Package ‘sdpt3r’

October 14, 2022

Type Package

Title Semi-Definite Quadratic Linear Programming Solver

Version 0.3

Date 2019-02-08

Author Kim-Chuan Toh (Matlab/C), Micheal Todd (Matlab/C), Reha Tutunco (Matlab/C), Adam Rahman (R/C Headers), Timothy A. Davis (symamd C code), Stefan I. Larimore (symamd C code)

Maintainer Adam Rahman <a45rahma@uwaterloo.ca>


License GPL-2 | GPL-3

Depends Matrix, R (>= 2.10)

RoxygenNote 6.1.1

NeedsCompilation yes

Imports methods, stats

Repository CRAN

Date/Publication 2019-02-11 08:50:03 UTC

R topics documented:

Andwd .............................. 2
Apdwd ................................ 3
Btep .................................. 3
Bgpp .................................. 3
Blogcheby ............................ 4
**Description**

*An Configuration Matrix for Distance Weighted Discrimination*

**Usage**

```r
data(Andwd)
```

**Format**

A matrix with 50 rows and 3 columns
### Apdwd

**Description**

Ap Configuration Matrix for Distance Weighted Discrimination

**Usage**

```r
data(Apdwd)
```

**Format**

A matrix with 50 rows and 3 columns

### Betp

**Description**

Symmetric Matrix for Educational Testing Problem

**Usage**

```r
data(Betp)
```

**Format**

A matrix with 5 rows and 5 columns

### Bgpp

**Description**

Adjacency Matrix for Graph Partitioning Problem

**Usage**

```r
data(Bgpp)
```

**Format**

A matrix with 10 rows and 10 columns
Blogcheby  
*B Matrix for the Log Chebyshev Approximation Problem*

**Description**
B Matrix for the Log Chebyshev Approximation Problem

**Usage**
```r
data(Blogcheby)
```

**Format**
A matrix with 10 rows and 10 columns

---

Bmaxcut  
*Adjacency Matrix for Max-Cut*

**Description**
Adjacency Matrix for Max-Cut

**Usage**
```r
data(Bmaxcut)
```

**Format**
A matrix with 10 rows and 10 columns

---

Bmaxkcut  
*Adjacency Matrix for Max-kCut*

**Description**
Adjacency Matrix for Max-kCut

**Usage**
```r
data(Bmaxkcut)
```

**Format**
A matrix with 10 rows and 10 columns
control_theory

Description

control_theory creates input for sqlp to solve the Control Theory Problem

Usage

control_theory(B)

Arguments

B a matrix object containing square matrices of size n

Details

Solves the control theory problem. Mathematical and implementation details can be found in the vignette

Value

X A list containing the solution matrix to the primal problem
y A list containing the solution vector to the dual problem
Z A list containing the solution matrix to the dual problem
pobj The achieved value of the primary objective function
dobj The achieved value of the dual objective function

Examples

B <- matrix(list(),2,1)
B[[1]] <- matrix(c(-.8,1.2,-.5,-1.1,-1,-2.5,2,.2,-1),nrow=3,byrow=TRUE)
B[[2]] <- matrix(c(-1.5,.5,-2,1.1,-2,.2,-1.4,1.1,-1.5),nrow=3,byrow=TRUE)
out <- control_theory(B)
### DoptDesign

**Test Vector Matrix for D-Optimal Design**

**Description**

Test Vector Matrix for D-Optimal Design

**Usage**

```r
data(DoptDesign)
```

**Format**

A matrix with 3 rows and 25 columns

### doptimal

**D-Optimal Experimental Design**

**Description**

doptimal creates input for sqlp to solve the D-Optimal Experimental Design problem - given an nxp matrix with p <= n, find the portion of points that maximizes det(A'A)

**Usage**

doptimal(V)

**Arguments**

- `V` a pxn matrix containing a set of n test vectors in dimension p (with p <= n)

**Details**

Solves the D-optimal experimental design problem. Mathematical and implementation details can be found in the vignette

**Value**

- `X` A list containing the solution matrix to the primal problem
- `y` A list containing the solution vector to the dual problem
- `Z` A list containing the solution matrix to the dual problem
- `pobj` The achieved value of the primary objective function
- `dobj` The achieved value of the dual objective function
dwd

**Examples**

```r
data(DoptDesign)
out <- doptimal(DoptDesign)
```

---

**dwd**

*Distance Weighted Discrimination*

**Description**

dwd creates input for sqlp to solve the Distance Weighted Discrimination problem - Given two sets of points An and Ap, find an optimal classification rule to group the points as accurately as possible for future classification.

