Draft version May 20, 2020
Preprint typeset using LaTeX style emulateapj v. 11/26/04

A TIME DEPENDENT CONVECTION MODEL IN A 2-DIMENSIONAL ENVIRONMENT.

Marius Torsvoll

Draft version May 20, 2020

ABSTRACT

The modelling of stellar convection in the stellar interior by a fluid-mechanical model using time evolving equations in a 2-d environment. Thus a numerical model must be devised for the time dependence to filter in. Thus it also becomes clear that the goal of this model is an animation and the time development of the gaseous parameters over time. Thus time snapshots shall be considered in this paper and the evolution shall be considered.

Subject headings: Convection, stellar interior, simulation.

1. INTRODUCTION

In this paper we shall explore a multidimensional model of a part of the stellar convection zone. The model aims to explain a fluid mechanical model of convection. Using a numerical model of convection and a discretisation of all relevant equations. Thus we use the theory of fluids to explain the rise of gas bubbles and thus energy transport. The model is rooted in the continuity equation of fluids and the state is based in the ideal gas law. This is the basis of the entire model and the simulation.

The model aims to have atleast 200 seconds of numerical stability and ideally more, but the model is fairly crude and thus a significant amount of stability is not to be expected. Thus to verify the model a sanity verification will be done with the hydrostatic case and if this passes for at least 100 seconds the model with the perturbation in the gas will be strengthened and assumed to have some validity. This is done through animation and observation of the visual changes, of no discernable change has happened it has passed.

The perturbation is the introduced and the model is studied, all of this is done through the Python 3.x programming language and the gas is visualised using an external module as shall be described later on. The model is then in 2 dimensions and evolves in time, the use of fluid mechanical principles aims to explain the evolution of gas in motion. The gas is assumed as mentioned to be ideal, monatomic and fully ionized.

2. METHOD

Here we will use the fluid mechanical coninuity equation to derive the needed time derivatives to explain the changes in density, flow velocity and internal energy as this fully describes the state of our gas. Thus it becomes clear that we are dealing with an ideal monatomic gas that we also assume to be fully ionized. This then gives us the basis for the equations of state that shall be used. And those are the following and we are stating just these without proof.

Electronic address: mariutel@astro.uio.no
[1] Institute of Theoretical Astrophysics, University of Oslo, P.O. Box 1029 Blindern, N-0315 Oslo, Norway

Thus for the internal energy:

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

So for the pressure gradient we start off assuming a system in hydrostatic equilibrium and thus we have the obvious pressure gradient that we can numerically evaluate for use in the internal energy equation.

$$\frac{\partial P}{\partial y} = -\rho \cdot g \tag{2}$$

Here the g is of course the standard newtonian gravitational acceleration. This is assumed to be constant for the entire simulation box. This is not too unreasonable as the relative change for such a system would not be too large.

Thus we further move on to

$$dT = \frac{T}{P} \cdot \rho g \nabla \tag{3}$$

This change in temperature stems from the following relation.

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

Then by replacement of ∂r with ∂y and a restocking of the equation it becomes obvious that:

$$dT = \frac{T}{P} \frac{\partial P}{\partial r} \nabla$$

$$\Rightarrow dT = \frac{-T}{P} \rho g \nabla \tag{4}$$

Due to hydrostatic equilibrium the partial derivative of the pressure disappears and we are left with the equation for hydrostatic equilibrium. Thus we have the change in temperature parameter.

Further on we move on to the continuity equation to find our main parameters.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u})$$

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

$$\frac{\partial e}{\partial t} + \nabla \cdot \rho \mathbf{u} = -P \nabla \cdot \mathbf{u} \tag{5}$$

So by using these three equations we can decompose them into workable equations that may be handled numerically. So we start off by using the following:

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \rho \mathbf{u}$$

$$\frac{\partial \rho}{\partial t} = -\frac{\partial \rho u}{\partial x} - \frac{\partial \rho w}{\partial y} \tag{6}$$

Thus we have our time independent density evolution of the time across time. Thus we have a basis for the evolution numerically, this we will return to. Then we move on to the flow velocities given by u and w, we can decompose this because we know that the following is true: $\nabla \cdot (\mathbf{u}\mathbf{u}) = \mathbf{u} \cdot (\nabla \mathbf{u})$ and find a time independent time evolutionary expression once again. we then decompose the following:

$$\frac{\partial \rho \mathbf{u}}{\partial t} = -\frac{\partial u^2}{\partial x} - \frac{\partial \rho u w}{\partial y} - \frac{\partial P}{\partial x} - \frac{\partial \rho w^2}{\partial y} - \frac{\partial \rho u w}{\partial x} - \frac{\partial P}{\partial y} - \rho g \tag{7}$$

