Stellar convection

Term project 3

The third term project involves using chapters 5 and 6 from the lecture notes to model convection in a star. This time we model convection in 2 dimensions and we move beyond the classic approach of treating convection in 1D like in chapter 5 and project 2. To achieve that, this project will entail writing a 2D hydrodynamics code, implementing the continuity equation with no sources or sinks (1), the momentum equation for both x and y, excluding the viscous stress tensor but including gravity (2)[1], and the energy equation (3).

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{1}$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = -\nabla P + \rho \mathbf{g} \tag{2}$$

$$\frac{\partial e}{\partial t} + \nabla \cdot (e \mathbf{u}) = -P \nabla \cdot \mathbf{u} \tag{3}$$

In order to solve this project, you need to write a code that solves the above equations using an *explicit* numerical scheme. We will take a closer look at what this means in the next section. The code should be written as modular as possible, and the python script `skeleton.py` is available at the course web page. Use this skeleton script for creating your hydrodynamics solver.

[1]Note that the vector product $\otimes$ in the momentum equation is not a dot product, but an *outer product*.

General theory

Hydrodynamics is the science of fluids. In this case it also includes gasses and plasma with short enough collisional mean free path. To model such a fluid, we generally split a volume into cells where each cell carries only one value for the used variables (ρ, $\mathbf{u}$, T etc.). In this case, we are dealing with a fluid which moves, and so we need to include the development in time while treating the whole volume (the collection of cells). To make everything as simple as possible, we will split our volume into identical cubic cells (think LEGO$^{\text{TM}}$ bricks) that each have a size Δx and Δy. The computational volume becomes $N_x \times \Delta x$ and $N_y \times \Delta y$, where N_x, N_y are the number of cells in each direction. The total number of cells are then $N_x \times N_y$, and they are each located at $(x, y) = (i\Delta x, j\Delta y)$ where $i \in [0, N_x - 1]$ and $j \in [0, N_y - 1]$.

In order to solve the hydrodynamic equations on such a computational grid, we need to approximate the derivatives in equations (1)–(3) using finite difference methods. These methods are explicit, meaning that we use parameters at the present time n to calculate the parameter at time $n + 1$. We show this by introducing the Forward Time Centred Space (FTCS) numerical scheme. Consider the two-dimensional advection equation (1)

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \mathbf{u}) = -\mathbf{u}\nabla\rho = -u\frac{\partial \rho}{\partial x} - w\frac{\partial \rho}{\partial y} \tag{4}$$

where the flow $\mathbf{u} = (u, w)$[2] is assumed to be constant. Constant flow means that the $\frac{\partial u}{\partial x}$ and $\frac{\partial w}{\partial y}$ terms equal to zero. We divide space and time into discrete positions and instants

$$x_i = x_0 + i\Delta x \qquad y_j = y_0 + j\Delta y \qquad t_n = t_0 + n\Delta t \tag{5}$$

and define

$$\rho^n_{i,j} \equiv \rho(x_i, y_j, t_n) \tag{6}$$

[2]Note that we here use u for the velocity along the x direction and w for the y direction. In the lecture notes, we use u_x and u_y.

The derivatives are then discretised into

$$\left[\frac{\partial \rho}{\partial t}\right]_{i,j}^{n} \approx \frac{\rho_{i,j}^{n+1} - \rho_{i,j}^{n}}{\Delta t} \tag{7}$$

$$\left[\frac{\partial \rho}{\partial x}\right]_{i,j}^{n} \approx \frac{\rho_{i+1,j}^{n} - \rho_{i-1,j}^{n}}{2\Delta x} \tag{8}$$

$$\left[\frac{\partial \rho}{\partial y}\right]_{i,j}^{n} \approx \frac{\rho_{i,j+1}^{n} - \rho_{i,j-1}^{n}}{2\Delta y} \tag{9}$$

The FTCS numerical scheme can now be written as

$$\frac{\rho_{i,j}^{n+1} - \rho_{i,j}^{n}}{\Delta t} = -u_{i,j}^{n}\left(\frac{\rho_{i+1,j}^{n} - \rho_{i-1,j}^{n}}{2\Delta x}\right) - w_{i,j}^{n}\left(\frac{\rho_{i,j+1}^{n} - \rho_{i,j-1}^{n}}{2\Delta y}\right) \tag{10}$$

Note that *forward differencing* means taking the difference with the next instance or position (therefore *Forward Time* in FTCS) and *central differencing* means taking the difference between neighbouring cells (therefore *Centred Space* in FTCS).

