SIGNATURE PAGE

This thesis for honors recognition has been approved for the

Department of Mathematics, Engineering, and Computer Science.

5-2-16

Director Date

4/29/16

Reader Date

Anthony M. Spilka 4/29/16

Reader Date

Analyzing the Effects of Topography on High-Altitude Balloon Descent

Elizabeth Carlson

Advisor: Eric Sullivan, Ph.D.

May 1st, 2016

Abstract

This project was developed to improve upon an experimentally-based descent prediction program for high-altitude balloon payloads at the Montana Space Grant Consortium's Balloon Outreach, Research, Exploration And Landscape Imaging System (BOREALIS) program. The high-altitude balloon payloads were launched from flat tarmacs but generally landed in or near mountainous terrain. Terrain has a significant effect on air flow, and thus this project was developed in order to improve the accuracy of the predicted landing latitude and longitude. The project was broken into two parts. The first part was simplifying and solving the Navier-Stokes equations to predict an external wind field incorporating the effects of topography. Since the payload would be entering a steady velocity field, we deduced, with the support of external research, that we could simplify the Reynolds Averaged Navier-Stokes equations. Our final simplified equations were solved on a 100×100 grid. The second portion was tracking the position of the payload through the grid using interpolation techniques and Newtonian physics.

Contents

List of Figures

1 Introduction

This project was developed to improve upon an experimentally-based descent prediction program for high-altitude balloon payloads at the Montana Space Grant Consortium's Balloon Outreach, Research, Exploration And Landscape Imaging System (BOREALIS) program. Since the prediction program was experimentally based, we estimated the descent velocity profile with the data from the ascent velocity profile. We launched the high-altitude balloon payloads from flat tarmacs; however, we often landed in or near mountainous terrain. Terrain has a significant effect on air flow, and thus this project was developed in order to improve the accuracy of the predicted landing latitude and longitude.

In the first section, we explain how to simplify the Navier-Stokes equations into the Reynolds Averaged Navier-Stokes equations. In the next section, we explain the assumptions we make in order to further simplify these equations for our purposes and resolve them into x and z components (horizontal and vertical, respectively). In the next section, we discuss our numerical solution of these differential equations on a fine grid. This returns a wind profile along a flat terrain. We then design and incorporate a terrain model based loosely on the shape of the Crazy Mountains. Now that we have a wind field, we want to predict how this force affects the trajectory of the high-altitude balloon payload. First, we have to incorporate all the forces which affect the trajectory of the payload, including gravity, drag, and the wind profile at a given point. The next section explains how to find the wind profile at any given point in the grid as well as solving the first order differential equations using Euler's Method. Finally, we explore the strengths and weaknesses of this model and future work.

2 Physics Backgroud

2.1 Introduction to Navier-Stokes Equation

Air flow across terrain is naturally described as a change in the direction and strength of the flow, and thus should be modeled using differential equations [14]. Based on Euler's original fluid flow equations for ideal (or Newtonian) fluids, Claude Navier and George Stokes both made significant improvements to the fluid flow differential equation [8], yielding the current form [1],

$$\rho\frac{\partial \boldsymbol{v}}{\partial t} + \rho\boldsymbol{v}\cdot\boldsymbol{\nabla}\boldsymbol{v} + \boldsymbol{\nabla}p - (\lambda+\mu)\boldsymbol{\nabla}(\boldsymbol{\nabla}\cdot\boldsymbol{v}) - \mu\boldsymbol{\nabla}\cdot\boldsymbol{\nabla}\boldsymbol{v} - \rho\boldsymbol{g} = 0. \tag{1}$$

Note that ρ represents the density of the fluid assumed constant, p represents pressure, μ the dynamic viscosity of the fluid (assumed constant), λ represents the bulk viscosity of the fluid (assumed constant), $\boldsymbol{v}$ represents velocity, $\boldsymbol{g}$ represents external accelerations present on the object, t represents time, and $\boldsymbol{\nabla}$ represents the gradient operator. This equation is an attempt to expressly describe fluid flow in terms of conservation of momentum. If we look at the first two terms from equation (1),

$$\rho\frac{\partial \boldsymbol{v}}{\partial t} + \rho\boldsymbol{v}\cdot\boldsymbol{\nabla}\boldsymbol{v},$$

this represents the material time derivative of velocity multiplied by density, $\rho\frac{D\boldsymbol{v}}{Dt}$. This material time derivative defines our concept of advection. In fact, the terms

$$\rho\frac{\partial \boldsymbol{v}}{\partial t} + \rho\boldsymbol{v}\cdot\boldsymbol{\nabla}\boldsymbol{v} - \rho\boldsymbol{g} + \boldsymbol{\nabla}p,$$

when set equal to 0, are the terms in Euler's original fluid dynamics equation. [11] In this case, $\boldsymbol{\nabla}p$ is the work done by pressure, and $\boldsymbol{g}$ respresents the accelerations affecting the fluid. This equation, however, disregards the forces caused by stress.

Stress is defined in engineering and mechanics as the internal forces within the fluid. Navier-Stokes accounts for stress using the term

$$-(\lambda+\mu)\boldsymbol{\nabla}(\boldsymbol{\nabla}\cdot\boldsymbol{v})-\mu\boldsymbol{\nabla}\cdot\boldsymbol{\nabla}\boldsymbol{v},$$

which was developed assuming nonlinearity in the stress tensor [5].

2.2 Introduction to the Reynolds Averaged Navier-Stokes Equation

For this particular project, we wanted to develop a steady-state wind field and could therefore use a time-averaged version of the Navier-Stokes equation. The time-averaged version of the Navier-Stokes equation is known as the Reynolds Averaged Navier-Stokes equation, abbreviated RANS.

Equation (1) can be simplified using Einstein notation and the simplifying assumption of air incompressibility. We assumed incompressibility because we have assumed density is constant and it is in a large space, the atmosphere, where the air is not necessarily being compressed. Incompressibility in the Navier-Stokes equations is represented by the term $\boldsymbol{\nabla}\cdot\boldsymbol{v}=0$ [1]. This is derived by setting the material time derivative of density $\frac{D\rho}{Dt}=0$ and substituting this into the conservation of mass equation, $\frac{D\rho}{Dt}+\rho\boldsymbol{\nabla}\cdot\boldsymbol{v}=0$. This leads to the conclusion, in Einstein's summation notation, that $\frac{\partial u_i}{\partial x_i}=0$. This simplifies equation (1), in Einstein notation, to

$$\rho\left[\frac{\partial u_i}{\partial t}+\frac{\partial(u_j u_i)}{\partial x_j}\right]=\frac{\partial}{\partial x_j}\left(\mu\frac{\partial u_i}{\partial x_j}\right)-\frac{\partial p}{\partial x_i}+\rho g_i, \tag{2}$$

where i,j,k are indices representing x, y, and z, where repeated use of the index indicates summation.

