

My Project

Generated by Doxygen 1.9.1

1 Todo List	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 QOCOCscMatrix Struct Reference	7
4.1.1 Detailed Description	7
4.1.2 Member Data Documentation	7
4.1.2.1 i	7
4.1.2.2 m	8
4.1.2.3 n	8
4.1.2.4 nnz	8
4.1.2.5 p	8
4.1.2.6 x	8
4.2 QOCOKKT Struct Reference	8
4.2.1 Detailed Description	9
4.2.2 Member Data Documentation	9
4.2.2.1 AtoKKT	9
4.2.2.2 bwork	10
4.2.2.3 D	10
4.2.2.4 delta	10
4.2.2.5 Dinv	10
4.2.2.6 Dinvruiz	10
4.2.2.7 Druiz	10
4.2.2.8 Einvruiz	10
4.2.2.9 Eruiz	10
4.2.2.10 etree	11
4.2.2.11 Finvruiz	11
4.2.2.12 Fruiz	11
4.2.2.13 fwork	11
4.2.2.14 GtoKKT	11
4.2.2.15 iwork	11
4.2.2.16 K	11
4.2.2.17 k	11
4.2.2.18 kinv	12
4.2.2.19 kktres	12
4.2.2.20 Li	12
4.2.2.21 Lnz	12
4.2.2.22 Lp	12

4.2.2.23 Lx	12
4.2.2.24 nt2kkt	12
4.2.2.25 ntdiag2kkt	12
4.2.2.26 p	13
4.2.2.27 pinv	13
4.2.2.28 Pnum_nzadded	13
4.2.2.29 Pnzadded_idx	13
4.2.2.30 PregtoKKT	13
4.2.2.31 rhs	13
4.2.2.32 xyz	13
4.2.2.33 xyzbuff1	13
4.2.2.34 xyzbuff2	14
4.3 QOCOPProblemData Struct Reference	14
4.3.1 Detailed Description	15
4.3.2 Member Data Documentation	15
4.3.2.1 A	15
4.3.2.2 At	15
4.3.2.3 b	15
4.3.2.4 c	15
4.3.2.5 G	15
4.3.2.6 Gt	15
4.3.2.7 h	16
4.3.2.8 l	16
4.3.2.9 m	16
4.3.2.10 n	16
4.3.2.11 nsoc	16
4.3.2.12 P	16
4.3.2.13 p	16
4.3.2.14 q	17
4.4 QOCOSettings Struct Reference	17
4.4.1 Detailed Description	17
4.4.2 Member Data Documentation	17
4.4.2.1 abstol	17
4.4.2.2 abstol_inacc	18
4.4.2.3 bisect_iters	18
4.4.2.4 iter_ref_iters	18
4.4.2.5 kkt_dynamic_reg	18
4.4.2.6 kkt_static_reg	18
4.4.2.7 max_iters	18
4.4.2.8 reltol	18
4.4.2.9 reltol_inacc	18
4.4.2.10 ruiz_iters	19

4.4.2.11 verbose	19
4.5 QOCOSolution Struct Reference	19
4.5.1 Member Data Documentation	19
4.5.1.1 dres	19
4.5.1.2 gap	20
4.5.1.3 iters	20
4.5.1.4 obj	20
4.5.1.5 pres	20
4.5.1.6 s	20
4.5.1.7 setup_time_sec	20
4.5.1.8 solve_time_sec	20
4.5.1.9 status	20
4.5.1.10 x	21
4.5.1.11 y	21
4.5.1.12 z	21
4.6 QOCOSolver Struct Reference	21
4.6.1 Detailed Description	22
4.6.2 Member Data Documentation	22
4.6.2.1 settings	22
4.6.2.2 sol	22
4.6.2.3 work	22
4.7 QOCOWorkspace Struct Reference	23
4.7.1 Detailed Description	24
4.7.2 Member Data Documentation	24
4.7.2.1 a	24
4.7.2.2 data	24
4.7.2.3 Ds	24
4.7.2.4 kkt	24
4.7.2.5 lambda	24
4.7.2.6 mu	25
4.7.2.7 s	25
4.7.2.8 sbar	25
4.7.2.9 sigma	25
4.7.2.10 solve_timer	25
4.7.2.11 ubuff1	25
4.7.2.12 ubuff2	25
4.7.2.13 ubuff3	25
4.7.2.14 W	26
4.7.2.15 Wfull	26
4.7.2.16 Winv	26
4.7.2.17 Winvfull	26
4.7.2.18 Wnnz	26

4.7.2.19 Wnnzfull	26
4.7.2.20 WtW	26
4.7.2.21 x	26
4.7.2.22 xbuff	27
4.7.2.23 y	27
4.7.2.24 ybuff	27
4.7.2.25 z	27
4.7.2.26 zbar	27
5 File Documentation	29
5.1 /home/govind/Desktop/git/qoco/include/cone.h File Reference	29
5.1.1 Detailed Description	31
5.1.2 LICENSE	31
5.1.3 DESCRIPTION	31
5.1.4 Function Documentation	31
5.1.4.1 bisection_search()	31
5.1.4.2 bring2cone()	32
5.1.4.3 compute_centering()	32
5.1.4.4 compute_mu()	32
5.1.4.5 compute_nt_scaling()	33
5.1.4.6 cone_division()	33
5.1.4.7 cone_product()	33
5.1.4.8 cone_residual()	34
5.1.4.9 exact_linesearch()	34
5.1.4.10 linesearch()	35
5.1.4.11 nt_multiply()	35
5.1.4.12 soc_division()	36
5.1.4.13 soc_product()	36
5.1.4.14 soc_residual()	37
5.1.4.15 soc_residual2()	37
5.2 /home/govind/Desktop/git/qoco/include/definitions.h File Reference	37
5.2.1 Detailed Description	39
5.2.2 LICENSE	39
5.2.3 DESCRIPTION	39
5.2.4 Macro Definition Documentation	39
5.2.4.1 qoco_abs	39
5.2.4.2 qoco_assert	39
5.2.4.3 qoco_calloc	39
5.2.4.4 qoco_free	39
5.2.4.5 qoco_malloc	40
5.2.4.6 qoco_max	40
5.2.4.7 qoco_min	40

5.2.4.8 qoco_sqrt	40
5.2.4.9 QOCFloat_MAX	40
5.2.4.10 QOCInt_MAX	40
5.2.4.11 safe_div	40
5.2.5 Typedef Documentation	41
5.2.5.1 QOCFloat	41
5.2.5.2 QOCInt	41
5.3 /home/govind/Desktop/git/qoco/include/enums.h File Reference	41
5.3.1 Enumeration Type Documentation	41
5.3.1.1 qoco_error_code	41
5.3.1.2 qoco_solve_status	42
5.4 /home/govind/Desktop/git/qoco/include/equilibration.h File Reference	42
5.4.1 Detailed Description	44
5.4.2 LICENSE	44
5.4.3 DESCRIPTION	44
5.4.4 Function Documentation	44
5.4.4.1 ruiz_equilibration()	44
5.4.4.2 unscale_variables()	45
5.5 /home/govind/Desktop/git/qoco/include/input_validation.h File Reference	45
5.5.1 Detailed Description	46
5.5.2 LICENSE	46
5.5.3 DESCRIPTION	46
5.5.4 Function Documentation	46
5.5.4.1 qoco_validate_data()	47
5.5.4.2 qoco_validate_settings()	47
5.6 /home/govind/Desktop/git/qoco/include/kkt.h File Reference	48
5.6.1 Detailed Description	49
5.6.2 LICENSE	50
5.6.3 DESCRIPTION	50
5.6.4 Function Documentation	50
5.6.4.1 allocate_kkt()	50
5.6.4.2 compute_kkt_residual()	50
5.6.4.3 construct_kkt()	51
5.6.4.4 construct_kkt_aff_rhs()	51
5.6.4.5 construct_kkt_comb_rhs()	51
5.6.4.6 initialize_ipm()	52
5.6.4.7 kkt_multiply()	52
5.6.4.8 kkt_solve()	52
5.6.4.9 predictor_corrector()	53
5.6.4.10 set_nt_block_zeros()	53
5.6.4.11 update_nt_block()	53
5.7 /home/govind/Desktop/git/qoco/include/linalg.h File Reference	53

5.7.1 Detailed Description	56
5.7.2 LICENSE	56
5.7.3 DESCRIPTION	56
5.7.4 Function Documentation	56
5.7.4.1 axpy()	56
5.7.4.2 col_inf_norm_USymm()	56
5.7.4.3 construct_identity()	57
5.7.4.4 copy_and_negate_arrayf()	57
5.7.4.5 copy_arrayf()	57
5.7.4.6 copy_arrayi()	58
5.7.4.7 create_transposed_matrix()	58
5.7.4.8 csc_symperm()	58
5.7.4.9 cumsum()	59
5.7.4.10 dot()	59
5.7.4.11 ew_product()	60
5.7.4.12 free_qoco_csc_matrix()	60
5.7.4.13 inf_norm()	60
5.7.4.14 invert_permutation()	61
5.7.4.15 max_arrayi()	61
5.7.4.16 new_qoco_csc_matrix()	62
5.7.4.17 regularize()	62
5.7.4.18 row_col_scale()	62
5.7.4.19 row_inf_norm()	63
5.7.4.20 scale_arrayf()	63
5.7.4.21 SpMtv()	63
5.7.4.22 SpMv()	65
5.7.4.23 unregularize()	65
5.7.4.24 USpMv()	65
5.8 /home/govind/Desktop/git/qoco/include/qoco.h File Reference	66
5.8.1 Detailed Description	66
5.8.2 LICENSE	66
5.8.3 DESCRIPTION	66
5.9 /home/govind/Desktop/git/qoco/include/qoco_api.h File Reference	67
5.9.1 Detailed Description	68
5.9.2 LICENSE	68
5.9.3 DESCRIPTION	68
5.9.4 Function Documentation	68
5.9.4.1 qoco_cleanup()	68
5.9.4.2 qoco_set_csc()	69
5.9.4.3 qoco_setup()	69
5.9.4.4 qoco_solve()	70
5.9.4.5 qoco_update_settings()	70

5.9.4.6 set_default_settings()	71
5.9.4.7 update_matrix_data()	71
5.9.4.8 update_vector_data()	71
5.10 /home/govind/Desktop/git/qoco/include/qoco_error.h File Reference	72
5.10.1 Function Documentation	73
5.10.1.1 qoco_error()	73
5.11 /home/govind/Desktop/git/qoco/include/structs.h File Reference	73
5.11.1 Detailed Description	75
5.11.2 LICENSE	75
5.11.3 DESCRIPTION	75
5.12 /home/govind/Desktop/git/qoco/include/timer.h File Reference	75
5.12.1 Detailed Description	76
5.12.2 LICENSE	76
5.12.3 DESCRIPTION	76
5.12.4 Function Documentation	76
5.12.4.1 get_elapsed_time_sec()	76
5.12.4.2 start_timer()	77
5.12.4.3 stop_timer()	77
5.13 /home/govind/Desktop/git/qoco/include/utils.h File Reference	77
5.13.1 Detailed Description	79
5.13.2 LICENSE	79
5.13.3 DESCRIPTION	79
5.13.4 Function Documentation	79
5.13.4.1 check_stopping()	79
5.13.4.2 copy_settings()	80
5.13.4.3 copy_solution()	80
5.13.4.4 log_iter()	80
5.13.4.5 print_arrayf()	81
5.13.4.6 print_arrayi()	81
5.13.4.7 print_footer()	81
5.13.4.8 print_header()	81
5.13.4.9 print_qoco_csc_matrix()	82
5.14 /home/govind/Desktop/git/qoco/src/cone.c File Reference	82
5.14.1 Detailed Description	84
5.14.2 LICENSE	84
5.14.3 Function Documentation	84
5.14.3.1 bisection_search()	84
5.14.3.2 bring2cone()	85
5.14.3.3 compute_centering()	85
5.14.3.4 compute_mu()	85
5.14.3.5 compute_nt_scaling()	86
5.14.3.6 cone_division()	86

5.14.3.7 cone_product()	86
5.14.3.8 cone_residual()	87
5.14.3.9 exact_linesearch()	87
5.14.3.10 linesearch()	88
5.14.3.11 nt_multiply()	88
5.14.3.12 soc_division()	89
5.14.3.13 soc_product()	89
5.14.3.14 soc_residual()	90
5.14.3.15 soc_residual2()	90
5.15 /home/govind/Desktop/git/qoco/src/equilibration.c File Reference	90
5.15.1 Function Documentation	91
5.15.1.1 ruiz_equilibration()	92
5.15.1.2 unscale_variables()	92
5.16 /home/govind/Desktop/git/qoco/src/input_validation.c File Reference	92
5.16.1 Detailed Description	93
5.16.2 LICENSE	93
5.16.3 Function Documentation	94
5.16.3.1 qoco_validate_data()	94
5.16.3.2 qoco_validate_settings()	94
5.17 /home/govind/Desktop/git/qoco/src/kkt.c File Reference	95
5.17.1 Detailed Description	96
5.17.2 LICENSE	96
5.17.3 Function Documentation	96
5.17.3.1 allocate_kkt()	96
5.17.3.2 compute_kkt_residual()	96
5.17.3.3 construct_kkt()	97
5.17.3.4 construct_kkt_aff_rhs()	97
5.17.3.5 construct_kkt_comb_rhs()	98
5.17.3.6 initialize_ipm()	98
5.17.3.7 kkt_multiply()	98
5.17.3.8 kkt_solve()	99
5.17.3.9 predictor_corrector()	99
5.17.3.10 set_nt_block_zeros()	99
5.17.3.11 update_nt_block()	99
5.18 /home/govind/Desktop/git/qoco/src/linalg.c File Reference	100
5.18.1 Detailed Description	102
5.18.2 LICENSE	102
5.18.3 Function Documentation	102
5.18.3.1 axpy()	102
5.18.3.2 col_inf_norm_USymm()	102
5.18.3.3 construct_identity()	103
5.18.3.4 copy_and_negate_arrayf()	103

