My Project

Generated by Doxygen 1.9.1

Chapter 1

Todo List

Member exact_linesearch (QOCOFloat *u, QOCOFloat *Du, QOCOFloat f, QOCOSolver *solver)

get exact_linesearch working for SOCs.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 QOCOCscMatrix Struct Reference

Compressed sparse column format matrices.

`#include <structs.h>`

Public Attributes

- QOCOInt m
- QOCOInt n
- QOCOInt nnz
- QOCOInt * i
- QOCOInt * p
- QOCOFloat * x

4.1.1 Detailed Description

Compressed sparse column format matrices.

4.1.2 Member Data Documentation

4.1.2.1 i

```
QOCOInt* QOCOCscMatrix::i
```

Row indices (length: nnz).

4.1.2.2 m

```
QOCOInt QOCOCscMatrix::m
```

Number of rows.

4.1.2.3 n

```
QOCOInt QOCOCscMatrix::n
```

Number of columns.

4.1.2.4 nnz

```
QOCOInt QOCOCscMatrix::nnz
```

Number of nonzero elements.

4.1.2.5 p

```
QOCOInt* QOCOCscMatrix::p
```

Column pointers (length: n+1).

4.1.2.6 x

```
QOCOFloat* QOCOCscMatrix::x
```

Data (length: nnz).

The documentation for this struct was generated from the following file:

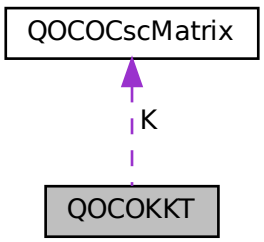
- /home/govind/Desktop/git/qoco/include/structs.h

4.2 QOCOKKT Struct Reference

Contains all data needed for constructing and modifying KKT matrix and performing predictor-corrector step.

```
#include <structs.h>
```

Collaboration diagram for QOCOKKT:



Public Attributes

- QOCOCscMatrix * K
- QOCOFloat * delta
- QOCOFloat * Druiz
- QOCOFloat * Eruiz
- QOCOFloat * Fruiz
- QOCOFloat * Dinvruiz
- QOCOFloat * Einvruiz
- QOCOFloat * Finvruiz
- QOCOFloat k
- QOCOFloat kinv
- QOCOInt * p
- QOCOInt * pinv
- QOCOInt * etree
- QOCOInt * Lnz
- QOCOFloat * Lx
- QOCOInt * Lp
- QOCOInt * Li
- QOCOFloat * D
- QOCOFloat * Dinv
- QOCOInt * iwork
- unsigned char * bwork
- QOCOFloat * fwork
- QOCOFloat * rhs
- QOCOFloat * xyz
- QOCOFloat * xyzbuff1
- QOCOFloat * xyzbuff2
- QOCOFloat * kktres
- QOCOInt * nt2kkt
- QOCOInt * ntdiag2kkt
- QOCOInt * PregtoKKT
- QOCOInt * Pnzadded_idx
- QOCOInt Pnum_nzadded
- QOCOInt * AtoKKT
- QOCOInt * GtoKKT

4.2.1 Detailed Description

Contains all data needed for constructing and modifying KKT matrix and performing predictor-corrector step.

4.2.2 Member Data Documentation

4.2.2.1 AtoKKT

```
QOCOInt* QOCOKKT::AtoKKT
```

Mapping from elements in A to elements in the KKT matrix.

4.2.2.2 bwork

```
unsigned char* QOCOKKT::bwork
```

4.2.2.3 D

```
QOCOFloat* QOCOKKT::D
```

4.2.2.4 delta

```
QOCOFloat* QOCOKKT::delta
```

Diagonal of scaling matrix.

4.2.2.5 Dinv

```
QOCOFloat* QOCOKKT::Dinv
```

4.2.2.6 Dinvruiz

```
QOCOFloat* QOCOKKT::Dinvruiz
```

Inverse of Druiz.

4.2.2.7 Druiz

```
QOCOFloat* QOCOKKT::Druiz
```

Diagonal of scaling matrix.

4.2.2.8 Einvruiz

```
QOCOFloat* QOCOKKT::Einvruiz
```

Inverse of Eruiz.

4.2.2.9 Eruiz

```
QOCOFloat* QOCOKKT::Eruiz
```

Diagonal of scaling matrix.

4.2.2.10 etree

```
QOCOInt* QOCOKKT::etree
```

Elimination tree for LDL factorization of K.

4.2.2.11 Finvruiz

```
QOCOFloat* QOCOKKT::Finvruiz
```

Inverse of Fruiz.

4.2.2.12 Fruiz

```
QOCOFloat* QOCOKKT::Fruiz
```

Diagonal of scaling matrix.

4.2.2.13 fwork

```
QOCOFloat* QOCOKKT::fwork
```

4.2.2.14 GtoKKT

```
QOCOInt* QOCOKKT::GtoKKT
```

Mapping from elements in G to elements in the KKT matrix.

4.2.2.15 iwork

```
QOCOInt* QOCOKKT::iwork
```

4.2.2.16 K

```
QOCOCscMatrix* QOCOKKT::K
```

KKT matrix in CSC form.

4.2.2.17 k

```
QOCOFloat QOCOKKT::k
```

Cost scaling factor.

4.2.2.18 kinv

```
QOCOFloat QOCOKKT::kinv
```

Inverse of cost scaling factor.

4.2.2.19 kktres

```
QOCOFloat* QOCOKKT::kktres
```

Residual of KKT condition.

4.2.2.20 Li

```
QOCOInt* QOCOKKT::Li
```

4.2.2.21 Lnz

```
QOCOInt* QOCOKKT::Lnz
```

4.2.2.22 Lp

```
QOCOInt* QOCOKKT::Lp
```

4.2.2.23 Lx

```
QOCOFloat* QOCOKKT::Lx
```

4.2.2.24 nt2kkt

```
QOCOInt* QOCOKKT::nt2kkt
```

Mapping from elements in the Nesterov-Todd scaling matrix to elements in the KKT matrix.

4.2.2.25 ntdiag2kkt

```
QOCOInt* QOCOKKT::ntdiag2kkt
```

Mapping from elements on the main diagonal of the Nesterov-Todd scaling matrices to elements in the KKT matrix. Used for regularization.

4.2.2.26 p

```
QOCOInt* QOCOKKT::p
```

Permutation vector.

4.2.2.27 pinv

```
QOCOInt* QOCOKKT::pinv
```

Inverse of permutation vector.

4.2.2.28 Pnum_nzadded

```
QOCOInt QOCOKKT::Pnum_nzadded
```

Number of elements of P->x that were added due to regularization.

4.2.2.29 Pnzadded_idx

```
QOCOInt* QOCOKKT::Pnzadded_idx
```

Indices of P->x that were added due to regularization.

4.2.2.30 PregtoKKT

```
QOCOInt* QOCOKKT::PregtoKKT
```

Mapping from elements in regularized P to elements in the KKT matrix.

4.2.2.31 rhs

```
QOCOFloat* QOCOKKT::rhs
```

RHS of KKT system.

4.2.2.32 xyz

```
QOCOFloat* QOCOKKT::xyz
```

Solution of KKT system.

4.2.2.33 xyzbuff1

```
QOCOFloat* QOCOKKT::xyzbuff1
```

Buffer of size n + m + p.

4.2.2.34 xyzbuff2

```
QOCOFloat* QOCOKKT::xyzbuff2
```

Buffer of size n + m + p.

The documentation for this struct was generated from the following file:

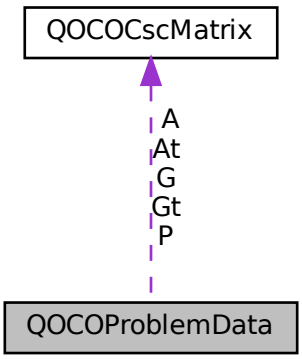
- /home/govind/Desktop/git/qoco/include/structs.h

4.3 QOCOProblemData Struct Reference

SOCP problem data.

```
#include <structs.h>
```

Collaboration diagram for QOCOProblemData:



Public Attributes

- QOCOCscMatrix * P
- QOCOFloat * c
- QOCOCscMatrix * A
- QOCOCscMatrix * At
- QOCOFloat * b
- QOCOCscMatrix * G
- QOCOCscMatrix * Gt
- QOCOFloat * h
- QOCOInt l
- QOCOInt nsoc
- QOCOInt * q
- QOCOInt n
- QOCOInt m
- QOCOInt p

4.3.1 Detailed Description

SOCP problem data.

4.3.2 Member Data Documentation

4.3.2.1 A

```
QOCOCscMatrix* QOCOProblemData::A
```

Affine equality constraint matrix.

4.3.2.2 At

```
QOCOCscMatrix* QOCOProblemData::At
```

Transpose of A (used in Ruiz for fast row norm calculations of A).

4.3.2.3 b

```
QOCOFloat* QOCOProblemData::b
```

Affine equality constraint offset.

4.3.2.4 c

```
QOCOFloat* QOCOProblemData::c
```

Linear cost term.

4.3.2.5 G

```
QOCOCscMatrix* QOCOProblemData::G
```

Conic constraint matrix.

4.3.2.6 Gt

```
QOCOCscMatrix* QOCOProblemData::Gt
```

Transpose of G (used in Ruiz for fast row norm calculations of G).

4.3.2.7 h

```
QOCOFloat* QOCOProblemData::h
```

Conic constraint offset.

4.3.2.8 l

```
QOCOInt QOCOProblemData::l
```

Dimension of non-negative orthant in cone C.

4.3.2.9 m

```
QOCOInt QOCOProblemData::m
```

Number of conic constraints.

4.3.2.10 n

```
QOCOInt QOCOProblemData::n
```

Number of primal variables.

4.3.2.11 nsoc

```
QOCOInt QOCOProblemData::nsoc
```

Number of second-order cones in C

4.3.2.12 P

```
QOCOCscMatrix* QOCOProblemData::P
```

Quadratic cost term.

4.3.2.13 p

```
QOCOInt QOCOProblemData::p
```

Number of affine equality constraints.

4.3.2.14 q

```
QOCOInt* QOCOProblemData::q
```

Dimension of each second-order cone (length of nsoc)

The documentation for this struct was generated from the following file:

- /home/govind/Desktop/git/qoco/include/structs.h

4.4 QOCOSettings Struct Reference

QOCO solver settings.

```
#include <structs.h>
```

Public Attributes

- QOCOInt max_iters
- QOCOInt bisect_iters
- QOCOInt ruiz_iters
- QOCOInt iter_ref_iters
- QOCOFloat kkt_static_reg
- QOCOFloat kkt_dynamic_reg
- QOCOFloat abstol
- QOCOFloat reltol
- QOCOFloat abstol_inacc
- QOCOFloat reltol_inacc
- unsigned char verbose

4.4.1 Detailed Description

QOCO solver settings.

4.4.2 Member Data Documentation

4.4.2.1 abstol

```
QOCOFloat QOCOSettings::abstol
```

Absolute tolerance.

4.4.2.2 abstol_inacc

```
QOCOFloat QOCOSettings::abstol_inacc
```

Low tolerance stopping criteria.

4.4.2.3 bisect_iters

```
QOCOInt QOCOSettings::bisect_iters
```

Number of bisection iterations for linesearch.

4.4.2.4 iter_ref_iters

```
QOCOInt QOCOSettings::iter_ref_iters
```

Number of iterative refinement iterations performed.

4.4.2.5 kkt_dynamic_reg

```
QOCOFloat QOCOSettings::kkt_dynamic_reg
```

Dynamic regularization parameter for KKT system.

4.4.2.6 kkt_static_reg

```
QOCOFloat QOCOSettings::kkt_static_reg
```

Static regularization parameter for KKT system.

4.4.2.7 max_iters

```
QOCOInt QOCOSettings::max_iters
```

Maximum number of IPM iterations.

4.4.2.8 reltol

```
QOCOFloat QOCOSettings::reltol
```

Relative tolerance.

4.4.2.9 reltol_inacc

```
QOCOFloat QOCOSettings::reltol_inacc
```

Low tolerance stopping criteria.

4.4.2.10 ruiz_iters

```
QOCOInt QOCOSettings::ruiz_iters
```

Number of Ruiz equilibration iterations.

4.4.2.11 verbose

```
unsigned char QOCOSettings::verbose
```

0 for quiet anything else for verbose.

The documentation for this struct was generated from the following file:

- /home/govind/Desktop/git/qoco/include/structs.h

4.5 QOCOSolution Struct Reference

```
#include <structs.h>
```

Public Attributes

- QOCOFloat * x
- QOCOFloat * s
- QOCOFloat * y
- QOCOFloat * z
- QOCOInt iters
- QOCOFloat setup_time_sec
- QOCOFloat solve_time_sec
- QOCOFloat obj
- QOCOFloat pres
- QOCOFloat dres
- QOCOFloat gap
- QOCOInt status

4.5.1 Member Data Documentation

4.5.1.1 dres

```
QOCOFloat QOCOSolution::dres
```

Dual residual.

4.5.1.2 gap

```
QOCOFloat QOCOSolution::gap
```

Duality gap.

4.5.1.3 iters

```
QOCOInt QOCOSolution::iters
```

Number of iterations.

4.5.1.4 obj

```
QOCOFloat QOCOSolution::obj
```

Optimal objective value.

4.5.1.5 pres

```
QOCOFloat QOCOSolution::pres
```

Primal residual.

4.5.1.6 s

```
QOCOFloat* QOCOSolution::s
```

Slack variable for conic constraints.

4.5.1.7 setup_time_sec

```
QOCOFloat QOCOSolution::setup_time_sec
```

Setup time.

4.5.1.8 solve_time_sec

```
QOCOFloat QOCOSolution::solve_time_sec
```

Solve time.

4.5.1.9 status

```
QOCOInt QOCOSolution::status
```

Solve status.

4.5.1.10 x

```
QOCOFloat* QOCOSolution::x
```

Primal solution.

4.5.1.11 y

```
QOCOFloat* QOCOSolution::y
```

Dual variables for affine equality constraints.

4.5.1.12 z

```
QOCOFloat* QOCOSolution::z
```

Dual variables for conic constraints.

The documentation for this struct was generated from the following file:

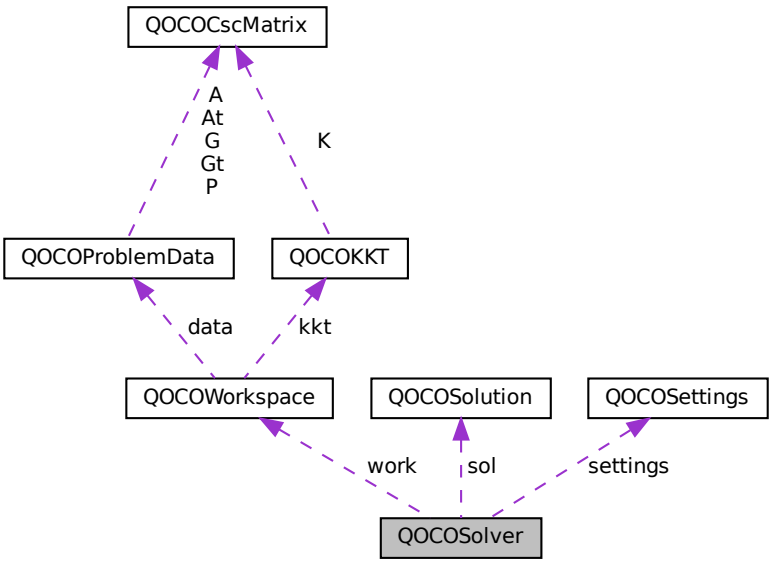
- /home/govind/Desktop/git/qoco/include/structs.h

4.6 QOCOSolver Struct Reference

QOCO Solver struct. Contains all information about the state of the solver.

```
#include <structs.h>
```

Collaboration diagram for QOCOSolver:



Public Attributes

- QOCOSettings * settings
- QOCOWorkspace * work
- QOCOSolution * sol

4.6.1 Detailed Description

QOCO Solver struct. Contains all information about the state of the solver.

4.6.2 Member Data Documentation

4.6.2.1 settings

```
QOCOSettings* QOCOSolver::settings
```

Solver settings.

4.6.2.2 sol

```
QOCOSolution* QOCOSolver::sol
```

Solution struct.

4.6.2.3 work

```
QOCOWorkspace* QOCOSolver::work
```

Solver workspace.

The documentation for this struct was generated from the following file:

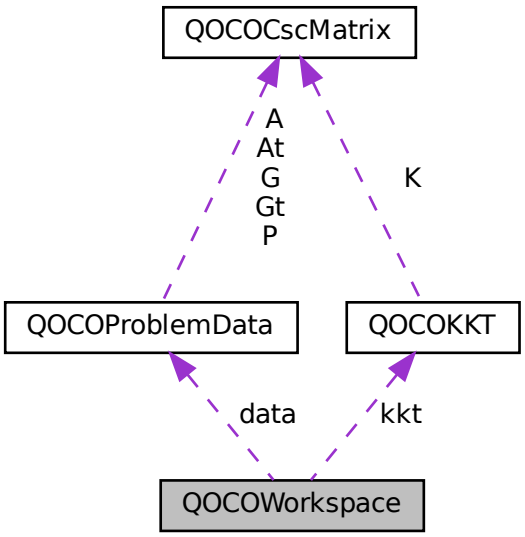
- /home/govind/Desktop/git/qoco/include/structs.h

4.7 QOCOWorkspace Struct Reference

QOCO Workspace.

`#include <structs.h>`

Collaboration diagram for QOCOWorkspace:



Public Attributes

- QOCOProblemData * data
- QOCOTimer solve_timer
- QOCOKKT * kkt
- QOCOFloat * x
- QOCOFloat * s
- QOCOFloat * y
- QOCOFloat * z
- QOCOFloat mu
- QOCOFloat a
- QOCOFloat sigma
- QOCOInt Wnnz
- QOCOInt Wnnzfull
- QOCOFloat * W
- QOCOFloat * Wfull
- QOCOFloat * Winv
- QOCOFloat * Winvfull
- QOCOFloat * WtW
- QOCOFloat * lambda
- QOCOFloat * sbar
- QOCOFloat * zbar

- QOCOFloat * xbuff
- QOCOFloat * ybuff
- QOCOFloat * ubuff1
- QOCOFloat * ubuff2
- QOCOFloat * ubuff3
- QOCOFloat * Ds

4.7.1 Detailed Description

QOCO Workspace.

4.7.2 Member Data Documentation

4.7.2.1 a

```
QOCOFloat QOCOWorkspace::a
```

Newton Step-size

4.7.2.2 data

```
QOCOProblemData* QOCOWorkspace::data
```

Contains SOCP problem data.

4.7.2.3 Ds

```
QOCOFloat* QOCOWorkspace::Ds
```

Search direction for slack variables. Length of m.

4.7.2.4 kkt

```
QOCOKKT* QOCOWorkspace::kkt
```

Contains all data related to KKT system.

4.7.2.5 lambda

```
QOCOFloat* QOCOWorkspace::lambda
```

Scaled variables.

4.7.2.6 mu

```
QOCOFloat QOCOWorkspace::mu
```

Gap (s'∗z / m)

4.7.2.7 s

```
QOCOFloat* QOCOWorkspace::s
```

Iterate of slack variables associated with conic constraint.

