

Package: diffcp (via r-universe)

June 4, 2026

Type Package

Title Differentiating Through Cone Programs

Version 0.1.1

Description A port of the 'python' 'diffcp' package. Computes the derivative of the optimal solution map of a convex cone program, treating the program as an implicit function of its data (constraint matrix, offset, objective coefficients, and optionally a quadratic), mirroring Agrawal et al. (2019) <[doi:10.48550/arXiv.1904.09043](https://doi.org/10.48550/arXiv.1904.09043)>.

URL <https://bnaras.github.io/diffcp/>

BugReports <https://github.com/bnaras/diffcp/issues>

License Apache License (>= 2)

Encoding UTF-8

Imports cli, clarabel, Matrix, methods, Rcpp

LinkingTo Rcpp, RcppEigen

Suggests knitr, pkgdown, rmarkdown, scs, testthat (>= 3.0.0)

VignetteBuilder knitr

SystemRequirements C++17

Config/testthat/edition 3

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Config/pak/sysreqs make libclang-dev

Repository <https://bnaras.r-universe.dev>

Date/Publication 2026-05-26 15:49:20 UTC

RemoteUrl <https://github.com/bnaras/diffcp>

RemoteRef HEAD

RemoteSha 109c12f15298f9cbee8407ddc7d211581e8a1a9c

Contents

parse_cone_dict	2
pi	2
solve_and_derivative	3
solve_only	4
unvec_symm	5
upstream_info	5
vec_symm	6

Index	7
--------------	----------

parse_cone_dict	<i>Parse an SCS-style cone dictionary into an ordered list</i>
-----------------	--

Description

Mirrors `diffcp.cones.parse_cone_dict`.

Usage

```
parse_cone_dict(cone_dict)
```

Arguments

cone_dict	A named list with keys among "z", "l", "q", "s", "ep", "ed". Values are scalar dimensions (for z, l, ep, ed) or integer vectors of dimensions (for q, s).
-----------	---

Value

A list of `list(name, size)` pairs, in the canonical SCS cone order (CONES).

pi	<i>Projection onto a Cartesian product of cones</i>
----	---

Description

Mirrors `diffcp.cones.pi`.

Usage

```
pi(x, cones, dual = FALSE)
```

Arguments

x	A numeric vector.
cones	A list of <code>list(name, size)</code> pairs, as produced by <code>parse_cone_dict</code> .
dual	If TRUE, project onto the dual cones.

Value

A numeric vector of the same length as x .

`solve_and_derivative` *Solve a cone program and return forward / adjoint derivative operators*

Description

Mirrors `diffcp.cone_program.solve_and_derivative`. Solves minimize $c^T x$ s.t. $Ax + s = b$, s in K (with optional QP $0.5 x^T P x$ term) and returns the optimal (x, y, s) together with closures D (forward) and DT (adjoint) that map perturbations of $(A, b, c, [P])$ to perturbations of (x, y, s) and vice versa.

Usage

```
solve_and_derivative(
  A,
  b,
  c,
  cone_dict,
  P = NULL,
  solve_method = "Clarabel",
  mode = "lsqr",
  warm_start = NULL,
  ...
)
```

Arguments

<code>A</code>	A sparse <code>dgMatrix</code> constraint matrix.
<code>b</code>	A numeric offset vector.
<code>c</code>	A numeric objective coefficient vector.
<code>cone_dict</code>	A named list with cone sizes (keys among "z", "l", "q", "s", "ep", "ed").
<code>P</code>	Optional sparse <code>dgMatrix</code> for QP objective.
<code>solve_method</code>	One of "Clarabel" (default) or "SCS".
<code>mode</code>	Differentiation mode: "lsqr" (default), "dense", or "lpgd".
<code>warm_start</code>	Optional warm-start list (x, y, s) .
<code>...</code>	Additional control parameters forwarded to the solver.

Value

A list with elements $x, y, s, \text{info}, D, DT$. `info` is the solver-status block returned by the underlying solver (`status, iter, solveTime, pobj, ...`); see `solve_only` for the same shape.

solve_only	<i>Solve a cone program (forward only)</i>
------------	--

Description

Mirrors `diffcp.cone_program.solve_only`. Solves minimize $c^T x (+ 0.5 x^T P x)$ subject to $Ax + s = b$, s in K and returns the optimal (x, y, s) . Unlike `solve_and_derivative` this function does not build the derivative closures.

Usage

```
solve_only(
  A,
  b,
  c,
  cone_dict,
  warm_start = NULL,
  solve_method = "Clarabel",
  P = NULL,
  ...
)
```

Arguments

<code>A</code>	A sparse <code>dgMatrix</code> constraint matrix.
<code>b</code>	A numeric offset vector.
<code>c</code>	A numeric objective coefficient vector.
<code>cone_dict</code>	A named list with cone sizes (keys among "z", "l", "q", "s", "ep", "ed").
<code>warm_start</code>	Optional warm-start <code>list(x, y, s)</code> .
<code>solve_method</code>	One of "Clarabel" (default) or "SCS".
<code>P</code>	Optional sparse <code>dgMatrix</code> for QP objective.
<code>...</code>	Additional control parameters forwarded to the solver.

Value

A list with elements `x`, `y`, `s`, `info`.

unvec_symm	<i>Inverse of vec_symm</i>
------------	----------------------------

Description

Inverse of `vec_symm`

Usage

```
unvec_symm(x, dim)
```

Arguments

<code>x</code>	A numeric vector of length $n*(n+1)/2$.
<code>dim</code>	The matrix dimension n .

Value

The corresponding $n \times n$ symmetric matrix.

upstream_info	<i>Upstream Python diffcp pin</i>
---------------	-----------------------------------

Description

Returns the upstream pin metadata for this R port: which `cvxgrp/diffcp` version / commit the R sources track, the date of that commit, and `diffcp`'s own CVXPY runtime constraint. The authoritative record lives in `inst/UPSTREAM.dcf` and is read at call time.

Usage

```
upstream_info()
```

Value

A named character vector with one element per DCF field (Upstream, Version, Commit, ShortCommit, Date, URL, CVXPY, Snapshot, Notes).

Examples

```
upstream_info()
upstream_info()[["Version"]]
```

`vec_symm`*Symmetric vectorization (SCS convention)*

Description

Returns a vectorized representation of a symmetric matrix X , with off-diagonal entries scaled by $\sqrt{2}$ to make the SCS-style dot product $\langle \text{vec_symm}(A), \text{vec_symm}(B) \rangle = \text{trace}(A B)$ hold.

Usage

```
vec_symm(X)
```

Arguments

X A symmetric matrix.

Details

Mirrors `diffcp.cones.vec_symm` (Python).

Value

A numeric vector of length $n*(n+1)/2$.

Index

parse_cone_dict, 2
pi, 2

solve_and_derivative, 3
solve_only, 4

unvec_symm, 5
upstream_info, 5

vec_symm, 6