f,[-2..2],[30],\text{output=plot},\text{color=red}); \\
> Q:=\text{plot}(f,-2..2,\text{color=blue},\text{style=point}); \\
> \text{plots[display]}\{(P1,P2,P3,Q),\text{view=[-2..2,0..2],axes=framed});
\]

The second kind of convergence occurs when we really want the conjugate \( f^* \) of a function \( f \) and not \( (f + I_{[a,b]})^* \) which is what is computed. In that case, we should take \( a \to -\infty \) and \( b \to +\infty \). Fig. 9 is the result of the following lines.
The Legendre-Fenchel Conjugate: Numerical Computation

Figure 8: The more we sample, the better

\begin{verbatim}
f:=x->1/2*x^2:
P1:=Conjugate(f,[-1/2..1/2],[20],
[-10..10],[20],output=plot,
color=yellow):
P2:=Conjugate(f,[-1..1],
[-10..10],[20],output=plot,
color=green):
P3:=Conjugate(f,[-4..4],
[-10..10],[20],output=plot,
color=red):
Q:=plot(f,-2..2,color=blue,
style=point):
plots[display](\{P1,P2,P3,Q\},
view=[-2..2,0..2],axes=framed);
\end{verbatim}

Figure 9: The larger the domains, the closer to the conjugate in maple

> P:=Conjugate(f,[-4..4],[20],
[-10..10],[20],output=plot,
color=red):

Conclusion

We have illustrated the package LLT and some of its applications. It is intended to help convex analysis users to investigate the behavior of the conjugate (and of the functionals that reduce to conjugate computation) or to build quickly simple examples.

The author hopes the package will be useful in that way.

Acknowledgment

I am thankful to Jon Borwein for his offer to work at his centre (where I wrote this package), and to Greg Fee for helpful suggestions with Maple.

References

[1] H. H. BAUSCHKE AND M. VON MOHREN-SCHILDT, Fenchel conjugates and subdifferentials


**Biography**

**Yves Lucet**, Department of Mathematics and Statistics, University of Victoria, PO BOX 3045 STN CSC, Victoria BC, V8W 3P4, Canada

Email: lucet@na-net.ornl.gov

Research partly supported by the Pacific Institute for the Mathematical Sciences.