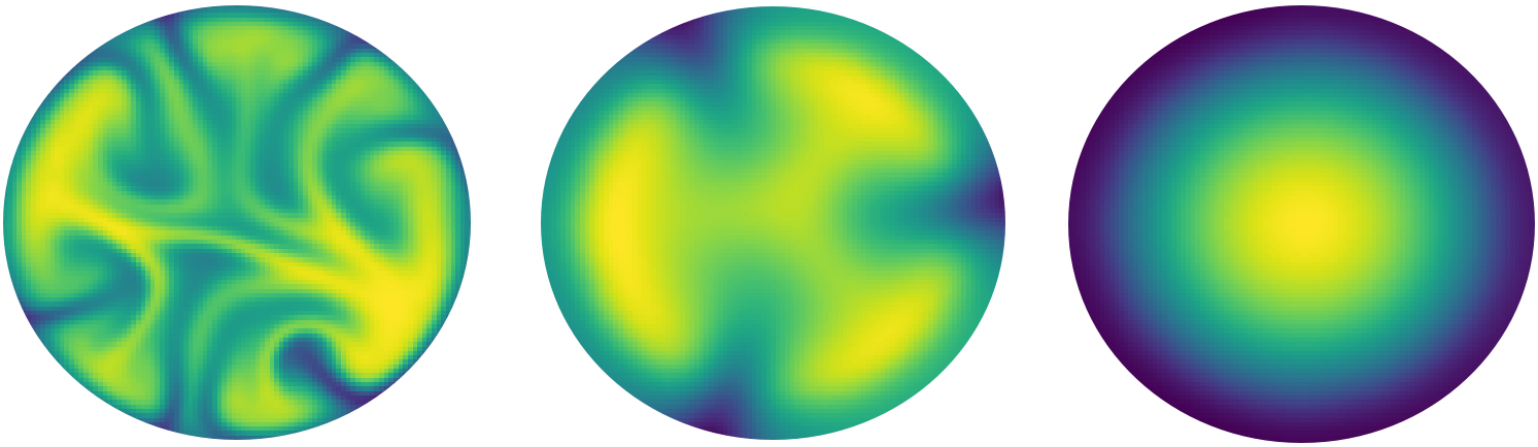
A Numerical Analysis of Convection in the Inner Core

Third Year Physics ASC

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

Maximilian Williams (u6338634)

October 2021



Abstract

Whether the Earth's Inner core convects or not has been a contentious topic ins geoscience. Recently, it has been proposed through seismic observations that Earth's inner core is convecting. Here we numerically model the Earth's inner core as a self-gravitating internally heated fluid in 2-dimensions using a streamfunction-vorticity method and a Thermal Lattice Boltzman approach coded from scratch in Python3 and goLang. By comparing results of two simple streamfunction-vorticity codes we conclude that the polar geometry of our 2-dimensional inner core model and the central region are important in modelling convection. Using a Thermal Lattice Boltzmann method (TLBM) we simulate the entire domain while respecting the geometry. Our TLBM simulations are in the high Prandtl number limit with $Pr = 5000$ and give an upper bound on the critical internally heated Rayleigh number of 10^5.

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 time and space dependent velocity $\vec{u}$ and density ρ. We assume that fluid is incompressible and its viscosity mu 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 the density variations are small and that the inner core is evolving on geologic timescales, allowing us to take the velocities $\vec{u}$ to be first order. These assumptions give 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 problem, the Prandtl number and the Rayleigh number. The Prandtl number Pr (equation 5) is the ratio of momentum and thermal diffusivity. WE also use 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:

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

Physical Picture

Here I talk about what I am modelling and my assumptions about the inner core. Throughout this report, we approximate the Earth's inner core as a perfect sphere and model a 2-dimensinal cross-section. We take the inner core as being internally heated except for a thin outer layer that is cooled. We ignore all affects of rotation or precession, assuming the core is in an inertial reference frame. We assume that the earth above the Inner core is spherically symetric in all respects and so model no variations in pressure or gravitation across the inner core's surface.

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 derivatives [1]. However, the streamfunction-vorticity method is limited to 2-dimensional and 3-dimensional symmetric flows and so has limited applications.

Here we take a 2-dimensional plane to simulate on with direction $\hat{z}$ pointing out of the plane. We define scalar vorticity ω by:

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

where the z subscript z is out of the simulation plane, and scalar streamfunction ψ implicitly by:

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

Given a coordinate system, and a clever definition of velocitites $\vec{u}$ 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: $$\frac{\partial T}{\partial t} + \frac{1}{r}\frac{\partial \psi}{\partial \theta}\frac{\partial T}{\partial r} - \frac{1}{r}\frac{\partial \psi}{\partial r}\frac{\partial T}{\partial \theta} = \kappa \nabla^2 T + H \qquad (14) \qquad \frac{\partial \omega}{\partial t} = -\frac{g_r}{\rho_0 r}\frac{\partial \rho'}{\partial \theta} + \nu \nabla^2 \omega.$$

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 to solve the streamfunction-vorticity-formulated governing equations in my streamfunction-vorticity codes is described.

