12-2011

Laminar Flame Speed Estimation from Experimental Data Using a Quasi-Dimensional Turbulent Flame Entrainment Combustion Simulation For Spark Ignition Engines

Akash Desai
Clemson University, akashd@clemson.edu

Follow this and additional works at: https://tigerprints.clemson.edu/all_theses
Part of the Mechanical Engineering Commons

Recommended Citation
https://tigerprints.clemson.edu/all_theses/1281

This Thesis is brought to you for free and open access by the Theses at TigerPrints. It has been accepted for inclusion in All Theses by an authorized administrator of TigerPrints. For more information, please contact kokeefe@clemson.edu.
Laminar Flame Speed Estimation from Experimental Data Using a Quasi-Dimensional Turbulent Flame Entrainment Combustion Simulation For Spark Ignition Engines

A Thesis
Presented to
The Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Mechanical Engineering

by
Akash Sanjay Desai
December 2011

Accepted by:
Dr. Robert Prucka, Committee Chair
Dr. John Wagner
Dr. Ardalan Vahidi
ABSTRACT

The goal of this research is to develop a thermodynamic simulation of spark-ignition engine combustion that uses a predictive burn-rate model. Previously done thermodynamic engine simulations in MATLAB [1] [2] are based on a specified burn rate model. Also, the effect of turbulence parameters on the rate of mass burn up is not considered. A predictive burn rate model is necessary to study the effect of different fuels on spark ignition engine combustion. The effect of laminar flame speed and turbulent intensity on combustion is difficult to assess experimental due to the difficulty in the measurement of these two variables. Thus, the aim of this simulation is to thermodynamically model the spark ignition engine cycle taking into account the turbulence parameters and their effect on the combustion process. The simulation can be used for predictive studies of combustion of different varieties of fuels by calibration of laminar flame speed, assuming the turbulent intensity to be a fuel-independent parameter.

The key parameters of interest are the turbulence intensity and laminar flame speed. Both these parameters play a key role in controlling mass burn-up rate. The simulation intends to calculate mass fraction burned profile and turbulence intensity from a predictive combustion model based on the concept of turbulent flame entrainment and diffusive burn-up [3] [4]. An empirical relation for laminar flame speed is used for known fuel (gasoline) which can be later calibrated for variety of fuels. The simulation calculates the mass fraction burned profile based on calculated cylinder pressure, temperature and species concentration.
The cylinder pressure and temperature predicted by the simulation match well with the experimental data for the engine used for calibration. The MFB10, MFB50 and MFB90 crank angle positions simulated by the predictive combustion model were compared with those calculated from a commercially available engine thermodynamics program (AVL Concerto) which is based on experimental data and were found within 5-10% of each other. The turbulence parameters, laminar flame speed and the length scales during combustion also showed results in accordance with those shown by previous experiments / simulation [5].

The quasi-dimensional simulation is done in MATLAB for computational speed and acceptable accuracy. Using MATLAB also makes inserting additional sub-models easier to make the model more detailed and accurate. This simulation will serve as a student guide to engine modeling and also can be used as a base for more advanced simulations.
ACKNOWLEDGMENTS

I would primarily like to acknowledge Dr. Robert Prucka of CU-ICAR. Without his guidance and support, this research would not have been possible. His help during this research and related courses has greatly improved my knowledge of the subject and I am very thankful for that. I would also like to thank Dr. Wagner and Dr. Vahidi for their cooperation during the final stages of the thesis.

My friends at Clemson University have given me a great deal of motivation and support during my research work and I greatly appreciate their concern and kindness. I would like to dedicate this thesis to my family. Without their encouragement, this thesis would not have been possible. Thank you. Go Tigers!!
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE PAGE</td>
<td>i</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>ii</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>LIST OF SYMBOLS, SUBSCRIPTS AND ABBREVIATIONS</td>
<td>x</td>
</tr>
<tr>
<td><strong>CHAPTERS</strong></td>
<td></td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1. Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2. Thesis Outline</td>
<td>3</td>
</tr>
<tr>
<td>2. THERMODYNAMIC MODEL DESCRIPTION</td>
<td>4</td>
</tr>
<tr>
<td>2.1. Model Description and Assumptions</td>
<td>4</td>
</tr>
<tr>
<td>2.2. Thermodynamic Analysis</td>
<td>5</td>
</tr>
<tr>
<td>2.3. Gas Exchange Model</td>
<td>9</td>
</tr>
<tr>
<td>2.4. Turbulent Flow Model</td>
<td>10</td>
</tr>
<tr>
<td>2.5. Heat Transfer Model</td>
<td>14</td>
</tr>
<tr>
<td>2.6. Combustion Model</td>
<td>16</td>
</tr>
<tr>
<td>2.7. Method of Solution</td>
<td>21</td>
</tr>
<tr>
<td>3. MATLAB SIMULATION OUTLINE</td>
<td>22</td>
</tr>
<tr>
<td>3.1. Simulation Framework</td>
<td>23</td>
</tr>
<tr>
<td>3.2. Simulation Inputs</td>
<td>23</td>
</tr>
<tr>
<td>3.3. Simulation Output</td>
<td>28</td>
</tr>
<tr>
<td>4. CALIBRATION OF MATLAB SIMULATION</td>
<td>30</td>
</tr>
<tr>
<td>4.1. Calibration Background</td>
<td>30</td>
</tr>
<tr>
<td>4.2. Description of Experimental Facility</td>
<td>30</td>
</tr>
</tbody>
</table>
# Table of Contents (Continued)

4.3. Calibration of the Model .......................................................... 32  
4.4. Validation of Calibration .......................................................... 34  

5. VALIDATION OF SUB-MODELS .................................................. 37  
   5.1. Introduction to Validation of Sub-Model ................................. 37  
   5.2. Main Variables .................................................................... 38  
   5.3. Gas Exchange Sub-Models ..................................................... 38  
   5.4. Turbulence Parameters ......................................................... 40  

6. CONCLUSION .................................................................................. 44  
   6.1. Conclusion ............................................................................. 44  
   6.2. Future Work ........................................................................... 45  

APPENDICES ......................................................................................... 47  
   A: List of Programs ........................................................................ 48  
   B: Sub-Models Used by Main Functions ........................................ 49  
   C: Matlab Programs ..................................................................... 50  

REFERENCES ..................................................................................... 110
<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Sample simulation inputs</td>
</tr>
<tr>
<td>4.1</td>
<td>Engine geometry specifications</td>
</tr>
<tr>
<td>4.2</td>
<td>Valve timings (referenced to TDC of compression as 0°)</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>P-V diagram of SI engine showing phases of engine cycle</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>Zero-dimensional turbulent energy cascade model</td>
<td>11</td>
</tr>
<tr>
<td>2.3</td>
<td>Schematic of turbulent flame propagation process</td>
<td>17</td>
</tr>
<tr>
<td>3.1</td>
<td>Flowchart of the cycle simulation</td>
<td>23</td>
</tr>
<tr>
<td>3.2</td>
<td>Valve lift and discharge coefficient data</td>
<td>27</td>
</tr>
<tr>
<td>3.3</td>
<td>Flame geometry data</td>
<td>28</td>
</tr>
<tr>
<td>3.4</td>
<td>Normalized flame area versus flame radius for pent roof combustion chamber at different crank positions</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>Engine test cell at CU-ICAR</td>
<td>31</td>
</tr>
<tr>
<td>4.2</td>
<td>Measured and predicted pressure traces for GM 5.3 L V-8 engine at 3000 RPM and spark at 22° BTDC</td>
<td>33</td>
</tr>
<tr>
<td>4.3</td>
<td>Predicted mass fraction entrained and burned profile</td>
<td>34</td>
</tr>
<tr>
<td>4.4</td>
<td>Predicted mass fraction entrained and burned profile</td>
<td>35</td>
</tr>
<tr>
<td>4.5</td>
<td>Predicted pressure trace for 5000 RPM (WOT)</td>
<td>37</td>
</tr>
<tr>
<td>5.1</td>
<td>Predicted temperature trace for 5000 RPM (WOT)</td>
<td>38</td>
</tr>
<tr>
<td>5.2</td>
<td>Mass flow rate through intake valve for 5000 RPM (WOT)</td>
<td>39</td>
</tr>
</tbody>
</table>
List of Figures (Continued)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4</td>
<td>Mass flow rate through exhaust valve for 5000 RPM (WOT)</td>
<td>40</td>
</tr>
<tr>
<td>5.6</td>
<td>Turbulence intensity ($u'$) at 5000 RPM (WOT)</td>
<td>41</td>
</tr>
<tr>
<td>5.7</td>
<td>Evaluation of turbulence length scales $\lambda$ &amp; $L$ during combustion</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>stroke at 5000 RPM (WOT)</td>
<td></td>
</tr>
<tr>
<td>5.8</td>
<td>Laminar flame speed trace during combustion stroke</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>at 5000 RPM (WOT)</td>
<td></td>
</tr>
</tbody>
</table>
LIST OF SYMBOLS, SUBSCRIPTS AND ABBREVIATIONS

SYMBOLS:

- $A_f$ Flame front area (m$^2$)
- $A_r$ Reference flow area (m$^2$)
- $B$ Engine bore (m)
- $a$ Heat transfer relation constant
- $C_p$ Specific heat at constant pressure (J/kg-K)
- $C_t$ Specific heat at constant temperature (J/kg-Pa)
- $C_D$ Discharge coefficient
- $c_\beta$ Turbulent dissipation constant
- $D_h$ Valve head diameter (m)
- $D_s$ Valve stem diameter (m)
- $d$ Heat transfer relation exponent
- $E$ Energy of the system (J)
- $f$ Residual gas fraction
- $h$ Enthalpy of gas (J/kg), Convective heat transfer coefficient (W/m$^2$K)
- $K$ Mean flow kinetic energy (J)
- $k$ Turbulent kinetic energy (J), Thermal conductivity (W/m-K)
- $l$ Characteristic size of large scale eddies (m)
- $L$ Macroscale of turbulence (m)
\( m_e \) Mass entrained (kg)
\( m_b \) Mass burned (kg)
\( Nu \) Nusselt number
\( Pr \) Prandtl number
\( P \) Rate of turbulent kinetic energy production (J/s)
\( \dot{Q}_w \) Rate of heat transfer to the walls (J/s)
\( R \) Gas constant (J/kg-K)
\( Re \) Reynolds number
\( S_L \) Laminar flame speed (m/s)
\( T_g \) Gas temperature (K)
\( T_w \) Wall temperature (K)
\( U \) Mean flow velocity (m/s)
\( u' \) Turbulent intensity (m/s)
\( V_j \) Jet velocity (m/s)
\( V_{ch} \) Characteristic velocity for heat transfer calculation (m/s)
\( W \) Work done (W)
\( \phi \) Equivalence ratio
\( \rho \) Density of gas (kg/m\(^3\))
\( \gamma \) Ratio of specific heats
\( \epsilon \) Rate of turbulent kinetic energy dissipation per unit mass (J/kg-s)
\( \mu_t \) Turbulent viscosity
\( \mu \) Dynamic viscosity (Pa-s)
\( \nu \) Kinematic viscosity (m\(^2\)/s)
\( \lambda \) Taylor microscale (m)
\( \eta \) Kolgomorov scale (m)

**SUBSCRIPTS:**

- \( u \) Unburned gas properties
- \( b \) Burned gas properties
- \( 1 \) Fresh charge properties
- \( 2 \) Residual gas properties

**ABBREVIATIONS:**

- IVO Intake valve open
- IVC Intake valve close
- EVO Exhaust valve open
- EVC Exhaust valve close
- MFB Mass fraction burned
- MFB10 10% mass fraction burned
- MFB50 50% mass fraction burned
- MFB90 90% mass fraction burned
- WOT Wide open throttle
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDC</td>
<td>Top dead center</td>
</tr>
<tr>
<td>BDC</td>
<td>Bottom dead center</td>
</tr>
<tr>
<td>ATDC</td>
<td>After top dead center</td>
</tr>
<tr>
<td>BTDC</td>
<td>Before top dead center</td>
</tr>
</tbody>
</table>
CHAPTER ONE
INTRODUCTION

1.1. Background

A cycle simulation of an engine is the mathematical model of the processes occurring in an engine over the complete operating cycle. Modeling studies are helpful to the research and development of combustion engines as these studies:

1. Create a need to get in-depth understanding of the process under study;
2. Identify the controlling variables during calibration of the model;
3. Reduce the cost of experimental development as a wide range of engine designs can be studied depending on the accuracy of the model;

Two types of models have been developed for engine governing processes. They can be characterized as zero-dimensional or multi-dimensional. Alternatively, they can also be classified as thermodynamic or fluid dynamic models depending upon whether the focus is on the energy conservation of the charge or the full analysis of fluid flow motion [5]. Zero-dimensional models require the rate of combustion as an input since they follow changes only in the bulk properties of the engine charge. Multi-dimensional models give parameters which are spatially resolved but need high computation time.

A quasi-dimensional thermodynamic model incorporates some geometric features to the zero-dimensional thermodynamic model along with a phenomenological model for the spark ignition combustion process. This simulation is based on such principle which
allows checking the effect of specific parameters of the spark ignition engine combustion with moderate computational requirements. The spark ignition combustion simulation developed in this work is based on a spark ignition simulation done by S.G. Poulos [6]. The framework for the simulation is adopted from a specified burn rate model originally developed by Ferguson [7] and incorporated in MATLAB by Buttsworth [1]. Additions to the framework were made first, to incorporate the intake and exhaust stroke in the model and also, to implement a predictive burn rate model to see the effect of turbulence parameters on the combustion performance. The predictive combustion model is based on two-step entrainment and burn-up theory developed by Blizard and Keck [3] and later modified by Tabaczynski et al. [4]. Evaluation of thermodynamic properties required for governing equations is done by polynomial fit to thermodynamic data for each species assuming frozen composition of unburned mixture and equilibrium combustion products. Coefficients for the curve fit to JANAF tables used in the NASA equilibrium program [8] are used given in Heywood [5] and Ferguson [7].

The simulation uses experimental measurements for valve data. Results from a similar simulation [9] are used for flame geometry values. The model is calibrated using measurement data from a GM 5.3. Liter V-8 engine running in a FEV 500 hp engine dynamometer test cell at CU-ICAR.

The simulation is capable of predicting the engine performance parameters with slight deviations. These deviations are caused largely due to the crevice volume considerations, constant pressure manifold assumption and approximate flame geometry data.
The turbulence parameters and the laminar flame speed have a significant effect on the rate of mass burn-up. Because of the inherent difficulty in measurement of these parameters, it is difficult experimentally quantify the dependence of mass burn rate on these parameters. Since this simulation is based on thermodynamic principles driving the combustion phenomenon, the relation between various engine parameters on the combustion process can be quantified.

The direct dependence of laminar flame speed on mass fraction burn rate allows manipulating the burn-rate by changing laminar flame speed. Thus, laminar flame speed data for different fuels can be extracted from the simulation by calibration of mass burn-up rate to experimentally available data.

1.2. Thesis Outline

Chapter 2 contains the assumptions and the mathematical description of the thermodynamic model used. The outline of the MATLAB code and the various inputs and outputs of the model are explained in chapter 3. Chapter 4 describes the calibration procedure used for the model. In chapter 5, results from the sub-models used in the simulation are explained. Chapter 6 gives a brief overview of the results and ideas for future work on the same lines. A list and brief description of MATLAB function and sub-function files required for each main function is given in Appendix A & B respectively. Appendix C contains all of the MATLAB programs.
CHAPTER TWO

THERMODYNAMIC MODEL DESCRIPTION

2.1 Model Description and Assumptions

The cycle simulation is modeled as a sequence of four continuous processes. These processes are intake, compression, combustion (includes expansion) and exhaust. The duration of these processes is specified by the input valve and spark timings.

- Intake: IVO - IVC
- Compression: IVC - Spark time
- Combustion: Spark time - EVO
- Exhaust: EVO – IVO

Figure (2.1): P-V diagram of SI engine showing phases of engine cycle
The instantaneous content of the cylinder charge is the system of interest. This system is open to mass (intake & exhaust) and energy transfer in the form of work and heat. The cylinder is treated as a variable volume plenum, spatially uniform in pressure.

During intake, compression and exhaust, the cylinder charge is assumed to be a homogeneous mixture of non-reacting gases. The cylinder contents are air-fuel and residual gas during intake and compression and products of complete combustion during exhaust. The composition of the cylinder charge is assumed to be frozen thus; a single mean temperature is used to define the state of the cylinder charge during these phases of the simulation.

For combustion phase, the cylinder charge is divided into two zones viz. unburned and burned. Even though the assumption for the spatially uniformity of pressure is implemented, there is a large difference between the temperature of both zones. Each zone is assumed to be uniform in temperature. Thus, during combustion zone, two temperature values are used to specify the state of the contents of the cylinder. The mathematical treatment of the unburned zone is similar to the intake and compression phase of simulation. The burned zone is treated as a homogeneous mixture of ideal gases in chemical equilibrium.

2.2. Thermodynamic Analysis

An expression for the time rate of change of temperature is required to track the changes in the thermodynamic state of the cylinder charge. The equation must not be explicitly dependent on rate of change of pressure of the system. By tracking the
temperature change, change in pressure can be evaluated using the ideal gas law. These equations are applied to each of the four phases of the cylinder cycle.

The general conservation of energy equation for an open thermodynamic system is given by:

\[ \dot{E} = \sum \dot{m}_j h_j - \dot{Q}_w - \dot{W} \]  \hspace{1cm} (2.1)

Where,

\[ \dot{E} = \frac{d}{dt}(m h) - \frac{d}{dt}(p V) \] : Rate of change of energy

\[ \sum \dot{m}_j h_j \] : Net rate of influx of enthalpy (J/s)

\[ \dot{Q}_w \] : Rate of heat transfer (W)

\[ \dot{W} = p \dot{V} \] : Rate of doing work (J/s)

The dots denote differentiation with respect to time. The convention used is heat loss from the system and work done by the system is positive.

Assuming the fresh charge and the residual gas to be ideal gases, the enthalpy and gas constant of the mixture are given by:

\[ h = x_1 h_1 + x_2 h_2 \] \hspace{1cm} (2.2)

\[ R = x_1 R_1 + x_2 R_2 \] \hspace{1cm} (2.3)

Where,

\[ x_i \] : Mass fraction of component \( i \) (1: Fresh charge; 2: Residual gas).

For ideal gases,

\[ \frac{\dot{p}}{\rho} = \frac{\dot{T}}{T} + \frac{\dot{p}}{\rho} + \frac{\dot{R}}{R} \] \hspace{1cm} (2.4)
\[ \dot{h}_i = C_{p_i} \dot{T} + C_{T_i} \dot{p} \; ; \; i = 1, 2 \quad (2.5) \]

Here,

\[ C_p = \left( \frac{\partial h}{\partial T} \right)_p \]
\[ C_T = \left( \frac{\partial h}{\partial p} \right)_T \]
\[ \dot{p} = \left( \frac{\partial \rho}{\partial T} \right)_p \dot{T} + \left( \frac{\partial \rho}{\partial p} \right)_T \dot{p} \quad (2.6) \]

Combining the above equation, we get an expression for change of pressure as:

\[ \dot{p} = \frac{\rho}{\left( \frac{\partial \rho}{\partial T} \right)_T} \left\{ \left( \frac{R_i - R_j}{R} \right) \dot{x}_i - \frac{\dot{V}}{V} - \frac{1}{\rho} \left( \frac{\partial \rho}{\partial T} \right)_p \dot{T} + \frac{\dot{m}}{m} \right\} \quad (2.7) \]

Here \( p \) is the system pressure and \( \rho \) is the uniform density of the cylinder charge.

Substituting equations (2.2) to (2.7) into the energy equation (2.1), we get an expression for the rate of change of temperature of a system spatially uniform in pressure as:

\[ \dot{T} = \frac{B}{A} \left\{ \dot{x}_i \left( \frac{R_i - R_j}{R} + \frac{h_j - h_i}{B} \right) + \frac{\dot{m}}{m} \left( 1 - \frac{h}{B} \right) - \frac{\dot{V}}{V} + \frac{1}{Bm} \left( \sum \dot{m}_j h_j - \dot{Q}_w \right) \right\} \quad (2.8) \]

Where,

\[ A = C_p + \frac{\left( \frac{\partial \rho}{\partial T} \right)_p}{\left( \frac{\partial \rho}{\partial T} \right)_T} \left( \frac{1}{\rho} - C_T \right) \quad (2.9) \]
\[ B = \frac{1 - \rho C_T}{\left( \frac{\partial \rho}{\partial T} \right)_T} \quad (2.10) \]

Equation (2.8) is independent of the rate of change of pressure thus can be evaluated knowing only the thermodynamic properties of the contents. Applying (2.8) to individual phases of the engine cycle, we get:
Intake phase:

\[
\dot{T} = \frac{B}{A} \left\{ \dot{x}_1 \left( \frac{R_1 - R_2}{R} + h_2 - h_1 \right) + \frac{\dot{m}}{m} \left( 1 - \frac{h}{B} \right) - \frac{\dot{V}}{V} + \frac{1}{Bm} \left( \dot{m}_{in} h_{in} - \dot{m}_{ex} h_{ex} + \dot{Q}_w \right) \right\} \tag{2.11}
\]

Compression phase:

\[
\dot{T} = \frac{B}{A} \left\{ -\frac{\dot{V}}{V} - \frac{\dot{Q}_w}{Bm} \right\} \tag{2.12}
\]

Combustion & expansion phase:

i. Unburned zone:

\[
\dot{T}_u = \frac{B_u}{A_u} \left\{ \dot{x}_b - \frac{\dot{V}_u}{V_u} - \frac{\dot{Q}_{wu}}{B_u m_u} \right\} \tag{2.13}
\]

ii. Burned zone:

\[
\dot{T}_b = \frac{B_b}{A_b} \left\{ \dot{x}_b \left( 1 + \frac{h_u - h_b}{B_b} \right) - \frac{\dot{V}_b}{V_b} - \frac{\dot{Q}_{wb}}{B_b m_b} \right\} \tag{2.14}
\]

Exhaust phase:

\[
\dot{T} = \frac{B}{A} \left\{ -\frac{\dot{m}}{m} \frac{\dot{V}}{V} - \frac{\dot{Q}_w}{Bm} \right\} \tag{2.15}
\]

The thermodynamic properties required in the above equation depend on the instantaneous contents of the cylinder charge. Separate expressions are used for fresh charge and combustion products. Expressions for properties of combustion products were developed by Olikara and Borman [10] and modified by Martin and Heywood [11]. The implementation of these equations in a computer simulation is based on a code developed by Ferguson [7] and implemented in Matlab by Buttsworth [1]. Some additions to this
program had to be made to derive certain additional properties needed for the solution of above equations.

2.3. Gas Exchange Model

An equation for one dimensional quasi-steady compressible flow through a restriction is used to describe the mass flow taking place between the engine cylinder and intake/exhaust manifolds. The manifolds are treated as constant pressure-infinite volume plenums. An experimentally determined discharge coefficient $C_D$ imparts real gas flow effects to this ideal equation. The intake charge consists of a mixture of fuel, air and if present, EGR, all at known temperature. The exhaust charge is assumed to compose of only products of complete combustion at the instantaneous temperature inside the cylinder. Backflow into the intake manifold during early stages of intake is assumed to be at a temperature of the instantaneous charge inside the cylinder. Mixing between the fresh charge and the backflow mass is ignored.

The flow rate relationship in terms of upstream stagnation pressure $p_0$, upstream stagnation temperature $T_0$ and pressure downstream of restriction, which is assumed to be equal to pressure at the restriction $p_r$ is given by [5]:

$$
\dot{m} = \frac{C_D A_p p_0}{(RT_0)^{1/2}} \left( \frac{p_r}{p_0} \right)^{\gamma/2} \left\{ \frac{2\gamma}{\gamma-1} \left( 1 - \left( \frac{p_r}{p_0} \right)^{(\gamma-1)/\gamma} \right) \right\}^{1/2}
$$

For choked flow, $p_r / p_0 \leq \left[ 2 / (\gamma + 1) \right]^{\gamma / (\gamma - 1)}$. The equation above then changes to:
\[ \dot{m} = \frac{C_D A_R p_0}{(RT_0)^{1/2} \sqrt{\gamma}} \left( \frac{2}{\gamma + 1} \right)^{(\gamma+1)/(\gamma-1)} \] (2.17)

It is important to note that the choice of reference area \( A_R \) depends on the methodology used to determine the discharge coefficient. For the data available, we use the deference between the port flow area minus the sectional area of the valve stem such that:

\[ A_R = \frac{\pi}{4} \left( D_h^2 - D_s^2 \right). \] (2.18)

Mass flow to the cylinder through intake valve and from the cylinder through the exhaust valve is taken as positive in the simulation. The mass of the charge inside the cylinder at any time \( t \) during the simulation is:

\[ m(t) = m_0 + \int_{t_0}^{t} \dot{m}_i dt - \int_{t_0}^{t} \dot{m}_e dt \] (2.19)

Where, \( m_0 \) is the mass of the cylinder charge at the start of simulation (IVO).

2.4. Turbulent Flow Model

Estimates of characteristic velocity inside the cylinder and length scales are required during the combustion model and the heat transfer model. The turbulence model described here is as used by Poulos and Heywood [9]. It is based on the turbulence model developed by Mansouri et al. [12] for diesel engine cycle simulation.