The velocity function u is a function of t,x,y, and z, and in order to time average

the variables, Osborne Reynolds developed what is known as the Reynolds decomposition: $u(x,y,z,t) = \overline{u}(x,y,z) + u'(x,y,z,t)$ and $p(x,y,z,t) = \overline{p}(x,y,z) + p'(x,y,z,t)$. The time-averaged component is the $\overline{u}(x,y,z)$ and $\overline{p}(x,y,z)$ while the "fluctuating component" is $u'(x,y,z,t)$ and $p'(x,y,z,t)$ [10]. Notice that this decomposes the functions into the sum of a time-independent component and a time-dependent component. Subsituting Reynolds decompositions into equation (2) gives

$$\rho\left[\frac{\partial(\overline{u}_i + u_i')}{\partial t} + \frac{\partial((\overline{u}_j + u_j')(\overline{u}_i + u_i'))}{\partial x_j}\right] = \frac{\partial}{\partial x_j}\left(\mu\frac{\partial(\overline{u}_i + u_i')}{\partial x_j}\right) - \frac{\partial(\overline{p}_i + p_i')}{\partial x_i} + \rho g_i. \tag{3}$$

This can be resolved into a simpler form, using the linearity of the derivative operator, to get

$$\begin{aligned}\rho\left[\frac{\partial\overline{u}_i}{\partial t} + \frac{\partial u_i'}{\partial t} + \frac{\partial(\overline{u}_j\overline{u}_i)}{\partial x_j} + \frac{\partial(\overline{u}_j u_i')}{\partial x_j} + \frac{\partial(\overline{u}_i u_j')}{\partial x_j} + \frac{\partial(u_j'u_i')}{\partial x_j}\right] \\ = \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial\overline{u}_i}{\partial x_j} + \frac{\partial u_i'}{\partial x_j}\right)\right) - \frac{\partial\overline{p}_i}{\partial x_i} - \frac{\partial p_i'}{\partial x_i} + \rho g_i.\end{aligned} \tag{4}$$

Equation (4) can be further simplified using the product rule to

$$\begin{aligned}\rho\left[\frac{\partial\overline{u}_i}{\partial t} + \frac{\partial u_i'}{\partial t} + \overline{u}_j\frac{\partial(\overline{u}_i)}{\partial x_j} + \overline{u}_i\frac{\partial(\overline{u}_j)}{\partial x_j} + \overline{u}_j\frac{\partial(u_i')}{\partial x_j} + u_i'\frac{\partial(\overline{u_j})}{\partial x_j} + \overline{u}_i\frac{\partial(u_j')}{\partial x_j} + u_j'\frac{\partial(\overline{u_i})}{\partial x_j} + \right. \\ \left.\frac{\partial(u_i'u_j')}{\partial x_j}\right] = \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial\overline{u}_i}{\partial x_j} + \frac{\partial u_i'}{\partial x_j}\right)\right) - \frac{\partial\overline{p}_i}{\partial x_i} - \frac{\partial p_i'}{\partial x_i} + \rho g_i,\end{aligned} \tag{5}$$

and then both sides of equation (5) are time-averaged,

$$\begin{aligned}\overline{\rho\left[\frac{\partial\overline{u}_i}{\partial t} + \frac{\partial u_i'}{\partial t} + \overline{u}_j\frac{\partial(\overline{u}_i)}{\partial x_j} + \overline{u}_i\frac{\partial(\overline{u}_j)}{\partial x_j} + \overline{u}_j\frac{\partial(u_i')}{\partial x_j} + u_i'\frac{\partial(\overline{u_j})}{\partial x_j} + \overline{u}_i\frac{\partial(u_j')}{\partial x_j} + u_j'\frac{\partial(\overline{u_i})}{\partial x_j} + \right.} \\ \overline{\left.\frac{\partial(u_i'u_j')}{\partial x_j}\right]} = \overline{\frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial\overline{u}_i}{\partial x_j} + \frac{\partial u_i'}{\partial x_j}\right)\right) - \frac{\partial\overline{p}_i}{\partial x_i} - \frac{\partial p_i'}{\partial x_i} + \rho g_i},\end{aligned} \tag{6}$$

To simplify farther, the following rules for time averaging [5, 13] can be implemented,

$$\overline{\overline{u_i}} = \overline{u_i} \tag{7}$$

$$\overline{u_i + u_j} = \overline{u_i} + \overline{u_j} \tag{8}$$

$$\overline{\overline{u_i} u_j} = \overline{u_i u_j} \tag{9}$$

$$\overline{u_i} \cdot \overline{u_i'} = 0 \tag{10}$$

$$\overline{\overline{u_i} + u_i'} = \overline{u_i} \tag{11}$$

$$\overline{\frac{\partial u_i}{\partial x_j}} = \frac{\partial \overline{u_i}}{\partial x_j} \tag{12}$$

Applying rule (8), equation (6) becomes

$$\rho \left[\overline{\frac{\partial \overline{u_i}}{\partial t}} + \overline{\frac{\partial u_i'}{\partial t}} + \overline{\overline{u}_j \frac{\partial (\overline{u}_i)}{\partial x_j}} + \overline{\overline{u}_i \frac{\partial (\overline{u}_j)}{\partial x_j}} + \overline{\overline{u}_j \frac{\partial (u_i')}{\partial x_j}} + \overline{u_i' \frac{\partial (\overline{u_j})}{\partial x_j}} + \overline{\overline{u}_i \frac{\partial (u_j')}{\partial x_j}} + \overline{u_j' \frac{\partial (\overline{u_i})}{\partial x_j}} + \overline{\frac{\partial (u_i' u_j')}{\partial x_j}} \right] = \overline{\frac{\partial}{\partial x_j}} \left(\mu \overline{\frac{\partial \overline{u_i}}{\partial x_j}} + \overline{\frac{\partial u_i'}{\partial x_j}} \right) - \overline{\frac{\partial \overline{p}_i}{\partial x_i}} - \overline{\frac{\partial p_i'}{\partial x_i}} + \rho \overline{g_i}. \tag{13}$$

Applying rules (10), (11), and (12), equation (13) simplifies to

$$\rho \left[\overline{u}_j \frac{\partial (\overline{u}_i)}{\partial x_j} + \overline{u}_i \frac{\partial (\overline{u}_j)}{\partial x_j} + \frac{\partial \overline{(u_i' u_j')}}{\partial x_j} \right] = \mu \overline{\frac{\partial \overline{u_i}}{\partial x_j \partial x_j}} - \frac{\partial \overline{p}_i}{\partial x_i} + \rho \overline{g_i}. \tag{14}$$

Also note that $\frac{\partial (\overline{u}_j)}{\partial x_j} = 0$ since air flow is incompressible [1], and thus equation (14) becomes

$$\rho \left[\overline{u}_j \frac{\partial (\overline{u}_i)}{\partial x_j} + \frac{\partial \overline{(u_i' u_j')}}{\partial x_j} \right] = \mu \overline{\frac{\partial \overline{u_i}}{\partial x_j \partial x_j}} - \frac{\partial \overline{p}_i}{\partial x_i} + \rho \overline{g_i}. \tag{15}$$

With some algebraic manipulation and adding an additional shear stress term [10] due to turbulent mixing, equation (15) yields the standard form for the Reynolds-Averaged Navier-Stokes equations,

$$\rho\overline{u_j}\frac{\partial\overline{u_i}}{\partial x_j} = \rho\overline{g_i} + \frac{\partial}{\partial x_j}\left[-\overline{p}\delta_{ij} + \mu\left(\frac{\partial\overline{u_i}}{\partial x_j} + \frac{\partial\overline{u_j}}{\partial x_i}\right) - \rho\overline{u_i'u_j'}\right], \tag{16}$$

where u represents velocity, μ represents air's viscosity, ρ represents air's density, and g represents external forces. [10, 13]

3 Fluid Flow: Simplifying the RANS Equations

3.1 Assumptions

Since we are only analyzing the trajectory of the balloon through a small portion of space, we made a simplifying assumption that pressure remains constant with respect to vertical and horizontal directions. We also know that there are no external forces affecting the air flow other than gravity. However, gravity can be ignored because we have assumed that ρ is constant and therefore the system is in hydrostatic equilibrium. This implies that $\overline{g_i} = 0$. This simplifies equation (16) to

$$\rho\overline{u_j}\frac{\partial\overline{u_i}}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial\overline{u_i}}{\partial x_j} + \frac{\partial\overline{u_j}}{\partial x_i}\right) - \rho\overline{u_i'u_j'}\right]. \tag{17}$$

Finally, we set $\rho\overline{u_i'u_j'} = 0$ (the Reynolds stress term) because we have a predominantly horizontal wind vector field with minimal updraft, since we are close to the Earth's surface. Using these simplifying assumptions, we ended up with the following set of equations in two dimensions, where u represents the velocity in the x-direction and w represents the velocity in the z-direction.

$$\rho u\frac{\partial u}{\partial x} + \rho w\frac{\partial u}{\partial z} = \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial z^2}\right) \tag{18}$$

$$\rho u\frac{\partial w}{\partial x} + \rho w\frac{\partial w}{\partial z} = \mu\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial z^2}\right) \tag{19}$$

Next, we need to solve these equations numerically. An added benefit of solving the time-independent version of the Navier-Stokes equations numerically is that we will be able to avoid the time instabilities that occur with solving the full Navier-Stokes numerically.

