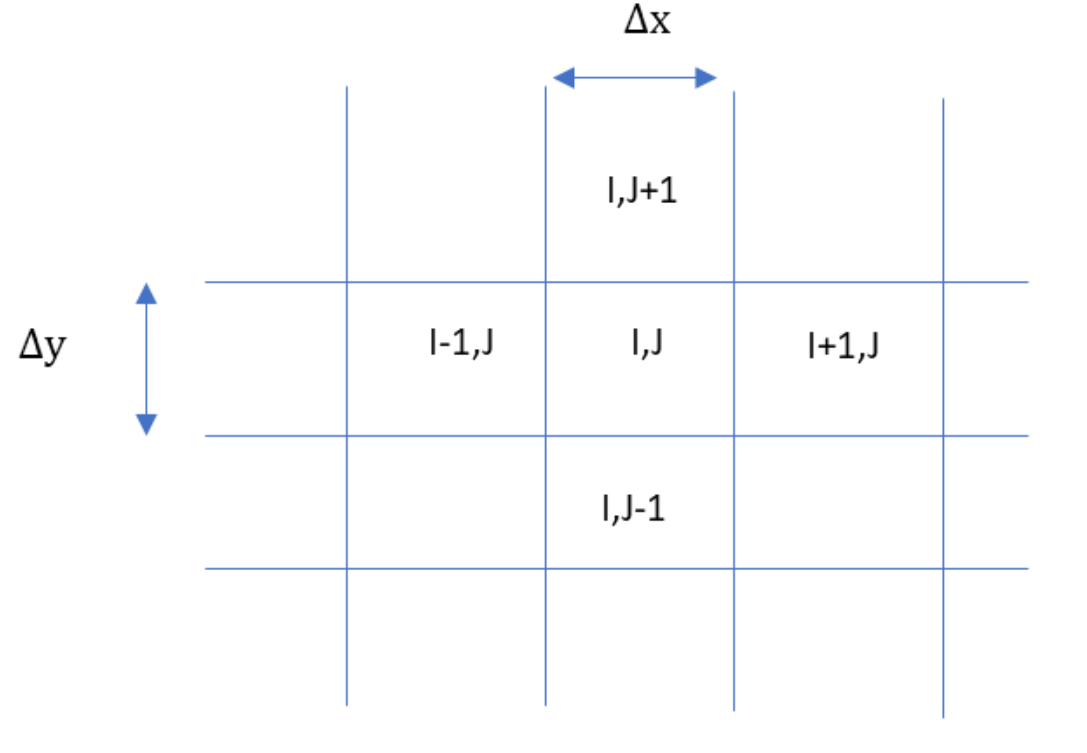
**Mesh Layout Summary**

**Indexing format:**

The vertices are stored in a column-based 2-D array. All the values are stored at the center of each vertex –



The actual variables that are solved for are stored in a 1-D vector Let U(idx) in the vector be U(I,j). So, U(I+1,J) = U(idx+1) U(I-1,J) = U(Idx – 1), U(I,j+1) = U(Idx – Nx) and U(I,j-1) = U(Idx+Nx)

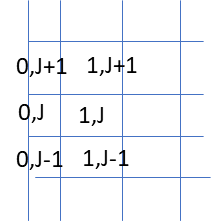
**Edge Values**

The values are defined at the center of the vertices. However, the subsequent equations require the values at edges as well. For these values, an averaging approximation is used –

**U(I+1/2,J) = [U(I,J) + U(I+1,J)]/2**

**Boundary conditions**

The boundary vertices are half-vertices. The edges are defined at the half-zones. For example, see the left boundary –



There are two types of boundary conditions –

1] Fixed Value (Inlet/Outlet) – The variables are explicitly defined here i.e. U(O,J) = Uo [Input by user] . As a result, none of the equations are solved and the computational stencil is simply [0, 0, 1, 0, 0]

2] Symmetry BCs – There is a mirror image about the boundary. Thus the fictitious U(-1,J) = U(1,J)

**Terms used in the equations -**

**Convective terms**

The convective term H for a variable phi stored at (I,J) is given by

**H = (V(I,J+1) \* phi [I,j+1/2] - V(I,J-1) \* phi [I,j-1/2])\*dely + (U(I+1,j) \* phi [I+1/2,J] - U(I,J-1) \* phi [I-1/2,J])\*delX**

At boundaries, the values are appropriately treated. It is made 0 for the inlet/ outlet BC, and for a symmetry BC, the value is mirrored about the boundary as described above.

**Diffusive Terms**

The RHS for diffusion for a vector phi is calculated as –

D = phi [I,J] + mu/rho\*dT/2/([dely\*delx]) \* ((right + left)

Right = phi [I+1,J] + phi [I-1,J] – 2\*phi [I,J]) \* dely/delx

Left = (phi[I,J+1]+phi[I,J-1] – 2\*phi[I,J]) \* delx/dely

**Coefficients for the tridiagonal solution**

The velocity components are solved as a linear tridiagonal system. They are solved once along the x-coordinate, and once along the y-coordinate. While solving along the y-coordinate, the vectors need to be reoriented in a row-based ordering to make phi [I – Nx] as phi [j-1].

The diagonals for the coefficients are –

Low :- -mu/rho\*dT/2/([dely\*delx]) \* phi [I+1,J] + phi [I-1,J] – 2\*phi [I,J]) \* dely/delx

High :- -mu/rho\*dT/2/([dely\*delx]) \* phi [I+1,J] + phi [I-1,J] – 2\*phi [I,J]) \* dely/delx

Main :- 1 + 2 \* mu/rho\*dT/2/([dely\*delx]) \* phi [I+1,J] + phi [I-1,J] – 2\*phi [I,J]) \* dely/delx

It is the same for discretization along y axis but only dely/dex is replaced by delx/dely

**RHS for the pressure Poisson equation**

The RHS for the pressure poisson equation is –

RHS = (U[I+1,J] – U[I-1,J])/(2 \*delx) + (V[I,J+1] – V[I,J-1])/(2 \*dely)

**Coefficients for the Pressure Poisson Equation**

Each P(I,j) will be solved by the Pressure Poisson equation. The coefficients of the pressure Poisson equation are –

**delT/rho / (dely\*delx) \* (P[I,J+1] + P[I,J-1] – 2\*P[I,J] ] \*dely/delx + (P[I+1,J] + P[I-1,J] – 2\*P[I,J]) \*delx/dely**