

Report_Christopher_Hopkins

by Christopher Hopkins

FILE	7278_CHRISTOPHER_HOPKINS_REPORT_CHRISTOPHER_HOPKINS_193356_1838633155.DOCX (1.77M)		
TIME SUBMITTED	04-MAY-2020 02:43AM (UTC+0100)	WORD COUNT	16717
SUBMISSION ID	126047968	CHARACTER COUNT	104427



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

**Using MATLAB to produce motorsport
regulation gasoline mixtures**

University of Wales Trinity Saint David

Motorcycle Engineering

19/04/2020

Major Project M6X00417

Christopher Hopkins
(P149588)

Declaration of Authenticity

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Signed: C.Hopkins

Date:03/05/2020

Abstract

The current study investigated the regulations of motorsport racing fuels, with the aim of creating a MATLAB script capable of mixing a MOTO GP regulation approved gasoline. A script could aid the fuel engineer within a race team, reducing the time taken for suitable mixtures to be created as per the users specified research octane number input.

The report begins with a literature review of the composition of gasoline. Following the development of the script, it was tested against the various MOTO GP regulations set for fuel by forcing illegal mixtures through the script. If the script was able to identify illegal components and provide an error message explaining this to the engineer, users would be able to manipulate components easily to produce a legal mixture.

From the investigation, the script was declared successful. Several tests were conducted to verify the script and its' ability to detect illegal concentrations and components. The final script provided the user with the correct volume of each hydrocarbon in litres to create the mixture.

Contents

Declaration of Authenticity	i
Abstract.....	i
1 Introduction.....	1
1.1 Aims.....	2
1.2 Objectives.....	2
2 Literature Review.....	3
2.1 Gasoline and its Composition.....	3
2.2 Paraffins	4
2.3 Aromatics.....	6
2.4 Naphthenes	7
2.5 Olefins	8
2.6 Oxygenate	8
2.6.1 Alcohols.....	8
2.6.2 Ethers.....	9
2.7 Thermochemistry.....	10
3 Methodology	11
3.1 Regulations.....	11
3.2 Component Tables	12
3.2.1 Paraffins Table	12
3.2.2 Aromatics Table	15
3.2.3 Naphthenes Table	15
3.2.4 Olefins Table	15
3.2.5 Oxygenates Table	16
3.3 Possible Combinations script	16
3.4 Main Script.....	20
3.5 Test outlines	35
3.5.1 RON value must be between 95 and 105:	35
3.5.2 MON value must be between 85 and 90:	35

3.5.3	Oxygen %(m/m) must be no higher than %2.7:	35
3.5.4	Benzene %(v/v) must be no higher than %1:	35
3.5.5	Olefins %(v/v) must be no higher than %18:	36
3.5.6	Aromatics %(v/v) must be no higher than %35:	36
3.5.7	Hydrocarbons with concentrations less than 5%m/m must be 30% of mixture	36
3.5.8	Component concentration for each carbon group must be below value	36
4	Results & Analysis	37
4.1	RON value must be between 95 and 105 & MON value must be between 85 and 90: 37	
4.1.1	RON value is set within range	37
4.1.2	RON value is set to value within range that cannot be achieved	37
4.1.3	RON set below 95	38
4.1.4	RON set above 105	38
4.1.5	RON set to 95	38
4.1.6	RON set to 105	39
4.2	Oxygen %(m/m) must be no higher than %2.7:	39
4.2.1	Oxygenates set to %14 of mixture	39
4.2.2	Oxygenates set to %15 of mixture	40
4.3	Benzene %(v/v) must be no higher than %1:	40
4.3.1	Benzene will be set to %1.5 of aromatics component	40
4.3.2	Benzene will be set to %10 of aromatics component	41
4.4	Olefins %(v/v) must be no higher than %18:	41
4.4.1	Olefins set to %20 of mixture	41
4.4.2	Olefins set to %21 of mixture	42
4.5	Aromatics %(v/v) must be no higher than %36:	43
4.5.1	Aromatics set to %38 of mixture	43
4.5.2	Aromatics set to %39 of mixture	43
4.6	Hydrocarbons with concentrations less than 5%m/m must be 30% of mixture	44
4.6.1	Cap set to 30%	44

4.6.2	Cap set to 20%	44
4.7	Component concentration for each carbon group must be below value	45
4.7.1	Standard mixture with no concentrations too high	45
4.7.2	Decreasing naphene concentration C5 upper limit	45
4.7.3	Decreasing olefin concentration C6 upper limit	45
4.7.4	Decreasing aromatics Concentration C7 upper limit	46
5	Recommended Further Work	47
6	Conclusion	48
7	References	49
8	Pandemic	54
9	Appendix.....	55
9.1	Appendix A	55
9.2	Appendix B	57
9.3	Appendix C	58
Figure 2.1	Normal fuel composition vs F1 race fuel (Groote, 2006)	3
Figure 2.2	DHA (Lee, et al., 2017)	3
Figure 2.3	Hydrocarbon prefix (Pulkrabek, 1997)	4
Figure 2.4	N-butane (Pulkrabek, 1997)	4
Figure 2.5	Iso-Butane (Pulkrabek, 1997)	5
Figure 2.6	Benzene ring (Pulkrabek, 1997)	6
Figure 2.7	Aromatic isomers (Pulkrabek, 1997)	6
Figure 2.8	Cyclopentane (Naphthene) (Pulkrabek, 1997)	7
Figure 2.9	Olefins examples (Pulkrabek, 1997)	8
Figure 3.1	DHA Paraffins (Elwardany, et al., 2016)	12
Figure 3.2	Mass fraction gasoline to mass fraction paraffin.....	13
Figure 3.3	Mass fraction conversion script	14
Figure 3.4	Combination script - Table load	16
Figure 3.5	Combination script - Properties Assign.....	16
Figure 3.6	Combination script - Solution concentration	17
Figure 3.7	Combination script - Total combinations.....	17
Figure 3.8	Combination script Column 1	19
Figure 3.9	Combination script - Other columns and 100% check.....	19

Figure 3.10 Main script - Table load	20
Figure 3.11 Main script - Property assignment	20
Figure 3.12 Main script - Final setup & Mix.....	21
Figure 3.13 Main script - Mole fraction error	21
Figure 3.14 Main script - RON&MON of component mixtures	21
Figure 3.15 Main script - Combination ID	22
Figure 3.16 Main script - RON ranges	22
Figure 3.17 Main script - RON range increments	23
Figure 3.18 Main script - RON variations table	23
Figure 3.19 Main script - Paraffins RON combinations.....	24
Figure 3.20 Main script - Other components RON combinations.....	24
Figure 3.21 Main script - Input Dialog	25
Figure 3.22 Main script - RON selection out of range error	25
Figure 3.23 Main script - User input RON and regulation MON.....	25
Figure 3.24 Main script - Final RON and MON ranges	25
Figure 3.25 Main script - Can't achieve RON error	26
Figure 3.26 Main script - Available combinations check	26
Figure 3.27 Main script - Real combination reduction.....	26
Figure 3.28 Main script - Real combinations check	26
Figure 3.29 Main script - Component composition reduction.....	27
Figure 3.30 Main script - Component properties.....	29
Figure 3.31 Main script - Separation of properties.....	29
Figure 3.32 Main script - saving ID of each component for real mixture.....	29
Figure 3.33 Main script - Volume of each substance	30
Figure 3.34 Main script - Density of Mixture	31
Figure 3.35 Main script - Final checks	32
Figure 3.36 Main script - Final checks removed	32
Figure 3.37 Main script - No remaining mixtures error.....	33
Figure 3.38 Main script - Mixture selection	33
Figure 3.39 Main script - select another fuel.....	34
Figure 4.1 Results - error 1	37
Figure 4.2 Results - error 2	38
Figure 4.3 2.7% mFuel.....	39
Figure 4.4 Oxygen content too high.....	40
Figure 4.5 Total mixture volume - Benzene	41
Figure 4.6 Benzene volume too high	41
Figure 4.7 Volume of olefins	42

Figure 4.8 Total volume	42
Figure 4.9 Olefin volume too high error	43
Figure 4.10 Aromatic volume too high error.....	44
Figure 4.11 Not enough hydrocarbons <5% <i>m/m</i> error.....	44
Figure 4.12 Naphthene carbon group concentration too high error	45
Figure 4.13 Olefin carbon group concentration too high error	45
Figure 4.14 Aromatic carbon group concentration too high error	46
Table 3-1 MOTO GP fuel regulations	11
Table 3-2 MOTO GP Regulations 2.....	11
Table 3-3 Paraffin Properties (ChemSpider, n.d.) (Chemical Book, n.d.) (CONCOA, n.d.) (McAllister, et al., 2011) (Ruscic & Bross, 2019) (NIST, n.d.) (Prasenjeet, et al., 2006) (Demirbas, et al., 2015) (Balaban, et al., 1992)	15
Table 3-4 Aromatics properties (NIST, n.d.) (ChemSpider, n.d.) (WoC, n.d.) (CarTech Books, n.d.) (EnggCyclopedia, n.d.)	15
Table 3-5 Naphthenes properties (NIST, n.d.) (ChemSpider, n.d.) (EnggCyclopedia, n.d.) (Prasenjeet, et al., 2006) (Demirbas, et al., 2015) (Balaban, et al., 1992)	15
Table 3-6 Olefins properties (NIST, n.d.) (ChemSpider, n.d.) (Balaban, et al., 1992) (ARGUS, 2019) (Yanowitz, 2011).....	16
Table 3-7 Oxygenates properties (NIST, n.d.) (ChemSpider, n.d.) (Prasenjeet, et al., 2006)	16
Table 3-8 Combination script - example process.....	18
Table 3-9 Main script - RON range table	23
Table 3-10 Main script - MON range table.....	23
Table 3-11 Main script - Volume table	31
Table 3-12 Main script - Final mixtures.....	33
Table 3-13 Main script - Volume required per substance in liters	34
Table 3-14 Mole fraction of mixture	35
Table 4-1 RON value is set within range	37
Table 4-2 RON value is set to value within range that cannot be achieved.....	37
Table 4-3 RON set below 95.....	38
Table 4-4 RON set above 105	38
Table 4-5 RON set to 95	38
Table 4-6 RON set to 105	39
Table 4-7 Oxygenates set to 14%.....	39
Table 4-8 Oxygenates set to 15%.....	40

Table 4-9 Benzene will be set at set to %1.5 of aromatics component	40
Table 4-10 Benzene will be set at set to %1.5 of aromatics component.....	41
Table 4-11 Olefins set to %20 of mixture.....	41
Table 4-12 Olefins set to %21 of mixture.....	42
Table 4-13 Aromatics set to %38 of mixture	43
Table 4-14 Aromatics set to %39 of mixture	43
Table 4-15 Hydrocarbons with concentrations less than 5% <i>m/m</i> must be 30% of mixture ..	44
Table 4-16 Hydrocarbons with concentrations less than 5% <i>m/m</i> must be 20% of mixture ..	44
Table 4-17 Standard mixture with no concentrations too high.....	45
Table 4-18 Decreasing naphene concentration C5 upper limit.....	45
Table 4-19 Decreasing olefin concentration C6 upper limit	45
Table 4-20 4.7.4 Decreasing aromatics concentration C7 upper limit.....	46
Equation 2-1 CO ₂	10
Equation 2-2 H ₂ O	10
Equation 2-3 Hess's law (Robinson, 2012).....	10
Equation 3-1 mass fraction conversion.....	12
Equation 3-2 <i>mFuel</i>	13
Equation 3-3 Moles (Pulkrabek, 1997).....	13
Equation 3-4 mole fraction	14
Equation 3-5 mass fraction verification	14
Equation 3-6 RON of mixture	21
Equation 3-7 Main script - Total combinations.....	23
Equation 3-8 Carbon/CO ₂ of mixture.....	27
Equation 3-9 H ₂ O	28
Equation 3-10 Enthalpy of formation.....	28
Equation 3-11 Enthalpy released from combustion	28
Equation 3-12 LHV.....	28
Equation 3-13 Density.....	30
Equation 3-14 Volume	30
Equation 3-15 Density Modified	30
Equation 3-17 Volume Required.....	34

1 Introduction

Motorsports have increasingly strict technical guidelines which vary between the type of motorsport being performed. Regulations are changed every few years to keep the sports interesting, but also provide development in the automotive industry towards more sustainable world. Thus, teams are constantly adapting to meet these changes. One section within most if not all regulations is fuel. Regulations are required in fuel for a number of reasons. For example, safety, maintaining a level playing field between teams, and adapting with advances in manufacturing more sustainable fuels. Evidence of this is given by the change in Formula 1 regulations between the 2021 Season:

“19.4.4 A minimum of 5.75% (m/m) of the fuel must comprise bio-components. “ (FIA, 2020)

And 2022:

“16.4.4 A minimum of 10% (m/m) of the fuel must comprise advanced sustainable Ethanol.” (FIA, 2020)

As seen from these regulations, there are marked differences in the time frame of one year. In this case not only does the mass % of a bio-component need to be increased from 5.75% to 10% the bio-component now needs to be Ethanol not any bio component.

Therefore, extensive time and research is required to not only produce a new fuel (gasoline) but to ensure it fits the regulations of the governing body. In the current project, a product will be created using MATLAB that will mix together multiple components of gasoline. Thus, enabling the development of a regulation friendly gasoline from the users input RON (research octane number). The product will be created based on MOTO GP 2020 regulations. However, simple modifications of the script will permit the calculator to be aimed at any motorsport.

To begin the report, a general understanding of the different components within gasoline composition is necessary. MOTO GP regulations will then be gathered and outlined clearly. Following this a list of gasoline components will be assembled with a realistic mole fraction range per component, and a MATLAB script will run through every possible mixture within each component mole fraction range, saving only those that total to 100%. All possible combinations will then be further scrutinised and reduced until legal fuels that fit the users specified RON are produced. This will enable the user to choose a gasoline and be provided with an output to create 22 liters of the fuel as per the MOTO GP tank capacity as stated in the regulations (FIM, 2020). The script will be verified by forcing numerous illegal fuel situations and to observe whether the script terminates with assistance to the user.

