Finite Element Method Programming
STE6291

Tatiana Kravetc

UiT - The Arctic University of Norway

5. February - 9. February, 2018

Abstract

In this course we will implement the finite element method for numerically solving a partial differential equation in two dimensions.

The approach is that we first rewrite the differential equation, describing a physical problem, in the variational form, and then seek an approximate solution in the space of continuous piecewise linear functions [2, 4].

Although the numerical methods presented are general in nature, our model problem will be the Poisson equation and application will be a deformation of a circular membrane.

1 Introduction

The process of approximating the behaviour of a continuum by "finite elements" which behave in a manner similar to the real, "discrete", elements can be introduced through the medium of particular physical applications or as a general mathematical concept.

In many phases of engineering the solution of stress and strain distributions in elastic continua is required. Special cases of such problems may range from two-dimensional plane stress or strain distributions, axisymmetric solids, plate bending, and shells, to fully three-dimensional solids. In all cases the number of interconnections between any "finite element" isolated by some imaginary boundaries and the neighbouring elements is continuous and therefore infinite. It is difficult to see at first glance how such problems may be discretized. The difficulty can be overcome (and the approximation made) in the following manner [3]:

1. The continuum is separated by imaginary lines or surfaces into a number of "finite elements".

2. The elements are assumed to be interconnected at a discrete number of nodal points situated on their boundaries and occasionally in their interior. The displacements of these nodal points will be the basic unknown parameters of the problem, just as in simple, discrete, structural analysis.

3. A set of functions is chosen to define uniquely the state of displacement within each "finite element" and on its boundaries in terms of its nodal displacements.

4. The displacement functions now define uniquely the state of strain within an element in terms of the nodal displacements. These strains, together with any initial strains and the constitutive properties of the material, define the state of stress throughout the element and, hence, also on its boundaries.

5. A system of "equivalent forces" concentrated at the nodes and equilibrating the boundary stresses and any distributed loads is determined, resulting in a stiffness relationship. The determination of these equivalent forces is done most conveniently and generally using the principle of virtual work which is a particular mathematical relation known as a weak form of the problem.

2 Problem description

Consider a circular domain $\Omega \in \mathbb{R}^2$ with its boundary $\partial\Omega$. Apply the force f to this area. In two dimensions Poisson's equation takes the form: find u such that

$$\begin{cases} \Delta u = f, \text{ in } \Omega, \\ u = 0, \text{ on } \partial\Omega. \end{cases} \tag{1}$$

where $\Delta = \frac{\partial^2}{\partial^2 x} + \frac{\partial^2}{\partial^2 y}$ is the Laplace operator.

Thus, u is the deformation level of the circular membrane in the case of force application f.

To derive a variational formulation of Poisson's equation we multiply (1) by a test function v, which is assumed to vanish on the boundary $\partial\Omega$, and integrate by parts. This yields the following variational formulation of the problem

$$-\int_\Omega \nabla u \cdot \nabla v = \int_\Omega f(s) v \, ds. \tag{2}$$

We shall find an approximation of the solution. The basic idea is to first construct spaces of piecewise functions that are easy to manipulate (e.g. differentiate and integrate), and then to show that one can approximate more complicated functions by these simple polynomials.

First, the underlying domain should be partitioned into simplex, such as triangles. *A triangulation*, or mesh, of Ω is a set of triangles such as the intersection of two triangles is either an edge, a corner, of empty. No triangle corner is allowed to lie on an edge or another triangle. The corners of the triangles are called *the nodes*.

Assume that Ω is partitioned by N nodes. The number of nodes is equal to number of internal points (without boundary points) belong to Ω. To compute the finite element approximation of u let v_{ij} be *a basis function* (also called *a shape function*) such that

$$v_{ij} = \begin{cases} 1, \ i = j \\ 0, \ i \neq j \end{cases}, \ i, j = 1, 2, ..., N. \tag{3}$$

Figure 1 illustrates a typical basis function v_j. From the figure one can see that each basis function is continuous, piecewise linear, and with support only on the set of triangles sharing node with index j.

We now can write the solution u as

$$u = \sum_{j=1}^{N} x_j v_j, \tag{4}$$

with N unknowns x_j, $j = 1, 2, ..., N$, to be determined.

Figure 1: A two-dimensional basis function v_j.

