ShapeOpt

Generated by Doxygen 1.8.6

Thu Jun 11 2015 00:56:18

Contents

Chapter 1

Introduzione

1.1 Descrizione

Questo programma permette di risolvere problemi di ottimizzazione di forma facendo ricorso a diverse tecniche.

Tutti i sorgenti e gli header sono scritti in linguaggio C++11.

Il software è stato progettato per l'utilizzo su un sistema operativo Unix-like.

1.2 Dipendenze

Il software richiede che sul sistema siano installate le seguenti dipendenze (tra parentesi la versione minima richiesta):

- `CMake` (versione 2.8), un tool di configurazione multi-piattaforma;
- `Make` (versione 3.8.1), un tool utilizzato per la compilazione dei sorgenti;
- `GCC` (versione 4.8), la suite di compilatori GNU Compiler Collection;
- `Eigen` (version 3.2), per gestire matrici, vettori e algebra lineare;
- `libMesh` (versione 0.9.3), un framework per risolvere problemi alle derivate parziali tramite il metodo degli elementi finiti.

Viene inoltre utilizzata la seguente libreria, fornita nella cartella *include/*:

- `GetPot` (versione 1.0), per effettuare il parsing da riga di comando e di file di configurazione.

1.3 Compilazione

Per generare l'eseguibile, aprire il file *CMakeLists.txt* (presente nella cartella principale) e, se necessario, modificarlo secondo le proprie esigenze.

Creare dunque una cartella di compilazione e aprirla:

```
$ mkdir build
$ cd build
```

Adesso il sistema è pronto per la configurazione:

```
$ cmake ..
```

Note

oppure, per generare anche i simboli per il debug:

```
$ cmake -DCMAKE_BUILD_TYPE=Debug ..
```

Inoltre, l'opzione:

```
-DCMAKE_INSTALL_PREFIX=my_dir
```

consentirà di specificare la directory d'installazione, che di default si trova in */usr/local*.

Infine:

```
$ make
```

genererà l'eseguibile *test* e la libreria condivisa *shapeopt* nelle cartelle *bin/* e *lib/* (o in quelle specificate nel file *CMakeLists.txt*) rispettivamente.

1.4 Installazione

Se si desidera installare:

- l'eseguibile, in *${CMAKE_INSTALL_PREFIX}/bin/*;
- la libreria condivisa, in *${CMAKE_INSTALL_PREFIX}/lib/*;
- gli header, in *${CMAKE_INSTALL_PREFIX}/include/shapeopt/*;

occorre eseguire il comando:

```
# make install
```

È possibile disinstallare tramite:

```
# make uninstall
```

Se Doxygen (versione 3.8.6) e GraphViz sono installati sul sistema, il seguente comando genererà questa documentazione nella cartella *doc/* (o in quella specificata in *CMakeLists.txt*):

```
$ make doc
```

1.5 Impostazione dei parametri

Note

La cartella di configurazione di default è *config/*.

Prima di poter eseguire il programma, occorre settare il file di configurazione (default: *config.pot*). Al suo interno sarà possibile modificare una lista di parametri, ciascuno dei quali è commentato per spiegarne il significato.

Ad esempio, si potrà specificare: la mesh da utilizzare (in formato Gmsh), la directory in cui salvare i file delle soluzioni, il problema da risolvere e la tecnica di ottimizzazione di forma da utilizzare.

È possibile creare diversi file di configurazione, ciascuno con i propri valori per i parametri: di volta in volta il file da usare può essere specificato da riga di comando prima di eseguire il programma.

1.6 Esegui!

Per eseguire utilizzando il file di configurazione predefinito (*config.pot*), spostarsi nella cartella in cui si trova l'eseguibile e digitare:

```
$ ./test
```

Di default, il file di configurazione viene cercato nella cartella *../config*.

Per specificare un diverso file di configurazione precedentemente salvato in questa directory:

```
$ ./test -f configuration_filename
```

oppure:

```
$ ./test --file configuration_filename
```

La variabile *configuration_filename* **non** deve contenere il path al file.

Warning

È anche possibile specificare una diversa directory in cui cercare il file di configurazione (attraverso un path relativo o assoluto rispetto alla cartella attuale), utilizzando il comando:

```
$ ./test -d configuration_directory
```

oppure:

```
$ ./test --directory configuration_directory
```

Una volta terminato il programma, i risultati verranno salvati nella cartella di output (relativa alla directory corrente) impostata nel file di configurazione.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

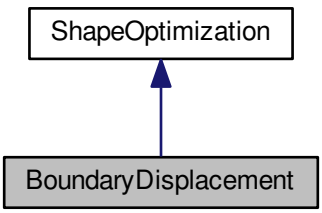
Class Documentation

5.1 BoundaryDisplacement Class Reference

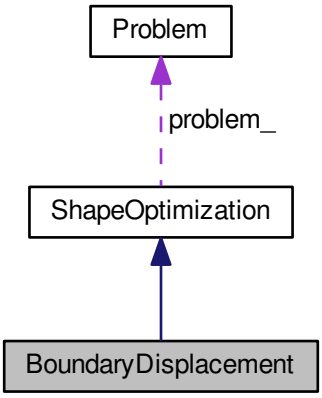
Classe che eredita da ShapeOptimization. Utilizza la tecnica del boundary local displacement per eseguire l'ottimizzazione di forma.

`#include <BoundaryDisplacement.h>`

Inheritance diagram for BoundaryDisplacement:



Collaboration diagram for BoundaryDisplacement:



Public Member Functions

- BoundaryDisplacement (const Problem &, const std::string &, const Real &, const Index &, const Real &, const bool &, const Real &=1.0e-4)

  Costruttore.
- virtual void computePerturbation (EquationSystems &, EquationSystems &)

  Calcola la deformazione della mesh.
- virtual void applyPerturbation (const EquationSystems &)

  Applica la deformazione alla mesh.

Additional Inherited Members

5.1.1 Detailed Description

Classe che eredita da ShapeOptimization. Utilizza la tecnica del boundary local displacement per eseguire l'ottimizzazione di forma.

Definition at line 28 of file BoundaryDisplacement.h.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 BoundaryDisplacement (const Problem & *problem,* const std::string & *directory,* const Real & *step,* const Index & *maxIterationsNo,* const Real & *tolerance,* const bool & *volume_constraint,* const Real & *armijoSlope* = `1.0e-4`)

Costruttore.

Parameters

in	*problem*	: Problema sul quale si vuole applicare la Shape Optimization
in	*directory*	: Directory in cui salvare i file di output
in	*step*	: Passo iniziale per il metodo di discesa del gradiente
in	*maxIterationsNo*	: Numero massimo di iterazioni
in	*tolerance*	: Tolleranza per il test d'arresto dell'incremento relativo
in	*volume_-constraint*	: Specifica se applicare o meno il vincolo di volume
in	*armijoSlope*	: Coefficiente di rilassamento per la regola di Armijo.

Definition at line 3 of file BoundaryDisplacement.cc.

5.1.3 Member Function Documentation

5.1.3.1 void computePerturbation (EquationSystems & *perturbation,* EquationSystems & *stateAdj*) [virtual]

Calcola la deformazione della mesh.

Parameters

out	*perturbation*	: Sistema d'equazioni contenente gli spostamenti da applicare alla mesh
in	*stateAdj*	: Sistema d'equazioni contenente stato e aggiunto

Implements ShapeOptimization.

Definition at line 6 of file BoundaryDisplacement.cc.

5.1.3.2 void applyPerturbation (const EquationSystems & *perturbation*) [virtual]

Applica la deformazione alla mesh.

Parameters

in	*perturbation*	: Sistema d'equazioni contenente gli spostamenti da applicare alla mesh

Implements ShapeOptimization.

Definition at line 11 of file BoundaryDisplacement.cc.

The documentation for this class was generated from the following files:

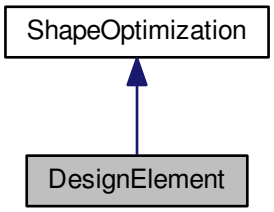
- src/BoundaryDisplacement.h
- src/BoundaryDisplacement.cc

5.2 DesignElement Class Reference

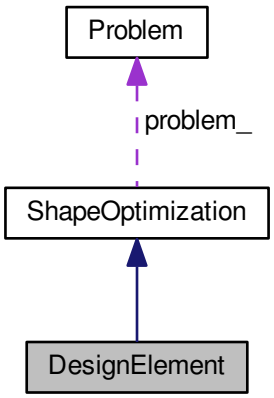
Classe che eredita dalla classe ShapeOptimization, utilizza il metodo del Design Element descrivendo il bordo superiore e quello inferiore con polinomi di grado arbitrario.

`#include <DesignElement.h>`

Inheritance diagram for DesignElement:



Collaboration diagram for DesignElement:



Public Member Functions

- DesignElement (const Problem &, const std::string &, const Real &, const Index &, const Real &, const bool &, const std::pair< Point, Point > &, const Index &, const Real &=1.0e-4)

  Costruttore.
- virtual void computePerturbation (EquationSystems &, EquationSystems &)

  Metodo astratto per calcolare la deformazione della mesh.
- virtual void applyPerturbation (const EquationSystems &)

  Metodo astratto per applicare la deformazione alla mesh.
- Point psi (const Point &) const

  mappa la scatola nel quadrato unitario
- Point psiInv (const Point &) const

  mappa il quadrato unitario nel rettangolo di partenza
- Point deform (const Point &) const

  applica la deformazione al punto

Protected Attributes

- Mesh reference_mesh_

  mesh di riferimento
- VectorXp reference_nodes_

  vettore contenente i nodi del bordo nella mesh di riferimento
- std::pair< Point, Point > boundingBox_

  coppia contenente i punti nord est e sud ovest che definiscono la scatola
- VectorXr mu_

  vettore contenente i coefficienti dei polinomi f_{up}, f_{down}
- VectorXr gradJ_

  *vettore contenente il gradiente ridotto rispetto a mu_*
- MatrixXr P_

  matrice di proiezione per fissare gli estremi
- bool firstTime_

  booleano: vero se è la prima volta che calcola la perturbazione dell'identità

5.2.1 Detailed Description

Classe che eredita dalla classe ShapeOptimization, utilizza il metodo del Design Element descrivendo il bordo superiore e quello inferiore con polinomi di grado arbitrario.

Definition at line 28 of file DesignElement.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 DesignElement (const Problem & *problem,* const std::string & *directory,* const Real & *step,* const Index & *maxIterationsNo,* const Real & *tolerance,* const bool & *volume_constraint,* const std::pair< Point, Point > & *boundingBox,* const Index & *order,* const Real & *armijoSlope* = `1.0e-4`)

Costruttore.

