Package ‘cccp’

October 30, 2020

Version 0.2-7

Date 2020-10-30

Title Cone Constrained Convex Problems

Maintainer Bernhard Pfaff <bernhard@pfaffikus.de>

Depends R (>= 3.0.1), methods

Suggests RUnit, numDeriv

LazyLoad yes

Description Routines for solving convex optimization problems with cone constraints by means of interior-point methods. The implemented algorithms are partially ported from CVXOPT, a Python module for convex optimization (see <http://cvxopt.org> for more information).

Imports Rcpp (>= 0.11.2)

LinkingTo Rcpp, RcppArmadillo

License GPL (>= 3)

RcppModules CPG

Author Bernhard Pfaff [aut, cre],
Lieven Vandenberghe [cph] (copyright holder of cvxopt),
Martin Andersen [cph] (copyright holder of cvxopt),
Joachim Dahl [cph] (copyright holder of cvxopt)

Repository CRAN

Repository/R-Forge/Project cccp

Repository/R-Forge/Revision 189

Repository/R-Forge/DateTimeStamp 2020-10-30 14:40:55

Date/Publication 2020-10-30 16:00:02 UTC

NeedsCompilation yes

R topics documented:

`cccp` *Solving linear and quadratic programs with cone constraints*

Description

This function is the main function for defining and solving convex problems in the form of either linear or quadratic programs with cone constraints.

Usage

```
cccp(P = NULL, q = NULL, A = NULL, b = NULL, cList = list(),
     x0 = NULL, f0 = NULL, g0 = NULL, h0 = NULL,
     nlfList = list(), nlgList = list(), nlhList = list(),
     optctrl = ctrl())
```

Arguments

`P`	An object of class `matrix` with dimension $N \times N$ or `NULL`.
`q`	An object of class `vector` with dimension $N \times 1$ or `NULL`.
`A`	An object of class `matrix` with dimension $p \times N$.
`b`	An object of class `vector` with dimension $p \times 1$.

`cList`	A `list` object containing the cone constraints. Elements must be of either S4-class `NNOC`, or `SOCC`, or `PSDC`.
`x0`	An object of class `vector` with dimension $n \times 1$ for the initial values. The point `x0` must be in the domain of the nonlinear constraints.
`f0`	`function`: the scalar-valued convex and twice-differentiable objective function (its first argument must be '`x`').
`g0`	`function`: the gradient function of the objective (its first argument must be '`x`').
`h0`	`function`: the Hessian function of the objective (its first argument must be '`x`').
`nlfList`	A `list` object containing the nonlinear constraints as its elements. The functions have to be specified with `x` as their first argument and must be casted in implicit form, *i.e.* $f(x) \leq 0$.
`nlgList`	A `list` object containing the gradient functions as its elements. The functions have to be specified with `x` as their first argument.
`nlhList`	A `list` object containing the Hessian functions as its elements. The functions have to be specified with `x` as their first argument.
`optctrl`	An object of S4-class `Rcpp_CTRL`.

Value

An object of class `Rcpp_CPS`.

`CPD-class` *Class* `"CPD"`

Description

Class union of `Rcpp_DLP`, `Rcpp_DQP`, `Rcpp_DCP` and `Rcpp_DNL`.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "CPD" in the signature.

CPG — *Rcpp module: CPG*

Description

Module for defining and solving convex programs.

Details

The module contains the following items: classes:

CONEC Class for inequality (cone) constraints.

CTRL Class for control parameters used in optimizations.

PDV Class for primal/dual variables.

DCP Class for definition of convex programs.

DLP Class for definition of linear programs.

DNL Class for definition of linear programs with non-linear constraints.

DQP Class for definition of quadratic programs.

CPS Class for solution of convex programs.

functions:

rpp Function for solving risk parity portfolios.

gpp Function for solving a geometric program.

cps — *Solving a convex program*

Description

This function returns an optimal point for a cone constraint convex program.

Usage

```
## S4 method for signature 'Rcpp_DCP,Rcpp_CTRL'
cps(cpd, ctrl)
## S4 method for signature 'Rcpp_DLP,Rcpp_CTRL'
cps(cpd, ctrl)
## S4 method for signature 'Rcpp_DNL,Rcpp_CTRL'
cps(cpd, ctrl)
## S4 method for signature 'Rcpp_DQP,Rcpp_CTRL'
cps(cpd, ctrl)
```

Arguments

`cpd`	An object belonging to the class union `CPD`.
`ctrl`	An object of reference-class `Rcpp_CTRL`.