Insert (4) to (2). Use the following notation

$$A_{ij} = \int_\Omega \nabla v_j \cdot \nabla v_i \, ds, \; i, j = 1, 2, ..., N, \tag{5}$$

$$b_i = \int_\Omega f v_i ds, \; i = 1, 2, ..., N, \tag{6}$$

we have

$$b_i = \sum_{j=1}^{N} A_{ij} x_j, \; i = 1, 2, ..., N,$$

which is a linear system for the unknowns x_j. In the matrix form we write this system as

$$b = Ax. \tag{7}$$

The matrix A is called *the stiffness matrix* and the vector b is called *the load vector*. Solving the matrix equation (7) we obtain the unknowns x_j, and thus u.

3 Application structure

3.1 Classes

The way of representing a triangulation is based on the GMlib [5] class called Triangle System Module.

```
#include <gmTriangleSystemModule>
```

An implementation of the problem starts from two classes:

1. `Node`,
2. `FEMObject`, which is derived class of `TriangleFacets`.

The first class represents one node and operations with it. As shown in Figure 1, the node consists of a vertex and an array of triangles.

The second class contains an array of nodes (`ArrayLX<Nodes>`) and FEM algorithm computations:

- triangulation, regular or random, (see Section 4),
- computation of the stiffness matrix and the load vector (see Sections 5, 6),
- solution of the matrix equation (7) and updating the height of the vertices.

3.2 Data storage structures

The following structures are used in the implementation:

- `Point<float, d>`;
- `Vector<float, d>`;
- `DMatrix<float> A` is the dynamic stiffness matrix;
- `DVector<float> b` is the dynamic load vector;
- `SqMatrix<float,n>` is a square $n \times n$ matrix;
- `TSVertex<float>`, `TSEdge<float>`, `TSTriangle<float>` are structures included in `<gmTriangleSystemModule>`.

3.3 Operations

The following operations are used for computations:

- `A.invert()` for solving the matrix equation $Ax = b \rightarrow x = A^{-1}b$;
- `triangle->getArea2D()` returns an area of the triangle;
- `std::swap(T a, T b)` swaps the values `a` and `b`;
- `std::abs(float a)` returns an absolute value of `a`;
- `Vector a * Vector b` and `Vector a` $\wedge$ `Vector b` are the inner and vector product of two vectors, respectively.

4 Mesh generation

In two dimensions there are efficient algorithms for creating a mesh on quite general domains. One of these is the *Delaunay algorithm* (see Section 8.3). Delaunay triangulation is optimal in the sense that the angles of all triangles are maximal.

We will create two types of triangulation: regular and random. Implementation consists of the construction of the set of points with specified coordinates.

4.1 Regular triangulation

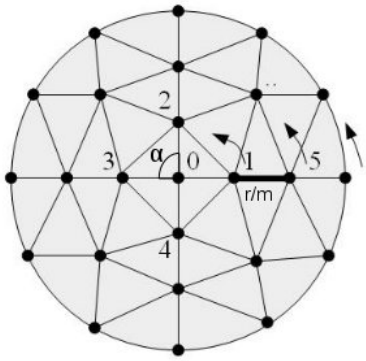


Figure 2: Regular triangulation.

The input values for regular triangulation function are radius of the membrane (r), number of points in the first row (n), number of rows (m). Each row is the ring with width r/m. The process of adding points is iterative. We start from adding zero point with coordinates $(0, 0)$.

Then we need to add the first row. The first point has coordinates $(r/m, 0)$. For computing the rotation angle we use the formula $\alpha = 2i\pi/n$, $i = 0, ..., n-1$. To complete the first row, add n points with coordinates $R\begin{pmatrix} r/m \\ 0 \end{pmatrix}$, where R is the rotation matrix $\begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}$.

Analogically, create the remaining $m-1$ rows by increasing the radius of the circle.

4.2 Random triangulation

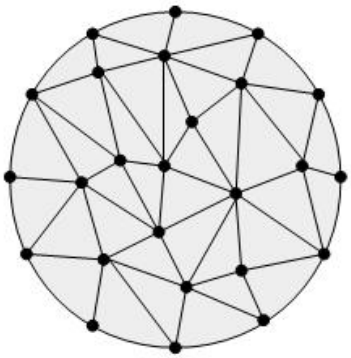


Figure 3: Random triangulation.

The input values for random triangulation function are the radius of the membrane (r) and number of points on the boundary (n).

Start from compact regular triangulation with appropriate high number of points. Swap elements in the array of internal points (exclude the boundary points) t times randomly. Keep the sufficient number of internal points (N) and remove the rest. A good practice is to find the suitable relation between n, r, t and N to get well looking distribution of points.

