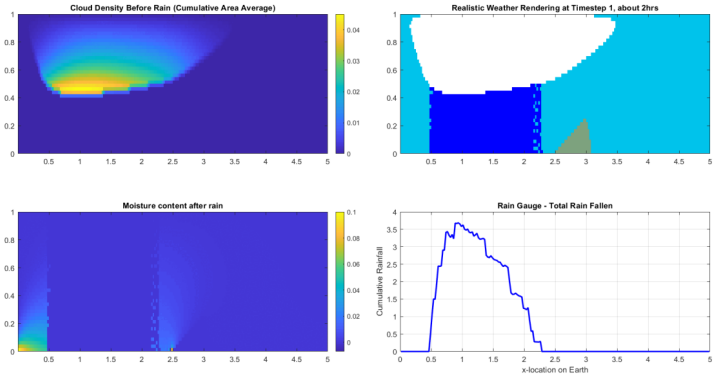


Rainshadow Simulation:

Using the SIMPLE Algorithm to Model Orographic Precipitation with CFD

Ted Yee

ME7310: Graduate Computational Fluid Dynamics with Heat Transfer

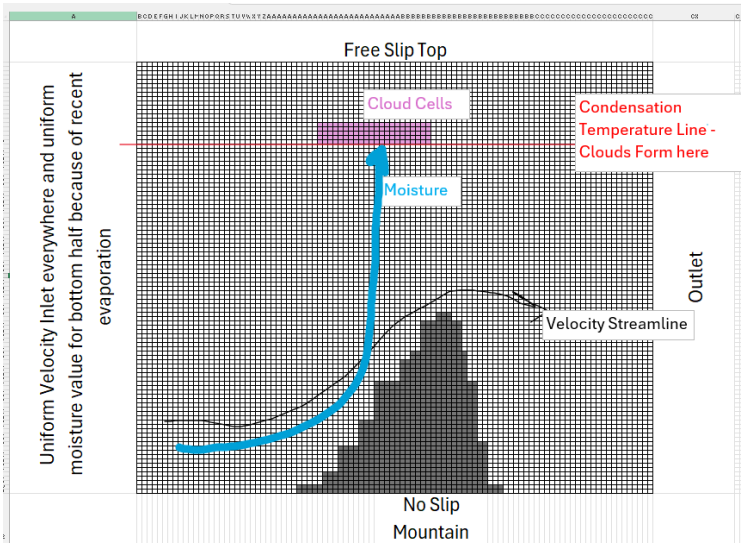
Term Project Phase 2 Report

Monday, April 21, 2025

1. PROJECT OVERVIEW

This codebase is a multi-use application for running the SIMPLE algorithm to solve finite volume approximations for simple setups. A prebuilt add-on is included that sets the parameters necessary to simulate a rainshadow (orographic precipitation) using a simplification of environmental physics and the SIMPLE Algorithm

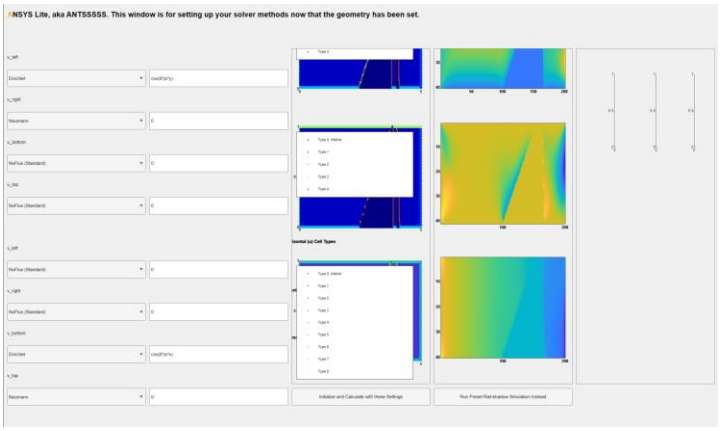
1a. Rainshadows



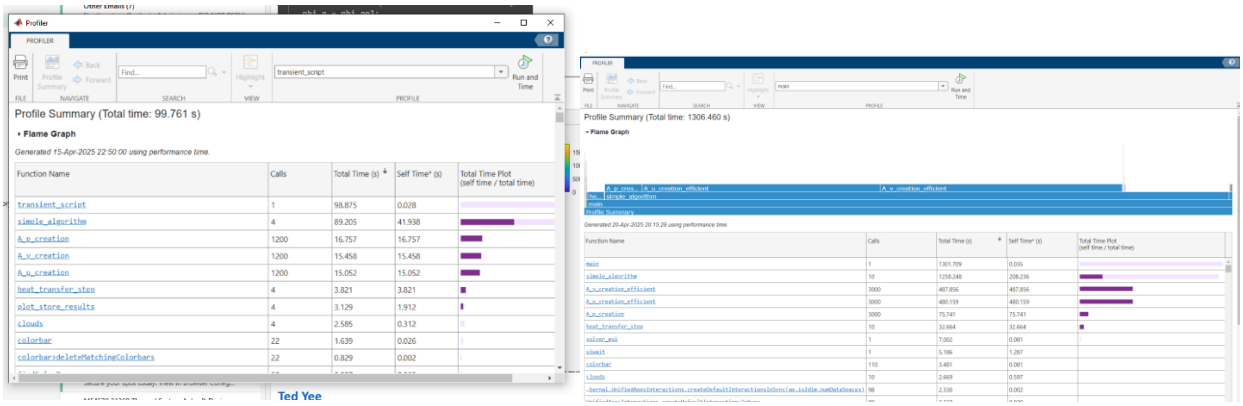
Orographic precipitation, also known as a 'rainshadow' is a phenomenon where a mountain range (usually on a western ocean coast) observes lots of rain on the upwind side, but very little in the downwind 'shadow' of the mountain. This is caused by the mountain forcing moist air upwards, where it cools and condenses. The increased condensation rate creates clouds and with enough moisture, rain removes that moisture from the air before it passes over the mountain range, hence the dry downwind shadow.

