6 Problem set #6: Putting it all together: 2-D BVP's and Rayleigh-Benard Convection

In this problem set we will use everything we've learned so far to put together a kick-ass solver for a classic fluid dynamics problem: 2-D Rayleigh Benard convection (this is the convection problem all the rest are built on). The dimensionless equations for this problem are

$$\frac{\partial T}{\partial t} + \mathbf{V}\cdot\boldsymbol{\nabla} T = \nabla^2 T \tag{6.1}$$

$$\nabla^2\omega = \mathrm{Ra}\frac{\partial T}{\partial x} \tag{6.2}$$

$$\nabla^2\psi^s = -\omega \tag{6.3}$$

where T is the temperature, ω is the vorticity ($\boldsymbol{\nabla}\times\mathbf{V}$), ψ^s is the 2-D streamfunction,$\mathbf{V} = \boldsymbol{\nabla}\times\psi^s\mathbf{j}$ is the fluid velocity[5] and

$$\mathrm{Ra} = \frac{\rho\alpha\Delta T g d^3}{\eta\kappa} \tag{6.4}$$

is the *Rayleigh Number* where ρ is the fluid density, α is the coefficient of thermal expansion, ΔT is the temperature difference across the layer, g is the acceleration due to gravity, d is the layer thickness, η is the fluid viscosity and κ is the thermal diffusivity. The Rayleigh number is the ratio of things that drive convection (increase vorticity) to those that oppose it. See Chapter 2 Section 2.2 for the derivation and more details.

Equations (6.1)–(6.3) is a highly non-linear system of PDE's which couple an advection-diffusion problem with two Poisson problems (BVP's). There are no known analytic solutions for these equations at high Rayleigh number, yet you will solve them easily (?) with the tools we've developed. For this problem we will solve Eqs. (6.1)–(6.3) in a 2×1 box with the following boundary and initial conditions. Boundary conditions on T are dirichlet top and bottom and reflection sides i.e.

$$T(x,0,t) = 1 \qquad T(x,1,t) = 0 \tag{6.5}$$

$$\frac{\partial T}{\partial x}(0,z,t) = 0 \qquad \frac{\partial T}{\partial x}(2,z,t) = 0 \tag{6.6}$$

Boundary condition on ω and ψ^s are *free stress* and *no-flux* i.e. $\omega = \psi^s = 0$ on all boundaries. For initial conditions we will do the *spin-up* problem where the temperature begins as a linear conductive ramp with a small cosine perturbation across it i.e.

$$T(x,z,0) = (1-z) + \epsilon\cos(2\pi x/x_{max}) \tag{6.7}$$

[5]For this problem we assume we are in the x,z plane with x increasing horizontally and z increasing vertically. The vorticity and stream-function are actually vectors that point perpendicular to the plane in the $\mathbf{j}$ direction.

6.1 Preliminaries

To get going, I have provided you with two sets of goodies. Download and unpack the code in `probset6.tar.gz` which will contain two subdirectories called `poispack` and `cnsl2d`. The first is a collection of test codes for solving the Poisson problems. The second directory contains source code for solving the Eq. (6.1) using a rather nifty 2-D all-in-one Semi-Lagrangian Crank-Nicholson multigrid advection-diffusion solver. In more detail:

Poispack contains four fortran codes for solving the standard Poisson test problem

$$\nabla^2\psi^s = \sin\left(\frac{m\pi x}{x_{max}}\right)\sin\left(\frac{n\pi z}{z_{max}}\right) \tag{6.8}$$

with dirichlet boundary conditions $\psi^s = 0$. The four codes are

1. **dspois1** A sparse LU direct solver using Y12M public domain code (sloooooooooooooooooooow)
2. **sorpois1** a home grown SOR (simultaneous over-relaxation) code based on Numerical Recipe's codes.
3. **crpois1** A cyclic reduction code based on the FISHPACK routine hwscrt (trés zippy).
4. **mgpois00, mgpois01** My own personal multi-grid codes. (trés chic). (mgpois00 stores one constant coefficient stencil for all points, mgpois01 stores an individual 5-point stencil for each point).

Type `makeall fast` to make them and use the matlab script `runall.m` to compare their runtimes for a 129×129 grid run on a 2×2 sin-cell solution. (Note, the fortran routines use the system calls dtime and etime which work fine on Suns. On an AIX machine they should be replaced by dtime_ and etime_). runall.m also implements a spectral solution `fftpois.m` and two direct solutions (gaussian elimination and LU decomposition) using Matlab sparse solvers. Play around with parameters and the script and understand how the different solvers work.

cnsl2d This code solves the standard advection diffusion problem Eq. (6.1) in the same geometry as problem set 5 except that the problem has been rescaled by the diffusion time such that the maximum velocity $|\mathbf{V}_{max}| = \mathrm{Pe}$. The code, solves the discrete stencil problem

$$\begin{bmatrix} & -1 & \\ -\alpha & \gamma + 2(1+\alpha) & -\alpha \\ & -1 & \end{bmatrix} T^{n+1}_{i,j} = \begin{bmatrix} & 1 & \\ \alpha & \gamma - 2(1+\alpha) & \alpha \\ & 1 & \end{bmatrix} T^{n}_{i',j'} \tag{6.9}$$

where $\alpha = (\Delta z/\Delta x)^2$ and $\gamma = 2\Delta x^2/\Delta t$. Point (i, j) is on the grid and point (i', j') is the interpolated point on the characteristic that goes through point (i, j). The code uses a semi-lagrangian scheme to evaluate the right-hand side and a multi-grid scheme to solve the penta-diagonal BVP that results. This scheme has no courant condition and works for everything from pure diffusion problems ($Pe = 0$) to pure advection problems ($Pe \to \infty$) but for that you better take an infinitesimal time step. Compare the solution time and accuracy with your results from Problem set 5.

6.2 The real problem

Okay, now that that is all over do the following

1. Write down in pseudo-code an algorithm for time-stepping the entire system of equations (6.1)–(6.3).

2. Choose one Poisson solver from Poispack (justify your choice) and modify csnl2d to implement your convection scheme.

3. Check your codes by running in a 2×1 box at $\mathrm{Ra} = 10^4, 10^5, 10^6$ and compare to runs on the web-page.

4. The *critical Rayleigh number* for this problem is ?? . What does this mean and how accurate is your code in reproducing this behaviour?

5. **Extra Credit** the nusselt Number $\mathrm{Nu} =$ is a measure of the efficiency of heat transfer by convection vs. conduction across the layer. Calculate the time averaged Nusselt number for your code and discuss how it scales with the Rayleigh number (hint: plot $\log \mathrm{Nu}$ against $\log \mathrm{Ra}$

6. **ExtraExtra Credit** Write a 3-D spherical non-constant viscosity mantle convection code with plates (in parallel). Now run for numerical god.