Numerical Analysis of Convection in the Inner Core (DRAFT)

Third Year Physics ASC

Start Date: 16th of August 2021 End Date: 29th of October 2021

Maximilian Williams (u6338634)

November 2021

Abstract

Convection in the Earth's inner core has been a contentious topic in geoscience. Recently, it has been proposed through seismic observations that Earths inner core is convecting. Here we numerically model the Earth's inner core as a self-gravitating internally heated fluid bousinesque fluid in 2-dimensions using a streamfunction-vorticity method and Thermal Lattice Boltzman approach. The streamfunction-vorticity approach is studied using two codes in Cartesian and polar coorindates. The streamfunction and Thermal Lattice Boltzmann codes are written from scratch in python3 and goLang. (this will be extended once I get my full results.)

Introduction

I still need to write this section. It will include answers to questions such as:
Why is this important?
What are some other works on this topic
Why did I take the approach that I did.

Governing Equations

In this section I introduce the physics of the problem; the governing equations that we wish to numerically solve.

Throughout analysis we describe the fluid in the Eulerian frame under a gravitational acceleration $\vec{g}$ which may vary in space. We give each location in the fluid a velocity $\vec{u}$ and density ρ that vary in space $\vec{x}$. We assume that fluid is incompressible and its viscosity μ and thermal diffusivity κ are constant. By conserving fluid momentum, we produce the Navier-Stokes equation:

$$\rho\frac{D\vec{u}}{Dt} = \rho\vec{g} - \nabla p + \mu\nabla^2\vec{u}, \tag{1}$$

where $\frac{D}{Dt} = \frac{\partial}{\partial t} + (\vec{u}\cdot\nabla)$ is the material derivative, p the pressure and ∇ the del operator. The dynamics of fluid temperature T are described the inhomogenous advection diffusion equation:

$$\frac{DT}{Dt} = \kappa\nabla^2 T + H, \tag{2}$$

where H is internal heating. The density of the fluid ρ is assumed to vary linearly in temperature according to the equation of state:

$$\rho = \rho_0(1 - \alpha(T - T_0)), \tag{3}$$

where α is the volumetric expansion coefficient and ρ_0 the density at a reference temperature T_0. We assume that density variations are small and that the inner core is evolving on geologic timescales, and as such take the velocities $\vec{u}$ to be first order. These assumptions allow us to make the slow flow boussinesq approximation to equation 1:

$$\frac{\partial\vec{u}}{\partial t} = \frac{\rho'}{\rho}\vec{g} - \frac{\nabla p'}{\rho_0} + \nu\nabla^2\vec{u}, \tag{4}$$

where $\rho' = -\alpha(T - T_0)$, ν the kinematic viscosity $\nu = \frac{\mu}{\rho_0}$ and p' a first order perturbation to the background pressure p_0.

$$Pr = \frac{\nu}{\kappa} \tag{5}$$

We use two numbers to characterise the convection in the problem, the Prandtl number and the Rayleigh number. The Prandtl number Pr (equation 5) gives that ratio of momentum and thermal diffusivity. The Rayleigh number Ra (equation **??**):

$$Ra = \frac{g\alpha\Delta T d^3}{\nu\kappa}, \tag{6}$$

where ΔT is the variation over the length scale of the problem d. The Rayleigh number is the ratio of timescales for thermal diffusion and thermal convection. High Rayleigh numbers $>\approx 650$ imply thermal convection. For internally heated problems such as ours, the temperature variation is poorly defined and so we use $\Delta T = \frac{Hd^2}{\kappa}$ instead, giving the internally heated Rayleigh number:

$$Ra_H = \frac{g\alpha H d^5}{\nu\kappa^2}. \tag{7}$$

Numerical Methods

Two numerical methods are introduced for solving the thermal convection problem, the Lattice Boltzmann Method and the streamfunction-vorticity formulation.

Streamfunction-Vorticity formulation

Here I introduce the streamfunction-vorticity method for use in 2 dimensions and use it to eliminate pressure terms in 4 and 2 in Cartesian and polar geometries giving a set of numerically solvable equations.

The streamfunction-vorticity formulation is a popular method for analytical and simple numerical analysis of incompressible fluids in two dimensions. Its main advantage is its elimination of all pressure terms, which would typically require iterative techniques to solve, an example being the SIMPLE algorithm and its variations. However, the streamfunction-vorticity method is limited to 2-dimensional and 3-dimensional symmetric flows and so has limited applications.
We define two scalar quantities, the vorticity ω and the streamfunction ψ. The vorticity ω is given by:

$$\omega = (\nabla \times \vec{u})_z, \tag{8}$$

where the z subscript denotes the component out of our 2D simulation plane, and ψ implicitly by:

$$\omega = -\nabla^2 \psi. \tag{9}$$

Given a coordinate system, and a clever definition of $\vec{u}$ we can rewrite equations 2 and 4 in terms of ω and ψ rather than $\vec{u}$ and p. In Cartesian coordinates (x, y) we pick

$$u = \frac{\partial \psi}{\partial y}, v = -\frac{\partial \psi}{\partial x}, \tag{10}$$

allowing us to write equations 2 and 4 as:

$$\frac{\partial T}{\partial t} + \frac{\partial \psi}{\partial y}\frac{\partial T}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial T}{\partial y} = \kappa \nabla^2 T + H \tag{11}$$

$$\frac{\partial \omega}{\partial t} =$$

Similarly, in polar coordinates (r, θ) we pick:

$$u = \frac{1}{r}\frac{\partial \psi}{\partial \theta}, v = -\frac{\partial \psi}{\partial r}, \tag{13}$$

giving:

$$\frac{\partial T}{\partial t} + \frac{1}{r}\frac{\partial \psi}{\partial \theta}\frac{\partial T}{\partial r} - \frac{1}{r}\frac{\partial \psi}{\partial r}\frac{\partial T}{\partial \theta} = \kappa \nabla^2 T + H \tag{14}$$

$$\frac{\partial \omega}{\partial t} = -\frac{g_r}{\rho_0 r}\frac{\partial \rho'}{\partial \theta} + \nu \nabla^2 \omega. \tag{15}$$

Importantly, our definitions of u and v in equations 10 and 13 satisfy equation 8 and 9. Equations 9, 11, 12 for the Cartesian case and 9, 14, 15 for the polar case can be directly solved.

