Tspeed

Generated by Doxygen 1.8.3.1

Sun Sep 8 2013 22:31:00

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

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 files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 Tspeed Namespace Reference

Namespaces

- namespace Geo

Classes

- class FESpace

  Functional space.
- class Parameters

  Class for the parameters λ, ρ, μ of the elastodynamics equations.
- class Force

  virtual base class for forces
- class PointWiseForce

  Time dependent force acting on a point.
- class Entity

  Base class for geometrical entities.
- class Mesh
- class BaseMat

  Base monodimensional matrix class.
- class MyMatBlockDiag

  Block diagonal monodimensional matrix (monodimensional block of mass and stress-strain matrices)
- class MyMat

  Block Matrix (monodimensial blocks of stability and interelement matrices)
- class MyMatMultiDim

  Multidimensional matrix.
- class MyMatMultiDimBlockDiag

  Matrix class for matrices where only the diagonal monodimensional matrices are non zero (i.e. the mass matrix)
- class QuadratureRule

  Base class for quadrature rules.
- class Gauss

  Gauss quadrature rule on the triangle.
- class Dunavant

  Dunavant [1] quadrature rule.
- class PointWiseEntity

A base class for pointwise entities, with the points and the basis function in that points.

- class Receivers

  A class for seismic receivers, i.e., receivers recording the movement at a point.
- class ShapeFunction

  Base class for the shared functions.
- class Dubiner

  Dubiner [1] basis.
- class BoundaryAdapted

  Boundary adapted [2] basis.
- class Matrices

  The class containing the matrices resulting from the spatial discretization.
- class TimeAdvance

  Base class for time stepping methods.
- class LeapFrog

  Implementation of the second order Leap-Frog explicit time stepping scheme.

Typedefs

- template<int N, typename Q = Gauss<N+1>, typename S = Dubiner<N>>
  using FESpace_ptr = std::shared_ptr< FESpace< N, Q, S >>

  template pointer to FESpace
- typedef std::shared_ptr< Force > Force_ptr
- typedef std::shared_ptr< Mesh > Mesh_ptr

  Shared pointer to an element of type mesh.

Enumerations

- enum Bc { Dirichlet, Neumann, Internal, Unassigned }

Functions

- MyMat operator* (double const &c, MyMat const &M)
- Eigen::VectorXd operator* (MyMat const &, Eigen::VectorXd const &)
- Eigen::VectorXd operator* (MyMatBlockDiag const &, Eigen::VectorXd const &)
- MyMat operator+ (MyMat a, MyMat const &b)
- MyMat operator+ (MyMat a, MyMatBlockDiag const &b)
- Eigen::ArrayXd jacobi_polynomial (int N, int alpha, int beta, Eigen::ArrayXd const &z)
- double mat_dot (Eigen::Matrix2d const &a, Eigen::Matrix2d const &b)

  Dot product between two 2x2 matrices.
- Eigen::Matrix2d CTensorProduct (Eigen::Matrix2d const &A, double lambda, double mu)

  tensor product between Hooke's tensor and matrix A
- Eigen::VectorXd operator* (MyMatMultiDimBlockDiag< MyMatBlockDiag > const &A, Eigen::VectorXd const &v)
- Eigen::VectorXd operator* (MyMatMultiDim< MyMat > &A, Eigen::VectorXd const &v)
- Eigen::VectorXd operator* (MyMatMultiDim< MyMatBlockDiag > &A, Eigen::VectorXd const &v)
- MyMatMultiDim< MyMat > operator+ (MyMatMultiDim< MyMat > const &a, MyMatMultiDim< MyMat > const &b)
- MyMatMultiDim< MyMat > operator+ (MyMatMultiDim< MyMat > const &a, MyMatMultiDim< MyMatBlockDiag > const &b)
- MyMat operator+ (MyMatBlockDiag const &b, MyMat a)

5.1.1 Typedef Documentation

5.1.1.1 template<int N, typename Q = Gauss<N+1>, typename S = Dubiner<N>> using **Tspeed::FESpace_ptr** = typedef std::shared_ptr<**FESpace**<N,Q,S>>

template pointer to FESpace

Template Parameters

N,Q,S	the same template parameters as FESpace

5.1.1.2 typedef std::shared_ptr<**Force**> **Tspeed::Force_ptr**

5.1.1.3 typedef std::shared_ptr<**Mesh**> **Tspeed::Mesh_ptr**

Shared pointer to an element of type mesh.

5.1.2 Enumeration Type Documentation

5.1.2.1 enum **Tspeed::Bc**

Enumerator

Dirichlet

Neumann

Internal

Unassigned

5.1.3 Function Documentation

5.1.3.1 Eigen::Matrix2d Tspeed::CTensorProduct (Eigen::Matrix2d const & *A*, double *lambda*, double *mu*)

tensor product between Hooke's tensor and matrix A

Parameters

A	the matrix
lambda,mu	Hooke's constants

Returns

5.1.3.2 Eigen::ArrayXd Tspeed::jacobi_polynomial (int *N*, int *alpha*, int *beta*, Eigen::ArrayXd const & *z*)

5.1.3.3 double Tspeed::mat_dot (Eigen::Matrix2d const & *a*, Eigen::Matrix2d const & *b*)

Dot product between two 2x2 matrices.

Parameters

a,b	the matrices $\mathbf{A},\mathbf{B}$

Returns

$\sum_{ij} A_{ij} B_{ij}$

5.1.3.4 Eigen::VectorXd Tspeed::operator∗ (MyMatMultiDimBlockDiag< MyMatBlockDiag > const & *A*, Eigen::VectorXd const & *v*)

5.1.3.5 Eigen::VectorXd Tspeed::operator∗ (MyMatMultiDim< MyMat > & *A*, Eigen::VectorXd const & *v*)

5.1.3.6 Eigen::VectorXd Tspeed::operator∗ (MyMatMultiDim< MyMatBlockDiag > & *A*, Eigen::VectorXd const & *v*)

5.1.3.7 MyMat Tspeed::operator∗ (double const & *c*, MyMat const & *M*)

5.1.3.8 Eigen::VectorXd Tspeed::operator∗ (MyMat const & *A*, Eigen::VectorXd const & *x*)

5.1.3.9 Eigen::VectorXd Tspeed::operator∗ (MyMatBlockDiag const & *A*, Eigen::VectorXd const & *x*)

5.1.3.10 MyMat Tspeed::operator+ (MyMat *a*, MyMat const & *b*)

5.1.3.11 MyMat Tspeed::operator+ (MyMat *a*, MyMatBlockDiag const & *b*)

5.1.3.12 MyMatMultiDim<MyMat> Tspeed::operator+ (MyMatMultiDim< MyMat > const & *a*, MyMatMultiDim< MyMat > const & *b*)

5.1.3.13 MyMatMultiDim<MyMat> Tspeed::operator+ (MyMatMultiDim< MyMat > const & *a*, MyMatMultiDim< MyMatBlockDiag > const & *b*)

5.1.3.14 MyMat Tspeed::operator+ (MyMatBlockDiag const & *b*, MyMat *a*)

5.2 Tspeed::Geo Namespace Reference

Classes

- class Point

  Class describing points.
- class Edge

  Class describing an edge.
- class Triangle

  Class describing a triangle.

Functions

- std::ostream & operator<< (std::ostream &, Triangle const &)
- std::ostream & operator<< (std::ostream &, Point const &)
- Point operator- (const Point &a, const Point &b)
- Point operator- (const Eigen::Vector2d &a, const Point &b)
- Point operator- (const Point &a, const Eigen::Vector2d &b)
- Point operator+ (const Eigen::Vector2d &a, const Point &b)
- Point operator+ (const Point &a, const Eigen::Vector2d &b)
- Point operator+ (const Point &a, const Point &b)
- Point operator∗ (const double &d, const Point &p)

5.2.1 Function Documentation

5.2.1.1 Point Tspeed::Geo::operator∗ (const double & *d*, const Point & *p*)

5.2.1.2 Point Tspeed::Geo::operator+ (const Eigen::Vector2d & *a*, const Point & *b*)

Parameters

a	vector
b	Point

Returns

Point

5.2.1.3 Point Tspeed::Geo::operator+ (const Point & *a*, const Eigen::Vector2d & *b*)

Parameters

a	Point
b	vector

Returns

Point

5.2.1.4 Point Tspeed::Geo::operator+ (const Point & *a*, const Point & *b*)

Parameters

a	first point
b	second point

Returns

sum of points

5.2.1.5 Point Tspeed::Geo::operator- (const Point & *a*, const Point & *b*)

Parameters

a	first point
b	second point

Returns

difference of points

5.2.1.6 Point Tspeed::Geo::operator- (const Eigen::Vector2d & *a*, const Point & *b*)

Parameters

a	first point
b	vector

Returns

point: difference of point a vector

5.2.1.7 Point Tspeed::Geo::operator- (const Point & *a*, const Eigen::Vector2d & *b*)

5.2.1.8 std::ostream & Tspeed::Geo::operator<< (std::ostream & *io*, Triangle const & *t*)

5.2.1.9 std::ostream & Tspeed::Geo::operator<< (std::ostream & *io*, Point const & *p*)

Chapter 6

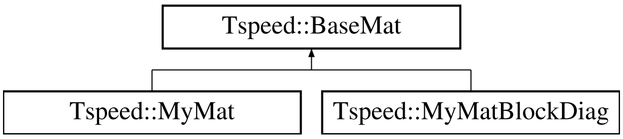
Class Documentation

6.1 Tspeed::BaseMat Class Reference

Base monodimensional matrix class.

`#include <MyMat.hpp>`

Inheritance diagram for Tspeed::BaseMat:



Public Member Functions

- BaseMat ()
- BaseMat (Mesh_ptr, unsigned int nln)
- Eigen::MatrixXd const & block (unsigned int i, unsigned int j) const

  Get sub-block generated by integration on elements i and j ($\neq 0$ if they are neighbors) (const version)
- Eigen::MatrixXd & block (unsigned int i, unsigned int j)

  Get sub-block generated by integration on elements i and j ($\neq 0$ if they are neighbors) (non-const version)
- void setblock (unsigned int i, unsigned int j, Eigen::MatrixXd const &M)

  Set sub-block generated by integration on elements i and j ($\neq 0$ if they are neighbors) (const version)
- BaseMat (BaseMat const &)
- virtual ∼BaseMat ()=default
- unsigned int nr () const

  Get number of rows of the matrix.
- std::vector< unsigned int > const & rowInd () const

  Get row indices of the matrix.
- std::vector< unsigned int > const & colInd () const

  Get column indices of the matrix.
- void set_rowInd (std::vector< unsigned int >const &v)

  Set row indices of the matrix.
- void set_colInd (std::vector< unsigned int >const &v)

  Set column indices of the matrix.
- std::vector< Eigen::MatrixXd >
  const & elem () const

Get a vector with all the sub-blocks (const version)
- std::vector< Eigen::MatrixXd > & elem ()

  Get a vector with all the sub-blocks (non-const version)
- Eigen::MatrixXd const & elem (int i) const

  Get sub-block linearly indexed by i.
- unsigned int size () const

  Get the number of sub-blocks.
- void vecMult (Eigen::VectorXd const &x, Eigen::VectorXd &out) const

  Multiplication by a vector.

Protected Attributes

- unsigned int M_nr
- unsigned int M_nc
- unsigned int M_nln
- std::vector< Eigen::MatrixXd > M_m
- std::vector< unsigned int > M_r
- std::vector< unsigned int > M_c

6.1.1 Detailed Description

Base monodimensional matrix class.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Tspeed::BaseMat::BaseMat ( )

6.1.2.2 Tspeed::BaseMat::BaseMat (Mesh_ptr , unsigned int *nln*)

6.1.2.3 Tspeed::BaseMat::BaseMat (BaseMat const &)

6.1.2.4 virtual Tspeed::BaseMat::∼BaseMat ( ) [virtual], [default]

6.1.3 Member Function Documentation

6.1.3.1 Eigen::MatrixXd const & Tspeed::BaseMat::block (unsigned int *i*, unsigned int *j*) const

Get sub-block generated by integration on elements i and j ($\neq 0$ if they are neighbors) (const version)

Parameters

i,j	the indices of the triangles

Returns

The sub-block

6.1.3.2 Eigen::MatrixXd & Tspeed::BaseMat::block (unsigned int *i*, unsigned int *j*)

Get sub-block generated by integration on elements i and j ($\neq 0$ if they are neighbors) (non-const version)

Parameters

i,j	the indices of the triangles

Returns

The sub-block

6.1.3.3 std::vector< unsigned int > const& Tspeed::BaseMat::colInd ( ) const `[inline]`

Get column indices of the matrix.

6.1.3.4 std::vector< Eigen::MatrixXd > const& Tspeed::BaseMat::elem ( ) const `[inline]`

Get a vector with all the sub-blocks (const version)

6.1.3.5 std::vector< Eigen::MatrixXd >& Tspeed::BaseMat::elem ( ) `[inline]`

Get a vector with all the sub-blocks (non-const version)

6.1.3.6 Eigen::MatrixXd const& Tspeed::BaseMat::elem (int *i*) const `[inline]`

Get sub-block linearly indexed by i.

Parameters

i	the linear index of the sub-block

6.1.3.7 unsigned int Tspeed::BaseMat::nr ( ) const `[inline]`

Get number of rows of the matrix.

Returns

The number of rows

6.1.3.8 std::vector< unsigned int > const& Tspeed::BaseMat::rowInd ( ) const `[inline]`

Get row indices of the matrix.

6.1.3.9 void Tspeed::BaseMat::set_colInd (std::vector< unsigned int >const & *v*) `[inline]`

Set column indices of the matrix.

6.1.3.10 void Tspeed::BaseMat::set_rowInd (std::vector< unsigned int >const & *v*) `[inline]`

Set row indices of the matrix.

6.1.3.11 void Tspeed::BaseMat::setblock (unsigned int *i,* unsigned int *j,* Eigen::MatrixXd const & *M*)

Set sub-block generated by integration on elements i and j ($\neq 0$ if they are neighbors) (const version)

Parameters

i,j	the indices of the triangles
M	the matrix assigned to the sub-block

6.1.3.12 unsigned int Tspeed::BaseMat::size ( ) const [inline]

Get the number of sub-blocks.

6.1.3.13 void Tspeed::BaseMat::vecMult (Eigen::VectorXd const & *x*, Eigen::VectorXd & *out*) const

Multiplication by a vector.

Parameters

in	*x*	term to be multiplied
out	*out*	result of the multiplication

6.1.4 Member Data Documentation

6.1.4.1 std::vector<unsigned int> Tspeed::BaseMat::M_c [protected]

6.1.4.2 std::vector<Eigen::MatrixXd> Tspeed::BaseMat::M_m [protected]

6.1.4.3 unsigned int Tspeed::BaseMat::M_nc [protected]

6.1.4.4 unsigned int Tspeed::BaseMat::M_nln [protected]

6.1.4.5 unsigned int Tspeed::BaseMat::M_nr [protected]

6.1.4.6 std::vector<unsigned int> Tspeed::BaseMat::M_r [protected]

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

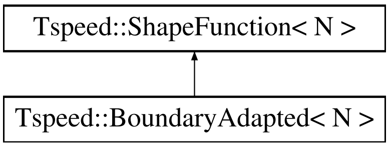
- lib/include/MyMat.hpp
- lib/src/MyMat.cpp

6.2 Tspeed::BoundaryAdapted< N > Class Template Reference

Boundary adapted [2] basis.

