Contents

1 Introduction

Many negative effects that can result when surface water is polluted. Harmful health effects, damaged ecosystems and expensive cleanup costs are just a few of the effects that can occur when surface water is contaminated. A model of how a surface contaminant diffuses through a fixed volume of water was constructed to determine the amount of time that is required for a fixed concentration of a contaminant to be completely distributed throughout the system. An analytical model was determined using the Law of Conservation of Mass and Ficks Equation of Diffusion. The model was further refined using a computer simulation (C++) based on the probability of the contaminant to diffuse to different point in the system. Finally, the model was compared to experimental data collected when a drop of food coloring was dropped from a specific height and the time required for total diffusion was measured. The model and the experimental data were found to be [consistent/inconsistent (will add after experiment is completed)].

2 Data Gathering

2.1 Starch Procedure

1. Fill rectangular container with regular tap water at room temperature.
2. Mix in (1,2,3,4,5 Tablespoons of corn starch)
3. Measure viscosity using $\frac{2(p_s - p_i)ga^2}{9v}$
4. Drop 1 unit of food coloring at top center, start timer.
5. Record time when each important point in the container is effected.

2.2 Data

(We will insert the data at a later point)

3 Analysis

3.1 Assumptions

1. Fixed volume of water and fixed amount of food coloring
2. No manual mixing of water
3. No manual mixing of the mixing solution
4. Follows Fick's law of diffusion
5. Follows the law of conservation of mass
6. Follows Stoke's Law for drag
7. Water and mixing solution maintains an ideal solution

3.2 Derivation of Diffusion Equation

Research shows that one method of modeling the spread of the food coloring through the body of water is through the diffusion equation. The following derivation is based off of the derivation in the second edition of the textbook Environmental Fluid Mechanics Part I: Mass Transfer and Diffusion by Scott A. Socolofsky and Gerhard H. Jirka. (Socolofsky)

The diffusion equation is derived from the law of conservation of mass which states that the change in mass of the concentration of contaminant is equal to the total number of units into the system minus the total number of units leaving the system.

$$\frac{\partial M}{\partial t} = \sum m_{in} - \sum m_{out} \tag{1}$$

The mass fluxes into and out of the system along the x-axis can be modeled with Fick's Law where D is the diffusion constant.
Fick's Law:

$$q_x = -D\frac{dC}{dx} \tag{2}$$

Diffusion Constant:

$$D = \frac{kT}{6\pi\mu\gamma} \tag{3}$$

Where, k = Boltzmann constant, T = temperature of fluid, μ = viscosity of fluid, γ = radius of diffusion particle.

$$q_{x,in} = D\frac{\partial C}{\partial x}_1 \tag{4}$$

$$q_{x,out} = -D\frac{\partial C}{\partial x}_2 \tag{5}$$

In order to obtain a total mass flux m, q_x can be multiplied by the surface area of the system $A = \delta y \delta z$ and then by adding equations 4 and 5. Therefore, the total net flux in the x direction is given by

$$\delta m_x = -D\delta y \delta z(\frac{\delta C}{\delta x}_1 - \frac{\delta C}{\delta x}_2) \tag{6}$$

Now, a method to evaluate $\frac{\partial C}{\partial x}$ is sought. To begin, a Taylor Series expansion is used.

$$f(x) = f(x_0) + \frac{\partial f}{\partial x}_{x_0} + ... \tag{7}$$

Substituting equation 8 into equation 6 for the total net flux yields the following equation.

$$\delta m_x = D\delta y \delta z \frac{\partial^2 C}{\partial x^2}\delta x \tag{8}$$

Similarly, the equations modeling the total net flux in the y and z direction can be represented by the following equations.

$$\delta m_y = D\delta x \delta z \frac{\partial^2 C}{\partial y^2}\delta y \tag{9}$$

$$\delta m_x = D\delta x \delta z \frac{\partial^2 C}{\partial z^2}\delta z \tag{10}$$

It can be noted that $\delta x \delta y \delta z$ is simply a constant M. Rewriting the equation with an M and then substituting into the conservation of mass yields the heat equation.

$$\frac{\partial C}{\partial t} = D(\frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2} + \frac{\partial^2 C}{\partial z^2}) \tag{11}$$

Equation 12 can be re-written with the following notation.

$$\frac{\partial C}{\partial t} = D\nabla^2 C \tag{12}$$

3.3 Model with Diffusion Equation

This model looks at the following equation and boundary conditions to model the spread of food coloring throughout the container.

$$\frac{1}{D}\frac{\partial C}{\partial t} = \frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2} + \frac{\partial^2 C}{\partial z^2} \tag{13}$$

