Nutrient Flow due to Natural Thermal Convection in Rectangular Cavities

Juan Chacon

April 9, 2019

1 Introduction

The thermal expansion occurring on fluids when its temperature changes, produces density fluctuations which generate an acceleration field, such a motion associated with this flow is called **thermal advection**. When the fluid motion is not generated by any other external source it is called **natural convection**, broadly studied [8, 9]. Due to the density fluctuation the continuity equation is not valid any more, however, the **Boussinesq approximation** allow us to fix this issue assuming the density is constant except in the term interacting with the gravity.

Another important phenomenon taking place in fluids is the transport of a given substance propagating in advective and diffusive ways [11], this takes particular importance in marine ecosystems, according to [12], the role of fluid motion in delivery of nutrients to phytoplankton cells is a fundamental question in biological and chemical oceanography, as those individuals are driven by submarine currents, which are caused in some extend by thermal advection effects. It has been reported that phytoplankton sinks to depths of thousands of meters in the deep ocean during the winter season and rises again during spring [10].

The aim of the present work is propose and study a numerical solver for the advection diffusion equation in presence of natural thermal convection in rectangular cavities, the governing equations and the used numerical method are presented in Sections 2 and 3 respectively, three benchmark examples are considered in Section 4, as long as, we point out some effects of the variation of the equations parameters in Section 5. We conclude we a brief section highlighting some relevant facts.

2 Governing equations

2.1 Thermal Convection

The momentum, energy and continuity equations in dimensional form for a fluid with gradients of temperature without presence of heat sources are given by:

$$\rho\frac{\partial\overrightarrow{u}}{\partial t}+\rho\overrightarrow{u}\cdot\nabla\overrightarrow{u}=-\nabla p+\mu\nabla^2\overrightarrow{u}+\rho\overrightarrow{g}$$
$$\frac{\partial T}{\partial t}+\overrightarrow{u}\cdot\nabla T=\alpha\nabla^2 T$$
$$\nabla\cdot\overrightarrow{u}=0$$

where $\overrightarrow{u}=(u,v)$, T, ρ and p are the fluid velocity, temperature, density and pressure, respectively, μ is the dynamic viscosity, α is the thermal diffusivity constant and $\overrightarrow{g}=(0,g)$ the gravity vector.

2.1.1 Boussinesq approximation

The principal effect of the temperature variations in a fluid is in its density, variations in the first cause changes in the second, producing buoyancy forces. A common approach to treat this is assume constant density ρ_0, except for the term multiplying the gravity, which is supposed linear respect to T around a reference temperature T_0. In other words, the momentum equation is replaced by:

$$\rho_0\frac{\partial\overrightarrow{u}}{\partial t}+\rho_0\overrightarrow{u}\cdot\nabla\overrightarrow{u}=-\nabla p+\mu\triangle\overrightarrow{u}+\rho_0\left(1-\beta\left(T-T_0\right)\right)\overrightarrow{g}$$

Where β is the coefficient of thermal expansion.

2.2 Mass advection diffusion

The concentration c of a substance that diffuses and transports in presence of a velocity field $\overrightarrow{u}$ follows the relation:

$$\frac{\partial c}{\partial t}+\overrightarrow{u}\cdot\nabla c=D\nabla^2 c$$

where D is the mass diffusivity coefficient.

2.3 Dimensionless formulation

Assuming $\triangle T=T_H-T_C$, as the difference between the highest an lowest temperature, and u_0 as a reference velocity, allow us introduce the following new variables,

$$T^*=\tfrac{T-T_0}{\triangle T},\ \beta^*=\triangle T,\ x^*=\tfrac{\mathbf{x}}{L},\ t^*=\tfrac{u_0 t}{L},\ \overrightarrow{u}^*=\tfrac{\overrightarrow{u}}{u_0},\ p^*=\tfrac{p-p_0}{\rho_0 u_0^2},\ \overrightarrow{g}^*=\overrightarrow{g}L$$

after replace in the governing equations, we obtain the dimensionless formulation for this system as:

$$\frac{\partial \overrightarrow{u}^*}{\partial t} + \overrightarrow{u}^* \cdot \nabla \overrightarrow{u}^* = -\nabla p^* + \frac{1}{Re}\nabla^2 \overrightarrow{u}^* + (1 - \beta^* T^*)\, \overrightarrow{g}^*$$
$$\nabla \cdot \overrightarrow{u}^* = 0$$
$$\frac{\partial T^*}{\partial t} + \overrightarrow{u}^* \cdot \nabla T^* = \frac{1}{RePr}\nabla^2 T^*$$
$$\frac{\partial c^*}{\partial t} + \overrightarrow{u}^* \cdot \nabla c^* = \frac{1}{ReSc}\nabla^2 c^*$$

Where $Re = u_0 \frac{\rho_0 L}{\mu}$, $Pr = \frac{\mu}{\rho_0 \alpha}$ and $Sc = \frac{\mu}{\rho_0 D}$, $(Pe = ReSc)$. Note here that all the differential operators are respect the appropriated $*$ variable. Henceforth, we will use this formulation dropping the $*$ symbol.

2.4 Stream function vorticity formulation

Let us define the vorticity function $\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$, and ψ the stream function, such that $\frac{\partial \psi}{\partial y} = u$ and $\frac{\partial \psi}{\partial x} = -v$. By adding the derivatives $\frac{\partial}{\partial y}$ and $\frac{\partial}{\partial x}$ of the first and second components of the dimensionless momentum equations we get:

$$\frac{\partial \omega}{\partial t} + \frac{\partial \psi}{\partial y}\frac{\partial \omega}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial \omega}{\partial y} = \frac{1}{Re}\nabla^2 \omega + \beta g \frac{\partial T}{\partial x} \tag{1}$$

Replacing the u and v in therms of ω and ψ in the energy and mass equations we get:

$$\frac{\partial T}{\partial t} + \frac{\partial \psi}{\partial y}\frac{\partial T}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial T}{\partial y} = \frac{1}{RePr}\nabla^2 T \tag{2}$$
$$\frac{\partial c}{\partial t} + \frac{\partial \psi}{\partial y}\frac{\partial c}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial c}{\partial y} = \frac{1}{ReSc}\nabla^2 c \tag{3}$$

Finally, the relation between ω and ψ is given by:

$$\nabla^2 \psi = -\omega \tag{4}$$

The initial conditions are:

- $T = T_c$ and $c = 0$ in the interior points.