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 at $r = 0$. We discretize time t by $t_n = n\Delta t$. For a function f, we use $f^n_{i,j}$ 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:

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

Second order space derivatives are approximated by a central difference: $$\frac{\partial^2 f_{i,j}}{\partial {x_1}^2} = \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{\Delta {x_1}^2}, \qquad (17$$

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:

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

and

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

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

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

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 [6].
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 the Jocobi Method for solving the Poisson equation, like 9 is introduced. This is the method used for solving 9 used in my streamfunction-vorticity codes.

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

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

Rearranging for ψ

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

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:

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

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 $\sum_{(i,j)\in\mathcal{D}'} \mid \psi^{(k+1)}_{i,j} - \psi^{(k)}_{i,j} \mid$ gets sufficiently small. A similar method is applied in polar coordinates.

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 code 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 to typical approaches which amount to directly numerically solving a set of partial differential equations. In the Thermal Lattice Boltzman method (TLBM) the internal energy is also simulated and used to simulate thermal affects which couple to the motion of the particles.
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 $\vec{e}_i$, $i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ [4]. In addition, each direction e_i gets a weight w_i [4]. 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

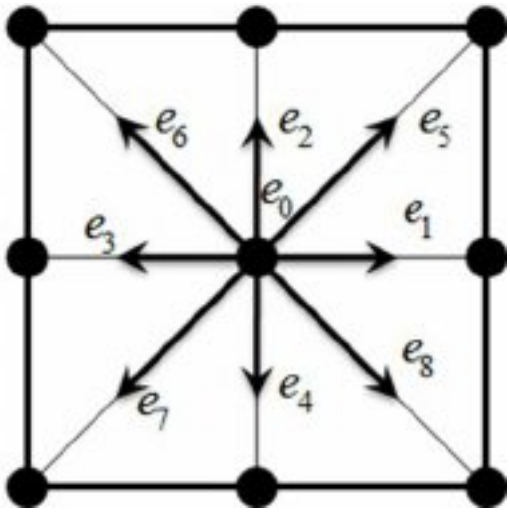


Figure 1: D2Q9 Lattice. Black nodes represent lattice points, vectors $e_0, .., e_8$ are the lattice vectors. Image sourced from [3]

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:

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

and a collision step:

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

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 [4]:

$$\tau_g = \frac{3\kappa}{S^2\Delta t} + \frac{1}{2}, \quad (29) \qquad \tau_f = \frac{3\nu}{S^2\Delta t} + \frac{1}{2}. \quad (30)$$

The equilibrium distributions are given by the BKG approximation [5]:

$$f_\alpha^{eq}(\vec{x}, t) = \rho w_\alpha(1 + 3\frac{\vec{e}_\alpha \cdot \vec{u}}{s^2} + \frac{9}{2}\frac{(\vec{e}_\alpha \cdot \vec{u})^2}{c^4} - \frac{3}{2}\frac{\vec{u} \cdot \vec{u}}{c^2}), \quad (31) \qquad g_\alpha^{eq}(\vec{x}, t) = \epsilon\rho w_\alpha(1 + 3\frac{\vec{e}_\alpha \cdot \vec{u}}{s^2} + \frac{9}{2}\frac{(\vec{e}_\alpha \cdot \vec{u})^2}{c^4} - \frac{3}{2}\frac{\vec{u} \cdot \vec{u}}{c^2}) \quad (32)$$

Here ρ, ϵ and $\vec{u}$ are the macroscopic density, internal energy and velocity given by [4]:

$$\rho = \sum_{i=0}^{8} f_i, \quad (33) \qquad \rho\vec{u} = \sum_{i=0}^{i=8} f_i\vec{e}_i \quad (34) \qquad \rho\epsilon = \sum_{i=0}^{i=8} g_i. \quad (35)$$

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:

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

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 [4]. The algorithm used to evolve the above TLBM equations is given in algorithm 1 and closely follows [4].

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

Boundary Conditions

In both streamfunction-vorticity and Thermal Lattice Boltzmann code, the boundaries are fluid-impermeable, non-slip and thermally insulating. In the streamfunction-vorticity code this is done by directly imposing that the $\psi = 0$ on the boundaries and manually setting tnagential velocities to zero. From this ω boundary conditions arrise. In the Thermal Lattice Boltzman code, these conditions are imposed by "bounce back" boundary conditions where during the streaming step (equation 25) distributions f and g are reflected off the boundaries. These boundary conditions were selected as they are easily implimented in both codes, particulaly the Lattice Boltzmann codes. 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.

