

Arbitrary order virtual element method for linear elastostatics

Generated by Doxygen 1.9.1

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	2
2.1 Class List	2
3 Class Documentation	3
3.1 geometry::Edge< PointType > Class Template Reference	3
3.1.1 Constructor & Destructor Documentation	4
3.1.2 Member Function Documentation	4
3.1.3 Friends And Related Function Documentation	7
3.2 EdgeDof Class Reference	7
3.3 FaceDof Class Reference	7
3.3.1 Constructor & Destructor Documentation	7
3.3.2 Member Function Documentation	8
3.4 Gauss::GaussData Struct Reference	9
3.5 GaussLobatto::GaussLobattoCache Class Reference	9
3.5.1 Member Function Documentation	9
3.6 LinearTrinomialPower Class Reference	10
3.6.1 Constructor & Destructor Documentation	10
3.7 LocalVirtualDofs Class Reference	11
3.7.1 Constructor & Destructor Documentation	11
3.7.2 Member Function Documentation	12
3.7.3 Friends And Related Function Documentation	15
3.8 LocalVirtualDofsCollection Class Reference	15
3.8.1 Constructor & Destructor Documentation	15
3.8.2 Member Function Documentation	16
3.9 Mesh< PointType, EdgeType, PolygonType, PolyhedronType > Class Template Reference	16
3.9.1 Constructor & Destructor Documentation	17
3.9.2 Member Function Documentation	17
3.10 Monomial< Dimension > Class Template Reference	21
3.10.1 Constructor & Destructor Documentation	22
3.10.2 Member Function Documentation	22
3.10.3 Friends And Related Function Documentation	24
3.11 Monomial2D Class Reference	24
3.11.1 Constructor & Destructor Documentation	25
3.11.2 Member Function Documentation	26
3.11.3 Friends And Related Function Documentation	28
3.12 Monomial3D Class Reference	29
3.12.1 Constructor & Destructor Documentation	30
3.12.2 Member Function Documentation	30
3.12.3 Friends And Related Function Documentation	33
3.13 IntegrationMonomial::MonomialsFaceIntegralsCache Class Reference	34

3.13.1 Member Function Documentation	34
3.14 Parameters Class Reference	36
3.14.1 Constructor & Destructor Documentation	37
3.14.2 Member Function Documentation	37
3.15 geometry::Point< Args > Class Template Reference	40
3.15.1 Constructor & Destructor Documentation	41
3.15.2 Member Function Documentation	41
3.15.3 Friends And Related Function Documentation	48
3.16 geometry::Polygon< EdgeType > Class Template Reference	49
3.16.1 Constructor & Destructor Documentation	50
3.16.2 Member Function Documentation	50
3.16.3 Friends And Related Function Documentation	55
3.17 geometry::Polyhedron< PolygonType > Class Template Reference	55
3.17.1 Constructor & Destructor Documentation	56
3.17.2 Member Function Documentation	56
3.17.3 Friends And Related Function Documentation	58
3.18 PolyhedronDof Class Reference	59
3.18.1 Constructor & Destructor Documentation	59
3.18.2 Member Function Documentation	59
3.19 Polynomial< Dimension > Class Template Reference	60
3.19.1 Constructor & Destructor Documentation	61
3.19.2 Member Function Documentation	61
3.20 Problem Class Reference	63
3.21 Solver Class Reference	63
3.21.1 Constructor & Destructor Documentation	64
3.21.2 Member Function Documentation	64
3.22 SolverVEM Class Reference	65
3.22.1 Constructor & Destructor Documentation	65
3.22.2 Member Function Documentation	65
3.23 VertexDof Class Reference	66
3.23.1 Constructor & Destructor Documentation	67
3.23.2 Member Function Documentation	67
3.24 VirtualDof Class Reference	68
3.24.1 Member Function Documentation	68
3.24.2 Friends And Related Function Documentation	69
3.25 VirtualDofsCollection Class Reference	70
3.25.1 Constructor & Destructor Documentation	70
3.25.2 Member Function Documentation	71
3.25.3 Friends And Related Function Documentation	73
3.26 VirtualFaceProjections Class Reference	73
3.26.1 Constructor & Destructor Documentation	73
3.26.2 Member Function Documentation	74

3.27 VirtualPolyhedronProjections Class Reference	75
3.27.1 Constructor & Destructor Documentation	75
3.27.2 Member Function Documentation	76
Index	79

1 Hierarchical Index

1.1 Class Hierarchy

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

geometry::Edge< PointType >	3
Gauss::GaussData	9
GaussLobatto::GaussLobattoCache	9
LocalVirtualDofs	11
LocalVirtualDofsCollection	15
Mesh< PointType, EdgeType, PolygonType, PolyhedronType >	16
Monomial< Dimension >	21
Monomial< 2 >	21
Monomial2D	24
Monomial< 3 >	21
Monomial3D	29
IntegrationMonomial::MonomialsFaceIntegralsCache	34
Parameters	36
geometry::Point< Args >	40
geometry::Point< real, real, real >	40
geometry::Polygon< EdgeType >	49
geometry::Polyhedron< PolygonType >	55
Polynomial< Dimension >	60
Polynomial< 2 >	60
LinearTrinomialPower	10
Problem	63
Solver	63
SolverVEM	65

VirtualDof	68
EdgeDof	7
FaceDof	7
PolyhedronDof	59
VertexDof	66
VirtualDofsCollection	70
VirtualFaceProjections	73
VirtualPolyhedronProjections	75

2 Class Index

2.1 Class List

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

geometry::Edge< PointType >	3
EdgeDof	7
FaceDof	7
Gauss::GaussData	9
GaussLobatto::GaussLobattoCache	9
LinearTrinomialPower	10
LocalVirtualDofs	11
LocalVirtualDofsCollection	15
Mesh< PointType, EdgeType, PolygonType, PolyhedronType >	16
Monomial< Dimension >	21
Monomial2D	24
Monomial3D	29
IntegrationMonomial::MonomialsFaceIntegralsCache	34
Parameters	36
geometry::Point< Args >	40
geometry::Polygon< EdgeType >	49
geometry::Polyhedron< PolygonType >	55
PolyhedronDof	59
Polynomial< Dimension >	60

Problem	63
Solver	63
SolverVEM	65
VertexDof	66
VirtualDof	68
VirtualDofsCollection	70
VirtualFaceProjections	73
VirtualPolyhedronProjections	75

3 Class Documentation

3.1 geometry::Edge< PointType > Class Template Reference

Public Member Functions

- [Edge](#) (const PointType &point1, const PointType &point2, bool _flipped=false)
Constructor with two points.
- void [setOtherHalfEdge](#) (const [Edge](#)< PointType > &otherEdge)
Setter method to set the other half-edge.
- [Edge](#)< PointType > & [getOtherHalfEdge](#) () const
Getter method to get the other half-edge.
- void [update](#) ()
Update the properties of the edge. When modifying one point, the edge automatically sees the change as they are references. It is not the case for other data structures.
- void [setId](#) (IndexType _id)
Set id.
- const IndexType & [getId](#) () const
Get id.
- const real & [getLength](#) () const
Get length.
- const PointType [getDirection](#) () const
Get direction.
- const PointType & [operator\[\]](#) (IndexType index) const
Constant getter.
- bool [operator<](#) (const [Edge](#)< PointType > &other) const
Define the comparison function based on edge ids.
- bool [operator==](#) (const [Edge](#)< PointType > &other) const
Equality operator for edges.

Friends

- std::ostream & [operator<<](#) (std::ostream &os, const [Edge](#)< PointType > &edge)
Stream output operator for the [Edge](#) class.

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Edge() `template<typename PointType >
geometry::Edge< PointType >::Edge (
 const PointType & point1,
 const PointType & point2,
 bool _flipped = false) [inline]`

Constructor with two points.

Parameters

<i>point1</i>	
<i>point2</i>	
<i>_flipped</i>	if reading direction is flipped

3.1.2 Member Function Documentation

3.1.2.1 getDirection() `template<typename PointType >
const PointType geometry::Edge< PointType >::getDirection () const [inline]`

Get direction.

Returns

const PointType

3.1.2.2 getId() `template<typename PointType >
const IndexType& geometry::Edge< PointType >::getId () const [inline]`

Get id.

Returns

const IndexType&

3.1.2.3 getLength() `template<typename PointType >`
`const real& geometry::Edge< PointType >::getLength () const [inline]`

Get length.

Returns

`const real&`

3.1.2.4 getOtherHalfEdge() `template<typename PointType >`
`Edge<PointType>& geometry::Edge< PointType >::getOtherHalfEdge () const [inline]`

Getter method to get the other half-edge.

Returns

`Edge<PointType>&`

3.1.2.5 operator<() `template<typename PointType >`
`bool geometry::Edge< PointType >::operator< (`
`const Edge< PointType > & other) const [inline]`

Define the comparison function based on edge ids.

Parameters

<i>other</i>	
--------------	--

Returns

`true`

`false`

3.1.2.6 operator==() `template<typename PointType >`
`bool geometry::Edge< PointType >::operator== (`
`const Edge< PointType > & other) const [inline]`

Equality operator for edges.

Parameters

<i>other</i>	
--------------	--

Returns

true
false

3.1.2.7 operator[]() `template<typename PointType >
const PointType& geometry::Edge< PointType >::operator[] (
IndexType index) const [inline]`

Constant getter.

Parameters

<i>index</i>	
--------------	--



Returns

const PointType&

3.1.2.8 setId() `template<typename PointType >
void geometry::Edge< PointType >::setId (
IndexType _id) [inline]`

Set id.

Parameters

	
	
<i>id</i>	

3.1.2.9 setOtherHalfEdge() `template<typename PointType >
void geometry::Edge< PointType >::setOtherHalfEdge (
const Edge< PointType > & otherEdge) [inline]`

Setter method to set the other half-edge.

Parameters

<i>otherEdge</i>	
------------------	--

3.1.3 Friends And Related Function Documentation

3.1.3.1 operator<< `template<typename PointType >`
`std::ostream& operator<< (`
`std::ostream & os,`
`const Edge< PointType > & edge) [friend]`

Stream output operator for the [Edge](#) class.

Parameters

<i>os</i>	
<i>edge</i>	

Returns

`std::ostream&`

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

- `include/edge.hpp`

3.2 EdgeDof Class Reference

Inheritance diagram for EdgeDof:

3.3 FaceDof Class Reference

Inheritance diagram for FaceDof:

Collaboration diagram for FaceDof:

Public Member Functions

- [FaceDof](#) (`std::size_t id, const Monomial2D &monomial_`)
Constructor.
- `std::size_t getId ()` const override
Getter for the id of the face.
- `const Monomial2D & getMonomial ()` const
Getter for the monomial of the Face DOF.
- `std::ostream & operator<< (std::ostream &os)` const override
Output stream operator.

3.3.1 Constructor & Destructor Documentation

3.3.1.1 FaceDof() `FaceDof::FaceDof (`
`std::size_t id,`
`const Monomial2D & monomial_) [inline]`

Constructor.

Parameters

<i>id</i>	
<i>monomial</i> \leftrightarrow	
—	

3.3.2 Member Function Documentation**3.3.2.1 getId()** `std::size_t FaceDof::getId () const [override], [virtual]`

Getter for the id of the face.

Returns

`std::size_t`

Reimplemented from [VirtualDof](#).

3.3.2.2 getMonomial() `const Monomial2D & FaceDof::getMonomial () const`

Getter for the monomial of the Face DOF.

Returns

`const Monomial2D&`

3.3.2.3 operator<<() `std::ostream& FaceDof::operator<< (std::ostream & os) const [inline], [override], [virtual]`

Output stream operator.

Parameters

<i>os</i>	
-----------	--

Returns

`std::ostream&`

Implements [VirtualDof](#).

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

- include/virtualDofs.hpp
- src/virtualDofs.cpp

3.4 Gauss::GaussData Struct Reference

Public Attributes

- unsigned int **N**
- std::array< real, MaxN3 > **x**
- std::array< real, MaxN3 > **y**
- std::array< real, MaxN3 > **z**
- std::array< real, MaxN3 > **w**

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

- include/integration.hpp

3.5 GaussLobatto::GaussLobattoCache Class Reference

Static Public Member Functions

- static void [initialize](#) (unsigned int n)
Initialize the cache.
- static const std::pair< std::vector< real >, std::vector< real > > & [getCache](#) (unsigned int n)
Get the cache.

3.5.1 Member Function Documentation

3.5.1.1 getCache() `const std::pair< std::vector< real >, std::vector< real > > & GaussLobatto::GaussLobattoCache::getCache (unsigned int n) [static]`

Get the cache.

Parameters

n	
-----	--

Returns

const std::pair<std::vector<real>, std::vector<real>>&

3.5.1.2 initialize() `void GaussLobatto::GaussLobattoCache::initialize (`
`unsigned int n) [static]`

Initialize the cache.

Parameters

<i>n</i>	
----------	--

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

- include/integration.hpp
- src/integration.cpp

3.6 LinearTrinomialPower Class Reference

Inheritance diagram for LinearTrinomialPower:

Collaboration diagram for LinearTrinomialPower:

Public Member Functions

- [LinearTrinomialPower](#) (const real &a, const real &b, const real &c, const unsigned int &power)
Construct a new Linear Trinomial Power object of the kind $(ax+by+c)^{\text{power}}$.

Additional Inherited Members

3.6.1 Constructor & Destructor Documentation

3.6.1.1 LinearTrinomialPower() `LinearTrinomialPower::LinearTrinomialPower (`
`const real & a,`
`const real & b,`
`const real & c,`
`const unsigned int & power)`

Construct a new Linear Trinomial Power object of the kind $(ax+by+c)^{\text{power}}$.