The boundary conditions at any time are:

- $\frac{\partial T}{\partial n} = 0$ and $\frac{\partial c}{\partial n} = 0$ at the east and west walls (adiabatic walls).
- $T = T_c$ and $c = 0$ at the north wall.
- $T = T_h$ and $c = 1$ at the south wall.

3 Numerical Method

The solver strategy studied based in the stream function vorticity formulation is based on [11] chapter 13, and consist in the following steps:

1. At the initial instant ψ, ω, u, v are defined zero in the complete domain, the temperature is T_c except in the bottom wall, and the concentration c is zero, except in the same boundary, where it values c_0.

2. Compute the vorticity, from the velocities using centered in space scheme (second order) in the interior, and using the second order backward difference for the boundary points, i.e.

$$\omega_{i,n_y+1} = \frac{4u_{i,n_y} - u_{i,n_y-1}}{2\Delta y};\ \ \omega_{i,1} = \frac{-4u_{i,2} + u_{i,3}}{2\Delta y};\ \omega_{1,j} = \frac{4v_{2,j} - u_{3,j}}{2\Delta x};$$
$$\omega_{n_x+1,j} = \frac{-4v_{n_x,j} + u_{n_x-1,j}}{2\Delta x}$$

   Same scheme is used to find $\frac{\partial T}{\partial x}$ for the interior points and $\frac{\partial T}{\partial x} = 0$ at the boundary.

3. Use explicit FTCS on Eq. 1 to get vorticity at the next time step.

4. Solve the Poisson equation Eq. 4 with Dirichlet to get ψ, assuming Dirichlet boundary condition constant, using the standard Laplacian operator (second order in space).

5. From ψ compute the velocities using FTCS.

6. Use explicit FTCS on Eq. 2 to get T. Using the mixed boundary condition previously specified.

7. Use explicit FTCS on Eq. 3 to get c. Using the mixed boundary condition previously specified.

8. Iterate from 2.

As a summary the Algorithm 1 states the shortly strategy:

Algorithm 1 Thermal convection - mass transfer solver using explicit FTCS

```
set up the initial conditions for ψ, ω, u and v.
for t in 0...t_sim:
    solve the vorticity equation. (using explicit FTCS).
    solve the Poisson equation for ψ. (using the standard discretization for ∇²).
    compute the new velocity from ψ. (using the
    solve the energy equation for T. (using explicit FTCS).
    solve the mass transfer equation for C. (using explicit FTCS).
```

Remark: the MATLAB code used for one of the benchmark problems is in the Appendix 1, as well as in the public repository https://github.com/juandados/CFD-Project.

3.1 Time Step Restriction

FTCS stability analysis

The following result on its multivariate formulation is proved on [12].

- **Theorem (FTCS 2D stability Restriction).**
  The FTCS discretization of the advection diffussion equation:

$$\frac{\partial \varphi}{\partial t} + u\frac{\partial \varphi}{\partial x} + v\frac{\partial \varphi}{\partial y} = k\nabla^2\varphi$$

  given by:

$$\begin{aligned} &\frac{\varphi_{ij}^{n+1} - \varphi_{ij}^{n}}{\Delta t} + u\frac{\varphi_{i+1j}^{n} - \varphi_{i-1j}^{n}}{2\Delta x} + v\frac{\varphi_{ij+1}^{n} - \varphi_{ij-1}^{n}}{2\Delta y} \\ &= k\left(\frac{\varphi_{i+1j}^{n} - 2\varphi_{ij}^{n} + \varphi_{i-1j}^{n}}{\Delta x^2} + \frac{\varphi_{ij+1}^{n} - 2\varphi_{ij}^{n} + \varphi_{ij-1}^{n}}{\Delta y^2}\right) \end{aligned}$$

  is stable (in the von Neumann sense) if and only if:

$$\Delta t \leq \frac{1}{2k\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)} \text{ and } \Delta t \leq \frac{2k}{u_{max}^2 + v_{max}^2}.$$

Notice our solver require to satisfy simultaneously the following restrictions, one per each evolution equation,

$$\Delta t \leq min\left\{\frac{Re}{2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)}, \frac{RePr}{2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)}, \frac{ReSc}{2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)}\right\},$$

$$\Delta t \leq min\left\{\frac{2Re}{u_{max}^2 + v_{max}^2}, \frac{2RePr}{u_{max}^2 + v_{max}^2}, \frac{2ReSc}{u_{max}^2 + v_{max}^2}\right\}.$$

3.2 Convergence rates

After a Taylor expansion analysis the expected accuracy for this method for the functions ψ, ω is **second order in space**, due to the centered differences, and **first order in time**, due to the explicit Euler approximation.

For the benchmark setup to observe the convergence rate the fluid parameters where $Re = 33.93$, $Pr = 1.25 \times 10^4$ and $\beta = 1.79 \times 10^{-3}$, the aspect ratio of the cavity was 4:1. The metric error was the relative error in the l^2 norm, contrasting the solutions in three different grids (16×4, 32×8, 64×16) respect to the computed solution for a grid of 128×32 points, with time step of $dt = 2 \times 10^{-3}$ for an experiment duration of (no normalized time of 5000s), leading to a simulation time of $6.5\,h$.

The expected spatial rates of convergence were not observed, the Figure 1 shows how the relative errors for velocity are higher than 1 (100%) for early times and there is not a monotone behavior of time and different grid size.