5.18.3.5 copy_arrayf()	103
5.18.3.6 copy_arrayi()	104
5.18.3.7 create_transposed_matrix()	104
5.18.3.8 csc_symperm()	104
5.18.3.9 cumsum()	105
5.18.3.10 dot()	105
5.18.3.11 ew_product()	106
5.18.3.12 free_qoco_csc_matrix()	106
5.18.3.13 inf_norm()	106
5.18.3.14 invert_permutation()	107
5.18.3.15 max_arrayi()	107
5.18.3.16 new_qoco_csc_matrix()	107
5.18.3.17 regularize()	108
5.18.3.18 row_col_scale()	108
5.18.3.19 row_inf_norm()	108
5.18.3.20 scale_arrayf()	109
5.18.3.21 SpMtv()	109
5.18.3.22 SpMv()	109
5.18.3.23 unregularize()	110
5.18.3.24 USpMv()	110
5.19 /home/govind/Desktop/git/qoco/src/qoco_api.c File Reference	110
5.19.1 Detailed Description	111
5.19.2 LICENSE	112
5.19.3 Function Documentation	112
5.19.3.1 qoco_cleanup()	112
5.19.3.2 qoco_set_csc()	112
5.19.3.3 qoco_setup()	113
5.19.3.4 qoco_solve()	113
5.19.3.5 qoco_update_settings()	114
5.19.3.6 set_default_settings()	114
5.19.3.7 update_matrix_data()	114
5.19.3.8 update_vector_data()	115
5.20 /home/govind/Desktop/git/qoco/src/qoco_error.c File Reference	115
5.20.1 Function Documentation	116
5.20.1.1 qoco_error()	116
5.21 /home/govind/Desktop/git/qoco/src/timer_linux.c File Reference	117
5.21.1 Function Documentation	117
5.21.1.1 get_elapsed_time_sec()	117
5.21.1.2 start_timer()	118
5.21.1.3 stop_timer()	118
5.22 /home/govind/Desktop/git/qoco/src/timer_macos.c File Reference	118
5.22.1 Function Documentation	119

5.22.1.1 <code>get_elapsed_time_sec()</code>	119
5.22.1.2 <code>start_timer()</code>	120
5.22.1.3 <code>stop_timer()</code>	120
5.23 <code>/home/govind/Desktop/git/qoco/src/utils.c</code> File Reference	120
5.23.1 Detailed Description	122
5.23.2 LICENSE	122
5.23.3 Function Documentation	122
5.23.3.1 <code>check_stopping()</code>	122
5.23.3.2 <code>copy_settings()</code>	122
5.23.3.3 <code>copy_solution()</code>	123
5.23.3.4 <code>log_iter()</code>	123
5.23.3.5 <code>print_arrayf()</code>	123
5.23.3.6 <code>print_arrayi()</code>	124
5.23.3.7 <code>print_footer()</code>	124
5.23.3.8 <code>print_header()</code>	124
5.23.3.9 <code>print_qoco_csc_matrix()</code>	124
Index	127

Chapter 1

Todo List

Member `exact_linesearch` (QOCFloat *u, QOCFloat *Du, QOCFloat 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:

QOCOCscMatrix	Compressed sparse column format matrices	7
QOCOKKT	Contains all data needed for constructing and modifying KKT matrix and performing predictor-corrector step	8
QOCOProblemData	SOC problem data	14
QOCOSettings	QOCO solver settings	17
QOCOSolution	19
QOCOSolver	QOCO Solver struct. Contains all information about the state of the solver	21
QOCOWorkspace	QOCO Workspace	23

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

/home/govind/Desktop/git/qoco/include/ cone.h	29
/home/govind/Desktop/git/qoco/include/ definitions.h	37
/home/govind/Desktop/git/qoco/include/ enums.h	41
/home/govind/Desktop/git/qoco/include/ equilibration.h	42
/home/govind/Desktop/git/qoco/include/ input_validation.h	45
/home/govind/Desktop/git/qoco/include/ kkt.h	48
/home/govind/Desktop/git/qoco/include/ linalg.h	53
/home/govind/Desktop/git/qoco/include/ qoco.h	66
/home/govind/Desktop/git/qoco/include/ qoco_api.h	67
/home/govind/Desktop/git/qoco/include/ qoco_error.h	72
/home/govind/Desktop/git/qoco/include/ structs.h	73
/home/govind/Desktop/git/qoco/include/ timer.h	75
/home/govind/Desktop/git/qoco/include/ utils.h	77
/home/govind/Desktop/git/qoco/src/ cone.c	82
/home/govind/Desktop/git/qoco/src/ equilibration.c	90
/home/govind/Desktop/git/qoco/src/ input_validation.c	92
/home/govind/Desktop/git/qoco/src/ kkt.c	95
/home/govind/Desktop/git/qoco/src/ linalg.c	100
/home/govind/Desktop/git/qoco/src/ qoco_api.c	110
/home/govind/Desktop/git/qoco/src/ qoco_error.c	115
/home/govind/Desktop/git/qoco/src/ timer_linux.c	117
/home/govind/Desktop/git/qoco/src/ timer_macos.c	118
/home/govind/Desktop/git/qoco/src/ utils.c	120

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:

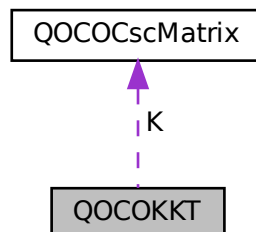
- [/home/govind/Desktop/git/qoco/include/structs.h](#)

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](#) * [Einvrui](#)
- [QOCOFLOAT](#) * [Finvrui](#)
- [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

`QOCFloat* QOCOKKT::Finvruiz`

Inverse of Fruiz.

4.2.2.12 Fruiz

`QOCFloat* QOCOKKT::Fruiz`

Diagonal of scaling matrix.

4.2.2.13 fwork

`QOCFloat* 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

`QOCFloat QOCOKKT::k`

Cost scaling factor.

4.2.2.18 kinv

`QOCOFloat QOCO KKT::kinv`

Inverse of cost scaling factor.

4.2.2.19 kktres

`QOCOFloat* QOCO KKT::kktres`

Residual of KKT condition.

4.2.2.20 Li

`QOCOInt* QOCO KKT::Li`

4.2.2.21 Lnz

`QOCOInt* QOCO KKT::Lnz`

4.2.2.22 Lp

`QOCOInt* QOCO KKT::Lp`

4.2.2.23 Lx

`QOCOFloat* QOCO KKT::Lx`

4.2.2.24 nt2kkt

`QOCOInt* QOCO KKT::nt2kkt`

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

4.2.2.25 ntdiag2kkt

`QOCOInt* QOCO KKT::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 $P \rightarrow x$ that were added due to regularization.

4.2.2.29 Pnzadded_idx

`QOCOInt* QOCOKKT::Pnzadded_idx`

Indices of $P \rightarrow x$ 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

`QOCFloat* QOCOKKT::xyzbuff2`

Buffer of size $n + m + p$.

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

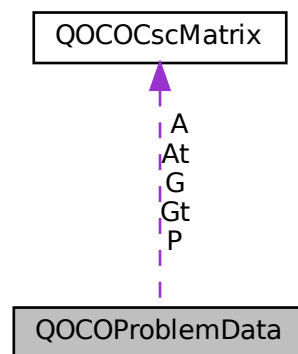
- `/home/govind/Desktop/git/qoco/include/structs.h`

4.3 QOCOProblemData Struct Reference

SOCP problem data.

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

Collaboration diagram for QOCOProblemData:



Public Attributes

- `QOCOCscMatrix * P`
- `QOCFloat * c`
- `QOCOCscMatrix * A`
- `QOCOCscMatrix * At`
- `QOCFloat * b`
- `QOCOCscMatrix * G`
- `QOCOCscMatrix * Gt`
- `QOCFloat * h`
- `QOCInt l`
- `QOCInt nsoc`
- `QOCInt * q`
- `QOCInt n`
- `QOCInt m`
- `QOCInt p`

4.3.1 Detailed Description

SOCP problem data.

4.3.2 Member Data Documentation

4.3.2.1 A

`QOCOCscMatrix*` `QOCOPProblemData::A`

Affine equality constraint matrix.

4.3.2.2 At

`QOCOCscMatrix*` `QOCOPProblemData::At`

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

4.3.2.3 b

`QOCFloat*` `QOCOPProblemData::b`

Affine equality constraint offset.

4.3.2.4 c

`QOCFloat*` `QOCOPProblemData::c`

Linear cost term.

4.3.2.5 G

`QOCOCscMatrix*` `QOCOPProblemData::G`

Conic constraint matrix.

4.3.2.6 Gt

`QOCOCscMatrix*` `QOCOPProblemData::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

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

4.5.1 Member Data Documentation

4.5.1.1 dres

`QOCFloat 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:

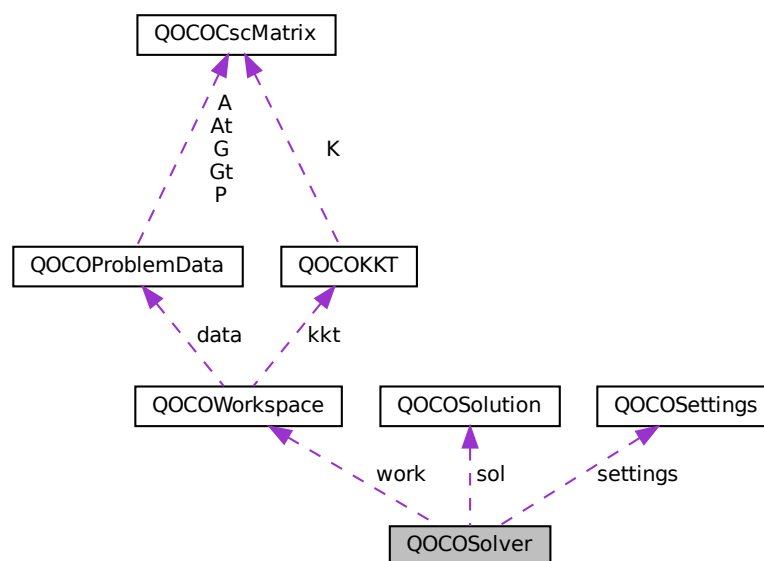
- `/home/govind/Desktop/git/qoco/include/structs.h`

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:

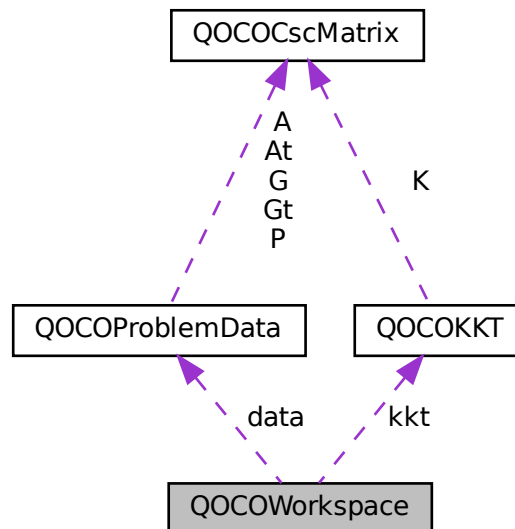
- [/home/govind/Desktop/git/qoco/include/structs.h](#)

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
- QOCOWINT Wnnz
- QOCOWINT 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

`QOCFloat QOCOWorkspace::mu`

Gap (s^*z / m)

4.7.2.7 s

`QOCFloat* QOCOWorkspace::s`

Iterate of slack variables associated with conic constraint.

4.7.2.8 sbar

`QOCFloat* QOCOWorkspace::sbar`

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

4.7.2.9 sigma

`QOCFloat QOCOWorkspace::sigma`

Centering parameter

4.7.2.10 solve_timer

`QOCOTimer QOCOWorkspace::solve_timer`

Solve timer.

4.7.2.11 ubuff1

`QOCFloat* QOCOWorkspace::ubuff1`

Temporary variable of length m .

4.7.2.12 ubuff2

`QOCFloat* QOCOWorkspace::ubuff2`

Temporary variable of length m .

