FCLIB - v - Package

Generated by Doxygen 1.8.5

Thu Feb 13 2014 08:40:58

Contents

1 Introduction

1.1 What is FCLIB ?

FCLIB is

- A open source collection of Frictional Contact (FC) problems stored in a specific `HDF5 format`
- A open source light implementation of Input/Output functions in C Language to read and write problems

1.2 Goals of the project

The goal of this work is to set up a collection of 2D and 3D Frictional Contact (FC) problems in order to

- set up a list of benchmarks
- provide a standard framework for testing available and new algorithms for solving discrete frictional contact problems
- share common formulations of problems in order to exchange data

1.3 How to download ?

see Download section

1.4 What is a Frictional contact problem ?

A Frictional contact problem is algebraic problem obtained after possible time and space discretizations of problems of mechanics of solid involving contact and Coulomb's friction. The mathematical structure of the problem is a second-order cone complementarity problem. For more details, you could have a look to the `fclib specifications`

1.4.1 The local Frictional Contact problem with equality constraints

Given

- a positive semi–definite matrix $W \in {\rm I\!R}^{m\times m}$
- a matrix $V \in {\rm I\!R}^{m\times p}$
- a matrix $R \in {\rm I\!R}^{p\times p}$
- a vector $q \in {\rm I\!R}^{m}$,
- a vector $s \in {\rm I\!R}^{p}$,
- a vector of coefficients of friction $\mu \in {\rm I\!R}^{n_c}$

the Mixed 3DFC problem is to find three vectors $u \in \mathbb{R}^m$, $r \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}^p$ denoted by $\mathrm{M3DFC}(R,V,W,q,s,\mu)$ such that

$$\begin{cases} V^T r + R\lambda + s = 0 \\ \\ \hat{u} = Wr + V\lambda + q + \left[\left[\begin{array}{c} \mu^\alpha \|u^\alpha_T\| \\ 0 \\ 0 \end{array}\right]^T, \alpha = 1\dots n_c\right]^T \\ \\ C^\star_\mu \ni \hat{u} \perp r \in C_\mu \end{cases}$$

where the Coulomb friction cone for a contact α is defined by

$$C^\alpha_{\mu^\alpha} = \{r^\alpha, \|r^\alpha_T\| \leq \mu^\alpha |r^\alpha_N|\}$$

and the set $C^{\alpha,\star}_{\mu^\alpha}$ is its dual.

1.4.2 The Global Frictional Contact problem with equality constraints

We are also dealing with global FC problem defined by

Given

- a symmetric positive definite matrix $M \in \mathbb{R}^{n\times n}$
- a vector $f \in \mathbb{R}^n$,
- a matrix $H \in \mathbb{R}^{n\times m}$
- a matrix $G \in \mathbb{R}^{n\times p}$
- a vector $w \in \mathbb{R}^m$,
- a vector $b \in \mathbb{R}^p$,
- a vector of coefficients of friction $\mu \in \mathbb{R}^{n_c}$

the Global Mixed 3DFC problem is to find four vectors $v \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $r \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}^p$ denoted by $\mathrm{GM3DFC}(M,H,G,w,b,\mu)$ such that

$$\begin{cases} Mv = Hr + G\lambda + f \\ \\ G^T v + b = 0 \\ \\ \hat{u} = H^T v + w + \left[\left[\begin{array}{c} \mu \|u^\alpha_T\| \\ 0 \\ 0 \end{array}\right]^T, \alpha = 1\dots n_c\right]^T \\ \\ C^\star_\mu \ni \hat{u} \perp r \in C_\mu \end{cases}$$

1.4.3 Problems without equality constraints

If the original problems do not contain inequality constraints, or if there are reduced, the problems do no have the variables λ as unknowns and can be simplified. However, the storage in HDF5 file remains the same.

1.4.4 functions.

The API provides also some Merit functions whixh measures it one set of vectors satifies the previous problems.

2 Download

2.1 How to download sources files of the API?

- latest version on the svn server access at `FCLIB Gforge`
- tar files available at `FCLIB Gforge`

2.2 How to download the collection of problems ?

- A preliminary version is avalaible here `FCLIB library v 0.1`

2.3 Binaries

- Coming soon at `FCLIB Gforge`

3 Contact us

For any information or help, send an email to

4 Related Publications

Coming soon ...

5 Class Index

5.1 Class List

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

6 File Index

6.1 File List

Here is a list of all files with brief descriptions:

7 Class Documentation

7.1 cs_dmperm_results Struct Reference

```
#include <csparse.h>
```

Public Attributes

- int ∗ P
- int ∗ Q
- int ∗ R
- int ∗ S
- int nb
- int rr [5]
- int cc [5]

7.1.1 Detailed Description

Definition at line 67 of file csparse.h.

7.1.2 Member Data Documentation

7.1.2.1 int∗ cs_dmperm_results::P

Definition at line 69 of file csparse.h.

Referenced by cs_dalloc(), cs_dfree(), cs_dmperm(), and cs_scc().

7.1.2.2 int∗ cs_dmperm_results::Q

Definition at line 70 of file csparse.h.

Referenced by cs_dalloc(), cs_dfree(), and cs_dmperm().

7.1.2.3 int∗ cs_dmperm_results::R

Definition at line 71 of file csparse.h.

Referenced by cs_dalloc(), cs_dfree(), cs_dmperm(), and cs_scc().

7.1.2.4 int∗ cs_dmperm_results::S

Definition at line 72 of file csparse.h.

Referenced by cs_dalloc(), cs_dfree(), and cs_dmperm().

7.1.2.5 int cs_dmperm_results::nb

Definition at line 73 of file csparse.h.

Referenced by cs_dmperm(), and cs_scc().

7.1.2.6 int cs_dmperm_results::rr[5]

Definition at line 74 of file csparse.h.

Referenced by cs_dmperm().

7.1.2.7 int cs_dmperm_results::cc[5]

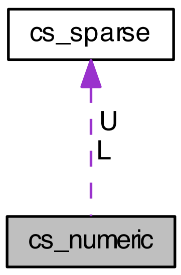
Definition at line 75 of file csparse.h.

Referenced by cs_dmperm().

7.2 cs_numeric Struct Reference

```
#include <csparse.h>
```

Collaboration diagram for cs_numeric:



Public Attributes

- cs ∗ L
- cs ∗ U
- int ∗ Pinv
- double ∗ B

7.2.1 Detailed Description

Definition at line 59 of file csparse.h.

7.2.2 Member Data Documentation

7.2.2.1 cs∗ cs_numeric::L

Definition at line 61 of file csparse.h.

Referenced by cs_chol(), cs_cholsol(), cs_lu(), cs_lusol(), cs_nfree(), cs_qr(), and cs_qrsol().

7.2.2.2 cs∗ cs_numeric::U

Definition at line 62 of file csparse.h.

Referenced by cs_lu(), cs_lusol(), cs_nfree(), cs_qr(), and cs_qrsol().

7.2.2.3 int∗ cs_numeric::Pinv

Definition at line 63 of file csparse.h.

Referenced by cs_lu(), cs_lusol(), and cs_nfree().

7.2.2.4 double∗ cs_numeric::B

Definition at line 64 of file csparse.h.

Referenced by cs_nfree(), cs_qr(), and cs_qrsol().

7.3 cs_sparse Struct Reference

```
#include <csparse.h>
```

Public Attributes

- int nzmax
- int m
- int n
- int ∗ p
- int ∗ i
- double ∗ x
- int nz

7.3.1 Detailed Description

Definition at line 14 of file csparse.h.

7.3.2 Member Data Documentation

7.3.2.1 int cs_sparse::nzmax

Definition at line 16 of file csparse.h.

Referenced by cs_amd(), cs_entry(), cs_lu(), cs_print(), cs_spalloc(), and cs_sprealloc().

7.3.2.2 int cs_sparse::m

Definition at line 17 of file csparse.h.

Referenced by cs_add(), cs_amd(), cs_counts(), cs_dmperm(), cs_dupl(), cs_entry(), cs_etree(), cs_maxtrans(), cs-_multiply(), cs_permute(), cs_print(), cs_qr(), cs_qrsol(), cs_spalloc(), cs_transpose(), cs_triplet(), and cs_vcount().

7.3.2.3 int cs_sparse::n

Definition at line 18 of file csparse.h.

Referenced by cs_add(), cs_amd(), cs_chol(), cs_cholsol(), cs_counts(), cs_dmperm(), cs_dupl(), cs_entry(), cs-_etree(), cs_fkeep(), cs_gaxpy(), cs_lsolve(), cs_ltsolve(), cs_lu(), cs_lusol(), cs_maxtrans(), cs_multiply(), cs_-norm(), cs_permute(), cs_print(), cs_qr(), cs_qrsol(), cs_reach(), cs_scc(), cs_schol(), cs_spalloc(), cs_splsolve(), cs_sprealloc(), cs_sqr(), cs_symperm(), cs_transpose(), cs_triplet(), cs_updown(), cs_usolve(), cs_utsolve(), and cs_vcount().

7.3.2.4 int∗ cs_sparse::p

Definition at line 19 of file csparse.h.

Referenced by cs_add(), cs_amd(), cs_augment(), cs_bfs(), cs_chol(), cs_counts(), cs_dfs(), cs_dmperm(), cs_-dupl(), cs_entry(), cs_ereach(), cs_etree(), cs_fkeep(), cs_gaxpy(), cs_happly(), cs_lsolve(), cs_ltsolve(), cs_lu(), cs_maxtrans(), cs_multiply(), cs_norm(), cs_permute(), cs_print(), cs_qr(), cs_reach(), cs_scatter(), cs_scc(), cs-_spalloc(), cs_spfree(), cs_splsolve(), cs_sprealloc(), cs_sqr(), cs_symperm(), cs_transpose(), cs_triplet(), cs_-updown(), cs_usolve(), cs_utsolve(), and cs_vcount().

7.3.2.5 int∗ cs_sparse::i

Definition at line 20 of file csparse.h.

Referenced by cs_amd(), cs_augment(), cs_bfs(), cs_chol(), cs_counts(), cs_dfs(), cs_dmperm(), cs_dupl(), cs-_entry(), cs_ereach(), cs_etree(), cs_fkeep(), cs_gaxpy(), cs_happly(), cs_lsolve(), cs_ltsolve(), cs_lu(), cs_-maxtrans(), cs_multiply(), cs_permute(), cs_print(), cs_qr(), cs_reach(), cs_scatter(), cs_spalloc(), cs_spfree(), cs_-splsolve(), cs_sprealloc(), cs_symperm(), cs_transpose(), cs_triplet(), cs_updown(), cs_usolve(), cs_utsolve(), and cs_vcount().

7.3.2.6 double* cs_sparse::x

Definition at line 21 of file csparse.h.

Referenced by cs_add(), cs_chol(), cs_dupl(), cs_entry(), cs_ereach(), cs_fkeep(), cs_gaxpy(), cs_happly(), cs_-lsolve(), cs_ltsolve(), cs_lu(), cs_multiply(), cs_norm(), cs_permute(), cs_print(), cs_qr(), cs_scatter(), cs_spalloc(), cs_spfree(), cs_splsolve(), cs_sprealloc(), cs_symperm(), cs_transpose(), cs_triplet(), cs_updown(), cs_usolve(), and cs_utsolve().

7.3.2.7 int cs_sparse::nz

Definition at line 22 of file csparse.h.

Referenced by cs_entry(), cs_print(), cs_spalloc(), cs_sprealloc(), and cs_triplet().

7.4 cs_symbolic Struct Reference

```
#include <csparse.h>
```

Public Attributes

- int * Pinv
- int * Q
- int * parent
- int * cp
- int m2
- int lnz
- int unz

7.4.1 Detailed Description

Definition at line 48 of file csparse.h.

7.4.2 Member Data Documentation

7.4.2.1 int* cs_symbolic::Pinv

Definition at line 50 of file csparse.h.

Referenced by cs_chol(), cs_cholsol(), cs_qr(), cs_qrsol(), cs_schol(), cs_sfree(), and cs_sqr().

7.4.2.2 int* cs_symbolic::Q

Definition at line 51 of file csparse.h.

Referenced by cs_lu(), cs_lusol(), cs_qr(), cs_qrsol(), cs_sfree(), and cs_sqr().

7.4.2.3 int* cs_symbolic::parent

Definition at line 52 of file csparse.h.

Referenced by cs_chol(), cs_qr(), cs_schol(), cs_sfree(), and cs_sqr().

7.4.2.4 int* cs_symbolic::cp

Definition at line 53 of file csparse.h.

Referenced by cs_chol(), cs_schol(), cs_sfree(), and cs_sqr().

7.4.2.5 int cs_symbolic::m2

Definition at line 54 of file csparse.h.

Referenced by cs_qr(), cs_qrsol(), and cs_sqr().

7.4.2.6 int cs_symbolic::lnz

Definition at line 55 of file csparse.h.

Referenced by cs_lu(), cs_qr(), cs_schol(), and cs_sqr().

7.4.2.7 int cs_symbolic::unz

Definition at line 56 of file csparse.h.

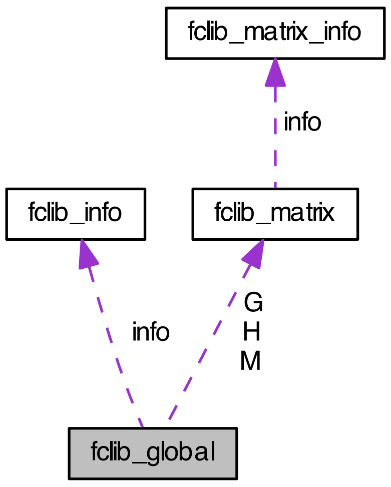
Referenced by cs_lu(), cs_qr(), cs_schol(), and cs_sqr().

7.5 fclib_global Struct Reference

The global frictional contact problem defined by.

```
#include <fclib.h>
```

Collaboration diagram for fclib_global:



Public Attributes

- struct fclib_matrix $*$ M

    the matrix M (see mathematical description below)
- struct fclib_matrix $*$ H

    the matrix M (see mathematical description below)
- struct fclib_matrix $*$ G

    the matrix M (see mathematical description below)
- double $*$ mu

the vector μ of coefficient of friction (see mathematical description below)

- double $*$ f

  the vector f (see mathematical description below)
- double $*$ b

  the vector b (see mathematical description below)
- double $*$ w

  the vector w (see mathematical description below)
- int spacedim

  the dimension , 2 or 3, of the local space at contact (2d or 3d friction contact laws)
- struct fclib_info $*$ info

  info on the problem

7.5.1 Detailed Description

The global frictional contact problem defined by.

Given

- a symmetric positive definite matrix $M \in {\rm I\!R}^{n\times n}$
- a vector $f \in {\rm I\!R}^{n}$,
- a matrix $H \in {\rm I\!R}^{n\times m}$
- a matrix $G \in {\rm I\!R}^{n\times p}$
- a vector $w \in {\rm I\!R}^{m}$,
- a vector $b \in {\rm I\!R}^{p}$,
- a vector of coefficients of friction $\mu \in {\rm I\!R}^{n_c}$

the Global Mixed 3DFC problem is to find four vectors $v \in {\rm I\!R}^{n}$, $u \in {\rm I\!R}^{m}$, $r \in {\rm I\!R}^{m}$ and $\lambda \in {\rm I\!R}^{p}$ denoted by $\mathrm{GM3DFC}(M,H,G,w,b,\mu)$ such that

$$\begin{cases} Mv = Hr + G\lambda + f \\ G^T v + b = 0 \\ \hat{u} = H^T v + w + \left[\left[\begin{array}{ccc} \mu \|u^\alpha_T\| & 0 & 0 \end{array} \right]^T, \alpha = 1 \dots n_c \right]^T \\ C^\star_\mu \ni \hat{u} \perp r \in C_\mu \end{cases}$$

where the Coulomb friction cone for a contact α is defined by

$$C^\alpha_{\mu^\alpha} = \{ r^\alpha, \|r^\alpha_T\| \leq \mu^\alpha |r^\alpha_N| \} *$$

and the set $C^{\alpha,\star}_{\mu^\alpha}$ is its dual.

Definition at line 174 of file fclib.h.

7.5.2 Member Data Documentation

7.5.2.1 struct **fclib_matrix**$*$ fclib_global::M

the matrix M (see mathematical description below)

Definition at line 177 of file fclib.h.

Referenced by compare_global_problems(), fclib_delete_global(), fclib_read_global(), fclib_write_global(), main(), random_global_problem(), random_global_solutions(), read_global_vectors(), and write_global_vectors().

7.5.2.2 struct fclib_matrix∗ fclib_global::H

the matrix M (see mathematical description below)

Definition at line 179 of file fclib.h.

Referenced by compare_global_problems(), fclib_delete_global(), fclib_read_global(), fclib_write_global(), main(), random_global_problem(), random_global_solutions(), read_global_vectors(), and write_global_vectors().

7.5.2.3 struct fclib_matrix∗ fclib_global::G

the matrix M (see mathematical description below)

Definition at line 181 of file fclib.h.

Referenced by compare_global_problems(), fclib_delete_global(), fclib_read_global(), fclib_write_global(), main(), random_global_problem(), random_global_solutions(), read_global_vectors(), and write_global_vectors().

7.5.2.4 double∗ fclib_global::mu

the vector μ of coefficient of friction (see mathematical description below)

Definition at line 183 of file fclib.h.

Referenced by compare_global_problems(), fclib_delete_global(), random_global_problem(), read_global_-vectors(), and write_global_vectors().

7.5.2.5 double∗ fclib_global::f

the vector f (see mathematical description below)

Definition at line 185 of file fclib.h.

Referenced by compare_global_problems(), fclib_delete_global(), random_global_problem(), read_global_-vectors(), and write_global_vectors().

7.5.2.6 double∗ fclib_global::b

the vector b (see mathematical description below)

Definition at line 187 of file fclib.h.

Referenced by compare_global_problems(), fclib_delete_global(), random_global_problem(), read_global_-vectors(), and write_global_vectors().

7.5.2.7 double∗ fclib_global::w

the vector w (see mathematical description below)

Definition at line 189 of file fclib.h.

Referenced by compare_global_problems(), fclib_delete_global(), random_global_problem(), read_global_-vectors(), and write_global_vectors().

7.5.2.8 int fclib_global::spacedim

the dimension , 2 or 3, of the local space at contact (2d or 3d friction contact laws)

Definition at line 191 of file fclib.h.

Referenced by compare_global_problems(), fclib_read_global(), fclib_write_global(), random_global_problem(), read_global_vectors(), and write_global_vectors().

7.5.2.9 struct fclib_info∗ fclib_global::info

info on the problem

Definition at line 193 of file fclib.h.

Referenced by compare_global_problems(), fclib_delete_global(), fclib_read_global(), fclib_write_global(), and random_global_problem().

7.6 fclib_info Struct Reference

This structure allows the user to enter a problem information as a title, a short description and known mathematical properties of the problem.

```
#include <fclib.h>
```

Public Attributes

- char ∗ title

  title of the problem
- char ∗ description

  short decription of the problem
- char ∗ math_info

  known properties of the problem (existence, uniqueness, ...)

7.6.1 Detailed Description

This structure allows the user to enter a problem information as a title, a short description and known mathematical properties of the problem.

Definition at line 91 of file fclib.h.

7.6.2 Member Data Documentation

7.6.2.1 char∗ fclib_info::title

title of the problem

Definition at line 94 of file fclib.h.

Referenced by compare_infos(), delete_info(), problem_info(), read_problem_info(), and write_problem_info().

7.6.2.2 char∗ fclib_info::description

short decription of the problem

Definition at line 96 of file fclib.h.

Referenced by compare_infos(), delete_info(), problem_info(), read_problem_info(), and write_problem_info().

7.6.2.3 char∗ fclib_info::math_info

known properties of the problem (existence, uniqueness, ...)

Definition at line 98 of file fclib.h.

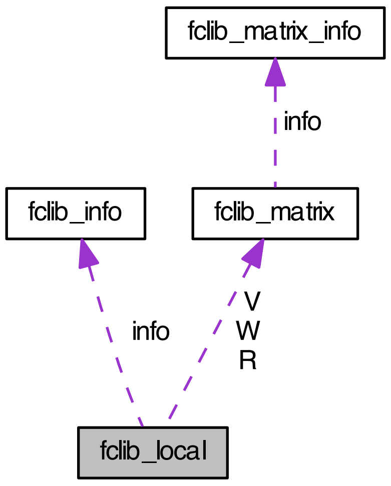
Referenced by compare_infos(), delete_info(), problem_info(), read_problem_info(), and write_problem_info().

7.7 fclib_local Struct Reference

The local frictional contact problem defined by.

```
#include <fclib.h>
```

Collaboration diagram for fclib_local:



Public Attributes

- struct fclib_matrix ∗ W

  the matrix W (see mathematical description below)
- struct fclib_matrix ∗ V

  the matrix V (see mathematical description below)
- struct fclib_matrix ∗ R

  the matrix R (see mathematical description below)
- double ∗ mu

  the vector μ of coefficient of friction (see mathematical description below)
- double ∗ q

  the vector q (see mathematical description below)
- double ∗ s

  the vector s (see mathematical description below)
- int spacedim

  the dimension , 2 or 3, of the local space at contact (2d or 3d friction contact laws)
- struct fclib_info ∗ info

  info on the problem

7.7.1 Detailed Description

The local frictional contact problem defined by.

given

- a positive semi–definite matrix $W \in \mathbb{R}^{m\times m}$
- a matrix $V \in \mathbb{R}^{m\times p}$
- a matrix $R \in \mathbb{R}^{p\times p}$

- a vector $q \in \mathbb{R}^m$,
- a vector $s \in \mathbb{R}^p$,
- a vector of coefficients of friction $\mu \in \mathbb{R}^{n_c}$

the Mixed 3DFC problem is to find three vectors $u \in \mathbb{R}^m$, $r \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}^p$ denoted by $\mathrm{M3DFC}(R,V,W,q,s,\mu)$ such that

$$* \begin{cases} V^T r + R\lambda + s = 0 \\ \hat{u} = Wr + V\lambda + q + \left[\left[\begin{array}{ccc} \mu^\alpha \|u_T^\alpha\| & 0 & 0 \end{array} \right]^T, \alpha = 1 \dots n_c \right]^T \\ C_\mu^\star \ni \hat{u} \perp r \in C_\mu \end{cases}$$

where the Coulomb friction cone for a contact α is defined by

$$C_{\mu^\alpha}^\alpha = \{r^\alpha, \|r_T^\alpha\| \leq \mu^\alpha |r_N^\alpha|\}$$

and the set $C_{\mu^\alpha}^{\alpha,\star}$ is its dual.

Definition at line 228 of file fclib.h.

7.7.2 Member Data Documentation

7.7.2.1 struct **fclib_matrix**$*$ fclib_local::W

the matrix W (see mathematical description below)

Definition at line 231 of file fclib.h.

Referenced by compare_local_problems(), fclib_delete_local(), fclib_merit_local(), fclib_read_local(), fclib_write_-local(), main(), random_local_problem(), random_local_solutions(), read_local_vectors(), and write_local_vectors().

7.7.2.2 struct **fclib_matrix**$*$ fclib_local::V

the matrix V (see mathematical description below)

Definition at line 233 of file fclib.h.

Referenced by compare_local_problems(), fclib_delete_local(), fclib_merit_local(), fclib_read_local(), fclib_write_-local(), random_local_problem(), and write_local_vectors().

7.7.2.3 struct **fclib_matrix**$*$ fclib_local::R

the matrix R (see mathematical description below)

Definition at line 235 of file fclib.h.

Referenced by compare_local_problems(), fclib_delete_local(), fclib_merit_local(), fclib_read_local(), fclib_write_-local(), main(), random_local_problem(), random_local_solutions(), read_local_vectors(), and write_local_vectors().

7.7.2.4 double$*$ fclib_local::mu

the vector μ of coefficient of friction (see mathematical description below)

Definition at line 237 of file fclib.h.

Referenced by compare_local_problems(), fclib_delete_local(), fclib_merit_local(), random_local_problem(), read_-local_vectors(), and write_local_vectors().

7.7.2.5 double$*$ fclib_local::q

the vector q (see mathematical description below)

Definition at line 239 of file fclib.h.

Referenced by compare_local_problems(), fclib_delete_local(), fclib_merit_local(), random_local_problem(), read_-local_vectors(), and write_local_vectors().

7.7.2.6 double∗ fclib_local::s

the vector s (see mathematical description below)

Definition at line 241 of file fclib.h.

Referenced by compare_local_problems(), fclib_delete_local(), fclib_merit_local(), random_local_problem(), read_-local_vectors(), and write_local_vectors().

7.7.2.7 int fclib_local::spacedim

the dimension , 2 or 3, of the local space at contact (2d or 3d friction contact laws)

Definition at line 243 of file fclib.h.

Referenced by compare_local_problems(), fclib_merit_local(), fclib_read_local(), fclib_write_local(), random_local-_problem(), read_local_vectors(), and write_local_vectors().

7.7.2.8 struct **fclib_info**∗ fclib_local::info

info on the problem

Definition at line 245 of file fclib.h.

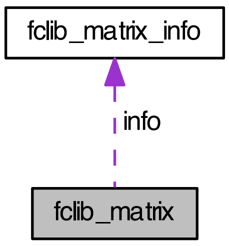
Referenced by compare_local_problems(), fclib_delete_local(), fclib_read_local(), fclib_write_local(), and random_-local_problem().

7.8 fclib_matrix Struct Reference

matrix in compressed row/column or triplet form

```
#include <fclib.h>
```

Collaboration diagram for fclib_matrix:



Public Attributes

- int nzmax

  maximum number of entries
- int m

  number of rows
- int n

  number of columns
- int ∗ p

  compressed: row (size m+1) or column (size n+1) pointers; triplet: row indices (size nz)

- int ∗ i

  compressed: column or row indices, size nzmax; triplet: column indices (size nz)
- double ∗ x

  numerical values, size nzmax
- int nz

  of entries in triplet matrix, -1 for compressed columns, -2 for compressed rows
- struct fclib_matrix_info ∗ info

  info for this matrix

7.8.1 Detailed Description

matrix in compressed row/column or triplet form

Definition at line 119 of file fclib.h.

7.8.2 Member Data Documentation

7.8.2.1 int fclib_matrix::nzmax

maximum number of entries

Definition at line 122 of file fclib.h.

Referenced by compare_matrices(), random_matrix(), read_matrix(), and write_matrix().

7.8.2.2 int fclib_matrix::m

number of rows

Definition at line 124 of file fclib.h.

Referenced by compare_global_problems(), compare_matrices(), main(), matrix_info(), random_matrix(), read-_global_vectors(), read_local_vectors(), read_matrix(), write_global_vectors(), write_local_vectors(), and write_-matrix().

7.8.2.3 int fclib_matrix::n

number of columns

Definition at line 126 of file fclib.h.

Referenced by compare_global_problems(), compare_local_problems(), compare_matrices(), fclib_merit_local(), main(), random_global_problem(), random_global_solutions(), random_local_solutions(), random_matrix(), read_-global_vectors(), read_matrix(), write_global_vectors(), and write_matrix().

7.8.2.4 int∗ fclib_matrix::p

compressed: row (size m+1) or column (size n+1) pointers; triplet: row indices (size nz)

Definition at line 128 of file fclib.h.

Referenced by compare_matrices(), delete_matrix(), random_matrix(), read_matrix(), and write_matrix().

7.8.2.5 int∗ fclib_matrix::i

compressed: column or row indices, size nzmax; triplet: column indices (size nz)

Definition at line 130 of file fclib.h.

Referenced by compare_matrices(), delete_matrix(), fclib_merit_local(), random_matrix(), read_matrix(), and write-_matrix().

7.8.2.6 double∗ fclib_matrix::x

numerical values, size nzmax

Definition at line 132 of file fclib.h.

Referenced by compare_matrices(), delete_matrix(), random_matrix(), read_matrix(), and write_matrix().

7.8.2.7 int fclib_matrix::nz

of entries in triplet matrix, -1 for compressed columns, -2 for compressed rows

Definition at line 134 of file fclib.h.

Referenced by compare_matrices(), random_matrix(), read_matrix(), and write_matrix().

7.8.2.8 struct fclib_matrix_info∗ fclib_matrix::info

info for this matrix

Definition at line 136 of file fclib.h.

Referenced by compare_matrices(), delete_matrix(), random_matrix(), read_matrix(), and write_matrix().

7.9 fclib_matrix_info Struct Reference

This structure allows the user to enter a description for a given matrix (comment, conditionning, determinant, rank.) if they are known.

```
#include <fclib.h>
```

Public Attributes

- char ∗ comment

  comment on the matrix properties
- double conditioning

  conditioning
- double determinant

  determinant
- int rank

  rank

7.9.1 Detailed Description

This structure allows the user to enter a description for a given matrix (comment, conditionning, determinant, rank.) if they are known.

Definition at line 104 of file fclib.h.

7.9.2 Member Data Documentation

7.9.2.1 char∗ fclib_matrix_info::comment

comment on the matrix properties

Definition at line 107 of file fclib.h.

Referenced by compare_matrix_infos(), delete_matrix_info(), matrix_info(), read_matrix(), and write_matrix().

7.9.2.2 double fclib_matrix_info::conditioning

conditioning

Definition at line 109 of file fclib.h.

Referenced by compare_matrix_infos(), matrix_info(), read_matrix(), and write_matrix().

7.9.2.3 double fclib_matrix_info::determinant

determinant

Definition at line 111 of file fclib.h.

Referenced by compare_matrix_infos(), matrix_info(), read_matrix(), and write_matrix().

7.9.2.4 int fclib_matrix_info::rank

rank

Definition at line 113 of file fclib.h.

Referenced by compare_matrix_infos(), matrix_info(), read_matrix(), and write_matrix().

7.10 fclib_solution Struct Reference

A solution or a guess for the frictional contact problem.

```
#include <fclib.h>
```

Public Attributes

- double ∗ v

  global velocity (or position/displacement for quasi-static problems) solution vector
- double ∗ u

  local velocity (or position/displacement for quasi-static problems) solution vector
- double ∗ r

  local contact forces (or impulses) solution vector
- double ∗ l

  multiplier for equlity constraints (λ) solution vector

7.10.1 Detailed Description

A solution or a guess for the frictional contact problem.

This structure allows to store a solution vector of a guess vector for the various frictional contact problems.

Definition at line 254 of file fclib.h.

7.10.2 Member Data Documentation

7.10.2.1 double∗ fclib_solution::v

global velocity (or position/displacement for quasi-static problems) solution vector

Definition at line 257 of file fclib.h.

Referenced by compare_solutions(), fclib_delete_solutions(), fclib_merit_local(), random_global_solutions(), random_local_solutions(), read_solution(), and write_solution().

7.10.2.2 double∗ fclib_solution::u

local velocity (or position/displacement for quasi-static problems) solution vector

Definition at line 259 of file fclib.h.

Referenced by compare_solutions(), fclib_delete_solutions(), fclib_merit_local(), random_global_solutions(), random_local_solutions(), read_solution(), and write_solution().

7.10.2.3 double∗ fclib_solution::r

local contact forces (or impulses) solution vector

Definition at line 261 of file fclib.h.

Referenced by compare_solutions(), fclib_delete_solutions(), fclib_merit_local(), random_global_solutions(), random_local_solutions(), read_solution(), and write_solution().

7.10.2.4 double∗ fclib_solution::l

multiplier for equlity constraints (λ) solution vector

Definition at line 263 of file fclib.h.

Referenced by compare_solutions(), fclib_delete_solutions(), fclib_merit_local(), random_global_solutions(), random_local_solutions(), read_solution(), and write_solution().

