My Project

Generated by Doxygen 1.8.18

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 BasicMethod Class Reference

Inheritance diagram for BasicMethod:

Collaboration diagram for BasicMethod:

Public Member Functions

- BasicMethod (GetPot const &dataFile, std::string const problem, Bulk &bulk1, Bulk &bulk2, const size_type dim, LinearSystem &extSys)
- virtual void assembleMatrix ()=0
- void assembleRHS () override
- void enforceStrongBC (size_type const domainIdx) override

  overriden method for the strong Dirichlet boundary conditions
- void treatIFaceDofs () override
- void solve () override
- virtual ~BasicMethod ()

Additional Inherited Members

4.1.1 Constructor & Destructor Documentation

4.1.1.1 BasicMethod()

```
BasicMethod::BasicMethod (
            GetPot const & dataFile,
            std::string const problem,
            Bulk & bulk1,
            Bulk & bulk2,
            const size_type dim,
            LinearSystem & extSys )
```

constructor

4.1.1.2 ~BasicMethod()

```
virtual BasicMethod::~BasicMethod ( )  [inline], [virtual]
```

destructor

4.1.2 Member Function Documentation

4.1.2.1 assembleMatrix()

```
virtual void BasicMethod::assembleMatrix ( )  [pure virtual]
```

a pure virtual method for the assembly of the matrix

Implements Problem.

Implemented in LinearElasticityBasic, and LaplacianBasic.

4.1.2.2 assembleRHS()

```
void BasicMethod::assembleRHS ( )  [override], [virtual]
```

overriden method for the assembly of the right hand side

Implements Problem.

4.1.2.3 enforceStrongBC()

```
void BasicMethod::enforceStrongBC (
            size_type const domainIdx )  [override], [virtual]
```

overriden method for the strong Dirichlet boundary conditions

This methods takes one argument

Parameters

domainIdx	index to identify if we are in the first or in the second domain

Implements Problem.

4.1.2.4 solve()

```
void BasicMethod::solve ( )  [override], [virtual]
```

overriden method for the resolution of the linear system

Implements Problem.

4.1.2.5 treatIFaceDofs()

```
void BasicMethod::treatIFaceDofs ( )  [override], [virtual]
```

overriden method for the management of the interface degrees of freedom

Implements Problem.

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

- include/BasicMethod.h

4.2 BC Class Reference

Public Member Functions

- BC (const GetPot &dataFile, const std::string &problem, const std::string &section)
- void **setBoundaries** (getfem::mesh &meshRef)
- std::vector< size_type > **getNeumBD** () const
- std::vector< size_type > **getDiriBD** () const
- scalar_type BCNeum (const base_node &x, const size_type what, const size_type &flag)
  method to to evaluate the Neumann boundary conditions.
- scalar_type BCDiri (const base_node &x, const size_type what, const size_type &flag)
  method to to evaluate the Dirichlet boundary conditions.

4.2.1 Constructor & Destructor Documentation

4.2.1.1 BC()

```
BC::BC (
            const GetPot & dataFile,
            const std::string & problem,
            const std::string & section )
```

constructor

4.2.2 Member Function Documentation

4.2.2.1 BCDiri()

```
scalar_type BC::BCDiri (
            const base_node & x,
            const size_type what,
            const size_type & flag )
```

method to to evaluate the Dirichlet boundary conditions.

This method takes three input parameters and returns a scalar value.

Parameters

x	coordinates of the point where we want to evaluate the function.
flag	index indicating the side of the domain
what	index 0 if the datum is a scalar or if we want the first component of the vector; index 1 if we want to evaluate the second component.

4.2.2.2 BCNeum()

```
scalar_type BC::BCNeum (
            const base_node & x,
            const size_type what,
            const size_type & flag )
```

method to to evaluate the Neumann boundary conditions.

This method takes three input parameters and returns a scalar value.

Parameters

x	coordinates of the point where we want to evaluate the function.
flag	index indicating the side of the domain
what	index 0 if the datum is a scalar or if we want the first component of the vector; index 1 if we want to evaluate the second component.

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

- include/BC.h

4.3 Bulk Class Reference

Public Member Functions

- Bulk (const GetPot &dataFile, const std::string &section="bulkData/", const std::string &sectionDomain="domain/", const std::string &sectionProblem="laplacian", const std::string &domainNumber="1")
- void exportMesh (std::string const nomefile) const

  method to export the mesh
- getfem::mesh & **getMeshRef** ()
- getfem::mesh **getMesh** () const
- scalar_type **Lx** () const
- scalar_type **Ly** () const
- scalar_type **nSubX** () const
- scalar_type **nSubY** () const

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Bulk()

```
Bulk::Bulk (
            const GetPot & dataFile,
            const std::string & section = "bulkData/",
            const std::string & sectionDomain = "domain/",
            const std::string & sectionProblem = "laplacian",
            const std::string & domainNumber = "1" )
```

constructor

4.3.2 Member Function Documentation

4.3.2.1 exportMesh()

```
void Bulk::exportMesh (
            std::string const nomefile ) const
```

method to export the mesh

This method takes one argument as input.

Parameters

nomefile	a string containing the name of the file where we want to save the exported mesh

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

- include/Bulk.h

4.4 BulkDatum Class Reference

Public Member Functions

- BulkDatum (const GetPot &dataFile, const std::string &section, const std::string &sectionProblem, const std::string &domainNumber, const std::string &datum)
- scalar_type getValue (const base_node &x, const size_type what)

  method to evaluate the string.

4.4.1 Constructor & Destructor Documentation

4.4.1.1 BulkDatum()

```
BulkDatum::BulkDatum (
             const GetPot & dataFile,
             const std::string & section,
             const std::string & sectionProblem,
             const std::string & domainNumber,
             const std::string & datum )
```

constructor

4.4.2 Member Function Documentation

4.4.2.1 getValue()

```
scalar_type BulkDatum::getValue (
             const base_node & x,
             const size_type what )
```

method to evaluate the string.

