**Tapered Grain Geometry and Statistical Learning for Solid Rocket Motor Simulation**

by

Tyler M. Sheils

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
May 5, 2023

Keywords: solid rocket motor, solid rocket grain, tapered grain geometry, statistical learning, neural networks, simulation



Approved by

Roy Hartfield, Chair, Woltosz Professor, Aerospace Engineering
Mark Carpenter, Co-Chair, Professor, Mathematics and Statistics
Joseph Majdalani, Professor and Francis Chair of Excellence, Aerospace Engineering

Abstract

This thesis investigates applying statistical learning techniques to a tapered grain solid rocket motor simulation. Tapered grain solid rocket motors (SRMs) have application in both defense and space industries. Tapered grain geometries offer an alternative to complex cross sections to control the thrust profile of the solid rocket motor. New analytical methods were developed to accurately model tapered solid rocket motor grain geometries. A tapered grain solid rocket motor internal ballistics code was developed in FORTRAN using Lagrangian grain regression assumptions, 1D flow assumptions, and new analytical methods developed as part of this work. This code can accurately model the internal ballistics of tapered grain motors, specifically for circular perforated and star grain geometries. This thesis will explore the development of analytical equations for tapered grains, the implementation into a code, and accompanying machine learning techniques and results. The SRM internal ballistics code was used to develop large databases for statistical learning. The SRM code contains a Monte Carlo simulation using a Latin Hypercube distribution that allows the user to robustly generate a multitude of SRM designs, and the resultant thrust-time profile based on desired inputs. Once large databases of performance data were generated, statistical learning methods such as regression analysis and neural networks were used to provide regression analysis and surrogate modeling capabilities.

Acknowledgments

I would like to thank my advisor Dr. Roy Hartfield for his support and guidance over the course of my research and graduate studies. I would also like to thank Dr. Mark Carpenter for the knowledge he has provided regarding statistical learning and applied statistics. I am also thankful for the help that Dr. Noel Cervantes (former graduate student) provided me throughout my research experience. Noel's knowledge of FORTRAN and Python machine learning libraries helped me in many ways. I would also like to thank Dr. Joseph Majdalani for being a member on my thesis committee.

I would like to thank my parents Tobie and Robin Sheils for their continuous support of all that I work towards. I also would like to thank my brothers, Connor and Trey, for their support and friendship as I have continued on in school. I would also like to thank Ella Guven for her support as I have worked towards completing my graduate studies. I am also thankful to all my friends who have supported me throughout my time in school. Above all things I finally want to give thanks to God for the blessings in my life, and for the ability to do this work.

*Ora et Labora*

Table of Contents

List of Figures

List of Tables

List of Abbreviations

| | | |
|---|---|---|
| $a$ | = | burn rate coefficient |
| $A_b$ | = | burn area |
| $A_e$ | = | exit area |
| $A_p$ | = | port area |
| $A_r$ | = | quadrilateral area |
| $A^*$ | = | nozzle throat area |
| $CP$ | = | circular perforated |
| $c^*$ | = | characteristic velocity |
| $\epsilon$ | = | angular fraction |
| $f$ | = | fillet radius |
| $F_t$ | = | thrust |
| $GL$ | = | grain length |
| $I$ | = | total impulse |
| $\lambda$ | = | nozzle correction factor |
| $\dot{m}$ | = | mass flow rate |
| $n$ | = | pressure exponent |
| $n_{dx}$ | = | number of Coordinate points per section |
| $n_{sp}$ | = | number of star points |
| $P_a$ | = | ambient pressure |
| $P_c$ | = | chamber pressure |

| | | |
|---|---|---|
| $P_e$ | = | nozzle exit pressure |
| $r$ | = | burn rate |
| $\rho_b$ | = | propellant density |
| $R_i$ | = | inner grain radius |
| $R_p$ | = | propellant grain radius |
| $R_o$ | = | outer grain radius |
| $S$ | = | burn perimeter |
| $SHAP$ | = | Shapley Additive exPlanations |
| $SRM$ | = | solid rocket motor |
| $t_b$ | = | burn time |
| $\theta$ | = | star point angle |
| $u_e$ | = | nozzle exit velocity |
| $web$ | = | total web thickness |
| $web_1$ | = | phase I web thickness |
| $web_2$ | = | phase II web thickness |
| $y$ | = | burn distance |

Subscripts

| | | |
|---|---|---|
| $a$ | = | aft end parameter |
| $f$ | = | forward end parameter |

# Chapter 1

# Introduction

Solid rocket motors have been used extensively both in space and military applications. Larger solid rocket motors such as the ASRM [1] and others mentioned in Reference [2] have been developed for space applications. These large solid rocket motors produce a relatively high amount of thrust used to propel launch vehicles into space. For military applications, solid rocket motors have been used to propel missiles, rockets, and anti-tank weapons [3, 4, 5, 6, 7, 8]. The solid rocket motor allows the payload to be accelerated at a high speed towards a target, or desired location. An integral part of the design process of solid rocket motors is choosing the correct grain design. The grain design of the solid rocket motor affects the burn-back characteristics of the motor. These characteristics effect the thrust-time profile of the motor, as well as the specific impulse, chamber pressure, and overall performance of the flight vehicle being propelled by the solid rocket motors. CP grains tend to burn in a more progressive manner, while the star grain can have regressive, neutral, and progressive burn phases.

The grain design of a solid rocket motor can be tailored to fit the requirements that the solid rocket motor must meet. The Space Shuttle solid rocket booster thrust profile is a relevant example [9] of a specific thrust profile that must be met. Optimization schemes such as genetic algorithms [10, 11] and particle swarm methods [12, 13, 14] have been developed to help optimize the performance of solid rocket motors given design constraints. In previous efforts, level set methods were also developed [15] to model grain geometries such as circular perforated, star, and wagon wheel grain designs [16, 17]. The majority of these works were focused on straight SRM grains. Oftentimes, the solid rocket motor grain has multiple sections with unique grain designs. In practice, these grain sections can be either straight, or tapered [18].

Solid rocket motors that make use of tapered grains include, and are not limited to the Titan IV, Ariane, Castor, along with other solid rocket boosters [19]. These tapered grains are used to control the thrust-time profile, while also limiting erosive burning [19, 20, 21]. For this reason, it is important that an internal ballistics tool be developed capable of modeling this feature of SRMs.

To accurately model tapered grain geometries, new analytical methods were developed to accurately model the tapered SRM grain. These methods make use of the designs proposed by Barrere [22] and Hartfield et al. [16, 17]. The methods proposed in this thesis offer an alternative to the methods developed by Ricciardi for tapered grains [23]. Once developed, these analytical methods were integrated into a 1D internal ballistics code to produce SRM performance data. Star grains and circular perforated (CP) grain designs are supported with this internal ballistics code. Building from legacy code that had been previously developed [11, 16, 17, 24, 25, 26, 27] this tapered solid rocket motor code could successful be developed. During the process of development, a Monte Carlo [28, 29, 30] scheme was integrated into the internal ballistics code so that large databases of SRM performance data could be produced. This Monte Carlo scheme allows the user of this 1D internal ballistics code to generate randomly sampled SRM designs, provided a range of inputs. The output of interest for this study were the thrust-time curves of a solid rocket motor. Figure 1.1 shows a sample thrust vs. time curve for a solid rocket motor.

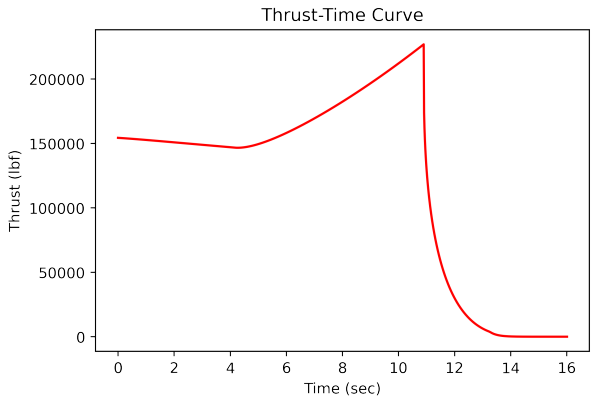



Figure 1.1: Sample Thrust vs. Time Curve

With large databases of thrust-time curves generated for both star grain and circular perforated grain designs, machine learning techniques could then be implemented to help analyze and model the thrust-time profiles, along with modeling SRM performance parameters. The two basic uses of machine learning for this effort were to first to be used to analyze the solid rocket motor data via regression analysis. Both traditional linear regression techniques and neural networks were used to perform this analysis. Once the regression analysis had been performed, neural networks were trained to act as surrogate models for the internal ballistics code. These networks are trained to predict, or produce, the thrust-time profiles that are generated with the internal ballistics code. Python packages such as SHAP [31] were used to help understand these machine learning models, specifically for the regression analysis. Finally, these neural networks could be tuned [32] to find the optimal hyper-parameters used to create the optimal neural networks.

In summary, the goals of this thesis are as follows: (1) develop and integrate methods to accurately model tapered SRM grain designs into an internal ballistics tool, (2) generate large databases of thrust-time data to be used for analysis, and (3) use machine learning techniques to analyze and model the solid rocket motor performance data. These three main goals were accomplished and can be further explained in the upcoming chapters of this thesis.

Chapter 2

Physics Modeling of Solid Rocket Motors

It is important to first understand the physics needed to properly model the internal ballistics of a solid propellant rocket. To solve this internal ballistics problem, it is imperative that the burn area of the solid rocket motor is properly modeled. To produce outputs such as thrust and chamber pressure, the burn area first must be modeled. Before considering any advances in grain geometry let us consider the schematic shown below, to get an understanding of how the thrust equation for solid rockets is developed.

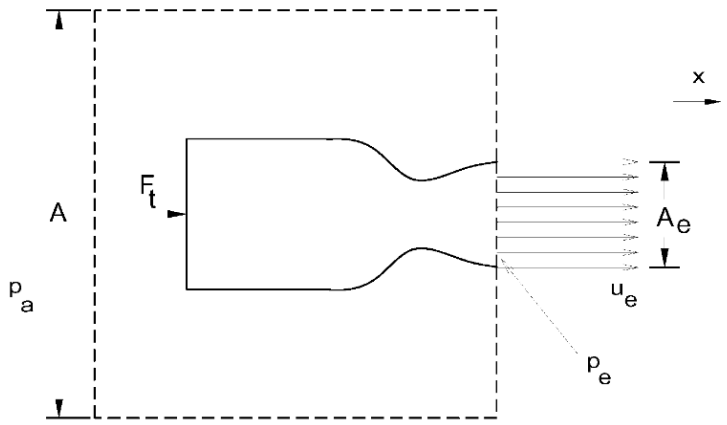



Figure 2.1: Solid Rocket Motor Control Volume [33]

The control volume above is important to understand when developing the equations for thrust of a solid rocket motor. The solid rocket motor exhausts into the ambient atmosphere surrounding the rocket body. This ambient pressure changes based on the location of the rocket in the atmosphere. The internal pressure is known as the chamber pressure of the solid rocket motor. Looking at the control volume shown in Fig. 2.1 we can see that the only place where

we have fluid exiting the control volume is at the exit plane of the nozzle. To solve this problem, we will apply the momentum equation, shown in Eq. 2.1.

$$\frac{d}{dt}\iiint_V \rho\vec{u}dV + \iint_S \vec{u}(\rho\vec{u}\cdot\vec{n})dA = \sum F \tag{2.1}$$

The forces that we consider is this solution are the external pressure forces and the thrust. This simplification can be seen in Eq. 2.2

$$\frac{d}{dt}\iiint_V \rho\vec{u}dV + \iint_S \vec{u}(\rho\vec{u}\cdot\vec{n})dA = F_t + F_{pressure} \tag{2.2}$$

With our external forces defined, we now need to begin to apply the assumptions of this problem. In applying the conservation of momentum above, we assume that we are dealing with 1-D steady flow. This assumption allows us to drop the time dependent term, resulting in the following form of Eq. 2.1.

$$\iint_S \vec{u}(\rho\vec{u}\cdot\vec{n})dA = F_t + F_{pressure} \tag{2.3}$$

We can also expand the pressure force term shown on the right hand side of Equation 2.3. Equation 2.4 shows the resultant pressure forces that act on the control volume.

$$F_{pressure} = P_a A_e - P_e A_e \tag{2.4}$$

The final step in this analysis is to apply the surface integral term of Equation 2.3 to the exit plane of the solid rocket motor nozzle. When applying the Equation 2.3 and applying what we know from Equation 2.4 we obtain the following simplified equation.

$$F_t = \rho_e A_e u_e^2 + A_e(P_e - P_a) \tag{2.5}$$

Simplifying further using mass conservation we can obtain the commonly used thrust equation shown in Equation 2.6.

$$F_t = \dot{m}u_e + A_e(P_e - P_a) \tag{2.6}$$

Equation 2.6 shows the basic version of the thrust equation that adequately models the thrust produced by a solid rocket motor. When modeling the solid rocket motor, the most important parameter to model is the burn area. The burn area is directly related to the chamber pressure of the rocket, which in turn effects the thrust produced by the solid rocket motor. Equations 2.7 and 2.8 show the relationship between the burn area ($A_b$) and chamber pressure ($P_c$) having a direct impact on the thrust. Modeling burn area as a function of time leads to results of chamber pressure and thrust as a function of time.

$$P_c = (\frac{A_b}{A^*} a \rho_b c^*)^{\frac{1}{1-n}} \tag{2.7}$$

$$F_t = \lambda(P_c A^* C_f) + A_e P_e \tag{2.8}$$

In order to model the burn area, we have to make assumptions about the burn rate. The tool developed for this work makes 1-D flow assumptions, and uses a bulk burn rate model. Equation 2.9 shows how the burn rate ($r$) was modeled with this tool.

$$r = aP_c^n \tag{2.9}$$

In Equation 2.9, $a$ represents the burn rate coefficient, $P_c$ represents the chamber pressure, and $n$ represents the pressure exponent. Both the burn rate coefficient and the pressure exponent are unique to the solid rocket fuel chosen to be modeled with the internal ballistics tool. Sutton provides a good overview of these values for typical solid propellants [34]. Now that we have derived the equations needed to develop thrust of the solid rocket motor, we can see what is needed to obtain a solution. With the basics developed, a more detailed explanation can be provided for the development of grain geometry equations. Sections 2.2, 2.3, and 2.5 provide the equations and analytical methods used to model the burn area for the SRM.

Accurately modeling the burn area of a solid rocket motor is not a trivial task. Effects such as ignition, along with 2-D and 3-D flow effects are not modeled in this thesis. For more detailed modeling of solid rocket motor performance, CFD codes may need to be used. Tools such as FlightStream® also exist that have been shown to be capable of modeling the internal ballistics of solid rocket motors [35, 36]. For this thesis we will be working with a 1-D internal ballistics code using a bulk burn rate model and uniform grain regression assumptions [16, 17].

## 2.1 Performance Metrics and Equations

With equations for thrust and chamber pressure developed, it is important to introduce some of the performance metrics used for this thesis. The first two metrics we will introduce are the maximum thrust and the average thrust of the solid rocket motor. Once the thrust-time curve has been generated, the maximum value and average value can easily be calculated. Simple Python functions have been developed and implemented to use the output data files from the solver to calculate the maximum thrust and average thrust for each solid rocket motor thrust-time curve.

The final two performance metrics we will analyze are the burn time and the total impulse of the solid rocket motor. For the work done in this thesis, the burn time will be defined as the total time that the solid rocket motor is burning. Other sources such as Sutton [34] may define this metric in different ways. The total impulse is essentially the area under the thrust-time curve. Equation 2.10 and 2.11 show the important equations used for the calculation of total impulse.

$$I = \int_0^{t_b} F_t \, dt \tag{2.10}$$

$$I = \bar{F}_t t_b \tag{2.11}$$

In Equation 2.11 $I$ represents the total impulse, $t_b$ represents the burn time, and $\bar{F}_t$ represents the average thrust produced by the SRM. Equation 2.10 could be applied to a thrust-time curve like the one seen in Fig. 1.1 to see that all it is calculating is the area under the curve. The SRM performance metrics used for this thesis can be summarized below in Table 2.1.

Table 2.1: Solid Rocket Motor Performance Metrics

| Performance Metric | Name | Units |
|---|---|---|
| Maximum Thrust | MAX_THRUST | $lb_f$ |
| Average Thrust | AVG_THRUST | $lb_f$ |
| Burn Time | BURN_TIME | $seconds$ |
| Total Impulse | TOTAL_IMPULSE | $lb_f - sec$ |

## 2.2 Star Grain Equations

As mentioned above, the star grain geometry design follows the work of Barrere [22] and Hartfield et al. [16, 17]. Figure 2.2 provides a schematic of the star grain geometry used for this work. A review of this method with the addition of short spoke wagon wheel designs can be found in Refs. [16, 17].

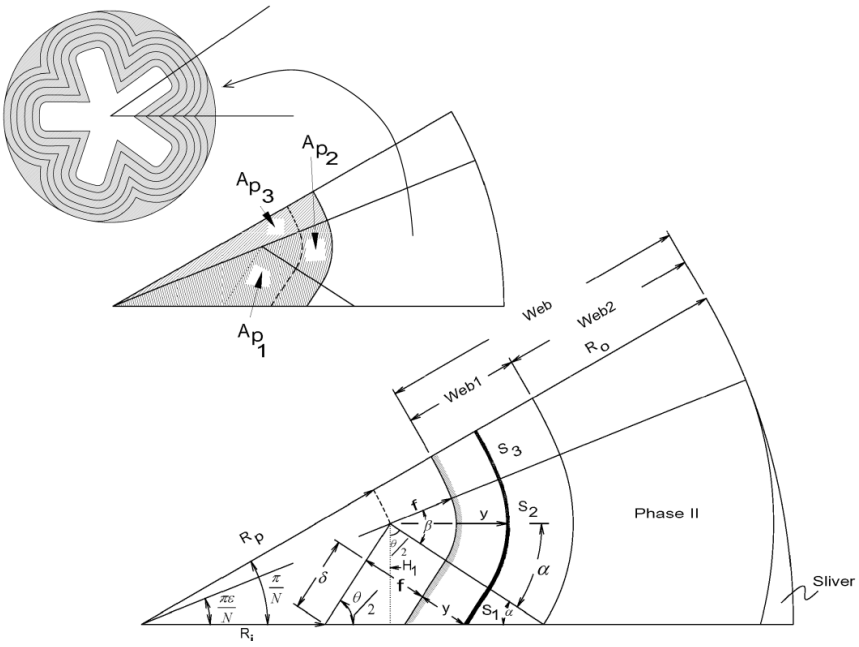



Figure 2.2: Star Grain Geometry

The port area and more importantly the burn area equations can be developed by taking advantage of the geometry provided in Figure 2.2. When calculating the burn areas, we will establish two unique geometries, Phase I and II, the first phase takes place while the burn distance ($y$) is less that first web thickness ($web_1$), the second burning phase begins when the burn distance is greater than the first web thickness ($web_1$). To calculate the burn area, the

burn perimeter must be calculated given $S_1$, $S_2$, and $S_3$. First for phase I, the following burn perimeter equations are developed [17].

$$S_1 = \frac{H_1}{\sin\frac{\theta}{2}} - (y+f)\cot\frac{\theta}{2} \tag{2.12}$$

$$S_2 = (y+f)(\frac{\pi}{2} - \frac{\theta}{2} + \frac{\pi\epsilon}{n_{sp}}) \tag{2.13}$$

$$S_3 = (R_p + y + f)(\frac{\pi}{n_{sp}} - \frac{\pi\epsilon}{n_{sp}}) \tag{2.14}$$

In Equations 2.12, 2.13, 2.14 the values for $H_1$ and $\theta/2$ are derived in the following two equations.

$$H_1 = R_p \sin\frac{\epsilon\pi}{n_{sp}} \tag{2.15}$$

$$\frac{\theta}{2} = \arctan\frac{H_1\tan\frac{\epsilon\pi}{n_{sp}}}{H_1 - R_i\tan\frac{\epsilon\pi}{n_{sp}}} \tag{2.16}$$

With these equations developed, the total burn perimeter can now be solved for along with the phase I burn area.

$$S = 2(n_{sp})(S_1 + S_2 + S_3) \tag{2.17}$$

$$A_b = S(GL) \tag{2.18}$$

Now that the phase I star grain burn area equations have been developed, the port area equations for phase I can similarly be developed. Looking at Figure 2.2, the the port area sections are labeled as $A_{p_1}$, $A_{p_2}$, and $A_{p_3}$. The following four equations are used to find the total port area.

$$A_{p_1} = \frac{1}{2}H_1[R_p\cos\frac{\epsilon\pi}{n_{sp}} + H_1\tan\frac{\theta}{2}] - \frac{1}{2}S_1^2\tan\frac{\theta}{2} \tag{2.19}$$

$$A_{p_2} = \frac{1}{2}(y+f)^2(\frac{\pi}{2} - \frac{\theta}{2} + \frac{\epsilon\pi}{n_{sp}}) \tag{2.20}$$

$$A_{p_3} = \frac{1}{2}(R_p + y + f)^2(\frac{\pi}{n_{sp}} - \frac{\epsilon\pi}{n_{sp}}) \tag{2.21}$$

$$A_p = 2(n_{sp})(A_{p_1} + A_{p_2} + A_{p_3}) \tag{2.22}$$

The phase II burn equations can be simplified from the phase I equations because the $S_1$ perimeter is burnt out when using the phase II geometry. The logic to switch between phase I and phase II geometry is when the burn distance is greater than the phase I web thickness. The equation for the phase I web thickness ($web_1$) can be seen below in 2.23. The equations for the total web thickness and phase II web thickness can be seen below as well in Equations 2.24 and 2.25.

$$web_1 = \frac{H_1}{\cos\theta/2} - f \tag{2.23}$$

$$web = R_o - R_p - f \tag{2.24}$$

$$web_2 = web - web_1 \tag{2.25}$$

Now that we have distinguished between the web thicknesses, two new parameters $\beta$ and $\gamma$ are defined to simplify the arithmetic for the phase II equations. Figure 2.3 shows a schematic of the phase II burn geometry with $\gamma$ included. Figure 2.3 shows that as the burn distance increases, the value for $\gamma$ changes.

$$\beta = (\frac{\pi}{2} - \frac{\theta}{2} + \frac{\epsilon\pi}{n_{sp}}) \tag{2.26}$$

$$\gamma = \arctan\frac{\sqrt{(y+f)^2 - H_1^2}}{H_1} - \frac{\theta}{2} \tag{2.27}$$

Now that these new parameters have been defined, the phase II burn area equations can be developed as follows.

$$S_2 = (y+f)(\beta - \gamma) \tag{2.28}$$

$$S_3 = (R_p + y + f)(\frac{\pi}{n_{sp}} - \frac{\epsilon\pi}{n_{sp}}) \tag{2.29}$$

$$S = 2(n_{sp})(S_2 + S_3) \tag{2.30}$$

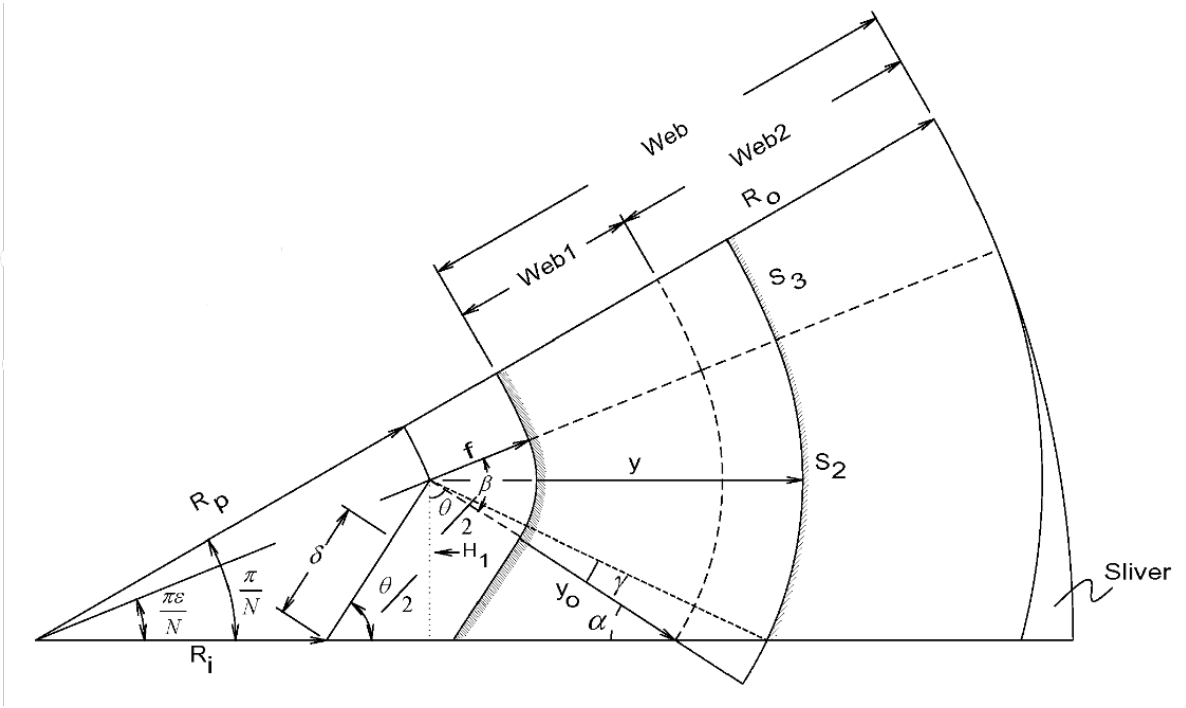



Figure 2.3: Phase II Geometric Design

Equation 2.18 can now be applied again to obtain the burn area for the phase II burn. Finally the port areas can be defined for the phase II burn with these three remaining equations.

$$A_{p_1} = \frac{1}{2} H_1 [R_p \cos \frac{\epsilon\pi}{n_{sp}} + \sqrt{(y+f)^2 - H_1^2}] \tag{2.31}$$

$$A_{p_2} = \frac{1}{2}(y+f)^2(\beta - \gamma) \tag{2.32}$$

$$A_{p_3} = \frac{1}{2}(R_p + y + f)^2(\frac{\pi}{n_{sp}} - \frac{\epsilon\pi}{n_{sp}}) \tag{2.33}$$

Equation 2.22 can now be applied to the phase II equations to find the total port area for phase II burning. These equations are applied until the maximum value of $y$ , $ymax$, is reached. Equation 2.34 shows the equation defining the maximum value of burn distance.

$$ymax = \sqrt{(R_o - R_p \cos \frac{\pi\epsilon}{n_{sp}})^2 + H_1^2} - f \tag{2.34}$$

After the value of $ymax$ is reached, the SRM burn enters into the tail-off period. The tail-off is modeled used a exponential decay that resembles $Ce^{-\alpha t}$, where $C$ and $\alpha$ are constants [33]. The grain design equations developed above are implemented into the solid rocket code

and propagated through time. If the burn area can be accurately modeled, the thrust and the pressure time curves can be developed from the burn area propagation.

## 2.3 Circular Perforated Grain Equations

The second geometry that is modeled with the tapered grain solid rocket code is the circular perforated or CP grain design. The code models the CP grain as a special case of the star grain when the angular fraction ($\epsilon$) is very small. A good value to use for the angular fraction is about 0.01 in the snglerun.dat file. Figure 2.4 shows the schematic of the important design parameters used when modeling a CP grain.

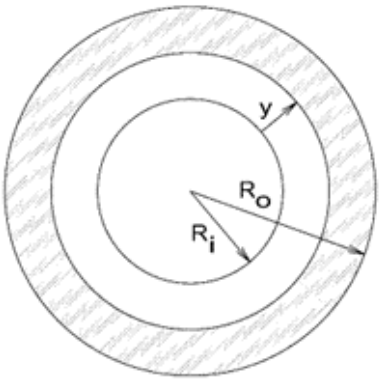



Figure 2.4: Schematic of a Circular Perforated Grain

The burn area equations can be easily developed using the geometric relationships seen in Figure 2.4. The burn area can be calculated as the following.

$$A_b = 2\pi(R_i + y)(GL) \tag{2.35}$$

The port area for the circular perforated grain can also be calculated as follows.

$$A_p = \pi(R_i + y)^2 \tag{2.36}$$

Now that the grain design equations for both star and CP grains have been developed, it is important to cover the logic used for extending these designs to support tapered solid rocket motor grains.

## 2.4 Tapered Grain Implementation

This section will cover the work done to extend the methods developed by Barrere [22] and Hartfield et al. [16, 17] for tapered grain solid rocket motors. Ricciardi proposed a method for tapering by adding longitudinal grain sections together [23]. The method that will be covered in this thesis is an alternative method to what is developed by Ricciardi [23]. Due to the taper of the solid rocket motor grain, new phases of the burn needed to be developed from what had been previously developed in References [16, 17, 22].