**Usage**

dwd(Ap, An, penalty)

**Arguments**

- Ap: An nxp point configuration matrix
- An: An nxp point configuration matrix
- penalty: A real valued scalar penalty for moving points across classification rule

**Details**

Solves the distance weighted discrimination problem. Mathematical and implementation details can be found in the vignette

**Value**

- **X**: A list containing the solution matrix to the primal problem
- **y**: A list containing the solution vector to the dual problem
- **Z**: A list containing the solution matrix to the dual problem
- **pobj**: The achieved value of the primary objective function
- **dobj**: The achieved value of the dual objective function

**Examples**

```r
data(Andwd)
data(Apdwd)
penalty <- 0.5
#Not Run
#out <- dwd(Apdwd,Andwd,penalty)
```
etp

*Educational Testing Problem*

**Description**

`etp` creates input for `sqlp` to solve the Educational Testing Problem - given a symmetric positive definite matrix $S$, how much can be subtracted from the diagonal elements of $S$ such that the resulting matrix is positive semidefinite definite.

**Usage**

`etp(B)`

**Arguments**

- **B**  
  A symmetric positive definite matrix

**Details**

Solves the education testing problem. Mathematical and implementation details can be found in the vignette.

**Value**

- **X**  
  A list containing the solution matrix to the primal problem
- **y**  
  A list containing the solution vector to the dual problem
- **Z**  
  A list containing the solution matrix to the dual problem
- **pobj**  
  The achieved value of the primary objective function
- **dobj**  
  The achieved value of the dual objective function

**Examples**

```r
data(Betp)
out <- etp(Betp)
```
flogcheby

**flogcheby**

*f vector for the Log Chebyshev Approximation Problem*

**Description**

f vector for the Log Chebyshev Approximation Problem

**Usage**

data(flogcheby)

**Format**

A vector with length 20

---

Ftoep

**Ftoep**

*Symmetric Matrix for the Toeplitz Approximation Problem*

**Description**

Symmetric Matrix for the Toeplitz Approximation Problem

**Usage**

data(Ftoep)

**Format**

A matrix with 10 rows and 10 columns

---

Glovasz

**Glovasz**

*Adjacency Matrix on which to find the Lovasz Number*

**Description**

Adjacency Matrix on which to find the Lovasz Number

**Usage**

data(Glovasz)

**Format**

A matrix with 10 rows and 10 columns
Description

gpp creates input for sqlp to solve the graph partitioning problem.

Usage

gpp(B, alpha)

Arguments

B
A weighted adjacency matrix

alpha
Any real value in (0,n^2)

Details

Solves the graph partitioning problem. Mathematical and implementation details can be found in the vignette

Value

X
A list containing the solution matrix to the primal problem

y
A list containing the solution vector to the dual problem

Z
A list containing the solution matrix to the dual problem

pobj
The achieved value of the primary objective function

dobj
The achieved value of the dual objective function

Examples

data(Bgpp)
alpha <- nrow(Bgpp)

out <- gpp(Bgpp, alpha)
**Hnearcorr**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approximate Correlation Matrix for Nearest Correlation Matrix Problem</td>
</tr>
</tbody>
</table>

**Usage**

data(Hnearcorr)

**Format**

A matrix with 5 rows and 5 columns

---

**lmi1**

*Linear Matrix Inequality 1*

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lmi1 creates input for sqlp to solve a linear matrix inequality problem</td>
</tr>
</tbody>
</table>

**Usage**

lmi1(B)

**Arguments**

- **B**
  - An mxn real valued matrix

**Details**

Solves the type-1 linear matrix inequality problem. Mathematical and implementation details can be found in the vignette

