The Legendre-Fenchel Conjugate: Numerical Computation

Yves Lucet*

August 27, 1998

Abstract

We describe a numerical implementation of the Linear-time Legendre Transform algorithm in Maple V.5. After a brief motivation on the importance of the Legendre–Fenchel transform, we illustrate the convergence of the algorithm. Next, we show various examples of computation and go on to describe the options of the main functions. The last section report the known bugs.

All packages describe here are available on the world wide web at the Computational Convex Analysis project web page at http://www.cecm.sfu.ca/projects/CCA. In particular, the present package is available at http://www.cecm.sfu.ca/projects/CCA/LLT.

The present paper deals only with the *numerical* computation of the conjugate. We refer to [1] for symbolic computation (see also the corresponding web pages: http://www.cecm.sfu.ca/projects/CCA/FENCHEL).

The algorithm presented here has been studied in [4, 3] (see also [2] about the convergence).

Note the package runs ONLY in Maple V.5. It does NOT run in Maple V.4 since local variables do not exist for that version.

To use the package, type the command: read LLT; where the LLT file is on the current directory or in a directory in the Maple path

```
> restart:read LLTv0
```

We will also need some other packages for that demo:

```
> with(linalg):with(plots):
```

```
Warning, new definition for norm

Warning, new definition for trace
```

1 Motivations: what do we want to compute and why?

Given a function f, the Legendre-Fenchel conjugate is defined by:

$$s \to \sup(< s, x > -\mathrm{f}(x))$$

where the supremum is computed for any x, and the notation s.x is the scalar product of x and s. In order to do the computation, this worsheet describes a numerical algorithm which transforms the supremum in a maximum over a finite number of values. The convergence is obtained when taking more and more values.

*CECM, Dept. Math. and Stat., Simon Fraser University, Burnaby BC, V5A 1S6 Canada, lucet@cecm.sfu.ca

1.1 Importance of the Legendre-Fenchel conjugate

The Legendre-Fenchel conjugate (or conjugate in short) is fundamental in convex analysis. To emphasize that point and motivate the present package, we sum up its most important properties and applications.

1.1.1 Duality theory

Consider the following problems for 2 extended valued functions, f defined on R^n and g defined on R^m , and a linear operator A from R^n to R^m:

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

(Dual) d:=-inf[f*(- A^Tz)+g*(z)]

where A^T denotes the transpose of A, and f*, g* the conjugates of f and g.

Then *weak duality* always hold:

$$d \leq p$$

In addition, under some *constraint qualifications* (for example: the intersection of A Dom f with intDom g is nonempty) strong duality holds:

p=d with the infimum attained in the Dual problem.

Finally, Fenchel's duality theorem characterizes when both problem attains their infima:

x solves (Primal) and z solves (Dual)

if and only if

Ax belongs to the convex subdifferential of g* at z and -A*z belongs to the subdifferential of f at x

All in all, the conjugate allows one to build a dual problem which may be easier to solve than the original (primal) problem. Moreover, it gives conditions for both problems to have a solution. As such it is extremely useful.

1.1.2 Regularization

The Moreau-Yosida approximate of a convex function can be computed by 2 conjugate computations. The same is true of the Lipschitz regularization. Hence the LLT algorithm can be used to study regularization of nonsmooth convex functions.

1.2 Need for a numerical algorithm: why not do symbolic computation

A symbolic computation package exist for univariate functions. Its name is fenchel and is available at the CCA web page.

First, the conjugate involves the computationof a supremum. So there are functions for which the conjugate cannot be written with a combination of usual functions (for example computing the conjugate of a 6th-degree poynomial requires solving a 5th-degree polynomial equation).

Next, even if the computation is possible the result may be so complicated only Maple can use it. For example, consider the conjugate of

```
> f:=x->5*(x^2-1)^2
```

$$f := x \to 5\,(x^2-1)^2$$

Computing the conjugate symbolically requires solving a 3rd-degree polynomial equation (which is always possible symbolically). A little computation in Maple gives:

$$s\tilde{} \to s\tilde{}\,\mathrm{piecewise}(s\tilde{} < -\frac{40}{9}\sqrt{3},$$

$$\mathrm{signum}(\frac{1}{40}\,s\tilde{} + \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192})\left|\frac{1}{40}\,s\tilde{} + \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192}\right|^{(1/3)}$$

$$+\,\mathrm{signum}(\frac{1}{40}\,s\tilde{} - \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192})\left|\frac{1}{40}\,s\tilde{} - \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192}\right|^{(1/3)},$$

$$s\tilde{} < \frac{40}{9}\sqrt{3},\ 2\,\Re(\mathrm{surd}(\frac{1}{40}\,s\tilde{} + \frac{1}{72}\,I\sqrt{-\frac{81}{25}\,s\tilde{}^{2} + 192},\ 3)),\ \frac{40}{9}\sqrt{3} < s\tilde{},$$

$$\mathrm{signum}(\frac{1}{40}\,s\tilde{} + \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192})\left|\frac{1}{40}\,s\tilde{} + \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192}\right|^{(1/3)}$$

$$+\,\mathrm{signum}(\frac{1}{40}\,s\tilde{} - \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192})\left|\frac{1}{40}\,s\tilde{} - \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192}\right|^{(1/3)}) - 5$$

$$\mathrm{piecewise}(s\tilde{} < -\frac{40}{9}\sqrt{3},$$

$$\mathrm{signum}(\frac{1}{40}\,s\tilde{} + \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192})\left|\frac{1}{40}\,s\tilde{} + \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192}\right|^{(1/3)}$$

$$+\,\mathrm{signum}(\frac{1}{40}\,s\tilde{} - \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192})\left|\frac{1}{40}\,s\tilde{} - \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192}\right|^{(1/3)},$$

$$s\tilde{} < \frac{40}{9}\sqrt{3},\ 2\,\Re(\mathrm{surd}(\frac{1}{40}\,s\tilde{} + \frac{1}{72}\,I\sqrt{-\frac{81}{25}\,s\tilde{}^{2} + 192},\ 3)),\ \frac{40}{9}\sqrt{3} < s\tilde{},$$

$$\mathrm{signum}(\frac{1}{40}\,s\tilde{} + \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192})\left|\frac{1}{40}\,s\tilde{} + \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192}\right|^{(1/3)}$$

$$+\,\mathrm{signum}(\frac{1}{40}\,s\tilde{} - \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192})\left|\frac{1}{40}\,s\tilde{} - \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192}\right|^{(1/3)})^4 + 10$$

$$\mathrm{piecewise}(s\tilde{} < -\frac{40}{9}\sqrt{3},$$

$$\mathrm{signum}(\frac{1}{40}\,s\tilde{} + \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192})\left|\frac{1}{40}\,s\tilde{} + \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192}\right|^{(1/3)}$$

$$+\,\mathrm{signum}(\frac{1}{40}\,s\tilde{} - \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192})\left|\frac{1}{40}\,s\tilde{} - \frac{1}{72}\sqrt{\frac{81}{25}\,s\tilde{}^{2} - 192}\right|^{(1/3)},$$

$$s^{\sim} < \frac{40}{9}\sqrt{3},\ 2\,\Re(\mathrm{surd}(\frac{1}{40}\,s^{\sim} + \frac{1}{72}\,I\,\sqrt{-\frac{81}{25}\,s^{\sim 2} + 192},\ 3)),\ \frac{40}{9}\sqrt{3} < s^{\sim},$$

$$\mathrm{signum}(\frac{1}{40}\,s^{\sim} + \frac{1}{72}\sqrt{\frac{81}{25}\,s^{\sim 2} - 192})\,\left|\frac{1}{40}\,s^{\sim} + \frac{1}{72}\sqrt{\frac{81}{25}\,s^{\sim 2} - 192}\right|^{(1/3)}$$

$$+\,\mathrm{signum}(\frac{1}{40}\,s^{\sim} - \frac{1}{72}\sqrt{\frac{81}{25}\,s^{\sim 2} - 192})\,\left|\frac{1}{40}\,s^{\sim} - \frac{1}{72}\sqrt{\frac{81}{25}\,s^{\sim 2} - 192}\right|^{(1/3)}\,)^2 - 5$$

1.3 Speed of convergence

The algorithm reduces multidimensional computation to several one-dimensional computations. So it is enough to look at its convergence for univariate functions. Assume f is twice continuously differentiable on the neigborhood of an interval [a,b]. Then Taylor expansion theorems gives the following upper bound between the discrete Legendre Transform (which we compute) and the conjugate:

$$C\frac{(b-a)^2}{2N^2}$$

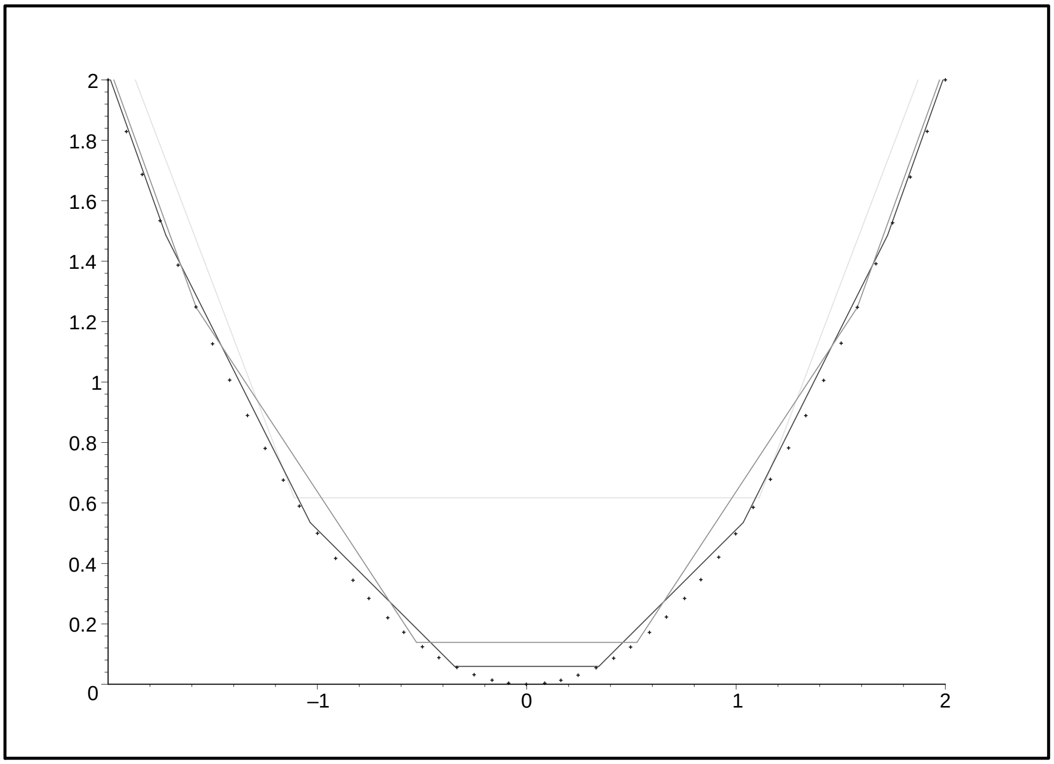
where C is the supremum of the second derivative on [a,b] and N is the number of points you take in [a,b].

