PHYS 391 – Lab 2: A) Measurement of the Speed of Light / B) Water Rocket Simulations

Key Concepts

- Statistical Inference: Mean, Standard Deviation
- Standard Deviation of the Mean
- Weighted Averages
- Scientific Programming for Simulations
- Comparison of Observations with Simulation Predictions

2.1 General Lab Introduction

For this lab you and your group will do either part A– Measurement of the Speed of Light, OR part B– Water Rocket Simulations (starts on page 7, but DO read the next 3 paragraphs)

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.

Measurment of Speed of Light Lab

2.2 Goals of this Lab

This lab is a MATLAB-based exploration of the key concepts of statistical inference and uncertainty in random variables. We will take multiple measurements of a physical quantity and then perform a weighted average to extract a best value for the speed of light. Some evaluation of possible systematic uncertainties will also be performed. This lab also will apply the concepts of MATLAB programming we first explored in Lab 1.

In this lab, when you are asked to calculate the mean and standard deviation of a sample of numbers, you may either use the built-in MATLAB functions or your own functions from the first lab. Make sure you know exactly what the built-in `std` function is doing, however, if you use it. For the weighted average, I am going to ask you to write your own function for this.

2.3 Speed of Light Overview

The speed of light is a fundamental property of the universe, and historically the discovery that light travelled at a very fast, but not infinite speed, was a very important result. Measurements of the speed of light are always difficult, and the first credited measurement was performed in 1676 by Ole Rømer who observed the eclipse times of Jupiter's moons and used these times to correctly measure the speed of light to an accuracy of 25%. More details on this rather amazing result can be found at the link on the Lab 2 webpage.

Experimentally, we often can not make pioneering measurements with high precision, and being able to extract useful data even from inaccurate tools is a sign of a good experimentalist. In this lab, we will attempt to make a measurement of the speed of light using nothing more than a computer (and the internet). Since long-distance communications now rely almost entirely on fiber optics, the time for signals to reach distant locations can be used to determine the speed of transmission in fibers, and from what we know about the index of refraction of an optical medium we can use that measurement to infer the speed of light in a vacuum (c).

Note, there are many sources of systematic uncertainty in this method. We will discuss some of them below, but even after evaluating some of these uncertainties, you shouldn't be too surprised if your result isn't in complete agreement with the accepted value. This discrepancy just serves to tell us that some of our assumptions aren't correct, and in the process we can learn something about the universe around us. The goal here is really not to make an accurate measurement of the speed of light, but rather to *try* to make an accurate measurement with the limited tools at our disposal.

2.4 Ping Times

2.4.1 general method

The general method, which is identical to most methods to measure c, is to send a signal to a distant site and time how long it takes for the signal to return. The speed of light in the medium v_n is then given by the distance-to-time ratio $v_n = d/t$ where d is the round-trip distance to the remote site, and t is the time elapsed. This can then be corrected for the known effects of the medium to infer the speed of light in a vacuum c.

To get an idea for the kinds of distances and times we need to measure, work out the expected speed of light in a communications fiber (which are almost always made of some kind of glass) and work out an expected round-trip time to a location 1,000 km away. This calculation should go in your log book. Get in the habit of trying to identify any sources of uncertainty in a calculation such as this, and (briefly) write down a few comments about things you may not know which could screw up your estimate.

If you were trying to make this measurement with a short and long (20 meter) cable connected between two computers in the lab (this was the original idea for this lab...) roughly estimate the timing resolution you would need to be able to measure the speed of light over these distances. In your lab book you should explain your thinking about the fractional uncertainty versus short and long distances.

2.4.2 using ping (or `hrping` on PCs)

To measure the transmission times, we will use a network utility called `ping` (you should use the improved Windows version of ping, `hrping`, on Windows computers in the lab) . This utility sends a small message to a remote computer, which then simply bounces the message back. The `ping` command measures the round-trip transit time and prints out the result once per second. See the Lab 2 website for more information on using `ping` if you are not familiar with this tool. `ping` is one of many *Unix* commands that are also accessible on mac and linux systems (the latter being an outgrowth of unix). Another extremely useful *Unix* command is `man` followed by the name of command, for example `man ping`. `man` will give you information about what a command does and how various *switches* (for example `-c`, below) affect the output of that command.

