HAL
open science

FCLIB: a collection of discrete 3D Frictional Contact problems

Vincent Acary, Maurice Brémond, Tomasz Koziara, Franck Pérignon

To cite this version:

Vincent Acary, Maurice Brémond, Tomasz Koziara, Franck Pérignon. FCLIB: a collection of discrete 3D Frictional Contact problems. [Technical Report] RT-0444, INRIA. 2014, pp.34. hal-00945820v2

HAL Id: hal-00945820

https://inria.hal.science/hal-00945820v2

Submitted on 14 Feb 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

FCLIB: a collection of discrete 3D Frictional Contact problems

Vincent Acary, Maurice Brémond, Tomasz Koziara, Franck Pérignon

TECHNICAL REPORT
N° 444
February 14, 2014
Project-Team Bipop

ISSN 0249-0803 ISRN INRIA/RT--444--FR+ENG

FCLIB: a collection of discrete 3D Frictional Contact problems

Vincent Acary*, Maurice Brémond†, Tomasz Koziara‡, Franck Pérignon§

Project-Team Bipop

Technical Report n° 444 — February 14, 2014 — 29 pages

Abstract: The goal of this work is to set up a collection of 3D Frictional Contact (3DFC) problems. The collection will provide a standard framework for testing available and new algorithms for solving discrete frictional contact problems. In this document, we describe the mathematical problems that we want to collect and the contents of the current collection.

Key-words: Coulomb's friction, unilateral contact, benchmarks

* INRIA Rhône–Alpes. LJK Laboratoire Jean Kuntzman. vincent.acary@inria.fr
† INRIA Rhône–Alpes. maurice.bremond@inria.fr
‡ School of Engineering and Computing Sciences.Durham University U.K. tomasz.koziara@durham.ac.uk
§ LJK Laboratoire Jean Kuntzman. franck.perignon@imag.fr

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

FCLIB: une collection de problèmes discrets tridimensionels de contact avec frottement

Résumé : Le but de ce travail est de mettre en place une collection de problèmes discrets de frottement de Coulomb avec contact unilatŕal. La collection fournira un environnement standard de test pour les algorithmes de résolution, existants et à venir. Dans ce document, on décrit les problèmes mathématiques que nous voulons collecter et le contenu de la collection courante.

Mots-clés : frottement de Coulomb, contact unilateral, benchmarks

Purpose of the document

The goal of this work is to set up a collection of 3D Frictional Contact (3DFC) problems. The collection will provide a standard framework for testing available and new algorithms for solving discrete frictional contact problems. In this document, we describe the mathematical problems that we want to collect and the contents of the current collection.

Notation

Let us denote by the integer n_c the number of contacts. The integer n is the number of degree of freedom of the system and $m = 3n_c$ the number of unknown variables at contacts.

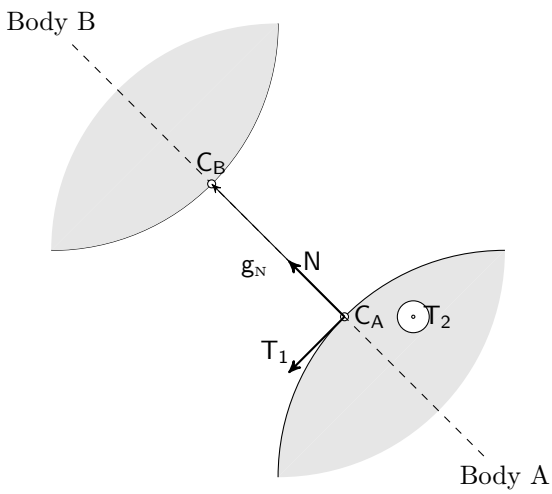


Figure 1: Local frame at contact

For each contact $\alpha \in \{1, \dots n_c\}$, the local velocity is denoted by $u^\alpha \in \mathbb{R}^3$. Its normal part is denoted by $u_{\mathrm{N}}^\alpha \in \mathbb{R}$ and its tangential part $u_{\mathrm{T}} \in \mathbb{R}^2$ (see Figure 1)

$$u^\alpha = \begin{bmatrix} u_{\mathrm{N}}^\alpha \\ u_{\mathrm{T}}^\alpha \end{bmatrix}. \tag{1}$$

