Engineering Model to Calculate Mass Flow Rate of a Two-Phase Saturated Fluid Through An Injector Orifice

Brian J. Solomon
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/gradreports

Part of the Aerospace Engineering Commons

Recommended Citation
https://digitalcommons.usu.edu/gradreports/110

This Report is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Plan B and other Reports by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.
ENGINEERING MODEL TO CALCULATE MASS FLOW RATE OF A TWO-PHASE SATURATED FLUID THROUGH AN INJECTOR ORIFICE

by

Brian J. Solomon

A report submitted in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE in Aerospace Engineering

Approved:

Dr. Stephen A. Whitmore
Major Professor

Dr. R. Rees Fullmer
Committee Member

Dr. Donald Cripps
Committee Member

UTAH STATE UNIVERSITY
Logan, Utah
2011
Abstract

ENGINEERING MODEL TO CALCULATE MASS FLOW RATE OF A TWO-PHASE SATURATED FLUID THROUGH AN INJECTOR ORIFICE

by

Brian J. Solomon, Master of Science
Utah State University, 2011

Major Professor: Dr. Stephen A. Whitmore
Department: Mechanical and Aerospace Engineering

An engineering model is developed to calculate mass flow rate of nitrous oxide, a self-pressurizing saturated oxidizer commonly used in hybrid rocket motors. While use of N$_2$O in a self-pressurizing oxidizer system has its advantages, there also exists some disadvantages. N$_2$O cannot be accurately modelled using traditional ideal gas, compressible, or incompressible flow assumptions. To obtain accurate mass flow rate this one-dimensional analysis includes both incompressible fluid and homogeneous equilibrium mass flow rate models. Massflow calculations from the two models are independently weighted and summed to obtain representative two-phase mass flow rate. Fluid properties are iterated in time by keeping track of fluid enthalpy and are propagated across the injector using either isentropic or adiabatic assumptions. The model excellently predicts mass flow rates as verified by comparison to experimental cold flow data. The experimental conditions resulting from the test apparatus set-up produces fluid stratification and mixing effects that cannot be modelled by the algorithm as developed. Thus the run tank and temperature drops as predicted by the model are significantly larger than measured.
Acknowledgments

I would like to thank my major professor, Dr. Stephen A. Whitmore for his guidance in the fields of rocketry and fluid dynamics. I learned a lot from him over the past several years. Shannon Eilers helped guide my project with his willingness to help me when I got lost, he has been very helpful in fielding my many questions. He was also instrumental in providing the experimental data used for model validation.

I would additionally like to thank my employer, ATK, for providing me with the financial means to complete this degree.

Finally, I would like to thank my wife, Megan, and two children, Scarlett and Benson. They have been very patient with me during the research and work involved in completing this project. I look forward to getting this behind me so that I can spent more time with them and less time with my computer.

Brian J. Solomon
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>iv</td>
</tr>
<tr>
<td>List of Tables</td>
<td>vii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>viii</td>
</tr>
<tr>
<td>Acronyms</td>
<td>ix</td>
</tr>
<tr>
<td>Nomenclature</td>
<td>x</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2 Literature Search and Review of Previous Work</td>
<td>3</td>
</tr>
<tr>
<td>2.1 Fluid Properties</td>
<td>3</td>
</tr>
<tr>
<td>2.1.1 NIST CO$_2$ Fluid Properties</td>
<td>3</td>
</tr>
<tr>
<td>2.1.2 Helmholtz Energy Method</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Mass Flow Models</td>
<td>5</td>
</tr>
<tr>
<td>2.2.1 Incompressible Viscous Fluid Model</td>
<td>8</td>
</tr>
<tr>
<td>2.2.2 Homogeneous Equilibrium Model</td>
<td>8</td>
</tr>
<tr>
<td>2.2.3 Non-Homogeneous Non-Equilibrium Model</td>
<td>9</td>
</tr>
<tr>
<td>2.2.4 NHNE Model Correction</td>
<td>10</td>
</tr>
<tr>
<td>3 Two-Phase Enthalpy Algorithm</td>
<td>12</td>
</tr>
<tr>
<td>3.1 Initial Conditions</td>
<td>12</td>
</tr>
<tr>
<td>3.2 Fluid Properties Calculation</td>
<td>14</td>
</tr>
<tr>
<td>3.3 Fluid Property Propagation</td>
<td>15</td>
</tr>
<tr>
<td>3.4 Mass Flow Rate Prediction</td>
<td>16</td>
</tr>
<tr>
<td>3.5 Update Fluid Properties</td>
<td>16</td>
</tr>
<tr>
<td>4 Experimental Validation</td>
<td>19</td>
</tr>
<tr>
<td>4.1 Experimental Apparatus</td>
<td>19</td>
</tr>
<tr>
<td>4.2 Experimental Procedures</td>
<td>21</td>
</tr>
<tr>
<td>4.3 Model Comparisons</td>
<td>22</td>
</tr>
<tr>
<td>5 Conclusions</td>
<td>29</td>
</tr>
<tr>
<td>References</td>
<td>31</td>
</tr>
</tbody>
</table>
Appendices

Appendix A  Fluid Constants ................................................. 33
    A.1  Comparison of CO$_2$ and N$_2$O Properties ................... 34
Appendix B  MATLAB Code .................................................. 35
    B.1  Blow Down Simulation ........................................... 35
    B.2  CO$_2$ Properties - Helmholtz Energy Method ................. 45
    B.3  CO$_2$ Properties - NIST Webbook Method ................... 58
    B.4  CO$_2$ Properties - Helmholtz vs. NIST Plotter ............. 63
    B.5  Density and Enthalpy to Temperature ......................... 69
    B.6  Enthalpy Plotter ................................................. 70
    B.7  Discrete Wavelet Transform Derivative ....................... 71
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Saturated liquid and vapor compressibility factors for CO$_2$ and N$_2$O</td>
<td>7</td>
</tr>
<tr>
<td>A.1 Comparison of CO$_2$ and N$_2$O Properties</td>
<td>34</td>
</tr>
</tbody>
</table>
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Pressure of CO₂ as calculated by NIST Webbook and Helmholtz Models.</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Specific Entropy of CO₂ as calculated by NIST Webbook and Helmholtz Models.</td>
<td>6</td>
</tr>
<tr>
<td>2.3</td>
<td>Specific Enthalpy of CO₂ as calculated by NIST Webbook and Helmholtz Models.</td>
<td>7</td>
</tr>
<tr>
<td>2.4</td>
<td>Injector orifice cross section defining mass flow parameters.</td>
<td>8</td>
</tr>
<tr>
<td>2.5</td>
<td>Weighting coefficients vs non-equilibrium parameter κ, as presented in Dyer.</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>Two-Phase Enthalpy algorithm flow chart.</td>
<td>13</td>
</tr>
<tr>
<td>3.2</td>
<td>CO₂ temperature as a function of density for different specific enthalpies.</td>
<td>18</td>
</tr>
<tr>
<td>4.1</td>
<td>Cold flow blowdown physical setup.</td>
<td>20</td>
</tr>
<tr>
<td>4.2</td>
<td>Comparison, Model vs cold flow, run tank fluid pressure.</td>
<td>23</td>
</tr>
<tr>
<td>4.3</td>
<td>Comparison, Model vs cold flow, run tank fluid temperature.</td>
<td>24</td>
</tr>
<tr>
<td>4.4</td>
<td>Comparison, Model vs cold flow, run tank total fluid mass from load cells.</td>
<td>24</td>
</tr>
<tr>
<td>4.5</td>
<td>Comparison, Model vs cold flow, fluid mass flow rate.</td>
<td>25</td>
</tr>
<tr>
<td>4.6</td>
<td>CO₂ liquid density comparison.</td>
<td>26</td>
</tr>
<tr>
<td>4.7</td>
<td>Comparison, curve fit vs Helmholtz energy density, fluid mass flow rate.</td>
<td>27</td>
</tr>
<tr>
<td>4.8</td>
<td>Model vs cold flow, fluid mass flow rate comparison, detailed initial time.</td>
<td>28</td>
</tr>
<tr>
<td>4.9</td>
<td>Numerical differentiation methods comparison, cold flow mass flow rates.</td>
<td>28</td>
</tr>
</tbody>
</table>
Acronyms

CO₂  carbon dioxide
HEM  homogeneous equilibrium model
N₂O  nitrous oxide
NHNE  non-homogeneous non-equilibrium model
Nomenclature

\( \chi_1 \)  Orifice upstream quality
\( \chi_o \)  Initial fluid quality
\( \delta \)  Reduced density
\( \dot{H} \)  Total enthalpy flow rate
\( \dot{m}_{HEM} \)  Mass flow rate, Homogeneous Equilibrium Model (HEM)
\( \dot{m}_{inc} \)  Mass flow rate, incompressible fluid
\( \dot{m}_{NHNE} \)  Mass flow rate, Non-Homogeneous Non-Equilibrium Model (NHNE)
\( \dot{m}_{venturi} \)  Mass flow rate, venturi
\( \kappa \)  Non-equilibrium parameter
\( \phi \)  Dimensionless Helmholtz energy
\( \phi^o \)  Dimensionless Helmholtz energy, ideal gas part
\( \phi^r \)  Dimensionless Helmholtz energy, residual part
\( \rho \)  Density
\( \rho^L \)  Liquid density
\( \rho^V \)  Vapor density
\( \rho_1 \)  Orifice upstream density
\( \rho_2 \)  Orifice downstream density
\( \rho_c \)  Critical density
\( \rho_o \)  Initial fluid density
\( \tau \)  Inverse reduced temperature
\( \tau_b \)  Bubble growth time
\( \tau_r \)  Residence time of fluid in an injector
\( A_t \)  Venturi throat cross sectional area
\( A_c \)  Cross sectional area
\( A_{in} \)  Venturi inlet cross sectional area
\( C_d \)  Discharge coefficient
\( h^L \) Liquid specific enthalpy
\( h^V \) Vapor specific enthalpy
\( h_1 \) Orifice upstream specific enthalpy
\( h_2 \) Orifice downstream specific enthalpy
\( H_o \) Initial fluid total enthalpy
\( h_o \) Initial fluid specific enthalpy
\( L \) Injector length
\( M_o \) Initial fluid mass
\( M^L_o \) Initial liquid mass
\( M^V_o \) Initial vapor mass
\( P \) Pressure
\( P_t \) Venturi throat fluid pressure
\( P_1 \) Orifice upstream pressure
\( P_2 \) Orifice downstream pressure
\( P_{\nu_1} \) Upstream fluid vapor pressure
\( P_{in} \) Venturi inlet fluid pressure
\( R \) Gas constant
\( s^L \) Liquid specific entropy
\( s^V \) Vapor specific entropy
\( s_1 \) Orifice upstream specific entropy
\( s_2 \) Orifice downstream specific entropy
\( s_o \) Initial fluid specific entropy
\( T_1 \) Orifice upstream temperature
\( T_c \) Critical temperature
\( T_o \) Initial fluid temperature
\( V_{tank} \) Tank volume
\( Z \) Compressibility factor
Chapter 1

Introduction

The use of nitrous oxide (N$_2$O) as a rocket fuel dates back to one of modern rocketry’s founding fathers, Robert H. Goddard. In one of his earliest patents, U.S. Patent 1,103,503 [1], Goddard described a rocket that would use N$_2$O and gasoline as fuels. In recent times N$_2$O has been used as an oxidizer in non-military suborbital hybrid rockets. Most notably, N$_2$O was used by Scaled Composites’ experimental spacecraft named SpaceShipOne [2]. In 2004, SpaceShipOne won the Ansari X Prize after sending the first human to space via a privately funded spacecraft [3]. Going forward the Scaled Composites team has partnered with Virgin GALACTIC becoming the world’s first space tourism company with initial flights planned for 2012. Like SpaceShipOne, SpaceShipTwo will use a hybrid rocket using N$_2$O as its oxidizer [4].

While N$_2$O does not offer as high as performance as cryogenic oxidizers like liquid oxygen (LOX) or non-saturated propellants like nitrogen tetraoxide (N$_2$O$_4$) or hydrogen peroxide (H$_2$O$_2$), nitrous oxide has several advantages that make it very competitive for hybrid applications. First, in contrast to N$_2$O$_4$ and H$_2$O$_2$, N$_2$O is non-toxic and can be handled without special precautions. Second, Unlike LOX, Nitrous Oxide is highly storable, and allows rocket systems to be loaded well far in advance of launch. Store-ability also presents significant advantages for in-space propulsion systems. Finally, N$_2$O is a self-pressurizing propellant, and this property can be used to reduce the complexity of the propellant delivery systems. Unlike other more widely used rocket propellants, N$_2$O exists as a saturated liquid at room temperature, and has a relatively high vapor pressure of 5.729 MPa at 293.15 K (830.9 psi at 68 °F). Using N$_2$O as a self-pressurizing propellant eliminates or reduces the need for complex, costly, and heavy pumps or other types of pressurization systems. In addition to simplification of a rocket propellant system design, N$_2$O also offers
safety improvements over other more highly energetic propellants. Though not without a history of lethal disasters, N\textsubscript{2}O is non-toxic and relatively more forgiving than other more energetic rocket propellants. For these reasons, N\textsubscript{2}O is preferred by commercial space flight operators and amateur rocket system designers.

While use of N\textsubscript{2}O in a self-pressurizing oxidizer system has its advantages, there also exists some disadvantages. Unlike more energetic propellants, N\textsubscript{2}O cannot be accurately modelled using traditional ideal gas, compressible, or incompressible flow assumptions. The model developed here will build upon earlier studies attempting to predict flow characteristics of self-pressurizing fluids for use in hybrid rockets. The two-phase flow model to be presented is developed for both nitrous oxide and carbon dioxide (CO\textsubscript{2}). Clearly, due to its non-energetic nature CO\textsubscript{2} cannot be used as a rocket propellant; however, its saturation properties including density, vapor pressure, and viscosity are very similar to N\textsubscript{2}O. Thus CO\textsubscript{2} makes a convenient test analog, and can be used to evaluate the accuracy of the developed two-phase flow models with reduced risk and cost. A detailed comparison of CO\textsubscript{2} and N\textsubscript{2}O properties is presented in Table A.1.
Chapter 2

Literature Search and Review of Previous Work

In this section two methods of determining fluid properties of N$_2$O and CO$_2$ are presented from the literature. Several mass flow models are also presented and discussed. A combination of two of the presented models is critiqued and redefined.