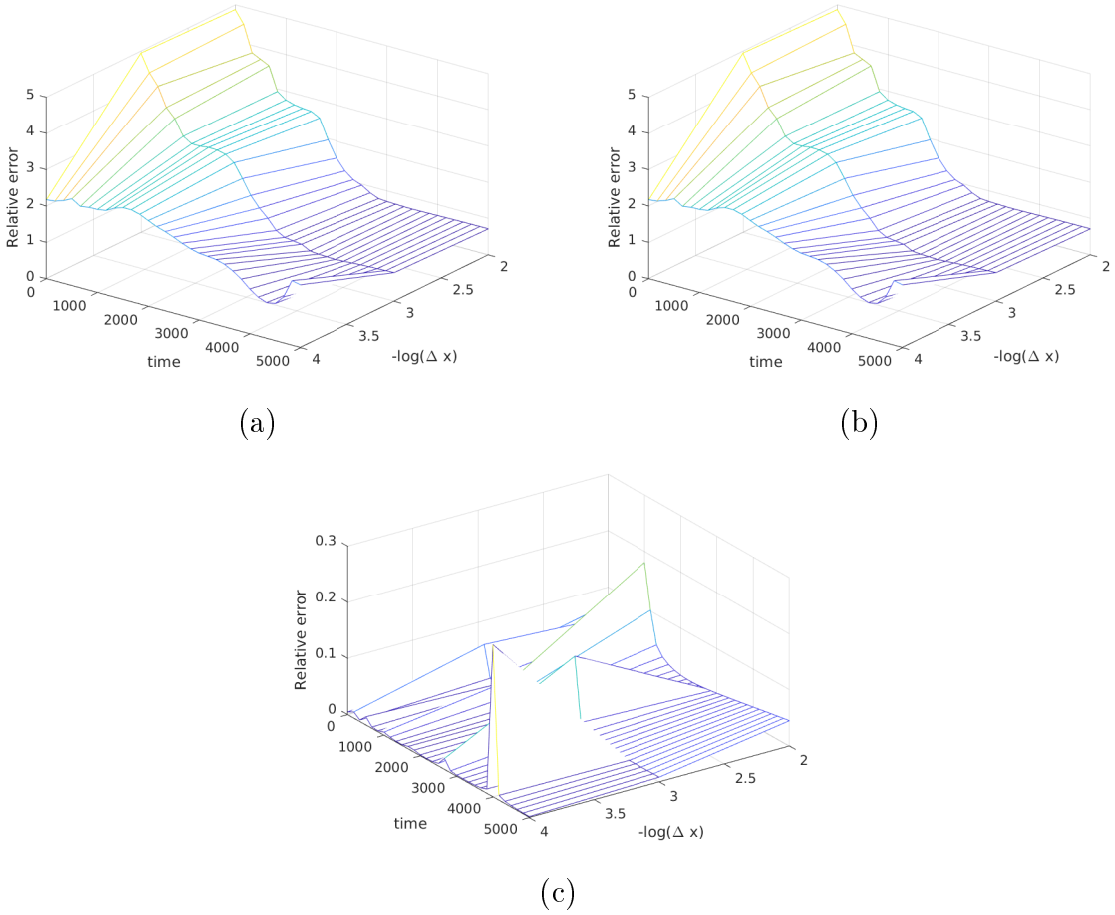

Figure 1: Relative numerical errors for (a) velocity u, (b) velocity v and (c) temperature.

Remarks:

- For short times the speeds are close to zero in the majority of the cavity, even less than the $\epsilon_{mach} = 1 \times 10^{-16}$, leading to relative errors higher than 100%, which makes the computation unreliable. For that reason we prefer for compute the steady state solution task computationally expensive.

- The Pe parameter was not included as our simulations focused in the thermal convection.

4 Example Applications

Boundary and Initial conditions

- The domain Ω for this problem will be a rectangular cavity $\Omega = [0, L_x] \times [0, L_y]$.
- The fluid starts stationary i.e. $\mathbf{u}(\mathbf{x}, t) = 0$, and it satisfies the no penetration and no slip condition for the boundary, $\mathbf{u}(\mathbf{x}, t)|_{\partial\Omega} = 0$ and $\mathbf{u} \cdot \overrightarrow{n}|_{\partial\Omega} = 0$.
- On the other hand, the temperature satisfies $T(x, 0, t) = T_H$ and $T(x, L_y, t) = T_C$. In addition, the right and left boundaries are thermally insulating, i.e. there is not heat flux.
- Finally, the nutrient concentration comes from the bottom of the square and the initial concentration is 0 out of that wall, i.e $c(x, 0, 0) = C$ and $c(x, y \neq 0, 0) = 0$.

4.1 Glycerin

This benchmark setup for Glycerin fluid was taken from [13], with experimental parameters:

$L_x\,[m] \times L_y\,[m]$	$T_C\ [K]$	$T_H\ [K]$	$\rho_0\ \left[\frac{kg}{m^3}\right]$	$\mu\ \left[\frac{kg}{ms}\right]$	$\beta\ \left[\frac{1}{K}\right]$	Pr	Re
0.38×0.04	291.2	294.78	1264.02	1.499	1.79×10^{-3}	1.25×10^4	33.73

Table 1: Glycerin setup experimental parameters

The reference velocity $u_0 = 1$, and the experiment duration is 1×10^4 seconds. Both the number of cells, and isothermal curves geometry match with [13] as is shown in the Figure 2, except for the direction of rotation.

4.2 Air

This benchmark setup for Glycerin fluid was taken from [13], with experimental parameters:

$L_x\,[m] \times L_y\,[m]$	$T_C\ [K]$	$T_H\ [K]$	$\rho_0\ \left[\frac{kg}{m^3}\right]$	$\mu\ \left[\frac{kg}{ms}\right]$	$\beta\ \left[\frac{1}{K}\right]$	Pr	Re
2.622×0.655	292.5	293.5	1.205	1.81×10^{-4}	3.4×10^{-3}	0.72	4365

Table 2: Air setup experimental parameters

From Figure 3 three phases can be differentiated qualitative based on the isotherm sets:

- The isotherm sets are horizontal straight lines moving vertically as the time advances.
- The isotherm sets curve forming shapes than change on time.

