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 ................................................................. 2
  CPD-class .......................................................... 3
  CPG ................................................................. 4
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

```r
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 \( p \times 1 \).
**CPD-class**

| **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.
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.

Solving a convex program

Description

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

Usage

```r
## 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)
```
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 feasible 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
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**

```r
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**

```r
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.

nlflList  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.
getFoo

Value
An object belonging to the reference-class DQP.

Extractor methods for reference class objects

Description
Returns a member of reference class objects.

Usage

```r
## S4 method for signature 'Rcpp_PDV'
gtx(object)
## S4 method for signature 'Rcpp_CPS'
gtx(object)
## S4 method for signature 'Rcpp_PDV'
gty(object)
## S4 method for signature 'Rcpp_CPS'
gty(object)
## S4 method for signature 'Rcpp_PDV'
gts(object)
## S4 method for signature 'Rcpp_CPS'
gts(object)
## S4 method for signature 'Rcpp_PDV'
gstate(object)
## S4 method for signature 'Rcpp_CPS'
gstatus(object)
## S4 method for signature 'Rcpp_CPS'
gniter(object)
## S4 method for signature 'Rcpp_CCTRL'
gparams(object)
```

Arguments

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

Value
The relevant member object of the class.
Description

This function solves a geometric program.

Usage

```r
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 'RcppCTRL', 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

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
\[
l1(P, q = \text{NULL}, \text{optctrl} = \text{ctrl}())
\]

Arguments
- \(P\): matrix of dimension \(m \times n\).
- \(q\): vector of length \(m\).
- \(\text{optctrl}\): 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 \((m\times1)\) 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**

```r
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 NNOC: 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**

```r
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

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

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

Examples
showClass("Rcpp_DCP")
Rcpp_DLP-class

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 constraints.
b: Object of class activeBindingFunction: Right-hand side of equality constraints.
cList: Object of class activeBindingFunction: Inequality constraints, class CONEC.

Examples
showClass("Rcpp_DLP")

Rcpp_DNL-class

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 constraints.
b: Object of class activeBindingFunction: Right-hand side of equality constraints.
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")
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 constraints.
- b: Object of class activeBindingFunction: Right-hand side of equality constraints.
- cList: Object of class activeBindingFunction: Inequality constraints, class CONEC.

Examples

showClass("Rcpp_DQP")

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**


---

**socc**  
*Definition of second-order cone inequality constraints*

---

**Description**

This function is the interface to the reference class SOCC for creating second-order 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 $\|Fx + g\|_2 \leq d^\prime x + f$. 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

**classes**
- CPD-class, 3
- nlfc, 11
- nnoc, 12
- psdc, 12
- Rcpp_CONEC-class, 13
- Rcpp_CPS-class, 13
- Rcpp_CTRL-class, 14
- Rcpp_DCP-class, 14
- Rcpp_DLP-class, 15
- Rcpp_DNL-class, 15
- Rcpp_DQP-class, 16
- Rcpp_PDV-class, 16
- socc, 17

**datasets**
- CPG, 4

**optimize**
- cccp, 2
- cps, 4
- ctrl, 5
- dcp, 6
- dlp, 7
- dn1, 7
- dqp, 8
- getFoo, 9
- gp, 10
- l1, 11
- rp, 17

---

### C++Object, 13–16
- cccp, 2
- CONEC (CPG), 4
- CPD-class, 3
- CPG, 4
- CPS (CPG), 4
- cps, 4
- cps, Rcpp_DCP, Rcpp_CTRL-method (cps), 4
- cps, Rcpp_DLP, Rcpp_CTRL-method (cps), 4
- cps, Rcpp_DNL, Rcpp_CTRL-method (cps), 4
- cps, Rcpp_DQP, Rcpp_CTRL-method (cps), 4
- CTRL (CPG), 4
- ctrl, 5
- DCP (CPG), 4
- dcp, 6
- DLP (CPG), 4
- dlp, 7
- DNL (CPG), 4
- dn1, 7
- DQP (CPG), 4
- dqp, 8
- envRefClass, 13–16

- getFoo, 9
- getniter (getFoo), 9
- getparams (getFoo), 9
- getparams, Rcpp_CPS-method (getFoo), 9
- gets (getFoo), 9
- gets, Rcpp_CPS-method (getFoo), 9
- gets, Rcpp_PDV-method (getFoo), 9
- getstate (getFoo), 9
- getstate, Rcpp_CPS-method (getFoo), 9
- getstatus (getFoo), 9
- getstatus, Rcpp_CPS-method (getFoo), 9
- getx (getFoo), 9
- getx, Rcpp_CPS-method (getFoo), 9
- getx, Rcpp_PDV-method (getFoo), 9
- gety (getFoo), 9
- gety, Rcpp_CPS-method (getFoo), 9
- gety, Rcpp_PDV-method (getFoo), 9
- getz (getFoo), 9
- getz, Rcpp_CPS-method (getFoo), 9
- getz, Rcpp_PDV-method (getFoo), 9
- gp, 10
- gpp (CPG), 4
- l1, 11
- nlfc, 11
nnoc, 12
PDV (CPG), 4
psdc, 12
Rcpp_CONEC-class, 13
Rcpp_CPS-class, 13
Rcpp_CTRL, 5
Rcpp_CTRL-class, 14
Rcpp_DCP-class, 14
Rcpp_DLP-class, 15
Rcpp_DNL-class, 15
Rcpp_DQP-class, 16
Rcpp_PDV-class, 16
rp, 17
rpp (CPG), 4
socc, 17