Nozzle Flow Analyzer Project

Software Design Document

Logan White

2/24/2023

CS 225, Spring 2023

Embry-Riddle Aeronautical University

Daytona Beach campus

1 Aerospace Boulevard

Daytona Beach, FL 32114

INTRODUCTION:

This Software Design Document (SDD) describes the functionality and software design of a Nozzle Flow analyzing system (NFA). In this system a flow path can be built from a list of set components, and the user can input data for calculating the properties of the flow. The user selects from a set number of conditions called flow nodes, referred to as such because each node can act independently of each other, with the exit conditions of one node acting as the inlet conditions of the next. For the system as a whole, isentropic flow relations are used with the exception of if a shock is detected. In essence this will be a basic CFD, computational fluid dynamics calculator.

The NFA project is going to be created as a project for Embry-Riddle Aeronautical University's (ERAU) Computer Science II course (CS225). This project contains information gained in the ERAU Compressible Flow class (AE308). This project contains all the features expected of a student project including, file input and output, exception handling, use of inheritance, and will also include GUI generation, and interaction.

The following documentation is composed of the following sections:

Flow Pieces

Graphical Display to the user.

Requirements

Implementation.

Flow Pieces

Figure 1

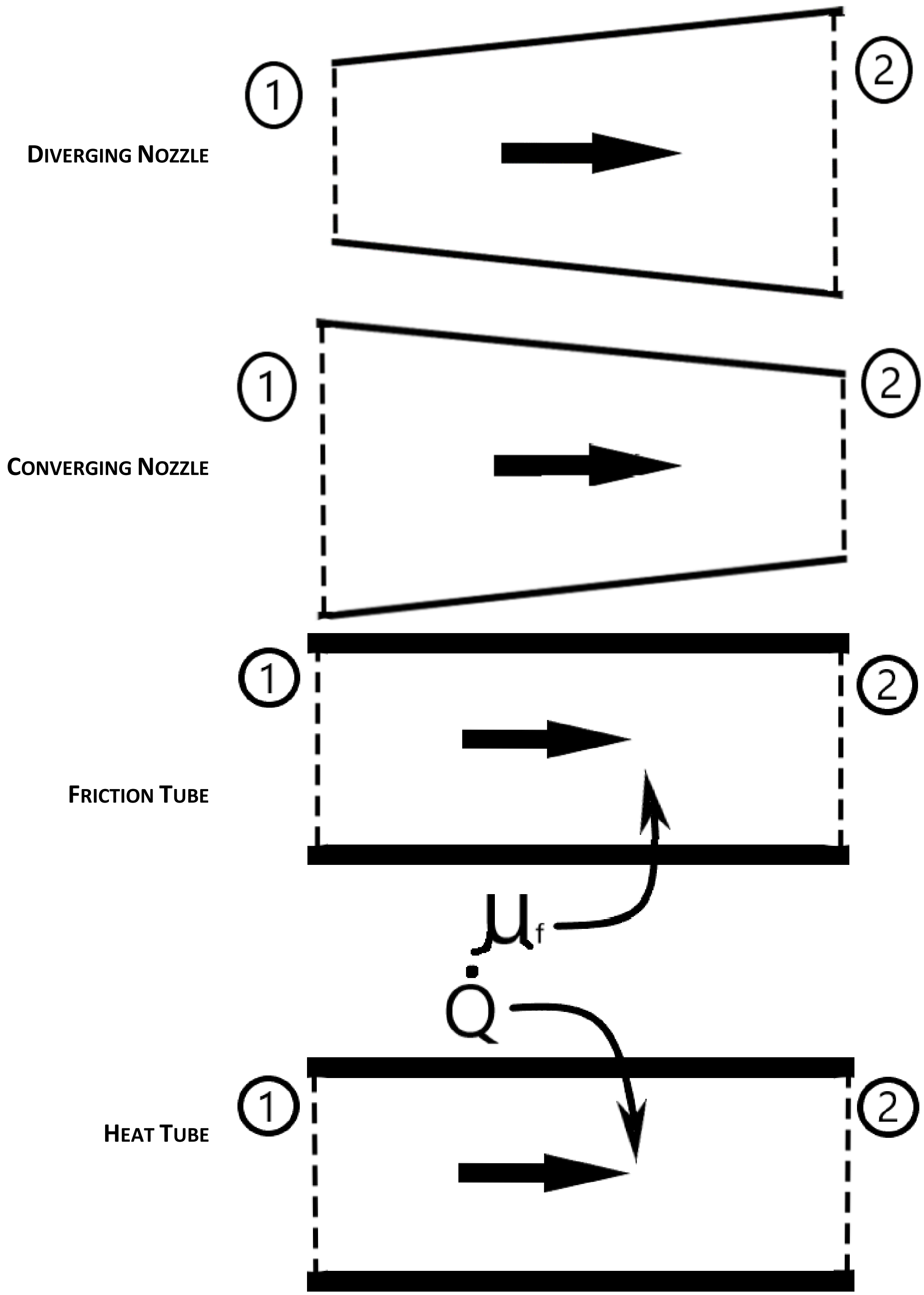


Figure 1 displays the four types of flow nodes that will be implemented. For reference the arrows point to the direction of flow, and the circled numbers next to the dotted lines refer to the location where data will be collected.

Depending on what pieces are used by the user also changes what inputs they will have to provide. For both the converging and diverging nozzle those will require, if first in the node setup, and stagnation pressure and temperature at the region one area, both will also need an area for the points at 1 and 2, initial velocity or Mach number is also required`. Converging nozzles speed up flow that is under Mach 1 but increase flow over a Mach speed of 1, whist Diverging nozzles do the exact opposite. Both Diverging and Converging nozzle use the same equation, and mathematically converging nozzles are treated as backwards diverging nozzles.