Solving the Streamfunction-Vorticity equations

The finite difference method used for solving the Streamfunction-Vorticity-formulated governing equations is shown

We first discretize our domain $\mathcal{D}$. In the Cartesian case, we use $(x_i, y_j) = (i\Delta x, j\Delta y)$ with integers i and j satisfying $0 \leq i < N_x$ $0 \leq j < N_y$. In the polar case, we use $(r_i, \theta_j) = (R_0 + i\Delta r, j\Delta\theta)$ again with $0 \leq i < N_r$ and $0 \leq j < N_\theta$. We impose $\Delta\theta = \frac{2\pi}{N_\theta - 1}$ for consistency with θ-periodic boundary conditions and an inner radius R_0 in polar coordinates to avoid singularities generated by $r = 0$. We discretize time t by $t_n = n\Delta t$. For a function f, we use $f_{i,j}^n$ to denote f evaluated at time n at position (x_i, y_j) in Cartesian coordinates or (r_i, θ_j) in polar coordinates.
To approximate derivatives we use a finite difference approach. All time derivatives are approximated by forward difference:

$$\frac{\partial f}{\partial t} = \frac{f^{n+1} - f^n}{\Delta t} \tag{16}$$

Second order space derivatives are approximated by a central difference:

$$\frac{\partial^2 f_{i,j}}{\partial {x_1}^2} = \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{\Delta {x_1}^2}, \tag{17}$$

$$\frac{\partial^2 f_{i,j}}{\partial {x_2}^2} = \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{\Delta {x_2}^2}, \tag{18}$$

where x_1 is the first coordinate and x_2 is the second coordinate. For example, in Cartesian coordinates (x, y), we would have $x_1 = x$ and $x_2 = y$. For non advection terms, we approximate first order spatial derivatives by:

$$\frac{\partial f_{i,j}}{\partial x_1} = \frac{f_{i+1,j} - f_{i-1,j}}{2\Delta x_1}, \tag{19}$$

and

$$\frac{\partial f_{i,j}}{\partial x_2} = \frac{f_{i,j+1} - f_{i,j+1}}{2\Delta x_2}. \tag{20}$$

For advection terms, of the form $a\frac{\partial f_{i,j}}{\partial x_1}$ we employ a first order Godunov scheme:

$$a\frac{\partial f_{i,j}}{\partial x_1} = \frac{1}{\Delta x}(\mid a \mid (\frac{1}{2}f_{i+1,j} - \frac{1}{2}f_{i-1,j}) - a(\frac{1}{2}f_{i+1,j} - f_{i,j} - \frac{1}{2}f_{i-1,j})). \tag{21}$$

This scheme is always upstream, regardless of the direction of the advecting field a.
Other more accurate, but substantially more complex methods for solving these equations, particularly the advection equation exists such as the Semi-Lagrange Crank-Nicholson scheme.
To solve the streamfunction-vorticity equations we assume a starting vorticity ω on our domain $\mathcal{D}$. We then apply the Jacobi method to solve equation 9 for ψ on the interior of the domain which we call $\mathcal{D}'$. Using ψ we update T on $\mathcal{D}'$ using equation 11 (or 14 for polar). Finally, ω is updated on $\mathcal{D}'$ using equation 12 (15 for polar). This process is repeated.

The Jacobi Method

Here a basic numerical method for solving the Poisson equation, the Jacobi method is outlined.

We cannot solve for ψ explicitly in equation 9. Instead we use an iterative Jacobi method. In Cartesian coordinates, equation 9 is:

$$\omega_{i,j} = \frac{\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j}}{\Delta x^2} + \frac{\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}}{\Delta y^2} \tag{22}$$

Rearranging for ψ

$$\psi_{i,j} = \frac{\Delta x^2 \Delta y^2}{2(\Delta x^2 + \Delta y^2)}(\frac{\psi_{i+1,j} + \psi_{i-1,j}}{\Delta x^2} + \frac{\psi_{i,j+1} + \psi_{i,j-1}}{\Delta y^2} + \omega_{i,j}). \tag{23}$$

We then use the result of $\psi_{i,j}$ back into equation 23 to solve for $\psi_{i,j}$ iterately as shown in equation 24:

$$\psi_{i,j}^{(k+1)} = \frac{\Delta x^2 \Delta y^2}{2(\Delta x^2 + \Delta y^2)}(\frac{\psi_{i+1,j}^{(k)} + \psi_{i-1,j}^{(k)}}{\Delta x^2} + \frac{\psi_{i,j+1}^{(k)} + \psi_{i,j-1}^{(k)}}{\Delta y^2} + \omega_{i,j}). \tag{24}$$

Where the superscript (k) means the result of the k^{th} iteration of the above equation and this operation us applied to all points in $\mathcal{D}'$. We terminate this iterative method once the error $\sum_{(i,j)\in\mathcal{D}'} \mid \psi_{i,j}^{(k+1)} - \psi_{i,j}^{(k)} \mid$ gets sufficiently small. A similar method is applied to the polar coordinate case.

Lattice Boltzmann Method

In this section I give a brief introduction to the Lattice Boltzmann Method and why its different from traditional techniques. I introduce a 2-dimensional lattice D2Q9 and describe the Thermal Lattice Boltzmann method (TLBM) which is employed by my numerical solution to simulate convection.

The Lattice Boltzmann Method (LBM) is a generalization of a Lattice Gas Automata (LGA), which are themselves a specialized Automata for simulating fluid flows. These Automata methods like, common fluid simulation techniques discretize space and time. They directly simulate the state of particles or their distributions and evolve in time according to rules which give the desired macroscopic fluid properties as an emergent effect. This is fundamentally different from typical approaches which amount to directly numerically solving a set of partial differential equations.
To discretize space, we place nodes at locations $(x_i, y_j) = (i, j)$ with i, j integers. Each node has attached to it a lattice, here the D2Q9 lattice shown in figure 1. The lattice defines unit vectors $\vec{e}_i$, $i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$. In addition, each direction e_i gets a weight w_i. In the D2Q9 lattice these are:

$$w_i = \begin{cases} \frac{4}{9} & \text{if } i = 0 \\ \frac{1}{9} & \text{if } i = 1, 2, 3, 4 \\ \frac{1}{36} & \text{if } i = 5, 6, 7, 8 \end{cases}.$$

We wish to simulate convection. For this, we require the particle motion and the internal energy throughout the lattice. We define two distribution functions $f_\alpha(\vec{x}, t)$ and $g_\alpha(\vec{x}, t)$ denoting the particle and internal energy distributions along direction α at lattice points $\vec{x}$ and times t. The direction can be thought of as the direction of flow for particles or energy. Each timestep there are two steps to updating f and g, a streaming step:

$$f_\alpha(\vec{x} + \vec{e}_\alpha, t + \Delta t) = f_\alpha(\vec{x}, t), \tag{25}$$

$$g_\alpha(\vec{x} + \vec{e}_\alpha, t + \Delta t) =$$

and a collision step:

$$f_\alpha(\vec{x}+\vec{e}_\alpha, t+\Delta t) = f_\alpha(\vec{x}, t)+\frac{1}{\tau_f}(f_\alpha^{eq}(\vec{x}, t)-f_\alpha(\vec{x}, t))+F_\alpha \tag{27}$$

$$g_\alpha(\vec{x}+\vec{e}_\alpha, t+\Delta t) = g_\alpha(\vec{x}, t)+\frac{1}{\tau_g}(g_\alpha^{eq}(\vec{x}, t)-$$

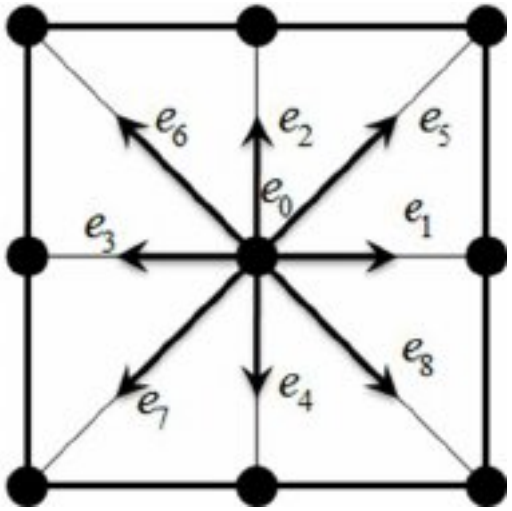

Figure 1: D2Q9 Lattice. Black nodes represent lattice points, vectors $e_0, .., e_8$ are the lattice vectors. Image sourced from [2]