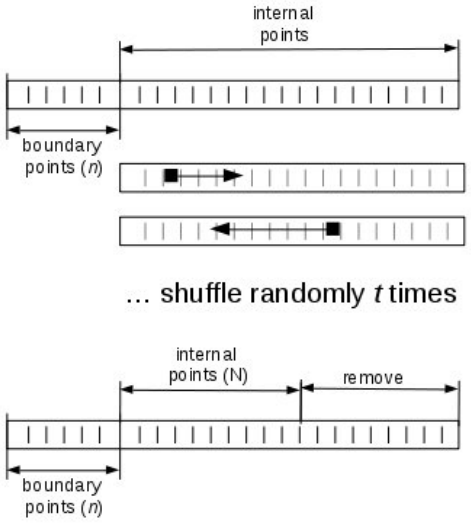


Figure 4: Schematic representation of the random triangulation algorithm.

5 Assembly of the stiffness matrix

5.1 Properties of the stiffness matrix

1. The stiffness matrix A is symmetric and positive definite.
2. $a_{ij} = 0$ if there is no edge ij.
3. $a_{ii} \neq 0$.

The stiffness matrix element is given by

$$a_{ij}^K = \int_K \nabla v_i \cdot \nabla v_j \, ds, \; i, j = 1, ..., N, \tag{8}$$

where K is an area of the intersection of the nodes i and j.

5.2 Non-diagonal elements

An intersection of two nodes Φ_i and Φ_j with common edge ij is represented by an area K consists of two connected triangles T_1 and T_2. Derive an element a_{ij}^K of the stiffness matrix A by using the formula (8)

$$a_{ij}^K = \int_K \nabla v_i \cdot \nabla v_j \, ds =$$

$$= \iint_{T_1} \frac{\partial v_i}{\partial x}\frac{\partial v_j}{\partial x} + \frac{\partial v_i}{\partial y}\frac{\partial v_j}{\partial y} \, dxdy + \iint_{T_2} \frac{\partial v_i}{\partial x}\frac{\partial v_j}{\partial x} + \frac{\partial v_i}{\partial y}\frac{\partial v_j}{\partial y} \, dxdy. \tag{9}$$

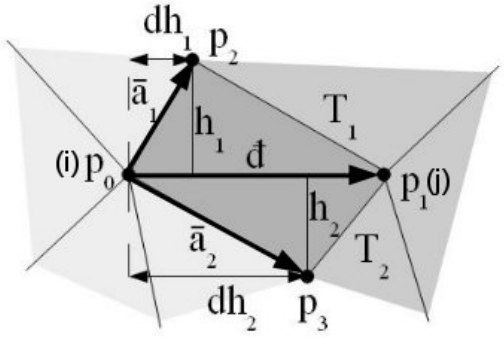


Figure 5: An intersection of two nodes.

Let p_0 and p_1 be vertices of the corresponding nodes Φ_i and Φ_j. Introduce three ordered (see Section 8.2) vectors $\mathbf{a_1}$, $\mathbf{a_2}$ and $\mathbf{d}$, shown in Figure 5, where $\mathbf{d}$ is the edge ij and $\mathbf{a_1}$, $\mathbf{a_2}$ are vectors beginning from the first node vertex.

We now present the transition from integral to vector representation of the stiffness matrix element computation. Rewrite (9) using vector notations

$$a_{ij} = \iint_{T_1} \begin{pmatrix} \frac{\partial v_i}{\partial x} & \frac{\partial v_i}{\partial y} \end{pmatrix} \begin{pmatrix} \frac{\partial v_j}{\partial x} \\ \frac{\partial v_j}{\partial y} \end{pmatrix} dxdy + \iint_{T_2} \begin{pmatrix} \frac{\partial v_i}{\partial x} & \frac{\partial v_i}{\partial y} \end{pmatrix} \begin{pmatrix} \frac{\partial v_j}{\partial x} \\ \frac{\partial v_j}{\partial y} \end{pmatrix} dxdy.$$

Partial derivative represents the instantaneous rate of change of the function in a specified direction. Find the corresponding partial derivatives of the basis functions in the vector representation.