1b. General Use

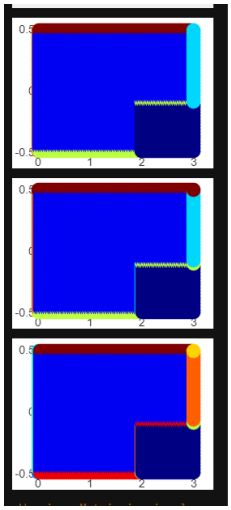
This codebase, nicknamed "ANSYS Lite, aka ANTSSSSS," is a multi-use low-level software built in MATLAB for simulating simple incompressible fluid flow. By inputting geometry and setup parameters to the system_parameters.m file, and the GUI, a variety of simple simulations can be performed. Note that no action can be taken until submit buttons are pressed. Also due to the lack of robustness as a GUI, clicking buttons more than once or going back results in MATLAB crashing.

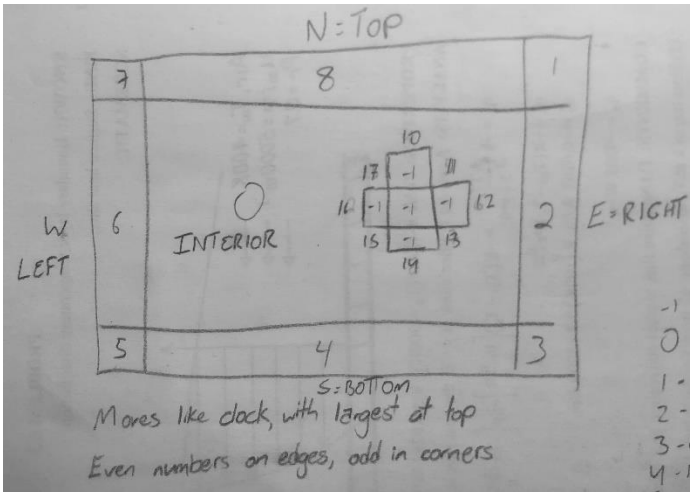


Please note that for a full transient simulation, a decent amount of time is required. Compared to the provided base code, however, this code is slow in the same areas. By timing, the base code for a blocked system, it's evident that the simple algorithm and its subparts are the main time sink for the code. The same is true for this codebase, which inspires confidence that none of the modifications are built poorly.



In order to set cell types, users have options to turn on and off a block obstruction on in the lower right corner as at right. In this codebase, all cell types are set following this key, which mimics clock enumeration as well as possible:





The borders are manually set by set_cell_type.m or set_cell_type_blocked.m, but when the preset rainshadow is selected (or the code could be modified to make it an option), a mountain is created. The function create_mountain.m take input parameters as below to create a triangle (with rounding) and automatically set border cells which are handled in the algorithm later.

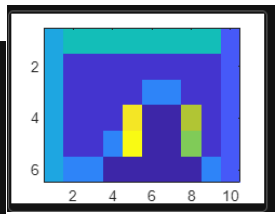
```
%% Mountain Parameters
rise_slope = .75;
fall_slope = -2;
x_mountain_start = floor(N.x_p*50/100);
x_mountain_peak = floor(N.x_p*80/100);
x_mountain_end = floor(N.x_p*120/100);
ambienthumidity = 0;
scalefactor     = 10000;                    %Default = 10000. Vertical scale factor in simulation length units to m, i.e., 1 simulation length unit is 10km

%%Atmospheric Parameters
T_surface       = 300;                      %Default = 300 Kelvin. Temperature at sea level
lapse_rate      = 9.8/1000*scalefactor;     %Default = 9.8/1000 Kelvin/meter.  Lapse rate
criticalmass    = .04;                      %Default = 3.2 for phi_wall=1,phi_in=0. Nonrealistic parameter controlling rain threshold
rainremovalrate = .75;                       %default = 0.6 under 0.5 is 100% removal,  up to 1 is 50%. it skips over cells via round(rainremovalrate*rand)

%% Set the heat transfer properties
phi_in = .04;               % how much moisture comes in with wind from  left side
phi_wall = 0;               % how much moisture comes from the top side. 0 in atmosphere
kappa = 26.3e-3/scalefactor; % 26.3e-3 for air at 300K
```

A smaller-scale model can be seen below:

```
p = 6×10
     6     8     8     8     8     8     8     8     8     2
     6     0     0     0     0     0     0     0     0     2
     6     0     0     0     0     4     4     0     0     2
     6     0     0     0    16    -1    -1    12     0     2
     6     0     0     4    17    -1    -1    11     0     2
     6     4     4    -1    -1    -1    -1    -1     4     2
```



As a general note, this code was heavily modified from the sample code used in the class and using compare features will not work because there are so many structural changes to make it modifiable. A summary of changes from the initial code is below:

- I created a structure containing the lengths of each grid called *N* which is created in mesh_generation and passed into set_cell_types.m
- I created a structure containing the coordinates of each grid called *grids* which is created in mesh_generation and passed into set_cell_types.m, guess_initialization, simple_algorithm, A_u_creation, and A_v_creation. This allows me to reference points like the x in the cos(2*pi*x) in the boundary condition of this problem. I can access any point but with less characters on screen with a structure
- I created a structure containing the initial speeds on each wall called *speeds*, which is passed of each grid called N which is created in mesh_generation and passed into simple_algorithm, A_u_creation, and A_v_creation. Instead of adding the situation-specific boundary conditions as arguments, they're always passed, again with less characters on screen
- I added an on/off toggle (0 or 1) to the main script for Cell Type visualization plotting, which controls whether or not to plot the left graph below. I also tried to label the colors, but the left 2 keep getting messed up.
- I moved to a live script since I'm more familiar with it and the slowdown of live scripts shouldn't do much since it's calling other scripts for the heavy lifting.