4.7.2.8 sbar

```
QOCOFloat* QOCOWorkspace::sbar
```

Temporary array needed in Nesterov-Todd scaling calculations. Length of max(q).

4.7.2.9 sigma

```
QOCOFloat QOCOWorkspace::sigma
```

Centering parameter

4.7.2.10 solve_timer

```
QOCOTimer QOCOWorkspace::solve_timer
```

Solve timer.

4.7.2.11 ubuff1

```
QOCOFloat* QOCOWorkspace::ubuff1
```

Temporary variable of length m.

4.7.2.12 ubuff2

```
QOCOFloat* QOCOWorkspace::ubuff2
```

Temporary variable of length m.

4.7.2.13 ubuff3

```
QOCOFloat* QOCOWorkspace::ubuff3
```

Temporary variable of length m.

4.7.2.14 W

```
QOCOFloat* QOCOWorkspace::W
```

Upper triangular part of Nesterov-Todd Scaling

4.7.2.15 Wfull

```
QOCOFloat* QOCOWorkspace::Wfull
```

Full Nesterov-Todd Scaling

4.7.2.16 Winv

```
QOCOFloat* QOCOWorkspace::Winv
```

Upper triangular part of inverse of Nesterov-Todd Scaling

4.7.2.17 Winvfull

```
QOCOFloat* QOCOWorkspace::Winvfull
```

Full inverse of Nesterov-Todd Scaling

4.7.2.18 Wnnz

```
QOCOInt QOCOWorkspace::Wnnz
```

Number of nonzeros in upper triangular part of Nesterov-Todd Scaling.

4.7.2.19 Wnnzfull

```
QOCOInt QOCOWorkspace::Wnnzfull
```

Number of nonzeros in full Nesterov-Todd Scaling.

4.7.2.20 WtW

```
QOCOFloat* QOCOWorkspace::WtW
```

Nesterov-Todd Scaling squared

4.7.2.21 x

```
QOCOFloat* QOCOWorkspace::x
```

Iterate of primal variables.

4.7.2.22 xbuff

```
QOCOFloat* QOCOWorkspace::xbuff
```

Temporary variable of length n.

4.7.2.23 y

```
QOCOFloat* QOCOWorkspace::y
```

Iterate of dual variables associated with affine equality constraint.

4.7.2.24 ybuff

```
QOCOFloat* QOCOWorkspace::ybuff
```

Temporary variable of length p.

4.7.2.25 z

```
QOCOFloat* QOCOWorkspace::z
```

Iterate of dual variables associated with conic constraint.

4.7.2.26 zbar

```
QOCOFloat* QOCOWorkspace::zbar
```

Temporary array needed in Nesterov-Todd scaling calculations. Length of max(q).

The documentation for this struct was generated from the following file:

- /home/govind/Desktop/git/qoco/include/structs.h

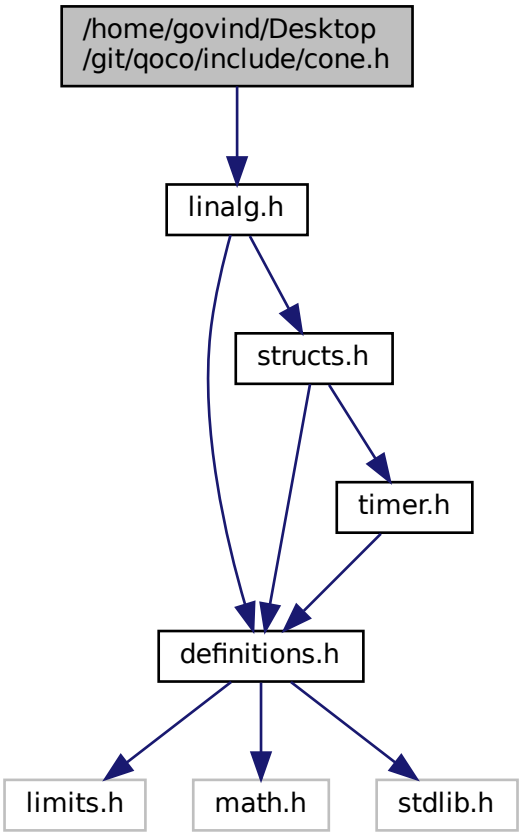
Chapter 5

File Documentation

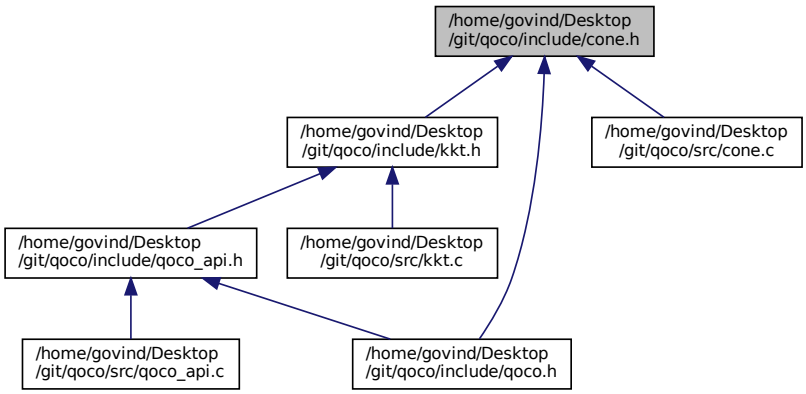
5.1 /home/govind/Desktop/git/qoco/include/cone.h File Reference

```
#include "linalg.h"
```

Include dependency graph for cone.h:



This graph shows which files directly or indirectly include this file:



Functions

- void soc_product (const QOCOFloat *u, const QOCOFloat *v, QOCOFloat *p, QOCOInt n)

  *Computes second-order cone product u * v = p.*
- void soc_division (const QOCOFloat *lam, const QOCOFloat *v, QOCOFloat *d, QOCOInt n)

  Commpues second-order cone division lambda # v = d.
- QOCOFloat soc_residual (const QOCOFloat *u, QOCOInt n)

  Computes residual of vector u with respect to the second order cone of dimension n.
- QOCOFloat soc_residual2 (const QOCOFloat *u, QOCOInt n)

  *Computes u0^2 - u1'*u1 of vector u with respect to the second order cone of dimension n.*
- void cone_product (const QOCOFloat *u, const QOCOFloat *v, QOCOFloat *p, QOCOInt l, QOCOInt nsoc, const QOCOInt *q)

  *Computes cone product u * v = p with respect to C.*
- void cone_division (const QOCOFloat *lambda, const QOCOFloat *v, QOCOFloat *d, QOCOInt l, QOCOInt nsoc, const QOCOInt *q)

  Computed cone division lambda # v = d.
- QOCOFloat cone_residual (const QOCOFloat *u, QOCOInt l, QOCOInt nsoc, const QOCOInt *q)

  Computes residual of vector u with respect to cone C.
- void bring2cone (QOCOFloat *u, QOCOProblemData *data)

  *Performs u = u + (1 + a) * e where e is the cannonical vector for each cone LP Cone: e = ones(n), second-order cone: e = (1,0,0,...) and a is the minimum scalar value such that u + (1 + a) * e is in cone C.*
- void nt_multiply (QOCOFloat *W, QOCOFloat *x, QOCOFloat *z, QOCOInt l, QOCOInt m, QOCOInt nsoc, QOCOInt *q)

  *Computes z = W * x where W is a full Nesterov-Todd scaling matrix. The NT scaling array for the LP cones are stored first, then the NT scalings for the second-order cones are stored in column major order.*
- void compute_mu (QOCOWorkspace *work)

  *Computes gap (z'*s / m) and stores in work->mu.*
- void compute_nt_scaling (QOCOWorkspace *work)

  Compute Nesterov-Todd scalings and scaled variables.
- void compute_centering (QOCOSolver *solver)

  Computes centering parameter.
- QOCOFloat linesearch (QOCOFloat *u, QOCOFloat *Du, QOCOFloat f, QOCOSolver *solver)

Conducts linesearch to compute a \in (0, 1] such that u + (a / f) ∗ Du \in C. For QPs this calls exact_linesearch() and for SOCPs this calls bisection_search()

- QOCOFloat bisection_search (QOCOFloat ∗u, QOCOFloat ∗Du, QOCOFloat f, QOCOSolver ∗solver)

  Conducts linesearch by bisection to compute a \in (0, 1] such that u + (a / f) ∗ Du \in C Warning: linesearch overwrites ubuff1. Do not pass in ubuff1 into u or Du. Consider a dedicated buffer for linesearch.

- QOCOFloat exact_linesearch (QOCOFloat ∗u, QOCOFloat ∗Du, QOCOFloat f, QOCOSolver ∗solver)

  Conducts exact linesearch to compute the largest a \in (0, 1] such that u + (a / f) ∗ Du \in C. Currently only works for LP cone.

5.1.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.1.2 LICENSE

Copyright (c) 2024, Govind M. Chari This source code is licensed under the BSD 3-Clause License

5.1.3 DESCRIPTION

Includes various functions necessary for cone operations.

5.1.4 Function Documentation

5.1.4.1 bisection_search()

```
QOCOFloat bisection_search (
            QOCOFloat * u,
            QOCOFloat * Du,
            QOCOFloat f,
            QOCOSolver * solver )
```

Conducts linesearch by bisection to compute a \in (0, 1] such that u + (a / f) ∗ Du \in C Warning: linesearch overwrites ubuff1. Do not pass in ubuff1 into u or Du. Consider a dedicated buffer for linesearch.

Parameters

u	Initial vector.
Du	Search direction.
f	Conservatism factor.
solver	Pointer to solver.

Returns

Step-size.

5.1.4.2 bring2cone()

```
void bring2cone (
            QOCOFloat * u,
            QOCOProblemData * data )
```

Performs u = u + (1 + a) $*$ e where e is the cannonical vector for each cone LP Cone: e = ones(n), second-order cone: e = (1,0,0,...) and a is the minimum scalar value such that u + (1 + a) $*$ e is in cone C.

Parameters

u	Vector to bring to cone.
data	Pointer to problem data.

5.1.4.3 compute_centering()

```
void compute_centering (
            QOCOSolver * solver )
```

Computes centering parameter.

Parameters

solver	Pointer to solver.

5.1.4.4 compute_mu()

```
void compute_mu (
            QOCOWorkspace * work )
```

Computes gap (z'$*$s / m) and stores in work->mu.

Parameters

work	Pointer to workspace.

5.1.4.5 compute_nt_scaling()

```
void compute_nt_scaling (
            QOCOWorkspace * work )
```

Compute Nesterov-Todd scalings and scaled variables.

Parameters

work	Pointer to workspace.

5.1.4.6 cone_division()

```
void cone_division (
            const QOCOFloat * lambda,
            const QOCOFloat * v,
            QOCOFloat * d,
            QOCOInt l,
            QOCOInt nsoc,
            const QOCOInt * q )
```

Computed cone division lambda # v = d.

Parameters

lambda	Input vector.
v	Input vector.
d	Cone quotient of lambda and v.
l	Dimension of LP cone.
nsoc	Number of second-order cones.
q	Dimension of each second-order cone.

5.1.4.7 cone_product()

```
void cone_product (
            const QOCOFloat * u,
            const QOCOFloat * v,
            QOCOFloat * p,
            QOCOInt l,
            QOCOInt nsoc,
            const QOCOInt * q )
```

Computes cone product u $*$ v = p with respect to C.

Parameters

u	Input vector.

Parameters

v	Input vector.
p	Cone product of u and v.
l	Dimension of LP cone.
nsoc	Number of second-order cones.
q	Dimension of each second-order cone.

5.1.4.8 cone_residual()

```
QOCOFloat cone_residual (
              const QOCOFloat * u,
              QOCOInt l,
              QOCOInt nsoc,
              const QOCOInt * q )
```

Computes residual of vector u with respect to cone C.

Parameters

u	Vector to be tested.
l	Dimension of LP cone.
nsoc	Number of second-order cones.
q	Dimension of each second-order cone.

Returns

Residual: Negative if the vector is in the cone and positive otherwise.

5.1.4.9 exact_linesearch()

```
QOCOFloat exact_linesearch (
              QOCOFloat * u,
              QOCOFloat * Du,
              QOCOFloat f,
              QOCOSolver * solver )
```

Conducts exact linesearch to compute the largest a \in (0, 1] such that u + (a / f) ∗ Du \in C. Currently only works for LP cone.

Todo get exact_linesearch working for SOCs.

Parameters

u	Initial vector.
Du	Search direction.
f	Conservatism factor.
solver	Pointer to solver.

Returns

Step-size.

5.1.4.10 linesearch()

```
QOCOFloat linesearch (
             QOCOFloat * u,
             QOCOFloat * Du,
             QOCOFloat f,
             QOCOSolver * solver )
```

Conducts linesearch to compute a \in (0, 1] such that u + (a / f) ∗ Du \in C. For QPs this calls exact_linesearch() and for SOCPs this calls bisection_search()

Parameters

u	Initial vector.
Du	Search direction.
f	Conservatism factor.
solver	Pointer to solver.

Returns

Step-size.

5.1.4.11 nt_multiply()

```
void nt_multiply (
             QOCOFloat * W,
             QOCOFloat * x,
             QOCOFloat * z,
             QOCOInt l,
             QOCOInt m,
             QOCOInt nsoc,
             QOCOInt * q )
```

Computes z = W ∗ x where W is a full Nesterov-Todd scaling matrix. The NT scaling array for the LP cones are stored first, then the NT scalings for the second-order cones are stored in column major order.

Parameters

W	Nesterov Todd scaling matrix.
x	Input vector.
z	Output vector.
l	Dimension of LP cone.
m	Length of x.
nsoc	Number of second-order cones in C.
q	Array of second-order cone dimensions.

5.1.4.12 soc_division()

```
void soc_division (
             const QOCOFloat * lam,
             const QOCOFloat * v,
             QOCOFloat * d,
             QOCOInt n )
```

Commpues second-order cone division lambda # v = d.

Parameters

lam	lam = (lam0, lam1) is a vector in second-order cone of dimension n.
v	v = (v0, v1) is a vector in second-order cone of dimension n.
d	Cone divisin of lam and v.
n	Dimension of second-order cone.

5.1.4.13 soc_product()

```
void soc_product (
             const QOCOFloat * u,
             const QOCOFloat * v,
             QOCOFloat * p,
             QOCOInt n )
```

Computes second-order cone product u $*$ v = p.

Parameters

u	u = (u0, u1) is a vector in second-order cone of dimension n.
v	v = (v0, v1) is a vector in second-order cone of dimension n.
p	Cone product of u and v.
n	Dimension of second-order cone.

5.1.4.14 soc_residual()

```
QOCOFloat soc_residual (
            const QOCOFloat * u,
            QOCOInt n )
```

Computes residual of vector u with respect to the second order cone of dimension n.

Parameters

u	u = (u0, u1) is a vector in second-order cone of dimension n.
n	Dimension of second order cone.

Returns

Residual: norm(u1) - u0. Negative if the vector is in the cone and positive otherwise.

5.1.4.15 soc_residual2()

```
QOCOFloat soc_residual2 (
            const QOCOFloat * u,
            QOCOInt n )
```

Computes u0^2 - u1'∗u1 of vector u with respect to the second order cone of dimension n.

Parameters

u	u = (u0, u1) is a vector in second order cone of dimension n.
n	Dimension of second order cone.

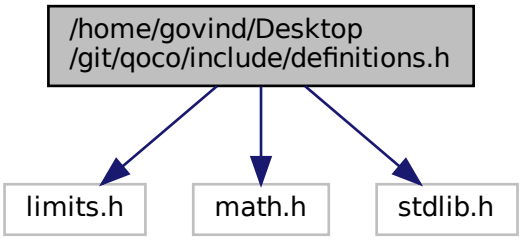
Returns

Residual: u0^2 - u1'∗u1.

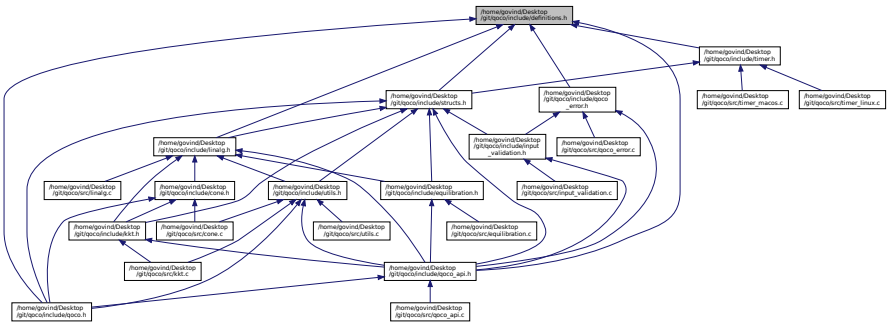
5.2 /home/govind/Desktop/git/qoco/include/definitions.h File Reference

```
#include <limits.h>
#include <math.h>
#include <stdlib.h>
```

Include dependency graph for definitions.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define QOCOInt_MAX INT_MAX
- #define QOCOFloat_MAX __DBL_MAX__
- #define qoco_max(a, b) (((a) > (b)) ? (a) : (b))
- #define qoco_min(a, b) (((a) < (b)) ? (a) : (b))
- #define qoco_abs(a) (((a) > 0) ? (a) : (-a))
- #define safe_div(a, b) (qoco_abs(b) > 1e-15) ? (a / b) : QOCOFloat_MAX
- #define qoco_sqrt(a) sqrt(a)
- #define qoco_assert(a)
- #define qoco_malloc malloc
- #define qoco_calloc calloc
- #define qoco_free free

Typedefs

- typedef int QOCOInt
- typedef double QOCOFloat

5.2.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.2.2 LICENSE

5.2.3 DESCRIPTION

Defines various macros used in qoco.

5.2.4 Macro Definition Documentation

5.2.4.1 qoco_abs

```
#define qoco_abs(
              a ) (((a) > 0) ?  (a) :  (-a))
```

5.2.4.2 qoco_assert

```
#define qoco_assert(
              a )
```

Value:

```
  do {                                                                \
  } while (0)
```

5.2.4.3 qoco_calloc

```
#define qoco_calloc calloc
```

5.2.4.4 qoco_free

```
#define qoco_free free
```

5.2.4.5 qoco_malloc

```
#define qoco_malloc malloc
```

5.2.4.6 qoco_max

```
#define qoco_max(
              a,
              b ) (((a) > (b)) ?  (a) :  (b))
```

5.2.4.7 qoco_min

```
#define qoco_min(
              a,
              b ) (((a) < (b)) ?  (a) :  (b))
```

5.2.4.8 qoco_sqrt

```
#define qoco_sqrt(
              a ) sqrt(a)
```

5.2.4.9 QOCOFloat_MAX

```
#define QOCOFloat_MAX __DBL_MAX__
```

5.2.4.10 QOCOInt_MAX

```
#define QOCOInt_MAX INT_MAX
```

5.2.4.11 safe_div

```
#define safe_div(
              a,
              b ) (qoco_abs(b) > 1e-15) ?  (a / b) :  QOCOFloat_MAX
```

5.2.5 Typedef Documentation

5.2.5.1 QOCOFloat

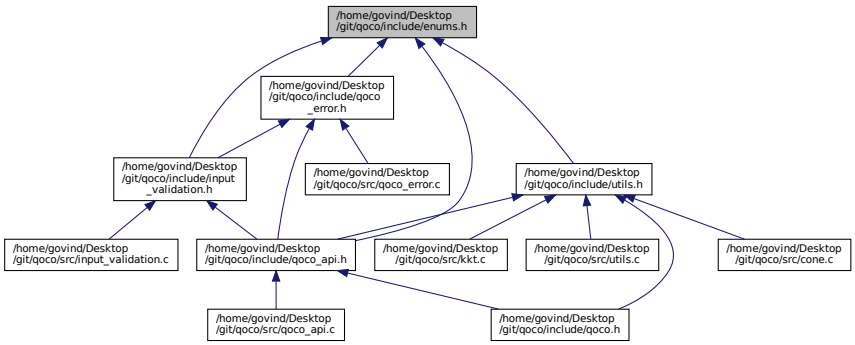
```
typedef double QOCOFloat
```

5.2.5.2 QOCOInt

```
typedef int QOCOInt
```

5.3 /home/govind/Desktop/git/qoco/include/enums.h File Reference

This graph shows which files directly or indirectly include this file:



Enumerations

- enum qoco_error_code {
  QOCO_NO_ERROR = 0 , QOCO_DATA_VALIDATION_ERROR , QOCO_SETTINGS_VALIDATION_ERROR , QOCO_SETUP_ERROR ,
  QOCO_AMD_ERROR , QOCO_MALLOC_ERROR }
  Enum for error codes.
- enum qoco_solve_status {
  QOCO_UNSOLVED = 0 , QOCO_SOLVED = 1 , QOCO_SOLVED_INACCURATE , QOCO_NUMERICAL_ERROR ,
  QOCO_MAX_ITER }
  Enum for solver status.