2.1 Fluid Properties

Two methods are investigated for calculating the saturated fluid properties. The first method uses NIST Chemistry WebBook database [5]. NIST Webbook contains thermophysical properties for multiple fluids, including both N$_2$O and CO$_2$. The second method used to calculate CO$_2$ properties, is by Span and Wagner [6]. This model formulates the equation of state explicitly in terms of the Helmholtz free energy. A similar method developed by Dyer [7] based upon the model of Span [8] is used for N$_2$O fluid properties.

2.1.1 NIST CO$_2$ Fluid Properties

A table of saturated CO$_2$ fluid properties can be downloaded from NIST Webbook website in ASCII format. Data may be tabulated in increments of temperature or pressure between the triple point and critical point. Both liquid and vapor properties for either temperature or pressure are provided:

\[ P, \rho^L, \rho^V, s^L, s^V, h^L, h^V = NISTWEBBOOK(216.592K \cdots 304.1282K) \]  \hspace{1cm} (2.1)

\[ T, \rho^L, \rho^V, s^L, s^V, h^L, h^V = NISTWEBBOOK(0.51795MPa \cdots 7.3773MPa) \]  \hspace{1cm} (2.2)

Eqs. 2.1 and 2.2 are functions coded to return select saturated CO$_2$ properties for given temperatures or pressures between the triple and critical points respectively. In both Eqs.
2.1 and 2.2 $\rho$ is the density, $s$ is the specific entropy, and $h$ is the specific enthalpy where the superscripts $L$ and $V$ represent the liquid and vapor properties respectively. In Eq. 2.1, $P$ is pressure and in Eq. 2.2 $T$ is the temperature. Identical functions are created for $N_2O$.

### 2.1.2 Helmholtz Energy Method

Span and Wagner present an equation of state that predicts CO$_2$ thermodynamic properties to within uncertainties of available experimental data. The fundamental equation given in Span is the dimensionless Helmholtz energy $\phi$, which is split into two parts. The first part $\phi^o$ depends upon the ideal-gas behaviour and the second part $\phi^r$ takes into account the residual fluid behaviour.

$$\phi(\delta, \tau) = \phi^o(\delta, \tau) + \phi^r(\delta, \tau) \quad (2.3)$$

In Eq. 2.3, $\tau$ is the inverse reduced temperature, and $\delta$ is the reduced density, and are defined as:

$$\tau = T_c/T \quad (2.4)$$

$$\delta = \rho/\rho_c \quad (2.5)$$

where $T_c$ and $\rho_c$ are the critical temperature and density respectively.

There exists a relationship between the Helmholtz energy defined in Eq. 2.3 and the thermodynamic properties needed in this analysis. Specifically the fluid pressure, specific entropy, and specific enthalpy are related to the Helmholtz energy by:

$$\frac{P(\delta, \tau)}{\rho RT} = 1 + \delta \phi^r_\delta \quad (2.6)$$

$$\frac{s(\delta, \tau)}{R} = \tau(\phi^o_r + \phi^r_r) - \phi^o - \phi^r \quad (2.7)$$

$$\frac{h(\delta, \tau)}{RT} = 1 + \tau(\phi^o_r + \phi^r_r) + \delta \phi^o_\delta \quad (2.8)$$
where \( R \) is the gas constant and the subscripted ideal gas and residual Helmholtz energies are the following derivatives:

\[
\phi^r_\delta = \frac{d\phi^r}{d\delta}, \quad \phi^o_\tau = \frac{d\phi^o}{d\tau}, \quad \phi^r_\tau = \frac{d\phi^r}{d\tau}, \quad \phi^o_\delta = \frac{d\phi^o}{d\delta} \tag{2.9}
\]

Span further defines the ideal gas and residual parts of the Helmholtz energy and their derivatives. The equations become long summations with hundreds of coefficients and exponents. The final equations are all presented in Span and can also be found as coded for this paper in the CO\(_2\) properties source code in Appendix B.2.

Equations presented by Span and Wagner [6] are compiled into an algorithm that returns the fluid properties of CO\(_2\).

\[
P, s^L, h^L = Helmholtz(\tau, \delta^L) \tag{2.10}
\]

\[
P, s^V, h^V = Helmholtz(\tau, \delta^V) \tag{2.11}
\]

CO\(_2\) pressure, specific entropy, and specific enthalpy as returned from the NIST Webbook, and Span-Wagner method are compared in Figures 2.1, 2.2, 2.3. The comparisons are essentially identical. Advantages to using Span-Wagner equations directly are: no need to interpolate between tabulated data sets, and properties outside of the saturation region are possible. Once again, the same is true for N\(_2\)O. The Webbbook comparisons are used primarily to validate the Span-Wagner equations as programmed by the author.

### 2.2 Mass Flow Models

The model developed here to predict mass flow rate through and orifice style injector will use a weighted average of two different mass flow rate models. The first being the traditional incompressible viscous fluid model and the second being the Homogeneous Equilibrium Model (HEM). Figure 2.4 depicts a cross section of an orifice with the upstream and downstream mass flow parameter locations defined. The dual model method is necessary for a saturated fluid like N\(_2\)O or CO\(_2\) because they do not behave like an incompressible liquid
Fig. 2.1: Pressure of CO$_2$ as calculated by NIST Webbook and Helmholtz Models.

Fig. 2.2: Specific Entropy of CO$_2$ as calculated by NIST Webbook and Helmholtz Models.
Fig. 2.3: Specific Enthalpy of CO\textsubscript{2} as calculated by NIST Webbook and Helmholtz Models.

or an ideal gas even at the saturated liquid and saturated vapor states. The compressibility factor, $Z$, for \textsubscript{N\textsubscript{2}O} and CO\textsubscript{2} as calculated by [9]:

$$Z = \frac{P}{\rho RT}$$

are shown in Table 2.1. A fluid with a compressibility factor of either zero or one could use an incompressible liquid or ideal gas assumption. It is clearly seen by the compressibility factors of either \textsubscript{N\textsubscript{2}O} or CO\textsubscript{2} that either of these assumptions would yield considerable errors in predicting mass flow rates.

Table 2.1: Saturated liquid and vapor compressibility factors for CO\textsubscript{2} and \textsubscript{N\textsubscript{2}O}

<table>
<thead>
<tr>
<th>Fluid</th>
<th>CO\textsubscript{2}</th>
<th>\textsubscript{N\textsubscript{2}O}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saturated Liquid</td>
<td>0.16</td>
<td>0.135</td>
</tr>
<tr>
<td>Saturated Vapor</td>
<td>0.473</td>
<td>0.533</td>
</tr>
</tbody>
</table>
2.2.1 Incompressible Viscous Fluid Model

For incompressible flows where fluid resides in the compressed liquid phase, mass flow rate, $\dot{m}_{inc}$, through an orifice or injector is defined as [10]:

$$\dot{m}_{inc} = C_d \cdot A_c \sqrt{2\rho_1(P_1 - P_2)}$$

(2.13)

where $C_d$ is the orifice or injector discharge coefficient, $A_c$ is the cross sectional area of the orifice, $\rho_1$ is the fluid density upstream of the orifice, and $P_1$ and $P_2$ are the fluid pressures upstream and downstream of the orifice respectively. This model accurately predicts the mass flow rate of a fluid with a compressibility factor near zero.

2.2.2 Homogeneous Equilibrium Model

For the flow cases of interest here, the static pressure of the fluid flowing into the injector is very close to the saturation state. As the flow begins to accelerate into the entrance of the injector, the pressure drops below the vapor pressure and the fluid begins to
flash from liquid to vapor. This results in a two-phase flow of vapor pockets or bubbles and liquid. One method proposed to predict mass flow of a two-phase flow through an injector is referred to as the Homogeneous Equilibrium Model (HEM) [11]. The HEM model assumes that the liquid-to-vapor phase change is isentropic, that the liquid and vapor portions of the flow are in thermodynamic equilibrium, and that there is no velocity difference between the fluid phases. The HEM mass flow rate \( \dot{m}_{\text{HEM}} \), is a function of the injector discharge coefficient \( C_d \), the injector cross sectional area \( A_c \), the downstream fluid density \( \rho_2 \), and the upstream and downstream fluid specific enthalpies \( h_1 \) and \( h_2 \):

\[
\dot{m}_{\text{HEM}} = C_d \cdot A_c \cdot \rho_2 \sqrt{2(h_1 - h_2)}
\]  

(2.14)

### 2.2.3 Non-Homogeneous Non-Equilibrium Model

Dyer et al. [7] have developed a model which combines the incompressible fluid model [Eq. 2.13] and the homogeneous equilibrium model (HEM) [Eq. 2.14] referred to here as non-homogeneous non-equilibrium (NHNE) model. The NHNE model weights incompressible fluid and HEM models using a "non-equilibrium" parameter. The non-equilibrium parameter, \( \kappa \) [Eq. 2.17], is defined in Dyer as the ratio of the bubble growth time, \( \tau_b \) [Eq. 2.15], to residence time of the fluid in the injector element, \( \tau_r \) [Eq. 2.16].

\[
\tau_b \equiv \sqrt{\frac{3}{2} \frac{\rho L}{P_{v_1} - P_2}}
\]  

(2.15)

\[
\tau_r \equiv L \sqrt{\frac{\rho L}{2 \cdot (P_1 - P_2)}}
\]  

(2.16)

\[
k = \frac{\tau_b}{\tau_r} = \sqrt{\frac{P_1 - P_2}{P_{v_1} - P_2}}
\]  

(2.17)

Note the injector length \( L \), a constants in \( \tau_b \) and \( \tau_r \) have been neglected by Dyer. Dropping the injector length and other constants has little effect on end results because \( \kappa \) is used to
weight the incompressible and HEM mass flow rates in an inversely proportional manner. The upstream fluid vapor pressure $P_{v_1}$, is simply the total pressure for a saturated fluid. Dyer et al. have also noted $\kappa$ is similar to the inverse of the cavitation number, except the denominator in $\kappa$ is the difference between vapor pressure and downstream pressure rather than vapor pressure and upstream pressure, as in the cavitation number. The NHNE mass flow rate $\dot{m}_{NHNE}$, is defined here as:

$$\dot{m}_{NHNE} = C_d \cdot A_c \cdot \left( \frac{1}{1 + \kappa} \dot{m}_{inc} + \left( 1 - \frac{1}{1 + \kappa} \right) \dot{m}_{HEM} \right)$$ \hspace{1cm} (2.18)

### 2.2.4 NHNE Model Correction

An inconsistency between description of Dyer et al. and the mass flow rate equation 2.18 exists. In the text Dyer et al. states:

"If the bubble growth time, $\tau_b$, is large compared with the liquid residence time, $\tau_r$, the flowrate should be well predicted by the classic incompressible $C_dA$ equation (Eq. 25). If, on the other hand, $\tau_r$ is much larger than $\tau_b$ the flowrate should approach the critical flowrate as predicted by the HEM."

In the flowrate equation in Dyer, the weighting coefficients are presented as:

Incompressible Coefficient = $\frac{1}{1 + \kappa}$ \hspace{1cm} (2.19)

HEM Coefficient = $1 - \left( \frac{1}{1 + \kappa} \right)$ \hspace{1cm} (2.20)

Figure 2.5 shows coefficients in Eq. 2.19 and Eq. 2.20 plotted against the non-equilibrium parameter, $\kappa$. It is clearly seen that for large $\kappa$ and therefore large bubble growth time, $\tau_b$, compared to liquid residence time, $\tau_r$, HEM dominates and not the incompressible portion, as was stated in Dyer.
The description used by Dyer is physically consistent, in contrast to Dyer’s original presentation of the NHNE massflow equation. Stating that liquid residence time, \( \tau_r \), is "large," is consistent with stating a specific mass of fluid takes a relatively long time to get through an injector. Stating that bubble growth time, \( \tau_b \), is "small," is consistent with stating a bubble develops quickly. When determining mass flowrate it is physically more consistent for a flow that develops bubbles quickly relative to the amount of time taken for a mass of fluid to get through an injector (early bubbly flow,) that flowrate is best predicted by the HEM model. Alternatively, it is physically more consistent for a flow that takes a long time for bubbles to develop relative to the amount of time it takes for a mass of fluid to get through the injector (late bubbly flow), that flowrate would be best predicted by the incompressible model. Based on this, it will be assumed Dyer’s description was intended, and coefficients in the NHNE equation will be redefined as:

\[
\dot{m}_{NHNE} = C_d \cdot A_c \cdot \left( \frac{1}{1 + \kappa} \right) \dot{m}_{inc} + \frac{1}{1 + \kappa} \dot{m}_{HEM} \tag{2.21}
\]
Chapter 3

Two-Phase Enthalpy Algorithm

A two-phase enthalpy algorithm is described in this section. The algorithm is designed to predict the mass flow rate of N\textsubscript{2}O or CO\textsubscript{2} from an oxidizer storage tank through an injector. This algorithm is similar to that developed by Whitmore and Chandler [12]. The Whitmore and Chandler algorithm assumes an isentropic fluid expansion in order to propagate the fluid properties forward in time, giving it the "Two-Phase Entropy Model" name. The algorithm presented here will differ in that it will propagate the fluid properties forward in time by keeping track of the total fluid enthalpy rather than the entropy. The isentropic expansion across the injector is known to be physically incorrect. The assumption of an injector discharge coefficient already violates the isentropic flow assumption. The enthalpy model represents a more physically plausible extension of the entropy model. Figure 3.1 shows a top level flow chart of the major steps of the developed algorithm described here.