4.7.2.13 ubuff3

`QOCFloat* 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`

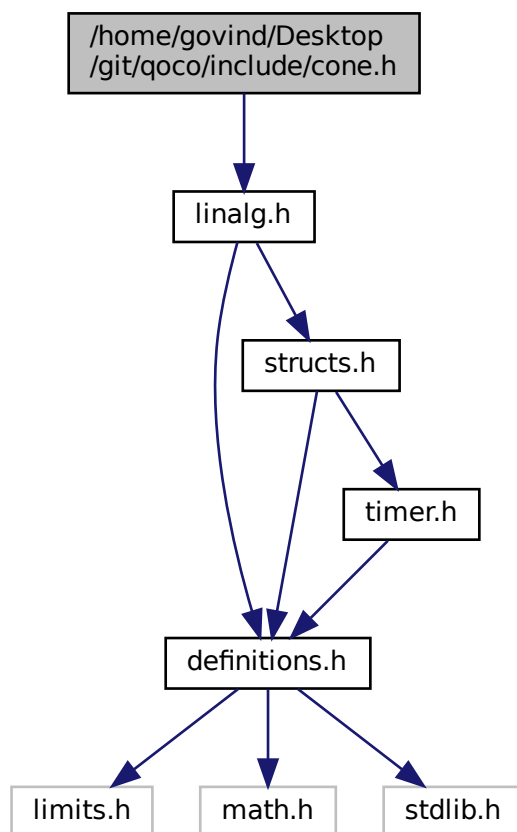
Chapter 5

File Documentation

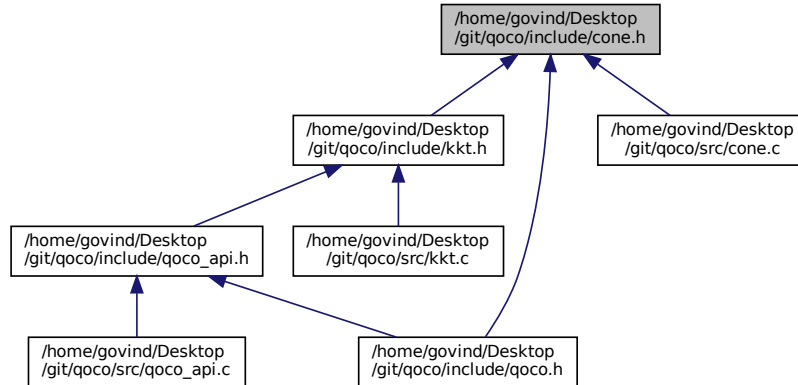
5.1 /home/govind/Desktop/git/qoco/include/cone.h File Reference

```
#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)
Computes 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 $u^0^2 - u^1 * u^1$ 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 canonical vector for each cone LP Cone: $e = \text{ones}(n)$, second-order cone: $e = (1, 0, 0, \dots)$ 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 λ 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 λ 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 λ 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 λ 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

<i>u</i>	Initial vector.
<i>Du</i>	Search direction.
<i>f</i>	Conservatism factor.
<i>solver</i>	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 canonical vector for each cone LP Cone: $e = \text{ones}(n)$, second-order cone: $e = (1, 0, 0, \dots)$ and a is the minimum scalar value such that $u + (1 + a) * e$ is in cone C .

Parameters

<i>u</i>	Vector to bring to cone.
<i>data</i>	Pointer to problem data.

5.1.4.3 compute_centering()

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

Computes centering parameter.

Parameters

<i>solver</i>	Pointer to solver.
---------------	--------------------

5.1.4.4 compute_mu()

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

Computes gap (z^*s / m) and stores in $\text{work} \rightarrow \mu$.

Parameters

<i>work</i>	Pointer to workspace.
-------------	-----------------------

5.1.4.5 compute_nt_scaling()

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

Compute Nesterov-Todd scalings and scaled variables.

Parameters

<i>work</i>	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

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

<i>u</i>	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

<i>u</i>	Initial vector.
<i>Du</i>	Search direction.
<i>f</i>	Conservatism factor.
<i>solver</i>	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 λ in $(0, 1]$ such that $u + (\lambda / f) * Du \in C$. For QPs this calls [exact_linesearch\(\)](#) and for SOCPs this calls [bisection_search\(\)](#)

Parameters

<i>u</i>	Initial vector.
<i>Du</i>	Search direction.
<i>f</i>	Conservatism factor.
<i>solver</i>	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	$u = (u_0, u_1)$ is a vector in second-order cone of dimension n .
n	Dimension of second order cone.

Returns

Residual: $\text{norm}(u_1) - u_0$. 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 $u_0^2 - u_1' u_1$ of vector u with respect to the second order cone of dimension n .

Parameters

u	$u = (u_0, u_1)$ is a vector in second order cone of dimension n .
n	Dimension of second order cone.

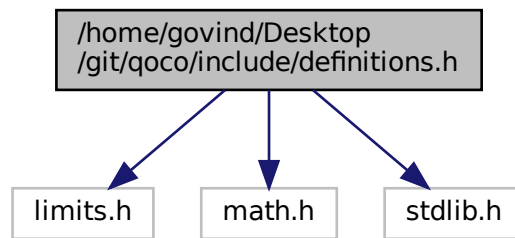
Returns

Residual: $u_0^2 - u_1' u_1$.

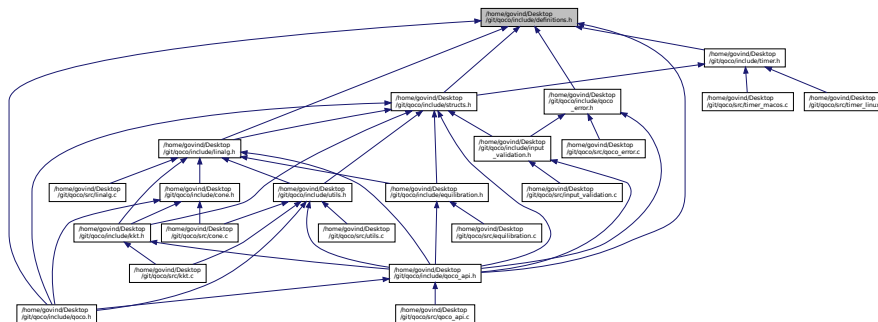
5.2 /home/govind/Desktop/git/qoco/include/definitions.h File Reference

```
#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 QOCFloat_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) : QOCFloat_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 QOCFloat`

5.2.1 Detailed Description

Author

Govind M. Chari govindchari1@gmail.com

5.2.2 LICENSE

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

```
#define QOCFloat_MAX __DBL_MAX__
```

5.2.4.10 QOCInt_MAX

```
#define QOCInt_MAX INT_MAX
```

5.2.4.11 safe_div

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

5.2.5 Typedef Documentation

5.2.5.1 QOCOFloat

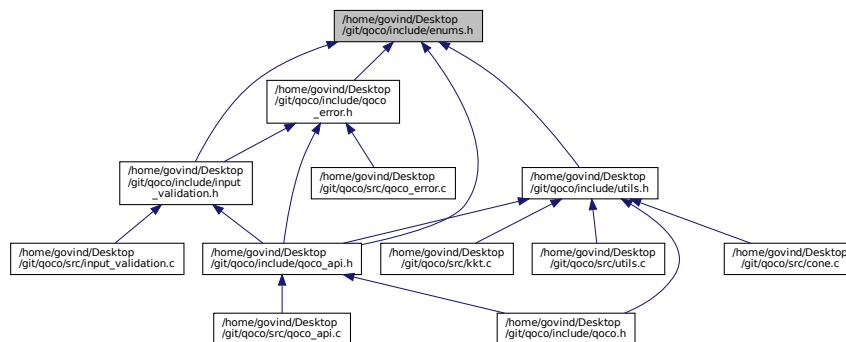
```
typedef double QOCOFloat
```

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_UNUNSOLVED` = 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.

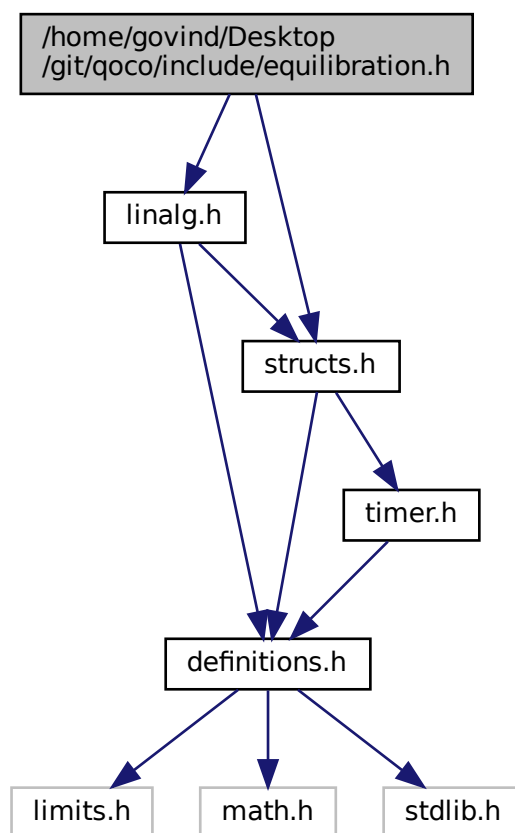
Enumerator

QOCO_UNSOLVED	
QOCO_SOLVED	
QOCO_SOLVED_INACCURATE	
QOCO_NUMERICAL_ERROR	
QOCO_MAX_ITER	

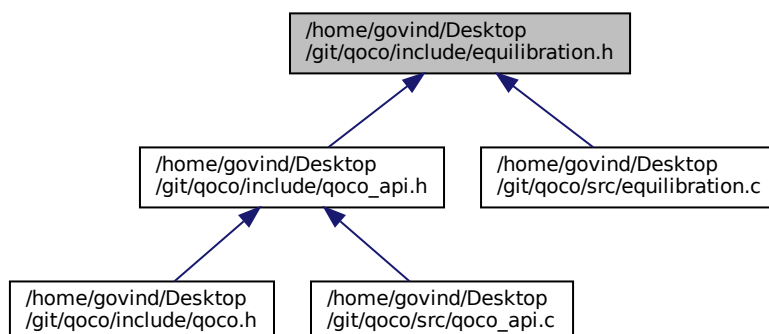
5.4 /home/govind/Desktop/git/qoco/include/equilibration.h File Reference

```
#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

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

$$[D][kPA^TG^T][D]|E||A00||E|[F][G00][F]$$

clang-format on

Parameters

<code>solver</code>	Pointer to solver.
---------------------	--------------------

5.4.4.2 unscale_variables()

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

Undo variable transformation induced by ruiz equilibration.

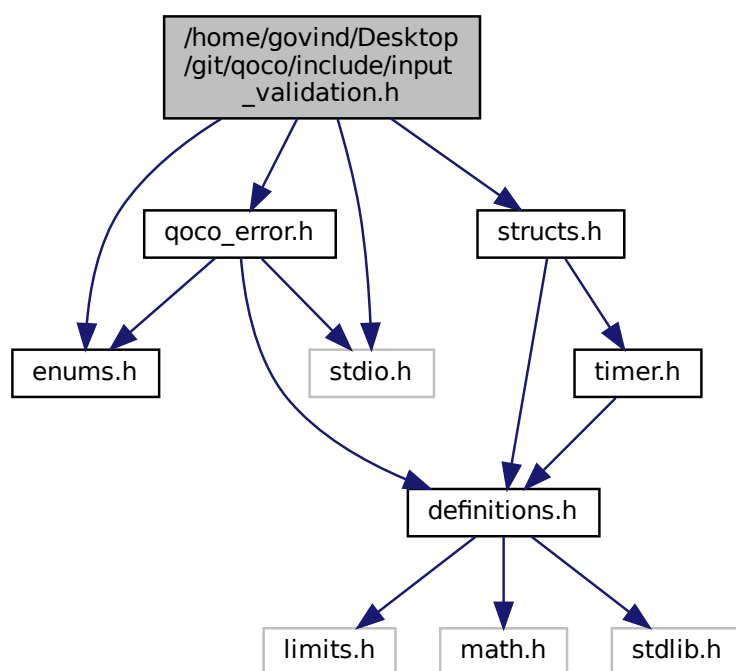
Parameters

<i>work</i>	Pointer to workspace.
-------------	-----------------------

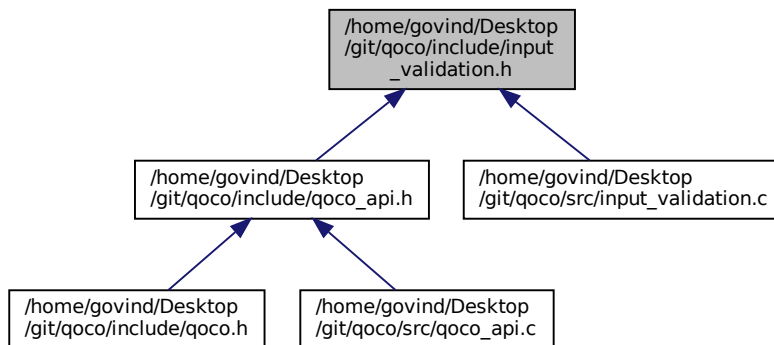
5.5 /home/govind/Desktop/git/qoco/include/input_validation.h File Reference

```
#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

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

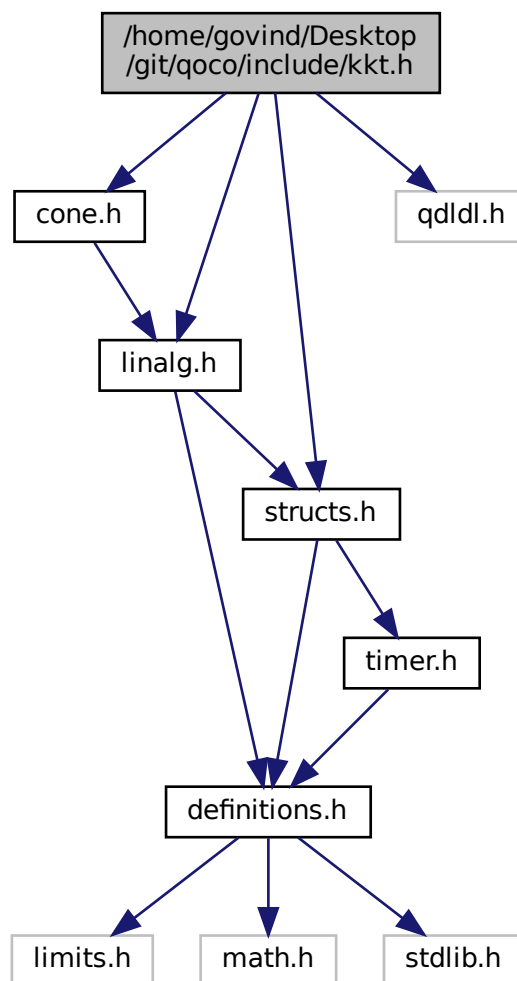
<i>settings</i>	Pointer to settings struct
-----------------	----------------------------

Returns

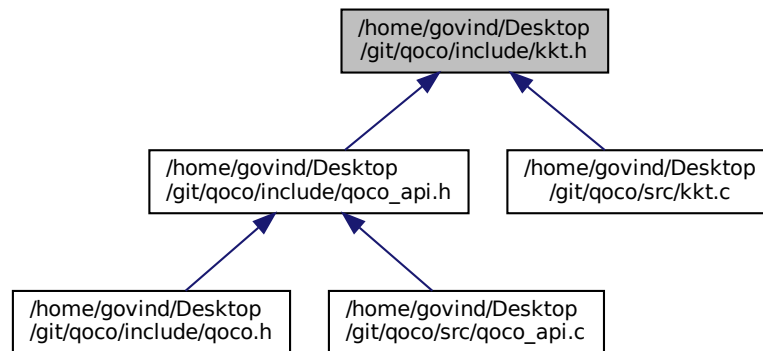
Exitflag to check (0 for success, failure otherwise)

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 Nesterov-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

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

<i>work</i>	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

$$\begin{bmatrix} P & A^T & G^T \end{bmatrix} \begin{bmatrix} x \end{bmatrix} - \begin{bmatrix} c \end{bmatrix}$$

$$\text{res} = \begin{bmatrix} A & 0 & 0 \end{bmatrix} \begin{bmatrix} y \end{bmatrix} + \begin{bmatrix} -b \end{bmatrix} - \begin{bmatrix} G & 0 & 0 \end{bmatrix} \begin{bmatrix} z \end{bmatrix} - \begin{bmatrix} -h + s \end{bmatrix}$$

clang-format on

Parameters

<i>solver</i>	Pointer to solver.
---------------	--------------------

5.6.4.3 construct_kkt()

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

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

clang-format off

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

$$K = \begin{bmatrix} A & 0 & 0 \\ 0 & G & 0 \\ 0 & 0 & -I \end{bmatrix}$$

clang-format on

Parameters

<i>solver</i>	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

<i>work</i>	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

<i>work</i>	Pointer to workspace.
-------------	-----------------------

$ds = -\text{cone_product}(\lambda, \lambda) - \text{settings.mehrotra} * \text{cone_product}((W' \setminus D_{\text{aff}}), (W * D_{\text{aff}}), p_{\text{data}}) + \sigma * \mu * e.$

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

<i>solver</i>	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

<i>solver</i>	Pointer to solver.
<i>x</i>	Pointer to input vector.
<i>y</i>	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

<i>solver</i>	Pointer to solver.
<i>b</i>	Pointer to rhs of kkt system.
<i>iters</i>	Number of iterations of iterative refinement performed.