5.3.1 Enumeration Type Documentation

5.3.1.1 qoco_error_code

```
enum qoco_error_code
```

Enum for error codes.

Enumerator

QOCO_NO_ERROR	
QOCO_DATA_VALIDATION_ERROR	
QOCO_SETTINGS_VALIDATION_ERROR	
QOCO_SETUP_ERROR	
QOCO_AMD_ERROR	
QOCO_MALLOC_ERROR	

5.3.1.2 qoco_solve_status

```
enum qoco_solve_status
```

Enum for solver status.

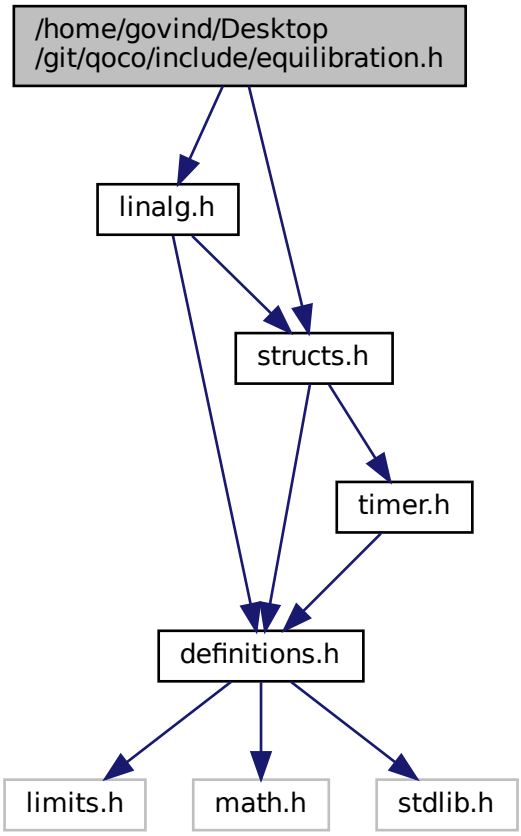
Enumerator

QOCO_UNSOLVED	
QOCO_SOLVED	
QOCO_SOLVED_INACCURATE	
QOCO_NUMERICAL_ERROR	
QOCO_MAX_ITER	

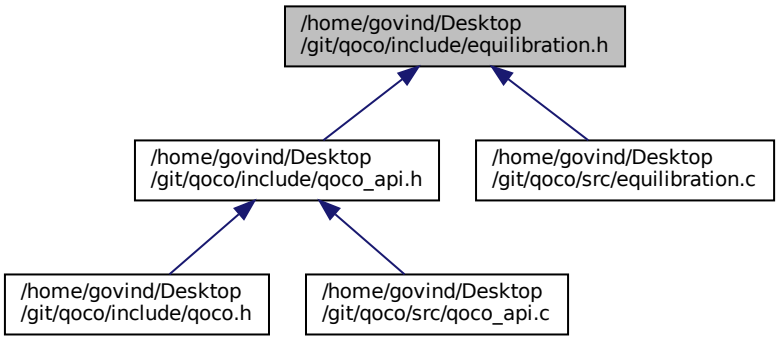
5.4 /home/govind/Desktop/git/qoco/include/equilibration.h File Reference

```
#include "linalg.h"
#include "structs.h"
```

Include dependency graph for equilibration.h:



This graph shows which files directly or indirectly include this file:



Functions

- void ruiz_equilibration (QOCOSolver *solver)

  Applies modified ruiz equilibration to scale data matrices. Computes D, E, F, and k as shown below to make the row and column infinity norms equal for the scaled KKT matrix.
- void unscale_variables (QOCOWorkspace *work)

  Undo variable transformation induced by ruiz equilibration.

5.4.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.4.2 LICENSE

5.4.3 DESCRIPTION

Provides functions to equilibrate problem data and scale variables.

5.4.4 Function Documentation

5.4.4.1 ruiz_equilibration()

```
void ruiz_equilibration (
            QOCOSolver * solver )
```

Applies modified ruiz equilibration to scale data matrices. Computes D, E, F, and k as shown below to make the row and column infinity norms equal for the scaled KKT matrix.

- clang-format off

[D] [kP A^T G^T] [D] | E | | A 0 0 | | E | [F] [G 0 0] [F]

clang-format on

Parameters

solver	Pointer to solver.

5.4.4.2 unscale_variables()

```
void unscale_variables (
            QOCOWorkspace * work )
```

Undo variable transformation induced by ruiz equilibration.

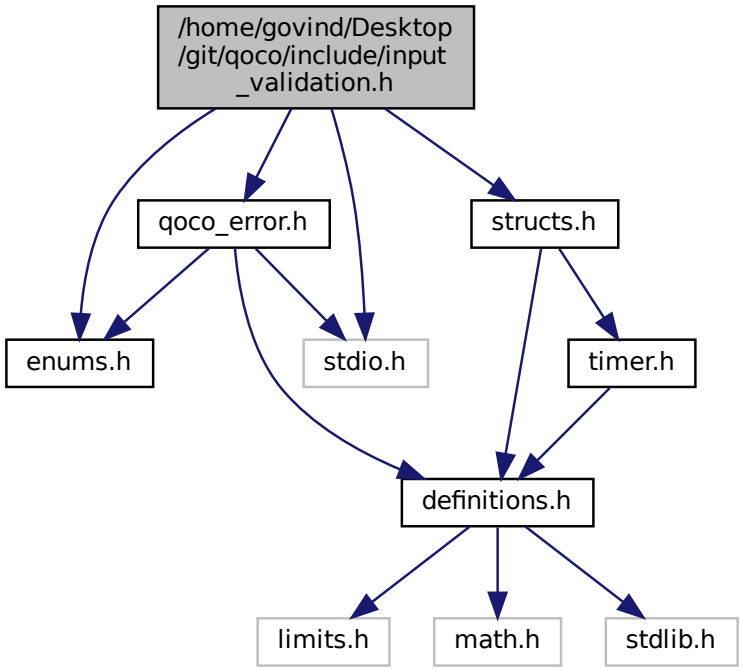
Parameters

work	Pointer to workspace.

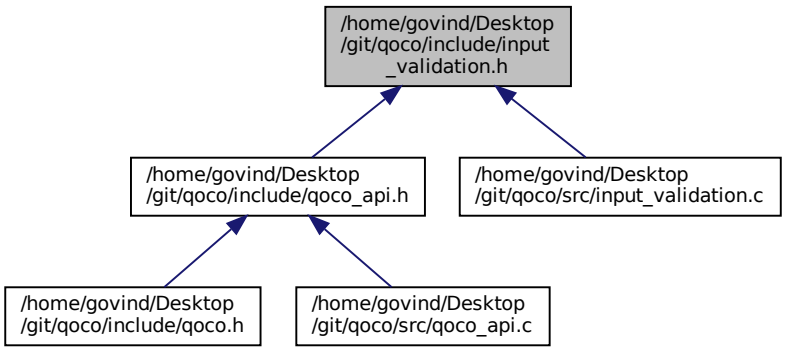
5.5 /home/govind/Desktop/git/qoco/include/input_validation.h File Reference

```
#include "enums.h"
#include "qoco_error.h"
#include "structs.h"
#include <stdio.h>
```

Include dependency graph for input_validation.h:



This graph shows which files directly or indirectly include this file:



Functions

- QOCOInt qoco_validate_settings (const QOCOSettings ∗settings)

  Validates solver settings.
- QOCOInt qoco_validate_data (const QOCOCscMatrix ∗P, const QOCOFloat ∗c, const QOCOCscMatrix ∗A, const QOCOFloat ∗b, const QOCOCscMatrix ∗G, const QOCOFloat ∗h, const QOCOInt l, const QOCOInt nsoc, const QOCOInt ∗q)

  Validate problem data.

5.5.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.5.2 LICENSE

Copyright (c) 2024, Govind M. Chari This source code is licensed under the BSD 3-Clause License

5.5.3 DESCRIPTION

Includes functions that validate any user-provided data.

5.5.4 Function Documentation

5.5.4.1 qoco_validate_data()

```
QOCOInt qoco_validate_data (
             const QOCOCscMatrix * P,
             const QOCOFloat * c,
             const QOCOCscMatrix * A,
             const QOCOFloat * b,
             const QOCOCscMatrix * G,
             const QOCOFloat * h,
             const QOCOInt l,
             const QOCOInt nsoc,
             const QOCOInt * q )
```

Validate problem data.

Parameters

P	Upper triangular part of quadratic cost Hessian in CSC form
c	Linear cost vector
A	Affine equality constraint matrix in CSC form
b	Affine equality constraint offset vector
G	Conic constraint matrix in CSC form
h	Conic constraint offset vector
l	Dimension of non-negative orthant
nsoc	Number of second-order cones
q	Dimension of each second-order cone

Returns

Exitflag to check (0 for success, failure otherwise)

5.5.4.2 qoco_validate_settings()

```
QOCOInt qoco_validate_settings (
             const QOCOSettings * settings )
```

Validates solver settings.

Parameters

settings	Pointer to settings struct

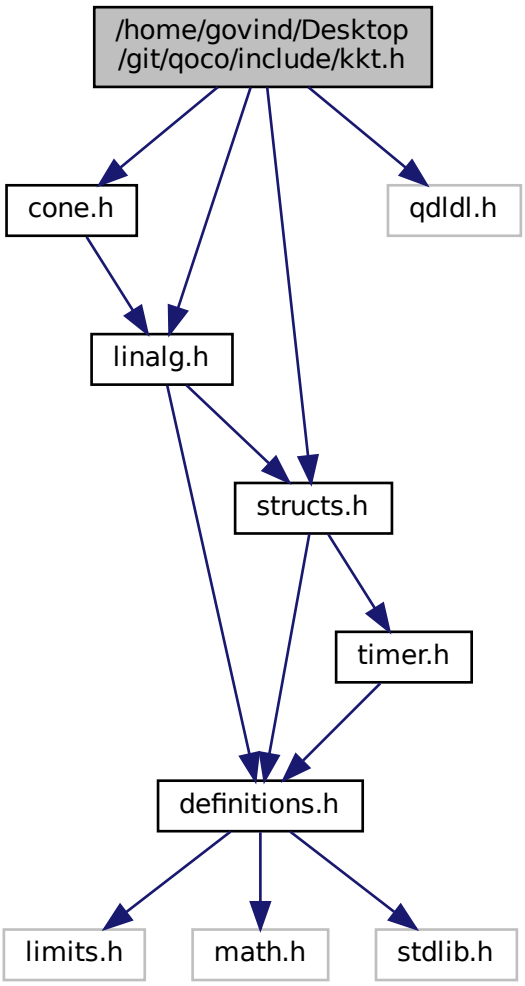
Returns

Exitflag to check (0 for success, failure otherwise)

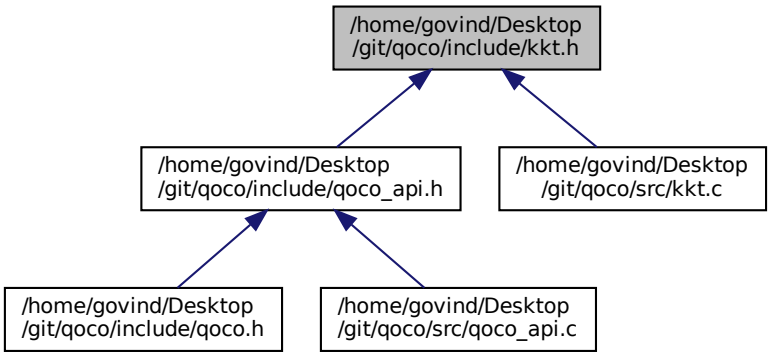
5.6 /home/govind/Desktop/git/qoco/include/kkt.h File Reference

```
#include "cone.h"
#include "linalg.h"
#include "qdldl.h"
#include "structs.h"
```

Include dependency graph for kkt.h:



This graph shows which files directly or indirectly include this file:



Functions

- void allocate_kkt (QOCOWorkspace *work)

  Allocate memory for KKT matrix.
- void construct_kkt (QOCOSolver *solver)

  Constructs upper triangular part of KKT matrix with -I for Nestrov-Todd scaling matrix (the (3,3) block)
- void initialize_ipm (QOCOSolver *solver)

  Gets initial values for primal and dual variables such that (s,z) \in C.
- void set_nt_block_zeros (QOCOWorkspace *work)

  Set the Nesterov-Todd block to be zeros. Used prior to compute_kkt_residual().
- void update_nt_block (QOCOSolver *solver)

  Updates and regularizes Nesterov-Todd scaling block of KKT matrix.
- void compute_kkt_residual (QOCOSolver *solver)

  Computes residual of KKT conditions and stores in work->kkt->rhs.
- void construct_kkt_aff_rhs (QOCOWorkspace *work)

  Constructs rhs for the affine scaling KKT system. Before calling this function, work->kkt->kktres must contain the residual of the KKT conditions as computed by compute_kkt_residual().
- void construct_kkt_comb_rhs (QOCOWorkspace *work)

  Constructs rhs for the combined direction KKT system. Before calling this function, work->kkt->kktres must contain the negative residual of the KKT conditions as computed by compute_kkt_residual().
- void predictor_corrector (QOCOSolver *solver)

  Performs Mehrotra predictor-corrector step.
- void kkt_solve (QOCOSolver *solver, QOCOFloat *b, QOCOInt iters)

  Solves Kx = b once K has been factored. Solves via triangular solves and applies iterative refinement afterwards.
- void kkt_multiply (QOCOSolver *solver, QOCOFloat *x, QOCOFloat *y)

  *Computes y = Kx where [P A^T G^T] K = | A 0 0 | [G 0 -W'W - e * I].*

5.6.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.6.2 LICENSE

5.6.3 DESCRIPTION

Provides various functions for solving, constructing and updating KKT systems.

5.6.4 Function Documentation

5.6.4.1 allocate_kkt()

```
void allocate_kkt (
            QOCOWorkspace * work )
```

Allocate memory for KKT matrix.

Parameters

work	Pointer to workspace.

5.6.4.2 compute_kkt_residual()

```
void compute_kkt_residual (
            QOCOSolver * solver )
```

Computes residual of KKT conditions and stores in work->kkt->rhs.

clang-format off

```
  [ P   A^T   G^T ] [ x ]   [    c   ]
```

res = | A 0 0 | | y] + | -b | [G 0 0] [z] [-h + s]

clang-format on

Parameters

solver	Pointer to solver.

5.6.4.3 construct_kkt()

```
void construct_kkt (
            QOCOSolver * solver )
```

Constructs upper triangular part of KKT matrix with -I for Nestrov-Todd scaling matrix (the (3,3) block)

clang-format off

```
[ P   A^T   G^T ]
```

K = | A 0 0 | [G 0 -I]

clang-format on

Parameters

solver	Pointer to solver

5.6.4.4 construct_kkt_aff_rhs()

```
void construct_kkt_aff_rhs (
            QOCOWorkspace * work )
```

Constructs rhs for the affine scaling KKT system. Before calling this function, work->kkt->kktres must contain the residual of the KKT conditions as computed by compute_kkt_residual().

Parameters

work	Pointer to workspace.

5.6.4.5 construct_kkt_comb_rhs()

```
void construct_kkt_comb_rhs (
            QOCOWorkspace * work )
```

Constructs rhs for the combined direction KKT system. Before calling this function, work->kkt->kktres must contain the negative residual of the KKT conditions as computed by compute_kkt_residual().

Parameters

work	Pointer to workspace.

ds = -cone_product(lambda, lambda) - settings.mehrotra ∗ cone_product((W' \ Dsaff), (W ∗ Dzaff), pdata) + sigma ∗ mu ∗ e.

5.6.4.6 initialize_ipm()

```
void initialize_ipm (
            QOCOSolver * solver )
```

Gets initial values for primal and dual variables such that (s,z) \in C.

Parameters

solver	Pointer to solver.

5.6.4.7 kkt_multiply()

```
void kkt_multiply (
            QOCOSolver * solver,
            QOCOFloat * x,
            QOCOFloat * y )
```

Computes y = Kx where [P A^T G^T] K = | A 0 0 | [G 0 -W'W - e * I].

Parameters

solver	Pointer to solver.
x	Pointer to input vector.
y	Pointer to output vector.

5.6.4.8 kkt_solve()

```
void kkt_solve (
            QOCOSolver * solver,
            QOCOFloat * b,
            QOCOInt iters )
```

Solves Kx = b once K has been factored. Solves via triangular solves and applies iterative refinement afterwards.

Parameters

solver	Pointer to solver.
b	Pointer to rhs of kkt system.
iters	Number of iterations of iterative refinement performed.

5.6.4.9 predictor_corrector()

```
void predictor_corrector (
            QOCOSolver * solver )
```

Performs Mehrotra predictor-corrector step.

Parameters

solver	Pointer to solver.

5.6.4.10 set_nt_block_zeros()

```
void set_nt_block_zeros (
            QOCOWorkspace * work )
```

Set the Nesterov-Todd block to be zeros. Used prior to compute_kkt_residual().

Parameters

work	Pointer to workspace.