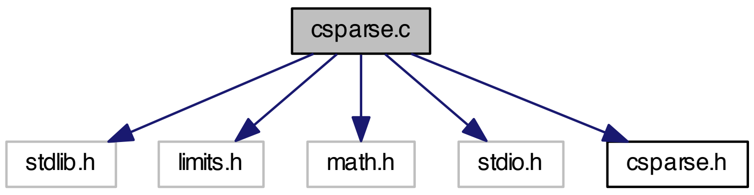
8 File Documentation

8.1 additionalpages.doxygen File Reference

8.2 csparse.c File Reference

```
#include <stdlib.h>
#include <limits.h>
#include <math.h>
#include <stdio.h>
#include "csparse.h"
```

Include dependency graph for csparse.c:



Functions

- cs ∗ cs_add (const cs ∗A, const cs ∗B, double alpha, double beta)
- static int cs_wclear (int mark, int lemax, int ∗w, int n)
- static int cs_diag (int i, int j, double aij, void ∗other)
- int ∗ cs_amd (const cs ∗A, int order)

- static int cs_ereach (const cs ∗A, int k, const int ∗parent, int ∗s, int ∗w, double ∗x, int top)
- csn ∗ cs_chol (const cs ∗A, const css ∗S)
- int cs_cholsol (const cs ∗A, double ∗b, int order)
- static void cs_cedge (int j, int i, const int ∗first, int ∗maxfirst, int ∗delta, int ∗prevleaf, int ∗ancestor)
- int ∗ cs_counts (const cs ∗A, const int ∗parent, const int ∗post, int ata)
- int cs_cumsum (int ∗p, int ∗c, int n)
- int cs_dfs (int j, cs ∗L, int top, int ∗xi, int ∗pstack, const int ∗Pinv)
- static int cs_bfs (const cs ∗A, int n, int ∗wi, int ∗wj, int ∗queue, const int ∗imatch, const int ∗jmatch, int mark)
- static void cs_matched (int m, const int ∗wi, const int ∗jmatch, int ∗P, int ∗Q, int ∗cc, int ∗rr, int set, int mark)
- static void cs_unmatched (int m, const int ∗wi, int ∗P, int ∗rr, int set)
- static int cs_rprune (int i, int j, double aij, void ∗other)
- csd ∗ cs_dmperm (const cs ∗A)
- static int cs_tol (int i, int j, double aij, void ∗tol)
- int cs_droptol (cs ∗A, double tol)
- static int cs_nonzero (int i, int j, double aij, void ∗other)
- int cs_dropzeros (cs ∗A)
- int cs_dupl (cs ∗A)
- int cs_entry (cs ∗T, int i, int j, double x)
- int ∗ cs_etree (const cs ∗A, int ata)
- int cs_fkeep (cs ∗A, int(∗fkeep)(int, int, double, void ∗), void ∗other)
- int cs_gaxpy (const cs ∗A, const double ∗x, double ∗y)
- int cs_happly (const cs ∗V, int i, double beta, double ∗x)
- double cs_house (double ∗x, double ∗beta, int n)
- int cs_ipvec (int n, const int ∗P, const double ∗b, double ∗x)
- cs ∗ cs_load (FILE ∗f)
- int cs_lsolve (const cs ∗L, double ∗x)
- int cs_ltsolve (const cs ∗L, double ∗x)
- csn ∗ cs_lu (const cs ∗A, const css ∗S, double tol)
- int cs_lusol (const cs ∗A, double ∗b, int order, double tol)
- void ∗ cs_malloc (int n, size_t size)
- void ∗ cs_calloc (int n, size_t size)
- void ∗ cs_free (void ∗p)
- void ∗ cs_realloc (void ∗p, int n, size_t size, int ∗ok)
- static void cs_augment (int k, const cs ∗A, int ∗jmatch, int ∗cheap, int ∗w, int ∗js, int ∗is, int ∗ps)
- int ∗ cs_maxtrans (const cs ∗A)
- cs ∗ cs_multiply (const cs ∗A, const cs ∗B)
- double cs_norm (const cs ∗A)
- cs ∗ cs_permute (const cs ∗A, const int ∗Pinv, const int ∗Q, int values)
- int ∗ cs_pinv (int const ∗P, int n)
- int ∗ cs_post (int n, const int ∗parent)
- int cs_print (const cs ∗A, int brief)
- int cs_pvec (int n, const int ∗P, const double ∗b, double ∗x)
- csn ∗ cs_qr (const cs ∗A, const css ∗S)
- int cs_qrsol (const cs ∗A, double ∗b, int order)
- int cs_reach (cs ∗L, const cs ∗B, int k, int ∗xi, const int ∗Pinv)
- int cs_scatter (const cs ∗A, int j, double beta, int ∗w, double ∗x, int mark, cs ∗C, int nz)
- csd ∗ cs_scc (cs ∗A)
- css ∗ cs_schol (const cs ∗A, int order)
- int cs_splsolve (cs ∗L, const cs ∗B, int k, int ∗xi, double ∗x, const int ∗Pinv)
- static int ∗ cs_vcount (const cs ∗A, const int ∗parent, int ∗m2, int ∗vnz)
- css ∗ cs_sqr (const cs ∗A, int order, int qr)
- cs ∗ cs_symperm (const cs ∗A, const int ∗Pinv, int values)
- int cs_tdfs (int j, int k, int ∗head, const int ∗next, int ∗post, int ∗stack)
- cs ∗ cs_transpose (const cs ∗A, int values)
- cs ∗ cs_triplet (const cs ∗T)

- int cs_updown (cs ∗L, int sigma, const cs ∗C, const int ∗parent)
- int cs_usolve (const cs ∗U, double ∗x)
- cs ∗ cs_spalloc (int m, int n, int nzmax, int values, int triplet)
- int cs_sprealloc (cs ∗A, int nzmax)
- cs ∗ cs_spfree (cs ∗A)
- csn ∗ cs_nfree (csn ∗N)
- css ∗ cs_sfree (css ∗S)
- csd ∗ cs_dalloc (int m, int n)
- csd ∗ cs_dfree (csd ∗D)
- cs ∗ cs_done (cs ∗C, void ∗w, void ∗x, int ok)
- int ∗ cs_idone (int ∗p, cs ∗C, void ∗w, int ok)
- csn ∗ cs_ndone (csn ∗N, cs ∗C, void ∗w, void ∗x, int ok)
- csd ∗ cs_ddone (csd ∗D, cs ∗C, void ∗w, int ok)
- int cs_utsolve (const cs ∗U, double ∗x)

8.2.1 Function Documentation

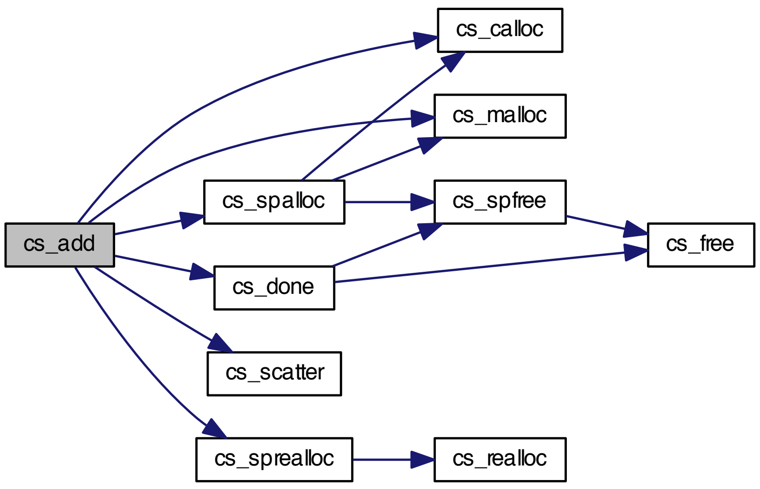
8.2.1.1 cs∗ cs_add (const cs ∗ *A*, const cs ∗ *B*, double *alpha*, double *beta*)

Definition at line 8 of file csparse.c.

References cs_calloc(), cs_done(), cs_malloc(), cs_scatter(), cs_spalloc(), cs_sprealloc(), cs_sparse::m, cs_-sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_amd().

Here is the call graph for this function:



8.2.1.2 static int cs_wclear (int *mark*, int *lemax*, int ∗ *w*, int *n*) [static]

Definition at line 50 of file csparse.c.

Referenced by cs_amd().

8.2.1.3 static int cs_diag (int *i,* int *j,* double *aij,* void ∗ *other*) [static]

Definition at line 73 of file csparse.c.

Referenced by cs_amd().

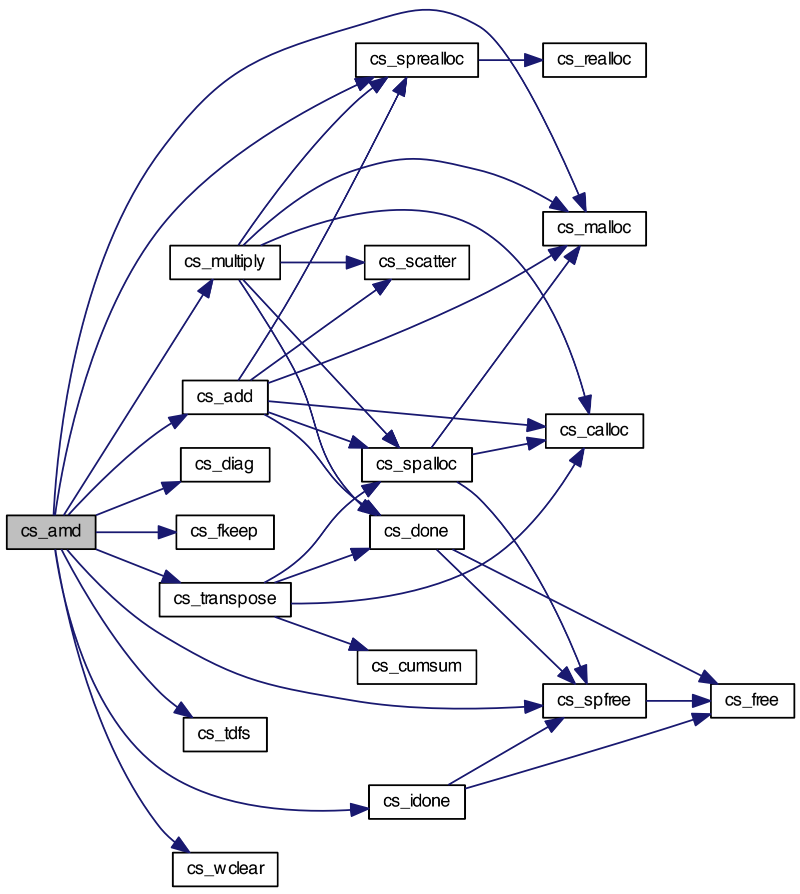
8.2.1.4 int∗ cs_amd (const cs ∗ *A,* int *order*)

Definition at line 79 of file csparse.c.

References cs_add(), cs_diag(), cs_fkeep(), CS_FLIP, cs_idone(), cs_malloc(), CS_MAX, CS_MIN, cs_multiply(), cs_spfree(), cs_sprealloc(), cs_tdfs(), cs_transpose(), cs_wclear(), cs_sparse::i, cs_sparse::m, cs_sparse::n, cs_-sparse::nzmax, and cs_sparse::p.

Referenced by cs_schol(), and cs_sqr().

Here is the call graph for this function:



8.2.1.5 static int cs_ereach (const cs ∗ *A,* int *k,* const int ∗ *parent,* int ∗ *s,* int ∗ *w,* double ∗ *x,* int *top*) [static]

Definition at line 453 of file csparse.c.

References cs_sparse::i, cs_sparse::p, and cs_sparse::x.

Referenced by cs_chol().

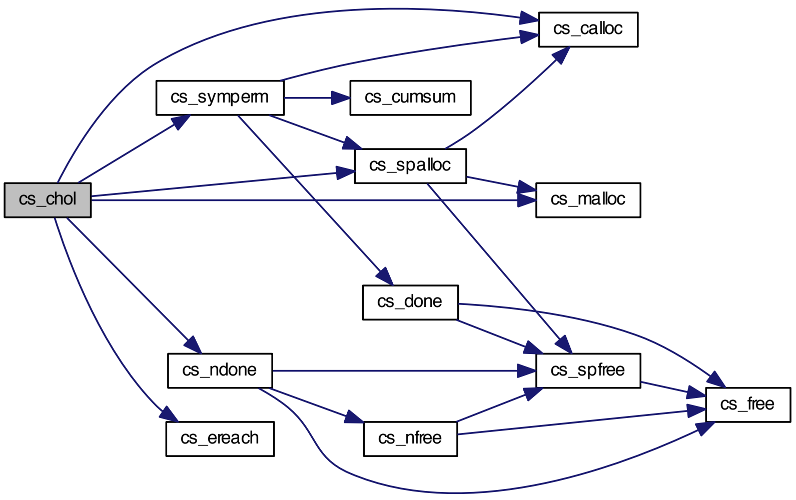
8.2.1.6 csn∗ cs_chol (const cs ∗ *A*, const css ∗ *S*)

Definition at line 474 of file csparse.c.

References cs_symbolic::cp, cs_calloc(), cs_ereach(), cs_malloc(), cs_ndone(), cs_spalloc(), cs_symperm(), cs_-sparse::i, cs_numeric::L, cs_sparse::n, cs_sparse::p, cs_symbolic::parent, cs_symbolic::Pinv, and cs_sparse::x.

Referenced by cs_cholsol().

Here is the call graph for this function:

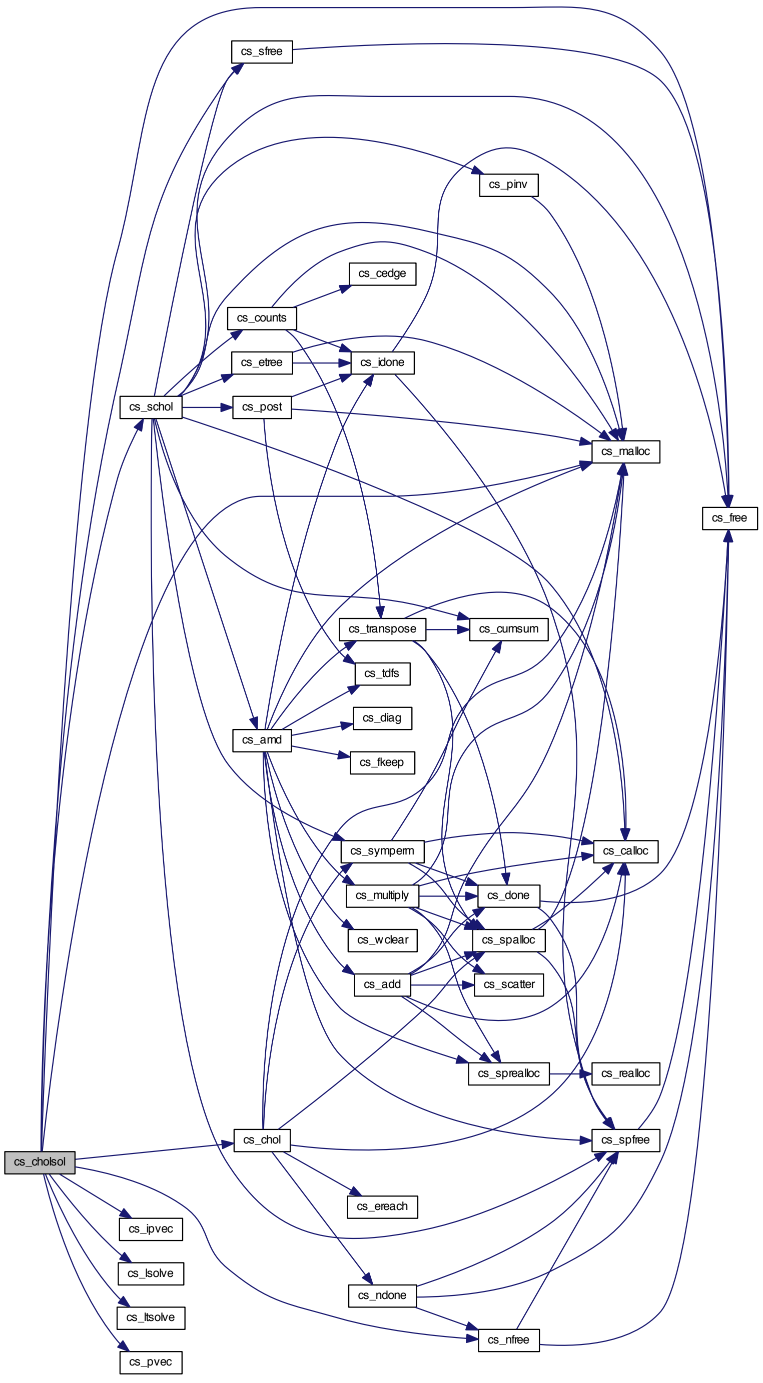


8.2.1.7 int cs_cholsol (const cs ∗ *A*, double ∗ *b*, int *order*)

Definition at line 533 of file csparse.c.

References cs_chol(), cs_free(), cs_ipvec(), cs_lsolve(), cs_ltsolve(), cs_malloc(), cs_nfree(), cs_pvec(), cs_schol(), cs_sfree(), cs_numeric::L, cs_sparse::n, and cs_symbolic::Pinv.

Here is the call graph for this function:



8.2.1.8 static void cs_cedge (int *j*, int *i*, const int ∗ *first*, int ∗ *maxfirst*, int ∗ *delta*, int ∗ *prevleaf*, int ∗ *ancestor*) [static]

Definition at line 559 of file csparse.c.

Referenced by cs_counts().

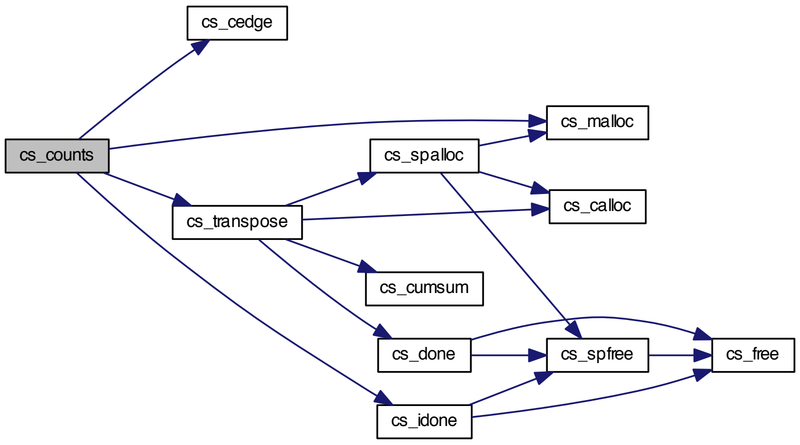
8.2.1.9 int∗ cs_counts (const cs ∗ *A,* const int ∗ *parent,* const int ∗ *post,* int *ata*)

Definition at line 582 of file csparse.c.

References cs_cedge(), cs_idone(), cs_malloc(), CS_MIN, cs_transpose(), cs_sparse::i, cs_sparse::m, cs_sparse::n, and cs_sparse::p.

Referenced by cs_schol(), and cs_sqr().

Here is the call graph for this function:



8.2.1.10 int cs_cumsum (int ∗ *p,* int ∗ *c,* int *n*)

Definition at line 651 of file csparse.c.

Referenced by cs_schol(), cs_symperm(), cs_transpose(), and cs_triplet().

8.2.1.11 int cs_dfs (int *j,* cs ∗ *L,* int *top,* int ∗ *xi,* int ∗ *pstack,* const int ∗ *Pinv*)

Definition at line 666 of file csparse.c.

References CS_MARK, CS_MARKED, CS_UNFLIP, cs_sparse::i, and cs_sparse::p.

Referenced by cs_reach(), and cs_scc().

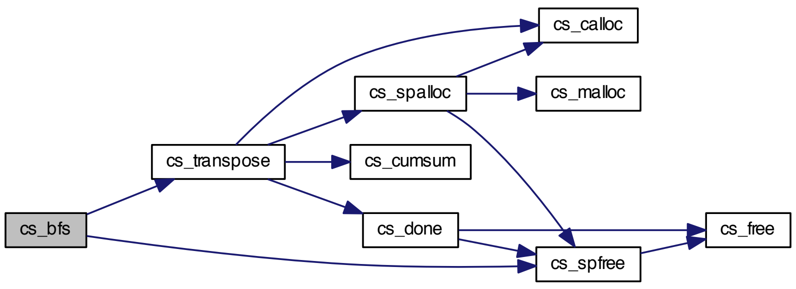
8.2.1.12 static int cs_bfs (const cs ∗ *A,* int *n,* int ∗ *wi,* int ∗ *wj,* int ∗ *queue,* const int ∗ *imatch,* const int ∗ *jmatch,* int *mark*) [static]

Definition at line 703 of file csparse.c.

References cs_spfree(), cs_transpose(), cs_sparse::i, and cs_sparse::p.

Referenced by cs_dmperm().

Here is the call graph for this function:



8.2.1.13 static void cs_matched (int *m*, const int ∗ *wi*, const int ∗ *jmatch*, int ∗ *P*, int ∗ *Q*, int ∗ *cc*, int ∗ *rr*, int *set*, int *mark*) `[static]`

Definition at line 738 of file csparse.c.

Referenced by cs_dmperm().

8.2.1.14 static void cs_unmatched (int *m*, const int ∗ *wi*, int ∗ *P*, int ∗ *rr*, int *set*) `[static]`

Definition at line 754 of file csparse.c.

Referenced by cs_dmperm().

8.2.1.15 static int cs_rprune (int *i*, int *j*, double *aij*, void ∗ *other*) `[static]`

Definition at line 767 of file csparse.c.

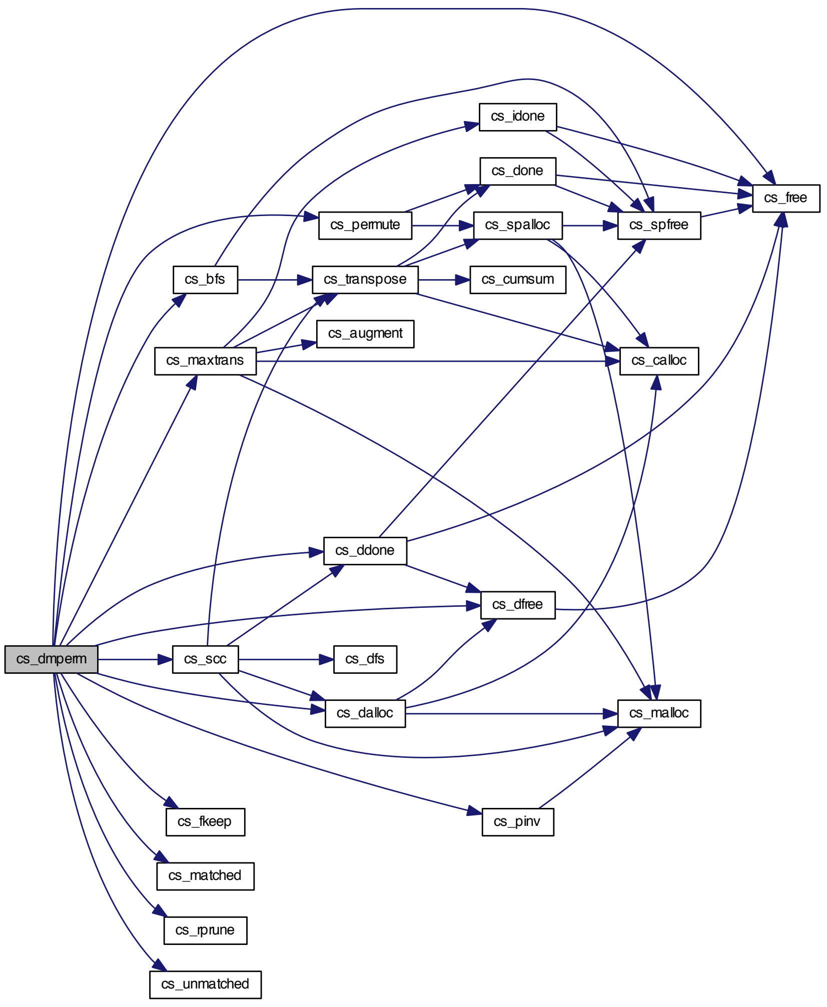
Referenced by cs_dmperm().

8.2.1.16 csd∗ cs_dmperm (const cs ∗ *A*)

Definition at line 774 of file csparse.c.

References cs_dmperm_results::cc, cs_bfs(), cs_dalloc(), cs_ddone(), cs_dfree(), cs_fkeep(), cs_free(), cs_-matched(), cs_maxtrans(), cs_permute(), cs_pinv(), cs_rprune(), cs_scc(), cs_unmatched(), cs_sparse::i, cs_-sparse::m, cs_sparse::n, cs_dmperm_results::nb, cs_sparse::p, cs_dmperm_results::P, cs_dmperm_results::Q, cs-_dmperm_results::R, cs_dmperm_results::rr, and cs_dmperm_results::S.

Here is the call graph for this function:



8.2.1.17 static int cs_tol (int *i*, int *j*, double *aij*, void ∗ *tol*) [static]

Definition at line 860 of file csparse.c.

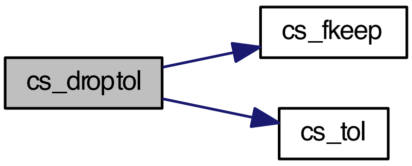
Referenced by cs_droptol().

8.2.1.18 int cs_droptol (cs ∗ *A*, double *tol*)

Definition at line 864 of file csparse.c.

References cs_fkeep(), and cs_tol().

Here is the call graph for this function:



8.2.1.19 static int cs_nonzero (int *i*, int *j*, double *aij*, void ∗ *other*) [static]

Definition at line 869 of file csparse.c.

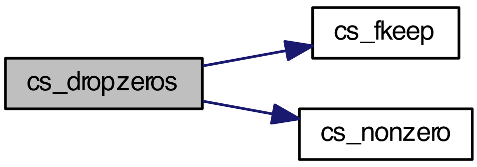
Referenced by cs_dropzeros().

8.2.1.20 int cs_dropzeros (cs ∗ *A*)

Definition at line 873 of file csparse.c.

References cs_fkeep(), and cs_nonzero().

Here is the call graph for this function:

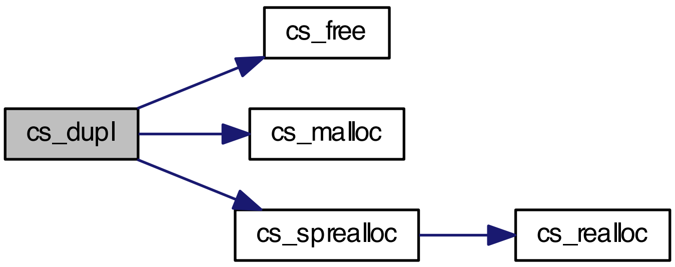


8.2.1.21 int cs_dupl (cs ∗ *A*)

Definition at line 877 of file csparse.c.

References cs_free(), cs_malloc(), cs_sprealloc(), cs_sparse::i, cs_sparse::m, cs_sparse::n, cs_sparse::p, and cs-_sparse::x.

Here is the call graph for this function:



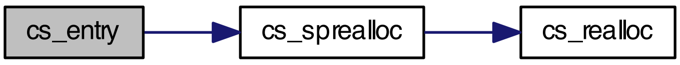
8.2.1.22 int cs_entry (cs ∗ *T,* int *i,* int *j,* double *x*)

Definition at line 926 of file csparse.c.

References CS_MAX, cs_sprealloc(), cs_sparse::i, cs_sparse::m, cs_sparse::n, cs_sparse::nz, cs_sparse::nzmax, cs_sparse::p, and cs_sparse::x.

Referenced by cs_load().

Here is the call graph for this function:



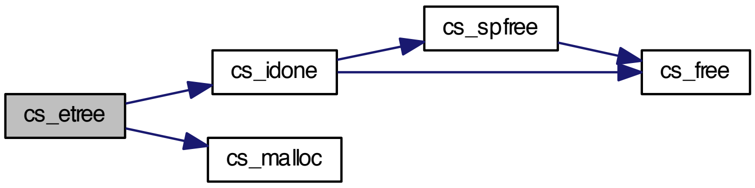
8.2.1.23 int∗ cs_etree (const cs ∗ *A,* int *ata*)

Definition at line 938 of file csparse.c.

References cs_idone(), cs_malloc(), cs_sparse::i, cs_sparse::m, cs_sparse::n, and cs_sparse::p.

Referenced by cs_schol(), and cs_sqr().

Here is the call graph for this function:



8.2.1.24 int cs_fkeep (cs ∗ *A*, int(∗)(int, int, double, void ∗) *fkeep*, void ∗ *other*)

Definition at line 972 of file csparse.c.

References cs_sparse::i, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_amd(), cs_dmperm(), cs_droptol(), and cs_dropzeros().

8.2.1.25 int cs_gaxpy (const cs ∗ *A*, const double ∗ *x*, double ∗ *y*)

Definition at line 998 of file csparse.c.

References cs_sparse::i, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by fclib_merit_local().

8.2.1.26 int cs_happly (const cs ∗ *V*, int *i*, double *beta*, double ∗ *x*)

Definition at line 1018 of file csparse.c.

References cs_sparse::i, cs_sparse::p, and cs_sparse::x.

Referenced by cs_qr(), and cs_qrsol().

8.2.1.27 double cs_house (double ∗ *x*, double ∗ *beta*, int *n*)

Definition at line 1040 of file csparse.c.

Referenced by cs_qr().

8.2.1.28 int cs_ipvec (int *n*, const int ∗ *P*, const double ∗ *b*, double ∗ *x*)

Definition at line 1062 of file csparse.c.

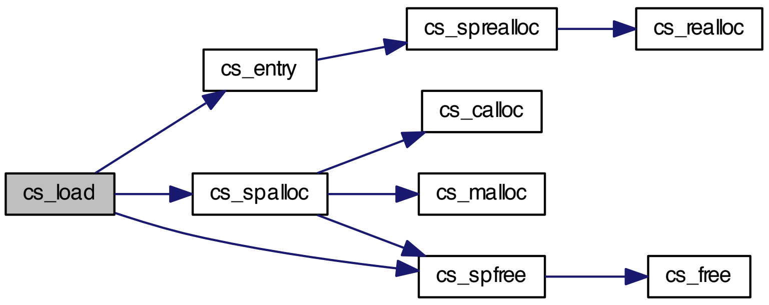
Referenced by cs_cholsol(), cs_lusol(), and cs_qrsol().

8.2.1.29 cs∗ cs_load (FILE ∗ *f*)

Definition at line 1069 of file csparse.c.

References cs_entry(), cs_spalloc(), and cs_spfree().

Here is the call graph for this function:



8.2.1.30 int cs_lsolve (const cs ∗ *L*, double ∗ *x*)

Definition at line 1093 of file csparse.c.

References cs_sparse::i, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_cholsol(), and cs_lusol().

8.2.1.31 int cs_ltsolve (const cs ∗ *L*, double ∗ *x*)

Definition at line 1127 of file csparse.c.

References cs_sparse::i, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_cholsol().

