Thermodynamical Analysis Tool in MATLAB

Development of a thermodynamical analysis tool for the Combustion Rig in the combustion laboratories at MARINTEK, Trondheim

Anders Rohde

Marine Technology
Submission date: June 2014
Supervisor: Harald Valland, IMT

Norwegian University of Science and Technology
Department of Marine Technology
Thermodynamical Analysis Tool in MATLAB

Development of a thermodynamical analysis tool for the Combustion Rig in the combustion laboratories at MARINTEK, Trondheim

Author: Anders Rohde
Supervisor: Prof. Harald Valland

Tuesday 10th June, 2014
Analysis of combustion of liquid fuel sprays and gaseous fuel jets in a constant volume combustion chamber.

Background

Combustion rig studies are fundamentally basic and are the starting point for a combustion investigation and help to isolate the process from other engine influences like piston motion, velocity fields, turbulence, and so on. The combustion chamber is charged with a burnable gas mixture which will be ignited and burn before the injection starts to create proper conditions for injection studies.

Overall Aim and Focus

To improve experiments results from the Combustion Rig (CR) laboratory means to decrease sources of instability and measurement errors. This assignment will deal with analysis of the thermodynamic processes in the combustion chamber based on accurate pressure measurements. The analysis should include charging of the combustion chamber, pre-combustion, cooling of the hot gases, fuel injection, ignition, and combustion during the main experiment. This analysis should result in one or more computer programs that will take dynamic pressure data as input and produce output data which will help evaluation the experimental results.

The assignment should be prepared based on following points:

1. Establish thermodynamic data for unburned and burned gases. The thermodynamic analysis should be based on first principles of thermodynamics and will require accurate thermodynamic data for all relevant properties of the unburned and burned gas mixtures. The unburned mixture should be consisting of carbon monoxide, hydrogen, oxygen and nitrogen. The burned gas mixture should include carbon dioxide, water, and a certain amount of unburned gases.

2. Determine the relationship between pressure and temperature of the burned gases, based on the composition of unburned gas mixture, and its pressure and temperature. In
this analysis it may be practical to assume a certain fraction of unburned gases in the burned gas mixture.

3. **Analyze the pre-combustion phase** in order to determine the fraction of unburned gas at the end of pre-combustion. The analysis should be based on a two-zone model of the combustion chamber with unburned gas in one zone and combustion products in the other zone.

4. **Analyze heat release during fuel injection and combustion of the test fuel.**

   Compute rate of heat release of the fuel spray/fuel jet combustion and determine characteristic data such as time of ignition, duration of combustion, maximum heat release, etc.

   Point 1 and 2 should be carried out first. The sequence of point 3 and 4 should be chosen according to availability of experimental data etc.

   This assignment will be in cooperation with PhD candidate Maximilian Malin.

   The assignment text must be included as a part of the MSc report.

   The report should be written like a research report, with an abstract, conclusions, contents list, reference list, etc. During preparation of the report it is important that the candidate emphasizes easily understood and well written text. For ease of reading the report should contain adequate references at appropriate places to related text, tables and figures. On evaluation, a lot of weight is put on thorough preparation of results, their clear presentation in the form of tables and/or graphs, and on comprehensive discussion. All used sources must be completely documented. For textbooks and periodicals, author, title, year, page number and eventually figure number must be specified.

   In accordance with current regulations NTNU reserves the right to use any results from the project work in connection with teaching, research, publication, or other activities.

   The report including computer programs should be delivered in formats according to current regulations at NTNU.

---

Department of Marine Technology, 2014-01-20

Harald Valland
professor
Abstract

To meet the increasing world populations need for transportation, fossil fueled ships will are ready to offer their services. Less fuel fuel consumed per cargo unit transported, is favourable for both the operator and the enviroment. Reduced fuel consumption means less money spent on fuel and less pollution to the air. Stricter regulations regarding air pollutions from ship also requiare actions to be take to meet these.

Better understanding of the combustion processes, is an important factor to improve the diesel and gas engines operating around the world today. To obtain this understanding and knowledge, experiments needs to be performed. Knowing the proper test conditions during an experiment is vital to improve the accuracy of the results.

This thesis have coped with a way to improve the experiments performed in a fixed volume combustion rig at MARINTEK. The problem at current time is to have a proper estimate of the gas compsiton in the righ when the test fuel is injected. Creating the proper test temperature and pressure is made by the combustion of a combustable gas. The problem is to know how much of the combustable species in the gas that have actually combusted.

Estimating the compostion have been done by creating a two-zone model, where the unburned gas is in one and the completely combusted is in the other. This make a set of four differential equations, that is solved simultaniously. Input to these calculations is the measured pressure from a finished experiment. The calucalations is performed offline after an experiment is performed.

Based on the estimate of the composition after solving the two-zone model, an estimate for the rate of heat release from the gas combustion is calculated based on the mass of gas burned and the exact known composition of the combustables.

A estimate for the ROHR of the test fuel is calculated, using a closed system analysis. The a plot of the ROHR for the pre-combustion and injection experiment is displayed in a graphical user interface developed for this purpose.

Accuracy of the assumptions and methodes used is discussed, and suggestions for improvements and future work is implemented in the disucussion.
Sammendrag

For å møte en voksende verdensbefolkningens økende behov for transport, står skip drevet av fossile brenser klare til å levere. Lavere fuelforbruk per enhet last transportert, er fordelaktig operatøren av skipet og for miljøet. Redusert fuelforbruk betyr penger spart og mindre utslipp av drivhusgasser. Lavere utslippskrav fra skip til luft medfører at tiltak må iverksettes for å møte disse.

En bedre forståelse av forbrenningsprosessen, er en av nøkkelfaktorene for å forbedre diesel og gasmotorer som er i drift verden rundt i dag. For å øke kunnskapen rundt forbrenningsprosessen, er eksperimenter nødvendig. Presis kjennskap til testforholdene under eksperimentene er nødvendig for å få nøyaktige resultater.

Denne masteroppgaven har sett på en måte å forbedre resultater fra eksperimenter utført i en forbrenningsbombe med konstant volum hos MARINTEK. På nåværende tidspunkt er det ikke noen presis måte å estimere sammensetningen av gassen i bomben når testdrivstoffet sprøytes inn. For å øke trykket og temperaturen til det nivået som trengs for å gjennomføre eksperimentet, brennes en gassblanding bestående av $CO$, $N_2$ og $O_2$. Hvor mye av denne gassen som Brenner er det laget en algoritme for å estimere.

Denne algoritmen tar utgangspunkt i at bomba kan deles opp i to soner, en med helt uforbrent og en med helt forbrent gass. Dette modelleres med total fire differensialligninger, som løses simulatant. Målt trykk er input til disse beregningene, som utføres etter at eksperimentet er ferdig.

Rate of heat release for gassen har blitt beregnet med løsningen av tosone-modellen. Et estimat for rate of heat release for test drivstoffet er også beregnet. Alle relevante resultater gis til brukeren gjennom et grafisk brukergrensensnitt.

Avslutningsvis er nøyaktigheten av de utførte antagelsene droftet, presisjonen på estimatet sett i forhold til nøyaktigheten på målingene, samt forslag til implementeringer som vil øke presisjonen.
Acknowledgements

This thesis is written during the spring semester of 2014 at Department of Marine Technology at NTNU in Trondheim. I will thank my supervisor prof. Harald Valland for thoughtful guidance and help in gaining insight in thermodynamical problemsolving. Thanks also to Maximilian Malin at MARINTEK for providing me with this thesis to help improving his research. I would like to thank the other members of office C1.076 for a good working environment this semester.

Thanks also to Ida Marlen Strand at AMOS for helping me structuring my thoughts and forcing me to keep going when problems seemed unsolvable.

Trondheim, June 10th

Anders Rohde
Contents

Abstract iv
Abstract - Norwegian vi
Acknowledgements viii
Contents x
List of Figures xiii
List of Tables xv
Abbreviations xvi
Physical Constants xvii
Symbols xviii

1 Introduction 1
  1.1 Background and motivation ......................................... 1
  1.2 Problem description .................................................. 3

2 Theoretical Background 4
  2.1 Analysis of the pre-combustion phase .............................. 4
     2.1.1 Ideal gas properties ......................................... 5
     2.1.2 Conservation of energy ....................................... 6
     2.1.3 Resulting equations .......................................... 8
     2.1.4 Fractions of each gas ....................................... 11
     2.1.5 Calculation of the thermodynamic properties ............... 12
     2.1.6 Specifications for the two-zone model ....................... 13
     2.1.7 Mixing of the gases in the two zones ....................... 14
     2.1.8 Heat losses ................................................... 14
  2.2 Rate of heat release (ROHR) ...................................... 17
     2.2.1 Pre-mixed combustion .................................... 17
     2.2.2 Diesel injection experiment ................................. 19
  2.3 Control theory ...................................................... 20

3 Implementation of theoretical background to develop the MATLAB algorithm 22
3.1 Creating structure and import of data ........................................ 22
3.2 Loading of constants and import of measurements ....................... 24
3.3 Thermodynamic properties .................................................... 26
3.4 Heat losses .............................................................................. 29
   3.4.1 Chemical reaction ............................................................. 29
   3.4.2 Fit of pressure curve and Eichelbergs coefficient ................... 29
3.5 Differential equations in the two-zone model ................................. 30
3.6 Rate of heat release .................................................................. 34
   3.6.1 Pre-combustion phase ....................................................... 34
3.7 Injection experiment .................................................................. 36
3.8 Graphical User Interface (GUI) ................................................... 38
4 Discussions regarding accuracy of calculations and future work to
   improve estimation ..................................................................... 42
   4.1 Assumptions .......................................................................... 42
      4.1.1 Perfect mixed gases ....................................................... 42
      4.1.2 Relative combustion rate of \( \text{H}_2 \) and \( \text{CO} \) ................. 43
   4.2 Constant temperature of the rig \( T_{\text{wall}} \) .................................. 44
   4.3 Unaccuracies in measured pressure .......................................... 44
   4.4 Precision of the algorithm ....................................................... 45
   4.5 Future work ......................................................................... 47
      4.5.1 Verification of gas mixture composition ............................. 47
      4.5.2 Heat losses ..................................................................... 47
      4.5.3 ROHR in injection experiment .......................................... 48
A Thermodynamic Coefficients from NASA GLENN DATABASE ......... 49
B Thermochemical Data .................................................................. 51
C Numerical differential ................................................................... 52
D Matlab scripts ............................................................................... 53
   D.1 GUI function and figure ....................................................... 53
   D.2 Function to initiate calculations ............................................. 53
   D.3 Import measurements and calculate gas composition based on user
      input ................................................................................. 55
   D.4 Gas composition after pre-combustion .................................... 62
   D.5 Heat loss coefficient calculation ............................................. 64
   D.6 Heat capacities calculated ..................................................... 67
   D.7 NASA GLENN COEFFICIENTS .............................................. 69
   D.8 Two-zone model execution .................................................... 70
   D.9 Function to update variables in two-zone model ...................... 74
   D.10 Function to calculate the mass fraction in zone 1 and zone 2 ....... 77
   D.11 Function to calculate the thermodynamic properties of zone 1 and
      zone 2 .............................................................................. 78
   D.12 Function to calculate the specific enthalpy in zone 1 and zone 2 ... 81
   D.13 Rate of heat release calculations ........................................... 83
   D.14 Function to calculate kappa for ROHR of injection experiment .... 89
D.15 Output generated on user request .......................... 91

Bibliography ................................................. 94
# List of Figures

1.1 MARPOL Annex VI NO\textsubscript{x} Emission Limits ........................................ 2
1.2 MARPOL Annex VI Fuel Sulfur Limits .................................................. 2
1.3 Combustion Rig layout, made by Maximilian Malin, MARINTEK ............... 3

2.1 Illustration of two-zone model in the CR ............................................. 5
2.2 Control volume around the CR ............................................................. 6
2.3 Control volume for zone 1 and zone 2 .................................................. 7
2.4 Unity-feedback control configuration ................................................ 20

3.1 Flowchart of the MATLAB algorithm .................................................. 23
3.2 Measured pressure from two dynamic and one static pressure sensor during one experiment .......................................................... 25
3.3 Specific heat capacity at constant pressure \( c_p \) ........................................ 26
3.4 Specific heat capacity at constant volume \( c_v \) ........................................ 27
3.5 Heat capacity ratio \( \kappa \) ........................................................................ 27
3.6 Specific enthalpy \( h \) .......................................................................... 28
3.7 Specific internal energy \( u \) ................................................................... 28
3.8 Rate of heat loss in cool down phase Newtons law of cooling with fitted value from Eichelbergs formula ......................................................... 30
3.9 \( \alpha \) from Eq. 2.49 and fitted value for Eichelbergs formula ....................... 31
3.10 p from measurements and fitted with Fourier ...................................... 32
3.11 Differentiation of pressure measurements, numerically and from fitted Fourier function ................................................................. 33
3.12 Mass transported from zone 1 to zone 2 during pre-combustion as % of total mass ........................................................... 34
3.13 Temperature in zone 1 during the pre-combustion ................................... 35
3.14 Volume in zone 1 during pre-combustion as % of total volume ................ 35
3.15 Temperature in zone 2 during pre-combustion ....................................... 36
3.16 Effect of no controller, P-controller and PI-controller on \( \Delta T \) .................... 37
3.17 Plot of ROHR for the pre-combustion phase with numerical differentiation of the mass and with fitted curve .................................................. 38
3.18 Plot of ROHR for the injection experiment with numerical differentiation of pressure and with smoothened curve ...................................... 39
3.19 GUI at startup ....................................................................................... 40
3.20 GUI after finished calculations ............................................................. 41

4.1 Laminar flame speed of CO and \( H_2 \) mixture .......................................... 43
4.2 ROHR from reference experiment plotted by Maximilian Malin, MARINTEK ................................................................. 46
4.3 Resulting temperature curves with varying $\eta_{\text{comb}}$  . . . . . . . . . . . . . . . . . . . . . . . . . . 46
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Molar masses and gas constants for components</td>
<td>24</td>
</tr>
<tr>
<td>4.1</td>
<td>Variation in combustion efficiency and resulting temperature</td>
<td>47</td>
</tr>
<tr>
<td>4.2</td>
<td>Variation in combustion efficiency and resulting temperature</td>
<td>47</td>
</tr>
<tr>
<td>A.1</td>
<td>Thermodynamic coefficients CO, CO₂ and H₂ in the range 200-1000 [K]</td>
<td>49</td>
</tr>
<tr>
<td>A.2</td>
<td>Thermodynamic coefficients N₂, O₂ and H₂O in the range 200-1000 [K]</td>
<td>50</td>
</tr>
<tr>
<td>A.3</td>
<td>Thermodynamic coefficients CO, CO₂ and H₂ in the range 1000-6000 [K]</td>
<td>50</td>
</tr>
<tr>
<td>A.4</td>
<td>Thermodynamic coefficients N₂, O₂ and H₂O in the range 1000-6000 [K]</td>
<td>50</td>
</tr>
<tr>
<td>B.1</td>
<td>Thermochemical data for combustion gases</td>
<td>51</td>
</tr>
</tbody>
</table>
Abbreviations

CR  Combustion Rig
LHV  Lower Heating Value
ROHR  Rate Of Heat Release
LNG  Liquified Natural Gas
Physical Constants

Gas constant \( R_0 = 8.31451 \frac{\text{J}}{\text{molK}} \)
## Symbols

$A$ area $\left[ m^2 \right]$

$a_1$ thermodynamic coefficient $\left[ K^2 \right]$

$a_2$ thermodynamic coefficient $\left[ K \right]$

$a_3$ thermodynamic coefficient $\left[ - \right]$

$a_4$ thermodynamic coefficient $\left[ K^{-1} \right]$

$a_5$ thermodynamic coefficient $\left[ K^{-2} \right]$

$a_6$ thermodynamic coefficient $\left[ K^{-3} \right]$

$a_7$ thermodynamic coefficient $\left[ K^{-4} \right]$

$b_1$ thermodynamic integration coefficient $\left[ K \right]$

$c$ specific heat capacity $\left[ \frac{kJ}{kg \cdot K} \right]$

$c_p$ specific heat capacity at constant pressure $\left[ \frac{kJ}{kg \cdot K} \right]$

$c_v$ specific heat capacity at constant volume $\left[ \frac{kJ}{kg \cdot K} \right]$

$h$ specific enthalpy $\left[ \frac{kJ}{kg} \right]$

$\Delta h_{f,i}^0$ Standard enthalpy of formation $\left[ \frac{MJ}{kg} \right]$

$\Delta T$ temperature difference pre-combustion ended $\left[ K \right]$

$m$ mass $\left[ kg \right]$

$m_1$ mass in zone 1 $\left[ kg \right]$

$m_2$ mass in zone 2 $\left[ kg \right]$

$p$ pressure 2 $\left[ Pa \right]$

$T$ temperature $\left[ K \right]$

$T_1$ temperature in zone 1 $\left[ K \right]$

$T_2$ temperature in zone 2 $\left[ K \right]$

$u$ specific internal energy $\left[ \frac{kJ}{kg} \right]$

$V$ volume $\left[ m^3 \right]$

$V_1$ volume in zone 1 $\left[ m^3 \right]$
<table>
<thead>
<tr>
<th>Symbols</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_2$</td>
<td>volume in zone 2</td>
<td>$[m^3]$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>convection factor</td>
<td>$[\frac{J}{m^2K}]$</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>correction factor in Eichelbergs formula</td>
<td>$[\frac{J}{m^2K\sqrt{PaK}}]$</td>
</tr>
<tr>
<td>$\eta_{comb}$</td>
<td>combustion efficiency</td>
<td>[%]</td>
</tr>
<tr>
<td>$\kappa$</td>
<td>specific heat capacity ratio</td>
<td>[-]</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Background and motivation

Reducing the emissions from combustion of fossil fuels are a topic of major interest. Governments and international organisations make regulations to limit the maximum amount of specific types of emissions from marine diesel engines. Operators want to get more power out of their machinery at the same fuel consumption, saving costs and reduce the production of emission gases.

Marine diesel operators need to meet the regulations made by IMO, especially regarding $NO_x$ and $S$ as IMO regulation in Fig. 1.1 and Fig. 1.2. $NO_x$ formation is due to reaction with the $N_2$ in the air and fuel, while $SO_x$ emissions emerge from sulphur in the fuel.

Better knowledge about the combustion process is a key factor to reduce the emissions from diesel engines. This could lead to reduced fuel consumption due to increased efficiency, lower $NO_x$ formation with a better understanding on how this formation occurs and even implementation of new fuels, like LNG. To gain this understanding experiments is necessary.

Combustion experiments can be performed in different equipment depending on what is to be investigated. Testing real operation properties, an engine equipped with pressure and temperature sensors can be used. For close up investigation of the flame propagation, combustion rigs or combustion bombs can be used. These provide easily visible access to the combustion, which then can be recorded with an high speed camera.
Chapter 1. Introduction and motivation

Figure 1.1: MARPOL Annex VI NO\textsubscript{x} Emission Limits

Figure 1.2: MARPOL Annex VI Fuel Sulfur Limits

The combustion rig (CR) this Master Thesis is related to, is located in the combustion laboratory at MARINTEK, Trondheim. This combustion rig is a fixed volume combustion bomb, with the layout seen in Fig. 1.3. When performing an experiment this rig is first scavanged with pressurized air, to make sure all exhaust gases from
previous experiments are removed. The air inlet is then closed, and when the pressure have stabilized, the exhaust is closed. Air at atmospheric pressure is then filling the CR.

The CR is then charged with a gas mixture consisting of $CO$, $N_2$ and $O_2$, up to a required pressure. Gas inlet valve is then closed. A spark plug then ignites the burnable gas mixture in the rig, increasing both the temperature and pressure. At a set time after the spark ignition, the experiment is performed with injection of liquid fuel. An new experiment can then be prepared.

![Combustion Rig layout](image)

**Figure 1.3:** Combustion Rig layout, made by Maximilian Malin, MARINTEK

### 1.2 Problem description

The aim of this master thesis is to develop a thermodynamic analysis tool useful to verify the test conditions in the combustion rig. Main parameters to investigate is the pressure and temperature when the combustion experiment starts. All calculations is to be done offline, using recorded data measurements from an already performed experiment.

The challenge is to determine the composition of the gas in the rig before injection. This requires to know the composition of the gas in the rig when it is charged with the burnable mixture and after this mixture have been combusted. When these values have been calculated, the rate of heat release in both the gas combustion and liquid fuel combustion is to be calculated.