5.6.4.11 update_nt_block()

```
void update_nt_block (
            QOCOSolver * solver )
```

Updates and regularizes Nesterov-Todd scaling block of KKT matrix.

```
[ P   A^T       G^T      ]
```

K = | A 0 0 | [G 0 -W'W - e ∗ I]

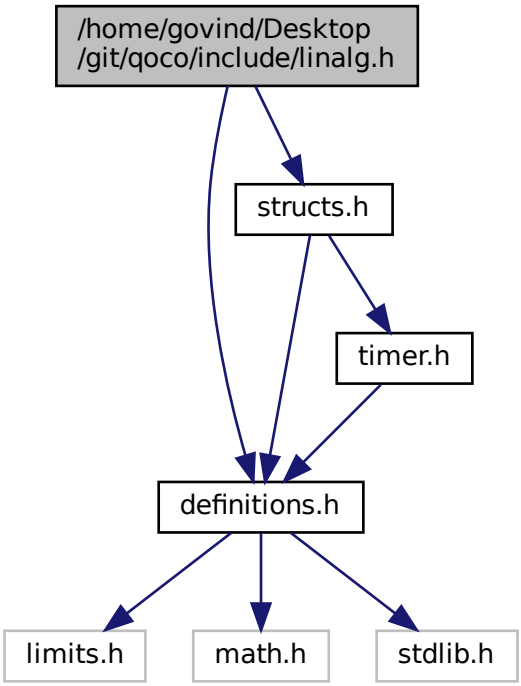
Parameters

solver	Pointer to solver.

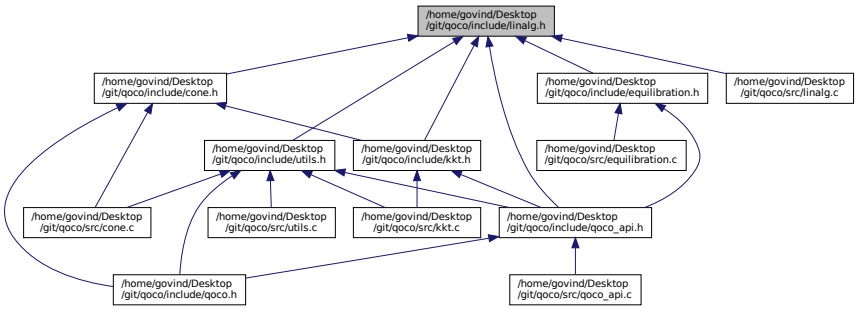
5.7 /home/govind/Desktop/git/qoco/include/linalg.h File Reference

```
#include "definitions.h"
#include "structs.h"
```

Include dependency graph for linalg.h:



This graph shows which files directly or indirectly include this file:



Functions

- QOCOCscMatrix * new_qoco_csc_matrix (const QOCOCscMatrix *A)

  Allocates a new csc matrix and copies A to it.
- QOCOCscMatrix * construct_identity (QOCOInt n, QOCOFloat lambda)

  *Allocates a new csc matrix that is lambda * I.*
- void free_qoco_csc_matrix (QOCOCscMatrix *A)

Frees all the internal arrays and the pointer to the QOCOCscMatrix. Should only be used if QOCOCscMatrix and all internal arrays were malloc'ed.

- void copy_arrayf (const QOCOFloat *x, QOCOFloat *y, QOCOInt n)

  Copies array of QOCOFloats from x to array y.
- void copy_and_negate_arrayf (const QOCOFloat *x, QOCOFloat *y, QOCOInt n)

  Copies and negates array of QOCOFloats from x to array y.
- void copy_arrayi (const QOCOInt *x, QOCOInt *y, QOCOInt n)

  Copies array of QOCOInts from x to array y.
- QOCOFloat dot (const QOCOFloat *u, const QOCOFloat *v, QOCOInt n)

  Computes dot product of u and v.
- QOCOInt max_arrayi (const QOCOInt *x, QOCOInt n)

  Computes maximum element of array of QOCOInts.
- void scale_arrayf (const QOCOFloat *x, QOCOFloat *y, QOCOFloat s, QOCOInt n)

  *Scales array x by s and stores result in y. y = s * x.*
- void axpy (const QOCOFloat *x, const QOCOFloat *y, QOCOFloat *z, QOCOFloat a, QOCOInt n)

  *Computes z = a * x + y.*
- void USpMv (const QOCOCscMatrix *M, const QOCOFloat *v, QOCOFloat *r)

  *Sparse matrix vector multiplication for CSC matrices where M is symmetric and only the upper triangular part is given. Computes r = M * v.*
- void SpMv (const QOCOCscMatrix *M, const QOCOFloat *v, QOCOFloat *r)

  *Sparse matrix vector multiplication for CSC matrices. Computes r = M * v.*
- void SpMtv (const QOCOCscMatrix *M, const QOCOFloat *v, QOCOFloat *r)

  *Sparse matrix vector multiplication for CSC matrices where M is first transposed. Computes r = M^T * v.*
- QOCOFloat inf_norm (const QOCOFloat *x, QOCOInt n)

  Computes the infinity norm of x.
- QOCOInt regularize (QOCOCscMatrix *M, QOCOFloat lambda, QOCOInt *nzadded_idx)

  *Adds lambda * I to a CSC matrix. Called on P prior to construction of KKT system in qoco_setup(). This function calls realloc() when adding new nonzeros.*
- void unregularize (QOCOCscMatrix *M, QOCOFloat lambda)

  *Subtracts lambda * I to a CSC matrix. Called on P when updating matrix data in update_matrix_data(). This function does not allocate and must be called after regularize.*
- void col_inf_norm_USymm (const QOCOCscMatrix *M, QOCOFloat *norm)

  Computes the infinity norm of each column (or equivalently row) of a symmetric sparse matrix M where only the upper triangular portion of M is given.
- void row_inf_norm (const QOCOCscMatrix *M, QOCOFloat *norm)

  Computes the infinity norm of each row of M and stores in norm.
- QOCOCscMatrix * create_transposed_matrix (const QOCOCscMatrix *A)

  Allocates and computes A^T.
- void row_col_scale (const QOCOCscMatrix *M, QOCOFloat *E, QOCOFloat *D)

  *Scales the rows of M by E and columns of M by D. M = diag(E) * M * diag(S)*
- void ew_product (QOCOFloat *x, const QOCOFloat *y, QOCOFloat *z, QOCOInt n)

  Computes elementwise product z = x . y.*
- void invert_permutation (const QOCOInt *p, QOCOInt *pinv, QOCOInt n)

  Inverts permutation vector p and stores inverse in pinv.
- QOCOInt cumsum (QOCOInt *p, QOCOInt *c, QOCOInt n)

  Computes cumulative sum of c.
- QOCOCscMatrix * csc_symperm (const QOCOCscMatrix *A, const QOCOInt *pinv, QOCOInt *AtoC)

  C = A(p,p) = PAP' where A and C are symmetric and the upper triangular part is stored.

5.7.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.7.2 LICENSE

5.7.3 DESCRIPTION

Provides various linear algebra operations.

5.7.4 Function Documentation

5.7.4.1 axpy()

```
void axpy (
            const QOCOFloat * x,
            const QOCOFloat * y,
            QOCOFloat * z,
            QOCOFloat a,
            QOCOInt n )
```

Computes z = a ∗ x + y.

Parameters

x	Input vector.
y	Input vector.
z	Result vector.
a	Scaling factor.
n	Length of vectors.

5.7.4.2 col_inf_norm_USymm()

```
void col_inf_norm_USymm (
            const QOCOCscMatrix * M,
            QOCOFloat * norm )
```

Computes the infinity norm of each column (or equivalently row) of a symmetric sparse matrix M where only the upper triangular portion of M is given.

Parameters

M	Upper triangular part of sparse symmetric matrix.
norm	Result vector of length n.

5.7.4.3 construct_identity()

```
QOCOCscMatrix* construct_identity (
             QOCOInt n,
             QOCOFloat lambda )
```

Allocates a new csc matrix that is lambda $*$ I.

Parameters

n	Size of identity matrix.
lambda	Scaling factor for identity.

Returns

Pointer to new constructed matrix.

5.7.4.4 copy_and_negate_arrayf()

```
void copy_and_negate_arrayf (
             const QOCOFloat * x,
             QOCOFloat * y,
             QOCOInt n )
```

Copies and negates array of QOCOFloats from x to array y.

Parameters

x	Source array.
y	Destination array.
n	Length of arrays.

5.7.4.5 copy_arrayf()

```
void copy_arrayf (
             const QOCOFloat * x,
```

```
            QOCOFloat * y,
            QOCOInt n )
```

Copies array of QOCOFloats from x to array y.

Parameters

x	Source array.
y	Destination array.
n	Length of arrays.

5.7.4.6 copy_arrayi()

```
void copy_arrayi (
            const QOCOInt * x,
            QOCOInt * y,
            QOCOInt n )
```

Copies array of QOCOInts from x to array y.

Parameters

x	Source array.
y	Destination array.
n	Length of arrays.

5.7.4.7 create_transposed_matrix()

```
QOCOCscMatrix* create_transposed_matrix (
            const QOCOCscMatrix * A )
```

Allocates and computes A^T.

Parameters

A	Input matrix.

5.7.4.8 csc_symperm()

```
QOCOCscMatrix* csc_symperm (
            const QOCOCscMatrix * A,
```

```
            const QOCOInt * pinv,
            QOCOInt * AtoC )
```

C = A(p,p) = PAP' where A and C are symmetric and the upper triangular part is stored.

Parameters

A	
pinv	
AtoC	

Returns

QOCOCscMatrix∗

5.7.4.9 cumsum()

```
QOCOInt cumsum (
            QOCOInt * p,
            QOCOInt * c,
            QOCOInt n )
```

Computes cumulative sum of c.

Returns

Cumulative sum of c.

5.7.4.10 dot()

```
QOCOFloat dot (
            const QOCOFloat * u,
            const QOCOFloat * v,
            QOCOInt n )
```

Computes dot product of u and v.

Parameters

u	Input vector.
v	Input vector.
n	Length of vectors.

Returns

Dot product of u and v.

5.7.4.11 ew_product()

```
void ew_product (
            QOCOFloat * x,
            const QOCOFloat * y,
            QOCOFloat * z,
            QOCOInt n )
```

Computes elementwise product z = x .* y.

Parameters

x	Input array.
y	Input array.
z	Output array.
n	Length of arrays.

5.7.4.12 free_qoco_csc_matrix()

```
void free_qoco_csc_matrix (
            QOCOCscMatrix * A )
```

Frees all the internal arrays and the pointer to the QOCOCscMatrix. Should only be used if QOCOCscMatrix and all internal arrays were malloc'ed.

Parameters

A	Pointer to QOCOCscMatrix.

5.7.4.13 inf_norm()

```
QOCOFloat inf_norm (
            const QOCOFloat * x,
            QOCOInt n )
```

Computes the infinity norm of x.

Parameters

x	Input vector.
n	Length of input vector.

Returns

Infinity norm of x.

5.7.4.14 invert_permutation()

```
void invert_permutation (
             const QOCOInt * p,
             QOCOInt * pinv,
             QOCOInt n )
```

Inverts permutation vector p and stores inverse in pinv.

Parameters

p	Input permutation vector.
pinv	Inverse of permutation vector.
n	Length of vectors.

5.7.4.15 max_arrayi()

```
QOCOInt max_arrayi (
             const QOCOInt * x,
             QOCOInt n )
```

Computes maximum element of array of QOCOInts.

Parameters

x	Input array.
n	Length of array.

Returns

Maximum element of x.

5.7.4.16 new_qoco_csc_matrix()

```
QOCOCscMatrix* new_qoco_csc_matrix (
            const QOCOCscMatrix * A )
```

Allocates a new csc matrix and copies A to it.

Parameters

A	Matrix to copy.

Returns

Pointer to new constructed matrix.

5.7.4.17 regularize()

```
QOCOInt regularize (
            QOCOCscMatrix * M,
            QOCOFloat lambda,
            QOCOInt * nzadded_idx )
```

Adds lambda $*$ I to a CSC matrix. Called on P prior to construction of KKT system in qoco_setup(). This function calls realloc() when adding new nonzeros.

Parameters

M	Matrix to be regularized.
lambda	Regularization factor.
nzadded_idx	Indices of elements of M->x that are added.

Returns

Number of nonzeros added to M->x.

5.7.4.18 row_col_scale()

```
void row_col_scale (
            const QOCOCscMatrix * M,
            QOCOFloat * E,
            QOCOFloat * D )
```

Scales the rows of M by E and columns of M by D. M = diag(E) $*$ M $*$ diag(S)

Parameters

M	An m by n sparse matrix.
E	Vector of length m.
D	Vector of length m.

5.7.4.19 row_inf_norm()

```
void row_inf_norm (
            const QOCOCscMatrix * M,
            QOCOFloat * norm )
```

Computes the infinity norm of each row of M and stores in norm.

Parameters

M	An m by n sparse matrix.
norm	Result vector of length m.

5.7.4.20 scale_arrayf()

```
void scale_arrayf (
            const QOCOFloat * x,
            QOCOFloat * y,
            QOCOFloat s,
            QOCOInt n )
```

Scales array x by s and stores result in y. y = s ∗ x.

Parameters

x	Input array.
y	Output array.
s	Scaling factor.
n	Length of arrays.

5.7.4.21 SpMtv()

```
void SpMtv (
            const QOCOCscMatrix * M,
```

```
const QOCOFloat * v,
QOCOFloat * r )
```

Sparse matrix vector multiplication for CSC matrices where M is first transposed. Computes r = M^T * v.

Parameters

M	Matrix in CSC form.
v	Vector.
r	Result.

5.7.4.22 SpMv()

```
void SpMv (
             const QOCOCscMatrix * M,
             const QOCOFloat * v,
             QOCOFloat * r )
```

Sparse matrix vector multiplication for CSC matrices. Computes r = M ∗ v.

Parameters

M	Matrix in CSC form.
v	Vector.
r	Result.

5.7.4.23 unregularize()

```
void unregularize (
             QOCOCscMatrix * M,
             QOCOFloat lambda )
```

Subtracts lambda ∗ I to a CSC matrix. Called on P when updating matrix data in update_matrix_data(). This function does not allocate and must be called after regularize.

Parameters

M	Matrix.
lambda	Regularization.

5.7.4.24 USpMv()

```
void USpMv (
             const QOCOCscMatrix * M,
             const QOCOFloat * v,
             QOCOFloat * r )
```

Sparse matrix vector multiplication for CSC matrices where M is symmetric and only the upper triangular part is given. Computes r = M ∗ v.

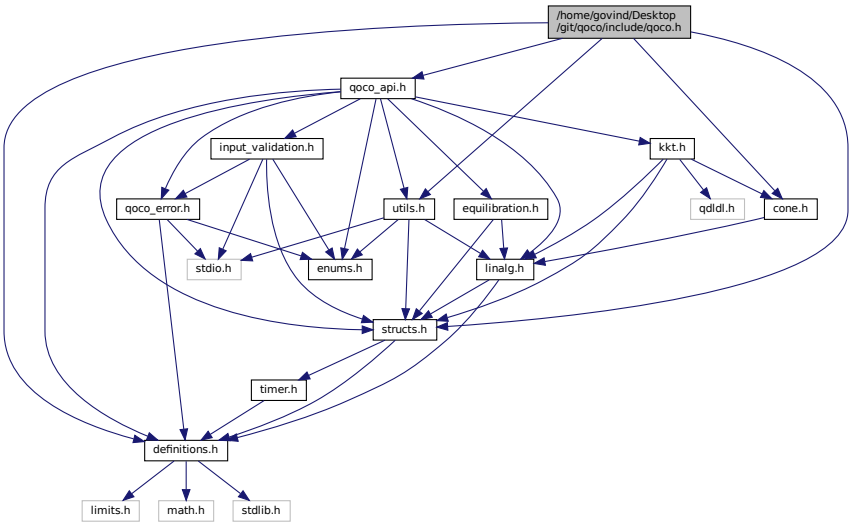
Parameters

M	Upper triangular part of M in CSC form.
v	Vector.
r	Result.

5.8 /home/govind/Desktop/git/qoco/include/qoco.h File Reference

```
#include "cone.h"
#include "definitions.h"
#include "qoco_api.h"
#include "structs.h"
#include "utils.h"
```

Include dependency graph for qoco.h:



5.8.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.8.2 LICENSE

Copyright (c) 2024, Govind M. Chari This source code is licensed under the BSD 3-Clause License

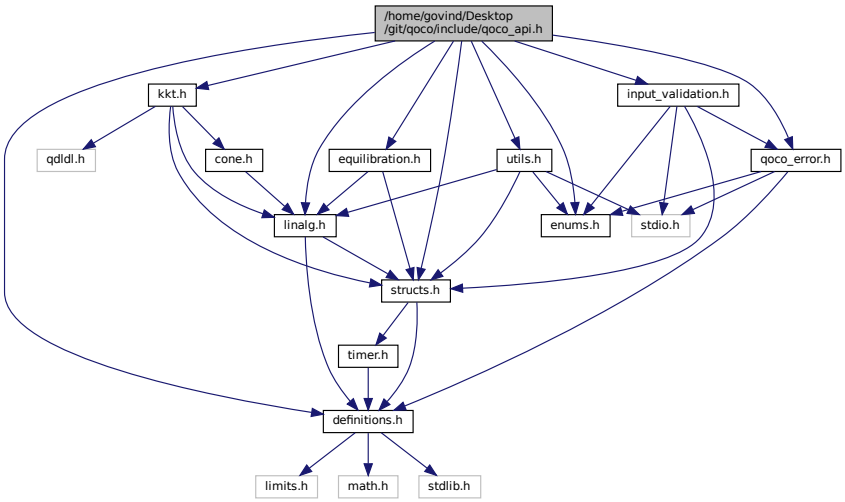
5.8.3 DESCRIPTION

This is the file that should be included when using QOCO.

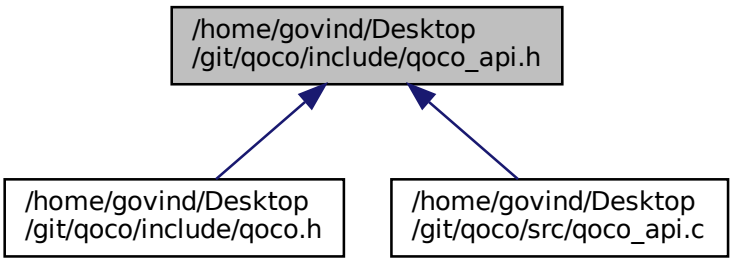
5.9 /home/govind/Desktop/git/qoco/include/qoco_api.h File Reference

```
#include "definitions.h"
#include "enums.h"
#include "equilibration.h"
#include "input_validation.h"
#include "kkt.h"
#include "linalg.h"
#include "qoco_error.h"
#include "structs.h"
#include "utils.h"
```

Include dependency graph for qoco_api.h:



This graph shows which files directly or indirectly include this file:



Functions

- QOCOInt qoco_setup (QOCOSolver *solver, QOCOInt n, QOCOInt m, QOCOInt p, QOCOCscMatrix *P, QOCOFloat *c, QOCOCscMatrix *A, QOCOFloat *b, QOCOCscMatrix *G, QOCOFloat *h, QOCOInt l, QOCOInt nsoc, QOCOInt *q, QOCOSettings *settings)

Allocates all memory needed for QOCO to solve the SOCP.