... on the cube $0 < x < a, 0 < y < b, 0 < z < c$
Boundary Conditions: $u'(0, y, z, t) = u'(a, y, z, t) = u'(x, 0, z, t) = u'(x, b, z, t) = u'(x, y, 0, t) = u'(x, y, c, t) = 0$
Initial Condition: $u(x, y, z, 0) = (x - \frac{a}{2})^2 + (y - \frac{b}{2})^2 + (z - c)^2 = r^2$

The Neumann Boundary Conditions represent a closed container where nothing is entering or leaving the system at these boundaries. The initial condition represents the placement and amount of initial food dye added. This initial condition assumes the drop will be added to the middle of the container at the very top of the boundary. Because the drop will be added from a low height, it will be assumed that the initial velocity of the drop is negligible and it can be reasonably assumed the initial position of the center is on the plane z = c and diffusion will begin when the drop reaches this point. The radius of the droplet can be measured based on the diameter of the pipet.

Integration by parts can be used to solve the differential with the given boundary and initial conditions to obtain the following general solution. ()

$$u(x, y, z, t) = \sum_{m=0}^{\text{inf}} \sum_{n=0}^{\text{inf}} \sum_{k=0}^{\text{inf}} a_{m,n,k} cos(\frac{m\pi x}{a}) cos(\frac{n\pi y}{b}) cos(\frac{k\pi z}{c}) e^{D[(\frac{m\pi}{a})^2 + (\frac{n\pi}{a})^2 + (\frac{k\pi}{a})^2]t} \tag{14}$$

... more added later.

3.4 Results

4 Simulation

4.1 Prototype Simulation

4.1.1 Assumptions

1. The solution is mixed if a few select units at the bottom of the cube have an amount greater than 0.

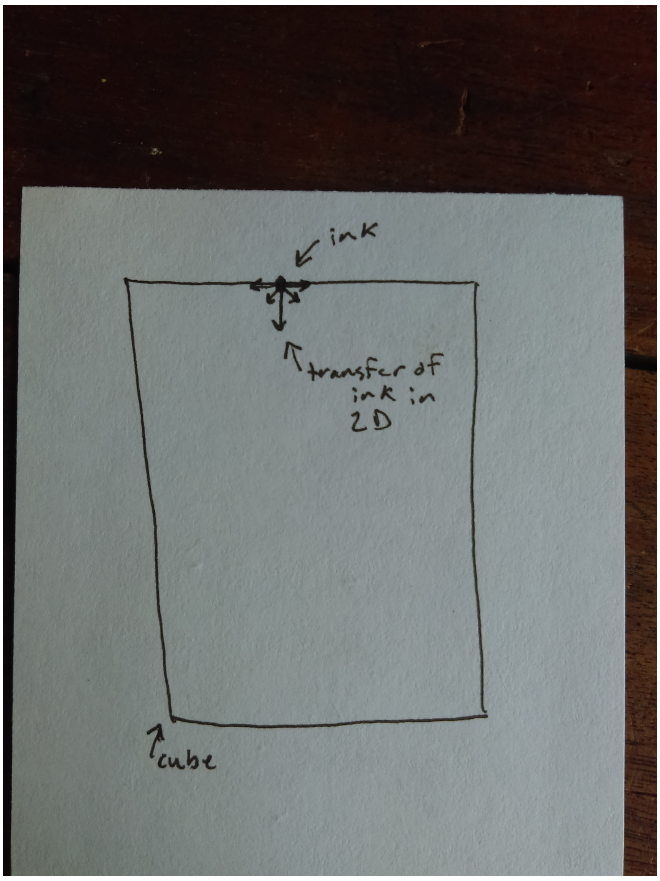
2. Every addition of corn starch will slow down the mixing rate linearly.