5.6.4.9 predictor_corrector()

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

Performs Mehrotra predictor-corrector step.

Parameters

<i>solver</i>	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

<i>work</i>	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.

$$\begin{bmatrix} P & A^T & G^T \end{bmatrix}$$

$$K = \begin{bmatrix} A & 0 & 0 \end{bmatrix} \begin{bmatrix} G & 0 & -W'W - e * I \end{bmatrix}$$

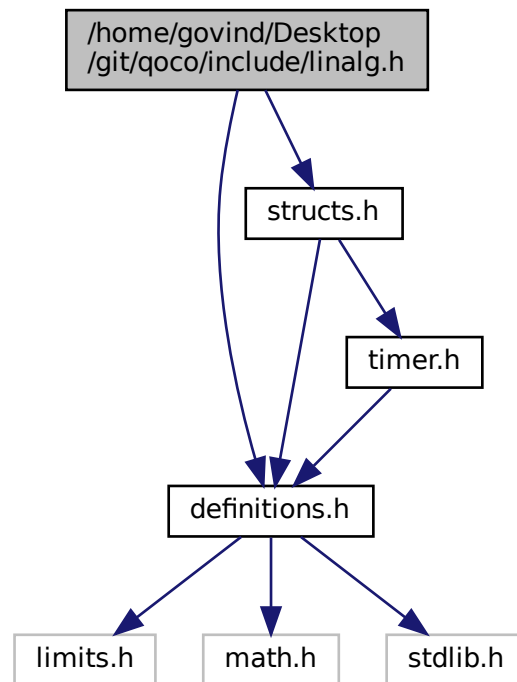
Parameters

<i>solver</i>	Pointer to solver.
---------------	--------------------

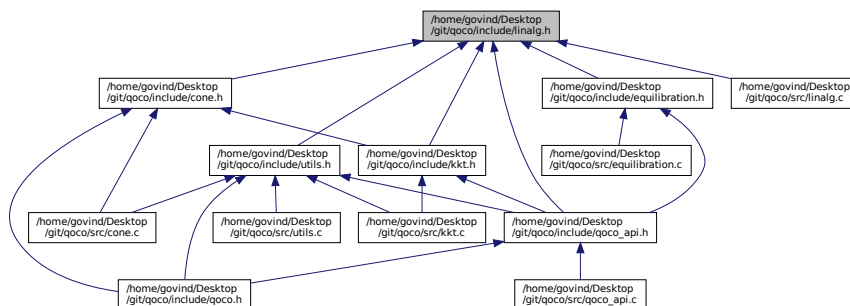
5.7 /home/govind/Desktop/git/qoco/include/linalg.h File Reference

```
#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, QOCFloat 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 [QOCFloat](#) *x, [QOCFloat](#) *y, [QOCInt](#) n)
Copies array of QOCFloats from x to array y.
 - void [copy_and_negate_arrayf](#) (const [QOCFloat](#) *x, [QOCFloat](#) *y, [QOCInt](#) n)
Copies and negates array of QOCFloats from x to array y.
 - void [copy_arrayi](#) (const [QOCInt](#) *x, [QOCInt](#) *y, [QOCInt](#) n)
Copies array of QOCInts from x to array y.
 - [QOCFloat](#) dot (const [QOCFloat](#) *u, const [QOCFloat](#) *v, [QOCInt](#) n)
Computes dot product of u and v.
 - [QOCInt](#) max_arrayi (const [QOCInt](#) *x, [QOCInt](#) n)
Computes maximum element of array of QOCInts.
 - void [scale_arrayf](#) (const [QOCFloat](#) *x, [QOCFloat](#) *y, [QOCFloat](#) s, [QOCInt](#) n)
*Scales array x by s and stores result in y. $y = s * x$.*
 - void [axpy](#) (const [QOCFloat](#) *x, const [QOCFloat](#) *y, [QOCFloat](#) *z, [QOCFloat](#) a, [QOCInt](#) n)
*Computes $z = a * x + y$.*
 - void [USpMv](#) (const [QOCOCscMatrix](#) *M, const [QOCFloat](#) *v, [QOCFloat](#) *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 [QOCFloat](#) *v, [QOCFloat](#) *r)
*Sparse matrix vector multiplication for CSC matrices. Computes $r = M * v$.*
 - void [SpMtv](#) (const [QOCOCscMatrix](#) *M, const [QOCFloat](#) *v, [QOCFloat](#) *r)
*Sparse matrix vector multiplication for CSC matrices where M is first transposed. Computes $r = M^T * v$.*
 - [QOCFloat](#) inf_norm (const [QOCFloat](#) *x, [QOCInt](#) n)
Computes the infinity norm of x.
 - [QOCInt](#) regularize ([QOCOCscMatrix](#) *M, [QOCFloat](#) lambda, [QOCInt](#) *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, [QOCFloat](#) 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, [QOCFloat](#) *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, [QOCFloat](#) *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, [QOCFloat](#) *E, [QOCFloat](#) *D)
*Scales the rows of M by E and columns of M by D. $M = \text{diag}(E) * M * \text{diag}(D)$*
 - void [ew_product](#) ([QOCFloat](#) *x, const [QOCFloat](#) *y, [QOCFloat](#) *z, [QOCInt](#) n)
Computes elementwise product $z = x . y$.*
 - void [invert_permutation](#) (const [QOCInt](#) *p, [QOCInt](#) *pinv, [QOCInt](#) n)
Inverts permutation vector p and stores inverse in pinv.
 - [QOCInt](#) cumsum ([QOCInt](#) *p, [QOCInt](#) *c, [QOCInt](#) n)
Computes cumulative sum of c.
 - [QOCOCscMatrix](#) * [csc_symperm](#) (const [QOCOCscMatrix](#) *A, const [QOCInt](#) *pinv, [QOCInt](#) *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

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

<i>M</i>	Upper triangular part of sparse symmetric matrix.
<i>norm</i>	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

<i>n</i>	Size of identity matrix.
<i>lambda</i>	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

<i>x</i>	Source array.
<i>y</i>	Destination array.
<i>n</i>	Length of arrays.

5.7.4.5 **copy_arrayf()**

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

```
QOCFloat * y,
QOCInt n )
```

Copies array of QOCFloats 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 QOCInt * x,
    QOCInt * y,
    QOCInt n )
```

Copies array of QOCInts 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

<i>A</i>	
<i>pinv</i>	
<i>AtoC</i>	

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

<i>u</i>	Input vector.
<i>v</i>	Input vector.
<i>n</i>	Length of vectors.

Returns

Dot product of u and v.

5.7.4.11 ew_product()

```
void ew_product (
    QOCFloat * x,
    const QOCFloat * y,
    QOCFloat * z,
    QOCInt 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()

```
QOCFloat inf_norm (
    const QOCFloat * x,
    QOCInt 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

<i>A</i>	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

<i>M</i>	Matrix to be regularized.
<i>lambda</i>	Regularization factor.
<i>nzadded_idx</i>	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. $M = \text{diag}(E) * M * \text{diag}(D)$

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,
    QOCFloat * 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 QOCFloat * x,
    QOCFloat * y,
    QOCFloat s,
    QOCInt 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

<i>M</i>	Matrix in CSC form.
<i>v</i>	Vector.
<i>r</i>	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

<i>M</i>	Matrix in CSC form.
<i>v</i>	Vector.
<i>r</i>	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

<i>M</i>	Matrix.
<i>lambda</i>	Regularization.

5.7.4.24 USpMv()

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

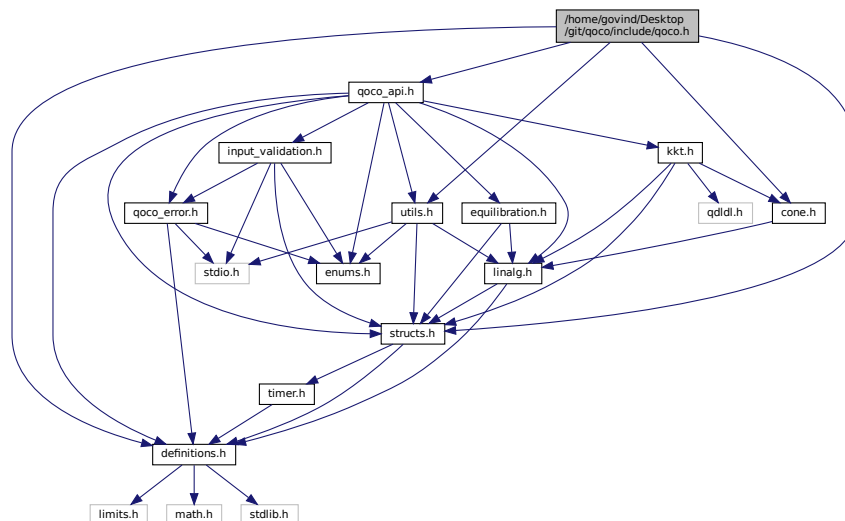
Sparse matrix vector multiplication for CSC matrices where M is symmetric and only the upper triangular part is given. Computes $r = M * v$.

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

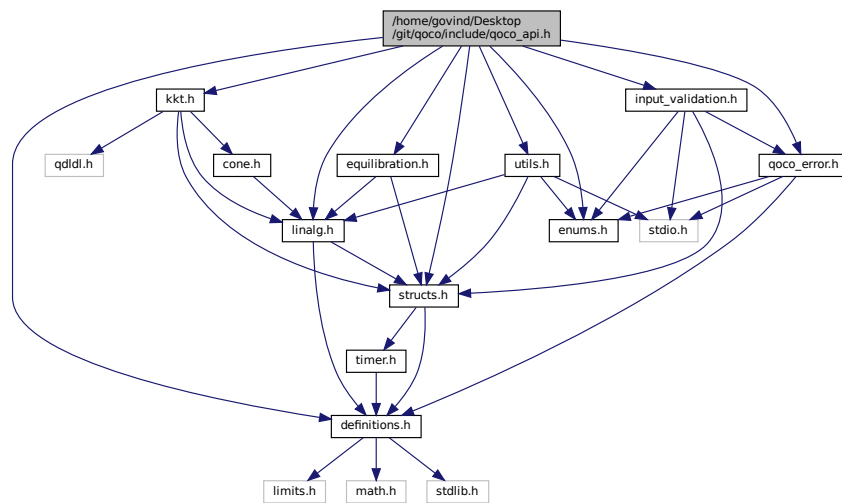
5.8.3 DESCRIPTION

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

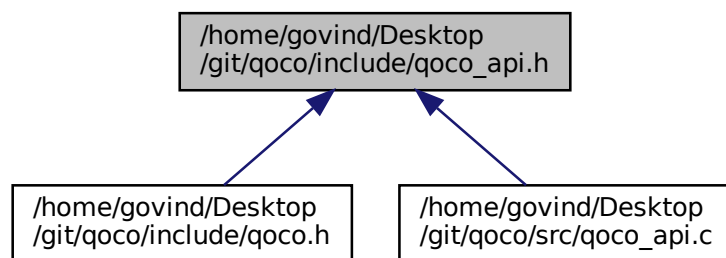
5.9 /home/govind/Desktop/git/qoco/include/qoco_api.h File Reference

```
#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

<code>solver</code>	Pointer to solver.
---------------------	--------------------

Returns

Exitflag to check (0 for success, failure otherwise)

5.9.4.2 qoco_set_csc()

```
void qoco_set_csc (
    QOCOCscMatrix *  $\bar{A}$ ,
    QOCOInt  $m$ ,
    QOCOInt  $n$ ,
    QOCOInt  $Annz$ ,
    QOCOFloat *  $Ax$ ,
    QOCOInt *  $\bar{A}p$ ,
    QOCOInt *  $\bar{A}i$  )
```

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.
$\bar{A}p$	Array of column pointers for the data.
$\bar{A}i$	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

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

<i>solver</i>	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

<i>solver</i>	Pointer to solver.
<i>new_settings</i>	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

<i>settings</i>	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

<i>solver</i>	Pointer to solver.
<i>Pxnew</i>	New data for P->x.
<i>Axnew</i>	New data for A->x.
<i>Gxnew</i>	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

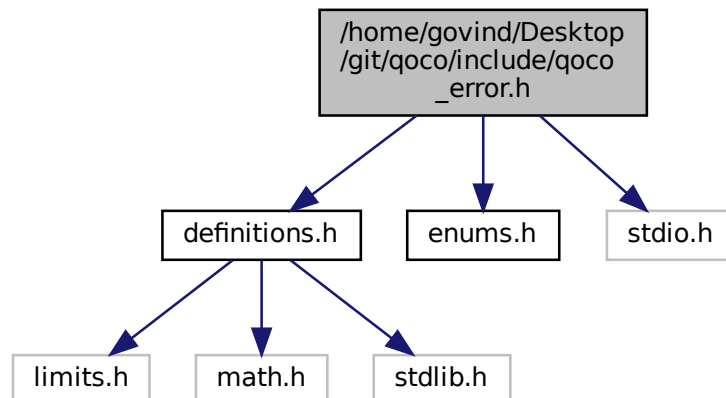
<i>solver</i>	Pointer to solver.
<i>cnew</i>	New c vector.
<i>bnew</i>	New b vector.
<i>hnew</i>	New h vector.

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

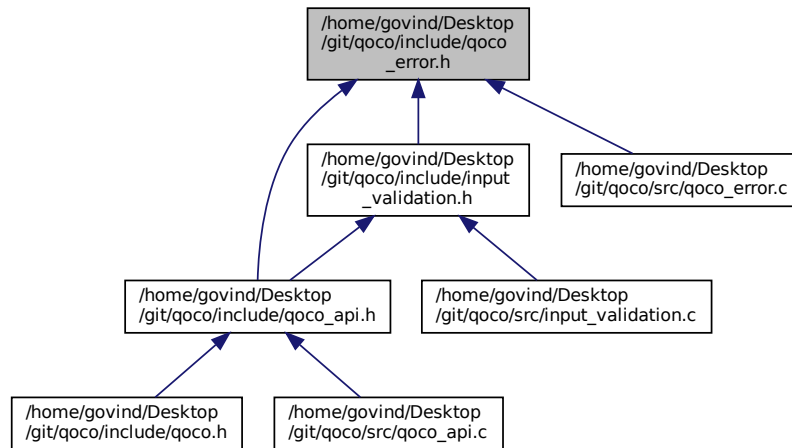
```

#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

<code>error_code</code>	
-------------------------	--

Returns

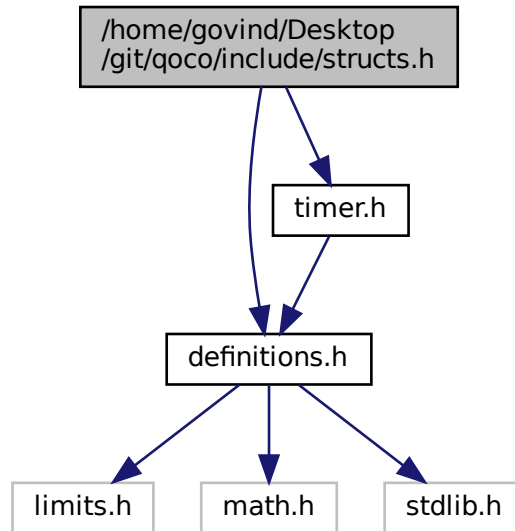
Error code as an QOCOInt.