3.2 Numerical Solutions to Simplified RANS Equations

After simplifying the Reynolds Averaged Navier-Stokes fluid flow equations to suit our particular physical situation, the next, and possibly most important, part of the project is to numerically solve these differential equations. In order to do this, we need to set up a meshgrid on which to solve them, select reasonable boundary conditions, and then select and implement a numerical solution method. We chose to implement these solution ideas in the mathematical software MATLAB® since it is the software with which we had the most experience.

3.2.1 Setting up the Meshgrid

First, since numerical methods give an approximate solution at a particular point in space, we set up a finite grid on which to solve the equations. This grid is on a 2 km $\times$ 2 km axis, such that $x = \{x \in \mathbb{R} \mid 0 \leq x \leq 2\}$ and $z = \{z \in \mathbb{R} \mid 0 \leq z \leq 2\}$. Since wind velocities are often reported using measurements of the velocity 20 feet above the ground or at the top of vegetation [6], this domain and range of 2 km $\times$ 2 km should be sufficient for topographical effects to affect the direction of wind velocity. We set up a simple meshgrid of 20 $\times$ 20 grid points for initial calculations, but this grid was later increased to 100 $\times$ 100 equally spaced grid points. Since this grid is square, $dx = dz = 0.0202$ m.

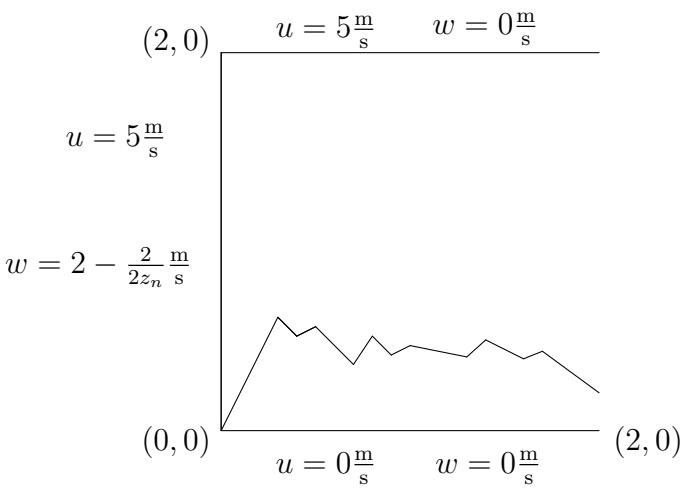

Figure 1: Rough Sketch of Boundary Conditions with Final Terrain

3.2.2 Boundary Conditions

The purpose of this project was to more realistically model air flow over terrain. Thus, using the data we generated from our original descent prediction program, we would be given boundary conditions. For the purposes of testing our code, we assumed some reasonable boundary conditions. Horizontally, we set the left and top edges to have a 5 $\frac{\text{m}}{\text{s}}$ wind, which is a slight breeze, equivalent in English units to 11.185 mph (for initial tests without terrain, we also set the bottom edge to be 5 $\frac{\text{m}}{\text{s}}$).

We also wanted to set a small initial updraft on the left edge. Not wanting to push air out of the system through the top, we set the updraft to be linearly decreasing in magnitude from 2 $\frac{\text{m}}{\text{s}}$ to 0 $\frac{\text{m}}{\text{s}}$ from the bottom to the top of the grid. Figure 1 is a sketch of these boundary conditions.

3.2.3 Discretizing Our Fluid-Flow Equations

In order to solve equations (18) and (19), we discretize them for numerical purposes. For this solution technique we worked with a central difference approximation of second order error [9], leading to

$$\rho u_{m,n}\frac{u_{m+1,n}-u_{m-1,n}}{2\Delta x}+\rho w_{m,n}\frac{u_{m,n+1}-u_{m,n-1}}{2\Delta z}= \mu\left(\frac{u_{m+1,n}-2u_{m,n}+u_{m-1,n}}{\Delta x^2}+\frac{u_{m,n+1}-2u_{m,n}+u_{m,n-1}}{\Delta z^2}\right) \tag{20}$$

$$\rho u_{m,n}\frac{w_{m+1,n}-w_{m-1,n}}{2\Delta x}+\rho w_{m,n}\frac{w_{m,n+1}-w_{m,n-1}}{2\Delta z}= \mu\left(\frac{w_{m+1,n}-2w_{m,n}+w_{m-1,n}}{\Delta x^2}+\frac{w_{m,n+1}-2w_{m,n}+w_{m,n-1}}{\Delta z^2}\right) \tag{21}$$

In these equations, m represents the grid row and n represents the grid column. Using these discretizations, our nonlinear solver will be solving for $u_{m,n}$ and $w_{m,n}$.

3.2.4 Newton's Nonlinear Solver

Equations (20) and (21) are both nonlinear and thus require the use of a nonlinear solver. For this project, we chose Newton's 2D nonlinear solver [15]. This method solves for the roots of the nonlinear discretized differential equations with respect to $u_{m,n}$ and $w_{m,n}$.

We begin by subtracting each term of equations (20) and (21) to the left hand side,

$$\begin{aligned}0 =&\mu\left(\frac{u_{m+1,n}-2u_{m,n}+u_{m-1,n}}{\Delta x^2}+\frac{u_{m,n+1}-2u_{m,n}+u_{m,n-1}}{\Delta z^2}\right)\\ &-\rho u_{m,n}\frac{u_{m+1,n}-u_{m-1,n}}{2\Delta x}-\rho w_{m,n}\frac{u_{m,n+1}-u_{m,n-1}}{2\Delta z},\end{aligned} \tag{22}$$

$$0 = \mu\left(\frac{w_{m+1,n} - 2w_{m,n} + w_{m-1,n}}{\Delta x^2} + \frac{w_{m,n+1} - 2w_{m,n} + w_{m,n-1}}{\Delta z^2}\right) - \rho u_{m,n}\frac{w_{m+1,n} - w_{m-1,n}}{2\Delta x} - \rho w_{m,n}\frac{w_{m,n+1} - w_{m,n-1}}{2\Delta z}. \tag{23}$$

We label the left-hand side of equations (22) and (23) as F_1 and F_2, respectively,

$$F_1 = \mu\left(\frac{u_{m+1,n} - 2u_{m,n} + u_{m-1,n}}{\Delta x^2} + \frac{u_{m,n+1} - 2u_{m,n} + u_{m,n-1}}{\Delta z^2}\right) - \rho u_{m,n}\frac{u_{m+1,n} - u_{m-1,n}}{2\Delta x} - \rho w_{m,n}\frac{u_{m,n+1} - u_{m,n-1}}{2\Delta z}, \tag{24}$$

$$F_2 = \mu\left(\frac{w_{m+1,n} - 2w_{m,n} + w_{m-1,n}}{\Delta x^2} + \frac{w_{m,n+1} - 2w_{m,n} + w_{m,n-1}}{\Delta z^2}\right) - \rho u_{m,n}\frac{w_{m+1,n} - w_{m-1,n}}{2\Delta x} - \rho w_{m,n}\frac{w_{m,n+1} - w_{m,n-1}}{2\Delta z}. \tag{25}$$

