

KU ScholarWorks

Design, Analysis, and Simulation of Rocket Propulsion System

Item Type	Thesis
Authors	Kulhanek, Sarah Logan
Publisher	University of Kansas
Rights	This item is protected by copyright and unless otherwise specified the copyright of this thesis/dissertation is held by the author.
Download date	2024-08-07 15:00:03
Link to Item	https://hdl.handle.net/1808/10198

Design, Analysis, and Simulation of Rocket Propulsion System

By

Sarah L. Kulhanek

Submitted to the graduate degree program in Aerospace Engineering and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Masters of Science.

Chairperson, Dr. Ray Taghavi

Committee member, Dr. Saeed Farokhi

Committee member, Dr. Shahriar Keshmiri

Date Defended: 6/6/2012

The Thesis Committee for Sarah L. Kulhanek
certifies that this is the approved version of the following thesis:

Design, Analysis, and Simulation of Rocket Propulsion System

Chairperson, Dr. Ray Taghavi,

Date approved: 6/6/2012

Abstract

This document details the functionality of a software program used to streamline a rocket propulsion system design, analysis and simulation effort. The program aids in unifying the nozzle, chamber and injector portions of a rocket propulsion system design effort quickly and efficiently using a streamlined graphical user interface (GUI). The program also allows for the selection of common nozzle profiles including 80% rao, conical, a user selected percentage bell, and a minimum length nozzle (MLN) using method of characteristics (MOC). Chamber dimensions, propellant selections, and injector selection between doublet or triplet allow for further refinement of the desired rocket system design.

The program takes the available selections and specifications made by the user and outputs key design parameters calculated from the input variables. A 2-D graphical representation of the nozzle and/or chamber is plotted and coordinates of the plotted line are displayed. Additional design calculations are determined and displayed within the program such as specific impulse, exhaust velocity, propellant weight flow, fundamental instability frequencies, etc.

The rocket propulsion system design coordinates are saved to a *.dat file which can be used in a CAD program to plot a 3-D model of the rocket propulsion system. The *.dat file is compatible for creating splines in Unigraphics NX, Catia, and SolidWorks. Coordinates of the injectors are saved to a *.dat file to be modeled in a CAD program as well. The program currently provides a symbolic link in the form of a button on the output page which will open Unigraphics NX CAD program.

The post-processing simulation of the rocket propulsion system is done in a computational fluid dynamics (CFD) program such as ANSYS ICEM CFD mesh generation software and ANSYS FLUENT CFD. The program provides a button on the output page which will open the ANSYS ICEM CFD mesh program and the ANSYS FLUENT CFD program. The user inputs the parasolid or IGES/STEP file of the CAD 3-D modeling of the rocket propulsion system into the ANSYS ICEM CFD meshing software. The geometry tolerant mesher program produces a volume or surface mesh to be read into the ANSYS FLUENT CFD software. Using ANSYS FLUENT CFD software, the user can choose to model the flow, turbulence, heat transfer, air flow over the rocket, combustion in the chamber, or various other options of the rocket propulsion system.

The rocket propulsion system is a graphical user interface (GUI) which is run through Matlab and is compatible for 2009-2011 Matlab versions.

Acknowledgements

First and foremost, I would like to thank my advisory committee chairperson, Dr. Ray Taghavi, for all your support, motivation, and guidance throughout this project. Your mentorship, friendship, and dedicated teaching throughout my undergraduate and graduate education at the University of Kansas will always be remembered.

Second, I would like to thank my committee members, Dr. Saeed Farokhi and Dr. Shahriar Keshmiri for their support and guidance throughout this project and my undergraduate and graduate education. The experiences I have gained through this work and my educational career will last a lifetime.

Thirdly, I want to express my deep love and appreciation to my fiancé Jakob Bowden. Thanks for your support, understanding, sacrifice, and encouragement through the hard times. You kept me going and made my stressful work enjoyable. You gave me the strength and motivation to work through the long days and late nights. You mean the world to me.

Also, I would like to express my deep love and appreciation to my family. Thanks to my mother and father for their support and constant love throughout my life. You guys motivated me to stick with it and be the independent strong willed girl you raised me to be. Thanks to my two sisters and brother for their support and encouragement throughout the years. You kept me going through the hard times and took the stressful work off my mind every time you called to talk with me. I would like to dedicate this thesis to my family.

Fourthly, I would like to thank Jeff and Karen Bowden for their love, support, and encouragement through my thesis. Thank you for the generosity of letting me stay with you these past few months and for all the support through the hard times. You helped me get through the late nights and difficult times.

Finally, I would like to thank all my friends in the KU aerospace department for making the last 7 years memorable, especially Phil Rich and Adam Saverino. I enjoyed working with you guys every day and having fun through the good and hard times of school. I will sorely miss seeing you guys every day, but I have made lifelong friends that will always be a part of my life.

Table of Contents

	Page #
Abstract	iii
Acknowledgements	v
List of Figures	viii
List of Figures (continued).....	ix
List of Symbols	x
List of Symbols (continued).....	xi
List of Symbols (continued).....	xii
List of Symbols (continued).....	xiii
1.0 Introduction	1
2.0 Literature Review	6
3.0 Theoretical Considerations.....	8
3.1 Rao Nozzle	8
3.2 Conical Nozzle	10
3.3 Minimum Length Nozzle (MLN) using Method of Characteristics (MOC).....	12
3.4 Combustion Chamber.....	21
3.5 Injectors.....	23
3.6 Propellant Calculations	24
3.7 Resonant Frequency for the 1 st Longitudinal, Radial, and Tangential Modes	25
4.0 Program Details.....	28
4.1 Overall Layout.....	28
4.2 Welcome Tab	30
4.3 Instructions Tab.....	31
4.4 Design Choices Tab	32
4.5 Input selections Tab	34
4.6 Output selections Tab.....	37
5.0 Program Example Run	39
5.1 Nozzle Selections	39
5.2 Combustion Chamber and Nozzle Selection.....	41
5.3 Minimum Length Nozzle (MLN) using Method of Characteristics (MOC).....	43
5.4 Three-Dimensional Modeling	45
5.5 Simulation Computational Fluid Dynamics (CFD) Example	48
6.0 Conclusion and Future Work	50
7.0 References	52
Appendix A: Main GUI Code.....	55
Appendix B: Function for Rao Nozzle	108
Appendix C: function conical	113
Appendix D: function MOC	114
Appendix F: fuel	120

List of Figures

Figure 1-1: Rao Nozzle (Reference 2)	1
Figure 1-2: Conical Nozzle (Reference 2)	2
Figure 1-3: Under-Expanded, Over-Expanded, & Perfectly Expanded Nozzles (Reference 1).....	2
Figure 1-4: Schematic of Supersonic Nozzle Design by the MOC (Reference 3)	3
Figure 1-5: Rocket Cylindrical Combustion Chamber (Reference 4)	4
Figure 1-6: Schematic Diagram of the Doublet and Triplet Injector Types (Reference 2)	5
Figure 3-1: Parabolic Approximation of Bell Nozzle Contour (Reference 4).....	8
Figure 3-2: Schematic Sketch of Conical Nozzle (Reference 4)	11
Figure 3-3: Streamline Geometry	14
Figure 3-4: Left- and Right-Running Characteristic Lines.....	16
Figure 3-5: Unit Processes for MOC	18
Figure 3-6: Schematic of Supersonic Nozzle Design by the MOC (Reference 3)	20
Figure 3-7: Schematic of Minimum-Length Nozzle (Reference 3).....	21
Figure 3-8: The Resonance Modes (Reference 8)	26
Figure 4-1: Layout Program.....	29
Figure 4-2: Welcome Tab	30
Figure 4-3: Instructions Tab.....	31
Figure 4-4: Design Choices Tab	32
Figure 4-5: Layout of the Design choice selections.....	33
Figure 4-6: Input Tab	34
Figure 4-7: Nozzle Input Parameters	35
Figure 4-8: Chamber Input Parameters.....	36
Figure 4-9: Injector Input Parameters.....	36
Figure 4-10: MLN Input Parameters.....	37
Figure 4-11: Output Tab	38
Figure 5-1: Input Parameters	39
Figure 5-2: 80% Rao Nozzle.....	40
Figure 5-3: 50% Rao Nozzle.....	40
Figure 5-4: 15 Degree Half Angle Conical Nozzle	40
Figure 5-5: Chamber and Nozzle Input Parameters.....	41
Figure 5-6: 80% Rao Nozzle with Chamber.....	42
Figure 5-7: 50% Rao Nozzle with Chamber.....	42
Figure 5-8: 15 Degree Half Angle Conical Nozzle with Chamber.....	42
Figure 5-9: MOC Input Parameters	43
Figure 5-10: Two-Dimensional Plot of the MLN using MOC for 25 lines	43
Figure 5-11: Two-Dimensional Plot of the MLN using MOC for 50 lines	44
Figure 5-12: Two-Dimensional Plot of the MLN using MOC for 100 lines	44
Figure 5-13: 3-D Model of a 80% Rao Nozzle and Chamber (Reference 12)	45
Figure 5-14: 3-D Model of 15 Degree Half Angle Conical Nozzle & Chamber (Reference 12). ..	46
Figure 5-15: 3-D Model of Doublet Injector (Reference 12)	47
Figure 5-16: 80% Rao Nozzle Mesh Grid (Reference 14)	48
Figure 5-17: Experimental, Computational Schlieren Images, and Mach contours for Baseline Nozzle Configuration at Various NPRs (Reference 26).....	49

List of Figures (continued)

Figure 5-18: Experimental, Computational Schlieren Images, and Mach contours for Baseline Nozzle Configuration at Various NPRs Continued (Reference 26) 49

List of Symbols

<u>Symbol</u>	<u>Description</u>	<u>Units</u>
a	Coefficient	(~)
a	Speed of sound	ft/s ²
A	Area	ft ²
b	Coefficient	(~)
c	Coefficient	(~)
c*	Characteristic velocity	ft/s
C	Characteristics	(~)
C _d	Discharge coefficient	(~)
D	Diameter	inches
E	Exit point	inches
f	Frequency	Hz
F	Thrust	lbf
g	Gravitational acceleration at sea level	ft/s ²
I _{sp}	Specific Impulse	seconds
k	Ratio of specific heats	(~)
K	Invariants along Characteristic lines	(~)
l	Wavelength	(~)
L	Length	inches
M	Mach number	(~)
MM	Molecular Mass	lbm/lb-mol
n	No. of characteristic lines	(~)

List of Symbols (continued)

<u>Symbol</u>	<u>Description</u>	<u>Units</u>
n	No. of injector holes	(~)
N	Inflection point	inches
p	Pressure	psi
r	Mixture ratio	(~)
R	Radius	inches
R'	Universal gas constant	ft-lb/lb mol-R
t _p	Time of operation	seconds
T	Temperature	Rankine
v ₂	Exhaust velocity	ft/s ²
V	Volume	in ³
Vdot	Volume flow rate	ft ³ /s
w	Width of nozzle	inches
wdot	Weight flow rate	lbf/s
x	x coordinate	inches
y	y coordinate	inches

<u>Greek</u>	<u>Description</u>	<u>Units</u>
γ	Ratio of specific heats	(~)
Δ	Increment difference	(~)
ε	Area ratio	(~)
θ	Inflection angle	degrees

List of Symbols (continued)

<u>Greek</u>	<u>Description</u>	<u>Units</u>
θ	Flow deflection angle	degrees
μ	Mach angle	degrees
ν	Prandtl-Meyer function	degrees
ρ	Density	lbm/ft ³
Φ	Full-velocity potential	degrees

Subscripts

	<u>Description</u>
c	Chamber
char	Characteristic line
e	Exit
f	Fuel
max	Maximum
ML	Minimum length
n	Nozzle
o	Oxidizer
t	Throat
w	Wall
x	X-direction
y	Y-direction
*	Characteristic chamber

List of Symbols (continued)

Subscripts

+	Left-running characteristic line
-	Right-running characteristic line

Description

Acronyms

CFD	Computational Fluid Dynamics
Dat	Data File
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
MLN	Minimum Length Nozzle
MOC	Method of Characteristics
NPR	Nozzle Pressure Ratio
RPA	Rocket Propulsion Analysis

Description

1.0 Introduction

Rocket propulsion system design pertains to conical, 80% Rao nozzle, percentage of contour bell nozzle, and method of characteristics (MOC) of a minimum length nozzle including the chamber and injectors calculations. This thesis presents a program that the user chooses input parameters pertinent to design a rocket nozzle and runs calculations that are then used to create a 3-D model and to perform CFD analysis.

The history of rocket nozzles, specifically rao nozzle comes from G. V. Rao back in 1958, when he derived analytically the wall contour of a nozzle by method of characteristics (Reference 1). The bell contour shape nozzle minimized the losses of the internal shock waves in the supersonic flow. According to Reference 1, bell shaped nozzles are used today for rocket nozzles since the 1960s for both liquid and solid propellant rockets. Conical nozzles were used primarily first before being modified to have a bell shaped exit nozzle. The shape of a Rao nozzle and conical nozzle are shown in Figure 1-1 and Figure 1-2, respectively.

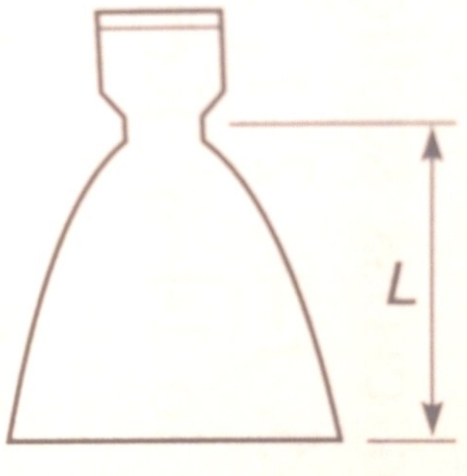


Figure 1-1: Rao Nozzle (Reference 2)

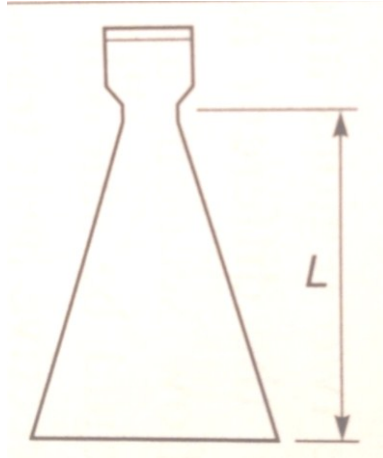


Figure 1-2: Conical Nozzle (Reference 2)

An under-expanded nozzle occurs when the exit pressure is greater than the ambient pressure at high altitudes. The exhaust plume continues to expand past the nozzle exit reducing efficiency. A perfectly expanded nozzle occurs when the exit pressure equals the ambient pressure, which results in maximum efficiency. An over-expanded nozzle occurs when the exit pressure is less than the ambient pressure at low altitudes such as sea level. The exhaust plume is pinched inward in fluid separation from the walls creating compression waves or shock waves inside the diverging nozzle section. (Reference 2) Figure 1-3 below shows the under-expanded, over-expanded, and perfectly expanded nozzles

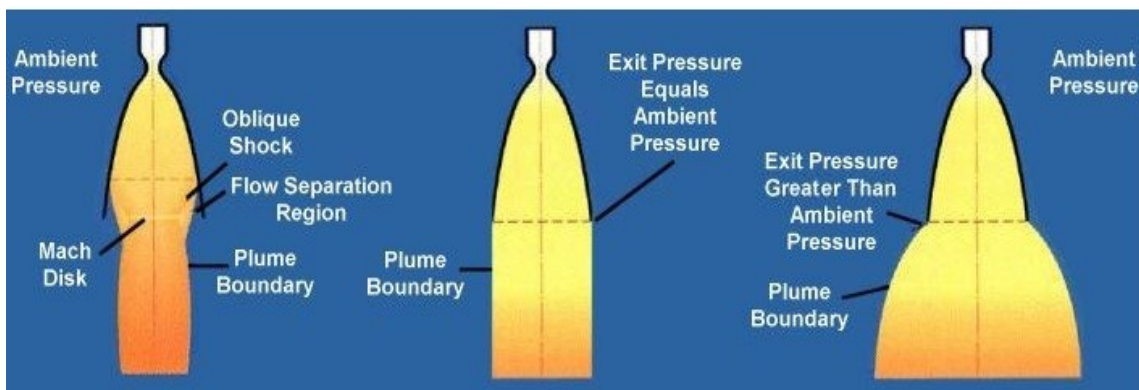


Figure 1-3: Under-Expanded, Over-Expanded, & Perfectly Expanded Nozzles (Reference 1)

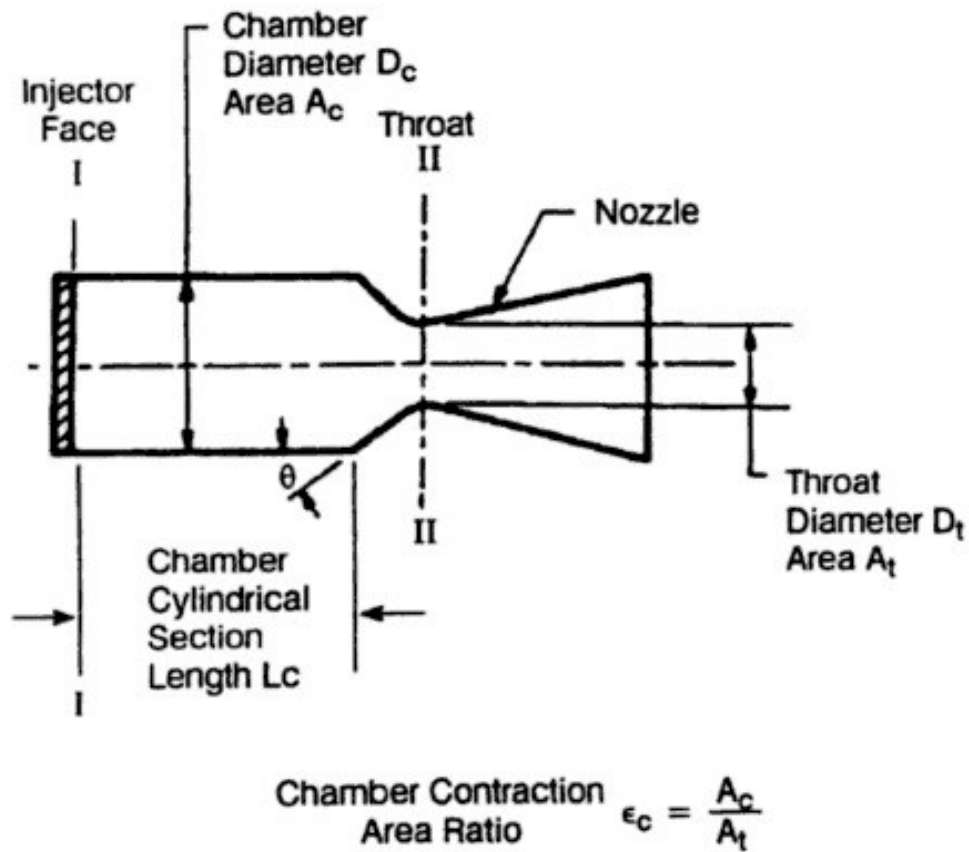


Figure 1-5: Rocket Cylindrical Combustion Chamber (Reference 4)

The injectors, specifically a doublet or triplet impinging stream pattern design are used to introduce liquid propellant into the combustion chamber (Reference 2). The doublet impinging stream pattern works best when the hole size of the fuel is equal to the oxidizer hole size. On the other hand, the triplet impinging stream pattern works best if the hole sizes between the fuel and oxidizers are not that same size (Reference 2). The injector design regulates how the propellant enters and distributes in the chamber. The injector is affected by chamber parameters such as pressure in the chamber, mixture ratio, propellant used, propellant temperature, and most importantly by the design selection of the injector itself. The use of a doublet or triplet injector design has been used most by the U.S. (Reference 1). Figure 1-6 below shows the schematic diagrams of the doublet and triplet injector types.

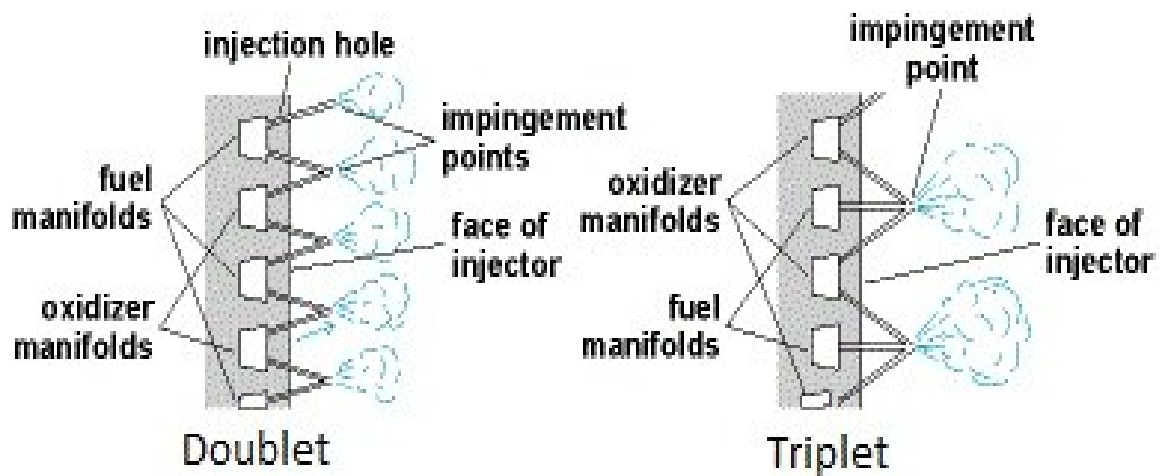


Figure 1-6: Schematic Diagram of the Doublet and Triplet Injector Types (Reference 2)

A literature review of current research and development in rocket nozzle design programs is documented in Chapter 2. A theoretical review of the rocket design methodology used in the program is provided in Chapter 3. Chapter 4 is an overview of the program layout, functionality, and compatibility. The rocket design program runs through various examples in Chapter 5. The conclusions and recommendations for future work on the design program are discussed in Chapter 6.

2.0 Literature Review

This section presents a literature review of current work in rocket design programming and analysis to provide insight into the development of the proposed rocket design, analysis, and simulation of propulsion system.

There are programs available for purchase that allow a user to design a rocket, analyze various parameters, and simulate various attributes of the rocket. However, most programs available don't have all three, design, analysis, and simulation of a rocket propulsion system, and for no cost.

A well-used program called RocketSim is used to design any size rocket and simulate its flight to see how high and fast it will fly. This program is best for design and simulation of model rockets. Design components are such things as the nose cone, body tube, fins, ring tail, tube fins, pod, bulkhead, engine block, and parachute. More specifically, the rocket motor design selections consists of the name of the engine, engine manufacturer from a list of choices, engine code, ejection delay, ignition delay, and overhang dimension. The rocket can be plotted in a graph once all the design parameters are completed. The simulation of the rocket is a flight simulation based on launch conditions input and the starting state with launch guide length and launch angle. (Reference 5)

Another well-used program is AeroSpike. AeroSpike performs 2-D and 3-D minimum length nozzle (MLN) design using the method of characteristics (MOC). In addition to annular and linear aerospike nozzle design, AeroSpike performs an expansion wave analysis from the throat of the thruster to each point on the plug contour to determine the shape of the optimized aerospike nozzle. (Reference 6)

An analysis program called Rocket Propulsion Analysis (RPA) can be purchased to perform calculations of heat transfer rate distributions with or without boundary layer coolant and/or thermal barrier coating layer, film cooling analysis, radiation cooling analysis, regenerative cooling analysis, thermal analysis of thrust chambers with combined cooling, hydraulic losses in the cooling passages, evaluate different propellant compositions, and design nozzle using method of characteristics. The output of all results is saved to plain text or HTML format. (Reference 7)

Programs like the ones listed above have design, analysis, and simulation features, just not all in one program. The rocket propulsion system design, analysis, and simulation to be designed by the author will be designed specifically to feature the ability to design the propulsion system, analyze the various design parameters, and simulate the rocket propulsion system post-process.

3.0 Theoretical Considerations

A theoretical review of the rocket nozzle design of a Rao, conical, minimum length nozzle using method of characteristics, combustion chamber, injectors, and various design calculations used in the program is shown in this chapter.

3.1 Rao Nozzle

The Rao nozzle design consists of three curves: a convergent curve, a divergent curve, and a parabolic curve. The convergent curve end point connects to the divergent curve starting point. Where the divergent curve and the parabolic curve meet is the inflection point. The slope of the parabolic curve is tangent to the inflection angle, θ_n where the divergent curve circle and the parabolic curve intersect. Figure 3-1 shows the parabolic approximation of the Rao nozzle contour used to design and plot the Rao nozzle.

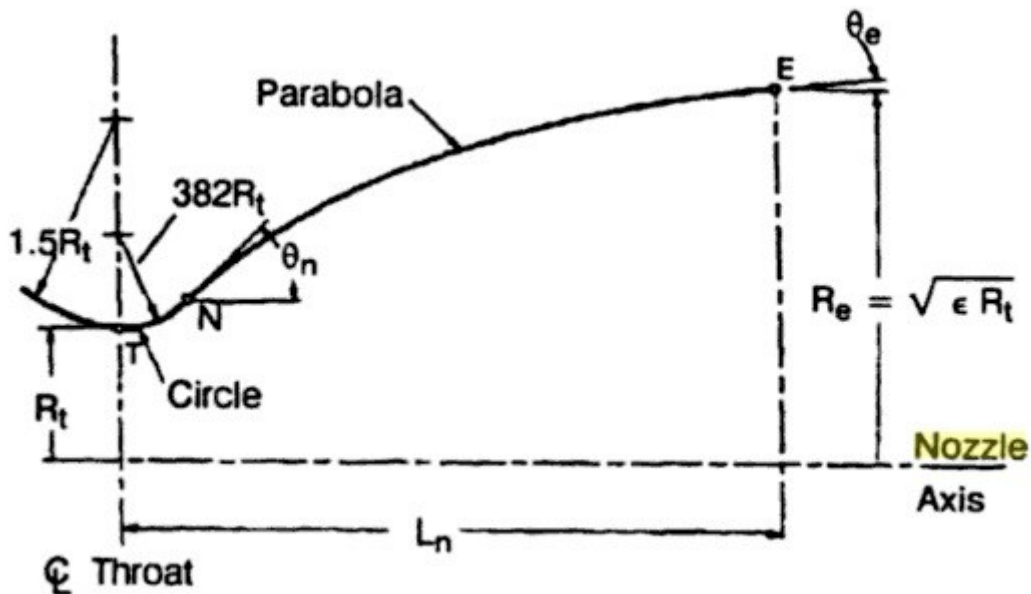


Figure 3-1: Parabolic Approximation of Bell Nozzle Contour (Reference 4)

The equations used to calculate the convergent curve, divergent curve, and parabolic curve are shown below as well as in the Matlab code in Appendix B.

The length of the nozzle, L_n , is calculated in equation 3.1 below, where R_t is the throat radius of the nozzle (Reference 2). The 0.80 is 80% of the conical nozzle, which is what is used for a Rao nozzle design.

$$L_n = \frac{0.80(\sqrt{\varepsilon} - 1)R_t}{\tan(15^\circ)} \quad (3.1)$$

For the convergent curve, which is the first curve, the x and y coordinate points are calculated using equation 3.2 and equation 3.3 below. These equations come from the geometry shown in Figure 3-1 above.

$$x_{first\ curve} = \cos(\theta_{first\ curve}) * 1.5 * R_t \quad (3.2)$$

$$y_{first\ curve} = \sin(\theta_{first\ curve}) * 1.5 * R_t + (1.5 * R_t + R_t) \quad (3.3)$$

For the divergent curve, referred to as the second curve, the x and y coordinate points are calculated using equation 3.4 and equation 3.5. These equations also come from the geometry shown in Figure 3-1 above.

$$x_{second\ curve} = \cos(\theta_{second\ curve}) * 0.382 * R_t \quad (3.4)$$

$$y_{second\ curve} = \sin(\theta_{second\ curve}) * 0.382 * R_t + (0.382 * R_t + R_t) \quad (3.5)$$

For the parabolic curve, which is the third curve, the x and y coordinate points are calculated using equation 3.6 and equation 3.7. The matrix below is used to solve for the coefficients a, b, and c based on non-linear algebraic equations, which are then used to solve the parabolic curve.