3.1 Initial Conditions

The algorithm initial conditions are derived by assuming an initial oxidizer tank volume \( V_{\text{tank}} \), mass of oxidizer loaded into the tank \( M_o \), and a temperature \( T_o \) at which the oxidizer is loaded. The fluid initial density \( \rho_o \) is calculated by assuming that the fluid in the tank is a uniform mixture of liquid (\( M_o^L \)) and vapor (\( M_o^V \)).

\[
\rho_o = \frac{M_o^L + M_o^V}{V_{\text{tank}}} = \frac{M_o}{V_{\text{tank}}} \tag{3.1}
\]

Since the initial saturated fluid temperature, and density, are known, the initial saturated pressure, \( P_o \), the initial saturated liquid and vapor enthalpies, \( h_o^L \) and \( h_o^V \), and...
Fig. 3.1: Two-Phase Enthalpy algorithm flow chart.
entropies, $s^L_o$ and $s^V_o$, are calculated using Eqs. 2.10 and 2.11. The initial fluid quality $\chi_o$, and therefore initial specific enthalpy $h_o$, and specific entropy $s_o$, is calculated by:

$$\chi_o = \frac{\rho^V}{\rho_o} \cdot \frac{\rho^L_o - \rho_o}{\rho^L - \rho^V} \quad (3.2)$$

$$h_o = h^V_o \cdot \chi_o + h^L_o \cdot (1 - \chi_o) \quad (3.3)$$

$$s_o = s^V_o \cdot \chi_o + s^L_o \cdot (1 - \chi_o) \quad (3.4)$$

The total initial fluid enthalpy $H_o$, is calculated by multiplying the total initial fluid mass in the tank $M_o$, by the initial specific enthalpy $h_o$:

$$H_o = M_o \cdot h_o \quad (3.5)$$

### 3.2 Fluid Properties Calculation

Since the fluid state is propagated forward in time by tracking the total fluid enthalpy $H$, the new state must be calculated from the updated density $\rho_{i+1}$, and specific enthalpy $h_{i+1}$, after each time step. This calculation is problematic because the Span and Wagner Helmoltz energy equations [Eqs. 2.10 and 2.11] require temperature and density as inputs. To resolve this problem Eqs. 2.10, 2.11, 3.2, 3.3, and 3.4 were combined into a single function that returns pressure, quality, specific enthalpy, and specific entropy:

$$[P, \chi, h, s] = f(T, \rho) \quad (3.6)$$

Since the state of a fluid can be calculated by any two independent properties, a non-linear solver is used with Eq. 3.6 to obtain all fluid properties from any two. There are several instances where this technique is implemented in this algorithm. In the first case, both of the known properties are outputs of Eq. 3.6 with neither of the two inputs being known.
For this case Eq. 3.7 is solved with $P_{\text{known}}$ and $\chi_{\text{known}}$ as knowns and everything else unknown:

$$
\begin{bmatrix}
P_{\text{known}}, \chi, h, s \\
P, \chi_{\text{known}}, h, s
\end{bmatrix}
= 
\begin{bmatrix}
f(T, \rho) \\
f(T, \rho)
\end{bmatrix}
$$

(3.7)

To solve the problem presented in Eq. 3.7 a least-squares method is used. The MATLAB optimization toolbox contains a function called lsqnonlin designed to solve non-linear equations using a least-squares approach. In this case, the function defined in Eq. 3.6 minus the known tank pressure $P_1$, and quality $\chi_1$, are input into the lsqnonlin function as shown here:

$$
[T_1, \rho_1] = \text{lsqnonlin}
\left(
\begin{bmatrix}
P(T, \rho) - P_1 \\
\chi(T, \rho) - \chi_1
\end{bmatrix}
\right)
$$

(3.8)

The lsqnonlin function returns the tank temperature $T_1$, and density $\rho_1$, that equate to the known pressure and quality. With the temperature and density known, the remaining properties are calculated using Eq. 3.6.

### 3.3 Fluid Property Propagation

The fluid properties are propagated across the injector to the downstream state using either an isentropic or adiabatic assumption. The isentropic case assumes that the flow across the injector is reversible, and therefore the downstream entropy $s_2$, is equal to the upstream entropy $s_1$, [Eq. 3.9]. The adiabatic case assumes no heat is lost or added to the fluid as it flows through the injector, and therefore the downstream enthalpy $h_2$, is equal to the upstream enthalpy $h_1$, [Eq. 3.10].

$$
s_1 = s_2 
$$

(3.9)

$$
h_1 = h_2
$$

(3.10)

For the case where the injector is venting to ambient conditions, the downstream pressure is taken to be atmospheric. As was done for the upstream fluid properties in Eqs. 3.7 and
3.8, with the atmospheric pressure being known and the isentropic or adiabatic injector assumption, the downstream injector fluid properties are calculated by:

\[
[T_2, \rho_2] = \text{lsqnonlin} \left( \begin{bmatrix} P(T, \rho) - P_2 \\ s(T, \rho) - s_2 \end{bmatrix} \right) \tag{3.11}
\]

or

\[
[T_2, \rho_2] = \text{lsqnonlin} \left( \begin{bmatrix} P(T, \rho) - P_2 \\ h(T, \rho) - h_2 \end{bmatrix} \right) \tag{3.12}
\]

where the remaining downstream fluid properties are calculated by Eq. 3.6.

It should be noted that CO\(_2\), due to it’s high triple point pressure cannot exist in liquid phase at pressures below that of it’s triple point pressure of 517.95 kPa. At any pressure below this value, the CO\(_2\) flow would consist of solid-vapor mixture. It is for this reason that CO\(_2\) exists as a ”Dry Ice” at atmospheric pressure and temperature. For this analysis, the downstream injector pressure used is 85.9 kPa; the atmospheric pressure at the approximate altitude of the test facility located in Logan, Utah (4500 feet above sea level). Since the downstream pressure used is below the triple point temperature of CO\(_2\), the fluid near the exit of the injector orifice is likely a solid gas mixture.

### 3.4 Mass Flow Rate Prediction

The injector mass flow rate is calculated using the NHNE model discussed earlier and defined in Eq. 2.21. From the mass flow rate, the total enthalpy flow rate is found to be:

\[
\dot{H} = h_1 \cdot \dot{m}_{NHNE} \tag{3.13}
\]

where \(\dot{H}\), is the total enthalpy flow rate.

### 3.5 Update Fluid Properties

To update the total tank fluid mass \(M_{i+1}\), and total tank fluid enthalpy \(H_{i+1}\), an Euler integration method is used. The flow rates calculated in Eqs. 2.21 and 3.13, are multiplied
by the time step $\Delta t$ and subtracted from the current total mass $M_i$, and total enthalpy $H_i$:

$$M_{i+1} = M_i - \dot{m}_{NHNE} \cdot \Delta t \quad (3.14)$$

$$H_{i+1} = H_i - \dot{H} \cdot \Delta t \quad (3.15)$$

The tank fluid total density and total specific enthalpy are updated from the new total fluid mass and total tank enthalpy:

$$\rho_{i+1} = \frac{M_{i+1}}{V_{tank}} \quad (3.16)$$

$$h_{i+1} = \frac{H_{i+1}}{M_{i+1}} \quad (3.17)$$

Once again the MATLAB `lsqnonlin` function is used to calculated the remaining fluid properties. In this case, one of each of the input and output properties of Eq. 3.6 are known. Equation 3.18 is solved using `lsqnonlin` with $h_1$ and $\rho_1$ as knowns with everything else unknown.

$$[P, \chi, h_{known}, s] = f(T, \rho_{known}) \quad (3.18)$$

$$[T_1] = \text{lsqnonlin}( h(T, \rho_1) - h_1 ) \quad (3.19)$$

In this case, the `lsqnonlin` function in Eq. 3.19 returns the upstream temperature $T_1$, that equates to the known density and enthalpy. Example results are shown in Figure 3.2. Here the unknown temperature is plotted against the known total density for different specific enthalpies. All indices are moved forward in time, and the loop is iterated forward in time.
Fig. 3.2: CO$_2$ temperature as a function of density for different specific enthalpies.
Chapter 4
Experimental Validation

To validate the model developed here, it is compared against cold flow data derived from laboratory tests performed at Utah State University. Figure 4.1 shows a piping and instrumentation diagram (P&ID) of the oxidizer flow test stand designed and built to facilitate hybrid rocket motor static firings. The stand has more capability than is shown here, including the ability to pressurize the run tank from the top with gaseous nitrogen or any other desired inert gas.

4.1 Experimental Apparatus

The run tank is constructed from a flat bottomed aluminium cylinder of size K with a total internal volume of 49.9 L (1.76 \( ft^3 \)). The flat bottom is ported to accommodate a pressure transducer, vent valve, and relief valve. In the test configuration, the bottle has been turned upside down, fitted with clamp rings and hung by two s-beam load cells. Insulation has been wrapped around the bottle to help reduce heat transfer during blowdown. The main port on the tank is fitted with a tee so that a 0.125 inch diameter thermocouple probe can be inserted into the tank. The second port on the tee has a flex line attached to help obtain the best possible tank weight measurement from the load cells during blowdown. The other end of the flex line is equipped with a pneumatically operated fill valve, used when filling the tank with fluid. After the fill valve there is a diverging-converging venturi fitted with two pressure transducers to measure mass flow rate. The first pressure transducer port leads to the inlet of the venturi prior to the divergent section, the second transducer port leads to the throat area of the venturi between the divergent and convergent sections. The outlet of the venturi flows into an additional pneumatically operated run valve, and finally into an orifice style injector.
Fig. 4.1: Cold flow blowdown physical setup.
4.2 Experimental Procedures

To perform a cold flow blowdown test where the self-pressurized fluid is vented to ambient conditions, saturated CO\textsubscript{2} or N\textsubscript{2}O is let into the tank through the fill valve. To prevent the saturated CO\textsubscript{2} or N\textsubscript{2}O from flashing to vapor when first introduced into the tank, the tank and plumbing are pressurized with gaseous nitrogen to slightly above the saturation pressure of the CO\textsubscript{2} or N\textsubscript{2}O. Additionally, the CO\textsubscript{2} or N\textsubscript{2}O source tanks are pressurized with gaseous nitrogen prior to beginning the fill process. To achieve the highest total oxidizer fluid mass possible in the run tank, the tank should be filled with mostly liquid. To begin the filling process, the vent and fill valves are opened and the run tank is filled with saturated liquid from the bottom while gaseous nitrogen and CO\textsubscript{2} or N\textsubscript{2}O vapor is vented from the top. If the pressure in the tank drops too quickly due to the venting, it is sometimes necessary to pulse the vent valve to minimize the amount of saturated CO\textsubscript{2} or N\textsubscript{2}O that flashes to vapor. Once the run tank has been filled with the desired amount of fluid, the blowdown test is performed. With the vent valve closed, the run valve is opened and the fluid in the run tank is vented to ambient conditions through the single injector. As saturated liquid exits the run tank at the bottom, there is also saturated liquid flashing to saturated vapor in the upper portion of the tank. There reaches a point during the blowdown when all of the liquid has been blown out and the remainder of the blowdown vents vapor only.

The data gathered during a blow down that is applicable to this model is the run tank pressure, the fluid temperature in the bottom portion of the tank, the tank mass, and the venturi pressures. Since the pressure transducer is located on the port in the top of the tank, the pressure it measures is that of the warmest fluid within the tank. Neglecting the hydrostatic portion of the pressure, the pressure is constant throughout the tank. The temperature measurement however is likely not as reliable throughout the tank as the pressure measurement is. In fact, placing thermocouples at different locations along the length of the outside of the run tank shows that temperature stratification does occur after loading and during blowdown. Since the temperature is a measurement of the cooler liquid
in the tank bottom, and the pressure is a measurement of the warmer vapor in the tank top, picking one over the other to initialize the fluid state with, yields slightly different results. For convenience, the model presented here is initialized from the temperature measurement, for cases in which a pressure measurement initialization is desired, the initial temperature is increased to match that corresponding to the initial pressure. The temperature stratification could be minimized by allowing enough time for the fluid temperature to equalize to the ambient conditions prior to performing a blowdown test.

The venturi pressures are used to calculate the mass flow rate of fluid exiting the tank. From the load cell data, the mass of fluid in the tank is known throughout the blowdown process. The numerical derivative of the fluid mass is calculated to also obtain mass flow rate from the tank. This massflow estimate, although somewhat noisy, is used to support the venturi massflow measurements.

4.3 Model Comparisons

Figures 4.2, 4.3, 4.4, and 4.5 show typical blowdown test results and compare the sensed tank pressure, temperature and exit massflow to the NHNE model predictions. The model was initialized using the initial temperature measured in the tank as well as the initial fluid mass. From examination of figures 4.3 and 4.4 the initial conditions are apparent as the temperatures and fluid mass agree at time zero. At time zero the model calculates a fluid pressure from the initial density and temperature using equations 3.1 and 3.6. Examining figure 4.2 shows an almost 0.5 MPa difference between the calculated initial pressure and the measured initial pressure. From this, it is apparent that there is temperature stratification in the tank even before the blowdown has begun. The initial measured pressure is a measurement of the warmest vapor in the top of the tank, while the initial measured temperature is a measurement of the cooler liquid in the bottom of the tank. Further evidence of temperature stratification is seen by looking at the cold flow temperature data in figure 4.3. For the first few seconds of the blowdown, the measured temperature increases before it starts to decrease. This temperature rise suggests that the fluid in the top of the tank is warmer than in the fluid in the bottom. The opening of the run valve has a mixing effect
in the tank, pulling warmer fluid from the top of the tank towards the bottom, causing the measured temperature to rise. The NHNE model as developed cannot account for this temperature stratification and subsequent mixing.

Examination of figures 4.4 and 4.5 show that the NHNE model does a good job of predicting the steady-state mass flow rate when compared to the incompressible and HEM models alone. The incompressible fluid model greatly over predicts the mass flow rate while the HEM model greatly under predicts it. The inflection points on each of the plots at approximately 20 seconds, represent the point during the blowdown where all of the liquid in the tank is exhausted and the flow transitions to purely vapor. Measured mass flow rates at the very beginning of blowdown is not predicted well by the model. This is because the plumbing between the run valve and injector is initially full of air. At time zero when the run valve is opened, air in the plumbing is forced out through the injector and fluid is flashing to vapor in the plumbing. Until fluid fills the plumbing between the run valve and injector, the venturi returned mass flow rate will be higher than predicted by the model.

Fig. 4.2: Comparison, Model vs cold flow, run tank fluid pressure.
Fig. 4.3: Comparison, Model vs cold flow, run tank fluid temperature.

