FEME - Finite Element Method Environment

Generated by Doxygen 1.8.13

Thu Dec 19 2019 11:36:49

Contents

1 Class Index

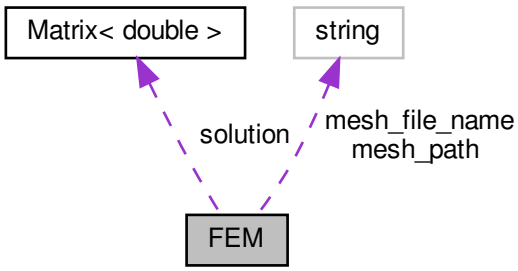
1.1 Class List

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

2 Class Documentation

2.1 FEM Class Reference

Collaboration diagram for FEM:



Public Member Functions

- void run ()

  Runs the Finite Element Method.

Public Attributes

- string mesh_path

  Specify the .msh file path.
- string mesh_file_name

  Specify the .msh file name.
- vector< int > setup_phys_region_ID

  Specify physical regions ID.
- vector< double > setup_phys_region_perm_rel

  Specify physical regions material property.

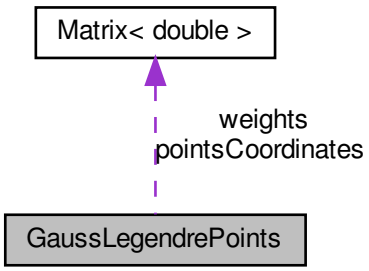
- vector< double > setup_phys_region_excitation
  Specify physical regions excitations.
- vector< int > setup_phys_BC_ID
  Specify the physical boundary conditions ID.
- vector< double > setup_phys_BC_val
  Specify the boundary condition values for each ID.
- Matrix< double > solution
  Acess the solution.

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

- include/FEM.h
- src/FEM.cpp

2.2 GaussLegendrePoints Class Reference

Collaboration diagram for GaussLegendrePoints:



Public Member Functions

- GaussLegendrePoints (int ElemType)

Public Attributes

- Matrix< double > pointsCoordinates
- Matrix< double > weights

2.2.1 Constructor & Destructor Documentation

2.2.1.1 GaussLegendrePoints()

```
GaussLegendrePoints::GaussLegendrePoints (
            int ElemType )
```

Calculates the Gauss points

Parameters

ElemType	element type

2.2.2 Member Data Documentation

2.2.2.1 pointsCoordinates

```
Matrix<double> GaussLegendrePoints::pointsCoordinates
```

Local coordinates

Returns

Matrix<double> (n,3) [u1 v1 p1, u2 v2 p2, un vn pn]

2.2.2.2 weights

```
Matrix<double> GaussLegendrePoints::weights
```

Weight for each point

Returns

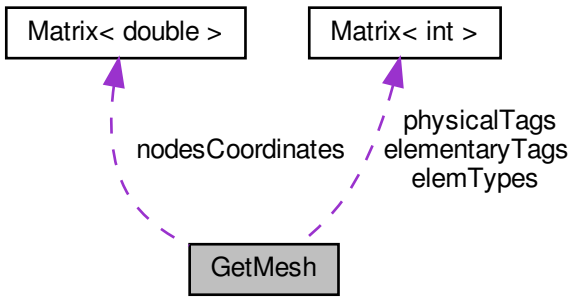
Matrix<double>(1,n) [W1 w2 ... Wn]

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

- include/ShapeFunctions.h
- src/ShapeFunctions.cpp

2.3 GetMesh Class Reference

Collaboration diagram for GetMesh:



Public Member Functions

- GetMesh (string filePath)

Public Attributes

- Matrix< double > nodesCoordinates

  Access the nodes coordinates.
- Matrix< int > elemTypes

  Access the element types.
- Matrix< int > physicalTags

  Access the physical ID tags.
- Matrix< int > elementaryTags

  Access the elementary tags.
- int numElemments

  Total number of elements.
- int numNodes

  Access the number of nodes.
- int numElements1D

  Access the number of 1D elements.
- int numElements2D

  Access the number of 2D elements.
- vector< vector< int > > elemNodes2D
- vector< vector< int > > elemNodes1D
- TODO **__pad0__**: check how to implement the destructor ~GetMesh()

2.3.1 Constructor & Destructor Documentation

2.3.1.1 GetMesh()

```
GetMesh::GetMesh (
            string filePath )
```

Reads the mesh from a .msh file .msh mesh version is 2.x

Parameters

filePath	file path (with name) of the .msh file

2.3.2 Member Data Documentation

2.3.2.1 elemNodes1D

```
vector<vector<int> > GetMesh::elemNodes1D
```

Nodes of each 2D element It uses vector<vector<int>> because the number of nodes may vary in meshes with different element types

2.3.2.2 elemNodes2D

`vector<vector<int> > GetMesh::elemNodes2D`

Nodes of each 1D element It uses vector<vector<int>> because the number of nodes may vary in meshes with different element types

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

- include/Gmsh_interface.h
- src/Gmsh_interface.cpp

2.4 Material_constants Class Reference

Public Member Functions

- Material_constants ()

  Set vacuum constants.

Public Attributes

- double mu0

  Access the vacuum permeability $$.
- double eps0

  Access the vacuum permitivity $$.

