

# Numerical Analysis of Convection in the Inner Core (DRAFT)

Maximilian Williams

September 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 Earth's inner core is convecting. Here we numerically model convection in the Inner core, using the streamfunction-vorticity formulation and lattice boltzman method in 2-dimensions.

## Introduction

Plan: I want to also talk about why this is actually important, why is this something that is worth studying

Here I want to introduce the physics of what I want to talk about. I want to introduce the basic equations that I will use.

## 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  $\rho$  that vary in space  $\vec{x}$  and time  $t$ . We assume the fluid's viscosity  $\mu$ , thermal diffusivity  $\kappa$  and specific heat capacity  $C_p$  are all constants. By conserving fluid momentum, we produce the Navier-Stokes equation:

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

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

$$\frac{DT}{Dt} = \kappa \nabla^2 T + \frac{Q}{C_v \rho}, \quad (2)$$

where  $Q$  is the heat generated per unit volume per unit time,  $C_v$  the specific heat at constant volume and  $\rho$  the fluid density. The density of the fluid  $\rho$  is assumed to vary linearly in temperature according to the equation of state:

$$\rho = \rho_0(1 - \alpha(T - T_0)), \quad (3)$$

where  $\alpha$  is the volumetric expansion coefficient and  $\rho_0$  the density at a reference temperature  $T_0$ . Through seismic imaging, variations in inner core density are  $\ll 1\%$ . We also assume that the inner cores evolution occurs over geologic timescales, and as such take  $\vec{u}$  to be first order. These assumptions allow us to make the slow flow boussinesq approximation to equation 1:

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

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

The Earths

## Numerical Methods

*Two numerical methods are introduced for solving the thermal convection problem, the Lattice Boltzman 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, as is used in the SIMPLE algorithm. However, the streamfunction-vorticity method is limited to 2-dimensional or 3-dimensional symmetric flows and so has limited applicability.

We define two scalar quantities, the vorticity  $\omega$  and the streamfunction  $\psi$ . The vorticity  $\omega$  is given by:

$$\omega = (\nabla \times \vec{u})_z, \quad (5)$$

where the  $z$  subscript denotes the component out of the plane. We also define a streamfunction  $\psi$  by:

$$\omega = -\nabla^2 \psi. \quad (6)$$

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

$$u = \frac{\partial \psi}{\partial y}, v = -\frac{\partial \psi}{\partial x}, \quad (7)$$

allowing us to write equations 2 and 4 as:

$$\frac{\partial T}{\partial t} + \frac{\partial \psi}{\partial y} \frac{\partial T}{\partial x} - \frac{\partial \psi}{\partial x} \frac{\partial T}{\partial y} = \kappa \nabla^2 T + \frac{Q}{\rho_0 C_v} \quad (8)$$

$$\frac{\partial \omega}{\partial t} = -\frac{g_y}{\rho_0} \frac{\partial \rho'}{\partial x} + \nu \nabla^2 \omega \quad (9)$$

Similary, in polar coordinates  $(r, \theta)$  we pick:

$$u = \frac{1}{r} \frac{\partial \psi}{\partial \theta}, v = -\frac{\partial \psi}{\partial r}, \quad (10)$$

giving:

$$\frac{\partial T}{\partial t} + \frac{1}{r} \frac{\partial \psi}{\partial \theta} \frac{\partial T}{\partial r} - \frac{1}{r} \frac{\partial \psi}{\partial r} \frac{\partial T}{\partial \theta} = \kappa \nabla^2 T + \frac{Q}{\rho_0 C_v} \quad (11)$$

and,

$$\frac{\partial \omega}{\partial t} = -\frac{g_r}{\rho_0 r} \frac{\partial \rho'}{\partial \theta} + \nu \nabla^2 \omega. \quad (12)$$