For this thesis, only a positive taper will be considered. This is the prevalent kind of tapering in solid rocket motor grain design. By positive taper, we are saying that the hold in the solid rocket motor grain grows towards the aft end of the grain section. The equations developed for this work allow for the tapering of five geometric grain parameters. Table 2.2 show the grain parameters that can be modified with the proposed method.

Table 2.2: Tapered Grain Parameters and Description

| Parameter | Description |
|---|---|
| $R_i$ | Inner Grain Radius |
| $R_o$ | Outer Grain Radius |
| $R_p$ | Propellant Grain Radius |
| $f$ | Fillet Radius |
| $\epsilon$ | Angular fraction |

The taper ratio is implemented into the 1D internal ballistic code using the following method. The user can define the taper ratio for any of the parameters shown above in Table 2.2. The taper ratio represents the percent increase of the geometric parameter of choice (see Table 2.2). Equation 2.37 shows the basic equation used to calculate the forward and aft end parameter sizes.

$$P_a = P_f + P_f * TR \tag{2.37}$$

In Equation 2.37, the $TR$ represents the user defined taper ratio, $P_a$ represents the aft end parameter, and $P_f$ represents the forward end parameter. The forward or aft end parameter can be any of the parameters described in Table 2.2. Typically the $R_p$ and $R_i$ are the parameters that

are tapered for this work. The analytical methods developed as part of this work assume linear taper between the forward and aft ends and only models positive tapering. The positive taper is commonly seen in space and defense solid rocket motors, along with being the prevalent taper design seen in the literature. It is important to note that when applying the taper, the design at the forward and aft end must be a CP or Barrere star [22] design. Depending on the forward end star grain, it is possible that certain taper ratios lead to a non-viable aft end star grain design. With this covered, we can now move into the methods implemented to model the tapered solid rocket motor grain. These analytical methods build on the work of Barrere [22] and Hartfield et al. [16, 17]. New burn phases and analytical methods for tapered solid rocket motor grains were developed as part of this thesis.

## 2.5 Analytical Methods for Tapered Grain Solid Rocket Motors

To model the tapered grain solid rocket motor, the basic grain design proposed by Barrere [22] is considered. An image of this grain design can be shown in Figure 2.2. We will only consider the star grain geometry in this derivation since the CP grain is a special case of the star grain when the angular fraction ($\epsilon$) is small. The grain parameters shown in Table 2.2 are defined at the forward end of the solid rocket motor grain, a taper ratio is then defines that allows the parameters to be defined at the aft end as well.

The burn perimeters $S_1$, $S_2$, and $S_3$ will first be discretized into Cartesian coordinates at the forward and aft end of the solid rocket motor grain. To accomplish this, five coordinates are defined in the XY plane of the solid rocket motor, the origin of this frame can be seen in Figure 2.5. These coordinate points make the discretization of the burn perimeters simple and easy to implement into a program. The Z component is essentially defining the grain length, at the forward end the Z component will be 0 and at the aft end it will be equal to the grain length. Eventually when the star points begin to reach the wall the Z component will need to be decremented for each coordinate, more will follow about that logic in the following sections. Figure 2.5 shows an image of the star grain with the five way points included.

To develop these Cartesian coordinates, the following equations are used to find points 1-5.

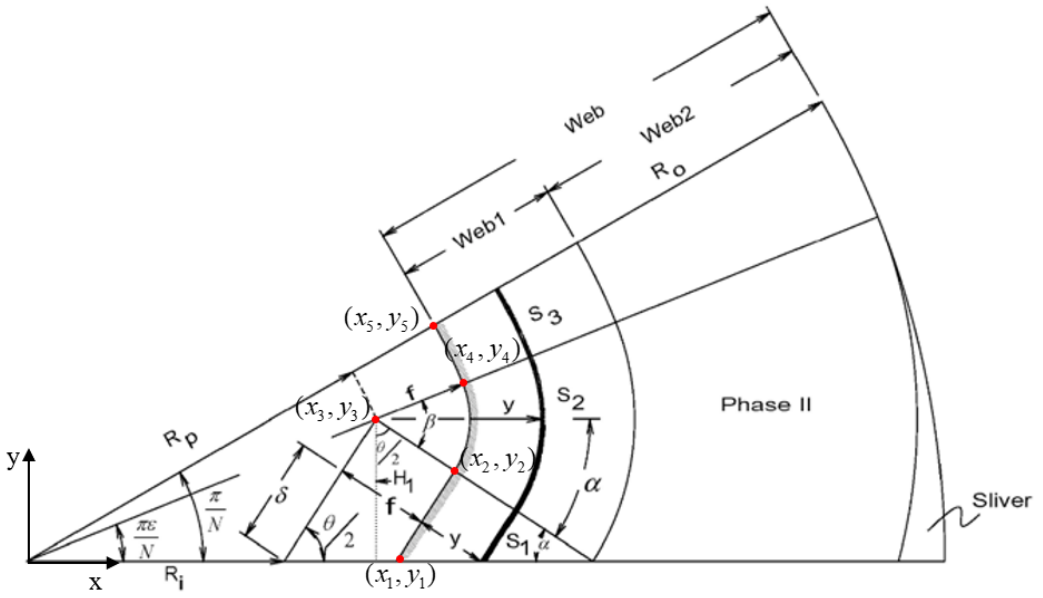



Figure 2.5: Star Grain Design with five supplementary points

$$(x_1, y_1) = (R_i + \frac{f}{\sin\theta/2}, 0) \tag{2.38}$$

$$(x_2, y_2) = (x_1 + S_1 \cos\theta/2, S_1 \sin\theta/2) \tag{2.39}$$

$$(x_3, y_3) = (R_p \cos\frac{\epsilon\pi}{n_{sp}}, R_p \sin\frac{\epsilon\pi}{n_{sp}}) \tag{2.40}$$

$$(x_4, y_4) = (x_3 + f\cos\frac{\epsilon\pi}{n_{sp}}, y_3 + f\sin\frac{\epsilon\pi}{n_{sp}}) \tag{2.41}$$

$$(x_5, y_5) = ((R_p + f)\cos\frac{\pi}{n_{sp}}, (R_p + f)\sin\frac{\pi}{n_{sp}}) \tag{2.42}$$

Now, with Equations 2.38-2.42 developed we can begin to explain how the burn perimeters were discretized. The total burn perimeter is discretized with the same number of points on the forward and aft end of the solid rocket motor grain. This will ensure an equal number of points that can be used to eventually calculate the burn area during the different burn phases SRM simulation.

### 2.5.1 Burn Area Implementation

The goal of breaking up the burn perimeters into Cartesian coordinates is to be able to calculate the burn area of small panels, defined by four points in 3D Cartesian space, and then sum those areas up to find the burn area of the solid rocket motor. This method makes use of the fact that there are the same number of coordinates on the forward end and aft end of the SRM grain. Knowing each point and the distance between them, small differential areas can be created and then added together to calculate the burn area. Figure 2.6 shows an image of the aft and forward end half star point with lines connecting them. These lines help show the differential areas $dAB's$ that are calculated and used to find the burn area of the SRM.

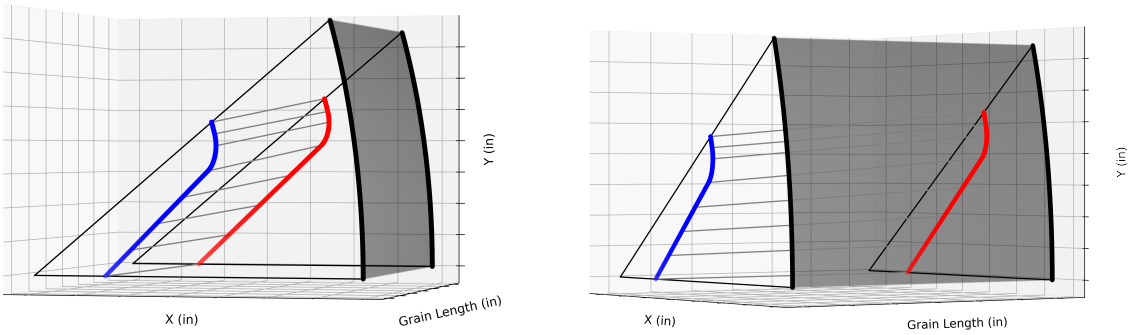



Figure 2.6: Image of Half Star Point

Figure 2.6 shows image of the discretized aft and forward end half star points. The red lines represent the aft end, the blue lines represent the forward end, and the shaded in section represents the outer wall of the solid rocket motor grain. In this case the SRM is tapered, so the aft end (red) star point is slightly larger than the forward end star point. Figure 2.7 helps show the difference in size of the star points. The aft end star in Figure 2.7 is larger than the forward end star, this represents a positive tapering of the solid rocket motor. Figure 2.6 is essentially showing a discretized 3D image of Figure 2.5. This 3D image shown in Figure 2.6, is what this internal ballistics code is modeling as a function of burn distance and time.

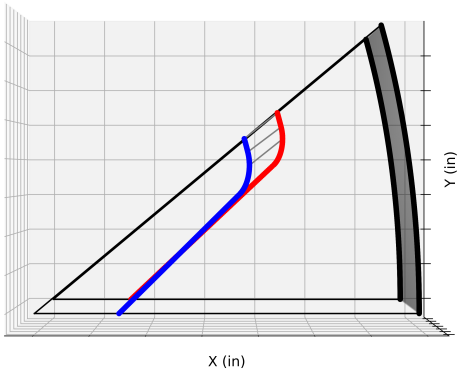



Figure 2.7: Tapered Grain Star Point Comparison

Now we can connect each of the points to form quadrilateral shapes that represent $dAB's$ of burn area, these $dAB's$ can be seen in Figure 2.6 by looking at the grey lines connecting the forward and aft end grain faces. By discretizing the burn perimeter with more points we can more closely approximate the curved surface of the star grain with $dAB's$. Figure 2.6 shows only a few $dAB's$, when in reality there are typically more than a hundred $dAB's$ that represent one half of the star point. Once the $dAB's$ have been created, we can split them into two triangles per rectangular shape ($dAB$), similar to what it shown in Figure 2.8.

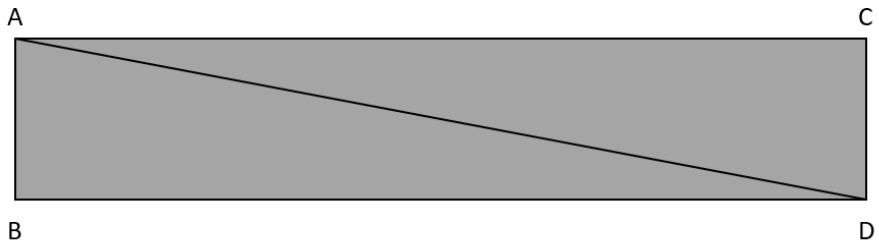



Figure 2.8: Schematic of a Rectangle/Quadrilateral Used to Calculate Burn Area

Now, the area of this quadrilateral, $A_r$, can be solved for as follows by using the cross product. Equation 2.43 shows the implementation of the equation mentioned above.

$$A_r = \frac{1}{2}|\vec{AC} \times \vec{AD}| + \frac{1}{2}|\vec{DA} \times \vec{DB}| = dAB \tag{2.43}$$

In practice, the vectors $\vec{AC}$ and $\vec{BD}$ may not be the same length. When the aft end begins to burn out, these lengths will change between each coordinate point. More information on that will be introduced in the following sections, specifically the section covering the final burn phase, Phase D. With the logic to find the area for one quadrilateral shape introduced, we can see how burn area is calculated as follows in Equation 2.44.

$$A_b = 2n_{sp} \sum_{i=1}^{n} A_{r_i} \tag{2.44}$$

The discretization only covers one half of one star grain, similar to what is shown in Refs. [16, 17, 22] for this reason the summation must be multiplied by $2 * n_{sp}$. The value $n_{sp}$ represents the number of star points and the number two is included to make up for the half angle discretization. Equation 2.44 is another form of Equation 2.18 shown in the section covering the basic star grain equations. In Equation 2.44, the parameter $n$ represents the total number of discretized points for one half of a star point. Note: When the burn area is being calculated, the first point used is Point 1 (or Point 2) depending on the burn phase, and the Cartesian coordinates are ordered to where the last point is Point 5. The order of burn area calculation is shown below in Figure 2.9 with the black arrow.

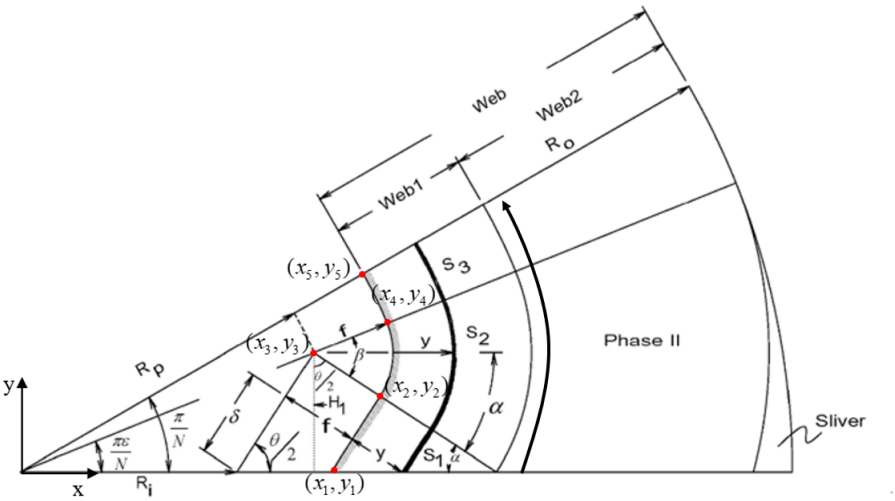



Figure 2.9: Burn Area Coordinate Path

Figure 2.9 shows the black arrow starting at the first point ($i = 1$) and ending at the last point ($i = n$), here $n$ is equal to the total number of points for a grain face. Using sets of points on the forward and aft end a differential burn area ($dAB$) can be calculated that then leads to the total burn area by using the equations shown above.

Now that we have talked about the logic to calculate the burn area, we can discuss a bit more the burn phases and how to obtain the Cartesian coordinates needed for each phase of the burn. In past work for modeling star grain motors, there have been three burn phases [16, 17] this work introduces a new burn phase when the larger end of the SRM begins to burn at the wall. For the purpose of this work we will introduce four new burn phases. These burn phases will be known as Phase A, B, C, and D. This helps us distinguish between the Phase I and II geometric equations developed by Barerre [22], with work added by Hartfield [16]. When talking about the phases of the burn the letters A, B, C and D are used, but when talking about the geometric equations and shape of the star grain Phase I and II will be used.

### 2.5.2 Phase A

The logic used to develop the equations for phase A of the burn is the same as what is shown in [16] and [22] in their work on star grain methods. The motor is burning in phase A as long as there is some length to $S_1$ or more easily shown when the value of y is less than the value of the first web thickness for the forward end ($web_{1_f}$). The equations shown for the five way points will be modified to now take into account the changing burn distance ($y$). Equations 2.45 - 2.49 show the way points for Phase A of the burn. These points will be known as Points 1 - 5. For this phase of the burn, Phase A, the design of the forward and aft end stars will be modeled with the Phase I geometry introduced by Barrere [22], along with work from Hartfield [16, 17].

$$(x_1, y_1) = (R_i + \frac{f + y}{\sin\theta/2}, 0) \tag{2.45}$$

$$(x_2, y_2) = (x_1 + S_1 \cos\theta/2, S_1 \sin\theta/2) \tag{2.46}$$

$$(x_3, y_3) = (R_p \cos\frac{\epsilon\pi}{n_{sp}}, R_p \sin\frac{\epsilon\pi}{n_{sp}}) \tag{2.47}$$

$$(x_4, y_4) = (x_3 + (f + y)\cos\frac{\epsilon\pi}{n_{sp}}, y_3 + (f + y)\sin\frac{\epsilon\pi}{n_{sp}}) \tag{2.48}$$

$$(x_5, y_5) = ((R_p + f + y)\cos\frac{\pi}{n_{sp}}, (R_p + f + y)\sin\frac{\pi}{n_{sp}}) \tag{2.49}$$

Now that the way points have been developed in Cartesian coordinates in the XY plane, the burn perimeters $S_1$, $S_2$, and $S_3$ and be discretized. The calculation of the points for the $S_1$ and $S_3$ perimeters is quite straightforward, while the calculations for the coordinated in the $S_2$ arc are a bit more involved. Figure 2.10 shows the paths (red arrows) used to discretize each burn surface for Phase I. The $S_1$ arc starts at Point 1 and moves up until it reaches the $S_2$ arc. The calculation of the $S_2$ arcs start at a middle point that is easily defined from Point 3 and move away from that middle point until they reach either the $S_1$ or $S_3$ arc, depending on if you are above or below that middle point. This middle point can be defined below.

$$(x_{middle}, y_{middle}) = (x_3 + f + y, y_3) \tag{2.50}$$

The $S_3$ points start at Point 5 and move down until they reach Point 4. Once all of these points have been discretized, they are then sorted in a new vector that starts at Point 1 and ends at Point 5.

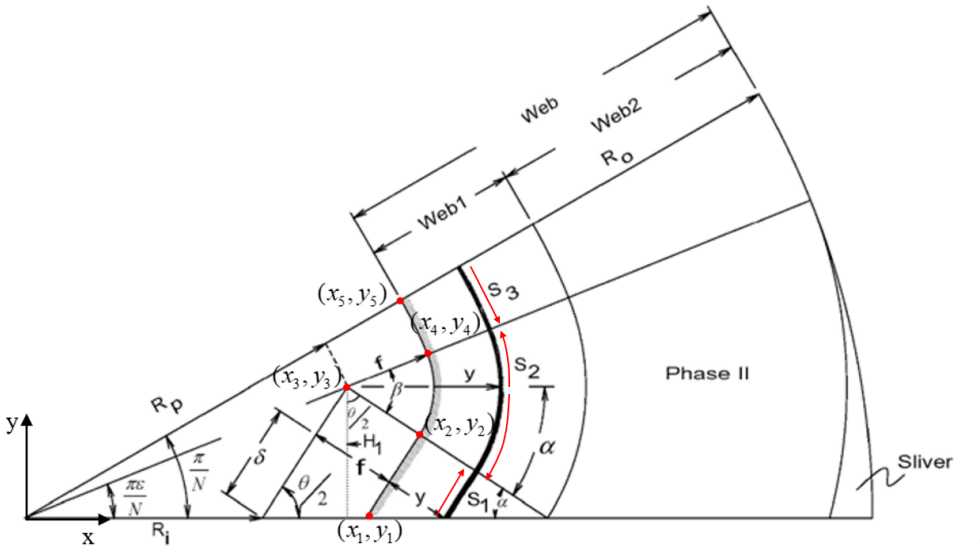



Figure 2.10: Schematic of a Star Grain - Phase I discretization

For the $S_1$ section, the burn perimeter is discretized into a select number of coordinates. This is done by calculating $S_1$ as follows in Equation 2.51. In equation 2.51, $\theta$ is the star point angle, and $H_1$ is calculated using equation 2.15.

$$S_1 = \frac{H_1}{\sin\theta/2} - (y+f)\cot\theta/2 \tag{2.51}$$

After $S_1$ is calculated, the line can be divided into multiple sections. Once the lines are divided up, the angle $\theta/2$ can be used to calculate the coordinates along the line. Keep in mind that these methods apply for discrete values of burn distance ($y$), so at each value of $y$, the value of $S_1$ is calculated and discretized. Now that we have talked about how to break up the $S_1$ burn section. We can quickly show the equations used to calculate the x and y coordinates along the $S_1$ line. The aforementioned equations can be seen below in Equation 2.52.

$$\begin{aligned} x_{S_{1i}} &= x_{S_{1i-1}} + dS_1\cos(\theta/2) \\ y_{S_{1i}} &= y_{S_{1i-1}} + dS_1\sin(\theta/2) \end{aligned} \tag{2.52}$$

In Equation 2.52 the value of $i$ goes from 1 to the number of times the $S_1$ line is discretized. The formula must be initialized, and the first values of x and y on the $S_1$ arc can be easily seen from Figure 2.5 as $R_i + (f+y)/\sin\theta/2$ for x and 0 for y. Also, from Equation 2.52, $dS_1$ can be defined below as the following.

$$dS_1 = \frac{S_1}{n_{dx}} \tag{2.53}$$

where $n_{dx}$ represents the number of times that the $S_1$ arc is discretized. Now that we have discussed how to calculate the coordinated along the $S_1$ perimeter, we can move on to the $S_2$ arc.

To discretize the $S_2$ arc we will break it into two smaller arcs. If we reference Figure 2.5 we can see that if you draw a horizontal line through $(x_3, y_3)$ the $S_2$ arc will be intercepted. This will be our dividing line for discretization of the $S_2$ arc. We will call the bottom arc $S_{2_b}$ and we will call the top arc $S_{2_a}$. The angle that makes up the bottom arc will be $\beta - \frac{\pi\epsilon}{n_{sp}}$ while the top angle will be $\frac{\pi\epsilon}{n_{sp}}$. These two angles will be broken up into smaller arcs and then the

XY coordinates can be calculated. The following equations will be used to calculate the XY coordinates for $S_{2_b}$ and $S_{2_a}$, respectively.

$$\begin{aligned} x_{S_{2_{b\,i}}} &= x_3 + (f+y)\cos\left(d\psi * (i-1)\right) \\ y_{S_{2_{b\,i}}} &= y_3 - (f+y)\sin\left(d\psi * (i-1)\right) \end{aligned} \tag{2.54}$$

$$\begin{aligned} x_{S_{2_{a\,i}}} &= x_3 + (f+y)\cos\left(d\chi * (i-1)\right) \\ y_{S_{2_{a\,i}}} &= y_3 + (f+y)\sin\left(d\chi * (i-1)\right) \end{aligned} \tag{2.55}$$

In Equations 2.54 and 2.55 the values $\psi$ and $\chi$ are defined as follows, and the value of $i$ is the counter of the do loop that goes from 1 to $n_{dx}$ for each section. Like mentioned earlier, $n_{dx}$ is the number of times each burn perimeter section is discretized.

$$d\psi = \frac{\beta - \frac{\pi\epsilon}{n_{sp}}}{n_{dx}} \tag{2.56}$$

$$d\chi = \frac{\frac{\pi\epsilon}{n_{sp}}}{n_{dx}} \tag{2.57}$$

Now that the discretization for $S_1$ and $S_2$ has been explained, the equation for how the $S_3$ arc is discretized can be explained for Phase A. We will be discretizing an arc through an angle that is defined as $\pi/n_{sp} - \pi\epsilon/n_{sp}$. This angle will be called $\phi$. Now that we have defined $\phi$ the equations can be shown to calculate the values of x and y for the $S_3$ arc.

$$\begin{aligned} x_{S_{3_i}} &= (R_p + f + y)\cos\left(\frac{\pi}{n_{sp}} - (d\phi * (i-1))\right) \\ y_{S_{3_i}} &= (R_p + f + y)\sin\left(\frac{\pi}{n_{sp}} - (d\phi * (i-1))\right) \end{aligned} \tag{2.58}$$

In Equation 2.58, $d\phi$ is defined below in Equation 2.59 and again $i$ is the counter in the do loop going from 1 to $n_{dx}$. It is worth to mention that the first value for the $S_3$ discretization is essentially $(x_5, y_5)$ from Figure 2.5.

$$d\phi = \frac{\phi}{n_{dx}} \tag{2.59}$$

Now the equations and methods have been developed for discretization of the burn perimeters described and shown in Figure 2.10. We now have $4 * n_{dx}$ coordinate pairs that define this star grain design. Using the methods shown previously to calculate the burn area we now can get burn area as a function of $y$ while the motor is in Phase A. It is important to notice that we have only defined the XY coordinates for Phase A of the burn. For Phase A, the Z component is equal to zero at the forward end, and equal to the grain length $GL$ at the aft end. Figure 2.11 shows the geometry of the star points at the forward and aft end during Phase A. Table 2.3 helps give a better understanding of the Phase A burn and required geometry.

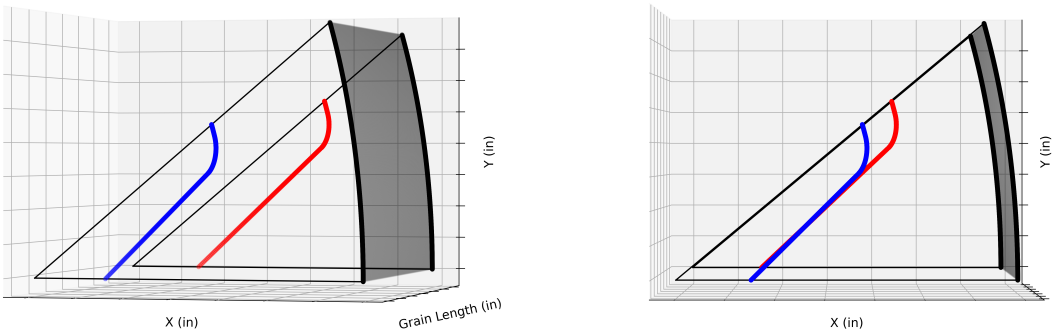



Figure 2.11: Phase A Burn Schematic

Table 2.3: Phase A Summary

| Grain Face | Geometric Design | Z Component |
|---|---|---|
| Forward | Phase I | 0 |
| Aft | Phase I | GL |

### 2.5.3 Phase B

Phase B begins when the burn perimeter $S_1$ burns out for the forward end of the SRM grain. To model this we will need to introduce a new set of equations at the forward end of the SRM, these equations are the Phase II geometric equations proposed by Hartfield et al. [16, 17]. The aft end of the SRM will use the Phase I geometry. The schematic of a Phase B burn can be see in Figure 2.12.

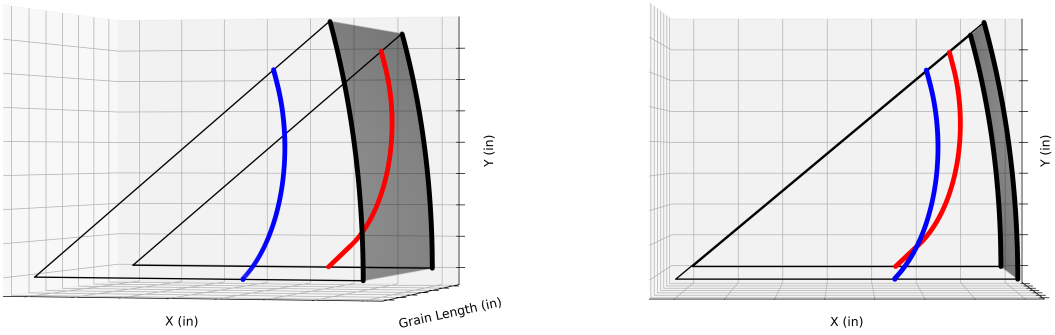



Figure 2.12: Phase B Burn Schematic

Looking at Figure 2.3 this shows a schematic of a the Phase B burn. For the Phase B burn, the forward end geometry is defined with Phase II geometry while the aft end is still defined with Phase I geometry. Figure 2.3 shows the aft end star point (red) still has a straight section $S_1$, while the forward section (blue) has no $S_1$.

The logic to make sure that the burn is in Phase B is as follows. If the value of burn distance ($y$) is greater than the forward end web thickness for Phase I geometry ($web_1$) and less than the aft end web thickness for Phase I geometry ($web_1$), we are in what we will define as Phase B. The equation for web thickness for a Phase I star grain geometry is as follows.

$$web_1 = \frac{H_1}{\cos\left(\theta/2\right)} - f \tag{2.60}$$

By bookkeeping the burn distance ($y$) we can easily tell what phase of the burn we are in. Like mentioned in the previous paragraph, the forward end star design is what is changing for this phase. The forward end of the grain is calculated using the Phase II geometric equations. Now that the logic had been explained to get to Phase B of the burn we can show the equations needed for Phase B. For the Phase II geometric discretization, the $S_1$ has burnt out, now we only need to worry about modeling the $S_2$ and $S_3$ arcs. Figure 2.13 shows the process of discretizing the burn perimeter used for the Phase II equations.

