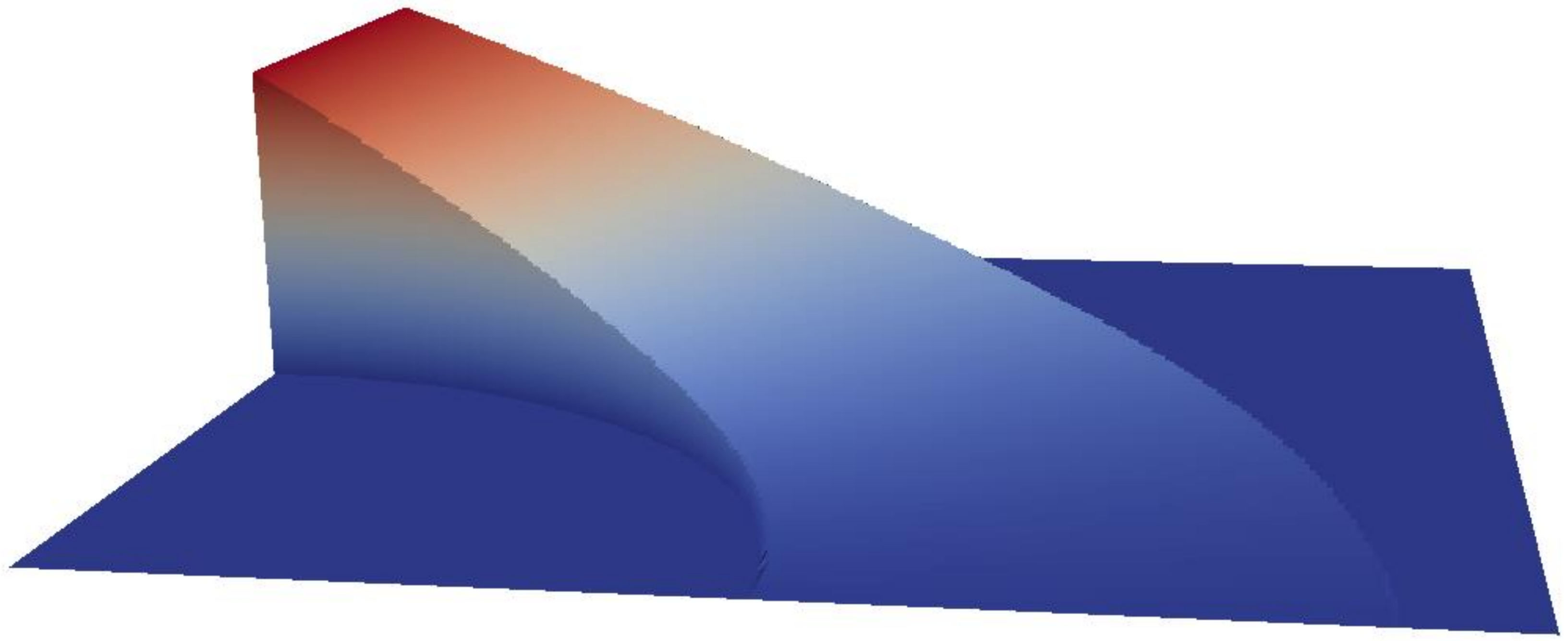
Project 2 – Heat Equation



Introduction

The heat equation is a standard partial differential equation that models pure diffusive transport, such as molecular diffusion or heat conduction.

In this project you should implement an algorithm to solve the stationary heat equation iteratively on complex domains.

The numerical scheme and the set up of the simulations will be explained on the following slides.

Mathematical background

The stationary heat equation for temperature T in two dimension is:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

The solution for $T(x,y)$ can be obtained by discretizing this equation with finite differences. The resulting linear system of equations is:

$$T(\mathrm{x} + \Delta x, y) + T(\mathrm{x} - \Delta x, y) + T(\mathrm{x}, y + \Delta x) + T(\mathrm{x}, y - \Delta x) - 4\mathrm{T}(\mathrm{x}, \mathrm{y}) = 0$$

In addition to the partial differential equation, boundary conditions are required to obtain a unique solution. These boundary conditions are the definition of temperatures on the boundary. In this project we consider in inhomogenous Diriclet boundary conditions, where the temperature of the boundary is fixed to any value, and homogenous Neumann boundary conditions, where the heat flux over the boundary is zero.

