SCHEDULE

- Feedback previous exercise
- Solving Laplace equation using the Liebmann method
- Reminder: Final in-class test material 2nd term: Wednesday 13th of May. Allowed is your logbook and Blackboard (including lecture notes). You can only use the university-provided computers.

FEEDBACK PREVIOUS EXERCISE ON NEUMANN BOUNDARY CONDITION

To incorporate the Neumann boundary condition on the diffusion equation, note that one has

$$\frac{d}{dx}u(t,x)\Big|_{x=x_L} = d$$

which means that the derivative at the boundary x_L is a constant d.

Discretising as before, this implies for the boundary at $x = x_0$ or $n = 0$:

$$u_{i+1,-1} = u_{i+1,1} - 2\Delta x d$$

For simplicity let us take $d = 0$ (insulating boundary condition), so that $u_{i+1,-1} = u_{i+1,1}$.

Hence the finite difference scheme around point $n = 0$ equals

$$\begin{aligned}
-cu_{i+1,1} + (1+2c)u_{i+1,0} - cu_{i+1,-1} &= u_{i,0} \\
-cu_{i+1,1} + (1+2c)u_{i+1,0} - cu_{i+1,1} &= u_{i,0} \text{ subsititute expression for virtual point } u_{i+1,-1} \\
-2cu_{i+1,1} + (1+2c)u_{i+1,0} &= u_{i,0} \text{ rearrange}
\end{aligned}$$

Hence the system of equations become

$$
\begin{aligned}
-2cu_{i+1,1} + (1+2c)u_{i+1,0} &= u_{i,0} \\
-cu_{i+1,2} + (1+2c)u_{i+1,1} - cu_{i+1,0} &= u_{i,1} \\
-cu_{i+1,3} + (1+2c)u_{i+1,2} - cu_{i+1,1} &= u_{i,2} \\
&\vdots \\
-cu_{i+1,n+1} + (1+2c)u_{i+1,n} - cu_{i+1,n-1} &= u_{i,n} \\
&\vdots \\
-cu_{i+1,N_x-1} + (1+2c)u_{i+1,N_x-2} - cu_{i+1,N_x-3} &= u_{i,N_x-2} \\
u_{i+1,N_x-1} &= u_R
\end{aligned}
$$

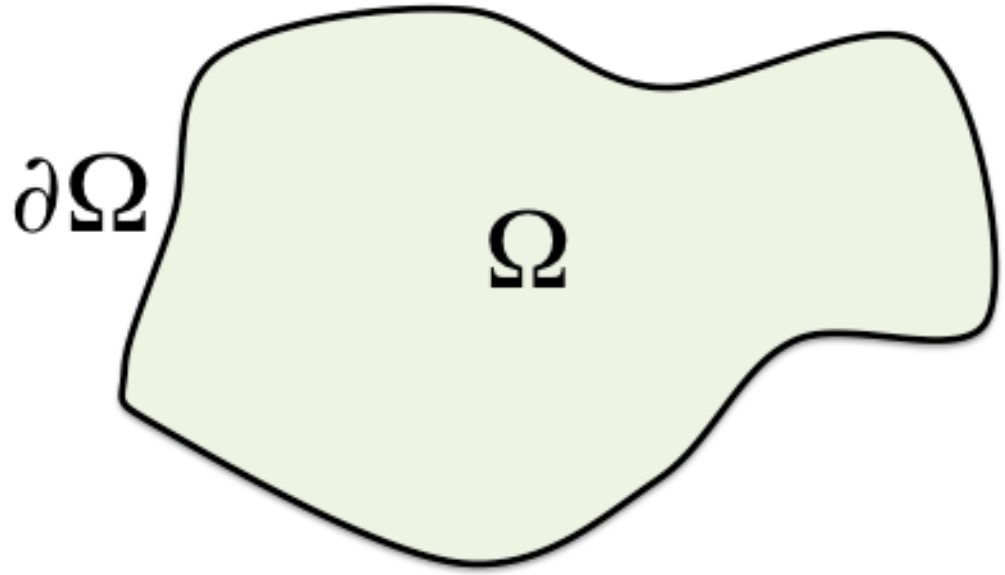
Note that the first equation has now changed due to the insulating boundary condition.

LAPLACE EQUATION

The Laplace equation reads

$$\nabla^2 u(\vec{r}) = 0$$

The solution of u in the domain Ω only depends on the boundary conditions. If u is specified at the



boundary $\partial\Omega$, then this will lead to a unique solution.

1D

The one-dimensional variant, $d^2u(x)/dx^2 = 0$, was already discussed by Matt in the first term.