Here F_α and G_α are the forcing terms terms and f_α^{eq} and g_α^{eq} are equilibrium distributions. The relaxation times τ_f and τ_g are related to the macroscopic thermal diffusivity (κ) and kinematic viscosity ν by:

$$\tau_g = \frac{3\kappa}{S^2\Delta t} + \frac{1}{2}, \tag{29}$$

$$\tau_f = \frac{3\nu}{S^2\Delta t} + \frac{1}{2}. \tag{30}$$

The equilibrium distributions are given by the BKG approximation:

$$f_\alpha^{eq}(\vec{x},t) = \rho w_\alpha(1 + 3\frac{\vec{e}_\alpha\cdot\vec{u}}{s^2} + \frac{9}{2}\frac{(\vec{e}_\alpha\cdot\vec{u})^2}{c^4} - \frac{3}{2}\frac{\vec{u}\cdot\vec{u}}{c^2}), \tag{31}$$

$$g_\alpha^{eq}(\vec{x},t) = \epsilon\rho w_\alpha(1 + 3\frac{\vec{e}_\alpha\cdot\vec{u}}{s^2} + \frac{9}{2}\frac{(\vec{e}_\alpha\cdot\vec{u})^2}{c^4} - \frac{3}{2}\frac{\vec{u}\cdot\vec{u}}{c^2}) \tag{32}$$

Here ρ, ϵ and $\vec{u}$ are the macroscopic density, internal energy and velocity given by:

$$\rho = \sum_{i=0}^{8} f_i, \tag{33}$$

$$\rho\vec{u} = \sum_{i=0}^{i=8} f_i\vec{e}_i \tag{34}$$

$$\rho\epsilon = \sum_{i=0}^{i=8} g_i. \tag{35}$$

The forcing terms F_i and G_i are problem dependent. For thermal convection $G_i = 0$ and F_i is a gravitational term Δf_α:

$$\Delta f_\alpha = -w_\alpha\rho\alpha\epsilon\frac{\vec{e}_\alpha}{\mid\vec{e}_\alpha\mid}\cdot\vec{g}\frac{1}{\tau_{grav}}, \tag{36}$$

where $||$ is the vector norm and $\vec{g}$ is the gravitational acceleration. Here τ_{grav} is the relaxation time for the gravitational field. We take $\tau_g = 0.6$ here, following [3]. The algorithm used to evolve the above TLBM equations is given in algorithm 1

Boundary Conditions

In both streamfunction-vorticity codes, boundaries are set to be fluid-impermeable, non-slip and insulating. In the Lattice Boltzmann code, "bounce back" boundary condition was used. This is a fluid-impermeable, insulating and slip boundary. These boundary conditions were selected due to simplicity, particularly, in the Lattice Boltzmann Method case. To generate a cooling boundary condition, the heating field near the boundary was set negative. This method is crude and necessary to avoid greater complexity in the Lattice Boltzmann case, but should in future be replaced with the traditional method of fixing heat flux across the boundary in the streamfunction case.

Advection tests

A particularly difficult part of the streamfunction-vorticity simulation is correctly modelling advection. In this section, I perform some tests to show that my advection scheme works.

Next we tested the accuracy of the advection schemes. Thermal diffusivity (κ) and thermal expansion (α) were set to zero to avoid convection. The fluid was set to temperature 0 (arb. units) except a small segment which was set to 1 shown in yellow top left of figure 2, 3, 4, 5. A streamfunction ψ_c was enforced to produce diagonal back and forth motion or a continuous horizontal or azimuthal motion to cross a periodic boundary. Details of the streamfunctions used are given in appendix.
The advection results for both codes are similar. For the periodic advection test figures 2 and 4 the temperature 1 region was advected across the periodic boundary without additional distortion. The centroid (red dot) also lagged the advection field by $\approx 1\%$ and $\approx 10\%$

Algorithm 1 Thermal Lattice Boltzmann Algorithm

$f_\alpha = \rho_0 w_\alpha$ ▷ Set particle distribution in domain
$g_\alpha = 0$ ▷ Set internal energy zero in domain
$\epsilon = 0$ ▷ Allocate memory for internal energy
H ▷ Set heat field in the domain
$\rho = 0$ ▷ Allocate memory for density
$\vec{u} = 0$ ▷ Allocate memory for velocities
while Simulation Running **do** ▷ Begin main simulation loop
  $g_\alpha += \rho \epsilon_b w_\alpha$ ▷ At each timestep, add the affect of the heating field
  if $x - \vec{e}_\alpha$ is solid **then** ▷ The following 3 lines enforce bounce-back boundary conditions
    $f_\alpha(x) = f_{\alpha'}(x)$
    $g_\alpha(x) = g_{\alpha'}(x)$
  else
    $f_\alpha(x) = f_\alpha(x - \vec{e}_\alpha)$ ▷ Streaming step
    $g_\alpha(x) = g_\alpha(x - \vec{e}_\alpha)$
  end if
  $\rho = \sum_{i=0}^{i=8} f_i$ ▷ t ▷ Compute and set macroscopic density
  $\vec{u} = \frac{(\sum_{i=0}^{i=8} f_i \vec{e}_i)}{\rho}$ ▷ Compute and set macroscopic velocity
  $\epsilon = \frac{\sum_{i=0}^{i=8} g_i}{\rho}$ ▷ Compute and set internal energy
  $f_\alpha^{eq} = \rho w_\alpha (1 + 3\frac{\vec{e}_\alpha \cdot \vec{u}}{s^2} + \frac{9}{2}\frac{(\vec{e}_\alpha \cdot \vec{u})^2}{c^4} - \frac{3}{2}\frac{\vec{u} \cdot \vec{u}}{c^2})$ ▷ Get equilibrium distribution f
  $g_\alpha^{eq}(\vec{x}, t) = \epsilon \rho w_\alpha (1 + 3\frac{\vec{e}_\alpha \cdot \vec{u}}{s^2} + \frac{9}{2}\frac{(\vec{e}_\alpha \cdot \vec{u})^2}{c^4} - \frac{3}{2}\frac{\vec{u} \cdot \vec{u}}{c^2})$ ▷ Get equilibrium distribution g
  $\Delta f_\alpha = -w_\alpha \rho \alpha \epsilon \frac{\vec{e}_\alpha}{|\vec{e}_\alpha|} \cdot \vec{g}$ ▷ Get gravitational term
  $\tau_g = \frac{3\kappa}{S^2 \Delta t} + \frac{1}{2}$ ▷ Compute relaxation times
  $\tau_f = \frac{3\nu}{S^2 \Delta t} + \frac{1}{2}$
  $f_\alpha = f_\alpha + \frac{f_\alpha^{eq} - f_\alpha}{\tau_f} + \Delta f_\alpha$ ▷ Collision step
  $g_\alpha = g_\alpha + \frac{g_\alpha^{eq} - g_\alpha}{\tau_g}$
end while

