PLANETARY ATMOSPHERE MODELS: A RESEARCH AND INSTRUCTIONAL WEB-BASED RESOURCE

by

SAMUEL AUGUSTINE GRAY

JOHN BAKER, COMMITTEE CHAIR
K.CLARK MIDKIFF
MARK WEAVER

A THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in the
Department of Mechanical Engineering
in the Graduate School of
The University of Alabama

TUSCALOOSA, ALABAMA

2014

ABSTRACT

The effects of altitude change on the temperature, pressure, density, and speed of sound were investigated. These effects have been documented in Global Reference Atmospheric Models (GRAMs) to be used in calculating the conditions in various parts of the atmosphere for several planets. Besides GRAMs, there are several websites that provide online calculators for the 1976 US Standard Atmosphere. This thesis presents the creation of an online calculator of the atmospheres of Earth, Mars, Venus, Titan, and Neptune. The websites consist of input forms for altitude and temperature adjustment followed by a results table for the calculated data.

The first phase involved creating a spreadsheet reference based on the 1976 US Standard Atmosphere and other planetary GRAMs available. Microsoft Excel was used to input the equations and make a graphical representation of the temperature, pressure, density, and speed of sound change as altitude changed using equations obtained from the GRAMs. These spreadsheets were used later as a reference for the JavaScript code in both the design and comparison of the data output of the calculators.

The websites were created using HTML, CSS, and JavaScript coding languages. The calculators could accurately display the temperature, pressure, density, and speed of sound of these planets from surface values to various stages within the atmosphere. These websites provide a resource for students involved in projects and classes that require knowledge of these changes in these atmospheres. This project also created a chance for new project topics to arise for future students involved in aeronautics and astronautics.

DEDICATIONS

This thesis is dedicated to everyone who has helped me through to completion of this project. Especially friends and family who provided insight and guidance through this learning process.

NOMENCLATURE

ρ (rho)	Density of air [kg/m^3]
ρ_1	Density at the bottom of an atmospheric layer [kg/m^3]
ρ_{SL}	Density at sea level [kg/m^3]
P	Pressure of air [Pa]
P_1	Pressure of air at the bottom of an atmospheric layer [Pa]
P_{SL}	Pressure of air at sea level [Pa]
T_1	Temperature at the bottom of an atmospheric layer [K]
T	Temperature at current altitude [K]
T_{SL}	Temperature at sea level [K]
R	Perfect gas constant [J/kg/K]
K	Temperature gradient, of an atmospheric layer [K/km]
g	Gravitational acceleration [m/s^2]
h_1	Altitude at the bottom of an atmospheric layer [m]
h	Current altitude [m]
E_R	Earth Radius [km]
a	Speed of Sound [m/s]
γ	Specific Heat Ratio (gamma)
M	Molecular Weight (gram/mole)
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets

GRAM	Global Reference Atmospheric Model
NOAA	National Oceanic and Atmospheric Administration
GCM	General Circulation Model
EDL	Entry, Descent and Landing

ACKNOWLEDGMENTS

I would like to take this opportunity to thank my friends and family who have helped me during this project. I would like to thank the Mechanical Engineering Department for their support and encouragement throughout my undergraduate and graduate years. I especially would like to thank my advisor and committee chairman, Dr. John Baker, who has guided me towards completion of this goal. I would also like to thank my committee members, Dr. Mark Weaver and Dr. Clark Midkiff, for aiding in the process of this project. I would like to thank Mrs. Betsy Singleton, Mrs. Lynn Hamric, and Mrs. Lisa Hinton for always providing a welcoming environment and ensuring I stayed on the correct path. I would like to thank my wife, Kamilah Gray, for her support and motivation to complete this project in time. Finally I would like to thank my parents, Harold and Kathy Gray, who took time out of their day to answer questions I had about my project whenever I asked them for help.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

This section introduces the study of planetary atmospheres and the procedures used to map the various layers of the atmosphere. It also provides a background to the examination of the standard atmosphere. An overview is included of the coding languages used in the websites' creation.

1.1.1 Introduction to Global Reference Atmosphere Models (GRAMS)

GRAMs were originally created to meet the need for a reference model that covered geographical variability, altitude ranges from sea level up to space, coverage of thermodynamic aspects, and wind components. These models can also show other features of the atmosphere to include disturbances that are a result of turbulence and other such atmospheric deviations [1]. GRAMS have been made for several planets and moons of interest to include: Earth, Mars, Jupiter's moon Titan, Venus, and Neptune.

1.1.2 Aeronautical Benefits of Reference Models

The use of atmosphere reference models and tables starts when most Aerospace Engineers are beginning to learn about Aerodynamics. There are several places that an engineer could go to find data that pertains to Earth's atmosphere. Many books and papers concerning

Aerodynamics and similar areas of study contain tables or reference material that concern the atmosphere. One important benefit of these models is the design and testing of aircraft. Creating a standard allows developers to reduce flight test to the same conditions to compare different flights as well as different planes [2].

1.1.3 Educational Benefits of Online Calculators

The internet has different types of calculators that range anywhere from simple arithmetic to differential equations. Besides their uses in pure math they also offer calculators from unit conversions to finances and economics. Sites such as Wolfram|Alpha, DigitalDutch, Google, and Engineering Toolbox offer user friendly online calculators for a wide variety of applications. These sites act as a great resource for students as they progress through their respective curricula. As online calculators continue to become more readily available and more advanced, their benefits to students and professionals will increase.

1.2 Approach

Before a website can be created several conditions must be met in order to achieve this goal. Programming languages must be used to mold the site into the final product. Websites require a domain as well as a server that will host the site. Finally, in order for the site work as intended it must be located in a place that is easy to access as well as user friendly.

1.2.1 Selection of Programming Languages

There are three main programming languages used in the development of a website. These include: HTML, CSS, and JavaScript but are not the only languages that can be used nor are there only these three variations. Each language has its own set of syntax and each language has its own set of rules. Learning these languages was the first step in the process of creating a user friendly website. Online there are several sites that provide FORTRAN code, written by NASA, for creating a program that calculates the standard atmosphere. This code acted as a valuable reference once it was translated [2].

1.2.2 Making the Website Easily Accessible to Students

For this online calculator to be a success, having it in a place easily approachable to students was top priority. The University of Alabama has systems in place to host new websites created by the various colleges. These websites will be linked to the Aerospace Department website and links for each individual website on each page.

1.3 Objective

On top of the classes offered to the students there are also various extracurricular projects that require knowledge of atmospheric behavior. Classes, such as aerodynamics and compressible flow, study the dynamics of gases and the effects of these gases on moving objects. Because of these topics and those of other classes, the atmosphere is often of interest [3]. This

information frequently requires the student to interpolate between given table values in order to calculate data not listed.

The objective of this thesis is to develop online calculators for temperature, density, pressure, and speed of sound in the atmosphere of Earth and select planets/moons of the solar system. These calculators will allow students easy access to reference atmospheric data, and to minimize time spent on studying the Standard Atmospheric tables. These will also prevent students from having to find information on planets/moons that are not typically covered in school books. Integration of these calculators into the Aerospace Department website will help students in their coursework during their time in school.

1.4 Thesis Outline

This thesis is divided into seven chapters. Chapter 2 provides a general view of literature in existence written about the use of atmospheric reference models in the aerospace field, and the use of programming languages in the creation of engineering tools. Chapter 3 details: the creation of the HTML code used to construct the website, the modifications made in order to become user friendly, and methods to blend in with pre-existing University of Alabama design. Chapter 4 provides the construction of the CSS code used to style and position the website. The creation and integration of the JavaScript code involved in making the site a calculator and the equations used in the calculator are the subjects of Chapter 5. Chapter 6 details the implementation and publishing of the website on the University domain. Finally, Chapter 7 discusses the conclusion to the thesis including the results from the project.

CHAPTER 2

LITERATURE REVIEW

2.1 Literature Review Introduction

The atmosphere that surrounds our planet has been and still is one of the most widely discussed topics in the fields of aerospace, meteorology, and weather both inside the classroom and in the workforce. Engineers are constantly working to improve designs on aeronautical vehicles by studying the atmosphere. The literature review in this chapter is intended to acquaint one with the use of online calculators for educational purposes having a distinct emphasis on the 1976 US Standard Atmosphere and atmospheres of Mars, Venus, Titan, and Neptune.

2.2 Website Design

As discussed in chapter 1 of this thesis, website design is comprised of several different programming languages used in their creation. In order to make a website that is straightforward, as well as aesthetically pleasing, the creator must have a considerable understanding of these languages.

2.2.1 HTML Programming

In 1989 Tim Berners-Lee, a member of the computer services section of the European Laboratory for Particle Physics, invented the Web with the hope that researchers from across the globe could collaborate and access each other's work from their computers [4]. In order for this

to work Mr. Lee needed a way to link these documents on his computer to other professionals' computers and thus the idea of using a hypertext was inspired. At the time hypertext was already introduced into the world in both the classroom and by computer companies such as Macintosh's HyperCard. Raggett, et al (1998). In an online article from 1995 a group explains that HTML is actually an application of the Standard Generalized Markup Language [5]. This system defines these structured documents and acts as the root language for HTML and similar languages. At the time, it was this globally accepted language that HTML was based on. Berners (1995).

Since its creation, HTML has grown and evolved to a much larger and more useful web language. On the forefront of its development, web browser developers such as Netscape, established new tags and attributes for the language [6]. After many versions and similar languages created, the current version is now at HTML5. The concept of HTML5 began back in 2004 with support from the browsers Mozilla Firefox and Opera. Due to many different groups and developers trying to implement changes to the language, Ian Hickson from WHATWG, states:

> "Features have thus arisen from many sources, and have not always been designed in especially consistent ways. Furthermore, because of the unique characteristics of the Web, implementation bugs have often become de-facto, and now de-jure, standards, as content is often unintentionally written in ways that rely on them before they can be fixed." [7]

Problems such as these present challenges to the developers but has not slowed down its progression.

2.2.2 CSS Programming

Although HTML can add styles and formats to its members there is another language, CSS, which was made specifically for this feature. CSS got its beginning in 1994 when the current style options offered by browsers at the time were too basic [8]. Now people consider HTML to be the structure of a website while CSS is known as the presentation in which both work together in its establishment. Style sheets are an organized set of instructions that show how elements in a given HTML code should be displayed on the page. This set of instructions ranges anywhere from the color and font to position and size of the items [9].

Like HTML, CSS has evolved from its initial release. As it currently stands CSS is on version CSS3 and maintains most of the original syntax. The idea of CSS is to have a system in place for designing that is a separate part of your website than its structure [10]. In Figures 2.1 and 2.2 below is a demonstration of what a site looks like without CSS and with CSS. As shown by the Figure 2.1, not only are many of the HTML words missing due to their color, but there is also no alignment or background at all due to the lack of CSS shaping the HTML.

Mars Venus Titan Neptune

SI Units or **English Units**

5928 m

0 Kelvin

249.63 Kelvin

47647.89 Pa

0.6649724 kg/m^3

316.72 m/s

Disclaimer Copyright © 2009

Data obtained from the 1976 US Standard Atmosphere

Figure 2.1. Sample of HTML code without CSS file

In Figure 2.2 the full effects of the style sheets can be seen by observing the added colors, alignments, and information previously hidden.

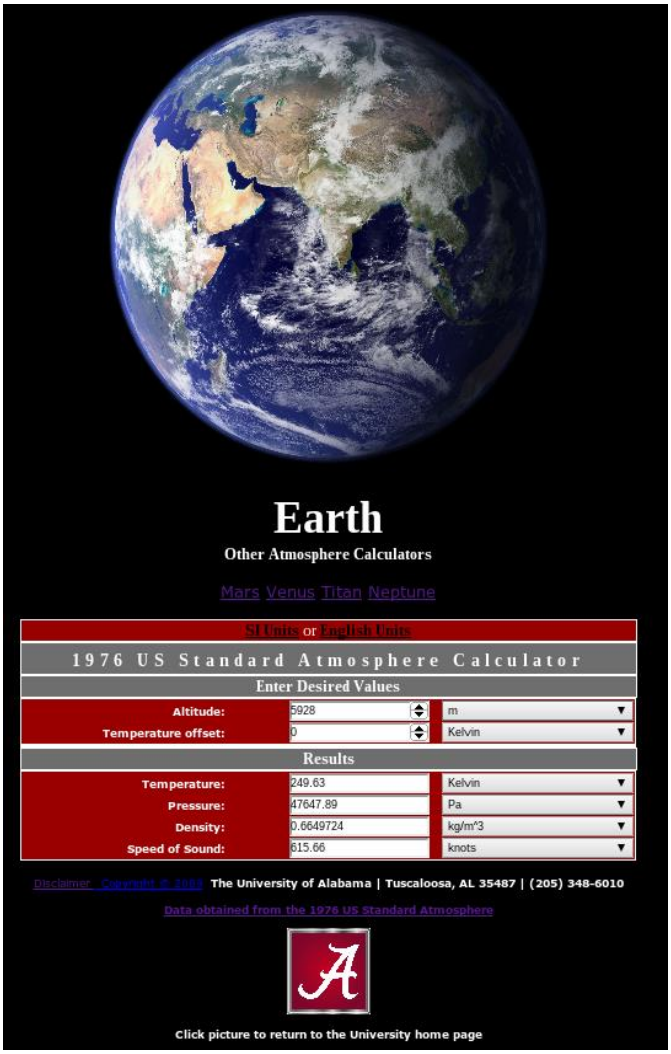

Figure 2.2. Sample of HTML code with corresponding CSS file

2.3 JavaScript Programming

JavaScript was created back in 1995 by a man named Brendan Eich. Netscape hired Mr. Eich with the condition that he create the prototype to a new language in ten days [11]. This condition was feasible because Eich had a background in developing new programming

languages since his days at the University of Illinois and at Silicon Graphics. Netscape wanted a language that would complement HTML in their browser, Netscape 2.0, which was similar to Java yet something that would also attract the more amateur programmers rather than just professionals. Both JavaScript and Java derive much of their syntax from the programming language, C, and are meant to supplement each other rather than compete. Severance (2012). In an online article Eich is quoted:

> "If I had done classes in JavaScript back in May 1995, I would have been told that it was too much like Java or that JavaScript was competing with Java … I was under marketing orders to make it look like Java but not make it too big for its britches … [it] needed to be a silly little brother language." Severance (2012).

JavaScript started out with more basic intentions and structure but has since evolved into a very dynamic and powerful programming language.

2.3.1 Functions as an Interactive Website Scripting Language

JavaScript soon became a very widely used language in the world of the web. More and more people were demanding a way to have their website do more than simply display information. JavaScript is utilized to make a web page dynamic by being responsive to user input, item movement, color changing on its own, or even pop-up windows [12]. These aspects create a much more versatile and dynamic web page. Due to its enormous amount of success, it soon became the international standard for online script work and has been called the "glue that holds web pages together". Eich said in an article. Andreesen (1998):

"Calling JavaScript 'the glue that holds web pages together' is short and easy to use, but doesn't do justice to what's going on. Glue sets and hardens, but JavaScript is more dynamic than glue. It can create a reaction and make things keep going, like a catalyst."

When JavaScript is used to manipulate HTML to create an interactive website it is called Dynamic HTML, or DHTML for short. As the language and the web grow so does the consumer base. This has allowed companies and people to create extensions and improvements for the language without altering the internal aspects. Along with these improvements, JavaScript is included on almost every consumer computer that leaves the factory. This makes it something anybody could use to enhance their creations without having to leave their home [13].

2.3.2 Language for Desktop Widgets and Applications

The web has become available to not only computers but to also to mobile devices. Applications and widgets have a very versatile use both online and offline. Due to its open source nature many varieties of JavaScript have become useful in the world of these applications and widgets. Unlike the original JavaScript made for the enhancement of web pages, it has become a much larger force in the programming world for applications and widgets [14].

2.4 Global Reference Atmospheric Model

Global Reference Atmospheric Models, or GRAMS, have been developed for Earth, Mars, Titan, Neptune, and Venus at NASA's Marshall Space Flight Center. These GRAMs were created to meet the need of atmospheric characterization at potential solar system destinations. As stated in a 2006 article about these models [15]:

"A common characteristic of all these models is their ability to simulate seasonal and time-of-day variability and uncertainty through the application of quasi-random perturbations to mean atmospheric parameters."

Because of these characteristics they assist in the development of aero-assist applications for current and future missions to these planets. Duval, et al. (2006). Besides their benefit towards future exploration, they also are used for "scientific study, orbital mechanics, and lifetime studies, vehicle design and performance criteria, altitude control analysis problems, analysis of effects of short-term density variation caused by geomagnetic storms, aero braking and aero capture analysis, and dynamic response to turbulence or density shears." [16]

The GRAMS, such as the model for Mars, are built based on both simulated experiments and observations by space probes and rovers. Using a combination of these techniques and the utilization of the Monte Carlo method, using a large number of repeated random sampling due to an unknown probabilistic entity to obtain numerical results, to create the GRAM for the planet [17]. Once created they can calculate the average temperature, pressure, density and wind conditions at any height within the atmosphere. Besides these conditions the GRAM is designed to also take into account variations between different times of day and different days in the planet's year. Addy, et al. (2004).

In order for a model to account for variations and random perturbation there is a standard atmosphere reference. This standard atmosphere is a topic widely studied on its own apart from the GRAM. The GRAM provides all the necessary information and equations necessary to map out what the atmosphere will look like on a standard day. With this information it is possible for engineers and engineering students to perform aeronautical calculations and experiments

concerning not only Earth but for other atmospheres as well. Justus, et al. (1999).

2.5 1976 US Standard Atmosphere

The FAA (2008) explains the atmosphere as being just like the oceans and the land in that it has its own unique structure and behaviors. It is one large gas mixture that envelopes the entire Earth extending from its surface up to space. Just like other fluids, the atmosphere is constantly flowing and changing pressure due to its nature. The perfect gas law states that pressure change has an effect on temperature and density. Furthermore, as temperature changes so does the speed of sound. The perfect gas law is also affected by which gases are being measured by changing the specific gas constant used in the law. Earth's atmosphere is 78.04% Nitrogen, 20.95% Oxygen, and approximately 1% other gases and gives a unique gas constant for that atmosphere. These characteristics are very important to the world of aeronautics because of how these changes affect aeronautical equipment [18].

Gyatt (2011) States the standard atmosphere is a scientific representation of the effects of altitude change on temperature, pressure, density, and speed of sound. The 1976 US Standard Atmosphere is defined by a 2011 article as:

> "It is an idealized, steady state representation of the earth's atmosphere from the surface to 1000 km, as it is assumed to exist during a period of moderate solar activity." [19]

CavCar (2011) discusses the atmosphere as never truly constant, although, the mean values offer an enormous benefit to the development of aeronautical equipment [20].

The atmosphere is broken up into the following layers:

- Troposphere (0 to 12 km)
- Stratosphere (12 to 50 km)
- Mesosphere (50 to 80 km)
- Thermosphere (80 to 700 km)
- Exosphere (>700 km).

The temperature is calculated differently for each layer and even has multiple equations within those layers. In a NASA document concerning the Earth GRAM, the authors discuss these different conditions as follows:

"The temperature below 86 km is specified by segments which are linear functions of geopotential altitude. Between 86 km and 91 km the atmosphere is assumed to be isothermal and between 110 km and 120 km the temperature is a linear function of altitude. Between 91 km and 110 km the temperature is represented by a section of an ellipse. Above 120 km the temperature increases exponentially and is asymptotic to 1000°K." [21]

Figure 2.3 shows a representation of the temperature profile of the 1976 US Standard Atmosphere up to 86 km using data as seen in Appendix D Tables D.1, D.2, and D.3.

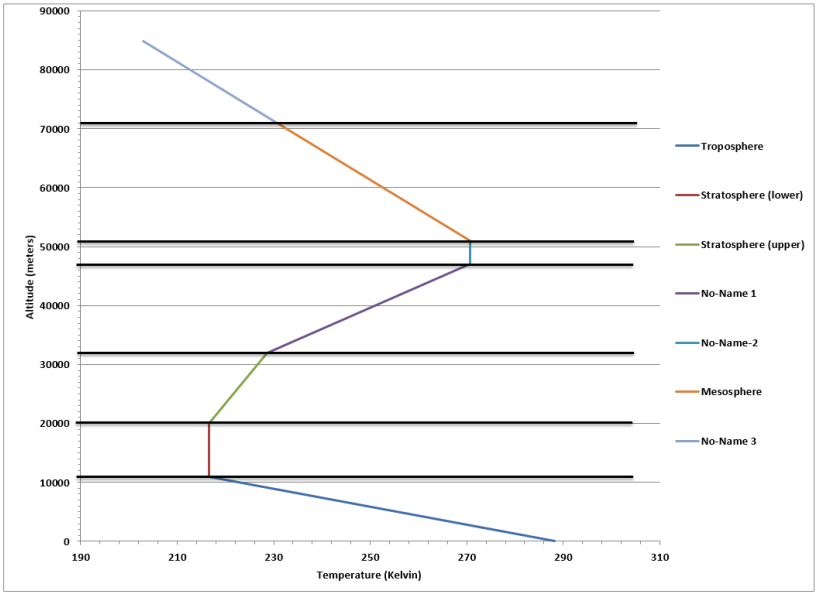

Figure 2.3. Temperature vs. altitude of the 1976 US Standard Atmosphere

Here it can be seen that the temperature linearly decreases, linearly increases, and remains constant as the altitude changes depending upon layer and thus changing the pressure, density, and speed of sound.

The purpose of this project is to create an online calculator for the atmosphere as it pertains to the Troposphere, Stratosphere, and Mesosphere. As stated previously, this is the area of the atmosphere under 86 km in which the temperature functions linearly with geopotential altitude. The 1976 US Standard Atmosphere is defined in Standard International (SI) units but will have the ability to convert between a wide variety of units. Provided by NASA is a paper that includes equations, constants, tables, and graphs of the data for the standard atmosphere [22]. Besides the prior information there is also an open source code, written in FORTRAN, provided by NASA and shared through several other websites that models the atmosphere [23].

2.6 Titan, Mars, Venus, and Neptune Atmospheres

Like Earth, other planets and their satellites in our solar system also have atmospheres. These atmospheres range from different sizes to different compositions making their profiles different and unique from that of Earth's. In order to plan for future missions, both manned and unmanned, these atmospheres must be understood so that engineers can make appropriate equipment. While creating these atmospheric profiles scientist must assume that laws of nature, such as the perfect gas law, still apply on these alien worlds. Figure 2.4 shows a recreation of the temperature profiles for each using data from Appendix D Tables D.4-D.11 from a 2007 paper [24].

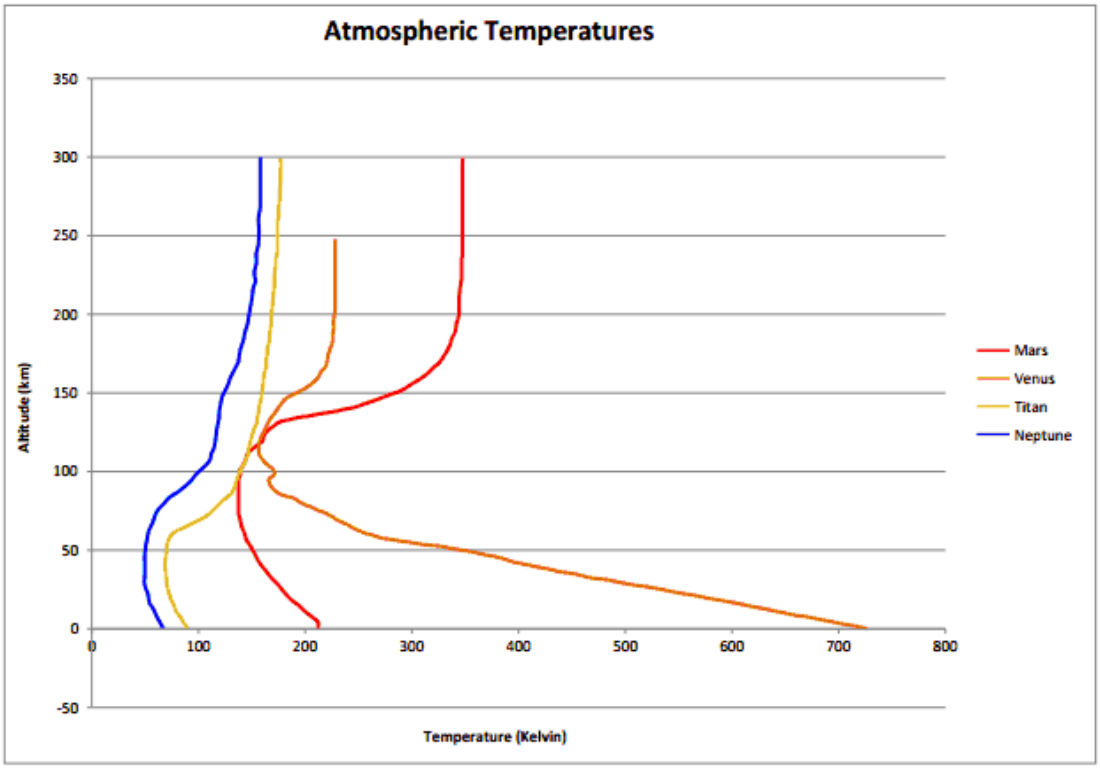

Figure 2.4. Temperature vs. altitude of each atmosphere. Justus, et al (2007).

Titan is Saturn's largest moon and is different than other moons that have been observed in the solar system. Unlike others, its atmosphere is much thicker and extends over 600 km out from its surface and is made up of 95% Nitrogen and 5% Methane gases [25]. During various unmanned missions to this moon many facts about the atmosphere were discovered. On the surface the temperature was calculated to be around 87.39 K, the pressure 147820 Pascal, and density 5.832 kg/m^3. Using the measurements made by the Voyager vehicle, scientists were able to create temperature, density, and pressure profiles of Titan's atmosphere [26].

Mars is the fourth planet from the sun and the second smallest. Despite the difference in its atmosphere from Earth's, being 95% Carbon Dioxide, 2.7% Nitrogen, 1.6% Argon and the remaining 0.7% various gases, it behaves similarly in many ways [27]. Both planets have the highest temperatures around their equators, close to the same rotation time, and similar seasonal patterns. On the surface the temperature was calculated to be around 212.06 K, the pressure 557.4 Pascal, and the density 0.0189 kg/m^3. Using the measurements made by the Mars rovers, space probes, and telescopes, scientist were able to create temperature, density, and pressure profiles of Mars's atmosphere [28].

Venus is the second planet from the sun and almost as large as Earth. The inside of the planet behaves like Earth but on the outside it operates under a very different atmosphere. Venus's atmosphere is made up of 96.5% Carbon Dioxide, 3.4% Nitrogen, and less than a percent various gases. On the surface the temperature was calculated to be 735 K, the pressure 9,210,000 Pascal, and the density of 64.79 kg/m^3 [29]. Due to the high temperature and pressure on Venus, space probe mission attempts to land on the surface were challenging. Using these

measurements scientist have been able to create a temperature, density, and pressure profiles of Venus's atmosphere [30].

Neptune is the eighth planet from the sun and the fourth largest in the solar system. The atmosphere is much larger in comparison to Earth's and its surface is liquid rather than solid. Inside the atmosphere the winds can get up to nine times stronger than that of Earth's, creating another challenge to missions headed to the planet [31]. Neptune's atmosphere is made up of 80% Hydrogen, 19% Helium, and 1% Methane and on the surface the temperature was calculated to be 67 k, the pressure is 1000 bars, and the density 412.45 kg/m^3 [32].

2.7 Literature Review Summary

In this chapter the application of online calculators in computing atmospheric data has been explored in great detail. The use of GRAMS and online calculators in industry uses, research, and in students' education has been discussed. The material presented in this chapter is only a small percentage of the results that pertain to the website coding as well as the atmospheric models. This thesis was started to create a resource in the aerospace and mechanical engineering departments through the development of these online calculators. In this respect, this thesis builds upon the experiences of students involved in classes and projects that pertain to aeronautical studies that will be valuable to students at the University of Alabama.

CHAPTER 3

DEVELOPMENT OF THE HTML CODE

3.1 Introduction to HTML in an Online Calculator

As stated in the Literature Review of this thesis, HTML is the skeleton or structure of a website. Any words, pictures, links, and information that is going to be presented on a website is created in the HTML portion of a website's code. For an online calculator to work, the HTML must create a layout that will make it possible for not only the student to use it, but one that allows the other code languages to perform their duties.

HTML follows its own set of rules and syntax that have to be learned like any other language. Websites, such as Codecademy, offer in-depth training on the applications and abilities of this language [33]. On top of training locations, there are also syntax references and glossaries, such as w3schools, that provide a list and description of the different syntax used by the language [34]. The creation of the HTML code for the online calculators is the topic of this chapter.

3.2 Linking Other Code to HTML

HTML, like most languages, comes with a dictionary full of built-in functions. These functions help keep the designer from having to re-invent something that has already been accomplished. These functions range anywhere from linking JavaScript and CSS files to the HTML code; to utilizing functions from those files with a simple command. In order for

JavaScript and CSS to be able to work as intended they must first be linked to the HTML. Figure 3.1 shows the CSS and JavaScript code linked to the HTML.

```
<!DOCTYPE html>
<html lang="en">

<head>
</style>
<link rel="stylesheet" type="text/css" href="style.css">
<meta charset=utf-8>
<title> 1976 US Standard Atmosphere Calculator</title>
<script type="text/javascript" src="script.js"></script>
```

Figure 3.1. HTML beginning and code linkage

This code represents the top section of HTML. The code on line one lets the web browser know which version of hypertext markup language is being used in the code. Then the document's default language is declared to be English. The links to JavaScript and CSS along with the website title to be displayed in the search bar are written within the head tags. As shown within this tag is a description of the link that is provided along with the type and file name. This set of code allows interactions between the three sets of code.