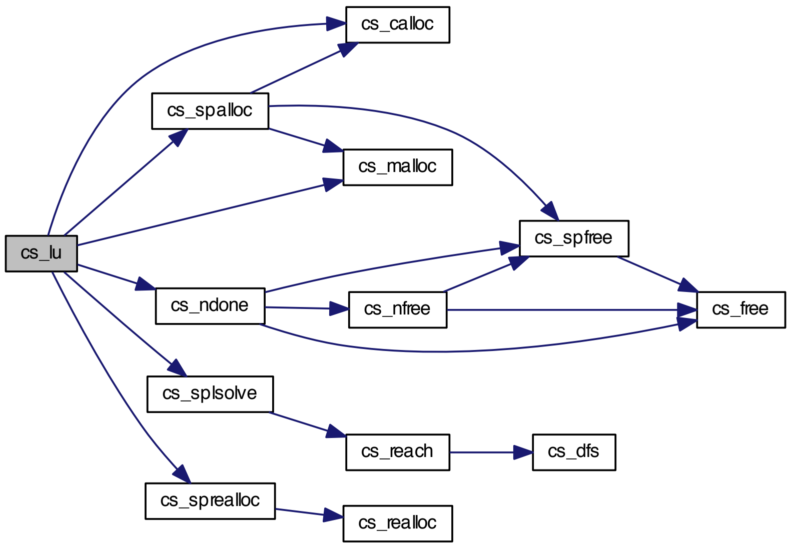
8.2.1.32 csn∗ cs_lu (const cs ∗ *A*, const css ∗ *S*, double *tol*)

Definition at line 1163 of file csparse.c.

References cs_calloc(), cs_malloc(), cs_ndone(), cs_spalloc(), cs_splsolve(), cs_sprealloc(), cs_sparse::i, cs_-numeric::L, cs_symbolic::lnz, cs_sparse::n, cs_sparse::nzmax, cs_sparse::p, cs_numeric::Pinv, cs_symbolic::Q, cs-_numeric::U, cs_symbolic::unz, and cs_sparse::x.

Referenced by cs_lusol().

Here is the call graph for this function:

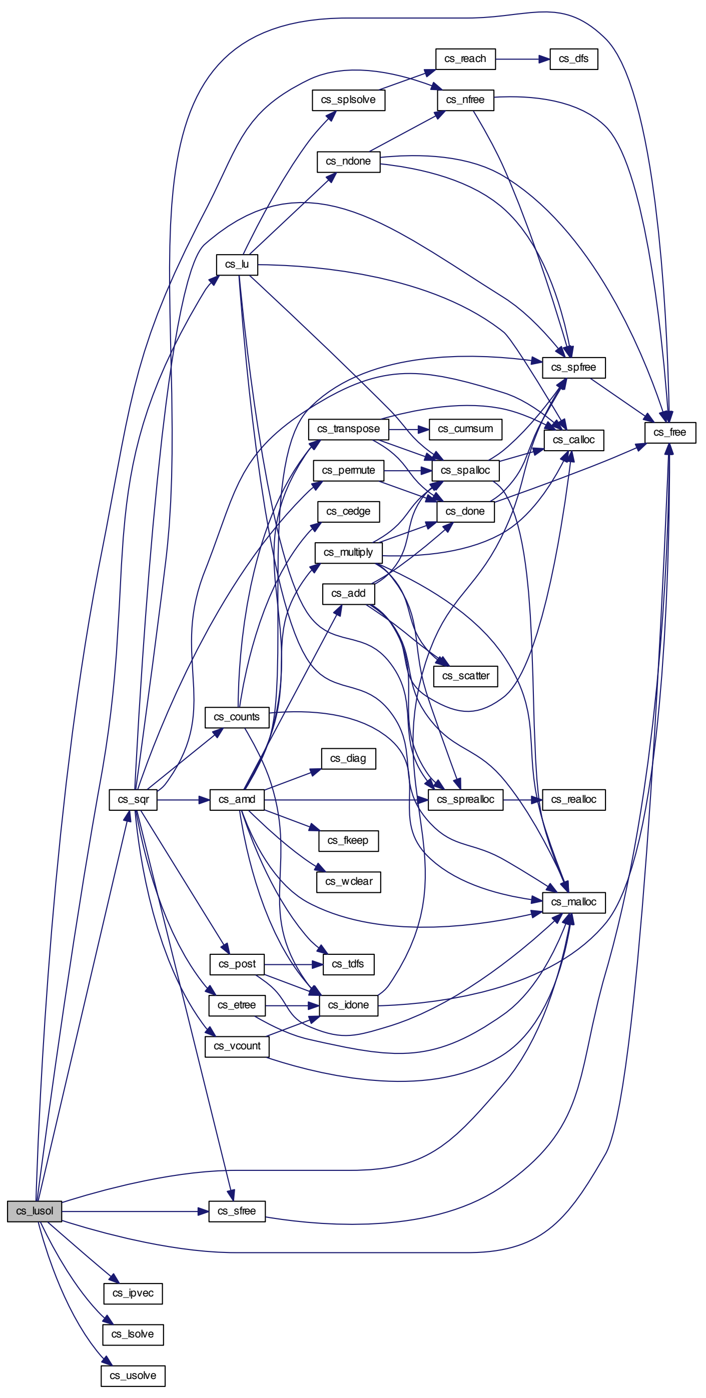


8.2.1.33 int cs_lusol (const cs ∗ *A,* double ∗ *b,* int *order,* double *tol*)

Definition at line 1255 of file csparse.c.

References cs_free(), cs_ipvec(), cs_lsolve(), cs_lu(), cs_malloc(), cs_nfree(), cs_sfree(), cs_sqr(), cs_usolve(), cs_numeric::L, cs_sparse::n, cs_numeric::Pinv, cs_symbolic::Q, and cs_numeric::U.

Here is the call graph for this function:



8.2.1.34 void∗ cs_malloc (int *n*, size_t *size*)

Definition at line 1288 of file csparse.c.

References CS_MAX, and CS_OVERFLOW.

Referenced by cs_add(), cs_amd(), cs_chol(), cs_cholsol(), cs_counts(), cs_dalloc(), cs_dupl(), cs_etree(), cs_-lu(), cs_lusol(), cs_maxtrans(), cs_multiply(), cs_pinv(), cs_post(), cs_qr(), cs_scc(), cs_schol(), cs_spalloc(), cs_-updown(), and cs_vcount().

8.2.1.35 void∗ cs_calloc (int *n*, size_t *size*)

Definition at line 1294 of file csparse.c.

References CS_MAX, and CS_OVERFLOW.

Referenced by cs_add(), cs_chol(), cs_dalloc(), cs_lu(), cs_maxtrans(), cs_multiply(), cs_qr(), cs_qrsol(), cs_-schol(), cs_spalloc(), cs_sqr(), cs_symperm(), cs_transpose(), and cs_triplet().

8.2.1.36 void∗ cs_free (void ∗ *p*)

Definition at line 1300 of file csparse.c.

Referenced by cs_cholsol(), cs_ddone(), cs_dfree(), cs_dmperm(), cs_done(), cs_dupl(), cs_idone(), cs_lusol(), cs_ndone(), cs_nfree(), cs_qrsol(), cs_schol(), cs_sfree(), cs_spfree(), cs_sqr(), and cs_updown().

8.2.1.37 void∗ cs_realloc (void ∗ *p*, int *n*, size_t *size*, int ∗ *ok*)

Definition at line 1307 of file csparse.c.

References CS_MAX, and CS_OVERFLOW.

Referenced by cs_sprealloc().

8.2.1.38 static void cs_augment (int *k*, const cs ∗ *A*, int ∗ *jmatch*, int ∗ *cheap*, int ∗ *w*, int ∗ *js*, int ∗ *is*, int ∗ *ps*) [static]

Definition at line 1318 of file csparse.c.

References cs_sparse::i, and cs_sparse::p.

Referenced by cs_maxtrans().

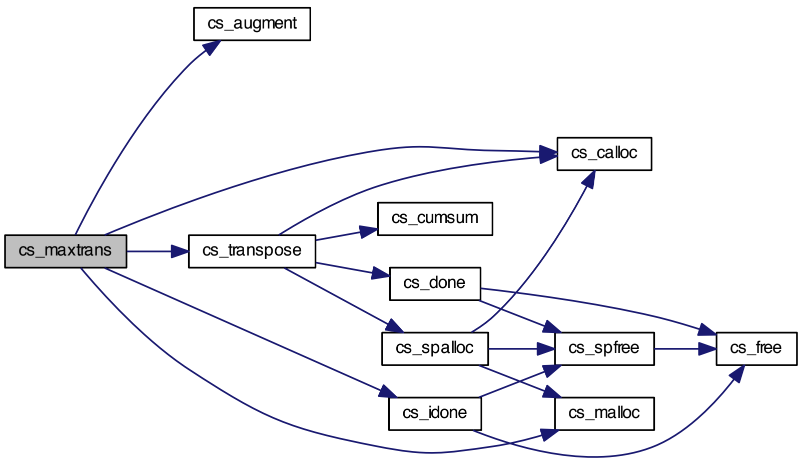
8.2.1.39 int∗ cs_maxtrans (const cs ∗ *A*)

Definition at line 1359 of file csparse.c.

References cs_augment(), cs_calloc(), cs_idone(), cs_malloc(), cs_transpose(), cs_sparse::i, cs_sparse::m, cs_-sparse::n, and cs_sparse::p.

Referenced by cs_dmperm().

Here is the call graph for this function:



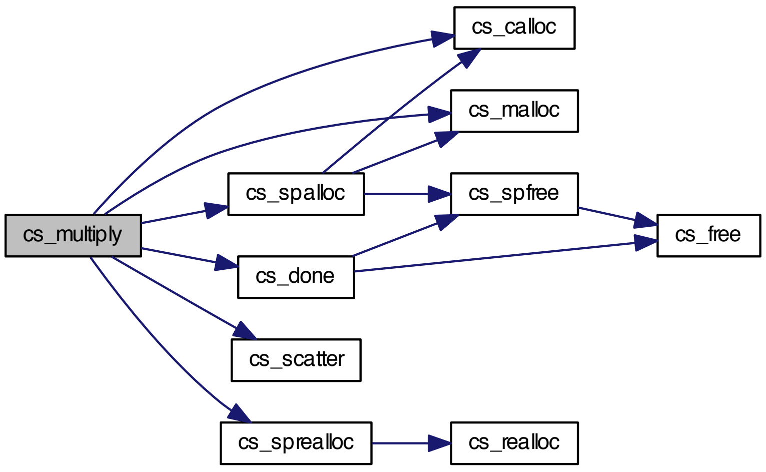
8.2.1.40 cs∗ cs_multiply (const cs ∗ *A,* const cs ∗ *B*)

Definition at line 1400 of file csparse.c.

References cs_calloc(), cs_done(), cs_malloc(), cs_scatter(), cs_spalloc(), cs_sprealloc(), cs_sparse::i, cs_sparse::m, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_amd().

Here is the call graph for this function:



8.2.1.41 double cs_norm (const cs ∗ *A*)

Definition at line 1440 of file csparse.c.

References CS_MAX, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_print().

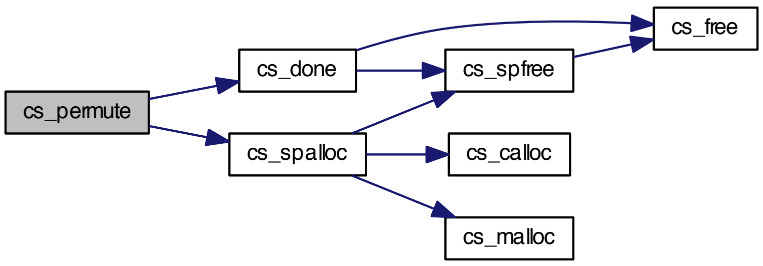
8.2.1.42 cs∗ cs_permute (const cs ∗ *A,* const int ∗ *Pinv,* const int ∗ *Q,* int *values*)

Definition at line 1457 of file csparse.c.

References cs_done(), cs_spalloc(), cs_sparse::i, cs_sparse::m, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_dmperm(), and cs_sqr().

Here is the call graph for this function:



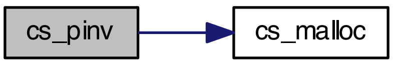
8.2.1.43 int∗ cs_pinv (int const ∗ *P*, int *n*)

Definition at line 1488 of file csparse.c.

References cs_malloc().

Referenced by cs_dmperm(), and cs_schol().

Here is the call graph for this function:



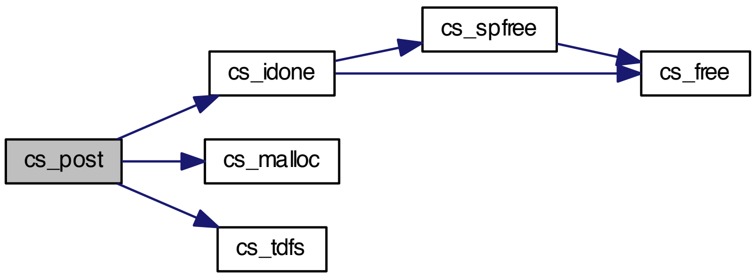
8.2.1.44 int∗ cs_post (int *n*, const int ∗ *parent*)

Definition at line 1499 of file csparse.c.

References cs_idone(), cs_malloc(), and cs_tdfs().

Referenced by cs_schol(), and cs_sqr().

Here is the call graph for this function:

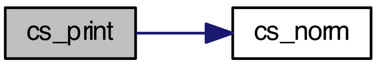


8.2.1.45 int cs_print (const cs ∗ *A,* int *brief*)

Definition at line 1525 of file csparse.c.

References CS_COPYRIGHT, CS_DATE, cs_norm(), CS_SUBSUB, CS_SUBVER, CS_VER, cs_sparse::i, cs_-sparse::m, cs_sparse::n, cs_sparse::nz, cs_sparse::nzmax, cs_sparse::p, and cs_sparse::x.

Here is the call graph for this function:



8.2.1.46 int cs_pvec (int *n,* const int ∗ *P,* const double ∗ *b,* double ∗ *x*)

Definition at line 1578 of file csparse.c.

Referenced by cs_cholsol(), and cs_qrsol().

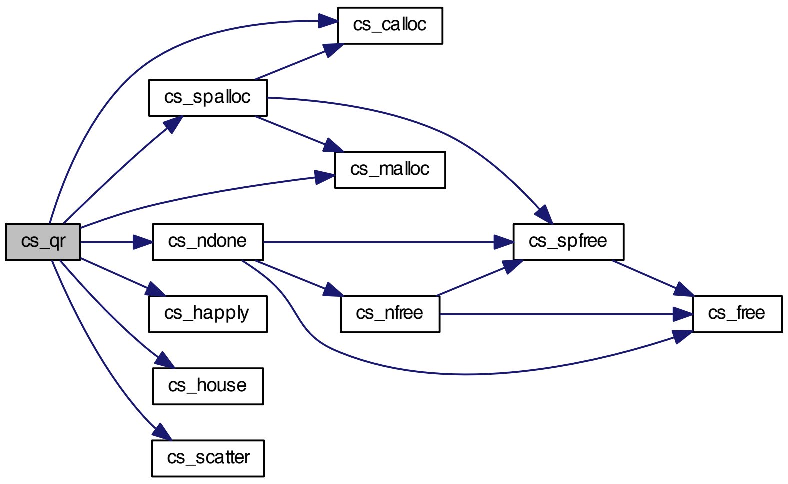
8.2.1.47 csn∗ cs_qr (const cs ∗ *A,* const css ∗ *S*)

Definition at line 1587 of file csparse.c.

References cs_numeric::B, cs_calloc(), cs_happly(), cs_house(), cs_malloc(), cs_ndone(), cs_scatter(), cs_-spalloc(), cs_sparse::i, cs_numeric::L, cs_symbolic::lnz, cs_sparse::m, cs_symbolic::m2, cs_sparse::n, cs_sparse-::p, cs_symbolic::parent, cs_symbolic::Pinv, cs_symbolic::Q, cs_numeric::U, cs_symbolic::unz, and cs_sparse::x.

Referenced by cs_qrsol().

Here is the call graph for this function:

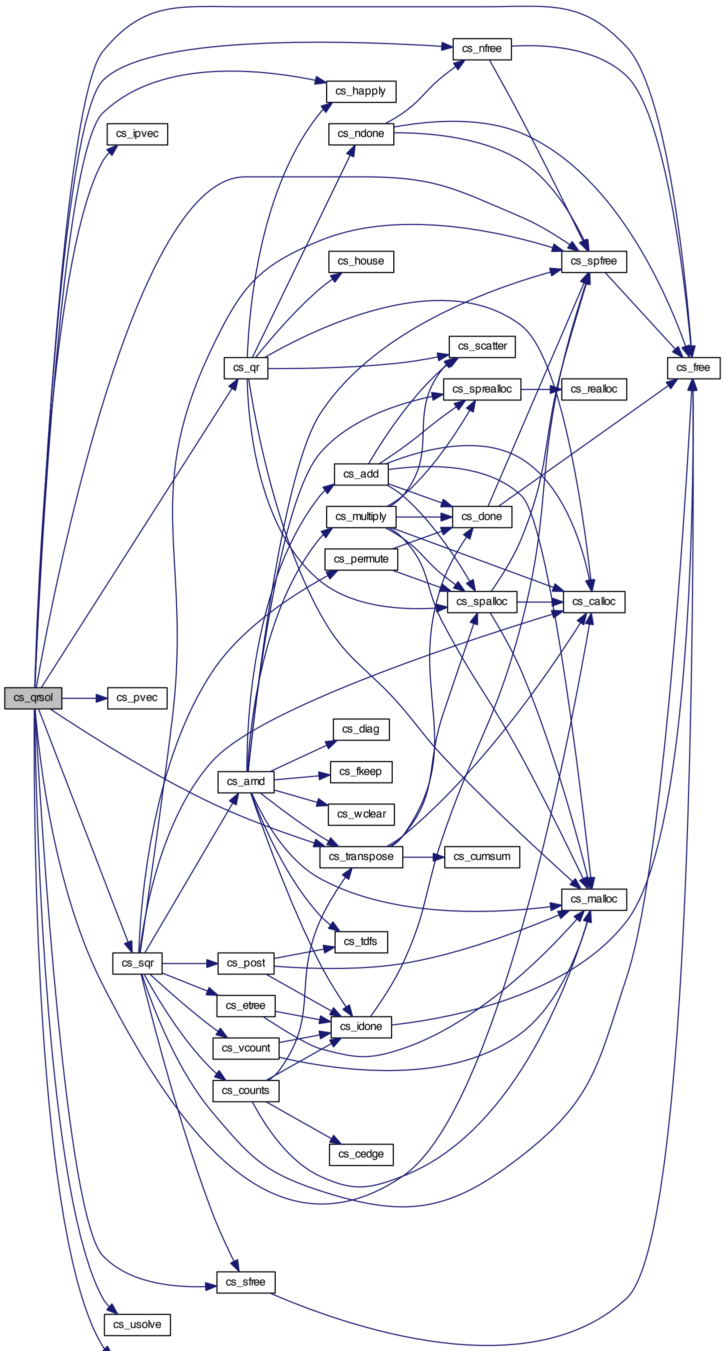


8.2.1.48 int cs_qrsol (const cs ∗ *A*, double ∗ *b*, int *order*)

Definition at line 1674 of file csparse.c.

References cs_numeric::B, cs_calloc(), cs_free(), cs_happly(), cs_ipvec(), cs_nfree(), cs_pvec(), cs_qr(), cs_-sfree(), cs_spfree(), cs_sqr(), cs_transpose(), cs_usolve(), cs_utsolve(), cs_numeric::L, cs_sparse::m, cs_symbolic-::m2, cs_sparse::n, cs_symbolic::Pinv, cs_symbolic::Q, and cs_numeric::U.

Here is the call graph for this function:



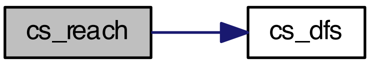
8.2.1.49 int cs_reach (cs ∗ *L,* const cs ∗ *B,* int *k,* int ∗ *xi,* const int ∗ *Pinv*)

Definition at line 1728 of file csparse.c.

References cs_dfs(), CS_MARK, CS_MARKED, cs_sparse::i, cs_sparse::n, and cs_sparse::p.

Referenced by cs_splsolve().

Here is the call graph for this function:



8.2.1.50 int cs_scatter (const cs ∗ *A,* int *j,* double *beta,* int ∗ *w,* double ∗ *x,* int *mark,* cs ∗ *C,* int *nz*)

Definition at line 1749 of file csparse.c.

References cs_sparse::i, cs_sparse::p, and cs_sparse::x.

Referenced by cs_add(), cs_multiply(), and cs_qr().

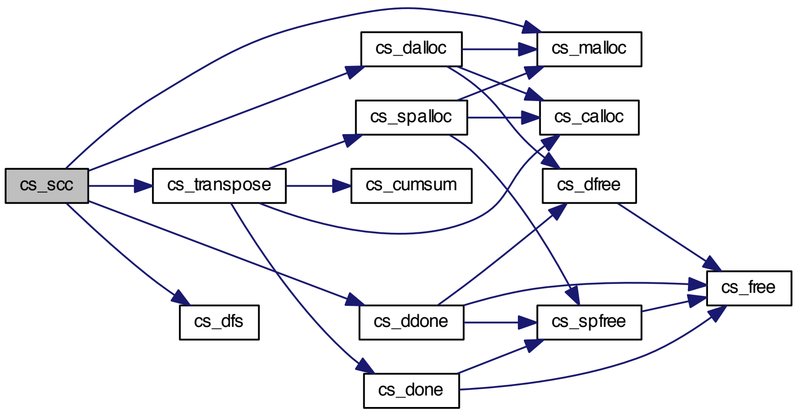
8.2.1.51 csd∗ cs_scc (cs ∗ *A*)

Definition at line 1774 of file csparse.c.

References cs_dalloc(), cs_ddone(), cs_dfs(), cs_malloc(), CS_MARK, CS_MARKED, cs_transpose(), cs_sparse-::n, cs_dmperm_results::nb, cs_sparse::p, cs_dmperm_results::P, and cs_dmperm_results::R.

Referenced by cs_dmperm().

Here is the call graph for this function:



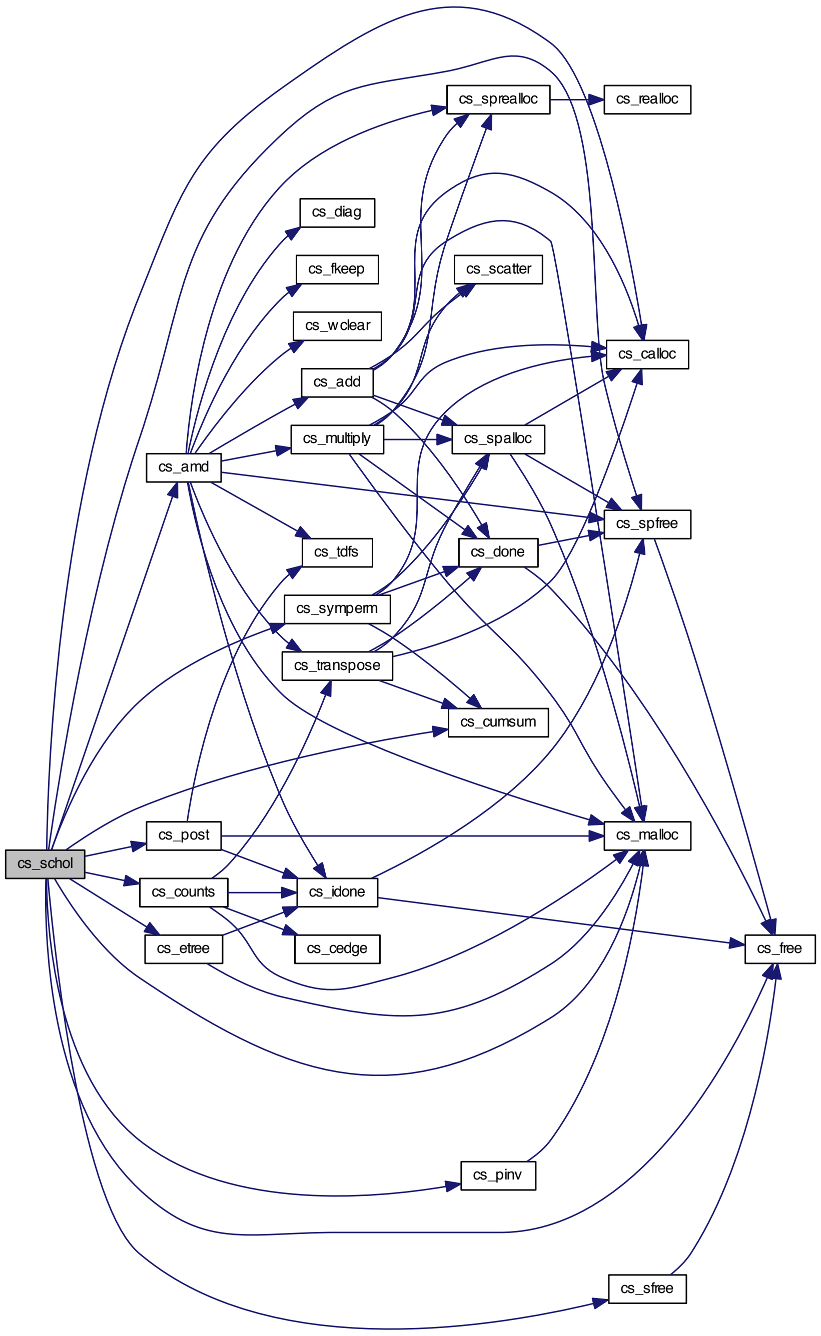
8.2.1.52 css∗ cs_schol (const cs ∗ *A,* int *order*)

Definition at line 1812 of file csparse.c.

References cs_symbolic::cp, cs_amd(), cs_calloc(), cs_counts(), cs_cumsum(), cs_etree(), cs_free(), cs_malloc(), cs_pinv(), cs_post(), cs_sfree(), cs_spfree(), cs_symperm(), cs_symbolic::lnz, cs_sparse::n, cs_symbolic::parent, cs_symbolic::Pinv, and cs_symbolic::unz.

Referenced by cs_cholsol().

Here is the call graph for this function:



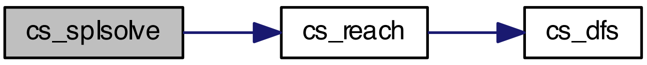
8.2.1.53 int cs_splsolve (cs * *L,* const cs * *B,* int *k,* int * *xi,* double * *x,* const int * *Pinv*)

Definition at line 1838 of file csparse.c.

References cs_reach(), cs_sparse::i, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_lu().

Here is the call graph for this function:



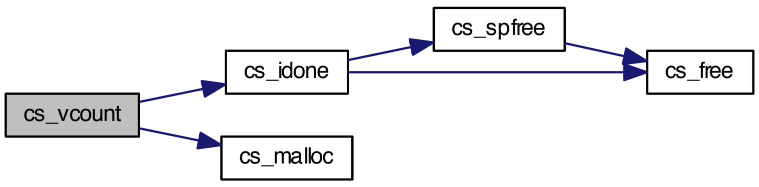
8.2.1.54 static int∗ cs_vcount (const cs ∗ *A*, const int ∗ *parent*, int ∗ *m2*, int ∗ *vnz*) [static]

Definition at line 1867 of file csparse.c.

References cs_idone(), cs_malloc(), cs_sparse::i, cs_sparse::m, cs_sparse::n, and cs_sparse::p.

Referenced by cs_sqr().

Here is the call graph for this function:



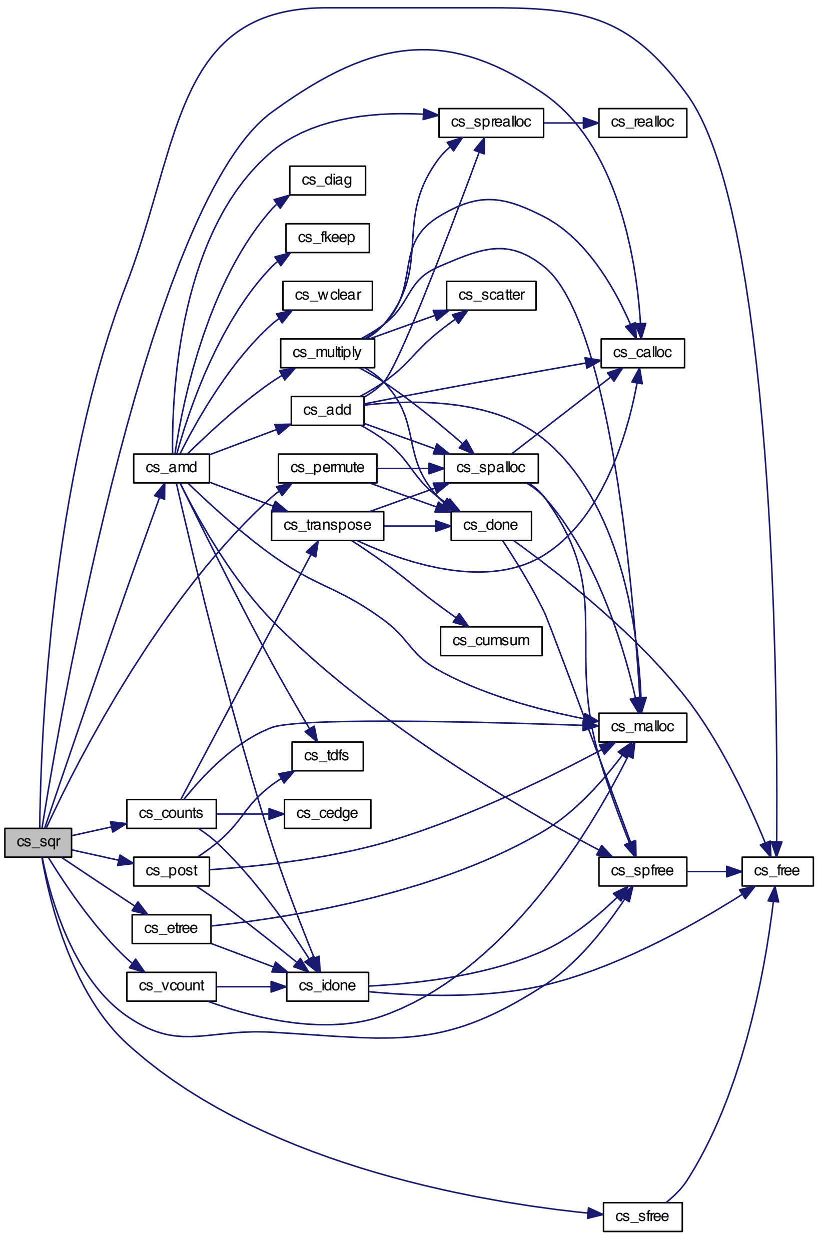
8.2.1.55 css∗ cs_sqr (const cs ∗ *A*, int *order*, int *qr*)

Definition at line 1917 of file csparse.c.

References cs_symbolic::cp, cs_amd(), cs_calloc(), cs_counts(), cs_etree(), cs_free(), cs_permute(), cs_post(), cs_sfree(), cs_spfree(), cs_vcount(), cs_symbolic::lnz, cs_symbolic::m2, cs_sparse::n, cs_sparse::p, cs_symbolic::parent, cs_symbolic::Pinv, cs_symbolic::Q, and cs_symbolic::unz.

Referenced by cs_lusol(), and cs_qrsol().

