HEATING AND REGENERATIVE COOLING MODEL FOR A ROTATING DETONATION ENGINE DESIGNED FOR UPPER STAGE PERFORMANCE

by

Timothy Philip Gurshin Jr.

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Aeronautics and Astronautics

School of Aeronautics & Astronautics

West Lafayette, Indiana

August 2019

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Stephen D. Heister, Chair
Department of Aeronautics and Astronautics
Dr. Timothée L. Pourpoint
Department of Aeronautics and Astronautics
Dr. Gregory A. Blaisdell
Department of Aeronautics and Astronautics

Approved by:

Dr. Weinong W. Chen
Head of the Graduate Program

For my parents, Olivia Henry and Timothy Gurshin,
who steadfastly encourage me to reach for the stars

ACKNOWLEDGMENTS

I would like to extend my thanks to Dr. Stephen Heister for giving me the opportunity to contribute to his group's research into rotating detonation engines. Thank you for your continual support and creative ideas for this project. I would also like to thank Dr. Timothée Pourpoint and Dr. Gregory Blaisdell for their constructive review of and feedback on this thesis.

I am grateful for my research group members, Dasheng Lim, Kota Mikoshiba, Alexis Harroun, Jenna Humble, Hasan Çelebi, Stephen Kubicki, and Kevin Dille – for their collaboration and help debugging, and for doing their own inspiring work.

I am thankful also for BJ Austin and Andrew Albers's support at IN Space LLC as I delved into learning about rotating detonation engines.

Finally, thank you so much, family and friends, both near and far, for your help editing this thesis and for encouraging me throughout my graduate studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Author: Gurshin Jr., Timothy, P. MSAAE
Institution: Purdue University
Degree Received: August 2019
Title: Heating and Regenerative Cooling Model for a Rotating Detonation Engine Designed for Upper Stage Performance
Committee Chair: Dr. Stephen Heister

Rotating detonation engines (RDE) have the potential to significantly advance the efficiency of chemical propulsion. They are approximately one order of magnitude shorter than constant pressure engines, a savings benefit that is especially important for upper stage engines. There are many challenges to advancing their technological readiness level, but one area this thesis attempts to help mitigate is the understanding of heat loads and the viability of regenerative jacket cooling.

A one-dimensional, steady-state heat transfer and regenerative cooling model for the upper stage RL10A-3-3A (RL10) engine is developed in MATLAB. This model considers forced convection in the boundary layer between the combustion product gases and the hot-gas-side wall, conduction through the wall, and forced convection in the boundary layer between the hydrogen coolant and coolant-side wall. Variable gas and coolant transport properties are utilized to increase physical accuracy. The model also quantifies pressure drop through the cooling channels due to wall friction. This allows for overall heat flux, and consequently hot-gas-side and coolant-side wall temperatures to be predicted along the length of the engine. Properties of the coolant can also be predicted including the jacket outlet temperature and pressure. These cooling circuit final parameters, temperature rise and pressure drop, were matched to a more detailed, three-dimensional, transient RL10 system model developed by NASA, thereby anchoring this model.

An RDE is designed to notionally meet the thrust level of RL10. Model design decisions are documented and explained, and a detailed comparison of the two engine geometries is made. The regenerative cooling model is adapted for the RDE considering such unique aspects as detonative heat flux and the centerbody/plug nozzle. Steady state heating and cooling analysis is performed on the RDE and the results are compared to RL10. Investigation into the effects of the RDE's differing cooling jacket output conditions on the turbine are calculated and discussed.

Appendix analyses consider more realistic detonative heat flux approximations according to recent RDE calorimetry studies and the effect of altering detonation chamber heat flux.

Even with the conservative assumption of throat-level heat flux everywhere in the RDE's annular combustion chamber, regenerative jacket cooling shows promise as a means of thermal survival. Wall temperatures are reasonable, coolant temperature rise is lower, and coolant pressure drop is lower. The reduced temperature rise presents the new challenge of correctly powering the turbine since the incoming coolant is less energized. Further studies should also look at channel optimization specific to the RDE to maximize cooling performance and ease of system integration.

1. INTRODUCTION

Chemical propulsion remains the sole method for accessing space. Since Sputnik's successful orbital launch in 1957, chemical rocket engines have been continually refined, but efficiency gains have become asymptotic to the point where improvements of a couple of percent are significant. Recent research into rotating detonation engines (RDE) designed for upper stage applications has estimated increases in specific impulse I_{sp} (a common parameter used to quantify rocket engine efficiency) as high as 12.2% [1].

1.1 RDE Overview

RDEs are a type of combustor that use the detonation mode of combustion instead of the traditionally used deflagration. Deflagrations are commonly known as flames and they are ubiquitous, from a match, to a gas burner stove, to the flame in a rocket engine. Detonation is a distinct phenomenon where combustion is coupled with a shock wave. Real detonations are complicated three-dimensional structures, but the ZND (named after Zeldovich, von Neumann, and Döring, three independent scientists) model provides a useful one-dimensional representation [2] and is depicted in Figure 1.1.

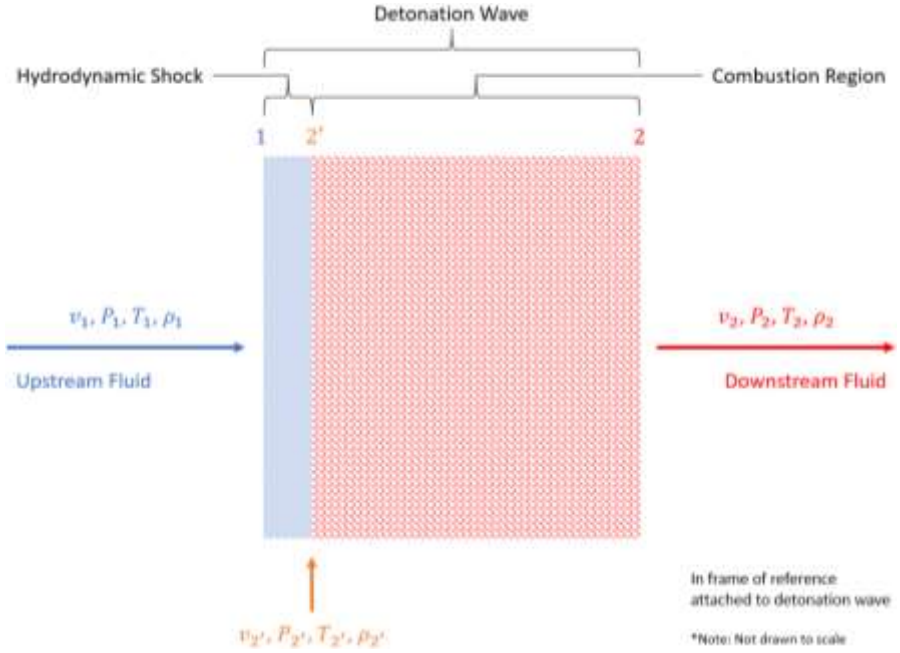


Figure 1.1 ZND Model of a Detonation Wave

When a pressure wave travels faster than the local speed of sound it becomes a shock wave, a region spanning only a few molecular mean free paths (on the order of a few hundreds of nanometers $\{O(100\ [nm])\}$ in air at Earth's surface [3] over which pressure P, temperature T, and density ρ increase sharply. A detonation wave features this sudden change with an immediately trailing combustion region that is significantly thicker. Combustion is an exothermic reaction where the rearrangement of chemical bonds leads to net heat release. This added energy further increases the shocked fluid's temperature and pressure, but it decreases its density. Note that the pressure ratio in Figure 1.2 had to be divided by two for the sake of the plot.

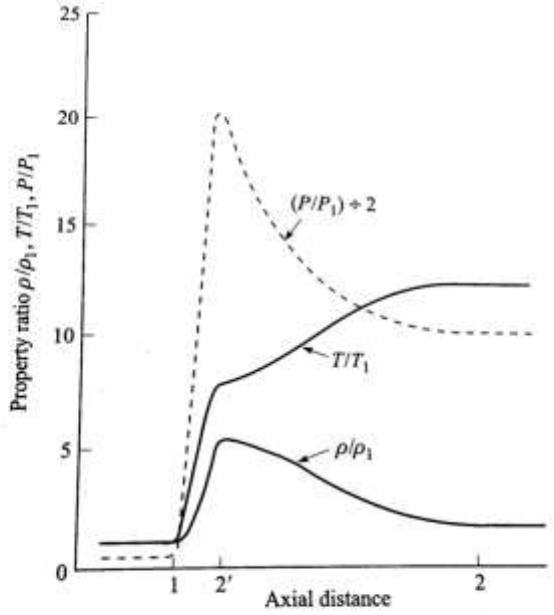


Figure 16.4 Schematic representation of ZND detonation structure. State points 1, 2', and 2 denote upstream conditions, conditions at the end of the leading shock, and the Chapman–Jouguet point, respectively.

Figure 1.2 Fluid Property Variations Across ZND Detonation from Turns [2]

Another way to appreciate detonations' energetic nature is through kinetic energy via velocity $\left(KE = \frac{1}{2}mv^2\right)$. In combustion science, the concept of flame speed and detonation speed are helpful in visualizing these discrete phenomena. For a deflagration, imagine being in a frame of reference attached to the flame. The speed at which unburned reactants enter the flame is known as the laminar flame speed and it is $O\left(10\ \left[\frac{cm}{s}\right]\right)$, about an order of magnitude slower than human

distance-running speed. Contrast this with detonation velocity which is $O\left(1\left[\frac{km}{s}\right]\right)$. Detonation waves are often traveling at hypersonic velocities.

1.2 RDE Benefits and Challenges

Detonation is a tantalizing hydrodynamic and thermochemical phenomenon that was actually first conceived by Russian scientists in the 1950s [4]. Difficulty in modeling, controlling, and withstanding this violent combustion has delayed its development to flight readiness. Despite the challenges, detonation engines have the ability to combust very fast reactants (conditions that often blow out deflagrations) and are theoretically more efficient, so research has persisted. Thermodynamically, they can be modeled as detonation cycles and compared against the Brayton cycle, the classic representation of gas-turbine engines. On a pressure and specific volume (P-v) diagram, extractable work is represented by the integral of the cycle curve as depicted in Figure 1.3.

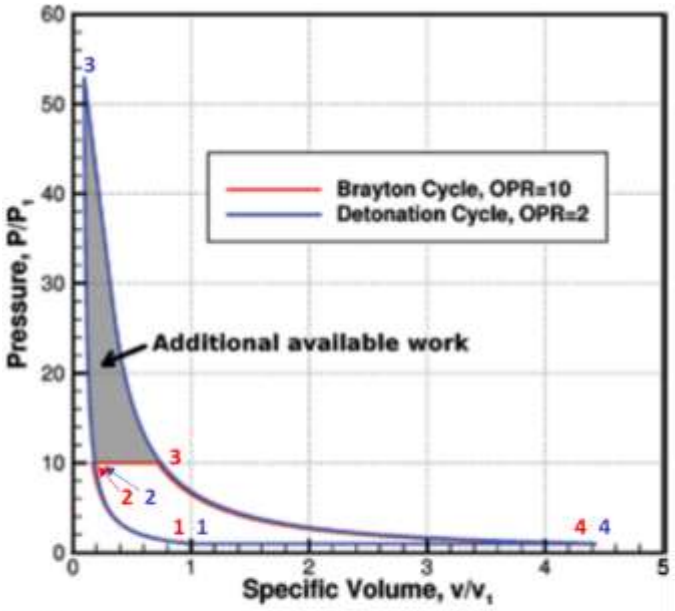


FIGURE 1
Comparison of Brayton and detonation cycles on a *P-v* diagram, with an operating pressure ratio (OPR) of 2 for the detonation cycle, and 10 for the Brayton cycle.

Figure 1.3 Detonation vs. Brayton Cycle P-v Diagram from Schwer [5]

The Brayton cycle models combustion (station 2 to 3) as constant pressure which is approximately true for deflagration. Detonation induces a massive jump in pressure during its combustion phase (also station 2-3). This is why detonation is considered a form of pressure gain combustion.

Different concepts for harnessing detonations have been conceived and tested over the decades, and one of the most promising is the rotating detonation engine. The rotation in the title refers to the circumferential travel of the detonation wave inside a cylindrical annular chamber. Reactants are generally injected along the engine's centerline axis and are intercepted perpendicularly by the traveling detonation wave. A detailed three-dimensional, non-premixed computational fluid dynamics simulation was performed by the Argonne National Laboratory and illustrates an RDE in action. In Figure 1.4 above, the detonation wave front is the red region and

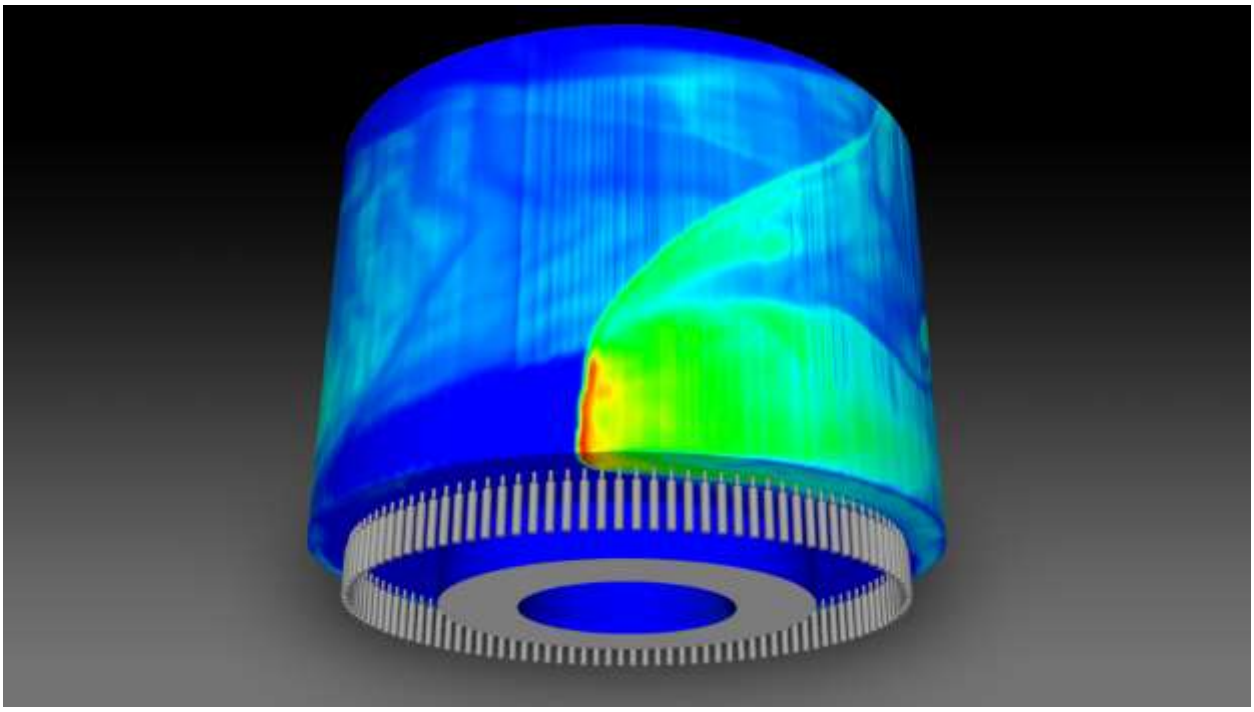


Figure 1.4 CFD Snapshot of Operating RDE from Argonne National Laboratory [6]

it is traveling to the left. The dark blue to the left represents unburned reactants; everything to the right of the wave has combusted. The trailing line extending up and to the right of the wave front is an expansion wave. Reactants are injected axially at the bottom of the RDE through the small gray cylinders; products exit through the top. As soon as the wave passes, fresh reactants can begin filling and mixing before the wave circumnavigates the chamber. The energy of fresh combustion

products propels the wave around continuously. Operation is inherently unsteady as the wave decelerates in between injectors and before being reenergized.

For rocket applications, RDEs offer excellent volume and mass savings, precious commodities in a launch budget given payload per kilogram cost, even considering the latest commercial offerings, being thousands of dollars. In an RDE, complete combustion occurs over a much shorter distance than constant pressure engines, so the whole chamber can be shortened. Additionally, the annular outflow is naturally shaped to take advantage of an aerospike nozzle. The 12.2% I_{sp} gain predicted in Stechmann's paper [1] assumed an aerospike versus the traditionally bell-shaped nozzle case which yielded "only" 9.0% improvement. This 3.2% increase was attributed to aerospikes' ability to adjust to the variations in operating pressure that an RDE produces.

Constructing a viable rocket RDE poses numerous engineering challenges. Structurally, they need to withstand extremely high peak post-detonation pressures that occur nearly cyclically as the wave travels around the chamber at kHz frequencies which thereby also induces severe fatigue. Propellants must be consistently and rapidly fed into the chamber to both mix prior to wave arrival and to sustain continuous detonation. When the wave passes, injectors are exposed to the high post-detonation pressure which creates a transient back pressure that acts against the desired direction of flow, trying to force reactants back up into injectors or manifolds. Thermally, the wall just downstream of the wave front is blasted by superheated products which induce very high, transient heat flux. Most of the time, however, the wave is not present, and temperature diminishes rapidly in the wave's absence. Nonetheless, most hot-fire tests to date have been very short $\{O(1\ [s])\}$, and even then, if conditions are too energetic, the test chambers were damaged or destroyed. Clearly, thermal management will be critical to bringing this technology out of the laboratory.

This study seeks to model these heat loads and evaluate the efficacy of a regenerative cooling jacket on a rotating detonation engine designed for upper stage performance. First a regenerative cooling MATLAB model for one-dimensional, steady state heat transfer along the engine's centerline was developed. Because of the simplifications, the model needed to be anchored to more comprehensive analyses and test data to establish validity. No upper stage (or any stage) RDEs have been successfully flown so constant pressure engines were instead considered. The process of compensating for the difference in combustion mode and heat flux by

incorporating analysis of recent RDE heat flux experimental data obtained by ISSI, Inc. and the Air Force Research Laboratory (AFLR) [7] is explained in 0. The natural choice for comparison to an upper stage engine was the RL10.

1.3 RL10 Overview

Pratt & Whitney developed the original RL10, the RL10A-1 in the late 1950s for NASA. Many variants have since been developed; RL10A-3-3A (see Figure 1.5), used on the Centaur upper stage of the Titan launch vehicle, and retired in 2003 [8], was the variant specifically analyzed since it has been extensively researched and flown. It uses liquid oxygen (LOX) as the

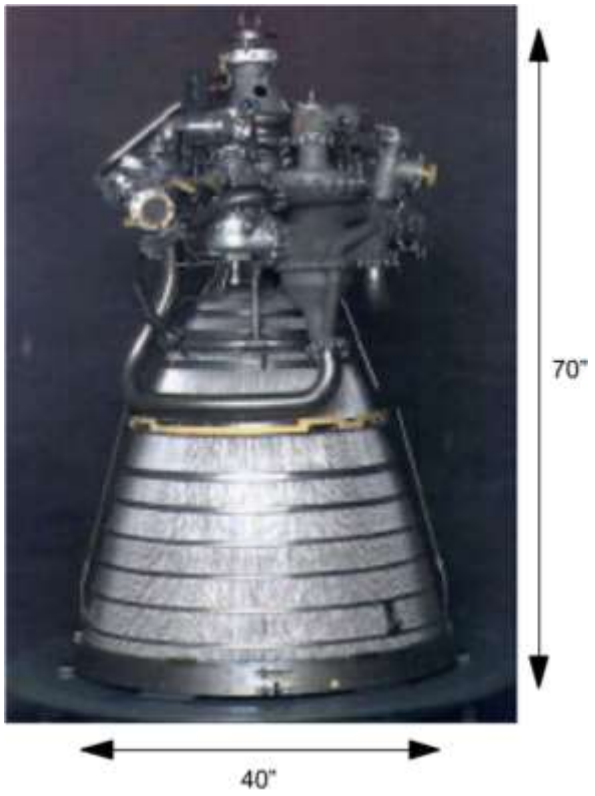


Figure 1.5 RL10-3-3A from ULA [9]

oxidizer and liquid hydrogen (LH_2) as the fuel, and operates at a nominal chamber pressure of 475 [psia], oxidizer-to-fuel mass flow rate ratio (O/F) of 5.0, and thrust of 74,000 [N] [10]. RL10A-3-3A is an expander cycle engine where the cryogenic LH_2 is first routed into a cooling manifold where it is energized by absorbing heat from the nozzle and engine before being fed into the turbine

which powers both the LH_2 pump directly and the LOX pump using a gear train. Binder [10] provided a schematic (see Figure 1.6) of RL103-A-3A's (hereafter RL10) expander cycle engine.

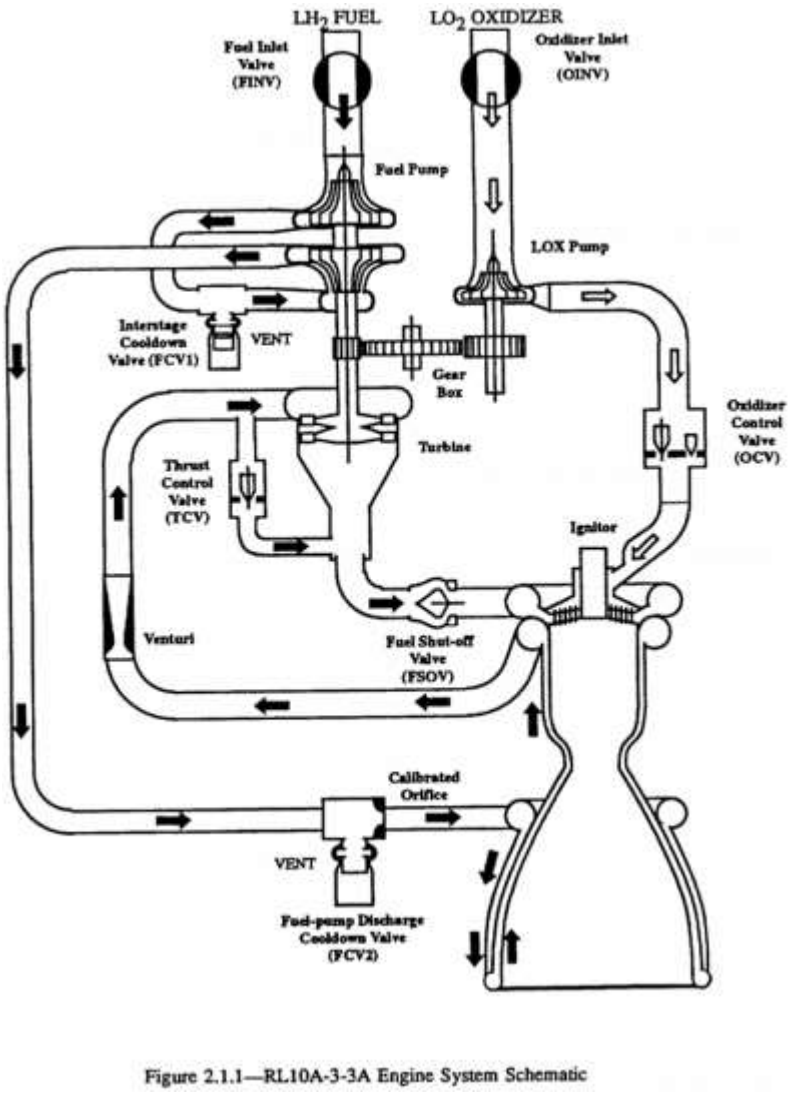


Figure 1.6 RL10A-3-3A Engine System Schematic from Binder [10]

Figure 1.6 shows fuel entering the diverging nozzle partially downstream of the throat which is the narrowest point of the thrust chamber assembly (combination of the combustion chamber and de Laval nozzle). This is qualitatively accurate in that the coolant first flows aft (toward the nozzle exit) in 180 "short," stainless steel tubes before entering a turnaround manifold and then dividing into 180 "long" tubes that run all the way from the nozzle exit to the beginning of the combustion chamber. These tubes are brazed together and form the inner wall of the thrust chamber assembly (TCA). Together they form the regenerative cooling jacket of RL10, called so

because heat that would otherwise be wasted is instead recycled to energize the pre-turbine flow. Binder's [10] illustration of the cooling jacket is shown in Figure 1.7.

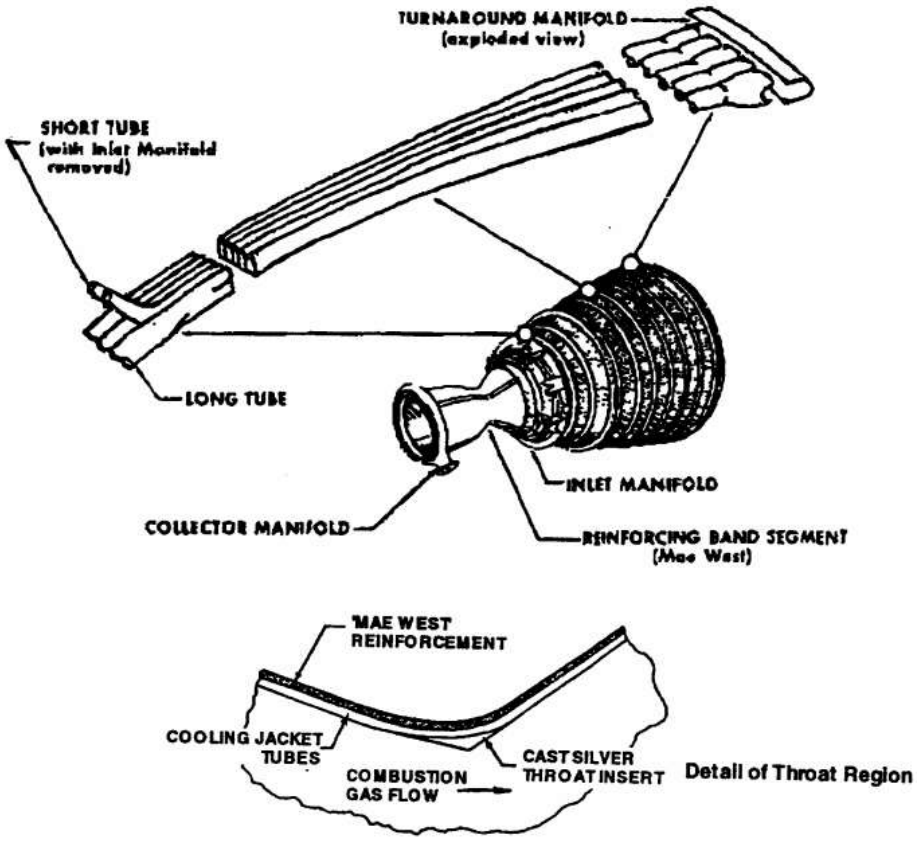


Figure 2.4.1—Structure of Regenerative Cooling Jacket, Chamber and Nozzle

Figure 1.7 RL10A-3-3A Cooling Jacket and TCA Illustration from Binder [10]

1.4 NASA Space Propulsion Technology Division RL10 Model

In 1990 NASA identified the need for an independent, publicly accessible computer model of the RL10 engine after a couple of in-flight failures. Binder of NYMA, Inc. and Tomsik and Veres of the Space Propulsion Technology Division (SPTD) at NASA Lewis Research Center [10] developed this system model over several years, incorporating new detailed component data and subsystem models. One of the main goals was to accurately predict heat transfer. In so doing they modeled three-dimensional, transient effects using the Rocket Thermal Evaluation (RTE) code available to the SPTD. In January 1997, they released their results in NASA Technical Memorandum 107318.

The Bartz correlation was used for hot-gas-side convective heat transfer coefficient h_g. Accurate gas thermochemical properties were generated from their CET93 code. They had to multiply the Bartz equation by a coefficient of 1.08 to better match RL10 test data. For coolant-side convective heat transfer coefficient h_c, they used the Colburn relation. Their model accounted for curvature within the coolant tubes. Their conduction model accounted for variable stainless-steel wall thermal conductivity. Finally, they included a coolant boiling heat transfer subroutine to try to reproduce pressure oscillations in the test data.

After optimizing the spatial resolution of their model, they were able to closely match RL10 test data relevant to regenerative cooling. The SPTD model predicts heat flux, hot-gas-side wall temperature, coolant pressure drop, and coolant pressure rise that closely matches both test data and Pratt & Whitney's proprietary RL10 model. Binder [10] provides initial coolant temperature and pressure, coolant temperature rise and pressure drop, cumulative heat rate, and more to which the one-dimensional, steady state model presented herein can be anchored.

1.5 Thesis Outline and Objectives

In Chapter 2, this thesis describes in detail the process of developing a one-dimensional, steady state regenerative cooling model that is anchored to Binder's [10] more physically nuanced and verified model of RL10.

Chapter 3 explains how an RDE inspired by RL10 and its upper stage use was designed. It also shows how the heat transfer model from chapter 2 was adapted for the RDE. Chapter 3 then provides numerous geometric and heat transfer model outputs in absolute terms but also in comparison with RL10. It also looks at potential impacts on the turbine.

Detailed justification for heat flux values in the RDE chamber is provided in 0. 0 investigates the effect of altering the RDE chamber heat flux.

Conclusions from the various studies are consolidated in Chapter 4. Recommendations for future model modifications and subsequent studies are provided.

2. MODEL DEVELOPMENT

Heating and cooling physical processes in rotating detonation engines are inherently unsteady and three-dimensional. They involve forced convection of multiple fluids, intermittently driven by post-detonation products at hypersonic velocities. In order to simplify the heat transfer analysis, a one-dimensional, steady state analysis was employed in MATLAB. The model's lower fidelity was compensated for by anchoring product gas forced convection heat transfer coefficient and coolant Darcy friction factor. Anchor points came from Binder's [10] more detailed RL10A-3-3A (hereafter "RL10") heat transfer modeling as part of the development of NASA's Space Propulsion Technology Division's Rocket Engine Transient Simulator code [10]. Once anchored the model could be extended to an RDE with comparable performance to the RL10 engine. This chapter details how the model was built and anchored.

2.1 RL10 Geometry

Figure 2.1 depicts the axial cross-section of the RL10 thrust chamber and nozzle.

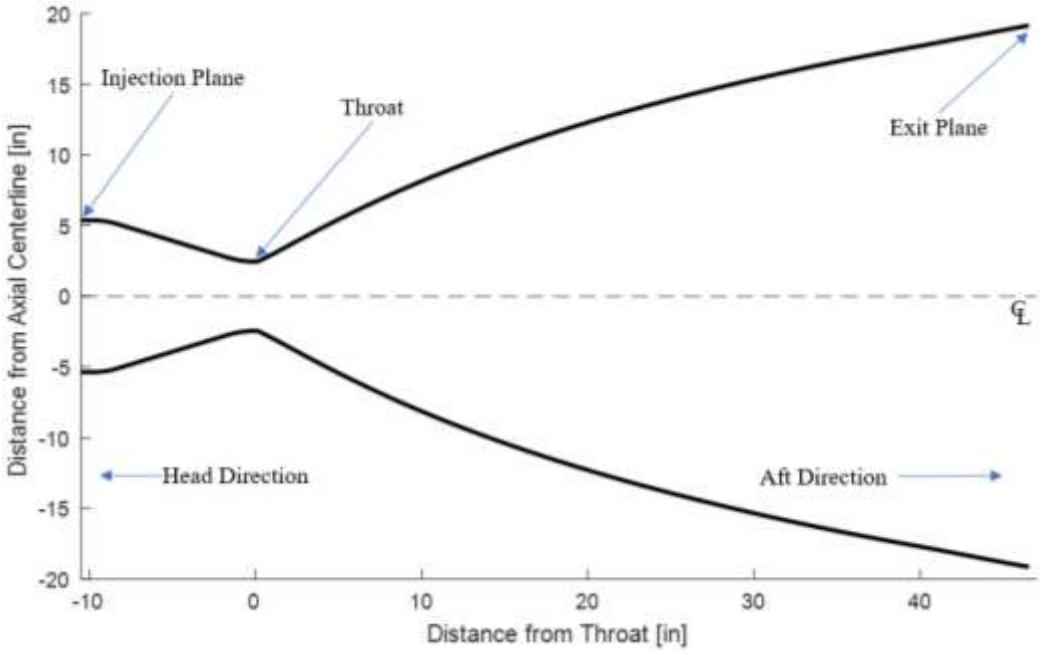


Figure 2.1 RL10 Engine Axial Cross Section

Geometry data for RL10 were obtained from **Error! Reference source not found.** which included engine outer mold line radius, and coolant tube perimeter and cross-sectional area, all corresponding to a location along the head to aft centerline axis (the "axial" axis). The data are plotted below for reference.

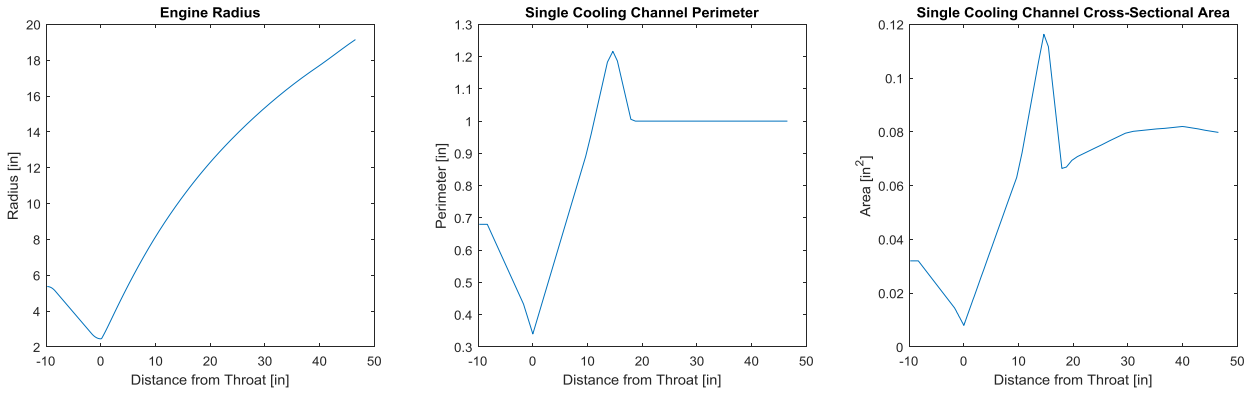


Figure 2.2 RL10 Geometric Data

These data had some duplicate values and was nonlinearly spaced, so an evenly spaced data set was linearly interpolated. It was also extended so that the chamber started at -12.143 [in] per Figure D1 in Binder [10]. Figure 2.3 shows the interpolated engine contour overlaid with the original demonstrating their strong agreement.

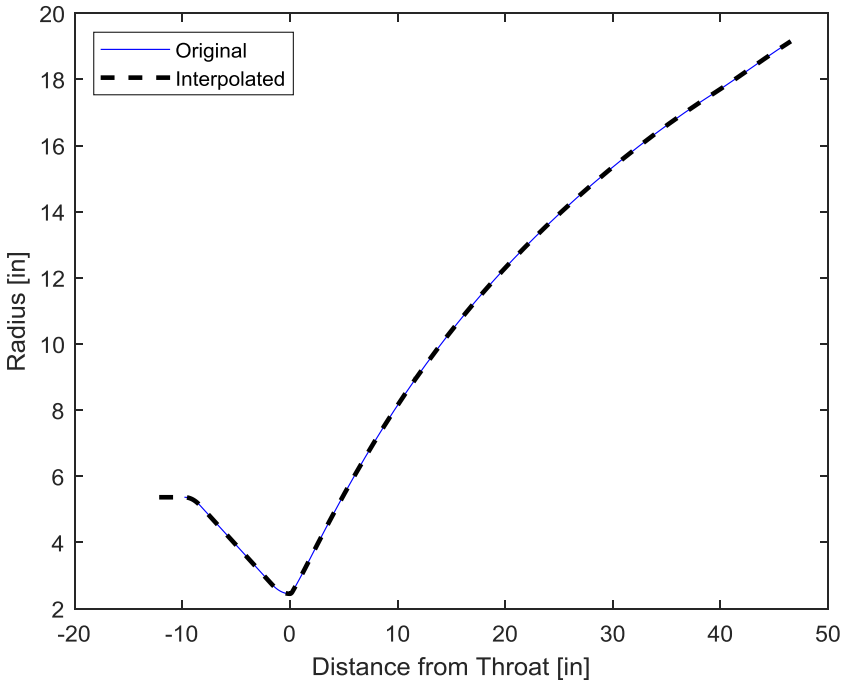


Figure 2.3 RL10 Axial Profile

Evenly spacing the geometry with a small discretization facilitates numerical integrations in the algorithm. The properties in Figure 2.2 were assumed constant from the minimum plotted x-value (-9.798 [in]) to -12.143 [in] and similarly discretized.

Using the known cooling channel cross-sectional area A_c and perimeter Per_c, the hydraulic diameter $D_{h,c}$ was found by

$$D_{h,c} = 4\frac{A_c}{Per_c}. \tag{2.1}$$

As shown in Binder's [10] Figure D1 (Figure 2.4 below),

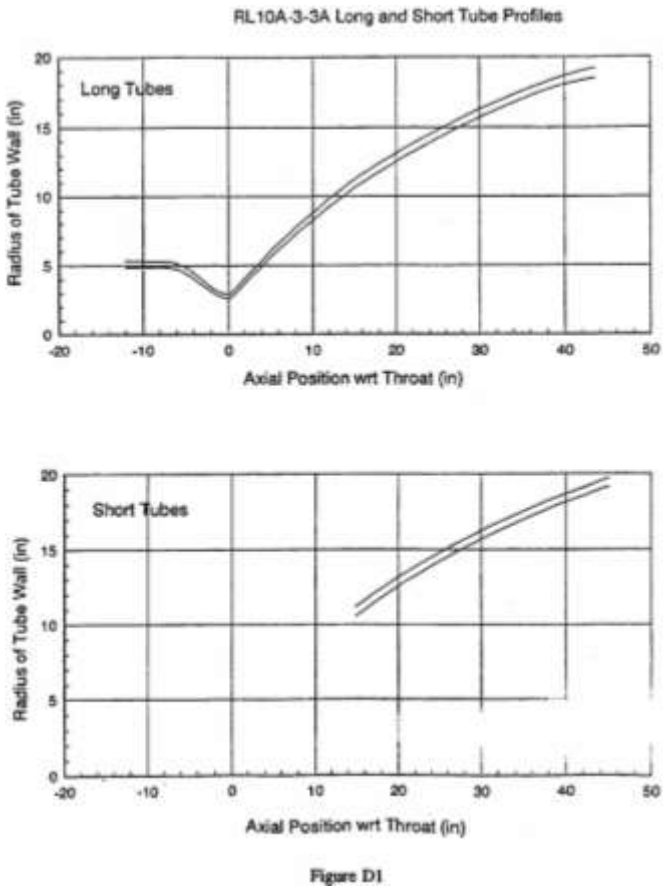


Figure 2.4 Binder [10] Figure D1 for RL10 Long and Short Tube Profiles

the cooling jacket is comprised of 180 “short” tubes which start approximately 32% down the length of the diverging nozzle, progress axially to the exit plane where they enter a turnaround manifold, and then flow back axially along the entire length of the contour to the injection plane, again in 180 “long” tubes. Figure 2.4 shows how the short and long tube radii match where they overlap from 14.846 [in] to the exit plane at 46.54 [in]. Per Figure D1 [10] and the RL10 geometry data [11], the channel hydraulic diameters along the engine axis are shown below.

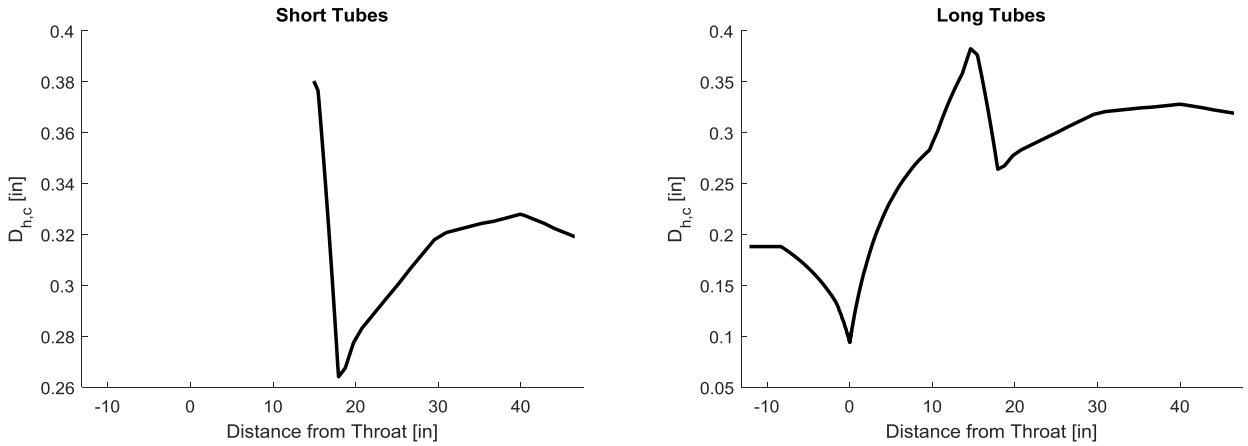


Figure 2.5 Cooling Channel Hydraulic Diameter Along Engine Axis

As shown in Figure 2.5, the long tube hydraulic diameter from the minimum x-value in the geometry data [11] to -12.143 [in] was assumed constant.

The RL10 cooling jacket in radial cross section would appear as shown in Figure 2.6.

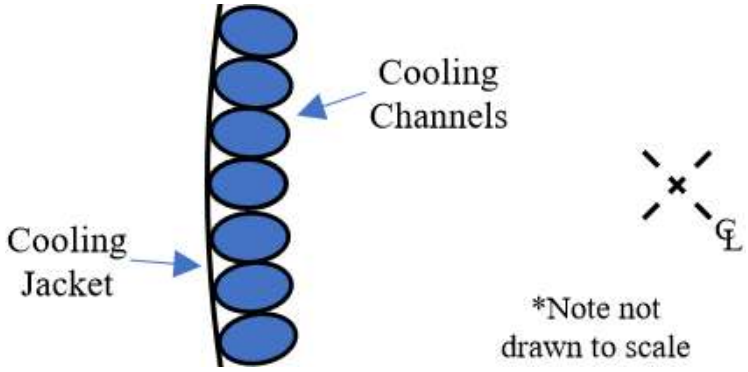


Figure 2.6 Cooling Jacket Radial Cross section

RL10 has the following key geometric characteristics collected in Table 2.1.

Table 2.1 RL10 Geometry

Property	**Symbol**	**Value**	**Units**	**Value**	**Units**
Expansion Ratio	ϵ	61.122			
Characteristic Length	L^*	0.953	m	37.499	in
<u>Diameter:</u>					
Chamber	D_c	0.273	m	10.733	in
Throat	D_t	0.124	m	4.899	in

Exit	D_e	0.973	m	38.301	in
Cross-sectional Area:					
Chamber	A_c	0.0584	m^2	90.479	in^2
Throat	A_t	0.0122	m^2	18.850	in^2
Exit	A_e	0.743	m^2	1.152□10^3	in^2

Note,

$$\epsilon = \frac{A_e}{A_t} \tag{2.2}$$

and

$$L^* = \frac{V_c}{A_t} \tag{2.3}$$

where chamber volume V_c includes the converging portion of the nozzle up to and including the throat. Volume was calculated using incremental frustum volumes $V_{frst,i}$

$$V_{frst,i} = \frac{1}{3}\pi h(R_{i-1}^2 + R_{i-1}R_i + R_i^2) \tag{2.4}$$

where R_{i-1} and R_i are the previous and current radii, respectively, and h is the axial distance

$$h = x_i - x_{i-1} \tag{2.5}$$

RL10 has key engine characteristics [10] as shown in Table 2.2.

Table 2.2 RL10 Engine Characteristics

Property	**Symbol**	**Value**	**Units**	**Value**	**Units**
Mixture Ratio	MR	5.26			
Mass Flow Rate:					

Total	$\dot{m}_{tot}$	16.96	kg/s	37.391	lbm/s
Fuel	$\dot{m}_f$	2.709	kg/s	5.973	lbm/s
Oxidizer	$\dot{m}_{ox}$	14.251	kg/s	31.418	lbm/s
Pressure:					
Chamber	P_c	$3.322 \square 10^6$	Pa	482	psia
LOX Pump Discharge	$P_{ox,inj}$	$4.551 \square 10^6$	Pa	660	psia

With mixture ratio defined as

$$MR = \frac{\dot{m}_{ox}}{\dot{m}_f} \tag{2.6}$$

and total mass flow rate defined as

$$\dot{m}_{tot} = \dot{m}_{ox} + \dot{m}_f. \tag{2.7}$$

From Eqs. ((2.6) and ((2.7), fuel mass flow rate is then

$$\dot{m}_f = \frac{\dot{m}_{tot}}{1+MR} \quad \text{and} \quad \dot{m}_{ox} = \dot{m}_{tot} - \dot{m}_f. \tag{2.8}$$

2.2 RL10 Chemistry

Combustion reaction products and resulting transport properties were analyzed using NASA's Chemical Equilibrium with Applications (CEA) [12]. This program was originally written in Fortran and NASA provides a downloadable executable for Windows. Philip Piper, a Purdue graduate student, created a MATLAB module which writes the input text file, runs the executable, reads the output text file, and formats the outputs into a MATLAB structure [13]. CEA's required inputs are shown in Table 2.3.

Table 2.3 CEA Inputs

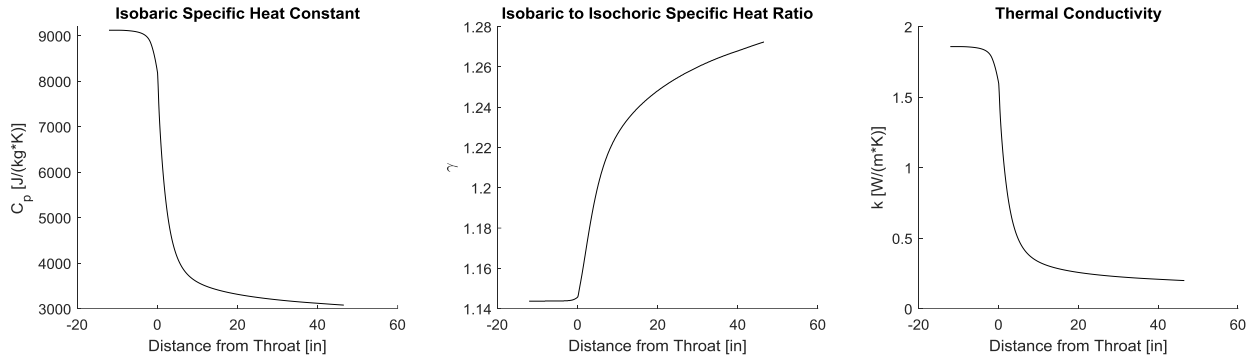
Input	**Value**
Mixture Ratio	5.26
Chamber Pressure	482
Chamber Pressure Units	psi
Chemistry Mode	Equilibrium
Fuel:	
Label	H2
Temperature	211.613
Temperature Units	K
Species Fraction of Composite Fuel	1
Oxidizer:	
Label	O2(L)
Temperature	90.15
Temperature Units	K
Species Fraction of Composite Fuel	1
Local Area Ratios:	
Subsonic	Various
Supersonic	Various

Local area ratios ϵ_i are defined as

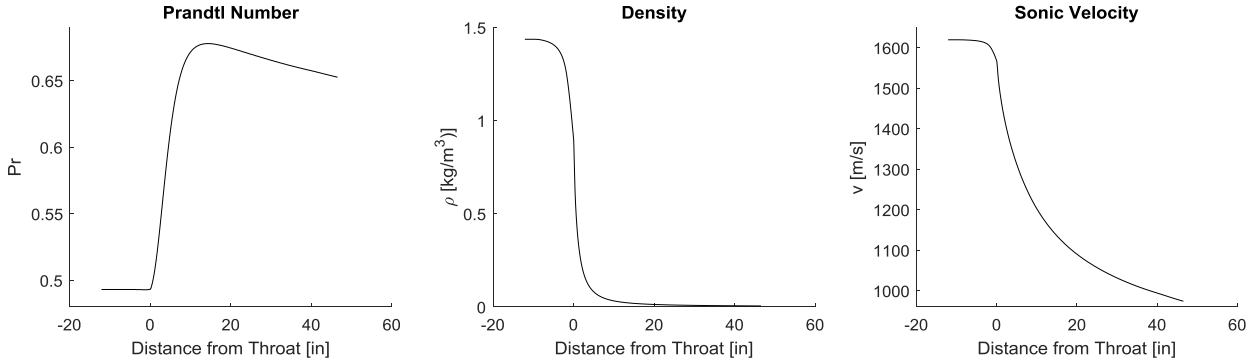
$$\epsilon_i = \frac{A_i}{A_t}. \tag{2.9}$$

The original CEA executable limits each call to six ϵ_i, so the length of RL10 is fully analyzed by iteration. Additionally, CEA only analyzes unique ϵ_i inputs, discarding any duplicate inputs, so RL10's contour data corresponding to unique radii was sorted in a subfunction using MATLAB's "unique" function into equally long vectors.

CEA outputs many properties as shown in Figure 2.7 along with the corresponding ϵ_i. A)



B)



C)

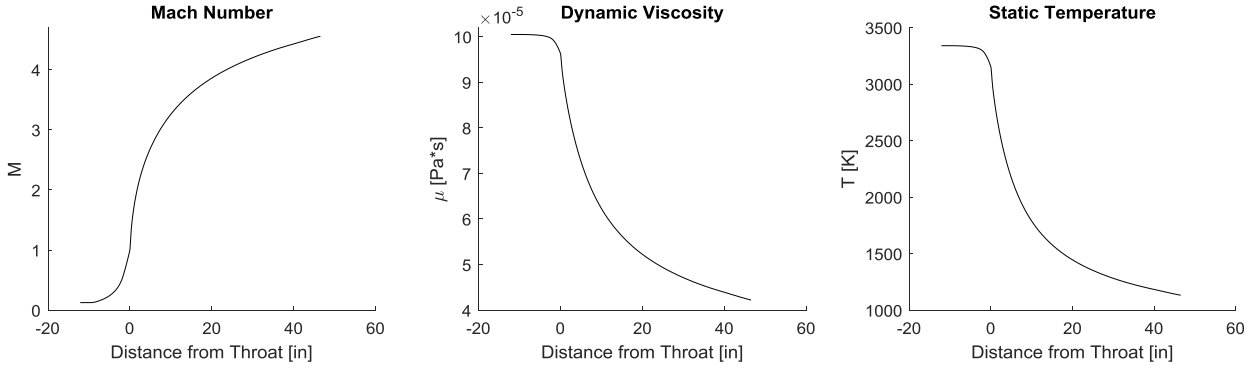
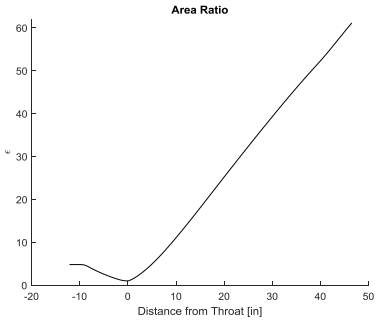


Figure 2.7 CEA Outputs Along RL10 Contour

Figure 2.7 continued

D)



Properties along the contour depict a typical isentropic gas expansion through a sonic throat (x = 0 [in]). Chamber temperature is a maximum at the head end of the combustion chamber with corresponding compressible flow properties in this subsonic region. Additionally, the combustion characteristic velocity c^* was calculated to be a constant 2,376.4 [m/s]. For reference, c^* is defined

$$c^* = \sqrt{\frac{R_u T_c}{\gamma \mathfrak{m}}} \left[\frac{2}{\gamma + 1}\right]^{-\frac{\gamma+1}{2(\gamma-1)}} \tag{2.10}$$

where R_u is the universal gas constant, T_c is the chamber temperature, γ is the isobaric to isochoric specific heat ratio, and $\mathfrak{m}$ is molecular weight of the of the chamber gas mixture [14].

2.3 RL10 Regenerative Cooling and Model Anchoring

With the product gas properties known along the contour, regenerative cooling analysis can begin. From [10] and [15], key cooling characteristics are known and collected in Figure 2.4.

Table 2.4 RL10 Cooling Characteristics

Property	**Symbol**	**Value**	**Units**	**Value**	**Units**
Cooling Channels:					
Number, Short	N_{chnl}	180			
Start Distance Along Nozzle Relative to Throat, Short		0.377	m	14.846	in
Number, Long	N_{chnl}	180			
Coolant:					
Species		Supercritical Hydrogen			
Initial Temperature	T_{init}	31.667	K	57	^{o}R
Initial Pressure	P_{init}	6.893□10^{6}	Pa	1000	psia
Temperature Rise	ΔT_{rse}	191.3	K	344.4	^{o}R
Cooling Jacket Pressure Drop	ΔP_{cl}	1.669□10^{6}	Pa	242.1	psid
Total Heat Transfer Rate	$\dot{Q}_{tot}$	8.434*10^{6}	W	7994	Btu/s
Chamber/Nozzle:					
Material		Stainless Steel 347			
Average Wall Thickness	th_w	3.302□10^{-4}	m	0.013	in
Thermal Conductivity	k_w	16.3	W/(m□K)	133	Btu/(hr□ft□^{o}F)

RL10 uses a silver throat insert that increases the exit expansion ratio to the 61.122 value used, but the effect of the silver's thermal conductivity on local heat transfer is neglected and any associated errors are compensated by the process of anchoring to Binder [10].

As the initially high-pressure liquid hydrogen flows through the tubes, it absorbs heat generated from combustion in the chamber, and soon transitions to a fully supercritical fluid. When the coolant's temperature and pressure change so do its transport properties which affect heat transfer. The transcritical hydrogen behavior leads to large variations in these properties which are accurately captured in the model. Due to small channel dimensions and high velocity, the coolant loses significant pressure due to friction with the passage walls.

Axial symmetry is assumed so only a single channel is analyzed with appropriate per channel flow. Coolant temperature increment looks at the entire circumference of incremental surface area and total coolant mass flow rate, thereby assuming the conditions in every tube at each station are the same.

Heat is transferred from the hot gas to the coolant through three major thermal resistances: forced hot gas convection in the chamber, conduction through the wall, and forced liquid/supercritical convection in the cooling passages. Thermal radiation is neglected. A diagram of this scenario is illustrated in Figure 2.8.

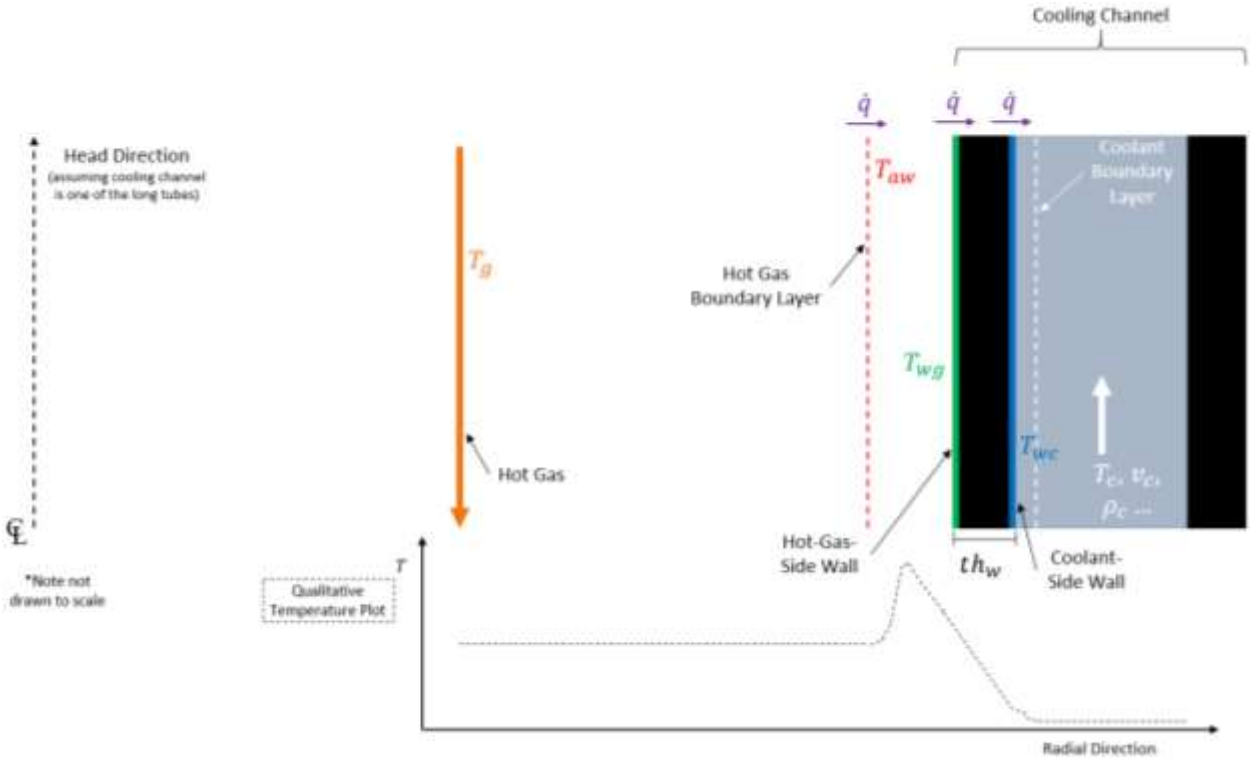


Figure 2.8 Regenerative Cooling Heat Transfer Scenario

Alternatively, the heat transfer scenario can be visualized like the thermal circuit shown in Figure 2.9.

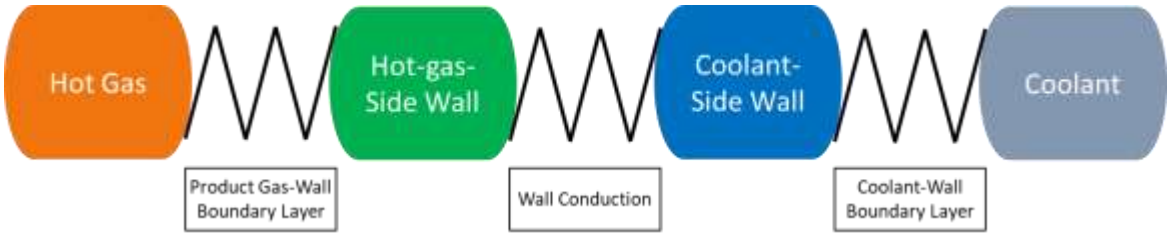


Figure 2.9 Regenerative Cooling Thermal Circuit Diagram

To begin the analysis, initial coolant temperature T_c as delivered by the upstream plumbing is known from Binder [10]. Hot-gas-side and coolant-side wall temperatures (T_{wg} and T_{wc}) are unknown, but gas temperature T_g is known from CEA. By guessing a reasonably low hot-gas wall temperature, the iteration coolant temperature $T_{c,iter}$ can be calculated. The first guess will be lower than the known coolant temperature, so by using a small increment on T_{wg}, the heat transfer can be iterated until T_c and $T_{c,iter}$ match. After reaching steady state, the whole energy balance is

$$\dot{q} = h_g\left(T_{aw} - T_{wg}\right) = \frac{k_w}{th_w}\left(T_{wg} - T_{wc}\right) = h_c\left(T_{wc} - T_{c,iter}\right), \tag{2.11}$$

where $\dot{q}$ is the heat rate flux through the whole circuit, h_g is the hot-gas-side convective heat transfer coefficient, and h_c is the coolant-side convective heat transfer coefficient.

Keeping in mind that a lack of subscript refers to local station conditions, temperature iteration begins by calculating the adiabatic wall temperature T_{aw}:

$$T_{aw} = T_{g,c}\left[\frac{1+\frac{\gamma-1}{2}M^2 r}{1+\frac{\gamma-1}{2}M^2}\right]. \tag{2.12}$$

The chamber stagnation gas temperature $T_{g,c}$ is multiplied by the bracketed factor which includes the gas's ratio of isobaric to isochoric specific heats γ Mach number M, and recovery factor number r. This factor accounts for the compression heating of the gas as its kinetic energy is converted to heat near the wall and is calculated by

$$r = Pr^n \tag{2.13}$$

where the Prandtl number ($Pr = \frac{\mu C_p}{k}$ and provided by CEA) is raised to a power ($n = \frac{1}{3}$ for this turbulent boundary layer). These first three temperature profiles for RL10 are overlaid in Figure 2.10.

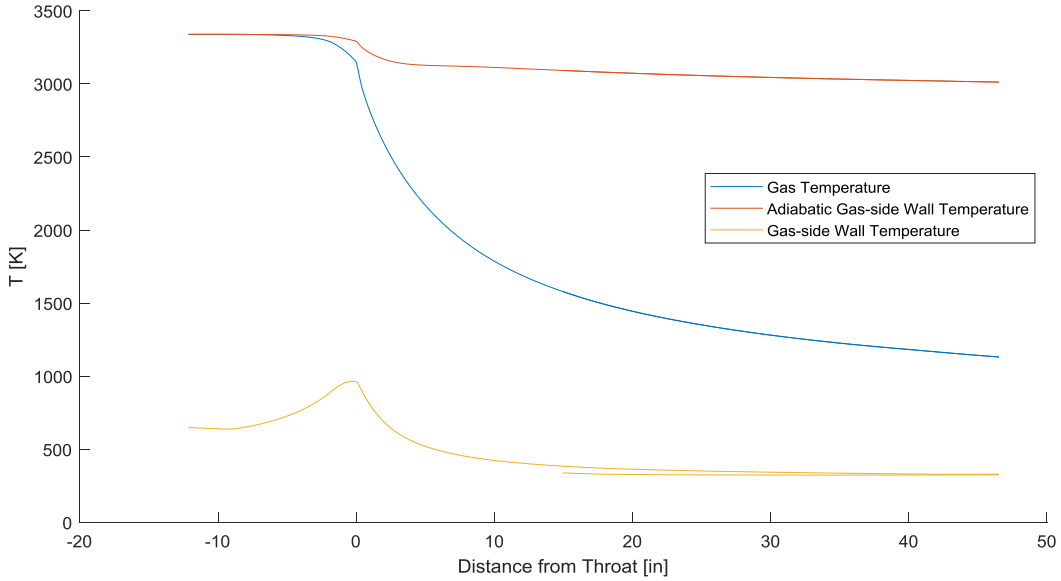


Figure 2.10 RL10 T_g, T_{aw}, and T_{wg} Profiles

Next the Bartz relation from Huzel and Huang [16] was used to find h_g:

$$h_g = a\left(\frac{A_t}{A}\right)^{0.9}\sigma. \tag{2.14}$$

The local inverse area ratio $\frac{A_t}{A}$ raised to the 0.9 power is multiplied by the transport properties factor a and the correction factor for properties across the boundary layer σ:

$$a = \frac{0.026}{D_t^{0.2}}\left(\frac{\mu^{0.2}C_p}{Pr^{0.6}}\right)_{ns}\left(\frac{(P_c)_{ns}g}{c^*}\right)^{0.8}\left(\frac{D_t}{R}\right)^{0.1}, \tag{2.15}$$

$$\sigma = \frac{1}{\left[\frac{1}{2}\frac{T_{wg}}{T_{g,c}}b+\frac{1}{2}\right]^{0.68} b^{0.12}}, \tag{2.16}$$

$$b = 1 + \frac{\gamma-1}{2}M^2. \tag{2.17}$$

Subscript "ns" refers to chamber stagnation conditions. a considers throat diameter D_t, dynamic viscosity μ, isobaric specific heat C_p, Prandtl number Pr, chamber pressure P_c and throat curvature R. Curvature was found using Chernov's [17] circle fitting MATLAB function which follows Taubin's method (1991). The σ parameter requires hot-gas-side wall temperature T_{wg}, chamber temperature $T_{g,c}$, isobaric to isochoric specific heat ratio γ, and Mach number M.