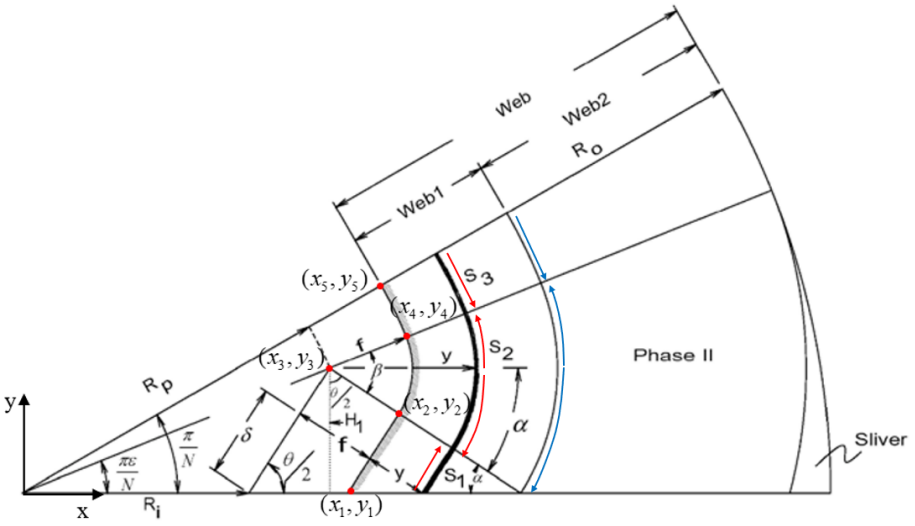



Figure 2.13: Schematic of a Star Grain - Phase II discretization

Looking at Figure 2.13, the blue arrows represent the Phase II discretization path. As mentioned previously, the $S_1$ arc is now burnt out. This simplifies our process slightly. Just like shown for Phase I, the $S_2$ arc is broken up into a top and bottom arc. These arcs begin at a middle point that can be defined as follows.

$$(x_{middle}, y_{middle}) = (x_3 + f + y, y_3) \tag{2.61}$$

The top arc goes until it meets Point 4 and the bottom arc goes until it meets the wall, which in this case will be where Point 2 is located during Phase II. The $S_3$ arc is discretized following the arrow from Point 5 until it reaches Point 4. Just like is done in Phase I, after the

points have been discretized for the grain face, the points are organized to trace from Point 2 up to Point 5. This is done for uniformity between the burn phases.

The $S_3$ arc is calculated the same way as it was done in Phase I, for completeness the Equation will be shown again in Equation 2.62.

$$\begin{aligned} x_{S_{3i}} &= (R_p + f + y)\cos\left(\frac{\pi}{n_{sp}} - (d\phi * (i-1))\right) \\ y_{S_{3i}} &= (R_p + f + y)\sin\left(\frac{\pi}{n_{sp}} - (d\phi * (i-1))\right) \end{aligned} \tag{2.62}$$

In Equation 2.62, $d\phi$ is defined in Equation 2.59 and again $i$ is the counter in the do loop going from 1 to $n_{dx}$. Now that we have shown how the equation for $S_3$ is discretized, we can now move on to the calculation and discretization of the $S_2$ arc. Figure 2.3 shows an image of what the $S_2$ arc looks like for the Phase II geometry of a star grain. Looking at Figure 2.3, we can see that eventually the the $S_1$ arc will burn out and we will only be left with $S_2$ and $S_3$. The Phase II geometry of the burn also introduces a new angle $\gamma$. We can define $\gamma$ below in Equation 2.63. Figure 2.3 in Section 2 gives an image of the Phase II geometry with $\gamma$ included.

$$\gamma = \arctan\frac{\sqrt{(y+f)^2 - H_1^2}}{H_1} - \frac{\theta}{2} \tag{2.63}$$

Now that $\gamma$ has been defined we can go ahead and develop the equations needed for the discretization of $S_2$ for the Phase II geometry. Again we will break up the $S_2$ arc up into two smaller arcs $S_{2_a}$ and $S_{2_b}$. The dividing line for these arcs a horizontal line thought the point $(x_3, y_3)$ from Figure 2.5. The equations for the top arc $S_{2_a}$ are the same as they were for a Phase I star geometry, there is a slight change in the $S_{2_b}$ equations, these sets of new equations will be shown below.

$$\begin{aligned} x_{S_{2_{b\,i}}} &= x_3 + (f + y)\cos\left(d\psi' * (i-1)\right) \\ y_{S_{2_{b\,i}}} &= y_3 - (f + y)\sin\left(d\psi' * (i-1)\right) \end{aligned} \tag{2.64}$$

$$x_{S_{2_a\,i}} = x_3 + (f + y)\cos\left(d\chi * (i-1)\right)$$
$$y_{S_{2_a\,i}} = y_3 + (f + y)\sin\left(d\chi * (i-1)\right) \tag{2.65}$$

In Equations 2.64 and 2.65 the values $\psi'$ and $\chi$ are defined as follows, and the value of $i$ is the counter of the do loop that goes from 1 to $n_{dx}$ for each section. Like mentioned earlier, $n_{dx}$ is the number of times each burn perimeter section is discretized.

$$d\psi' = \frac{\beta - \gamma - \frac{\pi\epsilon}{n_{sp}}}{n_{dx}} \tag{2.66}$$

$$d\chi = \frac{\frac{\pi\epsilon}{n_{sp}}}{n_{dx}} \tag{2.67}$$

Looking at Equations 2.64, 2.65, 2.66, and 2.67 we can see that the main difference between Phase I and Phase II geometries for the $S_2$ arc is the inclusion of $\gamma$ into the angle defining the $S_2$ arc. Now that the coordinates have been defined for a phase II geometry we again have $4 * n_{dx}$ coordinate pairs. The $S_1$ points are still defined as the point shown as point 2 in Figure 2.5, for a Phase II geometry $(x_2, y_2)$ would lie on the x-axis. Again we have described the process of calculating the XY coordinates for the second burning phase, Phase B. Just like for the Phase A burn the Z component must be defined. Again, the forward end Z component will be defined as 0 and the aft end Z component will be set to the grain length. Table 2.4 gives and overview of the geometry used in Phase B of the SRM burn.

Table 2.4: Phase B Summary

| Grain Face | Geometric Design | Z Component |
|---|---|---|
| Forward | Phase II | 0 |
| Aft | Phase I | GL |

### 2.5.4 Phase C

The SRM enters Phase C of the burn when the $S_1$ arc has been burnt out for both the forward and aft end of the solid rocket motor, phase C ends before the aft end star reaches the outer grain limit. The logic for Phase C is as follows: when the burn distance is greater than $web_1$

for the aft end, but less than the value of the total web thickness for the aft end. At this phase of the burn, both the forward end and the aft end of the solid rocket motor are designed using the Phase II geometric equations. Due to the fact that we are only modeling a positive taper ratio and we are assuming uniform grain regression, we know that the aft end star will burn out before the forward end star does. The visualization of Phase C can be seen below in Figure 2.14.

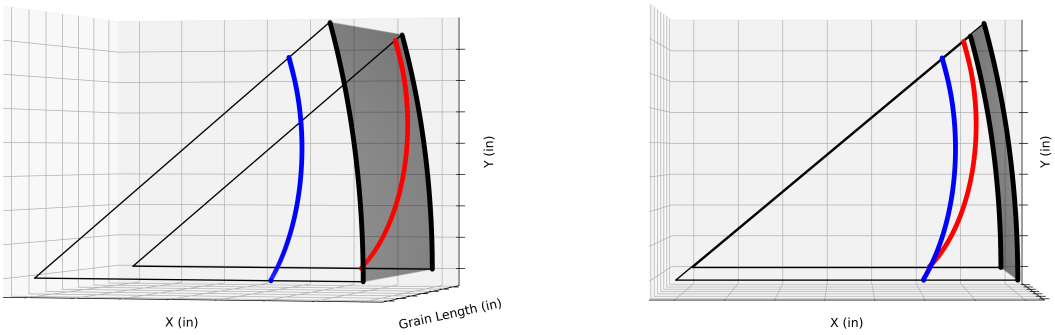



Figure 2.14: Phase C Burn Schematic

Figure 2.14 shows the forward and aft end burn faces for Phase C of the SRM burn. During Phase C of the burn, the star points are modeled using a discretized Phase II geometry. The process of discretizing the phase II geometric equations was developed in Section 2.5.3. Looking at Figure 2.14, it can be seen that the aft end star is beginning to approach the wall. Once this aft end star hits the wall (when $y$ is greater than the aft end web thickness) the burn Phase will switch to Phase D. While still in Phase C the Z component of the coordinate points is defined the same as Phase A and B. Table 2.5 shows the geometric model used for Phase C.

Table 2.5: Phase C Summary

| Grain Face | Geometric Design | Z Component |
|---|---|---|
| Forward | Phase II | 0 |
| Aft | Phase II | GL |

### 2.5.5 Phase D

The addition of the Phase D burn phase for the tapered star grain is one of the major additions to the literature on tapered grains. Phase D must consider what happens if the aft end of the star grain begins to reach the wall before the forward end does. Since we are assuming uniform burn distance ($y$), with a tapered grain the aft end will burn out before the forward end does. To accurately model the tapered grain, we must account for the new effective grain length, or what we can call the burn length. This new burn length will be used when calculating the burn area of the SRM. A schematic a Phase D can be seen below in Figure 2.15.

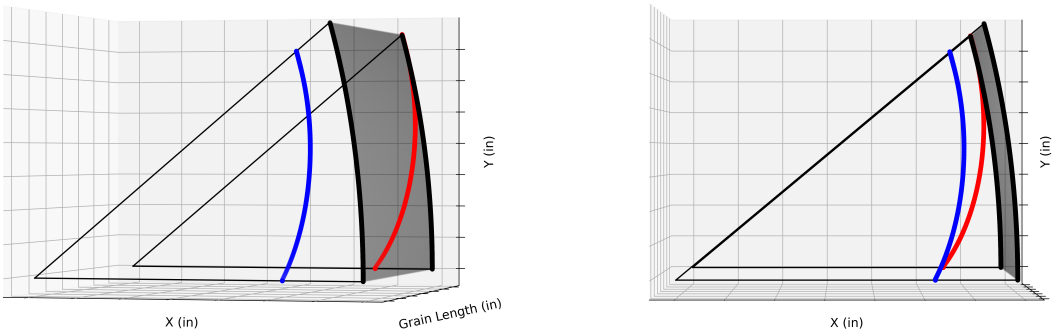



Figure 2.15: Phase D Burn Schematic

The equations used to discretize the burn perimeter are the discretized Phase II geometric equations shown in Section 2.5.3. As the phase II grain shape burns thought outer grain wall, some points will be outside the grain perimeter, it is grain length of those points that will be corrected for accurately model the burn area. Figure 2.16 gives a better image of the aft end

star design burning through the grain boundary. The grain boundary is shown in the dark grey in Figure 2.16.

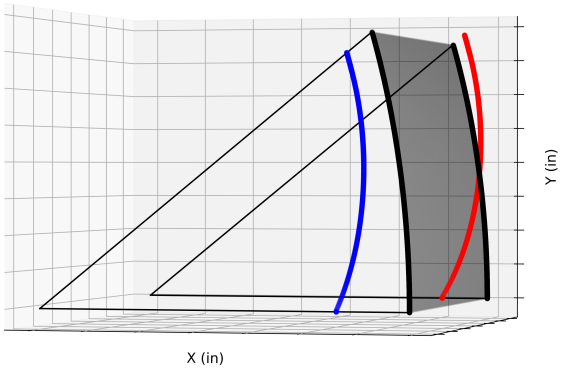



Figure 2.16: Phase D with points outside grain wall

To find the new grain length for each side of a $dAB$ we will first need to define the line connecting the forward and aft end points in 3D space. An image of this can be seen in Figure 2.17. Figure 2.17 shows a limited number of lines connecting the forward and aft points.

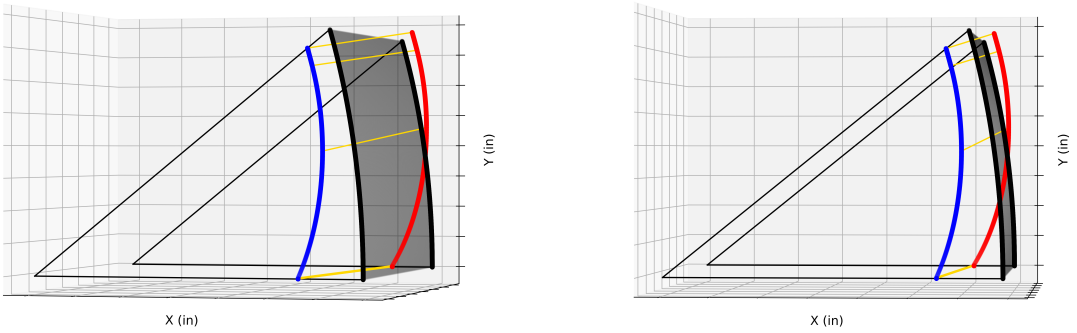



Figure 2.17: Phase D Burn with points outside grain wall (lines included)

To properly define the line in 3D space, we define the following three equations.

$$\begin{aligned} x &= x_0 + at \\ y &= y_0 + bt \\ z &= z_0 + ct \end{aligned} \tag{2.68}$$

These three equations shown together in Equation 2.68 represent the parametric equations for defining a line in 3D space. Using what we know about the problem and how the coordinate frames have been applied, we can change the equations to what is seen in Equation 2.69

$$\begin{aligned} x &= x_f + (x_a - x_f)t \\ y &= y_f + (y_a - y_f)t \\ z &= (GL_i)t \end{aligned} \tag{2.69}$$

Because the number of coordinates is the same for the forward and aft sections, we can just use the index of the coordinates to ensure we are connecting the proper points in Cartesian coordinate space. The subscript $a$ and $f$ represent x and y coordinates in on the forward and aft section of the SRM grain, and the parameter $GL_i$ is the initial grain length. Looking at the equations shown in 2.69, we still have one unknown $t$. The goal of this burn phase method is to find $t$, and using $t$ we can then calculate a new value of $z$ which will become for us the effective grain length that we will use in our burn area calculation.

To correctly find $t$, we can use what we know about the problem. We know that the points that will fall outside the wall us when the magnitude of the $x$ and $y$ is greater than the outer grain radius $R_o$. The logic can be seen below in Equation 2.70. Looking at Equation 2.70 we can see that the outer grain radius ($R_o$) acts as a bound to check if the coordinates are outside of the outer grain radius, $R_o$.

$$R_o = \sqrt{x^2 + y^2} \tag{2.70}$$

We can set up a numerical scheme to find the value of $t$ that comes from the above mentioned logic. Equation 2.71 shows the equation that will be used to find $t$. A bisection scheme is used to solve Equation 2.71 for $t$. It is worth noting that this method converges very quickly and is very fast as it was integrated into the Fortran code used to build and model these grains [37].

$$dr = \sqrt{((x_a - x_f)t + x_f)^2 + ((y_a - y_f)t + y_f)^2} - R_0 \tag{2.71}$$

The bisection method is a common root finding and typically converges quickly. The algorithm will try and search for the value of $t$ that makes the absolute value of $dr$ less than the error tolerance chosen for the bisection method. The error tolerance for this method was set to around 1e-8. Once the bisection method converges on the correct value for $t$, that value can be plugged into the $z$ equation from Equation 2.69 to obtain the new effective grain length to be used in the calculation of burn area for each panel. The bisection method is essentially finding the Z component of the location where a line connecting the points crosses the grain boundary, seen in Figure 2.17.

Table 2.6: Phase D Summary

| Grain Face | Geometric Design | Z Component |
|---|---|---|
| Forward | Phase II | 0 |
| Aft | Phase II | $GL_it$ |

Now that we have discussed the new burn phases A-D, we can summarize with the tables below. Table 2.7 shows the logic used to switch between the burn phases, while Table 2.8 shows the grain geometry for each of the new burn phases. With these new burn phases developed, they can be integrated into the 1D internal ballistics tool used for this work.

Table 2.7: Summary of Burn Phase Logic

| Phase | Logic |
|---|---|
| A | $y \leq web_{1_f}$ |
| B | $y \leq web_{1_a}$ |
| C | $y \leq (web_{1_a} + web_{2_a})$ |
| D | $y \leq ymax_f$ |

Table 2.8: Burn Phase Summary

| Phase | Grain Face | Grain Design | Z Component |
|---|---|---|---|
| A | Forward | Phase I | 0 |
| A | Aft | Phase I | GL |
| B | Forward | Phase II | 0 |
| B | Aft | Phase I | GL |
| C | Forward | Phase II | 0 |
| C | Aft | Phase II | GL |
| D | Forward | Phase II | 0 |
| D | Aft | Phase II | $GL_i t$ |

### 2.5.6 Geometric Verification

To verify the assumptions of this model, we will perform a simple case study to calculate the initial burn area of the tapered SRM grain. For this case we will look at the simple star grain design shown in Figure 2.18.

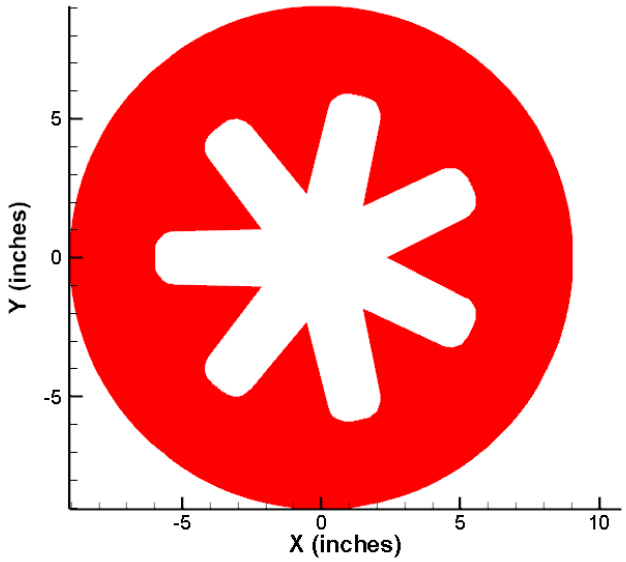



Figure 2.18: Star Grain Verification - Forward Grain Design

This star grain design has the following design parameters that can be seen below in Table 2.9. Using these parameters, we will geometrically verify the equations and methods developed in this section.

Looking at the parameters defined in Table 2.9, we can calculate the burn area in two ways. The first way is to solve for the burn area while calculating the forward and aft end

Table 2.9: Star Grain Verification Designs Parameters

| Parameter | Value |
|---|---|
| $R_{p_f}$ | 5.29178 ” |
| $R_{p_a}$ | 6.35014” |
| $R_i$ | 0.8292 ” |
| $R_o$ | 8.9643 ” |
| $f$ | 0.6879 ” |
| $GL$ | 72.255 ” |
| $\epsilon$ | 0.8885 |
| $n_{sp}$ | 7 |

burn perimeter. With these two parameters we can estimate the burn area using the following equation.

$$A_b = \frac{1}{2}(S_f + S_a)GL = \frac{1}{2}(A_{b_f} + A_{b_a}) \tag{2.72}$$

Equation 2.72 states that the burn area (initial) is equal to the average of the forward burn perimeter ($S_f$) and the aft burn perimeter ($S_a$) times the length of the grain. This is essentially the surface area of a frustum with a star design instead of a rectangle or circular design. To calculate the $S_a$ and the $S_f$ we can use the phase I burn area equations shown in Section 2.2. We can then compare the hand calculated value with the value of the initial burn area that is generated from the SRM internal ballistics tool when the burn distance ($y$) is equal to 0. The results of this comparison are shown below in Table 2.10.

Table 2.10: Star Grain Verification Results

| Method | Result |
|---|---|
| Code Calculation | 5109.51 $in^2$ |
| Hand Calculation | 5110 $in^2$ |
| Percent Error | 0.009 $\%$ |

Looking above at Table 2.10, we can see the values calculated with the code, and by hand for the simple star frustum type shape. These differences could even be attributed to rounding errors in calculation methods, especially the hand calculation. Regardless, this case study shows that our method predicts the burn area with less than one percent error for this simple case study. This case study shows that our predictions for burn area, and in return chamber pressure and thrust are verified conceptually.

# Chapter 3

# Solid Rocket Motor Internal Ballistics Tool

A tapered grain solid rocket motor internal ballistics tool was developed for this thesis. Based on legacy FORTRAN code that was used in previous work [11, 24, 25, 27], an updated FORTRAN tool was developed for this effort. The newest version of this code can support tapering of the solid rocket motor grain for both CP and star grain solid rocket motor designs. The user can define the grain geometry at the forward end of the grain, while also defining if the grain should be tapered or straight by setting the taper ratio for the grain design parameters.

Intel FORTRAN makes it easy to run this solid rocket motor tool. To use this tool, the Intel oneAPI Base Toolkit and the Intel oneAPI HPC Toolkit must be installed. The author recommends using Microsoft Visual Studio as the IDE when developing and running the code.

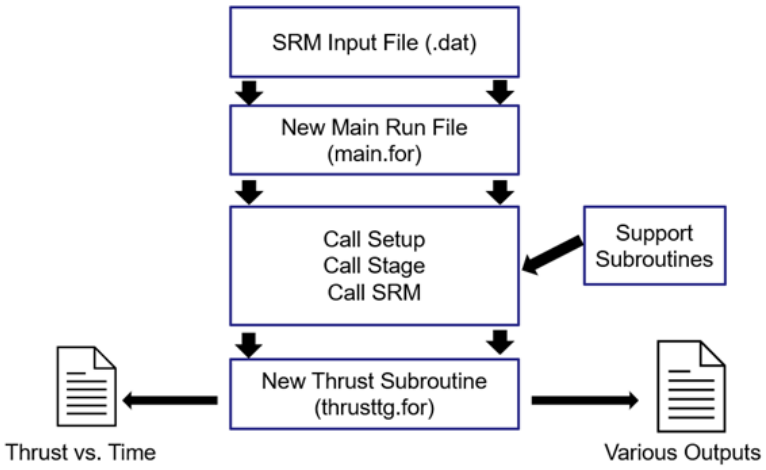



Figure 3.1: Flowchart for SRM Tool

Figure 3.1 shows the flowchart of the subroutines used when running the solid rocket motor internal ballistics tool. First, the input file is generated from the user for the single run

mode, the input file is snglerun.dat, and for a Monte Carlo run, the input file is gannlDIST.dat. Both of these files are essentially the same, except the gannlDIST.dat file is used within the Monte Carlo algorithm. Once the input file is modified the main.for file is called. The main file reads in all of the required inputs and then calls the setup FORTRAN file. Once the setup subroutine is called, the stage subroutine can be called which is only used as a first stage rocket for this analysis. After the stage subroutine, the SRM subroutine is called within the SRM subroutine the new tapered grain thrust subroutine (thrusttg.for) is called. From these subroutines the simulaiton can output pressure, burn area, and thrust as a function of time or burn distance.

Many of the subroutines used for this analysis are part of legacy solid rocket motor codes that have been developed in previous work [11, 16, 17, 24, 25, 27]. The work of this thesis mainly focused on the modification of some of the legacy codes, as well as the development of a new thrust subroutine that can support tapered CP and star grain designs.

## 3.1 Validation with Legacy Code

To insure conceptual accuracy of the code, a simple test case was used to compare the legacy codes [11, 16, 24, 25] with the new code that includes a new thrust subroutine. This validation case uses a simple straight grain to ensure that the new code can model the basics of solid rocket motor performance. Figure 3.2 shows the results of the new 1D internal ballistics code plotted against the legacy SRM internal ballistics code known as AUSRC [27].

Due to the time constraints of this work, a full validation case was not able to be performed. Based on the validation shown in this section, and the verification shown in Section 2.5.6 it is assumed that the 1D internal ballistics code is able to produce conceptual design level thrust-time curves.

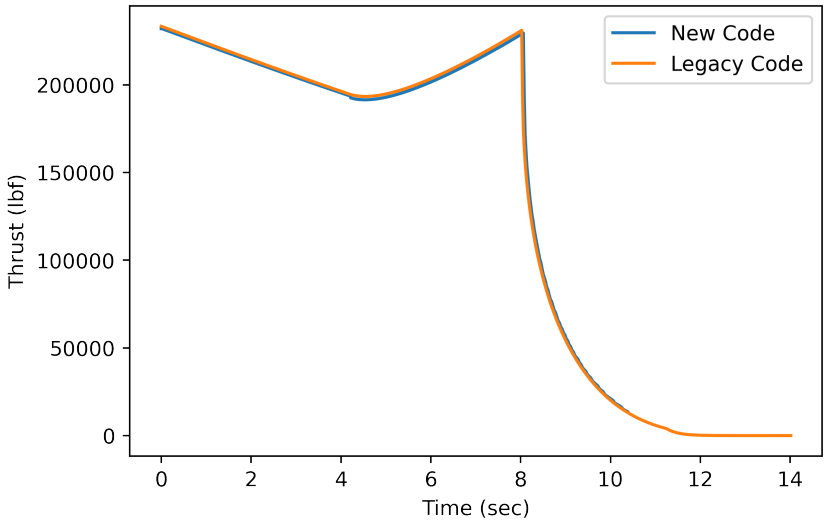



Figure 3.2: Validation Case for Star Grain Design

## 3.2 Subroutine **thrusttg**

Subroutine **thrusttg** is the subroutine that was added to the legacy 1D internal ballistics code to properly model tapered solid rocket motor grains. This subroutine takes in burn distance along with geometric parameters and calculates burn area, chamber pressure, and thrust as a function of both time and burn distance. The subroutine **thrusttg** uses the following subroutines to accurately model the tapered solid rocket motor grains. This subroutine includes the logic needed to switch between burn phases as the motor simulation progresses. Algorithm 1 shows the logic used to switch between burn phases for this simulation.

**Algorithm 1** Algorithm for Burn Phase Switching

**Require:** $y$
  **while** $y \leq ymax_f$ **do**
    **if** $y \leq web_{1_f}$ **then**
      $A_b \leftarrow$ Phase A
    **else if** $y \leq web_{1_a}$ **then**
      $A_b \leftarrow$ Phase B
    **else if** $y \leq (web_{1_a} + web_{2_a})$ **then**
      $A_b \leftarrow$ Phase C
    **else if** $y \leq ymax_f$ **then**
      $A_b \leftarrow$ Phase D
    **end if**
  **end while**

### 3.2.1 Subroutine **star_coords**

Subroutine **star_coords** takes in the grain design in one of the four burning phases and returns the discretized Cartesian coordinates. This subroutine is called for the forward and aft end coordinates. The number of Cartesian coordinates is user defined, and as a default is set to one-hundred per section for a total of four hundred coordinates. The **star_coords** subroutine is the main application and implementation of the analytical methods that were developed in Section 2.5 of this thesis.

### 3.2.2 Subroutines **section_area** and **cross_prod**

Subroutines **section_area** and **cross_prod** work together to calculate the burn area of the tapered SRM. The way that the area is calculated for the tapered solid rocket motor grains is to first form small quadrilateral type shapes by connecting the points of the aft end with their respective points from the forward end of the solid rocket motor. Once all of the quadrilaterals have been developed, the quadrilateral can be broken down into two triangles per quadrilateral, by connecting the opposing points with a line. These subroutines mentioned in this section are the practical implementation of the methods shown in Section 2.5.1. Algorithm 2 shows the practical implementation of subroutines **section_area** and **star_coords**. Algorithm 2 show how the geometric designs are assigned for each phase.

### 3.2.3 Subroutine **find_dr**

Subroutine find_dr is the subroutine that is along with a bisection method algorithm to calculate the effective grain length that needs to be used once the star grain shape is outside the radius of the solid rocket motor grain. The subroutine is essentially the following equation (seen in Equation 3.1) that is used with the bisection method for solve for the value of $t$.

$$dr = \sqrt{((x_a - x_f)t + x_f)^2 + ((y_a - y_f)t + y_f)^2} - R_0 \tag{3.1}$$

Using Equation 3.1 and a bisection method algorithm, the value for $t$ could be quickly calculated and plugged back into the parametric equations for a line in 3D to obtain the new

**Algorithm 2** Algorithm for Burn Phase Calculations

**Require:** $y$
  **while** $y \leq ymax_f$ **do**
    **if** $y \leq web1_f$ **then**
      $XYZ_f \leftarrow$ call star_coords(I) ▷ Phase I Geometry
      $XYZ_a \leftarrow$ call star_coords(I)
      $A_b \leftarrow$ call section_area(XYZ) ▷ Uses forward and aft end coordinates
    **else if** $y \leq web1_a$ **then**
      $XYZ_f \leftarrow$call star_coords(II) ▷ Phase II Geometry
      $XYZ_a \leftarrow$call star_coords(I)
      $A_b \leftarrow$ call section_area(XYZ)
    **else if** $y \leq (web1_a + web2_a)$ **then**
      $XYZ_f \leftarrow$ call star_coords(II)
      $XYZ_a \leftarrow$call star_coords(II)
      $A_b \leftarrow$ call section_area(XYZ)
    **else if** $y \leq ymax_f$ **then**
      $XYZ_f \leftarrow$ call star_coords(II)
      $XYZ_a \leftarrow$ call star_coords(II)
      $A_b \leftarrow$ call section_area(XYZ)
    **end if**
  **end while**

grain length for that section of the solid rocket motor grain. Once the new effective grain length has been calculated, the **section_area** subroutine can be used to calculate the burn area for one half of the star point. With that area calculated, the whole SRM burn area can be calculated using Equation 2.44. In Equation 2.44, the value $A_{R_i}$ is what is returned from subroutine **section_area**. The application of subroutine **find_dr** as part of subroutine **thrusttg** is shown below in Algorithm 3.