Here is the call graph for this function:



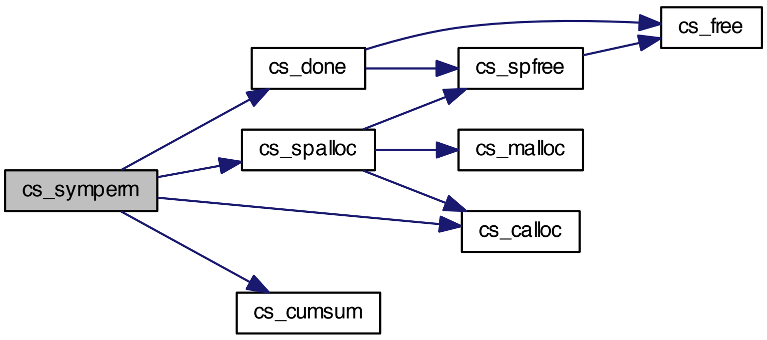
8.2.1.56 cs∗ cs_symperm (const cs ∗ *A*, const int ∗ *Pinv*, int *values*)

Definition at line 1949 of file csparse.c.

References cs_calloc(), cs_cumsum(), cs_done(), CS_MAX, CS_MIN, cs_spalloc(), cs_sparse::i, cs_sparse::n, cs-_sparse::p, and cs_sparse::x.

Referenced by cs_chol(), and cs_schol().

Here is the call graph for this function:



8.2.1.57 int cs_tdfs (int *j,* int *k,* int ∗ *head,* const int ∗ *next,* int ∗ *post,* int ∗ *stack*)

Definition at line 1993 of file csparse.c.

Referenced by cs_amd(), and cs_post().

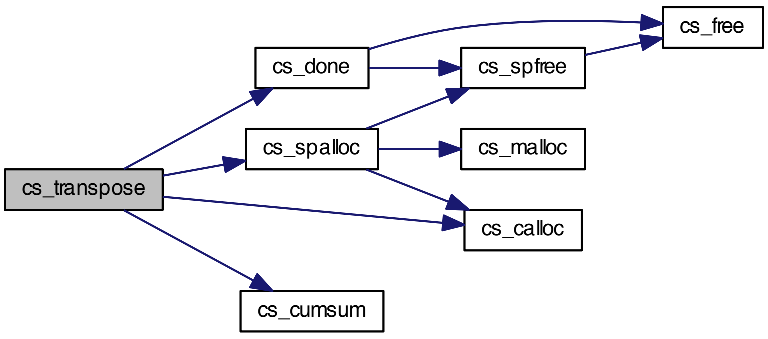
8.2.1.58 cs∗ cs_transpose (const cs ∗ *A,* int *values*)

Definition at line 2017 of file csparse.c.

References cs_calloc(), cs_cumsum(), cs_done(), cs_spalloc(), cs_sparse::i, cs_sparse::m, cs_sparse::n, cs_-sparse::p, and cs_sparse::x.

Referenced by cs_amd(), cs_bfs(), cs_counts(), cs_maxtrans(), cs_qrsol(), cs_scc(), and fclib_merit_local().

Here is the call graph for this function:

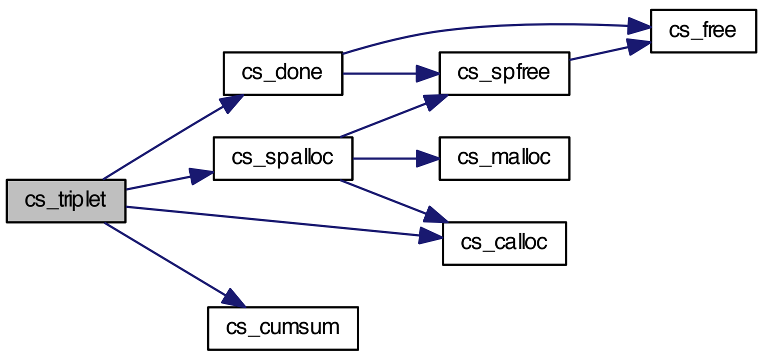


8.2.1.59 cs∗ cs_triplet (const cs ∗ *T*)

Definition at line 2048 of file csparse.c.

References cs_calloc(), cs_cumsum(), cs_done(), cs_spalloc(), cs_sparse::i, cs_sparse::m, cs_sparse::n, cs_-sparse::nz, cs_sparse::p, and cs_sparse::x.

Here is the call graph for this function:

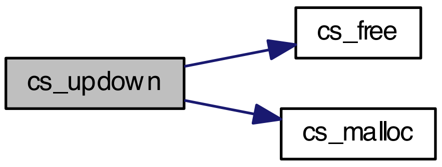


8.2.1.60 int cs_updown (cs ∗ *L,* int *sigma,* const cs ∗ *C,* const int ∗ *parent*)

Definition at line 2077 of file csparse.c.

References cs_free(), cs_malloc(), CS_MIN, cs_sparse::i, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Here is the call graph for this function:



8.2.1.61 int cs_usolve (const cs ∗ *U,* double ∗ *x*)

Definition at line 2119 of file csparse.c.

References cs_sparse::i, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_lusol(), and cs_qrsol().

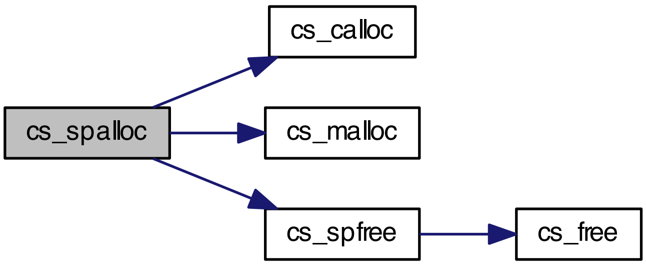
8.2.1.62 cs∗ cs_spalloc (int *m,* int *n,* int *nzmax,* int *values,* int *triplet*)

Definition at line 2140 of file csparse.c.

References cs_calloc(), cs_malloc(), CS_MAX, cs_spfree(), cs_sparse::i, cs_sparse::m, cs_sparse::n, cs_sparse::nz, cs_sparse::nzmax, cs_sparse::p, and cs_sparse::x.

Referenced by cs_add(), cs_chol(), cs_load(), cs_lu(), cs_multiply(), cs_permute(), cs_qr(), cs_symperm(), cs_transpose(), and cs_triplet().

Here is the call graph for this function:



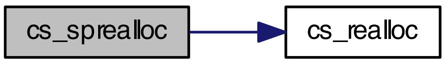
8.2.1.63 int cs_sprealloc (cs ∗ *A*, int *nzmax*)

Definition at line 2155 of file csparse.c.

References cs_realloc(), cs_sparse::i, cs_sparse::n, cs_sparse::nz, cs_sparse::nzmax, cs_sparse::p, and cs_-sparse::x.

Referenced by cs_add(), cs_amd(), cs_dupl(), cs_entry(), cs_lu(), and cs_multiply().

Here is the call graph for this function:



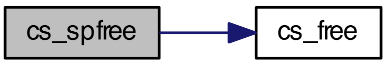
8.2.1.64 cs∗ cs_spfree (cs ∗ *A*)

Definition at line 2169 of file csparse.c.

References cs_free(), cs_sparse::i, cs_sparse::p, and cs_sparse::x.

Referenced by cs_amd(), cs_bfs(), cs_ddone(), cs_done(), cs_idone(), cs_load(), cs_ndone(), cs_nfree(), cs_-qrsol(), cs_schol(), cs_spalloc(), and cs_sqr().

Here is the call graph for this function:



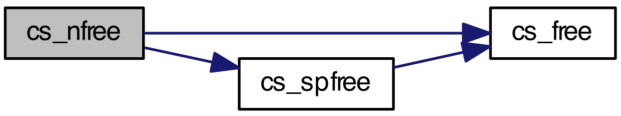
8.2.1.65 csn∗ cs_nfree (csn ∗ *N*)

Definition at line 2179 of file csparse.c.

References cs_numeric::B, cs_free(), cs_spfree(), cs_numeric::L, cs_numeric::Pinv, and cs_numeric::U.

Referenced by cs_cholsol(), cs_lusol(), cs_ndone(), and cs_qrsol().

Here is the call graph for this function:



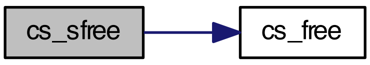
8.2.1.66 css∗ cs_sfree (css ∗ *S*)

Definition at line 2190 of file csparse.c.

References cs_symbolic::cp, cs_free(), cs_symbolic::parent, cs_symbolic::Pinv, and cs_symbolic::Q.

Referenced by cs_cholsol(), cs_lusol(), cs_qrsol(), cs_schol(), and cs_sqr().

Here is the call graph for this function:



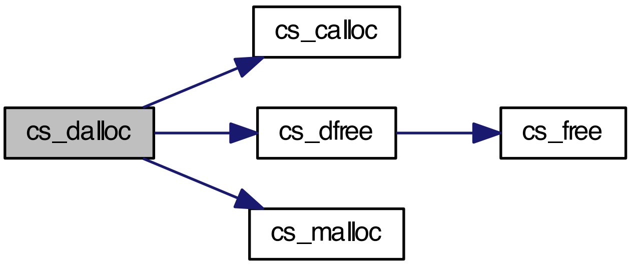
8.2.1.67 csd∗ cs_dalloc (int *m,* int *n*)

Definition at line 2201 of file csparse.c.

References cs_calloc(), cs_dfree(), cs_malloc(), cs_dmperm_results::P, cs_dmperm_results::Q, cs_dmperm_-results::R, and cs_dmperm_results::S.

Referenced by cs_dmperm(), and cs_scc().

Here is the call graph for this function:



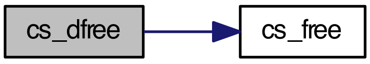
8.2.1.68 csd∗ cs_dfree (csd ∗ *D*)

Definition at line 2214 of file csparse.c.

References cs_free(), cs_dmperm_results::P, cs_dmperm_results::Q, cs_dmperm_results::R, and cs_dmperm_-results::S.

Referenced by cs_dalloc(), cs_ddone(), and cs_dmperm().

Here is the call graph for this function:



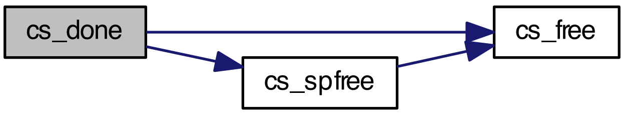
8.2.1.69 cs∗ cs_done (cs ∗ *C,* void ∗ *w,* void ∗ *x,* int *ok*)

Definition at line 2225 of file csparse.c.

References cs_free(), and cs_spfree().

Referenced by cs_add(), cs_multiply(), cs_permute(), cs_symperm(), cs_transpose(), and cs_triplet().

Here is the call graph for this function:



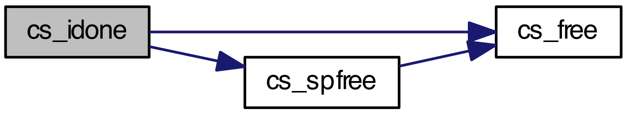
8.2.1.70 int∗ cs_idone (int ∗ *p*, cs ∗ *C*, void ∗ *w*, int *ok*)

Definition at line 2233 of file csparse.c.

References cs_free(), and cs_spfree().

Referenced by cs_amd(), cs_counts(), cs_etree(), cs_maxtrans(), cs_post(), and cs_vcount().

Here is the call graph for this function:



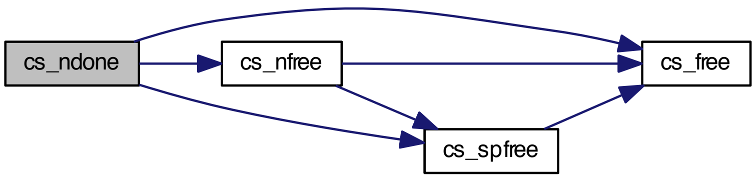
8.2.1.71 csn∗ cs_ndone (csn ∗ *N*, cs ∗ *C*, void ∗ *w*, void ∗ *x*, int *ok*)

Definition at line 2241 of file csparse.c.

References cs_free(), cs_nfree(), and cs_spfree().

Referenced by cs_chol(), cs_lu(), and cs_qr().

Here is the call graph for this function:



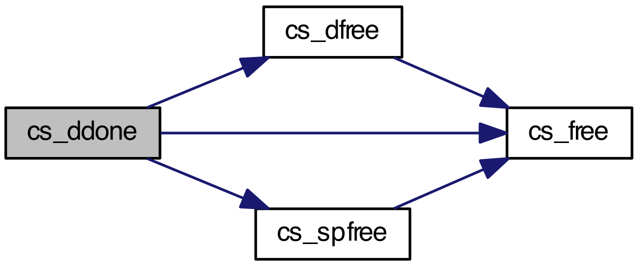
8.2.1.72 csd∗ cs_ddone (csd ∗ *D*, cs ∗ *C*, void ∗ *w*, int *ok*)

Definition at line 2250 of file csparse.c.

References cs_dfree(), cs_free(), and cs_spfree().

Referenced by cs_dmperm(), and cs_scc().

Here is the call graph for this function:



8.2.1.73 int cs_utsolve (const cs ∗ *U*, double ∗ *x*)

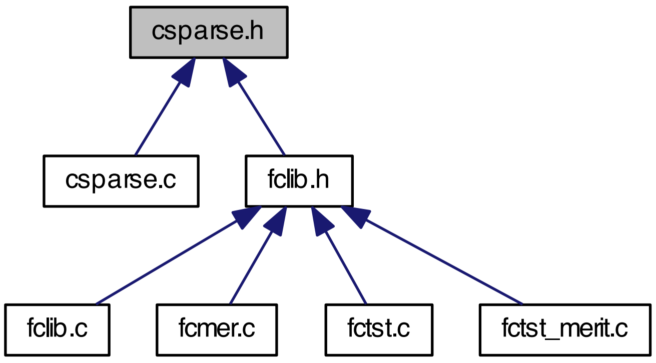
Definition at line 2258 of file csparse.c.

References cs_sparse::i, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_qrsol().

8.3 csparse.h File Reference

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



Classes

- struct cs_sparse
- struct cs_symbolic
- struct cs_numeric
- struct cs_dmperm_results

Macros

- #define CS_VER 1 /∗ CSparse Version 1.2.0 ∗/
- #define CS_SUBVER 2
- #define CS_SUBSUB 0
- #define CS_DATE "Mar 6, 2006" /∗ CSparse release date ∗/
- #define CS_COPYRIGHT "Copyright (c) Timothy A. Davis, 2006"
- #define CS_MAX(a, b) (((a) > (b)) ? (a) : (b))
- #define CS_MIN(a, b) (((a) < (b)) ? (a) : (b))
- #define CS_FLIP(i) (-(i)-2)
- #define CS_UNFLIP(i) (((i) < 0) ? CS_FLIP(i) : (i))
- #define CS_MARKED(Ap, j) (Ap [j] < 0)
- #define CS_MARK(Ap, j) { Ap [j] = CS_FLIP (Ap [j]) ; }
- #define CS_OVERFLOW(n, size) (n > INT_MAX / (int) size)

Typedefs

- typedef struct cs_sparse cs
- typedef struct cs_symbolic css
- typedef struct cs_numeric csn
- typedef struct cs_dmperm_results csd

Functions

- cs ∗ cs_add (const cs ∗A, const cs ∗B, double alpha, double beta)
- int cs_cholsol (const cs ∗A, double ∗b, int order)
- int cs_dupl (cs ∗A)
- int cs_entry (cs ∗T, int i, int j, double x)
- int cs_lusol (const cs ∗A, double ∗b, int order, double tol)
- int cs_gaxpy (const cs ∗A, const double ∗x, double ∗y)
- cs ∗ cs_multiply (const cs ∗A, const cs ∗B)
- int cs_qrsol (const cs ∗A, double ∗b, int order)
- cs ∗ cs_transpose (const cs ∗A, int values)
- cs ∗ cs_triplet (const cs ∗T)
- double cs_norm (const cs ∗A)
- int cs_print (const cs ∗A, int brief)
- cs ∗ cs_load (FILE ∗f)
- void ∗ cs_calloc (int n, size_t size)
- void ∗ cs_free (void ∗p)
- void ∗ cs_realloc (void ∗p, int n, size_t size, int ∗ok)
- cs ∗ cs_spalloc (int m, int n, int nzmax, int values, int triplet)
- cs ∗ cs_spfree (cs ∗A)
- int cs_sprealloc (cs ∗A, int nzmax)
- void ∗ cs_malloc (int n, size_t size)
- int ∗ cs_amd (const cs ∗A, int order)
- csn ∗ cs_chol (const cs ∗A, const css ∗S)
- csd ∗ cs_dmperm (const cs ∗A)
- int cs_droptol (cs ∗A, double tol)
- int cs_dropzeros (cs ∗A)
- int cs_happly (const cs ∗V, int i, double beta, double ∗x)
- int cs_ipvec (int n, const int ∗P, const double ∗b, double ∗x)
- int cs_lsolve (const cs ∗L, double ∗x)
- int cs_ltsolve (const cs ∗L, double ∗x)
- csn ∗ cs_lu (const cs ∗A, const css ∗S, double tol)

- cs ∗ cs_permute (const cs ∗A, const int ∗P, const int ∗Q, int values)
- int ∗ cs_pinv (const int ∗P, int n)
- int cs_pvec (int n, const int ∗P, const double ∗b, double ∗x)
- csn ∗ cs_qr (const cs ∗A, const css ∗S)
- css ∗ cs_schol (const cs ∗A, int order)
- css ∗ cs_sqr (const cs ∗A, int order, int qr)
- cs ∗ cs_symperm (const cs ∗A, const int ∗Pinv, int values)
- int cs_usolve (const cs ∗U, double ∗x)
- int cs_utsolve (const cs ∗U, double ∗x)
- int cs_updown (cs ∗L, int sigma, const cs ∗C, const int ∗parent)
- css ∗ cs_sfree (css ∗S)
- csn ∗ cs_nfree (csn ∗N)
- csd ∗ cs_dfree (csd ∗D)
- int ∗ cs_counts (const cs ∗A, const int ∗parent, const int ∗post, int ata)
- int cs_cumsum (int ∗p, int ∗c, int n)
- int cs_dfs (int j, cs ∗L, int top, int ∗xi, int ∗pstack, const int ∗Pinv)
- int ∗ cs_etree (const cs ∗A, int ata)
- int cs_fkeep (cs ∗A, int(∗fkeep)(int, int, double, void ∗), void ∗other)
- double cs_house (double ∗x, double ∗beta, int n)
- int ∗ cs_maxtrans (const cs ∗A)
- int ∗ cs_post (int n, const int ∗parent)
- int cs_reach (cs ∗L, const cs ∗B, int k, int ∗xi, const int ∗Pinv)
- csd ∗ cs_scc (cs ∗A)
- int cs_scatter (const cs ∗A, int j, double beta, int ∗w, double ∗x, int mark, cs ∗C, int nz)
- int cs_splsolve (cs ∗L, const cs ∗B, int k, int ∗xi, double ∗x, const int ∗Pinv)
- int cs_tdfs (int j, int k, int ∗head, const int ∗next, int ∗post, int ∗stack)
- csd ∗ cs_dalloc (int m, int n)
- cs ∗ cs_done (cs ∗C, void ∗w, void ∗x, int ok)
- int ∗ cs_idone (int ∗p, cs ∗C, void ∗w, int ok)
- csn ∗ cs_ndone (csn ∗N, cs ∗C, void ∗w, void ∗x, int ok)
- csd ∗ cs_ddone (csd ∗D, cs ∗C, void ∗w, int ok)

8.3.1 Macro Definition Documentation

8.3.1.1 #define CS_VER 1 /∗ CSparse Version 1.2.0 ∗/

Definition at line 7 of file csparse.h.

Referenced by cs_print().

8.3.1.2 #define CS_SUBVER 2

Definition at line 8 of file csparse.h.

Referenced by cs_print().

8.3.1.3 #define CS_SUBSUB 0

Definition at line 9 of file csparse.h.

Referenced by cs_print().

8.3.1.4 #define CS_DATE "Mar 6, 2006" /∗ CSparse release date ∗/

Definition at line 10 of file csparse.h.

Referenced by cs_print().

8.3.1.5 #define CS_COPYRIGHT "Copyright (c) Timothy A. Davis, 2006"

Definition at line 11 of file csparse.h.

Referenced by cs_print().

8.3.1.6 #define CS_MAX(*a,* *b*) (((a) $>$ (b)) ? (a) : (b))

Definition at line 127 of file csparse.h.

Referenced by cs_amd(), cs_calloc(), cs_entry(), cs_malloc(), cs_norm(), cs_realloc(), cs_spalloc(), and cs_-symperm().

8.3.1.7 #define CS_MIN(*a,* *b*) (((a) $<$ (b)) ? (a) : (b))

Definition at line 128 of file csparse.h.

Referenced by cs_amd(), cs_counts(), cs_symperm(), and cs_updown().

8.3.1.8 #define CS_FLIP(*i*) (-(i)-2)

Definition at line 129 of file csparse.h.

Referenced by cs_amd().

8.3.1.9 #define CS_UNFLIP(*i*) (((i) $<$ 0) ? CS_FLIP(i) : (i))

Definition at line 130 of file csparse.h.

Referenced by cs_dfs().

8.3.1.10 #define CS_MARKED(*Ap,* *j*) (Ap [j] $<$ 0)

Definition at line 131 of file csparse.h.

Referenced by cs_dfs(), cs_reach(), and cs_scc().

8.3.1.11 #define CS_MARK(*Ap,* *j*) { Ap [j] = CS_FLIP (Ap [j]) ; }

Definition at line 132 of file csparse.h.

Referenced by cs_dfs(), cs_reach(), and cs_scc().

8.3.1.12 #define CS_OVERFLOW(*n,* *size*) (n $>$ INT_MAX / (int) size)

Definition at line 133 of file csparse.h.

Referenced by cs_calloc(), cs_malloc(), and cs_realloc().

8.3.2 Typedef Documentation

8.3.2.1 typedef struct cs_sparse cs

8.3.2.2 typedef struct cs_symbolic css

8.3.2.3 typedef struct cs_numeric csn

8.3.2.4 typedef struct cs_dmperm_results csd

8.3.3 Function Documentation

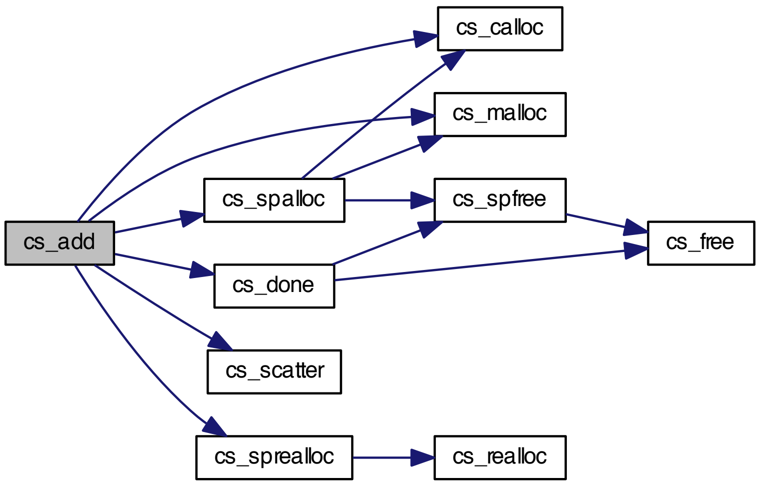
8.3.3.1 cs$*$ cs_add (const cs $*$ *A,* const cs $*$ *B,* double *alpha,* double *beta*)

Definition at line 8 of file csparse.c.

References cs_calloc(), cs_done(), cs_malloc(), cs_scatter(), cs_spalloc(), cs_sprealloc(), cs_sparse::m, cs_-sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_amd().

Here is the call graph for this function:

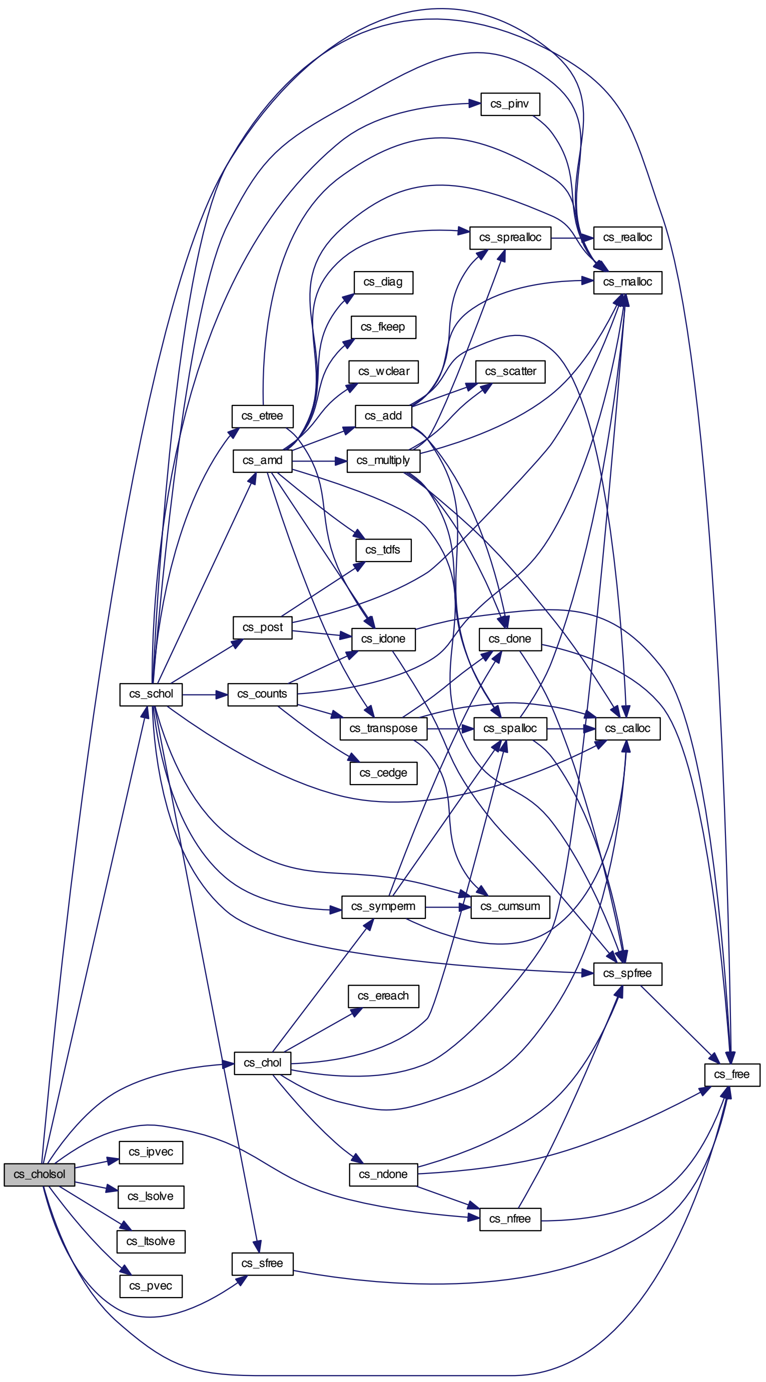


8.3.3.2 int cs_cholsol (const cs * *A*, double * *b*, int *order*)

Definition at line 533 of file csparse.c.

References cs_chol(), cs_free(), cs_ipvec(), cs_lsolve(), cs_ltsolve(), cs_malloc(), cs_nfree(), cs_pvec(), cs_schol(), cs_sfree(), cs_numeric::L, cs_sparse::n, and cs_symbolic::Pinv.

Here is the call graph for this function:

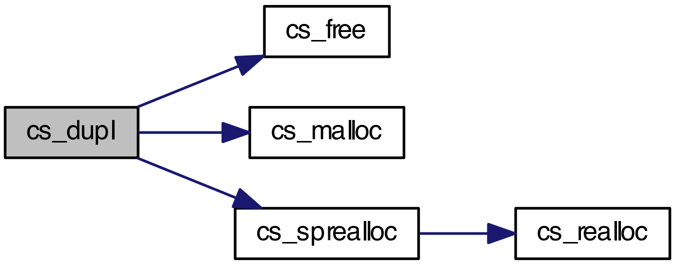


8.3.3.3 int cs_dupl (cs * *A*)

Definition at line 877 of file csparse.c.

References cs_free(), cs_malloc(), cs_sprealloc(), cs_sparse::i, cs_sparse::m, cs_sparse::n, cs_sparse::p, and cs-_sparse::x.

Here is the call graph for this function:



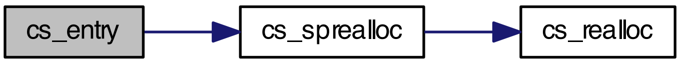
8.3.3.4 int cs_entry (cs ∗ *T*, int *i*, int *j*, double *x*)

Definition at line 926 of file csparse.c.

References CS_MAX, cs_sprealloc(), cs_sparse::i, cs_sparse::m, cs_sparse::n, cs_sparse::nz, cs_sparse::nzmax, cs_sparse::p, and cs_sparse::x.

Referenced by cs_load().

Here is the call graph for this function:

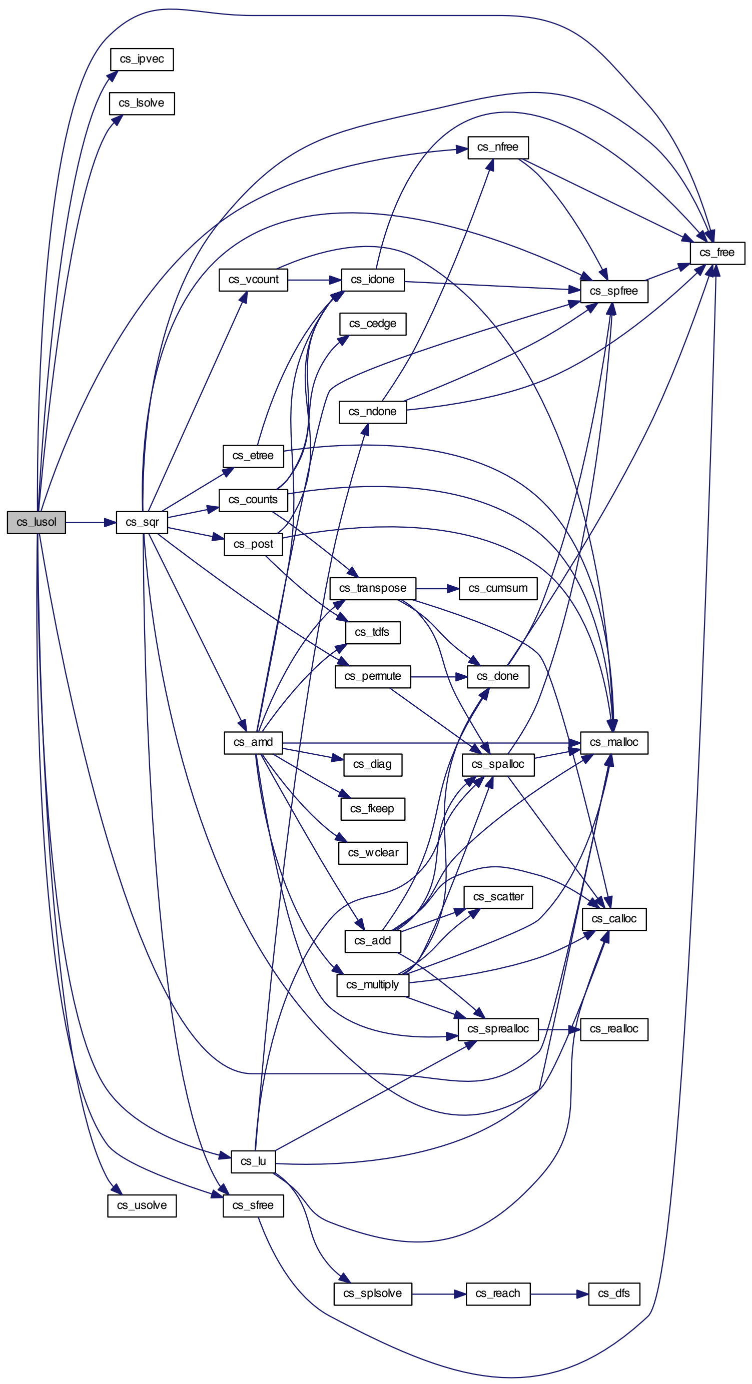


8.3.3.5 int cs_lusol (const cs ∗ *A*, double ∗ *b*, int *order*, double *tol*)

Definition at line 1255 of file csparse.c.