Careful attention is required not only for property location (local station or chamber stagnation), but also for property units when using the Bartz equation. These units are tabulated below for clarity.

Table 2.5 Bartz Convective Heat Transfer Coefficient Symbology and Units

Input	**Symbol**	**Units**
Throat Diameter	D_t	in
Dynamic Viscosity	μ	lbm/(in□s)
Prandtl Number	Pr	N/A
Chamber Pressure	P_c	psi
Earth's Gravitational Acceleration	g	ft/s^2
Engine Characteristic Velocity	c^*	ft/s
Throat Curvature	R	in
Hot-Gas-Side Wall Temperature	T_{wg}	^{o}R
Chamber Temperature	T_c	^{o}R
Isobaric to Isochoric Specific Heat Ratio	γ	N/A
Mach Number	M	N/A
Output	**Symbol**	**Units**
Convective Heat Transfer Coefficient	h_g	Btu/(s□in^2□^{o}R)

h_g can also be [Btu/(s□in^2□°F)] if combined with temperatures with units of [°F] in subsequent heat flux calculations. Figure 2.11 shows how h_g varies throughout RL10 with the expected peak at the engine's throat.

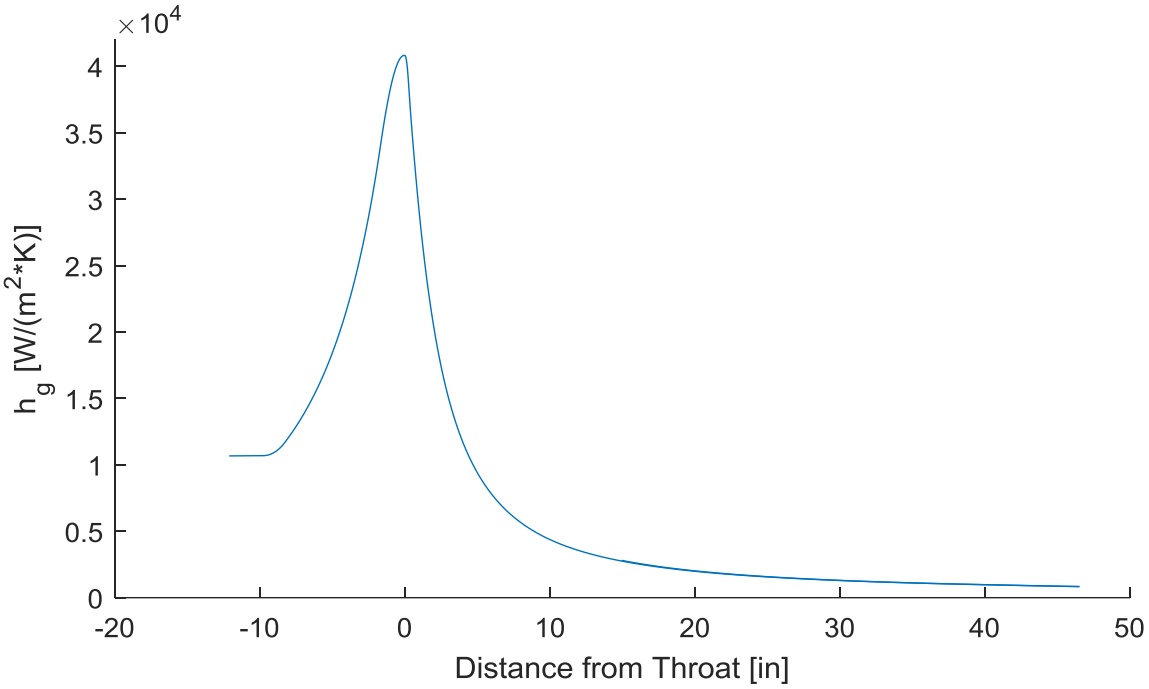


Figure 2.11 RL10 Hot-Gas-Side Convective Heat Transfer Coefficient Profile

After finding h_g, $\dot{q}$ can be found:

$$\dot{q} = h_g(T_{aw} - T_{wg}). \tag{2.18}$$

$\dot{q}$ varies like h_g as shown below in Figure 2.12, again peaking at the throat. Note that $\dot{q}$ has been

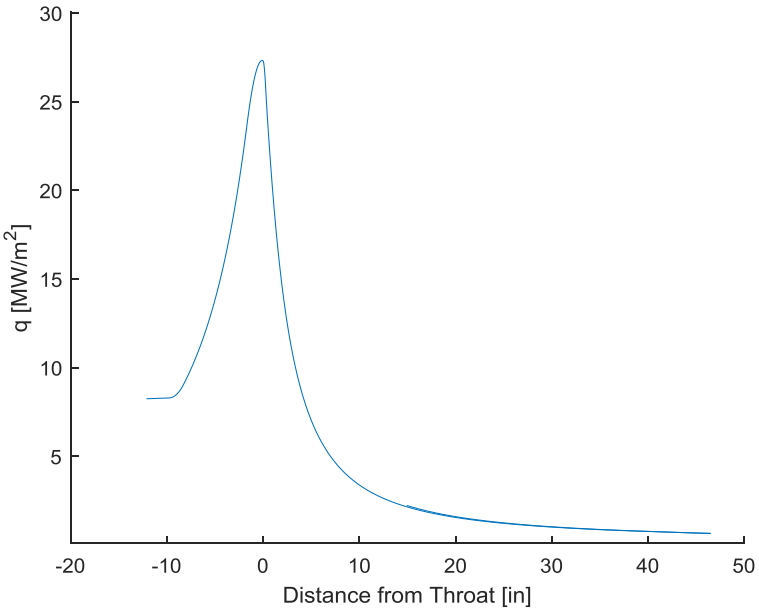


Figure 2.12 RL10 Heat Flux Profile

multiplied by the Binder-anchoring coefficient in the above plot. Heat flux from this model follows a trend similar to what Binder [10] predicted and peaks near the throat (x = -0.087 [in]) with a value of 16.7 [Btu/(s□in^2)]. Binder's [10] heat flux values (see Figure 2.13) are higher because

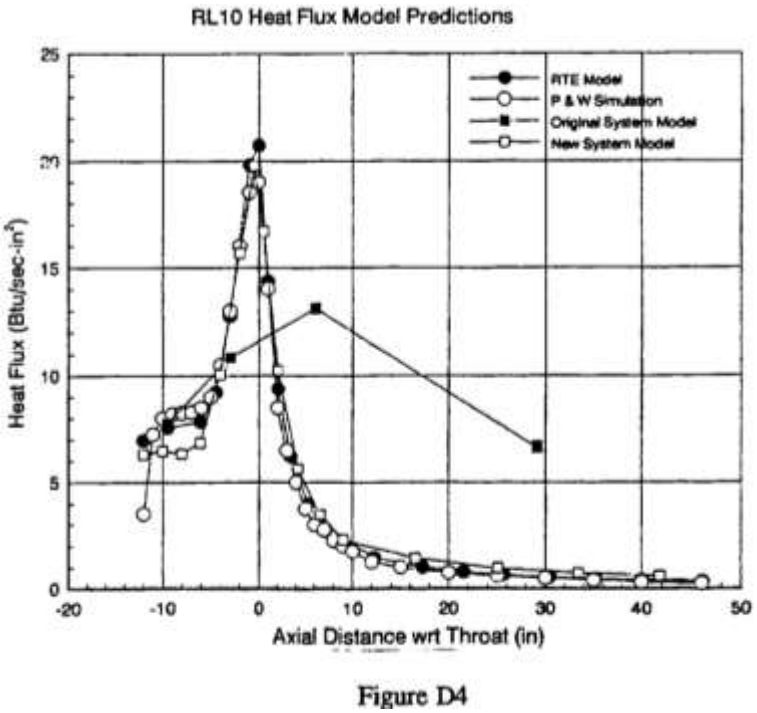


Figure 2.13 Binder [10] Figure D4 for RL10 Heat Flux

Figure 2.12's profile has already been multiplied by the Binder-anchoring coefficient for convective heat transfer coefficient which is less than 1.

Solving for T_{wc},

$$T_{wc} = T_{wg} - \left(\frac{th_w}{k_w}\dot{q}\right). \tag{2.19}$$

To find $T_{c,iter}$, first h_c must be found. According to [16], the appropriate relation for this scenario where heat is transferred through a vapor-film boundary layer with coolant at supercritical temperature and pressure is

$$h_c = \frac{0.029 C_p \mu^{0.2}}{Pr^{2/3}} \left(\frac{T_c}{T_{wc}}\right)^{0.55} \left(\frac{G^{0.8}}{D_{h,c}^{0.2}}\right) \tag{2.20}$$

Inputs for coolant at the current station's known temperature and pressure were returned from NIST's REFPROP substance property database [18]. NIST's MATLAB extension [18] requires

temperature be input in [K] and pressure in [kPa]. Properties are output in SI units. Also note the use of the T_c in the h_c calculation. G represents the coolant weight flow per channel.

$$G = \frac{\dot{m}_c}{A_c N_{chnl}}, \tag{2.21}$$

where $\dot{m}_c$ is the total coolant mass flow rate and A_c is a single tube's cross-sectional area (from RL10 contour data [11]). Here, $D_{h,c}$ again refers to a single tube's hydraulic diameter. The final analysis yielded profiles of $D_{h,c}$ and coolant velocity (see Figure 2.14) which clearly shows how constricting the channel's hydraulic diameter near the throat sharply increases the coolant velocity $\left(v_c \propto 1/D_{h,c}^2\right)$, thereby enhancing the heat transfer in this extreme region.

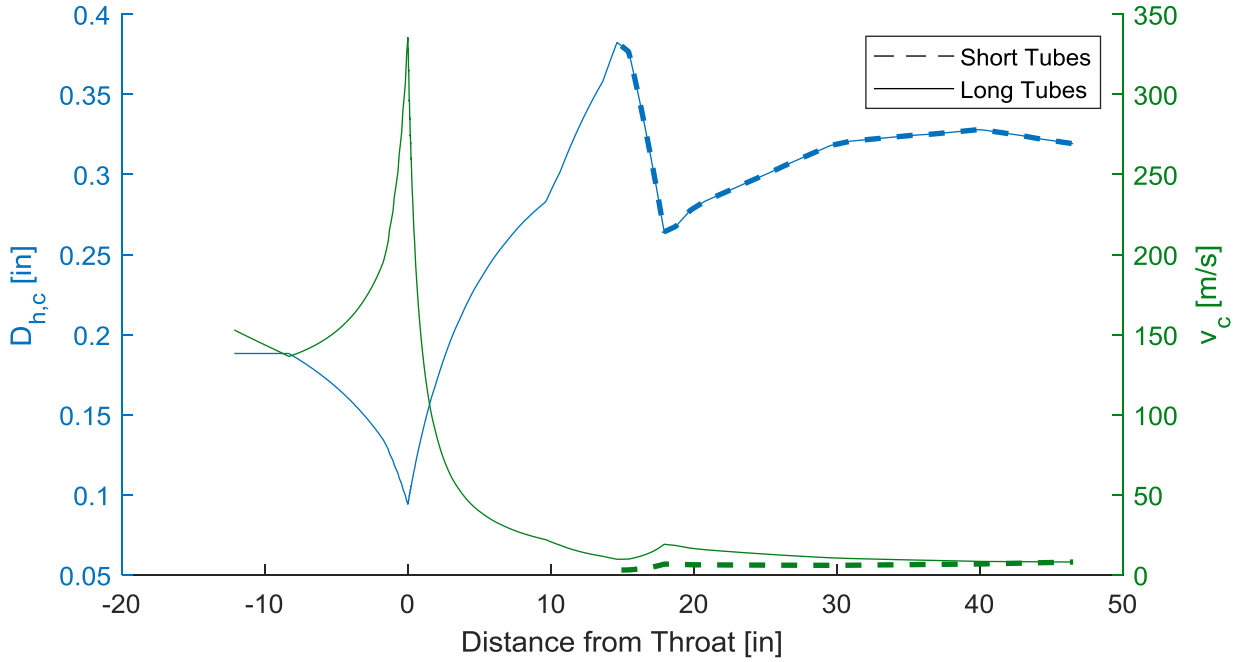


Figure 2.14 Coolant Channel Constriction Effect on Coolant Velocity

Figure 2.15 shows how h_c varies throughout RL10 with the expected peak at the throat where the cooling channels are most constricted.

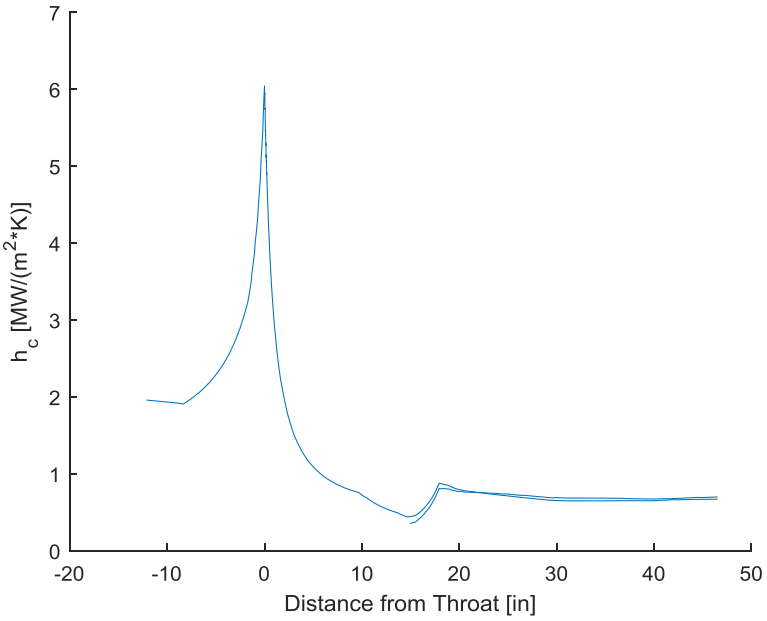


Figure 2.15 RL10 Coolant's Convective Heat Transfer Coefficient Profile

Solving for $T_{c,iter}$,

$$T_{c,iter} = T_{wc} - \frac{\dot{q}}{h_c}. \tag{2.22}$$

Again, the first iteration's resultant $T_{c,iter}$ will be too low. Increments on guessed hot-gas-side wall temperature were set to 1 ^{o}R to avoid significant overshoot on coolant temperature.

Progressing to the next station is accomplished by accounting for the coolant's incremental heat absorption and corresponding pressure drop. For the latter, the Darcy-Weisbach equation [19] determines the incremental pressure drop ΔP

$$\Delta P = f_D \frac{\rho}{2} \frac{v_c^2}{D_h} \Delta L. \tag{2.23}$$

The equation requires the local coolant density ρ, coolant velocity v_c, the incremental channel length ΔL, and the Darcy friction factor f_D. For turbulent flow in smooth pipes, f_D can be found from the Karman-Prandtl resistance equation [20]

$$\frac{1}{\sqrt{f_D}} = 1.93 \log\left(Re\sqrt{f_D}\right) - 0.537. \tag{2.24}$$

This nonlinear equation requires the coolant's local Reynolds number Re

$$Re = \frac{\rho|v|D_h}{\mu} \tag{2.25}$$

and was solved using the bisection method.

Incremental heat absorption came from multiplying the heat flux and incremental heated surface area SA_{inc}

$$\dot{Q} = \dot{q}SA_{inc} \tag{2.26}$$

SA_{inc} was found using the conical frustum surface area equation

$$SA_{inc} = \pi D_{avg}\sqrt{(R^i - R^{i+1})^2 + h^2} \tag{2.27}$$

which itself uses the frustum average diameter D_{avg}

$$D_{avg} = R^i + R^{i+1} \tag{2.28}$$

and the frustum height

$$h = x^i - x^{i+1}, \tag{2.29}$$

and where subscript "i" is current station and "i+1" is next station. Coolant temperature increase was calculated using the sensible thermal energy storage equation for flows

$$\dot{Q} = \dot{m} C_p^i \Delta T \tag{2.30}$$

where the current station's isobaric specific heat constant C_p^i was used instead of an average since T_c^{i+1} is not yet known to query REFPROP. The change in temperature from station to the next ΔT is

$$\Delta T = T_c^{i+1} - T_c^i, \tag{2.31}$$

so the next station's temperature T_c^{i+1} was

$$T_c^{i+1} = \frac{\dot{Q}}{\dot{m} C_p} + T_c^i \tag{2.32}$$

An edge case exists when the coolant reaches the turnaround manifold. Here at the exit plane, difference values that were forward-facing were instead made backward-facing. Additionally, pressure drop changes slightly to account for the stagnation and redirection. According to Babcock and Wilson Co. [21], the bend loss coefficient k_b for pipes bending 180° is 1.1. The altered pressure drop equation [21], [22] is

$$\Delta P = \frac{1}{2} f_D \rho |v|^2 \frac{\pi R_b}{D_h} \frac{\theta}{180} + \frac{1}{2} k_b \rho |v|^2. \tag{2.33}$$

This bend loss equation requires the bend radius R_b and the bend angle θ in degrees (180° for this full turnaround). R_b was assumed to be the local single channel hydraulic radius $\frac{D_{hc}}{2}$.

The same process was carried out for the long tubes running from the nozzle exit to the injection plane. Finally, total coolant temperature and pressure drop could be compared to the values reported by Binder [10]. Constant coefficients were applied to $\dot{q}$ and f_D (see Table 2.6) to alter this model's predicted change in coolant temperature and pressure to match Binder's [10]

Table 2.6 Binder-Anchoring Coefficients to Match Binder [10] Coolant ΔT and ΔP

Coefficient For	Symbol	Value
$\dot{q}$	$C_{B,hf}$	0.28685
f_D	$C_{B,fd}$	9.9606

results. The heat flux coefficient is lower which is expected since this analysis uses full CEA-predicted combustion temperature and associated properties everywhere, when in reality there are sections of mixture ratio biasing and other three-dimensional flow effects that make the average TCA environment less severe. Friction must be significantly increased because the supercritical coolant constrained in small channels will experience three dimensional effects like vortices which were not included in the model.

Using these Binder-anchoring coefficients, RL10's coolant pressure profile along with the corresponding f_D is shown in Figure 2.16. In this plot, f_D has already been multiplied by the

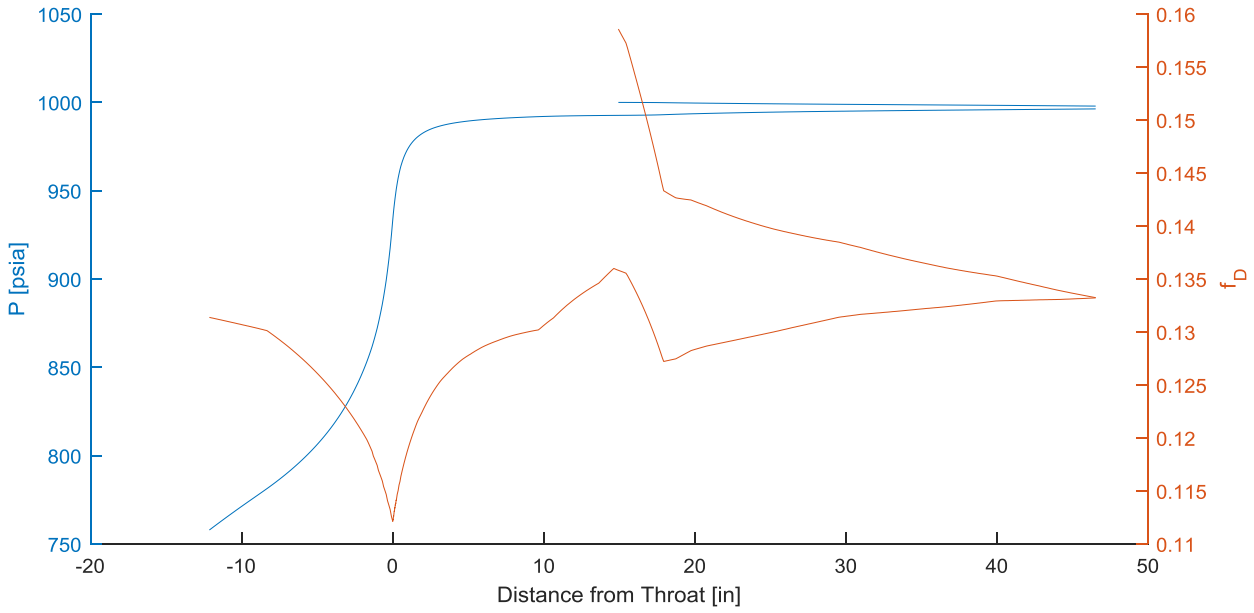


Figure 2.16 RL10 Coolant Pressure and Darcy Friction Factor Profiles

Binder-anchoring coefficient from Table 2.6. The blue pressure curve shows the same total coolant pressure drop as Binder [10] and follows a similar profile along the engine as shown in Figure D6 of [10] copied below (see Figure 2.17) for convenience. The final coolant pressure is

757.9 [psia]. In Binder's [10] plot, the "New System Model" curve represented by a solid black line and square markers is most pertinent for comparison.

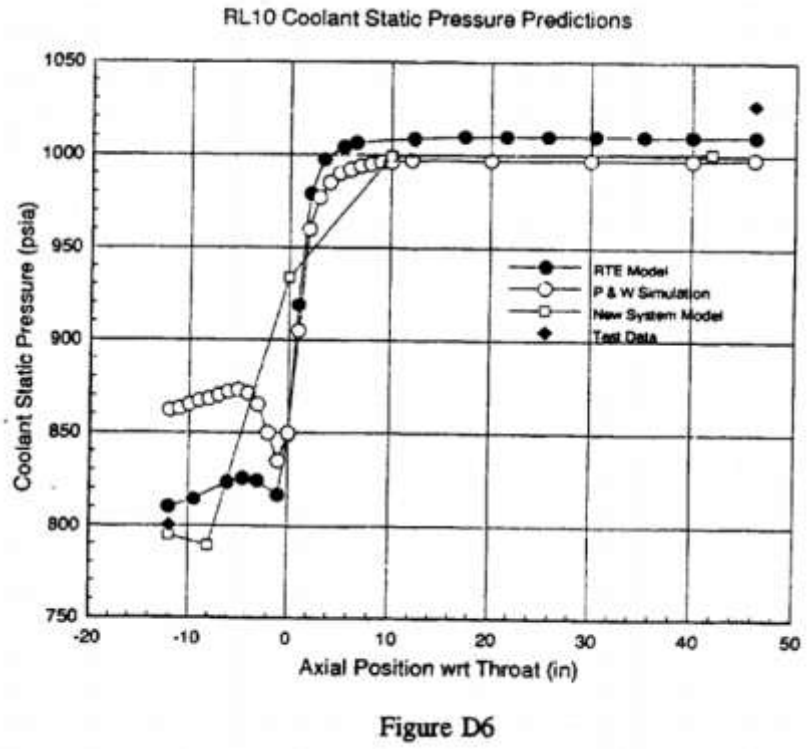


Figure 2.17 Binder [10] Figure D6 for RL10 Coolant Pressure

Wall and coolant temperatures are shown in Figure 2.18.

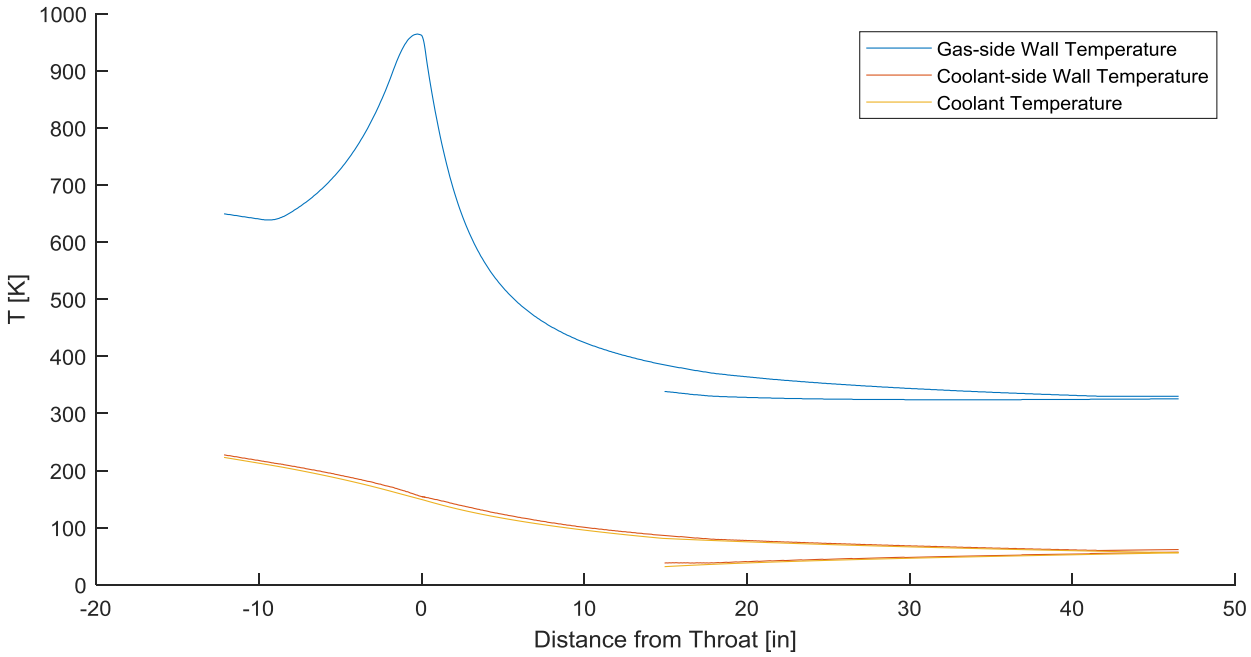


Figure 2.18 RL10 Wall and Coolant Temperature Profiles

The hot-gas wall temperature T_{wg} is moderate except in the throat region where it peaks at 964.4 [K] or 1,276.3 [^{o}F]. Binder's [10] hot-gas-side wall temperature results are included below (see Figure 2.19) for comparison. Binder's [10] New System Model maximum hot-gas-side wall temperature is approximately 1,044 [K] or 1,420 [^{o}F]; the minimum is approximately 264 [K] or 15 [^{o}F].

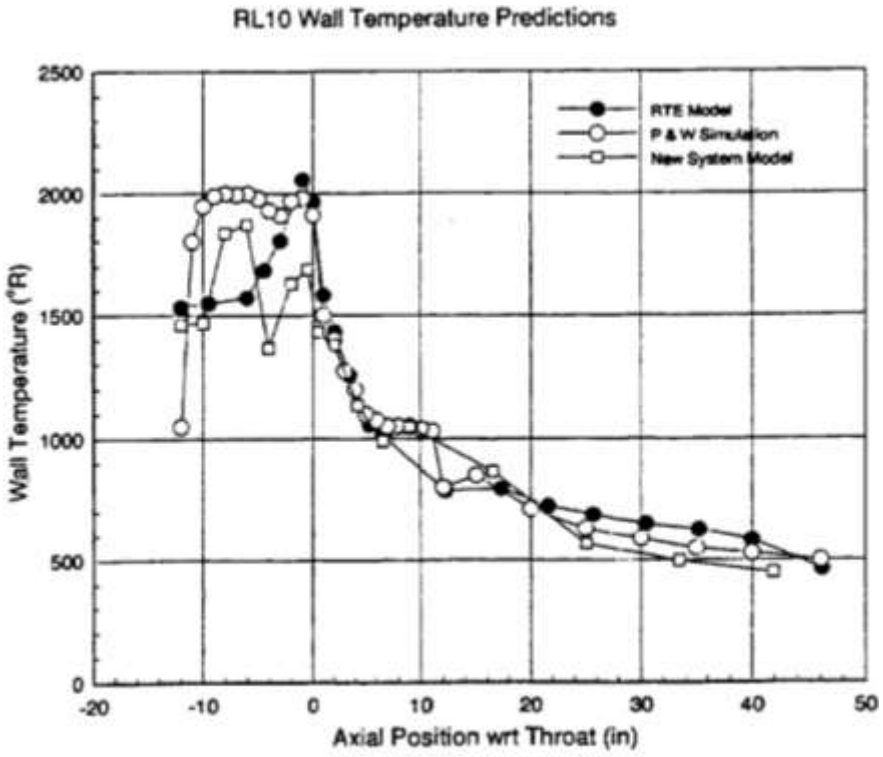


Figure 2.19 Binder [10] Figure D5 for RL10 Hot-Gas-Side Wall Temperature

Hydrogen's excellent coolant properties are evident in how dramatically cool it keeps the coolant-side of the very thin jacket wall, with a peak T_{wc} at the injection plane of only 227.3 [K] or -50.6 [^{o}F]. The coolant temperature itself peaks in the same location at 223 [K] or -58.3 [^{o}F]. T_{wg} follows a similar profile as Binder's [10] analysis depicted in his Figure D7 (see Figure 2.20). Maximum coolant temperature in Binder's [10] New System Model is approximately 250 [K] or -10 [oF]; the minimum is approximately 31 [K] or -404 [oF].

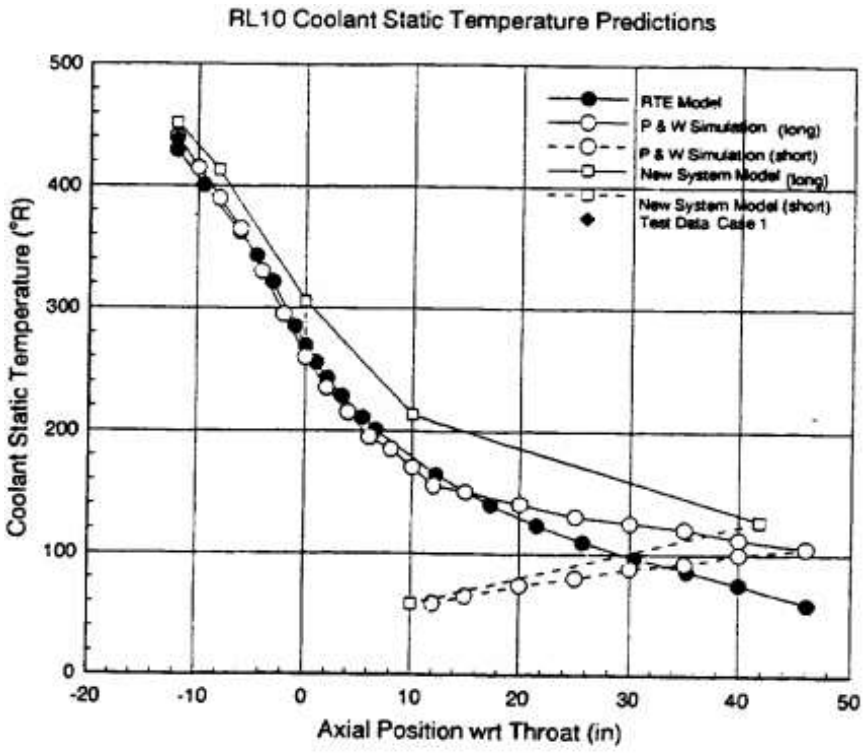


Figure D7

Figure 2.20 Binder [10] Figure D7 for RL10 Coolant Temperature

Given the matching coolant temperature rise and pressure drop, and the similar temperature and pressure axial profiles, the model is anchored. Finally, the total heat transfer rates of two models are compared in Figure 2.21 with this model's cumulative profile in blue.

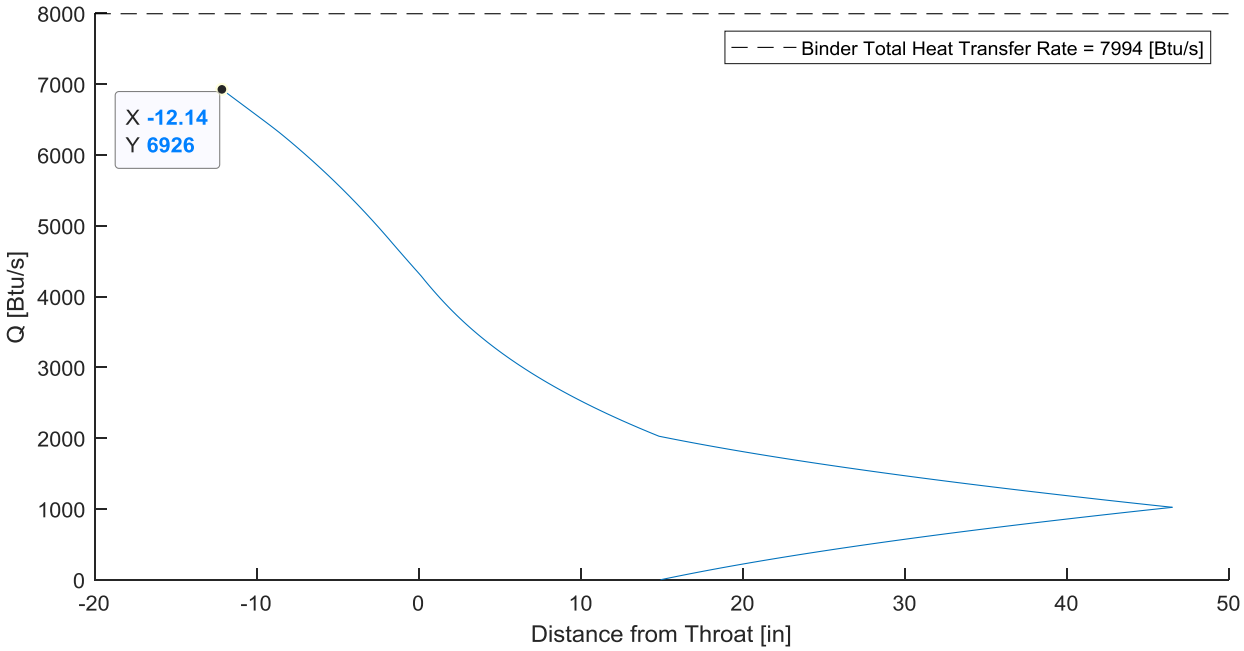


Figure 2.21 RL10 Cumulative Heat Transfer Rate

Binder's [10] prediction is slightly higher which is a result of his more detailed model including three-dimensional flow and different methods of obtaining fluid transport properties. Nevertheless, this simplified treatment yields results that closely approximate prior predictions and measurements of the RL10 engine.

3. RESULTS

3.1 Developing an RDE Thrust Chamber Comparable to RL10

RL10's geometric and performance characteristics (additionally distinguished with the subscript "RL10" when necessary for clarity) provide the necessary inputs to construct a comparable RDE model. The ultimate goal of this hypothetical RDE would be to replace the RL10 constant pressure engine as the upper stage of a future launch vehicle. For easier integration, the RDE would need to fit in the same volumetric envelope as the RL10, so the RDE's outer diameter D_o was set to RL10's chamber diameter. While RDE technology is immature, performance analysis on upper stage applications using hydrogen/oxygen propellants indicates potential for specific impulse improvements as high as 12.2% [1]. Cooling jacket manufacturing techniques have advanced substantially since the RL10 was developed in the late 1950s. To be conservative, propellant mass flow rates, mixture ratio, and wall material were also matched with RL10. Also, no cooling channel design optimization was performed specific to the RDE. The channel dimensions used are the RL10 cooling channel dimensions at the corresponding axial location, except for the inner circuit which was scaled down as explained later.

Rocket RDEs typically have straight annular detonation channels, and the product gases are sonically choked by the channel exit, so the channel can be thought of as the pressure gain engine analogue of a constant pressure engine throat. The RDE detonation channel cross-sectional area A_{chnl} was set equal to RL10's throat cross-sectional area $A_{t,RL10}$. To find the channel width, the RDE's solid centerbody cross-sectional area A_{cntr} was first determined:

$$A_{cntr} = A_o - A_{chnl}, \tag{3.1}$$

$$A_o = \frac{\pi D_o^2}{4}, \tag{3.2}$$

$$A_{chnl} = A_{t,RL10}. \tag{3.3}$$

Therefore, the centerbody diameter D_{cntr} is

$$D_{center} = \sqrt{\frac{4*A_{cntr}}{\pi}}. \tag{3.4}$$

Given the outer and center diameters, the channel width w_{chnl} was calculated by

$$w_{chnl} = \frac{D_o - D_{cntr}}{2}. \tag{3.5}$$

The length of the annular RDE combustion chamber is determined by the fill height L_{fill} of the reactants before they are detonated. The computed L_{fill} is multiplied a factor of safety of 2 ($FS_{fill} = 2$). Quasi-steady state detonation behavior can manifest as a number of different wave configurations which are non-trivial to control, however the single wave case would give the reactants the most time to fill before the next wave arrival. This conservative arrival time estimate t_{arvl} is therefore

$$t_{arvl} = \frac{D_o}{v_{CJ}}, \tag{3.6}$$

where Chapman-Jouguet detonation wave velocity v_{CJ}, calculated via NASA's Chemical Equilibrium with Applications detonation module, provides a good estimate of actual wave speed. Actual wave velocity may be slower due to effects like side wall area relief and parasitic deflagration but may even be greater due to certain detonation over-driving phenomena. FS_{fill} helps account for this variability so that no unreacted propellants escape the chamber between successive wave arrivals.

The fill height also depends on the injection design. It was calculated using two methods, one using mixed reactant temperature T_{mix} and molecular weight $\mathfrak{m}_{mix}$ and the other using an unhindered LOX jet, with the latter more conservative height being chosen for the model.

The first method started with liquid hydrogen temperature $T_{f,liq}$ increased by the known temperature rise through the regenerative cooling jacket $\Delta T_{f,clngjckt}$ [10] to find the fuel injection temperature $T_{f,inj}$:

$$T_{f,inj} = T_{f,liq} + \Delta T_{f,clngjckt}. \tag{3.7}$$

This temperature could be input to the hydrogen gas Shomate equation for isobaric specific heat corresponding to a temperature range of 298-1000 [K] [23]:

$$\begin{gathered} C_{p,f} = 33.066178 - 11.363417 * t + 11.432816 * t^2 - \cdots \\ \ldots 2.772874 * t^3 - 0.158558/t^2, \\ t = \frac{T_{f,inj}}{1000}. \end{gathered} \tag{3.8}$$

Similarly, oxygen specific heat corresponding to a temperature range of 100-700 [K] could be found using oxygen's standard liquid temperature (assuming no oxidizer was used for regenerative cooling) [24]:

$$\begin{gathered} C_{p,ox} = 31.32234 - 20.23531 * t + 57.86644 * t^2 - \cdots \\ \ldots 36.50624 * t^3 - 0.007374/t^2, \\ t = \frac{T_{ox,liq}}{1000}. \end{gathered} \tag{3.9}$$

The Shomate equation gives specific heat in units of [J/(mol□K)]. Reactant mixture pre-detonation temperature could be found from combining the first law of thermodynamics, extended to a power balance for the fuel and oxidizer less than energy required to vaporize the oxidizer, with the reactant mixture's thermal conductance:

$$T_{mix} = \frac{[\dot{m}_f C_{p,f} T_f + \dot{m}_{ox} C_{p,ox} T_{ox}] - \dot{m}_{ox} h_{v,ox}}{\dot{m}_f C_{p,f} + \dot{m}_{ox} C_{p,ox}}. \tag{3.10}$$

Reactant mass fractions

$$y_i = \frac{\dot{m}_i}{\dot{m}_{tot}} \tag{3.11}$$

and molecular weights $\mathfrak{m}_i$ (subscript "i" denotes the ith species) were used to find mixture molecular weight

$$\mathfrak{m}_{mix} = \frac{1}{\sum_{i=1}^{N} \frac{y_i}{\mathfrak{m}_i}} = \frac{1}{\frac{y_f}{\mathfrak{m}_f} + \frac{y_{ox}}{\mathfrak{m}_{ox}}} \tag{3.12}$$

where N is the total number of species.

The pre-detonation mixture can be approximated as an ideal gas, so the mixture density ρ_{mix} was

$$\rho_{mix} = \frac{P_c}{\frac{R_u}{\mathfrak{m}_{mix}} * T_{mix}} \tag{3.13}$$

where P_c is pre-detonation chamber pressure and R_u is the universal gas constant (8.314 [J/(mol□K)]). The mixed injection velocity v_{mix} is

$$v_{mix} = \frac{\dot{m}_{tot}}{\rho_{mix} * A_{chnl}}. \tag{3.14}$$

Combining t_{arvl} with this fill velocity v_{mix} gives the first method's fill height $L_{fill,1}$:

$$L_{fill,1} = v_{mix} * t_{arvl}. \tag{3.15}$$

The second method for determining fill height, the LOX jet, uses Bernoulli's equation for injection velocity. Assuming the LOX jet bulk velocity can be considered on a streamline and the fluid is incompressible, is steady, and experiences no viscous losses, then

$$P_{ox,inj} + \frac{1}{2}\rho_{ox,inj}v_{ox,inj}^2 + \rho_{ox,inj}gz_{ox,inj} = P_c + \frac{1}{2}\rho_{ox,inj}v_c^2 + \rho_c g z_c \tag{3.16}$$

where subscript "ox,inj" denotes properties at the injection plane and subscript "c" denotes properties in the chamber. $P_{ox,inj}$ was assumed to be RL10's LOX pump discharge pressure from Table 2.2.

The gravitational potential term is negligible given the proximity of the injector and chamber, and the velocity of the jet goes to near zero in the chamber due to atomization, so the relation simplifies to

$$v_{ox,inj} = \sqrt{\frac{2(P_{ox,inj} - P_c)}{\rho_{ox,inj}}}. \tag{3.17}$$

Multiplying $v_{ox,inj}$ and t_{arvl} gives the second method's fill height $L_{fill,2}$.

$$L_{fill,2} = v_{ox,inj} * t_{arvl} \tag{3.18}$$

Table 3.1 summarizes the fill height analyses; method 2, again, was chosen for conservatism.

Table 3.1 Fill Height Summary

Method	**Fill Velocity**		L_{fill} **w/** FS_{fill}	
	[m/s]	**[in/s]**	**[mm]**	**[in]**
1	33.666	$1.325 \square 10^3$	16.848	0.663
2	46.375	$1.826 \square 10^3$	23.209	0.914

RDEs' annular combustion chambers naturally lend themselves to aerospike expansion. After the annular section, the inner wall was assumed to have a 15-degree tapering profile ending at a certain minimum inner wall diameter, thereby forming a plug nozzle contour. Initially this plug nozzle termination diameter was assumed to be 1 [in] to approach an aerospike design. This proved too difficult to cool and manufacturing cooling channels small enough to fit near the plug would be challenging. Next the diameter was designed such that the plug nozzle terminated

where flow separation would likely occur when the exit pressure P_e was about 1/3 of the ambient pressure P_a at Earth's surface (101,325 [Pa]). A combination of a couple of the compressible flow equations for ideal rockets were iterated on to find the plug nozzle termination area ratio ϵ_{pn}

$$\epsilon_{pn} = \frac{A_e}{A_{chnl}} \tag{3.19}$$

where this exit pressure would exist. First a guess was made for the exit area ratio ($\epsilon_{e,gss}$) which, along with other known engine conditions, was input to CEA. This analysis yielded the corresponding exit isobaric to isochoric specific heat ratio γ. Then the exit Mach number M_e was

$$M_{pn} = \sqrt{\left(\frac{P_c}{P_{pn}}\right)^{\frac{\gamma-1}{\gamma}} - 1 * \frac{2}{\gamma-1}}. \tag{3.20}$$

The corresponding exit area ratio ϵ_e was found by

$$\epsilon_{pn} = \left(\frac{1}{M_{pn}}\right) * \left(\frac{2 + (\gamma-1) * M_{pn}^2}{\gamma+1}\right)^{\frac{\gamma+1}{2(\gamma-1)}} \tag{3.21}$$

ϵ_{pn} was known once it matched to a tight tolerance with $\epsilon_{pn,gss}$. This approach yielded a termination diameter of approximately 6.5 [in]. This considerably shortens the plug nozzle which makes cooling easier, but also reduces the usable reaction area for thrust at higher altitudes. Since this RDE is being designed for upper stage use like RL10, a compromise termination diameter of 4 [in] was chosen.

Finally, for the RDE's outer wall after the combustion chamber, it was decided to use the RL10 diverging nozzle profile. This profile was shifted radially to correctly attach to the RDE's D_o which is larger than RL10's D_t, and cut off where the RDE and RL10 exit area ratios ϵ_e matched.

Given these design inputs, Figure 3.1 shows the RL10-comparable RDE axial profile.

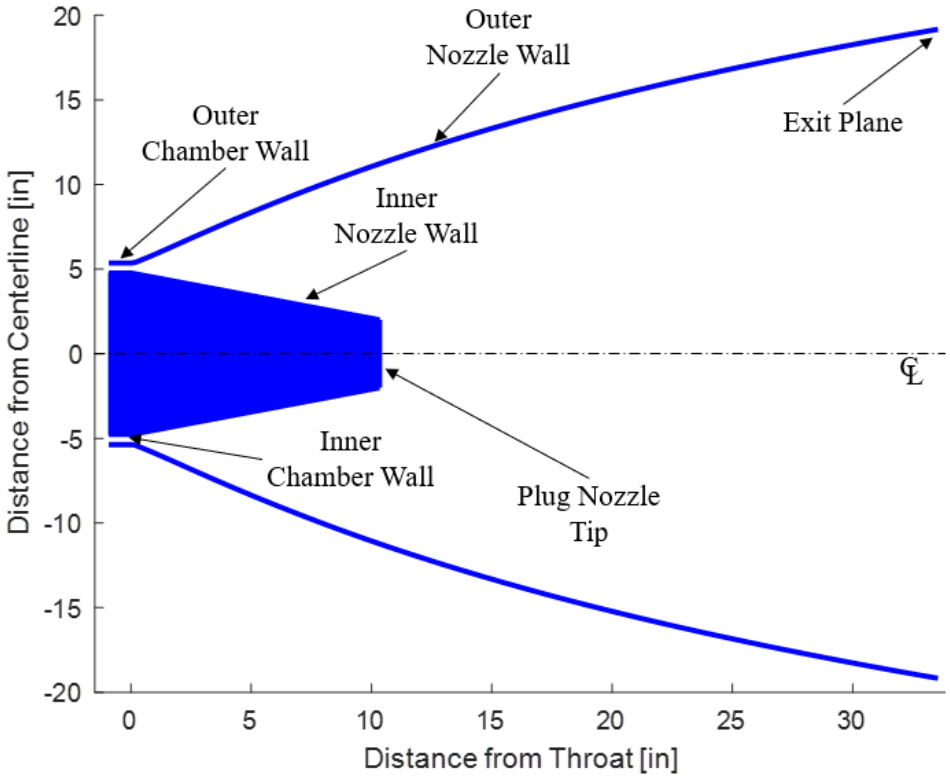


Figure 3.1 RL10-Comparable RDE Profile

Figure 3.2 shows a zoom on the annular combustion chamber and an equally long portion of the plug nozzle.

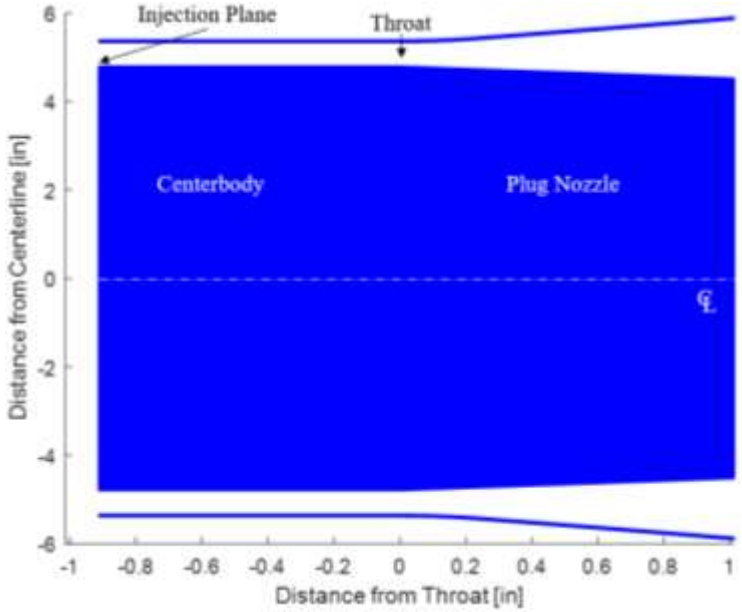


Figure 3.2 Zoom on RDE's Annular Chamber and Beginning of Plug Nozzle

Summary dimensions of RL10 and the comparable RDE are consolidated in Table 3.2 and Table 3.3.

Table 3.2 RL10 vs. RDE Geometry: Half-angle, Expansion Ratio, and Linear Dimensions

Property	**RL10**				**RDE**			
	Value	Units	Value	Units	Value	Units	Value	Units
Plug Nozzle Half-angle	N/A				15°			
Exit Expansion Ratio ε	61.122				61.254			
Detonation Channel Width w_d	N/A				0.0150	m	0.592	in
Detonation Channel Outer Circumference $c_{d,o}$	N/A				0.857	m	33.719	in
<u>Diameters</u>:								
Chamber, Outer D_o	0.273	m	10.733	in	0.273	m	10.733	in
Chamber, Hydraulic $D_{h,o}$	0.273	m	10.733	in	0.0301	m	1.183	in
Throat, Outer D_t	0.124	m	4.899	in	0.273	m	10.733	in
Throat, Hydraulic $D_{h,t}$	0.124	m	4.899	in	0.0301	m	1.183	in
Plug Nozzle Termination, Inner D_{pn}	N/A				0.102	m	4.010	in
Plug Nozzle Termination, Hydraulic $D_{h,pn}$	N/A				0.469	m	18.444	in
Exit, Outer D_e	0.973	m	38.301	in	0.974	m	38.342	in
Exit, Hydraulic $D_{h,e}$	0.973	m	38.301	in	0.974	m	38.342	in
<u>Lengths:</u>								
Characteristic Length L^*	0.953	m	37.499	in	0.0232	m	0.914	in
Combustion/Detonation Channel $L_{c,cyl}$ (only cylindrical portion for RL10)	0.0596	m	2.345	in	0.0232	m	0.914	in
Combustion/Detonation Channel $L_{c,conv}$ (up to and including throat for RL10)	0.308	m	12.143	in	0.0232	m	0.914	in
Plug Nozzle L_{pn}	N/A				0.263	m	10.338	in
Converging Nozzle L_{conv}	0.249	m	9.798	in	N/A			
Diverging Nozzle L_{div}	1.182	m	46.541	in	0.852	m	33.538	in
Total L_{tot}	1.491	m	58.683	in	0.875	m	34.452	in

Table 3.3 RL10 vs. RDE Geometry: Areas

Property	**RL10**				**RDE**			
	Value	Units	Value	Units	Value	Units	Value	Units
Cross-sectional Areas:								
Chamber A_c	0.0584	m^2	90.479	in^2	0.0122	m^2	18.850	in^2
Throat A_t (RL10) or A_{chnl} (RDE)	0.0122	m^2	18.850	in^2	0.0122	m^2	18.850	in^2
Exit A_e	0.743	m^2	1.152□10 3	in^2	0.745	m^2	1.155□10 3	in^2
Surface Areas:								
Chamber $SA_{c,cyl}$ (only cylindrical portion for RL10)	0.051	m^2	79.067	in^2	0.0376	m^2	58.225	in^2
Chamber $SA_{c,conv}$ (up to and including throat for RL10)	0.213	m^2	330.307	in^2	0.0376	m^2	58.225	in^2
Converging Nozzle	0.162	m^2	251.240	in^2	N/A			
Diverging Nozzle	2.483	m^2	3.848□10 3	in^2	1.949	m^2	3.021□10 3	in^2
Outer Cooling Circuit Heated, Short Tubes $SA_{clng,o,s}$	1.0168	m^2	1.576□10 3	in^2	0.650	m^2	1.008□10 3	in^2
Outer Cooling Circuit Heated, Long Tubes $SA_{clng,o,l}$	1.679	m^2	2.602□10 3	in^2	1.319	m^2	2.044□10 3	in^2
Outer Cooling Circuit Heated $SA_{clng,o}$	2.696	m^2	4.178□10 3	in^2	1.969	m^2	3.052□10 3	in^2
Inner Cooling Circuit Heated $SA_{clng,i}$	N/A				0.165	m^2	255.377	in^2
Total Heated SA_{tot}	2.696	m^2	4.178□10 3	in^2	2.134	m^2	3.307□10 3	in^2

RL10's L_c runs up to and includes the point where the throat is located. Exit area and therefore exit area ratio (Table 3.2) are slightly larger than RL10 simply due to the discretization granularity of the RDE's axial length. Surface area was incrementally summed using frustum surface area elements along the length of the engines except for the RDE chamber where cylinder surface area elements were used. During these summations, the short and long tube surface areas each received half of the incremental surface areas for the axial portion of the engine where they coexist along the nozzle.

For additional comparison, Table 3.4 provides key geometric ratios of interest including inner heated surface areas SA_c and SA_{tot}.

Table 3.4 Geometric Ratios of Interest

	$\frac{(D_{h,t})_{RDE}}{(D_{h,t})_{RL10}}$	$\frac{(L_c)_{RDE}}{(L_{c,cyl})_{RL10}}$	$\frac{(L_c)_{RDE}}{(L_{c,conv})_{RL10}}$	$\frac{(L_{div})_{RDE}}{(L_{div})_{RL10}}$	$\frac{(L_{tot})_{RDE}}{(L_{tot})_{RL10}}$	$\frac{(A_t)_{RDE}}{(A_t)_{RL10}}$
Ratio [%]	24.2	39.0	7.5	72.1	58.7	100
	$\frac{(SA_c)_{RDE}}{(SA_{c,cyl})_{RL10}}$	$\frac{(SA_c)_{RDE}}{(SA_{c,conv})_{RL10}}$	$\frac{(SA_{tot})_{RDE}}{(SA_{tot})_{RL10}}$	$\frac{(SA_{inr})_{RDE}}{(SA_{tot})_{RDE}}$	$\frac{(SA_{out})_{RDE}}{(SA_{tot})_{RDE}}$	$\frac{(SA_{inr})_{RDE}}{(SA_{out})_{RDE}}$
Ratio [%]	73.6	17.6	79.2	7.7	92.3	8.4

In terms of total cooled surface area, even though the RDE has a centerbody and plug nozzle, its shorter diverging nozzle leads to a total cooled surface about 80% that of RL10. The difference is more pronounced in the chamber where, if that chamber is defined from the head end to the throat, the RDE has only 18% the surface area to cool than does RL10.

Finally, Figure 3.3 shows a comparison of the two profiles by aligning the end of the RDE detonation channel with RL10's throat. Clearly the RDE saves length and volume, valuable rocket system-level commodities.

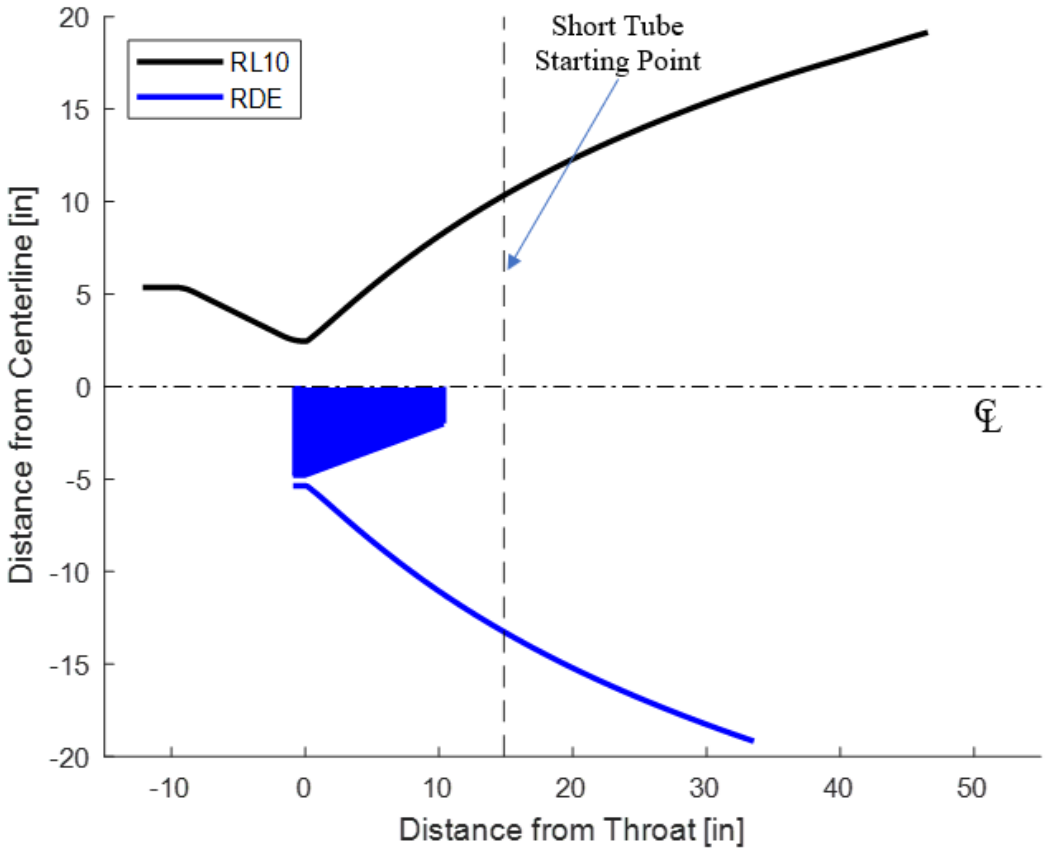
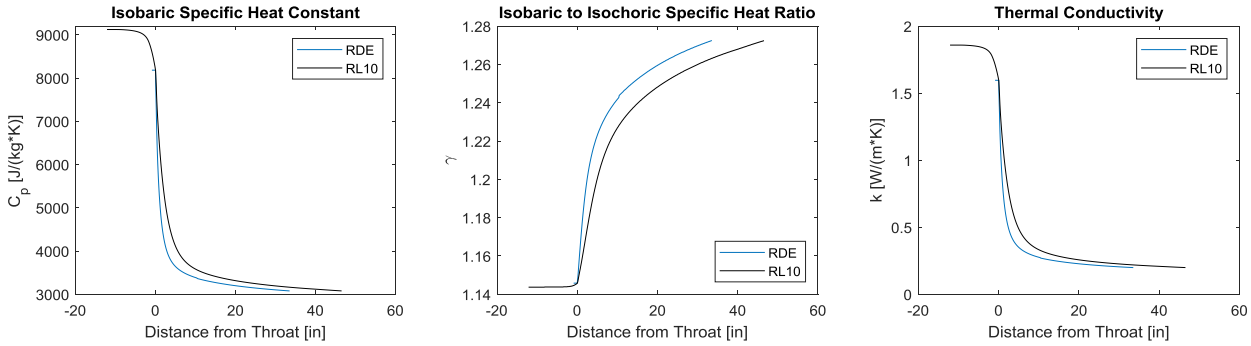


Figure 3.3 RL10 vs. RDE Axial Contour

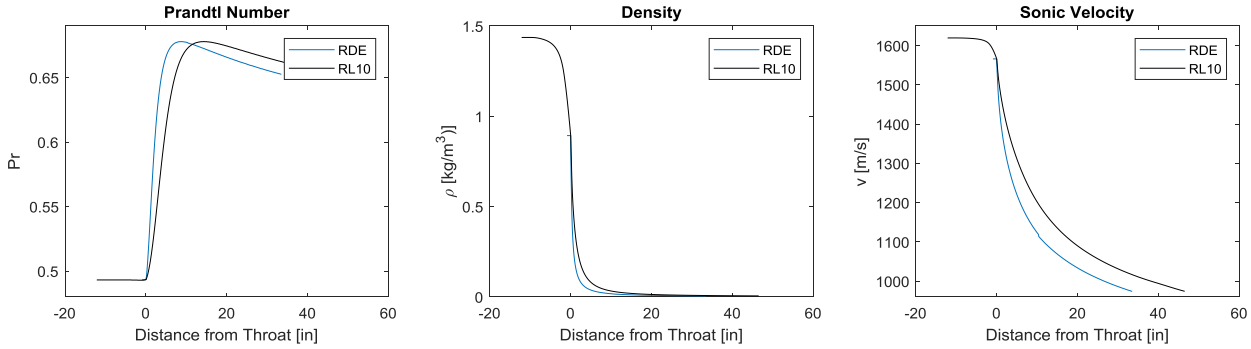
3.2 RDE Combustion

Combustion chemistry for the RL10-comparable RDE was analyzed using CEA in the same way as was performed for RL10. 0 justifies running the rocket equilibrium analysis instead of CEA's detonation mode and using properties corresponding to throat throughout the annular chamber. Essentially an ISSI/AFRL research study [7] showed that, if anything, the heat flux in the RDE chamber was lower than what was calculated using rocket equilibrium CEA and forcing throat-level heat flux everywhere [setting the inverse area ratio term in Bartz $\left(\frac{A_t}{A}\right)$ to 1] in the annular detonation channel. Figure 3.4 shows how these properties compare for the two engines.

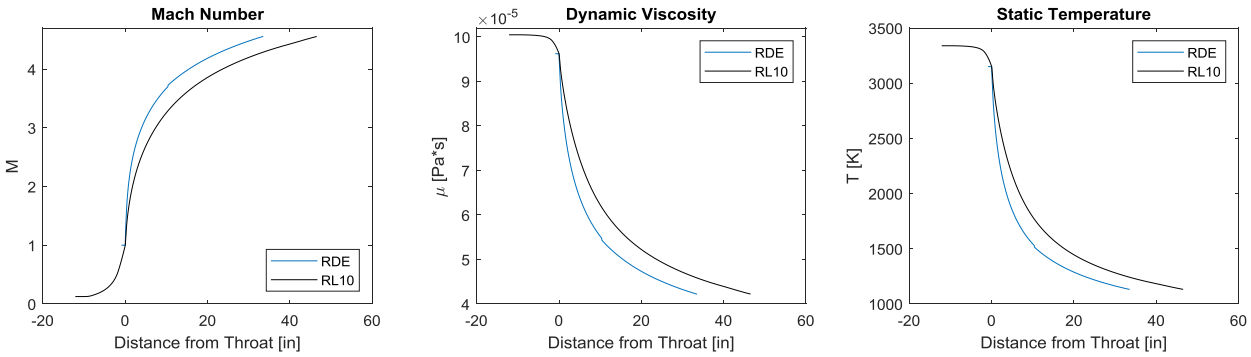
A)



B)



C)



D)

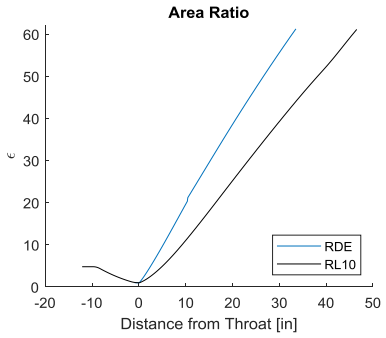


Figure 3.4 RDE vs. RL10 CEA Outputs Along Contour

Properties follow similar trends between the engines. The curves do not coincide because of the RDE's different geometry which leads to different area ratios along the centerline axis. The RDE's combustion characteristic velocity c^* was a constant 2,376.4 [m/s].

3.3 RDE Regenerative Cooling

To begin to evaluate the efficacy of regeneratively cooling the RL-10 comparable RDE, several of the cooling jacket properties were matched to RL10. This commonality helped to reduce modeling risk by making a fairer comparison. For the RDE, pressure drop and temperature rise of the hydrogen fuel coolant was not known a priori.