Parameters

<i>a</i>	
<i>b</i>	
<i>c</i>	
<i>power</i>	

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

- include/monomial.hpp
- src/monomial.cpp

3.7 LocalVirtualDofs Class Reference

Public Member Functions

- [LocalVirtualDofs](#) (const [Polyhedron](#)< [Polygon3D](#) > &P, const [VirtualDofsCollection](#) &DOFS)
Constructor.
- std::size_t [getID](#) (std::size_t id) const
Method to get the global id of the corresponding local dof id.
- template<typename DofType >
std::shared_ptr< DofType > [getDof](#) (std::size_t id) const
Method to get the corresponding specialized dof to a given id.
- std::size_t [VToLocalId](#) (const std::size_t &ID) const
Get the corresponding local dof for vertex-type global dof.
- std::vector< std::size_t > [EToLocalId](#) (const std::size_t &ID) const
Get the corresponding local dof for edge-type global dof.
- std::vector< std::size_t > [FToLocalId](#) (const std::size_t &ID) const
Get the corresponding local dof for face-type global dof.
- std::size_t [PToLocalId](#) (const std::size_t &ID) const
Get the corresponding local dof for polyhedron-type global dof.
- std::size_t [getnumVdofs](#) () const
Get the number of vertex-type dofs.
- std::size_t [getnumEdofs](#) () const
Get the number of edge-type dofs.
- std::size_t [getnumFdofs](#) () const
Get the number of face-type dofs.
- std::size_t [getnumPdofs](#) () const
Get the number of polyhedron-type dofs.
- std::size_t [getnumDofs](#) () const
Get the total number of local dofs.

Friends

- std::ostream & [operator<<](#) (std::ostream &os, const [LocalVirtualDofs](#) &dofsCollection)
Output stream operator for [LocalVirtualDofs](#).

3.7.1 Constructor & Destructor Documentation

3.7.1.1 LocalVirtualDofs() `LocalVirtualDofs::LocalVirtualDofs (`
 const [Polyhedron](#)< [Polygon3D](#) > & P,
 const [VirtualDofsCollection](#) & DOFS)

Constructor.

Parameters

<i>P</i>	
<i>DOFS</i>	

3.7.2 Member Function Documentation

3.7.2.1 EToLocalId() `std::vector<std::size_t> LocalVirtualDofs::EToLocalId (`
`const std::size_t & ID) const [inline]`

Get the corresponding local dof for edge-type global dof.

Parameters

<i>ID</i>	
-----------	--

Returns

`std::vector<std::size_t>`

3.7.2.2 FToLocalId() `std::vector<std::size_t> LocalVirtualDofs::FToLocalId (`
`const std::size_t & ID) const [inline]`

Get the corresponding local dof for face-type global dof.

Parameters

<i>ID</i>	
-----------	--

Returns

`std::vector<std::size_t>`

3.7.2.3 getDof() `template<typename DofType >`
`std::shared_ptr<DofType> LocalVirtualDofs::getDof (`
`std::size_t id) const [inline]`

Method to get the corresponding specialized dof to a given id.

Template Parameters

<i>DofType</i>	
----------------	--

Parameters

<i>id</i>	
-----------	--

Returns

`std::shared_ptr<DofType>`

3.7.2.4 getID() `std::size_t LocalVirtualDofs::getID (std::size_t id) const`

Method to get the global id of the corresponding local dof id.

Parameters

<i>id</i>	
-----------	--

Returns

`std::size_t`

3.7.2.5 getnumDofs() `std::size_t LocalVirtualDofs::getnumDofs () const`

Get the total number of local dofs.

Returns

`std::size_t`

3.7.2.6 getnumEdofs() `std::size_t LocalVirtualDofs::getnumEdofs () const`

Get the number of edge-type dofs.

Returns

`std::size_t`

3.7.2.7 getnumFdofs() `std::size_t LocalVirtualDofs::getnumFdofs () const`

Get the number of face-type dofs.

Returns

`std::size_t`

3.7.2.8 getnumPdofs() `std::size_t LocalVirtualDofs::getnumPdofs () const`

Get the number of polyhedron-type dofs.

Returns

`std::size_t`

3.7.2.9 getnumVdofs() `std::size_t LocalVirtualDofs::getnumVdofs () const`

Get the number of vertex-type dofs.

Returns

`std::size_t`

3.7.2.10 PToLocalId() `std::size_t LocalVirtualDofs::PToLocalId (
const std::size_t & ID) const [inline]`

Get the corresponding local dof for polyhedron-type global dof.

Parameters

<i>ID</i>	
-----------	--

Returns

`std::size_t`

3.7.2.11 VToLocalId() `std::size_t LocalVirtualDofs::VToLocalId (
const std::size_t & ID) const [inline]`

Get the corresponding local dof for vertex-type global dof.

Parameters

<i>ID</i>	
-----------	--

Returns

std::size_t

3.7.3 Friends And Related Function Documentation

3.7.3.1 operator<< std::ostream& operator<< (
 std::ostream & os,
 const LocalVirtualDofs & dofsCollection) [friend]

Output stream operator for LocalVirtualDofs.

Parameters

<i>os</i>	
<i>dofsCollection</i>	

Returns

std::ostream&

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

- include/virtualDofs.hpp
- src/virtualDofs.cpp

3.8 LocalVirtualDofsCollection Class Reference

Public Member Functions

- LocalVirtualDofsCollection (const Mesh< Point3D, Edge3D, Polygon3D, Polyhedron< Polygon3D >> &mesh, const VirtualDofsCollection &DOFS)
Constructor.
- const LocalVirtualDofs & getLocalDofs (const std::size_t &Id) const
Get the LocalVirtualDofs of the element Id.
- size_t numLocalDofsCollection () const
Get the number of LocalVirtualDofs in the collection.

3.8.1 Constructor & Destructor Documentation

3.8.1.1 LocalVirtualDofsCollection() LocalVirtualDofsCollection::LocalVirtualDofsCollection (
 const Mesh< Point3D, Edge3D, Polygon3D, Polyhedron< Polygon3D >> & mesh,
 const VirtualDofsCollection & DOFS)

Constructor.

Parameters

<i>mesh</i>	
<i>DOFS</i>	

3.8.2 Member Function Documentation

3.8.2.1 `getLocalDofs()` `const LocalVirtualDofs & LocalVirtualDofsCollection::getLocalDofs (`
`const std::size_t & Id) const`

Get the [LocalVirtualDofs](#) of the element *Id*.

Parameters

<i>Id</i>	
-----------	--

Returns

const [LocalVirtualDofs](#)&

3.8.2.2 `numLocalDofsCollection()` `size_t LocalVirtualDofsCollection::numLocalDofsCollection ()`
`const`

Get the number of [LocalVirtualDofs](#) in the collection.

Returns

size_t

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

- include/virtualDofs.hpp
- src/virtualDofs.cpp

3.9 `Mesh< PointType, EdgeType, PolygonType, PolyhedronType >` Class Template Reference**Public Member Functions**

- [Mesh](#) ()=default
Default constructor.
- [Mesh](#) (const std::string &filename)
Constructor reading entities from Gmsh .geo file.

- `std::size_t numVertices () const`
Method to get the number of vertices.
- `std::size_t numEdges () const`
Method to get the number of edges.
- `std::size_t numPolygons () const`
Method to get the number of polygons.
- `std::size_t numPolyhedra () const`
Method to get the number of polyhedra.
- `const PointType & getVertex (std::size_t index) const`
Getter for a vertex.
- `const EdgeType & getEdge (IndexType index) const`
Getter for an edge.
- `const PolygonType & getPolygon (IndexType index) const`
Getter for a polygon.
- `const PolyhedronType & getPolyhedron (std::size_t index) const`
Getter for a polyhedron.
- `const std::map< std::size_t, PointType > & getVertices () const`
Getter for the map of vertices.
- `const std::map< IndexType, EdgeType > & getEdges () const`
Getter for the map of edges.
- `const std::map< IndexType, PolygonType > & getPolygons () const`
Getter for the map of polygons.
- `const std::map< std::size_t, PolyhedronType > & getPolyhedra () const`
Getter for the map of polyhedra.
- `const real & getSize () const`
Getter for the average diameter.
- `void print () const`
Print mesh information.

3.9.1 Constructor & Destructor Documentation

3.9.1.1 Mesh() `template<typename PointType , typename EdgeType , typename PolygonType , typename PolyhedronType >`
`Mesh< PointType, EdgeType, PolygonType, PolyhedronType >::Mesh (`
`const std::string & filename) [inline]`

Constructor reading entities from Gmsh .geo file.

Parameters

<code>filename</code>	
-----------------------	--

3.9.2 Member Function Documentation

3.9.2.1 getEdge() `template<typename PointType , typename EdgeType , typename PolygonType ,
typename PolyhedronType >
const EdgeType& Mesh< PointType, EdgeType, PolygonType, PolyhedronType >::getEdge (
IndexType index) const [inline]`

Getter for an edge.

Parameters

<i>index</i>	
--------------	--

Returns

`const EdgeType&`

3.9.2.2 getEdges() `template<typename PointType , typename EdgeType , typename PolygonType ,
typename PolyhedronType >
const std::map<IndexType, EdgeType> Mesh< PointType, EdgeType, PolygonType, PolyhedronType
>::getEdges () const [inline]`

Getter for the map of edges.

Returns

`const std::map<IndexType, EdgeType>`

3.9.2.3 getPolygon() `template<typename PointType , typename EdgeType , typename PolygonType ,
typename PolyhedronType >
const PolygonType& Mesh< PointType, EdgeType, PolygonType, PolyhedronType >::getPolygon (
IndexType index) const [inline]`

Getter for a polygon.

Parameters

<i>index</i>	
--------------	--

Returns

`const PolygonType&`

3.9.2.4 getPolygons() `template<typename PointType , typename EdgeType , typename PolygonType ,
typename PolyhedronType >`

```
const std::map<IndexType, PolygonType> Mesh< PointType, EdgeType, PolygonType, PolyhedronType
>::getPolygons ( ) const [inline]
```

Getter for the map of polygons.

Returns

```
const std::map<IndexType, PolygonType>
```

```
3.9.2.5 getPolyhedra() template<typename PointType , typename EdgeType , typename PolygonType ,
typename PolyhedronType >
const std::map<std::size_t, PolyhedronType>& Mesh< PointType, EdgeType, PolygonType, PolyhedronType
>::getPolyhedra ( ) const [inline]
```

Getter for the map of polyhedra.

Returns

```
const std::map<std::size_t, PolyhedronType>&
```

```
3.9.2.6 getPolyhedron() template<typename PointType , typename EdgeType , typename PolygonType
, typename PolyhedronType >
const PolyhedronType& Mesh< PointType, EdgeType, PolygonType, PolyhedronType >::getPolyhedron
(
    std::size_t index ) const [inline]
```

Getter for a polyhedron.

Parameters

<i>index</i>	
--------------	--

Returns

```
const PolyhedronType&
```

```
3.9.2.7 getSize() template<typename PointType , typename EdgeType , typename PolygonType ,
typename PolyhedronType >
const real& Mesh< PointType, EdgeType, PolygonType, PolyhedronType >::getSize ( ) const [inline]
```

Getter for the average diameter.

Returns

```
const real&
```

3.9.2.8 getVertex() `template<typename PointType , typename EdgeType , typename PolygonType ,
typename PolyhedronType >
const PointType& Mesh< PointType, EdgeType, PolygonType, PolyhedronType >::getVertex (
std::size_t index) const [inline]`

Getter for a vertex.

Parameters

<i>index</i>	
--------------	--

Returns

`const PointType&`

3.9.2.9 getVertices() `template<typename PointType , typename EdgeType , typename PolygonType ,
typename PolyhedronType >
const std::map<std::size_t, PointType>& Mesh< PointType, EdgeType, PolygonType, PolyhedronType >::getVertices () const [inline]`

Getter for the map of vertices.

Returns

`const std::map<std::size_t, PointType>&`

3.9.2.10 numEdges() `template<typename PointType , typename EdgeType , typename PolygonType ,
typename PolyhedronType >
std::size_t Mesh< PointType, EdgeType, PolygonType, PolyhedronType >::numEdges () const
[inline]`

Method to get the number of edges.

Returns

`std::size_t`

3.9.2.11 numPolygons() `template<typename PointType , typename EdgeType , typename PolygonType
, typename PolyhedronType >
std::size_t Mesh< PointType, EdgeType, PolygonType, PolyhedronType >::numPolygons () const
[inline]`

Method to get the number of polygons.

Returns

`std::size_t`

3.9.2.12 numPolyhedra() `template<typename PointType , typename EdgeType , typename PolygonType , typename PolyhedronType >
std::size_t Mesh< PointType, EdgeType, PolygonType, PolyhedronType >::numPolyhedra () const
[inline]`

Method to get the number of polyhedra.

Returns

`std::size_t`

3.9.2.13 numVertices() `template<typename PointType , typename EdgeType , typename PolygonType , typename PolyhedronType >
std::size_t Mesh< PointType, EdgeType, PolygonType, PolyhedronType >::numVertices () const
[inline]`

Method to get the number of vertices.

Returns

`std::size_t`

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

- `include/mesh.hpp`

3.10 Monomial< Dimension > Class Template Reference

Public Member Functions

- [Monomial](#) ()
Default constructor.
- [Monomial](#) (std::vector< unsigned int > exponents, real coefficient)
Constructor.
- const std::vector< unsigned int > & [getExponents](#) () const
Getter for the exponents.
- real [getCoefficient](#) () const
Getter for the coefficient.
- void [setCoefficient](#) (real coeff)
Setter for the coefficient.
- [Monomial](#)< Dimension > [operator*](#) (const [Monomial](#)< Dimension > &other) const
Compute the product of two monomials.
- [Monomial](#)< Dimension > [derivative](#) (unsigned int variableIndex) const
Compute the derivative with respect to a variable.
- template<typename PointType >
real [evaluate](#) (const PointType &point) const
Evaluate the monomial at a point.
- unsigned int [getOrder](#) () const
Get the monomial order.

Friends

- `std::ostream & operator<< (std::ostream &os, const Monomial &monomial)`
Output stream operator.