Once the other files are linked, then the designer may begin to use functions defined in JavaScript and CSS for use in the HTML code. Figure 3.2 shows a series of code that has both JavaScript uses and CSS uses distributed throughout the HTML.

```
<body onload="loadTheSelection()"
 onunload="saveTheSelection()">
        <div class="overall">
        <div class="largegroup1">
        <div class="EarthPicture">
```

Figure 3.2. HTML assortment of code links

This code starts off with the body onload, and onunload functions created in the JavaScript program. This tells the web browser that upon loading or leaving the website to activate these functions. Below this there are several tags called divisions, or divs, which group the items into respective sections. The class identified next to the division is used as an identifier for the CSS to style that particular class with CSS code. These divisions are used so that the designer may group various parts together and even contain sub-groups within these divisions allowing for many options on the placement of each division.

Once the code has been broken up into divisions and linked then the designer can start placing the items in the order desired on the page. Here there is a linked picture of Earth and a quick description of the page with links to the other atmosphere online calculators. These links display the name of which calculator they lead to and provide the link to this individual website. As discussed previously, HTML has the ability to style words without the help of CSS and can be useful when styles are different within a class. Figure 3.3 shows these various links and the JavaScript functions used to populate the page with certain units upon a click.

```
<div class="PlanetCalc"><strong style = "color: white">Earth</strong></div>
<div class="planetlinks"><strong style = "color:white">
Other Atmosphere Calculators</strong>
<p class= "planetlinks" style="font-size: 24px; color: #000; text-shadow: 2px 2px 3px #b22e20">
    <a href="Insert Link Here"><font color="#FF2400">Mars</font></a>
    <a href="Insert Link Here "><font color="#CC6600">Venus</font></a>
    <a href="Insert Link Here "><font color="#FDD017">Titan</font></a>
    <a href="Insert Link Here "><font color="#00008B">Neptune</font></a>
</p>
</div>
</div>
<div class="largegroup2">
<div class="Titlebar"><a href="#" onclick="TheUnitsTheStandardWorldPick();return
false;"><font color="000000"><strong>SI Units</strong></font></a> or <a href="#"
onclick="TheEnglishUnits();return false;"><font color="000000"><strong>
English Units</strong></font></a>
</div>
```

Figure 3.3. JavaScript uses for HTML buttons

There are names and types defined here that will be used in the JavaScript functions such as TheEnglishUnits (). This can be seen in Appendix C where this name is used inside of JavaScript to perform a certain task. This is shown at the bottom of the section with options for SI and English unit selections that activate this JavaScript code upon the user clicking the mouse over the button.

3.3 Utilizing Pre-Existing Functions within Language

HTML has many functions as seen previously that are prebuilt by its developers to perform certain tasks when called. One such function allows the designer to create a drop down selection list using the option value tag. The option value tag can be seen in Figure 3.4.

```
<div class="Header" style = "color: white"><strong>Results</strong></div>

<table class="Results">
 <tr>
  <td class="tdclassstyle"style = "color: white"><strong>Temperature:</strong></td>
  <td><input type="text" size="20" name="TemperatureAdjustment" readonly></td>
  <td>
    <select name="TemperatureUnitID" onchange="calculationFunction();">
     <option value="Celsius">Celsius</option>
     <option selected="" value="Kelvin">Kelvin</option>
     <option value="Fahrenheit">Fahrenheit</option>
     <option value="Rankine">Rankine</option>
     <option value="Réaumure">Réaumure</option>
    </select>
  </td>
 </tr>
```

Figure 3.4. Option value tag

The option value tag allows the user to present as many choices as needed between these tags. As shown above not only is the name typed between the tags but the value as well. These values are all used as identifiers in functions discussed in the JavaScript chapter of this thesis. In each of these divisions the code provides a unique name for JavaScript identification and activates a function upon the user selecting one of these options. This tells the code to start a new calculation upon any change that takes place.

3.4 Creating a Website that Blends with University of Alabama Part I

The University of Alabama has its own style and attributes on its various web pages. Most of these styles are handled in the CSS portion of the website's code, but there is information that must be written in HTML. In Figure 3.5 there are items that lead the user to straight to the University Of Alabama Aerospace Department.

```
<div class="EarthPicture">
<a href="http://aem.eng.ua.edu/"><img
src="http://upload.wikimedia.org/wikipedia/commons/6/6f/Earth_Eastern_Hemisphere.jpg"
alt="http://upload.wikimedia.org/wikipedia/commons/thumb/5/50/ISS034E016601_-_Stratocumul
us_Clouds_-_Pacific_Ocean.jpg/640px-ISS034E016601_-_Stratocumulus_Clouds_-_Pacific_O
cean.jpg" width="250" height="250"></a>
</div>

<div class="PlanetCalc"><strong style = "color: white">Earth</strong></div>
```

Figure 3.5. Links to the University of Alabama

The first part of this code links a NASA image of Earth, taken by Apollo 17 in 1972, to the Aerospace Department when the user clicks the picture [35]. HTML also has code that allows alternate sources for links and images in case of the primary source's failure. This lets the designer set up a backup in case the browser does not load the first choice or if the first choice is taken off the web. The last class below the picture is a simple description for the user to inform which calculator.

The University of Alabama provides a copyright notice and a disclaimer at the bottom of their websites in order to provide the user with this information if needed. HTML displays objects in the order that they are presented in the code and since the copyright information needs to be at the bottom of the page, then it must be at the bottom of the HTML code. This code is represented in Figure 3.6.

```
<p class="uainfo" style="color: white">
<a href="http://www.ua.edu/disclaimer.html">Disclaimer</a>
<a href="http://www.ua.edu/copyright.html">   Copyright &copy; 2009</a><strong> 
The University of Alabama | Tuscaloosa, AL 35487 | (205) 348-6010</strong></p>
<p class="uainfo" style="color: white"> <a
href="http://modelweb.gsfc.nasa.gov/atmos/us_standard.html"><strong>Data obtained from the
1976 US Standard Atmosphere</strong></a></p>

 <div class="UAPic">

 <a href="http://www.ua.edu/">
<img src="http://visualid.ua.edu/download/Capstone%202013/UA_Square_Logo.jpg"
alt="http://visualid.ua.edu/download/Capstone%202013/UA_Stack_Logo_4c_noNameplate.jpg"
align="center" width="53" height="53"></a>
</div>

</body>
</html>
```

Figure 3.6. Final HTML code

Below the disclaimer and copyright is the University of Alabama link through a Capstone image provided by the Alabama webmaster. There is also a link to the source of the information used to obtain the data for the website so that the user may study the origins of the results. The last two lines of code are the final tags in an HTML document that close off the beginning tags. These tags can be considered as a period in a sentence used to express the end of the document.

3.5 User Interface

HTML has provided the tools necessary to make the website very easy to use and understand. The user immediately sees what atmosphere calculator they are using and the host

of the website. There are simple instructions and ability to adjust items as necessary to encompass any changes the user needs to apply to the data. Figure 3.7 shows the completed website and the interface presented to the user.

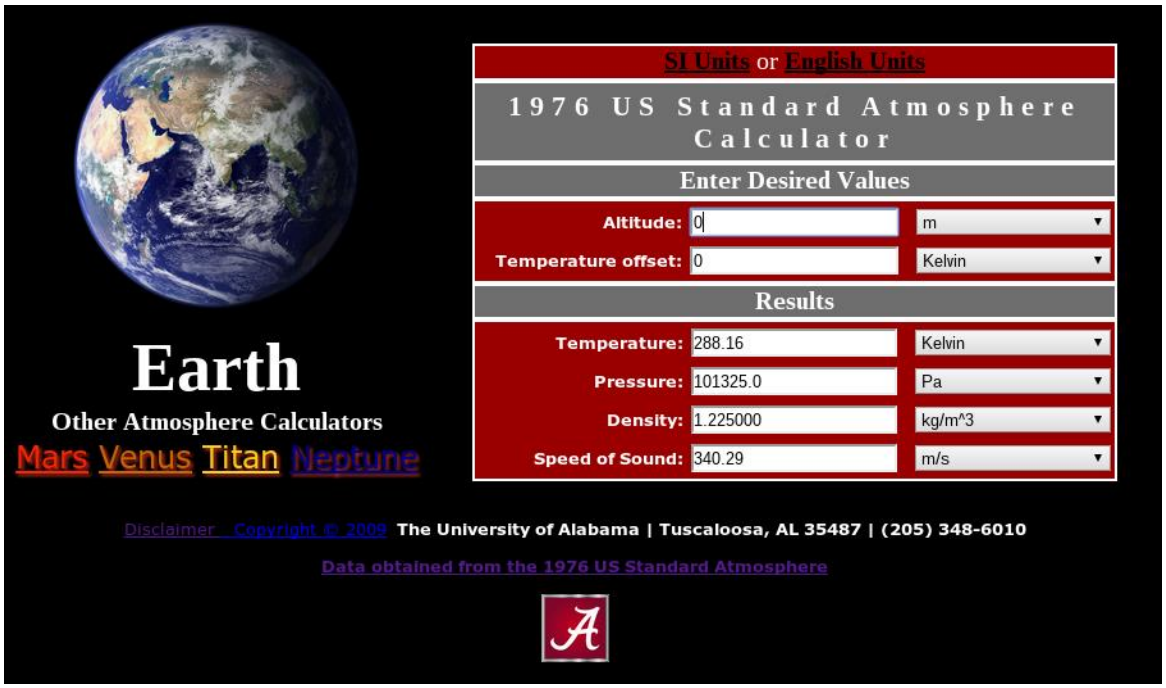

Figure 3.7. User interface and website layout of Earth

As shown above the webpage starts with a picture of Earth and "Earth" to let the user know which calculator they are using. Below Earth are several links that will bring the user to the calculators of other atmospheres for quick access. Next to this picture are the calculation tables where the user can input/ change the desired values and see the results populate the table. The bottom of the webpage displays the University of Alabama information and appropriate links that bring the user to the source of the atmospheric information and back to the university's home page.

3.6 HTML for Other Planetary Calculators

The HTML for the other four atmosphere calculators, for the most part, contains the same code. This process allows for the same layout as the Earth atmospheric calculator making them all familiar to the user. Figures 3.8 and 3.9 show the minute differences in the HTML code and website layout of Mars. Figures 3.10 and 3.11 show the minute differences in the HTML code and website layout of Venus. Figures 3.12 and 3.13 show the minute differences in the HTML code and website layout of Titan. Figures 3.14 and 3.15 show the minute differences in the HTML code and website layout of Neptune.

```
<div class="MarsPicture">
<a href="http://aem.eng.ua.edu/"><img
src="http://upload.wikimedia.org/wikipedia/commons/e/e4/Water_ice_clouds_hanging_above_T
harsis_PIA02653_black_background.jpg"
alt="http://upload.wikimedia.org/wikipedia/commons/7/7d/Mars_atmosphere.jpg"
 width="250" height="250"></a>
</div>

<div class="PlanetCalc"><strong style = "color: white">Mars</strong></div>

<div class="planetlinks"><strong style = "color:white">Other Atmosphere Calculators</strong>
   <p class= "planetlinks" style="font-size: 24px; color: #000; text-shadow: 2px 2px 3px
#b22e20"><a href="http://aem.eng.ua.edu/"><font color="#2B65EC">Earth</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#CC6600">Venus</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#FDD017">Titan</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#00008B">Neptune</font></a>
   </p>
</div>
</div>
```

Figure 3.8. Mars HTML code difference

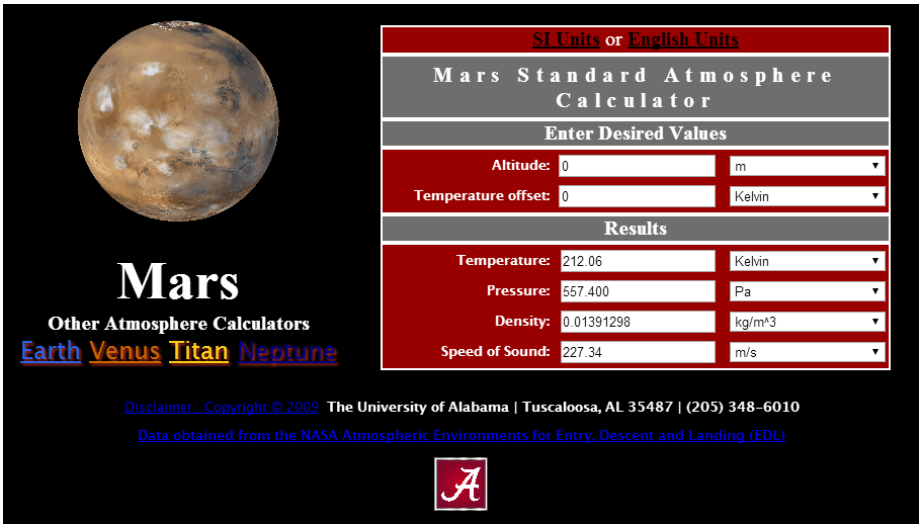

Figure 3.9. User interface and website layout of Mars

```
<div class="VenusPicture">
<a href="http://aem.eng.ua.edu/"><img
src="http://upload.wikimedia.org/wikipedia/commons/8/85/Venus_globe.jpg"
alt="http://upload.wikimedia.org/wikipedia/commons/e/e5/Venus-real_color.jpg"
width="250" height="250"></a>
</div>

<div class="PlanetCalc"><strong style = "color: white">Venus</strong></div>

<div class="planetlinks"><strong style = "color:white">Other Atmosphere Calculators</strong>
<p class= "planetlinks" style="font-size: 24px; color: #000; text-shadow: 2px 2px 3px #b22e20">
<a href="http://aem.eng.ua.edu/"><font color="#FF2400">Mars</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#2B65EC">Earth</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#FDD017">Titan</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#00008B">Neptune</font></a></p>
</div>
</div>
```

Figure 3.10. Venus HTML code difference

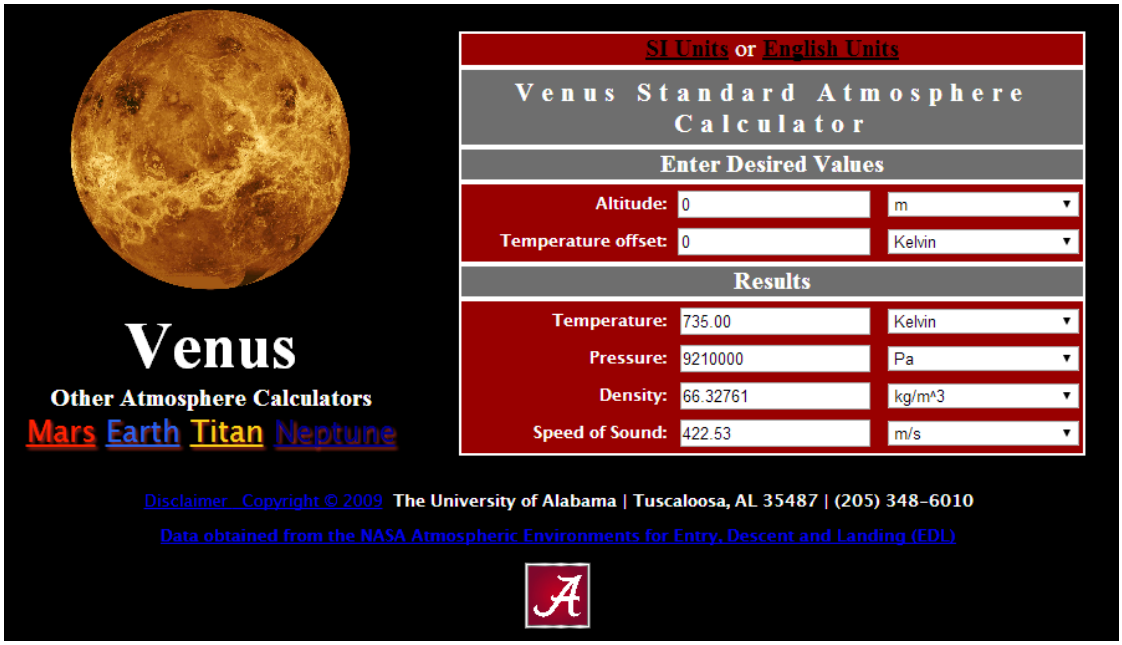

Figure 3.11. User interface and website layout of Venus

```
<div class="TitanPicture">
<a href="http://aem.eng.ua.edu/"><img
src="http://upload.wikimedia.org/wikipedia/commons/8/84/Titan_in_natural_color_Cassini.jpg"
alt="http://upload.wikimedia.org/wikipedia/commons/4/4a/Titan_and_rings_PIA14909.jpg"
width="250" height="250"></a>
</div>

<div class="PlanetCalc"><strong style = "color: white">Titan</strong></div>

<div class="planetlinks"><strong style = "color:white">Other Atmosphere Calculators</strong>
<p class= "planetlinks" style="font-size: 24px; color: #000; text-shadow: 2px 2px 3px #b22e20">
<a href="http://aem.eng.ua.edu/"><font color="#FF2400">Mars</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#CC6600">Venus</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#2B65EC">Earth</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#00008B">Neptune</font></a>
</p>
</div>
</div>
```

Figure 3.12. Titan HTML code difference

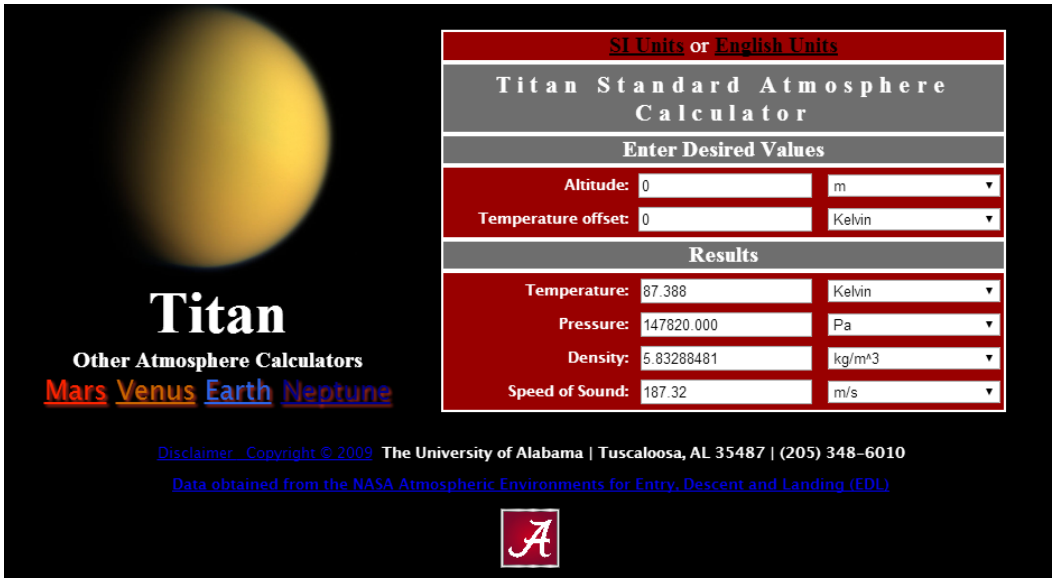

Figure 3.13. User interface and website layout of Titan

```
<div class="NeptunePicture">
 <a href="http://aem.eng.ua.edu/"><img
src="http://upload.wikimedia.org/wikipedia/commons/0/06/Neptune.jpg"
alt="http://upload.wikimedia.org/wikipedia/commons/8/8c/Neptune_storms.jpg"
width="259" height="250"></a>
</div>

<div class="PlanetCalc"><strong style = "color: white">Neptune</strong></div>

<div class="planetlinks"><strong style = "color:white">Other Atmosphere Calculators</strong>
<p class= "planetlinks" style="font-size: 24px; color: #000; text-shadow: 2px 2px 3px #b22e20">
<a href="http://aem.eng.ua.edu/"><font color="#2B65EC">Earth</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#CC6600">Venus</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#FDD017">Titan</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#FF2400">Mars</font></a>
 </p>
</div>
</div>
```

Figure 3.14. Neptune HTML code difference

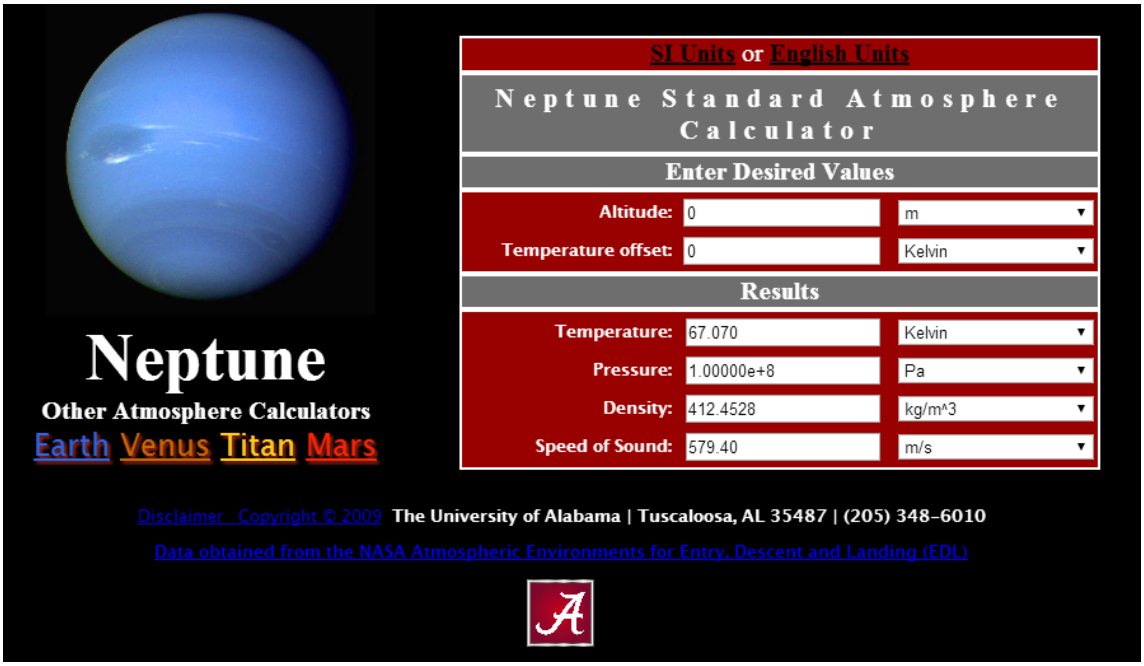

Figure 3.15. User interface and website layout of Neptune

As can be seen from the figures even the code that is different only has changes in the words themselves. Since they are different pages then the other planet calculator links must be different per planet. Besides these links the picture link to the appropriate picture is also different to display the correct planet/moon.

3.7 HTML Summary

A program has been developed to provide the online atmospheric calculator with the structure and information needed to perform correctly. The information for this code was based off of the University of Alabama websites and atmospheric data. This code was developed with the assistance of programming and website design sources cited in the reference section of this

thesis. The website, Codecademy, was especially helpful in the building of this code.

This program allows students to have access to all the information and calculation abilities of the website and links to other sites that may help them with further questions.

CHAPTER 4

DEVELOPMENT OF THE CSS CODE

4.1 Introduction to CSS

One of the key factors in a website's ability to be user friendly is how it is styled. A website can function without CSS and would not be completely void of style but having that code integrated with HTML allows for a more organized code and more options for the styling. Using CSS grants the designer access to a very wide range of tools. These tools not only provide the appearance of the website but control the set up entirely. This grants the site the ability to be both aesthetically pleasing and displayed in a manner that the user will not be confused by the choices.

4.2 Formatting in CSS

Formatting in CSS can be arranged by broad built in functions and also custom classes. CSS utilizes pre-defined classes for HTML manipulation and the ability for the constructor to define custom classes. In HTML, there are many different types of elements starting from the whole body of the code, the headlines, and down to single lines [36]. These are items that can be adjusted as groups in CSS shown in Figure 4.1.

```
body {
        background-repeat:no-repeat;
        background-attachment:fixed;
        background-position:center;
        background-size:auto;
        background-color:black;
        height: 100%;
        margin: auto;
        max-width: 960px;
        min-width: 750px;
        padding: 0;
      }

p, ul, td {
        font-family: "Lucida Sans",Lucida Sans Unicode,Verdana,Arial,sans-serif;
        font-size: 10pt;
        color:#333;
        text-align:right;
}
h1 {
        font-family:"Lucida Sans",Lucida Sans Unicode,Verdana,Arial,sans-serif;
        font-size: 16pt;
        font-weight: normal;
        text-align: left;
        margin-top: 0;
        padding-top: 10px;
        color:#333;
}
h2 {
        font-family: "Lucida Sans",Lucida Sans Unicode,Verdana,Arial,sans-serif;
        font-size: 12pt;
        font-weight: normal;
        text-align: left;
        margin-top: 0;
        padding-top: 10px;
        color:#333;
}
```

Figure 4.1. Pre-existing classes

Below each of these classes are a set of instructions for the code. They identify changes to the font, size, color, alignments, borders etc. When these classes are called they apply to any element that identifies as one of these classes. For instance the header class (h1) will apply the specified styles to all items in the HTML code that are that type of header. Having these pre-existing classes defined with styles creates a default style for any elements that do not contain a custom class. However, if the designer uses the custom class function for an element it will override the pre-existing class for that particular item.

4.3 Using Classes from HTML to Adjust Style

One way to custom style the HTML code is by using the class feature. This feature identifies parts of the HTML code as a certain class. Once identified the class can be adjusted so that the particular item can be changed individually. This type of code can be seen in Figure 4.2.

```
.overall {
    width: 1000pt;
    height:300pt;
    overflow: hidden;
}
.largegroup1 {
    width: 300pt;
    float:left;
    margin-top:0;
}
.largegroup2 {
    width: auto;
    float: left;
    margin-top:30px;
}
.EarthPicture {
    margin-left:75px;
    margin-right:auto;
    border: none;
    cursor:pointer;
}
```

Figure 4.2. Class identifiers

Here the class is identified with the .classname syntax and below this the custom design can be specified. Classes can be used on as many items that designer needs simply by linking them to the tag in HTML. Similar to other programming languages, these functions can have any name the designer wishes to use and typically a name that is a simple description works best. In

Figures 4.3 and 4.4 a demonstration of the effects of the custom class code on the HTML can be seen below.

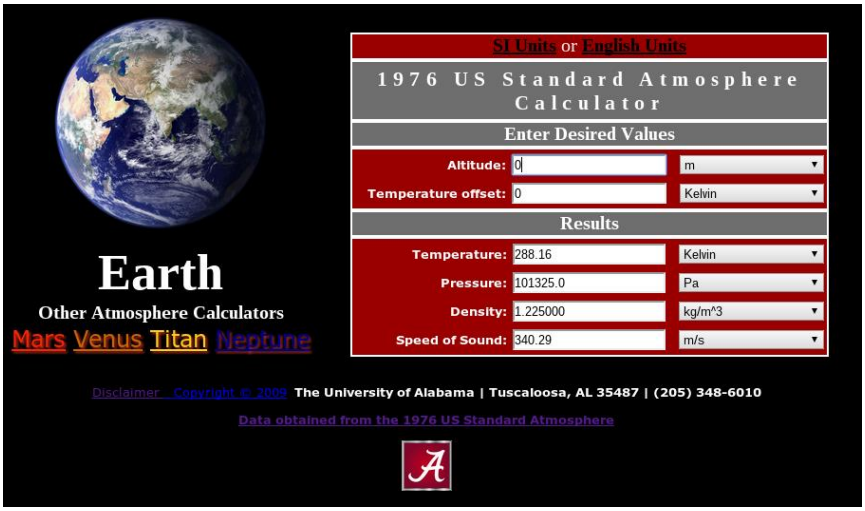

Figure 4.3. Custom class effects on HTML

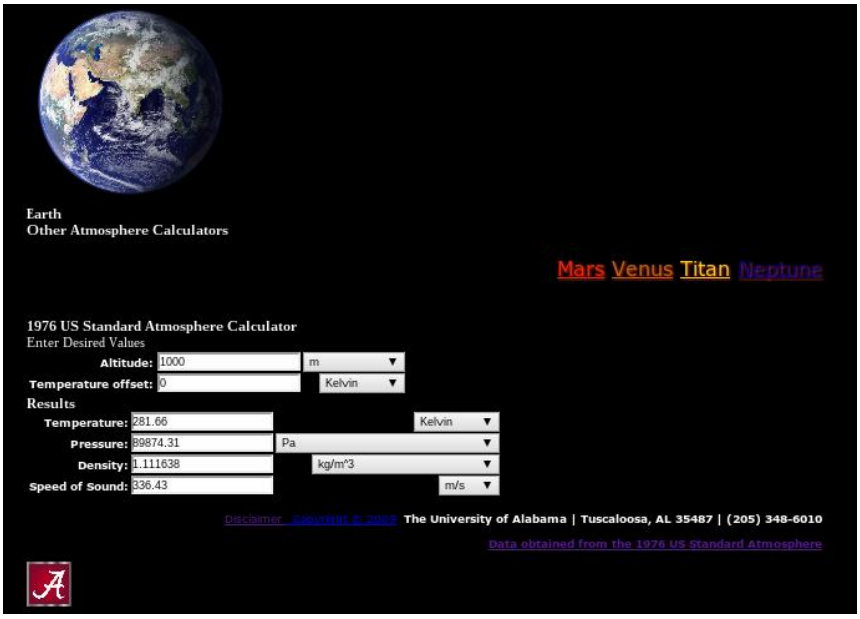

Figure 4.4. HTML without custom class effects

In these two figures the above had all classes in effect, both custom and default. Below the

custom classes were taken out but the default functions were left active. As can be seen, most of the original organization has stopped, leaving the webpage to be lackadaisical. Often times the default functions do not cover all the items in their description and many items that are of the same nature such as titles and pictures may need different constraints.