- I condensed the code and removed section breaks and whitespace where possible because I would personally rather see it all at once than see it broken up

1c. Preset Rainshadow Simulation

When a user clicks the "Use preset rainshadow simulation button,' the parameters are automatically set to create a simulation of a transient rainshadow. The difference between this preset option and the user-input parameters is that there are no block obstructions and that a triangular 'mountain' is created with correct boundary cells. During the time-marching loop, the final iteration of the SIMPLE algorithm and heat_transfer_setp.m are passed in the form of phi_np1 to a new script called clouds.m. phi_np1, crucially, is NOT a value of temperature in this simulation. It is a measure of moisture content, which diffuses and advects throughout the domain. clouds.m calculates the partial pressure of the water in the air at every point, and the pressure of a saturated vapor at every point, which is based off of temperature. Temperature in the domain is based purely off of lapse rate (temperature decrease with altitude) which also depends on a global length scaling factor.

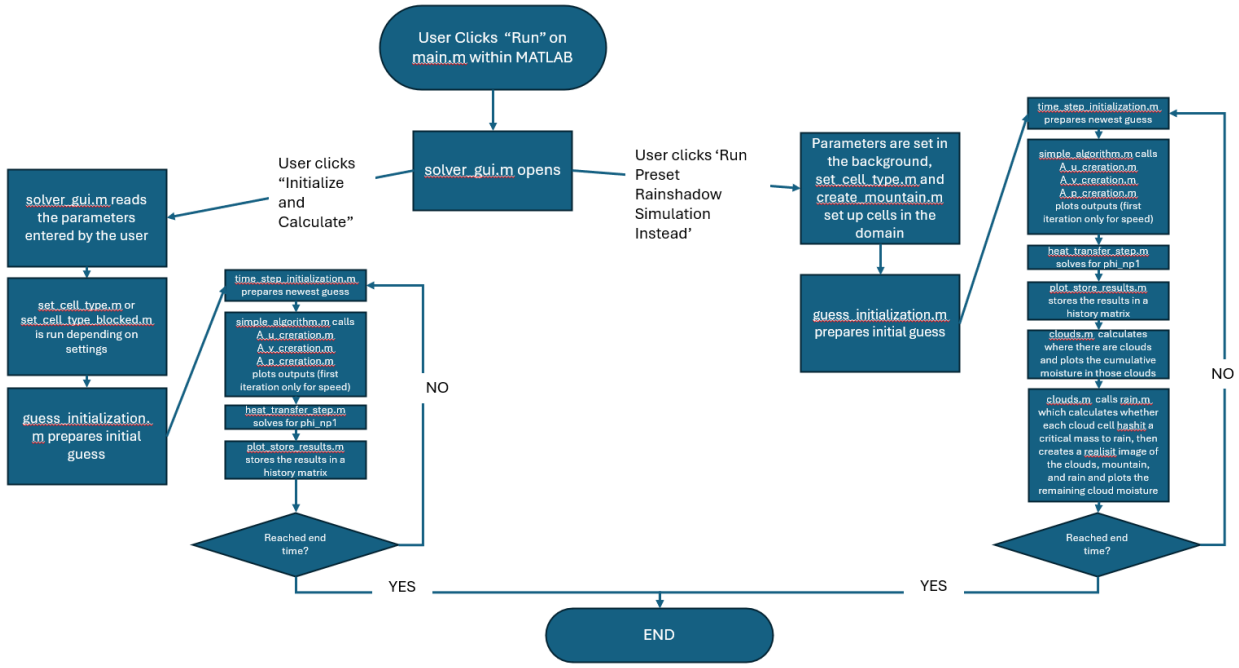
By default, this scaling factor is 10,000, meaning that a length unit of 1 in the simulation is equal to 10,000m or 10km, which is the ceiling at which lapse rate is no longer linear. Also important to note is that the time factor combined with the preset wind speed of 10 results in 100,000 meters per simulation time unit, or that for an average wind speed of 10km/hr, one time unit in this simulation is on the order of 10 hours in real life.

The function clouds.m then compares partial pressure of the water present with the pressure to reach saturated vapor. If the present partial pressure is larger, the cell is said to condense and form a 'cloud cell.' By calculating the cumulative moisture in an area (each cell depends on its neighbors, except at the borders, where it counts) rain.m, compares each value int eh cumulative_moisture_map matrix to a threshold or critical mass. Via trial and error, this threshold is set to 0.04. Units on this number would be "cumulative moisture points with neighbors." If the threshold is reached, there is said to be rain in the area and throughout the column, moisture (phi_np1) values are cleared. In order to simulate more realistic conditions, a rain_removal_rate parameter was created. This parameter scales so that values under 0.5 result in complete removal of moisture, and a value of 1 removes half the moisture in the column. It removes moisture randomly, skipping over cells via round(rainremovalrate*rand). By default, this value is 0.6, which removes most, but not all of the moisture in columns where it rains.

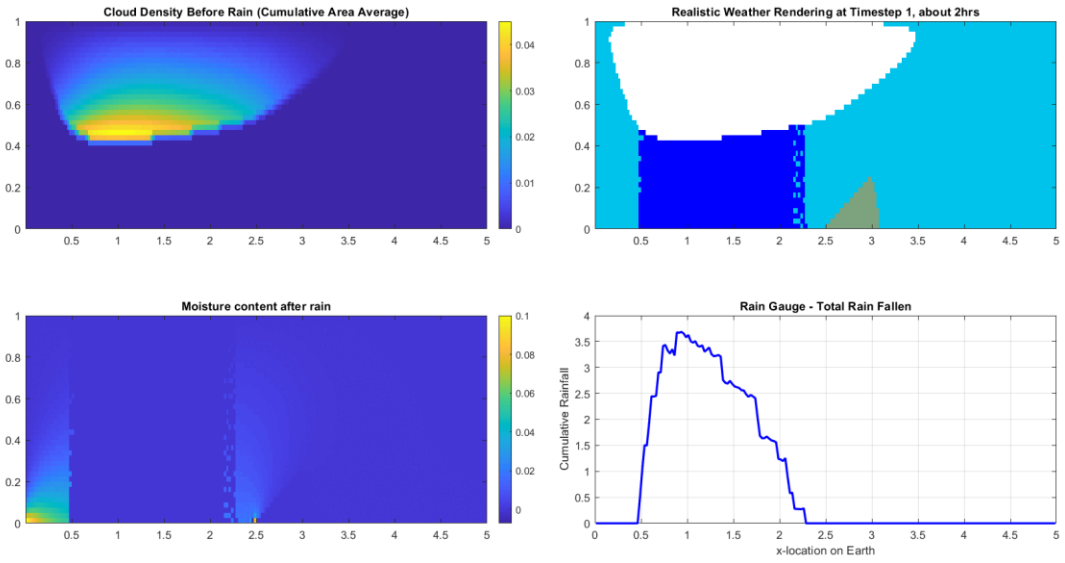
Once the calculations are complete, a realistic rendering of the sky, mountain, clouds, and rain is drawn at each time step. The updated value of phi_np1 (which is only changed where the rain has occurred) is then available for the next time step, and the loop repeats until complete.