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

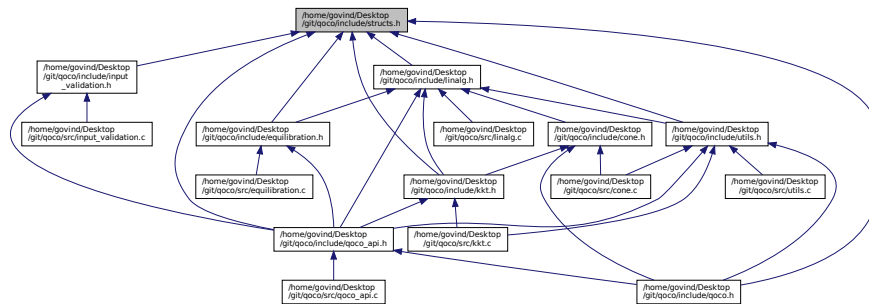
```
#include "definitions.h"
```

```
#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 [QOCOPProblemData](#)
SOC 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

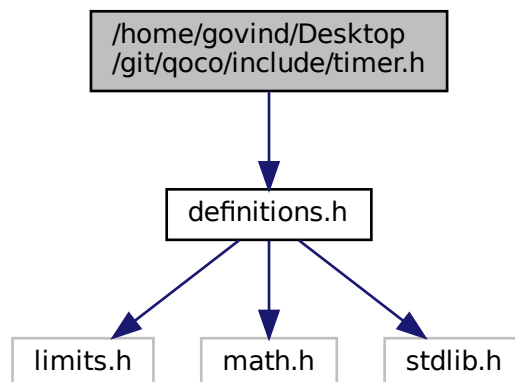
5.11.3 DESCRIPTION

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:



Parameters

<i>timer</i>	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

<i>timer</i>	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.

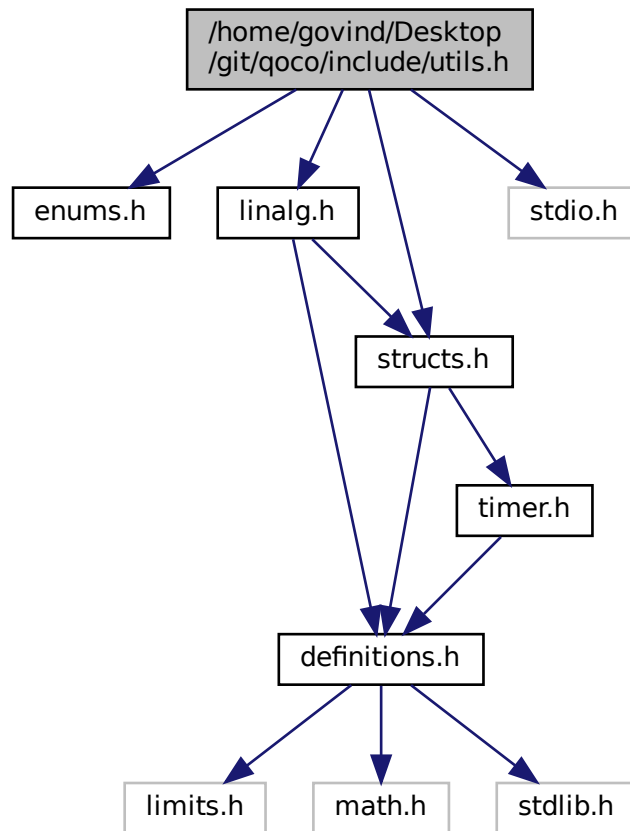
Parameters

<i>timer</i>	Pointer to timer struct.
--------------	--------------------------

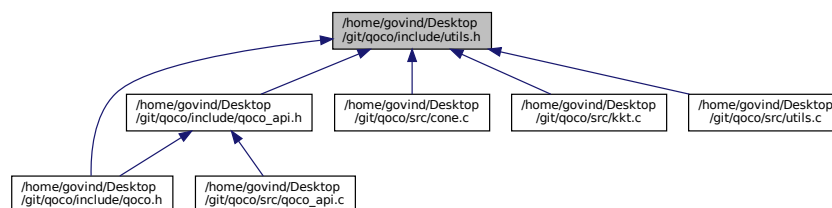
5.13 /home/govind/Desktop/git/qoco/include/utils.h File Reference

```
#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

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

<code>solver</code>	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

<i>settings</i>	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

<i>solver</i>	Pointer to solver.
---------------	--------------------

5.13.4.4 log_iter()

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

Print solver progress.

Parameters

<i>solver</i>	Pointer to solver.
---------------	--------------------

5.13.4.5 print_arrayf()

```
void print_arrayf (
    QOCFloat * x,
    QOCInt n )
```

Prints array of QOCFloats.

Parameters

<i>x</i>	Pointer to array.
<i>n</i>	Number of elements in array.

5.13.4.6 print_arrayi()

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

Prints array of QOCInts.

Parameters

<i>x</i>	Pointer to array.
<i>n</i>	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

<i>solution</i>	Pointer to solution struct.
<i>status</i>	Solve status.

5.13.4.8 print_header()

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

Prints QOCO header.

Parameters

<i>solver</i>	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.

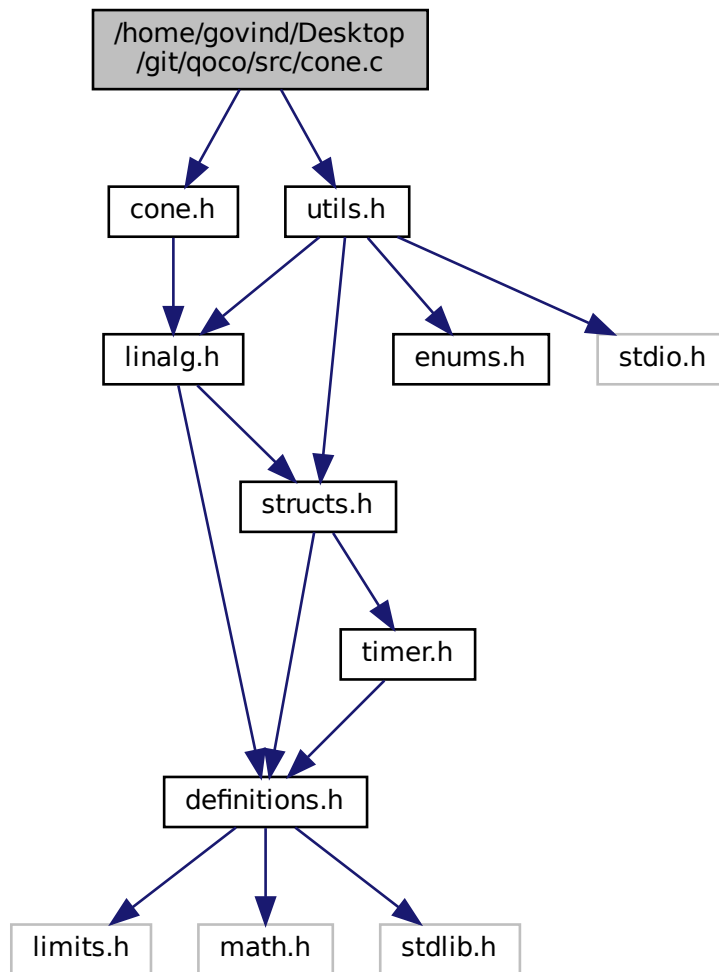
Parameters

<i>M</i>	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)
Computes 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 $u_0^2 - u_1 * u_1$ 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 canonical vector for each cone LP Cone: $e = \text{ones}(n)$, second-order cone: $e = (1, 0, 0, \dots)$ 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 λ 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 λ 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 λ 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

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

<i>u</i>	Initial vector.
<i>Du</i>	Search direction.
<i>f</i>	Conservatism factor.
<i>solver</i>	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 canonical vector for each cone LP Cone: $e = \text{ones}(n)$, second-order cone: $e = (1, 0, 0, \dots)$ and a is the minimum scalar value such that $u + (1 + a) * e$ is in cone C .

Parameters

<i>u</i>	Vector to bring to cone.
<i>data</i>	Pointer to problem data.

5.14.3.3 compute_centering()

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

Computes centering parameter.

Parameters

<i>solver</i>	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

<i>work</i>	Pointer to workspace.
-------------	-----------------------

5.14.3.5 compute_nt_scaling()

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

Compute Nesterov-Todd scalings and scaled variables.

Parameters

<i>work</i>	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

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

<i>u</i>	Initial vector.
<i>Du</i>	Search direction.
<i>f</i>	Conservatism factor.
<i>solver</i>	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 λ in $(0, 1]$ such that $u + (\lambda / f) * Du \in C$. For QPs this calls [exact_linesearch\(\)](#) and for SOCPs this calls [bisection_search\(\)](#)