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} Y_{second\ curve}^2 & Y_{second\ curve} & 1 \\ Y_{exit}^2 & Y_{exit} & 1 \\ 2Y_{second\ curve} & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} X_{second\ curve} \\ X_{exit} \\ \frac{1}{\tan \theta_n} \end{bmatrix}$$

$$x_{third\ curve} = ay^2 + by + c \quad (3.6)$$

$$y_{third\ curve} = \sqrt{\varepsilon R_t} \quad (3.7)$$

The exit angle, θ_{exit} , is calculated in equation 3.8 below.

$$\theta_{exit} = \tan^{-1} \left(\frac{\Delta Y}{\Delta X} \right) \quad (3.8)$$

3.2 Conical Nozzle

The Conical nozzle design consists of two curves: a convergent curve and straight curve with an inflection angle, θ_n . Figure 3-2 shows the parabolic approximation of the bell nozzle contour used to calculate the convergent curve. The straight curve is a slope based on the length of the nozzle, L_n and the slope of the curve.

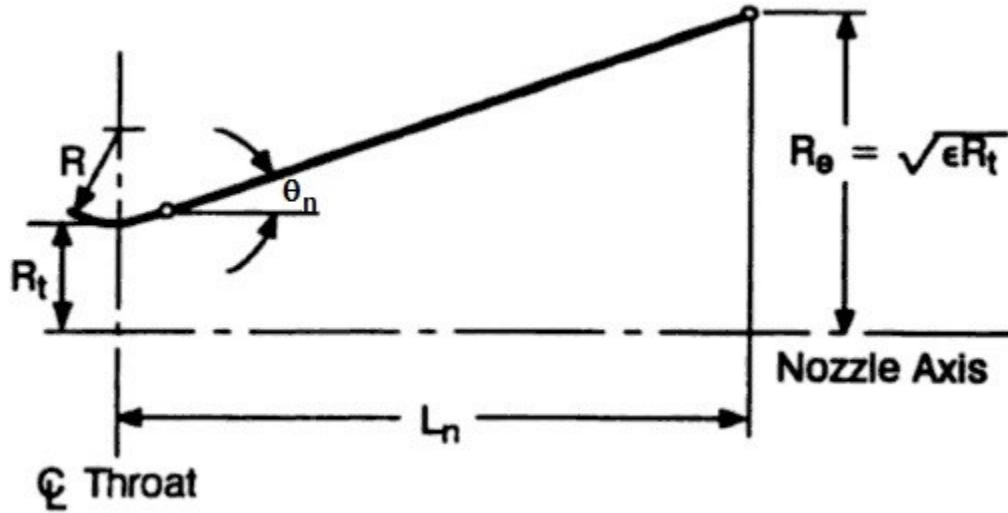


Figure 3-2: Schematic Sketch of Conical Nozzle (Reference 4)

The calculations for the two curves are shown below as well as in Appendix B. For the convergent curve, the first curve, equation 3.8 and equation 3.9 below are used to calculate the x and y coordinates.

$$x_{\text{first curve}} = \cos(\theta_{\text{first curve}}) * 1.5 * R_t \quad (3.8)$$

$$y_{\text{first curve}} = \sin(\theta_{\text{first curve}}) * 1.5 * R_t + (1.5 * R_t + R_t) \quad (3.9)$$

For the straight curve, the second curve, equation 3.10 and equation 3.11 are used to calculate the x and y coordinates. Equation 3.12 and equation 3.13 are used to calculate the coefficients of the y coordinate.

$$x_{\text{second curve}} = L_n \quad (3.10)$$

$$y_{\text{second curve}} = ax_{\text{second curve}} + b \quad (3.11)$$

$$a = \tan(\theta_n) \quad (3.12)$$

$$b = y - ax \quad (3.13)$$

3.3 Minimum Length Nozzle (MLN) using Method of Characteristics (MOC)

3.3.1 Characteristic Lines: Two-Dimensional Irrotational Flow

For steady, two-dimensional, irrotational flow, the determination of the characteristic lines is done using equation 3.14, which is the full velocity potential equation. Note that Φ is the velocity potential. (Reference 3)

$$\left(1 - \frac{\Phi_x^2}{a^2}\right)\Phi_{xx} + \left(1 - \frac{\Phi_y^2}{a^2}\right)\Phi_{yy} - \frac{2\Phi_x\Phi_y}{a^2}\Phi_{xy} = 0 \quad (3.14)$$

Note: Φ is the full-velocity potential

The velocity potential by definition is shown in equation 3.15. Also, recalling that $\Phi_x = f(x, y)$ then the following equations 3.16 and 3.17 are calculated. (Reference 3)

$$\Phi_x = u, \quad \Phi_y = v \quad \text{and} \quad V = u\hat{i} + v\hat{j} \quad (3.15)$$

$$d\Phi_x = \frac{\partial\Phi_x}{\partial x}dx + \frac{\partial\Phi_x}{\partial y}dy = \Phi_{xx}dx + \Phi_{xy}dy \quad (3.16)$$

$$d\Phi_y = \frac{\partial\Phi_y}{\partial x}dx + \frac{\partial\Phi_y}{\partial y}dy = \Phi_{xy}dx + \Phi_{yy}dy \quad (3.17)$$

Substituting the equation 3.15 into equations 3.14, 3.16, and 3.17 the following equations 3.18, 3.19, 3.20 are created. (Reference 3)

$$\left(1 - \frac{u^2}{a^2}\right)\Phi_{xx} - \frac{2uv}{a^2}\Phi_{xy} + \left(1 - \frac{v^2}{a^2}\right)\Phi_{yy} = 0 \quad (3.18)$$

$$(dx)\Phi_{xx} + (dy)\Phi_{xy} = du \quad (3.19)$$

$$(dx)\Phi_{xy} + (dy)\Phi_{yy} = dv \quad (3.20)$$

Equations 3.18 through 3.20 are a system of simultaneous, linear, algebraic equations in the variables Φ_{xx} , Φ_{yy} , and Φ_{xy} . By applying Cramer's rule, the solution for Φ_{xy} is found to be the following equation 3.21. (Reference 3)

$$\Phi_{xy} = \frac{\begin{vmatrix} 1 - \frac{u^2}{a^2} & 0 & 1 - \frac{v^2}{a^2} \\ dx & du & 0 \\ 0 & dv & dy \end{vmatrix}}{\begin{vmatrix} 1 - \frac{u^2}{a^2} & -\frac{2uv}{a^2} & 1 - \frac{v^2}{a^2} \\ dx & dy & 0 \\ 0 & dx & dy \end{vmatrix}} = \frac{N}{D} \quad (3.21)$$

As seen in Figure 3-3 below, a point A and its surrounding neighborhood in an arbitrary flowfield are shown. The derivative of the velocity potential, Φ_{xy} , has a specific value at point A. The solution for Φ_{xy} at point A for an arbitrary choice of dx and dy for an arbitrary direction away from point A defined by the choice of dx and dy. For the chosen dx and dy, there are corresponding values of the change in velocity du and dv. (Reference 3)

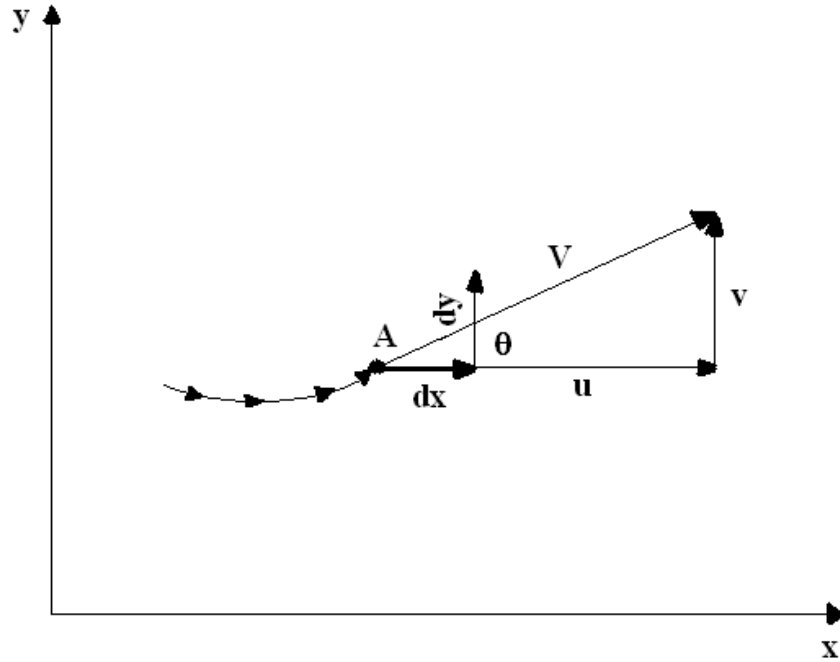


Figure 3-3: Streamline Geometry

The slope of the characteristic lines is shown below in equation 3.22 and equation 3.23.

(Reference 3)

$$\left(\frac{dy}{dx}\right)_{char} = \frac{-uv/a^2 \pm \sqrt{[(u^2 + v^2)/a^2] - 1}}{[1 - (u^2/a^2)]} \quad (3.22)$$

$$\frac{u^2 + v^2}{a^2} - 1 = \frac{V^2}{a^2} - 1 = M^2 - 1 \quad (3.23)$$

There are three important statements: (Reference 3)

1. If $M > 1$, there are two real characteristics through each point of the flowfield. Moreover, for this situation, equation 3.14 is defined as a hyperbolic partial differential equation.
2. If $M = 1$, there is one real characteristic through each point of the flow. Equation 3.14 is a parabolic partial differential equation.

3. If $M < 1$, the characteristics are imaginary, and equation 3.14 is an elliptic partial differential equation.

Two real characteristics exist through each point in a flow where $M > 1$, the method of characteristics (MOC) becomes a practical technique for solving supersonic flow.

The steady, two-dimensional supersonic flow, equation 3.22 is examined. Consider the streamline as shown in Figure 3-3. At point A, $u = V\cos\theta$ and $v = V\sin\theta$. Equation 3.22 becomes equation 3.24 shown below. (Reference 3)

$$\left(\frac{dy}{dx}\right)_{char} = \frac{\frac{-V^2 \cos \theta \sin \theta}{a^2} \pm \sqrt{\frac{V^2}{a^2} (\cos^2 \theta + \sin^2 \theta) - 1}}{\left[1 - \frac{V^2}{a^2} \cos^2 \theta\right]} \quad (3.24)$$

Since, $\mu = \sin^{-1}(1/M)$, equation 3.25 is obtained as follows. (Reference 3)

$$V^2 / a^2 = M^2 = 1 / \sin^2 \mu \quad (3.25)$$

Therefore, the following equation 3.26 is obtained. (Reference 3)

$$\left(\frac{dy}{dx}\right)_{char} = \frac{\frac{-\cos \theta \sin \theta}{\sin^2 \mu} \pm \sqrt{\frac{\cos^2 \theta + \sin^2 \theta}{\sin^2 \mu} - 1}}{\left[1 - \frac{\cos^2 \theta}{\sin^2 \mu}\right]} \quad (3.26)$$

From trigonometry and algebra, the slope of the characteristic lines becomes equation 3.27.

$$\left(\frac{dy}{dx}\right)_{char} = \tan(\theta \mp \mu) \quad (3.27)$$

A graphical interpretation of equation 3.27 is shown in Figure 3-4 below. At point A, the streamline makes an angle θ with the x axis. There are two characteristic passing through point A, one at the angle μ above the streamline, and the other at the angle μ below the streamline.

The characteristic lines are Mach lines. The characteristic given by the angle $\theta + \mu$ is called a C_+ characteristic; it is a left-running characteristic. The characteristic given by the angle $\theta - \mu$ is called a C_- characteristic; it is a right-running characteristic. The characteristics are curved due to flow properties changing from point to point in the flow.

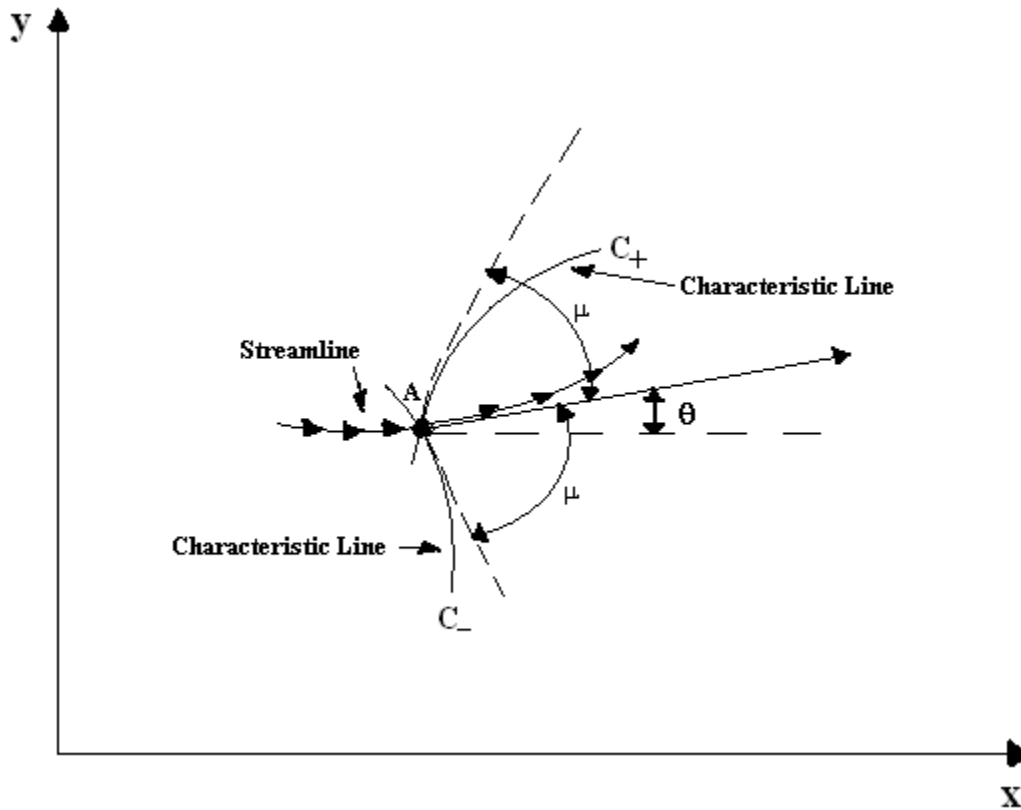


Figure 3-4: Left- and Right-Running Characteristic Lines

3.3.2 Compatibility Equations

The compatibility equation is the following equation 3.28. (Reference 3)

$$\frac{dv}{du} = \frac{-\left[1 - \left(u^2 / a^2\right)\right] dy}{\left[1 - \left(v^2 / a^2\right)\right] dx} \quad (3.28)$$

From equation 3.21, N is zero only when D is zero in order to keep the flowfield derivatives finite. When $D = 0$, only directions along the characteristic lines are considered as

well when $N = 0$. Therefore, equation 3.28 holds true only along the characteristic lines. Therefore, the following equation 3.29 is defined. (Reference 3)

$$\frac{dy}{dx} \equiv \left(\frac{dy}{dx} \right)_{char} \quad (3.29)$$

Putting equation 3.22 into equation 3.29 creates equation 3.30 below. (Reference 3)

$$\frac{dv}{du} = - \frac{\frac{u^2}{a^2} \mp \sqrt{\frac{u^2}{a^2} - 1}}{1 - \frac{v^2}{a^2}} \quad (3.30)$$

Since $u = V \cos \theta$ and $v = V \sin \theta$, equation 3.30 becomes equation 3.31. (Reference 3)

$$\frac{d(V \sin \theta)}{d(V \cos \theta)} = \frac{M^2 \cos \theta \sin \theta \mp}{1 - M^2 \sin^2 \theta} \quad (3.31)$$

The compatibility equation therefore is shown in equation 3.32 below. (Reference 3)

$$d\theta = \mp \frac{dV}{V} \quad (3.32)$$

There negative form of the equation applies along the C_- characteristic and the positive form of the equation applies to the C_+ characteristic. Equation 3.32 is identical to equation 3.33 shown below for Prandtl-Meyer flow. (Reference 3)

$$d\theta = \sqrt{M^2 - 1} \frac{dV}{V} \quad (3.33)$$

Since the equations are identical, equation 3.32 is replaced by the algebraic compatibility equations 3.34 and 3.35. (Reference 3)

$$\theta + \nu(M) = c_{nst} = K_- \quad (\text{along the } C_- \text{ characteristic}) \quad (3.34)$$

$$\theta - \nu(M) = c_{nst} = K_+ \quad (\text{along the } C_+ \text{ characteristic}) \quad (3.35)$$

3.3.3 Compatibility Equations Point by Point Along the Characteristics

3.3.3.1 Internal Flow

If the flowfield conditions are known at two points in the flow, the conditions at a third point can be found. The third point is located by the intersection of the C_- characteristic through the first point and the C_+ characteristic through the second point, as shown in Figure 3-5 below. (Reference 3)

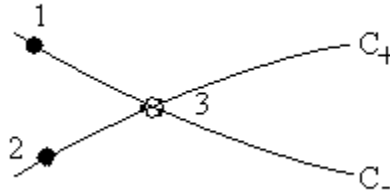


Figure 3-5: Unit Processes for MOC

The θ and v for the third point are found in terms of the known values of K_+ and K_- as shown in equation 3.36 and equation 3.37 below. (Reference 3)

$$\theta = \frac{1}{2} [(K_-)_1 + (K_+)_2] \quad (3.36)$$

$$v = \frac{1}{2} [(K_-)_1 - (K_+)_2] \quad (3.37)$$

Thus, the flow conditions at the third point are now determined from the know values at the first and second point. The v determines the Mach number using equation 3.38. (Reference 3)

$$v(M) = \sqrt{\frac{\gamma+1}{\gamma-1}} \tan^{-1} \sqrt{\frac{\gamma-1}{\gamma+1} (M^2 - 1)} - \tan^{-1} \sqrt{M^2 - 1} \quad (3.38)$$

The pressure, temperature, and density can be calculated after determining the Mach number through isentropic flow relations as shown below in equations 3.39, 3.40, and 3.41.

(Reference 3)

$$\frac{T_o}{T} = 1 + \frac{\gamma - 1}{2} M^2 \quad (3.39)$$

$$\frac{p_o}{p} = \left(1 + \frac{\gamma - 1}{2} M^2 \right)^{\gamma/(\gamma - 1)} \quad (3.40)$$

$$\frac{\rho_o}{\rho} = \left(1 + \frac{\gamma - 1}{2} M^2 \right)^{1/(\gamma - 1)} \quad (3.41)$$

Assuming that characteristics are straight-line segments between the grid points, with slopes that are average values. The C₋ characteristic through the first point is drawn as a straight line with an average slope angle as shown in equation 3.42 below. The C₊ characteristic through the second point is drawn as a straight line with an average slope angle as shown in equation 3.43 below. (Reference 3)

$$\left[\frac{1}{2}(\theta_1 + \theta_3) - \frac{1}{2}(\mu_1 + \mu_3) \right] \quad (3.42)$$

$$\left[\frac{1}{2}(\theta_2 + \theta_3) - \frac{1}{2}(\mu_2 + \mu_3) \right] \quad (3.43)$$

3.3.4 Supersonic Nozzle Design

The expansion of an internal steady flow through a duct from subsonic to supersonic speed, the duct has to be convergent-divergent in shape as seen in Figure 3-6 below. Assume the sonic line to be straight. The flow accelerates to sonic speed in the throat region. Downstream of the sonic line, the duct diverges. In minimum length nozzles the expansion section in Figure 3-7 is shrunk to a point and the expansion takes place through a centered Prandtl-Meyer wave

emanating from a sharp-center throat with an angle $\theta_{wmax, ML}$, as seen in Figure 3-7. The length of the supersonic nozzle, L , is the minimum value consistent with shock free, isentropic flow. Assume that the nozzle in Figure 3-6 and Figure 3-7 have the same exit Mach numbers. For the minimum-length nozzle shown in Figure 3-7, the expansion contour has a sharp corner at point a. The fluid encounters only two systems of waves, the right-running waves from point a and left-running waves from point d. The expansion angle of the wall downstream of the throat is shown below in equation 3.44. (Reference 3)

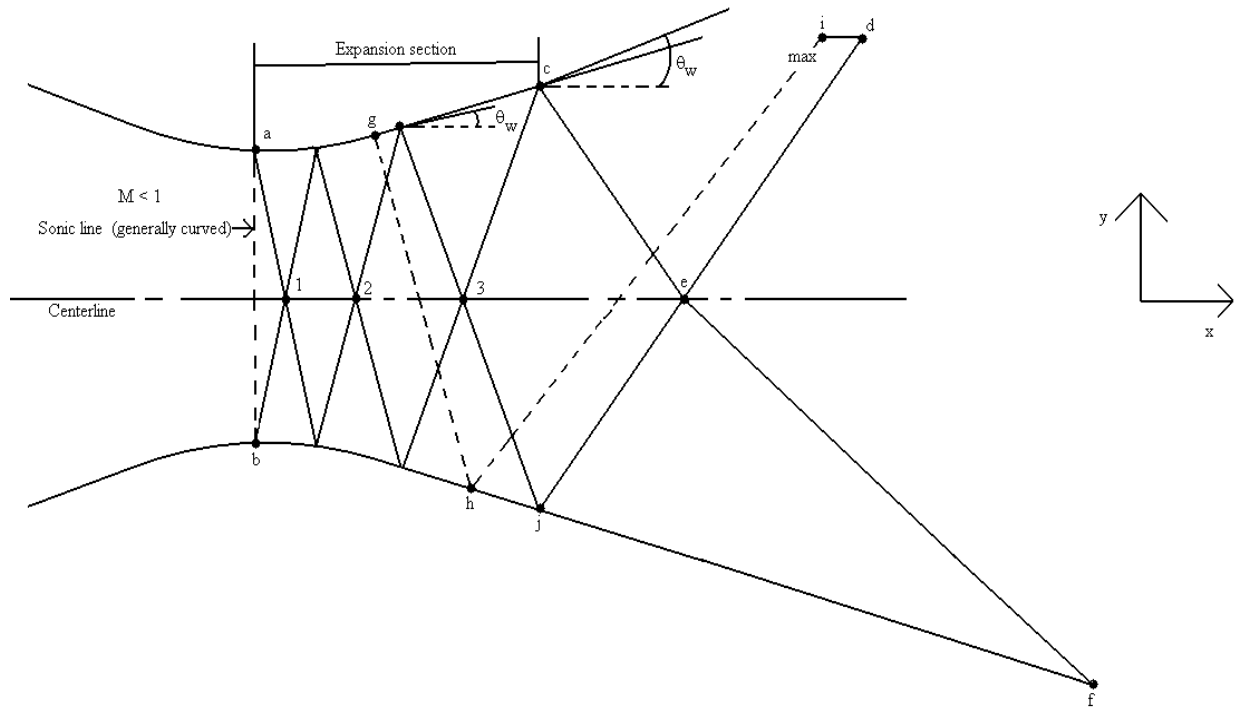


Figure 3-6: Schematic of Supersonic Nozzle Design by the MOC (Reference 3)

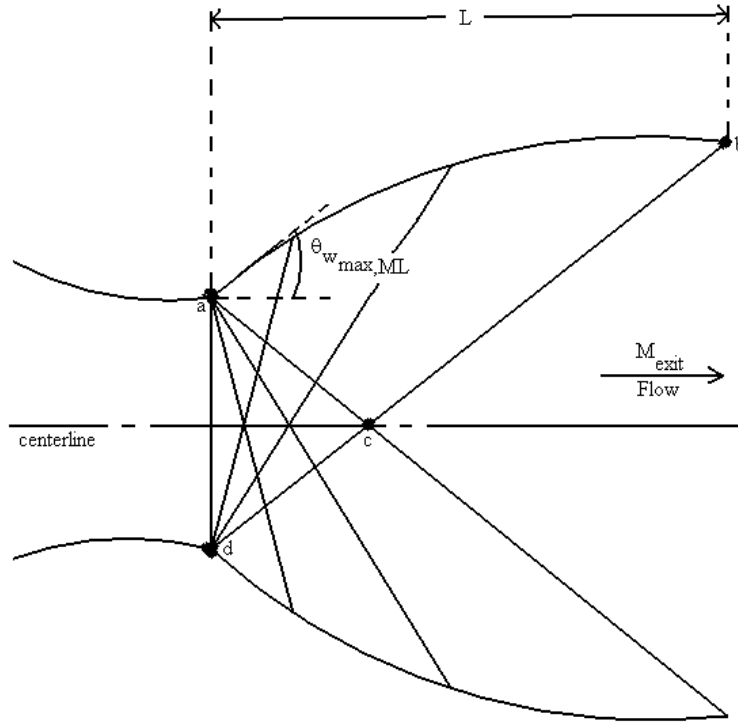


Figure 3-7: Schematic of Minimum-Length Nozzle (Reference 3)

$$\theta_{w_{\max, ML}} = \frac{V_M}{2} \quad (3.44)$$

When the centerline Mach number equals the design exit Mach number, it is point e. The expansion section is terminated at point c, which fixes both its length and the value of $\theta_{w_{\max}}$. The number of nodes is determined using equation 3.45 below.

$$\text{No. of Nodes} = [2 + (n+1)][n/2] \quad (3.45)$$

where n is the number of C_- characteristics

3.4 Combustion Chamber

The combustion chamber is where the burning/combustion of the propellant occurs. A cylindrical chamber is used for the propulsion system design selection.

The chamber volume, V_c , is defined as the volume up to the nozzle throat section and includes the cylindrical chamber and the converging cone frustum of the nozzle as shown in

equation 3.46 below. Note that L_1 is the cylinder length defined in equation 3.47, A_t/A_1 is the chamber contraction ratio, and L_c is the length of the conical frustrum. The length of the conical frustrum is modeled with an angle of 45 degrees and defined as shown in equation 3.48. Also, the characteristic chamber length, L^* , defined in equation 3.49 below, is the length that a chamber of the same volume would have if it were a straight tube and had no converging section. The cylinder volume, V_1 , is the remaining chamber volume that is computed by subtracting the frustrum volume from the chamber total volume, which leaves a cylindrical relationship to determine the cylinder chamber length as shown in equation 3.47 below. (Reference 2)

$$V_c = A_1 L_1 + A_1 L_c \left(1 + \sqrt{A_t / A_1} + A_t / A_1 \right) \quad (3.46)$$

$$L_1 = \frac{V_1}{\pi R_1^2} \quad (3.47)$$

$$L_c = \frac{R_t}{\tan(45^\circ)} \quad (3.48)$$

$$L^* = \frac{V_c}{A_t} \quad (3.49)$$

The following chamber considerations are evaluated straight from Reference 2. The volume has to be large enough for sufficient mixing, evaporation, and complete combustion of propellants. The chamber diameter and volume can influence the cooling requirements. If the chamber volume and the chamber diameter are large, the heat transfer rates to the walls will be reduced, the area exposed to heat will be large, and the walls are somewhat thicker. There is an optimum chamber volume and diameter where the total heat absorbed by the walls will be a minimum. All inert components should have minimum mass. The thrust chamber mass is a function of the chamber dimensions, chamber pressure, and nozzle area ratio, and method of cooling. Manufacturing considerations favor a simple chamber geometry, low cost materials,

and simple fabrication processes. In some applications the length of the chamber and the nozzle relate directly to the overall length of the vehicle. A large-diameter but short chamber can allow a somewhat shorter vehicle with a lower structural inert vehicle mass. The gas pressure drop for accelerating the combustion products within the chamber should be a minimum; any pressure reduction at the nozzle inlet reduces the exhaust velocity and the performance of the vehicle. These losses become appreciable when the chamber area is less than three times the throat area. For the same thrust the combustion volume and the nozzle throat area become smaller as the operating chamber pressure is increased. This means that the chamber length and the nozzle length also decrease with increasing chamber pressure. The performance also goes up with chamber pressure.

Based on the above chamber considerations from Reference 2, finding a solution that would satisfy most of the considerations is typically used for the chamber design.