The FTCS algorithm is simple and easy to implement, but a severe drawback is that it is unconditionally unstable. Therefore, one must consider other numerical schemes in order to avoid numerical errors. Upwind differencing can be used to model the transport properties of the system better. The first order upwind scheme considers the direction of the flow (u, w), where the spatial derivatives are discretised as

$$\left[\frac{\partial \rho}{\partial x}\right]_{i,j}^{n} \approx \begin{cases} \frac{\rho_{i,j}^{n} - \rho_{i-1,j}^{n}}{\Delta x} & \text{if} \quad u_{i,j}^{n} \geq 0 \\ \frac{\rho_{i+1,j}^{n} - \rho_{i,j}^{n}}{\Delta x} & \text{if} \quad u_{i,j}^{n} < 0 \end{cases} \tag{11}$$

$$\left[\frac{\partial \rho}{\partial y}\right]_{i,j}^{n} \approx \begin{cases} \frac{\rho_{i,j}^{n} - \rho_{i,j-1}^{n}}{\Delta y} & \text{if} \quad w_{i,j}^{n} \geq 0 \\ \frac{\rho_{i,j+1}^{n} - \rho_{i,j}^{n}}{\Delta y} & \text{if} \quad w_{i,j}^{n} < 0 \end{cases} \tag{12}$$

For a system of constant, positive flow (u, w), the first order upwind scheme is written as

$$\frac{\rho_{i,j}^{n+1} - \rho_{i,j}^{n}}{\Delta t} = -u_{i,j}^{n}\left(\frac{\rho_{i,j}^{n} - \rho_{i-1,j}^{n}}{\Delta x}\right) - w_{i,j}^{n}\left(\frac{\rho_{i,j}^{n} - \rho_{i,j-1}^{n}}{\Delta y}\right) \tag{13}$$

In this project you will discretise the hydrodynamic equations using both numerical algorithms presented in this section.

Algorithm

The 2D hydrodynamical equations have to be solved in a specific way in order to obtain stability in the numerical solution. We leave the implementation on the different methods to you, but will in this section provide you with the algorithms. First, we write the hydrodynamic equations by splitting them into components in x and y. The continuity, momentum and energy equations are then written as

$$\frac{\partial \rho}{\partial t} = -\frac{\partial \rho u}{\partial x} - \frac{\partial \rho w}{\partial y} \tag{14}$$

$$\frac{\partial \rho u}{\partial t} = -\frac{\partial \rho u^2}{\partial x} - \frac{\partial \rho u w}{\partial y} - \frac{\partial P}{\partial x} \tag{15}$$

$$\frac{\partial \rho w}{\partial t} = -\frac{\partial \rho w^2}{\partial y} - \frac{\partial \rho u w}{\partial x} - \frac{\partial P}{\partial y} + \rho g_y \tag{16}$$

$$\frac{\partial e}{\partial t} = -\frac{\partial e u}{\partial x} - \frac{\partial e w}{\partial y} - P\left(\frac{\partial u}{\partial x} + \frac{\partial w}{\partial y}\right) \tag{17}$$

where (15) and (16) are the horizontal and vertical momentum equations, respectively. Each partial derivative has to be discretised in a specific way in order to solve the equations numerically. Common for all the equations is that the left-hand side is discretised using forward time.