Fig. 4.4: Comparison, Model vs cold flow, run tank total fluid mass from load cells.
Additionally, the venturi measured mass flow rate as plotted, is not valid once the flow transitions from liquid to vapor. It can be seen in figure 4.5 that this is the case as the venturi mass flow rate starts to drop at the transition point, but then quickly starts giving erroneous values. This inaccuracy results from the equation used to calculate the venturi flow rate, $\dot{m}_{\text{venturi}}$, from the venturi inlet pressure $P_{\text{in}}$, and venturi throat pressure $P_t$, is:

$$\dot{m}_{\text{venturi}} = A_t \cdot C_d \cdot \rho \cdot \sqrt{\frac{2(P_{\text{in}} - P_t)}{\rho(1 - (A_t/A_{\text{in}})^2)}}$$  \hspace{1cm} (4.1)$$

In equation 4.1, $P_{\text{in}}$ and $P_t$ are the measured inlet and throat pressures, $A_{\text{in}}$ and $A_t$ are the inlet and throat cross sectional areas, $C_d$ is the experimentally determined discharge coefficient, and $\rho$ is the density of the fluid flowing through the venturi. For the cold flow data, $\rho$ is calculated from a second order polynomial fit that is a function of the measured tank temperature:

$$\rho = -0.136929 \cdot T^2 + 68.960274 \cdot T - 7677.810286$$  \hspace{1cm} (4.2)$$
The curve fit method of Eq. 4.2 is used to significantly reduce the run time of the code developed to parse the cold flow data during repetitive analysis. Equation 4.2 is plotted against the previously shown Helmholtz liquid density in figure 4.6. Figure 4.6 shows that the curve fit polynomial does a good job in predicting the liquid density between 260 and 300 K, but deviates significantly once the temperature drops below 260 K. Thus as the liquid is depleted from the tank and the temperature drops, the venturi massflow measurements become increasingly inaccurate. For typical blowdown tests, where the liquid portion of the blowdown is what is of interest, the fluid temperature during this time stays between 260 and 300 K. Figure 4.7 depicts the venturi mass flow rate of the cold flow blowdown case presented here as calculated by Eq. 4.1, where the liquid density is calculated by both the Helmholtz energy method presented in Eq. 2.10 and the curve fit method of Eq. 4.2. The plot shows that for a typical blowdown test where the fluid temperature stays in the range of accuracy, that the curve fit polynomial does a good job of calculating the fluid liquid density.
In addition to venturi derived mass flow rate, the load cell data was numerically differentiated to obtain mass flow rate. Figure 4.8 depicts the tank fluid mass zoomed in time to the first five seconds of the blowdown. Within the first second, the load cell data decreases as expected, but then makes a sudden leap before continuing to decline. This unexpected mass increase is due to the tank slightly jumping on its load cells from the sudden rush of fluid from the bottom of the tank when the run valve is opened. This "noise" causes problems for the numerical differentiation. To minimise this, a differentiation method based on discrete wavelet transformations presented by Luo [13] is used. When a wavelet function is the derivative of a smoothing function, the wavelet transformation has the combined properties of smoothing and differentiation [14]. Figure 4.9 shows the improvement gained by using the wavelet transformation method as opposed to a simple difference approach. The wavelet method significantly improves the derived mass flow rate, and likely best captures the initially high mass flow rate caused by the plumbing volume filling with fluid.
Fig. 4.8: Model vs cold flow, fluid mass flow rate comparison, detailed initial time.

Fig. 4.9: Numerical differentiation methods comparison, cold flow mass flow rates.
Chapter 5

Conclusions

Nitrous oxide is an attractive oxidizer due to its high vapor pressure at room temperature. Because of this property, nitrous oxide can be used to self pressurize eliminating the need for more complex and costly propellant delivery systems. Nitrous oxide is also relatively safe to work with as it is non toxic, non corrosive, and relatively non reactive. N$_2$O is however not a simple fluid to model because of its non ideal fluid properties in its saturated state. These non-ideal properties make incompressible liquid and ideal gas assumptions invalid and requires a more complex two-phase model to predict performance. N$_2$O has an additional advantage in that its performance and properties are closely mirrored by a more widely available and even safer fluid. CO$_2$ exhibits similar behaviour to N$_2$O and is therefore used as a non energetic checkout fluid for N$_2$O systems. In order to develop a model to predict the performance of N$_2$O or CO$_2$ in an oxidizer system, some key concepts are introduced and an algorithm to calculate mass flow rate is assembled.

The fluid properties of N$_2$O and CO$_2$ are fetched in two ways yielding the same results. The first method evaluated was a simple table lookup of values published in the on-line NIST Webbook database. The second method involved building a function around a helmholtz free energy based set of equations.

Two independent methods of mass flow rate prediction have been discussed and another authors assembly of these models was presented in the form of the weighted non-homogeneous non-equilibrium (NHNE) model. An analysis of the NHNE model was performed and a correction was proposed. The proposed correction involved swapping the weighting coefficients so that the intended outcome of the model was realized.

An enthalpy based blowdown algorithm was presented based around the corrected NHNE model. The purpose of the blowdown algorithm was to predict the mass flow rate of
saturated N_2O or CO_2 from a storage tank through an orifice style injector. The algorithm loops through time updating the tank fluid properties by recalculating them each time step from the total enthalpy of the fluid in the tank. The injector downstream fluid properties are calculated using either an isentropic or adiabatic injector assumption.

The enthalpy based blowdown algorithm is compared against experimentally obtained data from a cold flow blowdown of CO_2. The NHNE enthalpy algorithm does an excellent job of predicting the mass flow rate of the fluid through the injector as compared to the alternatives. However, experimental conditions resulting from the test apparatus set-up produces fluid stratification and mixing effects that cannot be modelled by the algorithm as developed. Thus the run tank and temperature drops as predicted by the model are significantly larger than measured. The experimental data could be improved by allowing the fluid in the run tank to reach a homogeneous temperature, prior to performing blow down. To improve the models ability to predict the run tank temperature and pressure, the run tank could be conceptually split into vertical sections with the fluid properties tracked from top to bottom. Additionally, energy could be added to the tank fluid over time to model the heat transfer that occurs from the tank mass to the fluid mass throughout the blow down.

Naturally the next step would be to predict the mass flow of N_2O into a hybrid rocket motor combustion environment. It is expected that the model developed here would perform well in this application, because there is a lower pressure differential across the injector than was accurately modelled here. Overall, the model developed here is a valuable addition to the rocket system designers tool set. It does an excellent job of performing its primary function by accurately predicting the mass flow rate of a self-pressuring fluid.
References


Appendices
Appendix A

Fluid Constants
### A.1 Comparison of CO$_2$ and N$_2$O Properties

<table>
<thead>
<tr>
<th>Fluid</th>
<th>CO$_2$</th>
<th>N$_2$O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Molar Mass</td>
<td>44.01 kg/kmol</td>
<td>44.013 kg/kmol</td>
</tr>
<tr>
<td>Gas Constant</td>
<td>0.1889241 kJ/(kg*K)</td>
<td>0.1889 kJ/(kg*K)</td>
</tr>
<tr>
<td>T triple</td>
<td>216.592 K</td>
<td>182.340 K</td>
</tr>
<tr>
<td></td>
<td>-69.804 °F</td>
<td>-131.458 °F</td>
</tr>
<tr>
<td>T critical</td>
<td>304.128 K</td>
<td>309.520 K</td>
</tr>
<tr>
<td></td>
<td>87.761 °F</td>
<td>97.500 °F</td>
</tr>
<tr>
<td>P triple</td>
<td>0.51795 MPa</td>
<td>0.08785 MPa</td>
</tr>
<tr>
<td></td>
<td>75.122 psi</td>
<td>12.742 psi</td>
</tr>
<tr>
<td>P critical</td>
<td>7.3773 MPa</td>
<td>7.245 MPa</td>
</tr>
<tr>
<td></td>
<td>1070.0 psi</td>
<td>1050.8 psi</td>
</tr>
<tr>
<td>$\rho$ critical</td>
<td>467.5 kg/m$^3$</td>
<td>452 kg/m$^3$</td>
</tr>
<tr>
<td></td>
<td>29.185 lb/ft$^3$</td>
<td>28.22 lb/ft$^3$</td>
</tr>
<tr>
<td>Compressibility Factor</td>
<td>Saturated Liquid - 0.16 @ 298 K</td>
<td>Saturated Liquid - 0.14 @ 298 K</td>
</tr>
<tr>
<td>Compressibility Factor</td>
<td>Saturated Vapor - 0.47 @ 298 K</td>
<td>Saturated Vapor - 0.53 @ 298 K</td>
</tr>
<tr>
<td>No Liquid If</td>
<td>$P &lt; P_{\text{triple}}$ or $T &lt; T_{\text{triple}}$</td>
<td>$P &lt; P_{\text{triple}}$ or $T &lt; T_{\text{triple}}$</td>
</tr>
<tr>
<td>Melting Point</td>
<td>216.59 K</td>
<td>182.29 K</td>
</tr>
<tr>
<td></td>
<td>-69.81 °F</td>
<td>-131.50 °F</td>
</tr>
<tr>
<td>Sublimation Point</td>
<td>194.75 K</td>
<td>184.67 K</td>
</tr>
<tr>
<td></td>
<td>-109.12 °F</td>
<td>-127.3 °F</td>
</tr>
<tr>
<td>Boiling Point</td>
<td>194.75 K</td>
<td>184.67 K</td>
</tr>
<tr>
<td></td>
<td>-109.12 °F</td>
<td>-127.3 °F</td>
</tr>
<tr>
<td>P vapor</td>
<td>5.72 MPa @ 293.15 K</td>
<td>5.15 MPa @ 298.15 K</td>
</tr>
<tr>
<td></td>
<td>829.626 psi @ 68 °F</td>
<td>746.94 psi @ 68 °F</td>
</tr>
<tr>
<td>Ref. Int. Energy</td>
<td>$U = 0$ at 273.16 K for saturated liquid</td>
<td>$U = 0$ at 273.16 K for saturated liquid</td>
</tr>
<tr>
<td>Ref. Entropy</td>
<td>$S = 0$ at 273.16 K for saturated liquid</td>
<td>$S = 0$ at 273.16 K for saturated liquid</td>
</tr>
</tbody>
</table>
Appendix B
Matlab Code

B.1 Blow Down Simulation

```matlab
clear all; clc;
close all;

% Initial Conditions
%==========================================================================
% Tank Properties
%−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
V = .0498; % Tank volume, m3
M_o = 12.76; % Initial tank fluid mass, kg
T1_o = 274.25; % Initial tank fluid temperature, K
%M_o = 10.15;
%M_o = 10.15;
%P_o = 4.025; % Initial tank fluid pressure, MPa
%X_o = 0.01; % Initial tank fluid quality,
R = 0.1889241; % Gas constant, kJ/(kg*K)

% Injector Properties
%==========================================================================
d_inj = 0.178; % Injector diameter, in
Ac = (pi/4)*(d_inj*.0254)^2; % Injector cross sectional area; m3
Cd = 0.8; % Injector discharge coefficient
Pamb = 85.9e-3; % Atmospheric pressure, MPa. (4500ft)
Inj_Switch = 1; % Isentropic = 1, Adiabatic = 2

% Time Iteration
%==========================================================================
tstop = 30; % Stop time, s
dt = .5; % Step time, s
```
% Initialize state

% --------------------------------------------------------------
rho1_o = M_o/V
Props1_o = CO2Props(T1_o,rho1_o);
%Props1_o = CO2PropsNIST(T1_o,rho1_o);
P1_o = Props1_o.P;
rhol_o = Props1_o.rho_l;
rhov_o = Props1_o.rho_v;
X1_o = (rhov_o/rho1_o)*(rhol_o-rho1_o)/(rhol_o-rhov_o);
h1_o = Props1_o.h;
H1_o = M_o*Props1_o.h;
st_o = Props1_o.state;

% Column 1: Time (sec)
M = M_o;
% Column 2: Tank fluid mass, M (kg)
rhol = rho1_o;
% Column 3: Tank fluid density, rho1 (kg/m^3)
T1 = T1_o;
% Column 4: Tank temperature, T1 (K)
P1 = P1_o;
% Column 5: Tank pressure, P1 (MPa)
X1 = X1_o;
% Column 6: Tank quality, X1 ()
h1 = h1_o;
% Column 7: Tank specific enthalpy, h1 (kJ/kg)
H1 = H1_o;
% Column 8: Tank total enthalpy, H1 (J) ?? Units
mdot = 0;
% Column 9: Tank mass flow rate, mdot (kg/s)
P2 = Pamb;
% Column 10: Injector outlet pressure, P2 (Pa)
st = st_o
% Column 11: Fluid state, -1— Input, 0=Liq, 1=Sat, 2=Gas

State = [t, M, rho1, T1, P1, X1, h1, H1, mdot, P2, st];
State2 = [0, 0];

% --------------------------------------------------------------
i = 0;
while t < tstop
i = i + 1

% Exit the loop when the tank is out of fluid
if M < 0; break; end

% Initial Tank Fluid Properties
%======================================================================
% Note: this guess MUST yield a quality less than 1 and >0
guess=[300 300]; %[T,rho]

% Set up function (need to match pressure and quality)
pFunc = @(v) [getfield(CO2Props(v(1),v(2)),'P')-P1; ...
             getfield(CO2Props(v(1),v(2)),'X')-X1];
%
% Solve for [T,rho] of saturated but pure liquid at P1
% lsqnonlin tries to find a T and rho that makes pFunc = 0
v1 = ...
lsqnonlin(pFunc,guess,0,inf,optimset('Display','off','TolFun',1e-14));
T1 = v1(1); rho1 = v1(2);
Props1 = CO2Props(T1,rho1);

% Propegate properties across the injector
%======================================================================
%Assume an isentropic (s1=s2) or adiabatic (h1=h2) injector, solve ...

for properties at P2

if Inj_Switch == 1

% Set up function to match pressure and entropy
%
pFunc = @(v) [getfield(CO2Props(v(1),v(2)),'P')-P2 ...
getfield(CO2Props(v(1),v(2)),'s')-s1];

elseif Inj_Switch == 2

% Set up function to match pressure and enthalpy
pFunc = @(v) [getfield(CO2PropsNIST(v(1),v(2)),'P')-P2 ...
getfield(CO2PropsNIST(v(1),v(2)),'h')-h1];
end

% Solve for T2 & rho2 downstream of the injector
%
guess=[300 300]; %[T,rho]
v2 = ...
lsqlnonlin(pFunc,guess,0,inf,optimset('Display','off','TolFun',1e-14));
T2 = v2(1); rho2 = v2(2);