1.1 Aims

1. Create a MATLAB script capable of not only creating a gasoline out of individual components but a gasoline that would be deemed legal under MOTO GP regulations.
2. The substances used in the gasoline should be chosen by the user as well as the usable range for each substance.
3. The script will have the capability of outputting how much of each substance in liters is required to match the MOTO GP tank capacity and or the users defined limit.

1.2 Objectives

1. In order to achieve feasible gasoline mixtures, a literature review will be conducted into the composition of gasoline. Furthermore, research will be conducted into the mixing of hydrocarbons and the effects it has on their physical properties
2. To ensure that the mixtures meet MOTO GP regulations, they will first be gathered and set out into measurable regulations.
3. An excel document will be created that allows for hydrocarbons to be listed and the mole fraction ranges. Therefore, the user has control over the mixtures produced.
4. To create an output for each substance within the mixture, the density of each hydrocarbon must be obtained. This is so that when combined with the fuel mass the volume can be obtained.

2 Literature Review

2.1 Gasoline and its Composition

Gasoline which is primarily used as fuel for spark ignition vehicles is a mixture of crude oil and other petroleum liquids (U.S. Energy Information Administration, 2020). Although some petroleum refinery's produce some finished gasoline, most is manufactured at blending terminals where gasoline blendstocks (unfinished gasoline) are blended with finished gasoline and fuel grade ethanol. Gasoline blends are also changed depending on the season, the main difference being the vapour pressure of the fuel. The characteristics of the gasoline depend on the ingredients as well as the type of crude oil used and setup of the refinery. (Dixon & Halderman, 2012) identify gasoline as a hydrocarbon fuel made up of hydrocarbons with 1 – 15 carbon atoms. Those Hydrocarbons are split into 6 categories; alkanes (N-Paraffins / Iso-Paraffins/Naphthenes), alkenes (Olefins/Diolefins), alkynes, Arenes (Aromatics) and alcohol (Oxygenates). Supporting this, (Groote, 2006) categorised gasoline components into 2 categories; saturated and non-saturated. The former containing enough hydrogen atoms to saturate the carbon atoms, whereas the latter do not. Paraffins and naphthenic components are saturated, whilst aromatics and diolefins are non-saturated. (Groote, 2006) also noted the following examples on typical fuel composition of normal fuel and F1 race fuel

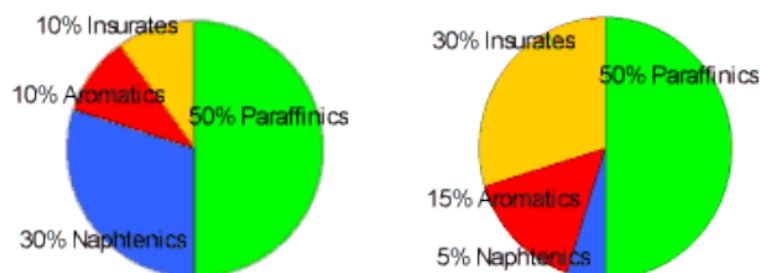


Figure 2.1 Normal fuel composition vs F1 race fuel (Groote, 2006)

These typical compositions were replicated by (CNBC International, 2014) who provide identical statistics. Figure 2.1 provides little information on oxygenates. However, (Lee, et al., 2017) performed a detailed hydrocarbon analysis (DHA) on 2 separate gasoline mixtures,

	Vol%	Mol%	Vol%	Mol%	demonstrating that
Oxygenates	8.2	16.8	4.0	8.2	oxygenates are included within the 50% paraffin section. Figure 2.2 shows this analysis.
Paraffins	13.4	12.2	10.0	10.1	
Iso-paraffins	33.7	25.1	38.5	31.9	
Olefins	6.8	6.3	10.0	11.2	
Naphthenes	15.2	15.6	5.1	5.0	
Aromatics	22.7	22.9	32.4	33.6	

Figure 2.2 DHA (Lee, et al., 2017)

(Stratiev, 2011) investigated the composition of four basic gasolines. The gasoline blends included n-paraffins, iso-paraffins, aromatics, naphthenes and olefins at varying percentages. This suggests that oxygenates are not a key component within gasoline and are seen more as an optional additive to the blend. This concept is further supported by a detailed hydrocarbon analysis (DHA) of gasoline, resulting in 5 component groups for the mixture used in (Stratiev, 2011) study. These results have also been obtained through catalytic cracking (Nazarova et al., 2016).

Hydrocarbons are assigned prefixes depending on their carbon atoms in main chain or ring as shown in Figure 2.3 (Pulkrabek, 1997).

TABLE 4-2 PREFIXES FOR HYDROCARBON FUEL COMPONENTS

Number of Carbon Atoms in Main Chain or Ring	Prefix
1	meth
2	eth
3	prop
4	but
5	pent
6	hex
7	hept
8	oct
9	non
10	dec
11	undec
12	dodec

Figure 2.3 Hydrocarbon prefix (Pulkrabek, 1997)

2.2 Paraffins

Paraffins are split into 2 separate categories. N-paraffins are single bonded, straight chained hydrocarbons, whereas iso-paraffins (short for isomers) have the same number of C and H atoms but arranged differently (Dixon & Halderman, 2012).

Paraffins are saturated hydrocarbons denoted by the formula C_nH_{2n+2} . Generally, paraffins with less than 5 carbon atoms are gaseous at room temperature. Those ranging between 5 – 15 carbon atoms are usually liquids whereas straight-chain paraffins containing more than 15 carbon atoms are solids. Branched chain paraffins however typically have a much higher-octane number and are therefore more sort after for gasoline components (Britannica, 2015).

Figures 2.4 and 2.5 show examples of straight-chained and branched-chained paraffins, both as butane molecules, with Figure 2.5 being an isomer.

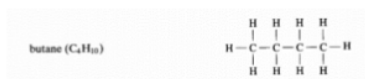


Figure 2.4 N-butane (Pulkrabek, 1997)

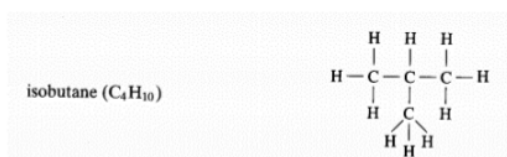


Figure 2.5 Iso-Butane (Pulkrabek, 1997)

Although both n-butane and iso-butane contain the same chemical formula, they both have different thermal and physical properties (Pulkrabek, 1997).

Paraffins form a large fraction of the hydrocarbons that are found within liquid fuels such as gasoline. Whereas, heavier paraffins exist in isolation as a wax-like solid. These heavier paraffins are soluble when mixed in lighter paraffins/hydrocarbons and can be found in diesel and fuel oils (Ester, 2018).

A DHA of gasoline demonstrated that the following n-paraffins were components of gasoline (Elwardany, et al., 2016):

- Propane
- N-butane
- N-pentane
- N-heptane

Additionally, the following isomer paraffins were found:

- Iso-butane
- Iso-pentane
- 2,2 dimethyl butane
- 2,3 dimethyl butane
- 2 methylpentane
- 3 methylpentane
- 2,4 dimethylpentane
- 2 methylhexane
- 2,3 dimethylpentane
- 3 methylhexane
- 3 ethylpentane
- 2,2,4 trimethylpentane
- 2,5 dimethylhexane
- 2,2,3 trimethylpentane
- 2,4 dimethylhexane

- 2,3,4 trimethylpentane
- 2 methyl- 3 -ethylpentane

2.3 Aromatics

Aromatics, another form of hydrocarbons possess an unsaturated ring structure. This type also includes a double carbon-carbon bond, meaning the carbon atoms are bonded by 4 electrons instead of just 2. The general chemical formula for an aromatic is C_nH_{2n-6} . The base molecule within this family is the benzene ring (Pulkrabek, 1997).

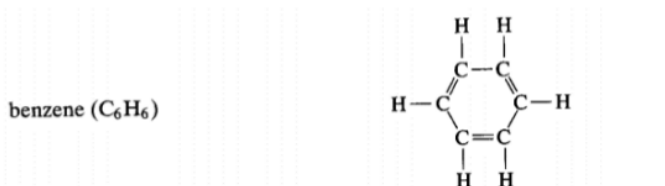


Figure 2.6 Benzene ring (Pulkrabek, 1997)

By replacing the hydrogen atoms located on the outside of the ring with various hydrogen and carbon groups, other molecules are then formed. Replacing multiple hydrogen atoms allows for many different isomers including orthoxylene, metaxylene, and paraxylene. See Figure 2.7 for these atom arrangements (Pulkrabek, 1997).

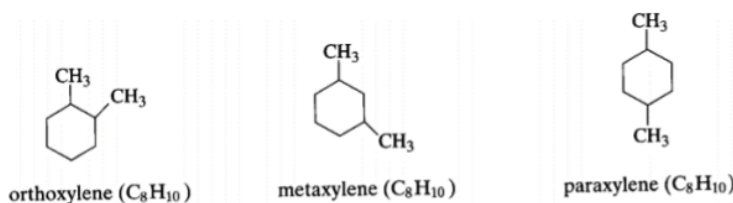


Figure 2.7 Aromatic isomers (Pulkrabek, 1997)

Aromatics are a good gasoline component that include high densities in liquid form paired with high energy content per unit volume. They do however require careful material selection for the fuel delivery system as their high solvency characteristics cause some gaskets to dissolve/swell (Pulkrabek, 1997).

Aromatic components in gasoline normally contain chemicals such as benzene, toluene and xylene with the purpose of increasing the octane rating of the fuel. This was previously achieved by lead up until 1995, when the component became banned due to being toxic and harmful to human health (Burroughs, 2015). More recently, these concerns were reinforced, and research demonstrates that the phase out of toxic octane enhancers such as lead and MTBE resulted in a volume increase of aromatics found within gasoline (Yang et al., 2019). Thus, meeting the anti-knock requirements and boosting the octane rating of gasoline. Both

(Yang, et al., 2019)& (El-Naggar & Al Majthoub, 2013) found that benzene, toluene, ethylbenzene and xylenes and commonly used with gasoline.

In (Elwardany, et al., 2016) study the DHA showed the following aromatics within the gasoline mixture:

- Benzene
- Toluene

2.4 Naphthenes

Naphthenes or cycloparaffins are an unsaturated hydrocarbon molecule containing a single-bond ring and chemical formula of C_nH_{2n} naphthenes allow for many variations of these molecules, one or more of the hydrogen atoms can be replaced with various chains. Naphthenes are also found within the composition of gasoline (Pulkrabek, 1997)

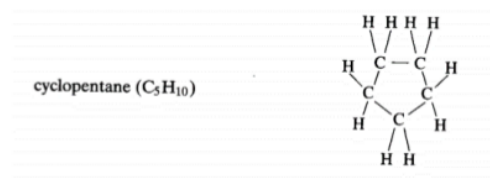


Figure 2.8 Cyclopentane (Naphthene) (Pulkrabek, 1997)

From the DHA results in (Elwardany, et al., 2016) study, the following naphthenes were found:

- Cyclopentane
- 1,1 dimethylcyclopentane
- 1t, 3 dimethylcyclopentane
- 1c,3 dimethylcyclopentane
- 1t,2 dimethylcyclopentane
- 1c,2 dimethylcyclopentane
- Methylcyclohexane
- Ethylcyclopentane
- 2t-ethylmethylcyclopentane

2.5 Olefins

Olefins are a chain hydrocarbon molecule containing one carbon double bond. Olefins are also unsaturated and follow the same prefixes listed in Figure 2.3 and the suffix of “ene”. The chemical formula for olefins is C_nH_{2n} . Like olefins, diolefins are chain molecules. However, there are 2 double carbon bonds rather than a single double carbon-carbon bond. These unsaturated compounds use the suffix “diene” and the chemical formula of C_nH_{2n-2} . (Pulkrabek, 1997)

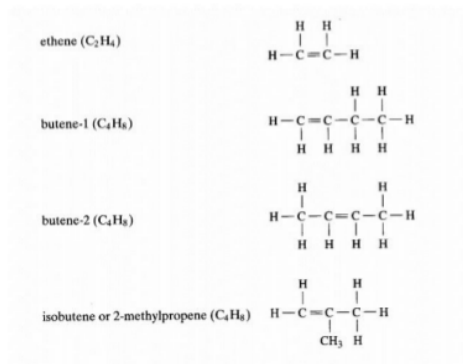


Figure 2.9 Olefins examples (Pulkrabek, 1997)

As the FIM regulations do not allow for diolefins, all compounds with the suffix of “diene” will not be selected for mixtures.

In the DHA from (Elwardany, et al., 2016) study, only one olefin was found within the mixture:

- Hexene-1

2.6 Oxygenate

2.6.1 Alcohols

Alcohols are a form of oxygenate, the molecules are fairly similar to those of paraffins except one of the hydrogen atoms is substituted for a hydroxyl radical (OH) (Pulkrabek, 1997).

Ethanol was the early automobiles fuel of choice to Henry Ford during development in the 1880s and 1890s (Houghton-alico, 1982). Petroleum is a non-renewable energy source, whereas ethanol is manufactured from natural products. Additionally, a high RON value meant ethanol demonstrated higher anti knock characteristics (Al-Hasan, 2003). (Bayraktar, 2005) investigated the effects of using gasoline ethanol blends on performance and emissions. Using a single cylinder, four-stroke, variable compression engine coupled to an electric dynamometer and exhaust gas analyser. The engine ran on a gasoline ethanol blend at 1.5%, 3%, 4.5%, 6%, 7.5%, 9%, 10.5% and 12%. The findings indicated that the effective power

(KW) of the engine increased until the ethanol in the mixture became 7.5%. Similar tests have been conducted (Al-Hasan, 2003) (Najafi, et al., 2015) (ARASU, et al., 2015) concluding similar effects from ethanol blends. However, peak performance varied. Therefore, the current literature shows a clear trend that adding ethanol to a gasoline blend benefits the performance output through an increased octane rating. Thus, allowing for higher compression ratios. However, there is a large degree of variation on what the optimum blend is.