The thermodynamic analysis tool have been made in MATLAB, providing the user with a graphical program to use.
Chapter 2

Theoretical Background

This chapter describes the theoretical background to build up a model for the combustion in the pre-mixed phase, to get a better estimate of the composition of the gas in the CR during the experiment. The equations needed to calculate the rate of heat release (ROHR) in the pre-mixed phase and in the fuel injection phase is also described.

2.1 Analysis of the pre-combustion phase

The pre-combustion of the mixed gases is taking place in a closed chamber having only heat interaction with the surroundings. This combustion process can be modelled as a reaction going from one zone with unburned gases to a zone with completely burned gases, as illustrated in Fig. 2.1. In both zones the gas is assumed to be perfectly mixed.

Between the two zones there is a mass transport $\dot{m}_{12}$, representing the gas burned. The gas composition in the two zones is clearly stated, zone 1 being the pre-charged gas mixture and zone 2 being products of the completely combustion of the mixture in zone 1. The gases present in each of the two zones are shown in Fig. 2.1. The composition is fixed in both zones during the pre-combustion. $N_2$ is assumed to not react during the combustion, and excess $O_2$ is present in zone 2.
2.1.1 Ideal gas properties

During the pre-combustion the pressure and temperature is relatively low, and it is assumed that the gases can be treated as ideal gases, following the ideal gas law:

\[ pV = mRT \]  \hspace{1cm} (2.1)

where \( p \) is the pressure, \( V \) is the volume, \( m \) is the mass, \( R \) the gas constant and \( T \) is the temperature.

To study the changes on one or several of the parameters, the ideal gas law can be logarithmically differentiated:

\[ \frac{\dot{p}}{p} + \frac{\dot{V}}{V} = \frac{\dot{m}}{m} + \frac{\dot{R}}{R} + \frac{\dot{T}}{T} \]  \hspace{1cm} (2.2)

where \( \dot{p}, \dot{V}, \dot{m}, \dot{R} \) and \( \dot{T} \) is the time derivative of pressure, volume, mass, gas constant and temperature. \( R \) is a constant, and will therfore not be time dependent. Taking \( R \) out of the equation and collect all factors on one side gives:

\[ \frac{\dot{p}}{p} - \frac{\dot{T}}{T} + \frac{\dot{V}}{V} - \frac{\dot{m}}{m} = 0 \]  \hspace{1cm} (2.3)
2.1.2 Conservation of energy

Through experiments, it have been proved that energy is conserved. Some of the experiment was performed by James Joule (1818-1889) (Tipler and Mosca, 2008). In Joule’s most famous experiment, he found the potential energy needed to heat 1 lb of water by 1 degree F. This was the starting point for what is known today as the first law of thermodynamics.

The combustion rig needs to be studied in two different ways regarding the first law of thermodynamics. First, it is treated globally as a closed system with a control volume (CV) around the CR, illustrated in Fig. 2.2. Across the CV border it is only heat interaction with the surroundings. Secondly, the CR is separated into two zones, each zone an open system with heat, mass and work interaction, illustrated in Fig. 2.3.

Generally, the first law of thermodynamics is:

\[ U = Q - W \]  \hspace{1cm} (2.4)

where \( U \) is the total internal energy in the system, \( Q \) is the total heat in or out of the system and \( W \) is the total of work either made on the system or from the system on the surroundings.

Evaluating the first law of thermodynamics with respect to time:
Figure 2.3: Control volume for zone 1 and zone 2

\[
\frac{\partial U}{\partial t} = \frac{\partial Q}{\partial t} - \frac{\partial W}{\partial t} \quad \text{or} \quad \dot{U} = \dot{Q} - \dot{W}
\] (2.5)

where both equations are equal, using Newton’s notation for time derivative with the dot above the variable.

For the change in time, the internal energy \( U \) can be specified:

\[
\dot{U} = mc\dot{T} + um
\] (2.6)

where \( m \) is the total mass of the system, \( c \) is the specific heat capacity, \( \dot{T} \) is the rate of temperature change, \( u \) is the specific internal energy and \( \dot{m} \) is the rate of mass change.

For the work \( W \) the rate of change is given as:

\[
\dot{W} = p\dot{V}
\] (2.7)

where \( p \) is the pressure and \( \dot{V} \) is the rate of volume change.
For a closed system as the combustion rig is assumed to be during the first phase of the experiment, there is no mass interaction or change in total volume. This reduces the first law of thermodynamics to:

\[ \dot{U} = \dot{Q} = mc_v \dot{T} \]  

(2.8)

Equation 2.8 can be used to calculate the internal energy change in the CR due to heat energy transferred from the gas to the walls in the CR.

For the mass and volume interaction between the two zones in Fig. 2.1, the combustion process is treated as an open system. This two open systems have mass and work interactions when the volume of each of the two zones change during the pre-combustion. The first law of thermodynamics for an open combustion system is given as:

\[ mc_v \dot{T} + p \dot{V} + u \dot{m} = \dot{Q} + h_F \dot{m}_F \]  

(2.9)

where \( h_F \) is specific enthalphy of the fuel and \( \dot{m}_F \) is the rate of fuel mass flow into the system.

### 2.1.3 Resulting equations

In the two-zone model zone 1 and zone 2 have the same pressure, as illustrated in Fig. 2.1. Temperature, volume and mass is determined inside each zone, seen in Fig. 2.3. For zone 1 Eq. 2.3 then becomes:

\[ \frac{\dot{p}}{p} \frac{T}{T_1} + \frac{\dot{V}}{V_1} - \frac{\dot{m}_1}{m_1} = 0 \]  

(2.10)

where the subscript 1 represent zone 1.

Fuel in the two-model is the burnable gas transported from zone 1 to zone 2. The rig is assumed to be completely sealed off against the surroundings, so no gas is lost to leakage or transported into the system. The rate of mass change in zone 1 \( \dot{m}_1 \) then equals the rate of fuel mass change \( \dot{m}_F \):
\[ m_1 = \dot{m}_F \quad (2.11) \]

The fuel being represented by zone 1, give that:

\[ h_1 = h_{F} \quad (2.12) \]

where \( h_1 \) is the specific enthalpy of zone 1 and \( h_{F} \) is the fuel specific enthalpy.

Combining Eq. 2.9, 2.11 and 2.12 for zone 1 give:

\[ m_1 c_{v1} \dot{T}_1 + p \dot{V}_1 + u_1 \dot{m}_1 = \dot{Q}_1 + h_{1} \dot{m}_1 \quad (2.13) \]

For zone 2 Eq. 2.3 becomes:

\[ \frac{\dot{p}}{p} \frac{T_2}{\dot{T}_2} + \frac{\dot{V}_2}{V_2} - \frac{\dot{m}_2}{m_2} = 0 \quad (2.14) \]

where subscript 2 represent zone 2.

The mass transport can only take place between zone 1 and zone 2, such that:

\[ \dot{m}_1 + \dot{m}_2 = 0 \rightarrow \dot{m}_2 = -\dot{m}_1 \quad (2.15) \]

where \( \dot{m}_2 \) is the rate of mass change in zone 2.

Now combining Eq. 2.9, 2.11, 2.12 and 2.15 give:

\[ m_2 c_{v2} \dot{T}_2 + p \dot{V}_2 + u_2 \dot{m}_2 = \dot{Q}_2 - h_{1} \dot{m}_2 \quad (2.16) \]

The total volume of the CR consists of the two zones:

\[ V_1 + V_2 = V \quad (2.17) \]
and since the CR is a constant volume combustion chamber the total volume change is zero:

\[ \dot{V}_1 + \dot{V}_2 = 0 \]  

(2.18)

A system of equations have been set up in matrix notation. For zone 1 the Eq. 2.10 and Eq. 2.11 is implemented, for zone 2 Eq. 2.14 and Eq. 2.16 and the relations between them described in Eq. 2.15 and Eq. 2.18 give the following system:

\[
\begin{bmatrix}
\frac{1}{p} & -\frac{1}{T_1} & \frac{1}{V_1} & -\frac{1}{m_1} & 0 & 0 \\
0 & m_1c_v & p & u_1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{p} & 0 & 0 & \frac{1}{m_2} & -\frac{1}{T_2} & \frac{1}{V_2} \\
0 & 0 & 0 & -u_2 & m_2c_{v_2} & p
\end{bmatrix}
\begin{bmatrix}
\dot{p} \\
\dot{T}_1 \\
\dot{V}_1 \\
\dot{m}_1 \\
\dot{T}_2 \\
\dot{V}_2
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
\dot{Q}_1 + h_1\dot{m}_1 \\
\dot{V}_1 \\
\dot{m}_1 \\
\dot{Q}_2 - h_1\dot{m}_2
\end{bmatrix}
\]

The relation in Eq. 2.18 also give:

\[ \dot{V}_1 = -\dot{V}_2 \]  

(2.19)

Implementing Eq. 2.19 the system of equations can be reduced to:

\[
\begin{bmatrix}
\frac{1}{p} & -\frac{1}{T_1} & \frac{1}{V_1} & -\frac{1}{m_1} & 0 \\
0 & m_1c_v & p & u_1 & 0 \\
0 & 0 & 0 & 1 & 0 \\
\frac{V_2}{p} & 0 & -1 & \frac{V_2}{m_2} & \frac{V_2}{T_2} \\
0 & 0 & -p & -u_2 & m_2c_{v_2}
\end{bmatrix}
\begin{bmatrix}
\dot{p} \\
\dot{T}_1 \\
\dot{V}_1 \\
\dot{m}_1 \\
\dot{T}_2
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
\dot{Q}_1 + h_1\dot{m}_1 \\
\dot{V}_1 \\
\dot{m}_1 \\
\dot{Q}_2 - h_1\dot{m}_2
\end{bmatrix}
\]

Defining the mass transport from zone 1 to zone 2 as:

\[ \dot{m}_1 = -\dot{m}_{12} \]  

(2.20)

Equation 2.20 and 2.15 reduces the system of equations to:
Finally the system of equations can be solved for $\dot{p}$, this being known from measurements:

\[
\begin{bmatrix}
\frac{1}{\dot{p}} & -\frac{1}{T_1} & \frac{1}{V_1} & 0 \\
0 & m_1 c_v & p & 0 \\
-\frac{1}{\dot{p}} & 0 & \frac{1}{V_2} & \frac{1}{T_2} \\
0 & 0 & p & -m_2 c_v
\end{bmatrix}
\begin{bmatrix}
\dot{p} \\
\dot{T}_1 \\
\dot{V}_1 \\
\dot{T}_2
\end{bmatrix}
= 
\begin{bmatrix}
\frac{-\dot{m}_{12}}{m_1} \\
\dot{Q}_1 + (u_1 - h_1)\dot{m}_{12} \\
\frac{-\dot{m}_{12}}{m_2} \\
\dot{Q}_2(u_2 - h_1)\dot{m}_{12}
\end{bmatrix}
\]

2.1.4 Fractions of each gas

Mass and molar fractions is necessary to perform the calculations. Mass fractions have been used to calculate the total thermodynamic state of a mixed gas. Molar fractions will be used to calculate the number of mole of each substance in the reactants and products of the combustion.

Definition of the molar fraction is:

\[
y_i = \frac{n_i}{\sum n_i}
\]

where $y_i$ is the molar fraction, $n_i$ is the number of moles of a substance, and $\sum n_i$ is the total amount of moles.

In an similar matter, the mass fraction is defined as:

\[
x_i = \frac{m_i}{\sum m}
\]

where $x_i$ is the mass fraction, $m_i$ mass of the substance and $\sum m$ is the total mass of the gas.

If the molar fraction is known, the mass fraction can be calculated with:
\[ x_i = \frac{y_i M_i}{\sum_{j=1}^{k} M_j y_j} \]  

(2.23)

where \( M_i \) is the molar mass of the specific substance, and \( \sum_{j=1}^{k} M_j y_j \) is the sum of molar mass multiplied with molar fraction for each substance of the gas, up to substance \( k \).

If the mass fraction is known, the molar fraction can be calculated with:

\[ y_i = \frac{x_i R_0}{M_i \sum_{j=1}^{k} R_j x_j} \]  

(2.24)

where \( R_0 \) is the universal gas constant, \( R_j \) is the gas constant for each gas and \( x_j \) is the mass fraction of each gas.

### 2.1.5 Calculation of the thermodynamic properties

With the gases in the unburned and burned zones perfect mixed, the thermodynamic properties for the gases are equal in the entire zone. Heat capacity for constant volume, \( c_v \), can be calculated based on the pressure and temperature for each specie and for the entire zone it can be found based on the mass fraction of each substance.

Using the equations in the NASA GLENN database for calculating specific heat capacity at constant pressure and specific enthalpy \([McBride et al., 2002]\):

\[ \frac{C_{p}^o(T)}{R} = a_1 T^{-2} + a_2 T^{-1} + a_3 + a_4 T + a_5 T^2 + a_6 T^3 + a_7 T^4 \]  

(2.25)

\[ \frac{H^o(T)}{RT} = -a_1 T^{-2} + a_2 \frac{ln(T)}{T} + a_3 + a_4 \frac{T}{2} + a_5 \frac{T^2}{3} + a_6 \frac{T^3}{4} + a_7 \frac{T^4}{5} + \frac{b_1}{T} \]  

(2.26)

here \( a_1 \) to \( a_7 \) and \( b_1 \) is factors given in the database, with temperature range from 200 - 1000 [K] and for 1000 - 6000 [K]. Values for all gases used can be found in App. A.

From this equations the thermodynamic properties can be evaluated:
Chapter 2.

\[ c_p(T) = \left[ \frac{C_p^0(T)}{R} \right] \cdot R \]  
\hspace{1cm} (2.27)

where \( c_p(T) \) is the specific heat capacity at constant pressure as a function of temperature. Further the heat capacity at constant volume \( c_v(T) \) as a function of temperature can be calculated with:

\[ c_v(T) = c_p(T) - R \]  
\hspace{1cm} (2.28)

The specific enthalphy as a function of temperature \( h(T) \) is calculated with:

\[ h(T) = \left[ \frac{H^0(T)}{RT} \right] \cdot RT \]  
\hspace{1cm} (2.29)

For all gases the specific internal energy \( u(T) \) can be calculated:

\[ u(T) = h(T) - RT \]  
\hspace{1cm} (2.30)

2.1.6 Specifications for the two-zone model

Knowing the thermodynamic properties for each gas at the temperature in the zone, the final properties can be calculated using the mass fraction of each gas in the respective zone. The equation used for the specific heat in zone 1 is seen below.

\[ c_v^{(1)}(T) = c_{v,CO}(T) \cdot x_{CO}^{(1)} + c_{v,H_2}(T) \cdot x_{H_2}^{(1)} + c_{v,O_2}(T) \cdot x_{O_2}^{(1)} + c_{v,N_2}(T) \cdot x_{N_2}^{(1)} \]  
\hspace{1cm} (2.31)

In zone two the equation is extended with the to gases of \( H_2O \) and \( CO_2 \) seen in Eq.

\[ c_v^{(2)}(T) = c_{v,CO}(T) \cdot x_{CO}^{(2)} + c_{v,H_2}(T) \cdot x_{H_2}^{(2)} + c_{v,CO_2}(T) \cdot x_{CO_2}^{(2)} + c_{v,O_2}(T) \cdot x_{O_2}^{(2)} + c_{v,N_2}(T) \cdot x_{N_2}^{(2)} + c_{v,H_2O}(T) \cdot x_{H_2O}^{(2)} \]  
\hspace{1cm} (2.32)
In the same manner the specific internal energy in zone 1 is found with:

\[ u^{(1)}(T) = u_{CO}(T) \cdot x_{CO}^{(1)} + u_{H_2}(T) \cdot x_{H_2}^{(1)} + u_{O_2}(T) \cdot x_{O_2}^{(1)} + u_{N_2}(T) \cdot x_{N_2}^{(1)} \]  

(2.33)

And the specific internal energy for zone 2 is calculated with:

\[ u^{(2)}(T) = u_{CO}(T) \cdot x_{CO}^{(2)} + u_{H_2}(T) \cdot x_{H_2}^{(2)} + u_{CO_2}(T) \cdot x_{CO_2}^{(2)} + u_{O_2}(T) \cdot x_{O_2}^{(2)} + u_{N_2}(T) \cdot x_{N_2}^{(2)} + u_{H_2O}(T) \cdot x_{H_2O}^{(2)} \]  

(2.34)

The specific enthalpy \( h \) represent the enthalphy of the fuel, and in the two-zone model this is the mass transported from zone 1 to zone 2. Then the specific is not necessary for zone 2, and for zone 1 it can be calculated with:

\[ h^{(1)}(T) = h_{CO}(T) \cdot x_{CO}^{(1)} + h_{H_2}(T) \cdot x_{H_2}^{(1)} + h_{O_2}(T) \cdot x_{O_2}^{(1)} + h_{N_2}(T) \cdot x_{N_2}^{(1)} \]  

(2.35)

### 2.1.7 Mixing of the gases in the two zones

After the pre-combustion is finished, the two zones mixes. This process is assumed to happen instantaneously. The resulting temperature after the mixing can be found by solving the equation:

\[ m_1 h_1(T) + m_2 h_2(T) = (m_1 + m_2) \cdot h_{mix}(T) \]  

(2.36)

where \( m_1 \) is the mass in zone 1, \( h_1(T) \) is the specific enthalphy in zone 1, \( m_2 \) is the mass in zone 2, \( h_2(T) \) is the enthalphy in zone 2 and \( h_{mix}(T) \) is the resulting enalphy after mixing.

### 2.1.8 Heat losses

Heat losses can be calculated with the first law of thermodynamics as described in Eq. 2.8. Another way of calculating heat losses, is with the convective law, known as Newton’s law of cooling from **Moran and Shapiro** [2010].
\[ \dot{Q} = \alpha(T) \cdot A \cdot (T_{\text{gas}} - T_{\text{wall}}) \]  

(2.37)

where \( \alpha(T) \) is heat transfer coefficient as a function of temperature \( T \), \( A \) is the area where the gas is exposed to the wall, \( T_{\text{gas}} \) is the gas temperature and \( T_{\text{wall}} \) is the wall temperature. The wall temperature \( T_{\text{wall}} \) is assumed constant over the entire experiment.

The gas temperature \( T_{\text{gas}} \) needs to be determined. Solving the ideal gas law, Eq. 2.1 for temperature, gives the gas temperature:

\[ T_{\text{gas}} = \frac{pV}{mR} \]  

(2.38)

Assuming the mass and volume does not change during one cycle, and that the pressure is measured, the unknown is the gas constant.

The gas constant for a mixture is dependend on the composition of the mixture. Using a combustion efficiency, \( \eta_{\text{comb}} \) defined as:

\[ \eta_{\text{comb}} = \frac{m_{\text{CO,comb}}}{m_{\text{CO}}} = \frac{m_{\text{H}_2,\text{comb}}}{m_{\text{H}_2}} \]  

(2.39)

where \( m_{\text{CO,comb}} \) is combusted mass of \( \text{CO} \), \( m_{\text{CO}} \) is the total mass of \( \text{CO} \), \( m_{\text{H}_2,\text{comb}} \) is the combusted mass of \( \text{H}_2 \) and is the total mass of \( \text{H}_2 \). The combustion efficency is assumed to be equal for \( \text{CO} \) and \( \text{H}_2 \).