A zero dimensional energy cascade is assumed to take place inside the cylinder. The supplied mean kinetic energy \( K \) is converted to turbulent kinetic energy \( k \) through turbulent dissipation. The rate of turbulent energy dissipation is given by \( \varepsilon \) which relates
to the turbulent intensity \( u' \). The turbulent kinetic energy inside the cylinder converts to heat through viscous dissipation. Figure (2.1) shows the schematic diagram of the cascade model.

\[ K = \frac{1}{2} mU^2 \]  \hspace{1cm} (2.20)

Also, the turbulent kinetic energy is related to the turbulent intensity \( u' \) by:

\[ k = \frac{3}{2} mu'^2 \]  \hspace{1cm} (2.21)

The factor 3 in (2.21) is from the assumption of isotropic turbulence, accounting for all 3 orthogonal directions.

The turbulent energy dissipation rate and the mean flow kinetic energy have to be calculated continuously since the flow field inside the cylinder is unsteady. The following equations give the time rate of change of mean and turbulent kinetic energy which are calculated at each time / crank angle step in the simulation:

\[ \frac{dK}{dt} = -\varepsilon \]  \hspace{1cm} (2.22)

\[ \frac{dk}{dt} = \frac{1}{3} \frac{dK}{dt} \]  \hspace{1cm} (2.23)

\[ \varepsilon = C_f \frac{m}{K} \]  \hspace{1cm} (2.24)

\[ C_f = 0.09 \]  \hspace{1cm} (2.25)

\[ \frac{dU}{dt} = \frac{1}{m} \frac{dK}{dt} \]  \hspace{1cm} (2.26)

\[ \frac{dU}{dt} = \frac{1}{m} \left( -\varepsilon + \frac{dK}{dt} \right) \]  \hspace{1cm} (2.27)

\[ \frac{dK}{dt} = \frac{1}{3} \left( -\varepsilon + \frac{dK}{dt} \right) \]  \hspace{1cm} (2.28)

\[ \frac{dU}{dt} = \frac{1}{3} \left( -\varepsilon + \frac{dK}{dt} \right) \]  \hspace{1cm} (2.29)

\[ \frac{dK}{dt} = \frac{1}{3} \left( -\varepsilon + \frac{dK}{dt} \right) \]  \hspace{1cm} (2.30)
\[ \frac{dK}{dt} = \frac{1}{2} \dot{m}_i V_i^2 - P - \frac{\dot{m}_i}{m} \]  \hspace{1cm} (2.22)

\[ \frac{dk}{dt} = P - m\varepsilon - k \frac{\dot{m}_i}{m} \]  \hspace{1cm} (2.23)

and

\[ \varepsilon = u'^3/l = (2k/3m)^{3/2}/l. \]

Here,

\( m \): Mass in cylinder (kg)

\( \dot{m}_i \): Mass flow rate into the cylinder (kg/s)

\( \dot{m}_e \): Mass flow rate out of the cylinder (kg/s)

\( V_i \): Jet velocity into the chamber (m/s)

\( P \): Rate of turbulent kinetic energy production (J/s)

\( \varepsilon \): Rate of turbulent kinetic energy dissipation per unit mass (J/kg-s)

\( l \): Characteristic size of large scale eddies (m)

The turbulent kinetic energy production is dependent on the local flow condition in the cylinder. However, since this model does not spatially resolve flow parameters, we have to express \( P \) in terms of mean flow quantities. The turbulence production is assumed similar to the turbulence production in a turbulent boundary layer over a flat plate. The turbulence production term is then given by [13]:

\[ P = \mu \left( \frac{\partial U}{\partial y} \right)^2. \]  \hspace{1cm} (2.24)
The turbulent viscosity $\mu_t$ is given by $\mu_t = c_\mu k^2 / (m \epsilon)$ where $c_\mu = 0.09$ is a universal constant. Due to lack of spatially resolved data, the velocity gradient in (2.24) is modeled as:

$$\left( \frac{\partial U}{\partial y} \right)^2 = c_\beta \left( \frac{U}{L} \right)^2$$  \hspace{1cm} (2.25)

$L$ is the geometric length scale of the engine cylinder. The factor $c_\beta$ is an adjustable constant used in the calibration of the simulation. Substituting (2.25) in to (2.24), we get an expression for the turbulent kinetic energy production:

$$P = 2(3/2)^{3/2} c_\mu c_\beta (K_l / L^2)(k / m)^{1/2}$$  \hspace{1cm} (2.26)

The simulation assumes that the macroscale of turbulence is equal to the geometric length scale given by:

$$l = L = V / (\pi B^2 / 4)$$  \hspace{1cm} (2.27)

Subject to condition that,

$$L \leq B / 2 .$$

Here, $V$ is the instantaneous cylinder volume and $B$ is the cylinder bore. Thus, we get a simplified equation for turbulent kinetic energy production as:

$$P = 0.3307 c_\beta (K / L)(k / m)^{1/2} .$$  \hspace{1cm} (2.28)

In the combustion phase, the turbulent intensity and the macroscale of turbulence is calculated using the law of conservation of angular momentum for large scale eddies. During combustion, the production term $P$ is assumed to be zero in the evolution of turbulent kinetic energy (2.23).
2.5. Heat Transfer Model

Due to the turbulent nature of flow inside the combustion chamber, the prime mode of heat transfer from the cylinder is convection. Radiation heat transfer and heat transfer in the exhaust manifold is neglected for the purposes of this simulation. The heat transfer rate due to convection can be expressed as:

\[ \dot{Q}_w = hA(T_g - T_w) \]  \hspace{1cm} (2.29)

Here,

\( \dot{Q}_w \): Heat transfer (W)

\( T_g, T_w \): Gas and wall temperature respectively (K)

\( h \): Convective heat transfer coefficient (W/m\(^2\)K)

\( A \): Surface area (m\(^2\))

The value of convective heat transfer coefficient is calculated using the flow parameters as [14]:

\[ Nu = a \operatorname{Re}^d \operatorname{Pr}^e \]  \hspace{1cm} (2.30)

Where,

\( Nu = hL / k \): Nusselt number

\( \operatorname{Re} = V_{ch}L / \nu \): Reynolds number

\( \operatorname{Pr} = \mu c_p / k \): Prandtl number = 1.0

\( a, d, e \): Constants used for calibration

\( L \): Characteristic length (m)

\( c_p \): Specific heat at constant pressure (J/Kg-K)
\( k \): Thermal conductivity (W/m-K)

\( V_{ch} \): Characteristic velocity (m/s)

\( \nu = \mu / \rho \): Kinematic viscosity (m\(^2\)/s)

\( \mu \): Dynamic viscosity (Pa-s)

\( \rho \): Density (kg/m\(^3\))

The characteristic length required in heat transfer calculations is same as that calculated by (2.27) used for turbulence production. To take into account the effect of turbulence, the characteristic velocity used above is expressed as an effective velocity due to the contributions of mean flow velocity \( U \), turbulent intensity \( u' \) and instantaneous piston speed \( V_p \):

\[
V_{ch} = \left( U^2 + u'^2 + \left( \frac{1}{2} V_p \right)^2 \right)^{1/2}.
\]

(2.31)

The viscosity and thermal conductivity needed for calculation of convective heat transfer coefficient are found from empirical correlations. For unburned gases, they are assumed to be same as that of air. For burned gas, correlations developed by Mansouri and Heywood [15] for hydrocarbon-air combustion products are used.

The gas temperature \( T_g \) is the mean temperature of the gas inside the combustion chamber. Heat transfer to the piston, cylinder head and cylinder walls is calculated separately and then added to give the total heat transfer. During intake, compression and exhaust, the surface area of heat transfer for cylinder head and piston is the total head and
piston area. The cylinder wall area changes with the crank angle position and can be calculated knowing the connecting rod and crank length.

During combustion, the heat transfer law (2.29) is applied separately to the unburned and the burned zone knowing the mean temperatures of both zones. The area of piston, cylinder head and cylinder wall exposed to each zone is interpolated by using the experimental flame geometry data.

2.6. Combustion Model

The combustion model used is a phenomenological model which combines fundamental knowledge with physics of combustion. It is a simplified model as compared to detailed model of turbulence which considers spatially varying quantities. The combustion model used here was first proposed by Blizard and Keck [3] and later extended by Tabacynski [4]. A simple turbulence model is added to link the in-cylinder flows to the combustion model. In the description below, the geometric flow field is described, and then the assumptions about the combustion mechanism are explained. Lastly, the equations governing the combustion mechanisms are explained.

2.6.1. Flow Field Assumptions

The turbulence spectrum in the current model is described by three length scales viz. macroscale of turbulence $L$, the Kolmogorov scale $\eta$ and the Taylor microscale $\lambda$. The characteristic velocity is the turbulence intensity $u'$. The flow field consists of coherent large scale eddies of the scale $L$. The size of the smallest eddies is given by the
Kolmogorov scale. Most of the dissipation of turbulent kinetic energy occurs at the scale of the smallest eddy size. The spacing between the vortex tubes, which are associated with the Kolmogorov scale, is given by the Taylor microscale $\lambda$.

The flow field in the combustion chamber is assumed to be isotropic and homogeneous. At the start of combustion, the values for the turbulence parameters viz. $u'$ and $L$ are given by the turbulence model. During combustion, the unburned gas is assumed to be compressed by the flame at a high rate; such that on the time rate of flame propagation, dissipation of turbulent kinetic energy in the large scale eddies can be neglected. Thus, the evolution of the macroscale of turbulence and turbulence intensity can be expressed assuming conservation of angular momentum for large scale eddies [6].

![Schematic of turbulent flame propagation process](image)

**Figure (2.3):** Schematic of turbulent flame propagation process [9].
2.6.2. Combustion Mechanism and Governing equations

The combustion is modeled as two simultaneous processes viz. entrainment of unburned gas and mass burn-up. The entrainment velocity is given by the sum of the convective component, the turbulent intensity and the diffusive component, the laminar flame speed. Mass conservation is applied to the moving flame front as:

\[ \dot{m}_e = \rho_u A_f (u' + S_L) \]  

(2.32)

Where,

\( \dot{m}_e \) : Rate of mass entrainment into the flame (kg/s)

\( \rho_u \) : Density of the unburned mass (kg/m\(^3\))

\( A_f \) : Outer area of the entrainment flame front (m\(^2\))

\( (u' + S_L) \) : Sum of turbulent intensity and laminar flame speed (m/s)

The laminar flame speed is calculated using empirical relation for the flame speed of iso-octane. A correction term is added as a function of burned gas fraction and is based on the data by Rhodes [16] for constant volume combustion. The expression for laminar flame speed is given as:

\[ \frac{S_L}{S_{L_w}} = \left( \frac{T_u}{T_w} \right)^a \left( \frac{p}{p_w} \right)^b (4.706 f^2 - 4.062 f + 1) \]  

(2.33)

Where,

\( T_u \) : Unburned gas temperature (K)

\( T_w \) : 298 K

\( p \) : Cylinder pressure (atm)
$p_\infty$: 1 atm

$S_{L_\infty}$: Laminar flame speed at $T_\infty$ and $p_\infty$ with $f = 0$

$\alpha, \beta$: Correlation parameters

$f$: Residual gas fraction

Considering reference laminar speed for iso-octane and using least-square polynomial fit to experimental data [5], we get:

$$S_{L_\infty} = 26.3 - 84.7(\phi - 1.13)^2 \quad (2.34)$$

With

$$\alpha = 2.18 - 0.8(\phi - 1)$$

$$\beta = 0.16 + 0.22(\phi - 1)$$

The second phase in the combustion is the burn-up of charge which occurs simultaneously with mass entrainment. Burning on the Kolmogorov scale is assumed instantaneous. On the Taylor microscale the burning is at the laminar flame speed. Therefore, the burned time for the entrained eddy is given by:

$$\tau = \frac{\lambda}{S_L} \quad (2.35)$$

With this two-step model, the burn-up of charge is limited by the rate of entrainment and by the time required for burn-up. The small eddies burn simultaneously. Thus, the overall mass burn-up rate is given by the total mass of unburned eddies within the flame divided by the burning time.
\[ \dot{m}_b = \frac{m_b - m_b}{\tau} \quad (2.36) \]

2.6.3. Evolution of Turbulence Parameters

Conservation of angular momentum is applied to large eddies individually. Assuming discrete coherent large eddies, as the flame propagates the mass of individual eddies is conserved. Conservation of mass implies that:

\[ \rho_u V_L = \rho_{uo} V_{Lo} \quad (2.37) \]

The subscript ‘0’ refers to conditions at the time of spark and \( \rho_u \) is the unburned gas density. Since \( V \sim L^3 \), we get:

\[ \frac{L}{L_o} = \left( \frac{\rho_{uo}}{\rho_u} \right)^{1/3} \quad (2.38) \]

Conservation of angular momentum of eddies requires that,

\[ u_{o0} L = u_{o00} L_o \quad (2.39) \]

Where \( u_{o0} \) is the characteristic velocity due to eddy vorticity. Assuming \( u_{o0} \sim u' \), and combining (2.39) and (2.38), we get an expression for the evolution of turbulence intensity during combustion:

\[ \frac{u'}{u'_o} = \left( \frac{\rho_u}{\rho_{uo}} \right)^{1/3} \quad (2.40) \]

The Taylor microscale is related to the macroscale of turbulence and the turbulent intensity by [13]:

\[ \frac{\lambda}{L} = \sqrt{15} \left( \frac{u'L}{A} \right)^{-1/2} \quad (2.41) \]
2.7. Method of Solution

The mathematical equations which describe the thermodynamics of engine operations need to be solved simultaneously at each time/crank angle step. Matlab’s inbuilt differential equation solver, ODE 45, is used to integrate the differential equations simultaneously. Each phase of the engine cycle, intake, compression, combustion and exhaust, are integrated separately. The integrated results of the previous phase are given as initial conditions for the next phase.
CHAPTER THREE
MATLAB SIMULATION OUTLINE

3.1. Simulation Framework

This chapter explains the outline of the simulation. When the thermodynamic principles explained in the previous chapter are resolved and analyzed, we get a list of differential equations which must be integrated simultaneously at each time step for the entire operating cycle. Some governing equations apply only during certain parts of the cycle such as mass flow rate applies only during intake and exhaust. Thus, equations such as these are programed as sub-routines to the main program to be called at appropriate times. The main program consists of a call function, which reads the user input data from the engine data input file and starts the simulation at IVO. Each phase of engine operation, viz. intake, compression, combustion, exhaust, is described by a different set of differential equations. Thus they are modeled in separate function files called by MATLAB’s ode45 solver. This structure allows for later addition of more outputs from a particular phase of engine operation. After end of cycle simulation, the results are stored in a .mat file for later use and also for plotting. The call to the plotting function gives the results of the cycle simulation. Figure (3.1) depicts the overall flowchart of the simulation with the basic subdivisions.
3.2. Simulation Inputs

This section discusses the input parameters that need to be specified depending on the test engine condition. Some of the inputs, which are experimentally found, can be found out by mathematical simulations which can be added to this simulation.
Engine Geometry Parameters

- Total Displacement (cm$^3$): 5300
- Compression Ratio: 9.95:1
- Bore (mm): 96
- Stroke (mm): 92
- Connecting Rod Length (mm): 155.1

Engine Thermo-Fluid Parameters

- Residual gas fraction: 0.1
- Fuel Type: Gasoline
- Equivalence Ratio ($\phi$): 1.0
- Engine Speed (RPM): 5000
- Cylinder Wall Temperature (K): 450
- Piston Surface Temperature (K): 450
- Cylinder Head Temperature (K): 450
- Exhaust Gas Recirculation (%): 0.0
- Heat Transfer Constant ($\alpha$): 0.1
- Heat Transfer Exponent ($d$): 0.65

Turbulence Parameters

- Turbulence Production Constant ($c_\beta$): 1.6
- Turbulence multiplier ($c_{mul}$): 1
- Mean Kinetic Energy at start (J): 0.002
- Turbulent Kinetic Energy at start (J): 0.001

Valve & Spark Timings

- Intake Valve Open (IVO): 281°
- Intake Valve Close (IVC): -116°
- Exhaust Valve Open (EVO): 153°
- Exhaust Valve Close (EVC): -251°
- Spark Timing (BTDC): 35°

Valve Geometry Specifications

- Intake Valve Head Diameter (mm): 50
- Exhaust Valve Head Diameter (mm): 38
- Intake Valve Stem Diameter (mm): 6
- Exhaust Valve Stem Diameter (mm): 6

Initial Conditions

- Pressure at start (IVO) (bar): 1.25
- Temperature at start (IVO) (K): 900
- Atmospheric Pressure (bar): 1.01325
- Atmospheric Temperature (K): 300
- Fresh Charge Temperature (K): 300
- Intake Manifold Pressure (bar): 0.4
- Exhaust Manifold Pressure (bar): 1.01325

Table (3.1): Sample simulation inputs
3.2.1. Engine Design Parameters

Engine design parameters required are engine geometrical parameters viz. engine bore, stroke, connecting rod length, compression ratio. These inputs are required for calculation for areas of heat transfer, for evaluation of instantaneous chamber height which is assumed to be equal to macroscale of turbulence and other geometrical parameters.

3.2.2. Operating Conditions

These include the fuel type (gasoline, ethanol, etc.). A list of fuels for which properties can be evaluated is given in fueldata.m. Operating conditions also include the engine speed, % EGR and operating fuel air equivalence ratio $\phi$.

3.2.3. Manifold Conditions

Intake and exhaust manifold pressures and temperatures need to be specified assuming the manifolds to be modeled as uniform pressure infinite volume plenums. Experimentally measured pressure values at each crank angle can be given as input to increase the accuracy of mass flow calculation. Ambient temperature and pressure have to be given as a reference.

3.2.4. Heat Transfer Parameters

This group contains the parameters associated with the specification of heat transfer characteristics of the simulation. Key parameters in this section are heat transfer
constant used in (2.30), $a$, and exponent for Reynolds number, $d$. These two parameters are used for calibration of the simulation to fit experimental data. Other variable which determine heat transfer characteristic are piston, cylinder head and wall temperatures. These temperatures are assumed constant over the entire cycle. The effect of turbulence of heat transfer is taken into account during the calculation of heat transfer coefficient based on the characteristic velocity.

### 3.2.5. Turbulence Parameters

The turbulence parameters, $c_\beta$ and $c_\mu$, have to be specified by the user. They are used for calibration purpose and are fixed for specific engine geometry. $c_\mu$, the turbulence multiplier, which acts during combustion stroke, is generally 1. The value of $c_\beta$ is decided based on the calibration of mass fraction burn curve.

### 3.2.6. Valve and Spark Timing Specifications

Valve open and close times for intake and exhaust stroke and spark time needs to be specified. These events are with the reference of TDC at compression as 0° CAD. Valve lift profiles and corresponding discharge coefficients are required as inputs to this simulation figure (3.2). However, they can be modeled mathematically which will give ideal values of mass flow through the valves. Experimental data was used for the simulation.
3.2.7. Flame Geometry

The flame geometry parameters required for calculation of mass entrained $m_e$ and evaluation of heat transfer in the two zone model have to be given in a specific format. These results are from a previous simulation to test the effect of combustion chamber geometry on SI engine combustion [6]. If detailed CAD model of combustion chamber is available, we can calculate the flame geometry based on the above mentioned simulation. Shown below is a part of the flame geometry data file which has to be given as an input to the simulation. This data is taken from a simulation study done on effect of combustion
chamber geometry on combustion in spark ignition engines [6] [9]. For additional information and method of evaluation of flame geometry, refer to the mentioned references. Each 50 data sets corresponded to a single CAD. The values of flame geometry variables are interpolated using burned volume and crank angle degrees as input. Figure (3.4) shows how the flame area varies with the variation in flame radius and crank angle position.

<table>
<thead>
<tr>
<th>Degree CA Step</th>
<th>Piston position for last data set</th>
<th>rft/bore normalized radius step</th>
<th>stroke/bore</th>
<th>cmmodlength/bore</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.000000</td>
<td>112.00000</td>
<td>.0525568</td>
<td>1.00000</td>
<td>1.60000</td>
</tr>
<tr>
<td>.000000</td>
<td>.000000</td>
<td>.000000</td>
<td>.097129</td>
<td>.000000</td>
</tr>
<tr>
<td>.025681</td>
<td>.004891</td>
<td>.000000</td>
<td>.097129</td>
<td>.000000</td>
</tr>
<tr>
<td>.051363</td>
<td>.016286</td>
<td>.000000</td>
<td>.097129</td>
<td>.000004</td>
</tr>
<tr>
<td>.076044</td>
<td>.034177</td>
<td>.000000</td>
<td>.097129</td>
<td>.0000965</td>
</tr>
<tr>
<td>.101725</td>
<td>.057536</td>
<td>.000000</td>
<td>.097129</td>
<td>.0000163</td>
</tr>
<tr>
<td>.126406</td>
<td>.088347</td>
<td>.000000</td>
<td>.097129</td>
<td>.0000436</td>
</tr>
<tr>
<td>.151088</td>
<td>.121552</td>
<td>.000000</td>
<td>.097129</td>
<td>.0000765</td>
</tr>
<tr>
<td>.175769</td>
<td>.165327</td>
<td>.000000</td>
<td>.097129</td>
<td>.00010488</td>
</tr>
<tr>
<td>.200450</td>
<td>.209108</td>
<td>.000000</td>
<td>.097129</td>
<td>.00015010</td>
</tr>
<tr>
<td>.225132</td>
<td>.252880</td>
<td>.000000</td>
<td>.097129</td>
<td>.00019963</td>
</tr>
<tr>
<td>.258813</td>
<td>.296652</td>
<td>.000000</td>
<td>.097129</td>
<td>.00025207</td>
</tr>
</tbody>
</table>

**Figure (3.3): Flame geometry data**

### 3.3. Simulation Output

The main simulation output is the crank angle by crank angle result. At the end of simulation, the pressure, temperature and heat transfer are returned resolved on a crank angle basis. We also get the mass fraction entrained and mass fraction burn profiles as results of combustion stoke simulation. During intake and exhaust, the mass flow rates are given as outputs.
Cycle performance results viz. total work done, thermal efficiency, indicated mean effective pressure, burn duration (MFB10-MFB90) and 50% MFB point are displayed in the MATLAB command window.

**Figure (3.4):** Normalized flame area versus flame radius for pent roof combustion chamber at different crank positions
CHAPTER FOUR

CALIBRATION OF MATLAB SIMULATION

4.1. Calibration Background

The calibration parameters have to be set so as to match with experimental data for the simulation to be used for predictive purpose. Simulation results were compared with measurements taken from a firing engine. The main parameter to be calibrated was the cylinder pressure and mass fraction burned values. The calibration parameters, heat transfer constant $a$, heat transfer exponent $d$ and the turbulence dissipation constant $c_\beta$, are adjusted to fit measurements at a specific operating condition. Then the simulation is checked by comparing it at different operating conditions. These calibration parameters are sensitive to the engine geometry. If the engine geometry changes, then the simulation results will gradually deteriorate depending upon the magnitude of change in the engine geometry. Some real world phenomenon such as variation in manifold pressure, crevice volume, accurate wall temperatures are not included in the simulation. Therefore, the simulation is calibrated to match the measured data. A more detailed model of the above mentioned variables can allow for a detailed calibration.

4.2. Description of Experimental Facility

The engine test data used for calibration was taken for a different study by Mr. Baitao Xiao and Dr. Robert Prucka. The engine testing was done in CU-ICAR in a FEV 580 HP AC Engine Dynamometer Test Cell. For data repeatability, test cell temperature
is closely controlled and engine is conditioned by specially designed systems intake air and fuel, oil. The measurements were from a GM 5.3 L V-8 engines running on gasoline. Data is collected by a state of art AVL-671 crank angle resolved 32-channel data acquisition system using piezoelectric pressure sensors. For combustion analysis, AVL Concerto is used which provides in-cylinder temperature data and 10%, 50% and 90% mass fraction burn positions. The ambient temperature is maintained at 30°C. The intake air temperature varies between 27-30°C. Piezoelectric pressure sensors are also used to measure the intake and exhaust port pressure. Intake pressure is controlled by the throttle which controls the loading of the engine. The data used for calibration is the average of 500 cycles taken at constant engine speed. Table 4.1 gives the geometrical specifications of the test engine. Valve timings are given in table 4.2.

**Figure (4.1):** Engine test cell at CU-ICAR (Photo Courtesy: Dr. Robert Prucka)
Table (4.1): Engine geometry specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Displacement (cm$^3$)</td>
<td>5300</td>
</tr>
<tr>
<td>Compression Ratio</td>
<td>9.95:1</td>
</tr>
<tr>
<td>Bore (mm)</td>
<td>96</td>
</tr>
<tr>
<td>Stroke (mm)</td>
<td>92</td>
</tr>
<tr>
<td>Connecting Rod Length (mm)</td>
<td>155.1</td>
</tr>
</tbody>
</table>

Table (4.2): Valve timings (referenced to TDC of compression as 0°)

<table>
<thead>
<tr>
<th>Valve Timing</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intake Valve Open (IVO)</td>
<td>281°</td>
</tr>
<tr>
<td>Intake Valve Close (IVC)</td>
<td>-116°</td>
</tr>
<tr>
<td>Exhaust Valve Open (EVO)</td>
<td>153°</td>
</tr>
<tr>
<td>Exhaust Valve Close (EVC)</td>
<td>-251°</td>
</tr>
</tbody>
</table>

4.3. Calibration of the Model

Calibration was done on the measurement data at 3000 RPM at 2/3 of full load. The turbulence dissipation constant $c_p = 1.65$ gives the best shape of mass fraction burned profile and agrees with the cylinder average 50% mass fraction burn point (MFB 50) at ~22° ATDC. The 10% mass fraction burn (MFB 10) position also agrees well with the values found from AVL Concerto. The heat transfer constant and exponent are used to calibrate the cylinder pressure data. A value of 0.65 for the heat transfer exponent and 0.1 heat transfer constant gave a good approximation of the cylinder pressure at the operating condition. The peak pressure occurs at ~5° after the measured pressure trace. Also, there is a slight difference between the rates of pressure rise during compression stroke. These discrepancies may be caused due to the mass lost in the crevice volume, which has not been considered as well as due to the heat transfer model. Also, a more
accurate pressure trace could be obtained by having crank angle based intake manifold pressure data.