References cs_free(), cs_ipvec(), cs_lsolve(), cs_lu(), cs_malloc(), cs_nfree(), cs_sfree(), cs_sqr(), cs_usolve(), cs_numeric::L, cs_sparse::n, cs_numeric::Pinv, cs_symbolic::Q, and cs_numeric::U.

Here is the call graph for this function:



8.3.3.6 int cs_gaxpy (const cs ∗ *A*, const double ∗ *x*, double ∗ *y*)

Definition at line 998 of file csparse.c.

References cs_sparse::i, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by fclib_merit_local().

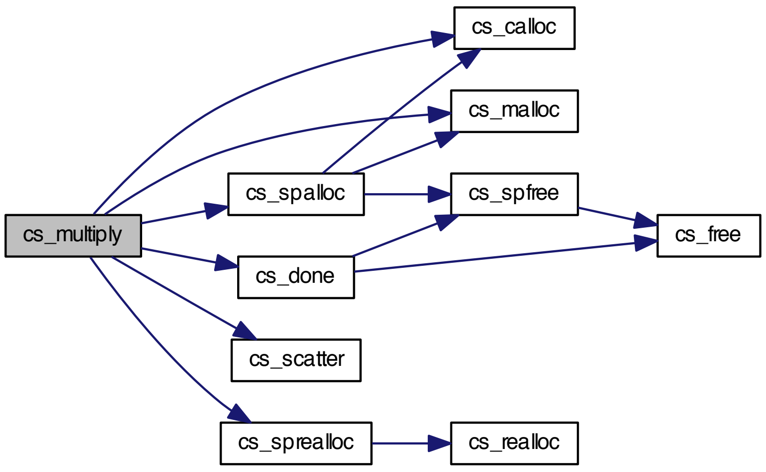
8.3.3.7 cs∗ cs_multiply (const cs ∗ *A*, const cs ∗ *B*)

Definition at line 1400 of file csparse.c.

References cs_calloc(), cs_done(), cs_malloc(), cs_scatter(), cs_spalloc(), cs_sprealloc(), cs_sparse::i, cs_sparse::m, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_amd().

Here is the call graph for this function:

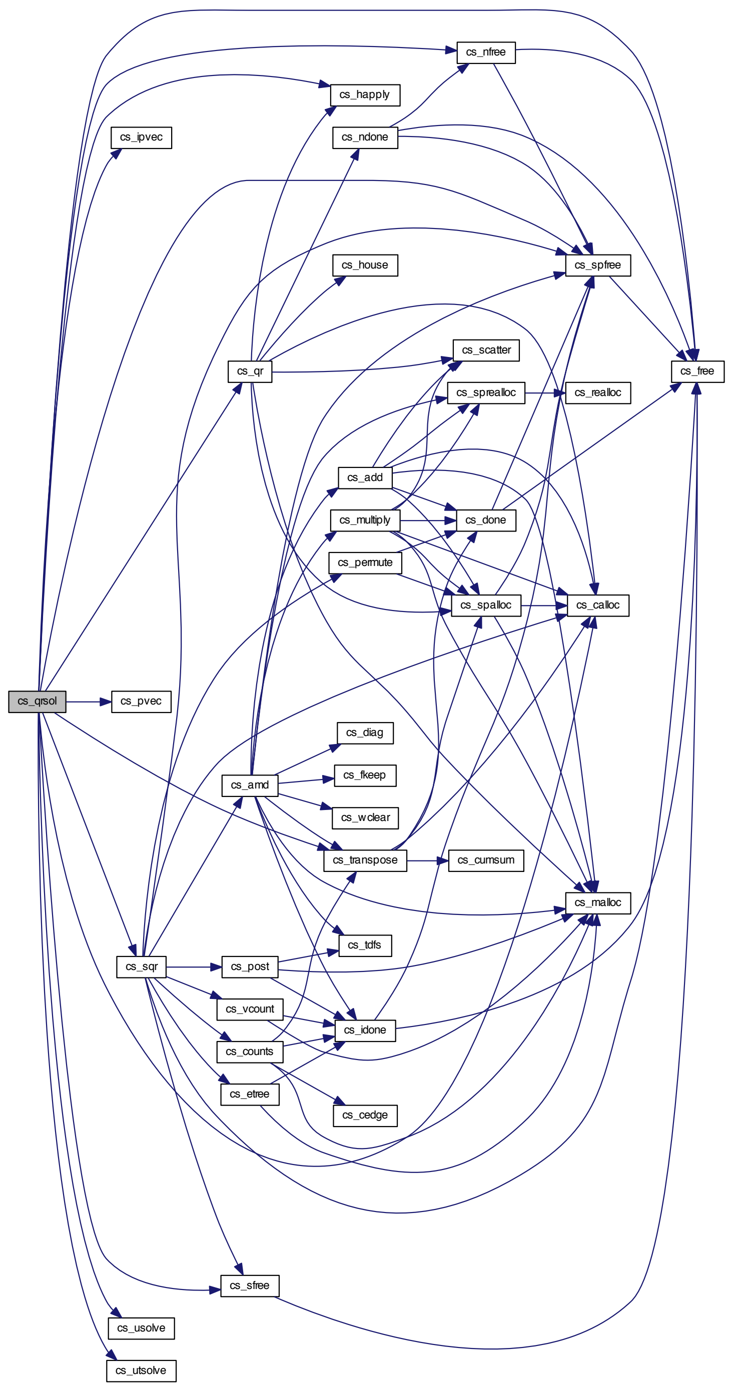


8.3.3.8 int cs_qrsol (const cs ∗ *A*, double ∗ *b*, int *order*)

Definition at line 1674 of file csparse.c.

References cs_numeric::B, cs_calloc(), cs_free(), cs_happly(), cs_ipvec(), cs_nfree(), cs_pvec(), cs_qr(), cs_sfree(), cs_spfree(), cs_sqr(), cs_transpose(), cs_usolve(), cs_utsolve(), cs_numeric::L, cs_sparse::m, cs_symbolic::m2, cs_sparse::n, cs_symbolic::Pinv, cs_symbolic::Q, and cs_numeric::U.

Here is the call graph for this function:



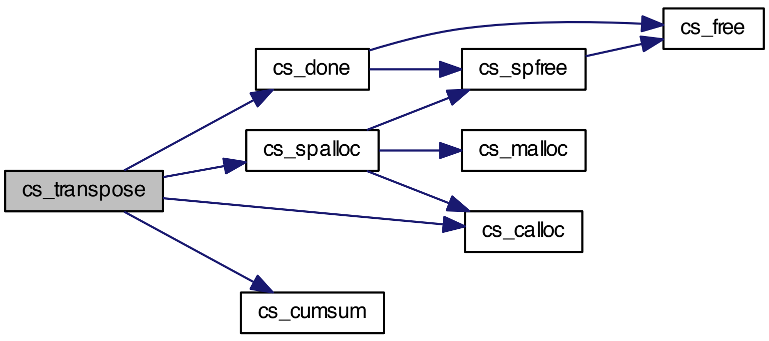
8.3.3.9 cs∗ cs_transpose (const cs ∗ *A*, int *values*)

Definition at line 2017 of file csparse.c.

References cs_calloc(), cs_cumsum(), cs_done(), cs_spalloc(), cs_sparse::i, cs_sparse::m, cs_sparse::n, cs_-sparse::p, and cs_sparse::x.

Referenced by cs_amd(), cs_bfs(), cs_counts(), cs_maxtrans(), cs_qrsol(), cs_scc(), and fclib_merit_local().

Here is the call graph for this function:

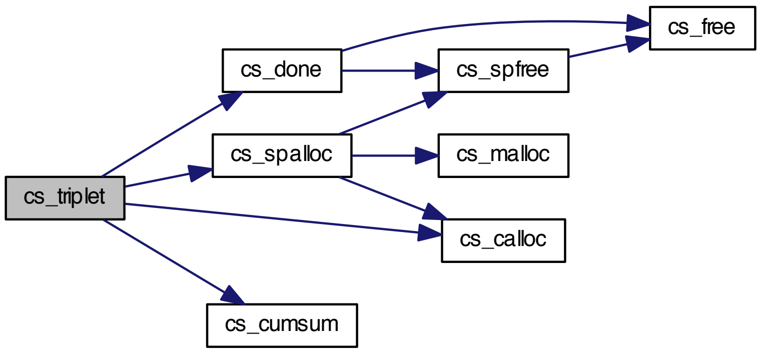


8.3.3.10 cs∗ cs_triplet (const cs ∗ *T*)

Definition at line 2048 of file csparse.c.

References cs_calloc(), cs_cumsum(), cs_done(), cs_spalloc(), cs_sparse::i, cs_sparse::m, cs_sparse::n, cs_-sparse::nz, cs_sparse::p, and cs_sparse::x.

Here is the call graph for this function:



8.3.3.11 double cs_norm (const cs ∗ *A*)

Definition at line 1440 of file csparse.c.

References CS_MAX, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_print().

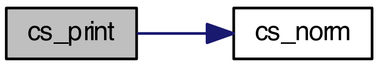
8.3.3.12 int cs_print (const cs ∗ *A*, int *brief*)

Definition at line 1525 of file csparse.c.

References CS_COPYRIGHT, CS_DATE, cs_norm(), CS_SUBSUB, CS_SUBVER, CS_VER, cs_sparse::i, cs_-

sparse::m, cs_sparse::n, cs_sparse::nz, cs_sparse::nzmax, cs_sparse::p, and cs_sparse::x.

Here is the call graph for this function:

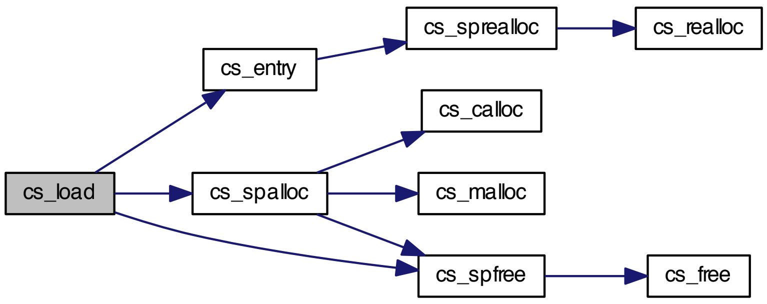


8.3.3.13 cs∗ cs_load (FILE ∗ *f*)

Definition at line 1069 of file csparse.c.

References cs_entry(), cs_spalloc(), and cs_spfree().

Here is the call graph for this function:



8.3.3.14 void∗ cs_calloc (int *n*, size_t *size*)

Definition at line 1294 of file csparse.c.

References CS_MAX, and CS_OVERFLOW.

Referenced by cs_add(), cs_chol(), cs_dalloc(), cs_lu(), cs_maxtrans(), cs_multiply(), cs_qr(), cs_qrsol(), cs_-schol(), cs_spalloc(), cs_sqr(), cs_symperm(), cs_transpose(), and cs_triplet().

8.3.3.15 void∗ cs_free (void ∗ *p*)

Definition at line 1300 of file csparse.c.

Referenced by cs_cholsol(), cs_ddone(), cs_dfree(), cs_dmperm(), cs_done(), cs_dupl(), cs_idone(), cs_lusol(), cs_ndone(), cs_nfree(), cs_qrsol(), cs_schol(), cs_sfree(), cs_spfree(), cs_sqr(), and cs_updown().

8.3.3.16 void∗ cs_realloc (void ∗ *p*, int *n*, size_t *size*, int ∗ *ok*)

Definition at line 1307 of file csparse.c.

References CS_MAX, and CS_OVERFLOW.

Referenced by cs_sprealloc().

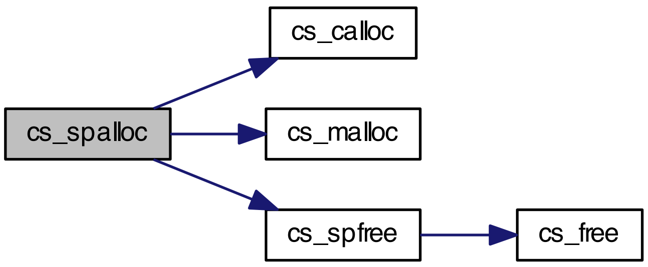
8.3.3.17 cs∗ cs_spalloc (int *m*, int *n*, int *nzmax*, int *values*, int *triplet*)

Definition at line 2140 of file csparse.c.

References cs_calloc(), cs_malloc(), CS_MAX, cs_spfree(), cs_sparse::i, cs_sparse::m, cs_sparse::n, cs_sparse::nz, cs_sparse::nzmax, cs_sparse::p, and cs_sparse::x.

Referenced by cs_add(), cs_chol(), cs_load(), cs_lu(), cs_multiply(), cs_permute(), cs_qr(), cs_symperm(), cs_transpose(), and cs_triplet().

Here is the call graph for this function:



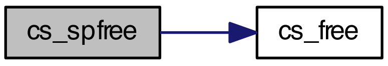
8.3.3.18 cs∗ cs_spfree (cs ∗ *A*)

Definition at line 2169 of file csparse.c.

References cs_free(), cs_sparse::i, cs_sparse::p, and cs_sparse::x.

Referenced by cs_amd(), cs_bfs(), cs_ddone(), cs_done(), cs_idone(), cs_load(), cs_ndone(), cs_nfree(), cs_qrsol(), cs_schol(), cs_spalloc(), and cs_sqr().

Here is the call graph for this function:



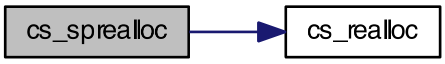
8.3.3.19 int cs_sprealloc (cs ∗ *A*, int *nzmax*)

Definition at line 2155 of file csparse.c.

References cs_realloc(), cs_sparse::i, cs_sparse::n, cs_sparse::nz, cs_sparse::nzmax, cs_sparse::p, and cs_sparse::x.

Referenced by cs_add(), cs_amd(), cs_dupl(), cs_entry(), cs_lu(), and cs_multiply().

Here is the call graph for this function:



8.3.3.20 void∗ cs_malloc (int *n,* size_t *size*)

Definition at line 1288 of file csparse.c.

References CS_MAX, and CS_OVERFLOW.

Referenced by cs_add(), cs_amd(), cs_chol(), cs_cholsol(), cs_counts(), cs_dalloc(), cs_dupl(), cs_etree(), cs_-lu(), cs_lusol(), cs_maxtrans(), cs_multiply(), cs_pinv(), cs_post(), cs_qr(), cs_scc(), cs_schol(), cs_spalloc(), cs_-updown(), and cs_vcount().

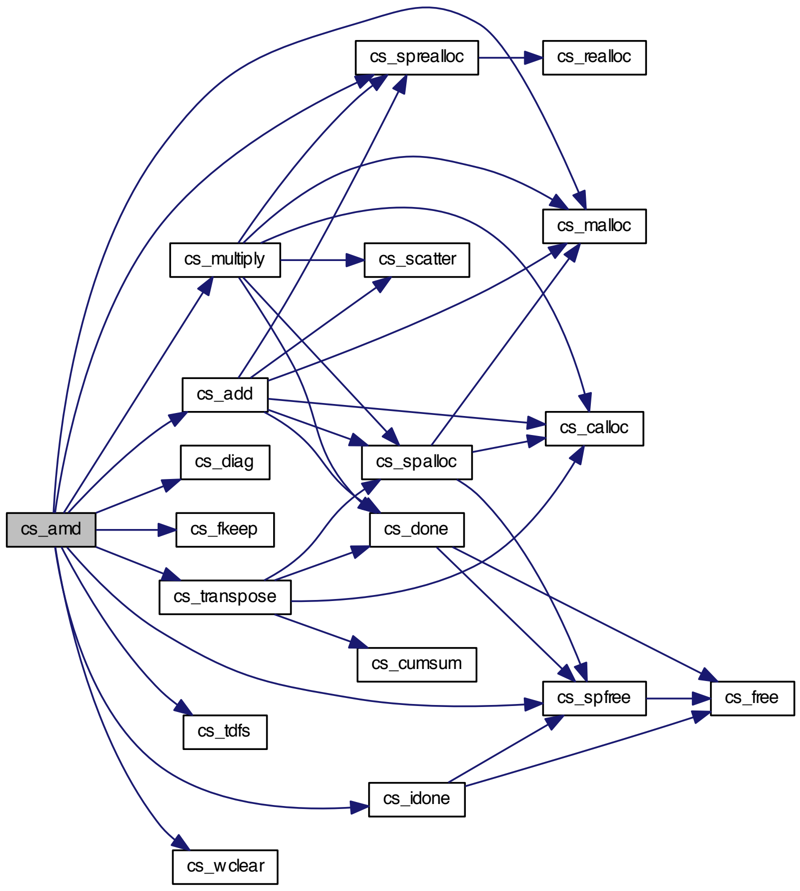
8.3.3.21 int∗ cs_amd (const cs ∗ *A,* int *order*)

Definition at line 79 of file csparse.c.

References cs_add(), cs_diag(), cs_fkeep(), CS_FLIP, cs_idone(), cs_malloc(), CS_MAX, CS_MIN, cs_multiply(), cs_spfree(), cs_sprealloc(), cs_tdfs(), cs_transpose(), cs_wclear(), cs_sparse::i, cs_sparse::m, cs_sparse::n, cs_-sparse::nzmax, and cs_sparse::p.

Referenced by cs_schol(), and cs_sqr().

Here is the call graph for this function:



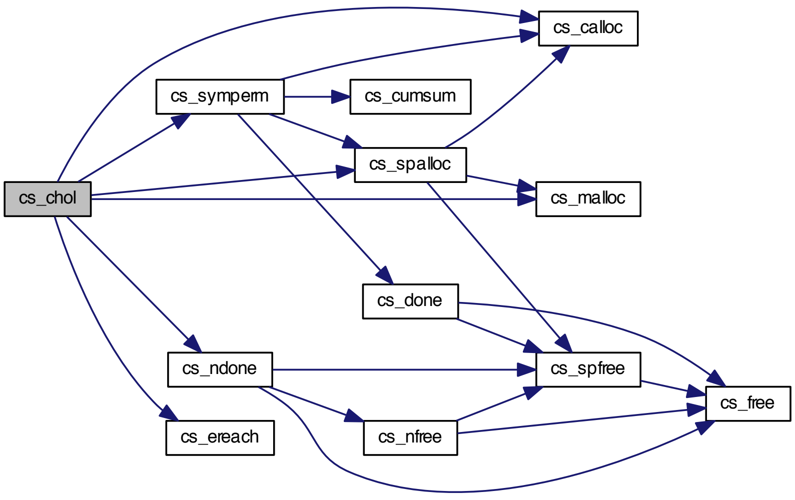
8.3.3.22 csn∗ cs_chol (const cs ∗ *A*, const css ∗ *S*)

Definition at line 474 of file csparse.c.

References cs_symbolic::cp, cs_calloc(), cs_ereach(), cs_malloc(), cs_ndone(), cs_spalloc(), cs_symperm(), cs_-sparse::i, cs_numeric::L, cs_sparse::n, cs_sparse::p, cs_symbolic::parent, cs_symbolic::Pinv, and cs_sparse::x.

Referenced by cs_cholsol().

Here is the call graph for this function:

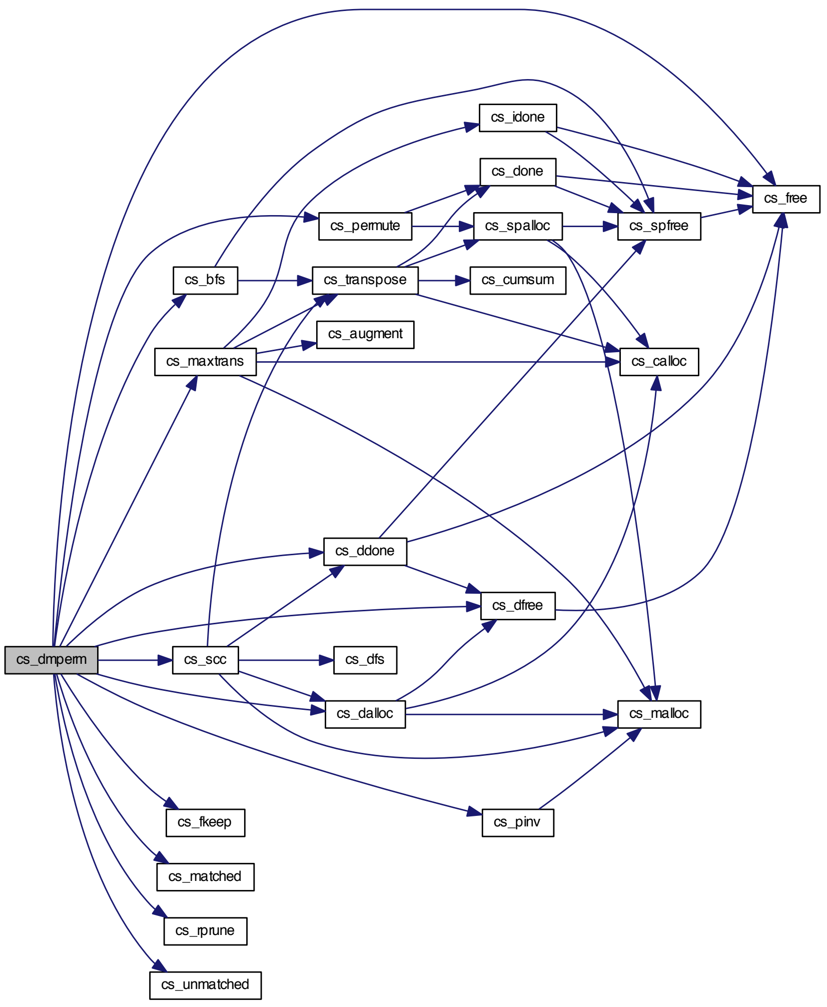


8.3.3.23 csd∗ cs_dmperm (const cs ∗ *A*)

Definition at line 774 of file csparse.c.

References cs_dmperm_results::cc, cs_bfs(), cs_dalloc(), cs_ddone(), cs_dfree(), cs_fkeep(), cs_free(), cs_-matched(), cs_maxtrans(), cs_permute(), cs_pinv(), cs_rprune(), cs_scc(), cs_unmatched(), cs_sparse::i, cs_-sparse::m, cs_sparse::n, cs_dmperm_results::nb, cs_sparse::p, cs_dmperm_results::P, cs_dmperm_results::Q, cs-_dmperm_results::R, cs_dmperm_results::rr, and cs_dmperm_results::S.

Here is the call graph for this function:

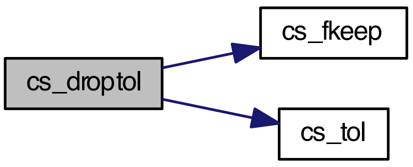


8.3.3.24 int cs_droptol (cs * *A*, double *tol*)

Definition at line 864 of file csparse.c.

References cs_fkeep(), and cs_tol().

Here is the call graph for this function:

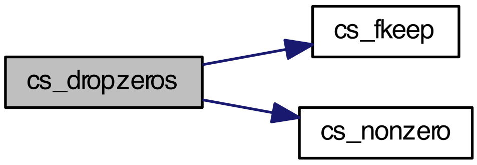


8.3.3.25 int cs_dropzeros (cs * *A*)

Definition at line 873 of file csparse.c.

References cs_fkeep(), and cs_nonzero().

Here is the call graph for this function:



8.3.3.26 int cs_happly (const cs * *V*, int *i*, double *beta*, double * *x*)

Definition at line 1018 of file csparse.c.

References cs_sparse::i, cs_sparse::p, and cs_sparse::x.

Referenced by cs_qr(), and cs_qrsol().

8.3.3.27 int cs_ipvec (int *n*, const int * *P*, const double * *b*, double * *x*)

Definition at line 1062 of file csparse.c.

Referenced by cs_cholsol(), cs_lusol(), and cs_qrsol().

8.3.3.28 int cs_lsolve (const cs * *L*, double * *x*)

Definition at line 1093 of file csparse.c.

References cs_sparse::i, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_cholsol(), and cs_lusol().

8.3.3.29 int cs_ltsolve (const cs * *L*, double * *x*)

Definition at line 1127 of file csparse.c.

References cs_sparse::i, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_cholsol().

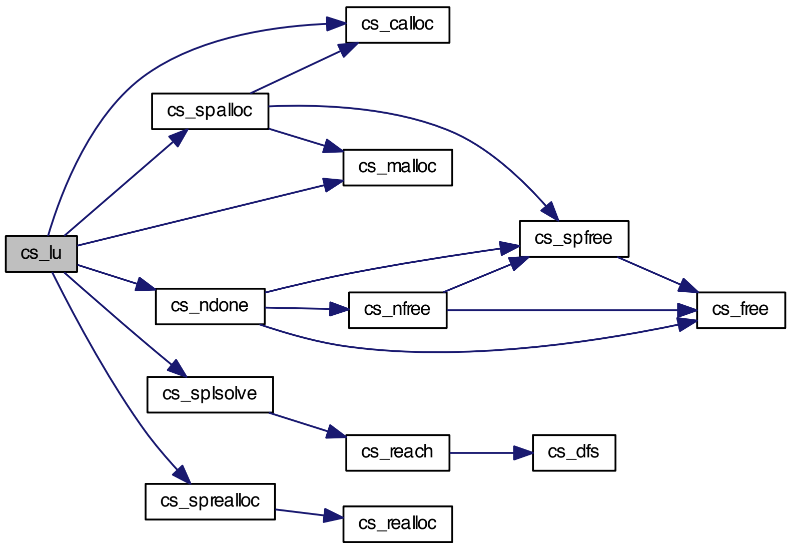
8.3.3.30 csn* cs_lu (const cs * *A*, const css * *S*, double *tol*)

Definition at line 1163 of file csparse.c.

References cs_calloc(), cs_malloc(), cs_ndone(), cs_spalloc(), cs_splsolve(), cs_sprealloc(), cs_sparse::i, cs_-numeric::L, cs_symbolic::lnz, cs_sparse::n, cs_sparse::nzmax, cs_sparse::p, cs_numeric::Pinv, cs_symbolic::Q, cs-_numeric::U, cs_symbolic::unz, and cs_sparse::x.

Referenced by cs_lusol().

Here is the call graph for this function:



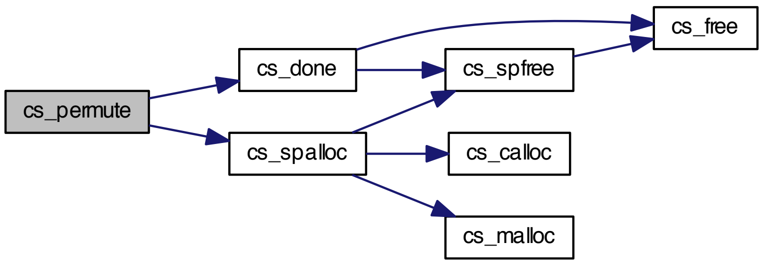
8.3.3.31 cs∗ cs_permute (const cs ∗ *A*, const int ∗ *P*, const int ∗ *Q*, int *values*)

Definition at line 1457 of file csparse.c.

References cs_done(), cs_spalloc(), cs_sparse::i, cs_sparse::m, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_dmperm(), and cs_sqr().

Here is the call graph for this function:



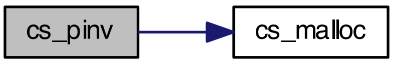
8.3.3.32 int∗ cs_pinv (const int ∗ *P*, int *n*)

Definition at line 1488 of file csparse.c.

References cs_malloc().

Referenced by cs_dmperm(), and cs_schol().

Here is the call graph for this function:



8.3.3.33 int cs_pvec (int *n*, const int ∗ *P*, const double ∗ *b*, double ∗ *x*)

Definition at line 1578 of file csparse.c.

Referenced by cs_cholsol(), and cs_qrsol().

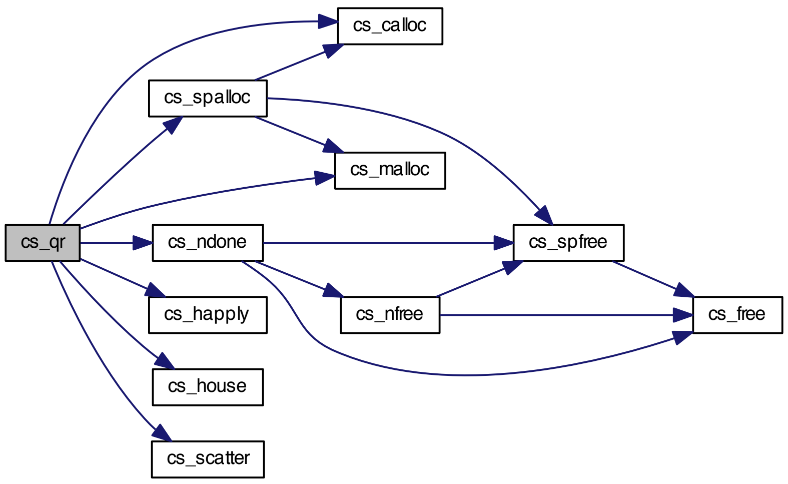
8.3.3.34 csn∗ cs_qr (const cs ∗ *A*, const css ∗ *S*)

Definition at line 1587 of file csparse.c.

References cs_numeric::B, cs_calloc(), cs_happly(), cs_house(), cs_malloc(), cs_ndone(), cs_scatter(), cs_-spalloc(), cs_sparse::i, cs_numeric::L, cs_symbolic::lnz, cs_sparse::m, cs_symbolic::m2, cs_sparse::n, cs_sparse-::p, cs_symbolic::parent, cs_symbolic::Pinv, cs_symbolic::Q, cs_numeric::U, cs_symbolic::unz, and cs_sparse::x.

Referenced by cs_qrsol().