% Solve for the remaining properties downstream of the injector
%
Props2=CO2Props(T2,rho2); % Downstream fluid properties
Pv2 = Props2.P; % Downstream fluid vapor pressure
h2 = Props2.h; % Downstream fluid enthalpy

% Calculate the Mass Flow
%======================================================================
% Non-Equalibrium Parameter
%k = sqrt(abs(P1-P2)/abs(Pv2-P1)); % Whitmores Equation
k = sqrt((P1-P2)/(Pv1-P2)); % Spencer and Stanfords Equation
% Weighting Coefficient
\( W = \frac{1}{k+1}; \)

% Incompressible fluid mass flow rate
\( \text{mdot}_{\text{inc}} = A_c \sqrt{2 \rho L_1 (P_1 - P_2) \times 1e6}; \)

% Homogeneous Equilibrium mass flow rate
\( \text{mdot}_{\text{HEM}} = \rho_2 A_c \sqrt{2 (h_1 - h_2)}; \)

% Weighted Non-Homogeneous Equilibrium (modified Stanford) mass flow rate
\( \text{mdot} = C_d ((1 - W) \times \text{mdot}_{\text{inc}} + W \times \text{mdot}_{\text{HEM}}); \quad \% \text{Shannon's Theory} \)

% Update the upstream fluid properties for the next step
%======================================================================
\( M = M - \text{mdot} \times dt; \quad \% \text{Update tank fluid mass} \)
\( \text{Hdot} = h_1 \times \text{mdot}; \quad \% \text{Enthalpy flow rate} \)
\( \text{H1} = \text{H1} - \text{Hdot} \times dt; \quad \% \text{Update tank total enthalpy} \)

% Calculate the new tank enthalpy and density
%---------------------------------------------------------------
\( \rho_1 = \frac{M}{V}; \quad \% \text{Update tank specific density} \)
\( h_1 = \frac{H1}{M}; \quad \% \text{Update tank specific enthalpy} \)

% Calculate the new tank temperature
%---------------------------------------------------------------
% Create a function for lsqnonlin to solve T(rho,h)
\( \text{pFunc} = @(T_{\text{Unknown}}) \text{getfield(CO2Props(T_{\text{Unknown}},\rho_1),'h')}-h_1; \)
% \( \text{pFunc} = @(T_{\text{Unknown}}) \text{getfield(CO2PropsNIST(T_{\text{Unknown}},\rho_1),'h')}-h_1; \)
% Since T_{\text{Unknown}} is not pre defined in pFunc, lsqnonlin will find a ...
\( T_1 = \text{lsqnonlin}([\text{pFunc},300,0,\text{inf},\text{optimset('Display','off','TolFun',1e-14)}]); \)
% \( [T_1] = \text{CO2\_rho\_h\_2T}(\rho_1,h_1,300); \)

% Calculate the new tank pressure and quality
%---------------------------------------------------------------
\( \text{Props1} = \text{CO2Props}(T_1,\rho_1); \)
% \( \text{Props1} = \text{CO2PropsNIST}(T_1,\rho_1); \)
\( P_1 = \text{Props1.P}; \)
\( X_1 = \text{Props1.X}; \)
st1 = Props1.state;

% Update the state
%-----------------------------------------------------
t = t + dt;

State = [State; t, M, rho1, T1, P1, X1, h1, H1, mdot, Pamb, st1];
State2 = [State2; mdot_inc, mdot_HEM];
end

% Open cold flow data for comparison
%==========================================================================
CF = open('Cold Flow Data\FillTest1.29_largeNoTop.mat');
%CF = open('Cold Flow Data\FillTest1.29_smallNoTop.mat');
%CF = open('Cold Flow Data\FillTest1.29_large.mat');
%CF = open('Cold Flow Data\FillTest1.29_small.mat');
% Find portion of cold flow data where the run valve is open
%----------------------------------------------------------
iter = 0;
for loop = 1:length(CF.Valvedat)
    if CF.Valvedat(loop,5) == 1
        iter = iter + 1;
        CF_t(iter) = CF.t(loop);
        CF_Weight(iter) = CF.Weight(loop);
        CF_PTank(iter) = CF.PRunTank(loop);
        CF_Mdot(iter) = CF.mdotVentIn(loop);
        CF_DeltaWeight(iter) = CF.DeltaWeightKg(loop);
        CF_T_Tank(iter) = CF.Tank_Temp(loop);
        CF_dP_Vent(iter) = CF.dpVenturi(loop);
        CF_rhol_H(iter) = CF.rhol_H(loop);
    end
end
CF_t = CF_t - CF_t(1);
% Calculate the venturi mass flow rate
\%--------------------------

d1=0.0254762;
d2=0.0078486;
A1=pi*d1^2/4;
A2=pi*d2^2/4;
C=0.980243097172394;

\text{CF}_\text{Mdot}_H = A2 \times C \times \text{CF}_rhol_H \times \sqrt{2 \times (\text{CF}_dP_{\text{Vent}}) / (\text{CF}_rhol_H \times (1 - (A2/A1)^2))};

% Calculate the mass flow rate by differentiating the tank mass
%-----------------------------------------------
\text{CF}_\text{Mdot}_\text{diff} = -\text{diff(CF}_\text{Weight}) / \text{diff(CF}_t);
\text{CF}_\text{dt} = \text{CF}_t(2) - \text{CF}_t(1);
%\text{CF}_\text{Mdot}_\text{wave} = \text{derivative_cwt(CF}_\text{Weight},'gaus1',16,\text{CF}_\text{dt},1);
\text{CF}_\text{Mdot}_\text{wave} = \text{derivative_dwt(CF}_\text{Weight},'spl',4,\text{CF}_\text{dt},1);

% Open cold flow data for comparison
%=======================================
\text{Whitmore} = \text{open('Whitmore Output\FillTest1_29_largeNoTop.mat')};
\text{W}_t = \text{Whitmore.t};
\text{W}_M = \text{Whitmore.M};
\text{W}_\text{rhol} = \text{Whitmore.rhol};
\text{W}_T1 = \text{Whitmore.T1};
\text{W}_P1 = \text{Whitmore.P1};
\text{W}_X1 = \text{Whitmore.X1};
\text{W}_H1 = \text{Whitmore.H1};
\text{W}_\text{mdot} = \text{Whitmore.mdot};
\text{W}_2 = \text{Whitmore.P2};

% Up-pack the state vector for plotting
%========================================
\text{t} = \text{State(:,1)}; \text{\% Column 1: Time (sec)}
\text{M} = \text{State(:,2)}; \text{\% Column 2: Tank fluid mass, M (kg)}
\text{rhol} = \text{State(:,3)}; \text{\% Column 3: Tank fluid density, rhol (kg/m^3)}
\text{T1} = \text{State(:,4)}; \text{\% Column 4: Tank temperature, T1 (K)}
P1 = State(:,5); % Column 5: Tank pressure, P1 (Pa)
X1 = State(:,6); % Column 6: Tank quality, X1 ()
h1 = State(:,7); % Column 7: Tank specific enthalpy, h1 (kJ/kg)
H1 = State(:,8); % Column 8: Tank total enthalpy, H1 (J) ??? Units
mdot = State(:,9); % Column 9: Tank mass flow rate, mdot (kg/s)
P2 = State(:,10); % Column 10: Injector outlet pressure, P2 (Pa)
st = State(:,11); % Column 11: Fluid state, −1=− Input, 0=Liq, ...

1=Sat, 2=Gas

mdot_inc = State2(:,1); % Mdot Incompressible
mdot_HEM = State2(:,2); % Mdot HEM

%% Plot
%==========================================================================
figure(2)
plot(t,M); hold on; grid on;
plot(W,t,W_M,'--g');
plot(CF,t,CF_Weight,'--r');
plot(CF,t,CF_DeltaWeight,'g');
%title('Tank Mass');
xlim([0 tstop]);
xlabel('Time, s'); ylabel('Tank Fluid Mass, kg');
legend('Model Predicted','Cold Flow');
saveas(2,'Tank_Mass.png')

%% Column 3 — Tank fluid specific density
%==========================================================================
figure(3);
plot(t,rh1); hold on; grid on;
plot(W,t,W_rh1,'--g');
plot(CF,t,CF_Weight/V,'r');
title('rh1');
xlabel('Time, s'); ylabel('Density, kg/m3');
legend('Model Predicted','Whitmore','Cold Flow');
saveas(3,'Tank_Density.png')

% Column 4 - Tank temperature
figure(4);
plot(t,T1); hold on; grid on;
%plot(W_t,W_T1,'--g');
plot(CF_t,CF_Tank,'--r');
%title('Temperature');
xlim([0 tstop]);
xlabel('Time, s'); ylabel('Temperature, K');
legend('Model Predicted','Cold Flow');
saveas(4,'Tank_Temperature.png')

% Column 5 - Tank pressure
figure (5)
plot(t,P1); hold on; grid on;
%plot(W_t,W_P1/1e3,'--g');
plot(CF_t,CF_PTank/1e6,'--r');
%title('P1');
xlim([0 tstop]);
xlabel('Time, s'); ylabel('Pressure, MPa');
legend('Model Predicted','Cold Flow');
saveas(5,'Tank_Pressure.png')

% Column 6 - Tank quality
figure (6)
plot(t,X1); hold on; grid on;
plot(W_t,W_X1,'--g');
ylim([0 1]);
title('X1');
xlabel('Time, s'); ylabel('Quality');
legend('Model Predicted','Whitmore');
saveas(6,'Tank_Quality.png')

% Column 7 — Tank specific enthalpy

figure (7)
plot(t,h1); hold on; grid on;
title('h1');
xlabel('Time, s'); ylabel('Specific Enthalpy, kJ/kg-K');
legend('Model Predicted');
saveas(7,'Tank_Specific_Enthalpy.png')

% Column 8 — Tank total enthalpy

figure (8)
plot(t,H1); hold on; grid on;
plot(Wt,W_H1,'--g');
title('H1'); xlabel('time, s'); ylabel('Enthalpy');
legend('Model Predicted','Whitmore');
saveas(8,'Tank_Total_Enthalpy.png')

% Column 9 — Mass flow rate

figure(9);
plot(t,mdot,'b'); hold on; grid on;
plot(CF_t,CF_Mdot,'--r')
plot(CF_t,CF_Mdot_H,'-.b')
xlabel('Time, s'); ylabel('Mass Flow Rate, kg/s');
xlim([0 tstop]); ylim([0 3]);
legend('Model Predicted','Venturi-Curve Fit Density','Venturi-Helmholtz ... 
Density');
saveas(9,'Tank_Mass_Flow_Rate_Venturi.png')
```matlab
figure(90);
plot(t,mdot,'b'); hold on; grid on;
%plot(W,T,W_mdot,'--g');
plot(CF_t,CF_Mdot,'--r')
plot(CF_t,-CF_Mdot_wave,'--b')
plot(t,mdot_inc,:b);
plot(t,mdot_HEM,'--bo');
xlabel('Time, s'); ylabel('Mass Flow Rate, kg/s');
xlim([0 tstop]); ylim([0 1.6]);
legend('Model Predicted','Cold Flow-Venturi','Cold Flow-Differentiated ...
Load','Incompressible Model','HEM Model');
saveas(90,'Tank_Mass_Flow_Rate.png')

figure(900);
plot(CF_t(2:end),CF_Mdot_diff); hold on; grid on;
plot(CF_t,-CF_Mdot_wave,'--r')
xlabel('Time, s'); ylabel('Mass Flow Rate, kg/s');
xlim([0 tstop]); ylim([-2 4]);
legend('Cold Flow-dM/dt-diff','Cold Flow-dM/dt-wavelet');
saveas(900,'Tank_Mass_Flow_Rate.Diff.png')

% Column 11 - Tank fluid state
%---------------------------------------------------------------------
figure(11);
plot(t,st); grid on;
title('Fluid State');
xlabel('Time, s'); ylabel('Fluid State');
ylim([-2 3]);
legend('Model Predicted');
saveas(11,'Tank_Fluid_State.png')
```

**B.2 CO₂ Properties - Helmholtz Energy Method**
function [Props] = CO2Props(T,rho)

% INTRODUCTION

% Purpose:
% Returns a structure containing the thermodynamic properties for carbon
dioxide (CO2) for a given density (kg/m^3) and temperature (K). Valid
% temperature and pressure ranges are:
% 216 K ≤ T ≤ 1100 K 0 MPa ≤ P ≤ 800 MPa

% The properties in the structure are:

% P  X  s  u  cv  cp  h  c
% rho_l  s_l  u_l  cv_l  cp_l  h_l  c_l
% rho_v  s_v  u_v  cv_v  cp_v  h_v  c_v

% Inputs:
% T  —  Temperature, K
% rho  —  Density, kg/m3

% Outputs:
% P  —  Pressure, MPa
% X  —  Quality, (Vapor Mass/Total Fluid Mass)
% s  —  Specific Entropy, kJ/(kg*K)
% u  —  Specific Internal Energy, kJ/kg
% cv  —  Specific Heat at Constant Volume, kJ/(kg*K)
% cp  —  Specific Heat at Constant Pressure, kJ/(kg*K)
% h  —  Specific Enthalpy, kJ/kg
% c  —  Speed of Sound, m/s
% rho  —  Density, kg/m3
% state  —  −1 = Negative Input, 0 = Liquid, 1 = Saturated, 2 = Gas
% _l  —  Liquid designator
% _v  —  Vapor designator

% Revision History:
% Written for a CO2 Blowdown model developed at Utah State University by
% Matthew Wilson
% 4130 Old Main Hill
% Logan, UT 84322-4130
% Recommended and checked by:
% Brian Solomon

% Based upon the Helmholtz Energy based equations of state described by
% Span, R. and Wagner, W. in
% "A New Equation of State for Carbon Dioxide Covering the Fluid Region
% from the Triple-Point Temperature to 1100 K at Pressures up to 800 MPa"
% Journal of Physical and Chemical Reference Data
% Vol 25, No. 6, 1996. Pp 1509-1596