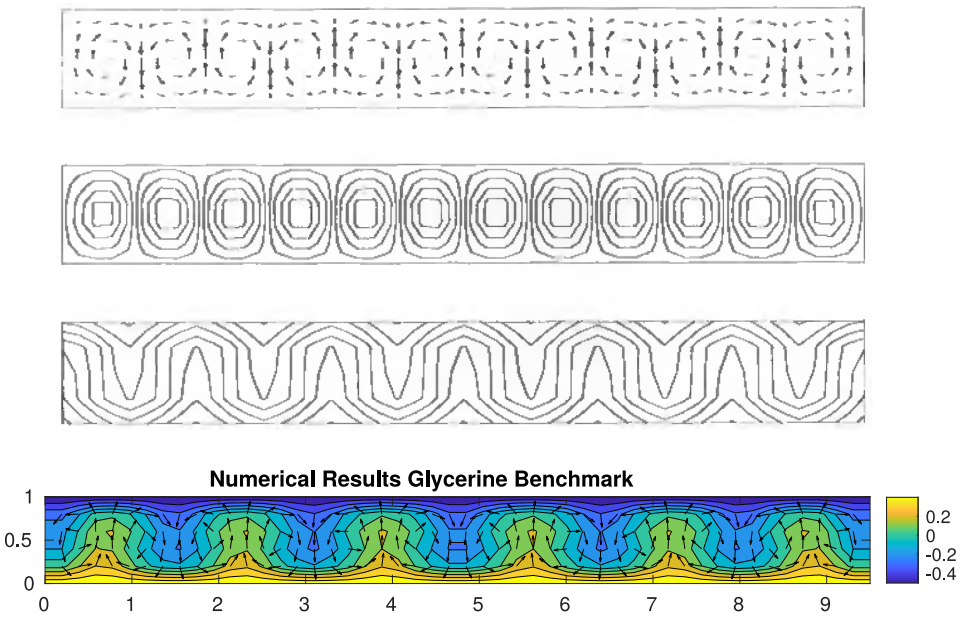

Figure 2: Vector field and temperature sets for glycerin setup, comparison with [13].

- The isotherm sets stabilize, reaching the steady state.

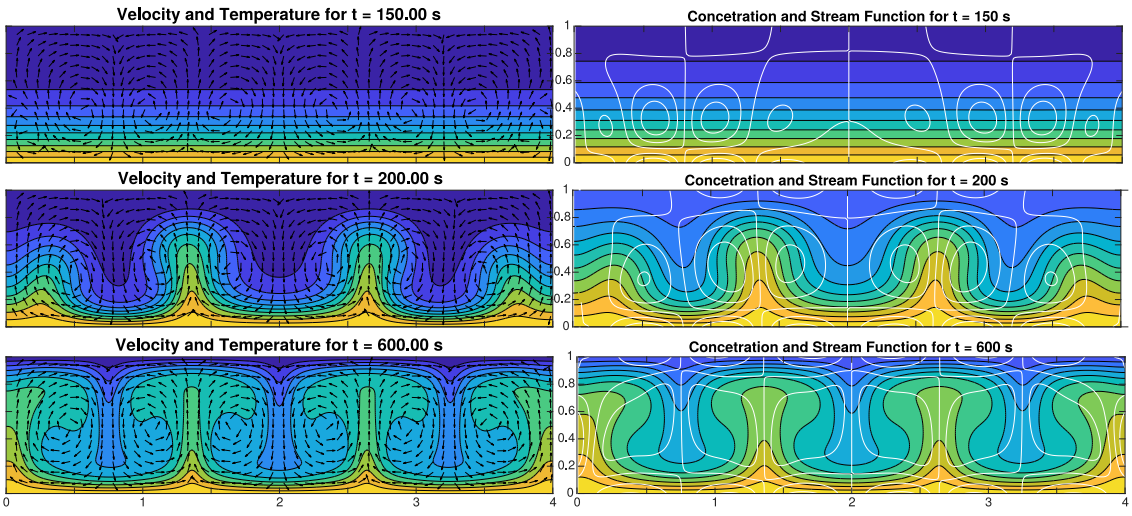

Figure 3: Velocity, stream function, temperature, concentration and Stream function time evolution on air setup.

4.3 Plankton

The parameters taken into account for this experiment were based on the following facts:

- The phytoplankton sinks in the oceans up to $160\,m$ per year (passive swimmers).

- The temperature in the ocean may vary up to $10K$ per year.

We wanted to study the effects of variations on temperature around $2K$ in a big rectangular area in the sea ($160\,m$ deep, $1.6\,km$ wide) with planktonic organisms in the bottom, getting as parameters:

$L_x\,[m] \times L_y\,[m]$	$T_C\ [K]$	$T_H\ [K]$	$\rho_0\ \left[\frac{kg}{m^3}\right]$	$\mu\ \left[\frac{kg}{ms}\right]$	$\beta\ \left[\frac{1}{K}\right]$	Pr	Re	Pe
160×1600	281	283	1.025	8.9×10^{-4}	2.4×10^{-3}	7.56	1.84×10^5	6965.39

Table 3: Plankton in the sea setup experimental parameters

Although the Reynolds number is enormous, some general structure of this phenomena might be preserved, e.g. the number of cells or the average concentration in specific areas.

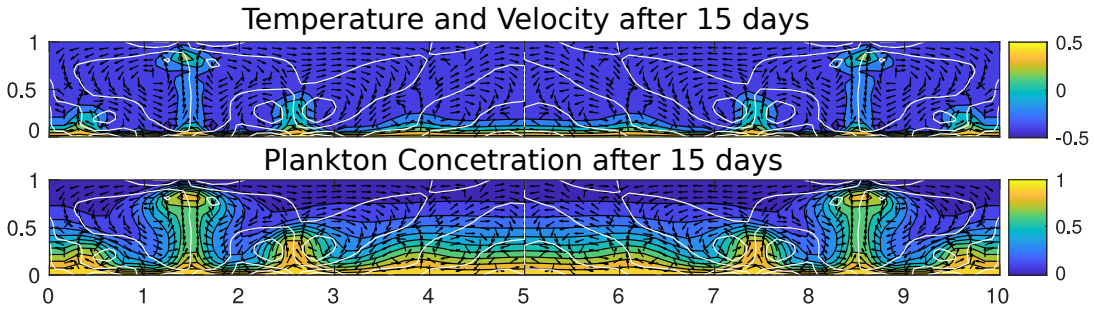

Figure 4: Velocity, stream function, temperature, concentration and Stream function on plankton setup.