Consequently, first one has to take a large enough interval [a,b], and next to sample the function enough (i.e. take N large) to obtain a good numerical accuracy.

1.3.1 Example of convergence when [a,b] is fixed and N is larger and larger

f is its own conjugate: f*=f

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



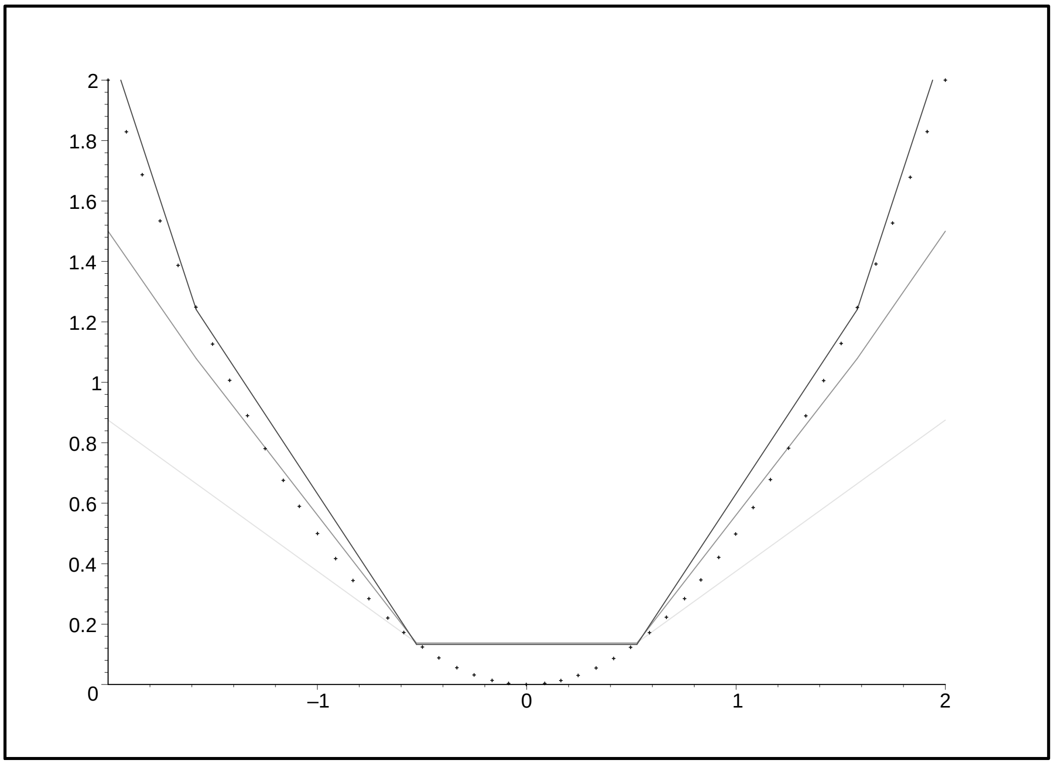
Because we compute the conjugate of f plus the indicator function of [a,b], i.e. we assume f is infinity outside of [a.b], the conjugate is the inf-convolution of f* with the support function of [a,b]. So it is extended linearly on all the real line. To reduce that effect, we should be careful to take the dual domain such that the maximum is attained in [a,b] for all the slopes in the dual domain. In particular, if f is convex, we should take the dual domain in [f'(a),f'(b)].

1.3.2 Example of convergence when [a,b] is taken larger and larger

f is its own conjugate: f*=f

```
> 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],[20],[-10..10],[20],output=plot,color=green):
```

```
> P3:=Conjugate(f,[-4..4],[20],[-10..10],[20],output=plot,color=red):
> Q:=plot(f,-2..2,color=blue,style=point):
> display({P1,P2,P3,Q },view=[-2..2,0..2],axes=framed)
```

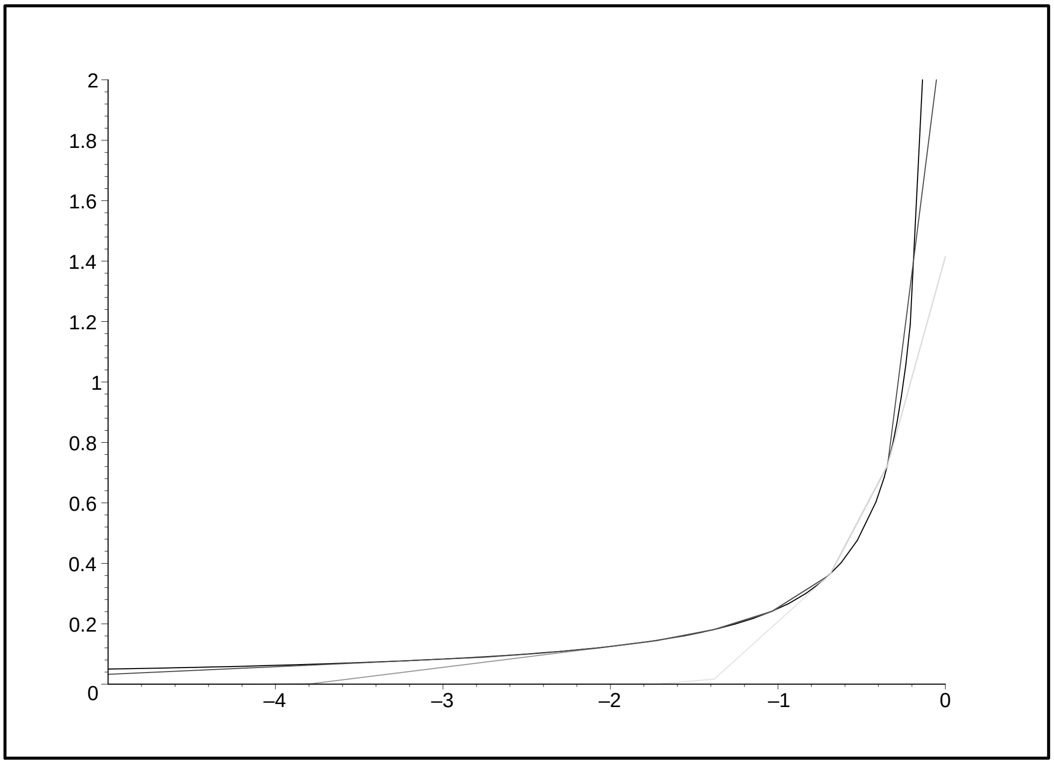


When the primal function has an infinite slope at one point, one should expect numerical inaccurracy around that point. The solution is to take more points in the primal domain (or to take them closer to the point with infinite slope). Moreover, one has to take larger and larger dual domain since the dual domain is infinite in that case.

1.3.3 Example of numerical problem around an infinite slope

```
> f:=x->-sqrt(x):
```

```
> P1:=Conjugate(f,[0..2],[5],[-10..0],[30],output=plot,color=yellow):
> P2:=Conjugate(f,[0..2],[30],[-10..0],[30],output=plot,color=green):
> P3:=Conjugate(f,[0..5],[200],[-10..0],[30],output=plot,color=red):
> Q:=plot(-1/(4*x),x=-5..0,color=black):
> display({P1,P2,P3,Q },axes=framed,view=[-5..0,0..2])
```



2 A tour of the LLT package

2.1 The conjugate of a univariate function

```
> f:=x->abs(x)^3/3g:=s->abs(s)^(3/2)/(3/2)
```

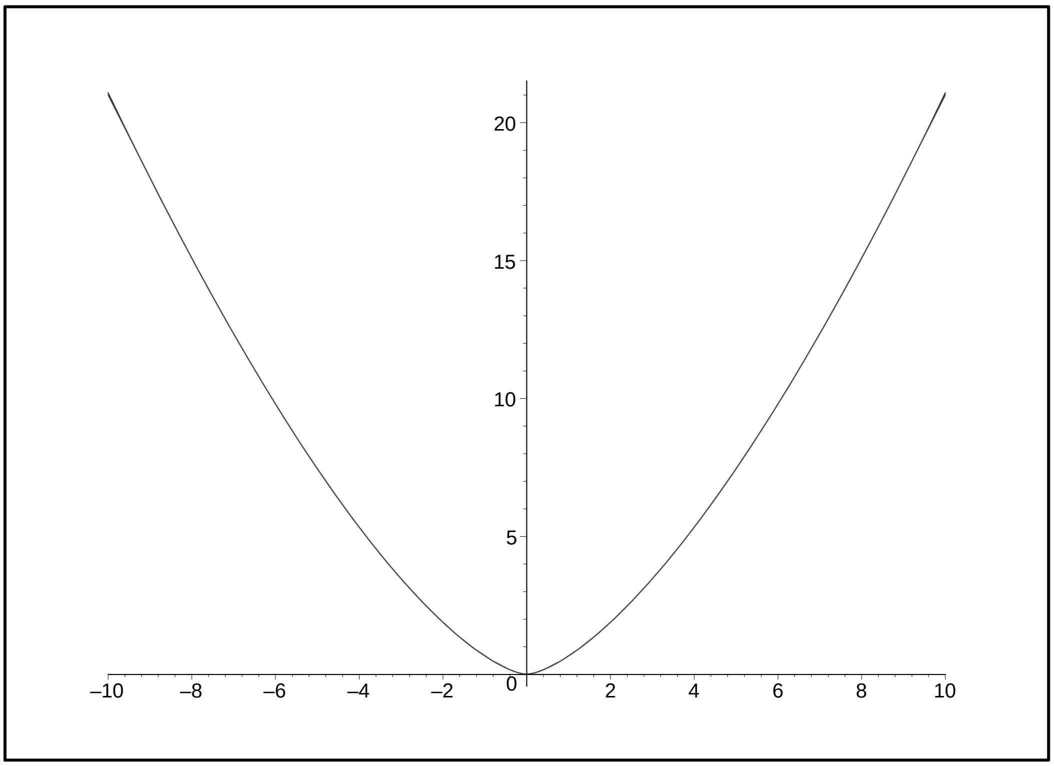
$$f := x \to \frac{1}{3}\,|x|^3$$

$$g := s \to \frac{2}{3}\,|s|^{(3/2)}$$

```
> P:=Conjugate(f,[-3..3],[100],output=plot):
> Q:=plot(g,-10..10,color=blue):
```