3.10.1 Constructor & Destructor Documentation

3.10.1.1 Monomial() `template<unsigned int Dimension>`
`Monomial< Dimension >::Monomial (`
 `std::vector< unsigned int > exponents,`
 `real coefficient) [inline]`

Constructor.

Parameters

<i>exponents</i>	
<i>coefficient</i>	

3.10.2 Member Function Documentation

3.10.2.1 derivative() `template<unsigned int Dimension>`
`Monomial<Dimension> Monomial< Dimension >::derivative (`
 `unsigned int variableIndex) const [inline]`

Compute the derivative with respect to a variable.

Parameters

<i>variableIndex</i>	
----------------------	--

Returns

`Monomial<Dimension>`

3.10.2.2 evaluate() `template<unsigned int Dimension>`
`template<typename PointType >`
`real Monomial< Dimension >::evaluate (`
 `const PointType & point) const [inline]`

Evaluate the monomial at a point.

Template Parameters

<i>PointType</i>	
------------------	--

Parameters

<i>point</i>	
--------------	--

Returns

real

3.10.2.3 getCoefficient() `template<unsigned int Dimension>
real Monomial< Dimension >::getCoefficient () const [inline]`

Getter for the coefficient.

Returns

real

3.10.2.4 getExponents() `template<unsigned int Dimension>
const std::vector<unsigned int>& Monomial< Dimension >::getExponents () const [inline]`

Getter for the exponents.

Returns

const std::vector<unsigned int>&

3.10.2.5 getOrder() `template<unsigned int Dimension>
unsigned int Monomial< Dimension >::getOrder () const [inline]`

Get the monomial order.

Returns

unsigned int

3.10.2.6 operator*() `template<unsigned int Dimension>
Monomial<Dimension> Monomial< Dimension >::operator* (
const Monomial< Dimension > & other) const [inline]`

Compute the product of two monomials.

Parameters

<i>other</i>	
--------------	--

Returns

Monomial<Dimension>

3.10.2.7 setCoefficient() `template<unsigned int Dimension>`
`void Monomial< Dimension >::setCoefficient (`
 `real coeff) [inline]`

Setter for the coefficient.

Parameters

<i>coeff</i>	
--------------	--

3.10.3 Friends And Related Function Documentation

3.10.3.1 operator<< `template<unsigned int Dimension>`
`std::ostream& operator<< (`
 `std::ostream & os,`
 `const Monomial< Dimension > & monomial) [friend]`

Output stream operator.

Parameters

<i>os</i>	
<i>monomial</i>	

Returns

std::ostream&

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

- include/monomial.hpp

3.11 Monomial2D Class Reference

Inheritance diagram for Monomial2D:

Collaboration diagram for Monomial2D:

Public Member Functions

- [Monomial2D](#) ()
Default constructor.
- [Monomial2D](#) (unsigned int expX, unsigned int expY, double coeff)
Constructor with exponents and coefficient.
- [Monomial2D](#) (const [Monomial](#)< 2 > &monomial)
Constructor for implicit conversion from [Monomial](#)<2> to [Monomial2D](#).
- [Monomial2D operator*](#) (const [Monomial2D](#) &other) const
Method to compute the product of two monomials and return a new [Monomial2D](#) instance.
- [Monomial2D derivative](#) (unsigned int variableIndex) const
Method to compute the derivative with respect to a variable and return a new [Monomial2D](#) instance.
- [Monomial2D dx](#) () const
Compute the derivative with respect to x.
- [Monomial2D dy](#) () const
Compute the derivative with respect to y.

Static Public Member Functions

- static void [computeMonomialsUpToOrder](#) (unsigned int order_)
Method to compute the monomials up to a given degree.
- static const std::vector< [Monomial2D](#) > [getMonomialsOrdered](#) (unsigned int order_)
Method to get the monomials up to a given degree.
- static void [computeLaplaciansToMonomialsOrdered](#) (unsigned int order_)
Method to compute the laplacians up to a given degree.
- static const std::vector< std::pair< std::pair< real, std::size_t >, std::pair< real, std::size_t > > > [getLaplaciansToMonomialsOrdered](#) (unsigned int order_)
Method to get the laplacians up to a given degree.

Friends

- std::ostream & [operator<<](#) (std::ostream &os, const [Monomial2D](#) &monomial)
Overriding the output stream operator to print "x, y" for [Monomial2D](#).

3.11.1 Constructor & Destructor Documentation

3.11.1.1 [Monomial2D](#)() [1/2] [Monomial2D::Monomial2D](#) (
 unsigned int expX,
 unsigned int expY,
 double coeff) [inline]

Constructor with exponents and coefficient.

Parameters

<i>expX</i>	
<i>expY</i>	
<i>coeff</i>	

3.11.1.2 Monomial2D() [2/2] `Monomial2D::Monomial2D (`
`const Monomial< 2 > & monomial) [inline]`

Constructor for implicit conversion from [Monomial<2>](#) to [Monomial2D](#).

Parameters

<i>monomial</i>	
-----------------	--

3.11.2 Member Function Documentation

3.11.2.1 computeLaplaciansToMonomialsOrdered() `void Monomial2D::computeLaplaciansToMonomialsOrdered (`
`unsigned int order_) [static]`

Method to compute the laplacians up to a given degree.

Parameters

<i>order_↔</i>	
—	

3.11.2.2 computeMonomialsUpToOrder() `void Monomial2D::computeMonomialsUpToOrder (`
`unsigned int order_) [static]`

Method to compute the monomials up to a given degree.

Parameters

<i>order_↔</i>	
—	

3.11.2.3 derivative() `Monomial2D Monomial2D::derivative (`
`unsigned int variableIndex) const [inline]`

Method to compute the derivative with respect to a variable and return a new [Monomial2D](#) instance.

Parameters

<i>variableIndex</i>	
----------------------	--

Returns

[Monomial2D](#)

3.11.2.4 dx() [Monomial2D](#) Monomial2D::dx () const [inline]

Compute the derivative with respect to x.

Returns

[Monomial2D](#)

3.11.2.5 dy() [Monomial2D](#) Monomial2D::dy () const [inline]

Compute the derivative with respect to y.

Returns

[Monomial2D](#)

3.11.2.6 getLaplaciansToMonomialsOrdered() const std::vector< std::pair< std::pair< real, std::size_t >, std::pair< real, std::size_t > > > Monomial2D::getLaplaciansToMonomialsOrdered (unsigned int order_) [static]

Method to get the laplacians up to a given degree.

Parameters

<i>order_↵</i>	
—	

Returns

const std::vector<std::pair<std::pair<real, std::size_t>, std::pair<real, std::size_t>>>>

3.11.2.7 getMonomialsOrdered() const std::vector< [Monomial2D](#) > Monomial2D::getMonomialsOrdered (unsigned int order_) [static]

Method to get the monomials up to a given degree.

Parameters

<i>order</i> ↔	
—	

Returns

const std::vector<Monomial2D>

3.11.2.8 operator*() [Monomial2D](#) Monomial2D::operator* (
const [Monomial2D](#) & *other*) const [inline]

Method to compute the product of two monomials and return a new [Monomial2D](#) instance.

Parameters

<i>other</i>	
--------------	--

Returns

[Monomial2D](#)

3.11.3 Friends And Related Function Documentation

3.11.3.1 operator<< std::ostream& operator<< (
std::ostream & *os*,
const [Monomial2D](#) & *monomial*) [friend]

Overriding the output stream operator to print "x, y" for [Monomial2D](#).

Parameters

<i>os</i>	
<i>monomial</i>	

Returns

std::ostream&

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

- include/monomial.hpp
- src/monomial.cpp

3.12 Monomial3D Class Reference

Inheritance diagram for Monomial3D:

Collaboration diagram for Monomial3D:

Public Member Functions

- [Monomial3D](#) ()
Default constructor.
- [Monomial3D](#) (unsigned int expX, unsigned int expY, unsigned int expZ, double coeff)
Constructor with exponents and coefficient.
- [Monomial3D](#) (const [Monomial](#)< 3 > &monomial)
Constructor for implicit conversion from [Monomial](#)<3> to [Monomial3D](#).
- [Monomial3D operator*](#) (const [Monomial3D](#) &other) const
Method to compute the product of two monomials and return a new [Monomial2D](#) instance.
- [Monomial3D derivative](#) (unsigned int variableIndex) const
Method to compute the derivative with respect to a variable and return a new [Monomial2D](#) instance.
- [Monomial3D dx](#) () const
Compute the derivative with respect to x.
- [Monomial3D dy](#) () const
Compute the derivative with respect to y.
- [Monomial3D dz](#) () const
Compute the derivative with respect to z.

Static Public Member Functions

- static void [computeMonomialsUpToOrder](#) (unsigned int order_)
Method to compute the ordered monomials up to a given degree.
- static const std::vector< [Monomial3D](#) > [getMonomialsOrdered](#) (unsigned int order_)
Method to get the ordered monomials up to a given degree.
- static void [computeGradientsToMonomialsOrdered](#) (unsigned int order_)
Method to compute the gradients of the ordered monomials up to a given degree.
- static const std::vector< std::array< std::pair< real, std::size_t >, 3 > > [getGradientsToMonomialsOrdered](#) (unsigned int order_)
ethod to get the gradients of the ordered monomials up to a given degree
- static void [computeLaplaciansToMonomialsOrdered](#) (unsigned int order_)
Method to compute the laplacians of the ordered monomials up to a given degree.
- static const std::vector< std::array< std::pair< real, std::size_t >, 3 > > [getLaplaciansToMonomialsOrdered](#) (unsigned int order_)
Method to get the laplacians of the ordered monomials up to a given degree.

Friends

- std::ostream & [operator<<](#) (std::ostream &os, const [Monomial3D](#) &monomial)
Overriding the output stream operator to print "x, y, z" for [Monomial3D](#).

3.12.1 Constructor & Destructor Documentation

3.12.1.1 Monomial3D() [1/2] `Monomial3D::Monomial3D (`
 `unsigned int expX,`
 `unsigned int expY,`
 `unsigned int expZ,`
 `double coeff) [inline]`

Constructor with exponents and coefficient.

Parameters

<i>expX</i>	
<i>expY</i>	
<i>expZ</i>	
<i>coeff</i>	

3.12.1.2 Monomial3D() [2/2] `Monomial3D::Monomial3D (`
 `const Monomial< 3 > & monomial) [inline]`

Constructor for implicit conversion from [Monomial<3>](#) to [Monomial3D](#).

Parameters

<i>monomial</i>	
-----------------	--

3.12.2 Member Function Documentation

3.12.2.1 computeGradientsToMonomialsOrdered() `void Monomial3D::computeGradientsToMonomials↵`
`Ordered (`
 `unsigned int order_) [static]`

Method to compute the gradients of the ordered monomials up to a given degree.

Parameters

<i>order↵</i>	
<i>—</i>	

3.12.2.2 computeLaplaciansToMonomialsOrdered() `void Monomial3D::computeLaplaciansToMonomialsOrdered (unsigned int order_) [static]`

Method to compute the laplacians of the ordered monomials up to a given degree.

Parameters

<i>order</i> ↔	
—	

3.12.2.3 computeMonomialsUpToOrder() `void Monomial3D::computeMonomialsUpToOrder (unsigned int order_) [static]`

Method to compute the ordered monomials up to a given degree.

Parameters

<i>order</i> ↔	
—	

3.12.2.4 derivative() `Monomial3D Monomial3D::derivative (unsigned int variableIndex) const [inline]`

Method to compute the derivative with respect to a variable and return a new [Monomial2D](#) instance.

Parameters

<i>variableIndex</i>	
----------------------	--

Returns

[Monomial3D](#)

3.12.2.5 dx() `Monomial3D Monomial3D::dx () const [inline]`

Compute the derivative with respect to x.

Returns

[Monomial3D](#)

3.12.2.6 `dy()` [Monomial3D](#) `Monomial3D::dy () const [inline]`

Compute the derivative with respect to y.

Returns

[Monomial3D](#)

3.12.2.7 `dz()` [Monomial3D](#) `Monomial3D::dz () const [inline]`

Compute the derivative with respect to z.

Returns

[Monomial3D](#)

3.12.2.8 `getGradientsToMonomialsOrdered()` `const std::vector< std::array< std::pair< real, std::size_t >, 3 > > Monomial3D::getGradientsToMonomialsOrdered (unsigned int order_) [static]`

ethod to get the gradients of the ordered monomials up to a given degree

Parameters

<i>order</i> ↔	
—	

Returns

`const std::vector<std::array<std::pair<real, std::size_t>, 3>>`

3.12.2.9 `getLaplaciansToMonomialsOrdered()` `const std::vector< std::array< std::pair< real, std::size_t >, 3 > > Monomial3D::getLaplaciansToMonomialsOrdered (unsigned int order_) [static]`

Method to get the laplacians of the ordered monomials up to a given degree.

Parameters

<i>order</i> ↔	
—	

Returns

```
const std::vector<std::array<std::pair<real, std::size_t>, 3>>
```

3.12.2.10 getMonomialsOrdered() `const std::vector< Monomial3D > Monomial3D::getMonomialsOrdered (unsigned int order_) [static]`

Method to get the ordered monomials up to a given degree.

Parameters

<i>order_↔</i>	
—	

Returns

```
const std::vector<Monomial3D>
```

3.12.2.11 operator*() `Monomial3D Monomial3D::operator* (const Monomial3D & other) const [inline]`

Method to compute the product of two monomials and return a new [Monomial2D](#) instance.

Parameters

<i>other</i>	
--------------	--

Returns

[Monomial3D](#)

3.12.3 Friends And Related Function Documentation

3.12.3.1 operator<< `std::ostream& operator<< (std::ostream & os, const Monomial3D & monomial) [friend]`

Overriding the output stream operator to print "x, y, z" for [Monomial3D](#).

