2D Rayleigh Benard Convection Code

Adarsh Ramtel
MSc Physics
IIT Roorkee

Code Architecture

Algorithm
based on
Gary Glatzmeier's book –
Introduction to Modelling
Convection in Planets and
Stars.
All credits to the author.

```
Initialization → Main Loop →
    [Compute Nonlinear Terms] →
    [Solve for ψ from ω] →
    [Update Fields] →
    [Save Frame to HDF5] →
Visualization
```

Libraries and Performance

Libraries used :

- Numpy,
- HDF5,
- Matplotlib
- Numba

Performance strategy:

- @njit(parallel=True) for heavy loops (not all loops will support this)
- Core allocation - set_num_threads(7) # using seven cores in this case
- Custom tridiagonal solver with Thomas algorithm

Core Functions

- create_tridiagonal_matrix: SciPy CSR matrix builder
- initialize_h5_movie_file: Storing Metadata - TimeStep, Temperature Value, Streamfunction Value, Vorticity Value, all at each coordinate.
- Visualization Utilities- compute_visualization_data: Transforms modal coefficients → real space
- compute_tem_omg_psi_values - Same idea, more plot friendly
- compute_derivatives - From ψ to velocity field (u, w)
- save_movie_frame_h5 - Store 2D slices per frame
- tridiagonal_solver_numba: Forward elimination + back substitution(Thomas Algorithm) , Avoids SciPy overhead

Initializing Temperature Modes

```python
@njit
def init_arrays(Nz, Nn):
    """Initialize arrays for simulation"""
    psi = np.zeros((Nz, Nn + 1))
    omg = np.zeros((Nz, Nn + 1))
    tem = np.zeros((Nz, Nn + 1))
    domgdt = np.zeros((Nz, Nn+1, 2))
    dtemdt = np.zeros((Nz, Nn+1, 2))

    ###################### Initialize temperature ###################

    z_vals = np.linspace(0, 1, Nz)
    tem[:, 1] = 0.07 * np.sin(np.pi * z_vals)#* np.random.uniform(-1,1,Nz)
    tem[:, 0] = 1 - z_vals
    tem[:, 8] = 0.07 * np.sin(np.pi * z_vals)

    return psi, omg, tem, domgdt, dtemdt
```

- Here, we initialize the 1st mode and the 8th mode, along with a background gradient on the zeroth mode.

Setting parameter Values

- frame_save – increase for longer simulations (or hdf5 will be very large in size)
- time_stamp – automatically uses unique names for every new simulation (according to time at starting the simulation)
- save_dir – directory where you want your simulation values to be stored (automatic folder generation with name as params_time_stamp)

```
Ra_val = 1e6
Pr = 10.0
Nz = int(101)
Nn = int(50)
Nx = int(201)       # for visualization
tsteps = int(1e6)
t_write = tsteps//10
savefreq = 20
dz = 1 / (Nz - 1)
a = 2             # aspect ratio
dt = 0.6*3e-6       # time step size
frame_save = 100    # frequency of intermediate file saves


time_stamp = time.strftime("%H%M")
save_dir = f"/home/PHN_643/adarsh/FinalBeforeSubmit/Saves/{time_stamp}
os.makedirs(save_dir, exist_ok=True)
```

Stability

- IMPORTANT - set sufficiently small dt for a given Ra, Nz, Nn
- As Ra increases, so should Nz and Nn (by a sufficient amount or the code will blow up)
- And as Nz and Nn increase, dt should decrease
- Also, keep Nx (for visualization purposes), sufficiently more than the Nx (or according to aspect ratio), or again, the plot will blow up
- Check for stability using the relation $\Delta t < (\Delta z)^2/4$ (2.19) in Gary Glatzmeier . But, take dt even smaller than this as we have included non-linear terms.
- Best bet – trial and error with 2.19 as starting point (usually relevant only at high Ra)
- HUMBLE REQUEST – Check the above constraints carefully before determining that the code has an issue. Only then proceed to make changes to code (main logic affecting stability) .
- Kindly communicate to me(adarshramtel28@gmail.com) the fixes and the issue, if applicable, and if you fixed it correctly (only regarding the stability).
- Feel Free to modify rest of the code according to your own liking and as and when required.

Running the Code

- Check and import required Libraries from slide 3
- Set temperature initialization (slide 5)
- Jump to #main execution block
- Set parameter values
- Check how much of the simulation is done(via t_write)
- Observe plots and determine if the code ran successfully or require tweaking dt (or Nz,Nn)
- Move to visualizing evolution of the contours

Animations

- Another code specifically for generating a gif out of the temperature evolution.
- Only require to specify the file_path to the hdf5 file, the rest is handled by the code itself.

```
# Main execution
# Just specify the path to your HDF5 file here
h5_file_path = "/home/PHN_643/adarsh/FinalBeforeSubmit/Saves/2149_Ra_1e+06_Pr_10.0_step
```

- Can tweak the fps for smoother gifs, also the dpi for better resolution

```
# Initial plot setup
cmap_temp = 'cividis'  ##################### Colour of Temp Contour Map
temp_plot = ax.contourf(x, z, np.zeros((nx, nz)).T, 50, cmap=cmap_temp, norm=temp_norm)
```

- Only change this line (line 74) for changing style of cmap_temp (for example ‘plasma’, ‘viridis’, ‘coolwarm’,etc)
- Changing cmap style in other lines is irrelevant and will not change output.
- Might need to wait a bit for the final frame, even after it shows frame saved 1000/1000

Animations

- Try different values for frame_save from slide 6, for a smooth animation, while also not being too big of a file size.
- Both of these will depend on the how long of a simulation you are running.
- For example, in1e6 timesteps, frame_save = 100 will result in far too big size for simulation, while frame_save = 1000 can result in skipping frames and hence loss of info.
- In the animation code, can tweak fps for smoother or coarser animation frames.

Thank You

All the best for your simulations with this code,

Feel free to contact me for any information regarding the work, if you face any issues

Feel free to use any of logic and structure for the codes that you write yourself (helps me out if you provide the credit) ,

Thanking you in advance for referencing this work, if it helped you in your own work, in any way.