Table 3.5 RDE Cooling Characteristics Pre-Regenerative Cooling Analysis

Property	**Symbol**	**Value**	**Units**	**Value**	**Units**
Cooling Channels:					
Number, Short	N_{chnl}	180			
Start Distance Along Nozzle Relative to Throat, Short		0.377	m	14.846	in
Number, Long	N_{chnl}	180			
Coolant:					
Species		Supercritical Hydrogen			
Initial Temperature	T_{init}	31.667	K	57	°R
Initial Pressure	P_{init}	6.893□10^6	Pa	1000	psia
Chamber/Nozzle:					
Material		Stainless Steel 347			
Average Wall Thickness	th_w	3.302□10^{-4}	m	0.013	in
Thermal Conductivity	k_w	16.3	W/(m□K)	133	Btu/(hr□ft□°F)

The same iterative cooling analysis used for RL10 was performed for the RDE except for a few key differences. First, the annular RDE combustor requires two cooling circuits: one for the centerbody and plug nozzle and one for the outer chamber and diverging nozzle. The cooling load on the small surface area comprising the base of the plug nozzle (aft-facing side) was not considered in the analysis as this is a low-speed, separated flow region that would induce

reduced forced convection in a manner that is difficult to characterize. The overall heat load in this region is considered negligible due to its small surface area and low heat fluxes. Since there is a finite amount of fuel mass flow rate, it was split between the two circuits so that their outlet pressures would be equal. This would facilitate recombining the energized hydrogen in a common manifold before being fed into the turbine stage.

Table 3.6 RDE Cooling Circuit Hydrogen Fuel Mass Flow Rate Split

Circuit	**Mass Flow Rate**		**% of Total Fuel Mass Flow Rate**
	[lbm/s]	**[kg/s]**	
Outer	4.421	2.005	74.0
Inner	1.552	0.704	26.0

From Table 3.4, the inner circuit comprises only 7.7% of the total surface area to be cooled and yet requires 26% of the total coolant mass flow rate. This is partially due to $\Delta P \propto v_c^2$. Coolant velocity in the inner circuit is higher because the channels must become very small so that all 180 of them fit as the plug nozzle narrows toward its termination point. Additionally, a greater fraction of the inner surface area is subjected to the hotter region of the combustion product gases. An optimized inner cooling circuit would better cool the inner walls and/or require a lesser portion of the total coolant mass flow rate.

Both the inner and outer circuits' coolants begin at RL10's coolant's initial temperature and pressure of 57 [^{o}R] and 482 [psia] at the short tube start point and plug nozzle termination point respectively. The short tubes again run aft first, but the inner circuit coolant flows toward the head end only. The plug nozzle terminates 10.3 [in] from the throat but the short tubes do not enter the cooling jacket until 14.8 [in]. Also, the centerbody and plug nozzle have smaller circumferences than RL10's diverging nozzle circumference where the short tubes begin, so the short tubes would only fit into the RDE's outer circuit. Even without the short tubes, the long tube dimensions axially collocated on the centerbody and plug nozzle would likely still be too large. For a geometrically rational approach, the inner circuit was comprised of the long tube dimensions linearly scaled with the axially collocated ratio of the inner D_{in} to outer diameters D_o,

$$Inner\ Circuit\ Channel\ Dimension\ Scaling\ Factor = \frac{D_{in}^{i}}{D_{o}^{i}}. \quad (3.22)$$

Figure 3.5 shows the ratio of the long tube hydraulic diameter multiplied by the number of channels to the engine perimeter as a function of axial distance for the inner and outer circuits.

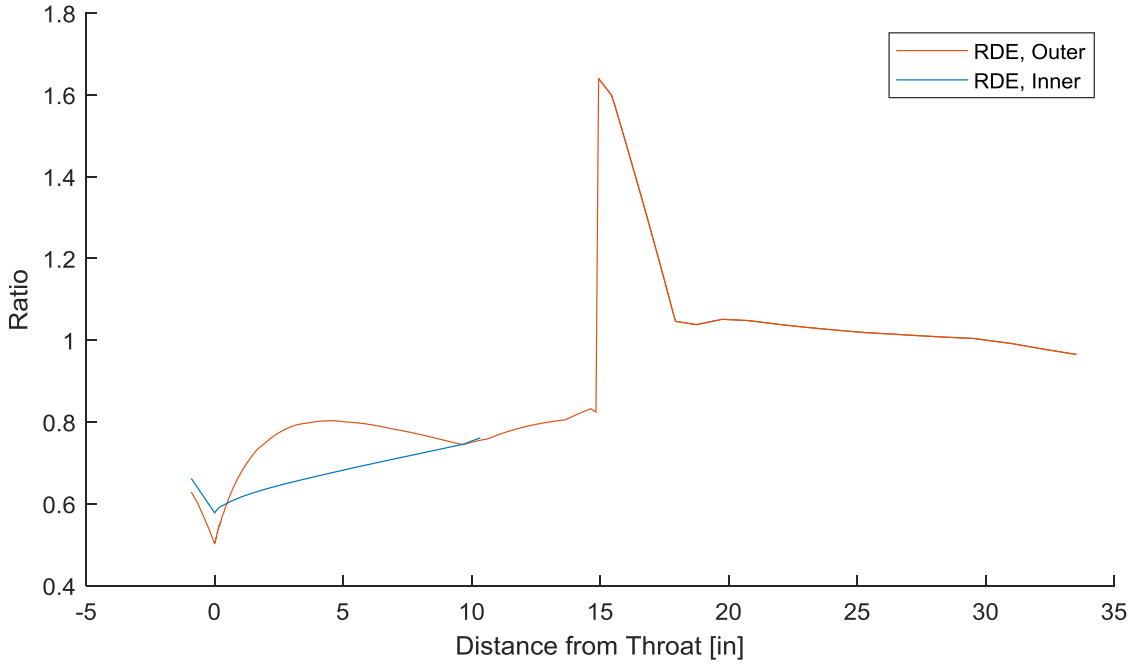


Figure 3.5 Ratio of Sum of Long Tube Hydraulic Diameters to Engine Perimeter

If the channel were circular, a ratio under unity would mean that the channels would all fit, neglecting the very thin (0.013 [in]) channel walls. Since the outer circuit's ratio in Figure 3.5 is greater than unity for most of the region where the short and long tubes coexist, and these are the actual dimensions known for the real RL10 engine, this must mean that the oval channels have a height-to-width ratio greater than 1.0. The inner circuit's scaling successfully keeps the ratio depicted in Figure 3.5 well under unity.

Despite analysis of the AFRL calorimetry studies showing that heat flux in the RDE chamber is less than a constant pressure engine's throat-level heat flux, the analysis was initially performed with no knockdown for greater conservatism. First, hot gas temperatures were compared in Figure 3.6.

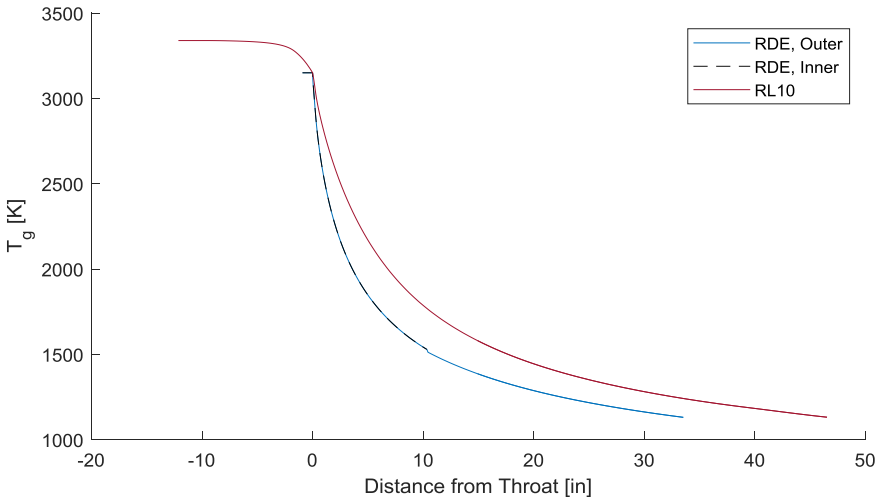


Figure 3.6 RDE vs. RL10: Hot Gas Temperature

Even though the inputs to CEA were the same for the RDE and RL10, except for their area ratios, the hot gas temperature for the RDE's chamber is slightly lower. In order to consistently model the RDE chamber as having throat-level heat transfer, CEA's constant pressure outputs corresponding to the throat were applied along the length of the RDE chamber. Throat-level temperature from CEA is less than the predicted constant pressure engine chamber stagnation temperature. Hence the RDE temperature extends horizontally to the left after x = 0 [in]; this point represents both the RDE throat and the aft most point of the annular chamber.

Adiabatic wall temperature was next compared. The RDE's inner and outer profiles

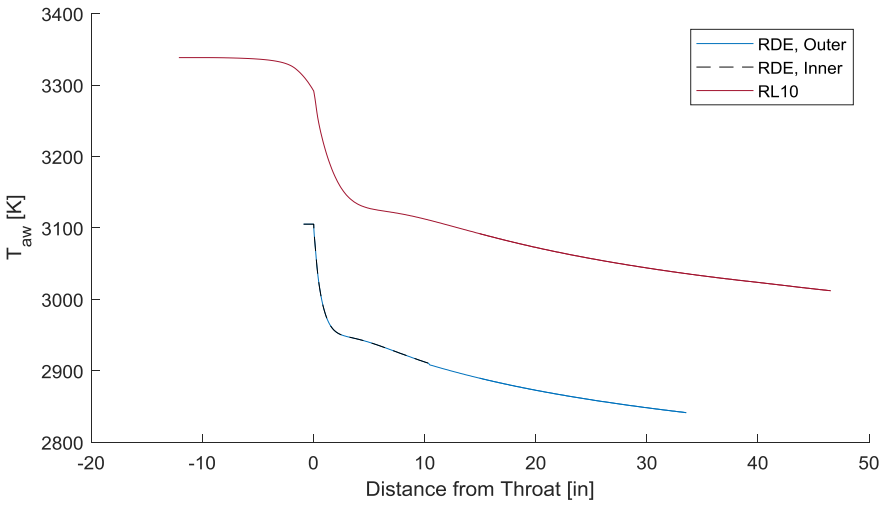


Figure 3.7 RDE vs. RL10: Adiabatic Wall Temperature

overlap because T_{aw} is not a function of any cooling circuit-specific properties.

Hot-gas-side wall temperature of RL10 and the RDE are compared in Figure 3.8.

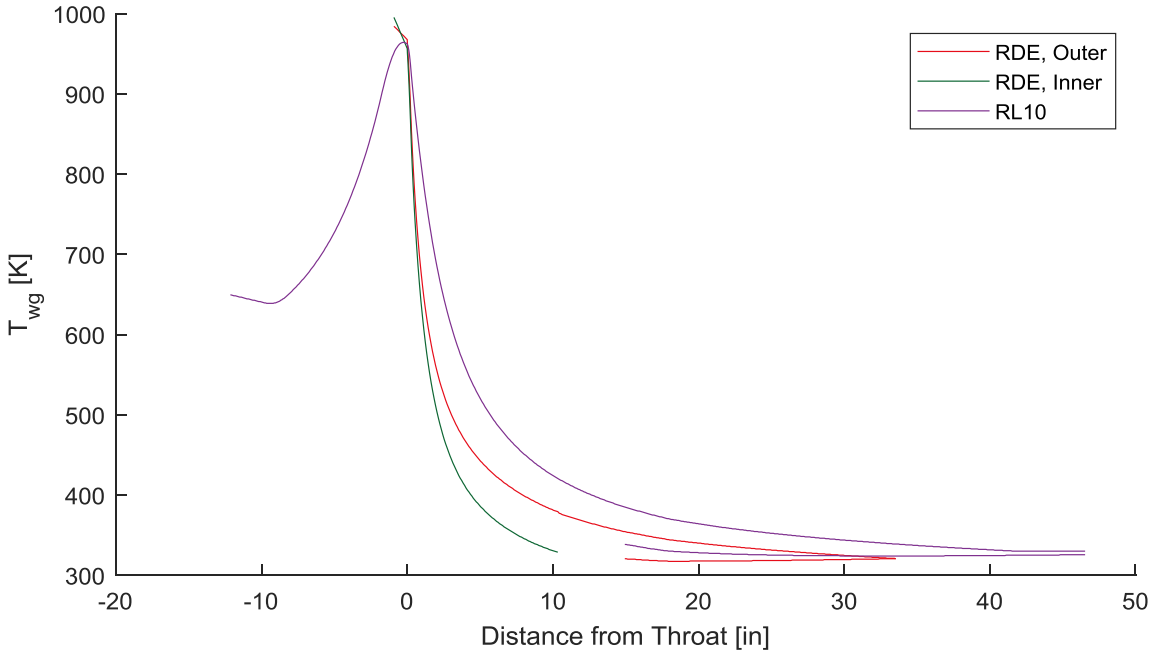


Figure 3.8 RDE vs. RL10: Hot-Gas-Side Wall Temperature

The RDE is cooler throughout the nozzle and only surpasses RL10's hot-gas-side wall temperature near and in the chamber. Inner and outer circuit hot-gas-side wall temperatures peak at 995.4 [K] and 984.3 [K] respectively; RL10's maximum T_{wg} is 964.4 [K]. The RDE's hotter T_{wg} (inner circuit) is 3.2% hotter than RL10's.

Hotter T_{wg} for the RDE is a result of the higher convective heat transfer coefficient h_g and the total coolant mass flow rate being split between the circuits.

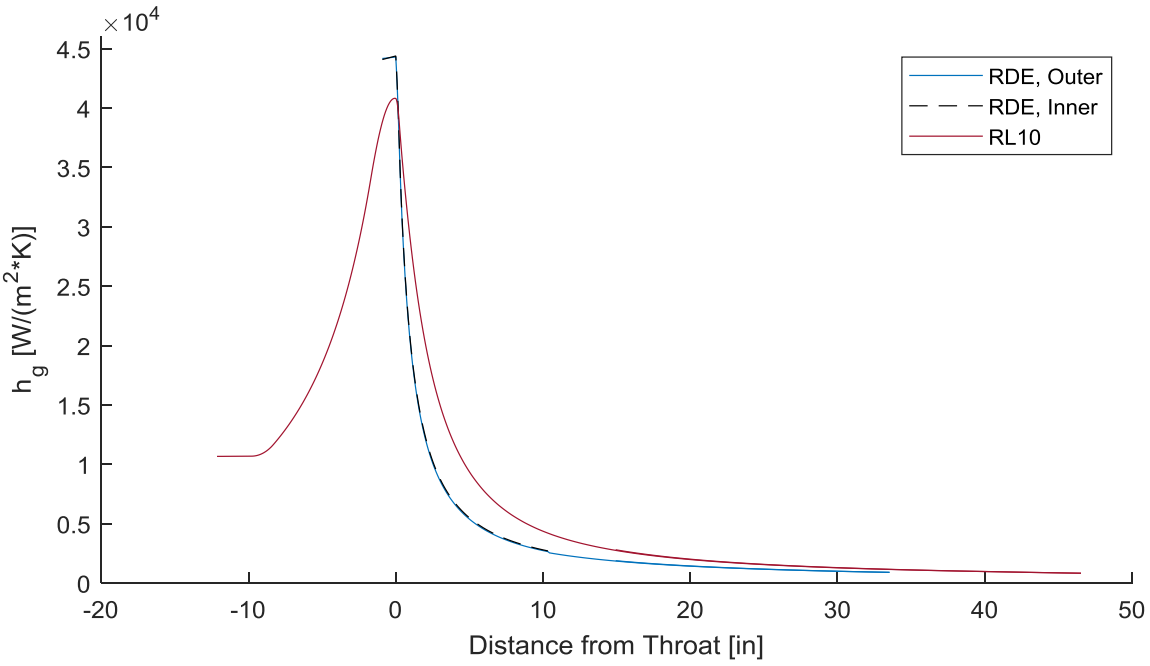


Figure 3.9 RDE vs. RL10: Hot Gas Convective Heat Transfer Coefficient

Hot gas heat transfer coefficient profiles for the inner and outer cooling circuits nearly coincide since Bartz is weakly affected by hot-gas-side wall temperature $\left(\left[\frac{1}{2}\frac{T_{wg}}{T_{g,c}}b+\frac{1}{2}\right]^{0.68}\right.$ term in denominator of boundary layer properties correction factor $\sigma\Big)$. The RDE has no throat curvature $(R = D_{h,t})$ and the whole chamber's heat flux was treated as throat-level ($A = A_t$ in the chamber). This leaves a $D_{h,t}^{0.2}$ in the denominator of the Bartz equation (Eq. (2.14) which increases h_g relative to RL10 since the RDE's hydraulic diameter is 24.2% the size of RL10's (Table 3.4). Ultimately the RDE has a maximum h_g of 44,389.6 [W/(m^2□K)] versus RL10's peak of 40,824.0 [W/(m^2□K)], approximately 10% higher. Per the boundary layer properties correction factor σ decreasing due to increasing gas and hot-gas-side wall temperatures in its denominator, h_g for the RDE decreases slightly after reaching its peak at the annular chamber's aft end.

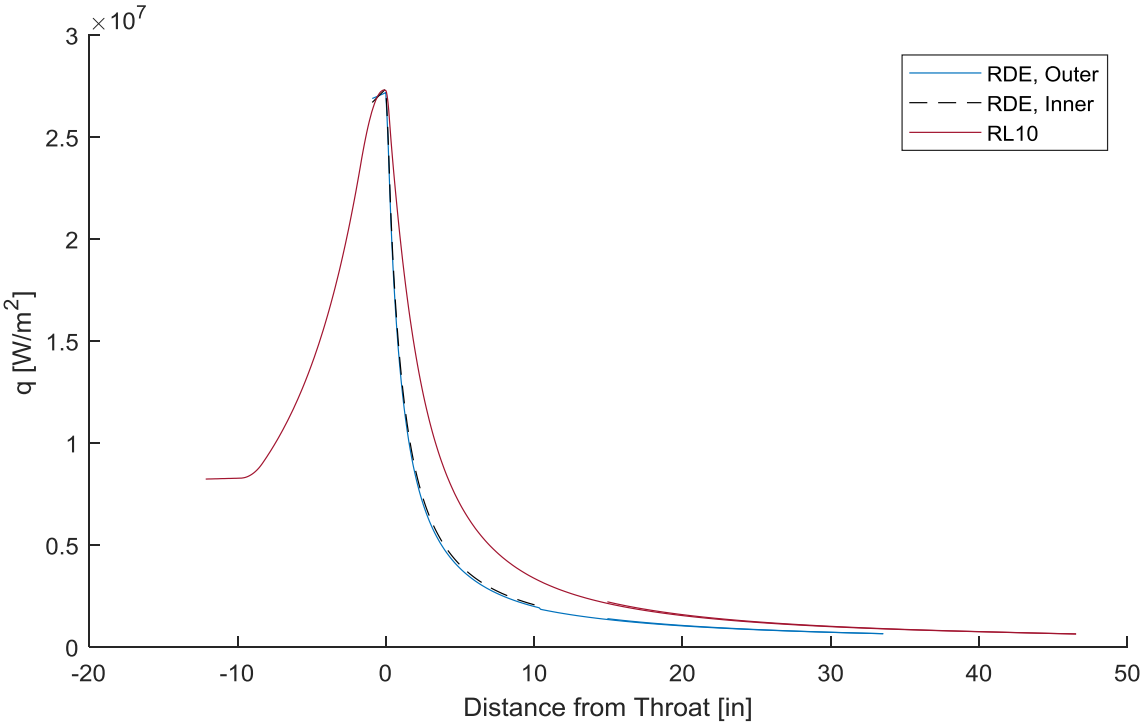


Figure 3.10 RDE vs. RL10: Heat Rate Flux

Heat rate flux $\dot{q}$ was next examined. Coincidentally, the engines' peak heat fluxes are nearly identical. The RDE's inner circuit sees that highest maximum heat rate flux of 27.4 [MW/m^2], 0.3% higher than RL10's maximum $\dot{q}$ of 27.3 [MW/m^2]. Figure 3.7 and Figure 3.8 show that adiabatic wall temperature is lower and the hot-gas-side wall temperature is higher, respectively, for the RDE versus RL10. Heat rate flux is the product of this change in temperature $(T_{aw} - T_{wg})$ with h_g. This change in temperature at the point of peak h_g in the RDE is 2,148.9 [K]; for RL10 it is 2,330.6 [K]. The RDE's ΔT happens to be 7.8% lower than RL10's which explains the chamber heat flux rates almost overlapping in Figure 3.10.

When transmitted through the cooling jacket wall, the heat transfer system yielded coolant-side wall temperatures T_{wc} shown in Figure 3.11.

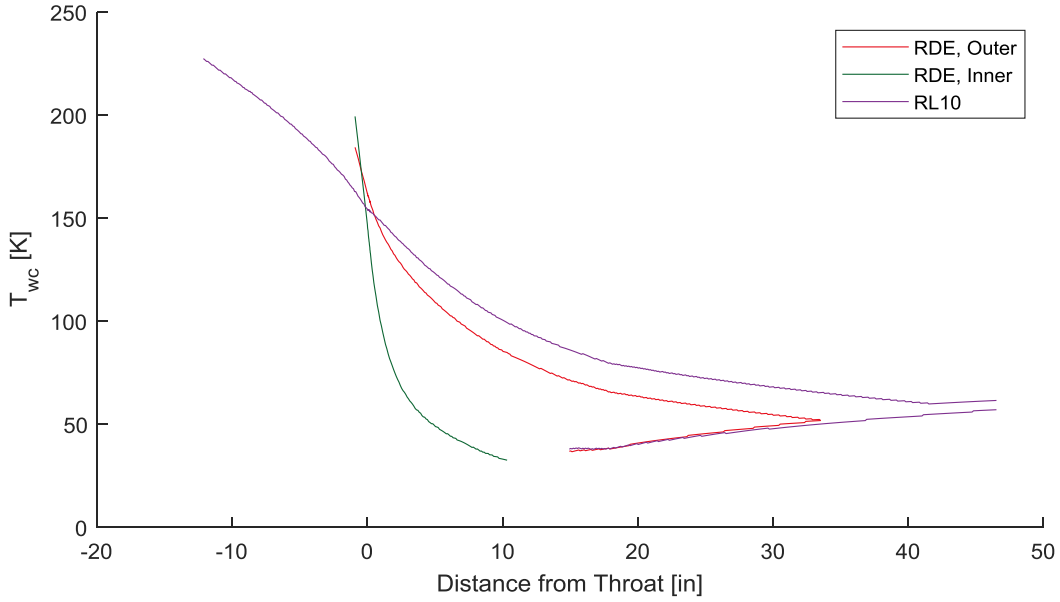


Figure 3.11 RDE vs. RL10: Coolant-Side Wall Temperature

Hydrogen again demonstrates its excellent cooling characteristics, keeping the coolant-side temperatures very low. The RDE's inner circuit has a maximum T_{wc} of 199.4 [K] and the outer circuit's maximum is 184.4 [K]. The hotter of these is approximately 12.3% cooler than RL10's maximum T_{wc} of 227.3 [K].

The coolant itself fared similarly as shown in Figure 3.12.

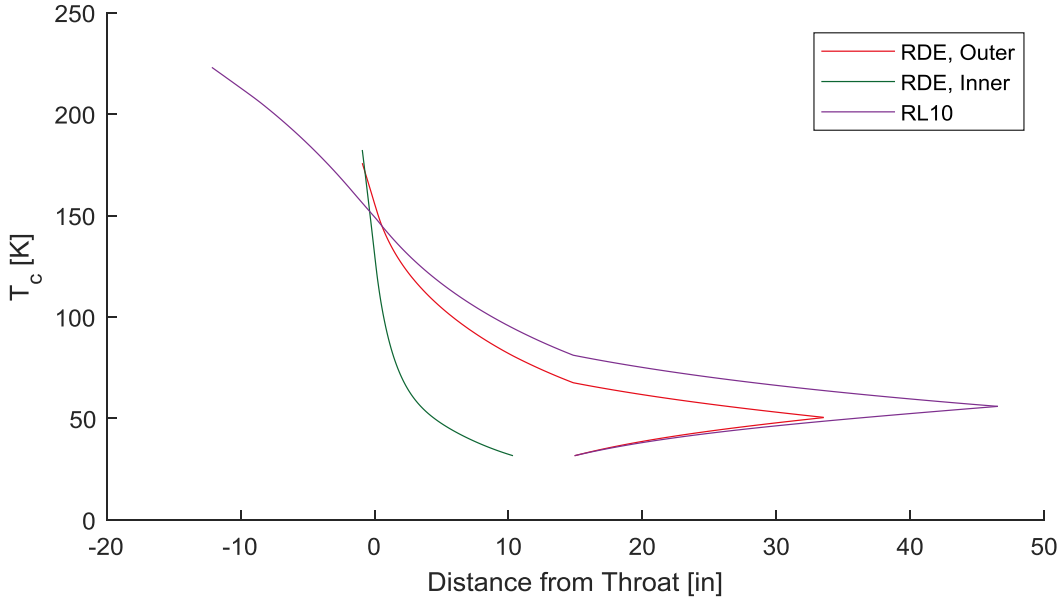


Figure 3.12 RDE vs. RL10: Coolant Temperature

Again, the RDE's inner circuit T_c is the hotter of the two with a maximum of 182.3 [K] compared to the outer's maximum at 175.8 [K]. This makes the maximum inner circuit T_c 18.2% cooler than RL10's maximum T_c of 223.0 [K]. Coolant always entered the jacket at 31.7 [K], and for RL10, both Binder [10] and this calibrated model predicted a temperature rise of 191.3 [K]. The RDE's cooling circuits both saw less of a temperature rise ΔT_{rse}. The outer circuit's ΔT_{rse} was 144.1 [K] and the inner circuit's ΔT_{rse} was 150.6 [K]. The latter is only 78.7% of RL10's ΔT_{rse}. In one sense this is positive in that the RDE coolant can absorb more heat before it reaches RL10's peak T_c, however the RDE's hot-gas-side wall temperatures are already slightly hotter. Additionally, lower coolant temperatures at the outlet of the cooling jacket mean that the coolant is less energized prior to entering the turbine. After the cooling jacket, the inner and outer circuit coolants would be mixed in a manifold prior to the turbine, and the mass flow rate-averaged mixed temperature

$$T_{c,mixed,outlet} = \frac{\dot{m}_{outer}}{\dot{m}_{tot}} * T_{c,outer,outlet} + \frac{\dot{m}_{inner}}{\dot{m}_{tot}} * T_{c,inner,outlet} \tag{3.23}$$

would be 177.5 [K] which is still 79.6% of RL10's cooling jacket outlet coolant temperature.

After splitting the total fuel mass flow rate between the circuits as listed in Table 3.6, the coolant pressure drops in the RDE's inner and outer circuits match to the 0.01 [psid]. Again, this was by design for easier subsequent mixing before the turbine. Their axial profiles are shown in Figure 3.13.

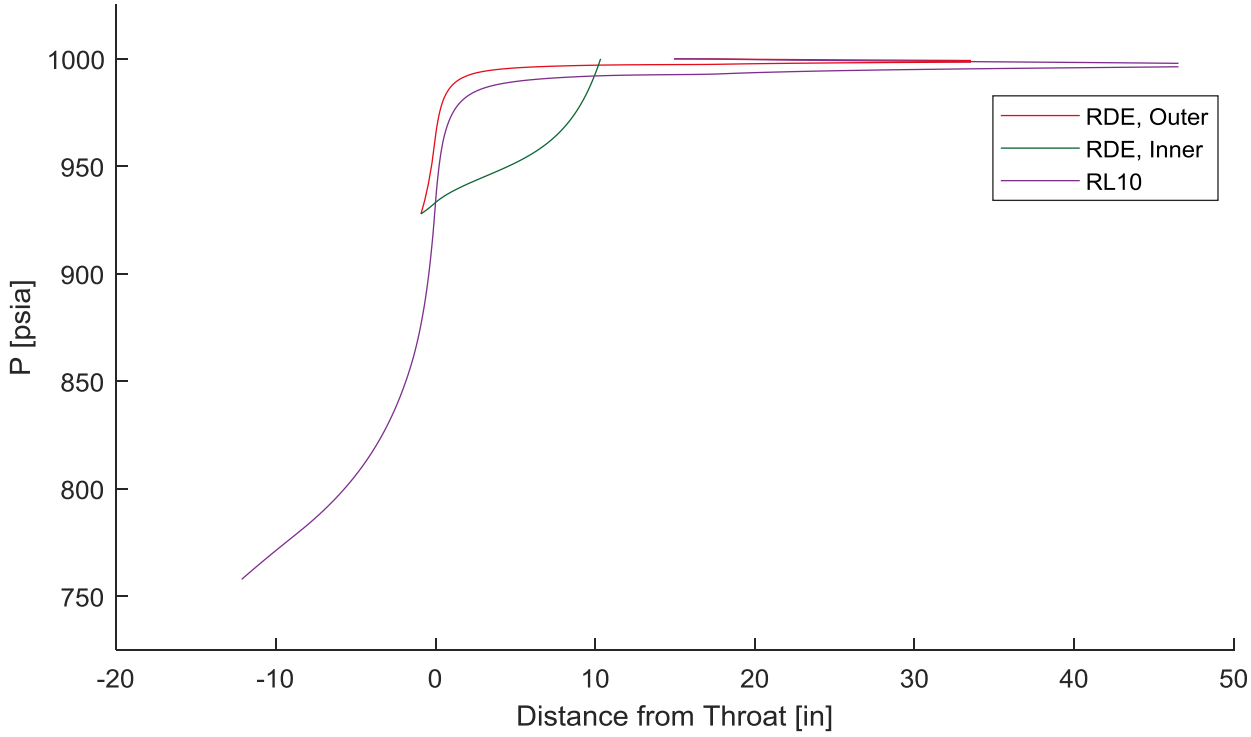


Figure 3.13 RDE vs. RL10: Coolant Pressure Profiles

Both RDE circuits started with 1000 [psia] of coolant pressure, lose 72.12 [psia], and so end at 927.9 [psia]. RL10 also started at 1000 [psia], loses 242.1 [psia], and so ends at 757.9 [psia]. The outer circuit receives most of the coolant mass flow rate (74.0%) which linearly increases coolant velocity (for fixed passage cross-sectional area and density, $v_c \propto \dot{m}$). The actual cooling jacket passages' cross-sectional areas are highly variable. Also, the inner circuit's passages had to be scaled down significantly to fit the narrowing structure. Since coolant velocity varies inversely with the square of channel hydraulic diameter, and pressure drop varies directly with the square of coolant velocity and inversely with hydraulic diameter, the smaller channels have a large impact ($\Delta P \propto 1/D_{h,c}^5$). Coolant channel dimensions for the RDE outer circuit were

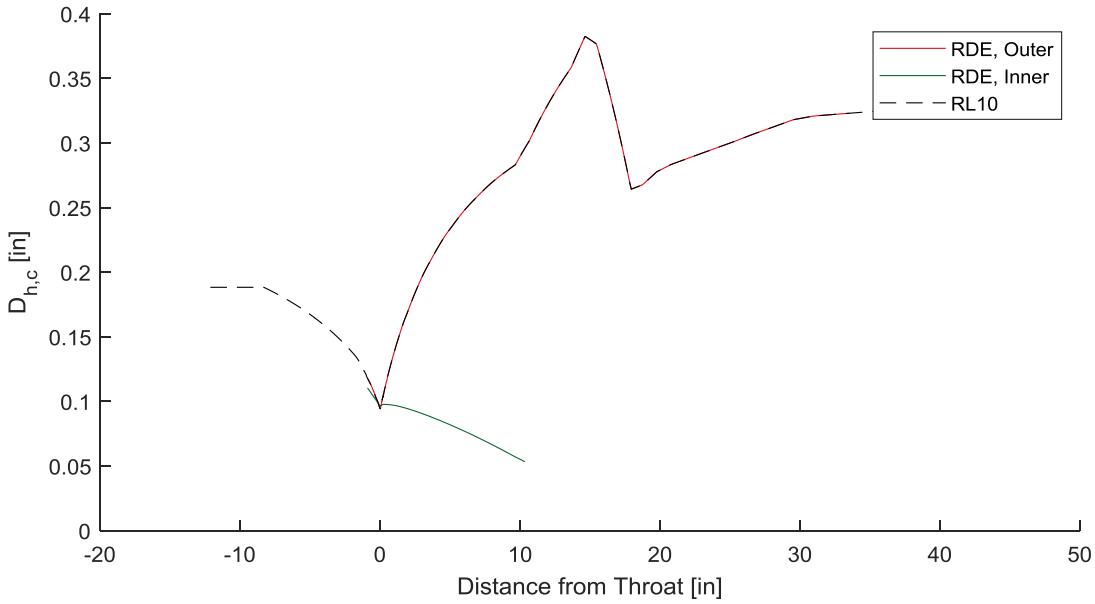


Figure 3.14 RDE vs. RL10: Coolant Channel Hydraulic Diameter

matched to RL10. The resulting hydraulic diameter profiles are shown in Figure 3.14. Even though the RDE uses RL10 cooling channel dimensions, its axially shorter nozzle means that the coolant experiences less cumulative wall friction.

With these dimensions in mind, coolant velocity is shown in Figure 3.15. The inner

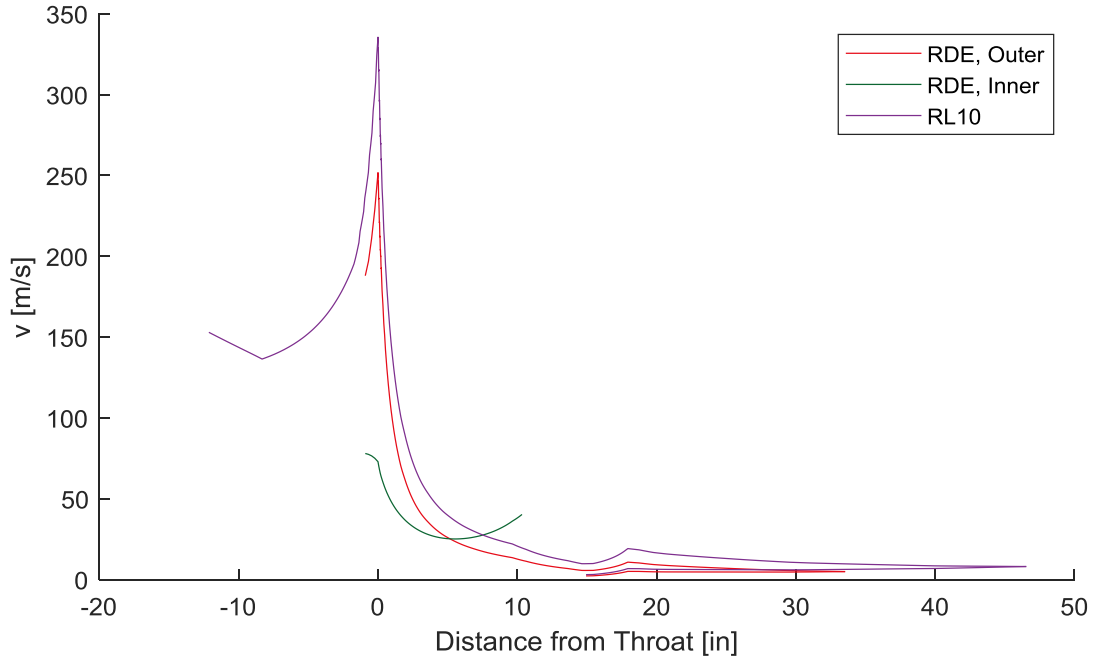


Figure 3.15 RDE vs. RL10: Coolant Velocity

circuit coolant spends almost all of its axial journey toward the head end in smaller channels and about half of that distance at higher velocities than the outer circuit coolant.

As designed and tuned, the cumulative heat rate $\dot{Q}$ for the RDE was compared to RL10 below in Figure 3.16. For the RDE, both inner and outer $\dot{Q}$ were summed. The lower final value

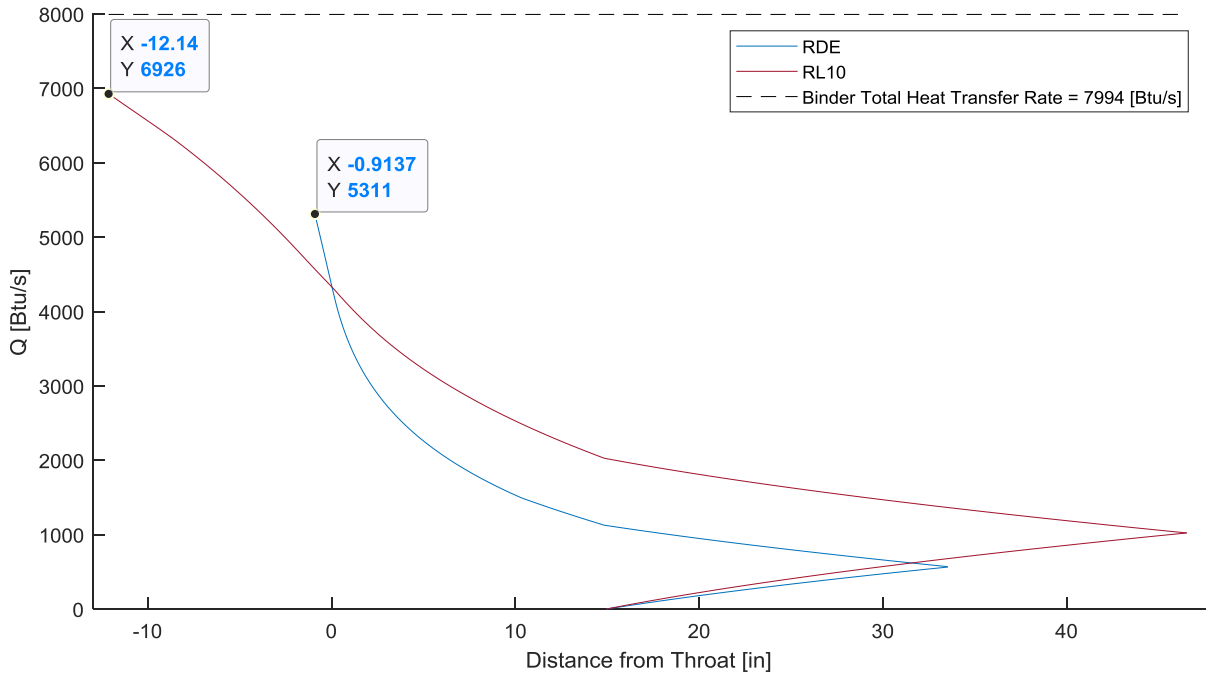


Figure 3.16 RDE vs. RL10: Cumulative Heat Rate

was foreshadowed by the lower coolant outlet pressures. It again reiterates that the RDE coolant egresses with less absorbed heat.

Table 3.7 summarizes the regenerative cooling analysis results.

Table 3.7 RDE Regenerative Cooling Analysis Results: Overall Coolant Changes

Property	**Symbol**	**Value**	**Units**	**Value**	**Units**
Coolant:					
Temperature Rise, Outer Circuit	$\Delta T_{rse,o}$	144.1	K	259.4	^{o}R
Temperature Rise, Inner Circuit	$\Delta T_{rse,i}$	150.6	K	271.1	^{o}R
Pressure Drop, Both Circuits	ΔP_{cl}	4.971□10^{5}	Pa	72.12	psid

The RDE coolant temperature only rises about 75% of RL10's coolant. Pressure drop in the RDE is only about 30% that of RL10.

Table 3.8 RDE Regenerative Cooling Analysis Results: Heat Transfer Rates

RDE Section	**Symbol**	**Value**	
		[MW]	**[Btu/s]**
Chamber, Inner Wall	$\dot{Q}_{c,i}$	0.480	454.676
Chamber, Outer Wall	$\dot{Q}_{c,o}$	0.538	510.094
Chamber	$\dot{Q}_{c}$	1.018	964.770
Plug Nozzle	$\dot{Q}_{pn}$	1.022	968.945
Diverging Nozzle	$\dot{Q}_{dn}$	3.563	3.377□10^{3}
Plug and Diverging Nozzle	$\dot{Q}_{pdn}$	4.585	4.346□10^{3}
Inner Wall	$\dot{Q}_{,i}$	1.502	1.424□10^{3}
Outer Wall	$\dot{Q}_{,o}$	4.101	3.887□10^{3}
Total	$\dot{Q}_{tot}$	5.603	5.311□10^{3}

As shown in Table 3.8, cumulative heat rate increases sharply in the chamber but is generally dominated by the outer wall heat loads.

Table 3.9 RDE Heat Transfer Rate Ratios of Interest

	$\frac{\dot{Q}_{c,i}}{\dot{Q}_{c,o}}$	$\frac{\dot{Q}_{,i}}{\dot{Q}_{,o}}$	$\frac{\dot{Q}_{c}}{\dot{Q}_{tot}}$	$\frac{\dot{Q}_{,i}}{\dot{Q}_{tot}}$	$\frac{\dot{Q}_{,o}}{\dot{Q}_{tot}}$
[%]	89.1	36.6	18.2	26.8	73.2
	$\frac{\dot{Q}_{c,i}}{L_c}$	$\frac{\dot{Q}_{c,o}}{L_c}$	$\frac{\dot{Q}_{c}}{L_c}$	$\frac{\dot{Q}_{pn}}{L_{pn}}$	$\frac{\dot{Q}_{dn}}{L_{dn}}$
[kW/mm]	20.669	23.188	43.857	3.893	4.183

Table 3.9 makes it is clear why 74% of the total coolant mass flow rate had to be diverted to the outer circuit given that the surface it cools sees 73.2% of the total heat load; likewise with the inner circuit's 26% coolant mass flow rate being required to cool its surface's 26.8% share of the total heat rate. "Outer Wall" refers to both the diverging nozzle and outer wall of the annular chamber whereas "Inner Wall" refers to both the plug nozzle and the inner wall of the chamber. Heat transfer loads spike in the chamber where the product gases are hottest.

Primary comparisons with RL10 are shown below in Table 3.10. The RDE experiences

Table 3.10 RDE Regenerative Cooling Analysis Result Ratios with RL10

	Max h_g	**Max** $\dot{q}$	**Max** T_{wg}	**Max** T_{wc}	**Max** T_c	**Outlet** T_c
RDE-to-RL10 Ratio [%]	108.7	100.3	103.2	87.7	81.8	79.6
	$\Delta T_{rse,o}$	$\Delta T_{rse,i}$	ΔP_{cl}	$\dot{Q}_{tot}$		
RDE-to-RL10 Ratio [%]	75.3	78.7	29.8	76.7		

higher peak heating in the chamber but lower overall coolant temperature rise. Since the coolant mass flow rate is split and the channels are shorter, there is less pressure drop. The net result is that the RDE is subjected to about approximately 75% of the total heat rate of RL10.

More supporting comparisons between the RDE and RL10 are included in 0.

3.4 Turbine Power Comparison

To properly evaluate the impact on the expander cycle's performance, the turbine powers were compared. Turbine power N is a function of mass flow rate $\dot{m}$, isobaric specific heat C_p, stagnation temperatures T_t (and optionally stagnation pressures P_t), and isobaric to isochoric specific heat ratio γ,

$$N = \dot{m}C_p\left(T_{t,1} - T_{t,2}\right) = \dot{m}C_pT_{t,1}\left(1 - \frac{T_{t,2}}{T_{t,1}}\right) = \dot{m}C_pT_{t,1}\left[1 - \left(\frac{P_{t,2}}{P_{t,1}}\right)^{\frac{\gamma-1}{\gamma}}\right]. \quad (3.24)$$

Subscripts "1" and "2" refer to inlet and outlet conditions respectively. Turbine efficiency is assumed to be 100% since this analysis is focused on finding an RDE-to-RL10 turbine power ratio. Any other efficiency would be assumed the same for both cases and would cancel out in the ratio calculation. Stagnation temperature is a function of static temperature T, γ, and Mach number M,

$$T_t = T\left(1 + \frac{\gamma-1}{2}M^2\right). \quad (3.25)$$

Mach number is the ratio of flow velocity to the local speed of sound a,

$$M = \frac{v}{a} \quad (3.26)$$

The local speed of sound a can be calculated by

$$a = \sqrt{\gamma R_{sp} T} \tag{3.27}$$

where is the specific gas constant R_{sp}, the result of dividing the universal gas constant R_u by the gas molecular weight ɱ (2.01588 [g/mol] for H_2),

$$R_{sp} = \frac{R_u}{ɱ} \tag{3.28}$$

and is equal to 4,124.253 [J/(kg□K)]. Stagnation pressure is a function of static pressure P, γ, and M.

$$P_t = P\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma - 1}} \tag{3.29}$$

Isobaric to isochoric specific heat ratio γ and C_p were found from REFPROP using the input pressures and temperatures specific to RL10 and the RDE. The goal of this comparison is to see how the RDE cooling jacket's different outlet conditions might affect turbine power. For the sake of comparison and since detailed dimensions of the manifold leading into the turbine were not known, the inlet velocity was kept the same for both engines. Finally, Binder [10] lists the turbine pressure ratio TPR as 1.39,

$$TPR = \frac{P_{t,1}}{P_{t,2}} \tag{3.30}$$

Table 3.11 summarizes the inputs for the turbine power comparison.

Table 3.11 Turbine Power Comparison Analysis Inputs

Engine	$\dot{m}$ **[kg/s]**	C_p **[J/(kg□K)]**	$v_{c,mix}$ **[m/s]**	γ	R_{sp} **[J/(kg□K)]**	M
RL10	2.709	$1.407\square10^4$	159.359	1.444	$4.124\square10^3$	0.138
RDE	2.709	$1.367\square10^4$	159.359	1.444	$4.124\square10^3$	0.152

Engine	T_1 **[K]**	$T_{t,1}$ **[K]**	P_1 **[Pa]**	$P_{t,1}$ **[Pa]**	$P_{t,2}$ **[Pa]**	
					Method 1	**Method 2**
RL10	223	223.946	$5.224\square10^6$	$5.297\square10^6$	$3.810\square10^6$	N/A
RDE	177.484	178.506	$6.396\square10^6$	$6.507\square10^6$	$4.682\square10^6$	$3.810\square10^6$

Turbine power was compared between the RDE and RL10 in Table 3.12.

Table 3.12 Turbine Power Comparison Results

Engine	N **[MW]**		**% RL10**	
	Method 1	**Method 2**	**Method 1**	**Method 2**
RL10	16.251	N/A	100	N/A
RDE	12.542	12.151	77.2	74.8

Method 1 in Table 3.11 and Table 3.12 refers to assuming the turbine will be maintain the same TPR which therefore causes the turbine outlet stagnation pressure for the RDE to be higher than RL10. Method 2 considers that the RDE turbine would output the same stagnation pressure as RL10. Either way, since the RDE coolant is energized less by the regenerative cooling jacket, it will likely yield a turbine power output approximately three quarters that of RL10. If this were intolerable, the cooling jacket would have to be redesigned to absorb more heat or the turbine redesigned to perform like RL10 with the RDE's lower inlet stagnation enthalpy.

4. CONCLUSION

An anchored, one-dimensional, steady state, variable fluid property regenerative cooling model for RL10 and a comparable RDE has been created in MATLAB. The model's RL10 predictions are acceptably close to the more detailed SPTD model [10]. Geometry for the RDE reflects reasonable decisions made toward emulating RL10's upper stage performance while respecting unique considerations like detonation chamber fill height. Dual, independent coolant circuits are employed to cool both the outer and inner walls of the RDE. Throat-level heat flux is presumed to blanket the entire chamber. While this is conservative according to analysis of airbreathing RDE heat flux measurements, rocket engine data are lacking.

The first obvious benefit of the RL10-comparable RDE is its reduced size. Reduced length and volume likely means less engine mass (depending on the size of the centerbody and if it is hollow) that must be lifted by the first stage and increased payload mass fraction. Detonation combustion requires a chamber only as long as the propellant fill height. Even with this RDE's factor of safety on fill height to account for variability from detonation's unsteadiness, the annular combustion chamber is only 39.0% the length of RL10's cylindrical chamber and 7.5% the length from RL10's injector face to its throat. Since it would be desired for the RDE to perform in the same environments as RL10, their exit areas were matched. Although the RDE used the same diverging nozzle profile as RL10, it was shifted radially to start at the annular chamber exit diameter which was larger than RL10's throat diameter. This added diameter caused the RDE's nozzle exit area ratio to reach the RL10's exit area ratio at a shorter axial length, yielding a diverging nozzle length and total length 72.1% and 58.7%, respectively, those of RL10. Despite having the inner walls of the centerbody and plug nozzle, the RDE's total cooled surface area is only 79.2% of RL10's. Considering the RL10's chamber to be just the cylindrical section, the entire RDE annular chamber has only 73.6% the surface area; if the chamber is defined as extending to the throat, the RDE surface area is only 17.6% of RL10's.

At steady state, the RDE experiences higher peak hot-gas convective heat transfer and heat flux at the throat; however, its lesser cooled surface area overcomes this intense region and leads to less cumulative heat rate through the walls and therefore less energy into the coolant. To achieve matching coolant circuit outlet pressures, 26.0% of the total coolant mass flow rate had to be diverted to the inner circuit despite it only comprising 8.4% of the total surface area. Mass

flow rate is more closely tied to cumulative heat rate. The inner circuit sees 26.8% of the total heat RDE heat load while the outer circuit receives the rest. In this design, the two circuits have comparable variable isobaric specific heats constant and net changes in temperature such that the remaining term in the sensible thermal energy storage equation, mass flow rate, is most predictive $(\dot{Q} \propto \dot{m})$. While the inner circuit is only 32.7% the axial length of the outer circuit (which is also the total axial length), 8.1% of it is comprised of the chamber compared to only 2.7% for the outer circuit. Outside the chamber cumulative heat load per unit length is relatively low: 3.9 [kW/mm] for the plug nozzle and 4.2 [kW/mm] for the diverging nozzle. In the chamber this spikes to an average of 21.9 [kW/mm] along either the outer or inner annular walls. The hotter products in this region drive large heat flux through the walls and into the coolant. When taken together the inner and outer walls of the RDE receive 5.6 [MW], a heat load 76.7% that of RL10, and enough to power approximately 5,600 homes [25]. Despite this enormous heat load, the RDE's hot-gas-side wall is hotter than RL10 only in the region of the shorter annular chamber, and at most by 3.2%. The coolant-side walls are cooler than RL10 at every corresponding axial station.

Reduced wall temperature bodes well for the RDE structure surviving, but it presents the challenge of less energized coolant. The expander cycle relies on the coolant absorbing a certain amount of energy from the hot surfaces it flows over, and then converting that to work on the turbine which powers the both the fuel and oxidizer pumps. If the RDE were integrated the RL10's turbine, similar to the total cumulative heat load ratio, the RDE coolant would allow the turbine to produce about 76.7% of the nominal power. The turbopump is a finely tuned device that is crucial for creating the correcting pressure heads for the two propellants. In order to increase turbine output power, the cooling jacket could be redesigned to have longer passages, rather than axial contours, to absorb more heat energy. Also, reducing the number of channels that comprise the jacket or changing coolant mass flow rate would reduce coolant velocity and increase jacket residence time for absorbing heat. Alternatively, the turbopump and upstream components could be redesigned to accept the lower enthalpy pre-turbine flow, but this is likely the more difficult option.

ISSI/AFRL research [7] indicates that average chamber heat flux is likely less than a comparable constant pressure engine's throat-level. When applying the most conservative estimate of 43% of throat-level, the RDE enjoys lower wall temperatures but the coolant also

absorbs less energy, accumulating 71.6% of RL10's total heat rate. This would make turbine integration challenge slightly harder.

The RDE's cooling jacket was based of RL10's dimensions. A full optimization specific to the RDE and its integration requirements would produce better cooling for the TCA and pre-turbine coolant energization. Computational fluid dynamics simulations on the RDE's nozzle to find the exact flow separation point may justify further shortening. Material-wise, the model could be updated to use variable metal thermal conductivity and the effect of the silver throat insert (at least in RL10). The engine operating space (e.g. mass flow rate, chamber pressure, mixture ratio, etc.) and its effects on heat transfer could be further explored. Model fidelity could be increased by increasing the dimensionality to include two or three-dimensional flow effects. Thermal transients at engine startup/shutdown, and from differences in the detonation wave front and its surroundings would also improve the model's physics.

APPENDIX A. RDE WITH THROAT-LEVEL HEAT FLUX SUPPLEMENTAL COMPARISONS

The model makes more detailed comparisons than were featured in Chapter 3. They are provided here for the interested reader.

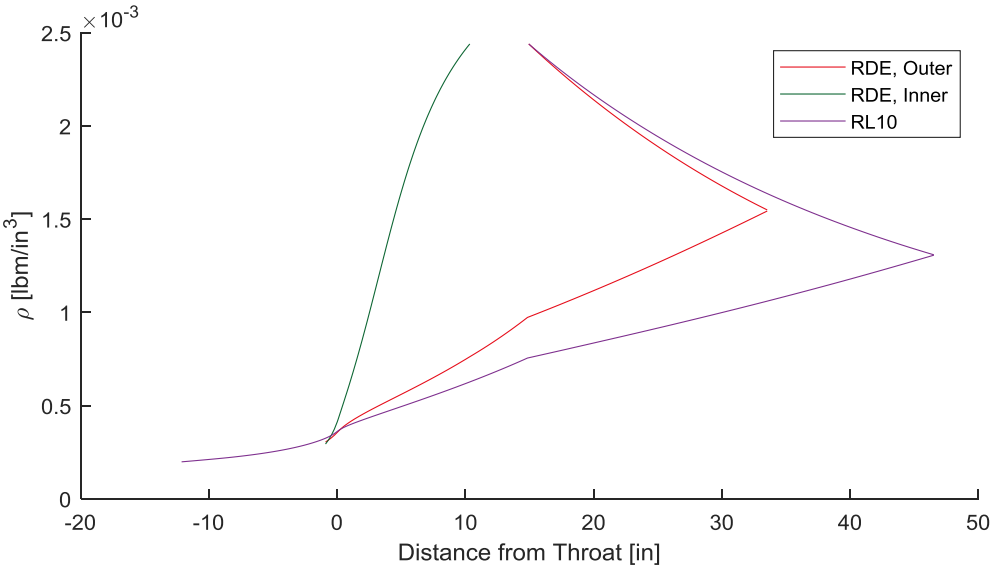


Figure **Error! No text of specified style in document.**.1 RDE vs. RL10: Coolant Density

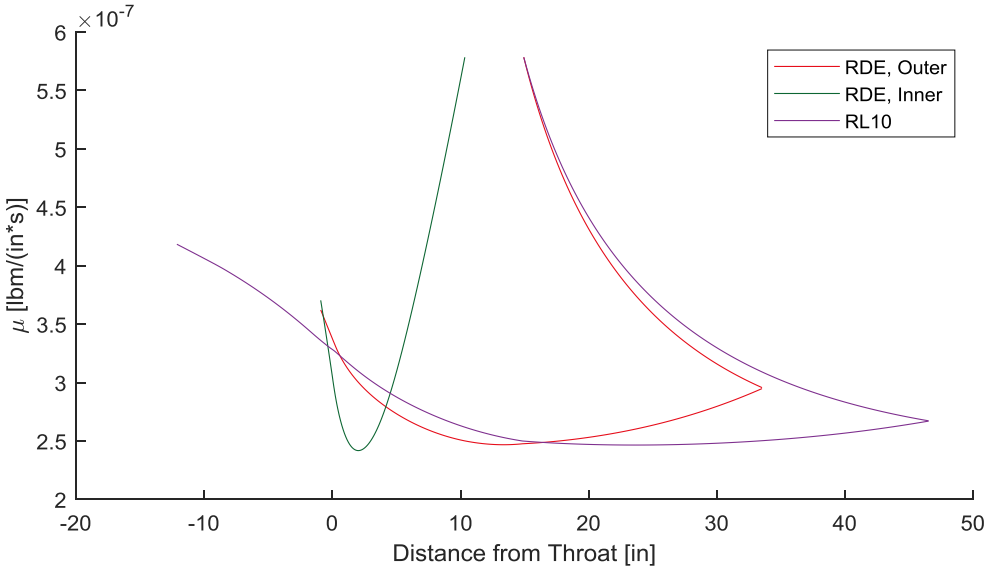


Figure **Error! No text of specified style in document.**.2 RDE vs. RL10: Coolant Dynamic Viscosity

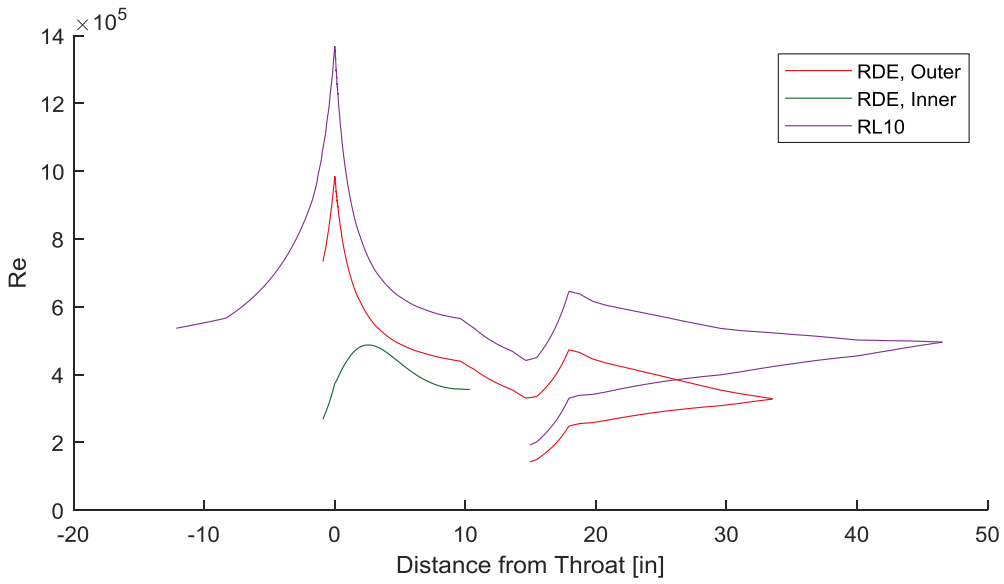


Figure **Error! No text of specified style in document.**.3 RDE vs. RL10: Coolant Reynolds Number

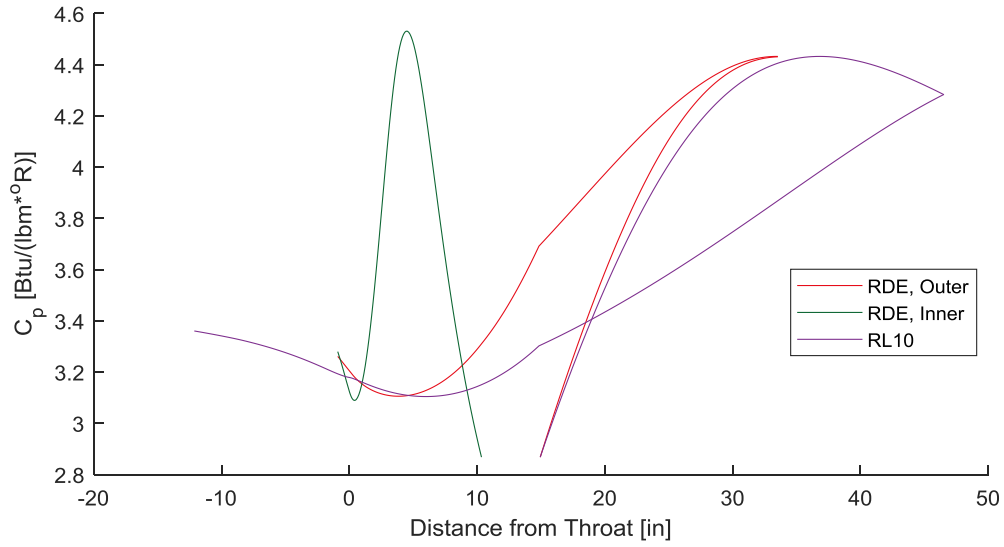
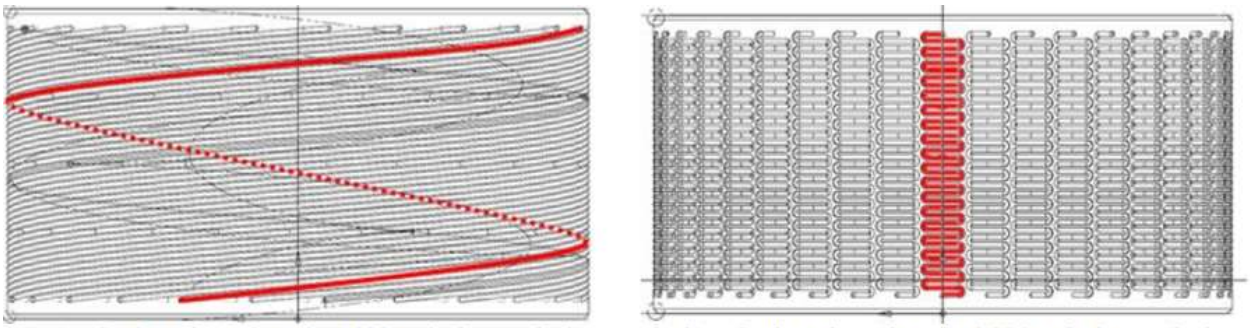


Figure **Error! No text of specified style in document.**.4 RDE vs. RL10: Coolant Isobaric Specific Heat Constant

APPENDIX B. CHAMBER HEAT FLUX KNOCKDOWN STUDY

Stevens et al. of ISSI Inc. and Schauer of the Air Force Research Laboratory (AFRL) collaborated on building and testing a water-cooled, airbreathing, hydrogen-fueled, rotating detonation engine in order to experimentally measure heat transfer properties [7]. Their methods were calorimetry and videography. They found that the hot-gas convection coefficient, critical to heat flux through the walls, is primarily affected by wave frequency, and secondarily affected by specific heat addition and mass flux. Large changes in wave frequency were primarily due to different numbers of waves establishing themselves in the detonation chamber. Equivalence ratio and mass flux were generalized to specific heat addition and mass flux, respectively, to extend the applicability of their findings.

Certain parameters of the RDE could be varied including wall thickness, cooling passage design, nozzle area ratio, reactant mass flow rate, and equivalence ratio. Examples of their cooling jacket designs are shown below in Figure **Error! No text of specified style in document.**.5.



a) Cooling channels in the 0.193 in thick outer-body

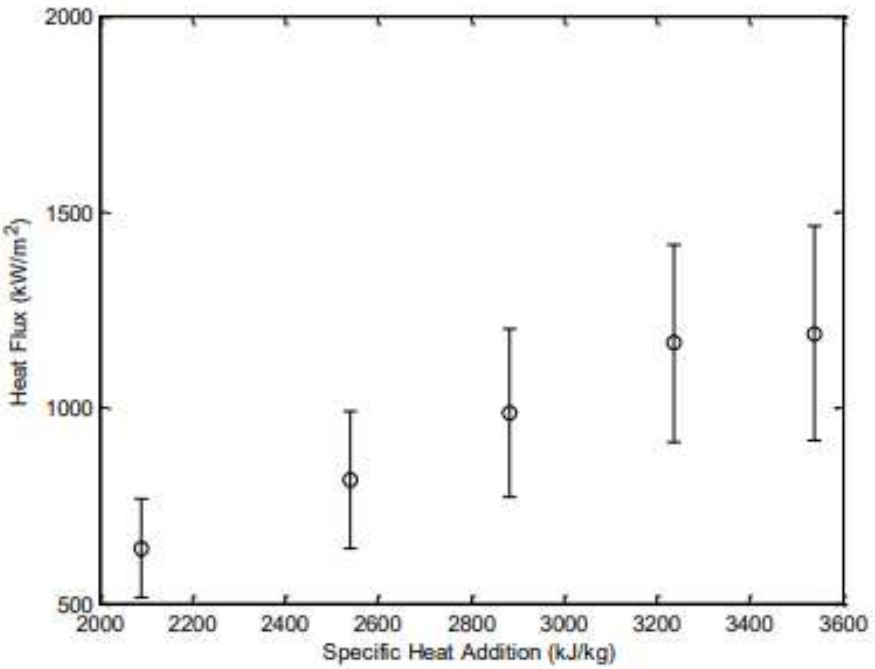
b) Cooling channels in the 0.256 in thick outer-body

Figure 5. Cooling channel geometries.