**Value**

- **X**
  - A list containing the solution matrix to the primal problem
- **y**
  - A list containing the solution vector to the dual problem
- **Z**
  - A list containing the solution matrix to the dual problem
- **pobj**
  - The achieved value of the primary objective function
- **dobj**
  - The achieved value of the dual objective function
Examples

\[ B \leftarrow \text{matrix(c(-1,5,1,0,-2,1,0,0,-1), nrow=3)} \]

#Not Run
#out \leftarrow \text{lmi1(B)}

---

**lmi2**  
*Linear Matrix Inequality 2*

Description

lmi2 creates input for sqlp to solve a linear matrix inequality problem

Usage

lmi2(A1, A2, B)

Arguments

- **A1**: An nxm real valued matrix
- **A2**: An nxm real valued matrix
- **B**: An nxp real valued matrix

Details

Solves the type-2 linear matrix inequality problem. Mathematical and implementation details can be found in the vignette

Value

- **X**: A list containing the solution matrix to the primal problem
- **y**: A list containing the solution vector to the dual problem
- **Z**: A list containing the solution matrix to the dual problem
- **pobj**: The achieved value of the primary objective function
- **dobj**: The achieved value of the dual objective function

Examples

\[ A1 \leftarrow \text{matrix(c(-1,0,1,0,-2,1,0,0,-1),3,3)} \]
\[ A2 \leftarrow A1 + 0.1*t(A1) \]
\[ B \leftarrow \text{matrix(c(1,3,5,2,4,6),3,2)} \]

out \leftarrow lmi2(A1,A2,B)
Description

1mi3 creates input for sqlp to solve a linear matrix inequality problem

Usage

1mi3(A, B, G)

Arguments

A An nxn real valued matrix
B An mxn real valued matrix
G An nxn real valued matrix

Details

Solves the type-3 linear matrix inequality problem. Mathematical and implementation details can be found in the vignette

Value

X A list containing the solution matrix to the primal problem
y A list containing the solution vector to the dual problem
Z A list containing the solution matrix to the dual problem
pobj The achieved value of the primary objective function
dobj The achieved value of the dual objective function

Examples

A <- matrix(c(-1,0,1,0,-2,1,0,0,-1),3,3)
B <- matrix(c(1,2,3,4,5,6), 2, 3)
G <- matrix(1,3,3)
out <- 1mi3(A,B,G)
**logcheby**

**Log Chebyshev Approximation**

**Description**

logcheby creates input for sqlp to solve the Chebyshev Approximation Problem

**Usage**

logcheby(B, f)

**Arguments**

- B: A pxm real valued matrix with p > m
- f: A vector of length p

**Details**

Solves the log Chebyshev approximation problem. Mathematical and implementation details can be found in the vignette

**Value**

- X: A list containing the solution matrix to the primal problem
- y: A list containing the solution vector to the dual problem
- Z: A list containing the solution matrix to the dual problem
- pobj: The achieved value of the primary objective function
- dobj: The achieved value of the dual objective function

**Examples**

data(Blogcheby)
data(flogcheby)

#Not Run
#out <- logcheby(Blogcheby, flogcheby)
**lovasz**  
*Lovasz Number of a Graph*

**Description**

`lovasz` creates input for `sqlp` to find the Lovasz Number of a graph

**Usage**

`lovasz(G)`

**Arguments**

`G`  
An adjacency matrix corresponding to a graph

**Details**

Finds the maximum Shannon entropy of a graph, more commonly known as the Lovasz number. Mathematical and implementation details can be found in the vignette

**Value**

- `X`  
A list containing the solution matrix to the primal problem  
- `y`  
A list containing the solution vector to the dual problem  
- `Z`  
A list containing the solution matrix to the dual problem  
- `pobj`  
The achieved value of the primary objective function  
- `dobj`  
The achieved value of the dual objective function

**Examples**

```r
data(Glovasz)
out <- lovasz(Glovasz)
```

**maxcut**  
*Max-Cut Problem*

**Description**

`maxcut` creates input for `sqlp` to solve the Max-Cut problem - given a graph B, find the maximum cut of the graph

**Usage**

`maxcut(B)`

**Arguments**

`B`  
A graph represented as an adjacency matrix

**Details**

This function finds the maximum cut of a graph using the `sqlp` solver. The example demonstrates how to use it with a graph `Glovasz`. The function returns a list containing various outputs, including the achieved values of both the primary and dual objective functions.

```r
data(Glovasz)
out <- maxcut(Glovasz)
```
maxkcut