Like ethanol, methanol is an alcohol permitted by the FIM. It has a much higher-octane rating than gasoline (Silva, et al., 2005), allowing for engines to run on higher compression ratios (Eyidogan, et al., 2010). The effects of methanol on engine performance and emissions were investigated by (Shenghua, et al., 2007), where the group studied the effects of low fraction methanol gasoline blends on a 3-cylinder engine. The experiment was conducted using various methanol gasoline blends from 0 – 25%. It was indicated that the power output and torque of the engine reduced as the concentration of methanol increased whereas the thermal efficiency increased. The results also demonstrated a reduction in CO emissions. These findings have been replicated, indicating a decline in emissions when methanol is blended into gasoline. (Agarwal, et al., 2014) (Shayan, et al., 2011). However, the Brake Torque and Brake Power increased as the compression ratio was increased, unlike Shenghua et al's (2007) results. These findings suggest that using methanol as an additive can boost the performance of the engine, providing that the compression ratio is increased alongside. Furthermore, that methanol on a whole reduces the production of emissions.

2.6.2 Ethers

Ethers are a gasoline additive that can improve the anti-knock index of a fuel. Methyl-tertiary butyl ether (MTBE) and ethyl tert-butyl ether (ETBE) are examples of short-chain oxygen containing hydrocarbons. These additives change the combustion characteristics of the fuel (Westphal, et al., 2010).

(Hamdan & Sunaih, 2002) explored the use of MTBE as an additive to gasoline, specifically investigating its impact on the performance of an engine. MTBE was added at 0%, 5%, 10%, 15%, 20%. As the MTBE concentration increased, the brake power of the engine also increased. This effect was greatest at 10%, suggesting that the use of MTBE positively benefits the performance of an engine when a variable compression ratio is used. Furthermore, (Silva, et al., 2005) conducted a similar study on both MTBE and ETBE, showing that increased concentration of the ethers directly and positively affected the RON of the gasoline. Although the power output of the engine was not directly investigated, with increased RON the engine could run at a higher compression ratio, potentially resulting in higher power outputs.

2.7 Thermochemistry

According to (Pulkrabek, 1997) the maximum chemical energy released from a fuel occurs when reacting with a stoichiometric amount of oxygen. Stoichiometric oxygen is just enough to convert the carbon atoms into CO_2 and the hydrogen atoms to H_2O . With no left-over oxygen, the balanced chemical equation of a hydrocarbon such as iso-octane C_8H_{18} would become 8CO_2 , as there are 8 carbon atoms and $9\text{H}_2\text{O}$ as there are 18 hydrogen atoms and H_2O have 2 hydrogen atoms. From this, 2 equations can be noted:

$$\text{CO}_2 = C$$

Equation 2-1 CO_2

$$\text{H}_2\text{O} = \frac{H}{2}$$

Equation 2-2 H_2O

When mixing fuels, the mole fraction of each hydrocarbon in the mixture directly effects the carbon and hydrogen atoms of the molecule i.e. if a mixture was 0.5 iso-octane(C_8H_{18}) and 0.5 methane (CH_4) the final molecule would be $\text{C}_{4.5}\text{H}_{11}$. Additionally, the RON and MON of a mixture can also be calculated this way.

Heat created from a combustion reaction of hydrocarbons and air is stated as the difference between the total enthalpy of products and reactants (enthalpy of reaction).

$$\Delta H_{\text{Reaction}}^{\circ} = \sum_{\text{Product}} N_i \Delta H_f^{\circ} - \sum_{\text{Reactants}} N_i \Delta H_f^{\circ}$$

Equation 2-3 Hess's law (Robinson, 2012)

Where N_i = stoichiometric coefficients

ΔH_f° = enthalpy of formation

(Robinson, 2012) states that that Hess's law can be used to determine the enthalpy change of a reaction if the enthalpy of formations for both products and the reactants are available.

Higher heating and lower heating value (also known as HHV&LHV) are the negative of the heat reaction for one unit of fuel. Higher heating value is used when H_2O in the exhaust products is in a liquid state and LHV is used when the H_2O is a vapour. Thus, the vaporisation of water results in a difference in heat. When conducting engine analysis, the LHV is used (Pulkrabek, 1997). Using this information, the enthalpy of H_2O as a gas is required. The NIST JANAF table states this to be 241.83kJ/mol (Ruscic & Bross, 2019).

3 Methodology

3.1 Regulations

The regulations set out by the FIM in the FIM WORLD CHAMPIONSHIP REGULATIONS 2020 (FIM, 2020) were recorded and will be monitored by the MATLAB script. Some properties such as visual appearance cannot be monitored by MATLAB and therefore will not be included in the regulations below. Additionally, unpermitted substances will not be entered. The script is designed to aid fuel engineers who are aware of substance regulation.

Table 3-1 MOTO GP fuel regulations

Property	Units	Min	Max
RON		95	105
MON		85	90
Oxygen	%(m/m)		2.7
Benzene	%(v/v)		1
Olefins	%(v/v)		18
Aromatics	%(v/v)		35

- The total of individual hydrocarbon components, containing only hydrogen and carbon, present at concentrations of less than 5% m/m must be at least 30% m/m of the fuel.
- The total concentration of naphthenes, olefins and aromatics in each carbon number group will not exceed the limits given in Table 3 – 2.

Table 3-2 MOTO GP Regulations 2

%m/m	C4	C5	C6	C7	C8	C9+
Naphthenes	0	5	10	10	10	10
Olefins	5	20	20	15	10	10
Aromatics			1.2	35	35	30

- Only the following oxygenates will be permitted: Methanol, Ethanol, Iso-propyl alcohol, Iso-butyl alcohol, Methyl tertiary butyl ether, Ethyl tertiary butyl ether, Tertiary amyl methyl ether, Di-isopropyl ether, n-Propyl alcohol, Tertiary butyl alcohol, n-Butyl alcohol, Secondary butyl alcohol.

All regulations documented on this page are taken directly from (FIM, 2020)

3.2 Component Tables

Each gasoline component requests a list of fuels to make up the component. A table is created in Excel for each component containing the following properties:

- Carbon atoms within the molecule
- Hydrogen atoms within the molecule
- Oxygen atoms within the molecule
- Research Octane Number (RON)
- Motor Octane Number (MON)
- Enthalpy of Formation
- Min mole fraction
- Max mole fraction
- Increment mole fraction
- Density

3.2.1 Paraffins Table

Using most of the n-paraffins and iso-paraffins from the DHA conducted in (Elwardany, et al., 2016) study, the minimum and maximum mole fractions were calculated by converting the mass fraction of paraffin by the mole fraction of the paraffin component.

	Group	component	% mass fraction
1	n-Paraffins	Propane	0.06
2		n-butane	0.55
3		n-pentane	0.05
4		n-heptane	13.94
5	Iso-paraffins	i-butane	3.27
6		i-pentane	3.05
7		2,2 dimethyl butane	0.28
8		2,3 dimethyl butane	1.81
9		2 methylpentane	5.55
10		3 methylpentane	2.06
11		2,4 dimethylpentane	2.22
12		2 methylhexane	10.26
13		2,3 dimethylpentane	6.40
14		3 methylhexane	13.77
15		3 ethylpentane	1.35
16		2,2,4 trimethylpentane	13.16
17		2,5 dimethylhexane	1.07
18		2,2,3 trimethylpentane	0.32
19		2,4 dimethylhexane	1.39
20		2,3,4 trimethylpentane	3.62

Figure 3.1 DHA Paraffins (Elwardany, et al., 2016)

Figure 3.1 details the mass fractions in gasoline, not the paraffins alone. Therefore, this data was first converted to determine mass fraction of the paraffin.

This was calculated by totalling the mass fraction of the paraffins being used (100% paraffin mass fraction) and then dividing the hydrocarbons mass fraction to the gasoline mixture by the sum of mass fraction that is paraffin. The values are produced in Figure 3.2.

$$\% m_{Fraction} = \frac{fuels m_{Fraction\ to\ gasoline}}{\sum m_{Fraction\ to\ paraffin}}$$

Equation 3-1 mass fraction conversion

Fuel		
2,3-DIMETHYLPENTANE	6.4	0.082357
Propane	0.06	0.000841
n-Butane	0.55	0.007719
iso-Butane	3.27	0.046252
iso-Pentane	3.05	0.045232
iso-Hexane	5.55	0.086207
2,2-DIMETHYLBUTANE	0.28	0.004759
2,3-DIMETHYLBUTANE	1.81	0.030914
3-METHYLPENTANE	2.06	0.036306
n-Heptane	13.94	0.254938
2,4-DIMETHYLPENTANE	2.22	0.054492
iso-heptane	10.26	0.266355
3-METHYLHEXANE	13.77	0.487261
iso-Octane	13.16	0.908213
2,2,3-TRIMETHYLPENTANE	0.32	0.240602
2-METHYL-3-ETHYLPENTANE	1.01	1

Figure 3.2 Mass fraction gasoline to mass fraction paraffin

The molar mass of each hydrocarbon was calculated by multiplying the atoms by their respective atomic mass and then summing the results.

- Hydrogen = 1
- Carbon = 12
- Oxygen = 16

(Wichers & Peiser, 2015)

$$m_{Fuel} = (C * carbon) + (H * hydrogen) + (O * Oxygen)$$

Equation 3-2 m_{Fuel}

The mass fraction was converted to a mole fraction utilising (Pulkrabek, 1997) method. Beginning by calculating the moles for each fuel.

$$N = m/M$$

Equation 3-3 Moles (Pulkrabek, 1997)

Where N is the amount of pure substance measured in moles, m is the mass of pure substance measured in grams and M is the molar mass of the pure substance measured in g mol^{-1} (AUS-e-TUTE, 2018).

This equation was used for each hydrocarbon where $m=1$ and M = molar mass of the hydrocarbon as calculated above, and the result multiplied by the hydrocarbons mass fraction. The total for each hydrocarbon (N) was calculated and summed to provide the overall total.

Finally, by dividing each individual hydrocarbons N by the total N the mole fraction was produced.

$$\text{mole Fraction} = \frac{N}{N_{\text{total}}}$$

Equation 3-4 mole fraction

Calculations were verified by calculating the molar mass of the blend as calculated above and then dividing the molar mass of the hydrocarbon by the molar mass of the blend.

$$\text{mass fraction} = \frac{\text{mass of individual fuel}}{\text{mass of mixture}}$$

Equation 3-5 mass fraction verification

This should produce the same numbers as the mass fraction input in Figure 3.2 column 3. Figure 3.3 demonstrates the above calculations in MATLAB. The full script can be found in appendix A.

```

32 %7 = mFuel
33 %8 = moles
34 %9 = mass Fraction
35 %nt = moles total
36 %10 = mole fraction
37 %13 = Carbon mix
38 %14 = Hydrogen mix
39 %15 = Oxygen mix
40 %16 = mFuel mix
41 %17 mass Fraction
42 PARAFFIN_PROPERTIES(:,7) = (PARAFFIN_PROPERTIES(:,1)*Carbon)+(PARAFFIN_PROPERTIES(:,2)*Hydrogen)+(PARAFF
43 for i = 1:size(PARAFFIN_PROPERTIES,1)
44 PARAFFIN_PROPERTIES(i,8) = 1/PARAFFIN_PROPERTIES(i,7);
45 end
46 PARAFFIN_PROPERTIES(:,8) = PARAFFIN_PROPERTIES(:,8).*PARAFFIN_PROPERTIES(:,9);
47 nt = sum(PARAFFIN_PROPERTIES(:,8))
48 PARAFFIN_PROPERTIES(:,10) = PARAFFIN_PROPERTIES(:,8)/nt;
49 PARAFFIN_PROPERTIES(:,13) = PARAFFIN_PROPERTIES(:,1).*PARAFFIN_PROPERTIES(:,10);
50 PARAFFIN_PROPERTIES(:,14) = PARAFFIN_PROPERTIES(:,2).*PARAFFIN_PROPERTIES(:,10);
51 PARAFFIN_PROPERTIES(:,15) = PARAFFIN_PROPERTIES(:,3).*PARAFFIN_PROPERTIES(:,10);
52 PARAFFIN_PROPERTIES(:,16) = (PARAFFIN_PROPERTIES(:,13)*Carbon)+(PARAFFIN_PROPERTIES(:,14)*Hydrogen)+(PA
53 mFuel = sum(PARAFFIN_PROPERTIES(:,16))
54 PARAFFIN_PROPERTIES(:,17) = PARAFFIN_PROPERTIES(:,16)/mFuel;

```

Figure 3.3 Mass fraction conversion script

With the molar fraction used in (Elwardany, et al., 2016) study found, the min and max mole fraction were set around these values. These are indicated by the maximum and minimum columns in Table 3-3

Table 3-3 Paraffin Properties (ChemSpider, n.d.) (Chemical Book, n.d.) (CONCOA, n.d.) (McAllister, et al., 2011) (Ruscic & Bross, 2019) (NIST, n.d.) (Prasenjeet, et al., 2006) (Demirbas, et al., 2015) (Balaban, et al., 1992)

Fuel	C	H	O	RON	MON	h	per	min	Study	max	i	p
2,3-DIMETHYLPENTANE	7	16	0	91	88	233.5	0.082357	5.00%	7.85%	9.00%	1.00%	0.695
Propane	3	8	0	112	97	105.03	0.000772	0.10%	0.17%	0.20%	0.05%	0.5
n-Butane	4	10	0	94	90	125.78	0.007078	0.80%	1.16%	1.40%	0.20%	0.5788
iso-Butane	4	10	0	102	98	134.64	0.04208	5.00%	6.91%	8.00%	1.00%	0.563
iso-Pentane	5	12	0	93	90	178.44	0.039248	3.00%	5.20%	6.00%	1.00%	0.62
iso-Hexane	6	14	0	104	94	204.02	0.071419	7.00%	7.91%	10.00%	1.00%	0.653
2,2-DIMETHYLBUTANE	6	14	0	91	93	213.4	0.003603	0.30%	0.40%	0.45%	0.05%	0.65
2,3-DIMETHYLBUTANE	6	14	0	104	94	207	0.023292	2.00%	2.58%	4.00%	1.00%	0.662
3-METHYLPENTANE	6	14	0	74	73	202	0.026509	2.00%	2.94%	3.00%	0.50%	0.669
n-Heptane	7	16	0	0	0	224.18	0.179385	14.00%	17.10%	18.00%	2.00%	0.684
2,4-DIMETHYLPENTANE	7	16	0	83	83	235	0.028568	2.00%	2.72%	3.00%	0.50%	0.673
iso-heptane	7	16	0	42	47	229.8	0.132029	10.00%	12.58%	16.00%	2.00%	0.679
3-METHYLHEXANE	7	16	0	52	55.8	227.4	0.177197	14.00%	16.89%	18.00%	2.00%	0.387
iso-Octane	8	18	0	100	100	259.2	0.169348	12.00%	14.16%	20.00%	2.00%	0.69
2,2,3-TRIMETHYLPENTANE	8	18	0	109	99	257.1	0.004118	0.30%	0.34%	0.50%	0.10%	0.72
2-METHYL-3-ETHYLPENTANE	8	18	0	87	88	249.7	0.012997	0.80%	1.09%	1.20%	0.20%	0.7

3.2.2 Aromatics Table

The aromatics table used the hydrocarbons found in (Elwardany, et al., 2016) study with the addition of m-Xylene, a substance commonly found in the composition of gasoline. The benzene mass fraction range was set at low values due to the regulation cap of 1%(V/V). Coinciding with (Elwardany et al., 2016) DHA findings, Toluene therefore made up the majority of the mixture.