The composition of the gas after an combustion with combustion efficency \( \eta_{\text{comb}} \) have been calculated requiring that the following equation is in chemical equilibrium:

On general form, the chemical equilibrium can be stated as:

\[ n_{\text{O}_2,\text{H}_2,\text{N}_2,\text{CO}_2,\text{H}_2\text{O},\text{H}_2} \]  

(2.40)

where \( n_j^R \) is the number of moles of gas \( j \) as a reactant, and \( n_j^P \) is the number of moles of gas \( j \) as a product of the reaction.
Using $\eta_{comb}$ defined in Eq. 2.39 the number of moles of reactants can be calculated with:

\[ n_{O_2}^P = n_{O_2}^R - \frac{1}{2} \eta_{comb} \left[ n_{CO}^R + n_{H_2}^R \right] \] (2.41)

\[ n_{H_2}^P = (1 - \eta_{comb}) \cdot n_{H_2}^R \] (2.42)

\[ n_{CO}^P = (1 - \eta_{comb}) \cdot n_{CO}^R \] (2.43)

\[ n_{N_2}^P = n_{N_2}^R \] (2.44)

\[ n_{CO_2}^P = \eta_{comb} \cdot n_{CO}^R \] (2.45)

\[ n_{H_2O}^P = \eta_{comb} \cdot n_{H_2}^R \] (2.46)

Including Eq. 2.41 to 2.46 into Eq. 2.40 results in:

\[ n_{O_2}^R \cdot O_2 + n_{CO}^R \cdot CO + n_{H_2}^R \cdot H_2 + n_{N_2}^R \cdot N_2 \rightarrow n_{O_2}^P \left( n_{O_2} - \frac{1}{2} \eta_{comb} \left[ n_{CO} + n_{H_2} \right] \right) O_2 \] (2.47)

\[ + (1 - \eta_{comb}) \cdot [n_{CO} \cdot CO + n_{H_2} \cdot H_2] + \eta_{comb} [n_{CO_2} \cdot CO_2 + n_{H_2O} \cdot H_2O] + n_{N_2} \cdot N_2 \]

With the composition after combustion known, the mass fraction can be found with Eq. 2.21 and Eq. 2.22. With the mass fraction, $x_i$ calculated for each gas, the gas constant $R$ for the mixture is found with:

\[ R = \sum_{j=1}^{j=k} \frac{R_0}{M_j} x_j = \sum_{j=1}^{j=k} R_j x_j \] (2.48)
where $R_0$ is the universal gas constant, $M_j$ is the molar mass of gas $j$, $x_j$ is the mass fraction of gas $j$, $R_j$ is the gas constant of gas $j$ and $k$ is total number of gases. $R_j$ is shown in Tab. 3.1.

Combining Eq. 2.8 and Eq. 2.37 solving for $\alpha$ gives:

$$\alpha = \frac{mc_v \dot{T}}{A \cdot (T_{gas} - T_{wall})} \quad (2.49)$$

Assuming that the combustion efficiency is close to unity, this $\alpha$ can be used to calculate the heat loss from zone 2 containing the burned gases.

To include the effect of pressure in the calculation of $\alpha$, Eichelberg’s formula from (Stapersma, 2010) can be:

$$\alpha = \epsilon \cdot \sqrt{pT} \quad (2.50)$$

where $\epsilon$ is a correction factor, $p$ is the pressure and $T$ is the temperature. $\epsilon$ can be found setting Eq. 2.49 and Eq. 2.50 equal in one point or by optimizing over a period.

## 2.2 Rate of heat release (ROHR)

Rate of Heat Release (ROHR) is defined as the rate at which the chemical energy in the fuel is converted into heat in the combustion.

### 2.2.1 Pre-mixed combustion

For the pre-mixed phase the rate of mass transport from zone 1 to zone 2 is known. The precise composition of the fuel, $H_2$ and CO is also known. ROHR ($\dot{Q}_{comb}$) is then calculated:

$$\dot{Q}_{comb} = h_n \cdot \dot{m}_f \quad (2.51)$$

where $h_n$ is the lower heating value of the fuel and $\dot{m}_f$ is the rate of mass flow.
From Ch. 3.5.2 and Ch. 3.5.3 in (Heywood, 1988), the lower heating value \( h_n \) when knowing the specific composition can be calculated with:

\[
H_0^P = \sum_{products} n_i \Delta h_{f,i}^0
\]  

(2.52)

\[
H_0^R = \sum_{reactants} n_i \Delta h_{f,i}^0
\]  

(2.53)

where \( H_0^P \) is the enthalpy of products, \( H_0^R \) is the enthalpy of reactants, \( n_i \) is the number of moles and \( \Delta h_{f,i}^0 \) is the standard enthalpy of formation for each substance. Combining Eq. 2.52 and Eq. 2.53 leads to:

\[
LHV = -(\Delta H) = H_0^P - H_0^R
\]  

(2.54)

where \( LHV \) is the lower heating value and \( \Delta H \) is the enthalpy increase due to the combustion.

An equivalent lower heating value for the mixture of \( CO \) and \( H_2 \) have been calculated with:

\[
LHV_{12} = LHV_{CO}x_{CO,1} + LHV_{H_2}x_{H_2,1}
\]  

(2.55)

where \( LHV_{12} \) is the lower heating value for the mixture containing \( CO \) and \( H_2 \), with mass fractions \( x_{CO,1} \) for \( CO \) and \( x_{H_2,1} \) for \( H_2 \) in zone 1.

Assuming that the \( H_2 \) and \( CO \) burns with the same rate as the mass transported from zone 1 to zone 2, ROHR for the pre-combustion phase can be calculated with:

\[
\dot{Q}_{12} = LHV_{12}\dot{m}_{12}
\]  

(2.56)

where \( \dot{m}_{12} \) is the rate of mass transport from zone 1 to zone 2.
2.2.2 Diesel injection experiment

For the diesel injection only the pressure is determined exactly. As described in Ch. 10.4 in [Heywood, 1988], the following problems also arise for the CR:

- Mass of fuel is added as a liquid. This vaporizing and mixing process between fuel and air produce non-uniform fuel/air ratio and is time variant.
- The composition of burned gases is unknown.
- The accuracy of available heat transfer predictions is not well defined.

From the first law of thermodynamics for an open system, stated in Eq. 2.9, an apparent net heat-release rate can be formulated as:

\[ \dot{Q}_n = \dot{Q}_{ch} - \dot{Q}_{ht} \]  

(2.57)

where \( \dot{Q}_{ch} \) is the gross heat-release rate, equal to the chemical energy released from the burning fuel and \( \dot{Q}_{ht} \) is the rate of heat-transfer to the walls.

Assuming that the fuel enthalpy \( h_F \) is negligible, Eq. 2.9 can be written as:

\[ \dot{Q}_n = p\dot{V} + mc_v\dot{T} \]  

(2.58)

With \( m, R \) and \( V \) assumed constant, it follows from the ideal gas law that:

\[ \frac{\dot{p}}{p} = \frac{\dot{T}}{T} \]  

(2.59)

Equation 2.59 can be used to eliminate \( T \) from Eq. 2.58 including that \( \dot{V} = 0 \):

\[ \dot{Q}_n = mc_v\dot{T}\frac{\dot{p}}{p} \]  

(2.60)

Transforming the ideal gas law to:
Chapter 2.

\[
\frac{V}{R} = \frac{T_m}{p}
\]

reduces Eq. (2.60) to:

\[
\dot{Q}_n = \frac{c_v}{R} V \dot{p}
\]

Finally the relation:

\[
\frac{c_v}{R} = \frac{c_v}{c_p - c_v} = \frac{c_v}{c_v \left( \frac{c_v}{c_p} - 1 \right)} = \frac{1}{\kappa - 1}
\]

can be implemented into Eq. (2.62)

\[
\dot{Q}_n = \frac{1}{\kappa - 1} V \dot{p}
\]

which is the form ROHR have been calculated in this thesis.

2.3 Control theory

A general description of a control system with a feedback loop is illustrated in Fig. 2.4.

![Unity-feedback control configuration](Figure 2.4: Unity-feedback control configuration)

\[ y_0 \]
\[ e \]
\[ Controller \]
\[ u \]
\[ System \]
\[ y \]

\[ y_0 \] is the set point, \( y \) is the output response from the system leading to the error \( e \) between the set point and the output, which is feed into the Controller, generating a controller output \( u \) feed into the System to adjust \( y \).

The controller could be a P-controller, where the controller output is calculated with:
\[ u(t) = K_p \cdot e(t) \]  

(2.65)

where \( K_p \) is the proportional gain, and both \( u \) and \( e \) are functions of time. \( K_p \) needs to be tuned to get a required reaction speed. If its value is too small, the reaction against the set value takes too long, and if it is too high it makes the system unstable and introduces oscillations on the system output. A P-controller can reduce the error, but may be unsufficient to erase it completely. The error may be reduced to a stable value unequal to zero.

Using an PI-controller with both proportional gain and integration constant makes it possible to remove the steady state error. From [Balchen et al. 2003](#) an continuous PI-controller can be modeled as:

\[
u(t) = K_p \left( e(t) + \frac{1}{T_i} \int_0^t e(\sigma) \, d\sigma \right)
\]

(2.66)

where \( T_i \) is the integration constant, and \( \sigma \) is a integration variable to integrate up the error \( e \) from start to point \( t \).

For a discrete case the PI-controller is modeled at point \( k \) with:

\[
u[k] = u[k - 1] + K_p \left( 1 + \frac{T}{2T_i} \right) e[k] - K_p \left( 1 - \frac{T}{2T_i} \right) e[k - 1]
\]

(2.67)

where \( T \) is the sampling time, given as the time between each measurement of the system output.
Chapter 3

Implementation of theoretical background to develop the MATLAB algorithm

This chapter presents how the theoretical background presented in chapter 2 have been implemented in MATLAB. Some of the intermediate results are presented for each step performed.

3.1 Creating structure and import of data

Several computations is required to find an reasonable value for the temperature in the CR when the injection experiment starts. To keep track of the different calulations performed, an logical structure had to be developed. All major calulations was decided to put in their own functions with logical names to make editing of them easier.

A simplified overview of the program structure is shown in Fig. 3.1. The circle on top is the input of measured data from the experiment of interest. First the exact composition of the mixture is determined based on charge pressure of air and combustable gas. Then the absolute pressure curve is determinied from the dynamic pressure sensor and the initial difference between the dynamic pressure sensor and the stativ pressure sensor.
Gas and air composition, combined with the pre-set combustion efficiency, is then used to calculate the composition in the CR after the pre-combustion. Knowing the pressure and composition of the gas, the coefficient $\epsilon$ in Eichelbergs formula in Eq. 2.50 can be found for the cool down phase. The cool down phase is the phase from pre-combustion is ended to liquid fuel is injected.

$\epsilon$ is then passed on to solving the two-zone model. With the pressure and heat loss known, the necessary mass of gas combusted going from zone 1 til zone 2 can be calculated.
The total mass transported give the combustion efficiency, $\eta_{comb}$. If $\eta_{comb}$ difference more than the set value between the current and last calculation, the calculation of pre-combustion composition, heat loss and two-zone solving is done again.

Another parameter to check against is the difference in temperature resulting from the two-zone model and the resulting composition from the chemical equation, named $\Delta T$. This difference is used to improve the estimate of rate of heat loss from zone 2. If the temperature difference $\Delta T$ becomes smaller than a set value, calculation of rate of heat release (ROHR) is performed. The result of this calculation is then sent to the output circle to be displayed in the GUI developed for simplifying the execution of this algorithm.

### 3.2 Loading of constants and import of measurements

When initiating the algorithm, molar masses and the universal gas constant is loaded. The universal gas constant value used is from [McBride et al. 2002], with a value of $R_0 = 8.31451 \left[\frac{J}{molK}\right]$. With the molar masses for all six gases implemented, the gas constant for each gas is calculated. The values used is seen in Tab. 3.1.

<table>
<thead>
<tr>
<th>Species</th>
<th>$M \left[\frac{g}{mol}\right]$</th>
<th>$R \left[\frac{J}{kgK}\right]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O_2$</td>
<td>31.99880</td>
<td>259.845</td>
</tr>
<tr>
<td>$N_2$</td>
<td>28.01340</td>
<td>296.805</td>
</tr>
<tr>
<td>$CO$</td>
<td>28.01010</td>
<td>296.840</td>
</tr>
<tr>
<td>$H_2$</td>
<td>2.015880</td>
<td>4124.506</td>
</tr>
<tr>
<td>$CO_2$</td>
<td>44.00950</td>
<td>188.925</td>
</tr>
<tr>
<td>$H_2O$</td>
<td>18.01528</td>
<td>461.525</td>
</tr>
</tbody>
</table>

**Table 3.1:** Molar masses and gas constants for components

Measured values from the experiment to be investigated, are stored in an Excel-spreadsheet. This spreadsheet have a known lay-out, which is necessary for the import of data to MATLAB. The imported data:

- Measurements from the dynamic pressure sensor
- Initial static pressure
- Initial temperature in the gas and the wall of the rig
- Time relative to ignition
• Delay from ignition to injection

Pressure curves made from measurements during an experiment performed January 31st 2014, is shown in Fig. 3.2. This measurement have been used to calculate the other results described in this thesis. The reference pressure in the laboratory is set to be \( p_{atm} = 100[kPa] \), so the pressure curves made from the dynamic pressure sensors are found with:

\[
p_i = p_{stat,0} - p_{dyn,0} + p_{dyn,i} + p_{atm}
\]  

(3.1)

where \( p_{stat,0} \) is the initial pressure measured by the static pressure sensor, \( p_{dyn,0} \) is the initial pressure measured by the dynamic pressure sensor, \( p_{dyn,i} \) is the dynamic pressure measured at time \( i \) and \( p_{atm} \) is to get the absolute pressure.

As can be seen in Fig. 3.2 the pressure from the static sensor and the second dynamic pressure sensor, gets and stays higher than the pressure curve based on the measurements.
of dynamic pressure sensor one. The static and second dynamic pressure sensor are exposed to direct gas flame in the end of the pre-combustion. This exposure introduce a large disvrtion to the sensors, and their values are discarded.

3.3 Thermodynamic properties

The necessary thermodynamic properties, \( u, h, c_v \) and \( \kappa \) are calculated using the equations described in Section 2.1.5. The calculated values for all six gases in the temperature range 250-3000 [K] are shown in Fig. 3.3 to 3.7.

For all calculations were one or several of these properties are needed, they are calculated in a specific function for that purpose. The first property to be calculated, is \( \kappa \) solving Eq. 2.64. This is done in several stages. First the gas composition and temperature of the gas mixture is passed to the calculating function. This function then pass the temperature on to a function containing all the coefficients in Tab. A.1 to A.4 and checking the temperature. If the temperature is lower than 200 [K], the function returns
Chapter 3. Implementation and development

**Figure 3.4:** Specific heat capacity at constant volume $c_v$

$CO, CO_2, N_2, O_2, H_2O$ to scale  
$H_2$ divided by 10

**Figure 3.5:** Heat capacity ratio $\kappa$
Figure 3.6: Specific enthalpy $h$

Figure 3.7: Specific internal energy $u$
an error stating: "Temperature to low". If the temperature is higher than 6000 [K], the function return an error stating: "Temperature to high". When the temperature is in the working range of the function, 200-6000 [K], a final control is performed to check which range the coefficients to return is in, 200-1000 [K] or 1000-6000 [K].

The coefficients are then returned to the function calling them, which calculate the required thermodynamic property for each gas. Finally $\kappa$ is calculated using the mass fraction of each substance, and return this to the main function. This if performed for all cases were an thermodynamic property is required.

### 3.4 Heat losses

#### 3.4.1 Chemical reaction

Initially the mass of each gas $m_i$ is calculated. Knowing the mass, the number of moles $n_i$ of each substance can be found with:

$$n_i = \frac{m_i}{M_i} \tag{3.2}$$

where $n_i$ is the number of moles, $m_i$ the mass fraction and $M_i$ is the molar mass of substance $i$.

For the first calculation, a combustion efficiency $\eta_{\text{comb},0}$ is assumed. Default value for $\eta_{\text{comb},0}$ is 95%, corresponding to the range for SI-engines of 93-98 % given in [Heywood, 1988]. The resulting composition after pre-combustion can then be calculated with Eq. 2.40.

The next step is to calculate the molar fraction in the products are calculated using Eq. 2.21, giving the mass fractions with Eq. 2.23.

#### 3.4.2 Fit of pressure curve and Eichelbergs coefficient

With the $R$ known after pre-combustion, the temperature in the cool-down phase can be precisely determined by the ideal gas law given in Eq. 2.59. Then the rate of heat
transport $\dot{Q}$ is then calculated with the last part of Eq. 2.64. The calculated $\dot{Q}$ with the resulting fit for Eichelbergs formula is shown in Fig. 3.8. The procedure to find $\kappa$ and all the other thermodynamic properties is described in section 2.1.5.

With the rate of heat loss calculated, the value for $\alpha$ in Eq. 2.49 is calculated. Using a built-in least-squares method in MATLAB, the value for the Eichelberg coefficient $\epsilon$ is estimated. The resulting fit for $\alpha$ is seen in Figure 3.9. The estimated $\epsilon$ is used to calculate the rate of heat loss from zone 2 when solving of the differential equations. Zone 2 is used since the resulting composition is assumed to be almost completely burned, and then more similar to zone 2 than zone 1.

### 3.5 Differential equations in the two-zone model

Having found an estimation for the heat losses, all necessary input to solve the two-zone model is found. Before combustion and computations start, zone 2 containing the
burned gases needs to be given an initial mass $m_{2,0}$ and an initial volume $V_{2,0}$. Due to some computational problems if the initial value is too small, these values are set to 3% of the total mass and volume of the CR.

Several methods were tried for calculating $\dot{p}$ from the pressure curve, such as the numerical differential method described in App. The final solution was to make a smooth curve with a Fourier fit function in MATLAB, on the form:

$$f(x) = a_0 + a_1 \cdot \cos(w \cdot x) + b_1 \cdot \sin(w \cdot x)$$
$$+ a_2 \cdot \cos(2 \cdot w \cdot x) + b_2 \cdot \sin(2 \cdot w \cdot x) \ldots$$
$$+ a_n \cdot \cos(n \cdot w \cdot x) + b_n \cdot \sin(n \cdot w \cdot x)$$

(3.3)

where $x$ is the input variable, $a_0$ to $a_n$ and $b_1$ to $b_n$ is fitting parameters, $w$ is to make the fit periodical. $n$ can be varied between 1 and 8. The Fourier fit against the pressure
curve for the pre-combustion phase is shown in Fig. 3.10. The resulting continuous $\dot{p}$ is plotted against the numerical solution in Fig. 3.11.

![Graph showing pressure $p$ over time](image)

**Figure 3.10:** $p$ from measurements and fitted with Fourier

The output from the two zone model is the mass transport from zone 1 to zone 2, which give the combustion efficiency $\eta_{comb}$:

$$\eta_{comb} = \frac{m_{12}}{m_{tot}}$$ \hspace{1cm} (3.4)

which is equal to $\eta_{comb}$ used in the chemical equation.

where $m_{12}$ is the mass transported from zone 1 to zone 2 and $m_{tot}$ is the total mass in the CR. The combustion efficiency from the solution of the two-zone model is returned to the chemical equation and heat loss coefficient calculation until the error between runs is smaller than a given value, here chosen to be 0.01. The results of mass from zone 1 to zone 2 $m_{12}$, temperature in zone 1 $T_1$, volume in zone 1 $V_1$ and temperature in zone 2 $T_2$ for the reference experiment is seen in Fig. 3.12 to 3.15.
Right after the pre-combustion is finished, the two zones are assumed to be completely mixed. Having different composition and temperature, the resulting temperature $T_{mix}$ of this mixing is found with Eq. 2.36.

To get a better model for the heat loss during pre-combustion, the difference between $T_{mix}$ and the temperature found in the heat loss model, is calculated. This difference is then used to vary the value of Eichelbers coefficient $\epsilon$ til the temperature difference is smaller than 5 [K].

Minimizing the error between the temperatures from the two-zone model and the chemical equation, a discrete PI-controller is implemented. This is necessary since the temperature difference is quite large without any controller. The temperature difference with and without controller is shown in Fig. 3.16. It ends up within the prescribed 5 [K] after a few runs.

If the set precision of $\Delta T$, $\eta_{comb}$ or both of them not is reached after 25 runs, the algorithm stops and displays the calculated result. This is to prevent the calculation to
run for an infinite time without increasing the accuracy. This can happen if the solution of the two-zone model is too stiff, and the MATLAB solver fails to calculate the solution. The algorithm then calculates the resulting temperature based on a combustion efficiency of 95%. The failure in solving the two-zone model is seen by a warning in MATLAB and that the ROHR for the pre-combustion is impossible to calculate.

### 3.6 Rate of heat release

#### 3.6.1 Pre-combustion phase

The initial gas mixture in the CR can contain both $H_2$ and $CO$. Since it is outside the scope of this thesis to investigate differences in combustion speed of these two gases, they are assumed to combust at a rate so that 1% of the mass of $CO$ combusts at the same time as 1% of the mass of $H_2$. 

![Figure 3.12: Mass transported from zone 1 to zone 2 during pre-combustion as % of total mass](image-url)
Figure 3.13: Temperature in zone 1 during the pre-combustion

Figure 3.14: Volume in zone 1 during pre-combustion as % of total volume
Assuming that both gases combust with the relative same rate, make it possible to simplify and have one LHV for the mixture, given in Eq. 2.55. The ROHR for the pre-combustion phase is then calculated with both numerical differentiation of $m_{12}$ with the formula in App. C and smoothened with fourier fit. The need for fitting is due to that the calculated differentials for solving the equations are variables only inside the calculating function. A resulting plot is shown in Fig. 3.17.