As for the accuracy of the model, there are obviously many simplifications, not the least of which is that the SIMPLE algorithm models incompressible flow, which is definitely not the case in stratified mediums with temperature effects such as the atmosphere. I do want to thank Professor Daniel Douglass of the Northeastern University Marine and Environmental Sciences Department for his input on the modelling. While he personally has no experience in modelling and did not have time to read my proposal, he independently came up with a very similar modelling setup, which contributes confidence to the realism of the model.

2. CODE SKETCH



3. EXECUTION AND ANALYSIS



3a. Results

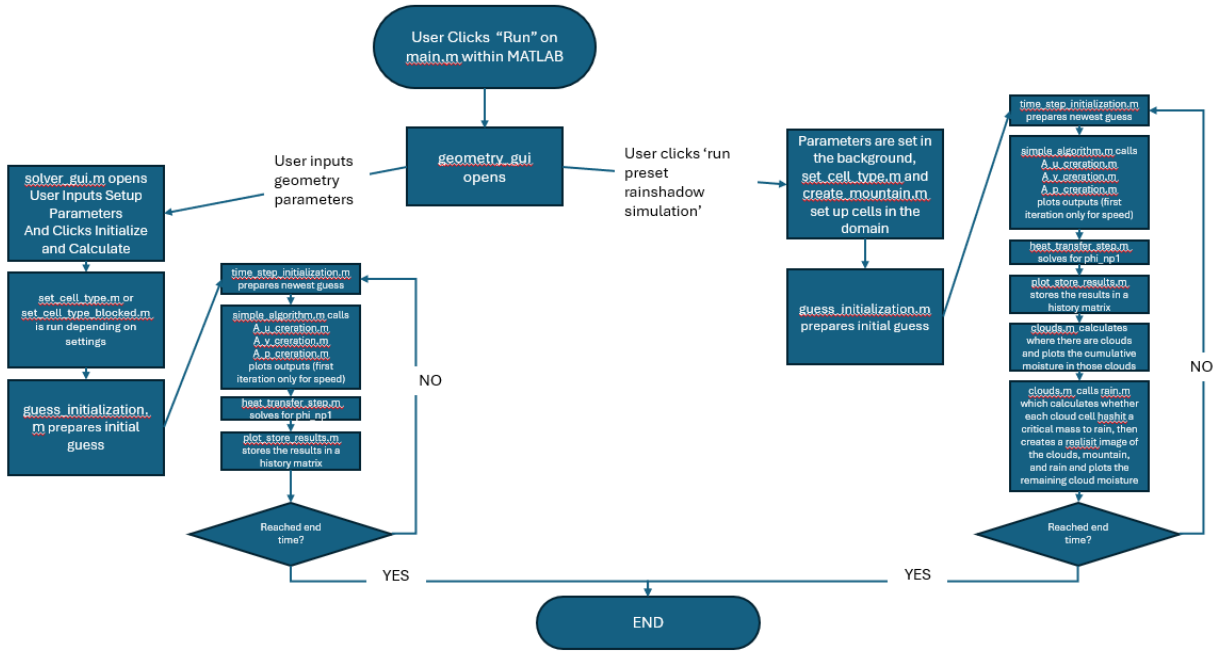
As can be seen in the attached .GIF and .PNG files as well as the graphics in the appendices, using the preset rainshadow settings successfully creates a pattern of rainfall that matches that of a rainshadow. The rain gauge shows a large amount of cumulative rain upwind of the mountain and none downwind. The visualization graphics show the cloud becoming less dense (as moisture is reduced by the rainfall) and moving slowly over the mountain. While the accuracy of any simplified atmospheric model can be put into doubt, this simulation successfully completed its mission: to simulate a rainfall pattern of more rain upwind of an obstruction and less ran downwind.

3b. Further Improvements

The most obvious improvement that could be made to the model would be to use an incompressible, density based solver for the air conditions. This simulation was limited by the extreme demand on the computers it seems to generate. Averaging 2 minutes per time step, mostly consumed in the iterations of the simple_algorithm.m script, getting just 10 frames (about 20hours of real-life time) takes almost twenty minutes to calculate. Improvements in the resolution (which is now just 40X200 square cells (each of the 8000 cells is 250m long on each edge) could yield more detailed results, but at a significant increase in calculations. As this model was a representative setup, it would be prudent to begin again with a new fluids solver to retrieve more accurate data.

Another further improvement that was considered, but deemed unnecessary was the incorporation of another GUI window where the user could set up the geometry and solver

parameters without having to enter the code. Currently all parameters are sufficiently set by system_parameters.m, so it was left out. Were it to be finished the code sketch would look like below:



3b. Grading

I have a uniform, structured grid with tunable mountain ranges/preset shapes (+20), a variety of boundary conditions (+10), and a transient simulation (+20) of a rainshadow (+30), which overall should hit at least 100 points.

4. APPENDIX

Please see attached codebase. All physical parameters either taken from communications with Prof. Daniel Douglass or are non-physical representations derived from trial and error.