3.5 **Injectors**

The injectors of the rocket propulsion system are designed as a doublet or triplet configuration. The discharge coefficient, C_d , and the pressure drop percentage are assumptions made. A typical pressure drop percentage is 20% according to Reference 2. The oxidizer and fuel densities, ρ_o and ρ_f , respectively, are determined by dividing the weight densities of the oxidizer and fuel by gravity. The oxidizer and fuel cross sectional areas are determined below in equation 3.50 and equation 3.51, respectively. (Reference 2)

$$A_o = \frac{\rho_o \dot{V}}{2\Delta P C_d} \quad (3.50)$$

$$A_f = \frac{\rho_f \dot{V}}{2\Delta P C_d} \quad (3.51)$$

Knowing the area required for the oxidizer and fuel, the number of holes for each is determined as shown in equation 3.52 and equation 3.53. The diameter of the holes, D , is adjusted in the design calculations based on the desire number of holes in the injector.

(Reference 2)

$$n_o = \frac{A_o}{\frac{\pi}{4} D_o^2} \quad (3.52)$$

$$n_f = \frac{A_f}{\frac{\pi}{4} D_f^2} \quad (3.53)$$

3.6 Propellant Calculations

The exit velocity of the nozzle is determined using equation 3.54 below. (Reference 2)

$$v_2 = \sqrt{\frac{2g_o k}{k-1} \frac{R'T_1}{MM} \left[1 - \left(\frac{p_2}{p_1} \right)^{(k-1)/k} \right]} \quad (3.54)$$

The ideal specific impulse is found using the exit velocity in equation 3.55. Using assumed correction values, the actual specific impulse is found as seen below in equation 3.56. From the actual specific impulse, the weight flow rate is determined using equation 3.57, where F is the engine thrust. (Reference 2)

$$I_{sp_{ideal}} = \frac{v_2}{g_o} \quad (3.55)$$

$$I_{sp_{actual}} = \xi_v I_{sp_{ideal}} \quad (3.56)$$

$$\dot{m} = \frac{F}{I_{sp_{actual}}} \quad (3.57)$$

The weight flow rate is then broken down into oxidizer and fuel flow rates, \dot{w}_o and \dot{w}_f , respectively shown in equations 3.58 and 3.59, where r is the mixture ratio of the propellant.

(Reference 2)

$$\dot{w}_o = \frac{\dot{w}}{1+r} \quad (3.58)$$

$$\dot{w}_f = \frac{\dot{w}r}{1+r} \quad (3.59)$$

With the weight flow rates for the oxidizer and fuel are determined, the volume flow rates are calculated below in equations 3.60 and 3.61. The total propellant requirements for specified time of operations are based on the weight and volume flow rates and the time of operation. Two seconds are assumed for start and stop transients. Equations 3.62, 3.63, 3.64, and 3.65 show the weight and volume of each oxidizer and fuel. (Reference 2)

$$\dot{V}_o = \frac{\dot{w}_o}{\rho_o} \quad (3.60)$$

$$\dot{V}_f = \frac{\dot{w}_f}{\rho_f} \quad (3.61)$$

$$w_o = \dot{w}_o t \quad (3.62)$$

$$w_f = \dot{w}_f t \quad (3.63)$$

$$V_o = \dot{V}_o t \quad (3.64)$$

$$V_f = \dot{V}_f t \quad (3.65)$$

3.7 Resonant Frequency for the 1st Longitudinal, Radial, and Tangential Modes

The rocket combustion instabilities can cause excessive vibration pressure forces. The high-frequency instability occurs in at two modes, longitudinal and transverse. The longitudinal

mode transmits along the axial planes of the combustion chamber and the pressure waves are reflected at the injector face and the converging nozzle cone. The transverse modes consist of the tangential and radial modes, which propagate along the faces perpendicular to the chamber axis. The resonance modes of the combustion chamber are shown below in Figure 3-8 (a) is the longitudinal mode, (b) is the tangential, and (c) is the radial mode. (Reference 2)

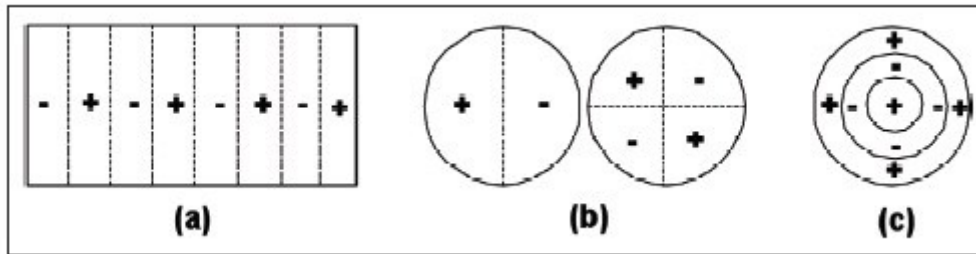


Figure 3-8: The Resonance Modes (Reference 8)

The resonant frequencies in the chamber are calculated based of the relationship between speed of sound, a , and the wavelength, λ . To calculate the resonant frequencies in the chamber, first the speed of sound is determined based on the chamber temperature as shown in equation 3.66. (Reference 2)

$$a = \sqrt{k \frac{R'}{MM} T_1} \quad (3.66)$$

The longitudinal, tangential, and radial frequencies, f , are calculated using the basic relationship between speed of sound, a , and wavelength, λ as shown in equation 3.67 below. (Reference 2)

$$f = \frac{a}{\lambda} \quad (3.67)$$

The wavelengths for each frequency mode, longitudinal, tangential, and radial modes are determined using the chamber characteristic as shown below in equations 3.68, 3.69, and 3.70.

(Reference 2)

$$\textit{Longitudinal}, l = L_1 + L_c \quad (3.68)$$

$$\textit{Tangential}, l = \pi R_1 \quad (3.69)$$

$$\textit{Radial}, l = R_1 \quad (3.70)$$

4.0 Program Details

The program layout of the design, analysis, and simulation of rocket propulsion systems is discussed in this chapter.

4.1 Overall Layout

The overall layout of the program is shown below in Figure 4-1. The features of this program allow the user to design a rocket propulsion system which includes the nozzle, chamber, and injector design. The analysis of the rocket propulsion system features two-dimensional plot of the design choices selected, coordinates of the rocket propulsion system to output to a *.dat file, and various design parameters pertinent to the user's design study. From the design parameters selected and calculated the user can simulate 3-D modeling of the rocket propulsion system design using a CAD program and conducting computational fluid dynamic (CFD) simulations.

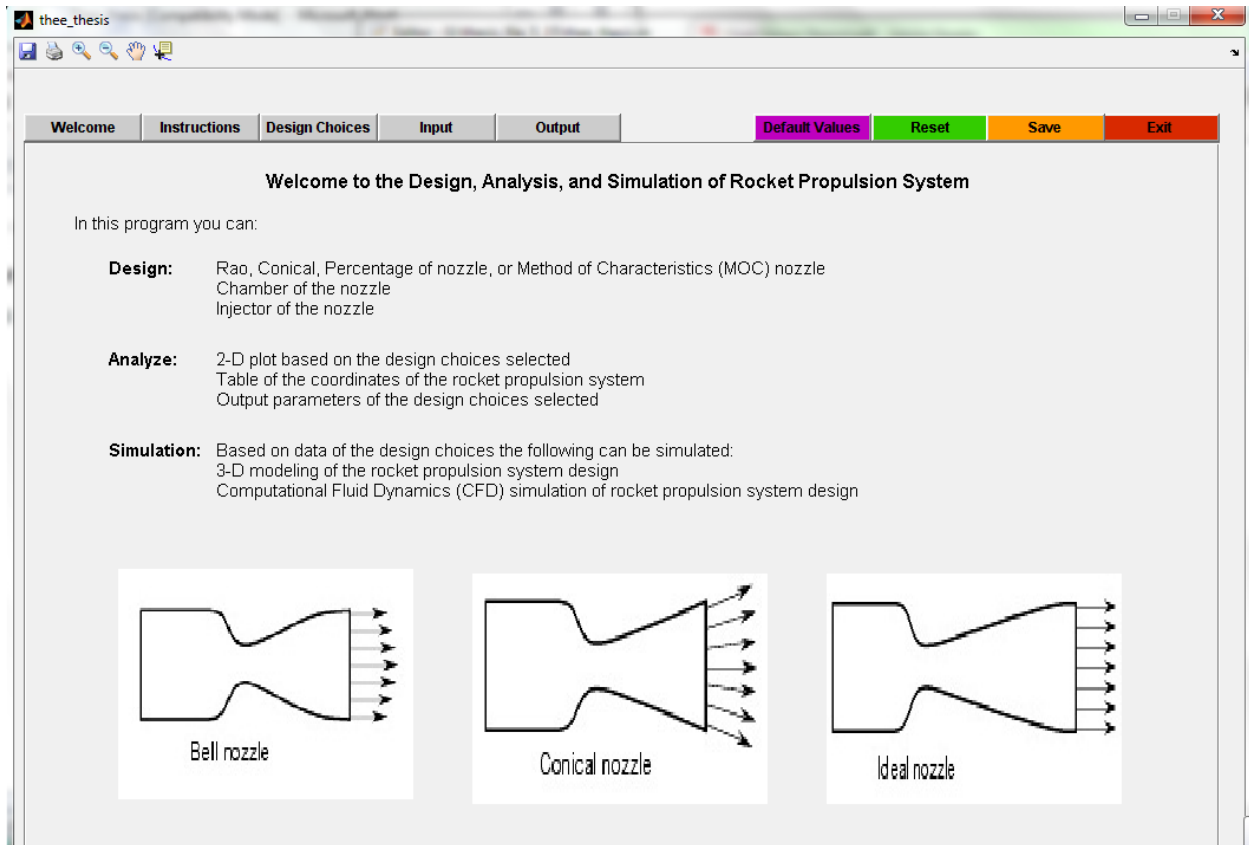


Figure 4-1: Layout Program

The program has a welcome tab, instructions tab, design choices tab, input tab, and output tab as seen in Figure 4-1 and each are discussed in the following sub sections. On the right hand side of the program no matter what tab is currently open, the default values, reset, save, and exit buttons are available for the user to select. The default values button sets default input parameters for a rocket propulsion system design. The reset button clears all input values selected by the user. The save button saves the GUI file of what the user had selected, values inputted, and/or output results. The exit button when selected will prompt the user if they want to exit the program by selecting yes or no in the dialog box. If no is selected, the program stays open and if yes is selected, the program closes without saving. Also, in the top left corner there is a toolbar of options, a save icon, printer icon, zoom in icon; zoom out icon, hand icon, and

data point selector icon. The save icon and printer icon are self-explanatory. The zoom in, zoom out, hands, and data point selector icons are used when in the output tab for the 2-D plots.

4.2 Welcome Tab

The welcome tab is the first to be displayed when opening the design, analysis, and simulation of rocket propulsion system. The welcome tab is shown below in Figure 4-2. The description about the features of the program is shown to the user. The images shown below are the simple shapes of a bell nozzle, conical nozzle, and ideal nozzle, from Reference 9.

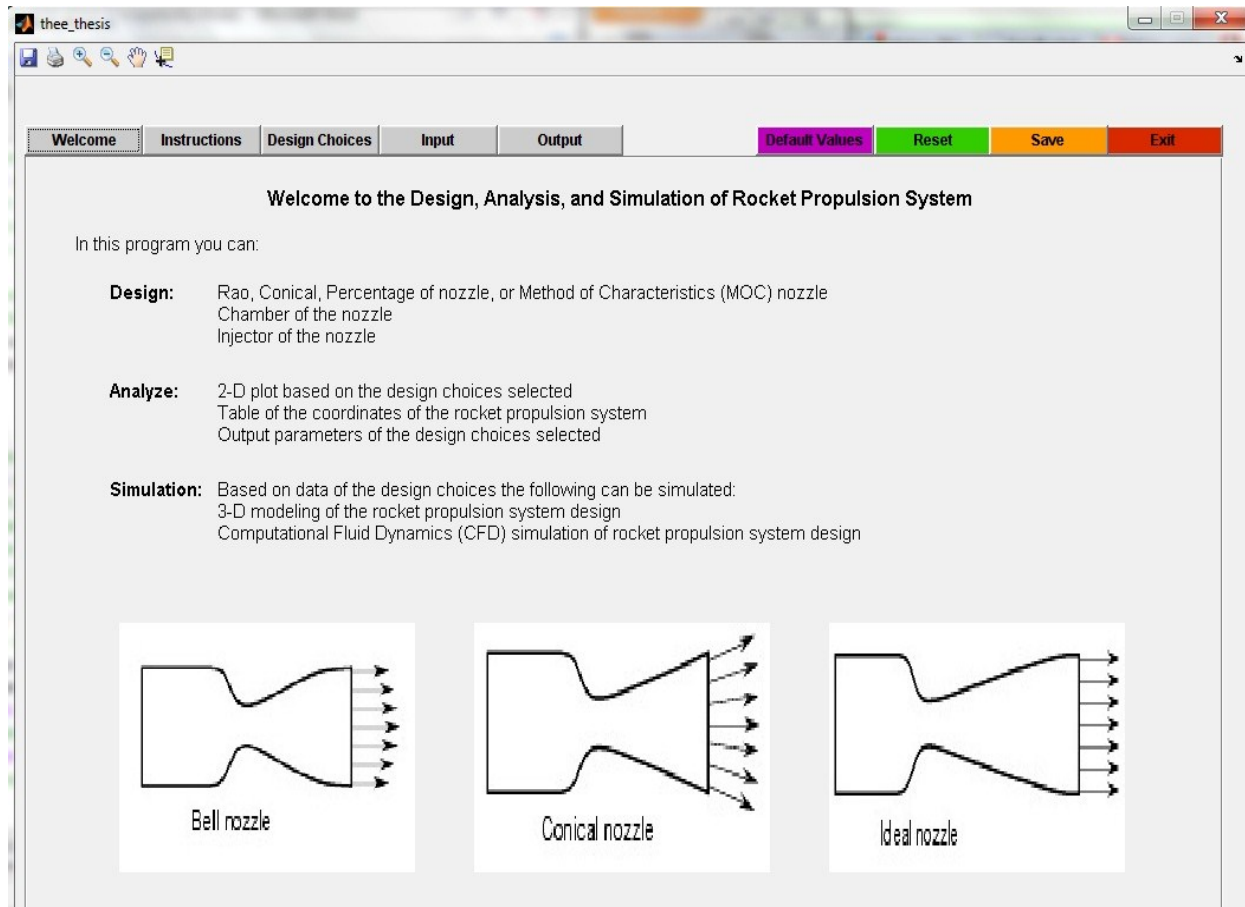


Figure 4-2: Welcome Tab

4.3 Instructions Tab

The instructions tab is the second tab option in the program and is shown below in Figure 4-3. The instructions tab purpose is to familiarize the user on how to use the program and what each tab is used for. The image of a space shuttle main engine is from Reference 10.

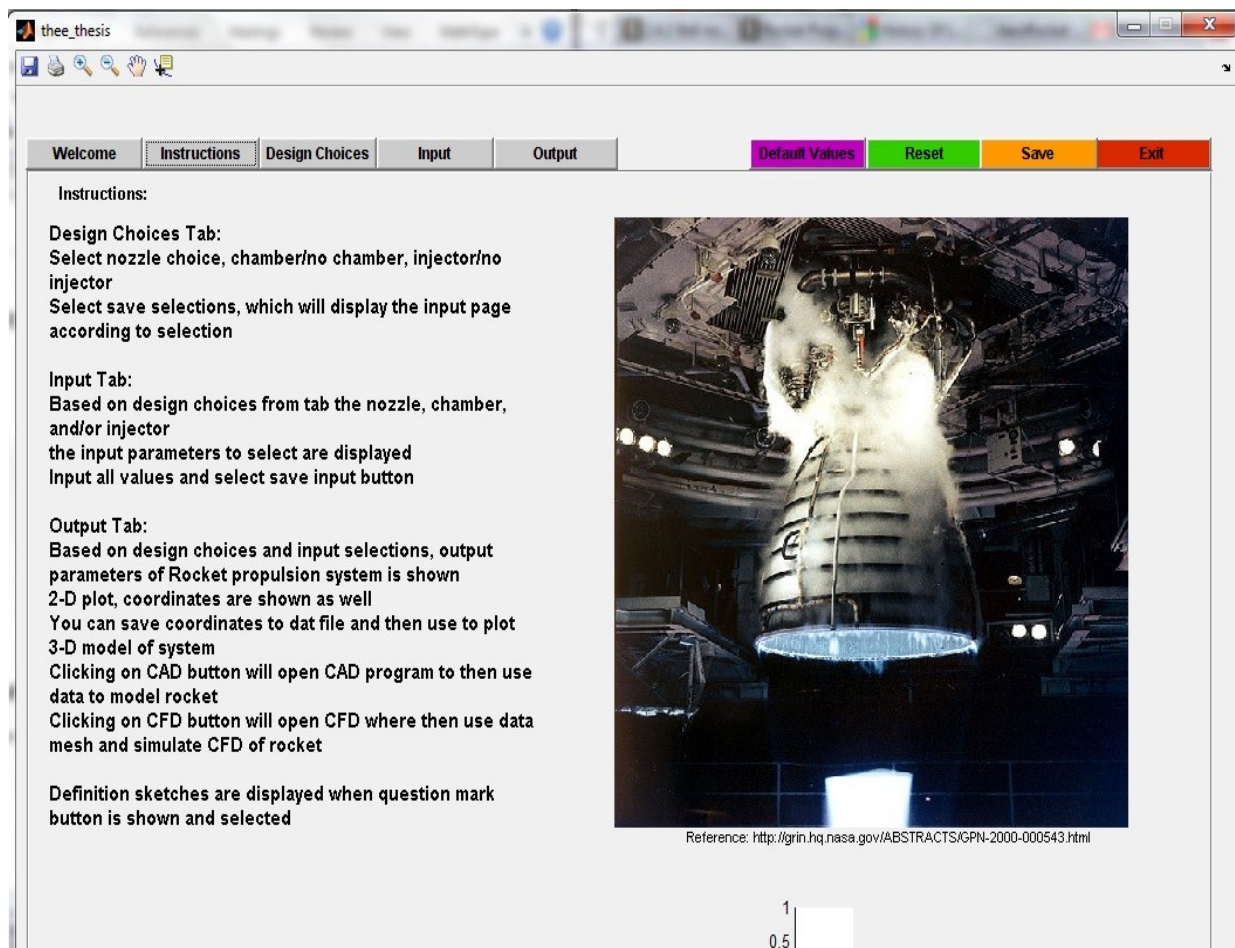


Figure 4-3: Instructions Tab

4.4 Design Choices Tab

The design choices tab is where the user makes the selections for what units to use, the nozzle selection, the chamber selection, and the injector selection. The design choices tab is shown below in Figure 4-4. The images on the right hand side of the tab are of the different flow behaviors of a nozzle, a cylindrical chamber definition sketch, and a doublet and triplet injector type, from Reference 4.

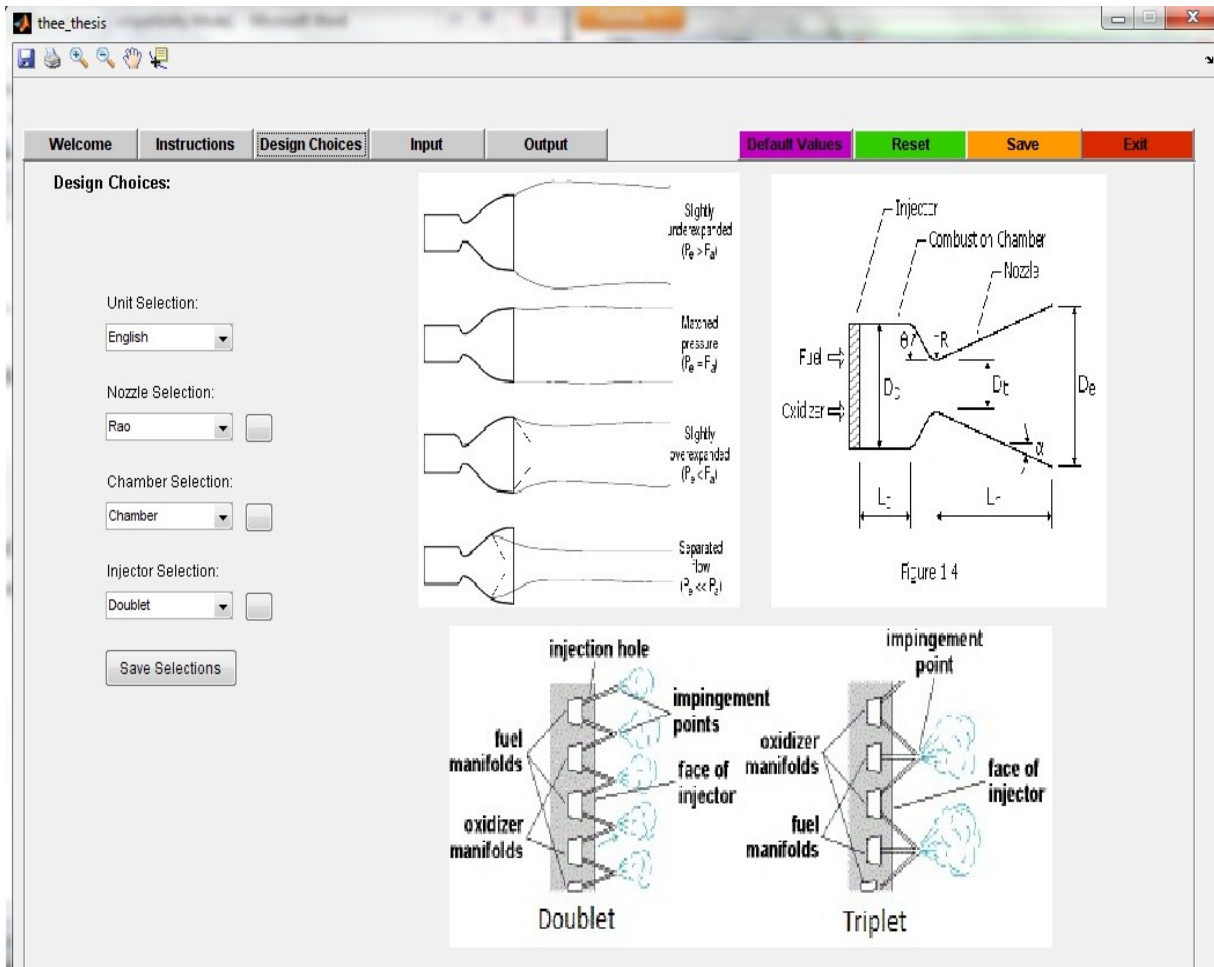


Figure 4-4: Design Choices Tab

The design choices layout of the choices available is shown below in Figure 4-5. Based on the design choices made the input tab discussed in the next sub section displays only that which is selected in the design choices tab to input values for. Once the user makes their design

choices they must click the save selections button and then proceed to the input tab, this is made aware of when the user hovers their mouse over the button, a message will appear to inform them of this.

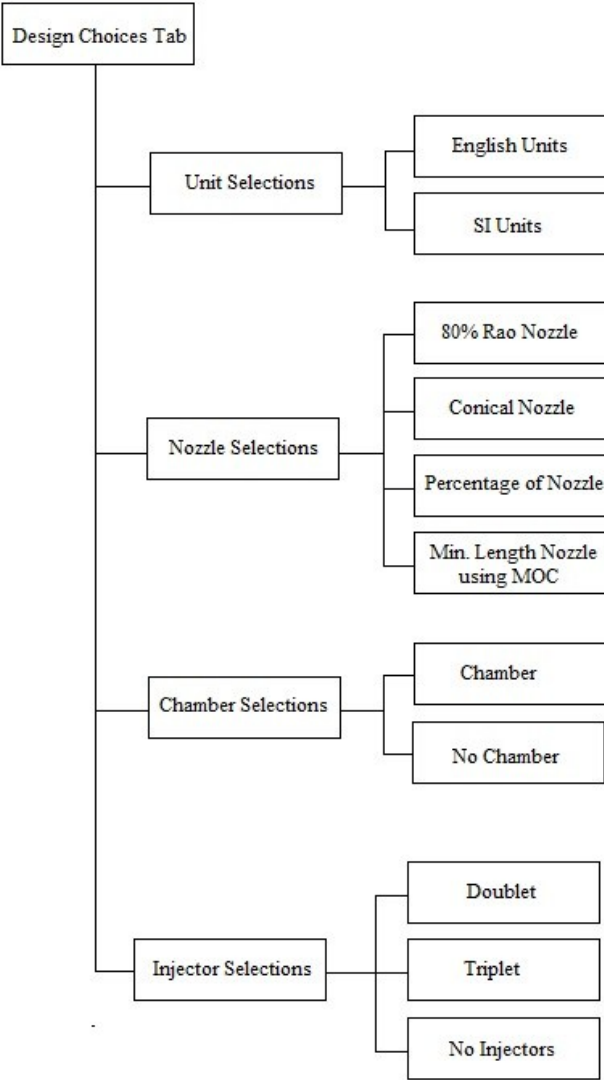


Figure 4-5: Layout of the Design choice selections

4.5 Input selections Tab

The input selections tab displays the panels based on what the user selected on the design choices tab. The input selections tab is shown below in Figure 4-6, note that all possible options were selected to show the user everything. If the user hovers over each of the input parameter values, a message appears with recommended ranges. The definition sketch on the right hand side of the tab is to help the user understand the design parameters and is from Reference 11. The following subsections discuss each input parameter panel within the input selection tab. After all required values are inputted the user must select the OK button on the right hand side to save the results in the GUI and display the results in the output tab.

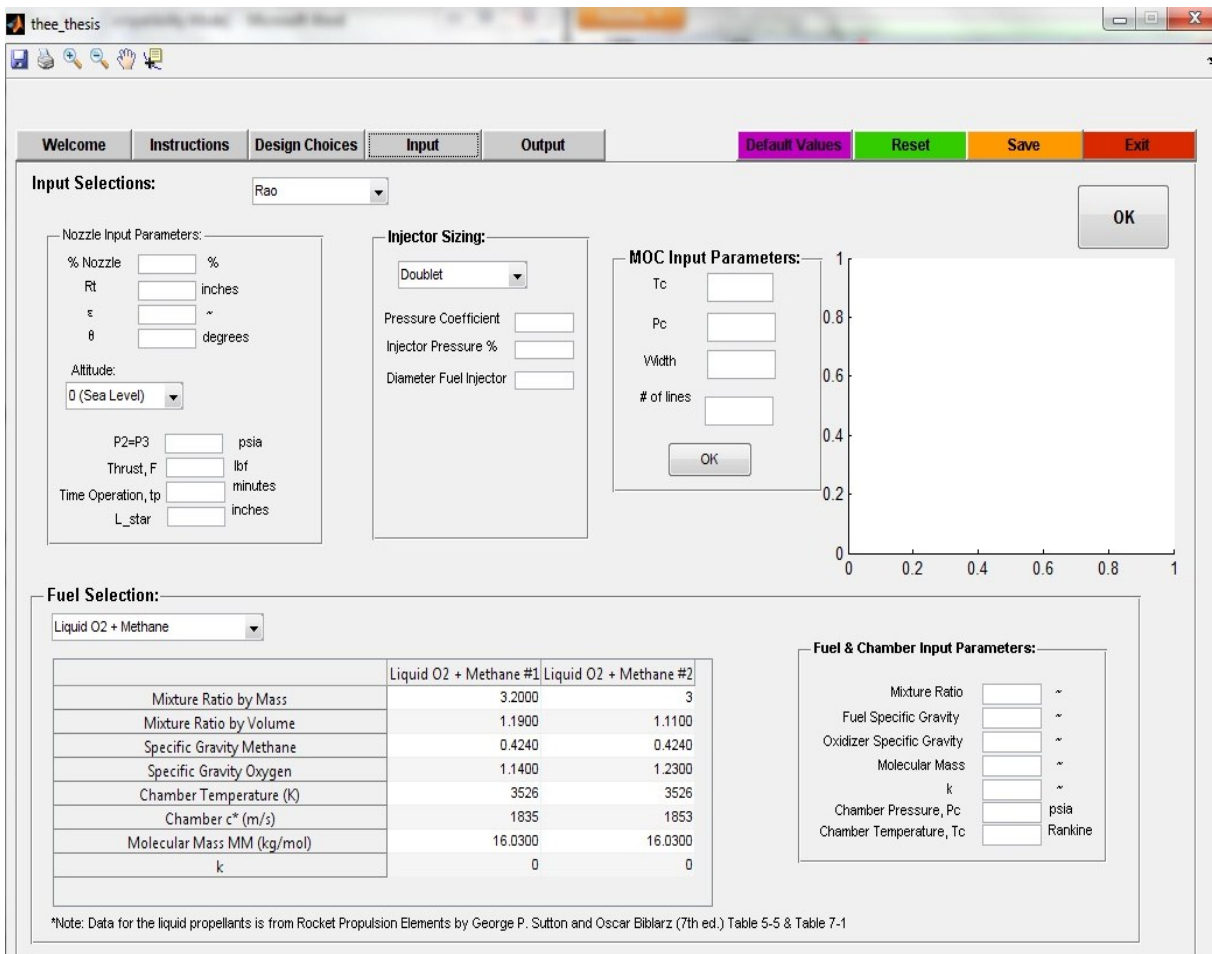


Figure 4-6: Input Tab

4.5.1 Nozzle Input Parameters

The nozzle input parameters are based on design calculations from Reference 2 and are shown in a close up view in Figure 4-7 below. Whether 80% Rao Nozzle, Conical Nozzle, or percentage of nozzle, the design inputs are the same. The percentage of the nozzle, the throat radius, R_t , the area ratio, ϵ , the inflection angle of the nozzle, θ_n , the altitude selection from popup menu, the atmospheric pressure, $p_2 = p_3$, the thrust, F , and the time operation, t_p are input values that the user chooses.