Continuity equation

In this project, the flow (u, w) is not assumed to be constant. Therefore, the continuity equation in (14) is split into additional terms

$$\frac{\partial \rho}{\partial t} = -\rho\left(\frac{\partial u}{\partial x} + \frac{\partial w}{\partial y}\right) - u\frac{\partial \rho}{\partial x} - w\frac{\partial \rho}{\partial y} \tag{18}$$

The spatial derivatives of u and w are approximated using a central scheme, while the spatial derivatives of ρ are approximated using upwind differencing. In order to avoid long expressions (which often lead to errors), the calculation is split into two parts. First, the left-hand side is kept as it is (not discretised), while the right-hand side is calculated as

$$\left[\frac{\partial \rho}{\partial t}\right]_{i,j}^{n} = -\rho_{i,j}^{n}\left(\left[\frac{\partial u}{\partial x}\right]_{i,j}^{n} + \left[\frac{\partial w}{\partial y}\right]_{i,j}^{n}\right) - u_{i,j}^{n}\left[\frac{\partial \rho}{\partial x}\right]_{i,j}^{n} - w_{i,j}^{n}\left[\frac{\partial \rho}{\partial y}\right]_{i,j}^{n} \tag{19}$$

where

$$\begin{aligned}
\left[\frac{\partial u}{\partial x}\right]_{i,j}^{n} &\approx \frac{u_{i+1,j}^{n} - u_{i-1,j}^{n}}{2\Delta x} \\
\left[\frac{\partial w}{\partial y}\right]_{i,j}^{n} &\approx \frac{w_{i,j+1}^{n} - w_{i,j-1}^{n}}{2\Delta y} \\
\left[\frac{\partial \rho}{\partial x}\right]_{i,j}^{n} &\approx \begin{cases} \frac{\rho_{i,j}^{n} - \rho_{i-1,j}^{n}}{\Delta x} & \text{if} \quad u_{i,j}^{n} \geq 0 \\ \frac{\rho_{i+1,j}^{n} - \rho_{i,j}^{n}}{\Delta x} & \text{if} \quad u_{i,j}^{n} < 0 \end{cases} \\
\left[\frac{\partial \rho}{\partial y}\right]_{i,j}^{n} &\approx \begin{cases} \frac{\rho_{i,j}^{n} - \rho_{i,j-1}^{n}}{\Delta y} & \text{if} \quad w_{i,j}^{n} \geq 0 \\ \frac{\rho_{i,j+1}^{n} - \rho_{i,j}^{n}}{\Delta y} & \text{if} \quad w_{i,j}^{n} < 0 \end{cases}
\end{aligned}$$

Then, the primary variable is advanced in time by rewriting (7) as

$$\rho_{i,j}^{n+1} = \rho_{i,j}^{n} + \left[\frac{\partial \rho}{\partial t}\right]_{i,j}^{n} \Delta t \tag{20}$$

and then for $\left[\frac{\partial \rho}{\partial t}\right]_{i,j}^{n}$ use the right-hand side of (19).

Note that in this equation with the time derivative of ρ on the left-hand side, it is the terms with spatial derivatives of ρ that needs to be treated with the upwind scheme in the direction of the derivative. Which neighboring point is used depends on the sign of the prefactor to the spatial derivative, e.g. $u_{i,j}^{n}$ for $\left[\frac{\partial \rho}{\partial x}\right]_{i,j}^{n}$ in Eq. (19). The first two terms on the right-hand side of Eq. (19), on the other hand, can be treated with central space as in FTCS.

Momentum equation

The horizontal momentum equation in (15) is split into additional terms

$$\frac{\partial \rho u}{\partial t} = -\rho u \left(\frac{\partial u}{\partial x} + \frac{\partial w}{\partial y} \right) - u \frac{\partial \rho u}{\partial x} - w \frac{\partial \rho u}{\partial y} - \frac{\partial P}{\partial x} \tag{21}$$

which is further expressed as

$$\begin{aligned}
\left[\frac{\partial \rho u}{\partial t}\right]_{i,j}^{n} = &- [\rho u]_{i,j}^{n} \left(\left[\frac{\partial u}{\partial x}\right]_{i,j}^{n} + \left[\frac{\partial w}{\partial y}\right]_{i,j}^{n} \right) \\
&- u_{i,j}^{n} \left[\frac{\partial \rho u}{\partial x}\right]_{i,j}^{n} - w_{i,j}^{n} \left[\frac{\partial \rho u}{\partial y}\right]_{i,j}^{n} - \left[\frac{\partial P}{\partial x}\right]_{i,j}^{n}
\end{aligned} \tag{22}$$