The vector u collects all the local velocities at each contact

$$u = [[u^\alpha]^T, \alpha = 1 \dots n_c]^T, \tag{2}$$

respectively for the normal part u_{N}

$$u_{\mathrm{N}} = [u_{\mathrm{N}}^\alpha, \alpha = 1 \dots n_c]^T, \tag{3}$$

and its tangential a part as

$$u_{\mathrm{T}} = [[u_{\mathrm{T}}^\alpha]^T, \alpha = 1 \dots n_c]^T. \tag{4}$$

For a contact α and a friction coefficient μ, the modified local velocity, denoted by $\hat{u}^\alpha$, is defined by

$$\hat{u}^\alpha = u^\alpha + \begin{bmatrix} \mu \|u_{\mathrm{T}}^\alpha\| \\ 0 \\ 0 \end{bmatrix}. \tag{5}$$

The vector $\hat{u}$ collects all the modified local velocity at each contact

$$\hat{u} = [[\hat{u}^\alpha]^T, \alpha = 1 \dots n_c]^T. \tag{6}$$

For each contact α, the reaction vector $r^\alpha \in \mathbb{R}^3$ is also decomposed in its normal part $r^\alpha_{\mathrm{N}} \in \mathbb{R}$ and its tangential part $r_{\mathrm{T}} \in \mathbb{R}^2$ as

$$r^\alpha = \begin{bmatrix} r^\alpha_{\mathrm{N}} \\ r^\alpha_{\mathrm{T}} \end{bmatrix}. \tag{7}$$

The Coulomb friction cone for a contact α is defined by

$$K^\alpha_{\mu^\alpha} = \{r^\alpha, \|r^\alpha_{\mathrm{T}}\| \leqslant \mu^\alpha |r^\alpha_{\mathrm{N}}|\} \tag{8}$$

and the set $K^{\alpha,\star}_{\mu^\alpha}$ is its dual. The set K_μ is the cartesian product of Coulomb's friction cone at each contact,

$$K_\mu = \prod_{\alpha=1\dots n_c} K^\alpha_{\mu^\alpha} \tag{9}$$

For more details, we refer to[1].

1 Linear discrete problems with Coulomb's friction and unilateral contact

In this section, we formulate basic discrete frictional contact problems considering a finite number n of degrees of freedom together with a discrete linear dynamics and possibly some bilateral constraints. We assume that a finite set of n_c contact points and their associated local frames has been defined.

Definition 1 (Frictional contact problem (3DFC))**.** *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 3DFC problem, denoted by 3DFC(W, q, μ)*, consists in finding two vectors* $u \in \mathbb{R}^m$ *and* $r \in \mathbb{R}^m$ *such that*

$$\begin{cases} \hat{u} = Wr + q + \left[\begin{bmatrix} \mu^\alpha \|u_T^\alpha\| \\ 0 \\ 0 \end{bmatrix}^T, \alpha = 1 \dots n_c \right]^T \\ \\ K_\mu^\star \ni \hat{u} \perp r \in K_\mu \end{cases} \tag{10}$$

□

Definition 2 (Global 3DFrictional contact problem (G3DFC))**.** *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 vector* $w \in \mathbb{R}^m$*,*
- *a vector of coefficients of friction* $\mu \in \mathbb{R}^{n_c}$

the Global 3DFC problem, denoted by G3DFC(M, H, f, w, μ)*, consists in finding three vectors* $v \in \mathbb{R}^n$*,* $u \in \mathbb{R}^m$ *and* $r \in \mathbb{R}^m$ *such that*

$$\begin{cases} Mv = Hr + f \\ \\ \hat{u} = H^T v + w + \left[\begin{bmatrix} \mu^\alpha \|u_T^\alpha\| \\ 0 \\ 0 \end{bmatrix}^T, \alpha = 1 \dots n_c \right]^T \\ \\ K_\mu^\star \ni \hat{u} \perp r \in K_\mu \end{cases} \tag{11}$$