3.7 Injection experiment

To estimate the value of heat release during injection of diesel, a closed system analysis have been performed. As a closed system, no mass is added, gas composition is constant and the pressure rise is due to heat interaction with the surroundings.

As seen in Eq. 2.64, the pressure difference is needed. For the few [ms] the combustion takes place, this is calculated using the formula in C. To get a smoother curve, an
The lower heating value (LHV) of the diesel is assumed to be constant over the entire experiment. Taking the integral of $\dot{Q}$ for the time of the injection experiment, give the total fuel energy released. Then the time a specific amount of the fuel take to burn can be found by an linear interpolation with:

$$t_p = t_{i-1} + (Q_{tot} \cdot p - Q_{i-1}) \cdot \frac{Q_i - Q_{i-1}}{t_i - t_{i-1}}$$  \hspace{1cm} (3.5)$$

where $t_p$ is the time when a given amount of fuel in [%] is burned, $Q_{tot}$ is the total heat released, $Q_{i-1}$ is the total heat released at time $t_{i-1}$ and $Q_i$ is the total heat released at time $t_i$. 

**Figure 3.16:** Effect of no controller, P-controller and PI-controller on $\Delta T$
3.8 Graphical User Interface (GUI)

With a goal to make the use of this MATLAB program easier, a graphical user interface (GUI) have been created using the built in "guide" function in MATLAB. The start up picture is seen in Fig. 3.19. The user need to input the name of the Microsoft Excel file where the measurements of interest is located. In the upper left corner mass fractions of the four gases in the pre-mixed combustable gas is to be typed in by the user. Below the panel for the gas mixture, volume fractions of the air is displayed. The air have a default composition of 79 % \( \text{N}_2 \) and 21 % \( \text{O}_2 \), but these two values can be changed by the user. Below the input of the file name, the calculated temperature at injection of diesel is displayed in degrees Celsius, together with the pressure at injection and the total mass of gas and air in the rig. At the bottom centre input for calculating time from injection to a given amount of the fuel is combusted. The default values are 2, 50 and 90 %. The time is then displayed in [ms] from injection started. On the top right the ROHR for the pre-combustion is displayed, and on the bottom right the ROHR for the injection.

![Figure 3.17: Plot of ROHR for the pre-combustion phase with numerical differentiation of the mass and with fitted curve.](image-url)
Figure 3.18: Plot of ROHR for the injection experiment with numerical differentiation of pressure and with smoothened curve

experiment is displayed. The final GUI is shown in Fig. 3.20. An output file containing the calculated values of interest is at default stored under the name "Output.txt", but this can be changed by the user.
Figure 3.19: GUI at startup
Figure 3.20: GUI after finished calculations
Chapter 4

Discussions regarding accuracy of calculations and future work to improve estimation

4.1 Assumptions

In this section the assumptions made to create the MATLAB algorithm is described. Their accuracy is discussed, and suggestions to improve the accuracy of the algorithm is included.

4.1.1 Perfect mixed gases

The gas in the CR is assumed to be perfectly mixed inside one control volume. The control volume (CV) could either be the entire rig, Fig. 2.2 or each of the two zones during the pre-combustion, as shown in Fig. 2.3. Compared to a reciprocating engine, the gas velocities inside the CR is very small. Due to the small gas velocities, the mixing of the gas after pre-combustion does not necessary be uniform.

This makes the gas mixture non-uniform and the temperature may vary through out the CV. To simulate this, a CFD-model of the gas flows could be made to increase the accuracy of the estimate of the gas temperature at injection of fuel.
4.1.2 Relative combustion rate of $H_2$ and $CO$

The combustable gases, $H_2$ and $CO$ is assumed to burn with the same relative speed. This is very simplified way to treat this combustion. As seen in Fig. 4.1, the laminar flame speed drop with increased amount of $CO$. $H_2$ will be combusted early in the pre-combustion, while the $CO$ takes some more time to be combusted.

This also indicates that the combustion efficacy $\eta_{comb}$ is unequal for both gases. To improve this model measurements of the gas before and after pre-combustion to clearly state the gas composition, can be performed.
4.2 Constant temperature of the rig \( T_{\text{wall}} \)

For calculating the temperature difference between the rig and the gas, needed in the convective law Eq. 2.37, the wall temperature in the rig is needed. This temperature is measured at the beginning of the experiment, and assumed to be constant during the entire experiment.

The energy released during the pre-combustion is in the range of 90 [kJ]. From Table 6.2 in Atkins and Jones (2008), the specific heat capacity \( c_{\text{steel}} \) for stainless steel is .51 \([\text{kJ/kgK}]\). For a mass of 100 [kg], the temperature rise will be:

\[
\Delta T = \frac{Q}{c_{\text{steel}} m} = \frac{90}{0.51 \cdot 100} = 1.77[K]
\]

(4.1)

This is about 0.1% of the maximum temperature of the gas, and neglecting this change during the experiment is reasonable.

4.3 Unaccuracies in measured pressure

Precision of the dynamic pressure measurements depends on several factors. The pressure sensor can have both a full scale and a relative accuracy, given % A high-temperature pressure sensor for combustion engine measurements from Kistler (Kistler, 2011) have an linearity smaller than 0.4 % of the full scale output (FSO). This means for a 50 [bar] sensor, the accuracy of the measured data in the entire range is ± 0.2 [bar]. Even such a small error resulting in an inaccuracy of the injection temperature \( T \) of:

\[
T = \frac{pV}{mR} = \pm 0.2 \cdot 10^5[Pa] \cdot 0.0049[m^3] = \pm 7.29[K]
\]

(4.2)

where the mass \( m \) and gas constant \( R \) is from the reference experiment.

High sampling frequency, 5000 [Hz], may introduce large errors when calculating the time derivative of the pressure, \( \dot{p} \). An small measurement error is escalating into a large
differential error, as seen in Fig. 3.11. The measured values seem like a smooth curve, but when trying to calculate the differential, this turns into a highly fluctuation curve.

Implementation of a low pass filter to exclude this high frequency measurements noise was tried, shown for three values of the filter time constant $\tau$. This smoothens the curve, but also reduce the peak value and introduce a phase shift from the original measurements. Since the both the combustion in the pre-combustion phase and the liquid fuel experiment takes place over such a short period of time, the phase shift made the filtering unapplicable.

MATLAB have a wide range of functions to fit against measured data. Both fourier functions and exponential functions have been used, since they fit the measured values in the range of interest very well and are easy to differentiate. This resulting in a smooth curve for the mass-transport from zone 1 til zone 2 in the pre-combustion and for the pressure drop during the cool down phase.

The numerical fit formula by O. Amble in App. C is used for the pressure rise in the diesel injection experiment. This is because the experimental time is very short, just a few [ms] and the pressure increase is very steep when the fuel ignites. To not lose information about exactly when this large pressure increase occurs and how large it is, the function is used. This result in a quite fluctuating ROHR, especially after the fuel is burned. ROHR function is then again smoothened with a integrated function in MATLAB, to increase readability and to come closer to the reference calculations. The ROHR for the reference experiment have been already been calculated, Fig. 4.2 and with the smoothening function these calculations is quite much closer than ROHR with the numerical fit, both plots shown in Fig. 3.18.

4.4 Precisiccion of the algorithm

The final algorithm performs control of two parameters, the $\eta_{comb}$ and $\Delta T$. These needs to stabilize before the calculations finish. In Fig. 4.3 the variation in temperature due to the different gas composition is shown for $\eta_{comb}$ from 80 % to 1 % combusted. Resulting values is shown in Tab. 4.1
Figure 4.2: ROHR from reference experiment plotted by Maximilian Malin, MARINTEK

Figure 4.3: Resulting temperature curves with varying $\eta_{comb}$
\begin{table}
\centering
\begin{tabular}{|c|c|}
\hline
\(\eta_{\text{comb}}\) [%] & T [K] \\
\hline
80 & 538.2 \\
90 & 543.5 \\
100 & 548.9 \\
\hline
\end{tabular}
\caption{Variation in combustion efficiency and resulting temperature}
\end{table}

Specifying a precision of the combustion efficiency to be less than 1 \%, give a precision in the temperature less than 1 [K]. The precision on the combustion efficiency can be made smaller, but this requires more calculation time.

The reduced \(\Delta T\) in Fig. 3.16 also influence \(\eta_{\text{comb}}\), as shown in Tab. 4.2.

\begin{table}
\centering
\begin{tabular}{|c|c|c|}
\hline
Controller type & \(\Delta T\) [K] & \(\eta_{\text{comb}}\) \\
\hline
No controller & -122.4 & 88.5 \\
P & -66.3 & 91.9 \\
PI & -0.2 & 96.8 \\
\hline
\end{tabular}
\caption{Variation in combustion efficiency and resulting temperature}
\end{table}

Combining the set precision of the combustion efficiency \(\eta_{\text{comb}}\) with a set precision of the temperature difference \(\Delta T\), increase the precision of the estimated temperature.

\section{4.5 Future work}

\subsection{4.5.1 Verification of gas mixture composition}

The gas mixture composition after pre-combustion can be verified by taking a gas sample from the CR and analyze its content. Doing this for several experiments, the two-zone model calculations can be verified and the model even corrected against analyze values.

\subsection{4.5.2 Heat losses}

Implementation of thermocouples to measure the instantaneous heat flux inside the CR can be performed to increase the heat loss model for the cool down phase. If the measurements are enable to change output value fast enough, the measured values can be implemented directly into the two-zone model and the rate of heat release during the injection phase. Solutions and implementation is described in Ch. 12.6 in \cite{Heywood1988}.
4.5.3 ROHR in injection experiment

Using an open system model for calculating the ROHR from the injection experiment, may increase the accuracy of these calculations. The quite small amount of fuel injected into the relatively large volume in the CR, is making the closed system analysis quite uncertain, since it does not include heat losses from the gas to the wall during this combustion.
Appendix A

Thermodynamic Coefficients from NASA GLENN DATABASE

<table>
<thead>
<tr>
<th></th>
<th>CO</th>
<th>CO₂</th>
<th>H₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$K^2$</td>
<td>1.489045326 $\cdot$ 10⁴</td>
<td>4.943650540 $\cdot$ 10⁴</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$K$</td>
<td>$-$2.922285939 $\cdot$ 10²</td>
<td>$-$6.264116010 $\cdot$ 10²</td>
</tr>
<tr>
<td>$a_3$</td>
<td>$-$</td>
<td>5.724527170</td>
<td>5.301725240</td>
</tr>
<tr>
<td>$a_4$</td>
<td>$K^{-1}$</td>
<td>$-$8.176235030 $\cdot$ 10⁻³</td>
<td>2.503813816 $\cdot$ 10⁻³</td>
</tr>
<tr>
<td>$a_5$</td>
<td>$K^{-2}$</td>
<td>1.456903469 $\cdot$ 10⁻⁵</td>
<td>$-$2.127308728 $\cdot$ 10⁻⁷</td>
</tr>
<tr>
<td>$a_6$</td>
<td>$K^{-3}$</td>
<td>$-$1.087746302 $\cdot$ 10⁻⁸</td>
<td>$-$7.689988780 $\cdot$ 10⁻¹⁰</td>
</tr>
<tr>
<td>$a_7$</td>
<td>$K^{-4}$</td>
<td>3.027941827 $\cdot$ 10⁻¹²</td>
<td>2.849677801 $\cdot$ 10⁻¹³</td>
</tr>
<tr>
<td>$b_1$</td>
<td>$K$</td>
<td>$-$1.303131878 $\cdot$ 10¹</td>
<td>$-$4.528198460 $\cdot$ 10⁴</td>
</tr>
</tbody>
</table>

Table A.1: Thermodynamic coefficients CO, CO₂ and H₂ in the range 200-1000 [K]
### Appendix A. Thermodynamic Coefficients

#### Table A.2: Thermodynamic coefficients $N_2$, $O_2$ and $H_2O$ in the range 200-1000 [K]

<table>
<thead>
<tr>
<th></th>
<th>$N_2$</th>
<th>$O_2$</th>
<th>$H_2O$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$ $[K^2]$</td>
<td>$2.210371497 \cdot 10^4$</td>
<td>$-3.425563420 \cdot 10^4$</td>
<td>$-3.947960830 \cdot 10^4$</td>
</tr>
<tr>
<td>$a_2$ $[K]$</td>
<td>$-3.818461820 \cdot 10^2$</td>
<td>$4.847000970 \cdot 10^2$</td>
<td>$5.755731020 \cdot 10^2$</td>
</tr>
<tr>
<td>$a_3$ $[-]$</td>
<td>$6.082738360$</td>
<td>$1.119010961$</td>
<td>$9.317826530 \cdot 10^{-1}$</td>
</tr>
<tr>
<td>$a_4$ $[K^{-1}]$</td>
<td>$-8.530914410 \cdot 10^{-3}$</td>
<td>$4.293889240 \cdot 10^{-3}$</td>
<td>$7.22712860 \cdot 10^{-3}$</td>
</tr>
<tr>
<td>$a_5$ $[K^{-2}]$</td>
<td>$1.384646189 \cdot 10^{-5}$</td>
<td>$-6.836300520 \cdot 10^{-7}$</td>
<td>$-7.342557370 \cdot 10^{-6}$</td>
</tr>
<tr>
<td>$a_6$ $[K^{-3}]$</td>
<td>$-9.625793620 \cdot 10^{-9}$</td>
<td>$-2.023372700 \cdot 10^{-9}$</td>
<td>$4.955043490 \cdot 10^{-9}$</td>
</tr>
<tr>
<td>$a_7$ $[K^{-4}]$</td>
<td>$2.519705890 \cdot 10^{-12}$</td>
<td>$1.03940018 \cdot 10^{-12}$</td>
<td>$-1.33693242 \cdot 10^{-12}$</td>
</tr>
<tr>
<td>$b_1$ $[K]$</td>
<td>$7.108460860 \cdot 10^2$</td>
<td>$-3.391454870 \cdot 10^3$</td>
<td>$-3.30974310 \cdot 10^4$</td>
</tr>
</tbody>
</table>

#### Table A.3: Thermodynamic coefficients $CO$, $CO_2$ and $H_2$ in the range 1000-6000 [K]

<table>
<thead>
<tr>
<th></th>
<th>$CO$</th>
<th>$CO_2$</th>
<th>$H_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$ $[K^2]$</td>
<td>$4.619197250 \cdot 10^5$</td>
<td>$1.176962419 \cdot 10^5$</td>
<td>$5.608128010 \cdot 10^5$</td>
</tr>
<tr>
<td>$a_2$ $[K]$</td>
<td>$-1.944704863 \cdot 10^3$</td>
<td>$-1.788791477 \cdot 10^3$</td>
<td>$-8.371504740 \cdot 10^2$</td>
</tr>
<tr>
<td>$a_3$ $[-]$</td>
<td>$5.916714180$</td>
<td>$8.291523190$</td>
<td>$2.975364532$</td>
</tr>
<tr>
<td>$a_4$ $[K^{-1}]$</td>
<td>$-5.664282830 \cdot 10^{-4}$</td>
<td>$-9.223156780 \cdot 10^{-5}$</td>
<td>$1.252249124 \cdot 10^{-3}$</td>
</tr>
<tr>
<td>$a_5$ $[K^{-2}]$</td>
<td>$1.398814540 \cdot 10^{-7}$</td>
<td>$4.863676880 \cdot 10^{-9}$</td>
<td>$-3.740716190 \cdot 10^{-7}$</td>
</tr>
<tr>
<td>$a_6$ $[K^{-3}]$</td>
<td>$-1.78760361 \cdot 10^{-11}$</td>
<td>$-1.891053312 \cdot 10^{-12}$</td>
<td>$5.93625200 \cdot 10^{-11}$</td>
</tr>
<tr>
<td>$a_7$ $[K^{-4}]$</td>
<td>$9.620935070 \cdot 10^{-16}$</td>
<td>$6.330035590 \cdot 10^{-16}$</td>
<td>$3.60699410 \cdot 10^{-15}$</td>
</tr>
<tr>
<td>$b_1$ $[K]$</td>
<td>$-2.466261084 \cdot 10^3$</td>
<td>$-3.908350590 \cdot 10^4$</td>
<td>$5.339824410 \cdot 10^3$</td>
</tr>
</tbody>
</table>

#### Table A.4: Thermodynamic coefficients $N_2$, $O_2$ and $H_2O$ in the range 1000-6000 [K]

<table>
<thead>
<tr>
<th></th>
<th>$N_2$</th>
<th>$O_2$</th>
<th>$H_2O$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$ $[K^2]$</td>
<td>$5.877124060 \cdot 10^5$</td>
<td>$-1.037939022 \cdot 10^6$</td>
<td>$1.034972096 \cdot 10^6$</td>
</tr>
<tr>
<td>$a_2$ $[K]$</td>
<td>$-2.239249073 \cdot 10^3$</td>
<td>$2.344830282 \cdot 10^3$</td>
<td>$-2.412698562 \cdot 10^3$</td>
</tr>
<tr>
<td>$a_3$ $[-]$</td>
<td>$6.066949220$</td>
<td>$1.819732036$</td>
<td>$4.646110780$</td>
</tr>
<tr>
<td>$a_4$ $[K^{-1}]$</td>
<td>$-6.139685500 \cdot 10^{-4}$</td>
<td>$1.267847582 \cdot 10^{-3}$</td>
<td>$2.291998307 \cdot 10^{-3}$</td>
</tr>
<tr>
<td>$a_5$ $[K^{-2}]$</td>
<td>$1.491806679 \cdot 10^{-7}$</td>
<td>$-2.188067988 \cdot 10^{-7}$</td>
<td>$-6.836830480 \cdot 10^{-7}$</td>
</tr>
<tr>
<td>$a_6$ $[K^{-3}]$</td>
<td>$-1.923105485 \cdot 10^{-11}$</td>
<td>$2.053719572 \cdot 10^{-11}$</td>
<td>$9.426468930 \cdot 10^{-11}$</td>
</tr>
<tr>
<td>$a_7$ $[K^{-4}]$</td>
<td>$1.061954386 \cdot 10^{-15}$</td>
<td>$-8.193467050 \cdot 10^{-16}$</td>
<td>$-4.822380530 \cdot 10^{-15}$</td>
</tr>
<tr>
<td>$b_1$ $[K]$</td>
<td>$1.283210415 \cdot 10^4$</td>
<td>$-1.689010929 \cdot 10^4$</td>
<td>$-1.384286509 \cdot 10^4$</td>
</tr>
</tbody>
</table>
Appendix B

Thermochemical Data

JANAF Thermochemical Tables Third Edition:

<table>
<thead>
<tr>
<th>Substance</th>
<th>Molar weight $[\text{g/mol}]$</th>
<th>Heat of formation $\Delta H_f^{0} [\text{kJ/mol}]$</th>
<th>Lower Heating Value $[\text{MJ/kg}]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$CO$</td>
<td>28.0104</td>
<td>-110.527</td>
<td>10.1032</td>
</tr>
<tr>
<td>$H_2$</td>
<td>2.01588</td>
<td>0</td>
<td>119.961</td>
</tr>
<tr>
<td>$CO_2$</td>
<td>44.0098</td>
<td>-393.522</td>
<td>-</td>
</tr>
<tr>
<td>$O_2$</td>
<td>31.9988</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$H_2O$</td>
<td>18.01528</td>
<td>-241.826</td>
<td>-</td>
</tr>
</tbody>
</table>

Table B.1: Thermochemical data for combustion gases. Ref: [Chase, 1986]
Appendix C

Numerical differential

Numerical differential formula for measured values with some uncertainty in the registered data points. The differential method includes some smoothening of the measured data.

\[
\left( \frac{dy}{dx} \right)_i = \frac{1}{h} \left[ \frac{2}{3} (y_{i+1} - y_{i-1}) - \frac{1}{12} (y_{i+2} - y_{i-2}) \right]
\]  \hspace{1cm} (C.1)

with

\[
h = x_{i+1} - x_i = \text{constant}
\]  \hspace{1cm} (C.2)

where \(y_{i-2}\) to \(y_{i+2}\) is the measured data points around point \(y_i\), where the differential is found. \(x_i\) is the free variable corresponding to \(y_i\).