%% CALCULATE THE SATURATION PROPERTIES
%==========================================================================
% CO2 Constants
%==========================================================================
R = 0.1889241; % Gas constant, kJ/(kg*K)
Tt = 216.592; % Triple point temperature, K (Eq. 3.1)
Pt = 0.51795; % Triple point pressure, MPa (Eq. 3.2)
Tc = 304.1282; % Critical temperature, K (Eq. 3.3)
Pc = 7.3773; % Critical pressure, MPa (Eq. 3.4)
rhoc = 467.6; % Critical density, kg/m3 (Eq. 3.5)
d = rho/rhoc; % Reduced density, \( \Delta = \phi/\phi_{critical} \), \( \phi = \) mass density

t = Tc/T; % Inverse reduced temperature, \( \tau = T_{critical}/T \), \( T \) ...
            - temperature

% Coefficients
%==========================================================================

% Melting pressure coefficients, Section 3.3
a_m = [1955.5390 2055.4593]';

% Sublimation pressure coefficients, Section 3.4
% Vapor pressure coefficients, Section 3.5
a_s = [-14.740846 2.4327015 -5.3061778]';

% Saturated liquid density coefficients, Section 3.6
a_l = [1.9245108 -0.62385555 -0.32731127 0.39245142]';

% Saturated vapor density coefficients, Section 3.7
a_v = [-1.7074879 -0.82274670 -4.6008549 -10.111178 -29.742252]';

t_p = [1.0 1.5 2.0 4.0]';
t_l = [0.34 0.5 10/6 11/6]';
t_v = [0.34 0.5 1.0 7/3 14/3]';

% Phase Property Calculations

% P_melt = Pt* (1 + a_m(1)*(T/Tt-1) + a_m(2)*(T/Tt-1)^2); % Melting Pressure, Eq. 3.10
% P_sub = Pt*exp(Tt/T*(a_s(1)*(1-T/Tt) + ... % Sublimation Pressure, Eq. 3.12
% a_s(2)*(1-T/Tt)^1.9 + ... % a_s(3)*(1-T/Tt)^2.9));
P_sat = Pc * exp(Tc/T*sum(a_p.*(1-T/Tc).^t_p)); % Vapor Pressure, Eq. 3.13
rho_l = rhoc * exp(sum(a_l.*(1-T/Tc).^t_l)); % Saturated Liquid Density, Eq. 3.14
rho_v = rhoc * exp(sum(a_v.*(1-T/Tc).^t_v)); % Saturated Vapor Density, Eq. 3.15

% CALCULATE THE PROPERTIES AT THE GIVEN TEMPERATURE AND DENSITY
%==========================================================================
% Account for negative density or temperature

if rho < 0 || T < 0
    X = NaN;
    P = NaN;
    s_v = NaN; u_v = NaN; cp_v = NaN;
    cv_v = NaN; h_v = NaN; c_v = NaN;
    s_l = NaN; u_l = NaN; cp_l = NaN;
    s_l = NaN; u_l = NaN; cp_l = NaN;
end
\[ \begin{align*}
\text{cv}_l &= \text{NaN}; \quad h_l = \text{NaN}; \quad c_l = \text{NaN}; \\
\text{s} &= s_v \times X + s_l \times (1-X); \\
\text{u} &= u_v \times X + u_l \times (1-X); \\
\text{h} &= h_v \times X + h_l \times (1-X); \\
\text{cp} &= cp_v \times X + cp_l \times (1-X); \\
\text{cv} &= cv_v \times X + cv_l \times (1-X); \\
\text{c} &= \text{NaN}; \\
\text{state} &= -1;
\end{align*} \]

\[
\begin{align*}
\% \text{ GAS} \\
\%-----------------------------
\% \text{ EISEIF} \rho < \rho_v \mid \mid \text{imag(rho}_l) \in\in\in\in\in\in\in\in \in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in\in..
else

    \[ X = \rho_v \frac{\rho_l - \rho}{\rho \cdot (\rho_l - \rho_v)}; \]

    \[ [P \ s_l \ u_l \ cp_l \ cv_l \ h_l \ c_l] = \text{Helmholtz}(t, \rho_l / \rho_c); \]

    \[ [P \ s_v \ u_v \ cp_v \ cv_v \ h_v \ c_v] = \text{Helmholtz}(t, \rho_v / \rho_c); \]

    \[ s = s_v X + s_l (1-X); \]

    \[ u = u_v X + u_l (1-X); \]

    \[ h = h_v X + h_l (1-X); \]

    \[ cp = cp_v X + cp_l (1-X); \]

    \[ cv = cv_v X + cv_l (1-X); \]

    \[ c = \text{NaN}; \]

    \[ P = P_{\text{sat}}; \]

    \[ \text{state} = 1; \]

end

%% CREATE THE OUTPUT STRUCTURE

Properties.P = P; % Pressure
Properties.X = X; % Quality
Properties.s = s; Properties.s_l = s_l; Properties.s_v = s_v; % Entropy
Properties.u = u; Properties.u_l = u_l; Properties.u_v = u_v; % Internal Energy
Properties.cv = cv; Properties.cv_l = cv_l; Properties.cv_v = cv_v; % Specific ...
    \text{Heat at Constant Volume}
Properties.cp = cp; Properties.cp_l = cp_l; Properties.cp_v = cp_v; % Specific ...
    \text{Heat at Constant Pressure}
Properties.h = h; Properties.h_l = h_l; Properties.h_v = h_v; % Enthalpy
Properties.c = c; Properties.c_l = c_l; Properties.c_v = c_v; % Speed of Sound
Properties.rho_l = rho_l; Properties.rho_v = rho_v; % Density
Properties.state = state; % State

%Properties.gma_l = cp_l/cv_l;
%Properties.gma_v = cp_v/cv_v;
end

%**************************************************************************

function \[ [P \ s \ u \ cp \ cv \ h \ c] = \text{Helmholtz}(t,d) \]
% Calculates CO2 properties using the residual and ideal gas portions of
% the Helmholtz Energy at a given temperature and density normalized to the
% critical point temperature and density.

% Based upon the Helmholtz Energy based equations of state described by
% Span, R. and Wagner, W. in
% "A New Equation of State for Carbon Dioxide Covering the Fluid Region
% from the Triple-Point Temperature to 1100 K at Pressures up to 800 MPa"
% Journal of Physical and Chemical Reference Data
% Vol 25, No. 6, 1996. Pp 1509–1596

% Input:
% t  -- Inverse Reduced Temperature, tau = T_critical/T, T -- Temperature
% d  -- Reduced Density, Δ = phi/phi_critical, phi -- Mass Density

% Output:
% P  -- Pressure, Pa
% s  -- Entropy, kJ/(kg*K)
% u  -- Internal energy, kJ/kg
% cp -- Isochoric heat capacity, kJ/(kg*K)
% cv -- Isobaric heat capacity, kJ/(kg*K)
% h  -- Entropy, kJ/kg
% c  -- Speed of sound, m/s

% RESIDUAL PART OF HELMHOLTZ

% CONSTANT PARAMETER DEFINITION
% Page 1544, Table 31. Coefficients and exponents of Eq. (6.5)

n7 = [ 0.38856823203161E0
       0.29385475942740E1]
\[52\]

\[202\] \[-0.55867188534934E1\]
\[203\] \[-0.76753199592477E0\]
\[204\] \[0.31729005580416E0\]
\[205\] \[0.54803315897767E0\]
\[206\] \[0.12279411220335E0\];
\[207\] \[d7 = [1 1 1 2 2 3]';\]
\[208\] \[t7 = [0.00 0.75 1.00 2.00 0.75 2.00 0.75]';\]
\[210\] \[n34 = [ 0.21658961543220E1\]
\[211\] \[0.15841735109724E1\]
\[212\] \[0.2313705405503E0\]
\[213\] \[0.58116916831436E-1\]
\[214\] \[0.55369137205382E0\]
\[215\] \[0.48946615909422E0\]
\[216\] \[0.24275739843501E-1\]
\[217\] \[0.62494790501678E-1\]
\[218\] \[0.12175860225246E0\]
\[219\] \[0.37055685270086E0\]
\[220\] \[0.16775879700426E-1\]
\[221\] \[0.11960736637987E0\]
\[222\] \[0.45619362508778E-1\]
\[223\] \[0.35612789270346E-1\]
\[224\] \[0.74427727132052E-2\]
\[225\] \[0.17395704902432E-2\]
\[226\] \[0.21810121289527E-1\]
\[227\] \[0.2432166559236E-1\]
\[228\] \[0.37440133423463E-1\]
\[229\] \[0.14383715756878E0\]
\[230\] \[0.13491969083286E0\]
\[231\] \[0.23151225053480E-1\]
\[232\] \[0.12363125492901E-1\]
\[233\] \[0.21058321972940E-2\]
\[234\] \[0.33958519026368E-3\]
\[235\] \[0.5599365171592E-2\]
\[236\] \[0.30335118055646E-3\];
\[ d34 = [1 \ 2 \ 4 \ 5 \ 5 \ 6 \ 6 \ 6 \ 1 \ 1 \ 4 \ 4 \ 4 \ 7 \ 8 \ 2 \ 3 \ 3 \ 5 \ 5 \ 6 \ 7 \ 8 \ 10 \ 4 \ 8]'; \]
\[ t34 = [1.5000 \ 1.5000 \ 2.5000 \ 0.0000 \ 1.5000 \ 2.0000 \ 0.0000 \ 1.0000 \ ... \\
2.0000 \ 3.0000 \ 6.0000 \ 3.0000 \ 6.0000 \ 8.0000 \ 8.0000 \ 6.0000 \ 0.0000 \ 7.0000 \ 12.0000 \ ... \\
16.0000 \ 22.0000 \ 24.0000 \ 16.0000 \ 24.0000 \ 8.0000 \ 2.0000 \ 28.0000 \ 14.0000]'; \]
\[ c34 = [1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 2 \ 2 \ 2 \ 2 \ 2 \ 2 \ 3 \ 3 \ 3 \ 4 \ 4 \ 4 \ 4 \ 5 \ 6]'; \]
\[
\% \nonumber \\
\]
\[ n39 = \{-0.21365488688320E3 \\
0.26641569149272E5 \\
-0.24027212204557E5 \\
-0.28341603423999E3 \\
0.21247284400179E3\}; \]
\[ d39 = [2 \ 2 \ 2 \ 3 \ 3]'\]
\[ t39 = [1.00 \ 0.00 \ 1.00 \ 3.00 \ 3.00]'\]
\[ a39 = [25 \ 25 \ 25 \ 15 \ 20]'\]
\[ b39 = [325 \ 300 \ 300 \ 275 \ 275]'\]
\[ y39 = [1.16 \ 1.19 \ 1.19 \ 1.25 \ 1.22]'\]
\[ e39 = [1.00 \ 1.00 \ 1.00 \ 1.00 \ 1.00]'\]
\[
\% \nonumber \\
\]
\[ n42 = \{-0.66642276540751E0 \\
0.72608632349897E0 \\
0.55068668612842E-1\}; \]
\[ a42 = [3.500 \ 3.500 \ 3.000]'\]
\[ b42 = [0.875 \ 0.925 \ 0.875]'\]
\[ bt42 = [0.300 \ 0.300 \ 0.300]'\]
\[ A42 = [0.700 \ 0.700 \ 0.700]'\]
\[ B42 = [0.3 \ 0.3 \ 1.0]'\]
\[ C42 = [10.0 \ 10.0 \ 12.5]'\]
\[ D42 = [275 \ 275 \ 275]'\]
\[
R = 0.1889241; \ % \text{Gas Constant, kJ/(kg*K)} \\
Tc = 304.1282; \ % \text{Critical Temperature, K} \\
rhoc = 467.6; \ % \text{Critical Density, kg/m3} \]
\[
\% \nonumber \\
\]
\% NONDIMENSIONALIZED HELMHOLTZ FREE ENERGY
% RESIDUAL PORTION OF THE HELMHOLTZ ENERGY

\[ tta = (1-t) + A42 \cdot ((d-1)^2)^{0.5/bt42}; \]
\[ Del = tta^2 + B42 \cdot ((d-1)^2)^{a42}; \]
\[ Psi = \exp(-C42 \cdot (d-1)^2 - D42 \cdot (t-1)^2); \]

\[ \Psi_{d} = -2 \cdot C42 \cdot (d-1) \cdot \Psi; \]
\[ \Psi_{t} = -2 \cdot D42 \cdot (t-1) \cdot \Psi; \]
\[ Del_{d} = (d-1) \cdot (A42 \cdot tta \cdot 2/bt42 \cdot ... \]
\[ ((d-1)^2)^{0.5/bt42} + ... \]
\[ 2 \cdot B42 \cdot a42 \cdot ((d-1)^2)^{(a42-1)}; \]

% Residual part of the Helmholtz energy, Eq. 6.5

\[ \text{phir} = \sum(n7 \cdot d^d7 \cdot t^t7) ... \]
\[ + \sum(n34 \cdot d^d34 \cdot t^t34 \cdot \exp(-d^c34)) ... \]
\[ + \sum(n39 \cdot d^d39 \cdot t^t39 \cdot ... \]
\[ \exp(-a39 \cdot (d-e39)^2 - bt39 \cdot (t-y39)^2)) ... \]
\[ + \sum(n42 \cdot Del^b42 \cdot d \cdot \Psi); \]

\[ \text{phir}_{d} = \sum(n7 \cdot d^d7 \cdot (d-1)^{d7-1}) \cdot t^t7) ... \]
\[ + \sum(n34 \cdot d^d34 \cdot (d^34-1) \cdot t^t34 \cdot ... \]
\[ \exp(-d^c34) \cdot (d34 - c34 \cdot d^c34)) ... \]
\[ + \sum(n39 \cdot d^d39 \cdot t^t39 \cdot ... \]
\[ \exp(-a39 \cdot (d-e39)^2 - bt39 \cdot (t-y39)^2) \cdot ... \]
\[ (d39./d - 2*a39.*(d-e39)) ... \]
+ \sum(n42.* (\text{Del}.^b42.* (\Psi + d.*\Psi_dd)) + ... \\
  b42.* \text{Del}.^\sim b42.* (\Psi + d.*\Psi_d) + ... \\