Table 3-4 Aromatics properties (NIST, n.d.) (ChemSpider, n.d.) (WoC, n.d.) (CarTech Books, n.d.) (EnggCyclopedia, n.d.)

Fuel	C	H	O	RON	MON	AKI	h		per	min	max	i		p
Toluene	7	8	0	121	107	114	12	3.87	0.045428	0.00%	89.00%	0.50%	178	0.867
m-Xylene	8	10	0	145	124	134.5	25.4	0	0.00%	25.00%	0.50%	50	0.866	
Benzene	6	6	0	115	114.8	114.9	49.04	0.75	0.008804	1.00%	1.50%	0.50%	1	0.874

3.2.3 Naphthenes Table

The naphthenes table followed the same process as section 3.2.1 to assign mole fraction ranges.

Table 3-5 Naphthenes properties (NIST, n.d.) (ChemSpider, n.d.) (EnggCyclopedia, n.d.) (Prasenjeet, et al., 2006) (Demirbas, et al., 2015) (Balaban, et al., 1992)

Fuel	C	H	O	RON	MON	AKI	h	per	min	max	i	p			
1,3 dimethylcyclopentane	7	14	0	79	72	75.5	168.2	0.66	0.007747	26.94%	21.50%	32.50%	0.50%	22	0.784
1,4 dimethylcyclopentane	7	14	0	79	72	75.5	170.2	0.62	0.007278	25.31%	20.00%	30.50%	0.50%	21	0.75
METHYLCYCLOHEXANE	7	14	0	79	73	73	190.2	0.47	0.005517	19.18%	15.50%	23.00%	0.50%	15	0.771
1,1-DIMETHYLCYCLOPENTANE	7	14	0	92	89	90.5	172.1	0.29	0.003404	11.84%	9.50%	14.50%	0.50%	10	0.79
Cyclopentane	5	10	0	101	84	92.5	105.9	0.2	0.002348	11.43%	9.00%	14.00%	0.50%	10	0.748

3.2.4 Olefins Table

As (Elwardany, et al., 2016) study only contained 1 olefin, the range of the mole fractions were set from 0% - 100%, and a second hydrocarbon was added with the same range to allow for variation within this component.

Table 3-6 Olefins properties (NIST, n.d.) (ChemSpider, n.d.) (Balaban, et al., 1992) (ARGUS, 2019) (Yanowitz, 2011)

Fuel	C	H	O	RON	MON	AKI	h	per	min	max	I	p
1-PENTENE	5	10	0	90	77	83.5	49	0	0.00%	100.00%	1.00%	100 0.64
1-HEXENE	6	12	0	76	63	69.5	73	0	0.00%	100.00%	1.00%	100 0.673

3.2.5 Oxygenates Table

For the oxygenates table, all the regulation approved substances as outlined in section 3.1 were included, allowing for some mole fraction ranges to be set to 0%.

Table 3-7 Oxygenates properties (NIST, n.d.) (ChemSpider, n.d.) (Prasenjeet, et al., 2006)

Fuel	C	H	O	RON	MON	AKI	h	min	max	I	p
Ethanol	2	6	1	107	89	98	277.48	0.00%	100.00%	1.00%	100 0.7893
Methanol	1	4	1	106	92	99	238.59	0.00%	100.00%	2.00%	50 0.791
MTBE	5	12	1	117	103	110	315	0.00%	10.00%	1.00%	10 0.742
ETBE	6	14	1	119	105	112	350.8	0.00%	10.00%	1.00%	10 0.75
TAME	6	14	1	112	98	105	341.79	0.00%	10.00%	1.00%	10 0.76
n-Propyl alcohol	3	8	1	104	89	96.5	303	0.00%	0.00%	1.00%	0 0.804
Iso-propyl alcohol	3	8	1	106	99	102.5	317	0.00%	0.00%	2.00%	0 0.785
Iso butyl alcohol	4	10	1	105	90	97.5	334.6	0.00%	0.00%	2.00%	0 0.802
n-Butyl alcohol	4	10	1	98	85	91.5	328	0.00%	0.00%	1.00%	0 0.81
Tertiary butyl alcohol	4	10	1	105	89	97	359.2	0.00%	0.00%	2.00%	0 0.789
Di-isopropyl ether	6	14	1	110	100	105	351.5	0.00%	0.00%	1.00%	0 0.725
Secondary butyl alcohol	4	10	1	105	93	99	342.6	0.00%	0.00%	1.00%	0 0.806

3.3 Possible Combinations script

All five components were completed along with each substance within the tables (% of the mixture). For example, Iso-Octane could range between 5% - 10% of the total Paraffin contribution to the fuel. Combinations for each component that add up to 100% were gathered and set aside. This was necessary so there were no mixtures made of 50% Iso-Octane and 51% n-heptane, totalling 101% of a mole.

This requires a script that can check every possible combination of a substance against every other substance's possible combinations until every single combination is complete.

```

4 - folder = fileparts(which('Calculator_ChrisHopkins.xlsx'));
5 - fullFileName = fullfile(folder, 'Calculator_ChrisHopkins.xlsx');
6 - [status, sheetNames] = xlsfinfo(fullFileName);
7 - numSheets = length(sheetNames);
8 - PARAFFIN_TABLE = readtable(fullFileName, 'sheet', 'Paraffins');
```

Figure 3.4 Combination script - Table load

Lines 4 – 8 show the script loading in the specified table from sections 3.2.1 –

3.2.5. Only one table is looked at per script run because components with high volumes of substances take extended periods of time. Looking at individual components allows the user to modify one table at a time without recalculating all 5 components. When switching to the next table, line 8 'Paraffins' needs to be changed to 'Sheet name', where sheet name is the name of the excel sheet.

```

10 - PARAFFIN = PARAFFIN_TABLE(:, 'Fuel');
11 - PARAFFIN_PROPERTIES(:, 8) = round(PARAFFIN_TABLE(:, 'min'), 4);
12 - PARAFFIN_PROPERTIES(:, 9) = round(PARAFFIN_TABLE(:, 'max'), 4);
```

Figure 3.5 Combination script - Properties Assign

With the table of properties loaded into the MATLAB

workspace, the relevant properties for this script can be assigned to usable variables within the workspace. This is shown in lines 11-12 where they are assigned to a matrix in Figure 3.5. Line 10 is not important for the running of the script, but instead just holds the names of the hydrocarbons.

```

16- H = size(PARAFFIN_PROPERTIES,1);
17-
18- for j = 1:H
19-     k=1;
20-     Percent=0;
21-     for i = PARAFFIN_PROPERTIES(j,8):PARAFFIN_PROPERTIES(j,10):PARAFFIN_PROPERTIES(j,9)
22-         Percent(k) = i;
23-         k=k+1;
24-     end
25-     Percent_Table(j,1)=Percent;
26- end

```

Figure 3.6 Combination script - Solution concentration

Following this the concentration values for each hydrocarbon are calculated using the range and increments set.

The saved values are added to a table and the amount of variations are calculated. In this section, variable H is assigned to the number of hydrocarbons that will be part of the mixture meaning lines 18 – 26 are repeated for each hydrocarbon.

```

28- N(1)= size(Percent_Table{1,1},2);
29- for i = 1:H
30-     S(i)= size(Percent_Table{i,1},2);
31- end
32- for i = 1:H-1
33-     N(i+1) = S(i+1)*N(i);
34- end

```

Figure 3.7 Combination script - Total combinations

The total number of combinations is then calculated in Figure 3.7 This is calculated by using the total number of % variations that

each fuel has and multiplying each by each other. I.e. if there were 4 hydrocarbons with the following % variations (4, 2, 2, 4) the total combinations would be 64.

$$4 * 2 * 2 * 4 = 64$$

To ensure the script does not miss any combinations, the following method has been adopted. The same method for % variations as described above should be used for this combination script.

Table 3-8 Combination script - example process

1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
1	2	1	1
2	2	1	1
3	2	1	1
4	2	1	1
1	1	2	1
2	1	2	1
3	1	2	1
4	1	2	1
1	2	2	1
2	2	2	1
3	2	2	1
4	2	2	1
1	1	1	2
2	1	1	2
3	1	1	2
4	1	1	2

In each iteration column 1 remains static. The second column increases by one variation until it reaches the maximum variation and resets back to its lowest value. The next column then increases to the next variation. This process is repeated across the columns until the last column reaches its' highest value. Due to memory capacity, combinations are looked at in chunks of the first hydrocarbon % variations, rather than every possibility simultaneously. Capacity is reached because some combinations create a matrix with dimensions greater than 1.0×10^9 by 16, requiring very large quantities of RAM.

```

42 - for i=1:S(1)
43 -     Combination(j,1)=Percent_Table{m,1}(1,i);
44 -     j=j+1;
45 - end

```

Figure 3.8 Combination script Column 1

Figure 3.8 shows how the first column of combinations is created. It has a separate section as it never changes, meaning deleting and replacing repeating numbers would be time consuming and a waste of computer resources. In small quantities, this may not present a large issue. However, with large total combinations this wasted time accumulates.

```

53 - for b = S(1):S(1):N(H)
54 -     for m = 2:H
55 -         Combination(:,m)=Percent_Table{m,1}(1,o(m));
56 -         n(m)=n(m)+1;
57 -         if n(m) == (R(m-1)+1) && o(m) < S(m)
58 -             o(m) = o(m)+1;
59 -             n(m) = 1;
60 -         elseif n(m) == (R(m-1)+1) && o(m) == S(m)
61 -             o(m) = 1;
62 -             n(m) = 1;
63 -         end
64 -     end
65 -     CombinationTotal = sum(Combination,2);
66 -     for i = 1:S(1)
67 -         if CombinationTotal(i,1) == 1
68 -             Possible(T,:) = Combination(i,:);
69 -             T=T+1;
70 -         end
71 -     end
72 - end
73 - Possible = array2table(Possible);
74 - save('Paraffins.mat', 'Possible');

```

Figure 3.9 Combination script - Other columns and 100% check

Figure 3.9 shows how the other columns are created. A For Loop that starts at the number of %variations that the first hydrocarbon has, increases by the same amount each iteration (as more than one combination is made per iteration) until reaching the total number of combinations.

In Figure 3.9 variable n = the number of iterations each column has gone through without increasing to the next %variation and o = the %variation the column is up to. The second for loop dictates the column number the script is currently modifying (line 54). Lines 65-69 executes a sum of all the combinations for this loop (1st For Loop), and then checks each of the combinations to see if any= 1 (100%). If this is true, the combination is saved to a separate

matrix and are written over for the next loop. Finally, the 100% combination matrix is converted to a table and saved as a file to be used in the main script.

This process is repeated for all 5 components by changing 'Paraffin' on lines 8 – 13 to the next component name.

3.4 Main Script

For the main fuel creation script both the property tables 3-3 to 3-7 as well as the possible combinations tables calculated from the prior run script in section 3.3

```
3- clc
4- clear
5-
6- folder = fileparts(which('Calculator_ChrisHopkins.xlsx'));
7- fullFileName = fullfile(folder, 'Calculator_ChrisHopkins.xlsx');
8- [status, sheetNames] = xlsfinfo(fullFileName);
9- numSheets = length(sheetNames);
10- PARAFFIN_TABLE = readtable(fullFileName, 'sheet', 'Paraffins');
11- AROMATIC_TABLE = readtable(fullFileName, 'sheet', 'Aromatics');
12- NAPHTHENE_TABLE = readtable(fullFileName, 'sheet', 'Naphthenes');
13- OLEFIN_TABLE = readtable(fullFileName, 'sheet', 'Olefins');
14- OXYGENATE_TABLE = readtable(fullFileName, 'sheet', 'Oxygenates');
15- Paraffin_Combinations = load('Paraffins');
16- Paraffin_Combinations = table2array(Paraffin_Combinations.Possible);
17- Aromatic_Combinations = load('Aromatics');
18- Aromatic_Combinations = table2array(Aromatic_Combinations.Possible);
19- Naphthene_Combinations = load('Naphthenes');
20- Naphthene_Combinations = table2array(Naphthene_Combinations.Possible);
21- Olefin_Combinations = load('Olefins');
22- Olefin_Combinations = table2array(Olefin_Combinations.Possible);
23- Oxygenate_Combinations = load('Oxygenates');
24- Oxygenate_Combinations = table2array(Oxygenate_Combinations.Possible);
```

Figure 3.10 Main script - Table load

This was achieved through a series of readtable functions, taking the properties tables for each component and assigning them to a workspace variable. The load function was then used to load all possible combinations that flagged for each component and assigns them to a table within the

workspace. The table2array function was also required here to convert the loaded table to a usable matrix.