Value

An object of reference-class `Rcpp_CPS`.

`ctrl` *Creating objects of reference-class CTRL*

Description

This function creates an object of reference-class `CTRL` which contains optimization parameters, *e.g.* the maximum number of iterations.

Usage

```
ctrl(maxiters = 100L, abstol = 1e-06, reltol = 1e-06,
     feastol = 1e-06, stepadj = 0.95, beta = 0.5, trace = TRUE)
```

Arguments

`maxiters`	`integer`, the maximum count of iterations.
`abstol`	`numeric`, the absolute level for convergence to be achieved.
`reltol`	`numeric`, the relative level for convergence to be achieved.
`feastol`	`numeric`, the feasable level for convergence to be achieved.
`stepadj`	`numeric`, step size adjustment in combined step.
`beta`	`numeric`, parameter in backtracking line search.
`trace`	`logical`, if `TRUE` (the default), the solver's progress during the iterations is shown.

Value

An object of reference-class `CTRL`.

Note

Either `abstol` or `reltol` can be set to a negative real number. `feastol` must be greater than zero.

See Also

`Rcpp_CTRL`

dcp *Creating a member object of the reference-class* DCP

Description

This function returns an object containing the definition of a convex program with non-linear constraints and (if provided) cone constraints. The returned object is a member of the reference-class DCP.

Usage

```
dcp(x0, f0, g0, h0, cList = list(), nlfList = list(), nlgList = list(),
    nlhList = list(), A = NULL, b = NULL)
```

Arguments

x0	An object of class vector with dimension $n \times 1$ for the initial values. The point x0 must be in the domain of the nonlinear constraints.
f0	function: the scalar-valued convex and twice-differentiable objective function (its first argument must be 'x').
g0	function: the gradient function of the objective (its first argument must be 'x'); returning a vector.
h0	function: the Hessian function of the objective (its first argument must be 'x'); returning a matrix.
cList	A list object containing the cone constraints. Elements must be of either S4-class NNOC, or SOCC, or PSDC or an empty list in case of no inequality constraints.
nlfList	A list object containing the nonlinear constraints as its elements. The functions have to be specified with x as their first argument and must be casted in implicit form, *i.e.* $f(x) \leq 0$.
nlgList	A list object containing the gradient functions as its elements. The functions have to be specified with x as their first argument.
nlhList	A list object containing the Hessian functions as its elements. The functions have to be specified with x as their first argument.
A	An object of class matrix with dimension $p \times n$ or NULL for problems without equality constraints.
b	An object of class vector with dimension $p \times 1$ or NULL for problems without equality constraints.

Value

An object belonging to the reference-class DCP.

dlp *Creating a member object of the reference-class* DLP

Description

This function returns an object containing the definition of a cone constrained linear program. The returned object is a member of the reference-class DLP.

Usage

```
dlp(q, A = NULL, b = NULL, cList = list())
```

Arguments

q	An object of class vector with dimension $n \times 1$.
A	An object of class matrix with dimension $p \times n$ or NULL for problems without equality constraints.
b	An object of class vector with dimension $p \times 1$ or NULL for problems without equality constraints.
cList	A list object containing the cone constraints. Elements must be of either reference-class NNOC, or SOCC, or PSDC or an empty list in case of no inequality constraints.

Value

An object belonging to the reference-class DLP.

dnl *Creating a member object of the reference-class* DNL

Description

This function returns an object containing the definition of a linear program with non-linear constraints and (if provided) cone constraints. The returned object is a member of the reference-class DNL.