Figure (4.2): Measured and predicted pressure trace for GM 5.3 L V-8 engine at 3000 RPM and spark at 22° BTDC.

There is a 1-2° CAD deviation in the MFB 50 point. This may be due to the approximation of flame geometry. From [17], we can see that there is a slight difference in the flame area and turbulence intensity values at different bore/stroke ratios. Thus the deviation in mass fraction burned profile may be due to the approximation of flame geometry data, which was obtained at slightly different bore/stroke ratio.
Figure (4.3): Predicted mass fraction entrained and burned profile for GM 5.3 L V-8 engine at 3000 RPM and spark at 22° BTDC.

4.4. Validation of Calibration

The first calibration was done at low speed and high load. To check the feasibility of calibration of the simulation, results of the simulation and experimental data were compared at different ‘high speed-low load’ configuration. Experimental data at 5000 RPM with 0.4 bars MAP (Manifold Average Pressure) were compared against the predicted results from the simulation. MFB 50 was observed at ~13° ATDC. There is a 2°
variation from AVL Concerto data. The pressure traces was found well in agreement with the experimental data, with ~9° over-prediction of peak pressure.

**Figure (4.4):** Predicted pressure trace for GM 5.3 L V-8 engine at 5000 RPM and spark at 35° BTDC
Figure (4.5): Predicted mass fraction entrained and burned profile for GM 5.3 L V-8 engine at 5000 RPM and spark at 35° BTDC.

The simulation is calibrated using experimental cylinder pressure data and 10%, 50%, 90% mass fraction burned crank angle positions evaluated from AVL Concerto. The simulation is able to predict the pressure and mass fraction burned traces well in accordance with the experimental data. The validation of the calibration shows that the calibration factor values can be used over the entire operating range of a particular engine. However, with change in engine geometry, the parameters will have to be recalibrated.
5.1. Introduction to Validation of Sub-models

The validation of the sub-models and other variables was done at 5000 RPM at wide open throttle (WOT) or full load. The main aim of this simulation was to evaluate the laminar flame speed at specific operating conditions. Laminar flame speed correlates directly to the mass burn up rate which is used to calibrate the spark timing for maximum efficiency and minimum emissions. This chapter discusses the variable other than pressure returned by the engine simulation.

![Figure (5.1): Predicted pressure trace for 5000 RPM WOT](image-url)
5.2. Main Variables

The pressure and temperature traces at 5000 RPM WOT are shown in figure (5.1) and figure (5.2). The pressure values are in accordance to what is expected. The rate of mass fraction burn is much faster at WOT.

![Graph showing pressure and temperature traces at 5000 RPM WOT](image)

**Figure (5.2):** Predicted temperature trace for 5000 RPM WOT

5.3. Gas Exchange Sub-models

The gas exchange during intake plays a key role in the evolution of turbulent intensity. The intake and exhaust manifolds are assumed as constant pressure infinite volume plenums. The pressure difference between the manifold and cylinder pressure
drive the flow past the valves. Figure (5.3) and figure (5.4) show the flow past intake and exhaust valves. The sudden increase in the mass flow rate at the start of exhaust stroke is due to the blow-down that occurs at the start of exhaust stroke after sudden opening of exhaust valve. This is a common phenomenon is SI engines due to high cylinder pressures at the time of EVO.

**Figure (5.3):** Mass flow rate through intake valve for 5000 RPM WOT
Figure(5.4) shows the change of fresh mass fraction in the engine cylinder during intake stroke. The calculation of residual mass fraction in the model adds additional accuracy to the intake stroke.

![Mass flow rate through exhaust valve for 5000 RPM WOT.](image)

**Figure (5.4):** Mass flow rate through exhaust valve for 5000 RPM WOT.

### 5.4. Turbulence Parameters

The turbulence parameters are the most important variable considered in this predictive burn rate simulation. The turbulence trend followed in a spark ignition engine has a steep increase during intake due to high jet velocity of the intake flow. This jet velocity is a function of the bore to stroke ratio and depends on the stroke length. There is a decay in the turbulence during compression. During combustion, the turbulence
intensity increases due to compression of unburned charge ahead of the moving flame front.

**Figure (5.5):** Turbulence intensity at 5000 RPM WOT

The second peak in the turbulence intensity corresponds to peak pressure point in the cylinder pressure. [17]. This expected trend is seen in the MATLAB simulation figure (5.6).
Figure (5.6): Evolution of turbulence length scales during combustion stroke at 5000 RPM (WOT)

The trend of turbulence length scales, the Taylor microscale $\lambda$ and the integral length scale $L$ during combustion are shown in figure (5.7). The Taylor microscale is proportional to the integral length scale but inversely proportional to the turbulence intensity.
Figure (5.7): Laminar flame speed trace during the combustion stroke at 5000 RPM WOT

The calculated laminar flame speed is shown in figure (5.8). The magnitude of the laminar flame speed is in good accordance with empirically fitted data. [5]. As the laminar flame speed correlated directly with the mass fraction burn profile through (2.36), manipulating the coefficients of reference laminar flame speed $\alpha$ and $\beta$ allows us to change the rate of mass burn.
CHAPTER SIX
CONCLUSION

6.1. Conclusion

A quasi-dimensional combustion simulation of spark ignition engine was done in MATLAB using earlier work done by Poulos [6] as reference. A model for spark ignition engine combustion has been done in MATLAB [1] but the model does not model the gas exchange phases of the engine cycle and uses a predetermined burn rate model. In this simulation the combustion is predictive in nature, modeled based on a two-step burn process [3] [4] of entrainment and burn-up of charge.

The model was calibrated at 3000 RPM for part load conditions. The calibration parameter \( c_{\beta} \) was set so as to match an approximate MFB 50 point. The heat transfer constant and exponent were calibrated to match the pressure trace at the operating point. After calibration, the model gave a good agreement with the experimental data at different speed-load configuration.

Validation of gas exchange and turbulence sub-models is also done which shows the expected trends in flow rates across valves, turbulent intensity and laminar flame speed values.

The key criteria to be considered in this simulation were the turbulence parameters and laminar flame speed. Experimental measurement of laminar flame speed is done in a spherical combustion bomb. It is difficult to measure actual value of laminar flame speed in engine combustion chamber. Measurement of turbulent intensity is also
difficult due to the conditions the sensor has to withstand inside the combustion chamber. Getting an estimate of laminar flame speed is useful as it closely related to the mass fraction burn. Assuming the variation in turbulent intensity to be independent of the fuel, we can manipulate the mass burn rate by manipulating / adjusting the coefficients of laminar flame speed.

The calibration parameters are sensitive to the engine geometry. For different engine geometry, after specifying required inputs, the model needs to be calibrated for test load-speed configurations and then can be used over the entire operating range of the particular engine.

6.2. Future Work

The simulation done in this work is a quasi-dimensional engine simulation with a phenomenologcial model for combustion. It can be used as a base model for more advanced engine simulations which take into account other engine parameters in detail. Some future work ideas based on this simulation are as follows:

- Addition of detailed models for intake and exhaust manifolds to better predict the mass flow rates through the valves
- Use the calibrated simulation to match the MFB of a different fuel to experimental data by manipulation of laminar flame speed $S_L$, assuming turbulent intensity to be fuel independent
- The output laminar flame speed of the fuel from above mentioned simulation can be used in the development of multi fuel adaptive engine by providing data for
model based combustion phasing algorithms which manipulate the engine operation variables (spark time, valve time in VVT engine, etc) to give maximum efficiency and minimum emissions for the input fuel.
APPENDICES
## Appendix A

### List of Programs

<table>
<thead>
<tr>
<th>FUNCTION/SCRIPT FILE</th>
<th>DESCRIPTION</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>enginedata</td>
<td>Input script file to specify engine parameters</td>
<td>50</td>
</tr>
<tr>
<td>main</td>
<td>Main program</td>
<td>53</td>
</tr>
<tr>
<td>Intake</td>
<td>Thermodynamic model of intake stroke</td>
<td>58</td>
</tr>
<tr>
<td>Compression</td>
<td>Thermodynamic model of compression stroke</td>
<td>62</td>
</tr>
<tr>
<td>Combustion</td>
<td>Thermodynamic model of combustion stroke</td>
<td>65</td>
</tr>
<tr>
<td>Exhaust</td>
<td>Thermodynamic model of exhaust stroke</td>
<td>69</td>
</tr>
<tr>
<td>Airdata</td>
<td>Polynomial coefficients for evaluation of thermodynamic properties of air</td>
<td>72</td>
</tr>
<tr>
<td>fueldata</td>
<td>Polynomial coefficients for thermodynamic properties of different fuels</td>
<td>75</td>
</tr>
<tr>
<td>farg</td>
<td>Evaluate thermodynamic properties of fuel+air+residual gas mixture</td>
<td>77</td>
</tr>
<tr>
<td>ecp</td>
<td>Evaluate thermodynamic properties of equilibrium combustion products</td>
<td>80</td>
</tr>
<tr>
<td>flowrate</td>
<td>Calculates mass flow rate across intake &amp; exhaust valves</td>
<td>85</td>
</tr>
<tr>
<td>flamespeed</td>
<td>Calculate laminar flame speed with RGF correction</td>
<td>87</td>
</tr>
<tr>
<td>flamegeometry</td>
<td>Interpolates flame geometry data from input file</td>
<td>88</td>
</tr>
<tr>
<td>heat_unburn</td>
<td>Calculates dynamic viscosity and thermal conductivity of unburned gas</td>
<td>97</td>
</tr>
<tr>
<td>heat_burn</td>
<td>Calculates dynamic viscosity and thermal conductivity of burned gas</td>
<td>98</td>
</tr>
<tr>
<td>Tadiabatic</td>
<td>Evaluation adiabatic flame temperature at the start of combustion</td>
<td>99</td>
</tr>
<tr>
<td>plot_main</td>
<td>Plots pressure, temperature, mass fraction burned and entrained</td>
<td>100</td>
</tr>
<tr>
<td>mass_flow_plot</td>
<td>Plots mass flow rates through intake and exhaust valve</td>
<td>102</td>
</tr>
<tr>
<td>mass_flow</td>
<td>Function used by mass_flow_plot.m</td>
<td>106</td>
</tr>
<tr>
<td>uprime_SL_plot</td>
<td>Evaluates and plots turbulent intensity and laminar flame speed</td>
<td>108</td>
</tr>
<tr>
<td>length_scale_plot</td>
<td>Plots integral length scale and Taylor microscale during combustion</td>
<td>109</td>
</tr>
</tbody>
</table>
### Appendix B

Sub-Models Used by Main Functions

<table>
<thead>
<tr>
<th>MAIN FUNCTION</th>
<th>SUB-MODELS REQUIRED</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>main</code></td>
<td>enginedata</td>
</tr>
<tr>
<td></td>
<td>intake</td>
</tr>
<tr>
<td></td>
<td>compression</td>
</tr>
<tr>
<td></td>
<td>Tadiabatic</td>
</tr>
<tr>
<td></td>
<td>combustion</td>
</tr>
<tr>
<td></td>
<td>exhaust</td>
</tr>
<tr>
<td><code>Intake</code></td>
<td>farg</td>
</tr>
<tr>
<td></td>
<td>flowrate</td>
</tr>
<tr>
<td></td>
<td>heat_unburn</td>
</tr>
<tr>
<td><code>Compression</code></td>
<td>farg</td>
</tr>
<tr>
<td></td>
<td>heat_unburn</td>
</tr>
<tr>
<td><code>Combustion</code></td>
<td>farg</td>
</tr>
<tr>
<td></td>
<td>ecp</td>
</tr>
<tr>
<td></td>
<td>heat_unburn</td>
</tr>
<tr>
<td></td>
<td>heat_burn</td>
</tr>
<tr>
<td></td>
<td>flamegeometry</td>
</tr>
<tr>
<td></td>
<td>flamespeed</td>
</tr>
<tr>
<td><code>Exhaust</code></td>
<td>ecp</td>
</tr>
<tr>
<td></td>
<td>farg</td>
</tr>
<tr>
<td></td>
<td>heat_burn</td>
</tr>
<tr>
<td></td>
<td>flowrate</td>
</tr>
</tbody>
</table>
Appendix C

MATLAB Function and Script Files

Enginedata.m

% enginedata.m
% Script file used by the function main.m to define engine parameters and other constants. Engine flame geometry file is also used to evaluate head and piston areas.

-- ENGINE GEOMETRY PARAMETERS --

b=96e-3; % Engine bore (m)
stroke=92e-3; % Engine stroke (m)
conrodl = 155.1e-3; % Connecting rod length (m)
eps=stroke/(2*conrodl); % Half stroke to rod ratio, s/2l
r=9.95; % Compression ratio
Vtdc=(pi/4*b^2*stroke)/(r-1); % Volume at TDC (m^3)
Vbdc=pi/4*b^2*stroke+Vtdc; % Volume at BDC (m^3)
GEO=importdata('GEO-PENT.txt',' ',1); % Read flame geometry data
A_head_normalized = max(GEO.data(:,3)); % Cylinder head area normalized
A_piston_normalized = max(GEO.data(:,5)); % Piston area normalized
Area_head = A_head_normalized*b*b; % Cylinder head area
Area_piston = A_piston_normalized*b*b; % Piston area

-- ENGINE THERMO-FLUID PARAMETERS --

f=0.1; % residual gas fraction
fueltype='gasoline';
airscheme='GMcB';
phi=1.0; % Equivalence ratio
RPM=5000; % Engine speed (RPM)
omega=RPM*pi/30; % Engine speed in rad/s
Tw=450; % Cylinder wall surface temperature (K)
Tpiston=450; % Piston surface temperature (K)
Thead=450; % Cylinder head surface temperature (K)
EGR=0.0; % Exhaust Gas Recirculation (%) htran_con=0.1; % Heat transfer constant
htran_exp=0.65; % Heat transfer exponent

-- TURBULENCE PARAMETERS --

cbeta = 1.6; % Turbulence production factor
cmult = 1; % Turbulence multiplier
mean_ke_start = 25e-4; % Mean Flow Kinetic Energy (J)
turb_ke_start = 11e-4; % Turbulent Kinetic Energy (J)
% All timings are referenced considering TDC as -360 deg CAD at start of simulation.

intake_valve_open = 281.0; % Intake valve open
intake_valve_close = -116.0; % Intake valve close
spark_timing = -35.0; % Spark timing
exhaust_valve_open = 153.0; % Exhaust valve open
exhaust_valve_close = -251.0; % Exhaust valve close

% Covert valve/spark timing from degree to radian:
IVO = intake_valve_open*pi/180;
IVC = intake_valve_close*pi/180;
thetas = spark_timing*pi/180;
EVO = exhaust_valve_open*pi/180;
EVC = exhaust_valve_close*pi/180;

% VALVE GEOMETRY SPECIFICATIONS
D_head_inlet = 50.3e-3; % Inlet valve head diameter (m)
D_head_exhaust = 38.9e-3; % Exhaust valve head diameter (m)
D_stem_inlet = 6e-3; % Intake valve stem diameter (m)
D_stem_exhaust = 6e-3; % Exhaust valve stem diameter (m)

% INITIAL CONDITIONS
pi=1.25e5; % Starting pressure (Pa)
T1=900; % Starting temperature (K)
patm = 1.01325e5; % Atmospheric pressure (Pa)
Tatm = 300; % Atmospheric Temperature (K)
Tfresh = 300; % Fresh charge temp (K)
Tegr = 300; % Residual gas fraction temperature (K)
Piman = 0.4e5; % Intake manifold pressure (pa)
Peman = patm; % Exhaust Manifold pressure (pa)

% THERMODYNAMIC PROPERTIES
[h1,u1,v1,s1,Y1,cp1,dlvlT1,dlvlp1,rho1,DRHODT1,DRHODP1,ct1,ADUMY1,BDUMY1] = ...
    farg(pi,T1,phi,1.0,fueltype,airscheme);

V_IVO = Vtdc*(1+(r-1)/2*(1-cos(IVO)+ ...
        1/eps*(1-(1-eps^2*sin(IVO).^2).^0.5)));

mass1=V_IVO/v1;
U1=u1*mass1;
dt\theta = \frac{60}{2\pi \text{RPM}}; \% \text{To convert from time to CAD derivative}
Q_{\text{fuel\_gasoline}} = 44.4e6; \% \text{Lower heating value of gasoline (J/kg)}
MFB_{\text{spark}} = 0.001; \% \text{Initial value of MFB as initial condition}
Main.m

% main.m

% Script file to determine the performance variables of a spark
% ignition engine using a quasi-dimensional thermodynamic simulation.
% The combustion is modeled using an entrainment model. 2 zone heat
% release model is incorporated.

% ********************************************************************
% Input:
% enginedata.m - this is another script file that defines all of the
% relevant engine parameters and operating conditions.
% output:
% main.mat - this file contains all of the variables. For plotting
% the results, see the example script file plotresults.m
% ********************************************************************

global b stroke eps r Cblowby f fueltype airscheme phi RPM thetas
thetab...
  omega heattransferlaw hcu hcb Tw Tpiston Thead Vtdc Vbdc...
  mass1 p1 T1 V1 dtdtheta EGR Texh IVC IVO EVO EVC conrod1 Piman...
  Peman D_head_inlet D_head_exhaust mass_array theta_array...
  Q_fuel_gasoline MFB_spark cbeta him gammaim Tfresh Rim hob_spark...
  rhou_spark macro_length_spark uprime_spark cmult ...
  D_stem_exhaust D_stem_inlet htran_con htran_exp Area_head
  Area_piston counter...
  Area_flame_entrain_stop theta_entrain_stop Area_flame_true
  theta_true...
  data_EV_lift data_EV_flow data_IV_lift data_IV_flow ...

clc;
warning off MATLAB:xlsread:ActiveX;
warning off MATLAB:nearlySingularMatrix;

disp(['--- QUASI-DIMENSIONAL COMBUSTION SIMULATION OF SI ENGINE ---
  ']);

% ---------------------------------------------------------------
% LOAD ENGINE AND INTEGRATION PARAMETERS
% ---------------------------------------------------------------
enginedata

% THERMODYNAMIC PROPERTY EVALUATION OF CHARGE IN INTAKE MANIFOLD:
[him,uim,vim,sim,Yim,cpim,dlvlTim,dlvlpim,rhoim,DRHODTim,DRHODPim,...
  ctim,ADUMYim,BDUMYim,gammaim,Rim] = ...
  farg(Piman,Tfresh,phi,0,fueltype,airscheme);

% INTEGRATION PARAMETERS:
counter=1;
dtheta=1*pi/180;
options=odeset('RelTol',1e-3);
% LOAD VALVE LIFT AND DISCHARGE COEFFICIENT DATA
%
% data_IV_lift = xlsread('Intake_Data_Final.xls');
data_IV_flow = importdata('Flow_coeff_intake.txt');
data_EV_lift = xlsread('Exhaust_Data_Final.xls');
data_EV_flow = importdata('Flow_coeff_exhaust.txt');

if error_p > 0.1 || error_T>0.1 || error_MKE>0.1 || error_TKE>0.1

% INTAKE STROKE
%
% disp(['Performing calculations for : Intake (IVO - IVC)']);

% INTAKE STROKE BEFORE TDC:
[thetaintake_BTDC,data_intake_BTDC] = ode45('Intake',...[IVO:dtheta:2*pi], ...[p1 T1 0 0 0 mean_ke_start turb_ke_start 0 0],options);

% SET INITIAL CONDITIONS FOR THE REMAINING INTAKE:
y_in(1) = interp1(thetaintake_BTDC,data_intake_BTDC(:,1),2*pi);
y_in(2) = interp1(thetaintake_BTDC,data_intake_BTDC(:,2),2*pi);
y_in(3) = interp1(thetaintake_BTDC,data_intake_BTDC(:,3),2*pi);
y_in(4) = interp1(thetaintake_BTDC,data_intake_BTDC(:,4),2*pi);
y_in(5) = interp1(thetaintake_BTDC,data_intake_BTDC(:,5),2*pi);
y_in(6) = interp1(thetaintake_BTDC,data_intake_BTDC(:,6),2*pi);
y_in(7) = interp1(thetaintake_BTDC,data_intake_BTDC(:,7),2*pi);
y_in(8) = interp1(thetaintake_BTDC,data_intake_BTDC(:,8),2*pi);
y_in(9) = interp1(thetaintake_BTDC,data_intake_BTDC(:,9),2*pi);

% INTAKE STROKE AFTER TDC:
[thetaintake,data_intake] = ode45('Intake', ...[-2*pi:dtheta:IVC], ...[y_in(1) y_in(2) y_in(3) y_in(4) y_in(5) y_in(6) ...y_in(7) y_in(8) y_in(9)],options);

% Calculation of Turbulent Intensity:
mass_intake_BTDC= mass1 + data_intake_BTDC(:,7) -
data_intake_BTDC(:,8);
mass_intake = mass1 + data_intake(:,7) - data_intake(:,8);
uprime_intake_BTDC =
sqrt((2/3)*(data_intake_BTDC(:,6)./mass_intake_BTDC));
uprime_intake =
sqrt((2/3)*(data_intake(:,6)./mass_intake));

% SET INITIAL CONDITIONS FOR START OF COMPRESSION:
p_comp = interp1(thetaintake,data_intake(:,1),IVC);
T_comp = interp1(thetaintake,data_intake(:,2),IVC);
W_comp = interp1(thetaintake,data_intake(:,3),IVC);
Q_comp = interp1(thetaintake,data_intake(:,4),IVC);
MKE_comp = interp1(thetaintake,data_intake(:,5),IVC);
TKE_comp = interp1(thetaintake,data_intake(:,6),IVC);
mass_in_comp = interp1(thetaintake,data_intake(:,7),IVC);
mass_ex_comp = interp1(thetaintake, data_intake(:,8), IVC);
mass_fresh_comp = interp1(thetaintake, data_intake(:,9), IVC);

%-----------------------------------------------------------------------------------------
%                             COMPRESSION STROKE                                           
%-----------------------------------------------------------------------------------------

disp('Performing calculations for: Compression (IVC - Spark)');

[thetacomp, data_comp] = ode45('Compression', ...
    [IVC:dtheta:thetas], ...
    [p_comp T_comp W_comp Q_comp MKE_comp TKE_comp ... 
     mass_in_comp mass_ex_comp mass_fresh_comp], ...
    options);

% CONDITIONS AT THE END OF COMPRESSION:
pb = interp1(thetacomp, data_comp(:,1), thetas);
Tub_spark = interp1(thetacomp, data_comp(:,2), thetas);
Wb = interp1(thetacomp, data_comp(:,3), thetas);
Qlb = interp1(thetacomp, data_comp(:,4), thetas);
mean_kinetic_energy_spark = interp1(thetacomp, data_comp(:,5), thetas);
turb_kinetic_energy_spark = interp1(thetacomp, data_comp(:,6), thetas);
mass_in_spark = interp1(thetacomp, data_comp(:,7), thetas);
mass_ex_spark = interp1(thetacomp, data_comp(:,8), thetas);
mass_fraction_fresh = interp1(thetacomp, data_comp(:,9), thetas);

% Properties of unburned zones at spark:

[hu, uu, v, s, Y, cpu, dlv1Tu, dlv1pu, rhou, DRHODTU, DRHODPU, ctu, ADUMYU, BDUMYU, 
gammau, Ru] = farg(pb, Tub_spark, phi, f, fueltype, airscheme);
rhou_spark = rhou;

% Initialization for Combustion Stroke:
mass_spark = mass1 + mass_in_spark - mass_ex_spark;
V_spark = Vtdc*(1+(r-1)/2*(1-cos(thetas) ...
    + 1/eps*(1-(1-eps^2*sin(thetas).^2).^0.5)));
p_spark = pb + (mass_spark*MFB_spark*Q_fuel_gasoline*(gammau-1))/V_spark;
mass_fraction_entrained_spark = MFB_spark;
mass_fraction_burn_spark = MFB_spark;

% Burned zone temperature (K):
Tb_spark = Tadiabatic(p_spark, Tub_spark, phi, f, fueltype, airscheme);

% Calculation of Turbulent Intensity in compression phase:
mass_comp = mass1 + pTuWQlHl(:,7) - pTuWQlHl(:,8);
uprime_comp = sqrt((2/3)*(pTuWQlHl(:,6)./mass_comp));

% Calculation of Turbulent Intensity at spark:
mass_spark = mass1 + mass_in_spark - mass_ex_spark;
uprime_spark = sqrt((2/3)*(turb_kinetic_energy_spark/mass_spark));

% Properties of burned zones at spark:

[hb,ub,vb,s,y,cpb,dlv1Tb,dlv1pb,rhob,DRHODTB,DRHODPB,ctb,ADUMYB, BDUMYB, gammab,Rb] = ... 
ecp(pb,Tb_spark,phi,fueltype,airscheme);

% STORE TURBULENCE PARAMETER VALUES AT SPARK:
mass_spark = mass1 + mass_in_spark - mass_ex_spark;
uprime_spark = sqrt((2/3)*(turb_kinetic_energy_spark/mass_spark));
macro_length_spark = V_spark/(0.25*pi*b*b);
rhob_spark = rhob;
V_unburn_spark = V_spark - ((mass_fraction_burn_spark*mass_spark)/rhob_spark);