The main script requires 7 properties from each of the component tables including carbon, hydrogen, oxygen atoms per fuel, RON and MON of each fuel, enthalpy of formation and the density of each fuel.

```
26- PARAFFIN(1,:) = PARAFFIN_TABLE(:, 'Fuel');
27- Paraffin_PROPERTIES(:,1) = PARAFFIN_TABLE(:, 'C');
28- Paraffin_PROPERTIES(:,2) = PARAFFIN_TABLE(:, 'H');
29- Paraffin_PROPERTIES(:,3) = PARAFFIN_TABLE(:, 'O');
30- Paraffin_PROPERTIES(:,4) = PARAFFIN_TABLE(:, 'RON');
31- Paraffin_PROPERTIES(:,5) = PARAFFIN_TABLE(:, 'MON');
32- Paraffin_PROPERTIES(:,6) = PARAFFIN_TABLE(:, 'h');
33- Paraffin_PROPERTIES(:,7) = PARAFFIN_TABLE(:, 'p');
```

Figure 3.11 Main script - Property assignment

Figure 3.11 shows how properties were picked out of the tables of properties. Properties such as % mole fraction were left out as all combination work was

completed by the prior script and would be a waste of memory. This process was repeated on all 5 tables. To save memory, the tables were then cleared from the workspace using the clear (variable) function.

The final setup section of the main script outlined the atomic mass of oxygen, carbon and hydrogen, saving them to a workspace variable. The enthalpy of combustion for both CO₂ and H₂O (Ruscic & Bross, 2019) as a gas were assigned to a variable and finally the mixture % for each component was outlined within the vector Mix. Figure 3.12 details the component mole fraction from left to right; Paraffin, Aromatic, Naphthene, Olefin, Oxygenates. This can then be verified using an if statement to check if the mix adds up to 100%. If not, the user is displayed with the error message in Figure 3.13 and the script terminates.

```

76 %-----FUEL INFORMATION VARIABLES-----
77 %-----MASS-----
78 Oxygen = 16;
79 Carbon = 12;
80 Hydrogen = 1;
81 Nitrogen = 14;
82 aR = 78.09/20.95;
83 Air = (Oxygen*2)+(Nitrogen*2*aR);
84 %-----MASS-----
85 %-----ENTHALPY OF COMBUSTION-----
86 hCO2 = 393.5;
87 hH2O_g = 241.83;
88 %-----ENTHALPY OF COMBUSTION-----
89 %%
90 Mix = [0.32 0.30 0.05 0.18 0.14];
91 if sum(Mix) ~= 1
92     error = errordlg('Mole fraction of mixture does not = 100%');
93     uiwait (error)
94     return
95 end

```

Figure 3.12 Main script - Final setup & Mix

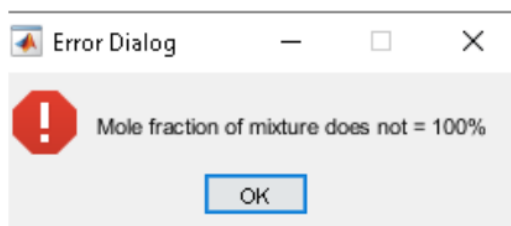


Figure 3.13 Main script - Mole fraction error

```

92 a_Paraffin(:,1) = (Paraffin_Combinations*Paraffin_PROPERTIES(:,4))*Mix(1);
93 a_Paraffin(:,2) = (Paraffin_Combinations*Paraffin_PROPERTIES(:,5))*Mix(1);

```

Figure 3.14 Main script - RON&MON of component mixtures

Using each component combination, the RON and MON values can be calculated for each component combination based on the properties table using this equation:

$$RON_{Mixture} = \sum RON_{HC} * mf$$

Equation 3-6 RON of mixture

Furthermore, the RON and MON can be scaled to the mole fraction by multiplying by variable Mix from line 91. These values are then given an ID number in the 3rd column which corresponds to the combination for later use as demonstrated by Figure 3.15.

```
3 - a_Paraffin(:,3) = 1:size(Paraffin_Combinations,1);
```

Figure 3.15 Main script - Combination ID

The next section of the script mixes possible RON values from the available mixes. This section will be used directly when selecting final mixtures. It removes the need to try every possible mixture between components, by stating what RON each component needs to be to achieve the final value.

The RON range is determined for each component first. This process begins with an if statement, checking to see whether the RON range rounded to the nearest whole number is 0. If true, the script will use rounded values of 2dp whereas if the statement returns false the values are only rounded to 1dp. This is important because if the RON values are all one whole number there is a potential for the first dp to be identical. This means that small RON ranges are identified by the script, but larger ones are not looked at in depth as a range can be identified without it.

```
112 - if max(round(a_Paraffin(:,1),0))-min(round(a_Paraffin(:,1),0)) == 0
113 -     RangeRON(1,1) = max(round(a_Paraffin(:,1),2));
114 -     RangeRON(2,1) = min(round(a_Paraffin(:,1),2));
115 - else
116 -     RangeRON(1,1) = max(round(a_Paraffin(:,1),1));
117 -     RangeRON(2,1) = min(round(a_Paraffin(:,1),1));
118 - end
119
```

Figure 3.16 Main script - RON ranges

The range is then split into increments that would allow for at least 10 steps in-between. This is to allow for a wider variety and therefore more refined search when selecting applicable components later. This was achieved using a while loop that checks whether increments of 1 would result in more than 10 variations. If it does not, 1 is divided by 10 and then the statement is rechecked and repeated until the variations are above 10.

```

.20 - while i==0
.21 -     if (RangeRON(1,1)-RangeRON(2,1))/RangeRON(4,1)<10
.22 -         RangeRON(4,1)=RangeRON(4,1)/10;
.23 -     else
.24 -         i = i+1;
.25 -     end
.26 - end

```

Figure 3.17 Main script - RON range increments

The processes outlined in Figures 3.16 and 3.17 were repeated for each component producing the following matrix:

Table 3-9 Main script - RON range table

23.1000	38.1000	4.1900	17.1000	15.4000
19.9000	37	4.0800	14.4000	14.8400
33	12	12	28	57
0.1000	0.1000	0.0100	0.1000	0.0100

This matrix is also required for MON. This section of the script is almost identical and therefore will be included in Appendix C.

Table 3-10 Main script - MON range table

22.7000	33.4100	3.8500	14.6000	13.3000
19.8000	32.6200	3.7900	12	12.5000
30	80	61	27	81
0.1000	0.0100	1.0000e-03	0.1000	0.0100

With the range tables complete, the script can find all combinations of RON mixtures. First, each possible variation for all component must be saved to a table. This was achieved using a similar script to the one described in section 3.3 Figure 3.9

First, the number of possible combinations is calculated using the following equation:

$$Total\ combinations = P_{Variations} * A_{Variations} * N_{Variations} * Ole_{Variations} * Oxy_{Variations}$$

Equation 3-7 Main script - Total combinations

```

274 - for i = 1:4
275 -     N(i+1) = RangeRON(3,i+1)*N(i);
276 - end
277 - R = [1 0 0 0 0];
278 - for i = 2:5
279 -     R(i)=R(i-1)*RangeRON(3,i);
280 - end
281 - for j = 1:5
282 -     k=1;
283 -     Ranges=0;
284 -     for i = RangeRON(2,j):RangeRON(4,j):RangeRON(1,j)
285 -         Ranges(k) = i;
286 -         k=k+1;
287 -     end
288 -     Ranges_Table_RON(1,j)=Ranges;
289 - end

```

Figure 3.18 Main script - RON variations table

This section of the script goes through each component individually and saves each RON variation to an indexed cell

Following this, the combinations can be produced. The first column (paraffins) was completed using For Loops. The first in the series ranged from 0 to the total combinations, with increments set to the variation amount for paraffins. This For Loop is used for matrix placement. The second loop writes all the variations recorded for the paraffins down the column. The first loop

```
295 - for j = 0:length(Ranges_Table RON{1,1}): (N(5)-4)
296 -     for i = 1:1:RangeRON(3,1)
297 -         combination RON(i+j,1)= Ranges_Table RON{1,1}(1,i);
298 -     end
299 - end
300
```

tells the second where to stop writing.

Figure 3.19 Main script - Paraffins RON combinations

Columns 2 – 5 require a slight variation to the script. The initial For Loop is kept the same but variations are written in chunks of the same number until enough combinations pass. The script then checks whether that column will need to change to the next variation on the coming For Loop and continues writing until reaching the last row. As outlined in Table 3-8.

```
05 - for j = 0:length(Ranges_Table RON{1,1}): (N(5)-4)
06 -
07 -     combination RON(j+1:j+length(Ranges_Table RON{1,1}),2)= Ranges_Table RON{1,2}(1,i);
08 -     k=k+length(Ranges_Table RON{1,1});
09 -     if k == R(1)*length(Ranges_Table RON{1,1})
10 -         k=0;
11 -         if i == RangeRON(3,2)
12 -             i=1;
13 -         else
14 -             i=i+1;
15 -         end
16 -     else
17 -         end
18 - end
19
```

Figure 3.20 Main script = Other components RON combinations

The section of the script between Figure 3.18 and 3.20 is repeated for MON combinations. Each combination is then totalled to produce the final RON and MON for each combination. This was calculated using the sum function on all rows within the matrix.

Following this, a dialog box opens for the user asking for an input RON, required for the final mixed fuel. The dialog box asks for an input between 95 and 105. This is the permitted range outlined in MOTO GP guidelines as detailed in section 3.1. To guarantee that this range has been selected, an If statement is used to check the user input. If the input does not fall into this range an error message is displayed and the user is asked to re-enter the values. The RON range is set to the value entered +0.1. Therefore, if the user inputs 105, the input is reduced by 0.1 so that the maximum is still 105. This ensures that the MON range is set to the regulation range. This can be overridden if the user wanted a range set from 95-105 by removing the dialog and setting the values manually. The input dialog is displayed in Figure 3.21 error message shown in Figure 3.22

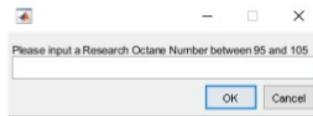


Figure 3.21 Main script - Input Dialog

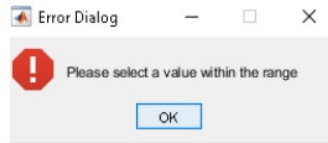


Figure 3.22 Main script – RON selection out of range error

```

while U == 1
    RON_min = str2double(inputdlg('Please input a Research Octane Number between 95 and 105'));
    if RON_min > 95 && RON_min < 105
        U = 0;
    else
        error = errordlg('Please select a value within the range');
        uiwait (error);
    end
end
if RON_min == 105
    RON_min = 104.9;
    RON_max = 105;
else
    RON_max = RON_min+0.1;
end
MON_min = 85;
MON_max = 90;

```

Figure 3.23 Main script - User input RON and regulation MON

Next, using the user Input RON value, the RON combinations variable (FinalRange_RON) is sorted through so that all RON values that are lower than RON_min and greater than the RON_max range are removed from the matrix. This is then repeated on the MON combinations (FinalRange_MON) using the regulation range.

```

494 - FinalRange_RON = FinalRange_RON(FinalRange_RON(:,8) >= RON_min, :);
495 - FinalRange_RON = FinalRange_RON(FinalRange_RON(:,8) <= RON_max, :);
496 - FinalRange_MON = FinalRange_MON(FinalRange_MON(:,8) >= MON_min, :);
497 - FinalRange_MON = FinalRange_MON(FinalRange_MON(:,8) <= MON_max, :);

```

Figure 3.24 Main script - Final RON and MON ranges

Empty variables can be produced which would result in the script crashing. To deal with this issue, a check is executed to see if either FinalRange_RON or FinalRange_MON are empty. If the latter is true, an error message is displayed to the user as shown in Figure 3.25, and the script is terminated. Due to memory problems, certain variables are removed throughout the script, including combinations outside of the ranges. The script needs to be restarted to change the RON value in this instance.

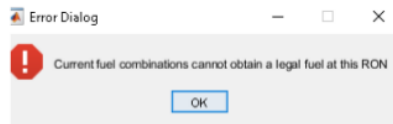


Figure 3.25 Main script – Can't achieve RON error

```

499 - if isempty(FinalRange_RON) == 1 || isempty(FinalRange_MON) == 1
500 -     error = errordlg('Current fuel combinations cannot obtain a legal fuel at this RON');
501 -     uiwait (error);
502 -     return
503 - else
504 - end

```

Figure 3.26 Main script - Available combinations check

The final range combinations for both RON and MON now produce the maximum and minimum RON and MON required for each component. These will be paired with other components so that the final fuel mix is within the ranges set. Using these minimum and maximum values enables the real component mixtures to be reduced, based on the format above. However, this time both MON and RON are reducing the same variable, leaving only the component combinations which produce RON and MON values within the ranges. This is repeated for each component.

```

a_Paraffin = a_Paraffin(a_Paraffin(:,1) >= min(FinalRange_RON(:,1)), :);
a_Paraffin = a_Paraffin(a_Paraffin(:,1) <= max(FinalRange_RON(:,1)+RangeRON(4,1)), :);
a_Paraffin = a_Paraffin(a_Paraffin(:,2) >= min(FinalRange_MON(:,1)), :);
a_Paraffin = a_Paraffin(a_Paraffin(:,2) <= max(FinalRange_MON(:,1)+RangeMON(4,1)), :);

```

Figure 3.27 Main script - Real combination reduction

Following the reduction in combinations for each component, an additional check is performed to identify any components with no combinations left. If the latter is true, another error message identical to Figure 3.25 is displayed to the user and the script terminates.

```

if isempty(a_Paraffin) == 1 || isempty(a_Aromatic) == 1 || isempty(a_Naphthene) == 1 || isempty(a_Olefin) == 1 || isempty(a_Oxygenat
error = errordlg('Current fuel combinations cannot obtain a legal fuel at this RON');
uiwait (error);
return
else
end

```

Figure 3.28 Main script - Real combinations check

With the real combination RON and MON values along with their ID number (column 3 as mentioned earlier) remaining, the real combination mix values can now be reduced. This includes every combination where the ID is not found within the real combination RON and MON matrix (e.g. a_Paraffin). To achieve this however, the matrix dimensions must match in the row direction. A second variable was created filled with zeros with matching rows to the

corresponding combinations variable (e.g. Paraffin_Combinations), and one column made up of the ID numbers (column 3) from the real combinations RON and MON matrix.