Test that `ping` is working by picking a machine at Oregon (the main web server `www.uoregon.edu` works well) and type the following.

```
ping www.uoregon.edu
or,
hrping www.uoregon.edu (on Windows machines)
```

This should give you one output line per second continuously until you type control-C to cancel the output. Note that `ping` finishes with a helpful status line which includes the mean time and the standard deviation. You can also use something like

```
ping -c 100 <hostname>
```

to run for a fixed number of cycles (in this case 100).

A machine located at UO should have a very short transit time, and should take less than 1 millisecond to respond. For short distances, the time is actually dominated by the latency (the time it takes the remote computer to respond or the time for intermediate switches to receive and retransmit the message). If you have a ping time significantly longer than about 1 ms, your route to the UO server is not very direct. **It will be very difficult to take good data for this lab if you are not running ping on a computer attached to the UO network!** From my house, Comcast sometimes sends signals to Seattle, then routes them back to Eugene to make the 3 block trip to campus. From the UO to any other US academic institution, traffic is routed on a special 'academic' network called Internet2. In addition to having high bandwidth and relatively low traffic, the connections tend to be more direct.

You can get an idea of the path your signal is taking with the command

```
traceroute <hostname>
```

which is another Unix network utility. There are likely fancy graphical applications for Windows which provide similar functionality. Many of the names returned by `traceroute` will be illegible, but there should be some obvious city names associated with any large hop as the signals travel through the main internet backbone routers.

You will need to use a `ping` command that gives results with a resolution of at least 100μs or better. Very old machines were limited to 1 ms precision, and this just won't work well, as all of your `ping` times will end up being identical and you will be unable to assess your statistical uncertainty.

2.4.3 ping targets and collecting data

Collect a series of data files where each file contains ping times to a single academic institution somewhere around the country or the world. Each file should have on the order of 100 independent measurements, most easily set by using the `-c 100` modifier to `ping` shown above. You should pick 5–10 servers including some that are close by (which can be used to measure the intrinsic latency) as well as servers that are far away. Please describe your criteria for picking your locations. Academic institutions are good, as their servers are almost always located on campus (which is not true for most companies) and almost all institutions have a web server with an address of www.something.edu which responds to `ping`. Institutes up and down I-5 are particularly good, as the fibers tend to generally follow the freeway route, and you can use the driving distance to get a pretty good estimate of the fiber distance. Long distances are good if you can figure out the route. Auckland is particularly nice, as there is an undersea cable which goes from Los Angeles through Hawaii directly to New Zealand. Some likely candidates are in Table 1 below. Feel free to use Google to find the address of your own favorite institution, but it is worth checking first that you can figure out reasonably well the route the `ping` traffic is taking.

Table 1: Addresses of a few academic web servers that respond to `ping`.

`www.uoregon.edu`
`www.oregonstate.edu`
`www.washington.edu`
`www.berkeley.edu`
`www.ucla.edu`
`www.uchicago.edu`
`www.mit.edu`
`www.cern.ch`
`www.auckland.ac.nz`

You should consider other academic web servers so as to get a reasonable distribution of distances and times, and tell me your rationale for this in your lab writeup. To get the `ping` output into a text file, either copy-paste from the command window, or else use the Unix 'redirection' facility to save the output directly to a file. One nice way to do this is to use the `tee` command, which will both print the `ping` output to the screen as well as save it to a file. This way you can check that the `ping` command is working properly as the data is being recorded. The syntax for this is shown below.

```
ping -c 100 www.somewhere.edu | tee myfile.txt
```

The vertical line is the Unix 'pipe' command which sends the output of `ping` to the input of `tee`. This pipeline syntax works for any Unix command.

Note that the very last line `ping` outputs will give the mean and standard deviation of the times. It is worth noting this down in your logbook so that when you recalculate these later you can check that you get something similar. You should also absolutely write down the *time* when each data file is recorded. The time of day can make a significant difference on latencies and in fact taking the data later at night or on the weekend could possibly give a significant difference. Testing a single site at various times of the day could actually be a very good way to evaluate systematics associated with varying latencies or other effects which are unrelated to the speed of light.