5 Steady State Solution Parameter Dependence

The ψ - ω formulation Eqs. 1 - 4 is four parameter dependent, (β, Re, Pr and Sc), the aim of this section is approach the effects of variation of those, as well as for the grid size and cavity aspect ratio.

5.1 Dependence on Grid Size and Time step

For the same setup used in 4.2 two different grid size were considered, $\Delta x = 156 \times 10^{-3}, \Delta y = 62.5 \times 10^{-3}$ $\Delta x = 39 \times 10^{-3}, \Delta y = 15.4 \times 10^{-3}$, Figure 5 shows that in the coarser discretization only 4 cells Rayleigh-Benard cells form, compared to 6 in the finer one. This indicates that some discretizations cannot completely resolve the structure of the flow.

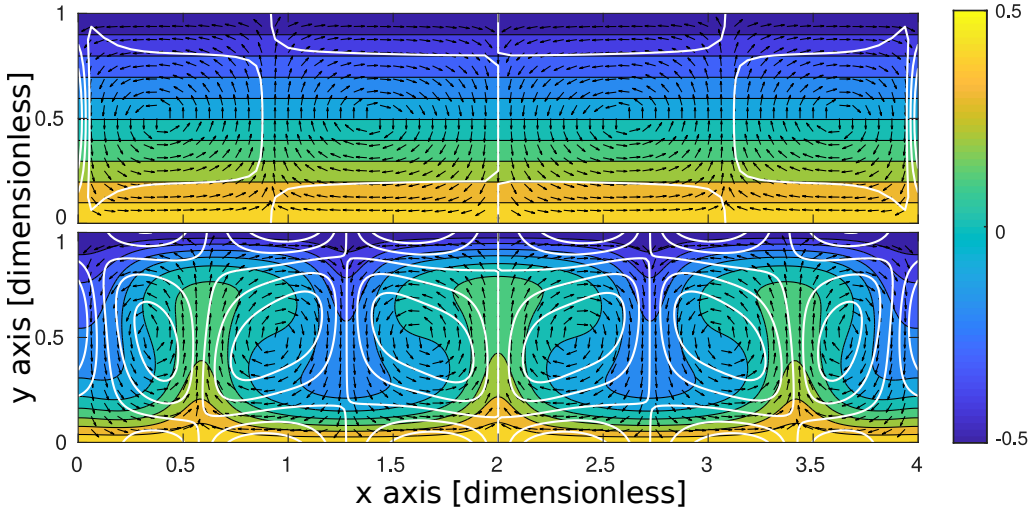

Figure 5: Difference on steady state fluid for grids glycerin setup: (top) $\Delta x = 156 \times 10^{-3}, \Delta y = 62.5 \times 10^{-3}$ (bottom) $\Delta x = 39 \times 10^{-3}, \Delta y = 15.4 \times 10^{-3}$.

Similarly, the time discretization seems to have a diffusive effect (I did not expected that!), smaller time step leads to loss of cells as can be seen in the Figure 6, this effect may be the cause of the problems seeing the expected convergence rate.

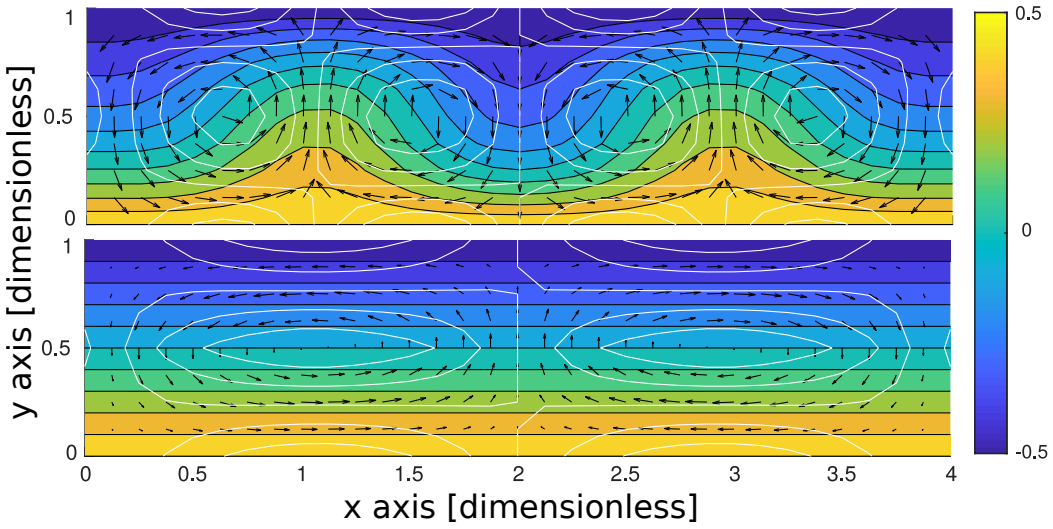

Figure 6: Difference on steady state fluid for time steps air setup: (top) $\Delta t = 1 \times 10^{-1}$, (bottom) $\Delta t = 1 \times 10^{-2}$

5.2 Dependence on Domain Aspect Ratio

Numerical experiments were ran for the glycerin benchmark, for aspect ratios 1:1, 1:4 and 1:6. The Figure 7 shows that both the number of cells and the isotherm sets geometry depend on the cavity aspect ratio.