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 **??**, **??**, **??**, **??**. A streamfunction ψ_c was enforced

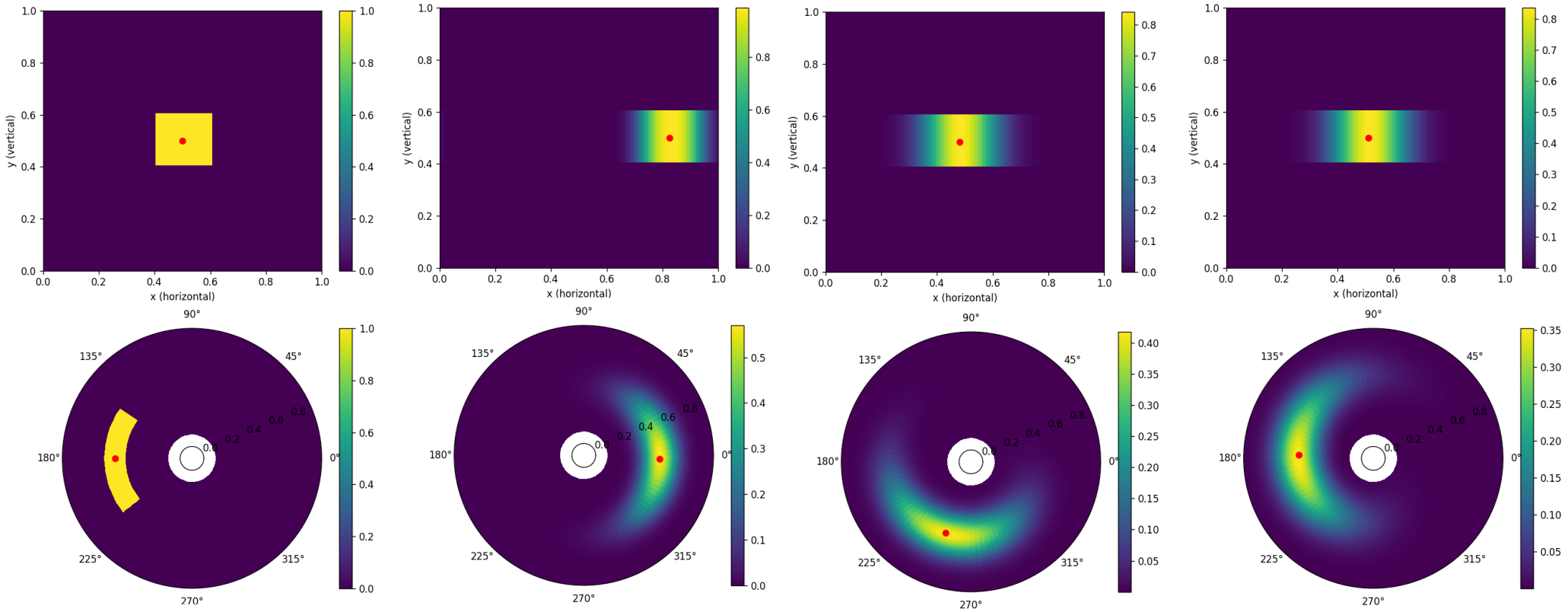


Figure 2: Cartesian (top) and polar (bottom) advection tests for horizontal (top) and azimuthal (bottom) advection. Advection fields move left to right (top) and clockwise (bottom). Red dot is centroid location, temperature indicated by colour. First column is the initial state, second column is crossing of periodic boundary conditions, third column is time when an exact scheme would return to the initial state, fourth column is advected field after returning to its initial location.