All spatial derivatives are approximated using upwind differencing, except for the vertical velocity gradient and the pressure gradient which are approximated using central differencing. The various spatial terms are discretised as

$$\left[\frac{\partial u}{\partial x}\right]_{i,j}^{n} \approx \begin{cases} \frac{u_{i,j}^n - u_{i-1,j}^n}{\Delta x} & \text{if} \quad u_{i,j}^n \geq 0 \\ \frac{u_{i+1,j}^n - u_{i,j}^n}{\Delta x} & \text{if} \quad u_{i,j}^n < 0 \end{cases}$$

$$\left[\frac{\partial w}{\partial y}\right]_{i,j}^{n} \approx \frac{w_{i,j+1}^n - w_{i,j-1}^n}{2\Delta y}$$

$$\left[\frac{\partial \rho u}{\partial x}\right]_{i,j}^{n} \approx \begin{cases} \frac{[\rho u]_{i,j}^n - [\rho u]_{i-1,j}^n}{\Delta x} & \text{if} \quad u_{i,j}^n \geq 0 \\ \frac{[\rho u]_{i+1,j}^n - [\rho u]_{i,j}^n}{\Delta x} & \text{if} \quad u_{i,j}^n < 0 \end{cases}$$

$$\left[\frac{\partial \rho u}{\partial y}\right]_{i,j}^{n} \approx \begin{cases} \frac{[\rho u]_{i,j}^n - [\rho u]_{i,j-1}^n}{\Delta y} & \text{if} \quad w_{i,j}^n \geq 0 \\ \frac{[\rho u]_{i,j+1}^n - [\rho u]_{i,j}^n}{\Delta y} & \text{if} \quad w_{i,j}^n < 0 \end{cases}$$

$$\left[\frac{\partial P}{\partial x}\right]_{i,j}^{n} \approx \frac{P_{i+1,j}^n - P_{i-1,j}^n}{2\Delta x}$$

The left-hand side of (22) is discretised using forward time

$$\left[\frac{\partial \rho u}{\partial t}\right]_{i,j}^{n} \approx \frac{[\rho u]_{i,j}^{n+1} - [\rho u]_{i,j}^{n}}{\Delta t} \tag{23}$$

Since $\rho_{i,j}^{n+1}$ has already been calculated, we only concern ourselves with advancing u in time. Rewriting (23) gives

$$u_{i,j}^{n+1} = \frac{[\rho u]_{i,j}^{n} + \left[\frac{\partial \rho u}{\partial t}\right]_{i,j}^{n} \Delta t}{\rho_{i,j}^{n+1}} \tag{24}$$

Based on the information given in this subsection, you can now discretise and calculate the vertical component of the momentum equation.

Energy equation

To correctly simulate convection, it is necessary to include an energy equation. The energy we are interested in is the internal energy of the gas, and it is in units of energy per volume. The previous subsections show how to discretise and find numerical algorithms to solve the continuity equation and the horizontal component of the momentum equation. Use the methods described in these subsection to find an algorithm for solving the energy equation in (17). Keep in mind that the flow (u, w) is non-constant, hence some of the spatial terms need to be split into additional terms.

Time step length

After the time derivatives of the primary variables (ρ, u, w and e) have been calculated, they can be used to determine the optimal time step length. This length is found by insisting that none of the primary variables get to change by more than a given percentage. For each primary variable ϕ, calculate the relative change $\Delta\phi/\phi$ per time step Δt:

$$\text{rel}(\phi) = \frac{\Delta\phi}{\phi} \cdot \frac{1}{\Delta t} = \left|\frac{\partial\phi}{\partial t} \cdot \frac{1}{\phi}\right| \tag{25}$$

Another requirement is that a fluid particle doesn't move so fast that our time step will move it past a whole grid point. This will be satisfied by calculating the relative change of position as well:

$$\begin{aligned}\text{rel}(x) &= \left|\frac{\partial x}{\partial t} \cdot \frac{1}{\Delta x}\right| = \left|\frac{u}{\Delta x}\right| & (26)\\ \text{rel}(y) &= \left|\frac{\partial y}{\partial t} \cdot \frac{1}{\Delta y}\right| = \left|\frac{w}{\Delta y}\right| & (27)\end{aligned}$$

(Here the change is calculated relative to the grid element size.) The above quantities are calculated for every grid point. We must use their maximum values on the grid to make sure that all grid points satisfy our condition.