Since we wish to solve for $u_{m,n}$ and $w_{m,n}$ as variables, we will relabel them as variables instead of as grid points. We label $u_{m,n} = U$ and $w_{m,n} = W$, and thus equations (24) and (25) become

$$F_1 = \mu\left(\frac{u_{m+1,n} - 2U + u_{m-1,n}}{\Delta x^2} + \frac{u_{m,n+1} - 2U + u_{m,n-1}}{\Delta z^2}\right) - \rho U\frac{u_{m+1,n} - u_{m-1,n}}{2\Delta x} - \rho W\frac{u_{m,n+1} - u_{m,n-1}}{2\Delta z}, \tag{26}$$

$$F_2 = \mu\left(\frac{w_{m+1,n} - 2W + w_{m-1,n}}{\Delta x^2} + \frac{w_{m,n+1} - 2W + w_{m,n-1}}{\Delta z^2}\right) - \rho W\frac{w_{m+1,n} - w_{m-1,n}}{2\Delta x} - \rho W\frac{w_{m,n+1} - w_{m,n-1}}{2\Delta z}. \tag{27}$$

At this point we wish to find the roots of F_1 and F_2. Note that Newton's basic

root-finding method [7] is

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)},$$

for some function $f(x)$, where x is the independent variable. The analog of this method for systems of equations is the expression

$$V_{new} = V_{old} - J^{-1}(V_{old})F(V_{old}), \tag{28}$$

where V_{new} represents the 2×1 vector of updated U_{new} and W_{new} values, V_{old} represents the 2×1 vector of initial U_{old} and W_{old} values, $F(U_{old}, W_{old})$ represents equations (26) and (27) evaluated at V_{old}, and J represents the Jacobian matrix.

The next step is to build a Jacobian matrix for equations (26) and (27),

$$\begin{aligned} J &= \begin{bmatrix} \frac{\partial F_1}{\partial U} & \frac{\partial F_1}{\partial W} \\ \frac{\partial F_2}{\partial U} & \frac{\partial F_2}{\partial W} \end{bmatrix} \\ &= \begin{bmatrix} -\rho\frac{w_{m+1,n}-w_{m-1,n}}{2\Delta x} & -2\frac{\mu}{\Delta x} - 2\frac{\mu}{\Delta z} - \rho\frac{w_{m,n+1}-w_{m,n-1}}{2\Delta z} \\ -2\frac{\mu}{\Delta x^2} - 2\frac{\mu}{\Delta z^2} - \rho\frac{u_{m+1,n}-u_{m-1,n}}{2\Delta x} & -\rho\frac{u_{m,n+1}-u_{m,n-1}}{2\Delta z} \end{bmatrix} \end{aligned} \tag{29}$$

Equation (28) must be iterated multiple times in order to converge to a viable solution, and we arbitrarily choose this iteration to run 10 times (this was also due to time constraints in running our simulation). We set $\rho = 1.2\ \frac{\text{kg}}{\text{m}^3}$, as that is the approximate density of air, and then we set our viscosity $\mu = 1\ \text{Pa} \cdot \text{s}$. Although the viscosity of air is on the order of 10^{-6}, the condition number of the Jacobian matrix became too large to allow equation (28) to yield a solution.

By using central differencing schemes, we have boundary conditions set at the left, top, and bottom edges. Thus, $u_{m+1,n+1}$, $u_{m,n+1}$, $u_{m+1,n}$, $w_{m+1,n+1}$, $w_{m,n+1}$, and $w_{m+1,n}$ are all initially set to 0. Therefore, when we solve for $U = u_{m,n}$ and $W = w_{m,n}$ using the root-finding method, we are using velocities in front, above, and below that have

not yet been solved for. Thus, we must perform Newton's nonlinear solving method multiple times in order to propagate the velocity vector field. We set this iteration at $k = 1500$; this number was based on experimenting with how many k iterations it took to propagate the effects of the boundary conditions without condition problems throughout the 20×20 grid, which we found to be approximately 100. Since our grid was scaled by 10 times in both directions, we calculated that the number of k iterations would have to be scaled by $10 \cdot 10 = 100$, however, condition issues with the Jacobian matrix occurred early in the simulation runs. Therefore, although more precise, the velocity field over the 100×100 grid could not be propagated as far (as the field near the left hand side of the grid was completely solved for).

The general code for this method is recorded in Appendix B.

4 Incorporating Topography

4.1 Basic Topography

Adding topography is mathematically analogous to adjusting the boundary conditions, i.e. the wind vector field underneath a landmass is 0. Thus, we adjust the boundary conditions by setting the grid points underneath our specific terrain to have a velocity of 0.

For a basic topography, we start with a simple triangular landmass of height 0.4 km centered at $x = 1$ km. Since we want a triangular "mountain", we will need two linear equations; we term the side of the mountain with positive slope $N(x)$ and the side of the "mountain" with negative slope $L(x)$. We restrict the domain of $N(x)$ to $\{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$, and we restrict the domain of $L(x)$ to $\{x \in \mathbb{R} \mid 1 \leq x \leq 2\}$,

$$N(x) = 0.4x, \tag{30}$$

$$L(x) = -0.4(x-1) + 0.4. \tag{31}$$

To set the grid points beneath the mountain to be zero, we run a series of if-then statements. We had labeled the grid rows as m and the grid columns as n; thus, $n \cdot dx$ is the x-value at any given grid point and $m \cdot dz$ is the z-value at any given grid point. Incorporating this into our Newton nonlinear solver, before we transfer the numerical solutions back into the meshgrid, we ask whether or not the grid point we are numerically solving for the solution at is beneath the set terrain. Thus, we develop two sets of if-then statements, in pseudocode,

```
if ndz <= 1 &&  mdx <= N(ndz) and
elseif ndz >= 1 &&  mdx <= L(ndz).
```

If the grid point (m, n) satisfies either of these requirements, we set $u_{m,n} = 0$ and $w_{m,n} = 0$. Figure 2 is an example of how this inserted code affects a very coarse grid.

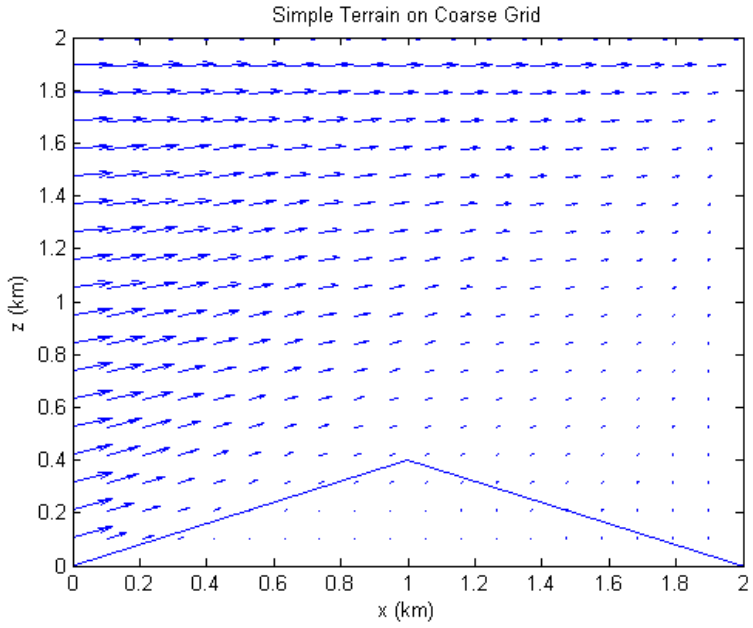

Figure 2: Simple Terrain on Coarse, 20 pt $\times$ 20 pt, Grid

Note in Figure 2 that some of the velocity vectors are still solved for through the

top of the terrain. This is most likely due to a coding error which we are as of yet unable to find.