As this is the time independent time evolutionary equation for the two-dimensional $\mathbf{u}$ we decompose this into x and y directions and thus we are left with:

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

$$\frac{\partial \rho w}{\partial t} = -\frac{\partial \rho w^2}{\partial y} - \frac{\partial u w}{\partial x} - \frac{\partial P}{\partial y} - \rho g_y \tag{8}$$

This becomes clear when we consider how the two dimensions behave with regards to the product rule, thus we are easily left with these expressions without any other mathematical tricks. It also becomes clear that since the gravitational acceleration only works in y direction and as it is constant for all points within our simulationary conditions. Thus we once again move on to the internal energy.

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

$$\frac{\partial e}{\partial t} = -\frac{\partial e u}{\partial x} - \frac{\partial e w}{\partial y} - P(\frac{\partial u}{\partial x} + \frac{\partial w}{\partial y}) \tag{9}$$

Again this result is achieved by basically writing out the scalar products and reviewing the result. So thus we have differential equations for all the main parameters of the model and we can start discretize all of the right hands of our equations. This will be done by using a forward time centered scheme and a upwind scheme. Thus we have the main numerical component of the model.

So for the upwind and FTCS schemes we have the following algorithms: So for the FTCS method we have the following:

$$\frac{\partial phi}{\partial x} = \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x}$$

$$\frac{\partial \phi}{\partial y} = \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y} \tag{10}$$

Should be noted about this when implemented numerically this becomes extremely more efficient when vectorized. This is also true with the upwind scheme, this becomes possible with the use of a roll function as this deals with all the values at once. This function then moves all matrix element either to the left or right and thus becomes the perfect tool for use with these schemes.

$$\frac{\partial \phi}{\partial x} = \frac{\phi_i(+) - \phi_{i-1}(+)}{\Delta x} + \frac{\phi_{i+1}(-) - \phi_i(-)}{\Delta x} \tag{11}$$

There is quite alot to unwrap here, the $\phi(+)$ signifies the advancement of the variable in question for when the flow in the same direction is positive and vice versa for the (-) notation. Thus we have a basis for the implementation in the program, this is done by finding out where the flow is positive and negative, masking is then used as way to extract the corresponding indices from the desired variable as to the positive and negative flow values, a matrix is then constructed and the upwind values is then returned in this. Rolling of the ϕ variable matrix is also yet again employed as this speeds up and streamlines the process. The process is the same in the y direction as for x, the flows and step lengths are obviously changed to the correct direction.

So for how the time step evolves in regard to the differentiated equations. We shall consider the following for all variables, now dubbed ϕ. Thus we have the following for the time step;

$$rel(\phi) = \frac{\Delta \phi}{\phi} \cdot \frac{1}{\Delta t} = |\frac{\partial \phi}{\partial t} \frac{1}{\phi}| \tag{12}$$

So we also need the fluid to not move with a velocity that allows it to travel outside the box thus we have for the combined velocity.

$$rel(\mathbf{u}) = |\frac{\mathbf{u}}{\partial r} \cdot \frac{1}{\Delta r}| = \frac{\mathbf{u}}{\Delta r} \tag{13}$$

This then can be split into the x and y direction and thus be applicable to the problem at hand. To further clarify the splitting then implies the splitting of $\mathbf{u}$ into u and w and r into x and y. Thus we have the requirement of the relative change. Then we move on to the calculation of the time step itself. Thus we introduce a p which is

the allowed change parameter. We shall also consider the max of the relative changes, giving us the largest optimal time step. Thus we have the following change;

$$\delta \cdot \Delta t = p$$

$$\Delta t = \frac{p}{\delta}$$

$$\delta = max(\sum(rel(\phi))) \tag{14}$$

Thus we have an expression of the time step and a way to impose the change parameter restriction.