We are then left with one value, max(rel(ϕ)), for each primary variable ϕ (as well as for position) describing the largest relative change on the grid for that variable (per time step length unit). To make sure that all the quantities satisfy our condition, we then choose the largest of these values, δ =max(max(rel(ϕ))). We have thus determined the largest relative change per time step for any of the quantities at any point on the grid.

The time step length Δt is then found by insisting that the maximum relative change during the time step has to be equal to some small number, that is

$$\delta \cdot \Delta t = p \quad \Rightarrow \quad \Delta t = \frac{p}{\delta} \tag{28}$$

where p is typically around 0.1. This requirement is called the Courant-Friedrichs-Lewy (CFL) condition, and p depends on the order of your numerical method. Higher order methods can usually bring p up to higher values. Since the velocity fields typically will have stationary points, these must be excluded from the calculation of relative change in (25), to avoid division by zero. In fact, it might be a good idea to also exclude points with very small, nonzero velocities, since they can result in unnecessarily small time steps.

2D convection simulation

Your final code should produce a cross-section of the solar interior that is a rectangular box with x along the horizontal direction and y along the vertical direction (considering that a star is a sphere, the vertical direction can also be referred to as the *radial* direction). The y-direction should enclose 4 Mm with the upper boundary at the solar surface (and lower boundary inside the Sun). The x-axis should encompass 12 Mm. The size of your computational box should be $N_x = 300$ and $N_y = 100$. You can assume an ideal gas with $\mu = 0.61$. The internal energy is

$$e = \frac{1}{(\gamma - 1)} n k_B T = \frac{1}{(\gamma - 1)} \frac{\rho}{\mu m_u} k_B T \tag{29}$$

where e has the units of energy per volume. The equation of state for an ideal gas is then given by

$$P = (\gamma - 1) e \tag{30}$$

You can also assume that gravity is constant in the box, so the gravitational acceleration can be assumed to be $|\mathbf{g}| = GM_\odot / R_\odot^2$ (make sure your gravity is in the right direction).

Initial conditions

The initial conditions for the temperature, pressure, density and energy follows from the following two requirements:

- The gas needs to be in hydrostatic equilibrium.

- The double logarithmic gradient

$$\nabla = \frac{\partial \ln T}{\partial \ln P} \tag{31}$$

  must be just slightly larger than 2/5.

From this gradient, and the fact that the gas needs to be in hydrostatic equilibrium, you will be able to calculate the temperature and pressure in the box (for the first time step) given the values at the top of the box. The top of the box must have temperature and pressure given by the values for the photosphere in Appendix B in the lecture notes. It is wise to find an expression for the temperature in terms of the gradient, before finding an expression for the pressure. Then, the density and internal energy are calculated based on the initial conditions for temperature and pressure in the box. The initial condition for the velocity is zero everywhere.

Boundary conditions

At the end points of the numerical grid ($i = 0$, $j = 0$, $i = N_x - 1$ and $j = N_y - 1$), the discretised hydrodynamic equations cannot be solved. This is because we do not have $\phi^n_{-1,j}$, $\phi^n_{N_x,j}$, $\phi^n_{i,-1}$ and ϕ^n_{i,N_y}, hence we need to apply boundary conditions that set the value of ϕ or the derivative of ϕ on the boundaries.

Since gravity is given, there is now an *up*- and a *down*-direction in the computational box. The boundaries will be called horizontal (x) and vertical (y). The boundary conditions are as follows:

Horizontal boundary

The horizontal boundary conditions are periodic, meaning that we set

$$\phi^n_{-1,j} = \phi^n_{N_x-1,j}$$
$$\phi^n_{N_x,j} = \phi^n_{0,j}$$

for each primary variable ϕ. The `numpy.roll` function is very useful for this.

Vertical boundaries

The vertical boundary conditions are the following:

Vertical boundary: Vertical velocity

The boundary conditions for the vertical component of the velocity should be zero both at the upper and lower boundary.

Vertical boundary: Horizontal velocity

The vertical gradient of the horizontal component of the velocity should be zero at the boundary.