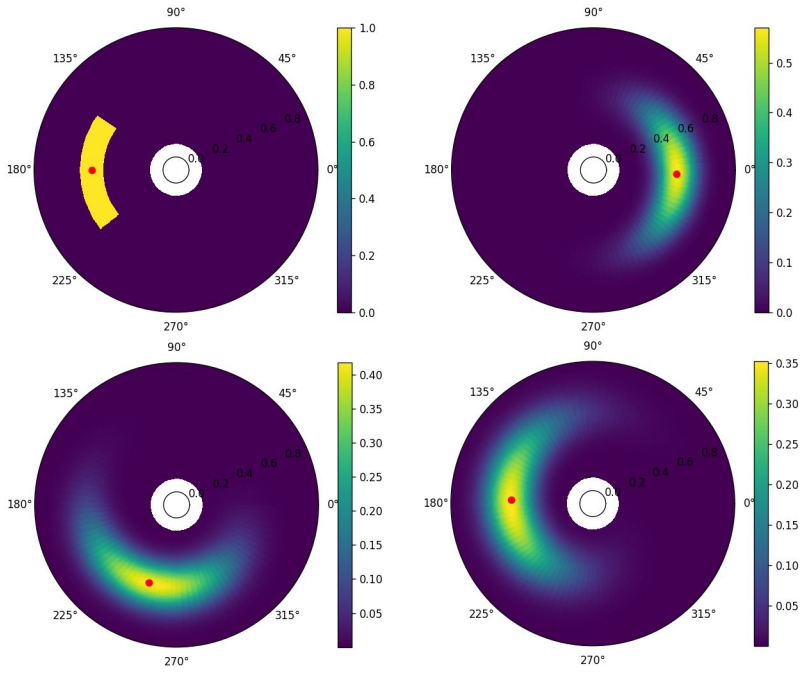

Figure 2: Polar azimuthal advection test. Temperature (color) is plotted in polar space. Red dot indicates temperature weighted mean position within the fluid and is taken as the location of the temperature 1 zone. Fluid is advected by an anticlockwise flow. Chronological order is Top left, top right, bottom left, bottom right. Top left shows the initial condition. Top right shows advection across periodic boundary. Bottom left is state when an exact scheme would have returned to the original position. Bottom right is the final state after being advected around one rotation.

for the Cartesian (figure 4) and polar case (figure 2) respectively. It is unknown why these differ substantially, but a probable cause is a smaller timestep used to produce the polar tests. In all cases, the bluring shows numerical diffusion which happens along the direction of the advection field as predicted by equation **??** (equation not produced here). The discontinuity in temperature in figure

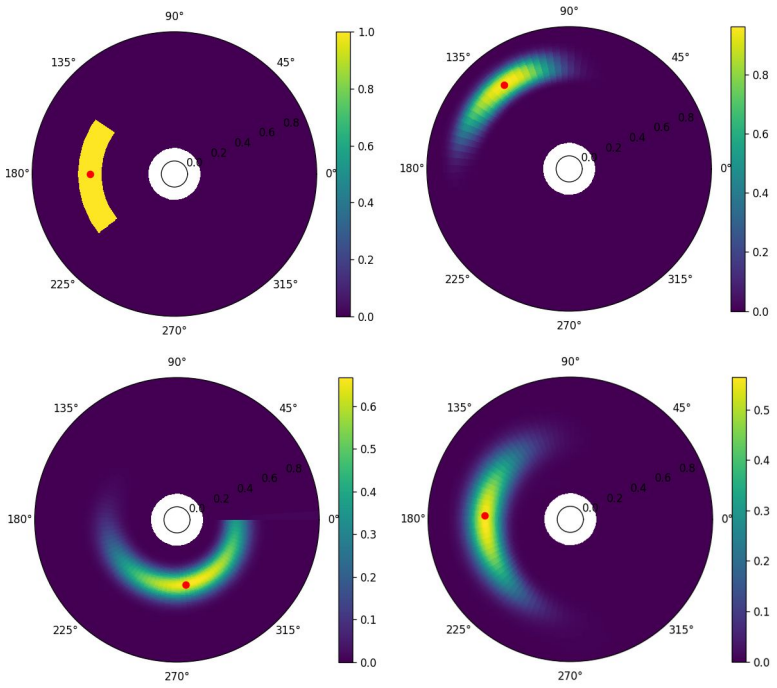

Figure 3: Polar coorindate diagonal advection test. Similar convection used to figure 2. Fluid is diagonally advected over a time t from the initial state (top left) to top right. The advection field is reversed for time $2t$ giving bottom left. The field is reversed again producing bottom right.

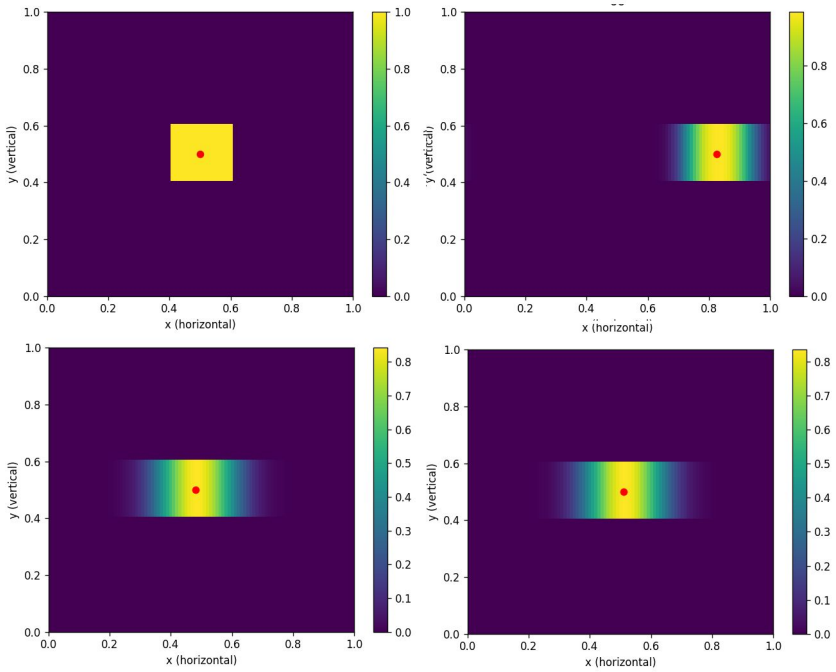

Figure 4: Cartesian coordinate advection test across a periodic boundary. Convention similar to figure 2.

3 (bottom left) near the azimuthal 0 to 360 boundary is a result of manually setting a streamfunction with a discontinuity there.

Thermal Lattice Boltzmann Tests

To test the validity of my thermal lattice Boltzmann code I replicated results from [3] in figure **??**. I used a "bounceback" boundary condition on all walls of the domain (figure 6 top four panes) which was compared against a simulation (figure 6 bottom) with the vertical walls having periodic boundary conditions.