\text{phir}_{dd} = \sum(n7.* d7.* (d7 - 1) .* d.^\sim (d7-2) .* t.^\sim t7) ... \\
  + \sum(n34.* \exp(-d.^\sim c34).* d.^\sim (d34-2) .* t.^\sim t34 .* ... \\
  ((d34 - c34.* d.^\sim c34).* (d34 -1-c34.* d.^\sim c34) - c34.^2.* ... \\
  d.^\sim c34)) ... \\
  + \sum(n39.* t.^\sim t39 .* ... \\
  \exp(-a39.*(d - e39).^2 - bt39.*(t - y39).^2) .* ... \\
  (-2.*a39.*d.^\sim d39 + 4.*a39.^2.*d.^\sim d39.*(d-e39).^2 - ... \\
  4.*d39.*a39.*d.^\sim (d39-1).* (d-e39) + d39.* (d39-1).* d.^\sim (d39-2))) ... \\
  + \sum(n42.* (\text{Del}.^\sim b42.* (2.*\Psi_d + d.*\Psi_{dd}) + ... \\
  2.*b42.* \text{Del}.^\sim b42.* (\Psi + d.*\Psi_{dd}) + ... \\
  b42.* d.*\Psi_{dd} + (\text{Del}.^\sim b42-1) .* \text{Del}.^\sim (\text{Del}.^\sim d2)));

\text{phir}_t = \sum(n7.* t7.* d.^\sim d7.* t.^\sim (t7-1)) ... \\
  + \sum(n34.* d.^\sim d34.* t34.* t.^\sim (t34-1) .* \exp(-d.^\sim c34)) ... \\
  + \sum(n39.* d.^\sim d39.* t.^\sim t39 .* ... \\
  \exp(-a39.*(d - e39).^2 - bt39.*(t - y39).^2) .* ... \\
  (t39./t - 2.*bt39.*(t-y39)) ... \\
  + \sum(n42.* d.* (2.*tta.*b42.*\text{Del}.^\sim (b42-1).* \Psi + ... \\
  \text{Del}.^\sim b42.* \Psi_{tt}));

\text{phir}_{tt} = \sum(n7.* t7.* (t7-1) .* d.^\sim d7.* t.^\sim (t7-2)) ... \\
  + \sum(n34.* t34.* (t34-1) .* d.^\sim d34.* t.^\sim (t34-2) .* ... \\
  \exp(-d.^\sim c34)) ... \\
  + \sum(n39.* d.^\sim d39.* t.^\sim t39 .* ... \\
  \exp(-a39.*(d - e39).^2 - bt39.*(t - y39).^2) .* ... \\
  ((t39./t - 2.*bt39.*(t-y39)).^2 - t39./(t.^2-2.*bt39)) ... \\
  + \sum(n42.* d.* (\Psi.*(2.*b42.*\text{Del}.^\sim (b42-1)) + ... \\
  4.*tta.* t2.*b42.* (b42-1).* \text{Del}.^\sim (b42-2)) - ... \\
  4.*tta.* b42.* \text{Del}.^\sim (b42-1).* \Psi_{tt} + \text{Del}.^\sim b42.* \Psi_{tt});

\text{phir}_{dt} = \sum(n7.* t7.* d7.* d.^\sim (d7-1) .* t.^\sim (t7-1)) ...
+ sum(n34 .* exp(-d.\^c34) .* t34 .* d.\(^{(d34-1) .* t.\(^{(t34-1) .* ...}}\) \\
(d34 - c34.* d.\^c34)) ... \\
+ sum(n39 .* d.\^d39 .* t.\^t39 .* ... \\
exp(-a39.*(d - e39).\(^2 - bt39.*(t - y39).\(^2) .* ... \\
(t39./t - 2.*bt39.*(t-y39)).*(d39./d - 2.*a39.*(d-e39))) ... \\
+ sum(n42.* ( Del.\^b42.*(Psi_t + d.*Psi_d) + ... \\
d.*b42.*Del.\(^{(b42-1) .*Del_d.*Psi_t - ... \\
2.*tta.*b42.*Del.\(^{(b42-1) .*Del_d.*Psi_t - ... \\
d.*Psi.*(-A42.*b42.*0.5./bt42.*Del.\(^{(b42-1) .*((d-1).\(^2).\(^{(0.5./bt42-1) ... \)

% % IDEAL GAS HELMHOLTZ 
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONSTANT PARAMETER DEFINITION - 
% Page 1540, Table 27. Coefficients of the correlation equations, Eq. (6.2) 
% and Eq. (6.3) 
%______________________________________________________________

ao3 = [8.37304456 -3.70454304 2.50000000]';
ao8 = [1.99427042 0.62105248 0.41195293 1.04028922 0.08327678];
ttao8 = [3.15163 6.11190 6.77708 11.32384 27.08792];

% Ideal Gas part of the Helmholtz energy, Eq. 6.3 
%______________________________________________________________

phio = log(d) + ao3(1) + ao3(2)*t + ao3(3)*log(t)...
+ sum(ao8.*log( 1-exp(-ttao8.*t) ));

% phio_d = 1/d;
% phio.dd = -1/d^2;
% phio.dt = 0;

phio.t = ao3(2) + ao3(3)/t + sum(ao8.*ttao8.*( (1-exp(-ttao8.*t)).*(-1) ... -1 ));

phio.tt = -ao3(3)/(t^2) - sum(ao8 .* ttao8.^2 .* exp(-ttao8.*t) .* ... 
( 1-exp(-ttao8.*t) ).*(-2) )

%% THERMODYNAMIC PROPERTY CALCULATIONS
%==========================================================================
% Page 1517, Table 3. Relations of thermodynamic properties to the
% dimensionless Helmholtz function phi consisting of phi_ideal_gas and
% phi_residual
%-----------------------------
% Pressure
P = 1000*d*rhoc*R*Tc/t*(1+d*phir.d);

% Entropy
s = R*(t*(phio.t + phir.t) - phio - phir);

% Internal energy
u = R*Tc/t*t*(phio.t+phir.t);

% Isochoric heat capacity
cv = -R*t^2*(phio.tt+phir.tt);

% Isobaric heat capacity
cp = R*(-t^2*(phio_tt+phir_tt) + ... 
(1+d*phir_d-d*t*phir_dt)^2/(1+2*d*phir_d+d^2*phir_dd));

% Enthalpy
h = R*Tc/t*(1+t*(phio_t+phir_t)+d*phir_d);

% Speed of sound
c = sqrt( R*Tc/t*1000*( 1 + 2*d*phir_d + d^2*phir_dd - ... 
( (1 + d*phir_d - d*t*phir_dt)^2/(t^2*(phio_tt+phir_tt))) ));

% Joule-Thompson coefficient
mu = -1/(R*d*rhoc)*(d*phir_d + d^2*phir_dd + d*t*phir_dt) ... 
/ ((1+d*phir_d-d*t*phir_dt)^2 - ... 
(1+2*d*phir_d+d^2*phir_dd));
% Fugacity
407 \psi = \exp(\phi_r + d \phi_{rd} - \log(1 + d \phi_{rd})) ;
408
% 2nd Virial
409 B = \lim(\phi_{rd})/\rho_c as d->0
410
% 3rd Virial
411 C = \lim(\phi_{rd})/\rho_c^2 as d->0
412
% Convert the energy terms from J/gram to J/kg
413 \textbf{========================================================================}
414 u = u*1000; s = s*1000; h = h*1000;
415
% Reference the energy terms to the same reference as the NIST tables
416 \textbf{========================================================================}
417 u_{NIST} = 506.8006990081582;
418 s_{NIST} = 2.739101354918014;
419 h_{NIST} = 506.77604151182834;
420
% Shift the energy terms to match the NIST reference point
421 \textbf{------------------------------------------------------------------------}
422 u = u + u_{NIST};
423 s = s + s_{NIST};
424 h = h + h_{NIST};
425
% end
426

\subsection{CO$_2$ Properties - NIST Webbook Method}

\begin{verbatim}
function [Props] = CO2PropsNIST(T,rho)

% INTRODUCTION
%==========================================================================
% Purpose:
% Returns a structure containing the thermodynamic properties for saturated
\end{verbatim}
% carbon dioxide (CO2) for a given density (kg/m^3) and temperature (K).
% Valid temperature and pressure ranges are:
% 216.59 K ≤ T ≤ 303.69K  0 MPa ≤ P ≤ 800 MPa
% The properties in the structure are:

% P  X  s  u  cv  cp  h  c
% rho_l  s_l  u_l  cv_l  cp_l  h_l  c_l
% rho_v  s_v  u_v  cv_v  cp_v  h_v  c_v

% Inputs:
% T  −  Temperature, K
% rho  −  Density, kg/m3
% Outputs:
% P  −  Pressure, MPa
% X  −  Quality, (Vapor Mass/Total Fluid Mass)
% s  −  Specific Entropy, kJ/(kg*K)
% u  −  Specific Internal Energy, kJ/kg
% cv  −  Specific Heat at Constant Volume, kJ/(kg*K)
% cp  −  Specific Heat at Constant Pressure, kJ/(kg*K)
% h  −  Specific Enthalpy, kJ/kg
% c  −  Speed of Sound, m/s
% rho  −  Density, kg/m3
% state  −  _l = Negative Input, 0 = Liquid, 1 = Saturated, 2 = Gas
% _l  −  Liquid designator
% _v  −  Vapor designator

% Revision History:
% Written for a CO2 Blowdown model developed at Utah State University by
% Brian Solomon
% Based upon the Helmholtz Energy based equations of state described by
% Span, R. and Wagner, W. in
% "A New Equation of State for Carbon Dioxide Covering the Fluid Region
% from the Triple-Point Temperature to 1100 K at Pressures up to 800 MPa"
%% CALCULATE THE SATURATION PROPERTIES

%==========================================================================

% CO2 Constants

R = 0.1889241; % Gas constant, kJ/(kg*K)
Tt = 216.592; % Triple point temperature, K (Eq. 3.1)
Pt = 0.51795; % Triple point pressure, MPa (Eq. 3.2)
Tc = 304.1282; % Critical temperature, K (Eq. 3.3)
Pc = 7.3773; % Critical pressure, MPa (Eq. 3.4)
rhoc = 467.6; % Critical density, kg/m3 (Eq. 3.5)

% Load the CO2 properties created by the NIST Webbook

[labels,TNIST,y] = readColData('SatCO2Props.txt',25);
P_NIST = y(:,1); % Pressure (MPa)
rho_l_NIST = y(:,2); % Density (l, kg/m3)
rho_v_NIST = y(:,14); % Density (v, kg/m3)
u_l_NIST = y(:,4); % Internal Energy (l, kJ/kg)
u_v_NIST = y(:,16); % Internal Energy (v, kJ/kg)
h_l_NIST = y(:,5); % Enthalpy (l, kJ/kg)
h_v_NIST = y(:,17); % Enthalpy (v, kJ/kg)
s_l_NIST = y(:,6); % Entropy (l, J/g*K)
s_v_NIST = y(:,18); % Entropy (v, J/g*K)
cv_l_NIST = y(:,7); % Cv (l, J/g*K)
cv_v_NIST = y(:,19); % Cv (v, J/g*K)

SatCO2Props = open('SatCO2Props.mat');
% Temperature (K)
P_NIST = SatCO2Props.P_NIST; % Pressure (MPa)
rho_l_NIST = SatCO2Props.rho_l_NIST; % Density (l, kg/m3)
rho_v_NIST = SatCO2Props.rho_v_NIST; % Density (v, kg/m3)
u_l_NIST = SatCO2Props.u_l_NIST; % Internal Energy (l, kJ/kg)
u_v_NIST = SatCO2Props.u_v_NIST; % Internal Energy (v, kJ/kg)
h_l_NIST = SatCO2Props.h_l_NIST; % Enthalpy (l, kJ/kg)
h_v_NIST = SatCO2Props.h_v_NIST; % Enthalpy (v, kJ/kg)
s_l_NIST = SatCO2Props.s_l_NIST; % Entropy (l, J/g*K)
s_v_NIST = SatCO2Props.s_v_NIST; % Entropy (v, J/g*K)
cvl_NIST = SatCO2Props.cv_l_NIST; % Cv (l, J/g*K)
cvv_NIST = SatCO2Props.cv_v_NIST; % Cv (v, J/g*K)
cpl_NIST = SatCO2Props.cp_l_NIST; % Cp (l, J/g*K)
 cpp_NIST = SatCO2Props.cp_v_NIST; % Cp (v, J/g*K)
c_l_NIST = SatCO2Props.c_l_NIST; % Sound Speed (l, m/s)
c_v_NIST = SatCO2Props.c_v_NIST; % Sound Speed (v, m/s)

% Interpolate to find the liquid and vapor properties

%==========================================================================
rho_l = interp1(T_NIST,rho_l_NIST,T,'spline');
rho_v = interp1(T_NIST,rho_v_NIST,T,'spline');
P = interp1(T_NIST,P_NIST,T,'spline');
s_l = interp1(T_NIST,s_l_NIST,T,'spline');
s_v = interp1(T_NIST,s_v_NIST,T,'spline');
u_l = interp1(T_NIST,u_l_NIST,T,'spline');
u_v = interp1(T_NIST,u_v_NIST,T,'spline');
cp_l = interp1(T_NIST,cp_l_NIST,T,'spline');
cp_v = interp1(T_NIST,cp_v_NIST,T,'spline');
cvl = interp1(T_NIST,cv_l_NIST,T,'spline');
cvv = interp1(T_NIST,cv_v_NIST,T,'spline');
h_l = interp1(T_NIST,h_l_NIST,T,'spline');
h_v = interp1(T_NIST,h_v_NIST,T,'spline');
c_l = interp1(T_NIST,c_l_NIST,T,'spline');
c_v = interp1(T_NIST,c_v_NIST,T,'spline');

% CALCULATE THE PROPERTIES AT THE GIVEN TEMPERATURE AND DENSITY
% Account for negative density or temperature

if rho < 0 || T < 0
    X = NaN;
    P = NaN;
    s_v = NaN; u_v = NaN; cp_v = NaN;
    cv_v = NaN; h_v = NaN; c_v = NaN;
    s_l = NaN; u_l = NaN; cp_l = NaN;
    cv_l = NaN; h_l = NaN; c_l = NaN;
    s = s_v*X + s_l*(1-X);
    u = u_v*X + u_l*(1-X);
    h = h_v*X + h_l*(1-X);
    cp = cp_v*X + cp_l*(1-X);
    cv = cv_v*X + cv_l*(1-X);
    c = NaN;
    state = -1;
else
    X = (rho_v/rho)*((rho_l-rho)/(rho_l-rho_v));
    s = s_v*X + s_l*(1-X);
    u = u_v*X + u_l*(1-X);
    h = h_v*X + h_l*(1-X);
    cp = cp_v*X + cp_l*(1-X);
    cv = cv_v*X + cv_l*(1-X);
    c = c_v*X + c_l*(1-X);
    state = 1;
end