There is no difference between the 2 curves: the numerical computation is accurate enough

```
> display({P,Q})
```



2.2 he conjugate of a quadratic bivariate function of a symetric positive definite matrix

```
> A:=matrix(2,2,[[2,1],[1,3]])B:=inverse(A)
```

$$A := \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$$

$$B := \begin{bmatrix} \frac{3}{5} & \frac{-1}{5} \\ \frac{-1}{5} & \frac{2}{5} \end{bmatrix}$$

```
> 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
```

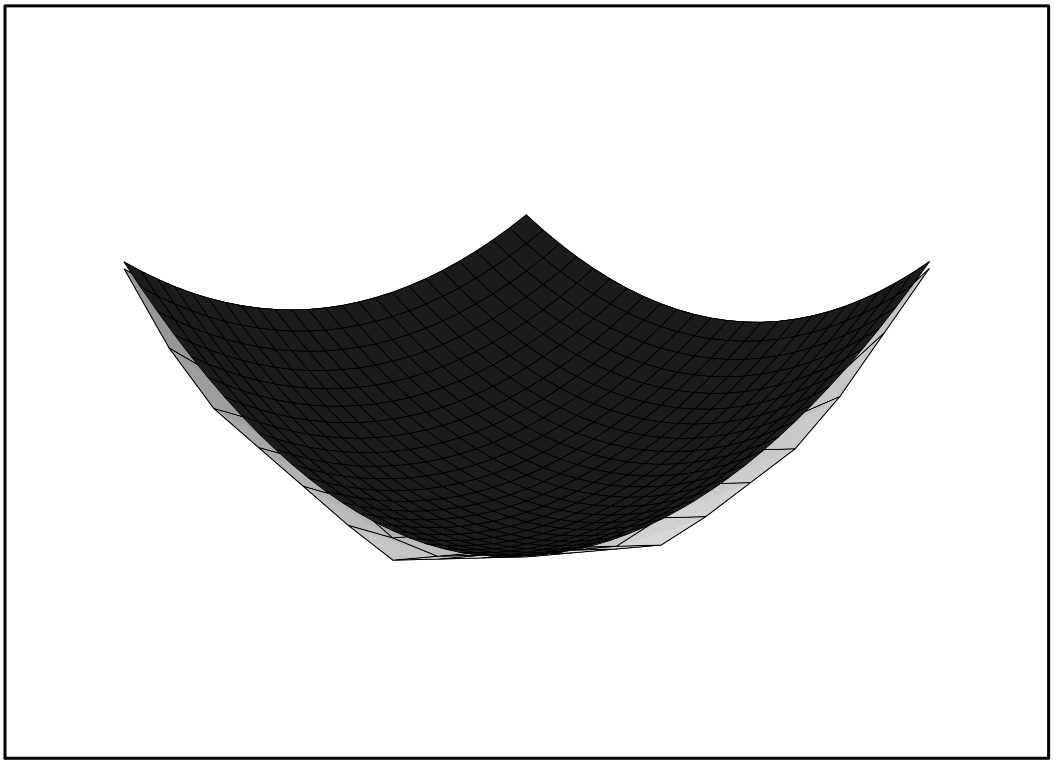
$$f := \mathbf{proc}(x,\, y)\, \mathbf{local}\, v;\; v := \mathrm{array}([x,\, y])\,;\; 1/2 \times \mathrm{multiply}(\mathrm{transpose}(v),\, \mathrm{multiply}(A,\, v))\, \mathbf{end}$$

$$g := \mathbf{proc}(x,\, y)\, \mathbf{local}\, v;\; v := \mathrm{array}([x,\, y])\,;\; 1/2 \times \mathrm{multiply}(\mathrm{transpose}(v),\, \mathrm{multiply}(B,\, v))\, \mathbf{end}$$

```
> 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,color=blue):
```

An accurate approximation of the conjugate is obtained.

```
> display({P,Q})
```



2.3 The biconjugate of the indicator function wraps around the original function

2.3.1 The indicator of a boxed set

The number 10000 should be infinity but Maple crashes when plotting with infinity.

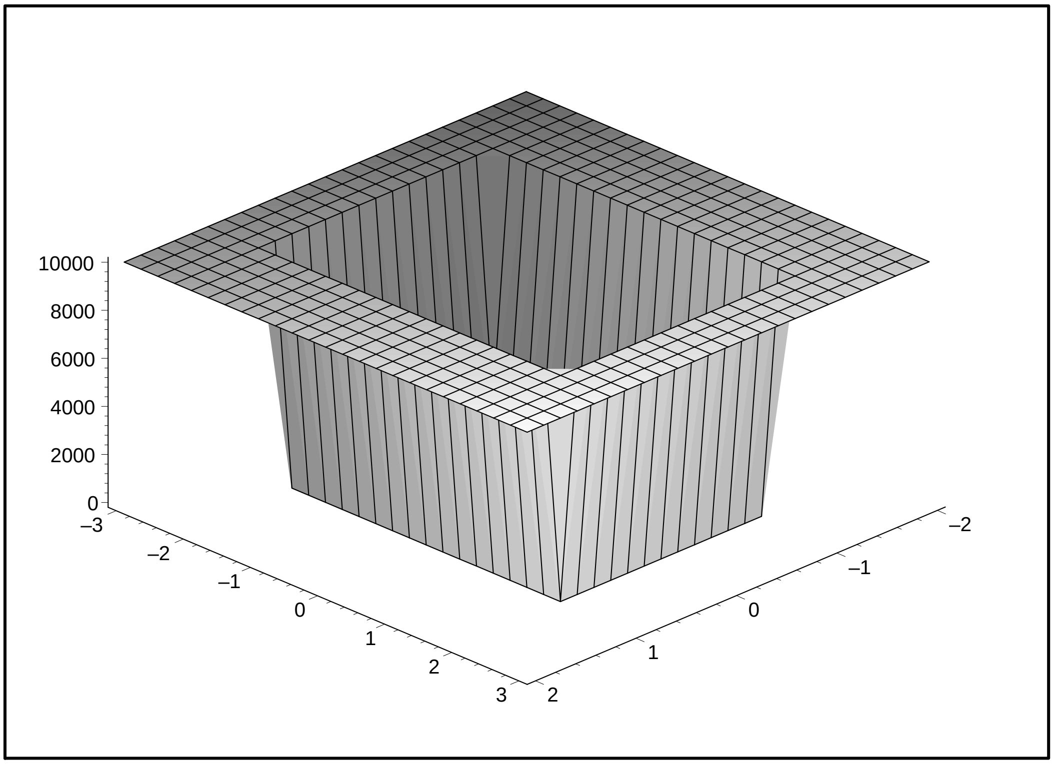
> `f:=(x,y)->if type(x,numeric) and type(y,numeric) then if abs(x)<=1 and abs(y)<=2 then 0 else infinityfi else 'f'(x,y)fi`

> `g:=(x,y)->if type(x,numeric) and type(y,numeric) then if abs(x)<=1 and abs(y)<=2 then 0 else 10000fi else 'g'(x,y)fi:`

f is understood as an indicator function: there should be no upper level but since Maple does not know how to display indicator functions, we gave an arbitrary large value instead of infinity

> `P:=plot3d(g,-2..2,-3..3):display(P,axes=framed)`

$f := \mathbf{proc}(x, y)$
$\quad \mathbf{option}\ \mathit{operator},\ \mathit{arrow};$
$\quad\quad \mathbf{if}\ \mathrm{type}(x,\ \mathit{numeric})\ \mathbf{and}\ \mathrm{type}(y,\ \mathit{numeric})\ \mathbf{then}$
$\quad\quad\quad \mathbf{if}\ \mathrm{abs}(x) \leq 1\ \mathbf{and}\ \mathrm{abs}(y) \leq 2\ \mathbf{then}\ 0\ \mathbf{else}\ \infty\ \mathbf{fi}$
$\quad\quad \mathbf{else}\ 'f'(x, y)$
$\quad\quad \mathbf{fi}$
$\quad \mathbf{end}$



Since the function f is the indicator of a boxed set, we can take into account that it is convex on its domain and use the outcompress option safely. Note how the outcompress option reduces the number of output point from 900 to 16.

```
> fs:=Conjugate(f,[-1..1,-2..2],[30,30],[-10..10,-10..10],[30,30],\
>  output=matrix,convex=true)
```