Parameters

<i>os</i>	
<i>monomial</i>	

Returns

std::ostream&

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

- include/monomial.hpp
- src/monomial.cpp

3.13 IntegrationMonomial::MonomialsFaceIntegralsCache Class Reference

Static Public Member Functions

- static void [initialize](#) (const [Mesh](#)< [Point3D](#), [Edge3D](#), [Polygon3D](#), [Polyhedron](#)< [Polygon3D](#) >> &mesh, unsigned int order)
Initialize the cache for the whole faces in the mesh.
- static void [initialize](#) (const [Polygon3D](#) &F, unsigned int order)
Initialize the cache for a face.
- static std::vector< real > & [getCacheMonomials](#) (const [Polygon3D](#) &F, unsigned int order)
Get the integrals of monomials up to a given order over a face.
- static real & [getCacheMonomial](#) (const [Polygon3D](#) &F, const [Monomial2D](#) &m)
Get the integrals of monomials up to a given order over a face.

3.13.1 Member Function Documentation

3.13.1.1 [getCacheMonomial\(\)](#) real & IntegrationMonomial::MonomialsFaceIntegralsCache::getCacheMonomial (const [Polygon3D](#) & F, const [Monomial2D](#) & m) [static]

Get the integrals of monomials up to a given order over a face.

Parameters

<i>F</i>	
<i>m</i>	

Returns

real&

3.13.1.2 [getCacheMonomials\(\)](#) std::vector< real > & IntegrationMonomial::MonomialsFaceIntegralsCache::getCacheMonomials (

```
const Polygon3D & F,  
unsigned int order ) [static]
```

Get the integrals of monomials up to a given order over a face.

Parameters

<i>F</i>	
<i>order</i>	

Returns

std::vector<real>&

3.13.1.3 initialize() [1/2] `void IntegrationMonomial::MonomialsFaceIntegralsCache::initialize (`
 `const Mesh< Point3D, Edge3D, Polygon3D, Polyhedron< Polygon3D >> & mesh,`
 `unsigned int order) [static]`

Initialize the cache for the whole faces in the mesh.

Parameters

<i>mesh</i>	
<i>order</i>	

3.13.1.4 initialize() [2/2] `void IntegrationMonomial::MonomialsFaceIntegralsCache::initialize (`
 `const Polygon3D & F,`
 `unsigned int order) [static]`

Initialize the cache for a face.

Parameters

<i>F</i>	
<i>order</i>	

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

- include/integration.hpp
- src/integration.cpp

3.14 Parameters Class Reference

Public Member Functions

- [Parameters](#) (const char *parametersFilename)
 Constructor.
- const std::string & [getInputMesh](#) () const
 Get the mesh string name.

- `const std::vector< std::function< real(real, real, real)> > & getHomogeneousDirichletBC () const`
Get the homogeneous dirichlet boundary conditions.
- `const std::function< std::array< real, 3 >real, real, real)> & getExactSolution () const`
Get the exact displacement solution.
- `const std::function< std::array< real, 6 >real, real, real)> & getExactStrains () const`
Get the exact strains.
- `const std::function< std::array< real, 3 >real, real, real)> & getForcing () const`
Get the forcing term.
- `const unsigned int & getOrder () const`
Get the order.
- `const bool & getInitializeFaceIntegrals () const`
Get the bool returning true if face integrals are required to be initialized.
- `const real & getYoungsModulus () const`
Get Young's modulus.
- `const real & getPoissonRatio () const`
Get Poisson's ratio.
- `const real & getFirstLame () const`
Get first Lamé's parameter.
- `const real & getSecondLame () const`
Get second Lamé's parameter.
- `void print () const`
Print parameter information.

3.14.1 Constructor & Destructor Documentation

3.14.1.1 Parameters() `Parameters::Parameters (const char * parametersFilename)`

Constructor.

Parameters

<code>parametersFilename</code>	
---------------------------------	--

3.14.2 Member Function Documentation

3.14.2.1 getExactSolution() `const std::function<std::array<real, 3>real, real, real)>& Parameters::getExactSolution () const [inline]`

Get the exact displacement solution.

Returns

`const std::function<std::array<real, 3>(real, real, real)>&`

3.14.2.2 getExactStrains() `const std::function<std::array<real, 6>real, real, real)>& Parameters::getExactStrains () const [inline]`

Get the exact strains.

Returns

`const std::function<std::array<real, 6>(real, real, real)>&`

3.14.2.3 getFirstLame() `const real& Parameters::getFirstLame () const [inline]`

Get first Lamé's parameter.

Returns

`const real&`

3.14.2.4 getForcing() `const std::function<std::array<real, 3>real, real, real)>& Parameters::getForcing () const [inline]`

Get the forcing term.

Returns

`const std::function<std::array<real, 3>(real, real, real)>&`

3.14.2.5 getHomogeneousDirichletBC() `const std::vector<std::function<real(real, real, real)>>& Parameters::getHomogeneousDirichletBC () const [inline]`

Get the homogeneous dirichlet boundary conditions.

Returns

`const std::vector<std::function<real(real, real, real)>>&`

3.14.2.6 getInitializeFaceIntegrals() `const bool& Parameters::getInitializeFaceIntegrals () const [inline]`

Get the bool returning true if face integrals are required to be initialized.

Returns

`true`

`false`

3.14.2.7 getInputMesh() `const std::string& Parameters::getInputMesh () const [inline]`

Get the mesh string name.

Returns

`const std::string&`

3.14.2.8 getOrder() `const unsigned int& Parameters::getOrder () const [inline]`

Get the order.

Returns

`const unsigned int&`

3.14.2.9 getPoissonRatio() `const real& Parameters::getPoissonRatio () const [inline]`

Get Poisson's ratio.

Returns

`const real&`

3.14.2.10 getSecondLame() `const real& Parameters::getSecondLame () const [inline]`

Get second Lame's parameter.

Returns

`const real&`

3.14.2.11 getYoungsModulus() `const real& Parameters::getYoungsModulus () const [inline]`

Get Young's modulus.

Returns

`const real&`

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

- include/parameters.hpp
- src/parameters.cpp

3.15 geometry::Point< Args > Class Template Reference

Public Member Functions

- [Point](#) ()
Default constructor to initialize coordinates to 0.
- [Point](#) (Args... args)
Construct a new [Point](#) object.
- [~Point](#) ()
Destructor to free the id when the point goes out of scope.
- constexpr std::size_t [getDimension](#) () const
- void [setId](#) (IndexType _id)
Set the Id object.
- const IndexType & [getId](#) () const
Get the Id object.
- const std::array< real, sizeof...(Args)> [getCoordinates](#) () const
Get the Coordinates object.
- const real & [operator\[\]](#) (std::size_t index) const
Access operator [].
- auto [operator*](#) (const real &scalar) const
*Overloaded * operator to compute the scalar multiplication of a point (point*scalar)*
- template<size_t... Indices>
auto [multiplyByScalar](#) (real scalar, std::index_sequence< Indices... >) const
scalar multiplication of a point
- auto [operator/](#) (const real &scalar) const
Overloaded / operator to compute the scalar division of a point (point/scalar)
- template<typename... OtherArgs>
auto [operator+](#) (const [Point](#)< OtherArgs... > &other) const
Overloaded + operator to compute the sum of two points.
- template<typename... OtherArgs>
auto [operator-](#) (const [Point](#)< OtherArgs... > &other) const
Overloaded - operator to compute the difference of two points.
- template<typename... OtherArgs>
auto [piecewiseMultiply](#) (const [Point](#)< OtherArgs... > &other) const
Piecewise multiplication of two points.
- template<typename... OtherArgs, size_t... Indices, typename Operation >
auto [binaryOperation](#) (const [Point](#)< OtherArgs... > &other, Operation operation, std::index_sequence< Indices... >) const
Binary operation in class.
- template<typename... OtherArgs>
auto [dot](#) (const [Point](#)< OtherArgs... > &other) const
Dot product in the form point1.dot(point2)
- template<typename... OtherArgs>
auto [cross](#) (const [Point](#)< OtherArgs... > &other) const
In-class cross product calculation (3D points only) in the form point1.cross(point2)
- template<typename... OtherArgs>
auto [distance](#) (const [Point](#)< OtherArgs... > &other) const
Euclidean distance.
- auto [norm](#) () const
Compute the norm.
- auto [normalize](#) () const
Normalize coordinates.

- bool `operator<` (const `Point`< Args... > &other) const
Define the comparison function based on edge ids.
- template<typename... OtherArgs>
bool `operator==` (const `Point`< OtherArgs... > &other) const
Custom definition of operator== for `Point`.
- template<typename... OtherArgs>
bool `operator!=` (const `Point`< OtherArgs... > &other) const

Friends

- auto `operator*` (const real &scalar, const `Point` &point)
*Overloaded * operator to support scalar * point multiplication.*
- std::ostream & `operator<<` (std::ostream &os, const `Point`< Args... > &point)
Output stream operator to stream coordinates.

3.15.1 Constructor & Destructor Documentation

3.15.1.1 `Point()` template<typename... Args>
`geometry::Point`< Args >::`Point` (
 Args... args) [inline]

Construct a new `Point` object.

Parameters

<i>args</i>	coordinates of the point
-------------	--------------------------

3.15.2 Member Function Documentation

3.15.2.1 `binaryOperation()` template<typename... Args>
 template<typename... OtherArgs, size_t... Indices, typename Operation >
 auto `geometry::Point`< Args >::`binaryOperation` (
 const `Point`< OtherArgs... > & other,
 Operation operation,
 std::index_sequence< Indices... >) const [inline]

Binary operation in class.

Template Parameters

<i>OtherArgs</i>	
<i>Indices</i>	
<i>Operation</i>	

Parameters

<i>other</i>	other point
<i>operation</i>	

Returns

auto

```
3.15.2.2 cross()  template<typename...  Args>
template<typename...  OtherArgs>
auto geometry::Point< Args >::cross (
    const Point< OtherArgs... > & other ) const  [inline]
```

In-class cross product calculation (3D points only) in the form point1.cross(point2)

Template Parameters

<i>OtherArgs</i>	
------------------	--

Parameters

<i>other</i>	other point
--------------	-------------

Returns

auto

```
3.15.2.3 distance()  template<typename...  Args>
template<typename...  OtherArgs>
auto geometry::Point< Args >::distance (
    const Point< OtherArgs... > & other ) const  [inline]
```

Euclidean distance.

Template Parameters

<i>OtherArgs</i>	
------------------	--

Parameters

<i>other</i>	other point
--------------	-------------

Returns

auto

```
3.15.2.4 dot() template<typename... Args>
template<typename... OtherArgs>
auto geometry::Point< Args >::dot (
    const Point< OtherArgs... > & other ) const [inline]
```

Dot product in the form point1.dot(point2)

Template Parameters

<i>OtherArgs</i>	
------------------	--

Parameters

<i>other</i>	other point
--------------	-------------

Returns

auto

```
3.15.2.5 getCoordinates() template<typename... Args>
const std::array<real, sizeof...(Args)> geometry::Point< Args >::getCoordinates ( ) const
[inline]
```

Get the Coordinates object.

Returns

const std::array<real, sizeof...(Args)>

```
3.15.2.6 getId() template<typename... Args>
const IndexType& geometry::Point< Args >::getId ( ) const [inline]
```

Get the Id object.

Returns

const IndexType&

```
3.15.2.7 multiplyByScalar() template<typename... Args>
template<size_t... Indices>
auto geometry::Point< Args >::multiplyByScalar (
    real scalar,
    std::index_sequence< Indices... > ) const [inline]
```

scalar multiplication of a point

Template Parameters

<i>Indices</i>	
----------------	--

Parameters

<i>scalar</i>	
---------------	--

Returns

auto

3.15.2.8 norm() `template<typename... Args>
auto geometry::Point< Args >::norm () const [inline]`

Compute the norm.

Returns

auto

3.15.2.9 normalize() `template<typename... Args>
auto geometry::Point< Args >::normalize () const [inline]`

Normalize coordinates.

Returns

auto

3.15.2.10 operator*() `template<typename... Args>
auto geometry::Point< Args >::operator* (
const real & scalar) const [inline]`

Overloaded * operator to compute the scalar multiplication of a point (point*scalar)

Parameters

<i>scalar</i>	
---------------	--

Returns

auto

```
3.15.2.11 operator+() template<typename... Args>
template<typename... OtherArgs>
auto geometry::Point< Args >::operator+ (
    const Point< OtherArgs... > & other ) const [inline]
```

Overloaded + operator to compute the sum of two points.

Template Parameters

<i>OtherArgs</i>	
------------------	--

Parameters

<i>other</i>	other point
--------------	-------------

Returns

auto

```
3.15.2.12 operator-() template<typename... Args>
template<typename... OtherArgs>
auto geometry::Point< Args >::operator- (
    const Point< OtherArgs... > & other ) const [inline]
```

Overloaded - operator to compute the difference of two points.

Template Parameters

<i>OtherArgs</i>	
------------------	--

Parameters

<i>other</i>	other point
--------------	-------------

Returns

auto

3.15.2.13 operator/() `template<typename... Args>`
`auto geometry::Point< Args >::operator/ (`
`const real & scalar) const [inline]`

Overloaded / operator to compute the scalar division of a point (point/scalar)

Parameters

<i>scalar</i>	
---------------	--

Returns

auto

3.15.2.14 operator<() `template<typename... Args>`
`bool geometry::Point< Args >::operator< (`
`const Point< Args... > & other) const [inline]`

Define the comparison function based on edge lds.

Parameters

<i>other</i>	other point
--------------	-------------

Returns

true

false

3.15.2.15 operator==() `template<typename... Args>`
`template<typename... OtherArgs>`
`bool geometry::Point< Args >::operator== (`
`const Point< OtherArgs... > & other) const [inline]`

Custom definition of operator== for [Point](#).