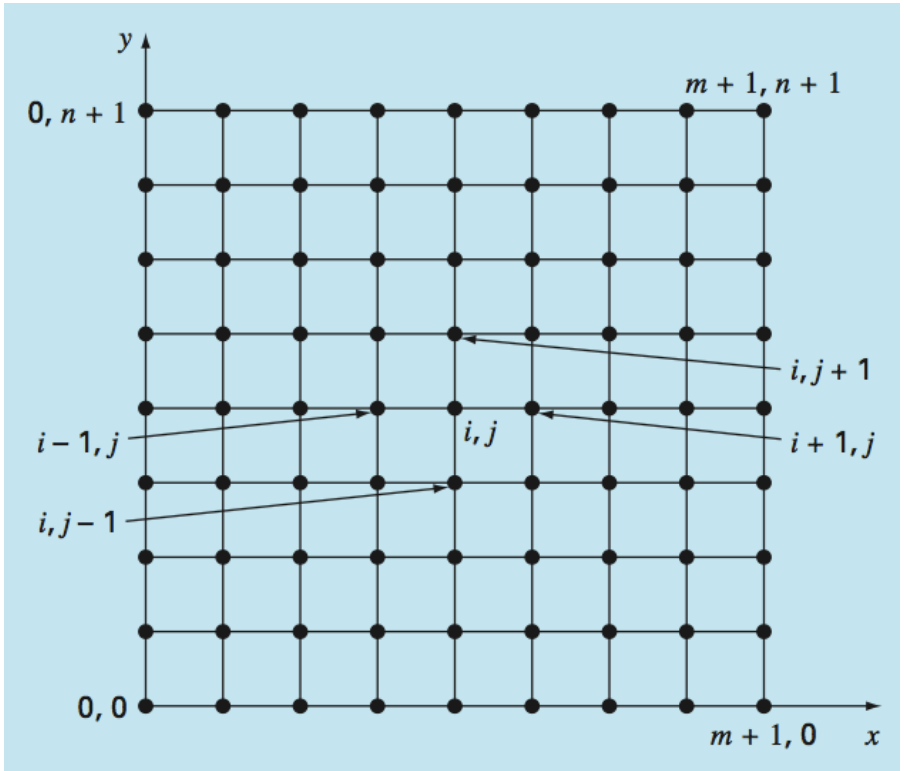
2D

The Laplace equation in two dimensions simplifies to

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

NUMERICAL SOLUTION FOR 2D

To solve it numerically we switch from continuous variables to discrete points in the $x - y$ plane:



$x = x_0 + i\Delta x$ and $y = y_0 + j\Delta y$, where now i and j are integers.

We found before that one can use finite differences to approximate second derivatives:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}$$

and

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}$$

The Laplace equation then becomes

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

Assuming a square grid, $\Delta x = \Delta y$, this simplifies to

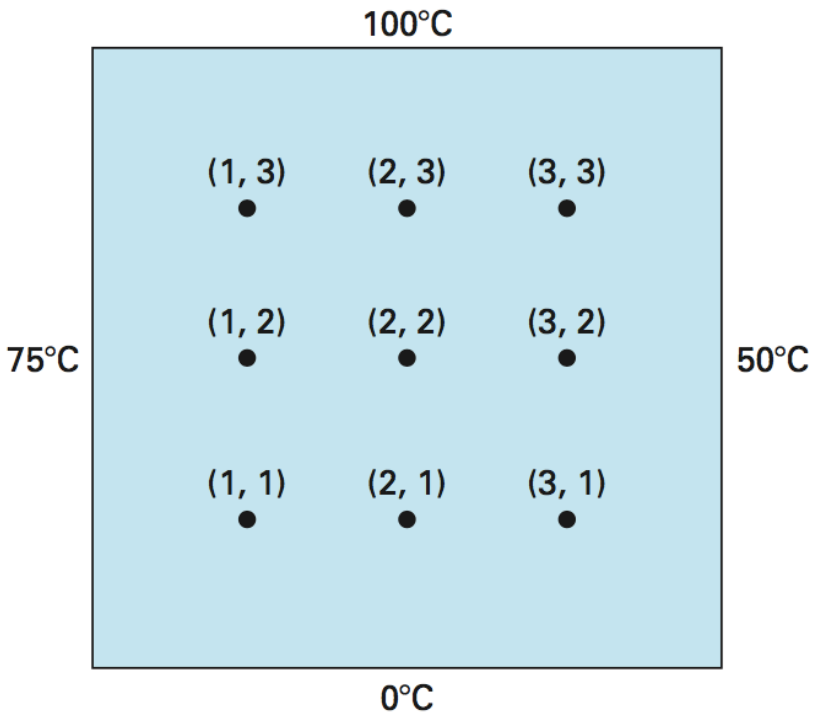
$$u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + u_{i,j+1} - 2u_{i,j} + u_{i,j-1} = 0$$

or

$$\boxed{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = 0}$$

This is called the *Laplacian difference equation*.