Parameters

in	*problem*	: Problema sul quale si vuole applicare la Shape Optimization
in	*directory*	: Directory in cui salvare i file di output
in	*step*	: Passo iniziale per il metodo di discesa del gradiente
in	*maxIterationsNo*	: Numero massimo di iterazioni
in	*tolerance*	: Tolleranza per il test d'arresto dell'incremento relativo
in	*volume_constraint*	: Specifica se applicare o meno il vincolo di volume
in	*boundingBox*	: Punti a nord est e a sud ovest indicanti il range della bounding box
in	*order*	: Grado del polinomio secondo cui vengono deformati i bordi superiore e inferiore del dominio
in	*armijoSlope*	: Coefficiente di rilassamento per la regola di Armijo.

Definition at line 3 of file DesignElement.cc.

5.2.3 Member Function Documentation

5.2.3.1 void computePerturbation (EquationSystems & *perturbation,* EquationSystems & *stateAdj*) `[virtual]`

Metodo astratto per calcolare la deformazione della mesh.

Parameters

out	*perturbation*	: Sistema d'equazioni contenente gli spostamenti da applicare alla mesh
in	*stateAdj*	: Sistema d'equazioni contenente stato e aggiunto

Implements ShapeOptimization.

Definition at line 35 of file DesignElement.cc.

5.2.3.2 void applyPerturbation (const EquationSystems & *perturbation*) [virtual]

Metodo astratto per applicare la deformazione alla mesh.

Parameters

in	*perturbation*	: Sistema d'equazioni contenente gli spostamenti da applicare alla mesh

Implements ShapeOptimization.

Definition at line 153 of file DesignElement.cc.

5.2.3.3 Point psi (const Point & *point*) const

mappa la scatola nel quadrato unitario

Parameters

in	*point*	: punto nella scatola da trasformare

Returns

le cordinate del punto nel quadrato unitario

Definition at line 213 of file DesignElement.cc.

5.2.3.4 Point psiInv (const Point & *ref_point*) const

mappa il quadrato unitario nel rettangolo di partenza

Parameters

in	*ref_point*	: punto nel quadrato di riferimento da trasformare

Returns

le cordinate del punto nel rettangolo di partenza

Definition at line 228 of file DesignElement.cc.

5.2.3.5 Point deform (const Point & *point*) const

applica la deformazione al punto

Parameters

in	*point*	: punto in cui calcolare la deformazione

Returns

punto deformato

Definition at line 240 of file DesignElement.cc.

The documentation for this class was generated from the following files:

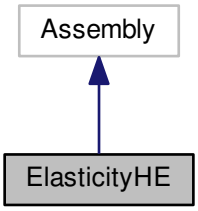
- src/DesignElement.h
- src/DesignElement.cc

5.3 ElasticityHE Class Reference

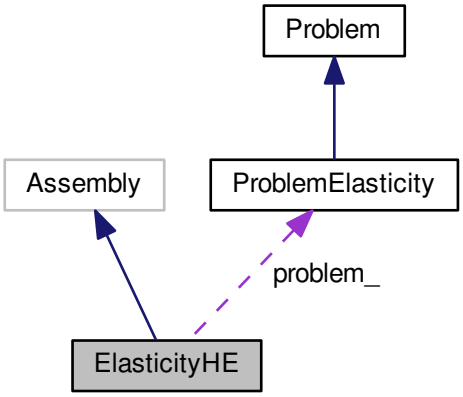
Classe contenente i metodi necessari per calcolare l'estensione armonica nel problema dell'elasticità

`#include <ProblemElasticity.h>`

Inheritance diagram for ElasticityHE:



Collaboration diagram for ElasticityHE:



Public Member Functions

- ElasticityHE (EquationSystems &, EquationSystems &, const Real &, const ProblemElasticity &)

  Costruttore.
- void assemble ()

  Assembla le matrici e i vettori per calcolare l'estensione armonica nel problema dell'elasticità

Private Attributes

- EquationSystems & perturbation_

  Sistemi d'equazioni per gestire lo spostamento della mesh.
- EquationSystems & stateAdj_

  Sistemi d'equazioni contenente lo stato e l'aggiunto.
- Real lagrange_

  Moltiplicatore di lagrange.
- const ProblemElasticity & problem_

  Riferimento costante al problema dell'elasticità

5.3.1 Detailed Description

Classe contenente i metodi necessari per calcolare l'estensione armonica nel problema dell'elasticità

Definition at line 97 of file ProblemElasticity.h.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 ElasticityHE (EquationSystems & *perturbation*, EquationSystems & *stateAdj*, const Real & *lagrange*, const ProblemElasticity & *problem*)

Costruttore.

Parameters

in	*perturbation*	: Riferimento ai sistemi d'equazioni per lo spostamento della mesh
in	*stateAdj*	: Riferimento ai sistemi d'equazioni contenente lo stato e l'aggiunto
in	*lagrange*	: Moltiplicatore di lagrange
in	*problem*	: Riferimento costante al problema dell'elasticità

Definition at line 261 of file ProblemElasticity.cc.

The documentation for this class was generated from the following files:

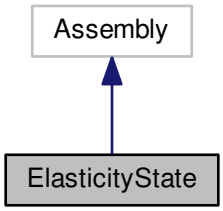
- src/ProblemElasticity.h
- src/ProblemElasticity.cc

5.4 ElasticityState Class Reference

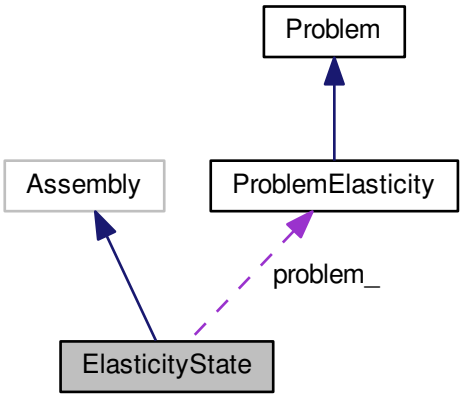
Classe contenente i metodi necessari per calcolare lo stato (il sistema è autoaggiunto) nel problema dell'elasticità

`#include <ProblemElasticity.h>`

Inheritance diagram for ElasticityState:



Collaboration diagram for ElasticityState:



Public Member Functions

- ElasticityState (EquationSystems &, const ProblemElasticity &)

  Costruttore.
- void assemble ()

  Assembla le matrici e i vettori per calcolare lo stato nel problema dell'elasticità
- Real evaluateElasticityTensor (Index i, Index j, Index k, Index l) const

  Calcola il valore del tensore elasticità negli indici desiderati.

Private Attributes

- EquationSystems & stateAdj_

  Sistemi d'equazioni contenente lo stato e l'aggiunto.
- const ProblemElasticity & problem_

  Riferimento costante al problema dell'elasticità

5.4.1 Detailed Description

Classe contenente i metodi necessari per calcolare lo stato (il sistema è autoaggiunto) nel problema dell'elasticità

Definition at line 125 of file ProblemElasticity.h.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 ElasticityState (EquationSystems & *stateAdj,* const ProblemElasticity & *problem*)

Costruttore.

Parameters

in	*stateAdj*	: Riferimento ai sistemi d'equazioni contenente lo stato e l'aggiunto
in	*problem*	: Riferimento costante al problema dell'elasticità

Definition at line 380 of file ProblemElasticity.cc.

5.4.3 Member Function Documentation

5.4.3.1 Real evaluateElasticityTensor (Index *i,* Index *j,* Index *k,* Index *l*) const

Calcola il valore del tensore elasticità negli indici desiderati.

Parameters

in	*i*	: Primo indice del tensore
in	*j*	: Secondo indice del tensore
in	*k*	: Terzo indice del tensore
in	*l*	: Quarto indice del tensore

Returns

$$D_{i,j,k,\ell} = \lambda \delta_{ij} \delta_{k\ell} + \mu (\delta_{ik} \delta_{j\ell} + \delta_{i\ell} \delta_{jk})$$

Definition at line 563 of file ProblemElasticity.cc.

The documentation for this class was generated from the following files:

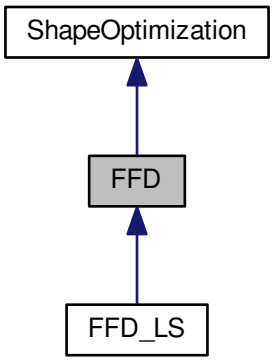
- src/ProblemElasticity.h
- src/ProblemElasticity.cc

5.5 FFD Class Reference

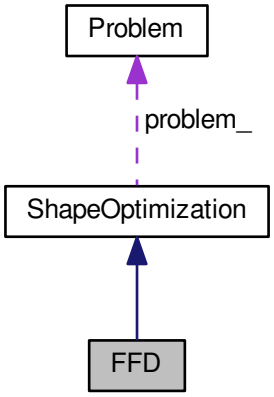
Classe che eredita dalla classe ShapeOptimization, utilizza il metodo della Free Form Deformation utilizzando come funzioni di base le B-Spline.

`#include <FFD.h>`

Inheritance diagram for FFD:



Collaboration diagram for FFD:



Public Member Functions

- FFD (const Problem &, const std::string &, const Real &, const Index &, const Real &, const bool &, const std::pair< Point, Point > &, const std::pair< Index, Index > &, const Real &=1.0e-4)

  Costruttore.
- virtual void computePerturbation (EquationSystems &, EquationSystems &)

  Calcola la deformazione della mesh.
- virtual void applyPerturbation (const EquationSystems &)

  Applica la deformazione alla mesh.
- virtual Real basisFunction (const Point &, const Index &, const Index &) const

  calcola la funzione di base k, l per il punto x

- Point psi (const Point &) const

  mappa la scatola nel quadrato unitario
- Point psiInv (const Point &) const

  mappa il quadrato unitario nel rettangolo di partenza
- Point deform (const Point &) const

  applica la deformazione al punto

Protected Attributes

- Mesh reference_mesh_

  mesh di riferimento
- VectorXp reference_nodes_

  vettore contenente i nodi del bordo nella mesh di riferimento
- std::pair< Point, Point > boundingBox_

  coppia contenente i punti nord est e sud ovest che definiscono la scatola
- std::pair< Index, Index > sub_

  coppia di numeri indicanti il numero di suddivisioni in orizzontale e in verticale
- MatrixXp CP_grid_

  matrice contenente i control point
- MatrixXp mu_

  matrice contenente gli spostamenti desiderati per i control point
- MatrixXp gradJ_

  matrice contenente il gradiente in funzione dei control point
- bool firstTime_

  booleano: vero se è la prima volta che calcola la perturbazione dell'identità

5.5.1 Detailed Description

Classe che eredita dalla classe ShapeOptimization, utilizza il metodo della Free Form Deformation utilizzando come funzioni di base le B-Spline.

Definition at line 28 of file FFD.h.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 FFD (const Problem & *problem*, const std::string & *directory*, const Real & *step*, const Index & *maxIterationsNo*, const Real & *tolerance*, const bool & *volume_constraint*, const std::pair< Point, Point > & *boundingBox*, const std::pair< Index, Index > & *sub*, const Real & *armijoSlope* = `1.0e-4`)

Costruttore.

