Assignment 4

CASPER BRESDAHL, whs715, University of Copenhagen, Denmark

1 INTRODUCTION

In this week we will first look into the theory of Finite Element Method and look at a 1D example derivation. We will then look at our first experiment where we will try to vary the mesh resolution before we end by our second experiment which will look at the effect of changing the boundaries.

2 THEORY

2.1 Finite Element Method

The Finite Element Method (FEM) is a method in five steps to numerically solve differential equations. FEM works by subdividing a large system into more smaller and simpler systems which are called *finite elements* which then can be discretized. FEM approximates the unknown function by taking the smaller systems that model the finite elements and assembles them into a larger system that models the entire problem and then solves the resulting system of equations. The five steps of FEM can be described as:

(1) **Write the volume integral.** In this initial step we are given some partial differential equation which evaluates to some known function, and we are given some boundary conditions. We then rewrite the partial differential equation such that the equation equals zero, then multiply by a trial function and then take the integral over the domain.
(2) **Perform integration by parts.** We then perform integration by parts which means we reduce our problem from *strong form* to *weak form*, meaning our original partial derivative had to be twice differentiable, but now only needs to be differentiable. However, the trial function we introduced now also needs to be differentiable.
(3) **Make an approximation.** We can now find an approximation to our continuous unknown function by breaking it into a discrete set of n values that we can combine by some weight scheme given by a shape function. We will go into details about shape functions in the next section. We can now substitute our unknown function by this approximation.
(4) **Choose a trial function.** We then define our trial function in a way which makes us realize we are solving a linear system.
(5) **Compute a solution.** Lastly we handle our boundary conditions and we solve our linear system to find an approximation to the unknown function.

2.2 Shape functions

The shape functions are the functions which interpolates the solution between discrete nodes in our mesh. When we make an approximation of the unknown function we use shape functions. These can be seen both globally and locally. We are interested in two properties of these shape functions, namely that they interpolate the points, and that they 'preserve mass', more formally called *partition of unity*. In 1D we would have the interpolation property of the global shape functions telling us:

$$N_i(x) = \begin{cases} 1 & x = x_i \\ 0 & x \neq x_i \end{cases}$$

Which means the geometric shape of the shape functions would be a lot triangles next to each other, spiking up to 1. A concern here is the shape functions might not be differentiable at all points. Because of this we define local shape functions. We define the local shape functions between two nodes x_i and x_j as:

$$N_i^e = \frac{x_j - x}{\Delta x}, \quad N_j^e = \frac{x - x_i}{\Delta x}$$

The partition of unity tells us:

$$\sum_i N_i(x) = 1$$

Which means that at all points the 'mass' is always 1. In 2D we would use triangles to constitute our elements, and thus we would use *barycentric coordinates* as shape functions. Locally this would be defined as:

$$N_i^e = \frac{A_i^e}{A^e}, \quad N_j^e = \frac{A_j^e}{A^e}, \quad N_k^e = \frac{A_k^e}{A^e}$$

Where the numerators are the area of the triangle made by having a point inside the original triangle and substituting the subscript vertex by that point and A^e is the area of the original triangle.

2.3 Example derivation

We will no apply the five steps of FEM to a 1D problem. We begin with the partial differential equation $\frac{\partial^2 y(x)}{\partial x^2} = c(x)$ where we would like to approximate $y(x)$ and are given $c(x)$. We are also given the boundary conditions $y(x_1) = a$ and $y(x_n) = b$. In **step 1** we rewrite our partial differential equation into a volume integral. We do this by multiplying by a *trial function* and integrating over our domain:

$$\int_{x_1}^{x_n} v(x)\left(\frac{\partial^2 y(x)}{\partial x^2} - c(x)\right)dx = \int_{x_1}^{x_n} v(x)\frac{\partial^2 y(x)}{\partial x^2}dx - \int_{x_1}^{x_n} v(x)c(x)dx = 0$$

Our trial function is an arbitrary function defined such that $v(x_1) = v(x_n) = 0$. In **step 2** we apply integration by parts which gives us that our first term can be rewritten:

$$\int_{x_1}^{x_n} v(x)\frac{\partial^2 y(x)}{\partial x^2}dx = \left[v(x)\frac{\partial y(x)}{\partial x}\right]_{x_1}^{x_n} - \int_{x_1}^{x_n} \frac{\partial v(x)}{\partial x}\frac{\partial y(x)}{\partial x}dx$$