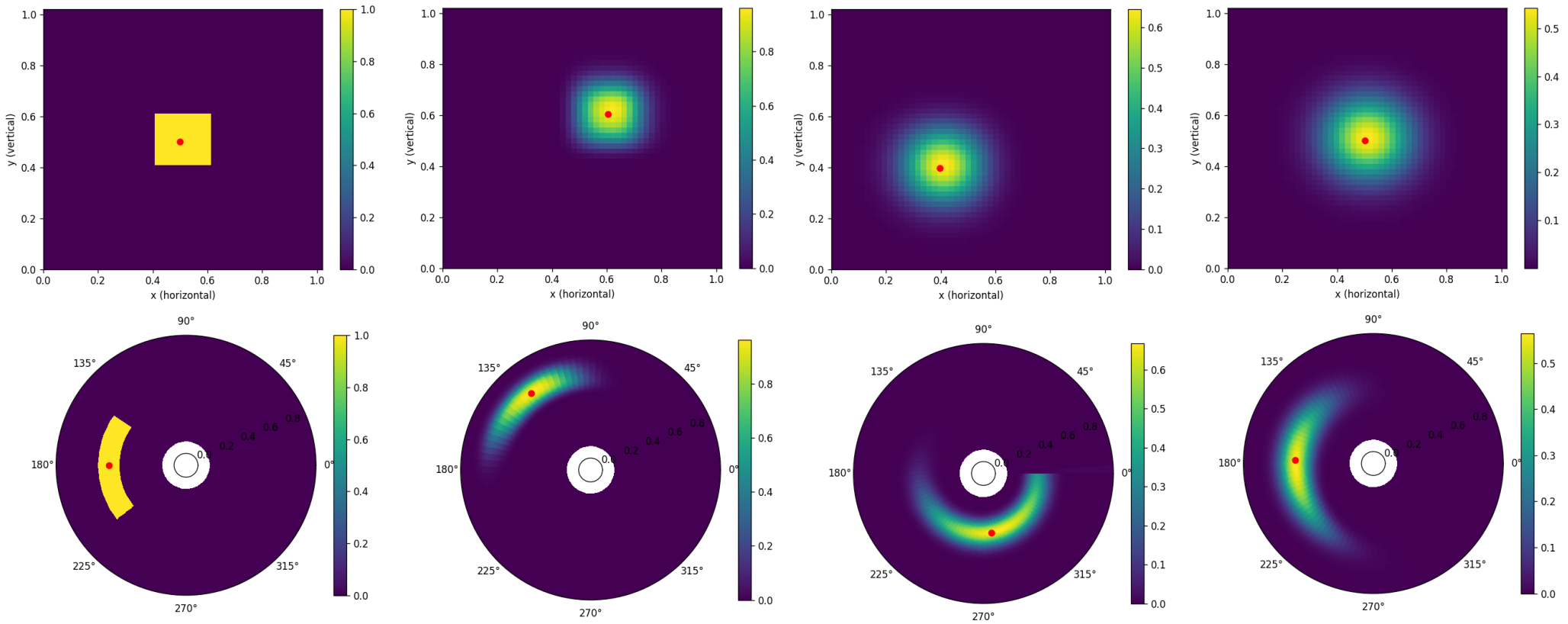


Figure 3: Cartesian (top) and polar (bottom) advection tests for advection back and forth along the secondary diagonal. Red dot is the centroid location, temperature indicated by colour. First column is the initial state, second column is one extreme of the diagonal, the third column is the other extreme and the final column is the state once the centroid has returned to its initial position.

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, top row figure 2 and 3 the temperature 1 region is advected across the periodic boundary conditions without distortion. The centroid (red dot) lagged the advection field by $\approx 1\%$ and $\approx 10\%$ in the horizontal and azimuthal advection case (figure 2) respectively. It is unknown exactly why these differe substantually, however a probable cause is the smaller timestep use to produce the polar tests. In all cases, the bluring shows numerical diffusion along the direction of travel which is predicted by equation **??** (equation not produced here). The discontinuity in temperature in the bottom row third column of figure 3 near the $0, 360$ azimuthal angle boundary is a direct result of manually setting the streamfunction.

Thermal Lattice Boltzmann Tests

To test the validity of my thermal lattice Boltzmann code I replicated results from [4] in figure **??**. I used a "bounceback" boundary condition on all walls of the domain (figure 5 top four panes) which was compared against a simulation (figure 5 bottom) with the vertical walls having periodic boundary conditions. In addition, I used random flucations in boundary temperature of 10% while [4] used random flucations of 25%. This explains the slower onset of instabilities in my code.

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, a substantual part of 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 while enforcing boundary conditions. Because of this, in the Cartesian case, we approximate the 2-dimensional inner core by "unrolling" the polar domain as shown in figure **??**, losing the polar geometry.

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 6, 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 8 and 9 show TLBM simulations for internally heated Rayleigh

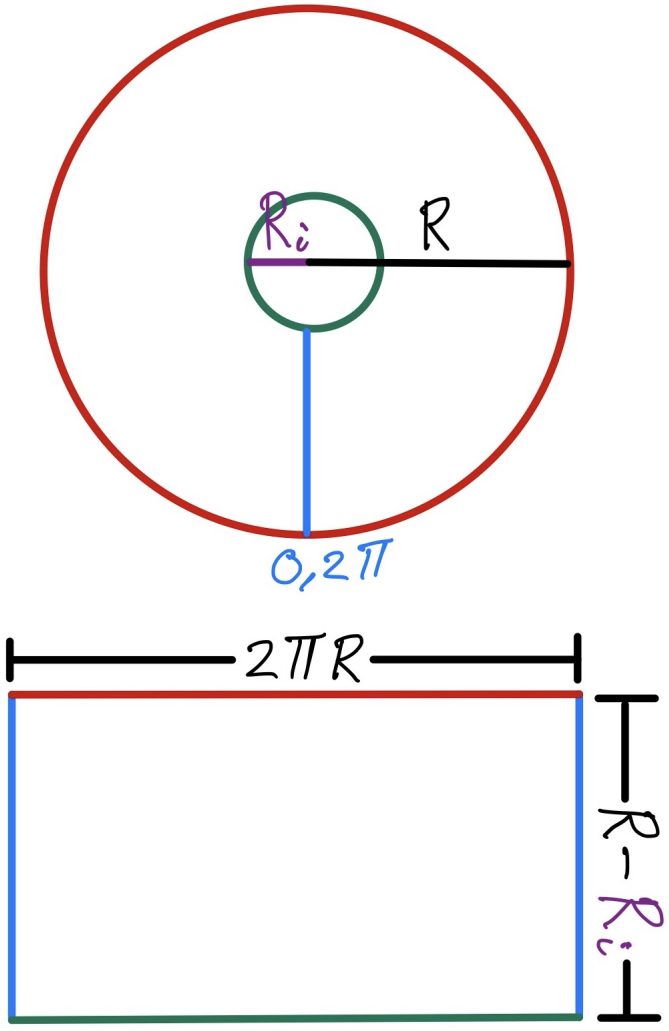


Figure 4: Figure showing how the 2-dimensional polar domain (top) is cut at the azimuthal angle $0, 2\pi$ boundary and stretched into the Cartesian geometry (bottom). Sides are color coded. R and R_i are the outer and inner radii of the polar domain.