**Algorithm 3** Bisection Implementation for Finding Burn Length with Tapered SRMs

**Require:** $y \geq (web_{1_a} + web_{2_a})$ & $y \leq ymax_f$
  **while** $y \leq ymax_f$ **do**
    $r \leftarrow \sqrt{x_a^2 + y_a^2}$
    **if** $r$ is greater than $R_o$ **then**
      **for** $j = 1$, max iterations **do**
        $\Delta r_0 \leftarrow$call find_dr(t0)
        $\Delta r_1 \leftarrow$call find_dr(t1)
        **if** $|\Delta r_0| \leq tol$ **then**
          $t \leftarrow t0$
        **else if** $|\Delta r_1| \leq tol$ **then**
          $t \leftarrow t1$
        **else**
          $t \leftarrow \frac{t0+t1}{2}$
          $\Delta r \leftarrow$ find_dr(t)
          **if** $(\Delta r_0 \Delta r) < 0$ **then**
            $t1 \leftarrow t$
          **else**
            $t0 \leftarrow t$
          **end if**
        **end if**
      **end for**
      $Z \leftarrow GL_i * t$
    **else**
      $Z \leftarrow GL_i$
    **end if**
  **end while**

Now that the important new subroutines have been explained, example thrust-time curves can be shown in the next section of this thesis.

## 3.3 Internal Ballistics Results

The purpose of this section is to show the variety of the results that can be generated with the internal ballistics tool developed for this work. This solid rocket internal ballistics tool can model both star grain and CP grain designs. The purpose of this section is to show possible designs and thrust time curves that can be generated with this code. Figures 3.3 - 3.6 show a few of the thrust-time profiles that can be generated with this internal ballistics code.

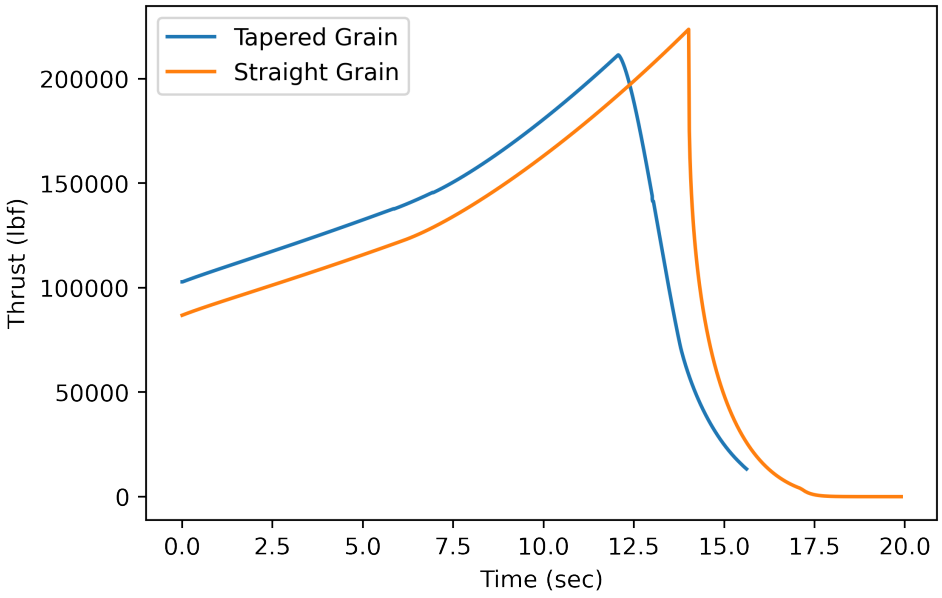



Figure 3.3: 5-point Star Grain Thrust-Time Curves

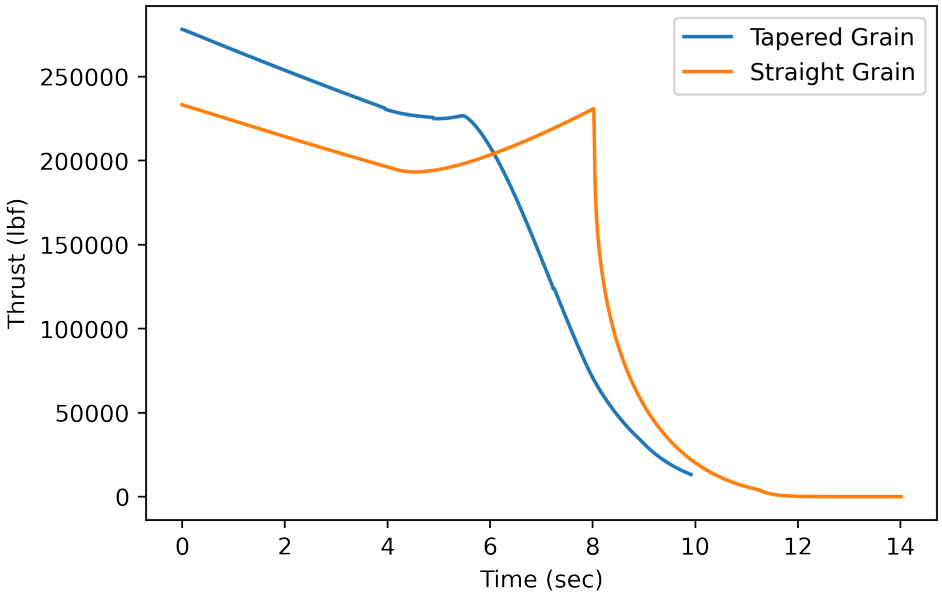



Figure 3.4: 7-point Star Grain Thrust-Time Curves

Figures 3.3-3.6 show the thrust-time curves for a fixed grain geometry with different number of star points. The orange curve represents the thrust-time curve for the straight grain, and the blue curve represents the tapered grain thrust-time curves. For each of these runs, the RP_TR was set to 0.2, so a twenty percent increase in $R_p$ as the motor goes aft. The effects of that tapering can be seen in Figures 3.3-3.6.

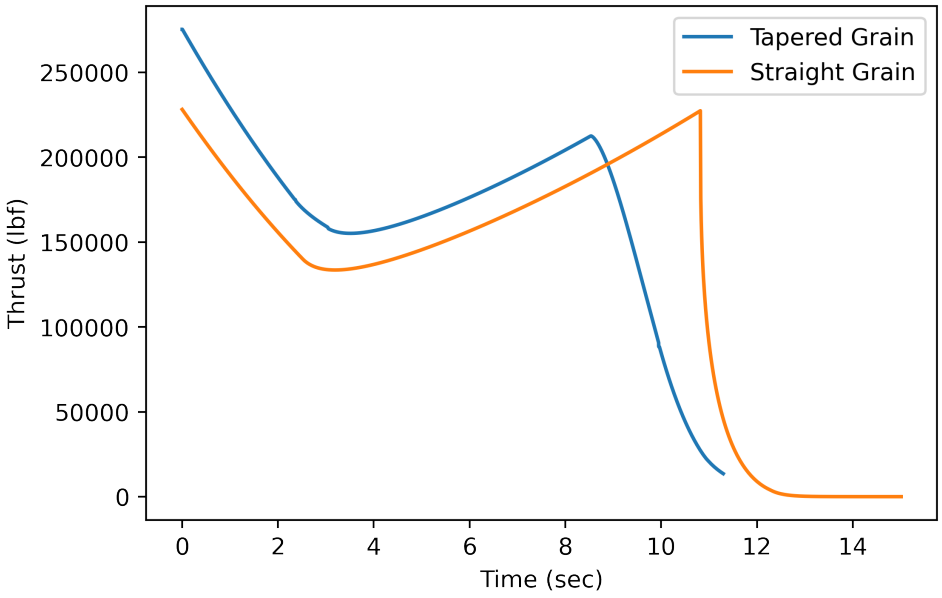



Figure 3.5: 9-point Star Grain Thrust-Time Curves

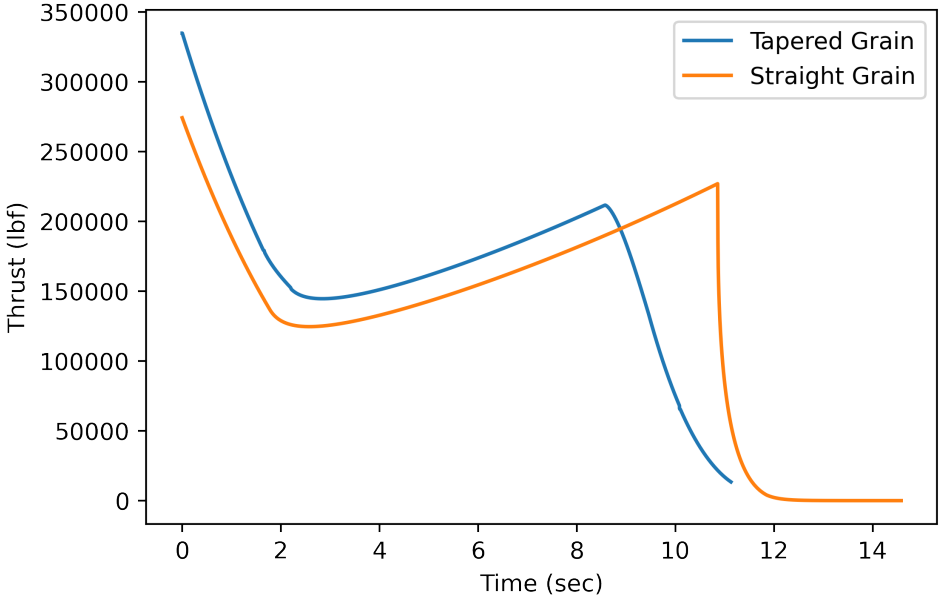
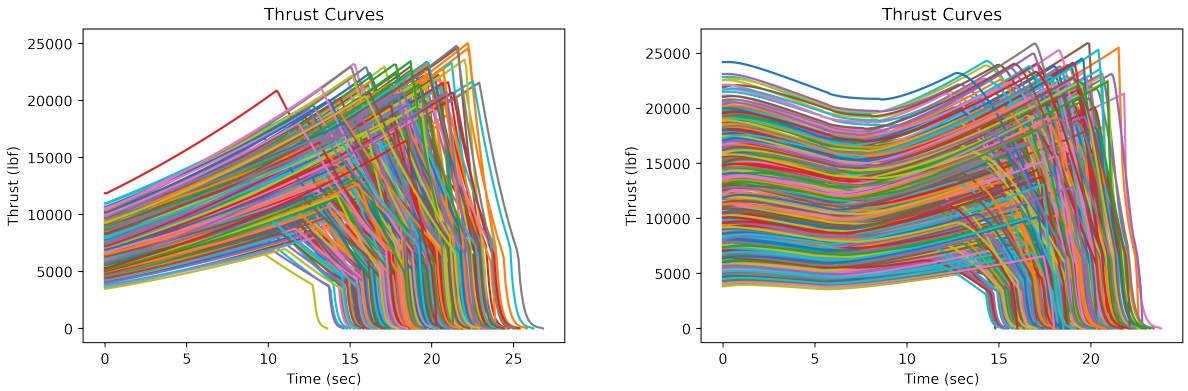



Figure 3.6: 11-point Star Grain Thrust-Time Curves

## 3.4 Solid Rocket Motor Data Generation

The data that is covered in this section will be used for the remainder of the thesis. For this work two main data sets were developed, one for CP grains and one for star grains. The solid rocket motor internal ballistics tool has a Latin Hypercube sampling scheme that can be called by the 1D internal ballistics code [38]. The user defines and maximum and minimum for the

input geometry, along with a desired number of samples and the code will produce that number of thrust-time curves. As shown above, the data generated for this thesis are thrust-time curves for various solid rocket motors. These thrust-time curves are written to a .dat file where the Jupyter Notebooks developed for this thesis are then used to read in and analyze the thrust-time curves. Figure 3.7 shows the full thrust-time curves generated for both the CP and star grain data. The SRM internal ballistics tool has filters to make sure that the code only produces possible thrust-time curves. Grain errors and other performance related issues are filtered out before the data is written. This is not unexpected due to the design space used with the Monte Carlo algorithm. It is possible that the SRM grain generated is not a viable design depending on the various inputs. Cervantes talks more about these errors and the filtering in Ref. [27].




(a) Tapered CP Grain

(b) Tapered Star Grain

Figure 3.7: SRM Thrust Curves

### 3.4.1 CP Grain Data

The first set of data generated for this thesis was the circular perforated or CP grain data. These solid rocket motors typically produce a more progressive thrust-time curve. When developing the data set for the CP grain motors, the body diameter, the throat diameter ratio, and the the grain radius taper ratio were all varied using the Latin Hypercube scheme. Table 3.1 shows the parameters that were varied using the internal ballistics tool for the CP grain data.

For the CP grain data, 500 unique thrust-time curves were ran using the internal ballistics tool. The maximum thrust, average thrust, burn time, and total impulse were all calculated

Table 3.1: Circular Perforated Grain Design Parameter Variation

| Parameter | Description | Min | Max |
|---|---|---|---|
| Body Diameter | Body diameter of the rocket (meters) | 0.4 | 0.6 |
| Non-dimensional throat diameter | Throat Diameter/Body Diameter | 0.30 | 0.36 |
| Propellant Grain Radius Parameter | $(R_p + f)$/DBODY | 0.4 | 0.5 |
| Propellant Grain Radius taper ratio | Parameter affecting the taper of the solid grain | 0.0 | 0.50 |

when post processing the data with Python. With that being said, the main output of choice for the solid rocket motor tool was the thrust-time curve data file (thrustcurve.dat).

### 3.4.2 Star Grain Data

The other grain design used for this analysis is the star grain design. As mentioned previously, the star grain design used with our tool follows the geometry introduced by Barrere [22]. The star grain solid rocket motors will typically burn in a more neutral and even a regressive-progressive manner depending on the other grain parameters. For the generation of the star grain data set, one new parameter was added to the analysis, that being the propellant inner radius taper ratio (RI_TR). This taper ratio effects the parameter $R_i$. Table 3.2 shows the parameter variation for the star grain data set.

Table 3.2: Star Grain Design Parameter Variation

| Parameter | Description | Min | Max |
|---|---|---|---|
| Body Diameter | Body diameter of the rocket (meters) | 0.4 | 0.6 |
| Non-dimensional throat diameter | Throat Diameter/Body Diameter | 0.3 | 0.33 |
| Propellant Radius Parameter | $(R_p + f)$/DBODY | 0.45 | 0.5 |
| Inner Grain Radius taper ratio | Parameter affecting the taper of the solid grain | 0.0 | 0.10 |
| Propellant Grain Radius taper ratio | Parameter affecting the taper of the solid grain | 0.0 | 0.40 |

For the star grain data set, the number of star points was fixed to 7. The internal ballistics tool was ran in Monte Carlo mode, using the Latin hypercube scheme, to generate 500 unique thrust-time curves for the 7 point star grain design. Figure 3.7b shows the star grain thrust-time curves.

# Chapter 4

# Statistical Learning Background and Application

Machine learning techniques have been shown to be applicable for work similar to what is covered in this section [27, 39]. The majority of statistical learning results covered in this thesis will be using both traditional and advanced machine learning methods to model, predict and analyze tapered grain solid rocket motor performance metrics. The Python programming language was used for the analysis part of this thesis, and Jupyter Notebooks were developed to complete these tasks. Scikit-Learn [40] was used to develop the linear regression models, while the TensorFlow package [41] was used to develop and train the neural networks. After the development of initial neural networks, the Keras Tuner [32] was used to find neural networks with optimal hyper parameters.

## 4.1 Linear Regression

The first method used in the analysis of solid rocket motors is basic linear regression. The most basic linear regression model can be shown below in Equation 4.1.

$$y = \beta_0 + \beta_1 X + \epsilon \tag{4.1}$$

Equation 4.1 represents the simple linear regression model that is covered in most introductory statistics courses. The $\beta_0$ represents the intercept of the linear regression model, while the $\beta_1$ represents the slope if the linear regression line. In this model shown in Equation 4.1, $\epsilon$ represents the normally distributed noise that is assumed for the model. Equation 4.1 shows the basic linear regression model for a response variable ($Y$), with one predictor variable ($X$).

For this thesis, linear regression models were developed for multiple response variables such as maximum thrust, average thrust, burn time, and total impulse of the various solid rocket motors. These predictor variables require more than one predictor variable to insure a good model fit. For the CP grain data, typically four predictor variables were used and for the star grain data five predictor variables were used. Equation 4.2 and 4.3 show the form of the linear regression models used for CP and star grain data respectively.

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \hat{\beta}_3 X_3 + \hat{\beta}_4 X_4 \tag{4.2}$$

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \hat{\beta}_3 X_3 + \hat{\beta}_4 X_4 + \hat{\beta}_5 X_5 \tag{4.3}$$

The equations above are first order models. It is sometimes necessary to use higher order models to better model the response variable of interest. The higher order models include interaction terms that are the product of predictor variables. The second or third order model will not only include the interaction terms, but will include the squared and cubed predictor values. Equation 4.4 shows a second order model for the response variable when two predictor variables are used and the interaction terms are included.

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \hat{\beta}_3 X_1 X_2 + \hat{\beta}_4 X_1^2 + \hat{\beta}_5 X_2^2 \tag{4.4}$$

Like mentioned above, for the data used for this thesis uses four or five predictor variables. So we can see from Equation 4.4 that for a higher order regression model with multiple predictor variables the equation can become very large.

## 4.2 Neural Networks

Neural networks have become very popular for uses where traditional regression method fail to accurately predict the response variable. Neural networks can also be trained to act as surrogate models. A surrogate model is defined as a model that can predict a complex function based on a set of training points [42]. For this thesis the neural networks will be used for two main tasks

(1) act as a regression method to accurately predict the regression response variable and (2) act as a surrogate model for solid rocket motor thrust-time curves.

Before continuing with the application, it is important to understand the basics of a neural networks and how it can be applied to solve the problems encountered in this work. Neural networks have great predictive power [43], this fact will be leveraged to use neural networks to predict and act as a surrogate model for thrust-time curves. For this work, we will be using multiple layer feed-forward neural networks. Due to the previously stated fact, all uses of the word *neural network* will be referring to a *feed forward neural network*.

The neural networks first takes in an input $X$ and develops a nonlinear function to model the output $Y$ [44]. We can consider the values of $X$ to be predictors and the values of $Y$ to be the response, similar to what was shown in the section introducing linear regression. Figure 4.1 sourced from [44], provides a great image of a single layer neural network.

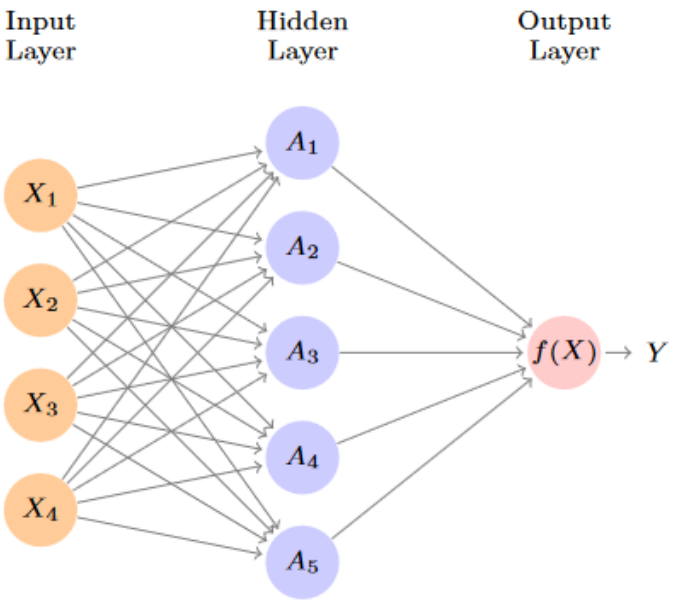



Figure 4.1: Example Feed Forward Neural Network [44]

The neural network model ends up taking the form of the nonlinear function shown below in Equation 4.5. These equations follow the derivation shown in [44].

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k \tag{4.5}$$

where

$$A_k = g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j) \tag{4.6}$$

Equation 4.6 shows what is called the activation for the neural network. In Equations 4.5 and 4.6 the $A_k$ stands for the activation, while the $g$ stands for the activation function. The location of the activation within the neural network can be seen above in Figure 4.1. The activation is made up of the activation function $g(X)$, a nonlinear function that is defined by the user. The activation function uses the weights and the predictor value to come up with an activation. The number of activations and units per layer are defined by the user of the neural network. Some common activation functions are ELU, RELU, and tanh. For more information on activation functions see Refs. [43, 44]. The neural network shown above in Figure 4.1 shows a fairly simple neural network. In reality, the neural networks developed for this work will be even more complicated than what is shown below in Figure 4.2. The neural network shown below takes in six inputs, passes them though three hidden layers of fourteen units each to predict one output. The neural networks that have been created using TensorFlow [41] for this work often have three hidden layers, with much greater than fifteen units per layer. Some of the neural network designs have up to one hundred units per layer.

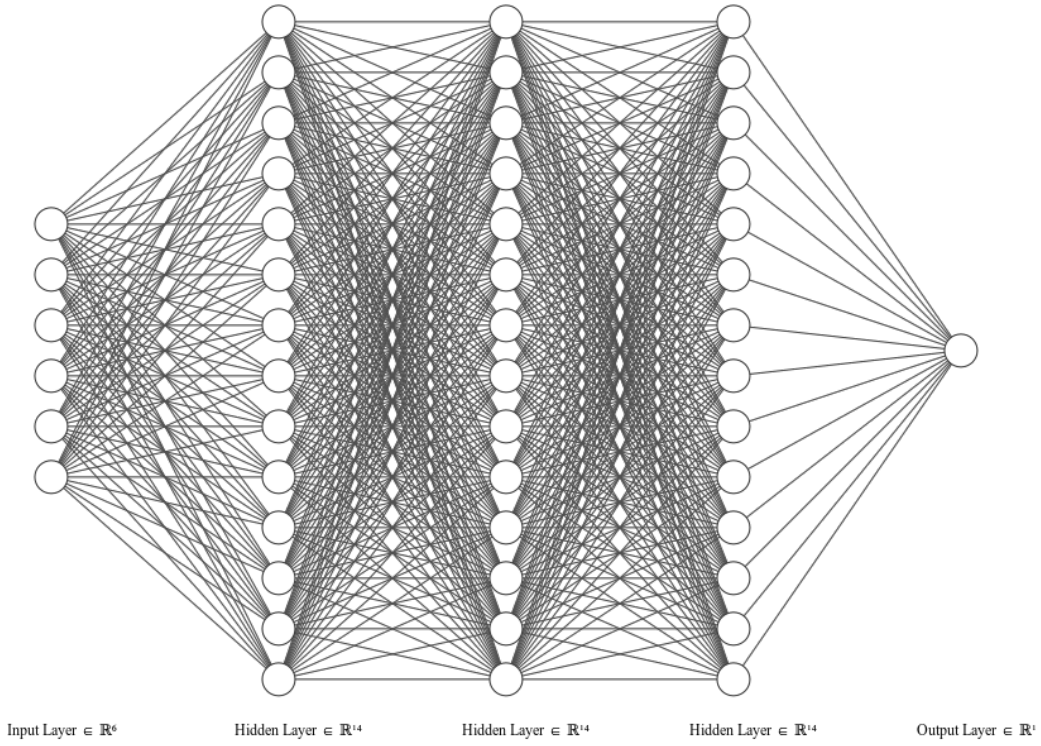



Figure 4.2: Example Feed Forward Neural Network

These neural networks will be trained on both the thrust-time curves produced by the 1D internal ballistics code, but will also be trained on the output performance parameters for a separate analysis. Once these models have been trained, they can be analyzed and then used in the future as a model to either predict performance parameters, or act as a surrogate model to the 1D internal ballistics code to predict and replicate thrust-time curves for SRMs. Like shown above in Figure 4.2, these networks can get increasingly complicated. The SHAP package has been developed and to help interpret machine learning models [31]. The following section on the application will talk more about how the SHAP package [31] is used along with TensorFlow [41] to better understand the effects that inputs $X$ have on the model output $Y$.

## 4.3 Applications

These techniques can be applied similarly to the process shown in Figure 4.3. This figure shows the flowchart of the machine learning goals of this thesis. The goal of the thesis is to better understand and model tapered grain solid rocket motors, seen as the blue arrow in Fig. 4.3.

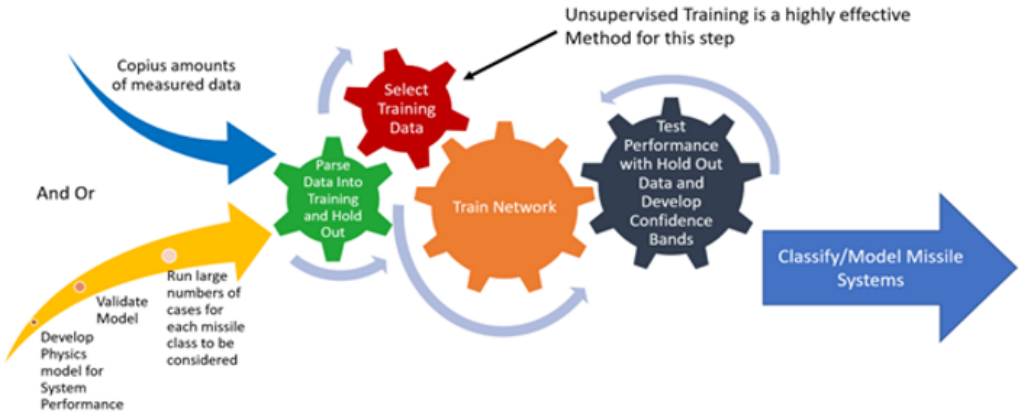



Figure 4.3: Flowchart for Statistical Learning Techniques

The first goal of the thesis was to develop and validate the models used for the tapered grain solid rocket motors, as shown in Chapters 2 and 3. Once the tool was validated, large data sets could be generated. These data sets are explained in detail in Section 3.4 of this thesis. With the required data sets were developed, the machine learning application could begin. There were two main applications of machine learning for this thesis (1) to use traditional regression techniques and neural networks to understand the effect that input parameters have on response variables and (2) train neural networks to act as surrogate models for thrust-time curves.

The first application is using traditional regression methods to model and predict the response variables from solid rocket motor simulation data. The response variables are maximum thrust, average thrust, burn time, and total impulse. The regression methods used are linear regression methods and neural networks trained to act as regression models for the solid rocket motor data set. The regression models will be looking to see the effect of the inputs of choice on the response variables of maximum thrust, average thrust, burn time, and total impulse of the tapered grain SRMs. The linear regression models are created using Scikit-Learn [40].

Once the models have been created, the regression coefficients and standardized regression coefficients can be used to better understand the model. The Seaborn package [45] is used to generate heat maps that are helpful when visualizing the regression coefficients.

When looking at the neural network regression model, created using Tensorflow [41], one does not have the luxury of interpretation with regression coefficients. Neural networks are inherently complicated, especially networks with multiple hidden layers with many units per layer. These larger, more complicated neural networks are known as deep neural networks. Due to the fact that neural networks are harder to interpret, the SHAP package will be used to better interpret and understand these machine learning models [31]. The SHAP package uses a game theory approach to help understand the effects that models have on the response variables [31]. The output of interest from the SHAP package is the SHAP value, specifically the mean absolute SHAP value. Again, the Seaborn [45] heat map will be used to visualize the SHAP values for the neural network regression models.

The SHAP package is commonly used by many for machine learning interpret ability. In the aerospace community, the following works have made use of the interpretability that the SHAP package provides [46, 47, 48]. Researchers have also used the SHAP package for work in traffic engineering [49], structural engineering [50, 51], along with numerous other machine learning applications [52, 53, 54, 55, 56, 57, 58].

The second application of machine learning for this thesis is the development of surrogate models used to predict thrust-time curves. Neural networks developed and trained using TensorFlow [41] is used to create these surrogate models. Surrogate models have been applied for numerous aerospace application such as the following works [59, 60, 61, 62, 63, 64, 65]. Carpenter and Hartfield have used similar methods to predict thrust-time curves for straight SRM grains [66] and similar work has been published for tapered grains [18]. This work extends the analysis of [18] to a more advanced data set, improving the capabilities of the modeling and simulation effort.

For the surrogate modeling effort, the data is split into training and testing data. This testing and training split is defined by the user, typically it is beneficial to train on more data than you are testing with. Typically for this work, the training percentage is somewhere around

70-90 percent with the testing data being around 30-10 percent. The data set that is used for the surrogate modeling comes into effect as well. As mentioned in [42] the surrogate model, or neural network in this case, is essentially just fitting the data provided. Keep in mind that the surrogate model is only going to be as good as the data it is trained with, the model is not expected to accurately predict far outside the bounds of the data trained on.

To apply the neural networks to the data set to act as a surrogate model, we need to first define the inputs and outputs of the surrogate model. The inputs and outputs of a surrogate model can be thought of as the predictor and response variables from a linear regression model. For the thrust-time curves the inputs to the neural network surrogate model will be the time, and any of the input variables that were varied using the Latin Hypercube distribution. For the data sets shown in Section 3.4, the inputs are DBDODY, RPVAR, RP_TR, DENSITY and THROAT for the CP grain and for the star grain the parameter RI_TR is included with that list. The output of the model is the thrust, keep in mind that this output of thrust is at each individual time step.