□

Definition 3 (Global Mixed Frictional contact problem (GM3DFC))**.** *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* $w \in \mathbb{R}^{\mathsf{m}}$,
- *a vector* $b \in \mathbb{R}^{\mathsf{p}}$,
- *a vector of coefficients of friction* $\mu \in \mathbb{R}^{\mathsf{n_c}}$

the Global Mixed 3DFC problem, denoted by GM3DFC(M, H, G, w, b, μ)*, consists in finding four vectors* $v \in \mathbb{R}^n$, $u \in \mathbb{R}^{\mathsf{m}}$, $r \in \mathbb{R}^{\mathsf{m}}$ *and* $\lambda \in \mathbb{R}^{\mathsf{p}}$ *such that*

$$\begin{cases} Mv = Hr + G\lambda + f \\ G^T v + b = 0 \\ \hat{u} = H^T v + w + \left[\begin{bmatrix} \mu \|u_T^\alpha\| \\ 0 \\ 0 \end{bmatrix}^T, \alpha = 1 \dots n_c \right]^T \\ K_\mu^\star \ni \hat{u} \perp r \in K_\mu \end{cases} \tag{12}$$

□

Definition 4 (Mixed 3DFrictional contact problem (M3DFC))**.** *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* $q \in \mathbb{R}^{\mathsf{m}}$,
- *a vector* $s \in \mathbb{R}^{\mathsf{p}}$,
- *a vector of coefficients of friction* $\mu \in \mathbb{R}^{\mathsf{n_c}}$

the Mixed 3DFC problem, denoted by M3DFC(R, V, W, q, s, μ) *, consists in finding three vectors* $u \in \mathbb{R}^{\mathsf{m}}$, $r \in \mathbb{R}^{\mathsf{m}}$ *and* $\lambda \in \mathbb{R}^{\mathsf{p}}$ *such that*

$$\begin{cases} V^T r + R\lambda + s = 0 \\ \hat{u} = Wr + V\lambda + q + \left[\begin{bmatrix} \mu^\alpha \|u_T^\alpha\| \\ 0 \\ 0 \end{bmatrix}^T, \alpha = 1 \dots n_c \right]^T \\ K_\mu^\star \ni \hat{u} \perp r \in K_\mu \end{cases} \tag{13}$$

□

Remark Note that the previous problems may be an instance of quasi-static problems: the matrix M plays the role of the stiffness matrix and the vector u is a position or a displacement. All the problems can also be an problem in terms of acceleration and forces that we find in event–driven schemes.

2 Details on the implementation

File format The proposed file format for storing and managing data is the HDF5 data format `http://www.hdfgroup.org/HDF5`
The data name should be defined as close as possible to the definition of this document.

Matrix storage Several matrix storages are considered :

1. dense format
2. sparse format : row compressed format, column compressed and triplet (see the description in [2]).

The storage of dense matrices is in column major mode (FORTRAN mode). For the sparse matrices, we use the sparse toolkit developed by T. Davis [2].

C implementation A C implementation is proposed for reading and writing each of 3DFC problems into HDF5 files. Some details of the C implementation are given in Appendix. More details can be found at `http://fclib.gforge.inria.fr`.

3 Additional description of the problems

The following additional information should be added in a reference document and in the HDF5 file.

- `TITLE` : a title for the problem
- `DESCRIPTION` : the field of application. Short description on how the problem is generated.
- `MATRIX_INFO` : the sparsity and the conditioning of the matrices.
- `MATH_INFO` : existence, uniqueness of solutions.
- ...

The following data can be optionally added in the HDF5 file

- `SOLUTION` : a reference solution
- `INITIAL_GUESS` : an initial guess
- ...

4 List of problems in FCLIB. version 0.2

- Hanging chain with initial velocity at the tip (see Section 4.1).
- 100 capsules dropped into a box (see Section 4.2).
- 50 boxes stacked under gravity (see Section 4.3).
- A tower of Kaplas (see Section 4.4).

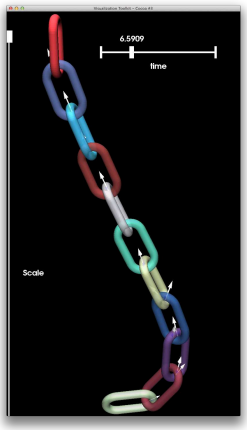
For each example, several configurations are available. Each problem is described in a file (HDF5 format) and mainly characterized by the number of degrees of freedom, the number of contacts and the formulation for the contact problem.

4.1 Hanging chain with initial velocity at the tip.

Authors	V. Acary, M. Brémond. (INRIA Rhône-Alpes)
Date	10/02/2014
Software	Siconos

This set of 1514 problems has been generated by Siconos with the help of Bullet contact detection library. It simulates an hanging chain with initial velocity at the tip. The chain is composed of 11 elements with the same geometry given by a unique mesh. The mass of each component is 1kg for a length of 13.7m and a thickness of 7.6m . The initial velocity at the tip is 50mm/s.