Solution of the linear system

The discretized heat equation results in a linear system, that can be solved by the iterative Gauss-Seidel algorithm.

This algorithm formulated for an arbitrary linear system

$$A\,x = b$$

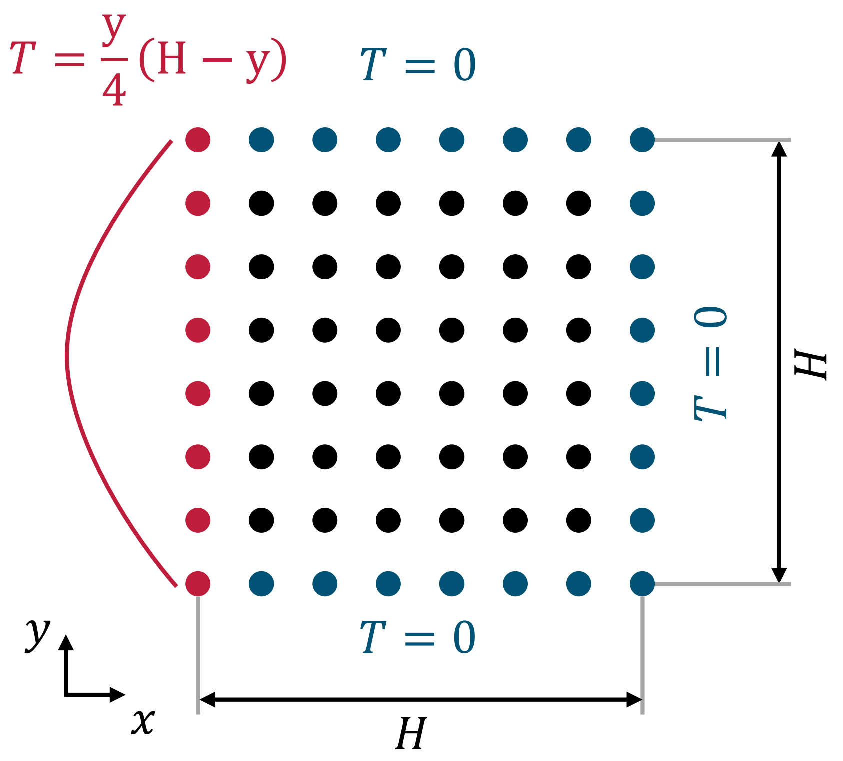
is

$$x_i^{n+1} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=0}^{i-1} a_{ij} x_j^{n+1} - \sum_{j=i+1}^{N} a_{ij} x_j^{n}\right).$$

Therein a_{ij} are the coefficients of the linear system. N is the number of degrees of freedom.

As a first task think about a way how to efficiently implement this algorithm on a two-dimensional grid for the heat equation.

1st Test: inhomogeneous Diriclet boundary conditions



In the first setup the temperature on three walls is set to zero and the temperature on the fourth wall is a quadratic function that is zero at first and last node.

Test this setup with the three resolutions 65×65, 129×129 and 257×257.