Then the only thing missing is the discretisized parameters for the advancement of the instance, this will then be used in the solver bit of the model which will be explained in the simulation part. As for the discretisations themselves we have the following:

$$\left[\frac{\partial \rho}{\partial t}\right]_{i,j}^{n+1} = -[\rho]_{i,j}^{n}\left(\left[\frac{\partial u}{\partial x}\right]_{i,j}^{n} + \left[\frac{\partial w}{\partial y}\right]_{i,j}^{n}\right) -$$

$$[u]_{i,j}^{n} \cdot \left[\frac{\partial \rho u}{\partial x}\right]_{i,j}^{n} - [w]_{i,j}^{n}\left[\frac{\partial \rho w}{\partial y}\right]_{i,j}^{n}$$

$$\left[\frac{\partial \rho u}{\partial t}\right]_{i,j}^{n+1} = [\rho]_{i,j}^{n}[u]_{i,j}^{n} \cdot \left(\left[\frac{\partial u^2}{\partial x}\right]_{i,j}^{n} + \left[\frac{\partial wu}{\partial y}\right]_{i,j}^{n}\right) -$$

$$[u]_{i,j}^{n}\left[\frac{\partial \rho u^2}{\partial x}\right]_{i,j}^{n} - [w]_{i,j}^{n}\left[\frac{\partial \rho uw}{\partial y}\right]_{i,j}^{n} - \left[\frac{\partial P}{\partial x}\right]_{i,j}^{n}$$

$$\left[\frac{\partial \rho w}{\partial t}\right]_{i,j}^{n+1} = [\rho]_{i,j}^{n}[w]_{i,j}^{n} \cdot \left(\left[\frac{\partial w^2}{\partial y}\right]_{i,j}^{n} + \left[\frac{\partial uw}{\partial x}\right]_{i,j}^{n}\right) -$$

$$[w]_{i,j}^{n}\left[\frac{\partial \rho w^2}{\partial y}\right]_{i,j}^{n} - [u]_{i,j}^{n}\left[\frac{\partial \rho wu}{\partial x}\right]_{i,j}^{n} - \left[\frac{\partial P}{\partial y}\right]_{i,j}^{n} + [\rho]_{i,j}^{n} g_y$$

$$\left[\frac{\partial e}{\partial t}\right]_{i,j}^{n+1} = -[e]_{i,j}^{n}\left(\left[\frac{\partial u}{\partial x}\right]_{i,j}^{n} + \left[\frac{\partial w}{\partial y}\right]_{i,j}^{n}\right) - [u]_{i,j}^{n}\left[\frac{\partial eu}{\partial x}\right]_{i,j}^{n} -$$

$$[w]_{i,j}^{n}\left[\frac{\partial ew}{\partial y}\right]_{i,j}^{n} - [P]_{i,j}^{n}\left(\left[\frac{\partial u}{\partial x}\right]_{i,j}^{n} + \left[\frac{\partial w}{\partial y}\right]_{i,j}^{n}\right) \tag{15}$$

So as to how these equations are updated we have the following update for w. This is how the parameter is updated after the partial derivatives has been calculated with the FTCS and upwind schemes. A fair distinction between the uses of these two schemes is that FTCS is a higher order method than upwind, but upwind is still used for when the partial derivatives i.e ρ and u for example are multiplied with each other. This is because even though FTCS is higher order upwind is made for derivatives of multiplied functions. Thus for all "clean" partial derivatives the FTCS scheme is employed and upwind for the remainder.

$$[w]_{i,j}^{n+1} = \frac{\left([\rho]_{i,j}^{n}[w]_{i,j}^{n} + \left[\frac{\partial \rho w}{\partial y}\right]_{i,j}^{n}\Delta t\right)}{[\rho]_{i,j}^{n+1}} \tag{16}$$

This is how we update the density and so for the internal energy we have the following:

$$[e]_{i,j}^{n+1} = [e]_{i,j}^{n} + \left[\frac{\partial e}{\partial t}\right]_{i,j}^{n}\Delta t \tag{17}$$

These updates are of course based on the discretizations found in 15. The partial derivatives there are the ones used for the

2.1. *Initial and boundary conditions.*

So for the initial conditions we have a need to initialise 30000 data points, this is done by the use of the solar photospheric conditions. The temperature is initialised by use of an integrated 4, here we of course refer to the expression for dT, by integrating this expression we get the following:

$$T = T_0 - \mu m_u g \nabla \frac{y}{k_B} \tag{18}$$

Thus with this we have a way to initalise our temperature, by iterating over y we have a given temperature for all of our x values. Thus we are easily able to initialise the temperature. For when the temperature perturbation is introduced this gaussian function is added to the temperature gradient, but this gaussian is just initialised as a bell function permeating the entire box multiplied with an amplitude and with a small standard deviation. This change in temperature affects all other initialised parameters except the flow velocities, but this is not a problem since the internal energy, density and pressure is initialised post temperature.