This method takes two input parameters and returns a scalar value.

Parameters

x	coordinates of the point where we want to evaluate the function.
what	index 0 if the datum is a scalar or if we want the first component of the vector; index 1 if we want to evaluate the second component.

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

- include/BulkDatum.h

4.5 FEM Class Reference

Public Member Functions

- FEM (const getfem::mesh &mesh, const GetPot &dataFile, const std::string &problem, const std::string &variable, const std::string &section="bulkData/", const size_type qdim=1)
- FEM (const getfem::mesh &mesh, const std::string femType, const size_type spaceDim)
- size_type **nb_dof** () const
- std::string **type** () const
- getfem::mesh_fem **getFEM** () const
- std::vector< base_node > **getDOFpoints** () const
- base_node **point_of_basic_dof** (size_type i) const

4.5.1 Constructor & Destructor Documentation

4.5.1.1 FEM() [1/2]

```
FEM::FEM (
              const getfem::mesh & mesh,
              const GetPot & dataFile,
              const std::string & problem,
              const std::string & variable,
              const std::string & section = "bulkData/",
              const size_type qdim = 1 )
```

constructor based on the input file

4.5.1.2 FEM() [2/2]

```
FEM::FEM (
              const getfem::mesh & mesh,
              const std::string femType,
              const size_type spaceDim )
```

constructor that is not based on the input file

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

- include/FEM.h

4.6 LaplacianBasic Class Reference

Inheritance diagram for LaplacianBasic:

Collaboration diagram for LaplacianBasic:

Public Member Functions

- LaplacianBasic (GetPot const &dataFile, Bulk &bulk1, Bulk &bulk2, LinearSystem &extSys)
- void assembleMatrix () override

Static Public Attributes

- static const size_type Qdim = 1

Additional Inherited Members

4.6.1 Constructor & Destructor Documentation

4.6.1.1 LaplacianBasic()

```
LaplacianBasic::LaplacianBasic (
            GetPot const & dataFile,
            Bulk & bulk1,
            Bulk & bulk2,
            LinearSystem & extSys )
```

constructor

4.6.2 Member Function Documentation

4.6.2.1 assembleMatrix()

```
void LaplacianBasic::assembleMatrix ( )  [override], [virtual]
```

overriden method for the assembly of the Matrix in the case of an elliptic scalar problem

Implements BasicMethod.

4.6.3 Member Data Documentation

4.6.3.1 Qdim

```
const size_type LaplacianBasic::Qdim = 1  [static]
```

static member for the dimension of the solution and the necessary FEM (1 because the problem is scalar)

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

- include/LaplacianBasic.h

4.7 LaplacianSymmetric Class Reference

Inheritance diagram for LaplacianSymmetric:

Collaboration diagram for LaplacianSymmetric:

Public Member Functions

- LaplacianSymmetric (GetPot const &dataFile, Bulk &bulk1, Bulk &bulk2, LinearSystem &extSys)
- void assembleMatrix () override

Static Public Attributes

- static const size_type Qdim = 1

Additional Inherited Members

4.7.1 Constructor & Destructor Documentation

4.7.1.1 LaplacianSymmetric()

```
LaplacianSymmetric::LaplacianSymmetric (
            GetPot const & dataFile,
            Bulk & bulk1,
            Bulk & bulk2,
            LinearSystem & extSys )
```

constructor

4.7.2 Member Function Documentation

4.7.2.1 assembleMatrix()

```
void LaplacianSymmetric::assembleMatrix ( )  [override], [virtual]
```

overriden method for the assembly of the Matrix in the case of an elliptic scalar problem

Implements SymmetricMethod.

4.7.3 Member Data Documentation

4.7.3.1 Qdim

```
const size_type LaplacianSymmetric::Qdim = 1  [static]
```

static member for the dimension of the solution and the necessary FEM (1 because the problem is scalar)

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

- include/LaplacianSymmetric.h

4.8 LinearElasticityBasic Class Reference

Inheritance diagram for LinearElasticityBasic:

Collaboration diagram for LinearElasticityBasic:

Public Member Functions

- LinearElasticityBasic (GetPot const &dataFile, Bulk &bulk1, Bulk &bulk2, LinearSystem &extSys)
- void assembleMatrix () override

Static Public Attributes

- static const size_type Qdim = 2

Additional Inherited Members

4.8.1 Constructor & Destructor Documentation

4.8.1.1 LinearElasticityBasic()

```
LinearElasticityBasic::LinearElasticityBasic (
            GetPot const & dataFile,
            Bulk & bulk1,
            Bulk & bulk2,
            LinearSystem & extSys )
```

constructor

4.8.2 Member Function Documentation

4.8.2.1 assembleMatrix()

```
void LinearElasticityBasic::assembleMatrix ( )  [override], [virtual]
```

overriden method for the assembly of the Matrix in the case of a linear Elasticity problem

Implements BasicMethod.

4.8.3 Member Data Documentation

4.8.3.1 Qdim

```
const size_type LinearElasticityBasic::Qdim = 2  [static]
```

static member for the dimension of the solution and the necessary FEM (2 because the problem is vectorial)

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

- include/LinearElasticityBasic.h

4.9 LinearElasticitySymmetric Class Reference

Inheritance diagram for LinearElasticitySymmetric:

Collaboration diagram for LinearElasticitySymmetric:

Public Member Functions

- LinearElasticitySymmetric (GetPot const &dataFile, Bulk &bulk1, Bulk &bulk2, LinearSystem &extSys)
- void assembleMatrix () override

Static Public Attributes

- static const size_type Qdim = 2

Additional Inherited Members

4.9.1 Constructor & Destructor Documentation

4.9.1.1 LinearElasticitySymmetric()

```
LinearElasticitySymmetric::LinearElasticitySymmetric (
            GetPot const & dataFile,
            Bulk & bulk1,
            Bulk & bulk2,
            LinearSystem & extSys )
```

constructor

4.9.2 Member Function Documentation

4.9.2.1 assembleMatrix()

```
void LinearElasticitySymmetric::assembleMatrix ( )  [override], [virtual]
```

overriden method for the assembly of the Matrix in the case of a linear Elasticity problem

Implements SymmetricMethod.