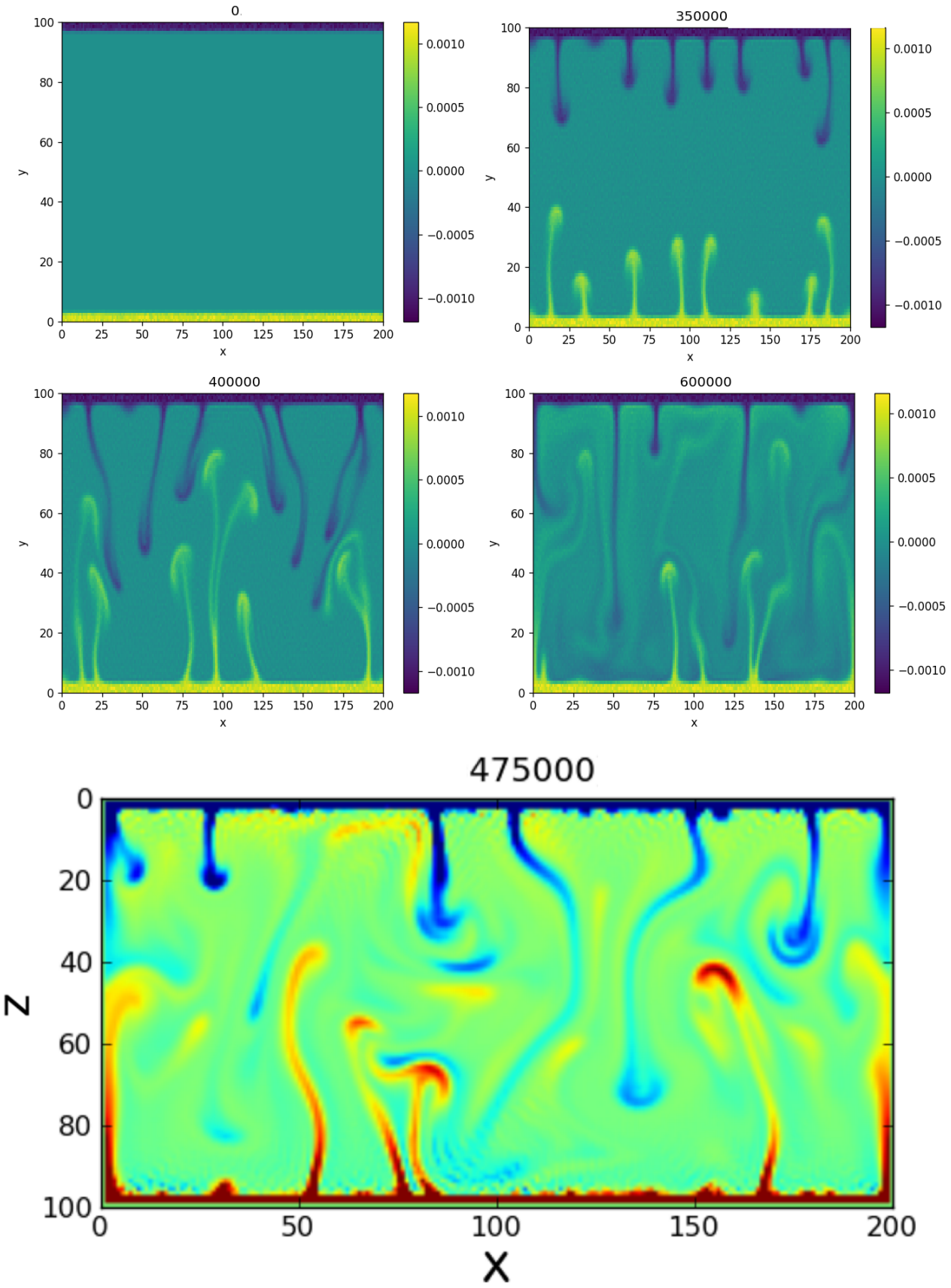


Figure 5: Comparison between Thermal Lattice Boltzmann coded here (top four figures) and Published Thermal Lattice Boltzmann results (bottom figure) [4]. Headings indicate timestep, red/yellow show hot regions while blue areas are cold. Top and bottom regions are held at constant temperature (no internal heating). All program settings used are similar. $Pr = 5000, Ra = 10^8$