% SET INITIAL CONDITIONS FOR START OF COMBUSTION:
p_comb = pb;
Tu_comb = Tub_spark;
Tb_comb = Tb_spark;
W_comb = Wb;
Q_comb = Qlb;
MKE_comb = mean_kinetic_energy_spark;
TKE_comb = turb_kinetic_energy_spark;
mass_in_comb = mass_in_spark;
mass_ex_comb = mass_ex_spark;
mass_fresh_comb = mass_fraction_fresh;
mass_entrain_comb = mass_fraction_entrained_spark;
mass_burn_comb = mass_fraction_burn_spark;
V_unburn_comb = V_unburn_spark;

%-----------------------------------------------------------------
%                         COMBUSTION STROKE                       
%-----------------------------------------------------------------

disp(['Performing calculations for : Combustion (Spark - EVO)']);
[thetacomb,pTbTuWQlHl] = ode45('RatesComb', ... 
[thetas:dtheta:EVO], ... 
[p_comb Tu_comb Tb_comb W_comb Q_comb ... 
MKE_comb TKE_comb mass_in_comb mass_ex_comb ... 
mass_fresh_comb mass_entrain_comb ... 
mass_burn_comb V_unburn_comb], options);

% SET INITIAL CONDITIONS FOR START OF EXHAUST:
pe = interp1(thetacomb,pTbTuWQlHl(:,1),EVO);
Tbe = interp1(thetacomb,pTbTuWQlHl(:,3),EVO);
We = interp1(thetacomb,pTbTuWQlHl(:,4),EVO);
Qle = interp1(thetacomb,pTbTuWQlHl(:,5),EVO);
MKE_e = interp1(thetacomb,pTbTuWQlHl(:,6),EVO);
TKE_e = interp1(thetacomb,pTbTuWQlHl(:,7),EVO);
mass_in_e = interp1(thetacomb,pTbTuWQlHl(:,8),EVO);
mass_ex_e = interp1(thetacomb,pTbTuWQlHl(:,9),EVO);

%-------------------------------------------------------------------
% EXHAUST STROKE

disp(['Performing calculations for : Exhaust (EVO - EVC)]);

[thetaexhaust, data_exhaust] = ode45('RatesExhaust', ...
    [EVO:dtheta:IVO], ...
    [pe Tbe We Qle MKE_e TKE_e mass_in_e mass_ex_e],options);

% SET INITIAL CONDITIONS IF ERROR IS GREATER THAN STATED VALUE
p_ex = interp1(thetaexhaust, data_exhaust(:,1), IVO);
T_ex = interp1(thetaexhaust, data_exhaust(:,2), IVO);
W_ex = interp1(thetaexhaust, data_exhaust(:,3), IVO);
Q_ex = interp1(thetaexhaust, data_exhaust(:,4), IVO);
MKE_ex = interp1(thetaexhaust, data_exhaust(:,5), IVO);
TKE_ex = interp1(thetaexhaust, data_exhaust(:,6), IVO);
mass_in_ex = interp1(thetaexhaust, data_exhaust(:,7), IVO);
mass_ex_ex = interp1(thetaexhaust, data_exhaust(:,8), IVO);

error_p = abs((p_ex - p1)/p1);
error_T = abs((T_ex - T1)/T1);
error_MKE = abs((MKE_ex - mean_ke_start)/mean_ke_start);
error_TKE = abs((TKE_ex - turb_ke_start)/turb_ke_start);

p1 = p_ex;
T1 = T_ex;
mean_ke_start = MKE_ex;
turb_ke_start = TKE_ex;
end

mass4 = mass1 + mass_in_ex - mass_ex_ex;

[h4, u4, v4, s4, Y4, cp4, dlv1T4, dlv1p4] = ... 
farg(p_ex, T_ex, phi, 1, fueltype, airscheme);
U4 = u4 * mass4;

% indicated mean effective pressure and thermal efficiency
imep = W_ex / (pi * b^2 / 4 * stroke);
eta = W_ex / mass1 * (1 + phi * 0.06548 * (1 - f)) / phi / 0.06548 / (1 - f) / 47870 / 1e3;

% save all data
save main.mat
clear
function yprime=Intake(theta,y,flag);
  
  yprime=Intake(theta,y,flag)
  
  Function that returns the derivatives of the following 9 variables
  w.r.t. crank angle for the intake stroke:
  1) pressure (pa);
  2) unburned temperature (K);
  3) work (J);
  4) heat transfer (J);
  5) mean kinetic energy (J);
  6) turbulent kinetic energy (J);
  7) Mass induced through intake valve (kg);
  8) Mass exhausted through exhaust valve (kg);
  9) Mass fraction of fresh charge (-);

  global b stroke eps r Cblowby f fueltype airscheme phi RPM conrodl... 
  thetas thetab omega heattransferlaw hcu htran_con htran_exp... 
  Tw Tpiston Thetad thetal Vtdc massl Piman Peman Tfresh T1 Texh... 
  cbeta EVO EVC D_head_inlet D_head_exhaust D_stem_inlet... 
  D_stem_exhaust dtdtheta EGR him gammaim Rim Area_head Area_piston...

  p = y(1);
  Tu = y(2);
  yprime = zeros(9,1);

  % THERMODYNAMIC PROPERTY EVALUATION:
  [h,u,v,s,Y,cp,dlvlT,dlvlp,rho,DRHODT,DRHODP,ct,ADUMY,BDUMY,gamma,R] = 
  ... 
  farg(p,Tu,phi,f,fueltype,airscheme);

  % EVALUATE HEAT TRANSFER PROPERTIES OF CYLINDER CHARGE:
  [dyvisc,thcon]=heat_unburn(Tu);
  kivisc = dyvisc/rho;

  % MASS IN CYLINDER (kg):
  mass = massl + y(7) - y(8);

  % CYLINDER VOLUME:
  V = Vtdc*(1+(r-1)/2*(1-cos(theta))+ ... 
  1/eps*(1-(1-eps^2*sin(theta).^2).^0.5))); 

  % DERIVATIVE OF VOLUME (CAD AND TIME):
  dVdtheta = Vtdc*(r-1)/2*(sin(theta)+ ... 
  eps/2*sin(2*theta)./sqrt(1-eps^2*sin(theta).^2));
  Vdot = dVdtheta/dtdtheta;
% READ INTAKE VALVE DATA:
theta_deg = theta*180/pi; % Convert CAD to degrees for look up
if theta_deg == 360
    theta_deg = -360;
end
data_IV_lift = xlsread('Intake_Data_Final.xls');
IV_CAD = data_IV_lift(:,1); % Reference CAD input
IV_lift = data_IV_lift(:,2); % Lift (mm)
lift_poly_IV = interp1(IV_CAD,IV_lift,'cubic','pp');
lift_IV_mm = ppval(lift_poly_IV,theta_deg); % Lift value for theta
lift_IV = lift_IV_mm/1000; % Valve lift in meters
data_IV_flow = importdata('Flow_coeff_intake.txt');
IV_std_lift = data_IV_flow(:,2); % Reference lift (mm)
IV_CD = data_IV_flow(:,3); % Discharge coefficient
CD_lift_poly_IV = interp1(IV_std_lift,IV_CD,'cubic','pp');
Cd_IV = ppval(CD_lift_poly_IV,lift_IV_mm) ; % Discharge coeff value for lift at theta

% EVALUATE MASS FLOW RATE THROUGH INTAKE VALVE:
if p <= Piman % flow INTO the cylinder (positive)
    [flowrate_in,flow_area_in] = ...
    flowrate(Cd_IV,D_head_inlet,D_stem_inlet,lift_IV,Piman,Tfresh,p,gammaim,Rim);
yprime(7) = flowrate_in;
VIV = flowrate_in/(rho*flow_area_in); % (m/s)
yprime(5)= 0.5*flowrate_in*VIV*VIV; % (J/s)
else % flow OUT of the cylinder (negative)
    [flowrate_in_reverse,flow_area_in_reverse] = ...
    flowrate(Cd_IV,D_head_inlet,D_stem_inlet,lift_IV,p,Tu,Piman,gamma,R);
yprime(7) = -flowrate_in_reverse;
VIV = -flowrate_in_reverse/(rho*flow_area_in_reverse); % (m/s)
yprime(5)= -flowrate_in_reverse*(y(5)/mass); % (J/s)
yprime(6)= -flowrate_in_reverse*(y(6)/mass); % (J/s)
end

% READ EXHAUST VALVE DATA:
data_EV_lift = xlsread('Exhaust_Data_Final.xls');
EV_CAD = data_EV_lift(:,1); % Reference CAD input
EV_lift = data_EV_lift(:,2); % Lift (mm)
lift_poly_EV = interp1(EV_CAD,EV_lift,'cubic','pp');
lift_EV_mm = ppval(lift_poly_EV,theta_deg); % Lift value for theta
lift_EV = lift_EV_mm/1000; % Valve lift in meters
data_EV_flow = importdata('Flow_coeff_exhaust.txt');
EV_std_lift = data_EV_flow(:,2); % Reference lift (mm)
EV_CD = data_EV_flow(:,3); % Discharge coefficient
CD_lift_poly_EV = interp1(EV_std_lift,EV_CD,'cubic','pp');
Cd_EV = ppval(CD_lift_poly_EV,lift_EV_mm) ; % Discharge coeff value for lift at theta
% EVALUATE MASS FLOW RATE THROUGH EXHAUST VALVE:
if p >= Peman % Flow OUT of cylinder (positive)
    [flowrate_out, flow_area_out] = ...
    flowrate(Cd_EV, D_head_exhaust, D_stem_exhaust, lift_EV, p, Tu, Peman, gamma, R);
    yprime(8) = flowrate_out;
    VEV = flowrate_out/(rho*flow_area_out);
    yprime(5) = yprime(5) - (flowrate_out*(y(5)/mass));
    yprime(6) = yprime(6) - (flowrate_out*(y(6)/mass));
else % Flow INTO the cylinder (negative)
    [flowrate_out_reverse, flow_area_out_reverse] = ...
    flowrate(Cd_EV, D_head_exhaust, D_stem_exhaust, lift_EV, Peman, Tu, p, gamma, R);
    yprime(8) = -flowrate_out_reverse;
    VEV = -flowrate_out_reverse/(rho*flow_area_out_reverse);
    yprime(5) = yprime(5) + (0.5*flowrate_out_reverse*VEV*VEV);
end

% TURBULENCE PARAMETERS:
macro_l = V/((pi*b^2)/4);
if macro_l >= (b/2)
    macro_l = b/2;
end
turb_P = 0.3307*cbeta*(y(5)/macro_l)*sqrt(abs(y(6)/mass));
turb_eps = ((2/3)^1.5)*((y(6)/mass)^1.5)*(1/macro_l);
yprime(5) = yprime(5) - turb_P;
yprime(6) = yprime(6) + turb_P - (mass*turb_eps);

% HEAT TRANSFER CALCULATIONS:
% Piston speed:
V_piston_norm=abs((pi/2)*sin(theta)*(1+(cos(theta)/sqrt((1/eps)^2 - sin(theta)^2))));
V_piston_mean=2*stroke*RPM;
V_piston=V_piston_norm*V_piston_mean;
uprime=sqrt((2/3)*(y(6)/mass));
V_mean_ke=sqrt(2*(y(5)/mass));
% Characteristic velocity for calculation of heat transfer coefficient:
V_char=sqrt(V_mean_ke^2 + uprime^2 + (V_piston/2)^2);
% Heat Transfer Coefficient:
htran_coeff = htran_con*(abs(V_char*macro_l/kivisc))^htran_exp *(thcon/macro_l);
% Calculate instantaneous wall, piston, head surface areas:
crankl=stroke/2;
instant_pin_to_pin=(crankl*cos(theta))+sqrt(conrodl^2-(crankl*sin(theta))^2);
instant_height=conrodl+crankl-instant_pin_to_pin;
A_cyl_wall=pi*b*instant_height;
A_piston=Area_piston;
A_head=Area_head;
% Heat transfer:
\[ Q_{\text{cyl \_wall}} = h\text{tran \_coeff} \times A_{\text{cyl \_wall}} \times (T_u - T_w) \]
\[ Q_{\text{piston}} = h\text{tran \_coeff} \times A_{\text{piston}} \times (T_u - T_{piston}) \]
\[ Q_{\text{head}} = h\text{tran \_coeff} \times A_{\text{head}} \times (T_u - T_{head}) \]

% Total heat transfer:
\[ y_{prime}(4) = Q_{\text{cyl \_wall}} + Q_{\text{piston}} + Q_{\text{head}} \]

% EVALUATION OF THERMODYNAMIC PROPERTIES OF FRESH CHARGE AND RGF:
% Properties of fresh charge (Subscript 1):
[\[h_1, u_1, v_1, s_1, Y_1, c_{p1}, d_{l1T1}, d_{l1p1}, rh_1, DRHODT_1, DRHODP_1, ct_1, ADUMY_1, BDUMY_1, \gamma_1, R_1\] = farg(p,T_u,phi,0,fueltype,airscheme);
% Properties of residual gas (Subscript 2):
[\[h_2, u_2, v_2, s_2, Y_2, c_{p2}, d_{l2T2}, d_{l2p2}, rh_2, DRHODT_2, DRHODP_2, ct_2, ADUMY_2, BDUMY_2, \gamma_2, R_2\] = farg(p,T_u,phi,1,fueltype,airscheme);

GDUMY = ((R_1 - R_2)/R);
HDUMY = GDUMY + (((h_2 - h_1)/BDUMY)) / mass; % rate of change of fresh mass fraction

% EVALUATION OF REMAINING DERIVATIVES:
y_{prime}(2) = (BDUMY/ADUMY) \times (y_{prime}(9) \times HDUMY +
\quad ((mdot/mass) \times (1 - (h/BDUMY)) - (V_{dot}/V) +
\quad ((1/(BDUMY\times mass)) \times ((y_{prime}(7) \times him) - (y_{prime}(8) \times h) - y_{prime}(4)))));
y_{prime}(1) = (\rho/DRHODP) \times ((GDUMY\times y_{prime}(9)) - (V_{dot}/V) -
\quad (y_{prime}(2) \times (1/\rho) \times DRHODT) + (mdot/mass));
y_{prime}(3) = p \times V_{dot};

% Convert time derivative to crank angle derivatives
for i=1:9
\quad y_{prime}(i) = y_{prime}(i) \times dt\_dtheta;
end
function yprime=Compression(theta,y,flag);
% yprime=Compression(theta,y,flag)
% Function that returns the driveivatives of the following 5 variables
% w.r.t. crank angle (theta) for the compression phase:
% 1) pressure (pa);
% 2) unburned temperature (K);
% 3) work (J);
% 4) heat transfer (J);
% 5) mean kinetic energy (J);
% 6) turbulent kinetic energy (J);
% 7) Mass induced (kg);
% 8) Mass exhausted (kg);
% 9) Mass fraction of fresh charge (-);

global b stroke eps r Cblowby f fueltype airscheme phi RPM conrodl... thetas thetab omega heattransferlaw hcu htran_con htran_exp... TwTpiston Thetahthead Vtdc massl Piman Peman Tfresh TI Texh... cbeta EVO EVC D_head_inlet D_head_exhaust D_stem_inlet... D_stem_exhaust dtdtheta EGR him gammaim Rim Area_head Area_piston...

p=y(1);
Tu=y(2);
yprime=zeros(9,1);
% THERMODYNAMIC PROPERTY EVALUATION:
[h,u,v,s,Y,cp,dlvlT,dlvlp,rho,DRHODT,DRHODP,ct,ADUMY,BDUMY, gamma,R] = ...
    farg(p,Tu,phi,f,fueltype,airscheme);  
% EVALUATE HEAT TRANSFER PROPERTIES OF CYLINDER CHARGE:
[dvvisc,thcon]=heat_unburn(Tu);
kivisc = dvvisc/rho;
% mass in cylinder accounting for blowby:
mass = massl + y(7) - y(8);
% volume of cylinder (m^3):
V=Vtdc*(1+(r-1)/2*(1-cos(theta)+ ...
    1/eps*(1-(1-eps^2*2*sin(theta).^2).^0.5)));
% derivate of volume (m^3/rad):
dVdtheta=Vtdc*(r-1)/2*(sin(theta)+ ... 
eps/2*sin(2*theta)/sqrt(1-eps^2*sin(theta).^2));  % m^3/rad

Vdot = dVdtheta/dtdtheta;  m^3/sec

%---------------------------------------------------------------------%
% Turbulent and Mean Kinetic Energy Terms:
%--------------------------------------------------------------

% Macroscopic length scale (m):
macro_l = V/((pi*b^2)/4) ;
if macro_l >= (b/2)
    macro_l = b/2;
end

% Turbulent energy production term:
turb_P = 0.3307*cbeta*(y(5)/macro_l)*sqrt(y(6)/mass);
turb_eps = ((2/3)^1.5)*((y(6)/mass)^1.5)*(1/macro_l);
yprime(5) = -turb_P;
yprime(6) = turb_P - (mass*turb_eps);

% HEAT TRANSFER CALCULATIONS:
% Piston speed:
V_piston_norm=abs((pi/2)*sin(theta)*(1+(cos(theta)/sqrt((1/eps)^2 -
                          sin(theta)^2))));
V_piston_mean=2*stroke*RPM;
V_piston=V_piston_norm*V_piston_mean;
uprime=sqrt((2/3)*(y(6)/mass));
V_mean_ke=sqrt(2*(y(5)/mass));
% Characteristic velocity for calculation of heat transfer coefficient:
V_char=sqrt(V_mean_ke^2 + uprime^2 + (V_piston/2)^2);
% Heat Transfer Coefficient:
htran_coeff = htran_con*(abs(V_char*macro_l/kivisc))^htran_exp *
                       (thcon/macro_l);
% Calculate instantaneous wall, piston, head surface areas:
crankl=stroke/2;
instant_pin_to_pin=(crankl*cos(theta)) + sqrt(conrodl^2-
                      (crankl*sin(theta))<sup>2</sup>);
instant_height=conrodl+crankl-instant_pin_to_pin;
A_cyl_wall=pi*b*instant_height;
A_piston=Area_piston;
A_head=Area_head;
% Heat transfer:
Q_cyl_wall=htran_coeff*A_cyl_wall*(Tu-Tw);
Q_piston=htran_coeff*A_piston*(Tu-Tpiston);
Q_head=htran_coeff*A_head*(Tu-Thead);
% Total heat transfer:
yprime(4)=Q_cyl_wall+Q_piston+Q_head;

[h,u,v,s,Y,cp,dlvlT,dlvlp,rho,DRHODT,DRHODP,ct,ADUMY,BDUMY,gamma,R]=
... farg(p,Tu,phi,f,fueltype,airscheme);

yprime(2)= -(BDUMY/ADUMY)*( (Vdot/V) + (yprime(4)/(BDUMY*mass)) );
yprime(1)=(rho/DRHODP)*(-Vdot/V) - (yprime(2)*DRHODT/rho) );
yprime(3)=p*Vdot;

for i = 1:9
yprime(i) = yprime(i)*dtdtheta;
end
Combustion.m

function yprime=Combustion(theta,y,flag);

% yprime=Combustion(theta,y,flag)
%
% Function that returns the derivatives of the following 13 variables
% w.r.t. crank angle for the combustion phase:
% 1) pressure;
% 2) unburned temperature;
% 3) burned temperature;
% 4) work;
% 5) heat transfer;
% 6) mean kinetic energy;
% 7) turbulent kinetic energy
% 8) Mass induced (kg);
% 9) Mass exhausted (kg);
% 10) Mass fraction of fresh charge (-);
% 11) Mass fraction entrained (-);
% 12) Mass fraction burned (-);
% 13) Volume of unburned mixture during combustion (m^3);

global b stroke eps r Cblowby f fueltype airscheme phi conrodl... thetas thetab RPM omega htran_con htran_exp... heattransferlaw hcu hcb Tpiston Thead Area_piston Area_head... p1 T1 V1 Tw thetal Vtdc Vbdc mass1 dttdtheta cbeta ... rhob_spark rhou_spark macro_length_spark uprime_spark cmult... counter Area_flame_true theta_true EVO Area_flame_entrain_stop... theta_entrain_stop...

p=y(1);
Tu=y(2);
Tb=y(3);
yprime=zeros(14,1);

% THERMODYNAMIC PROPERTY EVALUATION:
% Unburned zone:
[hu,uu,vu,s,Y,cpu,dlvlTu,dlvlpu,rhou,DRHODTU,DRHODPU,ctu,ADUMYU,BDUMYU, gammau,Ru]= ...
  farg(p,Tu,phi,f,fueltype,airscheme);
% Burned Zone:
if Tb>1000
  [hb,ub,vb,s,Y,cpb,dlvlTb,dlvlpb,rhob,DRHODTB,DRHODPB,ctb,ADUMYB,BDUMYB, gamma,b,Rb]= ...
  ecp(p,Tb,phi,fueltype,airscheme);
else
  [hb,ub,vb,s,Y,cpb,dlvlTb,dlvlpb,rhob,DRHODTB,DRHODPB,ctb,ADUMYB,BDUMYB, gamma,b,Rb]= ...
  farg(p,Tb,phi,1.0,fueltype,airscheme);
end

% EVALUATE HEAT TRANSFER PROPERTIES OF CYLINDER CHARGE:
% Unburned zone:
\[\text{[dyvisc\_unburn,thcon\_unburn,prandtl\_unburn]} = \text{heat\_unburn}(Tu)\;
\]
\[\text{kivisc\_unburn} = \frac{\text{dyvisc\_unburn}}{\rho u};\]

% Burned Zone:
\[\text{[dyvisc\_burn,thcon\_burn,prandtl\_burn]} = \text{heat\_burn}(Tb,\gamma_{\text{b}},c_{\text{pb}},\phi);\]
\[\text{kivisc\_burn} = \frac{\text{dyvisc\_burn}}{\rho b};\]

% MASS IN CYLINDER:
\[\text{mass} = \text{mass1} + y(8) - y(9);\]
\[\text{mass\_unburned} = \text{mass} \times (1 - y(12));\]

% CYLINDER VOLUME:
\[\text{V} = \text{Vtdc} \times \left(1 + \frac{(r - 1)}{2} \times \text{cos}(\theta) + \ldots \right)\]
\[\frac{1}{\text{eps}} \times \left(1 - \left(1 - \frac{\text{eps}^2 \times \text{sin}(\theta)}{2} \right) \times 0.5\right));\]

% ENTRAINED AND BURNED VOLUME:
\[\text{Volume\_entrained} = \text{V} - (\text{mass} / \rho u) \times (1 - y(11));\]
\[\text{Volume\_burned} = \text{V} - y(13);\]

% DERIVATIVE OF VOLUME (CAD AND TIME):
\[\frac{dV}{dt} = \frac{\text{Vtdc} \times (r - 1)}{2} \times \text{sin}(\theta) + \ldots \]
\[\frac{\text{eps}^2 \times \text{sin}^2(2 \times \theta)}{2} \times \sqrt{1 - \text{eps}^2 \times \text{sin}^2(\theta)});\]
\[\text{Vdot} = \frac{\text{dV}}{dt} \times \text{dt} / \text{dt}\;\theta;\]

% CALCULATE INSTANTANEOUS CYLINDER WALL AREA:
\[\text{crankl} = \text{stroke} / 2;\]
\[\text{instant\_pin\_to\_pin} = (\text{crankl} \times \text{cos}(\theta)) + \sqrt{\text{conrod}^2 - (\text{crankl} \times \text{sin}(\theta))}^2);\]
\[\text{instant\_height} = \text{conrod} + \text{crankl} - \text{instant\_pin\_to\_pin};\]
\[\text{A\_cyl\_wall} = \pi \times b \times \text{instant\_height};% m^2/\text{sec}\]

% FLAME AREA EVALUATION:
% \[\text{[Area\_flame,Volume\_flame]} = \text{flamegeometry}(\theta, \text{Volume\_burned}, b);\]
\[\text{[Area\_head, A\_head\_burn, A\_head\_unburn, A\_piston\_burn, A\_piston\_unburn, A\_cylinder\_burn, A\_cylinder\_unburn, Volume\_flame, Vub]} = \]
\[\text{flamegeometry}(\theta, \text{Volume\_burned}, b);\]

if \[y(\text{11}) \geq 0.99\]
\[\text{Area\_flame\_true(counter)} = \text{Area\_flame};\]
\[\theta\_true(counter) = \theta;\]
\[\text{Area\_flame\_entrain\_stop} = \text{Area\_flame\_true(1) ;}\]
\[\theta\_entrain\_stop = \theta\_true(1);\]
\[\text{counter} = \text{counter} + 1;\]
\[\text{Area\_flame} = \text{Area\_flame\_entrain\_stop(1)} \times ((EVO - \theta) / (EVO - \theta\_entrain\_stop))^2;\]
\[\text{A\_head\_burn} = \text{Area\_head};\]
\[\text{A\_piston\_burn} = \text{Area\_piston};\]
\[\text{A\_cylinder\_burn} = \text{A\_cyl\_wall};\]
\[\text{A\_head\_unburn} = 0;\]
\[\text{A\_piston\_unburn} = 0;\]
\[\text{A\_cylinder\_unburn} = 0;\]
end
% PREDICTED BURN RATE CALCULATIONS:
macro_l = macro_length.spark * (rhou_spark/rhou)^(1/3);
if macro_l >= b/2
    macro_l = b/2;
end
laminar_flame_speed = flamespeed(p,Tu,f);
uprime = cmult * uprime_spark * (rhou/rhou_spark)^(1/3);
micro_length = macro_l * sqrt(15) * (uprime*macro_l/kivisc_unburn)^(-1/2);
burn_time = micro_length/laminar_flame_speed;

yprime(11) = rhou*Area_flame*(uprime+laminar_flame_speed)/mass;
if y(11)>=0.995;
    yprime(11)=0;
end
if y(11)>=0.998;
    yprime(11)=0;
end
if y(11)>=0.999;
    y(11)=1.0;
end
if y(11)>=0.9999;
    y(11)=1.0;
end
yprime(12) = (y(11)-y(12))/burn_time;

