Simulation of a Cubic Floating Body Suspended in Viscous Fluid

Jackson Oleson[1]

Abstract
In this project, I used MATLAB to simulate the dynamics of a rigid cubic body interacting with a viscous fluid. The objective was to investigate rigid body dynamics when approximated by a discrete set of point masses. These points are distributed across the surface of the body, which, for simplicity, has no internal mass. This approximation is justified because real-world floating bodies, such as buoys, are often hollow, with their mass concentrated on the surface of the object. Additionally, contact forces act solely on the object's surface, and gravitational forces are evenly distributed.
Increasing the number of surface points relative to the size and mass of the object enhances the accuracy of the approximation, albeit at the cost of computational efficiency. This trade-off is central to balancing precision with performance in the simulation.

[1] *Department of Mathematics, New York University*
jo1449@nyu.edu

Contents

Introduction 1
1 Methods 2
1.1 Buoy Setup . . . 2
1.2 Equations of Motion . . . 2
1.3 Numerical Methods . . . 3
1.4 Validation . . . 3
2 Results and Discussion 4
2.1 Results . . . 5
Acknowledgments 5
References 5

Introduction

The simulation employs a numerical method, breaking the system's evolution into discrete time steps. Each step computes small changes in the system's state, with greater accuracy achieved by increasing the number of time steps—albeit at the cost of higher computational complexity.

To simulate the cubic buoy, I track the positions of a set of points in MATLAB within 3D space. These points are fixed relative to one another and to their shared center of mass, allowing them to be treated as a single rigid entity. Each point is assigned a mass and an 'area,' calculated as the total surface area divided by the number of points. These properties enable the calculation of the forces acting on each point at every time step, denoted as dt.

The body is initialized using a function called 'buoy', which generates the list of points. Initially, the it is centered at $x, y, z = (0, 0, 0)$, with the water level set at $z = 0$.

I begin by initializing the mass, volume, and velocity of each point, all of which can be adjusted as needed. In addition to gravity, pressure forces act on points submerged beneath the water's surface. These forces are contact forces that oppose the outward normal force exerted by each point, which act in the direction perpendicular to the face of the cube on which the point lies. While assigning a single normal direction per face would also work, maintaining a list of normal directions for individual points was conceptually simpler, easier in implementation, and did not affect the results.

The pressure forces not only push against the buoy but also create torque, causing it to rotate. The resulting rotation is governed by the cube's total angular velocity, Ω, about its center of mass. These pressure forces are counteracted by a damping force, which acts opposite to the outward normal direction. This damping force is proportional to velocity and is only applied to points submerged in water $(z \leq 0)$. Points above the water's surface are excluded, as are air-based pressure and damping forces, which are negligible compared to those exerted by water and do not significantly influence the system under study.

While actual damping forces include tangential and shear effects that counteract rotational torque, modeling these accurately is challenging. After exploring discussions on **Stack Exchange**, I learned that the transfer of velocity from the body to the surrounding fluid complicates calculations for

this scenario. As a practical substitute, I define a simplified damping force that effectively reduces the torque caused by pressure forces and slows the cube's angular momentum. Over time, this damping force brings the system to rest, providing a reasonable approximation of real-world behavior."

1. Methods

1.1 Buoy Setup

I implemented a function that takes the side length of the cube, the number of points along each edge, and the total mass as input parameters. The function outputs several key components: a list of (x,y,z) coordinates for the points, a corresponding list of masses, and a list of areas. Additionally, it provides the total number of points and the 3D coordinates of the center of mass, which is fixed relative to the points.

Although the center of mass itself does not possess any mass and does not experience any forces, its position is crucial for calculating the absolute positions of each point. The absolute position of a point is determined by adding its position relative to the center of mass to the true position of the center of mass.

Each point is assigned an area representing a small square slice of the cube's surface. For instance, if the side length of the cube is $2m$ and there are 5 points along each edge, the area assigned to a single point is $\left(\frac{2m}{5}\right)^2$. All points are assigned equal areas, ensuring that the sum of all individual areas equals the cube's total surface area.