A full derivation of equations 8, 9, 11, 12 are given in appendix. Importantly, our definitions of  $u$  and  $v$  in equations 7 and 10 satisfy equation 5 and 6. Equations 6, 8, 9 for the cartesian case and 6, 11, 12 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  $\theta$ -periodic boundary conditions and an inner radius  $R_0$  in polar coordinates to avoid singularities generated by  $r = 0$ . We also discretize time  $t$  by  $t_n = n\Delta t$ . For a function  $f$ , we use  $f_{i,j}^n$  to mean  $f$  evaluated at time  $n$  at position  $(x_i, y_j)$  in cartesian coordinates or  $(r_i, \theta_j)$  in polar coordinates. To approximate derivatives we use a finite difference approach. All time derivatives are approximated by forward difference:

$$\frac{\partial f}{\partial t} = \frac{f^{n+1} - f^n}{\Delta t} \quad (13)$$

Second order space derivatives are approximated by a central difference:

$$\frac{\partial^2 f_{i,j}}{\partial x_1^2} = \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{\Delta x_1^2}, \quad (14)$$

$$\frac{\partial^2 f_{i,j}}{\partial x_2^2} = \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{\Delta x_2^2}, \quad (15)$$

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:

$$\frac{\partial f_{i,j}}{\partial x_1} = \frac{f_{i+1,j} - f_{i-1,j}}{2\Delta x_1}, \quad (16)$$

and

$$\frac{\partial f_{i,j}}{\partial x_2} = \frac{f_{i,j+1} - f_{i,j-1}}{2\Delta x_2}. \quad (17)$$

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

$$a \frac{\partial f_{i,j}}{\partial x_1} = \frac{1}{\Delta x} (|a| (\frac{1}{2} f_{i+1,j} - \frac{1}{2} f_{i-1,j}) - a (\frac{1}{2} f_{i+1,j} - f_{i,j} - \frac{1}{2} f_{i-1,j})). \quad (18)$$

This scheme is always downstream, 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-nicolson scheme.

To solve the streamfunction-vorticity equations we assume a starting vorticity  $\omega$  on our domain  $\mathcal{D}$ . We then apply the Jacobi method to solve equation 6 for  $\psi$  on the interior of the domain which we call  $\mathcal{D}'$ . Using  $\psi$  we update  $T$  on  $\mathcal{D}'$  using equation 8 (or 11 for polar). Finally,  $\omega$  is updated on  $\mathcal{D}'$  using equation 9 (12 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  $\psi$  explicitly in equation 6. Instead we use an iterative jacobi method. In cartesian coordinates, equation 6 is:

$$\omega_{i,j} = \frac{\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j}}{\Delta x^2} + \frac{\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}}{\Delta y^2} \quad (19)$$

Rearranging for  $\psi$

$$\psi_{i,j} = \frac{\Delta x^2 \Delta y^2}{2(\Delta x^2 + \Delta y^2)} \left( \frac{\psi_{i+1,j} + \psi_{i-1,j}}{\Delta x^2} + \frac{\psi_{i,j+1} + \psi_{i,j-1}}{\Delta y^2} + \omega_{i,j} \right). \quad (20)$$

We then use the result of  $\psi_{i,j}$  back into equation 20 to solve for  $\psi_{i,j}$  iterately as shown in equation 21:

$$\psi_{i,j}^{(k+1)} = \frac{\Delta x^2 \Delta y^2}{2(\Delta x^2 + \Delta y^2)} \left( \frac{\psi_{i+1,j}^{(k)} + \psi_{i-1,j}^{(k)}}{\Delta x^2} + \frac{\psi_{i,j+1}^{(k)} + \psi_{i,j-1}^{(k)}}{\Delta y^2} + \omega_{i,j} \right). \quad (21)$$

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  $\sum_{(i,j) \in \mathcal{D}'} |\psi_{i,j}^{(k+1)} - \psi_{i,j}^{(k)}|$  gets suffeciently small. A similar method is applied to the polar coordinate case.