2.5 Ping Time Analysis

2.5.1 parsing the output

For each site you select, we want a best estimate of the time it takes a signal to travel that distance. You can either use the round-trip time and double the one-way distance, or use half of the round-trip time and estimate the one-way distance. Make sure it is clear in your log book which specific thing you are trying to measure.

The hardest part of this analysis is parsing the raw `ping` output into a list of numeric times. Ping output lines look something like the following, although this will vary slightly between different operating systems:

```
64 bytes from 169.229.216.200: icmp_seq=0 ttl=50 time=34.788 ms
64 bytes from 169.229.216.200: icmp_seq=1 ttl=50 time=34.394 ms
64 bytes from 169.229.216.200: icmp_seq=2 ttl=50 time=36.421 ms
64 bytes from 169.229.216.200: icmp_seq=3 ttl=50 time=34.334 ms
64 bytes from 169.229.216.200: icmp_seq=4 ttl=50 time=34.340 ms
```

This is always a tradeoff between spending a lot of time designing a very clever, flexible solution and just finding a quick solution that barely works. One could just copy these numbers by hand into a text file using a text editor, but this is probably completely unacceptable for hundreds of lines across many files. A global search-and-replace to just eliminate all of the unchanging text in the file is one viable option however.

There are sophisticated Unix command-line tools like `sed` which can parse complex patterns out of text files, but it would probably take hours to figure out the syntax. Unless you already have experience with these tools, this 'fancy' solution is also probably unacceptable. One pragmatic possibility is to use something like MATLAB's csvread. Because the numeric times are likely separated by both spaces and an equal sign, you may need to first delete all the equal signs from your files, or else do the parsing in two steps.

There are simpler Unix tools that can do enough simple text parsing to get the job done with only a little effort, for instance the `cut` command. You can get information on `cut` (or any Unix command) by typing `man cut` (`man` stands for manual). With the following syntax, `cut` uses an equals sign as a delimiter and selects the fourth delimited column from the file `myfile.txt`:

```
cut -f 4 -d "=" myfile.txt
```

This reduces my original text down to

```
34.788 ms
34.394 ms
36.421 ms
34.334 ms
34.340 ms
```

Now I could follow this with `cut -f 1 -d " "` to pick everything before the first space, which is the number I want. I can then use `tee` to redirect this into a new file. I can run all of these at once using the Unix pipe construction, and the entire command looks like this:

```
cut -f 4 -d "=" myfile.txt | cut -f 1 -d " " | tee mydata.txt
```

Note that this doesn't work on a Windows machine, but it will work on a Mac through the Terminal, as well as on any Unix/Linux machine. There may be a few extra lines at the start and end of each file that you need to clean up by hand, but it shouldn't take you more than about 10 minutes to get all of your data ready to read into MATLAB.

Another possible parsing option is to import the original file into a spreadsheet like Excel. These programs typically have very sophisticated adaptive parsing algorithms which may be able to find all the numbers for you automatically. I haven't tried this myself, but if you don't want to use command-line tools, this is another possibility.

2.5.2 time data analysis

For each location, you should read the raw data in, compute the mean and standard deviation of this data sample, and use those values to get a best estimate for the mean time (expressed as $\bar{t} \pm \delta\bar{t}$). You should also produce a histogram of t and check whether the data plausibly looks like it has come from a single parent distribution, or whether there are 'outliers' that clearly don't appear to belong. Make sure to use enough bins in your histogram so you can really see something (one bin per 100μs is probably a good goal). Can you think of reasons why certain ping times might be anomalously long or short? If necessary, describe some criteria for 'cleaning' your data, such as Chauvenet's criterion described in Taylor Chapter 6, and apply it to your times before calculating your final best time. You should only need to do this in extreme cases, and I am not encouraging you to use Chauvenet's criterion, but if you have a few points which are clearly way off, this is arguably a justification for removing them.

If you have some other funny looking distribution, you probably need to think about how to handle this. To select elements out of an array based on some condition, the easiest method is to use MATLAB's `find` command to index the array elements. So to get all elements of `x` which are greater than zero, use: `x(find(x>0))`. This will return a new array with the conditional expression `x>0` applied to each element of `x` in turn.

