NUMERICAL SOLUTIONS TO THE RAYLEIGH-BERNARD PROBLEM

G. NAVARRO

1. Introduction

We will study the problem of a 2D box of fluid heated from below in a gravitational field. It is known that this situation is unstable, and that the instability appears in the form of rectangular convection cells denominated *Bernard cells*, so it will be of interest of this project to simulate this behavior by solving numerically the nonlinear Navier-Stokes equations.

1.1. **Equations.** In this section we describe the fundamental equations that describe the phenomena. First we have the incompressible Navier-Stokes equations,

$$\nabla \cdot \mathbf{v} = 0, \tag{1.1a}$$

$$\rho_0 \frac{D\mathbf{v}}{Dt} = -\hat{\mathbf{j}}(\rho - \rho_0)g - \nabla p + \mu \nabla^2 \mathbf{v}, \tag{1.1b}$$

with the equation of state,

$$\rho = \rho_0[1 - \alpha(T - T_0)], \tag{1.1c}$$

where α is the coefficient of volume expansion, and T_0 is the temperature at which the fluid density is ρ_0. Since for typical fluids $\alpha \approx 10^{-4}$, we will assume that if the temperature variations are not too large, ρ can be considered constant everywhere except in the buoyant force term. Finally we have the energy equation,

$$\frac{DT}{Dt} = \kappa \nabla^2 T \tag{1.1d}$$

where $\kappa := k/\rho_0 c_v$ is the coefficient of thermometric conductivity and c_v is the constant-volume specific heat.

1.2. **Geometry.** We consider a 2D fluid contained in a square box of size H.

1.3. **Initial and boundary conditions.** Initially, the fluid is at rest ($\mathbf{v} \equiv 0$), with uniform temperature T_0, and uniform density ρ_0. For the boundary conditions of the box, we fixed the temperature: $T = T_1$ at $y = 0$, and $T = T_0$ at $y = H$, and the normal component of the velocity vector is 0 ($\mathbf{v} \cdot N = 0$) on all the walls of the box.

1.4. **Normalized equations.** Defining the new variables,

$$X = \frac{x}{H},\ Y = \frac{y}{H},\ T = \frac{\mu}{\rho_0 H^2} t,$$

$$\mathbf{V} = \frac{\rho_0 H}{\mu} \mathbf{v},\ \theta = \frac{T - T_0}{T_1 - T_0},\ P = \frac{\rho_0 H^2}{\mu^2} p,$$

and taking *curl* of (1.1b), the equations take the form,

$$U = \frac{\partial \Psi}{\partial Y}, \tag{1.2}$$

$$V = -\frac{\partial \Psi}{\partial X}, \tag{1.3}$$

$$\Omega = -\left(\frac{\partial^2}{\partial X^2} + \frac{\partial^2}{\partial Y^2}\right)\Psi, \tag{1.4}$$

$$\frac{\partial \Omega}{\partial T} + \frac{\partial}{\partial X}(U\Omega) + \frac{\partial}{\partial Y}(V\Omega) = [Gr]\frac{\partial \theta}{\partial X} + \left(\frac{\partial^2}{\partial X^2} + \frac{\partial^2}{\partial Y^2}\right)\Omega, \tag{1.5}$$

$$\frac{\partial \theta}{\partial T} + \frac{\partial}{\partial X}(U\theta) + \frac{\partial}{\partial Y}(V\theta) = [Pr]^{-1}\left(\frac{\partial^2}{\partial X^2} + \frac{\partial^2}{\partial Y^2}\right)\theta, \tag{1.6}$$

where $Gr := \frac{\alpha g(T_1 - T_0)\rho^2 H^3}{\mu^2}$, $Pr := \frac{\mu}{\rho_0 \kappa}$ are the *Grashof* and *Prandtl* numbers respectively, $\Omega = curl(\mathbf{V})$ is the vorticity, and Ψ is the *stream function*.

1.5. **Numerical scheme.** We used a standard cartesian grid of $n \times n$ rectangles to discretize the rectangular domain. Spacial derivatives were computed using central-differences, with the exception of the vorticity Ω on the boundary, where we used instead a one-sided approximation using 3 points for the right hand side of equation (1.4).

