1 Elliptic PDEs

This week we saw the start up of a new lecturer and a focus on elliptical PDEs,

$$\partial_{xx} f + \partial_{yy} f = 0,$$
$$\nabla^2 f = 0,$$
$$\Delta f = 0.$$

Here we see that the PDE has **no** temporal derivative and thus governs *steady-state* processes where the solution is influenced by the B.C. not the I.C.. As such problems are often defined by their type of boundary,

1. First boundary value problem (BVP) has Dirichlet boundaries, i.e. f is fixed on the bounding surface;
2. Second BVP has Neumann boundaries where $\partial_n f$ is defined on the bounding surface;
3. Third BVP has Robin or mixed boundaries where f may be prescribed on a portion of the boundary and $\partial_n f$ is prescribed on the remainder.

1.1 Polar coordinates

Often it can be useful to transform from a Cartesian type discretisation and use polar coordiantes to resolve a system. This however has to be taken into account in our governing equation,

$$x = r cos\theta \qquad y = r sin\theta$$
$$\implies \partial_{rr} + \partial_r f / r + \partial_{\theta\theta} f / r^2 = 0$$

The fundamental solutions in polar coordinates:

- source/sink: $f(r) = \frac{Qln(r)}{2\pi}$
- vortex: $f(\theta) = \frac{\Gamma\theta}{2\pi}$

1.2 Poisson equation

The Laplace equation is a homogeneous PDE, if we have a source term it is called Poisson's equation,

$$\partial_{xx} f + \partial_{yy} f = S(x, y).$$

1.3 Numerical solution for the Laplace equation

If we consider this for a steady state heat problem, we can discretise the Laplace equation using centred finite differences,

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

This gives us a two dimensional stencil which isn't typically convenient to work with so we often need to construct a single-ordinate representation of this. In doing so, we can assemble nodal equations in the form $Ax = b$, where A is our coefficient matrix, x is our unknown temperatures and b is full of known values.

To move from the (i, j) mesh, one technique in Python is to use lambda functions,

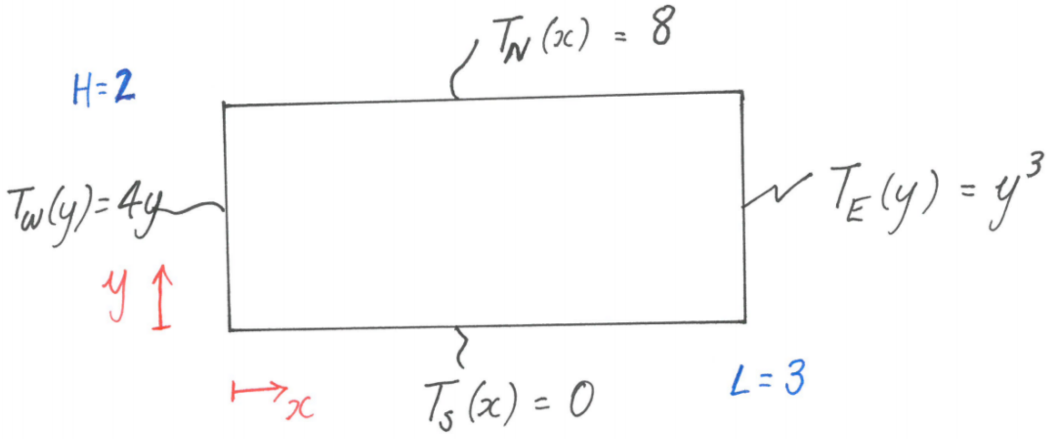
```
X, Y = ##domain length and height
dx, dy= ##chosen discretisation
nx, ny = X/dx + 1, Y/dy + 1
m = lambda i,j: j * nx + i
```

Using this, we can then formulate our matrix,

```python
A = scipy.sparse.lil_matrix((nx*ny,nx*ny))
for j in range(ny):
    for i in range(nx):
        p = m(i,j)
        A[p, m(i, (j-1) % ny)] = # i,j-1 stencil
        A[p, m((i-1) % nx, j)] = # i-1,j stencil
        A[p, p] = # main diagonal
        A[p, m(i, (j+1) % ny)] = # i,j+1 stencil
        A[p, m((i+1) % nx, j)] = # i+1,j stencil
A.tocsr() #compressed sparse row matrix
```

If the problem size is small, using direct methods of solving can suffice, however, when we move to larger scale problems e.g. potentially your assignment.... Iterative methods are often favoured, Gauss-Seidel, Conjugate gradient, steepest descent etc.

We can practice this on the question the lecture:



Example: Find the steady-state distribution of temperature in the plate of length 3 m and height 2 m. Boundary conditions are shown in the figure. Use the finite-difference solution method with 11 nodes in the i-direction and 6 nodes in the j-direction.