Appendix A: Term Project Instructions

ME 7310: Computational Fluid Dynamics with Heat Transfer

Spring 2025

Project Description

Objective: The goal of this project is to teach you how the finite volume method can be used to solve the Navier-Stokes equations by coding this yourself. You will be writing your own solver to model a unique problem.

Phase 1: The first phase of this project is to develop a finite volume code from scratch which solves the Navier-Stokes equations for an incompressible fluid. This code should read in an input script containing the domain mesh, boundary and initial conditions, the time steps, and output the velocity and pressure at the mesh points.

Deliverable 1 (Due 4/1/25): The first deliverable is a brief proposal of the problem you will be solving. This should include:

- What physical problem you are trying to model
- The domain and mesh style you will be using
- The boundary conditions of the system
- The type of additional analysis (e.g., parameter variation) you will perform
- A preliminary analysis of the problem using Fluent (if applicable).

No grade will be assessed at this point. This is simply a check that you are attempting a reasonable problem. If nothing is turned in, points will be deducted from the overall score. Please approach me beforehand if you need help selecting a problem.

Phase 2: You will develop a finite volume code that models the system you proposed. There will be template Navier-Stokes solvers that you will be able to adapt to your particular problem. This portion of the project is to be done independently. You will have opportunities to debug your code with the professor during designated office hours.

Deliverable 2 (Due: 4/22/25): You will need to turn in a report containing:

- A sketch of your code
- Evidence of the verification study compared to results from Fluent (if applicable)
- Execution and the results of the proposed analysis
- Full code library in "operating" condition.

Phase 3: To fully grasp the finite volume approach and to encourage robust/efficient code, you will be asked to modify your input file during the final exam window. You will have a number of options for code modification that you will need to perform in addition to a written exam.

Deliverable 3 (Due: 4/24/25): You will need to turn in a report containing:

- A description of what changes were made and how you implemented them in your code
- Figures showing velocity and pressure fields before and after the changes
- Your solutions to the written exam

Anticipated grading scheme: The following breakdown is open to change up to the proposal deadline, and any changes will be detailed in class. Projects will be graded based on 100 points.
Requirements (50 points):

- Proposal of what the project is and what features will be incorporated. (5 points)
- A functioning finite volume code that solves the proposed Navier-Stokes problem. (20 points)
- An additional analysis of the problem (i.e. flux through different portions of the domain, shear on different surfaces, total force on an internal obstruction etc.) (10 points)
- Written report with figures (15 points)

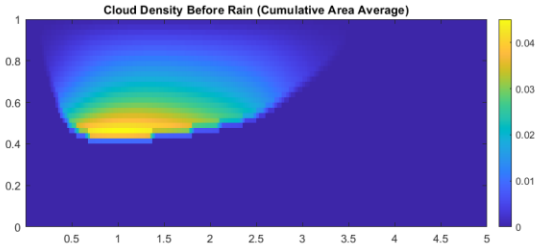
Minimum Requirements: The project will have a number of properties that add complexity. The simplest project that you can submit is a steady state simulation of flow through a constant height channel using a uniform structured mesh. This project receives no complexity points.
Complexity points: As you increase the complexity of the simulation, it is possible to earn more points. The categories for complexity are as follows:

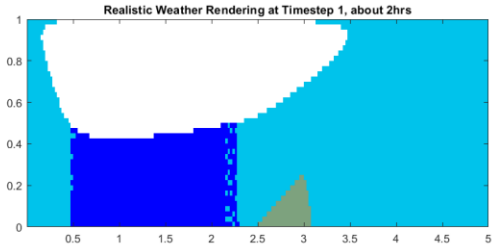
- **Mesh type**: Structured uniform mesh (baseline), Structured non-uniform mesh (+20), multigrid method (+30), or unstructured mesh (+40)
- **Geometry**: Single feature (baseline), multiple fixed features (+10), tunable features (+20, requires running multiple simulations), immersed boundary method (+30)
- **Boundary conditions**: Inlet/outlet boundary conditions (baseline), one of the following: wall, no slip, periodic, symmetry (baseline), two of the following: wall, no slip, periodic, symmetry (+10).
- **Temporal**: Steady state (baseline) or transient (+20)
- **Heat transfer:** No heat transfer (baseline) or heat transfer component (+20)
- **Ansys:** If you have access to Fluent, you can perform an analysis and comparison with your Matlab results (+10)

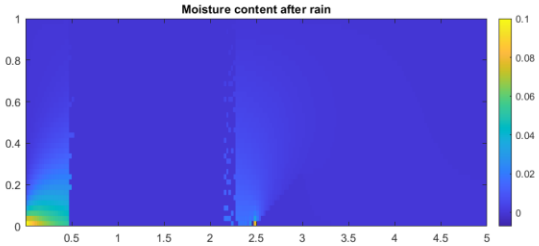
Examples:

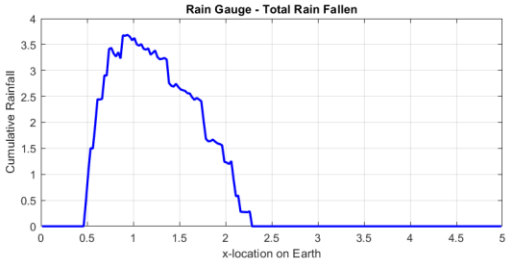
- Unstructured mesh with single feature, inlet/outlet/wall/free slip conditions, steady state. (+50, maximum score of 100/100)
- Structured mesh with non-uniform grid, with tunable geometric features, all types of boundary conditions, steady state. (+50, maximum score of 100/100)
- Structured mesh with non-uniform grid, with at least one geometric feature, multiple boundary conditions, transient or heat transfer incorporation. (+50, maximum score of 100/100)