To check the accuracy and capabilities of the surrogate model, the testing data or the hold-out data from Figure 4.3 is used. The thrust-time curves can be plotted together to see the predicted curve vs. the true thrust-time curve. In this case, the truth data is the data that comes from the 1D internal ballistics code. Just visualizing the thrust-time curves is not enough to show the ability of the model, we will also look at the residuals in thrust to see how well the neural network does at modeling the thrust-time curves. The $R^2$ is not used for this analysis due to the fact that the surrogate models are fitting a large number of data points, and typically have $R^2$ values greater than 0.98. For this reason the $R^2$ is not a very informative metric for this surrogate modeling task. Plots of the residuals in thrust can be very informative when trying to better understand the performance of the model. For these plots, the residual is simply defined as shown in Equation 4.7.

$$r = y_{pred} - y_{truth} \tag{4.7}$$

In Equation 4.7, shown above, $y_{pred}$ represents the thrust predicted from the neural network surrogate model, while $y_{truth}$ represents the thrust values from the thrust curves generated by the 1D internal ballistics code. An example plot for residuals should look like what is seen in Figure 4.4 residual.

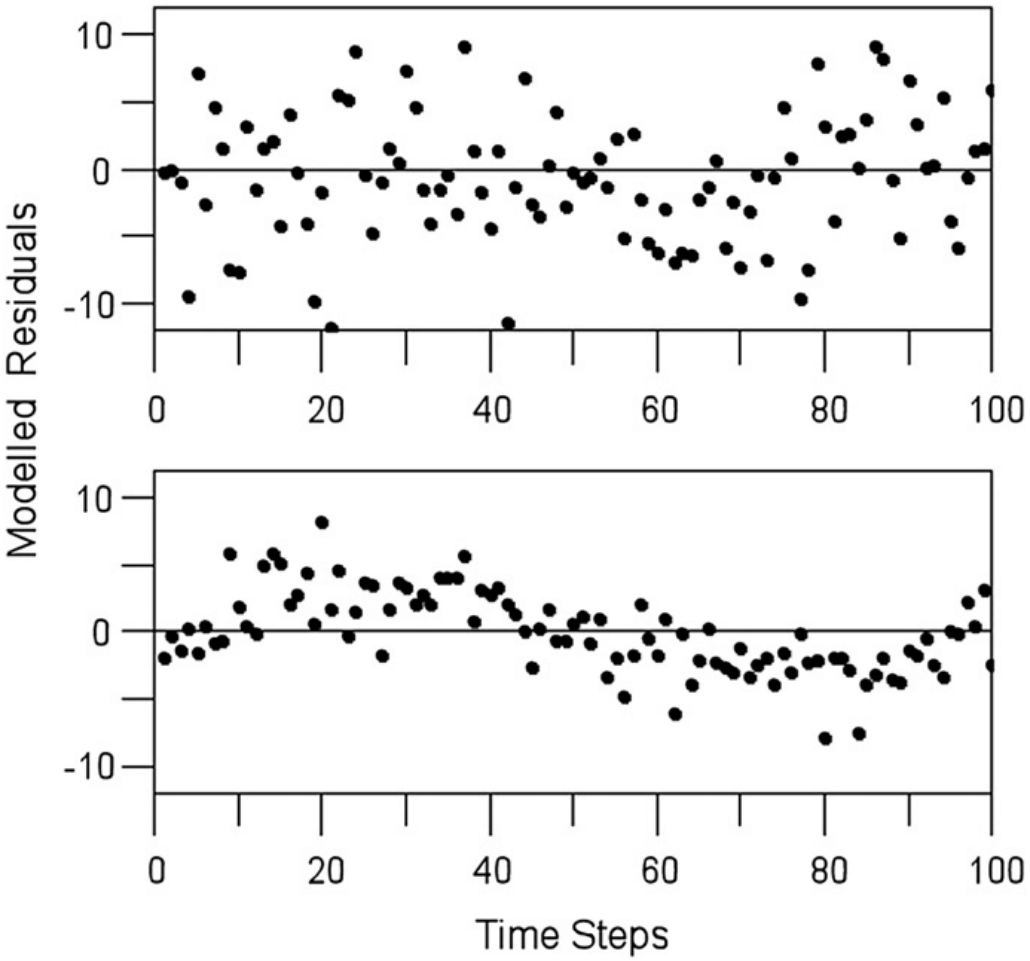



Figure 4.4: Example Residual Plot [67]

Figure 4.4 shows what an example of residuals. The residuals of the response should be centered around zero. With a perfect predictor all of the residuals would lie on the horizontal line at zero. Obviously, there is going to be error in the model that leads to the residuals being greater than zero. Nonetheless, we should still want to see the residuals centered around the horizontal line at zero.

In an effort to improve the initial surrogate models, the Keras Tuner [32] will also be used for the surrogate modeling task. The Keras Tuner develops and trains more optimal neural network designs given the input data. After the initial models have been developed for the

surrogate modeling task, the Keras Tuner will be used to produce models that will be compared against the original models. Now that we have discussed and explained the statistical learning techniques to be used in this paper, we can now show and explain the results in the next chapter.

# Chapter 5

# Statistical Learning Results

To begin the machine learning analysis, large databases of data were generated for this project. Section 3.4 provides a detailed overview of the data that is used for this thesis. It is important to note that the statistical learning results will be a result of the data that is chosen for the project. To develop different results, it is important to consider what data the statistical learning techniques have been applied to. The data that we will be looking at for this thesis is the tapered star grain SRM data that was explained in Section 3.4. Like previously mentioned, the statistical learning goals of this thesis are to (1) perform a regression analysis on performance metrics using linear regression techniques and neural networks, and (2) train neural networks to act as surrogate models for the 1D internal ballistics code. The SHAP package [31] will be used to help interpret these machine learning models developed for the regression task.

## 5.1 Regression Analysis and SHAP

The first results that will be shown as part of this thesis is the regression analysis section and applications of the SHAP package. This is a regression analysis for the response variables of maximum thrust, average thrust, burn time, and total impulse. The results for the CP grain SRM data will be shown first, followed by the star grain results.

### 5.1.1 CP Grain Results

The first section of results for this thesis will be on the regression analysis and the use of SHAP [31] to help interpret the regression models for the CP grain SRM data. The regression model

will be trying to predict the performance values of maximum thrust, average thrust, burn time, and total impulse. The first results we will look at are the first order regression results. Table 5.1 shows the $R^2$ scores for the models developed on the CP grain data set. Table 5.1 shows that for each of these models, the $R^2$ scores are all high. Of course the $R^2$ improves from first order to second, and then from second order to the neural network. There are circumstances when the first and second order models may not perform this well, these upcoming sections will discuss the results for each model as well as explaining what is gained with each model.

Table 5.1: Circular Perforated Grain Model Summary

| Model Type | $R^2$ |
|---|---|
| First Order | 0.981 |
| Second Order | 0.993 |
| Neural Network | 0.997 |

First Order Regression

Like mentioned previously, the response variables of choice for this work are the maximum thrust, average thrust, burn time, and total impulse. The first order linear regression model was developed in Scikit-Learn [40]. The output of the linear regression function developed for this work is the intercept, regression coefficients, and standardized regression coefficients. For the regression modeling we will focus on showing the standardized regression coefficients, these are free of units and thus easier to interpret. Figure 5.1 shows a heat map of the standardized regression coefficients for this first order model.

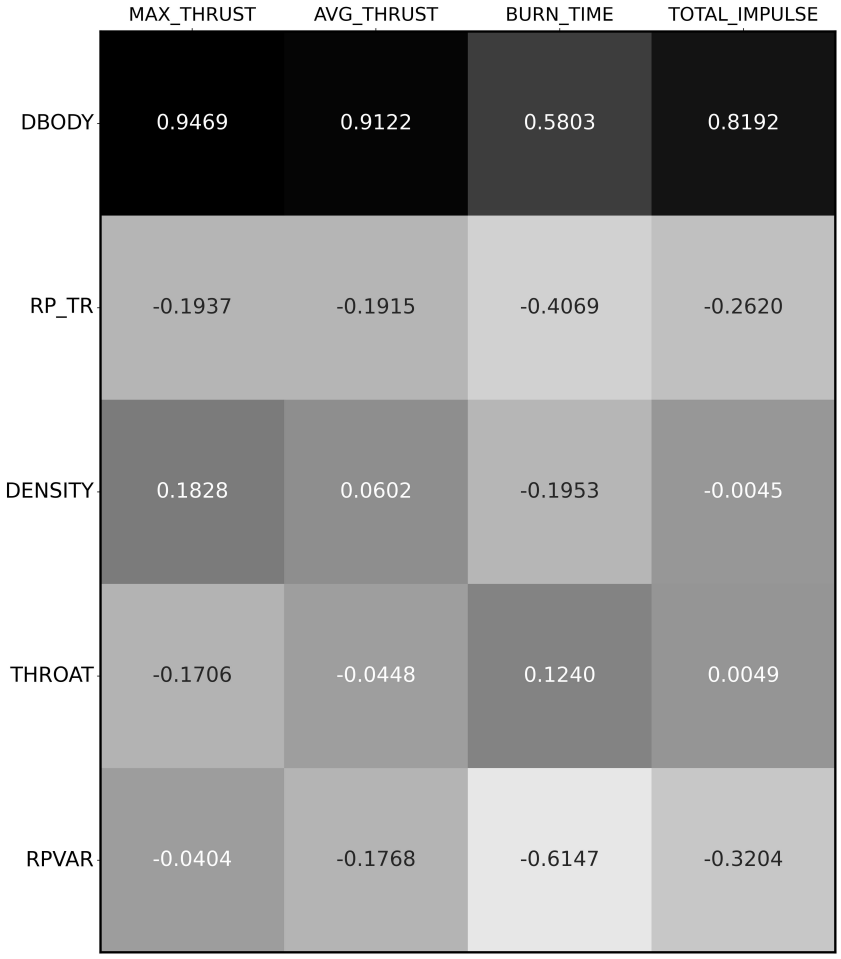
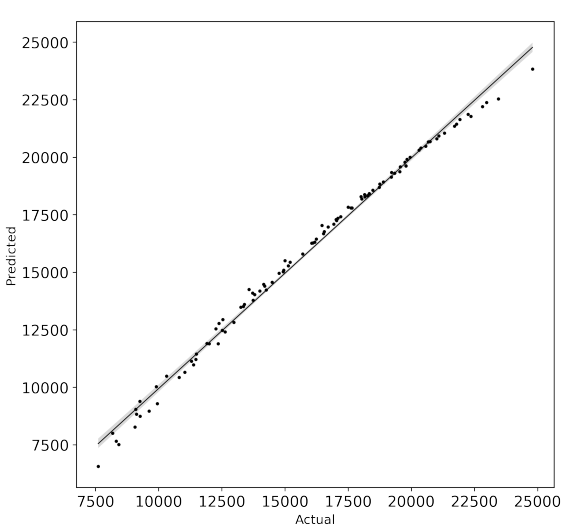



Figure 5.1: 1st Order Standardized Regression Coefficients

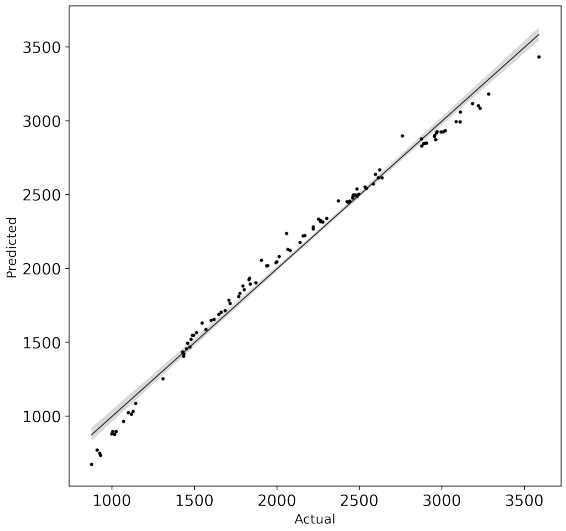
Looking at Figure 5.1, we can see that the body diameter (DBODY) has the largest effect on the response variables of maximum thrust, average thrust, burn time, and total impulse for these tapered SRMs. The parameter DBODY also positively effects the response variables for this data set. Looking at Figure 5.1 we can also see the effects of the other predictor variables, on the response variables for this data set. The propellant radius taper ratio (RP_TR) has a large negative effect on the response values for the data set. The parameter THROAT has the largest impact on the model when looking at maximum thrust, the impact is negative and shows that the increase in the THROAT parameter leads to a decrease in maximum thrust. The THROAT parameter also has a positive impact on the BURN_TIME, this shows that the increase in the throat area leads to longer burns. Again, this this makes physical sense, a larger throat area will lead to lesser chamber pressures (on average) and that leads to lesser burn rates. The predictor variable RPVAR, the parameter controlling the size of $R_p$ has a negative effect on the four

response variables. This shows that larger $R_p$ can lead to smaller thrust, total impulse, and burn rate. The larger $R_p$ means there is typically less propellant to burn, so assuming constant inputs, it is expected to see the trend shown above.

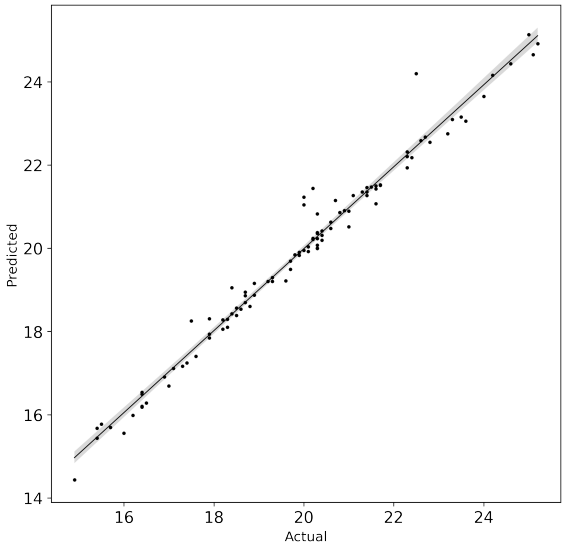
After looking at the standardized regression coefficients for the first model, shown in Figure 5.1 we can check plots to see how well the data is being fit. Figure 5.2 shows the predicted vs. actual plots for the first order CP grain regression model.



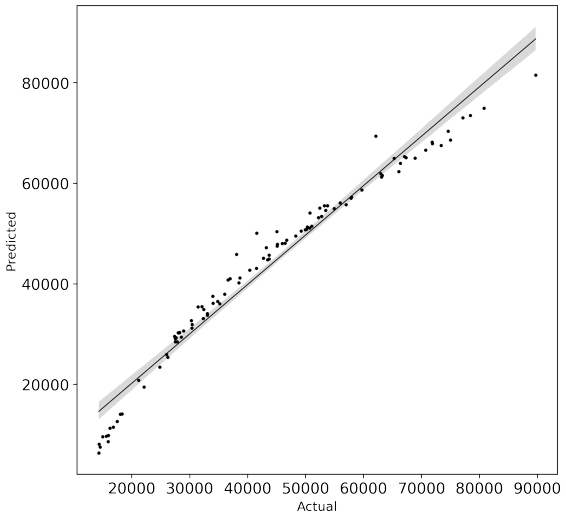

(a) Maximum Thrust $(lbf)$




(b) Average Thrust $(lbf)$




(c) Burn Time $(sec)$




(d) Total Impulse $(lbf - sec)$

Figure 5.2: Predicted vs. Actual for CP Grain First Order Linear Regression Model

Looking at Figure 5.2 there is some curvature on the total impulse plot. The fits for the thrusts and for the burn time could also be better. We can now move on to a second order

regression model, the second order model will include interaction terms giving us a better understanding of the physics taking place. The second order model should also improve the predicted vs. actual plots shown in Figure 5.2.

Second Order Regression

Now that we have shown the results for the first order regression, we will look to improve the model by using second order regression. The second order regression will be used to develop models for maximum thrust, average thrust, burn time, and total impulse for solid rocket motors. The full regression equations will not be written out due to the length, but the regression coefficients will be shown below. Again, like when we looked at the first order model we will again look at the standardized regression coefficients. Figure 5.3 shows the standardized regression coefficients for the second order regression model.

| | MAX_THRUST | AVG_THRUST | BURN_TIME | TOTAL_IMPULSE |
|---|---|---|---|---|
| DENSITY DBODY | 5.5815 | -0.2227 | -3.5115 | -4.6473 |
| DBODY | -3.6950 | 1.9785 | 4.4193 | 7.0394 |
| RP_TR | 3.1868 | 1.9890 | -4.3773 | 1.4921 |
| DENSITY RP_TR | -2.6065 | -0.3958 | 7.1089 | 1.1474 |
| RPVAR | 1.2900 | 2.1711 | -2.4527 | 2.7733 |
| DENSITY RPVAR | -1.0394 | -1.0393 | 3.3659 | -0.7402 |
| DBODY RP_TR | -0.8715 | -1.0444 | 0.3313 | -1.4768 |
| DBODY RPVAR | -0.6774 | -1.7964 | -0.6755 | -3.2102 |
| DENSITY THROAT | -0.4004 | 1.4199 | -0.2128 | 1.9837 |
| RP_TR THROAT | 0.3837 | -0.3520 | -3.4021 | -1.2178 |
| THROAT RPVAR | 0.2828 | -0.1213 | -1.5247 | -0.3611 |
| RP_TR RPVAR | -0.2524 | -0.3906 | -0.2495 | -0.2516 |
| DBODY THROAT | -0.1368 | 1.2891 | 0.2228 | 2.0742 |
| THROAT | 0.0719 | -1.8897 | 1.5531 | -2.4844 |
| DENSITY | -0.0178 | 0.0014 | -0.3964 | 0.1172 |

Figure 5.3: 2nd Order Standardized Regression Coefficients

Looking at Figure 5.3 we can see the standardized regression coefficients for the second order model. The output of maximum thrust has been sorted in order of importance, this may throw off the ordering for the magnitude of the impact for other response variables.

First looking a the maximum thrust of the CP grain SRM. A new predictor variable combination has shown to be the most important in affecting the model. The combination of DENSITY and DBODY has the largest impact on the maximum thrust. That combination is followed by the body diameter and the RP_TR on the impact of the model. One thing to note is that the DBODY now has a negative impact on the maximum thrust of the SRM for this second order model. For the other three response variables, the impact is still positive when considering DBODY. The predictor variable that has the largest impact on the average thrust is the RPVAR parameter. The combination of DENSITY and RP_TR has the largest impact on the burn time of the SRM, and the body diameter (DBODY) has the largest effect on the total impulse.

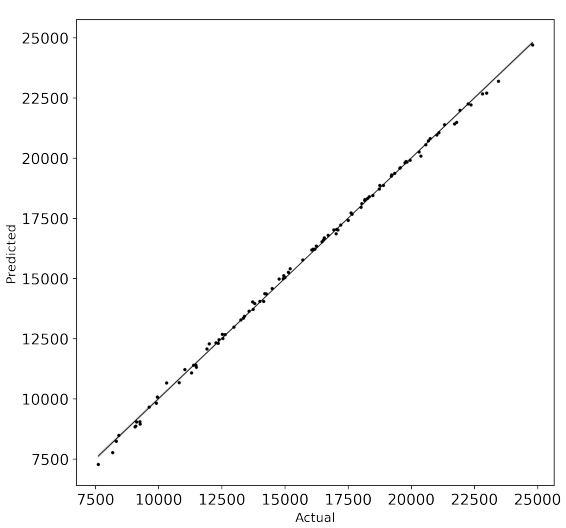
Figure 5.3 gives us insight about the model that was unavailable when just considering the first order regression model. The second order model showed the important interactions between density and geometric parameters that have an effect on the SRM performance. Now that we have shown the second order regression model we can move on to the final model, a neural network model. Figure 5.4 shows the predicted vs. actual plots for the second order regression data. We can see that some of the curvature shown in Figure 5.2 has been eliminated with the second order model shown in 5.4. To try and improve the model fit even more, a neural network will be trained as part of this regression analysis.

### Neural Network

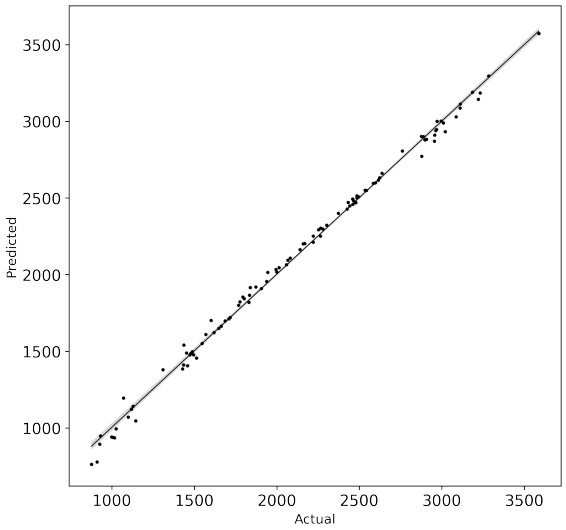
Now that we have looked at regression models, in an effort to develop a better model, we can show the results of the neural network that was trained on the CP grain results. Table 5.2 shows the neural network architecture used in TensorFlow to create this model.

Table 5.2: Circular Perforated Grain Neural Network Architecture

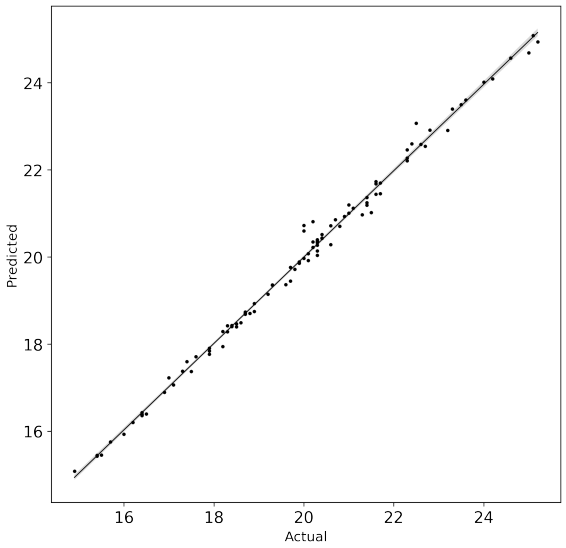
| Design Parameter | Value |
|---|---|
| Hidden Layers | 2 |
| Units Per Layer | 100 |
| Epochs trained | 5000 |



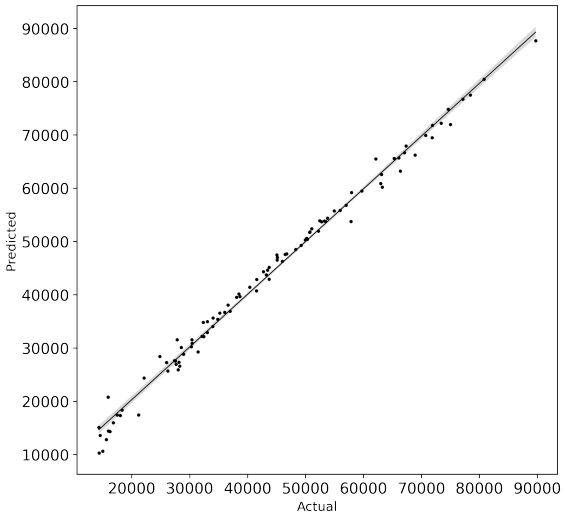

(a) Maximum Thrust $(lbf)$




(b) Average Thrust $(lbf)$




(c) Burn Time $(sec)$




(d) Total Impulse $(lbf - sec)$

Figure 5.4: Predicted vs. Actual for CP Grain Second Order Linear Regression Model

Using the neural network architecture proposed in Table 5.2 a model for the CP grain response variables was developed and trained. The activation function used was Exponential Linear unit, or ELU. The network was trained by minimizing the validation loss, and the loss metric was the mean absolute percentage error (MAPE). The deep neural network is often times though of as a so called black-box, for this reason SHAP values [31] are helpful when interpreting these machine learning models. Figure 5.5 shows the SHAP values for the CP grain neural network model.

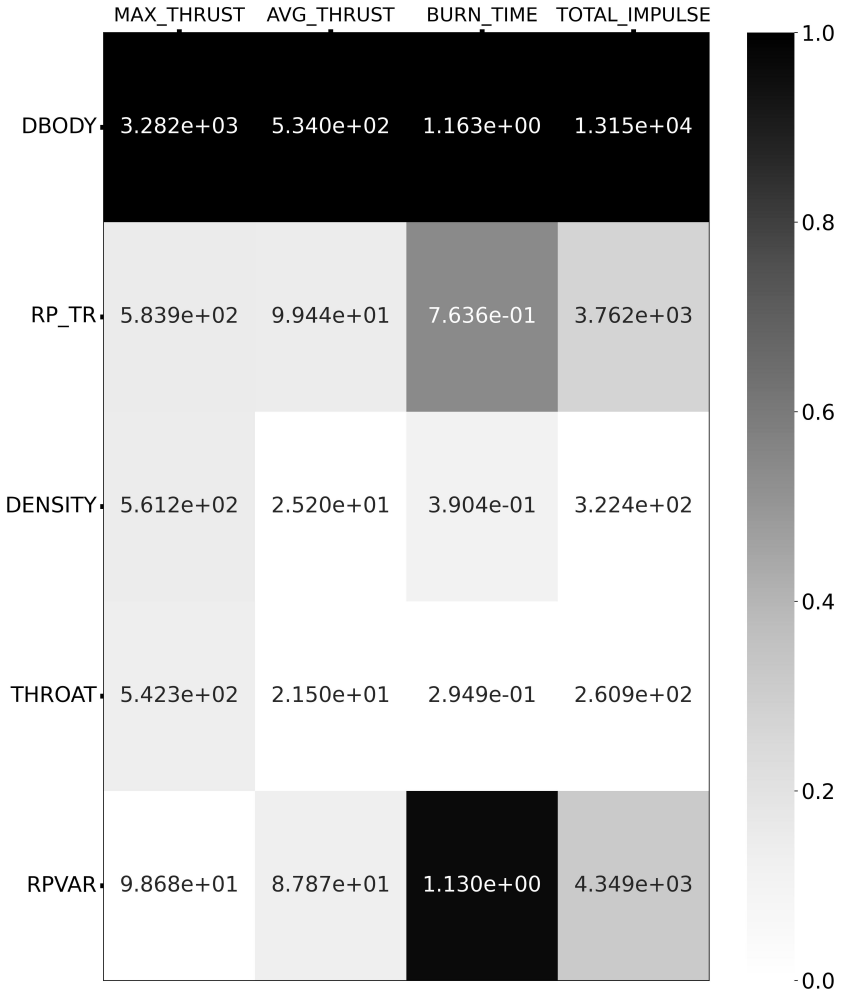
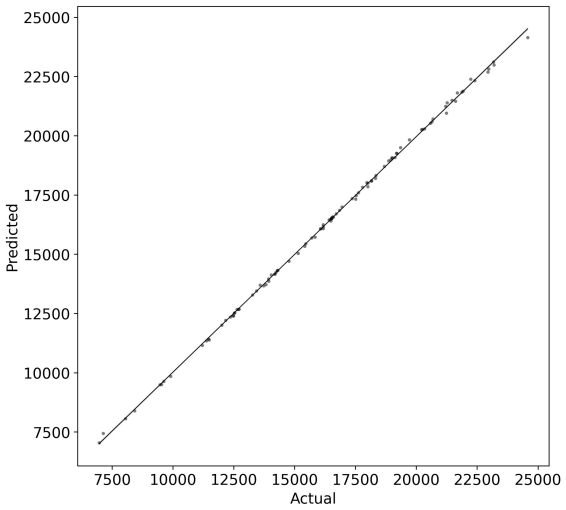


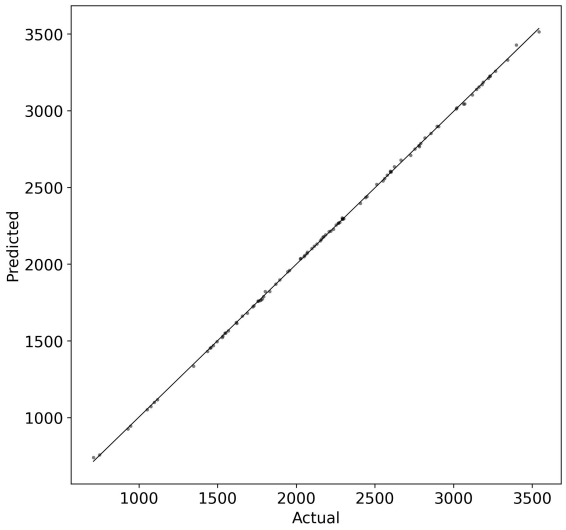

Figure 5.5: CP Grain SHAP Heatmap

For the neural network architecture, we have to rely of the SHAP values as a way to interpret the machine learning model. The linear regression models benefit from having standardized regression coefficients that can easily be interpreted to show the effect that the predictor variables have on the response.