Figure **Error! No text of specified style in document.**.5 Cooling Channel Geometries from Stevens [7]

Inlet and outlet cooling water temperatures could be used to calculate the total heat rate using the sensible thermal energy storage equation (Eq. (2.30). The authors worked backward from Q to determinate heat flux, hot-gas convective heat transfer coefficient, and wall temperatures.

Exact equivalence ratios ϕ were not provided, only plots of specific heat addition. CEA



b) Constant mass flux (50kg/m²-s)

Figure **Error! No text of specified style in document.**.6 Heat Flux vs. Specific Heat Addition at Constant Mass Flux from Stevens [7]

was repeatedly called with different ϕ to find the corresponding burned gas temperature T_f and specific heat C_p so that the specific heat addition SHA,

$$SHA = \frac{C_p}{T_f - T_i}, \quad \text{(Error! No text of specified style in document.}.1)$$

matched closely to the values plotted in the paper (see Figure **Error! No text of specified style in document.**.6). Initial gas temperature $T_i$was assumed to be 298.15 [K]; chamber pressure was assumed to be 40 [psia]. All calculations were performed with properties associated with an area ratio of 1 which corresponds to a constant pressure engine's throat. The estimated values from the paper and their corresponding heat fluxes are listed in Table **Error! No text of specified style in document.**.1.

Table **Error! No text of specified style in document.**.1 Specific Heat Addition and Heat Flux Values from Stevens [7]

$\boldsymbol{SHA}$ **[kJ/kg]**	2090	2550	2890	3230	3540
$\boldsymbol{\dot{q}}$ **[kW/m^2]**	610	800	990	1200	1250

After iteration it was found that the ϕs listed in Table **Error! No text of specified style in document.**.2 produced the corresponding T_f, C_p, and therefore SHA.

Table **Error! No text of specified style in document.**.2 T_f, C_p, and SHA

$\boldsymbol{\phi}$	0.63155	0.74576	0.82159	0.88710	0.93464
$\boldsymbol{T_f}$ **[K]**	1.677□10^3	1.863□10^3	1.978□10^3	2.071□10^3	2.133□10^3
$\boldsymbol{C_p}$ **[J/(kg□K)]**	1.929□10^3	1.822□10^3	1.720□10^3	1.629□10^3	1.516□10^3
$\boldsymbol{SHA}$ **[J/kg]**	2.090□10^3	2.550□10^3	2.890□10^3	3.230□10^3	3.540□10^3

With these strongly matching ϕ, the thermodynamic and transport outputs of CEA could be used to calculate the Bartz heat transfer coefficient exactly as explained in section 2.3. The Bartz equation for hot gas convective heat transfer coefficient requires the throat diameter. This dimension was not provided in [7], but was included in an earlier paper [4] by Shank et al. that reported results on the same test rig. The listed outer diameter D_o was 6.06 [in] and detonation channel width w_{chnl} was 0.3 [in]. Annular cross-sectional area A_{chnl} was then

$$A_{chnl} = \frac{\pi(D_o^2 - (D_o - 2 * w_{chnl})^2)}{4} \quad \text{(Error! No text of specified style in document.}.2\text{)}$$

and the perimeter of the channel Per_{chnl} is

$$Per_{chnl} = \pi\big(D_o + (D_o - 2 * w_{chnl})\big).$$ (**Error! No text of specified style in document.**.3)

The annular chamber's hydraulic diameter $D_{h,o}$ was calculated by

$$D_{h,o} = \frac{4A_{chnl}}{Per_{chnl}}$$ (**Error! No text of specified style in document.**.4)

and was equal to 0.6 [in] or 0.01524 [m]. Two assumed hot-gas-side wall temperatures were compared, 850 [K] and 950 [K] as shown in Table **Error! No text of specified style in document.**.3.

Table **Error! No text of specified style in document.**.3 Predicted Heat Flux

	$\boldsymbol{T_{wg} = 850}$ **[K]**	$\boldsymbol{T_{wg} = 950}$ **[K]**
$\boldsymbol{\phi}$	$\boldsymbol{\dot{q}}$ **[MW/m^2]**	$\boldsymbol{\dot{q}}$ **[MW/m^2]**
0.63155	1.647	1.447
0.74576	2.184	1.956
0.82159	2.659	2.401
0.88710	3.219	2.924
0.93464	3.724	3.393

Finally, the heat flux ratios C_{hf} between the measured heat flux and the predicted constant pressure engine throat-level heat flux corresponding to the same ϕ were calculated.

$$C_{hf} = \frac{\dot{q}_{measured}}{\dot{q}_{predicted}}$$ (**Error! No text of specified style in document.**.5)

These ratios are shown in Table **Error! No text of specified style in document.**.4.

Table **Error! No text of specified style in document.**.4 C_{hf} as a Function of ϕ and T_{wg}

	$\boldsymbol{T_{wg} = 850\ [K]}$	$\boldsymbol{T_{wg} = 950\ [K]}$
$\boldsymbol{\phi}$	$\boldsymbol{C_{hf}}$	$\boldsymbol{C_{hf}}$
0.63155	0.370	0.422
0.74576	0.366	0.409
0.82159	0.372	0.412
0.88710	0.373	0.410
0.93464	0.336	0.368

The maximum C_{hf} of 0.422 arises from a ϕ of 0.63155 and a T_{wg} 950 [K]. For conservatism, the chamber heat flux knockdown analysis in 0 uses the highest C_{hf} rounded up to the nearest hundredth (so 0.43) to give the chamber the least relief from throat-level heat flux.

APPENDIX C. EFFECT OF CHAMBER HEAT FLUX KNOCKDOWN

0 describes an analysis that showed that average heat flux in the RDE chamber is likely lower than throat-level. The most conservative, as in highest, heat flux coefficient C_{hf} was 0.43, implying that the average heat load is 43% of the throat-level. Since the referenced study measured heat flux only in the chamber, C_{hf} was only applied in the RL10-comparable RDE's annular detonation chamber, and not anywhere on the plug or diverging nozzles. The Binder-anchoring coefficients $C_{B,hf}$ and $C_{B,fd}$ for heat flux and pressure drop were still applied throughout the engine. In order to have both circuits yield coolant at the same pressure, the outer needed to receive slightly more total coolant mass flow rate (74.0468%) and the inner received the remainder. Inputs for this analysis were otherwise kept the same as the original RDE analysis with no chamber heat flux knockdown as shown in Table **Error! No text of specified style in document.**.5.

Table **Error! No text of specified style in document.**.5 Chamber Heat Flux Knockdown Analysis Inputs

Property	**Symbol**	**Value**	**Units**	**Value**	**Units**
Cooling Channels:					
Number, Short	N_{chnl}	180			
Start Distance Along Nozzle Relative to Throat, Short		0.377	m	14.846	in
Number, Long	N_{chnl}	180			
Coolant:					
Species		Supercritical Hydrogen			
Initial Temperature, Both Circuits	T_{init}	31.667	K	57	^{o}R
Initial Pressure, Both Circuits	P_{init}	6.893□10^{6}	Pa	1000	psia
Chamber/Nozzle:					
Material		Stainless Steel 347			
Average Wall Thickness	th_w	3.302□10^{-4}	m	0.013	in

Thermal Conductivity	k_w	16.3	W/(m□K)	133	Btu/(hr□ft□°F)

With the chamber heat flux coefficient applied C_{hf}, the heat flux profiles were compared in Figure **Error! No text of specified style in document.**.7. The sharp dip in heat flux occurs as soon as the coolant reaches the aft end of the detonation chamber.

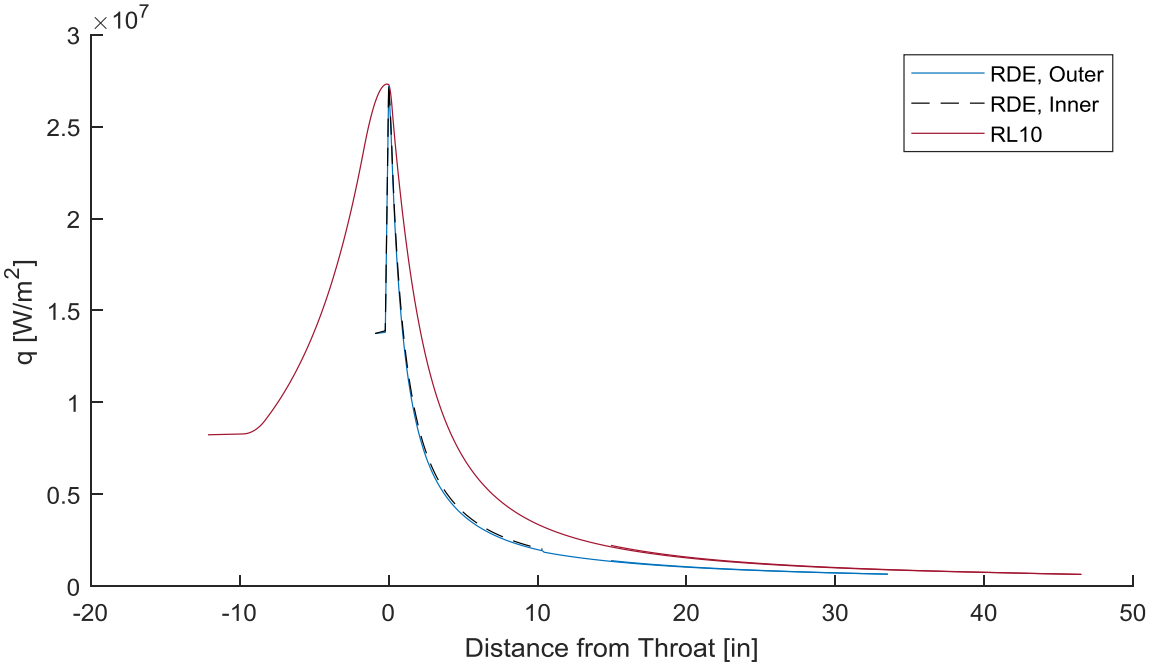


Figure **Error! No text of specified style in document.**.7 Effect of C_{hf}: Heat Flux

Table **Error! No text of specified style in document.**.6 summarizes the effect of C_{hf}.

Table **Error! No text of specified style in document.**.6 Effect of C_{hf}: Analysis Results

Property	**Symbol**	**Value**	**Units**	**Value**	**Units**
<u>Coolant:</u>					
Temperature Rise, Outer Circuit	$\Delta T_{rse,o}$	136.8	K	246.2	°R
Temperature Rise, Inner Circuit	$\Delta T_{rse,i}$	132.4	K	238.3	°R
Pressure Drop, Both Circuits	ΔP_{cl}	4.951□10^5	Pa	71.83	psid

Total Heat Transfer Rate	$\dot{Q}_{tot}$	5.231	MW	4.958□10^3	Btu/s

Table **Error! No text of specified style in document.**.7 puts these results into perspective with the full throat-level heat flux ratios.

Table **Error! No text of specified style in document.**.7 Effect of C_{hf}: RDE-to-RL10 Regen Cooling Analysis Ratios

	C_{hf}	Max h_g	Max $\dot{q}$	Max T_{wg}	Max T_{wc}	Max T_c	Outlet T_c
RDE-to-RL10 Ratio [%] with C_{hf} =	**0.43**	114.1	100.3	99.2	76.3	73.6	75.0
	1	108.7	100.3	103.2	87.7	81.8	79.6

	C_{hf}	$\Delta T_{rse,o}$	$\Delta T_{rse,i}$	ΔP_{cl}	$\dot{Q}_{tot}$
RDE-to-RL10 Ratio [%] with C_{hf} =	**0.43**	71.5	69.2	29.7	71.6
	1	75.3	78.7	29.8	76.7

Modeling the RDE chamber with the a lower, more realistic chamber heat flux correspondingly reduces the wall temperatures in that especially hot region. Hot-gas convective heat transfer coefficient increases because the peak hot-gas-side wall temperature has decreased and T_{wg} appears in part of the denominator of the σ term in the Bartz equation (Eq. (2.14).

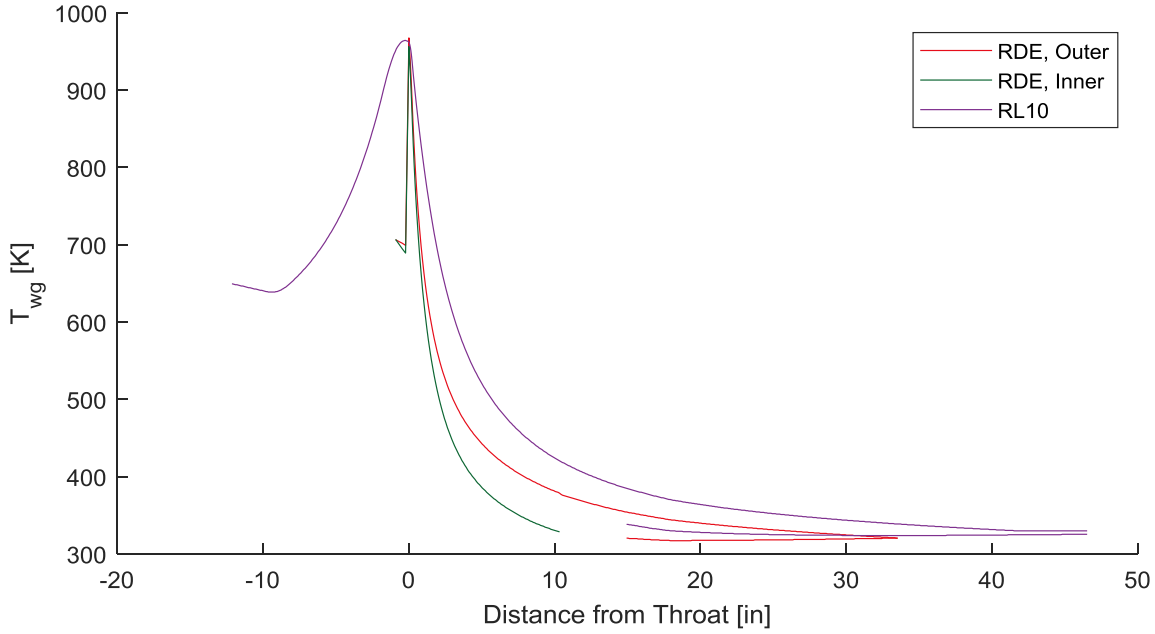


Figure **Error! No text of specified style in document.**.8 Effect of C_{hf}: Hot-Gas-Side Wall Temperature

Wall temperature drops precipitously where the where the chamber heat flux is discontinuously reduced as depicted in Figure **Error! No text of specified style in document.**.8. No data to support reducing the nozzle heat flux were available, but the heat flux and temperature profiles near the aft end of the chamber are probably smoother.

APPENDIX D. MATLAB CODE

Included herein is the code written for this model excluding unit conversion functions, some test functions for heat flux, Chernov's circle fitting function [17], Piper's CEA functions [13], and NIST's REFPROP functions [18].

I) *main*

```
%RDE vs. RL10 Comparison Wrapper

close all
clc
clearvars

tic

addpath('C:\Users\TPGur\OneDrive\Documents\Purdue\Research\RDE\CEA_MATLAB',...
    genpath('C:\Users\TPGur\OneDrive\Documents\Purdue\Research\MATLAB Common
Functions'),...
    genpath('C:\Users\TPGur\OneDrive\Documents\Purdue\Research\RDE\Heat Load
Comparison\RL10\Subfunctions'),...
    'C:\Users\TPGur\OneDrive\Documents\Purdue\Research\RDE\Heat Load
Comparison\RL10\MAT Files',...
    'C:\Users\TPGur\OneDrive\Documents\Purdue\Research\RDE\Heat Load
Comparison\RL10\Text Files',...
    '-end');

%% Global Toggles
rerunall = false;
pltall = false;

%% Engine Geometries

% Draw RL-10, output [in] & [in^2]
pltRL10 = false;
if pltall == true
    pltRL10 = true;
end
prntgeoprops_RL10 = true;
[x_RL10H_t2e,r_RL10H_t2e,x_RL10H_evnspc,r_RL10H_evnspc,Per_RL10H_evnspc,...

A_RL10H_evnspc,A_t_RL10,delx,thrtcrvtr_RL10H,x_shrtb_st,numcmbstrstnts,f3,h3,
...
    RL10Geodat] =...
    RL10Geo(pltRL10,prntgeoprops_RL10);

% Calculate and Draw Comparable RDE, output [m] & [m^2]
```

```
pltRDE = false;  % for true, need to also plot RL10 geo
if pltall == true
    pltRDE = true;
end
prntgeoprops_RDE = true;
[Arts_RDE,x_RDE,L_RDE_c,delx_RDE,delx_RDE_vec,Din_RDE,Dout_RDE,D_h_RDE,D_OD_R
DE,D_ctr,T_f_inj,T_ox_sat,T_mix,chmbrdscrtpts] =...

RDEGeo(x_RL10H_t2e,x_shrtb_st,r_RL10H_t2e,f3,h3,RL10Geodat,pltRDE,prntgeoprop
s_RDE);

%% Shared (Both RL10 & RDE) CEA Inputs

mode = 'eq';                    % set chemistry mode to equilibrium or frozen
P_c_u = 'psi';                  % chamber pressure units
ox = 'O2(L)';                   % oxidizer: liquid oxygen
ox_T = T_ox_sat;                % oxidizer temperature [K]
ox_pct = 1;                     % oxidizer percent by weight among oxidizers

%% RL10 CEA Inputs

prnt_engchrctrstcs_RL10 = true;

of_nom_RL10 = 5.26;
of_delta_RL10 = 1;
of_max_RL10 = of_nom_RL10-of_delta_RL10;
of_min_RL10 = of_nom_RL10+of_delta_RL10;

phi_RL10 = [];

p_nom_RL10 = 482;
p_delta_RL10 = 100;
p_min_RL10 = p_nom_RL10-p_delta_RL10;
p_max_RL10 = p_nom_RL10+p_delta_RL10;


P_c_RL10 = [p_min_RL10 p_nom_RL10 p_max_RL10];

of_RL10 = [of_min_RL10 of_nom_RL10 of_max_RL10];
fuel_RL10 = 'H2';                   % fuel: hydrogen gas
f_T_RL10 = T_f_inj;                 % fuel temperature [K]
f_pct_RL10 = 1;                     % fuel percent by weight among fuels (1
= 100%)
runtog_RL10 = true;                 % save toggle (true = will save)

inpfile_RL10 = 'aaaa.inp';          % can be anything (input is repressed in
CEA call)

if prnt_engchrctrstcs_RL10
    prntengchrctrstcsRL10
end

%% RDE CEA Inputs
```

```
prnt_engchrctrstcs_RDE = true;

% H2/O2 Case %%%%
of_nom_RDE = 5.26;
of_delta_RDE = 1;
of_max_RDE = of_nom_RDE+of_delta_RDE;
of_min_RDE = of_nom_RDE-of_delta_RDE;

phi_RDE = [];

p_nom_RDE = 482;
p_delta_RDE = 100;
p_min_RDE = p_nom_RDE-p_delta_RDE;
p_max_RDE = p_nom_RDE+p_delta_RDE;

P_c_RDE = [p_min_RDE p_nom_RDE p_max_RDE];

of_RDE = [of_min_RDE of_nom_RDE of_max_RDE];

% % CH4/O2 Case %%%%
%
% phi_nom_RDE = 1.64;
% phi_delta_RDE = 0.5;
% phi_max_RDE = phi_nom_RDE+phi_delta_RDE;
% phi_min_RDE = phi_nom_RDE-phi_delta_RDE;
%
% of_RDE = [];
%
% p_nom_RDE = 20*14.7;    % 20 [atm] in [psia]
% p_delta_RDE = 100;
% p_min_RDE = p_nom_RDE-p_delta_RDE;
% p_max_RDE = p_nom_RDE+p_delta_RDE;
%
% P_c_RDE = [p_min_RDE p_nom_RDE p_max_RDE];
%
% phi_RDE = [phi_min_RDE phi_nom_RDE phi_max_RDE];

fuel_RDE = 'H2';                    % fuel: hydrogen gas
f_T_RDE = T_f_inj;                  % fuel temperature [K]
f_pct_RDE = 1;                      % fuel percent by weight among fuels (1 =
100%)

runtog_RDE = true;                  % save toggle (true = will save)

inpfile_RDE = 'aaaa.inp';           % can be anything (input is repressed in
CEA call)

f8 = figure('units','normalized','outerposition',[0.1 0.1 0.9 0.9]);
f9 = figure('units','normalized','outerposition',[0.1 0.1 0.9 0.9]);
f10 = figure('units','normalized','outerposition',[0.1 0.1 0.9 0.9]);
f11 = figure('units','normalized','outerposition',[0.1 0.1 0.9 0.9]);
f12 = figure('units','normalized','outerposition',[0.1 0.1 0.9 0.9]);
f13 = cell(1,2);

if prnt_engchrctrstcs_RDE
```

```
    prntengchrctrstcsRDE
end

%% Shared Heat Flux Inputs

% Assumed wall temperatures for Bartz calculations later on
T_w_min = temp_conv(500,'F2K');     % [K]
T_w_max = temp_conv(1000,'F2K');    % [K]
T_w_del = temp_conv(500,'F2K');     % [K]
T_w = linspace(T_w_min,T_w_max,ceil((T_w_max-T_w_min)/T_w_del+1)); % [K]

%% CEA for RDE

% Toggles
CEArun_RDE = false;
if rerunall == true
    CEArun_RDE = true;
end

checkCEA_RDE = false;
if pltall == true
    checkCEA_RDE = true;
end

P_c_rgn_RDE = P_c_RDE(2);
if ~isempty(phi_RDE)
    phi_svnm = phi_RDE;
else
    if strcmp(fuel_RDE,'H2')
        of_st_H2 = (2*15.999)/(2*2.01588);
        phi_svnm = of_st_H2/of_nom_RDE;
    elseif strcmp(fuel_RDE,'CH4')
        of_st_CH4 = (4*15.999)/(12.0107+2*2.01588);
        phi_svnm = 4/of_nom_RDE;
    end
end

CEArsltsflnm = ['C:\Users\TPGur\OneDrive\Documents\Purdue\Research\RDE\Heat
Load Comparison\RL10\MAT Files\RDE_f=' fuel_RDE '_phi=' num2str(phi_svnm)
'_Pc=' num2str(P_c_rgn_RDE) '_CEA.mat'];

% Format area ratios for CEA_Piper
[eps_sub_RDE,eps_sup_RDE,x_RDE_CEA,r_in_RDE_CEA,r_out_RDE_CEA] =...
    frmtarrts_RDE(Arts_RDE,x_RDE,Din_RDE,Dout_RDE);

% Format CEA outputs
[C_p_sub,cstar_sub,gamma_sub,k_con_sub,Pr_sub,rho_sub,vel_sub,M_sub,mu_sub,T_
sub,...

C_p_sup,cstar_sup,gamma_sup,k_con_sup,Pr_sup,rho_sup,vel_sup,M_sup,mu_sup,T_s
up,...

C_p_RDE,cstar_RDE,gamma_RDE,k_con_RDE,Pr_RDE,rho_RDE,vel_RDE,M_RDE,mu_RDE,T_g
_RDE,...
 eps_tot_RDE,i_noz_st_RDE,x_RDE_plt_en] =...
```

```
CEA_Piper_RDE(CEArsltsflnm,mode,P_c_RDE,P_c_u,of_RDE,phi_RDE,fuel_RDE,f_T_RDE
,f_pct_RDE,ox,ox_T,ox_pct,inpfile_RDE,runtog_RDE,...
                  Arts_RDE,x_RDE,L_RDE_c,eps_sub_RDE,eps_sup_RDE,...
                  CEArun_RDE,checkCEA_RDE,f8,f9,f10,f11,f12,...
                  x_RDE_CEA,r_in_RDE_CEA,r_out_RDE_CEA);

%% Example Heat Flux for RDE

% Assumed wall temperatures for Bartz calculations later on

% Plot toggles
hgvphi = false; % need to run CEA w/phi for plot to work (CHECK)
hgvT = false;   % need to run CEA w/phi for plot to work (CHECK)
RDEplthxfer = false;
if pltall == true
    RDEplthxfer = true;
end

[sizeCp,h_g_1,h_g_2,h_g_3,q_f_1,Q_RDE,f2,lgdstr,h_g_1_RDE] =...

tstHtFlx_RDE(T_w,C_p_RDE,cstar_RDE,gamma_RDE,k_con_RDE,Pr_RDE,rho_RDE,vel_RDE
,M_RDE,mu_RDE,T_g_RDE,P_c_RDE,phi_RDE,...

eps_tot_RDE,i_noz_st_RDE,D_h_RDE,D_OD_RDE,D_ctr,delx_RDE_vec,Din_RDE,Dout_RDE
,x_RDE_plt_en,...
                 hgvphi,hgvT,RDEplthxfer,f9);

 %% CEA for RL10

clear eps_sub eps_sup...
    C_p_sub cstar_sub gamma_sub k_con_sub OF_sub Pr_sub rho_sub vel_sub M_sub
mu_sub T_sub...
    C_p_sup cstar_sup gamma_sup k_con_sup OF_sup Pr_sup rho_sup vel_sup M_sup
mu_sup T_sup...
    C_p cstar gamma k_con OF Pr rho vel M mu T...
    sizeCp size1 Re r T_r T_am rho_am w mu_am sig a b h_g_1_eng h_g_2_eng...
    h_g_3_eng h_g_1 h_g_2 h_g_3 Re_hg_dmnsnlss Pr_hg_dmnsnlss T_0 mu_0...
    C_p_0 k_0 q_1 q_2 q_3 q_f_1 q_f_2 q_f_3

% Toggles
CEArun_RL10 = false;
if rerunall == true
    CEArun_RL10 = true;
end

checkCEA_RL10 = false;
if pltall == true
    checkCEA_RL10 = true;
end

% Format area ratios for CEA_Piper
[eps_sub_RL10,eps_sup_RL10,x_RL10_CEA,r_RL10_CEA,Per_RL10_CEA,A_RL10_CEA,i_th
_RL10] =...
```

```
frmtarrts_RL10(x_RL10H_evnspc,r_RL10H_evnspc,Per_RL10H_evnspc,A_RL10H_evnspc)
;

% Format CEA outputs
[C_p_sub,cstar_sub,gamma_sub,k_con_sub,Pr_sub,rho_sub,vel_sub,M_sub,mu_sub,T_
sub,...

C_p_sup,cstar_sup,gamma_sup,k_con_sup,Pr_sup,rho_sup,vel_sup,M_sup,mu_sup,T_s
up,...

C_p_RL10,cstar_RL10,gamma_RL10,k_con_RL10,Pr_RL10,rho_RL10,vel_RL10,M_RL10,mu
_RL10,T_g_RL10,...
 eps_tot_RL10,x_RL10_plt_en,x_RL10_CEA,r_RL10_CEA,Per_RL10_CEA,A_RL10_CEA]
=...

CEA_Piper_RL10(mode,P_c_RL10,P_c_u,of_RL10,phi_RL10,fuel_RL10,f_T_RL10,f_pct_
RL10,ox,ox_T,ox_pct,inpfile_RL10,runtog_RL10,...

x_RL10H_evnspc,r_RL10H_evnspc,Per_RL10H_evnspc,A_RL10H_evnspc,i_th_RL10,eps_s
ub_RL10,eps_sup_RL10,...
                  CEArun_RL10,checkCEA_RL10,f8,f9,f10,f11,f12,...
                  x_RL10_CEA,r_RL10_CEA,Per_RL10_CEA,A_RL10_CEA);

 if ~checkCEA_RDE && ~checkCEA_RL10
     close(f8);close(f9);close(f10);close(f11);close(f12)
 end

%% Example Heat Flux for RL10

% Plot toggles
hgvphi = false; % need to run CEA w/phi for plot to work (CHECK)
hgvT = false;   % need to run CEA w/phi for plot to work (CHECK)
RL10plthxfer = false;
if pltall == true
    RL10plthxfer = true;
end

[sizeCp,h_g_1,h_g_2,h_g_3,q_f_1,h_g_1_RL10] =...

tstHtFlx_RL10(T_w,C_p_RL10,cstar_RL10,gamma_RL10,k_con_RL10,Pr_RL10,rho_RL10,
vel_RL10,M_RL10,mu_RL10,T_g_RL10,P_c_RL10,phi_RL10,of_RL10,...

eps_tot_RL10,i_th_RL10,r_RL10H_evnspc,A_t_RL10,delx,x_RL10_plt_en,...
                 hgvphi,hgvT,RL10plthxfer,f9,lgdstr,Q_RDE);

%% Regen Cooling for RL10

close all

P_c_rgn_RL10 = P_c_RL10(2);

calcregen_RL10 = false;
if rerunall == true
    calcregen_RL10 = true;
```

```
end
plt_regen_RL10 = false;
if pltall == true
    plt_regen_RL10 = true;
end
prnt_regen_rslts_RL10 = true;

frmtrgninpts_RL10;

[fdfctr_Bndr,hg_kd_Bndr,RL10rgn] =
regen_RL10(calcregen_RL10,plt_regen_RL10,prnt_regen_rslts_RL10,x_shrtb_st,...
    ...

x_RL10_plt_en,r_RL10H_evnspc,i_th_RL10,thrtcrvtr_RL10H,x_RL10_CEA,r_RL10_CEA,
Per_RL10_CEA,A_RL10_CEA,A_t_RL10,...
    ...

h_g_rgn_RL10,T_g_rgn_RL10,Pr_g_rgn_RL10,gamma_rgn_RL10,M_rgn_RL10,cstar_rgn_R
L10,...
    ...

T_g_chmbr_rgn_RL10,gamma_chmbr_rgn_RL10,mu_chmbr_rgn_RL10,C_p_chmbr_rgn_RL10,
Pr_chmbr_rgn_RL10,...
    ...
    P_c_rgn_RL10,eps_tot_RL10);

%% Regen Cooling for RDE

% % P_c_rgn_RDE = P_c_RDE(2);

% Case 1: Main
Chf = 1;
mdot_o_frctn = 0.74014;     % Chf = 1, H2, Pc = 482 [psia], MR = 5.26

% % Case 2: Chf = 0.43
% Chf = 0.43;
% mdot_o_frctn = 0.740468;     % Chf = 0.42, H2, Pc = 482 [psia], MR = 5.26


% if ~isempty(phi_RDE)
%     phi_svnm = phi_RDE;
% else
%     if strcmp(fuel_RDE,'H2')
%         of_st_H2 = (2*15.999)/(2*2.01588);
%         phi_svnm = of_st_H2/of_nom_RDE;
%     elseif strcmp(fuel_RDE,'CH4')
%         of_st_CH4 = (4*15.999)/(12.0107+2*2.01588);
%         phi_svnm = 4/of_nom_RDE;
%     end
% end

calcregen_RDE = false;
if rerunall == true
    calcregen_RDE = true;
end
```

```
plt_regen_RDE = true;
if pltall == true
    plt_regen_RDE = true;
end
prnt_regen_rslts_RDE = true;

frmtrgninpts_RDE;

rgnrsltsflnm = ['C:\Users\TPGur\OneDrive\Documents\Purdue\Research\RDE\Heat
Load Comparison\RL10\MAT Files\RDE_f=' fuel_RDE '_phi=' num2str(phi_svnm)
'_Pc=' num2str(P_c_rgn_RDE) '_mdto=' num2str(round(mdot_o_frctn,6)) '_Chf='
num2str(Chf) '_regen.mat'];

[mdot_c,T_c_in_mix_RDE,v_c_mix_RDE,P_c_mix] = regen_RDE(...
    ...

calcregen_RDE,plt_regen_RDE,prnt_regen_rslts_RDE,rgnrsltsflnm,Chf,mdot_o_frct
n,chmbrdscrtpts,x_shrtb_st,...
    ...

x_RDE,x_RDE_plt_en,Din_RDE,Dout_RDE,x_RDE_CEA,r_in_RDE_CEA,r_out_RDE_CEA,...
    ...

x_RL10_CEA,A_RL10_CEA,Per_RL10_CEA,A_t_RL10,h_g_rgn_RDE,T_g_rgn_RDE,Pr_g_rgn_
RDE,...
    gamma_rgn_RDE,M_rgn_RDE,cstar_rgn_RDE,...
    ...

T_g_chmbr_rgn_RDE,gamma_chmbr_rgn_RDE,mu_chmbr_rgn_RDE,C_p_chmbr_rgn_RDE,...
    Pr_chmbr_rgn_RDE,...
    ...
    P_c_rgn_RDE,eps_tot_RDE,...
    ...
    fdfctr_Bndr,hg_kd_Bndr,RL10rgn);

%% Turbine Power Calculations

prnt_trnbpwrcalcs = true;

T_c_in_RL10 = temp_conv(RL10rgn.T_c_2p(1),'R2K');           % [K]
P_c_in_RL10 = press_conv(RL10rgn.P_c_2p(1),'eng2SI')/1e3;   % [kPa]
P_c_in_RDE = P_c_mix;   % [kPa]
mdot_c_SI = mass_en2SI(mdot_c);  % [kg/s]
% v_c_mix_RDE is in [m/s]
% T_c_in_mix_RDE is in [K]
% P_c_in_RDE is in [kPa]

[xxxx] = trbnPwr(mdot_c_SI,T_c_in_RL10,T_c_in_mix_RDE,P_c_in_RL10,...
    P_c_in_RDE,v_c_mix_RDE,prnt_trnbpwrcalcs);
```

A) RL10Geo

```
function
[x_RL10H_t2e,r_RL10H_t2e,x_RL10H_evnspc,r_RL10H_evnspc,Per_RL10H_evnspc,...

A_RL10H_evnspc,A_t_en,delx,thrtcrvtr_RL10H,x_shrtb_st,numcmbstrstnts,f3,h3,RL
10Geodat] = RL10Geo(pltRL10,prntgeoprops)


% RL10 Geometry

fid = fopen('RL10EngGeo_2.txt');
C = textscan(fid,'%d%f%f%f%f','HeaderLines', 1);
fclose(fid);
i_RL10H = C{1,1};
x_RL10H = C{1,2};
r_RL10H = C{1,3};
Per_RL10H = C{1,4};
A_RL10H = C{1,5};

if pltRL10
    f1 = figure('units','normalized','outerposition',[0.2 0.1 0.7 0.6]);
    hold on

    subplot(1,3,1)
    plot(x_RL10H,r_RL10H)
    xlabel('Distance from Throat [in]');
    ylabel('Radius [in]')
    title('Engine Radius')

    subplot(1,3,2)
    plot(x_RL10H,Per_RL10H)
    xlabel('Distance from Throat [in]');
    ylabel('Perimeter [in]')
    title('Single Cooling Channel Perimeter')

    subplot(1,3,3)
    plot(x_RL10H,A_RL10H)
    xlabel('Distance from Throat [in]');
    ylabel('Area [in^2]')
    title('Single Cooling Channel Cross-Sectional Area')
end

i_RL10H(63) = [];
x_RL10H(63) = [];
r_RL10H(63) = [];
Per_RL10H(63) = [];
A_RL10H(63) = [];

Par = CircleFitByTaubin([x_RL10H(22:69,:),r_RL10H(22:69,:)]);
thrtcrvtr_RL10H = Par(3);

L_c_wrd = 0.89+0.27;
```

```
L_tot_wrd = 3.24;
L_tot_en = 67.485;  %[in]
sclng = L_tot_en/L_tot_wrd;
minX_Bnder = -(12+3/21);   % [in]
L_c_en = abs(minX_Bnder-min(x_RL10H));
L_c = len_conv(L_c_en,'engin2m');

j = 1;
x_RL10H_evnspc(j) = x_RL10H(j);
r_RL10H_evnspc(j) = r_RL10H(j);
Per_RL10H_evnspc(j) = Per_RL10H(j);
A_RL10H_evnspc(j) = A_RL10H(j);
delx = 0.1; % [in]

if pltRL10
    f2 = figure(2);
    plot(x_RL10H,r_RL10H,'b')
    xlabel('Distance from Throat [in]');
    ylabel('Radius [in]')
%     title('RL-10 Radial Profile')
end

for i = 2:length(x_RL10H)

    if abs(x_RL10H(i)-x_RL10H_evnspc(j)) <= delx %+ tol
        j = j + 1;
        x_RL10H_evnspc(j) = round(x_RL10H(i),3); % [in]
        r_RL10H_evnspc(j) = r_RL10H(i);          % [in]
        Per_RL10H_evnspc(j) = Per_RL10H(i);
        A_RL10H_evnspc(j) = A_RL10H(i);
    else
        while abs(x_RL10H(i)-x_RL10H_evnspc(j)) > delx %+ tol
            j = j + 1;
            x_RL10H_evnspc(j) = round(x_RL10H_evnspc(j-1) + delx,3);
            r_RL10H_evnspc(j) = interp1([x_RL10H(i-1) x_RL10H(i)],[r_RL10H(i-
1) r_RL10H(i)],x_RL10H_evnspc(j));
            Per_RL10H_evnspc(j) = interp1([x_RL10H(i-1)
x_RL10H(i)],[Per_RL10H(i-1) Per_RL10H(i)],x_RL10H_evnspc(j));
            A_RL10H_evnspc(j) = interp1([x_RL10H(i-1) x_RL10H(i)],[A_RL10H(i-
1) A_RL10H(i)],x_RL10H_evnspc(j));
        end
    end

end

j = j + 1;
x_RL10H_evnspc(j) = x_RL10H(end);
r_RL10H_evnspc(j) = r_RL10H(end);
Per_RL10H_evnspc(j) = Per_RL10H(end);
A_RL10H_evnspc(j) = A_RL10H(end);

i_t = find(x_RL10H==0);
i_pr_t = find(x_RL10H_evnspc<0,1,'last');
i_pst_t = find(x_RL10H_evnspc>0,1);
```

```
x_RL10H_t2e = [x_RL10H(i_t) x_RL10H_evnspc(i_pst_t:end)];
r_RL10H_t2e = [r_RL10H(i_t) r_RL10H_evnspc(i_pst_t:end)];

x_RL10H_evnspc = [x_RL10H_evnspc(1:i_pr_t) x_RL10H(i_t)
x_RL10H_evnspc(i_pst_t:end)];
r_RL10H_evnspc = [r_RL10H_evnspc(1:i_pr_t) r_RL10H(i_t)
r_RL10H_evnspc(i_pst_t:end)];
Per_RL10H_evnspc = [Per_RL10H_evnspc(1:i_pr_t) Per_RL10H(i_t)
Per_RL10H_evnspc(i_pst_t:end)];
A_RL10H_evnspc = [A_RL10H_evnspc(1:i_pr_t) A_RL10H(i_t)
A_RL10H_evnspc(i_pst_t:end)];

x_conv_st = x_RL10H_evnspc(1);

numcmbstrstnts = 0;
for i = 1:floor(L_c_en/delx)
    numcmbstrstnts = numcmbstrstnts + 1;
    x_RL10H_evnspc = [x_conv_st-i*delx x_RL10H_evnspc];
    r_RL10H_evnspc = [r_RL10H_evnspc(1) r_RL10H_evnspc];
    Per_RL10H_evnspc = [Per_RL10H_evnspc(1) Per_RL10H_evnspc];
    A_RL10H_evnspc = [A_RL10H_evnspc(1) A_RL10H_evnspc];
end

if mod(L_c_en,delx)~=0
    numcmbstrstnts = numcmbstrstnts + 1;
    x_RL10H_evnspc = [minX_Bnder x_RL10H_evnspc];
    r_RL10H_evnspc = [r_RL10H_evnspc(1) r_RL10H_evnspc];
    Per_RL10H_evnspc = [Per_RL10H_evnspc(1) Per_RL10H_evnspc];
    A_RL10H_evnspc = [A_RL10H_evnspc(1) A_RL10H_evnspc];
end

f3 = figure('units','normalized','outerposition',[0.6 0 0.4 1]);    %
placeholder
h3 = plot([0 0],[0 0],'k','LineWidth',2);   % placeholder

% x_shrtb_st = 14.7;  % short tube start point [in]
x_shrtb_st = 14 + 11/13;  % short tube start point [in]

D_chnl_RL10H_evnspc = 4*A_RL10H_evnspc./Per_RL10H_evnspc;

if pltRL10
    f3 = figure('units','normalized','outerposition',[0.6 0 0.4 1]);
    hold on
    h3 = plot(x_RL10H_evnspc,r_RL10H_evnspc,'k','LineWidth',2);
    figure(f2)
    hold on
    plot(x_RL10H_evnspc,r_RL10H_evnspc,'k--','LineWidth',2);
    legend('Original','Interpolated','location','Northwest')

    f4 = figure('units','normalized','outerposition',[0.6 0 0.4 1]);
    hold on
    h1 = plot(x_RL10H_evnspc,r_RL10H_evnspc,'k','LineWidth',2);
    h2 = plot(x_RL10H_evnspc,-r_RL10H_evnspc,'k','LineWidth',2);
    h3 = plot([-10.5 47],[0 0],'k--');
    xlabel('Distance from Throat [in]');
```

```
    ylabel('Distance from Axial Centerline [in]')
    xlim([-10.5 47])

%     f5 = figure('units','normalized','outerposition',[0.4 0.1 0.6 0.4]);
%     D_chnl_RL10H = 4*A_RL10H./Per_RL10H;
%     i_regen_strt = find(x_RL10H>=x_shrtb_st,1,'first');
%
%     subplot(1,2,1)
%     hold on
%     h1 =
plot(x_RL10H(i_regen_strt:end),D_chnl_RL10H(i_regen_strt:end),'k','LineWidth'
,2);
%     xlabel('Distance from Throat [in]');
%     ylabel('Channel Hydraulic Diameter [in]')
%     title('Short Tubes')
%     xlim([-10.5 47])
%
%     subplot(1,2,2)
%     hold on
%     h1 = plot(x_RL10H,D_chnl_RL10H,'k','LineWidth',2);
%     xlabel('Distance from Throat [in]');
%     ylabel('Channel Hydraulic Diameter [in]')
%     title('Long Tubes')
%     xlim([-10.5 47])

    f5 = figure('units','normalized','outerposition',[0.4 0.1 0.6 0.4]);
    i_regen_strt = find(x_RL10H_evnspc>=x_shrtb_st,1,'first');

    subplot(1,2,1)
    hold on
    h1 =
plot(x_RL10H_evnspc(i_regen_strt:end),D_chnl_RL10H_evnspc(i_regen_strt:end),'
k','LineWidth',2);
    xlabel('Distance from Throat [in]');
%     ylabel('Channel Hydraulic Diameter [in]')
    ylabel('D_{h,c} [in]')
    title('Short Tubes')
    xlim([-13.143 47.54])

    subplot(1,2,2)
    hold on
    h1 = plot(x_RL10H_evnspc,D_chnl_RL10H_evnspc,'k','LineWidth',2);
    xlabel('Distance from Throat [in]');
%     ylabel('Channel Hydraulic Diameter [in]')
    ylabel('D_{h,c} [in]')
    title('Long Tubes')
    xlim([-13.143 47.54])

end

numtubes_long = 180;
numtubes_shrt = 180;
Per_eng_RL10H_evnspc = 2*pi*r_RL10H_evnspc; % [in]
frctn_D_chnl_long =
numtubes_long*D_chnl_RL10H_evnspc./Per_eng_RL10H_evnspc; % []
D_chnl_RL10H_evnspc_shrt = D_chnl_RL10H_evnspc;
```

```
D_chnl_RL10H_evnspc_shrt(x_RL10H_evnspc<x_shrtb_st)=0;
frctn_D_chnl_shrt =
numtubes_shrt*D_chnl_RL10H_evnspc_shrt./Per_eng_RL10H_evnspc; % []
frctn_D_chnl_tot =
(numtubes_long*D_chnl_RL10H_evnspc+numtubes_shrt*D_chnl_RL10H_evnspc_shrt)./P
er_eng_RL10H_evnspc; % []

i_th_evnspc = find(x_RL10H_evnspc==0);
V_c_en = 0;             % volume from injection to throat [in^3]
SA_c_en = 0;            % chamber heated surface area [in^2]

% Volume up to throat for characteristic length (L*) calculation
for i = 2:i_th_evnspc

    % Frustum pieces
    R1 = r_RL10H_evnspc(i-1);
    R2 = r_RL10H_evnspc(i);
    h = x_RL10H_evnspc(i)-x_RL10H_evnspc(i-1);
    s = sqrt((R1-R2)^2+h^2);

    % Frustum volume
    Vol_inc = (1/3)*pi*h*(R1^2+R1*R2+R2^2);
    V_c_en = V_c_en + Vol_inc;   % [in^3]

    % Frustum surface area not including top and bottom surfaces
    SA_inc = pi*(R1+R2)*s;
    SA_c_en = SA_c_en + SA_inc;    % chamber heated surface area [in^2]

end

SA_shrttb_en = 0;
SA_lngtb_en = 0;
SA_noz_en = 0;
SA_tot = 0;         % total (chamber + nozzle) heated surface area [in^2]
SA_conv_en = 0;
SA_c_cyl_en = 0;

tol = 0.0001;

% Total Surface Area
for i = 2:length(x_RL10H_evnspc)

    % Frustum pieces
    R1 = r_RL10H_evnspc(i-1);
    R2 = r_RL10H_evnspc(i);
    h = x_RL10H_evnspc(i)-x_RL10H_evnspc(i-1);
    s = sqrt((R1-R2)^2+h^2);

    % Frustum surface area not including top and bottom surfaces
    SA_inc = pi*(R1+R2)*s;
    SA_tot = SA_tot+SA_inc;   % total (chamber + nozzle) heated surface area
[in^2]
    if x_RL10H_evnspc(i) > x_shrtb_st
        SA_shrttb_en = SA_shrttb_en + SA_inc/2;
        SA_lngtb_en = SA_lngtb_en + SA_inc/2;
```

```
    else
        SA_lngtb_en = SA_lngtb_en + SA_inc;
    end
    if x_RL10H_evnspc(i) > 0
        SA_noz_en = SA_noz_en + SA_inc;
    end
    if x_RL10H_evnspc(i) <= -9.7980+tol
        SA_c_cyl_en = SA_c_cyl_en + SA_inc;
    elseif x_RL10H_evnspc(i) > -9.7980+tol && x_RL10H_evnspc(i) <= 0+tol
        SA_conv_en = SA_conv_en + SA_inc;
    end
end

% fprintf('RL10Geo Total Surface Area = %0.5f
[m^2]\n\n',ar_conv(SA_tot,'in22m2'));

D_t = len_conv(r_RL10H_evnspc(i_th_evnspc)*2,'engin2m'); % [m]
D_c = len_conv(r_RL10H_evnspc(1)*2,'engin2m'); % [m]
D_e = len_conv(r_RL10H_evnspc(end)*2,'engin2m'); % [m]
A_t_en = pi*r_RL10H_evnspc(i_th_evnspc)^2;  %[in^2]
A_e_en = pi*r_RL10H_evnspc(end)^2;          %[in^2]
A_c_en = pi*r_RL10H_evnspc(1)^2;          %[in^2]
AeAt = A_e_en/A_t_en;
Lstar_en = V_c_en/A_t_en;               %[in]
Lstar = len_conv(Lstar_en,'engin2m');   % [m]

if prntgeoprops
    fprintf('RL10 Geometric Properties:\n\n')
    fprintf('  Expansion ratio = %0.3f []\n',AeAt)
%     fprintf('  Lengths:\n')
%
%     fprintf('    Combustion Chamber = %0.3f [in] or %0.4f [m]\n',-
minX_Bnder,len_conv(-minX_Bnder,'engin2m'))
    fprintf('  Diameters:\n')
    fprintf('    Chamber = %0.3f [in] or %0.4f
[m]\n',len_conv(D_c,'m2engin'),D_c)
    fprintf('    Throat = %0.3f [in] or %0.4f
[m]\n',len_conv(D_t,'m2engin'),D_t)
    fprintf('    Exit = %0.3f [in] or %0.4f
[m]\n',len_conv(D_e,'m2engin'),D_e)
    fprintf('  Cross Sectional Areas:\n')
    fprintf('    Chamber = %0.3f [in^2] or %0.4f
[m^2]\n',A_c_en,ar_conv(A_c_en,'in22m2'))
    fprintf('    Throat = %0.3f [in^2] or %0.4f
[m^2]\n',A_t_en,ar_conv(A_t_en,'in22m2'))
    fprintf('    Exit = %0.3f [in^2] or %0.4f
[m^2]\n',A_e_en,ar_conv(A_e_en,'in22m2'))
    fprintf('  Lengths:\n')
    fprintf('    Characteristic = %0.3f [in] or %0.4f [m]\n',Lstar_en,Lstar)
    fprintf('    Combustion Chamber (incl. Conv Noz) = %0.4f [m] or %0.3f
[in]\n',len_conv(abs(minX_Bnder),'engin2m'),abs(minX_Bnder))
    fprintf('    Combustion Chamber (Cyl. Only) = %0.4f [m] or %0.3f
[in]\n',len_conv(-9.7980-minX_Bnder,'engin2m'),-9.7980-minX_Bnder)
    fprintf('    Converging Nozzle = %0.4f [m] or %0.3f
[in]\n',len_conv(abs(-9.7980),'engin2m'),abs(-9.7980))
```

```
    fprintf('    Divering Nozzle = %0.4f [m] or %0.3f
[in]\n',len_conv(x_RL10H_evnspc(end),'engin2m'),x_RL10H_evnspc(end))
    fprintf('    Total = %0.4f [m] or %0.3f
[in]\n',len_conv(abs(minX_Bnder)+max(x_RL10H_evnspc),'engin2m'),abs(minX_Bnde
r)+max(x_RL10H_evnspc))
    fprintf('  Surface Areas:\n')
    fprintf('    Chamber (Cyl. Only) = %0.4f [m^2] or %0.3f
[in^2]\n',ar_conv(SA_c_cyl_en,'in22m2'),SA_c_cyl_en)
    fprintf('    Chamber (incl. Conv Noz) = %0.4f [m^2] or %0.3f
[in^2]\n',ar_conv(SA_c_en,'in22m2'),SA_c_en)
    fprintf('    Short Tubes = %0.4f [m^2] or %0.3f
[in^2]\n',ar_conv(SA_shrttb_en,'in22m2'),SA_shrttb_en)
    fprintf('    Long Tubes = %0.4f [m^2] or %0.3f
[in^2]\n',ar_conv(SA_lngtb_en,'in22m2'),SA_lngtb_en)
    fprintf('    Converging Nozzle = %0.4f [m^2] or %0.3f
[in^2]\n',ar_conv(SA_conv_en,'in22m2'),SA_conv_en)
    fprintf('    Diverging Nozzle = %0.4f [m^2] or %0.3f
[in^2]\n',ar_conv(SA_noz_en,'in22m2'),SA_noz_en)
    fprintf('    Total = %0.4f [m^2] or %0.3f
[in^2]\n',ar_conv(SA_tot,'in22m2'),SA_tot)
    fprintf('  Short Tubes:\n')
    fprintf('    Start Point = %0.4f [m] or %0.3f
[in]\n',len_conv(x_shrtb_st,'engin2m'),x_shrtb_st)
    fprintf('    Axial Length = %0.4f [m] or %0.3f
[in]\n',len_conv(max(x_RL10H_evnspc)-
x_shrtb_st,'engin2m'),max(x_RL10H_evnspc)-x_shrtb_st)
    fprintf('      %% Nozzle = %0.4f []\n',(max(x_RL10H_evnspc)-
x_shrtb_st)/max(x_RL10H_evnspc)*100)
    fprintf('      %% Total = %0.4f []\n',(max(x_RL10H_evnspc)-
x_shrtb_st)/(max(x_RL10H_evnspc)-min(x_RL10H_evnspc))*100)
    fprintf('    Cooled Surface Area (Already Divided by 2 When Shared w/Long
Tubes):\n')
    fprintf('      SA = %0.4f [m^2] or %0.3f
[in^2]\n',ar_conv(SA_shrttb_en,'in22m2'),SA_shrttb_en)
    fprintf('        %% Nozzle = %0.4f []\n',SA_shrttb_en/SA_noz_en*100)
    fprintf('        %% Long Tubes = %0.4f
[]\n',SA_shrttb_en/SA_lngtb_en*100)
    fprintf('        %% Total = %0.4f []\n\n',SA_shrttb_en/SA_tot*100)
end

RL10Geodat.L_chr = Lstar;                                   % [m]
RL10Geodat.L_c_cyl = len_conv(-9.7980-minX_Bnder,'engin2m'); % [m]
RL10Geodat.L_c_conv = len_conv(abs(minX_Bnder),'engin2m'); % [m]
RL10Geodat.L_conv = len_conv(abs(-9.7980),'engin2m'); % [m]
RL10Geodat.L_div = len_conv(x_RL10H_evnspc(end),'engin2m'); % [m]
RL10Geodat.L_tot = len_conv(abs(-
minX_Bnder)+x_RL10H_evnspc(end),'engin2m');    % [m]
RL10Geodat.D_c = D_c;                                       % [m]
RL10Geodat.D_hc = D_c;                                      % [m]
RL10Geodat.D_t = D_t;                                       % [m]
RL10Geodat.D_ht = D_t;                                      % [m]
RL10Geodat.D_e = D_e;                                       % [m]
RL10Geodat.D_he = D_e;                                      % [m]
RL10Geodat.A_c = ar_conv(A_c_en,'in22m2');                  % [m^2]
RL10Geodat.A_t = ar_conv(A_t_en,'in22m2');                  % [m^2]
RL10Geodat.A_e = ar_conv(A_e_en,'in22m2');                  % [m^2]
RL10Geodat.SA_c_cylonly = ar_conv(SA_c_cyl_en,'in22m2'); % [m^2]
```

```
RL10Geodat.SA_c_wconv = ar_conv(SA_c_en,'in22m2');      % [m^2]
RL10Geodat.SA_shrt = ar_conv(SA_shrttb_en,'in22m2');    % [m^2]
RL10Geodat.SA_lng = ar_conv(SA_lngtb_en,'in22m2');      % [m^2]
RL10Geodat.SA_conv = ar_conv(SA_conv_en,'in22m2');      % [m^2]
RL10Geodat.SA_div = ar_conv(SA_noz_en,'in22m2');        % [m^2]
RL10Geodat.SA_tot = ar_conv(SA_tot,'in22m2');           % [m^2]


end
```