This means that all cells not containing an ID will correspond to 0, not an ID number. An extra column is added to the initial combination variable assigning them an ID number to cross check with. This variable still holds every combination in order. The script then goes through deleting any combination with an ID number not listed within the matrix just created (e.g. b_Paraffin). This process is repeated for each component. Once completed all a_ and b_ variables and ID numbers are removed to maximise memory capacity.

```
540 - A=size(Paraffin_Combinations,1);
541 - B=size(Paraffin_Combinations,2)+1;
542 - b_Paraffin = zeros(A,1);
543 - b_Paraffin(1:size(a_Paraffin,1)) = a_Paraffin(:,3);
544 - Paraffin_Combinations(:,B) = (1:A);
545 - Paraffin_Combinations(~ismember(Paraffin_Combinations(:,B),b_Paraffin), :) = [];
```

Figure 3.29 Main script - Component composition reduction

With all the redundant combinations removed, the properties of each combination for each component can be calculated. The properties calculated are as follows:

- Carbon Atoms & CO₂ produced
- Hydrogen Atoms
- Oxygen Atoms
- H₂O produced
- Mass of fuel
- Enthalpy of Formation
- hfNet
- RON
- MON
- LHV

The carbon atoms and subsequent CO₂ produced for this component are calculated by multiplying each individual hydrocarbon's carbon atoms by its mole fraction and then summing the results. This process is then repeated on the hydrogen and oxygen atoms.

$$C = (C_1 * mf_1) + (C_2 * mf_2) \dots$$

Equation 3-8 Carbon/CO₂ of mixture

Where C_n is the amount of Carbon in one mole of the pure substance and mf_n is the mole fraction assigned to the hydrocarbon.

The H₂O produced from combustion is then calculated by halving the amount of hydrogen atoms.

$$H_2O = \frac{H}{2}$$

Equation 3-9 H₂O

Where H is hydrogen atoms in the mixture.

The atomic mass of the fuel is calculated as shown in Equation 3-2 where C, H and O are the atoms within the mixture.

The enthalpy of formation for the mixture can be calculated using the sum of the hydrocarbons enthalpy of formation multiplied by their respective mole fraction.

$$\Delta H_f = \sum \Delta H_{fn} * mf_n$$

Equation 3-10 Enthalpy of formation

Where ΔH_{fn} is the enthalpy of formation for the hydrocarbon and mf_n is the mole fraction of the mixture.

The enthalpy released from combustion can then be calculated. Using the CO₂ and H₂O produced from the mixture combusting, these values are multiplied by their respective enthalpy of combustion and then removing the enthalpy of formation for the fuel from the result.

$$\Delta H = (CO_2 * 393.5) + (H_2O * 241.83) - \Delta H_f$$

Equation 3-11 Enthalpy released from combustion

The RON and MON of the mixture can then be calculated by multiplying the individual hydrocarbon values by the mole fraction as shown above in Equation 3-6.

The LHV of the mixture is calculated using the following equation:

$$LHV = \frac{hfNet}{mFuel}$$

Equation 3-12 LHV

Where hfNet is the enthalpy from combustion and mFuel is the mass of the mixture.

After all components are calculated, each matrix is then multiplied by the final fuel mixture as shown by line 609.

```

598 - A_Paraffin_Mix = zeros(size(Paraffin_Combinations,1),10);
599 - A_Paraffin_Mix(:,1) = Paraffin_Combinations*Paraffin_PROPERTIES(:,1); %C and CO2
600 - A_Paraffin_Mix(:,2) = Paraffin_Combinations*Paraffin_PROPERTIES(:,2); %H
601 - A_Paraffin_Mix(:,3) = Paraffin_Combinations*Paraffin_PROPERTIES(:,3); %O
602 - A_Paraffin_Mix(:,4) = A_Paraffin_Mix(:,2)/2; %H2O
603 - A_Paraffin_Mix(:,5) = (A_Paraffin_Mix(:,1)*Carbon)+(A_Paraffin_Mix(:,2)*Hydrogen); %mFuel
604 - A_Paraffin_Mix(:,6) = Paraffin_Combinations*Paraffin_PROPERTIES(:,6); %Enthalpy of Formation
605 - A_Paraffin_Mix(:,7) = (A_Paraffin_Mix(:,1)*hCO2)+(A_Paraffin_Mix(:,4)*hH2O_g)-A_Paraffin_Mix(:,6);
606 - A_Paraffin_Mix(:,8) = Paraffin_Combinations*Paraffin_PROPERTIES(:,4); %RON
607 - A_Paraffin_Mix(:,9) = Paraffin_Combinations*Paraffin_PROPERTIES(:,5); %MON
608 - A_Paraffin_Mix(:,10) = A_Paraffin_Mix(:,7)./A_Paraffin_Mix(:,5); %LHV
609 - A_Paraffin_Mix = A_Paraffin_Mix*Mix(1);

```

Figure 3.30 Main script - Component properties

```

666 - A=size(Paraffin_Combinations,1);
667 - Usable_Mix_Par = A_Paraffin_Mix(:,8:10);
668 - Usable_Mix_Par(:,4) = (1:A);

```

Figure 3.31 Main script - Separation of properties

The next section of the script is responsible for creating the regulation legal gasoline mixtures.

To create the legal fuels however, the RON MON and LHV values are again separated from the actual mixtures and given ID numbers again. The separation is necessary as this section is within a for loop, and combinations will be removed that will need to be re added each loop.

```

587 - Usable_Mix_Par(Usable_Mix_Par(:,1) < FinalRange_RON(1,1) | Usable_Mix_Par(:,1) > FinalRange
588 - Usable_Mix_Aro(Usable_Mix_Aro(:,1) < FinalRange_RON(1,2) | Usable_Mix_Aro(:,1) > FinalRange
589 - Usable_Mix_Nap(Usable_Mix_Nap(:,1) < FinalRange_RON(1,3) | Usable_Mix_Nap(:,1) > FinalRange
590 - Usable_Mix_Ole(Usable_Mix_Ole(:,1) < FinalRange_RON(1,4) | Usable_Mix_Ole(:,1) > FinalRange
591 - Usable_Mix_Oxy(Usable_Mix_Oxy(:,1) < FinalRange_RON(1,5) | Usable_Mix_Oxy(:,1) > FinalRange
592 - if isempty(Usable_Mix_Par)==1 || isempty(Usable_Mix_Aro)==1 || isempty(Usable_Mix_Nap)==1 || ise
593 - else
594 -     sortrows(Usable_Mix_Par,3);
595 -     sortrows(Usable_Mix_Aro,3);
596 -     sortrows(Usable_Mix_Nap,3);
597 -     sortrows(Usable_Mix_Ole,3);
598 -     sortrows(Usable_Mix_Oxy,3);
599 -     Usable(p,1)=Usable_Mix_Par(1,4);
600 -     Usable(p,2)=Usable_Mix_Aro(1,4);
601 -     Usable(p,3)=Usable_Mix_Nap(1,4);
602 -     Usable(p,4)=Usable_Mix_Ole(1,4);
603 -     Usable(p,5)=Usable_Mix_Oxy(1,4);
604 -     p=p+1;
605 - end
606 -

```

Figure 3.32 Main script - saving ID of each component for real mixture

The script will then use the FinalRange_RON table. This table contains combinations of component RON values that add up to the user input RON, enabling values that are either equal to or 0.1 above the

regulated range. This number can be modified to find mixtures closer or further from the RON input. All values outside of this range are removed. This is achieved by removing the combinations with RON values either side of the threshold and then checking whether any combination is empty. If empty, the script returns to Figure 3.31. Otherwise, the components are sorted so that the combination with the highest LHV has the ID numbers for each component saved to a single variable row. The script then returns to Figure 3.31 (the script only selects 1 combination per loop to maximise memory capacity). Each pass will look at the next combinations row on the FinalRange_RON matrix until all rows have been checked. ID numbers are then paired to the A_ variables with the respective properties and each component property added together to create final fuel mixtures from the selection.

Once the final fuel mixtures have been created, the density of each mixture must be calculated.

$$\rho = \frac{m_{total}}{V_n + V_n \dots}$$

Equation 3-13 Density

Where m_{total} is the mass of the mixture and V_n is the volume of the substances within the components.

Volume can be calculated using the following equation:

$$V = \frac{m}{\rho}$$

Equation 3-14 Volume

by substituting V with Equation 3-14 the following equation is produced:

$$\rho = \frac{m_{total}}{\frac{m_n}{\rho_n} + \frac{m_n}{\rho_n} \dots}$$

Equation 3-15 Density Modified

Where m_n is the mass of the substance within the mixture and ρ_n is the density of the substance.

Equation 3-14 was used to calculate the volume of each substance.

```

708 - for j = 1:size(Usable,1)
709 -     for i = 1:size(Paraffin_PROPERTIES,1)
710 -         UP(i,1:3) = Paraffin_Combinations(Usable(j,1),i).*Paraffin_PROPERTIES(i,1:3);
711 -     end
712 -     UsableVolumes = ((UP(:,1)*Carbon)+(UP(:,2)*Hydrogen)+(UP(:,3)*Oxygen))*Mix(1);
713 -     Um{j,1} = UsableVolumes;
714 -     for i = 1:size(Paraffin_PROPERTIES,1)
715 -
716 -         UsableVolumes(i,1) = UsableVolumes(i,1)/Paraffin_PROPERTIES(i,7);
717 -     end
718 -     UV{j,1} = UsableVolumes;

```

Figure 3.33 Main script - Volume of each substance

This produced the following table which held the volumes of each component substance within a cell, where each row is a new combination.

Table 3-11 Main script - Volume table

1	16x1 double	[23.8754;8.8...	[2.0313;1.69...	[19.9500;0.9...	12x1 double
2	16x1 double	[23.8754;8.8...	[2.0313;1.63...	[19.9500;0.9...	12x1 double
3	16x1 double	[23.8754;8.8...	[2.0313;1.56...	[19.9500;0.9...	12x1 double
4	16x1 double	[23.8754;8.8...	[2.0313;1.79...	[20.5734;0.2...	12x1 double
5	16x1 double	[23.8754;8.8...	[2.0313;1.79...	[20.5734;0.2...	12x1 double
6	16x1 double	[23.8754;8.8...	[2.0313;1.99...	[20.5734;0.2...	12x1 double
7	16x1 double	[23.8754;8.8...	[2.0313;1.99...	[20.5734;0.2...	12x1 double
8	16x1 double	[23.8754;8.8...	[2.0313;1.99...	[20.5734;0.2...	12x1 double

Using Equation 3-15 the density for each component and every combination was calculated. Next, the overall density of the fuel mixture was computed and added to the properties of each mixture.

```

770 for j = 1:5
771     for i = 1:size(Usable_Mix,1)
772         Usable_Mix(i,11+j) = sum(cell2mat(Um(i,j)))/sum(cell2mat(UV(i,j)));
773     end
774 end
775 for i = 1:size(Usable_Mix,1)
776     Usable_Mix(i,11) = Usable_Mix(i,5)/((sum(cell2mat(Um(i,1)))/Usable_Mix(i,12))+(sum(cel
777         %Density of a mixture = m/(m/d+m/d)
778     end
779 Usable_Mix(:,17) = Usable_Mix(:,5)./Usable_Mix(:,11);
780 Usable_Mix(:,18) = (1:size(Usable_Mix,1));

```

Figure 3.34 Main script - Density of Mixture

Using Equation 3-14 the volume of the fuel mixture can be calculated using the mass of the fuel and the newly calculated density. ID numbers are then given to the mixtures for user selection.

With all fuels mixed, final checks are necessary to ensure that each mixture produced is legal. For this check, the mixture needs 1%, 18%, and 35% volumes calculated as well as 2.7% mass. Additionally, for aromatics, naphthenes, and olefins the carbon groups need to be identified by matrix coordinates this can be seen in the full script in appendix C. Each combination is then tested in 10 ways:

- MON > 85
- MON < 90
- Benzene content is < 1%(v/v)
- Oxygen content is < 2.7%(m/m)
- Aromatic content is < 35%(v/v)
- Olefin content is < 18%(v/v)
- RON > 105
- RON < 95
- HC <5% m/m are >30% mix
- Component carbon concentration

If any of these checks are flagged as illegal, the ID number of the combinations is noted along with the corresponding reason number.