For the Tubes similar inputs are needed with the user needing to provide a stagnation pressure and temperature at the 1st point, the tubes are constant area so the user will only have to provide one area for the tube. If it is a heat tube the Heat input (Qdot) will need to be provided, and if it is a friction tube the friction constant (Mu_f) will have to be provided. Each piece has different required values

Each of the Flow Pieces will be handled as an Extension of the Generic Nozzle.

Flow Equations:

A set of flow equations are required for the calculations, these equations are derived in, (Farokhi, 142), all of these equations come from the conservation equations of mass momentum, energy and entropy,

For any point in the flow the relation of:

$$\frac{P_2}{P_1} = \left(\frac{\rho_2}{\rho_1}\right)^{\gamma} = \left(\frac{T_2}{T_1}\right)^{\frac{\gamma}{\gamma-1}}$$

Is true, and allows for ratios through any piece,
Anywhere where the flow is determined to be isentropic Stagnation pressure and temperature are constant, which allows for relations to be used through the piece, in heat and friction tubes, they are never considered isentropic.

$$\frac{T_o}{T} = \left(1 + \frac{\gamma - 1}{2} M^2\right) \quad \frac{\rho_o}{\rho} = \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{1}{\gamma-1}} \quad \frac{P_o}{P} = \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma-1}}$$

When working through the nozzles, these relations are just a function of initial stagnation values, and Mach number, since static pressure is constant the equation can be combined to provide different pressure values at different points.

The relationships over a shock are based on initial Mach number and the conservation of entropy, (Farokhi, 155).

$$M_2^2 = \frac{1 + [(\gamma - 1)/2]M_1^2}{\gamma M_1^2 - (\gamma - 1)/2} \qquad \frac{\rho_2}{\rho_1} = \frac{u_1}{u_2} = \frac{(\gamma + 1)M_1^2}{2 + (\gamma - 1)M_1^2}$$

$$\frac{P_2}{P_1} = 1 + \frac{2\gamma}{\gamma + 1}(M_1^2 - 1) \qquad \frac{T_2}{T_1} = \left[1 + \frac{2\gamma}{\gamma + 1}(M_1^2 - 1)\right]\frac{2 + (\gamma - 1)M_1^2}{(\gamma + 1)M_1^2}$$

These equations are all assuming it is a normal shock and not at an angle, if the shock is at an angle the normal component of M1 needs to be what is used in these equations, but for all necessary purposes, this project will only use normal shocks, as oblique shocks cannot form in a nozzle.