4.9.3 Member Data Documentation

4.9.3.1 Qdim

```
const size_type LinearElasticitySymmetric::Qdim = 2  [static]
```

static member for the dimension of the solution and the necessary FEM (2 because the problem is vectorial)

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

- include/LinearElasticitySymmetric.h

4.10 LinearSystem Class Reference

Public Member Functions

- void **addToMatrix** (int ndof)
- void copySubMatrix (sparseMatrixPtr_Type source, int first_row, int first_column, scalar_type scale=1.0, bool transpose=false)
- void addSubMatrix (sparseMatrixPtr_Type source, int first_row, int first_column, scalar_type scale=1.0, bool transpose=false)
- void extractSubMatrix (sparseMatrixPtr_Type destination, int first_row, int number_rows, int first_column, int number_cols) const
- void copySubVector (scalarVectorPtr_Type source, int first_row, scalar_type scale=1.0)
- void addSubVector (scalarVectorPtr_Type source, int first_row, scalar_type scale=1.0)
- void extractSubVector (scalarVectorPtr_Type destination, int first_row, std::string where="sol") const
- void **extractSubVector** (scalarVector_Type &destination, int first_row, std::string where="sol") const
- sparseMatrixPtr_Type **getMatrix** () const
- scalarVectorPtr_Type **getRHS** () const
- scalarVectorPtr_Type **getSol** () const
- void **addSubSystem** (LinearSystem *small, size_type shiftRows, size_type shiftColumns)
- void solve ()
- void **computeInverse** ()
- void **saveMatrix** (const char *nomefile="Matrix.mm") const
- void **multAddToRHS** (scalarVectorPtr_Type V, int first_row, int first_column, int nrows, int ncols)
- void **multAddToRHS** (sparseMatrixPtr_Type M, scalarVectorPtr_Type V, int first_rowVector, int first_rowRHS, scalar_type scale=1.0, bool transposed=false)
- void **multAddToRHS** (sparseMatrixPtr_Type M, scalarVector_Type &V, int first_rowVector, int first_rowRHS, scalar_type scale=1.0, bool transposed=false)
- void eliminateRowsColumns (std::vector< size_type > indexes)
- void **cleanRHS** ()
- void **cleanMAT** ()
- void **setNullRow** (size_type which)
- void **setNullColumn** (size_type which)
- void **setMatrixValue** (size_type i, size_type j, scalar_type value)
- void **setRHSValue** (size_type i, scalar_type value)

4.10.1 Member Function Documentation

4.10.1.1 addSubMatrix()

```
void LinearSystem::addSubMatrix (
            sparseMatrixPtr_Type source,
            int first_row,
            int first_column,
            scalar_type scale = 1.0,
            bool transpose = false )
```

method to add the given matrix to the existing system starting from first_row, first_column

4.10.1.2 addSubVector()

```
void LinearSystem::addSubVector (
            scalarVectorPtr_Type source,
            int first_row,
            scalar_type scale = 1.0 )
```

method to add the given vector to existing system starting from first_row

4.10.1.3 copySubMatrix()

```
void LinearSystem::copySubMatrix (
            sparseMatrixPtr_Type source,
            int first_row,
            int first_column,
            scalar_type scale = 1.0,
            bool transpose = false )
```

method to copy the given matrix into the existing system starting from first_row, first_column

4.10.1.4 copySubVector()

```
void LinearSystem::copySubVector (
            scalarVectorPtr_Type source,
            int first_row,
            scalar_type scale = 1.0 )
```

method to copy the given vector into the existing system starting from first_row

4.10.1.5 eliminateRowsColumns()

```
void LinearSystem::eliminateRowsColumns (
            std::vector< size_type > indexes )
```

method that eliminates the rows and columns of the matrix of the system. It takes one argument as input.

Parameters

indexes	indexes of rows and columns we want to discard from the original matrix of the system

4.10.1.6 extractSubMatrix()

```
void LinearSystem::extractSubMatrix (
            sparseMatrixPtr_Type destination,
            int first_row,
            int number_rows,
```

```
            int first_column,
            int number_cols ) const
```

method that takes the submatrix of the existing system associated to the interval of indices specified by the ints and copies it into the destination

4.10.1.7 extractSubVector()

```
void LinearSystem::extractSubVector (
            scalarVectorPtr_Type destination,
            int first_row,
            std::string where = "sol" ) const
```

method that takes the subvector of the existing system (either the solution or the rhs, according to "where") from first_row to the dimension of destination and copies it into the destination

4.10.1.8 solve()

```
void LinearSystem::solve ( )
```

resolution of the linear system

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

- include/LinearSystem.h

4.11 LifeV::Parser Class Reference

Parser - A string parser for algebraic expressions.

```
#include <Parser.h>
```

Public Types

Public Types

- typedef std::vector< std::string > **stringsVector_Type**
- typedef std::string::const_iterator **stringIterator_Type**
- typedef ParserSpiritGrammar< stringIterator_Type > **calculator_Type**
- typedef calculator_Type::results_Type **results_Type**

Public Member Functions

Constructors & Destructor

- Parser ()
  Empty constructor (it needs a manual call to setString)
- Parser (const std::string &string)
  Constructor.
- Parser (const Parser &parser)
  Copy constructor.
- virtual ~Parser ()
  Destructor.

Operators

- Parser & operator= (const Parser &parser)
  Operator =.

Methods

- const scalar_type & evaluate (const ID &id=0)
- UInt countSubstring (const std::string &substring)
- void clearVariables ()
  Clear all the variables.

Set Methods

- void setString (const std::string &string, const std::string &stringSeparator=";")
- void setVariable (const std::string &name, const scalar_type &value)

Get Methods

- const scalar_type & variable (const std::string &name)

4.11.1 Detailed Description

Parser - A string parser for algebraic expressions.