Along with the spatial coordinates, each point is also assigned its 'outward normal direction' determined by the face on which the point lies. This normal direction is represented as a direction vector, which is updated for each point as it rotates. Although, in an actual cube, we would only need to know the six outward normal directions for each face, keeping track of a list of normal directions—one for each point—was conceptually simpler and easier to implement. This approach has the same effect, as the normal directions are simply direction vectors, and each point maintains its associated normal, allowing for the correct calculation of forces.

Edge Cases One conceptual challenge was handling the edge cases. While the direction of the forces applied to points lying on each face is clear, it is less straightforward to handle the pressure forces at the edges, where two faces intersect, and even more complex at the vertices, where three faces converge. To address this, I excluded those points from the calculation and only considered the points that lie internal to the edges of the cube. The cubic buoy is constructed face-by-face. Each of the six faces is independently constructed by assigning 3-D coordinates to each point relative to the origin, which serves as the initial center of mass. Figure 2 depicts the representation.

1.2 Equations of Motion

Consider a collection of discrete point masses M_k for $k \in \{1,...n\}$ such that $\sum_{k=1}^{n} M_k = M$. If we define the position of the center of mass of the system as X_{cm} and X_k as the position of point k relative to the center of mass, then $MX_{cm} = \sum_{k=1}^{n} M_k X_k$. We can also define the velocity of the center of mass as $\frac{dX_{cm}}{dt} = U_{cm}$. Since our body is rigid, the torque on element j by element k is equal to the negative of the torque on element k by element j, for any two arbitrary elements $j,k \in \{1,...n\}$, then the total torque on these elements sums to 0. Since these two elements are arbitrary, then the total torque of the system sums to 0, we are then left with $M\frac{dU_{cm}}{dt} = \sum_{k=1}^{n} \vec{F}_k = \vec{F}$, where the term $\vec{F}_k$ denotes the total force applied on point k at time t. The system has three forces acting on each point that determine its motion:

$$F_k = F_{buoyancy} - F_{gravity} - F_{damp} \tag{1}$$

- $F_{bouyancy} = \sum_{k=1}^{n} \rho * g * (area_k) * h_k$
  - h_k = distance below the surface of point k
  - $area_k$ = "area" given to point k
- $F_{gravity} = \sum_{k=1}^{n} M_k * g$
- $F_{damp} = -d * \vec{v}_k * area_k$
  - d = damping constant
  - $\vec{v}_k$ = velocity vector
  - Opposes direction of motion

We define the angular momentum of the system at time t as:

$$\vec{L}(t) = \sum_{k=1}^{n} M_k (X_k - X_{cm}) \times (U_k - U_{cm}) \tag{2}$$

We also define the total torque of the system as:

$$\vec{\tau} = \frac{d\vec{L}}{dt} = \sum_{k=1}^{n} (X_k - X_{cm}) \times \vec{F}_k \tag{3}$$

Because the system is a rigid body, these equations determine its motion. The velocity of the system is determined at any time t by the velocity of the center of mass $U_{cm}(t)$ and the angular velocity $\vec{\Omega}(t)$ of the body about its center of mass. The individual point velocities are given by

$$\vec{U}_k(t) = \vec{U}_{cm}(t) + \vec{\Omega}(t) \times (\vec{X}_k - \vec{X}_{cm}) \tag{4}$$

In matrix notation, this can be written as $\vec{L}(t) = I(t)\vec{\Omega}(t)$. Where $I(t)$ is the moment of inertia tensor. Because each point lies on the edge of the cube, the moment of inertia will be a scalar multiple of the identity matrix.[1]

[1] Add hyperlink: https://physics.stackexchange.com/questions/105229/tensor-of-inertia-of-a-hollow-cube