Parameters

in	*problem*	: Problema sul quale si vuole applicare la Shape Optimization
in	*directory*	: Directory in cui salvare i file di output
in	*step*	: Passo iniziale per il metodo di discesa del gradiente
in	*maxIterationsNo*	: Numero massimo di iterazioni
in	*tolerance*	: Tolleranza per il test d'arresto dell'incremento relativo

in	*volume_-constraint*	: Specifica se applicare o meno il vincolo di volume
in	*boundingBox*	: Punti a nord est e a sud ovest indicanti il range della bounding box
in	*sub*	: Coppia contenente il numero di intervalli in cui suddividere la base e l'altezza della bounding box
in	*armijoSlope*	: Coefficiente di rilassamento per la regola di Armijo.

Definition at line 3 of file FFD.cc.

5.5.3 Member Function Documentation

5.5.3.1 void computePerturbation (EquationSystems & *perturbation,* EquationSystems & *stateAdj*) [virtual]

Calcola la deformazione della mesh.

Parameters

out	*perturbation*	: Sistema d'equazioni contenente gli spostamenti da applicare alla mesh
in	*stateAdj*	: Sistema d'equazioni contenente stato e aggiunto

Implements ShapeOptimization.

Reimplemented in FFD_LS.

Definition at line 31 of file FFD.cc.

5.5.3.2 void applyPerturbation (const EquationSystems & *perturbation*) [virtual]

Applica la deformazione alla mesh.

Parameters

in	*perturbation*	: Sistema d'equazioni contenente gli spostamenti da applicare alla mesh

Implements ShapeOptimization.

Definition at line 143 of file FFD.cc.

5.5.3.3 Real basisFunction (const Point & *point,* const Index & *k,* const Index & *l*) const [virtual]

calcola la funzione di base k, l per il punto x

Parameters

in	*point*	: punto in cui calcolare la funzione di base
in	*k*	: indice orizzontale della griglia
in	*l*	: indice verticale della griglia

Returns

il valore della funzione

Definition at line 215 of file FFD.cc.

5.5.3.4 Point psi (const Point & *point*) const

mappa la scatola nel quadrato unitario

Parameters

in	*point*	: punto nella scatola da trasformare

Returns

le cordinate del punto nel quadrato unitario

Definition at line 224 of file FFD.cc.

5.5.3.5 Point psiInv (const Point & *ref_point*) const

mappa il quadrato unitario nel rettangolo di partenza

Parameters

in	*ref_point*	: punto nel quadrato di riferimento da trasformare

Returns

le cordinate del punto nel rettangolo di partenza

Definition at line 239 of file FFD.cc.

5.5.3.6 Point deform (const Point & *point*) const

applica la deformazione al punto

Parameters

in	*point*	: punto in cui calcolare la deformazione

Returns

punto deformato

Definition at line 251 of file FFD.cc.

The documentation for this class was generated from the following files:

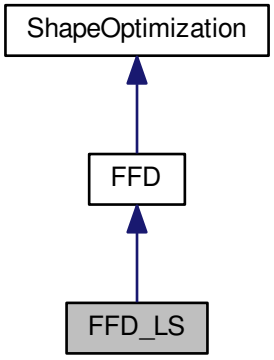
- src/FFD.h
- src/FFD.cc

5.6 FFD_LS Class Reference

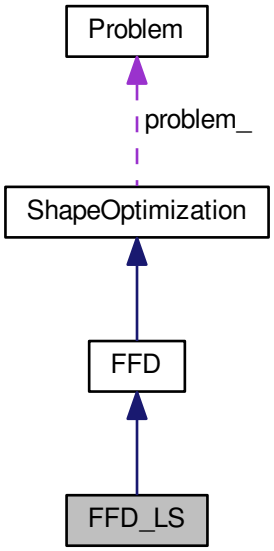
Classe che eredita dalla classe FFD, utilizza il metodo dei minimi quadrati con rilassamento per calcolare gli spostamenti da applicare ai control point.

`#include <FFD_LS.h>`

Inheritance diagram for FFD_LS:



Collaboration diagram for FFD_LS:



Public Member Functions

- FFD_LS (const Problem &, const std::string &, const Real &, const Index &, const Real &, const bool &, const std::pair< Point, Point > &, const std::pair< Index, Index > &, const Real &, const Real &=1.0e-4)

  Costruttore.
- virtual void computePerturbation (EquationSystems &, EquationSystems &)

  Calcola la deformazione della mesh.

Protected Attributes

- Real beta_

  Parametro di rilassamento per il metodo dei minimi quadrati.
- std::vector< Point > border_ref_

  Vettore contenente i punti del bordo.
- MatrixXr B_x

  Matrice con righe pari al numero di punti sul bordo e colonne pari al numero totale di control point. Rappresenta $\left(b_{k,\ell}^{K,L}(\vec{\psi}(\vec{x}))\mathfrak{B}\right)_x$ *per i nodi del bordo.*
- MatrixXr B_y

  Matrice con righe pari al numero di punti sul bordo e colonne pari al numero totale di control point. Rappresenta $\left(b_{k,\ell}^{K,L}(\vec{\psi}(\vec{x}))\mathfrak{B}\right)_y$ *per i nodi del bordo.*
- LDLT< MatrixXr > solver_x

  Robust Cholesky Decomposition con pivoting della matrice $\beta B\_x^T B\_x + (1-\beta)I$.
- LDLT< MatrixXr > solver_y

  Robust Cholesky Decomposition con pivoting della matrice $\beta B\_y^T B\_y + (1-\beta)I$.

5.6.1 Detailed Description

Classe che eredita dalla classe FFD, utilizza il metodo dei minimi quadrati con rilassamento per calcolare gli spostamenti da applicare ai control point.

La perturbazione dell'identità è rappresentata nel caso bidimensionale da $\vec{\theta}_{FFD} = (\theta_{FFD,x}, \theta_{FFD,y})^T \in \mathbb{R}^2$

$$\vec{\theta}_{FFD}(\vec{x}, \vec{\mu}) = \sum_{k=0}^{K} \sum_{\ell=0}^{L} b_{k,\ell}^{K,L}(\vec{\psi}(\vec{x}))\mathfrak{B}\vec{\mu}_{k,\ell}$$

Ora se uniamo la tripla sommatoria e trasformiamo in un vettore $\vec{\mu}_{k,\ell}$ e se consideriamo solo i NB nodi del bordo possiamo esprimere, definendo LL = K x L, la precedente formula con delle matrici $B_x, B_y \in \mathbb{R}^{NB \times LL}$ ottenendo

$$\vec{\theta}_{FFD,i} = B_i \vec{\mu}_i$$

Definition at line 39 of file FFD_LS.h.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 FFD_LS (const Problem & *problem,* const std::string & *directory,* const Real & *step,* const Index & *maxIterationsNo,* const Real & *tolerance,* const bool & *volume_constraint,* const std::pair< Point, Point > & *boundingBox,* const std::pair< Index, Index > & *sub,* const Real & *beta,* const Real & *armijoSlope* = `1.0e-4`)

Costruttore.

Parameters

in	*problem*	: Problema sul quale si vuole applicare la Shape Optimization
in	*directory*	: Directory in cui salvare i file di output
in	*step*	: Passo iniziale per il metodo di discesa del gradiente
in	*maxIterationsNo*	: Numero massimo di iterazioni
in	*tolerance*	: Tolleranza per il test d'arresto dell'incremento relativo
in	*volume_constraint*	: Specifica se applicare o meno il vincolo di volume

in	*boundingBox*	: Punti a nord est e a sud ovest indicanti il range della bounding box
in	*sub*	: Coppia contenente il numero di intervalli in cui suddividere la base e l'altezza della bounding box
in	*beta*	: Parametro di rilassamento per il metodo dei minimi quadrati
in	*armijoSlope*	: Coefficiente di rilassamento per la regola di Armijo.

Definition at line 3 of file FFD_LS.cc.

5.6.3 Member Function Documentation

5.6.3.1 void computePerturbation (EquationSystems & *perturbation*, EquationSystems & *stateAdj*) [virtual]

Calcola la deformazione della mesh.

Parameters

out	*perturbation*	: Sistema d'equazioni contenente gli spostamenti da applicare alla mesh
in	*stateAdj*	: Sistema d'equazioni contenente stato e aggiunto

Reimplemented from FFD.

Definition at line 47 of file FFD_LS.cc.

The documentation for this class was generated from the following files:

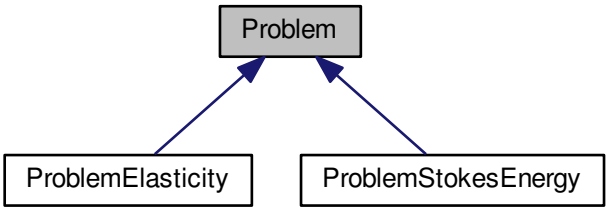
- src/FFD_LS.h
- src/FFD_LS.cc

5.7 Problem Class Reference

Classe astratta comune a tutti i problemi su cui si applica l'ottimizzazione.

`#include <Problem.h>`

Inheritance diagram for Problem:



Public Member Functions

- Problem (Mesh)

  Costruttore.
- virtual ~Problem ()=default

  Distruttore (defaulted)

- virtual void resolveStateAndAdjointEquation (EquationSystems &stateAdj, const Index &maxIterationsNo) const =0

  Metodo astratto per risolvere lo stato e l'aggiunto.
- virtual Real evaluateCostFunction (EquationSystems &stateAdj) const =0

  Metodo astratto per calcolare il valore del funzionale costo.
- virtual Real computeGradient (EquationSystems &stateAdj, const Point &p) const =0

  Metodo astratto per calcolare il valore del gradiente del funzionale costo in un punto.
- virtual Real sqrGradient (EquationSystems &stateAdj) const =0

  Metodo astratto per calcolare la norma L^2 del gradiente.
- virtual void harmonicExtension (EquationSystems &perturbation, EquationSystems &stateAdj, const Real &lagrange) const =0

  Metodo astratto per calcolare l'estensione armonica qualora fosse previsto dalla tecnica di ottimizzazione di forma.
- virtual bool toBeMoved (const Node &node) const =0

  Metodo astratto per valutare se un nodo può essere spostato o deve rimanere fisso.
- virtual void fixCP (const MatrixXp &CP_grid, MatrixXp &mu) const =0

  Metodo astratto per vincolare l'eventuale spostamento di control point.
- virtual Real lagrangeMult (EquationSystems &stateAdj) const =0

  Metodo astratto che calcola il moltiplicatore di lagrange.
- std::shared_ptr< Mesh > get_mesh () const

  Restituisce un puntatore alla mesh del problema.
- std::string get_name () const

  Restituisce il nome del sistema.