Author

(s) Cristiano Malossi, Gilles Fourestey

See `ParserSpiritGrammar` class for more details.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Parser() [1/2]

```
LifeV::Parser::Parser (
            const std::string & string )  [explicit]
```

Constructor.

Parameters

string	expression to parse

4.11.2.2 Parser() [2/2]

```
LifeV::Parser::Parser (
            const Parser & parser )  [explicit]
```

Copy constructor.

Parameters

parser	Parser

4.11.3 Member Function Documentation

4.11.3.1 countSubstring()

```
UInt LifeV::Parser::countSubstring (
            const std::string & substring )
```

Count how many times a substring is present in the string (utility for BCInterfaceFunction)

Parameters

substring	string to find

Returns

number of substring

4.11.3.2 evaluate()

```
const scalar_type& LifeV::Parser::evaluate (
            const ID & id = 0 )
```

Evaluate the expression

Parameters

id	expression index (starting from 0)

Returns

computed value

4.11.3.3 operator=()

```
Parser& LifeV::Parser::operator= (
            const Parser & parser )
```

Operator =.

Parameters

parser	Parser

Returns

reference to a copy of the class

4.11.3.4 setString()

```
void LifeV::Parser::setString (
            const std::string & string,
            const std::string & stringSeparator = ";" )
```

Set string function

Parameters

string	Expression to evaluate
stringSeparator	Separator identifier (default -> ";")

4.11.3.5 setVariable()

```
void LifeV::Parser::setVariable (
            const std::string & name,
            const scalar_type & value )
```

Set/replace a variable

Parameters

name	name of the parameter
value	value of the parameter

4.11.3.6 variable()

```
const scalar_type& LifeV::Parser::variable (
            const std::string & name )
```

Get variable

Parameters

name	name of the parameter

Returns

value of the variable

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

- include/Parser.h

4.12 LifeV::ParserSpiritGrammar< IteratorType, ResultsType > Class Template Reference

ParserSpiritGrammar - A string parser grammar based on `boost::spirit::qi`.

```
#include <ParserSpiritGrammar.h>
```

Inheritance diagram for LifeV::ParserSpiritGrammar< IteratorType, ResultsType >:

Collaboration diagram for LifeV::ParserSpiritGrammar< IteratorType, ResultsType >:

Public Types

Public Types

- typedef IteratorType **iterator_Type**
- typedef boost::iterator_range< iterator_Type > **iteratorRange_Type**
- typedef ResultsType **results_Type**

Public Member Functions

Constructors & Destructor

- ParserSpiritGrammar ()
  Constructor.
- ParserSpiritGrammar (const ParserSpiritGrammar &spiritGrammar)
  Copy constructor.
- virtual ~ParserSpiritGrammar ()
  Destructor.

Operators

- ParserSpiritGrammar & operator= (const ParserSpiritGrammar &spiritGrammar)
  Operator =.

Methods

- void assignVariable (const iteratorRange_Type &stringIteratorRange, const scalar_type &value)
- void clearVariables ()
  Clear all the variables.

Set Methods

- void setDefaultVariables ()
  Set default variables.
- void setVariable (const std::string &name, const scalar_type &value)

Get Methods

- scalar_type & variable (const std::string &name)

4.12.1 Detailed Description

template<typename IteratorType = std::string::const_iterator, typename ResultsType = std::vector < scalar_type >>
class LifeV::ParserSpiritGrammar< IteratorType, ResultsType >

ParserSpiritGrammar - A string parser grammar based on `boost::spirit::qi`.

Author

(s) Cristiano Malossi

`ParserSpiritGrammar` is a `boost::spirit::qi` based class to perform evaluation of `std::string` expressions.

EXAMPLE - HOW TO USE Let's consider the following example: suppose that we have this function:

[u,v,w] = f(x,y,z,t)

where

u(x) = a∗b∗x v(x,y) = a/b∗sqrt(x^2 + y^2) w(t) = b∗t;

with "a" and "b" constants such that a=5.12345, b=9.999999.

To evaluate function f(x,y,z,t), we use this syntax:

string = "a=5.12345 ; b=9.999999 ; (a∗b∗x, a/b∗sqrt(x^2 + y^2), b∗t)"

where semicolons (";") separate constants and commas (",") separate output functions.

NOTE: Currently ParserSpiritGrammar works with the following operators:

```
*  +, -, *, /, ^, sqrt(), sin(), cos(), tan(), exp(), log(), log10(), >, <.
*
```

4.12.2 Constructor & Destructor Documentation

4.12.2.1 ParserSpiritGrammar()

```
template<typename IteratorType , typename ResultsType >
LifeV::ParserSpiritGrammar< IteratorType, ResultsType >::ParserSpiritGrammar (
            const ParserSpiritGrammar< IteratorType, ResultsType > & spiritGrammar )  [explicit]
```

Copy constructor.

Parameters

ParserSpiritGrammar	ParserSpiritGrammar

4.12.3 Member Function Documentation

4.12.3.1 assignVariable()

```
template<typename IteratorType = std::string::const_iterator, typename ResultsType = std↩
::vector < scalar_type >>
void LifeV::ParserSpiritGrammar< IteratorType, ResultsType >::assignVariable (
            const iteratorRange_Type & stringIteratorRange,
            const scalar_type & value )  [inline]
```

Assign a variable using a `boost::iterator_range`

Parameters

stringIteratorRange	name of the parameter
value	value of the parameter

4.12.3.2 operator=()

```
template<typename IteratorType , typename ResultsType >
ParserSpiritGrammar< IteratorType, ResultsType > & LifeV::ParserSpiritGrammar< IteratorType,
ResultsType >::operator= (
            const ParserSpiritGrammar< IteratorType, ResultsType > & spiritGrammar )
```

Operator =.