Template Parameters

<i>OtherArgs</i>	
------------------	--

Parameters

<i>other</i>	other point
--------------	-------------

Returns

true
false

3.15.2.16 operator[]() `template<typename... Args>
const real& geometry::Point< Args >::operator[] (
std::size_t index) const [inline]`

Access operator [].

Parameters

<i>index</i>	index i corresponding to coordinate i
--------------	---------------------------------------

Returns

const real& coordinate i

3.15.2.17 piecewiseMultiply() `template<typename... Args>
template<typename... OtherArgs>
auto geometry::Point< Args >::piecewiseMultiply (
const Point< OtherArgs... > & other) const [inline]`

Piecewise multiplication of two points.

Template Parameters

<i>OtherArgs</i>	
------------------	--

Parameters

<i>other</i>	other point
--------------	-------------

Returns

auto

3.15.2.18 setId() `template<typename... Args>
void geometry::Point< Args >::setId (
IndexType _id) [inline]`

Set the Id object.

Parameters

↔	id
↔ _↔ id	

3.15.3 Friends And Related Function Documentation

3.15.3.1 operator* `template<typename... Args>`
`auto operator* (`
 `const real & scalar,`
 `const Point< Args > & point) [friend]`

Overloaded * operator to support scalar * point multiplication.

Parameters

<i>scalar</i>	
<i>point</i>	

Returns

`auto`

3.15.3.2 operator<< `template<typename... Args>`
`std::ostream& operator<< (`
 `std::ostream & os,`
 `const Point< Args... > & point) [friend]`

Output stream operator to stream coordinates.

Parameters

<i>os</i>	
<i>point</i>	

Returns

`std::ostream&`

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

- `include/point.hpp`

3.16 geometry::Polygon< EdgeType > Class Template Reference

Public Member Functions

- [Polygon](#) ()
Construct a new [Polygon](#) object.
- [Polygon](#) (const std::initializer_list< EdgeType > &edges_, bool _orientation=false)
Constructor taking individual edges.
- void [setOtherPolygon](#) (const [Polygon](#)< EdgeType > &otherPolygon_)
Setter method to set the other polygon.
- [Polygon](#)< EdgeType > & [getOtherPolygon](#) () const
Getter method to get the other polygon.
- void [setOrientation](#) (bool _orientation)
Set orientation.
- void [addEdge](#) (const EdgeType &edge)
Add an edge and its direction to the polygon.
- void [setId](#) (IndexType _id)
Set id.
- const IndexType & [getId](#) () const
Get id.
- std::size_t [numEdges](#) () const
Get the number of edges in the polygon.
- const EdgeType & [getEdge](#) (std::size_t index) const
Get an edge by index.
- const EdgeType & [operator\[\]](#) (IndexType index) const
Access edges through [].
- const EdgeType & [getPositiveEdge](#) (std::size_t index) const
Get original [Edge](#).
- bool [areEdgesConsistent](#) () const
Check if the edges are stored consistently.
- void [computeProperties](#) ()
Compute properties.
- void [computeOutwardNormalArea](#) ()
Compute outward normal unit vector.
- const [Point3D](#) [getOutwardNormal](#) () const
Get outward normal unit vector.
- const [Point3D](#) [get_e_x](#) () const
Get first local axis e_x .
- const [Point3D](#) [get_e_y](#) () const
Get second local axis e_y .
- real [getArea](#) () const
Get area.
- void [computeDiameter](#) ()
Compute diameter.
- real [getDiameter](#) () const
Get diameter.
- bool [operator<](#) (const [Polygon](#)< EdgeType > &other) const
Define the comparison function based on polygon Ids.
- bool [operator==](#) (const [Polygon](#)< EdgeType > &other) const
Comparison operator for polygons (==)

Friends

- `std::ostream & operator<< (std::ostream &os, const Polygon< EdgeType > &polygon)`
Stream output operator for the [Polygon](#) class.

3.16.1 Constructor & Destructor Documentation

3.16.1.1 Polygon() `template<typename EdgeType >
geometry::Polygon< EdgeType >::Polygon (
 const std::initializer_list< EdgeType > & edges_,
 bool _orientation = false) [inline]`

Constructor taking individual edges.

Parameters

<i>edges_</i>	
<i>_orientation</i>	

3.16.2 Member Function Documentation

3.16.2.1 addEdge() `template<typename EdgeType >
void geometry::Polygon< EdgeType >::addEdge (
 const EdgeType & edge) [inline]`

Add an edge and its direction to the polygon.

Parameters

<i>edge</i>	
-------------	--

3.16.2.2 areEdgesConsistent() `template<typename EdgeType >
bool geometry::Polygon< EdgeType >::areEdgesConsistent () const [inline]`

Check if the edges are stored consistently.

Returns

true
false

3.16.2.3 get_e_x() `template<typename EdgeType >``const Point3D geometry::Polygon< EdgeType >::get_e_x () const [inline]`

Get first local axis e_x.

Returns

const Point3D

3.16.2.4 get_e_y() `template<typename EdgeType >``const Point3D geometry::Polygon< EdgeType >::get_e_y () const [inline]`

Get second local axis e_y.

Returns

const Point3D

3.16.2.5 getArea() `template<typename EdgeType >``real geometry::Polygon< EdgeType >::getArea () const [inline]`

Get area.

Returns

real

3.16.2.6 getDiameter() `template<typename EdgeType >``real geometry::Polygon< EdgeType >::getDiameter () const [inline]`

Get diameter.

Returns

real

3.16.2.7 getEdge() `template<typename EdgeType >``const EdgeType& geometry::Polygon< EdgeType >::getEdge (
std::size_t index) const [inline]`

Get an edge by index.

Parameters

<i>index</i>	
--------------	--

Returns

const EdgeType&

3.16.2.8 getId() `template<typename EdgeType >
const IndexType& geometry::Polygon< EdgeType >::getId () const [inline]`

Get id.

Returns

const IndexType&

3.16.2.9 getOtherPolygon() `template<typename EdgeType >
Polygon<EdgeType>& geometry::Polygon< EdgeType >::getOtherPolygon () const [inline]`

Getter method to get the other polygon.

Returns

Polygon<EdgeType>&

3.16.2.10 getOutwardNormal() `template<typename EdgeType >
const Point3D geometry::Polygon< EdgeType >::getOutwardNormal () const [inline]`

Get outward normal unit vector.

Returns

const Point3D

3.16.2.11 getPositiveEdge() `template<typename EdgeType >
const EdgeType& geometry::Polygon< EdgeType >::getPositiveEdge (
std::size_t index) const [inline]`

Get original [Edge](#).

Parameters

<i>index</i>	
--------------	--

Returns

const EdgeType&

3.16.2.12 numEdges() `template<typename EdgeType >`
`std::size_t geometry::Polygon< EdgeType >::numEdges () const [inline]`

Get the number of edges in the polygon.

Returns

std::size_t

3.16.2.13 operator<() `template<typename EdgeType >`
`bool geometry::Polygon< EdgeType >::operator< (`
`const Polygon< EdgeType > & other) const [inline]`

Define the comparison function based on polygon ids.

Parameters

<i>other</i>	
--------------	--

Returns

true

false

3.16.2.14 operator==() `template<typename EdgeType >`
`bool geometry::Polygon< EdgeType >::operator==(`
`const Polygon< EdgeType > & other) const [inline]`

Comparison operator for polygons (==)

Parameters

<i>other</i>	
--------------	--

Returns

true
false

3.16.2.15 operator[]() `template<typename EdgeType >
const EdgeType& geometry::Polygon< EdgeType >::operator[] (
IndexType index) const [inline]`

Access edges through [].

Parameters

<i>index</i>	
--------------	--

Returns

const EdgeType&

3.16.2.16 setId() `template<typename EdgeType >
void geometry::Polygon< EdgeType >::setId (
IndexType _id) [inline]`

Set id.

Parameters

↩	
↩	
<i>id</i>	

3.16.2.17 setOrientation() `template<typename EdgeType >
void geometry::Polygon< EdgeType >::setOrientation (
bool _orientation) [inline]`

Set orientation.

Parameters

<i>_orientation</i>	
---------------------	--

3.16.2.18 setOtherPolygon() `template<typename EdgeType >`
`void geometry::Polygon< EdgeType >::setOtherPolygon (`
`const Polygon< EdgeType > & otherPolygon_) [inline]`

Setter method to set the other polygon.

Parameters

<i>other</i> ↔	
<i>Polygon_</i>	

3.16.3 Friends And Related Function Documentation

3.16.3.1 operator<< `template<typename EdgeType >`
`std::ostream& operator<< (`
`std::ostream & os,`
`const Polygon< EdgeType > & polygon) [friend]`

Stream output operator for the [Polygon](#) class.

Parameters

<i>os</i>	
<i>polygon</i>	

Returns

`std::ostream&`

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

- `include/polygon.hpp`

3.17 geometry::Polyhedron< PolygonType > Class Template Reference

Public Member Functions

- [Polyhedron](#) (`const std::initializer_list< PolygonType > &polygonsWithoutDirection`)
Constructor taking individual polygons.
- void [addPolygon](#) (`const PolygonType &polygon`)
Add a polygon and its direction to the polyhedron.
- void [setId](#) (`std::size_t id`)
Set id.
- `const std::size_t & getId () const`
Get id.
- `std::size_t numPolygons () const`

- *Method to get the number of polygons in the [Polyhedron](#).*
 • const PolygonType & [getPolygon](#) (std::size_t index) const
Method to get a polygon by index.
- const PolygonType & [operator\[\]](#) (std::size_t index) const
Access operator [] to get a polygon by index.
- void [computeDiameter](#) ()
Compute diameter.
- real [getDiameter](#) () const
Get diameter.

Friends

- std::ostream & [operator<<](#) (std::ostream &os, const [Polyhedron](#)< PolygonType > &polyhedron)
Stream output operator for the [Polyhedron](#) class.

3.17.1 Constructor & Destructor Documentation

3.17.1.1 Polyhedron() `template<typename PolygonType >
geometry::Polyhedron< PolygonType >::Polyhedron (
 const std::initializer_list< PolygonType > & polygonsWithoutDirection) [inline]`

Constructor taking individual polygons.

Parameters

<i>polygonsWithoutDirection</i>	
---------------------------------	--

3.17.2 Member Function Documentation

3.17.2.1 addPolygon() `template<typename PolygonType >
 void geometry::Polyhedron< PolygonType >::addPolygon (
 const PolygonType & polygon) [inline]`

Add a polygon and its direction to the polyhedron.

Parameters

<i>polygon</i>	
----------------	--

3.17.2.2 getDiameter() `template<typename PolygonType >`

```
real geometry::Polyhedron< PolygonType >::getDiameter ( ) const [inline]
```

Get diameter.

Returns

real

3.17.2.3 getId() `template<typename PolygonType >`

```
const std::size_t& geometry::Polyhedron< PolygonType >::getId ( ) const [inline]
```

Get id.

Returns

const std::size_t&

3.17.2.4 getPolygon() `template<typename PolygonType >`

```
const PolygonType& geometry::Polyhedron< PolygonType >::getPolygon (
    std::size_t index ) const [inline]
```

Method to get a polygon by index.

Parameters

<i>index</i>	
--------------	--

Returns

const PolygonType&

3.17.2.5 numPolygons() `template<typename PolygonType >`

```
std::size_t geometry::Polyhedron< PolygonType >::numPolygons ( ) const [inline]
```

Method to get the number of polygons in the [Polyhedron](#).

Returns

std::size_t

3.17.2.6 operator[]() `template<typename PolygonType >`

```
const PolygonType& geometry::Polyhedron< PolygonType >::operator[] (
    std::size_t index ) const [inline]
```

Access operator [] to get a polygon by index.

Parameters

<i>index</i>	
--------------	--

Returns

const PolygonType&

```
3.17.2.7 setid()  template<typename PolygonType >
void geometry::Polyhedron< PolygonType >::setId (
    std::size_t _id )  [inline]
```

Set id.

Parameters

↩	
↩	
<i>id</i>	

3.17.3 Friends And Related Function Documentation

```
3.17.3.1 operator<<  template<typename PolygonType >
std::ostream& operator<< (
    std::ostream & os,
    const Polyhedron< PolygonType > & polyhedron )  [friend]
```

Stream output operator for the [Polyhedron](#) class.

Parameters

<i>os</i>	
<i>polyhedron</i>	

Returns

std::ostream&

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

- include/polyhedron.hpp

3.18 PolyhedronDof Class Reference

Inheritance diagram for PolyhedronDof:

Collaboration diagram for PolyhedronDof:

Public Member Functions

- [PolyhedronDof](#) (std::size_t id, const [Monomial3D](#) &monomial_)
Constructor.
- std::size_t [getId](#) () const override
Getter for the id of the polyhedron.
- const [Monomial3D](#) & [getMonomial](#) () const
Getter for the monomial of the polyhedron DOF.
- std::ostream & [operator<<](#) (std::ostream &os) const override
Output stream operator.

3.18.1 Constructor & Destructor Documentation

3.18.1.1 PolyhedronDof() `PolyhedronDof::PolyhedronDof (std::size_t id, const Monomial3D & monomial_) [inline]`

Constructor.

Parameters

<i>id</i>	
<i>monomial_{<i>k</i>}</i>	
<i>_</i>	

3.18.2 Member Function Documentation

3.18.2.1 getId() `std::size_t PolyhedronDof::getId () const [override], [virtual]`

Getter for the id of the polyhedron.

Returns

std::size_t

Reimplemented from [VirtualDof](#).

3.18.2.2 getMonomial() `const Monomial3D & PolyhedronDof::getMonomial () const`

Getter for the monomial of the polyhedron DOF.

Returns

const [Monomial3D](#)&

3.18.2.3 operator<<() `std::ostream& PolyhedronDof::operator<< (std::ostream & os) const [inline], [override], [virtual]`

Output stream operator.

Parameters

<i>os</i>	
-----------	--

Returns

std::ostream&

Implements [VirtualDof](#).