- void qoco_set_csc (QOCOCscMatrix *A, QOCOInt m, QOCOInt n, QOCOInt Annz, QOCOFloat *Ax, QOCOInt *Ap, QOCOInt *Ai)

  Sets the data for a compressed sparse column matrix.
- void set_default_settings (QOCOSettings *settings)

  Set the default settings struct.
- QOCOInt qoco_update_settings (QOCOSolver *solver, const QOCOSettings *new_settings)

  Updates settings struct.
- void update_vector_data (QOCOSolver *solver, QOCOFloat *cnew, QOCOFloat *bnew, QOCOFloat *hnew)

  Updates data vectors. NULL can be passed in for any vector if that data will not be updated.
- void update_matrix_data (QOCOSolver *solver, QOCOFloat *Pxnew, QOCOFloat *Axnew, QOCOFloat *Gxnew)

  Updates data matrices. NULL can be passed in for any matrix data pointers if that matrix will not be updated. It is assumed that the new matrix will have the same sparsity structure as the existing matrix.
- QOCOInt qoco_solve (QOCOSolver *solver)

  Solves SOCP.
- QOCOInt qoco_cleanup (QOCOSolver *solver)

  Frees all memory allocated by qoco_setup.

5.9.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.9.2 LICENSE

Copyright (c) 2024, Govind M. Chari This source code is licensed under the BSD 3-Clause License

5.9.3 DESCRIPTION

Exposes the API for QOCO.

5.9.4 Function Documentation

5.9.4.1 qoco_cleanup()

```
QOCOInt qoco_cleanup (
            QOCOSolver * solver )
```

Frees all memory allocated by qoco_setup.

Parameters

solver	Pointer to solver.

Returns

Exitflag to check (0 for success, failure otherwise)

5.9.4.2 qoco_set_csc()

```
void qoco_set_csc (
             QOCOCscMatrix * A,
             QOCOInt m,
             QOCOInt n,
             QOCOInt Annz,
             QOCOFloat * Ax,
             QOCOInt * Ap,
             QOCOInt * Ai )
```

Sets the data for a compressed sparse column matrix.

Parameters

A	Pointer to the CSC matrix.
m	Number of rows in the matrix.
n	Number of columns in the matrix.
Annz	Number of nonzero elements in the matrix.
Ax	Array of data for the matrix.
Ap	Array of column pointers for the data.
Ai	Array of row indices for data.

5.9.4.3 qoco_setup()

```
QOCOInt qoco_setup (
             QOCOSolver * solver,
             QOCOInt n,
             QOCOInt m,
             QOCOInt p,
             QOCOCscMatrix * P,
             QOCOFloat * c,
             QOCOCscMatrix * A,
             QOCOFloat * b,
             QOCOCscMatrix * G,
             QOCOFloat * h,
             QOCOInt l,
             QOCOInt nsoc,
             QOCOInt * q,
             QOCOSettings * settings )
```

Allocates all memory needed for QOCO to solve the SOCP.

Parameters

solver	Pointer to solver.
n	Number of optimization variables.
m	Number of conic constraints.
p	Number of affine equality constraints.
P	Upper triangular part of quadratic cost Hessian in CSC form.
c	Linear cost vector.
A	Affine equality constraint matrix in CSC form.
b	Affine equality constraint offset vector.
G	Conic constraint matrix in CSC form.
h	Conic constraint offset vector.
l	Dimension of non-negative orthant.
nsoc	Number of second-order cones.
q	Dimension of each second-order cone.
settings	Settings struct.

Returns

0 if no error or flag containing error code.

5.9.4.4 qoco_solve()

```
QOCOInt qoco_solve (
            QOCOSolver * solver )
```

Solves SOCP.

Parameters

solver	Pointer to solver.

Returns

Exitflag to check (0 for success, failure otherwise)

5.9.4.5 qoco_update_settings()

```
QOCOInt qoco_update_settings (
            QOCOSolver * solver,
            const QOCOSettings * new_settings )
```

Updates settings struct.

Parameters

solver	Pointer to solver.
new_settings	New settings struct.

Returns

0 if update is successful.

5.9.4.6 set_default_settings()

```
void set_default_settings (
            QOCOSettings * settings )
```

Set the default settings struct.

Parameters

settings	Pointer to settings struct.

5.9.4.7 update_matrix_data()

```
void update_matrix_data (
            QOCOSolver * solver,
            QOCOFloat * Pxnew,
            QOCOFloat * Axnew,
            QOCOFloat * Gxnew )
```

Updates data matrices. NULL can be passed in for any matrix data pointers if that matrix will not be updated. It is assumed that the new matrix will have the same sparsity structure as the existing matrix.

Parameters

solver	Pointer to solver.
Pxnew	New data for P->x.
Axnew	New data for A->x.
Gxnew	New data for G->x.

5.9.4.8 update_vector_data()

```
void update_vector_data (
            QOCOSolver * solver,
```

```
            QOCOFloat * cnew,
            QOCOFloat * bnew,
            QOCOFloat * hnew )
```

Updates data vectors. NULL can be passed in for any vector if that data will not be updated.

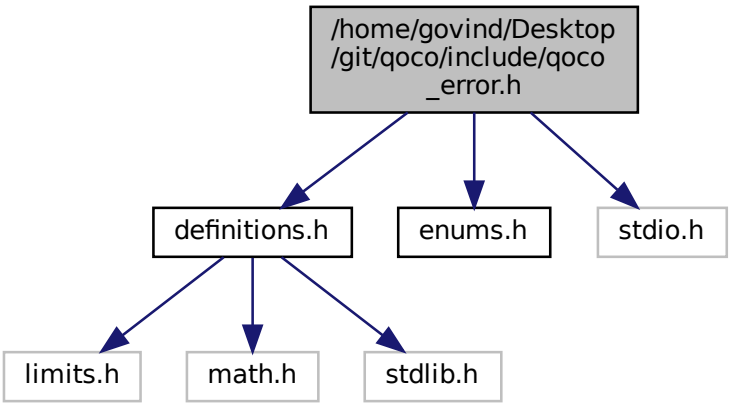
Parameters

solver	Pointer to solver.
cnew	New c vector.
bnew	New b vector.
hnew	New h vector.

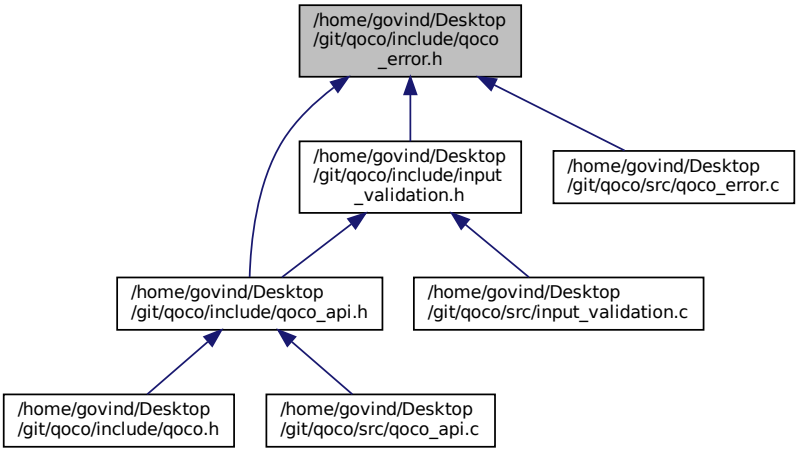
5.10 /home/govind/Desktop/git/qoco/include/qoco_error.h File Reference

```
#include "definitions.h"
#include "enums.h"
#include <stdio.h>
```

Include dependency graph for qoco_error.h:



This graph shows which files directly or indirectly include this file:



Functions

- QOCOInt qoco_error (enum qoco_error_code error_code)

  Function to print error messages.

5.10.1 Function Documentation

5.10.1.1 qoco_error()

```
QOCOInt qoco_error (
            enum qoco_error_code error_code )
```

Function to print error messages.

Parameters

error_code	

Returns

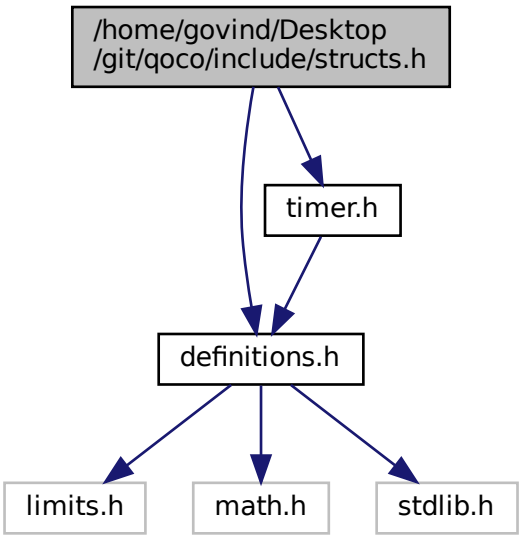
Error code as an QOCOInt.

5.11 /home/govind/Desktop/git/qoco/include/structs.h File Reference

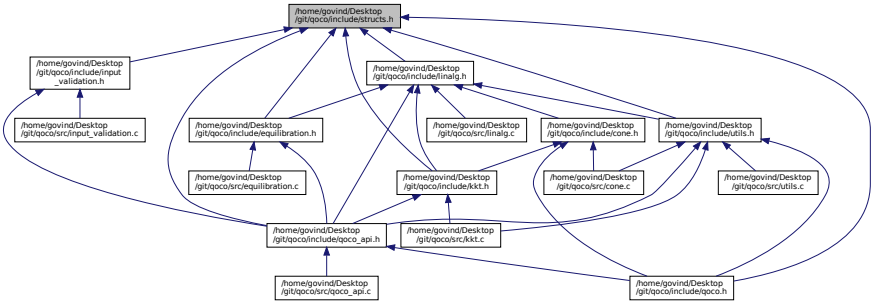
```
#include "definitions.h"
```

```
#include "timer.h"
```

Include dependency graph for structs.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct QOCOCscMatrix

  Compressed sparse column format matrices.
- struct QOCOProblemData

  SOCP problem data.
- struct QOCOSettings

  QOCO solver settings.
- struct QOCOKKT

  Contains all data needed for constructing and modifying KKT matrix and performing predictor-corrector step.
- struct QOCOWorkspace

QOCO Workspace.

- struct QOCOSolution
- struct QOCOSolver

  QOCO Solver struct. Contains all information about the state of the solver.

5.11.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.11.2 LICENSE

Copyright (c) 2024, Govind M. Chari This source code is licensed under the BSD 3-Clause License

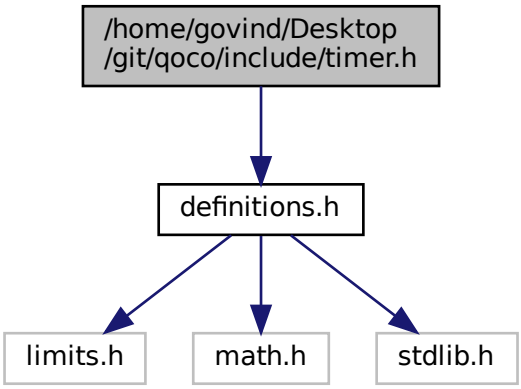
5.11.3 DESCRIPTION

Defines all structs used by QOCO.

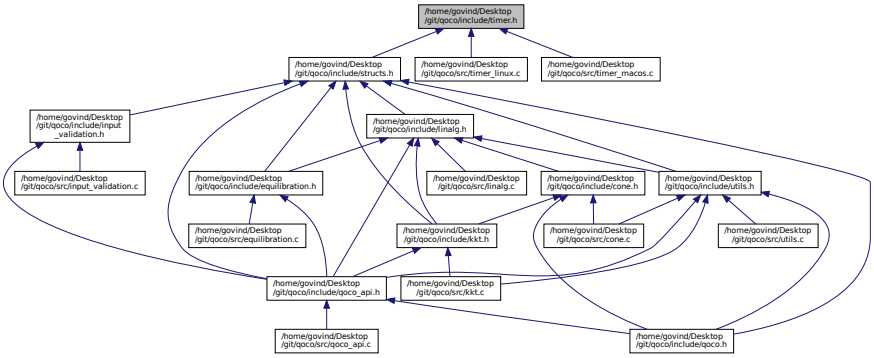
5.12 /home/govind/Desktop/git/qoco/include/timer.h File Reference

```
#include "definitions.h"
```

Include dependency graph for timer.h:



This graph shows which files directly or indirectly include this file:



Functions

- void start_timer (QOCOTimer ∗timer)

  Starts timer and sets tic field of struct to the current time.
- void stop_timer (QOCOTimer ∗timer)

  Stops timer and sets toc field of struct to the current time.
- QOCOFloat get_elapsed_time_sec (QOCOTimer ∗timer)

  Gets time in seconds recorded by timer. Must be called after start_timer() and stop_timer().

5.12.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.12.2 LICENSE

Copyright (c) 2024, Govind M. Chari This source code is licensed under the BSD 3-Clause License

5.12.3 DESCRIPTION

Provides timing functions.

5.12.4 Function Documentation

5.12.4.1 get_elapsed_time_sec()

```
QOCOFloat get_elapsed_time_sec (
            QOCOTimer * timer )
```

Gets time in seconds recorded by timer. Must be called after start_timer() and stop_timer().

Parameters

timer	Pointer to timer struct.

5.12.4.2 start_timer()

```
void start_timer (
            QOCOTimer * timer )
```

Starts timer and sets tic field of struct to the current time.

Parameters

timer	Pointer to timer struct.

5.12.4.3 stop_timer()

```
void stop_timer (
            QOCOTimer * timer )
```

Stops timer and sets toc field of struct to the current time.

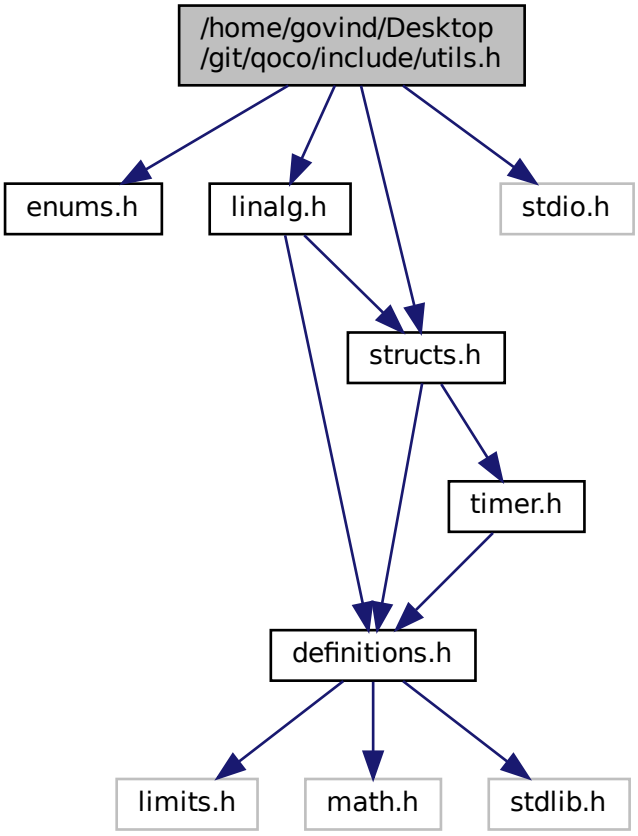
Parameters

timer	Pointer to timer struct.

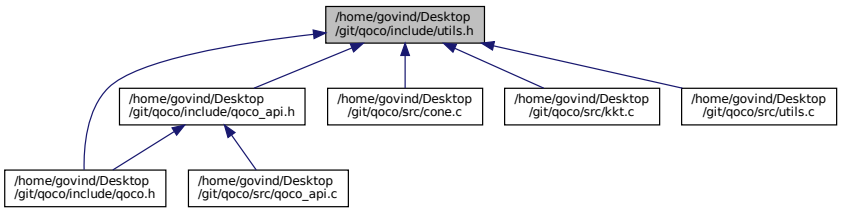
5.13 /home/govind/Desktop/git/qoco/include/utils.h File Reference

```
#include "enums.h"
#include "linalg.h"
#include "structs.h"
#include <stdio.h>
```

Include dependency graph for utils.h:



This graph shows which files directly or indirectly include this file:



Functions

- void print_qoco_csc_matrix (QOCOCscMatrix *M)

  Prints dimensions, number of nonzero elements, data, column pointers and row indices for a sparse matrix in CSC form.
- void print_arrayf (QOCOFloat *x, QOCOInt n)

Prints array of QOCOFloats.

- void print_arrayi (QOCOInt *x, QOCOInt n)

  Prints array of QOCOInts.
- void print_header (QOCOSolver *solver)

  Prints QOCO header.
- void log_iter (QOCOSolver *solver)

  Print solver progress.
- void print_footer (QOCOSolution *solution, enum qoco_solve_status status)

  Prints QOCO footer.
- unsigned char check_stopping (QOCOSolver *solver)

  Checks stopping criteria. Before calling this function, work->kkt->rhs must contain the residual of the KKT conditions as computed by compute_kkt_residual().
- void copy_solution (QOCOSolver *solver)

  Copies data to QOCOSolution struct when solver terminates.
- QOCOSettings * copy_settings (QOCOSettings *settings)

  Allocates and returns a copy of the input settings struct.

5.13.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.13.2 LICENSE

5.13.3 DESCRIPTION

Provides various utility functions.

5.13.4 Function Documentation

5.13.4.1 check_stopping()

```
unsigned char check_stopping (
            QOCOSolver * solver )
```

Checks stopping criteria. Before calling this function, work->kkt->rhs must contain the residual of the KKT conditions as computed by compute_kkt_residual().

Parameters

solver	Pointer to solver.

Returns

1 if stopping criteria met and 0 otherwise.

5.13.4.2 copy_settings()

```
QOCOSettings* copy_settings (
             QOCOSettings * settings )
```

Allocates and returns a copy of the input settings struct.

Parameters

settings	Input struct.

Returns

Pointer to constructed and copies settings struct.

5.13.4.3 copy_solution()

```
void copy_solution (
             QOCOSolver * solver )
```

Copies data to QOCOSolution struct when solver terminates.

Parameters

solver	Pointer to solver.

5.13.4.4 log_iter()

```
void log_iter (
             QOCOSolver * solver )
```

Print solver progress.

Parameters

solver	Pointer to solver.

5.13.4.5 print_arrayf()

```
void print_arrayf (
            QOCOFloat * x,
            QOCOInt n )
```

Prints array of QOCOFloats.

Parameters

x	Pointer to array.
n	Number of elements in array.

5.13.4.6 print_arrayi()

```
void print_arrayi (
            QOCOInt * x,
            QOCOInt n )
```

Prints array of QOCOInts.

Parameters

x	Pointer to array.
n	Number of elements in array.

5.13.4.7 print_footer()

```
void print_footer (
            QOCOSolution * solution,
            enum qoco_solve_status status )
```

Prints QOCO footer.

Parameters

solution	Pointer to solution struct.
status	Solve status.

5.13.4.8 print_header()

```
void print_header (
            QOCOSolver * solver )
```

Prints QOCO header.

Parameters

solver	Pointer to solver.

5.13.4.9 print_qoco_csc_matrix()

```
void print_qoco_csc_matrix (
             QOCOCscMatrix * M )
```

Prints dimensions, number of nonzero elements, data, column pointers and row indices for a sparse matrix in CSC form.