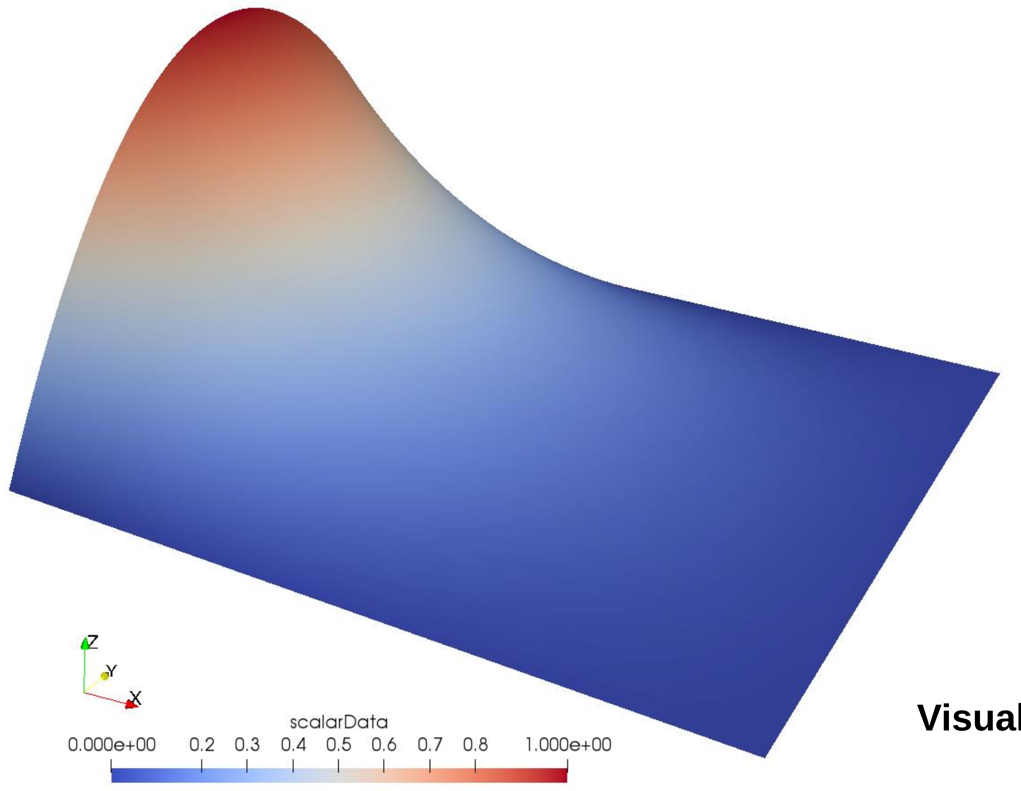
For each resolution compute the steady state temperature of the node in the center.

Compute the empirical rate of convergence by

$$q = -\ln\left(\frac{T_{257} - T_{129}}{T_{129} - T_{65}}\right)/\ln(2)$$

Is the result as expected?

1st Test: inhomogeneous Diriclet boundary conditions



Visualization: WarpByScalar filter

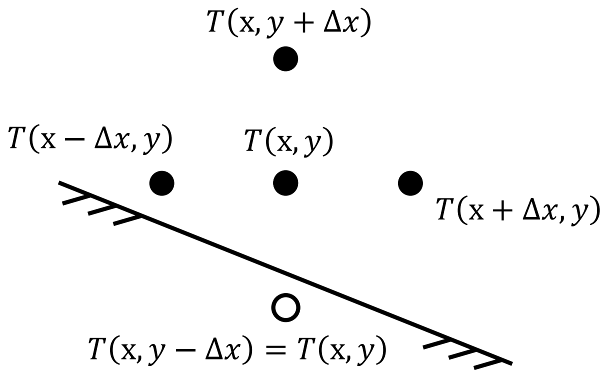
Homogeneous Neumann boundary condition

Homogeneous Neumann boundary conditions model a wall, where no flux passes the wall. For the heat equation this is an insulated wall.

Nodes outside the these walls, and hence outside the simulation domain, do not have to be updated. In the update of adjacent nodes the value of the outside nodes can be approximated as the value of the current node. For the example on the right that means:

$$T(\mathrm{x}, y - \Delta x) = \mathrm{T}(\mathrm{x}, \mathrm{y})$$

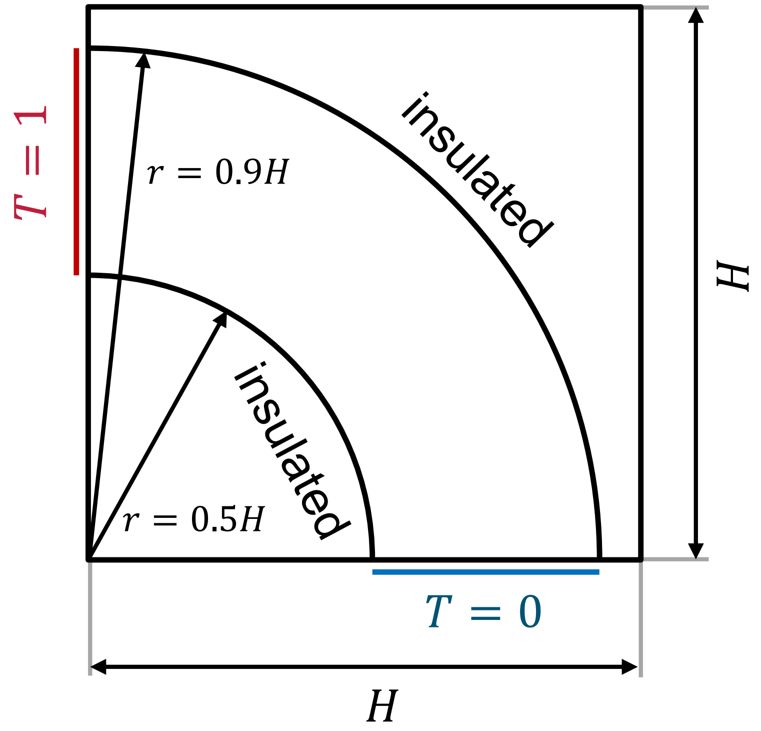
The nodes inside the simulation domain are updated as usual.