Nozzle Input Parameters:

% Nozzle %

Rt inches

ϵ ~

θ degrees

Altitude:

0 (Sea Level) ▼

P2=P3 psia

Thrust, F lbf

Time Operation, t_p minutes

L_star inches

Figure 4-7: Nozzle Input Parameters

4.5.2 Chamber Input Parameters

The chamber input parameters are based on what liquid propellant the user chooses for the chamber design and the design calculations from Reference 2 as seen in a close up view in Figure 4-8 below. The mixture ratio, r , fuel specific gravity, ρ_f , oxidizer specific gravity, ρ_o ,

molecular mass, MM , specific heat ratio, k , chamber pressure, p_c , chamber temperature, T_c , and the characteristic chamber length, L^* are input values that the user chooses.

Fuel Selection:

Liquid O2 + Methane

	Liquid O2 + Methane #1	Liquid O2 + Methane #2
Mixture Ratio by Mass	3.2000	3
Mixture Ratio by Volume	1.1900	1.1100
Specific Gravity Methane	0.4240	0.4240
Specific Gravity Oxygen	1.1400	1.2300
Chamber Temperature (K)	3526	3526
Chamber c^* (m/s)	1835	1853
Molecular Mass MM (kg/mol)	16.0300	16.0300
k	0	0

Fuel & Chamber Input Parameters:

Mixture Ratio ~

Fuel Specific Gravity ~

Oxidizer Specific Gravity ~

Molecular Mass ~

k ~

Chamber Pressure, P_c psia

Chamber Temperature, T_c Rankine

*Note: Data for the liquid propellants is from Rocket Propulsion Elements by George P. Sutton and Oscar Biblarz (7th ed.) Table 5-5 & Table 7-1

Figure 4-8: Chamber Input Parameters

4.5.3 Injector Input Parameters

The injector input parameters are based on the design calculations from Reference 2 and are shown in a close up view in Figure 4-9 below. The pressure coefficient, C_d , pressure drop in the chamber, ΔP , and the diameter of the fuel injector, d_f are input values that the user chooses.

Injector Sizing:

Doublet

Pressure Coefficient

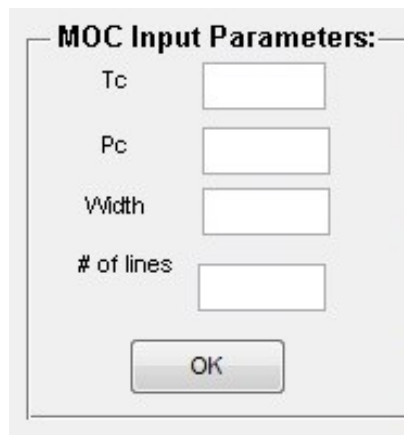
Injector Pressure %

Diameter Fuel Injector

Figure 4-9: Injector Input Parameters

4.5.4 Minimum Length Nozzle (MLN) using Method of Characteristics (MOC) Input Parameters

The minimum length nozzle (MLN) using method of characteristics (MOC) input parameters are based on the design calculations from Reference 3 and are shown in a close up view in Figure 4-10 below. The chamber temperature, T_c , the chamber pressure, P_c , the width of the nozzle, w , and the number of characteristic lines are input values that the user chooses.



The image shows a dialog box titled "MOC Input Parameters:". It contains four input fields, each with a label to its left: "Tc", "Pc", "Width", and "# of lines". Each label is followed by a rectangular text input box. At the bottom center of the dialog box is a button labeled "OK".

Figure 4-10: MLN Input Parameters

4.6 Output selections Tab

The output selections tab displays the results based on what the user selected on the design choices tab and what values were chosen for input to calculate the results. The calculations are based on equations from Reference 2 and Reference 3. The output tab is shown below in Figure 4-11. If the user hovers over each of the output parameter values, a message appears with the equations used to calculate the results. The results pertaining to the nozzle, combustion chamber, and injector design are of the geometry, propellant, and instability frequencies. The axes on the right hand side of the tab displays the plot of the nozzle and/or chamber. The x-y coordinates table displays the coordinates of that which is displayed in the plot. The save to dat

file button exports the x-y coordinates from the table and saves them to a dat file in the same location of the GUI program. The open CAD button when selected opens Unigraphics NX program if available on the user's computer. The *dat file saved is used to import the geometry to the CAD program used for 3-D modeling. The open CFD button when selected opens the CFD ANSYS ICEM meshing program and CFD ANSYS Fluent program if available on the user's computer. The 3-D model created in CAD from the *dat file is exported as a Parasolid or STEP/IGES file which then can be imported in CFD ANSYS ICEM meshing program to mesh the rocket propulsion system design. After creating a mesh on the design, the user then saves and opens the mesh file in Fluent to run CFD simulations. The user has an infinite number of options to choice from.

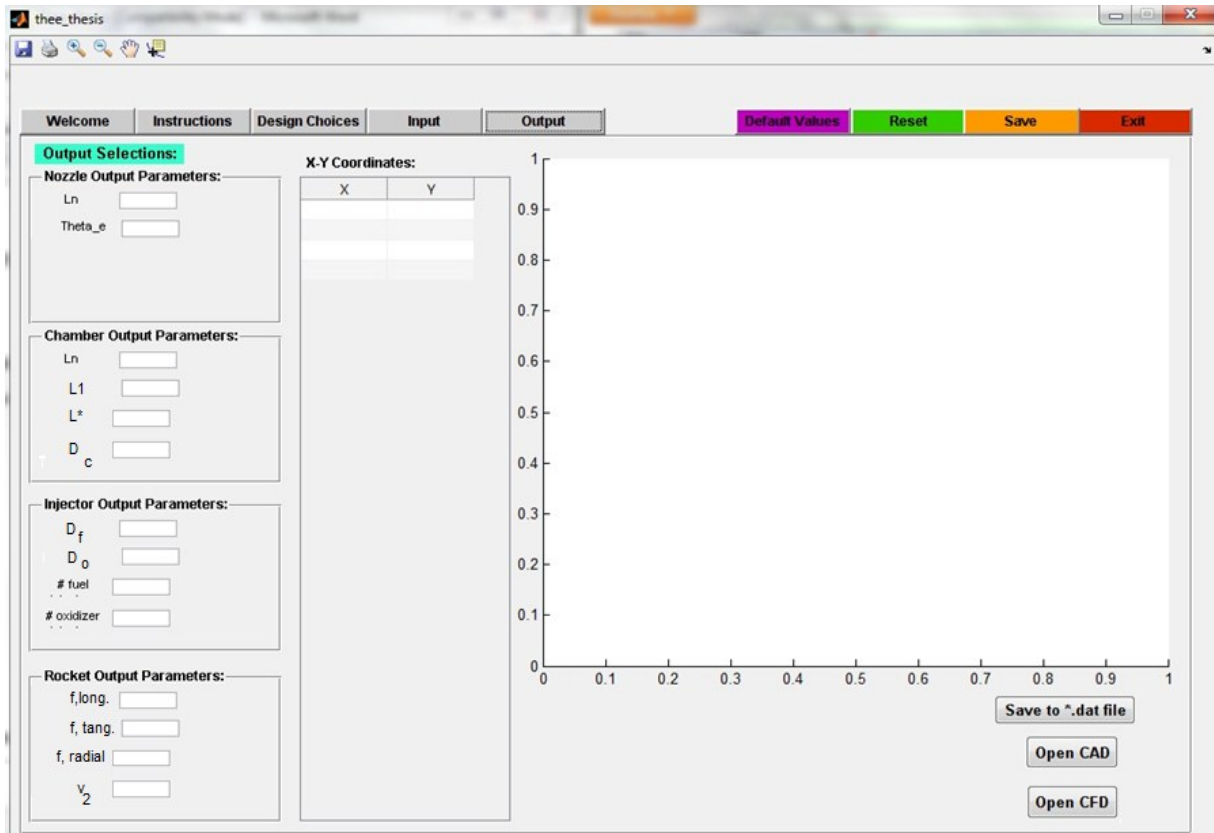


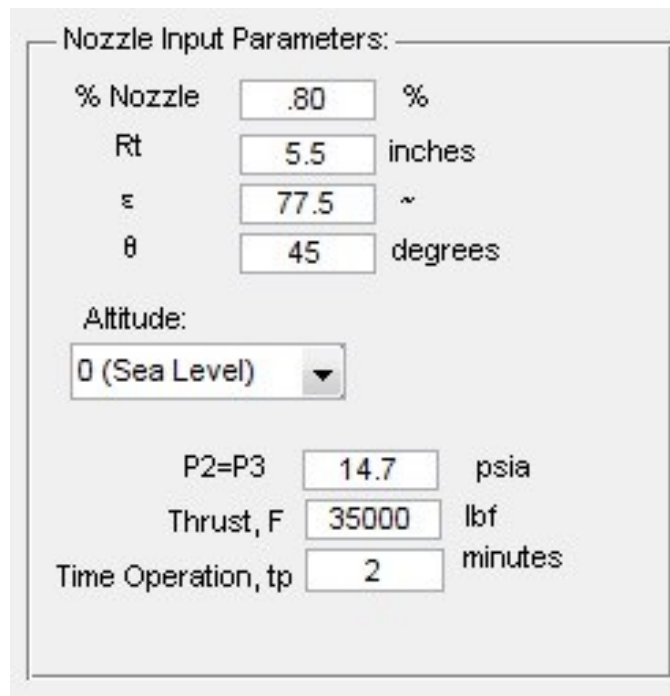
Figure 4-11: Output Tab

5.0 Program Example Run

This chapter shows examples of the program design, analysis, and possible simulations of various rocket propulsion system design choices.

5.1 Nozzle Selections

The nozzle selections of an 80% Rao nozzle, conical nozzle, and 50% Rao nozzle selections with the same throat radius, area ratio, inflection angle for the Rao nozzles, altitude, pressure, thrust, and time of operation as shown below in Figure 5-1. The 2-D plots of the three different nozzles are shown below in Figure 5-2, Figure 5-3, and Figure 5-4. As one can see the nozzle lengths are different for each nozzle type.



The image shows a software interface window titled "Nozzle Input Parameters:". It contains several input fields and a dropdown menu. The parameters and their values are as follows:

Parameter	Value	Unit
% Nozzle	.80	%
Rt	5.5	inches
ϵ	77.5	~
θ	45	degrees
Altitude	0 (Sea Level)	
P2=P3	14.7	psia
Thrust, F	35000	lbf
Time Operation, tp	2	minutes

Figure 5-1: Input Parameters

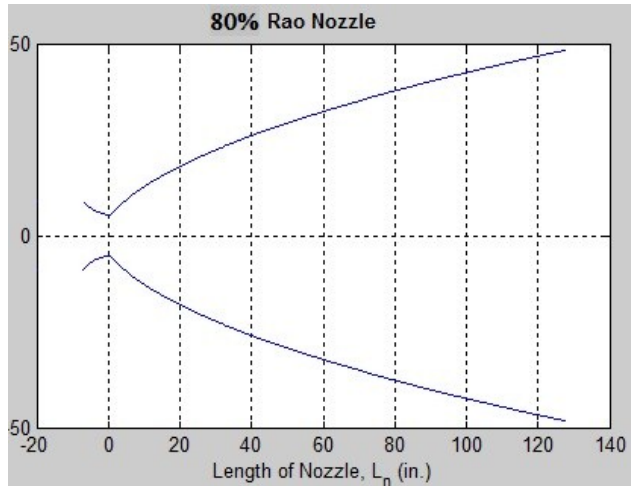


Figure 5-2: 80% Rao Nozzle

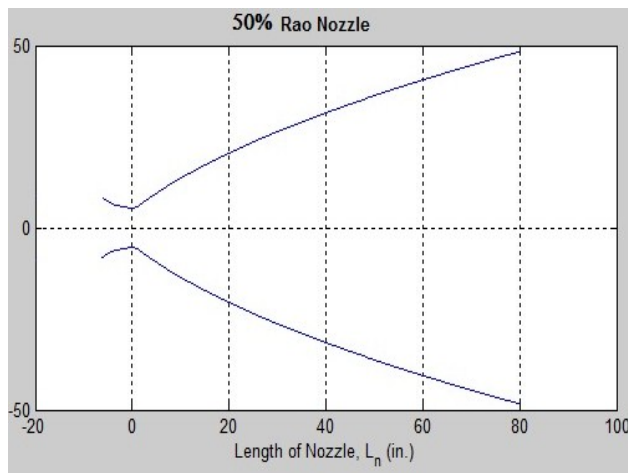


Figure 5-3: 50% Rao Nozzle

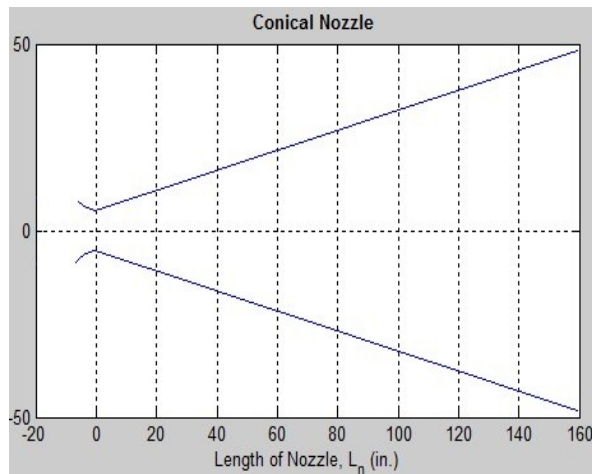


Figure 5-4: 15 Degree Half Angle Conical Nozzle

5.2 Combustion Chamber and Nozzle Selection

The combustion chamber and nozzle selections of an 80% Rao nozzle, conical nozzle, and 50% Rao nozzle selections with the same throat radius, area ratio, inflection angle for the Rao nozzles, altitude, pressure, thrust, time of operation, mixture ratio, fuel specific gravity, oxidizer specific gravity, molecular mass, specific heat ratio, chamber pressure, and chamber temperature for liquid oxygen and RP-1 as shown below in Figure 5-5. The 2-D plots of the three different nozzles are shown below in Figure 5-6, Figure 5-7, and Figure 5-8. As one can see the nozzle lengths are different for each nozzle type, but the chamber dimensions are the same.

Nozzle Input Parameters:

% Nozzle %

Rt inches

ϵ ~

θ degrees

Altitude:
 ▾

P2=P3 psia

Thrust, F lbf

Time Operation, tp minutes

L_star inches

Propellant Selection:

▾

	Liquid O2 + Methane #1	Liquid O2 + Methane #2
Mixture Ratio by Mass	3.2000	3
Mixture Ratio by Volume	1.1900	1.1100
Specific Gravity Methane	0.4240	0.4240
Specific Gravity Oxygen	1.1400	1.2300
Chamber Temperature (K)	3526	3526
Chamber c^* (m/s)	1835	1853
Molecular Mass MM (kg/mol)	16.0300	16.0300
k	0	0

*Note: Data for the liquid propellants is from Rocket Propulsion Elements by George P. Sutton and Oscar Biblarz (7th ed.) Table 5-5 & Table 7-1

Fuel & Chamber Input Parameters:

Mixture Ratio ~

Fuel Specific Gravity ~

Oxidizer Specific Gravity ~

Molecular Mass ~

k ~

Chamber Pressure, Pc psia

Chamber Temperature, Tc Rankine

Figure 5-5: Chamber and Nozzle Input Parameters

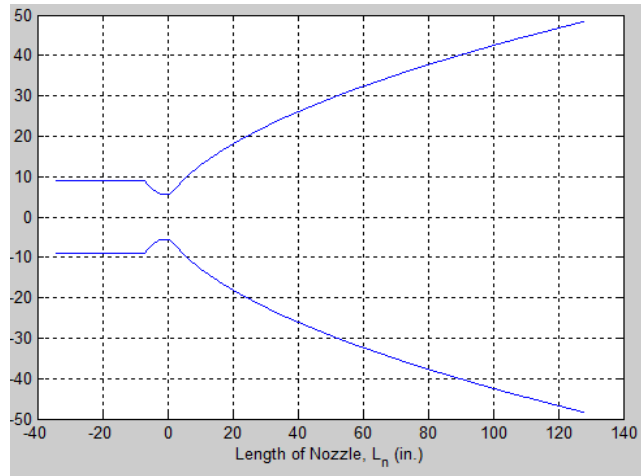


Figure 5-6: 80% Rao Nozzle with Chamber

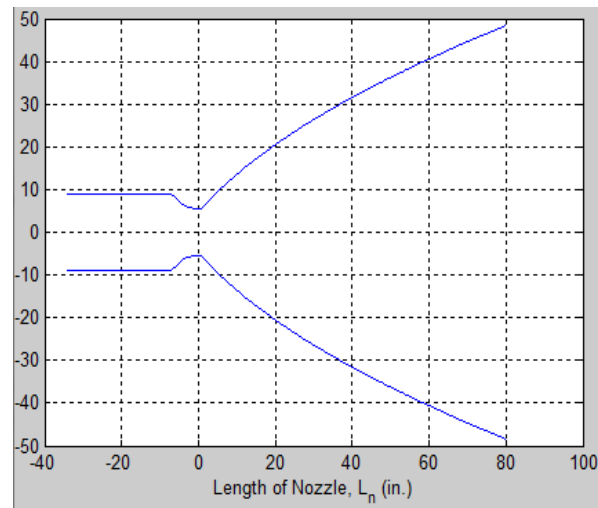


Figure 5-7: 50% Rao Nozzle with Chamber

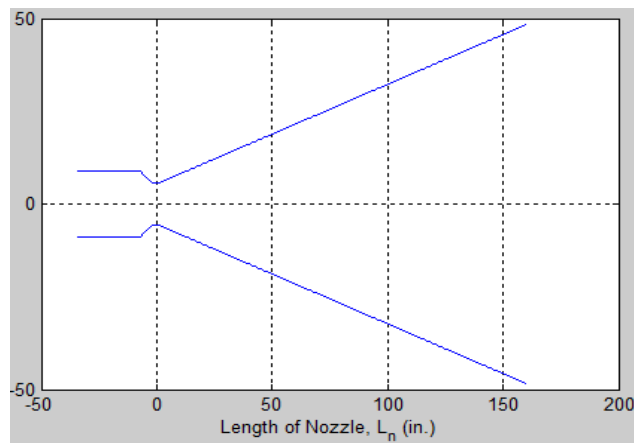
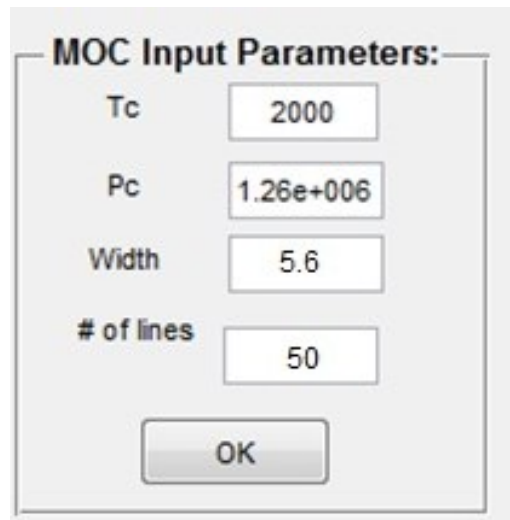


Figure 5-8: 15 Degree Half Angle Conical Nozzle with Chamber

5.3 Minimum Length Nozzle (MLN) using Method of Characteristics (MOC)

The MLN using MOC is selected with the following input parameters shown in Figure 5-9 below. The 2-D plot of the MLN for 25 lines, 50 lines, and 100 lines is shown in Figure 5-10, Figure 5-11, and Figure 5-12, respectively.



The image shows a dialog box titled "MOC Input Parameters:". It contains four input fields: "Tc" with the value "2000", "Pc" with the value "1.26e+006", "Width" with the value "5.6", and "# of lines" with the value "50". There is an "OK" button at the bottom.

Figure 5-9: MOC Input Parameters

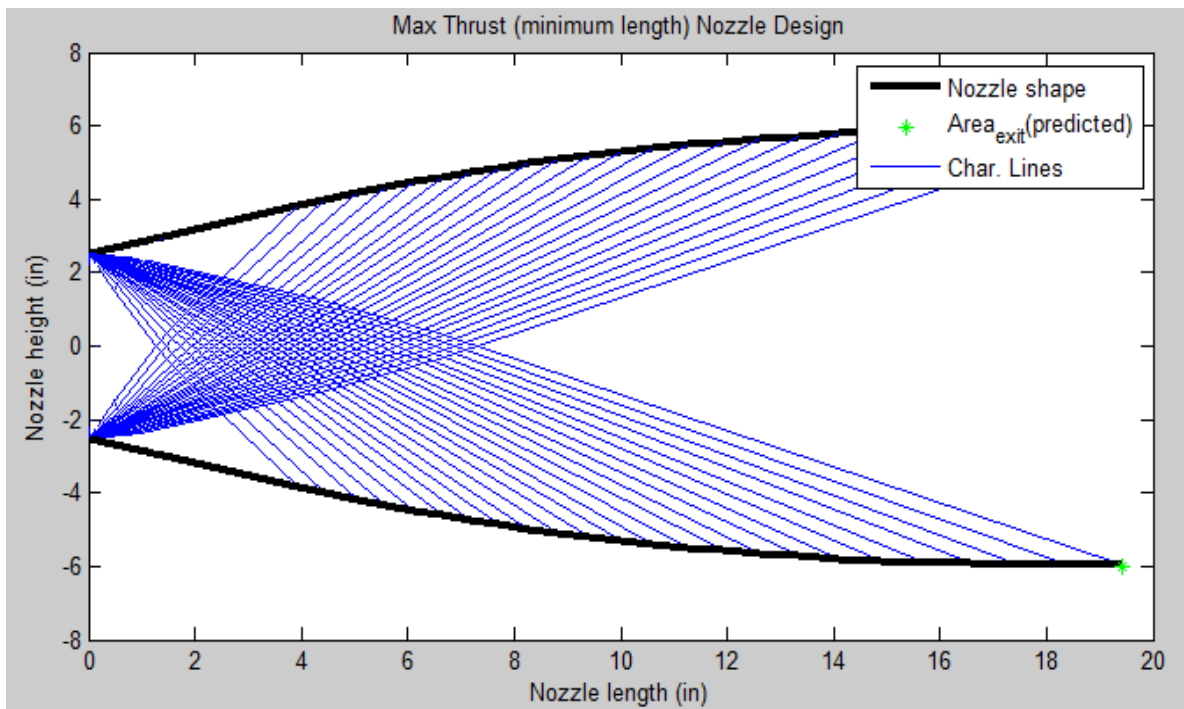


Figure 5-10: Two-Dimensional Plot of the MLN using MOC for 25 lines

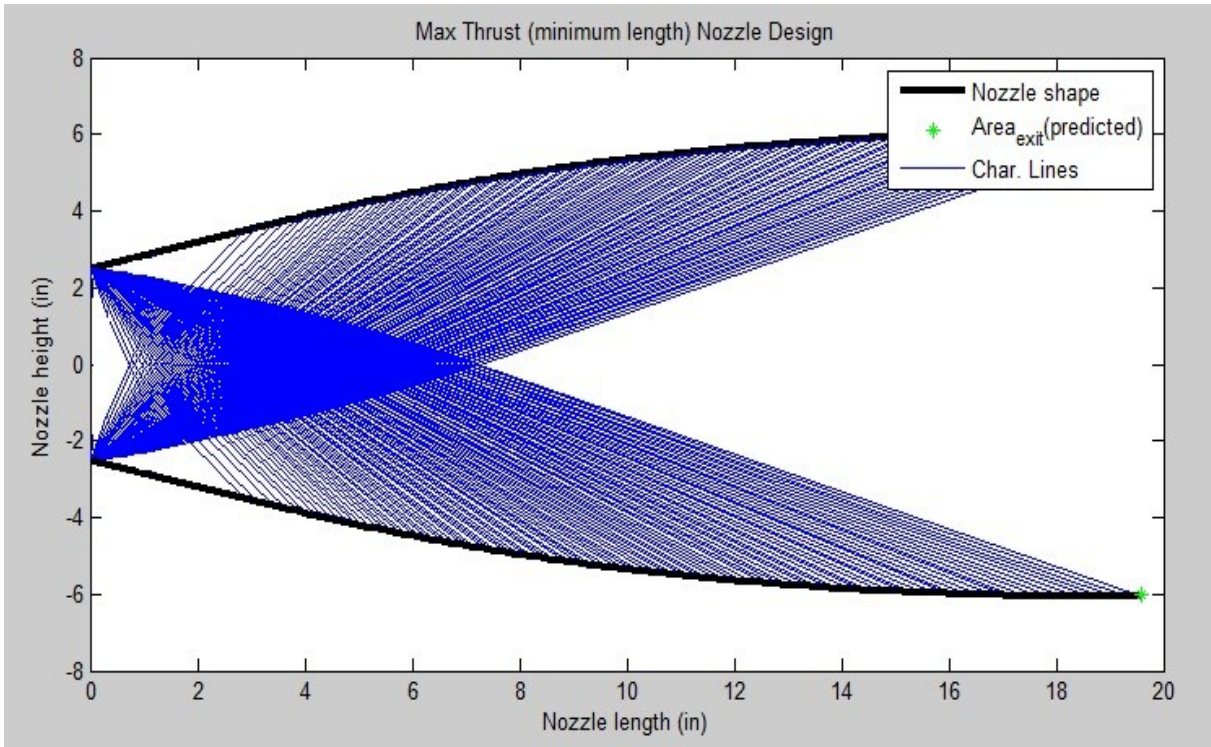


Figure 5-11: Two-Dimensional Plot of the MLN using MOC for 50 lines

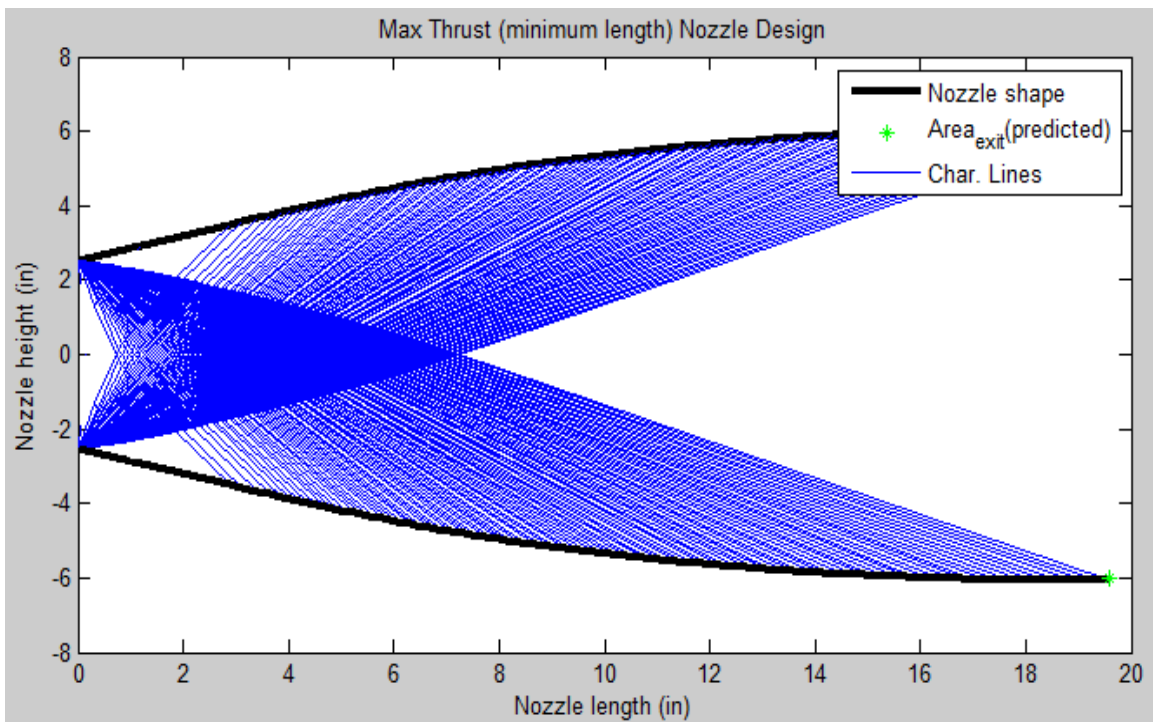


Figure 5-12: Two-Dimensional Plot of the MLN using MOC for 100 lines

5.4 Three-Dimensional Modeling

Three-dimensional modeling of the nozzle, chamber, and injector is done using a CAD program such as Catia, Unigraphics NX, or SolidWorks. In this case, Unigraphics NX was used. The *.dat file was imported into Unigraphics NX as a spline then revolved along the horizontal axis. A three-dimensional model of a 80% Rao nozzle and chamber is shown below in Figure 5-13. The conical nozzle and chamber three-dimensional model is shown in Figure 5-14. A three-dimensional model of a doublet injector face is shown in Figure 5-15.