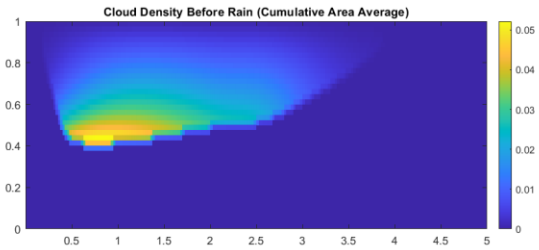
Appendix B: Simulation Graphics

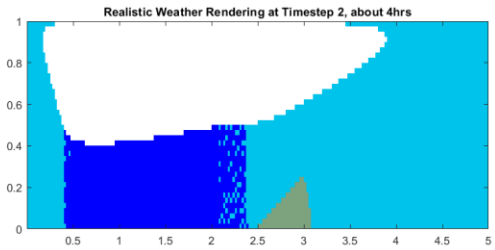


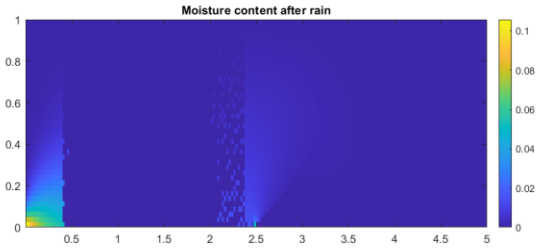


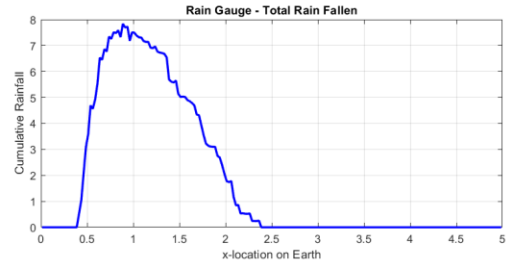


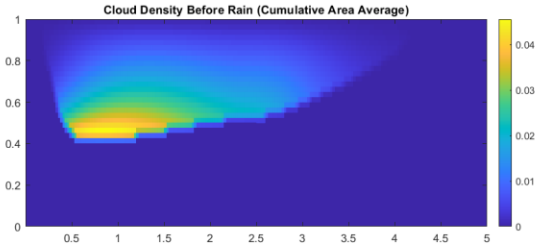


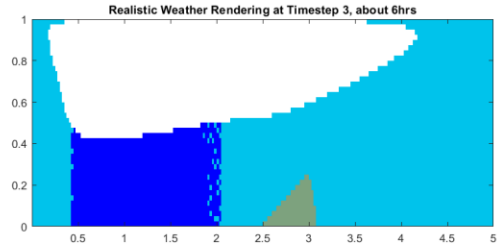


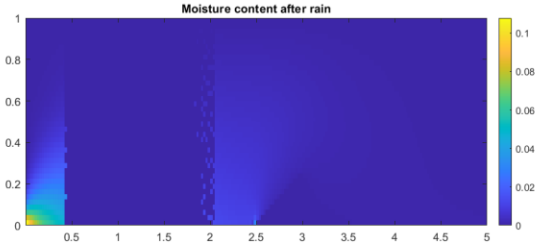


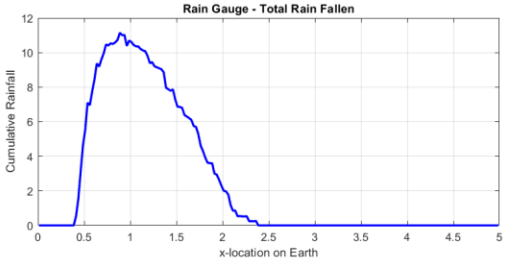


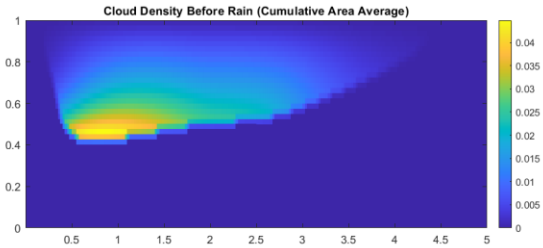


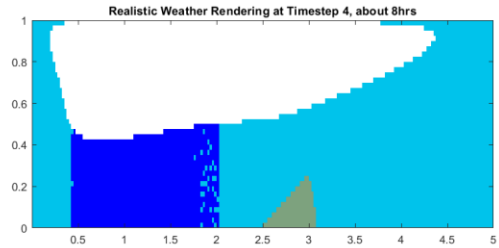

Cloud Density Before Rain (Cumulative Area Average)

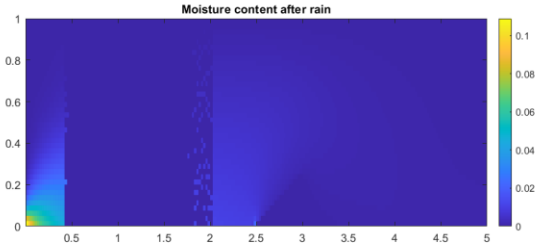

Realistic Weather Rendering at Timestep 3, about 6hrs

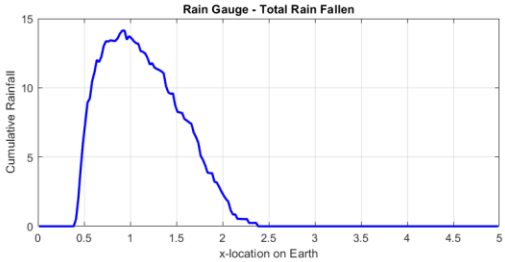

Moisture content after rain

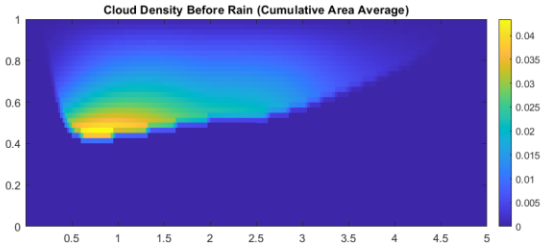

Rain Gauge - Total Rain Fallen
Cumulative Rainfall
x-location on Earth

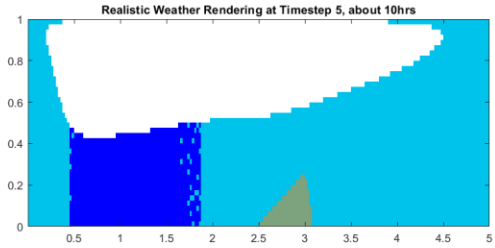

Cloud Density Before Rain (Cumulative Area Average)