4.4 Creating a Website that Blends with University of Alabama Part II

The University of Alabama prefers its family of websites to blend in with each other. This includes having the appropriate colors and the proper links that belong to the University. In CSS the way to identify such styles the designer must use the hex color reference number. Colors all have an associated hex reference number that can be found with a simple google search. Figure 4.5 shows an example of applying these styles to the website.

```
Titlebar {
 width: 500px;
 height: 25px;
 margin-left:auto;
 margin-right:auto;
 background-color:#990000;
 border: 2px solid white;
 font-size: 15pt;
 text-align:center;
 color:white;
```

Figure 4.5. Hex color styling

In this block of code the style for background-color is identified as #990000 for the class which is the crimson used by the University websites. The University of Alabama also provides a copyright notice and a disclaimer at the bottom of their websites in order to provide the user with

this information if needed. Figure 4.6 shows a simple set of styling code that has made this information easy for the user to identify.

```
.uainfo {
  font-size: 10pt;
  text-align:center;
  margin-top:0;
}
```

Figure 4.6 UA copyright and disclaimer style code

4.5 CSS for Other Planetary Calculators

Similar to the HTML code, the CSS for the other atmospheric calculators is almost completely identical to the 1976 Standard Atmosphere calculator. Since the style and layout of the websites are to be similar then CSS must also have most of the same code. The only difference in these sets of code is the reference to their pictures at the top of the webpage. This code differences can be seen below in Figures 4.7, 4.8, 4.9, 4.10.

```
.MarsPicture {
  margin-left:75px;
  margin-right:auto;
  border: none;
  cursor:pointer;
}

.MarsPicture img{
  cursor:pointer;
   border: none;
}
```

Figure 4.7. Mars CSS code difference

```
.VenusPicture {
  margin-left:75px;
  margin-right:auto;
  border: none;
  cursor:pointer;
}

.VenusPicture img{
  cursor:pointer;
   border: none;
}
```

Figure 4.8. Venus CSS code difference

```
 .TitanPicture {
   margin-left:75px;
   margin-right:auto;
   border: none;
   cursor:pointer;
 }

 .TitanPicture img{
   cursor:pointer;
    border: none;
 }
```

Figure 4.9. Titan CSS code difference

```
.NeptunePicture {
  margin-left:75px;
  margin-right:auto;
  border: none;
  cursor:pointer;
}

.NeptunePicture img{
  cursor:pointer;
   border: none;
}
```

Figure 4.10. Neptune CSS code difference

As shown above the only difference is the name of the class that controls the planet picture. The code within this class however is still the same as the other planets' CSS.

4.6 CSS Summary

A style program has been developed to provide the online atmospheric calculator with organization and a visually stimulating blueprint. The styling for this code was based off of styles from other University of Alabama websites. This code was developed with the assistance of programming and website design sources cited in the reference section of this thesis. The website, Codecademy, was especially helpful in the building of this code. This program allows students to navigate and understand the workings of the website.

CHAPTER 5

DEVELOPMENT OF THE JAVASCRIPT CODE

5.1 Introduction to JavaScript in an Online Calculator

One of the key factors in a website's ability to do more than just display information is JavaScript. A website can function without JavaScript but would not be able to incorporate interactive functions and options. Using JavaScript grants the designer access to a very wide variety of interactions and abilities that websites did not have before its creation. These interactions range anywhere from a website that can perform calculations for the user or making objects on the website move around in different fashions.

Using JavaScript to add this feature to a website that would calculate standard atmosphere data would provide a great resource to students. This could reduce time spent on trying to look up the needed values in tables or online documents. It would also prevent students from having to interpolate between data points in a table if the number that they are looking for is not provided. The application of JavaScript for the creation of a standard atmosphere model is the topic of this chapter.

5.2 Calculation of the Atmosphere Values

Before the JavaScript code could be made the results needed to be calculated first in order to compare and use in the program. The altitude is the key value in determining what layer the calculations are in and is used in the calculations. Each layer of the altitude has a minimum

and maximum altitude value that are used in the equations. Depending upon the layer these values change and must be taken into account in the equations. These will be discussed in more detail in their respective sections of this chapter and a flowchart of this algorithm is demonstrated in Figure 5.1.

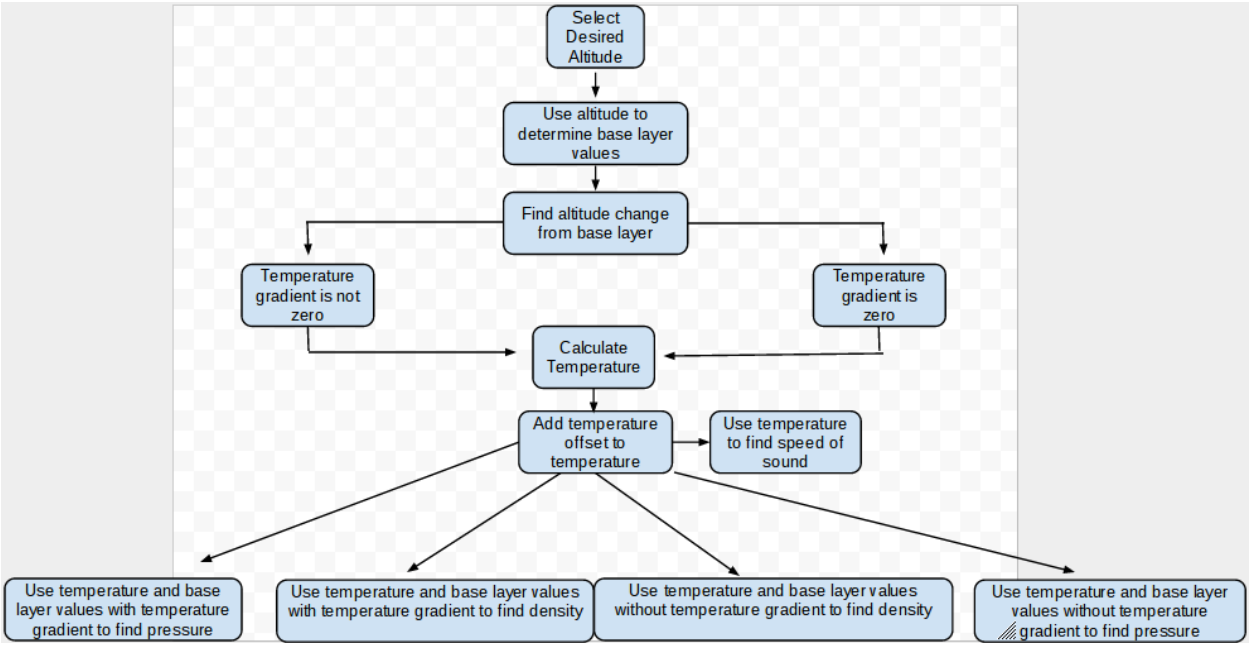

Figure 5.1. Flowchart of the algorithm used for calculating the atmosphere program.

Since each layer of the atmosphere needs base values then the first layer's base values are those found at sea level. Table 5.1 shows the initial values of the variables when at sea level [37]. These constants act as base values in the equations for the Troposphere.

Table 5.1. Sea Level Constants

Pressure (Pa)	Density (kg/m^3)	Temperature (K)	Gravity (m/s^2)
101325	1.225	288.16	9.807

Besides these base values at sea level there are another set of constants used throughout the

layers. In Table 5.2 these constants used in the equations throughout the standard atmosphere for air are shown below. Since the atmosphere is made of air then it can be assumed to follow the perfect gas law. These values of air are needed in order to satisfy all of the variables in the perfect gas law equation. [38]

Table 5.2. Properties of Air

Universal Gas Constant	Specific Heat Ratio	Molecular Weight	Specific Gas Constant
8.314	1.4	28.96	287.04

A standard atmosphere obeys the aero-hydrostatic equation as well as the perfect gas equation [39]. In the aero-hydrostatic equation the change in pressure is equal to gravity times the negative density and the change in geopotential altitude. This equation has the form:

$$dP = -\rho \cdot g \cdot dh \tag{1}$$

In order to account for the change in gravity as altitude changes the geopotential altitude is used instead of the geometric altitude. This height is a simplified representation of the height using a constant variable for gravity and height above the surface. The difference between geopotential altitude and geometric altitude is negligible in the standard atmosphere range used in this thesis. Lutze (2014).

In order to simplify further equations a hydrostatic variable is used that combines gravity, the molecular weight of air, and the gas constant:

$$gMR = \frac{g \cdot M}{R} \tag{2}$$

The perfect gas law is defined by this professor as pressure being equal to the density times the

perfect gas constant and the temperature. Lutze (2014). The perfect gas law has the form:

$$P = \rho \cdot R \cdot T \tag{3}$$

These equations are the governing equations used to calculate the density and pressure profiles of the atmospheres given a height and temperature.

5.2.1 Finding the Temperature Profile

The first objective is to obtain an equation that can represent the temperature at any layer of the atmosphere. In a paper by the National Oceanic and Atmospheric Administration (NOAA) they describe the temperature as a linear function of height where the temperature at the current height is equal to the temperature at the base of the atmosphere layer plus the lapse rate, or temperature gradient, times the difference in the current altitude and the base altitude [22]. This temperature profile equation has the form:

$$T(h) = T_1 + K \cdot (h - h_1) \tag{4}$$

This temperature equation represents all of the layers of the atmosphere up to 86 km. The lapse rate determines whether the temperature is increasing, decreasing, or constant in a layer based on its value as seen in Table 5.3.

Table 5.3. Lapse Rate by Layer. Lutze (2014).

Range (km)	Temperature Gradient (K/km)
0-11	-6.5
11-20	0
20-32	1
32-47	2.8
47-51	0
51-71	-2.8
71-84.852	-2

This equation is also used in collaboration with the other two governing equations to derive the pressure and density profiles.

5.2.2 Using the Temperature to Determine the Pressure and Density Profiles

Pressure and density profiles are derived into two separate equations depending on when the temperature is constant or when the lapse rate is constant. In the 1976 publication on the standard atmosphere, the NOAA combine the perfect gas law equation and the aero-hydrostatic equation in order to eliminate the density variable [22]:

$$\frac{dP}{P} = -\frac{g \cdot dh}{R \cdot T} \tag{5}$$

For a constant temperature layer, or where the temperature gradient is zero, the T in this new equation can be replaced with T_1. Due to the pressure and altitude terms being on opposite sides the first manipulation of the equation is integrating both sides of the equation. This creates a relationship between the ratio of the current pressure and the base layer pressure that is equal to the right hand side of equation decreasing exponentially:

$$\frac{P}{P_1} = e^{-\frac{g}{R \cdot T_1}(h-h_1)} \quad (6)$$

Then by the relationship between density and pressure using the ideal gas law the current density and the base layer density ratio is equal to the previously derived pressure ratio.

For a constant temperature gradient layer, or where the temperature changes linearly with altitude, the equation found previously for the temperature profile is inserted into where the T variable was in the new aero-hydrostatic equation. Unlike the previous equation, where there was only one temperature variable, this will take into account the effects of temperature as altitude changes. Using the same procedure as before both sides of the equation can be integrated since like terms are grouped together. Once integrated the equation is now in a form that shows a relationship between the change in pressure with the change in temperature:

$$\frac{P}{P_1} = \left(\frac{\mathrm{T(h)}}{\mathrm{T_1}}\right)^{\frac{-\mathrm{g}}{\mathrm{R \cdot K}}} \quad (7)$$

Just like before the density is related to the pressure through the ideal gas law. The density ratio between the current layer density and the bottom of the layer density is equal to the pressure ratio and the temperature ratio of that layer:

$$\frac{\rho}{\rho_1} = \left(\frac{\mathrm{P}}{\mathrm{P_1}}\right) \cdot \left(\frac{\mathrm{T_1}}{\mathrm{T(h)}}\right) \quad (8)$$

The four separate equations provide pressure and density profiles for a standard day up to 86 km.

5.2.3 Using the Temperature to Determine the Speed of Sound

Speed of sound has a direct relationship to temperature and therefore changes as

temperature changes. The speed of sound is calculated by taking the square root of the specific heat ratio, gas constant, and temperature as shown below [40]:

$$a = \sqrt{\gamma \cdot R \cdot T(h)} \tag{9}$$

5.3 Creation of the Standard Atmosphere Program

Once all of the necessary data was collected and calculated then the JavaScript program could be written. JavaScript, like most programming languages, work from top to bottom of the code. So the first step in creating a function that calculates the above equations correctly is declaring variables. The first set of variables declared are the global variables as seen in Figure 5.2.

```
// globals
var altitude;
var deltaT = 0;
var temperature;
var pressure;
var speedOfSound;
var density;
```

Figure 5.2. Global Variables.

These variables are called global variables because of their ability to be accessed anywhere in the code. This allows for JavaScript to access these variables in all functions and in the HTML code.

The next step in the code is declaring a function that when called will execute a series of calculations. This first function has its own set of variables and equations that are used to

calculate the temperature, pressure, density, and speed of sound in the atmosphere. A series of variables are declared inside the function that use the values from Tables 5.1 and 5.2 and retain values to be used in the atmospheric equations as seen in Figure 5.3.

```
//function that uses the atmospheric equations
function AtmosphereCalculatorEquations(){

  var i;                    //Will allow the appropriate value to be assigned in the equation
  var BottomLayerRelativePressure;
  var AltitudeChange;      //h-h1 where h1 is the altitude bottom of the layer
  var CurrentPressure;     //pressure variable/
  var BottomLayerTemp; //as sounds: temp at base of layer
  var tempGradient;        //Lapse rate [K/km]
  var BottomLayerDensity;

  //Default unchanging variables
  var temperatureatSeaLevel   = 288.16;    // Temperature at sea level [deg K]
  var densityatSeaLevel       = 1.225;     // Density at sea level [kg/m3]
  var pressureatSeaLevel      = 101325;    // Pressure at sea level [Pa]
  var R                       = 287.036;  //air gas constant [J/kg/K]
  var gamma                   = 1.4;
  var gravity                 = 9.80665;  // [m/s2]
```

Figure 5.3. Initial part of the equation calculator code where variables are declared.

Within functions it is usual to have checks and balances in order to make sure the correct information is inputted as shown in Figure 5.4. A switch statement was placed inside the function

```
// Input altitude check
switch (altitude,temperature,pressure,density,speedOfSound){
        case ((altitude < 0) || (altitude > 85000)):
        altitude          = 0;
        temperature       = 0;
        pressure          = 0;
        density           = 0;
        speedOfSound = 0;

        return "Alert: Altitude must in the range of 0 and 85000 meters.";
}
```

Figure 5.4. Input altitude check.

to ensure that the altitude selected by the user is within the ranges where these equations are still applicable. If the user accidentally tries to input an altitude value outside the correct ranges this code will inform the user of the correct values.

Throughout the code there will be several codes that are there to act as a check for different scenarios as they may arise. Another check like the above code was made to return the temperature offset value to zero if the user input a value that is not a number in Figure 5.5.

```
var GravityConstant = gravity * (1000 / R); //hydrostatic variable used to reduce
                                            //complexity of equations later on

// Makes deltaT return zero if input isn't a number
if (isNaN (deltaT)){
   switch (deltaT){
     default:
        return 0;
   }
}
 i = 0; //loop initial value
 // Finds the appropriate value in the array and updates the variable
 // altitude's value for the height chosen
 // while it is greater than the loop continues
 if (altitude > 0) {
   while (altitude > atmosphereLayers[i + 1]) {
     i = i + 1;
   }
 }
```

Figure 5.5. Defining the hydrostatic variable, temperature offset check, and array addition.

Here the hydrostatic variable is defined to allow the computer to simplify more complex equations. Also there is another if statement instructing the computer to assign appropriate values of "i" given an altitude depending upon which layer of the altitude. This if/while statement allows the use of the "i" variable defined earlier to gather information from arrays shown in Figure 5.6 and store these values into the variables as seen in Figure 5.7.

```
//Array to designate value at each atmospheric layer to be used as a base value in equations.
//Retrieved from NASA atmosphere tables
var atmosphereLayers              = new Array(0, 11000, 20000, 32000, 47000,51000, 71000,
                                              84852);
var PressureRatioatLayers         = new Array(1, 2.23355570684799e-1, 5.4030803694344e-2,
                                        8.56615261842e-3, 1.094455932173e-3,
                                        6.60565837016e-4, 3.9041059064e-5,
                                        3.684406018e-6);
var TemperatureatLayers           = new Array(288.16, 216.66, 216.66, 228.66, 270.66, 270.66,
                                         214.66, 186.956);
var TemperatureGradientsatLayers = new Array(-6.5, 0, 1, 2.8, 0, -2.8, -2, 0);
var DensityatLayers               = new Array(1.225, 3.63905e-1, 8.803e-2, 1.3224e-2,
                                              1.427e-3, 8.62e-4, 6.4e-5, 7e-6);
```

Figure 5.6. Atmosphere layer values stored as arrays.

```
//Using appropriate i from selection inputs correct value from array into these variables for
//equations
  BottomLayerTemp              = TemperatureatLayers[i];
  tempGradient                 = TemperatureGradientsatLayers[i] / 1000;
  BottomLayerRelativePressure  = PressureRatioatLayers[i];
  BottomLayerDensity           = DensityatLayers[i];
  AltitudeChange               = altitude - atmosphereLayers[i];
  temperature                  = BottomLayerTemp + tempGradient *AltitudeChange;
```

Figure 5.7. Array values stored inside of variables

Those variables that were defined above now have the ability to accept a given value based off of

which layer is needed. This is also how the temperature profile equation is defined for use in further equations.

The last part of the atmosphere calculator function is the profile equations themselves. In order for JavaScript to calculate them correctly if/else statements are used to determine which set of equations need to be used. Both the density and pressure equations change depending on the value of the lapse rate and are activated by this condition. In the code, the computer looks for a scenario when the lapse rate is extremely small, or zero in this case, so that it does not get trapped in an infinite loop. Figure 5.8 shows the code for switching between these equations and the final calculations for temperature, and speed of sound.

```
// Pressure Equation at Selected Altitude for when temperature gradient is "0" otherwise the
//equation shifts using different variables
 if (Math.abs(tempGradient) <  0.0000000001) {
   CurrentPressure = BottomLayerRelativePressure * Math.exp(-GravityConstant *
   AltitudeChange/1000 / BottomLayerTemp);
   }
   else {
   CurrentPressure =  BottomLayerRelativePressure*Math.pow(temperature/BottomLayerTemp,
                      -GravityConstant/tempGradient/1000);
      }
pressure = CurrentPressure * pressureatSeaLevel;
 // Equation for speed of sound at given temperature
 speedOfSound = Math.sqrt(gamma * R * temperature);

 if (Math.abs(tempGradient) <  0.0000000001){
 density = BottomLayerDensity * Math.exp(-GravityConstant * AltitudeChange/1000/
BottomLayerTemp);
  }
  else {
   density= BottomLayerDensity *
Math.pow(temperature/BottomLayerTemp,-((GravityConstant/tempGradient/1000) + 1));
      }
 temperature  = temperature + deltaT;
}
```

Figure 5.8. Pressure, Density, Speed of Sound, and Temperature equations.

The designer can insert words preceded by two backslashes in order for the code to be more easily interpreted. These are comments to help out the coder identify what the code is designed to do so that debugging it later is easier.

5.4 Variable Manipulation Code

Once the atmosphere program was established the other functions of the calculator had to be developed. Functions were made to provide the calculator with the ability to change units individually and all at once. The temperature and temperature offset function requires two inputs that when activated goes through a series of switch statements. Depending upon which unit the user selects the function will begin the unit conversion of the number as shown in Figure 5.9.

```
function SITemperatureConversion(ResultsTemperature, ResultsTemperatureUnit){

   switch(ResultsTemperatureUnit){
        case "Celsius": return ResultsTemperature - 273.15;
        case "Fahrenheit": return 9/5 * ResultsTemperature - 459.67;
        case "Kelvin": return ResultsTemperature;
        case "Rankine": return 9/5 * ResultsTemperature;
        case "Réaumure": return 4/5 * (ResultsTemperature - 273.15);
        default:alert ("Sorry"+ResultsTemperatureUnit+" unknown");
 }
}
function SIDeltaTConversion(ResultsTemperature, ResultsTemperatureUnit){

   switch (ResultsTemperatureUnit){
        case "Kelvin": return ResultsTemperature;
        case "Celsius": return ResultsTemperature + 273.15;
        case "Rankine": return 5/9 * ResultsTemperature;
        case "Fahrenheit": return (5/9*(ResultsTemperature- 32)) + 273.15;
        case "Réaumure": return 5/4 * ResultsTemperature;
        default: alert ("Sorry "+ResultsTemperatureUnit+" unknown");
 }
}
```

Figure 5.9. Temperature and temperature offset converter.

It can be seen that if an unrecognizable selection is made then JavaScript returns an alert message to the user.

In order for all of these functions discussed previously to work there must be a function that uses all of the information calculated by the others and disperses the information appropriately. The calculation function pulls information from the HTML using the document.forms and parse.float functions and sends this information through another series of switch statements as seen in Figure 5.10 [41].

```
function calculationFunction (){

   var RandomVariable;
   var x = document.forms.formID;
         deltaT = parseFloat(x.DeltaTAdjustment.value);
         deltaT = SIDeltaTConversion(deltaT, x.DeltaTUnitID.value);
         x.AltitudeSelector = x.AltitudeUnitID.selectedIndex;
         x.AltitudeSelectorUnit  = x.AltitudeUnitID.options[x.AltitudeSelector].text;
   var SetAltitudeSelectorUnitToSI = x.AltitudeUnitID.value;
   altitude = x.AltitudeAdjustment.value*SetAltitudeSelectorUnitToSI;
   var ThisIsTheLowestIWillGo= 0 / SetAltitudeSelectorUnitToSI;
   var ThisIsTheHighestIWillGo= 85000 / SetAltitudeSelectorUnitToSI;

    if ((altitude < 0) || (altitude > 85000)){
         switch (RandomVariable){
         default:RandomVariable = "Altitude must be at least " +
ThisIsTheLowestIWillGo.toFixed(0) + " and no higher than " +
ThisIsTheHighestIWillGo.toFixed(0) + " " + x.AltitudeSelectorUnit + ".";
                                        }
}else{
   RandomVariable = AtmosphereCalculatorEquations();
}
  displayIt(RandomVariable);
}
```

Figure 5.10. Calculator Function

In this code the document.forms function pulls the form data from the HTML files and inputs them into variables. A similar process occurs with the parse.float function in that it pulls the

information and saves it as a number within the designated variable. Once this information is loaded then JavaScript begins the if/else statements to determine if the altitude is invalid then it will activate the switch statement that opens an error message with the correct range in the selected unit, and lastly if the data is correct it will display the atmospheric calculations. This code only had to be changed slightly for the other planet calculators by changing the range of the altitude allowance.

The last functions that are used to change the values of selection are the SI and English unit selectors. These functions are activated when their button is clicked on the website. Using the document.forms function previously discussed it automatically populates the fields with a predetermined selection. Once complete, JavaScript then activates the two functions to carry out the calculations and to save the information. This code can be seen in Figure 5.11.

```
function TheEnglishUnits() {
 var x = document.forms.formID;
        x.AltitudeUnitID.selectedIndex          = 0;
        x.TemperatureUnitID.selectedIndex    = 2;
        x.DeltaTUnitID.selectedIndex          = 2;
        x.PressureUnitID.selectedIndex        = 5;
        x.DensityUnitID.selectedIndex          = 2;
        x.SpeedofSoundUnitID.selectedIndex  = 0;

 saveTheSelection();
 calculationFunction();
}

function TheUnitsTheStandardWorldPick() {
 var x = document.forms.formID;
        x.AltitudeUnitID.selectedIndex          = 2;
        x.TemperatureUnitID.selectedIndex    = 1;
        x.DeltaTUnitID.selectedIndex          = 0;
        x.PressureUnitID.selectedIndex        = 3;
        x.DensityUnitID.selectedIndex          = 0;
        x.SpeedofSoundUnitID.selectedIndex  = 3;

 saveTheSelection();
 calculationFunction();
}
```

Figure 5.11. SI and English unit selectors

5.5 Code Control Functions

A number of functions were created with the purpose of controlling the flow of information. One set of functions are used to locally store a set of parameters using the function localstorage.getitem and localstorage.setitem. Whenever these functions are called they will load the item using the built in function and save it locally. These functions are illustrated in Figure 5.12.

```
function infoLoader(RandomInput){
  return localStorage.getItem(RandomInput);
}

function DefaultLoaderCoder(RandomInput, ReturnThisZero){
  var selected = infoLoader(RandomInput);
          if (selected !== null) {
                  return selected;
          }
          else {
                  return ReturnThisZero;
                }
}

function infoSaver(RandomInput, saveStuff) {
  localStorage.setItem(RandomInput, saveStuff);
}
```

Figure 5.12. Loading and saving information locally

Here another function, DefaultLoaderCoder, utilizes the infoloader function to store an input into a variable and check if it is defined else it returns a different value that is defined as zero.

Another set of functions were designed to retrieve information and store it into a variable followed by a function used to save that data demonstrated in Figure 5.13. These functions help save and load information into the functions that connect to the HTML code.

```
function RetrieveFunction(itemRetrieved) {
  var selected = infoLoader(itemRetrieved.name);
  if (selected !== null) {
        itemRetrieved.selectedIndex = selected;
  }
}

function SaveFunction(itemRetrieved) {
  infoSaver(itemRetrieved.name, itemRetrieved.selectedIndex);
}
```

Figure 5.13. Functions that retrieve and save input information

Another function will then use these functions to retrieve appropriate information when the webpage is loaded. Since these functions are activated upon loading they automatically load and store this information into the correct elements for evaluation. These functions can be seen in Figure 5.14.

```
function loadTheSelection(){
 var x = document.forms.formID;
        x.a=RetrieveFunction.AltitudeUnitID;
        x.b=RetrieveFunction.TemperatureUnitID;
        x.c=RetrieveFunction.DeltaTUnitID;
        x.d=RetrieveFunction.PressureUnitID;
        x.e=RetrieveFunction.DensityUnitID;
        x.f=RetrieveFunction.SpeedofSoundUnitID;

 x.AltitudeAdjustment.value = DefaultLoaderCoder('AltitudeAdjustment', 0);
 calculationFunction();
}

function saveTheSelection() {
  var x = document.forms.formID;
        x.a = SaveFunction.AltitudeUnitID;
        x.b = SaveFunction.TemperatureUnitID;
        x.c = SaveFunction.DeltaTUnitID;
        x.d = SaveFunction.PressureUnitID;
        x.e = SaveFunction.DensityUnitID;
        x.f = SaveFunction.SpeedofSoundUnitID;

infoSaver('AltitudeAdjustment', x.AltitudeAdjustment.value);
}
```

Figure 5.14. Functions for loading and saving elements in the program

Once activated these functions also activate the calculate function to begin the calculations. These functions store the current input into the declared variables and load them when the user loads the page. These functions are used in the above code from Figure 5.11 to display the selected values.

Finally there are functions whose purpose is to activate when there is an error. These functions make use of the document.getElementbyId function that retrieves an item with the given identification and inputs into a variable. One uses this to display the error when called and the other function suppresses it by hiding the display. This utilize the built in function style.display to either display the error or display nothing referenced from the company, Refsnes Data (1997). These functions are shown in Figure 5.15.

```
function displayWhatsWrong(lnputthisNumber) {
  var RandomVariable = document.getElementByld("error");

  RandomVariable.innerHTML = lnputthisNumber;
  RandomVariable.style.display = 'block';
}

function getRidofit() {
  var RandomVariable = document.getElementByld("error");

  RandomVariable.innerHTML = "";
  RandomVariable.style.display = 'none';
}
```

Figure 5.15. Display and hide error functions

The last function calls these two functions using an if/else statement. This function is designed to display nothing if activated by an error else it will hide this error and populate the fields with

the values of whatever function activated it at the specified precision. This function is displayed in Figure 5.16.

```
function displayIt(InputthisNumber) {
  var adjustmentVariable;
  var x = document.forms.formID;
      if (InputthisNumber){
      displayWhatsWrong(InputthisNumber);
         x.TemperatureAdjustment.value    = ' ';
         x.PressureAdjustment.value       = ' ';
         x.DensityAdjustment.value        = ' ';
         x.SpeedOfSoundAdjustment.value = ' ';

      } else{
     getRidofit();
     adjustmentVariable = SITemperatureConversion(temperature,x.TemperatureUnitID.value);
     x.TemperatureAdjustment.value = adjustmentVariable.toPrecision(5);
     adjustmentVariable = pressure / x.PressureUnitID.value;
     x.PressureAdjustment.value = adjustmentVariable.toPrecision(7);
     adjustmentVariable = density / x.DensityUnitID.value;
     x.DensityAdjustment.value = adjustmentVariable.toPrecision(7);
     adjustmentVariable = speedOfSound / x.SpeedofSoundUnitID.value;
     x.SpeedOfSoundAdjustment.value = adjustmentVariable.toPrecision(5);
  }
}
```

Figure 5.16. Code for displaying variable adjustments

This function not only populates the results with blank values in case of an error but also adjusts how many significant digits the results have upon their calculation.

5.6 JavaScript Changes for other Atmospheric Calculators

Just as it was with the Earth Standard Atmosphere there are surface constants used to act as the initial values. Table 5.4 shows the surface value constants for each atmosphere and their respective acceleration due to gravity.

Table 5.4. Surface values and acceleration due to gravity for each atmosphere

	Pressure (Pa)	Density (kg/m^3)	Temperature (K)	Acceleration Due to Gravity (m/s^2)
Mars, Coffrey (2011) & Justus, et al (2001)	557.4	0.01391	212.06	3.716
Venus, Robbins (2006) & Cain (2012)	9210000	66.328	735	8.87
Titan, Coustenis, et al (2008) & Piazza (2009)	147820	5.833	87.39	1.35
Neptune, Burdick (2014) & Williams (2013)	100000000	412.45	67.07	11.15

Along with these values each atmosphere has a unique specific heat ratio and specific gas constant depending upon what gases occupy that space. In Table 5.5 below these authors also provide the specific gas constant and specific heat ratio values for each atmosphere.

