Package ‘qpmadr’

June 23, 2021

Type Package
Title Interface to the 'qpmad' Quadratic Programming Solver
Version 1.1.0-0
Date 2021-06-23
License GPL (>= 3)
URL https://github.com/anderic1/qpmadr
BugReports https://github.com/anderic1/qpmadr/issues
Depends R (>= 3.0.2)
Imports Rcpp, checkmate
LinkingTo Rcpp, RcppEigen (>= 0.3.3.3.0)
RoxygenNote 7.1.1
Encoding UTF-8
Suggests tinytest
NeedsCompilation yes
Author Eric Anderson [aut, cre],
 Alexander Sherikov [cph, ctb]
Maintainer Eric Anderson <anderic1@gmx.com>
Repository CRAN
Date/Publication 2021-06-23 10:00:02 UTC

R topics documented:

qpmadParameters .......................................................... 2
solveqp ................................................................. 3

Index 5
qpmadParameters

Set qpmad parameters

Description
Conveniently set qpmad parameters. Please always use named arguments since parameters can change without notice between releases. In a future version specifying the argument names will be mandatory.

Usage
```
qpmadParameters(
    isFactorized = FALSE,
    maxIter = -1,
    tol = 1e-12,
    checkPD = TRUE,
    factorizationType = "NONE",
    withLagrMult = FALSE,
    returnInvCholFac = FALSE
)
```

Arguments
- **isFactorized** Deprecated, will be removed in a future version. Please use `factorizationType` instead. If `TRUE` then $H$ is a lower Cholesky factor, overridden by `factorizationType`.
- **maxIter** Maximum number of iterations, if not positive then no limit.
- **tol** Convergence tolerance.
- **checkPD** Deprecated. Ignored, will be removed in a future release.
- **factorizationType**
  - IF "NONE" then $H$ is a Hessian (default), if "CHOLESKY" then $H$ is a (lower) cholesky factor. If "INV_CHOLESKY" then $H$ is the inverse of a cholesky factor, i.e. such that the Hessian is given by inv($HH'$).
- **withLagrMult** If TRUE then the Lagrange multipliers of the inequality constraints, along with their indexes and an upper / lower side indicator, will be returned.
- **returnInvCholFac** If TRUE then also return the inverse Cholesky factor of the Hessian.

Value
a list suitable to be used as the pars-argument to `solveqp`

See Also
- `solveqp`
solveqp

Examples

qpmadParameters(withLagrMult = TRUE)

solveqp          Quadratic Programming

Description

Solves

\[
\argmin x' H x + h' x
\]

s.t.

\[
\text{lb}_i \leq x_i \leq \text{ub}_i
\]

\[
\text{Alb}_i \leq ( Ax)_i \leq \text{Aub}_i
\]

Usage

solveqp(
    H,
    h = NULL,
    lb = NULL,
    ub = NULL,
    A = NULL,
    Alb = NULL,
    Aub = NULL,
    pars = list()
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>Symmetric positive definite matrix, n*n. Can also be a (inverse) Cholesky factor cf. qpmadParameters.</td>
</tr>
<tr>
<td>h</td>
<td>Optional, vector of length n.</td>
</tr>
<tr>
<td>lb, ub</td>
<td>Optional, lower/upper bounds of x. Will be repeated n times if length is one.</td>
</tr>
<tr>
<td>A</td>
<td>Optional, constraints matrix of dimension p*n, where each row corresponds to a constraint. For equality constraints let corresponding elements in Alb equal those in Aub</td>
</tr>
<tr>
<td>Alb, Aub</td>
<td>Optional, lower/upper bounds for Ax.</td>
</tr>
<tr>
<td>pars</td>
<td>Optional, qpmad-solver parameters, conveniently set with qpmadParameters</td>
</tr>
</tbody>
</table>
Value

At least one of \( lb \), \( ub \) or \( A \) must be specified. If \( A \) has been specified then also at least one of \( A_{lb} \) or \( A_{ub} \). Returns a list with elements \texttt{solution} (the solution vector), \texttt{status} (a status code) and \texttt{message} (a human readable message). If \texttt{status} = 0 the algorithm has converged. Possible status codes:

- 0: Ok
- -1: Numerical issue, matrix (probably) not positive definite
- 1: Inconsistent
- 2: Infeasible equality
- 3: Infeasible inequality
- 4: Maximal number of iterations

See Also

\qpmad\Parameters

Examples

```r
## Assume we want to minimize: -(0 5 0) \%\% b + 1/2 b^T b
## under the constraints: A^T b >= b0
## with b0 = (-8,2,0)^T
## and
## A = (-3 1 -2)
## ( 0 0 1)
## we can use solveqp as follows:
##
## Dmat <- diag(3)
dvec <- c(0,-5,0)
Amat <- t(matrix(c(-4,-3,0,2,1,0,0,-2,1),3,3))
bvec <- c(-8,2,0)
solveqp(Dmat,dvec,A=Amat,Alb=bvec)
```
Index

qpmadParameters, 2, 3, 4

solveqp, 2, 3