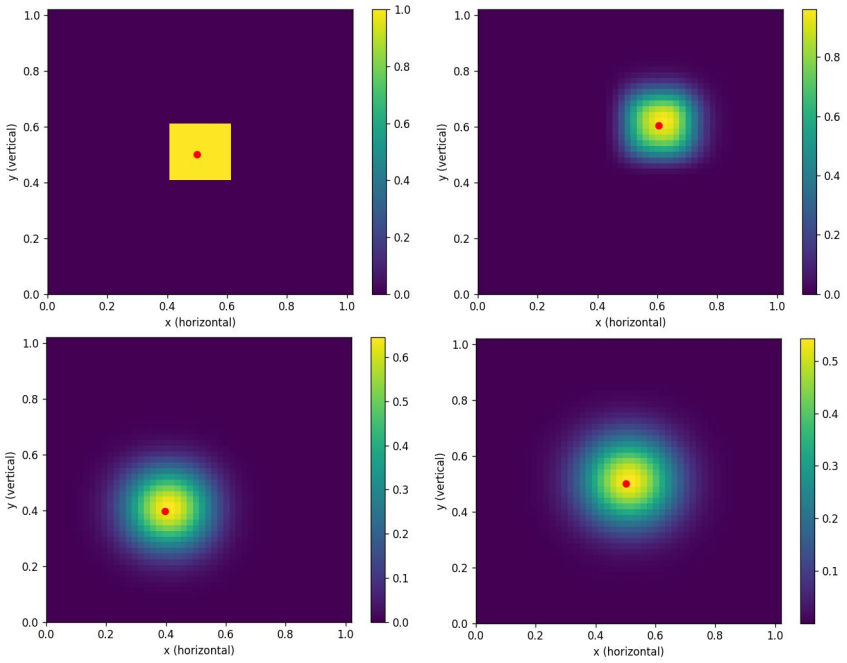

Figure 5: Cartesian diagonal advection test. Similar convention to figure 3.

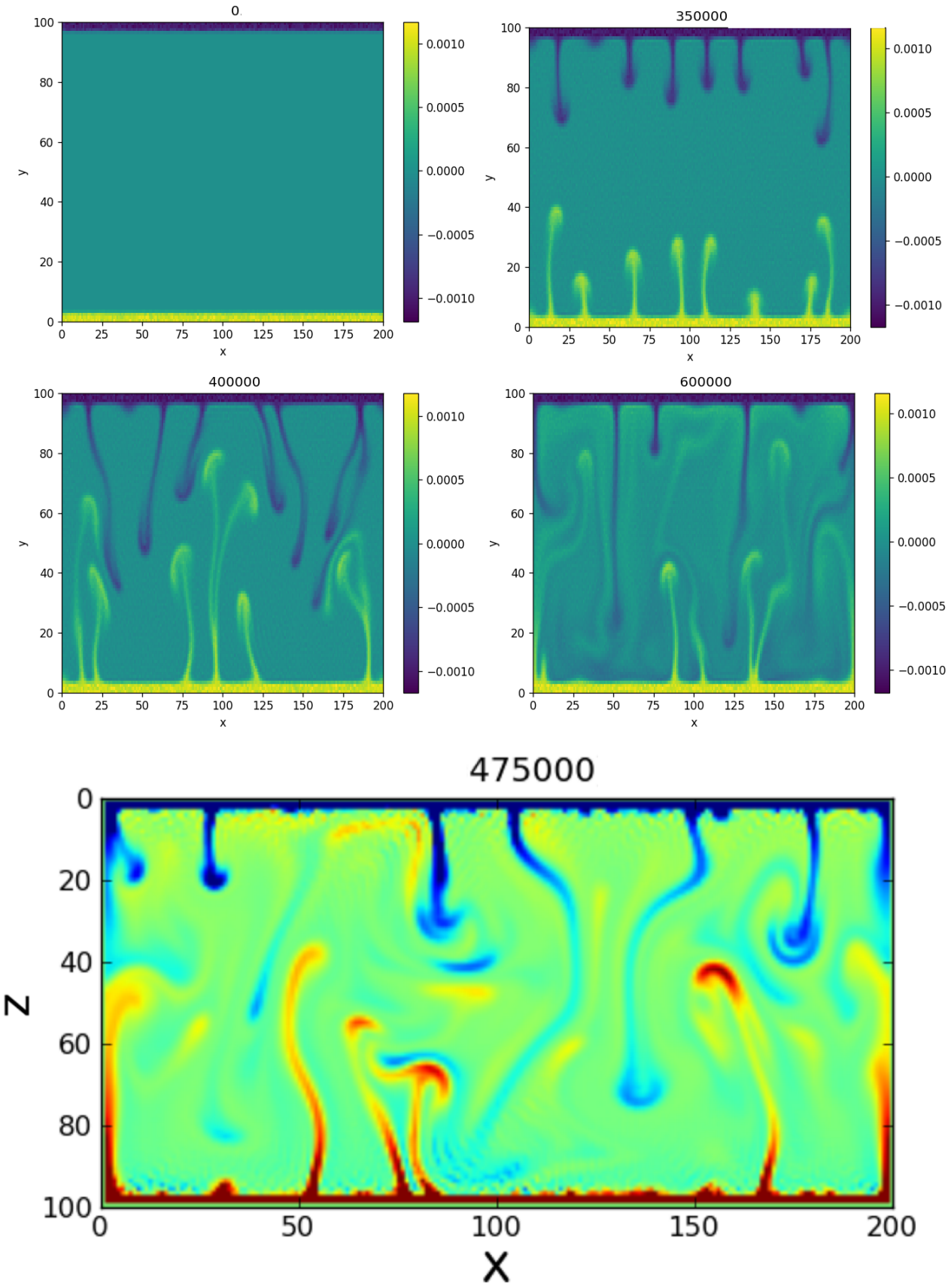

Figure 6: Comparison between Thermal Lattice Boltzmann coded here (top four figures) and Published Thermal Lattice Boltzmann results (bottom figure) [3]. Headings indicate timestep, red/yellow show hot regions while blue areas are cold. Top and bottom regions are held at constant temperature (no internal heating). All program settings used are similar. $Pr = 5000, Ra = 10^8$

Conservation of heat in the streamfunction code / tests for if the streamfunction code is working as it should / tests for how well my wacko boundary conditions match proper constant heat flux conditions

This section isn't done. I may not get time to do it. We will see how the time goes. Priority is:
1. check streamfunction reproduces results from other codes (might just check it against LBM) 2. check boundary conditions work 3. discussion of heat conservation.

Simulations

The Cartesian and polar streamfunction-vorticity codes are limited in two distinct ways. The polar geometry of the 2-dimensional inner core is well suited to polar coorindates. However, by using polar coorindates, a singularity in the governing equations occurs at the center $r \to 0$. For standard regularly spaced meshes as is used here, this causes the lattice spacing near the center to become infinitesimal requiring small timesteps and large amounts of compute time. For this reason, the inner region of the polar domain is excluded leaving out a critical region in the simulation. In Cartesian coorindates, there is no singularity and the full domain is included, however it becomes difficult to recreate the polar geometry and enforce boundary conditions. Because of this, we approximate the 2-dimensional inner core by "unrolling" the polar domain as shown in figure **??**, losing the affect of the polar geometry.