B) RDEGeo

```
function [Arts_RDE,x_RDE,L_RDE_c,delx_RDE,delx_RDE_vec,Din,Dout,D_h_RDE,...
          D_OD_RDE,D_ctr,T_f_inj,T_ox_sat,T_mix,chmbrdscrtpts] =...

RDEGeo(x_RL10H_t2e,x_shrtb_st,r_RL10H_t2e,f3,h3,RL10Geodat,pltRDE,prntgeoprop
s)

% RDE Geometry

R_O_RL10_txt = len_conv(5.3666,'engin2m');      % [m]
R_th_RL10_txt = len_conv(2.4495,'engin2m');    % [m]
% R_O_RL10_bndr = len_conv(5.13,'engin2m');      % [m]
% R_th_RL10_bndr = len_conv(2.47,'engin2m');     % [m]

R_O = R_O_RL10_txt;
R_th = R_th_RL10_txt;

D_OD_RL10 = R_O*2;                  % [m]
D_OD_RDE = D_OD_RL10;               % [m]

A_t_RL10 = pi*R_th^2;                           % [m^2]
A_t_RL10_eng = ar_SI2eng(A_t_RL10);             % [in^2]
A_e_RL10 = pi*len_in2m(r_RL10H_t2e(end))^2;     % [m^2]
A_e_RL10_eng = ar_SI2eng(A_t_RL10);             % [in^2]

% AeAt_RL10 = 61.094960433152835;

v_ox = mean([25 120]);% [ft/s]
v_f = mean([100 1000]);% [ft/s]
mdot_f_en = 5.973; % [lbm/s]
mdot_ox_en = 31.4; % [lbm/s]

A_chnl = A_t_RL10;                  % [m^2]
A_RDE_c = pi*D_OD_RDE^2/4;          % [m^2]
A_RDE_c_eng = ar_SI2eng(A_RDE_c);   % [in^2]
A_ctr = A_RDE_c - A_chnl;           % [m^2]
D_ctr = sqrt(4*A_ctr/pi);           % [m]
D_ctr_eng = len_m2in(D_ctr);        % [in]
w_chnl = (D_OD_RDE-D_ctr)/2;        % [m]
w_chnl_eng = len_m2in(w_chnl);      % [in]
Per_chnl = pi*(D_OD_RDE + D_ctr);

% mdot_en = 37.36;                    % [lbm/s]
mdot_en = 37.391;
mdot = mass_en2SI(mdot_en);         % [kg/s]

vel_CJ = 3422.7;                                % [m/s] detonation velocity
from CEA detonation calculation for H2, O2, P = 33.233 bar, O/F = 5.26, T_mix
= 119.308 K
numwaves = 1;                                   % assume 1 waves present
(conservative for fill height and therefore total length)
t_fill = pi*D_OD_RDE/(vel_CJ*numwaves);         % [s]
```

```
rho_O2_L = 1141;                                    % [kg/m^3]
pct_O2 = 1-5.973/37.391;
rho_H2_L = 70.8;                                    % [kg/m^3]
pct_H2 = 5.973/37.391;
rho_inj = rho_O2_L*pct_O2 + rho_H2_L*pct_H2;        % [kg/m^3]
AinjAchnl = 0.1;                                    % assume ratio of injector
area to full channel area
A_inj = A_chnl*AinjAchnl;                           % [m^2]
v_inj = mdot/(rho_inj*A_inj);                       % [m/s]
h_fill_1 = v_inj*t_fill;                                  % [m]
[h_fill_2,v_fill,T_f_inj,T_ox_sat,T_mix] =
fillHeight(mass_en2SI(mdot_f_en),mass_en2SI(mdot_ox_en),t_fill);
% L_RDE_c = 2*h;                                    % [m]
[h_fill_3, v_inj_3] = fillHeight_LOXonly(t_fill);
h_fill = max([h_fill_1,h_fill_2,h_fill_3]);

FS_fill = 2;    % fill height factor of safety for sizing chamber

L_RDE_c = FS_fill*h_fill;                                 % [m]

% Set up chamber
x_min = 0;
x_max = L_RDE_c;
delx_RDE = 0.005;

x_RDE = linspace(x_min,x_max,(x_max-x_min)/delx_RDE+1);
chmbrdscrtpts = length(x_RDE);
x_noz = x_RDE(find(x_RDE<L_RDE_c,1,'last'));

% Add on RL10 outer nozzle
x_RDE = [x_RDE zeros(1,length(x_RL10H_t2e(2:end)))];
Dout = zeros(1,length(x_RDE));
Din = zeros(1,length(x_RDE));
A_RDE = zeros(1,length(x_RDE));
Arts_RDE = zeros(1,length(x_RDE));
D_h_RDE = zeros(1,length(x_RDE));

noz_ang = 15;                       % [degrees]

x_plug_e = 0;
SA_inr = 0; % [m^2]
SA_out = 0; % [m^2]
SA_out_long = 0;
SA_out_short = 0;
SA_div = 0;

tol = 0.0001;

for i = 1:chmbrdscrtpts
    if x_RDE(i) <= L_RDE_c
        Dout(i) = D_OD_RDE;
        Din(i) = D_ctr;
        A_RDE(i) = pi*(Dout(i)^2-Din(i)^2)/4;
        Arts_RDE(i) = A_RDE(i)/A_chnl;
        D_h_RDE(i) = 4*A_RDE(i)/Per_chnl;
```

```
    end
    if x_RDE(i) < x_plug_e || x_plug_e == 0
        A_RDE(i) = pi/4*(Dout(i)^2 - Din(i)^2);
        Per_noz = pi*(Dout(i)+Din(i));
    else
        A_RDE(i) = pi/4*(Dout(i)^2);
        Per_noz = pi*Dout(i);
    end
    Arts_RDE(i) = A_RDE(i)/A_chnl;
    D_h_RDE(i) = 4*A_RDE(i)/Per_noz;
    if i>1
       SA_inr = SA_inr + pi*(Din(i)+Din(i-1))*abs(x_RDE(i)-x_RDE(i-1))/2;    %
[m^2]
       SA_out = SA_out + pi*(Dout(i)+Dout(i-1))*abs(x_RDE(i)-x_RDE(i-1))/2; %
[m^2]

        if x_RDE(i)-L_RDE_c > len_conv(x_shrtb_st,'engin2m')+tol
            SA_out_long = SA_out_long + pi*(Dout(i)+Dout(i-1))*abs(x_RDE(i)-
x_RDE(i-1))/2/2;
            SA_out_short = SA_out_short + pi*(Dout(i)+Dout(i-
1))*abs(x_RDE(i)-x_RDE(i-1))/2/2;
        else
            SA_out_long = SA_out_long + pi*(Dout(i)+Dout(i-1))*abs(x_RDE(i)-
x_RDE(i-1))/2;
        end
        if x_RDE(i)-L_RDE_c > 0+tol
            SA_div = SA_div + pi*(Dout(i)+Dout(i-1))*abs(x_RDE(i)-x_RDE(i-
1))/2;
        end

    end
end

SA_c_inr = SA_inr;
SA_c = SA_inr + SA_out;          % annular chamber heated surface area [m^2]

Dout(chmbrdscrtpts+1:end) = len_in2m(r_RL10H_t2e(2:end))*2 +
Dout(chmbrdscrtpts)-len_in2m(r_RL10H_t2e(1)*2);
x_RDE(chmbrdscrtpts+1:end) =
len_in2m(x_RL10H_t2e(2:end))+x_RDE(chmbrdscrtpts);

delx_RDE_vec = diff(x_RDE);

% AeAt_arplg = arplgD_H2(A_chnl,Pc,of,fuel,T_f_inj,ox,T_ox_sat);  % [in]
min_D_inr = 4;      % [in]



for i = chmbrdscrtpts+1:length(x_RDE)
    Din(i) = D_ctr - (x_RDE(i)-x_RDE(chmbrdscrtpts))*tand(noz_ang)*2;
    if Din(i) < len_in2m(min_D_inr)
        Din(i) = 0;
        if x_plug_e == 0
            x_plug_e = x_RDE(i-1);
        end
```

```
    end
    if x_RDE(i) < x_plug_e || x_plug_e == 0
        A_RDE(i) = pi/4*(Dout(i)^2 - Din(i)^2);
        Per_noz = pi*(Dout(i)+Din(i));
    else
        A_RDE(i) = pi/4*(Dout(i)^2);
        Per_noz = pi*Dout(i);
    end
    Arts_RDE(i) = A_RDE(i)/A_chnl;
    D_h_RDE(i) = 4*A_RDE(i)/Per_noz;
    if A_RDE(i) >= A_e_RL10
        break;
    end

    h = x_RDE(i)-x_RDE(i-1);        % height [m]
    if Din(i) ~= 0
        s_in = sqrt(((Din(i)-Din(i-1))/2)^2+h^2);     % inner slant height
[m]
        SA_inr = SA_inr + pi*(Din(i)+Din(i-1))/2*s_in;      % [m^2]
    else
        SA_inr = SA_inr + 0;
    end
    s_out = sqrt(((Dout(i)-Dout(i-1))/2)^2+h^2);  % outer slant height [m]
    if x_RDE(i)-L_RDE_c > len_conv(x_shrtb_st,'engin2m')+tol
        SA_out_long = SA_out_long + pi*(Dout(i)+Dout(i-1))/2*s_out/2;
        SA_out_short = SA_out_short + pi*(Dout(i)+Dout(i-1))/2*s_out/2;
    else
        SA_out_long = SA_out_long + pi*(Dout(i)+Dout(i-1))/2*s_out;
    end
    if x_RDE(i)-L_RDE_c > 0+tol
        SA_div = SA_div + pi*(Dout(i)+Dout(i-1))/2*s_out;   % [m^2]
    end
    SA_out = SA_out + pi*(Dout(i)+Dout(i-1))/2*s_out;    % [m^2]
end

SA_tot = SA_inr + SA_out;   % total (chamber + nozzle) heated surface area
[m^2]

A_RDE = A_RDE(A_RDE~=0);
Arts_RDE = Arts_RDE(1:length(A_RDE));
x_RDE = x_RDE(1:length(A_RDE));
delx_RDE_vec = delx_RDE_vec(1:length(A_RDE));
Din = Din(1:length(A_RDE));
Dout = Dout(1:length(A_RDE));
A_RDE_e = pi*Dout(end)^2/4;

i_th = find(Arts_RDE<=1+tol,1,'last');
Vol_c = 0;

for i = 2:i_th
    A1 = pi/4*(Dout(i-1)^2-Din(i-1)^2); % cylinder increment previous base
area [m^2]
    A2 = pi/4*(Dout(i)^2-Din(i)^2); % cylinder increment current base area
[m^2]
    h = x_RDE(i)-x_RDE(i-1);    % cylinder increment height [m]
    Vol_inc = (A1+A2)*h/2;  % increment volume [m^2]
```

```
    Vol_c = Vol_c + Vol_inc;     % chamber volume [m^2]
end

A_t = pi/4*(Dout(i_th)^2-Din(i_th)^2);  % throat area [m^2]
L_star = Vol_c/A_t;

Dh_c = Dout(1)-Din(1); % chamber diameter (hydraulic) [m]
Dh_t = Dout(i_th)-Din(i_th); % throat diameter (hydraulic) [m]
Dh_e = Dout(end)-Din(end); % exit diameter (hydraulic) [m]

D_cntr = Din(1); % chamber diameter (hydraulic) [m]
D_outr_c = Dout(1); % throat diameter (hydraulic) [m]
D_outr_e = Dout(end); % exit diameter (hydraulic) [m]

A_c = pi/4*(Dout(1)^2-Din(1)^2); % chamber area [m^2]
A_e = pi/4*(Dout(end)^2-Din(end)^2); % chamber area [m^2]

if pltRDE
    figure(f3)
    hold on
    h6 = plot(len_m2in(x_RDE-L_RDE_c),-len_m2in(Dout/2),'b','LineWidth',2);
    h7 = plot(len_m2in(x_RDE(find(x_RDE<=x_plug_e))-L_RDE_c),-
len_m2in(Din(find(x_RDE<=x_plug_e))/2),'b','LineWidth',2);
    h8 = plot([len_m2in(x_plug_e-L_RDE_c) len_m2in(x_plug_e-L_RDE_c)],[0 -
min_D_inr/2],'b','LineWidth',2);
    h_rec_1=rectangle('Position',[len_m2in(-L_RDE_c) len_m2in(-D_ctr/2)
len_m2in(L_RDE_c) len_m2in(D_ctr/2)],'FaceColor', 'b','EdgeColor','b');
    plot([-15 55],[0 0],'k-.')
    plot([x_shrtb_st x_shrtb_st],[-20 20],'k--')    % vertical line marking
where short tubes start
    ylim([-20 20])
    xlim([-15 55])
    legend('RL10','RDE','location','northwest')
    xlabel('Distance from Throat [in]');
    ylabel('Distance from Centerline [in]')
%     legend([h3 h6], {'RL10','RDE'},'location','northwest')

    f6 = figure(7);
    hold on
    h9 = plot(len_m2in(x_RDE-L_RDE_c),len_m2in(Dout/2),'b','LineWidth',2);
    h10 = plot(len_m2in(x_RDE-L_RDE_c),-len_m2in(Dout/2),'b','LineWidth',2);
    h11 = plot(len_m2in(x_RDE(find(x_RDE<=x_plug_e))-
L_RDE_c),len_m2in(Din(find(x_RDE<=x_plug_e))/2),'b','LineWidth',2);
    h12 = plot(len_m2in(x_RDE(find(x_RDE<=x_plug_e))-L_RDE_c),-
len_m2in(Din(find(x_RDE<=x_plug_e))/2),'b','LineWidth',2);
    h_rec_2 = rectangle('Position',[len_m2in(-L_RDE_c) len_m2in(-D_ctr/2)
len_m2in(L_RDE_c) len_m2in(D_ctr)],'FaceColor', 'b','EdgeColor','b');
    h13 = plot([len_m2in(x_plug_e-L_RDE_c) len_m2in(x_plug_e-L_RDE_c)],[-
min_D_inr/2 min_D_inr/2],'b','LineWidth',2);
    xlim([-1.5 34]);
    xlabel('Distance from Throat [in]');
    ylabel('Distance from Centerline [in]')

    f7 = figure(8);
    hold on
```

```
    h9 = plot(len_m2in(x_RDE-L_RDE_c),len_m2in(Dout/2),'b','LineWidth',2);
    h10 = plot(len_m2in(x_RDE-L_RDE_c),-len_m2in(Dout/2),'b','LineWidth',2);
    h11 = plot(len_m2in(x_RDE(find(x_RDE<=x_plug_e))-
L_RDE_c),len_m2in(Din(find(x_RDE<=x_plug_e))/2),'b','LineWidth',2);
    h12 = plot(len_m2in(x_RDE(find(x_RDE<=x_plug_e))-L_RDE_c),-
len_m2in(Din(find(x_RDE<=x_plug_e))/2),'b','LineWidth',2);
    h_rec_2 = rectangle('Position',[len_m2in(-L_RDE_c) len_m2in(-D_ctr/2)
len_m2in(L_RDE_c) len_m2in(D_ctr)],'FaceColor', 'b','EdgeColor','b');
    h13 = plot([len_m2in(x_plug_e-L_RDE_c) len_m2in(x_plug_e-L_RDE_c)],[-0.5
0.5],'b','LineWidth',2);
    xlabel('Distance from Throat [in]');
    ylabel('Distance from Centerline [in]')
    xlim([-len_conv(L_RDE_c,'m2engin')-0.1 len_conv(L_RDE_c,'m2engin')+0.1])
end

if prntgeoprops
    fprintf('RDE Geometric Properties:\n\n')
    fprintf('  Fill Velocities:\n')
    fprintf('    1 = %0.3f [m/s] or %0.3f
[in/s]\n',v_inj,vel_conv(v_inj,'SI2engin'))
    fprintf('    2 = %0.3f [m/s] or %0.3f
[in/s]\n',v_fill,vel_conv(v_fill,'SI2engin'))
    fprintf('    3 = %0.3f [m/s] or %0.3f
[in/s]\n',v_inj_3,vel_conv(v_inj_3,'SI2engin'))
    fprintf('  Fill Heights:\n')
    fprintf('    1 = %0.5f [mm] or %0.3f
[in]\n',h_fill_1*1e3*FS_fill,len_conv(h_fill_1,'m2engin')*FS_fill)
    fprintf('    2 = %0.5f [mm] or %0.3f
[in]\n',h_fill_2*1e3*FS_fill,len_conv(h_fill_2,'m2engin')*FS_fill)
    fprintf('    3 = %0.5f [mm] or %0.3f
[in]\n',h_fill_3*1e3*FS_fill,len_conv(h_fill_3,'m2engin')*FS_fill)
    fprintf('  Expansion ratio = %0.3f []\n',max(Arts_RDE))
    fprintf('  Chamber Volume = %0.7f [m^3] or %0.4f
[in^3]\n',Vol_c,Vol_c/0.0254^3)
    fprintf('  Lengths:\n')
    fprintf('    Characteristic = %0.3f [in] or %0.4f
[m]\n',len_conv(L_star,'m2engin'),L_star)
    fprintf('    Chamber = %0.3f [in] or %0.4f
[m]\n',len_conv(L_RDE_c,'m2engin'),L_RDE_c)
    fprintf('    Plug Nozzle = %0.3f [in] or %0.4f [m]\n',len_conv(x_plug_e-
L_RDE_c,'m2engin'),x_plug_e-L_RDE_c)
    L_RDE_div = x_RDE(end)-L_RDE_c; % [m]
    L_RDE_tot = x_RDE(end);         % [m]
    fprintf('    Diverging Nozzle = %0.3f [in] or %0.4f
[m]\n',len_conv(L_RDE_div,'m2engin'),L_RDE_div)
    fprintf('    Total = %0.4f [m] or %0.3f
[in]\n',max(x_RDE),len_conv(max(x_RDE),'m2engin'))
    fprintf('  Diameters:\n')
    fprintf('    Hydraulic:\n')
    fprintf('      Chamber = %0.3f [in] or %0.4f
[m]\n',len_conv(Dh_c,'m2engin'),Dh_c)
    fprintf('      Throat = %0.3f [in] or %0.4f
[m]\n',len_conv(Dh_t,'m2engin'),Dh_t)
    fprintf('      Exit = %0.3f [in] or %0.4f
[m]\n',len_conv(Dh_e,'m2engin'),Dh_e)
```

```
    fprintf('      Aeroplug Termination = %0.3f [in] or %0.4f
[m]\n',len_conv(Dout(find(Din~=0,1,'last'))-
min(Din(Din~=0)),'m2engin'),Dout(find(Din~=0,1,'last'))-min(Din(Din~=0)))
    fprintf('    Other:\n')
    fprintf('      Chamber, Outer  = %0.3f [in] or %0.4f
[m]\n',len_conv(D_outr_c,'m2engin'),D_outr_c)
    fprintf('      Chamber, Center = %0.3f [in] or %0.4f
[m]\n',len_conv(D_cntr,'m2engin'),D_cntr)
    fprintf('      Exit, Outer = %0.3f [in] or %0.4f
[m]\n',len_conv(D_outr_e,'m2engin'),D_outr_e)
    fprintf('      Aeroplug Termination = %0.3f [in] or %0.4f
[m]\n',len_conv(min(Din(Din~=0)),'m2engin'),min(Din(Din~=0)))
    fprintf('  Cross Sectional Areas:\n')
    fprintf('    Chamber = %0.3f [in^2] or %0.4f
[m^2]\n',ar_conv(A_c,'m22in2'),A_c)
    fprintf('    Throat = %0.3f [in^2] or %0.4f
[m^2]\n',ar_conv(A_t,'m22in2'),A_t)
    fprintf('    Exit = %0.3f [in^2] or %0.4f
[m^2]\n',ar_conv(A_e,'m22in2'),A_e)
    fprintf('  Detonation Channel:\n')
    fprintf('    Width = %0.4f [m] or %0.3f [in]\n',(D_outr_c-
D_cntr)/2,len_conv((D_outr_c-D_cntr)/2,'m2engin'))
    fprintf('    Circumference, Outer = %0.4f [m] or %0.3f
[in]\n',pi*D_outr_c,len_conv(pi*D_outr_c,'m2engin'))
    fprintf('    Length = %0.4f [m] or %0.3f
[in]\n',L_RDE_c,len_conv(L_RDE_c,'m2engin'))
    fprintf('  Surface Areas:\n')
    fprintf('    Chamber = %0.4f [m^2] or %0.3f
[in^2]\n',SA_c,ar_conv(SA_c,'m22in2'))
    fprintf('    Outer, Short = %0.4f [m^2] or %0.3f
[in^2]\n',SA_out_short,ar_conv(SA_out_short,'m22in2'))
    fprintf('    Outer, Long = %0.4f [m^2] or %0.3f
[in^2]\n',SA_out_long,ar_conv(SA_out_long,'m22in2'))
    fprintf('    Outer, Total = %0.4f [m^2] or %0.3f
[in^2]\n',SA_out,ar_conv(SA_out,'m22in2'))
    fprintf('    Inner = %0.4f [m^2] or %0.3f
[in^2]\n',SA_inr,ar_conv(SA_inr,'m22in2'))
    fprintf('    Diverging = %0.4f [m^2] or %0.3f
[in^2]\n',SA_div,ar_conv(SA_div,'m22in2'))
    fprintf('    Total = %0.4f [m^2] or %0.3f
[in^2]\n',SA_tot,ar_conv(SA_tot,'m22in2'))
    fprintf('  Ratios:\n')
    fprintf('    D_c_RDE/D_c_RL10 = %0.4f\n',D_outr_c/RL10Geodat.D_c)
    fprintf('    D_ht_RDE/D_ht_RL10 = %0.4f\n',Dh_t/RL10Geodat.D_ht)
    fprintf('    L_c_RDE/L_c_cyl_RL10 = %0.4f\n',L_RDE_c/RL10Geodat.L_c_cyl)
    fprintf('    L_c_RDE/L_c_conv_RL10
= %0.4f\n',L_RDE_c/RL10Geodat.L_c_conv)
    fprintf('    L_div_RDE/L_div_RL10 = %0.5f\n',L_RDE_div/RL10Geodat.L_div)
    fprintf('    L_tot_RDE/L_tot_RL10 = %0.5f\n',L_RDE_tot/RL10Geodat.L_tot)
    fprintf('    L_c_RDE/L_inr_RDE = %0.5f\n',L_RDE_c/x_plug_e)
    fprintf('    L_c_RDE/L_out_RDE = %0.5f\n',L_RDE_c/max(x_RDE))
    fprintf('    L_inr_RDE/L_out_RDE = %0.5f\n',x_plug_e/max(x_RDE))
    fprintf('    L_inr_RDE/L_tot_RDE = %0.5f\n',x_plug_e/L_RDE_tot)
    fprintf('    L_out_RDE/L_tot_RDE = %0.5f\n',max(x_RDE)/L_RDE_tot)
    fprintf('    A_t_RDE/A_t_RL10 = %0.4f\n',A_t/RL10Geodat.A_t)
    fprintf('    SA_c_RDE/SA_c_cyl_RL10
= %0.4f\n',SA_c/RL10Geodat.SA_c_cylonly)
```

```
    fprintf('    SA_c_RDE/SA_c_conv_RL10
= %0.4f\n',SA_c/RL10Geodat.SA_c_wconv)
    fprintf('    SA_tot_RDE/SA_tot_RL10 = %0.4f\n',SA_tot/RL10Geodat.SA_tot)
    fprintf('    SA_inr_RDE/SA_tot_RDE = %0.4f\n',SA_inr/SA_tot)
    fprintf('    SA_out_RDE/SA_tot_RDE = %0.4f\n',SA_out/SA_tot)
    fprintf('    SA_inr_RDE/SA_out_RDE = %0.4f\n\n',SA_inr/SA_out)

end

end
```

1) *fillHeight*

```
function [h_fill,v_fill,T_f_inj,T_ox_sat,T_mix] = fillHeight(mdot_f, mdot_ox,
t_fill)

    T_f_L = 20.28; % [K]
    T_f_cj = temp_R2K(344.4); % [K]
    T_f_inj = T_f_L + T_f_cj;% [K]

    T_ox_sat = 90.15; % [K]

    A = 33.066178;
    B = -11.363417;
    C = 11.432816;
    D = -2.772874;
    E = -0.158558;
    t = T_f_inj/1000;
    cp_f = A + B*t + C*t^2 + D*t^3 + E/t^2;% [J/(mol*K)]
    M_f = 2*1.00794; % [g/mol]
    cp_f = cp_f/(M_f/1000);% [J/(kg*K)]

    A = 31.32234;
    B = -20.23531;
    C = 57.86644;
    D = -36.50624;
    E = -0.007374;
    t = T_ox_sat/1000;
    cp_Gox = A + B*t + C*t^2 + D*t^3 + E/t^2;% [J/(mol*K)]
    M_ox = 31.9988; % [g/mol]
    cp_Gox = cp_Gox/(M_ox/1000);% [J/(kg*K)]

    h_v_Lox = 214; %[kJ/kg]
    h_v_Lox = 214*1000; %[J/kg]

%     mdot_ox_en = 31.4; % [lbm/s]
%     mdot_f_en = 5.973; % [lbm/s]
%     mdot_ox = mass_en2SI(mdot_ox_en); % [kg/s]
%     mdot_f = mass_en2SI(mdot_f_en); % [kg/s]
    mdot_tot = mdot_ox + mdot_f; % [kg/s]

    T_mix = (-mdot_ox*cp_Gox*T_ox_sat + mdot_ox*h_v_Lox -
mdot_f*cp_f*T_f_inj)/...
        (-(mdot_f*cp_f + mdot_ox*cp_Gox)); % [K]

    %O/F nom is 5.0
    y_f = mdot_f/mdot_tot;
    y_ox = mdot_ox/mdot_tot;

    M_mix = (y_f/M_f + y_ox/M_ox)^(-1); % [g/mol]
    M_mix = M_mix/1000;  % [kg/mol]

    P_c = 475; % nominal chamber pressure [psia]
    P_c = P_c*101325/14.7; % [Pa]
```

```
    R_u = 8.314; % [J/(mol*K)]

    rho_mix = P_c/((R_u/M_mix)*T_mix); % [kg/m^3]

    A_c = 0.0161; % [m^2]

    v_fill = mdot_tot/(rho_mix*A_c); % [m/s]

%     t_fill = 1.529527019300489e-04; % [s]

    h_fill = v_fill*t_fill; % [m] (old calc was 0.0038 [m])
end
```

2) *fillHeight_LOXonly*

```
function [h,v_inj] = fillHeight_LOXonly(t_fill)

addpath('C:\Users\TPGur\OneDrive\Documents\Purdue\Research\MATLAB Common
Functions\Unit Conversions',...
    '-end');

% addpath('C:\Users\TPGur\Documents\Tim Gurshin\Purdue\Fall
2018\Research\Unit Conversions', '-end');

% v_det = 3080;               % detonation wave speed for hydrogen/oxygen
[m/s]
v_det = 3422.7;
P_c = 482;                  % chamber pressure [psia]
% P_c = 460;                  % chamber pressure [psia]
P_c = press_eng2SI(P_c);    % chamber pressure [Pa]

P_p = 660;                  % LOX pump discharge pressure [psia]
P_p = press_eng2SI(P_p);    % LOX pump discharge pressure [Pa]

rho = 1141;                 % LOX injection density [kg/m^3]

v_inj = sqrt((P_p-P_c)*2/rho);          % LOX injection velocity [m/s]

D_ODE = len_in2m(5.367*2);  % [m]
% t_c = D_ODE*pi/v_det;

h = v_inj*t_fill;           % [m]

end
```

C) prntengchrctrstcsRL10

```
mdot_c = 5.973;                          % coolant mass flow rate [lbm/s]
mdot_tot = mdot_c*(1+of_nom_RL10);       %  total mass flow rate [lbm/s]
mdot_ox = mdot_tot-mdot_c;               % oxidizer flow rate [lbm/s]

mdot_c_SI = mdot_c/2.20462;              % coolant mass flow rate [kg/s]
mdot_tot_SI = mdot_tot/2.20462;          %  total mass flow rate [kg/s]
mdot_ox_SI = mdot_ox/2.20462;            % oxidizer flow rate [kg/s]
Pc_SI = p_nom_RL10/14.7*101325;          % [Pa]

fprintf('RL10 Engine Characteristics:\n\n')
fprintf('  Mixture Ratio = %0.2f\n',of_nom_RL10)
fprintf('  Mass Flow Rates:\n')
fprintf('    Total = %0.3f [kg/s] or %0.3f [lbm/s]\n',mdot_tot_SI,mdot_tot)
fprintf('    Fuel/Coolant = %0.3f [kg/s] or %0.3f
[lbm/s]\n',mdot_c_SI,mdot_c)
fprintf('    Oxidizer = %0.3f [kg/s] or %0.3f [lbm/s]\n',mdot_ox_SI,mdot_ox)
fprintf('  Chamber Pressure = %0.1f [Pa] or %0.3f [psia]\n',Pc_SI,p_nom_RL10)
fprintf(['  Fuel is ' fuel_RL10 '\n'])
fprintf('  Fuel Injection Temp = %0.3f [K] or %0.3f
[F]\n',T_f_inj,temp_conv(T_f_inj,'K2F'))
fprintf(['  Ox is ' ox '\n'])
fprintf('  Ox Injection Temp = %0.3f [K] or %0.3f
[F]\n\n',T_ox_sat,temp_conv(T_ox_sat,'K2F'))
```

D) prntengchrctrstcsRDE

```
mdot_c = 5.973;                          % coolant mass flow rate [lbm/s]
mdot_tot = mdot_c*(1+of_nom_RL10);       %  total mass flow rate [lbm/s]
mdot_ox = mdot_tot-mdot_c;               % oxidizer flow rate [lbm/s]

mdot_c_SI = mdot_c/2.20462;              % coolant mass flow rate [kg/s]
mdot_tot_SI = mdot_tot/2.20462;          %  total mass flow rate [kg/s]
mdot_ox_SI = mdot_ox/2.20462;            % oxidizer flow rate [kg/s]
Pc_SI = p_nom_RL10/14.7*101325;          % [Pa]

fprintf('RDE Engine Characteristics:\n\n')
fprintf('  Mixture Ratio = %0.2f\n',of_nom_RDE)
fprintf('  Mass Flow Rates:\n')
fprintf('    Total = %0.3f [kg/s] or %0.3f [lbm/s]\n',mdot_tot_SI,mdot_tot)
fprintf('    Fuel/Coolant = %0.3f [kg/s] or %0.3f
[lbm/s]\n',mdot_c_SI,mdot_c)
fprintf('    Oxidizer = %0.3f [kg/s] or %0.3f [lbm/s]\n',mdot_ox_SI,mdot_ox)
fprintf('  Chamber Pressure = %0.1f [Pa] or %0.3f [psia]\n',Pc_SI,p_nom_RDE)
fprintf(['  Fuel is ' fuel_RDE '\n'])
fprintf('  Fuel Injection Temp = %0.3f [K] or %0.3f
[F]\n',T_f_inj,temp_conv(T_f_inj,'K2F'))
fprintf(['  Ox is ' ox '\n'])
fprintf('  Ox Injection Temp = %0.3f [K] or %0.3f
[F]\n\n',T_ox_sat,temp_conv(T_ox_sat,'K2F'))
```

E) frmtarrts_RDE

```
function [eps_sub,eps_sup,x_RDE_CEA,r_in_RDE_CEA,r_out_RDE_CEA] =...
            frmtarrts_RDE(Arts_RDE,x_RDE,Din,Dout)

    eps_sub = [];
    indx_t = 0;
    tol = 0.0001;
    i_th = find(Arts_RDE<1+tol,1,'last');

    for i = i_th:length(Arts_RDE)  % Creates subsonic and supersonic area
ratios for CEA, was 1 instead of i_th
        if i == i_th % was 1
            eps_sub(i-i_th+1) = Arts_RDE(i);
        elseif (Arts_RDE(i)-Arts_RDE(i-1) <= 0 || Arts_RDE(i) == Arts_RDE(1))
&& indx_t == 0
            eps_sub(i-i_th+1) = Arts_RDE(i);
        else
            if indx_t == 0
                indx_t = i-1;
            end
            eps_sup(i-i_th+1-length(eps_sub)) = Arts_RDE(i);
        end
    end

    L_eps_sub_og = length(eps_sub);
    [eps_sub,IA,IC] = unique(eps_sub);
    eps_sub = flip(unique(eps_sub));
    IA = flip(IA)+i_th-1;
    IC = flip(IC)+i_th-1;
    x_RDE_CEA = x_RDE(IA);
    r_in_RDE_CEA = Din(IA)/2;
    r_out_RDE_CEA = Dout(IA)/2;
    eps_sup = [1.0 eps_sup];
    [eps_sup,IA,IC] = unique(eps_sup(2:end));
    eps_sup = [1.0 eps_sup];
%     x_RDE_tmp = x_RDE(indx_t:end);
%     r_in_RDE_tmp = Din(indx_t:end)/2;
%     r_out_RDE_tmp = Dout(indx_t:end)/2;
    x_RDE_tmp = x_RDE(indx_t+1:end);
    r_in_RDE_tmp = Din(indx_t+1:end)/2;
    r_out_RDE_tmp = Dout(indx_t+1:end)/2;
    x_RDE_CEA = [x_RDE_CEA x_RDE_tmp(IA)];
    r_in_RDE_CEA = [r_in_RDE_CEA r_in_RDE_tmp(IA)];
    r_out_RDE_CEA = [r_out_RDE_CEA r_out_RDE_tmp(IA)];
    L_eps_sub_diff = abs(length(eps_sub) - L_eps_sub_og);

    if length(eps_sub) == 1
        eps_sub = [eps_sub eps_sub];
    end

end
```

F) CEA_Piper_RDE

```
function
[C_p_sub,cstar_sub,gamma_sub,k_con_sub,Pr_sub,rho_sub,vel_sub,M_sub,mu_sub,T_
sub,...

C_p_sup,cstar_sup,gamma_sup,k_con_sup,Pr_sup,rho_sup,vel_sup,M_sup,mu_sup,T_s
up,...
          C_p,cstar,gamma,k_con,Pr,rho,vel,M,mu,T,...
          eps_tot,i_noz_st,x_RDE_plt_en] =...
          ...

CEA_Piper_RDE(CEArsltsflnm,mode,P_c,P_c_u,of,phi,fuel,f_T,f_pct,ox,ox_T,ox_pc
t,inpfile,runtog,...
                    Arts_RDE,x_RDE,L_RDE_c,eps_sub,eps_sup,...
                    CEArun_RDE,checkCEA_RDE,f8,f9,f10,f11,f12,...
                    x_RDE_CEA,r_in_RDE_CEA,r_out_RDE_CEA)

    % Calls Phil Piper's CEA in MATLAB main function "CEA.m" and formats
outputs

    if CEArun_RDE

        clc
        fprintf('RDE CEA:\n\n')

        if isempty(eps_sub) == 0
            debugCEA = false;
            CEAdat1 =
CEA(mode,P_c,P_c_u,of,[],fuel,f_T,f_pct,ox,ox_T,ox_pct,eps_sub,[],inpfile,run
tog,debugCEA);
        end
        debugCEA = false;
        CEAdat2 =
CEA(mode,P_c,P_c_u,of,[],fuel,f_T,f_pct,ox,ox_T,ox_pct,[],eps_sup,inpfile,run
tog,debugCEA);
        if exist(CEArsltsflnm,'file') == 2
            delete(CEArsltsflnm);
        end

        if isempty(eps_sub) == 0
            save(CEArsltsflnm, 'CEAdat1', 'CEAdat2');
        else
            save(CEArsltsflnm, 'CEAdat2');
        end

        clc
        fprintf('RDE CEA Complete\n\n')

    else
        load(CEArsltsflnm);

        fprintf('RDE CEA Loaded\n\n')
```

```
    end

    % Separate CEA outputs

    % Subsonic nozzle
    if isempty(eps_sub) == 0
        C_p_sub = squeeze(CEAdat1('cp'));        % isobaric specifich heat
constant [J/(kg*K)]'
        cstar_sub = squeeze(CEAdat1('cstar'));   % combustion efficiency
[m/s]
        gamma_sub = squeeze(CEAdat1('gammas')); % isobaric to isochoric
specific heat ratio
%         k_con_sub = squeeze(CEAdat1('k'))/1000;  % thermal conductivity
[Piper MATLAB CEA outputs (mW/m*K)] [W/(m*K)]
        k_con_sub = squeeze(CEAdat1('k'));       % thermal conductivity
[Piper MATLAB CEA outputs (W/m*K), checked with online CEA] [W/(m*K)]
        OF_sub = squeeze(CEAdat1('o/f'));        % oxidizer to fuel mixture
ratio (by mass)
        Pr_sub = squeeze(CEAdat1('prandtl'));    % Prandtl number (momentum
to thermal diffusivity ratio)
        rho_sub = squeeze(CEAdat1('rho'));       % density [kg/m^3]
        vel_sub = squeeze(CEAdat1('son'));       % sonic velocity [m/s]
        M_sub = squeeze(CEAdat1('mach'));        % Mach number [m/s]
        mu_sub = squeeze(CEAdat1('visc'));       % dynamic viscosity [Pa*s]
        T_sub = squeeze(CEAdat1('t'));           % temperature [K]'
    end

    % Supersonic nozzle
    C_p_sup = squeeze(CEAdat2('cp'));        % isobaric specifich heat
constant [J/(kg*K)]'
    cstar_sup = squeeze(CEAdat2('cstar'));   % combustion efficiency [m/s]
    gamma_sup = squeeze(CEAdat2('gammas'));  % isobaric to isochoric specific
heat ratio
%     k_con_sup = squeeze(CEAdat2('k'))/1000;  % thermal conductivity [Piper
MATLAB CEA outputs (mW/m*K)] [W/(m*K)]
    k_con_sup = squeeze(CEAdat2('k'));       % thermal conductivity [Piper
MATLAB CEA outputs (W/m*K)] [W/(m*K)]
    OF_sup = squeeze(CEAdat2('o/f'));        % oxidizer to fuel mixture ratio
(by mass)
    Pr_sup = squeeze(CEAdat2('prandtl'));    % Prandtl number (momentum to
thermal diffusivity ratio)
    rho_sup = squeeze(CEAdat2('rho'));       % density [kg/m^3]
    vel_sup = squeeze(CEAdat2('son'));       % sonic velocity [m/s]
    M_sup = squeeze(CEAdat2('mach'));        % Mach number [m/s]
    mu_sup = squeeze(CEAdat2('visc'));       % dynamic viscosity [Pa*s]
    T_sup = squeeze(CEAdat2('t'));           % temperature [K]'

    % Concatenate subsonic and supersonic nozzle results
    C_p = C_p_sup(:,:,2:end);        % isobaric specifich heat constant
[J/(kg*K)]'
    cstar = cstar_sup(:,:,2:end);
    gamma = gamma_sup(:,:,2:end);
    k_con = k_con_sup(:,:,2:end);
%     OF = OF_sup(:,:,2:end);
    Pr = Pr_sup(:,:,2:end);
    rho = rho_sup(:,:,2:end);
```

```
    vel = vel_sup(:,:,2:end);
    M = M_sup(:,:,2:end);
    mu = mu_sup(:,:,2:end);
    T = T_sup(:,:,2:end);

    % Match Length of RDE Chamber
    tol = 0.0001;
    i_noz_st = find(Arts_RDE>1+tol,1);
    sizeCp = size(C_p);
    C_p = cat(3,ones([sizeCp(1) sizeCp(1) i_noz_st-2]).*C_p(:,:,1),C_p);
    cstar = cat(3,ones([sizeCp(1) sizeCp(1) i_noz_st-
2]).*cstar(:,:,1),cstar);
    gamma = cat(3,ones([sizeCp(1) sizeCp(1) i_noz_st-
2]).*gamma(:,:,1),gamma);
    k_con = cat(3,ones([sizeCp(1) sizeCp(1) i_noz_st-
2]).*k_con(:,:,1),k_con);
%     OF = cat(3,ones([sizeCp(1) sizeCp(1) i_noz_st-2]).*OF(:,:,1),OF);
    Pr = cat(3,ones([sizeCp(1) sizeCp(1) i_noz_st-2]).*Pr(:,:,1),Pr);
    rho = cat(3,ones([sizeCp(1) sizeCp(1) i_noz_st-2]).*rho(:,:,1),rho);
    vel = cat(3,ones([sizeCp(1) sizeCp(1) i_noz_st-2]).*vel(:,:,1),vel);
    M = cat(3,ones([sizeCp(1) sizeCp(1) i_noz_st-2]).*M(:,:,1),M);
    mu = cat(3,ones([sizeCp(1) sizeCp(1) i_noz_st-2]).*mu(:,:,1),mu);
    T = cat(3,ones([sizeCp(1) sizeCp(1) i_noz_st-2]).*T(:,:,1),T);

    eps_tot = [ones(1,i_noz_st-2).*eps_sup(1) eps_sup];
    x_RDE_plt_en = [len_m2in(x_RDE(1:i_noz_st-2)-L_RDE_c),len_m2in(x_RDE_CEA-
L_RDE_c)];

    % y indicates number of full runs (1 is RDE, 2 is RL10, 1:2 is both)
    % RDE

    if checkCEA_RDE
        figure(f8)
        hold on

        subplot(3,4,1)
        hold on
        plot(x_RDE_plt_en,squeeze(C_p(2,2,:)))
        xlabel('Distance from Throat [in]')
        ylabel('C_p [J/(kg*K)]')
        title('Isobaric Specific Heat Constant')

        subplot(3,4,2)
        hold on
        plot(x_RDE_plt_en,squeeze(cstar(2,2,:)))
        xlabel('Distance from Throat [in]')
        ylabel('c* [m/s]')
        title('Combustion Characteristic Velocity')

        subplot(3,4,3)
        hold on
        plot(x_RDE_plt_en,squeeze(gamma(2,2,:)))
        xlabel('Distance from Throat [in]')
        ylabel('\gamma')
        title('Isobaric to Isochoric Specific Heat Ratio')
```

```
subplot(3,4,4)
hold on
plot(x_RDE_plt_en,squeeze(k_con(2,2,:)))
xlabel('Distance from Throat [in]')
ylabel('k [W/(m*K)]')
title('Thermal Conductivity')

subplot(3,4,5)
hold on
plot(x_RDE_plt_en,squeeze(Pr(2,2,:)))
xlabel('Distance from Throat [in]')
ylabel('Pr')
title('Prandtl Number')

subplot(3,4,6)
hold on
plot(x_RDE_plt_en,squeeze(rho(2,2,:)))
xlabel('Distance from Throat [in]')
ylabel('\rho [kg/m^3)]')
title('Density')

subplot(3,4,7)
hold on
plot(x_RDE_plt_en,squeeze(vel(2,2,:)))
xlabel('Distance from Throat [in]')
ylabel('v [m/s]')
title('Sonic Velocity')

subplot(3,4,8)
hold on
plot(x_RDE_plt_en,squeeze(M(2,2,:)))
xlabel('Distance from Throat [in]')
ylabel('M')
title('Mach Number')

subplot(3,4,9)
hold on
plot(x_RDE_plt_en,squeeze(mu(2,2,:)))
xlabel('Distance from Throat [in]')
ylabel('\mu [Pa*s]')
title('Dynamic Viscosity')

subplot(3,4,10)
hold on
plot(x_RDE_plt_en,squeeze(T(2,2,:)))
xlabel('Distance from Throat [in]')
ylabel('T [K]')
title('Static Temperature')

subplot(3,4,11)
hold on
plot(x_RDE_plt_en,eps_tot)
xlabel('Distance from Throat [in]')
ylabel('\epsilon')
```

```
        title('Area Ratio')

        %% Split Up

%         figure(f9)
%         hold on
%         subplot(1,3,1)
%         plot(x_RDE_plt_en,squeeze(C_p(2,2,:)))
%         xlabel('Distance from Throat [in]')
%         ylabel('C_p [J/(kg*K)]')
%         ylim([3000 9200])
%         title('Isobaric Specific Heat Constant')
%
%         subplot(1,3,2)
%         plot(x_RDE_plt_en,squeeze(cstar(2,2,:)))
%         xlabel('Distance from Throat [in]')
%         ylabel('c* [m/s]')
%         title('Combustion Characteristic Velocity')
%
%         subplot(1,3,3)
%         plot(x_RDE_plt_en,squeeze(gamma(2,2,:)))
%         xlabel('Distance from Throat [in]')
%         ylabel('\gamma')
%         ylim([1.14 1.28])
%         title('Isobaric to Isochoric Specific Heat Ratio')
%
%         figure(f10)
%         hold on
%         subplot(1,3,1)
%         plot(x_RDE_plt_en,squeeze(k_con(2,2,:)))
%         xlabel('Distance from Throat [in]')
%         ylabel('k [W/(m*K)]')
%         title('Thermal Conductivity')
%
%         subplot(1,3,2)
%         plot(x_RDE_plt_en,squeeze(Pr(2,2,:)))
%         xlabel('Distance from Throat [in]')
%         ylabel('Pr')
%         ylim([0.48 0.69])
%         title('Prandtl Number')
%
%         subplot(1,3,3)
%         plot(x_RDE_plt_en,squeeze(rho(2,2,:)))
%         xlabel('Distance from Throat [in]')
%         ylabel('\rho [kg/m^3)]')
%         title('Density')
%
%         figure(f11)
%         hold on
%         subplot(1,3,1)
%         plot(x_RDE_plt_en,squeeze(vel(2,2,:)))
%         xlabel('Distance from Throat [in]')
%         ylabel('v [m/s]')
%         ylim([960 1650])
%         title('Sonic Velocity')
%
```

```
%           subplot(1,3,2)
%           plot(x_RDE_plt_en,squeeze(M(2,2,:)))
%           xlabel('Distance from Throat [in]')
%           ylabel('M')
%           ylim([0 4.7])
%           title('Mach Number')
%
%           subplot(1,3,3)
%           plot(x_RDE_plt_en,squeeze(mu(2,2,:)))
%           xlabel('Distance from Throat [in]')
%           ylabel('\mu [Pa*s]')
%           ylim([4.220*1e-5 0.000102])
%           title('Dynamic Viscosity')
%
%           figure(f12)
%           hold on
%           subplot(1,2,1)
%           plot(x_RDE_plt_en,squeeze(T(2,2,:)))
%           xlabel('Distance from Throat [in]')
%           ylabel('T [K]')
%           title('Static Temperature')
%
%           subplot(1,2,2)
%           plot(x_RDE_plt_en,eps_tot)
%           xlabel('Distance from Throat [in]')
%           ylabel('\epsilon')
%           ylim([0 62])
%           title('Area Ratio')

          figure(f9)
          hold on
          subplot(1,3,1)
          plot(x_RDE_plt_en,squeeze(C_p(2,2,:)))
          xlabel('Distance from Throat [in]')
          ylabel('C_p [J/(kg*K)]')
          ylim([3000 9200])
          title('Isobaric Specific Heat Constant')

%           subplot(1,3,2)
%           plot(x_RDE_plt_en,squeeze(cstar(2,2,:)))
%           xlabel('Distance from Throat [in]')
%           ylabel('c* [m/s]')
%           title('Combustion Characteristic Velocity')

          subplot(1,3,2)
          plot(x_RDE_plt_en,squeeze(gamma(2,2,:)))
          xlabel('Distance from Throat [in]')
          ylabel('\gamma')
          ylim([1.14 1.28])
          title('Isobaric to Isochoric Specific Heat Ratio')

          subplot(1,3,3)
          plot(x_RDE_plt_en,squeeze(k_con(2,2,:)))
          xlabel('Distance from Throat [in]')
          ylabel('k [W/(m*K)]')
          title('Thermal Conductivity')
```

```
        figure(f10)
        hold on
        subplot(1,3,1)
        plot(x_RDE_plt_en,squeeze(Pr(2,2,:)))
        xlabel('Distance from Throat [in]')
        ylabel('Pr')
        ylim([0.48 0.69])
        title('Prandtl Number')

        subplot(1,3,2)
        plot(x_RDE_plt_en,squeeze(rho(2,2,:)))
        xlabel('Distance from Throat [in]')
        ylabel('\rho [kg/m^3)]')
        title('Density')

        subplot(1,3,3)
        plot(x_RDE_plt_en,squeeze(vel(2,2,:)))
        xlabel('Distance from Throat [in]')
        ylabel('v [m/s]')
        ylim([960 1650])
        title('Sonic Velocity')

        figure(f11)
        hold on
        subplot(1,3,1)
        plot(x_RDE_plt_en,squeeze(M(2,2,:)))
        xlabel('Distance from Throat [in]')
        ylabel('M')
        ylim([0 4.7])
        title('Mach Number')

        subplot(1,3,2)
        plot(x_RDE_plt_en,squeeze(mu(2,2,:)))
        xlabel('Distance from Throat [in]')
        ylabel('\mu [Pa*s]')
        ylim([4.220*1e-5 0.000102])
        title('Dynamic Viscosity')

        subplot(1,3,3)
        plot(x_RDE_plt_en,squeeze(T(2,2,:)))
        xlabel('Distance from Throat [in]')
        ylabel('T [K]')
        title('Static Temperature')

        figure(f12)
        hold on
%         subplot(1,2,2)
%         hold on
        plot(x_RDE_plt_en,eps_tot)
        xlabel('Distance from Throat [in]')
        ylabel('\epsilon')
        ylim([0 62])
        title('Area Ratio')
    end
```

```
end
```

G) frmtarrts_RL10

```
function [eps_sub,eps_sup,x_RL10_CEA,r_RL10_CEA,Per_RL10_CEA,A_RL10_CEA,i_th]
=...

frmtarrts_RL10(x_RL10H_evnspc,r_RL10H_evnspc,Per_RL10H_evnspc,A_RL10H_evnspc)

    i_th = find(x_RL10H_evnspc==0);
    A_t_RL10 = ar_eng2SI(pi*r_RL10H_evnspc(i_th)^2);
    Arts_RL10 = pi*len_in2m(r_RL10H_evnspc).^2/A_t_RL10;
    Arts_RL10_t = Arts_RL10(i_th);

    indx_t = 0;
    tol = 0.0001;
    i_cyl_end = find(x_RL10H_evnspc<-9.798+tol,1,'last');

    for i = i_cyl_end:length(Arts_RL10)
        if i == 1
            eps_sub(i-i_cyl_end+1) = Arts_RL10(i);
        elseif (Arts_RL10(i) > 1 || Arts_RL10(i) == Arts_RL10(1)) && indx_t
== 0
            eps_sub(i-i_cyl_end+1) = Arts_RL10(i);
        else
            if indx_t == 0
                indx_t = i;
            end
            eps_sup(i-i_cyl_end+1-length(eps_sub)) = Arts_RL10(i);
        end
    end

    L_eps_sub_og = length(eps_sub);
    L_eps_sup_og = length(eps_sup);
    [eps_sub,IA,IC] = unique(eps_sub);
    eps_sub = flip(eps_sub);
    IA = flip(IA)+i_cyl_end-1;
    IC = flip(IC)+i_cyl_end-1;
    x_RL10_CEA = x_RL10H_evnspc(IA);
    r_RL10_CEA = r_RL10H_evnspc(IA);
    Per_RL10_CEA = Per_RL10H_evnspc(IA);
    A_RL10_CEA = A_RL10H_evnspc(IA);
    L_eps_sub_diff = abs(length(eps_sub) - L_eps_sub_og);
    [eps_sup,IA,IC] = unique(eps_sup);
    x_RL10_tmp = x_RL10H_evnspc(indx_t:end);
    r_RL10_tmp = r_RL10H_evnspc(indx_t:end);
    Per_RL10_tmp = Per_RL10H_evnspc(indx_t:end);
    A_RL10_tmp = A_RL10H_evnspc(indx_t:end);
    x_RL10_CEA = [x_RL10_CEA x_RL10_tmp(IA)];
    r_RL10_CEA = [r_RL10_CEA r_RL10_tmp(IA)];
    Per_RL10_CEA = [Per_RL10_CEA Per_RL10_tmp(IA)];
    A_RL10_CEA = [A_RL10_CEA A_RL10_tmp(IA)];
    L_eps_sup_diff = abs(length(eps_sup) - L_eps_sup_og);

end
```

H) CEA_Piper_RL10

```
function
[C_p_sub,cstar_sub,gamma_sub,k_con_sub,Pr_sub,rho_sub,vel_sub,M_sub,mu_sub,T_
sub,...

C_p_sup,cstar_sup,gamma_sup,k_con_sup,Pr_sup,rho_sup,vel_sup,M_sup,mu_sup,T_s
up,...
          C_p,cstar,gamma,k_con,Pr,rho,vel,M,mu,T_g_RL10,...

eps_tot,x_RL10_plt_en,x_RL10_CEA,r_RL10_CEA,Per_RL10_CEA,A_RL10_CEA] =...
          ...

CEA_Piper_RL10(mode,P_c,P_c_u,of,phi,fuel,f_T,f_pct,ox,ox_T,ox_pct,inpfile,ru
ntog,...

x_RL10H_evnspc,r_RL10H_evnspc,Per_RL10H_evnspc,A_RL10H_evnspc,i_th,eps_sub,ep
s_sup,...
                    CEArun_RL10,checkCEA_RL10,f8,f9,f10,f11,f12,...
                    x_RL10_CEA,r_RL10_CEA,Per_RL10_CEA,A_RL10_CEA)

    % Calls Phil Piper's CEA in MATLAB main function "CEA.m" and formats
outputs

    if CEArun_RL10

        clc
        fprintf('RL10 CEA:\n\n')

        debugCEA = false;
        CEAdat1 =
CEA(mode,P_c,P_c_u,of,[],fuel,f_T,f_pct,ox,ox_T,ox_pct,eps_sub,[],inpfile,run
tog,debugCEA);
        debugCEA = false;
        CEAdat2 =
CEA(mode,P_c,P_c_u,of,[],fuel,f_T,f_pct,ox,ox_T,ox_pct,[],eps_sup,inpfile,run
tog,debugCEA);
        if exist('MAT Files/RL10main.mat','file') == 2
            delete('MAT Files/RL10main.mat');
        end
        save('MAT Files/RL10main.mat', 'CEAdat1', 'CEAdat2');

        clc
        fprintf('RL10 CEA Complete\n\n')

    else
        load('MAT Files/RL10main.mat');

        fprintf('RL10 CEA Loaded\n\n')
    end

    % Separate CEA outputs
```

```
    C_p_sub = squeeze(CEAdat1('cp'));        % isobaric specifich heat
constant [J/(kg*K)]'
    cstar_sub = squeeze(CEAdat1('cstar'));   % combustion efficiency [m/s]
    gamma_sub = squeeze(CEAdat1('gammas'));  % isobaric to isochoric specific
heat ratio
%     k_con_sub = squeeze(CEAdat1('k'))/1000;  % thermal conductivity [Piper
MATLAB CEA outputs (mW/m*K)] [W/(m*K)]
    k_con_sub = squeeze(CEAdat1('k'));       % thermal conductivity [Piper
MATLAB CEA outputs (W/m*K), checked with online CEA] [W/(m*K)]
    OF_sub = squeeze(CEAdat1('o/f'));        % oxidizer to fuel mixture ratio
(by mass)
    Pr_sub = squeeze(CEAdat1('prandtl'));    % Prandtl number (momentum to
thermal diffusivity ratio)
    rho_sub = squeeze(CEAdat1('rho'));       % density [kg/m^3]
    vel_sub = squeeze(CEAdat1('son'));       % sonic velocity [m/s]
    M_sub = squeeze(CEAdat1('mach'));        % Mach number [m/s]
    mu_sub = squeeze(CEAdat1('visc'));       % dynamic viscosity [Pa*s]
    T_sub = squeeze(CEAdat1('t'));           % temperature [K]'

    C_p_sup = squeeze(CEAdat2('cp'));        % isobaric specifich heat
constant [J/(kg*K)]'
    cstar_sup = squeeze(CEAdat2('cstar'));   % combustion efficiency [m/s]
    gamma_sup = squeeze(CEAdat2('gammas'));  % isobaric to isochoric specific
heat ratio
%     k_con_sup = squeeze(CEAdat2('k'))/1000;  % thermal conductivity [Piper
MATLAB CEA outputs (mW/m*K)] [W/(m*K)]
    k_con_sup = squeeze(CEAdat2('k'));       % thermal conductivity [Piper
MATLAB CEA outputs (W/m*K)] [W/(m*K)]
    OF_sup = squeeze(CEAdat2('o/f'));        % oxidizer to fuel mixture ratio
(by mass)
    Pr_sup = squeeze(CEAdat2('prandtl'));    % Prandtl number (momentum to
thermal diffusivity ratio)
    rho_sup = squeeze(CEAdat2('rho'));       % density [kg/m^3]
    vel_sup = squeeze(CEAdat2('son'));       % sonic velocity [m/s]
    M_sup = squeeze(CEAdat2('mach'));        % Mach number [m/s]
    mu_sup = squeeze(CEAdat2('visc'));       % dynamic viscosity [Pa*s]
    T_sup = squeeze(CEAdat2('t'));           % temperature [K]'

    C_p = cat(3, C_p_sub(:,:,2:end), C_p_sup(:,:,2:end));        % isobaric
specifich heat constant [J/(kg*K)]'
    cstar = cat(3, cstar_sub(:,:,2:end), cstar_sup(:,:,2:end));
    gamma = cat(3, gamma_sub(:,:,2:end), gamma_sup(:,:,2:end));
    k_con = cat(3, k_con_sub(:,:,2:end), k_con_sup(:,:,2:end));
    OF = cat(3, OF_sub(:,:,2:end), OF_sup(:,:,2:end));
    Pr = cat(3, Pr_sub(:,:,2:end), Pr_sup(:,:,2:end));
    rho = cat(3, rho_sub(:,:,2:end), rho_sup(:,:,2:end));
    vel = cat(3, vel_sub(:,:,2:end), vel_sup(:,:,2:end));
    M = cat(3, M_sub(:,:,2:end), M_sup(:,:,2:end));
    mu = cat(3, mu_sub(:,:,2:end), mu_sup(:,:,2:end));
    T_g_RL10 = cat(3, T_sub(:,:,2:end), T_sup(:,:,2:end));

    % Match Length of RL10 Cylindrical Chamber
    tol = 0.0001;
    i_cyl_end = find(x_RL10H_evnspc<-9.798+tol,1,'last');
    sizeCp = size(C_p);
    C_p = cat(3,ones([sizeCp(1) sizeCp(1) i_cyl_end-1]).*C_p(:,:,1),C_p);
```

```
    cstar = cat(3,ones([sizeCp(1) sizeCp(1) i_cyl_end-
1]).*cstar(:,:,1),cstar);
    gamma = cat(3,ones([sizeCp(1) sizeCp(1) i_cyl_end-
1]).*gamma(:,:,1),gamma);
    k_con = cat(3,ones([sizeCp(1) sizeCp(1) i_cyl_end-
1]).*k_con(:,:,1),k_con);
%     OF = cat(3,ones([sizeCp(1) sizeCp(1) 1]).*OF(:,:,1),OF);
    Pr = cat(3,ones([sizeCp(1) sizeCp(1) i_cyl_end-1]).*Pr(:,:,1),Pr);
    rho = cat(3,ones([sizeCp(1) sizeCp(1) i_cyl_end-1]).*rho(:,:,1),rho);
    vel = cat(3,ones([sizeCp(1) sizeCp(1) i_cyl_end-1]).*vel(:,:,1),vel);
    M = cat(3,ones([sizeCp(1) sizeCp(1) i_cyl_end-1]).*M(:,:,1),M);
    mu = cat(3,ones([sizeCp(1) sizeCp(1) i_cyl_end-1]).*mu(:,:,1),mu);
    T_g_RL10 = cat(3,ones([sizeCp(1) sizeCp(1) i_cyl_end-
1]).*T_g_RL10(:,:,1),T_g_RL10);

    eps_tot = [ones(1,i_cyl_end-1)*eps_sub(1) eps_sub eps_sup];

    % Concatenate subsonic and supersonic nozzle results
%     eps_tot = [eps_sub(1) eps_sub eps_sup];

    tol = 0.0001;
%     i_cyl_end = find(x_RL10H_evnspc<-9.798+tol,1,'last');
    [r_conv_un,i_r_conv_un] = unique(r_RL10H_evnspc(i_cyl_end:i_th-1));
    r_conv_un = flip(r_conv_un);
    i_r_conv_un = flip(i_r_conv_un);
    i_r_conv_un = i_r_conv_un + i_cyl_end - 1;
    [r_div_un,i_r_div_un] = unique(r_RL10H_evnspc(i_th:end));
    i_r_div_un = i_r_div_un + (i_th-1);
    i_r_tot_un = [i_r_conv_un;i_r_div_un];
    x_RL10_plt_en = x_RL10H_evnspc(i_r_tot_un);

    x_RL10_plt_en = [x_RL10H_evnspc(1:i_cyl_end-1) x_RL10_plt_en];
    x_RL10_CEA = [x_RL10H_evnspc(1:i_cyl_end-1) x_RL10_CEA];
    r_RL10_CEA = [ones(1,i_cyl_end-1)*r_RL10H_evnspc(1) r_RL10_CEA];
    Per_RL10_CEA = [ones(1,i_cyl_end-1)*Per_RL10H_evnspc(1) Per_RL10_CEA];
    A_RL10_CEA = [ones(1,i_cyl_end-1)*A_RL10H_evnspc(1) A_RL10_CEA];

%     x_RL10_plt_en = [x_RL10H_evnspc(1) x_RL10_plt_en];
%     x_RL10_CEA = [x_RL10H_evnspc(1) x_RL10_CEA];
%     r_RL10_CEA = [r_RL10H_evnspc(1) r_RL10_CEA];
%     Per_RL10_CEA = [Per_RL10H_evnspc(1) Per_RL10_CEA];
%     A_RL10_CEA = [A_RL10H_evnspc(1) A_RL10_CEA];

    if checkCEA_RL10

        figure(f8);
        hold on

        subplot(3,4,1)
        plot(x_RL10_plt_en,squeeze(C_p(2,2,:)),'k')
        xlabel('Distance from Throat [in]')
        ylabel('C_p [J/(kg*K)]')
        title('Isobaric Specific Heat Constant')
        legend('RDE','RL10','location','northeast')
```

```
subplot(3,4,2)
plot(x_RL10_plt_en,squeeze(cstar(2,2,:)),'k')
xlabel('Distance from Throat [in]')
ylabel('c* [m/s]')
title('Combustion Characteristic Velocity')
legend('RDE','RL10','location','northeast')

subplot(3,4,3)
plot(x_RL10_plt_en,squeeze(gamma(2,2,:)),'k')
xlabel('Distance from Throat [in]')
ylabel('\gamma')
title('Isobaric to Isochoric Specific Heat Ratio')
legend('RDE','RL10','location','southeast')

subplot(3,4,4)
plot(x_RL10_plt_en,squeeze(k_con(2,2,:)),'k')
xlabel('Distance from Throat [in]')
ylabel('k [W/(m*K)]')
title('Thermal Conductivity')
legend('RDE','RL10','location','northeast')

subplot(3,4,5)
plot(x_RL10_plt_en,squeeze(Pr(2,2,:)),'k')
xlabel('Distance from Throat [in]')
ylabel('Pr')
title('Prandtl Number')
legend('RDE','RL10','location','southeast')

subplot(3,4,6)
plot(x_RL10_plt_en,squeeze(rho(2,2,:)),'k')
xlabel('Distance from Throat [in]')
ylabel('\rho [kg/m^3)]')
title('Density')
legend('RDE','RL10','location','northeast')

subplot(3,4,7)
plot(x_RL10_plt_en,squeeze(vel(2,2,:)),'k')
xlabel('Distance from Throat [in]')
ylabel('v [m/s]')
title('Sonic Velocity')
legend('RDE','RL10','location','northeast')

subplot(3,4,8)
plot(x_RL10_plt_en,squeeze(M(2,2,:)),'k')
xlabel('Distance from Throat [in]')
ylabel('M')
title('Mach Number')
legend('RDE','RL10','location','southeast')

subplot(3,4,9)
plot(x_RL10_plt_en,squeeze(mu(2,2,:)),'k')
xlabel('Distance from Throat [in]')
ylabel('\mu [Pa*s]')
title('Dynamic Viscosity')
legend('RDE','RL10','location','northeast')
```

```
subplot(3,4,10)
plot(x_RL10_plt_en,squeeze(T_g_RL10(2,2,:)),'k')
xlabel('Distance from Throat [in]')
ylabel('T [K]')
title('Static Temperature')
legend('RDE','RL10','location','northeast')

subplot(3,4,11)
plot(x_RL10_plt_en,eps_tot,'k')
xlabel('Distance from Throat [in]')
ylabel('\epsilon')
title('Area Ratio')
legend('RDE','RL10','location','southeast')

%         figure(6);
%         hold on
%
%         subplot(3,4,1)
%         plot(x_RL10_plt_en,squeeze(C_p(2,2,:)),'k')
%         xlabel('Distance from Throat [in]')
%         ylabel('C_p [J/(kg*K)]')
%         title('Isobaric Specific Heat Constant')
%
%         subplot(3,4,2)
%         plot(x_RL10_plt_en,squeeze(cstar(2,2,:)),'k')
%         xlabel('Distance from Throat [in]')
%         ylabel('c* [m/s]')
%         title('Combustion Characteristic Velocity')
%
%         subplot(3,4,3)
%         plot(x_RL10_plt_en,squeeze(gamma(2,2,:)),'k')
%         xlabel('Distance from Throat [in]')
%         ylabel('\gamma')
%         title('Isobaric to Isochoric Specific Heat Ratio')
%
%         subplot(3,4,4)
%         plot(x_RL10_plt_en,squeeze(k_con(2,2,:)),'k')
%         xlabel('Distance from Throat [in]')
%         ylabel('k [W/(m*K)]')
%         title('Thermal Conductivity')
%
%         subplot(3,4,5)
%         plot(x_RL10_plt_en,squeeze(Pr(2,2,:)),'k')
%         xlabel('Distance from Throat [in]')
%         ylabel('Pr')
%         title('Prandtl Number')
%
%         subplot(3,4,6)
%         plot(x_RL10_plt_en,squeeze(rho(2,2,:)),'k')
%         xlabel('Distance from Throat [in]')
%         ylabel('\rho [kg/m^3)]')
%         title('Density')
%
%         subplot(3,4,7)
%         plot(x_RL10_plt_en,squeeze(vel(2,2,:)),'k')
```

```
%           xlabel('Distance from Throat [in]')
%           ylabel('v [m/s]')
%           title('Sonic Velocity')
%
%           subplot(3,4,8)
%           plot(x_RL10_plt_en,squeeze(M(2,2,:)),'k')
%           xlabel('Distance from Throat [in]')
%           ylabel('M')
%           title('Mach Number')
%
%           subplot(3,4,9)
%           plot(x_RL10_plt_en,squeeze(mu(2,2,:)),'k')
%           xlabel('Distance from Throat [in]')
%           ylabel('\mu [Pa*s]')
%           title('Dynamic Viscosity')
%
%           subplot(3,4,10)
%           plot(x_RL10_plt_en,squeeze(T_g_RL10(2,2,:)),'k')
%           xlabel('Distance from Throat [in]')
%           ylabel('T [K]')
%           title('Static Temperature')
%
%           subplot(3,4,11)
%           plot(x_RL10_plt_en,eps_tot,'k')
%           xlabel('Distance from Throat [in]')
%           ylabel('\epsilon')
%           title('Area Ratio')

%           figure(f9)
%           hold on
%           subplot(1,3,1)
%           hold on
%           plot(x_RL10_plt_en,squeeze(C_p(2,2,:)),'k')
%           xlabel('Distance from Throat [in]')
%           ylabel('C_p [J/(kg*K)]')
%           ylim([3000 9200])
%           title('Isobaric Specific Heat Constant')
%           legend('RDE','RL10','location','northeast')
%
%           subplot(1,3,2)
%           hold on
%           plot(x_RL10_plt_en,squeeze(cstar(2,2,:)),'k')
%           xlabel('Distance from Throat [in]')
%           ylabel('c* [m/s]')
%           title('Combustion Characteristic Velocity')
%           legend('RDE','RL10','location','northeast')
%
%           subplot(1,3,3)
%           hold on
%           plot(x_RL10_plt_en,squeeze(gamma(2,2,:)),'k')
%           xlabel('Distance from Throat [in]')
%           ylabel('\gamma')
%           ylim([1.14 1.28])
%           title('Isobaric to Isochoric Specific Heat Ratio')
%           legend('RDE','RL10','location','southeast')
%
%           figure(f10)
```

```
%         hold on
%         subplot(1,3,1)
%         hold on
%         plot(x_RL10_plt_en,squeeze(k_con(2,2,:)),'k')
%         xlabel('Distance from Throat [in]')
%         ylabel('k [W/(m*K)]')
%         title('Thermal Conductivity')
%         legend('RDE','RL10','location','northeast')
%
%         subplot(1,3,2)
%         hold on
%         plot(x_RL10_plt_en,squeeze(Pr(2,2,:)),'k')
%         xlabel('Distance from Throat [in]')
%         ylabel('Pr')
%         ylim([0.48 0.69])
%         title('Prandtl Number')
%         legend('RDE','RL10','location','northeast')
%
%         subplot(1,3,3)
%         hold on
%         plot(x_RL10_plt_en,squeeze(rho(2,2,:)),'k')
%         xlabel('Distance from Throat [in]')
%         ylabel('\rho [kg/m^3)]')
%         title('Density')
%         legend('RDE','RL10','location','northeast')
%
%         figure(f11)
%         hold on
%         subplot(1,3,1)
%         hold on
%         plot(x_RL10_plt_en,squeeze(vel(2,2,:)),'k')
%         xlabel('Distance from Throat [in]')
%         ylabel('v [m/s]')
%         ylim([960 1650])
%         title('Sonic Velocity')
%         legend('RDE','RL10','location','northeast')
%
%         subplot(1,3,2)
%         hold on
%         plot(x_RL10_plt_en,squeeze(M(2,2,:)),'k')
%         xlabel('Distance from Throat [in]')
%         ylabel('M')
%         ylim([0 4.7])
%         title('Mach Number')
%         legend('RDE','RL10','location','southeast')
%
%         subplot(1,3,3)
%         hold on
%         plot(x_RL10_plt_en,squeeze(mu(2,2,:)),'k')
%         xlabel('Distance from Throat [in]')
%         ylabel('\mu [Pa*s]')
%         ylim([4*1e-5 0.000102])
%         title('Dynamic Viscosity')
%         legend('RDE','RL10','location','northeast')
%
%         figure(f12)
%         hold on
```

```
%         subplot(1,2,1)
%         hold on
%         plot(x_RL10_plt_en,squeeze(T_g_RL10(2,2,:)),'k')
%         xlabel('Distance from Throat [in]')
%         ylabel('T [K]')
%         title('Static Temperature')
%         legend('RDE','RL10','location','northeast')
%
%         subplot(1,2,2)
%         hold on
%         plot(x_RL10_plt_en,eps_tot,'k')
%         xlabel('Distance from Throat [in]')
%         ylabel('\epsilon')
%         ylim([0 62])
%         title('Area Ratio')
%         legend('RDE','RL10','location','southeast')

        figure(f9)
        hold on
        subplot(1,3,1)
        hold on
        plot(x_RL10_plt_en,squeeze(C_p(2,2,:)),'k')
        xlabel('Distance from Throat [in]')
        ylabel('C_p [J/(kg*K)]')
        ylim([3000 9200])
        title('Isobaric Specific Heat Constant')
        legend('RDE','RL10','location','northeast')

%         subplot(1,3,2)
%         hold on
%         plot(x_RL10_plt_en,squeeze(cstar(2,2,:)),'k')
%         xlabel('Distance from Throat [in]')
%         ylabel('c* [m/s]')
%         title('Combustion Characteristic Velocity')
%         legend('RDE','RL10','location','northeast')

        subplot(1,3,2)
        hold on
        plot(x_RL10_plt_en,squeeze(gamma(2,2,:)),'k')
        xlabel('Distance from Throat [in]')
        ylabel('\gamma')
        ylim([1.14 1.28])
        title('Isobaric to Isochoric Specific Heat Ratio')
        legend('RDE','RL10','location','southeast')

        subplot(1,3,3)
        hold on
        plot(x_RL10_plt_en,squeeze(k_con(2,2,:)),'k')
        xlabel('Distance from Throat [in]')
        ylabel('k [W/(m*K)]')
        title('Thermal Conductivity')
        legend('RDE','RL10','location','northeast')

        figure(f10)
        hold on
        subplot(1,3,1)
```

```
hold on
plot(x_RL10_plt_en,squeeze(Pr(2,2,:)),'k')
xlabel('Distance from Throat [in]')
ylabel('Pr')
ylim([0.48 0.69])
title('Prandtl Number')
legend('RDE','RL10','location','northeast')

subplot(1,3,2)
hold on
plot(x_RL10_plt_en,squeeze(rho(2,2,:)),'k')
xlabel('Distance from Throat [in]')
ylabel('\rho [kg/m^3)]')
title('Density')
legend('RDE','RL10','location','northeast')

subplot(1,3,3)
hold on
plot(x_RL10_plt_en,squeeze(vel(2,2,:)),'k')
xlabel('Distance from Throat [in]')
ylabel('v [m/s]')
ylim([960 1650])
title('Sonic Velocity')
legend('RDE','RL10','location','northeast')

figure(f11)
hold on
subplot(1,3,1)
hold on
plot(x_RL10_plt_en,squeeze(M(2,2,:)),'k')
xlabel('Distance from Throat [in]')
ylabel('M')
ylim([0 4.7])
title('Mach Number')
legend('RDE','RL10','location','southeast')

subplot(1,3,2)
hold on
plot(x_RL10_plt_en,squeeze(mu(2,2,:)),'k')
xlabel('Distance from Throat [in]')
ylabel('\mu [Pa*s]')
ylim([4*1e-5 0.000102])
title('Dynamic Viscosity')
legend('RDE','RL10','location','northeast')

subplot(1,3,3)
hold on
plot(x_RL10_plt_en,squeeze(T_g_RL10(2,2,:)),'k')
xlabel('Distance from Throat [in]')
ylabel('T [K]')
title('Static Temperature')
legend('RDE','RL10','location','northeast')

figure(f12)
hold on
%         subplot(1,2,2)
```

```
%         hold on
          plot(x_RL10_plt_en,eps_tot,'k')
          xlabel('Distance from Throat [in]')
          ylabel('\epsilon')
          ylim([0 62])
          title('Area Ratio')
          legend('RDE','RL10','location','southeast')


          %% RL10 Only
          figure(13)
          hold on
          subplot(1,3,1)
          hold on
          plot(x_RL10_plt_en,squeeze(C_p(2,2,:)),'k')
          xlabel('Distance from Throat [in]')
          ylabel('C_p [J/(kg*K)]')
          ylim([3000 9200])
          title('Isobaric Specific Heat Constant')

%         subplot(1,3,2)
%         hold on
%         plot(x_RL10_plt_en,squeeze(cstar(2,2,:)),'k')
%         xlabel('Distance from Throat [in]')
%         ylabel('c* [m/s]')
%         title('Combustion Characteristic Velocity')

          subplot(1,3,2)
          hold on
          plot(x_RL10_plt_en,squeeze(gamma(2,2,:)),'k')
          xlabel('Distance from Throat [in]')
          ylabel('\gamma')
          ylim([1.14 1.28])
          title('Isobaric to Isochoric Specific Heat Ratio')

          subplot(1,3,3)
          hold on
          plot(x_RL10_plt_en,squeeze(k_con(2,2,:)),'k')
          xlabel('Distance from Throat [in]')
          ylabel('k [W/(m*K)]')
          title('Thermal Conductivity')

          figure(14)
          hold on
          subplot(1,3,1)
          hold on
          plot(x_RL10_plt_en,squeeze(Pr(2,2,:)),'k')
          xlabel('Distance from Throat [in]')
          ylabel('Pr')
          ylim([0.48 0.69])
          title('Prandtl Number')

          subplot(1,3,2)
          hold on
          plot(x_RL10_plt_en,squeeze(rho(2,2,:)),'k')
```

```
        xlabel('Distance from Throat [in]')
        ylabel('\rho [kg/m^3)]')
        title('Density')

        subplot(1,3,3)
        hold on
        plot(x_RL10_plt_en,squeeze(vel(2,2,:)),'k')
        xlabel('Distance from Throat [in]')
        ylabel('v [m/s]')
        ylim([960 1650])
        title('Sonic Velocity')

        figure(15)
        hold on
        subplot(1,3,1)
        hold on
        plot(x_RL10_plt_en,squeeze(M(2,2,:)),'k')
        xlabel('Distance from Throat [in]')
        ylabel('M')
        ylim([0 4.7])
        title('Mach Number')

        subplot(1,3,2)
        hold on
        plot(x_RL10_plt_en,squeeze(mu(2,2,:)),'k')
        xlabel('Distance from Throat [in]')
        ylabel('\mu [Pa*s]')
        ylim([4*1e-5 0.000102])
        title('Dynamic Viscosity')

        subplot(1,3,3)
        hold on
        plot(x_RL10_plt_en,squeeze(T_g_RL10(2,2,:)),'k')
        xlabel('Distance from Throat [in]')
        ylabel('T [K]')
        title('Static Temperature')

        figure(16)
        hold on
%         subplot(1,2,2)
        hold on
        plot(x_RL10_plt_en,eps_tot,'k')
        xlabel('Distance from Throat [in]')
        ylabel('\epsilon')
        ylim([0 62])
        title('Area Ratio')
    end

end
```