The iteration scheme worked as follows. At any instant of time t the velocity vector $(U(t), V(t))$ was computed using the stream function of the previous time $\Psi(t - \Delta t)$, where the initial step is given by the initial data $\Psi(0) \equiv 0$. Then, the boundary values for $\Omega(t)$ where computed using equation (1.4) for $\Psi(t - \Delta t)$ restricted to the boundary. Afterwards the evolution equations (1.5, 1.6), gives us the next values for $\Omega(t)$, $\theta(t)$ in the interior of the domain. Finally we recover the stream function in the next step $\Psi(t)$ by inverting the Laplacian from equation (1.4) with the new function $\Omega(t)$, to then re-insert $\Psi(t)$ towards the next iteration.

2. Results

2.1. **Vector field.** Using the value of the *Prandtl* number for water $Pr = 6.75$, and *Grashof* number of $Gr = 1$, we plotted the velocity of the fluid using the *quiver* function in Matlab. Initially the fluid is at rest, but already at time $t = 0.01$, convection starts to push down the fluid along the corners where the cold walls meet the hot wall, which induces fluid to move horizontally over the bottom boundary and into a clockwise/counter clockwise motion in the right and left corner respectively (see top right of Figure 1). At time $t = 0.1$, the convective cells are already shaping up, with the fluid raising in the middle and falling through the sides. At $t = 0.5$, the cells are almost consolidated, with the fluid in the left side spining counter clockwise, and in the right spining clockwise.

2.2. **Stream function.** A similar analysis can be done for the stream function Ψ. At time $t = 0.01$ a small bubble of negative value starts to form in the left bottom corner of the previously still fluid (see Figure 2). This starts to push the positive part of the stream function into a symmetrical configuration already at time $t = 0.1$, to continue their evolution into a stable configuration with negative values in the left box, and positive values on the right box.