Parameters

SpiritGrammar	ParserSpiritGrammar

Returns

reference to a copy of the class

4.12.3.3 setVariable()

```
template<typename IteratorType , typename ResultsType >
void LifeV::ParserSpiritGrammar< IteratorType, ResultsType >::setVariable (
             const std::string & name,
             const scalar_type & value )  [inline]
```

Set/replace a variable

Parameters

name	name of the parameter
value	value of the parameter

4.12.3.4 variable()

```
template<typename IteratorType = std::string::const_iterator, typename ResultsType = std↩
::vector < scalar_type >>
scalar_type& LifeV::ParserSpiritGrammar< IteratorType, ResultsType >::variable (
             const std::string & name )  [inline]
```

Get variable

Parameters

name	name of the parameter

Returns

value of the variable

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

- include/ParserSpiritGrammar.h

4.13 Problem Class Reference

Inheritance diagram for Problem:

Collaboration diagram for Problem:

Public Member Functions

- Problem (GetPot const &dataFile, std::string const problem, Bulk &bulk1, Bulk &bulk2, const size_type dim, LinearSystem &extSys)
- FEM **getFEM** (size_type const idx) const
- LinearSystem & **getSYS** () const
- size_type **getNDOF** (std::string const variable="all") const
- scalar_type **getL2ERR** () const
- scalar_type **getH1ERR** () const
- virtual void assembleMatrix ()=0
- virtual void assembleRHS ()=0
- virtual void enforceStrongBC (size_type const domainIdx)=0
- virtual void treatIFaceDofs ()=0
- virtual void solve ()=0
- void extractSol (scalarVector_Type &destSol, std::string const variable="all")
- void exportVtk (std::string const folder="./vtk", std::string const what="all")
- void computeErrors ()
- void printErrors (std::string const filename1, std::string const filename2, std::string const test)
- virtual ~Problem ()

Protected Attributes

- Bulk & **M_Bulk1**
- Bulk & **M_Bulk2**
- BC **M_BC1**
- BC **M_BC2**
- size_type **interfaceIdx1**
- size_type **interfaceIdx2**
- FEM **M_uFEM1**
- FEM **M_uFEM2**
- FEM **M_CoeffFEM1**
- FEM **M_CoeffFEM2**
- getfem::mesh_im **M_intMethod1**
- getfem::mesh_im **M_intMethod2**
- LinearSystem & **M_Sys**
- BulkDatum **M_exact_sol1**
- BulkDatum **M_exact_sol2**
- BulkDatum **M_source1**
- BulkDatum **M_source2**
- scalarVector_Type **M_uSol**
- scalarVector_Type **M_uSol1**
- scalarVector_Type **M_uSol2**
- size_type **M_nbDOF1**
- size_type **M_nbDOF2**
- size_type **M_nbTotDOF**
- size_type **M_nbDOFIFace**

- sizeVector_Type **dof_IFace1**
- sizeVector_Type **dof_IFace2**
- sizeVector_Type **M_rowsStrongBC1**
- sizeVector_Type **M_rowsStrongBC2**
- sizeVector_Type **M_rowsIFace1**
- sizeVector_Type **M_rowsIFace2**
- sizeVector_Type **M_rowsStrongBCFlags1**
- sizeVector_Type **M_rowsStrongBCFlags2**
- sizeVector_Type **M_rowsIFace**
- scalar_type **errL2**
- scalar_type **errH1**

4.13.1 Constructor & Destructor Documentation

4.13.1.1 Problem()

```
Problem::Problem (
            GetPot const & dataFile,
            std::string const problem,
            Bulk & bulk1,
            Bulk & bulk2,
            const size_type dim,
            LinearSystem & extSys )
```

constructor

4.13.1.2 ~Problem()

```
virtual Problem::~Problem ( )  [inline], [virtual]
```

destructor

4.13.2 Member Function Documentation

4.13.2.1 assembleMatrix()

```
virtual void Problem::assembleMatrix ( )  [pure virtual]
```

pure virtual method for the assembly of the matrix

Implemented in BasicMethod, SymmetricMethod, LinearElasticityBasic, LaplacianBasic, LaplacianSymmetric, and LinearElasticitySymmetric.

4.13.2.2 assembleRHS()

```
virtual void Problem::assembleRHS ( )  [pure virtual]
```

pure virtual method for the assembly of the RHS

Implemented in BasicMethod, and SymmetricMethod.

4.13.2.3 computeErrors()

```
void Problem::computeErrors ( )
```

method to compute errors in L^2 and H^1 norm

4.13.2.4 enforceStrongBC()

```
virtual void Problem::enforceStrongBC (
            size_type const domainIdx )  [pure virtual]
```

pure virtual method for the imposition of the Dirichlet boundary conditions

Implemented in BasicMethod, and SymmetricMethod.

4.13.2.5 exportVtk()

```
void Problem::exportVtk (
            std::string const folder = "./vtk",
            std::string const what = "all" )
```

method to export the solution in ./vtk extension

4.13.2.6 extractSol()

```
void Problem::extractSol (
            scalarVector_Type & destSol,
            std::string const variable = "all" )
```

method to extract in "destSol" the values of the variable specified by "variable", taking them from the global solution M_uSol; the default value of the std::string implies that the whole solutions M_uSol is extracted.

4.13.2.7 printErrors()

```
void Problem::printErrors (
            std::string const filename1,
            std::string const filename2,
            std::string const test )
```

method to print errors of the "test" in L^2 norm in "filename1" and the errors in H^1 norm in "filename2"

4.13.2.8 solve()

```
virtual void Problem::solve ( )  [pure virtual]
```

pure virtual method for the resolution of the linear system

Implemented in BasicMethod, and SymmetricMethod.

4.13.2.9 treatIFaceDofs()

```
virtual void Problem::treatIFaceDofs ( )  [pure virtual]
```

pure virtual method for the treatment of the interface degrees of freedom

Implemented in BasicMethod, and SymmetricMethod.