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

- include/virtualDofs.hpp
- src/virtualDofs.cpp

3.19 Polynomial< Dimension > Class Template Reference

Public Member Functions

- [Polynomial](#) ()=default
Default constructor.
- [Polynomial](#) (const std::vector< [Monomial](#)< Dimension >> &monomials)
Constructor.
- void [addMonomial](#) (const [Monomial](#)< Dimension > &monomial)
Method to add a monomial to the polynomial.
- unsigned int [getOrder](#) () const
Get the polynomial order.
- [Polynomial](#)< Dimension > [operator*](#) (const [Polynomial](#)< Dimension > &other) const
*Overload * operator to compute the product of two polynomials.*
- [Polynomial](#)< Dimension > [operator*](#) (const [Monomial](#)< Dimension > &other) const
*Overload * operator to compute the product of a polynomial and a monomial.*
- size_t [size](#) () const
get the number of monomials making up the polynomial
- const std::map< std::vector< unsigned int >, [Monomial](#)< Dimension > > & [getPolynomial](#) () const
Getter for the polynomial.

Protected Attributes

- `std::map< std::vector< unsigned int >, Monomial< Dimension > > polynomial`
- unsigned int **order** = 0

3.19.1 Constructor & Destructor Documentation

3.19.1.1 Polynomial() `template<unsigned int Dimension>`
`Polynomial< Dimension >::Polynomial (`
`const std::vector< Monomial< Dimension >> & monomials) [inline]`

Constructor.

Parameters

<i>monomials</i>	
------------------	--

3.19.2 Member Function Documentation

3.19.2.1 addMonomial() `template<unsigned int Dimension>`
`void Polynomial< Dimension >::addMonomial (`
`const Monomial< Dimension > & monomial) [inline]`

Method to add a monomial to the polynomial.

Parameters

<i>monomial</i>	
-----------------	--

3.19.2.2 getOrder() `template<unsigned int Dimension>`
`unsigned int Polynomial< Dimension >::getOrder () const [inline]`

Get the polynomial order.

Returns

unsigned int

3.19.2.3 getPolynomial() `template<unsigned int Dimension>`
`const std::map<std::vector<unsigned int>, Monomial<Dimension> >& Polynomial< Dimension >↵↵`
`::getPolynomial () const [inline]`

Getter for the polynomial.

Returns

`const std::map<std::vector<unsigned int>, Monomial<Dimension>>&`

3.19.2.4 operator*() `[1/2] template<unsigned int Dimension>`
`Polynomial<Dimension> Polynomial< Dimension >::operator* (`
`const Monomial< Dimension > & other) const [inline]`

Overload * operator to compute the product of a polynomial and a monomial.

Parameters

<i>other</i>	
--------------	--

Returns

`Polynomial<Dimension>`

3.19.2.5 operator*() `[2/2] template<unsigned int Dimension>`
`Polynomial<Dimension> Polynomial< Dimension >::operator* (`
`const Polynomial< Dimension > & other) const [inline]`

Overload * operator to compute the product of two polynomials.

Parameters

<i>other</i>	
--------------	--

Returns

`Polynomial<Dimension>`

3.19.2.6 size() `template<unsigned int Dimension>`
`size_t Polynomial< Dimension >::size () const [inline]`

get the number of monomials making up the polynomial

Returns

size_t

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

- include/monomial.hpp

3.20 Problem Class Reference

Public Member Functions

- [Problem](#) (const char *parametersFilename="parameters.dat", bool printError=false)
Constructor.

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

- include/problem.hpp
- src/problem.cpp

3.21 Solver Class Reference

Inheritance diagram for Solver:

Public Member Functions

- [Solver](#) ()
Default constructor.
- [Solver](#) (const Eigen::SparseMatrix< real > &K, const Eigen::VectorXd &F)
Constructor.
- void [solve](#) ()
Solve the system $KU=F$.
- const Eigen::VectorXd & [getRightHandSide](#) () const
Getter for F.
- const Eigen::VectorXd & [getSolutionDisplacementsEig](#) () const
Getter for U as Eigen::Vector.
- std::vector< real > [getSolutionDisplacements](#) () const
Getter for U as std::vector.
- real **computeH1error** (const [Mesh](#)< [Point3D](#), [Edge3D](#), [Polygon3D](#), [Polyhedron](#)< [Polygon3D](#) >> &mesh, const std::function< std::array< real, 3 >(real, real, real)> &Uex_func)

Protected Attributes

- Eigen::SparseMatrix< real > **K**
- Eigen::VectorXd **F**
- Eigen::VectorXd **U**
- std::vector< bool > **isConstrained**
- std::vector< std::size_t > **unconstrainedDofs**
- std::vector< std::size_t > **constrainedDofs**
- std::vector< real > **constrainedDofsValues**

3.21.1 Constructor & Destructor Documentation

3.21.1.1 Solver() `Solver::Solver (`
 `const Eigen::SparseMatrix< real > & K,`
 `const Eigen::VectorXd & F) [inline]`

Constructor.

Parameters

K	
F	

3.21.2 Member Function Documentation

3.21.2.1 getRightHandSide() `const Eigen::VectorXd & Solver::getRightHandSide () const`

Getter for F.

Returns

`const Eigen::VectorXd&`

3.21.2.2 getSolutionDisplacements() `std::vector< real > Solver::getSolutionDisplacements ()`
`const`

Getter for U as std::vector.

Returns

`std::vector<real>`

3.21.2.3 getSolutionDisplacementsEig() `const Eigen::VectorXd & Solver::getSolutionDisplacements↵`
`Eig () const`

Getter for U as Eigen::Vector.

Returns

`const Eigen::VectorXd&`

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

- include/solver.hpp
- src/solver.cpp

3.22 SolverVEM Class Reference

Inheritance diagram for SolverVEM:

Collaboration diagram for SolverVEM:

Public Member Functions

- **SolverVEM** (const [Parameters](#) ¶meters, const [Mesh](#)< [Point3D](#), [Edge3D](#), [Polygon3D](#), [Polyhedron](#)< [Polygon3D](#) >> &mesh, const [VirtualDofsCollection](#) &DOFS, const [LocalVirtualDofsCollection](#) &dofs, const [VirtualPolyhedronProjections](#) &vp)
Constructor, assembles K and F.
- void **enforceHomogeneousDirichletBC** (const [Mesh](#)< [Point3D](#), [Edge3D](#), [Polygon3D](#), [Polyhedron](#)< [Polygon3D](#) >> &mesh, const [VirtualDofsCollection](#) &DOFS, const std::vector< std::function< real(real, real, real)>> &constraintFunctions)
Enforce homogeneous Dirichlet boundary conditions.
- real **computeStrainError** (const [Mesh](#)< [Point3D](#), [Edge3D](#), [Polygon3D](#), [Polyhedron](#)< [Polygon3D](#) >> &mesh, const [LocalVirtualDofsCollection](#) &dofs, const [VirtualPolyhedronProjections](#) &vp, const std::function< std::array< real, 6 >(real, real, real)> &EpsEx_func)
Compute the error in the L2 strain norm.

Additional Inherited Members

3.22.1 Constructor & Destructor Documentation

3.22.1.1 SolverVEM() `SolverVEM::SolverVEM (`
 const [Parameters](#) & parameters,
 const [Mesh](#)< [Point3D](#), [Edge3D](#), [Polygon3D](#), [Polyhedron](#)< [Polygon3D](#) >> & mesh,
 const [VirtualDofsCollection](#) & DOFS,
 const [LocalVirtualDofsCollection](#) & dofs,
 const [VirtualPolyhedronProjections](#) & vp)

Constructor, assembles K and F.

Parameters

<i>parameters</i>	
<i>mesh</i>	
<i>DOFS</i>	
<i>dofs</i>	
<i>vp</i>	

3.22.2 Member Function Documentation

```

3.22.2.1 computeStrainError() real SolverVEM::computeStrainError (
    const Mesh< Point3D, Edge3D, Polygon3D, Polyhedron< Polygon3D >> & mesh,
    const LocalVirtualDofsCollection & dofs,
    const VirtualPolyhedronProjections & vp,
    const std::function< std::array< real, 6 >(real, real, real)> & EpsEx_func )

```

Compute the error in the L2 strain norm.

Parameters

<i>mesh</i>	
<i>dofs</i>	
<i>vp</i>	
<i>EpsEx_func</i>	

Returns

real

```

3.22.2.2 enforceHomogeneousDirichletBC() void SolverVEM::enforceHomogeneousDirichletBC (
    const Mesh< Point3D, Edge3D, Polygon3D, Polyhedron< Polygon3D >> & mesh,
    const VirtualDofsCollection & DOFS,
    const std::vector< std::function< real(real, real, real)>> & constraintFunctions
)

```

Enforce homogeneous Dirichlet boundary conditions.

Parameters

<i>mesh</i>	
<i>DOFS</i>	
<i>constraintFunctions</i>	

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

- include/solver.hpp
- src/solver.cpp

3.23 VertexDof Class Reference

Inheritance diagram for VertexDof:

Collaboration diagram for VertexDof:

Public Member Functions

- [VertexDof](#) (std::size_t id, const [Point3D](#) &vertex_)
Constructor.
- std::size_t [getId](#) () const override
Getter for the id of the vertex.
- const [Point3D](#) & [getVertex](#) () const
Getter for the vertex.
- std::ostream & [operator<<](#) (std::ostream &os) const override
Output stream operator.

3.23.1 Constructor & Destructor Documentation

3.23.1.1 VertexDof() `VertexDof::VertexDof (`
`std::size_t id,`
`const Point3D & vertex_) [inline]`

Constructor.

Parameters

<i>id</i>	
<i>vertex</i> ↔	
—	

3.23.2 Member Function Documentation

3.23.2.1 getId() `std::size_t VertexDof::getId () const [override], [virtual]`

Getter for the id of the vertex.

Returns

std::size_t

Reimplemented from [VirtualDof](#).

3.23.2.2 getVertex() `const Point3D & VertexDof::getVertex () const`

Getter for the vertex.

Returns

const [Point3D](#)&

3.23.2.3 operator<<() `std::ostream& VertexDof::operator<< (std::ostream & os) const [inline], [override], [virtual]`

Output stream operator.

Parameters

<i>os</i>	
-----------	--

Returns

`std::ostream&`

Implements [VirtualDof](#).

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

- include/virtualDofs.hpp
- src/virtualDofs.cpp

3.24 VirtualDof Class Reference

Inheritance diagram for VirtualDof:

Public Member Functions

- [VirtualDof](#) ()
Constructor.
- virtual [~VirtualDof](#) ()=default
Destructor.
- virtual `std::ostream & operator<< (std::ostream &os) const =0`
Pure virtual output stream operator.
- virtual `std::size_t getId () const`
Getter for the [VirtualDof](#) Id.

Friends

- `std::ostream & operator<< (std::ostream &os, const VertexDof &vDof)`
Output stream operators.
- `std::ostream & operator<< (std::ostream &os, const EdgeDof &eDof)`
- `std::ostream & operator<< (std::ostream &os, const FaceDof &fDof)`
- `std::ostream & operator<< (std::ostream &os, const PolyhedronDof &pDof)`

3.24.1 Member Function Documentation

3.24.1.1 `getId()` `virtual std::size_t VirtualDof::getId () const [inline], [virtual]`

Getter for the [VirtualDof](#) Id.

Returns

`std::size_t`

Reimplemented in [PolyhedronDof](#), [FaceDof](#), [EdgeDof](#), and [VertexDof](#).

3.24.1.2 `operator<<()` `virtual std::ostream& VirtualDof::operator<< (std::ostream & os) const [pure virtual]`

Pure virtual output stream operator.

Parameters

<i>os</i>	
-----------	--

Returns

`std::ostream&`

Implemented in [PolyhedronDof](#), [FaceDof](#), [EdgeDof](#), and [VertexDof](#).

3.24.2 Friends And Related Function Documentation

3.24.2.1 `operator<<` `std::ostream& operator<< (std::ostream & os, const VertexDof & vDof) [friend]`

Output stream operators.

Parameters

<i>os</i>	
<i>vDof</i>	

Returns

`std::ostream&`

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

- `include/virtualDofs.hpp`
- `src/virtualDofs.cpp`

3.25 VirtualDofsCollection Class Reference

Public Member Functions

- virtual [~VirtualDofsCollection](#) ()=default
Virtual default destructor.
- [VirtualDofsCollection](#) (const [Mesh](#)< [Point3D](#), [Edge3D](#), [Polygon3D](#), [Polyhedron](#)< [Polygon3D](#) >> &mesh, unsigned int order_)
Constructor to create VirtualDofs from a [Mesh](#) and order.
- std::size_t [getnumVdofs](#) () const
Get the number of vertex-type dofs.
- std::size_t [getnumEdofs](#) () const
Get the number of edge-type dofs.
- std::size_t [getnumFdofs](#) () const
Get the number of face-type dofs.
- std::size_t [getnumPdofs](#) () const
Get the number of polyhedron-type dofs.
- std::size_t [getnumDofs](#) () const
Get the total number of dofs.
- template<typename DofType >
std::shared_ptr< DofType > [getDof](#) (std::size_t id) const
Method to get the corresponding specialized dof to a given id.
- std::shared_ptr< [VirtualDof](#) > [getDof](#) (std::size_t id) const
Method to get the corresponding dof to a given id.
- void [print](#) () const
Print [VirtualDofsCollection](#) information.

Static Public Member Functions

- static unsigned int [getOrder](#) ()
Get the order of the virtual dofs collection.

Friends

- std::ostream & [operator<<](#) (std::ostream &os, const [VirtualDofsCollection](#) &dofsCollection)
Output stream operator for [VirtualDofsCollection](#).

3.25.1 Constructor & Destructor Documentation

3.25.1.1 VirtualDofsCollection() `VirtualDofsCollection::VirtualDofsCollection (const Mesh< Point3D, Edge3D, Polygon3D, Polyhedron< Polygon3D >> & mesh, unsigned int order_)`

Constructor to create VirtualDofs from a [Mesh](#) and order.