`#include <ShapeFunctions.hpp>`

Inheritance diagram for Tspeed::BoundaryAdapted< N >:



Public Types

- enum { is_orthonormal = false }

The basis is not orthonormal.

Public Member Functions

- virtual ~BoundaryAdapted ()
- BoundaryAdapted ()

Additional Inherited Members

6.2.1 Detailed Description

template<int N>class Tspeed::BoundaryAdapted< N >

Boundary adapted [2] basis.

Template Parameters

N	degree of the space $\mathbb{N}$

6.2.2 Member Enumeration Documentation

6.2.2.1 template<int N> anonymous enum

The basis is not orthonormal.

Enumerator

is_orthonormal

6.2.3 Constructor & Destructor Documentation

6.2.3.1 template<int N> virtual Tspeed::BoundaryAdapted< N >::~BoundaryAdapted ( ) [inline], [virtual]

6.2.3.2 template<int N> Tspeed::BoundaryAdapted< N >::BoundaryAdapted ( )

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

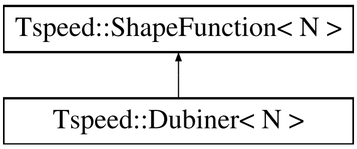
- lib/include/ShapeFunctions.hpp
- lib/include/ShapeFunctions_imp.hpp

6.3 Tspeed::Dubiner< N > Class Template Reference

Dubiner [1] basis.

```
#include <ShapeFunctions.hpp>
```

Inheritance diagram for Tspeed::Dubiner< N >:



Public Types

- enum { is_orthonormal = true }

  Dubiner basis is orthonormal.

Public Member Functions

- virtual ∼Dubiner ()
- Dubiner ()

Additional Inherited Members

6.3.1 Detailed Description

template<int N>class Tspeed::Dubiner< N >

Dubiner [1] basis.

Template Parameters

N	degree of the space $\mathbb{N}$

6.3.2 Member Enumeration Documentation

6.3.2.1 template<int N> anonymous enum

Dubiner basis is orthonormal.

Enumerator

is_orthonormal

6.3.3 Constructor & Destructor Documentation

6.3.3.1 template<int N> virtual Tspeed::Dubiner< N >::∼Dubiner ( ) `[inline], [virtual]`

6.3.3.2 template<int N> Tspeed::Dubiner< N >::Dubiner ( )

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

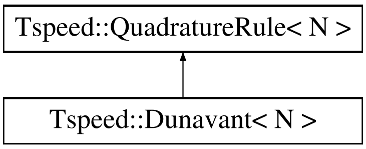
- lib/include/ShapeFunctions.hpp
- lib/include/ShapeFunctions_imp.hpp

6.4 Tspeed::Dunavant< N > Class Template Reference

Dunavant [1] quadrature rule.

```
#include <QuadratureRule.hpp>
```

Inheritance diagram for Tspeed::Dunavant< N >:



Public Types

- enum { nqn2d = dunavant_num_points<N>() }
- enum { nqn1d = N }
- typedef QuadratureRule< N >::Vec Vec
- typedef QuadratureRule< N >::Vec Mat
- typedef QuadratureRule< N >::Vec Vec2

Public Member Functions

- Dunavant ()

Additional Inherited Members

6.4.1 Detailed Description

template<int N>class Tspeed::Dunavant< N >

Dunavant [1] quadrature rule.

Template Parameters

N	the order of the rule.

A rule of order N integrates exactly polynomials of order N

6.4.2 Member Typedef Documentation

6.4.2.1 template<int N> typedef QuadratureRule<N>::Vec Tspeed::Dunavant< N >::Mat

6.4.2.2 template<int N> typedef QuadratureRule<N>::Vec Tspeed::Dunavant< N >::Vec

6.4.2.3 template<int N> typedef QuadratureRule<N>::Vec Tspeed::Dunavant< N >::Vec2

6.4.3 Member Enumeration Documentation

6.4.3.1 template<int N> anonymous enum

Enumerator

nqn2d

6.4.3.2 template<int N> anonymous enum

Enumerator

nqn1d

6.4.4 Constructor & Destructor Documentation

6.4.4.1 template<int N> Tspeed::Dunavant< N >::Dunavant ( )

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

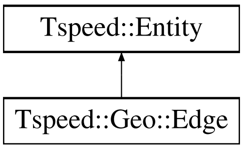
- lib/include/QuadratureRule.hpp
- lib/include/QuadratureRule_imp.hpp

6.5 Tspeed::Geo::Edge Class Reference

Class describing an edge.

`#include <Geometry.hpp>`

Inheritance diagram for Tspeed::Geo::Edge:



Public Member Functions

- Edge ()

  Default constructor: extremal points are initialized to 0.
- Edge (const Point &a, const Point &b)

  Constructor.
- Edge (const Edge &e)

  copy constructor
- virtual ∼Edge ()
- double length () const

  Length of the edge.
- Eigen::Vector2d normal () const

  Normal to the edge.
- Edge & operator= (const Edge &)

  Assignement.

Additional Inherited Members

6.5.1 Detailed Description

Class describing an edge.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 Tspeed::Geo::Edge::Edge ( ) [inline]

Default constructor: extremal points are initialized to 0.

6.5.2.2 Tspeed::Geo::Edge::Edge (const Point & *a*, const Point & *b*) `[inline]`

Constructor.

Parameters

a	One extreme
b	The other extreme

6.5.2.3 Tspeed::Geo::Edge::Edge (const Edge & *e*) `[inline]`

copy constructor

Parameters

e	

6.5.2.4 virtual Tspeed::Geo::Edge::∼Edge ( ) `[inline], [virtual]`

6.5.3 Member Function Documentation

6.5.3.1 double Tspeed::Geo::Edge::length ( ) const `[inline]`

Length of the edge.

6.5.3.2 Eigen::Vector2d Tspeed::Geo::Edge::normal ( ) const

Normal to the edge.

Returns

The normalized vector

6.5.3.3 Edge & Tspeed::Geo::Edge::operator= (const Edge & *e*)

Assignement.

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

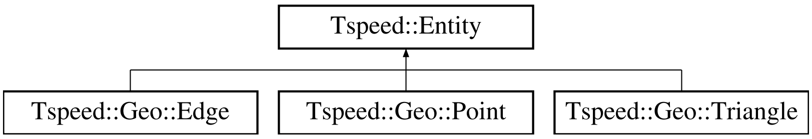
- lib/include/Geometry.hpp
- lib/src/Geometry.cpp

6.6 Tspeed::Entity Class Reference

Base class for geometrical entities.

`#include <Geometry.hpp>`

Inheritance diagram for Tspeed::Entity:



Public Types

- typedef unsigned int Id

Public Member Functions

- Entity ()
- bool unassignedId () const

  Check if element has assigned Id.
- bool unassignedBc () const

  Check if element has assigned boundary condition.
- bool unassignedReg () const

  Check if element has assigned region.
- Id & reg ()

  Get region id.
- Id const & reg () const

  Get region id (const version)
- Id & id ()

  Get element id (const version)
- Id const & id () const

  Get element id (const version)
- Bc & bcId ()

  Get boundary id (const version)
- Bc const & bcId () const

  Get boundary id (const version)

Protected Attributes

- Id M_reg

  Region number.
- Id M_id

  Id of the element.
- Bc M_bcId

  Boundary Id of the element.

6.6.1 Detailed Description

Base class for geometrical entities.

6.6.2 Member Typedef Documentation

6.6.2.1 typedef unsigned int Tspeed::Entity::Id

6.6.3 Constructor & Destructor Documentation

6.6.3.1 Tspeed::Entity::Entity ( ) [inline]

6.6.4 Member Function Documentation

6.6.4.1 Bc& Tspeed::Entity::bcId ( ) [inline]

Get boundary id (const version)

Returns

Constant reference to boundary id

6.6.4.2 Bc const& Tspeed::Entity::bcId ( ) const [inline]

Get boundary id (const version)

Returns

Constant reference to boundary id

6.6.4.3 Id& Tspeed::Entity::id ( ) [inline]

Get element id (const version)

Returns

Constant reference to element id

6.6.4.4 Id const& Tspeed::Entity::id ( ) const [inline]

Get element id (const version)

Returns

Constant reference to element id

6.6.4.5 Id& Tspeed::Entity::reg ( ) [inline]

Get region id.

Returns

Reference to region id

6.6.4.6 Id const& Tspeed::Entity::reg ( ) const [inline]

Get region id (const version)

Returns

Constant reference to region id

6.6.4.7 bool Tspeed::Entity::unassignedBc ( ) const [inline]

Check if element has assigned boundary condition.

Returns

TRUE if Bc is not assigned

6.6.4.8 bool Tspeed::Entity::unassignedId ( ) const [inline]

Check if element has assigned Id.

Returns

TRUE if Id is not assigned

6.6.4.9 bool Tspeed::Entity::unassignedReg ( ) const [inline]

Check if element has assigned region.

Returns

TRUE if region is not assigned

6.6.5 Member Data Documentation

6.6.5.1 Bc Tspeed::Entity::M_bcId [protected]

Boundary Id of the element.

6.6.5.2 Id Tspeed::Entity::M_id [protected]

Id of the element.

6.6.5.3 Id Tspeed::Entity::M_reg [protected]

Region number.

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

- lib/include/Geometry.hpp

6.7 Tspeed::FESpace< N, Q, S > Class Template Reference

Functional space.

`#include <FESpace.hpp>`

Public Member Functions

- FESpace (Mesh_ptr m)

  Constructor from the mesh.
- virtual ∼FESpace ()
- Mesh_ptr mesh () const

  Get pointer ot mesh.
- Q const & quad () const

  Get quadrature rule.
- ShapeFunction< N > const & shape () const

  Get Shapefunction.
- unsigned int nln () const

  Number of degrees of freedom per element.
- unsigned int ne () const

  Number of elements in the mesh.
- Eigen::Vector2d grad (unsigned int k, unsigned int i) const

  Get value of the gradient of the basis function i on quadrature point k.
- Eigen::VectorXd b_edge (unsigned int k, unsigned int iedg) const

  Get value of all basis function on quadrature node k, on edge iedg.
- Eigen::Vector2d g_edge (unsigned int k, unsigned int i, unsigned short int edg) const

  value of the gradient of basis function i, on edge edg, quadrature node k
- Eigen::VectorXd inverse_transform (std::function< std::array< double, 2 >(double, double)> const &fun) const

  Transform a function $u(x,y)$ into its expantion modes $\hat{u}_i$ s.t.

  $$\sum_i \hat{u}_i \psi_i(x,y) = u(x,y)$$

  .
- double L2error (std::function< std::array< double, 2 >(double, double)> const &uex, Eigen::VectorXd const &uh) const

  L2 norm of the difference uex-uh.
- Eigen::VectorXd loc_rhs (Geo::Triangle const &ie, std::function< std::array< double, 2 >(double, double)> const &fun) const

  Integration of a function against all basis function, in triangle ie, i.e.

  $$l_i = \int_{\mathrm{ie}} f \psi_i$$

  .
- void points_out (std::string const &fname) const

  Write a set of points of the mesh to file.
- void field_out (std::string const &fname, Eigen::VectorXd const &uh, unsigned int step) const

  Write a field to file, on the points given by points_out.
- Eigen::MatrixXd base_mass () const

  Evaluate the mass matrix $\mathbf{M}$ s.t.

  $$M_{ij} = \int_{\mathscr{T}^2} \psi_i \psi_j$$

  .
- Eigen::MatrixXd base_invmass () const

  Evaluate the inverse of the mass matrix $\mathbf{M}^{-1}$.

Public Attributes

- EIGEN_MAKE_ALIGNED_OPERATOR_NEW

6.7.1 Detailed Description

template<int N, typename Q = Gauss<N+1>, typename S = Dubiner<N>>class Tspeed::FESpace< N, Q, S >

Functional space.

Template Parameters

N	order of the polynomials
Q	quadrature rule
S	basis functions

6.7.2 Constructor & Destructor Documentation

6.7.2.1 template<int N, typename Q , typename S > Tspeed::FESpace< N, Q, S >::FESpace (Mesh_ptr *m*) [explicit]

Constructor from the mesh.

Parameters

m	shared pointer to Mesh

6.7.2.2 template<int N, typename Q = Gauss<N+1>, typename S = Dubiner<N>> virtual Tspeed::FESpace< N, Q, S >::~FESpace ( ) [inline], [virtual]

6.7.3 Member Function Documentation

6.7.3.1 template<int N, typename Q = Gauss<N+1>, typename S = Dubiner<N>> Eigen::VectorXd Tspeed::FESpace< N, Q, S >::b_edge (unsigned int *k*, unsigned int *iedg*) const [inline]

Get value of all basis function on quadrature node k, on edge iedg.

Parameters

k	index of the quadrature point
iedg	index of the edge (=0,1,2)

Returns

vector of all the values

6.7.3.2 template<int N, typename Q = Gauss<N+1>, typename S = Dubiner<N>> Eigen::MatrixXd Tspeed::FESpace< N, Q, S >::base_invmass ( ) const [inline]

Evaluate the inverse of the mass matrix $\mathbf{M}^{-1}$.

Returns

$\mathbf{M}^{-1}$

6.7.3.3 template<int N, typename Q = Gauss<N+1>, typename S = Dubiner<N>> Eigen::MatrixXd Tspeed::FESpace< N, Q, S >::base_mass ( ) const [inline]

Evaluate the mass matrix $\mathbf{M}$ s.t.

$$M_{ij} = \int_{\mathscr{T}^2} \psi_i \psi_j$$

.

Returns

the matrix $\mathbf{M}$

6.7.3.4 template<int N, typename Q , typename S > void Tspeed::FESpace< N, Q, S >::field_out (std::string const & *fname*, Eigen::VectorXd const & *uh*, unsigned int *step*) const

Write a field to file, on the points given by points_out.

Parameters

fname	name of the output file
uh	vector of the coefficients of a FE function
step	the time step (gets appended to the file name)

6.7.3.5 template<int N, typename Q = Gauss<N+1>, typename S = Dubiner<N>> Eigen::Vector2d Tspeed::FESpace< N, Q, S >::g_edge (unsigned int *k*, unsigned int *i*, unsigned short int *edg*) const [inline]

value of the gradient of basis function i, on edge edg, quadrature node k

Parameters

k	index of the quadrature node
i	indec of the basis function
edg	edg number

Returns

a vector with the x and y derivatives

6.7.3.6 template<int N, typename Q = Gauss<N+1>, typename S = Dubiner<N>> Eigen::Vector2d Tspeed::FESpace< N, Q, S >::grad (unsigned int *k*, unsigned int *i*) const [inline]

Get value of the gradient of the basis function i on quadrature point k.