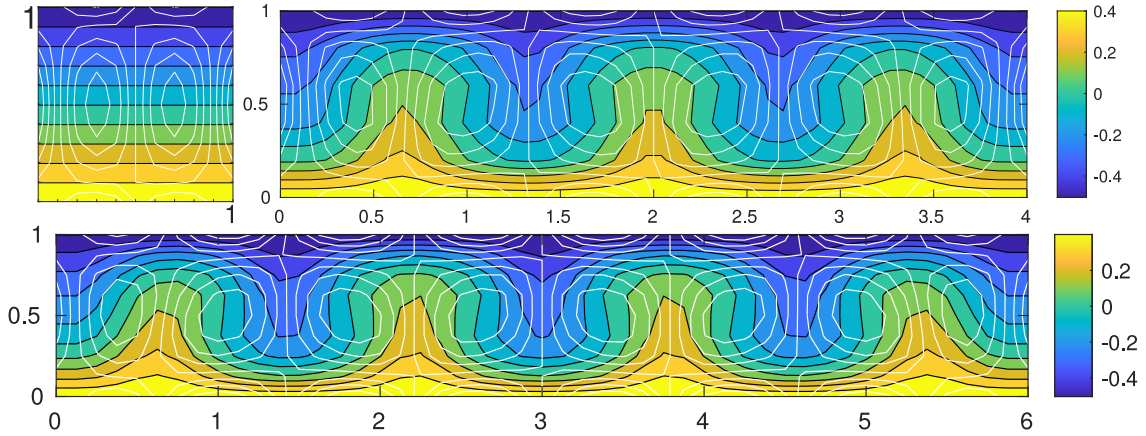

Figure 7: Different cell formation for glycerine setup depending on the aspect ratios

5.3 Dependence on Reynolds Prandtl Number

The dependence on the fluid steady state is shown in Figures 8 and 9, higher Re, Pr and β leads to sharp defined thermal plumes and more cells.

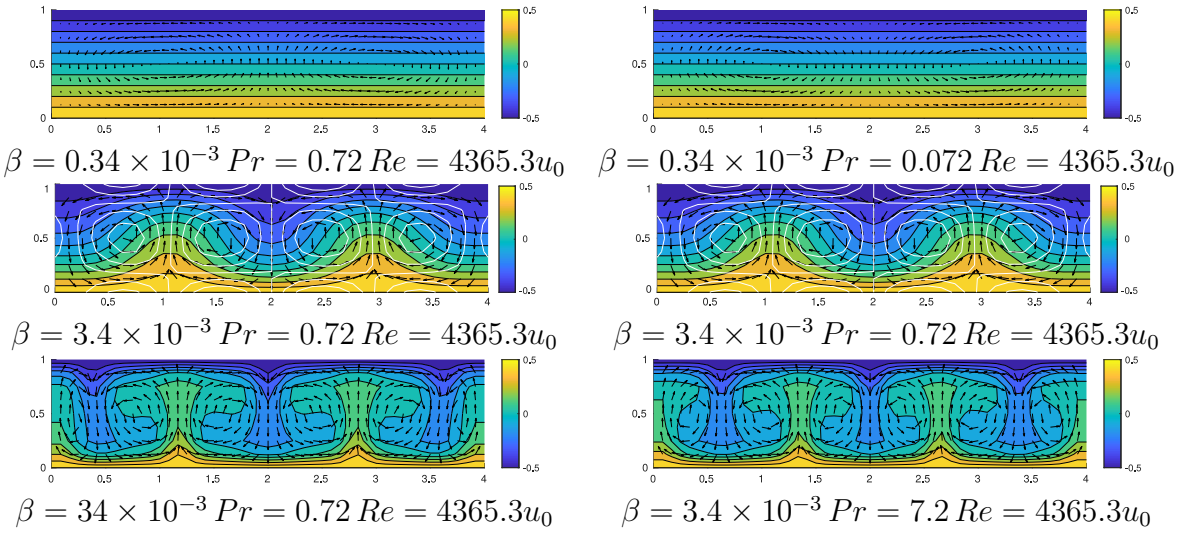

Figure 8: Effects on the steady state solution due to the variations on (left): β, (right) Pr.

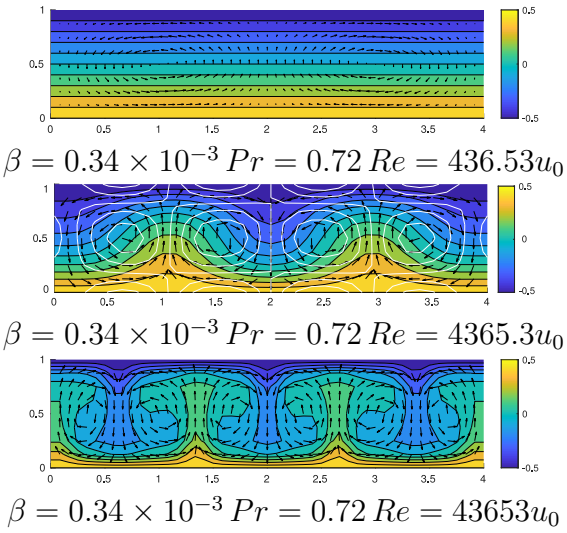

Figure 9: Effects on the steady state solution due to the variations on (down) Re.

6 Conclusions

- The behavior of the thermal cells depends on the size of the spatial discretization and time step.
- The aspect ratios of the cavity affects the cells count and the isotherm sets geometry.
- Higher Re, Pr and β leads to sharp defined thermal plumes and more cells.
- Even with high Reynolds numbers the implemented method seems to show properly the general structure of the fluid.

Pendings

- A solver using the projection method was considered, however, the staggered grid was not work as expected for the energy equation.
- An implementation using the second order FTBS scheme will help to deal with higher Reynolds numbers.

References

[1] Prasopchingchana, U., Pirompugd, W., Laipradit, P., & Boonlong, K. (2013). Numerical study of natural convection of air in an inclined square enclosure. momentum, 3, 2.

[2] Kosec, G., & Šarler, B. (2011). Numerical solution of natural convection problems by a meshless method. In Convection and Conduction Heat Transfer. InTech.

[3] Yao, H. (1999). Studies of natural convection in enclosures using the finite volume method.

[4] Marc Fermigier. Thermal Convection. https://blog.espci.fr/marcfermigier/files/2017/04/thermalconv.pdf

[5] Karp-Boss, L., Boss, E., & Jumars, P. A. (1996). Nutrient fluxes to planktonic osmotrophs in the presence of fluid motion. Oceanography and Marine Biology, 34, 71-108.

[6] Musielak, M. M., Karp-Boss, L., Jumars, P. A., & Fauci, L. J. (2009). Nutrient transport and acquisition by diatom chains in a moving fluid. Journal of Fluid Mechanics, 638, 401-421.

