AST3310 Project 3

Stellar Convection

Caspar William Bruenech

May 20, 2020

1 Introduction

In this experiment we will create a numerical model for simulating how energy is transported in stellar interiors closer to the surface by way of a hot, low-density gas "bubble". In this first part of this article we will look at the hydrodynamic equations we will be using to model the gas, and rewrite these to a more practical form, before discretizing them and approximating using either central- or upwind difference schemes. We will then look at how we can use some assumptions about the initial state of the gas to produce analytical expressions for the initial values of the various parameters, before delving into the problem of the boundary conditions. Here we will use forward and backward difference approximations to get expressions for the parameter values at the two vertical boundaries. Finally, we will explain how the code is written, what it contains, and how it is run.

2 Method

2.1 Governing Equations

Our first part consists of rewriting the time evolution equations for the quantities we are interested in. For the density ρ we will be using the continuity equation with no sources or sinks, for the velocity $\vec{u}$, we will be looking at the momentum equation for x and y, and for the energy we will be using the energy equation. In the following sections we are using that $\vec{\nabla} = \left(\frac{\partial}{\partial x}\hat{x}, \frac{\partial}{\partial y}\hat{y}\right)$, and that $\vec{u} = (u\hat{x}, w\hat{y})$.

2.1.1 Continuity Equation

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

$$\Rightarrow \frac{\partial \rho}{\partial t} = -\vec{\nabla} \cdot (\rho \vec{u}) = -\frac{\partial \rho u}{\partial x} - \frac{\partial \rho w}{\partial y}$$

2.1.2 Momentum Equation

$$\frac{\partial \rho \vec{u}}{\partial t} + \nabla \cdot (\rho \vec{u}\vec{u}) = -\nabla P + \rho \vec{g}$$

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

Thus, we can collect the contributions to x- and y-direction separately, and we get

$$\frac{\partial \rho u}{\partial t} = -\frac{\partial \rho u^2}{\partial x} - \frac{\partial \rho uw}{\partial y} - \frac{\partial P}{\partial x} \tag{1}$$

$$\frac{\partial \rho w}{\partial t} = -\frac{\partial \rho w^2}{\partial y} - \frac{\partial \rho uw}{\partial x} - \frac{\partial P}{\partial y} + \rho g \tag{2}$$

Here we have used that gravity only works in the y-direction. This is because we set the y-direction equivalent the radial direction in the star. Thus, as gravity only works in the radial direction, we have that $\vec{g} = g\hat{y}$.

2.1.3 Energy Equation

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

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

2.2 Discretization

In order to advance the variables in time numerically, we need to discretize the equations. For momentum, we first rewrite the equation

$$\frac{\partial \rho w}{\partial t} = -\frac{\partial \rho w^2}{\partial y} - \frac{\partial \rho uw}{\partial x} - \frac{\partial P}{\partial y} + \rho g$$

$$= -w\frac{\partial \rho w}{\partial y} - \rho w\frac{\partial w}{\partial y} - u\frac{\partial \rho w}{\partial x} - \rho w\frac{\partial u}{\partial x} - \frac{\partial P}{\partial y} + \rho g$$

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

Thus we can discretize the equation into grid points with coordinates (i, j), where i denotes the x (horizontal) positions and j denotes the y (vertical) position. Additionally, we will use n to denote timestep n. Finally, for simplicity, we will be working with the quantity ρu and ρw as one unit. This gives us

$$\left[\frac{\partial \rho w}{\partial t}\right]_{i,j}^{n} = -[\rho w]_{i,j}^{n}\left(\left[\frac{\partial u}{\partial x}\right]_{i,j}^{n} + \left[\frac{\partial w}{\partial y}\right]_{i,j}^{n}\right) - u_{i,j}^{n}\left[\frac{\partial \rho w}{\partial x}\right]_{i,j}^{n} - w_{i,j}^{n}\left[\frac{\partial \rho w}{\partial y}\right]_{i,j}^{n} - \left[\frac{\partial P}{\partial y}\right]_{i,j}^{n} + g\rho_{i,j}^{n} \quad (3)$$

Here we see that all terms apart from the final two are multiplied with a velocity quantity. As a result, upwind differencing will be used for all the differentials apart from $\partial P/\partial y$.