% CREATE THE OUTPUT STRUCTURE

Props.P = P; % Pressure
Props.X = X; % Quality
Props.s = s; Props.s_l = s_l; Props.s_v = s_v; % Entropy
B.4 CO₂ Properties - Helmholtz vs. NIST Plotter

```matlab
clc; clear all;

% Create a temperature and density vector to calculate CO₂ properties at
%---------------------------------------------
T = linspace(216.59,300,50);  % Temperature vector, K (-27.67 to 116.33F)
%T = [T linspace(300,304.1,15)];
T = [T linspace(300,307,50)];
%rho = linspace(200,1100);  % Density vector, kg/m³
rho = 500;

% Load the CO₂ properties created by the NIST Webbook
%---------------------------------------------
[labels,x,y] = readColData('SatCO2Props.txt',25);
n = 3;
T_NIST = downsample(x,n);  
P_NIST = downsample(y(:,1),n);
rho_l_NIST = downsample(y(:,2),n);
rho_v_NIST = downsample(y(:,14),n);
```
% Loop through temperature and density vectors to populate the CO2 properties

for i = 1:length(T)
    for j = 1:length(rho)
        Props = CO2Props(T(i),rho(j));
        P(i,j) = Props.P; % Pressure
        X(i,j) = Props.X; % Quality
        s(i,j) = Props.s; % Entropy
        s_l(i,j) = Props.s_l;
        s_v(i,j) = Props.s_v;
        u(i,j) = Props.u; % Internal Energy
        u_l(i,j) = Props.u_l;
        u_v(i,j) = Props.u_v;
        cv(i,j) = Props.cv; % Specific Heat at Constant Volume
        cv_l(i,j) = Props.cv_l;
        cv_v(i,j) = Props.cv_v;
        cp(i,j) = Props.cp; % Specific Heat at Constant Pressure
        cp_l(i,j) = Props.cp_l;
        cp_v(i,j) = Props.cp_v;
        h(i,j) = Props.h; % Enthalpy
        h_l(i,j) = Props.h_l;
        h_v(i,j) = Props.h_v;
\begin{verbatim}
c(i,j) = Props.c;  % Speed of Sound
c_l(i,j) = Props.c_l;
c_v(i,j) = Props.c_v;
rho_l(i,j) = Props.rho_l;  % Density
rho_v(i,j) = Props.rho_v;
state(i,j) = Props.state;  % State
end
e
end

% Shift energy terms so that the reference matches

%-----------------------------------------------------------
udiff = u_l.NIST(1) - u.l(1)
s_diff = s_l.NIST(1) - s_l(1)
h_diff = h_l.NIST(1) - h_l(1)

% Change the independent variables to temperature and pressure.
%-----------------------------------------------------------
% [rhoArr gmaArr] = CO2tp(T_NIST, P_NIST);
%
% Parse mosster data CO2 density
%-----------------------------------------------------------
rho_parse=-0.136928567045648.*T.^2+68.960274348176327.*T-7677.810285569415;
%
% Plot the pressure vs temperature
%-----------------------------------------------------------
figure(1);
plot(T,P(:,1),'b'); hold on; grid on;
plot(T_NIST,P_NIST,'ob');
%title('CO2 Pressure');
legend('Helmholtz','NIST','Location','SE');
xlabel('Temperature, K'); ylabel('Pressure, MPa');
saveas(1,'CO2_Pressure.png')
%saveas(1,'LaTeX_Plots/CO2_Pressure.eps','epsc')

% Plot the density vs temperature
\end{verbatim}
% Plot the internal energy vs temperature

figure(3);
plot(T,u_l(:,1),'b'); hold on; grid on;
plot(T,NIST,u_l_NIST,'ob');
plot(T,u_v(:,1),'-r')
plot(T,NIST,u_v_NIST,'sr')
title('CO2 Internal Energy');
legend(['Liquid-Helmholtz','Liquid-NIST','Vapor-Helmholtz','Vapor-NIST']);
xlabel('Temperature, K'); ylabel('Internal Energy, kJ/kg');
saveas(3,'CO2_Internal_Energy.png')
%saveas(3,'LaTeX_Plots/CO2_Internal_Energy.eps','epsc')

% Plot the enthalpy vs temperature

figure(4);
plot(T,h_l(:,1),'b'); hold on; grid on;
plot(T,NIST,h_l_NIST,'ob');
plot(T,h_v(:,1),'-r')
plot(T,NIST,h_v_NIST,'sr')
%title('CO2 Enthalpy');
legend('Liquid−Helmholtz','Liquid−NIST','Vapor−Helmholtz','Vapor−NIST','Location','SE');
%legend('Liquid−NIST','Vapor−NIST');
xlabel('Temperature, K'); ylabel('Enthalpy, kJ/kg');
saveas(4,'CO2_Enthalpy.png')
%saveas(4,'LaTeX_Plots/CO2_Enthalpy.eps','epsc')

% Plot the entropy vs temperature
%------------------------------------------------------------------------
figure(5);
plot(T,s_l(:,1),'b'); hold on; grid on;
plot(T,NIST,s_l_NIST,'ob');
plot(T,v(:,1),'-r')
plot(T,NIST,s_v_NIST,'sr')
%title('CO2 Entropy');
legend('Liquid−Helmholtz','Liquid−NIST','Vapor−Helmholtz','Vapor−NIST');
xlabel('Temperature, K'); ylabel('Entropy, J/g*K');
saveas(5,'CO2_Entropy.png')
%saveas(5,'LaTeX_Plots/CO2_Entropy.eps','epsc')

% Plot the specific heat at constant volume vs temperature
%------------------------------------------------------------------------
figure(6);
plot(T,cv_l(:,1),'b'); hold on; grid on;
plot(T,NIST,cv_l_NIST,'ob');
plot(T,v(:,1),'-r')
plot(T,NIST,cv_v_NIST,'sr')
title('CO2 Specific Heat at Constant Volume');
legend('Liquid−Helmholtz','Liquid−NIST','Vapor−Helmholtz','Vapor−NIST');
xlabel('Temperature, K'); ylabel('Specific Heat at Constant Volume, J/g*K');
saveas(6,'CO2_Specific_Heat_at_Constant_Volume.png')
%saveas(6,'LaTeX_Plots/CO2_Specific_Heat_at_Constant_Volume.eps','epsc')

% Plot the specific heat at constant pressure vs temperature
%------------------------------------------------------------------------
figure(7);
plot(T.cp(:,1),'b'); hold on; grid on;
plot(T_NIST,cp_NIST,'ob');
plot(T,cp_v(:,1),'--r')
plot(T_NIST,cp_v_NIST,'sr')
title('CO2 Specific Heat at Constant Pressure');
legend('Liquid-Helmholtz','Liquid-NIST','Vapor-Helmholtz','Vapor-NIST');
xlabel('Temperature, K'); ylabel('Specific Heat at Constant Pressure, J/g*K');
saveas(7,'CO2_Specific Heat at Constant Pressure.png')
%saveas(7,'LaTeX_Plots/CO2_Specific Heat at Constant Pressure.eps','epsc')

% Plot the speed of sound vs temperature
%-----------------------------------------------
figure(8);
plot(T,c_l(:,1),'b'); hold on; grid on;
plot(T_NIST,c_l_NIST,'ob');
plot(T,c_v(:,1),'--r')
plot(T_NIST,c_v_NIST,'sr')
title('CO2 Speed of Sound');
legend('Liquid-Helmholtz','Liquid-NIST','Vapor-Helmholtz','Vapor-NIST');
xlabel('Temperature, K'); ylabel('Speed of Sound, m/s');
saveas(8,'CO2_Speed_of_Sound.png')
%saveas(8,'LaTeX_Plots/CO2_Speed_of_Sound.eps','epsc')

% Plot the enthalpy vs density
%-----------------------------------------------
figure(9);
%plot(T,h_l(:,1),'b');
plot(rho_l,h_l,'--b');hold on; grid on;
%plot(T,h_v(:,1),'r')
plot(rho_v,h_v,'r')
%title('CO2 Enthalpy');
%legend('Liquid-Helmholtz','Liquid-NIST','Vapor-Helmholtz','Vapor-NIST');
legend('Liquid-Helmholtz','Vapor-Helmholtz');
B.5 Density and Enthalpy to Temperature

```matlab
function [T] = CO2_rhoh2T(rho, h, T_guess)

% Purpose:
% This function places a nonlinear solver around the CO2Props function in order to change the independent variable from T & rho to rho & h.

% Inputs:
% rho — Density, kg/m3
% h — Enthalpy, kJ/kg

% Outputs:
% T — Temperature, K

rho_Known = rho; % Save rho to a more intuitive name
h_Known = h; % Save h to a more intuitive name
T_guess = 300; % Guess value for T for lsqnonlin to use

% Create a function for lsqnonlin to solve T(rho,h)
pFunc = @(T_Unknown) getfield(CO2Props(T_Unknown,rho_Known),'h')-h_Known;
[pFunc = @(T_Unknown) ...
    getfield(CO2PropsNIST(T_Unknown,rho_Known),'h')-h_Known;
% Since T_Unknown is not pre defined in pFunc, lsqnonlin will find a T for
% rho_Known and h_Known
```
B.6 Enthalpy Plotter

clc; clear all;

% Creates a plot of CO2 temperature vs density for different enthalpies
% Create a temperature and density vector to calculate CO2 properties at
% Temperature vector, K (−27.67 to 116.33F)
Tarr = linspace(240,320); % Temperature vector, K (-27.67 to 116.33F)
T = 250;
rho = linspace(100,1000); % Density vector, kg/m3
%rho = 1000;

% Loop through temperature and density vectors to populate the CO2 properties
for i = 1:length(T)
    for j = 1:length(rho)
        Props = CO2Props(T(i),rho(j));
        h(i,j) = Props.h;  % Save Enthalpy
    end
end

% Create a vector of enthalpies to plot vs density and temperature
h_Known = h(1:15:length(h));
% Preallocate an array to save the calculated temperature to
Tsave = zeros(length(h_Known),length(rho));

% Calculate the temperature at the previously defined enthalpies and their
% corresponding densities by looping through them and saving temperature
for iter = 1:length(h_Known)
    for k = 1:length(rho)
        Tsave(iter,k) = CO2_rho_h_2T(rho(k), h(iter), 300);
    end
end

% Plot temperature vs density at each of the enthalpies
figure(1);
plot(rho,Tsave(1,:),':b'); hold on; grid on;
plot(rho,Tsave(2,:),':r');
plot(rho,Tsave(3,:),':b');
plot(rho,Tsave(4,:),':r');
plot(rho,Tsave(5,:),':b');
plot(rho,Tsave(6,:),':r');
plot(rho,Tsave(7,:),':b');

%title('CO2 Temperature vs Density at Specific Enthalpies');
legend(['h = ' num2str(h_Known(1),3) ' kJ/kg', ... 
    ['h = ' num2str(h_Known(2),3) ' kJ/kg', ... 
    ['h = ' num2str(h_Known(3),3) ' kJ/kg', ... 
    ['h = ' num2str(h_Known(4),3) ' kJ/kg', ... 
    ['h = ' num2str(h_Known(5),3) ' kJ/kg', ... 
    ['h = ' num2str(h_Known(6),3) ' kJ/kg', ... 
    ['h = ' num2str(h_Known(7),3) ' kJ/kg', ...
    'Location','SE');
xlabel('Density, kg/m3'); ylabel('Temperature, K');
saveas(1,'CO2 Temp vs Density for Diff Enthalpies.png')

B.7 Discrete Wavelet Transform Derivative

function dudx=derivative_dwt(u,wt_name,wt_level,dx,trt_flag)

% Differentiation (Derivative) of Sampled Data Based on Discrete Wavelet Transform...
% dudx=derivative_dwt(u,wt_name,wt_level,dx)
% 
% u: uniformly-sampled data
% wt_name: name of the wavelet function (haar or spl)
% wt_level: level of wavelet decomposition
% dx: sampling interval (default=1)
% trt_flag: flag of translation-rotation transformation for boundary ...
% effect (default=1)
% dudx: differentiations of data (u)
% 
% See also
% derivative_cwt
%
% Reference:
% J. W. Luo, J. Bai, and J. H. Shao,
% "Application of the wavelet transforms on axial strain calculation in ...
% ultrasound elastography,"

if nargin<5
    trt_flag=1;
end
if nargin<4
dx=1;
end

if trt_flag
    x=(1:length(u))*dx;
    a=(u(end)-u(1))/(x(end)-x(1));
    b=u(1)-a*x(1);
    u=u-a*x-b;
else
    a=0;
end
wt_name=lower(wt_name);

if strcmp(wt_name,'haar')
    h0=[sqrt(2)/2 sqrt(2)/2];  %the decomposition low-pass filter
    h1=[-sqrt(2)/2 sqrt(2)/2];  %the decomposition high-pass filter
elseif strcmp(wt_name,'spl')
    h0=[0.125 0.375 0.375 0.125]*sqrt(2);
    h1=[-2 2]*sqrt(2);
else
    error('wavelet name error');
end

y0=u;

% Algorithme a Trous
for n=1:wt_level
    h0_atrous=[h0' zeros(length(h0),2^(n-1)-1)]';
    h0_atrous=h0_atrous(1:(length(h0)-1)*(2^(n-1)-1)+length(h0));
    h1_atrous=[h1' zeros(length(h1),2^(n-1)-1)]';
    h1_atrous=h1_atrous(1:(length(h1)-1)*(2^(n-1)-1)+length(h1));
    y1=conv(y0,h1_atrous);
    y0=conv(y0,h0_atrous);
end

index=round(length(y1)/2-length(u)/2+[1:length(u)]);
dudx=y1(index);

wt_scale=2^wt_level;

if strcmp(wt_name,'haar')
    dudx=dudx/wt_scale^(3/2)*4;
elseif strcmp(wt_name,'spl')
dudx = dudx/wt_scale^(3/2);
else
    error('wavelet name error');
end
dudx = dudx/dx+a;