CHALMERS UNIVERSITY OF TECHNOLOGY
Fluid Dynamics, M2
412 96 Göteborg

MTF073 Computational Fluid Dynamics

H. Nilsson November 21, 2025

Task 2

You should implement a code that solves the unsteady two-dimensional convection-diffusion equation for temperature without any source term, given by

$$\rho\frac{\partial T}{\partial t}+\frac{\partial}{\partial x}\left(\rho uT\right)+\frac{\partial}{\partial y}\left(\rho vT\right)=\frac{\partial}{\partial x}\left(\Gamma\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\Gamma\frac{\partial T}{\partial y}\right),\qquad \Gamma=\frac{k}{c_p}. \tag{1}$$

The fully implicit scheme should be used for the time discretization. The central differencing scheme should be used for diffusion. The hybrid scheme should be used for the fluxes at the faces. See section 5.7 in the book, but note that **you have to derive the scheme for non-equidistant grids!** Note that you will be given a velocity field in the nodes, and continuity will not be fulfilled when linearly interpolating the velocity to the faces (to get the face fluxes). Nevertheless, you should **omit the continuity error in the final discretized equations**.

The template code is set up to easily switch between **steady** and **unsteady** modes, and your code should be written so it can use that switch (in the way it is provided). You should understand how that works.

The linear system should be solved using either **Gauss-Seidel** or **TDMA**, using the supplied switch.

The task should be carried out using Python in the same groups as in Task 1. This document contains all the relevant details (such as fluid properties, boundary conditions, etc.) that are pertinent to each group. The computational grid and the velocity field in the nodes are provided and automatically imported by the Python template.

Your group should work on one of the 25 cases shown at the end of the document. Your group number specifies your case. For example, group 20 should choose case 20, group 30 should pick case 5, and group 50 should choose case number 25, and so on.

Please read the entire document carefully before proceeding with the task.

How to proceed

1. **The very first step when implementing a code is to NOT start implementing it!**. Instead, do these absolutely necessary preparations:

   - Have a look at your case description at the end of this document and make sure that you understand it.
   - Start by discretizing eq. 1 (on paper) for a non-equidistant mesh and a constant time step, yielding the discretized equation in standard form with identified coefficients (a_E, a_W, a_N, a_S and a_P) and source terms (S_u, and S_P). The discretization must be performed in accordance to your case description. Make sure that the discretization is as implicit as possible.
   - Have a particular look at the discretization for the first nodes inside the boundaries to make sure that the given boundary conditions are introduced correctly (as implicit as possible). The first nodes inside the boundaries may have different discretization compared to the interior nodes. **Note that you now have both walls and inlets/outlets, which may influence the discretization!**
   - Download the Python code template and identify **the nomenclature that you HAVE TO USE in the code**.
   - Rewrite (on paper) your final discretization(s) with the correct code nomenclature and indices, to make it easier for you to do the implementation.
   - Investigate the template code structure and make up a plan for the implementation. Think about how to check every part of the code separately, since you will not be able to implement the entire code correctly at your first attempt.

2. Start implementing the code, following the provided code template with its REQUIRED nomenclature and code structure. This reduces your risk to approach problems. In particular, the assistants will NOT help out with codes that do not follow the template.

   - The template has a section where the grid and velocity is imported from a directory that you also need to download and locate in the same directory as the Python code. Do not make any changes to that section! This data is available both for a coarse and a fine grid, and you can figure out from the template how to switch between these. Use the coarse grid when developing and the fine grid for the final analysis.
   - Identify the sections of the code that you need to implement, and carefully implement what you have derived on paper. Don't implement the entire code at once, but focus on one section at a time and make sure that you did it correctly. Use the provided test code and the Variable Explorer to check that you get correct values.