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

- include/Problem.h

4.14 SymmetricMethod Class Reference

Inheritance diagram for SymmetricMethod:

Collaboration diagram for SymmetricMethod:

Public Member Functions

- SymmetricMethod (GetPot const &dataFile, std::string const problem, Bulk &bulk1, Bulk &bulk2, const size↩ _type dim, LinearSystem &extSys)
- virtual void assembleMatrix ()=0
- void assembleRHS () override
- void enforceStrongBC (size_type const domainIdx) override

  overriden method for the strong Dirichlet boundary conditions
- void treatIFaceDofs () override
- void solve () override
- virtual ~SymmetricMethod ()

Protected Attributes

- BulkDatum **M_jump1**
- BulkDatum **M_jump2**
- scalarVector_Type **M_q01**
- scalarVector_Type **M_q02**

4.14.1 Constructor & Destructor Documentation

4.14.1.1 SymmetricMethod()

```
SymmetricMethod::SymmetricMethod (
            GetPot const & dataFile,
            std::string const problem,
            Bulk & bulk1,
            Bulk & bulk2,
            const size_type dim,
            LinearSystem & extSys )
```

constructor

4.14.1.2 ~SymmetricMethod()

```
virtual SymmetricMethod::~SymmetricMethod ( )  [inline], [virtual]
```

destructor

4.14.2 Member Function Documentation

4.14.2.1 assembleMatrix()

```
virtual void SymmetricMethod::assembleMatrix ( )  [pure virtual]
```

a pure virtual method for the assembly of the matrix

Implements Problem.

Implemented in LaplacianSymmetric, and LinearElasticitySymmetric.

4.14.2.2 assembleRHS()

```
void SymmetricMethod::assembleRHS ( )  [override], [virtual]
```

overriden method for the assembly of the right hand side

Implements Problem.

4.14.2.3 enforceStrongBC()

```
void SymmetricMethod::enforceStrongBC (
            size_type const domainIdx )  [override], [virtual]
```

overriden method for the strong Dirichlet boundary conditions

This methods takes one argument as input

Parameters

domainIdx	index to identify if we are in the first or in the second domain

Implements Problem.

4.14.2.4 solve()

```
void SymmetricMethod::solve ( )  [override], [virtual]
```

overriden method for the resolution of the linear system

Implements Problem.

4.14.2.5 treatIFaceDofs()

```
void SymmetricMethod::treatIFaceDofs ( )  [override], [virtual]
```

overriden method for the management of the interface degrees of freedom

Implements Problem.

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

- include/SymmetricMethod.h

Chapter 5

File Documentation

5.1 include/BasicMethod.h File Reference

An abstract class to group the common features of the basic formulation.

```
#include "LinearSystem.h"
#include "Operators.h"
#include "UsefulFunctions.h"
#include "StringUtility.h"
#include "Problem.h"
```

Include dependency graph for BasicMethod.h:

5.2 include/BC.h File Reference

This is a class for the management of the boundary conditions.

```
#include "Core.h"
#include "Parser.h"
```

Include dependency graph for BC.h: This graph shows which files directly or indirectly include this file:

Classes

- class BC

5.2.1 Detailed Description

This is a class for the management of the boundary conditions.

This class contains the functions to evaluate the boundary conditions and is generic with respect to the PDEs system we want to solve.

5.3 include/Bulk.h File Reference

This class is for the management of a 2-dimensional domain.

```
#include "Core.h"
#include "Parser.h"
```

Include dependency graph for Bulk.h: This graph shows which files directly or indirectly include this file:

Classes

- class Bulk

5.3.1 Detailed Description

This class is for the management of a 2-dimensional domain.

This class contains all the data related to the domain and its triangulation. It does not include data regarding the specific problem to solve.

5.4 include/BulkDatum.h File Reference

This is a class for any kind of data related to the problem.

```
#include "Core.h"
#include "Parser.h"
```

Include dependency graph for BulkDatum.h: This graph shows which files directly or indirectly include this file:

Classes

- class BulkDatum

5.4.1 Detailed Description

This is a class for any kind of data related to the problem.

This class is used to store the string describing the parameters (diffusion, Lamè...) and the functions (forcing, exact solution) related to the problem we are interested in. They can be both scalars and vectors.

5.5 include/Core.h File Reference

This file contains all the necessary "include" and definition of Getfem++ types we will be using.

```
#include <getfem/getfem_regular_meshes.h>
#include <getfem/getfem_interpolation.h>
#include <getfem/getfem_derivatives.h>
#include <getfem/getfem_config.h>
#include <getfem/getfem_assembling.h>
#include <getfem/getfem_import.h>
#include <getfem/getfem_export.h>
#include <gmm/gmm.h>
#include <gmm/gmm_inoutput.h>
#include <gmm/gmm_MUMPS_interface.h>
#include <gmm/gmm_superlu_interface.h>
#include <getfem/bgeot_mesh.h>
#include "GetPot"
#include <string>
#include <memory>
#include <iostream>
#include <iomanip>
```

Include dependency graph for Core.h: This graph shows which files directly or indirectly include this file:

Typedefs

- using **sparseVector_Type** = gmm::rsvector< scalar_type >
- using **sparseMatrix_Type** = gmm::row_matrix< sparseVector_Type >
- using **sparseMatrixPtr_Type** = std::shared_ptr< sparseMatrix_Type >
- using **scalarVector_Type** = std::vector< scalar_type >
- using **scalarVectorPtr_Type** = std::shared_ptr< scalarVector_Type >
- using **sizeVector_Type** = std::vector< size_type >
- using **sizeVectorPtr_Type** = std::shared_ptr< sizeVector_Type >
- using **LifeV::Double** = scalar_type
- using **LifeV::UInt** = size_type
- using **LifeV::ID** = size_type
- using **LifeV::Int** = int

5.5.1 Detailed Description

This file contains all the necessary "include" and definition of Getfem++ types we will be using.