$$\begin{aligned} a_{ij} = \iint_{T_1} \begin{pmatrix} -\frac{1}{|\mathbf{d}|} & \frac{\mathbf{a_1}\cdot\mathbf{d}}{|\mathbf{d}\times\mathbf{a_1}|} \end{pmatrix} \begin{pmatrix} \frac{1}{|\mathbf{d}|} \\ 1 - \frac{\mathbf{a_1}\cdot\mathbf{d}}{|\mathbf{d}\times\mathbf{a_1}|} \end{pmatrix} dT_1 + \\ + \iint_{T_2} \begin{pmatrix} -\frac{1}{|\mathbf{d}|} & \frac{\mathbf{a_2}\cdot\mathbf{d}}{|\mathbf{d}\times\mathbf{a_2}|} \end{pmatrix} \begin{pmatrix} \frac{1}{|\mathbf{d}|} \\ 1 - \frac{\mathbf{a_2}\cdot\mathbf{d}}{|\mathbf{d}\times\mathbf{a_2}|} \end{pmatrix} dT_2 = \\ = \left(-\frac{1}{|\mathbf{d}|^2} + \frac{\mathbf{a_1}\cdot\mathbf{d}}{|\mathbf{d}\times\mathbf{a_1}|} \left(1 - \frac{\mathbf{a_1}\cdot\mathbf{d}}{|\mathbf{d}\times\mathbf{a_1}|} \right) \right) \frac{|\mathbf{a_1}\times\mathbf{d}|}{2} + \\ + \left(-\frac{1}{|\mathbf{d}|^2} + \frac{\mathbf{a_2}\cdot\mathbf{d}}{|\mathbf{d}\times\mathbf{a_2}|} \left(1 - \frac{\mathbf{a_2}\cdot\mathbf{d}}{|\mathbf{d}\times\mathbf{a_2}|} \right) \right) \frac{|\mathbf{a_2}\times\mathbf{d}|}{2}. \end{aligned}$$

Since geometrical sense of the vector product is an area of parallelogram (see Section 8.1), constructed on the corresponding vectors, the area of triangles T_1 and T_2 can be found by the formulas $\frac{|\mathbf{a_1}\times\mathbf{d}|}{2}$ and $\frac{|\mathbf{a_2}\times\mathbf{d}|}{2}$, respectively.

For the implementation we define the following

```
dd = 1/|d|^2;
dh1 = dd * <a1, d>;
dh2 = dd * <a2, d>;
area1 = |d^a1|;
area2 = |d^a2|;
h1 = dd * area1^2;
h2 = dd * area2^2;
```

Then, finally

```
A[i][j] = A[j][i] = (dh1 * (1 - dh1)/h1 - dd) * area1/2 +
                    (dh2 * (1 - dh2)/h2 - dd) * area2/2;
```

5.3 Diagonal elements

These elements are represented by the intersection of a node with itself, as shown in Figure 6. Consider an example of the node Φ_i consists of ℓ triangles T_k, $k = 1, .., \ell$. The formula (8) for this case takes the following form

$$a_{ii}^K = \int_K \nabla v_i \cdot \nabla v_i \, ds =$$

$$= \sum_{k=1}^{\ell} \underbrace{\iint_{T_k} \left(\left(\frac{\partial v_i}{\partial x} \right)^2 + \left(\frac{\partial v_i}{\partial y} \right)^2 \right) dxdy}_{T_k} = \sum_{k=1}^{\ell} T_k. \tag{10}$$

According to Section 5.2, we will find the gradient of the basis function in the vector form. Let p_0 be a vertex of the node Φ_i, p_1 and p_2 be two remaining vertices of the triangle T_k. Consider three ordered vectors (see Section 8.2) $\mathbf{d_1}$, $\mathbf{d_2}$, $\mathbf{d_3}$. An integral T_k from the equation (10) can be evaluated by the formula:

$$T_k = \frac{\mathbf{d_3} \cdot \mathbf{d_3}}{2|\mathbf{d_1} \times \mathbf{d_2}|}.$$

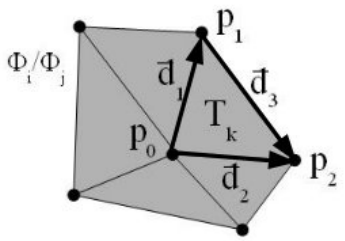


Figure 6: An intersection of the node with itself.

6 Assembly of the load vector

The load vector b is assembled by summing element load vectors b^K over the mesh. The area K consists of ℓ triangles T_k, $k = 1, ..., \ell$.

$$b_i = \int_K f v_i ds = f V_{pyr} = f \frac{1}{3} S_{base} h = f \frac{1}{3} \sum_{k=1}^{\ell} S_{tr}.$$

The function f is the specified load function. An integral of the basis function is equal to the volume of pyramid based on the node with hight is equal to 1 (see Figure 1). An area of the base S_{base} is equal to sum of triangle areas S_{tr}^k, $k = 1, ..., \ell$.

7 Basic Finite Element algorithm

The following algorithm summarizes the basic steps for computing the finite element solution.