Here is the call graph for this function:



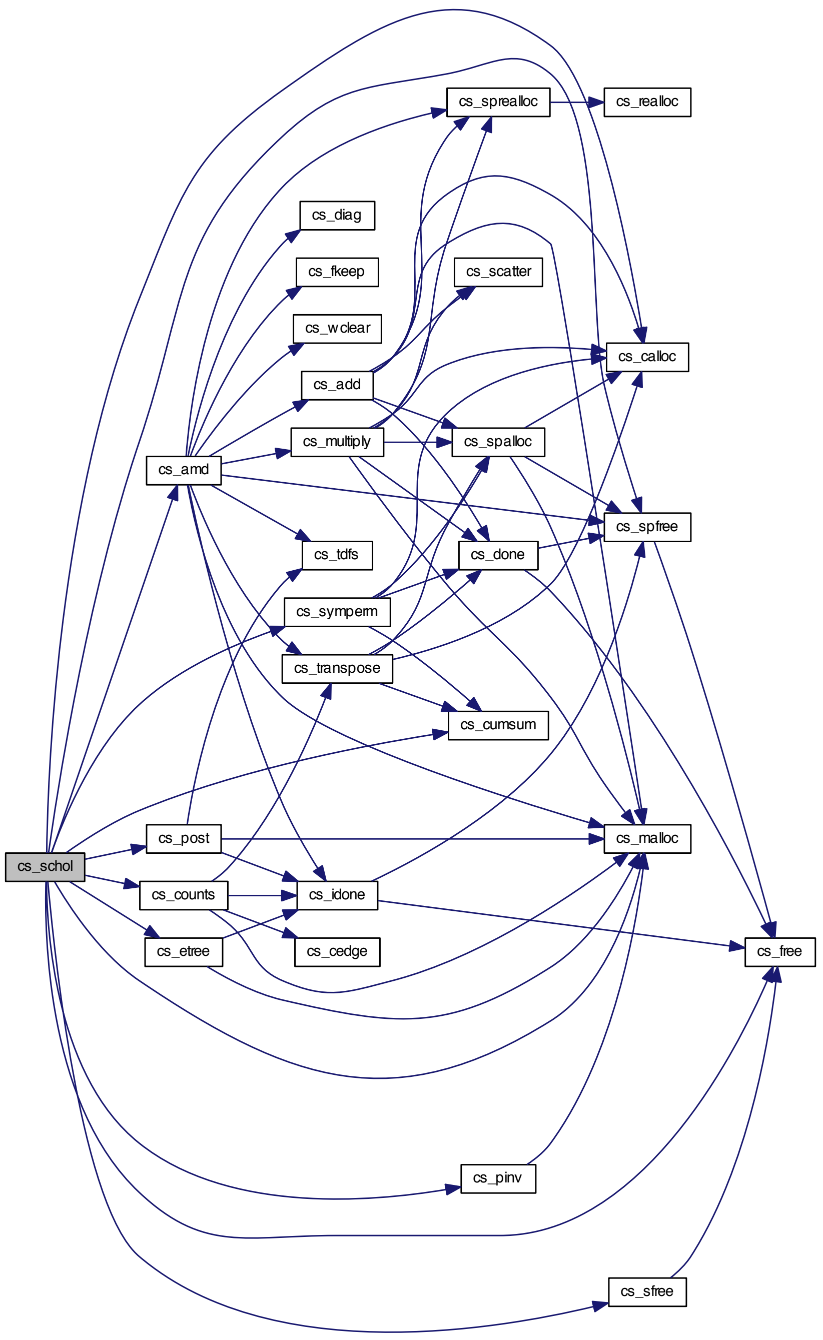
8.3.3.35 css∗ cs_schol (const cs ∗ *A*, int *order*)

Definition at line 1812 of file csparse.c.

References cs_symbolic::cp, cs_amd(), cs_calloc(), cs_counts(), cs_cumsum(), cs_etree(), cs_free(), cs_malloc(), cs_pinv(), cs_post(), cs_sfree(), cs_spfree(), cs_symperm(), cs_symbolic::lnz, cs_sparse::n, cs_symbolic::parent, cs_symbolic::Pinv, and cs_symbolic::unz.

Referenced by cs_cholsol().

Here is the call graph for this function:



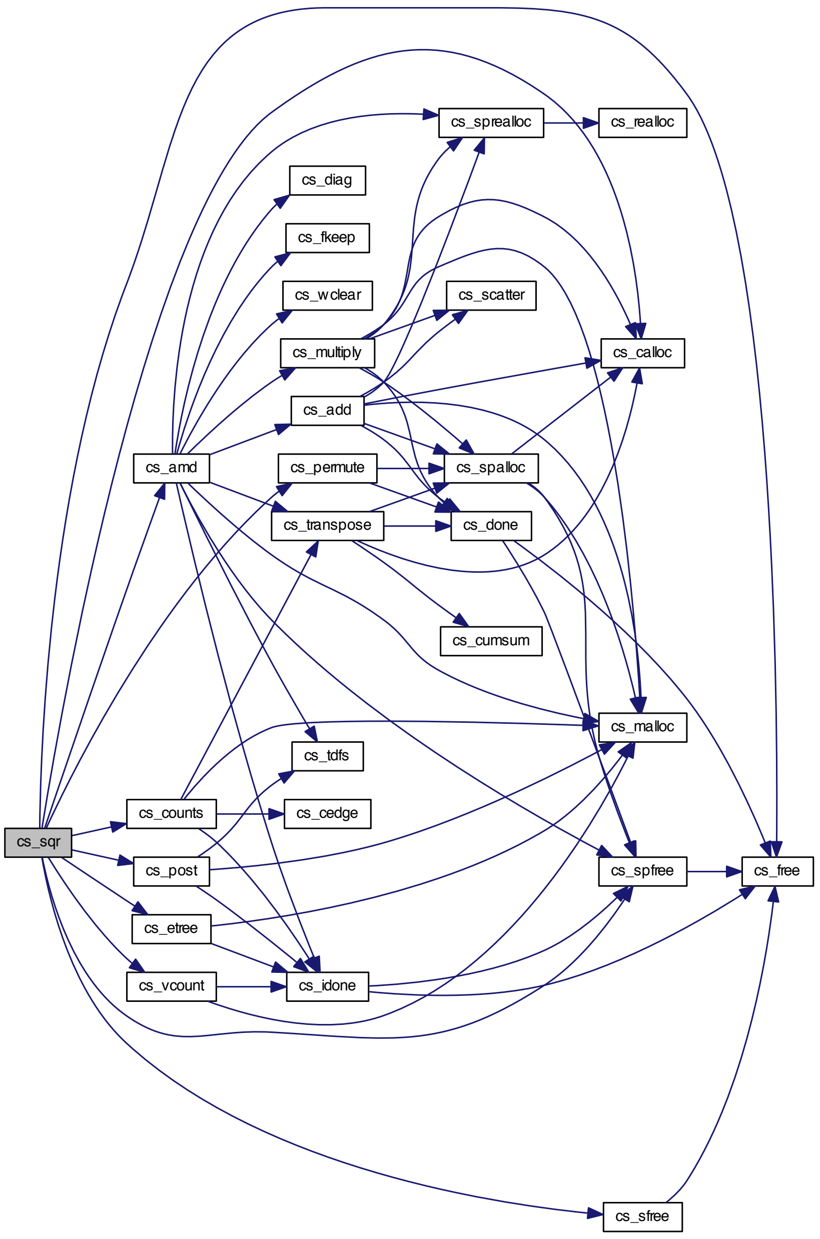
8.3.3.36 css∗ cs_sqr (const cs ∗ *A*, int *order*, int *qr*)

Definition at line 1917 of file csparse.c.

References cs_symbolic::cp, cs_amd(), cs_calloc(), cs_counts(), cs_etree(), cs_free(), cs_permute(), cs_post(), cs_sfree(), cs_spfree(), cs_vcount(), cs_symbolic::lnz, cs_symbolic::m2, cs_sparse::n, cs_sparse::p, cs_symbolic::parent, cs_symbolic::Pinv, cs_symbolic::Q, and cs_symbolic::unz.

Referenced by cs_lusol(), and cs_qrsol().

Here is the call graph for this function:



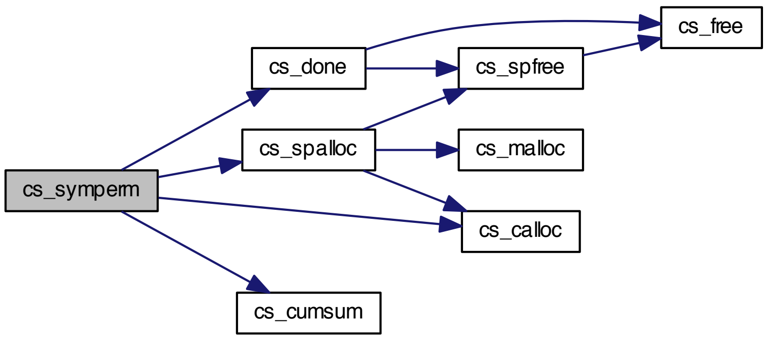
8.3.3.37 cs∗ cs_symperm (const cs ∗ *A,* const int ∗ *Pinv,* int *values*)

Definition at line 1949 of file csparse.c.

References cs_calloc(), cs_cumsum(), cs_done(), CS_MAX, CS_MIN, cs_spalloc(), cs_sparse::i, cs_sparse::n, cs-_sparse::p, and cs_sparse::x.

Referenced by cs_chol(), and cs_schol().

Here is the call graph for this function:



8.3.3.38 int cs_usolve (const cs ∗ *U,* double ∗ *x*)

Definition at line 2119 of file csparse.c.

References cs_sparse::i, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_lusol(), and cs_qrsol().

8.3.3.39 int cs_utsolve (const cs ∗ *U,* double ∗ *x*)

Definition at line 2258 of file csparse.c.

References cs_sparse::i, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

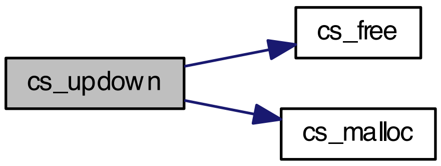
Referenced by cs_qrsol().

8.3.3.40 int cs_updown (cs ∗ *L,* int *sigma,* const cs ∗ *C,* const int ∗ *parent*)

Definition at line 2077 of file csparse.c.

References cs_free(), cs_malloc(), CS_MIN, cs_sparse::i, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Here is the call graph for this function:



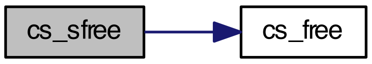
8.3.3.41 css∗ cs_sfree (css ∗ *S*)

Definition at line 2190 of file csparse.c.

References cs_symbolic::cp, cs_free(), cs_symbolic::parent, cs_symbolic::Pinv, and cs_symbolic::Q.

Referenced by cs_cholsol(), cs_lusol(), cs_qrsol(), cs_schol(), and cs_sqr().

Here is the call graph for this function:



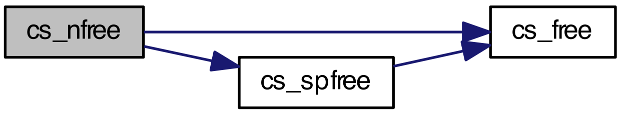
8.3.3.42 csn∗ cs_nfree (csn ∗ *N*)

Definition at line 2179 of file csparse.c.

References cs_numeric::B, cs_free(), cs_spfree(), cs_numeric::L, cs_numeric::Pinv, and cs_numeric::U.

Referenced by cs_cholsol(), cs_lusol(), cs_ndone(), and cs_qrsol().

Here is the call graph for this function:



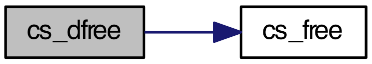
8.3.3.43 csd∗ cs_dfree (csd ∗ *D*)

Definition at line 2214 of file csparse.c.

References cs_free(), cs_dmperm_results::P, cs_dmperm_results::Q, cs_dmperm_results::R, and cs_dmperm_-results::S.

Referenced by cs_dalloc(), cs_ddone(), and cs_dmperm().

Here is the call graph for this function:



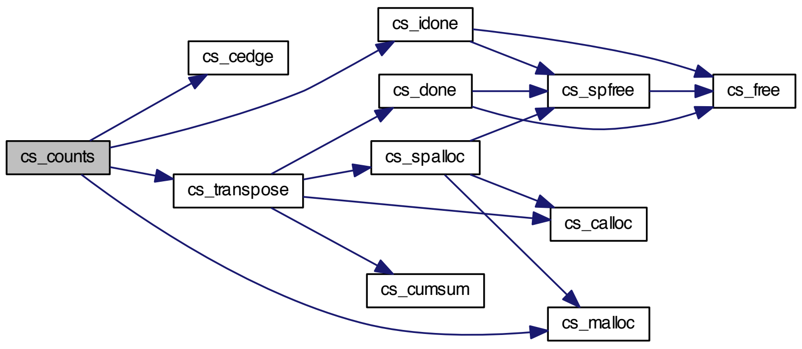
8.3.3.44 int∗ cs_counts (const cs ∗ *A*, const int ∗ *parent*, const int ∗ *post*, int *ata*)

Definition at line 582 of file csparse.c.

References cs_cedge(), cs_idone(), cs_malloc(), CS_MIN, cs_transpose(), cs_sparse::i, cs_sparse::m, cs_sparse::n, and cs_sparse::p.

Referenced by cs_schol(), and cs_sqr().

Here is the call graph for this function:



8.3.3.45 int cs_cumsum (int ∗ *p*, int ∗ *c*, int *n*)

Definition at line 651 of file csparse.c.

Referenced by cs_schol(), cs_symperm(), cs_transpose(), and cs_triplet().

8.3.3.46 int cs_dfs (int *j*, cs ∗ *L*, int *top*, int ∗ *xi*, int ∗ *pstack*, const int ∗ *Pinv*)

Definition at line 666 of file csparse.c.

References CS_MARK, CS_MARKED, CS_UNFLIP, cs_sparse::i, and cs_sparse::p.

Referenced by cs_reach(), and cs_scc().

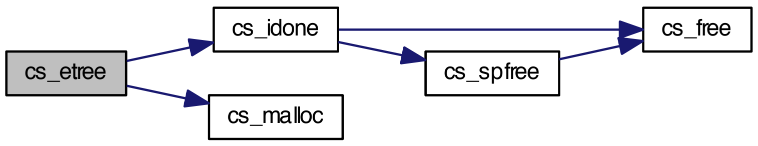
8.3.3.47 int∗ cs_etree (const cs ∗ *A*, int *ata*)

Definition at line 938 of file csparse.c.

References cs_idone(), cs_malloc(), cs_sparse::i, cs_sparse::m, cs_sparse::n, and cs_sparse::p.

Referenced by cs_schol(), and cs_sqr().

Here is the call graph for this function:



8.3.3.48 int cs_fkeep (cs * *A,* int(*)(int, int, double, void *) *fkeep,* void * *other*)

Definition at line 972 of file csparse.c.

References cs_sparse::i, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_amd(), cs_dmperm(), cs_droptol(), and cs_dropzeros().

8.3.3.49 double cs_house (double * *x,* double * *beta,* int *n*)

Definition at line 1040 of file csparse.c.

Referenced by cs_qr().

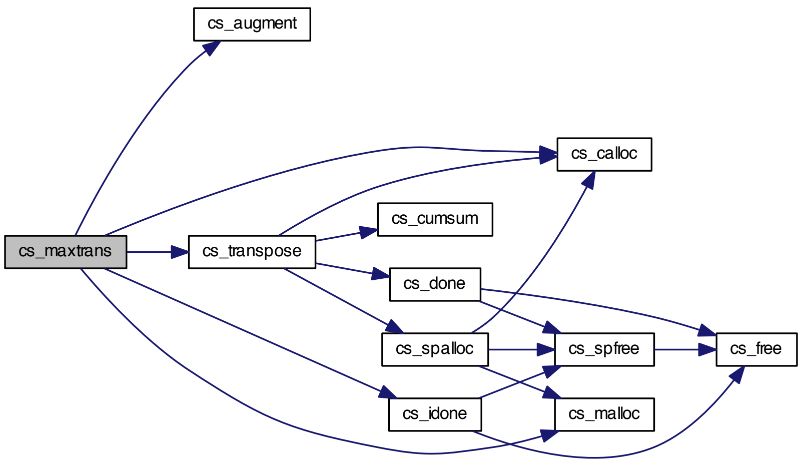
8.3.3.50 int* cs_maxtrans (const cs * *A*)

Definition at line 1359 of file csparse.c.

References cs_augment(), cs_calloc(), cs_idone(), cs_malloc(), cs_transpose(), cs_sparse::i, cs_sparse::m, cs_-sparse::n, and cs_sparse::p.

Referenced by cs_dmperm().

Here is the call graph for this function:



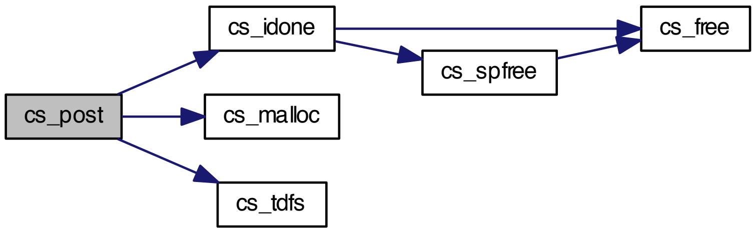
8.3.3.51 int* cs_post (int *n,* const int * *parent*)

Definition at line 1499 of file csparse.c.

References cs_idone(), cs_malloc(), and cs_tdfs().

Referenced by cs_schol(), and cs_sqr().

Here is the call graph for this function:



8.3.3.52 int cs_reach (cs ∗ *L,* const cs ∗ *B,* int *k,* int ∗ *xi,* const int ∗ *Pinv*)

Definition at line 1728 of file csparse.c.

References cs_dfs(), CS_MARK, CS_MARKED, cs_sparse::i, cs_sparse::n, and cs_sparse::p.

Referenced by cs_splsolve().

Here is the call graph for this function:

cs_reach
cs_dfs

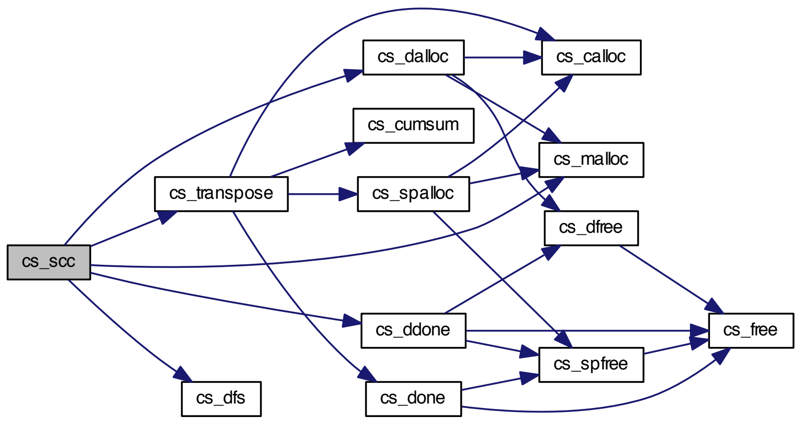
8.3.3.53 csd∗ cs_scc (cs ∗ *A*)

Definition at line 1774 of file csparse.c.

References cs_dalloc(), cs_ddone(), cs_dfs(), cs_malloc(), CS_MARK, CS_MARKED, cs_transpose(), cs_sparse::n, cs_dmperm_results::nb, cs_sparse::p, cs_dmperm_results::P, and cs_dmperm_results::R.

Referenced by cs_dmperm().

Here is the call graph for this function:



8.3.3.54 int cs_scatter (const cs ∗ *A,* int *j,* double *beta,* int ∗ *w,* double ∗ *x,* int *mark,* cs ∗ *C,* int *nz*)

Definition at line 1749 of file csparse.c.

References cs_sparse::i, cs_sparse::p, and cs_sparse::x.

Referenced by cs_add(), cs_multiply(), and cs_qr().

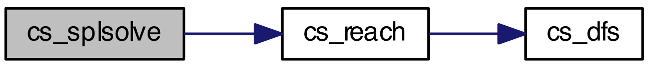
8.3.3.55 int cs_splsolve (cs ∗ *L,* const cs ∗ *B,* int *k,* int ∗ *xi,* double ∗ *x,* const int ∗ *Pinv*)

Definition at line 1838 of file csparse.c.

References cs_reach(), cs_sparse::i, cs_sparse::n, cs_sparse::p, and cs_sparse::x.

Referenced by cs_lu().

Here is the call graph for this function:



8.3.3.56 int cs_tdfs (int *j,* int *k,* int ∗ *head,* const int ∗ *next,* int ∗ *post,* int ∗ *stack*)

Definition at line 1993 of file csparse.c.

Referenced by cs_amd(), and cs_post().

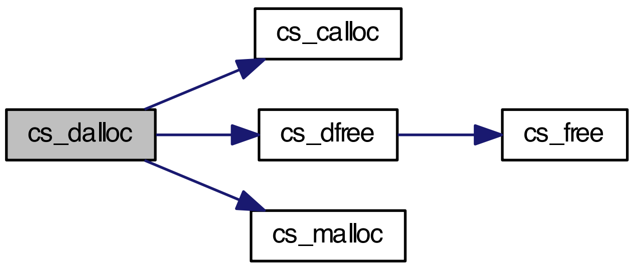
8.3.3.57 csd∗ cs_dalloc (int *m,* int *n*)

Definition at line 2201 of file csparse.c.

References cs_calloc(), cs_dfree(), cs_malloc(), cs_dmperm_results::P, cs_dmperm_results::Q, cs_dmperm_-results::R, and cs_dmperm_results::S.

Referenced by cs_dmperm(), and cs_scc().

Here is the call graph for this function:



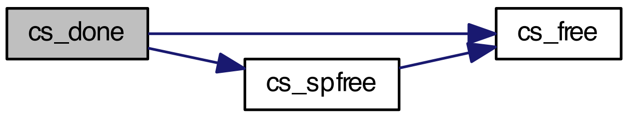
8.3.3.58 cs∗ cs_done (cs ∗ *C*, void ∗ *w*, void ∗ *x*, int *ok*)

Definition at line 2225 of file csparse.c.

References cs_free(), and cs_spfree().

Referenced by cs_add(), cs_multiply(), cs_permute(), cs_symperm(), cs_transpose(), and cs_triplet().

Here is the call graph for this function:



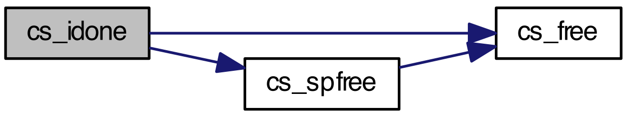
8.3.3.59 int∗ cs_idone (int ∗ *p*, cs ∗ *C*, void ∗ *w*, int *ok*)

Definition at line 2233 of file csparse.c.

References cs_free(), and cs_spfree().

Referenced by cs_amd(), cs_counts(), cs_etree(), cs_maxtrans(), cs_post(), and cs_vcount().

Here is the call graph for this function:



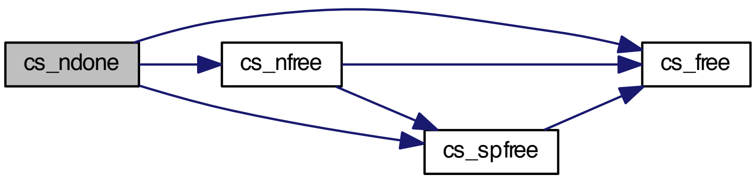
8.3.3.60 csn∗ cs_ndone (csn ∗ *N*, cs ∗ *C*, void ∗ *w*, void ∗ *x*, int *ok*)

Definition at line 2241 of file csparse.c.

References cs_free(), cs_nfree(), and cs_spfree().

Referenced by cs_chol(), cs_lu(), and cs_qr().

Here is the call graph for this function:



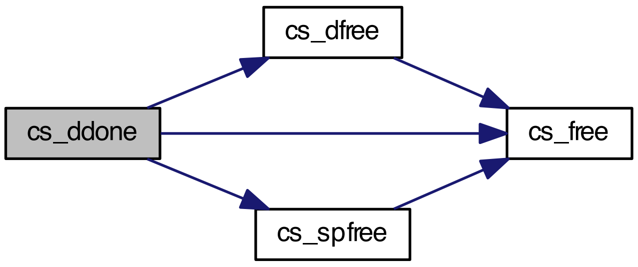
8.3.3.61 csd∗ cs_ddone (csd ∗ *D*, cs ∗ *C*, void ∗ *w*, int *ok*)

Definition at line 2250 of file csparse.c.

References cs_dfree(), cs_free(), and cs_spfree().

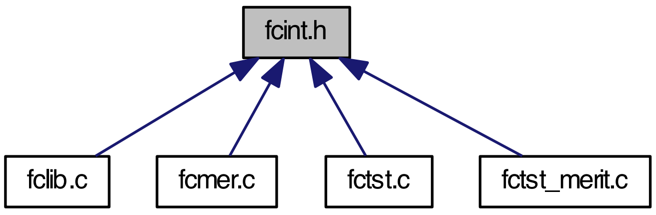
Referenced by cs_dmperm(), and cs_scc().

Here is the call graph for this function:



8.4 fcint.h File Reference

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



Macros

- #define ASSERT(Test,...)
- #define IO(Call) ASSERT ((Call) >= 0, "ERROR: HDF5 call failed")
- #define MM(Call) ASSERT ((Call), "ERROR: out of memory")

8.4.1 Macro Definition Documentation

8.4.1.1 #define ASSERT(*Test,* *...*)

Value:

```
do {\
  if (! (Test)) { fprintf (stderr, "%s: %d => ", __FILE__, __LINE__);\
    fprintf (stderr, __VA_ARGS__);\
    fprintf (stderr, "\n"); exit (1); } } while (0)
```

Definition at line 11 of file fcint.h.

Referenced by fclib_write_global(), fclib_write_local(), main(), read_global_vectors(), read_local_vectors(), read_-matrix(), read_solution(), write_global_vectors(), write_local_vectors(), write_matrix(), and write_solution().

8.4.1.2 #define IO(*Call*) ASSERT ((Call) >= 0, "ERROR: HDF5 call failed")

Definition at line 17 of file fcint.h.

Referenced by fclib_read_global(), fclib_read_guesses(), fclib_read_local(), fclib_read_solution(), fclib_write-_global(), fclib_write_guesses(), fclib_write_local(), fclib_write_solution(), read_global_vectors(), read_local_-vectors(), read_matrix(), read_nvnunrnl(), read_problem_info(), read_solution(), write_global_vectors(), write_local-_vectors(), write_matrix(), write_problem_info(), and write_solution().

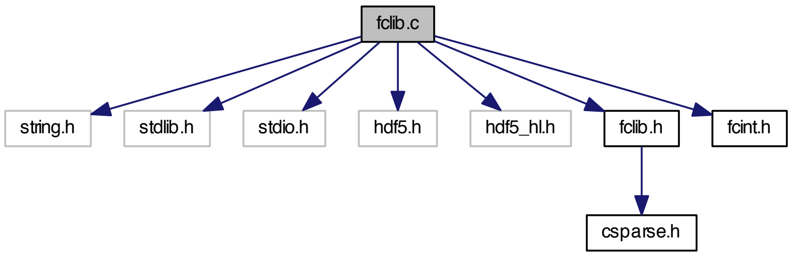
8.4.1.3 #define MM(*Call*) ASSERT ((Call), "ERROR: out of memory")

Definition at line 18 of file fcint.h.

Referenced by fclib_read_global(), fclib_read_guesses(), fclib_read_local(), fclib_read_solution(), matrix_info(), problem_info(), random_global_problem(), random_global_solutions(), random_local_problem(), random_local_-solutions(), random_matrix(), random_vector(), read_global_vectors(), read_local_vectors(), read_matrix(), read-_problem_info(), and read_solution().