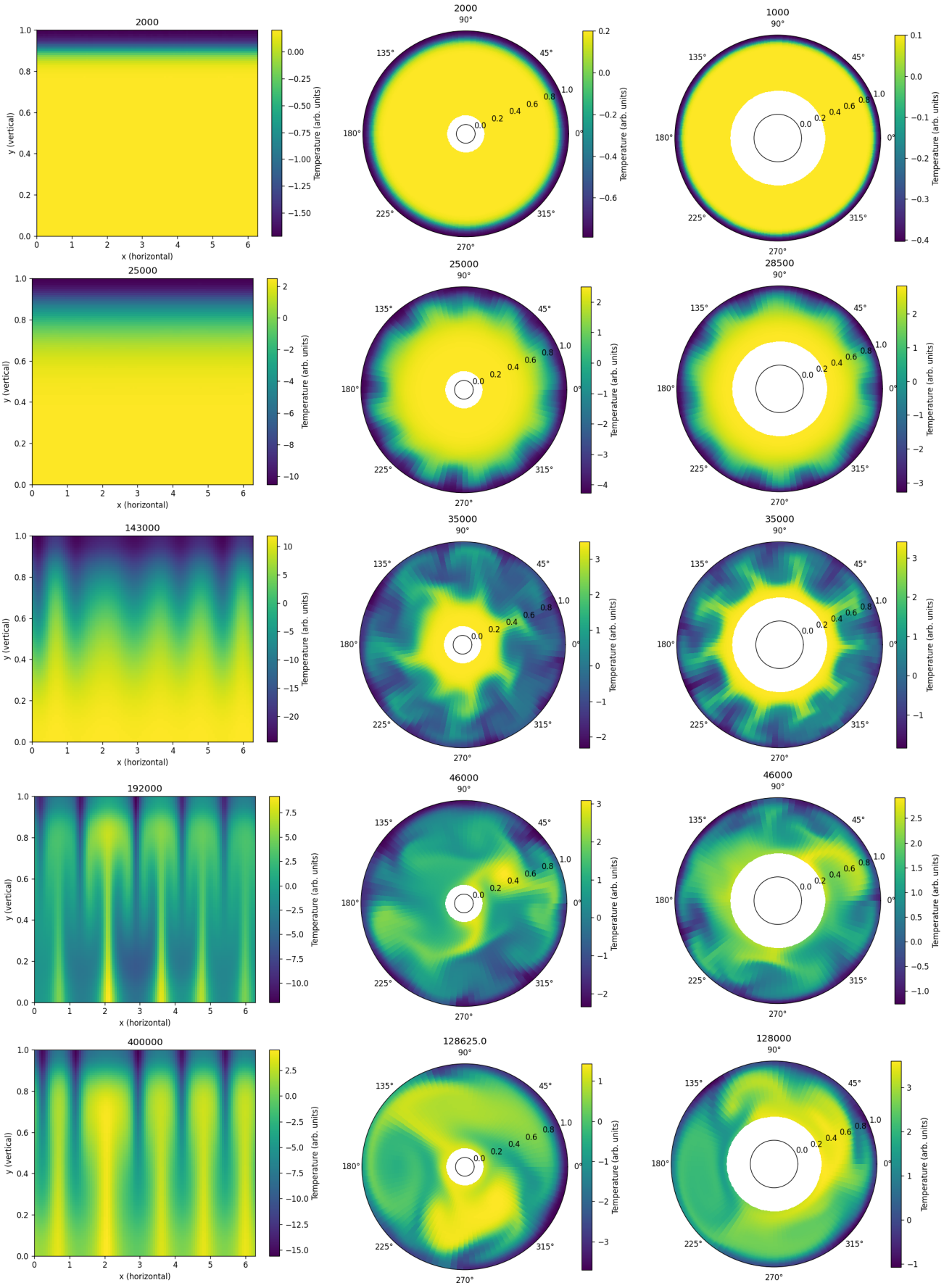


Figure 6: Simulations of internally heated, self-gravitating convection in Cartesian (left), polar with inner radius 0.1 (middle) and polar with inner radius 0.2 (right) coorindates. Times indicated by headings. Apart from inner radius, simulation settings are common. $Ra_H = 10^8, Pr = 5000$.

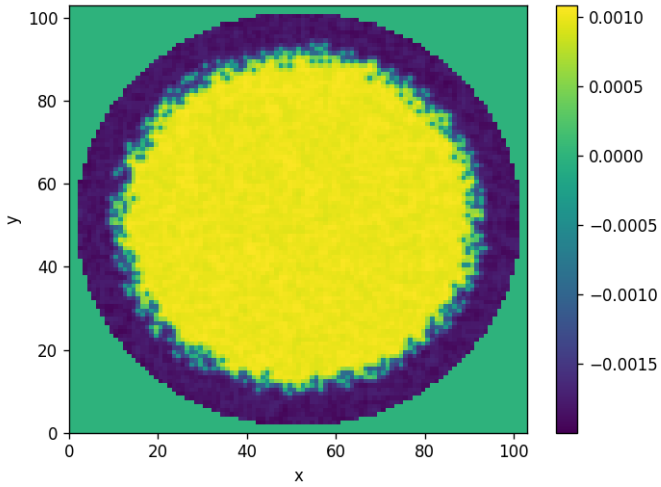


Figure 7: Example Initial state showing random fluctuations of the heating field of 10% throughout with random fluctuations across the hot/cold boundary.