Parameters

k	index of the quadrature node
i	index of the basis function

Returns

A vector with the x and y derivatives

6.7.3.7 template< int N, typename Q , typename S > Eigen::VectorXd Tspeed::FESpace< N, Q, S >::inverse_transform (std::function< std::array< double, 2 >(double, double)> const & *fun*) const

Transform a function $u(x,y)$ into its expantion modes $\hat{u}_i$ s.t.

$$\sum_i \hat{u}_i \psi_i(x,y) = u(x,y)$$

.

Parameters

fun	the function

Returns

the vector of $\hat{u}_i$

6.7.3.8 template< int N, typename Q , typename S > double Tspeed::FESpace< N, Q, S >::L2error (std::function< std::array< double, 2 >(double, double)> const & *uex,* Eigen::VectorXd const & *uh*) const

L2 norm of the difference uex-uh.

Parameters

uex	A function
uh	A vector of expansion modes

Returns

The norm

6.7.3.9 template< int N, typename Q = Gauss< N+1 >, typename S = Dubiner< N >> Eigen::VectorXd Tspeed::FESpace< N, Q, S >::loc_rhs (Geo::Triangle const & *ie,* std::function< std::array< double, 2 >(double, double)> const & *fun*) const [inline]

Integration of a function against all basis function, in triangle ie, i.e.

$$l_i = \int_{\text{ie}} f \psi_i$$

.

Parameters

ie	the index of the triangle
fun	the function to be integrated

Returns

the vector made by l_i

6.7.3.10 template<int N, typename Q = Gauss<N+1>, typename S = Dubiner<N>> Mesh_ptr Tspeed::FESpace< N, Q, S >::mesh ( ) const [inline]

Get pointer ot mesh.

Returns

pointer to the mesh

6.7.3.11 template<int N, typename Q = Gauss<N+1>, typename S = Dubiner<N>> unsigned int Tspeed::FESpace< N, Q, S >::ne ( ) const [inline]

Number of elements in the mesh.

6.7.3.12 template<int N, typename Q = Gauss<N+1>, typename S = Dubiner<N>> unsigned int Tspeed::FESpace< N, Q, S >::nln ( ) const [inline]

Number of degrees of freedom per element.

6.7.3.13 template<int N, typename Q , typename S > void Tspeed::FESpace< N, Q, S >::points_out (std::string const & *fname*) const

Write a set of points of the mesh to file.

Parameters

fname	name of the outpu file

6.7.3.14 template<int N, typename Q = Gauss<N+1>, typename S = Dubiner<N>> Q const& Tspeed::FESpace< N, Q, S >::quad ( ) const [inline]

Get quadrature rule.

6.7.3.15 template<int N, typename Q = Gauss<N+1>, typename S = Dubiner<N>> ShapeFunction<N> const& Tspeed::FESpace< N, Q, S >::shape ( ) const [inline]

Get Shapefunction.

6.7.4 Member Data Documentation

6.7.4.1 template<int N, typename Q = Gauss<N+1>, typename S = Dubiner<N>> Tspeed::FESpace< N, Q, S >::EIGEN_MAKE_ALIGNED_OPERATOR_NEW

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

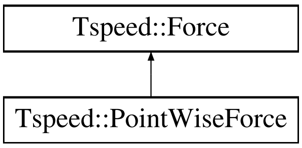
- lib/include/FESpace.hpp
- lib/include/FESpace_imp.hpp

6.8 Tspeed::Force Class Reference

virtual base class for forces

```
#include <Force.hpp>
```

Inheritance diagram for Tspeed::Force:



Public Types

- typedef Eigen::SparseVector < double > SPVec
- typedef Eigen::VectorXd Vec

Public Member Functions

- Force ()
- Force (std::function< std::array< double, 2 >(const double &)> const &fun)

  Constructor, taking the function (time dependent)
- virtual ~Force ()
- virtual Vec eval (const double &) const =0

Protected Attributes

- std::function< std::array < double, 2 >const double &)> M_f

6.8.1 Detailed Description

virtual base class for forces

6.8.2 Member Typedef Documentation

6.8.2.1 typedef Eigen::SparseVector<double> **Tspeed::Force::SPVec**

6.8.2.2 typedef Eigen::VectorXd **Tspeed::Force::Vec**

6.8.3 Constructor & Destructor Documentation

6.8.3.1 Tspeed::Force::Force ( ) `[inline]`

6.8.3.2 Tspeed::Force::Force (std::function< std::array< double, 2 >(const double &)> const & *fun*)

Constructor, taking the function (time dependent)

Parameters

fun	The function

6.8.3.3 virtual Tspeed::Force::∼Force ( ) [inline], [virtual]

6.8.4 Member Function Documentation

6.8.4.1 virtual Vec Tspeed::Force::eval (const double &) const [pure virtual]

Implemented in Tspeed::PointWiseForce.

6.8.5 Member Data Documentation

6.8.5.1 std::function<std::array<double,2>const double &)> Tspeed::Force::M_f [protected]

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

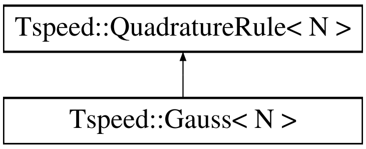
- lib/include/Force.hpp

6.9 Tspeed::Gauss< N > Class Template Reference

Gauss quadrature rule on the triangle.

```
#include <QuadratureRule.hpp>
```

Inheritance diagram for Tspeed::Gauss< N >:



Public Types

- enum { nqn2d = N∗N }
- enum { nqn1d = N }
- typedef QuadratureRule< N >::Vec Vec
- typedef QuadratureRule< N >::Vec Mat
- typedef QuadratureRule< N >::Vec Vec2

Public Member Functions

- Gauss ()

Additional Inherited Members

6.9.1 Detailed Description

template<int N>class Tspeed::Gauss< N >

Gauss quadrature rule on the triangle.

Template Parameters

N	the order of the rule

The internal nodes will be N^2; N+1 is the required order to integrate polynomials of order 2N.

6.9.2 Member Typedef Documentation

6.9.2.1 template<int N> typedef QuadratureRule<N>::Vec Tspeed::Gauss< N >::Mat

6.9.2.2 template<int N> typedef QuadratureRule<N>::Vec Tspeed::Gauss< N >::Vec

6.9.2.3 template<int N> typedef QuadratureRule<N>::Vec Tspeed::Gauss< N >::Vec2

6.9.3 Member Enumeration Documentation

6.9.3.1 template<int N> anonymous enum

Enumerator

nqn2d

6.9.3.2 template<int N> anonymous enum

Enumerator

nqn1d

6.9.4 Constructor & Destructor Documentation

6.9.4.1 template<int N> Tspeed::Gauss< N >::Gauss ( )

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

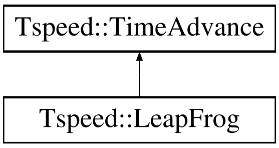
- lib/include/QuadratureRule.hpp
- lib/include/QuadratureRule_imp.hpp

6.10 Tspeed::LeapFrog Class Reference

Implementation of the second order Leap-Frog explicit time stepping scheme.

```
#include <TimeAdvance.hpp>
```

Inheritance diagram for Tspeed::LeapFrog:



Public Member Functions

- template<int N, typename Q , typename S >
  LeapFrog (FESpace_ptr< N, Q, S >, Parameters const &, Receivers const &)
- void first_step ()

  First step for the Leap-Frog method.
- void step (double)

Step at time t for the Leap-Frog method.

Additional Inherited Members

6.10.1 Detailed Description

Implementation of the second order Leap-Frog explicit time stepping scheme.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 template<int N, typename Q , typename S > Tspeed::LeapFrog::LeapFrog (FESpace_ptr< N, Q, S > *Xh*, Parameters const & *p*, Receivers const & *r*)

6.10.3 Member Function Documentation

6.10.3.1 void Tspeed::LeapFrog::first_step ( )

First step for the Leap-Frog method.

6.10.3.2 void Tspeed::LeapFrog::step (double *t*)

Step at time t for the Leap-Frog method.

Parameters

t	time

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

- lib/include/TimeAdvance.hpp
- lib/include/TimeAdvance_imp.hpp
- lib/src/TimeAdvance.cpp

6.11 Tspeed::Matrices Class Reference

The class containing the matrices resulting from the spatial discretization.

```
#include <TimeAdvance.hpp>
```

Public Types

- typedef Eigen::SparseMatrix
  < double > SpMat

Public Member Functions

- template<int N, typename Q , typename S >
  Matrices (FESpace_ptr< N, Q, S >, Parameters const &)

  Constructor taking the space and the parameters.
- MyMatMultiDim< MyMatBlockDiag >
  const & getA ()

Get the stiffness matrix

$$A_{ij} = \sum_K \int_K \sigma(\phi_j) : \varepsilon(\phi_i)$$

.

- MyMatMultiDim< MyMat > const & getS ()

  Get the stability matrix

  $$S_{ij} = \sum_e \int_e [[\phi_j]] : [[\phi_i]]$$

  .

- MyMatMultiDim< MyMat > const & getI ()

  Get the interelement matrix

  $$S_{ij} = \sum_e \int_e \{\{\sigma(\phi_j)\}\} : [[\phi_i]] + \{\{\sigma(\phi_i)\}\} : [[\phi_j]]$$

  .

- MyMatMultiDimBlockDiag
  < MyMatBlockDiag > const & getinvM ()

  Get inverse of the global mass matrix.

- template< int N, typename Q , typename T >
  Matrices (FESpace_ptr< N, Q, T > Xh, Parameters const &P)

6.11.1 Detailed Description

The class containing the matrices resulting from the spatial discretization.

6.11.2 Member Typedef Documentation

6.11.2.1 typedef Eigen::SparseMatrix<double> **Tspeed::Matrices::SpMat**

6.11.3 Constructor & Destructor Documentation

6.11.3.1 template< int N, typename Q , typename S > Tspeed::Matrices::Matrices (**FESpace_ptr< N, Q, S >** , **Parameters const &**)

Constructor taking the space and the parameters.

6.11.3.2 template< int N, typename Q , typename T > Tspeed::Matrices::Matrices (**FESpace_ptr< N, Q, T >** *Xh*, **Parameters const &** *P*)

6.11.4 Member Function Documentation

6.11.4.1 **MyMatMultiDim<MyMatBlockDiag>** const& Tspeed::Matrices::getA ( ) [inline]

Get the stiffness matrix

$$A_{ij} = \sum_K \int_K \sigma(\phi_j) : \varepsilon(\phi_i)$$

.

Returns

$\mathbf{A}$

6.11.4.2 MyMatMultiDim< MyMat > const& Tspeed::Matrices::getI ( ) [inline]

Get the interelement matrix

$$S_{ij} = \sum_e \int_e \{\{\sigma(\phi_j)\}\} : [[\phi_i]] + \{\{\sigma(\phi_i)\}\} : [[\phi_j]]$$

.

Returns

$\mathbf{S}$

6.11.4.3 MyMatMultiDimBlockDiag< MyMatBlockDiag > const& Tspeed::Matrices::getinvM ( ) [inline]

Get inverse of the global mass matrix.

Returns

$\mathbf{M}^{-1}$

6.11.4.4 MyMatMultiDim< MyMat > const& Tspeed::Matrices::getS ( ) [inline]

Get the stability matrix

$$S_{ij} = \sum_e \int_e [[\phi_j]] : [[\phi_i]]$$

.

Returns

$\mathbf{S}$

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

- lib/include/TimeAdvance.hpp
- lib/include/Matrices_imp.hpp

6.12 Tspeed::Mesh Class Reference

```
#include <Mesh.hpp>
```

Public Types

- typedef unsigned int size_type
- typedef std::vector
  < Geo::Triangle,
  Eigen::aligned_allocator
  < Eigen::Vector2d > > AlignedVecT
- typedef std::vector< Geo::Edge,
  Eigen::aligned_allocator
  < Eigen::Vector2d > > AlignedVecE
- typedef std::vector
  < Geo::Point,
  Eigen::aligned_allocator
  < Eigen::Vector2d > > AlignedVecP

Public Member Functions

- Mesh (const std::string fileName)

  Generate mesh from a Gmsh mesh.
- Geo::Triangle const & operator[] (size_t i) const

  Get triangle with index i (const)
- Geo::Triangle & operator[] (size_t i)

  Get triangle with index i (non-const)
- AlignedVecT const & elements () const

  Get all triangles in the mesh (const)
- AlignedVecT & elements ()

  Get all triangles in the mesh (non-const)
- ∼Mesh ()
- void stats () const

  Print stats about the mesh (e.g. number of elements, anisotropy etc.)
- unsigned int ne () const

  Get number of triangles in the mesh.
- void printallNeigh () const

  Print neighbors structure.

Public Attributes

- std::vector< std::pair
  < unsigned int, unsigned int > > M_bed_map

6.12.1 Member Typedef Documentation

6.12.1.1 typedef std::vector<**Geo::Edge**,Eigen::aligned_allocator<Eigen::Vector2d> > **Tspeed::Mesh::AlignedVecE**

6.12.1.2 typedef std::vector<**Geo::Point**,Eigen::aligned_allocator<Eigen::Vector2d> > **Tspeed::Mesh::AlignedVecP**

6.12.1.3 typedef std::vector<**Geo::Triangle**,Eigen::aligned_allocator<Eigen::Vector2d> > **Tspeed::Mesh::AlignedVecT**

6.12.1.4 typedef unsigned int **Tspeed::Mesh::size_type**

6.12.2 Constructor & Destructor Documentation

6.12.2.1 Tspeed::Mesh::Mesh (const std::string *fileName*) `[explicit]`

Generate mesh from a Gmsh mesh.

Parameters

fileName	Name of the .msh file containing the mesh

6.12.2.2 Tspeed::Mesh::∼Mesh ( ) `[inline]`

6.12.3 Member Function Documentation

6.12.3.1 AlignedVecT const& Tspeed::Mesh::elements ( ) const [inline]

Get all triangles in the mesh (const)

Returns

Constant reference to a vector of triangles

6.12.3.2 AlignedVecT& Tspeed::Mesh::elements ( ) [inline]

Get all triangles in the mesh (non-const)

Returns

Reference to a vector of triangles

6.12.3.3 unsigned int Tspeed::Mesh::ne ( ) const [inline]

Get number of triangles in the mesh.

6.12.3.4 Geo::Triangle const& Tspeed::Mesh::operator[] (size_t *i*) const [inline]

Get triangle with index i (const)

Parameters

i	index of the triangle

Returns

Constant reference to triangle

6.12.3.5 Geo::Triangle& Tspeed::Mesh::operator[] (size_t *i*) [inline]

Get triangle with index i (non-const)

Parameters

i	index of the triangle

Returns

Reference to triangle

6.12.3.6 void Tspeed::Mesh::printallNeigh ( ) const [inline]

Print neighbors structure.

6.12.3.7 void Tspeed::Mesh::stats ( ) const

Print stats about the mesh (e.g. number of elements, anisotropy etc.)