The same procedure needs to be done for the energy equation. Expanding the equation we get

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

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

Discretizing gives us then

$$\left[\frac{\partial e}{\partial t}\right]_{i,j}^{n} = -[e]_{i,j}^{n}\left(\left[\frac{\partial u}{\partial x}\right]_{i,j}^{n} + \left[\frac{\partial w}{\partial y}\right]_{i,j}^{n}\right) - [u]_{i,j}^{n}\left[\frac{\partial e}{\partial x}\right]_{i,j}^{n} - [w]_{i,j}^{n}\left[\frac{\partial e}{\partial y}\right]_{i,j}^{n} - [P]_{i,j}^{n}\left(\left[\frac{\partial u}{\partial x}\right]_{i,j}^{n} + \left[\frac{\partial w}{\partial y}\right]_{i,j}^{n}\right) \quad (4)$$

Similarly, there are two terms who are multiplied by a velocity, $\partial e/\partial x$ and $\partial e/\partial y$. Thus we need to use upwind differencing with the corresponding velocity for these, and central difference scheme for the remaining derivatives in the equation.

All quantities are stepped in time by using forward time approximation. To update the velocity components, since we will compute the *momentum* time derivatives, we divide the forward time approximation of the momentum by the density in order to get the velocity.

2.3 Initial Conditions

Before starting our simulation, we need to initialize the values of the various quantities. For the velocities u and w, we will assume they are both initially zero at all points on the grid. To find the initial condition for T, we will make two assumptions; (1) the system starts in hydrostatic equilibrium, and (2) the gradient

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

needs to be slightly larger than 2/5. We then start by rewriting the gradient

$$\nabla = \frac{\partial \ln T}{\partial \ln P} = \frac{P}{T}\frac{\partial T}{\partial y}\frac{\partial y}{\partial P}$$

Which gives us

$$\frac{\partial T}{\partial y} = \nabla \frac{T}{P}\frac{\partial P}{\partial y}.$$

Since we assumed hydrostatic equilibrium initially, we know that $\partial P/\partial y = -g\rho$. We are also modeling the gas in the star as an ideal gas, meaning that we can write the density ρ in terms of the pressure and temperature

$$PV = Nk_BT = \frac{M}{\mu m_u}k_BT = \frac{M}{V}\frac{k_B}{\mu m_u}T = \rho\frac{k_B}{\mu m_u}T.$$

Inserting this into our expression for $\partial T/\partial y$, we get

$$\frac{\partial T}{\partial y} = -\nabla\frac{T}{P}g\rho = -\nabla g\frac{T}{P}\frac{\mu m_u}{k_B}\frac{P}{T} = -\nabla\frac{g\mu m_u}{k_B}.$$

Denoting then the surface of the star as $y = 0$, we can integrate this from $y = 0$ to y, giving us

$$T(y) = T_0 - \nabla\frac{g\mu m_u}{k_B}y \tag{5}$$

where T_0 is the initial temperature at the surface.

For the initial pressure, we can again use the hydrostatic equilibrium condition

$$\frac{\partial P}{\partial y} = -g\rho = -g\frac{P}{T}\frac{\mu m_u}{k_B}.$$

We can then divide by P, and insert the expression we found for the initial temperature. This gives us

$$\frac{\partial P}{\partial y}\frac{1}{P} = -\frac{g\mu m_u}{k_B}\frac{1}{T_0 - \nabla\frac{g\mu m_u}{k_B}y} = -\frac{g\mu m_u}{k_BT_0 - \nabla g\mu m_u y},$$

where in the last step we have just multiplied with k_B/k_B.
Similarly, we can then integrate this from $y' = 0$ to $y' = y$, giving us

$$\ln P|_{P_0}^{P} = \left[\frac{g\mu m_u \ln\left(k_B T_0 - \nabla g\mu m_u y\right)}{\nabla g\mu m_u}\right]_0^y$$

