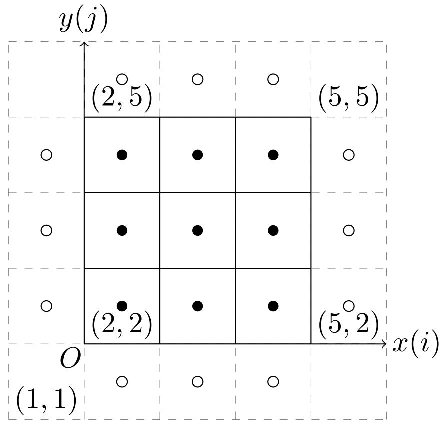
1 Poisson

When I started my junior research project, I learned how to solve Poisson with Dirichlet conditions easily using finite differences. It is much faster than the explicit method and the error has the same order as long as you use the same formulas. Feel free to correct me if I am wrong. You may want to know a little of finite differences before proceeding.

1.1 Grid

Before we decide which formulas to use, let's choose a grid. I though the staggered grid would be better as I would get second order central derivative easily for the boundary conditions. I would need a one-sided finite difference formula in the case of a grid where the values are stored in the corners, instead of the center of the cells in the mesh. An example of the grid is available below. We will show the formulas to each of the cells, so it is easier to have a small grid and then you find out an algorithm to create the matrix.



1.2 Discretization

For the discretization, I will use second order finite differences. Let's begin! We want to solve the equation:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x,y) \tag{1}$$

Let's also consider that in our square grid, x and y vary from 0 to 1, so we have a unit square. Central differences of second order for the partial second derivatives will be used. For an internal point, one has:

$$\frac{u_{i-1j} - 2u_{ij} + u_{i+1j}}{\Delta x^2} + \frac{u_{ij-1} - 2u_{ij} + u_{ij+1}}{\Delta x^2} = f_{ij} \tag{2}$$

For the explicit method, you just isolate u_{ij} and iterate. On the other hand, for the implicit method, one has to consider all equations that the system gives you in order to solve it. In this case, one has to solve a problem of the type:

$$Ax = b \tag{3}$$

Matrix A depends on our discretization and also the boundary conditions if they are of the type Neumann. b is made with values of f and boundary conditions. x is a vector that has all the internal points of the grid. The external points are easy to find once you have the internal ones, you just need to apply the equations for the boundary conditions. Now, for the nine internal points, we have:

$$-4u_{22} + u_{32} + \boldsymbol{u_{12}} + u_{23} + \boldsymbol{u_{21}} = \Delta x^2 f_{22} \tag{4}$$
$$-4u_{23} + u_{33} + \boldsymbol{u_{13}} + u_{24} + u_{22} = \Delta x^2 f_{23} \tag{5}$$
$$-4u_{24} + u_{34} + \boldsymbol{u_{14}} + \boldsymbol{u_{25}} + u_{23} = \Delta x^2 f_{24} \tag{6}$$

$$-4u_{32} + u_{42} + u_{22} + u_{33} + \boldsymbol{u_{31}} = \Delta x^2 f_{32} \tag{7}$$
$$-4u_{33} + u_{43} + u_{23} + u_{34} + u_{32} = \Delta x^2 f_{33} \tag{8}$$
$$-4u_{34} + u_{44} + u_{24} + \boldsymbol{u_{35}} + u_{33} = \Delta x^2 f_{34} \tag{9}$$

$$-4u_{42} + \boldsymbol{u_{52}} + u_{32} + u_{43} + \boldsymbol{u_{41}} = \Delta x^2 f_{42} \tag{10}$$
$$-4u_{43} + \boldsymbol{u_{53}} + u_{33} + u_{44} + u_{42} = \Delta x^2 f_{43} \tag{11}$$
$$-4u_{44} + \boldsymbol{u_{54}} + u_{34} + \boldsymbol{u_{45}} + u_{43} = \Delta x^2 f_{44} \tag{12}$$

The bold points are boundary points. Notice we have 9 equations, but there are 21 unknown values. What to do now? The rest of the equations are actually defined by boundary conditions. As we have Neumann conditions, those equations define the derivatives on the boundary. We have the following:

$$u_x(0,y) = (u_{2j} - u_{1j})/\Delta x \tag{13}$$
$$u_x(1,y) = (u_{5j} - u_{4j})/\Delta x \tag{14}$$
$$u_y(x,0) = (u_{i2} - u_{i1})/\Delta x \tag{15}$$
$$u_y(x,0) = (u_{i5} - u_{i4})/\Delta x \tag{16}$$

From the equations above, we get the twelve equations that the system is missing. Instead of adding it directly as new equations, one could substitute them in the nine equations from before. I will let the substitution to the reader and I will skip for the end matrix system.

$$\begin{bmatrix} -2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -3 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} u_{22} \\ u_{23} \\ u_{24} \\ u_{32} \\ u_{33} \\ u_{34} \\ u_{42} \\ u_{43} \\ u_{44} \end{bmatrix} = b \tag{17}$$

$$b = \Delta x \begin{bmatrix} u_x\left(0, \frac{\Delta x}{2}\right) + u_y\left(\frac{\Delta x}{2}, 0\right) \\ u_x\left(0, \frac{3\Delta x}{2}\right) \\ u_x\left(0, \frac{5\Delta x}{2}\right) - u_y\left(\frac{\Delta x}{2}, 1\right) \\ u_y\left(\frac{3\Delta x}{2}, 0\right) \\ 0 \\ -u_y\left(\frac{3\Delta x}{2}, 1\right) \\ -u_x\left(1, \frac{\Delta x}{2}\right) + u_y\left(\frac{5\Delta x}{2}, 0\right) \\ -u_x\left(1, \frac{3\Delta x}{2}\right) \\ -u_x\left(1, \frac{5\Delta x}{2}\right) - u_y\left(\frac{5\Delta x}{2}, 1\right) \end{bmatrix} + \Delta x^2 \begin{bmatrix} f_{22} \\ f_{23} \\ f_{24} \\ f_{32} \\ f_{33} \\ f_{34} \\ f_{42} \\ f_{43} \\ f_{44} \end{bmatrix} \tag{18}$$

Before implementing that, let's take a look at A's eigenvalues:

$$\left[-6, -4, -4, -3, -3, -2, -1, -1, -2.5327 \times 10^{-16}\right] \tag{19}$$

Those values were obtained with the aid of a computer. Well, the last value seems like 0. It the eigenvalue of a matrix is 0, it doesn't have an inverse so we can't use it to solve a linear system. Well, what about pseudo-inverse? I thought of that, but I haven't tried it. What does this eigenvalue of zero mean? Well, it means that you have one degree of freedom. You can actually pick a point to be what you want and then solve it. I want the gradient of the function only, for my problem of Navier Stokes. Maybe you want a function with mean 0 for your purposes... or whatever, you can just select a point to the value you want and check if the matrix will be invertible. Choose one of the boundaries. I chose $u_{12} = 0$. The only modifications are:

$$A[1,1] = -3 \tag{20}$$

$$b[1] = u_y\left(\frac{\Delta x}{2}, 0\right)\Delta x + \Delta x^2 f_{22} \tag{21}$$

You can make an algorithm for that. I have made one as well, I wrote it in Julia language.[1]

Any suggestions, feel free to contact me through the gist or at ataiasreis at gmail.

[1]Code available at: `https://gist.github.com/ataias/05ebdad6dfc6ee56e44f`