Protected Attributes

- std::shared_ptr< Mesh > mesh_

  puntatore alla mesh su cui è definito il problema
- std::string name_

  nome del problema che si vuole risolvere

5.7.1 Detailed Description

Classe astratta comune a tutti i problemi su cui si applica l'ottimizzazione.

Definition at line 29 of file Problem.h.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Problem (Mesh *mesh*)

Costruttore.

Parameters

in	*mesh*	: puntatore alla mesh sul quale è definito il problema

Definition at line 3 of file Problem.cc.

5.7.3 Member Function Documentation

5.7.3.1 virtual void resolveStateAndAdjointEquation (EquationSystems & *stateAdj,* const Index & *maxIterationsNo*) const [pure virtual]

Metodo astratto per risolvere lo stato e l'aggiunto.

Parameters

out	*stateAdj*	: Sistema d'equazioni che conterrà lo stato e l'aggiunto
in	*maxIterationsNo*	: Numero massimo di iterazioni

Implemented in ProblemElasticity, and ProblemStokesEnergy.

5.7.3.2 virtual Real evaluateCostFunction (EquationSystems & *stateAdj*) const [pure virtual]

Metodo astratto per calcolare il valore del funzionale costo.

Parameters

in	*stateAdj*	: Sistema d'equazioni che contiene lo stato e l'aggiunto

Returns

il valore del funzionale costo

Implemented in ProblemElasticity, and ProblemStokesEnergy.

5.7.3.3 virtual Real computeGradient (EquationSystems & *stateAdj,* const Point & *p*) const [pure virtual]

Metodo astratto per calcolare il valore del gradiente del funzionale costo in un punto.

Parameters

in	*stateAdj*	: Sistema d'equazioni che contiene lo stato e l'aggiunto
in	*p*	: Punto in cui calcolare il gradiente

Returns

il valore del gradiente nel punto

Implemented in ProblemElasticity, and ProblemStokesEnergy.

5.7.3.4 virtual Real sqrGradient (EquationSystems & *stateAdj*) const [pure virtual]

Metodo astratto per calcolare la norma L^2 del gradiente.

Parameters

in	*stateAdj*	: Sistema d'equazioni che contiene lo stato e l'aggiunto

Returns

il valore della norma L^2 del gradiente

Implemented in ProblemElasticity, and ProblemStokesEnergy.

5.7.3.5 virtual void harmonicExtension (EquationSystems & *perturbation,* EquationSystems & *stateAdj,* const Real & *lagrange*) const [pure virtual]

Metodo astratto per calcolare l'estensione armonica qualora fosse previsto dalla tecnica di ottimizzazione di forma.

Parameters

out	*perturbation*	: Sistema d'equazioni contenente gli spostamenti da applicare alla mesh
in	*stateAdj*	: Sistema d'equazioni contenente stato e aggiunto
in	*lagrange*	: lagrangiano

Implemented in ProblemElasticity, and ProblemStokesEnergy.

5.7.3.6 virtual bool toBeMoved (const Node & *node*) const [pure virtual]

Metodo astratto per valutare se un nodo può essere spostato o deve rimanere fisso.

Parameters

in	*node*	: nodo sul quale si vuole avere l'informazione

Returns

vero se il nodo può essere spostato

Implemented in ProblemElasticity, and ProblemStokesEnergy.

5.7.3.7 virtual void fixCP (const MatrixXp & *CP_grid,* MatrixXp & *mu*) const [pure virtual]

Metodo astratto per vincolare l'eventuale spostamento di control point.

Parameters

in	*CP_grid*	: la griglia dei control point
in	*mu*	: matrice contenente lo spostamento di ciascun control point nelle due direzioni

Implemented in ProblemElasticity, and ProblemStokesEnergy.

5.7.3.8 virtual Real lagrangeMult (EquationSystems & *stateAdj*) const [pure virtual]

Metodo astratto che calcola il moltiplicatore di lagrange.

Parameters

in	*stateAdj*	: Sistema d'equazioni contenente stato e aggiunto

Returns

il moltiplicatore di lagrange, che in particolare è l'integrale del gradiente moltiplicato per lo spostamento in direzione normale sul bordo diviso l'integrale degli spostamenti in direzione normale sul bordo

Implemented in ProblemElasticity, and ProblemStokesEnergy.

5.7.3.9 std::shared_ptr< Mesh > get_mesh ( ) const [inline]

Restituisce un puntatore alla mesh del problema.

Returns

il puntatore alla mesh

Definition at line 135 of file Problem.h.

5.7.3.10 std::string get_name ( ) const [inline]

Restituisce il nome del sistema.

Returns

il nome del sistema

Definition at line 140 of file Problem.h.

The documentation for this class was generated from the following files:

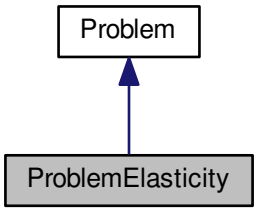
- src/Problem.h
- src/Problem.cc

5.8 ProblemElasticity Class Reference

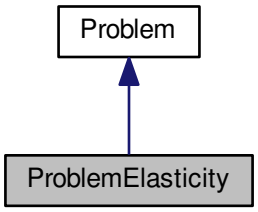
Classe che eredita da Problem e che rappresenta il problema dell'elasticità lineare.

`#include <ProblemElasticity.h>`

Inheritance diagram for ProblemElasticity:



Collaboration diagram for ProblemElasticity:



Public Member Functions

- ProblemElasticity (Mesh, const Real &, const Real &)

Costruttore.

- virtual void resolveStateAndAdjointEquation (EquationSystems &, const Index &) const

  Metodo per risolvere lo stato e l'aggiunto.
- virtual Real evaluateCostFunction (EquationSystems &) const

  Calcola il funzionale costo $\int_1 u_y d\sigma$.
- virtual Real computeGradient (EquationSystems &, const Point &) const

  Metodo per calcolare il valore del gradiente del funzionale costo in un punto.
- virtual Real sqrGradient (EquationSystems &) const

  Metodo per calcolare la norma L^2 *del gradiente.*
- virtual void harmonicExtension (EquationSystems &, EquationSystems &, const Real &) const

  Metodo astratto per calcolare l'estensione armonica qualora fosse previsto dalla tecnica di ottimizzazione di forma.;
- virtual bool toBeMoved (const Node &) const
- virtual void fixCP (const MatrixXp &, MatrixXp &) const
- virtual Real lagrangeMult (EquationSystems &) const

Protected Attributes

- Real coeff_lambda_

  Coefficiente di Lamé λ.
- Real coeff_mu_

  Coefficiente di Lamé μ.

Friends

- class **ElasticityHE**
- class **ElasticityState**

5.8.1 Detailed Description

Classe che eredita da Problem e che rappresenta il problema dell'elasticità lineare.

Definition at line 27 of file ProblemElasticity.h.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 ProblemElasticity (Mesh *mesh,* const Real & *lambda,* const Real & *mu*)

Costruttore.

Parameters

in	*mesh*	: puntatore alla mesh sul quale è definito il problema
in	*lambda*	: Coefficiente di Lamé λ
in	*mu*	: Coefficiente di Lamé μ

Definition at line 3 of file ProblemElasticity.cc.

5.8.3 Member Function Documentation

5.8.3.1 void resolveStateAndAdjointEquation (EquationSystems & *stateAdj,* const Index & *n*) const [virtual]

Metodo per risolvere lo stato e l'aggiunto.

Parameters

out	*stateAdj*	: Sistema d'equazioni che conterrà lo stato e l'aggiunto
in	*n*	: Numero massimo di iterazioni

Implements Problem.

Definition at line 9 of file ProblemElasticity.cc.

5.8.3.2 Real evaluateCostFunction (EquationSystems & *stateAdj*) const [virtual]

Calcola il funzionale costo $\int_1 u_y d\sigma$.

Parameters

out	*stateAdj*	: Sistema d'equazioni che contiene lo stato e l'aggiunto

Returns

il valore del funzionale costo

Implements Problem.

Definition at line 41 of file ProblemElasticity.cc.

5.8.3.3 Real computeGradient (EquationSystems & *stateAdj*, const Point & *p*) const [virtual]

Metodo per calcolare il valore del gradiente del funzionale costo in un punto.

Parameters

in	*stateAdj*	: Sistema d'equazioni che contiene lo stato e l'aggiunto
in	*p*	: Punto in cui calcolare il gradiente

Returns

il valore del gradiente nel punto

In questo caso il valore di $-2\mu|\varepsilon(u)|^2 - \lambda(tr(\varepsilon(u))^2)$ in un punto

Implements Problem.

Definition at line 92 of file ProblemElasticity.cc.

5.8.3.4 Real sqrGradient (EquationSystems & *stateAdj*) const [virtual]

Metodo per calcolare la norma L^2 del gradiente.

Parameters

in	*stateAdj*	: Sistema d'equazioni che contiene lo stato e l'aggiunto

Returns

il valore della norma L^2 del gradiente

Implements Problem.

Definition at line 111 of file ProblemElasticity.cc.

5.8.3.5 void harmonicExtension (EquationSystems & *perturbation,* EquationSystems & *stateAdj,* const Real & *lagrange*) const [virtual]

Metodo astratto per calcolare l'estensione armonica qualora fosse previsto dalla tecnica di ottimizzazione di forma.;

Parameters

out	*perturbation*	: Sistema d'equazioni contenente gli spostamenti da applicare alla mesh
in	*stateAdj*	: Sistema d'equazioni contenente stato e aggiunto
in	*lagrange*	: lagrangiano;

Implements Problem.

Definition at line 159 of file ProblemElasticity.cc.

5.8.3.6 bool toBeMoved (const Node &) const [virtual]

Implements Problem.

Definition at line 189 of file ProblemElasticity.cc.

5.8.3.7 void fixCP (const MatrixXp & *CP_grid,* MatrixXp & *mu*) const [virtual]

Implements Problem.

Definition at line 194 of file ProblemElasticity.cc.

5.8.3.8 Real lagrangeMult (EquationSystems & *stateAdj*) const [virtual]

Implements Problem.

Definition at line 211 of file ProblemElasticity.cc.

The documentation for this class was generated from the following files:

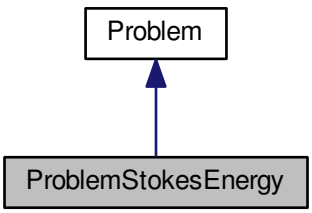
- src/ProblemElasticity.h
- src/ProblemElasticity.cc

5.9 ProblemStokesEnergy Class Reference

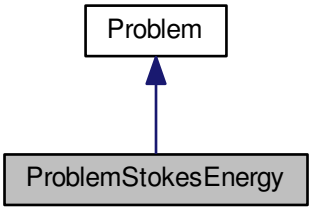
Classe che eredita da Problem e che rappresenta il problema di Stokes.