- Normalize your residuals as described in the section on Convergence in the present document.
- A few notes:
  - The temperatures in the case descriptions are given in Kelvin, and the plots in the template use that unit. Just use the values that are provided for your case (do not add any number to transform them). The solution to this particular task does not depend on if the temperature is in Kelvin or Celsius. Think about why that is the case, and when it would matter.
  - The homogeneous Neumann boundary condition should be applied unless otherwise stated.
  - Any heat flux specified in your case description is defined as positive INTO the computational domain. Note that the equation is divided by the constant specific heat c_P, so any provided heat flux must be divided by c_P. Make sure that you understand why.
  - Since the equation is divided by the constant specific heat, c_P, you need to multiply by c_P if you need to calculate the actual heat flux/rate.
  - You can identify inlets/outlets by the velocity at the boundaries. The template makes sure that the velocity at the walls is zero, which can be used to make sure that your code is general under those conditions.
  - Make sure that your code is as implicit as possible, to speed up the simulations!
  - The template code provides a possibility to animate the unsteady results (in a file named `animated_contour.gif`). This makes it possible to see how the temperature changes over time. Be careful not to save too many frames, overloading the memory of the computer. It also takes some time to save the animation, so keep it off if you don't need it.

Convergence

Compute the residual as

$$\varepsilon = \frac{1}{F}\left(\sum_{\text{all cells}} |a_P T_P - (a_E T_E + a_W T_W + a_N T_N + a_S T_S + S_u)|\right),$$

where F is a normalization factor. In this task we will simply **use the first non-normalized residual at the first time step as normalization factor**.

The solution may be considered as converged when $\varepsilon < 0.001$ (at each time step or for steady-state). You can of course change this threshold to see if any differences are visible.

In unsteady mode, the template code monitors and plots the values of the

dependent variable in a number of probe positions. You can relocate those probes and add more probes (just extend the `probeX` and `probeY` lists) where the dependent variable changes the most, to see that the values at all probes asymptotically reach steady values. The steady and unsteady results should be the same after a sufficient time.

Reporting of the work

The work should be presented in a short report of maximum 12 pages (without the code). **This is a strict limit!** The Python code **must** be submitted separately so it can be executed to check that it works (for your original case only, prepared for both coarse and fine grids).

The report must include the following parts:

- For your original case:
  - A clear description of your case.
  - A derivation of the discretized equation for:
    - an internal node,
    - a node with a wall at one boundary face,
    - a node with an inlet at one boundary face,
    - a node with an outlet at one boundary face,

    for a non-equidistant mesh.
    Start from eq. 1 and end up with the expressions for a_P, a_E, a_W, a_N, a_S, S_P, and S_u.
  - Comparisons between your steady-state results for the coarse mesh and the provided validation results. Use a plot of the temperature contour and the wall-normal heat flux vectors (calculated from the internal temperature gradient).
  - A comparison of steady-state temperature contour and wall-normal heat flux vectors with coarse and fine meshes. Discuss the results from a physical perspective and causes of differences in the results using the different meshes.
  - Other discussions of the steady-state results from a physical and numerical point of view. Present relevant results to support your findings.
  - Other comparisons of relevant steady-state results from the meshes you have used.
  - An analysis of sensitivity to the linear solver for steady-state. Which one is faster (note that number of iterations is not necessarily related to speed)? Does an alternating of directions for the sweeps of the TDMA solver influence the convergence rate and speed?
  - Plot(s) of normalized residuals against the number of iterations. Note that the residual plot must be reported using a log-scale on the y-axis! Describe why the plot looks the way it looks in steady vs. unsteady mode.

- Plot(s) and discussions of the development of the dependent variable in relevant points along time. Make sure that it asymptotically approaches the correct solution at steady-state (both in steady and unsteady mode).
- In unsteady mode, an investigation of the sensitivity of the time step. Discuss/show how you have found a time step that is small enough so the time-evolution of the dependent variable does not change with further reduction of the time step. The final (steady-state) result should be independent of the time step length, but not the evolution of the solution (unless it is small enough).

• Broader studies (also compulsory):

- An analysis of the sensitivity to the conductivity k and specific heat c_p. Increase and reduce the values by a factor of 100 (or any other value that gives interesting things to discuss). Describe how the solution changes and explain what you observe from a physical perspective.

Feel free to implement any other case. Note in particular that the derivations for all settings from all cases may appear at the written exam.