For detecting shock in the diverging converging nozzles the area equation can be used,

$$\frac{A}{A^*} = \frac{1}{M}\left(\frac{1 + \frac{\gamma - 1}{2}M^2}{1 + \frac{\gamma - 1}{2}}\right)^{\frac{\gamma+1}{2(\gamma-1)}}$$

The star in this equation denotes area at which sonic conditions exits, this equation assumes a M1 of 1, so the Mach used in this equation is M2, (Farokhi, 144).

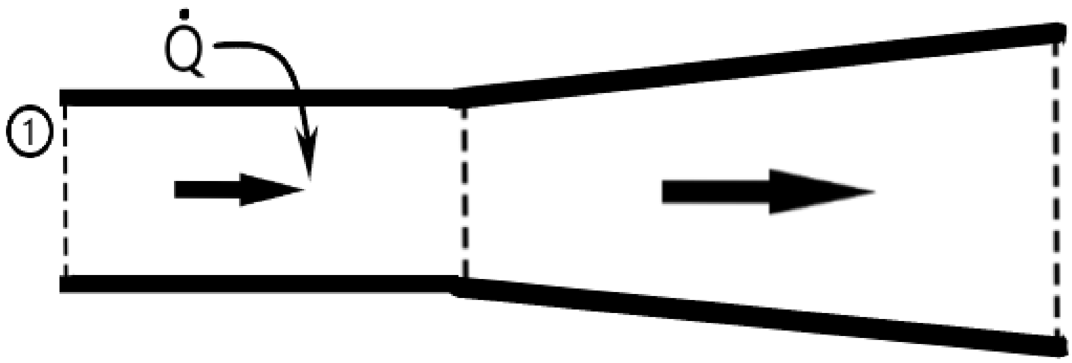
Using the Conservation of mass, exit Mach becomes an equation of P1, T1, and Pb, so when an exit Mach is calculated, it can be placed in the area relationship equation above, if A* is smaller than the Area at the inlet, a shock is not present in the nozzle, however if it is bigger a shock can be placed in the nozzle, Shocks do not affect the Exit Mach since it is based on mass flow rate. Exit conditions will be further discussed later in this document.

Graphical Display:

The GUI itself is going to be basic with Vector based lines, and an appearance similar to the figures shown in the Section about flow pieces. The user inputs will be collected in the GUI via text boxes next to the inlet of the piece they have built.

When it comes to building a piece, Their will be an add piece option in the top right corner with a drop down menu showing the four different options. Each node will attempt to “snap” onto the previous flow node, Shown in Figure 2.

Figure 2



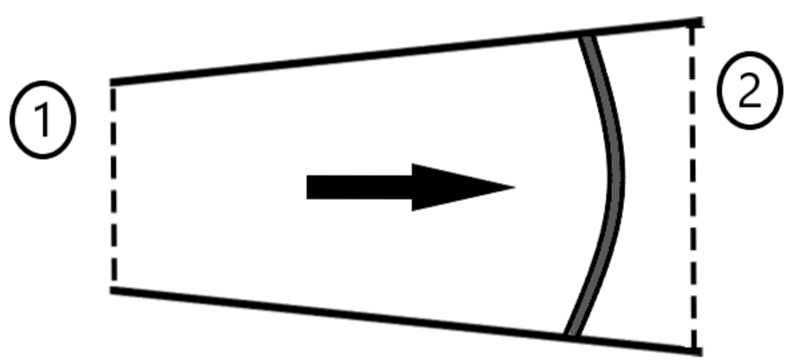
In Figure 2 this shows how the nodes would connect together as displayed in the GUI, in a case like this with a heat tube and then a diverging nozzle. Properties from each carry over to each other so since an area would be provided for the tube, an initial area for the diverging nozzle would not have to be provided. Also for this, all of the equations would calculate exit conditions, and those exit conditions from the heat tube would be inlet conditions for the nozzle.

The GUI will also show the data in the form of a table underneath the graphics, as well as within the generated drawings.