### Solution Stability and Accuracy

#### Lattice Boltzman Method

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

The Lattice Boltzman 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 accoring 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 latitce, 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:

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

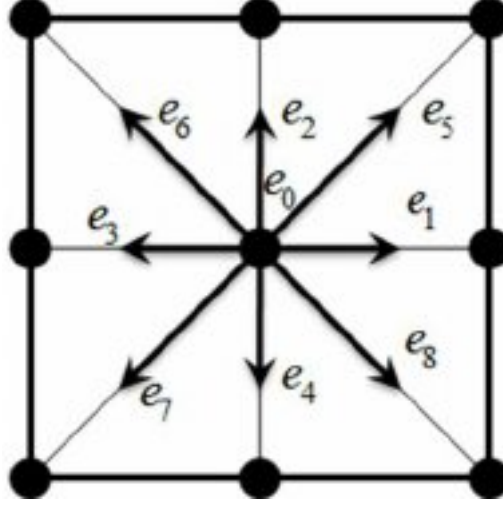


Figure 1: D2Q9 Lattice. Black notes represent lattice points, vectors  $e_0, \dots, e_8$  are the lattice vectors. Image sourced from [1]

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  $\alpha$  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 common streaming step:

$$f_\alpha(\vec{x} + \vec{e}_\alpha, t + \Delta t) = f_\alpha(\vec{x}, t), \quad (22)$$

and different collision steps:

$$f_\alpha(\vec{x} + \vec{e}_\alpha, t + \Delta t) = f_\alpha(\vec{x}, t) + \frac{1}{\tau_f}(f_\alpha^{eq}(\vec{x}, t) - f_\alpha(\vec{x}, t)) + F_\alpha \quad (23)$$

$$g_\alpha(\vec{x} + \vec{e}_\alpha, t + \Delta t) = g_\alpha(\vec{x}, t) + \frac{1}{\tau_g}(g_\alpha^{eq}(\vec{x}, t) - g_\alpha(\vec{x}, t)) + G_\alpha. \quad (24)$$

Here  $F_\alpha$  and  $G_\alpha$  are the forcing terms and  $f_\alpha^{eq}$  and  $g_\alpha^{eq}$  are equilibrium distributions. The relaxation times  $\tau_f$  and  $\tau_g$  are related to the macroscopic thermal diffusivity ( $\kappa$ ) and kinematic viscosity  $\nu$  by:

$$\tau_g = \frac{3\kappa}{S^2\Delta t} + \frac{1}{2} \quad (25)$$

and,

$$\tau_f = \frac{3\nu}{S^2\Delta t} + \frac{1}{2}. \quad (26)$$

The equilibrium distributions are given by the BKG approximation:

$$f_\alpha^{eq}(\vec{x}, t) = \rho w_\alpha \left( 1 + 3 \frac{\vec{e}_\alpha \cdot \vec{u}}{s^2} + \frac{9}{2} \frac{(\vec{e}_\alpha \cdot \vec{u})^2}{c^4} - \frac{3}{2} \frac{\vec{u} \cdot \vec{u}}{c^2} \right), \quad (27)$$

$$g_\alpha^{eq}(\vec{x}, t) = \epsilon \rho w_\alpha \left( 1 + 3 \frac{\vec{e}_\alpha \cdot \vec{u}}{s^2} + \frac{9}{2} \frac{(\vec{e}_\alpha \cdot \vec{u})^2}{c^4} - \frac{3}{2} \frac{\vec{u} \cdot \vec{u}}{c^2} \right), \quad (28)$$

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

$$\rho = \sum_{i=0}^8 f_i, \quad (29)$$

$$\rho \vec{u} = \sum_{i=0}^8 f_i \vec{e}_i \quad (30)$$

and,

$$\rho \epsilon = \sum_{i=0}^8 g_i. \quad (31)$$

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

$$\Delta f_\alpha = -w_\alpha \rho \alpha \epsilon \frac{\vec{e}_\alpha}{|\vec{e}_\alpha|} \cdot \vec{g} \frac{1}{\tau_{grav}}, \quad (32)$$

where  $||$  is the vector norm and  $\vec{g}$  is the gravitational acceleration. Here  $\tau_{grav}$  is the relaxation time for the gravitational field. We take  $\tau_g = 0.6$  here, following [2]. The algorithm used to evolve the above TLBM equations is given in algorithm 1

### Boundary Conditions

In both streamfunction-vorticity codes, boundaries were assumed to be fluid-impermeable, non-slip and insulating. In the Lattice Boltzman code, "bounce back" boundary condition was used. This is a fluid-impermeable, insulating and slip boundry.