I) frmtrgninpts_RL10

```
% Format regen inputs for RL10

h_g_rgn_RL10 = squeeze(h_g_1_RL10(2,2,:,1));
T_g_rgn_RL10 = squeeze(T_g_RL10(2,2,:));
Pr_g_rgn_RL10 = squeeze(Pr_RL10(2,2,:));
gamma_rgn_RL10 = squeeze(gamma_RL10(2,2,:));
M_rgn_RL10 = squeeze(M_RL10(2,2,:));
cstar_rgn_RL10 = squeeze(cstar_RL10(2,2,:));
T_g_chmbr_rgn_RL10 = T_g_RL10(2,2,1);
gamma_chmbr_rgn_RL10 = gamma_RL10(2,2,1);
mu_chmbr_rgn_RL10 = mu_RL10(2,2,1);
C_p_chmbr_rgn_RL10 = C_p_RL10(2,2,1);
Pr_chmbr_rgn_RL10 = Pr_RL10(2,2,1);
```

J) regen_RL10

```
function [fdfctr_Bndr,hg_kd_Bndr,RL10rgn] = regen_RL10(...
        calcregen_RL10,plt_regen_RL10,prnt_regen_rslts,x_shrtb_st,...
        ...

x_RL10_plt_en,r_RL10H_evnspc,i_th,thrtcrvtr_RL10H,x_RL10_CEA,r_RL10_CEA,Per_R
L10_CEA,A_RL10_CEA,A_t_RL10,...
        ...

h_g,T_g,Pr_g,gamma,M,cstar,T_g_chmbr,gamma_chmbr,mu_chmbr,C_p_chmbr,Pr_g_chmb
r,P_c,eps_tot)

    rgnrsltsflnm =
'C:\Users\TPGur\OneDrive\Documents\Purdue\Research\RDE\Heat Load
Comparison\RL10\MAT Files\RL10_regen.mat';

    if calcregen_RL10

        fprintf('RL10 Regen Started:\n')

        %% Analysis Preparation

        % Constants
        g_eng = 32.2; % gravitational acceleration [ft/s^2]

        % Regenerative Cooling Jacket Channel
        i_regen_strt = find(x_RL10_plt_en>=x_shrtb_st,1,'first');
        x_RL10_plt_en_1p = x_RL10_plt_en(i_regen_strt:end);     %[in]
        num_chnls = 180;

         % Create vectors for plotting/analysis
        intlz_rgn_vecs_RL10

        % Friction & convective heat transfer coefficients to match Binder
        fdfctr_Bndr = 9.9606; % final
        hg_kd_Bndr = 0.28685; % final

        k_w_st = k_SI2eng(16.3);            % 347 stainless steel thermal
conductivity [Btu/(s*in*oF)] (https://www.sandmeyersteel.com/images/Alloy347-
SpecSheet.pdf)
        w_th_st = len_conv(0.3302/1000,'m2engin'); % average RL10 hot wall
thickness [in]
        mdot_c = 5.973;                     % coolant mass flow rate [lbm/s]
%         P_drp_inj = 62.3;                   % injector pressure drop from
Binder (Table, 2.6.1 pg. 13/184) [psia]
        P_drp_clnjckt = 242.1;              % cooling jacket pressure drop
from Binder (Table, 2.4.1 pg. 11/184) [psia]
%         Pclnt_vec_1p(i_regen_strt) = P_c+P_drp_inj+P_drp_clnjckt;   %
initial coolant pressure
        Pclnt_vec_1p(i_regen_strt) = 1000;  % initial coolant pressure from
Binder's pressure drop results (pg. 138/184) [psia]
```

```
        T_c_st_vec_1p(i_regen_strt) = 57;     % starting hydrogen temperature
per Binder [R]
        T_rs_clngjckt = 344.4;              % cooling jacket temperature rise
[oR]
        tlt_hxfr_rt_Bndr = 7994;            % Total heat transfer rate from
Binder [Btu/s]

        % Gas-side wall temperature guess increment
        T_wg_inc = 1 ;

        % Hydrogen properties
        H2props = xlsread('LH2_properties.xls');
        T_H2 = H2props(:,1);                % [R]
        rho_H2 = H2props(:,3);              % [lb/in^3]
        Cp_H2 = H2props(:,4);               % [Btu/(lb*R)]
        k_H2 = H2props(:,5);                % [Btu/(in*sec*R)]
        mu_H2 = H2props(:,6);               % [lbm/(in*sec)]

        strflg = 1;

%         A_vec_tot(1) = 0;

        %% Start of 1st pass/short tubes to the nozzle end/turnaround
manifold

        for i = i_regen_strt:length(eps_tot)

            fprintf('At RL10 Axial Station %i\n',i)

            P_H2_i = Pclnt_vec_1p(i);       % [psia]
            T_H2_i = T_c_st_vec_1p(i);      % [R]

            % Iteration intital guessed coolant temperature [R]
            T_c_temp_st = 36;

            % Iteration inital guessed gas-side wall temp [R]
%             T_wg_st = temp_K2R(575);
            if strflg == 1
                T_wg_st = temp_conv(300,'K2R');
                strflg = 0;
            else
                T_wg_st = temp_conv(300,'K2R');
            end

            % Energy Balance
            while T_c_temp_st < T_c_st_vec_1p(i)

                % Gas calculations
                Tg_st = temp_K2R(T_g(i));           % gas temp [R]
                Tc_ns_st = temp_K2R(T_g_chmbr);     % gas CC stag temp [R]
                r_st = Pr_g(i)^(1/3);               % recovery factor (assume
turbulent freestream boundary layer)
                T_aw_st = Tc_ns_st*((1 + (gamma(i)-1)/2*M(i)^2*r_st)/(1 +
(gamma(i)-1)/2*M(i)^2));  % adiabatic wall temperature [R]
```

```
                % Gas convective heat transfer coefficient [Btu/(in^2*s*oF)]
%                 b_st = (1+(gamma_chmbr-1)/2*M(i)^2);
                b_st = (1+(gamma(i)-1)/2*M(i)^2);
                sig_st = (((1/2)*(T_wg_st/Tc_ns_st)*b_st + 1/2)^0.68 *
b_st^0.12)^(-1);
                AtA = eps_tot(i)^(-1);
                R = thrtcrvtr_RL10H;
                a_st = (0.026/(r_RL10H_evnspc(i_th)*2)^0.2)*...

(mu_SI2eng(mu_chmbr)^0.2*cp_SI2eng(C_p_chmbr)/Pr_g_chmbr^0.6)...

*(P_c*g_eng/vel_ms2fs(cstar(i)))^0.8*(r_RL10H_evnspc(i_th)*2/R)^0.1;
                hg_st = a_st*(AtA)^(0.9)*sig_st;
%                 hg_st = hg_st*hg_kd_Bndr;
                hg_st_SI = hg_Bartz2SI(hg_st);      % [W/(m^2*K)]

                % Heat Flux [Btu/(in^2*s)]  (used [R] to calculate)
                q_g_st = hg_st*(T_aw_st-T_wg_st)*hg_kd_Bndr;

                % Coolant-side wall temp [R]
%                 T_wc_st = T_wg_st - (w_th_st/k_w_st)*q_g_st;
                T_wc_st = T_wg_st - temp_conv((w_th_st/k_w_st)*q_g_st,'F2R');

                % Local hydrogen properties
                %   Cp_c_st   [Btu/(lbm*R)]
                %   mu_c_st   [lbm/(in*s)]
                %   k_c_st    [Btu/(in*s*R)]
                %   Pr_c_st
                %   rho_c_st  [lbm/in^3]
                [Cp_c_st,mu_c_st,Pr_c_st,rho_c_st] =
H2props_refprop(T_H2_i,P_H2_i);

                D_h_c_st = 4*A_RL10_CEA(i)/Per_RL10_CEA(i); % channel
hydraulic diameter [in]
                G_st = mdot_c/(pi*D_h_c_st^2/4);            % weight flow
rate per unit area [lb/(in^2*s)]
                v_c_st = G_st/rho_c_st/num_chnls;           % velocity [in/s]
                Re_c_st = rho_c_st*v_c_st*D_h_c_st/mu_c_st; % Reynolds number

                % Coolant convective heat transfer coefficient (Sieder-
Tate) % [Btu/(in^2*s*oR)]
                h_c_st =
hcH2HuzelHuang(Cp_c_st,mu_c_st,Pr_c_st,G_st,D_h_c_st,T_c_st_vec_1p(i),T_wc_st
);

                % Iteration calculated coolant temperature [R]
                T_c_temp_st = T_wc_st - q_g_st/h_c_st;

                % Increment guessed gas-side wall temp [R]
                T_wg_st = T_wg_st + T_wg_inc;

            end

            % Iterate to find Darcy friction factor
```

```
            f_D = frctnDarcy(Re_c_st);

            if i ~= length(eps_tot)

                % Calculate coolant heat absorption
                delx = len_conv(sqrt((r_RL10_CEA(i)-r_RL10_CEA(i+1))^2 +
(x_RL10_CEA(i)-x_RL10_CEA(i+1))^2),'engin2m'); % frustum slant height [m]
                Davg = sqrt(eps_tot(i)*A_t_RL10*4/pi)/2 +
sqrt(eps_tot(i+1)*A_t_RL10*4/pi)/2;   % Same as frustum R1+R2 [in]
                dA = pi*Davg*len_conv(delx,'m2engin');   % frustum surface
area excluding top/bottom [in^2]

                dQ_st = dA*q_g_st/2;              % [Btu/s], add "/2" because
short tubes interspersed w/long tubes
                Q_st_vec_1p(i+1) = dQ_st + Q_st_vec_1p(i);
                T_c_st_vec_1p(i+1) = dQ_st/(mdot_c*Cp_c_st) +
T_c_st_vec_1p(i);  % [R]

                % Calculate coolant pressure drop
                delP = press_conv(...
                    fdfctr_Bndr*f_D*...

(rho_eng2SI_in3(rho_c_st)/2)*(vel_conv(v_c_st,'engin2SI')^2/len_conv(D_h_c_st
,'engin2m'))*delx,...
                    'SI2eng');
                Pclnt_vec_1p(i+1) = Pclnt_vec_1p(i) - delP;
            else
                dA = 0;
                dQ_st = 0;
            end

            if i == i_regen_strt
                dA_back = 0;
                A_vec_1p(i) = dA_back;     % [in^2]
            elseif i > i_regen_strt
                delx = sqrt((r_RL10_CEA(i)-r_RL10_CEA(i-1))^2 +
(x_RL10_CEA(i)-x_RL10_CEA(i-1))^2); % frustum slant height [in]
                Davg = sqrt(eps_tot(i)*A_t_RL10*4/pi)/2 + sqrt(eps_tot(i-
1)*A_t_RL10*4/pi)/2;   % Same as frustum R1+R2 [in^2]
                dA_back = pi*Davg*delx;   % frustum surface area excluding
top/bottom, backwards facing (unlike forward facing dA for temp/press) [in^2]
                A_vec_1p(i) = A_vec_1p(i-1) + dA_back;     % [in^2]
            end

            % Collect current values into vectors
            asgn_vec_vals_1p_RL10

        end

        %% Short-to-Long Tube Transition
        delx = len_conv(sqrt((r_RL10_CEA(i)-r_RL10_CEA(i-1))^2 +
(x_RL10_CEA(i)-x_RL10_CEA(i-1))^2),'engin2m');
        dA = pi*sqrt(eps_tot(i)*A_t_RL10*4/pi)*len_conv(delx,'m2engin');   %
[in^2]
        dQ_st = dA*q_g_st;              % [Btu/s]
```

```
        Q_st_vec(end) = dQ_st + Q_st_vec_1p(i);
        T_c_st_vec(end) = dQ_st/(mdot_c*Cp_c_st) + T_c_st_vec_1p(i);

        % Assume turnaround manifold bend radius is equal to coolant tube
radius
        kb = 1.1; % Bend loss coefficients for a pipe from Babcock & Wilson
Co. (fig. 3 on Thermopedia http://www.thermopedia.com/content/577/)
        Pclnt_vec(end) = Pclnt_vec_1p(end) -...
            press_conv(...

fdfctr_Bndr*f_D*(rho_eng2SI_in3(rho_c_st)/2)*(vel_conv(v_c_st,'engin2SI')^2/l
en_conv(D_h_c_st,'engin2m'))*pi*D_h_c_st/2 +...

(1/2)*kb*rho_eng2SI_in3(rho_c_st)*vel_conv(v_c_st,'engin2SI')^2,...
            'SI2eng');

        %% End of nozzle to injection plane

        for i = length(eps_tot):-1:1

            fprintf('At RL10 Axial Station %i\n',i)

            P_H2_i = Pclnt_vec(i);
            T_H2_i = T_c_st_vec(i);

            % Iteration intital guessed coolant temperature [R]
            T_c_temp_st = 36;

            % Iteration inital guessed gas-side wall temp [R]
            if x_RL10_plt_en(i) > 8
                T_wg_st = temp_conv(330,'K2R');
            elseif x_RL10_plt_en(i) <= 8 && x_RL10_plt_en(i) > 3.5
                T_wg_st = temp_conv(440,'K2R');
            elseif x_RL10_plt_en(i) <= 3.5
                T_wg_st = temp_conv(550,'K2R');
            end

            % Energy Balance
            while T_c_temp_st < T_c_st_vec(i)

                % Gas calculations
                Tg_st = temp_K2R(T_g(i));           % gas temp [R]
                Tc_ns_st = temp_K2R(T_g_chmbr);     % gas CC stag temp [R]
                r_st = Pr_g(i)^(1/3);               % recovery factor (assume
turbulent freestream boundary layer)
                T_r_st = Tg_st*(1 + (gamma(i)-1)/2*M(i)^2*r_st);     %
recovery temperature [R]
                T_aw_st = Tc_ns_st*((1 + (gamma(i)-1)/2*M(i)^2*r_st)/(1 +
(gamma(i)-1)/2*M(i)^2));  % adiabatic wall temperature [R]

                % Gas convective heat transfer coefficient [Btu/(in^2*s*oF)]
%                 b_st = (1+(gamma_chmbr-1)/2*M(i)^2);
                b_st = (1+(gamma(i)-1)/2*M(i)^2);
                sig_st = (((1/2)*(T_wg_st/Tc_ns_st)*b_st + 1/2)^0.68 *
b_st^0.12)^(-1);
```

```
                AtA = eps_tot(i)^(-1);
                R = thrtcrvtr_RL10H;
                a_st = (0.026/(r_RL10H_evnspc(i_th)*2)^0.2)*...
(mu_SI2eng(mu_chmbr)^0.2*cp_SI2eng(C_p_chmbr)/Pr_g_chmbr^0.6)...
*(P_c*g_eng/vel_ms2fs(cstar(i)))^0.8*(r_RL10H_evnspc(i_th)*2/R)^0.1;
                hg_st = a_st*(AtA)^(0.9)*sig_st;
%                 hg_st = hg_st*hg_kd_Bndr;
                hg_st_SI = hg_Bartz2SI(hg_st);      % [W/(m^2*K)]

                % Heat Flux [Btu/(in^2*s)]  (used [R] to calculate)
                q_g_st = hg_st*(T_aw_st-T_wg_st)*hg_kd_Bndr;   %
[Btu/(in^2*s)]

                % Coolant-side wall temp [R]
%                 T_wc_st = T_wg_st - (w_th_st/k_w_st)*q_g_st;    % [R]
                T_wc_st = T_wg_st -
temp_conv((w_th_st/k_w_st)*q_g_st,'F2R');    % [R]

                % Local hydrogen properties
                %   Cp_c_st [Btu/(lbm*R)]
                %   mu_c_st [lbm/(in*s)]
                %   k_c_st [Btu/(in*s*R)]
                %   Pr_c_st
                %   rho_c_st [lbm/in^3]
                [Cp_c_st,mu_c_st,Pr_c_st,rho_c_st] =
H2props_refprop(T_H2_i,P_H2_i);

                D_h_c_st = 4*A_RL10_CEA(i)/Per_RL10_CEA(i); % hydraulic
diameter [in]
                G_st = mdot_c/(pi*D_h_c_st^2/4);      % coolant weight flow
rate per unit area [lb/(in^2*s)]
                v_c_st = G_st/rho_c_st/num_chnls;     % [in/s]
                Re_c_st = rho_c_st*v_c_st*D_h_c_st/mu_c_st;

                % Coolant convective heat transfer coefficient (Sieder-
Tate) % [Btu/(in^2*s*oR)]
                h_c_st =
hcH2HuzelHuang(Cp_c_st,mu_c_st,Pr_c_st,G_st,D_h_c_st,T_c_st_vec(i),T_wc_st);

                % Iteration calculated coolant temperature [R]
                T_c_temp_st = T_wc_st - q_g_st/h_c_st;        % [R]

                % Increment guessed gas-side wall temp [R]
                T_wg_st = T_wg_st + T_wg_inc;

            end

            % Iterate to find Darcy friction factor
            f_D = frctnDarcy(Re_c_st);

            if i ~= 1 % was ~= length(h_g_1_RL10(2,2,:,1))

                % Calculate coolant heat absorption
```

```
                delx = len_conv(sqrt((r_RL10_CEA(i)-r_RL10_CEA(i-1))^2 +
(x_RL10_CEA(i)-x_RL10_CEA(i-1))^2),'engin2m'); % frustum slant height [m]
                Davg = sqrt(eps_tot(i)*A_t_RL10*4/pi)/2 + sqrt(eps_tot(i-
1)*A_t_RL10*4/pi)/2; % Same as frustum R1+R2 [in]
                dA = pi*Davg*len_conv(delx,'m2engin');   % [in^2]
                if i >= i_regen_strt
                    dQ_st = dA*q_g_st/2;            % [Btu/s], add "/2"
because long tubes interspersed w/short tubes in this region
                else
                    dQ_st = dA*q_g_st;              % [Btu/s]
                end
                Q_st_vec(i-1) = dQ_st + Q_st_vec(i);
                T_c_st_vec(i-1) = dQ_st/(mdot_c*Cp_c_st) +
T_c_st_vec(i);      % [R] (was i+1 instead of i-1)

                % Calculate coolant pressure drop
                delP = press_conv(...
                    fdfctr_Bndr*f_D*...

(rho_eng2SI_in3(rho_c_st)/2)*(vel_conv(v_c_st,'engin2SI')^2/len_conv(D_h_c_st
,'engin2m'))*delx,...
                    'SI2eng');
                Pclnt_vec(i-1) = Pclnt_vec(i) - delP;
            else
                dA = 0;
                dQ_st = 0;
            end

            if i == length(eps_tot)
                dA_back = 0;
                A_vec(i) = max(A_vec_1p);     % [in^2]
            elseif i < length(eps_tot)
                delx = sqrt((r_RL10_CEA(i)-r_RL10_CEA(i+1))^2 +
(x_RL10_CEA(i)-x_RL10_CEA(i+1))^2); % frustum slant height [in]
                Davg = sqrt(eps_tot(i)*A_t_RL10*4/pi)/2 +
sqrt(eps_tot(i+1)*A_t_RL10*4/pi)/2;       % Same as frustum R1+R2 [in^2]
                dA_back = pi*Davg*delx;   % frustum surface area excluding
top/bottom, backwards facing (unlike forward facing dA for temp/press) [in^2]
                A_vec(i) = A_vec(i+1) + dA_back;     % [in^2]
            end

            asgn_vec_vals_RL10

        end

            % Save results
            if i == 1
                if exist(rgnrsltsflnm,'file') == 2
                    delete(rgnrsltsflnm);
                end
                save(rgnrsltsflnm,...
                    'x_RL10_plt_en',...

'T_g_st_vec','T_aw_st_vec','T_wg_vec','T_wc_st_vec','T_c_st_vec',....
                    'q_g_st_vec','h_c_st_vec','hg_st_vec','Pclnt_vec',...
                    'f_D_vec','D_h_c_st_vec','v_c_st_vec','Q_st_vec',...
```

```
'dA_vec','A_vec','h_c_st_vec','Re_c_st_vec','rho_c_st_vec','mu_c_st_vec',...
                    'Cp_c_st_vec','Pr_c_st_vec','dA_back_vec',...
                    ...
                    'x_RL10_plt_en_1p',...

'T_g_st_vec_1p','T_aw_st_vec_1p','T_wg_vec_1p','T_wc_st_vec_1p','T_c_st_vec_1
p',...

'q_g_st_vec_1p','h_c_st_vec_1p','hg_st_vec_1p','Pclnt_vec_1p',...

'f_D_vec_1p','D_h_c_st_vec_1p','v_c_st_vec_1p','Q_st_vec_1p',...

'dA_vec_1p','A_vec_1p','h_c_st_vec_1p','Re_c_st_vec_1p','rho_c_st_vec_1p','mu
_c_st_vec_1p',...
                    'Cp_c_st_vec_1p','Pr_c_st_vec_1p','dA_back_vec_1p',...
                    ...
                    'P_drp_clnjckt','T_rs_clngjckt','i_regen_strt',...
                    'fdfctr_Bndr','hg_kd_Bndr','tlt_hxfr_rt_Bndr');
            end

    else
        load(rgnrsltsflnm);
        fprintf('RL10 Regen Loaded:\n')
%         fprintf('regen_RL10_2 Total Surface Area = %0.5f
[m^2]\n',ar_conv(A_vec(1)-A_vec_1p(end),'in22m2'));
    end

    % Values to pass out
    RL10rgn.x_plt_1p = x_RL10_plt_en_1p';    % [in]
    RL10rgn.x_plt_2p = x_RL10_plt_en';       % [in]
    RL10rgn.SA_1p = A_vec_1p;                % [in^2]
    RL10rgn.SA_2p = A_vec;                   % [in^2]
    RL10rgn.dA_1p = dA_vec_1p;               % [in^2]
    RL10rgn.dA_2p = dA_vec;                  % [in^2]
    RL10rgn.dA_bck_1p = dA_back_vec_1p;      % [in^2]
    RL10rgn.dA_bck_2p = dA_back_vec;         % [in^2]
    RL10rgn.T_g_1p = T_g_st_vec_1p;          % [R]
    RL10rgn.T_g_2p = T_g_st_vec;             % [R]
    RL10rgn.T_aw_1p = T_aw_st_vec_1p;        % [R]
    RL10rgn.T_aw_2p = T_aw_st_vec;           % [R]
    RL10rgn.T_wg_1p = T_wg_vec_1p;           % [R]
    RL10rgn.T_wg_2p = T_wg_vec;              % [R]
    RL10rgn.T_wc_1p = T_wc_st_vec_1p;        % [R]
    RL10rgn.T_wc_2p = T_wc_st_vec;           % [R]
    RL10rgn.T_c_1p = T_c_st_vec_1p;          % [R]
    RL10rgn.T_c_2p = T_c_st_vec;             % [R]
    RL10rgn.P_c_1p = Pclnt_vec_1p;           % [psia]
    RL10rgn.P_c_2p = Pclnt_vec;              % [psia]
    RL10rgn.Q_1p = Q_st_vec_1p;              % [Btu/s]
    RL10rgn.Q_2p = Q_st_vec;                 % [Btu/s]
    RL10rgn.hg_1p = hg_st_vec_1p;            % [Btu/(s*in^2*R)]
    RL10rgn.hg_2p = hg_st_vec;               % [Btu/(s*in^2*R)]
    RL10rgn.q_1p = q_g_st_vec_1p;            % [Btu/(s*in^2*R)]
    RL10rgn.q_2p = q_g_st_vec;               % [Btu/(s*in^2*R)]
    RL10rgn.Dhc_1p = D_h_c_st_vec_1p;        % [in]
```

```
    RL10rgn.Dhc_2p = D_h_c_st_vec;          % [in]
    RL10rgn.hc_1p = h_c_st_vec_1p;          % [Btu/(s*in^2*R)]
    RL10rgn.hc_2p = h_c_st_vec;             % [Btu/(s*in^2*R)]
    RL10rgn.Re_c_1p = Re_c_st_vec_1p;       % []
    RL10rgn.Re_c_2p = Re_c_st_vec;          % []
    RL10rgn.Cp_c_1p = Cp_c_st_vec_1p;       % [Btu/(lbm*R)]
    RL10rgn.Cp_c_2p = Cp_c_st_vec;          % [Btu/(lbm*R)]
    RL10rgn.Pr_c_1p = Pr_c_st_vec_1p;       % [Btu/(lbm*R)]
    RL10rgn.Pr_c_2p = Pr_c_st_vec;          % [Btu/(lbm*R)]
    RL10rgn.rho_c_1p = rho_c_st_vec_1p;     % [lbm/in^3]
    RL10rgn.rho_c_2p = rho_c_st_vec;        % [lbm/in^3]
    RL10rgn.v_c_1p = v_c_st_vec_1p;         % [in/s]
    RL10rgn.v_c_2p = v_c_st_vec;            % [in/s]
    RL10rgn.mu_c_1p = mu_c_st_vec_1p;       % [lbm/(in*s)]
    RL10rgn.mu_c_2p = mu_c_st_vec;          % [lbm/(in*s)]
    RL10rgn.Binder.Q = tlt_hxfr_rt_Bndr;    % [Btu/s]
    RL10rgn.Binder.delT = T_rs_clngjckt;    % [R]
    RL10rgn.Binder.delP = P_drp_clnjckt;    % [psia]

    if plt_regen_RL10

        f10 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1 =
plot(x_RL10_plt_en_1p,temp_R2F(T_g_st_vec_1p(T_g_st_vec_1p~=0)),'color',[0.63
50, 0.0780, 0.1840]);
        p2 = plot(x_RL10_plt_en,temp_R2F(T_g_st_vec),'color',[0.6350, 0.0780,
0.1840]);
        p3 =
plot(x_RL10_plt_en_1p,temp_R2F(T_wg_vec_1p(T_wg_vec_1p~=0)),'color',[0.8500,
0.3250, 0.0980]);
        p4 = plot(x_RL10_plt_en,temp_R2F(T_wg_vec),'color',[0.8500, 0.3250,
0.0980]);
        p5 =
plot(x_RL10_plt_en_1p,temp_R2F(T_wc_st_vec_1p(T_wc_st_vec_1p~=0)),'color',[0.
9290, 0.6940, 0.1250]);
        p6 = plot(x_RL10_plt_en,temp_R2F(T_wc_st_vec),'color',[0.9290,
0.6940, 0.1250]);
        p7 =
plot(x_RL10_plt_en_1p,temp_R2F(T_c_st_vec_1p(T_c_st_vec_1p~=0)),'color',[0,
0.4470, 0.7410]);
        p8 = plot(x_RL10_plt_en,temp_R2F(T_c_st_vec),'color',[0, 0.4470,
0.7410]);
        p9 = plot([-20,50],temp_R2F([75 75]),'m--');
        p10 = plot([-20,50],temp_R2F([50 50]),'k--');
        p11 = plot([-20,50],temp_R2F([500 500]),'k--');
        xlabel('Distance from Throat [in]')
        ylabel('T [^oF]')
        title('Temperature Profiles')
        legend([p1 p3 p5 p7 p9 p10],'Gas Temperature','Gas-Side Wall
Temperature','Coolant-Side Wall Temperature','Hydrogen Fuel Coolant
Temperature','H_2 Critical Temperature','H_2 Property Temperature
Bounds','location','northeast')

        f11 = figure(11);
        hold on
```

```
        % plot(x_RL10_plt_en,squeeze(hg_Bartz2eng(h_g_1_RL10(2,2,:,1))))

plot(x_RL10_plt_en_1p,cnvctvHtXfer_conv(h_c_st_vec_1p(h_c_st_vec_1p~=0),'eng2
SI')/1e6,'color',[0, 0.4470, 0.7410])

plot(x_RL10_plt_en,cnvctvHtXfer_conv(h_c_st_vec,'eng2SI')/1e6,'color',[0,
0.4470, 0.7410])
        xlabel('Distance from Throat [in]')
        ylabel('h_c [MW/(m^2*K)]')
%         title('Liquid-Side Convective Heat Transfer Coefficient (Sieder-
Tate)')

        f12 = figure(12);
        hold on
%         plot(x_RL10_plt_en_1p,q_g_st_vec_1p(q_g_st_vec_1p~=0),'color',[0,
0.4470, 0.7410])
%         plot(x_RL10_plt_en,q_g_st_vec,'color',[0, 0.4470, 0.7410])

plot(x_RL10_plt_en_1p,htFlx_conv(q_g_st_vec_1p(q_g_st_vec_1p~=0),'eng2SI')/1e
6,'color',[0, 0.4470, 0.7410])
        plot(x_RL10_plt_en,htFlx_conv(q_g_st_vec,'eng2SI')/1e6,'color',[0,
0.4470, 0.7410])
        xlabel('Distance from Throat [in]')
%         ylabel('q [Btu/(s*in^2)]')
        ylabel('q [MW/m^2]')
%         title('Heat Flux')
%         grid on

        f13 = figure(13);
        hold on

plot(x_RL10_plt_en_1p,cnvctvHtXfer_conv(hg_st_vec_1p(hg_st_vec_1p~=0),'eng2SI
'),'color',[0, 0.4470, 0.7410])
        plot(x_RL10_plt_en,cnvctvHtXfer_conv(hg_st_vec,'eng2SI'),'color',[0,
0.4470, 0.7410])
%         plot(x_RL10_plt_en,hg_st_vec,'color',[0, 0.4470, 0.7410])
        xlabel('Distance from Throat [in]')
        ylabel('h_g [W/(m^2*K)]')
%         ylabel('h_g [Btu/(in^2*s*^oR)]')
        ylim([0 42000])
%         title('Convective Heat Transfer Coefficient')

        f14 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        yyaxis left
        plot(x_RL10_plt_en_1p,Pclnt_vec_1p(Pclnt_vec_1p~=0),'color',[0,
0.4470, 0.7410])
        plot(x_RL10_plt_en,Pclnt_vec,'color',[0, 0.4470,
0.7410],'linestyle','-')
        ylabel('P [psia]')
        ylim([750 1050])
        ax = gca;
        ax.YColor = [0, 0.4470, 0.7410];
        yyaxis right
        plot(x_RL10_plt_en_1p,f_D_vec_1p(f_D_vec_1p~=0),'color',[0.8500,
0.3250, 0.0980])
```

```
        plot(x_RL10_plt_en,f_D_vec,'color',[0.8500, 0.3250,
0.0980],'linestyle','-')
        xlabel('Distance from Throat [in]')
        ylabel('f_D')
%         title('Regen Cooling Jacket Pressure')
        ax = gca;
        ax.YColor = [0.8500, 0.3250, 0.0980];

        f15 = figure(15);
        set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 0.6, 0.6])
        left_color = [0, 0.4470, 0.7410];
        right_color = [0 .5 .1];
        set(f15,'defaultAxesColorOrder',[left_color; right_color]);
        hold on
        yyaxis left
        p2 = plot(x_RL10_plt_en,D_h_c_st_vec,'color',left_color);
        p1 =
plot(x_RL10_plt_en_1p,D_h_c_st_vec_1p(D_h_c_st_vec_1p~=0),'color',left_color,
'linestyle','--','linewidth',2);
        ylabel('D_{h,c} [in]')
        yyaxis right
%         p3 =
plot(x_RL10_plt_en_1p,v_c_st_vec_1p(v_c_st_vec_1p~=0)/12,'color',right_color,
'linestyle','--','linewidth',2);
%         p4 =
plot(x_RL10_plt_en,v_c_st_vec/12,'color',right_color,'linestyle','-');
        p3 =
plot(x_RL10_plt_en_1p,vel_conv(v_c_st_vec_1p(v_c_st_vec_1p~=0),'engin2SI'),'c
olor',right_color,'linestyle','--','linewidth',2);
        p4 =
plot(x_RL10_plt_en,vel_conv(v_c_st_vec,'engin2SI'),'color',right_color,'lines
tyle','-');
%         ylabel('|v| [ft/s]')
%         ylabel('v_c [ft/s]')
        ylabel('v_c [m/s]')
        lgnd(1) = plot(nan, nan, 'k--');
        lgnd(2) = plot(nan, nan, 'k-');
        legend(lgnd,'Short Tubes','Long Tubes')
        xlabel('Distance from Throat [in]')
%         title('Hydraulic Diameter & Coolant Velocity')

        f16 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p3 =
plot(x_RL10_plt_en_1p,temp_R2F(T_wg_vec_1p(T_wg_vec_1p~=0)),'color',[0.8500,
0.3250, 0.0980]);
        p4 = plot(x_RL10_plt_en,temp_R2F(T_wg_vec),'color',[0.8500, 0.3250,
0.0980]);
        p5 =
plot(x_RL10_plt_en_1p,temp_R2F(T_wc_st_vec_1p(T_wc_st_vec_1p~=0)),'color',[0.
9290, 0.6940, 0.1250]);
        p6 = plot(x_RL10_plt_en,temp_R2F(T_wc_st_vec),'color',[0.9290,
0.6940, 0.1250]);
        p7 =
plot(x_RL10_plt_en_1p,temp_R2F(T_c_st_vec_1p(T_c_st_vec_1p~=0)),'color',[0,
0.4470, 0.7410]);
```

```
        p8 = plot(x_RL10_plt_en,temp_R2F(T_c_st_vec),'color',[0, 0.4470,
0.7410]);
        p9 = plot([-20,50],temp_R2F([75 75]),'m--');
        p10 = plot([-20,50],temp_R2F([50 50]),'k--');
        p11 = plot([-20,50],temp_R2F([500 500]),'k--');
        xlabel('Distance from Throat [in]')
        ylabel('T [^oF]')
        title('Temperature Profiles')
        legend([p3 p5 p7 p9 p10],'Gas-Side Wall Temperature','Coolant-Side
Wall Temperature','Hydrogen Fuel Coolant Temperature','H_2 Critical
Temperature','H_2 Property Temperature Bounds','location','northeast')

        f17 = figure('units','normalized','outerposition',[0.1 0.2 0.6 0.6]);
        hold on
        p7 =
plot(x_RL10_plt_en_1p,temp_R2F(T_c_st_vec_1p(T_c_st_vec_1p~=0)),'color',[0,
0.4470, 0.7410]);
        p8 = plot(x_RL10_plt_en,temp_R2F(T_c_st_vec),'color',[0, 0.4470,
0.7410]);
        p9 = plot([-20,50],temp_R2F([75 75]),'m--');
        p10 = plot([-20,50],temp_R2F([50 50]),'k--');
        p11 = plot([-20,50],temp_R2F([500 500]),'k--');
        xlabel('Distance from Throat [in]')
        ylabel('T [^oF]')
        title('Temperature Profiles')
        legend([p7 p9 p10],'Hydrogen Fuel Coolant Temperature','H_2 Critical
Temperature','H_2 Property Temperature Bounds','location','northeast')
        ylim([-450 50])

        f18 = figure('units','normalized','outerposition',[0.1 0.2 0.6 0.6]);
        hold on
        p7 =
plot(x_RL10_plt_en_1p,[0;Q_st_vec_1p(Q_st_vec_1p~=0)],'color',[0, 0.4470,
0.7410]);
%         p7 = plot(x_RL10_plt_en_1p,Q_st_vec_1p(Q_st_vec_1p~=0),'color',[0,
0.4470, 0.7410]);
        p8 = plot(x_RL10_plt_en,Q_st_vec,'color',[0, 0.4470, 0.7410]);
        p9 = plot([-20 50],[tlt_hxfr_rt_Bndr tlt_hxfr_rt_Bndr],'linestyle','-
-','color','k');
        xlabel('Distance from Throat [in]')
        ylabel('Q [Btu/s]')
%         title('Cumulative Heat Transfer Rate')
        legend([p9],sprintf('Binder Total Heat Transfer Rate = %i
[Btu/s]',tlt_hxfr_rt_Bndr),'location','northeast')

        f19 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p3 =
plot(x_RL10_plt_en_1p,temp_conv(T_wg_vec_1p(T_wg_vec_1p~=0),'R2K'),'color',[0
.8500, 0.3250, 0.0980]);
        p4 = plot(x_RL10_plt_en,temp_conv(T_wg_vec,'R2K'),'color',[0.8500,
0.3250, 0.0980]);
        p5 =
plot(x_RL10_plt_en_1p,temp_conv(T_wc_st_vec_1p(T_wc_st_vec_1p~=0),'R2K'),'col
or',[0.9290, 0.6940, 0.1250]);
```

```
        p6 = plot(x_RL10_plt_en,temp_conv(T_wc_st_vec,'R2K'),'color',[0.9290,
0.6940, 0.1250]);
        p1 =
plot(x_RL10_plt_en_1p,temp_conv(T_c_st_vec_1p(T_c_st_vec_1p~=0),'R2K'),'color
',[0, 0.4470, 0.7410]);
        p2 = plot(x_RL10_plt_en,temp_conv(T_c_st_vec,'R2K'),'color',[0,
0.4470, 0.7410]);
        xlabel('Distance from Throat [in]')
        ylabel('T [K]')
%         title('Wall Temperature Profiles')
        legend([p3 p5 p1],'Gas-Side Wall Temperature','Coolant-Side Wall
Temperature','Coolant Temperature','location','northeast')

        f20 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        subplot(1,2,1)
        hold on
        p2 = plot(x_RL10_plt_en,dA_vec,'color',[0, 0.4470,
0.7410],'linestyle','-');
        p1 = plot(x_RL10_plt_en_1p,[dA_vec_1p(dA_vec_1p~=0);0],'color',[0, 0,
0],'linestyle','--');
        xlabel('Distance from Throat [in]')
        ylabel('dA [in^2]')
        title('Incremental Area')
        legend([p1 p2],'1st Pass','2nd Pass','location','northeast')

        subplot(1,2,2)
        hold on
%         p3 = plot(x_RL10_plt_en,A_vec+A_vec_1p,'color',[0.8500, 0.3250,
0.0980]);
        p2 = plot(x_RL10_plt_en,A_vec,'color',[0, 0.4470,
0.7410],'linestyle','-');
        p1 = plot(x_RL10_plt_en_1p,[0;A_vec_1p(A_vec_1p~=0)],'color',[0, 0,
0],'linestyle','--');
        xlabel('Distance from Throat [in]')
        ylabel('A [in^2]')
        title('Cumulative Area')
        legend([p1 p2],'1st Pass','2nd Pass','location','northeast')

        f21 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1 =
plot(x_RL10_plt_en_1p,temp_R2F(T_g_st_vec_1p(T_g_st_vec_1p~=0)),'color',[0.63
50, 0.0780, 0.1840]);
        p2 = plot(x_RL10_plt_en,temp_R2F(T_g_st_vec),'color',[0.6350, 0.0780,
0.1840]);
        p3 =
plot(x_RL10_plt_en_1p,temp_R2F(T_aw_st_vec_1p(T_aw_st_vec_1p~=0)),'color',[0.
4940, 0.1840, 0.5560]);
        p4 = plot(x_RL10_plt_en,temp_R2F(T_aw_st_vec),'color',[0.4940,
0.1840, 0.5560]);
        p5 =
plot(x_RL10_plt_en_1p,temp_R2F(T_wg_vec_1p(T_wg_vec_1p~=0)),'color',[0.8500,
0.3250, 0.0980]);
        p6 = plot(x_RL10_plt_en,temp_R2F(T_wg_vec),'color',[0.8500, 0.3250,
0.0980]);
```

```
        p7 =
plot(x_RL10_plt_en_1p,temp_R2F(T_wc_st_vec_1p(T_wc_st_vec_1p~=0)),'color',[0.
9290, 0.6940, 0.1250]);
        p8 = plot(x_RL10_plt_en,temp_R2F(T_wc_st_vec),'color',[0.9290,
0.6940, 0.1250]);
        p9 =
plot(x_RL10_plt_en_1p,temp_R2F(T_c_st_vec_1p(T_c_st_vec_1p~=0)),'color',[0,
0.4470, 0.7410]);
        p10 = plot(x_RL10_plt_en,temp_R2F(T_c_st_vec),'color',[0, 0.4470,
0.7410]);
        xlabel('Distance from Throat [in]')
        ylabel('T [^oF]')
        title('Temperature Profiles')
        legend([p1 p3 p5 p7 p9],'Gas Temperature','Adiabatic Gas-side Wall
Temperature','Gas-side Wall Temperature','Coolant-side Wall
Temperature','Hydrogen Fuel Coolant Temperature','location','northeast')

%         f22 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
%         hold on
%         p1 =
plot(x_RL10_plt_en_1p,temp_R2F(T_g_st_vec_1p(T_g_st_vec_1p~=0)),'color',[0,
0.4470, 0.7410]);
%         p2 = plot(x_RL10_plt_en,temp_R2F(T_g_st_vec),'color',[0, 0.4470,
0.7410]);
%         p3 =
plot(x_RL10_plt_en_1p,temp_R2F(T_aw_st_vec_1p(T_aw_st_vec_1p~=0)),'color',[0.
8500, 0.3250, 0.0980]);
%         p4 = plot(x_RL10_plt_en,temp_R2F(T_aw_st_vec),'color',[0.8500,
0.3250, 0.0980]);
%         p5 =
plot(x_RL10_plt_en_1p,temp_R2F(T_wg_vec_1p(T_wg_vec_1p~=0)),'color',[0.9290,
0.6940, 0.1250]);
%         p6 = plot(x_RL10_plt_en,temp_R2F(T_wg_vec),'color',[0.9290, 0.6940,
0.1250]);
%         xlabel('Distance from Throat [in]')
%         ylabel('T [^oF]')
%         legend([p1 p3 p5],'Gas Temperature','Adiabatic Gas-side Wall
Temperature','Gas-side Wall Temperature','location','northeast')

        f22 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1 =
plot(x_RL10_plt_en_1p,temp_conv(T_g_st_vec_1p(T_g_st_vec_1p~=0),'R2K'),'color
',[0, 0.4470, 0.7410]);
        p2 = plot(x_RL10_plt_en,temp_conv(T_g_st_vec,'R2K'),'color',[0,
0.4470, 0.7410]);
        p3 =
plot(x_RL10_plt_en_1p,temp_conv(T_aw_st_vec_1p(T_aw_st_vec_1p~=0),'R2K'),'col
or',[0.8500, 0.3250, 0.0980]);
        p4 = plot(x_RL10_plt_en,temp_conv(T_aw_st_vec,'R2K'),'color',[0.8500,
0.3250, 0.0980]);
        p5 =
plot(x_RL10_plt_en_1p,temp_conv(T_wg_vec_1p(T_wg_vec_1p~=0),'R2K'),'color',[0
.9290, 0.6940, 0.1250]);
        p6 = plot(x_RL10_plt_en,temp_conv(T_wg_vec,'R2K'),'color',[0.9290,
0.6940, 0.1250]);
```

```
        xlabel('Distance from Throat [in]')
        ylabel('T [K]')
        legend([p1 p3 p5],'Gas Temperature','Adiabatic Gas-side Wall
Temperature','Gas-side Wall Temperature','location','northeast')

        f23 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1 =
plot(x_RL10_plt_en_1p,temp_conv(T_wg_vec_1p(T_wg_vec_1p~=0),'R2K'),'color',[0
, 0.4470, 0.7410]);
        p2 = plot(x_RL10_plt_en,temp_conv(T_wg_vec,'R2K'),'color',[0, 0.4470,
0.7410]);
        p3 =
plot(x_RL10_plt_en_1p,temp_conv(T_wc_st_vec_1p(T_wc_st_vec_1p~=0),'R2K'),'col
or',[0.8500, 0.3250, 0.0980]);
        p4 = plot(x_RL10_plt_en,temp_conv(T_wc_st_vec,'R2K'),'color',[0.8500,
0.3250, 0.0980]);
        p5 =
plot(x_RL10_plt_en_1p,temp_conv(T_c_st_vec_1p(T_c_st_vec_1p~=0),'R2K'),'color
',[0.9290, 0.6940, 0.1250]);
        p6 = plot(x_RL10_plt_en,temp_conv(T_c_st_vec,'R2K'),'color',[0.9290,
0.6940, 0.1250]);
        xlabel('Distance from Throat [in]')
        ylabel('T [K]')
        legend([p1 p3 p5],'Gas-side Wall Temperature','Coolant-side Wall
Temperature','Coolant Temperature','location','northeast')

    end

    if prnt_regen_rslts
        delP_tot = Pclnt_vec_1p(i_regen_strt)-Pclnt_vec(1);
        fprintf('RL10 Regen Results:\n\n')
        fprintf('  Analysis coefficients to match Binder:\n')
        fprintf('    Darcy friction factor multiplier = %0.2f\n',fdfctr_Bndr)
        fprintf('    Bartz convective heat transfer multiplier
= %0.2f\n\n',hg_kd_Bndr)
        fprintf('  Coolant:\n')
        fprintf('    delP = %0.2f [psid] vs. known delP = %0.2f
[psid]\n',delP_tot,P_drp_clnjckt)
        fprintf('    Max P = %0.2f [psia], Min P = %0.2f
[psia]\n',max(Pclnt_vec_1p),min(Pclnt_vec))
        fprintf('    delT = %0.2f [oF] or [oR] vs. known delT = %0.2f [oF] or
[oR]\n',temp_R2F(T_c_st_vec(1))-
temp_R2F(T_c_st_vec_1p(i_regen_strt)),T_rs_clngjckt)
        fprintf('    Max T = %0.2f [K] or %0.2f
[oF]\n',max(temp_conv(T_c_st_vec,'R2K')),max(temp_conv(T_c_st_vec,'R2F')))
        fprintf('    Min T = %0.2f [K] or %0.2f
[oF]\n',min(temp_conv(T_c_st_vec_1p(T_c_st_vec_1p~=0),'R2K')),min(temp_conv(T
_c_st_vec_1p(T_c_st_vec_1p~=0),'R2F')))
        fprintf('  Coolant-side Wall:\n')
        fprintf('    Max T = %0.2f [K] or %0.2f
[oF]\n',max(temp_conv(T_wc_st_vec,'R2K')),max(temp_conv(T_wc_st_vec,'R2F')))
        fprintf('    Min T = %0.2f [K] or %0.2f
[oF]\n',min(temp_conv(T_wc_st_vec_1p(T_wc_st_vec_1p~=0),'R2K')),min(temp_conv
(T_wc_st_vec_1p(T_wc_st_vec_1p~=0),'R2F')))
        fprintf('  Gas-side Wall:\n')
```

```
        fprintf('    Max T = %0.2f [K] or %0.2f
[oF]\n',max(temp_conv(T_wg_vec,'R2K')),max(temp_conv(T_wg_vec,'R2F')))
        fprintf('    Min T = %0.2f [K] or %0.2f
[oF]\n',min(temp_conv(T_wg_vec_1p(T_wg_vec_1p~=0),'R2K')),min(temp_conv(T_wg_
vec_1p(T_wg_vec_1p~=0),'R2F')))
        fprintf('    Min Long Tube T = %0.2f [K] or %0.2f
[oF]\n',min(temp_conv(T_wg_vec,'R2K')),min(temp_conv(T_wg_vec,'R2F')))
        fprintf('  Heat Transfer:\n')
        fprintf('    Max heat flux was %0.3f [Btu/(s*in^2)] at %0.3f [in]
from throat:\n\n',max(q_g_st_vec),x_RL10_plt_en(q_g_st_vec==max(q_g_st_vec)))
    end

end
```

3) *intlz_rgn_vecs_RL10*

```
% Initialzes RL10 regen module parameter vectors (mostly for debugging)

T_c_st_vec_1p = zeros(length(h_g(:)),1);
% T_c_st_vec_1p(i_regen_strt) = temp_F2R(-423);     % liquid hydrogen
temperature [R] (was 1 instead of end)
% T_c_st_vec_1p(i_regen_strt) = 57;     % starting hydrogen temperature per
Binder [R]
T_g_st_vec_1p = zeros(length(T_c_st_vec_1p),1);
T_aw_st_vec_1p = zeros(length(T_c_st_vec_1p),1);
T_wc_st_vec_1p = zeros(length(T_c_st_vec_1p),1);
T_wg_vec_1p = zeros(length(T_c_st_vec_1p),1);
q_g_st_vec_1p = zeros(length(T_c_st_vec_1p),1);
h_c_st_vec_1p = zeros(length(T_c_st_vec_1p),1);
hg_st_vec_1p = zeros(length(T_c_st_vec_1p),1);
Pclnt_vec_1p = zeros(length(T_c_st_vec_1p),1);
f_D_vec_1p = zeros(length(T_c_st_vec_1p),1);
D_h_c_st_vec_1p = zeros(length(T_c_st_vec_1p),1);
v_c_st_vec_1p = zeros(length(T_c_st_vec_1p),1);
Q_st_vec_1p = zeros(length(T_c_st_vec_1p),1);
dA_vec_1p = zeros(length(T_c_st_vec_1p),1);
dA_back_vec_1p = zeros(length(T_c_st_vec_1p),1);
A_vec_1p = zeros(length(T_c_st_vec_1p),1);
Cp_c_st_vec_1p = zeros(length(T_c_st_vec_1p),1);
Pr_c_st_vec_1p = zeros(length(T_c_st_vec_1p),1);
rho_c_st_vec_1p = zeros(length(T_c_st_vec_1p),1);
mu_c_st_vec_1p = zeros(length(T_c_st_vec_1p),1);
Re_c_st_vec_1p = zeros(length(T_c_st_vec_1p),1);

T_c_st_vec = zeros(length(h_g(:)),1);
T_g_st_vec = zeros(length(T_c_st_vec),1);
T_aw_st_vec = zeros(length(T_c_st_vec),1);
T_wc_st_vec = zeros(length(T_c_st_vec),1);
T_wg_vec = zeros(length(T_c_st_vec),1);
q_g_st_vec = zeros(length(T_c_st_vec),1);
h_c_st_vec = zeros(length(T_c_st_vec),1);
hg_st_vec = zeros(length(T_c_st_vec),1);
Pclnt_vec = zeros(length(T_c_st_vec),1);
f_D_vec = zeros(length(T_c_st_vec),1);
D_h_c_st_vec = zeros(length(T_c_st_vec),1);
v_c_st_vec = zeros(length(T_c_st_vec),1);
Q_st_vec = zeros(length(T_c_st_vec),1);
dA_vec = zeros(length(T_c_st_vec),1);
dA_back_vec = zeros(length(T_c_st_vec),1);
A_vec = zeros(length(T_c_st_vec),1);
Cp_c_st_vec = zeros(length(T_c_st_vec),1);
Pr_c_st_vec = zeros(length(T_c_st_vec),1);
rho_c_st_vec = zeros(length(T_c_st_vec),1);
mu_c_st_vec = zeros(length(T_c_st_vec),1);
Re_c_st_vec = zeros(length(T_c_st_vec),1);
```

4) H2props_refprop

```
function [Cp,mu,Pr,rho] = H2props_refprop(T,P)

P = press_conv(P,'eng2SI')/1e3; % [psia] to [kPa]
T = temp_conv(T,'R2K');         % [R] to [K]

rho = refpropm('D','T',T,'P',P,'hydrogen.fld');    % [kg/m^3], density
Cp = refpropm('C','T',T,'P',P,'hydrogen.fld');     % [J/(kg*K)], isobaric
specific heat
mu = refpropm('V','T',T,'P',P,'hydrogen.fld');     % [Pa*s], dynamic viscosity
% k = refpropm('L','T',T,'P',P,'hydrogen.fld');      % [W/(m*K)], thermal
conductivity
Pr = refpropm('^','T',T,'P',P,'hydrogen.fld');     % [], Prandtl number (ratio
of momentum diffusivity to thermal diffusivity)

rho = rho_SI2eng_in3(rho);  % [lbm/in^3]
Cp = cp_SI2eng(Cp);         % [Btu/(lbm*F)] or [Btu/(lbm*R)]
mu = mu_SI2eng(mu);         % [lbm/(in*s)]

end
```

5) *hcH2HuzelHuang*

```
function hc = hcH2HuzelHuang(Cp,mu,Pr,G,d,T_co,T_wc)

    hc = (0.029*Cp*mu^0.2/Pr^(2/3))*(G^0.8/d^0.2)*(T_co/T_wc)^0.55;

end
```

6) *frctnDarcy*

```
function f_D = frctnDarcy(Re)

    f_D = 0.1;
    LHS = 1/sqrt(f_D);
    RHS = 1.93*log(Re*sqrt(f_D))-0.537;
    fDerr = abs(LHS-RHS);
    tol = 0.0001;

    f_D_high = f_D;
    f_D_low = 0.01;

    while fDerr > tol
        f_D = mean([f_D_high,f_D_low]);
        LHS = 1/sqrt(f_D);
        RHS = 1.93*log10(Re*sqrt(f_D))-0.537;
        fDerr = abs(LHS-RHS);
        if LHS-RHS < 0
            f_D_high = f_D;
        elseif LHS-RHS > 0
            f_D_low = f_D;
        end
    end

end
```

7) *asgn_vec_vals_1p_RL10*

```
% Assigns RL10 current iteration values to tube collection vectors for 1st
pass/short tubes

T_g_st_vec_1p(i) = Tg_st;
T_aw_st_vec_1p(i) = T_aw_st;
T_wc_st_vec_1p(i) = T_wc_st;
T_wg_vec_1p(i) = T_wg_st - T_wg_inc;
q_g_st_vec_1p(i) = q_g_st;
h_c_st_vec_1p(i) = h_c_st;
hg_st_vec_1p(i) = hg_st;
D_h_c_st_vec_1p(i) = D_h_c_st;
v_c_st_vec_1p(i) = v_c_st;
f_D_vec_1p(i) = f_D*fdfctr_Bndr;
dA_vec_1p(i) = dA;
dA_back_vec_1p(i) = dA_back;
Cp_c_st_vec_1p(i) = Cp_c_st;
Pr_c_st_vec_1p(i) = Pr_c_st;
rho_c_st_vec_1p(i) = rho_c_st;
mu_c_st_vec_1p(i) = mu_c_st;
Re_c_st_vec_1p(i) = Re_c_st;
```

8) *asgn_vec_vals_RL10*

```
% Assigns RL10 current iteration values to tube collection vectors for 2nd
pass/long tubes

T_g_st_vec(i) = Tg_st;
T_aw_st_vec(i) = T_aw_st;
T_wc_st_vec(i) = T_wc_st;
T_wg_vec(i) = T_wg_st - T_wg_inc;
q_g_st_vec(i) = q_g_st;
h_c_st_vec(i) = h_c_st;
hg_st_vec(i) = hg_st;
D_h_c_st_vec(i) = D_h_c_st;
v_c_st_vec(i) = v_c_st;
f_D_vec(i) = f_D*fdfctr_Bndr;
dA_vec(i) = dA;
dA_back_vec(i) = dA_back;
Cp_c_st_vec(i) = Cp_c_st;
Pr_c_st_vec(i) = Pr_c_st;
rho_c_st_vec(i) = rho_c_st;
mu_c_st_vec(i) = mu_c_st;
Re_c_st_vec(i) = Re_c_st;
```

K) frmtrgninpts_RDE

```
% Format regen inputs for RDE comparable to RL10

h_g_rgn_RDE = squeeze(h_g_1_RDE(2,2,:,1));
T_g_rgn_RDE = squeeze(T_g_RDE(2,2,:));
Pr_g_rgn_RDE = squeeze(Pr_RDE(2,2,:));
gamma_rgn_RDE = squeeze(gamma_RDE(2,2,:));
M_rgn_RDE = squeeze(M_RDE(2,2,:));
cstar_rgn_RDE = squeeze(cstar_RDE(2,2,:));
T_g_chmbr_rgn_RDE = T_g_RDE(2,2,1);
gamma_chmbr_rgn_RDE = gamma_RDE(2,2,1);
mu_chmbr_rgn_RDE = mu_RDE(2,2,1);
C_p_chmbr_rgn_RDE = C_p_RDE(2,2,1);
Pr_chmbr_rgn_RDE = Pr_RDE(2,2,1);
```