$$\ln P - \ln P_0 = \frac{1}{\nabla}\left[\ln\frac{k_B T}{k_B T_0}\right]$$

$$\Rightarrow P = \exp\left[\ln\left(T/T_0\right)^{1/\nabla} + \ln P_0\right]$$

$$P(y) = P_0\left(\frac{T}{T_0}\right)^{1/\nabla}$$

where P_0 is the initial pressure on the surface.
We can then use the consequences of the gas being an ideal gas to get the initial conditions for the density and energy

$$\rho = \frac{P}{T}\frac{\mu m_u}{k_B} \tag{6}$$

$$e = P(\gamma - 1) \tag{7}$$

where $\gamma \equiv c_p/c_v = 5/3$ (ideal gas).

2.4 Boundary Conditions

Since we are using differencing schemes that depend on the position of neighbouring grid points, we are not able to directly compute derivatives on the boundaries, as we do not have access to all the neighbours at these positions, thus we need to implement specific rules when differencing these points. For a two-dimensional problem like this, we have four cases of boundary conditions; two vertical and two horizontal. For the horizontal boundaries we will implement periodic boundary conditions. However, for the vertical boundaries we need to find analytical expressions for the values at these points. To do this, we will use the forward and backward difference approximations

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

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

At the top vertical border j is zero, and thus we need to use the forward difference approximation. Expressing it for $\phi_{i,0}^{n}$ we get

$$\phi_{i,0}^{n} = \frac{4\phi_{i,1}^{n} - \phi_{i,2}^{n} - 2\Delta y\left[\frac{\partial\phi}{\partial y}\right]_{i,j}^{n}}{3}$$

At the bottom vertical border, $j = n_y - 1$ (we have a total of n_y vertical grid points), we have to use backward difference approximation, and equivalently we get

$$\phi^n_{i,n_y-1} = \frac{2\Delta y \left[\frac{\partial \phi}{\partial y}\right]^n_{i,j} + 4\phi^n_{i,n_y-2} - \phi^n_{i,n_y-3}}{3}.$$

We then need to find analytical expressions for the vertical derivative at these boundaries. To do this, we will make certain assumptions for each of the variables we need boundary points for, namely u, w, ρ and e.

2.4.1 Velocity

For the velocities u and w, we will assume that w is zero at the vertical boundaries, and that the vertical gradient for u is zero at the boundaries. That gives us

$$w_{i,0} = w_{i,n_y-1} = 0$$

$$u_{i,0} = \frac{4u^n_{i,1} - u^n_{i,2}}{3}$$

$$u_{i,n_y-1} = \frac{4u^n_{i,n_y-2} - u_{i,n_y-3}}{3}$$

2.4.2 Energy & Density

For the energy, we will start with the expression from an ideal gas

$$e = \frac{P}{\gamma - 1}$$

Which gives us

$$\frac{\partial e}{\partial y} = \frac{1}{\gamma - 1}\frac{\partial P}{\partial y}$$

We can then use the assumption of hydrostatic equilibrium, and the equation for the density using the ideal gas law, giving us

$$\frac{\partial e}{\partial y} = -g\frac{1}{\gamma - 1}\frac{\mu m_u(\gamma - 1)}{k_B}\frac{e}{T} = -\frac{\mu m_u g}{k_B}\frac{e}{T} = -C_1\frac{e}{T}.$$

Which means that we get the discretized values at the boundaries as

$$\left[\frac{\partial e}{\partial y}\right]^n_{i,k} = -C_1\frac{e^n_{i,k}}{T^n_{i,k}}$$

where $k = (0, n_y - 1)$ and $C_1 \equiv -\mu m_u g/k_B$. Since this derivative contains the boundary value we are looking to compute $(e^n_{i,k})$, we need to insert this into the expression for the boundaries and solve for this value. For $k = 0$ we get

$$e^n_{i,0} = \frac{4e^n_{i,1} - e^n_{i,2}}{3} * \left(1 - \frac{2\Delta y C_1}{3T^n_{i,0}}\right)^{-1} = \frac{4e^n_{i,1} - e^n_{i,2}}{3 - 2\Delta y C_1 / T^n_{i,0}}.$$