Table 5.5. Specific heat ratio and specific gas constant for each atmosphere

	Specific Gas Constant	Specific Heat Ratio
Mars. Coffrey (2011) & Justus, et al (2001)	188.9	1.29
Venus . Robbins (2006) & Cain (2012)	188.9	1.29
Titan. Coustenis, et al (2008) & Piazza (2009)	290	1.38
Neptune. Burdick (2014) & Williams (2013)	3614.9	1.38

Using these values and equations derived from temperature profiles the temperature, pressure, density, and speed of sound were able to be calculated for each atmosphere.

As with the HTML and CSS, the majority of the JavaScript code could remain the same as the 1976 Standard Atmosphere calculator. The portion of the code that changed involved the

equations used to calculate the atmospheric data. Provided with an unique set of equations, with data based off of graphs discussed in chapter 2, it has allowed the other calculators to perform the necessary math. Figures 5.17, 5.18, 5.19, 5.20, 5.21, 5.22, 5.23, 5.24, and 5.25 display the differences for each planet in the atmosphere function.

```
function AtmosphereCalculatorEquations(){
  var temperatureatSurface    = 212.065;   // Temperature at surface [deg K]
  var densityatSurface        = 0.0189;    // Density at surface [kg/m3]
  var pressureatSurface       = 557.4;     // Pressure at surface [Pa]
  var R                       = 188.92;  //air gas constant [J/kg/K]
  var gravity                 = 3.716;   //m/s^2
  var gamma                   = 1.29;

 switch (altitude,temperature,pressure,density,speedOfSound){
  case ((altitude < 0) || (altitude > 200000)):
    altitude           = 0;
    temperature        = 0;
    pressure           = 0;
    density            = 0;
    speedOfSound = 0;
    return "Alert: Altitude must in the range of 0 and 200000 meters.";
  }

    if (isNaN (deltaT)){
    switch (deltaT){
      default:
          return 0;
    }
}
```

Figure 5.17. Mars JavaScript code atmosphere function

```
if (altitude <= 4000){
    temperature = 212.065;

} else if ((altitude > 4000) &&(altitude <= 50000)){
   temperature= ((altitude/1000) - 153.39)/(-0.7127);
} else if ((altitude > 50000) &&(altitude <= 73000)) {
    temperature = ((altitude/1000) - 343.9)/(-1.9749);
} else if ((altitude > 73000) &&(altitude <= 101000)) {
    temperature = 137.447;
} else if ((altitude > 101000) &&(altitude <= 130000)) {
    temperature = ((altitude/1000) + 15.673)/(0.8529);
} else if ((altitude > 130000) &&(altitude <= 140000)) {
    temperature = ((altitude/1000) - 109.44)/(0.1272);
} else if ((altitude > 140000) &&(altitude <= 153000)) {
    temperature = ((altitude/1000) - 80.249)/(0.2456);
} else if ((altitude > 153000) &&(altitude <= 170000)) {
    temperature = ((altitude/1000) + 2.399)/(0.5258);
} else{
    temperature = ((altitude/1000) + 448.41)/(1.8781);
}
```

Figure 5.18. Code for temperature calculation at different layers of Mars

```
function AtmosphereCalculatorEquations(){
 var temperatureatSurface   = 735;          // Temperature at surface [deg K]
 var densityatSurface       = 64.79;       // Density at surface [kg/m3]
 var pressureatSurface      = 9210000;  // Pressure at surface [Pa]
 var R                      = 188.92;    //air gas constant [J/kg/K]
 var gamma                  = 1.2857;
 var gravity                = 8.87;

switch (altitude,temperature,pressure,density,speedOfSound){
 case ((altitude < 0) || (altitude > 250000)):
  altitude     = 0;
  temperature  = 0;
  pressure     = 0;
  density      = 0;
  speedOfSound = 0;
  return "Alert: Altitude must in the range of 0 and 250000 meters.";
 }

 if (isNaN (deltaT)){
  switch (deltaT){
    default:
      return 0;
  }
}
```

Figure 5.19. Venus JavaScript code atmosphere function

```
if (altitude === 0){
    temperature = temperatureatSurface;

 } else if ((altitude > 0) &&(altitude <= 58000)){
   temperature= ((altitude/1000) - 93.671)/(-0.1287);
 } else if ((altitude > 58000) &&(altitude <= 92000)) {
    temperature = ((altitude/1000) - 143.38)/(-0.3214);
 } else if ((altitude > 92000) &&(altitude <= 98000)) {
    temperature = ((altitude/1000) + 3.1715)/(0.5916);
 } else if ((altitude > 98000) &&(altitude <= 102000)) {
    temperature = ((altitude/1000) - 221.07)/(-0.7068);
 } else if ((altitude > 102000) &&(altitude <= 146000)) {
    temperature = ((altitude/1000) + 61.359)/(1.1542);
 } else if ((altitude > 146000) &&(altitude <= 157000)) {
    temperature = ((altitude/1000) - 77.329)/(0.3792);
 } else if ((altitude > 157000) &&(altitude <= 172000)) {
    temperature = ((altitude/1000) + 66.91)/(1.0701);
 } else if ((altitude > 172000) &&(altitude <= 200000)) {
    temperature = ((altitude/1000) + 851.07)/(4.6092);
 } else {
    temperature = 228.1536;
 }
```

Figure 5.20. Code for temperature calculation at different layers of Venus

```
function AtmosphereCalculatorEquations(){

 var temperatureatSurface  = 87.388;     // Temperature at surface [deg K]
 var densityatSurface      = 5.832;      // Density at surface [kg/m3]
 var pressureatSurface     = 147820;   // Pressure at surface [Pa]
 var R                     = 290;       //gas constant [J/kg/K]
 var gamma                 = 1.3846;
 var gravity               = 1.35;

switch (altitude,temperature,pressure,density,speedOfSound){
 case ((altitude < 0) || (altitude > 200000)):
  altitude      = 0;
  temperature   = 0;
  pressure      = 0;
  density       = 0;
  speedOfSound = 0;

  return "Alert: Altitude must in the range of 0 and 200000 meters.";
 }
  if (isNaN (deltaT)){
       switch (deltaT){
              default:
              return 0;
  }
}
```

Figure 5.21. Titan JavaScript code atmosphere function

```
if (altitude === 0){
    temperature = temperatureatSurface;

 } else if ((altitude > 0) &&(altitude <= 26000)){
   temperature= ((altitude/1000) - 114.88)/(-1.2897);
 } else if ((altitude > 26000) &&(altitude <= 39000)) {
    temperature = ((altitude/1000) - 494.42)/(-6.6191);
 } else if ((altitude > 39000) &&(altitude <= 42000)) {
    temperature = 68.74;
 } else if ((altitude > 42000) &&(altitude <= 58000)) {
    temperature = ((altitude/1000) + 326.7461)/(5.3722);
 } else if ((altitude > 58000) &&(altitude <= 70000)) {
    temperature = ((altitude/1000) - 33.276)/(0.3572);
 } else if ((altitude > 70000) &&(altitude <= 85000)) {
    temperature = ((altitude/1000) - 7.3985)/(0.6007);
 } else if ((altitude > 85000) &&(altitude <= 107000)) {
    temperature = ((altitude/1000) + 120.99)/(1.5845);
 } else if ((altitude > 107000) &&(altitude <= 158000)) {
    temperature = ((altitude/1000) + 317.55)/(2.929);
 } else{
    temperature = ((altitude/1000) + 1299.1)/(8.9259);
 }
```

Figure 5.22. Code for temperature calculation at different layers of Titan

```
function AtmosphereCalculatorEquations(){

 var temperatureatSurface   = 67.07;          // Temperature at surface [deg K]
 var pressureatSurface      = 100000000;  // Pressure at surface [Pa]
 var R                      = 3614.91;      //air gas constant [J/kg/K]
 var gravity                = 11.15;        //m/s^2
 var gamma                  = 1.3846;

 switch (altitude,temperature,pressure,density,speedOfSound){
  case ((altitude < 0) || (altitude > 260000)):
   altitude           = 0;
   temperature        = 0;
   pressure           = 0;
   density            = 0;
   speedOfSound = 0;

   return "Alert: Altitude must in the range of 0 and 260000 meters.";
  }

   if (isNaN (deltaT)){
      switch (deltaT){
        default:
        return 0;
   }
}
```

Figure 5.23. Neptune JavaScript code atmosphere function

```
if (altitude === 0){
    temperature = temperatureatSurface;
} else if ((altitude > 0) &&(altitude <= 16000)){
   temperature= ((altitude/1000) - 85.212)/(-1.267);
} else if ((altitude > 16000) &&(altitude <= 26000)) {
    temperature = ((altitude/1000) - 159.44)/(-2.6283);
} else if ((altitude > 26000) &&(altitude <= 47000)) {
    temperature = 49.73;
} else if ((altitude > 47000) &&(altitude <= 71000)) {
    temperature = ((altitude/1000) + 53.317)/(2.1236);
} else if ((altitude > 71000) &&(altitude <= 85000)) {
    temperature = ((altitude/1000) - 23.077)/(0.8351);
} else if ((altitude > 85000) &&(altitude <= 108000)) {
    temperature = ((altitude/1000) - 31.004)/(0.687);
} else if ((altitude > 108000) &&(altitude <= 147000)) {
    temperature = ((altitude/1000) + 304.8)/(3.6911);
} else if ((altitude > 147000) &&(altitude <= 169000)) {
    temperature = ((altitude/1000) + 34.187)/(1.4867);
} else if ((altitude > 169000) &&(altitude <= 196000)) {
    temperature = ((altitude/1000) + 200.1)/(2.7028);
} else if ((altitude > 196000) &&(altitude <= 220000)) {
    temperature = ((altitude/1000) + 329.02)/(3.5917);
} else{
    temperature = 157.9525;
}
```

Figure 5.24. Code for temperature calculation at different layers of Neptune

The pressure, density, and speed of sound were acquired the same way for each atmosphere using the hydrostatic equation, perfect gas law, and speed of sound equation. Figure 5.25 displays the code used for these atmospheres.

```
  pressure=pressureatSurface*Math.exp(-(gravity/(R*temperature)*altitude));
// Equation for speed of sound at given temperature
  speedOfSound = Math.sqrt(gamma * R * temperature);

// standard density equation uses sea level values of temperature and density with pressure and
// temperature at current altitude to find density at that altitude
  density     = pressure/(R*temperature);

// Adjusts the temperature variable with the Temperature Offset selection
  temperature  = temperature + deltaT;
}
```

Figure 5.25. Pressure, density, and speed of sound code for each atmosphere

5.7 JavaScript Summary

Programs have been developed to provide online atmospheric calculators to assist students in their classes and extracurricular projects. The 1976 US Standard Atmosphere equations and their derivatives were written into this function along with those of the other atmospheres using curve fit software. This code was developed with the assistance of open source FORTRAN code provided by NASA for the development of the atmosphere functions provided by Carmichael, R (2013). The website, Codecademy, and several JavaScript reference books were also helpful resources in the building of this code. This program allows students to calculate the properties of these atmospheres at various altitudes from their computers.

CHAPTER 6

PUBLISHING AND IMPLEMENTATION OF WEBSITES

6.1 Introduction to Implementing the Websites

One of the principal requirements for an effective online calculator is that it is accessible. Factors that subscribe to the accessibility of this website include: knowledge of website location, web browser compatibility, individual knowledge of the data required. Besides accessibility there must also be classes and projects that require this type of knowledge for these websites to serve their purpose. This chapter discusses how the code developed in Chapters 3,4, and 5 can be implemented to the student body.

6.2 Implementation in Student Projects

At the University of Alabama there are a wide variety of student projects that involve the atmosphere. These projects and classes now have an easily accessible online calculator to help them find atmospheric data. These projects and classes include: Rocket Girls, BamaSAT, propulsion systems, and aerodynamics all of which involve knowledge of the atmospheric profiles.

Rocket Girls is a team of engineering students who compete in NASA's University Student Launch Initiative, USLI, against other university teams. This competition involves the designing and building of a rocket with a scientific payload to take measurements upon its launch. This experiment must be relevant research and approved by NASA. The appropriate

calculations in the atmosphere must be made in order for a successful launch and having an online calculator will provide the team with more time to focus on more difficult problems [42].

BamaSAT is a near-space platform project created for undergraduates who are interested in space. This project involves the design of a free-floating balloon satellite that can transport a payload to a near-space altitude. An altitude of 100 km is the Karman line and is internationally known as the boundary between earth and space. Near-space however is described as anywhere from 30-100 km and for the members of the BamaSAT team the platform is created to travel to 30 km [43]. The online calculator will be able to assist these students in calculations up to this point during the life of the project.

Lastly there are many classes across multiple disciplinary engineering fields that discuss areas involving the atmosphere. Classes such as Aerodynamics and Propulsion systems not only study properties of the atmosphere but also the effects of the atmosphere on aeronautical vehicles, flight, and other types of propulsion systems. In general these books will often times have a table of the 1976 US Standard Atmosphere in one of their appendices, but are usually provided in large altitude steps and few unit choices. These books also do not come with tables for other planet/moon atmospheres requiring students to spend too much research time trying to track down this information. Having an online calculator can assist these students with quick atmospheric calculations and provide them with more time to focus on the fundamentals of these classes.

6.3 Examples of Applications

Two examples are provided here to demonstrate the use of the Standard Atmosphere online calculator developed in this thesis. The first example shows a simple aerodynamics example that provides a set of given data and asks the student to find a variety of different variables. Consider the following example:

> Example 1: Estimate the pressure at an altitude of 6 kilometers in the Earth's atmosphere in both SI units and Imperial Units.

This first example can be easily solved using the online calculator. The first step is to input the given altitude into the online calculator as seen in Figure 6.1 for SI units and Figure 6.2 shows the same altitude input with the English units selected.

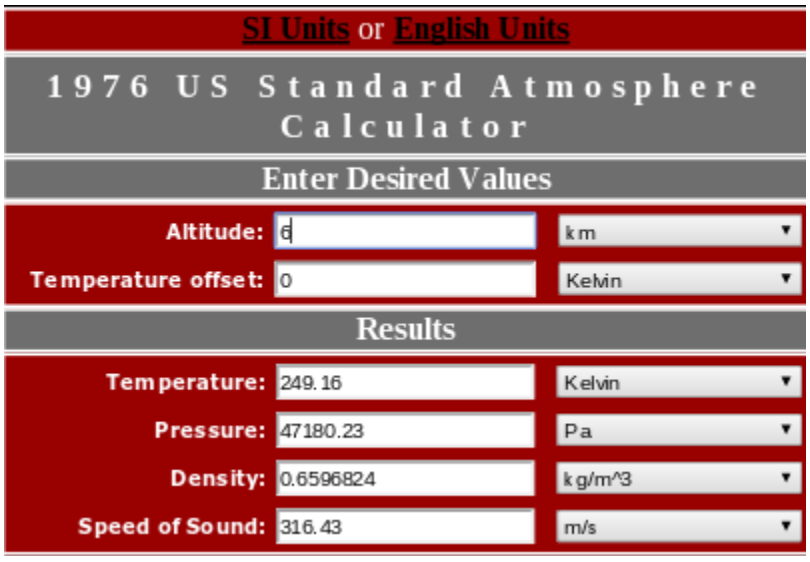

Figure 6.1. Example 1 input with answers in SI units

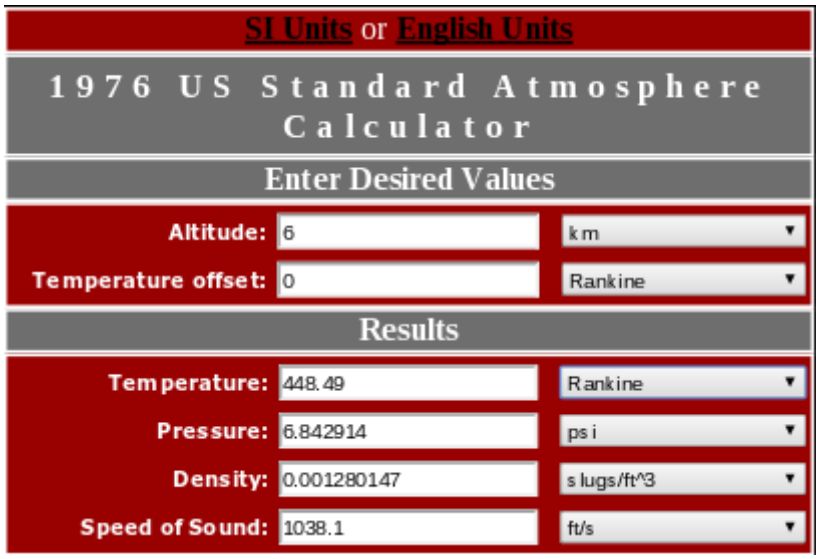

Figure 6.2. Example 1 input with answers in Imperial units

These provide the pressure as the problem asked for in both units required of the question.

While this example was relatively simple, it demonstrated just how easily the calculator obtained the data. In the field of propulsion there are much more complex problems that require the student to know the values of temperature and pressure at a given altitude in order to correctly work the problem. For a more complex example, consider:

> Example 2: Determine the maximum specific thrust with varying fan pressure ratio of a turbofan jet engine. Given that you are at a height of 13,000 meters on a standard day with a flight mach number of 0.8 and a heating value of 43,100 kj/kg assume average efficiencies.

This type of problem is a common question in the study of propulsion and requires the student to solve a very long list of equations and conditions. For times when the atmosphere pressure and temperature not given in the problem statement the calculator provides the student with a quick alternative looking up the values in a table. Figure 6.3 shows the results from the calculator input of an altitude of 13000 meters.

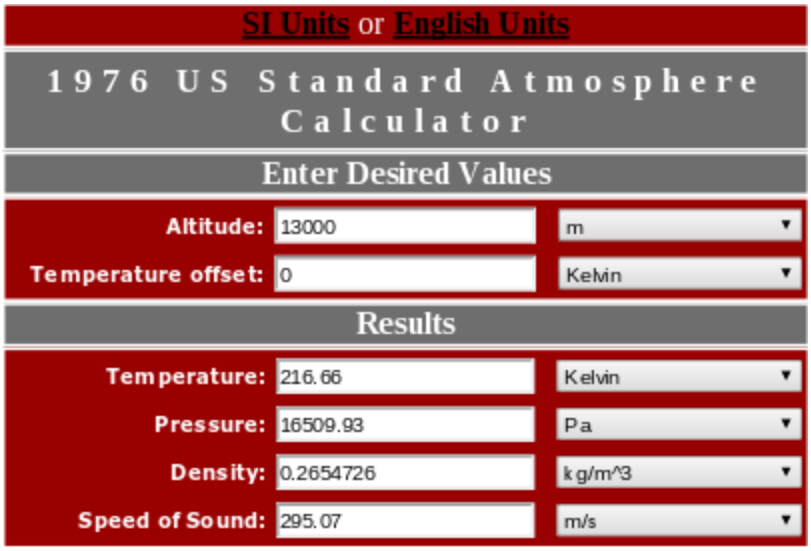

Figure 6.3. Propulsion example results from calculator

From here the student may then begin to use these values in the turbofan engine equations. These equations require the entrance pressure and temperature to be known in order to calculate temperatures and pressures within the engine itself.

6.4 Publishing Process

Publishing a website requires a domain name and a server to act as the host. The calculators are to be linked to the Aerospace Engineering and Mechanics website so that students would have no difficulty locating the websites. The domain name and server host will be provided by the University Of Alabama College Of Engineering. The code is saved in three different files for each website for the HTML, CSS, and JavaScript and will be located on the server.

6.5 Publishing and Implementation of Website Summary

In this chapter the applications and implementation of this online calculator was introduced. The examples shown in this chapter demonstrate the quick ability of this calculator to provide the student with the needed information. Also in this chapter a discussion on what type of projects and classes that this calculator can offer direct and useful assistances to those participating in those projects and classes. Finally the process for which the website was officially implemented and published onto the University of Alabama's selection of webpages. This online calculator allows the student to determine several atmospheric property values for multiple planetary bodies and it provides an aesthetically pleasing and easy to use problem solver.

CHAPTER 7

RESULTS AND CONCLUSION

7.1 Introduction to Results and Conclusion

To gauge the effectiveness of these online calculators discussed in the previous chapters the results of their calculations needed to be compared to pre-existing data. The comparison of these results was used to demonstrate the accuracy of these websites in order to be a viable asset for students now and in the upcoming years. This chapter explains these comparisons, future work that may evolve from this thesis, and lastly the conclusion.

7.2 Results of Calculator Output versus Atmospheric Tables

As stated in the introduction to this chapter the results are compared in order to gauge the accuracy of the calculators. The data being used to compare will range from NASA generated data, data calculated using spreadsheets, and the corresponding output data from each website. With this data the graphical representations can display the similarities and differences between the calculator output and data obtained from other sources.

As expected the data for the spreadsheet calculations and the calculator outputs were identical. This is due to the same equations being used in both scenarios and the JavaScript code working the way it was intended. Although each of these results originated from NASA, the sample output data from various NASA documents were not all identical to the results from the spreadsheet and online calculator outputs. This is mainly due to the many different types of

variations that occur in an atmosphere. The results for comparison were calculated at not only different times than the calculators but even different locations causing a slight variation between the data.

The first calculator for comparison is the 1976 US Standard Atmosphere. Outside data about this atmosphere varied much less between scholars and thus was easier to compare. In Figures 7.1, 7.2, and 7.3 the results from the NOAA paper on the 1976 US Standard Atmosphere vs. the website output for temperature, pressure, and density from Tables D.1, D.2, and D.3 are compared. NOAA (1975).

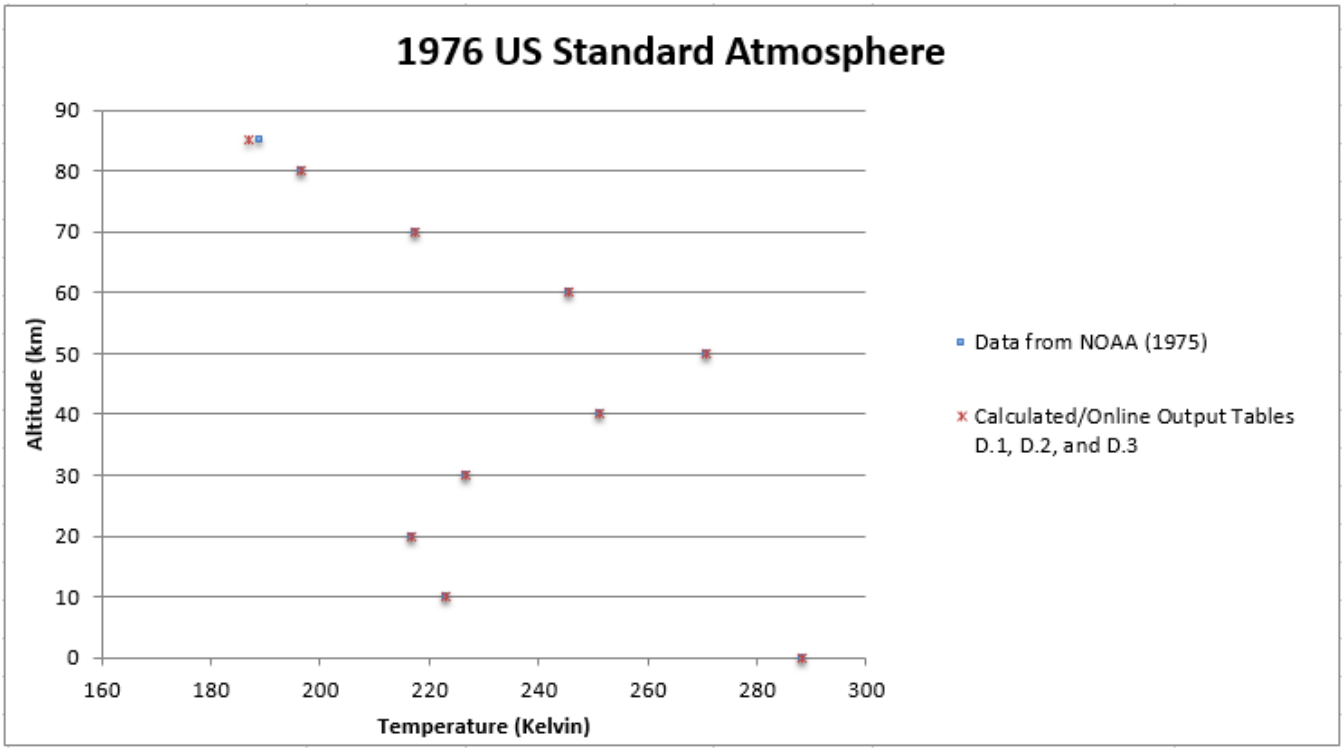

Figure 7.1. NOAA vs. calculator 1976 US Standard Atmosphere temperature profile

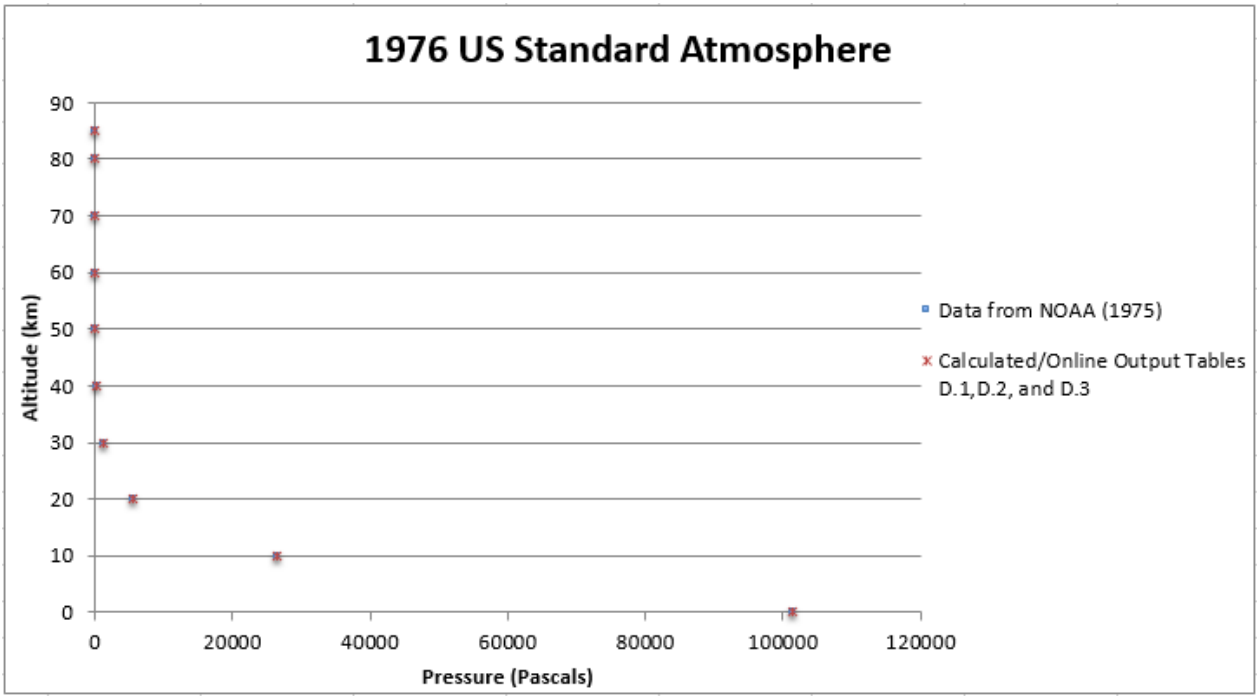

Figure 7.2. NOAA vs. calculator 1976 US Standard Atmosphere pressure profile

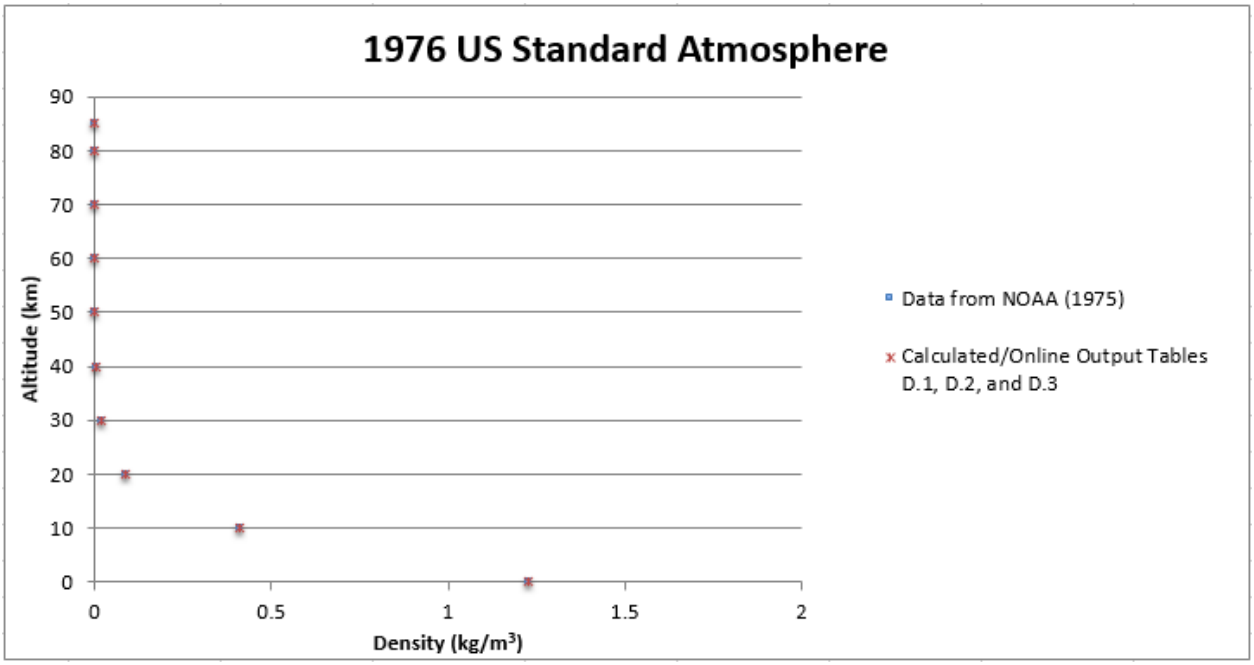

Figure 7.3. NOAA vs. calculator 1976 US Standard Atmosphere density profile

It can be seen that there are only very slight differences between the data points and can be noted that the 1976 standard atmosphere is an accepted benchmark. The difference is most likely due to the numbers being rounded off differently between the sources.