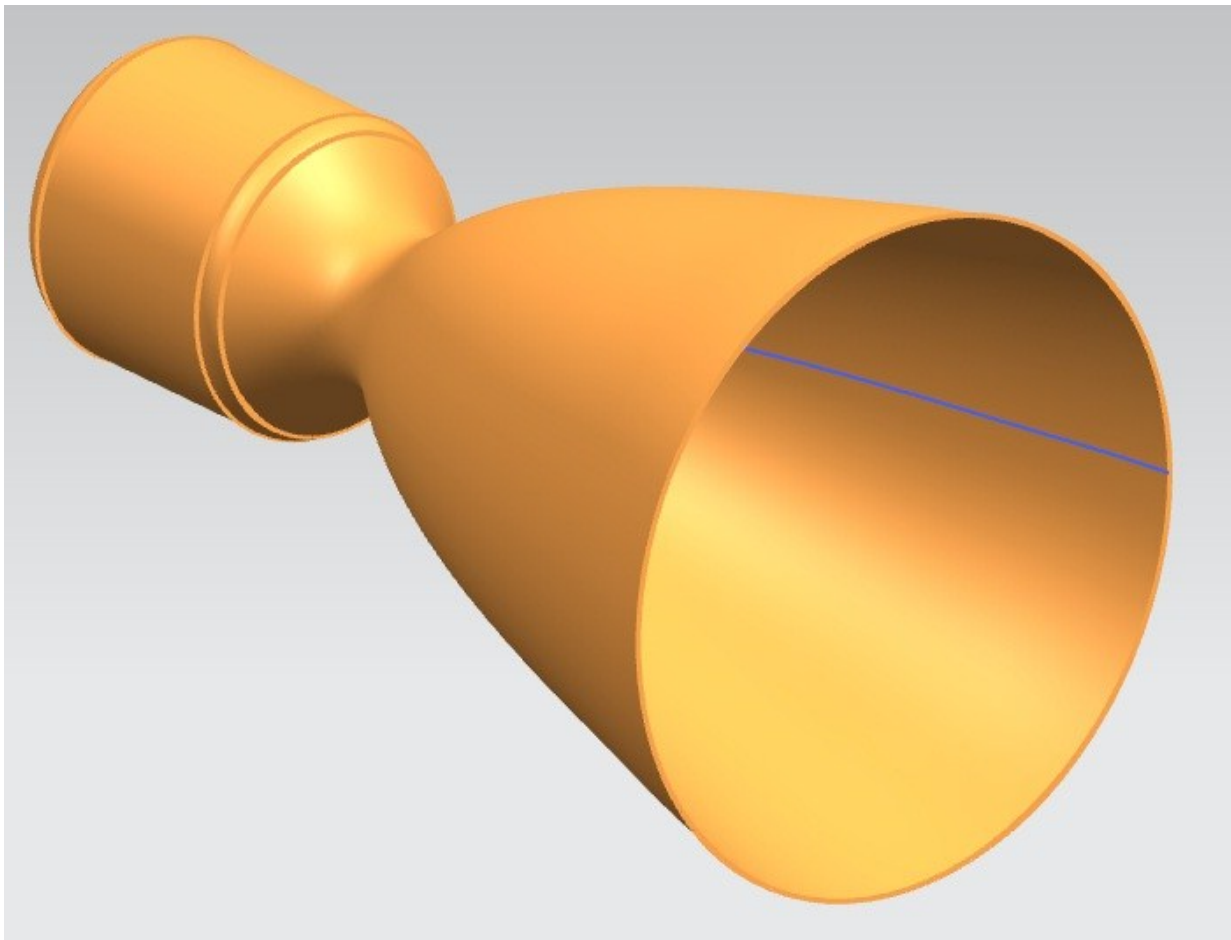


Figure 5-13: 3-D Model of a 80% Rao Nozzle and Chamber (Reference 12)

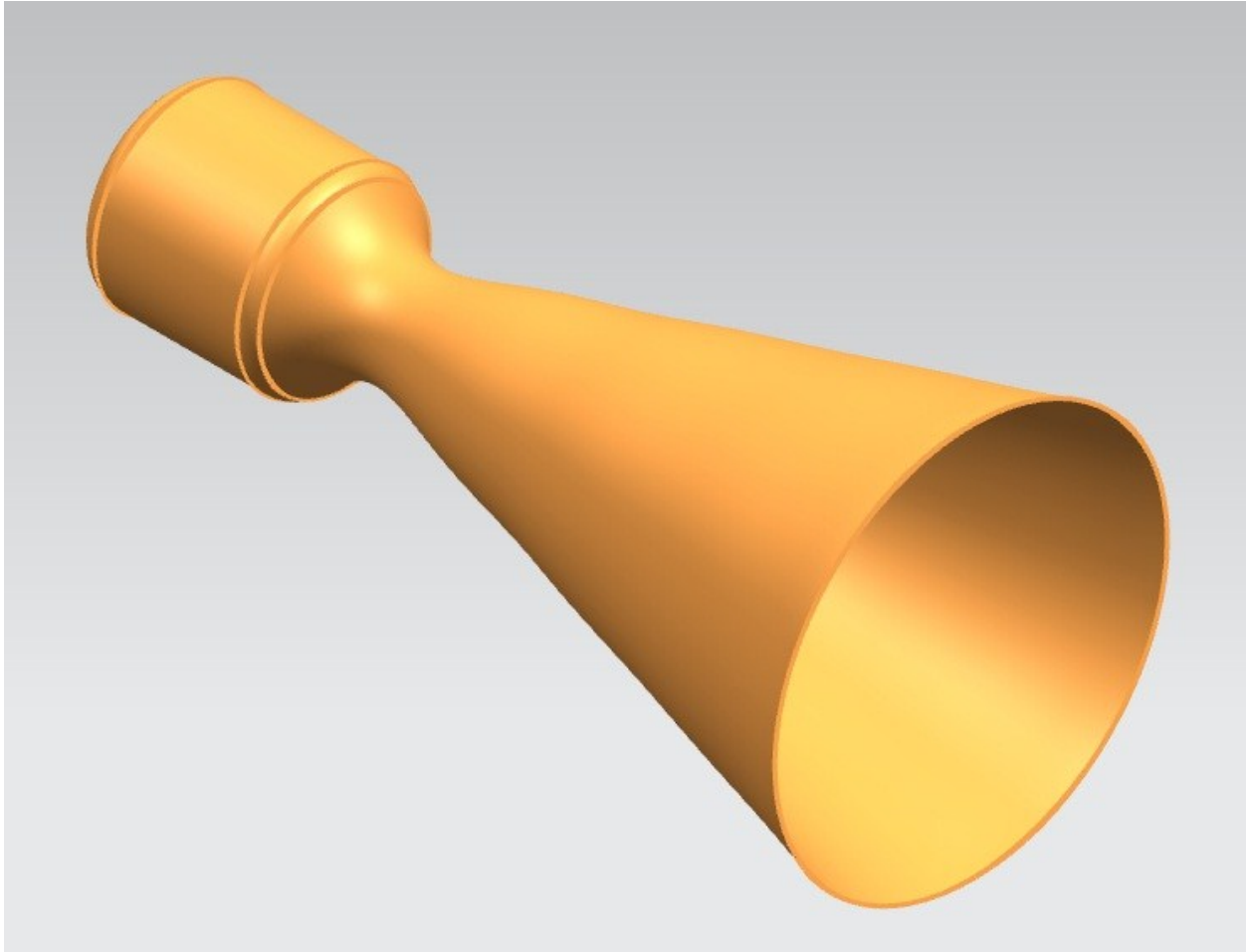


Figure 5-14: 3-D Model of 15 Degree Half Angle Conical Nozzle & Chamber (Reference 12)

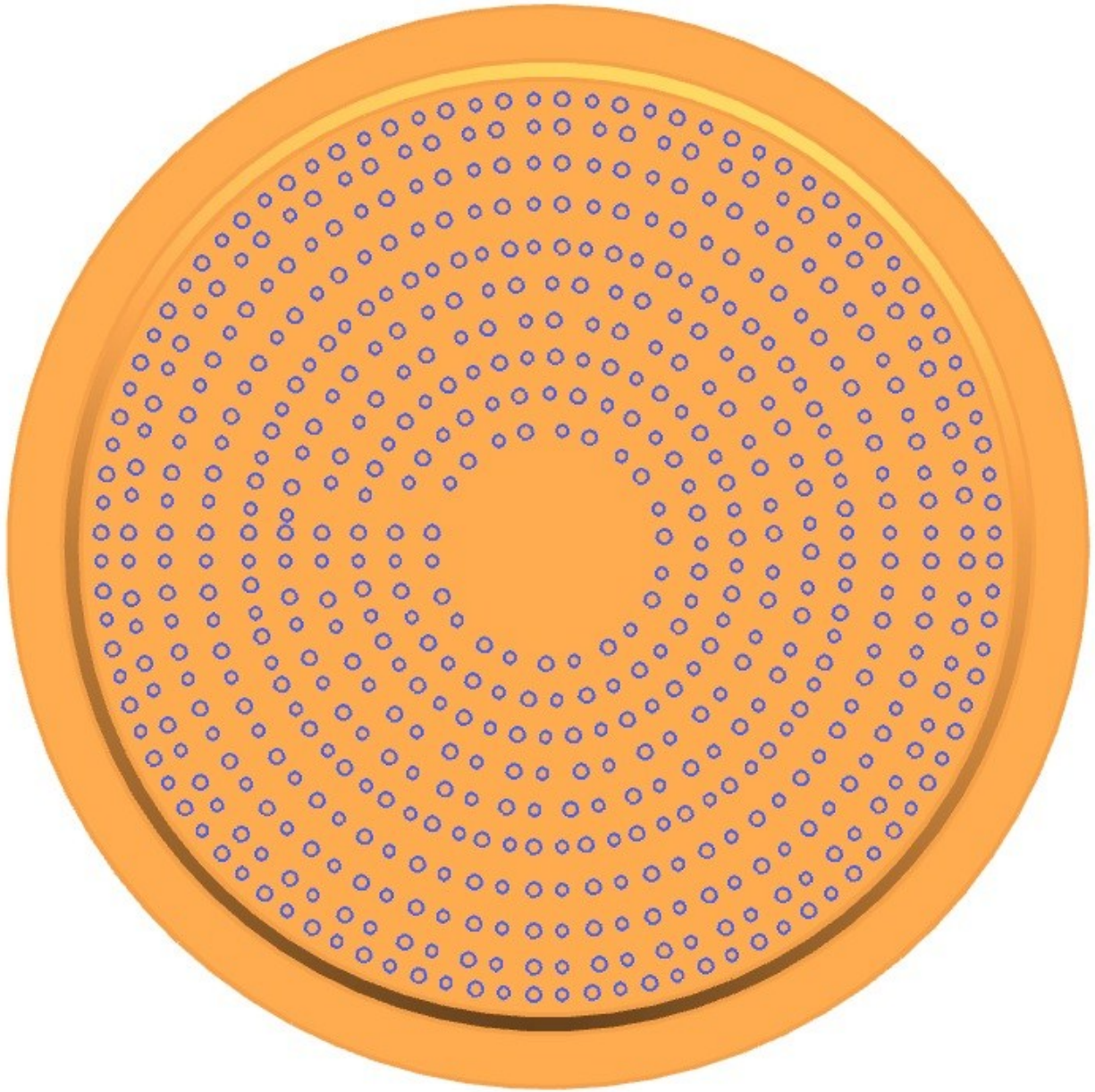


Figure 5-15: 3-D Model of Doublet Injector (Reference 12)

5.5 Simulation Computational Fluid Dynamics (CFD) Example

Computational fluid dynamics (CFD) simulation of a bell nozzle is shown below in Figure 5-17 and Figure 5-18 from Reference 26 as a post process. Unigraphics NX was used to plot the 3-D model of the rocket. The *.dat file was imported into Unigraphics NX as a spline then revolved along the horizontal axis. A three-dimensional model of a 80% Rao nozzle was created. The Rao nozzle mesh grid was created in from ANSYS ICEM CFD and shown below in Figure 5-16.

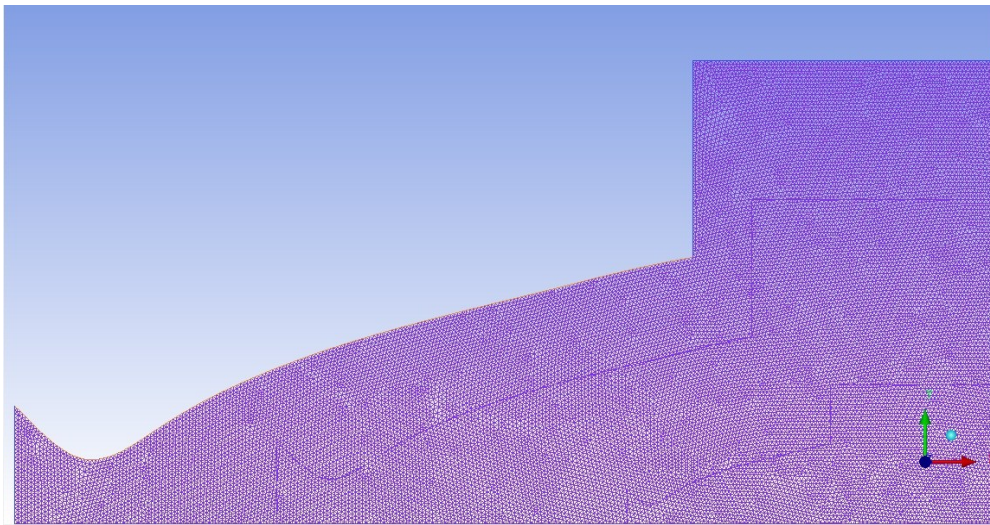


Figure 5-16: 80% Rao Nozzle Mesh Grid (Reference 14)

The CFD model of a steady, fully detached, 2-D nozzle with flow field nozzle pressure ratio, NPR is between 2 to 3.41 that is shown in Figure 5-17 and Figure 5-18 below from Reference 26. The nozzle was over-expanded and dominated by shock-induced boundary layer separation (Reference 26). All the solid walls were treated as no-slip adiabatic surfaces, and the bottom of the entire domain was defined by a symmetry boundary condition in ANSYS Fluent CFD program, according to Reference 26. As the NPR increases the shock increases in size and moves downstream (Reference 26). The results are in agreement with the experimental data shown side by side at each NPR value according to Reference 26.

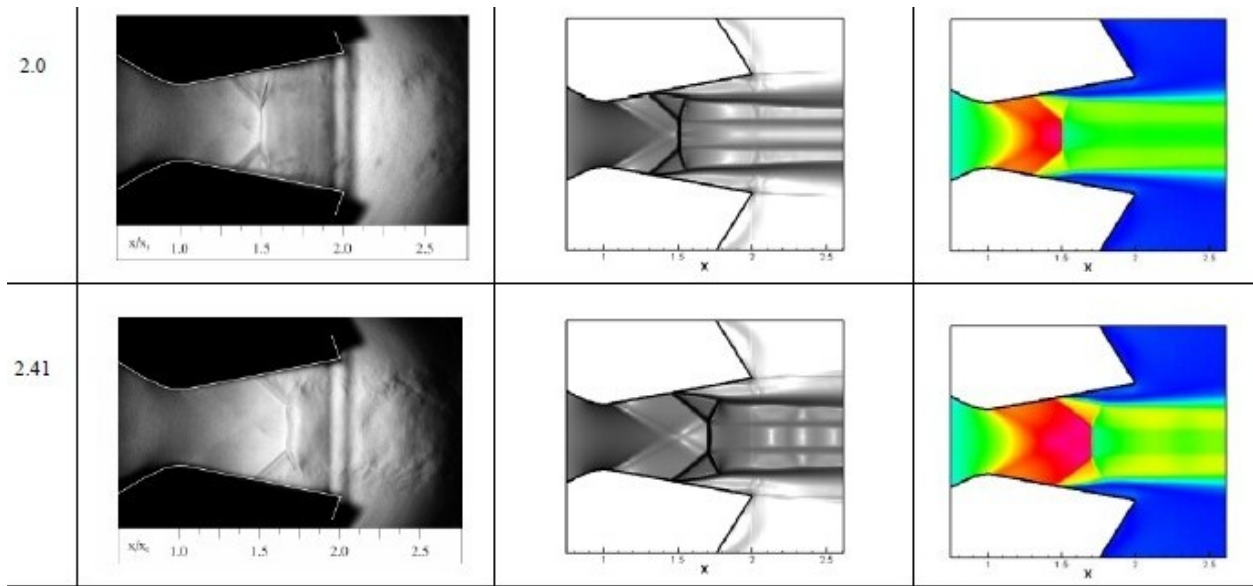


Figure 5-17: Experimental, Computational Schlieren Images, and Mach contours for Baseline Nozzle Configuration at Various NPRs (Reference 26)

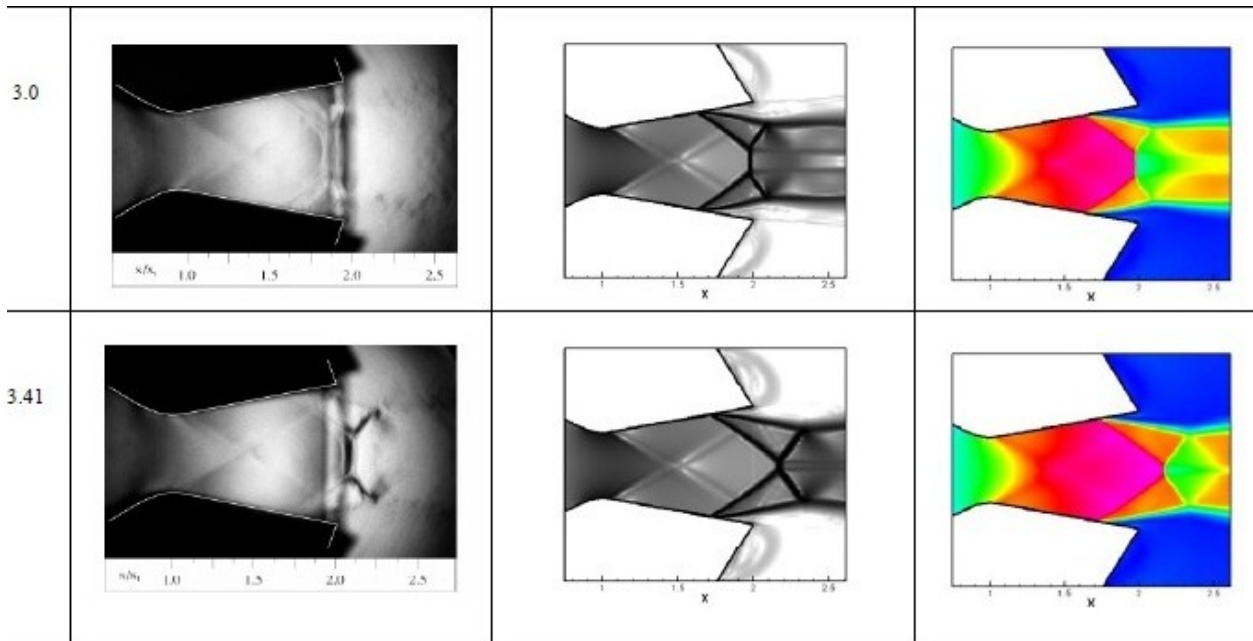


Figure 5-18: Experimental, Computational Schlieren Images, and Mach contours for Baseline Nozzle Configuration at Various NPRs Continued (Reference 26)

6.0 Conclusion and Future Work

In conclusion, the design, analysis, and simulation of the rocket propulsion system is useful in conducting trade studies of rocket propulsion systems. The program aids in unifying the nozzle, chamber and injector portions of a rocket propulsion system design effort quickly and efficiently using a streamlined graphical user interface (GUI). The program also allows for the selection of common nozzle profiles including Rao, conical, bell, and minimum length nozzles (MLN) using method of characteristics (MOC). Chamber dimensions, propellant selections, and injector selection between doublet or triplet allow for further refinement of the desired rocket system design. The program takes the available selections and specifications made by the user and outputs key design parameters calculated from the input variables.

A 2-D graphical representation of the nozzle and/or chamber is plotted and coordinates of the plotted line are displayed. The rocket propulsion system design coordinates are saved to a *.dat file which can be used in a CAD program to plot a 3-D model of the rocket propulsion systems. Coordinates of the injectors are saved to a *.dat file to be modeled in a CAD program as well. The program currently provides a symbolic link in the form of a button on the output page which will open Unigraphics NX CAD program.

The post-processing simulation of the rocket propulsion system is done in a computational fluid dynamics (CFD) program such as ANSYS ICEM CFD mesh generation software and ANSYS FLUENT CFD. The user reads the parasolid or IGES/STEP file of the CAD 3-D modeling of the rocket propulsion system into the ANSYS ICEM CFD meshing software. Using ANSYS FLUENT CFD software, the user can choose to model the flow, turbulence, heat

transfer, air flow over the rocket, combustion in the chamber, or various other options of the rocket propulsion system.

Future work on the design, analysis, and simulation of the rocket propulsion system would be the following:

- Integrated 3-D modeling graphics in the program instead having to open CAD program and manually importing the rocket propulsion system design coordinates to model.
- Multiple design sweeps at the same time in the same program window.
- Detailed injector design selections including more than doublet and triplet injector types.
- More chamber design selections in addition to cylindrical combustion chamber.
- CFD mesh setup in the program instead of importing CAD model into CFD mesh program.
- CFD simulation analysis for multiple nozzle design studies and 3-D unsteady and steady CFD calculations.
- Material choices for the nozzle.
- Heat transfer calculations.
- Commercialize the program for use.

7.0 References

1. Sutton, George P., *History of Liquid Propellant Rocket Engines*, American Institute of Aeronautics and Astronautics, Inc., Reston, Virginia, 2006.
2. Sutton, George P. and Oscar Biblarz, *Rocket Propulsion Elements*, John Wiley & Sons, Inc., 6th Edition. New York, 2001.
3. Anderson, John D, *Modern Compressible Flow with Historical Perspective*, McGraw Hill, 3rd Edition. Boston, 2003.
4. Huzel, Dieter K., and Huang, David H., *Modern Engineering for Design of Liquid-Propellant Rocket Engines*, American Institute of Aeronautics and Astronautics, Inc., Washington, D.C., 1992.
5. Apogee Rockets, “Apogee Components,” RocketSim Software, http://www.apogeerockets.com/Rocket_Software/RockSim/ [retrieved 2 March, 2012].
6. Cipolla, John/ AeroRocket, “AeroSpike,” AeroSpike Software, <http://aerorocket.com/MOC/MOC.html> [retrieved 2 March, 2012].
7. Ponomarenko, Alexander, “Rocket Propulsion Analysis,” Rocket Propulsion Analysis (RPA) Software, <http://propulsion-analysis.com/> [retrieved 14 April, 2012].
8. Pirk, Rogerio, Souto, Carlos d’Andrade, Silveira, Dimas Donizeti da, Souza, Candido Magno de, and Goes, Luiz Carlos Sandoval, *Liquid Rocket Combustion Chamber Acoustic Characterization*, Journal of Aerospace Technology Management, Sao Jose dos Campos, Vol.2, No.3, September-December 2010, pg.269-278.
9. Frey, M. and Hagemann, G., *Status of Flow Separation Prediction in Rocket Nozzles*, AIAA-98-3619, July 1998.

10. NASA, "GRIN: Great Images in NASA," Stennis Space Center Image # 81-201-1, asa.gov/ABSTRACTS/GPN-2000-000543.html [retrieved 25 March 2012].
11. Hulka, James, Vigor Yang, Mohammed Habiballah, and Michael Popp, *Liquid Rocket Thrust Chambers: Aspects of Modeling, Analysis, and Design*, American Institute of Aeronautics and Astronautics, Inc., Reston, Virginia, Vol. 200.
12. Anon., "Unigraphics NX 7.0, Siemens Inc., 2012.
13. Anon., "Matlab R2009a, MathWorks, Inc., 2012.
14. Anon., "ANSYS ICEM CFD, ANSYS, Inc., 2012.
15. Anon., "ANSYS Fluent, ANSYS, Inc., 2012.
16. Hagemann, Gerald, Immicch, Hans, Van Nguyen, Thong, and Dumnov, Gennady E., *Advanced Rocket Nozzles*, Journal of Propulsion and Power, Vol. 14, No. 5, September-October 1998.
17. Shebalin, J-P, and Tiwari, S. N., *NOZ-OP-2D: A CFD-Based Optimization System for Axially Symmetric Rocket Nozzles*, AIAA-2001-1062, January 2001.
18. Welle, R. P., Hardy, B. S., Murdock, J. W., Majamaki, A. J., and Hawkins, G. F., *Separation Instabilities in Overexpanded Nozzles*, AIAA-2003-5239, July 2003.
19. Martelli, Emanuele, Nasuti, Francesco, and Onofri, Marcello, *Thermo-Fluid-Dynamics Analysis of Film Cooling in Overexpanded Rocket NOzzles*, AIAA-2006-5207, July 2006.
20. Nasuti, Francesco, Onofri, Marcello, and Martelli, Emanuele, *Numerical Analysis of Flow Separation Structures in Rocket Nozzles*, AIAA-2007-5473, July 2007.
21. Colonno, M. R., Van der Weide, E., and Alonso, J. J., *The Optimum Vacuum Nozzle: an MDO Approach*, AIAA-2008-911, January 2008.