4.1.2 Code

The prototype simulation generated a series of basic steps that distributed the mixing solution to all surrounding horizontal units and surrounding units that are below the original. As demonstrated here:

```
/*
    BASE SIMULATION - measuring the time it takes for the mixing solution to full spread.

    @param temp    the tempeture of the fuild to determine the transfer delay
    @param thickness    the thickiness of the fuild (with an increase in corn starch).
*/
chrono::seconds base_Simulation(float temp, float thickness) {
    /*
        How fast the color spreads will be determined by the experiment(s).

        The transfer speed will increase/decrease based on a weighted amount of the given tempeture/thickiness.
    */
    double weight_temp = .0001;
    double weight_thick = .01;
    double directDown = .05 + (temp*weight_temp) - (thickness*weight_thick);
    double sideDown = .025 + (temp*weight_temp) - (thickness*weight_thick);
    double sideUnit = .0125 + (temp*weight_temp) - (thickness*weight_thick);

    auto start_timer = chrono::high_resolution_clock::now();
    bool notFullyMixed = true;
    while (notFullyMixed) {
        for (int z = 0; z < Z_DIM - 1; z++) {
            for (int x = 1; x < X_DIM - 1; x++) {
                for (int y = 1; y < Y_DIM - 1; y++) {
                    float dot = cube[x][y][z];
                    if (dot > 0) {
                        /*
                            Unit directly below current unit
                        */
                        cube[x][y][z + 1] += dot*directDown;
                        /*
                            Unit next to direct-below unit
                        */
                        next_to_direct_below(x, y, z, dot, sideDown);
                        /*
                            Unit to the side of current unit.
                        */
                        side(x, y, z, dot, sideUnit);
                        /*
                            Remove transfer from original
                        */
                        cube[x][y][z] -= dot*directDown + dot*sideDown * 8 + dot*sideUnit * 8;
                    }
                }
            }
        }
        if (cube[(X_DIM / 2) - 1][(Y_DIM / 2) - 1][Z_DIM - 1] && cube[0][Y_DIM - 1][Z_DIM - 1]
            && cube[X_DIM - 1][Y_DIM - 1][Z_DIM - 1] && cube[X_DIM - 1][0][Z_DIM - 1] && cube[0][0][Z_DIM - 1]) {
            notFullyMixed = false;
        }
    }

    auto end_timer = chrono::high_resolution_clock::now();
    return chrono::duration_cast<chrono::seconds>(end_timer - start_timer);
}
```

Visually the simulation behaved like so:



With each direction having a specific transfer rate. Each direction's transfer rate was determined by observing the mixture of ink and water during our experiment. We noticed the majority of the ink moved straight downward more than any other direction and so the simulation reflected that behavior by having the unit directly below take the majority of the ink from the original unit.

4.1.3 Results

In order to compare the results of this simulation and the results of the actual data, we had decided our measure of success could be viewed as a minimal difference between the simulation's duration and the actual duration until

the mixing solution had reached the bottom corners of our container/cube. After we collected data from our 1st and 2nd experimental session, we had realized our data turned out unreliable to use because of too many changing variables during each test. As a result the weights were configure to match only the time it took the ink to mix with the bottom area in plain water at room temperature.

4.2 Naiver-Stokes Inspired Simulation

4.2.1 Assumptions

1. The naiver-stokes inspired simulation will make it easier to fit the actual data.

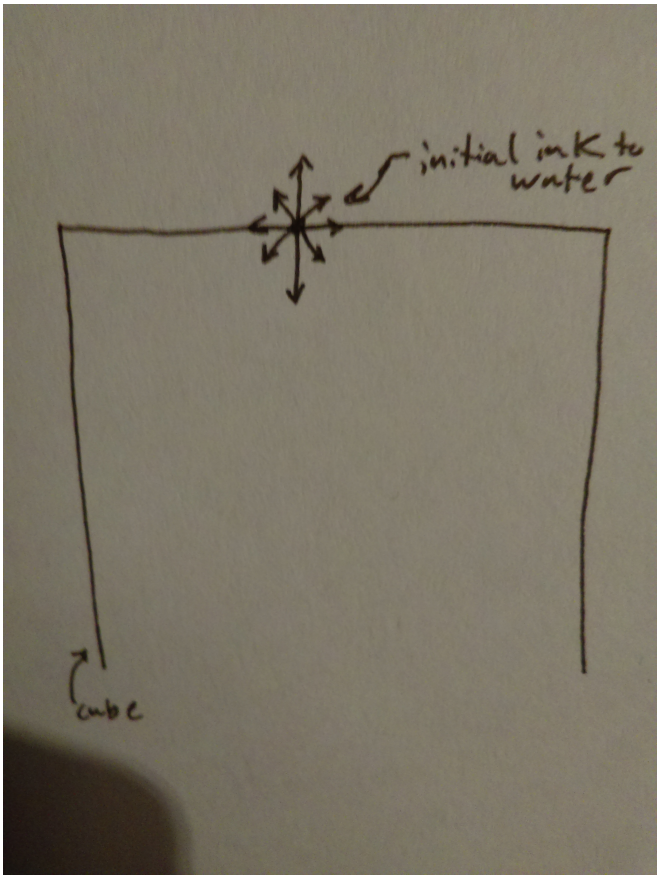
2. The naiver-stokes inspired simulation will mimic a more realistic environment even though the bottom corners are the only observed units.