$T(\mathrm{x}, y + \Delta x)$
$T(\mathrm{x} - \Delta x, y)$
$T(\mathrm{x}, y)$
$T(\mathrm{x} + \Delta x, y)$
$T(\mathrm{x}, y - \Delta x) = T(\mathrm{x}, y)$

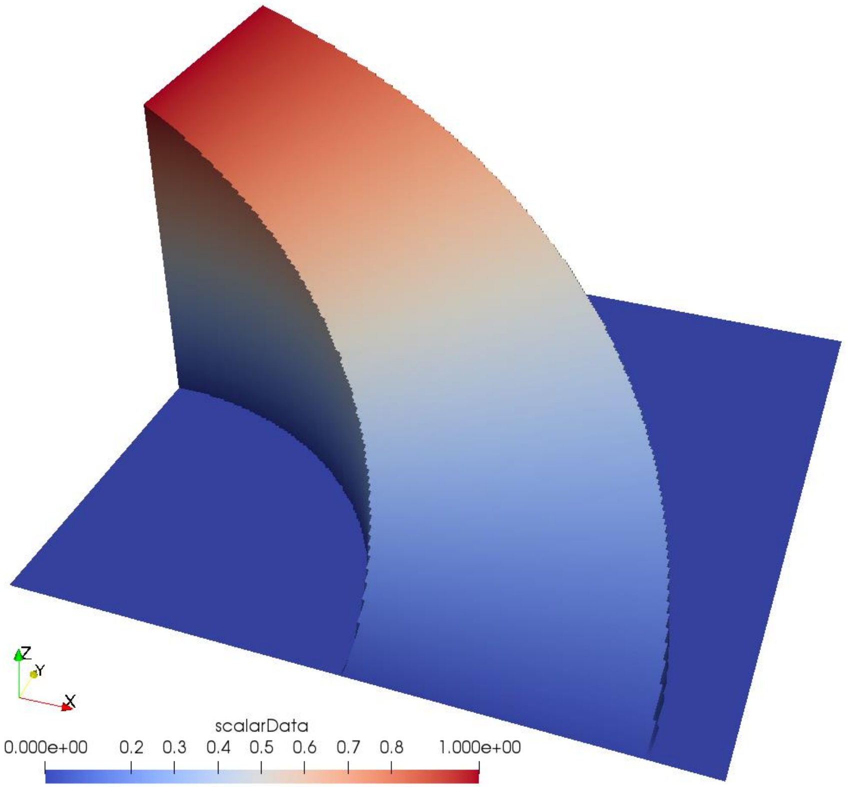
2nd Test: homogeneous Neumann boundary conditions

In this second test you should solve the heat equation on a curved domain. You can embed this curved domain in your regular grid.

The inner radius is $0.5H$ and the outer radius $0.9H$. The temperature on the top left is one and on the bottom right zero.



2nd Test: homogeneous Neumann boundary conditions



Visualization: WarpByScalar filter

Submission Requirements

- Your code as CMake project. It should compile without errors with the gcc compiler.
  - The code should be commented, such that it is straight forward to understand.
- Written documentation (PDF file) of your project of maximal two DIN A 4 pages. This should only contain text, pictures and equations, but no code!
  - This document should also contain the convergence study of the 1st task.
- Pictures of the solutions of both tasks in JPEG or PNG format.

⇒ Everything should be packed in one zip file!