Usage

```
dnl(q, A = NULL, b = NULL, cList = list(),
    x0, nlfList = list(), nlgList = list(), nlhList = list())
```

Arguments

q	`vector` of length n for the coefficients in the objective.
A	An object of class `matrix` with dimension $p \times n$ or `NULL` for problems without equality constraints.
b	An object of class `vector` with dimension $p \times 1$ or `NULL` for problems without equality constraints.
cList	A `list` object containing the cone constraints. Elements must be of either S4-class `NNOC`, or `SOCC`, or `PSDC` or an empty list in case of no inequality constraints.
x0	An object of class `vector` with dimension $n \times 1$ for the initial values. The point `x0` must be in the domain of the nonlinear constraints.
nlfList	A `list` object containing the nonlinear constraints as its elements. The functions have to be specified with `x` as their first argument and must be casted in implicit form, *i.e.* $f(x) \leq 0$.
nlgList	A `list` object containing the gradient functions as its elements. The functions have to be specified with `x` as their first argument.
nlhList	A `list` object containing the Hessian functions as its elements. The functions have to be specified with `x` as their first argument.

Value

An object belonging to the reference-class `DNL`.

---

dqp — *Creating a member object of the reference-class* `DQP`

Description

This function returns an object containing the definition of a cone constrained quadratic program. The returned object is a member of the reference-class `DQP`.

Usage

```
dqp(P, q, A = NULL, b = NULL, cList = list())
```

Arguments

P	An object of class `matrix` with dimension $n \times n$.
q	An object of class `vector` with dimension $n \times 1$.
A	An object of class `matrix` with dimension $p \times n$ or `NULL` for problems without equality constraints.
b	An object of class `vector` with dimension $p \times 1$ or `NULL` for problems without equality constraints.
cList	A `list` object containing the cone constraints. Elements must be of either reference-class `NNOC`, or `SOCC`, or `PSDC` or an empty list in case of no inequality constraints.

Value

An object belonging to the reference-class `DQP`.

`getFoo` *Extractor methods for reference class objects*

Description

Returns a member of reference class objects.

Usage

```
## S4 method for signature 'Rcpp_PDV'
getx(object)
## S4 method for signature 'Rcpp_CPS'
getx(object)
## S4 method for signature 'Rcpp_PDV'
gety(object)
## S4 method for signature 'Rcpp_CPS'
gety(object)
## S4 method for signature 'Rcpp_PDV'
gets(object)
## S4 method for signature 'Rcpp_CPS'
gets(object)
## S4 method for signature 'Rcpp_PDV'
getz(object)
## S4 method for signature 'Rcpp_CPS'
getz(object)
## S4 method for signature 'Rcpp_CPS'
getstate(object)
## S4 method for signature 'Rcpp_CPS'
getstatus(object)
## S4 method for signature 'Rcpp_CPS'
getniter(object)
## S4 method for signature 'Rcpp_CTRL'
getparams(object)
```

Arguments

`object`	An object of either reference-class `Rcpp_PDV` or `Rcpp_CPS`, or `Rcpp_CTRL`.

Value

The relevant member object of the class.

gp *Geometric program*

Description

This function solves a geometric program.

Usage

```
gp(F0, g0, FList = list(), gList = list(), nno = NULL,
   A = NULL, b = NULL, optctrl = ctrl())
```

Arguments

`F0`	Matrix in the objective function.
`g0`	Matrix in the objective function (affine terms).
`FList`	List of matrices in posinomial functions.
`gList`	List of matrices in posinomial functions (affine terms).
`nno`	Object created by a call to `nnoc()`.
`A`	Lefthand-side matrix of equality constraints.
`b`	Lefthand-side matrix of equality constraints.
`optctrl`	Object of reference class 'Rcpp_CTRL', created by a call to `ctrl()`.

Details

Solves a geometric program casted in its epigraph form.

Value

An object of S4-class `Rcpp_CPS`.

References

Boyd, S., Kim, S.-J., Vandenberghe, L. and A. Hassibi (2007), A tutorial on geometric programming, *Optim Eng*, Educational Section, **8**:67–127, Springer.

`l1` *Minimizing L1-norm*

Description

This function minimizes a L1-norm of the form $||Pu - q||_1$, whereby P is a $(m \times n)$ matrix and q is a $m \times 1$ vector. This function is wrapper function for invoking the `cps`-method of Linear Programs.

Usage

```
l1(P, q = NULL, optctrl = ctrl())
```

Arguments

`P`	`matrix` of dimension $m \times n$.
`q`	`vector` of length m.
`optctrl`	An object of S4-class `Rcpp_CTRL`.

Value

An object of S4-class `Rcpp_CPS`.

`nlfc` *Definition of nonlinear inequality constraints*

Description

This function is the interface to the reference class `NLFC` for creating nonlinear constraints.