% TURBULENCE PARAMETERS:
turbulent_ke = 1.5*mass*uprime^2;
yprime(6) = -0.3307*cbeta*(y(6)/macro_l)*sqrt(turbulent_ke/mass);

% HEAT TRANSFER CALCULATIONS:
% Piston speed:
V_piston_norm=abs((pi/2)*sin(theta)*(1+(cos(theta)/sqrt((1/eps)^2 - sin(theta)^2))));
V_piston_mean=2*stroke*RPM;
V_piston=V_piston_norm*V_piston_mean;
V_mean_ke=sqrt(2*(y(6)/mass));
% Characteristic velocity for calculation of heat transfer coefficient:
V_char=sqrt(V_mean_ke^2 + uprime^2 + (V_piston/2)^2);

% Heat Transfer Coefficient:
htran_con*\(\frac{(V_char \cdot macro_l/kivisc_unburn)^{htran\_exp} \cdot (thcon\_unburn/macrol)^0.33 \cdot (thcon\_burn/macrol)}{htran\_coef\_unburn} = htran_con*\(\frac{(V_char \cdot macro_l/kivisc\_burn)^{htran\_exp}}{htran\_coef\_burn} \cdot (thcon\_burn/macrol)^0.33 \cdot (thcon\_burn/macrol)};

% Heat transfer:
% Unburned zone:
Q_cyl_wall_unburn=htran_coef_unburn*A_cylinder_unburn*(Tu-Tw);
Q_piston_unburn=htran_coef_unburn*A_piston_unburn*(Tu-Tpiston);
Q_head_unburn=htran_coef_unburn*A_head_unburn*(Tu-Thead);
% Burned zone:
Q_cyl_wall_burn = htran_coeff_burn*A_cylinder_burn*(T_b - T_w);
Q_piston_burn = htran_coeff_burn*A_piston_burn*(T_b - T_piston);
Q_head_burn = htran_coeff_burn*A_head_burn*(T_b - T_head);

% Total heat transfer:
Q_unburn = Q_cyl_wall_unburn + Q_piston_unburn + Q_head_unburn;
Q_burn = Q_cyl_wall_burn + Q_piston_burn + Q_head_burn;
yprime(5) = Q_unburn + Q_burn;

% EVALUATION OF REMAINING DERIVATIVES:
CDUMU = 1 - (DRHODTU*BDUMYU)/(rhou*ADUMYU);
CDUMB = 1 - (DRHODTB*BDUMYB)/(rhob*ADUMYB);
DDUMY = (yprime(12)/(y(12)-1)) - ...
   ((1/(1/CDUMU))*(Q_unburn/(BDUMYU*mass_unburned)));
EDUMY = (Volume_burned/y(13))*...
   ((Vdot/Volume_burned)-(yprime(12)/y(12)) - ...
   (1/(1/CDUMB))*((yprime(12)/y(12))*)((hu-hb)/BDUMYB)) - ...
   (Q_burn/(BDUMYB*y(12)*mass)));
FDUMY = ((Volume_burned/y(13))*(DRHODPB/(CDUMB*rhob)) + ... 
   (DRHODPU/(CDUMU*rhou));

yprime(1) = (DDUMY-EDUMY)/FDUMY;
yprime(13) = y(13)*(DDUMY - (DRHODPU*yprime(1)/(CDUMU*rhou)));
DVDTB = Vdot - yprime(13);
yprime(2) = (BDUMYU/ADUMYU)*((yprime(12)/(y(12)-1)) - 
   (yprime(13)/y(13))... 
   - (Q_unburn/(BDUMYU*mass_unburned)));
yprime(3) = (BDUMYB/ADUMYB)*(((yprime(12)/y(12)))*(1+((hu-hb)/BDUMYB)) 
   ... 
   - (DVDTB/Volume_burned) - (Q_burn/(BDUMYB*y(12)*mass)));
yprime(4)=p*Vdot;

for i=1:13
    yprime(i) = yprime(i)*dt*dtheta;
end

68
function yprime=Exhaust(theta,y,flag);
%
% yprime=Exhaust(theta,y,flag)
%
% Function that returns the derivatives of the following 8 variables
% w.r.t. crank angle (theta) for the intake stroke:
% 1) pressure;
% 2) unburned temperature;
% 3) work;
% 4) heat transfer; and
% 5) mean kinetic energy;
% 6) turbulent kinetic energy
% 7) Mass induced (kg);
% 8) Mass exhausted (kg);

global b stroke eps r Cblowby f fueltype airscheme phi conrodl... 
thetas thetab omega ... 
heattransferlaw hcu htran_con htran_exp... 
Twr Tpiston Thedl thetadl Vtdc massl ... 
Pisman Peman Tfresh ... 
cbeta Area_piston Area_head... 
D_head_inlet D_head_exhaust D_stem_inlet D_stem_exhaust dtdtheta EGR 
RPM... 
data_EV_lift data_EV_flow...

p = y(1); 
Tb = y(2); 
yprime = zeros(8,1);

% Evaluate thermodynamic properties of the in-cylinder charge:
if Tb<1000
[h,u,v,s,Y,cp,dvl,T,dvlp,rho,DRHODT,DRHODP,ct,ADUMY,BDUMY,gamma,R] = ... 
    farg(p,Tb,phi,1,fueltype,airscheme);
else
[h,u,v,s,Y,cp,dvl,T,dvlp,rho,DRHODT,DRHODP,ct,ADUMY,BDUMY,gamma,R] = ... 
    ecp(p,Tb,phi,fueltype,airscheme);
end

% Burned Zone:
[dyvisc_burn,thcon]=heat_burn(Tb,gamma,cp,phi);
kivisc = dyvisc_burn/rho;

% Mass in cylinder accounting for blowby (kg):
mass = massl + y(7) - y(8);

% Instantaneous cylinder volume (m3):
V = Vtdc*(1+(r-1)/2*(1-cos(theta))+ ... 
    1/eps*(1-(1-eps^2*sin(theta).^2).^(0.5)));
% Derivative of volume w.r.t. CAD:
\[ \frac{dV}{d\theta} = \frac{V_{tdc}}{2} \left( r - 1 \right) \left( \sin(\theta) + \frac{\varepsilon}{2} \sin(2\theta) \right) \sqrt{1 - \varepsilon^2 \sin^2(\theta)}; \]

\[ V_{dot} = \frac{dV}{d\theta} \frac{1}{d\theta_{\text{theta}}}; \]
\[ \rho = \frac{1}{\nu}; \]

% EVALUATE MASS FLOW RATE THROUGH EXHAUST VALVE:
\[ \theta_{\text{deg}} = \theta \frac{180}{\pi}; \]
\[ \text{EV}_{\text{CAD}} = \text{data}_{\text{EV lift}}(:,1); \] % Reference CAD input
\[ \text{EV}_{\text{lift}} = \text{data}_{\text{EV lift}}(:,2); \] % Lift (mm)
\[ \text{lift}_{\text{poly EV}} = \text{interp1} (\text{EV}_{\text{CAD}}, \text{EV}_{\text{lift}}, 'cubic', 'pp'); \]
\[ \text{lift}_{\text{EV mm}} = \text{ppval} (\text{lift}_{\text{poly EV}}, \theta_{\text{deg}}); \] % Lift value for theta
\[ \text{lift}_{\text{EV}} = \text{lift}_{\text{EV mm}}/1000; \] % valve lift in meters
\[ \text{EV}_{\text{std lift}} = \text{data}_{\text{EV flow}}(:,2); \] % Reference lift (mm)
\[ \text{EV}_{\text{CD}} = \text{data}_{\text{EV flow}}(:,3); \] % Discharge coefficient
\[ \text{CD}_{\text{lift poly EV}} = \text{interp1} (\text{EV}_{\text{std lift}}, \text{EV}_{\text{CD}}, 'cubic', 'pp'); \]
\[ \text{Cd}_{\text{EV}} = \text{ppval} (\text{CD}_{\text{lift poly EV}}, \text{lift}_{\text{EV mm}}); \] % Discharge coeff value for lift at theta

if \( p \geq \text{Peman} \) % Flow OUT of cylinder (positive)
\[ [\text{flowrate out}, \text{flow area out}] = \ldots \]
\[ \text{flowrate}(\text{Cd}_{\text{EV}}, \text{D}_{\text{head exhaust}}, \text{D}_{\text{stem exhaust}}, \text{lift}_{\text{EV}}, p, \text{Tb}, \text{Peman}, \gamma, \text{R}); \]
\[ \text{yprime(8)} = \text{flowrate out}; \]
\[ \text{VEV} = \text{flowrate out} / (\rho \times \text{flow area out}); \]
\[ \text{yprime(5)} = \text{yprime(5)} - (\text{flowrate out} \times (\text{y(5)} / \text{mass})); \]
\[ \text{yprime(6)} = \text{yprime(6)} - (\text{flowrate out} \times (\text{y(6)} / \text{mass})); \]
else% Flow INTO the cylinder (negative)
\[ [\text{flowrate out reverse}, \text{flow area out reverse}] = \ldots \]
\[ \text{flowrate}(\text{Cd}_{\text{EV}}, \text{D}_{\text{head exhaust}}, \text{D}_{\text{stem exhaust}}, \text{lift}_{\text{EV}}, p, \text{Tb}, \text{Peman}, \gamma, \text{R}); \]
\[ \text{yprime(8)} = -\text{flowrate out reverse}; \]
\[ \text{VEV} = -\text{flowrate out reverse} / (\rho \times \text{flow area out reverse}); \]
\[ \text{yprime(5)} = \text{yprime(5)} + (0.5 \times \text{flowrate out reverse} \times \text{VEV}^2); \]
end

% TURBULENCE PARAMETERS:
\[ \text{macro l} = V / ((\pi \times b^2)/4); \]
if \( \text{macro l} \geq (b/2) \)
\[ \text{macro l} = b/2; \]
end
\[ \text{turb} P = 0.3307 \times c_{\beta} \times (\text{y(5)} / \text{macro l}) \times \sqrt{\text{abs}(\text{y(6)} / \text{mass})}; \]
\[ \text{turb eps} = ((2/3) \times 1.5) \times ((\text{y(6)} / \text{mass})^{1.5} \times (1 / \text{macro l}); \]
\[ \text{yprime(5)} = \text{yprime(5)} - \text{turb P}; \]
\[ \text{yprime(6)} = \text{yprime(6)} + \text{turb P} - (\text{mass} \times \text{turb eps}); \]
\[ \text{mdot} = -\text{yprime(8)}; \] % kg/sec

% HEAT TRANSFER CALCULATIONS:
% Piston speed:
V_piston_norm=abs((pi/2)*sin(theta)*(1+(cos(theta)/sqrt((1/eps)^2 -
sin(theta)^2))));
V_piston_mean=2*stroke*RPM;
V_piston=V_piston_norm*V_piston_mean;
uprime=sqrt((2/3)*(y(6)/mass));
V_mean_ke=sqrt(2*(y(5)/mass));
V_blowdown=4*yprime(8)/(pi*rho*b^2);
if yprime(8)<0
    V_blowdown=0;
end
% Characteristic velocity for calculation of heat transfer coefficient:
V_char=sqrt(V_mean_ke^2 + uprime^2 + (V_piston/2)^2 + V_blowdown^2);

% Heat Transfer Coefficient:
htran_coeff = htran_con*(abs(V_char*macro_1/kivisc))^htran_exp *
(thcon/macro_1);

% Calculate instantaneous wall, piston, head surface areas:
crankl=stroke/2;
instant_pin_to_pin=(crankl*cos(theta))+sqrt(conrodl^2-
(crankl*sin(theta))^2);
instant_height=conrodl+crankl-instant_pin_to_pin;
A_cyl_wall=pi*b*instant_height;
A_piston=Area_piston;
A_head=Area_head;

% Heat transfer:
Q_cyl_wall=htran_coeff*A_cyl_wall*(Tb-
Tw);
Q_piston=htran_coeff*A_piston*(Tb-Tpiston);
Q_head=htran_coeff*A_head*(Tb-Thead);

% Total heat transfer:
yprime(4)=Q_cyl_wall+Q_piston+Q_head;

yprime(2) = (BDUMY/ADUMY) * ( - (yprime(8)/mass) - (Vdot/V)...
- (yprime(4)/(BDUMY*mass)) );

yprime(1) = (rho/DRHODP)*(-Vdot/V) - (yprime(2)*(1/rho)*DRHODT)...
+ (mdot/mass) );

yprime(3) = p*Vdot;
for i=1:8
    yprime(i)=yprime(i)*dt/dtheta;
end
Airdata.m

function A=airdata(scheme);
%
% A=airdata(scheme)
%
% Routine to specify the thermodynamic properties of air and
% combustion products.
% Data taken from:
% Calculation of Complex Chemical Equilibrium Composition, Rocket
% Performance, Incident and Reflected Shocks, and Chapman-Jouguet
% Detonations," NASA SP-273. As reported in Ferguson, C. R., 1986,
% "Internal Combustion Engines", Wiley.
% Sandia Report, SAND87-8215B. As reported in
% Turns, S. R., 1996, "An Introduction to Combustion:
% % Programed in MATLAB by David R. Buttsworth
% 3. David R. Buttsworth, "Spark Ignition Internal Combustion Engine
% Modeling using MATLAB", University of Southern Queensland, ISSN
% 1446-1846, 2002
% ********************************************************************
% input:
% scheme switch:
% 'GMcB_low' - Gordon and McBride 300 < T < 1000 K
% 'GMcB_hi' - Gordon and McBride 1000 < T < 5000 K
% 'Chemkin_low' - Chemkin 300 < T < 1000 K
% 'Chemkin_hi' - Chemkin 1000 < T < 5000 K
% output:
% A - matrix of polynomial coefficients for cp/R, h/RT, and s/R
% of the form h/RT=a1+a2*T/2+a3*T^2/3+a4*T^3/4+a5*T^4/5*a6/T (for
% example) where T is expressed in K
% columns 1 to 7 are coefficients a1 to a7, and
% rows 1 to 10 are species CO2 H2O N2 O2 CO H2 H O OH and NO
% ************************************************************************
switch scheme
 case 'GMcB_low'
 A=[ 0.24007797E+01 0.87350957E-02 -0.66070878E+05 0.20021861E-08 ...
 0.63274039E-15 -0.48377527E+05 0.96951457E+01
 0.40701275E+01 -0.11084999E-02 0.41521180E-05 -0.29637404E-08 ...
 0.80702103E-12 -0.30279722E+05 -0.32270046E+00
 0.36748261E+01 -0.12081500E-02 0.23240102E-05 -0.63217559E-09 ...
 0.22577253E-12 -0.10611589E+04 0.23580424E+01
 0.36255985E+01 -0.18782184E-02 0.70554544E-05 -0.67365137E-08 ...
 0.21555993E-11 -0.10475226E+04 0.43052778E+01
 0.37100928E+01 -0.16190964E-02 0.36923594E-05 -0.20319674E-08 ...
 0.23953344E-12 -0.14356310E+05 0.29553535E+01
 0.30574451E+01 -0.26765200E-02 -0.58099162E-05 0.55210391E-08 ...
 0.18122739E-11 -0.98890474E+03 -0.22997056E+01
 0.25000000E+01 0.00000000E+00 0.00000000E+00 0.00000000E+00 ...
 0.00000000E+00 0.25471627E+05 -0.46011762E+00
end
0.29464287E+01 -0.16381665E-02 0.24210316E-05 -0.16028432E-08 ...
0.38906964E-12 0.29147644E+05 0.29639949E+01
0.38375934E+01 -0.10778858E-02 0.96830378E-06 0.18713972E-09 ...
-0.22571094E-12 0.3641823E+04 0.49370009E+00
0.40459521E+01 -0.34181738E-02 0.79819190E-05 -0.61139316E-08 ...
0.15919076E-11 0.97453934E+04 0.29974988E+01];
case 'GMcB_hi'
A=[ 0.44608041E+01 0.30981719E-02 -0.12392571E-05 0.22741325E-09 ...
-0.15525954E-13 -0.48961442E+05 -0.98635982E+00
0.27167633E+01 0.29451374E-02 -0.80224374E-06 0.10226682E-09 ...
-0.48472145E-14 -0.29905826E+05 0.66305671E+01
0.28963194E+01 0.15154866E-02 -0.57235277E-06 0.99807393E-10 ...
-0.65223555E-14 -0.90586184E+03 0.61615148E+01
0.36219535E+01 0.73618264E-03 -0.19652228E-06 0.36201558E-10 ...
-0.28945627E-14 -0.12019825E+04 0.36150960E+01
0.29840696E+01 0.14891390E-02 -0.57899684E-06 0.10364577E-09 ...
-0.69353550E-14 -0.14245228E+05 0.63479156E+01
0.31001901E+01 0.51119464E-03 0.52644210E-07 -0.34909973E-10 ...
0.36945345E-14 -0.87738042E+03 -0.19629421E+01
0.25000000E+01 0.00000000E+00 0.00000000E+00 0.00000000E+00 ...
0.00000000E+00 0.25471627E+05 -0.46011763E+00
0.25420596E+01 -0.27550619E-04 -0.31028033E-08 0.45510674E-11 ...
-0.43680515E-15 -0.29230830E+05 0.49203080E+01
0.29106427E+01 0.95931650E-03 -0.19441702E-06 0.13756646E-10 ...
0.14224542E-15 0.39335181E+04 0.54423445E+01
0.31890000E+01 0.13382818E-02 -0.52899318E-06 0.95919332E-10 ...
-0.64847932E-14 0.98283290E+04 0.67458126E+01];
case 'Chemkin_low'
A=[ 0.02275724E+02 0.09922072E-01 -0.10409113E-04 0.06866686E-07 ...
-0.02117280E-10 -0.04837314E+06 0.10188488E+02
0.03386842E+02 0.03474982E-01 -0.06354696E-04 0.06968581E-07 ...
-0.02506588E-10 -0.03020811E+06 0.02590232E+02
0.03298677E+02 0.14082404E-02 -0.03963222E-04 0.05641515E-07 ...
-0.00244485E-10 -0.10208999E+04 0.03950372E+02
0.03212936E+02 0.11274864E-02 -0.05756150E-05 0.13138773E-08 ...
-0.08768554E-11 -0.10052490E+04 0.06034737E+02
0.03262451E+02 0.15119409E-02 -0.03881755E-04 0.05581944E-07 ...
-0.02474951E-10 -0.14310539E+05 0.04848897E+02
0.03298124E+02 0.08249441E-02 -0.08143015E-05 -0.09475434E-09 ...
0.04134872E-11 -0.10125209E+04 -0.03294094E+02
0.02500000E+02 0.00000000E+00 0.00000000E+00 0.00000000E+00 ...
0.00000000E+00 0.02547162E+06 -0.04601176E+01
0.02946428E-02 -0.16381665E-02 0.02421031E-04 -0.16028431E-08 ...
0.03890696E-11 0.02914764E+06 0.02963995E+02
0.03637266E+02 0.01850910E-02 -0.16761646E-05 0.02387202E-07 ...
-0.08431442E-11 0.03606781E+05 0.13588605E+01
0.03376541E+02 0.12530634E-02 -0.03302750E-04 0.05217810E-07 ...
-0.02446262E-10 0.09817961E+05 0.05829590E+02];
case 'Chemkin_hi'
A=[ 0.04453623E+02 0.03140168E-01 -0.12784105E-05 0.02393996E-08 ...
-0.16690333E-13 -0.04896696E+06 -0.09553959E+01
0.02672145E+02 0.03056293E-01 -0.08730260E-05 0.12009964E-09 ...
-0.06391618E+13 -0.02989921E+06 0.06862817E+02
0.02926640E+02 0.14879768E-02 -0.05684760E-05 0.10097038E-09 ...
73
-0.06753351E-13 -0.09227977E+04 0.05980528E+02
0.03697578E+02 0.06135197E-02 -0.12588420E-06 0.01775281E-09 ...
-0.11364354E-14 -0.12339301E+04 0.03189165E+02
0.03025078E+02 0.14426885E-02 -0.05630827E-05 0.10185813E-09 ...
-0.06910951E-13 -0.14268350E+05 0.06108217E+02
0.02991423E+02 0.07000644E-02 -0.05633828E-06 -0.09231578E-10 ...
0.15827519E-14 -0.08350340E+04 -0.13551101E+01
0.02500000E+02 0.00000000E+00 0.00000000E+00 0.00000000E+00 ...
0.00000000E+00 0.02547162E+06 -0.04601176E+01
0.02542059E+02 -0.02755061E-03 -0.03102803E-07 0.04551067E-10 ...
-0.04368051E-14 0.02923080E+06 0.04920308E+02
0.02882730E+02 0.10139743E-02 -0.02276877E-05 0.02174683E-09 ...
-0.05126305E-14 0.03886888E+05 0.05595712E+02
0.03245435E+02 0.12691383E-02 -0.05015890E-05 0.09169283E-09 ...
-0.06275419E-13 0.09800840E+05 0.06417293E+02];
end
function [alpha,beta,gamma,delta,Afuel]=fueldata(fuel);

% Routine to specify the thermodynamic properties of a fuel.
% Data taken from:
% McGraw-Hill; and
% Programed in MATLAB by David R. Buttsworth

% input:
% fuel switch
% from Ferguson: 'gasoline', 'diesel', 'methane', 'methanol',
% 'nitromethane', 'benzene';
% from Heywood: 'methane_h', 'propane', 'hexane', 'isoctane_h',
% 'methanol_h', 'ethanol', 'gasoline_h1', gasoline_h2', 'diesel_h';
% from Raine: 'toluene', 'isoctane'.
% output:
% alpha, beta, gamma, delta - number of C, H, O, and N atoms
% Afuel - vector of polynomial coefficients for cp/R, h/RT, and s/R
% of the form h/RT=a1+a2*T/2+a3*T^2/3+a4*T^3/4-a5/T^2+a6/T (for
% example) where T is expressed in K.

