PHYS 391 – Lab 1: Water Rocket Simulations

Key Concepts

- Use of Approximations in Physics/Rocketry
- Scientific Programming for Simulations
- Comparison of Observations with Simulation Predictions

1.1 General Lab Introduction

For this lab you and your group will be responsible for a developing a Water Rocket Simulation, comparing simulated to real experiment data, and refining your simulation as needed to match real data (or discuss differences)

All relevant data, calculations, observations, and responses to questions in the lab handouts should be recorded in a lab notebook. Prelab writeups are not required, although reading through the lab instructions before you begin will be quite useful. All material to be turned in should be included in the lab notebook, aside from any code requested which should be emailed directly to your instructor. Label your notebook entries with the corresponding section number from the handout. If you want to take your data first then go back and answer the questions later, you can always take notes on one side of your lab book leaving the other sheet blank, then go back and write all your answers on the other side.

While it is encouraged for students to work together collecting data, writing code, and understanding the concepts in these lab assignments, it is expected that each student does their own work analyzing the data and answering the questions. In particular, it is impossible to gain proficiency with writing MATLAB code by watching somebody else do it. It is a good idea to always have a prediction of what you expect for any of the tasks assigned in the labs. If you don't observe what you expect, you should think about whether this makes sense, or whether you made a mistake. You should not assume that you will always get the 'accepted' answer, however.

For this lab, a summary has been provided at the end of this write-up to help you identify all of the specific items you need to provide. You also need to check the lab 2 web page to find the code assignment for this lab. In addition to these items, you will also partly be graded on providing some useful discussion in your notebook related to what you are doing, observations, thoughts, analysis, and conclusions as you work through these steps.

Water Rocket Simulations Lab

1.2 Water Rocket Simulation Goals

This lab is a MATLAB-based exploration of scientific programming for simulations, writing code that, given initial conditions, uses physical principles to estimate how those conditions- air pressure in a water rocket, thrust generated at the rocket nozzle, height of the rocket, etc. in this case- evolve over time. You will be provided with data from previous class experiments, and asked to compare your runs of simulations with various initial conditions (matching those of the experiments) to the previous class experiments. A complete lab will include rationalization of simulated vs. (previously) observed data.

1.3 Water Rocket Simulation Overview

The process of writing a simulation is nominally three-fold:

- 1) Research and analysis of mathematical formula governing simulation
- 2) Coding to emulate governing math. relationships
- 3) Comparing and reconciling simulation output to actual data

1.4 1) Incorporating Physics into Simulations

Physics research, formulation of governing equations: There is the physics to be researched, entailing figuring out how thrust, impulse, momentum transfer and conservation, fluid drag, etc. all relate within a correct (but possibly complicated and non-linear) set of governing equations. Though your final goal may be to simply predict how high a rocket will fly, given a certain set of initial conditions, In reality, you need to keep track of all variables in the governing equations (for example, air pressure, P, in the rocket and, thus, thrust, T as a function of time. The idea here is NOT to analytically try to solve the governing equations but, rather, to simulate changes in these variable propagated over small steps in time.

Your initial conditions, passed to a MATLAB function as input variables, will include things like how much water is in the rocket, how much pressure has been applied, etc. An American Journal of Physics paper by *Finney (2000)* is a great starting point, as it presents one way of finding "thrust, drag and mass of a rocket as a function of time in order to find acceleration, velocity, and position (particularly, height).

1.4.1 perusing Finney

Looking at Finney (2000), how is the initial equation, (1), useful? (hint, what are its units). How will characterizing thrust, T, be helpful in simulating the flight of a water rocket?

Note that Finney moves from there to consideration of Bernoulli's equation in order to characterize pressure (P) in the rocket. This provides for relating the rocket internal pressure to water velocity just outside the nozzle, v_e, which is also in the equation for thrust, and thus an expression for thrust in terms of internal pressure (P), outside pressure (P_a), and the area of the nozzle (which he calls A_e). This suggests that the area of the nozzle could be another variable controlling the flight of a water rocket.

Note the simplifying assumptions regarding use of Bernoulli's equation, do you understand why they might be made?

As Finney notes (and as one might intuit), the thrust depends on finding the pressure inside the rocket as a function of time. At this point in analyzing the physics of the problem, this suggests that a simulation might be built around calculating (tracking) the pressure inside the rocket as time steps forward in small increments.