5.6 include/FEM.h File Reference

This class contains all the necessary features for a generic finite element method.

```
#include "Core.h"
```

Include dependency graph for FEM.h: This graph shows which files directly or indirectly include this file:

Classes

- class FEM

5.6.1 Detailed Description

This class contains all the necessary features for a generic finite element method.

This class is actually a wrapper of the already existing getfem::fem class in the library Getfem++. It incorporates only the instruments that were really needed, such as the problem's dimension, the definition of the degrees of freedom and the selection f the functional space for the trial and test functions.

5.7 include/LaplacianBasic.h File Reference

This is the class for the management of a Laplacian problem with the basic formulation.

`#include "BasicMethod.h"`
Include dependency graph for LaplacianBasic.h:

Classes

- class LaplacianBasic

5.7.1 Detailed Description

This is the class for the management of a Laplacian problem with the basic formulation.

It is a final class of the hierarchy, where the method for the assembly of the matrix has been overriden.

5.8 include/LaplacianSymmetric.h File Reference

This is the class for the management of a Laplacian problem with the symmetric formulation.

`#include "SymmetricMethod.h"`
Include dependency graph for LaplacianSymmetric.h:

Classes

- class LaplacianSymmetric

5.8.1 Detailed Description

This is the class for the management of a Laplacian problem with the symmetric formulation.

It is a final class of the hierarchy, where the method for the assembly of the matrix has been overriden

5.9 include/LinearElasticityBasic.h File Reference

This is the class for the management of a linear Elasticity problem with the basic formulation.

```
#include "BasicMethod.h"
```
Include dependency graph for LinearElasticityBasic.h:

Classes

- class LinearElasticityBasic

5.9.1 Detailed Description

This is the class for the management of a linear Elasticity problem with the basic formulation.

It is a final class of the hierarchy, where the method for the assembly of the matrix has been overriden.

5.10 include/LinearElasticitySymmetric.h File Reference

This is the class for the management of a linear Elasticity problem with the symmetric formulation.

```
#include "SymmetricMethod.h"
```
Include dependency graph for LinearElasticitySymmetric.h:

Classes

- class LinearElasticitySymmetric

5.10.1 Detailed Description

This is the class for the management of a linear Elasticity problem with the symmetric formulation.

It is a final class of the hierarchy, where the method for the assembly of the matrix has been overriden.

5.11 include/LinearSystem.h File Reference

This is the class for the management of a linear system.

```
#include "Core.h"
```
Include dependency graph for LinearSystem.h: This graph shows which files directly or indirectly include this file:

Classes

- class LinearSystem

5.11.1 Detailed Description

This is the class for the management of a linear system.

This class is endowed with different methods that let us copy, add or extract parts of the matrix, of the right hand side or of the solution vector. It also provides functions to get and set values, to save the matrix and algorithms that invert the matrix and solve the linear system.

5.12 include/OperatorsBD.h File Reference

This file includes the method for the evaluation of natural boundary conditions.

```
#include "Core.h"
#include "FEM.h"
#include "Bulk.h"
#include "BC.h"
```

Include dependency graph for OperatorsBD.h: This graph shows which files directly or indirectly include this file:

Functions

- void stressRHS (scalarVectorPtr_Type V, const Bulk &medium, BC &bcRef, const FEM &femSol, const FEM &femDatum, const getfem::mesh_im &im)

5.12.1 Detailed Description

This file includes the method for the evaluation of natural boundary conditions.

5.12.2 Function Documentation

5.12.2.1 stressRHS()

```
void stressRHS (
            scalarVectorPtr_Type V,
            const Bulk & medium,
            BC & bcRef,
            const FEM & femSol,
            const FEM & femDatum,
            const getfem::mesh_im & im )
```

method to evaluate the Neumann boundary conditions using the function getfem::asm_source_term

5.13 include/OperatorsBulk.h File Reference

This file assembles different methods related to the bulk that can be employed in several contexts.

```
#include "Core.h"
#include "FEM.h"
#include "Bulk.h"
#include "BC.h"
#include "BulkDatum.h"
```

Include dependency graph for OperatorsBulk.h: This graph shows which files directly or indirectly include this file:

Functions

- void stiffness (sparseMatrixPtr_Type M, const FEM &femSol, const FEM &femCoef, BulkDatum &Diff, const getfem::mesh_im &im)
- void linearElasticity (sparseMatrixPtr_Type M, const FEM &femSol, const FEM &femCoef, BulkDatum &Mu, BulkDatum &Lambda, const getfem::mesh_im &im)
- void bulkLoad (scalarVectorPtr_Type V, const FEM &FemSol, const FEM &FemSource, BulkDatum &Source, const getfem::mesh_im &im)
- void exactSolution (scalarVectorPtr_Type V, const FEM &FemSol, BulkDatum &Solution)
- void jump (scalarVectorPtr_Type V, const FEM &FemSol, BulkDatum &Jump)

5.13.1 Detailed Description

This file assembles different methods related to the bulk that can be employed in several contexts.

5.13.2 Function Documentation

5.13.2.1 bulkLoad()

```
void bulkLoad (
            scalarVectorPtr_Type V,
            const FEM & FemSol,
            const FEM & FemSource,
            BulkDatum & Source,
            const getfem::mesh_im & im )
```

method to compute the volumetric source term

5.13.2.2 exactSolution()

```
void exactSolution (
            scalarVectorPtr_Type V,
            const FEM & FemSol,
            BulkDatum & Solution )
```

method to evaluate the exact solution in every degree of freedom of the FEM

5.13.2.3 jump()

```
void jump (
             scalarVectorPtr_Type V,
             const FEM & FemSol,
             BulkDatum & Jump )
```

method to evaluate the jump in every degree of freedom of the FEM

5.13.2.4 linearElasticity()

```
void linearElasticity (
             sparseMatrixPtr_Type M,
             const FEM & femSol,
             const FEM & femCoef,
             BulkDatum & Mu,
             BulkDatum & Lambda,
             const getfem::mesh_im & im )
```

method to compute the stiffness matrix for the linear elasticity problem

5.13.2.5 stiffness()

```
void stiffness (
             sparseMatrixPtr_Type M,
             const FEM & femSol,
             const FEM & femCoef,
             BulkDatum & Diff,
             const getfem::mesh_im & im )
```

method to compute the stiffness matrix for the elliptic scalar problem

5.14 include/Parser.h File Reference

File containing the Parser interface.

```
#include "Core.h"
#include "ParserDefinitions.h"
#include "ParserSpiritGrammar.h"
```

Include dependency graph for Parser.h: This graph shows which files directly or indirectly include this file:

Classes

- class LifeV::Parser

  Parser - A string parser for algebraic expressions.