% Set values for conversion of Heywood data to nondimensional format
% with T expressed in K
SVal=4.184e3/8.31434;
SVec=SVal*[1e-3 1e-6 1e-9 1e-12 1e3 1 1];

switch fuel
  case 'gasoline' % Ferguson
    alpha=7; beta=17; gamma=0; delta=0;
    Afuel=[4.0652 6.0977E-02 -1.8801E-05 0 0 -3.5880E+04 15.45];
  case 'diesel' % Ferguson
    alpha=14.4; beta=24.9; gamma=0; delta=0;
    Afuel=[7.9710 1.1954E-01 -3.6858E-05 0 0 -2.5254E+04 1.50884E+01];
  case 'methane' % Ferguson
    alpha=1; beta=4; gamma=0; delta=0;
    Afuel=[1.9713 2.07101E-02 -9.930422E+03 8.873728];
  case 'methanol' % Ferguson
    alpha=1; beta=4; gamma=1; delta=0;
    Afuel=[1.779819 1.262503E-02 -8.142134E-06 0 0 -1.026351E+04 1.917126E+01];
  case 'nitromethane' % Ferguson
    alpha=1; beta=3; gamma=2; delta=1;
    Afuel=[1.412633 2.087101E-02 -8.142134E-06 0 0 -1.026351E+04 1.917126E+01];
end
case 'benzene' % Ferguson
alpha=6; beta=6; gamma=0; delta=0;
Afuel=[-2.545087 4.79554E-02 -2.030765E-05 0 0 8.782234E+03 3.348825E+01];

case 'toluene' % Raine
alpha=7; beta=8; gamma=0; delta=0;
Afuel=[-2.09053 5.654331e-2 -2.350992e-5 0 0 4331.441411 34.55418257];

case 'isooctane' % Raine
alpha=8; beta=18; gamma=0; delta=0;
Afuel=[6.678E-1 8.398E-2 -3.334E-5 0 0 -3.058E+4 2.351E+1];

case 'methane_h' % Heywood
alpha=1; beta=4; gamma=0; delta=0;
Afuel=[-0.29149 26.327 -10.610 1.5656 0.16573 18.331 19.9887/SVal].*SVec;

case 'propane' % Heywood
alpha=8; beta=10; gamma=0; delta=0;
Afuel=[-1.4867 74.339 -39.065 8.0543 0.01219 -27.313 26.4796/SVal].*SVec;

case 'hexane' % Heywood
alpha=6; beta=14; gamma=0; delta=0;
Afuel=[-20.777 210.48 -164.125 52.832 0.56635 -39.836 79.5542/SVal].*SVec;

case 'isooctane_h' % Heywood
alpha=8; beta=18; gamma=0; delta=0;
Afuel=[-0.6788-1 8.398E-2 -3.334E-5 0 0 -3.058E+4 2.351E+1];

case 'methane_h' % Heywood
alpha=1; beta=4; gamma=0; delta=0;
Afuel=[-0.29149 26.327 -10.610 1.5656 0.16573 18.331 19.9887/SVal].*SVec;

case 'ethanol' % Heywood
alpha=2; beta=6; gamma=1; delta=0;
Afuel=[6.990 39.741 -11.926 0 0 -60.214 8.01623/SVal].*SVec;

case 'gasoline_h1' % Heywood
alpha=8.26; beta=15.5; gamma=0; delta=0;
Afuel=[-24.078 256.63 -201.68 64.750 0.5808 -27.562 NaN].*SVec;

case 'gasoline_h2' % Heywood
alpha=7.76; beta=13.1; gamma=0; delta=0;
Afuel=[-22.501 227.99 -177.26 56.048 0.4845 -17.578 NaN].*SVec;

case 'diesel_h' % Heywood
alpha=10.8; beta=18.7; gamma=0; delta=0;
Afuel=[-9.1063 246.97 -143.74 32.329 0.0518 -50.128 NaN].*SVec;
end
function [h,u,v,s,Y,cp,dlv1T,dlv1p,rho,DRHODT,DRHODP,ct,ADUMY,BDUMY,gamma,R]=farg(p,T,phi,f,fueltype,airscheme);

% Routine to determine the state of mixtures of fuel, air
% and residual combustion products at low temperatures.
% Method closely follows that of:
% who uses the results of:
% 2. Hires, S.D., Ekchian, A., Heywood, J.B., Tabaczynski, R.J., and
% Wall, J.C., 1976, "Performance and NOx Emissions Modeling of a Jet
% Ignition Pre-Chamber Stratified Charge Engine", SAE Trans., Vol 85,
% Paper 760161.
% Programed in MATLAB by David R. Buttsworth
% 3. David R. Buttsworth, "Spark Ignition Internal Combustion Engine
% Modeling using MATLAB", University of Southern Queensland, ISSN
% 1446-1846, 2002
% ********************************************************************
% input:
% p, T, phi - pressure (Pa), temperature (K), and equivalence ratio
% f - residual mass fraction; set f=0 if no combustion products
% are present and f=1 if only combustion products are present
% fueltype - 'gasoline', 'diesel', etc - see fueldata.m for full list
% airscheme - 'GMcB' (Gordon and McBride) or 'Chemkin'
% output:
% h - enthalpy (J/kg), u - internal energy (J/kg),
% v - specific volume (m^3/kg), s - entropy (J/kgK),
% Y - mole fractions of 6 species: CO2, H2O, N2, O2, CO, and H2,
% cp - specific heat (J/kgK),
% dlv1T - partial derivative of log(v) wrt log(T)
% dlv1p - partial derivative of log(v) wrt log(p)
% Additional variables added by Akash Desai:
% rho - density (kg/m^3)
% DRHODT - partial derivative of rho wrt temperature
% DRHODP - partial derivative of rho wrt pressure
% ct - Specific heat at constant temperature
% ADUMY, BDUMY - dummy variables
% gamma - ratio of specific heats
% R - gas constant
% Reference : Cherian Olikara & Gary Borman, "A Computer Program for
% Calculating Properties of Equilibrium Combustion Products with Some
% Applications to I.C. Engines", SAE Paper No. 750468, 1975
% ********************************************************************

[alpha,beta,gamma,delta,Afuel]=fueldata(fueltype);
switch airscheme
    case 'GMcB'
        A=airdata('GMcB_low');
    case 'Chemkin'
        A=airdata('Chemkin_low');
Ru=8314.34; % J/kmolK
table=[[-1 1 0 0 1 -1]'];
M=[44.01 18.02 28.008 32.000 28.01 2.018]'; % kg/kmol
MinMol=le-25;
dvlT=1; dvlp=-1;
eps=0.210/(alpha+0.25*beta-0.5*gamma);
if phi <= 1.0 % stoichiometric or lean
nu=[alpha*phi*eps beta*phi*eps/2 0.79+delta*phi*eps/2 ... 
0.21*(1-phi) 0 0]';
dcdT=0;
else % rich
z=1000/T;
K=exp(2.743+z*(-1.761+z*(-1.611+z*0.2803)));
dKdT=K*(-1.761+z*(-3.222+z*0.8409))/1000;
a=1-K;
b=0.42*phi*eps*(2*alpha-gamma)+K*(0.42*(phi-1)+alpha*phi*eps);
c=-0.42*alpha*phi*eps*(phi-1)*K;
num={-b+sqrt(b^2-4*a*c)}/2/a;
dcdT=dKdT*(num^2-num*(0.42*(phi-1)+alpha*phi*eps)+ ... 
0.42*alpha*phi*eps*(phi-1))/(2*num*a+b);
nu=[alpha*phi*eps-num 0.42-phi*eps*(2*alpha-gamma)+num ... 
0.79+delta*phi*eps/2 0 num 0.42*(phi-1)-num]';
end

% mole fractions and molecular weight of residual
tmoles=sum(nu);
Y=nu/tmoles;

% mole fractions and molecular weight of fuel-air
fuel=eps*phi/(1+eps*phi);
o2=0.21/(1+eps*phi);
n2=0.79/(1+eps*phi);
Mfa=fuel*(12.01*alpha+1.008*beta+16.000*gamma+14.01*delta)+ ... 
32*o2+28.02*n2;
% mole fractions of fuel-air-residual gas
Yres=f/(f+Mres/Mfa*(1-f));
Y=Y*Yres;
Yfuel=fuel*(1-Yres);
Y(3)=Y(3)+n2*(1-Yres);
Y(4)=Y(4)+o2*(1-Yres);
% component properties
Tcp0=[1 T T^2 T^3 T^4]';
Th0=[1 T/2 T^2/3 T^3/4 T^4/5 1/T]';
Ts0=[log(T) T T^2/2 T^3/3 T^4/4 1]';
cp0=A(1:6,1:5)*Tcp0;
h0=A(1:6,1:6)*Th0;
s0=A(1:6,1:5)*Ts0;
Mfuel=12.01*alpha+1.008*beta+16.000*gamma+14.01*delta;
a0=Afuel(1); b0=Afuel(2); c0=Afuel(3); d0=Afuel(6); e0=Afuel(7);
cpfuel=Afuel(5)*[1 T T^2 T^3 1/T^2]' ;
hfuel=Afuel(1:6)*[1 T/2 T^2/3 T^3/4 -1/T^2 1/T]';
s0fuel=Afuel([1:5 7])*[log(T) T T^2/2 T^3/3 -1/T^2 2 1]';
% set min value of composition so log calculations work
if Yfuel<MinMol
Yfuel=MinMol;
end
i=find(Y<MinMol);
Y(i)=ones(length(i),1)*MinMol;
% properties of mixture
h=hfuel*Yfuel+sum(h0.*Y);
s=(s0fuel-log(Yfuel/101.325e3))*Yfuel+sum((s0-log(Y)).*Y);
cp=cpfuel*Yfuel+sum(cp0.*Y)+sum(h0.*table*T*dcdT*Yres/tmoles);
MW=Mfuel*Yfuel+sum(Y.*M);
R=Ru/MW;
h=R*T*h;
u=h-R*T;
v=R*T/p;
s=R*(-log(p/101.325e3)+s);
cp=R*cp;
cr=0;
rho=1/v;

DRHODT=-rho/T;% kg/m^3-K
DRHODP=rho/p;% kg/m^3-Pa
ADUMY = cp + ((DRHODT/DRHODP)*((1/rho) - ct)); % J/kg-K
BDUMY = (1 - rho*ct)/DRHODP; % J/kg
gamma = cp/(cp-R);
function [h,u,v,s,Y,cp,dlvlT,dlvlp,rho,DRHODT,DRHODP,ct,ADUMY,BDUMY,gamma,R]=ecp(p,T,phi,fueltype,airscheme,Yguess);
%%
% Routine to determine the equilibrium state of combustion products.
% Method closely follows that of:
% which uses the method described by:
% Calculating Properties of Equilibrium Combustion Products with
% Some Applications to I.C. Engines", SAE Paper 750468.
%%
% Programed in MATLAB by David R. Buttsworth
% 3. David R. Buttsworth, "Spark Ignition Internal Combustion Engine
% Modeling using MATLAB", University of Southern Queensland, ISSN
% 1446-1846, 2002
% ********************************************************************
% input:
% p,T,phi - pressure (Pa), temperature (K), and equivalence ratio
% fueltype - 'gasoline', 'diesel', etc - see fueldata.m for full list
% airscheme - 'GMcB' (Gordon and McBride) or 'Chemkin'
% Yguess - (optional) initial estimate for mole fractions of the
% species CO2 H2O N2 O2 CO H2 H O OH and NO
% output:
% h - enthalpy (J/kg), u - internal energy (J/kg),
% v - specific volume (m^3/kg), s - entropy (J/kgK),
% Y - mole fractions of 10 species, cp - specific heat (J/kgK),
% dlvlT - partial derivative of log(v) wrt log(T)
% dlvlp - partial derivative of log(v) wrt log(p)
%%
% Additional variables added by Akash Desai:
% rho - density (kg/m^3)
% DRHODT - partial derivative of rho wrt temperature
% DRHODP - partial derivative of rho wrt pressure
% ct - Specific heat at constant temperature
% ADUMY, BDUMY - dummy variables
% gamma - ratio of specific heats
% R - gas constant
% Reference : Cherian Olikara & Gary Borman, "A Computer Program for
% Calculating Properties of Equilibrium Combustion Products with Some
% Applications to I.C. Engines", SAE Paper No. 750468, 1975
% ********************************************************************

[alpha,beta,gamma,delta,Afuel]=fueldata(fueltype);
switch airscheme
  case 'GMcB'
    A0=airdata('GMcB_hi');
  case 'Chemkin'
    A0=airdata('Chemkin_hi');
end
% Equilibrium constant data from Olikara and Borman via Ferguson
Kp=[ 0.432168E+00 -0.112464E+05 0.267269E+01 -0.745744E-04 0.242484E-08 
0.310805E+00 -0.129540E+05 0.321779E+01 -0.738336E-04 0.344645E-08 
-0.141784E+00 -0.213308E+04 0.853461E+00 0.355015E-04 -0.30227E-08 
0.150879E-01 -0.470959E+04 0.646096E+00 0.272805E-05 -0.154444E-08 
-0.752364E+00 0.124210E+05 -0.260286E+01 0.259556E-03 -0.162687E-07 
-0.415302E-02 0.148627E+05 -0.475746E+01 0.124699E-03 -0.900227E-08];

MinMol=1e-25;
tol=3e-12;
Ru=8314.34; % J/kmol.K
M=[44.01 18.02 28.008 32.000 28.01 2.018 1.009 16 17.009 30.004]'; % kg/kmol

dcdT=zeros(4,1);
dcdp=zeros(4,1);
dfdT=zeros(4,1);
dfdp=zeros(4,1);
dYdT=zeros(10,1);
dYdp=zeros(10,1);
B=zeros(4,1);

% check if solid carbon will form
eps=0.210/(alpha+0.25*beta-0.5*gamma);
if phi>(0.210/eps/(0.5*alpha-0.5*gamma))
    error('phi too high - c(s) and other species will form');
end

if nargin==5 % no Yguess so estimate the composition using farg
    [h,u,v,s,Y,cp,dlvT,dlvP]=farg(p,T,phi,1,fueltype,airscheme);
    Y(7:10)=ones(4,1)*MinMol; % since farg only returns first 6 species
else
    Y=Yguess;
end

% evaluate constants
patm=p/101.325e3; % convert Pa to atmospheres
TKp=[log(T/1000) 1/T 1 T T^2]';
K=10.^(Kp*TKp);
c=K.*[1/sqrt(patm) 1/sqrt(patm) 1 1 sqrt(patm) sqrt(patm)]';
d=[beta/alpha (gamma+0.42/eps/phi)/alpha (delta+1.58/eps/phi)/alpha]';

if abs(phi-1)<tol
    phi=phi*(1+tol*sign(phi-1));
end

i=find(Y<MinMol);
Y(i)=ones(length(i),1)*MinMol;

DY3to6=2*tol*ones(4,1);
MaxIter=500;
MaxVal=max(abs(DY3to6));
Iter=0;
DoneSome=0;
while (Iter<MaxIter)&((MaxVal>tol)|(DoneSome<1))
    Iter=Iter+1;
    if Iter>2,
        DoneSome=1;
    end
    D76=0.5*c(1)/sqrt(Y(6));
    D84=0.5*c(2)/sqrt(Y(4));
    D94=0.5*c(3)*sqrt(Y(6)/Y(4));
    D96=0.5*c(3)*sqrt(Y(4)/Y(6));
D103 = 0.5*c(4)*sqrt(Y(4)/Y(3));
D104 = 0.5*c(4)*sqrt(Y(3)/Y(4));
D24 = 0.5*c(5)*Y(6)/sqrt(Y(4));
D26 = c(5)*sqrt(Y(4));
D14 = 0.5*c(6)*Y(5)/sqrt(Y(4));
D15 = c(6)*sqrt(Y(4));
A(1,1) = 1 + D103;
A(1,2) = D14 + D24 + 1 + D84 + D104 + D94;
A(1,3) = D15 + 1;
A(1,4) = D26 + 1 + D76 + D96;
A(2,1) = 0;
A(2,2) = 2*D24 + D94 - d(1)*D14;
A(2,3) = -d(1)*D15 - d(1);
A(2,4) = 2*D26 + 2*D76 + D96;
A(3,1) = D103;
A(3,2) = 2*D14 + D24 + 2*D84 + D94 + D104 - d(2)*D14;
A(3,3) = 2*D15 + 1 - d(2)*D15 - d(2);
A(3,4) = D26 + D96;
A(4,1) = 2 + D103;
A(4,2) = D104 - d(3)*D14;
A(4,3) = -d(3)*D15 - d(3);
A(4,4) = 0;
B(1) = -(sum(Y) - 1);
B(2) = -(2*Y(2) + 2*Y(6) + Y(7) + Y(9) - d(1)*Y(1) - d(1)*Y(5));
B(3) = -(2*Y(1) + Y(2) + 2*Y(4) + Y(8) + Y(9) + Y(10) - d(2)*Y(1) - d(2)*Y(5));
B(4) = -(2*Y(3) + Y(10) - d(3)*Y(1) - d(3)*Y(5));
invA = inv(A);
DY3to6 = invA*B;
MaxVal = max(abs(DY3to6));
Y(3:6) = Y(3:6) + DY3to6/10;
i = find(Y < MinMol);
Y(i) = ones(length(i),1)*MinMol;
Y(7) = c(1)*sqrt(Y(6));
Y(8) = c(2)*sqrt(Y(4));
Y(9) = c(3)*sqrt(Y(4)*Y(6));
Y(10) = c(4)*sqrt(Y(4)*Y(3));
Y(2) = c(5)*sqrt(Y(4))*Y(6);
Y(1) = c(6)*sqrt(Y(4))*Y(5);
end
if Iter >= MaxIter
warning('convergence failure in composition loop');
end
TdKdT = [1/T - 1/T^2 1 2*T];
dKdT = 2.302585*K.*(Kp(:,[1 2 4 5])*TdKdT);
dcdT(1) = dKdT(1)/sqrt(patm);
dcdT(2) = dKdT(2)/sqrt(patm);
dcdT(3) = dKdT(3);
dcdT(4) = dKdT(4);
dcdT(5) = dKdT(5)*sqrt(patm);
dcdT(6) = dKdT(6)*sqrt(patm);
dcdp(1) = -0.5*c(1)/p;
dcdp(2) = -0.5*c(2)/p;
dcdp(5) = 0.5*c(5)/p;
dcdp(6) = 0.5*c(6)/p;
\[x_1 = \frac{Y(1)}{c(6)};\]
\[x_2 = \frac{Y(2)}{c(5)};\]
\[x_7 = \frac{Y(7)}{c(1)};\]
\[x_8 = \frac{Y(8)}{c(2)};\]
\[x_9 = \frac{Y(9)}{c(3)};\]
\[x_{10} = \frac{Y(10)}{c(4)};\]

\[\text{dfdT}(1) = dcdT(6) \times x_1 + dcdT(5) \times x_2 + dcdT(1) \times x_7 + dcdT(2) \times x_8 + ... \]
\[dcdT(3) \times x_9 + dcdT(4) \times x_{10};\]

\[\text{dfdT}(2) = 2 \times dcdT(5) \times x_2 + dcdT(1) \times x_7 + dcdT(3) \times x_9 - d(1) \times dcdT(6) \times x_1;\]

\[\text{dfdT}(3) = 2 \times dcdT(6) \times x_1 + dcdT(5) \times x_2 + dcdT(2) \times x_8 + dcdT(3) \times x_9 + ... \]
\[dcdT(4) \times x_{10} - d(2) \times dcdT(6) \times x_1;\]

\[\text{dfdT}(4) = dcdT(4) \times x_{10} - d(3) \times dcdT(6) \times x_1;\]

\[\text{dfdp}(1) = dcdp(6) \times x_1 + dcdp(5) \times x_2 + dcdp(1) \times x_7 + dcdp(2) \times x_8;\]

\[\text{dfdp}(2) = 2 \times dcdp(5) \times x_2 + dcdp(1) \times x_7 - d(1) \times dcdp(6) \times x_1;\]

\[\text{dfdp}(3) = 2 \times dcdp(6) \times x_1 + dcdp(5) \times x_2 + dcdp(2) \times x_8 - d(2) \times dcdp(6) \times x_1;\]

\[\text{dfdp}(4) = -d(3) \times dcdp(6) \times x_1;\]

\[B = -\text{dfdT};\]

\[\text{dYdT}(3:6) = \text{invA} \times B;\]

\[\text{dYdT}(1) = \sqrt{Y(4)} \times Y(5) \times dcdT(6) + D14 \times dYdT(4) + D15 \times dYdT(5);\]
\[\text{dYdT}(2) = \sqrt{Y(4)} \times Y(6) \times dcdT(5) + D24 \times dYdT(4) + D26 \times dYdT(6);\]
\[\text{dYdT}(7) = \sqrt{Y(6)} \times dcdT(1) + D76 \times dYdT(6);\]
\[\text{dYdT}(8) = \sqrt{Y(4)} \times dcdT(2) + D84 \times dYdT(4);\]
\[\text{dYdT}(9) = \sqrt{Y(4)} \times Y(6) \times dcdT(3) + D94 \times dYdT(4) + D96 \times dYdT(6);\]
\[\text{dYdT}(10) = \sqrt{Y(4)} \times Y(3) \times dcdT(4) + D104 \times dYdT(4) + D103 \times dYdT(3);\]

\[B = -\text{dfdp};\]

\[\text{dYdp}(3:6) = \text{invA} \times B;\]

\[\text{dYdp}(1) = \sqrt{Y(4)} \times Y(5) \times dcdp(6) + D14 \times dYdp(4) + D15 \times dYdp(5);\]
\[\text{dYdp}(2) = \sqrt{Y(4)} \times Y(6) \times dcdp(5) + D24 \times dYdp(4) + D26 \times dYdp(6);\]
\[\text{dYdp}(7) = \sqrt{Y(6)} \times dcdp(1) + D76 \times dYdp(6);\]
\[\text{dYdp}(8) = \sqrt{Y(4)} \times dcdp(2) + D84 \times dYdp(4);\]
\[\text{dYdp}(9) = D94 \times dYdp(4) + D96 \times dYdp(6);\]
\[\text{dYdp}(10) = D104 \times dYdp(4) + D103 \times dYdp(3);\]

% calculate thermodynamic properties
\[\text{Tcp}_0 = [1 \ T \ T^2 \ T^3 \ T^4]';\]

\[\text{Th}_0 = [\log(T) \ T^2/2 \ T^3/3 \ T^4/4 \ 1/T]';\]

\[\text{Ts}_0 = [\log(T) \ T^2/2 \ T^3/3 \ T^4/4 \ 1]';\]

\[\text{cp} = A0(:,1:5) \times \text{Tcp}_0;\]

\[h0 = A0(:,1:6) \times \text{Th}_0;\]

\[s0 = A0(:,[1:5 7]) \times \text{Ts}_0;\]

% Y(1) and Y(2) reevaluated
\[Y(1) = (2 \times Y(3) + Y(10)) \div (d(3) - Y(5));\]
\[Y(2) = (d(1) \div (d(3) \times (2 \times Y(3) + Y(10))) - 2 \times Y(6) - Y(7) - Y(9)) \div 2;\]

\[i = \text{find}(Y < \text{MinMol});\]

% properties of mixture
\[h = \text{sum}(h0 \times Y);\]
\[s = \text{sum}((s0 - \log(Y)) \times Y);\]
\[\text{cp} = \text{sum}(Y \times \text{cp}0 + h0 \times \text{dYdT} \times T);\]
\[\text{ct} = \text{sum}(h0 \times \text{dYdp} \times T);\]
\[\text{MW} = \text{sum}(Y \times \text{M});\]
\[\text{MT} = \text{sum}(\text{dYdT} \times \text{M});\]
\[\text{Mp} = \text{sum}(\text{dYdp} \times \text{M});\]
\[\text{R} = \text{Ru} \div \text{MW};\]
\[v = \text{R} \times T \div p;\]
cp=R*(cp-h*T*MT/MW);
ct=R*(ct-h*T*Mp/MW);
dlv1T=1+max(-T*MT/MW,0);
dlvlp=-1-max(p*Mp/MW,0);
rho=1/v;
h=R*T*h;
s=R*(-log(patm)+s);
u=h-R*T;
DRHODT=(-rho/T)*dlv1T;
DRHODP=(-rho/p)*dlvlp;
ADUMY = cp + ((DRHODT/DRHODP)*((1/rho) - ct)); % J/kg-K
BDUMY = (1 - rho*ct)/DRHODP; % J/kg
gamma = cp/(cp-R);
flowrate.m

function [mflow, area] = flowrate(Cd, D_head, D_stem, lift, P0, T0, Pt, gamma, R)

% FLOWRATE Calculates mass flow rate through the intake/exhaust valves.
% [mflow,area] = flowrate(Cd, lift, P0, MW, T0, Pt, gamma, R)
% Inputs -
% Cd - Discharge coefficient of valve (CAD)
% D_head - valve head diameter (m)
% lift - valve lift (CAD) (to calculate area)
% P0 - upstream pressure (pa) (for intake valve: P0 = Piman; for
% exhaust
% valve: P0 = Pcyl)
% T0 - upstream temperature(K) (for intake valve: T0 = Timan; for
% exhaust
% valve: T0 = Tcyl)
% Pt = downstream pressure (pa) (for intake valve: Pt = Pcyl;
% for exhaust valve: Pt = Peman)
% gamma - ratio of specific heats;
% R - Specific gas constant (J/kgK);
% Output -
% mflow - mass flow rate (kg/s)
% area - Flow area (m^2)
% Method -
% Flow through the orifice is treated as one dimensional quasi steady
and
isentropic (modified by the discharge coefficient). Valve head
diameter
is used to calculate the reference flow area (valve curtain
area).(Ref: % Internal Combustion Engines; John B. Heywood; (6.13)

mflow = 0;

if (P0 <= Pt)
mflow = 0; % No flow across the valves is no pressure difference
else
area = (pi/4)*(D_head^2 - D_stem^2);
ratio = Pt/P0;
constant = (Cd*area*P0) / sqrt(R*T0);
choke = (2/(gamma+1))^((gamma)/(gamma-1));

if ratio <= choke
mflow = constant * sqrt(gamma) *
(2/(gamma+1))^((gamma+1)/(2*(gamma-1)));
else
term = (2*gamma/(gamma-1))*(1-(ratio)^(gamma-1)/gamma));

end
end

end
mflow = constant*(ratio)^(1/gamma)*sqrt(term);
end
end
flamespeed.m

function SL_m_per_s_corrected = flamespeed(p,T,Xrgf);
% Function to calculate laminar flame speed of the fuel being used
% using empirical relation for flame speed of octane.
% Input:
% p - Cylinder pressure (Pa)
% T - Unburned temperature (K)
% Xrgf - Residual gas fraction (-)
%
% Output:
% SL_m_per_s_corrected - Laminar flame speed (m/s)
%
%---------------------------------------------------------------------%

p_cyl_atm = p*9.86923267e-6; % Cylinder pressure (Pa to Atm)
T_ref = 298.0; % Reference temperature (K)
p_ref = 1.0; % Reference pressure (atm)
phi = 1;

SL_reference = 26.3 - 84.7*(phi - 1.13)^2;
% Laminar speed of iso-octane (cm/s)
alpha = 2.18 - 0.8*(phi-1);
beta = -0.16 + 0.22*(phi-1);
SL_cm_per_sec = SL_reference * ((T/T_ref)^alpha) * ((p_cyl_atm/p_ref)^beta);
% cm/s

SL_cm_per_sec_corrected = SL_cm_per_sec*(4.70617*Xrgf*Xrgf-4.06185*Xrgf+1);
% Correction for residual gas content

SL_m_per_s_corrected = SL_cm_per_sec_corrected/100;% Convert from cm/s to m/s
end
function [Af1_m2,A_head_burn_m2,A_head_unburn_m2,A_piston_burn_m2,A_piston_unburn_m2,A_cylinder_burn_m2,A_cylinder_unburn_m2,Vfl_m3,Vub_m3]=flamegeometry(theta,Volume_Burned_Gas_m3,Bore)
%
% Function to evaluate the flame geometry variables from available data
% file
% Interpolation is done both in CAD and burned volume
%
% Input:
% V_bunred_gas_m3 - Burned gas volume (m^3)
% Bore - Engine bore (m)
% theta - crank angle position (radians)
%
% Output:
% Af1_m2 - Flame frontal area (m^2)
% A_head_burn_m2 - Area of cylinder head exposed to burned zone (m^2)
% A_head_unburn_m2 - Area of cylinder head exposed to unburned zone
% (m^2)
% A_piston_burn_m2 - Area of piston exposed to burned zone (m^2)
% A_piston_unburn_m2 - Area of piston exposed to unburned zone (m^2)
% A_cylinder_burn_m2 - Area of cylinder walls exposed to burned zone
% (m^2)
% A_cylinder_unburn_m2 - Area of cylinder walls exposed to unburned
% zone (m^2)
% Vfl_m3 - Flame volume (m^3)
% Vub_m3 - Unburned zone volume (m^3)
%
% ---------------------------------------------------------------------
--%