8.5 fclib.c File Reference

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <hdf5.h>
#include <hdf5_hl.h>
#include "fclib.h"
#include "fcint.h"
```
Include dependency graph for fclib.c:



Macros

- #define H5Gcreate_vers 2
- #define H5Gopen_vers 2

Functions

- static hid_t H5Gmake (hid_t loc_id, const char ∗name)

  make grourp
- static void write_matrix (hid_t id, struct fclib_matrix ∗mat)

  write matrix
- struct fclib_matrix ∗ read_matrix (hid_t id)

  read matrix

- static void write_global_vectors (hid_t id, struct fclib_global ∗problem)

  write global vectors
- static void read_global_vectors (hid_t id, struct fclib_global ∗problem)

  read global vectors
- static void write_local_vectors (hid_t id, struct fclib_local ∗problem)

  write local vectors
- static void read_local_vectors (hid_t id, struct fclib_local ∗problem)

  read local vectors
- static void write_problem_info (hid_t id, struct fclib_info ∗info)

  write problem info
- static struct fclib_info ∗ read_problem_info (hid_t id)

  read problem info
- static void write_solution (hid_t id, struct fclib_solution ∗solution, hsize_t nv, hsize_t nr, hsize_t nl)

  write solution
- static void read_solution (hid_t id, hsize_t nv, hsize_t nr, hsize_t nl, struct fclib_solution ∗solution)

  read solution
- static int read_nvnunrnl (hid_t file_id, int ∗nv, int ∗nr, int ∗nl)

  read solution sizes
- static void delete_matrix_info (struct fclib_matrix_info ∗info)

  delete matrix info
- static void delete_matrix (struct fclib_matrix ∗mat)

  delete matrix
- static void delete_info (struct fclib_info ∗info)

  delete problem info
- int fclib_write_global (struct fclib_global ∗problem, const char ∗path)

  write global problem; return 1 on success, 0 on failure
- int fclib_write_local (struct fclib_local ∗problem, const char ∗path)

  write local problem; return 1 on success, 0 on failure
- int fclib_write_solution (struct fclib_solution ∗solution, const char ∗path)

  write solution; return 1 on success, 0 on failure
- int fclib_write_guesses (int number_of_guesses, struct fclib_solution ∗guesses, const char ∗path)

  write initial guesses; return 1 on success, 0 on failure
- struct fclib_global ∗ fclib_read_global (const char ∗path)

  read global problem; return problem on success; NULL on failure
- struct fclib_local ∗ fclib_read_local (const char ∗path)

  read local problem; return problem on success; NULL on failure
- struct fclib_solution ∗ fclib_read_solution (const char ∗path)

  read solution; return solution on success; NULL on failure
- struct fclib_solution ∗ fclib_read_guesses (const char ∗path, int ∗number_of_guesses)

  read initial guesses; return vector of guesses on success; NULL on failure; output numebr of guesses in the variable pointed by 'number_of_guesses'
- void fclib_delete_global (struct fclib_global ∗problem)

  delete global problem
- void fclib_delete_local (struct fclib_local ∗problem)

  delete local problem
- void fclib_delete_solutions (struct fclib_solution ∗data, int count)

  delete solutions or guesses

8.5.1 Detailed Description

frictional contact library input/output

Definition in file fclib.c.

8.5.2 Macro Definition Documentation

8.5.2.1 #define H5Gcreate_vers 2

Definition at line 24 of file fclib.c.

8.5.2.2 #define H5Gopen_vers 2

Definition at line 25 of file fclib.c.

8.5.3 Function Documentation

8.5.3.1 static hid_t H5Gmake (hid_t *loc_id,* const char ∗ *name*) [static]

make grourp

Definition at line 37 of file fclib.c.

Referenced by fclib_write_global(), fclib_write_guesses(), fclib_write_local(), and fclib_write_solution().

8.5.3.2 static void write_matrix (hid_t *id,* struct fclib_matrix ∗ *mat*) [static]

write matrix

Definition at line 51 of file fclib.c.

References ASSERT, fclib_matrix_info::comment, fclib_matrix_info::conditioning, fclib_matrix_info::determinant, fclib_matrix::i, fclib_matrix::info, IO, fclib_matrix::m, fclib_matrix::n, fclib_matrix::nz, fclib_matrix::nzmax, fclib_-matrix::p, fclib_matrix_info::rank, and fclib_matrix::x.

Referenced by fclib_write_global(), and fclib_write_local().

8.5.3.3 struct fclib_matrix∗ read_matrix (hid_t *id*)

read matrix

Definition at line 96 of file fclib.c.

References ASSERT, fclib_matrix_info::comment, fclib_matrix_info::conditioning, fclib_matrix_info::determinant, fclib_matrix::i, fclib_matrix::info, IO, fclib_matrix::m, MM, fclib_matrix::n, fclib_matrix::nz, fclib_matrix::nzmax, fclib_-matrix::p, fclib_matrix_info::rank, and fclib_matrix::x.

Referenced by fclib_read_global(), and fclib_read_local().

8.5.3.4 static void write_global_vectors (hid_t *id,* struct fclib_global ∗ *problem*) [static]

write global vectors

Definition at line 156 of file fclib.c.

References ASSERT, fclib_global::b, fclib_global::f, fclib_global::G, fclib_global::H, IO, fclib_matrix::m, fclib_global-::M, fclib_global::mu, fclib_matrix::n, fclib_global::spacedim, and fclib_global::w.

Referenced by fclib_write_global().

8.5.3.5 static void read_global_vectors (hid_t *id,* struct fclib_global ∗ *problem*) [static]

read global vectors

Definition at line 180 of file fclib.c.

References ASSERT, fclib_global::b, fclib_global::f, fclib_global::G, fclib_global::H, IO, fclib_matrix::m, fclib_global-::M, MM, fclib_global::mu, fclib_matrix::n, fclib_global::spacedim, and fclib_global::w.

Referenced by fclib_read_global().

8.5.3.6 static void write_local_vectors (hid_t *id,* struct **fclib_local** ∗ *problem*) [static]

write local vectors

Definition at line 199 of file fclib.c.

References ASSERT, IO, fclib_matrix::m, fclib_local::mu, fclib_local::q, fclib_local::R, fclib_local::s, fclib_local-::spacedim, fclib_local::V, and fclib_local::W.

Referenced by fclib_write_local().

8.5.3.7 static void read_local_vectors (hid_t *id,* struct **fclib_local** ∗ *problem*) [static]

read local vectors

Definition at line 220 of file fclib.c.

References ASSERT, IO, fclib_matrix::m, MM, fclib_local::mu, fclib_local::q, fclib_local::R, fclib_local::s, fclib_local-::spacedim, and fclib_local::W.

Referenced by fclib_read_local().

8.5.3.8 static void write_problem_info (hid_t *id,* struct **fclib_info** ∗ *info*) [static]

write problem info

Definition at line 237 of file fclib.c.

References fclib_info::description, IO, fclib_info::math_info, and fclib_info::title.

Referenced by fclib_write_global(), and fclib_write_local().

8.5.3.9 static struct **fclib_info**∗ read_problem_info (hid_t *id*) [static]

read problem info

Definition at line 245 of file fclib.c.

References fclib_info::description, IO, fclib_info::math_info, MM, and fclib_info::title.

Referenced by fclib_read_global(), and fclib_read_local().

8.5.3.10 static void write_solution (hid_t *id,* struct **fclib_solution** ∗ *solution,* hsize_t *nv,* hsize_t *nr,* hsize_t *nl*) [static]

write solution

Definition at line 282 of file fclib.c.

References ASSERT, IO, fclib_solution::l, fclib_solution::r, fclib_solution::u, and fclib_solution::v.

Referenced by fclib_write_guesses(), and fclib_write_solution().

8.5.3.11 static void read_solution (hid_t *id,* hsize_t *nv,* hsize_t *nr,* hsize_t *nl,* struct **fclib_solution** ∗ *solution*) [static]

read solution

Definition at line 293 of file fclib.c.

References ASSERT, IO, fclib_solution::l, MM, fclib_solution::r, fclib_solution::u, and fclib_solution::v.

Referenced by fclib_read_guesses(), and fclib_read_solution().

8.5.3.12 static int read_nvnunrnl (hid_t *file_id,* int ∗ *nv,* int ∗ *nr,* int ∗ *nl*) [static]

read solution sizes

Definition at line 317 of file fclib.c.

References IO.

Referenced by fclib_read_guesses(), fclib_read_solution(), fclib_write_guesses(), and fclib_write_solution().

8.5.3.13 static void delete_matrix_info (struct **fclib_matrix_info** ∗ *info*) [static]

delete matrix info

Definition at line 349 of file fclib.c.

References fclib_matrix_info::comment.

Referenced by delete_matrix().

8.5.3.14 static void delete_matrix (struct **fclib_matrix** ∗ *mat*) [static]

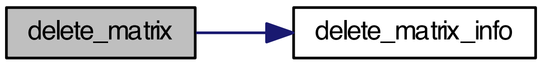
delete matrix

Definition at line 359 of file fclib.c.

References delete_matrix_info(), fclib_matrix::i, fclib_matrix::info, fclib_matrix::p, and fclib_matrix::x.

Referenced by fclib_delete_global(), and fclib_delete_local().

Here is the call graph for this function:



8.5.3.15 static void delete_info (struct **fclib_info** ∗ *info*) [static]

delete problem info

Definition at line 372 of file fclib.c.

References fclib_info::description, fclib_info::math_info, and fclib_info::title.

Referenced by fclib_delete_global(), and fclib_delete_local().

8.5.3.16 int fclib_write_global (struct **fclib_global** ∗ *problem,* const char ∗ *path*)

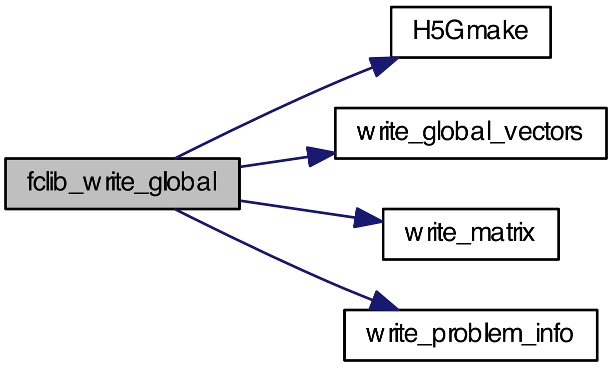
write global problem; return 1 on success, 0 on failure

Definition at line 386 of file fclib.c.

References ASSERT, fclib_global::G, fclib_global::H, H5Gmake(), fclib_global::info, IO, fclib_global::M, fclib_global-::spacedim, write_global_vectors(), write_matrix(), and write_problem_info().

Referenced by main().

Here is the call graph for this function:



8.5.3.17 int fclib_write_local (struct **fclib_local** ∗ *problem,* const char ∗ *path*)

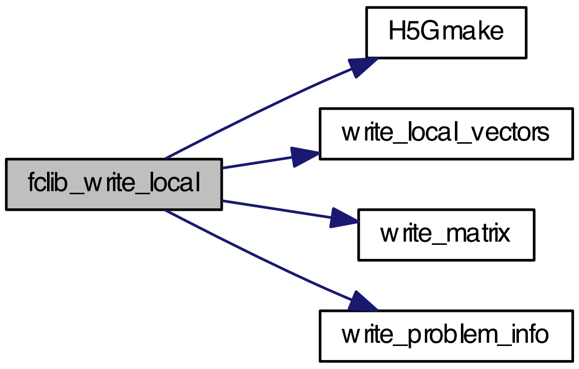
write local problem; return 1 on success, 0 on failure

Definition at line 454 of file fclib.c.

References ASSERT, H5Gmake(), fclib_local::info, IO, fclib_local::R, fclib_local::spacedim, fclib_local::V, fclib_local-::W, write_local_vectors(), write_matrix(), and write_problem_info().

Referenced by main().

Here is the call graph for this function:



8.5.3.18 int fclib_write_solution (struct **fclib_solution** ∗ *solution,* const char ∗ *path*)

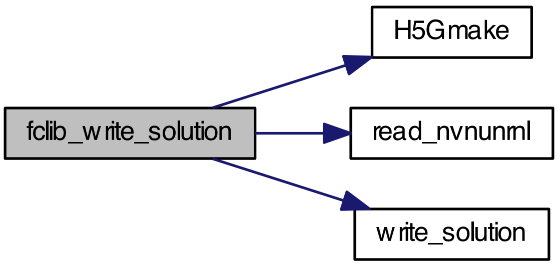
write solution; return 1 on success, 0 on failure

Definition at line 522 of file fclib.c.

References H5Gmake(), IO, read_nvnunrnl(), and write_solution().

Referenced by main().

Here is the call graph for this function:



8.5.3.19 int fclib_write_guesses (int *number_of_guesses,* struct **fclib_solution** ∗ *guesses,* const char ∗ *path*)

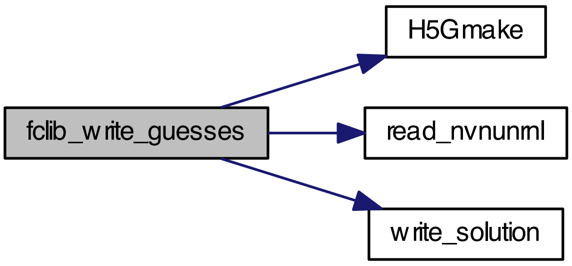
write initial guesses; return 1 on success, 0 on failure

Definition at line 562 of file fclib.c.

References H5Gmake(), IO, read_nvnunrnl(), and write_solution().

Referenced by main().

Here is the call graph for this function:



8.5.3.20 struct **fclib_global**∗ fclib_read_global (const char ∗ *path*)

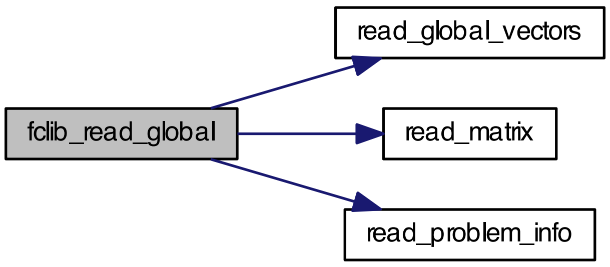
read global problem; return problem on success; NULL on failure

Definition at line 612 of file fclib.c.

References fclib_global::G, fclib_global::H, fclib_global::info, IO, fclib_global::M, MM, read_global_vectors(), read_-matrix(), read_problem_info(), and fclib_global::spacedim.

Referenced by main().

Here is the call graph for this function:



8.5.3.21 struct **fclib_local**∗ fclib_read_local (const char ∗ *path*)

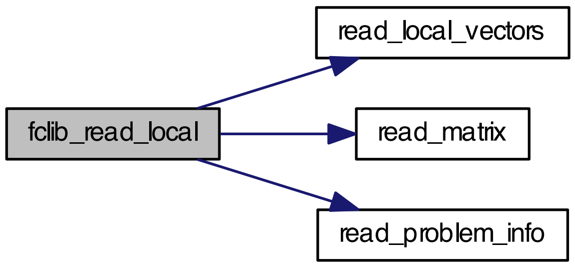
read local problem; return problem on success; NULL on failure

Definition at line 662 of file fclib.c.

References fclib_local::info, IO, MM, fclib_local::R, read_local_vectors(), read_matrix(), read_problem_info(), fclib_-local::spacedim, fclib_local::V, and fclib_local::W.

Referenced by main().

Here is the call graph for this function:



8.5.3.22 struct **fclib_solution**∗ fclib_read_solution (const char ∗ *path*)

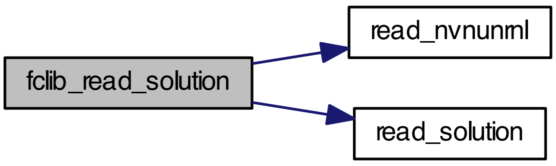
read solution; return solution on success; NULL on failure

Definition at line 712 of file fclib.c.

References IO, MM, read_nvnunrnl(), and read_solution().

Referenced by main().

Here is the call graph for this function:



8.5.3.23 struct **fclib_solution**∗ fclib_read_guesses (const char ∗ *path,* int ∗ *number_of_guesses*)

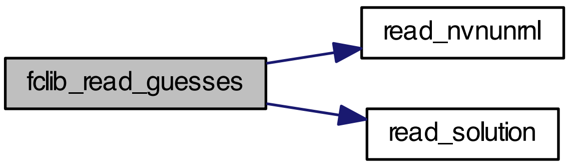
read initial guesses; return vector of guesses on success; NULL on failure; output numebr of guesses in the variable pointed by 'number_of_guesses'

Definition at line 740 of file fclib.c.

References IO, MM, read_nvnunrnl(), and read_solution().

Referenced by main().

Here is the call graph for this function:



8.5.3.24 void fclib_delete_global (struct **fclib_global** ∗ *problem*)

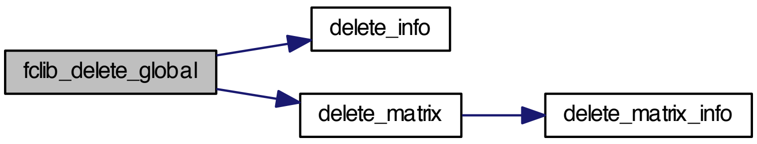
delete global problem

Definition at line 780 of file fclib.c.

References fclib_global::b, delete_info(), delete_matrix(), fclib_global::f, fclib_global::G, fclib_global::H, fclib_global-::info, fclib_global::M, fclib_global::mu, and fclib_global::w.

Referenced by main().

Here is the call graph for this function:



8.5.3.25 void fclib_delete_local (struct **fclib_local** ∗ *problem*)

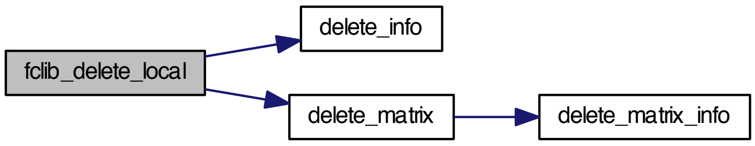
delete local problem

Definition at line 793 of file fclib.c.

References delete_info(), delete_matrix(), fclib_local::info, fclib_local::mu, fclib_local::q, fclib_local::R, fclib_local::s, fclib_local::V, and fclib_local::W.

Referenced by main().

Here is the call graph for this function:



8.5.3.26 void fclib_delete_solutions (struct **fclib_solution** ∗ *data,* int *count*)

delete solutions or guesses

Definition at line 805 of file fclib.c.

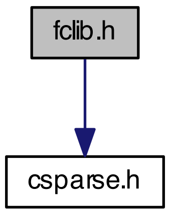
References fclib_solution::l, fclib_solution::r, fclib_solution::u, and fclib_solution::v.

Referenced by main().

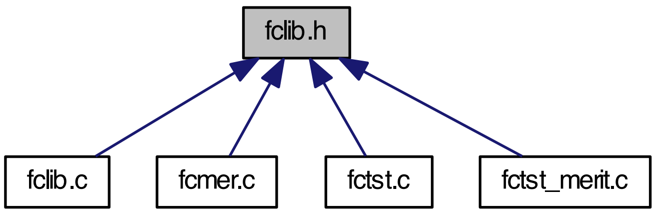
8.6 fclib.h File Reference

```
#include "csparse.h"
```

Include dependency graph for fclib.h:



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



Classes

- struct fclib_info

  This structure allows the user to enter a problem information as a title, a short description and known mathematical properties of the problem.
- struct fclib_matrix_info

  This structure allows the user to enter a description for a given matrix (comment, conditionning, determinant, rank.) if they are known.
- struct fclib_matrix

  matrix in compressed row/column or triplet form
- struct fclib_global

  The global frictional contact problem defined by.
- struct fclib_local

  The local frictional contact problem defined by.
- struct fclib_solution

  A solution or a guess for the frictional contact problem.

Enumerations

- enum fclib_merit { MERIT_1, MERIT_2 }

  MERIT_1 is a implementation of the merit function based on the natural map for a SOCCP.

Functions

- int fclib_write_global (struct fclib_global ∗problem, const char ∗path)

  write global problem; return 1 on success, 0 on failure
- int fclib_write_local (struct fclib_local ∗problem, const char ∗path)

  write local problem; return 1 on success, 0 on failure
- int fclib_write_solution (struct fclib_solution ∗solution, const char ∗path)

  write solution; return 1 on success, 0 on failure
- int fclib_write_guesses (int number_of_guesses, struct fclib_solution ∗guesses, const char ∗path)

  write initial guesses; return 1 on success, 0 on failure
- struct fclib_global ∗ fclib_read_global (const char ∗path)

  read global problem; return problem on success; NULL on failure
- struct fclib_local ∗ fclib_read_local (const char ∗path)

  read local problem; return problem on success; NULL on failure
- struct fclib_solution ∗ fclib_read_solution (const char ∗path)

  read solution; return solution on success; NULL on failure
- struct fclib_solution ∗ fclib_read_guesses (const char ∗path, int ∗number_of_guesses)

  read initial guesses; return vector of guesses on success; NULL on failure; output numebr of guesses in the variable pointed by 'number_of_guesses'
- double fclib_merit_global (struct fclib_global ∗problem, enum fclib_merit merit, struct fclib_solution ∗solution)

  calculate merit function for a global problem
- double fclib_merit_local (struct fclib_local ∗problem, enum fclib_merit merit, struct fclib_solution ∗solution)

  calculate merit function for a local problem
- void fclib_delete_global (struct fclib_global ∗problem)

  delete global problem
- void fclib_delete_local (struct fclib_local ∗problem)

  delete local problem
- void fclib_delete_solutions (struct fclib_solution ∗data, int count)

  delete solutions or guesses

8.6.1 Detailed Description

frictional contact library interface

This C API provides functions to read and write Frictional contact problemes in HDF5 format Two kind of problem are considered Given

- a symmetric positive semi–definite matrix $W \in \mathbb{R}^{m\times m}$
- a vector $q \in \mathbb{R}^m$
- a vector of coefficients of friction $\mu \in \mathbb{R}^{n_c}$

the local FC problem is to find two vectors $u \in \mathbb{R}^m$, the relative local velocity and $r \in \mathbb{R}^m$, the contact forces denoted by $\mathrm{FC}(W,q,\mu)$ such that

$$\begin{cases} \hat{u} = Wr + q + \left[\left[\begin{array}{ccc} \mu^\alpha \|u^\alpha_T\| & 0 & 0 \end{array}\right]^T, \alpha = 1\ldots n_c\right]^T \\ C^\star_\mu \ni \hat{u} \perp r \in C_\mu \end{cases}$$

where the Coulomb friction cone for a contact α is defined by

$$C^\alpha_{\mu^\alpha} = \{r^\alpha, \|r^\alpha_T\| \leq \mu^\alpha |r^\alpha_N|\}*$$

and the set $C^{\alpha,\star}_{\mu^\alpha}$ is its dual. We are also dealing with global FC problem defined by Given

- a symmetric positive definite matrix $M \in {\rm I\!R}^{n\times n}$
- a vector $f \in {\rm I\!R}^{n}$,
- a matrix $H \in {\rm I\!R}^{n\times m}$
- a matrix $G \in {\rm I\!R}^{n\times p}$
- a vector $w \in {\rm I\!R}^{m}$,
- a vector $b \in {\rm I\!R}^{p}$,
- a vector of coefficients of friction $\mu \in {\rm I\!R}^{n_c}$

the Global Mixed 3DFC problem is to find four vectors $v \in {\rm I\!R}^{n}$, $u \in {\rm I\!R}^{m}$, $r \in {\rm I\!R}^{m}$ and $\lambda \in {\rm I\!R}^{p}$ denoted by $\mathrm{GM3DFC}(M,H,G,w,b,\mu)$ such that

$$\begin{cases} Mv = Hr + G\lambda + f \\ G^T v + b = 0 \\ \hat{u} = H^T v + w + \left[\left[\begin{array}{ccc} \mu \|u^\alpha_T\| & 0 & 0 \end{array}\right]^T, \alpha = 1\dots n_c\right]^T \\ C^\star_\mu \ni \hat{u} \perp r \in C_\mu \end{cases}$$

Definition in file fclib.h.

8.6.2 Enumeration Type Documentation

8.6.2.1 enum **fclib_merit**

MERIT_1 is a implementation of the merit function based on the natural map for a SOCCP.

Enumerator

MERIT_1

MERIT_2

Definition at line 269 of file fclib.h.

8.6.3 Function Documentation

8.6.3.1 int fclib_write_global (struct **fclib_global** ∗ *problem,* const char ∗ *path*)

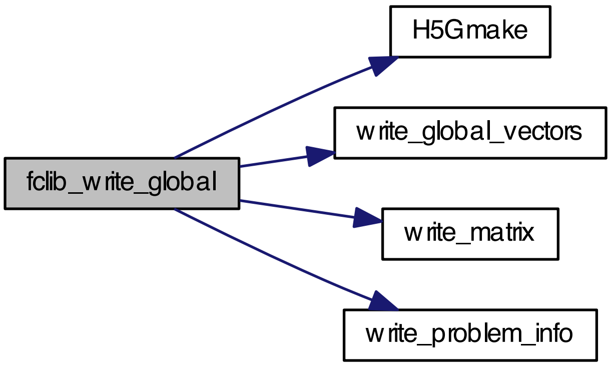
write global problem; return 1 on success, 0 on failure

Definition at line 386 of file fclib.c.

References ASSERT, fclib_global::G, fclib_global::H, H5Gmake(), fclib_global::info, IO, fclib_global::M, fclib_global::spacedim, write_global_vectors(), write_matrix(), and write_problem_info().

Referenced by main().

Here is the call graph for this function:



8.6.3.2 int fclib_write_local (struct **fclib_local** ∗ *problem,* const char ∗ *path*)

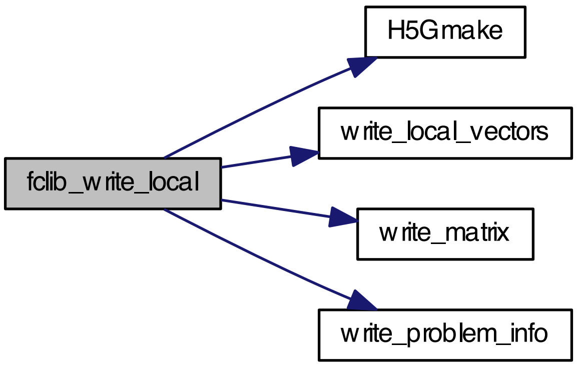
write local problem; return 1 on success, 0 on failure

Definition at line 454 of file fclib.c.

References ASSERT, H5Gmake(), fclib_local::info, IO, fclib_local::R, fclib_local::spacedim, fclib_local::V, fclib_local-::W, write_local_vectors(), write_matrix(), and write_problem_info().

Referenced by main().

Here is the call graph for this function:



8.6.3.3 int fclib_write_solution (struct **fclib_solution** ∗ *solution,* const char ∗ *path*)

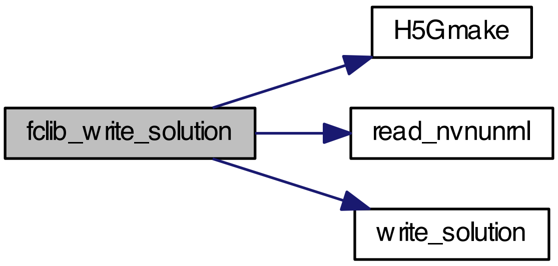
write solution; return 1 on success, 0 on failure

Definition at line 522 of file fclib.c.

References H5Gmake(), IO, read_nvnunrnl(), and write_solution().

Referenced by main().

Here is the call graph for this function:



8.6.3.4 int fclib_write_guesses (int *number_of_guesses,* struct **fclib_solution** ∗ *guesses,* const char ∗ *path*)

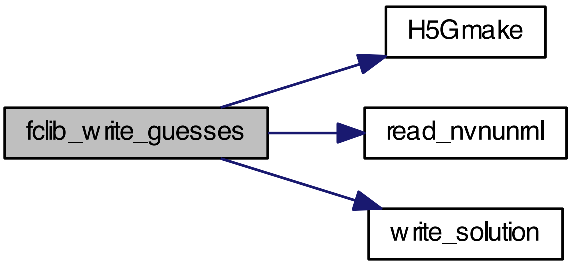
write initial guesses; return 1 on success, 0 on failure

Definition at line 562 of file fclib.c.

References H5Gmake(), IO, read_nvnunrnl(), and write_solution().

Referenced by main().

Here is the call graph for this function:



8.6.3.5 struct **fclib_global**∗ fclib_read_global (const char ∗ *path*)

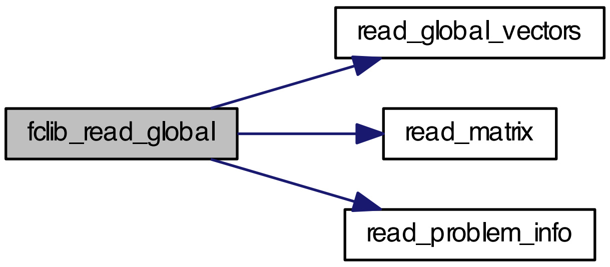
read global problem; return problem on success; NULL on failure

Definition at line 612 of file fclib.c.

References fclib_global::G, fclib_global::H, fclib_global::info, IO, fclib_global::M, MM, read_global_vectors(), read_-matrix(), read_problem_info(), and fclib_global::spacedim.

Referenced by main().

Here is the call graph for this function:



8.6.3.6 struct **fclib_local**∗ fclib_read_local (const char ∗ *path*)

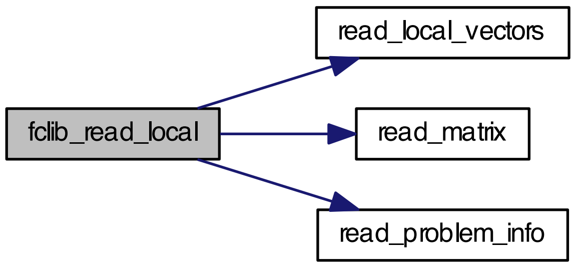
read local problem; return problem on success; NULL on failure

Definition at line 662 of file fclib.c.

References fclib_local::info, IO, MM, fclib_local::R, read_local_vectors(), read_matrix(), read_problem_info(), fclib_-local::spacedim, fclib_local::V, and fclib_local::W.

Referenced by main().

Here is the call graph for this function:



8.6.3.7 struct **fclib_solution**∗ fclib_read_solution (const char ∗ *path*)

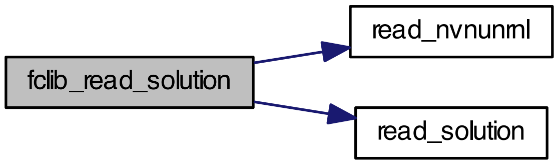
read solution; return solution on success; NULL on failure

Definition at line 712 of file fclib.c.

References IO, MM, read_nvnunrnl(), and read_solution().

Referenced by main().

Here is the call graph for this function:



8.6.3.8 struct **fclib_solution**∗ fclib_read_guesses (const char ∗ *path,* int ∗ *number_of_guesses*)

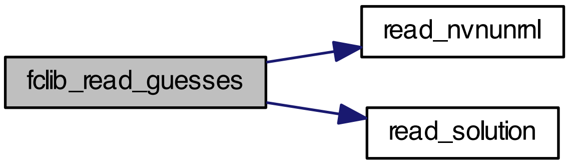
read initial guesses; return vector of guesses on success; NULL on failure; output numebr of guesses in the variable pointed by 'number_of_guesses'

Definition at line 740 of file fclib.c.

References IO, MM, read_nvnunrnl(), and read_solution().

Referenced by main().

Here is the call graph for this function:



8.6.3.9 double fclib_merit_global (struct **fclib_global** ∗ *problem,* enum **fclib_merit** *merit,* struct **fclib_solution** ∗ *solution*)

calculate merit function for a global problem

Definition at line 80 of file fcmer.c.

Referenced by main().

8.6.3.10 double fclib_merit_local (struct **fclib_local** ∗ *problem,* enum **fclib_merit** *merit,* struct **fclib_solution** ∗ *solution*)

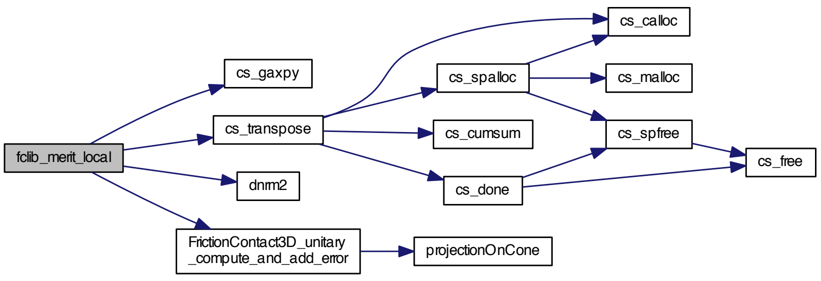
calculate merit function for a local problem

Definition at line 86 of file fcmer.c.

References cs_gaxpy(), cs_transpose(), dnrm2(), FrictionContact3D_unitary_compute_and_add_error(), fclib_-matrix::i, fclib_solution::l, MERIT_1, fclib_local::mu, fclib_matrix::n, fclib_local::q, fclib_local::R, fclib_solution::r, fclib_local::s, fclib_local::spacedim, fclib_solution::u, fclib_local::V, fclib_solution::v, and fclib_local::W.

Referenced by main().

Here is the call graph for this function:



8.6.3.11 void fclib_delete_global (struct **fclib_global** ∗ *problem*)

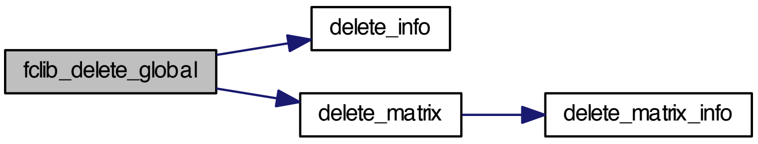
delete global problem

Definition at line 780 of file fclib.c.

References fclib_global::b, delete_info(), delete_matrix(), fclib_global::f, fclib_global::G, fclib_global::H, fclib_global-::info, fclib_global::M, fclib_global::mu, and fclib_global::w.

Referenced by main().

Here is the call graph for this function:



8.6.3.12 void fclib_delete_local (struct **fclib_local** ∗ *problem*)

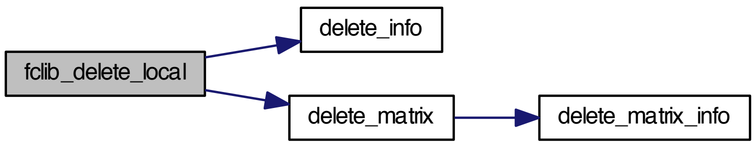
delete local problem

Definition at line 793 of file fclib.c.

References delete_info(), delete_matrix(), fclib_local::info, fclib_local::mu, fclib_local::q, fclib_local::R, fclib_local::s, fclib_local::V, and fclib_local::W.

Referenced by main().

Here is the call graph for this function:



8.6.3.13 void fclib_delete_solutions (struct **fclib_solution** ∗ *data*, int *count*)

delete solutions or guesses

Definition at line 805 of file fclib.c.

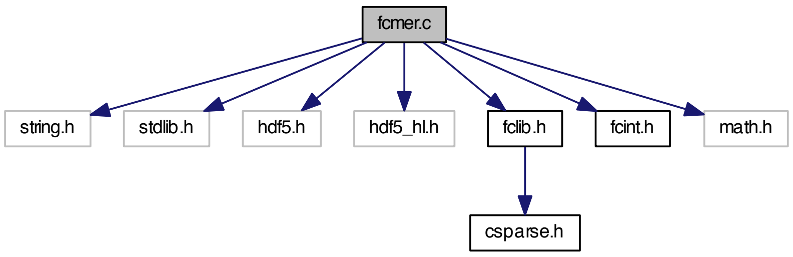
References fclib_solution::l, fclib_solution::r, fclib_solution::u, and fclib_solution::v.

Referenced by main().

8.7 fcmer.c File Reference