Given the definition of our trial function we have the first part of our new expression is simply zero and after multiplying by -1 we now have the equation:

$$\int_{x_1}^{x_n} \frac{\partial v(x)}{\partial x}\frac{\partial y(x)}{\partial x}dx + \int_{x_1}^{x_n} v(x)c(x)dx = 0$$

In **step 3** we now find an approximation of our continuous unknown y function. We do this by breaking it into a discrete set of n points

Author's address: Casper Bresdahl, whs715, University of Copenhagen, Copenhagen, Denmark, whs715@alumni.ku.dk.

$(\hat{y}_i, x_i)$. We then have:

$$y(x) \approx \tilde{y}(x) = \sum_{i=1}^{n} N_i(x)\hat{y}_i = \begin{bmatrix} N_1(x), \dots, N_n(x) \end{bmatrix} \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \mathbf{N}\hat{y}$$

We can now replace $y(x)$ with our approximation which gives us:

$$\int_{x_1}^{x_n} \frac{\partial v(x)}{\partial x} \frac{\partial \mathbf{N}(x)}{\partial x} \hat{y} dx + \int_{x_1}^{x_n} v(x)c(x)dx = 0$$

In **step 4** we define our trial function further, defining it as $v(x) = \mathbf{N}\delta\mathbf{y}$ where $\delta\mathbf{y}$ is a vector of random values for all discrete points. Defining our trial function using our shape functions is known as the *Galerkin method*. We can now replace our trial functions with this new definition. Realizing neither $\delta\mathbf{y}$ or $\hat{y}$ depends on x we can move them outside our integrals. Further realizing we now are multiplying random values on our integrals, and they still need to evaluate to zero, we can completely disregard $\delta\mathbf{y}$ and we now have:

$$\left(\int_{x_1}^{x_n} \frac{\partial \mathbf{N}^T(x)}{\partial x} \frac{\partial \mathbf{N}(x)}{\partial x} dx \right) \hat{y} + \int_{x_1}^{x_n} \mathbf{N}^T(x)c(x)dx = 0$$

We can now define our first integral as $\mathbf{K}$ and our second integral as $\mathbf{-f}$ and see we now have the linear system $\mathbf{K}\hat{y} = \mathbf{f}$. In **step 5** we can now insert our boundary conditions in $\mathbf{K}$ and $\mathbf{f}$ which in our case means we define $\mathbf{K}_{11} = 1$ and the rest of the elements in row 1 as 0 and $\mathbf{K}_{nn} = 1$ and the rest of the elements of row n as 0. We then also define $\mathbf{f}_1 = a$ and $\mathbf{f}_n = b$. This way we have now applied our original boundary conditions $y(x_1) = a$ and $y(x_n) = b$. We would now be able to solve our system for our approximation of the y function. This is an approximation in a global setting. One concern would be if our shape functions are non differentiable at certain points. To alleviate this concern, we can derive this global setting from examining the local setting:

$$\int_{x_1}^{x_n} \frac{\partial \mathbf{N}^T(x)}{\partial x} \frac{\partial \mathbf{N}(x)}{\partial x} \hat{y} dx + \int_{x_1}^{x_n} \mathbf{N}^T(x)c(x)dx =$$

$$\sum_e \left(\int_{x_i}^{x_j} \frac{\partial (N^e)^T(x)}{\partial x} \frac{\partial \mathbf{N}^e(x)}{\partial x} \hat{y}^e dx + \int_{x_i}^{x_j} (N^e)^T(x)c(x)dx \right) = 0$$

Where e is the element between two consecutive nodes x_i and x_j. Here $\mathbf{N}^e(x)$ is now composed of the local shape functions and $\hat{y}^e$ is composed of the discretized y-value at x_i and x_j. And this shows us that we can assemble the local elements into a global system as:

$$\mathbf{K}\hat{y} = \sum_e \mathbf{K}^e \hat{y}^e = \sum_e \mathbf{f}^e = \mathbf{f}$$