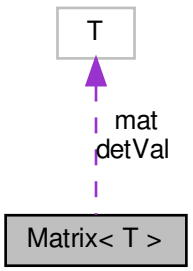
The documentation for this class was generated from the following files:

- include/Material_constants.h
- src/Material_constants.cpp

2.5 Matrix< T > Class Template Reference

`#include <Matrix.h>`

Collaboration diagram for Matrix< T >:



Public Member Functions

- Matrix ()
- Matrix (int n, int m)
- **Matrix** (const Matrix &)
- void Alloc (int n, int m)

  Allocate a Matrix.
- void SolveLinearSystem (Matrix &lhs, Matrix &rhs)
- Matrix Transpose ()

  Transpose the matrix.
- Matrix Inverse ()

  Inverse a matrix.
- void SetLineValue (int line, T value)
- void SetValue (T)

  Set a value to the entire matrix.
- void print_matrix ()

  Print the matrix.
- void calcDet ()

  Calculates the determinant.
- void writeToFile (string path, string fileName)
- void write2DVectorToFile (vector< vector< int >> twoDArrayData, string path, string fileName)
- Matrix & **operator=** (Matrix)
- Matrix **operator*** (const Matrix &)
- Matrix **operator+** (const Matrix &)
- Matrix **operator*** (T const &)

Public Attributes

- T ** mat

  Access to the array.
- int rows

  Access the number of rows.
- int cols

  Access the number of cols.
- T detVal

  Access the determinant.

2.5.1 Detailed Description

template<class T>
class Matrix< T >

Provides 2D array object with contiguous memory allocation ...

Parameters

T	array type

2.5.2 Constructor & Destructor Documentation

2.5.2.1 Matrix() [1/2]

```
template<class T >
Matrix< T >::Matrix ( )
```

Creates a 2D array with (0,0) rows and columns

2.5.2.2 Matrix() [2/2]

```
template<class T >
Matrix< T >::Matrix (
            int n,
            int m )
```

Creates a 2D array with (rows,cols) ...

Parameters

n	number of rows
m	number of cols

2.5.3 Member Function Documentation

2.5.3.1 SetLineValue()

```
template<class T>
void Matrix< T >::SetLineValue (
            int line,
            T value )
```

Set a value to a entire line of the matrix ...

Parameters

line	line to set
value	value to set the entire line

2.5.3.2 SolveLinearSystem()

```
template<class T >
void Matrix< T >::SolveLinearSystem (
```

```
            Matrix< T > & lhs,
            Matrix< T > & rhs )
```

Solve a linear system using the Lapack DGESV ...

Parameters

lhs	left hand side
rhd	right hand side

Returns

Return the result in the rhs

2.5.3.3 write2DVectorToFile()

```
template<class T >
void Matrix< T >::write2DVectorToFile (
            vector< vector< int >> twoDArrayData,
            string path,
            string fileName )
```

Writes a vector<vector<int>> to a txt file

Parameters

twoDArrayData	data to write
path	directory
fileName	file name

2.5.3.4 writeToFile()

```
template<class T >
void Matrix< T >::writeToFile (
            string path,
            string fileName )
```

Writes a matrix to a txt file ...

Parameters

path	directory
fileName	file name

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

- include/Matrix.h
- src/Matrix.cpp

2.6 Messages Class Reference

Public Member Functions

- logMessage (string message)
- void NotImplementedElement (int elemType, string whereHapp)

Public Attributes

- TODO **__pad0__**: check whether it is necessary ~Messages()

2.6.1 Member Function Documentation

2.6.1.1 logMessage()

```
string Messages::logMessage (
            string message )
```

Prints a message

Parameters

message	message to print

2.6.1.2 NotImplementedElement()

```
void Messages::NotImplementedElement (
            int elemType,
            string whereHapp )
```

Print a specific message of non-implemented element

Parameters

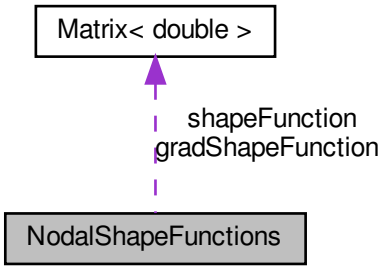
elemType	element type
whereHapp	function where it happened

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

- include/Messages.h
- src/Messages.cpp

2.7 NodalShapeFunctions Class Reference

Collaboration diagram for NodalShapeFunctions:



Public Member Functions

- void GetNodalShapeFunctions (int ElemType, double u, double v, double p)
- void GetGradNodalShapeFunction (int ElemType, double u=0, double v=0, double p=0)

Public Attributes

- Matrix< double > shapeFunction
- int **ElementType**
- Matrix< double > gradShapeFunction

2.7.1 Member Function Documentation

2.7.1.1 GetGradNodalShapeFunction()

```
void NodalShapeFunctions::GetGradNodalShapeFunction (
            int ElemType,
            double u = 0,
            double v = 0,
            double p = 0 )
```

Calculates the gadient of nodal shape functions

Parameters

ElemType	element type
u,v,p	local coordinates

Returns

2D Matrix [dN1/du dN2/du ... dNn/du, dN1/dv dN2/dv ... dNn/dv, dN1/dp dN2/dp ... dNn/dp]

2.7.1.2 GetNodalShapeFunctions()

```
void NodalShapeFunctions::GetNodalShapeFunctions (
            int ElemType,
            double u,
            double v,
            double p )
```

Calculates the nodal shape functions

Parameters

ElemType	element type
u,v,p	local coordinates

Returns

2D Matrix [N1 N2...Nn]

2.7.2 Member Data Documentation

2.7.2.1 gradShapeFunction

```
Matrix<double> NodalShapeFunctions::gradShapeFunction
```

Access the gadient of nodal shape functions values

2.7.2.2 shapeFunction

```
Matrix<double> NodalShapeFunctions::shapeFunction
```

Access the nodal shape functions values

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

- include/ShapeFunctions.h
- src/ShapeFunctions.cpp

2.8 PerformanceTestes Class Reference

Public Member Functions

- void vector_matrix ()

  Performance test of allocanting a high number of small vector<double> and Matrix<double>
- void **VectorMatrixMult** ()

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

- include/PerformanceTestes.h
- src/PerformanceTestes.cpp

Index