Sediment Transport under Landslide-Induced Wave in Reservoir
Biweekly Report – June 14th, 2021

Alexandre Erich S. Georges - Graduate Student Researcher
University of California, Berkeley

Project Background. This numerical study will be an extension of my Spring 2021 CEE200B Final Project which used the provided MATLAB CFD code to study sediment transport and deposition in a reservoir entrance. The provided code will first be converted to Python for a better work environment and sediment transport at different locations in a reservoir will be looked at under a landslide-induced surge wave. An upgrade to a 3D model will be of the order.

Accomplishments.

1. Conversion of CE200B code from MATLAB to Python

The MATLAB code for my CE200B Sediment Transport project has been converted to Python. All functions and behaviors have been translated to the exception of the implementation of a reservoir entrance, which might be handled by 3rd-party models for the time being. The major difference from the MATLAB code here is the implementation of an Object-Oriented approach.
The new Python code is hosted on GitHub and split in two branches:

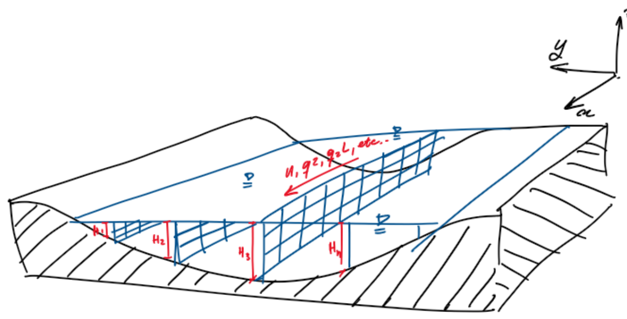
- Main branch which will serve for the remainder of this project:
https://github.com/AlexandreSeb97/Sediment-Transport-in-Reservoir-under-Landslide-Induced-Surge-Wave
- Secondary branch which will keep code relevant to the CE200B material:
https://github.com/AlexandreSeb97/Sediment-Transport-in-Reservoir-under-Landslide-Induced-Surge-Wave/tree/ce200b-material

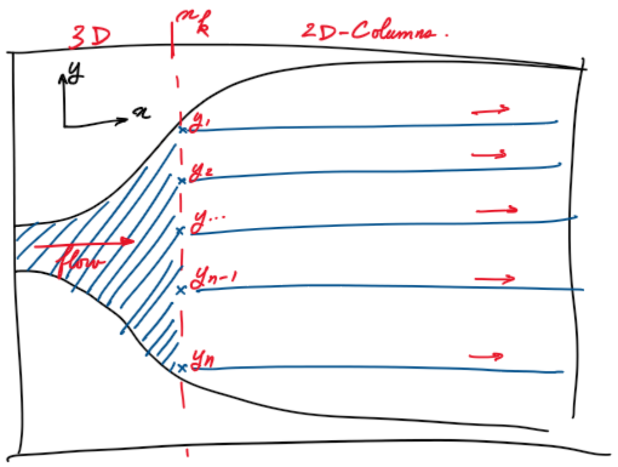
2. Objected-Oriented Implementation

The column and particle components are implemented as object classes to facilitate instantiating several of them at the same time. This is extremely useful for tracking particles as particle properties such as position, diameter and specific density can be randomized and easily kept track of. The Column component being an object makes it easy to instantiate case studies with parameters and even make different instances at the same time, letting us track particles at different locations, as we will see in the next accomplishment.

3. Ideas on how to implement particle tracking in reservoir.

To track particles throughout the reservoir, I intend to instantiate several 2D columns and track particles over several 2D planes, giving us a general idea of particle movement without making a full 3D simulation of the reservoir which would computationally costly.





Discussions and Questions.

- Implementing a system of 2D columns, while useful for reducing computational cost could be problematic in the case of a period in our wave, particularly if we proceed with computing the reservoir entrance with a 3rd party 3D solver.
- Why exactly is the Kluge condition checking for smaller than 1e-06 and not smaller than zero? Aren't we missing out for small but relevant turbulence?
- While debugging abnormal particle tracking, I have made the following noteworthy observations:
  - The "pulse" pattern that arose in our particle tracking seems to come from a numerical instability in solving the turbulent length scale, as once turbulence mixing is enabled, velocity and by extension particle movement present the pattern. This has been temporarily fixed by changing to the following parameters that are stable:
    N = 80, H = 10m, dt = 60s, px0 =0.0001
  - The "shooting up" and systematic accumulation of particles at the top boundary layer was due to the wrong NumPy function being used for computing randomness in the particle z position in:

    ```
    z = self.z - ws*dt + math.sqrt(2*abs(kzp[index])*dt)*np.random.randn()
    ```

    Previously I used np.random.rand() which only generated random numbers between 0 and 1, making the particle only be able to go up randomly. In contrast, np.random.randn() is a normal distribution and can generate negative numbers as well.

Next Objectives.

From this, the following next objectives will be pursued for the next week:

- Perform a Van Neumann stability analysis to find way to optimize parameters, preventing unstable conditions when instantiating several columns of different properties at the same time.
- Expand the Column model to a 2D one, adding an x-dimension additionally to the z-dimension. This will involve implementing solving of velocity in V.
- Look into 3rd party 3D Flow solvers such as Delft3D and FLOW-3D.
- Implement wave mechanics and under-wave fluid dynamics in our columns.