L) regen_RDE

```
function [mdot_c,T_c_mix,v_c_mix,P_c_mix] =
regen_RDE(calcregen,plt_regen,prnt_regen_rslts,rgnrsltsflnm,Chf,mdot_o_frctn,
numcmbstrstnts,x_shrtb_st,...
        ...
        x,x_RDE_plt_en,Din,Dout,x_CEA,r_i_CEA,r_o_CEA,...
        ...
        x_RL10_CEA,A_RL10_CEA,Per_RL10_CEA,A_t,h_g,T_g,Pr_g,gamma,M,cstar,...
        T_g_chmbr,gamma_chmbr,mu_chmbr,C_p_chmbr,Pr_g_chmbr,...
        ...
        P_c,eps_tot,...
        ...
        fdfctr_Bndr,hg_kd_Bndr,RL10rgn)

    % Note "_st" suffix refers to Sieder-Tate convective heat transfer,
    % however, a different relation is used for hydrogen as recommended by
    % Huzel and Huang
    %
    % r_i_CEA & r_o_CEA
    %   1st entry is single representative value in chamber
    %   2nd entry is "throat" (end of detonation chamber)

    xxxx = 'placeholder';
    thrtcrvtr = len_conv(Dout(1)-Din(1),'m2engin');    %   defined for no
curvature (R term = 1) [in]

    if calcregen

        fprintf('RDE Regen Started:\n')

        %% Analysis Preparation

        %
%         flvCEA = length(x)-length(x_CEA);

        % Index where aerospike starts
        i_arspk = find(Din>0,1,'last');

        % Constants
        g_eng = 32.2; % gravitational acceleration [ft/s^2]

        % Regenerative Cooling Jacket Channel
        i_regen_strt = find(x_RDE_plt_en>=x_shrtb_st,1,'first');
%         i_regen_strt = length(x_plt_en);
%         x_RL10_plt_en_1p = x_plt_en(i_regen_strt:end);     %[in]
%         x_RDE_plt_en_1p = x_plt_en(1:i_regen_strt);     %[in]
        x_RDE_plt_en_1p = x_RDE_plt_en(i_regen_strt:end);     %[in]
        num_chnls_o = 180;
        num_chnls_i = 180;
        dvsr_Tc_inc_prchnl = 1;

        % Create vectors for plotting/analysis
```

```
        intlz_rgn_vecs_RDE_2

        % Shared Properties w/RL10
%         k_w_st = k_conv(400,'SI2eng');          % copper thermal
conductivity [BTU/(s*in*F)]
        k_w_st = k_conv(16.3,'SI2eng');          % 347 stainless steel
thermal conductivity [Btu/(s*in*oF)]
(https://www.sandmeyersteel.com/images/Alloy347-SpecSheet.pdf)
        w_th_st = len_conv(0.3302/1000,'m2engin'); % average RL10 hot wall
thickness [in]
        mdot_c = 5.973;                         % coolant mass flow rate
[lbm/s]
%         mdot_o_frctn = 0.276;
%         mdot_o_frctn = 0.89475;                  % isobaric hydrogen props
%         mdot_o_frctn = 0.88446;                  % bivariate hydrogen props
%         mdot_o_frctn = 0.7378;                  % bivariate hydrogen props,
1000 psia initial coolant
%         mdot_o_frctn = 0.7385;                  % bivariate hydrogen props,
1000 psia & 57 R initial coolant
%         mdot_o_frctn = 0.74168;                  % bivariate hydrogen props,
1000 psia & 57 R initial coolant, local gamma in b_st
%         mdot_o_frctn = 0.74164;                  % bivariate hydrogen props,
1000 psia & 57 R initial coolant, local gamma in b_st, fixed CEA geo
%         mdot_o_frctn = 0.74141;                  % bivariate hydrogen props,
1000 psia & 57 R initial, local gamma in b_st, fixed CEA geo, RL10 min x geo
fix
%         mdot_o_frctn = 0.739443;                  % final
%         mdot_o_frctn = 0.74014;                  % final, fixed inner
circuit h_g calcs, Chf = 1
        mdot_c_o = mdot_c*mdot_o_frctn;
        mdot_c_i = mdot_c-mdot_c_o;
%         mdot_c_o = mdot_c;
%         mdot_c_i = mdot_c;
        P_drp_inj = 62.3;                       % injector pressure drop from
Binder (Table, 2.6.1 pg. 13/184) [psia]
        P_drp_clnjckt_Bndr = 242.1;             % cooling jacket pressure
drop from Binder (Table, 2.4.1 pg. 11/184) [psia]
%         Pclnt_init = P_c+P_drp_inj+P_drp_clnjckt_Bndr;   % initial coolant
pressure\
        Pclnt_init = 1000;  % initial coolant pressure from Binder's pressure
drop results (pg. 138/184) [psia]
        Pclnt_o_1p_vec(i_regen_strt) = Pclnt_init;
        Pclnt_i_vec(i_arspk) = Pclnt_init;
        T_c_st_o_1p_vec(i_regen_strt) = 57;     % starting hydrogen
temperature per Binder [R]
        T_c_st_i_vec(i_arspk) = 57;     % starting hydrogen temperature per
Binder [R]
        T_rs_clngjckt_Bndr = 344.4;              % cooling jacket temperature
rise [oR]
        ttl_hxfr_rt_Bndr = 7994;            % Total heat transfer rate from
Binder [Btu/s]

        % Gas-side wall temperature guess increment
        T_wg_inc =1 ;       % [R]

        % Hydrogen properties
```

```
        H2props = xlsread('LH2_properties.xls');
        T_H2 = H2props(:,1);                % [R]
        rho_H2 = H2props(:,3);              % [lb/in^3]
        Cp_H2 = H2props(:,4);               % [Btu/(lb*R)]
        k_H2 = H2props(:,5);                % [Btu/(in*sec*R)]
        mu_H2 = H2props(:,6);               % [lbm/(in*sec)]

         %% Start of 1st pass/short tubes to the nozzle end/turnaround
manifold

        for i = i_regen_strt:length(eps_tot)

            fprintf('At RDE Axial Station %i\n',i)

            P_H2_o_i = Pclnt_o_1p_vec(i);
            T_H2_o_i = T_c_st_o_1p_vec(i);

            if i-numcmbstrstnts+1 > 0
                hg_kd_RDE = 1;      % gas-side convective heat transfer
coefficient knockdown
            elseif i-numcmbstrstnts+1 <= 0

%                 hg_kd_RDE = 0.33;
%                 hg_kd_RDE = 0.42;
                hg_kd_RDE = Chf;        % for comparison
            end

            A_chnl_o =
interp1(x_RL10_CEA,A_RL10_CEA,x_RDE_plt_en(i));       % cooling channel cross
sectional area [in^2]
            Per_chnl_o =
interp1(x_RL10_CEA,Per_RL10_CEA,x_RDE_plt_en(i));   % cooling channel
perimeter [in]

            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
            % Outer Wall
            % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            % Iteration intital guessed coolant temperature [R]
            T_c_temp_st = 36;           % eqvlnt: -423.7 F, -253.2 C, 20 K

            % Iteration inital guessed gas-side wall temp [R]
            T_wg_st = 500;  % [R]

            % Energy Balance
            while T_c_temp_st < T_c_st_o_1p_vec(i)

                % Gas calculations
                Tg_st = temp_conv(T_g(i),'K2R');            % gas temp [R]
                Tc_ns_st = temp_conv(T_g_chmbr,'K2R');      % gas CC stag temp
[R]
```

```
                r_st = Pr_g(i)^(1/3);               % recovery factor (assume
turbulent freestream boundary layer)
                T_r_st = Tg_st*(1 + (gamma(i)-1)/2*M(i)^2*r_st);     %
recovery temperature [R]
                T_aw_st = Tc_ns_st*((1 + (gamma(i)-1)/2*M(i)^2*r_st)/(1 +
(gamma(i)-1)/2*M(i)^2));  % adiabatic wall temperature [R]

                % Gas convective heat transfer coefficient [BTU/(in^2*s*oF)]
%                 b_st = (1+(gamma_chmbr-1)/2*M(i)^2);
                b_st = (1+(gamma(i)-1)/2*M(i)^2);
                sig_st = (((1/2)*(T_wg_st/Tc_ns_st)*b_st + 1/2)^0.68 *
b_st^0.12)^(-1);
                AtA = eps_tot(i)^(-1);
                R = thrtcrvtr;
                a_st = (0.026/len_conv(Dout(1)-Din(1),'m2engin')^0.2)*...

(mu_SI2eng(mu_chmbr)^0.2*cp_SI2eng(C_p_chmbr)/Pr_g_chmbr^0.6)...
                    *(P_c*g_eng/vel_ms2fs(cstar(i)))^0.8*(len_conv(Dout(1)-
Din(1),'m2engin')/R)^0.1;
                hg_st = a_st*(AtA)^(0.9)*sig_st;
%                 hg_st = hg_st*hg_kd_Bndr*hg_kd_RDE;                %
[Btu/(s*in^2*R)] or [Btu/(s*in^2*F)]
%                 hg_st_SI = cnvctvHtXfer_conv(hg_st,'eng2SI');      %
[W/(m^2*K)]

                % Heat Flux [BTU/(in^2*s)]  (used [R] to calculate)
                q_g_st = hg_st*(T_aw_st-T_wg_st)*hg_kd_Bndr*hg_kd_RDE;

                % Coolant-side wall temp [R]
%                     T_wc_st = T_wg_st - (w_th_st/k_w_st)*q_g_st;
                T_wc_st = T_wg_st - temp_conv((w_th_st/k_w_st)*q_g_st,'F2R');

                % Local hydrogen properties
                %   Cp_c_st [Btu/(lbm*R)] or [Btu/(lbm*F)]
                %   mu_c_st [lbm/(in*s)]
                %   k_c_st [Btu/(in*s*R)] or [Btu/(in*s*F)]
                %   Pr_c_st
                %   rho_c_st [lbm/in^3]

%                 [Cp_c_st,mu_c_st,Pr_c_st,rho_c_st] =
intrpH2lcl(T_H2,Cp_H2,mu_H2,k_H2,rho_H2,T_c_st_o_1p_vec(i));
                [Cp_c_st,mu_c_st,Pr_c_st,rho_c_st] =
H2props_refprop(T_H2_o_i,P_H2_o_i);
%                 [Cp_c_st,mu_c_st,Pr_c_st,rho_c_st] =
intrpH2lcl(T_H2,Cp_H2,mu_H2,k_H2,rho_H2,T_c_temp_st);
%                     D_h_c_st = 4*A_RL10_CEA(i)/Per_CEA(i);       %
hydraulic diameter [in]
                D_h_c_st = 4*A_chnl_o/Per_chnl_o;       % hydraulic diameter
[in]
                G_st = mdot_c_o/(pi*D_h_c_st^2/4);        % coolant weight
flow rate per unit area [lbf/(in^2*s)]
                v_c_st = G_st/rho_c_st/num_chnls_o;       % coolant velocity
in single passage [in/s]
                Re_c_st = rho_c_st*v_c_st*D_h_c_st/mu_c_st;
```

```
                % Coolant convective heat transfer coefficient (Sieder-
Tate) % [BTU/(in^2*s*R)]
                h_c_st =
hcH2HuzelHuang(Cp_c_st,mu_c_st,Pr_c_st,G_st,D_h_c_st,T_c_st_o_1p_vec(i),T_wc_
st);
%                 h_c_st =
hcSiederTate(Cp_c_st,mu_c_st,Pr_c_st,G_st,D_h_c_st,T_c_temp_st,T_wc_st);

                % Iteration calculated coolant temperature [R]
                T_c_temp_st = T_wc_st - q_g_st/h_c_st;        % [R]
%                 T_c_temp_st = T_wc_st - temp_F2R(q_g_st/h_c_st);        %
[R]

                % Increment guessed gas-side wall temp [R]
                T_wg_st = T_wg_st + T_wg_inc;

            end

            % Iterate to find Darcy friction factor
            f_D = frctnDarcy(Re_c_st);

            if i ~= length(eps_tot) % was ~= length(h_g_1_RL10(2,2,:,1))

                % Calculate coolant heat absorption
                R_cur = Dout(i)/2;          % [m]
                R_nxt = Dout(i+1)/2;        % [m]
                dh = sqrt((R_cur-R_nxt)^2 + (x(i+1)-x(i))^2);   % incremental
slant height [m]
                dA =
pi*len_conv((R_cur+R_nxt),'m2engin')*len_conv(dh,'m2engin');   % [in^2]
%                 A =
pi*sqrt(eps_tot(i)*A_t*4/pi)*len_conv(dh,'m2engin');   % [in^2]
%                 A = D_h_c_st*len_conv(delx,'SI2engin');   % [in^2]
%                 dQ_st = dA*q_g_st;              % [Btu/s]
                dQ_st = dA*q_g_st/2;              % [Btu/s], add "/2" because
short tubes interspersed w/long tubes
                Q_st_o_1p_vec(i+1) = Q_st_o_1p_vec(i) + dQ_st;
                T_c_st_o_1p_vec(i+1) = T_c_st_o_1p_vec(i) +
dQ_st/(mdot_c_o*Cp_c_st)/dvsr_Tc_inc_prchnl;      % [R]

                % Calculate coolant pressure drop
                dP = press_conv(...
                    fdfctr_Bndr*f_D*...

(rho_eng2SI_in3(rho_c_st)/2)*(vel_conv(v_c_st,'engin2SI')^2/len_conv(D_h_c_st
,'engin2m'))*dh,...
                    'SI2eng');
                Pclnt_o_1p_vec(i+1) = Pclnt_o_1p_vec(i) - dP;
            else
                dA = 0;
                dQ_st = 0;

            end

            if i == i_regen_strt
```

```
                dA_back = 0;
                A_o_1p_vec(i) = dA_back;      % [in^2]
            elseif i > i_regen_strt
                R_cur = Dout(i)/2;           % [m]
                R_prv = Dout(i-1)/2;         % [m]
                dh = sqrt((R_cur-R_prv)^2 + (x(i-1)-x(i))^2);    % incremental
slant height [m]
                dA_back =
pi*len_conv((R_cur+R_prv),'m2engin')*len_conv(dh,'m2engin');    % [in^2]
                A_o_1p_vec(i) = A_o_1p_vec(i-1) + dA_back;      % [in^2]
            end

            eng_per = 2*pi*R_cur;
            chnl_per_tot = num_chnls_o*D_h_c_st*2;  % *2 because short and
long tubes overlap

            asgn_vec_vals_o_1p_RDE


        end

        %% Short-to-Long Tube Transition
        delx = sqrt(((Dout(i)-Dout(i-1))/2)^2 + (x(i)-x(i-1))^2);           %
[m]
        dA = pi*len_conv(Dout(i),'m2engin')*len_conv(delx,'m2engin');    %
[in^2]
        dQ_st = dA*q_g_st;              % [Btu/s]
        Q_st_o_vec(end) = dQ_st + Q_st_o_1p_vec(i);
        T_c_st_o_vec(end) = dQ_st/(mdot_c*Cp_c_st)/dvsr_Tc_inc_prchnl +
T_c_st_o_1p_vec(i);

        % Assume turnaround manifold bend radius is equal to coolant tube
radius
        kb = 1.1; % Bend loss coefficients for a pipe from Babcock & Wilson
Co. (fig. 3 on Thermopedia http://www.thermopedia.com/content/577/)
        Pclnt_o_vec(end) = Pclnt_o_1p_vec(end) -...
            press_conv(...

fdfctr_Bndr*f_D*(rho_eng2SI_in3(rho_c_st)/2)*(vel_conv(v_c_st,'engin2SI')^2/l
en_conv(D_h_c_st,'engin2m'))*pi*D_h_c_st/2 +...

(1/2)*kb*rho_eng2SI_in3(rho_c_st)*vel_conv(v_c_st,'engin2SI')^2,...
            'SI2eng');


        %% End of nozzle to injection plane

        for i = length(eps_tot):-1:1

            fprintf('At RDE Axial Station %i\n',i)

            P_H2_o_i = Pclnt_o_vec(i);
            T_H2_o_i = T_c_st_o_vec(i);
            P_H2_i_i = Pclnt_i_vec(i);
            T_H2_i_i = T_c_st_i_vec(i);
```

```
            if i-numcmbstrstnts+1 > 0
                hg_kd_RDE = 1;      % gas-side convective heat transfer
coefficient knockdown
            elseif i-numcmbstrstnts+1 <= 0
%                 hg_kd_RDE = 0.479;  % conservative (1.5 times greatest
calculated RDE chamber-to-CP throat level heat flux) knockdown from Meyer and
Stevens 2018 papers
%                 hg_kd_RDE = 0.33;
%                 hg_kd_RDE = 0.42;
                hg_kd_RDE = Chf;        % for comparison
            end

            A_chnl_o =
interp1(x_RL10_CEA,A_RL10_CEA,x_RDE_plt_en(i));       % cooling channel cross
sectional area [in^2]
            Per_chnl_o =
interp1(x_RL10_CEA,Per_RL10_CEA,x_RDE_plt_en(i));   % cooling channel
perimeter [in]
            Per_chnl_i = Per_chnl_o*Din(i)/Dout(i);   % cooling channel
perimeter [in]
            A_chnl_i = pi*(Per_chnl_i/(2*pi))^2;       % cooling channel
cross sectional area [in^2]

%             if i > flvCEA+1
%                 x_cur = x_CEA(i);
%                 x_nxt = x_CEA(i-1);
%             else


            if i > i_arspk

                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
                % Outer Wall
                % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                % Iteration intital guessed coolant temperature [R]
                T_c_temp_st = 36;           % eqvlnt: -423.7 F, -253.2 C, 20
K

                % Iteration inital guessed gas-side wall temp [R]
%                 T_wg_st = temp_conv(575,'K2R');                 % eqvlnt:
575.3 F, 301.9 C, 1035 R
%                 T_wg_st = temp_conv(100,'K2R');
                T_wg_st = 500;  % [R]
%                 T_wg_st = 100;  % [R]

                % Energy Balance
                while T_c_temp_st < T_c_st_o_vec(i)

                    % Gas calculations
                    Tg_st = temp_conv(T_g(i),'K2R');           % gas temp [R]
```

```
                    Tc_ns_st = temp_conv(T_g_chmbr,'K2R');     % gas CC stag
temp [R]
                    r_st = Pr_g(i)^(1/3);               % recovery factor
(assume turbulent freestream boundary layer)
                    T_r_st = Tg_st*(1 + (gamma(i)-1)/2*M(i)^2*r_st);     %
recovery temperature [R]
                    T_aw_st = Tc_ns_st*((1 + (gamma(i)-1)/2*M(i)^2*r_st)/(1 +
(gamma(i)-1)/2*M(i)^2));  % adiabatic wall temperature [R]

                    % Gas convective heat transfer coefficient
[BTU/(in^2*s*oF)]
%                     b_st = (1+(gamma_chmbr-1)/2*M(i)^2);
                    b_st = (1+(gamma(i)-1)/2*M(i)^2);
                    sig_st = (((1/2)*(T_wg_st/Tc_ns_st)*b_st + 1/2)^0.68 *
b_st^0.12)^(-1);
                    AtA = eps_tot(i)^(-1);
                    R = thrtcrvtr;
                    a_st = (0.026/len_conv(Dout(1)-Din(1),'m2engin')^0.2)*...

(mu_SI2eng(mu_chmbr)^0.2*cp_SI2eng(C_p_chmbr)/Pr_g_chmbr^0.6)...

*(P_c*g_eng/vel_ms2fs(cstar(i)))^0.8*(len_conv(Dout(1)-
Din(1),'m2engin')/R)^0.1;
                    hg_st = a_st*(AtA)^(0.9)*sig_st;
%                     hg_st = hg_st*hg_kd_Bndr*hg_kd_RDE;                %
[Btu/(s*in^2*R)] or [Btu/(s*in^2*F)]
    %                 hg_st_SI = cnvctvHtXfer_conv(hg_st,'eng2SI');      %
[W/(m^2*K)]

                    % Heat Flux [BTU/(in^2*s)]  (used [R] to calculate)
                    q_g_st = hg_st*(T_aw_st-T_wg_st)*hg_kd_Bndr*hg_kd_RDE;

                    % Coolant-side wall temp [R]
%                     T_wc_st = T_wg_st - (w_th_st/k_w_st)*q_g_st;
                    T_wc_st = T_wg_st -
temp_conv((w_th_st/k_w_st)*q_g_st,'F2R');

                    % Local hydrogen properties
                    %   Cp_c_st [Btu/(lbm*R)] or [Btu/(lbm*F)]
                    %   mu_c_st [lbm/(in*s)]
                    %   k_c_st [Btu/(in*s*R)] or [Btu/(in*s*F)]
                    %   Pr_c_st
                    %   rho_c_st [lbm/in^3]

%                     [Cp_c_st,mu_c_st,Pr_c_st,rho_c_st] =
intrpH2lcl(T_H2,Cp_H2,mu_H2,k_H2,rho_H2,T_c_st_o_vec(i));
                    [Cp_c_st,mu_c_st,Pr_c_st,rho_c_st] =
H2props_refprop(T_H2_o_i,P_H2_o_i);
%                     [Cp_c_st,mu_c_st,Pr_c_st,rho_c_st] =
intrpH2lcl(T_H2,Cp_H2,mu_H2,k_H2,rho_H2,T_c_temp_st);
%                     D_h_c_st = 4*A_RL10_CEA(i)/Per_CEA(i);       %
hydraulic diameter [in]
                    D_h_c_st = 4*A_chnl_o/Per_chnl_o;       % hydraulic
diameter [in]
                    G_st = mdot_c_o/(pi*D_h_c_st^2/4);        % coolant
weight flow rate per unit area [lbf/(in^2*s)]
```

```
                    v_c_st = G_st/rho_c_st/num_chnls_o;       % coolant
velocity in single passage [in/s]
                    Re_c_st = rho_c_st*v_c_st*D_h_c_st/mu_c_st;

                    % Coolant convective heat transfer coefficient (Sieder-
Tate) % [BTU/(in^2*s*R)]
                    h_c_st =
hcH2HuzelHuang(Cp_c_st,mu_c_st,Pr_c_st,G_st,D_h_c_st,T_c_st_o_vec(i),T_wc_st)
;
    %                 h_c_st =
hcSiederTate(Cp_c_st,mu_c_st,Pr_c_st,G_st,D_h_c_st,T_c_temp_st,T_wc_st);

                    % Iteration calculated coolant temperature [R]
                    T_c_temp_st = T_wc_st - q_g_st/h_c_st;        % [R]
    %                 T_c_temp_st = T_wc_st -
temp_F2R(q_g_st/h_c_st);        % [R]

                    % Increment guessed gas-side wall temp [R]
                    T_wg_st = T_wg_st + T_wg_inc;

                end

                % Iterate to find Darcy friction factor
                f_D = frctnDarcy(Re_c_st);

                if i ~= 1 % was ~= length(h_g_1_RL10(2,2,:,1))

                    % Calculate coolant heat absorption
                    R_cur = Dout(i)/2;          % [m]
                    R_nxt = Dout(i-1)/2;        % [m]
                    dh = sqrt((R_cur-R_nxt)^2 + (x(i)-x(i-1))^2);   %
incremental slant height [m]
%                     dh = sqrt((r_o_CEA(i)-r_o_CEA(i-1))^2 + (x_CEA(i)-
x_CEA(i-1))^2);
    %                 R_cur = sqrt(eps_tot(i)*A_t*4/pi);
    %                 R_nxt = sqrt(eps_tot(i-1)*A_t*4/pi);
                    dA =
pi*len_conv((R_cur+R_nxt),'m2engin')*len_conv(dh,'m2engin');   % [in^2]
    %                 A =
pi*sqrt(eps_tot(i)*A_t*4/pi)*len_conv(dh,'m2engin');   % [in^2]
    %                 A = D_h_c_st*len_conv(delx,'SI2engin');   % [in^2]
%                     dQ_st = dA*q_g_st;              % [BTU/s]
                    if i >= i_regen_strt
                        dQ_st = dA*q_g_st/2;            % [Btu/s], add "/2"
because long tubes interspersed w/short tubes in this region
                    else
                        dQ_st = dA*q_g_st;              % [Btu/s]
                    end
                    Q_st_o_vec(i-1) = Q_st_o_vec(i) + dQ_st;
                    T_c_st_o_vec(i-1) = T_c_st_o_vec(i) +
dQ_st/(mdot_c_o*Cp_c_st)/dvsr_Tc_inc_prchnl;      % [R]

                    % Calculate coolant pressure drop
                    dP = press_conv(...
                        fdfctr_Bndr*f_D*...
```

```
(rho_eng2SI_in3(rho_c_st)/2)*(vel_conv(v_c_st,'engin2SI')^2/len_conv(D_h_c_st
,'engin2m'))*dh,...
                        'SI2eng');
                    Pclnt_o_vec(i-1) = Pclnt_o_vec(i) - dP;
                else
                    dA = 0;
                    dQ_st = 0;

                end

%                 if i < length(eps_tot)
%                     A_o_vec(i) = A_o_vec(i+1) + dA;     % [in^2]
%                 end

                if i == length(eps_tot)
                    dA_back = 0;
                    A_o_vec(i) = max(A_o_1p_vec);     % [in^2]
                elseif i < length(eps_tot)
                    R_cur = Dout(i)/2;          % [m]
                    R_prv = Dout(i+1)/2;        % [m]
                    dh = sqrt((R_cur-R_prv)^2 + (x(i+1)-x(i))^2);   %
incremental slant height [m]
                    dA_back =
pi*len_conv((R_cur+R_prv),'m2engin')*len_conv(dh,'m2engin');   % [in^2]
                    A_o_vec(i) = A_o_vec(i+1) + dA_back;     % [in^2]
                end

                eng_per = 2*pi*R_cur;
                if i >= i_regen_strt
                    chnl_per_tot = num_chnls_o*D_h_c_st*2;
                else
                    chnl_per_tot = num_chnls_o*D_h_c_st;
                end

                asgn_vec_vals_o_RDE

            elseif i <= i_arspk
            % i-numcmbstrstnts+1 <= 0

                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
                % Outer Wall
                % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                % Iteration intital guessed coolant temperature [R]
                T_c_temp_st = 36;           % eqvlnt: -423.7 F, -253.2 C, 20
K

                % Iteration inital guessed gas-side wall temp [R]
%                 T_wg_st = 500;  % [R]
                if x_RDE_plt_en(i) > 10
                    T_wg_st = temp_conv(100,'F2R');
                elseif x_RDE_plt_en(i) <= 10 && x_RDE_plt_en(i) > 5
```

```
                    T_wg_st = temp_conv(200,'F2R');
                elseif x_RDE_plt_en(i) <= 5 && x_RDE_plt_en(i) > 1.5
                    T_wg_st = temp_conv(315,'F2R');
                elseif x_RDE_plt_en(i) < 1.5 && x_RDE_plt_en(i) > 0.5
                    T_wg_st = temp_conv(575,'F2R');
                elseif x_RDE_plt_en(i) <= 0.5 && x_RDE_plt_en(i) > 0
                    T_wg_st = temp_conv(850,'F2R');
                elseif x_RDE_plt_en(i) <= 0
                    T_wg_st = temp_conv(100,'F2R');
                end


                % Energy Balance
                while T_c_temp_st < T_c_st_o_vec(i)

                    % Gas calculations
                    Tg_st = temp_conv(T_g(i),'K2R');           % gas temp [R]
                    Tc_ns_st = temp_conv(T_g_chmbr,'K2R');     % gas CC stag
temp [R]
                    r_st = Pr_g(i)^(1/3);               % recovery factor
(assume turbulent freestream boundary layer)
                    T_r_st = Tg_st*(1 + (gamma(i)-1)/2*M(i)^2*r_st);     %
recovery temperature [R]
                    T_aw_st = Tc_ns_st*((1 + (gamma(i)-1)/2*M(i)^2*r_st)/(1 +
(gamma(i)-1)/2*M(i)^2));  % adiabatic wall temperature [R]

                    % Gas convective heat transfer coefficient
[BTU/(in^2*s*oF)]
%                     b_st = (1+(gamma_chmbr-1)/2*M(i)^2);
                    b_st = (1+(gamma(i)-1)/2*M(i)^2);
                    sig_st = (((1/2)*(T_wg_st/Tc_ns_st)*b_st + 1/2)^0.68 *
b_st^0.12)^(-1);
                    AtA = eps_tot(i)^(-1);
                    R = thrtcrvtr;
                    a_st = (0.026/len_conv(Dout(1)-Din(1),'m2engin')^0.2)*...

(mu_SI2eng(mu_chmbr)^0.2*cp_SI2eng(C_p_chmbr)/Pr_g_chmbr^0.6)...

*(P_c*g_eng/vel_ms2fs(cstar(i)))^0.8*(len_conv(Dout(1)-
Din(1),'m2engin')/R)^0.1;
                    hg_st = a_st*(AtA)^(0.9)*sig_st;
%                     hg_st = hg_st*hg_kd_Bndr*hg_kd_RDE;                %
[Btu/(s*in^2*R)] or [Btu/(s*in^2*F)]
    %                 hg_st_SI = cnvctvHtXfer_conv(hg_st,'eng2SI');      %
[W/(m^2*K)]

                    % Heat Flux [BTU/(in^2*s)]  (used [R] to calculate)
                    q_g_st = hg_st*(T_aw_st-T_wg_st)*hg_kd_Bndr*hg_kd_RDE;

                    % Coolant-side wall temp [R]
%                     T_wc_st = T_wg_st - (w_th_st/k_w_st)*q_g_st;
                    T_wc_st = T_wg_st -
temp_conv((w_th_st/k_w_st)*q_g_st,'F2R');

                    % Local hydrogen properties
```

```
                    %   Cp_c_st [Btu/(lbm*R)] or [Btu/(lbm*F)]
                    %   mu_c_st [lbm/(in*s)]
                    %   k_c_st [Btu/(in*s*R)] or [Btu/(in*s*F)]
                    %   Pr_c_st
                    %   rho_c_st [lbm/in^3]

%                     [Cp_c_st,mu_c_st,Pr_c_st,rho_c_st] =
intrpH2lcl(T_H2,Cp_H2,mu_H2,k_H2,rho_H2,T_c_st_o_vec(i));
                    [Cp_c_st,mu_c_st,Pr_c_st,rho_c_st] =
H2props_refprop(T_H2_o_i,P_H2_o_i);
%                   [Cp_c_st,mu_c_st,Pr_c_st,rho_c_st] =
intrpH2lcl(T_H2,Cp_H2,mu_H2,k_H2,rho_H2,T_c_temp_st);
%                     D_h_c_st = 4*A_RL10_CEA(i)/Per_CEA(i);       %
hydraulic diameter [in]
                    D_h_c_st = 4*A_chnl_o/Per_chnl_o;       % hydraulic
diameter [in]
                    G_st = mdot_c_o/(pi*D_h_c_st^2/4);        % coolant
weight flow rate per unit area [lbf/(in^2*s)]
                    v_c_st = G_st/rho_c_st/num_chnls_o;       % coolant
velocity in single passage [in/s]
                    Re_c_st = rho_c_st*v_c_st*D_h_c_st/mu_c_st;

                    % Coolant convective heat transfer coefficient (Sieder-
Tate) % [BTU/(in^2*s*R)]
                    h_c_st =
hcH2HuzelHuang(Cp_c_st,mu_c_st,Pr_c_st,G_st,D_h_c_st,T_c_st_o_vec(i),T_wc_st)
;
    %                 h_c_st =
hcSiederTate(Cp_c_st,mu_c_st,Pr_c_st,G_st,D_h_c_st,T_c_temp_st,T_wc_st);

                    % Iteration calculated coolant temperature [R]
                    T_c_temp_st = T_wc_st - q_g_st/h_c_st;        % [R]
%                     T_c_temp_st = T_wc_st -
temp_F2R(q_g_st/h_c_st);        % [R]

                    % Increment guessed gas-side wall temp [R]
                    T_wg_st = T_wg_st + T_wg_inc;

                end

                % Iterate to find Darcy friction factor
                f_D = frctnDarcy(Re_c_st);

                if i ~= 1 % was ~= length(h_g_1_RL10(2,2,:,1))

                    % Calculate coolant heat absorption
                    R_cur = Dout(i)/2;          % [m]
                    R_nxt = Dout(i-1)/2;        % [m]
                    dh = sqrt((R_cur-R_nxt)^2 + (x(i)-x(i-1))^2);   %
incremental slant height [m]
%                     dh = sqrt((r_o_CEA(i)-r_o_CEA(i-1))^2 + (x_CEA(i)-
x_CEA(i-1))^2);
%                     R_cur = Dout(i)/2;          % [m]
%                     R_nxt = Dout(i-1)/2;        % [m]
    %                 R_cur = sqrt(eps_tot(i)*A_t*4/pi);
```

```
    %                 R_nxt = sqrt(eps_tot(i-1)*A_t*4/pi);
                  dA =
pi*len_conv((R_cur+R_nxt),'m2engin')*len_conv(dh,'m2engin');   % [in^2]
    %                 A =
pi*sqrt(eps_tot(i)*A_t*4/pi)*len_conv(dh,'m2engin');   % [in^2]
    %                 A = D_h_c_st*len_conv(delx,'SI2engin');   % [in^2]
%                   dQ_st = dA*q_g_st;              % [BTU/s]
                  if i >= i_regen_strt
                      dQ_st = dA*q_g_st/2;            % [Btu/s], add "/2"
because long tubes interspersed w/short tubes in this region
                  else
                      dQ_st = dA*q_g_st;              % [Btu/s]
                  end
                  Q_st_o_vec(i-1) = Q_st_o_vec(i) + dQ_st;
                  T_c_st_o_vec(i-1) = T_c_st_o_vec(i) +
dQ_st/(mdot_c_o*Cp_c_st)/dvsr_Tc_inc_prchnl;      % [R]

                  % Calculate coolant pressure drop
                  dP = press_conv(...
                      fdfctr_Bndr*f_D*...

(rho_eng2SI_in3(rho_c_st)/2)*(vel_conv(v_c_st,'engin2SI')^2/len_conv(D_h_c_st
,'engin2m'))*dh,...
                      'SI2eng');
                  Pclnt_o_vec(i-1) = Pclnt_o_vec(i) - dP;
              else
                  dA = 0;
                  dQ_st = 0;
              end

              if i == length(eps_tot)
                  dA_back = 0;
                  A_o_vec(i) = max(A_o_1p_vec);     % [in^2]
              elseif i < length(eps_tot)
                  R_cur = Dout(i)/2;          % [m]
                  R_prv = Dout(i+1)/2;        % [m]
                  dh = sqrt((R_cur-R_prv)^2 + (x(i+1)-x(i))^2);   %
incremental slant height [m]
                  dA_back =
pi*len_conv((R_cur+R_prv),'m2engin')*len_conv(dh,'m2engin');   % [in^2]
                  A_o_vec(i) = A_o_vec(i+1) + dA_back;     % [in^2]
              end

%               if i < length(eps_tot)
%                   A_o_vec(i) = A_o_vec(i+1) + dA;     % [in^2]
%               end

%               if i == length(eps_tot)
%                   A_o_vec(i) = dA;
%               elseif i < length(eps_tot)
%                   A_o_vec(i) = A_o_vec(i+1) + dA;     % [in^2]
%               end

              eng_per = 2*pi*R_cur;
              chnl_per_tot = num_chnls_o*D_h_c_st;
```

```
                asgn_vec_vals_o_RDE

                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
                % Inner Wall
                % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                % Iteration intital guessed coolant temperature [R]
                T_c_temp_st = 36;

                % Iteration inital guessed gas-side wall temp [R]
%                 T_wg_st = temp_conv(575,'K2R');
%                 T_wg_st = temp_conv(100,'K2R');
%                 T_wg_st = 500;  % [R]
%                 T_wg_st = 100;  % [R]

                if x_RDE_plt_en(i) > 5
                    T_wg_st = temp_conv(100,'F2R');
                elseif x_RDE_plt_en(i) <= 5 && x_RDE_plt_en(i) > 1.5
                    T_wg_st = temp_conv(200,'F2R');
                elseif x_RDE_plt_en(i) <= 1.5 && x_RDE_plt_en(i) > 0.5
                    T_wg_st = temp_conv(475,'F2R');
                elseif x_RDE_plt_en(i) <= 0.5 && x_RDE_plt_en(i) > 0
                    T_wg_st = temp_conv(800,'F2R');
                elseif x_RDE_plt_en(i) <= 0
                    T_wg_st = temp_conv(100,'F2R');
                end

                % Energy Balance
                while T_c_temp_st < T_c_st_i_vec(i)

%                     % Gas calculations
%                     Tg_st = temp_conv(T_g(i),'K2R');           % gas temp
[R]
%                     Tc_ns_st = temp_conv(T_g_chmbr,'K2R');     % gas CC
stag temp [R]
%                     r_st = Pr_g(i)^(1/3);               % recovery factor
(assume turbulent freestream boundary layer)
%                     T_r_st = Tg_st*(1 + (gamma(i)-1)/2*M(i)^2*r_st);     %
recovery temperature [R]
%                     T_aw_st = Tc_ns_st*((1 + (gamma(i)-1)/2*M(i)^2*r_st)/(1
+ (gamma(i)-1)/2*M(i)^2));  % adiabatic wall temperature [R]
%
%                     % Gas convective heat transfer coefficient
[BTU/(in^2*s*oF)]
%                     b_st = (1+(gamma_chmbr-1)/2*M(i)^2);
%                     sig_st = (((1/2)*(T_wg_st/Tc_ns_st)*b_st + 1/2)^0.68 *
b_st^0.12)^(-1);
%                     AtA = eps_tot(i)^(-1);
%                     R = thrtcrvtr;
%                     a_st = (0.026/(r_evnspc(i_th)*2)^0.2)*...
%
(mu_SI2eng(mu_chmbr)^0.2*cp_SI2eng(C_p_chmbr)/Pr_g_chmbr^0.6)...
%
*(P_c*g_eng/vel_ms2fs(cstar(i)))^0.8*(r_evnspc(i_th)*2/R)^0.1;
```

```
%                     hg_st = a_st*(AtA)^(0.9)*sig_st;
%                     hg_st = hg_st*hg_kd_Bndr*hg_kd_RDE;                  %
[Btu/(s*in^2*R)] or [Btu/(s*in^2*F)]
%                     hg_st_SI = cnvctvHtXfer_conv(hg_st,'eng2SI');        %
[W/(m^2*K)]

                    % Gas calculations
                    Tg_st = temp_conv(T_g(i),'K2R');           % gas temp [R]
                    Tc_ns_st = temp_conv(T_g_chmbr,'K2R');     % gas CC stag
temp [R]
                    r_st = Pr_g(i)^(1/3);               % recovery factor
(assume turbulent freestream boundary layer)
                    T_r_st = Tg_st*(1 + (gamma(i)-1)/2*M(i)^2*r_st);     %
recovery temperature [R]
                    T_aw_st = Tc_ns_st*((1 + (gamma(i)-1)/2*M(i)^2*r_st)/(1 +
(gamma(i)-1)/2*M(i)^2));  % adiabatic wall temperature [R]

                    % Gas convective heat transfer coefficient
[BTU/(in^2*s*oF)]
%                     b_st = (1+(gamma_chmbr-1)/2*M(i)^2);
                    b_st = (1+(gamma(i)-1)/2*M(i)^2);
                    sig_st = (((1/2)*(T_wg_st/Tc_ns_st)*b_st + 1/2)^0.68 *
b_st^0.12)^(-1);
                    AtA = eps_tot(i)^(-1);
                    R = thrtcrvtr;
                    a_st = (0.026/len_conv(Dout(1)-Din(1),'m2engin')^0.2)*...

(mu_SI2eng(mu_chmbr)^0.2*cp_SI2eng(C_p_chmbr)/Pr_g_chmbr^0.6)...

*(P_c*g_eng/vel_ms2fs(cstar(i)))^0.8*(len_conv(Dout(1)-
Din(1),'m2engin')/R)^0.1;
                    hg_st = a_st*(AtA)^(0.9)*sig_st;
%                     hg_st = hg_st*hg_kd_Bndr*hg_kd_RDE;                  %
[Btu/(s*in^2*R)] or [Btu/(s*in^2*F)]
    %                 hg_st_SI = cnvctvHtXfer_conv(hg_st,'eng2SI');        %
[W/(m^2*K)]

                    % Heat Flux [BTU/(in^2*s)]  (used [R] to calculate)
                    q_g_st = hg_st*(T_aw_st-
T_wg_st)*hg_kd_Bndr*hg_kd_RDE;   % [BTU/(in^2*s)]

                    % Coolant-side wall temp [R]
%                     T_wc_st = T_wg_st - (w_th_st/k_w_st)*q_g_st;    % [R]
                    T_wc_st = T_wg_st -
temp_conv((w_th_st/k_w_st)*q_g_st,'F2R');    % [R]

                    % Local hydrogen properties
                    % Cp_c_st [BTU/(lbm*R)] = [BTU/(lbm*F)]
                    % mu_c_st [lbm/(in*s)]
                    % k_c_st [BTU/(in*s*R)] = [BTU/(in*s*F)]
                    % Pr_c_st
                    % rho_c_st [lbm/in^3]

%                     [Cp_c_st,mu_c_st,Pr_c_st,rho_c_st] =
intrpH2lcl(T_H2,Cp_H2,mu_H2,k_H2,rho_H2,T_c_st_i_vec(i));
```

```
                    [Cp_c_st,mu_c_st,Pr_c_st,rho_c_st] =
H2props_refprop(T_H2_i_i,P_H2_i_i);
%                   [Cp_c_st,mu_c_st,Pr_c_st,rho_c_st] =
intrpH2lcl(T_H2,Cp_H2,mu_H2,k_H2,rho_H2,T_c_temp_st);
%                     D_h_c_st = 4*A_RL10_CEA(i)/Per_CEA(i); % hydraulic
diameter [in]
                    D_h_c_st = 4*A_chnl_i/Per_chnl_i;       % hydraulic
diameter [in]
                    G_st = mdot_c_i/(pi*D_h_c_st^2/4);      % coolant weight
flow rate per unit area [lb/(in^2*s)]
                    v_c_st = G_st/rho_c_st/num_chnls_i;     % [in/s]
                    Re_c_st = rho_c_st*v_c_st*D_h_c_st/mu_c_st;

                    % Coolant convective heat transfer coefficient (Sieder-
Tate) % [BTU/(in^2*s*oR)]
                    h_c_st =
hcH2HuzelHuang(Cp_c_st,mu_c_st,Pr_c_st,G_st,D_h_c_st,T_c_st_i_vec(i),T_wc_st)
;
    %                 h_c_st =
hcH2HuzelHuang(Cp_c_st,mu_c_st,Pr_c_st,G_st,D_h_c_st,T_c_temp_st,T_wc_st);

                    % Iteration calculated coolant temperature [R]
                    T_c_temp_st = T_wc_st - q_g_st/h_c_st;        % [R]
%                     T_c_temp_st = T_wc_st -
temp_F2R(q_g_st/h_c_st);        % [R]

                    % Increment guessed gas-side wall temp [R]
                    T_wg_st = T_wg_st + T_wg_inc;

                end

                % Iterate to find Darcy friction factor
                f_D = frctnDarcy(Re_c_st);

                if i ~= 1 % was ~= length(h_g_1_RL10(2,2,:,1))

                    % Calculate coolant heat absorption
                    R_cur = Din(i)/2;           % [m]
                    R_nxt = Din(i-1)/2;         % [m]
                    dh = sqrt((R_cur-R_nxt)^2 + (x(i)-x(i-1))^2);   %
incremental slant height [m]
%                     dh = sqrt((r_i_CEA(i)-r_i_CEA(i-1))^2 + (x_CEA(i)-
x_CEA(i-1))^2);
%                     R_cur = Din(i)/2;           % [m]
%                     R_nxt = Din(i-1)/2;         % [m]
    %                 R_cur = sqrt(eps_tot(i)*A_t*4/pi);
    %                 R_nxt = sqrt(eps_tot(i-1)*A_t*4/pi);
                    dA =
pi*len_conv((R_cur+R_nxt),'m2engin')*len_conv(dh,'m2engin');   % [in^2]
%                     A =
pi*sqrt(eps_tot(i)*A_t*4/pi)*len_conv(dh,'SI2engin');   % [in^2]
    %                 A = D_h_c_st*len_conv(delx,'SI2engin');   % [in^2]
                    dQ_st = dA*q_g_st;              % [Btu/s]
                    Q_st_i_vec(i-1) = Q_st_i_vec(i) + dQ_st;
```

```
                    T_c_st_i_vec(i-1) = T_c_st_i_vec(i) +
dQ_st/(mdot_c_i*Cp_c_st)/dvsr_Tc_inc_prchnl;      % [R] (was i+1 instead of
i-1)

                    % Calculate coolant pressure drop
                    dP = press_conv(...
                        fdfctr_Bndr*f_D*...

(rho_eng2SI_in3(rho_c_st)/2)*(vel_conv(v_c_st,'engin2SI')^2/len_conv(D_h_c_st
,'engin2m'))*dh,...
                        'SI2eng');
                    Pclnt_i_vec(i-1) = Pclnt_i_vec(i) - dP;
                else
                    dA = 0;
                    dQ_st = 0;

                end

                if i == length(eps_tot)
                    dA_back = 0;
                    A_i_vec(i) = dA_back;
                elseif i < length(eps_tot)
                    R_cur = Din(i)/2;           % [m]
                    R_prv = Din(i+1)/2;         % [m]
                    dh = sqrt((R_cur-R_prv)^2 + (x(i+1)-x(i))^2);    %
incremental slant height [m]
                    dA_back =
pi*len_conv((R_cur+R_prv),'m2engin')*len_conv(dh,'m2engin');    % [in^2]
                    A_i_vec(i) = A_i_vec(i+1) + dA_back;     % [in^2]
                end

                eng_per = 2*pi*R_cur;    % [m]
                chnl_per_tot = num_chnls_i*D_h_c_st;  % [in]

                asgn_vec_vals_i_RDE
            end

            % Save results
            if i == 1
                if exist(rgnrsltsflnm,'file') == 2
                    delete(rgnrsltsflnm);
                end
                save(rgnrsltsflnm,...
                    'x_RDE_plt_en_1p',...
                    ...

'T_g_o_1p_vec','T_aw_o_1p_vec','T_wg_o_1p_vec','T_wc_st_o_1p_vec','T_c_st_o_1
p_vec','q_g_st_o_1p_vec',...

'h_c_st_o_1p_vec','hg_st_o_1p_vec','Pclnt_o_1p_vec','f_D_o_1p_vec','D_h_c_st_
o_1p_vec','v_c_st_o_1p_vec',...

'egn_per_o_1p_vec','chnl_per_tot_o_1p_vec','Re_c_st_o_1p_vec','h_c_st_o_1p_ve
c',...
```

```
'rho_c_st_o_1p_vec','mu_c_st_o_1p_vec','Cp_c_st_o_1p_vec','Pr_c_st_o_1p_vec',
'dA_o_back_1p_vec',...
                    ...

'T_g_o_vec','T_aw_o_vec','T_wg_o_vec','T_wc_st_o_vec','T_c_st_o_vec','q_g_st_
o_vec',...

'h_c_st_o_vec','hg_st_o_vec','Pclnt_o_vec','f_D_o_vec','D_h_c_st_o_vec','v_c_
st_o_vec','dA_o_back_vec',...
                    ...

'T_g_i_vec','T_aw_i_vec','T_wg_i_vec','T_wc_st_i_vec','T_c_st_i_vec','q_g_st_
i_vec',...

'h_c_st_i_vec','hg_st_i_vec','Pclnt_i_vec','f_D_i_vec','D_h_c_st_i_vec','v_c_
st_i_vec','dA_i_back_vec',...
                    ...

'dA_o_1p_vec','A_o_1p_vec','A_chnl_o_1p_vec','Per_chnl_o_1p_vec',...

'dQ_st_o_1p_vec','Q_st_o_1p_vec','hg_kd_o_1p_vec','egn_per_o_vec','chnl_per_t
ot_o_vec',...
                    ...
                    'dA_o_vec','A_o_vec','A_chnl_o_vec','Per_chnl_o_vec',...

'dQ_st_o_vec','Q_st_o_vec','hg_kd_o_vec','Re_c_st_o_vec','h_c_st_o_vec',...

'rho_c_st_o_vec','mu_c_st_o_vec','Cp_c_st_o_vec','Pr_c_st_o_vec',...
                    ...
                    'dA_i_vec','A_i_vec','A_chnl_i_vec','Per_chnl_i_vec',...

'dQ_st_i_vec','Q_st_i_vec','hg_kd_i_vec','egn_per_i_vec','chnl_per_tot_i_vec'
,'Re_c_st_i_vec','h_c_st_i_vec',...

'rho_c_st_i_vec','mu_c_st_i_vec','Cp_c_st_i_vec','Pr_c_st_i_vec',...
                    ...

'P_drp_clnjckt_Bndr','T_rs_clngjckt_Bndr','i_regen_strt','i_arspk','mdot_c','
mdot_o_frctn',...
                    'fdfctr_Bndr','hg_kd_Bndr','ttl_hxfr_rt_Bndr');
            end
        end

    else
        load(rgnrsltsflnm);
        fprintf('RDE Regen Loaded:\n')
    end

    if plt_regen

        clr_out_crct = [0.9, 0.05, 0.1];
        clr_inr_crct = [0.05, 0.4, 0.2];
        clr_RL10 = [0.4940, 0.1840, 0.5560];

        f6 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
```

```
        hold on
        p1_1p =
plot(x_RDE_plt_en_1p,temp_R2F(T_g_o_1p_vec(T_g_o_1p_vec~=0)),'color',[0.6350,
0.0780, 0.1840]);
        p1 = plot(x_RDE_plt_en,temp_R2F(T_g_o_vec),'color',[0.6350, 0.0780,
0.1840]);
        p3_1p =
plot(x_RDE_plt_en_1p,temp_R2F(T_wg_o_1p_vec(T_wg_o_1p_vec~=0)),'color',[0.850
0, 0.3250, 0.0980]);
        p3 = plot(x_RDE_plt_en,temp_R2F(T_wg_o_vec),'color',[0.8500, 0.3250,
0.0980]);
        p4 = plot(x_RDE_plt_en,temp_R2F(T_wg_i_vec),'color',[0.8500, 0.3250,
0.0980],'linestyle','--');
        p5_1p =
plot(x_RDE_plt_en_1p,temp_R2F(T_wc_st_o_1p_vec(T_wc_st_o_1p_vec~=0)),'color',
[0.9290, 0.6940, 0.1250]);
        p5 = plot(x_RDE_plt_en,temp_R2F(T_wc_st_o_vec),'color',[0.9290,
0.6940, 0.1250]);
        p6 = plot(x_RDE_plt_en,temp_R2F(T_wc_st_i_vec),'color',[0.9290,
0.6940, 0.1250],'linestyle','--');
        p7_1p =
plot(x_RDE_plt_en_1p,temp_R2F(T_c_st_o_1p_vec(T_c_st_o_1p_vec~=0)),'color',[0
, 0.4470, 0.7410]);
        p7 = plot(x_RDE_plt_en,temp_R2F(T_c_st_o_vec),'color',[0, 0.4470,
0.7410]);
        p8 = plot(x_RDE_plt_en,temp_R2F(T_c_st_i_vec),'color',[0, 0.4470,
0.7410],'linestyle','--');
        p9 = plot([-20,50],temp_R2F([75 75]),'m--');
        p10 = plot([-20,50],temp_R2F([50 50]),'k--');
        p11 = plot([-20,50],temp_R2F([500 500]),'k--');
        xlabel('Distance from Throat [in]')
        ylabel('T [^oF]')
        title('Temperature Profiles')
        legend([p1 p3 p4 p5 p6 p7 p8 p9 p10],'Gas Temperature','Gas-Side Wall
Temperature, Outer','Gas-Side Wall Temperature, Inner','Coolant-Side Wall
Temperature, Outer','Coolant-Side Wall Temperature, Inner','Hydrogen Fuel
Coolant Temperature, Outer','Hydrogen Fuel Coolant Temperature, Inner','H_2
Critical Temperature','H_2 Property Temperature
Bounds','location','northeast')

        f7 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        subplot(1,2,1)
        hold on
        p3_1p =
plot(x_RDE_plt_en_1p,temp_R2F(T_wg_o_1p_vec(T_wg_o_1p_vec~=0)),'color',[0.850
0, 0.3250, 0.0980]);
        p3 = plot(x_RDE_plt_en,temp_R2F(T_wg_o_vec),'color',[0.8500, 0.3250,
0.0980]);
        p5_1p =
plot(x_RDE_plt_en_1p,temp_R2F(T_wc_st_o_1p_vec(T_wc_st_o_1p_vec~=0)),'color',
[0.9290, 0.6940, 0.1250]);
        p5 = plot(x_RDE_plt_en,temp_R2F(T_wc_st_o_vec),'color',[0.9290,
0.6940, 0.1250]);
        p7_1p =
plot(x_RDE_plt_en_1p,temp_R2F(T_c_st_o_1p_vec(T_c_st_o_1p_vec~=0)),'color',[0
, 0.4470, 0.7410]);
```

```
        p7 = plot(x_RDE_plt_en,temp_R2F(T_c_st_o_vec),'color',[0, 0.4470,
0.7410]);
        p9 = plot([-20,50],temp_R2F([75 75]),'m--');
        p10 = plot([-20,50],temp_R2F([50 50]),'k--');
        p11 = plot([-20,50],temp_R2F([500 500]),'k--');
        xlabel('Distance from Throat [in]')
        ylabel('T [^oF]')
        title('Temperature Profiles, Outer')
        legend([p3 p5 p7 p9 p10],'Gas-Side Wall Temperature, Outer','Coolant-
Side Wall Temperature, Outer','Hydrogen Fuel Coolant Temperature, Outer','H_2
Critical Temperature','H_2 Property Temperature
Bounds','location','northeast')

        subplot(1,2,2)
        hold on
        p4 = plot(x_RDE_plt_en,temp_R2F(T_wg_i_vec),'color',[0.8500, 0.3250,
0.0980]);
        p6 = plot(x_RDE_plt_en,temp_R2F(T_wc_st_i_vec),'color',[0.9290,
0.6940, 0.1250]);
        p8 = plot(x_RDE_plt_en,temp_R2F(T_c_st_i_vec),'color',[0, 0.4470,
0.7410]);
        p9 = plot([-20,50],temp_R2F([75 75]),'m--');
        p10 = plot([-20,50],temp_R2F([50 50]),'k--');
        p11 = plot([-20,50],temp_R2F([500 500]),'k--');
        xlabel('Distance from Throat [in]')
        ylabel('T [^oF]')
        title('Temperature Profiles, Inner')
        legend([p4 p6 p8 p9 p10],'Gas-Side Wall Temperature, Inner','Coolant-
Side Wall Temperature, Inner','Hydrogen Fuel Coolant Temperature, Inner','H_2
Critical Temperature','H_2 Property Temperature
Bounds','location','northeast')

        f8 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        subplot(1,2,1)
        hold on
        p1_1p =
plot(x_RDE_plt_en_1p,[dA_o_1p_vec(dA_o_1p_vec~=0);0],'color',[0.6350, 0.0780,
0.1840]);
        p1 = plot(x_RDE_plt_en,dA_o_vec,'color',[0.6350, 0.0780, 0.1840]);
        p2 = plot(x_RDE_plt_en,dA_i_vec,'color',[0.8500, 0.3250, 0.0980]);
        p3_1p =
plot(x_RDE_plt_en_1p,[dA_o_back_1p_vec(dA_o_1p_vec~=0);0],'color',[0.6350,
0.0780, 0.1840],'linestyle','--');
        p3 = plot(x_RDE_plt_en,dA_o_back_vec,'color',[0.6350, 0.0780,
0.1840],'linestyle','--');
        p4 = plot(x_RDE_plt_en,dA_i_back_vec,'color',[0.8500, 0.3250,
0.0980],'linestyle','--');
        xlabel('Distance from Throat [in]')
        ylabel('dA [in^2]')
        title('Incremental Area')
        legend([p1 p2 p3 p4],'Outer, Foward','Inner, Foward','Outer,
Backward','Inner, Backward')

        subplot(1,2,2)
        hold on
```

```
        p1 = plot(x_RDE_plt_en,A_o_vec-
A_o_1p_vec(end)+A_i_vec,'color',[0.6350, 0.0780, 0.1840]);
        p2 = plot(x_RDE_plt_en,A_o_vec-A_o_1p_vec(end),'color',[0.8500,
0.3250, 0.0980]);
        p3 = plot(x_RDE_plt_en,A_i_vec,'color',[0.9290, 0.6940, 0.1250]);
        xlabel('Distance from Throat [in]')
        ylabel('A [in^2]')
        title('Cumulative Area')
        legend([p1 p2 p3],'Total','Outer','Inner')

        f9 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1_1p =
plot(x_RDE_plt_en_1p,A_chnl_o_1p_vec(A_chnl_o_1p_vec~=0),'color',[0.6350,
0.0780, 0.1840]);
        p1 = plot(x_RDE_plt_en,A_chnl_o_vec,'color',[0.6350, 0.0780,
0.1840]);
        p2 = plot(x_RDE_plt_en,A_chnl_i_vec,'color',[0.8500, 0.3250,
0.0980]);
        xlabel('Distance from Throat [in]')
        ylabel('A [in^2]')
        title('Cooling Channel Cross Sectional Area')
        legend([p1 p2],'Outer','Inner')

        f10 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        plot(x_RDE_plt_en,Per_chnl_o_vec,'-',x_RDE_plt_en,Per_chnl_i_vec,'--
')
        p1_1p =
plot(x_RDE_plt_en_1p,Per_chnl_o_1p_vec(Per_chnl_o_1p_vec~=0),'color',[0.6350,
0.0780, 0.1840]);
        p1 = plot(x_RDE_plt_en,Per_chnl_o_vec,'color',[0.6350, 0.0780,
0.1840]);
        p2 = plot(x_RDE_plt_en,Per_chnl_i_vec,'color',[0.8500, 0.3250,
0.0980]);
        xlabel('Distance from Throat [in]')
        ylabel('Per [in]')
        title('Cooling Channel Perimeter')
        legend([p1 p2],'Outer','Inner')

        f11 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1 = plot(x_RDE_plt_en,D_h_c_st_o_vec,'color',[0.6350, 0.0780,
0.1840]);
        p1_1p =
plot(x_RDE_plt_en_1p,D_h_c_st_o_1p_vec(D_h_c_st_o_1p_vec~=0),'color',[0, 0,
0],'linestyle','--');
        p2 =
plot(x_RDE_plt_en(D_h_c_st_i_vec~=0),D_h_c_st_i_vec(D_h_c_st_i_vec~=0),'color
',[0.8500, 0.3250, 0.0980]);
        xlabel('Distance from Throat [in]')
        ylabel('D_h [in]')
        title('Cooling Channel Hydraulic Diameter')
        legend([p1_1p p1 p2],'Outer, Short','Outer, Long','Inner (Long
Only)')
```

```
        f12 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        subplot(1,2,1)
        hold on
        p1_1p =
plot(x_RDE_plt_en_1p,[dQ_st_o_1p_vec(dQ_st_o_1p_vec~=0);0],'color',[0.6350,
0.0780, 0.1840]);
        p1 = plot(x_RDE_plt_en,dQ_st_o_vec,'color',[0.6350, 0.0780, 0.1840]);
        p2 = plot(x_RDE_plt_en,dQ_st_i_vec,'color',[0.8500, 0.3250, 0.0980]);
        xlabel('Distance from Throat [in]')
        ylabel('dQ [Btu/s]')
        title('Incremental Heat Rate')
        legend([p1 p2],'Outer','Inner')

        subplot(1,2,2)
        hold on
        p1 = plot(x_RDE_plt_en,Q_st_o_vec+Q_st_i_vec,'color',[0.6350, 0.0780,
0.1840]);
        p2 = plot(x_RDE_plt_en,Q_st_o_vec,'color',[0.8500, 0.3250, 0.0980]);
        p3 = plot(x_RDE_plt_en,Q_st_i_vec,'color',[0.9290, 0.6940, 0.1250]);
        xlabel('Distance from Throat [in]')
        ylabel('Q [Btu/s]')
        title('Cumulative Heat Rate')
        legend('Total','Outer','Inner')

        f13 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1_1p =
plot(x_RDE_plt_en_1p,hg_kd_o_1p_vec(hg_kd_o_1p_vec~=0),'color',[0.6350,
0.0780, 0.1840]);
        p1 = plot(x_RDE_plt_en,hg_kd_o_vec,'color',[0.6350, 0.0780, 0.1840]);
        p2 = plot(x_RDE_plt_en,hg_kd_i_vec,'color',[0.8500, 0.3250, 0.0980]);
        xlabel('Distance from Throat [in]')
        ylabel('Knockdown')
        title('Total h_g Knockdown')
        legend([p1 p2],'Outer','Inner')

        f14 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1_1p =
plot(x_RDE_plt_en_1p,hg_st_o_1p_vec(hg_st_o_1p_vec~=0),'color',[0.6350,
0.0780, 0.1840]);
        p1 = plot(x_RDE_plt_en,hg_st_o_vec,'color',[0.6350, 0.0780, 0.1840]);
        p2 = plot(x_RDE_plt_en,hg_st_i_vec,'color',[0.8500, 0.3250, 0.0980]);
        xlabel('Distance from Throat [in]')
        ylabel('h_g')
        title('Convective Heat Transfer Coefficient')
        legend([p1 p2],'Outer','Inner')

        f15 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1_1p =
plot(x_RDE_plt_en_1p,[0;Q_st_o_1p_vec(Q_st_o_1p_vec~=0)],'linestyle','-
','color',[0, 0.4470, 0.7410]);
        p1 = plot(x_RDE_plt_en,Q_st_o_vec+Q_st_i_vec,'linestyle','-
','color',[0, 0.4470, 0.7410]);
```

```
        p2 =
plot(RL10rgn.x_plt_1p,[0;RL10rgn.Q_1p(RL10rgn.Q_1p~=0)],'color',[0.6350,
0.0780, 0.1840]);
        p3 = plot(RL10rgn.x_plt_2p,RL10rgn.Q_2p,'color',[0.6350, 0.0780,
0.1840]);
        p4 = plot([min(RL10rgn.x_plt_2p)
max(RL10rgn.x_plt_2p)],[RL10rgn.Binder.Q RL10rgn.Binder.Q],'k--');
        xlim([-13 47.54])
        xlabel('Distance from Throat [in]')
        ylabel('Q [Btu/s]')
%         title('Cumulative Heat Rate')
        legend([p1 p2 p4],'RDE','RL10',sprintf('Binder Total Heat Transfer
Rate = %i [Btu/s]',RL10rgn.Binder.Q),'location','northeast')

        f16 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1_1p =
plot(x_RDE_plt_en_1p,cnvctvHtXfer_conv(hg_st_o_1p_vec(hg_st_o_1p_vec~=0),'eng
2SI'),'linestyle','-','color',[0, 0.4470, 0.7410]);
        p1 =
plot(x_RDE_plt_en,cnvctvHtXfer_conv(hg_st_o_vec,'eng2SI'),'linestyle','-
','color',[0, 0.4470, 0.7410]);
        p2 =
plot(x_RDE_plt_en(hg_st_i_vec~=0),cnvctvHtXfer_conv(hg_st_i_vec(hg_st_i_vec~=
0),'eng2SI'),'linestyle','-','color',[0, 0, 0],'linestyle','--');
        p3 =
plot(RL10rgn.x_plt_1p,cnvctvHtXfer_conv(RL10rgn.hg_1p(RL10rgn.hg_1p~=0),'eng2
SI'),'color',[0.6350, 0.0780, 0.1840]);
        p4 =
plot(RL10rgn.x_plt_2p,cnvctvHtXfer_conv(RL10rgn.hg_2p,'eng2SI'),'color',[0.63
50, 0.0780, 0.1840]);
        xlabel('Distance from Throat [in]')
        ylabel('h_g [W/(m^2*K)]')
        ylim([0 50000])
%         title('Hot Gas Convective Heat Transfer Coefficient')
        legend([p1 p2 p3],'RDE, Outer','RDE,
Inner','RL10','location','northeast')

        f17 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1_1p =
plot(x_RDE_plt_en_1p,htFlx_conv(q_g_st_o_1p_vec(q_g_st_o_1p_vec~=0),'eng2SI')
,'linestyle','-','color',[0, 0.4470, 0.7410]);
        p1 =
plot(x_RDE_plt_en,htFlx_conv(q_g_st_o_vec,'eng2SI'),'linestyle','-
','color',[0, 0.4470, 0.7410]);
        p2 =
plot(x_RDE_plt_en(q_g_st_i_vec~=0),htFlx_conv(q_g_st_i_vec(q_g_st_i_vec~=0),'
eng2SI'),'linestyle','-','color',[0, 0, 0],'linestyle','--');
        p3 =
plot(RL10rgn.x_plt_1p,htFlx_conv(RL10rgn.q_1p(RL10rgn.q_1p~=0),'eng2SI'),'col
or',[0.6350, 0.0780, 0.1840]);
        p4 =
plot(RL10rgn.x_plt_2p,htFlx_conv(RL10rgn.q_2p,'eng2SI'),'color',[0.6350,
0.0780, 0.1840]);
        xlabel('Distance from Throat [in]')
```

```
        ylabel('q [W/m^2]')
%         title('Heat Flux')
        legend([p1 p2 p3],'RDE, Outer','RDE,
Inner','RL10','location','northeast')

        f18 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1_1p =
plot(x_RDE_plt_en_1p,[0;ar_conv(A_o_1p_vec(A_o_1p_vec~=0),'in22m2')],'linesty
le','-','color',[0, 0.4470, 0.7410]);
        p1 =
plot(x_RDE_plt_en,ar_conv(A_o_vec+A_i_vec,'in22m2'),'linestyle','-
','color',[0, 0.4470, 0.7410]);
        p2 =
plot(RL10rgn.x_plt_1p,[0;ar_conv(RL10rgn.SA_1p(RL10rgn.SA_1p~=0),'in22m2')],'
color',[0.6350, 0.0780, 0.1840]);
        p3 =
plot(RL10rgn.x_plt_2p,ar_conv(RL10rgn.SA_2p,'in22m2'),'color',[0.6350,
0.0780, 0.1840]);
        xlabel('Distance from Throat [in]')
        ylabel('A [m^2)]')
        title('Cumulative Area (Double Pass)')
        legend([p1 p2],'RDE','RL10','location','northeast')

        f19 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1 = plot(x_RDE_plt_en,ar_conv(A_o_vec-
A_o_1p_vec(end)+A_i_vec,'in22m2'),'linestyle','-','color',[0, 0.4470,
0.7410]);
        p2 = plot(RL10rgn.x_plt_2p,ar_conv(RL10rgn.SA_2p-
RL10rgn.SA_1p(end),'in22m2'),'color',[0.6350, 0.0780, 0.1840]);
        xlabel('Distance from Throat [in]')
        ylabel('A [m^2)]')
        title('Cumulative Area (Single Pass)')
        legend([p1 p2],'RDE','RL10','location','northeast')

        f20 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
%         p3_1p =
plot(x_RDE_plt_en_1p,Pclnt_o_1p_vec(Pclnt_o_1p_vec~=0),'color',[0.8500,
0.3250, 0.0980]);
%         p3 = plot(x_RDE_plt_en,Pclnt_o_vec,'color',[0.8500, 0.3250,
0.0980]);
%         p5 =
plot(x_RDE_plt_en(Pclnt_i_vec~=0),Pclnt_i_vec(Pclnt_i_vec~=0),'color',[0.9290
, 0.6940, 0.1250]);
%         p7 =
plot(RL10rgn.x_plt_1p,RL10rgn.P_c_1p(RL10rgn.P_c_1p~=0),'color',[0, 0.4470,
0.7410]);
%         p8 = plot(RL10rgn.x_plt_2p,RL10rgn.P_c_2p,'color',[0, 0.4470,
0.7410]);
        p3_1p =
plot(x_RDE_plt_en_1p,Pclnt_o_1p_vec(Pclnt_o_1p_vec~=0),'color',clr_out_crct);
        p3 = plot(x_RDE_plt_en,Pclnt_o_vec,'color',clr_out_crct);
```

```
        p5 =
plot(x_RDE_plt_en(Pclnt_i_vec~=0),Pclnt_i_vec(Pclnt_i_vec~=0),'color',clr_inr
_crct);
        p7 =
plot(RL10rgn.x_plt_1p,RL10rgn.P_c_1p(RL10rgn.P_c_1p~=0),'color',clr_RL10);
        p8 = plot(RL10rgn.x_plt_2p,RL10rgn.P_c_2p,'color',clr_RL10);
        ylim([725 1025])
        xlabel('Distance from Throat [in]')
        ylabel('P [psia]')
%         title('Coolant Pressure Profiles')
        legend([p3 p5 p7],'RDE, Outer','RDE,
Inner','RL10','location','northeast')

        f21 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        subplot(1,2,1)
        hold on
        p3_1p =
plot(x_RDE_plt_en_1p,temp_R2F(T_wg_o_1p_vec(T_wg_o_1p_vec~=0)),'color',[0.850
0, 0.3250, 0.0980]);
        p3 = plot(x_RDE_plt_en,temp_R2F(T_wg_o_vec),'color',[0.8500, 0.3250,
0.0980]);
        p5_1p =
plot(x_RDE_plt_en_1p,temp_R2F(T_wc_st_o_1p_vec(T_wc_st_o_1p_vec~=0)),'color',
[0.9290, 0.6940, 0.1250]);
        p5 = plot(x_RDE_plt_en,temp_R2F(T_wc_st_o_vec),'color',[0.9290,
0.6940, 0.1250]);
        p7_1p =
plot(x_RDE_plt_en_1p,temp_R2F(T_c_st_o_1p_vec(T_c_st_o_1p_vec~=0)),'color',[0
, 0.4470, 0.7410]);
        p7 = plot(x_RDE_plt_en,temp_R2F(T_c_st_o_vec),'color',[0, 0.4470,
0.7410]);
        p8 =
plot(RL10rgn.x_plt_1p,temp_R2F(RL10rgn.T_wg_1p(RL10rgn.T_wg_1p~=0)),'color',[
0.4940, 0.1840, 0.5560]);
        p9 = plot(RL10rgn.x_plt_2p,temp_R2F(RL10rgn.T_wg_2p),'color',[0.4940,
0.1840, 0.5560]);
        xlabel('Distance from Throat [in]')
        ylabel('T [^oF]')
        title('Temperature Profiles')
        legend([p3 p5 p7 p8],'RDE Gas-Side Wall Temperature, Outer','RDE
Coolant-Side Wall Temperature, Outer','RDE Hydrogen Fuel Coolant Temperature,
Outer','RL10 Gas-Side Wall Temperature','location','northeast')

        subplot(1,2,2)
        hold on
        p4 =
plot(x_RDE_plt_en(T_wg_i_vec~=0),temp_R2F(T_wg_i_vec(T_wg_i_vec~=0)),'color',
[0.8500, 0.3250, 0.0980]);
        p6 =
plot(x_RDE_plt_en(T_wc_st_i_vec~=0),temp_R2F(T_wc_st_i_vec(T_wc_st_i_vec~=0))
,'color',[0.9290, 0.6940, 0.1250]);
        p8 =
plot(x_RDE_plt_en(T_c_st_i_vec~=0),temp_R2F(T_c_st_i_vec(T_c_st_i_vec~=0)),'c
olor',[0, 0.4470, 0.7410]);
```

```
        p9 =
plot(RL10rgn.x_plt_1p,temp_R2F(RL10rgn.T_wg_1p(RL10rgn.T_wg_1p~=0)),'color',[
0.4940, 0.1840, 0.5560]);
        p10 =
plot(RL10rgn.x_plt_2p,temp_R2F(RL10rgn.T_wg_2p),'color',[0.4940, 0.1840,
0.5560]);
        xlabel('Distance from Throat [in]')
        ylabel('T [^oF]')
        title('Temperature Profiles')
        legend([p4 p6 p8 p9],'Gas-Side Wall Temperature, Inner','Coolant-Side
Wall Temperature, Inner','Hydrogen Fuel Coolant Temperature, Inner','RL10
Gas-Side Wall Temperature','location','northeast')

        f22 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p3_1p =
plot(x_RDE_plt_en_1p,chnl_per_tot_o_1p_vec(chnl_per_tot_o_1p_vec~=0)./len_con
v(egn_per_o_1p_vec(egn_per_o_1p_vec~=0),'m2engin'),'color',[0.8500, 0.3250,
0.0980]);
        p3 =
plot(x_RDE_plt_en,chnl_per_tot_o_vec./len_conv(egn_per_o_vec,'m2engin'),'colo
r',[0.8500, 0.3250, 0.0980]);
        p5 =
plot(x_RDE_plt_en,chnl_per_tot_i_vec./len_conv(egn_per_i_vec,'m2engin'),'colo
r',[0, 0.4470, 0.7410]);
        xlabel('Distance from Throat [in]')
        ylabel('Ratio')
%         title('Ratio of Sum of Long Tube Coolant Channel Hydraulic
Diameters to Engine Perimeter')
        legend([p3 p5],'RDE, Outer','RDE, Inner','location','northeast')

        f23 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1_1p =
plot(x_RDE_plt_en_1p,rho_c_st_o_1p_vec(rho_c_st_o_1p_vec~=0),'color',clr_out_
crct);
        p1 = plot(x_RDE_plt_en,rho_c_st_o_vec,'color',clr_out_crct);
        p2 =
plot(x_RDE_plt_en(rho_c_st_i_vec~=0),rho_c_st_i_vec(rho_c_st_i_vec~=0),'color
',clr_inr_crct);
        p3_1p =
plot(RL10rgn.x_plt_1p,RL10rgn.rho_c_1p(RL10rgn.rho_c_1p~=0),'color',clr_RL10)
;
        p3 = plot(RL10rgn.x_plt_2p,RL10rgn.rho_c_2p,'color',clr_RL10);
        xlabel('Distance from Throat [in]')
        ylabel('\rho [lbm/in^3]')
%         title('Coolant Density')
        legend([p1 p2 p3],'RDE, Outer','RDE,
Inner','RL10','location','northeast')

        f24 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1_1p =
plot(x_RDE_plt_en_1p,vel_conv(v_c_st_o_1p_vec(v_c_st_o_1p_vec~=0),'engin2SI')
,'color',clr_out_crct);
```

```
        p1 =
plot(x_RDE_plt_en,vel_conv(v_c_st_o_vec,'engin2SI'),'color',clr_out_crct);
        p2 =
plot(x_RDE_plt_en(v_c_st_i_vec~=0),vel_conv(v_c_st_i_vec(v_c_st_i_vec~=0),'en
gin2SI'),'color',clr_inr_crct);
        p3_1p =
plot(RL10rgn.x_plt_1p,vel_conv(RL10rgn.v_c_1p(RL10rgn.v_c_1p~=0),'engin2SI'),
'color',[0.4940, 0.1840, 0.5560]);
        p3 =
plot(RL10rgn.x_plt_2p,vel_conv(RL10rgn.v_c_2p,'engin2SI'),'color',clr_RL10);
        xlabel('Distance from Throat [in]')
%         ylabel('v [in/s]')
        ylabel('v [m/s]')
%         title('Coolant Velocity')
        legend([p1 p2 p3],'RDE, Outer','RDE,
Inner','RL10','location','northeast')

        f25 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1_1p =
plot(x_RDE_plt_en_1p,mu_c_st_o_1p_vec(mu_c_st_o_1p_vec~=0),'color',clr_out_cr
ct);
        p1 = plot(x_RDE_plt_en,mu_c_st_o_vec,'color',clr_out_crct);
        p2 =
plot(x_RDE_plt_en(mu_c_st_i_vec~=0),mu_c_st_i_vec(mu_c_st_i_vec~=0),'color',c
lr_inr_crct);
        p3_1p =
plot(RL10rgn.x_plt_1p,RL10rgn.mu_c_1p(RL10rgn.mu_c_1p~=0),'color',clr_RL10);
        p3 = plot(RL10rgn.x_plt_2p,RL10rgn.mu_c_2p,'color',clr_RL10);
        xlabel('Distance from Throat [in]')
        ylabel('\mu [lbm/(in*s)]')
%         title('Coolant Dynamic Viscosity')
        legend([p1 p2 p3],'RDE, Outer','RDE,
Inner','RL10','location','northeast')

        f26 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1_1p =
plot(x_RDE_plt_en_1p,Re_c_st_o_1p_vec(Re_c_st_o_1p_vec~=0),'color',clr_out_cr
ct);
        p1 = plot(x_RDE_plt_en,Re_c_st_o_vec,'color',clr_out_crct);
        p2 =
plot(x_RDE_plt_en(Re_c_st_i_vec~=0),Re_c_st_i_vec(Re_c_st_i_vec~=0),'color',c
lr_inr_crct);
        p3_1p =
plot(RL10rgn.x_plt_1p,RL10rgn.Re_c_1p(RL10rgn.Re_c_1p~=0),'color',clr_RL10);
        p3 = plot(RL10rgn.x_plt_2p,RL10rgn.Re_c_2p,'color',clr_RL10);
        xlabel('Distance from Throat [in]')
        ylabel('Re')
%         title('Coolant Reynolds Number')
        legend([p1 p2 p3],'RDE, Outer','RDE,
Inner','RL10','location','northeast')

        f27 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
```

```
        p1_1p =
plot(x_RDE_plt_en_1p,h_c_st_o_1p_vec(h_c_st_o_1p_vec~=0),'color',clr_out_crct
);
        p1 = plot(x_RDE_plt_en,h_c_st_o_vec,'color',clr_out_crct);
        p2 =
plot(x_RDE_plt_en(h_c_st_i_vec~=0),h_c_st_i_vec(h_c_st_i_vec~=0),'color',clr_
inr_crct);
        p3_1p =
plot(RL10rgn.x_plt_1p,RL10rgn.hc_1p(RL10rgn.hc_1p~=0),'color',clr_RL10);
        p3 = plot(RL10rgn.x_plt_2p,RL10rgn.hc_2p,'color',clr_RL10);
        xlabel('Distance from Throat [in]')
        ylabel('h_c [Btu/(in^2*s*^oR)]')
%         title('Coolant Convective Heat Transfer Coefficient (Huzel/Huang)')
        legend([p1 p2 p3],'RDE, Outer','RDE,
Inner','RL10','location','northeast')

        f28 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1_1p =
plot(x_RDE_plt_en_1p,Cp_c_st_o_1p_vec(Cp_c_st_o_1p_vec~=0),'color',clr_out_cr
ct);
        p1 = plot(x_RDE_plt_en,Cp_c_st_o_vec,'color',clr_out_crct);
        p2 =
plot(x_RDE_plt_en(Cp_c_st_i_vec~=0),Cp_c_st_i_vec(Cp_c_st_i_vec~=0),'color',c
lr_inr_crct);
        p3_1p =
plot(RL10rgn.x_plt_1p,RL10rgn.Cp_c_1p(RL10rgn.Cp_c_1p~=0),'color',clr_RL10);
        p3 = plot(RL10rgn.x_plt_2p,RL10rgn.Cp_c_2p,'color',clr_RL10);
        xlabel('Distance from Throat [in]')
        ylabel('C_p [Btu/(lbm*^oR)]')
%         title('Coolant Isobaric Specfic Heat Constant')
        legend([p1 p2 p3],'RDE, Outer','RDE,
Inner','RL10','location','northeast')

        f29 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1_1p =
plot(x_RDE_plt_en_1p,Pr_c_st_o_1p_vec(Pr_c_st_o_1p_vec~=0),'color',clr_out_cr
ct);
        p1 = plot(x_RDE_plt_en,Pr_c_st_o_vec,'color',clr_out_crct);
        p2 =
plot(x_RDE_plt_en(Pr_c_st_i_vec~=0),Pr_c_st_i_vec(Pr_c_st_i_vec~=0),'color',c
lr_inr_crct);
        p3_1p =
plot(RL10rgn.x_plt_1p,RL10rgn.Pr_c_1p(RL10rgn.Pr_c_1p~=0),'color',clr_RL10);
        p3 = plot(RL10rgn.x_plt_2p,RL10rgn.Pr_c_2p,'color',clr_RL10);
        xlabel('Distance from Throat [in]')
        ylabel('Pr')
%         title('Coolant Prandtl Number')
        legend([p1 p2 p3],'RDE, Outer','RDE,
Inner','RL10','location','northeast')

        f30 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
```

```
%         p1_1p =
plot(x_RDE_plt_en_1p,temp_R2F(T_wg_o_1p_vec(T_wg_o_1p_vec~=0)),'color',clr_ou
t_crct);
%         p1 = plot(x_RDE_plt_en,temp_R2F(T_wg_o_vec),'color',clr_out_crct);
%         p2 =
plot(x_RDE_plt_en(T_wg_i_vec~=0),temp_R2F(T_wg_i_vec(T_wg_i_vec~=0)),'color',
clr_inr_crct);
%         p3_1p =
plot(RL10rgn.x_plt_1p,temp_R2F(RL10rgn.T_wg_1p(RL10rgn.T_wg_1p~=0)),'color',c
lr_RL10);
%         p3 =
plot(RL10rgn.x_plt_2p,temp_R2F(RL10rgn.T_wg_2p),'color',clr_RL10);
        p1_1p =
plot(x_RDE_plt_en_1p,temp_conv(T_wg_o_1p_vec(T_wg_o_1p_vec~=0),'R2K'),'color'
,clr_out_crct);
        p1 =
plot(x_RDE_plt_en,temp_conv(T_wg_o_vec,'R2K'),'color',clr_out_crct);
        p2 =
plot(x_RDE_plt_en(T_wg_i_vec~=0),temp_conv(T_wg_i_vec(T_wg_i_vec~=0),'R2K'),'
color',clr_inr_crct);
        p3_1p =
plot(RL10rgn.x_plt_1p,temp_conv(RL10rgn.T_wg_1p(RL10rgn.T_wg_1p~=0),'R2K'),'c
olor',clr_RL10);
        p3 =
plot(RL10rgn.x_plt_2p,temp_conv(RL10rgn.T_wg_2p,'R2K'),'color',clr_RL10);
        xlabel('Distance from Throat [in]')
%         ylabel('T [^oF]')
        ylabel('T_{wg} [K]')
%         title('Gas-Side Wall Temperature')
        legend([p1 p2 p3],'RDE, Outer','RDE,
Inner','RL10','location','northeast')

        f31 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
%         p1_1p =
plot(x_RDE_plt_en_1p,temp_R2F(T_wc_st_o_1p_vec(T_wc_st_o_1p_vec~=0)),'color',
clr_out_crct);
%         p1 =
plot(x_RDE_plt_en,temp_R2F(T_wc_st_o_vec),'color',clr_out_crct);
%         p2 =
plot(x_RDE_plt_en(T_wc_st_i_vec~=0),temp_R2F(T_wc_st_i_vec(T_wc_st_i_vec~=0))
,'color',clr_inr_crct);
%         p3_1p =
plot(RL10rgn.x_plt_1p,temp_R2F(RL10rgn.T_wc_1p(RL10rgn.T_wc_1p~=0)),'color',c
lr_RL10);
%         p3 =
plot(RL10rgn.x_plt_2p,temp_R2F(RL10rgn.T_wc_2p),'color',clr_RL10);
        p1_1p =
plot(x_RDE_plt_en_1p,temp_conv(T_wc_st_o_1p_vec(T_wc_st_o_1p_vec~=0),'R2K'),'
color',clr_out_crct);
        p1 =
plot(x_RDE_plt_en,temp_conv(T_wc_st_o_vec,'R2K'),'color',clr_out_crct);
        p2 =
plot(x_RDE_plt_en(T_wc_st_i_vec~=0),temp_conv(T_wc_st_i_vec(T_wc_st_i_vec~=0)
,'R2K'),'color',clr_inr_crct);
```

```
        p3_1p =
plot(RL10rgn.x_plt_1p,temp_conv(RL10rgn.T_wc_1p(RL10rgn.T_wc_1p~=0),'R2K'),'c
olor',clr_RL10);
        p3 =
plot(RL10rgn.x_plt_2p,temp_conv(RL10rgn.T_wc_2p,'R2K'),'color',clr_RL10);
        xlabel('Distance from Throat [in]')
%         ylabel('T_{wc} [^oF]')
        ylabel('T_{wc} [K]')
%         title('Coolant-Side Wall Temperature')
        legend([p1 p2 p3],'RDE, Outer','RDE,
Inner','RL10','location','northeast')

        f32 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
%         p1_1p =
plot(x_RDE_plt_en_1p,temp_R2F(T_c_st_o_1p_vec(T_c_st_o_1p_vec~=0)),'color',cl
r_out_crct);
%         p1 =
plot(x_RDE_plt_en,temp_R2F(T_c_st_o_vec),'color',clr_out_crct);
%         p2 =
plot(x_RDE_plt_en(T_c_st_i_vec~=0),temp_R2F(T_c_st_i_vec(T_c_st_i_vec~=0)),'c
olor',clr_inr_crct);
%         p3_1p =
plot(RL10rgn.x_plt_1p,temp_R2F(RL10rgn.T_c_1p(RL10rgn.T_c_1p~=0)),'color',clr
_RL10);
%         p3 =
plot(RL10rgn.x_plt_2p,temp_R2F(RL10rgn.T_c_2p),'color',clr_RL10);
        p1_1p =
plot(x_RDE_plt_en_1p,temp_conv(T_c_st_o_1p_vec(T_c_st_o_1p_vec~=0),'R2K'),'co
lor',clr_out_crct);
        p1 =
plot(x_RDE_plt_en,temp_conv(T_c_st_o_vec,'R2K'),'color',clr_out_crct);
        p2 =
plot(x_RDE_plt_en(T_c_st_i_vec~=0),temp_conv(T_c_st_i_vec(T_c_st_i_vec~=0),'R
2K'),'color',clr_inr_crct);
        p3_1p =
plot(RL10rgn.x_plt_1p,temp_conv(RL10rgn.T_c_1p(RL10rgn.T_c_1p~=0),'R2K'),'col
or',clr_RL10);
        p3 =
plot(RL10rgn.x_plt_2p,temp_conv(RL10rgn.T_c_2p,'R2K'),'color',clr_RL10);
        xlabel('Distance from Throat [in]')
%         ylabel('T_c [^oF]')
        ylabel('T_c [K]')
%         title('Coolant Temperature')
        legend([p1 p2 p3],'RDE, Outer','RDE,
Inner','RL10','location','northeast')

        f33 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
%         p1_1p =
plot(x_RDE_plt_en_1p,temp_R2F(T_g_o_1p_vec(T_g_o_1p_vec~=0)),'color',[0,
0.4470, 0.7410]);
%         p1 = plot(x_RDE_plt_en,temp_R2F(T_g_o_vec),'color',[0, 0.4470,
0.7410]);
```

```
%         p2 =
plot(x_RDE_plt_en(T_g_i_vec~=0),temp_R2F(T_g_i_vec(T_g_i_vec~=0)),'color',[0
0 0],'linestyle','--');
%         p3_1p =
plot(RL10rgn.x_plt_1p,temp_R2F(RL10rgn.T_g_1p(RL10rgn.T_g_1p~=0)),'color',[0.
6350, 0.0780, 0.1840]);
%         p3 =
plot(RL10rgn.x_plt_2p,temp_R2F(RL10rgn.T_g_2p),'color',[0.6350, 0.0780,
0.1840]);
        p1_1p =
plot(x_RDE_plt_en_1p,temp_conv(T_g_o_1p_vec(T_g_o_1p_vec~=0),'R2K'),'color',[
0, 0.4470, 0.7410]);
        p1 = plot(x_RDE_plt_en,temp_conv(T_g_o_vec,'R2K'),'color',[0, 0.4470,
0.7410]);
        p2 =
plot(x_RDE_plt_en(T_g_i_vec~=0),temp_conv(T_g_i_vec(T_g_i_vec~=0),'R2K'),'col
or',[0 0 0],'linestyle','--');
        p3_1p =
plot(RL10rgn.x_plt_1p,temp_conv(RL10rgn.T_g_1p(RL10rgn.T_g_1p~=0),'R2K'),'col
or',[0.6350, 0.0780, 0.1840]);
        p3 =
plot(RL10rgn.x_plt_2p,temp_conv(RL10rgn.T_g_2p,'R2K'),'color',[0.6350,
0.0780, 0.1840]);
        xlabel('Distance from Throat [in]')
%         ylabel('T_g [^oF]')
        ylabel('T_g [K]')
%         title('Gas Temperature')
        legend([p1 p2 p3],'RDE, Outer','RDE,
Inner','RL10','location','northeast')

        f34 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
%         p1_1p =
plot(x_RDE_plt_en_1p,temp_R2F(T_aw_o_1p_vec(T_aw_o_1p_vec~=0)),'color',[0,
0.4470, 0.7410]);
%         p1 = plot(x_RDE_plt_en,temp_R2F(T_aw_o_vec),'color',[0, 0.4470,
0.7410]);
%         p2 =
plot(x_RDE_plt_en(T_aw_i_vec~=0),temp_R2F(T_aw_i_vec(T_aw_i_vec~=0)),'color',
[0 0 0],'linestyle','--');
%         p3_1p =
plot(RL10rgn.x_plt_1p,temp_R2F(RL10rgn.T_aw_1p(RL10rgn.T_aw_1p~=0)),'color',[
0.6350, 0.0780, 0.1840]);
%         p3 =
plot(RL10rgn.x_plt_2p,temp_R2F(RL10rgn.T_aw_2p),'color',[0.6350, 0.0780,
0.1840]);
        p1_1p =
plot(x_RDE_plt_en_1p,temp_conv(T_aw_o_1p_vec(T_aw_o_1p_vec~=0),'R2K'),'color'
,[0, 0.4470, 0.7410]);
        p1 = plot(x_RDE_plt_en,temp_conv(T_aw_o_vec,'R2K'),'color',[0,
0.4470, 0.7410]);
        p2 =
plot(x_RDE_plt_en(T_aw_i_vec~=0),temp_conv(T_aw_i_vec(T_aw_i_vec~=0),'R2K'),'
color',[0 0 0],'linestyle','--');
        p3_1p =
plot(RL10rgn.x_plt_1p,temp_conv(RL10rgn.T_aw_1p(RL10rgn.T_aw_1p~=0),'R2K'),'c
olor',[0.6350, 0.0780, 0.1840]);
```

```
        p3 =
plot(RL10rgn.x_plt_2p,temp_conv(RL10rgn.T_aw_2p,'R2K'),'color',[0.6350,
0.0780, 0.1840]);
        xlabel('Distance from Throat [in]')
%         ylabel('T_{aw} [^oF]')
        ylabel('T_{aw} [K]')
%         title('Adiabatic Wall Temperature')
        legend([p1 p2 p3],'RDE, Outer','RDE,
Inner','RL10','location','northeast')

        f35 = figure('units','normalized','outerposition',[0 0.1 0.7 0.7]);
        hold on
        p1 = plot(x_RDE_plt_en,D_h_c_st_o_vec,'color',clr_out_crct);
        p2 =
plot(x_RDE_plt_en(D_h_c_st_i_vec~=0),D_h_c_st_i_vec(D_h_c_st_i_vec~=0),'color
',clr_inr_crct);
        p3 = plot(RL10rgn.x_plt_2p,RL10rgn.Dhc_2p,'color',[0 0
0],'linestyle','--');
        xlabel('Distance from Throat [in]')
        ylabel('D_{h,c} [in]')
        ylim([0 0.4])
%         title('Cooling Channel Hydraulic Diameter (Long Only)')
        legend([p1 p2 p3],'RDE, Outer','RDE,
Inner','RL10','location','northeast')


    end

    if prnt_regen_rslts
        delP_o = Pclnt_o_1p_vec(i_regen_strt)-Pclnt_o_vec(1);
        delP_i = Pclnt_i_vec(i_arspk)-Pclnt_i_vec(1);
        delP_RL10calc = max(RL10rgn.P_c_1p)-min(RL10rgn.P_c_2p);

        delT_o = T_c_st_o_vec(1)-T_c_st_o_1p_vec(i_regen_strt);
        delT_i = T_c_st_i_vec(1)-T_c_st_i_vec(i_arspk);
        delT_RL10calc = max(RL10rgn.T_c_2p)-
min(RL10rgn.T_c_1p(RL10rgn.T_c_1p~=0));

        fprintf('RDE Regen Results:\n\n')

        fprintf('  Chf = %0.3f\n\n',Chf)

        fprintf('  Coolant:\n')
        fprintf('    delP, outer = %0.3f [psid] or  %0.3f [Pa] vs. RL10 Calc
delP = %0.3f [psid]\n',delP_o,press_conv(delP_o,'eng2SI'),delP_RL10calc)
        fprintf('    P, outer, inlet = %0.3f
[psia]\n',Pclnt_o_1p_vec(i_regen_strt),press_conv(Pclnt_o_1p_vec(i_regen_strt
),'eng2SI'))
        fprintf('    P, outer, exit = %0.3f
[psia]\n',Pclnt_o_vec(1),press_conv(Pclnt_o_vec(1),'eng2SI'))
        fprintf('    delP, inner = %0.3f [psid] or  %0.3f [Pa] vs. RL10 Calc
delP = %0.3f [psid]\n',delP_i,press_conv(delP_i,'eng2SI'),delP_RL10calc)
        fprintf('    P, inner, inlet = %0.3f [psia] or %0.3f
[Pa]\n',Pclnt_i_vec(i_arspk),press_conv(Pclnt_i_vec(i_arspk),'eng2SI'))
```

```
        fprintf('    P, inner, exit = %0.3f [psia] or %0.3f
[Pa]\n',Pclnt_i_vec(1),press_conv(Pclnt_i_vec(1),'eng2SI'))
        fprintf('    delT, outer = %0.3f [oF] or [oR] (%0.3f [K]) vs. RL10
Calc delT = %0.3f [oF] or [oR] (%0.3f
[K])\n',delT_o,temp_conv(delT_o,'R2K'),delT_RL10calc,temp_conv(delT_RL10calc,
'R2K'))
        fprintf('    T, outer, inlet = %0.3f [K] or %0.3f
[oR]\n',temp_conv(T_c_st_o_1p_vec(i_regen_strt),'R2K'),T_c_st_o_1p_vec(i_rege
n_strt))
        fprintf('    T, outer, exit = %0.3f [K] or %0.3f
[oR]\n',temp_conv(T_c_st_o_vec(1),'R2K'),T_c_st_o_vec(1))
        fprintf('    delT, inner = %0.3f [oF] or [oR] (%0.3f [K]) vs. RL10
Calc delT = %0.3f [oF] or [oR] (%0.3f
[K])\n',delT_i,temp_conv(delT_i,'R2K'),delT_RL10calc,temp_conv(delT_RL10calc,
'R2K'))
        fprintf('    T, inner, inlet = %0.3f [K] or %0.3f
[oR]\n',temp_conv(T_c_st_i_vec(i_arspk),'R2K'),T_c_st_i_vec(i_arspk))
        fprintf('    T, inner, exit = %0.3f [K] or %0.3f
[oR]\n\n',temp_conv(T_c_st_i_vec(1),'R2K'),T_c_st_i_vec(1))

        fprintf('    Mass Flow Rate Split:\n')
        fprintf('      Outer Circuit: %0.3f [%%] which is %0.3f [lbm/s]
or %0.3f
[kg/s]\n',mdot_o_frctn*100,mdot_c*mdot_o_frctn,mass_en2SI(mdot_c*mdot_o_frctn
))
        fprintf('      Inner Circuit: %0.3f [%%] which is %0.3f [lbm/s]
or %0.3f [kg/s]\n\n',(1-mdot_o_frctn)*100,mdot_c*(1-
mdot_o_frctn),mass_en2SI(mdot_c*(1-mdot_o_frctn)))

        fprintf('  Hot-gas-side Wall:\n')
        fprintf('    RDE max T_wg, Outer = %0.2f
[K]\n',temp_conv(max(T_wg_o_vec),'R2K'))
        fprintf('    RDE max T_wg, Inner = %0.2f
[K]\n',temp_conv(max(T_wg_i_vec),'R2K'))
        fprintf('    RL10 max T_wg = %0.2f
[K]\n\n',temp_conv(max(RL10rgn.T_wg_2p),'R2K'))

        fprintf('  Coolant-side Wall:\n')
        fprintf('    RDE max T_wc, Outer = %0.2f
[K]\n',temp_conv(max(T_wc_st_o_vec),'R2K'))
        fprintf('    RDE max T_wc, Inner = %0.2f
[K]\n',temp_conv(max(T_wc_st_i_vec),'R2K'))
        fprintf('    RL10 max T_wc = %0.2f
[K]\n\n',temp_conv(max(RL10rgn.T_wc_2p),'R2K'))

        fprintf('  Coolant Temp:\n')
        fprintf('    RDE max T_c, Outer = %0.2f
[K]\n',temp_conv(max(T_c_st_o_vec),'R2K'))
        fprintf('    RDE max T_c, Inner = %0.2f
[K]\n',temp_conv(max(T_c_st_i_vec),'R2K'))
        fprintf('    RL10 max T_c = %0.2f
[K]\n\n',temp_conv(max(RL10rgn.T_c_2p),'R2K'))

        fprintf('  Pre-Turbine Coolant:\n')
        T_c_mix = temp_conv(T_c_st_o_vec(1)*mdot_o_frctn+T_c_st_i_vec(1)*(1-
mdot_o_frctn),'R2K');       % [K]
```

```
        v_c_mix = vel_conv(v_c_st_o_vec(1)*mdot_o_frctn+v_c_st_i_vec(1)*(1-
mdot_o_frctn),'engin2SI');   % [m/s]
        P_c_mix = press_conv(Pclnt_o_vec(1)*mdot_o_frctn+Pclnt_i_vec(1)*(1-
mdot_o_frctn),'eng2SI')/1e3; % [kPa]
        fprintf('    Mixed T_c = %0.3f [K]\n',T_c_mix)
        fprintf('    Mixed v_c = %0.3f [m/s]\n',v_c_mix)
        fprintf('    Mixed P_clnt = %0.3f [kPa]\n\n',P_c_mix)

        fprintf('  Heat Transfer:\n')
        fprintf('    RDE max hg = %0.2f [W/(m^2*K)] vs. RL10 max hg = %0.2f
[W/(m^2*K)]\n',max([max(cnvctvHtXfer_conv(hg_st_o_vec,'eng2SI'))
max(cnvctvHtXfer_conv(hg_st_i_vec,'eng2SI'))]),max(cnvctvHtXfer_conv(RL10rgn.
hg_2p,'eng2SI')))
        fprintf('      RDE max hg at x = %0.2f
[in]\n',x_RDE_plt_en(hg_st_i_vec==max(hg_st_i_vec)))
        fprintf('      Inner Circuit T_aw at max hg is %0.2f
[K]\n',temp_conv(T_aw_i_vec(hg_st_i_vec==max(hg_st_i_vec)),'R2K'))
        fprintf('      Inner Circuit T_wg at max hg is %0.2f
[K]\n',temp_conv(T_wg_i_vec(hg_st_i_vec==max(hg_st_i_vec)),'R2K'))
        fprintf('      Inner Circuit del T at max hg is %0.2f
[K]\n',temp_conv(T_aw_i_vec(hg_st_i_vec==max(hg_st_i_vec)),'R2K')-
temp_conv(T_wg_i_vec(hg_st_i_vec==max(hg_st_i_vec)),'R2K'))
        fprintf('    RL10 hg max at x = %0.3f
[in]\n',RL10rgn.x_plt_2p(RL10rgn.hg_2p==max(RL10rgn.hg_2p)))
        fprintf('      RL10 T_aw at max hg is %0.2f
[K]\n',temp_conv(RL10rgn.T_aw_2p(RL10rgn.hg_2p==max(RL10rgn.hg_2p)),'R2K'))
        fprintf('      RL10 T_wg at max hg is %0.2f
[K]\n',temp_conv(RL10rgn.T_wg_2p(RL10rgn.hg_2p==max(RL10rgn.hg_2p)),'R2K'))
        fprintf('      RL10 del T at max hg is %0.2f
[K]\n',temp_conv(RL10rgn.T_aw_2p(RL10rgn.hg_2p==max(RL10rgn.hg_2p)),'R2K')-
temp_conv(RL10rgn.T_wg_2p(RL10rgn.hg_2p==max(RL10rgn.hg_2p)),'R2K'))
        fprintf('    RDE max q = %0.2f [MW/m^2] vs. RL10 max q = %0.2f
[MW/m^2]\n',max([max(htFlx_conv(q_g_st_o_vec,'eng2SI'))
max(htFlx_conv(q_g_st_i_vec,'eng2SI'))])/1e6,max(htFlx_conv(RL10rgn.q_2p,'eng
2SI'))/1e6)

        Q_tot = Q_st_o_vec(1)+Q_st_i_vec(1);  % [Btu/s] total heat rate
        Q_c = Q_st_o_vec(1)-Q_st_o_vec(x_RDE_plt_en==0)+Q_st_i_vec(1)-
Q_st_i_vec(x_RDE_plt_en==0);  % [Btu/s] chamber heat rate
        Q_c_o = Q_st_o_vec(1)-Q_st_o_vec(x_RDE_plt_en==0);  % [Btu/s] chamber
heat rate, outer
        Q_c_i = Q_st_i_vec(1)-Q_st_i_vec(x_RDE_plt_en==0);  % [Btu/s] chamber
heat rate, inner
        Q_pn = Q_st_i_vec(x_RDE_plt_en==0);  % [Btu/s] plug nozzle
        Q_dn = Q_st_o_vec(x_RDE_plt_en==0);  % [Btu/s] divering nozzle
        Q_pdn = Q_pn + Q_dn;

        fprintf('    RDE max Q = %0.3f [MW] or %0.3f [Btu/s] vs. RL10 max Q
= %0.2f [MW] or %0.3f
[Btu/s]\n',htRt_conv(Q_tot,'eng2SI')/1e6,Q_tot,htRt_conv(RL10rgn.Q_2p(1),'eng
2SI')/1e6,RL10rgn.Q_2p(1))
        fprintf('    RDE chamber Q = %0.3f [MW] or %0.3f
[Btu/s]\n',htRt_conv(Q_c,'eng2SI')/1e6,Q_c)
        fprintf('    RDE chamber inner Q = %0.3f [MW] or %0.3f
[Btu/s]\n',htRt_conv(Q_c_i,'eng2SI')/1e6,Q_c_i)
```

```
        fprintf('    RDE chamber outer Q = %0.3f [MW] or %0.3f
[Btu/s]\n',htRt_conv(Q_c_o,'eng2SI')/1e6,Q_c_o)
        fprintf('    RDE inner Q = %0.3f [MW] or %0.3f
[Btu/s]\n',htRt_conv(Q_st_i_vec(1),'eng2SI')/1e6,Q_st_i_vec(1))
        fprintf('    RDE outer Q = %0.3f [MW] or %0.3f
[Btu/s]\n',htRt_conv(Q_st_o_vec(1),'eng2SI')/1e6,Q_st_o_vec(1))
        fprintf('    RDE plug nozzle Q = %0.3f [MW] or %0.3f
[Btu/s]\n',htRt_conv(Q_pn,'eng2SI')/1e6,Q_pn)
        fprintf('    RDE diverging nozzle Q = %0.3f [MW] or %0.3f
[Btu/s]\n',htRt_conv(Q_dn,'eng2SI')/1e6,Q_dn)
        fprintf('    RDE plug & diverging nozzle Q = %0.3f [MW] or %0.3f
[Btu/s]\n\n',htRt_conv(Q_pn+Q_dn,'eng2SI')/1e6,Q_pn+Q_dn)

        L_chmbr = len_conv(abs(x_RDE_plt_en(1)),'engin2m')*1e3; % [mm] length
chamber
        L_pn = len_conv(x_RDE_plt_en(i_arspk),'engin2m')*1e3; % [mm] length
plug nozzle
        L_dn = len_conv(x_RDE_plt_en(end),'engin2m')*1e3; % [mm] length
diverging nozzle

        fprintf('  Ratios:\n')
        fprintf('    max hg: RDE vs. RL10
= %0.3f\n',max([max(cnvctvHtXfer_conv(hg_st_o_vec,'eng2SI'))
max(cnvctvHtXfer_conv(hg_st_i_vec,'eng2SI'))])/max(cnvctvHtXfer_conv(RL10rgn.
hg_2p,'eng2SI')))
        fprintf('    max q: RDE vs. RL10
= %0.3f\n',max([max(htFlx_conv(q_g_st_o_vec,'eng2SI'))
max(htFlx_conv(q_g_st_i_vec,'eng2SI'))])/max(htFlx_conv(RL10rgn.q_2p,'eng2SI'
)))
        fprintf('    del T at max hg: RDE vs. RL10
= %0.3f\n',(temp_conv(T_aw_i_vec(hg_st_i_vec==max(hg_st_i_vec)),'R2K')-
temp_conv(T_wg_i_vec(hg_st_i_vec==max(hg_st_i_vec)),'R2K'))/(temp_conv(RL10rg
n.T_aw_2p(RL10rgn.hg_2p==max(RL10rgn.hg_2p)),'R2K')-
temp_conv(RL10rgn.T_wg_2p(RL10rgn.hg_2p==max(RL10rgn.hg_2p)),'R2K')))
        fprintf('    max T_wg: RDE, Inner vs. RL10
= %0.3f\n',max(T_wg_i_vec)/max(RL10rgn.T_wg_2p));
        fprintf('    max T_wc: RDE, Inner vs. RL10
= %0.3f\n',max(T_wc_st_i_vec)/max(RL10rgn.T_wc_2p));
        fprintf('    max T_c: RDE, Inner vs. RL10
= %0.3f\n',max(T_c_st_i_vec)/max(RL10rgn.T_c_2p));
        fprintf('    del T_c: RDE, Inner vs. RL10
= %0.3f\n',delT_i/delT_RL10calc);
        fprintf('    Exit Tc: RDE vs. RL10
= %0.3f\n',(T_c_st_o_vec(1)*mdot_o_frctn+T_c_st_i_vec(1)*(1-
mdot_o_frctn))/RL10rgn.T_c_2p(1));
        fprintf('    delT,o: RDE vs. RL10 = %0.3f\n',delT_o/delT_RL10calc);
        fprintf('    delT,i: RDE vs. RL10 = %0.3f\n',delT_i/delT_RL10calc);
        fprintf('    delP,both: RDE vs. RL10 = %0.3f\n',mean([delP_o
delP_i])/delP_RL10calc);
        fprintf('    Q: RDE vs. RL10 = %0.3f\n',Q_tot/RL10rgn.Q_2p(1));
        fprintf('    Q: RDE chamber vs. RDE tot = %0.3f\n',Q_c/Q_tot);
        fprintf('    Q: RDE inner vs. RDE tot
= %0.3f\n',Q_st_i_vec(1)/Q_tot);
        fprintf('    Q: RDE outer vs. RDE tot
= %0.3f\n',Q_st_o_vec(1)/Q_tot);
```

```
        fprintf('    Q: RDE chamber inner vs. RDE chamber outer
= %0.3f\n',Q_c_i/Q_c_o);
        fprintf('    Q: RDE inner vs. RDE outer
= %0.3f\n',Q_st_i_vec(1)/Q_st_o_vec(1));
        fprintf('    Q: RDE inner vs. RDE tot
= %0.3f\n',Q_st_i_vec(1)/Q_tot);
        fprintf('    Q: RDE outer vs. RDE tot
= %0.3f\n',Q_st_o_vec(1)/Q_tot);

        fprintf('    Q RDE chamber inner vs. L RDE chamber  = %0.3f
[kW/mm]\n',(htRt_conv(Q_c_i,'eng2SI')/1e3)/L_chmbr);
        fprintf('    Q RDE chamber outer vs. L RDE chamber  = %0.3f
[kW/mm]\n',(htRt_conv(Q_c_o,'eng2SI')/1e3)/L_chmbr);
        fprintf('    Q RDE chamber vs. L RDE chamber  = %0.3f
[kW/mm]\n',(htRt_conv(Q_c,'eng2SI')/1e3)/L_chmbr);
        fprintf('    Q RDE plug nozzle vs. L RDE plug nozzle  = %0.3f
[kW/mm]\n',(htRt_conv(Q_pn,'eng2SI')/1e3)/L_pn);
        fprintf('    Q RDE diverging nozzle vs. L RDE diverging nozzle
= %0.3f [kW/mm]\n\n',(htRt_conv(Q_dn,'eng2SI')/1e3)/L_dn);
    end

end
```