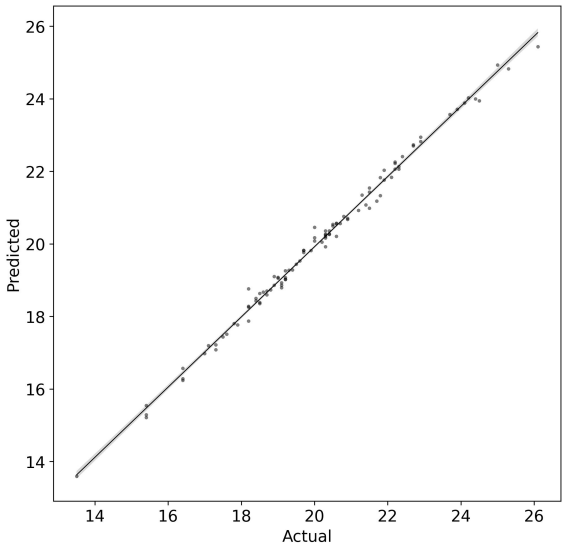
Looking at Figure 5.5, we can see that across the board that the parameter DBODY has the largest effect on the outputs of maximum thrust, average thrust, burn time, and total impulse. This makes physical sense, we expect that rockets with larger diameters to produce more thrust, burn for longer, and as a result have larger total impulse. When looking at most of the response variables, the RPVAR has a low impact, but for the burn time the RPVAR plays a large role. This makes sense when considering the grain design, the larger the bore through the center of the grain is, the shorter burn time can be expected.



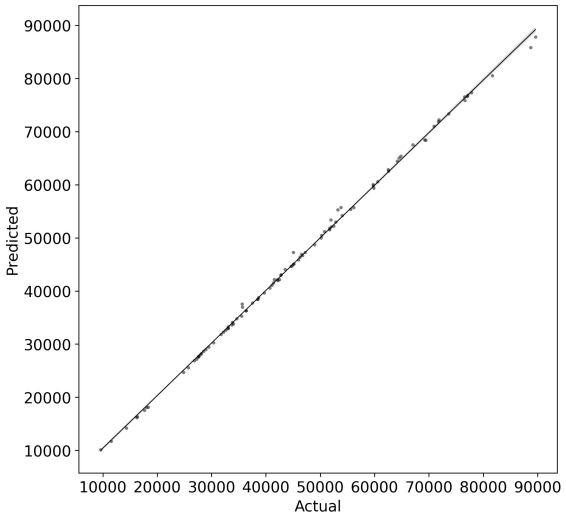

(a) Maximum Thrust $(lbf)$




(b) Average Thrust $(lbf)$




(c) Burn Time $(sec)$




(d) Total Impulse $(lbf - sec)$

Figure 5.6: Predicted vs. Actual for CP Grain Neural Network Model

Figure 5.6 shows the predicted vs. actual plots for the CP grain neural network model. The predicted vs. actual plots are best when looking at the neural network compared with the linear regression model. The results of the CP grain analysis show that the neural network model produces the best fit while also having a slightly better $R^2$ score. For this CP grain regression analysis, the neural network is the top performer.

### 5.1.2 Star Grain Results

Now that the regression analysis results have been shown for the CP grain data, the results from the star data can be shown in the following sections. A first and second order model will be developed, and finally a neural network will be trained to act as a regression model for maximum thrust, average thrust, burn time and total impulse. Table 5.3 shows the $R^2$ scores for each of the models developed in this section. The $R^2$ improves from first to second order, and then from the second order model to the neural network model. Regardless, the $R^2$ scores show that each of these three models predict the variability in the data set well. These following sections will focus on explaining what can be gained with each model, and comparing results between the models.

Table 5.3: Star Grain Model Summary

| Model Type | $R^2$ |
|---|---|
| First Order | 0.979 |
| Second Order | 0.981 |
| Neural Network | 0.996 |

#### First Order Regression

The first model that is considered for the modeling of the star grain data set is a first order regression model for the response variables of maximum thrust, average thrust, burn time, and total impulse. The star grain results section will follow the same procedure as shown in the CP grain results section. For the star grain data set, a new predictor variable, RI_TR, was included in the data set. Figure 5.7 shows the first order standardized regression coefficients for this data set.

Looking at Figure 5.7, the standardized regression coefficients are shown. Like was shown for the CP grain data set, the DBODY has the largest effect on the response variables for this data set. The tapering parameters RP_TR and RI_TR play a smaller role in the effects that have on the response variables. The density has the largest effect on the maximum thrust when comparing it with other response variables. Now that the first order regression model has been developed and shown, a second order model was developed to capture any interaction effects

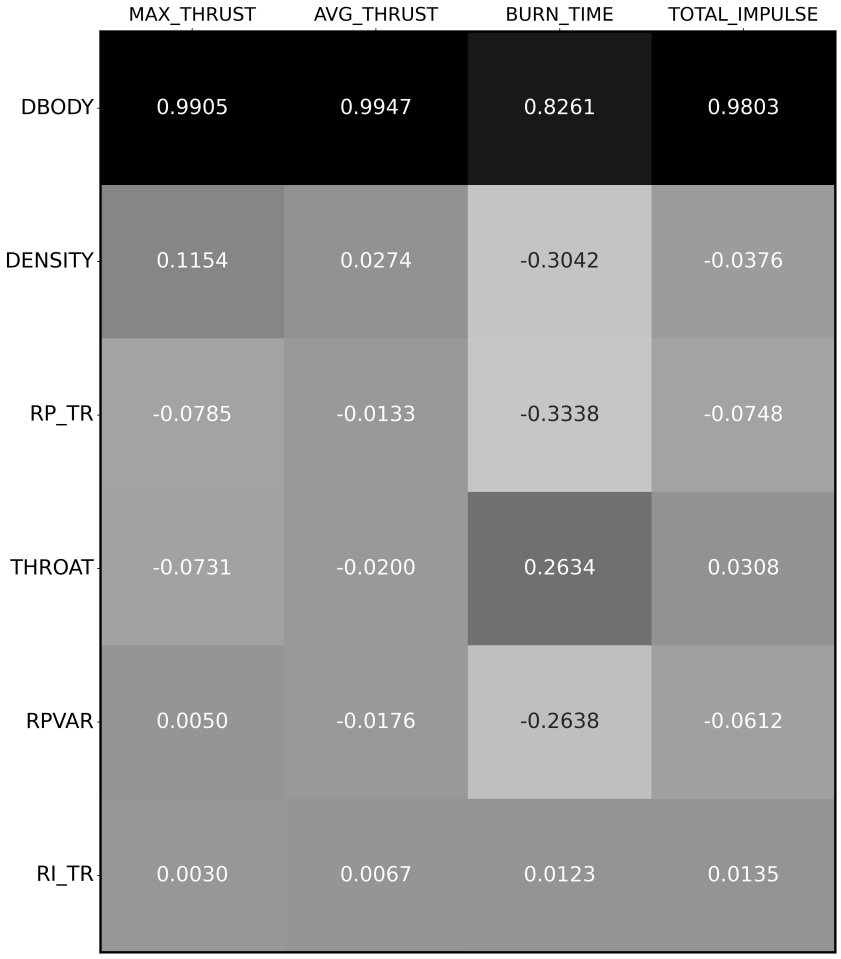
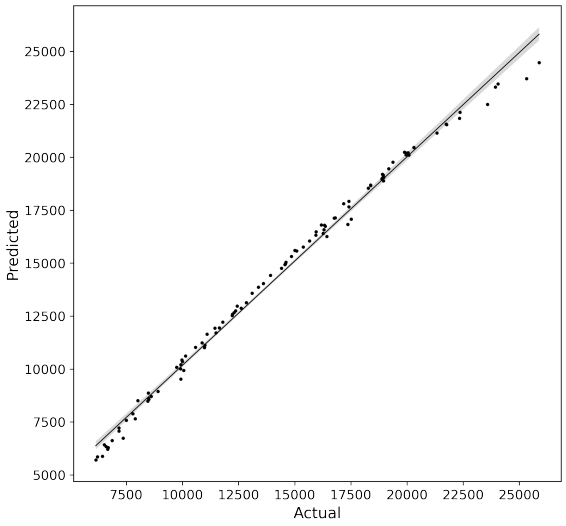
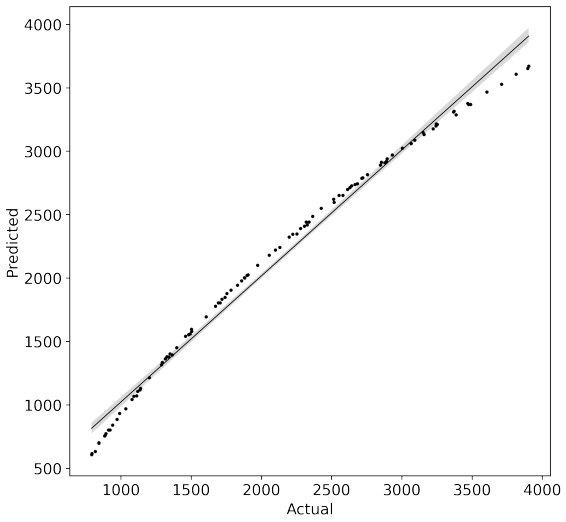



Figure 5.7: First Order Standardized Regression Coefficients

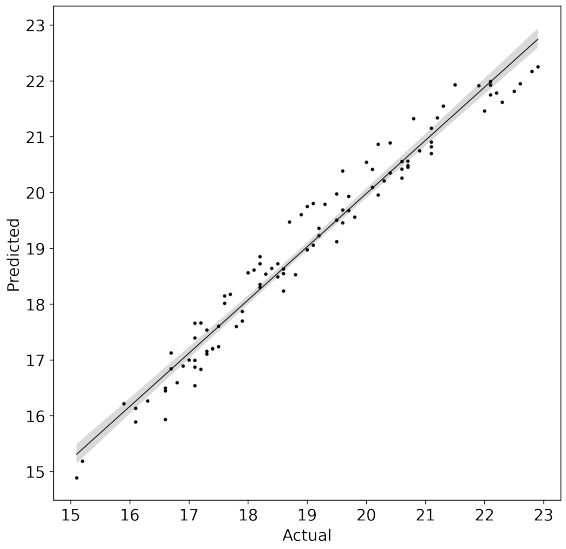
not seen in the first order model. Figure 5.8 shows the predicted vs. actual plots for the first order star grain data. Looking at Figure 5.8 some curvature can be seen for the total impulse, average thrust, and even the slightly for the maximum thrust. The burn time does not seem to show any curvature, but the fit is not as tight. To try and attempt to remove the curvature from the model, a second order linear regression model will be used.



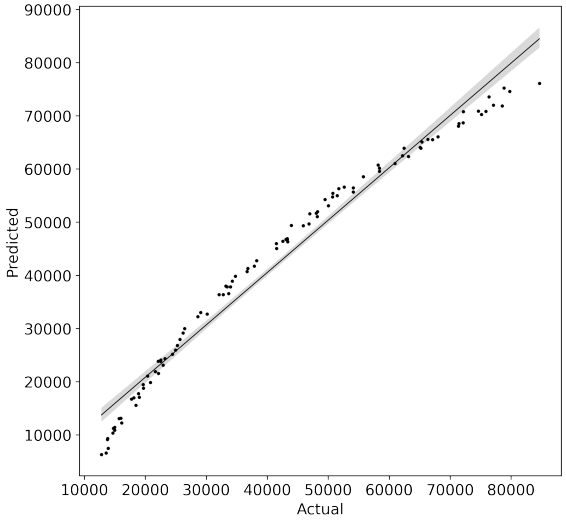

(a) Maximum Thrust $(lbf)$




(b) Average Thrust $(lbf)$




(c) Burn Time $(sec)$




(d) Total Impulse $(lbf - sec)$

Figure 5.8: Predicted vs. Actual for Star Grain First Order Linear Regression Model

### Second Order Regression

Like mentioned above, the second model shown for the star grain data was a second order linear regression model. The second order model includes interaction terms that are not seen in the first order model. Figure 5.9 shows the standardized regression coefficients for the second order model.

Looking at Figure 5.9, the predictor variables are sorted as their order of importance for maximum thrust. Looking at the first column of the heat map, the most important predictor for

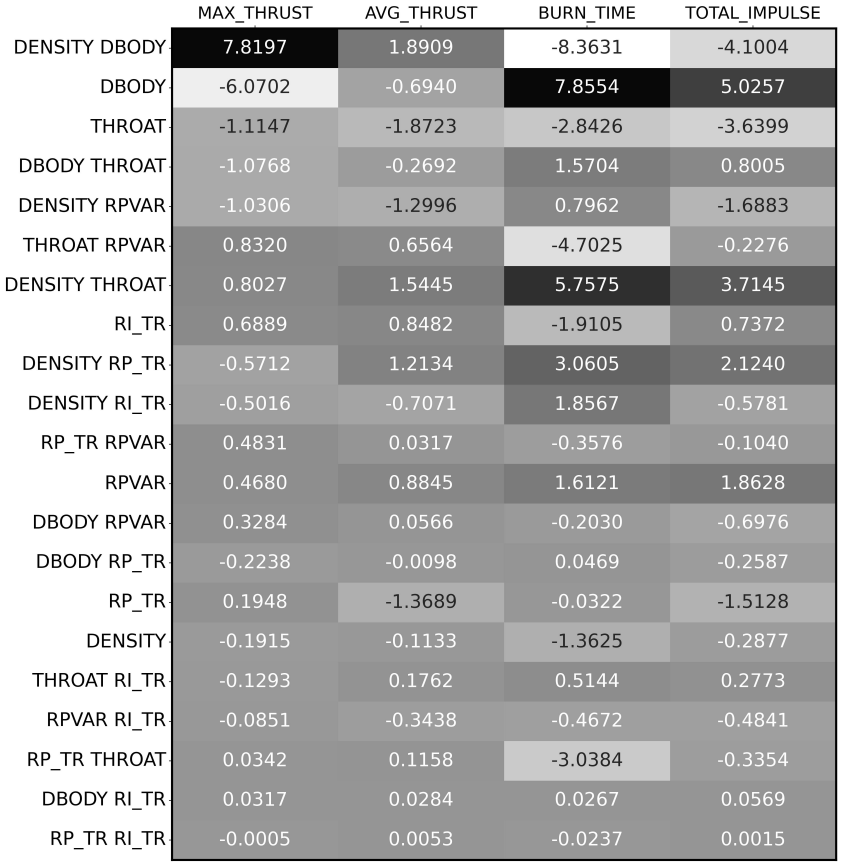
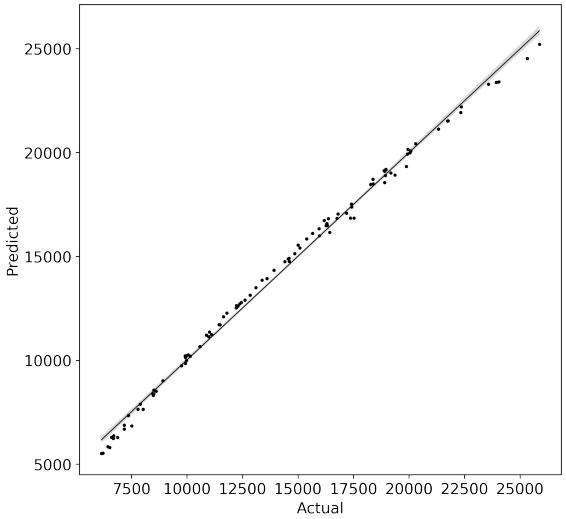



Figure 5.9: First Order Standardized Regression Coefficients

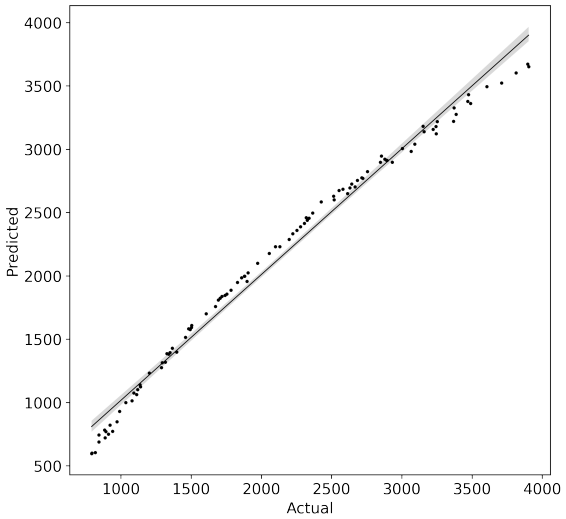
the response is the interaction between density and body diameter. This interaction could not be seen with the first order model alone. The interaction between propellant density and the body diameter is followed by DBODY and THROAT for the maximum thrust. When looking at the average thrust, the important parameters to consider are the interaction between density and DBODY, the THROAT parameter, and the interactions of density with both RPVAR and THROAT. These responses would have been unseen if only investigating the first order model.

The star grain results show the THROAT having a larger impact when compared to the CP grain results shown in Figure 5.3. This seems to show that for a similar sized body rocket, the throat area (or area ratio in this case) seems to be more sensitive for a star grain when compared to a CP grain. Figure 5.10 shows the results of the predicted vs. actual for the second order star grain regression model. The second order model still shows some curvature for the total impulse and the average thrust. To look at improving these results, a neural network model

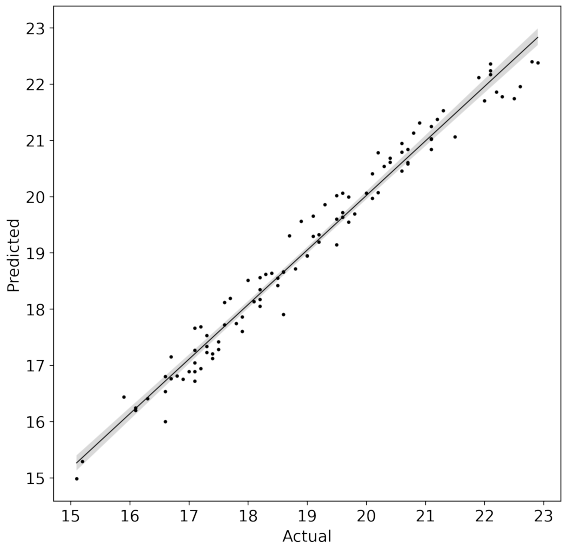
is trained to solve this regression problem. The neural network model is shown in the next section.



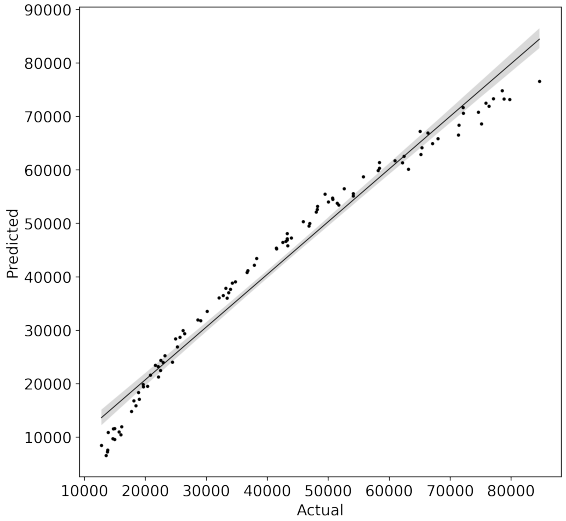

(a) Maximum Thrust $(lbf)$




(b) Average Thrust $(lbf)$




(c) Burn Time $(sec)$




(d) Total Impulse $(lbf - sec)$

Figure 5.10: Predicted vs. Actual for Star Grain Second Order Linear Regression Model

### Neural Network

The final model that we will consider for the regression analysis is a neural network model. A neural network was developed and trained in TensorFlow [41] to predict the response variables, given the predictor variables. Table 5.4 shows the architecture that was used for the neural network model.

Table 5.4: Star Grain Model Architecture

| Design Parameter | Value |
|---|---|
| Hidden Layers | 2 |
| Units Per Layer | 50 |
| Epochs trained | 5000 |

Looking at Table 5.4 we can see that this network has been developed with two hidden layers, with 50 units per layer. Since this network was a little smaller it was trained for 5000 epochs. The ELU activation function was used for both layers, and the mean squared error was the loss metric to be minimized. The SHAP values for this neural network model can be see below in Figure 5.11.

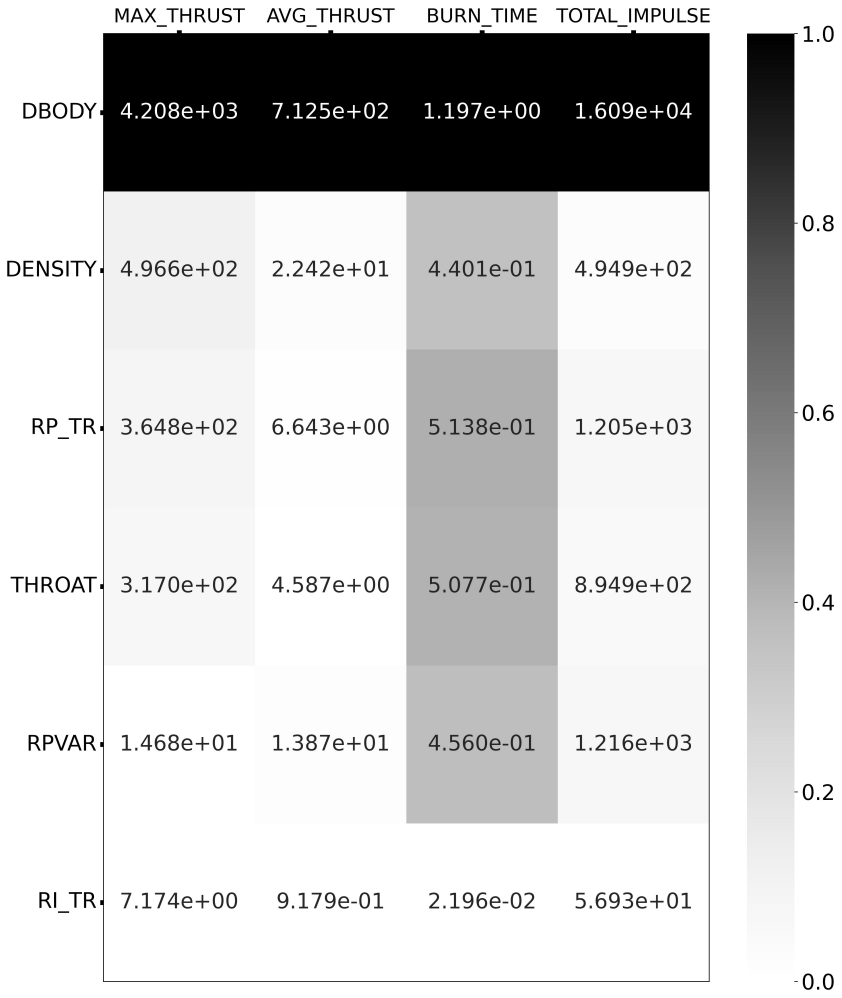
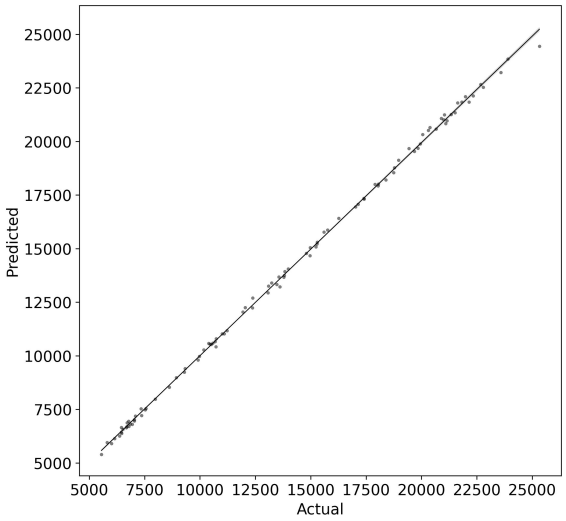



Figure 5.11: Star Grain SHAP Heatmap

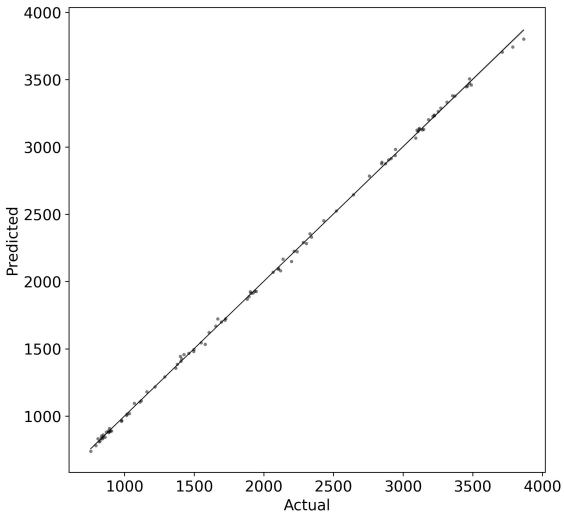
Figure 5.11 shows the SHAP values for the star grain data set generated for this work. Like shown for the earlier regression models, the predictor variable DBODY has the largest impact

on the response values of maximum thrust, average thrust, burn time, and total impulse. The color bar on the right shows the impact that the predictors have on the model. When looking at the maximum thrust, the DBODY is followed by DENSITY and RP_TR in terms of order of importance. When looking at the average thrust, the DBODY is followed by the DENSITY and RPVAR in order if model importance. The burn time shares the same order of importance as with the maximum thrust. Finally, when investigating the total impulse of the star grain SRM designs, it is seen that following the DBODY, RP_TR and RPVAR have the largest impact on the total impulse. This gives us some inclination of the effect that tapering has on the response variables for solid rocket motor simulation.

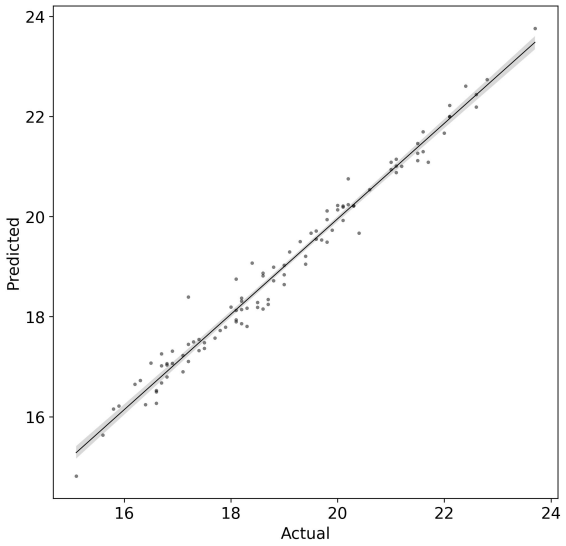
Figure 5.12 shows the predicted vs. actual plots for the star grain regression model. Looking at the figures below, the curvature has been eliminated up with the neural network model. The predicted vs. actual plot for burn time still shows a bit of a spread, but we can still consider this a good prediction.



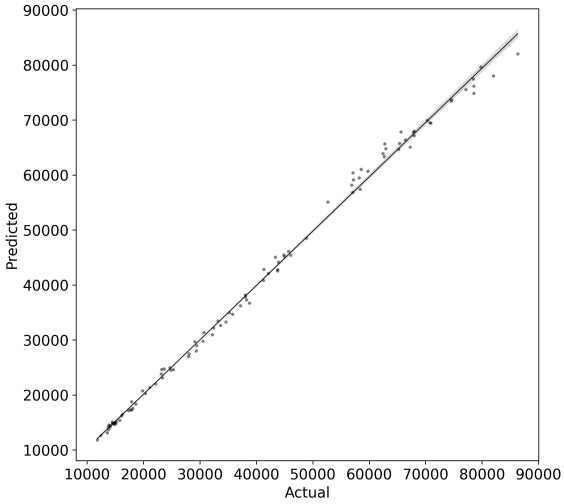

(a) Maximum Thrust $(lbf)$




(b) Average Thrust $(lbf)$




(c) Burn Time $(sec)$




(d) Total Impulse $(lbf - sec)$

Figure 5.12: Predicted vs. Actual for Star Grain Neural Network Model

## 5.2 Surrogate Modeling of Thrust-Time Curves

Now that the results of the regression analysis have been shown for this work. We can move on to the use of neural networks as surrogate models for thrust-time curves. TensorFlow [41] is used to develop and train neural networks on the output of the 1D SRM internal ballistics code. The results of this section follows the statistical learning methods shown in [18], with changes made to the internal ballistics solver and the resultant data set.

