MECH3750 PBL Content Summary

Week 11

Content:

- Elliptic PDEs
  - Numerical Solution
  - Grid Generation

Upcoming assessment:

- Problem Sheet 11 (due before Week 12 PBL session)
- Assignment II (Due tomorrow)

Tutors: Nathan Di Vaira, Alex Muirhead, William Snell, Tristan Samson, Nicholas Maurer, Jakob Ivanhoe, Robert Watt

1 Elliptic PDEs

The final type of PDE we look at in this course are elliptic PDEs. The most simple example of an elliptic PDE is Laplace's equation,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

$$\nabla^2 u = 0$$

$$\Delta u = 0$$

The case when there is a source term (i.e., the PDE is non-homogenous) is known as Poisson's equation.

Elliptic PDEs have no time derivative, and hence typically solve steady state systems (temperature distributions and potential fields). The solution is therefore influenced only by boundary conditions, not initial conditions. As with other PDEs, types of boundary value problems may include Dirichlet, Neumann or mixed boundaries.

1.1 Numerical Solution

To solve elliptic PDEs numerically, we take a centred finite difference approximation for each spatial derivative,

$$\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{(\Delta x)^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{(\Delta y)^2} = 0.$$

which now deals with two discrete dimensions, i and j. Note that, while the solution nodes are in two-dimensions, the solution vector must be assembled as a single-ordinate vector. For a soluton with $m \times n$ spatial nodes, we introduce a single-ordinate counter,

$$p = i + jm.$$

The solution matrix would therefore have dimensions $(m \times n, m \times n)$ – a single row of the solution matrix is assembled by moving along each row of spatial nodes. The example python script attached to the *W11L03* lecture performs this matrix construction and solution. Due to the large size of the solution matrix, iterative methods, such as Gauss-Seidel, may be required to solve...

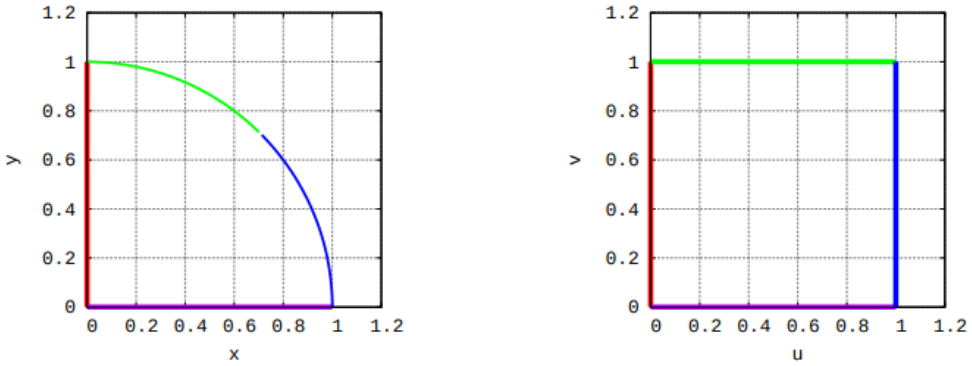
An alternative solution method is to rearrange to give the general update equation for inner nodes,

$$u_{i,j} = \frac{(\Delta y)^2(u_{i-1,j} + u_{i+1,j}) + (\Delta x)^2(u_{i,j-1} + u_{i,j+1})}{2((\Delta x)^2 + (\Delta y)^2)},$$

start will an initial guess for all nodes i, j, and iteratively step through each node i, j to find an updated value until the solution change between steps in adequately small.

1.2 Application to Grid Generation

It turns out that elliptic PDEs are particularly useful for handling complex computational grids. The general idea is that we treat the complex *physical* grid, such as a curve, as a *computational* square grid.



The edge points in the x-y domain become the boundary conditions in the u-v domain, and we then solve for x and y using two different eliptic equations,

$$\frac{\partial^2 x}{\partial u^2} + \frac{\partial^2 x}{\partial v^2} = 0$$

$$\frac{\partial^2 y}{\partial u^2} + \frac{\partial^2 y}{\partial v^2} = 0$$

where x and y are the variables being solved for in each.

The example attached to the *W11L03* lecture gives an excellent example of how this is implemented.

1.3 Other Considerations

When handling Neumann boundaries, a simple approach is to use a one-sided finite difference approximation,

$$\frac{\partial u}{\partial x} \approx \frac{u_{N,j} - u_{N-1,j}}{\Delta x}.$$

Additionally, using the central difference approximations shown above for interior nodes give a second order global spatial order of accuracy. Note that boundary approximations of one order less than the interior approximations generally do not reduce the global second order accuracy.