**Finite Element Method implemented in C++**

Argyrios Ntoumanis

July 2022

### Class list

* Matrix
* Mesh
* Basis

### Class Documentation

#### 2.1 Matrix

Define a matrix entity either by specifying amount of rows and columns or by copying another Matrix

Public member functions

* void setValue(int row, int col, double val);

sets the element ij equals to val

* void calcDet();

calculates the determinant

* void print();

prints the matrix

* Matrix transpose();

returns the transpose matrix

* Matrix inverse();

returns the inverse matrix

* double norm(std::vector<double>& v);

computes the 2-norm of vector v

* std::vector<double> linsolve(std::vector<double> b, int iters);

solves the linear system defined of the specified matrix and right hand side b, using the Jacobi iterative method

Public member variables

* int rows;

the number of rows

* int cols;

the number of columns

* double\*\* mat;

stores the elements of the matrix in a 2d dynamic array

* double detVal;

the determinant

#### 2.2 Mesh

Define a mesh entity either by specifying node and connectivity arrays or by copying another Mesh

Public member functions

* void calcInteriorNodes();

identify interior nodes

Public member variables

* std::vector<std::vector<double>> nodes;

contains the coordinates of each node

* std::vector<std::vector<int>> connectivity;

contains the nodes of each element

* int numNodes;

the number of nodes

* int numElements;

the number of elements

* std::vector<int> interior\_nodes;

all interior nodes

#### 2.3 Basis

Define a basis entity either by specifying a Mesh or by copying another Basis

Public member functions

* void print(std::vector<double>&);

prints a vector of doubles

* void print(std::vector<int>&);

prints a vector of ints

* std::vector<double> solve(int);

solves the linear system of FEM

Private member functions

* void calcStiff();

computes the global stiffness matrix

* void calcRhs();

computes the global right hand side

* void calcLocalStiff(std::vector<double>, std::vector<double>, std::vector<double>);

computes a local stiffness matrix on an element

* void apply\_boundary();

elliminates boundary nodes

* void expand\_sol(std::vector<double>&);

exports the final solution

Public member variables

* Mesh my\_mesh;

specifies the mesh on which the basis is defined

* Matrix stiff;

the stiffness matrix

* std::vector<double> rhs;

the right hand side

* std::vector<double> solution;

the final solution

Private member variables

* Matrix localStiff;

a local stiffness matrix on an element