Project 1 - Buoyancy effects

I. BACKGROUND

Buoyancy significantly affects many flows in environmental, geophysical, and industrial applications, and can either have a stabilizing or destabilizing influence on the motion. Here, we will use Diablo - a 2D CFD solver to examine two examples of buoyancy-affected flows: internal gravity waves[1,2], and gravity currents[3,4]. The lecture notes and the references listed at the bottom of this document provide an overview of the dynamics of these flows.

FIG. 1: Left: A massive dust storm over Phoenix, Arizona is an example of a gravity current (komonews.com), Right: The surface manifestation of internal waves in the ocean, generated as stratified water flows over a sill in the Strait of Gibralter (Envisat).

II. INTRODUCTION

Diablo solves the incompressible Boussinesq equations in a 2D, x-y plane. Although this version of the code is written in Matlab and is designed to run quickly on relatively small grids, the numerics (which are described in detail in the Diablo User's guide) are sophisticated, and are taken from a functioning research code. The code is also written to be as flexible as possible, and during the computational projects, we will use it to simulate flows ranging from bioconvection to Rossby waves.

In this project, we will use Diablo to explore some buoyancy-driven and buoyancy-affected flows. We will consider flow in a horizontal/vertical slice, and will let y denote the vertical direction (even though the typical convention is to use z for the vertical direction in GFD problems). Keep in mind that while we can examine many flows in a 2D geometry, some of their properties will be different than in fully three-dimensional flow.

In its basic form, Diablo solves the incompressible, Boussinesq equations:

$$\frac{\partial \mathbf{u}^*}{\partial t^*} + \mathbf{u}^* \cdot \nabla_* \mathbf{u}^* = -\nabla_* p^* + \frac{1}{Re}\nabla_*^2 \mathbf{u}^* + Ri\, b^* \hat{g}, \tag{1}$$

$$\frac{\partial b^*}{\partial t^*} + \mathbf{u}^* \cdot \nabla_* b^* = \frac{1}{RePr}\nabla_*^2 b^*, \tag{2}$$

$$\nabla_* \cdot \mathbf{u}^* = 0, \tag{3}$$

where $\mathbf{u}^* = (u^*, v^*)$ is the 2D velocity vector, $\nabla^* = (\partial/\partial x^*, \partial/\partial y^*)$, and $\hat{g}$ is a unit vector pointing in the direction of gravity. Here $*$ indicates non-dimensional quantities, which have been normalized using a length scale, L, velocity, U_0, and buoyancy, B_0. Note that the constant density, ρ_0, has been absorbed into the definition of the non-dimensional pressure, p^*. In this case, the non dimensional Reynolds, Richardson, and Prandtl numbers are:

$$Re \equiv \frac{U_0 L}{\nu}, \quad Ri \equiv \frac{B_0 L}{U_0^2}, \quad Pr \equiv \frac{\nu}{\kappa}, \tag{4}$$

and ν and κ are the kinematic viscosity and molecular diffusivity, respectively. Note that Diablo can be easily adapted to simulate other forms of the Boussinesq equations (including using dimensional variables) by re-defining the parameters Re and Ri. For example, we could simulate the dimensional equations for momentum and density, by letting $Re = 1/\nu$, and $Ri = -g/\rho_0$.

III. DIRECTORY STRUCTURE

Inside the main Diablo directory, which you should find in your home directory, are several Matlab scripts, and several sub-directories. The Matlab scripts are all files that you will need to modify during the computational projects to pick different parameter values, boundary conditions, initial conditions, etc. The core of the Diablo code is in the *code* sub-directory. During the exercises, you might want to look through these scripts to learn more about the workings of a CFD code. For example, the script *rk_step.m* contains the main time-stepping loop and is well commented. Also within the Diablo directory are sub-directories for each of the four computational projects. Some of these have additional Matlab scripts that will be needed for a particular project. Others, like *project1* are empty, but you can use these directories to save files and figures. You also might want to save the Diablo Matlab scripts that you modified for each project in case you want to go back to look at a problem again later.

IV. GETTING STARTED

Open Matlab, change directory to the main Diablo directory, and type "*diablo*" to run the code. The parameters are set up to start with an unstable, linear density profile, with a small, random velocity field. You should see convective plumes develop and mix the density field. The code is set now to run 1000 timesteps, but you can stop it at any time by pressing CONTROL-C. As Diablo runs, it periodically calculates and saves flow variables (defined in *save_flow.m*) and some basic metrics (defined in *save_stats.m*). After running Diablo, plot some of these variables to

get familiar with the standard output. During the computational projects, you will need to add some of your own diagnostics to *save_flow.m* to calculate more than the mean and *rms* fields.

V. INTERNAL GRAVITY WAVES

Now, we will use Diablo to simulate internal gravity waves. Generally, each time we want to set up a new simulation, we will need to go through the following steps:

1. Set the flow parameters and the number of gridpoints, etc. in *set_params.m*.
2. Set the boundary conditions in *set_bcs.m*.
3. Specify the initial conditions in *create_flow.m*.
4. If we want to use a stretched grid (the default is a uniform grid), set the values in *create_grid.m*.
5. Edit *display_flow.m* to plot something useful while the simulation runs to keep track of the progress.
6. Define any metrics that you will need for post-processing and analysis in *save_stats.m*.

In this case, we just want to flip the density profile from an unstable to a stable stratification, so we can keep most of the parameters the same. For now, edit *create_flow.m* and delete (or comment) the lines specifying an unstable density profile, and uncomment the lines defining a stable stratification, including the Gaussian perturbation. You can edit these scripts from within the Matlab GUI, or in your favorite text editor. Then, edit *set_bcs.m* so that the boundary conditions on TH (the density field) match the initial conditions.