Parameters

<i>u</i>	Initial vector.
<i>Du</i>	Search direction.
<i>f</i>	Conservatism factor.
<i>solver</i>	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 )
```

Computes second-order cone division $\lambda v = d$.

Parameters

λ	$\lambda = (\lambda_0, \lambda_1)$ is a vector in second-order cone of dimension n .
v	$v = (v_0, v_1)$ is a vector in second-order cone of dimension n .
d	Cone division of λ 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	$u = (u_0, u_1)$ is a vector in second-order cone of dimension n .
v	$v = (v_0, v_1)$ 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	$u = (u_0, u_1)$ is a vector in second-order cone of dimension n .
n	Dimension of second order cone.

Returns

Residual: $\text{norm}(u_1) - u_0$. 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 $u_0^2 - u_1' * u_1$ of vector u with respect to the second order cone of dimension n .

Parameters

u	$u = (u_0, u_1)$ is a vector in second order cone of dimension n .
n	Dimension of second order cone.

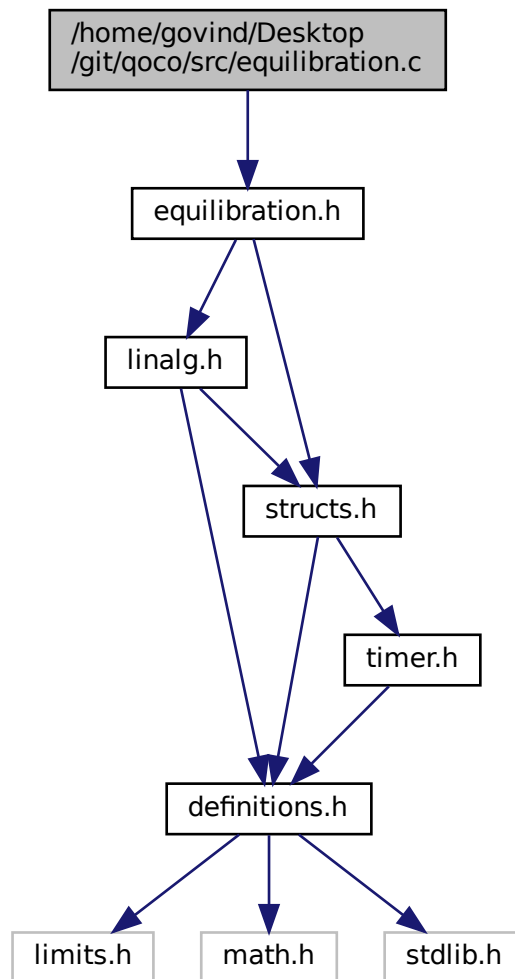
Returns

Residual: $u_0^2 - u_1' * u_1$.

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

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

Include dependency graph for equilibration.c:



Functions

- void `ruiz_equlibration` (`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

$$[D][kPA^TG^T][D]|E||A00||E|[F][G00][F]$$

clang-format on

Parameters

<i>solver</i>	Pointer to solver.
---------------	--------------------

5.15.1.2 unscale_variables()

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

Undo variable transformation induced by ruiz equilibration.

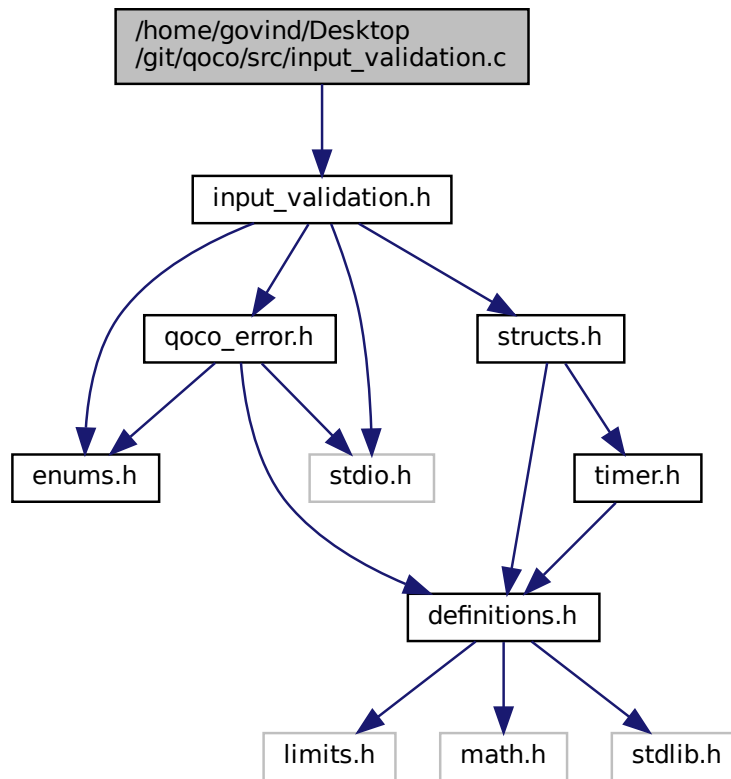
Parameters

<i>work</i>	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

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

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

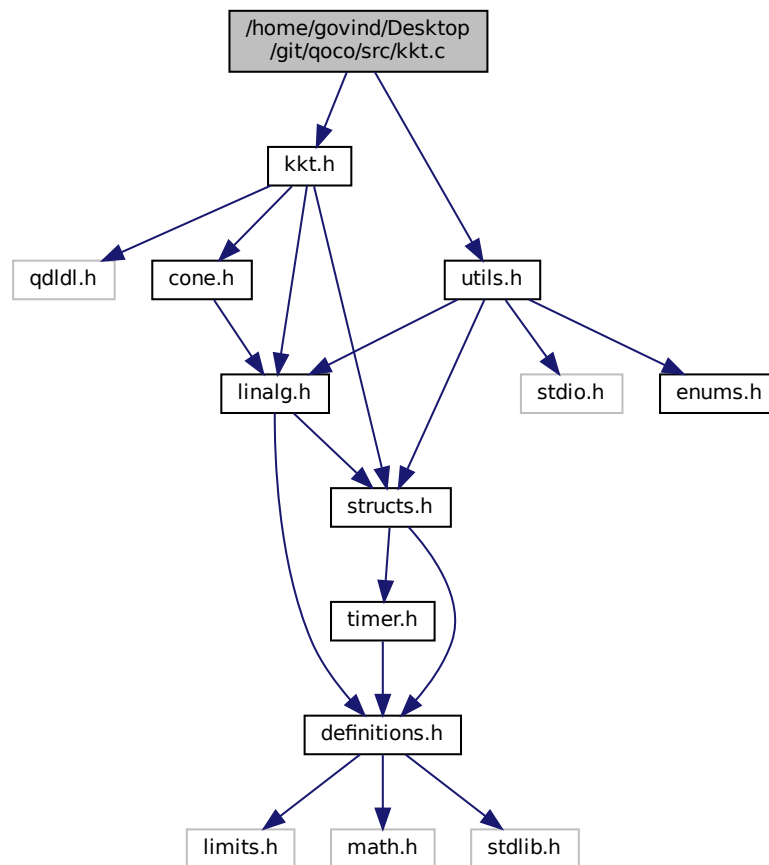
<i>settings</i>	Pointer to settings struct
-----------------	----------------------------

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 Nesterov-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

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

5.17.3 Function Documentation

5.17.3.1 `allocate_kkt()`

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

Allocate memory for KKT matrix.

Parameters

<code>work</code>	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   ]
```

$$\text{res} = \begin{bmatrix} A & 0 & 0 \end{bmatrix} \begin{bmatrix} y \end{bmatrix} + \begin{bmatrix} -b \end{bmatrix} \begin{bmatrix} G & 0 & 0 \end{bmatrix} \begin{bmatrix} z \end{bmatrix} \begin{bmatrix} -h + s \end{bmatrix}$$

clang-format on

Parameters

<i>solver</i>	Pointer to solver.
---------------	--------------------

5.17.3.3 construct_kkt()

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

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

clang-format off

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

$$K = \begin{bmatrix} A & 0 & 0 \end{bmatrix} \begin{bmatrix} G & 0 & -I \end{bmatrix}$$

clang-format on

Parameters

<i>solver</i>	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

<i>work</i>	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

<i>work</i>	Pointer to workspace.
-------------	-----------------------

$ds = -\text{cone_product}(\lambda, \lambda) - \text{settings.mehrotra} * \text{cone_product}((W' \setminus D_{\text{aff}}), (W * D_{\text{aff}}), p_{\text{data}}) + \sigma * \mu * e.$

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

<i>solver</i>	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

<i>solver</i>	Pointer to solver.
<i>x</i>	Pointer to input vector.
<i>y</i>	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

<i>solver</i>	Pointer to solver.
<i>b</i>	Pointer to rhs of kkt system.
<i>iters</i>	Number of iterations of iterative refinement performed.

5.17.3.9 predictor_corrector()

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

Performs Mehrotra predictor-corrector step.

Parameters

<i>solver</i>	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

<i>work</i>	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]

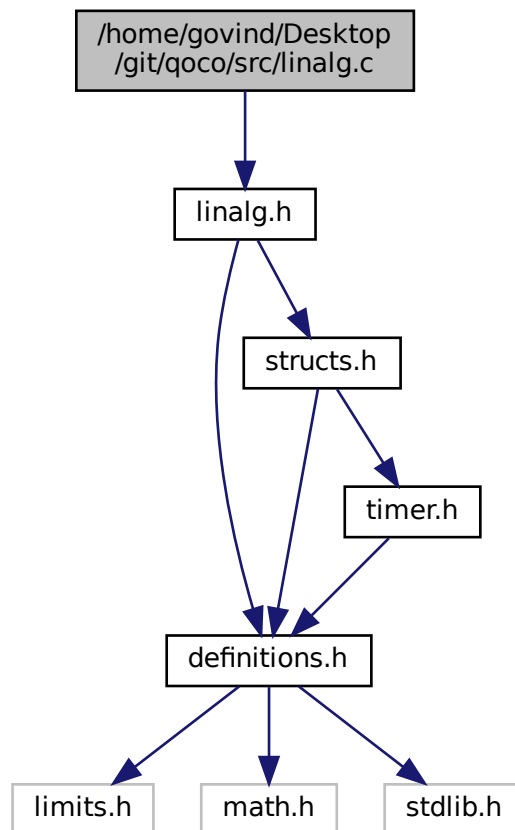
$K = \begin{bmatrix} A & 0 & 0 \\ 0 & -W & W - e * I \end{bmatrix}$

Parameters

<i>solver</i>	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, QOCFloat 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 QOCFloat *x, QOCFloat *y, QOCOInt n)
Copies array of QOCFloats from x to array y.
- void [copy_and_negate_arrayf](#) (const QOCFloat *x, QOCFloat *y, QOCOInt n)
Copies and negates array of QOCFloats from x to array y.
- void [copy_arrayi](#) (const QOCOInt *x, QOCOInt *y, QOCOInt n)
Copies array of QOCOInts from x to array y.
- [QOCFloat dot](#) (const [QOCFloat](#) *u, const [QOCFloat](#) *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 [QOCFloat](#) *x, [QOCFloat](#) *y, [QOCFloat](#) s, QOCOInt n)
*Scales array x by s and stores result in y. $y = s * x$.*
- void [axpy](#) (const [QOCFloat](#) *x, const [QOCFloat](#) *y, [QOCFloat](#) *z, [QOCFloat](#) a, QOCOInt n)
*Computes $z = a * x + y$.*
- void [USpMv](#) (const [QOCOCscMatrix](#) *M, const [QOCFloat](#) *v, [QOCFloat](#) *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 [QOCFloat](#) *v, [QOCFloat](#) *r)
*Sparse matrix vector multiplication for CSC matrices. Computes $r = M * v$.*
- void [SpMtv](#) (const [QOCOCscMatrix](#) *M, const [QOCFloat](#) *v, [QOCFloat](#) *r)
*Sparse matrix vector multiplication for CSC matrices where M is first transposed. Computes $r = M^T * v$.*
- [QOCFloat inf_norm](#) (const [QOCFloat](#) *x, QOCOInt n)
Computes the infinity norm of x.
- [QOCOInt regularize](#) ([QOCOCscMatrix](#) *M, [QOCFloat](#) 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, [QOCFloat](#) 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, [QOCFloat](#) *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, [QOCFloat](#) *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, [QOCFloat](#) *E, [QOCFloat](#) *D)
*Scales the rows of M by E and columns of M by D. $M = \text{diag}(E) * M * \text{diag}(D)$*
- void [ew_product](#) ([QOCFloat](#) *x, const [QOCFloat](#) *y, [QOCFloat](#) *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

Copyright (c) 2024, Govind M. Chari This source code is licensed under the BSD 3-Clause 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 $\lambda \cdot I$.

Parameters

<i>n</i>	Size of identity matrix.
<i>lambda</i>	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

<i>x</i>	Source array.
<i>y</i>	Destination array.
<i>n</i>	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

<i>A</i>	
<i>pinv</i>	
<i>AtoC</i>	

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

<i>u</i>	Input vector.
<i>v</i>	Input vector.
<i>n</i>	Length of vectors.

Returns

Dot product of u and v.

5.18.3.11 ew_product()

```
void ew_product (
    QOCFloat * x,
    const QOCFloat * y,
    QOCFloat * z,
    QOCInt 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()

```
QOCFloat inf_norm (
    const QOCFloat * x,
    QOCInt 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

<i>p</i>	Input permutation vector.
<i>pinv</i>	Inverse of permutation vector.
<i>n</i>	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

<i>x</i>	Input array.
<i>n</i>	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

<i>A</i>	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

<i>M</i>	Matrix to be regularized.
<i>lambda</i>	Regularization factor.
<i>nzadded_idx</i>	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. $M = \text{diag}(E) * M * \text{diag}(S)$

Parameters

<i>M</i>	An m by n sparse matrix.
<i>E</i>	Vector of length m.
<i>D</i>	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

<i>M</i>	Matrix in CSC form.
<i>v</i>	Vector.
<i>r</i>	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

<i>M</i>	Matrix.
<i>lambda</i>	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

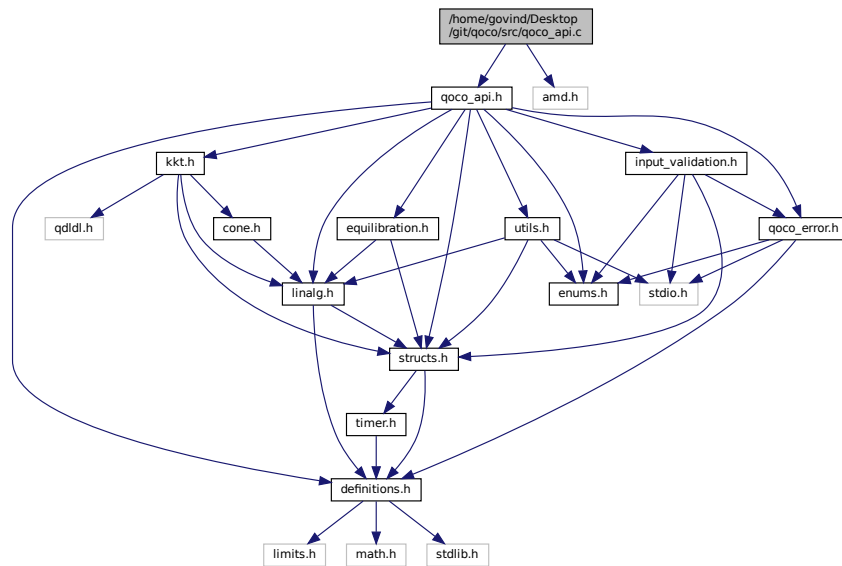
<i>M</i>	Upper triangular part of M in CSC form.
<i>v</i>	Vector.
<i>r</i>	Result.

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

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

<i>solver</i>	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

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

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

<i>solver</i>	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

<i>solver</i>	Pointer to solver.
<i>new_settings</i>	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

<i>settings</i>	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

<i>solver</i>	Pointer to solver.
<i>Pxnew</i>	New data for P->x.
<i>Axnew</i>	New data for A->x.
<i>Gxnew</i>	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.

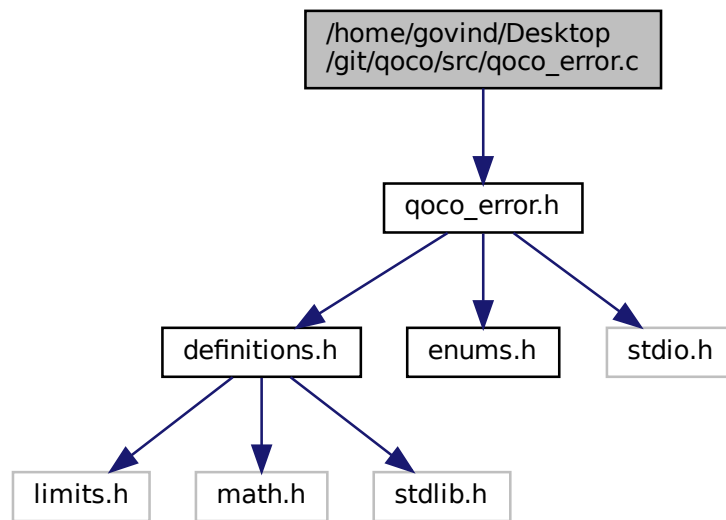
Parameters

<i>solver</i>	Pointer to solver.
<i>cnew</i>	New c vector.
<i>bnew</i>	New b vector.
<i>hnew</i>	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

<code>error_code</code>	
-------------------------	--

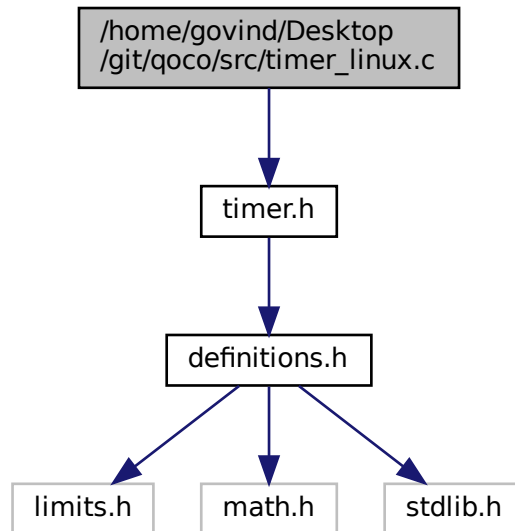
Returns

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.
- QOCFloat [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()

```
QOCFloat get_elapsed_time_sec (
    QOCOTimer * timer )
```

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

Parameters

<i>timer</i>	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

<i>timer</i>	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.