[7] Seibold, B. (2008). A compact and fast Matlab code solving the incompressible Navier-Stokes equations on rectangular domains mit18086 navierstokes. m. Massachusetts Institute of Technology.

[8] Chorin, A. J. (1967). The numerical solution of the Navier-Stokes equations for an incompressible fluid. Bulletin of the American Mathematical Society, 73(6), 928-931.

[9] Barakos, G., Mitsoulis, E., & Assimacopoulos, D. (1994). Natural convection flow in a square cavity revisited: laminar and turbulent models with wall functions. International Journal for Numerical Methods in Fluids, 18(7), 695-719.

[10] Backhaus, J. O., Wehde, H., Hegseth, E. N., & Kämpf, J. (1999). \'Phytoconvection\': the role of oceanic convection in primary production. Marine Ecology Progress Series, 189, 77-92.

[11] Spurk, J. H. (1997). Pozrikidis, C.: Introduction to Theoretical and Computational Fluid Dynamics. New York/Oxford, Oxford University Press 1997. XII, 675 pp.,£ 52.50. ISBN 0-19-509320-8. Zeitschrift Angewandte Mathematik und Mechanik, 77, 924-924.

[12] Hindmarsh, A. C., Gresho, P. M., & Griffiths, D. F. (1984). The stability of explicit Euler time-integration for certain finite difference approximations of the multi-dimensional advection–diffusion equation. International journal for numerical methods in fluids, 4(9), 853-897.

[13] Griebel, M., Dornseifer, T., & Neunhoeffer, T. (1997). Numerical simulation in fluid dynamics: a practical introduction (Vol. 3). Siam.