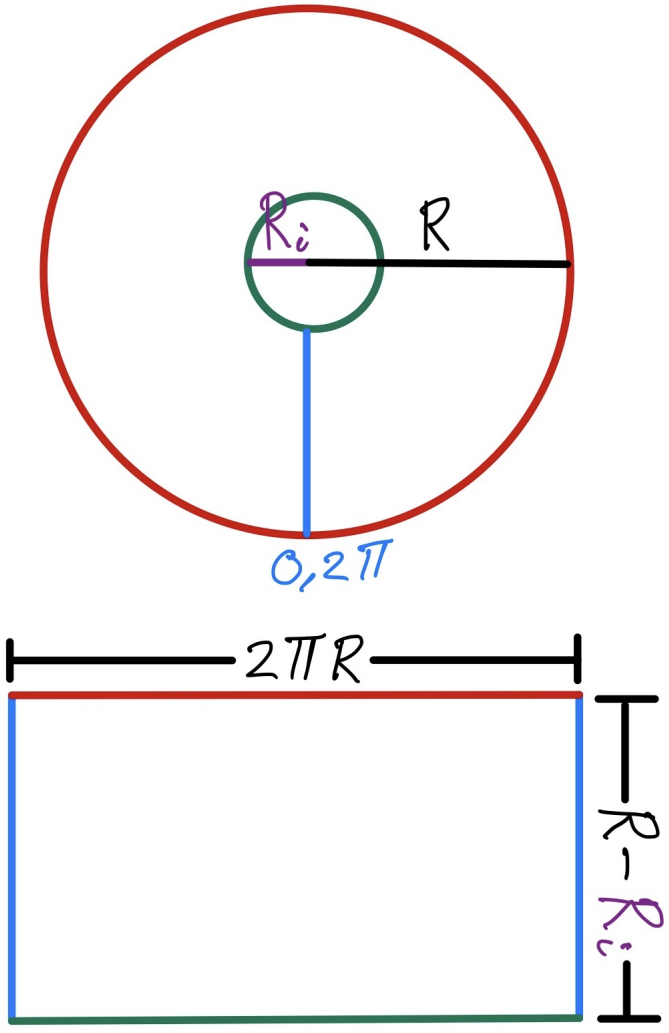

Figure 7: Figure showing how the 2-dimensional polar domain (top) is cut at the azimuthal angle $0, 2\pi$ boundary and stretched into the Cartesian geometry (bottom). Sides are color coded. R and R_i are the outer and inner radii of the polar domain.

Geometry and the Central region of a self-gravitating fluid

In this section, I present results from my streamfunction-vorticity codes in Cartesian and polar coorindates. I show through comparison between simulations that taking into account the geometry and center of a 2-dimensional self gravitating fluid are important and significantly change both the dynamics and long term convective patterns.

A gravitational field linearly proportional to height and radius for the Cartesian and polar codes respectively was set. A constant internal heating field H was set for all but the upper 10% of both domains. A compensating heating field was set at the remaining top of the domain to balance the total heat generation. Note this heat field is different in the Cartesian and polar cases. In all heating fields time-independent random fluctuations of $\approx 1\%$ were set to facilitate instabilities.
As seen in figure 9, the two geometries give vastly different convective behaviour. The Cartesian geometry (left) begins forming convective plumes at time ≈ 143000, long after the time 25000 when the polar (middle) model begins to convect. The convection patterns formed in the Cartesian model are consistent in time, forming the typical Rayleigh-Bernard convection cells, while the polar cases are much more erratic. The larger inner radius polar simulation (right) shows similar initial behaviour to the smaller internal radius polar simulation (middle), however, it initially forms higher frequency convective structures. Importantly, over large timescales, the typical wavelength for features in the Cartesian geometry is $3-5$ times longer. Therefore, in simulating a self-gravitating internally heated fluid such as the inner core accurately, the geometry and central region must be simulated. For traditional approaches which discretize a set of equations, this requires either complex spatial meshings or boundary conditions.

Thermal Lattice Boltzmann Simulations

In this section, I describe how I simulate a self-gravitating fluid including its central region while respecting the geometry of the problem. I then present my Thermal Lattice Boltzmann Results, the main results of the report and discuss them

We have seen from the streamfunction-vorticity codes that geometry and internal region alter convection significantly, however, neither streamfunction-vorticty code can incorporate both. Figures 10 and 11 show TLBM simulations for internally heated Rayleigh numbers between 10^8 and 10^4 for Prandtl numbers of $Pr = 5000$. A high Prandtl number was selected to show the applicability to geologic flows which operate in the larger Prandtl number limit. The heating field H for these is constant below a radius of 90% and the remaining 10% cooled so that the internal energy is constant. To stimulate convection, random fluctuations of 10% in the heating field were set as well as random fluctuations to the heating/cooling boundary. The initial state common to simulations in figures 10 and 11 is shown in figure 8.
Figures 10 and 11 show that internally heated convection in our model begins at a thermal Rayleigh number below $Ra_H = 10^5$. This lower bound is significantly higher than the critical Rayleigh number of $Ra \approx 650$. However, these this critical Rayleigh number does not account for the polar geometry or non-constant gravitational field simulated here. The thermal Rayleigh number (Ra_H) is also related to the Rayleigh number (Ra) by scaling arguments and so is not directly comparable. We also do not know if the lack of convection in small thermal Rayleigh numbers like $Ra = 10^4$ in figure 10 is due to physical constraints, or the fact that we only simulated a finite amount of time. In future, the TLBM program could be parallelised (which is easy with goLang) and run for longer to simulate more steps and better bound the critical Ra_H from above. The thickness of the cooling boundary layer used here is also not analysed, in future this could be done away with by using more complex thermal boundary conditions.
he characteristic wavelength of the convective patterns in figures 10 and 11 decreases with increasing thermal Rayleigh number (Ra_H). For small convecting thermal Rayleigh numbers, $Ra_H = 10^6, Ra_H = 10^5$ the convection wavelength is fairly consistent through time, while for higher values $Ra_H = 10^8, Ra_H = 10^7$ the wavelength is more dynamic. In general, the characteristic convection cells seen in Cartesian convection such as that in figure 9 (left bottom) is not seen. Instead, we see periodic plumes with skinny stalks and wide heads. These patterns are not static, best seen by the dumbbell-shaped oscillations in figure 11 (left) indicating that these systems have not reached equilibrium.

Difficulties using the Thermal Lattice Boltzmann Method

The TLBM coded here, uses a fixed unit timestep and lattice spaceing making the lattice speed $C = 1$. The TLBM remains stable as long as the macroscopic velocities remain a small fraction of the lattice speed. For internally heated convective problems, the time to heat/cool the fluid before convective plumes can develop can be significant compared to the timescale for plume motion, particularly at smaller or near critical Rayleigh numbers. Because of this, much simulation time is spent computing an uninteresting part of the solution. In future, the initial non-convective part of the solution could be skipped by initially setting the temperature of fluid regions according to simple diffusion models.
Extreme local heating or cooling was also an issue that would occur at high Rayleigh numbers $Ra > 10^9$ and induce velocities comparable to the lattice speed, causing instabilities to grow to infinity and crash the program.

Conclusion

Not yet done.