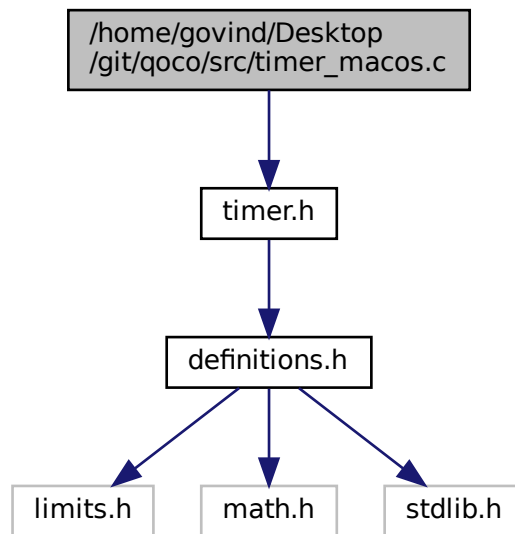
Parameters

<i>timer</i>	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.
- QOCFloat [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()

```
QOCFloat get_elapsed_time_sec (
    QOCOTimer * timer )
```

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

Parameters

<i>timer</i>	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

<i>timer</i>	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.

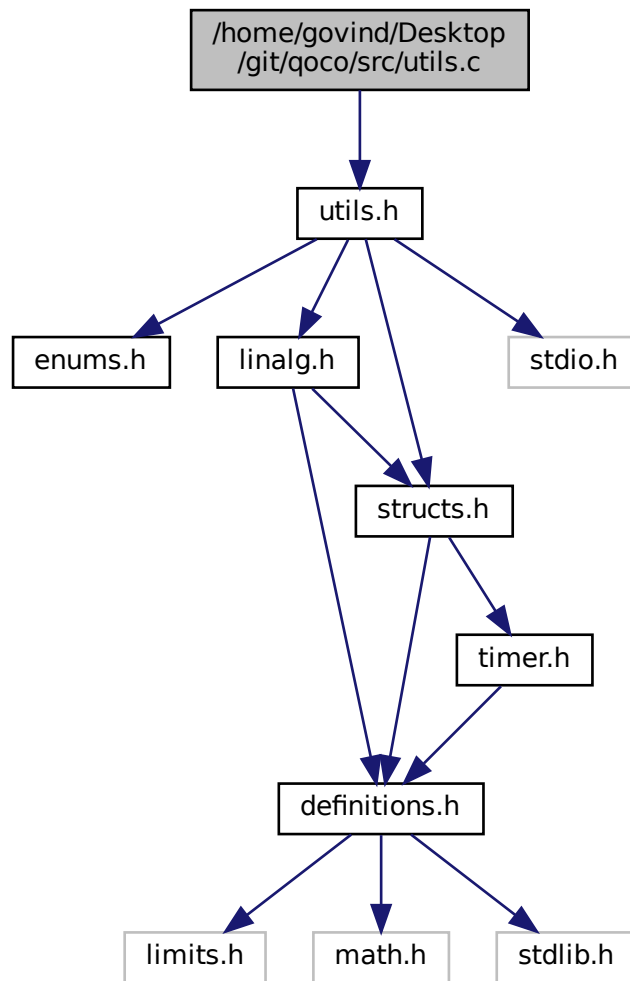
Parameters

<i>timer</i>	Pointer to timer struct.
--------------	--------------------------

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

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

Include dependency graph for utlis.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` (`QOCFloat *x`, `QOCOInt n`)
Prints array of `QOCFloats`.
- 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

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

<code>solver</code>	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

<i>settings</i>	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

<i>solver</i>	Pointer to solver.
---------------	--------------------

5.23.3.4 log_iter()

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

Print solver progress.

Parameters

<i>solver</i>	Pointer to solver.
---------------	--------------------

5.23.3.5 print_arrayf()

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

Prints array of QOCFloats.

Parameters

<i>x</i>	Pointer to array.
<i>n</i>	Number of elements in array.

5.23.3.6 print_arrayi()

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

Prints array of QOCOInts.

Parameters

<i>x</i>	Pointer to array.
<i>n</i>	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

<i>solution</i>	Pointer to solution struct.
<i>status</i>	Solve status.

5.23.3.8 print_header()

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

Prints QOCO header.

Parameters

<i>solver</i>	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

<i>M</i>	Pointer to QOCOCscMatrix that will be printed.
----------	--

Index

- [/home/govind/Desktop/git/qoco/include/cone.h](#), 29
- [/home/govind/Desktop/git/qoco/include/definitions.h](#), 37
- [/home/govind/Desktop/git/qoco/include/enums.h](#), 41
- [/home/govind/Desktop/git/qoco/include/equilibration.h](#), 42
- [/home/govind/Desktop/git/qoco/include/input_validation.h](#), 45
- [/home/govind/Desktop/git/qoco/include/kkt.h](#), 48
- [/home/govind/Desktop/git/qoco/include/linalg.h](#), 53
- [/home/govind/Desktop/git/qoco/include/qoco.h](#), 66
- [/home/govind/Desktop/git/qoco/include/qoco_api.h](#), 67
- [/home/govind/Desktop/git/qoco/include/qoco_error.h](#), 72
- [/home/govind/Desktop/git/qoco/include/structs.h](#), 73
- [/home/govind/Desktop/git/qoco/include/timer.h](#), 75
- [/home/govind/Desktop/git/qoco/include/utils.h](#), 77
- [/home/govind/Desktop/git/qoco/src/cone.c](#), 82
- [/home/govind/Desktop/git/qoco/src/equilibration.c](#), 90
- [/home/govind/Desktop/git/qoco/src/input_validation.c](#), 92
- [/home/govind/Desktop/git/qoco/src/kkt.c](#), 95
- [/home/govind/Desktop/git/qoco/src/linalg.c](#), 100
- [/home/govind/Desktop/git/qoco/src/qoco_api.c](#), 110
- [/home/govind/Desktop/git/qoco/src/qoco_error.c](#), 115
- [/home/govind/Desktop/git/qoco/src/timer_linux.c](#), 117
- [/home/govind/Desktop/git/qoco/src/timer_macos.c](#), 118
- [/home/govind/Desktop/git/qoco/src/utils.c](#), 120
- A
 - [QOCOPProblemData](#), 15
- a
 - [QOCOWorkspace](#), 24
- abstol
 - [QOCOSettings](#), 17
- abstol_inacc
 - [QOCOSettings](#), 17
- allocate_kkt
 - [kkt.c](#), 96
 - [kkt.h](#), 50
- At
 - [QOCOPProblemData](#), 15
- AtoKKT
 - [QOCOKKT](#), 9
- axpy
 - [linalg.c](#), 102
 - [linalg.h](#), 56
- b
 - [QOCOPProblemData](#), 15
- bisect_iters
 - [QOCOSettings](#), 18
- bisection_search
 - [cone.c](#), 84
 - [cone.h](#), 31
- bring2cone
 - [cone.c](#), 85
 - [cone.h](#), 32
- bwork
 - [QOCOKKT](#), 9
- c
 - [QOCOPProblemData](#), 15
- check_stopping
 - [utils.c](#), 122
 - [utils.h](#), 79
- col_inf_norm_USymm
 - [linalg.c](#), 102
 - [linalg.h](#), 56
- compute_centering
 - [cone.c](#), 85
 - [cone.h](#), 32
- compute_kkt_residual
 - [kkt.c](#), 96
 - [kkt.h](#), 50
- compute_mu
 - [cone.c](#), 85
 - [cone.h](#), 32
- compute_nt_scaling
 - [cone.c](#), 86
 - [cone.h](#), 32
- cone.c
 - [bisection_search](#), 84
 - [bring2cone](#), 85
 - [compute_centering](#), 85
 - [compute_mu](#), 85
 - [compute_nt_scaling](#), 86
 - [cone_division](#), 86
 - [cone_product](#), 86
 - [cone_residual](#), 87
 - [exact_linesearch](#), 87
 - [linesearch](#), 88
 - [nt_multiply](#), 88
 - [soc_division](#), 89
 - [soc_product](#), 89
 - [soc_residual](#), 89
 - [soc_residual2](#), 90
- cone.h
 - [bisection_search](#), 31
 - [bring2cone](#), 32
 - [compute_centering](#), 32
 - [compute_mu](#), 32

- compute_nt_scaling, 32
- cone_division, 33
- cone_product, 33
- cone_residual, 34
- exact_linesearch, 34
- linesearch, 35
- nt_multiply, 35
- soc_division, 36
- soc_product, 36
- soc_residual, 36
- soc_residual2, 37
- cone_division
 - cone.c, 86
 - cone.h, 33
- cone_product
 - cone.c, 86
 - cone.h, 33
- cone_residual
 - cone.c, 87
 - cone.h, 34
- construct_identity
 - linalg.c, 103
 - linalg.h, 57
- construct_kkt
 - kkt.c, 97
 - kkt.h, 50
- construct_kkt_aff_rhs
 - kkt.c, 97
 - kkt.h, 51
- construct_kkt_comb_rhs
 - kkt.c, 98
 - kkt.h, 51
- copy_and_negate_arrayf
 - linalg.c, 103
 - linalg.h, 57
- copy_arrayf
 - linalg.c, 103
 - linalg.h, 57
- copy_arrayi
 - linalg.c, 104
 - linalg.h, 58
- copy_settings
 - utils.c, 122
 - utils.h, 80
- copy_solution
 - utils.c, 123
 - utils.h, 80
- create_transposed_matrix
 - linalg.c, 104
 - linalg.h, 58
- csc_symperm
 - linalg.c, 104
 - linalg.h, 58
- cumsum
 - linalg.c, 105
 - linalg.h, 59
- D
 - QOCOKKT, 10
- data
 - QOCOWorkspace, 24
- definitions.h
 - qoco_abs, 39
 - qoco_assert, 39
 - qoco_calloc, 39
 - qoco_free, 39
 - qoco_malloc, 39
 - qoco_max, 40
 - qoco_min, 40
 - qoco_sqrt, 40
 - QOCFloat, 41
 - QOCFloat_MAX, 40
 - QOCInt, 41
 - QOCInt_MAX, 40
 - safe_div, 40
- delta
 - QOCOKKT, 10
- Dinv
 - QOCOKKT, 10
- Dinvruiz
 - QOCOKKT, 10
- dot
 - linalg.c, 105
 - linalg.h, 59
- dres
 - QOCOSolution, 19
- Druiz
 - QOCOKKT, 10
- Ds
 - QOCOWorkspace, 24
- Einvrui
 - QOCOKKT, 10
- enums.h
 - QOCO_AMD_ERROR, 42
 - QOCO_DATA_VALIDATION_ERROR, 42
 - qoco_error_code, 41
 - QOCO_MALLOC_ERROR, 42
 - QOCO_MAX_ITER, 42
 - QOCO_NO_ERROR, 42
 - QOCO_NUMERICAL_ERROR, 42
 - QOCO_SETTINGS_VALIDATION_ERROR, 42
 - QOCO_SETUP_ERROR, 42
 - qoco_solve_status, 42
 - QOCO_SOLVED, 42
 - QOCO_SOLVED_INACCURATE, 42
 - QOCO_UNSOLVED, 42
- equilibration.c
 - ruiz_equilibration, 91
 - unscale_variables, 92
- equilibration.h
 - ruiz_equilibration, 44
 - unscale_variables, 44
- Eruiz
 - QOCOKKT, 10
- etree
 - QOCOKKT, 10
- ew_product

- linalg.c, [105](#)
 - linalg.h, [60](#)
- exact_linesearch
 - cone.c, [87](#)
 - cone.h, [34](#)
- Finvruiz
 - QOCOKKT, [11](#)
- free_qoco_csc_matrix
 - linalg.c, [106](#)
 - linalg.h, [60](#)
- Fruiz
 - QOCOKKT, [11](#)
- fwork
 - QOCOKKT, [11](#)
- G
 - QOCOPProblemData, [15](#)
- gap
 - QOCOSolution, [19](#)
- get_elapsed_time_sec
 - timer.h, [76](#)
 - timer_linux.c, [117](#)
 - timer_macos.c, [119](#)
- Gt
 - QOCOPProblemData, [15](#)
- GtoKKT
 - QOCOKKT, [11](#)
- h
 - QOCOPProblemData, [15](#)
- i
 - QOCOCscMatrix, [7](#)
- inf_norm
 - linalg.c, [106](#)
 - linalg.h, [60](#)
- initialize_ipm
 - kkt.c, [98](#)
 - kkt.h, [51](#)
- input_validation.c
 - qoco_validate_data, [94](#)
 - qoco_validate_settings, [94](#)
- input_validation.h
 - qoco_validate_data, [46](#)
 - qoco_validate_settings, [47](#)
- invert_permutation
 - linalg.c, [107](#)
 - linalg.h, [61](#)
- iter_ref_iters
 - QOCOSettings, [18](#)
- iters
 - QOCOSolution, [20](#)
- iwork
 - QOCOKKT, [11](#)
- K
 - QOCOKKT, [11](#)
- k
 - QOCOKKT, [11](#)
- kinv
 - QOCOKKT, [11](#)
- kkt
 - QOCOWorkspace, [24](#)
- kkt.c
 - allocate_kkt, [96](#)
 - compute_kkt_residual, [96](#)
 - construct_kkt, [97](#)
 - construct_kkt_aff_rhs, [97](#)
 - construct_kkt_comb_rhs, [98](#)
 - initialize_ipm, [98](#)
 - kkt_multiply, [98](#)
 - kkt_solve, [98](#)
 - predictor_corrector, [99](#)
 - set_nt_block_zeros, [99](#)
 - update_nt_block, [99](#)
- kkt.h
 - allocate_kkt, [50](#)
 - compute_kkt_residual, [50](#)
 - construct_kkt, [50](#)
 - construct_kkt_aff_rhs, [51](#)
 - construct_kkt_comb_rhs, [51](#)
 - initialize_ipm, [51](#)
 - kkt_multiply, [52](#)
 - kkt_solve, [52](#)
 - predictor_corrector, [52](#)
 - set_nt_block_zeros, [53](#)
 - update_nt_block, [53](#)
- kkt_dynamic_reg
 - QOCOSettings, [18](#)
- kkt_multiply
 - kkt.c, [98](#)
 - kkt.h, [52](#)
- kkt_solve
 - kkt.c, [98](#)
 - kkt.h, [52](#)
- kkt_static_reg
 - QOCOSettings, [18](#)
- kktres
 - QOCOKKT, [12](#)
- l
 - QOCOPProblemData, [16](#)
- lambda
 - QOCOWorkspace, [24](#)
- Li
 - QOCOKKT, [12](#)
- linalg.c
 - axpy, [102](#)
 - col_inf_norm_USymm, [102](#)
 - construct_identity, [103](#)
 - copy_and_negate_arrayf, [103](#)
 - copy_arrayf, [103](#)
 - copy_arrayi, [104](#)
 - create_transposed_matrix, [104](#)
 - csc_symperm, [104](#)
 - cumsum, [105](#)
 - dot, [105](#)