Similarly, for $k = n_y - 1$ we have

$$e^n_{i,n_y-1} = \frac{4e^n_{i,n_y-2} - e^n_{i,n_y-3}}{3} * \left(1 + \frac{2\Delta y C_1}{3T^n_{i,n_y-1}}\right)^{-1} = \frac{4e^n_{i,n_y-2} - e^n_{i,n_y-3}}{3 + 2\Delta y C_1 / T^n_{i,n_y-1}}.$$

For the density, we can use the expression from ideal gas law, i.e.

$$[\rho]^n_{i,k} = \frac{(\gamma - 1)\mu m_u}{k_B} \frac{[e]^n_{i,k}}{[T]^n_{i,k}}$$

and use the values of e at the boundaries that we compute with the above equation.

Since the equations for e and ρ are coupled, we need to make an assumptions when computing these. The boundary conditions will be set after we have stepped the equations one time step. This means that we are actually setting the boundaries for time step $n + 1$. Since we update T by using the density ρ, we do not (yet) have the value of T^{n+1} which we need when setting the boundaries. As a result, we need to make an approximation by using the previous value T^n, in hope that the value does not change so much within the next step that a large error accumulates.

2.5 The Code

The program for solving the differential equations is written in the Julia Language, with visualizations being done with the *fvis3* fluid visualizer for Python [1]. The module contains a function for computing variable step length. This is to increase the numerical stability by making sure no quantity is increased by more than a certain percentage of its previous value. This is done by, for each time step, computing first all the time derivatives of the main variables, then computing the quantities

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

at each grid point. To also ensure that a fluid particle does not step past a whole grid point, we compute

$$rel(x) = \left| \frac{u}{\Delta x} \right|, \qquad rel(y) = \left| \frac{w}{\Delta y} \right|.$$

Finally, we take the maximum value of all the grid points for each quantity, followed by the maximum value of these *rel*-values. The time step is then computed with $\Delta t = p/\delta$, where p is the change percentage criterion (set to 0.1 in this experiment), and δ is the maximum value we found. This ensures that the quantity with the largest change in the next step does not change more than a fraction p of its previous value. When doing this, we need to be aware of grid points where the velocity is zero, as this would lead to division by zero. To circumvent this, we simply filter out the values that are close to zero by applying a mask.

The module contains functions for computing both upwind- and central differential schemes for both x- and y-direction. For the x-direction, all grid points are computed using periodic

boundary conditions, while for the y-direction, only the inner points (from $j = 0$ to $j = n_y - 2$) are computed, while the boundaries are kept as zero since these boundary values are not updated using the derivatives. The boundary conditions are set in the hydro_solver function, which steps the variables one time step. The boundary conditions are set *after* the variables have been incremented by one time step. This is because we need to update the boundaries with the newly stepped variable values, otherwise the boundaries would always have the values computed using the inner values in the previous time step. Thus we would not get the correct values when updating P and T.

3 Implementation & Results

In order to ensure that the system was working properly, a system was initialized without adding any gaussian noise to the temperature. To check that the system was initially in hydrostatic equilibrium, the quantity

$$\left| \frac{\partial P}{\partial y} - g\rho \right|$$

was computed. The results gave a mean absolute difference of 0.13, and a maximum absolute difference of 0.23, with the highest values appearing at the top of the box (surface of the star). While this was not exactly zero, as it should be in hydrostatic equilibrium, it was deemed small enough to satisfy our criterion for stability. To make sure the system remained stable for a relevant duration of simulation time, the simulation was run for a total of 100 seconds. Snapshots of the temperature at time $t = 1s$ and $t = 100s$ is shown in figure (1).

