Buoyancy Simulation

A Conceptual Introduction to Vertical Motion in Stratified Fluids

Francisco Machín
francisco.machin@ulpgc.es

November 23, 2024

Conceptual Framework

This exercise illustrates the vertical motion of an object immersed in a stratified fluid, where density varies with depth. The motion is governed by the balance between buoyant and gravitational forces. The buoyancy force depends on the density difference between the object and the surrounding fluid, and in this case, the fluid is assumed to follow a stable density stratification characterized by the Brunt-Väisälä frequency squared N^2.

Two scenarios are considered:

- **No friction:** the object oscillates indefinitely about its equilibrium depth with constant amplitude.
- **With friction:** the oscillations decay over time, and the object gradually settles at equilibrium.

The vertical acceleration is governed by Newton's second law:

$$\frac{dw}{dt} = -g\left(\frac{\rho_o - \rho(z)}{\rho_o}\right) - rw,$$

where w is vertical velocity, g is gravitational acceleration, ρ_o is the object's density, $\rho(z)$ is the ambient fluid density at depth z, and r is the friction parameter.

The background stratification is defined as:

$$\rho(z) = \rho_{ref}\left(1 - \frac{N^2 z}{g}\right),$$

with ρ_{ref} the surface reference density.

This example is inspired by Exercise 3 of Jochen Kaempf's *Ocean Modelling for Beginners*. The theoretical framework behind this simulation is detailed in his book.

Code and Animation

- **Code available at:** https://bit.ly/OOM_Buoyancy
- **Animation available at:** https://www.youtube.com/watch?v=9ZQaudulkRk

Description

This Python simulation solves the vertical motion of a buoyant object using an explicit time-stepping scheme. The user can adjust the object density, fluid stratification (N^2), and damping (r) to explore different dynamical behaviors. The simulation outputs an animation that shows the object's depth as a function of time, highlighting the nature of its oscillatory or damped motion. The equilibrium depth is indicated by a dashed line.

The exercise also includes alternative implementations in Fortran and Scilab, separating the numerical integration from the visualization. The Python version offers a streamlined and self-contained workflow.

Acknowledgment: A huge thanks to Jochen Kaempf for his incredible contribution to ocean modeling. If you're interested in his book, contact him via ResearchGate for a copy.