Usage

```
nlfc(G, h)
```

Arguments

`G`	Object of class `"matrix"`: A $(m \times n)$ matrix containing the coefficients of the lefthand-side linear inequality constraints.
`h`	Object of class `NLFV`: A $(m \times 1)$ vector containing the coefficients of the righthand-side linear inequality constraints as slot `u`.

Value

List with elements: `conType`, `G` and `h`.

nnoc *Definition of linear inequality constraints*

Description

This function is the interface to the reference class NNOC for creating linear constraints.

Usage

```
nnoc(G, h)
```

Arguments

G	Object of class "matrix": A $(m \times n)$ matrix containing the coefficients of the lefthand-side linear inequality constraints.
h	Object of class NNOV: A $(m \times 1)$ vector containing the coefficients of the righthand-side linear inequality constraints as slot u.

Value

List with elements: conType, G and h.

psdc *Definition of positive semidefinite cone inequality constraints*

Description

This function is the interface to the reference class PSDC for creating positive semidefinite cone constraints.

Usage

```
psdc(Flist, F0)
```

Arguments

Flist	Object of class "list": A list with the matrices appearing on the left-hand side of the matrix inequality.
F0	Object of class "matrix": The matrix appearing on the righthand-side.

Details

A psd-cone constraint is given as $\sum_{i=1}^{n} x_i F_i \leq F_0$. The matrix G is created as $G = [\text{vech}(F_1)| \ldots |\text{vech}(F_n)]$ and the vector h is constructed as $h = [\text{vech}(F_0)]$.

Value

List with elements: conType, G and h.

Rcpp_CONEC-class *Class* "Rcpp_CONEC"

Description

Class for inequality (cone) constraints.

Extends

Class "C++Object", directly. All reference classes extend and inherit methods from "envRefClass".

Fields

`cone`: Object of class `activeBindingFunction`: Type of cone constraints.

`G`: Object of class `activeBindingFunction`: Left-hand side of inequality constraints.

`h`: Object of class `activeBindingFunction`: Right-hand side of inequality constraints.

`sidx`: Object of class `activeBindingFunction`: Row index for subsets of cone constraints.

`dims`: Object of class `activeBindingFunction`: Dimension of cone constraints.

`K`: Object of class `activeBindingFunction`: Count of inequality constraints.

`n`: Object of class `activeBindingFunction`: Count of variables in objective.

Examples

```
showClass("Rcpp_CONEC")
```

Rcpp_CPS-class *Class* "Rcpp_CPS"

Description

Class for solution of convex programs.

Extends

Class "C++Object", directly. All reference classes extend and inherit methods from "envRefClass".

Fields

`pdv`: Object of class `activeBindingFunction`: Primal-dual variables.

`state`: Object of class `activeBindingFunction`: Vector of state variables in convex programs.

`status`: Object of class `activeBindingFunction`: Character indicating the status of the returned solution.

`niter`: Object of class `activeBindingFunction`: Integer, count of iterations.

`sidx`: Object of class `activeBindingFunction`: Integer matrix, start and end indices of slack variables.

Examples

```
showClass("Rcpp_CPS")
```

Rcpp_CTRL-class *Class* "Rcpp_CTRL"

Description

Class for control options used in optimization routines.

Extends

Class "C++Object", directly. All reference classes extend and inherit methods from "envRefClass".

Fields

ctrlparams: Object of class activeBindingFunction: List of control parameters.

Examples

```
showClass("Rcpp_CTRL")
```

Rcpp_DCP-class *Class* "Rcpp_DCP"

Description

Class for definition of convex programs with non-linear constraints.

Extends

Class "C++Object", directly. All reference classes extend and inherit methods from "envRefClass".

Fields

x0: Object of class activeBindingFunction: Initial values.

cList: Object of class activeBindingFunction: Inequality constraints, class CONEC.

nList: Object of class activeBindingFunction: List with elements of functions for evaluating non-linear constraints, their associated gradients and their associated Hessians.

A: Object of class activeBindingFunction: Left-hand side of equality cosntraints.

b: Object of class activeBindingFunction: Right-hand side of equality cosntraints.

Examples

```
showClass("Rcpp_DCP")
```

Rcpp_DLP-class *Class* "Rcpp_DLP"

Description

Class for definition of linear programs.