The pressure is initialised at the same time and very much in the same way as the temperature. So the expression itself looks as follows:

$$P = P_0\left(\frac{T}{T_0}\right)^{\frac{1}{\nabla}} \tag{19}$$

So as to where this expression for the pressure stems from. It comes from the expression for hydrostatic equilibrium. By replacing the density and use of ideal gas this expression is easily obtained.

Thus we move on to the vertical boundary conditions here the main relevant ones are ρ, u and e. For the horizontal flow u in the vertical boundaries we need to impose a condition on the gradient for u. We need $\frac{\partial u}{\partial y} = 0$ This is done by setting the u term in the vertical boundary to be equal to $\frac{4u_{i,1} - u_{i,2}}{3}$ This ensures the gradient limit and sets that change to be 0. The same process is analogous for the bottom most vertical boundary here the j values

will only take on 98 and 97 given that indexing starts at 0. So for the internal energy conditions we have that

$$e_{i,0} = \frac{-e_{i,2} + 4e_{i,1}}{3 - 2\mu m_u g \frac{\Delta y}{k_B \cdot T_{i,0}}} \tag{20}$$

The same is true for the bottom of the box, but yet again the j index is set to be 99 for 0 and so on, to summarize the indices are adapted to fit the bottom of the box and the - in the denominator is swapped to a +.

So for the density as the pressure and thus the density can be found by the use of the internal energy of the gas. We have the following, this is of course a direct consequence of the assumption of ideal gas. Thus the density can easily be updated in the boundaries by the following:

$$\rho_{i,0} = e_{i,0} \frac{2\mu m_u}{3 k_B T_{i,0}} \tag{21}$$

This is also again true for the lower most boundary, just the temperature is switched and index 0 is replaced with 99 again.

3. SIMULATION

The simulations were carried out numerically using python 3, the animations were made using a solver class to advance the time step, but the looping and animations were made using an external module. The external module is called Fluid visualiser and is credited to Lars Frogner of ITA at UiO. The visualiser was in the 1.1.7 version. As mentioned this class is only used for the animation, the solution advancement is determined by the model described in this project.

So as the advancer class itself is based mostly on the singular step advancement. The main part of the solution is then the advancer class, basically this solver is a class that deals with the gaseous state in singular time states. To simplify it calculates the velocities and other parameters for a snapshot at a time. Thus it becomes ever more clear that the time advancement is performed by the visualiser module. But to move on to the explanation of the advancer class itself, it is very computational basic, it consists of a series of functions that all handle their part of the advancement. The first bit of the class consists of initialising the stellar parameters in the computational box we are using. This is done for all parameters, in the case for where convection is present and the gas is unstable, a gaussian perturbation is introduced in the initialised temperature with a centering around the center of the box, the gaussian bell encompass the entire box, but with such a relative standard deviation the amplitude quickly falls. The amplitude of the bell function is set to be 10000K, this gives a not too unrealistic bubble of unstable gas.

The next part of the advancer is then dedicated to the advancement of the variables, so the methods for implementing the forward time centered space scheme is then implemented and the upwind scheme is also implemented. These two methods are then split into the x and y direction for implementation, as the step length in x and y direction are not equal. Thus we have our numerical discretisations implemented, now we need to consider the time step length calculations. For the theoretical basis for this i refer to equation 14. The only notable thing about the implementation of this equation is that the flow velocities are extracted by the use of masking the matrices and the others are just extracted using vectorized numpy functions.

Thus we move on to the fluid mechanical solver itself and its use in the time step length. Shortly to explain how these hang together, as has been explained the differentiated parameters is used for the calculation of the time step. The differentiated parameters is then used for the time step length calculations, but before the time step method is called the boundary conditions is imposed. Thus setting the vertical boundaries and imposing the periodicality of the horizontal boundaries. Thus we then have the actual basis for calculating the time step needed for the visualiser module. The solver itself is simply based on updating the different parameters numerically as described in the discretization in the method 15. The density, internal energy and flow velocites are updated and then the boundary conditions are imposed on the updated parameters, finally as they are dependent on the other four parameters the pressure and temperature are updated. In all of this the time step is updated this is done before all the other time steps. As the relative time change as explained in 12 is dependent on the relative change in relation to the parameter.

Then an instance of the class and a perturbation is introduced as a matrix that is just added to the original temperature matrix, this is just a manual operation made by the user. The model runs the exact same way with the perturbation enabled post initialisation. The perturbation is introduced as a 2 dimensional gaussian bell multiplied by a constant to increase the temperature gradient.