4.2 Final Topography

Our final topography was based loosely on the shape of the Crazy Mountains in Eastern Montana. We used the following linear equations evaluated on specific domains to create a somewhat jagged terrrain. The equations labeled N_n are the parts of the terrain with positive slope, whereas the equations labeled L_n are the parts of the terrain with negative slope, where $n = 1, 2, ...6$. Figure 3 is a graph of the shape generated by this topography.

$$N_1 = \frac{0.4 + 0.2}{0.3}x \qquad x = \{x \in \mathbb{R} \mid 0 \leq x < 0.3\}$$

$$L_1 = \frac{-0.1}{0.1}(x - 0.3) + 0.6 \qquad x = \{x \in \mathbb{R} \mid 0.3 \leq x < 0.4\}$$

$$N_2 = \frac{0.05}{0.1}(x - 0.4) + 0.5 \qquad x = \{x \in \mathbb{R} \mid 0.4 \leq x < 0.5\}$$

$$L_2 = \frac{-0.2}{0.2}(x - 0.5) + 0.55 \qquad x = \{x \in \mathbb{R} \mid 0.5 \leq x < 0.7\}$$

$$N_3 = \frac{0.15}{0.1}(x - 0.7) + 0.35 \qquad x = \{x \in \mathbb{R} \mid 0.7 \leq x < 0.8\}$$

$$L_3 = \frac{-0.1}{0.1}(x - 0.8) + 0.5 \qquad x = \{x \in \mathbb{R} \mid 0.8 \leq x < 0.9\}$$

$$N_4 = \frac{0.05}{0.1}(x - 0.9) + 0.4 \qquad x = \{x \in \mathbb{R} \mid 0.9 \leq x < 1\}$$

$$L_4 = \frac{-0.06}{0.3}(x - 1) + 0.45 \qquad x = \{x \in \mathbb{R} \mid 1 \leq x < 1.3\}$$

$$N_5 = \frac{0.09}{0.1}(x - 1.3) + 0.39 \qquad x = \{x \in \mathbb{R} \mid 1.3 \leq x < 1.4\}$$

$$L_5 = \frac{-0.1}{0.2}(x - 1.4) + 0.48 \qquad x = \{x \in \mathbb{R} \mid 1.4 \leq x < 1.6\}$$

$$N_6 = \frac{0.04}{0.1}(x - 1.6) + 0.38 \qquad x = \{x \in \mathbb{R} \mid 1.6 \leq x < 1.7\}$$

$$L_6 = \frac{-0.22}{0.3}(x - 1.7) + 0.42 \qquad x = \{x \in \mathbb{R} \mid 1.7 \leq x \leq 2\}$$

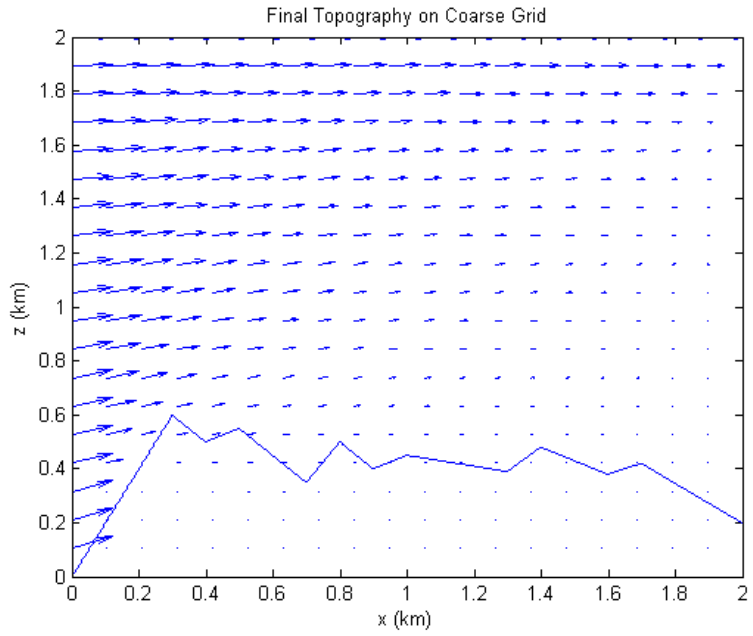

Figure 3: Final Terrain on Coarse, 20 pt × 20 pt, Grid

5 High-Altitude Balloon Payload Trajectory

5.1 Solving for Position: Euler's Method

Having solved the system of equations to get a viable wind field across rugged terrain, we need to track the high-altitude balloon payload as it travels through the grid. Thus, we use Newtonian physics to develop

$$m\boldsymbol{y}'' = -mg + k(\boldsymbol{y}')^2 + f, \tag{32}$$

where $\boldsymbol{y}$ represents the position of the payload, m represents the mass of the object, g represents the force of gravity at a particular altitude, k represents the drag coefficient, and f represents an external force on the object (in this case, the force of our wind field).

The BOREALIS team usually puts together a series of payload boxes that weigh in total 12 pounds or less, per FAA regulations. [4] For simplifying purposes, we assume

that we are flying one rectangular payload that weighs 12 pounds ($m = 5.44311$ kg). By assuming that we have a rectangular payload, we also assumed that the wider side would be on the bottom, allowing us to use the indexed drag coefficient of $k = 2.1\ \frac{1}{\text{m}}$. [3]

Next, we wanted to calculate the gravitational pull of the Earth in the particular area where we are launching the high-altitude balloons. We researched the equation for the strength of gravitational acceleration, [16]

$$g = 9.8\left(\frac{r}{(r+h)}\right)^2, \tag{33}$$

where $r = 6400$ km is the radius of the Earth, and h is the altitude at which we are launching/landing our payload. MSGC's BOREALIS program often launches from Harlowton, MT, and thus we used the altitude of this area to input for $h = 1.278941$ km. This gave us a final gravitational constant of about $g = 9.7961\ \frac{\text{m}}{\text{s}^2}$.

To calculate the position of the balloon, we follow two steps. First, we rewrite equation (32) as two first order differential equations in terms of velocity in the x-direction and z-direction,

$$mv_x' = -kv_x^2 + f_u, \tag{34}$$

$$mv_z' = -mg + kv_z^2 + f_w, \tag{35}$$

where f_u represents the force of the external wind field in the x-direction and f_w represents the force of the external wind field in the z-direction. Notice the horizontal drag is opposing the force of the horizontal wind vector f_u, whereas the vertical is

opposing the force of gravity. We discretize using a central difference scheme,

$$m\frac{v_{x_{t+1}} - v_{x_t}}{\Delta t} = -kv_{x_t}^2 + f_u, \text{ or}$$
$$v_{x_{t+1}} = \frac{-kv_{x_t}^2 + f_u}{m}\Delta t + v_{x_t}, \tag{36}$$

$$m\frac{v_{z_{t+1}} - v_{z_t}}{\Delta t} = -mg + kv_{z_t}^2 + f_w, \text{ or}$$
$$v_{z_{t+1}} = \frac{-mg + kv_{z_t}^2 + f_w}{m}\Delta t + v_{z_t}, \tag{37}$$

and solve these equations using Euler's method. This calculates a series of velocities in the x- and z-directions.