This formula is taken from O. Amble, Numerical Methods II, lectures held at NTH fall 1966 - spring 1967.
Appendix D

Matlab scripts

D.1 GUI function and figure

Digital version handed in with the thesis

D.2 Function to initiate calculations

```matlab
% Run file to initiate all parameters and calculations

function Global = RUN(Global)

% Load measurements, calculate absolute pressure and fractions of each gas
% at start of experiment
[ Global ] = CONSTANTS(Global);

% Calculate resulting composition after pre-combustion with the initial
% combustion efficiency
[ Global ] = CHEMICAL_REACTION( Global );

% Calculate the heat loss coefficient based on gas constant after
% pre-combustion and pressure drop from pre-combustion ends to injection
% starts
[ Global ] = HEAT_LOSS_COEFFICIENT( Global );
```
% Solve the two-zone model
[ Global ] = TWO_ZONES( Global );

% If the values for combustion efficiency and/or temperature difference
% between the two-model solution and chemical reaction, the calculations
% are performed again with PI-controller to the temp-difference.
while ((abs((Global.eta_comb(Global.s+1)...
       -Global.eta_comb(Global.s)))>0.01) ||
       (abs(Global.T_diff(end))>5)) && Global.s<25

[ Global ] = CHEMICAL_REACTION( Global );

[ Global ] = HEATLOSS_COEFFICIENT( Global );

Global.s = Global.s+1;

[ Global ] = TWO_ZONES( Global );
end

% If while loop stops by maximum number of runs, calculate composition for
% eta_comb = 0.95 and display the resulting values
if Global.s == 25;

    Global.eta_comb = 0.95;
    Global.s = 1;

    [ Global ] = CONSTANTS(Global);

    [ Global ] = CHEMICAL_REACTION( Global );

    [ Global ] = HEATLOSS_COEFFICIENT( Global );

    [ Global ] = TWO_ZONES( Global );

end

% Rate of heat release calculated for pre-cobustion and injection
% experiment
[ Global ] = ROHR( Global );


% Print calculated values to text-file
OUTPUT( Global )

D.3 Import measurements and calculate gas composition based on user input

function [ Global ] = CONSTANTS(Global)

% Constants to use in Thermodynamic Analysis Tool (TDAT)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Gas constant from the NASA GLENN DATABASE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Global.R0 = 8.31451; % [J mol-1 K-1]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Molar weights for O2, N2, CO, H2, CO2, H2O from the NASA GLENN DATABASE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Global.MO2 = 31.9988000; % Molar weight O2 [g/mol]
Global.MN2 = 28.0134000; % Molar weight N2 [g/mol]
Global.MCO = 28.0101000; % Molar weight CO [g/mol]
Global.MH2 = 2.0158800; % Molar weight H2 [g/mol]
Global.MCO2 = 44.0095000; % Molar weight CO2 [g/mol]
Global.MH2O = 18.0152800; % Molar weight H2O [g/mol]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Gas constants for O2, N2, CO, H2, CO2, H2O from the NASA GLENN DATABASE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Global.RCO = Global.R0/Global.MCO*1e3; % Gas constant CO [J/kgK]
Global.RCO2 = Global.R0/Global.MCO2*1e3; % Gas constant CO2 [J/kgK]
Global.RH2 = Global.R0/Global.MH2*1e3; % Gas constant H2 [J/kgK]
Global.RO2 = Global.R0/Global.MO2*1e3; % Gas constant O2 [J/kgK]
Global.RH2O = Global.R0/Global.MH2O*1e3; % Gas constant H2O [J/kgK]
Global.RN2 = Global.R0/Global.MN2*1e3; % Gas constant N2 [J/kgK]
Appendix D. Matlab scripts

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

% Volume of combustion rig
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Radius of rig
Global.r = 0.125; % [m]
% Height of rig
Global.h = 0.1; % [m]

% Area of the combustion rig
Global.A = 2*pi*Global.r^2 + pi*2*Global.r*Global.h; % [m^2]
% Volume of combustion rig
Global.V = pi * Global.r^2 * Global.h; % [m^3]

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

% Initial conditions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Setting parameters for importing data from the excel-spreadsheets

file = Global.file_name;
input = 'Data_Input';

% Time relative to ignition pulse start
Global.time = xlsread(file,input,'A6:A50005'); % [s]

% Time between spark ignition and diesel injection
Global.injection_time = xlsread(file,input,'F2')/1e6; % [s]

% Temperature in combustion rig wall
Global.Tw = xlsread(file,input,'G6') + 273.15; % [K]

% Temperature in 1mm sensor converted to [K]
Global.T0 = xlsread(file,input,'C6') + 273.15;
% Static and gas charge pressure
Global.p0 = xlsread(file,input,'H6')*1e5; % [Pa]

% Air pressure set to 1 [bar]
Global.pair = 1*1e5; % [Pa]
% Static pressure sensor
p_stat = (xlsread(file,input,'H6:H50005')*1e5); % [Pa]

% Temperature in 1mm sensor
Global.T1mm = xlsread(file,input,'C6:C50005')+ 273.15; % [K]

% Temperature in 3mm sensor
Global.T3mm = xlsread(file,input,'B6:B50005') + 273.15; % [K]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dynamic pressure implementation and numeric differation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Dynamic pressure sensor 1
pdyn1 = xlsread(file,input,'K6:K50005')*1e5; % [Pa]
% Dynamic pressure sensor 2
pdyn2 = xlsread(file,input,'M6:M50005')*1e5; % [Pa]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Numerical differation from measured values of pdyn with uncertainties
% Method from O. Amble; "Numeriske metoder II", lectures fall 1966 and
% spring 1967
% \[ dy = 1/h \left( \frac{2}{3} (y_{i+1} - y_{i-1}) - \frac{1}{12} (y_{i+2} - y_{i-2}) \right) \] with \( h = x_{i+1} - x_i \) constant. Here it is the time between measurements
% This method also partly smoothens the measurements
% Differential from i-1 to 2 is lost, but assume \( p(2) = p(1) \)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initial difference between dynamic pressure sensor 1 and static pressure
% sensor
pdiff1 = Global.p0-pdyn1(1); % [Pa]
% Initial difference between dynamic pressure sensor 2 and static pressure
% sensor
pdiff2 = Global.p0-pdyn2(1); % [Pa]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initiating matrices to increase calculation speed
p1 = zeros(1,length(pdyn1));
p2 = zeros(1,length(pdyn2));

for m=1:length(pdyn1)
p1(m) = pdyn1(m) + pdiff1 + 1e5;

p2(m) = pdyn2(m) + pdiff2 + 1e5;
end

Global.p = p1; % [Pa]
Global.p_stat = p_stat + 1e5; % [Pa]
Global.p1 = p1; % [Pa]
Global.p2 = p2; % [Pa]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TIME FOR SIMULATION OF COOL DOWN PHASE FOR HEAT LOSS SIMULATIONS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Set the timespan for the calculations
[~, idx] = max(p1);
% Add 250*0.002 = 0.05 [s] to finish the pre-combustion
Global.combustion_end = idx + 250;
% Find the index for the injection start
Global.injection_start = find(Global.time==Global.injection_time);

% Finding the max value after injection
[~, id_max] = max(Global.p(Global.injection_start:end));

% Set finishing time for calculating ROHR from the injection experiment
Global.injection_end = id_max*2 + Global.injection_start;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Gas composition
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Gas constant gas mixture
Global.Rgas = Global.RCO * Global.xCOgas + Global.RN2 * Global.xN2gas ... 
+ Global.RO2 * Global.xO2gas; % [J/kgK]

% Molar mass of gas mixture
Global.Mgas = Global.R0/Global.Rgas*1e3; % [g/mol]

% Calculates volumfractions based on y_i = x_i * R0 / (M_i + sum[R_i*x_i])
Global.yCOgas = (Global.xCOgas * Global.R0)/(Global.MCO*... 
(Global.RCO*Global.xCOgas + Global.RN2*Global.xN2gas + ... 
Global.RO2*Global.xO2gas + Global.RH2*Global.xH2gas)) * 1e3;
Global.yH2gas = (Global.xH2gas * Global.R0)/(Global.MH2*...
(Global.RCO*Global.xCOgas + Global.RN2*Global.xN2gas + ... 
Global.RO2*Global.xO2gas + Global.RH2*Global.xH2gas))\times 1e3; 

Global.yO2gas = (Global.xO2gas \times Global.R0)/(Global.MO2*...
+ Global.RO2*Global.xO2gas + Global.RH2*Global.xH2gas))\times 1e3; 

Global.yN2gas = (Global.xN2gas \times Global.R0)/(Global.MN2*...
+ Global.RO2*Global.xO2gas + Global.RH2*Global.xH2gas))\times 1e3; 

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Air composition and properties
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Mass fraction O2 [-] 
Global.xO2air = Global.yO2air*Global.MO2/
(Global.yO2air*Global.MO2 + Global.yN2air*Global.MN2); 

% Mass fraction N2 [-] 
Global.xN2air = Global.yN2air*Global.MN2/
(Global.yO2air*Global.MO2 + Global.yN2air*Global.MN2); 

% Gas constant air 
Global.Rair = Global.RO2 * Global.xO2air ... 
+ Global.RN2 * Global.xN2air; % [J/kgK] 

% Molar mass air 
Global.Mair = Global.R0/Global.Rair*1e3; % [g/mol] 

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Masses in rig 
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Mass of each gas in air and mixture 
Global.mO2air = Global.V/Global.T0 * ((Global.pair*Global.xO2air)/... 
Global.Rair); % [kg] 

Global.mN2air = Global.V/Global.T0 * ((Global.pair*Global.xN2air)/... 
Global.Rair); % [kg] 

Global.mO2gas = Global.V/Global.T0 * ((Global.p0*Global.xO2gas)/... 
Global.Rgas); % [kg] 

Global.mN2gas = Global.V/Global.T0 * ((Global.p0*Global.xN2gas)/... 
Global.Rgas); % [kg] 

Global.mCOgas = Global.V/Global.T0 * ((Global.p0*Global.xCOgas)/...
Global.Rgas); % [kg]
Global.mH2gas = Global.V/Global.T0 * ((Global.p0*Global.xH2gas)/...
    Global.Rgas); % [kg]

% Mass of air and gas in rig
Global.mair = Global.mO2air + Global.mN2air; % [kg]
Global.mgas = Global.mO2gas + Global.mN2gas + Global.mCOgas; % [kg]

% Masses each component in gas mixture
Global.mO2mix = Global.mO2air + Global.mO2gas; % [kg]
Global.mN2mix = Global.mN2air + Global.mN2gas; % [kg]
Global.mCOmix = Global.mCOgas; % [kg]
Global.mH2mix = Global.mH2gas; % [kg]

% Mass fractions gas mix
Global.xO2mix = Global.mO2mix / (Global.mO2mix + Global.mN2mix ...
    + Global.mCOmix + Global.mH2mix);
Global.xN2mix = Global.mN2mix / (Global.mO2mix + Global.mN2mix ...
    + Global.mCOmix + Global.mH2mix);
Global.xCOmix = Global.mCOmix / (Global.mO2mix + Global.mN2mix ...
    + Global.mCOmix + Global.mH2mix);
Global.xH2mix = Global.mH2mix / (Global.mO2mix + Global.mN2mix + ...
    Global.mCOmix + Global.mH2mix);

% Gas constant gas mix
Global.Rmix = Global.RCO*Global.xCOmix + ...
    Global.RN2*Global.xN2mix + Global.RO2*Global.xO2mix ...
    + Global.RH2*Global.xH2mix; % [J/kgK]

% Molar mass gas mix
Global.Mmix = Global.R0/Global.Rmix*1e3; % [kg/kmol]

% Total mass in rig summing mass of air and mass of gas
Global.mrig = Global.mair + Global.mgas; % [kg]

% Total mass in rig for control summing each gas in the air and in the rig
Global.mrig_control = Global.mO2air + Global.mN2air + Global.mCOgas ...
    + Global.mN2gas + Global.mO2gas + Global.mH2gas; % [kg]
% Variable to count number of runtroughs
Global.s = 1;

% Initial set combustion efficiency
Global.eta_comb(1) = 0.95;

% Initial temperature difference between temperature calculated from the
% combustion efficiency and the Two-zone model solving
Global.T_diff(1) = 0;

% Initial value for gain in PI-control of k in Eichelbergs heat coefficient
% formula
Global.v(1) = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ROHR for pre-combustion phase
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Lower heating value CO from Heywood Ch. 3.5.2
Global.hn_CO = 10.1032*1e6; % [J/kg]

% Lower heating value H2 from Heywood Ch. 3.5.2
Global.hn_H2 = 119.961*1e6; % [J/kg]

% For the two zone model total mass transport dm = dm12
% ROHR = dQ/dt = h_n*dm_f = hn_CO*dm_CO + hn_H2*dmH2
% ROHR = (hn_CO*xCOmix + hn_H2*xH2mix)*dm12

% Lower heating value for mixture of H2 and CO,
% to be multiplied with total mass, m12
Global.hn = Global.hn_CO*Global.xCOmix...
    + Global.hn_H2*Global.xH2mix; % [J/kg]

% Fuel energy in CO
Global.Q_CO = Global.hn_CO*Global.mCOmix; % [J]

% Fuel energy in H2
Global.Q_H2 = Global.hn_H2*Global.mH2mix; % [J]
% Total fuel energy in the charged bomb
Global.Q_tot = Global.Q_CO + Global.Q_H2; % [J]
D.4 Gas composition after pre-combustion

function [ Global ] = CHEMICAL_REACTION( Global )
%CHEMICAL_REACTION Calculate composition after pre-combustion
% This function use the combustion efficency to calculate composition of
% the gas in the combustion rig after pre-combustion

%Combustion efficiency, first initial value and later runs based on the
two-zone calculations
eta_comb = Global.eta_comb(Global.s);

% Number of moles in the mix of gas and air
nO2 = Global.mO2mix/Global.MO2;
nCO = Global.mCOmix/Global.MCO;
nN2 = Global.mN2mix/Global.MN2;
nH2 = Global.mH2mix/Global.MH2;

% Reactants
Global.nO2r = nO2;
Global.nCOr = nCO;
Global.nN2r = nN2;
Global.nH2r = nH2;

% Molar fractions of the reactants
Global.yO2r = nO2/(nO2+nCO+nN2+nH2);
Global.yCOr = nCO/(nO2+nCO+nN2+nH2);
Global.yN2r = nN2/(nO2+nCO+nN2+nH2);
Global.yH2r = nH2/(nO2+nCO+nN2+nH2);

% Calculate the products
for i = 1:length(eta_comb)

% Number of moles
Global.nO2p(i) = nO2 - 0.5*eta_comb(i)*(nCO+nH2);
Global.nN2p = nN2;
Global.nCO2p(i) = eta_comb(i) * nCO;
Global.nH2Op(i) = eta_comb(i) * nH2;
Global.nH2p(i) = (1-eta_comb(i)) * nH2;
Global.nCOp(i) = (1-eta_comb(i)) * nCO;
end
Appendix D. Matlab scripts

% Mole fractions
Global.yO2p(i) = Global.nO2p(i) / (Global.nO2p(i) + Global.nN2p ...
  + Global.nCO2p(i) + Global.nCOp(i) + Global.nH2p(i) + Global.nH2Op(i));
Global.yN2p = Global.nN2p / (Global.nO2p(i) + Global.nN2p ...
  + Global.nCO2p(i) + Global.nCOp(i) + Global.nH2p(i) + Global.nH2Op(i));
Global.yCOp(i) = Global.nCOp(i) / (Global.nO2p(i) + Global.nN2p ...
  + Global.nCO2p(i) + Global.nCOp(i) + Global.nH2p(i) + Global.nH2Op(i));
Global.yCO2p(i) = Global.nCO2p(i) / (Global.nO2p(i) + Global.nN2p ...
  + Global.nCO2p(i) + Global.nCOp(i) + Global.nH2p(i) + Global.nH2Op(i));
Global.yH2p(i) = Global.nH2p(i) / (Global.nO2p(i) + Global.nN2p ...
  + Global.nCO2p(i) + Global.nCOp(i) + Global.nH2p(i) + Global.nH2Op(i));
Global.yH2Op(i) = Global.nH2Op(i) / (Global.nO2p(i) + Global.nN2p ...
  + Global.nCO2p(i) + Global.nCOp(i) + Global.nH2p(i) + Global.nH2Op(i));

% Mass fractions
Global.xO2p(i) = Global.yO2p(i)*Global.MO2/...
  (Global.yO2p(i)*Global.MO2...
  + Global.yN2p*Global.MN2 + Global.yCOp(i)*Global.MCO...
  + Global.yCO2p(i)*Global.MCO2 + Global.yH2p(i)*Global.MH2...
  + Global.yH2Op(i)*Global.MH2O);
Global.xN2p = Global.yN2p*Global.MN2/(Global.yO2p(i)*Global.MO2 + ...
  Global.yN2p*Global.MN2 + Global.yCOp(i)*Global.MCO + ...
  Global.yCO2p(i)*Global.MCO2 + Global.yH2p(i)*Global.MH2 +
  Global.yH2Op(i)*Global.MH2O);
Global.xCOp(i) = Global.yCOp(i)*Global.MCO/...
  (Global.yO2p(i)*Global.MO2 + Global.yN2p*Global.MN2...
  + Global.yCOp(i)*Global.MCO + Global.yCO2p(i)*Global.MCO2...
  + Global.yH2p(i)*Global.MH2 + Global.yH2Op(i)*Global.MH2O);
Global.xCO2p(i) = Global.yCO2p(i)*Global.MCO2/...
  (Global.yO2p(i)*Global.MO2 + Global.yN2p*Global.MN2...
  + Global.yCOp(i)*Global.MCO + Global.yCO2p(i)*Global.MCO2...
  + Global.yH2p(i)*Global.MH2 + Global.yH2Op(i)*Global.MH2O);
Global.xH2p(i) = Global.yH2p(i)*Global.MH2/...
  (Global.yO2p(i)*Global.MO2 + Global.yN2p*Global.MN2...
  + Global.yCOp(i)*Global.MCO + Global.yCO2p(i)*Global.MCO2...
  + Global.yH2p(i)*Global.MH2 + Global.yH2Op(i)*Global.MH2O);
Global.xH2Op(i) = Global.yH2Op(i)*Global.MH2O/...
  (Global.yO2p(i)*Global.MO2 + Global.yN2p*Global.MN2...
  + Global.yCOp(i)*Global.MCO + Global.yCO2p(i)*Global.MCO2...
  + Global.yH2p(i)*Global.MH2 + Global.yH2Op(i)*Global.MH2O);
% Resulting gas constant
Global.Rburned(i) = Global.RO2 .* Global.xO2p(i) + ...
Global.RN2 = Global.xN2p + ...
Global.RCO = Global.xCOp(i) + ...
Global.RCO2 = Global.xCO2p(i) ...
  + Global.RH2 = Global.xH2p(i) ...
  + Global.RH2O = Global.xH2Op(i);
end
end

D.5 Heat loss coefficient calcualtion

function [ Global ] = HEAT_LOSS_COEFFICIENT( Global )
%HEAT_LOSS_COEFFICIENT Calculates the heat loss coefficient
% This function calculates a first approximation of the heat loss in the
% CR based on Eichelbergs formula alpha=2.47(p*T)ˆ0.5 and using the gas
% compostion based on a fixed combustion efficency

% Area of wall in rig
A = Global.A; % [m^2]
% Volume of the CR
V = Global.V; % [m^3]
% Gas constant of burned gas
R = Global.Rburned; % [J/kgK]
% Total mass in rig
m = Global.mrig; % [kg]
% Temperature of combustion rig
T_w = Global.Tw; % [K]
% Pressure in the cool down phase
p = Global.p(Global.combustion_end:Global.injection_start); % [Pa]
% Time span for the cool down phase
cool_down_time = ...
  Global.time(Global.combustion_end:Global.injection_start); % [s]
% Counting variable
s = Global.s; % [-]
% Temperature difference between T12 and T(1) in cool down phase
T_diff = Global.T_diff; % [K]
% Volume of the CR
v = Global.v; % [m^3]

% Calculates the temperature in the zone assuming prefect mixed gas and
% ideal gas properties
% \( pV = mRT \)

% Creating initial matrixes to increase calculation speed
alpha_N = zeros(length(R),length(p));
dQ_N = zeros(length(R),length(p));
Cv = zeros(length(R),length(p));
kappa = zeros(length(R),length(p));

% Temperature calculated using the ideal gas law
temp = ((V.*p)/(m*R)); % [K]

% Time step
h = cool_down_time(2)-cool_down_time(1); % [s]

% Finding curve fit
f2 = fit(cool_down_time,p','exp2');

% Coefficents from the fit to the cool down curve with
% \( f2(x) = a*exp(b*x) + c*exp(d*x) \)
a2 = f2.a;
b2 = f2.b;
c2 = f2.c;
d2 = f2.d;