`#include <ProblemStokesEnergy.h>`

Inheritance diagram for ProblemStokesEnergy:



Collaboration diagram for ProblemStokesEnergy:



Public Member Functions

- ProblemStokesEnergy (Mesh, const Real &, const Real &)

  Costruttore.
- virtual void resolveStateAndAdjointEquation (EquationSystems &, const Index &) const

  Metodo per risolvere lo stato e l'aggiunto.
- virtual Real evaluateCostFunction (EquationSystems &) const

  Metodo astratto per calcolare il valore del funzionale costo.
- virtual Real computeGradient (EquationSystems &, const Point &) const

  Metodo per calcolare il valore del gradiente del funzionale costo in un punto.
- virtual Real sqrGradient (EquationSystems &) const

  Metodo per calcolare la norma L^2 del gradiente.
- virtual void harmonicExtension (EquationSystems &, EquationSystems &, const Real &) const

  Metodo astratto per calcolare l'estensione armonica qualora fosse previsto dalla tecnica di ottimizzazione di forma.;
- virtual bool toBeMoved (const Node &) const
- virtual void fixCP (const MatrixXp &, MatrixXp &) const
- virtual Real lagrangeMult (EquationSystems &) const

Protected Attributes

- Real ux_

  Componente lungo l'asse x della velocità in ingresso.
- Real uy_

  Componente lungo l'asse y della velocità in ingresso.

Friends

- class **StokesEnergyHE**
- class **StokesEnergyState**
- class **StokesEnergyAdjoint**

5.9.1 Detailed Description

Classe che eredita da Problem e che rappresenta il problema di Stokes.

Definition at line 26 of file ProblemStokesEnergy.h.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 ProblemStokesEnergy (Mesh *mesh,* const Real & *ux,* const Real & *uy*)

Costruttore.

Parameters

in	*mesh*	: puntatore alla mesh sul quale è definito il problema
in	*ux*	: Componente lungo l'asse x della velocità in ingresso
in	*uy*	: Componente lungo l'asse y della velocità in ingresso

Definition at line 3 of file ProblemStokesEnergy.cc.

5.9.3 Member Function Documentation

5.9.3.1 void resolveStateAndAdjointEquation (EquationSystems & *stateAdj,* const Index & *n*) const [virtual]

Metodo per risolvere lo stato e l'aggiunto.

Parameters

out	*stateAdj*	: Sistema d'equazioni che conterrà lo stato e l'aggiunto
in	*n*	: Numero massimo di iterazioni

Implements Problem.

Definition at line 9 of file ProblemStokesEnergy.cc.

5.9.3.2 Real evaluateCostFunction (EquationSystems & *stateAdj*) const [virtual]

Metodo astratto per calcolare il valore del funzionale costo.

Parameters

in	*stateAdj*	: Sistema d'equazioni che contiene lo stato e l'aggiunto

Returns

il valore del funzionale costo

Implements Problem.

Definition at line 128 of file ProblemStokesEnergy.cc.

5.9.3.3 Real computeGradient (EquationSystems & *stateAdj,* const Point & *p*) const [virtual]

Metodo per calcolare il valore del gradiente del funzionale costo in un punto.

Parameters

in	*stateAdj*	: Sistema d'equazioni che contiene lo stato e l'aggiunto
in	*p*	: Punto in cui calcolare il gradiente

Returns

il valore del gradiente nel punto

Implements Problem.

Definition at line 166 of file ProblemStokesEnergy.cc.

5.9.3.4 Real sqrGradient (EquationSystems & *stateAdj*) const [virtual]

Metodo per calcolare la norma L^2 del gradiente.

Parameters

in	*stateAdj*	: Sistema d'equazioni che contiene lo stato e l'aggiunto

Returns

il valore della norma L^2 del gradiente

Implements Problem.

Definition at line 177 of file ProblemStokesEnergy.cc.

5.9.3.5 void harmonicExtension (EquationSystems & *perturbation,* EquationSystems & *stateAdj,* const Real & *lagrange*) const [virtual]

Metodo astratto per calcolare l'estensione armonica qualora fosse previsto dalla tecnica di ottimizzazione di forma.;

Parameters

out	*perturbation*	: Sistema d'equazioni contenente gli spostamenti da applicare alla mesh
in	*stateAdj*	: Sistema d'equazioni contenente stato e aggiunto
in	*lagrange*	: lagrangiano;

Implements Problem.

Definition at line 225 of file ProblemStokesEnergy.cc.

5.9.3.6 bool toBeMoved (const Node & *node*) const [virtual]

Implements Problem.

Definition at line 255 of file ProblemStokesEnergy.cc.

5.9.3.7 void fixCP (const MatrixXp & *CP_grid,* MatrixXp & *mu*) const [virtual]

Implements Problem.

Definition at line 261 of file ProblemStokesEnergy.cc.

5.9.3.8 Real lagrangeMult (EquationSystems & *stateAdj*) const [virtual]

Implements Problem.

Definition at line 278 of file ProblemStokesEnergy.cc.

The documentation for this class was generated from the following files:

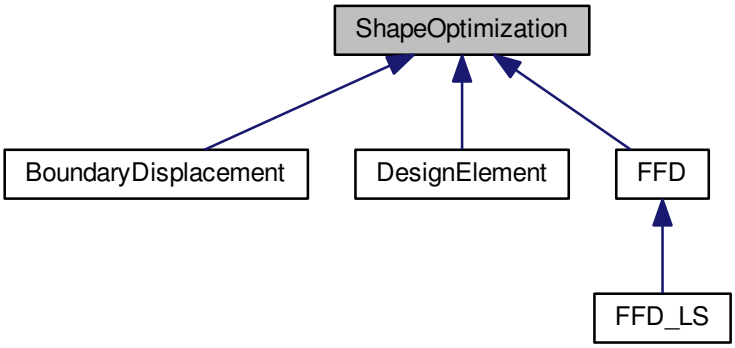
- src/ProblemStokesEnergy.h
- src/ProblemStokesEnergy.cc

5.10 ShapeOptimization Class Reference

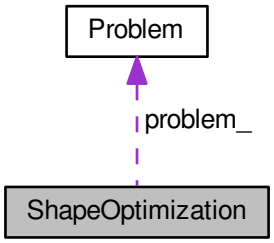
Classe astratta comune a tutte le tecniche di ottimizzazione.

`#include <ShapeOptimization.h>`

Inheritance diagram for ShapeOptimization:



Collaboration diagram for ShapeOptimization:



Public Member Functions

- ShapeOptimization (const Problem &, const std::string &, const Real &, const Index &, const Real &, const bool &, const Real &=1.0e-4)

  Costruttore.
- virtual ~ShapeOptimization ()=default

  Distruttore (defaulted)
- void apply ()

  Esegue il ciclo d'ottimizzazione.
- virtual void computePerturbation (EquationSystems &perturbation, EquationSystems &stateAdj)=0

  Metodo astratto per calcolare la deformazione della mesh.
- virtual void applyPerturbation (const EquationSystems &perturbation)=0

  Metodo astratto per applicare la deformazione alla mesh.
- void updateLagrange (const Real &)

  Aggiorna il valore del moltiplicatore di lagrange $l_{k+1} = \frac{l+l_k}{2} + \frac{V-V_0}{V_0}$.
- Real getVolume () const

Misura l'area della mesh.

- void checkDomain () const

  Controlla se non si sono invertiti dei triangoli della mesh in seguito alla deformazione.

Protected Attributes

- const Problem & problem_

  problema che si vuole ottimizzare
- std::string plotName_

  nome utilizzato nella generazione dei file di output
- std::shared_ptr< Mesh > mesh_

  puntatore alla mesh su cui è definito il problema
- Real step_

  passo utilizzato per il metodo di discesa del gradiente
- Index maxIterationsNo_

  numero massimo di iterazioni
- Real tolerance_

  tolleranza per il test d'arresto dell'incremento relativo
- bool volume_constraint_

  specifica se applicare o meno il vincolo di volume
- Real armijoSlope_

  coefficiente di rilassamento per la regola di Armijo
- Real old_lagrange_

  valore del lagrangiano al passo d'ottimizzazione precedente
- Real actual_lagrange_

  valore del lagrangiano al passo d'ottimizzazione attuale
- Real initialVolume_

  area iniziale della mesh

5.10.1 Detailed Description

Classe astratta comune a tutte le tecniche di ottimizzazione.

Definition at line 34 of file ShapeOptimization.h.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 ShapeOptimization (const Problem & *problem*, const std::string & *directory*, const Real & *step*, const Index & *maxIterationsNo*, const Real & *tolerance*, const bool & *volume_constraint*, const Real & *armijoSlope* = `1.0e-4`)

Costruttore.

Parameters

`in`	*problem*	: Problema sul quale si vuole applicare la Shape Optimization
`in`	*directory*	: Directory in cui salvare i file di output
`in`	*step*	: Passo iniziale per il metodo di discesa del gradiente
`in`	*maxIterationsNo*	: Numero massimo di iterazioni

in	*tolerance*	: Tolleranza per il test d'arresto dell'incremento relativo
in	*volume_-constraint*	: Specifica se applicare o meno il vincolo di volume
in	*armijoSlope*	: Coefficiente di rilassamento per la regola di Armijo.

Definition at line 3 of file ShapeOptimization.cc.

5.10.3 Member Function Documentation

5.10.3.1 virtual void computePerturbation (EquationSystems & *perturbation,* EquationSystems & *stateAdj*) [pure virtual]

Metodo astratto per calcolare la deformazione della mesh.

Parameters

out	*perturbation*	: Sistema d'equazioni contenente gli spostamenti da applicare alla mesh
in	*stateAdj*	: Sistema d'equazioni contenente stato e aggiunto

Implemented in FFD_LS, DesignElement, FFD, and BoundaryDisplacement.

5.10.3.2 virtual void applyPerturbation (const EquationSystems & *perturbation*) [pure virtual]

Metodo astratto per applicare la deformazione alla mesh.

Parameters

in	*perturbation*	: Sistema d'equazioni contenente gli spostamenti da applicare alla mesh

Implemented in DesignElement, FFD, and BoundaryDisplacement.

5.10.3.3 void updateLagrange (const Real & *lagrange*)

Aggiorna il valore del moltiplicatore di lagrange $l_{k+1} = \frac{l+l_k}{2} + \frac{V-V_0}{V_0}$.

Parameters

in	*lagrange*	: media del gradiente sul bordo $l = \frac{\int_{\partial\Omega} -\nabla J d\sigma}{\int_{\partial\Omega} d\sigma}$

Definition at line 169 of file ShapeOptimization.cc.

5.10.3.4 Real getVolume ( ) const

Misura l'area della mesh.

Returns

il valore dell'area della mesh

Definition at line 175 of file ShapeOptimization.cc.