```
#include <string.h>
#include <stdlib.h>
#include <hdf5.h>
#include <hdf5_hl.h>
#include "fclib.h"
#include "fcint.h"
#include "math.h"
```

Include dependency graph for fcmer.c:



Functions

- double dnrm2 (double ∗v, int n)
- void projectionOnCone (double ∗r, double mu)
- void FrictionContact3D_unitary_compute_and_add_error (double ∗z, double ∗w, double mu, double ∗error)
- double fclib_merit_global (struct fclib_global ∗problem, enum fclib_merit merit, struct fclib_solution ∗solution)

  calculate merit function for a global problem
- double fclib_merit_local (struct fclib_local ∗problem, enum fclib_merit merit, struct fclib_solution ∗solution)

calculate merit function for a local problem

8.7.1 Function Documentation

8.7.1.1 double dnrm2 (double ∗ *v*, int *n*)

Definition at line 27 of file fcmer.c.

Referenced by fclib_merit_local().

8.7.1.2 void projectionOnCone (double ∗ *r*, double *mu*)

Definition at line 37 of file fcmer.c.

Referenced by FrictionContact3D_unitary_compute_and_add_error().

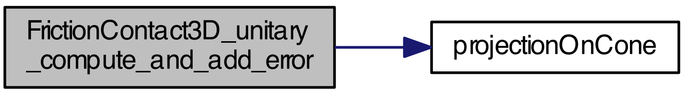
8.7.1.3 void FrictionContact3D_unitary_compute_and_add_error (double ∗ *z*, double ∗ *w*, double *mu*, double ∗ *error*)

Definition at line 61 of file fcmer.c.

References projectionOnCone().

Referenced by fclib_merit_local().

Here is the call graph for this function:



8.7.1.4 double fclib_merit_global (struct **fclib_global** ∗ *problem*, enum **fclib_merit** *merit*, struct **fclib_solution** ∗ *solution*)

calculate merit function for a global problem

Definition at line 80 of file fcmer.c.

Referenced by main().

8.7.1.5 double fclib_merit_local (struct **fclib_local** ∗ *problem*, enum **fclib_merit** *merit*, struct **fclib_solution** ∗ *solution*)

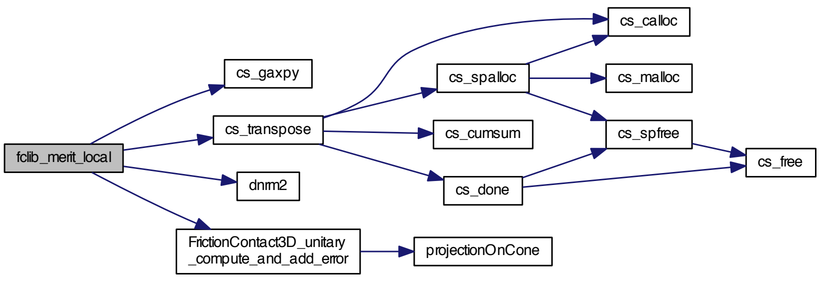
calculate merit function for a local problem

Definition at line 86 of file fcmer.c.

References cs_gaxpy(), cs_transpose(), dnrm2(), FrictionContact3D_unitary_compute_and_add_error(), fclib_-matrix::i, fclib_solution::l, MERIT_1, fclib_local::mu, fclib_matrix::n, fclib_local::q, fclib_local::R, fclib_solution::r, fclib_local::s, fclib_local::spacedim, fclib_solution::u, fclib_local::V, fclib_solution::v, and fclib_local::W.

Referenced by main().

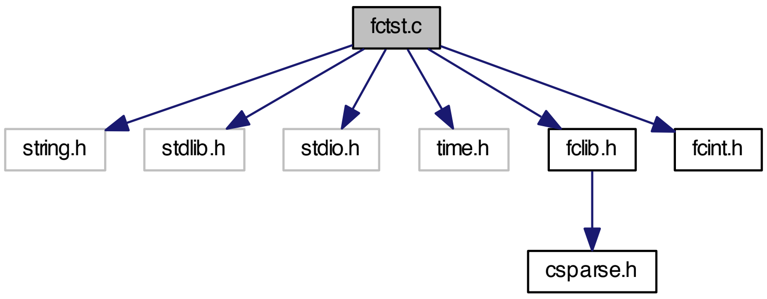
Here is the call graph for this function:



8.8 fctst.c File Reference

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include "fclib.h"
#include "fcint.h"
```

Include dependency graph for fctst.c:



Functions

- static struct fclib_matrix_info ∗ matrix_info (struct fclib_matrix ∗mat, char ∗comment)
- static struct fclib_matrix ∗ random_matrix (int m, int n)
- static double ∗ random_vector (int n)
- static struct fclib_info ∗ problem_info (char ∗title, char ∗desc, char ∗math)
- static struct fclib_global ∗ random_global_problem (int global_dofs, int contact_points, int neq)
- static struct fclib_solution ∗ random_global_solutions (struct fclib_global ∗problem, int count)
- static struct fclib_local ∗ random_local_problem (int contact_points, int neq)
- static struct fclib_solution ∗ random_local_solutions (struct fclib_local ∗problem, int count)
- static int compare_matrix_infos (struct fclib_matrix_info ∗a, struct fclib_matrix_info ∗b)
- static int compare_matrices (char ∗name, struct fclib_matrix ∗a, struct fclib_matrix ∗b)
- static int compare_vectors (char ∗name, int n, double ∗a, double ∗b)

- static int compare_infos (struct fclib_info ∗a, struct fclib_info ∗b)
- static int compare_global_problems (struct fclib_global ∗a, struct fclib_global ∗b)
- static int compare_local_problems (struct fclib_local ∗a, struct fclib_local ∗b)
- static int compare_solutions (struct fclib_solution ∗a, struct fclib_solution ∗b, int nv, int nr, int nl)
- int main (int argc, char ∗∗argv)

8.8.1 Function Documentation

8.8.1.1 static struct **fclib_matrix_info**∗ matrix_info (struct **fclib_matrix** ∗ *mat,* char ∗ *comment*) [static]

Definition at line 33 of file fctst.c.

References fclib_matrix_info::comment, fclib_matrix_info::conditioning, fclib_matrix_info::determinant, fclib_matrix-::m, MM, and fclib_matrix_info::rank.

Referenced by random_matrix().

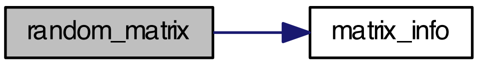
8.8.1.2 static struct **fclib_matrix**∗ random_matrix (int *m,* int *n*) [static]

Definition at line 48 of file fctst.c.

References fclib_matrix::i, fclib_matrix::info, fclib_matrix::m, matrix_info(), MM, fclib_matrix::n, fclib_matrix::nz, fclib-_matrix::nzmax, fclib_matrix::p, and fclib_matrix::x.

Referenced by random_global_problem(), and random_local_problem().

Here is the call graph for this function:



8.8.1.3 static double∗ random_vector (int *n*) [static]

Definition at line 91 of file fctst.c.

References MM.

Referenced by random_global_problem(), random_global_solutions(), random_local_problem(), and random_local-_solutions().

8.8.1.4 static struct **fclib_info**∗ problem_info (char ∗ *title,* char ∗ *desc,* char ∗ *math*) [static]

Definition at line 102 of file fctst.c.

References fclib_info::description, fclib_info::math_info, MM, and fclib_info::title.

Referenced by random_global_problem(), and random_local_problem().

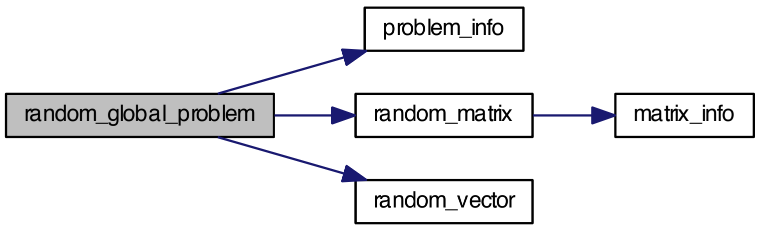
8.8.1.5 static struct **fclib_global**∗ random_global_problem (int *global_dofs,* int *contact_points,* int *neq*) [static]

Definition at line 118 of file fctst.c.

References fclib_global::b, fclib_global::f, fclib_global::G, fclib_global::H, fclib_global::info, fclib_global::M, MM, fclib-_global::mu, fclib_matrix::n, problem_info(), random_matrix(), random_vector(), fclib_global::spacedim, and fclib_-global::w.

Referenced by main().

Here is the call graph for this function:



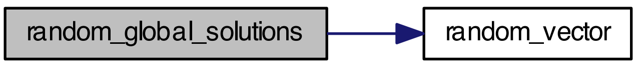
8.8.1.6 static struct **fclib_solution**∗ random_global_solutions (struct **fclib_global** ∗ *problem,* int *count*) [static]

Definition at line 141 of file fctst.c.

References fclib_global::G, fclib_global::H, fclib_solution::l, fclib_global::M, MM, fclib_matrix::n, fclib_solution::r, random_vector(), fclib_solution::u, and fclib_solution::v.

Referenced by main().

Here is the call graph for this function:



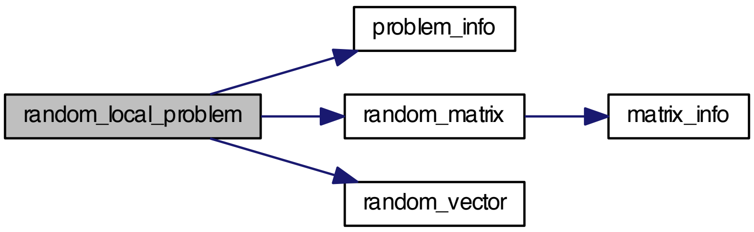
8.8.1.7 static struct **fclib_local**∗ random_local_problem (int *contact_points,* int *neq*) [static]

Definition at line 161 of file fctst.c.

References fclib_local::info, MM, fclib_local::mu, problem_info(), fclib_local::q, fclib_local::R, random_matrix(), random_vector(), fclib_local::s, fclib_local::spacedim, fclib_local::V, and fclib_local::W.

Referenced by main().

Here is the call graph for this function:



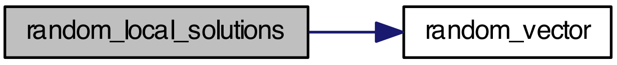
8.8.1.8 static struct **fclib_solution**∗ random_local_solutions (struct **fclib_local** ∗ *problem,* int *count*) [static]

Definition at line 189 of file fctst.c.

References fclib_solution::l, MM, fclib_matrix::n, fclib_local::R, fclib_solution::r, random_vector(), fclib_solution::u, fclib_solution::v, and fclib_local::W.

Referenced by main().

Here is the call graph for this function:



8.8.1.9 static int compare_matrix_infos (struct **fclib_matrix_info** ∗ *a,* struct **fclib_matrix_info** ∗ *b*) [static]

Definition at line 209 of file fctst.c.

References fclib_matrix_info::comment, fclib_matrix_info::conditioning, fclib_matrix_info::determinant, and fclib_-matrix_info::rank.

Referenced by compare_matrices().

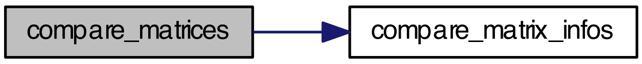
8.8.1.10 static int compare_matrices (char ∗ *name,* struct **fclib_matrix** ∗ *a,* struct **fclib_matrix** ∗ *b*) [static]

Definition at line 222 of file fctst.c.

References compare_matrix_infos(), fclib_matrix::i, fclib_matrix::info, fclib_matrix::m, fclib_matrix::n, fclib_matrix-::nz, fclib_matrix::nzmax, fclib_matrix::p, and fclib_matrix::x.

Referenced by compare_global_problems(), and compare_local_problems().

Here is the call graph for this function:



8.8.1.11 static int compare_vectors (char ∗ *name*, int *n*, double ∗ *a*, double ∗ *b*) `[static]`

Definition at line 327 of file fctst.c.

Referenced by compare_global_problems(), compare_local_problems(), and compare_solutions().

8.8.1.12 static int compare_infos (struct fclib_info ∗ *a*, struct fclib_info ∗ *b*) `[static]`

Definition at line 347 of file fctst.c.

References fclib_info::description, fclib_info::math_info, and fclib_info::title.

Referenced by compare_global_problems(), and compare_local_problems().

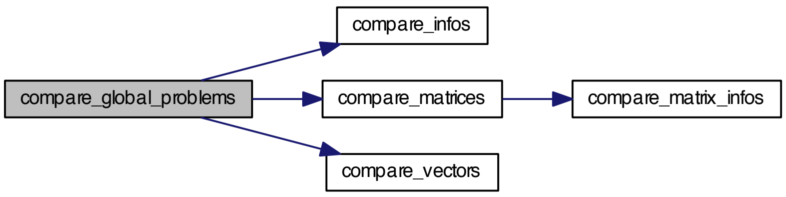
8.8.1.13 static int compare_global_problems (struct fclib_global ∗ *a*, struct fclib_global ∗ *b*) `[static]`

Definition at line 359 of file fctst.c.

References fclib_global::b, compare_infos(), compare_matrices(), compare_vectors(), fclib_global::f, fclib_global-::G, fclib_global::H, fclib_global::info, fclib_matrix::m, fclib_global::M, fclib_global::mu, fclib_matrix::n, fclib_global-::spacedim, and fclib_global::w.

Referenced by main().

Here is the call graph for this function:



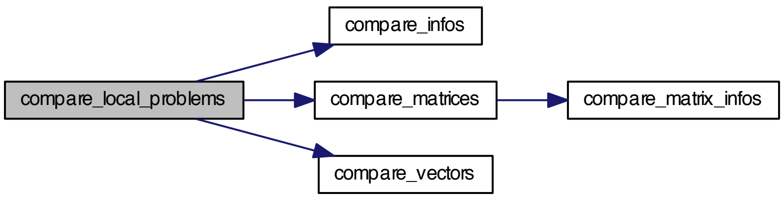
8.8.1.14 static int compare_local_problems (struct fclib_local ∗ *a*, struct fclib_local ∗ *b*) `[static]`

Definition at line 375 of file fctst.c.

References compare_infos(), compare_matrices(), compare_vectors(), fclib_local::info, fclib_local::mu, fclib_matrix-::n, fclib_local::q, fclib_local::R, fclib_local::s, fclib_local::spacedim, fclib_local::V, and fclib_local::W.

Referenced by main().

Here is the call graph for this function:



8.8.1.15 static int compare_solutions (struct **fclib_solution** ∗ *a*, struct **fclib_solution** ∗ *b*, int *nv*, int *nr*, int *nl*) [static]

Definition at line 390 of file fctst.c.

References compare_vectors(), fclib_solution::l, fclib_solution::r, fclib_solution::u, and fclib_solution::v.

Referenced by main().

Here is the call graph for this function:

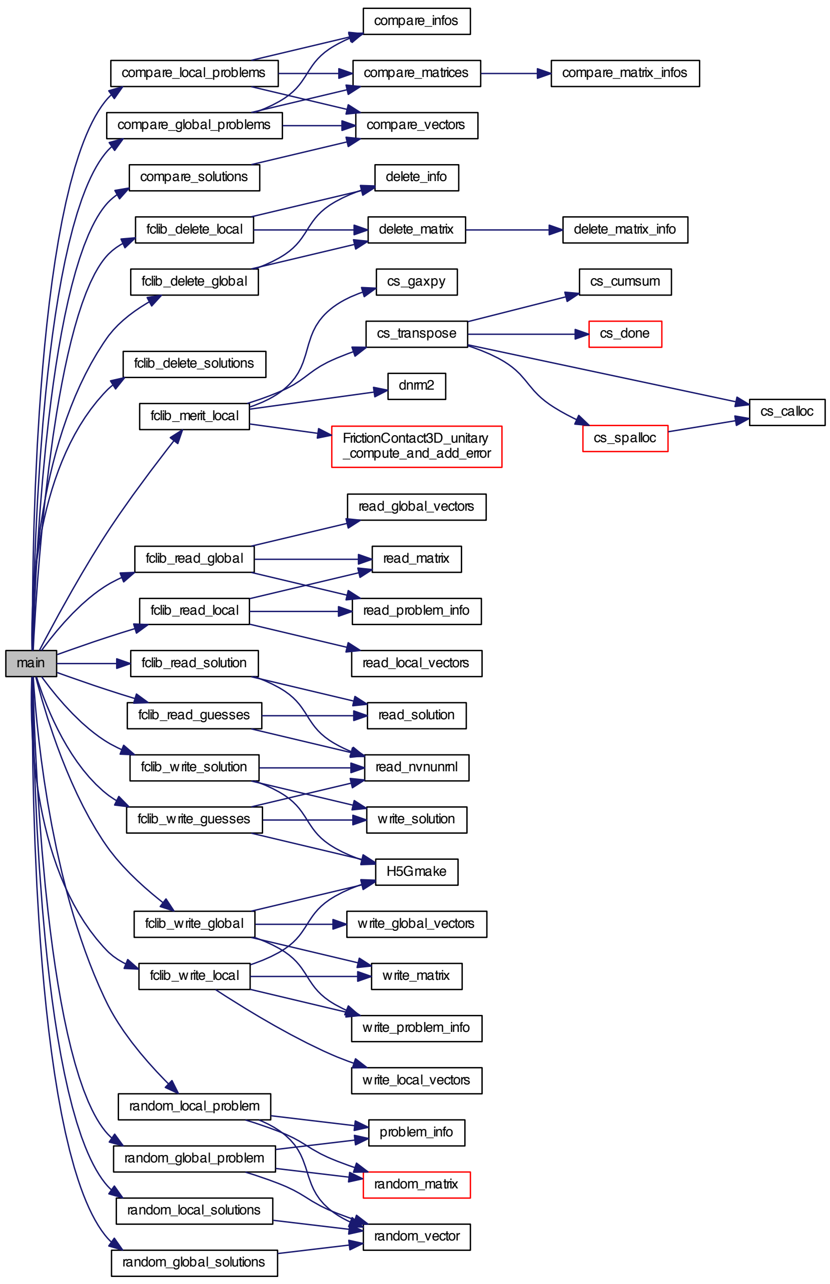
compare_solutions
compare_vectors

8.8.1.16 int main (int *argc*, char ∗∗ *argv*)

Definition at line 400 of file fctst.c.

References ASSERT, compare_global_problems(), compare_local_problems(), compare_solutions(), fclib_delete_-global(), fclib_delete_local(), fclib_delete_solutions(), fclib_merit_local(), fclib_read_global(), fclib_read_guesses(), fclib_read_local(), fclib_read_solution(), fclib_write_global(), fclib_write_guesses(), fclib_write_local(), fclib_write-_solution(), fclib_global::G, fclib_global::H, fclib_matrix::m, fclib_global::M, MERIT_1, fclib_matrix::n, fclib_local::-R, random_global_problem(), random_global_solutions(), random_local_problem(), random_local_solutions(), and fclib_local::W.

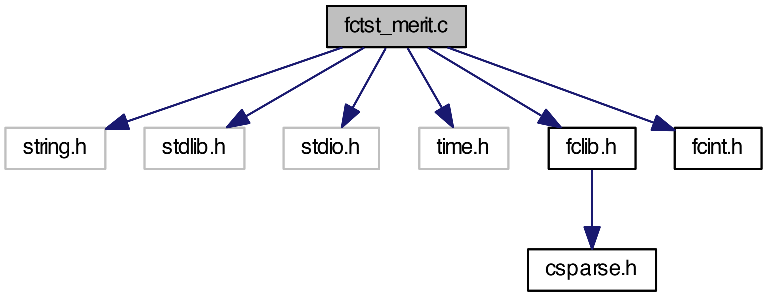
Here is the call graph for this function:



8.9 fctst_merit.c File Reference

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include "fclib.h"
#include "fcint.h"
```

Include dependency graph for fctst_merit.c:



Functions

- static struct fclib_matrix_info ∗ matrix_info (struct fclib_matrix ∗mat, char ∗comment)
- static struct fclib_matrix ∗ random_matrix (int m, int n)
- static double ∗ random_vector (int n)
- static struct fclib_info ∗ problem_info (char ∗title, char ∗desc, char ∗math)
- static struct fclib_global ∗ random_global_problem (int global_dofs, int contact_points, int neq)
- static struct fclib_solution ∗ random_global_solutions (struct fclib_global ∗problem, int count)
- static struct fclib_local ∗ random_local_problem (int contact_points, int neq)
- static struct fclib_solution ∗ random_local_solutions (struct fclib_local ∗problem, int count)
- static int compare_matrix_infos (struct fclib_matrix_info ∗a, struct fclib_matrix_info ∗b)
- static int compare_matrices (char ∗name, struct fclib_matrix ∗a, struct fclib_matrix ∗b)
- static int compare_vectors (char ∗name, int n, double ∗a, double ∗b)
- static int compare_infos (struct fclib_info ∗a, struct fclib_info ∗b)
- static int compare_global_problems (struct fclib_global ∗a, struct fclib_global ∗b)
- static int compare_local_problems (struct fclib_local ∗a, struct fclib_local ∗b)
- static int compare_solutions (struct fclib_solution ∗a, struct fclib_solution ∗b, int nv, int nr, int nl)
- int main (int argc, char ∗∗argv)

8.9.1 Function Documentation

8.9.1.1 static struct **fclib_matrix_info**∗ matrix_info (struct **fclib_matrix** ∗ *mat,* char ∗ *comment*) `[static]`

Definition at line 33 of file fctst_merit.c.

References fclib_matrix_info::comment, fclib_matrix_info::conditioning, fclib_matrix_info::determinant, fclib_matrix-::m, MM, and fclib_matrix_info::rank.

Referenced by random_matrix().

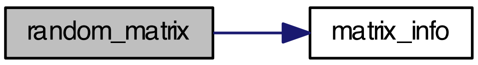
8.9.1.2 static struct **fclib_matrix**∗ random_matrix (int *m,* int *n*) `[static]`

Definition at line 48 of file fctst_merit.c.

References fclib_matrix::i, fclib_matrix::info, fclib_matrix::m, matrix_info(), MM, fclib_matrix::n, fclib_matrix::nz, fclib-_matrix::nzmax, fclib_matrix::p, and fclib_matrix::x.

Referenced by random_global_problem(), and random_local_problem().

Here is the call graph for this function:



8.9.1.3 static double∗ random_vector (int *n*) [static]

Definition at line 91 of file fctst_merit.c.

References MM.

Referenced by random_global_problem(), random_global_solutions(), random_local_problem(), and random_local-_solutions().

8.9.1.4 static struct **fclib_info**∗ problem_info (char ∗ *title,* char ∗ *desc,* char ∗ *math*) [static]

Definition at line 102 of file fctst_merit.c.

References fclib_info::description, fclib_info::math_info, MM, and fclib_info::title.

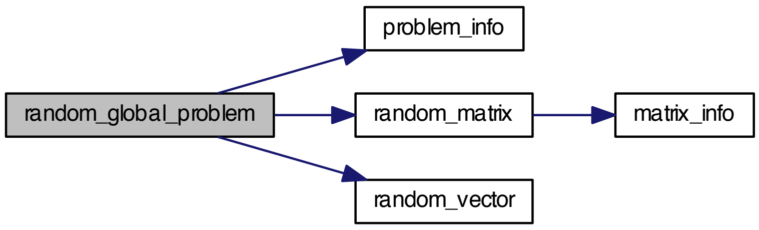
Referenced by random_global_problem(), and random_local_problem().

8.9.1.5 static struct **fclib_global**∗ random_global_problem (int *global_dofs,* int *contact_points,* int *neq*) [static]

Definition at line 118 of file fctst_merit.c.

References fclib_global::b, fclib_global::f, fclib_global::G, fclib_global::H, fclib_global::info, fclib_global::M, MM, fclib-_global::mu, fclib_matrix::n, problem_info(), random_matrix(), random_vector(), fclib_global::spacedim, and fclib_-global::w.

Here is the call graph for this function:

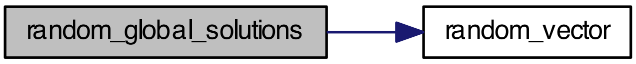


8.9.1.6 static struct **fclib_solution**∗ random_global_solutions (struct **fclib_global** ∗ *problem,* int *count*) [static]

Definition at line 141 of file fctst_merit.c.

References fclib_global::G, fclib_global::H, fclib_solution::l, fclib_global::M, MM, fclib_matrix::n, fclib_solution::r, random_vector(), fclib_solution::u, and fclib_solution::v.

Here is the call graph for this function:

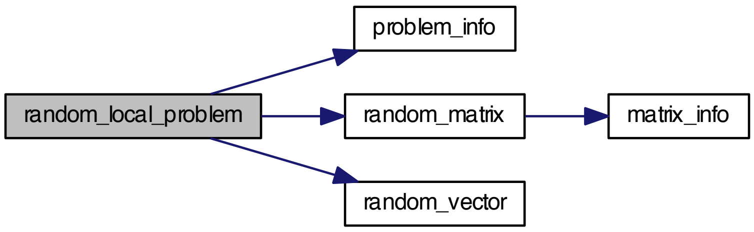


8.9.1.7 static struct **fclib_local**∗ random_local_problem (int *contact_points,* int *neq*) [static]

Definition at line 161 of file fctst_merit.c.

References fclib_local::info, MM, fclib_local::mu, problem_info(), fclib_local::q, fclib_local::R, random_matrix(), random_vector(), fclib_local::s, fclib_local::spacedim, fclib_local::V, and fclib_local::W.

Here is the call graph for this function:

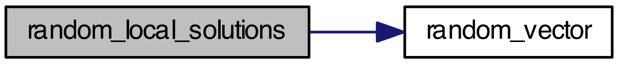


8.9.1.8 static struct **fclib_solution**∗ random_local_solutions (struct **fclib_local** ∗ *problem,* int *count*) [static]

Definition at line 189 of file fctst_merit.c.

References fclib_solution::l, MM, fclib_matrix::n, fclib_local::R, fclib_solution::r, random_vector(), fclib_solution::u, fclib_solution::v, and fclib_local::W.

Here is the call graph for this function:



8.9.1.9 static int compare_matrix_infos (struct **fclib_matrix_info** ∗ *a,* struct **fclib_matrix_info** ∗ *b*) [static]

Definition at line 209 of file fctst_merit.c.

References fclib_matrix_info::comment, fclib_matrix_info::conditioning, fclib_matrix_info::determinant, and fclib_-matrix_info::rank.

Referenced by compare_matrices().

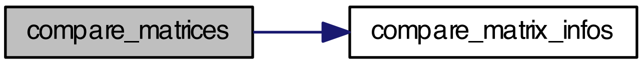
8.9.1.10 static int compare_matrices (char ∗ *name,* struct **fclib_matrix** ∗ *a,* struct **fclib_matrix** ∗ *b*) `[static]`

Definition at line 222 of file fctst_merit.c.

References compare_matrix_infos(), fclib_matrix::i, fclib_matrix::info, fclib_matrix::m, fclib_matrix::n, fclib_matrix-::nz, fclib_matrix::nzmax, fclib_matrix::p, and fclib_matrix::x.

Referenced by compare_global_problems(), and compare_local_problems().

Here is the call graph for this function:



8.9.1.11 static int compare_vectors (char ∗ *name,* int *n,* double ∗ *a,* double ∗ *b*) `[static]`

Definition at line 327 of file fctst_merit.c.

Referenced by compare_global_problems(), compare_local_problems(), and compare_solutions().

8.9.1.12 static int compare_infos (struct **fclib_info** ∗ *a,* struct **fclib_info** ∗ *b*) `[static]`

Definition at line 347 of file fctst_merit.c.

References fclib_info::description, fclib_info::math_info, and fclib_info::title.

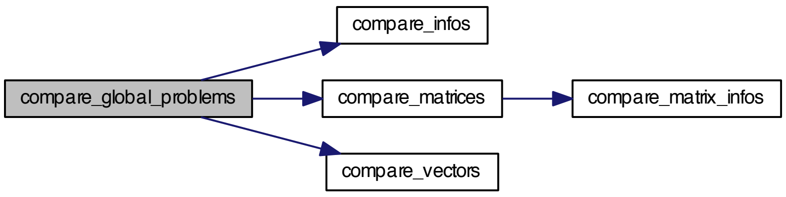
Referenced by compare_global_problems(), and compare_local_problems().

8.9.1.13 static int compare_global_problems (struct **fclib_global** ∗ *a,* struct **fclib_global** ∗ *b*) `[static]`

Definition at line 359 of file fctst_merit.c.

References fclib_global::b, compare_infos(), compare_matrices(), compare_vectors(), fclib_global::f, fclib_global-::G, fclib_global::H, fclib_global::info, fclib_matrix::m, fclib_global::M, fclib_global::mu, fclib_matrix::n, fclib_global-::spacedim, and fclib_global::w.

Here is the call graph for this function:

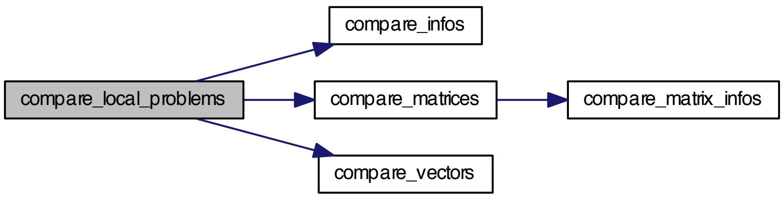


8.9.1.14 static int compare_local_problems (struct fclib_local ∗ *a*, struct fclib_local ∗ *b*) [static]

Definition at line 375 of file fctst_merit.c.

References compare_infos(), compare_matrices(), compare_vectors(), fclib_local::info, fclib_local::mu, fclib_matrix::n, fclib_local::q, fclib_local::R, fclib_local::s, fclib_local::spacedim, fclib_local::V, and fclib_local::W.

Here is the call graph for this function:

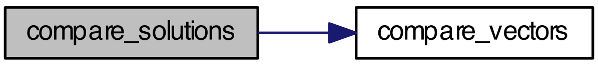


8.9.1.15 static int compare_solutions (struct fclib_solution ∗ *a*, struct fclib_solution ∗ *b*, int *nv*, int *nr*, int *nl*) [static]

Definition at line 390 of file fctst_merit.c.

References compare_vectors(), fclib_solution::l, fclib_solution::r, fclib_solution::u, and fclib_solution::v.

Here is the call graph for this function:

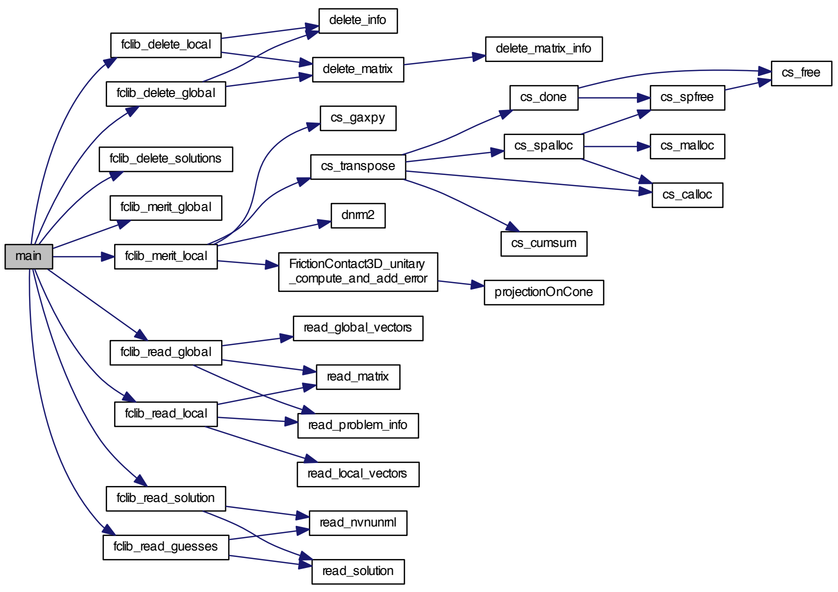


8.9.1.16 int main (int *argc*, char ∗∗ *argv*)

Definition at line 400 of file fctst_merit.c.

References fclib_delete_global(), fclib_delete_local(), fclib_delete_solutions(), fclib_merit_global(), fclib_merit_local(), fclib_read_global(), fclib_read_guesses(), fclib_read_local(), fclib_read_solution(), and MERIT_1.

Here is the call graph for this function:



8.10 mainpage.doxygen File Reference