The Mars online calculator data was much harder to locate due to many of the files have different locations then what is referenced in various papers. The data compared was taken from the Mars GRAM user guide that provided sample output values from the Mars GRAM for user comparison. This data was based on a standard input for the FORTRAN code and was provided for comparison to personal results. Justus, et al. (2001). In Figures 7.4, 7.5, and 7.6 the results from this user guide vs. the website output for temperature, pressure, and density from Tables D.4 and D.5 are compared.

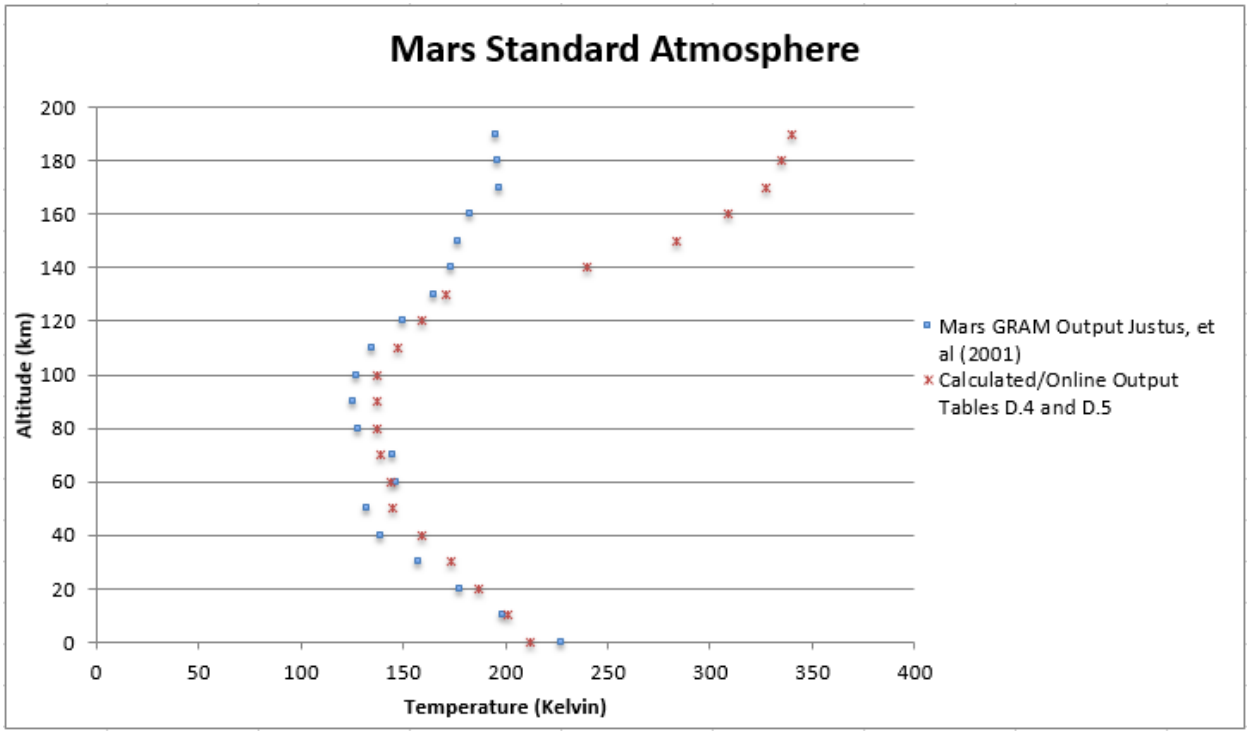

Figure 7.4. Mars GRAM output vs. calculator Mars atmosphere temperature profile

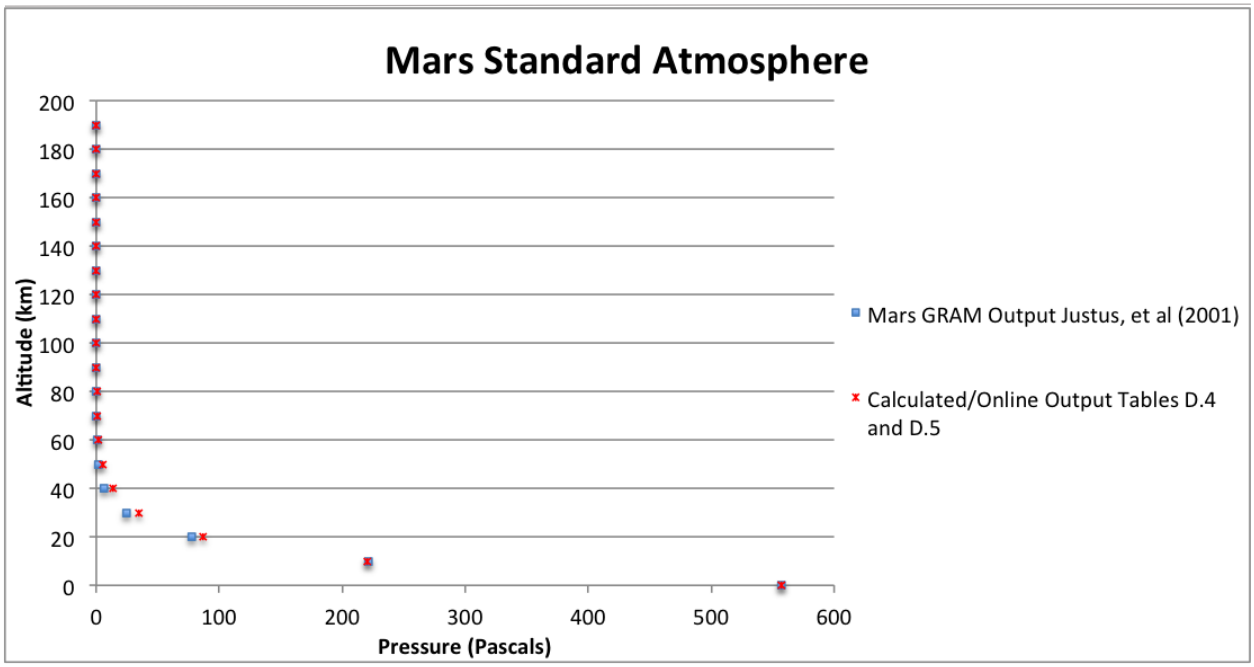

Figure 7.5. Mars GRAM output vs. calculator Mars atmosphere pressure profile

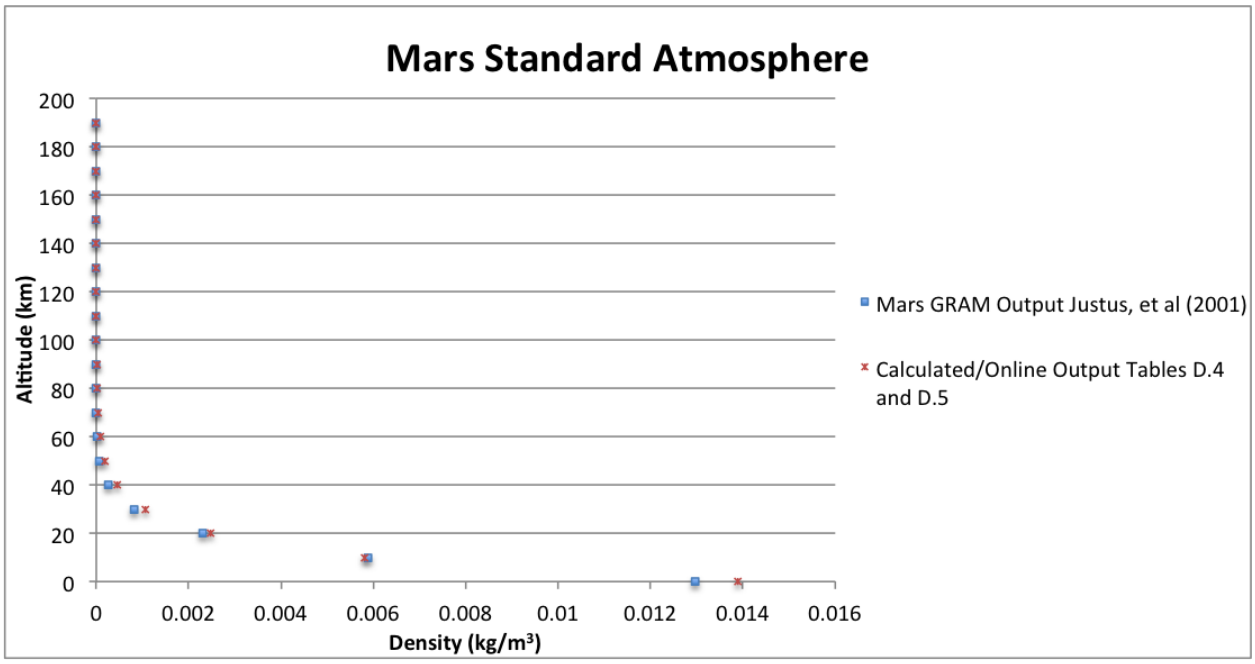

Figure 7.6. Mars GRAM output vs. calculator Mars atmosphere density profile

Like Earth's Atmosphere, Mars experiences seasonal changes and temperature variation due to location. The temperatures are highest near the equator and during the summer months and coldest around the north and south poles and during the winter months. Lewis (2003). The temperature profile for Mars varied with the comparison data more than the other atmospheres, but is still within the expected range. This was expected due to many of the sample outputs provided by NASA probes varied widely depending on location and seasonal effects.

The Venus comparison data was provided by a NASA Technical Manual (TM) for the atmospheric flight on Venus [44]. Like Mars, the information was unavailable in most locations with only graphs and few details regarding their origins. In order to obtain enough data a curve mapping technique was applied. In Figures 7.7, 7.8, and 7.9 the results from the TM vs. the online calculator outputs for temperature, pressure, and density from Tables D.8 and D.9 are compared. Landis, et al (2002).

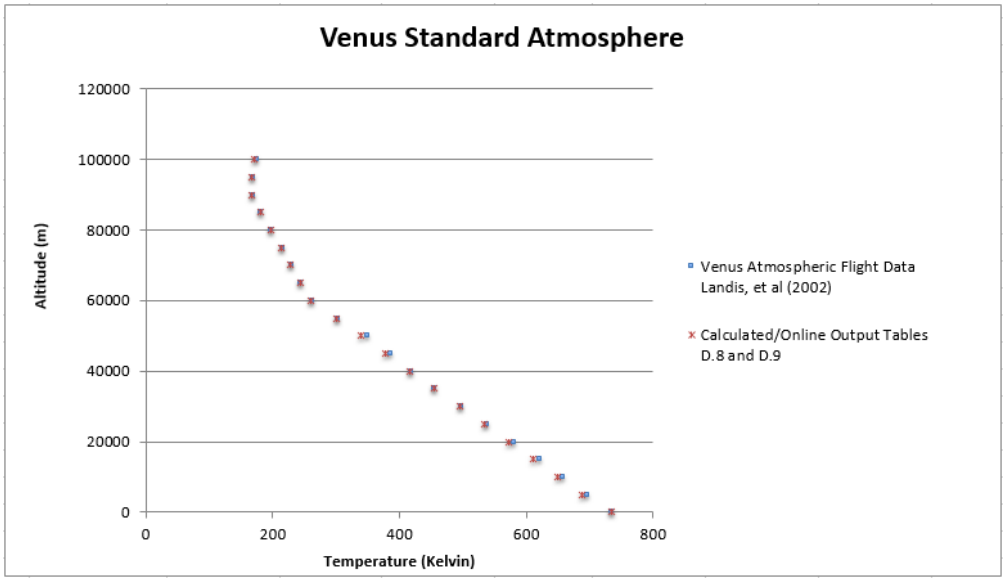

Figure 7.7. Venus TM data vs. calculator Venus atmosphere temperature profile

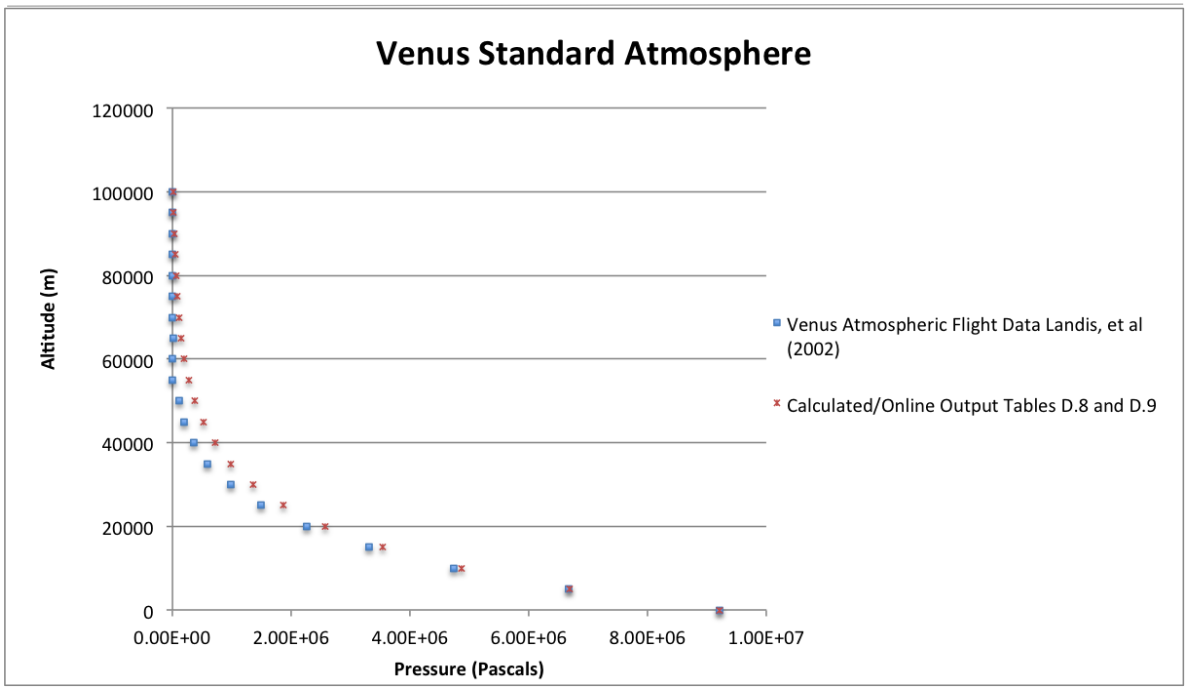

Figure 7.8. Venus TM data vs. calculator Venus atmosphere pressure profile

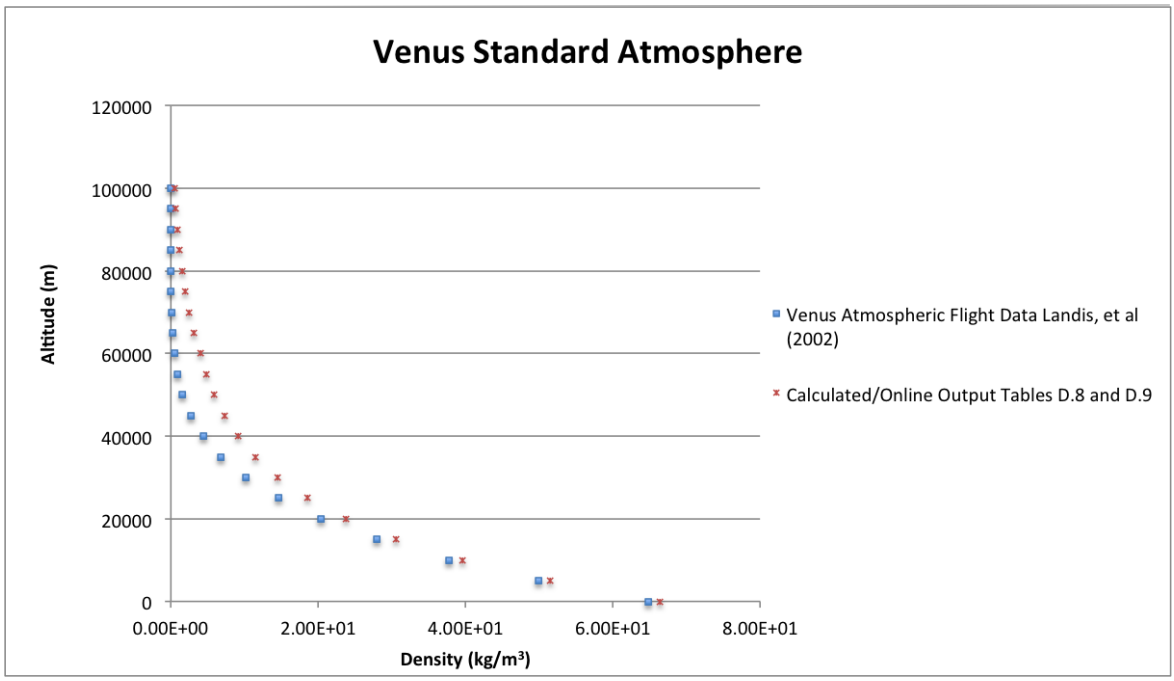

Figure 7.9. Venus TM data vs. calculator Venus atmosphere density profile

The data follows the same pathway with only slight variation as is expected due to variations in when and where the measurements were taken. These graphs help confirm the equations used in the calculator will calculate data that follows a similar trend as other data for Venus. Unlike Earth, Venus displays a more controlled decreasing path for most of the atmosphere below 100 kilometers without changing between increasing and decreasing temperatures.

The Titan data for comparison was provided by a report on General Circulation Models from the University of Oxford [45]. This report provided a very large amount of altitude points allowing more to be compared to the calculator. In Figures 7.10, 7.11, and 7.12 the GCM output vs. the online calculator outputs for temperature, pressure, density from Tables D.6 and D.7 are compared.

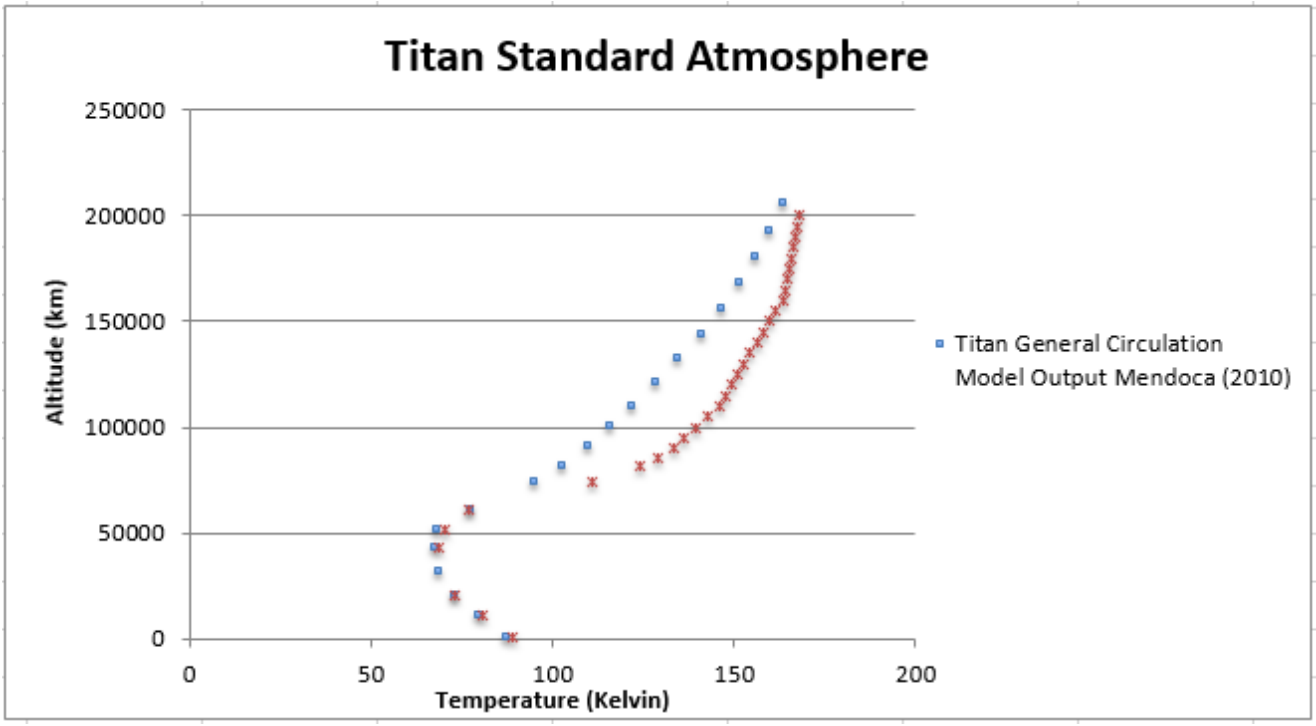

Figure 7.10. Titan GCM output vs. calculator Titan atmosphere temperature profile

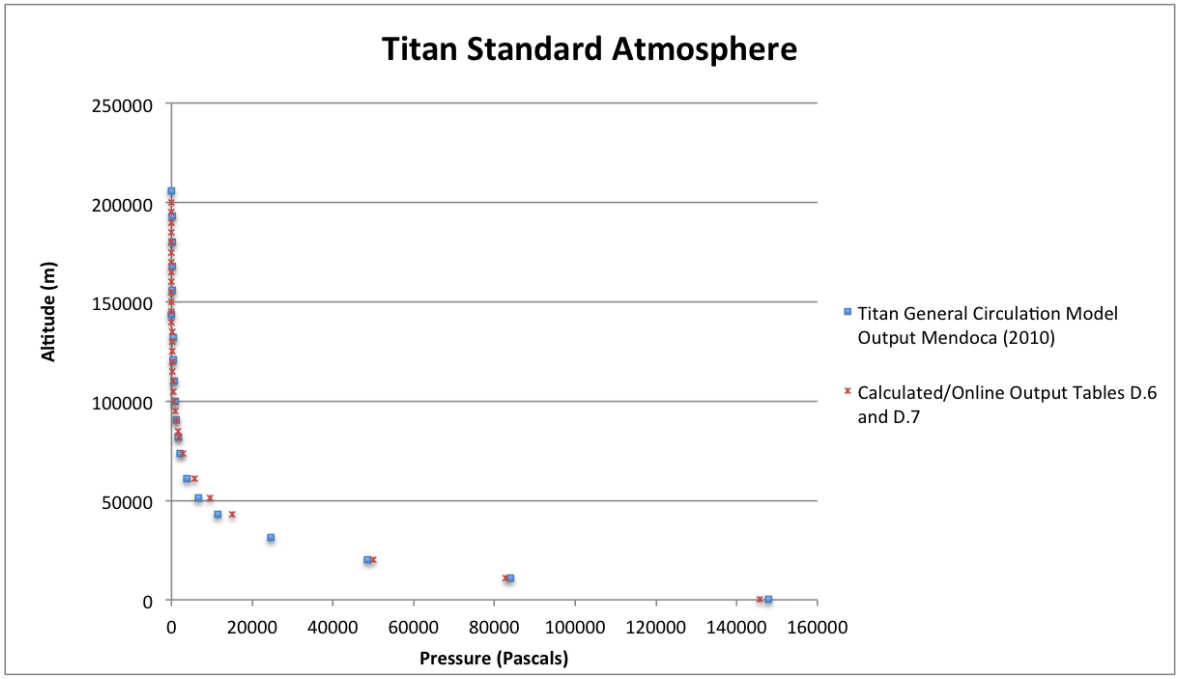

Figure 7.11. Titan GCM output vs. calculator Titan atmosphere pressure profile

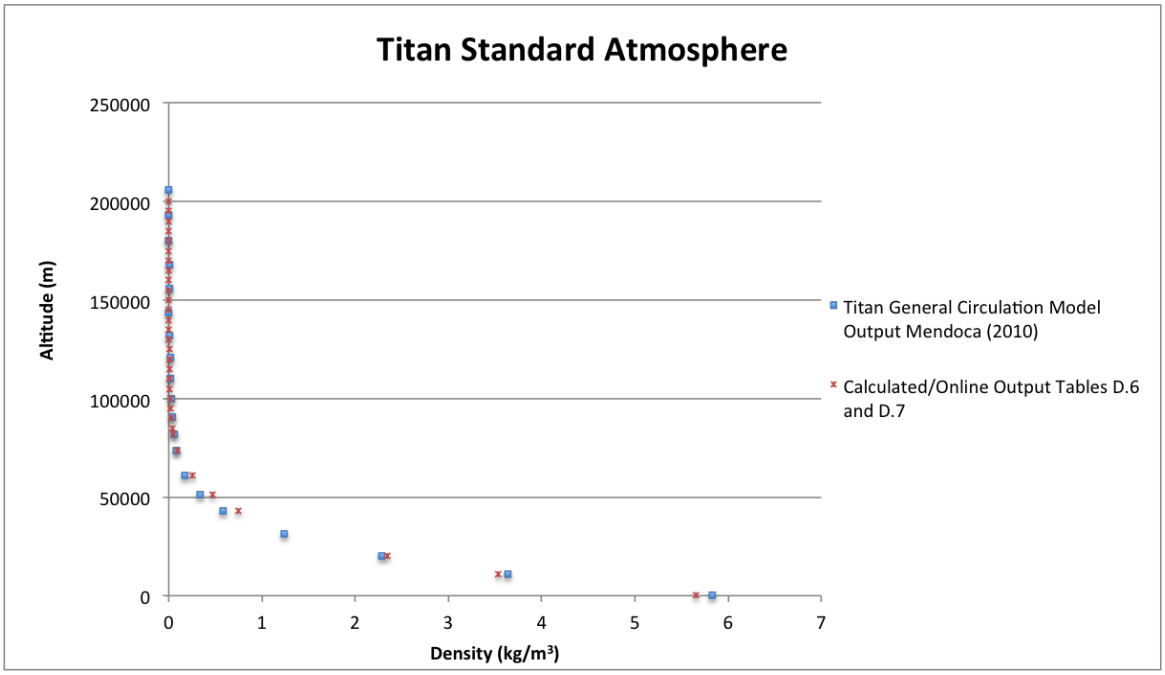

Figure 7.12. Titan GCM output vs. calculator Titan atmosphere density profile

The data follows the same pathway with only slight variation as is expected due to the measurements being taken at different times and locations. Like previous figures the pressure and density are based on standard equations. These calculations have produced the same trends in those profiles as with the circulation model. Titan's temperature starts similarly to other atmospheres with an initial decreasing trend but as the layer changes the temperature behavior also changes.

The last calculator, Neptune, data from Tables D.10 and D.11 will be compared to the graph provided by NASA on atmospheric entry, descent and landing. Justus et al. (2007). In Figures 7.13 and 7.14 a comparison of this data vs. the NASA graphs are shown below.