The GUI after user input will display any areas that a shock is determined to form, Figure 3 displays how will be shown to the user:

Figure 3

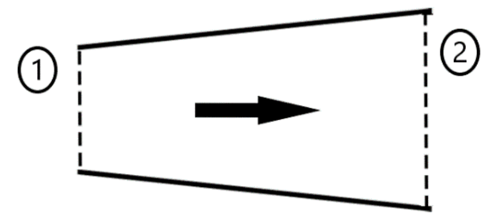
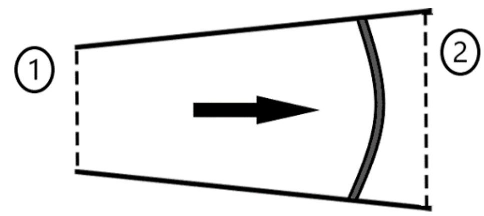
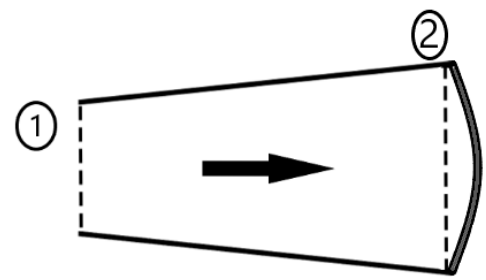
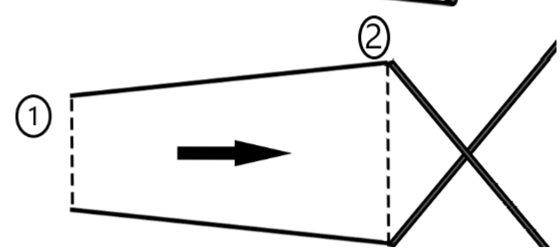
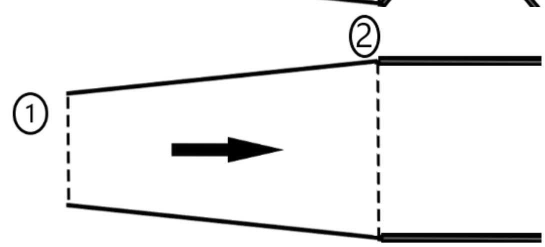
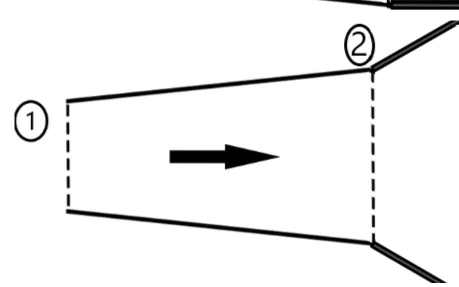
Shock:
Shown as Black and Grey curved line, in program the lines will be blue for easy viewership



This is just an example and the locations of where they are displayed will be calculated by the program.

For the last Node in the series the Exit Condition will also be calculated, the calculated value will be shown in Table 1:

Table 1

Choked Flow	
Shock in Nozzle	
Shock at Exit	
Under Expanded	
Perfectly Expanded	
Over Expanded	

Requirements

Program matches tabulated values from APX A - D, in Faroknhi, S. (2021). On compressible flow with area changing tubes, heat tubes, friction tubes.

See the requirements PDF for a more detailed coverage of software requirements. But to be put briefly, the program should calculate values based on user inputs, display the user generated node configuration in a gui setting, and be savable in a CSV file format.

Implementation

The Implementation of the nozzles as nodes will be as extensions of a class within the Generic Nodes, all of the isentropic relations can be calculated in the generic nodes class and the differing calculations such as, velocity addition due to heat/friction, and acceleration/deceleration due to a change in area will be handled by the extensions.

If the exit conditions are within certain data ranges a the method will call a normal shocks class to work backwards in the node until shock will be placed accordingly in the nozzle thus changing the final conditions. Getters and setters will be utilized for Mach, density, Temp and pressure, all of the calculated values will be in the terms of ratios of initial over exit.

The Nodes will be handled by an array that will be determined by the quantity of nodes, saving and transferring all the values that way. The actual Temp Mach pressure and density values will be calculated in a method within the manager file for the program. The GUI will pull data from the Manager file.