Once each combination is checked, any non-legal combinations are removed. If every combination is removed, the user is notified using a dialog output from MATLAB that indicates why the combinations were deleted and the script terminates (Figure 3.37).

```

if UV(i,2)(3,1) > Benzene
    Del(j,1)=i;
    j=j+1;
elseif Usable_Mix(i, 9) < MON_min
    Del(j,1)=i;
    Del(j,2)=1;
    j=j+1;
elseif Usable_Mix(i, 9) > MON_max
    Del(j,1)=i;
    Del(j,2)=2;
    j=j+1;
elseif Usable_Mix(i, 8) < RON_min
    Del(j,1)=i;
    Del(j,2)=3;
    j=j+1;
elseif Usable_Mix(i,3)*oxygen > mOxygenfuel
    Del(j,1)=i;
    Del(j,2)=4;
    j=j+1;
elseif Usable_Mix(i,13) > VolumeA
    Del(j,1)=i;
    Del(j,2)=5;
    j=j+1;
elseif Usable_Mix(i,15) > VolumeO
    Del(j,1)=i;
    Del(j,2)=6;
    j=j+1;
elseif Usable_Mix(i, 8) < RON_Rmin
    Del(j,1)=i;
    Del(j,2)=7;
    j=j+1;
elseif Usable_Mix(i, 8) < RON_Rmax
    Del(j,1)=i;
    Del(j,2)=8;
    j=j+1;

```

Figure 3.35 Main script - Final checks

```

826- if isempty(Del) == 0
827-     Usable_Mix(Del, i) = [];
828-     UV(Del, i) = [];
829-     Um(Del, i) = [];
830- else
831- end
832- if isempty(Usable_Mix) == 1
833-     tf(1) = any(Del(:,2)==1);
834-     tf(2) = any(Del(:,2)==2);
835-     tf(3) = any(Del(:,2)==3);
836-     tf(4) = any(Del(:,2)==4);
837-     tf(5) = any(Del(:,2)==5);
838-     tf(6) = any(Del(:,2)==6);
839-     tf(7) = any(Del(:,2)==7);
840-     tf(8) = any(Del(:,2)==8);
841-
842-     error = erroridg(['No combinations. 1 indicates issue.', sprintf('\n') ' | MON below reg min ', num2str(tf(1)), sprintf('\n') ' | MON abo
843-     uiwait (error);
844-
845-     return
846- end

```

Figure 3.36 Main script - Final checks removed

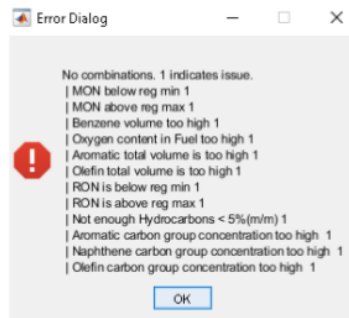


Figure 3.37 Main script - No remaining mixtures error

Following these checks, the remaining combination are displayed to the user in a table (Table 3-12).

Table 3-12 Main script - Final mixtures

	C	H	O	H ₂ O	mFuel	Enthalpy of Formation	hFuel_g	RON	MON	LI-W	p	p Par	p Aro	p Nap	p Ole	p Oxy	Total Volume
1	5.8273	10.7745	0.1400	5.3873	82.9416	129.6318	3.4682e+03	97.2136	86.9532	40.6376	0.7038	0.8025	0.8668	0.7703	0.8415	0.7887	117.8387
2	5.8283	10.7725	0.1400	5.3863	82.9276	129.5996	3.4689e+03	97.2241	86.9587	40.6379	0.7038	0.8025	0.8668	0.7703	0.8415	0.7887	117.8211
3	5.8293	10.7705	0.1400	5.3853	82.9136	129.5675	3.4696e+03	97.2346	86.9642	40.6380	0.7038	0.8025	0.8668	0.7704	0.8415	0.7887	117.8035
4	5.8241	10.7681	0.1400	5.3841	82.8968	129.6261	3.4642e+03	97.2236	86.9600	40.6382	0.7037	0.8025	0.8668	0.7695	0.8404	0.7887	117.8031
5	5.8241	10.7681	0.1400	5.3841	82.8968	129.6261	3.4642e+03	97.2236	86.9600	40.6382	0.7037	0.8025	0.8668	0.7695	0.8404	0.7887	117.8031
6	5.8241	10.7681	0.1400	5.3841	82.8968	129.5870	3.4642e+03	97.2416	86.9620	40.6387	0.7037	0.8025	0.8668	0.7690	0.8404	0.7887	117.8069
7	5.8241	10.7681	0.1400	5.3841	82.8968	129.5825	3.4642e+03	97.2401	86.9593	40.6387	0.7037	0.8025	0.8668	0.7691	0.8404	0.7887	117.8062
8	5.8241	10.7681	0.1400	5.3841	82.8968	129.5734	3.4642e+03	97.2551	87.0033	40.6388	0.7037	0.8025	0.8668	0.7693	0.8404	0.7887	117.8048
9	5.8241	10.7681	0.1400	5.3841	82.8968	129.5749	3.4642e+03	97.2641	87.0153	40.6388	0.7037	0.8025	0.8668	0.7699	0.8404	0.7887	117.7987
10	5.8236	10.7671	0.1400	5.3836	82.8898	129.5593	3.4639e+03	97.2724	87.0220	40.6388	0.7037	0.8025	0.8668	0.7702	0.8404	0.7887	117.7888
11	5.8226	10.7651	0.1400	5.3826	82.8758	129.5271	3.4633e+03	97.2829	87.0275	40.6389	0.7037	0.8025	0.8668	0.7702	0.8404	0.7887	117.7702
12	5.8171	10.7541	0.1400	5.3771	82.7988	129.4347	3.4599e+03	97.1954	86.9565	40.6363	0.7036	0.8025	0.8668	0.7703	0.8404	0.7887	117.6794
13	5.8166	10.7513	0.1400	5.3766	82.7790	129.3470	3.4581e+03	97.2326	86.9620	40.6369	0.7038	0.8024	0.8668	0.7703	0.8404	0.7887	117.6222
14	5.8216	10.7631	0.1400	5.3816	82.8618	129.4950	3.4627e+03	97.2834	87.0330	40.6390	0.7037	0.8025	0.8668	0.7703	0.8404	0.7887	117.7517
15	5.8161	10.7521	0.1400	5.3761	82.7648	129.4025	3.4583e+03	97.1959	86.9620	40.6364	0.7036	0.8025	0.8668	0.7703	0.8404	0.7887	117.6598
16	5.8146	10.7493	0.1400	5.3746	82.7650	129.3157	3.4585e+03	97.2431	86.9675	40.6400	0.7038	0.8024	0.8668	0.7703	0.8404	0.7887	117.6037
17	5.8151	10.7501	0.1400	5.3751	82.7708	129.3704	3.4587e+03	97.2564	86.9675	40.6365	0.7036	0.8025	0.8668	0.7704	0.8404	0.7887	117.6413
18	5.8138	10.7473	0.1400	5.3738	82.7510	129.2836	3.4579e+03	97.2836	86.9730	40.6401	0.7038	0.8024	0.8668	0.7704	0.8404	0.7887	117.5852
19	5.8395	10.7969	0.1400	5.3895	83.1124	130.0995	3.4735e+03	97.2143	86.9141	40.7011	0.7040	0.8025	0.8668	0.7702	0.8415	0.7886	118.0579
20	5.8395	10.7969	0.1400	5.3895	83.0984	130.0673	3.4729e+03	97.2248	86.9196	40.7012	0.7040	0.8025	0.8668	0.7703	0.8415	0.7886	118.0293
21	5.8375	10.7949	0.1400	5.3875	83.0844	130.0352	3.4723e+03	97.2353	86.9251	40.7013	0.7040	0.8025	0.8668	0.7703	0.8415	0.7886	118.0206
22	5.8395	10.7928	0.1400	5.3895	83.0704	130.0030	3.4717e+03	97.2458	86.9306	40.7013	0.7040	0.8025	0.8668	0.7704	0.8415	0.7886	118.0022
23	5.8393	10.7905	0.1400	5.3893	83.0536	130.0616	3.4709e+03	97.2340	86.9524	40.7016	0.7038	0.8025	0.8668	0.7695	0.8404	0.7886	118.0028
24	5.8393	10.7905	0.1400	5.3893	83.0536	130.0616	3.4709e+03	97.2340	86.9524	40.7016	0.7038	0.8025	0.8668	0.7695	0.8404	0.7886	118.0028
25	5.8393	10.7905	0.1400	5.3893	83.0536	130.0226	3.4709e+03	97.2528	86.9594	40.7020	0.7038	0.8025	0.8668	0.7690	0.8404	0.7886	118.0066
26	5.8393	10.7905	0.1400	5.3893	83.0536	130.0226	3.4709e+03	97.2528	86.9594	40.7020	0.7038	0.8025	0.8668	0.7690	0.8404	0.7886	118.0066

The user can look through the available fuels and select one to receive mixing volumes by inputting the fuel ID (row).

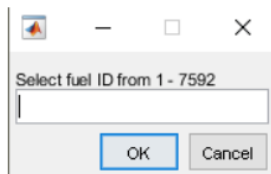


Figure 3.38 Main script - Mixture selection

The volumes of each hydrocarbon within the mixture are calculated by multiplying volumes computed in Figure 3.34 by the Volume the user wants to mix. The volume is set to 22 as a default, as this is the tank volume limit. This value is then divided by the volume of the mixture located in the properties table.

$$V_{\text{substance}} = V_n * \left(\frac{V_{\text{required}}}{V_{\text{Mixture}}} \right)$$

Equation 3-16 Volume Required

Where V_n is the volume of the substance from a 1kmol of the mixture, V_{required} is 22 and V_{mixture} is the volume of 1 kmol of the mixture.

These values in litres are displayed to the user using a table that is positioned above the Table 3-12.

Table 3-13 Main script - Volume required per substance in liters

	3-DIMETHYLPENTANE	Propane	n-Butane	iso-Butane	iso-Pentane	iso-Hexane	2,2-DIMETHYLBUTANE	2,3-DIMETHYLBUTANE	3-METHYLPENTANE	n-Heptane	2,4-DIMETHYLPENTANE
	0.6018	0.0105	0.0838	0.5050	0.4163	0.7869	0.0237	0.3105	0.1536	1.2230	

	C	H	O	H2O	mFuel	Enthalpy of Formation	hffNet_g	RON	MON	LHV	p	p Par	p A
1	5.8273	10.7745	0.1400	5.3873	62.9416	129.6318	3.4662e+03	97.2136	86.9532	40.6378	0.7039	0.6025	

The user is then asked whether they would like to change the mixture via a dialogue box (Figure 3.39). If they select yes, the user is returned to Figure 3.38, otherwise the script ends.

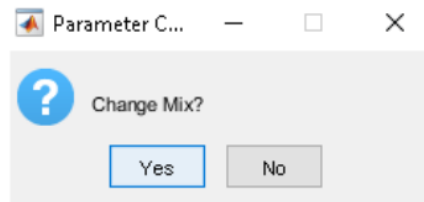


Figure 3.39 Main script - select another fuel

3.5 Test outlines

The script was tested to ensure that the regulations imposed by MOTO GP were being followed. A set of situations were manufactured in order to test blatantly illegal mixtures for each regulation monitored. For all tests the mixture % will be set as shown in Table 3-14, unless specified otherwise.

Table 3-14 Mole fraction of mixture

Paraffin	Aromatic	Naphthene	Olefin	Oxygenate
32%	30%	5%	19%	14%

3.5.1 RON value must be between 95 and 105:

1. RON value is set within range
2. RON value is set to value within range that cannot be achieved
3. RON set below 95
4. RON set above 105
5. RON set to 95
6. RON set to 105

For these tests, the script was assessed on whether the user input is allowed and then if so, what values RON are produced from the final mixtures.

3.5.2 MON value must be between 85 and 90:

For this assumption, the script was assessed alongside the RON tests, by checking all final mixture results have a MON between the values of 85 and 90.

3.5.3 Oxygen %(m/m) must be no higher than %2.7:

- Oxygenates set to %14 of mixture
- Oxygenates set to %15 of mixture

In order to increase the amount of oxygen within the mixture, the oxygenates need to make up a greater portion of a mole. This assumption was tested through the script, informing the user once the %(m/m) became too high. The amount is also checked manually to ensure the error is correct. Fuels that the script deems legal will also be checked to make sure no error has been made.

3.5.4 Benzene %(v/v) must be no higher than %1:

- Benzene will be set at set to %1.5 of aromatics component
- Benzene will be set at set to %10 of aromatics component

Benzene is a portion of the aromatic component and therefore the composition of the component is modified to force values below and above the threshold. This means that the script can allow some tests and stop others. All results were manually checked to ensure that the script acts appropriately.

3.5.5 Olefins %(v/v) must be no higher than %18:

- Olefins set to %20 of mixture
- Olefins set to %21 of mixture

The olefins mole fraction was modified to produce %(v/v) both above and below the threshold. The script was checked to see whether it was able to pick out mixtures breaking the regulation. All results were manually checked.

3.5.6 Aromatics %(v/v) must be no higher than %35:

- Aromatics set to %38 of mixture
- Aromatics set to %39 of mixture

The aromatics mole fraction was modified to produce %(v/v) both above and below the threshold. The script was checked to see whether it was able to pick out mixtures that cause the regulations to be broken. All results will then be checked manually.

3.5.7 Hydrocarbons with concentrations less than 5%m/m must be 30% of mixture

- Cap set to 30%
- Cap set to 20%

The MOTO GP regulations (FIM, 2020) states that this regulation is monitored by $A=100-B-C$, where A is hydrocarbons with $<5\%m/m$, B is concentration of oxygenates present in the fuel $\%m/m$, and C is the hydrocarbons with $>5\%m/m$. For this test, the cap was set at 20% and 30%.

3.5.8 Component concentration for each carbon group must be below value

- Standard mixture with no concentrations too high
- Decreasing naphene concentration upper limit
- Decreasing olefin concentration upper limit
- Decreasing aromatics Concentration upper limit

The upper limit for one carbon group per component was reduced to trigger mixtures that were non regulation friendly. This verified whether or not the script could identify all 3 groups breaking the regulations and identifying the component causing the infraction.

4 Results & Analysis

4.1 RON value must be between 95 and 105 & MON value must be between 85 and 90:

4.1.1 RON value is set within range

Table 4-1 RON value is set within range

RON input	Input response	Final mixture results RON	Final mixture results MON
97	Allowed	97 – 97.2640	87.0075 - 86.5742

Inputting 97 as the RON value (within the regulation range) flagged up no errors. This means that the script continued and was running as expected. Once the script had finished creating mixtures the maximum and minimum RON values were pulled from the table, resulting in the range 97 – 97.264. Therefore, the RON value does not drop below the users input value and does not increase too far above. The maximum and minimum MON values were then pulled from the mixtures table resulting in a range of 87.0075 - 86.5742. This range also fell within the MON regulations. These results demonstrate that when input with a regulation friendly RON which can be mixed, the script acts as it should and is therefore successful.

4.1.2 RON value is set to value within range that cannot be achieved

Table 4-2 RON value is set to value within range that cannot be achieved

RON input	Input response	Final mixture results RON	Final mixture results MON
104	Error	N/A	N/A

When inputting a RON value that is within the regulations, but cannot be achieved with the component mixtures, an error is flagged. Using 104 as the input, the script returned the error shown in Figure 4.1. As expected, this caused the script to end and no mixtures were made. Thus, no regulations were broken, and the results of the test were successful.