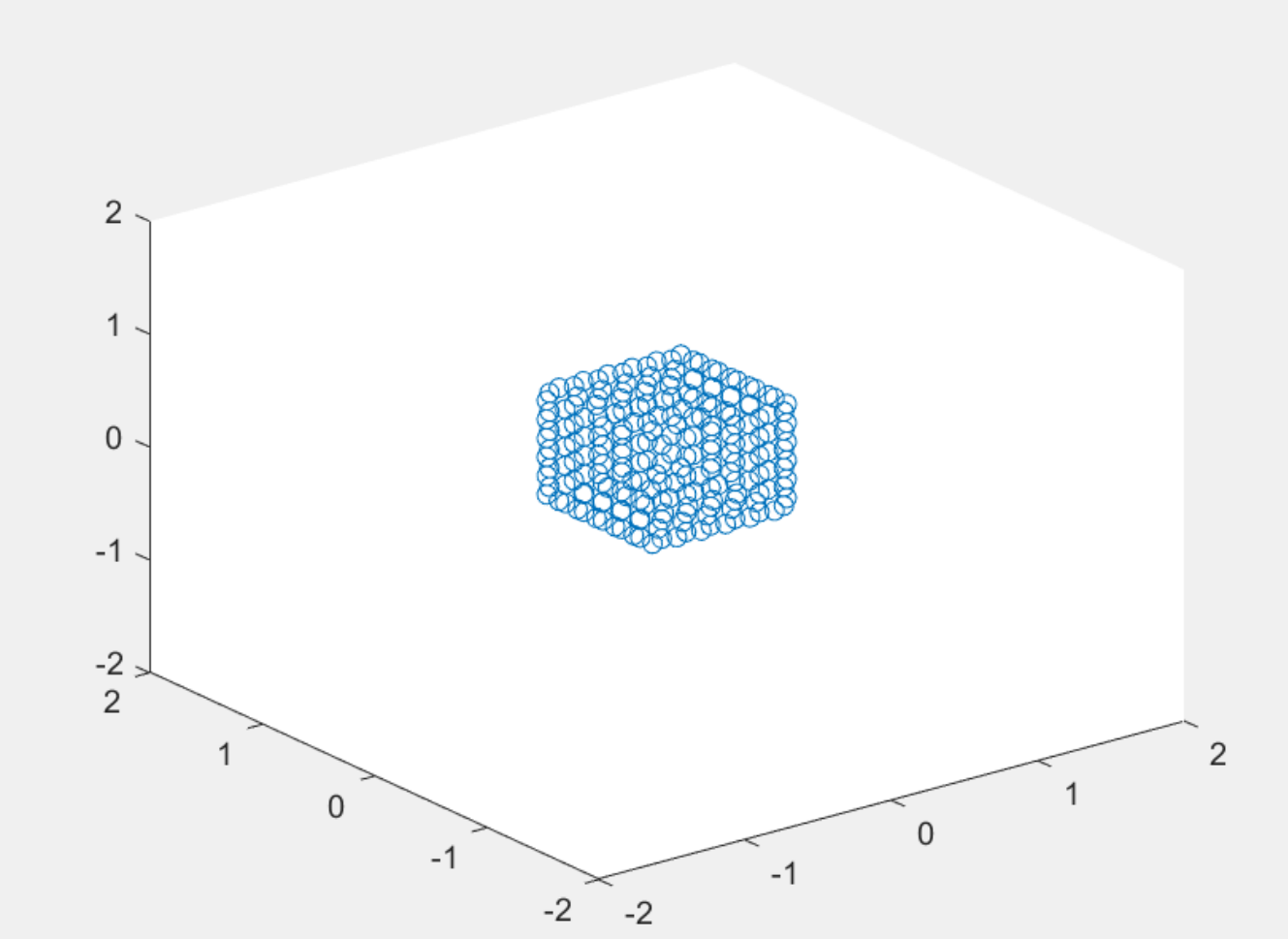

Figure 1. The Rigid Cubic Buoy is plotted as a collection of discrete points

1.3 Numerical Methods

I define a numerical method for updating the system as follows:

The position of the center of mass is updated like so:

$$\vec{X}_{cm}(t+\Delta t) = \vec{X}_{cm}(t) + \Delta t * \vec{U}_{cm}(t) \tag{5}$$

The velocity of the center of mass is updated similarly:

$$M\frac{\vec{U}_{cm}(t+\Delta t) - \vec{U}_{cm}(t)}{\Delta t} = \sum_{k=1}^{n} \vec{F}(t+\Delta t) \tag{6}$$