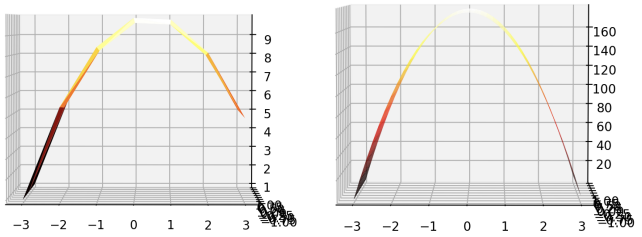
We can now see how we started with one large system which we then subdivide into smaller systems, one for each element in the system, and used the local shape functions to approximate the integrals to avoid non differentiable points, and then we assemble these smaller systems back to one large system that models the entire problem.

3 EXPERIMENTS

In the following experiments we will look at a 2D problem. We will solve the partial differential equation $\frac{\partial^2 u(p))}{\partial x^2} + \frac{\partial^2 u(p)}{\partial y^2} = c(p)$ where p is a point in our domain and $c(p)$ is a known function. From this definition we can see $c(p)$ expresses the curvature of the u function which we want to approximate. This means when if we define $c(p) = 0$ we will get a linear plane as a solution, and otherwise our solution will have a curve to it.

3.1 Experiment 1 - Varying the mesh resolution

In this experiment we would like to investigate what happens when we vary the resolution of the mesh. The effect of varying the resolution is we get more nodes in our domain and thus we would also expect a better approximation to the unknown function. We can begin by examining the effect of increased mesh resolution for a solution where our curvature condition is $c(p) = 1$. We define our domain as a 6 by 2 square centered around (0,0), and we will then create a low resolution mesh consisting of 6 by 2 nodes, and a high resolution mesh consisting of 36 by 12 nodes. Our boundary conditions will be 1 and 5. The resulting low and high resolution meshes can be seen in Figure 1.

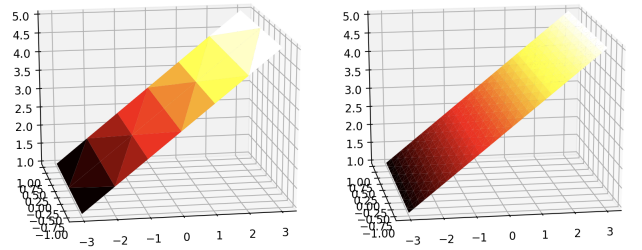


(a) Low resolution mesh with curvature condition $c(p) = 1$.

(b) High resolution mesh with curvature condition $c(p) = 1$.

Fig. 1. Low and high resolution meshes with curvature condition $c(p) = 1$.

We here note the much more jagged shaped of Figure 1a compared to the very smooth shape of Figure 1b. We also note that both meshes are defined over the same domain, but the arc of Figure 1b goes much higher up than Figure 1a. We can now take a look at the same mesh resolutions but with $c(p) = 0$. The results are shown in Figure 2.



(a) Low resolution mesh with curvature condition $c(p) = 0$.

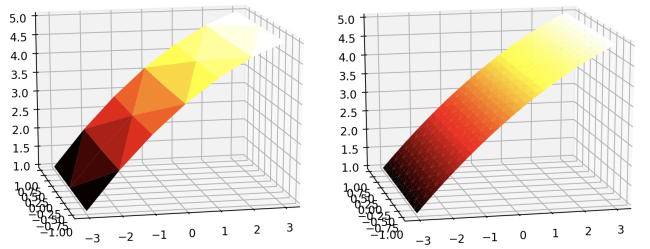
(b) High resolution mesh with curvature condition $c(p) = 0$.

Fig. 2. Low and high resolution meshes with curvature condition $c(p) = 0$.

Rather unsurprisingly we here see two solutions which looks very much the same. They both go completely linearly from the first to the second boundary condition. However, we do note we

have a lot more nodes in the high resolution mesh, which means this is a much better approximation to the unknown function.

3.1.1 Discussion of results. From Figure 2a and Figure 2b we see that our results matches our expectations as increasing the resolution simply gives us the same solution but with more nodes in the mesh and thus a better approximation. However, the same is not true for Figure 1a and Figure 1b. We here see that when we have a fixed curvature for all elements, the curvature condition will apply between all nodes and thus get applied a lot more on high resolution meshes. This is why the arc in Figure 1b goes a lot higher than the arc in Figure 1a. As we now understand that the number of nodes in the mesh influences the curvature of the mesh we could try to divide the wanted curvature by the number of nodes in the mesh. The results can be seen in Figure 3.