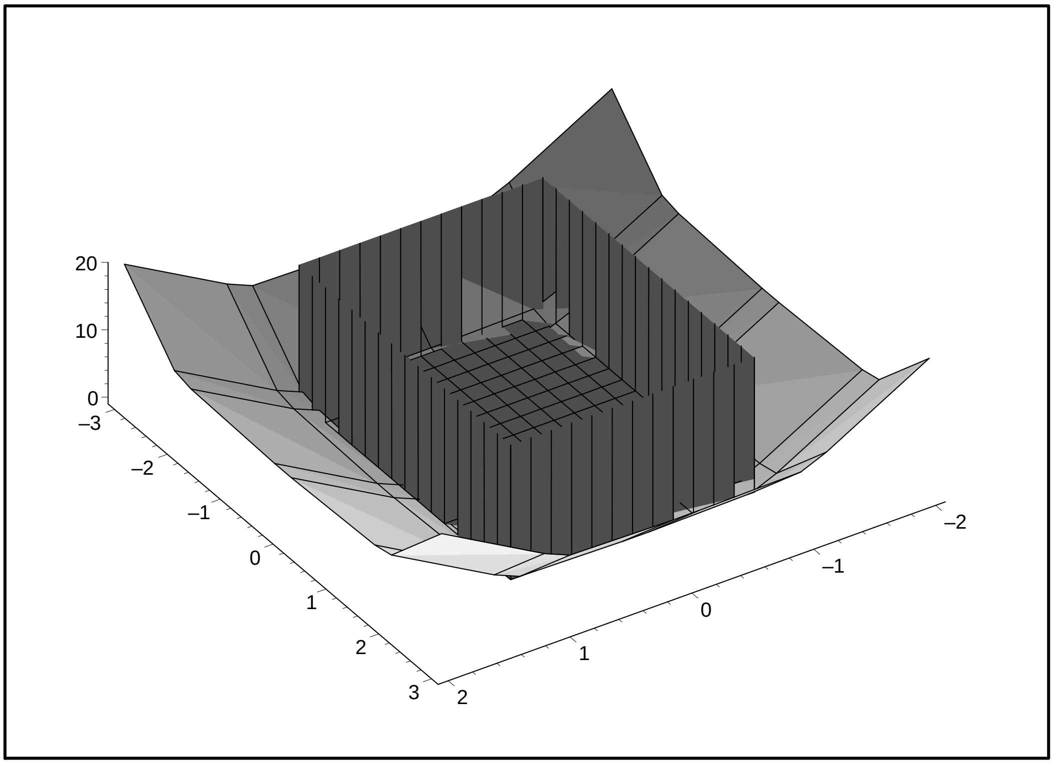
$$fs := \left[[[-10., -.3448275862, .3448275862, 10.], [-10., -.3448275862, .3448275862, 10.]], \begin{bmatrix} 30. & 10.68965517 & 10.68965517 & 30. \\ 20.34482759 & 1.034482759 & 1.034482759 & 20.34482759 \\ 20.34482759 & 1.034482759 & 1.034482759 & 20.34482759 \\ 30. & 10.68965517 & 10.68965517 & 30. \end{bmatrix} \right]$$

We now compute the biconjugate which is also the convex hull

```
> Q:=Conjugate(fs[2],fs[1],[-2..2,-3..3],[20,20],output=plot,convex=true):
> P:=plot3d(g,-2..2,-3..3,color=red):
```

The biconjugate wraps around the original function. Taking larger values in the dual domain will make the wrapping tighter.

```
> display({P,Q },axes=framed,view=-1..20,orientation=[57,23])
```



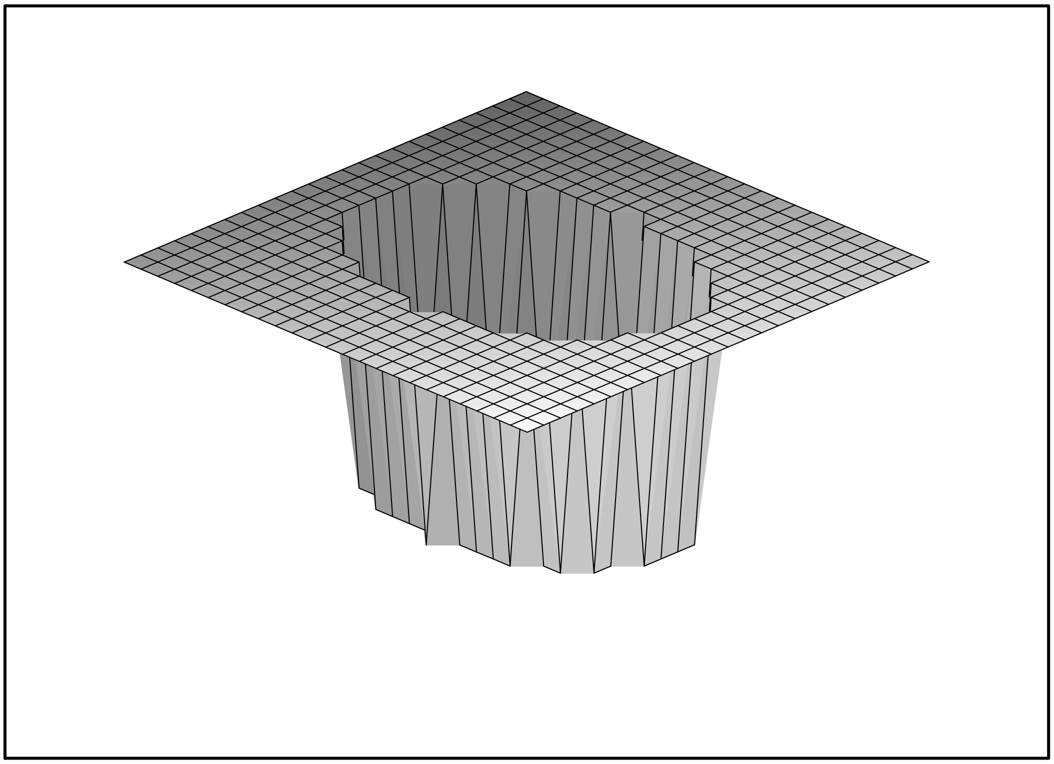
2.3.2 The indicator of a nonboxed set

f is the indicator function of an elliptic set. Again we have to be careful not to plot f.

```
> f:=(x,y)->if type(x,numeric) and type(y,numeric) then if x^2+2*y^2<=1 then 0 \
> else infinityfi 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 10000fi else 'g'(x,y)fi:
> plot3d(g(x,y),x=-2..2,y=-1..1)
```

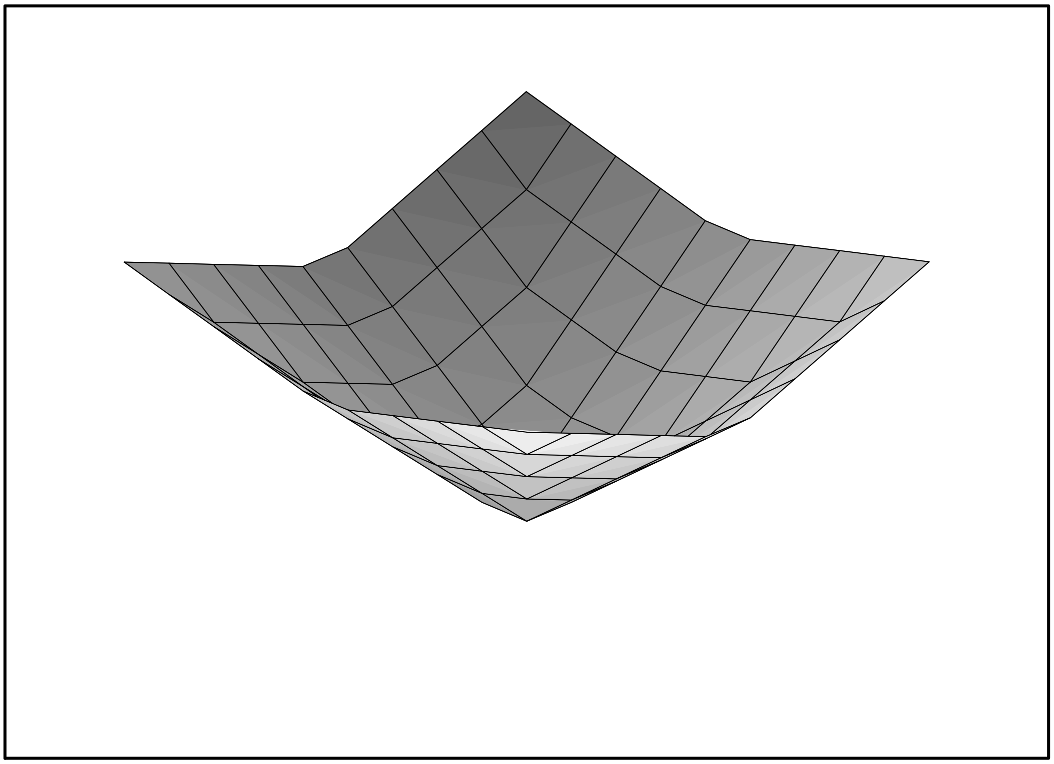
$f := \mathbf{proc}(x,\, y)$
$\quad \mathbf{option}\ \mathit{operator},\ \mathit{arrow};$

if type(x, *numeric*) **and** type(y, *numeric*) **then if** $x^2 + 2 \times y^2 \leq 1$ **then** 0 **else** ∞ **fi**
else 'f'(x, y)
fi
end



The conjugate is the support function of the set:

```
> Conjugate(f,[-1..1,-2..2],[10,10],[-10..10,-10..10],[10,10],
>  output=plot,convex=false,outcompress=false)
```

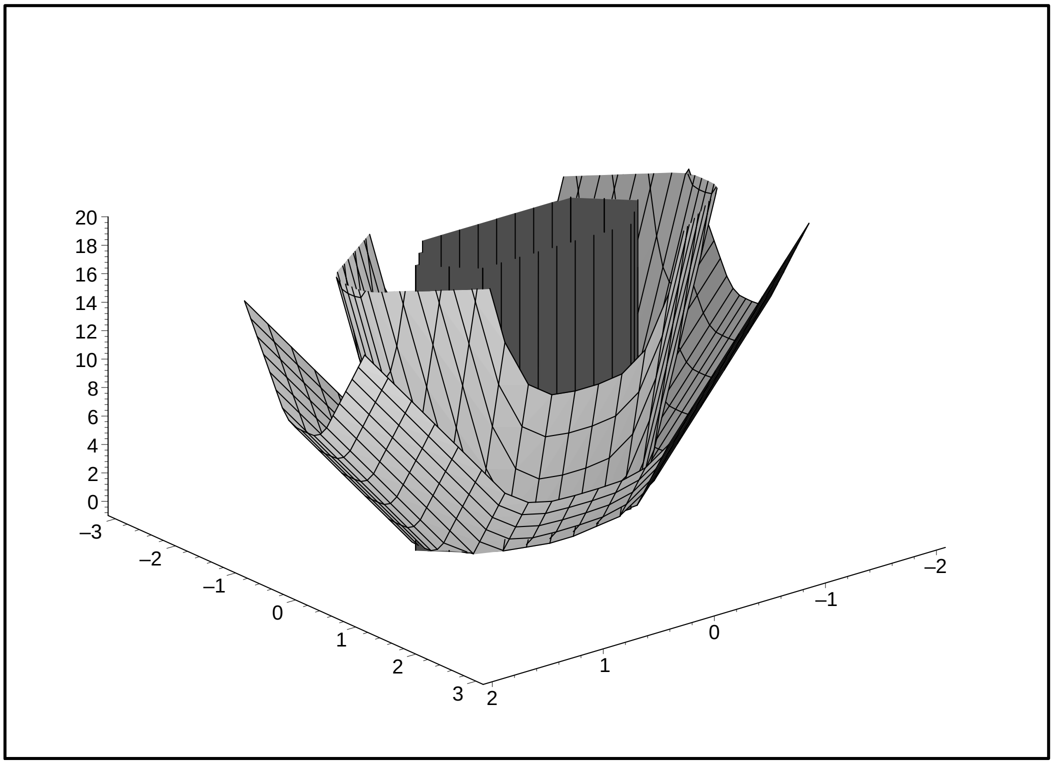


Now we compute the biconjugate. Note that the larger we take the dual domain the more accurate the convex hull. Note also that we have to use the convex = false and outcompress = false options to obtain the right result.

```
> 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],[30,30],[-30..30,-30..30],[30,30],
>  convex=false,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])
```