4. RESULTS

Should be noted about all the figures present is that the x axis is supposed to span from 0 to 12Mm but it is only spanning 0 to 1, thus any fractional motion along the x axis is significantly larger than implied by the figure. The same is the case with the y axis only that it is 4Mm long. This is something to be aware of while reading the figures, this problem was attempted to be resolved, but to no avail hence this disclaimer.

So for the hydrostatic case we get exactly the thing we expected a gas with the velocity vectors being a static quantity and the temperature is a constant for the y level and the temperature only evolves in the y direction and then forms a gradient for the y direction. .As the temperature i stays constant we know from the ideal gas assumptions that the otheer parameters then are constant as temperature, pressure and density remains constant. As is expected this does not evolve in time, but as is also

expected as with numerical solutions the numerical decay happens but this only happens after the model has been verified. So for the hydrostatic case we have stability until the time of 200 seconds although the solution probably is stable for a time longer than this we stopped the simulation at this time due to significant computational times and the wish to experiment with the model the exhaustion of the hydrostatic case.

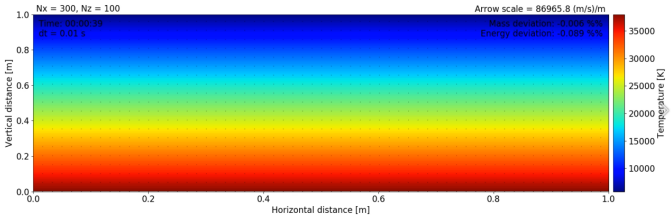

FIG. 1.— The hydrostatic case, the model remains like this for 200 seconds. This is useful for confirmation of the model and making sure it has no significant instabilities.

For the gaussian perturbation placed in the middle of the convective zone the hotter gas quickly rose to the top as the colder gas sank towards the bottom of the box, the flow moved inwards towards the centre of the perturbation. The temperature gradient expanded in a shock wave similar to that one would expect of an explosion. Which is not too unfair of a comparison as what we introduced kind of equates to that of an explosion. As an explosion is the rapid expansion of gas and a temperature and mainly pressure gradient towards an area which then will want to equalize with the rest of the gas. Thus the temperature gradient and pressure gradient will shock wave outwards. So in essence the gaussian perturbation as introduced is an explosion in slow motion. Thus amusingly one may again reiterate that the sun is just one massive continuous thermonuclear explosion.

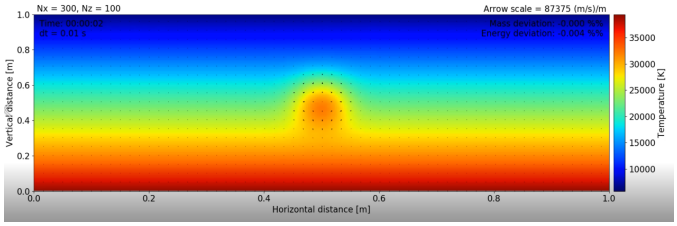

FIG. 2.— This is the initial state of the gas with a gaussian perturbation in center, this is the initial state of which we will study the convection process further. So for now we have our popping bubble or convection cell.

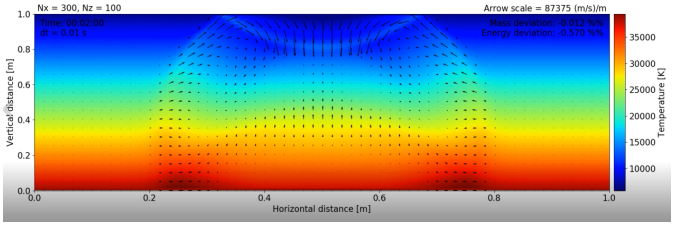

FIG. 5.— In this case we have the same situation as in 3, but the simulation has been allowed to evolve in time. Here we can see that the expanding or pooping bubble has reached our vertical boundaries, and has bounced off them, this is fairly realistic as the bubble will not be able to expand outside the stellar interior or into the non convective zone of the star. although this would for both cases not be exactly the same hard border as we have in the model.

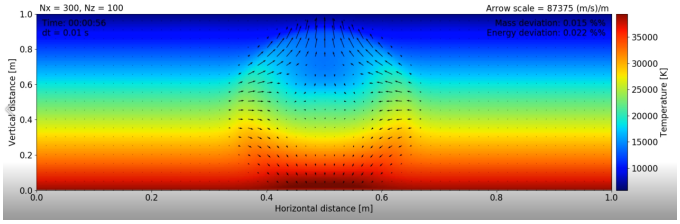

FIG. 3.— This is the same case as figure 2 although it has been allowed to expand, as discussed earlier here we can clearly see our "shock wave" moving radially out away from the center. As is clear in the movie but a bit hard to see from the figure the flow vectors on the expanding "shock wave" point towards the closest low temperature or pressure area.