The total angular momentum of the system is updated by:

$$\frac{\vec{L}(t+\Delta t) - \vec{L}(t)}{\Delta t} = \sum_{k=1}^{n} \tilde{X}_k(t+\Delta t) \times \vec{F}_k(t+\Delta t) \tag{7}$$

Where $\tilde{X}_k = \vec{X}_k - \vec{X}_c m$ is the relative position of point k from the center of mass. Because there is no torque on the center of mass, we can update the relative positions of the points $\tilde{X}$ when a rotation is applied:

$$\tilde{X}_k(t+\Delta t) = R(\vec{\Omega}(t), \Delta t)\tilde{X}_k(t) \tag{8}$$

Where R defines an exact rotation about the center of mass, at angular velocity $\vec{\Omega}$ over a finite time interval Δt.[2]

Finally, the list of 'normal directions' is given the same rotation as the relative positions $\tilde{X}$ so that the forces applied on them

1.4 Validation

It is important to note that calculating the total energy of the system is challenging. The buoy is conceptualized as floating in an 'infinite' body of water, making it impractical to compute the displacement of the water itself. However, I am able to track and display other important quantities such as the total forces acting on the system, the location of the center of mass, and the angular momentum of the cube.

The correctness of the forces applied on the cube can be verified by initially placing it at the depth at which it will eventually come to rest. Given the cube has the same density as ice: $900\frac{kg}{m^3}$, the total submerged volume calculates to $\frac{900}{1000}$. At rest, the center of mass should lie at $z = (-0.9) + (0.5) = (-0.4) \rightarrow (0, 0, -0.4)$. Figure 3 confirms this result. However,

[2] For further information on the derivation of R, see Professor Peskin's Rigid Body notes which I will reference below

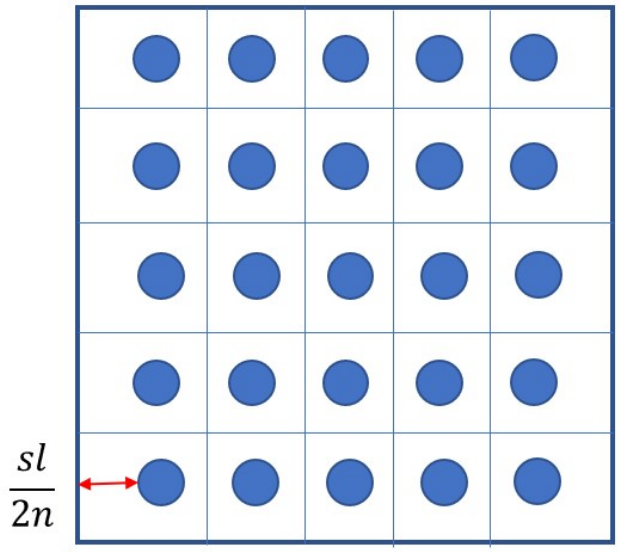

Figure 2. Discrete point method can very nearly approximate the surface. Adding more points will more nearly approximate the true system but will increase computational complexity

in Matlab, a very small initial velocity must be added to avoid division by zero when calculating the rotation R of each point. This small initial velocity causes the observed oscillation about -0.4.