That computation may take 30s of CPU time, depending on your computer



2.4 The conjugate of the exponential

```
> f:=x->-1+exp(-x)
> g:=y->if type(y,numeric) then if y<0 then -y*ln(-y)+1+y
> elif y=0 then 1 else infinityfielse 'g'(y)fi
```

$$f := x \to -1 + e^{(-x)}$$

$g := \textbf{proc}(y)$
    option *operator*, *arrow*;
        $\textbf{if}\,\mathrm{type}(y,\ \mathit{numeric})\,\textbf{then}\ \textbf{if}\,y < 0\,\textbf{then}\ -y \times \ln(-y) + 1 + y\ \textbf{elif}\,y = 0\,\textbf{then}\,1\,\textbf{else}\,\infty\,\textbf{fi}$
        else 'g'(y)
        fi
    end

The domain of the conjugate is the set of points where it is finite. It can be computed with the Domain function which uses the Maple limit function.

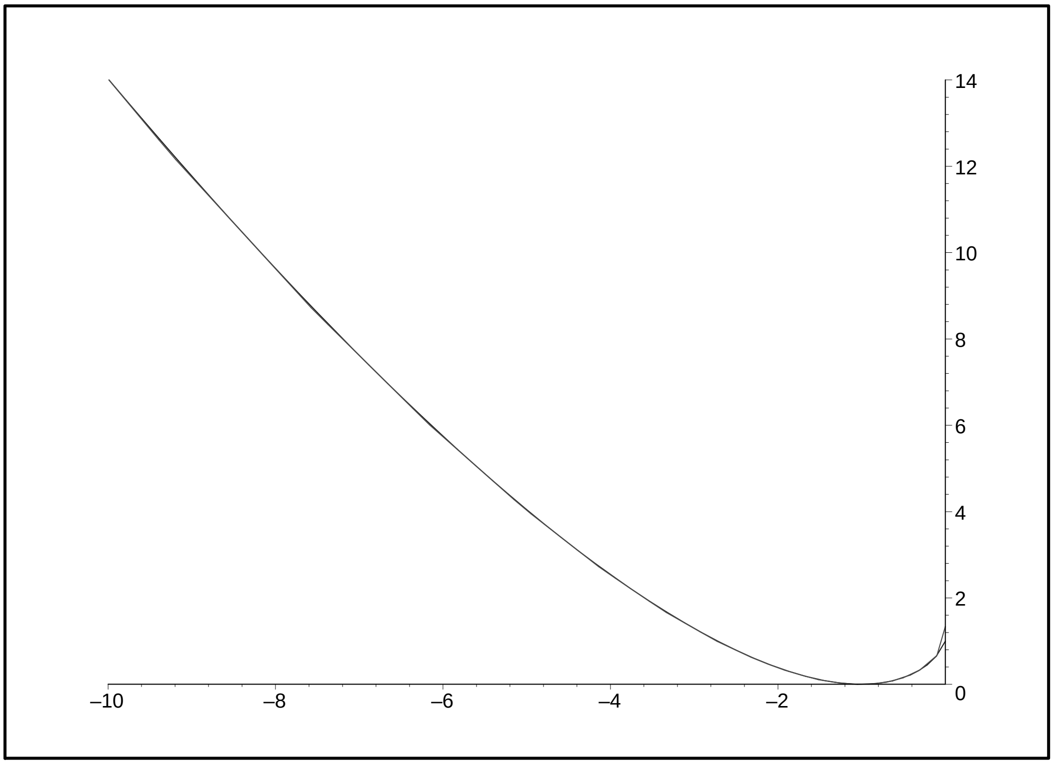
```
> Domain(f,-infinity..infinity)
```

$$-\infty..0$$

```
> P:=Conjugate(f,[-10..10],[100],output=plot):
> P:=display(eval(P),view=0..14):
> Q:=plot(g(y),y=-10..0,color=blue):
```

We obtain an accurate approximation of the conjugate with only 29 points (the set of 100 slopes was compressed to 29 by removing affine parts)

```
> display({P,Q})
```



3 The functions available in the LLT package

3.1 The main function: Conjugate

The Conjugate function compute an approximation of the Legendre-Fenchel conjugate. More precisely, it takes as input some function on various forms and gives as ouput a (discrete) approximation of the conjugate. The algorithm used is the Linear-time Legendre Transform. It has a linear worst-case time complexity, and converges as soon as the original function is upper semicontinuous. The smoother the function, the faster the convergence.

3.1.1 The various input forms

3.1.2 The (primal) function

The primal function can be entered as a function. It is then discretized on its domain before the algorithm is applied.

```
> Conjugate(x->exp(x),[-10..10],[10])
```

$$[[[-10., -1.111111111, 1.111111111, 3.333333333, 10.]], [99.99995460, 11.11106571, -1.563760889, .665971926, 8.073379333]]$$

One can enter multidimensional functions. Here is an example in 4 variables. The only limit is the computation time which increases linearly with the number of points.

```
> f:=(x,y,z,u)->x^2+y^2+z^2+u^2:
> Conjugate(f,[1..5,1..5,1..5,1..5],[3,3,3,3],[1..3,1..3,1..3,1..3],\
> [3,3 ,3,3],convex=false,outcompress=true,output=matrix,method=sequential)
```

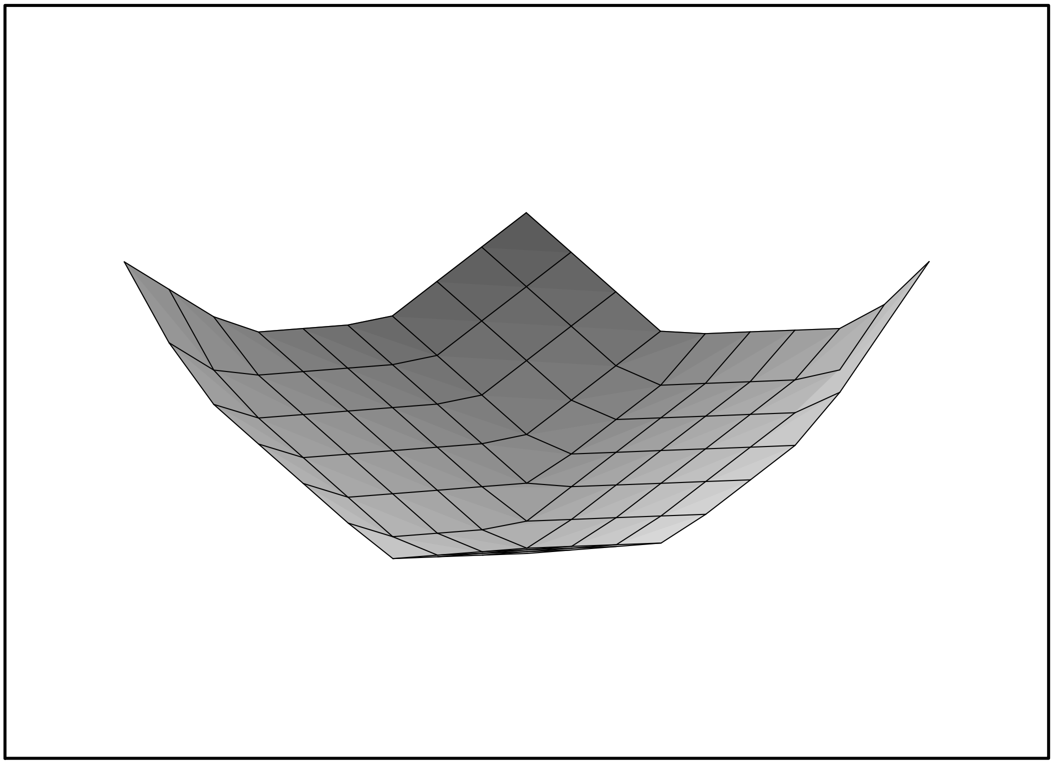
$$[[[1., 3.], [1., 3.], [1., 3.], [1., 3.]], \mathrm{array}(1..2, 1..2, 1..2, 1..2, [$$
$$(1, 1, 1, 1) = 0$$
$$(1, 1, 1, 2) = 2.$$
$$(1, 1, 2, 1) = 2.$$
$$(1, 1, 2, 2) = 4.$$
$$(1, 2, 1, 1) = 2.$$
$$(1, 2, 1, 2) = 4.$$
$$(1, 2, 2, 1) = 4.$$
$$(1, 2, 2, 2) = 6.$$
$$(2, 1, 1, 1) = 2.$$
$$(2, 1, 1, 2) = 4.$$
$$(2, 1, 2, 1) = 4.$$
$$(2, 1, 2, 2) = 6.$$
$$(2, 2, 1, 1) = 4.$$
$$(2, 2, 1, 2) = 6.$$
$$(2, 2, 2, 1) = 6.$$
$$(2, 2, 2, 2) = 8.$$
$$])]$$

It can also be a procedure

```
> A:=matrix(2,2,[[2,1],[1,3]])
> f:=proc(x,y) local v v:=array([x,y]) 1/2*multiply(transpose(v),multiply(A,v))end
> Conjugate(f,[-50..50,-50..50],[20,20],[-9..9,-12..12],[10,10],output=plot,
> ,outcompress=false)
```

$$A := \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$$

$f := \mathbf{proc}(x,\, y)\, \mathbf{local}\, v;\ v := \mathrm{array}([x,\, y])\,;\ 1/2 \times \mathrm{multiply}(\mathrm{transpose}(v),\, \mathrm{multiply}(A,\, v))\, \mathbf{end}$