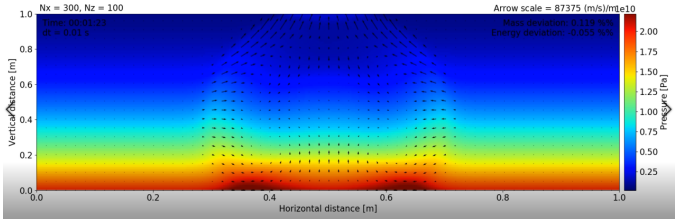

FIG. 4.— This is the pressure at a time not to different to 3 although at a slightly later moment. Here again we can see the same flow phenomena as in the aforementioned figure. The flow will on the wave move towards the nearest low pressure area which in the case foe the wave itself is in the opposite direction of the wavelike motion or vertically as the greatest pressure or temperature gradient is in that direction.

As can be seen from figure 3 the gas in the bottom hotter region will flow upwards moving energy and hotter gas towards the surface. As shown by figure 4the pressure wave moves radially out towards the boundaries of the box.

Thus as the sun is a rather complex system of rising gas, magnetic fields and other such niceties it would be rather interesting to see how the model behaved in a situation with several of these perturbations. It is also of interest if the model could handle several such perturbations, especially given that the model is based on the flow and evolution of a generic fluid model this would be interesting and as with the singular perturbation case we have the same evolutionary states although they infer some rather interesting effects on each other. This case was run for a shorter time period as it becomes unstable quicker than the singular case. This is simply due to the

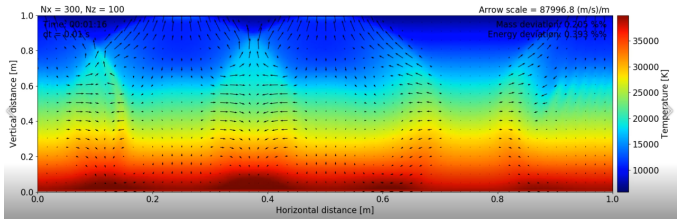

FIG. 6.— The case for several disturbances in the gas, the interesting part here is the interaction between the cells themselves as they reinforce the rise of gas towards the surface where the bubbles interact.

increase in the numerical simulation that forces inaccuracies quicker. This was done mainly just out of interest and the result is as shown here.

5. DISCUSSION AND CONCLUSIONS

As we can see from the model and the snapshots the model really has some advantages as it is not too computationally heavy and allows a relatively long simulation time. It is also stable for the hydrostatic case for at least 200 seconds, and some may have been able to achieve numerical stability using a laptop for up to 5.5 minutes. Although this does not sound like a massive amount of time the velocities involved are fairly massive and the whole span of the box is almost affected by the expanding gas bubble, thus implying that the shock waves are capable of moving with astonishing speeds as it is as mentioned 12x4 Mm large.

As we can see when the density and pressure inside the perturbation is higher than the surrounding gas the pressure and temperature will equalize itself in a wave like motion, it behaves in many waves like an explosion, but the more direct analogue for this situation is the rise of a gas bubble that then pops. This again is analogous to an explosions. But we shall stick to the popping of a bubble analogy for the moment, as seen from the simulations the density, temperature and pressure tries to equalize in a wave like motion similar to that of a droplet falling onto a water surface. Though the shape of the dispersion of this wave is different as the difference in density in the box gives a slightly different dispersion pattern.

6. LEARNED SUMMARY

So to summarize what I have learned in this project it is a few things, I have actually learned how to do object oriented programming. Which may come advantageous at later points in my life, I also learned that I don't want to study astrophysics in the direct case I want to angle it more towards a nuclear physics perspective. Now as for this course, I learned that lectures are really important for the conceptual and intuitive understanding of a subject which this course has suffered greatly under. I also learned a few things about the mathematical basis of fluid mechanics and how mathematically convection is defined. But as to what this course set out to do that's about it.

Acknowledgements

I wish to extend my most sincere gratitude towards Helle Bakke for being the MVP in my semester, for helping with projects generally and answering questions on piazza.

I also wish to thank Torstein for helping me actually understand what the hell I am doing, and being a great coding and debugging partner.

Also, God save the queen

REFERENCES