Vfl_m3=0;
Af1_m2=0;
CAD_comb = theta*180/pi; % Radians to degrees

% Read flame geometry file:
GEO=importdata(GEO-PENT.txt',' ',' ',1);

% This is the GEO.dat file. It contains the flame geometry data.
Following
% are the columns:
% 1 - rf1/Bore
% 2 - Af1/Bore^2
% 3 - A_head_burned/Bore^2
% 4 - A_head_unburned/Bore^2
% 5 - A_piston_burned/Bore^2
% 6 - A_piston_unburned/Bore^2
% 7 - A_cylinder_burned/Bore^2
% 8 - A_cylinder_unburned/Bore^2
% 9 - V_burned/Bore^3 ------------------------INPUT
% 10 - V_unburned/Bore^3
% Normalized $V_{burned}$ to compare to geometrical data:
$V_{burned\_normalized} = \frac{Volume\_Burned\_Gas\_m^3}{(Bore\_Bore\_Bore)}$;

$V_b = V_{burned\_normalized}$;  % Input burned volume
CA = CAD\_comb;  % Shift CAD by 360 so TDC compression is zero CAD
CAD = round(abs(CA));  % Take absolute value for flame area map

Block\_Matrix = [0
  4
  8
  12
  16
  20
  24
  28
  32
  36
  40
  44
  48
  52
  56
  60
  64
  68
  72
  76
  80
  84
  88
  92
  96
  100
  104
  108
  112];  % this is the CA of each block in the flame area maps

% Loop to decide the interpolation procedure depending upon the actual
% CAD and the available CAD steps.
Block\_Matrix\_Length = length(Block\_Matrix);  % Considers all data
points for interpolating for actual CAD position
count = 1;

while count <= Block\_Matrix\_Length  % CAD must be a single value
  coming into this loop
  Reference\_CA\_1 = Block\_Matrix(count,1);  % 0,4,8,...
  Reference\_CA\_2 = Reference\_CA\_1 + 1;  % 1,5,9,...
  Reference\_CA\_3 = Reference\_CA\_1 + 2;  % 2,6,10,...
  Reference\_CA\_4 = Reference\_CA\_1 + 3;  % 3,7,11,...
if CAD==Reference_CA_1
    CA_interp=1;
    count=1000; % Exits loop when correct CAD position is found
end
if CAD==Reference_CA_2
    CA_interp=2;
    count=1000;
end
if CAD==Reference_CA_3
    CA_interp=3;
    count=1000;
end
if CAD==Reference_CA_4
    CA_interp=4;
    count=1000;
end
count=count+1;
end %end of while loop

% Loop used to find column position that matches Vb calculated (input):

h=1;
while h<=50 % This loop pulls the blocks of numbers off the table to be processed locally
    if CAD>112
        CAD=112;
        CA_interp=1;
    end
    % Data Already available for these CAD positions (0,4,8....). No interpolation required.
    if CA_interp==1
        index=(CAD/4)*50+h;
        Radius_norm(h,1)=GEO.data(index,1);
        Afl_norm(h,1)=GEO.data(index,2);
        A_head_burn_norm(h,1)=GEO.data(index,3);
        A_head_unburn_norm(h,1)=GEO.data(index,4);
        A_piston_burn_norm(h,1)=GEO.data(index,5);
        A_piston_unburn_norm(h,1)=GEO.data(index,6);
        A_cylinder_burn_norm(h,1)=GEO.data(index,7);
        A_cylinder_unburn_norm(h,1)=GEO.data(index,8);
        Vb_norm(h,1)=GEO.data(index,9);
        VuB_norm(h,1)=GEO.data(index,10);
    end
    % 1,5,9..... Weighed interpolation done using available data.
    if CA_interp==2
        CA_back=CAD-1; % 0,4,8 - Data available.
        CA_forward=CAD+3; % 4,8,12 - Next CAD step in data.
        index_back=(CA_back/4)*50+h;
        % code for interpolation
    end
index_forward=(CA_forward/4)*50+h;
Radius_norm_back(h,1)=GEO.data(index_back,1);
Radius_norm_forward(h,1)=GEO.data(index_forward,1);
Afl_norm_back(h,1)=GEO.data(index_back,2);
Afl_norm_forward(h,1)=GEO.data(index_forward,2);
A_head_burn_norm_back(h,1)=GEO.data(index_back,3);
A_head_burn_norm_forward(h,1)=GEO.data(index_forward,3);
A_head_unburn_norm_back(h,1)=GEO.data(index_back,4);
A_head_unburn_norm_forward(h,1)=GEO.data(index_forward,4);
A_piston_burn_norm_back(h,1)=GEO.data(index_back,5);
A_piston_burn_norm_forward(h,1)=GEO.data(index_forward,5);
A_piston_unburn_norm_back(h,1)=GEO.data(index_back,6);
A_piston_unburn_norm_forward(h,1)=GEO.data(index_forward,6);
A_cylinder_burn_norm_back(h,1)=GEO.data(index_back,7);
A_cylinder_burn_norm_forward(h,1)=GEO.data(index_forward,7);
A_cylinder_unburn_norm_back(h,1)=GEO.data(index_back,8);
A_cylinder_unburn_norm_forward(h,1)=GEO.data(index_forward,8);
Vb_norm_back(h,1)=GEO.data(index_back,9);
Vb_norm_forward(h,1)=GEO.data(index_forward,9);
Vub_norm_back(h,1)=GEO.data(index_back,10);
Vub_norm_forward(h,1)=GEO.data(index_forward,10);

Radius_norm(h,1)=(Radius_norm_forward(h,1)-
Radius_norm_back(h,1))*(1/4)+Radius_norm_back(h,1);
Afl_norm(h,1)=(Afl_norm_forward(h,1)-
Afl_norm_back(h,1))*(1/4)+Afl_norm_back(h,1);
A_head_burn_norm(h,1)=(A_head_burn_norm_forward(h,1)-
A_head_burn_norm_back(h,1))*(1/4)+A_head_burn_norm_back(h,1);
A_head_unburn_norm(h,1)=(A_head_unburn_norm_forward(h,1)-
A_head_unburn_norm_back(h,1))*(1/4)+A_head_unburn_norm_back(h,1);
A_piston_burn_norm(h,1)=(A_piston_burn_norm_forward(h,1)-
A_piston_burn_norm_back(h,1))*(1/4)+A_piston_burn_norm_back(h,1);
A_piston_unburn_norm(h,1)=(A_piston_unburn_norm_forward(h,1)-
A_piston_unburn_norm_back(h,1))*(1/4)+A_piston_unburn_norm_back(h,1);
A_cylinder_burn_norm(h,1)=(A_cylinder_burn_norm_forward(h,1)-
A_cylinder_burn_norm_back(h,1))*(1/4)+A_cylinder_burn_norm_back(h,1);
A_cylinder_unburn_norm(h,1)=(A_cylinder_unburn_norm_forward(h,1)-
A_cylinder_unburn_norm_back(h,1))*(1/4)+A_cylinder_unburn_norm_back(h,1);
Vb_norm(h,1)=(Vb_norm_forward(h,1)-
Vb_norm_back(h,1))*(1/4)+Vb_norm_back(h,1);
Vub_norm(h,1)=(Vub_norm_forward(h,1)-
Vub_norm_back(h,1))*(1/4)+Vub_norm_back(h,1);
end

% 2,6,10..... Weighed interpolation done using available data.
if CA_interp==3
    CA_back=CA_D+2;
    CA_forward=CA_D+2;
    index_back=(CA_back/4)*50+h;
    index_forward=(CA_forward/4)*50+h;
    Radius_norm_back(h,1)=GEO.data(index_back,1);
    Radius_norm_forward(h,1)=GEO.data(index_forward,1);
    Afl_norm_back(h,1)=GEO.data(index_back,2);
    Afl_norm_forward(h,1)=GEO.data(index_forward,2);
    A_head_burn_norm_back(h,1)=GEO.data(index_back,3);
    A_head_burn_norm_forward(h,1)=GEO.data(index_forward,3);
    A_head_unburn_norm_back(h,1)=GEO.data(index_back,4);
    A_head_unburn_norm_forward(h,1)=GEO.data(index_forward,4);
    A_piston_burn_norm_back(h,1)=GEO.data(index_back,5);
    A_piston_burn_norm_forward(h,1)=GEO.data(index_forward,5);
    A_piston_unburn_norm_back(h,1)=GEO.data(index_back,6);
    A_piston_unburn_norm_forward(h,1)=GEO.data(index_forward,6);
    A_cylinder_burn_norm_back(h,1)=GEO.data(index_back,7);
    A_cylinder_burn_norm_forward(h,1)=GEO.data(index_forward,7);
    A_cylinder_unburn_norm_back(h,1)=GEO.data(index_back,8);
    A_cylinder_unburn_norm_forward(h,1)=GEO.data(index_forward,8);
    Vb_norm_back(h,1)=GEO.data(index_back,9);
    Vb_norm_forward(h,1)=GEO.data(index_forward,9);
    Vub_norm_back(h,1)=GEO.data(index_back,10);
    Vub_norm_forward(h,1)=GEO.data(index_forward,10);

%weighted average of properties to generate properties at
the specific
%crank angle (linear interpolation)
Radius_norm(h,1)=(Radius_norm_forward(h,1)-
Radius_norm_back(h,1))*(2/4)+Radius_norm_back(h,1);
Afl_norm(h,1)=(Afl_norm_forward(h,1)-
Afl_norm_back(h,1))*(2/4)+Afl_norm_back(h,1);
A_head_burn_norm(h,1)=(A_head_burn_norm_forward(h,1)-
A_head_burn_norm_back(h,1))*(2/4)+A_head_burn_norm_back(h,1);
A_head_unburn_norm(h,1)=(A_head_unburn_norm_forward(h,1)-
A_head_unburn_norm_back(h,1))*(2/4)+A_head_unburn_norm_back(h,1);
A_piston_burn_norm(h,1)=(A_piston_burn_norm_forward(h,1)-
A_piston_burn_norm_back(h,1))*(2/4)+A_piston_burn_norm_back(h,1);
A_piston_unburn_norm(h,1)=(A_piston_unburn_norm_forward(h,1)-
A_piston_unburn_norm_back(h,1))*(2/4)+A_piston_unburn_norm_back(h,1);
A_cylinder_burn_norm(h,1)=(A_cylinder_burn_norm_forward(h,1)-
A_cylinder_burn_norm_back(h,1))*(2/4)+A_cylinder_burn_norm_back(h,1);

92
A_cylinder_unburn_norm(h,1) = (A_cylinder_unburn_norm_forward(h,1) - \
A_cylinder_unburn_norm_back(h,1)) * (2/4) + A_cylinder_unburn_norm_back(h,1); 

Vb_norm(h,1) = (Vb_norm_forward(h,1) - \
Vb_norm_back(h,1)) * (2/4) + Vb_norm_back(h,1); 

end 

% 3, 7, 11..... Weighted interpolation done using available data. 
if CA_interp==4 
CA_back=CAD-3; 
CA_forward=CAD+1; 
index_back=(CA_back/4)*50+h; 
index_forward=(CA_forward/4)*50+h; 
Radius_norm_back(h,1)=GEO.data(index_back,1); 
Radius_norm_forward(h,1)=GEO.data(index_forward,1); 
Afl_norm_back(h,1)=GEO.data(index_back,2); 
Afl_norm_forward(h,1)=GEO.data(index_forward,2); 
A_head_burn_norm_back(h,1)=GEO.data(index_back,3); 
A_head_burn_norm_forward(h,1)=GEO.data(index_forward,3); 
A_head_unburn_norm_back(h,1)=GEO.data(index_back,4); 
A_head_unburn_norm_forward(h,1)=GEO.data(index_forward,4); 
A_piston_burn_norm_back(h,1)=GEO.data(index_back,5); 
A_piston_burn_norm_forward(h,1)=GEO.data(index_forward,5); 
A_piston_unburn_norm_back(h,1)=GEO.data(index_back,6); 
A_piston_unburn_norm_forward(h,1)=GEO.data(index_forward,6); 
A_cylinder_burn_norm_back(h,1)=GEO.data(index_back,7); 
A_cylinder_burn_norm_forward(h,1)=GEO.data(index_forward,7); 
A_cylinder_unburn_norm_back(h,1)=GEO.data(index_back,8); 
A_cylinder_unburn_norm_forward(h,1)=GEO.data(index_forward,8); 
Vb_norm_back(h,1)=GEO.data(index_back,9); 
Vb_norm_forward(h,1)=GEO.data(index_forward,9); 
Vub_norm_back(h,1)=GEO.data(index_back,10); 
Vub_norm_forward(h,1)=GEO.data(index_forward,10); 

% weighted average of properties to generate properties at the specific 
% crank angle (linear interpolation) 
Radius_norm(h,1) = (Radius_norm_forward(h,1) - \
Radius_norm_back(h,1)) * (3/4) + Radius_norm_back(h,1); 
Afl_norm(h,1) = (Afl_norm_forward(h,1) - \
Afl_norm_back(h,1)) * (3/4) + Afl_norm_back(h,1); 
A_head_burn_norm(h,1) = (A_head_burn_norm_forward(h,1) - \
A_head_burn_norm_back(h,1)) * (3/4) + A_head_burn_norm_back(h,1); 
A_head_unburn_norm(h,1) = (A_head_unburn_norm_forward(h,1) - \
A_head_unburn_norm_back(h,1)) * (3/4) + A_head_unburn_norm_back(h,1);
\[ A_{\text{piston\_burn\_norm}}(h,1) = (A_{\text{piston\_burn\_norm\_forward}}(h,1) - A_{\text{piston\_burn\_norm\_back}}(h,1)) \times (3/4) + A_{\text{piston\_burn\_norm\_back}}(h,1); \]

\[ A_{\text{piston\_unburn\_norm}}(h,1) = (A_{\text{piston\_unburn\_norm\_forward}}(h,1) - A_{\text{piston\_unburn\_norm\_back}}(h,1)) \times (3/4) + A_{\text{piston\_unburn\_norm\_back}}(h,1); \]

\[ A_{\text{cylinder\_burn\_norm}}(h,1) = (A_{\text{cylinder\_burn\_norm\_forward}}(h,1) - A_{\text{cylinder\_burn\_norm\_back}}(h,1)) \times (3/4) + A_{\text{cylinder\_burn\_norm\_back}}(h,1); \]

\[ A_{\text{cylinder\_unburn\_norm}}(h,1) = (A_{\text{cylinder\_unburn\_norm\_forward}}(h,1) - A_{\text{cylinder\_unburn\_norm\_back}}(h,1)) \times (3/4) + A_{\text{cylinder\_unburn\_norm\_back}}(h,1); \]

\[ V_{\text{b\_norm}}(h,1) = (V_{\text{b\_norm\_forward}}(h,1) - V_{\text{b\_norm\_back}}(h,1)) \times (3/4) + V_{\text{b\_norm\_back}}(h,1); \]

\[ V_{\text{ub\_norm}}(h,1) = (V_{\text{ub\_norm\_forward}}(h,1) - V_{\text{ub\_norm\_back}}(h,1)) \times (3/4) + V_{\text{ub\_norm\_back}}(h,1); \]

\[ \text{end} \]

\[ h = h + 1; \]

\[ \text{end}\% \text{end of while loop to call and interpret from the geo tables} \]

\[ q = 1; \]

\[ \text{while } q \leq 50 \]

\[ \text{if } q < 50 \]

\[ V_{\text{b\_norm\_CA}} = V_{\text{b\_norm}}(q,1); \]

\[ V_{\text{b\_norm\_CA\_forward}} = V_{\text{b\_norm}}(q+1,1); \]

\[ \text{if } V_{\text{b}} = V_{\text{b\_norm\_CA}} \]

\[ \text{Radius\_norm\_burned} = \text{Radius\_norm}(q,1); \% \text{no interpolation} \]

\[ \text{needed in this case} \]

\[ q = 1000; \% \text{will exit loop if it is done} \]

\[ \% \text{fprintf}('V_{\text{b}} \text{ Equal to V}_{\text{b\_norm}} \ %d\n',y); \]

\[ \text{end} \]

\[ \text{if } V_{\text{b}} > V_{\text{b\_norm\_CA}} \&\& V_{\text{b}} < V_{\text{b\_norm\_CA\_forward}} \]

\[ X_{\text{mult}} = (V_{\text{b}} - V_{\text{b\_norm\_CA}}) / (V_{\text{b\_norm\_CA\_forward}} - V_{\text{b\_norm\_CA}}); \% \text{determines interpolation scaling factor} \]

\[ \text{Radius\_norm\_burned} = X_{\text{mult}} \times (\text{Radius\_norm}(q+1,1) - \text{Radius\_norm}(q,1)) + \text{Radius\_norm}(q,1); \% \text{interpolates Afl using the same scaling factor as V}_{\text{b}} \]

\[ q = 1000; \% \text{will exit loop if it is done} \]

\[ \% \text{fprintf}('V_{\text{b}} \text{ interpolated V}_{\text{b\_norm}} \ %d\n',y); \]

\[ \text{end} \]

\[ \% \text{end d<50 loop} \]

\[ \text{if } q = 50 \]

\[ \text{Radius\_norm\_burned} = \text{Radius\_norm}(q,1); \% \text{interpolates Afl using the same scaling factor as V}_{\text{b}} \]

\[ q = 1000; \% \text{will exit loop if it is done} \]

\[ \text{end} \]

\[ \text{end} \]

\[ \text{end} \]

\[ \text{end} \% \text{end of while loop to call and interpret from the geo tables} \]

\[ q = 1; \]

\[ \text{while } q \leq 50 \]

\[ \text{if } q < 50 \]

\[ V_{\text{b\_norm\_CA}} = V_{\text{b\_norm}}(q,1); \]

\[ V_{\text{b\_norm\_CA\_forward}} = V_{\text{b\_norm}}(q+1,1); \]

\[ \text{if } V_{\text{b}} = V_{\text{b\_norm\_CA}} \]

\[ \text{Radius\_norm\_burned} = \text{Radius\_norm}(q,1); \% \text{no interpolation needed in this case} \]

\[ q = 1000; \% \text{will exit loop if it is done} \]

\[ \% \text{fprintf}('V_{\text{b}} \text{ Equal to V}_{\text{b\_norm}} \ %d\n',y); \]

\[ \text{end} \]

\[ \text{if } V_{\text{b}} > V_{\text{b\_norm\_CA \&\& V}_{\text{b}} < V_{\text{b\_norm\_CA\_forward}} \]

\[ X_{\text{mult}} = (V_{\text{b}} - V_{\text{b\_norm\_CA}}) / (V_{\text{b\_norm\_CA\_forward}} - V_{\text{b\_norm\_CA}}); \% \text{determines interpolation scaling factor} \]

\[ \text{Radius\_norm\_burned} = X_{\text{mult}} \times (\text{Radius\_norm}(q+1,1) - \text{Radius\_norm}(q,1)) + \text{Radius\_norm}(q,1); \% \text{interpolates Afl using the same scaling factor as V}_{\text{b}} \]

\[ q = 1000; \% \text{will exit loop if it is done} \]

\[ \% \text{fprintf}('V_{\text{b}} \text{ interpolated V}_{\text{b\_norm}} \ %d\n',y); \]

\[ \text{end} \]

\[ \% \text{end d<50 loop} \]

\[ \text{if } q = 50 \]

\[ \text{Radius\_norm\_burned} = \text{Radius\_norm}(q,1); \% \text{interpolates Afl using the same scaling factor as V}_{\text{b}} \]

\[ q = 1000; \% \text{will exit loop if it is done} \]
%fprintf('GEO Column Limit Reached %d\n',y);
end
q=q+1;
end %end of block interpolation while loop

Radius_Entrained_m=(Bore*Radius_norm_burned);
Radius_Entrained_norm=(Radius_Entrained_m)/Bore;

% end

d=1;
while d<=50

if d<50
    Radius_norm_CA=Radius_norm(d,1);
    Radius_norm_CA_forward=Radius_norm(d+1,1);

    if Radius_Entrained_norm==Radius_norm_CA
        Afl_norm_final=Afl_norm(d,1); %no interpolation needed in this case
        Ve_norm_final=Vb_norm(d,1);
        A_head_burn_norm_final=A_head_burn_norm(d,1);
        A_head_unburn_norm_final=A_head_unburn_norm(d,1);
        A_piston_burn_norm_final=A_piston_burn_norm(d,1);
        A_piston_unburn_norm_final=A_piston_unburn_norm(d,1);
        A_cylinder_burn_norm_final=A_cylinder_burn_norm(d,1);
        A_cylinder_unburn_norm_final=A_cylinder_unburn_norm(d,1);
        Vub_norm_final=Vub_norm(d,1);
        d=1000; %will exit loop if it is done
        %fprintf('Vb Equal to Vb_norm %d\n',y);
    end

    if Radius_Entrained_norm>Radius_norm_CA && Radius_Entrained_norm<Radius_norm_CA_forward
        Xmult=(Radius_Entrained_norm-
               Radius_norm_CA)/(Radius_norm_CA_forward-
               Radius_norm_CA); %determines interpolation scaling factor
        Afl_norm_final=Xmult*(Afl_norm(d+1,1)-
                              Afl_norm(d,1))+Afl_norm(d,1); %interpolates Afl using the same scaling factor as Radius entrained
        A_head_burn_norm_final=Xmult*(A_head_burn_norm(d+1,1)-
                                      A_head_burn_norm(d,1))+A_head_burn_norm(d,1);
        A_head_unburn_norm_final=Xmult*(A_head_unburn_norm(d+1,1)-
                                         A_head_unburn_norm(d,1))+A_head_unburn_norm(d,1);
        A_piston_burn_norm_final=Xmult*(A_piston_burn_norm(d+1,1)-
                                         A_piston_burn_norm(d,1))+A_piston_burn_norm(d,1);
    end

endif

if Radius_Entrained_norm>Radius_norm_CA && Radius_Entrained_norm<Radius_norm_CA_forward

    Xmult=(Radius_Entrained_norm-
           Radius_norm_CA)/(Radius_norm_CA_forward-
           Radius_norm_CA); %determines interpolation scaling factor
    Afl_norm_final=Xmult*(Afl_norm(d+1,1)-
                          Afl_norm(d,1))+Afl_norm(d,1); %interpolates Afl using the same scaling factor as Radius entrained
    A_head_burn_norm_final=Xmult*(A_head_burn_norm(d+1,1)-
                                   A_head_burn_norm(d,1))+A_head_burn_norm(d,1);
    A_head_unburn_norm_final=Xmult*(A_head_unburn_norm(d+1,1)-
                                     A_head_unburn_norm(d,1))+A_head_unburn_norm(d,1);
    A_piston_burn_norm_final=Xmult*(A_piston_burn_norm(d+1,1)-
                                         A_piston_burn_norm(d,1))+A_piston_burn_norm(d,1);

endif

end
A_piston_unburn_norm_final=Xmult*(A_piston_unburn_norm(d+1,1) - A_piston_unburn_norm(d,1))+A_piston_unburn_norm(d,1);

A_cylinder_burn_norm_final=Xmult*(A_cylinder_burn_norm(d+1,1) - A_cylinder_burn_norm(d,1))+A_cylinder_burn_norm(d,1);

A_cylinder_unburn_norm_final=Xmult*(A_cylinder_unburn_norm(d+1,1) - A_cylinder_unburn_norm(d,1))+A_cylinder_unburn_norm(d,1);

Ve_norm_final=Xmult*(Vb_norm(d+1,1) - Vb_norm(d,1))+Vb_norm(d,1); %interpolates Afl using the same scaling factor as Radius entrained

Vub_norm_final=Xmult*(Vub_norm(d+1,1) - Vub_norm(d,1))+Vub_norm(d,1);

d=1000; %will exit loop if it is done
end
end

if d==50
  Afl_norm_final=Afl_norm(d,1); %interpolates Afl using the same scaling factor as Vb
  Ve_norm_final=Vb_norm(d,1);
  A_head_burn_norm_final=A_head_burn_norm(d,1);
  A_head_unburn_norm_final=A_head_unburn_norm(d,1);
  A_piston_burn_norm_final=A_piston_burn_norm(d,1);
  A_piston_unburn_norm_final=A_piston_unburn_norm(d,1);
  A_cylinder_burn_norm_final=A_cylinder_burn_norm(d,1);
  A_cylinder_unburn_norm_final=A_cylinder_unburn_norm(d,1);
  Vub_norm_final=Vub_norm(d,1);
  d=1000; %will exit loop if it is done
end

d=d+1;
end %end of block interpolation while loop

Ve_m3=(Ve_norm_final*Bore*Bore*Bore); %convert back to true burned volume
Afl_m2=(Afl_norm_final*Bore*Bore); %convert back to true flame area
A_head_burn_m2=(A_head_burn_norm_final*Bore*Bore);
A_head_unburn_m2=(A_head_unburn_norm_final*Bore*Bore);
A_piston_burn_m2=(A_piston_burn_norm_final*Bore*Bore);
A_piston_unburn_m2=(A_piston_unburn_norm_final*Bore*Bore);
A_cylinder_burn_m2=(A_cylinder_burn_norm_final*Bore*Bore);
A_cylinder_unburn_m2=(A_cylinder_unburn_norm_final*Bore*Bore);
Vub_m3=(Vub_norm_final*Bore*Bore*Bore);
end
heat_unburn.m

function
[dynamic_viscosity_unburned, thermal_conductivity_unburned, prandtl] = heat_unburn(T)
%
% Function to evaluate the dynamic viscosity and thermal conductivity of
% unburned charge in the engine cylinder:
% Input:
% Temperature (K)
%
% Output:
% 1. Dynamic Viscosity (kg/m-s or Pa-s)
% 2. Thermal Conductivity (W/m-K)
%
% Reference: S. H Mansouri and J. B. Heywood, "Correlations for the
% viscosity and prandtl number of hydrocarbon-air combustion products",
% Combustion Science and Technology, Vol. 23, pp.21-256, 1980

prandtl = 1.0;
dynamic_viscosity_unburned = 14.58e-6 * (T^1.5)/(T + 110.4) * 1e-1;
thermal_conductivity_unburned = 2.6464e2*sqrt(T)/(1+245.4*(10^(-12/T))/T)*1e-5;
end
`heat_burn.m`

function [dynamic_viscosity,thermal_conductivity,prandtl]=heat_burn(T,gamma,cp,phi)