Extends

Class "C++Object", directly. All reference classes extend and inherit methods from "envRefClass".

Fields

q: Object of class activeBindingFunction: Matrix related to linear term in objective.

A: Object of class activeBindingFunction: Left-hand side of equality cosntraints.

b: Object of class activeBindingFunction: Right-hand side of equality cosntraints.

cList: Object of class activeBindingFunction: Inequality constraints, class CONEC.

Examples

```
showClass("Rcpp_DLP")
```

Rcpp_DNL-class *Class* "Rcpp_DNL"

Description

Class for definition of linear programs with non-linear constraints.

Extends

Class "C++Object", directly. All reference classes extend and inherit methods from "envRefClass".

Fields

q: Object of class activeBindingFunction: Matrix related to linear term in objective.

A: Object of class activeBindingFunction: Left-hand side of equality cosntraints.

b: Object of class activeBindingFunction: Right-hand side of equality cosntraints.

cList: Object of class activeBindingFunction: Inequality constraints, class CONEC.

x0: Object of class activeBindingFunction: Initial values.

nList: Object of class activeBindingFunction: List with elements of functions for evaluating non-linear constraints, their associated gradients and their associated Hessians.

Examples

```
showClass("Rcpp_DNL")
```

Rcpp_DQP-class *Class* "Rcpp_DQP"

Description

Class for definition of quadratic programs.

Extends

Class "C++Object", directly. All reference classes extend and inherit methods from "envRefClass".

Fields

- **P:** Object of class activeBindingFunction: Matrix related to quadratic term in objective.
- **q:** Object of class activeBindingFunction: Matrix related to linear term in objective.
- **A:** Object of class activeBindingFunction: Left-hand side of equality cosntraints.
- **b:** Object of class activeBindingFunction: Right-hand side of equality cosntraints.
- **cList:** Object of class activeBindingFunction: Inequality constraints, class CONEC.

Examples

```
showClass("Rcpp_DQP")
```

Rcpp_PDV-class *Class* "Rcpp_PDV"

Description

Class for primal/dual variables in convex programs.

Extends

Class "C++Object", directly. All reference classes extend and inherit methods from "envRefClass".

Fields

- **x:** Object of class activeBindingFunction: Primal variables.
- **y:** Object of class activeBindingFunction: Dual variables.
- **s:** Object of class activeBindingFunction: Primal slack variables.
- **z:** Object of class activeBindingFunction: Dual slack variables.
- **kappa:** Object of class activeBindingFunction: Self-dual embedding variable; used in LPs, only.
- **tau:** Object of class activeBindingFunction: Self-dual embedding variable; used in LPs, only.

Examples

```
showClass("Rcpp_PDV")
```

---

rp *Risk-parity optimization*

Description

This function determines a risk-parity solution of a long-only portfolio with a budget-constraint.

Usage

```
rp(x0, P, mrc, optctrl = ctrl())
```

Arguments

`x0`	`matrix` of dimension $n \times 1$; starting values.
`P`	`matrix` of dimension $n \times n$; dispersion matrix.
`mrc`	`matrix` of dimension $n \times 1$; the marginal risk contributions.
`optctrl`	An object of S4-class `Rcpp_CTRL`.

Value

An object of S4-class `Rcpp_CPS`.

References

Spinu, F. (2013), An Algorithm for Computing Risk Parity Weights, SSRN, *OMERS Capital Markets*, July 2013.

---

socc *Definition of second-oder cone inequality constraints*

Description

This function is the interface to the reference class `SOCC` for creating second-oder cone constraints.

Usage

```
socc(F, g, d, f)
```

Arguments

- `F` Object of class "matrix": The matrix appearing in the norm-expression on the left-hand side of a second-order cone constraint.
- `g` Object of class "numeric": The vector appearing in the norm-expression on the left-hand side of a second-order cone constraint.
- `d` Object of class "numeric": The vector appearing on the right-hand side of a second-order cone constraint.
- `f` Object of class "numeric": The scalar appearing on the right-hand side of a second-order cone constraint.

Details

A second-order cone constraint is given as $||Fx + g||_2 \leq d'x + f$. The matrix G is created as $G = [-d, -F]$ and the vector h is constructed as $h = [f, g]$.

Value

List with elements: `conType`, `G` and `h`.

Index