Arguments

B A (weighted) adjacency matrix corresponding to a graph

Details

Determines the maximum cut for a graph B. Mathematical and implementation details can be found in the vignette

Value

X A list containing the solution matrix to the primal problem
y A list containing the solution vector to the dual problem
Z A list containing the solution matrix to the dual problem
pobj The achieved value of the primary objective function
dobj The achieved value of the dual objective function

Examples

data(Bmaxcut)
out <- maxcut(Bmaxcut)

maxkcut

Max-kCut Problem

Description

maxkcut creates input for sqlp to solve the Max-kCut Problem - given a graph object B, determine if a cut of at least size k exists.

Usage

maxkcut(B, K)

Arguments

B A (weighted) adjacency matrix
K An integer value, the minimum number of cuts in B

Details

Determines if a cut of at least size k exists for a graph B. Mathematical and implementation details can be found in the vignette
**minelips**

**Value**
- \( X \) A list containing the solution matrix to the primal problem
- \( y \) A list containing the solution vector to the dual problem
- \( Z \) A list containing the solution matrix to the dual problem
- \( \text{pobj} \) The achieved value of the primary objective function
- \( \text{dobj} \) The achieved value of the dual objective function

**Examples**

```r
data(Bmaxkcut)
out <- maxkcut(Bmaxkcut,2)
```

---

**minelips**  
*The Minimum Ellipsoid Problem*

**Description**

`minelips` creates input for sqpl to solve the minimum ellipsoid problem - given a set of \( n \) points, find the minimum volume ellipsoid that contains all the points.

**Usage**

`minelips(V)`

**Arguments**

- \( V \) An \( nxp \) matrix consisting of the points to be contained in the ellipsoid

**Details**

For a set of points \((x_1, ..., x_n)\) determines the ellipse of minimum volume that contains all points. Mathematical and implementation details can be found in the vignette

**Value**

- \( X \) A list containing the solution matrix to the primal problem
- \( y \) A list containing the solution vector to the dual problem
- \( Z \) A list containing the solution matrix to the dual problem
- \( \text{pobj} \) The achieved value of the primary objective function
- \( \text{dobj} \) The achieved value of the dual objective function
nearcorr

**Nearst Correlation Matrix Problem**

**Description**

nearcorr creates input for sqlp to solve the nearest correlation matrix problem - given a approximate correlation matrix H, find the nearest correlation matrix X.

**Usage**

nearcorr(H)

**Arguments**

- **H**
  A symmetric matrix

**Details**

For a given approximate correlation matrix H, determines the nearest correlation matrix X. Mathematical and implementation details can be found in the vignette.

**Value**

- **X**
  A list containing the solution matrix to the primal problem
- **y**
  A list containing the solution vector to the dual problem
- **Z**
  A list containing the solution matrix to the dual problem
- **pobj**
  The achieved value of the primary objective function
- **dobj**
  The achieved value of the dual objective function

**Examples**

data(Hnearcorr)

out <- nearcorr(Hnearcorr)
Create a Symmetric Matrix

Description

smat takes a vector and creates a symmetric matrix.

Usage

smat(blk, p, At, isspM = NULL)

Arguments

- **blk**: Lx2 matrix detailing the type of matrices ("s", "q", "l", "u"), and the size of each matrix.
- **p**: Row of blk to be used during matrix creation.
- **At**: Vector to be turned into a symmetric matrix.
- **isspM**: If At is sparse, isspM = 1, 0 otherwise. Default is to assume M is dense.

Value

- **M**: A Symmetric Matrix.

Examples

```r
y <- c(1, 0.00000279, 3.245, 2.140, 2.44, 2.321, 4.566)
blk <- matrix(list(), 1, 2)
blk[[1,1]] <- "s"
blk[[1,2]] <- 3
P <- smat(blk, 1, y)
```

Semidefinite Quadratic Linear Programming Solver

Description

sqlp solves a semidefinite quadratic linear programming problem using the SDPT3 algorithm of Toh et. al. (1999) returning both the primal solution X and dual solution Z.

