Department of Mechanical Engineering
University of Bath

ME40064: Systems Modelling & Simulation
ME50344: Engineering Systems Simulation

Tutorial 2: Deriving the Transport Equation & Mesh Data Structure

Part A: Finite Element Mesh Data Structure

In this part of the tutorial you will familiarise yourselves with the data structure that is used to represent a finite element mesh.

1. To begin, download the following two files from the course Moodle page: `OneDimLinearMeshGen.m` and `displayMesh.m`.

The file `OneDimLinearMeshGen.m` defines a function that creates a linear, uniformly spaced mesh. To use it, enter the following into the Matlab command window:

```
mesh = OneDimLinearMeshGen(0,1,10);
```

Display a graphical representation of this mesh with the function:

```
displayMesh(mesh)
```

Now access the components of the mesh data structure using the dot notation introduced in Lecture 2 & Tutorial 1. For example, entering

```
mesh.ne
```

into the command line accesses the member `ne` of the variable mesh, which in this case stores the number of elements.

Similarly, `mesh.ngn` stores the number of global nodes in the mesh. Read the comments in `OneDimLinearMeshGen.m` to see what the other members of the mesh data structure contain.

2. Now write a Matlab script that uses a `for` loop to quickly generate an array of meshes for the domain x = [0,1] with the following numbers of elements: `1, 2, 4, 8, 16`.

   For each mesh, use `displayMesh` to generate a figure of the mesh and then save this figure to a jpeg file.

Some useful hints

- The notation x =[0,1] uses interval notation to indicate a domain between the coordinates x = 0 and x = 1, inclusive of 0 and 1. More information on interval notation is on Wikipedia here:

https://en.wikipedia.org/wiki/Interval_(mathematics)#Notations_for_intervals

- You could store the different number of elements in a vector:

  ```
  NumElems = [1 2 4 8 16];
  ```

  Then use your loop counter to access the appropriate value in this array, which can then be used as an input to `OneDimLinearMeshGen.`

- Use `saveas(gcf,...,...)` to save the figure to a file. `gcf` means get current figure, which tells the function which figure it is supposed to save to file.

- `strcat` is a function which concatenates, or joins together, different strings (or words). This allows you to combine different strings to automatically construct your filename. If used with `num2str,` below, this will allow you to assign a different filename for each figure.

- `num2str(var)` will convert the numerical value stored in the variable `var,` (i.e. your loop counter) into a string. This string can then be used inside the `strcat` function to construct part of the filename for your figure. This will allow you to increment the filename with each iteration of the loop.

Department of Mechanical Engineering
University of Bath

Part B: Linear Nodal Lagrange Basis Functions

1. The domain x=[0,1] is meshed with three elements using linear nodal Lagrange basis functions, defined by the following nodal positions:

$$x_0 = 0.0, \quad x_1 = 0.2, \quad x_2 = 0.4, \quad x_3 = 1.0$$

Calculate the value of *x* in *element 1* for:

$$\xi = -0.5$$

Calculate the value of *x* in *element 2* for:

$$\xi = 0.0$$

Calculate the value of *x* in *element 3* for:

$$\xi = 0.2$$

Remember that for a general element, *e*, the interpolations are defined:

$$x(\xi) = x_{e-1}\psi_0(\xi) + x_e\psi_1(\xi)$$

that is:

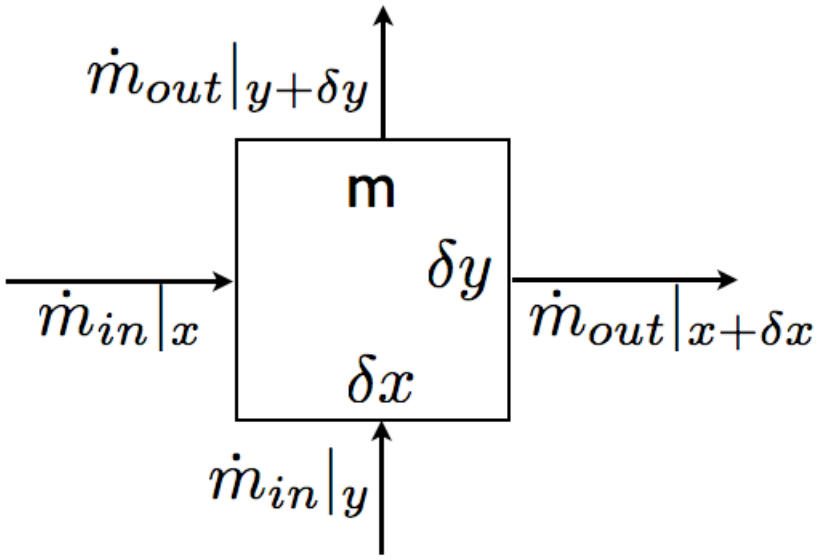
$$x(\xi) = x_{e-1}\left(\frac{1-\xi}{2}\right) + x_e\left(\frac{1+\xi}{2}\right)$$

2. Show that for a mesh where space is interpolated using the linear nodal Lagrange basis functions, the following relationship is true:

$$x^{elem2}(\xi = 1) = x^{elem3}(\xi = -1)$$

Part C: Deriving Transport Equations

1. a. In the lecture a derivation for the general continuity equation was presented, for a 2D domain, and a flux that was aligned in the x-axis. Following the same procedure, derive the same equation for a 2D domain with fluxes along both the x and y axes, as shown below:



In this example, you might begin by writing the flux as:

$$\mathbf{J} = J_x\mathbf{i} + J_y\mathbf{j}$$

where **i** and **j** are unit vectors in the x and y directions respectively.

b. Then substitute Fick's first law of diffusion to derive the transient diffusion equation in 2D.

Perform the derivation keeping the differential operators explicitly stated and then check it is equivalent to using the nabla operator (in 2D). See the formulae below for more on this.

Optional Question

2. For the 3D version of the equation derived in question 1, how would this differ if the diffusion coefficient was no longer a scalar, but had different values in each axis, and could be specified in the following tensor form?

$$\mathbf{D} = \begin{bmatrix} D_{xx} & 0 & 0 \\ 0 & D_{yy} & 0 \\ 0 & 0 & D_{zz} \end{bmatrix}$$

Some useful formulae

The following are all given for a 3D Cartesian coordinate system, but removing the z dependency converts them to the 2D Cartesian system.

Nabla operator in 3D Cartesian coordinates is:

$$\nabla = \frac{\partial}{\partial x}\mathbf{i} + \frac{\partial}{\partial y}\mathbf{j} + \frac{\partial}{\partial z}\mathbf{k}$$

Divergence of vector **u** is:

$$\nabla \cdot \mathbf{u} = \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}$$

where **u** is given by:

$$\mathbf{u} = u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k}$$

Gradient of a scalar c (or grad c) is:

$$\nabla c = \frac{\partial c}{\partial x}\mathbf{i} + \frac{\partial c}{\partial y}\mathbf{j} + \frac{\partial c}{\partial z}\mathbf{k}$$

Laplacian (or grad squared) of scalar c is:

$$\nabla^2 c = \frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} + \frac{\partial^2 c}{\partial z^2}$$