1. Create a triangulation of Ω and define the corresponding basis functions.

2. Assemble the $[N \times N]$ stiffness matrix A and $[N \times 1]$ load vector b, with entries

$$A_{ij} = \int_{\Omega} \nabla v_j \cdot \nabla v_i \, ds, \; i, j = 1, 2, ..., N,$$

$$b_i = \int_{\Omega} f v_i ds, \; i = 1, 2, ..., N.$$

3. Solve the matrix equation

$$Ax = b.$$

4. Set the solution to internal vertices.

8 Additions

In this section we consider some reminders and notes from [1] used in the implementation.

8.1 Vector operations

Point/vector operations

- point − point = vector,
- point + point = *undefined*,
- point + vector = point,
- const · vector = vector,
- vector + vector = vector,
- vector − vector = vector.

Vector products

- An inner product between two vectors $\mathbf{a}$ and $\mathbf{b} \in \mathbb{R}^3$ is denoted as

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3 \in \mathbb{R}.$$

- A vector product in $\mathbb{R}^3$, also called wedge product, is denoted as

$$\mathbf{a} \times \mathbf{b} = \left(\begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix}, \begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix}, \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} \right).$$

Properties of the vector products:

- $||\mathbf{a} \times \mathbf{b}||$ is the positive area of a parallelogram having $\mathbf{a}$ and $\mathbf{b}$ as sides,
- $\mathbf{a} \times \mathbf{b}$ is orthogonal to both of its vectors,
- $||\mathbf{x}|| = \sqrt{\mathbf{x} \cdot \mathbf{x}}$.

8.2 Arrangement of vectors

An arrangement of vectors (edges) in the triangles from nodes intersection is important for the stiffness matrix element computation. There are two types of nodes intersection: when two nodes have common edge and intersection of the node with itself. Let Φ_i, Φ_j be considered nodes. In the first case they have a common edge ij, in the second case they are equal $\Phi_i = \Phi_j$.

1. In the first case there are two triangles in the intersection as shown in Figure 5.

   Implement a function, which input is a common edge between considered nodes and output is an array of coordinates of the vectors $\overrightarrow{p_0p_1}$, $\overrightarrow{p_0p_2}$, $\overrightarrow{p_0p_3}$.

   ```
   Vector<Vector<float, 2>, 3> findVectors(TSEdge* edge){ ... }
   ```

2. In the second case there are ℓ triangles, belonging to the considered node. An input of the implemented function is one triangle T_k of the node (see Figure 6). An output is an array of coordinates of the vectors $\overrightarrow{p_0p_1}$, $\overrightarrow{p_0p_2}$, $\overrightarrow{p_1p_2}$. Note that p_0 is the vertex of the node.

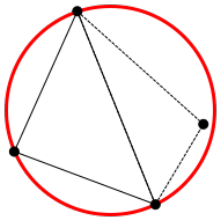
   ```
   Vector<Vector<float, 2>, 3> findVectors(TSTriangle* tr,
                                           Node* node){ ... }
   ```

8.3 Delaunay triangulation

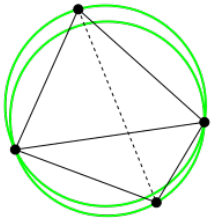
There is an implemented function in the class `<gmTriangleSystemModule>`, which provides Delaunay triangulation.

```
this -> triangulateDelaunay();
```

Delaunay triangulation is a triangulation such that no point in the set is inside the circumcircle of any triangle in the triangulation.



This pair of triangles does not meet the Delaunay condition (the circumcircle contains more than three points).



Flipping the common edge produces a valid Delaunay triangulation for the four points.

Figure 7: Examples of the Delaunay triangulation.

The main property is that if $\gamma + \alpha \leq 180°$, (see Figure 8), the triangles meet the Delaunay condition.

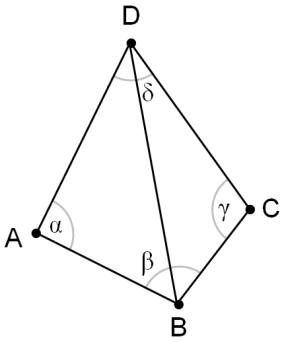


Figure 8: This triangulation does not meet the Delaunay condition (the sum of α and γ is bigger than 180°).

References

[1] A. Lakså, *Blending technics for curve and surface constructions*, Narvik University College, Narvik, Norway, 2012.

[2] Mats G. Larson, Fredrik Bengzon, *The Finite Element Method: Theory, Implementation, and Practice*, Springer, 2010.

[3] O.C. Zienkiewicz, R.L. Taylor, J.Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals*, Sixth edition, 2005.

[4] A. Ern, J.-L. Guermond, *Theory and Practice of Finite Elements*, Springer, 2004.

[5] *Geometric Modeling Library*, `https://source.uit.no/groups/gmlib`.