Matrix assembly process for the solution of Elliptic equations

(Process only without Python code - See W05 for Python code)

Example

Consider Laplace's equation

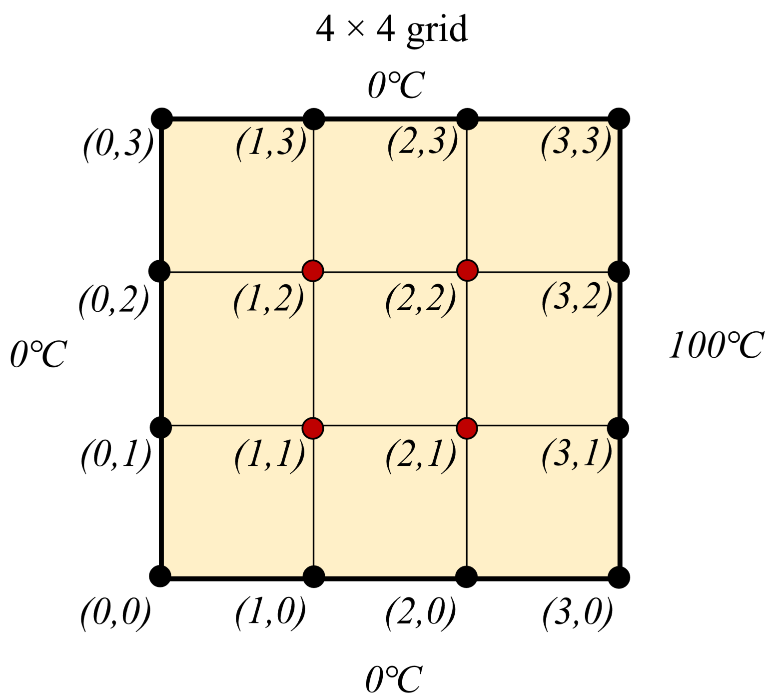
$$0 = T_{xx} + T_{yy}$$

subject to the boundary conditions

$$\begin{aligned} T(x,0) &= 0 \qquad & 0.0 \le x \le 0.2 \\ T(x,0.2) &= 0 \qquad & 0.0 \le x \le 0.2 \\ T(0,y) &= 0 \qquad & 0.0 \le y \le 0.2 \\ T(0.2,y) &= 100 \qquad & 0.0 \le y \le 0.2. \end{aligned}$$

1) Construct the grid

To see how the matrix assembly works, we are going to start with a small grid (therefore small number of equations). We will use a 4×4 grid, as shown in the figure below.



We know that the extent in the x direction is 0.2 m and the extent in the y direction is 0.2 m. Therefore the spacings at each direction are

$$\Delta x = \frac{0.2}{4} = 0.05m$$
$$\Delta y = \frac{0.2}{4} = 0.05m$$

2) Set boundary conditions

We need to set the right boundary $T(0.2, y) = 100$ for $0.0 \leq y \leq 0.2$.

3) Matrix assembly

From the figure above, we see that we have 4 interior points where we want to find the solution (red circles), therefore we will have 4 equations and 4 unknowns.

We can re-arrange the finite difference approximations and form a system of equations to be solved. The solution at each point using Jacobi iteration (other schemes can be used similarly too) is

$$u_{i,j}^{n+1} = \frac{1}{2(1+\beta^2)} \left(u_{i+1,j}^n + u_{i-1,j}^n + \beta^2 (u_{i,j+1}^n + u_{i,j-1}^n) \right)$$

where

$$\beta = \frac{\Delta x}{\Delta y}$$

Moving all terms on one side

$$u_{i,j}^{n+1} - \frac{1}{2(1+\beta^2)} \left(u_{i+1,j}^n + u_{i-1,j}^n + \beta^2 (u_{i,j+1}^n + u_{i,j-1}^n) \right) = 0$$

$$u_{i,j}^{n+1} - \frac{1}{2(1+\beta^2)} \left(u_{i+1,j}^n + u_{i-1,j}^n \right) - \frac{\beta^2}{2(1+\beta^2)} \left(u_{i,j+1}^n + u_{i,j-1}^n \right)$$

Therefore, the coefficients of the neighboring points are:

$$R_x = -\frac{1}{2(1+\beta^2)}$$
$$R_y = -\frac{\beta^2}{2(1+\beta^2)}.$$

Now we can write the Jacobi update equation using R_x and R_y as:

$$R_x \cdot u_{i+1,j}^n + R_x \cdot u_{i-1,j}^n + u_{i,j}^{n+1} + R_y \cdot u_{i,j+1}^n + R_y \cdot u_{i,j-1}^n = 0$$