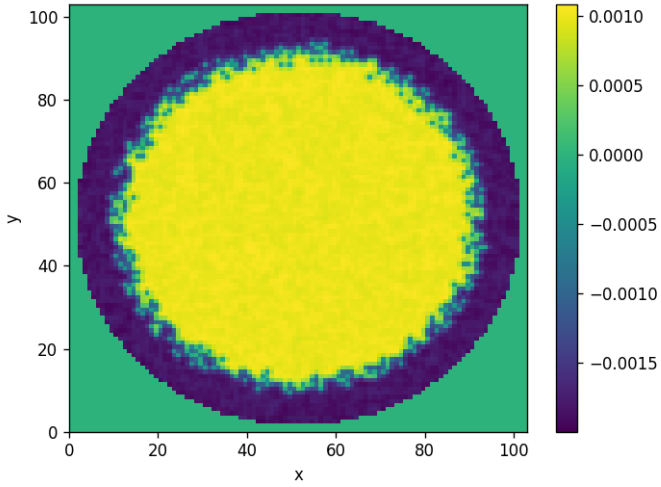

Figure 8: Example Initial state showing random fluctuations of the heating field of 10% throughout with random fluctuations across the hot/cold boundary.

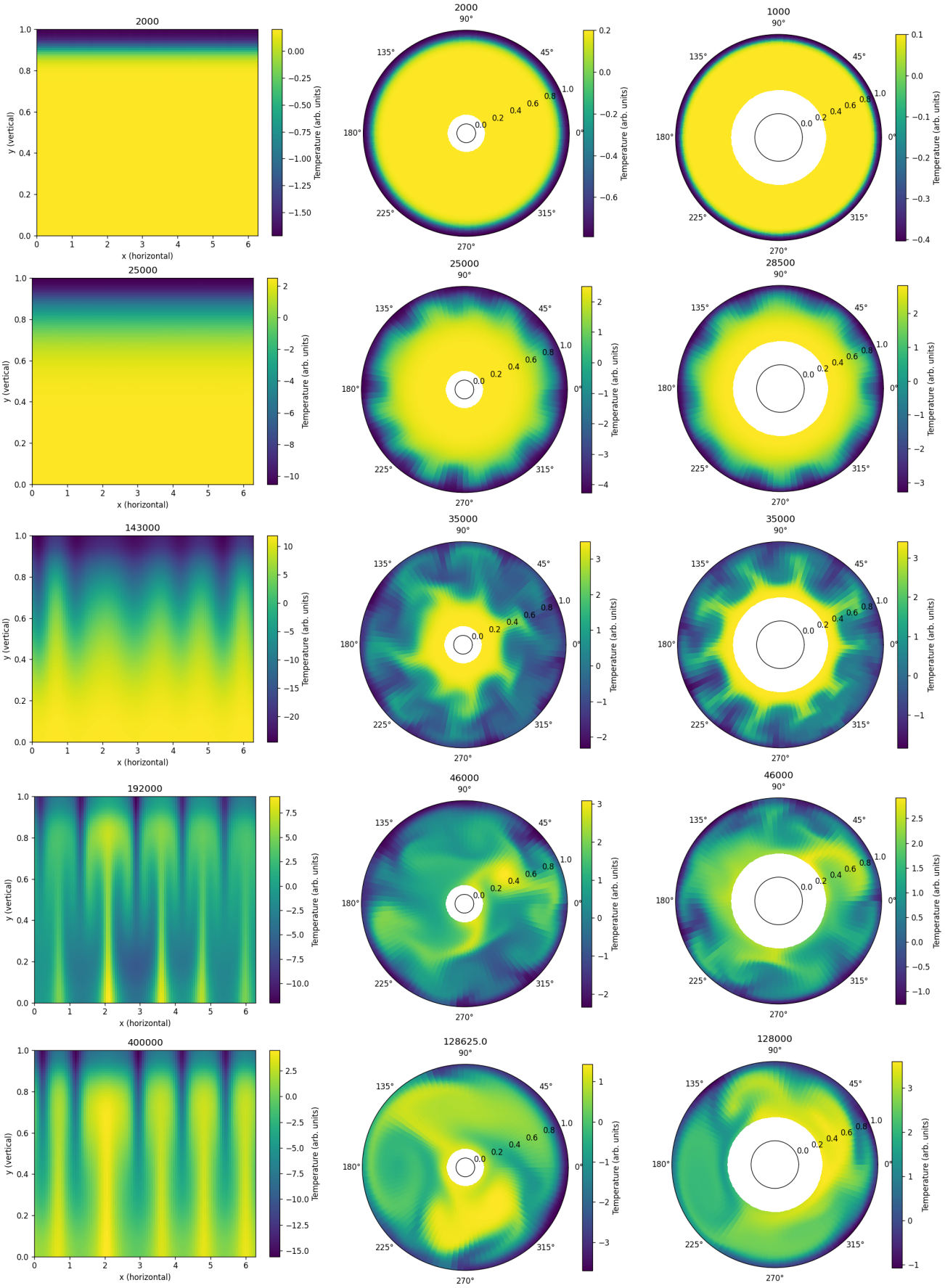

Figure 9: Simulations of internally heated, self-gravitating convection in Cartesian (left), polar with inner radius 0.1 (middle) and polar with inner radius 0.2 (right) coorindates. Times indicated by headings. Apart from inner radius, simulation settings are common. $Ra_H = 10^8, Pr = 5000$.