6.12.4 Member Data Documentation

6.12.4.1 std::vector<std::pair<unsigned int,unsigned int> > Tspeed::Mesh::M_bed_map

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

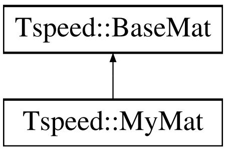
- lib/include/Mesh.hpp
- lib/src/Mesh.cpp

6.13 Tspeed::MyMat Class Reference

Block Matrix (monodimensial blocks of stability and interelement matrices)

```
#include <MyMat.hpp>
```

Inheritance diagram for Tspeed::MyMat:



Public Member Functions

- MyMat ()
- MyMat (Mesh_ptr, unsigned int nln)

  Contructor from the mesh.
- MyMat (MyMat &&)=default
- MyMat (MyMat const &)
- MyMat & operator= (MyMat &&)=default
- MyMat & operator= (MyMat const &)=default
- virtual ∼MyMat () noexcept(true)=default
- void symmetrize ()

  sum with self transposed
- void sumtranspose (MyMat const &ot)

  Sum with ot transposed.
- MyMat operator+= (MyMat const &)
- MyMat operator+= (MyMatBlockDiag const &)
- MyMat operator∗ (double const &) const

Additional Inherited Members

6.13.1 Detailed Description

Block Matrix (monodimensial blocks of stability and interelement matrices)

6.13.2 Constructor & Destructor Documentation

6.13.2.1 Tspeed::MyMat::MyMat ( ) [inline]

6.13.2.2 Tspeed::MyMat::MyMat (Mesh_ptr *Th,* unsigned int *nln*)

Contructor from the mesh.

Parameters

Th	pointer to the mesh
nln	number of degrees of freedom per element

6.13.2.3 Tspeed::MyMat::MyMat (**MyMat** &&) [default]

6.13.2.4 Tspeed::MyMat::MyMat (**MyMat** const & *m*)

6.13.2.5 virtual Tspeed::MyMat::∼MyMat ( ) [virtual], [default], [noexcept]

6.13.3 Member Function Documentation

6.13.3.1 **MyMat** Tspeed::MyMat::operator∗ (double const & *c*) const

6.13.3.2 **MyMat** Tspeed::MyMat::operator+= (**MyMat** const & *a*)

6.13.3.3 **MyMat** Tspeed::MyMat::operator+= (**MyMatBlockDiag** const & *a*)

6.13.3.4 **MyMat**& Tspeed::MyMat::operator= (**MyMat** &&) [default]

6.13.3.5 **MyMat**& Tspeed::MyMat::operator= (**MyMat** const &) [default]

6.13.3.6 void Tspeed::MyMat::sumtranspose (**MyMat** const & *ot*)

Sum with ot transposed.

Parameters

ot	The const matrix to be transposed and summed to current

6.13.3.7 void Tspeed::MyMat::symmetrize ( )

sum with self transposed

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

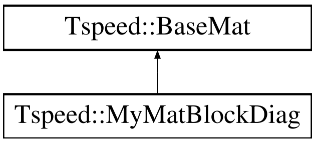
- lib/include/MyMat.hpp
- lib/src/MyMat.cpp

6.14 Tspeed::MyMatBlockDiag Class Reference

Block diagonal monodimensional matrix (monodimensional block of mass and stress-strain matrices)

`#include <MyMat.hpp>`

Inheritance diagram for Tspeed::MyMatBlockDiag:



Public Member Functions

- MyMatBlockDiag ()
- MyMatBlockDiag (Mesh_ptr Th, unsigned int nln)

  Contructor from the mesh.
- MyMatBlockDiag (MyMatBlockDiag &&)=default
- MyMatBlockDiag (MyMatBlockDiag const &)=default
- MyMatBlockDiag & operator= (MyMatBlockDiag &&)=default
- MyMatBlockDiag & operator= (MyMatBlockDiag const &)=default
- virtual ∼MyMatBlockDiag () noexcept(true)=default

Additional Inherited Members

6.14.1 Detailed Description

Block diagonal monodimensional matrix (monodimensional block of mass and stress-strain matrices)

6.14.2 Constructor & Destructor Documentation

6.14.2.1 Tspeed::MyMatBlockDiag::MyMatBlockDiag ( ) `[inline]`

6.14.2.2 Tspeed::MyMatBlockDiag::MyMatBlockDiag (**Mesh_ptr** *Th,* unsigned int *nln*)

Contructor from the mesh.

Parameters

Th	pointer to the mesh
nln	number of degrees of freedom per element

6.14.2.3 Tspeed::MyMatBlockDiag::MyMatBlockDiag (**MyMatBlockDiag** &&) `[default]`

6.14.2.4 Tspeed::MyMatBlockDiag::MyMatBlockDiag (**MyMatBlockDiag** const &) `[default]`

6.14.2.5 virtual Tspeed::MyMatBlockDiag::∼MyMatBlockDiag ( ) `[virtual], [default], [noexcept]`

6.14.3 Member Function Documentation

6.14.3.1 **MyMatBlockDiag**& Tspeed::MyMatBlockDiag::operator= (**MyMatBlockDiag** &&) `[default]`

6.14.3.2 **MyMatBlockDiag**& Tspeed::MyMatBlockDiag::operator= (**MyMatBlockDiag** const &) `[default]`

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

- lib/include/MyMat.hpp
- lib/src/MyMat.cpp

6.15 Tspeed::MyMatMultiDim< T > Class Template Reference

Multidimensional matrix.

```
#include <MyMat.hpp>
```

Public Member Functions

- MyMatMultiDim ()=default
- virtual ∼MyMatMultiDim ()=default
- MyMatMultiDim (Mesh_ptr, unsigned int nln)
- T & component (int i, int j)

  Return monodimensimenial block (i,j)
- T const & component (int i, int j) const

  Return monodimensimenial block (i,j) (const version)
- void symmetrize ()

  Sum matrix to self transposed.
- void vecMult (Eigen::VectorXd const &x, Eigen::VectorXd &out) const

  Multiplication by a vector.
- MyMatMultiDim (MyMatMultiDim &a)
- MyMatMultiDim & operator= (MyMatMultiDim &&)=default
- Eigen::VectorXd operator∗ (Eigen::VectorXd const &v) const

Friends

- MyMatMultiDim< MyMat > operator+ (MyMatMultiDim< MyMat > const &a, MyMatMultiDim< MyMat > const &b)
- MyMatMultiDim< MyMat > operator+ (MyMatMultiDim< MyMat > const &a, MyMatMultiDim< MyMatBlockDiag > const &b)
- MyMatMultiDim< T > operator∗ (double const &x, MyMatMultiDim< T > const &A)

6.15.1 Detailed Description

template<typename T>class Tspeed::MyMatMultiDim< T >

Multidimensional matrix.

Template Parameters

T	type of the monodimensional blocks

6.15.2 Constructor & Destructor Documentation

6.15.2.1 template<typename T> Tspeed::MyMatMultiDim< T >::MyMatMultiDim ( ) [default]

6.15.2.2 template<typename T> virtual Tspeed::MyMatMultiDim< T >::∼MyMatMultiDim ( ) [virtual], [default]

6.15.2.3 template<typename T > Tspeed::MyMatMultiDim< T >::MyMatMultiDim (Mesh_ptr *m*, unsigned int *nln*)

6.15.2.4 template<typename T> Tspeed::MyMatMultiDim< T >::MyMatMultiDim (MyMatMultiDim< T > & *a*) [inline]

6.15.3 Member Function Documentation

6.15.3.1 template<typename T> T& Tspeed::MyMatMultiDim< T >::component (int *i*, int *j*) [inline]

Return monodimensimenial block (i,j)

Parameters

i	"row" index
j	"column" index

Returns

A monodimensional matrix of type

6.15.3.2 template<typename T> T const& Tspeed::MyMatMultiDim< T >::component (int *i*, int *j*) const [inline]

Return monodimensimenial block (i,j) (const version)

Parameters

i	"row" index
j	"column" index

Returns

A monodimensional matrix of type

6.15.3.3 template<typename T > Eigen::VectorXd Tspeed::MyMatMultiDim< T >::operator* (Eigen::VectorXd const & *v*) const

6.15.3.4 template<typename T> MyMatMultiDim& Tspeed::MyMatMultiDim< T >::operator= (MyMatMultiDim< T > &&) [default]

6.15.3.5 template<typename T > void Tspeed::MyMatMultiDim< T >::symmetrize ( )

Sum matrix to self transposed.

6.15.3.6 template<typename T > void Tspeed::MyMatMultiDim< T >::vecMult (Eigen::VectorXd const & *x*, Eigen::VectorXd & *out*) const

Multiplication by a vector.

Parameters

`in`	*x*	term to be multiplied
`out`	*out*	result of the multiplication

6.15.4 Friends And Related Function Documentation

6.15.4.1 template<typename T> MyMatMultiDim<T> operator* (double const & *x*, MyMatMultiDim< T > const & *A*) [friend]

6.15.4.2 template<typename T> MyMatMultiDim<MyMat> operator+ (MyMatMultiDim< MyMat > const & *a*, MyMatMultiDim< MyMat > const & *b*) [friend]

6.15.4.3 template<typename T> MyMatMultiDim<MyMat> operator+ (MyMatMultiDim< MyMat > const & *a*, MyMatMultiDim< MyMatBlockDiag > const & *b*) [friend]

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

- lib/include/MyMat.hpp

6.16 Tspeed::MyMatMultiDimBlockDiag< T > Class Template Reference

Matrix class for matrices where only the diagonal monodimensional matrices are non zero (i.e. the mass matrix)

```
#include <MyMat.hpp>
```

Public Member Functions

- MyMatMultiDimBlockDiag ()=default
- virtual ∼MyMatMultiDimBlockDiag ()=default
- MyMatMultiDimBlockDiag (Mesh_ptr, unsigned int nln)
- T & component (int i)

  Get monodimensional component (0,0) if i=0 or (1,1) if i=1:
- T const & component (int i) const

  Get monodimensional component (0,0) if i=0 or (1,1) if i=1 (const version)
- void vecMult (Eigen::VectorXd const &x, Eigen::VectorXd &out) const

  Multiplication by a vector.
- MyMatMultiDimBlockDiag (MyMatMultiDimBlockDiag &&)=default
- MyMatMultiDimBlockDiag & operator= (MyMatMultiDimBlockDiag &&)=default
- unsigned int nr () const

  Get total number of rows.
- Eigen::VectorXd operator∗ (Eigen::VectorXd const &v) const

Friends

- MyMatMultiDimBlockDiag const & operator∗ (double const &x, MyMatMultiDimBlockDiag const &A)

6.16.1 Detailed Description

template<typename T>class Tspeed::MyMatMultiDimBlockDiag< T >

Matrix class for matrices where only the diagonal monodimensional matrices are non zero (i.e. the mass matrix)

Template Parameters

T	

6.16.2 Constructor & Destructor Documentation

6.16.2.1 template<typename T> Tspeed::MyMatMultiDimBlockDiag< T >::MyMatMultiDimBlockDiag ( ) [default]

6.16.2.2 template<typename T> virtual Tspeed::MyMatMultiDimBlockDiag< T >::∼MyMatMultiDimBlockDiag ( ) [virtual], [default]

6.16.2.3 template<typename T > Tspeed::MyMatMultiDimBlockDiag< T >::MyMatMultiDimBlockDiag (Mesh_ptr *m*, unsigned int *nln*)

6.16.2.4 template<typename T> Tspeed::MyMatMultiDimBlockDiag< T >::MyMatMultiDimBlockDiag (MyMatMultiDimBlockDiag< T > &&) [default]

6.16.3 Member Function Documentation

6.16.3.1 template<typename T> T& Tspeed::MyMatMultiDimBlockDiag< T >::component (int *i*) [inline]

Get monodimensional component (0,0) if i=0 or (1,1) if i=1:

Parameters

i	Select component

Returns

Monodimensional matrix of type T

6.16.3.2 template<typename T> T const& Tspeed::MyMatMultiDimBlockDiag< T >::component (int *i*) const [inline]

Get monodimensional component (0,0) if i=0 or (1,1) if i=1 (const version)

Parameters

i	Select component

Returns

Monodimensional matrix of type T

6.16.3.3 template<typename T> unsigned int Tspeed::MyMatMultiDimBlockDiag< T >::nr ( ) const [inline]

Get total number of rows.

6.16.3.4 template<typename T > Eigen::VectorXd Tspeed::MyMatMultiDimBlockDiag< T >::operator∗ (Eigen::VectorXd const & *v*) const

6.16.3.5 template<typename T> MyMatMultiDimBlockDiag& Tspeed::MyMatMultiDimBlockDiag< T >::operator= (MyMatMultiDimBlockDiag< T > &&) [default]

6.16.3.6 template<typename T > void Tspeed::MyMatMultiDimBlockDiag< T >::vecMult (Eigen::VectorXd const & *x*, Eigen::VectorXd & *out*) const

Multiplication by a vector.

Parameters

in	*x*	term to be multiplied
out	*out*	result of the multiplication

6.16.4 Friends And Related Function Documentation

6.16.4.1 template<typename T> MyMatMultiDimBlockDiag const& operator∗ (double const & *x*, MyMatMultiDimBlockDiag< T > const & *A*) [friend]

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

- lib/include/MyMat.hpp

6.17 Tspeed::Parameters Class Reference

Class for the parameters λ,ρ,μ of the elastodynamics equations.

```
#include <FESpace.hpp>
```

Public Member Functions

- virtual ∼Parameters ()
- Parameters (Mesh_ptr m)
- void setp (std::string const &p, int const lab, double const lambda)

  Set a parameter.
- double const & lambda (int i) const

  Get lambda on element i.
- double const & mu (int i) const

  Get mu on element i.
- double const & rho (int i) const

  Get rho on element i.
- double avg_p (std::string const &p, int i, int j) const

  Get the value of the harmonic average of a parameter between two elements.

6.17.1 Detailed Description

Class for the parameters λ,ρ,μ of the elastodynamics equations.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 virtual Tspeed::Parameters::∼Parameters ( ) [inline], [virtual]

6.17.2.2 Tspeed::Parameters::Parameters (Mesh_ptr *m*) [inline]

6.17.3 Member Function Documentation

6.17.3.1 double Tspeed::Parameters::avg_p (std::string const & *p*, int *i*, int *j*) const

Get the value of the harmonic average of a parameter between two elements.

Parameters

p	string with the name of the parameter ("lambda", "mu", "rho")
i	index of one element
j	index of the seocnd element

Returns

harmonic average

6.17.3.2 double const& Tspeed::Parameters::lambda (int *i*) const `[inline]`

Get lambda on element i.

Parameters

i	the index of the element

6.17.3.3 double const& Tspeed::Parameters::mu (int *i*) const `[inline]`

Get mu on element i.

Parameters

i	the index of the element

6.17.3.4 double const& Tspeed::Parameters::rho (int *i*) const `[inline]`

Get rho on element i.

Parameters

i	the index of the element

6.17.3.5 void Tspeed::Parameters::setp (std::string const & *p,* int const *lab,* double const *lambda*)

Set a parameter.