Another Class will be exit condition, called upon when the reservoir pressure is input, this will calculate the requirements for every exit condition type. Exit conditions are based ONLY on the reservoir pressure and Mach number. Three of the Exit conditions have exact reservoir pressure values that they can only occur at, Perfectly Expanded, Shock at Exit, and Choked flow. So the method in this class will calculate the required reservoir pressure for these to occur, and if the given Po is placed at one of those conditions it will place it as such, if it is between one of those conditions then it will choose accordingly,

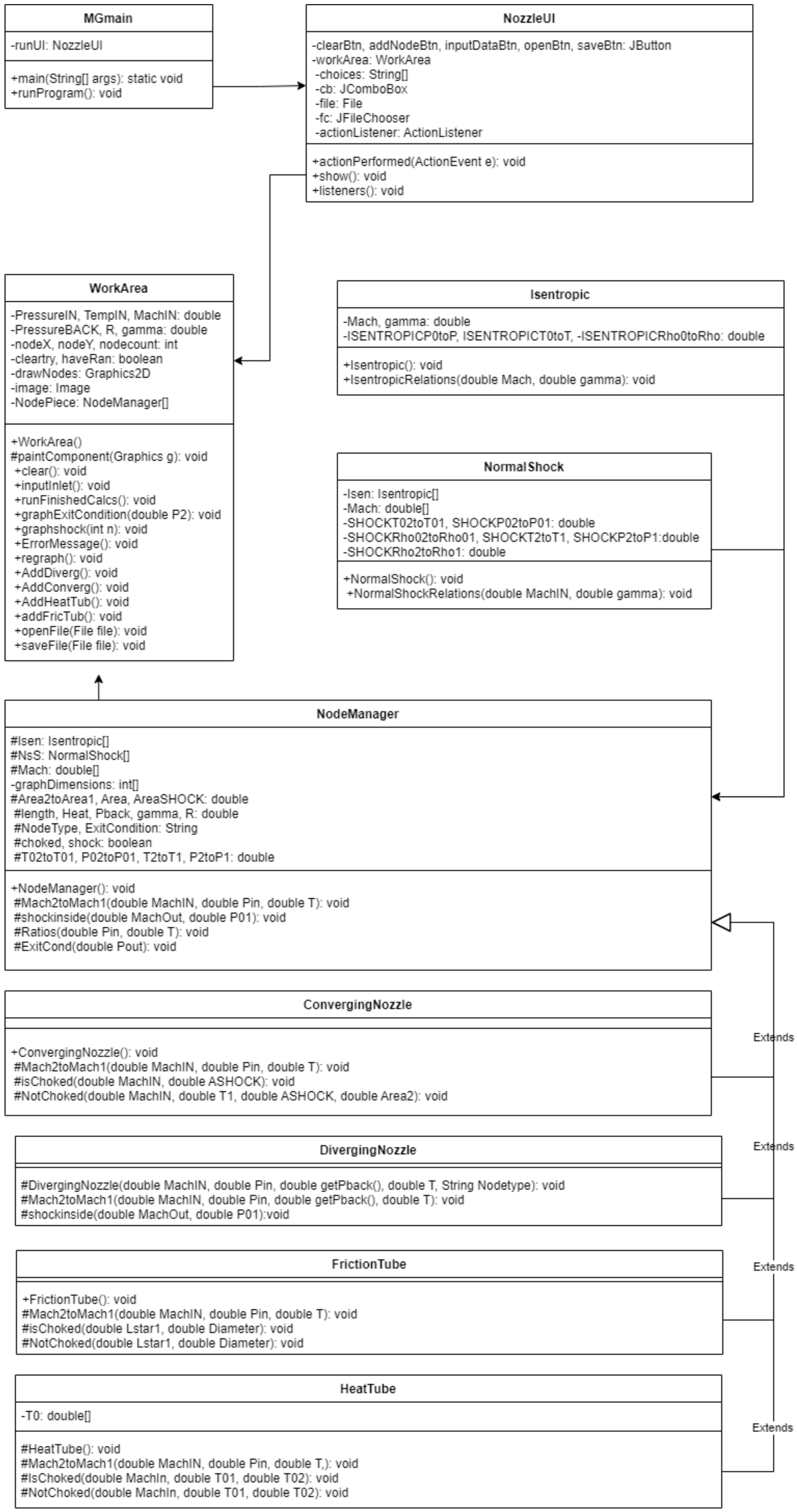
These Exit conditions are based on geometry, reservoir pressure, and Exit pressure, they are not part of the problem statement, but required for the solution of the final values.