% Calculate the pressure fit for the cool down phase
dp_exp2 = a2*b2*exp(b2.*cool_down_time) + c2*d2*exp(d2.*cool_down_time);

% Using Newtons law of Convection to determine the heat loss
% \( dQdt = \alpha \cdot A \cdot (T_{\text{gas}}-T_{\text{wall}}) = m \cdot c_v \cdot dTdt \)
for i = 1:length(R)
    for j = 1:length(p)

        % Calculating the total heat capacity and kappa at the specified
        % temperature
        [ Cv(j), kappa(j) ] = HEAT_CAPACITY( Global, temp(i,j));
    end
end
% Ideal gas: \( R = c_p - c_v \), \( \frac{c_p}{c_v} = \kappa \) and \( \frac{dTdt}{dpdt} = \frac{V}{mR} \)

\[
dQ_N(j) = \frac{1}{(\kappa(j)-1)} \cdot V \cdot \text{dp}_\text{exp2}(j);
\]

% Solving for alpha
\[
\alpha_N(j) = \frac{dQ_N(j)}{(A \cdot (\text{temp}(i,j) - T_w))};
\]

end
end

% Fit the Eichelberg formula \( \alpha = k \cdot \sqrt{p \cdot T} \) to the alpha calculated
% by Newtons formula

% Guess of Eichelberg factor
\[
x_0 = -0.003;
\]
\[
xdata = \text{cool_down_time};
\]
\[
ydata = \alpha_N;
\]

% Create the anonymous function to calculate alpha with Eichelbergs formula
\[
\text{fun} = @(x,xdata)(x.*\text{sqrt}(\text{p}.*\text{temp}));
\]

% Turn off the option of printing the resulting fitting value
\[
\text{options} = \text{optimoptions}(@\text{lsqcurvefit},'\text{Display}','
\text{off}');
\]
\[
\text{lb} = [];
\]
\[
\text{ub} = [];
\]

% Coefficient to fit Eichelbergs formula against Newtons law of cooling
\[
\epsilon = \text{lsqcurvefit}(\text{fun},x_0,xdata,ydata,\text{lb},\text{ub},\text{options});
\]

% PI-controller for tuning Eichelbergs constant with difference in
% temperature in start of cool down phase based on temperatures calculated
% the two-zone model and from the comopstion after chemical reaction

\[
T = 1; \quad % Universal sampling time
\]
\[
T_i = 1; \quad % Integration constant
\]
\[
K_p = 2.5e-6; \quad % Proportional gain
\]
\[
e = T_{\text{diff}}; \quad % Error between set and measured value
\]

% Calculate the controller output to correct epsilon
\[
\text{if } s > 1
\]
\[
\quad \text{for } z = s:s
\]
\[
\quad \quad v(z) = v(z-1) + K_p \cdot (((1+T/(2 \cdot T_i)) \cdot e(z) - (1-T/(2 \cdot T_i)) \cdot e(z-1));
\]
\[
\text{end}
\]
\[
\text{end}
\]
% Store epsilon to calculate the heat loss from zone 2 when solving the % two-zone model
Global.epsilon = epsilon = v(s);

% Global.T_comb_end = temp(1);

% Global.cooldown_time = cool_down_time;

% Calculated values to be displayed in GUI
Global.p_injection = p(end);
Global.T_injection = temp(end);
Global.T_cooldown = temp(1);

% Set the correction from the controller to store in the Global struct to % be able to check it for later runs
Global.v = v;

end

D.6 Heat capacities calculated

function [ Cv, kappa ] = HEAT_CAPACITY( Global, T)
%HEAT_CAPACITY Calculate cv and kappa
% Uses the NASA GLENN DATABASE to calculate Cv and kappa for the gas after
% the pre-combustion based on the chemical equation

% Defines the gas constants for use in this function
RCO = Global.RCO/1000; % [kJ/kgK]
RCO2 = Global.RCO2/1000; % [kJ/kgK]
RH2 = Global.RH2/1000; % [kJ/kgK]
RN2 = Global.RN2/1000; % [kJ/kgK]
RO2 = Global.RO2/1000; % [kJ/kgK]
RH2O = Global.RH2O/1000; % [kJ/kgK]
% Calculates \( \frac{C_p^0(T)}{R} \) and \( \frac{H^0(T)}{RT} \) using NASA GLENN DATABASE polynomials

\[
\begin{align*}
[C_O, C_{O_2}, H_2, N_2, O_2, H_2O] &= \text{COEFFICIENTS}(T); \\
C_{pR, CO} &= (C_O(1) \cdot T^{-2} + C_O(2) \cdot T^{-1} + \ldots) \\
&\quad + (C_O(3) + C_O(4) \cdot T + C_O(5) \cdot T^2 + \ldots) \\
&\quad + (C_O(6) \cdot T^3 + C_O(7) \cdot T^4); \\
C_{pR, CO_2} &= (C_{O_2}(1) \cdot T^{-2} + C_{O_2}(2) \cdot T^{-1} + \ldots) \\
&\quad + (C_{O_2}(3) + C_{O_2}(4) \cdot T + C_{O_2}(5) \cdot T^2 + \ldots) \\
&\quad + (C_{O_2}(6) \cdot T^3 + C_{O_2}(7) \cdot T^4); \\
C_{pR, H_2} &= (H_2(1) \cdot T^{-2} + H_2(2) \cdot T^{-1} + \ldots) \\
&\quad + (H_2(3) + H_2(4) \cdot T + H_2(5) \cdot T^2 + \ldots) \\
&\quad + (H_2(6) \cdot T^3 + H_2(7) \cdot T^4); \\
C_{pR, N_2} &= (N_2(1) \cdot T^{-2} + N_2(2) \cdot T^{-1} + \ldots) \\
&\quad + (N_2(3) + N_2(4) \cdot T + N_2(5) \cdot T^2 + \ldots) \\
&\quad + (N_2(6) \cdot T^3 + N_2(7) \cdot T^4); \\
C_{pR, O_2} &= (O_2(1) \cdot T^{-2} + O_2(2) \cdot T^{-1} + \ldots) \\
&\quad + (O_2(3) + O_2(4) \cdot T + O_2(5) \cdot T^2 + \ldots) \\
&\quad + (O_2(6) \cdot T^3 + O_2(7) \cdot T^4); \\
C_{pR, H_2O} &= (H_2O(1) \cdot T^{-2} + H_2O(2) \cdot T^{-1} + \ldots) \\
&\quad + (H_2O(3) + H_2O(4) \cdot T + H_2O(5) \cdot T^2 + \ldots) \\
&\quad + (H_2O(6) \cdot T^3 + H_2O(7) \cdot T^4); \\
\end{align*}
\]

% Calculates \( C_p(T) \) [J/kgK] with \( C_p(T) = \frac{C_p^0(T)}{R} \cdot R \) for each gas

\[
\begin{align*}
C_p_{CO} &= C_{pR, CO} \cdot R_{CO}; \\
C_p_{CO_2} &= C_{pR, CO_2} \cdot R_{CO_2}; \\
C_p_{H_2} &= C_{pR, H_2} \cdot R_{H_2}; \\
C_p_{N_2} &= C_{pR, N_2} \cdot R_{N_2}; \\
C_p_{O_2} &= C_{pR, O_2} \cdot R_{O_2}; \\
C_p_{H_2O} &= C_{pR, H_2O} \cdot R_{H_2O};
\end{align*}
\]

% Calculates \( C_v(T) \) [J/kgK] with \( C_v(T) = C_p(T) - R \) for each gas

\[
\begin{align*}
C_v_{CO} &= C_p_{CO} - R_{CO}; \\
C_v_{CO_2} &= C_p_{CO_2} - R_{CO_2}; \\
C_v_{H_2} &= C_p_{H_2} - R_{H_2};
\end{align*}
\]
Appendix D. Matlab scripts

\[ \begin{align*}
C_{v,N2} &= C_{p,N2} - R_{N2}; \\
C_{v,O2} &= C_{p,O2} - R_{O2}; \\
C_{v,H2O} &= C_{p,H2O} - R_{H2O}; \\
C_v &= C_{v,CO} \cdot \text{Global.xCOp} + C_{v,CO2} \cdot \text{Global.xCO2p} \\
&\quad + C_{v,H2} \cdot \text{Global.xH2p} + C_{v,N2} \cdot \text{Global.xN2p} \\
&\quad + C_{v,O2} \cdot \text{Global.xO2p} + C_{v,H2O} \cdot \text{Global.xH2Op}; \\
\end{align*} \]

% Calculates kappa [-] with kappa = Cp/Cv for each gas
\[
\begin{align*}
kappa_{CO} &= C_{p,CO} / C_{v,CO}; \\
kappa_{CO2} &= C_{p,CO2} / C_{v,CO2}; \\
kappa_{H2} &= C_{p,H2} / C_{v,H2}; \\
kappa_{N2} &= C_{p,N2} / C_{v,N2}; \\
kappa_{O2} &= C_{p,O2} / C_{v,O2}; \\
kappa_{H2O} &= C_{p,H2O} / C_{v,H2O}; \\
kappa &= \kappa_{CO} \cdot \text{Global.xCOp} + \kappa_{CO2} \cdot \text{Global.xCO2p} \\
&\quad + \kappa_{H2} \cdot \text{Global.xH2p} + \kappa_{N2} \cdot \text{Global.xN2p} \\
&\quad + \kappa_{O2} \cdot \text{Global.xO2p} + \kappa_{H2O} \cdot \text{Global.xH2Op}; \\
\end{align*} \]

D.7 NASA GLENN COEFFICIENTS

\[
\text{function } [\text{CO}, \text{CO2}, \text{H2}, \text{N2}, \text{O2}, \text{H2O}] = \text{COEFFICIENTS(T)}
\]

% Coefficients a1 to a7 divided into temperature range from
% T = 200 to 1000 K and from 1000 to 6000 [K]
\[
\begin{align*}
\text{if } T &> 200 \\
\text{if } T &< 1000 \\
\text{CO} &= [1.489045326e04, -2.922285939e02, 5.724527170e+00, ..., \\
&\quad -8.176235030e-03, 1.456903469e-05, -1.087746302e-08, ..., \\
&\quad 3.027941827e-12, -1.303131878e+04, -7.859241350e+00]; \\
\text{CO2} &= [4.943650540e+04, -6.264116010e+02, 5.301725240e+00, ..., \\
&\quad 2.503813816e-03, -2.127308728e-07, -7.689988780e-10, ..., \\
&\quad 2.849677801e-13, -4.528198460e+04, -7.048279944e+00]; \\
\text{H2} &= [4.078323210e+04, -8.009186040e+02, 8.214702010e+00, ..., \\
&\quad -1.26971457e-02, 1.753605076e-05, -1.202860270e-08, ..., \\
&\quad 3.368093490e-12, 2.682484665e+03, -3.043788844e+01];
\end{align*} \]
Appendix D. Matlab scripts

N2 = [2.210371497e+04, -3.818461820e+02, 6.082738360e+00, ... 
     -8.53091410e-03, 1.384646189e-05, -9.625793620e-09, ... 
     2.519705809e-12, 7.108460860e+02, -1.076003744e+01];
O2 = [-3.425563420e+04, 4.847000970e+02, 1.19010961e+00, ... 
     4.29389240e-03, -6.836300520e-07, -2.023372700e-09, ... 
     1.03904018e-12, -3.39145870e+03, 1.849699470e+01];
H2O = [-3.947960830e+04, 5.755731020e+00, ... 
     10839303246e-12, -3.309332410e+04, 1.724205775e+01];

elseif ((T > 1000) && (T < 6000))
    CO = [4.619197250e+05, -1.944704863e+03, 5.916714180, ... 
     -5.64282830e-04, 1.398814540e-07, -1.878680361e-11, ... 
     9.62093570e-16, -2.466261084e+03, -1.387413108e+01];
    CO2 = [1.176962419e+05, -1.788791477e+03, 8.291523190e+00, ... 
     -9.223156780e-05, 4.863676880e-09, -1.891053312e-12, ... 
     6.330036590e-16, -3.908350590e+04, -2.652669281e+01];
    H2 = [ 5.608128010e+05, -8.371504740e+02, 2.975364532e+00, ... 
     -1.25224912e-03, -3.704716190e-07, -2.063625200e-11, ... 
     9.606994100e-15, 5.339824410e+03, -2.20774769e+00];
    N2 = [5.877124060e+05, -2.39249073e+03, 6.06949220e+00, ... 
     -6.139685500e-04, 1.491806679e-07, -1.923105485e-11, ... 
     1.061954386e-15, 1.283210415e+04, -1.586640027e+01];
    O2 = [-1.037939022e+06, 2.344830282e+03, 1.819732036e+00, ... 
     1.267847582e-03, -2.188067988e-07, 2.053719572e+11, ... 
     -8.193467050e-16, -1.689019299e+04, 1.738716506e+01];
    H2O = [ 1.034972096e+06, -2.412968562e+03, 4.646110780e+00, ... 
     2.291998307e-03, -6.836830480e-07, 9.426468930e-11, ... 
     -4.822308530e-15, -1.384286509e+04, -7.978148510e+00];
else
    % Display a warning if the temperature exceeds 6000 [K]
    disp('Temperature to high')
end
else
    % Display a warning if the temperature is below 200[K]
    disp('Temperature to low')
end

D.8 Two-zone model execution
function [ Global ] = TWO_ZONES( Global )

% TWO_ZONES Function for solving the two-zone model in the pre-combustion phase

% Time in pre-combustion phase
time = Global.time(1:Global.combustion_end+1); % \[s\]

% Pressure in pre-combustion phase
p = Global.p(1:Global.combustion_end+1); % \[Pa\]

% Volume of combustion rig
V = Global.V; % \[m^3\]

% Mass in rig
mrig = Global.mrig; % [kg]

% Initial temperature in rig when gas combustion starts
T0 = Global.T0; % [K]

% Initial part of total mass and volume in zone 2
initial_part_zone2 = 0.03; % [-]

% Initial volume of zone 1
V1_0 = V; % \[m^3\]

% Counter variable for number of runs
s = Global.s; % [-]

% Temperature at beginning of cool down phase
T_cooldown = Global.T_cooldown; % [K]

% Make curve fit for making a smooth differential

% Make a fit for the pressure curve using the General model Fourier8:

% \[ f(time) = a_0 + a_1 \cos(w_\cdot \text{time}) + b_1 \sin(w_\cdot \text{time}) + a_2 \cos(2\cdot w_\cdot \text{time}) \ldots \]
% + b_2 \sin(2\cdot w_\cdot \text{time}) + a_3 \cos(3\cdot w_\cdot \text{time}) + b_3 \sin(3\cdot w_\cdot \text{time}) \ldots \]
% + a_4 \cos(4\cdot w_\cdot \text{time}) + b_4 \sin(4\cdot w_\cdot \text{time}) + a_5 \cos(5\cdot w_\cdot \text{time}) \ldots \]
% + b_5 \sin(5\cdot w_\cdot \text{time}) + a_6 \cos(6\cdot w_\cdot \text{time}) + b_6 \sin(6\cdot w_\cdot \text{time}) \ldots \]
% + a_7 \cos(7\cdot w_\cdot \text{time}) + b_7 \sin(7\cdot w_\cdot \text{time}) + a_8 \cos(8\cdot w_\cdot \text{time}) \ldots \]
% + b_8 \sin(8\cdot w_\cdot \text{time}) \]

p_fit = fit(time,p,'fourier8');

a0 = p_fit.a0;
a1 = p_fit.a1;
a2 = p_fit.a2;
a3 = p_fit.a3;
a4 = p_fit.a4;
a5 = p_fit.a5;
a6 = p_fit.a6;
a7 = p_fit.a7;
a8 = p_fit.a8;
b1 = p_fit.b1;
b2 = p_fit.b2;
b3 = p_fit.b3;
b4 = p_fit.b4;
b5 = p_fit.b5;
b6 = p_fit.b6;
b7 = p_fit.b7;
b8 = p_fit.b8;
w = p_fit.w;

% Calculate differential of pressure from fitted function
dp_fit = w*(-a1*sin(w.*time) - 2*a2*sin(2*w.*time) - 3*a3*sin(3*w.*time)...
- 4*a4*sin(4*w.*time) - 5*a5*sin(5*w.*time) - 6*a6*sin(6*w.*time)...
- 7*a7*sin(7*w.*time) - 8*a8*sin(8*w.*time)...
+ b1*cos(w.*time) + 2*b2*cos(2*w.*time) + 3*b3*cos(3*w.*time)...
+ 4*b4*cos(4*w.*time) + 5*b5*cos(5*w.*time) + 6*b6*cos(6*w.*time)...
+ 7*b7*cos(7*w.*time) + 8*b8*cos(8*w.*time));

% Calculate pressure from fitted function
p_fit_calc = a0 + a1*cos(w.*time) + b1*sin(w.*time) + a2*cos(2*w.*time)...
+ b2*sin(2*w.*time)+ a3*cos(3*w.*time) + b3*sin(3*w.*time)...
+ a4*cos(4*w.*time) + b4*sin(4*w.*time) + a5*cos(5*w.*time)...
+ b5*sin(5*w.*time) + a6*cos(6*w.*time) + b6*sin(6*w.*time)...
+ a7*cos(7*w.*time) + b7*sin(7*w.*time) + a8*cos(8*w.*time)...
+ b8*sin(8*w.*time);

% Initial conditions where x0(1) = m12, x0(2) = T1, x0(3) = V1, x0(4) = T2
x0 = [0 T0 V1 0 T0];

% Set timespan for the solving
timespan = [0 time(Global.combustion_end)];
% Set integration tolerances
options=odeset('RelTol',1e-6,'AbsTol',1e-6*ones(1,4));

% Solve the differential equations
[T,X] = ode15s(@ODEFUN,timespan,x0,options,...
    Global, p_fit_calc, dp_fit, initial_part_zone2);

% Finding resulting temperature in rig after pre-combustion
T1 = X(end,2);
T2 = X(end,4);
m2 = X(end,1);
m1 = mrig - m2;

% Check if temperature is above the minimum limit by coefficients in NASA
% GLENN DATABASE
% The resulting value may be too low when the solver fails early and the
% value of T2 drops
if T2 >= 200
% Set up composition of each zone
[ zone1, zone2 ] = ZONES_DATA( Global );

% Calculate enthalpy in zone 1
[ ~, ~, ~, h ] = ZONES_THERMO_PROPERTIES( T1, Global, zone1 );
h1 = h.zone; % [J/kg]

% Calculate enthalpy in zone 2
[ ~, ~, ~, h ] = ZONES_THERMO_PROPERTIES( T2, Global, zone2 );
h2 = h.zone; % [J/kg]

% Calculate the enthalphy of the CR after pre-combustion,
% both zones completely mixed, based on resulting temperature in zone 2
[ h, temp ] = ENTALPHY(T2, Global);

% Solve the equation: m1*h1 + m2*h2 = (m1+m2)*h for h12
h12 = (m1*h1 + m2*h2)/(m1+m2); % [kJ/kg]

% Find the resulting temperature
ind = find(h12-h < 10, 1, 'first');
T12 = temp(ind); % [K]

% Calculate the resulting difference
Global.T_diff(s) = T12 - T_cooldown; % [K]

% Resulting combustion efficiency
Global.eta_comb(s+1) = max(X(:,1))/mrig;

else
    Global.eta_comb(s+1) = Global.eta_comb(s)-0.001;
    Global.T_diff(s) = 0;
end

% Set final variables to display them in the GUI
Global.ml2 = X(:,1); % [kg]
Global.T1 = X(:,2); % [K]
Global.V1 = X(:,3); % [m^3]
Global.T2 = X(:,4); % [K]
Global.time_two_zones = T; % [s]
end

D.9 Function to update variables in two-zone model

function dx = ODEFUN(t, x, Global, p, dp, initial_part_zone2)
% ODEFUN Calculate and update the state matrix for the two-zone model
% This function take the solved states from ode15s to update the A matrix
% used to calculate dotx = A^-1 * x

% Time in calculation
time = Global.time; % [s]
% Volume of rig
V = Global.V; % [m^3]
% Area of rig
A = Global.A; % [m^2]
% Rig wall temperature
Tw = Global.Tw; % [K]
% Mass in rig
mrig = Global.mrig; % [kg]
% Time step between measurements
h = time(2)-time(1); % [s]
% Coefficient in Eichelbergs formula
epsilon = Global.epsilon;