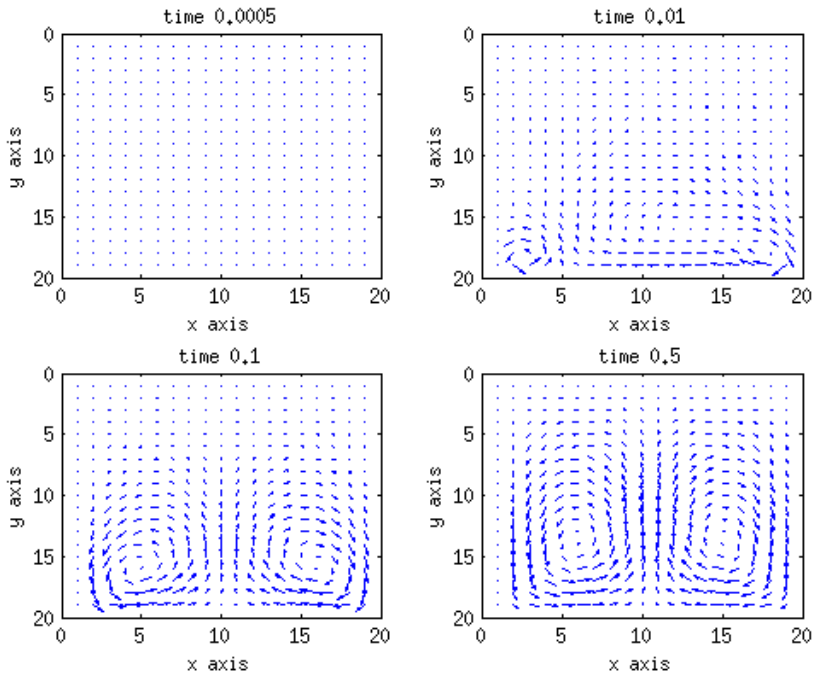

FIGURE 1. The velocity vector field for different times

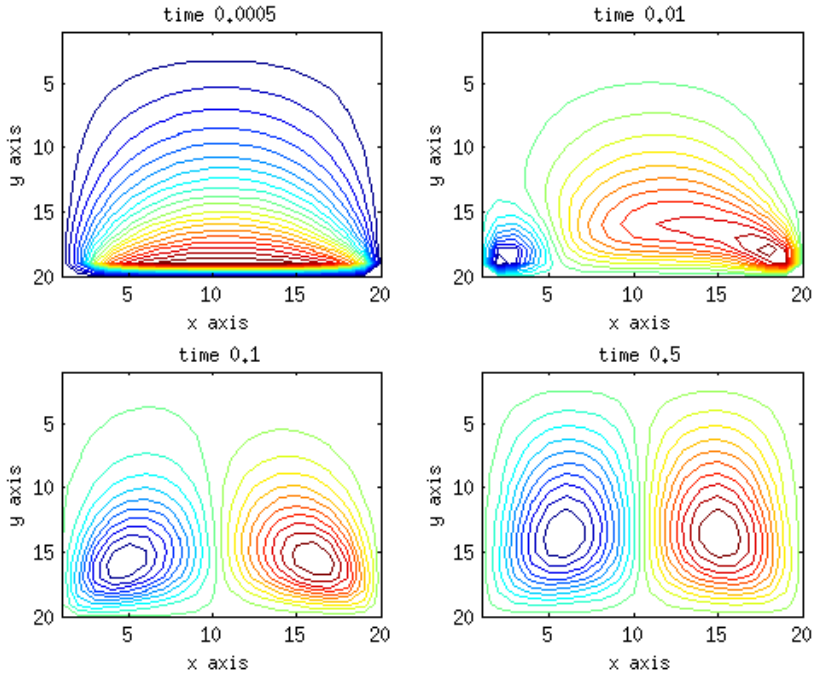

FIGURE 2. Contour plot of the Stream function Ψ for various times

2.3. **Temperature.** The temperature field follows the intuitive pattern where heat propagates upwards in the box through the center, but at a slower time scale that the other variables of the system. See figure 3.

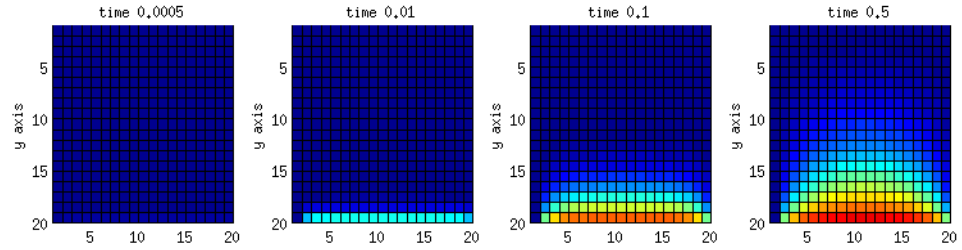

FIGURE 3. The temperature distribution at different times.

2.4. **Analysis.** The first thing to notice on this project is the necessity to use a very small time step,

$$\Delta t = \frac{1}{5}(\Delta x)^2 = \frac{1}{5n^2},$$

where n is the number of subintervals in which we partitioned each side of the box domain. This follows from the well-known Von Neumann stability requirement for the *Forward-Time Central-Space* discretization scheme.

2.4.1. *Parameters.* Modifications of the paramenter Pr produced the same results, but the time scale would change, making a slower evolution the larger we considered Pr (≈ 1000), which corresponds to the physical interpretation of a more viscous fluid and therefore less convection. On the other hand, the fluid would evolve faster when Pr was smaller ($Pr \approx 0.5$), and getting a strange unstable solution when the parameter was below 0.45.

Variations on the Grashof number would produce a lot of interesting changes in the behaviour of the solution. For low values of Gr (≈ 100), the temperature would vary very slowly and converge into a steady solution of the form observed in the right side of figure 3. For larger values ($Gr \approx 10^4$), the solution would become unstable, and it would no longer form the *Rayleigh-Bernard* cells, and instead it would shift constantly due to the stronger coupling between the heat equation and the convection of the fluid. See for example figure 4.

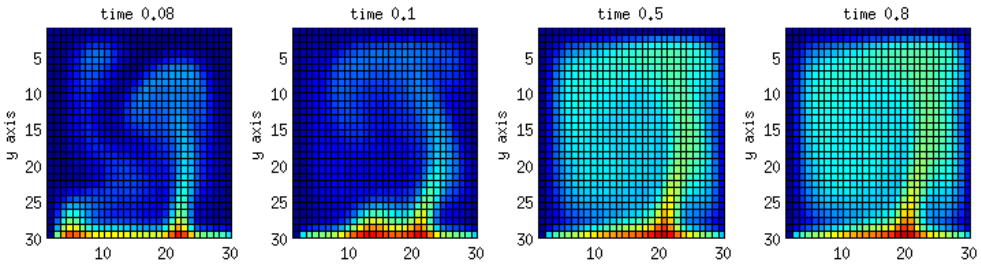

FIGURE 4. Temperature evolution for very large Grashof numbers.

2.5. **Further Work.** One of the most important unresolved issues of this project is the *error analysis* of the algorithm used to produce these graphics. The approach should follow the careful compilation of all the errors from the finite differences and see how they propagate along the iteration scheme. Another important future study would be to consider a wider range of parameter variation and interpretation along with real physical phenomena, along

with the study of time-asymptotics. Finally, the study of full compressible Navier-Stokes would be interesting as a challenge to learn the discretization without the use of the *Stream function*.