9) *intlz_rgn_vecs_RDE_2*

```
% Initialzes RDE regen module parameter vectors (mostly for debugging)

T_c_st_o_1p_vec = zeros(length(h_g(:)),1);
% T_c_st_o_1p_vec(i_regen_strt) = temp_conv(-423,'F2R');     % liquid
hydrogen temperature [R] (was 1 instead of end)
% T_c_st_o_1p_vec(i_regen_strt) = 57;     % starting hydrogen temperature per
Binder [R]
T_g_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
T_aw_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
T_wc_st_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
T_wg_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
q_g_st_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
h_c_st_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
hg_st_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
Pclnt_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
f_D_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
D_h_c_st_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
v_c_st_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
dQ_st_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
Q_st_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
dA_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
dA_o_back_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
A_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
A_chnl_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
Per_chnl_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
hg_kd_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
egn_per_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
chnl_per_tot_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
Cp_c_st_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
Pr_c_st_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
rho_c_st_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
mu_c_st_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
Re_c_st_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);
h_c_st_o_1p_vec = zeros(length(T_c_st_o_1p_vec),1);

T_c_st_o_vec = zeros(length(h_g(:)),1);
T_c_st_o_vec = zeros(length(T_c_st_o_vec),1);     % liquid hydrogen
temperature [R] (was 1 instead of end)
T_g_o_vec = zeros(length(T_c_st_o_vec),1);
T_aw_o_vec = zeros(length(T_c_st_o_vec),1);
T_wc_st_o_vec = zeros(length(T_c_st_o_vec),1);
T_wg_o_vec = zeros(length(T_c_st_o_vec),1);
q_g_st_o_vec = zeros(length(T_c_st_o_vec),1);
h_c_st_o_vec = zeros(length(T_c_st_o_vec),1);
hg_st_o_vec = zeros(length(T_c_st_o_vec),1);
Pclnt_o_vec = zeros(length(T_c_st_o_vec),1);
f_D_o_vec = zeros(length(T_c_st_o_vec),1);
D_h_c_st_o_vec = zeros(length(T_c_st_o_vec),1);
v_c_st_o_vec = zeros(length(T_c_st_o_vec),1);
dQ_st_o_vec = zeros(length(T_c_st_o_vec),1);
Q_st_o_vec = zeros(length(T_c_st_o_vec),1);
dA_o_vec = zeros(length(T_c_st_o_vec),1);
dA_o_back_vec = zeros(length(T_c_st_o_vec),1);
```

```
A_o_vec = zeros(length(T_c_st_o_vec),1);
A_chnl_o_vec = zeros(length(T_c_st_o_vec),1);
Per_chnl_o_vec = zeros(length(T_c_st_o_vec),1);
hg_kd_o_vec = zeros(length(T_c_st_o_vec),1);
egn_per_o_vec = zeros(length(T_c_st_o_vec),1);
chnl_per_tot_o_vec = zeros(length(T_c_st_o_vec),1);
Cp_c_st_o_vec = zeros(length(T_c_st_o_vec),1);
Pr_c_st_o_vec = zeros(length(T_c_st_o_vec),1);
rho_c_st_o_vec = zeros(length(T_c_st_o_vec),1);
mu_c_st_o_vec = zeros(length(T_c_st_o_vec),1);
Re_c_st_o_vec = zeros(length(T_c_st_o_vec),1);
h_c_st_o_vec = zeros(length(T_c_st_o_vec),1);

T_c_st_i_vec = zeros(length(h_g(:)),1);
% T_c_st_i_vec(i_arspk) = temp_conv(-423,'F2R');     % liquid hydrogen
temperature [R] (was 1 instead of end)
T_g_i_vec = zeros(length(T_c_st_i_vec),1);
T_aw_i_vec = zeros(length(T_c_st_i_vec),1);
T_wc_st_i_vec = zeros(length(T_c_st_i_vec),1);
T_wg_i_vec = zeros(length(T_c_st_i_vec),1);
q_g_st_i_vec = zeros(length(T_c_st_i_vec),1);
h_c_st_i_vec = zeros(length(T_c_st_i_vec),1);
hg_st_i_vec = zeros(length(T_c_st_i_vec),1);
Pclnt_i_vec = zeros(length(T_c_st_i_vec),1);
f_D_i_vec = zeros(length(T_c_st_i_vec),1);
D_h_c_st_i_vec = zeros(length(T_c_st_i_vec),1);
v_c_st_i_vec = zeros(length(T_c_st_i_vec),1);
dQ_st_i_vec = zeros(length(T_c_st_i_vec),1);
Q_st_i_vec = zeros(length(T_c_st_i_vec),1);
dA_i_vec = zeros(length(T_c_st_i_vec),1);
dA_i_back_vec = zeros(length(T_c_st_i_vec),1);
A_i_vec = zeros(length(T_c_st_i_vec),1);
A_chnl_i_vec = zeros(length(T_c_st_i_vec),1);
Per_chnl_i_vec = zeros(length(T_c_st_i_vec),1);
hg_kd_i_vec = zeros(length(T_c_st_i_vec),1);
egn_per_i_vec = zeros(length(T_c_st_i_vec),1);
chnl_per_tot_i_vec = zeros(length(T_c_st_i_vec),1);
Cp_c_st_i_vec = zeros(length(T_c_st_i_vec),1);
Pr_c_st_i_vec = zeros(length(T_c_st_i_vec),1);
rho_c_st_i_vec = zeros(length(T_c_st_i_vec),1);
mu_c_st_i_vec = zeros(length(T_c_st_i_vec),1);
Re_c_st_i_vec = zeros(length(T_c_st_i_vec),1);
h_c_st_i_vec = zeros(length(T_c_st_i_vec),1);
```

10) asgn_vec_vals_o_1p_RDE

```
% Assigns RDE outer wall current iteration values to tube collection vectors
for 2nd pass/long tubes

T_g_o_1p_vec(i) = Tg_st;
T_aw_o_1p_vec(i) = T_aw_st;
T_wc_st_o_1p_vec(i) = T_wc_st;
T_wg_o_1p_vec(i) = T_wg_st - T_wg_inc;
q_g_st_o_1p_vec(i) = q_g_st;
h_c_st_o_1p_vec(i) = h_c_st;
hg_st_o_1p_vec(i) = hg_st;
D_h_c_st_o_1p_vec(i) = D_h_c_st;
v_c_st_o_1p_vec(i) = v_c_st;
f_D_o_1p_vec(i) = f_D*fdfctr_Bndr;
dQ_st_o_1p_vec(i) = dQ_st;             % [Btu/s]
dA_o_1p_vec(i) = dA;
dA_o_back_1p_vec(i) = dA_back;
A_chnl_o_1p_vec(i) = A_chnl_o;        %[in^2]
Per_chnl_o_1p_vec(i) = Per_chnl_o;    % [in]
hg_kd_o_1p_vec(i) = hg_kd_Bndr*hg_kd_RDE;
egn_per_o_1p_vec(i) = eng_per;
chnl_per_tot_o_1p_vec(i) = chnl_per_tot;
Cp_c_st_o_1p_vec(i) = Cp_c_st;
Pr_c_st_o_1p_vec(i) = Pr_c_st;
rho_c_st_o_1p_vec(i) = rho_c_st;
mu_c_st_o_1p_vec(i) = mu_c_st;
Re_c_st_o_1p_vec(i) = Re_c_st;
h_c_st_o_1p_vec(i) = h_c_st;
```

11) asgn_vec_vals_o_RDE

```
% Assigns RDE outer wall current iteration values to tube collection vectors
for 2nd pass/long tubes

T_g_o_vec(i) = Tg_st;
T_aw_o_vec(i) = T_aw_st;
T_wc_st_o_vec(i) = T_wc_st;
T_wg_o_vec(i) = T_wg_st - T_wg_inc;
q_g_st_o_vec(i) = q_g_st;
h_c_st_o_vec(i) = h_c_st;
hg_st_o_vec(i) = hg_st;
D_h_c_st_o_vec(i) = D_h_c_st;
v_c_st_o_vec(i) = v_c_st;
f_D_o_vec(i) = f_D*fdfctr_Bndr;
dQ_st_o_vec(i) = dQ_st;            % [Btu/s]
dA_o_vec(i) = dA;
dA_o_back_vec(i) = dA_back;
A_chnl_o_vec(i) = A_chnl_o;       %[in^2]
Per_chnl_o_vec(i) = Per_chnl_o;   % [in]
hg_kd_o_vec(i) = hg_kd_Bndr*hg_kd_RDE;
egn_per_o_vec(i) = eng_per;
chnl_per_tot_o_vec(i) = chnl_per_tot;
Cp_c_st_o_vec(i) = Cp_c_st;
Pr_c_st_o_vec(i) = Pr_c_st;
rho_c_st_o_vec(i) = rho_c_st;
mu_c_st_o_vec(i) = mu_c_st;
Re_c_st_o_vec(i) = Re_c_st;
h_c_st_o_vec(i) = h_c_st;
```

12) asgn_vec_vals_i_RDE

```
% Assigns RDE inner wall current iteration values to tube collection vectors
for 2nd pass/long tubes

T_g_i_vec(i) = Tg_st;
T_aw_i_vec(i) = T_aw_st;
T_wc_st_i_vec(i) = T_wc_st;
T_wg_i_vec(i) = T_wg_st - T_wg_inc;
q_g_st_i_vec(i) = q_g_st;
h_c_st_i_vec(i) = h_c_st;
hg_st_i_vec(i) = hg_st;
D_h_c_st_i_vec(i) = D_h_c_st;
v_c_st_i_vec(i) = v_c_st;
f_D_i_vec(i) = f_D*fdfctr_Bndr;
dQ_st_i_vec(i) = dQ_st;             % [Btu/s]
dA_i_vec(i) = dA;
dA_i_back_vec(i) = dA_back;
A_chnl_i_vec(i) = A_chnl_i;           % [in^2]
Per_chnl_i_vec(i) = Per_chnl_i;       % [in]
hg_kd_i_vec(i) = hg_kd_Bndr*hg_kd_RDE;
egn_per_i_vec(i) = eng_per;
chnl_per_tot_i_vec(i) = chnl_per_tot;
Cp_c_st_i_vec(i) = Cp_c_st;
Pr_c_st_i_vec(i) = Pr_c_st;
rho_c_st_i_vec(i) = rho_c_st;
mu_c_st_i_vec(i) = mu_c_st;
Re_c_st_i_vec(i) = Re_c_st;
h_c_st_i_vec(i) = h_c_st;
```

M) trbnPwr

```
function [xxxx] =
trbnPwr(mdot_c,T_c_in_RL10,T_c_in_mix_RDE,P_c_in_RL10,P_c_in_RDE,v_c,prnt_trn
bpwrcalcs)

% REFPROP P in [kPa] and T in [K], v_c is mixed RDE coolant velocity,
% mdot_c in [kg/s]

Cp_RL10 = refpropm('C','T',T_c_in_RL10,'P',P_c_in_RL10,'hydrogen.fld');    %
[J/(kg*K)], isobaric specific heat;
Cp_RDE = refpropm('C','T',T_c_in_mix_RDE,'P',P_c_in_RDE,'hydrogen.fld');    %
[J/(kg*K)], isobaric specific heat;

gam_RL10 = refpropm('K','T',T_c_in_RL10,'P',P_c_in_RL10,'hydrogen.fld');    %
isobaric to isochoric specific heat ratio;
gam_RDE =
refpropm('K','T',T_c_in_mix_RDE,'P',P_c_in_RDE,'hydrogen.fld');    % isobaric
to isochoric specific heat ratio;

R_u = 8.31432;      % universal gas constant [J/(mol*K)]
mw_H2 = 2.01588;    % H2 molecular weight [g/mol]
R_sp = R_u/mw_H2*1e3;  % H2 specific gas constant [J/(kg*K)]

a_RL10 = sqrt(gam_RL10*R_sp*T_c_in_RL10);   % local speed of sound [m/s]
M_RL10 = v_c/a_RL10;                        % local Mach number

a_RDE = sqrt(gam_RDE*R_sp*T_c_in_mix_RDE);   % local speed of sound [m/s]
M_RDE = v_c/a_RDE;                         % local Mach number

T_t1_RL10 = T_c_in_RL10*(1+((gam_RL10-1)/2)*M_RL10^2);  % total/stagnation
inlet temp [K]
P_t1_RL10 = (P_c_in_RL10*1e3)*(1+((gam_RL10-
1)/2)*M_RL10^2)^(gam_RL10/(gam_RL10-1));    % total/stagnation inlet pressure
[Pa]
TPR = 1.39;     % Turbine Pressure Ratio, from Binder
P_t2_RL10 = P_t1_RL10/TPR;  % total/stagnation exit pressure [Pa]

T_t1_RDE = T_c_in_mix_RDE*(1+((gam_RDE-1)/2)*M_RDE^2);  % total/stagnation
inlet temp [K]
P_t1_RDE = (P_c_in_RDE*1e3)*(1+((gam_RDE-1)/2)*M_RDE^2)^(gam_RDE/(gam_RDE-
1));    % total/stagnation inlet pressure [Pa]
P_t2_RDE_1 = P_t1_RDE/TPR;    % total/stagnation inlet pressure [Pa]
P_t2_RDE_2 = P_t2_RL10;    % total/stagnation inlet pressure [Pa]

N_RL10 = mdot_c*Cp_RL10*T_t1_RL10*(1+(P_t2_RL10/P_t1_RL10)^((gam_RL10-
1)/gam_RL10)); % [W]
N_RDE_1 = mdot_c*Cp_RDE*T_t1_RDE*(1+(P_t2_RDE_1/P_t1_RDE)^((gam_RDE-
1)/gam_RDE)); % using TPR [W]
N_RDE_2 = mdot_c*Cp_RDE*T_t1_RDE*(1+(P_t2_RDE_2/P_t1_RDE)^((gam_RDE-
1)/gam_RDE)); % setting RDE's P_t2 to RL10's P_t2 [W]
```

```
h_RL10 = refpropm('H','T',T_c_in_RL10,'P',P_c_in_RL10,'hydrogen.fld');     %
[J/kg], enthalpy;
h_RDE = refpropm('H','T',T_c_in_mix_RDE,'P',P_c_in_RDE,'hydrogen.fld');    %
[J/kg], enthalpy;

h_t1_RL10 = h_RL10 + v_c^2/2;     % [J/kg], stagnation enthalpy;
h_t1_RDE = h_RDE + v_c^2/2;       % [J/kg], stagnation enthalpy;

if prnt_trnbpwrcalcs
    fprintf('Turbine Calcs:\n\n')

    fprintf('RL10:\n')
    fprintf('  mdot = %0.3f [kg/s]\n',mdot_c)
    fprintf('  T = %0.3f [K]\n',T_c_in_RL10)
    fprintf('  P = %0.3f [kPa] or %0.3f
[psia]\n',P_c_in_RL10,press_conv(P_c_in_RL10*1e3,'SI2eng'))
    fprintf('  Cp = %0.3f [J/(kg*K)]\n',Cp_RL10)
    fprintf('  gamma = %0.3f\n',gam_RL10)
    fprintf('  h_t1 = %0.3f [J/kg]\n',h_t1_RL10)
    fprintf('  vel = %0.3f [m/s]\n',v_c)
    fprintf('  R_sp = %0.3f [J/(kg*K)]\n',R_sp)
    fprintf('  a = %0.3f [m/s]\n',a_RL10)
    fprintf('  M = %0.3f\n',M_RL10)
    fprintf('  T_t1 = %0.3f [K]\n',T_t1_RL10)
    fprintf('  P_t1 = %0.3f [Pa] or %0.3f
[psia]\n',P_t1_RL10,press_conv(P_t1_RL10,'SI2eng'))
    fprintf('  P_t2 = %0.3f [Pa] or %0.3f
[psia]\n',P_t2_RL10,press_conv(P_t2_RL10,'SI2eng'))
    fprintf('  N = %0.3f [MW]\n\n',N_RL10/1e6)

    fprintf('RDE:\n')
    fprintf('  mdot = %0.3f [kg/s]\n',mdot_c)
    fprintf('  T = %0.3f [K]\n',T_c_in_mix_RDE)
    fprintf('  P = %0.3f [kPa] or %0.3f
[psia]\n',P_c_in_RDE,press_conv(P_c_in_RDE*1e3,'SI2eng'))
    fprintf('  Cp = %0.3f [J/(kg*K)]\n',Cp_RDE)
    fprintf('  gamma = %0.3f\n',gam_RDE)
    fprintf('  h_t1 = %0.3f [J/kg]\n',h_t1_RDE)
    fprintf('  vel = %0.3f [m/s]\n',v_c)
    fprintf('  R_sp = %0.3f [J/(kg*K)]\n',R_sp)
    fprintf('  a = %0.3f [m/s]\n',a_RDE)
    fprintf('  M = %0.3f\n',M_RDE)
    fprintf('  T_t1 = %0.3f [K]\n',T_t1_RDE)
    fprintf('  P_t1 = %0.3f [Pa] or %0.3f
[psia]\n',P_t1_RDE,press_conv(P_t1_RDE,'SI2eng'))
    fprintf('  P_t2 using TPR = %0.3f [Pa] or %0.3f
[psia]\n',P_t2_RDE_1,press_conv(P_t2_RDE_1,'SI2eng'))
    fprintf('  P_t2 set equal to RL10 = %0.3f [Pa] or %0.3f
[psia]\n',P_t2_RDE_2,press_conv(P_t2_RDE_2,'SI2eng'))
    fprintf('  N_1 (using TPR) = %0.3f [MW]\n',N_RDE_1/1e6)
    fprintf('  N_2 (setting RDE''s P_t2 to RL10''s P_t2) = %0.3f
[MW]\n\n',N_RDE_2/1e6)

    fprintf('Ratios:\n')
    fprintf('  N (using TPR): RDE_1 vs. RL10 = %0.3f\n',N_RDE_1/N_RL10)
```

```
    fprintf('  N (setting RDE''s P_t2 to RL10''s P_t2): RDE_2 vs. RL10
= %0.3f\n\n',N_RDE_2/N_RL10)
end

xxxx = 'placeholder';

end
```

REFERENCES

[1] Stechmann, David P.; Heister, Stephen D.; and Harroun, Alexis. (2018, October 15). *Rotating Detonation Engine Performance Model for Rocket Applications*. Journal of Spacecraft and Rockets, AIAA.

[2] Turns, Stephen R. (2012). *An Introduction to Combustion: Concepts and Applications*, Third Edition. McGraw-Hill Education.

[3] Jennings, S (1988). *The mean free path in air*. Journal of Aerosol Science.

[4] Shank, Jason C.; King, Paul I.; Karnesky, James; Schauer, Frederick R. (2012, January 9-12). *Development Testing of a Modular Rotating Detonation Engine*. AIAA Aerospace Sciences Meeting, Nashville, Tennessee. AIAA Paper 2012-0120.

[5] Schwer and Kailasanath (2011). *Rotating Detonation-Wave Engines*. NRL Review. Retrieved from: https://www.nrl.navy.mil/content_images/11_FA2.pdf

[6] Napolitano, Jo. (2019, June 6). *Argonne Computational Model To Accelerate Engine Development for Hypersonic Flight*. Retrieved from R&D: https://www.rdmag.com/news/2019/06/argonne-computational-model-accelerate-engine-development-hypersonic-flight

[7] Stevens, Christopher A; Fotia, Mathew L.; Hoke, John L.; Schauer, Frederick R. (2018, January 8-12). *Quasi-Steady Heat Transfer Measurements in an RDE*. AIAA SciTech Forum, AIAA Aerospace Sciences Meeting, Kissimmee, Florida. AIAA Paper 2018-1884.

[8] Pratt & Whitney (2003, September 3). *RL10 Model To Be Retired After Upcoming Flight*. Retrieved from: http://www.defense-aerospace.com/articles-view/release/3/25100/p%26w-to-retire-rl10-after-titan-flight-(sept.-4).html

[9] ULA. *History of the Titan Centaur Launch Vehicle*. Retrieved from: https://www.ulalaunch.com/docs/default-source/upper-stages/history-of-the-titan-centaur-launch-vehicle.pdf

[10] Binder, Michael; Tomsik, Thomas; Veres, Joseph P. (1997, January). RL10A-3-3A Rocket Engine Modeling Project. NASA Technical Memorandum 107318. Retrieved from: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19970010379.pdf

[11] Heister, Stephen D. (2018). RL10 Geometry Data [Personal Communication].

[12] NASA (2016, February 4). Chemical Equilibrium with Applications (CEA).

[13] Piper, Philip (2017). NASA CEA Executable Adaptation [MATLAB Functions].

[14] Heister, Stephen D.; Anderson, William E.; Pourpoint, Timothee. *Rocket Propulsion*. [Textbook] Purdue University, School of Aeronautics and Astronautics

[15] 347 Stainless Steel, annealed and cold drawn, bar. Retrieved from Matweb: http://www.matweb.com/search/datasheet_print.aspx?matguid=6f501e45a6e2442fb3fd3694befd_ba45

[16] Huzel, Dieter and Huang, David (1967). Design of Liquid Propellant Rocket Engines. NASA SP-125. Scientific and Technical Information Division, Office of Technology Utilization, National Aeronautics and Space Administration, Washington, D.C. Retrieved from: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19710019929.pdf

[17] Chernov, Nikolai (2009). CircleFitByTaubin [MATLAB function].

[18] NIST (2010). REFPROP 9 [Database and MATLAB functions].

[19] Brown, Glenn. (2000, June 22). Henry Darcy and His Law. Retrieved from http://bae.okstate.edu/faculty-sites/Darcy/DarcyWeisbach/Darcy-WeisbachEq.htm

[20] Rouse, H (1946). *Elementary Mechanics of Fluids*. John Wiley & Sons.

[21] Babcock & Wilcox Co. (1978). Bend loss coefficients for a pipe.

[22] Jayanti, Sreenivas (2011, March 16). Bends, Flow and Pressure Drop in. Retrieved from http://www.thermopedia.com/content/577/

[23] Hydrogen. NIST Chemistry Webbook, SRD 69. Retrieved from https://webbook.nist.gov/cgi/cbook.cgi?ID=C1333740

[24] Oxygen. NIST Chemistry Webbook, SRD 69. Retrieved from https://webbook.nist.gov/cgi/cbook.cgi?ID=7782-44-7

[25] California Energy Commission. *California ISO Glossary*. Retrieved from: https://www.energy.ca.gov/glossary/ISO_GLOSSARY.PDF

VITA

Timothy P. Gurshin Jr.

EDUCATION

Master of Science in Aeronautics and Astronautics Engineering student in Propulsion at Purdue University, West Lafayette, Indiana. August 2017 – August 2019. Thesis title: "Heating and Regenerative Cooling Model for a Rotating Detonation Engine Designed for Upper Stage Performance."

Bachelor of Science in Engineering with a concentration in Aeronautics and Astronautics, Stanford University, Stanford, California. September 2010 – June 2014.

PROFESSIONAL EXPERIENCE

Lockheed Martin Space Systems Company. November 2014 – June 2017.
Orion Multi-Purpose Crew Vehicle Crew Module propulsion engineer.
SkyFire cubesat. Trajectory and CONOPS analyst.

IN Space LLC. Summer 2018 – Spring 2019.
Aerospace engineer for high pressure fluid testing hydraulic piston design and rotating detonation heat and cooling model.

Stanford Aeronautics and Astronautics Structures and Composites Lab. Summer 2013.
Structural health monitoring undergraduate researcher.

NASA Ames Research Center. Summer 2012.
Mission design and microwave propulsion engineering intern

PROFESSIONAL MEMBERSHIP

Sigma Gamma Tau, the National Honor Society in Aerospace Engineering. April 2018 – Present.
Alpha Chapter, Purdue University.