Now consider the following example



Consider the point $i = 1, j = 1$ (1,1):

Then the aformentioned Laplacian difference equation becomes for this node

$$u_{2,1} + u_{0,1} + u_{1,2} + u_{1,0} - 4u_{1,1} = 0$$

Now the temperature at the left is 75 degrees, so $u_{0,1} = 75$ degrees and at the bottom it is 0 degrees so $u_{1,0} = 0$ degrees.

$$u_{2,1} + u_{1,2} - 4u_{1,1} = -75$$

One can do the same for the other points.

This will give a system of equations. These can, in principle, be solved by using matrix inversion. This, however, can take a long time to compute for large systems.

A better method is to use the **Liebmann's method** in PDE's (it is also known as the **Gauss-Siedel** method). It goes as follows

Rewrite the Laplacian difference equation

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = 0$$

as

$$u_{i,j} = \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{4}$$

and call the LHS new and the RHS old:

$$\boxed{u_{i,j}^{\text{new}} = \frac{u_{i+1,j}^{\text{old}} + u_{i-1,j}^{\text{old}} + u_{i,j+1}^{\text{old}} + u_{i,j-1}^{\text{old}}}{4}}$$

This will be our update scheme.

Now repeat this till the u's don't change any more.

It can be shown that this produces the correct solution.

(the reason that it converges is that the matrix A associated with the system of equations is *diagonally dominant*, meaning that $|a_{ii}| \geq \sum_{j \neq i} |a_{ij}|$ for all i)

One repeats the iteration till the change is less than a small number: $\left|u_{i,j}^{\text{new}} - u_{i,j}^{\text{old}}\right| < \epsilon$ for all i, j. E.g., $\epsilon = 10^{-4}$.

Example for $L = 10$ cm. The boundary conditions are at $x = 0$ and $x = L$, and at $y = 0$ and $y = L$. The remaining nodes are the internal points (the values for the corners of the square don't matter for the internal points).

HOW TO IMPLEMENT?

1. Create two 2-dimensional arrays: `u_xy` and `unew_xy` with sizes equal to the amount of grid points $N_x \times N_y$.

2. Initialize `u_xy` with initial condition (this can be an arbitrary value, for example 0)

3. Set the boundary conditions in the array `u_xy`

4. Loop over the internal points using a double loop, one for the index corresponding to x and one for the index corresponding to y, and set `unew_xy` for each point according to our update scheme by using the values in `u_xy`.

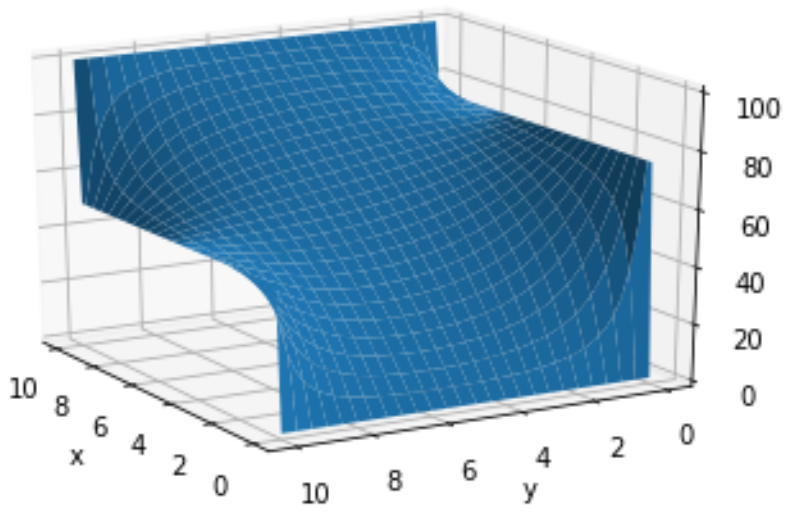
5. Calculate the maximum difference between `u_xy` and `unew_xy`. For this, one could first set the maximum difference to zero. Then one could use a double loop and calculate the difference betweeen the two arrays at the specific array index and compare it with the maximum difference so far. If the former is larger, update the maximum difference so far.

6. Replace the values of `u_xy` by the values of `unew_xy`.

7. If the maximum difference in step 5 is still larger than the error threshold, then repeat steps 4--7.

SOLUTION

SOLUTION



EXERCISES SESSION 10

- What is Liebmann's method, and what type of equation can be solved with it?
- See assessment section on Blackboard.