To move forward, Finney assumes the ideal gas law applies to the air inside the rocket (and what remains inside is water of course), and that we assume the air expands isothermally (what does that mean?). Thus equation 5 is arrived at, and one can take the derivative of pressure with respect to time (6) and thus 'step' through pressure inside the rocket during our simulation. Some substitutions lead to (7), the derivative of pressure again, but a complicated expression. In looking at (7), ask yourself what would make for larger, initial time-step changes in pressure.

1.4.2 incorporating drag

For purposes of this lab, we won't pursue an analytical solution such as (8). Instead, we will proceed to forming up a numerical solution that can be programmed in Matlab. However, we do need to consider section B, Drag, in order to work out the dynamics and kinematics of the rocket's flight.

Looking at equation (9) for drag (F_d), do you think you could come up with this equation with a little thinking? I would likely posit that the density of the air or liquid that the rocket is moving through (rho) would be in there, also, the cross-sectional area (A). I wouldn't know to use a 'coefficient of drag' (C_d) and, though I would expect more drag as the velocity of the rocket increases, I wouldn't know to relate it to *the square of* velocity. Realize, of course, that F_d is a better approximation to reality within certain velocity ranges, as noted in the text.

For now we won't worry a lot about D, other than coming up with a reasonable estimation of it later for moving forward with our simulation. Skipping ahead to III Numerical Solution, it's worth thinking about (16) and (17). That's where we transition from thinking about just physics to considering how to numerically solve these equations, which is discussed next.

1.5 2) Coding Your Simulation

1.5.1 discretizing the governing equations

The second part of the problem is coding a Matlab simulation that solve the governing equations. As stated before, ideally this simulation is a Matlab 'function' such that one can pass it various values for initial water volume, pressure, etc., and it will pass back, say, simulated position vs. time data for graphing and

comparison to real experiments. I typically develop a MATLAB script that calls this *water rocket function*, perhaps several times under varying initial conditions, makes plots of the results, etc.

Looking at Finney's equations (16) and (17), note first the format of each equation in (16) (and convince yourself you understand where the pressure equation comes from....). This suggests what is known as an 'iterative solution,' where we calculate the initial values for variables (P, v, y), divide the time of flight of the rocket into small, equal steps (in time), then step through a programmatic loop and calculate the next iterations values for P v and y until it makes sense to quit. Before we launch into coding, however, convince yourself that:

- (16b) makes sense to you. Where does this equation come from?
- (16c) makes sense to you. Where does this equation come from?
- To understand how (17) was constructed, ask yourself what is the underlying, fundamental physics equation that is used to derive (17)?

1.5.2 make a MATLAB function

Now let's think about how to construct a Matlab *function* that will do our bidding, i.e. calculate successive (or iterative) values for P, v, and y, for a given time step (*deltat*) and for initial values for rocket air pressure, volume of water, etc.

- ARITHMETIC AND ARRAYS: First, you should familiarize yourself with how arithmetic operations are performed and how *variables* (scalar, vector and matrix arrays) work in Matlab. To do so, consult Chapter 2 of Pratap (Lessons 1 and 2).
- FUNCTIONS: Lesson 5 of Pratap Chapter 2 gives background on making a Matlab function. The general idea is that you pass numbers (variables such as initial pressure, volume of water, mass of empty rocket, area of nozzle, etc. in this instance) to the function, and it passes back the results of the simulation, usually as arrays (of time, height, velocity, etc. in this case).
- SETTING CONSTANTS: It's typical to set some constants (e.g., a value for 'g' as -9.8 [m/s/s]) within the function. These are things that won't change from one call of the function to the next.
- SET INITIAL VALUES: You might need to do some simple arithmetic with passed variables to set up the initial value for drag (D), for example. Typically one also *initializes* the first values of simulated data arrays (time, height, etc.) within the function.
- FOR OR WHILE LOOPS: This is the critical part of the function (simulation), where you step through (iterate) time and calculate the next values of various working parameters (pressure, mass, acceleration, velocity position) from the previous (steps) values. The general form of a FOR loop is:

```
t(1) = 0;
dt = 0.1;
P(1) = P0 % this is the initial air pressure in rocket, passed to function

for i=2:1000
t(i) = t(i-1) + dt;
P(i) = Pn - ( Pn^2 / (P0*V0) ) * Ae * sqrt(2*(Pn-Pa)/rho_w)*dt;
% and yet more math calculating next values for mass, acceleration, velocity, etc.

end
```

In this case, the for will loop 999 times, with i starting at 2, then 3, etc. until it reaches 1000. After 1000, the loop completes.