4.2.2 Code

The goal for this simulation was to apply the true Naiver-Stokes algorithm to the experiment and record what happens. During the many hours studying different papers on the application of the Naiver-Stokes we realized it isn't the easiest algorithm to deal with and it isn't really useful unless you have a GPU to handle all of simultaneous computations. (Daniels) Crunched on time, we decided to come up with an improved prototype model that's inspired by the velocity of fluids found in the Naiver-Stokes model (Xu). The code below demonstrates that combination:

```
chrono::seconds navier_Stokes(float temp, float thickness) {
    /*
    How fast the color spreads will be determined by the experiment(s).

    The transfer speed will increase/decrease based on a weighted amount of the given tempeture/thickiness.
    */
    double weight_temp = .0001;
    double weight_thick = .01;
    double directUp = .037 + (temp*weight_temp) - (thickness*weight_thick);
    double sideUp = .001 + (temp*weight_temp) - (thickness*weight_thick);
    double directDown = .06 + (temp*weight_temp) - (thickness*weight_thick);
    double sideDown = .02 + (temp*weight_temp) - (thickness*weight_thick);
    double sideUnit = .001 + (temp*weight_temp) - (thickness*weight_thick);

    auto start_timer = chrono::high_resolution_clock::now();
    bool notFullyMixed = true;
    while (notFullyMixed) {
        for (int z = 1; z < Z_DIM - 1; z++) {
            for (int x = 1; x < X_DIM - 1; x++) {
                for (int y = 1; y < Y_DIM - 1; y++) {
                    float dot = nav_cube[x][y][z];
                    if (dot > 0) {
                        /*
                        Unit directly above current unit
                        */
                        nav_cube[x][y][z - 1] += dot*directUp;
                        /*
                        Units next to direct-above unit
                        */
                        nav_next_to_direct_above(x, y, z, dot, sideUp);
                        /*
                        Unit directly below current unit
                        */
                        nav_cube[x][y][z + 1] += dot*directDown;
                        /*
                        Unit next to direct-below unit
                        */
                        nav_next_to_direct_below(x, y, z, dot, sideDown);
                        /*
                        Unit to the side of current unit.
                        */
                        nav_side(x, y, z, dot, sideUnit);
                        /*
                        Remove transfer from original
                        */
                        nav_cube[x][y][z] -= dot*directDown + dot*sideDown * 8 + dot*sideUnit * 8 + dot*sideUp*8 + dot*directUp;
                    }
                }
            }
        }
        if (nav_cube[(X_DIM / 2) - 1][(Y_DIM / 2) - 1][Z_DIM - 1] && nav_cube[0][Y_DIM - 1][Z_DIM - 1]
            && nav_cube[X_DIM - 1][Y_DIM - 1][Z_DIM - 1] && nav_cube[X_DIM - 1][0][Z_DIM - 1] && nav_cube[0][0][Z_DIM - 1]) {
            notFullyMixed = false;
        }
```

Visually this simulation behaved like so:



Where the additional arrows pointing upward symbolize as the splash back created with the impact of the ink with every collection of water moving against it. This new set of directions with their respective transfer rate produced a simulation that was more realistic than the base simulation. It would seem that the closer we get to modeling reality, the more difficult it becomes for the simulation to handle all the data. As a result the sequential program could not complete the simulation fast enough to consider any result as something we could actually use.

4.2.3 Results

The results for the base simulation were manufactured since our data collection wasn't dependable by almost any means. Due to the lack of dependable data, the weights for each parameter were manually adjusted to minic what

we found with ink mixing with plain water at room temperature.

5 Conclusions

5.1 Analytical

5.2 Simulation

The two simulations give a good base to how we could of modeled the actual event, but with unreliable data, the simulations could only fit the behavior of ink to water and we had to extrapolate what would have happened beyond that. If we had another chance to do this experiment, having the right supplies for the mixing of cornstarch with water and a stable drop would give us a gold rush of useful data.

6 References

1. Socolofsky, Scott A. and Jirka, Gerhard H. Chapter 1. Environmental Fluid Mechanics Part 1: Mass Transfer and Diffusion. 2nd ed. Germany: Institute fr Hydromechanik: Universitt Karlsruhe, 2002. Institute for Hydromechanics: Karlsruhe Institute of Technology. Web. 28 November 2017.

2. Hansen, Caleb . How To Measure The Level Of Viscous Liquids. APG, 10 June 2015, www.apgsensors.com/about-us/blog/how-to-measure-the-level-of-viscous-liquids.

3. Cppreference. Date and time utilities. Date and time utilities - cppreference.com, Cppreference, en.cppreference.com/w/cpp/chrono.

4. Xu, Shibiao, et al. Interactive visual simulation of dynamic ink diffusion effects. Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry - VRCAI 11, 2011, doi:10.1145/2087756.2087770.

5. Daniels, Helmut, and Alexander Peters. PASTIS-3D A Parallel Finite Element Projection Code for the Time- Dependent Incompressible Navier-Stokes Equations. Numerical methods for the Navier-Stokes equations, 1994, pp. 3139., $doi : 10.1007/978 - 3 - 663 - 14007 - 8_4$.