Apendix A Code

```
function [xx,yy,Us,Vs,Ts,Cs,Ws] = psiw_thermal_convection(nsteps)
   % Solves the unsteady thermal convection problem using the
```

```
% streamfunction-vorticity algorithm described in
% Pozrikidis (1997), pages 608-9.
%
% Author:  Juan Chacon
% Department of Mathematics
% Simon Fraser University
% jchaconl@sfu.ca
%
% Date:  Apr 11, 2019
%----------------------------- Defining Defaults -------------------
nsteps = 2; % number of steps with graphic output
nx = 80;ny = 20;
%-------------------- Physical constants Air-------------------
Ly = 0.6557; % Cavity height [m]
Lx = Ly*4; % Cavity width [m]
L = Ly; % Reference lenght [m]
lx = Lx/L; % Normalized cavity width []
ly = Ly/L; % Normalized cavity height []
G = 9.8126; % Gravity acceleration [m/s^2]
g = L*G; % Gravity acceleration [m/s^2]
TC = 292.5; % Cold temperature [K]
TH = 293.5; % Hot temperature [K]
T0 = 293; % Reference Temperature [K]
Tc = (TC-T0)/(TH-TC); % Cold temperature [K]
Th = (TH-T0)/(TH-TC); % Hot temperature [K]
rho = 1.205; % Air density [kg/m^3]
mu = 1.81e-4; % Dynamic viscosity []
nu = mu/rho; % Kinematic viscosity []
B = 3.4e-3; % coeficient of thermal expansion [1/K]
Pr = 0.72; % Prandtl number
u0 = 1; % Reference velocity [m/s]
Re = rho*u0*L/mu; % Reynolds number
b = (Th-Tc)*B; % normalized coeficient of thermal expansion []
Gr = G*B*(Th-Tc)*(L^3/nu^2);% Grashof number
Ra = Pr*Gr; % Rayleigh number
tS = 1e4; % Real Experiment time [s] tS = 1e4;
tend = tS/L; % Simulation time []
D = 10; % Nutrient diffusion constant relative;
Pe = Re*Pr*0.5; % Peclet number;
c = 1; % Nutrient concentration in the bottom wall;
%-------------------------- Derived parameters ---------------
dx = lx/nx; dy = ly/ny
% Show time step based on the minimum of the two time step
% restrictions that come from the von Neumann analysis of the
% 2D linear advection-diffusion equation.
dt1 = min(dx,dy); % advection restriction
dt2 = 0.5 / nu / (1/dx^2 + 1/dy^2); % diffusion restriction
dt3 = nu / Vd^2; % mixed (*not used*)
safetyfac = 0.8; % "safety factor" (should be < 1)
nt = floor(tend / (min(dt1,dt2) * safetyfac));
dt = tend / nt;
nt = floor(tend / dt);
if dt1 < dt2, tstr = 'advection-limited';
 else tstr = 'diffusion-limited';
```

```
end;
fprintf( 1, 'Actual time step:  %e (%s)\n', dt, tstr );
% Set up an equally-spaced rectangular grid.
[xx,yy] = meshgrid(0:dx:lx, 0:dy:ly);
% ----------------------STEP 1-----------------------------
% Set up the initial conditions for psi, u and v, which are
% zero except for the lid velocity.  The unknowns are all
% node-centered and arrays are of size (nx+1)-by-(ny+1).
psi = 0*xx;
w = psi;
u = psi;
v = psi;
T = psi+Tc; T(end,:)  = Tc; T(1,:)  = Th;
u(end,:)  = Vd; % lid BC
C = psi; C(1,:)  = c;
% Set up the matrix for the Poisson equation for psi.  The ordering
% of unknowns is row-wise:
% (1,1), (1,2), (1,3), ...  (1,Nx+1), (2,1), (2,2), ...
mx = 1/dx^2;
my = 1/dy^2;
e = repmat( ones(nx-1,1), ny-1, 1);
esub = repmat( [ones(nx-2,1); 0], ny-1, 1);
esup = repmat( [0; ones(nx-2,1)], ny-1, 1);
nn = (nx-1)*(ny-1);
A = spdiags( [my*e, mx*esub, -2*(mx+my)*e, mx*esup, my*e], ...
[-(nx-1), -1, 0, 1, nx-1], nn, nn );
% These integer index vectors refer to interior grid points and are
% useful when indexing arrays below.
ii = 2:nx;
jj = 2:ny;
h1 = figure(1);
k = 1; tt = 1e3;
for i = 1 :  nt,
 tic
 %---------------------------- STEP 2.----------------------------
 % Compute vorticity by differencing the velocities,
 % first in the interior:
 w(jj,ii) = (v(jj,ii+1) - v(jj,ii-1)) / (2*dx) ...
 - (u(jj+1,ii) - u(jj-1,ii)) / (2*dy);
 % ...  and then on the boundaries:
 w(jj,1) = (4*v(jj,2) - v(jj,3)) / (2*dx);
 w(jj,end) = (v(jj,end-2) - 4*v(jj,end-1)) / (2*dx);
 w(1,ii) = (-4*u(2,ii) + u(3,ii)) / (2*dy);
 w(end,ii) = (-u(end-2,ii) + 4*u(end-1,ii) - 3*Vd) / (2*dy);
 %---------------------------- STEP 3.----------------------------
 % Solve the vorticity equation using an explicit FTCS scheme.
 w(jj,ii) = w(jj,ii) - dt * ...
 ( u(jj,ii) .* (w(jj,ii+1) - w(jj,ii-1)) / (2*dx) ...
 + v(jj,ii) .* (w(jj+1,ii) - w(jj-1,ii)) / (2*dy) ) ...
 + (1/Re)*dt*(w(jj,ii+1) - 2*w(jj,ii) + w(jj,ii-1)) / dx^2 ...
 + (1/Re)*dt*(w(jj+1,ii) - 2*w(jj,ii) + w(jj-1,ii)) / dy^2 ...
 + dt*b*g*(T(jj,ii+1) - T(jj,ii-1)) / (2*dx);
 %---------------------------- STEP 4.----------------------------
 % Solve the Poisson equation for streamfunction, by first
```

```
% setting up the right hand side for Delta psi = -w.
rhs = -w(jj,ii); % this is a (nx-1) by (ny-1) matrix
% Fix up the RHS for the boundary conditions.
rhs(1,:)  = rhs(1,:)  - psi(1,ii) / dy^2;
rhs(end,:)  = rhs(end,:)  - psi(end,ii) / dy^2;
rhs(:,1) = rhs(:,1) - psi(jj,1) / dx^2;
rhs(:,end) = rhs(:,end) - psi(jj,end) / dx^2;
% Finally, convert the RHS to a vector and solve the linear system.
rhs = reshape( rhs', nn, 1 ); % this is a (nx-1)*(ny-1)-vector
psivec = A \ rhs;
psi(jj,ii) = reshape(psivec, nx-1, ny-1)';
%--------------------------- STEP 5.----------------------------
% Compute the new velocity components by differencing
% the streamfunction.
u(jj,ii) = (psi(jj+1,ii) - psi(jj-1,ii)) / (2*dy);
v(jj,ii) = -(psi(jj,ii+1) - psi(jj,ii-1)) / (2*dx);
%--------------------------- STEP 6.0----------------------------
% impose the boundary conditions boundaries for T:
T(jj,1) = T(jj,2); T(jj,end) = T(jj,end-1);
T(1,ii) = Th; T(end,ii) = Tc;
%--------------------------- STEP 6.1 ----------------------------
% Compute the solution for the termal advection diffusion
T(jj,ii) = T(jj,ii) - dt * ...
((u(jj,ii+1).*T(jj,ii+1) - (u(jj,ii-1).*T(jj,ii-1))) / (2*dx) ...
+ (v(jj+1,ii).*T(jj+1,ii) - (v(jj-1,ii).*T(jj-1,ii))) / (2*dy)) ...
+ (1/(Pr*Re))*dt*(T(jj,ii+1) - 2*T(jj,ii) + T(jj,ii-1)) / dx^2 ...
+ (1/(Pr*Re))*dt*(T(jj+1,ii) - 2*T(jj,ii) + T(jj-1,ii)) / dy^2;
% STEP 7.0 impose the boundary conditions boundaries for T:
C(jj,1) = C(jj,2); C(jj,end) = C(jj,end-1);
C(1,ii) = c; C(end,ii) = 0;
%--------------------------- STEP 7 ----------------------------
% Compute the solution for the termal advection diffusion
C(jj,ii) = C(jj,ii) - dt * ...
((u(jj,ii+1).*C(jj,ii+1) - (u(jj,ii-1).*C(jj,ii-1))) / (2*dx) ...
+ (v(jj+1,ii).*C(jj+1,ii) - (v(jj-1,ii).*C(jj-1,ii))) / (2*dy)) ...
+ (1/(Pe))*dt*(C(jj,ii+1) - 2*C(jj,ii) + C(jj,ii-1)) / dx^2 ...
+ (1/(Pe))*dt*(C(jj+1,ii) - 2*C(jj,ii) + C(jj-1,ii)) / dy^2;
if numel(tt) < 1000
tt(i) = toc;
end
nalerts = 100;
if (floor(nalerts*i/nt)>floor(nalerts*(i-1)/nt))
 fprintf('estimated time:  %0.2f\n',median(tt)*(nt-i));
end
%--------------------------- Present graphics ----------------------
if (i==1|floor(nsteps*i/nt)>floor(nsteps*(i-1)/nt))
 Len = sqrt(u.^2+v.^2+eps);
 Len = sqrt(u.^2+v.^2)+eps;
 % ------Plot 1:
 subplot(2,1,1);
 contourf(0:dx:lx,0:dy:ly,T);
 colorbar; hold on;
 quiver(xx,yy,u,v,0.6,'k-');
 axis equal, axis([0 lx 0 ly]);
```

```
   % ------plot 2:
   subplot(2,1,2);
   contourf(0:dx:lx,0:dy:ly,C); colorbar;hold on;
   contour(xx,yy,w,20,'k-');
   hold off, axis equal, axis([0 lx 0 ly]);
   title(sprintf('Re = %0.1g t = %0.2g',Re,i*dt));
   drawnow
   % ------saving arrays
   Us(:,:,k) = u; Vs(:,:,k) = v; Ts(:,:,k) = T;
   Ws(:,:,k) = w; Cs(:,:,k) = C;
   k = k + 1;
  end
 end
end
```