Geometry and pre-calculated velocity field

A general description of the geometry and boundaries for all the cases is presented in Fig. 1. Capital letters A, B, C and D indicate the positions of possible inlets/outlets. The provided template code includes a section for plotting the pre-calculated velocity field (in the nodes), showing where your case has inlets/outlets. The template makes sure that the velocity at the walls is exactly zero, so you can also identify inlets and outlets by identifying boundary velocity vectors pointing into or out of the computational domain (is useful when you implement the code!).

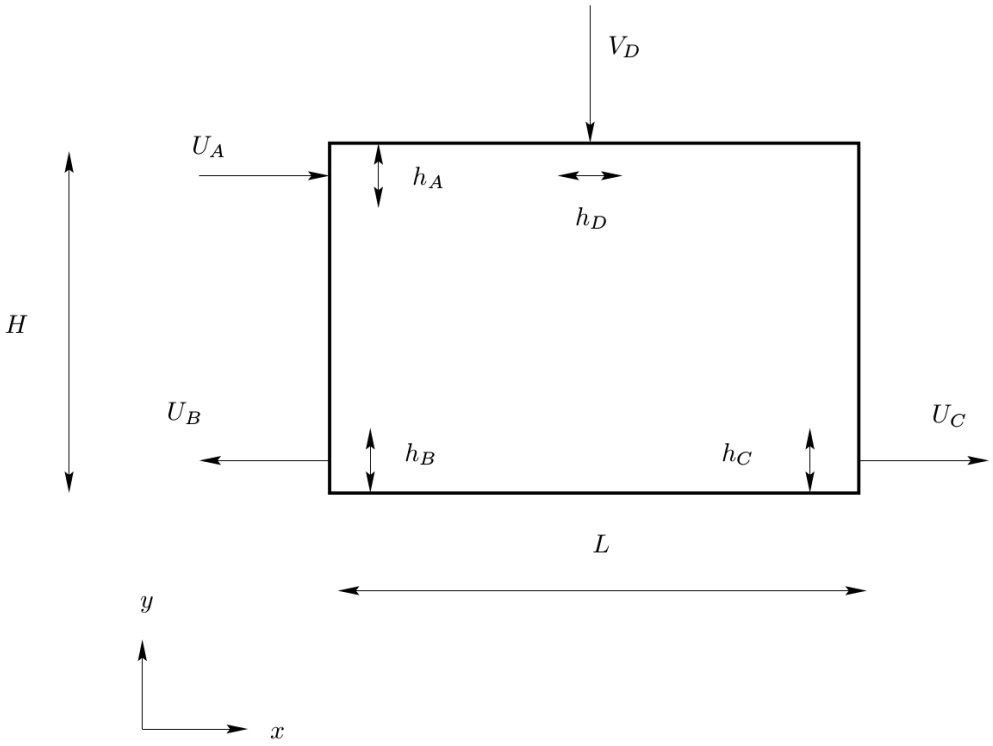

Figure 1: General description of geometry and boundaries.

Case descriptions

Settings for all the cases are given below. NOTE that any specified wall heat flux (q_{in}) is positive into the computational domain! Temperatures are given in Kelvin. It is really cold, but it doesn't matter for this exercise.

The internal initial temperature is set to zero in the provided template, and should be kept like that for the studies.

Cases 1 to 5

Grid 1 (automatically imported in template).
Physical data: $\rho = 1$, $k = 1$ and $c_p = 500$.
Inlet boundary conditions: $T_A = 20$.
Outlet boundary conditions: $\partial T/\partial n = 0$.
Wall boundary conditions not in table: $\partial T/\partial n = 0$.
Any specified wall heat flux, q_{in}, positive INTO domain!

Case	b.c.
1	$y = H \rightarrow T = 10$
2	$y = H \rightarrow q_{in} = 100\ \mathrm{W/m^2}$
3	$x = L \rightarrow T = 10$
4	$x = L \rightarrow T = 50$
5	$y = 0 \rightarrow q_{in} = 100\ \mathrm{W/m^2}$

Table 1: Boundary conditions for Case 1 to 5.