A WHILE loop exits when a condition is met. For example:

```
t(1) = 0;
dt = 0.1;
P(1) = P0 % this is the initial air pressure in rocket, passed to function
Pd = 1;
```

```
i = 2;
while (Pd > 0)
t(i) = t(i-1) + dt;
P(i) = Pn - ( Pn^2 / (P0*V0) ) * Ae * sqrt(2*(Pn-Pa)/rho_w)*dt;
Pd = P(i) - Pa;
i = i + 1;

end
```

In this case, the while loop continues until the difference in pressure (Pd) between inside the rocket and outside (which is Pa, or atmospheric pressure) reaches zero. Once Pd reaches zero, the while loop completes. (I'm not saying this is the best variable to 'while loop' on....)

PASSING SIMULATED DATA BACK: Once your function has completed its calculation, you pass it back to the 'calling' routine (or the Matlab command line). This is all taken care of in the top line of the function, which could look like:

```
function [t,y,v,a,P,ymax] = my_spiffy_rocket_simulator(nzd, C, dt, P0, y0, v0, V0, mr, VT)

(do MATLAB things here including calculating outputs)

return
```

In this example, I've passed parameters (variables) and initial values within the '()'s on the RHS of the function call The Matlab expressions (which I calculate in the function) between the '[]'s are passed back to the calling routine.

It's good practice to insert comments using the % character in your code. Indeed, if you set up a *comment block* starting on the line immediately after your top line (e.g., function call) in your function, typing *help (function name)* (minus the parentheses) on the command line will echo the comment block. Thus you can embed your own reminders about what a function does, what inputs it wants, what outputs it calculates, right in your function.

1.6 comparison of simulations to real data

I will post some data (of maximum water rocket height for trials with various (stated) initial air pressures, water volumes, and a given nozzle diameter) on Blackboard under Course Documents. Please run your simulation for these particular initial conditions, compare your simulation predictions (for maximum height) vs. the observations. The data from a previous class will note particulars about individual rocket launches, such as (horizontal) wind speed, temperatures, etc. These might help explain discrepancies between simulated and observed data, but try to do a more systematic analysis (of many simulations vs. actual trials) to explain more fundamental discrepancies, for example, assumptions about drag. I expect to find a table comparing simulation vs. actual data and notes about particular trials within this analysis.

1.7 Final Thoughts

Even though obtaining real data of water rocket height (and perhaps internal air pressure and water volume, etc.) during a launch is beyond the scope of this course, coding a water rocket simulation is best approached iteratively, in small steps of time, and updating 'state variables' (above) as you move through the launch. Why? For one, the equation governing how pressure changes within the rocket, for example, is decidedly non-linear (it has terms of P^2 and $\sqrt{(}P)$ for example). So small steps help a linearized simulation maintain fidelity. For another reason, think about what happens as the launch proceeds. Does the rocket quit rising just as the water inside runs out (is completely expelled), or otherwise? In other words, do conditions change substantially during a launch? Also, what about drag, what is it related to and how would you solve analytically for it instead of iteratively?

Ultimately it IS possible to come up with a reasonable simulation, so keep this in mind. Simulations are extremely important to science and technology, from collisions of galaxies to the evolution of starts to climate change modeling on Earth to testing new car designs before building them to positing (numerically) how particles might be created and propagate from a collision of high energy protons in the Large Hadron Collider.

1.8 Required Elements

This list summarizes the minimal set of required elements you should provide for this lab. This is intended to help you identify all of the questions posed in the lab instructions. Please also check the web site to find the code assignment for this lab.

(fix this, below)

- Sec. 2.13.2 please detail how you incorporate drag into your simulation. State your assumptions regarding C_d or D, how they were modified to fit real experimental data, what ranges make sense, etc.
- Sec. 2.14.2: Please address the questions asked as bullet points in this section. Also, please give a full accounting of how you used the referenced equations (16, 17, etc.) to iteratively track those variables. Also, note any simulation (code) accommodations for condition changes during launch, as noted above.
- Sec. 2.14.2 Please send me your simulation function and be sure to include adequate commenting so that I can follow what you are doing. In commenting your code, put a 'header comment block' just below the first line so that *help my simulation* returns some sort of summary and operational clues about the function, explain what your simulation variables map to (e.g., v_n is my variable that will contain an array of velocities (vs. time) once the simulation is complete), etc.
- Sec. 2.15 I've given information about what I'd like to see for this section under this heading.
- Sec. 2.16 I'd appreciate your 'final thoughts' on the lab, including what you accomplished.

FOR THIS LAB: your challenge is to write a working Matlab function that will calculate water rocket height vs. time, given initial values and parameters. You will be given sets of data for actual water rocket launches (including their initial conditions) to compare and reconcile against your simulation results (in your lab notebook).