Parameters

p	string with the name of the parameter ("lambda", "mu", "rho")
lab	attribute of the mesh partition on which the parameter is set
lambda	value of the parameter

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

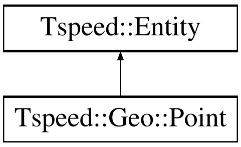
- lib/include/FESpace.hpp
- lib/src/Parameters.cpp

6.18 Tspeed::Geo::Point Class Reference

Class describing points.

```
#include <Geometry.hpp>
```

Inheritance diagram for Tspeed::Geo::Point:



Public Member Functions

- Point (const double &x=0, const double &y=0)

  Constructor taking the two coordinates of the point. By default, a point is initialized to the origin.
- Point (const Point &p)

  Copy constructor. Everything is copied.
- Point (const Eigen::Vector2d &v)

  Constructor taking an Eigen fixed size vector. The components are copied, ids are not assigned.
- virtual ∼Point ()
- double x () const

  first coordinate
- double y () const

  second coordinate
- double & x ()

  Reference to first coordinate.
- double & y ()

  Reference to second coordinate.
- Point & operator= (const Point &)

  assignemnt operator
- Point operator∗ (const double &a) const

  Multuply a point by a scalar.
- double norm () const

  Norm of the vector from the origin to the point.
- Eigen::Vector2d toEig () const

  convert the point into an Eigen::vector. Useful for matrix tranformations

Friends

- Point operator+ (const Point &a, const Point &b)

  Operator summing two points.
- Point operator+ (const Eigen::Vector2d &a, const Point &b)

  Sum a point and en eigen vector.
- Point operator+ (const Point &a, const Eigen::Vector2d &b)

  Sum a point and en eigen vector.
- Point operator- (const Point &a, const Point &b)

  Operator sutracting two points.
- Point operator- (const Eigen::Vector2d &a, const Point &b)

  Operator sutracting a point and a vector.
- Point operator- (const Point &, const Eigen::Vector2d &)
- Point operator∗ (const double &, const Point &)
- double dot (const Point &a, const Point &b)

  vector-style dot product between points

Additional Inherited Members

6.18.1 Detailed Description

Class describing points.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 Tspeed::Geo::Point::Point (const double & *x* = 0, const double & *y* = 0) [inline]

Constructor taking the two coordinates of the point. By default, a point is initialized to the origin.

Parameters

x	x-coordinate
y	y-coordinate

6.18.2.2 Tspeed::Geo::Point::Point (const Point & *p*) [inline]

Copy constructor. Everything is copied.

Parameters

p	A point

6.18.2.3 Tspeed::Geo::Point::Point (const Eigen::Vector2d & *v*) [inline]

Constructor taking an Eigen fixed size vector. The components are copied, ids are not assigned.

Parameters

v	

6.18.2.4 virtual Tspeed::Geo::Point::∼Point ( ) [inline], [virtual]

6.18.3 Member Function Documentation

6.18.3.1 double Tspeed::Geo::Point::norm ( ) const [inline]

Norm of the vector from the origin to the point.

Returns

the euler norm

6.18.3.2 Point Tspeed::Geo::Point::operator∗ (const double & *a*) const

Multuply a point by a scalar.

Parameters

a	scalar

Returns

a both with both coordinates multiplied

6.18.3.3 Point & Tspeed::Geo::Point::operator= (const Point & *p*)

assignemnt operator

6.18.3.4 Eigen::Vector2d Tspeed::Geo::Point::toEig ( ) const [inline]

convert the point into an Eigen::vector. Useful for matrix tranformations

Returns

the eigen vector with the coordinates as components

6.18.3.5 double Tspeed::Geo::Point::x ( ) const [inline]

first coordinate

Returns

x-coordinate

6.18.3.6 double& Tspeed::Geo::Point::x ( ) [inline]

Reference to first coordinate.

Returns

Reference to x-coord

6.18.3.7 double Tspeed::Geo::Point::y ( ) const [inline]

second coordinate

Returns

y-coordinate

6.18.3.8 double& Tspeed::Geo::Point::y ( ) [inline]

Reference to second coordinate.

Returns

Reference to y-coord

6.18.4 Friends And Related Function Documentation

6.18.4.1 double dot (const **Point** & *a*, const **Point** & *b*) [friend]

vector-style dot product between points

Parameters

a	first point
b	second point

Returns

a scalar

6.18.4.2 **Point** operator∗ (const double & *d*, const **Point** & *p*) [friend]

6.18.4.3 **Point** operator+ (const **Point** & *a*, const **Point** & *b*) [friend]

Operator summing two points.

Parameters

a	first point
b	second point

Returns

sum of points

6.18.4.4 **Point** operator+ (const Eigen::Vector2d & *a*, const **Point** & *b*) [friend]

Sum a point and en eigen vector.

Parameters

a	vector
b	Point

Returns

Point

6.18.4.5 **Point** operator+ (const **Point** & *a*, const Eigen::Vector2d & *b*) [friend]

Sum a point and en eigen vector.

Parameters

a	Point
b	vector

Returns

Point

6.18.4.6 Point operator- (const Point & *a*, const Point & *b*) [friend]

Operator sutracting two points.

Parameters

a	first point
b	second point

Returns

difference of points

6.18.4.7 Point operator- (const Eigen::Vector2d & *a*, const Point & *b*) [friend]

Operator sutracting a point and a vector.

Parameters

a	first point
b	vector

Returns

point: difference of point a vector

6.18.4.8 Point operator- (const Point & *a*, const Eigen::Vector2d & *b*) [friend]

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

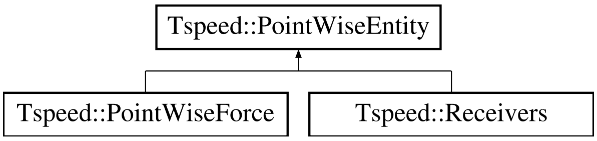
- lib/include/Geometry.hpp
- lib/src/Geometry.cpp

6.19 Tspeed::PointWiseEntity Class Reference

A base class for pointwise entities, with the points and the basis function in that points.

`#include <Receivers.hpp>`

Inheritance diagram for Tspeed::PointWiseEntity:



Public Member Functions

- virtual ~PointWiseEntity ()
- Eigen::ArrayXd const & shape (int i) const

  All shape functions at point i.
- Geo::Point const & point (int i) const

  Point i, with coordinates in the reference triangle.
- unsigned int const & elem (int i) const

  The index of the element where point i resides.
- unsigned int size () const

  The number of points.

Protected Member Functions

- template< int N, typename Q , typename S >
  void M_add (FESpace_ptr< N, Q, S >, Geo::Point const &)

Protected Attributes

- std::vector< unsigned int > M_ie
- std::vector< Geo::Point > M_relp
- std::vector< Eigen::ArrayXd > M_shape
- unsigned int M_nel

6.19.1 Detailed Description

A base class for pointwise entities, with the points and the basis function in that points.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 virtual Tspeed::PointWiseEntity::~PointWiseEntity ( ) [inline], [virtual]

6.19.3 Member Function Documentation

6.19.3.1 unsigned int const& Tspeed::PointWiseEntity::elem (int *i*) const [inline]

The index of the element where point i resides.

Parameters

i	the index of the point

Returns

the index of the triangle

6.19.3.2 template< int N, typename Q , typename S > void Tspeed::PointWiseEntity::M_add (FESpace_ptr< N, Q, S > *Xh*, Geo::Point const & *p*) [protected]

6.19.3.3 Geo::Point const& Tspeed::PointWiseEntity::point (int *i*) const [inline]

Point i, with coordinates in the reference triangle.

Parameters

i	the index of the point

Returns

The point

6.19.3.4 Eigen::ArrayXd const& Tspeed::PointWiseEntity::shape (int *i*) const `[inline]`

All shape functions at point i.

Parameters

i	the index of the point

Returns

an array of all functions

6.19.3.5 unsigned int Tspeed::PointWiseEntity::size ( ) const `[inline]`

The number of points.

6.19.4 Member Data Documentation

6.19.4.1 std::vector<unsigned int> Tspeed::PointWiseEntity::M_ie `[protected]`

6.19.4.2 unsigned int Tspeed::PointWiseEntity::M_nel `[protected]`

6.19.4.3 std::vector<Geo::Point> Tspeed::PointWiseEntity::M_relp `[protected]`

6.19.4.4 std::vector<Eigen::ArrayXd> Tspeed::PointWiseEntity::M_shape `[protected]`

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

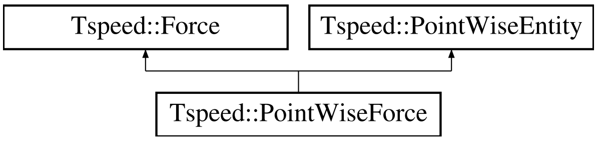
- lib/include/Receivers.hpp
- lib/include/Receivers_imp.hpp

6.20 Tspeed::PointWiseForce Class Reference

Time dependent force acting on a point.

`#include <Force.hpp>`

Inheritance diagram for Tspeed::PointWiseForce:



Public Member Functions

- template<int N, typename Q , typename S >
  PointWiseForce (std::function< std::array< double, 2 >(const double &)> const &f, Geo::Point p, FESpace_ptr< N, Q, S > Xh)

  Costructor taking the function, the point where the force is applied and the function space.
- virtual ∼PointWiseForce ()
- Vec eval (const double &t) const

  Get value of force vector, i.e.

  $$r_i = \int_K f \psi_i$$

  at time t, where f is non null in K.

Additional Inherited Members

6.20.1 Detailed Description

Time dependent force acting on a point.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 template<int N, typename Q , typename S > Tspeed::PointWiseForce::PointWiseForce (std::function< std::array< double, 2 >(const double &)> const & *f*, Geo::Point *p*, FESpace_ptr< N, Q, S > *Xh*)

Costructor taking the function, the point where the force is applied and the function space.

Template Parameters

N,Q,S	the template parameters of the function space

Parameters

f	the force
p	the point
Xh	the space

6.20.2.2 virtual Tspeed::PointWiseForce::∼PointWiseForce ( ) [inline], [virtual]

6.20.3 Member Function Documentation

6.20.3.1 Eigen::VectorXd Tspeed::PointWiseForce::eval (const double & *t*) const [virtual]

Get value of force vector, i.e.

$$r_i = \int_K f \psi_i$$

at time t, where f is non null in K.

Parameters

t	the time

Returns

a vector with elements r_i

Implements Tspeed::Force.

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

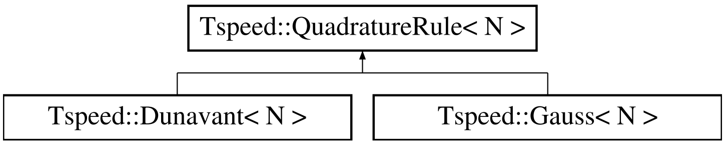
- lib/include/Force.hpp
- lib/include/Force_imp.hpp
- lib/src/Force.cpp

6.21 Tspeed::QuadratureRule< N > Class Template Reference

Base class for quadrature rules.

```
#include <QuadratureRule.hpp>
```

Inheritance diagram for Tspeed::QuadratureRule< N >:



Public Member Functions

- QuadratureRule ()=default
- Vec edge_weights () const

    Get weights on the edge.
- Vec2 int_weights () const

    Get internal weights.
- Geo::Point inode (unsigned int i) const

    Get i-th qudrature node.
- double eweight (unsigned int i) const

    Get i-th weight on the edge.
- double iweight (unsigned int i) const

    Get i-th interior weight.
- size_t edge_n () const

    Number of nodes or weights on the edge.
- size_t int_n () const

    Number of interior nodes or weights.
- Mat int_nodes () const

    Get all interior nodes.
- Eigen::Matrix< double, N, 2 > edge_nodes (int i) const

    Get all nodes on edge i of the reference triangle.
- virtual ~QuadratureRule ()
- unsigned int nqn2d () const

    Number of internal nodes/weights.

Public Attributes

- EIGEN_MAKE_ALIGNED_OPERATOR_NEW

  Required by Eigen.

Protected Types

- typedef Eigen::Matrix< double, Eigen::Dynamic, 1 > Vec2
- typedef Eigen::Matrix< double, Eigen::Dynamic, 2 > Mat
- typedef Eigen::Matrix< double, N, 1 > Vec

Protected Attributes

- Vec2 M_w_2D
- Mat M_node_2D
- Vec M_node_1D
- Vec M_w_1D
- size_t M_nqn_1D
- size_t M_nqn_2D

6.21.1 Detailed Description

template<int N>class Tspeed::QuadratureRule< N >

Base class for quadrature rules.

Template Parameters

N	order of the rule

6.21.2 Member Typedef Documentation

6.21.2.1 template<int N> typedef Eigen::Matrix<double, Eigen::Dynamic, 2> Tspeed::QuadratureRule< N >::Mat [protected]

6.21.2.2 template<int N> typedef Eigen::Matrix<double, N, 1> Tspeed::QuadratureRule< N >::Vec [protected]

6.21.2.3 template<int N> typedef Eigen::Matrix<double, Eigen::Dynamic, 1> Tspeed::QuadratureRule< N >::Vec2 [protected]

6.21.3 Constructor & Destructor Documentation

6.21.3.1 template<int N> Tspeed::QuadratureRule< N >::QuadratureRule ( ) [default]

6.21.3.2 template<int N> virtual Tspeed::QuadratureRule< N >::~QuadratureRule ( ) [inline], [virtual]

6.21.4 Member Function Documentation

6.21.4.1 template<int N> size_t Tspeed::QuadratureRule< N >::edge_n ( ) const [inline]

Number of nodes or weights on the edge.

6.21.4.2 template<int N> Eigen::Matrix< double, N, 2 > Tspeed::QuadratureRule< N >::edge_nodes (int *i*) const [inline]

Get all nodes on edge i of the reference triangle.

Note that ede nodes always have a Gauss-Legendre distribution, whatever rule is used in the interior

Parameters

i	The edge of the reference triangle

Returns

A matrix of size number of nodes x 2

6.21.4.3 template<int N> Vec Tspeed::QuadratureRule< N >::edge_weights ( ) const [inline]

Get weights on the edge.

Returns

The vector of the weights

6.21.4.4 template<int N> double Tspeed::QuadratureRule< N >::eweight (unsigned int *i*) const [inline]

Get i-th weight on the edge.

Parameters

i	the index of the weight

Returns

the weight

6.21.4.5 template<int N> Geo::Point Tspeed::QuadratureRule< N >::inode (unsigned int *i*) const [inline]

Get i-th qudrature node.

Parameters

i	The index of the node

Returns

A Point

6.21.4.6 template<int N> size_t Tspeed::QuadratureRule< N >::int_n ( ) const [inline]

Number of interior nodes or weights.

6.21.4.7 template<int N> Mat Tspeed::QuadratureRule< N >::int_nodes ( ) const [inline]

Get all interior nodes.

Returns

A matrix of size number of nodes x 2

6.21.4.8 template<int N> Vec2 Tspeed::QuadratureRule< N >::int_weights ( ) const [inline]

Get internal weights.

Returns

The vector of the weights

6.21.4.9 template<int N> double Tspeed::QuadratureRule< N >::iweight (unsigned int *i*) const [inline]

Get i-th interior weight.

Parameters

i	the index of the weight

Returns

the weight

6.21.4.10 template<int N> unsigned int Tspeed::QuadratureRule< N >::nqn2d ( ) const [inline]

Number of internal nodes/weights.