% Find the pressure corresponding to the simulation time
i = find((time-t)<=h,1,'first');
psolver = p(i); % [Pa]
dpsolver = dp(i); % [Pa]
% Initial mass in zone two, must be unequal to zero
m2_0 = mrig*initial_part_zone2; % [kg]
% Initial mass in rig, all in zone one
m1_0 = mrig - m2_0; % [kg]
% Initial volume zone 2
V2_0 = V*initial_part_zone2; % [m^3]

% Mass in zone decrease with m12
m1 = m1_0 - x(1); % [kg]
% Mass in zone increase with m12
m2 = m2_0 + x(1); % [kg]
% Temperature in zone 1
T1 = x(2); % [K]
% Volume in zone 1
V1 = x(3); % [m^3]
% Temperature in zone 2
T2 = x(4); % [K]
% Volume in zone 2 V = V1+V2
V2 = V2_0 + V-V1; % [m^3]

% Calculation of composition in zone 1 and zone 2
[ zone1, zone2 ] = ZONES_DATA( Global );

% Check if the temperature in zone 1 is above required 200 K, which can
% occur in the initiating phase of the differential solution
if T1>200
  % Calculate the thermodynamic properties in zone 1
  [ ~, Cv, ~, h, u ] = ZONES_THERMO_PROPERTIES( T1, Global, zone1 );
  h1 = h.zone; % [J/kg]
  u1 = u.zone; % [J/kg]
  Cv1 = Cv.zone; % [J/kgK]
else
  % If the temperature is to low, set to the minimum value to continue
  % calculations
  T1 = 200;
  % Calculate the thermodynamic properties in zone 1
  [ ~, Cv, ~, h, u ] = ZONES_THERMO_PROPERTIES( T1, Global, zone1 );
  h1 = h.zone; % [J/kg]
  u1 = u.zone; % [J/kg]
  Cv1 = Cv.zone; % [J/kgK]
end

% Check if the temperature in zone 2 is above required 200 K, which can
% occur in the initiating phase of the differential solution
if T2>200
    % Calculate the thermodynamic properties in zone 2
    [~, Cv, ~, ~, u ] = ZONES_THERMO_PROPERTIES( T2, Global, zone2 );
    u2 = u.zone; % [J/kg]
    Cv2 = Cv.zone; % [J/kgK]
else
    % If the temperature is to low, set to the minimum value to continue
    % calculations
    T2 = 200;
    % Calculate the thermodynamic properties in zone 2
    [~, Cv, ~, ~, u ] = ZONES_THERMO_PROPERTIES( T2, Global, zone2 );
    u2 = u.zone; % [J/kg]
    Cv2 = Cv.zone; % [J/kgK]
end

% Heat losses in zone 1 and zone 2 with \( \dot{Q} = \alpha A (T - Tw) \)

% Heat loss from zone 1
alpha_1 = epsilon * sqrt(psolver*T1);
% dQ1 = 0 because the gas composition used to calculate epsilon is assumed
% to be more similar to zone 2
dQ1 = 0; %alpha_1 * A * (T1-Tw) * V1/V;

% Heat loss from zone 2
alpha_2 = epsilon * sqrt(psolver*T2);
dQ2 = alpha_2 * A * (T2-Tw);

% Matrixes for solving the two zone model \( A \cdot \Delta x / \Delta t = b \)
A = [1/m1, 1/T1, 1/V1, 0; ... 
    (h1-u1), m1*Cv1, psolver, 0; ... 
    1/m2, 0 , 1/V2, 1/T2;...
    (h1-u2), 0, psolver, -m2*Cv2];

b = [-dpsolver/psolver; dQ1 ; dpsolver/psolver; -dQ2 ];

% Return changes in the states
dx = A\b;
D.10  Function to calculate the mass fraction in zone 1 and zone 2

```matlab
% Function to calculate the mass fraction in zone 1 and zone 2
function [ zone1, zone2 ] = ZONES_DATA( Global )

% ZONES_DATA Calculates the mass fraction of each gas
% This function calculates the mass fraction of each gas in zone 1 and zone 2 to determine the thermodynamic state of the entire zone based on the thermodynamic state of each substance

nO2 = (Global.mrig * Global.xO2mix)/Global.MO2;
nCO = (Global.mrig * Global.xCOmix)/Global.MCO;
nN2 = (Global.mrig * Global.xN2mix)/Global.MN2;
nH2 = (Global.mrig * Global.xH2mix)/Global.MH2;

% Composition of zone 1 expressed in mass fractions
zone1.xCO = Global.xCOmix;
zone1.xN2 = Global.xN2mix;
zone1.xO2 = Global.xO2mix;
zone1.xH2 = Global.xH2mix;
zone1.xCO2 = 0;
zone1.xH2O = 0;

% Composition of zone 2 expressed in mass fractions
zone2.nO2 = nO2-0.5*(nCO+nH2);
zone2.nN2 = nN2;
zone2.nCO2 = nCO;
zone2.nH2O = nH2;
zone2.yO2 = zone2.nO2/(zone2.nO2+zone2.nN2+zone2.nCO2+zone2.nH2O);
zone2.yN2 = zone2.nN2/(zone2.nO2+zone2.nN2+zone2.nCO2+zone2.nH2O);
zone2.yCO2 = zone2.nCO2/(zone2.nO2+zone2.nN2+zone2.nCO2+zone2.nH2O);
zone2.yH2O = zone2.nH2O/(zone2.nO2+zone2.nN2+zone2.nCO2+zone2.nH2O);
zone2.xO2 = zone2.yO2*Global.MO2/(zone2.yO2*Global.MO2 + ...
zone2.yN2*Global.MN2 + zone2.yCO2*Global.MCO2);
zone2.xN2 = zone2.yN2*Global.MN2/(zone2.yO2*Global.MO2 + ...
zone2.yN2*Global.MN2 + zone2.yCO2*Global.MCO2);
```

zone2.xCO2 = zone2.yCO2*Global.MCO2/(zone2.yO2*Global.MO2 + ...
    zone2.yN2*Global.MN2 + zone2.yCO2*Global.MCO2);
zone2.xCO = 0;
zone2.xH2 = 0;
zone2.xH2O = zone2.yH2O*Global.MH2O/(zone2.yO2*Global.MO2 + ...
    zone2.yN2*Global.MN2 + zone2.yCO2*Global.MCO2);
end

D.11 Function to calculate the thermodynamic properties
of zone 1 and zone 2

function [ Cp, Cv, kappa, h, u ] = ZONES_THERMO_PROPERTIES( ...
    temp, Global, zone )
%ZONES_THERMO_PROPERTIES Calculates the thermodynamic properties of the
%combustion gases for both zone 1 and zone 2

% Set matrices for increased calculation speed
CpR_CO = zeros(1,length(temp));
CpR_CO2 = zeros(1,length(temp));
CpR_H2 = zeros(1,length(temp));
CpR_N2 = zeros(1,length(temp));
CpR_O2 = zeros(1,length(temp));
CpR_H2O = zeros(1,length(temp));

h.CO = zeros(1,length(temp));
h.CO2 = zeros(1,length(temp));
h.H2 = zeros(1,length(temp));
h.N2 = zeros(1,length(temp));
h.O2 = zeros(1,length(temp));
h.H2O = zeros(1,length(temp));

u.CO = zeros(1,length(temp));
u.CO2 = zeros(1,length(temp));
u.H2 = zeros(1,length(temp));
u.N2 = zeros(1,length(temp));
u.O2 = zeros(1,length(temp));
u.H2O = zeros(1,length(temp));
% Defines the gas constants for use in this function
RCO = Global.RCO; % [J/kgK]
RCO2 = Global.RCO2; % [J/kgK]
RH2 = Global.RH2; % [J/kgK]
RN2 = Global.RN2; % [J/kgK]
RO2 = Global.RO2; % [J/kgK]
RH2O = Global.RH2O; % [J/kgK]

% Calculates Cp'0(T)/R and H'0(T)/RT using NASA GLENN DATABASE polynom
for i = 1:length(temp)

    % Import coefficients for the calculations
    [CO, CO2, H2, N2, O2, H2O] = COEFFICIENTS(temp(i));

    CpR_CO(i) = (CO(1)*temp(i)^-2 + CO(2)*temp(i)^-1 + ... 
                    CO(3) + CO(4)*temp(i) + CO(5)*temp(i)^2 + ... 
                    CO(6)*temp(i)^3 + CO(7)*temp(i)^4);

    CpR_CO2(i) = (CO2(1)*temp(i)^-2 + CO2(2)*temp(i)^-1 + ... 
                   CO2(3) + CO2(4)*temp(i) + CO2(5)*temp(i)^2 + ... 
                   CO2(6)*temp(i)^3 + CO2(7)*temp(i)^4);

    CpR_H2(i) = (H2(1)*temp(i)^-2 + H2(2)*temp(i)^-1 + ... 
                 H2(3) + H2(4)*temp(i) + H2(5)*temp(i)^2 + ... 
                 H2(6)*temp(i)^3 + H2(7)*temp(i)^4);

    CpR_N2(i) = (N2(1) * temp(i)^-2 + N2(2) * temp(i)^-1 + ... 
                  N2(3) + N2(4)*temp(i) + N2(5)*temp(i)^2 + ... 
                  N2(6)*temp(i)^3 + N2(7)*temp(i)^4);

    CpR_O2(i) = (O2(1) * temp(i)^-2 + O2(2) * temp(i)^-1 + ... 
                 O2(3) + O2(4)*temp(i) + O2(5)*temp(i)^2 + ... 
                 O2(6)*temp(i)^3 + O2(7)*temp(i)^4);

    CpR_H2O(i) = (H2O(1)*temp(i)^-2 + H2O(2)*temp(i)^-1 + ... 
                 H2O(3) + H2O(4)*temp(i) + H2O(5)*temp(i)^2 + ... 
                 H2O(6)*temp(i)^3 + H2O(7)*temp(i)^4);

    % Calculates h(T) [J/kg] with h(T) = [H'0(T)/RT]*RT for each gas
h.CO(i) = (-CO(1)*temp(i)^-2 + CO(2)*(log(temp(i))/temp(i)) ... 
   + CO(3) + CO(4)*temp(i)/2 + CO(5)*((temp(i)^2)/3) ... 
   + CO(6)*((temp(i)^3)/4 + CO(7)*((temp(i)^4)/5) ... 
   + CO(8)/temp(i)) * RCO*temp(i);

h.CO2(i) = (-CO2(1)*temp(i)^-2 + CO2(2)*(log(temp(i))/temp(i)) + ... 
   CO2(3) + CO2(4)*temp(i)/2 + CO2(5)*((temp(i)^2)/3) + ... 
   CO2(6)*((temp(i)^3)/4 + CO2(7)*((temp(i)^4)/5) + ... 
   CO2(8)/temp(i)) * RCO2*temp(i);

h.H2(i) = (-H2(1)*temp(i)^-2 + H2(2)*temp(i)/2 + H2(3) + H2(4)*temp(i)/2 + H2(5)*((temp(i)^2)/3) + ... 
   H2(6)*((temp(i)^3)/4 + H2(7)*((temp(i)^4)/5) + ... 
   H2(8)/temp(i)) * RH2*temp(i);

h.N2(i) = (-N2(1)*temp(i)^-2 + N2(2)*temp(i)/2 + N2(3) + N2(4)*temp(i)/2 + N2(5)*((temp(i)^2)/3) + ... 
   N2(6)*((temp(i)^3)/4 + N2(7)*((temp(i)^4)/5) + ... 
   N2(8)/temp(i)) * RN2*temp(i);

h.O2(i) = (-O2(1)*temp(i)^-2 + O2(2)*temp(i)/2 + O2(3) + O2(4)*temp(i)/2 + O2(5)*((temp(i)^2)/3) + ... 
   O2(6)*((temp(i)^3)/4 + O2(7)*((temp(i)^4)/5) + ... 
   O2(8)/temp(i)) * RO2*temp(i);

h.H2O(i) = (-H2O(1)*temp(i)^-2 + H2O(2)*temp(i)/2 + H2O(3) + H2O(4)*temp(i)/2 + H2O(5)*((temp(i)^2)/3) + ... 
   H2O(6)*((temp(i)^3)/4 + H2O(7)*((temp(i)^4)/5) + ... 
   H2O(8)/temp(i)) * RH2O*temp(i);

% Calculates h(T) for the specific zone in the two zone model
h.zone = h.CO*zone.xCO + h.CO2*zone.xCO2 + h.H2*zone.xH2 + ... 
   h.N2*zone.xN2 + h.O2*zone.xO2 + h.H2O*zone.xH2O;

% Calculates u(T) [J/kg] with u(T) = h(T) - RT for each gas
u.CO(i) = h.CO(i) - RCO*temp(i);

u.CO2(i) = h.CO2(i) - RCO2*temp(i);

u.H2(i) = h.H2(i) - RH2*temp(i);

u.N2(i) = h.N2(i) - RN2*temp(i);

u.O2(i) = h.O2(i) - RO2*temp(i);

u.H2O(i) = h.H2O(i) - RH2O*temp(i);
% Calculates $u(T)$ for the specific zone in the two zone model
\[
\text{u.zone} = \text{u.CO} \times \text{zone.xCO} + \text{u.CO2} \times \text{zone.xCO2} + \text{u.H2} \times \text{zone.xH2} + \ldots \]
\[
\text{u.N2} \times \text{zone.xN2} + \text{u.O2} \times \text{zone.xO2} + \text{u.H2O} \times \text{zone.xH2O};
\]

% Calculates $C_p(T)$ [J/kgK] with $C_p(T) = \left(\frac{C_p^0(T)}{R}\right) \times R$ for each gas
\[
\text{Cp.CO} = \text{CpR.CO} \times R_{CO};
\]
\[
\text{Cp.CO2} = \text{CpR.CO2} \times R_{CO2};
\]
\[
\text{Cp.H2} = \text{CpR.H2} \times R_{H2};
\]
\[
\text{Cp.N2} = \text{CpR.N2} \times R_{N2};
\]
\[
\text{Cp.O2} = \text{CpR.O2} \times R_{O2};
\]
\[
\text{Cp.H2O} = \text{CpR.H2O} \times R_{H2O};
\]

% Calculates $C_v(T)$ [J/kgK] with $C_v(T) = C_p(T) - R$ for each gas
\[
\text{Cv.CO} = \text{Cp.CO} - R_{CO};
\]
\[
\text{Cv.CO2} = \text{Cp.CO2} - R_{CO2};
\]
\[
\text{Cv.H2} = \text{Cp.H2} - R_{H2};
\]
\[
\text{Cv.N2} = \text{Cp.N2} - R_{N2};
\]
\[
\text{Cv.O2} = \text{Cp.O2} - R_{O2};
\]
\[
\text{Cv.H2O} = \text{Cp.H2O} - R_{H2O};
\]

% Calculates $C_v(T)$ for the specific zone in the two zone model
\[
\text{Cv.zone} = \text{Cv.CO} \times \text{zone.xCO} + \text{Cv.CO2} \times \text{zone.xCO2} + \text{Cv.H2} \times \text{zone.xH2} + \ldots \]
\[
\text{Cv.N2} \times \text{zone.xN2} + \text{Cv.O2} \times \text{zone.xO2} + \text{Cv.H2O} \times \text{zone.xH2O};
\]

% Calculates kappa [-] with $kappa = \frac{C_p}{C_v}$ for each gas
\[
\text{kappa.CO} = \frac{\text{Cp.CO}}{\text{Cv.CO}};
\]
\[
\text{kappa.CO2} = \frac{\text{Cp.CO2}}{\text{Cv.CO2}};
\]
\[
\text{kappa.H2} = \frac{\text{Cp.H2}}{\text{Cv.H2}};
\]
\[
\text{kappa.N2} = \frac{\text{Cp.N2}}{\text{Cv.N2}};
\]
\[
\text{kappa.O2} = \frac{\text{Cp.O2}}{\text{Cv.O2}};
\]
\[
\text{kappa.H2O} = \frac{\text{Cp.H2O}}{\text{Cv.H2O}};
\]

D.12 Function to calculate the specific entalphy in zone 1 and zone 2
function [ h, temp ] = ENTHALPHY(T2, Global)

% ENTHALPHY Calculates the enthalpy for the mixed gases in the combustion rig
% This function generates the enthalpy from NASA GLENN DATABASE
% for the mixed gases from zone 1 and zone 2 after pre-combustion
% is completed.

% Set temperature span for the calculations
T = round(T2);
% Temperatures 100 K above and 100 K below the final temperature in zone 2
temp = T-200:0.001:T+200;

% Set matrixes for increased calculation speed
h_CO = zeros(1,length(temp));
h_CO2 = zeros(1,length(temp));
h_H2 = zeros(1,length(temp));
h_N2 = zeros(1,length(temp));
h_O2 = zeros(1,length(temp));
h_H2O = zeros(1,length(temp));

% Defines the gas constants for use in this function
RCO = Global.RCO/1000; % [kJ/kgK]
RCO2 = Global.RCO2/1000; % [kJ/kgK]
RH2 = Global.RH2/1000; % [kJ/kgK]
RN2 = Global.RN2/1000; % [kJ/kgK]
RO2 = Global.RO2/1000; % [kJ/kgK]
RH2O = Global.RH2O/1000; % [kJ/kgK]

% Calculates Cp°(T)/R and H°(T)/RT using NASA GLENN DATABASE polynom

for i = 1:length(temp)

% Load coefficients from COEFFICIENTS function
[CO, CO2, H2, N2, O2, H2O] = COEFFICIENTS(temp(i));

% Calculates h(T) [J/kg] with h(T) = [H°(T)/RT]*RT for each gas
h_CO(i) = (-CO(1)*(temp(i)^-2) + CO(2)*(log(temp(i))/temp(i)) + CO(3) + CO(4)*temp(i)/2 + CO(5)*(temp(i)^2)/3 ... + CO(6)*(temp(i)^3)/4 + CO(7)*(temp(i)^4)/5 ... + CO(8)/temp(i)) * RCO*temp(i);

h_CO2(i) = (-CO2(1)*temp(i)^-2 + CO2(2)*(log(temp(i))/temp(i)) + ...
Appendix D. Matlab scripts

\[
\begin{align*}
\text{CO}_2(3) + \text{CO}_2(4) & \times \text{temp}(i) / 2 + \text{CO}_2(5) \times (\text{temp}(i)^2 / 3) + \ldots \\
\text{CO}_2(6) & \times (\text{temp}(i)^3 / 4) + \text{CO}_2(7) \times (\text{temp}(i)^4 / 5) + \ldots \\
\text{CO}_2(8) & / \text{temp}(i) \times \text{RCO}_2 \times \text{temp}(i);
\end{align*}
\]

\[
\begin{align*}
\text{h}_{\text{H}_2}(i) &= (-\text{H}_2(1) \times \text{temp}(i)^-2 + \text{H}_2(2) \times (\log(\text{temp}(i))/\text{temp}(i)) + \ldots \\
\text{H}_2(3) + \text{H}_2(4) & \times \text{temp}(i)/2 + \text{H}_2(5) \times (\text{temp}(i)^2 / 3) + \ldots \\
\text{H}_2(6) & \times (\text{temp}(i)^3 / 4) + \text{H}_2(7) \times (\text{temp}(i)^4 / 5) + \ldots \\
\text{H}_2(8) & / \text{temp}(i) \times \text{RH}_2 \times \text{temp}(i);
\end{align*}
\]

\[
\begin{align*}
\text{h}_{\text{N}_2}(i) &= (-\text{N}_2(1) \times \text{temp}(i)^-2 + \text{N}_2(2) \times (\log(\text{temp}(i))/\text{temp}(i)) + \ldots \\
\text{N}_2(3) + \text{N}_2(4) & \times \text{temp}(i)/2 + \text{N}_2(5) \times (\text{temp}(i)^2 / 3) + \ldots \\
\text{N}_2(6) & \times (\text{temp}(i)^3 / 4) + \text{N}_2(7) \times (\text{temp}(i)^4 / 5) + \ldots \\
\text{N}_2(8) & / \text{temp}(i) \times \text{RN}_2 \times \text{temp}(i);
\end{align*}
\]

\[
\begin{align*}
\text{h}_{\text{O}_2}(i) &= (-\text{O}_2(1) \times \text{temp}(i)^-2 + \text{O}_2(2) \times (\log(\text{temp}(i))/\text{temp}(i)) + \ldots \\
\text{O}_2(3) + \text{O}_2(4) & \times \text{temp}(i)/2 + \text{O}_2(5) \times (\text{temp}(i)^2 / 3) + \ldots \\
\text{O}_2(6) & \times (\text{temp}(i)^3 / 4) + \text{O}_2(7) \times (\text{temp}(i)^4 / 5) + \ldots \\
\text{O}_2(8) & / \text{temp}(i) \times \text{RO}_2 \times \text{temp}(i);
\end{align*}
\]

\[
\begin{align*}
\text{h}_{\text{H}_2\text{O}}(i) &= (-\text{H}_2\text{O}(1) \times \text{temp}(i)^-2 + \text{H}_2\text{O}(2) \times (\log(\text{temp}(i))/\text{temp}(i)) + \ldots \\
\text{H}_2\text{O}(3) + \text{H}_2\text{O}(4) & \times \text{temp}(i)/2 + \text{H}_2\text{O}(5) \times (\text{temp}(i)^2 / 3) + \ldots \\
\text{H}_2\text{O}(6) & \times (\text{temp}(i)^3 / 4) + \text{H}_2\text{O}(7) \times (\text{temp}(i)^4 / 5) + \ldots \\
\text{H}_2\text{O}(8) & / \text{temp}(i) \times \text{RH}_2\text{O} \times \text{temp}(i);
\end{align*}
\]

end

% Calculates the resulting enthalpy for the CR
h = h\_CO\_s.\_Global.\_xCOp + h\_CO\_2.\_Global.\_xCO2p + h\_H2.\_Global.\_xH2p ...
  