22. Martelli, Emanuele, Nasuti, Francesco, and Onofri, Marcello, *Numerical Analysis of Film Cooling in Advanced Rocket Nozzles*, AIAA Journal, Vol. 47, No. 11, November 2009.
23. Boccaletto, Luca, and Dussauge, Jean-Paul, *High-Performance Rocket Nozzle Concept*, Journal of Propulsion and Power, Vol. 26, No. 5, September-October 2010.
24. Shimizu, Taro, Kodera, Masatoshi, and Tsuboi, Nobuyuki, *Internal and External Flow of Rocket Nozzle*, Journal of the Earth Simulator, Vol. 9, March 2008.
25. Louisos, W. F., and Hitt, D. L., *Numerical Studies of Thrust Production in 2-D Supersonic Bell Micronozzles*, AIAA-2008-5233, July 2008.
26. Elmiligui, Alaa, Abdol-Hamid, K.S., and Hunter, Craig A., *Numerical Investigation of Flow in an Over-expanded Nozzle with Porous Surface*, AIAA 2005-4159, July 2005.

Appendix A: Main GUI Code

Filename: Thee_thesis.m

```
function varargout = thee_thesis(varargin)
% THEE_THESIS M-file for thee_thesis.fig
%   THEE_THESIS, by itself, creates a new THEE_THESIS or raises the
existing
%   singleton*.
%
%
%   H = THEE_THESIS returns the handle to a new THEE_THESIS or the handle
to
%   the existing singleton*.
%
%   THEE_THESIS('CALLBACK',hObject,eventData,handles,...) calls the local
function named CALLBACK in THEE_THESIS.M with the given input
arguments.
%
%   THEE_THESIS('Property','Value',...) creates a new THEE_THESIS or
raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before thee_thesis_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to thee_thesis_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help thee_thesis

% Last Modified by GUIDE v2.5 16-May-2012 11:25:26

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',  gui_Singleton, ...
                  'gui_OpeningFcn', @thee_thesis_OpeningFcn, ...
                  'gui_OutputFcn',  @thee_thesis_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
%-----
```

```

%-----
% --- Executes just before thee_thesis is made visible.
function thee_thesis_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to thee_thesis (see VARARGIN)

% Choose default command line output for thee_thesis
handles.output = hObject;
set(handles.tab1, 'Visible', 'off')
set(handles.tab2, 'Visible', 'off')
set(handles.tab3, 'Visible', 'off')
set(handles.tab4, 'Visible', 'on')
set(handles.tab5, 'Visible', 'off')

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes thee_thesis wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%Plot picture on startup
raonoz = imread('rao_nozzle.jpeg');
axes(handles.axes6);
image(raonoz);
axis off;

connoz = imread('conical_nozzle.jpeg');
axes(handles.axes7);
image(connoz);
axis off;

idealnoz = imread('ideal_nozzle.jpeg');
axes(handles.axes8);
image(idealnoz);
axis off;
%-----

%-----
% --- Outputs from this function are returned to the command line.
function varargout = thee_thesis_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

%Tab 1: Instructions
function tab1_ResizeFcn(hObject, eventdata, handles)

```

```

% hObject    handle to uipanel1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

%Tab 2: Input

```

```

function tab2_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to uipanel1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

%Tab 3: Output

```

```

function tab3_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to uipanel1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

%Tab 4: Welcome

```

```

function tab4_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to tab4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

%Tab 5: Design Choices

```

```

function tab5_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to tab5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

%-----
%
% [a,map]=imread('images.jpg');
% [r,c,d]=size(a);
% x=ceil(r/60);
% y=ceil(c/60);
% g=a(1:x:end,1:y:end,:);
% g(g==255)=5.5*255;
% set(handles.pushbutton1,'CData',g);
%-----

```

```

%-----
%Tab 1: Instructions

```

```

% --- Executes on button press in instructionsbutton.
function instructionsbutton_Callback(hObject, eventdata, handles)
% hObject    handle to instructionsbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

handles = guidata(thee_thesis);

```

```

set(handles.tab1, 'Visible', 'on');
set(handles.tab2, 'Visible', 'off');
set(handles.tab3, 'Visible', 'off');
set(handles.tab4, 'Visible', 'off');

```

```

set(handles.tab5, 'Visible', 'off');

%Display image
ssme = imread('ssme.jpg');
axes(handles.axes1);
image(ssme);
axis off;

%-----
%Tab 2: Input
% --- Executes on button press in inputbutton.
function inputbutton_Callback(hObject, eventdata, handles)
% hObject    handle to inputbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles = guidata(thee_thesis);

    set(handles.tab1, 'Visible', 'off');
    set(handles.tab2, 'Visible', 'on');
    set(handles.tab3, 'Visible', 'off');
    set(handles.tab4, 'Visible', 'off');
    set(handles.tab5, 'Visible', 'off');
    set(handles.methane_table, 'Visible', 'on');
    set(handles.hydrazine_table, 'Visible', 'off');
    set(handles.hydrogen_table, 'Visible', 'off');
    set(handles.RP1_table, 'Visible', 'off');
    set(handles.UDMH_table, 'Visible', 'off');
    set(handles.flu_hydra_table, 'Visible', 'off');
    set(handles.flu_hydro_table, 'Visible', 'off');
    set(handles.nitrotetro_hydra_table, 'Visible', 'off');
    set(handles.nitrotetro_RP1_table, 'Visible', 'off');
    set(handles.nitrotetro_MMH_table, 'Visible', 'off');

%-----
%Tab 3: Output
% --- Executes on button press in outputbutton.
function outputbutton_Callback(hObject, eventdata, handles)
% hObject    handle to outputbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles = guidata(thee_thesis);

set(handles.tab1, 'Visible', 'off');
set(handles.tab2, 'Visible', 'off');
set(handles.tab3, 'Visible', 'on');
set(handles.tab4, 'Visible', 'off');
set(handles.tab5, 'Visible', 'off');

%-----
%Tab 4: Welcome
% --- Executes on button press in welcomebutton.
function welcomebutton_Callback(hObject, eventdata, handles)
% hObject    handle to welcomebutton (see GCBO)

```



```

% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

handles = guidata(thee_thesis);

set(handles.tab1, 'Visible', 'off');
set(handles.tab2, 'Visible', 'off');
set(handles.tab3, 'Visible', 'off');
set(handles.tab4, 'Visible', 'on');
set(handles.tab5, 'Visible', 'off');

%-----
%Tab 5: Design Choices
% --- Executes on button press in designbutton.
function designbutton_Callback(hObject, eventdata, handles)
% hObject handle to designbutton (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

handles = guidata(thee_thesis);

set(handles.tab1, 'Visible', 'off');
set(handles.tab2, 'Visible', 'off');
set(handles.tab3, 'Visible', 'off');
set(handles.tab4, 'Visible', 'off');
set(handles.tab5, 'Visible', 'on');

%Display image
nozz = imread('nozzlepicture.jpg');
axes(handles.axes5);
image(nozz);
axis off;

cham = imread('rocketss.jpg');
axes(handles.axes10);
image(cham);
axis off;

inject = imread('injectors.jpg');
axes(handles.axes11);
image(inject);
axis off;
%-----
%-----
% --- Executes on button press in selectionbutton.
function selectionbutton_Callback(hObject, eventdata, handles)
% hObject handle to selectionbutton (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

u = get(handles.popupunits, 'Value');
n = get(handles.popupnozzle, 'Value');
c = get(handles.popupchamber, 'Value');
e = get(handles.popupinjector, 'Value');

```

```

%-----
%English

if ( u == 1  && n == 1 && c == 1 && e == 1)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 2 && c == 1 && e == 1)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 3 && c == 1 && e == 1)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 4 && c == 1 && e == 1)
    set(handles.nozzlepanel, 'Visible', 'off');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'on');

%-----
elseif ( u == 1  && n == 1 && c == 1 && e == 2)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 2 && c == 1 && e == 2)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 3 && c == 1 && e == 2)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 4 && c == 1 && e == 2)
    set(handles.nozzlepanel, 'Visible', 'off');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'on');

%-----
elseif ( u == 1  && n == 1 && c == 1 && e == 3)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'on');

```

```

set(handles.injectorpanel, 'Visible', 'off');
set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 2  && c == 1  && e == 3)
set(handles.nozzlepanel, 'Visible', 'on');
set(handles.chamberpanel, 'Visible', 'on');
set(handles.injectorpanel, 'Visible', 'off');
set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 3  && c == 1  && e == 3)
set(handles.nozzlepanel, 'Visible', 'on');
set(handles.chamberpanel, 'Visible', 'on');
set(handles.injectorpanel, 'Visible', 'off');
set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 4  && c == 1  && e == 3)
set(handles.nozzlepanel, 'Visible', 'off');
set(handles.chamberpanel, 'Visible', 'on');
set(handles.injectorpanel, 'Visible', 'off');
set(handles.mocpanel, 'Visible', 'on');
%-----
elseif ( u == 1  && n == 1  && c == 2  && e == 1)
set(handles.nozzlepanel, 'Visible', 'on');
set(handles.chamberpanel, 'Visible', 'off');
set(handles.injectorpanel, 'Visible', 'on');
set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 2  && c == 2  && e == 1)
set(handles.nozzlepanel, 'Visible', 'on');
set(handles.chamberpanel, 'Visible', 'off');
set(handles.injectorpanel, 'Visible', 'on');
set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 3  && c == 2  && e == 1)
set(handles.nozzlepanel, 'Visible', 'on');
set(handles.chamberpanel, 'Visible', 'off');
set(handles.injectorpanel, 'Visible', 'on');
set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 4  && c == 2  && e == 1)
set(handles.nozzlepanel, 'Visible', 'off');
set(handles.chamberpanel, 'Visible', 'off');
set(handles.injectorpanel, 'Visible', 'on');
set(handles.mocpanel, 'Visible', 'on');
%-----
elseif ( u == 1  && n == 1  && c == 2  && e == 2)
set(handles.nozzlepanel, 'Visible', 'on');
set(handles.chamberpanel, 'Visible', 'off');
set(handles.injectorpanel, 'Visible', 'on');
set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 2  && c == 2  && e == 2)
set(handles.nozzlepanel, 'Visible', 'on');
set(handles.chamberpanel, 'Visible', 'off');
set(handles.injectorpanel, 'Visible', 'on');

```

```

    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 3 && c == 2 && e == 2)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'off');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 4 && c == 2 && e == 2)
    set(handles.nozzlepanel, 'Visible', 'off');
    set(handles.chamberpanel, 'Visible', 'off');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'on');
%-----
elseif ( u == 1  && n == 1 && c == 2 && e == 3)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'off');
    set(handles.injectorpanel, 'Visible', 'off');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 2 && c == 2 && e == 3)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'off');
    set(handles.injectorpanel, 'Visible', 'off');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 3 && c == 2 && e == 3)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'off');
    set(handles.injectorpanel, 'Visible', 'off');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 1  && n == 4 && c == 2 && e == 3)
    set(handles.nozzlepanel, 'Visible', 'off');
    set(handles.chamberpanel, 'Visible', 'off');
    set(handles.injectorpanel, 'Visible', 'off');
    set(handles.mocpanel, 'Visible', 'on');
%-----
%Metric

elseif ( u == 2  && n == 1 && c == 1 && e == 1)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 2  && n == 2 && c == 1 && e == 1)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 2  && n == 3 && c == 1 && e == 1)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'on');

```

```

    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 2  && n == 4  && c == 1  && e == 1)
    set(handles.nozzlepanel, 'Visible', 'off');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'on');
%-----
elseif ( u == 2  && n == 1  && c == 1  && e == 2)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 2  && n == 2  && c == 1  && e == 2)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 2  && n == 3  && c == 1  && e == 2)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 2  && n == 4  && c == 1  && e == 2)
    set(handles.nozzlepanel, 'Visible', 'off');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'on');
%-----
elseif ( u == 2  && n == 1  && c == 1  && e == 3)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'off');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 2  && n == 2  && c == 1  && e == 3)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'off');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 2  && n == 3  && c == 1  && e == 3)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'off');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 2  && n == 4  && c == 1  && e == 3)
    set(handles.nozzlepanel, 'Visible', 'off');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'off');

```



```

elseif ( u == 2  && n == 2 && c == 2 && e == 3)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'off');
    set(handles.injectorpanel, 'Visible', 'off');
    set(handles.mocpanel, 'Visible', 'off');

elseif ( u == 2  && n == 3 && c == 2 && e == 3)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'off');
    set(handles.injectorpanel, 'Visible', 'off');
    set(handles.mocpanel, 'Visible', 'off');

else ( u == 2  && n == 4 && c == 2 && e == 3)
    set(handles.nozzlepanel, 'Visible', 'off');
    set(handles.chamberpanel, 'Visible', 'off');
    set(handles.injectorpanel, 'Visible', 'off');
    set(handles.mocpanel, 'Visible', 'on');

end

%-----

%-----

%Popup1: Rao/Conical
% --- Executes on selection change in popup1.
function popup1_Callback(hObject, eventdata, handles)
% hObject      handle to popup1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popup1 contents as cell
array
%      contents{get(hObject,'Value')} returns selected item from popup1

%Pop-up menu Nozzle Selection choices
currentEntry = get(handles.popup1, 'Value');
currentEntry
Captions = get(handles.popup1, 'String');
Captions
Cell = Captions( currentEntry);
Cell
v = char( Cell )
switch v
    case 'Rao'
        set(handles.text3, 'FontName', 'Symbol');
        set(handles.text3, 'String', 'e');
        set(handles.text4, 'FontName', 'Symbol');
        set(handles.text4, 'String', 'q');

    case 'Conical'
        set(handles.text3, 'FontName', 'Symbol');
        set(handles.text3, 'String', 'e');
        set(handles.text4, 'FontName', 'Symbol');
        set(handles.text4, 'String', 'q');

```

```

        case 'Percent of Rao Nozzle'
            set(handles.text3,'FontName','Symbol');
            set(handles.text3,'String','e');
            set(handles.text4,'FontName','Symbol');
            set(handles.text4,'String','q');

end

% --- Executes during object creation, after setting all properties.
function popup1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popup1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%-----

%-----
%Popup2: Fuel Selections
% --- Executes on selection change in popup2.
function popup2_Callback(hObject, eventdata, handles)
% hObject    handle to popup2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popup2 contents as cell
array
%         contents{get(hObject,'Value')} returns selected item from popup2

%Pop-up menu Nozzle Selection choices
currentfuel = get(handles.popup2, 'Value');
currentfuel
Captionsfuel = get(handles.popup2, 'String');
Captionsfuel
Cellfuel = Captionsfuel( currentfuel );
Cellfuel
f = char( Cellfuel )
switch f
    case 'Liquid O2 + Methane'
        set(handles.methane_table, 'Visible', 'on');
        set(handles.hydrazine_table, 'Visible', 'off');
        set(handles.hydrogen_table, 'Visible', 'off');
        set(handles.RP1_table, 'Visible', 'off');
        set(handles.UDMH_table, 'Visible', 'off');
        set(handles.flu_hydra_table, 'Visible', 'off');
        set(handles.flu_hydro_table, 'Visible', 'off');
        set(handles.nitrotetro_hydra_table, 'Visible', 'off');
        set(handles.nitrotetro_RP1_table, 'Visible', 'off');
        set(handles.nitrotetro_MMH_table, 'Visible', 'off');
    case 'Liquid O2 + Hydrazine'
        set(handles.methane_table, 'Visible', 'off');

```



```

set(handles.hydrazine_table, 'Visible', 'on');
set(handles.hydrogen_table, 'Visible', 'off');
set(handles.RP1_table, 'Visible', 'off');
set(handles.UDMH_table, 'Visible', 'off');
set(handles.flu_hydra_table, 'Visible', 'off');
set(handles.flu_hydro_table, 'Visible', 'off');
set(handles.nitrotetro_hydra_table, 'Visible', 'off');
set(handles.nitrotetro_RP1_table, 'Visible', 'off');
set(handles.nitrotetro_MMH_table, 'Visible', 'off');
case 'Liquid O2 + Hydrogen'
set(handles.methane_table, 'Visible', 'off');
set(handles.hydrazine_table, 'Visible', 'off');
set(handles.hydrogen_table, 'Visible', 'on');
set(handles.RP1_table, 'Visible', 'off');
set(handles.UDMH_table, 'Visible', 'off');
set(handles.flu_hydra_table, 'Visible', 'off');
set(handles.flu_hydro_table, 'Visible', 'off');
set(handles.nitrotetro_hydra_table, 'Visible', 'off');
set(handles.nitrotetro_RP1_table, 'Visible', 'off');
set(handles.nitrotetro_MMH_table, 'Visible', 'off');
case 'Liquid O2 + RP-1'
set(handles.methane_table, 'Visible', 'off');
set(handles.hydrazine_table, 'Visible', 'off');
set(handles.hydrogen_table, 'Visible', 'off');
set(handles.RP1_table, 'Visible', 'on');
set(handles.UDMH_table, 'Visible', 'off');
set(handles.flu_hydra_table, 'Visible', 'off');
set(handles.flu_hydro_table, 'Visible', 'off');
set(handles.nitrotetro_hydra_table, 'Visible', 'off');
set(handles.nitrotetro_RP1_table, 'Visible', 'off');
set(handles.nitrotetro_MMH_table, 'Visible', 'off');
case 'Liquid O2 + UDMH'
set(handles.methane_table, 'Visible', 'off');
set(handles.hydrazine_table, 'Visible', 'off');
set(handles.hydrogen_table, 'Visible', 'off');
set(handles.RP1_table, 'Visible', 'off');
set(handles.UDMH_table, 'Visible', 'on');
set(handles.flu_hydra_table, 'Visible', 'off');
set(handles.flu_hydro_table, 'Visible', 'off');
set(handles.nitrotetro_hydra_table, 'Visible', 'off');
set(handles.nitrotetro_RP1_table, 'Visible', 'off');
set(handles.nitrotetro_MMH_table, 'Visible', 'off');
case 'Liquid Fluorine + Hydrazine'
set(handles.methane_table, 'Visible', 'off');
set(handles.hydrazine_table, 'Visible', 'off');
set(handles.hydrogen_table, 'Visible', 'off');
set(handles.RP1_table, 'Visible', 'off');
set(handles.UDMH_table, 'Visible', 'off');
set(handles.flu_hydra_table, 'Visible', 'on');
set(handles.flu_hydro_table, 'Visible', 'off');
set(handles.nitrotetro_hydra_table, 'Visible', 'off');
set(handles.nitrotetro_RP1_table, 'Visible', 'off');
set(handles.nitrotetro_MMH_table, 'Visible', 'off');
case 'Liquid Fluorine + Hydrogen'
set(handles.methane_table, 'Visible', 'off');
set(handles.hydrazine_table, 'Visible', 'off');
set(handles.hydrogen_table, 'Visible', 'off');

```

```

set(handles.RP1_table, 'Visible', 'off');
set(handles.UDMH_table, 'Visible', 'off');
set(handles.flu_hydra_table, 'Visible', 'off');
set(handles.flu_hydro_table, 'Visible', 'on');
set(handles.nitrotetro_hydra_table, 'Visible', 'off');
set(handles.nitrotetro_RP1_table, 'Visible', 'off');
set(handles.nitrotetro_MMH_table, 'Visible', 'off');
case 'Liquid Nitrogen tetroxide + Hydrazine'
set(handles.methane_table, 'Visible', 'off');
set(handles.hydrazine_table, 'Visible', 'off');
set(handles.hydrogen_table, 'Visible', 'off');
set(handles.RP1_table, 'Visible', 'off');
set(handles.UDMH_table, 'Visible', 'off');
set(handles.flu_hydra_table, 'Visible', 'off');
set(handles.flu_hydro_table, 'Visible', 'off');
set(handles.nitrotetro_hydra_table, 'Visible', 'on');
set(handles.nitrotetro_RP1_table, 'Visible', 'off');
set(handles.nitrotetro_MMH_table, 'Visible', 'off');
case 'Liquid Nitrogen tetroxide + RP-1'
set(handles.methane_table, 'Visible', 'off');
set(handles.hydrazine_table, 'Visible', 'off');
set(handles.hydrogen_table, 'Visible', 'off');
set(handles.RP1_table, 'Visible', 'off');
set(handles.UDMH_table, 'Visible', 'off');
set(handles.flu_hydra_table, 'Visible', 'off');
set(handles.flu_hydro_table, 'Visible', 'off');
set(handles.nitrotetro_hydra_table, 'Visible', 'off');
set(handles.nitrotetro_RP1_table, 'Visible', 'on');
set(handles.nitrotetro_MMH_table, 'Visible', 'off');
case 'Liquid Nitrogen tetroxide + MMH (Monomethyl-hydrazine)'
set(handles.methane_table, 'Visible', 'off');
set(handles.hydrazine_table, 'Visible', 'off');
set(handles.hydrogen_table, 'Visible', 'off');
set(handles.RP1_table, 'Visible', 'off');
set(handles.UDMH_table, 'Visible', 'off');
set(handles.flu_hydra_table, 'Visible', 'off');
set(handles.flu_hydro_table, 'Visible', 'off');
set(handles.nitrotetro_hydra_table, 'Visible', 'off');
set(handles.nitrotetro_RP1_table, 'Visible', 'off');
set(handles.nitrotetro_MMH_table, 'Visible', 'on');
end

% --- Executes during object creation, after setting all properties.
function popup2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popup2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUiControlBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end
end
%-----

```

```

%-----
% --- Executes on selection change in popup3.
function popup3_Callback(hObject, eventdata, handles)
% hObject    handle to popup3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popup3 contents as
cell array
%         contents{get(hObject,'Value')} returns selected item from popup3

currentAlt = get(handles.popup3, 'Value');
currentAlt
CaptionAlt = get(handles.popup3, 'String');
CaptionAlt
CellAlt = CaptionAlt( currentAlt );
CellAlt
aa = char( CellAlt )
switch aa
    case '0 (Sea Level)'
        set(handles.edit43, 'String', 14.7);
    case '5,000'
        set(handles.edit43, 'String', 12.228);
    case '10,000'
        set(handles.edit43, 'String', 10.108);
    case '15,000'
        set(handles.edit43, 'String', 8.297);
    case '20,000'
        set(handles.edit43, 'String', 6.759);
    case '25,000'
        set(handles.edit43, 'String', 5.461);
    case '30,000'
        set(handles.edit43, 'String', 4.373);
    case '35,000'
        set(handles.edit43, 'String', 3.468);
    case '40,000'
        set(handles.edit43, 'String', 2.730);
    case '45,000'
        set(handles.edit43, 'String', 2.149);
    case '50,000'
        set(handles.edit43, 'String', 1.692);
    case '60,000'
        set(handles.edit43, 'String', 1.049);
    case '70,000'
        set(handles.edit43, 'String', 0.651);
    case '80,000'
        set(handles.edit43, 'String', 0.406);
    case '90,000'
        set(handles.edit43, 'String', 0.255);
    case '100,000'
        set(handles.edit43, 'String', 0.162);
end

% --- Executes during object creation, after setting all properties.
function popup3_CreateFcn(hObject, eventdata, handles)

```

```

% hObject    handle to popup3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%-----

%-----
% --- Executes on selection change in popup4.
function popup4_Callback(hObject, eventdata, handles)
% hObject    handle to popup4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popup4 contents as
cell array
%         contents{get(hObject,'Value')} returns selected item from popup4

currentInj = get(handles.popup4, 'Value');
currentInj
CaptionInj = get(handles.popup4, 'String');
CaptionInj
CellInj = CaptionInj( currentInj );
CellInj
inj = char( CellInj )
switch inj
    case 'Doublet'
        set(handles.edit59, 'String', 0.8);
        set(handles.edit60, 'String', 0.2);
        set(handles.edit62, 'String', 0.035);
    case 'Triplet'
        set(handles.edit59, 'String', 0.8);
        set(handles.edit60, 'String', 0.2);
        set(handles.edit62, 'String', 0.035);
end

% --- Executes during object creation, after setting all properties.
function popup4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popup4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%-----

```

```

%-----
% --- Executes on button press in okbutton1.
function okbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to okbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
validity = 0;

    strper = get( handles.edit39, 'String' );
    strper = char( strper );
    [per, status] = str2num( strper );
    if( 1 == status )
        validity = validity + 1;
    end

    strRt = get( handles.edit1, 'String' );
    strRt = char( strRt );
    [Rt, status] = str2num( strRt );
    [CRT, status] = str2num( strRt);
    if( 1 == status )
        validity = validity + 1;
    end

    strarea_ratio = get( handles.edit2, 'String' );
    strarea_ratio = char( strarea_ratio );
    [area_ratio, status] = str2num( strarea_ratio );
    [Carea_ratio, status] = str2num( strarea_ratio );
    if( 1 == status )
        validity = validity + 1;
    end

    strtheta = get( handles.edit3, 'String' );
    strtheta = char( strtheta );
    [theta, status] = str2num( strtheta );
    [Ctheta, status] = str2num( strtheta );
    if( 1 == status )
        validity = validity + 1;
    end

    strP3 = get( handles.edit43, 'String' );
    strP3 = char( strP3 );
    [P3, status] = str2num( strP3 );
    if( 1 == status )
        validity = validity + 1;
    end

    strF = get( handles.edit41, 'String' );
    strF = char( strF );
    [F, status] = str2num( strF );
    if( 1 == status )
        validity = validity + 1;
    end

    strtp = get( handles.edit14, 'String' );

```

```

strtp = char( strtp );
[tp, status] = str2num( strtp );
    if( 1 == status )
        validity = validity + 1;
    end

strL_star = get( handles.edit45, 'String' );
strL_star = char( strL_star );
[L_star, status] = str2num( strL_star );
    if( 1 == status )
        validity = validity + 1;
    end

strr = get( handles.edit6, 'String' );
strr = char( strr );
[r, status] = str2num( strr );
    if( 1 == status )
        validity = validity + 1;
    end

strSG_f = get( handles.edit7, 'String' );
strSG_f = char( strSG_f );
[den_f, status] = str2num( strSG_f );
    if( 1 == status )
        validity = validity + 1;
    end

strSG_o = get( handles.edit8, 'String' );
strSG_o = char( strSG_o );
[den_o, status] = str2num( strSG_o );
    if( 1 == status )
        validity = validity + 1;
    end

strmol = get( handles.edit10, 'String' );
strmol = char( strmol );
[mol, status] = str2num( strmol );
    if( 1 == status )
        validity = validity + 1;
    end

strgamma = get( handles.edit11, 'String' );
strgamma = char( strgamma );
[gamma, status] = str2num( strgamma );
    if( 1 == status )
        validity = validity + 1;
    end

strPc = get( handles.edit13, 'String' );
strPc = char( strPc );
[P1, status] = str2num( strPc );
    if( 1 == status )
        validity = validity + 1;
    end

```

```

strTc = get( handles.edit9, 'String' );
strTc = char( strTc );
[T1, status] = str2num( strTc );
    if( 1 == status )
        validity = validity + 1;
    end

strCd = get( handles.edit59, 'String' );
strCd = char( strCd );
[Cd, status] = str2num( strCd );
if( 1 == status )
    validity = validity + 1;
end

strPd = get( handles.edit60, 'String' );
strPd = char( strPd );
[Pd, status] = str2num( strPd );
if( 1 == status )
    validity = validity + 1;
end

strDf = get( handles.edit62, 'String' );
strDf = char( strDf );
[Df, status] = str2num( strDf );
if( 1 == status )
    validity = validity + 1;
end

if ( 18 == validity)
    h = get(handles.popup1, 'Value' );
    h
    ff = get(handles.popup2, 'Value');
    ff
    inje = get(handles.popup4, 'Value');
    inje
    if(h == 1 )
        [Rao_x, Rao_y,x_conver,y_conver,x_cyl,y_cyl,Ln,theta_exit,nf] =
Rao(per,Rt,area_ratio,theta,P3,F,tp,L_star,r,den_f,den_o,mol,gamma,P1,T1,Cd,P
d,Df)
        % [Rao_x, Rao_y,x_conver,y_conver,x_cyl,y_cyl,Ln,theta_exit] =
Rao(per,Rt,area_ratio,theta,P3,F,tp,L_star,r,den_f,den_o,mol,gamma,P1,T1)
        set(handles.edit18, 'String', num2str(Ln));
        set(handles.edit19, 'String', num2str(theta_exit));
        set(handles.edit46, 'String', num2str(nf));
        % set(handles.edit47, 'String', num2str(no));
        plot(handles.axes2,x_cyl,y_cyl,'b',x_cyl,-
y_cyl,'b',x_conver,y_conver,'b',x_conver,-y_conver,'b',Rao_x,
Rao_y,'b',Rao_x, -Rao_y,'b');
        title(handles.axes2, 'Rao Nozzle', 'FontWeight', 'bold');
        xlabel(handles.axes2, 'Length of Nozzle, L_n (in.)');
        ylabel(handles.axes2, 'Radius of Nozzle, R_n (in.)');
        set(handles.axes2, 'XGrid', 'on');
        grid on;
        set(handles.axes2, 'YGrid', 'on');
        grid on;