6.21.5 Member Data Documentation

6.21.5.1 template<int N> Tspeed::QuadratureRule< N >::EIGEN_MAKE_ALIGNED_OPERATOR_NEW

Required by Eigen.

6.21.5.2 template<int N> Vec Tspeed::QuadratureRule< N >::M_node_1D [protected]

6.21.5.3 template<int N> Mat Tspeed::QuadratureRule< N >::M_node_2D [protected]

6.21.5.4 template<int N> size_t Tspeed::QuadratureRule< N >::M_nqn_1D [protected]

6.21.5.5 template<int N> size_t Tspeed::QuadratureRule< N >::M_nqn_2D [protected]

6.21.5.6 template<int N> Vec Tspeed::QuadratureRule< N >::M_w_1D [protected]

6.21.5.7 template<int N> Vec2 Tspeed::QuadratureRule< N >::M_w_2D [protected]

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

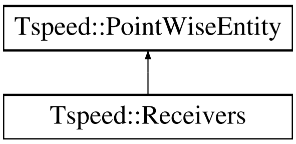
- lib/include/QuadratureRule.hpp
- lib/include/QuadratureRule_imp.hpp

6.22 Tspeed::Receivers Class Reference

A class for seismic receivers, i.e., receivers recording the movement at a point.

`#include <Receivers.hpp>`

Inheritance diagram for Tspeed::Receivers:



Public Member Functions

- template< int N, typename Q , typename S >
  Receivers (FESpace_ptr< N, Q, S > Xh, std::string const &fname)

  Constructor taking the function space and a file with the coordinates of the receivers listed (x-coord and y-coord on every row)
- template< int N, typename Q , typename S >
  Receivers (FESpace_ptr< N, Q, S > Xh, Geo::Point const &p)

  Constructor taking the function space and a point.
- void add (double const &x, double const &y, unsigned int const &ir, unsigned int const &step)

  Add the the value (x,y) of receiver ir at time step step.
- void write (std::string const &fn) const

  Write all recorded values to file.

Additional Inherited Members

6.22.1 Detailed Description

A class for seismic receivers, i.e., receivers recording the movement at a point.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 template< int N, typename Q , typename S > Tspeed::Receivers::Receivers (FESpace_ptr< N, Q, S > *Xh*, std::string const & *fname*)

Constructor taking the function space and a file with the coordinates of the receivers listed (x-coord and y-coord on every row)

Template Parameters

N,Q,S	the template parameters of the function space

Parameters

Xh	the space
fname	the name of the file with the receivers

6.22.2.2 template<int N, typename Q , typename S > Tspeed::Receivers::Receivers (FESpace_ptr< N, Q, S > *Xh*, Geo::Point const & *p*)

Constructor taking the function space and a point.

Template Parameters

N,Q,S	the template parameters of the function space

Parameters

Xh	the space
p	the point where the receiver is

6.22.3 Member Function Documentation

6.22.3.1 void Tspeed::Receivers::add (double const & *x*, double const & *y*, unsigned int const & *ir*, unsigned int const & *step*)

Add the the value (x,y) of receiver ir at time step step.

Parameters

x	the x displacement recorded
y	the y displacement recorded
ir	the index of the receiver
step	the time step of the simulation

6.22.3.2 void Tspeed::Receivers::write (std::string const & *fn*) const

Write all recorded values to file.

Parameters

fn	the name of the output file

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

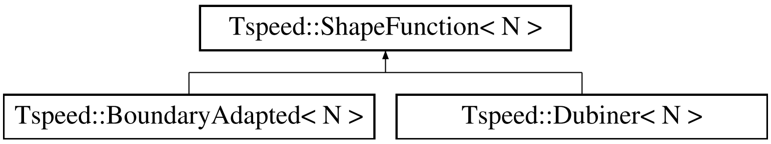
- lib/include/Receivers.hpp
- lib/include/Receivers_imp.hpp
- lib/src/Receivers.cpp

6.23 Tspeed::ShapeFunction< N > Class Template Reference

Base class for the shared functions.

```
#include <ShapeFunctions.hpp>
```

Inheritance diagram for Tspeed::ShapeFunction< N >:



Public Types

- enum { gdl = (N+1)∗(N+2)/2 }

  Number of degrees of freedom.
- enum { is_orthonormal = false }

  Orthonormality of the basis.

Public Member Functions

- virtual ∼ShapeFunction ()
- ShapeFunction ()
- Eigen::ArrayXd phi (unsigned int s, Arr const &v, Arr const &w) const

  Get value of base function with index s, on points (v,w)
- double phi (unsigned int s, double x, double y) const

  Get value of base function with index s, on point (x,y)
- ArrG grad (unsigned int s, Arr const &v, Arr const &w)

  Get values of gradient of basis function s, on points (v,w)

Protected Attributes

- std::vector< std::function
  < Arr(Arr const &, Arr const &)> > M_phi
- std::vector< std::function
  < ArrG(Arr const &, Arr const &)> > M_grad

6.23.1 Detailed Description

template<int N>class Tspeed::ShapeFunction< N >

Base class for the shared functions.

Template Parameters

N	degree of the space $\mathbb{P}_N$

6.23.2 Member Enumeration Documentation

6.23.2.1 template<int N> anonymous enum

Number of degrees of freedom.

Enumerator

gdl

6.23.2.2 template< int N > anonymous enum

Orthonormality of the basis.

Enumerator

is_orthonormal

6.23.3 Constructor & Destructor Documentation

6.23.3.1 template< int N > virtual Tspeed::ShapeFunction< N >::~ShapeFunction ( ) [inline], [virtual]

6.23.3.2 template< int N > Tspeed::ShapeFunction< N >::ShapeFunction ( ) [inline]

6.23.4 Member Function Documentation

6.23.4.1 template< int N > ArrG Tspeed::ShapeFunction< N >::grad (unsigned int *s*, Arr const & *v*, Arr const & *w*) [inline]

Get values of gradient of basis function s, on points (v,w)

Parameters

s	Index f the function
v	x-coordinates of the points
w	y-coordinates of the points

Returns

An array of dimension length(v) , 2 with the values

6.23.4.2 template< int N > Eigen::ArrayXd Tspeed::ShapeFunction< N >::phi (unsigned int *s*, Arr const & *v*, Arr const & *w*) const [inline]

Get value of base function with index s, on points (v,w)

Parameters

s	index of the basis function
v	x-coordinates of the points
w	y-coordinates of the points

Returns

An Eigen array with the values

6.23.4.3 template< int N > double Tspeed::ShapeFunction< N >::phi (unsigned int *s*, double *x*, double *y*) const [inline]

Get value of base function with index s, on point (x,y)

Parameters

s	index of the basis function
x	x-coordinates of the point
y	y-coordinates of the point

Returns

the value

6.23.5 Member Data Documentation

6.23.5.1 template<int N> std::vector<std::function<ArrG(Arr const &, Arr const &)> > Tspeed::ShapeFunction< N >::M_grad `[protected]`

6.23.5.2 template<int N> std::vector<std::function<Arr(Arr const &, Arr const &)> > Tspeed::ShapeFunction< N >::M_phi `[protected]`

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

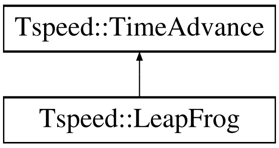
- lib/include/ShapeFunctions.hpp

6.24 Tspeed::TimeAdvance Class Reference

Base class for time stepping methods.

`#include <TimeAdvance.hpp>`

Inheritance diagram for Tspeed::TimeAdvance:



Public Member Functions

- template<int N, typename Q , typename S >
  TimeAdvance (FESpace_ptr< N, Q, S > Xh, Parameters const &p, Receivers const &r)

  constructor from the space, parameters and receivers
- void first_step ()

  The first step of the method (which is different for 2nd order methods)
- void step (double t)

  step at time t
- virtual ∼TimeAdvance ()
- void set_dt (double dt)

  Set time step δt.
- void set_tmax (double tmax)

  Set end time of the simulation.
- void set_penalty (double p)

  set penalty for the stability matrix
- void add_force (std::shared_ptr< Force > f)

Add the force.

- template< int N, typename Q , typename S >
  void set_initial_v (FESpace_ptr< N, Q, S > Xh, std::function< std::array< double, 2 >(double, double)> fun)

  Set initial speed $\dot{u}$.
- template< int N, typename Q , typename S >
  void set_initial_u (FESpace_ptr< N, Q, S > Xh, std::function< std::array< double, 2 >(double, double)> fun)

  Set initial displacement u.
- bool is_running ()

  Check if the method has arrived at the final time.
- Vec const & get_uh () const

  Get the coefficients of the numerical solution u_h.
- Vec const & u () const
- void eval_receivers ()

  Compute and store the value of the displacement at the receivers.
- void write_receivers (std::string const &fn) const

  Write the time series of the displacement at the receivers.

Protected Member Functions

- void update_variables (double t)

Protected Attributes

- double M_penalty
- double M_dt
- double M_tmax
- Vec f
- Vec fold
- Vec foldold
- Vec uh
- Vec uhold
- Vec uholdold
- Vec initial_v
- Receivers M_recv
- Matrices M_mat
- MyMatMultiDim< MyMat > B
- std::shared_ptr< Force > M_f
- bool M_completed
- double M_last_step
- unsigned int M_recv_written
- unsigned int M_nln
- unsigned int M_ne

6.24.1 Detailed Description

Base class for time stepping methods.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 template<int N, typename Q , typename S > Tspeed::TimeAdvance::TimeAdvance (**FESpace_ptr**< N, Q, S > *Xh*, **Parameters** const & *p*, **Receivers** const & *r*)

constructor from the space, parameters and receivers

Template Parameters

N,Q,S	template parameters of the space

Parameters

Xh	the function space
p	the parameters of the materials
r	the receivers

6.24.2.2 virtual Tspeed::TimeAdvance::~TimeAdvance ( ) `[inline]`, `[virtual]`

6.24.3 Member Function Documentation

6.24.3.1 void Tspeed::TimeAdvance::add_force (std::shared_ptr< **Force** > *f*) `[inline]`

Add the force.

Parameters

f	force

6.24.3.2 void Tspeed::TimeAdvance::eval_receivers ( )

Compute and store the value of the displacement at the receivers.

6.24.3.3 void Tspeed::TimeAdvance::first_step ( )

The first step of the method (which is different for 2nd order methods)

6.24.3.4 Vec const& Tspeed::TimeAdvance::get_uh ( ) const `[inline]`

Get the coefficients of the numerical solution u_h.

Returns

u_h

6.24.3.5 bool Tspeed::TimeAdvance::is_running ( ) `[inline]`

Check if the method has arrived at the final time.

Returns

TRUE if it is still running

6.24.3.6 void Tspeed::TimeAdvance::set_dt (double *dt*) [inline]

Set time step δt.

Parameters

dt	the time step

6.24.3.7 template< int N, typename Q , typename S > void Tspeed::TimeAdvance::set_initial_u (FESpace_ptr< N, Q, S > *Xh,* std::function< std::array< double, 2 >(double, double)> *fun*)

Set initial displacement u.

Parameters

Xh	the function space
fun	$u(x,y)$

6.24.3.8 template< int N, typename Q , typename S > void Tspeed::TimeAdvance::set_initial_v (FESpace_ptr< N, Q, S > *Xh,* std::function< std::array< double, 2 >(double, double)> *fun*)

Set initial speed $\dot{u}$.

Parameters

Xh	the function space
fun	$\dot{u}(x,y)$

6.24.3.9 void Tspeed::TimeAdvance::set_penalty (double *p*) [inline]

set penalty for the stability matrix

Parameters

p	the penalty value

6.24.3.10 void Tspeed::TimeAdvance::set_tmax (double *tmax*) [inline]

Set end time of the simulation.

Parameters

tmax	the end time

6.24.3.11 void Tspeed::TimeAdvance::step (double *t*)

step at time t

Parameters

t	the time

6.24.3.12 Vec const& Tspeed::TimeAdvance::u ( ) const [inline]

6.24.3.13 void Tspeed::TimeAdvance::update_variables (double *t*) [inline], [protected]

6.24.3.14 void Tspeed::TimeAdvance::write_receivers (std::string const & *fn*) const [inline]

Write the time series of the displacement at the receivers.

Parameters

fn	Base output file name

6.24.4 Member Data Documentation

6.24.4.1 MyMatMultiDim<MyMat> Tspeed::TimeAdvance::B [protected]

6.24.4.2 Vec Tspeed::TimeAdvance::f [protected]

6.24.4.3 Vec Tspeed::TimeAdvance::fold [protected]

6.24.4.4 Vec Tspeed::TimeAdvance::foldold [protected]

6.24.4.5 Vec Tspeed::TimeAdvance::initial_v [protected]

6.24.4.6 bool Tspeed::TimeAdvance::M_completed [protected]

6.24.4.7 double Tspeed::TimeAdvance::M_dt [protected]

6.24.4.8 std::shared_ptr<Force> Tspeed::TimeAdvance::M_f [protected]

6.24.4.9 double Tspeed::TimeAdvance::M_last_step [protected]

6.24.4.10 Matrices Tspeed::TimeAdvance::M_mat [protected]

6.24.4.11 unsigned int Tspeed::TimeAdvance::M_ne [protected]

6.24.4.12 unsigned int Tspeed::TimeAdvance::M_nln [protected]

6.24.4.13 double Tspeed::TimeAdvance::M_penalty [protected]

6.24.4.14 Receivers Tspeed::TimeAdvance::M_recv [protected]

6.24.4.15 unsigned int Tspeed::TimeAdvance::M_recv_written [protected]

6.24.4.16 double Tspeed::TimeAdvance::M_tmax [protected]

6.24.4.17 Vec Tspeed::TimeAdvance::uh [protected]

6.24.4.18 Vec Tspeed::TimeAdvance::uhold [protected]

6.24.4.19 Vec Tspeed::TimeAdvance::uholdold [protected]

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

- lib/include/TimeAdvance.hpp
- lib/include/TimeAdvance_imp.hpp

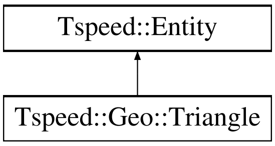
- lib/src/TimeAdvance.cpp

6.25 Tspeed::Geo::Triangle Class Reference

Class describing a triangle.

```
#include <Geometry.hpp>
```

Inheritance diagram for Tspeed::Geo::Triangle:



Public Member Functions

- Triangle ()
- Triangle (const Point &a, const Point &b, const Point &c)

  Create a triangle from three points.
- Triangle (const Triangle &)=default

  Copy constructor.
- Triangle & operator= (const Triangle &)

  Assignement.
- virtual ∼Triangle ()
- std::array< Point, 3 > all_pts () const

  Get all points of the triangle.
- std::array< Edge, 3 > all_edges () const

  Get all edges of a triangle.
- Point const & pt (int i) const

  Get a point.
- Edge const & edg (int i) const

  Get a edge.
- Eigen::Matrix2d Jac () const

  Get Jacobian of the transformation from the reference triangle.
- Eigen::Matrix2d invJac () const

  Get the inverse Jacobian of the transformation from the reference triangle.
- double detJ () const

  Get the determinant of the Jacobian of the tranformation from the reference triangle.
- Point map (Point const &p) const

  Map a point from its relative position in the reference triangle to the physical point.
- Point invmap (Point const &p) const

  Map a point from the physical triangle to the reference one.
- int const & neigh (int i) const

  Ged index of neighboring triangle on edge i.
- int const & neighedges (int i) const

  Index of the edge i in the nieghboring triangle.
- void setNeigh (int i, int j)

  Set neighboring triangle.
- void setNeighedges (int i, int j)

Set index of the edge of the nieghboring triangle.