It can also be entered as an array of values (but not as a list).

```
> Conjugate(array([2,1,0,1,2]),[array([-2,-1,0,1,2])],[array([-2,-1,0,1,2])])
```

$$[[[-2,\, -1,\, 1,\, 2]],\, [2,\, 0,\, 0,\, 2]]$$

The multidimensional case consists of an array of values:

```
> Conjugate(array([[2,1,0,1,2],[2,1,0,1,2],[2,1,0,1,2],[2,1,0,1,2],[2,1,0,1,2]]),\
> [array([-2,-1,0,1,2]),array([-2,-1,0,1,2])],[array([-2,-1,0,1,2]),
> array([-2,-1,0,1,2])])
```

$$\left[[[-2, 0, 2], [-2, -1, 1, 2]], \begin{bmatrix} 6 & 4 & 4 & 6 \\ 2 & 0 & 0 & 2 \\ 6 & 4 & 4 & 6 \end{bmatrix} \right]$$

3.1.3 The primal domain

The primal domain can be entered as a list of range. In that case you have to add the number of points as a list of positive integers. In the next example, the function exp is sampled at 10 points regularly spaced on [-10..10]

```
> Conjugate(x->exp(x),[-10..10],[10])
```

$$[[[-10., -1.111111111, 1.111111111, 3.333333333, 10.]],$$
$$[99.99995460, 11.11106571, -1.563760889, .665971926, 8.073379333]]$$

It can also be a list of array of values at which the function is considered. That allows to sample more efficiently around points with infinite slope and thus takle numerical difficulties.

```
> Conjugate(abs,[array([-2,-1,0,1,2])],[array([-2,-1,0,1,2])])
```

$$[[[-2, -1, 0, 1, 2]], [2., 0, 0, 0, 2.]]$$

3.1.4 The dual domain

As for the primal domain, the dual domain can be entered as a list of ranges followed by a list of the number of points in each range

```
> Conjugate((x,y)->exp(x)+exp(y),[-10..10,-10..10],[10,10],[-5..0,-3..1],[4,7])
```

$$[[[-5., 0], [-3., -.3333333333, .3333333333, 1.]],$$
$$\begin{bmatrix} 79.99990920 & 53.33324253 & 49.30039124 & 48.55965050 \\ 29.99990920 & 3.333242533 & -.6996087580 & -1.440349499 \end{bmatrix}]$$

If your function is univariate, the dual domain can be computed using limits with the Domain function. So you can avoid entering it in that case.

```
> f:=x->-1+exp(-x)
> g:=y->if type(y,numeric) then if y<0 then -y*ln(-y)+1+y \
> elif y=0 then 1 else infinityfielse 'g'(y)fi
> Domain(f,-infinity..infinity)
> Conjugate(f,[-10..10],[10])
```

$$f := x \to -1 + e^{(-x)}$$

$g := \textbf{proc}(y)$
  $\textbf{option}\ operator,\ arrow;$
    $\textbf{if}\,\mathrm{type}(y,\ numeric)\,\textbf{then}\,\textbf{if}\,y < 0\,\textbf{then}\ -y \times \ln(-y) + 1 + y\,\textbf{elif}\,y = 0\,\textbf{then}\,1\,\textbf{else}\,\infty\,\textbf{fi}$
    $\textbf{else}\,\text{'}g\text{'}(y)$
    $\textbf{fi}$
  $\textbf{end}$

$$-\infty..0$$

$$[[[-10., -3.333333333, -1.111111111, 1.111111111, 10.]],\ [9.073379333, 1.665971926, -.5637608888, 12.11106571, 100.9999546]]$$

The dual domain can also be a list of array. Remenber that it MUST be a boxed domain to use the linear-time algorithm.

```
> Conjugate(abs,[array([-2,-1,0,1,2])],[array([-2,-1,0,1,2])])
```

$$[[[-2, -1, 0, 1, 2]], [2., 0, 0, 0, 2.]]$$

3.1.5 The various Options of Conjugate

3.1.6 The convex option, default: convex=false

The convex option is useful to speed up computations: the algorithm does not have to compute any convex hull and directly compute the conjugate, assuming the original function is convex

```
> f:=x->exp(x):
> t:=time():Conjugate(f,[-10..10],[1000],convex=true,outcompress=false):time()-t
> t:=time():Conjugate(f,[-10..10],[1000],convex=false,outcompress=false):time()-t
```

$$26.095$$

$$34.547$$

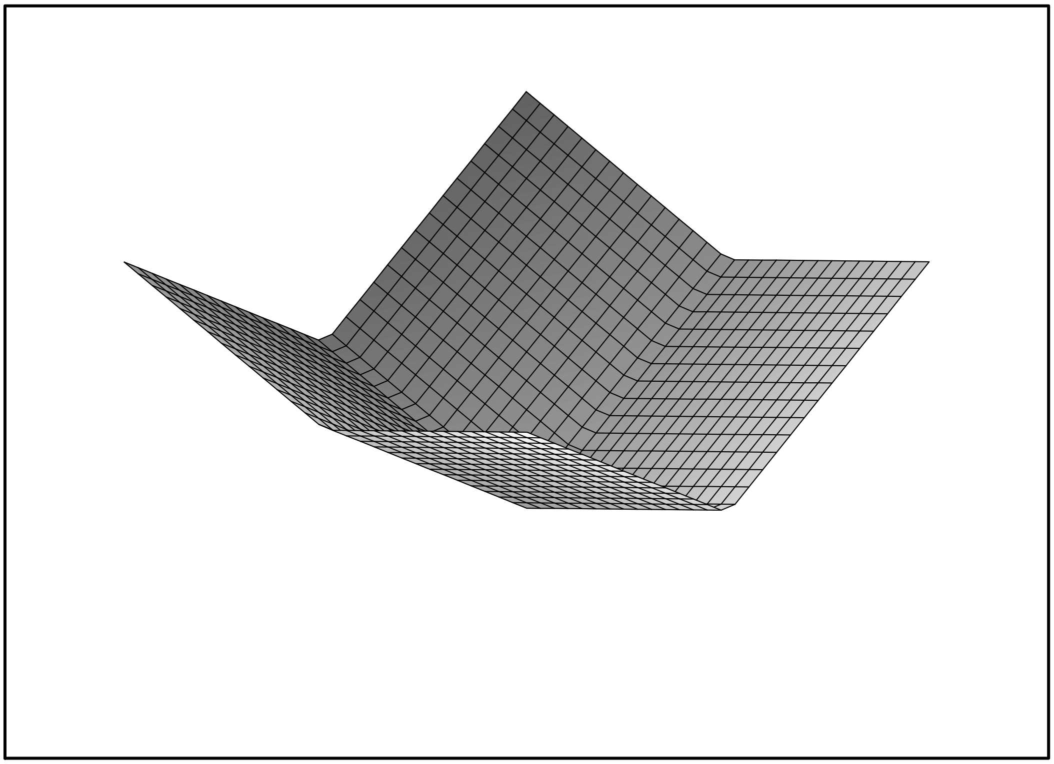
3.1.7 The outcompress option, default: outcompress=true

```
> f:=(x,y)->if type(x,numeric) and type(y,numeric) then if x^2+2*y^2<=1 then 0
> else 10000fi else 'f'(x,y)fi
```

No output compression is done: you obtain 900 output points

```
> Conjugate(f,[-2..2,-3..3],[10,10],[-10..10,-10..10],[30,30],
> output=plot,outcompress=false)
```

$f := \textbf{proc}(x,\ y)$
  $\textbf{option}\ operator,\ arrow;$
    $\textbf{if}\,\mathrm{type}(x,\ numeric)\ \textbf{and}\ \mathrm{type}(y,\ numeric)\,\textbf{then}$
      $\textbf{if}\,x^2 + 2 \times y^2 \leq 1\,\textbf{then}\,0\,\textbf{else}\,10000\,\textbf{fi}$
    $\textbf{else}\,\text{'}f\text{'}(x,\ y)$
    $\textbf{fi}$
  $\textbf{end}$

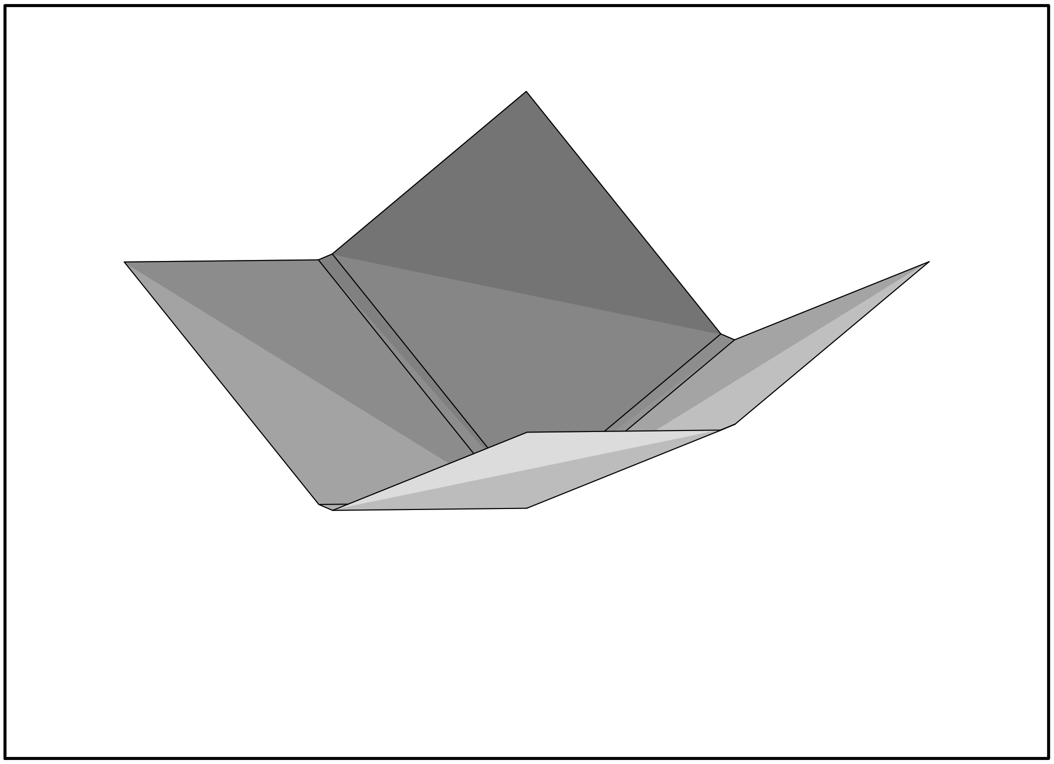


Output compression is done: all affine parts are removed. You obtain a 4 by 4 matrix, i.e. only 16 points. All the other points are on affine parts and so carry no information.

```
> Conjugate(f,[-2..2,-3..3],[10,10],[-10..10,-10..10],[30,30],outcompress=true)
> Conjugate(f,[-1..1,-2..2],[10,10],[-10..10,-10..10],[30,30],output=plot,convex=true)
```

$$\Big[[[-10., -.3448275862, .3448275862, 10.], [-10., -.3448275862, .3448275862, 10.]],$$

$$\left[\begin{matrix} 10.00000000 & 6.781609196 & 6.781609196 & 10.00000000 \\ 3.563218391 & .3448275862 & .3448275862 & 3.563218391 \\ 3.563218391 & .3448275862 & .3448275862 & 3.563218391 \\ 10.00000000 & 6.781609196 & 6.781609196 & 10.00000000 \end{matrix}\right]\Bigg]$$