The script that generates this example can be obtained from the Siconos development team. On Figure 2, the distribution of the number of contacts, the number of d.o.f and the ratio number of contacts unknowns/number of d.o.f are illustrated.



coefficient of friction	0.3
number of problems	1514
number of degrees of freedom	[48:60]
number of contacts	[8:28]
required accuracy	10^{-8}

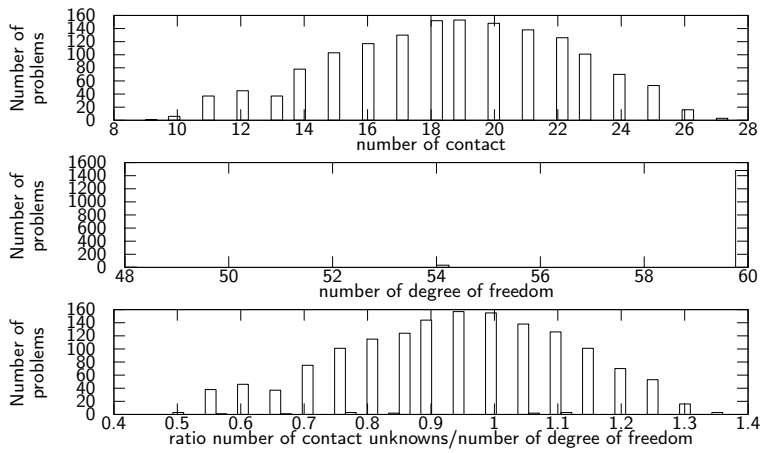


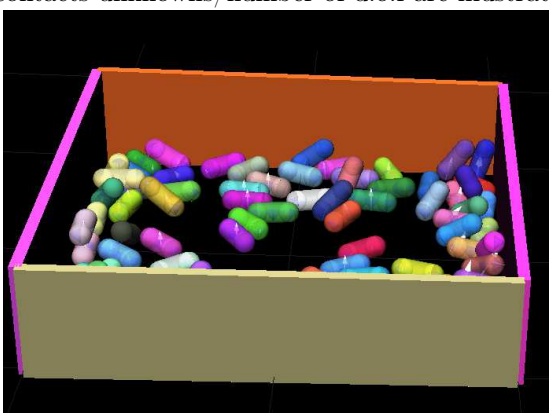
Figure 2: distribution of the number of contacts, the number of d.o.f and their ratio

4.2 100 capsules dropped into a box.

Authors	V. Acary, M. Brémond. (INRIA Rhône-Alpes)
Date	10/02/2014
Software	Siconos

This set of 1514 problems has been generated by Siconos with the help of Bullet contact detection library. It simulates 100 capsules dropped into a box. The Mass of each capsule is 1kg and length is 5m. The radius is 1m.

The script that generates this example can be obtained from the Siconos development team. On Figure 3, the distribution of the number of contacts, the number of d.o.f and the ratio number of contacts unknowns/number of d.o.f are illustrated.



coefficient of friction	0.7
number of problems	1705
number of degrees of freedom	[6:600]
number of contacts	[0:300]
required accuracy	10^{-8}

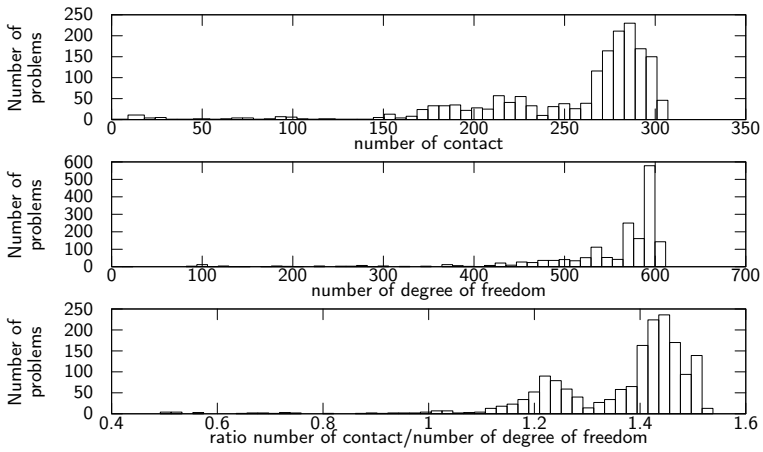
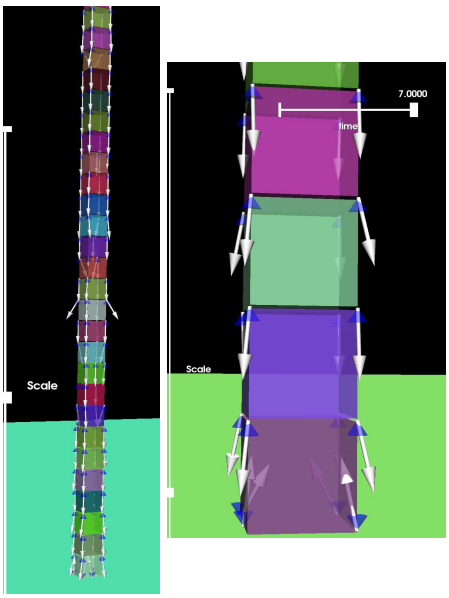