- void printNeigh () const

  Pirnt neighbors for the current triangle.

- bool intriangle (const Point &p) const

  Check if point p is in triangle.

Static Public Attributes

- static const int numVertices =3

Additional Inherited Members

6.25.1 Detailed Description

Class describing a triangle.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 Tspeed::Geo::Triangle::Triangle ( )

6.25.2.2 Tspeed::Geo::Triangle::Triangle (const **Point** & *a*, const **Point** & *b*, const **Point** & *c*)

Create a triangle from three points.

Parameters

a,b,c	The three points

6.25.2.3 Tspeed::Geo::Triangle::Triangle (const **Triangle** &) `[default]`

Copy constructor.

6.25.2.4 virtual Tspeed::Geo::Triangle::∼Triangle ( ) `[inline]`, `[virtual]`

6.25.3 Member Function Documentation

6.25.3.1 std::array<**Edge**,3> Tspeed::Geo::Triangle::all_edges ( ) const `[inline]`

Get all edges of a triangle.

Returns

An array of the three edge

6.25.3.2 std::array<**Point**,3> Tspeed::Geo::Triangle::all_pts ( ) const `[inline]`

Get all points of the triangle.

Returns

An array of three points

6.25.3.3 double Tspeed::Geo::Triangle::detJ ( ) const

Get the determinant of the Jacobian of the tranformation from the reference triangle.

Returns

the determinant of the Jacobian (i.e., Area(T)∗2)

6.25.3.4 Edge const& Tspeed::Geo::Triangle::edg (int *i*) const [inline]

Get a edge.

Parameters

i	Number of the edge in the triangle

Returns

The i-th edge

6.25.3.5 bool Tspeed::Geo::Triangle::intriangle (const Point & *p*) const

Check if point p is in triangle.

Parameters

p	The point

Returns

TRUE if the point is in the triangle

6.25.3.6 Eigen::Matrix2d Tspeed::Geo::Triangle::invJac ( ) const

Get the inverse Jacobian of the transformation from the reference triangle.

Returns

The inverse Jacobian, in matrix form

6.25.3.7 Point Tspeed::Geo::Triangle::invmap (Point const & *p*) const

Map a point from the physical triangle to the reference one.

Parameters

p	The physical point

Returns

The point in the reference triangle

6.25.3.8 Eigen::Matrix2d Tspeed::Geo::Triangle::Jac ( ) const

Get Jacobian of the transformation from the reference triangle.

Returns

The Jacobian, in matrix form

6.25.3.9 Point Tspeed::Geo::Triangle::map (Point const & *p*) const

Map a point from its relative position in the reference triangle to the physical point.

Parameters

p	The point in the reference triangle

Returns

The physical point in the actual triangle

6.25.3.10 int const& Tspeed::Geo::Triangle::neigh (int *i*) const `[inline]`

Ged index of neighboring triangle on edge i.

Parameters

i	Edge of the triangle

Returns

Index of the neighbor

6.25.3.11 int const& Tspeed::Geo::Triangle::neighedges (int *i*) const `[inline]`

Index of the edge i in the nieghboring triangle.

Parameters

i	Edge of the present triangle

Returns

Index of the edge in the neighboring triangle

6.25.3.12 Triangle & Tspeed::Geo::Triangle::operator= (const Triangle & *t*)

Assignement.

6.25.3.13 void Tspeed::Geo::Triangle::printNeigh ( ) const `[inline]`

Pirnt neighbors for the current triangle.

6.25.3.14 Point const& Tspeed::Geo::Triangle::pt (int *i*) const [inline]

Get a point.

Parameters

i	Number of the point in the triangle

Returns

The i-th point

6.25.3.15 void Tspeed::Geo::Triangle::setNeigh (int *i*, int *j*) [inline]

Set neighboring triangle.

Parameters

i	Edge of the current triangle
j	Index of the neighbor

6.25.3.16 void Tspeed::Geo::Triangle::setNeighedges (int *i*, int *j*) [inline]

Set index of the edge of the nieghboring triangle.

Parameters

i	Edge of the current trinagle
j	Edge in the neighboring triangle

6.25.4 Member Data Documentation

6.25.4.1 const int Tspeed::Geo::Triangle::numVertices =3 [static]

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

- lib/include/Geometry.hpp
- lib/src/Geometry.cpp

Chapter 7

File Documentation

7.1 Examples/src/Lamb.cpp File Reference

```
#include "TSPEED.hpp"
#include <iostream>
```

Functions

- int main ()

7.1.1 Function Documentation

7.1.1.1 int main ( )

7.2 Examples/src/wedge.cpp File Reference

```
#include "TSPEED.hpp"
#include <iostream>
#include <memory>
```

Functions

- void wedge_init_param (double l, double m, double rho, double cf, double csurf, double &k, double &q, double &s, double &beta)
- int main ()

7.2.1 Function Documentation

7.2.1.1 int main ( )

7.2.1.2 void wedge_init_param (double *l*, double *m*, double *rho*, double *cf*, double *csurf*, double & *k*, double & *q*, double & *s*, double & *beta*)

7.3 lib/include/Dunavant.hpp File Reference

Header files for the implementation of the Dunavant rules.

Functions

- int dunavant_degree (int rule)
- int dunavant_order_num (int rule)
- void dunavant_rule (int rule, int order_num, double xy[ ], double w[ ])
- int dunavant_rule_num (void)
- int ∗ dunavant_suborder (int rule, int suborder_num)
- int dunavant_suborder_num (int rule)
- void dunavant_subrule (int rule, int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_01 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_02 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_03 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_04 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_05 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_06 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_07 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_08 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_09 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_10 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_11 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_12 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_13 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_14 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_15 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_16 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_17 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_18 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_19 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_20 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void file_name_inc (char ∗file_name)
- int i4_max (int i1, int i2)
- int i4_min (int i1, int i2)
- int i4_modp (int i, int j)
- int i4_wrap (int ival, int ilo, int ihi)
- double r8_huge (void)
- int r8_nint (double x)
- void reference_to_physical_t3 (double t[ ], int n, double ref[ ], double phy[ ])
- int s_len_trim (char ∗s)
- void timestamp (void)
- char ∗ timestring (void)
- double triangle_area (double t[2 ∗3])
- void triangle_points_plot (char ∗file_name, double node_xy[ ], int node_show, int point_num, double point_xy[ ], int point_show)

7.3.1 Detailed Description

Header files for the implementation of the Dunavant rules.

Author

John Burkardt

7.3.2 Function Documentation

7.3.2.1 int dunavant_degree (int *rule*)

7.3.2.2 int dunavant_order_num (int *rule*)

7.3.2.3 void dunavant_rule (int *rule,* int *order_num,* double *xy[ ],* double *w[ ]*)

7.3.2.4 int dunavant_rule_num (void)

7.3.2.5 int∗ dunavant_suborder (int *rule,* int *suborder_num*)

7.3.2.6 int dunavant_suborder_num (int *rule*)

7.3.2.7 void dunavant_subrule (int *rule,* int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.8 void dunavant_subrule_01 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.9 void dunavant_subrule_02 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.10 void dunavant_subrule_03 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.11 void dunavant_subrule_04 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.12 void dunavant_subrule_05 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.13 void dunavant_subrule_06 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.14 void dunavant_subrule_07 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.15 void dunavant_subrule_08 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.16 void dunavant_subrule_09 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.17 void dunavant_subrule_10 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.18 void dunavant_subrule_11 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.19 void dunavant_subrule_12 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.20 void dunavant_subrule_13 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.21 void dunavant_subrule_14 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.22 void dunavant_subrule_15 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.23 void dunavant_subrule_16 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.24 void dunavant_subrule_17 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.25 void dunavant_subrule_18 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.26 void dunavant_subrule_19 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.27 void dunavant_subrule_20 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.3.2.28 void file_name_inc (char ∗ *file_name*)

7.3.2.29 int i4_max (int *i1,* int *i2*)

7.3.2.30 int i4_min (int *i1,* int *i2*)

7.3.2.31 int i4_modp (int *i,* int *j*)

7.3.2.32 int i4_wrap (int *ival,* int *ilo,* int *ihi*)

7.3.2.33 double r8_huge (void)

7.3.2.34 int r8_nint (double *x*)

7.3.2.35 void reference_to_physical_t3 (double *t[ ],* int *n,* double *ref[ ],* double *phy[ ]*)

7.3.2.36 int s_len_trim (char ∗ *s*)

7.3.2.37 void timestamp (void)

7.3.2.38 char∗ timestring (void)

7.3.2.39 double triangle_area (double *t[2 ∗3]*)

7.3.2.40 void triangle_points_plot (char ∗ *file_name,* double *node_xy[ ],* int *node_show,* int *point_num,* double *point_xy[ ],* int *point_show*)

7.4 lib/include/FESpace.hpp File Reference

Header file for the Galerkin space and for the parameters of the elastodynamics equation.

```
#include "QuadratureRule.hpp"
#include "ShapeFunctions.hpp"
#include "Mesh.hpp"
#include <Eigen/Dense>
#include <Eigen/StdVector>
#include <functional>
#include "FESpace_imp.hpp"
```

Classes

- class Tspeed::FESpace< N, Q, S >

  Functional space.
- class Tspeed::Parameters

  Class for the parameters λ,ρ,μ of the elastodynamics equations.

Namespaces

- namespace Tspeed

Typedefs

- template<int N, typename Q = Gauss<N+1>, typename S = Dubiner<N>>
  using Tspeed::FESpace_ptr = std::shared_ptr< FESpace< N, Q, S >>

template pointer to FESpace

7.4.1 Detailed Description

Header file for the Galerkin space and for the parameters of the elastodynamics equation.

Author

Carlo Marcati

Date

2013-09-08

7.5 lib/include/FESpace_imp.hpp File Reference

Implementation of the functional space class methods.

Namespaces

- namespace Tspeed

7.5.1 Detailed Description

Implementation of the functional space class methods.

Author

Carlo Marcati

Date

2013-09-08

7.6 lib/include/Force.hpp File Reference

Header file for the force.

```
#include <functional>
#include <Eigen/SparseCore>
#include "Receivers.hpp"
#include "FESpace.hpp"
#include <array>
#include "Force_imp.hpp"
```

Classes

- class Tspeed::Force

  virtual base class for forces
- class Tspeed::PointWiseForce

  Time dependent force acting on a point.

Namespaces

- namespace Tspeed

Typedefs

- typedef std::shared_ptr< Force > Tspeed::Force_ptr

7.6.1 Detailed Description

Header file for the force.

Author

Carlo Marcati

Date

2013-09-08

7.7 lib/include/Force_imp.hpp File Reference

Implementation of the Pointwise force template methods.

```
#include "Force.hpp"
```

Namespaces

- namespace Tspeed

7.7.1 Detailed Description

Implementation of the Pointwise force template methods.

Author

Carlo Marcati

Date

2013-09-08

7.8 lib/include/Geometry.hpp File Reference

Header file for the geometrical entities.

```
#include <array>
#include <cmath>
#include <Eigen/Dense>
#include <memory>
#include <limits>
#include <iostream>
```

Classes

- class Tspeed::Entity

  Base class for geometrical entities.
- class Tspeed::Geo::Point

  Class describing points.
- class Tspeed::Geo::Edge

  Class describing an edge.
- class Tspeed::Geo::Triangle

  Class describing a triangle.

Namespaces

- namespace Tspeed
- namespace Tspeed::Geo

Enumerations

- enum Tspeed::Bc { Tspeed::Dirichlet, Tspeed::Neumann, Tspeed::Internal, Tspeed::Unassigned }

Functions

- std::ostream & Tspeed::Geo::operator<< (std::ostream &, Triangle const &)
- std::ostream & Tspeed::Geo::operator<< (std::ostream &, Point const &)

Variables

- const unsigned int NVAL =std::numeric_limits<unsigned int>::max()

7.8.1 Detailed Description

Header file for the geometrical entities.

Author

Carlo Marcati

Date

2013-09-08

7.8.2 Variable Documentation

7.8.2.1 const unsigned int NVAL =std::numeric_limits<unsigned int>::max()

7.9 lib/include/Matrices_imp.hpp File Reference

Implementation of the matrices for the method - templated part.

Namespaces

- namespace Tspeed

7.9.1 Detailed Description

Implementation of the matrices for the method - templated part.

Author

Carlo Marcati

Date

2013-09-08

7.10 lib/include/Mesh.hpp File Reference

Header file for the mesh.

```
#include <string>
#include <fstream>
#include <iostream>
#include <algorithm>
#include <map>
#include <Eigen/StdVector>
#include "Geometry.hpp"
```

Classes

- class Tspeed::Mesh

Namespaces

- namespace Tspeed

Typedefs

- typedef std::shared_ptr< Mesh > Tspeed::Mesh_ptr

  Shared pointer to an element of type mesh.

7.10.1 Detailed Description

Header file for the mesh.

Author

Carlo Marcati

Date

2013-09-08

7.11 lib/include/MyMat.hpp File Reference

Header file for the matrices specialized for the Discontinuous Galerkin method.

```
#include <Eigen/Dense>
#include <vector>
#include "Mesh.hpp"
#include <fstream>
```

Classes

- class Tspeed::BaseMat

  Base monodimensional matrix class.
- class Tspeed::MyMatBlockDiag

  Block diagonal monodimensional matrix (monodimensional block of mass and stress-strain matrices)
- class Tspeed::MyMat

  Block Matrix (monodimensial blocks of stability and interelement matrices)
- class Tspeed::MyMatMultiDim< T >

  Multidimensional matrix.
- class Tspeed::MyMatMultiDimBlockDiag< T >

  Matrix class for matrices where only the diagonal monodimensional matrices are non zero (i.e. the mass matrix)

Namespaces

- namespace Tspeed

Functions

- MyMat Tspeed::operator∗ (double const &c, MyMat const &M)
- Eigen::VectorXd Tspeed::operator∗ (MyMat const &, Eigen::VectorXd const &)
- Eigen::VectorXd Tspeed::operator∗ (MyMatBlockDiag const &, Eigen::VectorXd const &)
- MyMat Tspeed::operator+ (MyMat a, MyMat const &b)
- MyMat Tspeed::operator+ (MyMat a, MyMatBlockDiag const &b)

7.11.1 Detailed Description

Header file for the matrices specialized for the Discontinuous Galerkin method.

Author

Carlo Marcati

Date

2013-09-08

7.12 lib/include/QuadratureRule.hpp File Reference

Header file for the quadrature rules.

```
#include <Eigen/Dense>
#include <limits>
#include <iostream>
#include "Geometry.hpp"
#include "Dunavant.hpp"
#include "QuadratureRule_imp.hpp"
```

Classes

- class Tspeed::QuadratureRule< N >

  Base class for quadrature rules.
- class Tspeed::Gauss< N >

  Gauss quadrature rule on the triangle.
- class Tspeed::Dunavant< N >

  Dunavant [1] quadrature rule.