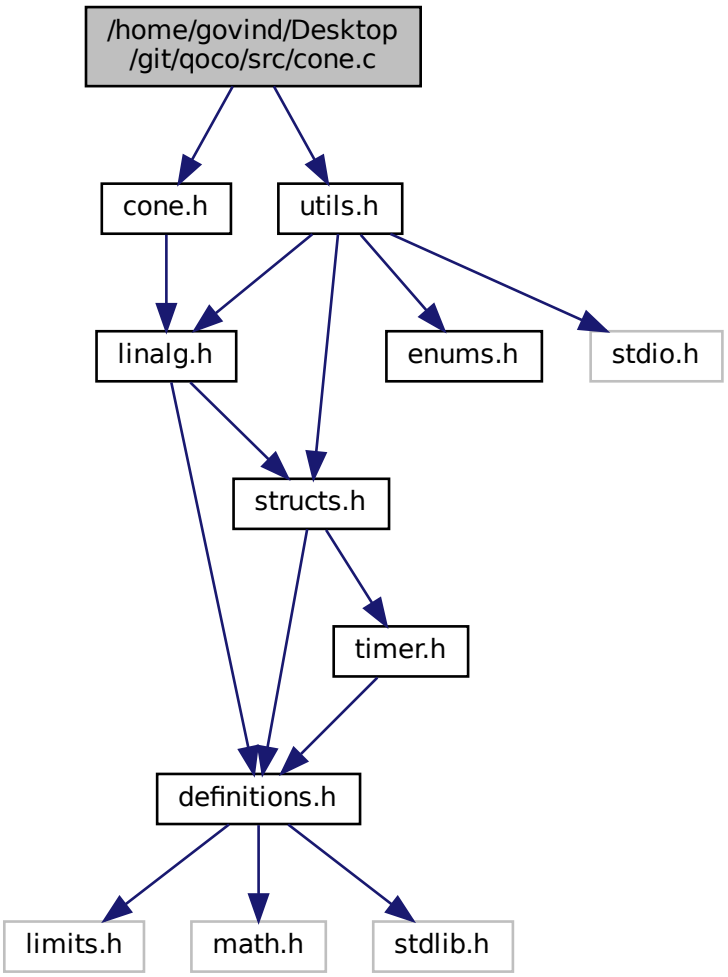
Parameters

M	Pointer to QOCOCscMatrix that will be printed.

5.14 /home/govind/Desktop/git/qoco/src/cone.c File Reference

```
#include "cone.h"
#include "utils.h"
```

Include dependency graph for cone.c:



Functions

- void soc_product (const QOCOFloat *u, const QOCOFloat *v, QOCOFloat *p, QOCOInt n)

  *Computes second-order cone product u * v = p.*
- void soc_division (const QOCOFloat *lam, const QOCOFloat *v, QOCOFloat *d, QOCOInt n)

  Commpues second-order cone division lambda # v = d.
- QOCOFloat soc_residual (const QOCOFloat *u, QOCOInt n)

  Computes residual of vector u with respect to the second order cone of dimension n.
- QOCOFloat soc_residual2 (const QOCOFloat *u, QOCOInt n)

  *Computes u0^2 - u1'*u1 of vector u with respect to the second order cone of dimension n.*
- void cone_product (const QOCOFloat *u, const QOCOFloat *v, QOCOFloat *p, QOCOInt l, QOCOInt nsoc, const QOCOInt *q)

  *Computes cone product u * v = p with respect to C.*
- void cone_division (const QOCOFloat *lambda, const QOCOFloat *v, QOCOFloat *d, QOCOInt l, QOCOInt nsoc, const QOCOInt *q)

Computed cone division lambda # v = d.

- QOCOFloat cone_residual (const QOCOFloat *u, QOCOInt l, QOCOInt nsoc, const QOCOInt *q)

  Computes residual of vector u with respect to cone C.
- void bring2cone (QOCOFloat *u, QOCOProblemData *data)

  *Performs u = u + (1 + a) * e where e is the cannonical vector for each cone LP Cone: e = ones(n), second-order cone: e = (1,0,0,...) and a is the minimum scalar value such that u + (1 + a) * e is in cone C.*
- void nt_multiply (QOCOFloat *W, QOCOFloat *x, QOCOFloat *z, QOCOInt l, QOCOInt m, QOCOInt nsoc, QOCOInt *q)

  *Computes z = W * x where W is a full Nesterov-Todd scaling matrix. The NT scaling array for the LP cones are stored first, then the NT scalings for the second-order cones are stored in column major order.*
- void compute_mu (QOCOWorkspace *work)

  *Computes gap (z'*s / m) and stores in work->mu.*
- void compute_nt_scaling (QOCOWorkspace *work)

  Compute Nesterov-Todd scalings and scaled variables.
- void compute_centering (QOCOSolver *solver)

  Computes centering parameter.
- QOCOFloat linesearch (QOCOFloat *u, QOCOFloat *Du, QOCOFloat f, QOCOSolver *solver)

  *Conducts linesearch to compute a \in (0, 1] such that u + (a / f) * Du \in C. For QPs this calls exact_linesearch() and for SOCPs this calls bisection_search()*
- QOCOFloat bisection_search (QOCOFloat *u, QOCOFloat *Du, QOCOFloat f, QOCOSolver *solver)

  *Conducts linesearch by bisection to compute a \in (0, 1] such that u + (a / f) * Du \in C Warning: linesearch overwrites ubuff1. Do not pass in ubuff1 into u or Du. Consider a dedicated buffer for linesearch.*
- QOCOFloat exact_linesearch (QOCOFloat *u, QOCOFloat *Du, QOCOFloat f, QOCOSolver *solver)

  *Conducts exact linesearch to compute the largest a \in (0, 1] such that u + (a / f) * Du \in C. Currently only works for LP cone.*

5.14.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.14.2 LICENSE

5.14.3 Function Documentation

5.14.3.1 bisection_search()

```
QOCOFloat bisection_search (
            QOCOFloat * u,
            QOCOFloat * Du,
            QOCOFloat f,
            QOCOSolver * solver )
```

Conducts linesearch by bisection to compute a \in (0, 1] such that u + (a / f) * Du \in C Warning: linesearch overwrites ubuff1. Do not pass in ubuff1 into u or Du. Consider a dedicated buffer for linesearch.

Parameters

u	Initial vector.
Du	Search direction.
f	Conservatism factor.
solver	Pointer to solver.

Returns

Step-size.

5.14.3.2 bring2cone()

```
void bring2cone (
            QOCOFloat * u,
            QOCOProblemData * data )
```

Performs u = u + (1 + a) $*$ e where e is the cannonical vector for each cone LP Cone: e = ones(n), second-order cone: e = (1,0,0,...) and a is the minimum scalar value such that u + (1 + a) $*$ e is in cone C.

Parameters

u	Vector to bring to cone.
data	Pointer to problem data.

5.14.3.3 compute_centering()

```
void compute_centering (
            QOCOSolver * solver )
```

Computes centering parameter.

Parameters

solver	Pointer to solver.

5.14.3.4 compute_mu()

```
void compute_mu (
            QOCOWorkspace * work )
```

Computes gap (z'$*$s / m) and stores in work->mu.

Parameters

work	Pointer to workspace.

5.14.3.5 compute_nt_scaling()

```
void compute_nt_scaling (
            QOCOWorkspace * work )
```

Compute Nesterov-Todd scalings and scaled variables.

Parameters

work	Pointer to workspace.

5.14.3.6 cone_division()

```
void cone_division (
            const QOCOFloat * lambda,
            const QOCOFloat * v,
            QOCOFloat * d,
            QOCOInt l,
            QOCOInt nsoc,
            const QOCOInt * q )
```

Computed cone division lambda # v = d.

Parameters

lambda	Input vector.
v	Input vector.
d	Cone quotient of lambda and v.
l	Dimension of LP cone.
nsoc	Number of second-order cones.
q	Dimension of each second-order cone.

5.14.3.7 cone_product()

```
void cone_product (
            const QOCOFloat * u,
            const QOCOFloat * v,
```

```
                QOCOFloat * p,
                QOCOInt l,
                QOCOInt nsoc,
                const QOCOInt * q )
```

Computes cone product u ∗ v = p with respect to C.

Parameters

u	Input vector.
v	Input vector.
p	Cone product of u and v.
l	Dimension of LP cone.
nsoc	Number of second-order cones.
q	Dimension of each second-order cone.

5.14.3.8 cone_residual()

```
QOCOFloat cone_residual (
                const QOCOFloat * u,
                QOCOInt l,
                QOCOInt nsoc,
                const QOCOInt * q )
```

Computes residual of vector u with respect to cone C.

Parameters

u	Vector to be tested.
l	Dimension of LP cone.
nsoc	Number of second-order cones.
q	Dimension of each second-order cone.

Returns

Residual: Negative if the vector is in the cone and positive otherwise.

5.14.3.9 exact_linesearch()

```
QOCOFloat exact_linesearch (
                QOCOFloat * u,
                QOCOFloat * Du,
                QOCOFloat f,
                QOCOSolver * solver )
```

Conducts exact linesearch to compute the largest a \in (0, 1] such that u + (a / f) ∗ Du \in C. Currently only works for LP cone.

Todo get exact_linesearch working for SOCs.

Parameters

u	Initial vector.
Du	Search direction.
f	Conservatism factor.
solver	Pointer to solver.

Returns

Step-size.

5.14.3.10 linesearch()

```
QOCOFloat linesearch (
             QOCOFloat * u,
             QOCOFloat * Du,
             QOCOFloat f,
             QOCOSolver * solver )
```

Conducts linesearch to compute a \in (0, 1] such that u + (a / f) ∗ Du \in C. For QPs this calls exact_linesearch() and for SOCPs this calls bisection_search()

Parameters

u	Initial vector.
Du	Search direction.
f	Conservatism factor.
solver	Pointer to solver.

Returns

Step-size.

5.14.3.11 nt_multiply()

```
void nt_multiply (
             QOCOFloat * W,
             QOCOFloat * x,
             QOCOFloat * z,
             QOCOInt l,
             QOCOInt m,
             QOCOInt nsoc,
             QOCOInt * q )
```

Computes z = W ∗ x where W is a full Nesterov-Todd scaling matrix. The NT scaling array for the LP cones are stored first, then the NT scalings for the second-order cones are stored in column major order.

Parameters

W	Nesterov Todd scaling matrix.
x	Input vector.
z	Output vector.
l	Dimension of LP cone.
m	Length of x.
nsoc	Number of second-order cones in C.
q	Array of second-order cone dimensions.

5.14.3.12 soc_division()

```
void soc_division (
            const QOCOFloat * lam,
            const QOCOFloat * v,
            QOCOFloat * d,
            QOCOInt n )
```

Commpues second-order cone division lambda # v = d.

Parameters

lam	lam = (lam0, lam1) is a vector in second-order cone of dimension n.
v	v = (v0, v1) is a vector in second-order cone of dimension n.
d	Cone divisin of lam and v.
n	Dimension of second-order cone.

5.14.3.13 soc_product()

```
void soc_product (
            const QOCOFloat * u,
            const QOCOFloat * v,
            QOCOFloat * p,
            QOCOInt n )
```

Computes second-order cone product u $*$ v = p.

Parameters

u	u = (u0, u1) is a vector in second-order cone of dimension n.
v	v = (v0, v1) is a vector in second-order cone of dimension n.
p	Cone product of u and v.
n	Dimension of second-order cone.

5.14.3.14 soc_residual()

```
QOCOFloat soc_residual (
             const QOCOFloat * u,
             QOCOInt n )
```

Computes residual of vector u with respect to the second order cone of dimension n.

Parameters

u	u = (u0, u1) is a vector in second-order cone of dimension n.
n	Dimension of second order cone.

Returns

Residual: norm(u1) - u0. Negative if the vector is in the cone and positive otherwise.

5.14.3.15 soc_residual2()

```
QOCOFloat soc_residual2 (
             const QOCOFloat * u,
             QOCOInt n )
```

Computes u0^2 - u1'∗u1 of vector u with respect to the second order cone of dimension n.

Parameters

u	u = (u0, u1) is a vector in second order cone of dimension n.
n	Dimension of second order cone.

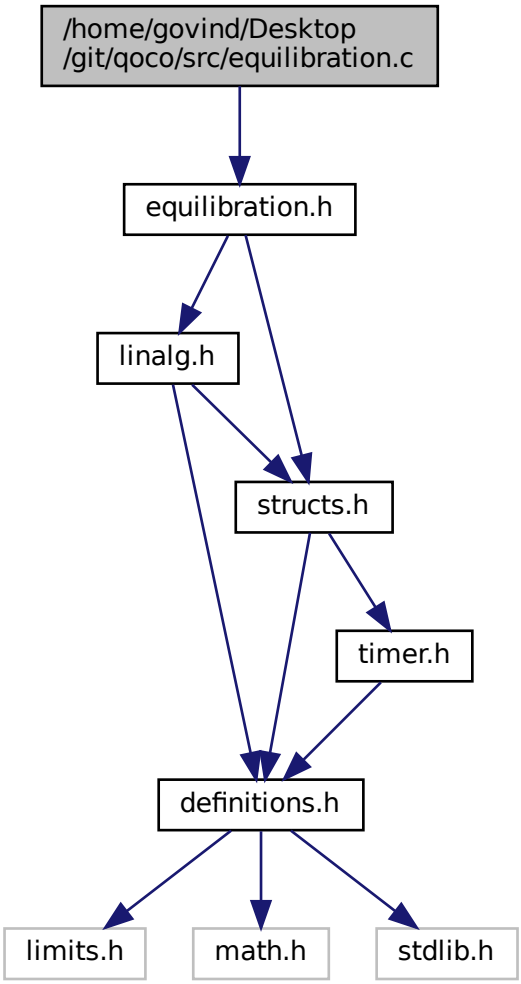
Returns

Residual: u0^2 - u1'∗u1.

5.15 /home/govind/Desktop/git/qoco/src/equilibration.c File Reference

```
#include "equilibration.h"
```

Include dependency graph for equilibration.c:



Functions

- void ruiz_equilibration (QOCOSolver *solver)

  Applies modified ruiz equilibration to scale data matrices. Computes D, E, F, and k as shown below to make the row and column infinity norms equal for the scaled KKT matrix.
- void unscale_variables (QOCOWorkspace *work)

  Undo variable transformation induced by ruiz equilibration.

5.15.1 Function Documentation

5.15.1.1 ruiz_equilibration()

```
void ruiz_equilibration (
            QOCOSolver * solver )
```

Applies modified ruiz equilibration to scale data matrices. Computes D, E, F, and k as shown below to make the row and column infinity norms equal for the scaled KKT matrix.

- clang-format off

[D] [kP A^T G^T] [D] | E | | A 0 0 | | E | [F] [G 0 0] [F]

clang-format on

Parameters

solver	Pointer to solver.

5.15.1.2 unscale_variables()

```
void unscale_variables (
            QOCOWorkspace * work )
```

Undo variable transformation induced by ruiz equilibration.

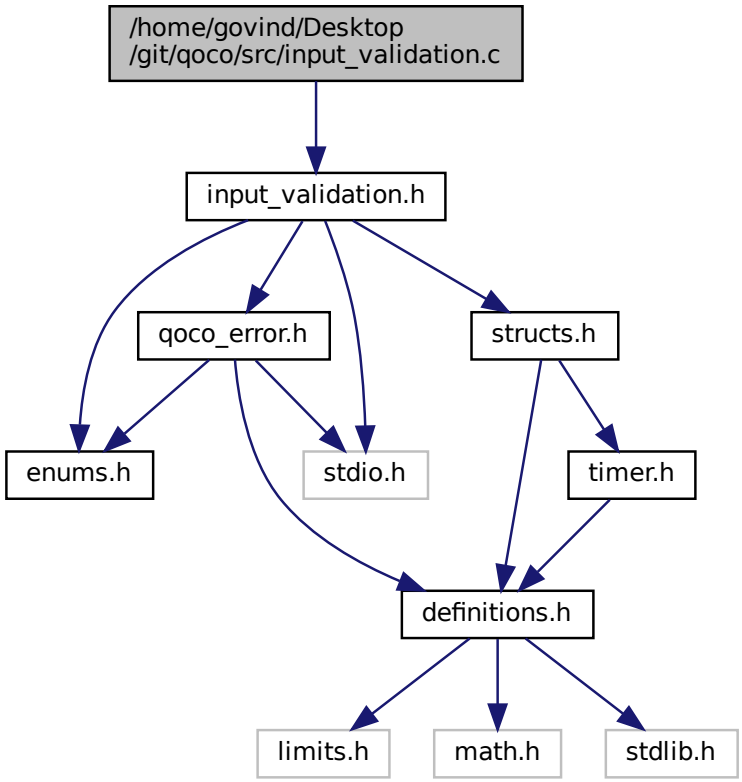
Parameters

work	Pointer to workspace.

5.16 /home/govind/Desktop/git/qoco/src/input_validation.c File Reference

```
#include "input_validation.h"
```

Include dependency graph for input_validation.c:



Functions

- QOCOInt qoco_validate_settings (const QOCOSettings *settings)

  Validates solver settings.
- QOCOInt qoco_validate_data (const QOCOCscMatrix *P, const QOCOFloat *c, const QOCOCscMatrix *A, const QOCOFloat *b, const QOCOCscMatrix *G, const QOCOFloat *h, const QOCOInt l, const QOCOInt nsoc, const QOCOInt *q)

  Validate problem data.

5.16.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.16.2 LICENSE

5.16.3 Function Documentation

5.16.3.1 qoco_validate_data()

```
QOCOInt qoco_validate_data (
            const QOCOCscMatrix * P,
            const QOCOFloat * c,
            const QOCOCscMatrix * A,
            const QOCOFloat * b,
            const QOCOCscMatrix * G,
            const QOCOFloat * h,
            const QOCOInt l,
            const QOCOInt nsoc,
            const QOCOInt * q )
```

Validate problem data.

Parameters

P	Upper triangular part of quadratic cost Hessian in CSC form
c	Linear cost vector
A	Affine equality constraint matrix in CSC form
b	Affine equality constraint offset vector
G	Conic constraint matrix in CSC form
h	Conic constraint offset vector
l	Dimension of non-negative orthant
nsoc	Number of second-order cones
q	Dimension of each second-order cone

Returns

Exitflag to check (0 for success, failure otherwise)

5.16.3.2 qoco_validate_settings()

```
QOCOInt qoco_validate_settings (
            const QOCOSettings * settings )
```

Validates solver settings.