## Advection and Heat Conservation

We next tested the accuracy of the advection schemes. Thermal diffusivity ( $\kappa$ ) and thermal expansion ( $\alpha$ ) were set to zero to avoid convection and a segment of the fluid was set to 1 unit of temperature with all other regions set to 0 units of temperature. In the first test, the streamfunction was set so that the velocity field was purely horizontal and azimuthal in the cartesian and polar cases respectively.

The second set of tests moved the fluid in a diagonal motion over the domain bringing it back to its origin state.

I will also have a test of heat conservation. Here. This will be simple, just a few pictures of the codes running (show some nice plumes) and the energy vs time graph. It shows that neither of these conserve thermal energy.

## Results

## References

- [1] 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.



shutterstock.com • 1188115291

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. (a) shows the initial condition, (b) shows the fluid being advected through the periodic boundary condition, (c) the analytic result for when the fluid would return to its initial state and (d) the final state of the fluid. Times are given above each pane.



shutterstock.com · 1188115291

Figure 3: Azimuthal advection test. Temperature (color) is plotted in polar space.





shutterstock.com • 1188115291

Figure 4: Cartesian diagonal 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.

---

**Algorithm 1** Thermal Lattice Boltzman Algorithm

---

```
 $f_\alpha = \rho_0 w_\alpha$   
 $\epsilon = 0$   
 $\epsilon_b$   
 $\rho = 0$   
 $\vec{u} = 0$   
while Simulation Running do  
   $g_\alpha \leftarrow \rho \epsilon_b w_\alpha$   
  if  $x - \vec{e}_\alpha$  is solid then  
     $f_\alpha(x) = f_{\alpha'}(x)$   
     $g_\alpha(x) = g_{\alpha'}(x)$   
  else  
     $f_\alpha(x) = f_\alpha(x - \vec{e}_\alpha)$   
     $g_\alpha(x) = g_\alpha(x - \vec{e}_\alpha)$   
  end if  
   $\rho = \sum_{i=0}^{i=8} f_i$   
   $\vec{u} = \frac{(\sum_{i=0}^{i=8} f_i \vec{e}_i)}{\rho}$   
   $\epsilon = \frac{\sum_{i=0}^{i=8} g_i}{\rho}$   
   $f_\alpha^{eq} = \rho w_\alpha (1 + 3 \frac{\vec{e}_\alpha \cdot \vec{u}}{s^2} + \frac{9}{2} \frac{(\vec{e}_\alpha \cdot \vec{u})^2}{c^4} - \frac{3}{2} \frac{\vec{u} \cdot \vec{u}}{c^2})$   
   $g_\alpha^{eq}(\vec{x}, t) = \epsilon \rho w_\alpha (1 + 3 \frac{\vec{e}_\alpha \cdot \vec{u}}{s^2} + \frac{9}{2} \frac{(\vec{e}_\alpha \cdot \vec{u})^2}{c^4} - \frac{3}{2} \frac{\vec{u} \cdot \vec{u}}{c^2})$   
   $\Delta f_\alpha = -w_\alpha \rho \alpha \epsilon \frac{\vec{e}_\alpha}{|\vec{e}_\alpha|} \cdot \vec{g}$   
   $\tau_g = \frac{3\kappa}{S^2 \Delta t} + \frac{1}{2}$   
   $\tau_f = \frac{3\nu}{S^2 \Delta t} + \frac{1}{2}$   
   $f_\alpha = f_\alpha + \frac{f_\alpha^{eq} - f_\alpha}{\tau_f} + \Delta f_\alpha$   
   $g_\alpha = g_\alpha + \frac{g_\alpha^{eq} - g_\alpha}{\tau_g}$   
end while
```

---

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