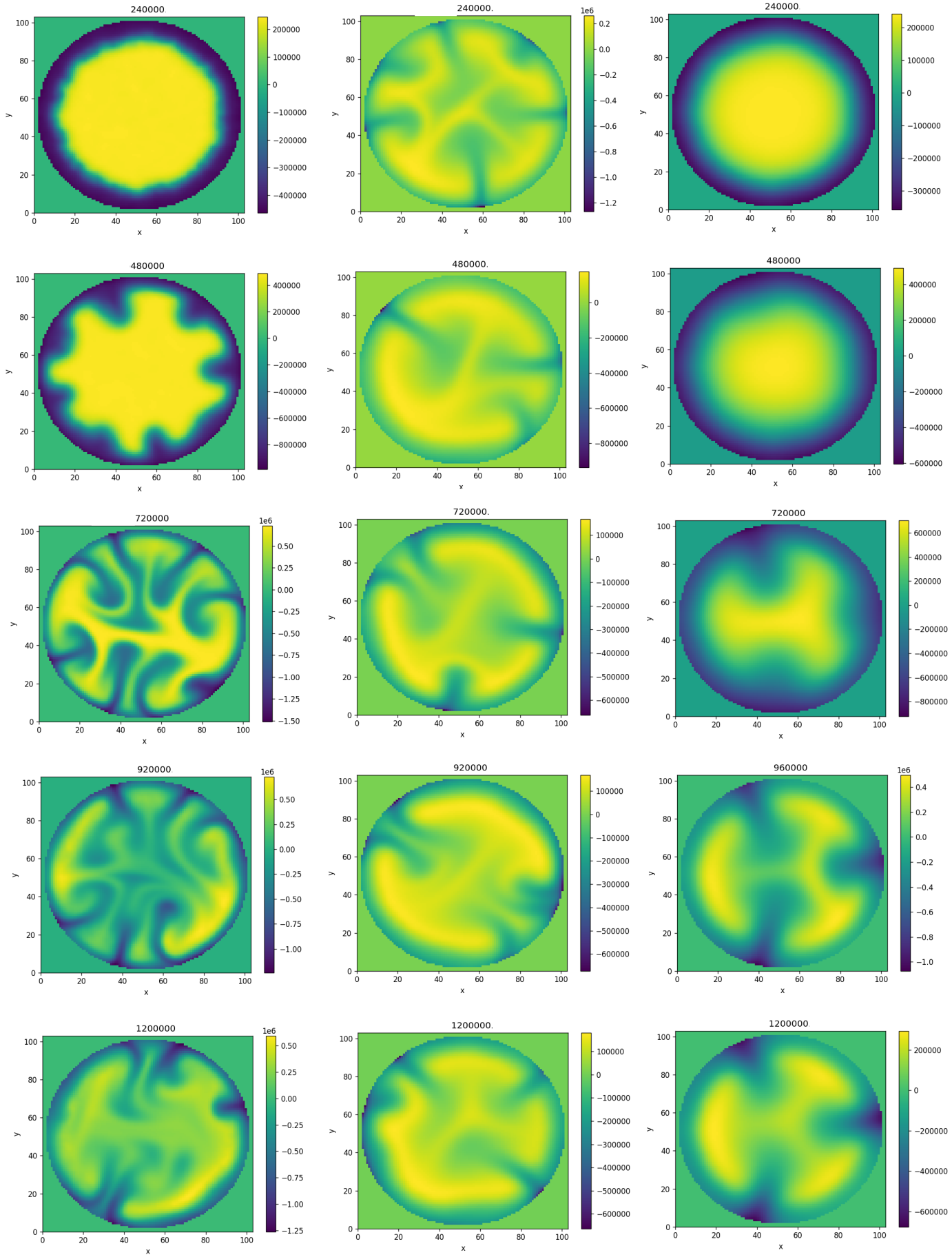

Figure 10: High Rayleigh number $Ra_H = 10^8$ (left column) $Ra_H = 10^7$ (middle column) $Ra_H = 10^6$ (right column) from Thermal Lattice Boltzmann simulations for an internally heated self-gravitating fluid with $Pr = 5000$. Headings are simulation timestep

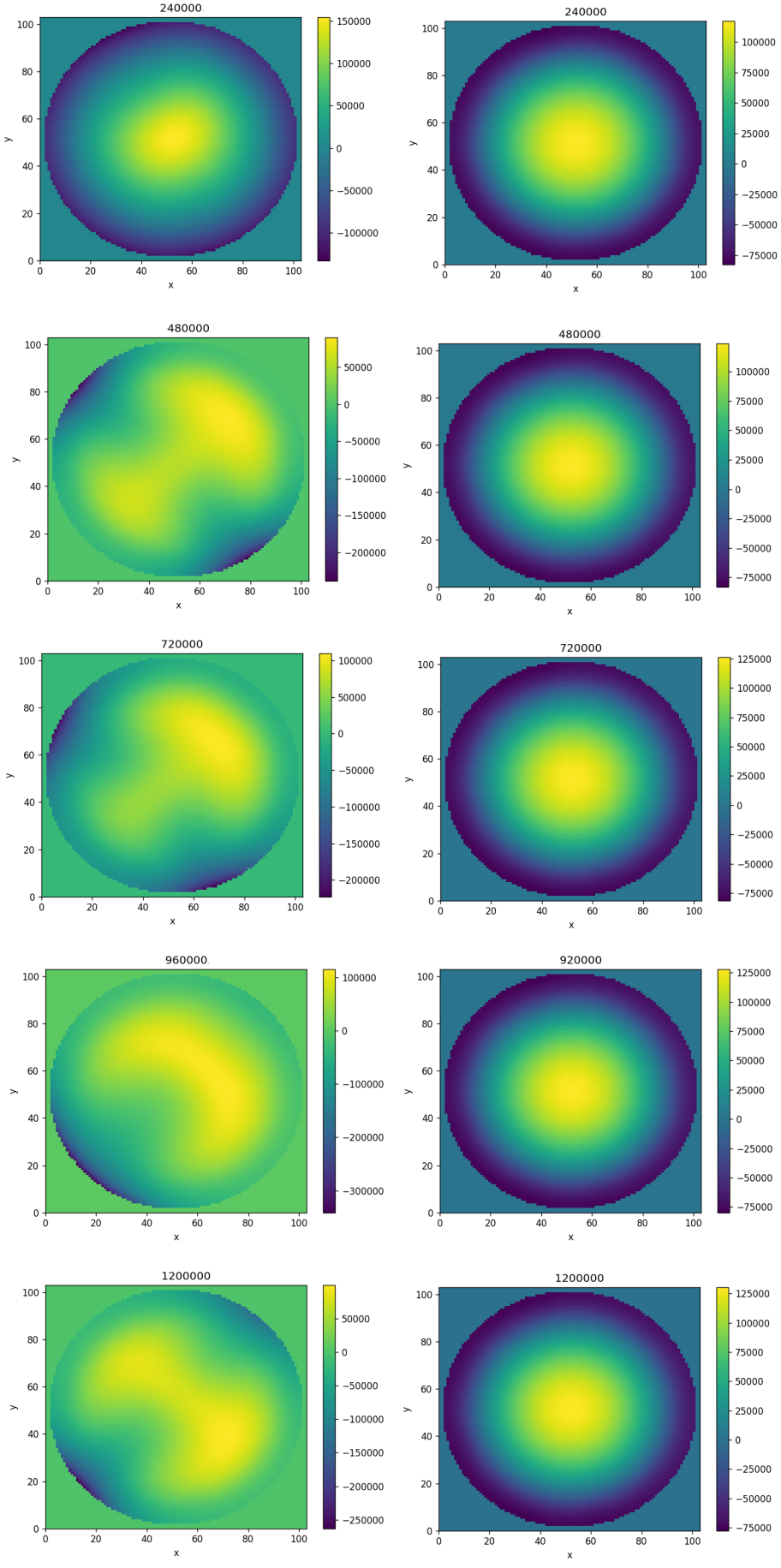

Figure 11: High Rayleigh number $Ra_H = 10^5$ (left column) $Ra_H = 10^4$ (right column) Thermal Lattice Boltzmann simulations for an internally heated self-gravitating fluid with $Pr = 5000$. Headings are simulation timestep

Appendix

Not done yet, will do at end.

Derivation of Gravitational Field in a self gravitating fluid

This is a short derivation, do it at the end.

Some Scales that I used

This is an important source here [1].
We chose the following legnth scales for the problem. For length we used d, the height or radius of the domain, for timescale we used $\frac{d^2}{\kappa}$, for pressure we used $\frac{\rho_0 d^2}{\kappa}$, for temperature we used $\frac{d^2 H}{\kappa}$. I took the rayleigh number as $Ra = \frac{g\alpha H d^5}{\nu\kappa^2}$ and the Prantl number $Pr = \frac{\nu}{\kappa}$. I additionally define a timescale for transport via flow:

$$\tau = \frac{\kappa\nu}{g\alpha H d^3} \tag{37}$$

References

[1] D. Goluskin. *Internally heated convection and Rayleigh-Bénard convection.* Springer, 2016.

[2] R. Khazaeli, M. Ashrafizaadeh, and S. Mortazavi. A ghost fluid approach for thermal lattice boltzmann method in dealing with heat flux boundary condition in thermal problems with complex geometries. 2015.

[3] P. Mora and D. A. Yuen. Simulation of plume dynamics by the lattice boltzmann method. *Geophysical Journal International*, 210(3):1932–1937, 2017.