### 5.2.1 CP Grain Results

Now that we have discussed the first task of the regression analysis, we can move on to the second task of developing the surrogate models for the SRM thrust-time curves. Neural networks for exclusively used for the task of developing and training these models. Table 5.5 shows the neural network architecture that was used to develop the tapered CP grain surrogate model. This neural network was trained my minimizing the validation loss, and the loss metric used here was the mean squared error (MSE). The mean squared error was chosen for this neural network, since the MAPE can have issues with convergence when data values are near zero. Some of the thrust-time curve data points can take on small values, so the MSE was chosen as the loss metric for this model. The exponential linear unit activation was used for this neural network architecture.

Table 5.5: Circular Perforated Grain Surrogate Model Architecture

| Design Parameter | Value |
|---|---|
| Hidden Layers | 3 |
| Units Per Layer | 100 |
| Epochs trained | 1000 |

To train the neural network model, training data was set aside to split the data set into test data and training data. One-hundred thrust-time curves were held out of the training set to act as testing data. Because the model was trying to predict the full thrust-time curves, it is important that the time series data was kept in the correct order.

The inputs to the neural network model were the SRM code inputs that had been varied using the Latin hypercube scheme, along with the time vector at a uniform time step. The outputs that the network was trained on was the thrust at resultant time step. Once the model was trained using TensorFlow, the testing data set could be used to see how well the model performed. Figures 5.13 and 5.14 show some of the thrust-time curves that were used to test the neural network model. Figures 5.13 and 5.14 show the results of the model when predicting inputs from the testing data. These curves shown below are part of the testing data, and were not trained on when developing the model.

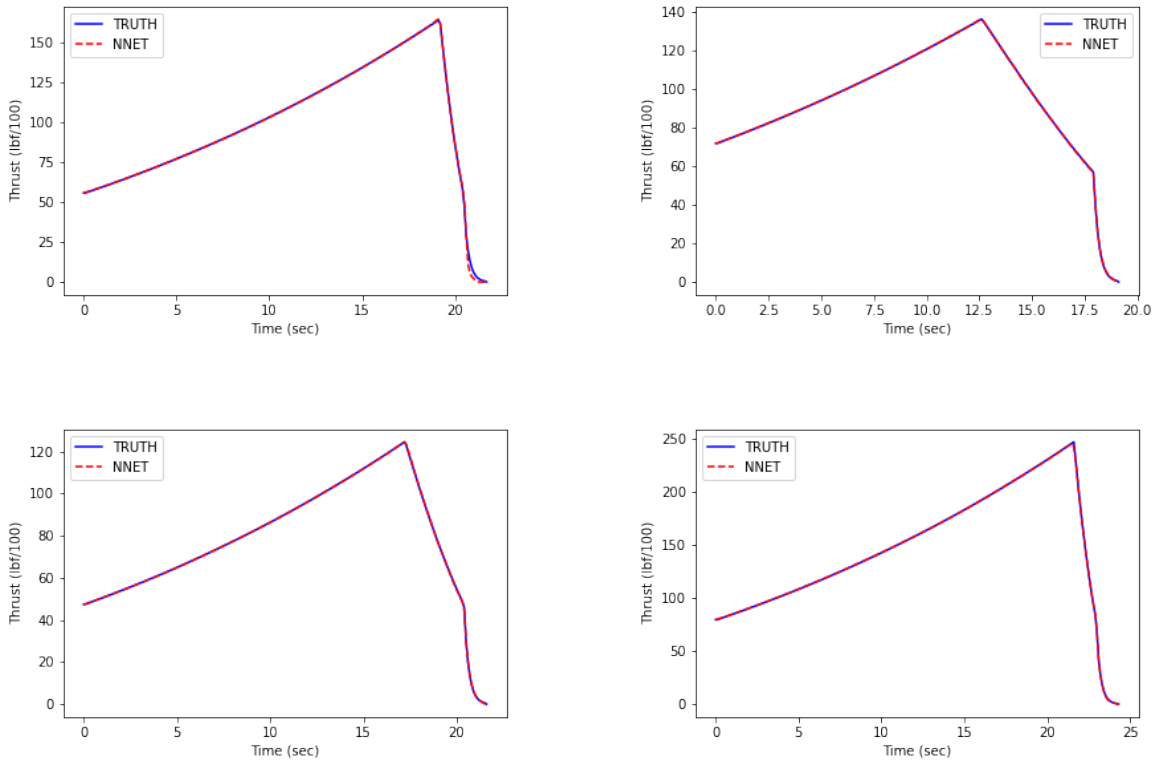



Figure 5.13: CP Grain SRM Modeling Results

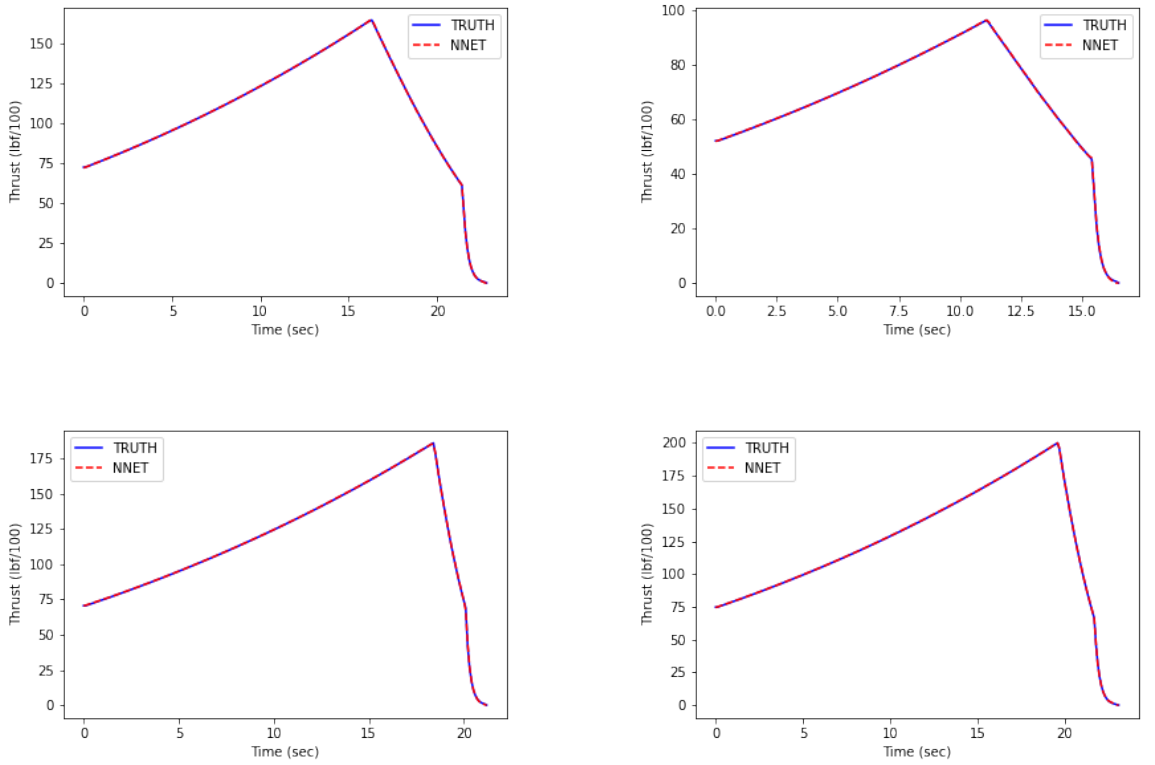



Figure 5.14: CP Grain SRM Modeling Results

Visually inspecting Figures 5.13 and 5.14 it seem that a good surrogate model has been generated. Inspecting the $R^2$ scores of the testing data shows that nearly each of these thrust-time curves has an $R^2$ greater than 0.98. Just inspecting the models visually that is not enough to say that this is a good surrogate model. Figure 5.15 shows the residuals in thrust for this CP grain surrogate model. The residual plot shows in greater detail the error between our model and the truth data.

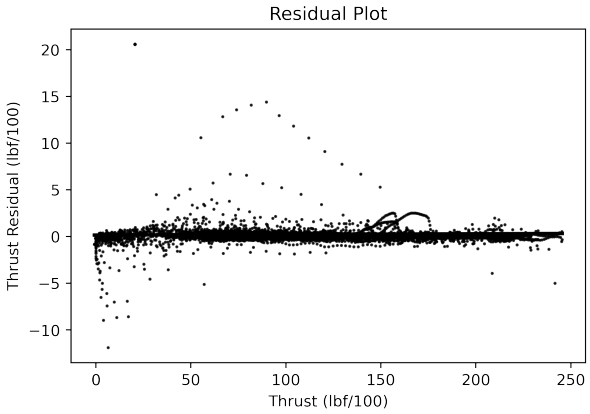



Figure 5.15: Residual Plot for CP grain data

Looking at Figure 5.15, we can see that the majority of the residuals in thrust seem to lie near zero. There are a few outliers in thrust, these are likely due to temporal shifts when predicting the thrust-time curves. A small temporal shift in the predicted curve could lead to large residuals in thrust due to the shape of the thrust-time curve, especially near the peak. Regardless, we are still able to capture the physics of the tapered grain CP SRM data with this surrogate model. Figure 5.16 shows an example prediction that could have lead to larger residuals for the CP grain surrogate model. Inspecting the end of the prediction shown in Figure 5.16, one can see that there is a slight mismatch between the neural network model and the truth curve. Regardless, the physics of the thrust-time profile is still captured with this CP grain surrogate model.

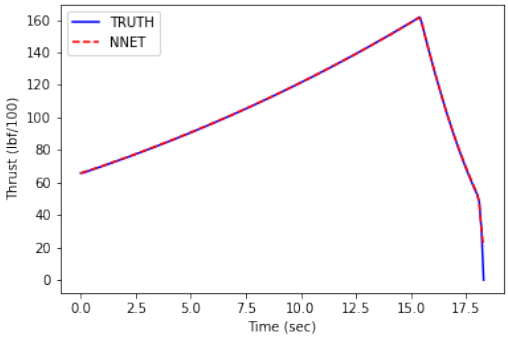



Figure 5.16: CP Grain SRM Modeling Result with larger residual

After developing this first surrogate model, the Keras Tuner [32] was used to find optimal hyper-parameters for another CP grain surrogate model. The Keras Tuner [32] can be used with TensorFlow [41] to find optimal hyper-parameters given a design space. Table 5.6 shows the hyper-parameters developed for this model when using the Keras Tuner.

Table 5.6: Tuned Circular Perforated Grain Surrogate Model Architecture

| Design Parameter | Value |
|---|---|
| Hidden Layers | 3 |
| Units Per Layer 1 | 45 |
| Units Per Layer 2 | 35 |
| Units Per Layer 3 | 5 |
| Epochs Trained | 1000 |

To develop this model the Keras Tuner was used with the random search algorithm to find the optimal hyper-parameters for this model. The random search algorithm tested 30 unique neural network designs while monitoring the validation loss. The loss function used for the network was the mean squared error (MSE). The networks developed from the random search algorithm were trained on for 100 epochs to find the optimal hyper-parameters for the given design space. The design shown in Table 5.6 shows the results of the Keras Tuner random search algorithm. The activation function for the first two layers was exponential linear unit (ELU), while the activaton function used for the thrid layer was the rectified linear unit (RELU). Figure 5.17 and 5.18 shows the modeling results when the Keras Tuner was used on the CP grain data set. Figures 5.17 and 5.18 show the results of the model when predicting inputs from

the testing data. These curves shown below are part of the testing data, and were not trained on when developing the model.

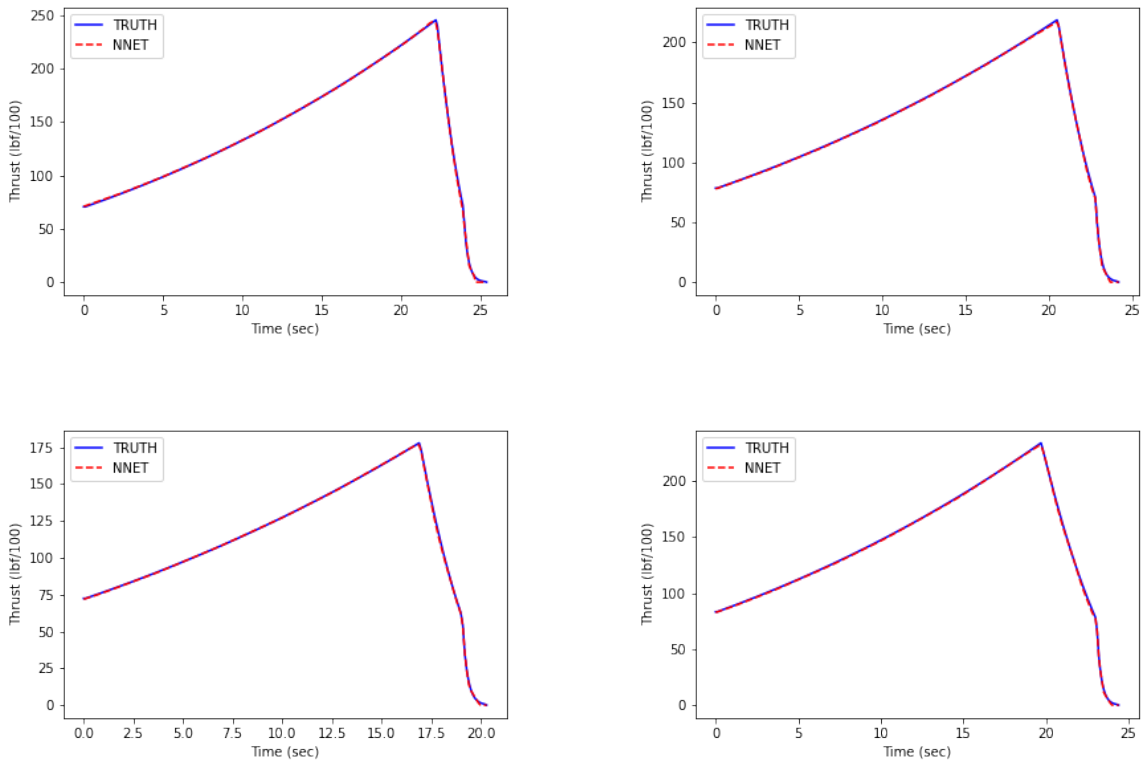



Figure 5.17: CP Grain SRM Modeling Results - Keras Tuner

Looking at Figure 5.17 and 5.18 above, the Keras Tuner [32] model seems to adequately model the thrust-time curves generated for the tapered CP grain SRM design. Like previously mentioned, just showing the images of the fitted thrust-time curves for a limited number of cases is not adequate to show that the model is good. To get a better understanding of the performance of the neural network model will look at the residual plot shown in Figure 5.19.

Figure 5.19 shows the residuals in thrust for the Keras Tuner network trained on the tapered CP grain data. Looking above most of the residuals in thrust lie near zero. There are a few data points that are outliers, and these are likely due to a temporal shift in the prediction of the thrust-time curve. Figure 5.20 shows an thrust-time curve prediction that could have lead to a larger residuals in thrust for both a tuned and untuned model.

Looking at Figure 5.20, near the end of the burn it can be seen that the neural network prediction stops early. This could cause large residuals due to the temporal shift in the truth

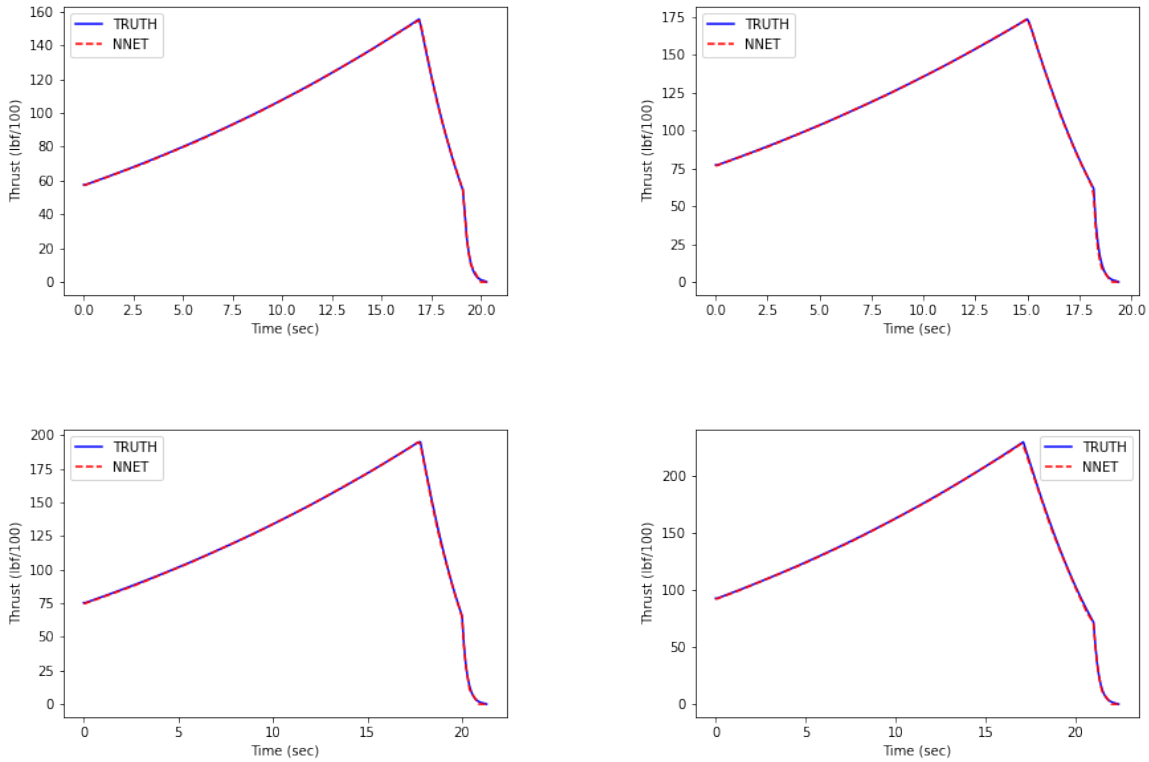



Figure 5.18: CP Grain SRM Modeling Results - Keras Tuner

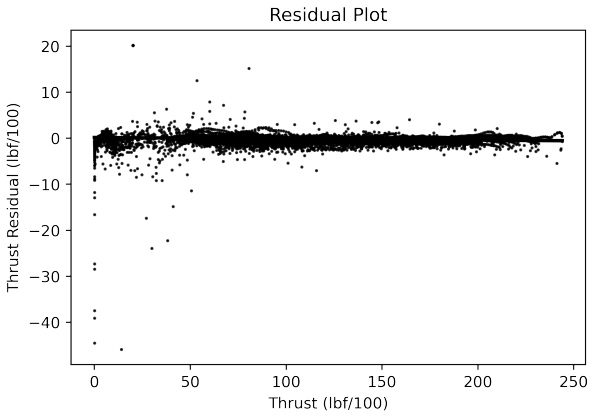



Figure 5.19: Residual Plot for CP grain data - Keras Tuner

data compared with the prediction from the neural network model. Even with the results of the residuals shown in Figure 5.19 this model does well predicting the thrust-time curves for the CP grain data.

Comparing Figure 5.19 with Figure 5.15 we can see the residuals for the tuned model compared with the first model developed. The results shown in the first neural network seem to be favorable when comparing Figures 5.19 and 5.15. This result is not entirely surprising.

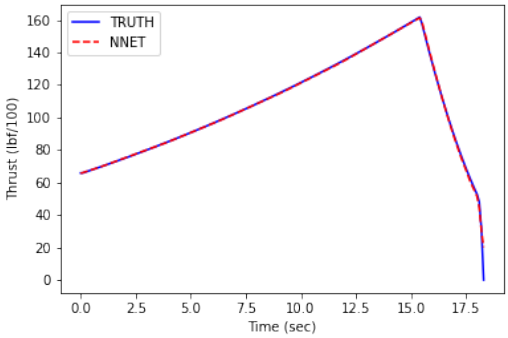



Figure 5.20: CP Grain SRM Modeling Results - Keras Tuner with larger residual

When using the Keras Tuner, the model will only find the local optimum. What that means is based on the Tuner constraints, it will find the best model. For the CP grain data set, the better model may be the model trained with 3 layers and 100 units per layer. Regardless, both models are good choices for predicting CP grain thrust-time curves. Now that the models developed for the CP grain data set have been shown, the star grain surrogate modeling results can be shown in the next section.

### 5.2.2 Star Grain Results

Now that the results have been shown for the CP grain SRM data, we will move on by now looking at tapered star grain designs. These star grain designs follow the data introduced in Section 3.4. To accurately model and predict these thrust-time curves, a neural network was developed and trained using TensorFlow [41]. The training-test data split was 80-20. 80 percent of the data went to training, while 20 percent went to testing. For this data set, that meant the network was trained with 400 thrust curves, while 100 curves were set aside for testing. Table 5.7 shows the architecture of the neural network developed to model these star grain thrust-time curves.

Table 5.7: Star Grain Surrogate Model Architecture

| Design Parameter | Value |
|---|---|
| Hidden Layers | 3 |
| Units Per Layer | 100 |
| Epochs trained | 1000 |

Table 5.7 shows that the neural network model has 3 hidden layers, with 100 units per layer. The activation function used for each layer was ELU, or exponential linear unit. The neural network was trained by trying to minimize the mean squared error (MSE). After this model was trained, the testing data could be used to check the performance of the neural network model. Figures 5.21 and 5.22 show some of the results of the model at predicting these thrust-time curves.

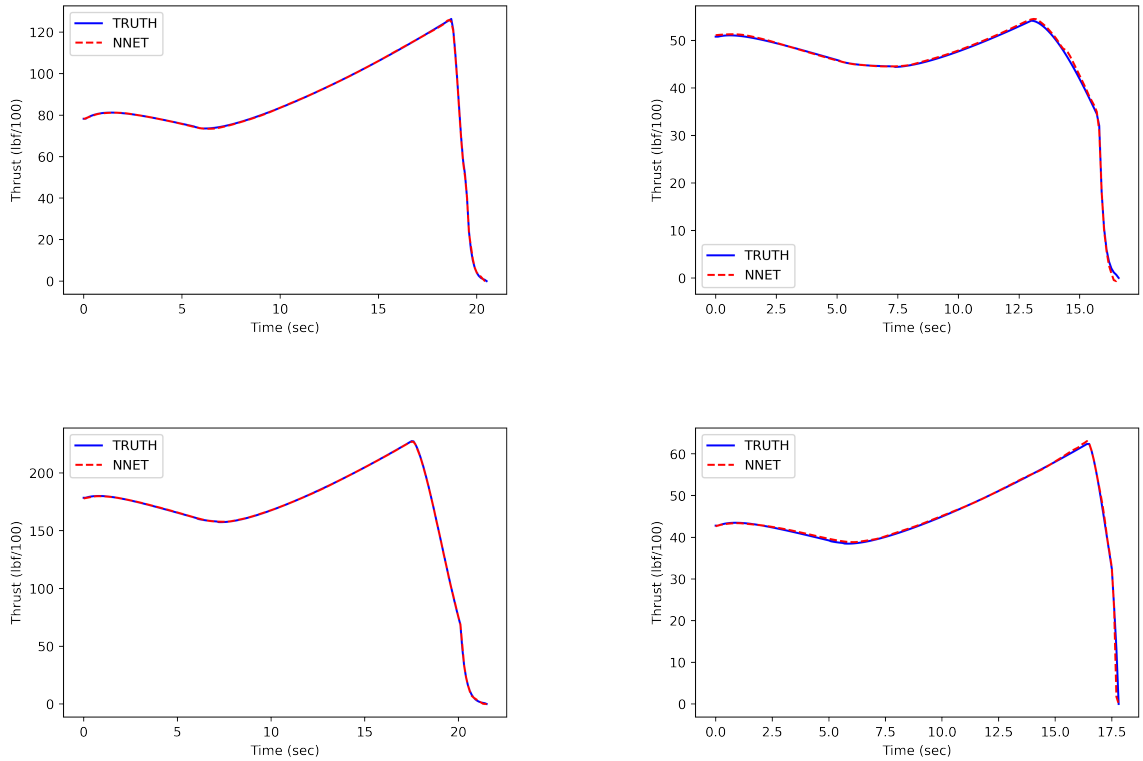



Figure 5.21: Star Grain SRM Modeling Results

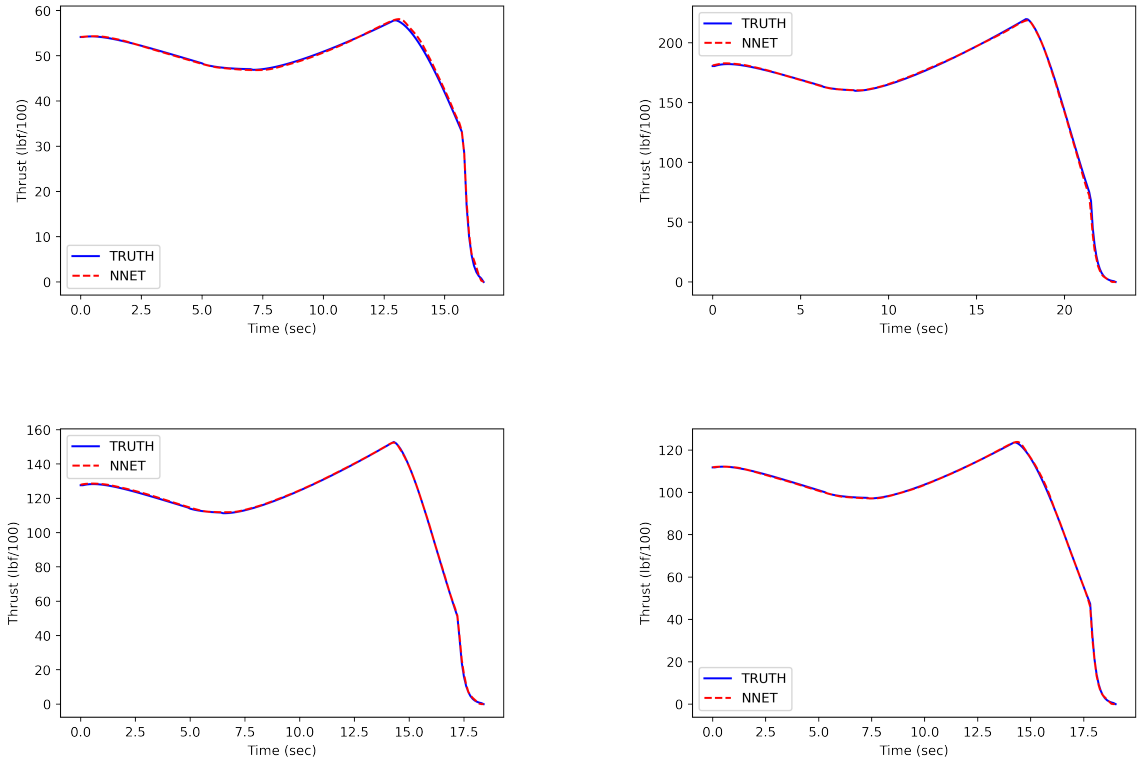



Figure 5.22: Star Grain SRM Modeling Results

Visually inspecting the results shown above in Figures 5.21 and 5.22 we seem to match the thrust-time profiles for this star grain data set quite well. Just showing a sample of results does not suffice to say a good model has been created. To further investigate the performance of the model, the residual will be plotted for the thrust. Figure 5.23 shows the residual plot for the star grain data.

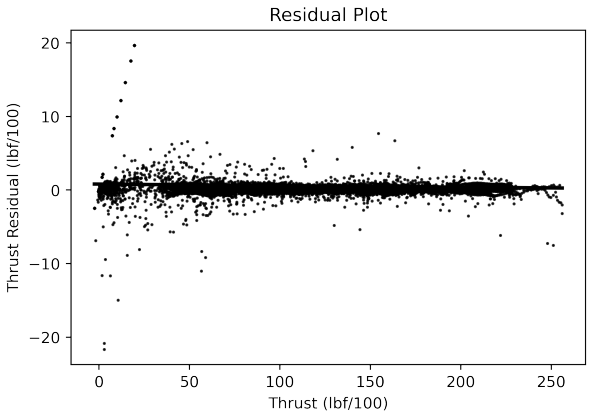



Figure 5.23: Star Grain Residual Plot

Looking at Figure 5.23 it is seen that most of the residuals lie near zero, or are centered around zero. There are a few outliers, but these points only represent single points along the thrust-time curve. This shows us that very few points are off, and if so they should not have a large effect on the overall thrust-time curve prediction. Slight temporal shifts in the predicted vs. actual thrust-time profile can lead to these larger residuals. Figure 5.24 show some of the star grain predictions that could have lead to these larger residuals shown in Figure 5.23. Inspecting the ends of the thrust-time curves shown in Figure 5.24, the neural network prediction seems to stop early. This difference in the actual vs. predicted curves leads to larger residuals shown in Figure 5.23. Regardless, these curves with larger residuals still predict the majority of the star grain thrust-time profile.