Another system was then initialized, this time with the addition of a normally distributed temperature, with a standard deviation of 15, an amplitude of 10^5K, and means of $\mu_x = x_{max}/2$, $\mu_y = y_{max}/2 - 0.25 * y_{max}$.

The system was simulated for 300 seconds, and snapshots of the temperature T and vertical velocity component w at times $t = 30s$ and $t = 299s$ are shown in figure (2), while videos of all the parameters are provided in the attachment. It was observed that at some point during the simulation, the time step became so small that it took an extremely long time to finish the simulation. Thus, an if-test was added to the timestep-function, wherein a time step of $10^{-2}s$ was returned if the computed time step was smaller than $10^{-2}s$. For all simulations we used a temperature gradient value of $\nabla = 0.402$.

4 Discussion

The results of the hydrostatic equilibrium sanity check shows us that the system remained in equilibrium throughout the 100 second simulation, with a maximum vertical velocity deviation of 1.25m/s at the end of the simulation. Additionally, the mean and maximum difference between the vertical pressure gradient and ρg were, respectively, 0.25 and 1.41 after 100 seconds. In figure (1) we see that the temperature and its velocity vectors are visually indistinguishable at the start and end time of the simulation. With these results we can conclude that further data produced by the model can be trusted.

A heatmap of the temperature in the box showed how the added temperature perturbation produced a spherical area with a higher temperature than its surrounding. As the idea of this perturbation is to simulate a hot, spherical gas bubble, this is a satisfactory result. A similar

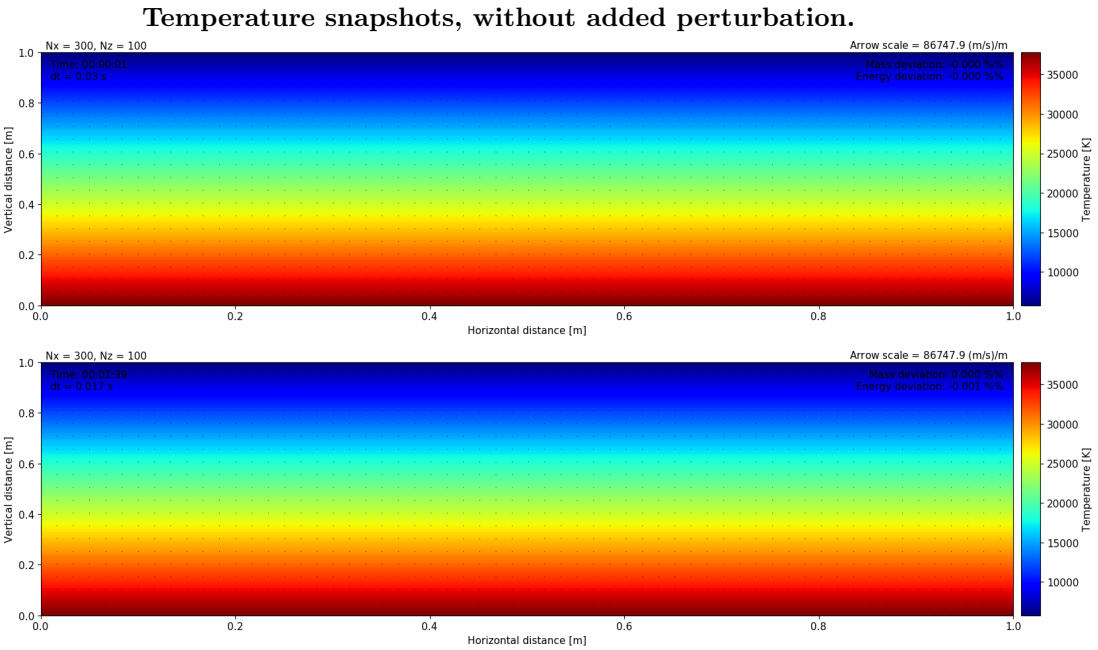

Figure 1: Snapshots of the temperature in the computational box at times $t = 1s$ (top) and $t = 99s$ (bottom).