Cases 6 to 10

Grid 2 (automatically imported in template).
Physical data: $\rho = 1$, $k = 1$ and $c_p = 500$.
Inlet boundary conditions: $T_A = 20$.
Outlet boundary conditions: $\partial T/\partial n = 0$.
Wall boundary conditions not in table: $\partial T/\partial n = 0$.
Any specified wall heat flux, q_{in}, positive INTO domain!

Case	b.c.
6	$x = 0 \rightarrow T = 10$
7	$x = 0 \rightarrow T = 5$
8	$x = 0 \rightarrow q_{in} = 100\ \mathrm{W/m^2}$
9	$x = 0,\ y/H \geq 0.5 \rightarrow\ T = 10$
10	$x = 0,\ y/H \leq 0.5 \rightarrow\ T = 10$

Table 2: Boundary conditions for Case 6 to 10.

Cases 11 to 15

Grid 3 (automatically imported in template).
Physical data: $\rho = 1$, $k = 1$ and $c_p = 200$.
Inlet boundary conditions: $T_D = 10$.
Outlet boundary conditions: $\partial T/\partial n = 0$.
Wall boundary conditions not in table: $\partial T/\partial n = 0$.
Any specified wall heat flux, q_{in}, positive INTO domain!

Case	b.c.
11	$x = 0 \rightarrow T = 20;\ \ x = L \rightarrow T = 20;\ \ y = 0 \rightarrow T = 0$
12	$x = 0 \rightarrow T = 0;\ \ x = L \rightarrow T = 20;\ \ y = 0 \rightarrow T = 0$
13	$x = L \rightarrow T = 20;\ \ y = 0 \rightarrow T = 10$
14	$x = 0 \rightarrow T = 20;\ \ x = L \rightarrow T = 20$
15	$y = 0 \rightarrow q_{in} = 100\mathrm{W/m}^2$

Table 3: Boundary conditions for Case 11 to 15.

Cases 16 to 20

Grid 4 (automatically imported in template).
Physical data: $\rho = 1$, $k = 1$ and $c_p = 200$.
Inlet boundary conditions: $T_D = 10$.
Outlet boundary conditions: $\partial T/\partial n = 0$.
Wall boundary conditions not in table: $\partial T/\partial n = 0$.
Any specified wall heat flux, q_{in}, positive INTO domain!

Case	b.c.
16	$x = 0 \rightarrow T = 20;\ \ x = L \rightarrow T = 0;\ \ y = 0 \rightarrow T = 20$
17	$x = L \rightarrow T = 40;\ \ y = 0 \rightarrow T = 20$
18	$x = 0 \rightarrow T = 20;\ \ x = L \rightarrow q_{in} = -100W/m^2;\ \ y = 0 \rightarrow T = 30$
19	$x = 0 \rightarrow T = 70, x = L \rightarrow T = 0;\ \ y = 0 \rightarrow T = 30$
20	$x = 0 \rightarrow T = 40;\ \ y = 0 \rightarrow q_{in} = 50W/m^2$

Table 4: Boundary conditions for Case 16 to 20.

Cases 21 to 25

Grid 5 (automatically imported in template).
Physical data: $\rho = 1$, $k = 1$ and $c_p = 200$.
Inlet boundary conditions: $T_D = 10$.
Outlet boundary conditions: $\partial T/\partial n = 0$.
Wall boundary conditions not in table: $\partial T/\partial n = 0$.
Any specified wall heat flux, q_{in}, positive INTO domain!

Case	b.c.
21	$x = 0 \rightarrow T = 20;\ \ y = 0 \rightarrow T = 10$
22	$x = 0 \rightarrow T = 20;\ \ x = L \rightarrow T = 30;\ \ y = 0 \rightarrow T = 10$
23	$x = 0 \rightarrow T = 20;\ \ x = L \rightarrow T = 0;\ \ y = 0 \rightarrow T = 10$
24	$x = L \rightarrow T = 30;\ \ y = 0 \rightarrow T = 30$
25	$x = 0 \rightarrow T = 10;\ \ x = L \rightarrow q_{in} = 150W/m^2;\ \ y = 0 \rightarrow T = 40$

Table 5: Boundary conditions for Case 21 to 25.