The unknowns are the solutions $u_{1,1}, u_{2,1}, u_{1,2}$ and $u_{2,2}$ at the corresponding interior points. The 4 equations are:

$$R_x \cdot u_{2,1}^n + R_x \cdot u_{0,1}^n + u_{1,1}^{n+1} + R_y \cdot u_{1,2}^n + R_y \cdot u_{1,0}^n = 0$$

$$R_x \cdot u_{3,1}^n + R_x \cdot u_{1,1}^n + u_{2,1}^{n+1} + R_y \cdot u_{2,2}^n + R_y \cdot u_{2,0}^n = 0$$

$$R_x \cdot u_{2,2}^n + R_x \cdot u_{0,2}^n + u_{1,2}^{n+1} + R_y \cdot u_{1,3}^n + R_y \cdot u_{1,1}^n = 0$$

$$R_x \cdot u_{3,2}^n + R_x \cdot u_{1,2}^n + u_{2,2}^{n+1} + R_y \cdot u_{2,3}^n + R_y \cdot u_{2,1}^n = 0$$

We see that each equation depends on the the known boundary conditions. Substituting the boundary conditions gives

$$R_x \cdot u_{2,1}^n + u_{1,1}^{n+1} + R_y \cdot u_{1,2}^n = 0$$

$$R_x \cdot 100 + R_x \cdot u_{1,1}^n + u_{2,1}^{n+1} + R_y \cdot u_{2,2}^n = 0$$

$$R_x \cdot u_{2,2}^n + u_{1,2}^{n+1} + R_y \cdot u_{1,1}^n = 0$$

$$R_x \cdot 100 + R_x \cdot u_{1,2}^n + u_{2,2}^{n+1} + R_y \cdot u_{2,1}^n = 0$$

We move the known values to the RHS

$$R_x \cdot u_{2,1}^n + u_{1,1}^{n+1} + R_y \cdot u_{1,2}^n = 0$$

$$R_x \cdot u_{1,1}^n + u_{2,1}^{n+1} + R_y \cdot u_{2,2}^n = -R_x \cdot 100$$

$$R_x \cdot u_{2,2}^n + u_{1,2}^{n+1} + R_y \cdot u_{1,1}^n = 0$$

$$R_x \cdot u_{1,2}^n + u_{2,2}^{n+1} + R_y \cdot u_{2,1}^n = -R_x \cdot 100$$

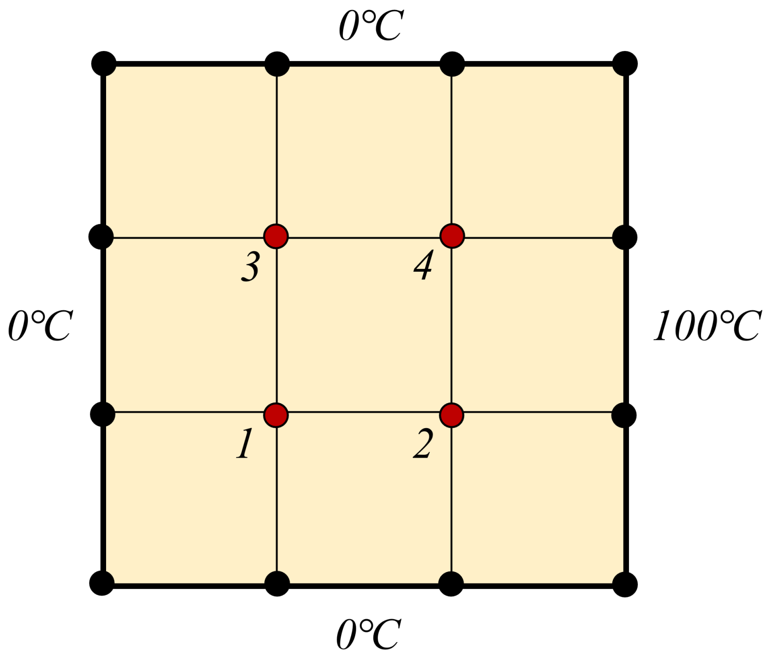
Now we can write this as a system of equations

$$\underbrace{\begin{bmatrix} 1 & R_x & R_y & 0 \\ R_x & 1 & 0 & R_y \\ R_y & 0 & 1 & R_x \\ 0 & R_y & R_x & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{1,2} \\ u_{2,2} \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} 0 \\ -100R_x \\ 0 \\ -100R_x \end{bmatrix}}_{\mathbf{b}}$$

Although not obvious from this small grid, for large system most of the elements of matrix A will be zero. To save storage, in a computer program we are saving the A matrix as a sparse matrix, where only the non-zero elements are stored. The sparse matrix knows the structure of the original matrix and the position of each coefficient in the matrix!

To assemble the A and b matrices, first we change the numbering of each interior point in the grid, i.e. change the indices of each point to the index k such as $u_{i,j} = u_k$, as shown in the figure

$$\text{Index } k = (j - 1)N_i + i$$



Now the 4 interior points are identified by index k = 1,2,3 or 4, respectively, for each point.

4) Finally, we can solve the system of equations Ax=b to find the solution at the interior points using a linear algebra solver (such as the bi-conjugate gradient stabilised matrix solver (BiCGStab) or other scheme).