(a) Low resolution mesh with curvature condition $c(p) = 1/(6 \cdot 2)$.

(b) High resolution mesh with curvature condition $c(p) = 1/(36 \cdot 12)$.

Fig. 3. Results of normalizing the fixed curvature by the number of nodes in the mesh.

We now see very similar results as to Figure 2a and Figure 2b where both meshes looks very much the same, and both meshes goes from the lower boundary to the upper boundary, but Figure 3b has a lot more nodes than Figure 3a. We thus conclude that increasing the mesh resolution greatly increases approximation accuracy, but if we have a curvature condition we will need to normalize it by the number of nodes in our mesh as our curvature condition is applied to all elements in the mesh.

3.2 Experiment 2 - Varying boundary conditions

In this experiment we will examine the effect of having different boundary conditions along the boundary. We know when we define $c(p) = 0$ our solution will be a linear plane between going from the lower boundary condition to the upper boundary condition. We will now try to give 'nonsense' boundary conditions in the sense that our boundaries will now be tilted instead of uniform. The result can be seen in Figure 4. We here see the lower boundary values going linearly from 0 to 2 whereas the upper boundaries going linearly from 4 to 6. The resulting approximation looks a lot like the result when using uniform boundary conditions. In the middle the solution seems to be the same, and at the boundaries it seems the solutions has tried to 'adapt' to the 'rotated' boundary. This is in contrast to what we might have expected, namely the entire surface had rotated to match the new boundaries. We can now attempt to use some more random values at the boundaries to create a jagged boundary.

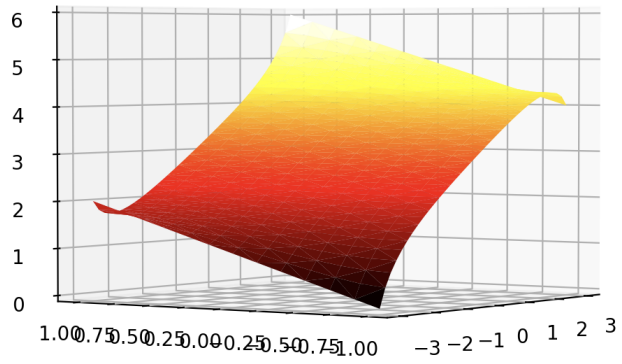


Fig. 4. Linear 'rotated' boundary conditions.

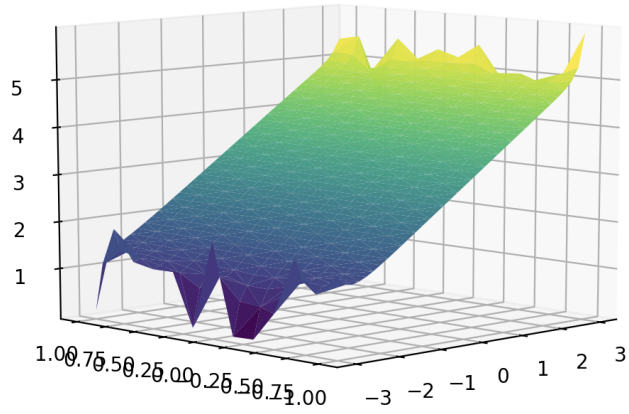


Fig. 5. Jagged boundary conditions.

The result can be seen in Figure 5. We here see almost the entire surface approximates the 'regular' solution with uniform boundary conditions and only barely changes due to the jagged boundaries.

3.2.1 Discussion of results. We end this experiment by concluding the approximations for the most part does not change as we change the boundaries, and rather than redefining the solution the boundary conditions gets incorporated into the solution.

4 CONCLUSION

We end by concluding FEM is a five step approximation method which becomes very similar in 1D and 2D problems. We have also looked at the effect of increasing the mesh resolution where we conclude we find better approximations to the unknown function as the mesh resolution increases, but we need to take into consideration that curvature conditions are applied at every element. We also conclude the boundary conditions do not define the solution of our unknown, rather the borders are attempted incorporated into the solution.