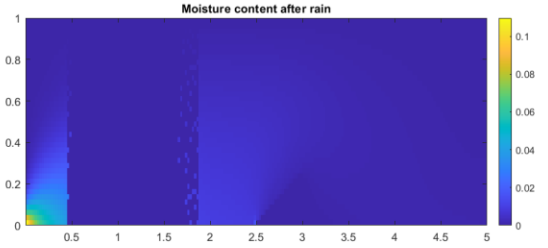

Realistic Weather Rendering at Timestep 4, about 8hrs

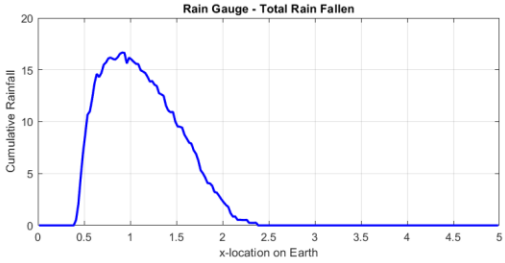

Moisture content after rain

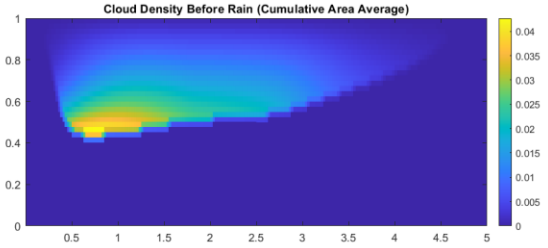

Rain Gauge - Total Rain Fallen
Cumulative Rainfall
x-location on Earth

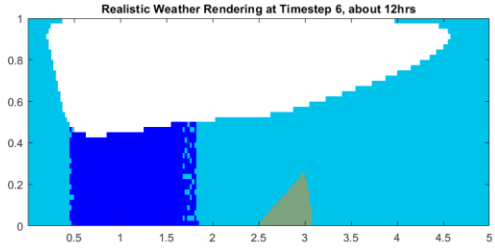

Cloud Density Before Rain (Cumulative Area Average)

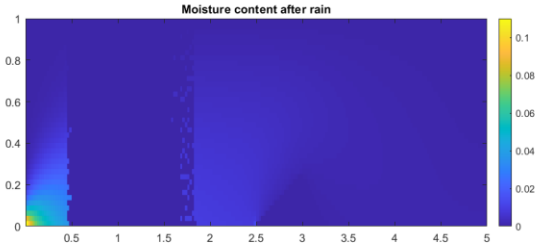

Realistic Weather Rendering at Timestep 5, about 10hrs

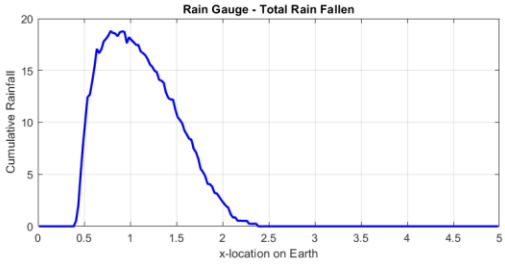

Moisture content after rain

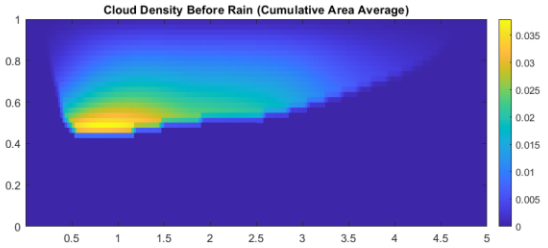

Rain Gauge - Total Rain Fallen
Cumulative Rainfall
x-location on Earth

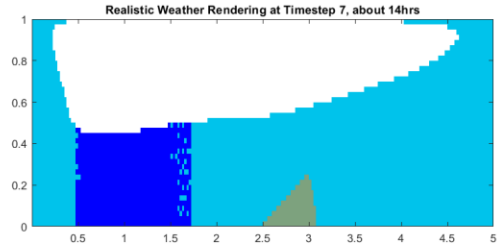

Cloud Density Before Rain (Cumulative Area Average)

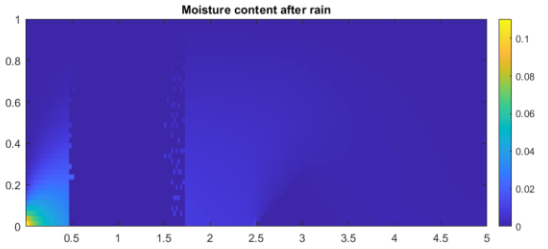

Realistic Weather Rendering at Timestep 6, about 12hrs

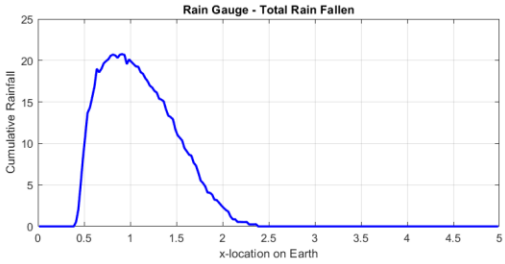

Moisture content after rain

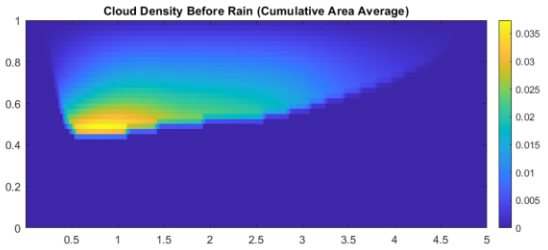

Rain Gauge - Total Rain Fallen
Cumulative Rainfall
x-location on Earth

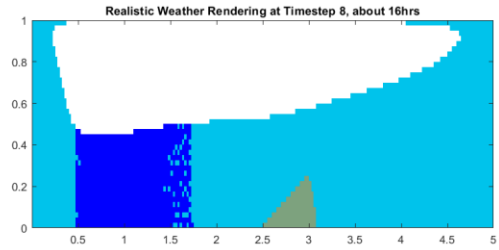

Cloud Density Before Rain (Cumulative Area Average)