Usage

sqlp(blk = NULL, At = NULL, C = NULL, b = NULL, control = NULL, 
X0 = NULL, y0 = NULL, Z0 = NULL)
Arguments

- **blk**: A named-list object describing the block diagonal structure of the SQLP data
- **At**: A list object containing constraint matrices for the primal-dual problem
- **C**: A list object containing the constant $c$ matrices in the primal objective function
- **b**: A vector containing the right hand side of the equality constraints in the primal problem
- **control**: A list object specifying the values of certain parameters. If not provided, default values are used
- **X0**: An initial iterate for the primal solution variable $X$. If not provided, an initial iterate is computed internally.
- **y0**: An initial iterate for the dual solution variable $y$. If not provided, an initial iterate is computed internally.
- **Z0**: An initial iterate for the dual solution variable $Z$. If not provided, an initial iterate is computed internally.

Details

A full mathematical description of the problem to be solved, details surrounding the input variables, and discussion regarding the output variables can be found in the accompanying vignette.

Value

- **X**: A list containing the solution matrix to the primal problem
- **y**: The solution vector to the dual problem
- **Z**: A list containing the solution matrix to the dual problem
- **pobj**: The achieved value of the primary objective function
- **dobj**: The achieved value of the dual objective function

References


Examples

```r
blk = c("l" = 2)
C = matrix(c(1,1),nrow=1)
A = matrix(c(1,3,4,-1), nrow=2)
At = t(A)
b = c(12,10)
out = sqlp(blk,list(At),list(C),b)
```
svec

Upper Triangular Vectorization

Description

svec takes the upper triangular matrix (including the diagonal) and vectorizes it column-wise.

Usage

svec(blk, M, isspx = NULL)

Arguments

blk 1x2 matrix detailing the type of matrix ("s", "q", "l", "u"), and the size of the matrix
M matrix which is to be vectorized
isspx if M is sparse, isspx = 1, 0 otherwise. Default is to assume M is dense.

Value

x vector of upper triangular components of x

Examples

data(Hnearcorr)
blk <- matrix(list(),1,2)
blk[[1]] <- "s"
blk[[2]] <- nrow(Hnearcorr)
svec(blk,Hnearcorr)

toep

Toeplitz Approximation Problem

Description

toep creates input for sqlp to solve the Toeplitz approximation problem - given a symmetric matrix F, find the nearest symmetric positive definite Toeplitz matrix.

Usage

toep(A)
Arguments

A

A symmetric matrix

Details

For a symmetric matrix A, determines the closest Toeplitz matrix. Mathematical and implementation details can be found in the vignette.

Value

X

A list containing the solution matrix to the primal problem

y

A list containing the solution vector to the dual problem

Z

A list containing the solution matrix to the dual problem

pobj

The achieved value of the primary objective function

dobj

The achieved value of the dual objective function

Examples

data(Ftoep)

#Not Run
#out <- toep(Ftoep)

Vminelips

Configuration Matrix for Minimum Ellipse Problem

Description

Configuration Matrix for Minimum Ellipse Problem

Usage

data(Vminelips)

Format

A matrix with 2 rows and 2 columns
Index

* datasets
  Andwd, 2
  Apdwd, 3
  Betp, 3
  Bgpp, 3
  Blogcheby, 4
  Bmaxcut, 4
  Bmaxkcut, 4
  DoptDesign, 6
  flogcheby, 9
  Ftoep, 9
  Glovasz, 9
  Hnearcorr, 11
  Vminelips, 22

Andwd, 2
Apdwd, 3

Betp, 3
Bgpp, 3
Blogcheby, 4
Bmaxcut, 4
Bmaxkcut, 4

c control_theory, 5
DoptDesign, 6
doptimal, 6
dwd, 7

etp, 8
flogcheby, 9
Ftoep, 9

Glovasz, 9
gpp, 10

Hnearcorr, 11
lmi1, 11

lmi2, 12
lmi3, 13
logcheby, 14
lovasz, 15

maxcut, 15
maxkcut, 16
minelips, 17

nearcorr, 18

smat, 19
sqlp, 19
svec, 21

toep, 21

Vminelips, 22