CMIS Hand-in 4: Finite Element Method

Nikolai Plambech Nielsen
lpk331@alumni.ku.dk
Niels Bohr Institute, University of Copenhagen

1 FINITE ELEMENT METHOD

The essence of the finite element method is to represent the domain as a finite set of elements (hence the name), and compute an approximate solution for each of these elements, adding up to the global solution. The actual elements vary on the dimensionality of the system. Straight-line segments for 1D problems, triangles for 2D and tetrahedra for 3D are the ones we focus on here due to their simplicity, but others can be used as well.

The actual method can be summarised in a series of steps:

(1) Convert the differential equation to a volume integral
(2) Define the elements
(3) Choose a shape and trial function for the elements
(4) Compute the elementwise integrals
(5) Assemble the global matrix and source vector
(6) Apply boundary conditions
(7) Compute the solution

In this hand-in we focus on implementing the method in 1D and 2D, and use it to solve the Poisson equation for a simple source term:

$$\nabla^2 u = c \tag{1}$$

where to begin with we set $c = 0$, and later $c \in \mathbb{R}$.

In the first step of the recipe we multiply each side by a function $v(\mathbf{r})$, which must be continuously differentiable and 0 on the boundary of the domain ($v(\mathbf{r}) = 0,\ \forall\ \mathbf{r} \in \Gamma$). Then we integrate over the domain on both sides to get

$$\int_\Omega v(\mathbf{r})\nabla^2 u(\mathbf{r})\ \mathrm{d}\Omega = \int_\Omega v(\mathbf{r})c\ \mathrm{d}x \tag{2}$$

This formulation of the problem is still equivalent to the original form, and is called "Strong Form". Next we perform integration by parts on the left hand side:

$$\int_\Omega v(\mathbf{r})\nabla^2 u(\mathbf{r})\ \mathrm{d}\Omega = \int_\Gamma v(\mathbf{r})\nabla u(\mathbf{r})\ \mathrm{d}\Gamma - \int_\Omega \nabla v(\mathbf{r})\nabla u(\mathbf{r})\ \mathrm{d}\Omega \tag{3}$$

But the first term is zero by definition, leading to the "Weak Form" of the problem:

$$\int_\Omega \nabla v(\mathbf{r})\nabla u(\mathbf{r})\ \mathrm{d}\Omega = -\int_\Omega v(\mathbf{r})c\ \mathrm{d}x \tag{4}$$

This takes care of the first step. In the second step we use the methods from last weeks hand-in on the generation of computational meshes. In particular I define the geometry of the setup in a `.poly` file and relegate the mesh generation to the `Triangle` program written by J. R. Shewchuk.

For the third step we approximate the unknown u by the function

$$u \approx \sum_{e\in\Omega} \mathrm{N}^e \hat{u}^e, \tag{5}$$

where the sum runs over all elements in the system, $\hat{u}^e$ is a vector of the value of u on the 3 vertices in the e'th element (with vertices counted counter clockwise): $\hat{u}^e = [u_i^e\ u_j^e\ u_k^e]^T$, and N^e is the "shape function" for the element. The shape function must have the following properties

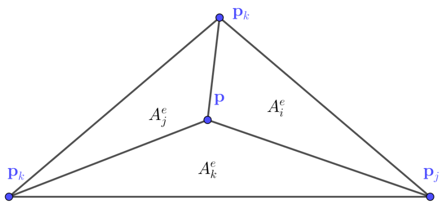

Figure 1: Barycentric coordinates for a triangular element.

$$N_i(\mathbf{r}) = \begin{cases} 1 & \mathbf{r} = \mathbf{r}_i, \\ 0, & \mathbf{r}_j \wedge j \neq i \end{cases}, \quad \sum_i N_i(\mathbf{r}) = 1. \tag{6}$$

The first ensures proper interpolation for the points, whilst the second ensures that the approximation of $\mathbf{u}$ is exact on all vertices in the system. In one dimension we use a hat-function:

$$N_i(x) = \max\left(0, 1 - \left|\frac{x_i - x}{\Delta x}\right|\right) \tag{7}$$

which are 1 on the i'th vertex, and 0 on all other (assuming equal spacing of nodes in the domain Δx). And for triangles in two dimensions we use barycentric coordinates:

$$N_i^e(\mathbf{r}) = \frac{A_i^e}{A^e} \tag{8}$$

where the geometry is as seen in figure 1. Where the areas can be found as the magnitudes of cross products:

$$N_i^e = \frac{|(\mathbf{p}_k - \mathbf{p}_j) \times (\mathbf{p} - \mathbf{p}_j)|}{2A^e}, \quad A^e = \frac{1}{2}\left|(\mathbf{p}_j - \mathbf{p}_i) \times (\mathbf{p}_k - \mathbf{p}_i)\right| \tag{9}$$

Now, the points of the triangles are two dimensional, so the cross products are also just the determinant of the 2×2 matrix $[(\mathbf{p}_k - \mathbf{p}_j)\ (\mathbf{p} - \mathbf{p}_j)]$