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 8 and 9 is shown in figure 7.

Figures 8 and 9 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 8 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 8 and 9 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 6 (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 9 (left) indicating that these systems have not reached equilibrium.

Difficulties with 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

A 2-dimensional self-gravitating internally heated model of that inner core was pressented. Codes were developed from scratch in python3 and goLang to simulate convection in the 2-dimensional inner core model using streamfunction and Lattice Boltzmann approachs. The streamfunction codes based in Cartesian and polar geometries were compared showing that the geometry and central region of the inner core were important to accurately simulating internally heated convection, however neither code could account for both. Finally, results from the TLBM code which accounted for both geometry and the central region were pressented for high Prandtl number ($Pr = 5000$) giving an upper bound on the critical internally heated Rayleigh number of $Ra_H < 10^5$.

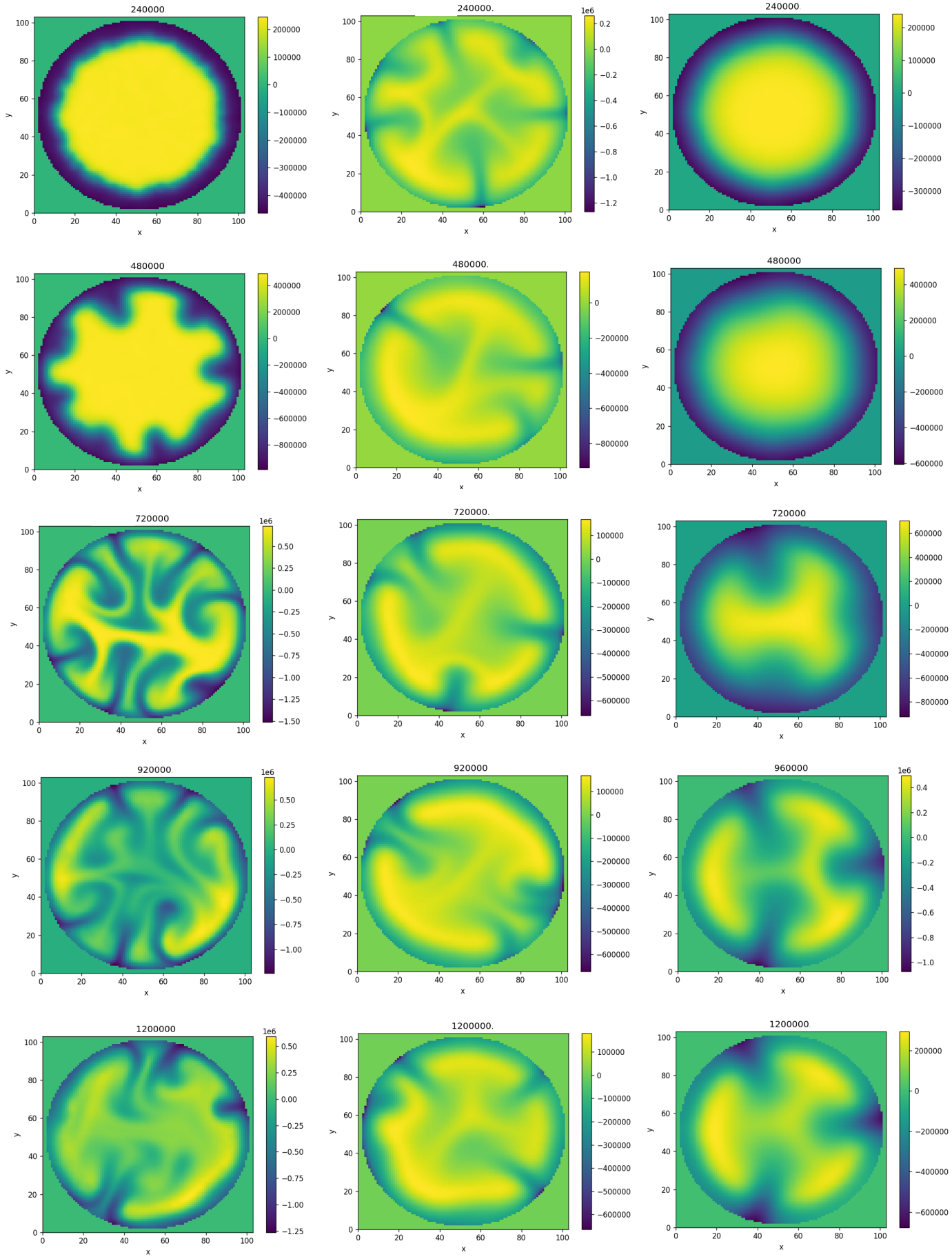


Figure 8: High Rayleigh number $Ra_H = 10^8$ (left column) $Ra_H = 10^7$ (middle column) $Ra_H = 10^6$ (right column) from Thermal Lattice Boltzmann simulations for an internally heated self-gravitating fluid with $Pr = 5000$. Headings are simulation timestep