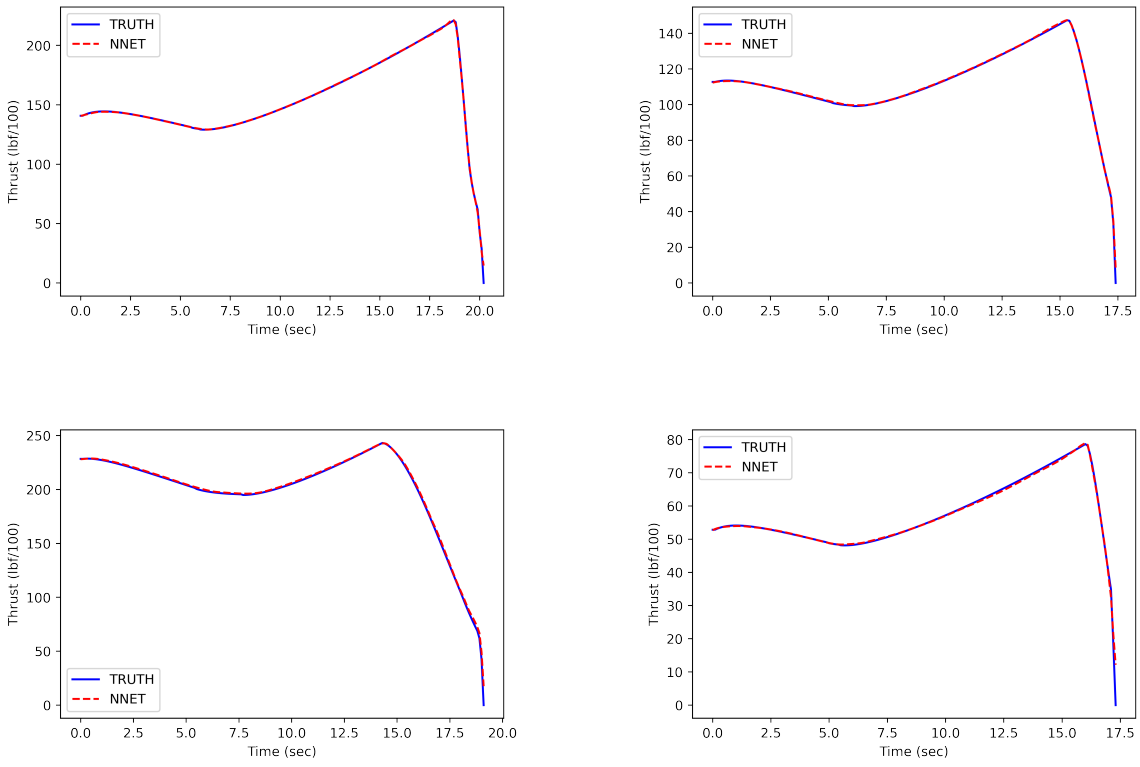



Figure 5.24: Star Grain SRM Modeling Results with larger residuals

After the development of the star grain neural network model, the Keras Tuner [32] was again used to find optimal hyper parameters for the star grain data set. Using the random search algorithm within Keras Tuner, 30 unique network deigns were trained until an optimal design was found. The objective function used was the validation loss. Each of these neural networks

was trained for 200 epochs to figure out the optimal combination of hyper-parameters. After the optimal network design was found, that network was trained for longer to develop the thrust-time curve model. Table 5.8 shows the resultant neural network design for the star grain model.

Table 5.8: Tuned Star Grain Surrogate Model Architecture

| Design Parameter | Value |
|---|---|
| Hidden Layers | 3 |
| Units Per Layer 1 | 35 |
| Units Per Layer 2 | 35 |
| Units Per Layer 3 | 25 |
| Epochs Trained | 1000 |

Table 5.8 shows the results of the Keras Tuner for the star grain data set. The model has three layers, with 35, 35, and 25 units per layer. The first two layers use the ELU activation function, while the third layer used the RELU activation function. This model was trained my minimizing the mean squared error (MSE). The results of this network are shown below in Figures 5.25 and 5.26.

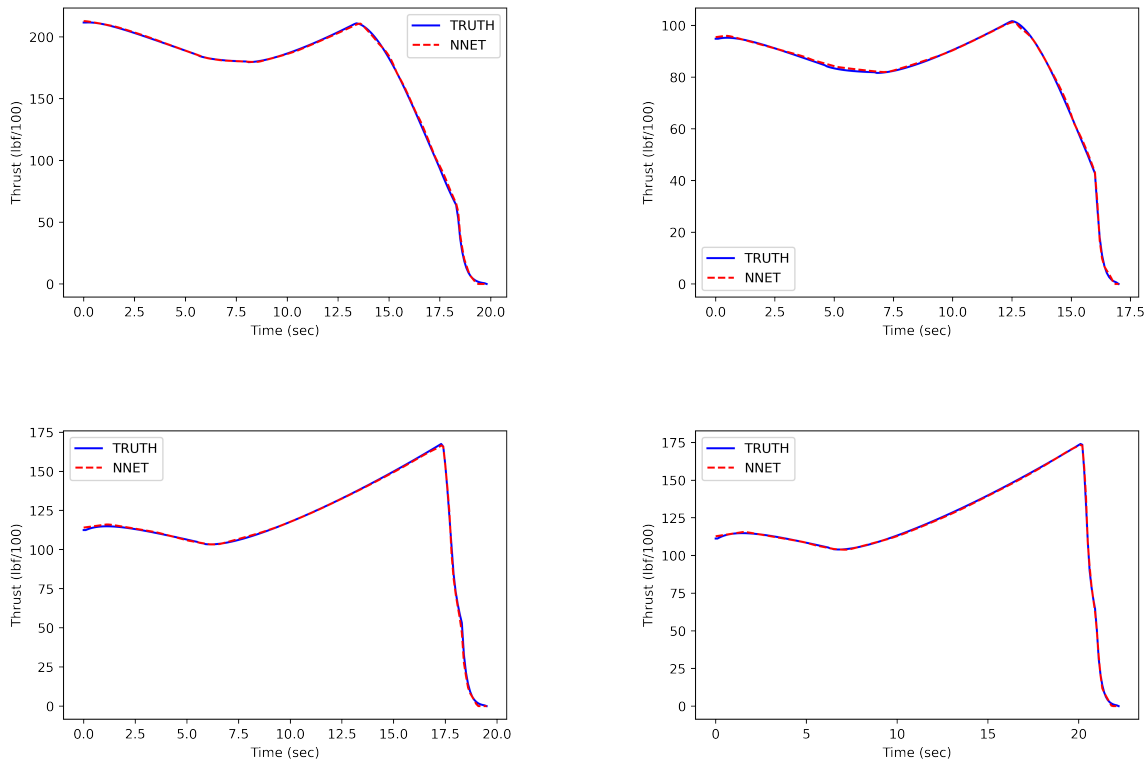



Figure 5.25: Star Grain SRM Modeling Results - Keras Tuner

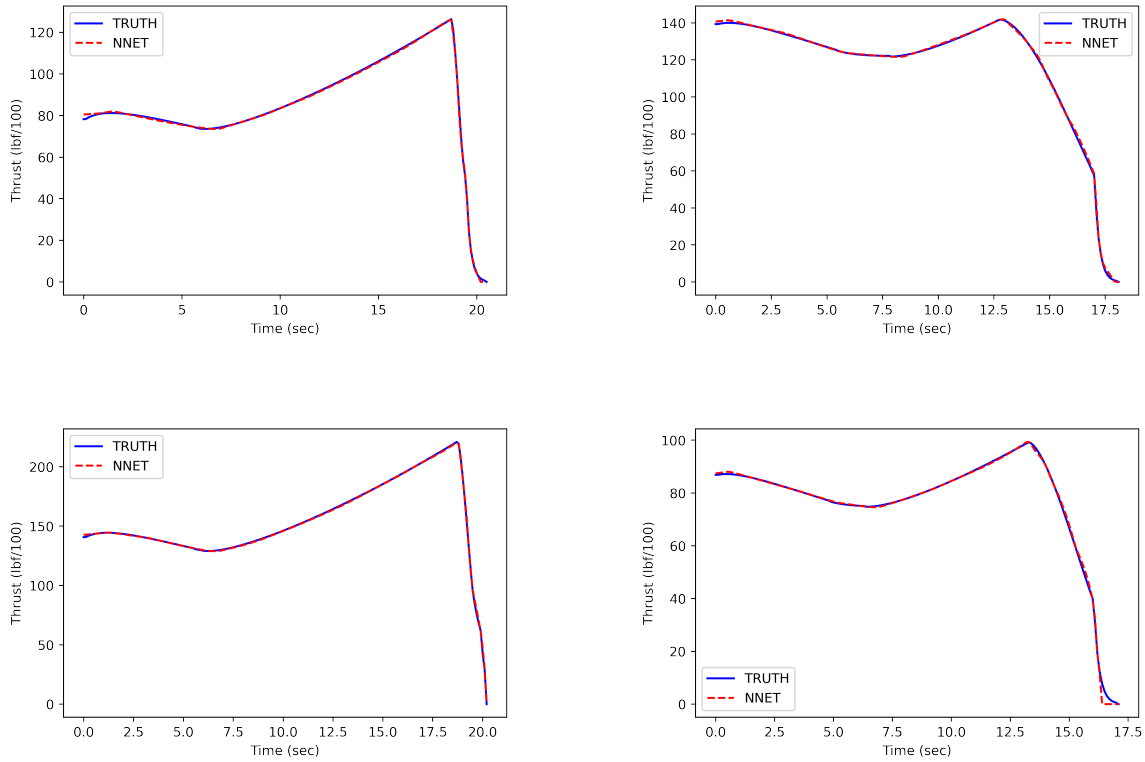



Figure 5.26: Star Grain SRM Modeling Results - Keras Tuner

Like seen before, these results match the truth quite well. The Keras Tuner neural network provided a simpler network design that was able to still accurately predict the thrust-time curves for this star grain design. Figure 5.27 shows the residual plot of thrust to help understand the performance of the model.

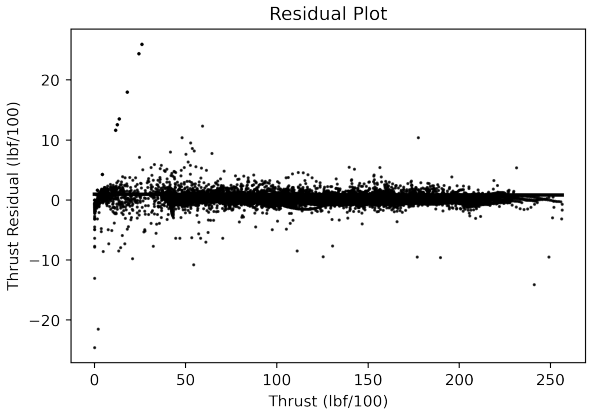



Figure 5.27: Star Grain Residual Plot - Keras Tuner

Looking at Figure 5.27 we can see the residuals for thrust for the Keras Tuner. Again, the residuals lie along zero for the thrust. This figure shows that the neural network does well

predicting the thrust-time curves. The residuals shown in Figure 5.27 seem to be similar to what was shown in Figure 5.23. Figure 5.28 shows some examples of the thrust-time curves with larger residuals.

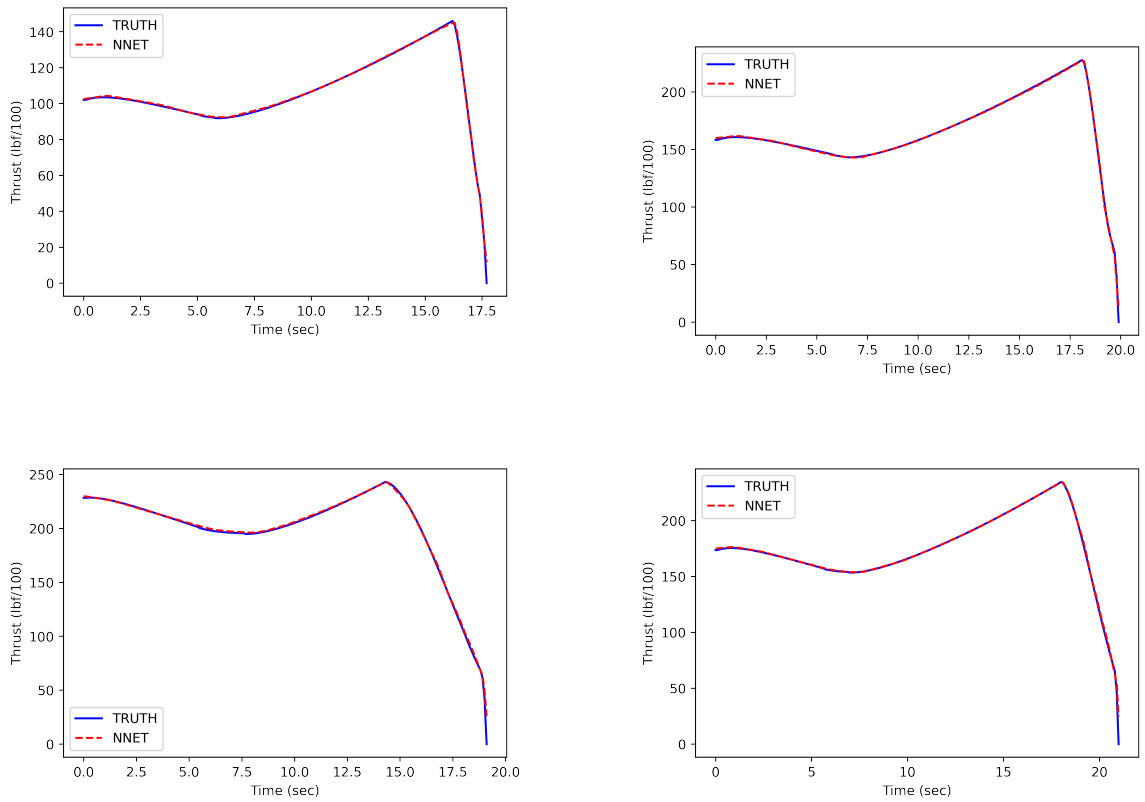



Figure 5.28: Star Grain SRM Modeling Results - Keras Tuner with larger residuals

Figure 5.28 shows some of the thrust-time results that contributed to the larger residuals in thrust shown in Figure 5.27. Inspecting the end of these thrust-time curves, a difference in the truth and neural network prediction can be seen. The neural network solution seems to stop predicting early. This difference could lead to larger residuals when calculated. Regardless, these predictions still capture the physics of the thrust-time curves, even with these differences the model still performs well.

# Chapter 6

## Conclusion and Recommendations

The analytical methods developed for this work provide a new capability in modeling tapered grain solid rocket motors. New analytical methods and burn phases have been developed as the SRM modeling and simulation part of this thesis. The new analytical methods and burn phases allow for the modeling of tapered grain solid rocket motors that was not capable previously. These methods have been developed for CP and star grain designs. The analytical methods developed as part of this work were integrated into a 1D internal ballistics code that uses 1D flow assumptions and uniform burn rate assumptions to produce conceptual design level thrust-time curves. With the new modeling and simulation capabilities, the internal ballistics code can now model both straight and tapered CP and star solid rocket motors.

The results of this thesis show how machine learning can be used to analyze tapered grain solid rocket motors from a performance standpoint. Using machine learning techniques, the performance metrics of solid rocket motors can be successfully modeled and understood. Regression models were created for both the CP grain and star grain data sets. There regression models were able to accurately predict the maximum thrust, average thrust, total impulse, and burn time for these SRM deigns. After the regression analysis, neural networks were trained to act as surrogate models for the full thrust-time profile. The analysis used here could be applied to more accurate solid rocket motor data sets. If raw test data was available for large numbers of solid rocket motor designs, it is possible the machine learning applications used in this thesis could be used on the data. Due to the requirement of a larger data set, simulation results are typically more useful.

Future work in the analytical methods could include the implementation of the tapering methodology to more advanced grain designs such as short spoke and long spoke wagon wheels. The similar methodology could be applied to the wagon wheel geometries proposed by Hartfield et al. [16, 17] to be able to model tapered wagon wheel geometries. Further work could be done to ensure the efficiency and the accuracy of the 1D internal ballistics code that was generated for this work. This new tapered geometry could be used to generate rocket classes that could then be used to create a classification problem [68, 69, 70] for new solid rocket motor geometries. Machine learning techniques, similar to what is shown in this thesis could be used to solve this classification problem. More research into advanced machine learning algorithms and methods of interpretation could be more beneficial for similar research problems.

References


[1] M. A. Edgar, F. W. Jordan, and L. W. Stockham. “A Robust Ballistic Design Approach for the Space Shuttle Advanced Solid Rocket Motor”. In: (1993).

[2] S. J. Backland and J. N. Rossen. “A Study of Performance and Cost Improvement Potential of the 120-IN.-(3.05 M) Diameter Solid Rocket Motor”. In: (1971).

[3] Jonathan McDowel and Jonathan McDowel. “Kick in the apogee-40 years of upper stage applications for solid rocket motors, 1957-1997”. In: *33rd Joint Propulsion Conference and Exhibit*. 1997, p. 3133.

[4] Edward Price. “History of solid rocket motors (1940-1960)”. In: *34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*. 1998, p. 3978.

[5] A McDonald. “Solid rockets-An affordable solution to future space propulsion needs”. In: *20th Joint Propulsion Conference*. 1984, p. 1188.

[6] Max Calabro and Jean Perret. “History of solid propellant rocket motors at Aerospatiale”. In: *28th Joint Propulsion Conference and Exhibit*. 1992, p. 3615.

[7] Michael Kaiserman et al. “An overview of the hypervelocity anti-tank missile (HATM) development program”. In: *41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*. 2005, p. 4171.

[8] Thomas Moore and Hugh McSpadden. “From Bombs to Rockets at McGregor, Texas”. In: *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*. 2009, p. 1163.

[9] Lorenzo Federici et al. “Integrated optimization of first-stage SRM and ascent trajectory of multistage launch vehicles”. In: *Journal of Spacecraft and Rockets* 58.3 (2021), pp. 786–797.

[10] John D. Dyer et al. “Aerospace Design Optimization Using a Steady State Real-Coded Genetic Algorithm”. In: (2008).

[11] M. Anderson, J. Burkhalter, and R. Jenkins. “Design of a Guided Missile Interceptor Using a Genetic Algorithm”. In: *Journal of Spacecraft and Rockets* 38.1 (2001), pp. 28–35.

[12] Kevin M. Albarado et al. “Solid Rocket Motor Performance Matching Using Pattern Search/Particle Swarm Optimization”. In: (2011).

[13] Roy J. Hartfield, Rhonald M. Jenkins, and Kevin M. Albarado. “Evolving Swarm–A Genetically Modified Particle Swarm Optimizer with Localized Pattern Search Capability for Aerospace Propulsion Systems”. In: (2012).

[14] Z. J. Kiyak and Roy J. Hartfield. “Solid Rocket Motor Design Using a Modified Ant Colony Optimization Metaheuristic with Local Search Capability”. In: (2013).

[15] K. Albarado. “Application of the Level Set Method to Solid Rocket Motor Simulation”. Auburn, AL: Auburn University, 2012.

[16] R. Hartfield et al. “A Review of Analytical Methods for Solid Rocket Motor Grain Analysis”. In: (2003). DOI: 10.2514/6.2003-4506.

[17] Roy J. Hartfield et al. “Analytical Methods for Predicting Grain Regression in Tactical Solid-Rocket Motors”. In: *Journal of Spacecraft and Rockets* 41.4 (2004), pp. 689–693.

[18] T. Sheils et al. “Statistical Analysis of Tapered Grain Solid Rocket Motor Performance”. In: *AIAA SciTech Forum*. Washington, DC: AIAA, 2023. DOI: 10.2514/6.2023-1315.

[19] Oliver C Sams IV, Joseph Majdalani, and Tony Saad. “Mean flow approximations for solid rocket motors with tapered walls”. In: *Journal of Propulsion and Power* 23.2 (2007), pp. 445–456.

[20] Oliver Sams, Joseph Majdalani, and Gary Flandro. “Higher flowfield approximations for solid rocket motors with tapered bores”. In: *40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*. 2004, p. 4051.

[21] Tony Saad, Oliver C Sams IV, and Joseph Majdalani. “Rotational flow in tapered slab rocket motors”. In: *Physics of Fluids* 18.10 (2006), p. 103601.

[22] M. Barrere et al. “Rocket Propulsion”. In: Amsterdam: Elsevier Publishing Company, 1960.

[23] Andrea Ricciardi. “Generalized geometric analysis of right circular cylindrical star perforated and tapered grains”. In: *Journal of Propulsion and Power* 8.1 (1992), pp. 51–58.

[24] M. Anderson, J. Burkhalter, and R. Jenkins. “Missile Shape Optimization Using Genetic Algorithms”. In: *Journal of Spacecraft and Rockets* 3.5 (2000), pp. 663–669.

[25] M. Anderson, J. Burkhalter, and R. Jenkins. “Intelligent Systems Approach to Designing an Interceptor to Defeat Highly Manueverable Targets”. In: (2001).

[26] Jonathan Metts et al. “Reverse Engineering of Solid Rocket Missiles with a Genetic Algorithm”. In: *45th AIAA Aerospace Sciences Meeting and Exhibit*. 2007, p. 363.

[27] N. Cervantes. “Engineering and Statistical Analysis of Solid and Liquid Missile Systems”. Auburn, AL, 2022.

[28] Robert L Harrison. “Introduction to Monte Carlo simulation”. In: *AIP conference proceedings*. Vol. 1204. 1. American Institute of Physics. 2010, pp. 17–21.

[29] Samik Raychaudhuri. “Introduction to monte carlo simulation”. In: *2008 Winter simulation conference*. IEEE. 2008, pp. 91–100.

[30] GA Bird. “Monte-Carlo simulation in an engineering context”. In: *Progress in Astronautics and Aeronautics* 74 (1981), pp. 239–255.

[31] Scott M Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 4765–4774. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[32] Tom O’Malley et al. *Keras Tuner*. https://github.com/keras-team/keras-tuner. 2019.

[33] Roy Hartfield. *Rocket Propulsion Class Notes*. Auburn University, 2023.

[34] George P Sutton and Oscar Biblarz. *Rocket propulsion Elements*. John Wiley & Sons, 2016.

[35] Griffin A DiMaggio et al. “Solid rocket motor internal ballistics using an enhanced surface-vorticity panel technique”. In: *Physics of Fluids* 33.10 (2021), p. 103613.

[36] Griffin A DiMaggio et al. “Solid Rocket Motor Internal Ballistics with a Surface-Vorticity Solver”. In: *AIAA SCITECH 2022 Forum*. 2022, p. 1898.

[37] *Bisection Method - Fortran*. 2021. URL: https://www.mycompiler.io/view/9NadoT1 (visited on 04/20/2023).

[38] Michael Stein. “Large Sample Properties of Simulations Using Latin Hypercube Sampling”. In: *Technometrics* 29.2 (1987), pp. 143–151. DOI: 10.1080/00401706.1987.10488205. eprint: https://www.tandfonline.com/doi/pdf/10.1080/00401706.1987.10488205. URL: https://www.tandfonline.com/doi/abs/10.1080/00401706.1987.10488205.

[39] Roy J. Hartfield, D. M. Carpenter, and Noel Cervantes. “Statistical Learning for Solid Propellant Performance”. In: (2021).

[40] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[41] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[42] John T Hwang and Joaquim RRA Martins. "A fast-prediction surrogate model for large datasets". In: *Aerospace Science and Technology* 75 (2018), pp. 74–87.

[43] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[44] Gareth James et al. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. URL: https://faculty.marshall.usc.edu/gareth-james/ISL/.

[45] Michael L. Waskom. "seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.

[46] Tanish Jain, Shlok Misra, and Dothang Truong. "Using Deep Learning to Predict Unstable Approaches for General Aviation Aircraft". In: *Journal of Aerospace Information Systems* 19.12 (2022), pp. 811–817.

[47] Daniel Wesely et al. "A Machine Learning Approach to Predict Aircraft Landing Times using Mediated Predictions from Existing Systems". In: *AIAA AVIATION 2021 FORUM*. 2021, p. 2402.

[48] Frederick Wieland et al. "Predicting Sector Complexity Using Machine Learning". In: *AIAA AVIATION 2022 Forum*. 2022, p. 3754.

[49] Amir Bahador Parsa et al. "Toward safer highways, application of XGBoost and SHAP for real-time accident detection and feature analysis". In: *Accident Analysis & Prevention* 136 (2020), p. 105405.

[50] De-Cheng Feng et al. “Interpretable XGBoost-SHAP machine-learning model for shear strength prediction of squat RC walls”. In: *Journal of Structural Engineering* 147.11 (2021), p. 04021173.

[51] Sujith Mangalathu, Seong-Hoon Hwang, and Jong-Su Jeon. “Failure mode and effects analysis of RC members based on machine-learning-based SHapley Additive exPlanations (SHAP) approach”. In: *Engineering Structures* 219 (2020), p. 110927.

[52] Guy Van den Broeck et al. “On the tractability of SHAP explanations”. In: *Journal of Artificial Intelligence Research* 74 (2022), pp. 851–886.

[53] Sam J Silva, Christoph A Keller, and Joseph Hardin. “Using an Explainable Machine Learning Approach to Characterize Earth System Model Errors: Application of SHAP Analysis to Modeling Lightning Flash Occurrence”. In: *Journal of Advances in Modeling Earth Systems* 14.4 (2022), e2021MS002881.

[54] Chejarla Santosh Kumar et al. “Dimensionality Reduction based on SHAP Analysis: A Simple and Trustworthy Approach”. In: *2020 International Conference on Communication and Signal Processing (ICCSP)*. 2020, pp. 558–560. DOI: 10.1109/ICCSP48568.2020.9182109.

[55] Wilson E. Marcílio and Danilo M. Eler. “From explanations to feature selection: assessing SHAP values as feature selection mechanism”. In: *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. 2020, pp. 340–347. DOI: 10.1109/SIBGRAPI51738.2020.00053.

[56] Ryota Kitani and Shinya Iwata. “Verification of Interpretability of Phase-Resolved Partial Discharge Using a CNN With SHAP”. In: *IEEE Access* 11 (2023), pp. 4752–4762. DOI: 10.1109/ACCESS.2023.3236315.

[57] Faizeh Hatami et al. “Non-Linear Associations Between the Urban Built Environment and Commuting Modal Split: A Random Forest Approach and SHAP Evaluation”. In: *IEEE Access* 11 (2023), pp. 12649–12662. DOI: 10.1109/ACCESS.2023.3241627.

[58] Ripan Kumar Kundu et al. *LiteVR: Interpretable and Lightweight Cybersickness Detection using Explainable AI*. 2023. DOI: 10.48550/ARXIV.2302.03037. URL: https://arxiv.org/abs/2302.03037.

[59] Jennifer Abras et al. "Machine Learning-Based Surrogate Modeling for Aerodynamic Loads Predictions". In: *AIAA SCITECH 2023 Forum*. 2023, p. 0232.

[60] Jincheng Zhang and Xiaowei Zhao. "Machine-learning-based surrogate modeling of aerodynamic flow around distributed structures". In: *AIAA Journal* 59.3 (2021), pp. 868–879.

[61] Hasan Karali et al. "Design of a deep learning based nonlinear aerodynamic surrogate model for UAVs". In: *AIAA Scitech 2020 forum*. 2020, p. 1288.

[62] Hamidreza Karbasian and Wim M van Rees. "A Deep-Learning Surrogate Model Approach for Optimization of Morphing Airfoils". In: *AIAA SCITECH 2023 Forum*. 2023, p. 1619.

[63] Siddharth Jain, Rakesh K Kapania, and Daniel Hammerand. "Development of Surrogate Model to Predict Errors in FEM solutions using Deep Convolutional Neural Networks". In: *AIAA SCITECH 2022 Forum*. 2022, p. 0972.

[64] Cooper Cone et al. "Reward Function Optimization of a Deep Reinforcement Learning Collision Avoidance System". In: *arXiv preprint arXiv:2212.00855* (2022).

[65] Moritz Krügener et al. "Coaxial-Injector Surrogate Modeling Based on Reynolds-Averaged Navier–Stokes Simulations Using Deep Learning". In: *Journal of Propulsion and Power* 38.5 (2022), pp. 783–798.

[66] Roy J Hartfield and Mark Carpenter. "Statistical Learning for Solid Propellant Performance". In: *AIAA Propulsion and Energy 2021 Forum*. 2021, p. 3704.

[67] N.D. Bennett et al. "Characterising performance of environmental models". In: *Environ. Model. Softw. 40(February 2013)* (Nov. 2012), pp. 1–20.

[68] Jordan Eckert et al. "Classification of Intermediate Range Missiles During Launch". In: *AIAA Scitech 2020 Forum*. 2020, p. 1852.

[69] Mark Carpenter, Norman Speakman, and Roy J Hartfield. “Rapid Characterization of Munitions Using Neural Networks”. In: *AIAA Atmospheric Flight Mechanics Conference*. 2016, p. 0787.

[70] J. Eckert. “On the Classification of Intermediate Range Missiles During Launch”. Auburn University, 2020.