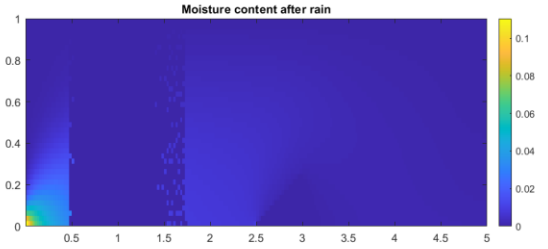

Realistic Weather Rendering at Timestep 7, about 14hrs

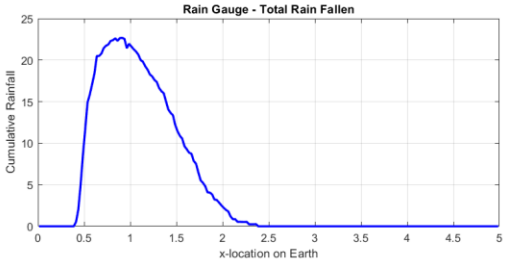

Moisture content after rain

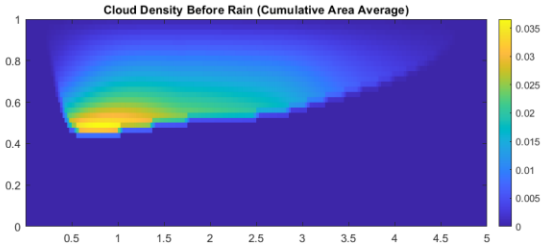

Rain Gauge - Total Rain Fallen
Cumulative Rainfall
x-location on Earth

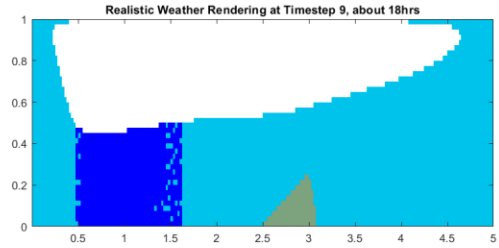

Cloud Density Before Rain (Cumulative Area Average)

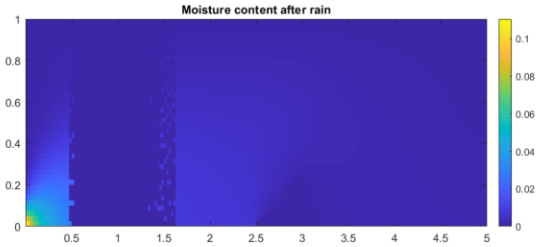

Realistic Weather Rendering at Timestep 8, about 16hrs

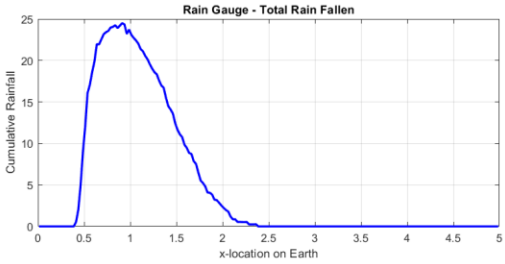

Moisture content after rain

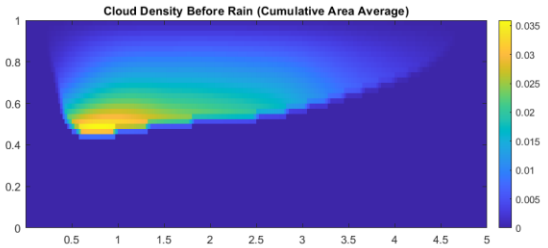

Rain Gauge - Total Rain Fallen
Cumulative Rainfall
x-location on Earth

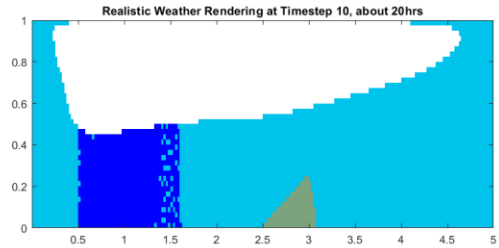

Cloud Density Before Rain (Cumulative Area Average)

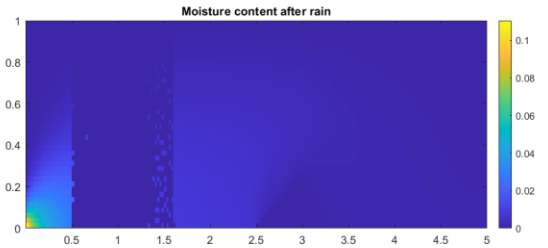

Realistic Weather Rendering at Timestep 9, about 18hrs

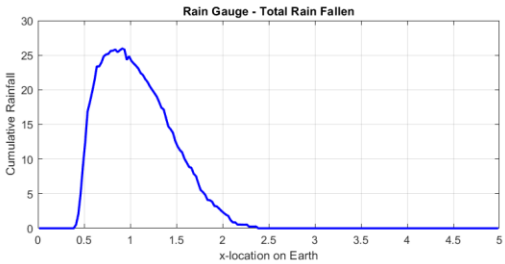

Moisture content after rain


Rain Gauge - Total Rain Fallen
Cumulative Rainfall
x-location on Earth


Cloud Density Before Rain (Cumulative Area Average)


Realistic Weather Rendering at Timestep 10, about 20hrs


Moisture content after rain


Rain Gauge - Total Rain Fallen
Cumulative Rainfall
x-location on Earth