Figure 3: distribution of the number of contacts, the number of d.o.f and their ratio

4.3 50 boxes stacked under gravity

Authors	V. Acary, M. Brémond. (INRIA Rhône-Alpes)
Date	10/02/2014
Software	Siconos

This set of 1514 problems has been generated by Siconos with the help of Bullet contact detection library. It simulates 50 boxes stacked under gravity. The mass of the box is 1kg and the size is 2×2m. The script that generates this example can be obtained from the Siconos development team. On Figure 5, the distribution of the number of contacts, the number of d.o.f and the ratio number of contacts unknowns/number of d.o.f are illustrated.



coefficient of friction	0.7
number of problems	1159
number of degrees of freedom	[6:300]
number of contacts	[0:200]
required accuracy	10^{-8}

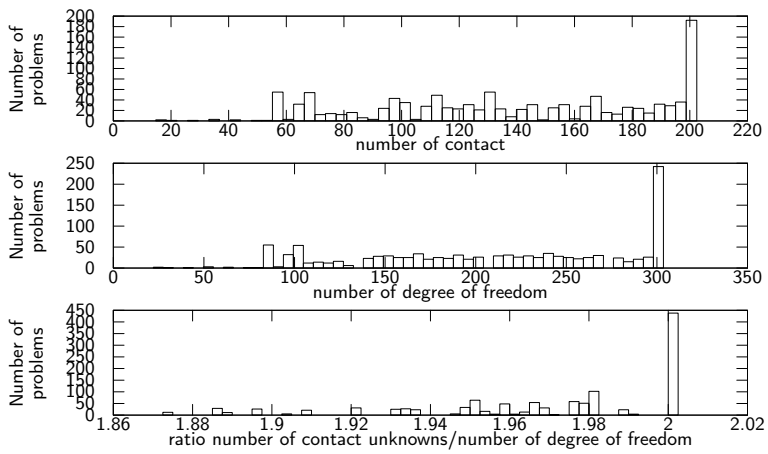
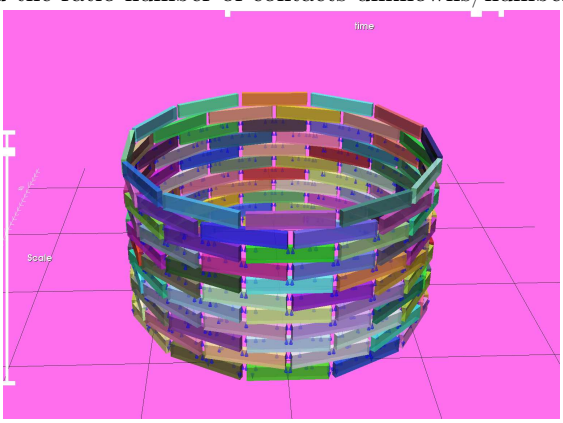


Figure 4: distribution of the number of contacts, the number of d.o.f and their ratio

4.4 A tower of Kaplas

Authors	V. Acary, M. Brémond. (INRIA Rhône-Alpes)
Date	10/02/2014
Software	Siconos