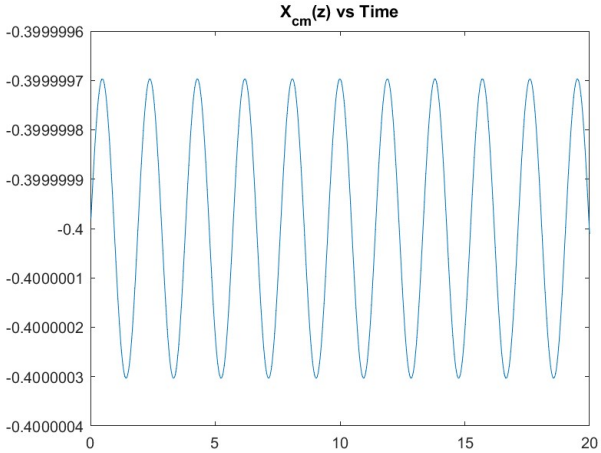

Figure 3. Small Oscillations of X_{cm} about $z = -0.4$

Next I will demonstrate that the $2-$Norm of angular velocity of the system remains constant with respect to time. To achieve this, I introduce an initial angular velocity $\vec{\Omega} = [-1,1,0]$ to create a tiny perturbance at the simulation's beginning. I maintain the mass at $900\frac{kg}{m^3}$ and the initial center of mass position at $(0,0,-0.4)$ to isolate the angular velocity. Figures 4 and 5 display a graph of $||\vec{L}||_2$ over time. Notably, it increases about halfway through, but only by a tiny amount.

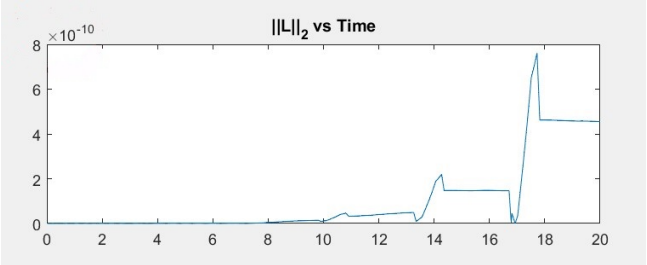

Figure 4. $||\vec{L}||_2$ vs Time

I decided to make the time step smaller, just in case this increase was meaningful.

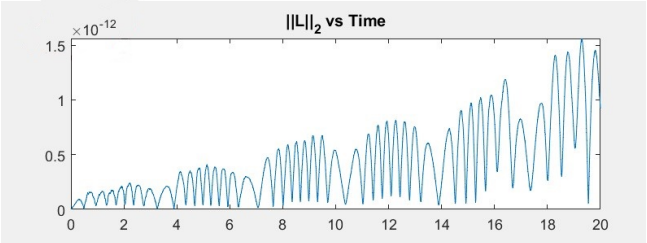

Figure 5. $||\vec{L}||_2$ vs Time

This resulted in an even smaller change in angular momentum.

Finally I check that the total forces sum to zero when the body is at rest. Keeping the cube at its resting position, the gravity force should equal the buoyant force, while the pressure forces in the x and y directions should also sum to zero. Figure 6 shows the 2-Norm of the total forces.

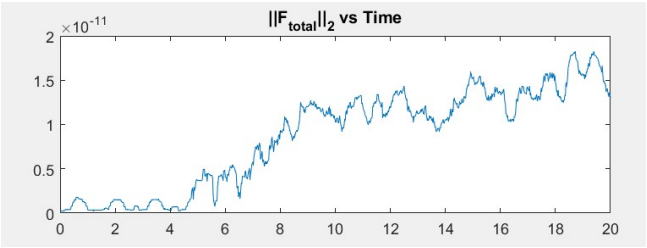

Figure 6. Very tiny changes in $||F_{total}||_2$ over time

2. Results and Discussion

Initial Challenges This project posed significant challenges at every stage. From the outset, the primary task was to model a continuous pressure force. A cube, represented as a discrete set of points, was chosen as the foundational shape due to its simplicity in calculating force directions and its straightforward implementation. While the cube proved effective, exploring alternative shapes could offer intriguing possibilities for future work.

After setting up the buoy, the next challenge was ensuring the system's motion behaved as expected. Initially, I encountered issues with aligning the forces correctly. This was ultimately traced to an error in the directions where the forces were applied, which was easily resolved once identified. At first, I attributed the problem to numerical error—an issue that persists but remains within acceptable limits.

2.1 Results

Once the simulation was successfully operational, a natural next step—and a method of validation—was to study the effects of adjusting the density. Increasing the density caused the cube to sink and disappear from view, while decreasing it led to the cube bouncing above the surface. The most compelling scenario occurred when the cube's density approximated that of ice $\left(900\frac{kg}{m^3}\right)$. In this case, its motion closely resembled that of an ice cube dropped into water, as expected. Without a damping force, this oscillatory motion persisted until the simulation timed out.