The documentation for this class was generated from the following files:

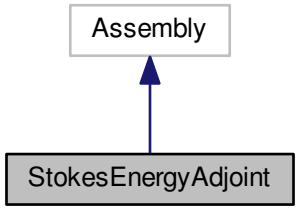
- src/ShapeOptimization.h
- src/ShapeOptimization.cc

5.11 StokesEnergyAdjoint Class Reference

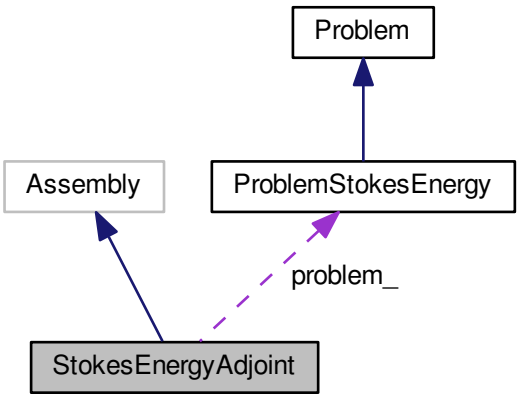
Classe contenente i metodi necessari per calcolare l'aggiunto nel problema di Stokes.

`#include <ProblemStokesEnergy.h>`

Inheritance diagram for StokesEnergyAdjoint:



Collaboration diagram for StokesEnergyAdjoint:



Public Member Functions

- StokesEnergyAdjoint (EquationSystems &, const ProblemStokesEnergy &)

  Costruttore.
- void assemble ()

  Assembla le matrici e i vettori per calcolare l'aggiunto nel problema di Stokes.

Private Attributes

- EquationSystems & stateAdj_

Sistemi d'equazioni contenente lo stato e l'aggiunto.

- const ProblemStokesEnergy & problem_

  Riferimento costante al problema di Stokes.

5.11.1 Detailed Description

Classe contenente i metodi necessari per calcolare l'aggiunto nel problema di Stokes.

Definition at line 143 of file ProblemStokesEnergy.h.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 StokesEnergyAdjoint (EquationSystems & *stateAdj*, const ProblemStokesEnergy & *problem*)

Costruttore.

Parameters

in	*stateAdj*	: Riferimento ai sistemi d'equazioni contenente lo stato e l'aggiunto
in	*problem*	: Riferimento costante al problema di Stokes

Definition at line 587 of file ProblemStokesEnergy.cc.

The documentation for this class was generated from the following files:

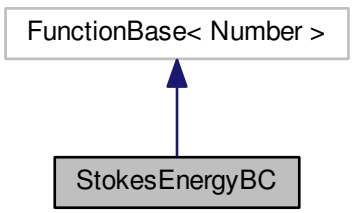
- src/ProblemStokesEnergy.h
- src/ProblemStokesEnergy.cc

5.12 StokesEnergyBC Class Reference

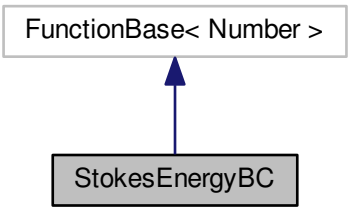
Classe contenente i metodi necessari per imporre le condizioni al bordo nel problema di Stokes.

`#include <ProblemStokesEnergy.h>`

Inheritance diagram for StokesEnergyBC:



Collaboration diagram for StokesEnergyBC:



Public Member Functions

- StokesEnergyBC (const Index &u_var, const Index &v_var, const Real &ux, const Real &uy)

  Costruttore.
- virtual Number operator() (const Point &, const Real=0)

  Operatore() non implementato.
- virtual void operator() (const Point &p, const Real time, DenseVector< Number > &output)

  Operatore() che restituisce il valore delle condizioni al bordo.
- virtual AutoPtr< FunctionBase < Number > > clone () const

  Metodo per clonare l'oggetto.

Private Attributes

- const Index u_var_

  Indice della variabile che fa riferimento alla componente x della velocità
- const Index v_var_

  Indice della variabile che fa riferimento alla componente Y della velocità
- const Real ux_

  Componente lungo l'asse x della velocità in ingresso.
- const Real uy_

  Componente lungo l'asse y della velocità in ingresso.

5.12.1 Detailed Description

Classe contenente i metodi necessari per imporre le condizioni al bordo nel problema di Stokes.

Definition at line 167 of file ProblemStokesEnergy.h.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 StokesEnergyBC (const Index & *u_var,* const Index & *v_var,* const Real & *ux,* const Real & *uy*) [inline]

Costruttore.

Parameters

in	*u_var*	: Indice della variabile che fa riferimento alla componente x della velocità
in	*v_var*	: Indice della variabile che fa riferimento alla componente y della velocità
in	*ux*	: Componente lungo l'asse x della velocità in ingresso
in	*uy*	: Componente lungo l'asse y della velocità in ingresso

Definition at line 178 of file ProblemStokesEnergy.h.

5.12.3 Member Function Documentation

5.12.3.1 virtual void operator() (const Point & *p*, const Real *time*, DenseVector< Number > & *output*) [inline], [virtual]

Operatore() che restituisce il valore delle condizioni al bordo.

Parameters

in	*p*	: Punto del bordo in cui valutare le condizioni al bordo
in	*time*	: Istante temporale considerato
out	*output*	: Vettore contenente le componenti delle condizioni al bordo nel punto e all'istante considerato

Definition at line 200 of file ProblemStokesEnergy.h.

5.12.3.2 virtual AutoPtr<FunctionBase<Number> > clone ( ) const [inline], [virtual]

Metodo per clonare l'oggetto.

Returns

una nuova copia dell'oggetto attuale.

Definition at line 216 of file ProblemStokesEnergy.h.

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

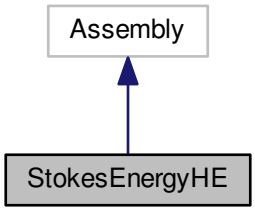
- src/ProblemStokesEnergy.h

5.13 StokesEnergyHE Class Reference

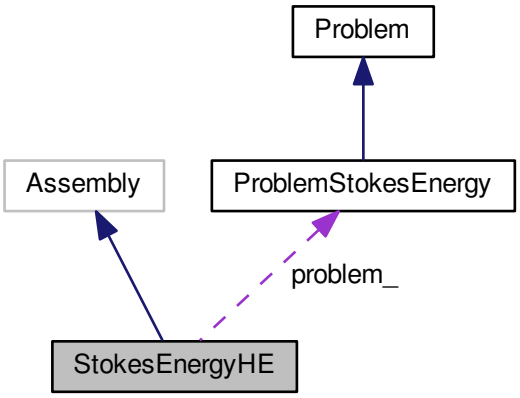
Classe contenente i metodi necessari per calcolare l'estensione armonica nel problema di Stokes.

`#include <ProblemStokesEnergy.h>`

Inheritance diagram for StokesEnergyHE:



Collaboration diagram for StokesEnergyHE:



Public Member Functions

- StokesEnergyHE (EquationSystems &, EquationSystems &, const Real &, const ProblemStokesEnergy &)

  Costruttore.
- void assemble ()

  Assembla le matrici e i vettori per calcolare l'estensione armonica nel problema dell'elasticità

Private Attributes

- EquationSystems & perturbation_

  Sistemi d'equazioni per gestire lo spostamento della mesh.
- EquationSystems & stateAdj_

  Sistemi d'equazioni contenente lo stato e l'aggiunto.
- Real lagrange_

  Moltiplicatore di lagrange.

- const ProblemStokesEnergy & problem_

  Riferimento costante al problema di Stokes.

5.13.1 Detailed Description

Classe contenente i metodi necessari per calcolare l'estensione armonica nel problema di Stokes.

Definition at line 91 of file ProblemStokesEnergy.h.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 StokesEnergyHE (EquationSystems & *perturbation,* EquationSystems & *stateAdj,* const Real & *lagrange,* const ProblemStokesEnergy & *problem*)

Costruttore.

Parameters

in	*perturbation*	: Riferimento ai sistemi d'equazioni per lo spostamento della mesh
in	*stateAdj*	: Riferimento ai sistemi d'equazioni contenente lo stato e l'aggiunto
in	*lagrange*	: Moltiplicatore di lagrange
in	*problem*	: Riferimento costante al problema di Stokes

Definition at line 331 of file ProblemStokesEnergy.cc.

The documentation for this class was generated from the following files:

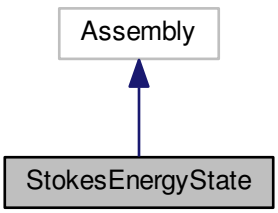
- src/ProblemStokesEnergy.h
- src/ProblemStokesEnergy.cc

5.14 StokesEnergyState Class Reference

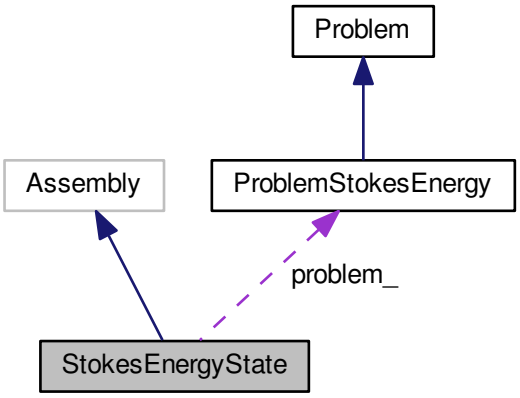
Classe contenente i metodi necessari per calcolare lo stato nel problema di Stokes.

`#include <ProblemStokesEnergy.h>`

Inheritance diagram for StokesEnergyState:



Collaboration diagram for StokesEnergyState:



Public Member Functions

- StokesEnergyState (EquationSystems &, const ProblemStokesEnergy &)

  Costruttore.
- void assemble ()

  Assembla le matrici e i vettori per calcolare lo stato nel problema di Stokes.

Private Attributes

- EquationSystems & stateAdj_

  Sistemi d'equazioni contenente lo stato e l'aggiunto.
- const ProblemStokesEnergy & problem_

  Riferimento costante al problema di Stokes.

5.14.1 Detailed Description

Classe contenente i metodi necessari per calcolare lo stato nel problema di Stokes.

Definition at line 119 of file ProblemStokesEnergy.h.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 StokesEnergyState (EquationSystems & *stateAdj*, const ProblemStokesEnergy & *problem*)

Costruttore.

Parameters

in	*stateAdj*	: Riferimento ai sistemi d'equazioni contenente lo stato e l'aggiunto
in	*problem*	: Riferimento costante al problema di Stokes

Definition at line 446 of file ProblemStokesEnergy.cc.