Using the trapezoid method for simple integration [7], we can track the (x, z) position of the balloon throughout the grid,

$$x_{t+1} = x_t + \frac{v_{x_t} + v_{x_{t+1}}}{2\Delta t} \tag{38}$$

$$z_{t+1} = z_t + \frac{v_{z_t} + v_{z_{t+1}}}{2\Delta t} \tag{39}$$

Once the position (x, z) hits the ground, we terminate the code.

5.2 Linear and Bilinear Interpolation

In equations (36) and (37), we have the terms f_u and f_w, which represent the force of the wind field generated previously. We calculated the wind *speed* in the x- and z-directions at the beginning of this project, and thus it needed to be transformed into a *force* in order for the units to be consistent in equations (36) and (37). The

typical equations for converting wind speed into a force are, [17]

$$f_u = 1 \cdot \rho \frac{u(x_t, z_t)^2}{2} A, \tag{40}$$

$$f_w = 1 \cdot \rho \frac{w(x_t, z_t)^2}{2} A, \tag{41}$$

where t represents time, 1 represents our shape factor/drag coefficient, and A represents the surface area of one side of the box that is exposed to the wind. From recollection of measurements from our work with BOREALIS, the side of the payload container is approximately $1.25'$ x $1'$, which is approximately $A = 0.12$ m^2.

Note that equations (40) and (41) require that we use the particular wind speed at the point (x_t, z_t). This point may not fall on our grid, and thus we need to use averaging techniques to approximate the u and w velocities at (x_t, z_t). Two different techniques were applied. If (x_t, z_t) was directly in between two grid points (x_1, z_1) and (x_2, z_2), we approximated the velocities using a linear interpolation method

$$u(x_t, z_t) = u(x_1, z_1) + \frac{u(x_2, z_2) - u(x_1, z_1)}{x_2 - x_1}(x_t - x_1), \tag{42}$$

$$w(x_t, z_t) = w(x_1, z_1) + \frac{w(x_2, z_2) - w(x_1, z_1)}{z_2 - z_1}(z_t - z_1), \tag{43}$$

If (x_t, z_t) was not on a grid point or did not fall exactly between two grid points, we used a bilinear interpolation method. Thus, (x_t, z_t) would be bound between four points on the grid, where x_1 represents the smaller x-value, x_2 represents the larger x-value, z_1 represents the smaller z-value, and z_2 represents the larger z-value. In this case, we want to interpolate the velocities in between the grid points. This

interpolation is achieved with the bilinear interpolation formulas [2]

$$u(x_t, z_t) = a_0 + a_1 x_t + a_2 z_t + a_3 x_t z_t \quad \text{and}$$
$$w(x_t, z_t) = b_0 + b_1 x_t + b_2 z_t + b_3 x_t z_t,$$

where $a_0, a_1, a_2, a_3, b_0, b_1, b_2$, and b_3 are all constants that weight the wind speeds at the grid points accordingly. The constants are calculated from the meshpoints by solving the following linear systems

$$\begin{bmatrix} 1 & x_1 & z_1 & x_1 z_1 \\ 1 & x_1 & z_2 & x_1 z_2 \\ 1 & x_2 & z_1 & x_2 z_1 \\ 1 & x_2 & z_2 & x_2 z_2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} u(x_1, z_1) \\ u(x_1, z_2) \\ u(x_2, z_1) \\ u(x_2, z_2) \end{bmatrix}, \text{ and}$$
$$\begin{bmatrix} 1 & x_1 & z_1 & x_1 z_1 \\ 1 & x_1 & z_2 & x_1 z_2 \\ 1 & x_2 & z_1 & x_2 z_1 \\ 1 & x_2 & z_2 & x_2 z_2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} w(x_1, z_1) \\ w(x_1, z_2) \\ w(x_2, z_1) \\ w(x_2, z_2) \end{bmatrix}.$$

6 Results

Our final wind field was propogated over a 100×100 grid. We dropped the payload into the field at five points in the steadiest region of the wind field: $(0.1, 1.9)$, $(0.45, 1.9)$, $(0.55, 1.9)$, $(0.6, 1.9)$, and $(0.65, 1.9)$. Note these are at a constant height, near the top of the wind field (close to where we would expect the payload to be entering the field). Figures 4, 5, 6, 7, and 8 are graphs of the position of this payload.

In Figure 4, we see no deviation in the x-position of the payload. This is most likely due to the fact that the wind field is relatively consistent here, and has a more significant w component.