heatmap made with the initial density showed exactly what we would expect; namely a lower density in the same region. The energy and density were initialized in such a way that this added perturbation did not affect their initial values. This is because including the perturbation in the pressure would equivalate to the high temperature bubble appearing suddenly, similar to performing a small explosion underwater. This is not in line with reality, where the bubbles gradually appears as it heats up and then slowly begin to ascend, which is what we observe in the simulation. In figure (2) we see two snapshots at $t = 30s$ and $t = 299s$, for both the temperature and the vertical velocity component. We see that by 30 seconds, the "bubble" has already ascended a small amount, and is continuing to rise. We see this from the larger velocity vectors surrounding the bubble, and the fact that the vertical velocity component has a large positive value in the same area. At $t = 299$ seconds, the bubble has risen from 25% of the height (its initial position) to around 40%. At this point we observe the curl produced by the rising temperature bubble. This is also something we would expect, as when the hot gas rises, the cooler air surrounding it is pushed aside and sucked into the vacuum left behind by the ascending bubble. This is further emphasised by the vertical velocity component, as we see a large, positive value in a smaller radius centered on the bubble, which is surrounded by larger negative values, indicating positions where the gas is descending. In the attached video of the horizontal velocity component, this is manifested as a clover-shape of positive and negative values.

The accuracy limitations of the numerical implementation and discretization of the differential equations starts to become apparent at $t = 300$ seconds. From the snapshots at this time, we see the birth of asymmetry in the vertical velocity component in the form of a red "tail" protruding from the gas bubble, which manifests as an increased curl in the temperature plot in this area. Simulations run for a total of 500 seconds showed that after 300 seconds, the system became chaotic and soon caused an overflow in the velocity vectors. This is a result of the combination of