The documentation for this class was generated from the following files:

- src/ProblemStokesEnergy.h
- src/ProblemStokesEnergy.cc

Chapter 6

File Documentation

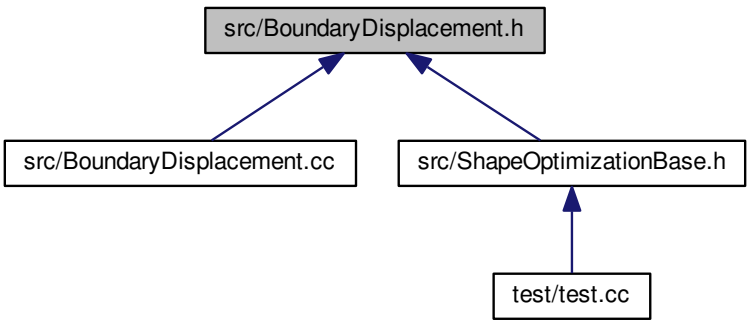
6.1 src/BoundaryDisplacement.h File Reference

Confronto tra alcune tecniche per l'ottimizzazione di forma.

```
#include "ShapeOptimization.h"
```

Include dependency graph for BoundaryDisplacement.h:

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



Classes

- class BoundaryDisplacement

  Classe che eredita da ShapeOptimization. Utilizza la tecnica del boundary local displacement per eseguire l'ottimizzazione di forma.

6.1.1 Detailed Description

Confronto tra alcune tecniche per l'ottimizzazione di forma.

Author

Pasquale Claudio Africa pasquale.africa@mail.polimi.it, Luca Ratti luca3.ratti@mail.polimi.it, Abele Simona abele.simona@mail.polimi.it

Date

2015

Questo file fa parte del progetto "ShapeOpt".

Copyright

Definition in file BoundaryDisplacement.h.

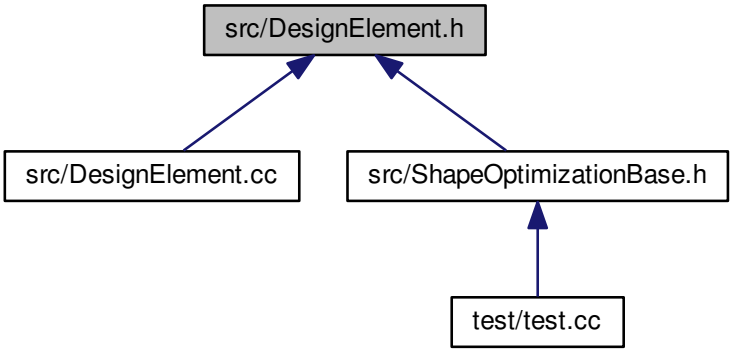
6.2 src/DesignElement.h File Reference

Confronto tra alcune tecniche per l'ottimizzazione di forma.

```
#include "ShapeOptimization.h"
```

Include dependency graph for DesignElement.h:

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



Classes

- class DesignElement

  Classe che eredita dalla classe ShapeOptimization, utilizza il metodo del Design Element descrivendo il bordo superiore e quello inferiore con polinomi di grado arbitrario.

6.2.1 Detailed Description

Confronto tra alcune tecniche per l'ottimizzazione di forma.

Author

Pasquale Claudio Africa pasquale.africa@mail.polimi.it, Luca Ratti luca3.ratti@mail.polimi.it, Abele Simona abele.simona@mail.polimi.it

Date

2015

Questo file fa parte del progetto "ShapeOpt".

Copyright

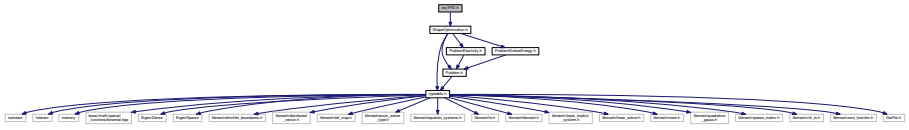
Definition in file DesignElement.h.

6.3 src/FFD.h File Reference

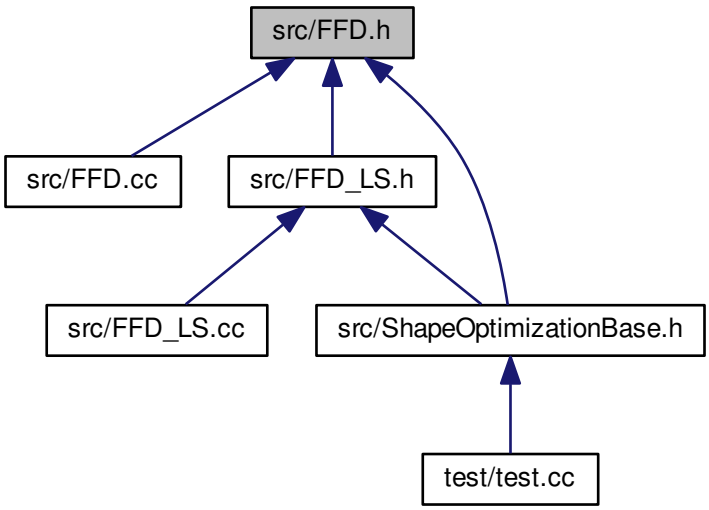
Confronto tra alcune tecniche per l'ottimizzazione di forma.

```
#include "ShapeOptimization.h"
```

Include dependency graph for FFD.h:



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



Classes

- class FFD

  Classe che eredita dalla classe ShapeOptimization, utilizza il metodo della Free Form Deformation utilizzando come funzioni di base le B-Spline.

6.3.1 Detailed Description

Confronto tra alcune tecniche per l'ottimizzazione di forma.

Author

Pasquale Claudio Africa pasquale.africa@mail.polimi.it, Luca Ratti luca3.ratti@mail.polimi.it, Abele Simona abele.simona@mail.polimi.it

Date

2015

Questo file fa parte del progetto "ShapeOpt".

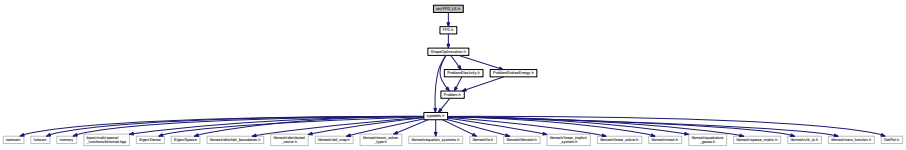
Copyright

Definition in file FFD.h.

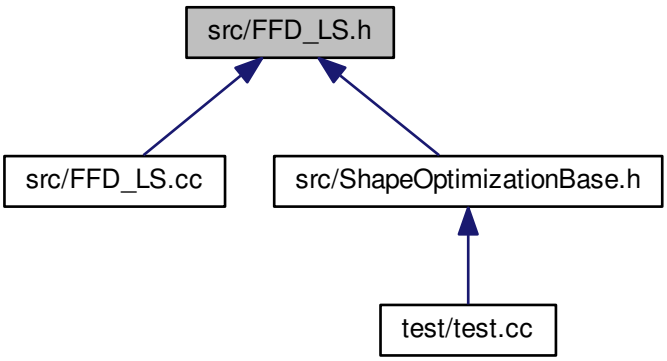
6.4 src/FFD_LS.h File Reference

Confronto tra alcune tecniche per l'ottimizzazione di forma.

`#include "FFD.h"`
Include dependency graph for FFD_LS.h:



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



Classes

- class FFD_LS

  Classe che eredita dalla classe FFD, utilizza il metodo dei minimi quadrati con rilassamento per calcolare gli spostamenti da applicare ai control point.

6.4.1 Detailed Description

Confronto tra alcune tecniche per l'ottimizzazione di forma.

Author

Pasquale Claudio Africa `pasquale.africa@mail.polimi.it`, Luca Ratti `luca3.ratti@mail.polimi.it`, Abele Simona `abele.simona@mail.polimi.it`

Date

2015

Questo file fa parte del progetto "ShapeOpt".

Copyright

Definition in file FFD_LS.h.

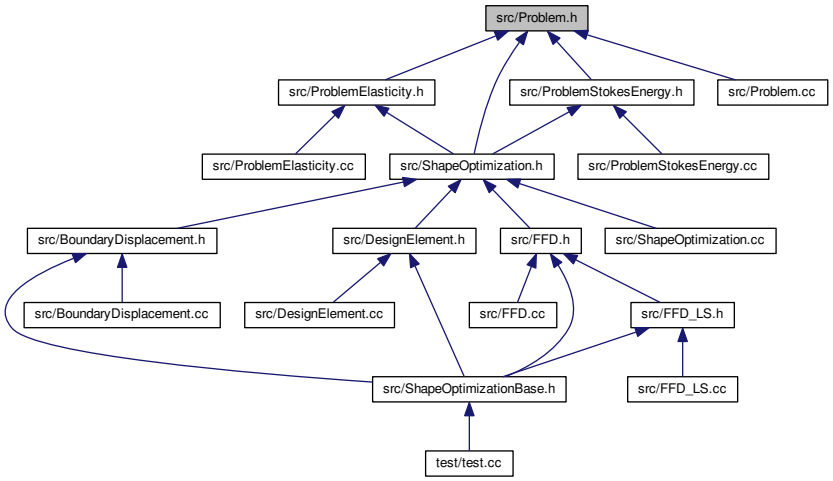
6.5 src/Problem.h File Reference

Confronto tra alcune tecniche per l'ottimizzazione di forma.

```
#include "typedefs.h"
```

Include dependency graph for Problem.h:

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



Classes

- class Problem

  Classe astratta comune a tutti i problemi su cui si applica l'ottimizzazione.

6.5.1 Detailed Description

Confronto tra alcune tecniche per l'ottimizzazione di forma.

Author

Pasquale Claudio Africa pasquale.africa@mail.polimi.it, Luca Ratti luca3.ratti@mail.polimi.it, Abele Simona abele.simona@mail.polimi.it

Date

2015

Questo file fa parte del progetto "ShapeOpt".

Copyright

Definition in file Problem.h.

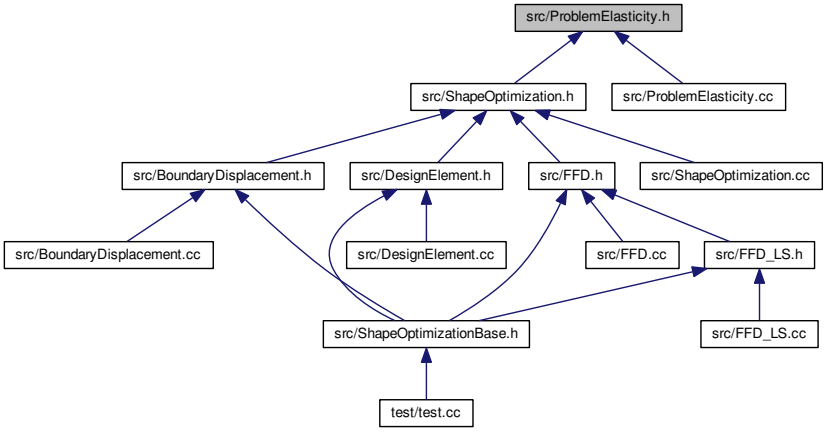
6.6 src/ProblemElasticity.h File Reference

Confronto tra alcune tecniche per l'ottimizzazione di forma.

```
#include "Problem.h"
```

Include dependency graph for ProblemElasticity.h:

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



Classes

- class ProblemElasticity

  Classe che eredita da Problem e che rappresenta il problema dell'elasticità lineare.
- class ElasticityHE

  Classe contenente i metodi necessari per calcolare l'estensione armonica nel problema dell'elasticità
- class ElasticityState

  Classe contenente i metodi necessari per calcolare lo stato (il sistema è autoaggiunto) nel problema dell'elasticità

6.6.1 Detailed Description

Confronto tra alcune tecniche per l'ottimizzazione di forma.