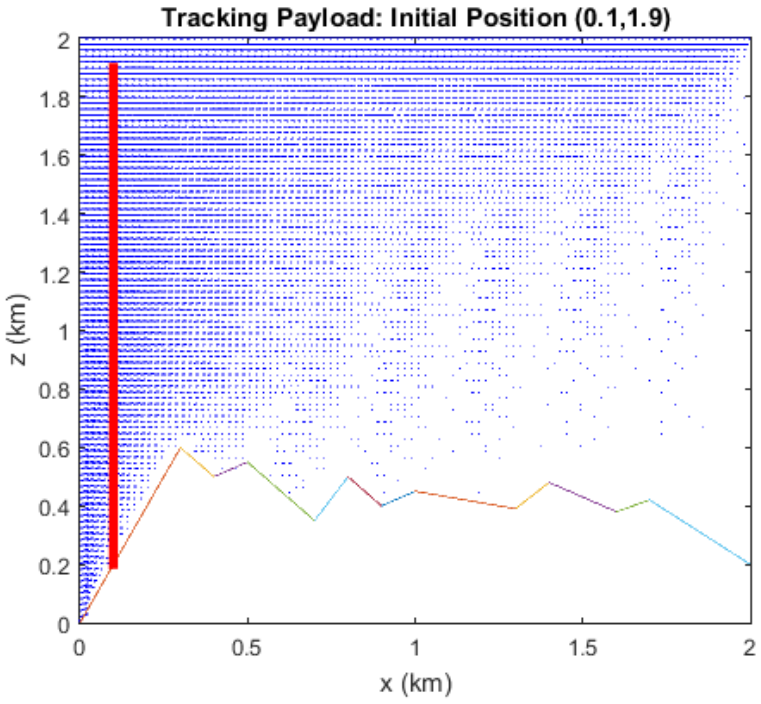

Figure 4: Payload Track with Initial Position $(0.1, 1.9)$

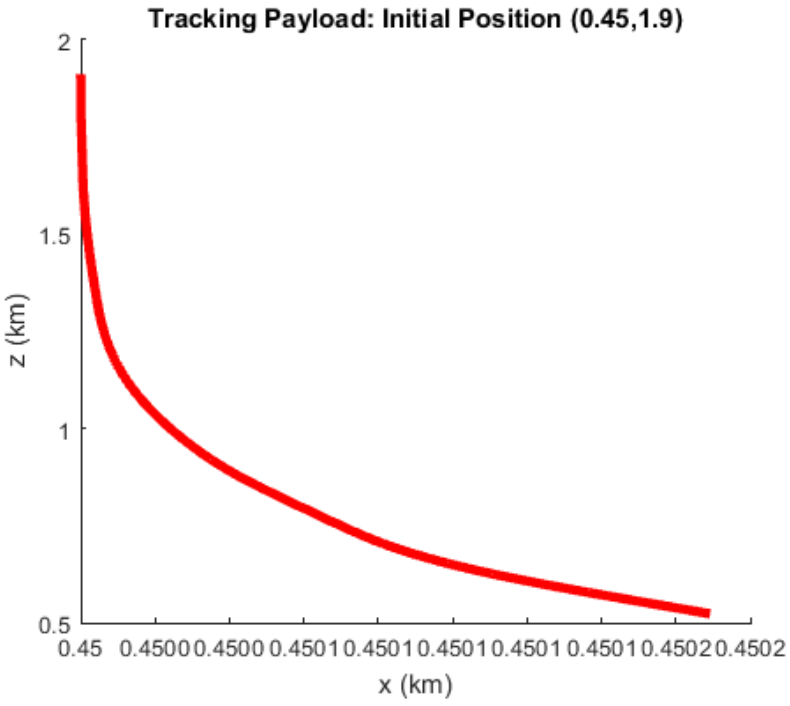

Figure 5: Payload Track with Initial Position $(0.45, 1.9)$

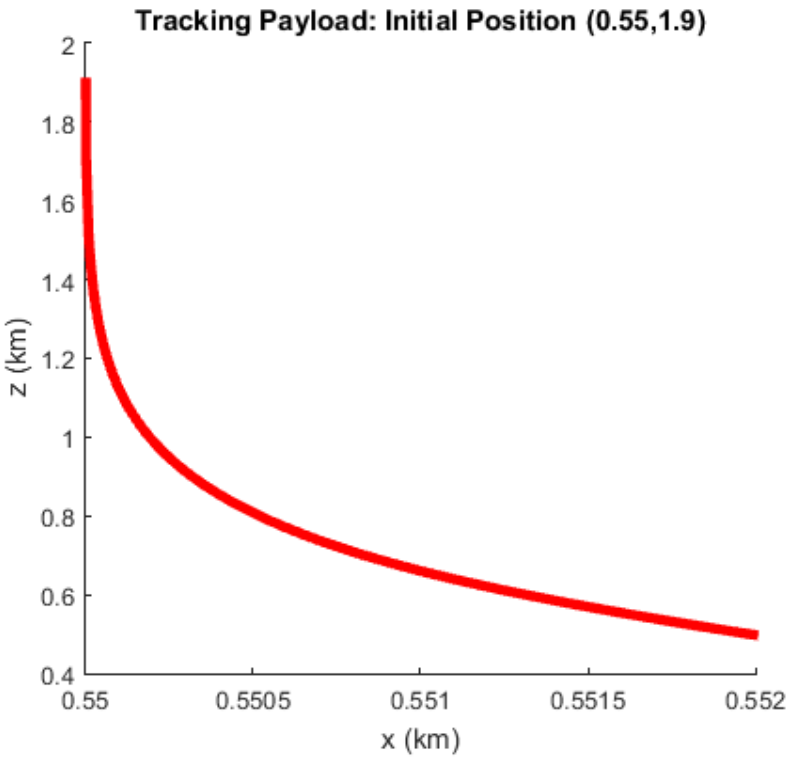

Figure 6: Payload Track with Initial Position $(0.45, 1.9)$

Figure 5 is the graph with the most interesting behavior. We see for roughly the first 0.5 kilometers that the payload's trajectory is rather steady. Below this, the payload's position is pushed more towards the right with two slight deviations in a smooth push to the right. This follows the behavior of the wind field. The image has been magnified for clarity and detail.

In Figure 6, we see a much more significant effect of the wind field on the payload. This is likely due to how the wind field is affected by the terrain closer to the ground. Figures 7 and 8 are smoother than Figure 5 and behave similarly to Figure 6 but yet again with more force in the x-direction.

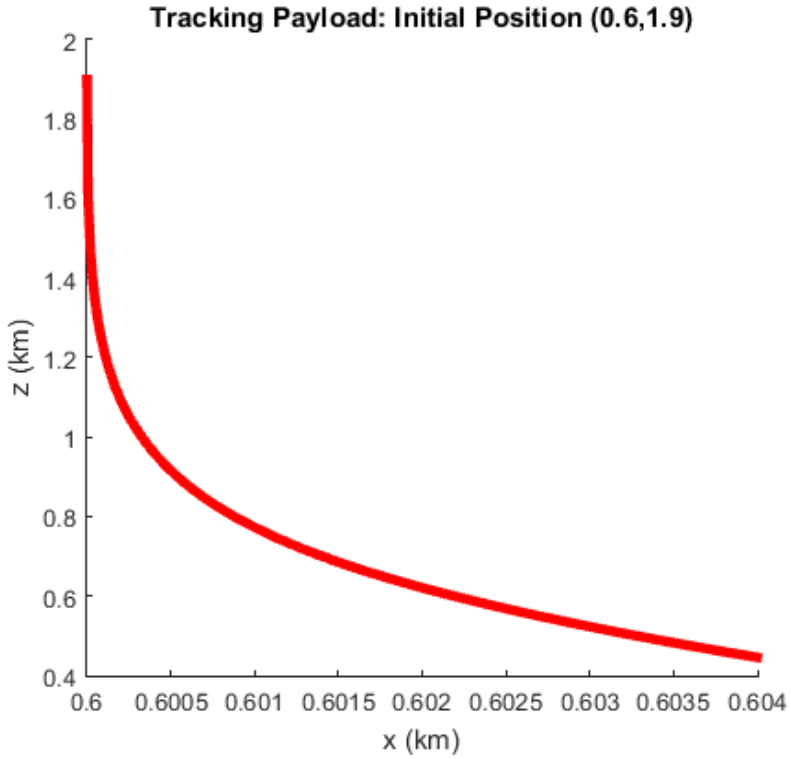

Figure 7: Payload Track with Initial Position $(0.55, 1.9)$

7 Conclusion

7.1 Strengths

This model is a very robust model in that it is developed in two distinct sections. The effects of topography on the wind field are developed first, and then the numerical results are utilized in solving for the position of the payload. This allowed us to work on two separate aspects of the project at the same time. It also allows for further work in both sections independently.

7.2 Weaknesses

One of the greatest weaknesses of this model was the choice to use the Newton nonlinear solver to find solutions to our equations. This led to significant restrictions to the parameter μ, and even restricted how high we set our boundary conditions,

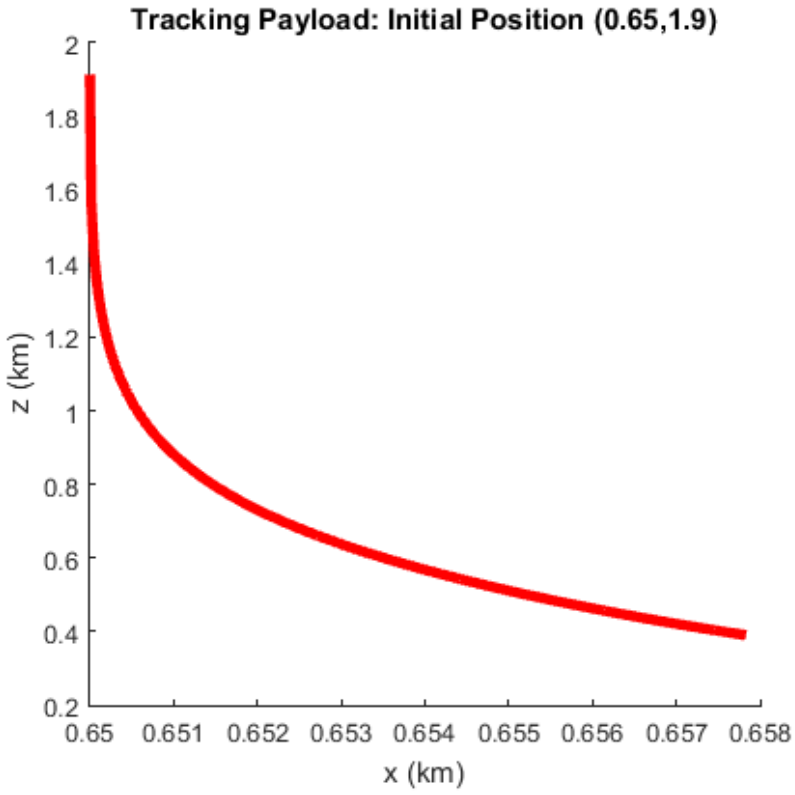

Figure 8: Payload Track with Initial Position $(0.6, 1.9)$

as the Jacobian was easily susceptible to having large condition numbers and led to invalid results. This is often a problematic occurrence when solving the Navier-Stokes equations.