Once you have final numbers, make a table in your logbook for each location and give the final average time including an uncertainty. In this case, the uncertainty is only a statistical uncertainty based on the random variations in times observed. We will think about possible systematic uncertainties later. Leave a column in your table for the distance, which we will determine next.

2.6 Distance Analysis

This is probably the most inaccurate part of this entire measurement. For each site, you need to determine an estimate of the distance and an uncertainty on that estimate. Use your own judgement and *describe your reasoning* in your log book. There is no right answer here, although there are plenty of wrong answers. Use `traceroute` or some similar program to try to estimate the actual path, and make some estimate of the distance and possible uncertainty, especially in terms of how much extra cabling there might be within a city. Los Angeles, for instance, is a big place, so knowing that a router is in LA still carries a significant uncertainty. A useful *Unix* command for doing this is `nslookup nnn.nn.nnn.n`, where all the nns represent an IP address (for example, UO's IP address is 128.223.142.71). `nslookup` oftentimes give information that is useful for locating particular relay points. Use Google or any other mapping utilities which you can find and try to do as good of a job as you can here, although in the end you will still likely have significant uncertainties on each distance. These uncertainties don't all have to be the same however, and after justifying your estimates in your log book, complete your table with distances to go with your ping times.

2.7 Speed of Light

We now have all the raw data we need, and we need to determine a method to combine these to give a best value of the speed of light in vacuum. I always prefer to work with values close to the actual measured values and only apply significant corrections at the end. Philosophically, I find it more honest to first quote the quantity that was actually measured, and then apply any additional assumptions or interpretations at the end of the analysis. In that spirit, you should work out a way to measure the speed of light in fiber v_n, and only once you have a final result for that value should you worry about converting this into c.

2.7.1 make a plot?

The most obvious method to determine v_n is to make a plot of distance vs. time and fit a line to that data. The slope of the line (assuming there is a linear relationship) will be the speed. This method has the added advantage of naturally taking out any offset of the line from the origin and giving a direct measurement of the slope. Why might there be a non-zero ping time, even at a distance very close to zero? Evaluate whether your distance measurements or time measurements are more uncertain and provide error bars for the most uncertain variable using something like the `errorbar` function. Note that `errorbar` will only put errors on the vertical axis, so arrange your variables accordingly. Make this plot and tape it into your log book. If you can't get this to work, at least plot the points and draw the errors on by hand. *Don't* connect the individual data points with lines in MATLAB, although you can draw an estimate of a best-fit line onto the points with a pencil. Actually, you're encouraged to use Matlab to estimate a best-fit line through the data points, draw it on the plot (of disconnected points with error bars), and discuss what it means in the text.

Write down any observations you might have, especially if there are obvious trends in your data, such as anomalous points, or a different trend for the overseas sites from the US-based ones. If you really have holes in your coverage, you might think about picking one or two more sites at particular distances to help fill in your graph. Do all of the data support a linear relationship between distance and time? If not, can you think of any reasons why?

2.7.2 or not?

Each site should give you an independent measurement of v_n. A method for combining the data which doesn't involve fitting a line is to evaluate these separate measurements then combine these taking into account their uncertainty as a weighted average. As we will see later in the quarter, aside from the issue of any possible offset at zero, these two methods are equivalent.

First, decide whether there is any substantial latency in your time measurements, and either come up with a strategy to correct these (don't forget error propagation) or else just treat this as an additional source of systematic uncertainty and possibly add a systematic uncertainty in quadrature with your statistical uncertainty on t if it appears to be significant.

If you are really doing a careful job, you might try to estimate the 'latency-per-hop' and correct different sites according to the number of routers encountered along the path, although I am a bit skeptical that this would make too much of a difference for long routes.

So explain what you have chosen to do in analyzing your data, and calculate a single value of v_n and a total uncertainty for each site you pinged. Don't forget your error propagation since you will have an uncertainty on both the transit time and the distance. You can either do this calculation in MATLAB, by hand, with a spreadsheet, or any other method you choose, although I would encourage you to start with your measured distance-time data (including uncertainties) in a single text file, one line per site, and read this file into MATLAB and do the entire analysis there. The end result should be a table with individual v_n values and uncertainties for each site.