Introducing a damping force allowed the motion to slow down gradually, eventually bringing the system to rest. At equilibrium, the cube settled in a slightly rotated position, offset from its initial alignment with the *xyz*-axes. Below are visualizations of the cube's resting position, along with the 2-norms of angular momentum and force, shown for two different damping constants: 0.1 and 0.25.

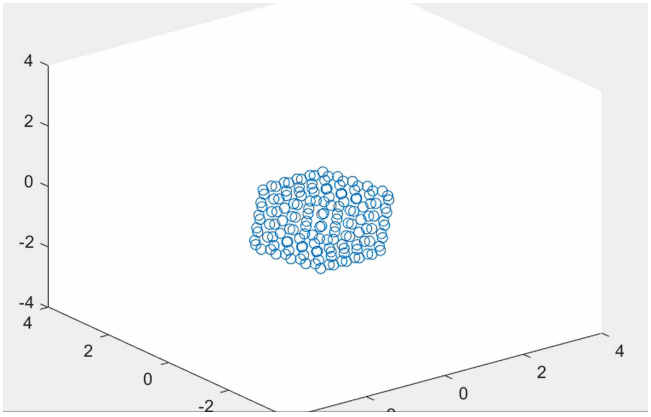

Figure 7. Resting Position of the Cube

Conclusion

This project was both highly engaging and challenging. The majority of the effort—around 99%—was spent on setup and debugging, which has been the primary focus of this write-up. However, once the cube was fully operational, it was incredibly satisfying and fascinating to observe its simultaneous rotation and bobbing motion. Adjusting the initial angular velocity influenced the direction of rotation, while changes to the damping constant primarily affected how quickly the

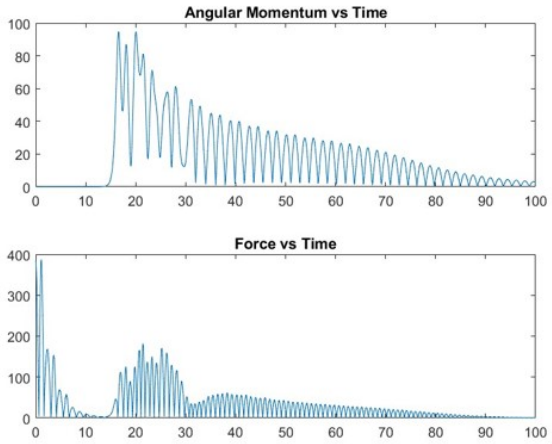

Figure 8. $||\vec{L}||_2$ and $||\vec{F}_{total}||_2$ vs Time for Damping Constant of $d = 0.1$

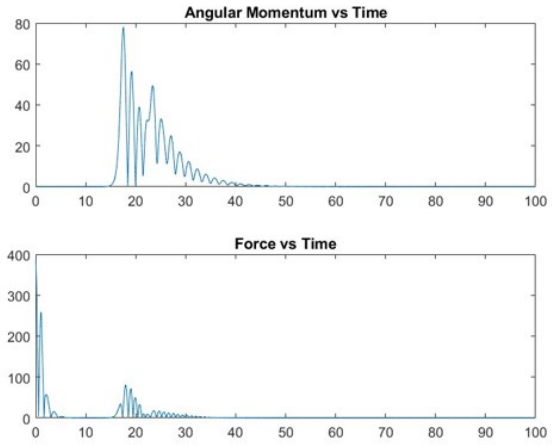

Figure 9. $||\vec{L}||_2$ and $||\vec{F}_{total}||_2$ vs Time for damping constant $d = 0.25$

system came to rest. Altering the density mainly impacted the position of the cube's center of mass.

To see the system in action, I've provided a **link** to a YouTube video showcasing the simulation. The video is over five minutes long, so feel free to skip through sections as you like

References

Professor Peskin's Rigid Body Notes: `https://www.math.nyu.edu/~peskin/modsim_lecture_notes/rigid_body_motion.pdf`

StackOverflow Article on Moment of Inertia Tensor for Hollow Cube: `https://physics.stackexchange.com/questions/105229/tensor-of-inertia-of-a-hollow-cub`