This set of 201 problems has been generated by Siconos with the help of Bullet contact detection library. It simulates a tower of 144 kaplas of dimension $11.4 \times 2.348 \times 0.78267$cm. The mass of each is 1kg. The script that generates this example can be obtained from the Siconos development team. On Figure 5, the distribution of the number of contacts, the number of d.o.f and the ratio number of contacts unknowns/number of d.o.f are illustrated.



coefficient of friction	0.3
number of problems	201
number of degrees of freedom	[72:864]
number of contacts	[0:950]
required accuracy	10^{-8}

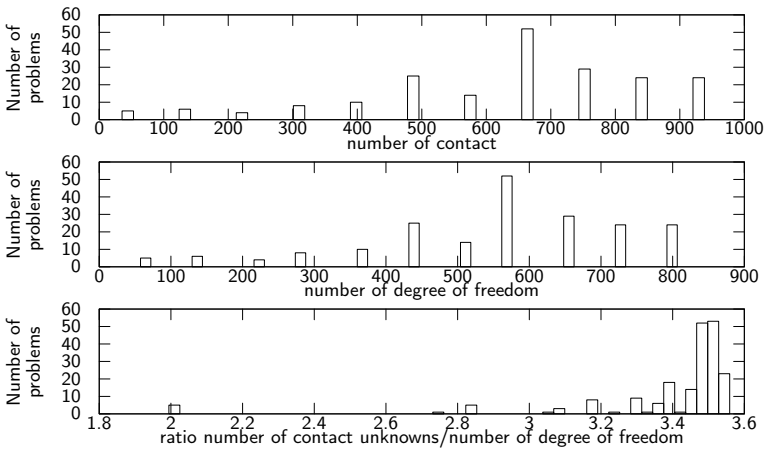


Figure 5: distribution of the number of contacts, the number of d.o.f and their ratio

References

[1] V. Acary and B. Brogliato. *Numerical methods for nonsmooth dynamical systems. Applications in mechanics and electronics.* Lecture Notes in Applied and Computational Mechanics 35. Berlin: Springer. xxi, 525 p. , 2008.

[2] Timothy A. Davis. *Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms 2).* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2006.

A Class Index

A.1 Class List

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

cs_dmperm_results **14**

cs_numeric **15**

cs_sparse **16**

cs_symbolic **17**

fclib_global
The global frictional contact problem defined by **18**

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 **21**

fclib_local
The local frictional contact problem defined by **22**

fclib_matrix
Matrix in compressed row/column or triplet form **25**

fclib_matrix_info
This structure allows the user to enter a description for a given matrix (comment, conditionning, determinant, rank.) if they are known **27**

fclib_solution
A solution or a guess for the frictional contact problem **28**

B Class Documentation

B.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]

B.1.1 Detailed Description

Definition at line 67 of file csparse.h.

B.1.2 Member Data Documentation

int∗ cs_dmperm_results::P Definition at line 69 of file csparse.h.
Referenced by cs_dalloc(), cs_dfree(), cs_dmperm(), and cs_scc().

int∗ cs_dmperm_results::Q Definition at line 70 of file csparse.h.
Referenced by cs_dalloc(), cs_dfree(), and cs_dmperm().

int∗ cs_dmperm_results::R Definition at line 71 of file csparse.h.
Referenced by cs_dalloc(), cs_dfree(), cs_dmperm(), and cs_scc().

int∗ cs_dmperm_results::S Definition at line 72 of file csparse.h.
Referenced by cs_dalloc(), cs_dfree(), and cs_dmperm().

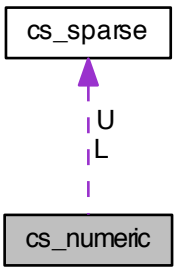
int cs_dmperm_results::nb Definition at line 73 of file csparse.h.
Referenced by cs_dmperm(), and cs_scc().

int cs_dmperm_results::rr[5] Definition at line 74 of file csparse.h.
Referenced by cs_dmperm().

int cs_dmperm_results::cc[5] Definition at line 75 of file csparse.h.
Referenced by cs_dmperm().

B.2 cs_numeric Struct Reference

`#include <csparse.h>`
Collaboration diagram for cs_numeric:



Public Attributes

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

B.2.1 Detailed Description

Definition at line 59 of file csparse.h.

B.2.2 Member Data Documentation

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().

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().

int∗ cs_numeric::Pinv Definition at line 63 of file csparse.h.
Referenced by cs_lu(), cs_lusol(), and cs_nfree().

double∗ cs_numeric::B Definition at line 64 of file csparse.h.
Referenced by cs_nfree(), cs_qr(), and cs_qrsol().