Parameters

settings	Pointer to settings struct

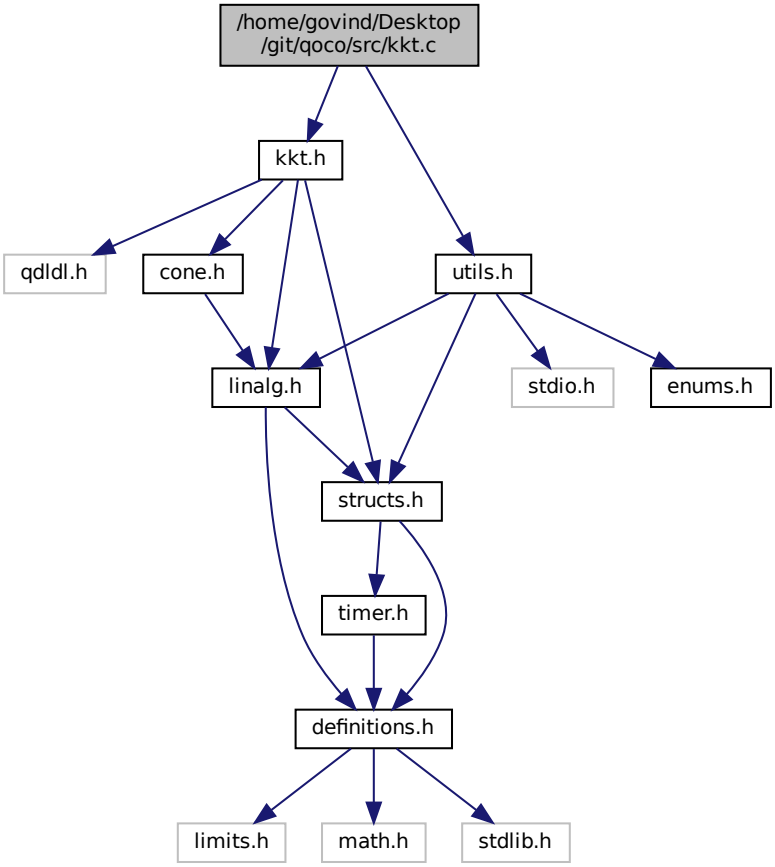
Returns

Exitflag to check (0 for success, failure otherwise)

5.17 /home/govind/Desktop/git/qoco/src/kkt.c File Reference

```
#include "kkt.h"
#include "utils.h"
```

Include dependency graph for kkt.c:



Functions

- void allocate_kkt (QOCOWorkspace *work)

  Allocate memory for KKT matrix.
- void construct_kkt (QOCOSolver *solver)

  Constructs upper triangular part of KKT matrix with -I for Nestrov-Todd scaling matrix (the (3,3) block)
- void initialize_ipm (QOCOSolver *solver)

  Gets initial values for primal and dual variables such that (s,z) \in C.
- void set_nt_block_zeros (QOCOWorkspace *work)

  Set the Nesterov-Todd block to be zeros. Used prior to compute_kkt_residual().

- void update_nt_block (QOCOSolver *solver)

  Updates and regularizes Nesterov-Todd scaling block of KKT matrix.
- void compute_kkt_residual (QOCOSolver *solver)

  Computes residual of KKT conditions and stores in work->kkt->rhs.
- void construct_kkt_aff_rhs (QOCOWorkspace *work)

  Constructs rhs for the affine scaling KKT system. Before calling this function, work->kkt->kktres must contain the residual of the KKT conditions as computed by compute_kkt_residual().
- void construct_kkt_comb_rhs (QOCOWorkspace *work)

  Constructs rhs for the combined direction KKT system. Before calling this function, work->kkt->kktres must contain the negative residual of the KKT conditions as computed by compute_kkt_residual().
- void predictor_corrector (QOCOSolver *solver)

  Performs Mehrotra predictor-corrector step.
- void kkt_solve (QOCOSolver *solver, QOCOFloat *b, QOCOInt iters)

  Solves Kx = b once K has been factored. Solves via triangular solves and applies iterative refinement afterwards.
- void kkt_multiply (QOCOSolver *solver, QOCOFloat *x, QOCOFloat *y)

  *Computes y = Kx where [P A^T G^T] K = | A 0 0 | [G 0 -W'W - e * I].*

5.17.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.17.2 LICENSE

5.17.3 Function Documentation

5.17.3.1 allocate_kkt()

```
void allocate_kkt (
            QOCOWorkspace * work )
```

Allocate memory for KKT matrix.

Parameters

work	Pointer to workspace.

5.17.3.2 compute_kkt_residual()

```
void compute_kkt_residual (
```

```
            QOCOSolver * solver )
```

Computes residual of KKT conditions and stores in work->kkt->rhs.

clang-format off

```
  [ P   A^T   G^T ] [ x ]   [    c   ]
```

res = | A 0 0 | | y] + | -b | [G 0 0] [z] [-h + s]

clang-format on

Parameters

solver	Pointer to solver.

5.17.3.3 construct_kkt()

```
void construct_kkt (
            QOCOSolver * solver )
```

Constructs upper triangular part of KKT matrix with -I for Nestrov-Todd scaling matrix (the (3,3) block)

clang-format off

```
[ P   A^T   G^T ]
```

K = | A 0 0 | [G 0 -I]

clang-format on

Parameters

solver	Pointer to solver

5.17.3.4 construct_kkt_aff_rhs()

```
void construct_kkt_aff_rhs (
            QOCOWorkspace * work )
```

Constructs rhs for the affine scaling KKT system. Before calling this function, work->kkt->kktres must contain the residual of the KKT conditions as computed by compute_kkt_residual().

Parameters

work	Pointer to workspace.

5.17.3.5 construct_kkt_comb_rhs()

```
void construct_kkt_comb_rhs (
            QOCOWorkspace * work )
```

Constructs rhs for the combined direction KKT system. Before calling this function, work->kkt->kktres must contain the negative residual of the KKT conditions as computed by compute_kkt_residual().

Parameters

work	Pointer to workspace.

ds = -cone_product(lambda, lambda) - settings.mehrotra * cone_product((W' \ Dsaff), (W * Dzaff), pdata) + sigma * mu * e.

5.17.3.6 initialize_ipm()

```
void initialize_ipm (
            QOCOSolver * solver )
```

Gets initial values for primal and dual variables such that (s,z) \in C.

Parameters

solver	Pointer to solver.

5.17.3.7 kkt_multiply()

```
void kkt_multiply (
            QOCOSolver * solver,
            QOCOFloat * x,
            QOCOFloat * y )
```

Computes y = Kx where [P A^T G^T] K = | A 0 0 | [G 0 -W'W - e * I].

Parameters

solver	Pointer to solver.
x	Pointer to input vector.
y	Pointer to output vector.

5.17.3.8 kkt_solve()

```
void kkt_solve (
            QOCOSolver * solver,
            QOCOFloat * b,
            QOCOInt iters )
```

Solves Kx = b once K has been factored. Solves via triangular solves and applies iterative refinement afterwards.

Parameters

solver	Pointer to solver.
b	Pointer to rhs of kkt system.
iters	Number of iterations of iterative refinement performed.

5.17.3.9 predictor_corrector()

```
void predictor_corrector (
            QOCOSolver * solver )
```

Performs Mehrotra predictor-corrector step.

Parameters

solver	Pointer to solver.

5.17.3.10 set_nt_block_zeros()

```
void set_nt_block_zeros (
            QOCOWorkspace * work )
```

Set the Nesterov-Todd block to be zeros. Used prior to compute_kkt_residual().

Parameters

work	Pointer to workspace.

5.17.3.11 update_nt_block()

```
void update_nt_block (
            QOCOSolver * solver )
```

Updates and regularizes Nesterov-Todd scaling block of KKT matrix.

```
[ P   A^T      G^T      ]
```

K = | A 0 0 | [G 0 -W'W - e ∗ I]

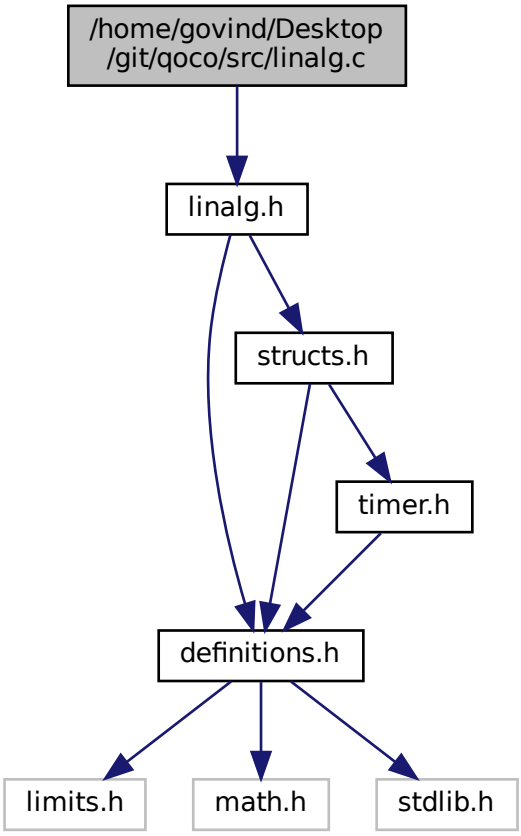
Parameters

solver	Pointer to solver.

5.18 /home/govind/Desktop/git/qoco/src/linalg.c File Reference

```
#include "linalg.h"
```

Include dependency graph for linalg.c:



Functions

- QOCOCscMatrix ∗ new_qoco_csc_matrix (const QOCOCscMatrix ∗A)

  Allocates a new csc matrix and copies A to it.

- QOCOCscMatrix * construct_identity (QOCOInt n, QOCOFloat lambda)

  *Allocates a new csc matrix that is lambda * I.*
- void free_qoco_csc_matrix (QOCOCscMatrix *A)

  Frees all the internal arrays and the pointer to the QOCOCscMatrix. Should only be used if QOCOCscMatrix and all internal arrays were malloc'ed.
- void copy_arrayf (const QOCOFloat *x, QOCOFloat *y, QOCOInt n)

  Copies array of QOCOFloats from x to array y.
- void copy_and_negate_arrayf (const QOCOFloat *x, QOCOFloat *y, QOCOInt n)

  Copies and negates array of QOCOFloats from x to array y.
- void copy_arrayi (const QOCOInt *x, QOCOInt *y, QOCOInt n)

  Copies array of QOCOInts from x to array y.
- QOCOFloat dot (const QOCOFloat *u, const QOCOFloat *v, QOCOInt n)

  Computes dot product of u and v.
- QOCOInt max_arrayi (const QOCOInt *x, QOCOInt n)

  Computes maximum element of array of QOCOInts.
- void scale_arrayf (const QOCOFloat *x, QOCOFloat *y, QOCOFloat s, QOCOInt n)

  *Scales array x by s and stores result in y. y = s * x.*
- void axpy (const QOCOFloat *x, const QOCOFloat *y, QOCOFloat *z, QOCOFloat a, QOCOInt n)

  *Computes z = a * x + y.*
- void USpMv (const QOCOCscMatrix *M, const QOCOFloat *v, QOCOFloat *r)

  *Sparse matrix vector multiplication for CSC matrices where M is symmetric and only the upper triangular part is given. Computes r = M * v.*
- void SpMv (const QOCOCscMatrix *M, const QOCOFloat *v, QOCOFloat *r)

  *Sparse matrix vector multiplication for CSC matrices. Computes r = M * v.*
- void SpMtv (const QOCOCscMatrix *M, const QOCOFloat *v, QOCOFloat *r)

  *Sparse matrix vector multiplication for CSC matrices where M is first transposed. Computes r = M^T * v.*
- QOCOFloat inf_norm (const QOCOFloat *x, QOCOInt n)

  Computes the infinity norm of x.
- QOCOInt regularize (QOCOCscMatrix *M, QOCOFloat lambda, QOCOInt *nzadded_idx)

  *Adds lambda * I to a CSC matrix. Called on P prior to construction of KKT system in qoco_setup(). This function calls realloc() when adding new nonzeros.*
- void unregularize (QOCOCscMatrix *M, QOCOFloat lambda)

  *Subtracts lambda * I to a CSC matrix. Called on P when updating matrix data in update_matrix_data(). This function does not allocate and must be called after regularize.*
- void col_inf_norm_USymm (const QOCOCscMatrix *M, QOCOFloat *norm)

  Computes the infinity norm of each column (or equivalently row) of a symmetric sparse matrix M where only the upper triangular portion of M is given.
- void row_inf_norm (const QOCOCscMatrix *M, QOCOFloat *norm)

  Computes the infinity norm of each row of M and stores in norm.
- QOCOCscMatrix * create_transposed_matrix (const QOCOCscMatrix *A)

  Allocates and computes A^T.
- void row_col_scale (const QOCOCscMatrix *M, QOCOFloat *E, QOCOFloat *D)

  *Scales the rows of M by E and columns of M by D. M = diag(E) * M * diag(S)*
- void ew_product (QOCOFloat *x, const QOCOFloat *y, QOCOFloat *z, QOCOInt n)

  Computes elementwise product z = x . y.*
- void invert_permutation (const QOCOInt *p, QOCOInt *pinv, QOCOInt n)

  Inverts permutation vector p and stores inverse in pinv.
- QOCOInt cumsum (QOCOInt *p, QOCOInt *c, QOCOInt n)

  Computes cumulative sum of c.
- QOCOCscMatrix * csc_symperm (const QOCOCscMatrix *A, const QOCOInt *pinv, QOCOInt *AtoC)

  C = A(p,p) = PAP' where A and C are symmetric and the upper triangular part is stored.

5.18.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.18.2 LICENSE

5.18.3 Function Documentation

5.18.3.1 axpy()

```
void axpy (
            const QOCOFloat * x,
            const QOCOFloat * y,
            QOCOFloat * z,
            QOCOFloat a,
            QOCOInt n )
```

Computes z = a ∗ x + y.

Parameters

x	Input vector.
y	Input vector.
z	Result vector.
a	Scaling factor.
n	Length of vectors.

5.18.3.2 col_inf_norm_USymm()

```
void col_inf_norm_USymm (
            const QOCOCscMatrix * M,
            QOCOFloat * norm )
```

Computes the infinity norm of each column (or equivalently row) of a symmetric sparse matrix M where only the upper triangular portion of M is given.

Parameters

M	Upper triangular part of sparse symmetric matrix.
norm	Result vector of length n.

5.18.3.3 construct_identity()

```
QOCOCscMatrix* construct_identity (
            QOCOInt n,
            QOCOFloat lambda )
```

Allocates a new csc matrix that is lambda $*$ I.

Parameters

n	Size of identity matrix.
lambda	Scaling factor for identity.

Returns

Pointer to new constructed matrix.

5.18.3.4 copy_and_negate_arrayf()

```
void copy_and_negate_arrayf (
            const QOCOFloat * x,
            QOCOFloat * y,
            QOCOInt n )
```

Copies and negates array of QOCOFloats from x to array y.

Parameters

x	Source array.
y	Destination array.
n	Length of arrays.

5.18.3.5 copy_arrayf()

```
void copy_arrayf (
            const QOCOFloat * x,
            QOCOFloat * y,
            QOCOInt n )
```

Copies array of QOCOFloats from x to array y.

Parameters

x	Source array.
y	Destination array.
n	Length of arrays.

5.18.3.6 copy_arrayi()

```
void copy_arrayi (
            const QOCOInt * x,
            QOCOInt * y,
            QOCOInt n )
```

Copies array of QOCOInts from x to array y.

Parameters

x	Source array.
y	Destination array.
n	Length of arrays.

5.18.3.7 create_transposed_matrix()

```
QOCOCscMatrix* create_transposed_matrix (
            const QOCOCscMatrix * A )
```

Allocates and computes A^T.

Parameters

A	Input matrix.

5.18.3.8 csc_symperm()

```
QOCOCscMatrix* csc_symperm (
            const QOCOCscMatrix * A,
            const QOCOInt * pinv,
            QOCOInt * AtoC )
```

C = A(p,p) = PAP' where A and C are symmetric and the upper triangular part is stored.

Parameters

A	
pinv	
AtoC	

Returns

QOCOCscMatrix*

5.18.3.9 cumsum()

```
QOCOInt cumsum (
              QOCOInt * p,
              QOCOInt * c,
              QOCOInt n )
```

Computes cumulative sum of c.

Returns

Cumulative sum of c.

5.18.3.10 dot()

```
QOCOFloat dot (
              const QOCOFloat * u,
              const QOCOFloat * v,
              QOCOInt n )
```

Computes dot product of u and v.

Parameters

u	Input vector.
v	Input vector.
n	Length of vectors.

Returns

Dot product of u and v.

5.18.3.11 ew_product()

```
void ew_product (
             QOCOFloat * x,
             const QOCOFloat * y,
             QOCOFloat * z,
             QOCOInt n )
```

Computes elementwise product z = x .* y.

Parameters

x	Input array.
y	Input array.
z	Output array.
n	Length of arrays.

5.18.3.12 free_qoco_csc_matrix()

```
void free_qoco_csc_matrix (
             QOCOCscMatrix * A )
```

Frees all the internal arrays and the pointer to the QOCOCscMatrix. Should only be used if QOCOCscMatrix and all internal arrays were malloc'ed.

Parameters

A	Pointer to QOCOCscMatrix.

5.18.3.13 inf_norm()

```
QOCOFloat inf_norm (
             const QOCOFloat * x,
             QOCOInt n )
```

Computes the infinity norm of x.

Parameters

x	Input vector.
n	Length of input vector.

Returns

Infinity norm of x.

5.18.3.14 invert_permutation()

```
void invert_permutation (
            const QOCOInt * p,
            QOCOInt * pinv,
            QOCOInt n )
```

Inverts permutation vector p and stores inverse in pinv.

Parameters

p	Input permutation vector.
pinv	Inverse of permutation vector.
n	Length of vectors.

5.18.3.15 max_arrayi()

```
QOCOInt max_arrayi (
            const QOCOInt * x,
            QOCOInt n )
```

Computes maximum element of array of QOCOInts.

Parameters

x	Input array.
n	Length of array.

Returns

Maximum element of x.

5.18.3.16 new_qoco_csc_matrix()

```
QOCOCscMatrix* new_qoco_csc_matrix (
            const QOCOCscMatrix * A )
```

Allocates a new csc matrix and copies A to it.

Parameters

A	Matrix to copy.

Returns

Pointer to new constructed matrix.

5.18.3.17 regularize()

```
QOCOInt regularize (
            QOCOCscMatrix * M,
            QOCOFloat lambda,
            QOCOInt * nzadded_idx )
```

Adds lambda $*$ I to a CSC matrix. Called on P prior to construction of KKT system in qoco_setup(). This function calls realloc() when adding new nonzeros.

Parameters

M	Matrix to be regularized.
lambda	Regularization factor.
nzadded_idx	Indices of elements of M->x that are added.

Returns

Number of nonzeros added to M->x.

5.18.3.18 row_col_scale()

```
void row_col_scale (
            const QOCOCscMatrix * M,
            QOCOFloat * E,
            QOCOFloat * D )
```

Scales the rows of M by E and columns of M by D. M = diag(E) $*$ M $*$ diag(S)

Parameters

M	An m by n sparse matrix.
E	Vector of length m.
D	Vector of length m.

5.18.3.19 row_inf_norm()

```
void row_inf_norm (
            const QOCOCscMatrix * M,
            QOCOFloat * norm )
```

Computes the infinity norm of each row of M and stores in norm.

Parameters

M	An m by n sparse matrix.
norm	Result vector of length m.

5.18.3.20 scale_arrayf()

```
void scale_arrayf (
            const QOCOFloat * x,
            QOCOFloat * y,
            QOCOFloat s,
            QOCOInt n )
```

Scales array x by s and stores result in y. y = s ∗ x.

Parameters

x	Input array.
y	Output array.
s	Scaling factor.
n	Length of arrays.

5.18.3.21 SpMtv()

```
void SpMtv (
            const QOCOCscMatrix * M,
            const QOCOFloat * v,
            QOCOFloat * r )
```

Sparse matrix vector multiplication for CSC matrices where M is first transposed. Computes r = M^T ∗ v.

Parameters

M	Matrix in CSC form.
v	Vector.
r	Result.

5.18.3.22 SpMv()

```
void SpMv (
            const QOCOCscMatrix * M,
```

```
            const QOCOFloat * v,
            QOCOFloat * r )
```

Sparse matrix vector multiplication for CSC matrices. Computes r = M ∗ v.

Parameters

M	Matrix in CSC form.
v	Vector.
r	Result.