Parameters

<i>mesh</i>	
<i>order</i> ↔	
—	

3.25.2 Member Function Documentation

3.25.2.1 `getDof()` [1/2] `template<typename DofType >`
`std::shared_ptr<DofType> VirtualDofsCollection::getDof (`
`std::size_t id) const [inline]`

Method to get the corresponding specialized dof to a given id.

Template Parameters

<i>DofType</i>	
----------------	--

Parameters

<i>id</i>	
-----------	--

Returns

`std::shared_ptr<DofType>`

3.25.2.2 `getDof()` [2/2] `std::shared_ptr< VirtualDof > VirtualDofsCollection::getDof (`
`std::size_t id) const`

Method to get the corresponding dof to a given id.

Parameters

<i>id</i>	
-----------	--

Returns

`std::shared_ptr<VirtualDof>`

3.25.2.3 getnumDofs() `std::size_t VirtualDofsCollection::getnumDofs () const`

Get the total number of dofs.

Returns

`std::size_t`

3.25.2.4 getnumEdofs() `std::size_t VirtualDofsCollection::getnumEdofs () const`

Get the number of edge-type dofs.

Returns

`std::size_t`

3.25.2.5 getnumFdofs() `std::size_t VirtualDofsCollection::getnumFdofs () const`

Get the number of face-type dofs.

Returns

`std::size_t`

3.25.2.6 getnumPdofs() `std::size_t VirtualDofsCollection::getnumPdofs () const`

Get the number of polyhedron-type dofs.

Returns

`std::size_t`

3.25.2.7 getnumVdofs() `std::size_t VirtualDofsCollection::getnumVdofs () const`

Get the number of vertex-type dofs.

Returns

`std::size_t`

3.25.2.8 getOrder() `unsigned int VirtualDofsCollection::getOrder () [static]`

Get the order of the virtual dofs collection.

Returns

unsigned int

3.25.3 Friends And Related Function Documentation

3.25.3.1 operator<< `std::ostream& operator<< (std::ostream & os, const VirtualDofsCollection & dofsCollection) [friend]`

Output stream operator for [VirtualDofsCollection](#).

Parameters

<i>os</i>	
<i>dofsCollection</i>	

Returns

std::ostream&

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

- include/virtualDofs.hpp
- src/virtualDofs.cpp

3.26 VirtualFaceProjections Class Reference

Public Member Functions

- [VirtualFaceProjections](#) (const [VirtualDofsCollection](#) &dofs, const [Mesh](#)< [Point3D](#), [Edge3D](#), [Polygon3D](#), [Polyhedron](#)< [Polygon3D](#) >> &mesh, const unsigned int &order)
Constructor.
- std::vector< [Polynomial](#)< 2 > > [computeFaceProjection](#) (const [VirtualDofsCollection](#) &dofs, const [Polygon3D](#) &face, const unsigned int &order, bool checkConsistency=false)
Compute the projections of the basis functions corresponding to the dofs defined on the face.
- const std::vector< [Polynomial](#)< 2 > > & [getFaceProjection](#) (const std::size_t &Id) const
Get the face projections.

3.26.1 Constructor & Destructor Documentation

3.26.1.1 VirtualFaceProjections() `VirtualFaceProjections::VirtualFaceProjections (`
`const VirtualDofsCollection & dofs,`
`const Mesh< Point3D, Edge3D, Polygon3D, Polyhedron< Polygon3D >> & mesh,`
`const unsigned int & order)`

Constructor.

Parameters

<i>dofs</i>	VirtualDofsCollection
<i>mesh</i>	
<i>order</i>	

3.26.2 Member Function Documentation

3.26.2.1 computeFaceProjection() `std::vector< Polynomial< 2 > > VirtualFaceProjections↵
::computeFaceProjection (`
`const VirtualDofsCollection & dofs,`
`const Polygon3D & face,`
`const unsigned int & order,`
`bool checkConsistency = false)`

Compute the projections of the basis functions corresponding to the dofs defined on the face.

Parameters

<i>dofs</i>	VirtualDofsCollection
<i>face</i>	face over which the projection is performed
<i>order</i>	
<i>checkConsistency</i>	if true the code checks the Frobenius norm of the difference between the matrix Gf and matrix BfDf, exploiting the identity BfDf=Gf

Returns

`std::vector<Polynomial<2>>`

3.26.2.2 getFaceProjection() `const std::vector< Polynomial< 2 > > & VirtualFaceProjections↵
::getFaceProjection (`
`const std::size_t & Id) const`

Get the face projections.

Parameters

<i>Id</i>	
-----------	--

Returns

const std::vector<Polynomial<2>>>&

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

- include/virtualProjections.hpp
- src/virtualProjections.cpp

3.27 VirtualPolyhedronProjections Class Reference

Public Member Functions

- [VirtualPolyhedronProjections](#) (const real &E, const real &nu, const [VirtualFaceProjections](#) &faceProjections, const [LocalVirtualDofsCollection](#) &dofs, const [Mesh](#)< [Point3D](#), [Edge3D](#), [Polygon3D](#), [Polyhedron](#)< [Polygon3D](#) >> &mesh, const std::function< real(real, real, real)> &funcx, const std::function< real(real, real, real)> &funcy, const std::function< real(real, real, real)> &funcz, const unsigned int &order)
Constructor with the materials parameters and the forcing function.
- [VirtualPolyhedronProjections](#) (const [Parameters](#) ¶meters, const [VirtualFaceProjections](#) &faceProjections, const [LocalVirtualDofsCollection](#) &dofs, const [Mesh](#)< [Point3D](#), [Edge3D](#), [Polygon3D](#), [Polyhedron](#)< [Polygon3D](#) >> &mesh)
Constructor taking an instance of parameters.
- Eigen::SparseMatrix< real > [computePolyhedronProjections](#) (const [VirtualFaceProjections](#) &faceProjections, const [LocalVirtualDofs](#) &dofs, const [Polyhedron](#)< [Polygon3D](#) > &polyhedron, const std::function< real(real, real, real)> &funcx, const std::function< real(real, real, real)> &funcy, const std::function< real(real, real, real)> &funcz, const unsigned int &order)
Compute the projections over the polyhedron.
- const std::map< std::size_t, Eigen::SparseMatrix< real > > & [getInverseMatricesG](#) () const
Get the matrices G.
- const std::map< std::size_t, Eigen::SparseMatrix< real > > & [getPolyhedronProjections](#) () const
Get the projections C.
- const std::map< std::size_t, Eigen::SparseMatrix< real > > & [getElasticMatrices](#) () const
Get the elastic matrices E.
- const std::map< std::size_t, Eigen::SparseMatrix< real > > & [getDeformationRBMMatrices](#) () const
Get the deformation and rigid body motion matrices Tdr.
- const std::map< std::size_t, Eigen::VectorXd > & [getforcingProjections](#) () const
Get the forcing projections.
- const unsigned int & [getOrder](#) () const
Get the order.

3.27.1 Constructor & Destructor Documentation

3.27.1.1 VirtualPolyhedronProjections() [1/2] VirtualPolyhedronProjections::VirtualPolyhedronProjections (

```
const real & E,
const real & nu,
const VirtualFaceProjections & faceProjections,
const LocalVirtualDofsCollection & dofs,
const Mesh< Point3D, Edge3D, Polygon3D, Polyhedron< Polygon3D >> & mesh,
const std::function< real(real, real, real)> & funcx,
const std::function< real(real, real, real)> & funcy,
const std::function< real(real, real, real)> & funcz,
const unsigned int & order )
```

Constructor with the materials parameters and the forcing function.

Parameters

<i>E</i>	youngs modulus
<i>nu</i>	poisson ratio
<i>faceProjections</i>	
<i>dofs</i>	VirtualDofsCollection
<i>mesh</i>	
<i>funcx</i>	forcing function x direction
<i>funcy</i>	forcing function y direction
<i>funcz</i>	forcing function z direction
<i>order</i>	

3.27.1.2 VirtualPolyhedronProjections() [2/2] `VirtualPolyhedronProjections::VirtualPolyhedronProjections (`

```
const Parameters & parameters,
const VirtualFaceProjections & faceProjections,
const LocalVirtualDofsCollection & dofs,
const Mesh< Point3D, Edge3D, Polygon3D, Polyhedron< Polygon3D >> & mesh )
```

Constructor taking an instance of parameters.

Parameters

<i>parameters</i>	
<i>faceProjections</i>	
<i>dofs</i>	VirtualDofsCollection
<i>mesh</i>	

3.27.2 Member Function Documentation

3.27.2.1 computePolyhedronProjections() `Eigen::SparseMatrix< real > VirtualPolyhedronProjections::computePolyhedronProjections (`

```
const VirtualFaceProjections & faceProjections,
const LocalVirtualDofs & dofs,
const Polyhedron< Polygon3D > & polyhedron,
const std::function< real(real, real, real)> & funcx,
const std::function< real(real, real, real)> & funcy,
const std::function< real(real, real, real)> & funcz,
const unsigned int & order )
```

Compute the projections over the polyhedron.

Parameters

<i>faceProjections</i>	
------------------------	--

Parameters

<i>dofs</i>	VirtualDofsCollection
<i>polyhedron</i>	
<i>funcx</i>	forcing function x direction
<i>funcy</i>	forcing function y direction
<i>funcz</i>	forcing function z direction
<i>order</i>	

Returns

Eigen::SparseMatrix<real>

3.27.2.2 getDeformationRBMMatrices() `const std::map< std::size_t, Eigen::SparseMatrix< real > > & VirtualPolyhedronProjections::getDeformationRBMMatrices () const`

Get the deformation and rigid body motion matrices Tdr.

Returns

`const std::map<std::size_t, Eigen::SparseMatrix<real>>&`

3.27.2.3 getElasticMatrices() `const std::map< std::size_t, Eigen::SparseMatrix< real > > & VirtualPolyhedronProjections::getElasticMatrices () const`

Get the elastic matrices E.

Returns

`const std::map<std::size_t, Eigen::SparseMatrix<real>>&`

3.27.2.4 getforcingProjections() `const std::map< std::size_t, Eigen::VectorXd > & VirtualPolyhedronProjections::getforcingProjections () const`

Get the forcing projections.

Returns

`const std::map<std::size_t, Eigen::VectorXd>&`

3.27.2.5 getInverseMatricesG() `const std::map< std::size_t, Eigen::SparseMatrix< real > > & VirtualPolyhedronProjections::getInverseMatricesG () const`

Get the matrices G.

Returns

`const std::map<std::size_t, Eigen::SparseMatrix<real>>&`

3.27.2.6 getOrder() `const unsigned int& VirtualPolyhedronProjections::getOrder () const [inline]`

Get the order.

Returns

`const unsigned int&`

3.27.2.7 getPolyhedronProjections() `const std::map< std::size_t, Eigen::SparseMatrix< real > > & VirtualPolyhedronProjections::getPolyhedronProjections () const`

Get the projections C.

Returns

`const std::map<std::size_t, Eigen::SparseMatrix<real>>&`

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

- include/virtualProjections.hpp
- src/virtualProjections.cpp

Index

- addEdge
 - geometry::Polygon< EdgeType >, 50
- addMonomial
 - Polynomial< Dimension >, 61
- addPolygon
 - geometry::Polyhedron< PolygonType >, 56
- areEdgesConsistent
 - geometry::Polygon< EdgeType >, 50
- binaryOperation
 - geometry::Point< Args >, 41
- computeFaceProjection
 - VirtualFaceProjections, 74
- computeGradientsToMonomialsOrdered
 - Monomial3D, 30
- computeLaplaciansToMonomialsOrdered
 - Monomial2D, 26
 - Monomial3D, 30
- computeMonomialsUpToOrder
 - Monomial2D, 26
 - Monomial3D, 31
- computePolyhedronProjections
 - VirtualPolyhedronProjections, 76
- computeStrainError
 - SolverVEM, 65
- cross
 - geometry::Point< Args >, 42
- derivative
 - Monomial< Dimension >, 22
 - Monomial2D, 26
 - Monomial3D, 31
- distance
 - geometry::Point< Args >, 42
- dot
 - geometry::Point< Args >, 43
- dx
 - Monomial2D, 27
 - Monomial3D, 31
- dy
 - Monomial2D, 27
 - Monomial3D, 31
- dz
 - Monomial3D, 32
- Edge
 - geometry::Edge< PointType >, 4
- EdgeDof, 7
- enforceHomogeneousDirichletBC
 - SolverVEM, 66
- EToLocalId
 - LocalVirtualDofs, 12
- evaluate
 - Monomial< Dimension >, 22
- FaceDof, 7
- getId, 8
- getMonomial, 8
- operator<<, 8
- FToLocalId
 - LocalVirtualDofs, 12
- Gauss::GaussData, 9
- GaussLobatto::GaussLobattoCache, 9
 - getCache, 9
 - initialize, 9
- geometry::Edge< PointType >, 3
 - Edge, 4
 - getDirection, 4
 - getId, 4
 - getLength, 4
 - getOtherHalfEdge, 5
 - operator<, 5
 - operator<<, 7
 - operator==, 5
 - operator[], 6
 - setId, 6
 - setOtherHalfEdge, 6
- geometry::Point< Args >, 40
 - binaryOperation, 41
 - cross, 42
 - distance, 42
 - dot, 43
 - getCoordinates, 43
 - getId, 43
 - multiplyByScalar, 43
 - norm, 44
 - normalize, 44
 - operator<, 46
 - operator<<, 48
 - operator*, 44, 48
 - operator+, 45
 - operator-, 45
 - operator/, 45
 - operator==, 46
 - operator[], 47
 - piecewiseMultiply, 47
 - Point, 41
 - setId, 47
- geometry::Polygon< EdgeType >, 49
 - addEdge, 50
 - areEdgesConsistent, 50
 - get_e_x, 50
 - get_e_y, 51
 - getArea, 51
 - getDiameter, 51
 - getEdge, 51
 - getId, 52
 - getOtherPolygon, 52
 - getOutwardNormal, 52