% Function to evaluate the dynamic viscosity and thermal conductivity of
% unburned charge in the engine cylinder:
% Input:
% 1. T - Temperature (K)
% 2. gamma - ratio of specific heats
% 3. cp - Specific heat at constant pressure (J/kgK)
% 4. phi - A/F ratio
%
% Output:
% 1. Dynamic Viscosity (kg/m-s or Pa-s)
% 2. Thermal Conductivity (W/m-K)
%
% Reference: S. H Mansouri and J. B. Heywood, "Correlations for the
% viscosity and prandtl number of hydrocarbon-air combustion products",
% Combustion Science and Technology, Vol. 23, pp.21-256, 1980

dynamic_viscosity = (3.3e-7*(T^0.7)/(1+(0.027*phi))); % kg/m-s
prandtl = 0.05 + 4.2*(gamma - 1) - 6.7*(gamma - 1)^2; % unitless number

if phi>1.0 || T>1500
    prandtl=prandtl/(1+1.5e-8*phi^2*T^2);
end

thermal_conductivity = dynamic_viscosity*cp/prandtl; % W/m-K
end
function Tb=Tadiabatic(p,Tu,phi,f,fueltype,airscheme);

% Routine for calculating the adiabatic flame temperature.
% Method involves iteratively selecting flame temperatures until
% the enthalpy of the combustion products (in equilibrium) matches
% the enthalpy of the initial gas mixture.
% farg.m is used to determine the enthalpy of the unburned mixture,
% and ecp.m is used to determine the enthalpy of the burned gas.
% input:
% p - pressure (Pa)
% Tu - temperature of the unburned mixture (K)
% phi - equivalence ratio
% f - residual mass fraction; set f=0 if no combustion products
% are present and f=1 if only combustion products are present
% fueltype - 'gasoline', 'diesel', etc - see fueldata.m for full list
% airscheme - 'GMcB' (Gordon and McBride) or 'Chemkin'
% output:
% Tb - temperature of the burned gas (K) - adiabatic flame temperature
% Programed in MATLAB by David R. Buttsworth:
% David R. Buttsworth, "Spark Ignition Internal Combustion Engine
% Modelling using MATLAB", University of Southern Queensland, ISSN
% 1446-1846, 2002
% MaxIter=50;
% Tol=0.00001; % allowable error in temperature calculation
% Tb=2000; % Initial estimate
Iter=0;
DeltaT=2*Tol*Tb;

MaxIter=50;
Tol=0.00001; % allowable error in temperature calculation
Tb=2000; % Initial estimate
Iter=0;
[hu,u,v,s,Y,cp,dlvlT,dlvlp,rho,DRHODT,DRHODP,ct,ADUMY,BDUMY,gamma,R]=farg(p,Tu,phi,f,fueltype,airscheme);
while (Iter<MaxIter)&(abs(DeltaT/Tb)>Tol)
    Iter=Iter+1;
    [hb,u,v,s,Y,cp,dlvlT,dlvlp,rho,DRHODT,DRHODP,ct,ADUMY,BDUMY]=ecp(p,Tb,phi,f,fueltype,airscheme);
    DeltaT=(hu-hb)/cp;
    Tb=Tb+DeltaT;
end
if Iter>=MaxIter
    warning('convergence failure in adiabatic flame temperature loop');
end
plot_main.m

% plot_main.m
% % This script file plots the main results: Pressure, temperature and mass fraction
% % burned and entrained.
% % Additional plots can be added by making the required additions
% % to this script file.

load main.mat; % results from ahrind.mat
close all;

figure(1);
pressure_data = importdata('5000RPM_04MAP_35SPK_20CA50_LAMBA_1_PRESSURE_AVERAGE_CYLINDER_5.TXT');
plot(thetaintake_BTDC*180/pi,data_intake_BTDC(:,1)*1e-5,'b-','linewidth',2); hold on;
plot(pressure_data(:,1),pressure_data(:,2),'r--');
pspark=interp1(thetacomp,data_comp(:,1),thetas);
plot(thetaintake*180/pi,data_intake(:,1)*1e-5,'b-','linewidth',2); hold on;
plot(thetacomp*180/pi,data_comp(:,1)*1e-5,'b-','linewidth',2); hold on;
plot(thetaexhaust*180/pi,data_exhaust(:,1)*1e-5,'b-','linewidth',2);
pmax=max(pTbTuWQlHl(:,1))*1e-5;
axis([-360 360 0 pmax+10]);
legend('Prediction','Measured');
xlabel('CRANK ANGLE (degrees ATC)');
ylabel('PRESSURE (Bar)');
title(strcat('Engine Speed: ',num2str(RPM),'; Spark time (BTDC): ',num2str(abs(thetas*180/pi))));
set(gca,'FontSize',NFS)
set(gca,'FontName','Helvetica')
set(gca,'LineWidth',NLW)
set(gca,'XTick',[-180 -90 0 90 180]);
set(gca,'XMinorTick','on');
set(gca,'YMinorTick','on');
set(gca,'XTickLabel',[-180 -90 0 90 180]);

figure(2);
plot(thetaintake_BTDC*180/pi,data_intake_BTDC(:,2),'linewidth',1.5);
hold on;
plot(thetacomb*180/pi,data_comb(:,3),'r-','linewidth',1.5); hold on
plot(thetaexhaust*180/pi,data_exhaust(:,2),'r-','linewidth',1.5);
plot(thetaintake*180/pi,data_intake(:,2),'linewidth',1.5); hold on;
plot(thetacomp*180/pi,data_comp(:,2),'linewidth',1.5); hold on;
plot(thetacomb*180/pi,data_comb(:,2),'linewidth',1.5);
axis([-180 180 0 3000]);
xlabel('CRANK ANGLE (degrees ATC)');
ylabel('TEMPERATURE (K)')
title(strcat('Engine Speed: ',num2str(RPM),'; Spark time (BTDC): ' ,num2str(abs(thetas*180/pi))),'fontsize',12);
set(gca,'FontSize',NFS)
set(gca,'LineWidth',NLW)
set(gca,'XTick',[-360 -270 -180 -90 0 90 180 270 360]);
set(gca,'XTickLabel',[-360 -270 -180 -90 0 90 180 270 360]);
set(gca,'XMinorTick','on');
set(gca,'YMinorTick','on');
legend('Unburned Gas','Burned Gas');

figure(3);
plot(thetacomb*180/pi,data_comb(:,13),'linewidth',1.5); hold on;
plot(thetacomb*180/pi,data_comb(:,12),'r:','linewidth',1.5);
axis([thetas*180/pi exhaust_valve_open 0 1.01]);
MFB_10=interp1(data_comb(:,13),thetacomb,0.1)*180/pi;
MFB_50=interp1(data_comb(:,13),thetacomb,0.5)*180/pi;
MFB_90=interp1(data_comb(:,13),thetacomb,0.9)*180/pi;
Burn_Duration=MFB_90-MFB_10;
xlabel('CRANK ANGLE (degrees ATC)')
ylabel('MASS FRACTION BURNED')
title(strcat('Engine Speed: ',num2str(RPM),'; Spark time (BTDC): ' ,num2str(abs(thetas*180/pi))),'fontsize',12);
set(gca,'FontSize',NFS)
set(gca,'LineWidth',NLW)
set(gca,'YTick',[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0]);
set(gca,'YTickLabel',[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0]);
set(gca,'XTick',[thetas*180/pi:20:exhaust_valve_open]);
set(gca,'XTickLabel',[thetas*180/pi:20:exhaust_valve_open]);
set(gca,'XMinorTick','on');
set(gca,'YMinorTick','on');
legend('Mass Fraction Burned','Mass Fraction Entrained','Location','SouthEast');
disp(['Theta at 50% Mass Fraction Burn (MFB50): ' ,num2str(MFB_50)]);
mass_flow_plot.m

% mass_flow_plot.m
% Script file to plot the mass flow rates through intake and exhaust valves
% and lift and discharge coefficient profiles. mass_flow.m function
% required to calculate the mass flow rates based instantaneous pressure
% and temperatures calculated by the main program.

NLW=1; % normal line width
NFS=10; % normal font size
NMS=1; % normal marker size

theta_BTDC=thetaintake_BTDC*180/pi;
theta_intake=thetaintake*180/pi;
theta_exhaust=thetaexhaust*180/pi;

CAD_BTDC=size(thetaintake_BTDC);
for i=1:CAD_BTDC
theta=thetaintake_BTDC(i);
p=data_intake_BTDC(i,1);
T=data_intake_BTDC(i,2);
flowrate_BTDC_IV(i) = mass_flow(p,T,theta,1);
flowrate_BTDC_EV(i) = mass_flow(p,T,theta,2);
end

CAD_intake=size(thetaintake);
for i=1:CAD_intake
theta=thetaintake(i);
p=data_intake(i,1);
T=data_intake(i,2);
flowrate_intake_IV(i) = mass_flow(p,T,theta,1);
flowrate_intake_EV(i) = mass_flow(p,T,theta,2);
end

CAD_exhaust=size(thetaexhaust);
for i=1:CAD_exhaust
theta=thetaexhaust(i);
p=data_exhaust(i,1);
T=data_exhaust(i,2);
flowrate_exhaust_EV(i) = mass_flow(p,T,theta,2);
end

data_IV_lift = xlsread('Intake_Data_Final.xls');
IV_CAD = data_IV_lift(:,1);% Reference CAD input
IV_lift = data_IV_lift(:,2);% Lift (mm)
lift_poly_IV = interp1(IV_CAD,IV_lift,'cubic','pp');

data_IV_flow = importdata('Flow_coeff_intake.txt');
IV_std_lift = data_IV_flow(:,2);% Reference lift (mm)
IV_CD = data_IV_flow(:,3); % Discharge coefficient
CD_lift_poly_IV = interp1(IV_std_lift,IV_CD,'cubic','pp');

data_EV_lift = xlsread('Exhaust_Data_Final.xls');
EV_CAD = data_EV_lift(:,1); % Reference CAD input
EV_lift = data_EV_lift(:,2); % Lift (mm)
lift_poly_EV = interp1(EV_CAD,EV_lift,'cubic','pp');

data_EV_flow = importdata('Flow_coeff_exhaust.txt');
EV_std_lift = data_EV_flow(:,2); % Reference lift (mm)
EV_CD = data_EV_flow(:,3); % Discharge coefficient
CD_lift_poly_EV = interp1(EV_std_lift,EV_CD,'cubic','pp');

for i=1:1:720
    theta_plot(i)=-359 + i;
lift_IV(i)=ppval(lift_poly_IV,theta_plot(i));
    Cd_IV(i) = ppval(CD_lift_poly_IV,lift_IV(i)) ;
lift_EV(i)=ppval(lift_poly_EV,theta_plot(i));
    Cd_EV(i) = ppval(CD_lift_poly_EV,lift_EV(i)) ;
end

figure(1)
plot(theta_plot,lift_IV,'LineWidth',2); hold on;
axis([-360 360 0 13])
box on
set(gca,'FontSize',NF)
set(gca,'LineWidth',NLW)
set(gca,'XTick',[-360 -270 -180 -90 0 90 180 270 360]);
set(gca,'XTickLabel',[-360 -270 -180 -90 0 90 180 270 360]);
set(gca,'YTick',[0 2 4 6 8 10 12]);
set(gca,'YTickLabel',[0 2 4 6 8 10 12]);
set(gca,'XMinorTick','on')
set(gca,'YMinorTick','on')
ylabel('IV Lift [mm]')

figure(2)
plot(theta_plot,lift_EV,'LineWidth',2); hold on;
axis([-360 360 0 13])
box on
set(gca,'FontSize',NF)
set(gca,'LineWidth',NLW)
set(gca,'XTick',[-360 -270 -180 -90 0 90 180 270 360]);
set(gca,'XTickLabel',[-360 -270 -180 -90 0 90 180 270 360]);
set(gca,'YTick',[0 2 4 6 8 10 12]);
set(gca,'YTickLabel',[0 2 4 6 8 10 12]);
set(gca,'XMinorTick','on')
set(gca,'YMinorTick','on')
ylabel('EV Lift [mm]')

figure(3)
plot(theta_plot,Cd_IV,'LineWidth',2); hold on;
axis([-360 360 0 0.8])
box on
set(gca,'FontSize',NFS)
set(gca,'LineWidth',NLW)
set(gca,'XTick',[-360 -270 -180 -90 0 90 180 270 360]);
set(gca,'XTickLabel',[-360 -270 -180 -90 0 90 180 270 360]);
set(gca,'YTick',[0 0.2 0.4 0.6 0.8]);
set(gca,'YTickLabel',[0 0.2 0.4 0.6 0.8]);
set(gca,'XMinorTick','on')
set(gca,'YMinorTick','on')
ylabel('IV Discharge Coefficient [-]')

figure(4)
plot(theta_plot,Cd_EV,'LineWidth',2); hold on;
axis([-360 360 0 0.8])
box on
set(gca,'FontSize',NFS)
set(gca,'LineWidth',NLW)
set(gca,'XMinorTick','on')
set(gca,'YMinorTick','on')
set(gca,'XTick',[-360 -270 -180 -90 0 90 180 270 360]);
set(gca,'XTickLabel',[-360 -270 -180 -90 0 90 180 270 360]);
set(gca,'YTick',[0 0.2 0.4 0.6 0.8]);
set(gca,'YTickLabel',[0 0.2 0.4 0.6 0.8]);
ylabel('EV Discharge Coefficient [-]')
xlabel('CAD')

figure(5)
plot(theta_BTDC,flowrate_BTDC_IV,'LineWidth',2); hold on;
plot(theta_intake,flowrate_intake_IV,'LineWidth',2); hold on
axis([-360 360 -0.2 0.2])
box on
set(gca,'FontSize',NFS)
set(gca,'LineWidth',NLW)
set(gca,'XMinorTick','on')
set(gca,'YMinorTick','on')
set(gca,'XTick',[-360 -270 -180 -90 0 90 180 270 360]);
set(gca,'XTickLabel',[-360 -270 -180 -90 0 90 180 270 360]);
set(gca,'YTick',[-0.2 -0.1 0 0.1 0.2]);
set(gca,'YTickLabel',[-0.2 -0.1 0 0.1 0.2]);
set(gca,'XMinorTick','on')
set(gca,'YMinorTick','on')
ylabel('IV flow (kg/s)')

figure(6)
plot(theta_BTDC-720,flowrate_BTDC_EV,'LineWidth',2); hold on;
plot(theta_intake,flowrate_intake_EV,'LineWidth',2); hold on
plot(theta_intake,flowrate_intake_EV,'LineWidth',2); hold on
plot(theta_exhaust-720,flowrate_exhaust_EV,'LineWidth',2)
box on
axis([EVO*180/pi-720 EVC*180/pi -0.1 0.1])
set(gca,'FontSize',NFS)
set(gca,'LineWidth',NLW)
set(gca,'XMinorTick','on')
set(gca,'YMinorTick','on')
set(gca,'XTick',[-360 -270 -180 -90 0 90 180 270 360]);
set(gca,'XTickLabel',[-360 -270 -180 -90 0 90 180 270 360]);
set(gca,'YTick',[-0.1 -0.05 0 0.05 0.1]);
set(gca,'YTickLabel',[-0.1 -0.05 0 0.05 0.1]); ylabel('EV flow (kg/s)')

figure(7)
plot(theta_BTDC-720,flowrate_BTDC_IV,'LineWidth',2); hold on;
plot(theta_intake,flowrate_intake_IV,'LineWidth',2); hold on
hold on
box on
axis([IVO*180/pi-720 IVC*180/pi -0.15 0.35])
set(gca,'XMinorTick','on')
set(gca,'YMinorTick','on')
set(gca,'YTick',[-0.05:0.05:0.35]);
set(gca,'YTickLabel',[-0.05:0.05:0.35]);
xlabel('CAD')
ylabel('INTAKE MASS FLOW RATE [kg/s]')
mass_flow.m

function mass_flow_kg_sec=mass_flow(p,Tu,theta,valveflag)
% Function to calculate the mass flow rates through the intake and
% exhaust
% valve knowing the integrated pressure and temperature values.
% Input:
% 1. p - pressure(pa)
% 2. T - temperature (K)
% 3. theta - crank angle position (radians)
% 4. valveflag = 1 : Intake; 2 : Exhaust
% Output:
% mass_flow_kg_sec = mass flow rate (kg/s)

theta_deg = theta*180/pi;

% Properties of charge in intake manifold:
[him,uim,vim,sim,Yim,cpim,dlvlTim,dlvlpim,rhoim,DRHODTim,DRHODPim,ctim,
ADUMYim,BDUMYim,gammaim,Rim] = ...
    farg(Piman,Tfresh,phi,f,fueltype,airscheme);

if valveflag==1
    [h,u,v,s,Y,cp,dlvlT,dlvlp,rho,DRHODT,DRHODP,ct,ADUMY,BDUMY,gamma,R] = ...
        farg(p,Tu,phi,f,fueltype,airscheme);

    data_IV_lift = xlsread('Intake_Data_Final.xls');
    IV_CAD = data_IV_lift(:,1);
    IV_lift = data_IV_lift(:,2);
    lift_poly_IV = interp1(IV_CAD,IV_lift,'cubic','pp');
    lift_IV_mm = ppval(lift_poly_IV,theta_deg);
    lift_IV = lift_IV_mm/1000;
    data_IV_flow = importdata('Flow_coeff_intake.txt');
    IV_std_lift = data_IV_flow(:,2);
    IV_CD = data_IV_flow(:,3);
    CD_lift_poly_IV = interp1(IV_std_lift,IV_CD,'cubic','pp');
    Cd_IV = ppval(CD_lift_poly_IV,lift_IV_mm);

    if p <= Piman
        [flowrate_in,flow_area_in] = ...
            flowrate(Cd_IV, D_head_inlet, D_stem_inlet, lift_IV, Piman,
                      Tfresh, p, gammaim, Rim);
        mass_flow_kg_sec = flowrate_in;
    else
        [flowrate_in_reverse,flow_area_in_reverse] = ...
            flowrate(Cd_IV, D_head_inlet, D_stem_inlet, lift_IV, p, Tu,
                     Piman, gamma, R);
        mass_flow_kg_sec = - flowrate_in_reverse;
if valveflag==2
    if Tu<1000
        [h,u,v,s,Y,cp,dlvlT,dlvlp,rho,DRHODT,DRHODP,ct,ADUMY,BDUMY,gamma,R] = ...
        farg(p,Tu,phi,1,fueltype,airscheme);
    else
        [h,u,v,s,Y,cp,dlvlT,dlvlp,rho,DRHODT,DRHODP,ct,ADUMY,BDUMY,gamma,R] = ...
        ecp(p,Tu,phi,fueltype,airscheme);
    end
    data_EV_lift = xlsread('Exhaust_Data_Final.xls');
    EV_CAD = data_EV_lift(:,1);
    EV_lift = data_EV_lift(:,2);
    lift_poly_EV = interp1(EV_CAD,EV_lift,'cubic','pp');
    lift_EV_mm = ppval(lift_poly_EV,theta_deg);
    lift_EV = lift_EV_mm/1000  ;
    data_EV_flow = importdata('Flow_coeff_exhaust.txt');
    EV_std_lift = data_EV_flow(:,2);
    EV_CD = data_EV_flow(:,3);
    CD_lift_poly_EV = interp1(EV_std_lift,EV_CD,'cubic','pp');
    Cd_EV = ppval(CD_lift_poly_EV,lift_EV_mm) ;
    if p >= Peman
        [flowrate_out,flow_area_out] = ...
        flowrate(Cd_EV, D_head_exhaust, D_stem_exhaust, lift_EV, p, Tu,
        Peman, gamma, R);
        mass_flow_kg_sec = flowrate_out;
    else
        [flowrate_out_reverse,flow_area_out_reverse] = ...
        flowrate(Cd_EV, D_head_exhaust, D_stem_exhaust, lift_EV, Peman,
        Tu, p, gamma, R);
        mass_flow_kg_sec = - flowrate_out_reverse;
    end
end
% Script file to calculate and plot turbulent intensity (m/s) and laminar
% flame speed (m/s) during combustion:
%
mass_intake_BTDC = mass1 + data_intake_BTDC(:,7) -
data_intake_BTDC(:,8);
mass_intake = mass1 + data_intake(:,7) - data_intake(:,8);
mass_comp = mass1 + data_comp(:,7) - data_comp(:,8);
mass_exhaust = mass1 + data_exhaust(:,7) - data_exhaust(:,8);
uprime_intake_BTDC =
sqrt((2/3)*(data_intake_BTDC(:,6)./mass_intake_BTDC));
uprime_intake = sqrt((2/3)*(data_intake(:,6)./mass_intake));
uprime_comp = sqrt((2/3)*(data_comp(:,6)./mass_comp));
uprime_exhaust = sqrt((2/3)*(data_exhaust(:,6)./mass_exhaust));

for i=1:length(thetacomb)
p=data_comb(i,1);
Tu=data_comb(i,2);
[hu,uu,vu,s,Y,cpu,dlvlTu,dlvlpu,rhou,DRHODTU,DRHODPU,ctu,ADUMYU,BDUMYU,
gammau,Ru]= ...
farg(p,Tu,phi,f,fueltype,airscheme);
uprime_comb(i) = cmult * uprime_spark * ((rhou/rhou_spark)^(1/3));
laminar_flame_speed(i) = flamespeed(p,Tu,f);
end

figure(1)
plot(thetaintake_BTDC*180/pi,uprime_intake_BTDC);hold on;
plot(thetaintake*180/pi,uprime_intake);hold on;
plot(thetacomp*180/pi,uprime_comp); hold on;
plot(thetacomb*180/pi,uprime_comb); hold on;
plot(thetaexhaust*180/pi,uprime_exhaust); hold on;
axis([-360 360 0 50]);
set(gca,'XTick',[-360 -270 -180 -90 0 90 180 270 360]);
set(gca,'XTickLabel',[-360 -270 -180 -90 0 90 180 270 360]);
xlabel('CAD')
ylabel('Turbulent Intensity (m/s)')
figure(2)
flame_speed_max = max(laminar_flame_speed);
plot(thetacomb*180/pi,laminar_flame_speed)
axis([thetas*180/pi EVO*180/pi 0 flame_speed_max+0.5]);
xlabel('CAD during combustion')
ylabel('Laminar Flame Speed (m/s)')
% Script file to evaluate and plot the integral length scale and Taylor % microscale.

for i=1:length(thetacomb)
    p=data_comb(i,1);
    Tu=data_comb(i,2);
    [hu,uu,vu,s,Y,cu,pu,dlvTu,dlvpu,rhou,DRHODTU,DRHODPU,ctu,ADUMYU,BDUMYU,
     gammau,Ru]= ... 
    farg(p,Tu,phi,f,fueltype,airscheme);
    [dynamic_visc_unburn,thermal_cond,prandtl] = heat_unburn(Tu);
    kinetic_visc_unburn = dynamic_visc_unburn/rhou;
    macro_l_comb(i) = macro_length_spark * ((rhou_spark/rhou)^(1/3));
    if macro_l_comb(i) >= b/2
        macro_l(i) = b/2;
    end
    uprime(i) = cmult * uprime_spark * ((rhou/rhou_spark)^(1/3));
    micro_length(i) = macro_l_comb(i) * 
    sqrt(15*kinetic_visc_unburn/(uprime(i)*macro_l_comb(i)));
end

figure(1);
[AX,H1,H2] = 
plotyy(thetacomb*180/pi,macro_l_comb,thetacomb*180/pi,micro_length*1000 
); hold on;
xlabel('CRANK ANGLE');
set(get(AX(1),'Ylabel'),'String','INTEGRAL LENGTH SCALE (m)')
set(get(AX(2),'Ylabel'),'String','TAYLOR MICROSCALE (mm)')
set(AX(1),'YLim',[0 0.06])
set(AX(1),'YTick',[0:0.01:0.06])
set(AX(1),'XMinorTick','on')
set(AX(2),'XMinorTick','on')
set(AX(1),'YMinorTick','on')
set(AX(2),'YMinorTick','on')
set(AX(2),'YLim',[0 0.6])
set(AX(2),'YTick',[0:0.1:0.6])
set(H1,'linewidth',1.5,'linestyle','--');
set(H2,'linewidth',1.5,'linestyle','--');
legend ('Integral Length Scale', 'Taylor Microscale')
REFERENCES


Technology, vol. 23, no. 5-6, pp. 251-256, 1980.