5.14.1 Detailed Description

File containing the Parser interface.

Date

07-04-2009

Author

Cristiano Malossi cristiano.malossi@epfl.ch

@contributor Gilles Fourestey gilles.fourestey@epfl.ch @maintainer Cristiano Malossi cristiano.malossi@epfl.ch

5.15 include/ParserDefinitions.h File Reference

File containing the Parser definitions.

```
#include <map>
#include <iomanip>
#include <string>
#include <algorithm>
#include <boost/algorithm/string.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/spirit/include/qi.hpp>
#include <boost/spirit/include/phoenix_bind.hpp>
#include <boost/spirit/include/phoenix_operator.hpp>
```

Include dependency graph for ParserDefinitions.h: This graph shows which files directly or indirectly include this file:

5.15.1 Detailed Description

File containing the Parser definitions.

Date

29-01-2010

Author

Cristiano Malossi cristiano.malossi@epfl.ch

@maintainer Cristiano Malossi cristiano.malossi@epfl.ch

5.16 include/ParserSpiritGrammar.h File Reference

File containing the Parser grammar.

```
#include "ParserDefinitions.h"
```

Include dependency graph for ParserSpiritGrammar.h: This graph shows which files directly or indirectly include this file:

Classes

- class LifeV::ParserSpiritGrammar< IteratorType, ResultsType >

  ParserSpiritGrammar - A string parser grammar based on `boost::spirit::qi`.

5.16.1 Detailed Description

File containing the Parser grammar.

Date

05-02-2010

Author

Cristiano Malossi cristiano.malossi@epfl.ch

@maintainer Cristiano Malossi cristiano.malossi@epfl.ch

5.17 include/Problem.h File Reference

This is the base abstract class. It contains all the methods andattributes that both the “symmetric” and the “basic” approach need without specialization.

```
#include "LinearSystem.h"
#include "Operators.h"
#include "UsefulFunctions.h"
#include "StringUtility.h"
```

Include dependency graph for Problem.h: This graph shows which files directly or indirectly include this file:

Classes

- class Problem

5.17.1 Detailed Description

This is the base abstract class. It contains all the methods andattributes that both the “symmetric” and the “basic” approach need without specialization.

This class contains the functions that extract the solution and compute or print errors in the L^2 and H^1 norm. The methods which deeply change according to the PDEs system and the algebraic formulation have been defined virtual. These are the ones for the construction of the matrix, of the right-hand-sideand the treatment of the boundary and interface conditions and the resolution of the linear system.

5.18 include/StringUtility.h File Reference

std::string utilities

```
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iosfwd>
#include <iostream>
#include <list>
#include <map>
#include <sstream>
#include <string>
#include <vector>
#include <boost/algorithm/string.hpp>
#include "Core.h"
```

Include dependency graph for StringUtility.h: This graph shows which files directly or indirectly include this file:

Functions

- std::istream & LifeV::eatLine (std::istream &s)
- std::istream & LifeV::eatComments (std::istream &s)

  skip lines starting with '!%#;$'
- std::istream & LifeV::nextGoodLine (std::istream &s, std::string &line)

  gets next uncommented line
- std::string & LifeV::setStringLength (std::string &s, unsigned int len, char c)
- int LifeV::atoi (const std::string &s)

  extends atoi to STL std::strings (from Stroustrup)
- std::string **LifeV::operator+** (const std::string &str, const int i)
- std::string **LifeV::operator+** (const std::string &str, const long int i)
- std::string **LifeV::operator+** (const std::string &str, const unsigned int i)
- template<typename EntryType >
  void **LifeV::parseList** (const std::string &slist, std::list< EntryType > &list)
- double **LifeV::string2number** (const std::string &s)
- template<typename NumberType >
  std::string **LifeV::number2string** (const NumberType &n)
- template<typename EnumeratorType >
  std::string **LifeV::enum2String** (const EnumeratorType &Enum, const std::map< std::string, Enumerator↩
  Type > &Map)
- template<typename NumberType >
  void **LifeV::string2numbersVector** (const std::string &string, std::vector< NumberType > &numberVector)

5.18.1 Detailed Description

std::string utilities

Date

13-12-2010

Author

@maintainer Radu Popescu radu.popescu@epfl.ch

5.18.2 Function Documentation

5.18.2.1 eatLine()

```
std::istream& LifeV::eatLine (
            std::istream & s )
```

It gets a the next line from std::istream

5.18.2.2 setStringLength()

```
std::string& LifeV::setStringLength (
            std::string & s,
            unsigned int len,
            char c )
```

always return a std::string with len characters

- if the s has more than len characters : keep only the first len
- if the s has less than len characters : complete with c until len

5.19 include/SymmetricMethod.h File Reference

An abstract class to group the common features of the symmetric formulation.

```
#include "LinearSystem.h"
#include "Operators.h"
#include "UsefulFunctions.h"
#include "StringUtility.h"
#include "Problem.h"
```

Include dependency graph for SymmetricMethod.h: This graph shows which files directly or indirectly include this file:

Classes

- class SymmetricMethod

5.19.1 Detailed Description

An abstract class to group the common features of the symmetric formulation.

This class inherits from the "Problem" class and specializes the methods for the assembly of the matrix, the assembly of the right hand side, the imposition of the boundary and interface conditions and the function for the resolution of the linear system.

Index