Another weakness of this model was as we increased the grid size, the distance covered by the solver without causing the condition of the Jacobian matrix to increase became shorter. This was most likely due to the number of iterations, as when the velocities at the left end of the meshgrid begin to stabilize, the Jacobian condition number grew to an extent that it was no longer able to perform the iterations.

7.3 Further Research

Currently, in order to improve this model, we want to further explore multiple avenues in both sections of this project. We would like to upgrade to a numerical solution

method that relies less heavily on for loops or solving $A\vec{x} = b$, but rather involves matrix operations. This would significantly reduce the time it takes to run the code. This would also (most likely) mitigate problems caused by singularity of the Jacobian matrix, including the problems caused by the grid size.

In the second part of this project, we would want to investigate incorporating the aerodynamic effects of the parachute and latex balloon tail on two or three separate payloads, each connected by string and weighing a *total* of 12 pounds. This is more generally the set-up used by the high-altitude ballooning team we worked with. We would most likely begin by assuming two simple spheres (these are the most aerodynamically easy to model) connected by a spring. This spring system would most likely form a system of ordinary differential equations to solve. Attempting to model this would allow us to make a better approximation of the trajectory of the payload through the troposphere.

8 References

[1] Bennethum, Lynn Schreyer. Notes for Introduction to Continuum Mechanics. 2011.

[2] "Bilinear Interpolation Example." Bilinear Interpolation Example -csserver. University of Evansville. Web. <http://csserver.evansville.edu/~richardson/courses/EE499_Image_Processing/resources/lectures/l05/BilinearInterpolationExample.pdf>.

[3] "Drag Coefficient." Drag Coefficient. Engineering Toolbox. Web. <http://www.engineeringtoolbox.com/drag-coefficient-d_627.html>.

[4] "FAA Regulations regarding Unmanned Free Balloon Flights." Eta Kuram Na Smekh. Web. <http://etakuramnasmekh.com/faa-regulations-regarding-unmanned-free-balloon-flights/>.

[5] Gibiansky, Andrew. "Fluid Dynamics: The Navier-Stokes Equations." Fluid Dynamics: The Navier-Stokes Equations. Web. <http://andrew.gibiansky.com/downloads/pdf/FluidDynamics:TheNavier-StokesEquations.pdf>.

[6] Jenkins, Michael. "Unit 6: Local And General Winds." Free Online Course Materials. Utah State University, 2008. Web. 21 Mar. 2016. <http://ocw.usu.edu/Forest__Range__and_Wildlife_Sciences/Wildland_Fire_Management_and_Planning/unit6.html>.

[7] Kaw, Autar. Holistic Numerical Methods. University of South Florida, 23 Dec. 2009. Web. <http://nm.mathforcollege.com/#sthash.JO8PWPzZ.dpbs>.

[8] "Navier-Stokes Equations." – CFD-Wiki, the Free CFD Reference. CFD Online. Web. <http://www.cfd-online.com/Wiki/Navier-Stokes_equations>.

[9] "Part II: Partial Differential Equations." Partial Differential Equations. Web. 3 Feb. 2016. <http://pauli.uni-muenster.de/tp/fileadmin/lehre/NumMethoden/WS1011/script1011PDE.pdf.>.

[10] "Reynolds-Averaged Navier-Stokes (RANS) Equations." California Polytechnic State University. Web. <http://www.calpoly.edu/~kshollen/ME554/RANS.pdf>.

[11] "Stress-energy Tensor: Conservation Equations." Physics Pages. 27 May 2014. Web. <http://www.physicspages.com/2014/05/27/stress-energy-tensor-conservation-equations/>.

[12] "Stress-energy Tensor: Conservation Equations." Physics Pages. 27 May 2014. Web. <http://www.physicspages.com/2014/05/27/stress-energy-tensor-conservation-equations/>.

[13] Thermal-FluidsPedia *Reynolds-Averaged Navier Stokes Equations* https://www.thermalfluidscentral.org/encyclopedia/index.php/Reynolds-Averaged_Navier_Stokes_Equations

[14] Trakhtenbrot, A., G.G. Katul, and R. Nathan. "Mechanistic Modeling of Seed Dispersal by Wind over Hilly Terrain." Ecological Modelling 274 (2014): 29-40. Web. 19 July 2015.

[15] Wang, Yingwei. "Methods for Solving Nonlinear Equations." Dept. of Mathematics, Purdue University. Web. <https://www.math.purdue.edu/~wang838/notes/newton.pdf>.

[16] "What Is the Gravitational Acceleration at Altitudes of 100 Km, 1000 Km, and 10,000 Km?" Yahoo! Answers. Yahoo! Web. <https://answers.yahoo.com/question/index?qid=20111022121603AAmsmkp>.

[17] "Wind Speed and Wind Pressure." Wind Speed and Wind Pressure. Web. <http://projects.knmi.nl/hydra/faq/press.html>.

A Notation Index

- $\frac{D}{Dt}$ material time derivative
- $\boldsymbol{\nabla}$ gradient operator
- p pressure
- f_i accelerations acting on the fluid
- u velocity in x-direction (when indexed, just velocity)
- w velocity in z-direction
- ρ density of the air
- μ dynamic viscosity (of air)
- λ bulk viscosity
- m row index
- n column index
- Δx discrete approximation of dx
- Δz discrete approximation of dz
- F_1 first discretized differential equation
- F_2 second discretized differential equation
- J Jacobian matrix
- N_n function for sides of"mountain" with positive slope

- L_n function for sides of "mountain" with negative slope
- k drag coefficient of rectangular box
- f_u force of wind in x-direction
- f_w force of wind in z-direction
- A surface are of payload

B Appendix - Newton Nonlinear Solver Code

```
for kk=1:100
  for m = (mMax-1):-1:2 %mMax: maximum number of grid points in the x direction
    for n = (nMax-1):-1:2 %nMax: maximum number of grid points in the z direction


F = @(UU,WW) [ mu*((w(m+1,n)-2*WW+w(m-1,n))/(dx^2)+(w(m,n+1)-2*WW+w(m,n-1))/dz^2)-...
    rho*UU*(w(m+1,n)-w(m-1,n))/(2*dx)-rho*WW*(w(m,n+1)-w(m,n-1))/(2*dz) ;
    mu*((u(m+1,n)-2*UU+u(m-1,n))/(dx^2)+(u(m,n+1)-2*UU+u(m,n-1))/dz^2)-...
     rho*UU*(u(m+1,n)-u(m-1,n))/(2*dx)-rho*WW*(u(m,n+1)-u(m,n-1))/(2*dz) ];


J = [ -rho*(w(m+1,n)-w(m-1,n))/(2*dx), -2*mu/(dx^2)-2*mu/(dz^2)-rho*(w(m,n+1)-w(m,n-1
    -2*mu/(dx^2)-2*mu/(dz^2)-rho*(u(m+1,n)-u(m-1,n))/(2*dx), -rho*(u(m,n+1)-u(m,n-1))


            Uold = u(m,n);
            Wold = w(m,n);
            UWold = [Uold;Wold];


    for j=1:10
    UWnew = UWold - J\F(UWold(1), UWold(2));
```

```
    UWold = UWnew;
    end
  end
pause(0.1)
quiver(x,z,u,w,'blue')
end
```