- ew_product, 105
- free_qoco_csc_matrix, 106
- inf_norm, 106
- invert_permutation, 107
- max_arrayi, 107
- new_qoco_csc_matrix, 107
- regularize, 108
- row_col_scale, 108
- row_inf_norm, 108
- scale_arrayf, 109
- SpMtv, 109
- SpMv, 109
- unregularize, 110
- USpMv, 110
- linalg.h
 - axpy, 56
 - col_inf_norm_USymm, 56
 - construct_identity, 57
 - copy_and_negate_arrayf, 57
 - copy_arrayf, 57
 - copy_arrayi, 58
 - create_transposed_matrix, 58
 - csc_symperm, 58
 - cumsum, 59
 - dot, 59
 - ew_product, 60
 - free_qoco_csc_matrix, 60
 - inf_norm, 60
 - invert_permutation, 61
 - max_arrayi, 61
 - new_qoco_csc_matrix, 61
 - regularize, 62
 - row_col_scale, 62
 - row_inf_norm, 63
 - scale_arrayf, 63
 - SpMtv, 63
 - SpMv, 65
 - unregularize, 65
 - USpMv, 65
- linesearch
 - cone.c, 88
 - cone.h, 35
- Lnz
 - QOCOKKT, 12
- log_iter
 - utils.c, 123
 - utils.h, 80
- Lp
 - QOCOKKT, 12
- Lx
 - QOCOKKT, 12
- m
 - QOCOCscMatrix, 7
 - QOCOPProblemData, 16
- max_arrayi
 - linalg.c, 107
 - linalg.h, 61
- max_iters
 - QOCOSettings, 18
- mu
 - QOCOWorkspace, 24
- n
 - QOCOCscMatrix, 8
 - QOCOPProblemData, 16
- new_qoco_csc_matrix
 - linalg.c, 107
 - linalg.h, 61
- nnz
 - QOCOCscMatrix, 8
- nsoc
 - QOCOPProblemData, 16
- nt2kkt
 - QOCOKKT, 12
- nt_multiply
 - cone.c, 88
 - cone.h, 35
- ntdiag2kkt
 - QOCOKKT, 12
- obj
 - QOCOSolution, 20
- P
 - QOCOPProblemData, 16
- p
 - QOCOCscMatrix, 8
 - QOCOKKT, 12
 - QOCOPProblemData, 16
- pinv
 - QOCOKKT, 13
- Pnum_nzadded
 - QOCOKKT, 13
- Pnzadded_idx
 - QOCOKKT, 13
- predictor_corrector
 - kkt.c, 99
 - kkt.h, 52
- PregtoKKT
 - QOCOKKT, 13
- pres
 - QOCOSolution, 20
- print_arrayf
 - utils.c, 123
 - utils.h, 80
- print_arrayi
 - utils.c, 124
 - utils.h, 81
- print_footer
 - utils.c, 124
 - utils.h, 81
- print_header
 - utils.c, 124
 - utils.h, 81
- print_qoco_csc_matrix
 - utils.c, 124
 - utils.h, 82

q

- QOCOPProblemData, 16
- qoco_abs
 - definitions.h, 39
- QOCO_AMD_ERROR
 - enums.h, 42
- qoco_api.c
 - qoco_cleanup, 112
 - qoco_set_csc, 112
 - qoco_setup, 113
 - qoco_solve, 113
 - qoco_update_settings, 114
 - set_default_settings, 114
 - update_matrix_data, 114
 - update_vector_data, 115
- qoco_api.h
 - qoco_cleanup, 68
 - qoco_set_csc, 69
 - qoco_setup, 69
 - qoco_solve, 70
 - qoco_update_settings, 70
 - set_default_settings, 71
 - update_matrix_data, 71
 - update_vector_data, 71
- qoco_assert
 - definitions.h, 39
- qoco_calloc
 - definitions.h, 39
- qoco_cleanup
 - qoco_api.c, 112
 - qoco_api.h, 68
- QOCO_DATA_VALIDATION_ERROR
 - enums.h, 42
- qoco_error
 - qoco_error.c, 116
 - qoco_error.h, 73
- qoco_error.c
 - qoco_error, 116
- qoco_error.h
 - qoco_error, 73
- qoco_error_code
 - enums.h, 41
- qoco_free
 - definitions.h, 39
- qoco_malloc
 - definitions.h, 39
- QOCO_MALLOC_ERROR
 - enums.h, 42
- qoco_max
 - definitions.h, 40
- QOCO_MAX_ITER
 - enums.h, 42
- qoco_min
 - definitions.h, 40
- QOCO_NO_ERROR
 - enums.h, 42
- QOCO_NUMERICAL_ERROR
 - enums.h, 42
- qoco_set_csc
 - qoco_api.c, 112
 - qoco_api.h, 69
- QOCO_SETTINGS_VALIDATION_ERROR
 - enums.h, 42
- qoco_setup
 - qoco_api.c, 113
 - qoco_api.h, 69
- QOCO_SETUP_ERROR
 - enums.h, 42
- qoco_solve
 - qoco_api.c, 113
 - qoco_api.h, 70
- qoco_solve_status
 - enums.h, 42
- QOCO_SOLVED
 - enums.h, 42
- QOCO_SOLVED_INACCURATE
 - enums.h, 42
- qoco_sqrt
 - definitions.h, 40
- QOCO_UNSOLVED
 - enums.h, 42
- qoco_update_settings
 - qoco_api.c, 114
 - qoco_api.h, 70
- qoco_validate_data
 - input_validation.c, 94
 - input_validation.h, 46
- qoco_validate_settings
 - input_validation.c, 94
 - input_validation.h, 47
- QOCOCscMatrix, 7
 - i, 7
 - m, 7
 - n, 8
 - nnz, 8
 - p, 8
 - x, 8
- QOCOFloat
 - definitions.h, 41
- QOCOFloat_MAX
 - definitions.h, 40
- QOCOInt
 - definitions.h, 41
- QOCOInt_MAX
 - definitions.h, 40
- QOCOKKT, 8
 - AtoKKT, 9
 - bwork, 9
 - D, 10
 - delta, 10
 - Dinv, 10
 - Dinvruiz, 10
 - Druiz, 10
 - Einvruiz, 10
 - Eruiz, 10
 - etree, 10

- FinvruiZ, 11
- Fruiz, 11
- fwork, 11
- GtoKKT, 11
- iwork, 11
- K, 11
- k, 11
- kinv, 11
- kktres, 12
- Li, 12
- Lnz, 12
- Lp, 12
- Lx, 12
- nt2kkt, 12
- ntdiag2kkt, 12
- p, 12
- pinv, 13
- Pnum_nzadded, 13
- Pnzadded_idx, 13
- PregtoKKT, 13
- rhs, 13
- xyz, 13
- xyzbuff1, 13
- xyzbuff2, 13
- QOCOProblemData, 14
 - A, 15
 - At, 15
 - b, 15
 - c, 15
 - G, 15
 - Gt, 15
 - h, 15
 - l, 16
 - m, 16
 - n, 16
 - nsoc, 16
 - P, 16
 - p, 16
 - q, 16
- QOCOSettings, 17
 - abstol, 17
 - abstol_inacc, 17
 - bisect_iters, 18
 - iter_ref_iters, 18
 - kkt_dynamic_reg, 18
 - kkt_static_reg, 18
 - max_iters, 18
 - reltol, 18
 - reltol_inacc, 18
 - ruiz_iters, 18
 - verbose, 19
- QOCOSolution, 19
 - dres, 19
 - gap, 19
 - iters, 20
 - obj, 20
 - pres, 20
 - s, 20
 - setup_time_sec, 20
 - solve_time_sec, 20
 - status, 20
 - x, 20
 - y, 21
 - z, 21
- QOCOSolver, 21
 - settings, 22
 - sol, 22
 - work, 22
- QOCOWorkspace, 23
 - a, 24
 - data, 24
 - Ds, 24
 - kkt, 24
 - lambda, 24
 - mu, 24
 - s, 25
 - sbar, 25
 - sigma, 25
 - solve_timer, 25
 - ubuff1, 25
 - ubuff2, 25
 - ubuff3, 25
 - W, 25
 - Wfull, 26
 - Winv, 26
 - Winvfull, 26
 - Wnnz, 26
 - Wnnzfull, 26
 - WtW, 26
 - x, 26
 - xbuff, 26
 - y, 27
 - ybuff, 27
 - z, 27
 - zbar, 27
- regularize
 - linalg.c, 108
 - linalg.h, 62
- reltol
 - QOCOSettings, 18
- reltol_inacc
 - QOCOSettings, 18
- rhs
 - QOCOKKT, 13
- row_col_scale
 - linalg.c, 108
 - linalg.h, 62
- row_inf_norm
 - linalg.c, 108
 - linalg.h, 63
- ruiz_equilibration
 - equilibration.c, 91
 - equilibration.h, 44
- ruiz_iters
 - QOCOSettings, 18

s
 QOCOSolution, 20
 QOCOWorkspace, 25
 safe_div
 definitions.h, 40
 sbar
 QOCOWorkspace, 25
 scale_arrayf
 linalg.c, 109
 linalg.h, 63
 set_default_settings
 qoco_api.c, 114
 qoco_api.h, 71
 set_nt_block_zeros
 kkt.c, 99
 kkt.h, 53
 settings
 QOCOSolver, 22
 setup_time_sec
 QOCOSolution, 20
 sigma
 QOCOWorkspace, 25
 soc_division
 cone.c, 89
 cone.h, 36
 soc_product
 cone.c, 89
 cone.h, 36
 soc_residual
 cone.c, 89
 cone.h, 36
 soc_residual2
 cone.c, 90
 cone.h, 37
 sol
 QOCOSolver, 22
 solve_time_sec
 QOCOSolution, 20
 solve_timer
 QOCOWorkspace, 25
 SpMtv
 linalg.c, 109
 linalg.h, 63
 SpMv
 linalg.c, 109
 linalg.h, 65
 start_timer
 timer.h, 77
 timer_linux.c, 118
 timer_macos.c, 120
 status
 QOCOSolution, 20
 stop_timer
 timer.h, 77
 timer_linux.c, 118
 timer_macos.c, 120
 timer.h
 get_elapsed_time_sec, 76
 start_timer, 77
 stop_timer, 77
 timer_linux.c
 get_elapsed_time_sec, 117
 start_timer, 118
 stop_timer, 118
 timer_macos.c
 get_elapsed_time_sec, 119
 start_timer, 120
 stop_timer, 120
 ubuff1
 QOCOWorkspace, 25
 ubuff2
 QOCOWorkspace, 25
 ubuff3
 QOCOWorkspace, 25
 unregularize
 linalg.c, 110
 linalg.h, 65
 unscale_variables
 equilibration.c, 92
 equilibration.h, 44
 update_matrix_data
 qoco_api.c, 114
 qoco_api.h, 71
 update_nt_block
 kkt.c, 99
 kkt.h, 53
 update_vector_data
 qoco_api.c, 115
 qoco_api.h, 71
 USpMv
 linalg.c, 110
 linalg.h, 65
 utils.c
 check_stopping, 122
 copy_settings, 122
 copy_solution, 123
 log_iter, 123
 print_arrayf, 123
 print_arrayi, 124
 print_footer, 124
 print_header, 124
 print_qoco_csc_matrix, 124
 utils.h
 check_stopping, 79
 copy_settings, 80
 copy_solution, 80
 log_iter, 80
 print_arrayf, 80
 print_arrayi, 81
 print_footer, 81
 print_header, 81
 print_qoco_csc_matrix, 82
 verbose
 QOCOSettings, 19

W

QOCOWorkspace, [25](#)

Wfull

QOCOWorkspace, [26](#)

Winv

QOCOWorkspace, [26](#)

Winvfull

QOCOWorkspace, [26](#)

Wnnz

QOCOWorkspace, [26](#)

Wnnzfull

QOCOWorkspace, [26](#)

work

QOCOSolver, [22](#)

WtW

QOCOWorkspace, [26](#)

x

QOCOCscMatrix, [8](#)

QOCOSolution, [20](#)

QOCOWorkspace, [26](#)

xbuff

QOCOWorkspace, [26](#)

xyz

QOCOKKT, [13](#)

xyzbuff1

QOCOKKT, [13](#)

xyzbuff2

QOCOKKT, [13](#)

y

QOCOSolution, [21](#)

QOCOWorkspace, [27](#)

ybuff

QOCOWorkspace, [27](#)

z

QOCOSolution, [21](#)

QOCOWorkspace, [27](#)

zbar

QOCOWorkspace, [27](#)