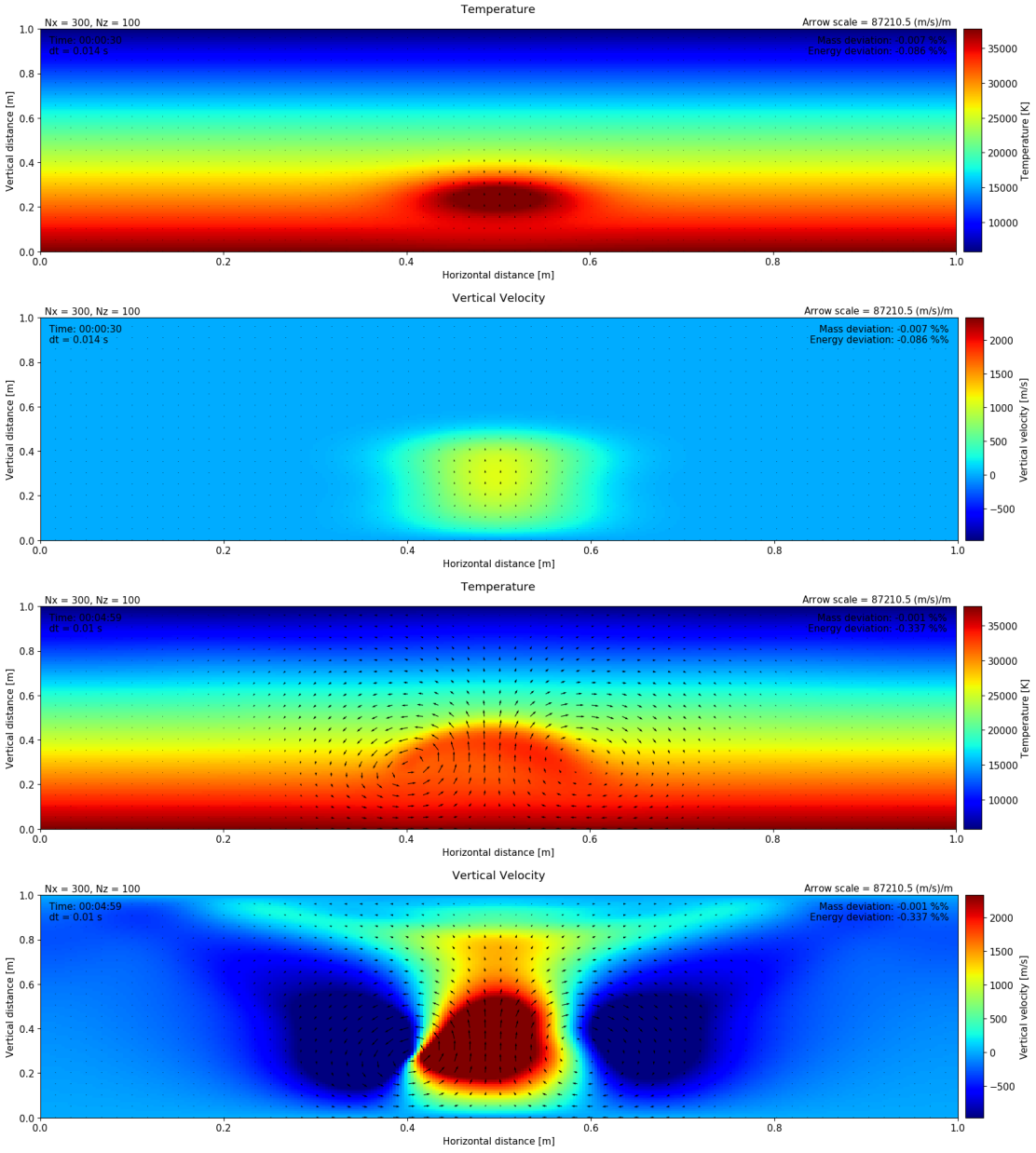

Figure 2: Snapshots of the temperature and vertical velocity component at times $t = 30$ seconds and $t = 299$ seconds.

assumptions and simplifications that we have done when translating the problem to a numerical implementation. For example, by approximating a derivative with a set step size, we produce an error, as the true derivative is defined for the step size as it approaches zero. When then calculating these derivatives thousands of times during our simulation, the error accumulates and eventually blows up. While a successful simulation of 300 seconds is a good results, it is possible that we could have had even longer stability times if we had not included the hardcoded time step value of $10^{-2}s$ (when the computed time step was too small). Unfortunately, the time and computational power was not available for this, but it remains a potential improvement for future reference.

A large portion of what we have learned in this experiment is how to increase the stability of a numerical hydrodynamics simulation. This includes the implementation of upwind difference schemes, and how they can be used in places where a non-constant velocity parameter is present to improve stability. Additionally, we looked at how to compute a variable step length to ensure parameters do not change too much in one time step. On top of this, we have looked at how one can use forward and backward difference approximations to find analytical expressions for non-trivial boundary conditions, such as with coupled equations. Finally, we have learned how to implement a hydrodynamics solver numerically using the Julia language, which has taught us a great deal about object-orientation with this language, and how it is different from the class-system in Python.

5 Conclusion

In this experiment we have created a numerical simulation of heat transport via convection in the upper surface of a star. We discretized various hydrodynamic equations to get methods for advancing the physical parameters of the gas in time. Analytical expressions for the initial values of the parameters were found by combining the hydrostatic equilibrium criterion with a set value of the double logarithmic temperature gradient, ∇, which we set to have a value of 0.402. Boundary conditions were implemented horizontally as periodic, while analytical expressions for the vertical boundaries were found using forward and back difference schemes. By implementing various methods for increasing the numerical stability of the system, we successfully produced a solver which managed to remain in hydrostatic equilibrium for at least 100 seconds. By adding a Gaussian perturbation with a standard deviation of 15 and an amplitude of 10^5 K in order to mimic the presence of a hot gas bubble, we simulated the gas for a total of 300 seconds, which produced satisfying results of how the energy is transported up towards the surface of the star due to the smaller density of the hot bubble compared to its surroundings. After 300 seconds, the system became unstable and soon exploded. While numerical error is unavoidable for these systems, it is possible that even longer stability times could have been produced by always using a variable time step length at the cost of more CPU time.

References

[1] Lars Frogner. “Using the FVis module”. In: ().