3.1.8 The output option, default: output=matrix

The output option specifies in what format the output should be. There is a choice among 4 possibilities: matrix, points, procedure, plot

3.1.9 output = matrix

```
> Conjugate((x,y)->1/2*(x^2+y^2),[-2..2,-2..2],[10,10],[-2..2,-2..2],[4,4],output=matrix)
```

The result is a list. The first element is a list of arrays which correspond to the dual domain as a cartesian product. This is useful since the dual domain can change with the outcompress option. The second element is a matrix storing the values of the conjugate. For example, from the data below, the conjugate has value 4 at point (-2,-2) and value .44444 at point (-.6666,-.6666).

$$\left[[[-2., -.6666666667, .6666666667, 2.], [-2., -.6666666667, .6666666667, 2.]], \right.$$
$$\left. \begin{bmatrix} 4. & 2.222222222 & 2.222222222 & 4. \\ 2.222222223 & .4444444444 & .4444444444 & 2.222222223 \\ 2.222222223 & .4444444444 & .4444444444 & 2.222222223 \\ 4. & 2.222222222 & 2.222222222 & 4. \end{bmatrix} \right]$$

3.1.10 output = points

```
> Conjugate((x,y)->1/2*(x^2+y^2),[-2..2,-2..2],[10,10],[-2..2,-2..2],[4,4],output=points)
```

The result is a list of points. Each point is a list of numerical values. It is a sequential reading of the data store in the matrix form.

$$\begin{aligned} &[[[-2., -2., 4.], [-.6666666667, -2., 2.222222223], [.6666666667, -2., 2.222222223], \\ &\quad [2., -2., 4.]], [[-2., -.6666666667, 2.222222222], \\ &\quad [-.6666666667, -.6666666667, .4444444444], \\ &\quad [.6666666667, -.6666666667, .4444444444], [2., -.6666666667, 2.222222222]], [\\ &\quad [-2., .6666666667, 2.222222222], [-.6666666667, .6666666667, .4444444444], \\ &\quad [.6666666667, .6666666667, .4444444444], [2., .6666666667, 2.222222222]], [\\ &\quad [-2., 2., 4.], [-.6666666667, 2., 2.222222223], [.6666666667, 2., 2.222222223], \\ &\quad [2., 2., 4.]]] \end{aligned}$$

3.1.11 output = procedure

```
> h:=Conjugate(x->1/2*x^2,[-2..2],[10],[-2..2],[4],output=procedure)
```

The result is a procedure that compute the conjugate at any points by interpolation.

$h :=$ **proc**(*Sin*::{*numeric*, *array*, list(array(*numeric*)), list(*list*)})
local *din*, *Min*, *Minp*, *j*, *jin*, *fin*, *U*, *Ind*, *m*, *l*, *k*, *Pt*, *SS*, *J*, *Compt*, *M*, *Sint*, *S*, *L*;
$S :=$ [array(1 .. 4,[(4)=2.,(3)=.6666666667,(2)=-.6666666667,(1)=-2.])] ;
$L :=$
table([(4)=2.000000000,(3)=.2222222222,(2)=.2222222222,(1)=2.000000000]) ;

if type(Sin, {list($array$), list($list$)}) **then**
din := nops(Sin) ;
if $din \neq 1$ **then**
ERROR("Points of the dual domain belong to Rˆ", 1, " and not to Rˆ", din)
fi;
Min := map($Nbelm$, Sin) ;
M := [4] ;
U := array(op(map($x \to 1..x$, Min))) ;
J := array(1 .. 1,[(1)=[1]]) ;
jin := [0] ;
k := 'k' ;
l := 'l' ;
while not inc([''jin'$_k$' \$ (k = 1..Nbelm(jin))], Min, 'i') **do**
fin := $false$;
l := 'l' ;
Pt := [seq($Sin_{l\,jin_l}$, l = 1..1)] ;
m := 'm' ;
l := 'l' ;
m := array(1..1) ;
for l **to** 1 **do**
for m_l **from** J_{il} **to** M_l **while** $S_{l\,m_l} < Pt_l$ **do od** ;
fin := $S_{l\,m_l} < Pt_l$;
if fin **then** $break$ **fi**
od;
m := convert(m, $list$) ;
for l **to** i **do** J_l := m **od** ;
l := 'l' ;
k := 'k' ;
if fin **then** ERROR("The point ", eval(Pt), " does not\t\tbelong to the grid ", S,
" so we cannot give a value at that point")
fi;
SS := array(1..2) ;
k := [0] ;
$Compt$:= 1 ;
while not inc([''k'$_l$' \$ (l = 1..1)], [2 \$ (l = 1..1)], 'jj') **do**
SS_{Compt} :=
[seq($S_{l\,m_l - k_l + 1}$, l = 1..1), $L_{\mathrm{seq}(m_l - k_l + 1,\, l = 1..1)}$];
$Compt$:= $Compt + 1$;
l := 'l'
od;
if $Compt \neq 3$ **then** ERROR("Something is wrong here") **fi** ;
k := 'k' ;

$m :=$ 'm' ;
$Sint := [\mathrm{seq}(Sin_{k\,jin_k},\ k = 1..1)]$;
$U_{\mathrm{op}(jin)} := \mathrm{Approx\_affine}(Sint,\ \mathrm{convert}(SS,\ list))$;
$k :=$ 'k'
od;
eval(U)
fi
end

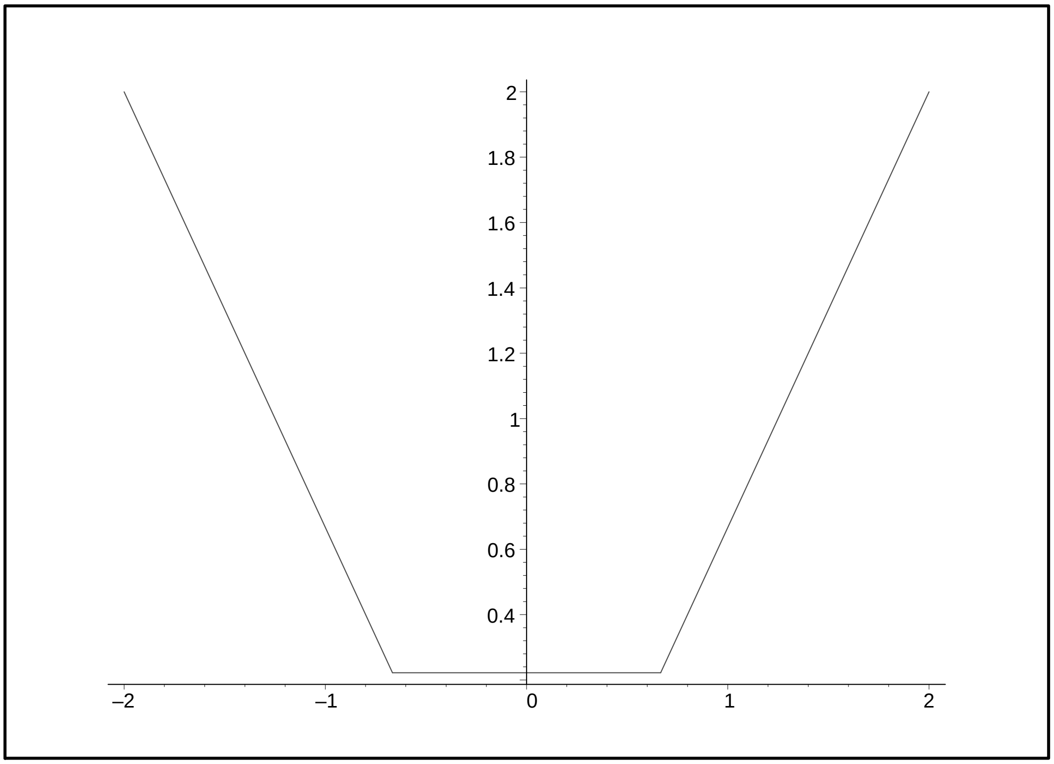
> `h([[0]])`

$$[.2222222222]$$

3.1.12 output = plot

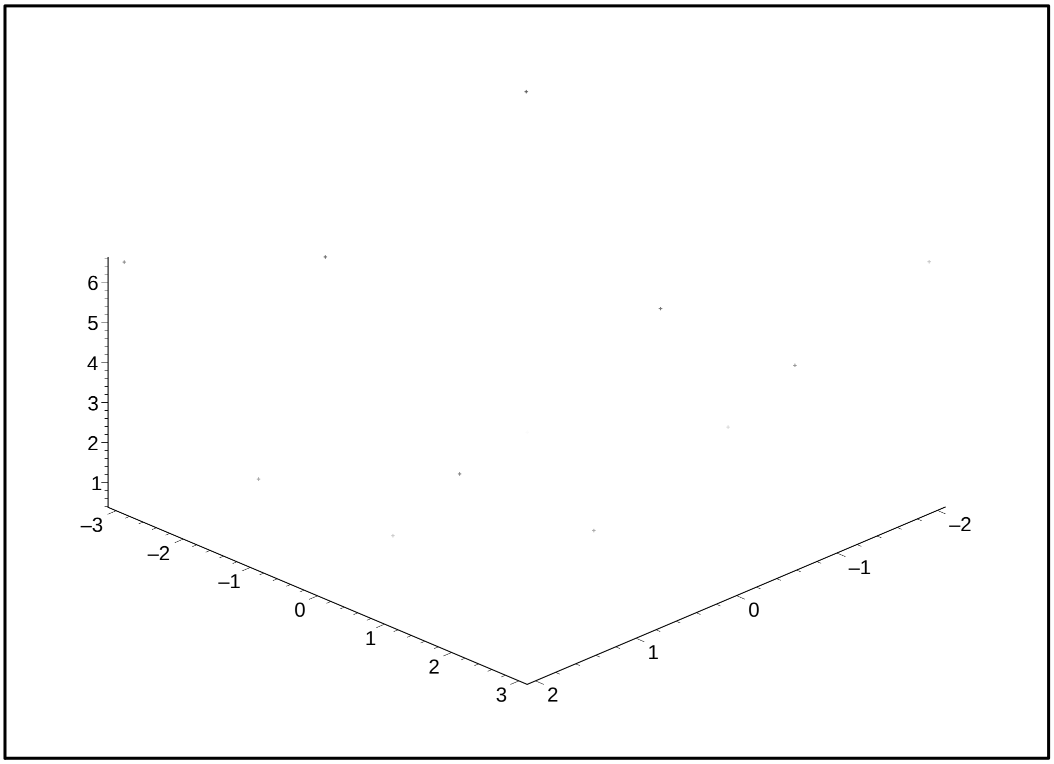
> `Conjugate(x->1/2*x^2,[-2..2],[10],[-2..2],[4],output=plot)`