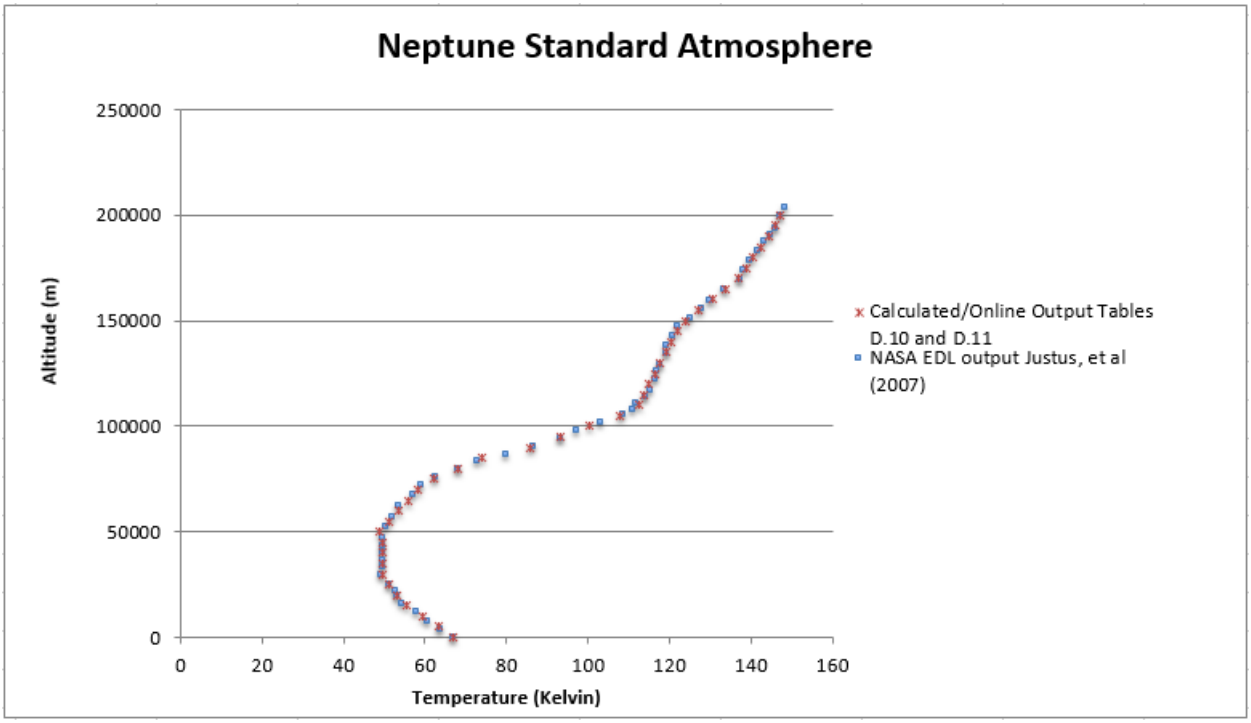

Figure 7.13. Neptune EDL vs. calculator atmosphere temperature profile

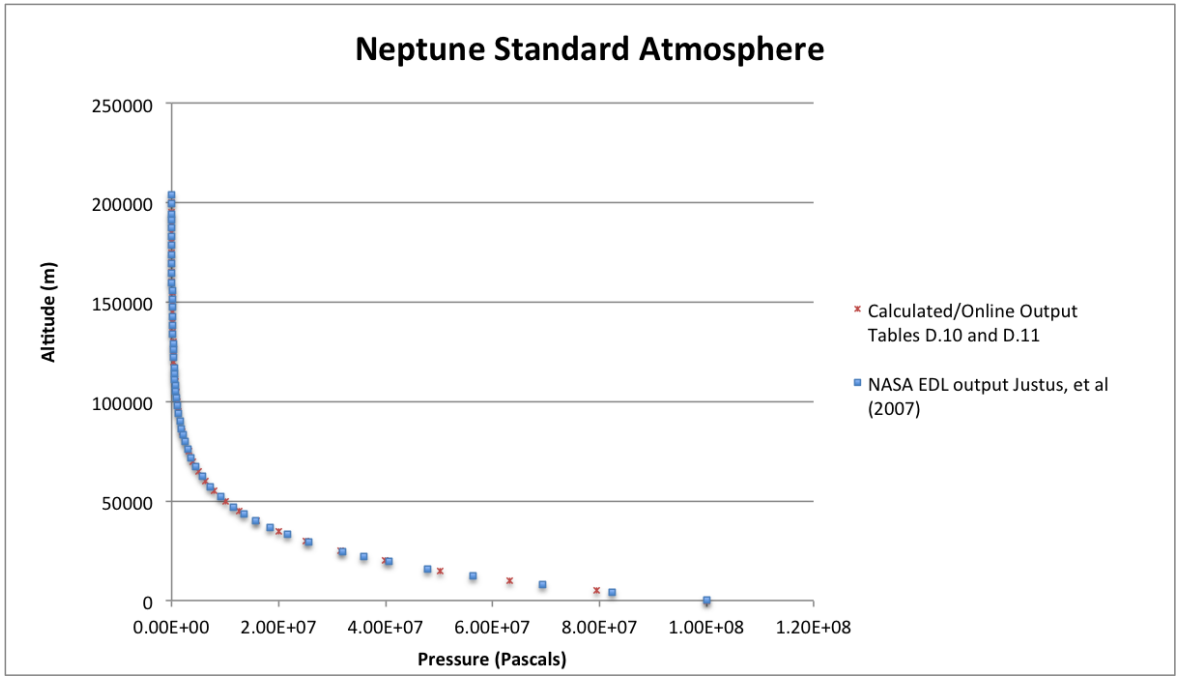

Figure 7.14. Neptune EDL vs. calculator atmosphere pressure profile

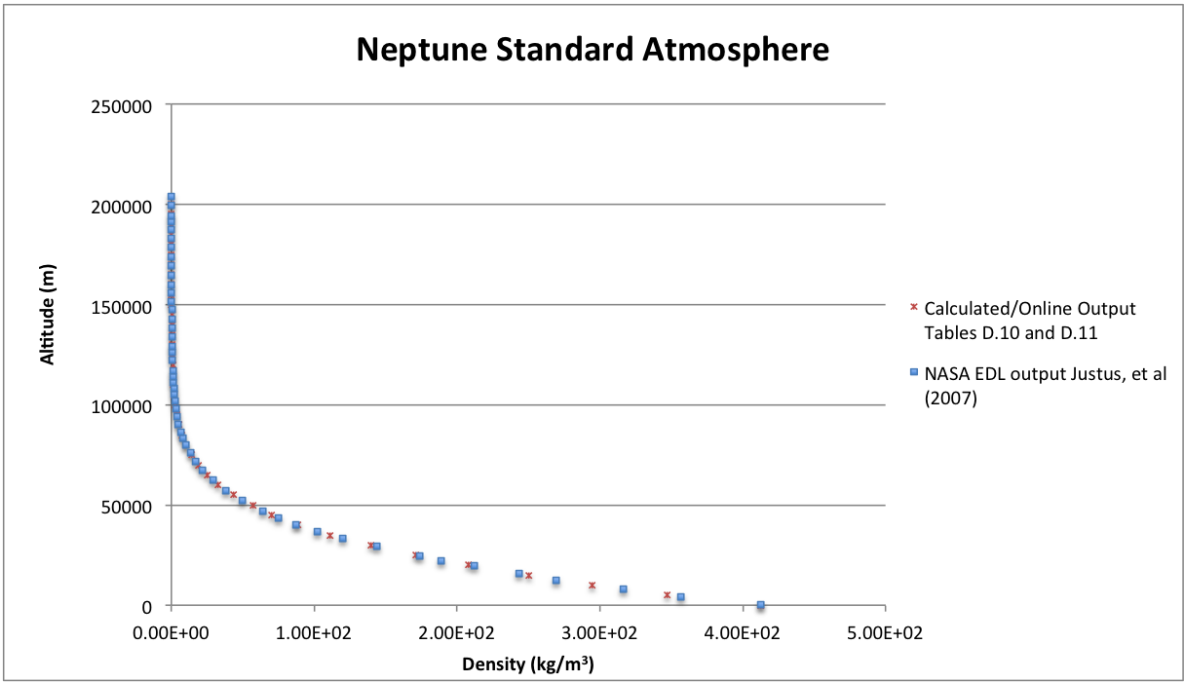

Figure 7.15. Neptune EDL vs. calculator atmosphere density profile

Unlike the other atmospheric data, Neptune did not have as easily accessible information to compare results. The data above is the comparison between the graph that was used as reference material to the output of the calculator. These results are almost identical as was expected showing that this calculator provides accurate information comparable with NASA's own data. The pressure and density were calculated using the standard equations discussed in Chapter 5 and created the desired profiles. The temperature profile for Neptune is very similar to Titan and Mars with an initial decreasing trend followed by a change to increasing temperature with altitude.

Each figure above display similar qualities that should be noted. The pressure and density profiles are all exponential curves that decrease with increasing altitude. The temperature profiles are different for each atmosphere but all show the amount of change that occurs in temperature as the altitude increases. This changes shows not only the relation between temperature and altitude, but also the effects of the atmosphere on temperature as different layers are viewed.

7.3 Results of Calculator Output versus Other Online Calculators

One of the purposes of this thesis is to provide an online calculator for Earth and additionally for other planetary bodies. Similarly to the earlier discussion there are several websites that calculate the 1976 US Standard Atmosphere conditions but for other planets there are either none in existence or they are not searchable in search engines. The 1976 US Standard Atmosphere will be compared to other online calculators of the 1976 US Standard Atmosphere.

Figures 7.15 and 7.16 display two separate inputs with different units in the calculator created in this thesis.

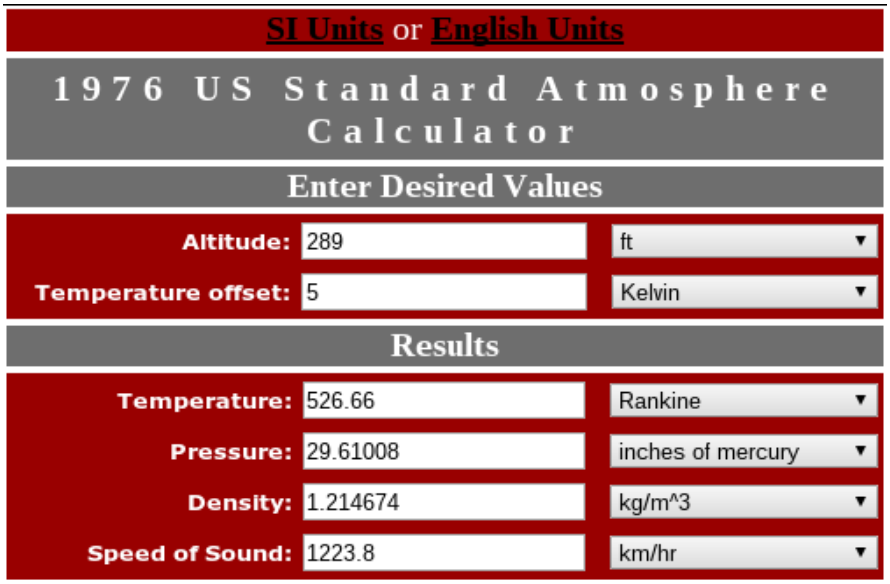

Figure 7.16. First test input data for comparison

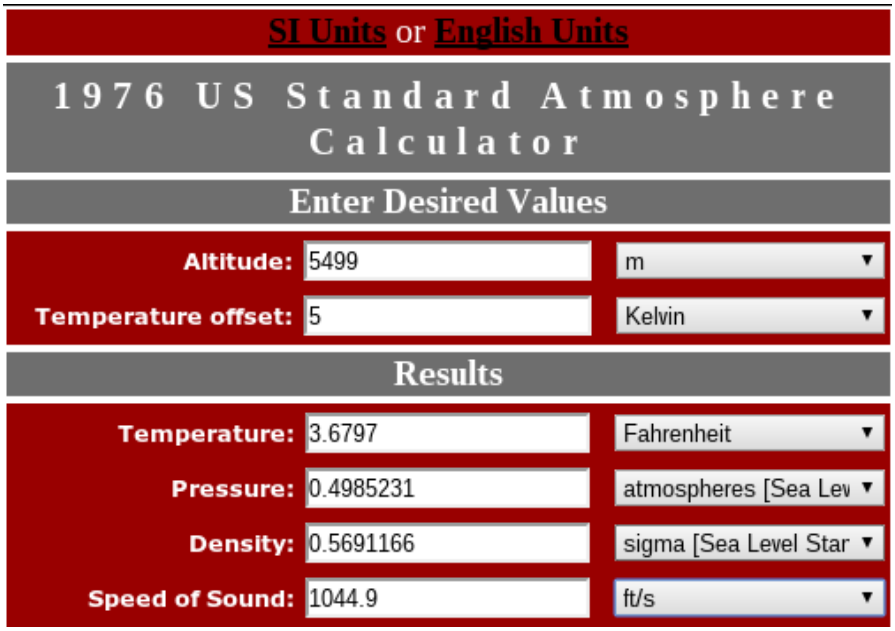

Figure 7.17. Second test input data for comparison

Here the inputs, outputs, and units are clearly displayed and can now be compared to the same inputs from the other websites. Figures 7.17, 7.18, 7.19, and 7.20 show the inputs and units used in the previous figures to compare calculated by other online calculators [46] [47]. As can be seen from these figures the results match with a high degree of precision with the exception of when the other calculators do not have the same capabilities as the calculator designed in this thesis.

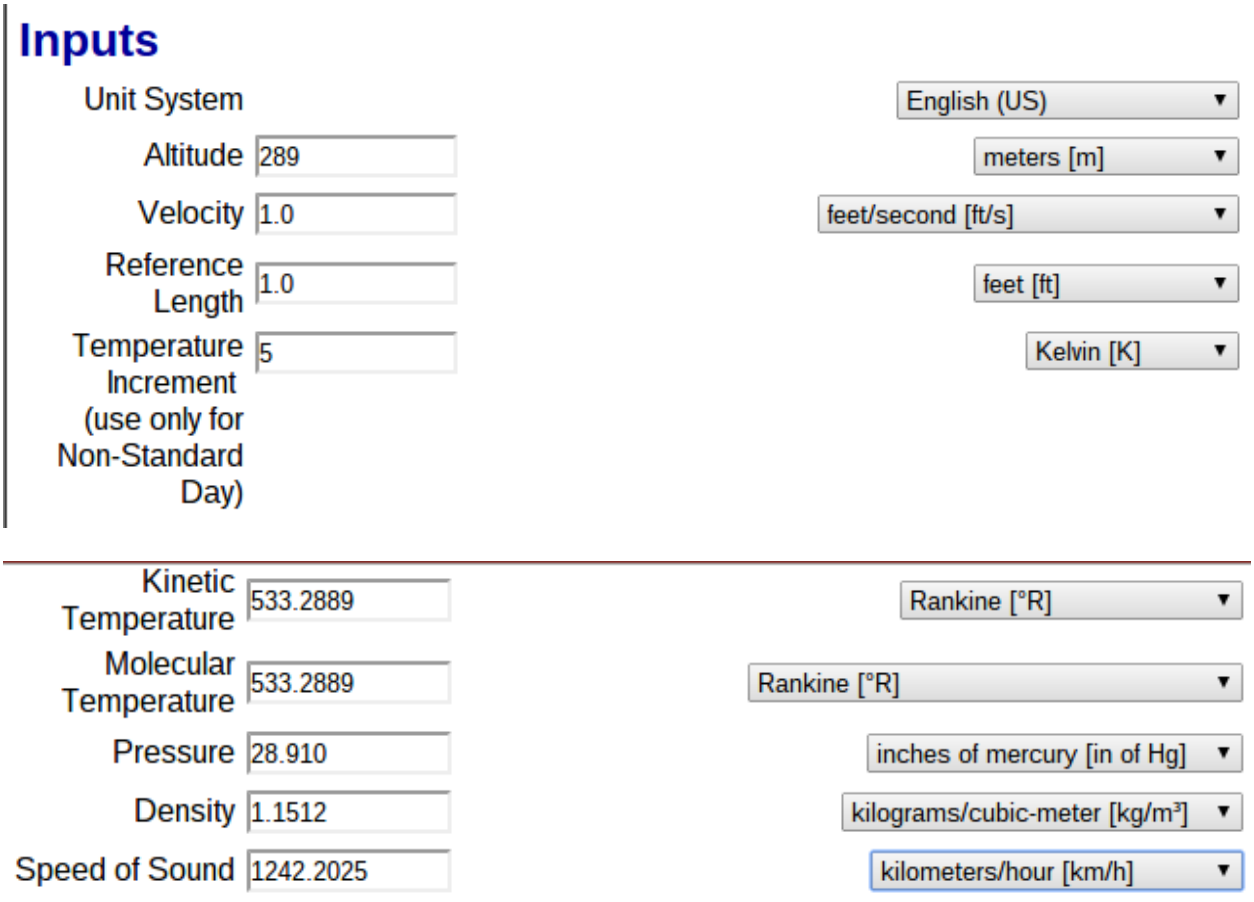

Figure 7.18. Aerospaceweb.org first test data input comparison

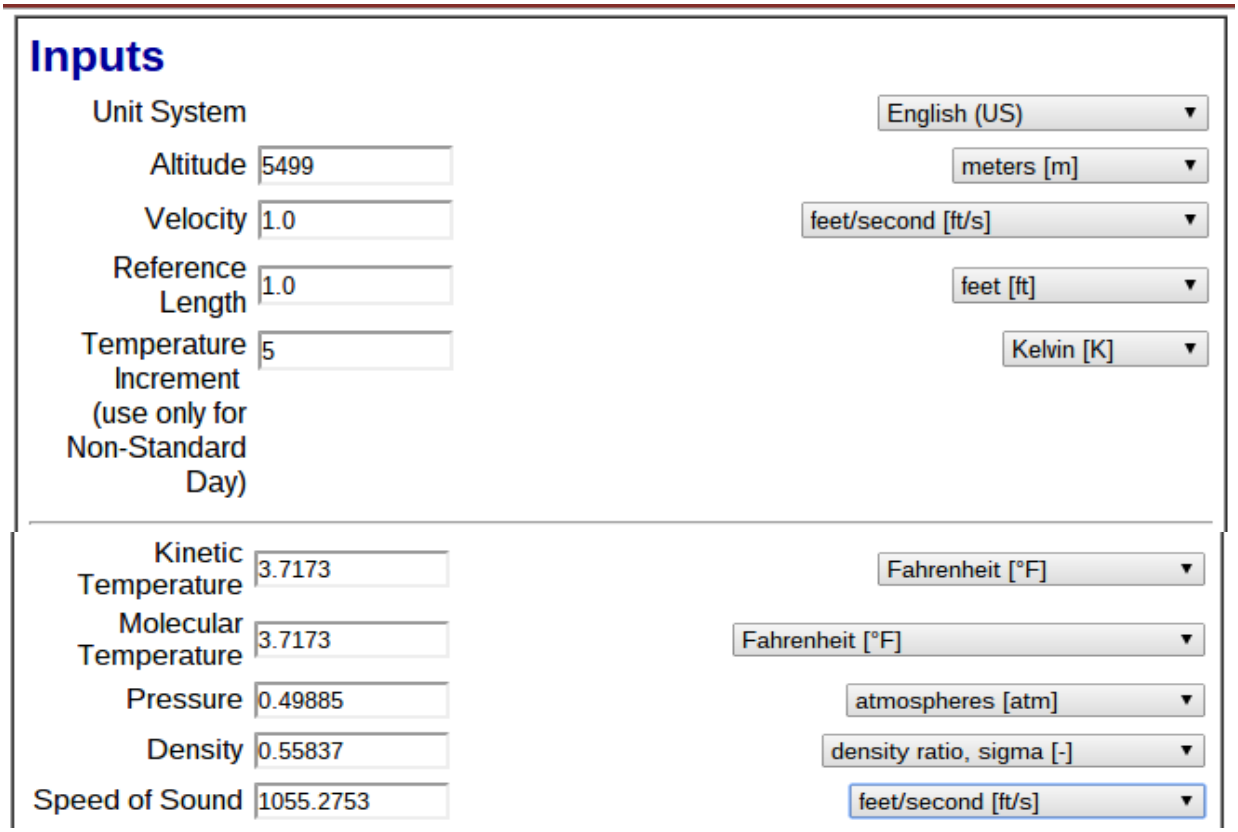

Figure 7.19. Aerospaceweb.org second test data input comparison

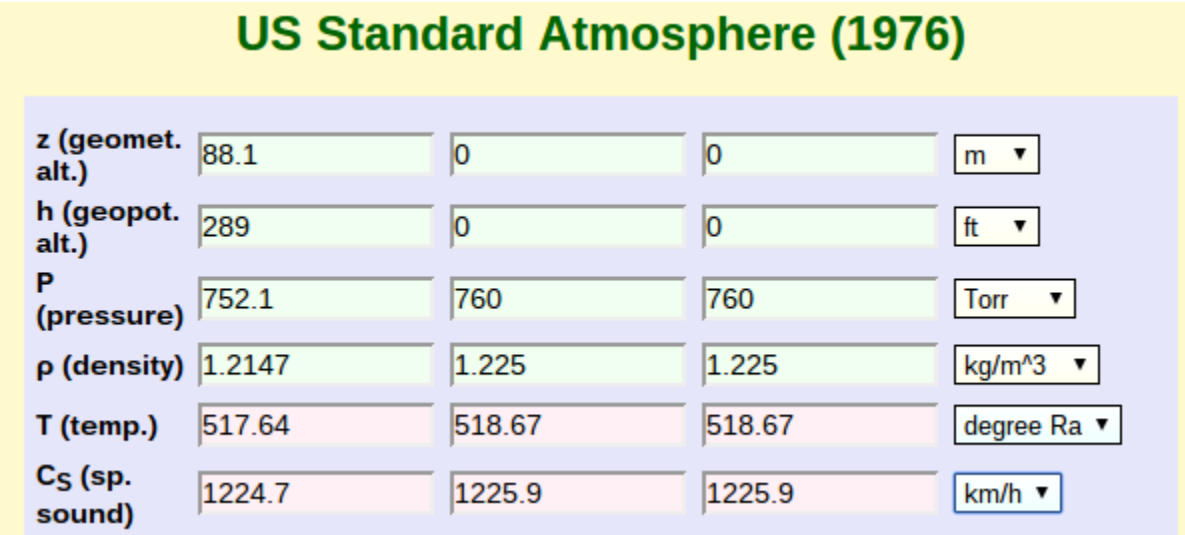

Figure 7.20. Cactus2000 first test data input comparison

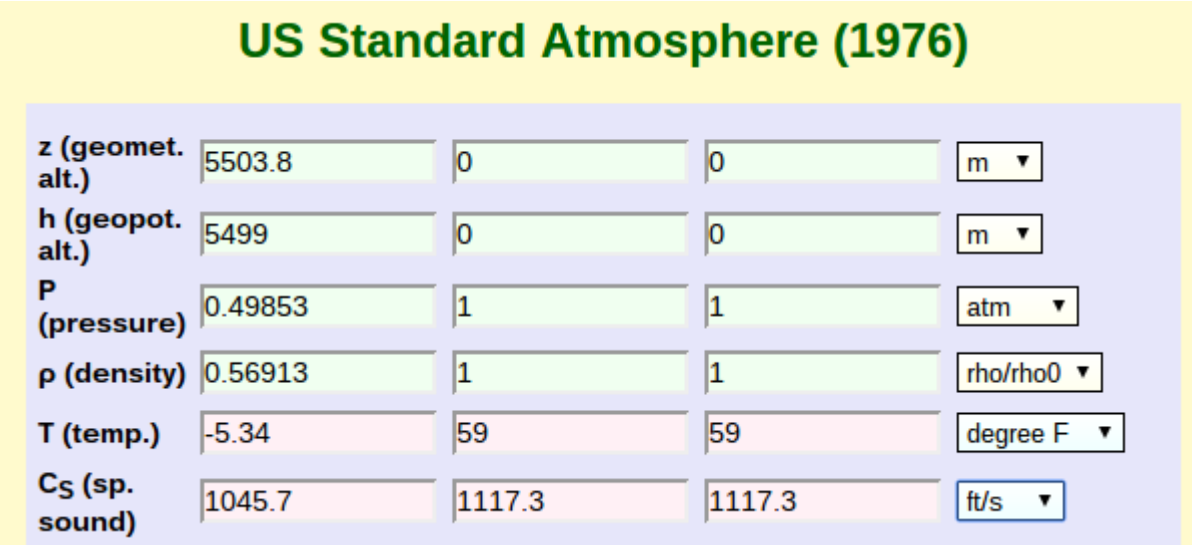

Figure 7.21. Cactus2000 second test data input comparison

Another website used for comparison was Digital Dutch. This website provided a great resource for understanding the capabilities of an online calculator. Although none of the code for Digital Dutch was used it still acted a resource for studying the uses of various built-in functions in webpage code. This website also provided useful links and references showing the user where the information used on the calculator was found [48].

7.4 Future Work

The calculators developed in this thesis were only made for the atmospheres stated previously. These calculators provide the bulk of the code necessary for someone to make websites for other planets and moons in the future. Besides atmospheres, it promotes the use of online calculators for multiple types of sciences across the engineering departments. The internet is a resource already being used almost exclusively by students to find the information

they need for their studies. The more resources that can be provided on the internet the more likely future students will be able to learn about the material.

Calculators for unique functions are not just found online or in Microsoft Excel. The world of computer, tablet, and smartphone applications has risen to the ability of downloading complex calculators to these devices. In future work these calculators and others could be made into the form of an application or desktop widget for the user to have on his/her mobile device.

7.5 Conclusion

This thesis focused on the development of a group of online calculators for the 1976 US Standard Atmosphere, Mars, Venus, Titan, and Neptune. These websites would help promote student learning and encourage new projects concerning space aeronautics. At the end of this thesis the following has been obtained:

- Website created for 1976 US Standard Atmosphere
    - Website could calculate Temperature, Pressure, Density, and Speed of Sound at a given altitude
    - Website could change units as a whole and for each individual variable
    - Website is user friendly and calculates accurate data as seen in Figures 7.1, 7.2, and 7.3
- Website created for Mars Standard Atmosphere
    - Website could calculate Temperature, Pressure, Density, and Speed of Sound at a given altitude

    - Website could change units as a whole and for each individual variable
    - Website is user friendly and calculates accurate data as seen in Figures 7.4, 7.5, and 7.6
- Website created for Venus Standard Atmosphere
    - Website could calculate Temperature, Pressure, Density, and Speed of Sound at a given altitude
    - Website could change units as a whole and for each individual variable
    - Website is user friendly and calculates accurate data as seen in Figures 7.7, 7.8, and 7.9
- Website created for Titan Standard Atmosphere
    - Website could calculate Temperature, Pressure, Density, and Speed of Sound at a given altitude
    - Website could change units as a whole and for each individual variable
    - Website is user friendly and calculates accurate data as seen in Figures 7.10, 7.11, and 7.12
- Website created for Neptune Standard Atmosphere
    - Website could calculate Temperature, Pressure, Density, and Speed of Sound at a given altitude
    - Website could change units as a whole and for each individual variable
    - Website is user friendly and calculates accurate data as seen in Figures 7.13, 7.14, and 7.15

REFERENCES

[1] Justus, C & Johnson, D (1999). *The NASA/MSFC Global Reference Atmospheric Model-1999 Version (GRAM-99).* Marshall Space Flight Center, Alabama: National Aeronautics and Space Administration.

[2] Carmichael, R (2 March 2013). *Properties Of The U.S. Standard Atmosphere 1976. retrieved* January 28 2014, from Public Domain Aeronautical Software Web Site: http://www.pdas.com/atmos.html

[3] (2012). *The Aviation Maintenance Technician Handbook.* Airman Testing Standards Branch, AFS-630, P.O Box 25082, Oklahoma City, OK 73125: United States Department of Transportation, Federal Aviation Administration.

[4] Raggett, D Lam, J, Alexander, I, & Kmiec, M (1998). *Raggett on HTML 4.* Harlow, England: Addison Wesley Longman Limited.

[5] Berners, T & D. Connolly (1995). Hypertext Markup Language - 2.0. *Standards Track,* retrieved January 30, 2014, from http://tools.ietf.org/html/rfc1866

[6] Engelfriet, A (2006). *Introduction to Wilbur*. retrieved January 30 2014, from Web Design Group Web Site: http://htmlhelp.com/reference/wilbur/intro.html

[7] Hickson, I (2004). HTML Living Standard. , . retrieved 30 January 2014, from http://www.whatwg.org/specs/web-apps/current-work/

[8] Lie, H.W. & Bos, B (2005). *Cascading Style Sheets, designing for the Web.* Boston, Massachusetts: Addison-Wesley Professional.

[9] Walsh N (1996). An Introduction to Cascading Style Sheets. *The World Wide Web Journal*. *2(4)*.

[10] Stenström, E (2006). *Beginner's guide to CSS*. retrieved January 30, 2014, from Friendly Bit Web Site: http://friendlybit.com/css/beginners-guide-to-css-and-standards/

[11] Severance C (2012). Java Script: Designing a Language in 10 Days. *Computer*. *45(2),* 7-8.

[12] Andreessen, M (1998). Innovators of the Net: Brendan Eich and JavaScript. *Columns TechVision*. retrieved February 2, 2014, from https://web.archive.org/web/20080208124612/http://wp.netscape.com/comprod/columns/techvision/innovators_be.html

[13] Kowal, K (2009, December, 1). *CommonJS effort sets JavaScript on path for world domination*. retrieved February 2 2014, from Ministry of Innovation / Business of Technology Web Site: http://arstechnica.com/business/2009/12/commonjs-effort-sets-javascript-on-path-for-world-domination/

[14] Brehm, S (2013, November, 8). *The future of web Apps is-ready?-isomorphic JavaScript*. retrieved February 2, 2014, from VentureBeat Web Site: http://venturebeat.com/2013/11/08/the-future-of-web-apps-is-ready-isomorphic-javascript/

[15] Duvall, A Justus, C, & Keller, V (2006). *GLOBAL REFERENCE ATMOSPHERIC MODELS FOR AEROASSIST APPLICATIONS*. retrieved February 3 2014, from Solar System Exploration Web Site: https://solarsystem.nasa.gov/docs/58_duvall.pdf

[16] Addy, G Bauman, W, Cook, D, & et al (2004). *Guide to Reference and Standard Atmosphere Models.* Reston, Virginia: American Institute of Aeronautics and Astronautics.

[17] Lewis, S (2003). Modelling the Martian atmosphere. *Astronomy & Geophysics, 44(4)*. retrieved February 3 2014, from http://astrogeo.oxfordjournals.org/content/44/4/4.6.full

[18] FAA (2008). *A COMPUTER VERSION OF THE U. S. STANDARD ATMOSPHERE, 1978.* Oklahoma City, Oklahoma: Federal Aviation Administration

[19] Gyatt, G (2011). *The Standard Atmosphere*. retrieved February 4 2014, from atmosculator Web Site: http://www.atmosculator.com/The%20Standard%20Atmosphere.html?

[20] CavCar, M (2011). *The International Standard Atmosphere (ISA)*. retrieved February 4 2014, from home.anadolu.edu.tr/ Web Site: http://home.anadolu.edu.tr/~mcavcar/common/ISAweb.pdf

[21] King, R (1978). *A COMPUTER VERSION OF THE U. S. STANDARD ATMOSPHERE, 1978.* Springfield, Virginia: U.S. DEPARTMENT OF COMMERCE.

[22] United States Committee on Extension to the Standard Atmosphere (1976). *U.S. Standard Atmosphere, 1976.* Rockville, Maryland: National Oceanic and Atmospheric Administration.

[23] Carmichael, R (2 March 2013). *Program For Calculating Pressure*. retrieved January 28 2014, from Public Domain Aeronautical Software Web Site: http://www.pdas.com/pressure.html

[24] Justus, C & Braun, R (2007). *Atmospheric Environments for Entry, Descent and Landing (EDL).*Georgia Institute of Technology, Georgia: NASA.

[25] Piazza, E (2009). *About Saturn & Its Moons*. retrieved February 5 2014, from Cassini Solstice Mission Web Site: http://saturn.jpl.nasa.gov/science/index.cfm?SciencePageID=75

[26] Coustenis, A & Taylor, F (2008). *Titan Exploring an Earthlike World.* Tuck Link, Singapore: World Scientific Publishing Co. Pte. Ltd.