- getPositiveEdge, [52](#)
- numEdges, [53](#)
- operator<, [53](#)
- operator<=, [55](#)
- operator==, [53](#)
- operator[], [54](#)
- Polygon, [50](#)
- setId, [54](#)
- setOrientation, [54](#)
- setOtherPolygon, [54](#)
- geometry::Polyhedron< PolygonType >, [55](#)
 - addPolygon, [56](#)
 - getDiameter, [56](#)
 - getId, [57](#)
 - getPolygon, [57](#)
 - numPolygons, [57](#)
 - operator<=, [58](#)
 - operator[], [57](#)
 - Polyhedron, [56](#)
 - setId, [58](#)
- get_e_x
 - geometry::Polygon< EdgeType >, [50](#)
- get_e_y
 - geometry::Polygon< EdgeType >, [51](#)
- getArea
 - geometry::Polygon< EdgeType >, [51](#)
- getCache
 - GaussLobatto::GaussLobattoCache, [9](#)
- getCacheMonomial
 - IntegrationMonomial::MonomialsFaceIntegralsCache, [34](#)
- getCacheMonomials
 - IntegrationMonomial::MonomialsFaceIntegralsCache, [34](#)
- getCoefficient
 - Monomial< Dimension >, [23](#)
- getCoordinates
 - geometry::Point< Args >, [43](#)
- getDeformationRBMMatrices
 - VirtualPolyhedronProjections, [77](#)
- getDiameter
 - geometry::Polygon< EdgeType >, [51](#)
 - geometry::Polyhedron< PolygonType >, [56](#)
- getDirection
 - geometry::Edge< PointType >, [4](#)
- getDof
 - LocalVirtualDofs, [12](#)
 - VirtualDofsCollection, [71](#)
- getEdge
 - geometry::Polygon< EdgeType >, [51](#)
 - Mesh< PointType, EdgeType, PolygonType, PolyhedronType >, [17](#)
- getEdges
 - Mesh< PointType, EdgeType, PolygonType, PolyhedronType >, [18](#)
- getElasticMatrices
 - VirtualPolyhedronProjections, [77](#)
- getExactSolution
 - Parameters, [37](#)
- getExactStrains
 - Parameters, [37](#)
- getExponents
 - Monomial< Dimension >, [23](#)
- getFaceProjection
 - VirtualFaceProjections, [74](#)
- getFirstLame
 - Parameters, [38](#)
- getForcing
 - Parameters, [38](#)
- getforcingProjections
 - VirtualPolyhedronProjections, [77](#)
- getGradientsToMonomialsOrdered
 - Monomial3D, [32](#)
- getHomogeneousDirichletBC
 - Parameters, [38](#)
- getID
 - LocalVirtualDofs, [13](#)
- getId
 - FaceDof, [8](#)
 - geometry::Edge< PointType >, [4](#)
 - geometry::Point< Args >, [43](#)
 - geometry::Polygon< EdgeType >, [52](#)
 - geometry::Polyhedron< PolygonType >, [57](#)
 - PolyhedronDof, [59](#)
 - VertexDof, [67](#)
 - VirtualDof, [68](#)
- getInitializeFaceIntegrals
 - Parameters, [38](#)
- getInputMesh
 - Parameters, [38](#)
- getInverseMatricesG
 - VirtualPolyhedronProjections, [77](#)
- getLaplaciansToMonomialsOrdered
 - Monomial2D, [27](#)
 - Monomial3D, [32](#)
- getLength
 - geometry::Edge< PointType >, [4](#)
- getLocalDofs
 - LocalVirtualDofsCollection, [16](#)
- getMonomial
 - FaceDof, [8](#)
 - PolyhedronDof, [59](#)
- getMonomialsOrdered
 - Monomial2D, [27](#)
 - Monomial3D, [33](#)
- getnumDofs
 - LocalVirtualDofs, [13](#)
 - VirtualDofsCollection, [71](#)
- getnumEdofs
 - LocalVirtualDofs, [13](#)
 - VirtualDofsCollection, [72](#)
- getnumFdofs
 - LocalVirtualDofs, [13](#)
 - VirtualDofsCollection, [72](#)
- getnumPdofs
 - LocalVirtualDofs, [14](#)

- VirtualDofsCollection, 72
- getnumVdofs
 - LocalVirtualDofs, 14
 - VirtualDofsCollection, 72
- getOrder
 - Monomial< Dimension >, 23
 - Parameters, 39
 - Polynomial< Dimension >, 61
 - VirtualDofsCollection, 72
 - VirtualPolyhedronProjections, 78
- getOtherHalfEdge
 - geometry::Edge< PointType >, 5
- getOtherPolygon
 - geometry::Polygon< EdgeType >, 52
- getOutwardNormal
 - geometry::Polygon< EdgeType >, 52
- getPoissonRatio
 - Parameters, 39
- getPolygon
 - geometry::Polyhedron< PolygonType >, 57
 - Mesh< PointType, EdgeType, PolygonType, PolyhedronType >, 18
- getPolygons
 - Mesh< PointType, EdgeType, PolygonType, PolyhedronType >, 18
- getPolyhedra
 - Mesh< PointType, EdgeType, PolygonType, PolyhedronType >, 19
- getPolyhedron
 - Mesh< PointType, EdgeType, PolygonType, PolyhedronType >, 19
- getPolyhedronProjections
 - VirtualPolyhedronProjections, 78
- getPolynomial
 - Polynomial< Dimension >, 61
- getPositiveEdge
 - geometry::Polygon< EdgeType >, 52
- getRightHandSide
 - Solver, 64
- getSecondLame
 - Parameters, 39
- getSize
 - Mesh< PointType, EdgeType, PolygonType, PolyhedronType >, 19
- getSolutionDisplacements
 - Solver, 64
- getSolutionDisplacementsEig
 - Solver, 64
- getVertex
 - Mesh< PointType, EdgeType, PolygonType, PolyhedronType >, 19
 - VertexDof, 67
- getVertices
 - Mesh< PointType, EdgeType, PolygonType, PolyhedronType >, 20
- getYoungsModulus
 - Parameters, 39
- initialize
 - GaussLobatto::GaussLobattoCache, 9
 - IntegrationMonomial::MonomialsFaceIntegralsCache, 36
 - IntegrationMonomial::MonomialsFaceIntegralsCache, 34
 - getCacheMonomial, 34
 - getCacheMonomials, 34
 - initialize, 36
 - LinearTrinomialPower, 10
 - LinearTrinomialPower, 10
 - LocalVirtualDofs, 11
 - EToLocalId, 12
 - FToLocalId, 12
 - getDof, 12
 - getID, 13
 - getnumDofs, 13
 - getnumEdofs, 13
 - getnumFdofs, 13
 - getnumPdofs, 14
 - getnumVdofs, 14
 - LocalVirtualDofs, 11
 - operator<<, 15
 - PToLocalId, 14
 - VToLocalId, 14
 - LocalVirtualDofsCollection, 15
 - getLocalDofs, 16
 - LocalVirtualDofsCollection, 15
 - numLocalDofsCollection, 16
 - Mesh
 - Mesh< PointType, EdgeType, PolygonType, PolyhedronType >, 17
 - Mesh< PointType, EdgeType, PolygonType, PolyhedronType >, 16
 - getEdge, 17
 - getEdges, 18
 - getPolygon, 18
 - getPolygons, 18
 - getPolyhedra, 19
 - getPolyhedron, 19
 - getSize, 19
 - getVertex, 19
 - getVertices, 20
 - Mesh, 17
 - numEdges, 20
 - numPolygons, 20
 - numPolyhedra, 20
 - numVertices, 21
 - Monomial
 - Monomial< Dimension >, 22
 - Monomial< Dimension >, 21
 - derivative, 22
 - evaluate, 22
 - getCoefficient, 23
 - getExponents, 23
 - getOrder, 23
 - Monomial, 22
 - operator<<, 24

- operator*, 23
- setCoefficient, 24
- Monomial2D, 24
 - computeLaplaciansToMonomialsOrdered, 26
 - computeMonomialsUpToOrder, 26
 - derivative, 26
 - dx, 27
 - dy, 27
 - getLaplaciansToMonomialsOrdered, 27
 - getMonomialsOrdered, 27
 - Monomial2D, 25, 26
 - operator<<, 28
 - operator*, 28
- Monomial3D, 29
 - computeGradientsToMonomialsOrdered, 30
 - computeLaplaciansToMonomialsOrdered, 30
 - computeMonomialsUpToOrder, 31
 - derivative, 31
 - dx, 31
 - dy, 31
 - dz, 32
 - getGradientsToMonomialsOrdered, 32
 - getLaplaciansToMonomialsOrdered, 32
 - getMonomialsOrdered, 33
 - Monomial3D, 30
 - operator<<, 33
 - operator*, 33
- multiplyByScalar
 - geometry::Point< Args >, 43
- norm
 - geometry::Point< Args >, 44
- normalize
 - geometry::Point< Args >, 44
- numEdges
 - geometry::Polygon< EdgeType >, 53
 - Mesh< PointType, EdgeType, PolygonType, PolyhedronType >, 20
- numLocalDofsCollection
 - LocalVirtualDofsCollection, 16
- numPolygons
 - geometry::Polyhedron< PolygonType >, 57
 - Mesh< PointType, EdgeType, PolygonType, PolyhedronType >, 20
- numPolyhedra
 - Mesh< PointType, EdgeType, PolygonType, PolyhedronType >, 20
- numVertices
 - Mesh< PointType, EdgeType, PolygonType, PolyhedronType >, 21
- operator<
 - geometry::Edge< PointType >, 5
 - geometry::Point< Args >, 46
 - geometry::Polygon< EdgeType >, 53
- operator<<
 - FaceDof, 8
 - geometry::Edge< PointType >, 7
 - geometry::Point< Args >, 48
- geometry::Polygon< EdgeType >, 55
- geometry::Polyhedron< PolygonType >, 58
- LocalVirtualDofs, 15
- Monomial< Dimension >, 24
- Monomial2D, 28
- Monomial3D, 33
- PolyhedronDof, 60
- VertexDof, 67
- VirtualDof, 69
- VirtualDofsCollection, 73
- operator*
 - geometry::Point< Args >, 44, 48
 - Monomial< Dimension >, 23
 - Monomial2D, 28
 - Monomial3D, 33
 - Polynomial< Dimension >, 62
- operator+
 - geometry::Point< Args >, 45
- operator-
 - geometry::Point< Args >, 45
- operator/
 - geometry::Point< Args >, 45
- operator==
 - geometry::Edge< PointType >, 5
 - geometry::Point< Args >, 46
 - geometry::Polygon< EdgeType >, 53
- operator[]
 - geometry::Edge< PointType >, 6
 - geometry::Point< Args >, 47
 - geometry::Polygon< EdgeType >, 54
 - geometry::Polyhedron< PolygonType >, 57
- Parameters, 36
 - getExactSolution, 37
 - getExactStrains, 37
 - getFirstLame, 38
 - getForcing, 38
 - getHomogeneousDirichletBC, 38
 - getInitializeFaceIntegrals, 38
 - getInputMesh, 38
 - getOrder, 39
 - getPoissonRatio, 39
 - getSecondLame, 39
 - getYoungsModulus, 39
 - Parameters, 37
- piecewiseMultiply
 - geometry::Point< Args >, 47
- Point
 - geometry::Point< Args >, 41
- Polygon
 - geometry::Polygon< EdgeType >, 50
- Polyhedron
 - geometry::Polyhedron< PolygonType >, 56
- PolyhedronDof, 59
 - getId, 59
 - getMonomial, 59
 - operator<<, 60
 - PolyhedronDof, 59
- Polynomial

- Polynomial< Dimension >, 61
- Polynomial< Dimension >, 60
 - addMonomial, 61
 - getOrder, 61
 - getPolynomial, 61
 - operator*, 62
 - Polynomial, 61
 - size, 62
- Problem, 63
- PToLocalId
 - LocalVirtualDofs, 14
- setCoefficient
 - Monomial< Dimension >, 24
- setId
 - geometry::Edge< PointType >, 6
 - geometry::Point< Args >, 47
 - geometry::Polygon< EdgeType >, 54
 - geometry::Polyhedron< PolygonType >, 58
- setOrientation
 - geometry::Polygon< EdgeType >, 54
- setOtherHalfEdge
 - geometry::Edge< PointType >, 6
- setOtherPolygon
 - geometry::Polygon< EdgeType >, 54
- size
 - Polynomial< Dimension >, 62
- Solver, 63
 - getRightHandSide, 64
 - getSolutionDisplacements, 64
 - getSolutionDisplacementsEig, 64
 - Solver, 64
- SolverVEM, 65
 - computeStrainError, 65
 - enforceHomogeneousDirichletBC, 66
 - SolverVEM, 65
- VertexDof, 66
 - getId, 67
 - getVertex, 67
 - operator<<, 67
 - VertexDof, 67
- VirtualDof, 68
 - getId, 68
 - operator<<, 69
- VirtualDofsCollection, 70
 - getDof, 71
 - getnumDofs, 71
 - getnumEdofs, 72
 - getnumFdofs, 72
 - getnumPdofs, 72
 - getnumVdofs, 72
 - getOrder, 72
 - operator<<, 73
 - VirtualDofsCollection, 70
- VirtualFaceProjections, 73
 - computeFaceProjection, 74
 - getFaceProjection, 74
 - VirtualFaceProjections, 73
- VirtualPolyhedronProjections, 75
 - computePolyhedronProjections, 76
 - getDeformationRBMMatrices, 77
 - getElasticMatrices, 77
 - getforcingProjections, 77
 - getInverseMatricesG, 77
 - getOrder, 78
 - getPolyhedronProjections, 78
 - VirtualPolyhedronProjections, 75, 76
- VToLocalId
 - LocalVirtualDofs, 14