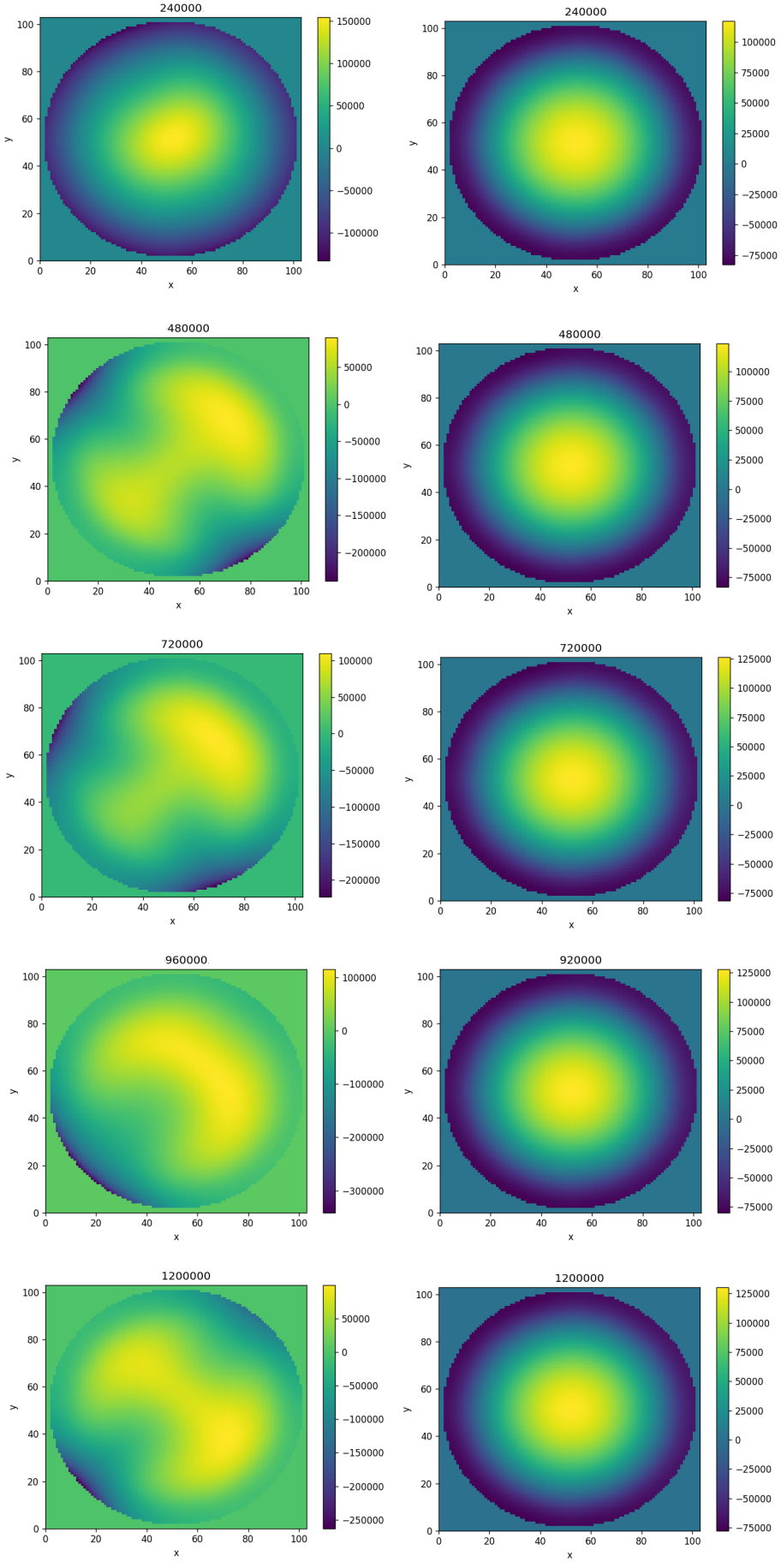


Figure 9: High Rayleigh number $Ra_H = 10^5$ (left column) $Ra_H = 10^4$ (right column) Thermal Lattice Boltzmann simulations for an internally heated self-gravitating fluid with $Pr = 5000$. Headings are simulation timestep

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 [2].
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] L. C. Earn, T. W. Yen, and T. L. Ken. The investigation on simple and simpler algorithm through lid driven cavity. *Journal of Advanced Research in Fluid Mechanics and Thermal Sciences*, 29(1):10–22, 2017.

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

[3] 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.

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

[5] Y.-H. Qian, D. d'Humières, and P. Lallemand. Lattice bgk models for navier-stokes equation. *EPL (Europhysics Letters)*, 17(6):479, 1992.

[6] M. Spiegelman and R. F. Katz. A semi-lagrangian crank-nicolson algorithm for the numerical solution of advection-diffusion problems. *Geochemistry, Geophysics, Geosystems*, 7(4), 2006.