5.18.3.23 unregularize()

```
void unregularize (
            QOCOCscMatrix * M,
            QOCOFloat lambda )
```

Subtracts lambda ∗ I to a CSC matrix. Called on P when updating matrix data in update_matrix_data(). This function does not allocate and must be called after regularize.

Parameters

M	Matrix.
lambda	Regularization.

5.18.3.24 USpMv()

```
void USpMv (
            const QOCOCscMatrix * M,
            const QOCOFloat * v,
            QOCOFloat * r )
```

Sparse matrix vector multiplication for CSC matrices where M is symmetric and only the upper triangular part is given. Computes r = M ∗ v.

Parameters

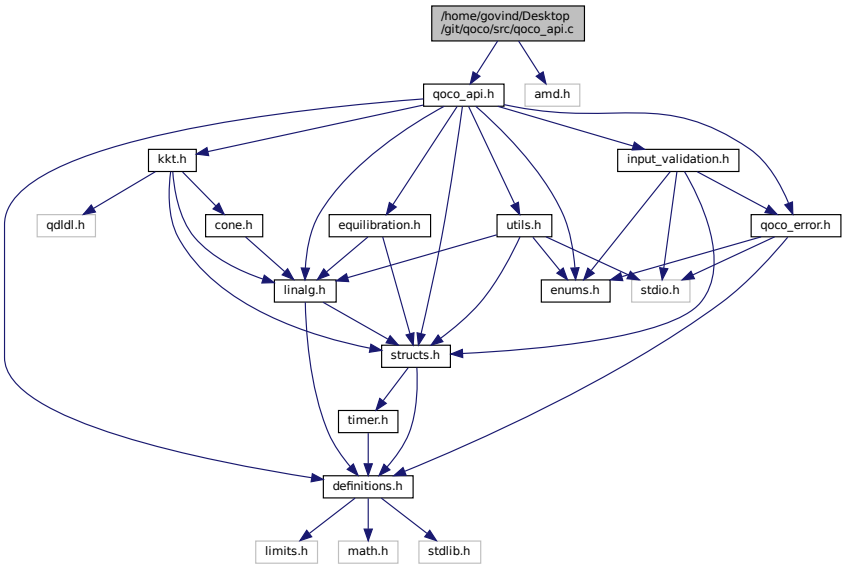
M	Upper triangular part of M in CSC form.
v	Vector.
r	Result.

5.19 /home/govind/Desktop/git/qoco/src/qoco_api.c File Reference

```
#include "qoco_api.h"
```

```
#include "amd.h"
```

Include dependency graph for qoco_api.c:



Functions

- QOCOInt qoco_setup (QOCOSolver *solver, QOCOInt n, QOCOInt m, QOCOInt p, QOCOCscMatrix *P, QOCOFloat *c, QOCOCscMatrix *A, QOCOFloat *b, QOCOCscMatrix *G, QOCOFloat *h, QOCOInt l, QOCOInt nsoc, QOCOInt *q, QOCOSettings *settings)

  Allocates all memory needed for QOCO to solve the SOCP.
- void qoco_set_csc (QOCOCscMatrix *A, QOCOInt m, QOCOInt n, QOCOInt Annz, QOCOFloat *Ax, QOCOInt *Ap, QOCOInt *Ai)

  Sets the data for a compressed sparse column matrix.
- void set_default_settings (QOCOSettings *settings)

  Set the default settings struct.
- QOCOInt qoco_update_settings (QOCOSolver *solver, const QOCOSettings *new_settings)

  Updates settings struct.
- void update_vector_data (QOCOSolver *solver, QOCOFloat *cnew, QOCOFloat *bnew, QOCOFloat *hnew)

  Updates data vectors. NULL can be passed in for any vector if that data will not be updated.
- void update_matrix_data (QOCOSolver *solver, QOCOFloat *Pxnew, QOCOFloat *Axnew, QOCOFloat *Gxnew)

  Updates data matrices. NULL can be passed in for any matrix data pointers if that matrix will not be updated. It is assumed that the new matrix will have the same sparsity structure as the existing matrix.
- QOCOInt qoco_solve (QOCOSolver *solver)

  Solves SOCP.
- QOCOInt qoco_cleanup (QOCOSolver *solver)

  Frees all memory allocated by qoco_setup.

5.19.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.19.2 LICENSE

5.19.3 Function Documentation

5.19.3.1 qoco_cleanup()

```
QOCOInt qoco_cleanup (
            QOCOSolver * solver )
```

Frees all memory allocated by qoco_setup.

Parameters

solver	Pointer to solver.

Returns

Exitflag to check (0 for success, failure otherwise)

5.19.3.2 qoco_set_csc()

```
void qoco_set_csc (
            QOCOCscMatrix * A,
            QOCOInt m,
            QOCOInt n,
            QOCOInt Annz,
            QOCOFloat * Ax,
            QOCOInt * Ap,
            QOCOInt * Ai )
```

Sets the data for a compressed sparse column matrix.

Parameters

A	Pointer to the CSC matrix.
m	Number of rows in the matrix.
n	Number of columns in the matrix.
Annz	Number of nonzero elements in the matrix.
Ax	Array of data for the matrix.
Ap	Array of column pointers for the data.
Ai	Array of row indices for data.

5.19.3.3 qoco_setup()

```
QOCOInt qoco_setup (
            QOCOSolver * solver,
            QOCOInt n,
            QOCOInt m,
            QOCOInt p,
            QOCOCscMatrix * P,
            QOCOFloat * c,
            QOCOCscMatrix * A,
            QOCOFloat * b,
            QOCOCscMatrix * G,
            QOCOFloat * h,
            QOCOInt l,
            QOCOInt nsoc,
            QOCOInt * q,
            QOCOSettings * settings )
```

Allocates all memory needed for QOCO to solve the SOCP.

Parameters

solver	Pointer to solver.
n	Number of optimization variables.
m	Number of conic constraints.
p	Number of affine equality constraints.
P	Upper triangular part of quadratic cost Hessian in CSC form.
c	Linear cost vector.
A	Affine equality constraint matrix in CSC form.
b	Affine equality constraint offset vector.
G	Conic constraint matrix in CSC form.
h	Conic constraint offset vector.
l	Dimension of non-negative orthant.
nsoc	Number of second-order cones.
q	Dimension of each second-order cone.
settings	Settings struct.

Returns

0 if no error or flag containing error code.

5.19.3.4 qoco_solve()

```
QOCOInt qoco_solve (
            QOCOSolver * solver )
```

Solves SOCP.

Parameters

solver	Pointer to solver.

Returns

Exitflag to check (0 for success, failure otherwise)

5.19.3.5 qoco_update_settings()

```
QOCOInt qoco_update_settings (
            QOCOSolver * solver,
            const QOCOSettings * new_settings )
```

Updates settings struct.

Parameters

solver	Pointer to solver.
new_settings	New settings struct.

Returns

0 if update is successful.

5.19.3.6 set_default_settings()

```
void set_default_settings (
            QOCOSettings * settings )
```

Set the default settings struct.

Parameters

settings	Pointer to settings struct.

5.19.3.7 update_matrix_data()

```
void update_matrix_data (
            QOCOSolver * solver,
```

```
            QOCOFloat * Pxnew,
            QOCOFloat * Axnew,
            QOCOFloat * Gxnew )
```

Updates data matrices. NULL can be passed in for any matrix data pointers if that matrix will not be updated. It is assumed that the new matrix will have the same sparsity structure as the existing matrix.

Parameters

solver	Pointer to solver.
Pxnew	New data for P->x.
Axnew	New data for A->x.
Gxnew	New data for G->x.

5.19.3.8 update_vector_data()

```
void update_vector_data (
            QOCOSolver * solver,
            QOCOFloat * cnew,
            QOCOFloat * bnew,
            QOCOFloat * hnew )
```

Updates data vectors. NULL can be passed in for any vector if that data will not be updated.

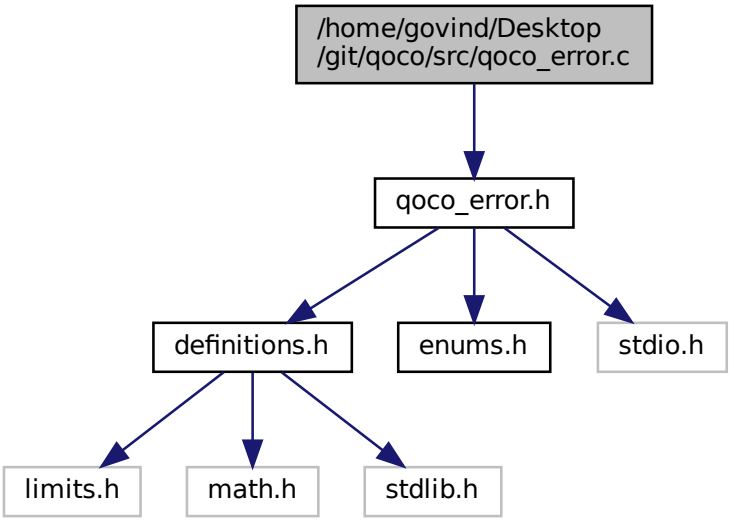
Parameters

solver	Pointer to solver.
cnew	New c vector.
bnew	New b vector.
hnew	New h vector.

5.20 /home/govind/Desktop/git/qoco/src/qoco_error.c File Reference

```
#include "qoco_error.h"
```

Include dependency graph for qoco_error.c:



Functions

- QOCOInt qoco_error (enum qoco_error_code error_code)

  Function to print error messages.

5.20.1 Function Documentation

5.20.1.1 qoco_error()

```
QOCOInt qoco_error (
            enum qoco_error_code error_code )
```

Function to print error messages.

Parameters

error_code	

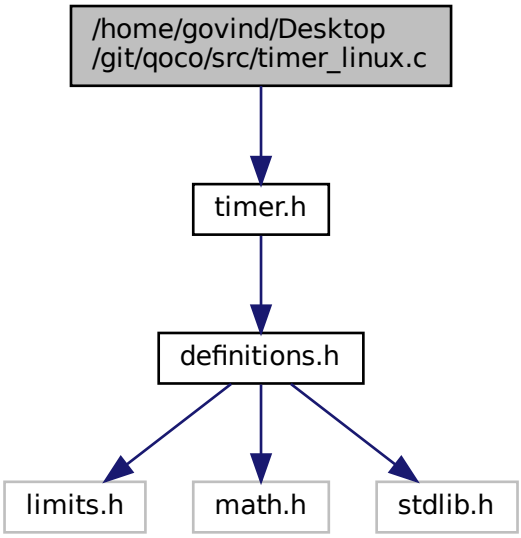
Returns

Error code as an QOCOInt.

5.21 /home/govind/Desktop/git/qoco/src/timer_linux.c File Reference

```
#include "timer.h"
```

Include dependency graph for timer_linux.c:



Functions

- void start_timer (QOCOTimer *timer)

  Starts timer and sets tic field of struct to the current time.
- void stop_timer (QOCOTimer *timer)

  Stops timer and sets toc field of struct to the current time.
- QOCOFloat get_elapsed_time_sec (QOCOTimer *timer)

  Gets time in seconds recorded by timer. Must be called after start_timer() and stop_timer().

5.21.1 Function Documentation

5.21.1.1 get_elapsed_time_sec()

```
QOCOFloat get_elapsed_time_sec (
            QOCOTimer * timer )
```

Gets time in seconds recorded by timer. Must be called after start_timer() and stop_timer().

Parameters

timer	Pointer to timer struct.

5.21.1.2 start_timer()

```
void start_timer (
            QOCOTimer * timer )
```

Starts timer and sets tic field of struct to the current time.

Parameters

timer	Pointer to timer struct.

5.21.1.3 stop_timer()

```
void stop_timer (
            QOCOTimer * timer )
```

Stops timer and sets toc field of struct to the current time.

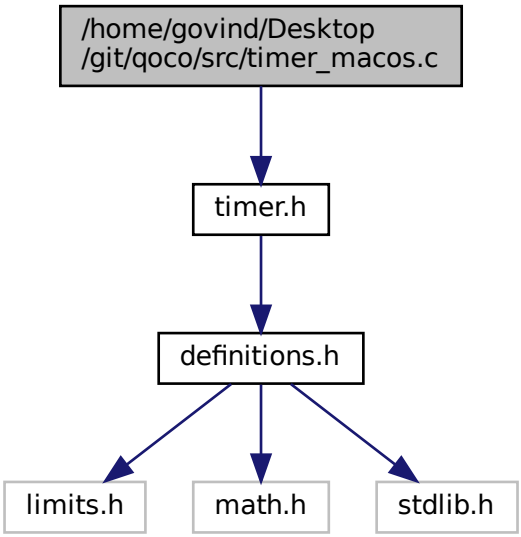
Parameters

timer	Pointer to timer struct.

5.22 /home/govind/Desktop/git/qoco/src/timer_macos.c File Reference

```
#include "timer.h"
```

Include dependency graph for timer_macos.c:



Functions

- void start_timer (QOCOTimer *timer)

  Starts timer and sets tic field of struct to the current time.
- void stop_timer (QOCOTimer *timer)

  Stops timer and sets toc field of struct to the current time.
- QOCOFloat get_elapsed_time_sec (QOCOTimer *timer)

  Gets time in seconds recorded by timer. Must be called after start_timer() and stop_timer().

5.22.1 Function Documentation

5.22.1.1 get_elapsed_time_sec()

```
QOCOFloat get_elapsed_time_sec (
            QOCOTimer * timer )
```

Gets time in seconds recorded by timer. Must be called after start_timer() and stop_timer().

Parameters

timer	Pointer to timer struct.

5.22.1.2 start_timer()

```
void start_timer (
            QOCOTimer * timer )
```

Starts timer and sets tic field of struct to the current time.

Parameters

timer	Pointer to timer struct.

5.22.1.3 stop_timer()

```
void stop_timer (
            QOCOTimer * timer )
```

Stops timer and sets toc field of struct to the current time.

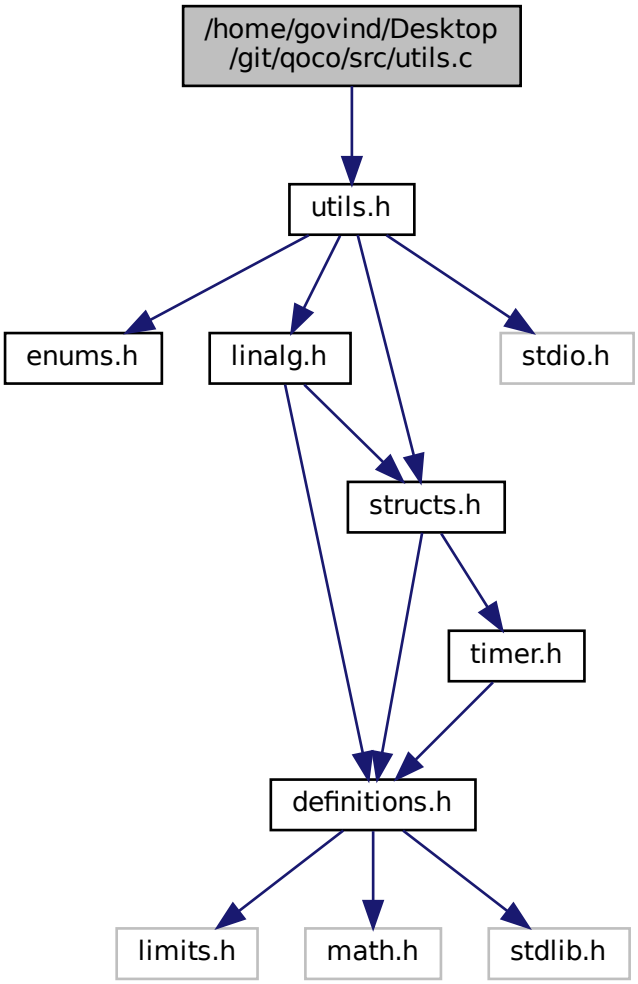
Parameters

timer	Pointer to timer struct.

5.23 /home/govind/Desktop/git/qoco/src/utils.c File Reference

```
#include "utils.h"
```

Include dependency graph for utils.c:



Functions

- void print_qoco_csc_matrix (QOCOCscMatrix *M)

  Prints dimensions, number of nonzero elements, data, column pointers and row indices for a sparse matrix in CSC form.
- void print_arrayf (QOCOFloat *x, QOCOInt n)

  Prints array of QOCOFloats.
- void print_arrayi (QOCOInt *x, QOCOInt n)

  Prints array of QOCOInts.
- void print_header (QOCOSolver *solver)

  Prints QOCO header.
- void log_iter (QOCOSolver *solver)

  Print solver progress.
- void print_footer (QOCOSolution *solution, enum qoco_solve_status status)

Prints QOCO footer.

- unsigned char check_stopping (QOCOSolver *solver)

  Checks stopping criteria. Before calling this function, work->kkt->rhs must contain the residual of the KKT conditions as computed by compute_kkt_residual().

- void copy_solution (QOCOSolver *solver)

  Copies data to QOCOSolution struct when solver terminates.

- QOCOSettings * copy_settings (QOCOSettings *settings)

  Allocates and returns a copy of the input settings struct.

5.23.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.23.2 LICENSE

5.23.3 Function Documentation

5.23.3.1 check_stopping()

```
unsigned char check_stopping (
            QOCOSolver * solver )
```

Checks stopping criteria. Before calling this function, work->kkt->rhs must contain the residual of the KKT conditions as computed by compute_kkt_residual().

Parameters

solver	Pointer to solver.

Returns

1 if stopping criteria met and 0 otherwise.

5.23.3.2 copy_settings()

```
QOCOSettings* copy_settings (
            QOCOSettings * settings )
```

Allocates and returns a copy of the input settings struct.

Parameters

settings	Input struct.

Returns

Pointer to constructed and copies settings struct.

5.23.3.3 copy_solution()

```
void copy_solution (
            QOCOSolver * solver )
```

Copies data to QOCOSolution struct when solver terminates.

Parameters

solver	Pointer to solver.

5.23.3.4 log_iter()

```
void log_iter (
            QOCOSolver * solver )
```

Print solver progress.

Parameters

solver	Pointer to solver.

5.23.3.5 print_arrayf()

```
void print_arrayf (
            QOCOFloat * x,
            QOCOInt n )
```

Prints array of QOCOFloats.

Parameters

x	Pointer to array.
n	Number of elements in array.

5.23.3.6 print_arrayi()

```
void print_arrayi (
            QOCOInt * x,
            QOCOInt n )
```

Prints array of QOCOInts.

Parameters

x	Pointer to array.
n	Number of elements in array.

5.23.3.7 print_footer()

```
void print_footer (
            QOCOSolution * solution,
            enum qoco_solve_status status )
```

Prints QOCO footer.

Parameters

solution	Pointer to solution struct.
status	Solve status.

5.23.3.8 print_header()

```
void print_header (
            QOCOSolver * solver )
```

Prints QOCO header.

Parameters

solver	Pointer to solver.

5.23.3.9 print_qoco_csc_matrix()

```
void print_qoco_csc_matrix (
```

```
QOCOCscMatrix * M )
```

Prints dimensions, number of nonzero elements, data, column pointers and row indices for a sparse matrix in CSC form.

Parameters

M	Pointer to QOCOCscMatrix that will be printed.

Index