Vertical boundary: Density and energy

The boundary conditions for density and energy are coupled (as seen from (29)), which means that you need to be careful when implementing them into your code. A requirement for the boundary conditions is that hydrostatic equilibrium must be fulfilled, meaning that the pressure gradient is given. A good starting place is to identify the pressure gradient, and use that to find the boundary conditions for ρ and e. Think about what parameter you need to calculate first.

Hint

Useful relations regarding the boundary conditions are given by the 3-point forward difference approximation

$$\left[\frac{\partial\phi}{\partial y}\right]_{i,j}^{n} = \frac{-\phi_{i,j+2}^{n} + 4\phi_{i,j+1}^{n} - 3\phi_{i,j}^{n}}{2\Delta y} \tag{32}$$

and the 3-point backward difference approximation

$$\left[\frac{\partial\phi}{\partial y}\right]_{i,j}^{n} = \frac{3\phi_{i,j}^{n} - 4\phi_{i,j-1}^{n} + \phi_{i,j-2}^{n}}{2\Delta y} \tag{33}$$

This is a second-order scheme that allows more accuracy by including 3 data points.

Visualisation

A module has been written to help you create movies and figures. It is available along with its **user guide**. It is important that you read the documentation, as minor mistakes in the implementation can cause serious errors in the visualisation.

Sanity test

In order to verify that your calculations and implementations are correct, you should check that your system is in hydrostatic equilibrium. Run your code with the visualisation module for 60 s. If there are no changes in the computational box, the system is in hydrostatic equilibrium. If you are using `numpy.zeros`, you must exercise caution as the sanity test passes when the elements of the velocity arrays are zero (which they are initially). Make sure that you are filling these arrays with values as the system evolves in time. You do not need snapshots from this run in your report, but you need to include the movie when submitting your project. 10 points

Code

The code has to be written using the `python 3` programming language. You can use this **skeleton code** to get started. The way your code is written impacts the number of points you get for this project. It should be easy to read, well commented and logically structured. The instructors should be able to run your code. 10 points

The report

You are required to write a report in this exercise. The report should discuss problems you have had underway and what you have found out while solving this exercise (such as numerical problems, resolution problems and problems with the physics). How the report is written impacts the amount of points you get on this project. All figures should have a reference in the main text and their content should be discussed.

You should include answers and explanations to the following bullet points. Try to include the answers to the points in the report, in the order they are given here. Some of the bullet points require you to complete the previous ones, but some are possible to complete independently. This is important to remember if you start to run out of time.

1. Show equations (14)–(17) using the continuity equation in (1), momentum equation in (2) and internal energy equation in (3). Why is gravity in the y-direction only? 5 points

2. Write out the algorithms used to calculate $w_{i,j}^{n+1}$ and $e_{i,j}^{n+1}$, and include the discretisations. Which terms are calculated using upwind differencing, and why? 10 points

3. Explain how you calculated the initial conditions for T and P, and the vertical boundary conditions for u, ρ and e. 15 points

4. Explain your code and how it has been put together. Was there anything in particular you needed to think about when updating the variables? When did you call the **boundary_conditions** function? 10 points

5. After your code is stable in hydrostatic equilibrium (run sanity test to verify), provoke the gas to become convectively unstable by implementing a 2D Gaussian perturbation in the initial temperature. When do you add the perturbation in order to get convective motions? It should be possible to turn the perturbation on and off. You are free to use any library for the Gaussian, but you need to give and explain the analytical function in the report. 10 points

6. Include movies of your 2D convection simulation for different parameters. One movie should have velocity visible as vectors and temperature shown in colour. In the additional movie(s) you are free to choose the parameter(s) you want. Use snapshots at different run times to explain how the system changes over time. 15 points

7. Your report should be well structured and easy to read. After the conclusion, you should write a "Reflection" section where you explain what you have learned from this exercise and what you have struggled with. 15 points

The project is delivered at http://devilry.ifi.uio.no. Before delivery, you should make a tarball that includes your report, code (and all files needed to run it) and movies. This is done by typing the following in a terminal (works on Linux/MacOS machines):

```
$ tar -cvf name.tar /path/to/directory
```

There will be no extensions to the deadline, except in case of documented medical circumstances. If you cannot solve the project, write the report anyway, explaining your problems and what you tried in order to solve them.