B.3 cs_sparse Struct Reference

`#include <csparse.h>`

Public Attributes

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

B.3.1 Detailed Description

Definition at line 14 of file csparse.h.

B.3.2 Member Data Documentation

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().

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().

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().

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().

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().

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().

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().

B.4 cs_symbolic Struct Reference

`#include <csparse.h>`

Public Attributes

- int ∗ Pinv
- int ∗ Q
- int ∗ parent
- int ∗ cp

- int m2
- int lnz
- int unz

B.4.1 Detailed Description

Definition at line 48 of file csparse.h.

B.4.2 Member Data Documentation

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().

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().

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().

int∗ cs_symbolic::cp Definition at line 53 of file csparse.h.
Referenced by cs_chol(), cs_schol(), cs_sfree(), and cs_sqr().

int cs_symbolic::m2 Definition at line 54 of file csparse.h.
Referenced by cs_qr(), cs_qrsol(), and cs_sqr().

int cs_symbolic::lnz Definition at line 55 of file csparse.h.
Referenced by cs_lu(), cs_qr(), cs_schol(), and cs_sqr().

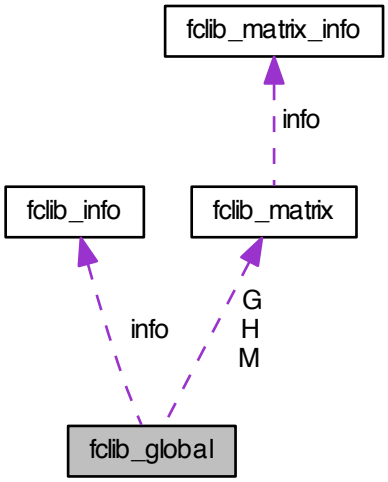
int cs_symbolic::unz Definition at line 56 of file csparse.h.
Referenced by cs_lu(), cs_qr(), cs_schol(), and cs_sqr().

B.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

B.5.1 Detailed Description

The global frictional contact 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 GM3DFC(M, H, G, w, b, μ) such that

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

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

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

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

Definition at line 174 of file fclib.h.

B.5.2 Member Data Documentation

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().

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().

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().

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().

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().

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().

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().

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().

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().

B.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, ...)

B.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.

B.6.2 Member Data Documentation

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().

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().

char∗ fclib_info::math_info known properties of the problem (existence, uniqueness, ...)

Definition at line 98 of file fclib.h.

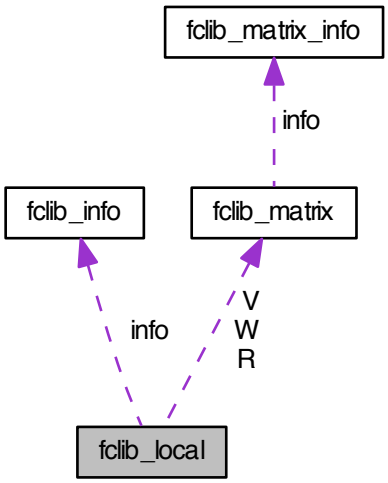
Referenced by compare_infos(), delete_info(), problem_info(), read_problem_info(), and write_problem_info().

B.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

B.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 M3DFC(R, V, W, q, s, μ) such that

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

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

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

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

Definition at line 228 of file fclib.h.

B.7.2 Member Data Documentation

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().

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().

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().

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().

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().

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().

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().

struct fclib_info∗ fclib_local::info info on the problem

Definition at line 245 of file fclib.h.

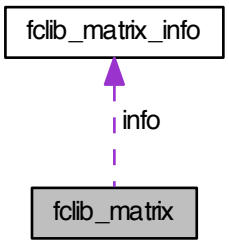
Referenced by compare_local_problems(), fclib_delete_local(), fclib_read_local(), fclib_write_local(), and random_local_problem().

B.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

B.8.1 Detailed Description

matrix in compressed row/column or triplet form
Definition at line 119 of file fclib.h.

B.8.2 Member Data Documentation

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().

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().

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().

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().

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().

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().

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().

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().

B.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

B.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.

B.9.2 Member Data Documentation

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().

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().

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().

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().

B.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

B.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.

B.10.2 Member Data Documentation

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().

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().

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().

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().

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-0803