[27] Coffey, J (2011, March, 06). *What Is Mars Atmosphere Made Of*. retrieved February 6 2014, from Universe Today Web Site: http://www.universetoday.com/84657/what-is-mars-atmosphere-made-of/

[28] Justus, C & Johnson, D (2001). *Mars Global Reference Atmospheric Model 2001 Version (Mars-GRAM 2001): Users Guide.* Marshall Space Flight Center, Alabama: National Aeronautics and Space Administration.

[29] Robbins, S (2006, September, 14). *Venus*. retrieved 6 February 2014, from Welcome to Journey Through the Galaxy Web Site: http://burro.cwru.edu/stu/advanced/venus.html

[30] Cain, F (2012, March, 12). *Venus*. retrieved 6 February 2014, from Universe Today Web Site: http://www.universetoday.com/14069/venus/

[31] Burdick, A (2014, January, 14 (last updated)). *Neptune*. retrieved 6 February 2014, from Solar Systems Exploration Web Site: http://solarsystem.nasa.gov/planets/profile.cfm?Object=Neptune&Display=OverviewLong

[32] Williams, D (2013, July, 19 (last updated)). *Neptune Fact Sheet*. retrieved 6 February 2014, from National Space Science Data Center Web Site: http://nssdc.gsfc.nasa.gov/planetary/factsheet/neptunefact.html

[33] Weinstein, E *HTML & CSS*. (n.d) retrieved October 31 2013, from Codecademy Web Site: http://www.codecademy.com/tracks/web

[34] Refsnes Data (1997). *W3Schools References*. retrieved 6 February 2014, from w3schools Web Site: http://www.w3schools.com/sitemap/sitemap_references.asp

[35] NASA (2003, January, 29). *Earth-Apollo 17*. retrieved February 11 2014, from National Space Science Data Center Web Site: http://nssdc.gsfc.nasa.gov/imgcat/html/object_page/a17_h_148_22727.html

[36] Splashpress Media *CSS Classes*. (n.d) retrieved February 9 2014, from CSS BASICS Web Site: http://www.cssbasics.com/css-classes/

[37] *U.S Standard Atmosphere*. (n.d) retrieved 6 February 2014, from The Engineering Toolbox Web Site: http://www.engineeringtoolbox.com/standard-atmosphere-d_604.html

[38] Raymer, D (1992). *Aircraft Design: A Conceptual Approach.* Washington, D.C.: American Institute of Aeronautics and Astronautics.

[39] Lutze, F *Introduction to Aerospace Engineering*. (n.d) retrieved February 6 2014, from AOE 3134 Web Site: http://www.dept.aoe.vt.edu/~lutze/AOE2104/3atmosphere.pdf

[40] Hill, P & Peterson, C (1991). *Mechanics and Thermodynamics of Propulsion.* Upper Saddle River, New Jersey: Prentice Hall.

[41] Powell, T & Schneider, F (2004). *The Complete Reference.* Berkeley, California: Osborne/McGraw-Hill.

[42] Clift, J & Jeffries, E (2014, February, 5 (last updated)). *Student Launch*. retrieved February 11 2014, from National Aeronautics and Space Administration Web Site: http://www.nasa.gov/offices/education/programs/descriptions/Student_Launch_Projects.html#.Uvpm2UtX_Ww

[43] Baker, J Cole, S, Abdel-Raouf, E, & Taylor, B (2006). *Near-Space and the BamaSAT Program.* ASEE Southeast Section Conference.

[44] Landis, G Colozza, A, & LaMarre, C (2002). *Atmospheric Flight on Venus.* Cleveland, Ohio: National Aeronautics and Space Administration.

[45] Mendoca, J (2010). *Studies of Venus using a General Circulation Model.* Oxford, England: University of Oxford.

[46] Scott, J et, al (2005, August). *Atmospheric Properties Calculator*. retrieved February 18 2014, from Aerospaceweb Web Site: http://www.aerospaceweb.org/design/scripts/atmosphere/

[47] Krüger, M (2010). *US Standard Atmosphere (1976)*. retrieved February 18 2014, from Cactus2000 Web Site: http://www.cactus2000.de/uk/unit/masssta.shtml

[48] *1976 Standard Atmosphere Calculator*. (1999). Retrieved March 6 2014, from Digital Dutch Web Site: http://www.digitaldutch.com/atmoscalc/

[49] Flanagan, D (2011). *JavaScript: The Definitive Guide.* Sebastopol, California: O'Reilly ` Media, Inc.

[50] Lee, L *JavaScript*. (n.d) retrieved October 4 2013, from Codecademy Web Site: http://www.codecademy.com/tracks/javascript

APPENDIX A

Hypertext Markup Language Code

Code for the atmosphere calculators was written and tested for web browser compatibility. The code for the HTML introduction, input fields, and calculator links are shown below.

```
<!DOCTYPE html>
<html lang="en">

<head>
</style>
<link rel="stylesheet" type="text/css" href="style.css">
<meta charset=utf-8>
<title> 1976 US Standard Atmosphere Calculator</title>

<script type="text/javascript" src="script.js"></script>
</head>
<body onload="loadTheSelection()" onunload="saveTheSelection()">
<div class="overall">
<div class="largegroup1">
```

Figure A.1. Code for HTML linkage to CSS and JavaScript with title

```
<div class="EarthPicture">
<a href="http://aem.eng.ua.edu/">
<img src="http://upload.wikimedia.org/wikipedia/commons/6/6f/Earth_Eastern_Hemisphere.jpg"
alt="http://upload.wikimedia.org/wikipedia/commons/thumb/5/50/ISS034E016601_-
_Stratocumulus_Clouds_-_Pacific_Ocean.jpg/640px-ISS034E016601_-_Stratocumulus_Clouds_-
_Pacific_Ocean.jpg"
width="250" height="250"></a>
</div>
<div class="PlanetCalc"><strong style = "color: white">Earth</strong></div>

<div class="planetlinks"><strong style = "color:white">Other Atmosphere Calculators</strong>
<p class= "planetlinks" style="font-size: 24px; color: #000; text-shadow: 2px 2px 3px #b22e20">
    <a href="http://aem.eng.ua.edu/"><font color="#FF2400">Mars</font></a>
    <a href="http://aem.eng.ua.edu/"><font color="#CC6600">Venus</font></a>
    <a href="http://aem.eng.ua.edu/"><font color="#FDD017">Titan</font></a>
    <a href="http://aem.eng.ua.edu/"><font color="#00008B">Neptune</font></a>
</p>
</div>
</div>
```

Figure A.2. Code for Earth picture link and other planet calculator links

```
<div class="largegroup2">
<div class="Titlebar"><a href="#" onclick="TheUnitsTheStandardWorldPick();
return false;"><font color="000000"><strong>SI Units</strong></font></a> or
<a href="#" onclick="TheEnglishUnits();return false;"><font color="000000">
<strong>English Units</strong></font></a>
</div>
<div class="title"><strong style = "color: white">
 1976 US Standard Atmosphere Calculator</strong>
</div>
<form id="formID">
```

Figure A.3. Code for unit change buttons with form identification

```
<div class="Header"style = "color: white">Enter Desired Values</div>
<table class="Results">
 <tr>
  <td class="tdclassstyle"style = "color: white">
  <strong>Altitude:</strong>
  </td>
  <td><input type="value" name="AltitudeAdjustment" value="0"
        oninput="calculationFunction()">
  </td>
  <td>
    <select name="AltitudeUnitID" onchange="calculationFunction(); return null;">
    <option value="0.3048">ft</option>
    <option value="1000">km</option>
    <option selected value="1">m</option>
    <option value="1852">nautical miles</option>
   </select>
  </td>
 </tr>
 <tr>
```

Figure A.4. Code for the input table title and altitude input options

```
<td class="tdclassstyle"style = "color: white"><strong>Temperature offset:</strong></td>
<td><input type="value" name="DeltaTAdjustment" value="0"
      oninput="calculationFunction()">
</td>
<td><select name="DeltaTUnitID" id="DeltaTUnitID" onchange="calculationFunction();">
  <option value="Kelvin" selected>Kelvin</option>
  <option value="Celsius">Celsius</option>
  <option value="Rankine">Rankine</option>
  <option value="Fahrenheit">Fahrenheit</option>
  <option value="Réaumure">Réaumure</option>
  </select>
</td>
```

Figure A.5. Code for temperature offset input and options

HTML code for the result fields and the options of those fields are shown below

```
<table class="tableoptions">
</table>
<div id="error"></div>
<div class="Header" style = "color: white"><strong>Results</strong></div>
<table class="Results">
 <tr>
  <td class="tdclassstyle"style = "color: white"><strong>Temperature:</strong></td>
  <td><input type="text" size="20" name="TemperatureAdjustment" readonly></td>
  <td>
    <select name="TemperatureUnitID" onchange="calculationFunction();">
     <option value="Celsius">Celsius</option>
     <option selected="" value="Kelvin">Kelvin</option>
     <option value="Fahrenheit">Fahrenheit</option>
     <option value="Rankine">Rankine</option>
     <option value="Réaumure">Réaumure</option>
    </select>
  </td>
 </tr>
 <tr>
```

Figure A.6. Code for results table class and temperature field with options

```
  <td class="tdclassstyle"style = "color: white"><strong>Pressure:</strong></td>
  <td><input type="text" size="20" name="PressureAdjustment" readonly></td>
  <td>
   <select name="PressureUnitID" onchange="calculationFunction();">
    <option value="101325">atmospheres [Sea Level Standard=1]</option>
    <option value="3386.388">inches of mercury</option>
    <option value="100">millibars</option>
    <option selected="" value="1">Pa</option>
    <option value="47.88">lbf/ft^2</option>
    <option value="6894.757">psi</option>
   </select>
  </td>
 </tr>
 <tr>
```

Figure A.7. Code for pressure field with options

```
<td class="tdclassstyle" style = "color: white"><strong>Density:</strong></td>
<td><input type="text" size="20" name="DensityAdjustment" readonly></td><td>
 <select name="DensityUnitID" onchange="calculationFunction();">
  <option selected="" value="1">kg/m^3</option>
  <option value="1.225">sigma [Sea Level Standard=1]</option>
  <option value="515.31788206">slugs/ft^3</option>
 </select>
</td>
</tr>
<tr>
```

Figure A.8. Code for density field with options

```
<td class="tdclassstyle" style = "color: white"><strong>Speed of Sound:</strong></td>
<td><input type="text" size="20" name="SpeedOfSoundAdjustment" readonly></td>
<td>
 <select name="SpeedofSoundUnitID" onchange="calculationFunction();">
  <option value="0.3048">ft/s</option>
  <option value="0.2777777777777778">km/hr</option>
  <option value="0.514444">knots</option>
  <option selected="" value="1">m/s</option>
 </select>
</td>
</tr>
</table>
</div>
</div>
</form>
```

Figure A.9. Code for speed of sound field with options and results end statements

HTML code for the University of Alabama information links and code for website reference link are shown below.

```
<p class="uainfo" style="color: white">
<a href="http://www.ua.edu/disclaimer.html">Disclaimer</a>
<a href="http://www.ua.edu/copyright.html">   Copyright &copy; 2009</a>
<strong>  The University of Alabama | Tuscaloosa, AL 35487 | (205) 348-6010</strong>
</p>
 <p class="uainfo" style="color: white">
<a href="http://modelweb.gsfc.nasa.gov/atmos/us_standard.html"><strong>Data obtained from the
1976 US Standard Atmosphere</strong></a>
</p>
```

Figure A.10. Code for University of Alabama information and reference link

```
 <div class="UAPic">

<a href="http://www.ua.edu/">
<img src="http://visualid.ua.edu/download/Capstone%202013/UA_Square_Logo.jpg"
alt="http://visualid.ua.edu/download/Capstone%202013/UA_Stack_Logo_4c_noNameplate.jpg"
align="center" width="53" height="53"></a>
</div>

</body>
</html>
```

Figure A.11. Code for University of Alabama picture link and HTML end statements

```
<div class="MarsPicture">
<a href="http://aem.eng.ua.edu/"><img
src="http://upload.wikimedia.org/wikipedia/commons/e/e4/Water_ice_clouds_hanging_above_T
harsis_PIA02653_black_background.jpg"
alt="http://upload.wikimedia.org/wikipedia/commons/7/7d/Mars_atmosphere.jpg"
 width="250" height="250"></a>
</div>

<div class="PlanetCalc"><strong style = "color: white">Mars</strong></div>

<div class="planetlinks"><strong style = "color:white">Other Atmosphere Calculators</strong>
   <p class= "planetlinks" style="font-size: 24px; color: #000; text-shadow: 2px 2px 3px
#b22e20"><a href="http://aem.eng.ua.edu/"><font color="#2B65EC">Earth</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#CC6600">Venus</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#FDD017">Titan</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#00008B">Neptune</font></a>
   </p>
</div>
</div>
```

Figure A.12. Code for Mars HTML difference from original

```
<div class="TitanPicture">
<a href="http://aem.eng.ua.edu/"><img
src="http://upload.wikimedia.org/wikipedia/commons/8/84/Titan_in_natural_color_Cassini.jpg"
alt="http://upload.wikimedia.org/wikipedia/commons/4/4a/Titan_and_rings_PIA14909.jpg"
width="250" height="250"></a>
</div>

<div class="PlanetCalc"><strong style = "color: white">Titan</strong></div>

<div class="planetlinks"><strong style = "color:white">Other Atmosphere Calculators</strong>
<p class= "planetlinks" style="font-size: 24px; color: #000; text-shadow: 2px 2px 3px #b22e20">
<a href="http://aem.eng.ua.edu/"><font color="#FF2400">Mars</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#CC6600">Venus</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#2B65EC">Earth</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#00008B">Neptune</font></a>
</p>
</div>
</div>
```

Figure A.13. Code for Titan HTML difference from original

```
<div class="VenusPicture">
<a href="http://aem.eng.ua.edu/"><img
src="http://upload.wikimedia.org/wikipedia/commons/8/85/Venus_globe.jpg"
alt="http://upload.wikimedia.org/wikipedia/commons/e/e5/Venus-real_color.jpg"
width="250" height="250"></a>
</div>

<div class="PlanetCalc"><strong style = "color: white">Venus</strong></div>

<div class="planetlinks"><strong style = "color:white">Other Atmosphere Calculators</strong>
<p class= "planetlinks" style="font-size: 24px; color: #000; text-shadow: 2px 2px 3px #b22e20">
<a href="http://aem.eng.ua.edu/"><font color="#FF2400">Mars</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#2B65EC">Earth</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#FDD017">Titan</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#00008B">Neptune</font></a></p>
</div>
</div>
```

Figure A.14. Code for Venus HTML difference from original

```
<div class="NeptunePicture">
 <a href="http://aem.eng.ua.edu/"><img
src="http://upload.wikimedia.org/wikipedia/commons/0/06/Neptune.jpg"
alt="http://upload.wikimedia.org/wikipedia/commons/8/8c/Neptune_storms.jpg"
width="259" height="250"></a>
</div>

<div class="PlanetCalc"><strong style = "color: white">Neptune</strong></div>

<div class="planetlinks"><strong style = "color:white">Other Atmosphere Calculators</strong>
<p class= "planetlinks" style="font-size: 24px; color: #000; text-shadow: 2px 2px 3px #b22e20">
<a href="http://aem.eng.ua.edu/"><font color="#2B65EC">Earth</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#CC6600">Venus</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#FDD017">Titan</font></a>
<a href="http://aem.eng.ua.edu/"><font color="#FF2400">Mars</font></a>
 </p>
</div>
</div>
```

Figure A.15. Code for Neptune HTML difference from original

APPENDIX B

Cascading Style Sheet

Code for the atmosphere calculators was written and tested for web browser compatibility. The code for the CSS general styling classes and large page divisions are shown below.

```
body {

        background-repeat:no-repeat;
        background-attachment:fixed;
        background-position:center;
        background-size:auto;
        background-color:black;
        height: 100%;
        margin: auto;
         max-width: 960px;
        min-width: 750px;
        padding: 0;
}
p, ul, td {
        font-family: "Lucida Sans",Lucida Sans Unicode,Verdana,Arial,sans-serif;
        font-size: 10pt;
        color:#333;
        text-align:right;
}
```

Figure B.1. Code for generalized body and child tag styling

```
h1 {
        font-family:"Lucida Sans",Lucida Sans Unicode,Verdana,Arial,sans-serif;
        font-size: 16pt;
        font-weight: normal;
        text-align: left;
        margin-top: 0;
        padding-top: 10px;
        color:#333;
}
h2 {
        font-family: "Lucida Sans",Lucida Sans Unicode,Verdana,Arial,sans-serif;
        font-size: 12pt;
        font-weight: normal;
        text-align: left;
        margin-top: 0;
        padding-top: 10px;
        color:#333;
}
h3 {
        font-family: "Lucida Sans",Lucida Sans Unicode,Verdana,Arial,sans-serif;
        font-size: 10pt;
        font-weight: bold;
        text-align: left;
        margin-top: 0;
        margin-bottom: 0;
        padding-top: 10px;
        padding-bottom: 0;
}
```

Figure B.2. Code for generalized header styling for all three size types

Code for the custom styling and formatting of the website's individual classes is shown below.

```
.EarthPicture {
        margin-left:75px;
        margin-right:auto;
        border: none;
        cursor:pointer;
}
.PlanetCalc {
        text-align:center;
        font-size: 40pt;
}
.EarthPicture img{
        cursor:pointer;
        border: none;
}
.planetlinks{
        text-align:center;
        font-size:15pt;
        margin-top:auto;
}
```

Figure B.3. Code for positioning and styling of the picture of Earth and planet links

```
.overall {
        width: 1000pt;
        height:300pt;
        overflow: hidden;
}
.largegroup1 {
        width: 300pt;
        float:left;
        margin-top:0;
}
.largegroup2 {
        width: auto;
        float: left;
        margin-top:30px;
}
```

Figure B.4. Code for the styling and formatting of the large divisions

```
.Titlebar {
        width: 500px;
        height: 25px;
        margin-left:auto;
        margin-right:auto;
        background-color:#990000;
        border: 2px solid white;
        font-size: 15pt;
        text-align:center;
        color:white;
}
.title {
        width: 500px;
        margin-left:auto;
        margin-right:auto;
        font-size: 16pt;
        letter-spacing: 0.3em;
        padding-top: 6px;
        padding-bottom: 4px;
        text-align: center;
        background-color: #6E6E6E;
        border: 2px solid white;
        color:#333;
}
```

Figure B.5. Code for the styling and positioning of the title classes

```
.Results {
        width: 504px;
        background-color:#990000;
        border: 2px solid white;
        padding: 0;
        position: relative;
        margin:0;
}
.Results td {
        width: 33%;
}
.Results select {
        width: 95%;
}
```

Figure B.6. Code for the styling and formatting of the results table

```
.Header {
        background-color: #6E6E6E;
        margin-left:auto;
        margin-right:auto;
        border: 2px solid white;
        padding: 0;
        font-weight: bold;
        width:500px;
        text-align:center;
        font-size:15pt;
}
.uainfo {
        font-size: 10pt;
        text-align:center;
        margin-top:0;
}
.UAPic {
        margin-left:auto;
        margin-right:auto;
        margin-top:0;
        border: none;
        cursor:pointer;
        text-align:center;
}
```

Figure B.7. Code for the styling of headers and University of Alabama information

```
.tdclassstyle {
        text-align: right;
}
.tableoptions {
        background-color: #990000;
        padding: 0;
        width: 500px;
        text-align:right;
        margin:0;
}
.tdleftoptions {
        width: 35%;
        text-align: right;
}
```

Figure B.8. Code for styling and formatting of the input tables and individual fields

```
#error {
        display: none;
        margin-left:auto;
        margin-right:auto;
        padding: 2px 4px;
        background-color: white;
        border: 1px solid #990000;
        color:#990000;
        position: relative;
        text-align: center;
        max-width: 745px;
        min-width: 250px;
}
```

Figure B.9. Code for the styling and positioning of the altitude error box

```
.MarsPicture {
  margin-left:75px;
  margin-right:auto;
  border: none;
  cursor:pointer;
 }

 .MarsPicture img{
  cursor:pointer;
   border: none;
 }
```

Figure B.10. Code for Mars CSS difference from original

```
.VenusPicture {
  margin-left:75px;
  margin-right:auto;
  border: none;
  cursor:pointer;
 }

 .VenusPicture img{
  cursor:pointer;
   border: none;
 }
```

Figure B.11. Code for Titan CSS difference from original

```
.TitanPicture {
  margin-left:75px;
  margin-right:auto;
  border: none;
  cursor:pointer;
}

.TitanPicture img{
  cursor:pointer;
   border: none;
}
```

Figure B.12. Code for Venus CSS difference from original

```
.NeptunePicture {
  margin-left:75px;
  margin-right:auto;
  border: none;
  cursor:pointer;
}

.NeptunePicture img{
  cursor:pointer;
   border: none;
}
```

Figure B.13. Code for Neptune CSS difference from original

APPENDIX C

JavaScript Code

Functions for the atmosphere calculators were written and tested for web browser compatibility.

The code for the JavaScript atmosphere equations is shown below [49] [50].

```
// globals
var altitude;
var deltaT = 0;
var temperature;
var pressure;
var speedOfSound;
var density;
```

Figure C.1. Global variables declared for use throughout program

```
//function that uses the atmospheric equations
function AtmosphereCalculatorEquations(){

  var i;                        //Will allow the appropriate value to be assigned in the equation
  var BottomLayerRelativePressure;
  var AltitudeChange;      //h-h1 where h1 is the altitude bottom of the layer
  var CurrentPressure;    //pressure variable/
  var BottomLayerTemp; //as sounds: temp at base of layer
  var tempGradient;        //Lapse rate [K/km]
  var BottomLayerDensity;

  //Default unchanging variables
  var temperatureatSeaLevel    = 288.16;    // Temperature at sea level [deg K]
  var densityatSeaLevel          = 1.225;     // Density at sea level [kg/m3]
  var pressureatSeaLevel        = 101325;   // Pressure at sea level [Pa]
  var R                                 = 287.036;  //air gas constant [J/kg/K]
  var gamma                         = 1.4;
  var gravity                        = 9.80665;  // [m/s2]
```

Figure C.2. Atmosphere equation function with initial variables

```
//Array to designate value at each atmospheric layer to be used as a base value in equations.
//Retrieved from NASA atmosphere tables
var atmosphereLayers              = new Array(0, 11000, 20000, 32000, 47000,51000, 71000,
                                              84852);
var PressureRatioatLayers         = new Array(1, 2.23355570684799e-1, 5.4030803694344e-2,
                                        8.56615261842e-3, 1.094455932173e-3,
                                        6.60565837016e-4, 3.9041059064e-5,
                                        3.684406018e-6);
var TemperatureatLayers           = new Array(288.16, 216.66, 216.66, 228.66, 270.66, 270.66,
                                         214.66, 186.956);
var TemperatureGradientsatLayers = new Array(-6.5, 0, 1, 2.8, 0, -2.8, -2, 0);
var DensityatLayers               = new Array(1.225, 3.63905e-1, 8.803e-2, 1.3224e-2,
                                              1.427e-3, 8.62e-4, 6.4e-5, 7e-6);
```

Figure C.3. Arrays for storing information at each altitude layer

```
// Input altitude check
switch (altitude,temperature,pressure,density,speedOfSound){
        case ((altitude < 0) || (altitude > 85000)):
        altitude      = 0;
        temperature   = 0;
        pressure      = 0;
        density       = 0;
        speedOfSound = 0;

        return "Alert: Altitude must in the range of 0 and 85000 meters.";
 }
```

Figure C.4. Switch and case statement for checking altitude input

```
i = 0;
        if (altitude > 0) {
                while (altitude > atmosphereLayers[i + 1]) {
                i = i + 1;
                 }
         }
```

Figure C.5. Code for altitude array loop

```
var GravityConstant = gravity * (1000 / R);

if (isNaN (deltaT)){
   switch (deltaT){
     default:
        return 0;
   }
}
```

Figure C.6. Code for Hydrostatic variable and temperature offset check

```
//Using appropriate i from selection inputs correct value from array into these variables for
//equations
  BottomLayerTemp              = TemperatureatLayers[i];
  tempGradient                 = TemperatureGradientsatLayers[i] / 1000;
  BottomLayerRelativePressure  = PressureRatioatLayers[i];
  BottomLayerDensity           = DensityatLayers[i];
  AltitudeChange               = altitude - atmosphereLayers[i];
  temperature                  = BottomLayerTemp + tempGradient *AltitudeChange;
```

Figure C.7. Code for array storage into variables for calculations

```
// Pressure Equation at Selected Altitude for when temperature gradient is "0" otherwise the
//equation shifts using different variables
  if (Math.abs(tempGradient) <  0.0000000001) {
    CurrentPressure = BottomLayerRelativePressure * Math.exp(-GravityConstant *
    AltitudeChange/1000 / BottomLayerTemp);
    }
    else {
    CurrentPressure =  BottomLayerRelativePressure*Math.pow(temperature/BottomLayerTemp,
                       -GravityConstant/tempGradient/1000);
        }
pressure = CurrentPressure * pressureatSeaLevel;
 // Equation for speed of sound at given temperature
  speedOfSound = Math.sqrt(gamma * R * temperature);

  if (Math.abs(tempGradient) <  0.0000000001){
  density = BottomLayerDensity * Math.exp(-GravityConstant * AltitudeChange/1000/
BottomLayerTemp);
   }
   else {
    density= BottomLayerDensity *
Math.pow(temperature/BottomLayerTemp,-((GravityConstant/tempGradient/1000) + 1));
       }
  temperature  = temperature + deltaT;
}
```

Figure C.8. Code for final temperature, pressure, density, and speed of sound equations

Functions for the loading and saving of information in the JavaScript program and within the HTML code are shown below.

```
function infoLoader(RandomInput){

  return localStorage.getItem(RandomInput);
}
```

Figure C.9. Code for loading information stored locally

```
function DefaultLoaderCoder(RandomInput, ReturnThisZero){

 var selected = infoLoader(RandomInput);

 if (selected !== null) {
  return selected;
 }
 else {
  return ReturnThisZero;
 }
}
```

Figure C.10. Code for loading value or zero

```
function RetrieveFunction(itemRetrieved) {

 var selected = infoLoader(itemRetrieved.name);

 if (selected !== null) {
  itemRetrieved.selectedIndex = selected;
 }
}
```

Figure C.11. Code for retrieving information from selected index

```
function infoSaver(RandomInput, saveStuff) {
 localStorage.setItem(RandomInput, saveStuff);
}

function SaveFunction(itemRetrieved) {
 infoSaver(itemRetrieved.name, itemRetrieved.selectedIndex);
}
```

Figure C.12. Code for saving information into local storage

```
function loadTheSelection(){

 var x = document.forms.formID;

          x.a=RetrieveFunction.AltitudeUnitID;
          x.b=RetrieveFunction.TemperatureUnitID;
          x.c=RetrieveFunction.DeltaTUnitID;
          x.d=RetrieveFunction.PressureUnitID;
          x.e=RetrieveFunction.DensityUnitID;
          x.f=RetrieveFunction.SpeedofSoundUnitID;

          x.AltitudeAdjustment.value = DefaultLoaderCoder('AltitudeAdjustment', 0);

 calculationFunction();
}
```

Figure C.13. Code for loading values into the HTML fields

```
        function saveTheSelection() {

          var x = document.forms.formID;

                x.a = SaveFunction.AltitudeUnitID;
                x.b = SaveFunction.TemperatureUnitID;
                x.c = SaveFunction.DeltaTUnitID;
                x.d = SaveFunction.PressureUnitID;
                x.e = SaveFunction.DensityUnitID;
                x.f = SaveFunction.SpeedofSoundUnitID;

                infoSaver('AltitudeAdjustment', x.AltitudeAdjustment.value);


        }
```

Figure C.14. Code for saving values into the HTML fields

```
function TheEnglishUnits() {
  var x = document.forms.formID;
          x.AltitudeUnitID.selectedIndex          = 0;
          x.TemperatureUnitID.selectedIndex     = 2;
          x.DeltaTUnitID.selectedIndex          = 2;
          x.PressureUnitID.selectedIndex        = 5;
          x.DensityUnitID.selectedIndex          = 2;
          x.SpeedofSoundUnitID.selectedIndex   = 0;

  saveTheSelection();
  calculationFunction();
}

function TheUnitsTheStandardWorldPick() {
  var x = document.forms.formID;
          x.AltitudeUnitID.selectedIndex          = 2;
          x.TemperatureUnitID.selectedIndex     = 1;
          x.DeltaTUnitID.selectedIndex          = 0;
          x.PressureUnitID.selectedIndex        = 3;
          x.DensityUnitID.selectedIndex          = 0;
          x.SpeedofSoundUnitID.selectedIndex   = 3;

  saveTheSelection();
  calculationFunction();
}
```

Figure C.15. Code for loading the units in HTML to default values

Functions for value conversions, displaying information and errors, and calculations are shown in the code below.

```
function displayWhatsWrong(lnputthisNumber) {
  var RandomVariable = document.getElementByld("error");

  RandomVariable.innerHTML = lnputthisNumber;
  RandomVariable.style.display = 'block';
}

function getRidofit() {
  var RandomVariable = document.getElementByld("error");

  RandomVariable.innerHTML = "";
  RandomVariable.style.display = 'none';
}
```

Figure C.16. Code for displaying errors in HTML and for hiding errors

```
function SIDeltaTConversion(ResultsTemperature, ResultsTemperatureUnit){

  switch (ResultsTemperatureUnit){
          case "Kelvin": return ResultsTemperature;
          case "Celsius": return ResultsTemperature + 273.15;
          case "Rankine": return 5/9 * ResultsTemperature;
          case "Fahrenheit": return (5/9*(ResultsTemperature- 32)) + 273.15;
          case "Réaumure": return 5/4 * ResultsTemperature;
          default: alert ("Sorry "+ResultsTemperatureUnit+" unknown");
  }
}
```