References

[1] Finite Difference Methods for Ordinary and Partial Differential Equations. Steady states and Time-dependent problems. *Randall J. LeVeque*. Siam, 2007.
[2] An introduction to computational fluid dynamics by examples. *Sedat Biringen and Chuen-Yen Chow*. John Wiley and Sons, INC., 2011.

A. Code

```
n=20;
step = 1/n;

Gr = 1; %Grashof number measures how strong is
%the coupling between heat and convection
%Gr= 1;
dt= 1/(Gr*5*(n^2)); % time step
Pr = 6.75; %Prandtl number


%initial data
% stream function & temperature

omega = zeros(n,n); %vorticity
psi = zeros(n,n); %stream function
theta = zeros(n,n); %temperature
U = zeros(n,n); % x-velocity
V= zeros(n,n); % y-velocity

% boundary values
a=1;
theta(:,1) = 0; %left
psi(:,1) = 0;
theta(:,n) = 0; %right
psi(:,n) = 0;
psi(n,:) = 0;
theta(1,:) = 0; %top
psi(1,:) = 0;
theta(n,:) = 1; %bottom
%iteration scheme
for t=1:1/dt; %time

    for i=2:n-1; %x direction
        for j=2:n-1; %y direction
            % Velocity vector
            U(j,i) = (psi(j+1,i) - psi(j-1,i))/(2*step);
            V(j,i) = -(psi(j,i+1) - psi(j,i-1))/(2*step);
            % omega on the boundary
            omega(1,i) = (-4*U(2,i)+U(3,i))/(2*step); %top boundary
```

```
            omega(j,1) = (4*V(j,2)-V(j,3))/(2*step); %left boundary
            omega(n,i) = (4*U(n-1,i)-U(n-2,i))/(2*step); %bottom
            omega(j,n) = (-4*V(j,n-1)+V(j,n-2))/(2*step); %right
            % Time evolution
            omega(j,i) = omega(j,i) + dt*(-Gr*(theta(j,i+1)
-theta(j,i-1))/(2*step) + (omega(j,i-1) + omega(j,i+1)
+omega(j-1,i)+omega(j+1,i)
-4.*omega(j,i))./(step^2) - (U(j,i+1)*omega(j,i+1)
- U(j,i-1)*omega(j,i-1))/(2*step) -(V(j+1,i)*omega(j+1,i)
 - V(j-1,i)*omega(j-1,i))/(2*step));
            theta(j,i) = theta(j,i) + dt*((Pr^(-1))*(theta(j,i-1)
 + theta(j,i+1)+theta(j-1,i)+theta(j+1,i)-4.*theta(j,i))./(step^2)
 - (U(j,i+1)*theta(j,i+1) - U(j,i-1)*theta(j,i-1))/(2*step)
-(V(j+1,i)*theta(j+1,i) - V(j-1,i)*theta(j-1,i))/(2*step));
         end
    end
    % Invert laplacian
    psi(2:(n-1),2:(n-1)) = -invert_laplacian(omega(:,:),n);


   % Plotting the solutions

   if t==1 | t== 0.01*1/dt | t== 0.1*1/dt | t==0.5*1/dt
    %subplot(2,2,a); quiver(U,V), set(gca,'YDir','reverse'),
 xlabel('x axis'), ylabel('y axis'), title(['time ',num2str(t*dt')]);

    subplot(1,4,a); surf(theta), view(2),set(gca,'YDir','reverse'),
 xlabel('x axis'), ylabel('y axis'), title(['time ',num2str(t*dt')]);
    %subplot(2,2,a); contour(psi,20), set(gca,'YDir','reverse'),
xlabel('x axis'), ylabel('y axis'), title(['time ',num2str(t*dt')]);
   a =a+1;

   end

end

%%%%%%%%%%%%%%%%%%%%%%%%%

function psi=invert_laplacian(f,n)

% Laplacian matrix
I = eye(n-2);
e = ones(n-2,1);
T = spdiags([e -4*e e],[-1 0 1],n-2,n-2);
S = spdiags([e e],[-1 1],n-2,n-2);
A = (kron(I,T) + kron(S,I))*n^2;

% reorder forcing matrix f into a column vector
F = reshape(f(2:(n-1),2:(n-1)),[(n-2)^2, 1]);
```

```
% reshape the linear system solution vector back into matrix form
psi = reshape(A\F,[(n-2),(n-2)]);
```