```

```

Raox = [x_cyl(1,:),x_conver(:,:),Rao_x(:,:)] ;
Raoy = [y_cyl(1,:),y_conver(:,:),Rao_y(:,:)] ;
array1 = num2cell( [Raox', Raoy'] );
set(handles.xy_table,'Data', array1);

else(h == 2 )
%      && ff == 2
[Cx_FC, Cy_FC,Cx_CSC, Cy_CSC,Rao_Cx, Rao_Cy,CLn,Ctheta_exit] =
Conical(CRT,Carea_ratio,Ctheta);
set(handles.edit18,'String',num2str(CLn));
set(handles.text10,'Visible','off');
set(handles.edit19,'Visible','off');
plot(handles.axes2, Cx_FC, Cy_FC,'b',Cx_FC, -Cy_FC,'b',Cx_CSC,
Cy_CSC,'b', Cx_CSC, -Cy_CSC,'b',Rao_Cx, Rao_Cy,'b', Rao_Cx,-Rao_Cy,'b');
title(handles.axes2,'Conical Nozzle','FontWeight','bold');
xlabel(handles.axes2,'Length of Nozzle, L_n (in.)');
ylabel(handles.axes2,'Radius of Nozzle, R_n (in.)');
set(handles.axes2,'XGrid','on');
grid on;
set(handles.axes2,'YGrid','on');
grid on;

end
end
%-----
%-----
% --- Executes on button press in okbutton2.
function okbutton2_Callback(hObject, eventdata, handles)
% hObject      handle to okbutton2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

validity1 = 0;

strT_c= get( handles.edit20, 'String' );
strT_c = char( strT_c );
[T_c, status] = str2num( strT_c );
if( 1 == status )
    validity1 = validity1 + 1;
end

strP_c= get( handles.edit21, 'String' );
strP_c = char( strP_c );
[P_c, status] = str2num( strP_c );
if( 1 == status )
    validity1 = validity1 + 1;
end

strwidth= get( handles.edit22, 'String' );
strwidth = char( strwidth);
[width, status] = str2num( strwidth );
if( 1 == status )

```



```

        validity1 = validity1 + 1;
    end

    strnum= get( handles.edit38, 'String' );
    strnum = char( strnum);
    [num, status] = str2num( strnum );
    if( 1 == status )
        validity1 = validity1 + 1;
    end

if ( 4 == validity1)
    h = get( handles.popup1, 'Value' );
    h
%     if(strcmp( h, 'Rao' ) == 1 )
%     if(h == 1)
        [Ae,TT,A_max,Max_thrust,noz,i,c1,c2,aa,cc] =
MOC_code(T_c,P_c,width,num)
        plot(handles.axes3,noz(:,1),noz(:,2),'k',noz(:,1),-noz(:,2),'k',aa,-
A_max/width/2,'r*',aa,A_max/width/2,'r*',c1,c2,'b',c1,-c2,'b')
%         plot(handles.axes3,c1,c2,c1,-c2)
%     hold on;
%     plot(char(i,:,1),-char(i,:,2))
%         %plot(handles.axes3,Ae,TT);
        title(handles.axes3,'MOC', 'FontWeight', 'bold');
        xlabel(handles.axes3,'MOC Length, L_n (in.)');
        ylabel(handles.axes3,'MOC, R_n (in.)');
        set(handles.axes3,'XGrid','on');
        grid on;
        set(handles.axes3,'YGrid','on');
        grid on;
    else(h == 2)
        [Ae,TT,A_max,Max_thrust,noz,i,c1,c2,aa,cc] =
MOC_code(T_c,P_c,width,num)
        plot(handles.axes3,Ae,TT);
        title(handles.axes3,'Concial Nozzle', 'FontWeight', 'bold');
        xlabel(handles.axes3,'Length of Nozzle, L_n (in.)');
        ylabel(handles.axes3,'Radius of Nozzle, R_n (in.)');
        set(handles.axes3,'XGrid','on');
        grid on;
        set(handles.axes3,'YGrid','on');
        grid on;

    end
end
%-----
%-----
% --- Executes on button press in default_button.
function default_button_Callback(hObject, eventdata, handles)
% hObject      handle to default_button (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

set(handles.nozzlepanel,'Visible','on');
set(handles.chamberpanel,'Visible','on');
set(handles.injectorpanel,'Visible','on');

```

```

set(handles.mocpanel, 'Visible', 'off');
set(handles.edit1, 'String', 5);
set(handles.edit2, 'String', 77.5);
set(handles.edit3, 'String', 45);
set(handles.edit6, 'String', 2.24);
set(handles.edit7, 'String', 1.14);
set(handles.edit8, 'String', 0.58);
set(handles.edit9, 'String', 6500);
set(handles.edit10, 'String', 21.9);
set(handles.edit11, 'String', 1.24);
set(handles.edit41, 'String', 35000);
set(handles.edit13, 'String', 2000);
set(handles.edit14, 'String', 2);
set(handles.edit20, 'String', 2000);
set(handles.edit21, 'String', 1.26e6);
set(handles.edit22, 'String', 0.1);
set(handles.edit38, 'String', 15);
set(handles.edit45, 'String', 118);
set(handles.edit43, 'String', 3.44);
set(handles.edit39, 'String', 0.8);
set(handles.edit59, 'String', 0.8);
set(handles.edit60, 'String', 0.2);
set(handles.edit62, 'String', 0.035);
%-----

%-----
% --- Executes on button press in resetbutton.
function resetbutton_Callback(hObject, eventdata, handles)
% hObject      handle to resetbutton (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

reset = get(handles.resetbutton, 'Value');
reset;
if (reset == 1)
    set(handles.nozzlepanel, 'Visible', 'on');
    set(handles.chamberpanel, 'Visible', 'on');
    set(handles.injectorpanel, 'Visible', 'on');
    set(handles.mocpanel, 'Visible', 'on');
    set(handles.edit1, 'String', {});
    set(handles.edit2, 'String', {});
    set(handles.edit3, 'String', {});
    set(handles.edit6, 'String', {});
    set(handles.edit7, 'String', {});
    set(handles.edit8, 'String', {});
    set(handles.edit9, 'String', {});
    set(handles.edit10, 'String', {});
    set(handles.edit11, 'String', {});
    set(handles.edit41, 'String', {});
    set(handles.edit13, 'String', {});
    set(handles.edit14, 'String', {});
    set(handles.edit18, 'String', {});
    set(handles.edit19, 'String', {});
    set(handles.edit20, 'String', {});
    set(handles.edit21, 'String', {});
    set(handles.edit22, 'String', {});

```

```

        set(handles.edit38,'String',{});
        set(handles.edit45,'String',{});
        set(handles.edit43,'String',{});
        set(handles.edit39,'String',{});
        set(handles.xy_table,'Data',{});
        cla(handles.axes2);
        cla(handles.axes3);
else
end

end
%-----
%-----
% --- Executes on button press in savebutton.
function savebutton_Callback(hObject, eventdata, handles)
% hObject      handle to savebutton (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

fileName = inputdlg('Please enter the name for your program');
directoryName = uigetdir('','Please select a folder to save to');
if directoryName == 0      %# User pressed the "Cancel" button...
    directoryName = '';    %# ...so choose the empty string for the folder
end
filePath = fullfile(directoryName,fileName{1}); %# Create the file path
hgsave(filePath);

%Opens figure of plot to save
% [x_FC, y_FC,x_SC, y_SC, Rao_x,
Rao_y,x,y,ln,theta_exit,x_cyl,y_cyl,x_conver,y_conver] =
Rao(Rt,area_ratio,theta,r,SG_f,SG_o,Tl,mol,gamma,F,P1,tp)
% hplot = plot(handles.axes2,x_FC,y_FC);
%
% ftmp = figure; atmp = axes;
% copyobj(hplot, atmp);
% saveas(ftmp, FileName);
% delete(ftmp);

%SAVES SCREENSHOT
% fileName = inputdlg('Please enter the name for your figures');
% directoryName = uigetdir('','Please select a folder to save to');
% if directoryName == 0      %# User pressed the "Cancel" button...
%     directoryName = '';    %# ...so choose the empty string for the
folder
% end
% filePath = fullfile(directoryName,fileName{1}); %# Create the file path
% extensions = {'fig','bmp'};
% for k = 1:length(extensions)
%     saveas(gcf,filePath,extensions{k}); %# Save the file
%     set(gcf,'PaperPositionMode','auto');
% end
% --- Executes on button press in savedat_file.
%-----

```

```

%-----
function savedat_file_Callback(hObject, eventdata, handles)
% hObject      handle to savedat_file (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

%Create a Import File for NX Spline
% fid=fopen('RaoNozzle.dat','w+')
%     for i =1: length(Rao_x)
%         fprintf(fid, '%6.2f %6.2f 0\n', Rao_x(i),Rao_y(i)')
%     end
%     else(h == 2)
%     end
% end

validity = 0;

strper = get( handles.edit39, 'String' );
strper = char( strper );
[per, status] = str2num( strper );
if( 1 == status )
    validity = validity + 1;
end

strRt = get( handles.edit1, 'String' );
strRt = char( strRt );
[Rt, status] = str2num( strRt );
[CRt,status] = str2num( strRt);
if( 1 == status )
    validity = validity + 1;
end

strarea_ratio = get( handles.edit2, 'String' );
strarea_ratio = char( strarea_ratio );
[area_ratio, status] = str2num( strarea_ratio );
[Carea_ratio, status] = str2num( strarea_ratio );
if( 1 == status )
    validity = validity + 1;
end

strtheta = get( handles.edit3, 'String' );
strtheta = char( strtheta );
[theta, status] = str2num( strtheta );
[Ctheta, status] = str2num( strtheta );
if( 1 == status )
    validity = validity + 1;
end

% stralt= get( handles.edit40, 'String' );
% stralt = char( stralt );
% [alt, status] = str2num( stralt );
%     if( 1 == status )
%         validity = validity + 1;
%     end

```

```

strP3= get( handles.edit43, 'String' );
strP3 = char( strP3 );
[P3, status] = str2num( strP3 );
    if( 1 == status )
        validity = validity + 1;
    end

strF= get( handles.edit41, 'String' );
strF = char( strF );
[F, status] = str2num( strF );
    if( 1 == status )
        validity = validity + 1;
    end

strtp= get( handles.edit14, 'String' );
strtp = char( strtp );
[tp, status] = str2num( strtp );
    if( 1 == status )
        validity = validity + 1;
    end

strL_star= get( handles.edit45, 'String' );
strL_star = char( strL_star );
[L_star, status] = str2num( strL_star );
    if( 1 == status )
        validity = validity + 1;
    end

strr= get( handles.edit6, 'String' );
strr = char( strr );
[r, status] = str2num( strr );
    if( 1 == status )
        validity = validity + 1;
    end

strSG_f= get( handles.edit7, 'String' );
strSG_f = char( strSG_f );
[den_f, status] = str2num( strSG_f );
    if( 1 == status )
        validity = validity + 1;
    end

strSG_o= get( handles.edit8, 'String' );
strSG_o = char( strSG_o );
[den_o, status] = str2num( strSG_o );
    if( 1 == status )
        validity = validity + 1;
    end

strmol= get( handles.edit10, 'String' );
strmol = char( strmol );
[mol, status] = str2num( strmol );
    if( 1 == status )
        validity = validity + 1;
    end

```

```

end

strgamma= get( handles.edit11, 'String' );
strgamma = char( strgamma );
[gamma, status] = str2num( strgamma );
if( 1 == status )
    validity = validity + 1;
end

strPc= get( handles.edit13, 'String' );
strPc = char( strPc );
[P1, status] = str2num( strPc );
if( 1 == status )
    validity = validity + 1;
end

strTc= get( handles.edit9, 'String' );
strTc = char( strTc );
[T1, status] = str2num( strTc );
if( 1 == status )
    validity = validity + 1;
end

if ( 15 == validity)
    h = get( handles.popup1, 'Value' );
    h
    ff = get(handles.popup2, 'Value');
    ff
    if(h == 1 )
        [Rao_x, Rao_y,x_conver,y_conver,x_cyl,y_cyl,Ln,theta_exit] =
Rao(per,Rt,area_ratio,theta,P3,F,tp,L_star,r,den_f,den_o,mol,gamma,P1,T1)
        Raox = [x_cyl(1,:),x_conver(:,,:),Rao_x(:,,:)];
        Raoy = [y_cyl(1,:),y_conver(:,,:),Rao_y(:,,:) ];
        array1 = num2cell( [Raox', Raoy'] );
        gg = get(handles.xy_table, 'Data');
    end
end

% uiputfile
% rao=[Raox Raoy]
% %Create a Import File for NX Spline
% fid=fopen('Nozzle_data.dat','w+')
% for i = 1:length(Raox)
%     fprintf(fid, '%2.2f %2.2f 0\n',rao);
% end

% fprintf(fid,%)
%-----
%-----
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text

```

```

%         str2double(get(hObject,'String')) returns contents of edit1 as a
double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%         str2double(get(hObject,'String')) returns contents of edit2 as a
double

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%         str2double(get(hObject,'String')) returns contents of edit3 as a
double

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)

```

```

% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%         str2double(get(hObject,'String')) returns contents of edit6 as a
double

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%         str2double(get(hObject,'String')) returns contents of edit7 as a
double

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.

```



```

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%         str2double(get(hObject,'String')) returns contents of edit8 as a
double

% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%         str2double(get(hObject,'String')) returns contents of edit9 as a
double

% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a
double

% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11 as a
double

% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text

```

```

%         str2double(get(hObject,'String')) returns contents of edit12 as a
double

% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit13 as text
%         str2double(get(hObject,'String')) returns contents of edit13 as a
double

% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit14 as text
%         str2double(get(hObject,'String')) returns contents of edit14 as a
double

% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)

```

```

% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit18_Callback(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit18 as text
%         str2double(get(hObject,'String')) returns contents of edit18 as a
double

% --- Executes during object creation, after setting all properties.
function edit18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit19_Callback(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit19 as text
%         str2double(get(hObject,'String')) returns contents of edit19 as a
double

% --- Executes during object creation, after setting all properties.
function edit19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.

```

```

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit20_Callback(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit20 as text
%         str2double(get(hObject,'String')) returns contents of edit20 as a
double

% --- Executes during object creation, after setting all properties.
function edit20_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit21_Callback(hObject, eventdata, handles)
% hObject    handle to edit21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit21 as text
%         str2double(get(hObject,'String')) returns contents of edit21 as a
double

% --- Executes during object creation, after setting all properties.
function edit21_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

function edit22_Callback(hObject, eventdata, handles)
% hObject    handle to edit22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit22 as text
%        str2double(get(hObject,'String')) returns contents of edit22 as a
double

% --- Executes during object creation, after setting all properties.
function edit22_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit38_Callback(hObject, eventdata, handles)
% hObject    handle to edit38 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit38 as text
%        str2double(get(hObject,'String')) returns contents of edit38 as a
double

% --- Executes during object creation, after setting all properties.
function edit38_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit38 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%-----

%-----
% --- Executes on button press in opencad_button.
function opencad_button_Callback(hObject, eventdata, handles)
% hObject    handle to opencad_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

```

```

% handles      structure with handles and user data (see GUIDATA)
!C:\Program Files\UGS\NX 7.5\UGII\ugraf.exe
%-----

%-----
% --- Executes on button press in opencfd.
function opencfd_Callback(hObject, eventdata, handles)
% hObject      handle to opencfd (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
!C:\Program Files\ANSYS Inc\v140\icemcfd\win\bin
!C:\Program Files\ANSYS Inc\v140\icemcfd\win\bin\icemcfd
%-----

%-----
% --- Executes on button press in pushbutton21.
function pushbutton21_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton21 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
%-----

%-----
% --- Executes when selected object is changed in unitspanel.
function unitspanel_SelectionChangeFcn(hObject, eventdata, handles)
% hObject      handle to the selected object in unitspanel
% eventdata    structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue:  handle of the previously selected object or empty if none was
selected
%   NewValue:  handle of the currently selected object
% handles      structure with handles and user data (see GUIDATA)

if handles.english
    set(handles.text12, 'String', 'uno')
else handles.metric
    set(handles.text12, 'String', 'dos')
end
%-----

%-----
function plot_ax2_Callback(hObject, eventdata, handles)
% hObject      handle to plot_ax2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Displays contents of axes1 at larger size in a new figure

% Create a figure to receive this axes' data
axes2fig = figure;
% Copy the axes and size it to the figure
axes2copy = copyobj(handles.axes2, axes2fig);
set(axes2copy, 'Units', 'Normalized', ...
    'Position', [.05, .20, .90, .60])
% Assemble a title for this new figure

```

```

% str = [get(handles.uipanel3,'Title') ' for ' ...
%       get(handles.poplable,'String')];
% title(str,'Fontweight','bold')
% Save handles to new fig and axes in case
% we want to do anything else to them
handles.axes2fig = axes2fig;
handles.axes2copy = axes2copy;
guidata(hObject,handles);
% -----
function plot_axes2_Callback(hObject, eventdata, handles)
% hObject    handle to plot_axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% -----

% -----
function plot_ax3_Callback(hObject, eventdata, handles)
% hObject    handle to plot_ax3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Displays contents of axes1 at larger size in a new figure

% Create a figure to receive this axes' data
axes3fig = figure;
% Copy the axes and size it to the figure
axes3copy = copyobj(handles.axes3,axes3fig);
set(axes3copy,'Units','Normalized',...
     'Position',[.05,.20,.90,.60])
% Assemble a title for this new figure
% str = [get(handles.uipanel3,'Title') ' for ' ...
%       get(handles.poplable,'String')];
% title(str,'Fontweight','bold')
% Save handles to new fig and axes in case
% we want to do anything else to them
handles.axes3fig = axes3fig;
handles.axes3copy = axes3copy;
guidata(hObject,handles);

% -----
function plot_axes3_Callback(hObject, eventdata, handles)
% hObject    handle to plot_axes3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% -----

% -----
function edit39_Callback(hObject, eventdata, handles)
% hObject    handle to edit39 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit39 as text
%       str2double(get(hObject,'String')) returns contents of edit39 as a
double

```



```

% --- Executes during object creation, after setting all properties.
function edit39_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit39 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit40_Callback(hObject, eventdata, handles)
% hObject    handle to edit40 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit40 as text
%         str2double(get(hObject,'String')) returns contents of edit40 as a
double

% --- Executes during object creation, after setting all properties.
function edit40_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit40 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit41_Callback(hObject, eventdata, handles)
% hObject    handle to edit41 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit41 as text
%         str2double(get(hObject,'String')) returns contents of edit41 as a
double

% --- Executes during object creation, after setting all properties.
function edit41_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit41 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))

```

```

        set(hObject, 'BackgroundColor', 'white');
end

function edit43_Callback(hObject, eventdata, handles)
% hObject    handle to edit43 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject, 'String') returns contents of edit43 as text
%        str2double(get(hObject, 'String')) returns contents of edit43 as a
double

% --- Executes during object creation, after setting all properties.
function edit43_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit43 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUiControlBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

function edit44_Callback(hObject, eventdata, handles)
% hObject    handle to edit44 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject, 'String') returns contents of edit44 as text
%        str2double(get(hObject, 'String')) returns contents of edit44 as a
double

% --- Executes during object creation, after setting all properties.
function edit44_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit44 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUiControlBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

function edit45_Callback(hObject, eventdata, handles)
% hObject    handle to edit45 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject, 'String') returns contents of edit45 as text
%        str2double(get(hObject, 'String')) returns contents of edit45 as a
double

```

```

% --- Executes during object creation, after setting all properties.
function edit45_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit45 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%-----

%-----
function edit46_Callback(hObject, eventdata, handles)
% hObject    handle to edit46 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit46 as text
%         str2double(get(hObject,'String')) returns contents of edit46 as a
double

% --- Executes during object creation, after setting all properties.
function edit46_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit46 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit47_Callback(hObject, eventdata, handles)
% hObject    handle to edit47 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit47 as text
%         str2double(get(hObject,'String')) returns contents of edit47 as a
double

% --- Executes during object creation, after setting all properties.
function edit47_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit47 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

```

```

%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit48_Callback(hObject, eventdata, handles)
% hObject    handle to edit48 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit48 as text
%       str2double(get(hObject,'String')) returns contents of edit48 as a
double

% --- Executes during object creation, after setting all properties.
function edit48_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit48 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit49_Callback(hObject, eventdata, handles)
% hObject    handle to edit49 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit49 as text
%       str2double(get(hObject,'String')) returns contents of edit49 as a
double

% --- Executes during object creation, after setting all properties.
function edit49_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit49 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit50_Callback(hObject, eventdata, handles)
% hObject    handle to edit50 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

% Hints: get(hObject,'String') returns contents of edit50 as text
%         str2double(get(hObject,'String')) returns contents of edit50 as a
double

% --- Executes during object creation, after setting all properties.
function edit50_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit50 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit51_Callback(hObject, eventdata, handles)
% hObject    handle to edit51 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit51 as text
%         str2double(get(hObject,'String')) returns contents of edit51 as a
double

% --- Executes during object creation, after setting all properties.
function edit51_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit51 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit52_Callback(hObject, eventdata, handles)
% hObject    handle to edit52 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit52 as text
%         str2double(get(hObject,'String')) returns contents of edit52 as a
double

% --- Executes during object creation, after setting all properties.
function edit52_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit52 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

```

```

%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit53_Callback(hObject, eventdata, handles)
% hObject    handle to edit53 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit53 as text
%       str2double(get(hObject,'String')) returns contents of edit53 as a
double

% --- Executes during object creation, after setting all properties.
function edit53_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit53 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit54_Callback(hObject, eventdata, handles)
% hObject    handle to edit54 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit54 as text
%       str2double(get(hObject,'String')) returns contents of edit54 as a
double

% --- Executes during object creation, after setting all properties.
function edit54_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit54 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit55_Callback(hObject, eventdata, handles)
% hObject    handle to edit55 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

% Hints: get(hObject,'String') returns contents of edit55 as text
%         str2double(get(hObject,'String')) returns contents of edit55 as a
double

% --- Executes during object creation, after setting all properties.
function edit55_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit55 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit56_Callback(hObject, eventdata, handles)
% hObject    handle to edit56 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit56 as text
%         str2double(get(hObject,'String')) returns contents of edit56 as a
double

% --- Executes during object creation, after setting all properties.
function edit56_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit56 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit57_Callback(hObject, eventdata, handles)
% hObject    handle to edit57 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit57 as text
%         str2double(get(hObject,'String')) returns contents of edit57 as a
double

% --- Executes during object creation, after setting all properties.
function edit57_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit57 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

```

```

% Hint: edit controls usually have a white background on Windows.
%     See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit58_Callback(hObject, eventdata, handles)
% hObject    handle to edit58 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit58 as text
%        str2double(get(hObject,'String')) returns contents of edit58 as a
double

% --- Executes during object creation, after setting all properties.
function edit58_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit58 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%     See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit59_Callback(hObject, eventdata, handles)
% hObject    handle to edit59 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit59 as text
%        str2double(get(hObject,'String')) returns contents of edit59 as a
double

% --- Executes during object creation, after setting all properties.
function edit59_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit59 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%     See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```



```

function edit60_Callback(hObject, eventdata, handles)
% hObject      handle to edit60 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit60 as text
%         str2double(get(hObject,'String')) returns contents of edit60 as a
double

% --- Executes during object creation, after setting all properties.
function edit60_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit60 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit62_Callback(hObject, eventdata, handles)
% hObject      handle to edit62 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit62 as text
%         str2double(get(hObject,'String')) returns contents of edit62 as a
double

% --- Executes during object creation, after setting all properties.
function edit62_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit62 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupunits.
function popupunits_Callback(hObject, eventdata, handles)
% hObject      handle to popupunits (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupunits
contents as cell array
%         contents{get(hObject,'Value')} returns selected item from popupunits

```

```

% --- Executes during object creation, after setting all properties.
function popupunits_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupunits (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupnozzle.
function popupnozzle_Callback(hObject, eventdata, handles)
% hObject    handle to popupnozzle (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupnozzle
contents as cell array
%         contents{get(hObject,'Value')} returns selected item from
popupnozzle

% --- Executes during object creation, after setting all properties.
function popupnozzle_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupnozzle (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupchamber.
function popupchamber_Callback(hObject, eventdata, handles)
% hObject    handle to popupchamber (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupchamber
contents as cell array
%         contents{get(hObject,'Value')} returns selected item from
popupchamber

% --- Executes during object creation, after setting all properties.
function popupchamber_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupchamber (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

```

```

% Hint: popupmenu controls usually have a white background on Windows.
%     See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupinjector.
function popupinjector_Callback(hObject, eventdata, handles)
% hObject     handle to popupinjector (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupinjector
contents as cell array
%     contents{get(hObject,'Value')} returns selected item from
popupinjector

% --- Executes during object creation, after setting all properties.
function popupinjector_CreateFcn(hObject, eventdata, handles)
% hObject     handle to popupinjector (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%     See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton25.
function pushbutton25_Callback(hObject, eventdata, handles)
% hObject     handle to pushbutton25 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton26.
function pushbutton26_Callback(hObject, eventdata, handles)
% hObject     handle to pushbutton26 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton27.
function pushbutton27_Callback(hObject, eventdata, handles)
% hObject     handle to pushbutton27 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

```

```

function edit71_Callback(hObject, eventdata, handles)
% hObject      handle to edit71 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit71 as text
%          str2double(get(hObject,'String')) returns contents of edit71 as a
double

% --- Executes during object creation, after setting all properties.
function edit71_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit71 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%          See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit72_Callback(hObject, eventdata, handles)
% hObject      handle to edit72 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit72 as text
%          str2double(get(hObject,'String')) returns contents of edit72 as a
double

% --- Executes during object creation, after setting all properties.
function edit72_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit72 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%          See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit73_Callback(hObject, eventdata, handles)
% hObject      handle to edit73 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

```

```
% Hints: get(hObject,'String') returns contents of edit73 as text
%         str2double(get(hObject,'String')) returns contents of edit73 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit73_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit73 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit74_Callback(hObject, eventdata, handles)
% hObject    handle to edit74 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit74 as text
%         str2double(get(hObject,'String')) returns contents of edit74 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit74_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit74 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit67_Callback(hObject, eventdata, handles)
% hObject    handle to edit67 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit67 as text
%         str2double(get(hObject,'String')) returns contents of edit67 as a
double
```

```

% --- Executes during object creation, after setting all properties.
function edit67_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit67 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit68_Callback(hObject, eventdata, handles)
% hObject    handle to edit68 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit68 as text
%         str2double(get(hObject,'String')) returns contents of edit68 as a
double

% --- Executes during object creation, after setting all properties.
function edit68_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit68 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit69_Callback(hObject, eventdata, handles)
% hObject    handle to edit69 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit69 as text
%         str2double(get(hObject,'String')) returns contents of edit69 as a
double

% --- Executes during object creation, after setting all properties.
function edit69_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit69 (see GCBO)

```

```

% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

function edit70_Callback(hObject, eventdata, handles)
% hObject handle to edit70 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit70 as text
% str2double(get(hObject,'String')) returns contents of edit70 as a
double

```

```

% --- Executes during object creation, after setting all properties.
function edit70_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit70 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

function edit63_Callback(hObject, eventdata, handles)
% hObject handle to edit63 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit63 as text
% str2double(get(hObject,'String')) returns contents of edit63 as a
double

```

```

% --- Executes during object creation, after setting all properties.
function edit63_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit63 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

```

```

%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit64_Callback(hObject, eventdata, handles)
% hObject    handle to edit64 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit64 as text
%       str2double(get(hObject,'String')) returns contents of edit64 as a
double