Namespaces

- namespace Tspeed

7.12.1 Detailed Description

Header file for the quadrature rules. A base class is implemented, with derived classes which implement Gauss quadrature on the triangle and Dunavant quadrature

Reference: [1] D. A. Dunavant, High degree efficient symmetrical Gaussian quadra- ture rules for the triangle, Internat. J. Numer. Methods Engrg. 21 (1985), no. 6, 1129–1148.

Author

Carlo Marcati

Date

2013-09-08

7.13 lib/include/QuadratureRule_imp.hpp File Reference

Implementation of the quadrature rules.

Namespaces

- namespace Tspeed

7.13.1 Detailed Description

Implementation of the quadrature rules.

Author

Carlo Marcati

Date

2013-09-08

7.14 lib/include/Receivers.hpp File Reference

Header file containg the class for receivers and a base class for pointwise entities.

```
#include <string>
#include "Geometry.hpp"
#include "FESpace.hpp"
#include <fstream>
#include <vector>
#include "Receivers_imp.hpp"
```

Classes

- class Tspeed::PointWiseEntity

  A base class for pointwise entities, with the points and the basis function in that points.
- class Tspeed::Receivers

  A class for seismic receivers, i.e., receivers recording the movement at a point.

Namespaces

- namespace Tspeed

7.14.1 Detailed Description

Header file containg the class for receivers and a base class for pointwise entities.

Author

Carlo Marcati

Date

2013-09-08

7.15 lib/include/Receivers_imp.hpp File Reference

Implementation of the pointwise entity and receivers class.

```
#include "Receivers.hpp"
```

Namespaces

- namespace Tspeed

7.15.1 Detailed Description

Implementation of the pointwise entity and receivers class.

Author

Carlo Marcati

Date

2013-09-08

7.16 lib/include/ShapeFunctions.hpp File Reference

Header file for the definition of the shape functions.

```
#include <functional>
#include <vector>
#include <Eigen/Dense>
#include "ShapeFunctions_imp.hpp"
```

Classes

- class Tspeed::ShapeFunction< N >

  Base class for the shared functions.
- class Tspeed::Dubiner< N >

  Dubiner [1] basis.
- class Tspeed::BoundaryAdapted< N >

  Boundary adapted [2] basis.

Namespaces

- namespace Tspeed

7.16.1 Detailed Description

Header file for the definition of the shape functions. A base class is used, and the Dubiner and Boundary adapted derived classes are implemented. See

[1] M. Dubiner, Spectral methods on triangles and other domains, Journal of Scientific Computing 6 (1991), no. 4, 345–390

[2] G. E. Karniadakis and S. J. Sherwin, Spectral/hp element methods for computational fluid dynamics, second ed., Numerical Mathemat ics and Scientific Computation, Oxford University Press, New York.

Author

Carlo Marcati

Date

2013-09-08

7.17 lib/include/ShapeFunctions_imp.hpp File Reference

Implementation of the shape functions.

Namespaces

- namespace Tspeed

Functions

- Eigen::ArrayXd Tspeed::jacobi_polynomial (int N, int alpha, int beta, Eigen::ArrayXd const &z)

7.17.1 Detailed Description

Implementation of the shape functions.

Author

Carlo Marcati

Date

2013-09-08

7.18 lib/include/TimeAdvance.hpp File Reference

Header file for the implementation of the time stepping and of the matrices for the method.

```
#include <Eigen/SparseCore>
#include <Eigen/Dense>
#include "FESpace.hpp"
#include "Receivers.hpp"
#include "Geometry.hpp"
#include "Force.hpp"
#include "MyMat.hpp"
#include <memory>
#include <limits>
#include "Matrices_imp.hpp"
#include "TimeAdvance_imp.hpp"
```

Classes

- class Tspeed::Matrices

  The class containing the matrices resulting from the spatial discretization.
- class Tspeed::TimeAdvance

  Base class for time stepping methods.
- class Tspeed::LeapFrog

  Implementation of the second order Leap-Frog explicit time stepping scheme.

Namespaces

- namespace Tspeed

Functions

- double Tspeed::mat_dot (Eigen::Matrix2d const &a, Eigen::Matrix2d const &b)

    Dot product between two 2x2 matrices.
- Eigen::Matrix2d Tspeed::CTensorProduct (Eigen::Matrix2d const &A, double lambda, double mu)

    tensor product between Hooke's tensor and matrix A

7.18.1 Detailed Description

Header file for the implementation of the time stepping and of the matrices for the method.

Author

Carlo Marcati

Version

Date

2013-09-08

7.19 lib/include/TimeAdvance_imp.hpp File Reference

Implementation of the time advancing template methods.

```
#include "TimeAdvance.hpp"
```

Namespaces

- namespace Tspeed

7.19.1 Detailed Description

Implementation of the time advancing template methods.

Author

Carlo Marcati

Date

2013-09-08

7.20 lib/include/TSPEED.hpp File Reference

```
#include "QuadratureRule.hpp"
#include "ShapeFunctions.hpp"
#include "FESpace.hpp"
#include "Mesh.hpp"
#include "Receivers.hpp"
#include "Force.hpp"
#include "TimeAdvance.hpp"
#include "MyMat.hpp"
```

7.21 TSPEED.hpp File Reference

```
#include "QuadratureRule.hpp"
#include "ShapeFunctions.hpp"
#include "FESpace.hpp"
#include "Mesh.hpp"
#include "Receivers.hpp"
#include "Force.hpp"
#include "TimeAdvance.hpp"
#include "MyMat.hpp"
```

7.22 lib/src/Dunavant.cpp File Reference

Functions for Dunavant quadrature (nodes and weights, tabulated)

```
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <iomanip>
#include <cmath>
#include <ctime>
#include <cstring>
#include "Dunavant.hpp"
```

Macros

- #define TIME_SIZE 40
- #define TIME_SIZE 40

Functions

- int dunavant_degree (int rule)
- int dunavant_order_num (int rule)
- void dunavant_rule (int rule, int order_num, double xy[ ], double w[ ])
- int dunavant_rule_num ()
- int ∗ dunavant_suborder (int rule, int suborder_num)
- int dunavant_suborder_num (int rule)
- void dunavant_subrule (int rule, int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_01 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])

- void dunavant_subrule_02 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_03 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_04 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_05 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_06 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_07 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_08 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_09 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_10 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_11 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_12 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_13 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_14 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_15 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_16 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_17 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_18 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_19 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void dunavant_subrule_20 (int suborder_num, double suborder_xyz[ ], double suborder_w[ ])
- void file_name_inc (char ∗file_name)
- int i4_max (int i1, int i2)
- int i4_min (int i1, int i2)
- int i4_modp (int i, int j)
- int i4_wrap (int ival, int ilo, int ihi)
- double r8_huge ()
- int r8_nint (double x)
- void reference_to_physical_t3 (double t[ ], int n, double ref[ ], double phy[ ])
- int s_len_trim (char ∗s)
- void timestamp ()
- char ∗ timestring ()
- double triangle_area (double t[2 ∗3])
- void triangle_points_plot (char ∗file_name, double node_xy[ ], int node_show, int point_num, double point_xy[ ], int point_show)

7.22.1 Detailed Description

Functions for Dunavant quadrature (nodes and weights, tabulated)

Author

John Burkardt

7.22.2 Macro Definition Documentation

7.22.2.1 #define TIME_SIZE 40

7.22.2.2 #define TIME_SIZE 40

7.22.3 Function Documentation

7.22.3.1 int dunavant_degree (int *rule*)

7.22.3.2 int dunavant_order_num (int *rule*)

7.22.3.3 void dunavant_rule (int *rule,* int *order_num,* double *xy[ ],* double *w[ ]*)

7.22.3.4 int dunavant_rule_num (void)

7.22.3.5 int∗ dunavant_suborder (int *rule,* int *suborder_num*)

7.22.3.6 int dunavant_suborder_num (int *rule*)

7.22.3.7 void dunavant_subrule (int *rule,* int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.8 void dunavant_subrule_01 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.9 void dunavant_subrule_02 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.10 void dunavant_subrule_03 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.11 void dunavant_subrule_04 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.12 void dunavant_subrule_05 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.13 void dunavant_subrule_06 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.14 void dunavant_subrule_07 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.15 void dunavant_subrule_08 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.16 void dunavant_subrule_09 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.17 void dunavant_subrule_10 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.18 void dunavant_subrule_11 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.19 void dunavant_subrule_12 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.20 void dunavant_subrule_13 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.21 void dunavant_subrule_14 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.22 void dunavant_subrule_15 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.23 void dunavant_subrule_16 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.24 void dunavant_subrule_17 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.25 void dunavant_subrule_18 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.26 void dunavant_subrule_19 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.27 void dunavant_subrule_20 (int *suborder_num,* double *suborder_xyz[ ],* double *suborder_w[ ]*)

7.22.3.28 void file_name_inc (char ∗ *file_name*)

7.22.3.29 int i4_max (int *i1,* int *i2*)

7.22.3.30 int i4_min (int *i1,* int *i2*)

7.22.3.31 int i4_modp (int *i*, int *j*)

7.22.3.32 int i4_wrap (int *ival*, int *ilo*, int *ihi*)

7.22.3.33 double r8_huge (void)

7.22.3.34 int r8_nint (double *x*)

7.22.3.35 void reference_to_physical_t3 (double *t[ ]*, int *n*, double *ref[ ]*, double *phy[ ]*)

7.22.3.36 int s_len_trim (char ∗ *s*)

7.22.3.37 void timestamp (void)

7.22.3.38 char∗ timestring (void)

7.22.3.39 double triangle_area (double *t[2 ∗3]*)

7.22.3.40 void triangle_points_plot (char ∗ *file_name*, double *node_xy[ ]*, int *node_show*, int *point_num*, double *point_xy[ ]*, int *point_show*)

7.23 lib/src/Force.cpp File Reference

Implementation of the Force method.

```
#include "Force.hpp"
```

Namespaces

- namespace Tspeed

7.23.1 Detailed Description

Implementation of the Force method.

Author

Carlo Marcati

Date

2013-09-08

7.24 lib/src/Geometry.cpp File Reference

Implementation of the geometrical entities.

```
#include "Geometry.hpp"
```

Namespaces

- namespace Tspeed
- namespace Tspeed::Geo

Functions

- std::ostream & Tspeed::Geo::operator<< (std::ostream &, Point const &)
- Point Tspeed::Geo::operator- (const Point &a, const Point &b)
- Point Tspeed::Geo::operator- (const Eigen::Vector2d &a, const Point &b)
- Point Tspeed::Geo::operator- (const Point &a, const Eigen::Vector2d &b)
- Point Tspeed::Geo::operator+ (const Eigen::Vector2d &a, const Point &b)
- Point Tspeed::Geo::operator+ (const Point &a, const Eigen::Vector2d &b)
- Point Tspeed::Geo::operator+ (const Point &a, const Point &b)
- Point Tspeed::Geo::operator* (const double &d, const Point &p)
- std::ostream & Tspeed::Geo::operator<< (std::ostream &, Triangle const &)

7.24.1 Detailed Description

Implementation of the geometrical entities.

Author

Carlo Marcati

Date

2013-09-08

7.25 lib/src/Mesh.cpp File Reference

Implementation of the mesh.

```
#include "Mesh.hpp"
```

Namespaces

- namespace Tspeed

7.25.1 Detailed Description

Implementation of the mesh.

Author

Carlo Marcati

Date

2013-09-08

7.26 lib/src/MyMat.cpp File Reference

Implementation of the matrix classes.

```
#include "MyMat.hpp"
```

Namespaces

- namespace Tspeed

Functions

- Eigen::VectorXd Tspeed::operator∗ (MyMatMultiDimBlockDiag< MyMatBlockDiag > const &A, Eigen::VectorXd const &v)
- Eigen::VectorXd Tspeed::operator∗ (MyMatMultiDim< MyMat > &A, Eigen::VectorXd const &v)
- Eigen::VectorXd Tspeed::operator∗ (MyMatMultiDim< MyMatBlockDiag > &A, Eigen::VectorXd const &v)
- MyMat Tspeed::operator∗ (double const &c, MyMat const &M)
- MyMatMultiDim< MyMat > Tspeed::operator+ (MyMatMultiDim< MyMat > const &a, MyMatMultiDim< MyMat > const &b)
- MyMatMultiDim< MyMat > Tspeed::operator+ (MyMatMultiDim< MyMat > const &a, MyMatMultiDim< MyMatBlockDiag > const &b)
- MyMat Tspeed::operator+ (MyMatBlockDiag const &b, MyMat a)
- MyMat Tspeed::operator+ (MyMat a, MyMatBlockDiag const &b)
- MyMat Tspeed::operator+ (MyMat a, MyMat const &b)
- Eigen::VectorXd Tspeed::operator∗ (MyMatBlockDiag const &, Eigen::VectorXd const &)
- Eigen::VectorXd Tspeed::operator∗ (MyMat const &, Eigen::VectorXd const &)

7.26.1 Detailed Description

Implementation of the matrix classes.

Author

Carlo Marcati

Date

2013-09-08

7.27 lib/src/Parameters.cpp File Reference

Implementation of the Parameters methods.

```
#include "FESpace.hpp"
```

Namespaces

- namespace Tspeed

7.27.1 Detailed Description

Implementation of the Parameters methods.

Author

Carlo Marcati

Date

2013-09-08

7.28 lib/src/Receivers.cpp File Reference

Implementation of the Receivers methods.

```
#include "Receivers.hpp"
```

Namespaces

- namespace Tspeed

7.28.1 Detailed Description

Implementation of the Receivers methods.

Author

Carlo Marcati

Date

2013-09-08

7.29 lib/src/ShapeFunctions.cpp File Reference

Implementation of the jacobi polynomials.

```
#include "ShapeFunctions.hpp"
```

Typedefs

- typedef Eigen::ArrayXd Arr

7.29.1 Detailed Description

Implementation of the jacobi polynomials.

Author

Carlo Marcati

Date

2013-09-08

7.29.2 Typedef Documentation

7.29.2.1 typedef Eigen::ArrayXd Arr

7.30 lib/src/TimeAdvance.cpp File Reference

Implmentation of the TimeAdvance methods.

```
#include "TimeAdvance.hpp"
```

Namespaces

- namespace Tspeed

Functions

- Eigen::Matrix2d Tspeed::CTensorProduct (Eigen::Matrix2d const &A, double lambda, double mu)

  tensor product between Hooke's tensor and matrix A
- double Tspeed::mat_dot (Eigen::Matrix2d const &a, Eigen::Matrix2d const &b)

  Dot product between two 2x2 matrices.

7.30.1 Detailed Description

Implmentation of the TimeAdvance methods.

Author

Carlo Marcati

Date

2013-09-08

7.31 main.cpp File Reference

```
#include "Dunavant.hpp"
#include <iostream>
```

Functions

- int main ()

7.31.1 Function Documentation

7.31.1.1 int main ( )

Index