Author

Pasquale Claudio Africa pasquale.africa@mail.polimi.it, Luca Ratti luca3.ratti@mail.polimi.it, Abele Simona abele.simona@mail.polimi.it

Date

2015

Questo file fa parte del progetto "ShapeOpt".

Copyright

Definition in file ProblemElasticity.h.

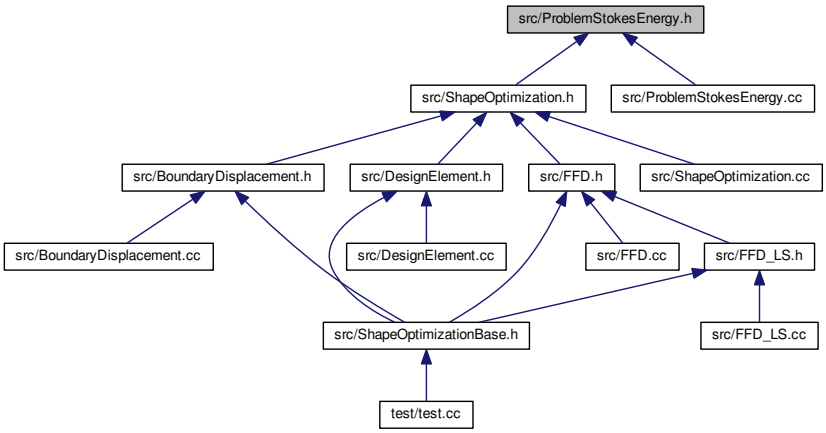
6.7 src/ProblemStokesEnergy.h File Reference

Confronto tra alcune tecniche per l'ottimizzazione di forma.

```
#include "Problem.h"
```

Include dependency graph for ProblemStokesEnergy.h:

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



Classes

- class ProblemStokesEnergy

  Classe che eredita da Problem e che rappresenta il problema di Stokes.
- class StokesEnergyHE

  Classe contenente i metodi necessari per calcolare l'estensione armonica nel problema di Stokes.
- class StokesEnergyState

  Classe contenente i metodi necessari per calcolare lo stato nel problema di Stokes.
- class StokesEnergyAdjoint

  Classe contenente i metodi necessari per calcolare l'aggiunto nel problema di Stokes.
- class StokesEnergyBC

  Classe contenente i metodi necessari per imporre le condizioni al bordo nel problema di Stokes.

6.7.1 Detailed Description

Confronto tra alcune tecniche per l'ottimizzazione di forma.

Author

Pasquale Claudio Africa pasquale.africa@mail.polimi.it, Luca Ratti luca3.ratti@mail.polimi.it, Abele Simona abele.simona@mail.polimi.it

Date

2015

Questo file fa parte del progetto "ShapeOpt".

Copyright

Definition in file ProblemStokesEnergy.h.

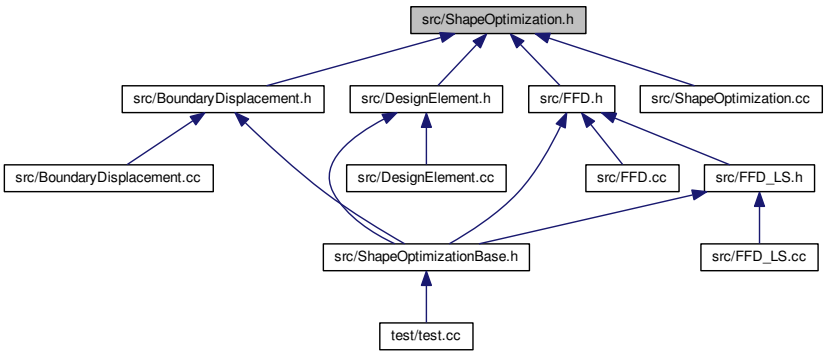
6.8 src/ShapeOptimization.h File Reference

Confronto tra alcune tecniche per l'ottimizzazione di forma.

```
#include "typedefs.h"
#include "Problem.h"
#include "ProblemElasticity.h"
#include "ProblemStokesEnergy.h"
```

Include dependency graph for ShapeOptimization.h:

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



Classes

- class ShapeOptimization

  Classe astratta comune a tutte le tecniche di ottimizzazione.

6.8.1 Detailed Description

Confronto tra alcune tecniche per l'ottimizzazione di forma.

Author

Pasquale Claudio Africa pasquale.africa@mail.polimi.it, Luca Ratti luca3.ratti@mail.polimi.it, Abele Simona abele.simona@mail.polimi.it

Date

2015

Questo file fa parte del progetto "ShapeOpt".

Copyright

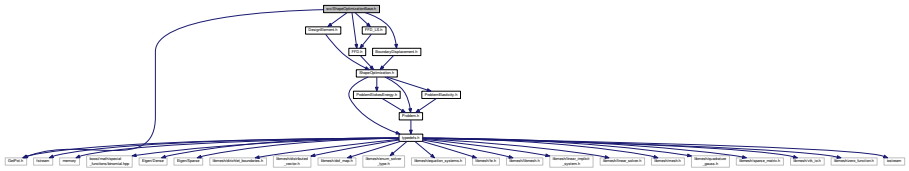
Definition in file ShapeOptimization.h.

6.9 src/ShapeOptimizationBase.h File Reference

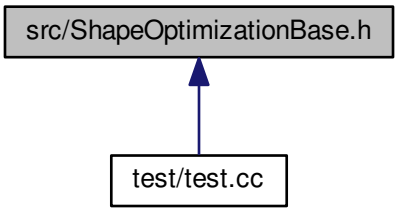
Confronto tra alcune tecniche per l'ottimizzazione di forma.

```
#include "BoundaryDisplacement.h"
#include "DesignElement.h"
#include "FFD.h"
#include "FFD_LS.h"
#include "GetPot.h"
```

Include dependency graph for ShapeOptimizationBase.h:



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



6.9.1 Detailed Description

Confronto tra alcune tecniche per l'ottimizzazione di forma.

Author

Pasquale Claudio Africa pasquale.africa@mail.polimi.it, Luca Ratti luca3.ratti@mail.polimi.it, Abele Simona abele.simona@mail.polimi.it

Date

2015

Questo file fa parte del progetto "ShapeOpt".

Copyright

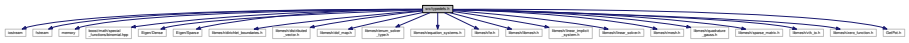
Definition in file ShapeOptimizationBase.h.

6.10 src/typedefs.h File Reference

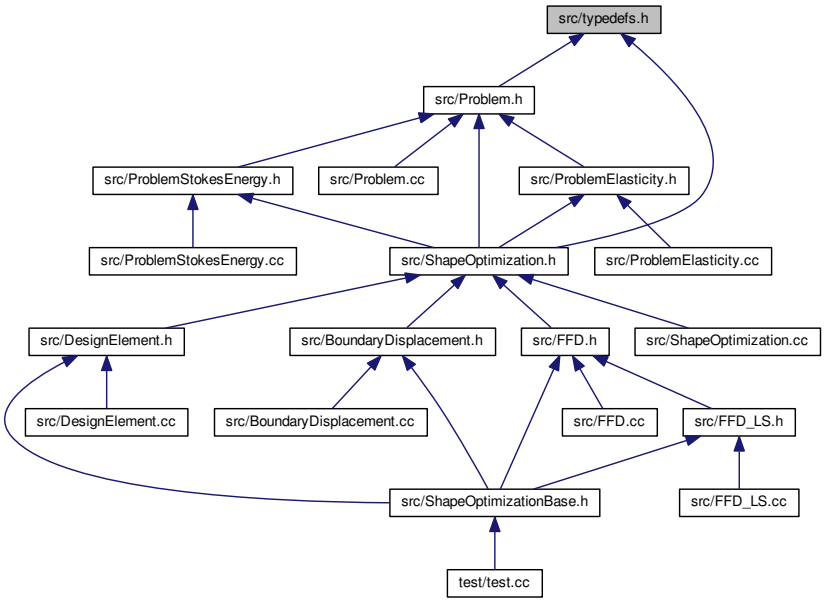
Confronto tra alcune tecniche per l'ottimizzazione di forma.

```
#include <iostream>
#include <fstream>
#include <memory>
#include <boost/math/special_functions/binomial.hpp>
#include <Eigen/Dense>
#include <Eigen/Sparse>
#include "libmesh/dirichlet_boundaries.h"
#include "libmesh/distributed_vector.h"
#include "libmesh/dof_map.h"
#include "libmesh/enum_solver_type.h"
#include "libmesh/equation_systems.h"
#include "libmesh/fe.h"
#include "libmesh/libmesh.h"
#include "libmesh/linear_implicit_system.h"
#include "libmesh/linear_solver.h"
#include "libmesh/mesh.h"
#include "libmesh/quadrature_gauss.h"
#include "libmesh/sparse_matrix.h"
#include "libmesh/vtk_io.h"
#include "libmesh/zero_function.h"
#include "GetPot.h"
```

Include dependency graph for typedefs.h:



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



Typedefs

- using Real = double

  Typedef for real numbers.

- using Index = ptrdiff_t

  Typedef for indexing variables.
- using MatrixXp = Matrix< Point, Dynamic, Dynamic >

  Typedef for dense dynamic-sized matrices of points.
- using VectorXp = Matrix< Point, Dynamic, 1 >

  Typedef for dense dynamic-sized column vectors of points.
- using MatrixXr = Matrix< Real, Dynamic, Dynamic >

  Typedef for dense real-valued dynamic-sized matrices.
- using VectorXr = Matrix< Real, Dynamic, 1 >

  Typedef for dense real-valued dynamic-sized column vectors.
- using SparseXr = Eigen::SparseMatrix< Real >

  Typedef for sparse real-valued dynamic-sized matrices.

6.10.1 Detailed Description

Confronto tra alcune tecniche per l'ottimizzazione di forma.

Author

Pasquale Claudio Africa pasquale.africa@mail.polimi.it, Luca Ratti luca3.ratti@mail.polimi.it, Abele Simona abele.simona@mail.polimi.it

Date

2015

Questo file fa parte del progetto "ShapeOpt".

Copyright

Definition in file typedefs.h.

Index