2.7.3 make a second plot

Make a second plot, similar to the one made in Section 2.7.1, but showing v_n vs. distance. Here the error bars should reflect the total uncertainty. It is probably worth commenting on whether these v_n values appear to be consistent with each other. This is the same information as went into the plot in Section 2.7.1 so these two plots should be consistent with each other, but trends may be more apparent here. If your uncertainties have been evaluated correctly, these all should be more or less consistent (within Gaussian probabilities). If there is more scatter in your measured values than the uncertainties would predict, that is a sure sign that there are other uncontrolled systematics lurking in the measurement (or you have made a mistake). You don't have to be quantitative here, but make a rough assessment of the agreement to demonstrate (or refute) the accuracy of your errors. If a single site is way off, consider whether you should exclude that point from the measurement, and provide some plausible explanation for why this may not be consistent to justify your decision.

2.7.4 make a weighted average

Once you have your v_n values, combine them using the rules for performing a weighted average from Taylor Chapter 7. First, write down the actual formulas that you need to use, then write a MATLAB function

that will take two input arrays (one with values and the second with uncertainties), performs the weighted average calculation, and also returns the uncertainty on the weighted average. Finally, put it all together into a script which performs the weighted average calculation from an input data file (either time-distance data, or v_n data) and prints out the final result (plus uncertainty).

2.8 Interpretation

Finally, apply any additional known corrections (including possible uncertainties) to evaluate the speed of light in vacuum. How well does your answer agree with the 'expected' value? You should certainly be within an order of magnitude of the correct answer, although you may not be consistent within your quoted uncertainties.

Remember, what you have actually measured is the 'speed of light on the internet' and you have made assumptions about how you can relate that to the speed of light in a vacuum. Discuss whether it is possible that you have underestimated some of the uncertainties you have identified, and in particular think about whether some of these uncertainties might push the final measured value in one direction or the other preferentially. Also, try to think of other effects that have not been previously considered which might effect this measurement. In a real experiment, where the 'correct' answer is not known, this reflection on the result is important to identify additional systematics that ideally can be constrained with further measurements. Write a paragraph or so discussing these issues, as well as any additional things that you might chose to do differently if you did this over again.

2.9 Final Thoughts

The relative precision of a random process improves as $1/\sqrt{N}$ since the standard deviation of the mean scales with this factor. In this measurement, you can easily take enough data to make the statistical uncertainties negligibly small, which then leaves you with systematic effects which need to be evaluated. Systematic effects which involve some level of variation can additionally be constrained by making multiple measurements, either of the same thing over and over again, or in different systems which should give the same answer. You have done this here by making multiple measurements at different distances, and you could have also repeated the measurements at different times of the day to further assess systematic variations (perhaps you did)!

The most difficult uncertainties to assess are systematic uncertainties which lead to inaccurate results. Understanding whether there is a fixed scale error throwing your measurement off can be maddeningly difficult to assess, and this is why the scientific method depends on others being able to reproduce your results, particularly with different techniques. If two measurements give inconsistent results, there is likely something wrong either in the measurements themselves or the interpretations of those measurements. This is why almost no discovery is really believed until it has been confirmed independently.

2.10 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.

- Sec. 2.4.1: Expected speed of light in fiber, expected round-trip times, necessary time resolution to make this measurement using a 20 meter cable
- Sec. 2.4.3: Description of data acquisition
- Sec. 2.5.1: Explanation of what you are measuring, description of data parsing
- Sec. 2.5.2: Histograms of t for at least a few sites, discussion of outliers and data cleaning (if needed), table with at least times and (statistical) uncertainties
- Sec. 2.6: Describe your distance analysis and make some estimate of the uncertainty
- Sec. 2.7.1: Plot of distance vs. time
- Sec. 2.7.2: Table of v_n values with uncertainties

- Sec. 2.7.3: Plot of v_n vs. distance, discussion of consistency
- Sec. 2.7.4: Weighted average combination of v_n
- Sec. 2.8: Discussion of results

Water Rocket Simulations Lab

2.11 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.

2.12 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

2.13 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).

2.13.1 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.

2.13.2 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.

2.14 2) Coding Your Simulation

2.14.1 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)?

2.14.2 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.

2.15 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.

2.16 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.

2.17 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).