The result is a plot structure. This option is only meaningful for functions of 1 or 2 variables.



You can also use the usual plot options:

```
> Conjugate((x,y)->1/2*(x^2+y^2),[-2..2,-3..3],[3,4],[-2..2,-3..3],[3,4],\
> output=plot,style=point,axes=framed)
```



3.1.13 The method option, default: method=sequential

The method option is used to describe which method of computation the Conjugate function should use. The 4 possibilities are: direct, dynamic, sequential, formal

3.1.14 method = direct

This option is only useful for comparaison purposes and evaluation of the speed of the LLT algorithm. It computes the conjugate directly, i.e. with a quadratic algorithm

```
> f:=x->exp(x):
> t:=time():Conjugate(f,[-10..10],[100],method=direct):time()-t
> t:=time():Conjugate(f,[-10..10],[100],method=sequential):time()-t
```

$$158.058$$

$$2.706$$

3.1.15 method = sequential

This is the default method. It computes the conjugate using the Linear-time Legendre Transform algorithm, and has a linear worst-case computation time.

```
> f:=x->exp(x):
> t:=time():Conjugate(f,[-10..10],[30],method=direct):time()-t
> t:=time():Conjugate(f,[-10..10],[30],method=sequential):time()-t
```

$$15.129$$

$$.511$$

3.1.16 method = dynamic

This option is not implemented yet. The idea is to compute the conjugate with an on-line algorithm, i.e. adding one slope at a time.

3.1.17 method = formal

This option is not implemented yet and would give errors. In the future, it will compute the conjugate with symbolic computation and return an expression not numerical values. To make it work, one needs to integrate the fenchel package and work around the piecewise function.

3.2 Computing the domain of the conjugate, the Domain function

The Domain function is a short function which computes the domain of the conjugate by using limits. It is used in the Conjugate function but may be useful by itself. The syntax is simple: give a univariate function and a range.

```
> f:=x->-1+exp(-x)
> Domain(f,-infinity..infinity)
```

$$f := x \to -1 + e^{(-x)}$$

$$-\infty..0$$

3.3 Computing convex hull: the planar_convex_hull function

The planar_convex_hull function computes the (lower) convex hull of a set of points. More precisely, given 2 arrays (or lists) corresponding to points in the plane, it returns a list of 2 arrays which store the verteces of the lower convex hull. This function is used in the Conjugate function to convexify the data prior to computing the conjugate.

```
> planar_convex_hull([-2,-1,0,1,2],[1,0,1,0,1])
```

The point (0,1) has been removed since it is not a vertex of the convex hull

$$[[-2, -1, 1, 2], [1, 0, 0, 1]]$$

4 The known bugs

4.1 Pb while computing Domains

g has domain -infinity..0

```
> g:=y->if type(y,numeric) then if y<0 then -y*ln(-y)+1+y
> elif y=0 then 1 else infinityfielse 'g'(y)fi:Domain(g,-infinity..0)
```

Maple has trouble computing limits with a procedure such as g. The result should be -infinity..infinity.

$$-(\lim_{t\to\infty}\frac{\mathrm{g}(-\infty-t)}{t})..\infty$$

Note that we do need to return unevaluated when the input is not numeric:

```
> g:=y->if y<0 then -y*ln(-y)+1+y elif y=0 then 1 else infinityfi:
> Domain(g,-infinity..0)
```

```
Error, (in g) cannot evaluate boolean
```

A solution is to use no if clause:

```
> g:=y->-y*ln(-y)+1+yDomain(g,-infinity..0)
```

$$g := y \to -y\ln(-y)+1+y$$

$$-\infty..\infty$$

4.2 Cannot input scattered points

```
> i:='i'X:=[seq([i,i^2,i],i=1..5)] Conjugate(X,[1..3,1..3],[3,3])
```

$$i := i$$

$$X := [[1, 1, 1], [2, 4, 2], [3, 9, 3], [4, 16, 4], [5, 25, 5]]$$

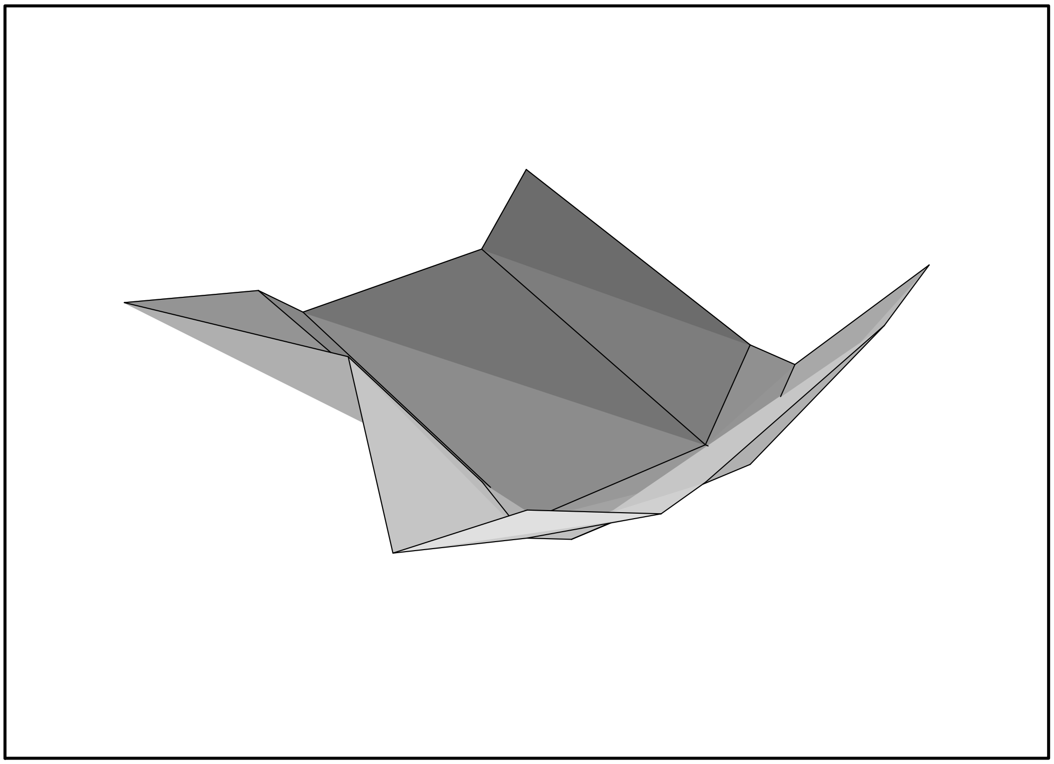
$$L$$

4.3 Pb with outcompress in 2 dimensions

Pb with outcompress in 2 dimensions. Do not use that option except (maybe) on boxed primal domains.

```
> 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 vv:=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=true):display( {P})
```

The ouput should be convex



4.4 Pb with infinity: plot3d crashes when asking to plot an indicator function

The solution is to compute with infinity since the LLT algorithm tackle it but to plot with a large value. See the example on indicator functions.

```
> f:=(x,y)->if type(x,numeric) and type(y,numeric) then if x^2+2*y^2<=1 then 0\
> else infinityfi else 'f'(x,y)fi
```

$f :=$ **proc**$(x,\, y)$
option *operator*, *arrow*;
if type(x, *numeric*) **and** type(y, *numeric*) **then if** $x^2 + 2 \times y^2 \leq 1$ **then** 0 **else** ∞ **fi**
else 'f'$(x,\, y)$

fi
end

```
> f(0,0)f(1,0)f(2,1)f(0,1/2)f(2,2)
```

$$0$$

$$0$$

$$\infty$$

$$0$$

$$\infty$$

The next command freezes Maple:

```
> plot3d(f,-2..2,-3..3)
```

The next command gives an error:

```
> plot3d(f(x,y),x=-2..2,y=-3..3)

Plotting error, empty plot
```

4.5 Wrong result when the input is an expression

The following is a wrong input: the first argument must be a function and NOT an expression.

```
> Conjugate(exp(x),[-10..10],[10])
```

$$[[[-10., 10.]], [-\infty, -\infty]]$$

Here is the right syntax:

```
> Conjugate(x->exp(x),[-10..10],[10])
```

$$[[[-10., -1.111111111, 1.111111111, 3.333333333, 10.]],\\ [99.99995460, 11.11106571, -1.563760889, .665971926, 8.073379333]]$$

4.6 The "method = procedure" option should be greatly improved. As for now it is not much useful

References

[1] H. H. Bauschke and M. von Mohrenschildt, *Fenchel conjugates and subdifferentials in maple*, Tech. Rep. CORR 97-23, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada, Nov 1997.

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

[3] Y. Lucet, *Faster than the fast Legendre transform, the linear-time Legendre transform*, Numer. Algorithms, 16 (1997), pp. 171–185 (1998).

[4] Y. Lucet, *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, Université Paul Sabatier, 118 Route de Narbonne, 31062 Toulouse Cedex 4, France, Feb 1997.