The Order biased on a raising Po for exit conditions is,

Choked flow Po < Shock in nozzle Po < Shock at exit Po < Underexpaned Po < Perfectly Expanded Po < Overexpaned Po

All of which can be be seen graphicaly in Table 1,

The UML goes into how these will be handled by the code:


MGmain
-runUI: NozzleUI
+main(String[] args): static void
+runProgram(): void
NozzleUI
-clearBtn, addNodeBtn, inputDataBtn, openBtn, saveBtn: JButton
-workArea: WorkArea
-choices: String[]
-cb: JComboBox
-file: File
-fc: JFileChooser
-actionListener: ActionListener
+actionPerformed(ActionEvent e): void
+show(): void
+listeners(): void
WorkArea
-PressureIN, TempIN, MachIN: double
-PressureBACK, R, gamma: double
-nodeX, nodeY, nodecount: int
-cleartry, haveRan: boolean
-drawNodes: Graphics2D
-image: Image
-NodePiece: NodeManager[]
+WorkArea()
#paintComponent(Graphics g): void
+clear(): void
+inputInlet(): void
+runFinishedCalcs(): void
+graphExitCondition(double P2): void
+graphshock(int n): void
+ErrorMessage(): void
+regraph(): void
+AddDiverg(): void
+AddConverg(): void
+AddHeatTub(): void
+addFricTub(): void
+openFile(File file): void
+saveFile(File file): void
Isentropic
-Mach, gamma: double
-ISENTROPICP0toP, ISENTROPICT0toT, -ISENTROPICRho0toRho: double
+Isentropic(): void
+IsentropicRelations(double Mach, double gamma): void
NormalShock
-Isen: Isentropic[]
-Mach: double[]
-SHOCKT02toT01, SHOCKP02toP01: double
-SHOCKRho02toRho01, SHOCKT2toT1, SHOCKP2toP1:double
-SHOCKRho2toRho1: double
+NormalShock(): void
+NormalShockRelations(double MachIN, double gamma): void
NodeManager
#Isen: Isentropic[]
#NsS: NormalShock[]
#Mach: double[]
-graphDimensions: int[]
#Area2toArea1, Area, AreaSHOCK: double
#length, Heat, Pback, gamma, R: double
#NodeType, ExitCondition: String
#choked, shock: boolean
#T02toT01, P02toP01, T2toT1, P2toP1: double
+NodeManager(): void
#Mach2toMach1(double MachIN, double Pin, double T): void
#shockinside(double MachOut, double P01): void
#Ratios(double Pin, double T): void
#ExitCond(double Pout): void
ConvergingNozzle
+ConvergingNozzle(): void
#Mach2toMach1(double MachIN, double Pin, double T): void
#isChoked(double MachIN, double ASHOCK): void
#NotChoked(double MachIN, double T1, double ASHOCK, double Area2): void
Extends
DivergingNozzle
#DivergingNozzle(double MachIN, double Pin, double getPback(), double T, String Nodetype): void
#Mach2toMach1(double MachIN, double Pin, double getPback(), double T): void
#shockinside(double MachOut, double P01):void
Extends
FrictionTube
+FrictionTube(): void
#Mach2toMach1(double MachIN, double Pin, double T): void
#isChoked(double Lstar1, double Diameter): void
#NotChoked(double Lstar1, double Diameter): void
Extends
HeatTube
-T0: double[]
#HeatTube(): void
#Mach2toMach1(double MachIN, double Pin, double T,): void
#IsChoked(double MachIn, double T01, double T02): void
#NotChoked(double MachIn, double T01, double T02): void
Extends

References:

Farokhi, S. (2021). Aircraft Propulsion. John Wiley and Sons.

Appendices:

Appendices Blank For Now Will be Updated as Project Progresses