  + h\_N2.\_Global.\_xH2p + h\_O2.\_Global.\_xO2p + h\_H2\_O\_Global.\_xH2Op;

end

D.13 Rate of heat release calculations

function [ Global ] = ROHR( Global )
% ROHR Calculates rate of heat release (ROHR)
% ROHR is first calculated for the pre-combustion, then ROHR from the injection experiment is calculated
% Setting time span for the two-zone combustion
\texttt{time\_tz = Global.time\_two\_zones}; % [s]

% Mass transport from zone 1 to zone 2
\texttt{m12 = Global.m12}; % [kg]

% Lower heating value for mixture of CO an H2
\texttt{hn = Global.hn}; % [J/kg]

% Pressure in injection phase
\texttt{p = Global.p(Global.injection\_start:Global.injection\_end)}; % [Pa]

% Setting timespan for injection experiment
\texttt{time = Global.time(Global.injection\_start:Global.injection\_end)}; % [s]

% Temperature at injection start
\texttt{T\_injection = Global.T\_injection}; % [K]

% Volume of the combustion rig
\texttt{V = Global.V}; % [m^3]

% Rate of heat release for the pre-combustion phase

% Fit the mass function with a fourier fit function
\texttt{m12\_fit = fit(time\_tz,m12,'fourier8')};

% Collect the parameters from the fit
\texttt{m\_a1 = m12\_fit.a1;}
\texttt{m\_a2 = m12\_fit.a2;}
\texttt{m\_a3 = m12\_fit.a3;}
\texttt{m\_a4 = m12\_fit.a4;}
\texttt{m\_a5 = m12\_fit.a5;}
\texttt{m\_a6 = m12\_fit.a6;}
\texttt{m\_a7 = m12\_fit.a7;}
\texttt{m\_a8 = m12\_fit.a8;}
\texttt{m\_b1 = m12\_fit.b1;}
\texttt{m\_b2 = m12\_fit.b2;}
\texttt{m\_b3 = m12\_fit.b3;}
\texttt{m\_b4 = m12\_fit.b4;}
\texttt{m\_b5 = m12\_fit.b5;}
\texttt{m\_b6 = m12\_fit.b6;}
\texttt{m\_b7 = m12\_fit.b7;}
\texttt{m\_b8 = m12\_fit.b8;}
\texttt{m\_w = m12\_fit.w;}

% Calculate the rate of mass change from zone 1 to zone 2 with the fitted
% function to get a smooth curve
dm12_fit = m_w*(-m_a1*sin(m_w.*time_tz) - 2*m_a2*sin(2*m_w.*time_tz) ... 
      - 3*m_a3*sin(3*m_w.*time_tz) - 4*m_a4*sin(4*m_w.*time_tz) ... 
      - 5*m_a5*sin(5*m_w.*time_tz) - 6*m_a6*sin(6*m_w.*time_tz) ... 
      - 7*m_a7*sin(7*m_w.*time_tz) - 8*m_a8*sin(8*m_w.*time_tz) ... 
      + m_b1*cos(m_w.*time_tz) + 2*m_b2*cos(2*m_w.*time_tz) ... 
      + 3*m_b3*cos(3*m_w.*time_tz) + 4*m_b4*cos(4*m_w.*time_tz) ... 
      + 5*m_b5*cos(5*m_w.*time_tz) + 6*m_b6*cos(6*m_w.*time_tz) ... 
      + 7*m_b7*cos(7*m_w.*time_tz) + 8*m_b8*cos(8*m_w.*time_tz));

% Numerical differentiation of the mass curve
dm12 = zeros(1,length(m12));

for i = 3:(length(m12)-2)
    y = time_tz(i+1)-time_tz(i);
    dm12(i) = 1/y * (2/3*(m12(i+1)-m12(i-1)) ... 
                      - 1/12*(m12(i+2)-m12(i-2)));
end

% Assume linearity in the three first and three last points
dm12(1) = dm12(3);
dm12(2) = dm12(3);
dm12(end) = dm12(end-2);
dm12(end-1) = dm12(end-2);

% Calculate the ROHR for the pre-combustion phase both numerical
% and fitted with Fourier 8
ROHR_pre_combustion_numeric = hn.*dm12;
ROHR_pre_combustion = hn.*dm12_fit;

% Calculates the total heat release in the pre-combustion phase
Q_pre_fit = trapz(time_tz,ROHR_pre_combustion);

% Rate of heat release for the injection experiment, closed system analysis
% Set injection start to start of calculation
time_injection = time - Global.time(Global.injection_start);

% Create array to increase numerical differential calculation
dp_injection = zeros(1,length(p));
% Time between measurements
h = time(2)-time(1);

for j = 3:(length(p)-2)
    dp_injection(j) = 1/h * (2/3*(p(j+1)-p(j-1)) ...
        - 1/12*(p(j+2)-p(j-2)));
end

% Assume linearity in the three first and three last points
dp_injection(1) = dp_injection(3);
dp_injection(2) = dp_injection(3);
dp_injection(end) = dp_injection(end-2);
dp_injection(end-1) = dp_injection(end-2);

% Find a smooth pressure differential curve
dp_smooth = smooth(dp_injection');

% Calculate kappa for the injection experiment
[kappa] = KAPPA( Global, T_injection);

% Set arrays to increase calculation speed
ROHR_injection = zeros(1,length(p));
ROHR_injection_smooth = zeros(1,length(p));
Q_numeric = zeros(1,length(p));
Q_smooth = zeros(1,length(p));

% Calculate the ROHR for the injection experiment, and the total fuel
% energy released for both the numerically fitted pressure differential
% and the smooth curve
for l = 1:length(dp_injection)
    ROHR_injection(l) = 1/(kappa-1)*dp_injection(l)*V; %[J/s]
    ROHR_injection_smooth(l) = 1/(kappa-1)*dp_smooth(l)*V; %[J/s]
    Q_numeric(l) = h * trapz(ROHR_injection(1:l));
    Q_smooth(l) = h * trapz(ROHR_injection_smooth(1:l));
end

% Numerically fitted
% Find the maximum value
Q_numeric_100 = max(Q_numeric);

% Set values for three points of mass of fuel burned
Q1 = Q_numeric_100*Global.MFB1;
Q2 = Q_numeric_100*Global.MFB2;
Q3 = Q_numeric_100*Global.MFB3;

% Find the index where the set mass of fuel is burned
ind_1 = find(Q_numeric/Q_numeric_100>=Global.MFB1,1,'first');
ind_2 = find(Q_numeric/Q_numeric_100>=Global.MFB2,1,'first');
ind_3 = find(Q_numeric/Q_numeric_100>=Global.MFB3,1,'first');

% Use linear interpolation to find a better estimate of the time
if ind_1>=1
    Global.MFB_time_1 = (time_injection(ind_1-1) + (Q1-Q_numeric(ind_1-1))... 
    / Q_numeric(ind_1)-Q_numeric(ind_1-1)))\*1000;
else
    % If the calculation fails, set all values to zero
    Global.MFB_time_1 = 0;
end

% Smooth curve

% Find the maximum value
Q_smooth_100 = max(Q_smooth);

% Set values for three points of mass of fuel burned
Q1_smooth = Q_smooth_100*Global.MFB1;
Q2_smooth = Q_smooth_100*Global.MFB2;
Q3_smooth = Q_smooth_100*Global.MFB3;

% Find the index where the set mass of fuel is burned
ind1_s = find(Q_smooth/Q_smooth_100>=Global.MFB1,1,'first');
ind2_s = find(Q_smooth/Q_smooth_100>=Global.MFB2,1,'first');
ind3_s = find(Q_smooth/Q_smooth_100>=Global.MFB3,1,'first');

% Use linear interpolation to find a better estimate of the time
if ind1_s>1
    Global.MFB_time_1_smooth = (time_injection(ind1_s-1)...
        + (Q1_smooth-Q_smooth(ind1_s-1))...
        * ((time_injection(ind1_s)-time_injection(ind1_s-1))...
        /(Q_smooth(ind1_s)-Q_smooth(ind1_s-1))))*1000;
end

if ind2_s>1
    Global.MFB_time_2_smooth = (time_injection(ind2_s-1)...
        + (Q2_smooth-Q_smooth(ind2_s-1))...
        * ((time_injection(ind2_s)-time_injection(ind2_s-1))...
        /(Q_smooth(ind2_s)-Q_smooth(ind2_s-1))))*1000;
end

if ind3_s>1
    Global.MFB_time_3_smooth = (time_injection(ind3_s-1)...
        + (Q3_smooth-Q_smooth(ind3_s-1))...
        * ((time_injection(ind3_s)-time_injection(ind3_s-1))...
        /(Q_smooth(ind3_s)-Q_smooth(ind3_s-1))))*1000;
end
else
    % If the calculation fails, set all values to zero
    Global.MFB_time_1_smooth = 0;
    Global.MFB_time_2_smooth = 0;
    Global.MFB_time_3_smooth = 0;
end

% Set global parameters to display results in the GUI
Global.time_injection = time - Global.time(Global.injection_start);
Global.ROHR_injection_numeric = ROHR_injection;
Global.ROHR_injection_smooth = ROHR_injection_smooth;
Global.ROHR_pre_combustion = ROHR_pre_combustion;
Global.ROHR_pre_combustion_numeric = ROHR_pre_combustion_numeric;
Global.Q_pre_combustion = Q_pre_fit;
Global.Q_injection = trapz(time,ROHR_injection_smooth);
end
D.14 Function to calculate kappa for ROHR of injection experiment

```matlab
function [ kappa ] = KAPPA( Global, temp)
%Termo_properties Calculates the termodynamic properties of the combustion gases
% Uses the NASA GLENN DATABASE to do the calculations for finding Cp, Cv,
% and kappa = Cp/Cv for CO, CO2, H2, N2, O2, H2O

% Set matrixes for increased calculation speed
CpR_CO = zeros(1,length(temp));
CpR_CO2 = zeros(1,length(temp));
CpR_H2 = zeros(1,length(temp));
CpR_N2 = zeros(1,length(temp));
CpR_O2 = zeros(1,length(temp));
CpR_H2O = zeros(1,length(temp));

% Defines the gas constants for use in this function
RCO = Global.RCO/1000; % [kJ/kgK]
RCO2 = Global.RCO2/1000; % [kJ/kgK]
RH2 = Global.RH2/1000; % [kJ/kgK]
RN2 = Global.RN2/1000; % [kJ/kgK]
RO2 = Global.RO2/1000; % [kJ/kgK]
RH2O = Global.RH2O/1000; % [kJ/kgK]

% Calculates Cp^0(T)/R and H^0(T)/RT using NASA GLENN DATABASE polynom
for i = 1:length(temp)

    [CO, CO2, H2, N2, O2, H2O] = COEFFICIENTS(temp(i));

    CpR_CO(i) = (CO(1)*temp(i)^-2 + CO(2)*temp(i)^-1 + ...
               CO(3) + CO(4)*temp(i) + CO(5)*temp(i)^2 + ...
               CO(6)*temp(i)^3 + CO(7)*temp(i)^4);

    CpR_CO2(i) = (CO2(1)*temp(i)^-2 + CO2(2)*temp(i)^-1 + ...
                  CO2(3) + CO2(4)*temp(i) + CO2(5)*temp(i)^2 + ...
                  CO2(6)*temp(i)^3 + CO2(7)*temp(i)^4);
```
Appendix D. Matlab scripts

CpR\_H2(i) = (H2(1)*temp(i)^-2 + H2(2)*temp(i)^-1 + ... 
   H2(3) + H2(4)*temp(i) + H2(5)*temp(i)^2 + ... 
   H2(6)*temp(i)^3 + H2(7)*temp(i)^4);

CpR\_N2(i) = (N2(1) * temp(i)^-2 + N2(2) * temp(i)^-1 + ... 
   N2(3) + N2(4)*temp(i) + N2(5)*temp(i)^2 + ... 
   N2(6)*temp(i)^3 + N2(7)*temp(i)^4);

CpR\_O2(i) = (O2(1) * temp(i)^-2 + O2(2) * temp(i)^-1 + ... 
   O2(3) + O2(4)*temp(i) + O2(5)*temp(i)^2 + ... 
   O2(6)*temp(i)^3 + O2(7)*temp(i)^4);

CpR\_H2O(i) = (H2O(1)*temp(i)^-2 + H2O(2)*temp(i)^-1 + ... 
   H2O(3) + H2O(4)*temp(i) + H2O(5)*temp(i)^2 + ... 
   H2O(6)*temp(i)^3 + H2O(7)*temp(i)^4);

end

% Calculates Cp(T) [J/kgK] with Cp(T) = [Cp^0(T)/R]*R for each gas
Cp\_CO = CpR\_CO .* RCO;
Cp\_CO2 = CpR\_CO2 .* RCO2;
Cp\_H2 = CpR\_H2 .* RH2;
Cp\_N2 = CpR\_N2 .* RN2;
Cp\_O2 = CpR\_O2 .* RO2;
Cp\_H2O = CpR\_H2O .* RH2O;

% Calculates Cv(T) [J/kgK] with Cv(T) = Cp(T) - R for each gas
Cv\_CO = Cp\_CO - RCO;
Cv\_CO2 = Cp\_CO2 - RCO2;
Cv\_H2 = Cp\_H2 - RH2;
Cv\_N2 = Cp\_N2 - RN2;
Cv\_O2 = Cp\_O2 - RO2;
Cv\_H2O = Cp\_H2O - RH2O;

% Calculates kappa [-] with kappa = Cp/Cv for each gas
kappa\_CO = Cp\_CO./Cv\_CO;
kappa\_CO2 = Cp\_CO2./Cv\_CO2;
kappa\_H2 = Cp\_H2./Cv\_H2;
kappa\_N2 = Cp\_N2./Cv\_N2;
kappa\_O2 = Cp\_O2./Cv\_O2;
kappa\_H2O = Cp\_H2O./Cv\_H2O;
% Calculates kappa for the total mixture

kappa = kappa_CO*Global.xCOp + kappa_CO2*Global.xCO2p ...
    + kappa_H2*Global.xH2p + kappa_N2*Global.xN2p ...
    + kappa_O2*Global.xO2p + kappa_H2O*Global.xH2Op;
end

D.15 Output generated on user request

function [ ] = OUTPUT( Global )
% OUTPUT Function to write all calculated values to a textfile
% Takes all calculated values from TDAT and save them with the name
% stated by the user.

fileID = fopen(Global.Output,'w');

fprintf(fileID,'%20s %30s
',['Output file for ',Global.file_name]);
fprintf(fileID,'
');
fprintf(fileID,'%20s %12.3f %8s
','Rig radius = ',Global.r,'[m]');
fprintf(fileID,'%20s %12.3f %8s
','Rig height = ',Global.h,'[m]');
fprintf(fileID,'%20s %12.4f %8s
','Rig volume = ',Global.V,'[m^3]');
fprintf(fileID,'%20s %12.4f %8s
','Rig area = ',Global.A,'[m^2]');
fprintf(fileID,'
');
fprintf(fileID,'%20s %12.4f %8s
','x_CO = ',Global.xCOgas,'[-]');
fprintf(fileID,'%20s %12.4f %8s
','x_H2 = ',Global.xH2gas,'[-]');
fprintf(fileID,'%20s %12.4f %8s
','x_N2 = ',Global.xN2gas,'[-]');
fprintf(fileID,'%20s %12.4f %8s
','x_O2 = ',Global.xO2gas,'[-]');
fprintf(fileID,'
');
fprintf(fileID,'%19s %12.4f %7s
','m_gas = ',Global.mgas,'[kg]');
fprintf(fileID,'%19s %12.4f %7s
','m_CO = ',Global.mCOmix,'[kg]');
fprintf(fileID,'%19s %12.4f %7s
','m_H2 = ',Global.mH2mix,'[kg]');
fprintf(fileID,'
');
fprintf(fileID,'%20s %12.4f %8s
','Hn_CO = ',...
Global.hn_CO/1e6, '[MJ/kg]');
fprintf(fileID,'%20s %12.4f %8s \n','Hn_CO = ',Global.hn_CO/1e6, '[MJ/kg]');
fprintf(fileID,'\n');
fprintf(fileID,'%20s %12.4f %8s \n','Q_CO = ',Global.Q_CO/1000, '[kJ]');
fprintf(fileID, '\n');
fprintf(fileID,'%20s %12.4f %8s \n','Q_CO = ',Global.Q_CO/1000, '[kJ]');
fprintf(fileID,'\n');
fprintf(fileID,'%20s %12.4f %8s \n','Q_H2 = ',Global.Q_H2/1000, '[kJ]');
fprintf(fileID,'\n');
fprintf(fileID,'%20s %12.4f %8s \n','Q_tot = ',Global.Q_tot/1000, '[kJ]');
fprintf(fileID,'\n');
fprintf(fileID,'%20s %12.4f %8s \n','Q_pre_comb = ',... 
Global.Q_pre_combustion/1000, '[kJ]');
fprintf(fileID,'%20s %12.4f %8s \n','Q_fuel = ',... 
Global.Q_injection/1000, '[kJ]');
fprintf(fileID,'\n');
fprintf(fileID,'%20s %12.4f %8s \n','m_air = ',Global.mair, '[kg]');
fprintf(fileID,'%20s %12.4f %8s \n','m_O2_air = ',Global.mO2air, '[kg]');
fprintf(fileID,'%20s %12.4f %8s \n','m_N2_air = ',Global.mN2air, '[kg]');
fprintf(fileID,'\n');
fprintf(fileID,'%20s %12.4f %8s \n','m_CO_mix = ',Global.mCOmix, '[kg]');
fprintf(fileID,'%20s %12.4f %8s \n','m_O2_mix = ',Global.mO2mix, '[kg]');
fprintf(fileID,'\n');
fprintf(fileID,'%20s %12.4f %8s \n','xCO_mix = ',Global.xCOmix, '[-]');
fprintf(fileID,'%20s %12.4f %8s \n','xN2_mix = ',Global.xN2mix, '[-]');
fprintf(fileID,'\n');
fprintf(fileID,'%20s %12.2f %8s \n','R_mix = ',Global.Rmix, '[J/kgK]');
fprintf(fileID,'%20s %12.2f %8s \n','M_mix = ',Global.Mmix, '[g/mol]');
fprintf(fileID,'\n');
fprintf(fileID,'%20s %12.4f %8s \n','eta_comb = ',... 
Global.eta_comb(end)*100, ' [%]');
fprintf(fileID,'%20s %12.4f %8s \n','T_diff = ',... 
Global.T_diff(end), '[K]');
fprintf(fileID,'\n');
fprintf(fileID,'%20s %12.4f %8s \n','xCO_burned = ',Global.xCOp, '[-]');
fprintf(fileID,'%20s %12.4f %8s \n','xCO2_burned = ',Global.xCO2p, '[-]');
fprintf(fileID,'%20s %12.4f %8s \n','xO2_burned = ',Global.xO2p, '[-]');
fprintf(fileID,'\n');
fprintf(fileID,'%20s %12.2f %8s \n','R_burned = ',... 
Global.Rburned, '[J/kgK]');
fprintf(fid, '%20s %12.1f %8s \n', 'Injection time = ', ...
    Global.injection_time, '[s]');

fclose(fid);

end
Bibliography