Figure C.17. Switch case statements for temperature offset conversion to SI units

```
function SITemperatureConversion(ResultsTemperature, ResultsTemperatureUnit){

  switch(ResultsTemperatureUnit){
        case "Celsius": return ResultsTemperature - 273.15;
        case "Fahrenheit": return 9/5 * ResultsTemperature - 459.67;
        case "Kelvin": return ResultsTemperature;
        case "Rankine": return 9/5 * ResultsTemperature;
        case "Réaumure": return 4/5 * (ResultsTemperature - 273.15);
        default:alert ("Sorry"+ResultsTemperatureUnit+" unknown");
  }
 }
```

Figure C.18. Switch case statements for temperature conversion to SI units

```
function displayIt(InputthisNumber) {
  var adjustmentVariable;
  var x = document.forms.formID;
       if (InputthisNumber){
       displayWhatsWrong(InputthisNumber);
          x.TemperatureAdjustment.value    = ' ';
          x.PressureAdjustment.value       = ' ';
          x.DensityAdjustment.value        = ' ';
          x.SpeedOfSoundAdjustment.value = ' ';

       } else{
     getRidofit();
     adjustmentVariable = SITemperatureConversion(temperature,x.TemperatureUnitID.value);
     x.TemperatureAdjustment.value = adjustmentVariable.toPrecision(5);
     adjustmentVariable = pressure / x.PressureUnitID.value;
     x.PressureAdjustment.value = adjustmentVariable.toPrecision(7);
     adjustmentVariable = density / x.DensityUnitID.value;
     x.DensityAdjustment.value = adjustmentVariable.toPrecision(7);
     adjustmentVariable = speedOfSound / x.SpeedofSoundUnitID.value;
     x.SpeedOfSoundAdjustment.value = adjustmentVariable.toPrecision(5);
  }
}
```

Figure C.19. Code for displaying information from other functions

```
function calculationFunction (){

   var RandomVariable;
   var x = document.forms.formID;
         deltaT = parseFloat(x.DeltaTAdjustment.value);
         deltaT = SIDeltaTConversion(deltaT, x.DeltaTUnitID.value);
         x.AltitudeSelector = x.AltitudeUnitID.selectedIndex;
         x.AltitudeSelectorUnit  = x.AltitudeUnitID.options[x.AltitudeSelector].text;
   var SetAltitudeSelectorUnitToSI = x.AltitudeUnitID.value;
   altitude = x.AltitudeAdjustment.value*SetAltitudeSelectorUnitToSI;
   var ThisIsTheLowestIWillGo= 0 / SetAltitudeSelectorUnitToSI;
   var ThisIsTheHighestIWillGo= 85000 / SetAltitudeSelectorUnitToSI;

    if ((altitude < 0) || (altitude > 85000)){
         switch (RandomVariable){
         default:RandomVariable = "Altitude must be at least " +
ThisIsTheLowestIWillGo.toFixed(0) + " and no higher than " +
ThisIsTheHighestIWillGo.toFixed(0) + " " + x.AltitudeSelectorUnit + ".";
                                        }
}else{
   RandomVariable = AtmosphereCalculatorEquations();
}
 displayIt(RandomVariable);
}
```

Figure C.20. Code for calculating atmosphere function with appropriate data

The Code differences between the other atmospheric calculators and Earth can be seen below.

```
function AtmosphereCalculatorEquations(){
 var temperatureatSurface   = 212.065;   // Temperature at surface [deg K]
 var densityatSurface       = 0.0189;    // Density at surface [kg/m3]
 var pressureatSurface      = 557.4;     // Pressure at surface [Pa]
 var R                      = 188.92;  //air gas constant [J/kg/K]
 var gravity                = 3.716;   //m/s^2
 var gamma                  = 1.29;

 switch (altitude,temperature,pressure,density,speedOfSound){
 case ((altitude < 0) || (altitude > 200000)):
   altitude      = 0;
   temperature   = 0;
   pressure      = 0;
   density       = 0;
   speedOfSound = 0;
   return "Alert: Altitude must in the range of 0 and 200000 meters.";
 }

   if (isNaN (deltaT)){
   switch (deltaT){
     default:
         return 0;
   }
}
```

Figure C.21. Code for Mars atmosphere function difference from original

```
  if (altitude <= 4000){
      temperature = 212.065;

   } else if ((altitude > 4000) &&(altitude <= 50000)){
     temperature= ((altitude/1000) - 153.39)/(-0.7127);
   } else if ((altitude > 50000) &&(altitude <= 73000)) {
      temperature = ((altitude/1000) - 343.9)/(-1.9749);
   } else if ((altitude > 73000) &&(altitude <= 101000)) {
      temperature = 137.447;
   } else if ((altitude > 101000) &&(altitude <= 130000)) {
      temperature = ((altitude/1000) + 15.673)/(0.8529);
   } else if ((altitude > 130000) &&(altitude <= 140000)) {
      temperature = ((altitude/1000) - 109.44)/(0.1272);
   } else if ((altitude > 140000) &&(altitude <= 153000)) {
      temperature = ((altitude/1000) - 80.249)/(0.2456);
   } else if ((altitude > 153000) &&(altitude <= 170000)) {
      temperature = ((altitude/1000) + 2.399)/(0.5258);
   } else{
      temperature = ((altitude/1000) + 448.41)/(1.8781);
   }
```

Figure C.22. Code for Mars atmosphere equation difference from original

```
function AtmosphereCalculatorEquations(){

 var temperatureatSurface  = 87.388;     // Temperature at surface [deg K]
 var densityatSurface      = 5.832;      // Density at surface [kg/m3]
 var pressureatSurface     = 147820;   // Pressure at surface [Pa]
 var R                     = 290;       //gas constant [J/kg/K]
 var gamma                 = 1.3846;
 var gravity               = 1.35;

 switch (altitude,temperature,pressure,density,speedOfSound){
 case ((altitude < 0) || (altitude > 200000)):
   altitude      = 0;
   temperature   = 0;
   pressure      = 0;
   density       = 0;
   speedOfSound = 0;

   return "Alert: Altitude must in the range of 0 and 200000 meters.";
 }
   if (isNaN (deltaT)){
         switch (deltaT){
                default:
                return 0;
   }
}
```

Figure C.23. Code for Titan atmosphere function difference from original

```
if (altitude === 0){
    temperature = temperatureatSurface;

 } else if ((altitude > 0) &&(altitude <= 26000)){
    temperature= ((altitude/1000) - 114.88)/(-1.2897);
 } else if ((altitude > 26000) &&(altitude <= 39000)) {
    temperature = ((altitude/1000) - 494.42)/(-6.6191);
 } else if ((altitude > 39000) &&(altitude <= 42000)) {
    temperature = 68.74;
 } else if ((altitude > 42000) &&(altitude <= 58000)) {
    temperature = ((altitude/1000) + 326.7461)/(5.3722);
 } else if ((altitude > 58000) &&(altitude <= 70000)) {
    temperature = ((altitude/1000) - 33.276)/(0.3572);
 } else if ((altitude > 70000) &&(altitude <= 85000)) {
    temperature = ((altitude/1000) - 7.3985)/(0.6007);
 } else if ((altitude > 85000) &&(altitude <= 107000)) {
    temperature = ((altitude/1000) + 120.99)/(1.5845);
 } else if ((altitude > 107000) &&(altitude <= 158000)) {
    temperature = ((altitude/1000) + 317.55)/(2.929);
 } else{
    temperature = ((altitude/1000) + 1299.1)/(8.9259);
 }
```

Figure C.24. Code for Titan atmosphere equation difference from original

```
function AtmosphereCalculatorEquations(){
 var temperatureatSurface    = 735;           // Temperature at surface [deg K]
 var densityatSurface        = 64.79;       // Density at surface [kg/m3]
 var pressureatSurface       = 9210000;  // Pressure at surface [Pa]
 var R                       = 188.92;    //air gas constant [J/kg/K]
 var gamma                   = 1.2857;
 var gravity                 = 8.87;

 switch (altitude,temperature,pressure,density,speedOfSound){
 case ((altitude < 0) || (altitude > 250000)):
  altitude     = 0;
  temperature  = 0;
  pressure     = 0;
  density      = 0;
  speedOfSound = 0;
  return "Alert: Altitude must in the range of 0 and 250000 meters.";
 }

 if (isNaN (deltaT)){
  switch (deltaT){
    default:
        return 0;
  }
}
```

Figure C.25. Code for Venus atmosphere function difference from original

```
  if (altitude === 0){
      temperature = temperatureatSurface;

    } else if ((altitude > 0) &&(altitude <= 58000)){
      temperature= ((altitude/1000) - 93.671)/(-0.1287);
    } else if ((altitude > 58000) &&(altitude <= 92000)) {
      temperature = ((altitude/1000) - 143.38)/(-0.3214);
    } else if ((altitude > 92000) &&(altitude <= 98000)) {
      temperature = ((altitude/1000) + 3.1715)/(0.5916);
    } else if ((altitude > 98000) &&(altitude <= 102000)) {
      temperature = ((altitude/1000) - 221.07)/(-0.7068);
    } else if ((altitude > 102000) &&(altitude <= 146000)) {
      temperature = ((altitude/1000) + 61.359)/(1.1542);
    } else if ((altitude > 146000) &&(altitude <= 157000)) {
      temperature = ((altitude/1000) - 77.329)/(0.3792);
    } else if ((altitude > 157000) &&(altitude <= 172000)) {
      temperature = ((altitude/1000) + 66.91)/(1.0701);
    } else if ((altitude > 172000) &&(altitude <= 200000)) {
      temperature = ((altitude/1000) + 851.07)/(4.6092);
    } else {
      temperature = 228.1536;
    }
```

Figure C.26. Code for Venus atmosphere equation difference from original

```
function AtmosphereCalculatorEquations(){

 var temperatureatSurface   = 67.07;          // Temperature at surface [deg K]
 var pressureatSurface      = 100000000;  // Pressure at surface [Pa]
 var R                      = 3614.91;      //air gas constant [J/kg/K]
 var gravity                = 11.15;        //m/s^2
 var gamma                  = 1.3846;

 switch (altitude,temperature,pressure,density,speedOfSound){
 case ((altitude < 0) || (altitude > 260000)):
  altitude      = 0;
  temperature   = 0;
  pressure      = 0;
  density       = 0;
  speedOfSound = 0;

  return "Alert: Altitude must in the range of 0 and 260000 meters.";
 }

  if (isNaN (deltaT)){
      switch (deltaT){
        default:
        return 0;
  }
}
```

Figure C.27. Code for Neptune atmosphere function difference from original

```
 if (altitude === 0){
    temperature = temperatureatSurface;
  } else if ((altitude > 0) &&(altitude <= 16000)){
    temperature= ((altitude/1000) - 85.212)/(-1.267);
  } else if ((altitude > 16000) &&(altitude <= 26000)) {
    temperature = ((altitude/1000) - 159.44)/(-2.6283);
  } else if ((altitude > 26000) &&(altitude <= 47000)) {
    temperature = 49.73;
  } else if ((altitude > 47000) &&(altitude <= 71000)) {
    temperature = ((altitude/1000) + 53.317)/(2.1236);
  } else if ((altitude > 71000) &&(altitude <= 85000)) {
    temperature = ((altitude/1000) - 23.077)/(0.8351);
  } else if ((altitude > 85000) &&(altitude <= 108000)) {
    temperature = ((altitude/1000) - 31.004)/(0.687);
  } else if ((altitude > 108000) &&(altitude <= 147000)) {
    temperature = ((altitude/1000) + 304.8)/(3.6911);
  } else if ((altitude > 147000) &&(altitude <= 169000)) {
    temperature = ((altitude/1000) + 34.187)/(1.4867);
  } else if ((altitude > 169000) &&(altitude <= 196000)) {
    temperature = ((altitude/1000) + 200.1)/(2.7028);
  } else if ((altitude > 196000) &&(altitude <= 220000)) {
    temperature = ((altitude/1000) + 329.02)/(3.5917);
  } else{
    temperature = 157.9525;
  }
```

Figure C.28. Code for Neptune atmosphere equation difference from original

```
  pressure=pressureatSurface*Math.exp(-(gravity/(R*temperature)*altitude));
// Equation for speed of sound at given temperature
  speedOfSound = Math.sqrt(gamma * R * temperature);

// standard density equation uses sea level values of temperature and density with pressure and
// temperature at current altitude to find density at that altitude
  density      = pressure/(R*temperature);

// Adjusts the temperature variable with the Temperature Offset selection
  temperature  = temperature + deltaT;
}
```

Figure C.29. Code for pressure, speed of sound, and density difference from original

APPENDIX D

Atmosphere Data

The spreadsheet calculations for the Earth atmospheric data used in the website are shown here for validation purposes. Lutze (2014). The units are as follows: meters for the range, kelvin for the temperature, Pascal for the pressure, kg/m^3 for density, and meters per second for speed of sound

Table D.1. Earth atmosphere values from 0 to 30 kilometers. Lutze (2014).

Range	Temperature	Pressure	Density	Speed of Sound
0	288.16	101325	1.225000	3.40E+02
1000	281.66	89874.34	1.111639	3.36E+02
2000	275.16	79494.81	1.006483	3.33E+02
3000	268.66	70108.01	0.909113	3.29E+02
4000	262.16	61639.61	0.819118	3.25E+02
5000	255.66	54019.24	0.736103	3.21E+02
6000	249.16	47180.33	0.659684	3.16E+02
7000	242.66	41060.05	0.589487	3.12E+02
8000	236.16	35599.13	0.525153	3.08E+02
9000	229.66	30741.81	0.466334	3.04E+02
10000	223.16	26435.66	0.412693	2.99E+02
11000	216.66	22631.50	0.363905	2.95E+02
12000	216.66	19329.89	0.310816	2.95E+02
13000	216.66	16509.94	0.265473	2.95E+02
14000	216.66	14101.38	0.226744	2.95E+02
15000	216.66	12044.19	0.193665	2.95E+02
16000	216.66	10287.12	0.165412	2.95E+02
17000	216.66	8786.38	0.141281	2.95E+02
18000	216.66	7504.57	0.120670	2.95E+02
19000	216.66	6409.76	0.103066	2.95E+02
20000	216.66	5474.67	0.088030	2.95E+02
21000	217.66	4677.69	0.074870	2.96E+02
22000	218.66	3999.62	0.063724	2.96E+02
23000	219.66	3422.28	0.054277	2.97E+02
24000	220.66	2930.36	0.046265	2.98E+02
25000	221.66	2510.90	0.039464	2.98E+02
26000	222.66	2152.99	0.033686	2.99E+02
27000	223.66	1847.36	0.028775	3.00E+02
28000	224.66	1586.20	0.024597	3.00E+02
29000	225.66	1362.89	0.021041	3.01E+02
30000	226.66	1171.80	0.018011	3.02E+02

Table D.2. Earth atmosphere values from 31 to 60 kilometers. Lutze (2014).

Range	Temperature	Pressure	Density	Speed of Sound
30000	226.66	1171.80	0.018011	3.02E+02
31000	227.66	1008.17	0.015428	3.02E+02
32000	228.66	867.97	0.013224	3.03E+02
33000	231.46	748.18	0.011261	3.05E+02
34000	234.26	646.08	0.009608	3.07E+02
35000	237.06	558.89	0.008213	3.09E+02
36000	239.86	484.28	0.007034	3.10E+02
37000	242.66	420.34	0.006035	3.12E+02
38000	245.46	365.43	0.005186	3.14E+02
39000	248.26	318.20	0.004465	3.16E+02
40000	251.06	277.50	0.003851	3.18E+02
41000	253.86	242.38	0.003326	3.19E+02
42000	256.66	212.01	0.002878	3.21E+02
43000	259.46	185.72	0.002494	3.23E+02
44000	262.26	162.92	0.002164	3.25E+02
45000	265.06	143.12	0.001881	3.26E+02
46000	267.86	125.90	0.001637	3.28E+02
47000	270.66	110.90	0.001427	3.30E+02
48000	270.66	97.74	0.001258	3.30E+02
49000	270.66	86.15	0.001109	3.30E+02
50000	270.66	75.94	0.000977	3.30E+02
51000	270.66	66.93	0.000862	3.30E+02
52000	267.86	58.96	0.000767	3.28E+02
53000	265.06	51.86	0.000682	3.26E+02
54000	262.26	45.56	0.000605	3.25E+02
55000	259.46	39.97	0.000537	3.23E+02
56000	256.66	35.01	0.000475	3.21E+02
57000	253.86	30.62	0.000420	3.19E+02
58000	251.06	26.75	0.000371	3.18E+02
59000	248.26	23.33	0.000327	3.16E+02
60000	245.46	20.31	0.000288	3.14E+02

Table D.3. Earth atmosphere values from 61 to 84.852 kilometers. Lutze (2014).

Range	Temperature	Pressure	Density	Speed of Sound
61000	242.66	17.66	0.000254	3.12E+02
62000	239.86	15.33	0.000223	3.10E+02
63000	237.06	13.28	0.000195	3.09E+02
64000	234.26	11.49	0.000171	3.07E+02
65000	231.46	9.92	0.000149	3.05E+02
66000	228.66	8.55	0.000130	3.03E+02
67000	225.86	7.36	0.000113	3.01E+02
68000	223.06	6.32	0.000099	2.99E+02
69000	220.26	5.42	0.000086	2.98E+02
70000	217.46	4.63	0.000074	2.96E+02
71000	214.66	3.96	0.000064	2.94E+02
72000	212.66	3.37	0.000055	2.92E+02
73000	210.66	2.87	0.000047	2.91E+02
74000	208.66	2.44	0.000041	2.90E+02
75000	206.66	2.07	0.000035	2.88E+02
76000	204.66	1.75	0.000030	2.87E+02
77000	202.66	1.48	0.000025	2.85E+02
78000	200.66	1.25	0.000022	2.84E+02
79000	198.66	1.05	0.000018	2.83E+02
80000	196.66	0.89	0.000016	2.81E+02
81000	194.66	0.74	0.000013	2.80E+02
82000	192.66	0.62	0.000011	2.78E+02
83000	190.66	0.52	0.000010	2.77E+02
84000	188.66	0.44	0.000008	2.75E+02
84852	186.956	0.37	0.000007	2.74E+02

Below are the spreadsheet calculations for the other atmospheric data used in the websites. This data is the calculated data based on the graphical representations from NASA. Justus, et al (2007). The units are as follows: meters for the range, kelvin for the temperature, Pascal for the pressure, kg/m^3 for density, and meters per second for speed of sound.

Table D.4. Mars atmosphere values from 0 to 100 kilometers. Justus, et al (2007).

Range	Temperature	Pressure	Density	Speed of Sound
0	212.07	5.57E+02	1.89E-02	227.34
5000	208.21	3.51E+02	8.91E-03	225.26
10000	201.19	2.20E+02	5.80E-03	221.43
15000	194.18	1.39E+02	3.78E-03	217.54
20000	187.16	8.72E+01	2.47E-03	213.57
25000	180.15	5.48E+01	1.61E-03	209.53
30000	173.13	3.45E+01	1.05E-03	205.41
35000	166.11	2.17E+01	6.91E-04	201.20
40000	159.10	1.36E+01	4.54E-04	196.91
45000	152.08	8.58E+00	2.99E-04	192.52
50000	145.07	5.40E+00	1.97E-04	188.03
55000	146.29	3.39E+00	1.23E-04	188.81
60000	143.75	2.13E+00	7.86E-05	187.17
65000	141.22	1.34E+00	5.03E-05	185.52
70000	138.69	8.44E-01	3.22E-05	183.85
75000	137.45	5.31E-01	2.04E-05	183.02
80000	137.45	3.34E-01	1.29E-05	183.02
85000	137.45	2.10E-01	8.09E-06	183.02
90000	137.45	1.32E-01	5.09E-06	183.02
95000	137.45	8.31E-02	3.20E-06	183.02
100000	137.45	5.22E-02	2.01E-06	183.02

Table D.5. Mars atmosphere values from 100 to 200 kilometers. Justus, et al (2007).

Range	Temperature	Pressure	Density	Speed of Sound
100000	137.45	5.22E-02	2.01E-06	183.02
105000	141.49	3.29E-02	1.23E-06	185.69
110000	147.35	2.07E-02	7.42E-07	189.50
115000	153.21	1.30E-02	4.49E-07	193.23
120000	159.07	8.17E-03	2.72E-07	196.89
125000	164.93	5.14E-03	1.65E-07	200.49
130000	170.80	3.23E-03	1.00E-07	204.02
135000	200.94	2.03E-03	5.35E-08	221.29
140000	240.25	1.28E-03	2.82E-08	241.97
145000	263.64	8.04E-04	1.61E-08	253.48
150000	284.00	5.06E-04	9.42E-09	263.08
155000	299.35	3.18E-04	5.62E-09	270.10
160000	308.86	2.00E-04	3.43E-09	274.36
165000	318.37	1.26E-04	2.09E-09	278.55
170000	327.88	7.91E-05	1.28E-09	282.68
175000	331.94	4.97E-05	7.93E-10	284.42
180000	334.60	3.13E-05	4.95E-10	285.56
185000	337.26	1.97E-05	3.09E-10	286.69
190000	339.92	1.24E-05	1.93E-10	287.82
195000	342.59	7.78E-06	1.20E-10	288.95
200000	252.87	4.89E-06	1.02E-10	248.25

Table D.6. Titan atmosphere values from 0 to 100 kilometers. Justus, et al (2007).

Range	Temperature	Pressure	Density	Speed of Sound
0	87.39	1.48E+05	5.83E+00	187.32
5000	85.20	1.13E+05	4.58E+00	184.96
10000	81.32	8.68E+04	3.68E+00	180.70
15000	77.44	6.65E+04	2.96E+00	176.34
20000	73.57	5.09E+04	2.39E+00	171.87
25000	69.69	3.90E+04	1.93E+00	167.28
30000	70.16	2.99E+04	1.47E+00	167.85
35000	69.41	2.29E+04	1.14E+00	166.94
40000	68.74	1.76E+04	8.81E-01	166.14
45000	69.20	1.34E+04	6.70E-01	166.69
50000	70.13	1.03E+04	5.07E-01	167.81
55000	71.06	7.89E+03	3.83E-01	168.92
60000	74.82	6.05E+03	2.79E-01	173.32
65000	88.81	4.63E+03	1.80E-01	188.84
70000	102.81	3.55E+03	1.19E-01	203.18
75000	112.54	2.72E+03	8.34E-02	212.57
80000	120.86	2.08E+03	5.95E-02	220.30
85000	130.00	1.60E+03	4.24E-02	228.47
90000	133.16	1.22E+03	3.17E-02	231.23
95000	136.31	9.37E+02	2.37E-02	233.95
100000	139.47	7.18E+02	1.78E-02	236.65

Table D.7. Titan atmosphere values from 100 to 200 kilometers. Justus, et al (2007).

Range	Temperature	Pressure	Density	Speed of Sound
100000	139.47	7.18E+02	1.78E-02	236.65
105000	142.63	5.50E+02	1.33E-02	239.31
110000	145.97	4.22E+02	9.96E-03	242.10
115000	147.68	3.23E+02	7.54E-03	243.51
120000	149.39	2.48E+02	5.71E-03	244.91
125000	151.09	1.90E+02	4.33E-03	246.31
130000	152.80	1.45E+02	3.28E-03	247.70
135000	154.51	1.11E+02	2.48E-03	249.08
140000	156.21	8.53E+01	1.88E-03	250.45
145000	157.92	6.53E+01	1.43E-03	251.81
150000	159.63	5.01E+01	1.08E-03	253.17
155000	161.33	3.84E+01	8.20E-04	254.52
160000	163.47	2.94E+01	6.20E-04	256.20
165000	164.03	2.25E+01	4.73E-04	256.64
170000	164.59	1.73E+01	3.61E-04	257.08
175000	165.15	1.32E+01	2.76E-04	257.51
180000	165.71	1.01E+01	2.11E-04	257.95
185000	166.27	7.76E+00	1.61E-04	258.38
190000	166.83	5.95E+00	1.23E-04	258.82
195000	167.39	4.56E+00	9.38E-05	259.25
200000	167.95	3.49E+00	7.17E-05	259.69

Table D.8. Venus atmosphere values from 0 to 100 kilometers. Justus, et al (2007).

Range	Temperature	Pressure	Density	Speed of Sound
0	735.00	9.21E+06	6.63E+01	422.53
5000	688.97	6.69E+06	5.14E+01	409.08
10000	650.12	4.86E+06	3.96E+01	397.38
15000	611.27	3.53E+06	3.06E+01	385.32
20000	572.42	2.57E+06	2.37E+01	372.88
25000	533.57	1.87E+06	1.85E+01	360.00
30000	494.72	1.36E+06	1.45E+01	346.65
35000	455.87	9.85E+05	1.14E+01	332.76
40000	417.02	7.15E+05	9.08E+00	318.27
45000	378.17	5.20E+05	7.28E+00	303.08
50000	339.32	3.78E+05	5.89E+00	287.09
55000	300.47	2.74E+05	4.83E+00	270.15
60000	259.43	1.99E+05	4.07E+00	251.02
65000	243.87	1.45E+05	3.14E+00	243.38
70000	228.31	1.05E+05	2.44E+00	235.49
75000	212.76	7.65E+04	1.90E+00	227.33
80000	197.20	5.56E+04	1.49E+00	218.86
85000	181.64	4.04E+04	1.18E+00	210.05
90000	166.09	2.93E+04	9.35E-01	200.85
95000	165.94	2.13E+04	6.80E-01	200.76
100000	171.29	1.55E+04	4.79E-01	203.98

Table D.9. Venus atmosphere values from 100 to 200 kilometers. Justus, et al (2007).

Range	Temperature	Pressure	Density	Speed of Sound
100000	171.29	1.55E+04	4.79E-01	203.98
105000	144.13	1.13E+04	4.13E-01	187.11
110000	148.47	8.18E+03	2.92E-01	189.90
115000	152.80	5.94E+03	2.06E-01	192.65
120000	157.13	4.32E+03	1.45E-01	195.36
125000	161.46	3.14E+03	1.03E-01	198.04
130000	165.79	2.28E+03	7.28E-02	200.67
135000	170.13	1.66E+03	5.15E-02	203.28
140000	174.46	1.20E+03	3.65E-02	205.85
145000	178.79	8.74E+02	2.59E-02	208.39
150000	191.67	6.35E+02	1.75E-02	215.77
155000	204.85	4.62E+02	1.19E-02	223.07
160000	212.05	3.35E+02	8.37E-03	226.95
165000	216.72	2.44E+02	5.95E-03	229.43
170000	221.39	1.77E+02	4.23E-03	231.89
175000	222.61	1.29E+02	3.06E-03	232.53
180000	223.70	9.35E+01	2.21E-03	233.10
185000	224.78	6.79E+01	1.60E-03	233.66
190000	225.87	4.93E+01	1.16E-03	234.23
195000	226.95	3.59E+01	8.36E-04	234.79
200000	228.04	2.60E+01	6.05E-04	235.35

Table D.10. Neptune atmosphere values from 0 to 100 kilometers. Justus, et al (2007).

Range	Temperature	Pressure	Density	Speed of Sound
0	67.07	1.00E+08	4.12E+02	579.32
5000	63.31	7.95E+07	3.47E+02	562.84
10000	59.36	6.31E+07	2.94E+02	545.02
15000	55.42	5.02E+07	2.50E+02	526.59
20000	53.05	3.99E+07	2.08E+02	515.24
25000	51.15	3.17E+07	1.71E+02	505.92
30000	49.73	2.52E+07	1.40E+02	498.84
35000	49.73	2.00E+07	1.11E+02	498.84
40000	49.73	1.59E+07	8.84E+01	498.84
45000	49.73	1.26E+07	7.02E+01	498.84
50000	48.65	1.00E+07	5.70E+01	493.41
55000	51.01	7.97E+06	4.32E+01	505.21
60000	53.36	6.33E+06	3.28E+01	516.73
65000	55.72	5.03E+06	2.50E+01	528.01
70000	58.07	4.00E+06	1.90E+01	539.05
75000	62.18	3.18E+06	1.41E+01	557.79
80000	68.16	2.52E+06	1.02E+01	584.02
85000	74.15	2.01E+06	7.48E+00	609.13
90000	85.87	1.59E+06	5.13E+00	655.52
95000	93.15	1.27E+06	3.76E+00	682.74
100000	100.43	1.01E+06	2.77E+00	708.91

Table D.11. Neptune atmosphere values from 100 to 200 kilometers. Justus, et al (2007).

Range	Temperature	Pressure	Density	Speed of Sound
100000	100.43	1.01E+06	2.77E+00	708.91
105000	107.71	8.00E+05	2.05E+00	734.15
110000	112.38	6.35E+05	1.56E+00	749.89
115000	113.73	5.05E+05	1.23E+00	754.40
120000	115.09	4.01E+05	9.64E-01	758.88
125000	116.44	3.19E+05	7.57E-01	763.33
130000	117.80	2.53E+05	5.95E-01	767.76
135000	119.15	2.01E+05	4.67E-01	772.16
140000	120.51	1.60E+05	3.67E-01	776.53
145000	121.86	1.27E+05	2.88E-01	780.89
150000	123.89	1.01E+05	2.25E-01	787.36
155000	127.25	8.02E+04	1.74E-01	797.98
160000	130.62	6.37E+04	1.35E-01	808.45
165000	133.98	5.06E+04	1.05E-01	818.79
170000	136.93	4.02E+04	8.13E-02	827.77
175000	138.78	3.20E+04	6.37E-02	833.34
180000	140.63	2.54E+04	5.00E-02	838.88
185000	142.48	2.02E+04	3.92E-02	844.38
190000	144.33	1.60E+04	3.07E-02	849.84
195000	146.18	1.27E+04	2.41E-02	855.27
200000	147.29	1.01E+04	1.90E-02	858.50