A. Free waves

Run Diablo and watch the response for a few wave periods. After the simulation is over (or after you end it) plot the density at a single point in space as a function of time (the density and time are saved in TH_save and TIME_save). You may need to improve the temporal resolution by changing how often variables are saved in *save_flow.m* - to do this, edit N_SAVE_FLOW in *set_params.m*. What is the frequency of the waves that you see? Does this match what you expected? If not, think about how the setup of these simulations might affect the wave frequency, and see if you can change the parameters to get a different result.

B. Forced waves

To continuously generate internal waves and control their frequency, we can add a forcing term to the right hand side of the momentum equations. Edit *create_flow.m* to remove the initial density perturbation (but keep the stable linear profile). Forcing terms can be easily added to Diablo by editing the script *user_rhs.m*. Here, you will find a

few lines of code to add a sinusoidal forcing bounded by a Gaussian envelope to the right-hand-side of the vertical (y) momentum equation. (You can add your own terms to the right hand side of the x-momentum, y-momentum, and scalar equations by adding a function to F1, F2, FTH, respectively). Uncomment the lines adding an internal wave forcing to F2 and run the code. You might want to edit *display_flow.m* to plot only the difference between the density, TH, and the initial condition to better visualize the waves. You should see wave crests propagate away from the source region with a preferred angle. How does this angle compare with what you expect from linear internal wave theory? Try changing the forcing frequency - how do the waves respond?

VI. GRAVITY CURRENTS

In internal gravity waves, buoyancy acts as a restoring force. In other situations, buoyancy changes can generate a flow through the conversion of potential energy into kinetic energy. One example of this are gravity currents. Here, let's create a gravity current by using Diablo to simulate a lock-exchange flow. In this example, we will need to modify the Diablo scripts more heavily, so you might want to save a copy of the ones that are in your main Diablo directory now, particularly if you want to try one of the "further investigations" below dealing with internal waves. Before you start, remove the added internal wave forcing term in *user_rhs.m* by commenting out the relevant lines.

To simulate a lock-exchange flow, we want to create a rectangular geometry, longer in the x-direction than the vertical y-direction. To do this, edit *set_params.m* with a non-dimensional x-domain size of LX=4, keeping LY=1. Since our domain is now larger in the x-direction, also increase the number of gridpoints in that direction, NX (NX=200 should be sufficient to start with, but you can try increasing this later). To prevent small scale instabilities, we also need to decrease the Reynolds number somewhat - start with a value of Re=1000. Now, we need to change the boundary conditions. Edit *set_bcs.m* to use free-slip boundary conditions at all boundaries (set the normal derivative of the tangential velocity, pressure, and density to zero, and set the normal velocity to zero on the boundary). Edit *create_flow.m* to initialize the flow. Start out with zero velocity, and with the following buoyancy (TH) field:

$$b = \frac{\Delta b}{2} \tanh\left(\frac{(x - LX/8)}{LX/50}\right), \tag{5}$$

with $\Delta b = 1$ to start.

Based on scaling arguments, a semi-infinite gravity current should propagate at a speed U, where

$$U \sim \sqrt{\Delta b H}, \tag{6}$$

and Δb is the buoyancy difference between the gravity current and the ambient fluid, and H is the gravity current height. Run Diablo for several different values of Δb, and calculate the propagation speed in each case. Does the scaling law in Eq. 6 hold?

VII. SUGGESTED FURTHER INVESTIGATIONS

A. Gravity currents in stable stratification

Try repeating the gravity current experiments, but start with a stable buoyancy profile in the ambient fluid. Start out with a case where the density of the fluid to the left of the 'lock' is equal to the densest fluid in the ambient. Can you detect internal waves propagating in the ambient fluid ahead of the gravity current? To do this, you may need to reduce the width of the lock or make the domain larger in x. It also might be helpful to create a second, passive tracer and use it to 'dye' the fluid in the lock to track the gravity current. Try varying the density in the fluid in the lock in relation to the minimum and maximum buoyancy in the stratified ambient. For example, what happens when the lock fluid has the same density as the average of the ambient fluid?

B. Internal wave reflections from sloping walls

Internal waves behave somewhat counter-intuitively when reflecting off rigid walls. Because the angle formed between the wavenumber vector and the horizontal direction depends on the wave frequency through the dispersion relation, incident and reflected internal waves do not necessarily form the same angle with a boundary. In fact, it is possible for reflected waves to be *parallel* to a reflecting surface. This latter case is called a *critical* reflection. We can't get critical reflections from vertical or horizontal walls, but it is possible to have critical reflections at sloping boundaries.

Since the geometry of Diablo is rectangular, we can't easily add a sloping boundary inside the domain. Instead, we can simulate internal waves reflecting from sloping boundaries by tilting the entire domain. To do this, you will need to change the direction in which gravity points (change GRAV_X, etc. in *set_params.m*), change the density profile to have flat isopycnals in the rotated reference frame, and change the boundary conditions to be consistent with the initial conditions. The paper by Slinn and Riley[5], referenced below might be helpful in visualizing the setup. Adjust the frequency of the internal wave forcing to generate critical reflections from the now sloping boundaries. What happens to the wave energy in this case?

VIII. REFERENCES

[1] B. Sutherland, *Internal gravity waves* (Cambridge Univ. Pr., 2010).

[2] C. Staquet and J. Sommeria, Annual review of fluid mechanics **34**, 559 (2002).

[3] J. Simpson, Annual Review of Fluid Mechanics **14**, 213 (1982).

[4] M. Flynn and B. Sutherland, Journal of Fluid Mechanics **514**, 355 (2004).

[5] D. Slinn and J. Riley, Theoretical and computational fluid dynamics **11**, 281 (1998).