% --- Executes during object creation, after setting all properties.
function edit64_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit64 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit65_Callback(hObject, eventdata, handles)
% hObject    handle to edit65 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit65 as text
%       str2double(get(hObject,'String')) returns contents of edit65 as a
double

% --- Executes during object creation, after setting all properties.
function edit65_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit65 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```


end

```
function edit66_Callback(hObject, eventdata, handles)
% hObject    handle to edit66 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit66 as text
%        str2double(get(hObject,'String')) returns contents of edit66 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit66_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit66 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in exitbutton.
function exitbutton_Callback(hObject, eventdata, handles)
% hObject    handle to exitbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% % Get the current position of the GUI from the handles structure
% % to pass to the modal dialog.
pos_size = get(handles.figure1,'Position');
% % Call modaldlg with the argument 'Position'.
user_response = modaldlg('Title','Confirm Close');
switch user_response
case {'No'}
    % take no action
case 'Yes'
    % Prepare to close GUI application window
    %
    %
    %
    delete(handles.figure1)
end
```

Appendix B: Function for Rao Nozzle

Filename: Rao.m

```
function [Rao_x, Rao_y, x_conver, y_conver, x_cyl, y_cyl, Ln, theta_exit, nf] =  
Rao(per, Rt, area_ratio, theta, P3, F, tp, L_star, r, den_f, den_o, mol, gamma, P1, T1, Cd, P  
d, Df)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%INPUT VARIABLES%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% per = 0.8;  
% Rt = 5.15;  
% area_ratio = 77.5;  
% theta = 40; %deg  
% F = 35000; %lbf  
% P1 = 2000; %psia  
% r = 2.24;  
% mol = 21.9;  
% den_o = 1.14*62.42;  
% den_f = 0.58*62.42;  
% T1 = 3400; %Rankine  
% gamma = 1.24;  
% P3 = 3.844; %psia  
% tp = 2; %minutes  
% L_star = 118;  
% Cd = 0.8  
% Pd = 0.2 %  
% Df = .035 %ft
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Set Values%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
g_c = 32.174;  
R_dash = 1554; %lbf-ft/lb-mol-R
```

```
v_correction = 0.97;  
CF_correction = 0.90;  
theta_conver = 45;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Calculations%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
Ln = per*(sqrt(area_ratio)-1)*Rt/tand(15);  
theta_n = theta*pi/180; %rad  
tp = tp*60;  
% P2 = P3;  
den_o = den_o*62.42;  
den_f = den_f*62.42;  
-----  
v2_i=sqrt(((2*gamma)/(gamma-1))*(R_dash/mol)*T1*(1-(P3/P1)^((gamma-  
1)/gamma)));  
v2_a=v2_i*v_correction;  
c=v2_a;  
I_sp_i=v2_i/g_c;  
I_sp_a=v2_a/g_c;  
m_dot=F/c;  
w_dot=m_dot*g_c;
```

```

%-----
CF_i=sqrt(((2*gamma^2)/(gamma-1))*(2/(gamma+1))^( (gamma+1)/(gamma-1) )*(1-
(P3/P1)^( (gamma-1)/gamma)));
CF_a=CF_i*CF_correction;
%-----
At=F/(CF_a*P1);
A2=area_ratio*At;
%-----
Dt=sqrt((4*At)/pi);
D2=sqrt((4*A2)/pi);
D1=2*Dt; % Diameter of the cylindrical part of combustion chamber
R2=D2/2;
%-----
w_dot_f=w_dot/(r+1);
w_dot_o=(w_dot*r)/(r+1);
V_dot_f=w_dot_f/(den_f);
V_dot_o=w_dot_o/(den_o);
V_dot=V_dot_o+V_dot_f;
%-----
w_f=w_dot_f*tp;
w_o=w_dot_o*tp;
w_p=w_dot*tp;
V_o=V_dot_o*tp;
V_f=V_dot_f*tp;
V_p=V_dot*tp;
%-----
V_c=L_star*At; % Total Volume of the combustion chamber (Cylinder+Conical)
L_conver=(D1-Dt)/(2*(tan(theta_conver))); %Length of convergent part (not the
slant height, but straight one).
V_conver=(1/3)*pi*L_conver*((Dt/2)^2+(D1/2)*(Dt/2)+D1^2); %Volume of the
convergent part
V_cyl=V_c-V_conver; % Volume of cylindrical part
L1=(V_cyl-V_conver)/((pi/4)*D1^2); % Length of cylindrical part
%-----
%Injector Sizing
del_P = Pd*P1; %pressure drop from injector to chamber in psi
pf = den_f/g_c; %fuel density in slugs/ft^3
po = den_o/g_c; %oxidizer density in slugs/ft^3

% Do = Df;
Af=sqrt(pf/(2*del_P*144))*V_dot_f/Cd; %fuel total area
Ao=sqrt(po/(2*del_P*144))*V_dot_o/Cd; %oxidizer area

% nf=Af/((pi/4)*Df^2) %number of fuel holes
% no=Ao/((pi/4)*Do^2) %number of oxidizer holes

% cd=0.9; %from Table 8-2
%total area

%mm
%area of each fuel injector
fia=(pi/4)*Df^2;
%number of fuel injectors
nf=round(Af/fia); %rounded number based on previous fin
% fia=Af/fin; %sq.ft.
% fid=sqrt(fia*(4/pi));

```

```

%      %oxidizer ijector amount equals fuel injector amount
%      oin=fin;
%      oia=Ao/oin; %sq.ft. oxidizer injector area
%      oid=sqrt(oia*(4/pi));
% %angles
% aod=30; %angle of oxidizer degress
% afd=25; %angle of fuel degress
% ao=aod*pi/180; af=afd*pi/180; %converting degrees to rads
% vf=Vdf/Af; %fuel velocity ft/s
% vo=Vdo/Ao; %ox ft/s
% delta=atan( ((wdo/g0)*vo*sin(ao)-(wdf/g0)*vf*sin(af))/...
%      ((wdo/g0)*vo*cos(ao)+(wdf/g0)*vf*cos(af)));
% Delta=delta*180/pi;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Ploting Calculations

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%FIRST CURVE (FC)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Angles for First Curve
Angle_FC=-3*pi/4;
FC_step=(-pi/2-Angle_FC)/9;
theta_FC=(-3*pi/4):FC_step:(-pi/2);

% Coordinates for First Curve
x_FC=cos(theta_FC)*1.5*Rt;
y_FC=sin(theta_FC)*1.5*Rt+(1.5*Rt+Rt);
x_FC1 = x_FC';
y_FC1 = y_FC';

%-----
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%SECOND CURVE (SC)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Angle for Second Curve
Angle_SC=-pi/2;
SC_step=((theta_n-pi/2)-Angle_SC)/4;
theta_SC=-pi/2:SC_step:(theta_n-pi/2);

% Coordinates for Second Curve
x_SC=cos(theta_SC)*0.382*Rt;
y_SC=sin(theta_SC)*0.382*Rt+(0.382*Rt+Rt);
x_SC1 = x_SC';
y_SC1 = y_SC';

%-----
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%THIRD CURVE (TC)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Angle for Third Curve
x_TC=cos(theta_n-pi/2)*0.382*Rt;
y_TC=sin(theta_n-pi/2)*0.382*Rt+(0.382*Rt+Rt);

y_exit=sqrt(area_ratio)*Rt;

matrix_y=[y_TC^2 y_TC 1; y_exit^2 y_exit 1; 2*y_TC 1 0];
matrix_x=[x_TC; Ln ;1/tan(theta_n)];
x_exit=matrix_y^-1*matrix_x;
a=x_exit(1,1);
b=x_exit(2,1);
c=x_exit(3,1);

```

```

y=y_TC:.1:y_exit;
x=a*y.^2+b*y+c;

% Coordinate of Third Curve
Rao_x=[x_FC x_SC x];
Rao_y=[y_FC y_SC y];
rao=[Rao_x Rao_y];

% plot(x_FC, y_FC,'r', x_SC, y_SC,'r', Rao_x, Rao_y,'r');

%-----
%%%%%%%%%%Chamber Convergent Curve (Conver)%%%%%%%%%%
%Initial starting point
x_ct = x_FC(1,1);
y_ct = y_FC(1,1);
x_conver1=x_ct-L_conver;
y_conver1=((D1/2)-y_ct)/(-L_conver-x_ct)*(x_conver1-x_ct)+y_ct;
x_conver = [x_conver1 x_ct];
y_conver = [y_conver1 y_ct];

% x_conver=-L_conver:1:x_ct;
% x_conver = x_conver';
% for i=1:1:size(x_conver,1)
%     y_conver(i) (((D1/2)-y_ct)/(-L_conver-x_ct))*(x_conver(i)-x_ct)+y_ct;
% end
% y_conver=(y_conver)';
%-----
%%%%%%%%%%Chamber Cylindrical Curve (cyl)%%%%%%%%%%
x_cy = x_conver(1,1);
y_cy = y_conver(1,1);
x_cyl=-(L1+L_conver)-x_cy;
x_cyl= [x_cyl x_cy];
y_cyl=y_conver(1,1);
y_cyl=[y_cyl y_cy];

% plot(x_cyl,y_cyl,'g',x_cyl,-y_cyl,'g',x_conver, y_conver,'r-',x_conver,-
y_conver,'r-',Rao_x,Rao_y,'k',Rao_x,-Rao_y,'k')

% Dth = sqrt(At/(pi/4));
% Dc = Dth*2;
% ---CHAMBER---%
% %chamber cone line
% c_slope=tan(pi-45*pi/180); %slope
% cN=[x_FC(1),y_FC(1)]; %starting point
% i=1;cy(i)=cN(2);cx(i)=cN(1);
% while cy(i)<Dc/2
%     i=i+1; cx(i)=cx(i-1)-0.1;
%     cy(i)=c_slope*(cx(i)-cN(1))+cN(2);
% end
% cx=flipdim(cx,2); cy=flipdim(cy,2);
%
% %Chamber Volume and L*
% Ac=pi*Rc^2;
% Vc=L_star*At;

```

```
% Lc=(Vc-(1/3)*pi*(-cx(1))*((2*Rt)^2+2*Rt*Rt+Rt^2))/Ac;
%
% chamber body line
% fcx=[cx(1)-Lc, cx(1)-Lc, cx(1)];
% fcy=[0,Dc/2,Dc/2];
% l_long=abs(cx(1)-Lc);

theta_exit = atan((2.*a*Ln)+b);
% end
```

Appendix C: function conical

Filename: Conical.m

```
function [Cx_FC, Cy_FC, Cx_CSC, Cy_CSC, Rao_Cx, Rao_Cy, CLn, Ctheta_exit] =
Conical(CRt, Carea_ratio, Ctheta)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Conical Nozzle %%%%%%%%%%
%CRt=1;
%Carea_ratio=77.5;
CLn=(sqrt(Carea_ratio)-1)*CRt/tand(15);
%Ctheta=15; %deg
Ctheta_n=Ctheta*pi/180; %rad

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%FIRST CURVE (CFC) %%%%%%%%%%
%Angles for First Curve
CAngle_FC=-3*pi/4;
CFC_step=((Ctheta_n-pi/2)-CAngle_FC)/4;
Ctheta_FC=(-3*pi/4):CFC_step:(Ctheta_n-pi/2);

%Coordinates for First Curve
Cx_FC=cos(Ctheta_FC)*1.5*CRt;
Cy_FC=sin(Ctheta_FC)*1.5*CRt+(1.5*CRt+CRt);
Cy_FCn=sin(Ctheta_n-pi/2)*1.5*CRt+(1.5*CRt+CRt);
Cx_FCn=cos(Ctheta_n-pi/2)*1.5*CRt;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%SECOND CURVE (CSC) %%%%%%%%%%
%Coordinates for Second Curve
CSCa=tan(Ctheta_n);
CSCb=Cy_FCn-CSCa*Cx_FCn;

step_CSC=(CLn-Cx_FCn)/9;
Cx_CSC=Cx_FCn:step_CSC:CLn;
Cy_CSC=CSCa*Cx_CSC+CSCb;

%Coordinates of Third Curve
Rao_Cx=[Cx_FC Cx_CSC];
Rao_Cy=[Cy_FC Cy_CSC];
rao=[Rao_Cx Rao_Cy];
%plot(Cx_FC, Cy_FC, 'b', Cx_CSC, Cy_CSC, 'b', Rao_Cx, Rao_Cy, 'b');
Ctheta_exit = atand((2*CSCa*CLn)+CSCb);

end
```

Appendix D: function MOC

Filename: MOC_code.m

```
function [Ae,TT,A_max,Max_thrust,noz,i,c1,c2,aa,cc] =
MOC_code(T_c,P_c,width,num)

Based on open source code

%find where P becomes u
h(1) = h_th;
A_star = h_th*width;
M =1;
dM1 = .1;
for i=1: max_iter
    h(i) = h(1) + (i-1)*dh;
    Ae(i) = h(i)*width;
    A_Asq = (Ae(i)/A_star)^2;
    A_ratio(i)=sqrt(A_Asq);

    %Newton Rhapson on Eq. 5.20 - Anderson text
    res = 1;
    if i > 1
        M = Ma(i-1);
    end

    while res > .001
        M2 = M + dM1;
        funa1 = -A_Asq + (1/M^2)*((2/(gamma+1))*(1+(gamma-
1)*M^2/2))^( (gamma+1)/(gamma-1));
        funa2 = -A_Asq + (1/M2^2)*((2/(gamma+1))*(1+(gamma-
1)*M2^2/2))^( (gamma+1)/(gamma-1));
        dv_dm = (funa2-funa1)/dM1;

        M = M - funa1/dv_dm;
        res = abs(funa1);

    end
    Ma(i) = M;

    % Find Pressure
    P(i) = P_c*(1+(gamma-1)*Ma(i)^2/2)^(-gamma/(gamma-1));

    % Find thrust for each point
    Te(i) = T_c/(1+(gamma-1)*Ma(i)^2/2);
    Tt(i) = T_c/(1+(gamma-1)/2);
    Ve(i) = Ma(i)*sqrt(Te(i)*gamma*R);
    Vt(i) = sqrt(Tt(i)*gamma*R);
    rhot(i) = P(i)/(R*Te(i));
    mdot(i) = rhot(i)*Ve(i)*Ae(i);
    TT(i) = mdot(i)*Ve(i) + (P(i) - P_amb)*Ae(i);

    if P(i) < P_amb
```



```

%Calculate the pressure if shock wave exists at the exit plane
P_exit = P(i)*(1+(gamma*2/(gamma+1))*(Ma(i)^2-1));

    if P_exit <= P_amb
        P(i) = P_exit;
        break
    else
    end

else
end

end

[a,b]=max(TT);
% Over or Underexpand the nozzle
b = b;
A_max = Ae(b);
Max_thrust = TT(b);
hold on;
% plot(A_max,Max_thrust,'r*')
% legend('Thrust Curve','Max Thrust')

M_e = Ma(b);          %Mach number at ideal exit

%Find theta_max by using equation 11.33
theta_max = (180/pi)*(sqrt((gamma+1)/(gamma-1))*atan((sqrt((gamma-1)*(M_e^2-1)/(gamma+1)))-atan(sqrt(M_e^2-1)))/2;

% D_theta for each char line
del_theta = (theta_max - theta_i)/(num-1);

% Find

for i=1:num
    % Initialize mach numeber

    for j=1:num
        if i==1
            %Theta for each line (first lines)
            theta(i,j) = theta_i + del_theta*(j-1);
            nu(i,j) = theta(i,j);
            K_m(i,j) = theta(i,j) + nu(i,j);
            K_p(i,j) = theta(i,j) - nu(i,j);

        elseif i > 1

            K_p(i,j) = -K_m(1,i);

            % Find Thetas
            if j >= i
                theta(i,j) = del_theta*(j-i);
            else

```

```

        %theta(i,j) = theta(j,i-1);
        theta(i,j) = theta(j,i);

    end
    nu(i,j) = theta(i,j) - K_p(i,j);
    K_m(i,j) = theta(i,j) + nu(i,j);
end

% Prandtl-Meyer function (using Newton Rhapson)
dM = .1; % Leave at about .1
if j == 1
    M_ex(i,j) = 1.00;
else
    M_ex(i,j) = M_ex(i,j-1);
end
M = M_ex(i,j);

res = 1;
while res > .01
    M2 = M + dM;
    funv1 = (-nu(i,j)*(pi/180)+(sqrt((gamma+1)/(gamma-
1))*atan((sqrt((gamma-1)*(M^2-1)/(gamma+1))))-atan(sqrt(M^2-1))));
    funv2 = (-nu(i,j)*(pi/180)+(sqrt((gamma+1)/(gamma-
1))*atan((sqrt((gamma-1)*(M2^2-1)/(gamma+1))))-atan(sqrt(M2^2-1))));
    dv_dm = (funv2-funv1)/dM;

    M = M - funv1/dv_dm;
    res = abs(funv1);

end
M_ex(i,j) = M;

% Find the angle mu
mu(i,j) = (180/pi)*asin(1/M_ex(i,j));

end

% Add last point to char line
theta(i,num+1) = theta(i,num);
nu(i,num+1) = nu(i,num);
K_m(i,num+1) = K_m(i,num);
K_p(i,num+1) = K_p(i,num);
end

char = zeros(num,num+1,2);
for i=1:num

    for j=1:num+1

% Draw points of intersection
% Point 1 of all char lines
if j == 1
    char(i,j,1) = 0;
    char(i,j,2) = h_th/2;
end

```

```

% Where first line hits the symmetry line
if i == 1 & j==2
    char(i,j,1) = (-h_th/2)/tan((pi/180)*(theta(1,j-1)-mu(1,j-1)));
    char(i,j,2) = 0;
end

% Where all other lines hit the symmetry line
if j == i+1 & j>2
    char(i,j,1) = -char(i-1,j,2)/tan((pi/180)*(.5*theta(i,j-2)-
.5*(mu(i,j-2)+mu(i,j-1)))) + char(i-1,j,1);
    char(i,j,2) = 0;
    test(i,j) = (theta(i,j-2)-.5*(mu(i,j-2)+mu(i,j-1)));
    testpty(i,j) = char(i-1,j,2);
    testptx(i,j) = char(i-1,j,1);
end

% All other data points for char 1 calculated
if i ==1 & j>2 & j ~= i+1
    C_p = tan((pi/180)*(.5*(theta(i,j-2)+theta(i,j-1))+.5*(mu(i,j-
2)+mu(i,j-1)))));
    C_m = tan((pi/180)*(.5*(theta(j-1,1)+theta(i,j-1))- .5*(mu(j-
1,1)+mu(i,j-1)))));
    A = [1,-C_m;1,-C_p];
    B = [char(1,1,2) - char(1,1,1)*C_m;
char(1,j-1,2) - char(1,j-1,1)*C_p];
    iterm(1,:) = inv(A)*B;
    char(i,j,1) = iterm(1,2);
    char(i,j,2) = iterm(1,1);
end

% All other points for all char lines calculated
if i > 1 & j~=i+1 & j>2
    C_p = tan((pi/180)*(.5*(theta(i,j-2)+theta(i,j-1))+.5*(mu(i,j-
2)+mu(i,j-1)))));
    C_m = tan((pi/180)*(.5*(theta(i-1,j-1)+theta(i,j-1))- .5*(mu(i-
1,j-1)+mu(i,j-1)))));
    A = [1,-C_m;1,-C_p];
    B = [char(i-1,j,2) - char(i-1,j,1)*C_m; char(i,j-1,2) - char(i,j-
1,1)*C_p];

    iterm(1,:) = inv(A)*B;
    char(i,j,1) = iterm(1,2);
    char(i,j,2) = iterm(1,1);
end
end
end

% Fill in similar points (where char lines share points)
for i = 2:num
    for j=2:num
        char(j,i,1) = char(i-1,j+1,1);
        char(j,i,2) = char(i-1,j+1,2);
    end
end

```

```

end

% *****Make the nozzle shape and extend the char lines to wall*****

% Initial start point of the nozzle (at throat)
noz(1,1) = 0;
noz(1,2) = h_th/2;

% Find all the points of the nozzle
for i = 2 : num
    % Find different slopes and points to intersect
    m1 = tan((pi/180)*(theta(i-1,num)+mu(i-1,num)));
    if i ==2
        m2 = (pi/180)*theta_max;
    else
        m2 = ((pi/180)*(theta(i-1,num+1)));
    end
    m3 = ((pi/180)*(theta(i-1,num)));
    m4 = tan((m2+m3)/2);

    A = [1,-m4; 1,-m1];
    B = [noz(i-1,2) - noz(i-1,1)*m4; char(i-1,num+1,2) - char(i-
1,num+1,1)*m1];

    iterm(1,:) = inv(A)*B;
    noz(i,1) = iterm(1,2);
    noz(i,2) = iterm(1,1);

    % Extend char lines to wall
    char(i-1,num+2,1)= noz(i,1);
    char(i-1,num+2,2)= noz(i,2);
end

%Last line
m1 = tan((pi/180)*(theta(num,num)+ mu(num,num)));
m2 = ((pi/180)*(theta(num-1,num)));
m3 = ((pi/180)*(theta(num,num+1)));
m4 = tan((m2+m3)/2);

A = [1,-m4; 1,-m1];
B = [noz(num,2) - noz(num,1)*m4; char(num,num+1,2) - char(num,num+1,1)*m1];

iterm(1,:) = inv(A)*B;
noz(num+1,1) = iterm(1,2);
noz(num+1,2) = iterm(1,1);

% Extend char lines to wall
char(num,num+2,1)= noz(num+1,1);
char(num,num+2,2)= noz(num+1,2);

if plotter ==1

```

```

[a,b] = max(noz);
cc = A_max/width/2;
aa=a(1);

c1 = char(i, :, 1)
c2 = char(i, :, 2)
end
% Find % errors in A/A* and Mexit
error_Area = 100*(width*2*noz(num,2) - A_max)/(A_max);
error_Mach = 100*(M_e - M_ex(num,num))/M_e;

M = Mnoz(1);
for i=1: size(noz,1)
    Ae(i) = 2*noz(i,2)*width;
    A_Asq = (Ae(i)/A_star)^2;
    A_ratio(i)=sqrt(A_Asq);

    %Newton Rhapson on Eq. 5.20 - Anderson text
    res = 1;
    if i > 1
        M = Mnoz(i-1);

        while res > .001
            M2 = M + dM1;
            funa1 = -A_Asq + (1/M^2)*((2/(gamma+1))*(1+(gamma-
1)*M^2/2))^( (gamma+1)/(gamma-1));
            funa2 = -A_Asq + (1/M2^2)*((2/(gamma+1))*(1+(gamma-
1)*M2^2/2))^( (gamma+1)/(gamma-1));
            dv_dm = (funa2-funa1)/dM1;

            M = M - funa1/dv_dm;
            res = abs(funa1);

        end
        Mnoz(i) = M;
    end
    % Find Pressure
    Pnoz(i) = P_c*(1+(gamma-1)*Mnoz(i)^2/2)^(-gamma/(gamma-1));
end
end

```

Appendix F: fuel

Filename: Fuel.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Fuel Selection
%Table 5-5: Theoretical Performance of Liquid Rocket Propellant
%Combinations
%TABLE 7-1: SOME PHYSICAL PROPERTIES OF SEVERAL COMMON LIQUID PROPELLANTS
%Book: Rocket Propulsion Elements by George P. Sutton and Oscar Biblarz
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Notes (Table 5-5)
%Combustion chamber pressure--1000 psia (6895 kN.m^2)
%Nozzle exit pressure--14.7 psia (1 atm)
%Optimum expansion
%Adiabatic combustion and isentropic expansion of ideal gas
%The specific gravity at the boiling point was used for those oxidizers or
%fuels that boil below 20C at 1 atm pressure.
%Mixture ratios are for approx. maximum value of Is.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Notes (Table 7-1)
%a: Red fuming nitric acid (RFNA) has 5% to 20% dissolved NO2 with an avg.
%molecular weight of about 60, and a density and vapor pressure somewhat
%higher than those of pure nitric acid.
%b: At boiling point.
%c: Reference for specific gravity ratio: 10^3 kg/m^3 or 62.42 lbm/ft^3

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

methanefuel = {3.20, 3.00; %Mixture Ratio by mass
              1.19, 1.11; %Mixture Ratio by volume
              0.424,0.424; %Specific gravity methane
              1.14, 1.23; %Specific Gravity oxygen
              3526, 3526; %Chamber temperature (K)
              1835, 1853; %Chamber c*
              16.03,16.03; %Molecular mass MM
              0, 0;}; %k

hydrazinefuel = {0.74, 0.90; %Mixture Ratio by mass
                0.66, 0.80; %Mixture Ratio by volume
                1.005, 0.952;%Specific gravity hydrazine
                1.14, 1.23; %Specific Gravity oxygen
                3285, 3404; %Chamber temperature (K)
                1871, 1892; %Chamber c*
                18.3, 19.3; %Molecular mass MM
                1.25, 1.25;}; %k

hydrogenfuel = { 3.40, 4.02; %Mixture Ratio by mass
                0.21, 0.25; %Mixture Ratio by volume
                0.071,0.076;%Specific gravity hydrogen
```

```

1.14, 1.23; %Specific Gravity oxygen
2959, 2999; %Chamber temperature (K)
2428, 2432; %Chamber c*
8.9, 10.0; %Molecular mass MM
1.26, 1.26;}; %k

RP1fuel = { 2.24, 2.56; %Mixture Ratio by mass
1.59, 1.82; %Mixture Ratio by volume
0.58,0.807;%Specific gravity RP-1
1.14, 1.23; %Specific Gravity oxygen
3571, 3677; %Chamber temperature (K)
1774, 1800; %Chamber c*
21.9, 23.3; %Molecular mass MM
1.24, 1.24;}; %k

UDMHfuel = { 1.39, 1.65; %Mixture Ratio by mass
0.96, 1.14; %Mixture Ratio by volume
0.856,0.784;%Specific gravity UDMH
1.14, 1.23; %Specific Gravity oxygen
3542, 3594; %Chamber temperature (K)
1835, 1864; %Chamber c*
19.8, 21.3; %Molecular mass MM
1.25, 1.25;}; %k

flu_hydrafuel = { 1.83, 2.30; %Mixture Ratio by mass
1.22, 1.54; %Mixture Ratio by volume
1.005,0.952;%Specific gravity hydrazine
1.636, 1.44; %Specific Gravity fluorine
4553, 4713; %Chamber temperature (K)
2128, 2208; %Chamber c*
18.5, 19.4; %Molecular mass MM
1.33, 1.33;}; %k

flu_hydrofuel = { 4.54, 7.60; %Mixture Ratio by mass
0.21, 0.35; %Mixture Ratio by volume
0.071,0.076;%Specific gravity hydrogen
1.636, 1.44; %Specific Gravity fluorine
3081, 3900; %Chamber temperature (K)
2534, 2549; %Chamber c*
8.9, 11.8; %Molecular mass MM
1.33, 1.33;}; %k

Nitrotetro_hydrafuel = { 1.08, 1.34; %Mixture Ratio by mass
0.75, 0.93; %Mixture Ratio by volume
1.005,0.952;%Specific gravity hydrazine
1.447, 1.38; %Specific Gravity Nitrogen Tetroxide
3258, 3152; %Chamber temperature (K)
1765, 1782; %Chamber c*
19.5, 20.9; %Molecular mass MM
1.26, 1.26;}; %k

Nitrotetro_RP1fuel = { 3.4, 0; %Mixture Ratio by mass
1.05, 0 ; %Mixture Ratio by volume
1.005,0.952;%Specific gravity RP-1
1.447, 1.38; %Specific Gravity Nitrogen Tetroxide
3290, 0 ; %Chamber temperature (K)

```

```

        0 ,0 ; %Chamber c*
        24.1, 0 ; %Molecular mass MM
        1.23, 0 ;}; %k

Nitrotetro_MMHfuel ={ 2.15, 1.65; %Mixture Ratio by mass
        1.30, 1.00; %Mixture Ratio by volume
        0.8788,0.857;%Specific gravity MMH (Monomethyl-
hydrazine)
        1.447, 1.38; %Specific Gravity Nitrogen Tetroxide
        3396, 3200; %Chamber temperature (K)
        1747, 1591; %Chamber c*
        22.3, 21.7; %Molecular mass MM
        1.23, 1.23;}; %k

```