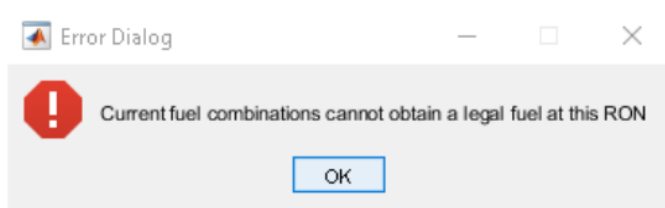


Figure 4.1 Results - error 1

4.1.3 RON set below 95

Table 4-3 RON set below 95

RON input	Input response	Final mixture results RON	Final mixture results MON
94.9	Error	N/A	N/A

When inputting 94.9 as the required output RON number the script produces the error shown in Figure 4.2. As this number is out of the regulation range for RON, all the mixtures were deleted by the end of the script. The script acted as expected in this case.

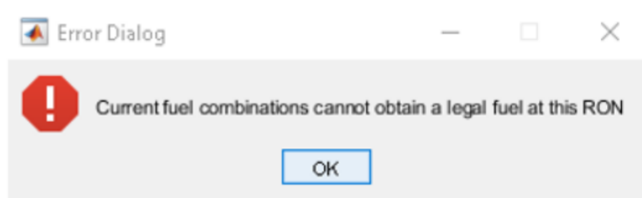


Figure 4.2 Results - error 2

4.1.4 RON set above 105

Table 4-4 RON set above 105

RON input	Input response	Final mixture results RON	Final mixture results MON
105.1	Error	N/A	N/A

When inputting 105.1 as the required output RON, the results were identical to the results from section 4.1.3. This result was expected and shows that the script prevents the user from selecting a value higher than the RON regulation range.

4.1.5 RON set to 95

Table 4-5 RON set to 95

RON input	Input response	Final mixture results RON	Final mixture results MON
95	Allowed	95.0001 - 95.3274	85 - 85.4206

By setting the RON value to 95 (minimum regulations value) no errors were flagged, meaning that the script attempted to create fuel mixtures as expected. Once the mixtures were complete, the maximum and minimum RON were selected from the mixtures table providing a range of 95.0001 - 95.3274. Therefore, the RON value does not drop below the regulation minimum and provides a small range for RON to the user. The maximum and minimum MON values were then retrieved producing a range of 85 - 85.4206. These values were located

within the regulation range. Thus, the script was running as expected across all aspects of this test.

4.1.6 RON set to 105

Table 4-6 RON set to 105

RON input	Input response	Final mixture results RON	Final mixture results MON
105	Error	N/A	N/A

When inputting 105 as the RON value for the mixture, the error displayed in Figure 4.1 is displayed to the user. The component mixtures at the mole fraction display in Table 3-14 did not allow for this RON value. However, the script did allow for the input.

4.2 Oxygen %(m/m) must be no higher than %2.7:

4.2.1 Oxygenates set to %14 of mixture

Table 4-7 Oxygenates set to 14%

Paraffin	Aromatic	Naphthene	Olefin	Oxygenate
32%	30%	5%	19%	14%
RON input		Produce mixtures?		Error message
97		Yes		None

With the oxygenates mole fraction set to 14%, the script ran until the end and produced final mixtures. This would suggest that at 14%, the %(m/m) of the mixtures produced are below 2.7% or at least some are. This was verified by checking that the total mass of oxygen atoms from a handful of mixtures were in fact below 2.7% of the overall mixtures mass. Each mixture in this case contained 0.14 of oxygen. Therefore, resulting in a mass of 2.24kg of the mixture. When comparing this to 2.7 %(m/m) of the top 5 mixtures as shown in Figure 4.3, it can clearly be seen that the oxygen %(m/m) is very close to 2.7%. As these values are below the threshold, the script is acting as expected and therefore this test was a success. As these values were so close to the threshold, it is expected that a 1% increase would result in no mixtures being produced.

	1
1	2.2480
2	2.2480
3	2.2480
4	2.2478
5	2.2451

Figure 4.3 2.7% mFuel

4.2.2 Oxygenates set to %15 of mixture

Table 4-8 Oxygenates set to 15%

Paraffin	Aromatic	Naphthene	Olefin	Oxygenate
32%	30%	5%	18%	15%
RON input	Produce mixtures?		Error message	
97	No		Correct	

The oxygenates mole fraction was then increased to 15% which was expected to produce mixtures with larger %(m/m) of oxygen, and therefore not produce mixtures. With larger oxygen proportions, no mixtures were produced. The error message displayed in Figure 4.4 was displayed to the user.

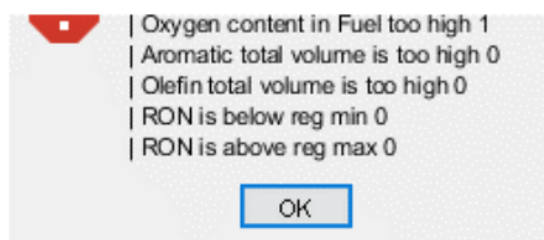


Figure 4.4 Oxygen content too high

4.3 Benzene %(v/v) must be no higher than %1:

4.3.1 Benzene will be set to %1.5 of aromatics component

Table 4-9 Benzene will be set at set to %1.5 of aromatics component

RON input	Benzene mole fraction	Produce mixtures?	Error message
97	1.5%	Yes	None

Locking the benzene mole fraction within the aromatic's component to 1.5% minimum and maximum range lead to full completion of the script as well as producing a list of legal substances. This was manually checked using benzene volume in litres and compared to 1% of the overall mixture volume. In this instance, benzene always has a volume of 0.2677 litres. Taking into account 1% of the top 5 mixtures (Figure 4.5), the lowest volume was 118.0735 litres (1.180735) which is larger than the benzene content. Thus, the script acted as expected and the test was successful.

17
Total Volume
118.1939
118.1924
118.1874
118.1765
118.0735
118.0735

Figure 4.5 Total mixture volume - Benzene

4.3.2 Benzene will be set to %10 of aromatics component

Table 4-10 Benzene will be set at set to %1.5 of aromatics component

RON input	Benzene mole fraction	Produce mixtures?	Error message
97	10%	Yes	None

The benzene mole fraction was increased and locked within the aromatic's component to 10% of the minimum and maximum range. The script was unable to produce any mixtures as the benzene %(v/v) was too high. Figure 4.6 shows the error message which was displayed to the user. This suggests that the test was successful as the script acted as expected and correctly identified the problem with the mixtures.

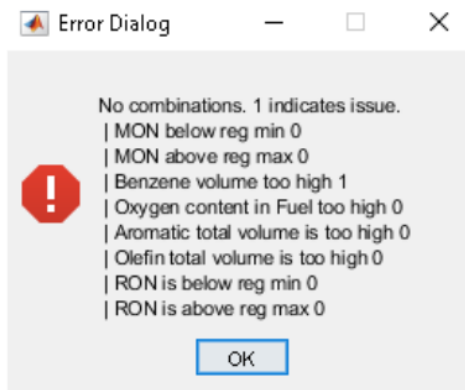


Figure 4.6 Benzene volume too high

4.4 Olefins %(v/v) must be no higher than %18:

4.4.1 Olefins set to %20 of mixture

Table 4-11 Olefins set to %20 of mixture

Paraffin	Aromatic	Naphthene	Olefin	Oxygenate
32%	30%	5%	20%	13%
RON input	Produce mixtures?		Error message	
97	Yes		None	

The olefin mole fraction was set to 19%, and able to run through the entire script to produce mixtures. This would suggest that at 19, at least some of the %(v/v) of the mixtures produced are below 18%. This was verified by checking the volume of the olefins against the entire mixtures volume to ensure the %(v/v) is below 18%. In the top 5 mixtures, the volume of olefins is 21.8750 litres (Figure 4.7). In comparison to 18% of the mixture with the lowest overall volume within the top 5 (121.5609 litres, 21.88L), the olefins %(v/v) are just below the threshold. Therefore, the script is acting as expected, making this test a success.

4
[21.8750;0]
[21.8750;0]
[21.8750;0]
[21.8750;0]
[21.8750;0]

Figure 4.7 Volume of olefins

Total Volume
121.5609
121.5609
121.5609
121.5647
121.5639

Figure 4.8 Total volume

4.4.2 Olefins set to %21 of mixture

Table 4-12 Olefins set to %21 of mixture

Paraffin	Aromatic	Naphthene	Olefin	Oxygenate
32%	30%	5%	21%	12%
RON input		Produce mixtures?		Error message
97		No		Correct

The olefin mole fraction was then increased to 21%. This should have produced mixtures with a larger %(v/v) of olefins than the regulations permit. With larger olefin concentrations, the script produced no mixtures and the error message in Figure 4.9 was displayed to the user. As 20% is so close to the threshold, these results indicate that the script is working as expected.

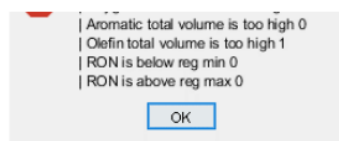


Figure 4.9 Olefin volume too high error

4.5 Aromatics %(v/v) must be no higher than %36:

4.5.1 Aromatics set to %38 of mixture

Table 4-13 Aromatics set to %38 of mixture

Paraffin	Aromatic	Naphthene	Olefin	Oxygenate
30%	38%	5%	19%	8%
RON input		Produce mixtures?		Error message
99.5		Yes		None

With the Aromatics set to 38% of the overall fuel mixture (mole fraction), the script was able to complete with a list of viable fuel options. This would suggest that at 38%, the volume of aromatics is below 35% of the overall volume. This was verified manually by calculating the volume of aromatics on a selection of mixture and comparing it to the volume of the entire mixture as shown in section 4.4.1.

4.5.2 Aromatics set to %39 of mixture

Table 4-14 Aromatics set to %39 of mixture

Paraffin	Aromatic	Naphthene	Olefin	Oxygenate
29%	39%	5%	19%	8%
RON input		Produce mixtures?		Error message
99.5		No		Correct

Increasing the aromatics mole fraction to 39% of the mixture and therefore increasing the %(v/v) of aromatics resulted in no mixtures being created. As the volume % had been increased past the maximum point, the script acted as expected and the error message in Figure 4.10 was displayed to the user.

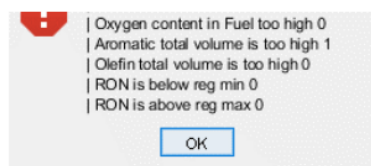


Figure 4.10 Aromatic volume too high error

4.6 Hydrocarbons with concentrations less than 5%*m/m* must be 30% of mixture

4.6.1 Cap set to 30%

Table 4-15 Hydrocarbons with concentrations less than 5%*m/m* must be 30% of mixture

Regulation limit	RON input	Produce mixtures?	Error message
30%	97	No	Correct

With the lower limit on hydrocarbons containing only hydrogen and carbon present at concentrations less than 5%*m/m* set to 30% as stated in the regulations, no mixtures were produced, and the error message in Figure 4.11 was displayed. This suggested that in these circumstances, the script was working correctly. The available mixtures did not meet the regulations. Although no mixtures were produced the script acted as it should have and therefore this test was successful.

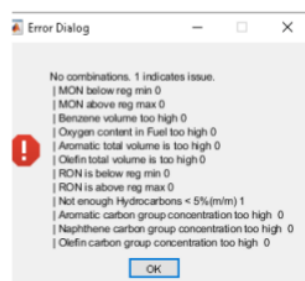


Figure 4.11 Not enough hydrocarbons <5%*m/m* error

4.6.2 Cap set to 20%

Table 4-16 Hydrocarbons with concentrations less than 5%*m/m* must be 20% of mixture

Regulation limit	RON input	Produce mixtures?	Error message
20%	97	Yes	None

As the previous test resulted in no mixtures being produced, the lower limit regulation was temporarily reduced to 20%. This resulted in mixtures being produced with no errors, verifying that the script allows mixtures providing this regulation is met. Therefore, the script responded as expected and therefore this test was also a success.

4.7 Component concentration for each carbon group must be below value

4.7.1 Standard mixture with no concentrations too high

Table 4-17 Standard mixture with no concentrations too high

C group increased	RON input	Produce mixtures?	Error message
None	97	Yes	None

Setting up the mixtures with standard composition resulted in the script running without errors. As the concentrations met the regulations, the script was expected to run in this way. The following 3 tests will determine whether the script is monitoring this correctly.

4.7.2 Decreasing naphene concentration C5 upper limit

Table 4-18 Decreasing naphene concentration C5 upper limit

C group increased	RON input	Produce mixtures?	Error message
5	97	No	Correct

The C5 hydrocarbon maximum concentration for the Napthene component was decreased so that the regulation was broken (set to 0.1%*m/m* max). As a result, no mixtures were produced, and the user was presented with the error message shown in Figure 4.12. Therefore, the script was running as expected and this test was a success.

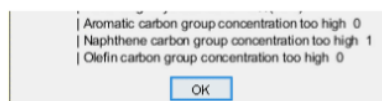


Figure 4.12 Naphthene carbon group concentration too high error

4.7.3 Decreasing olefin concentration C6 upper limit

Table 4-19 Decreasing olefin concentration C6 upper limit

C group increased	RON input	Produce mixtures?	Error message
6	97	No	Correct

The maximum concentration of C6 hydrocarbons within the Olefin component was decreased to 0.1%*m/m*. The script was therefore unable to produce regulation friendly mixtures and was executed as expected and therefore the test was successful. Figure 4.13 shows the error message displayed to the user.



Figure 4.13 Olefin carbon group concentration too high error

4.7.4 Decreasing aromatics Concentration C7 upper limit

Table 4-20 4.7.4 Decreasing aromatics concentration C7 upper limit

C group increased	RON input	Produce mixtures?	Error message
7	97	No	Correct

Finally, setting the C7 hydrocarbon group maximum concentration for aromatics to 0.1%*m/m* lead to no mixtures being produced, and the error message in Figure 4.14 was displayed to the user. As a result, this test was therefore successful.

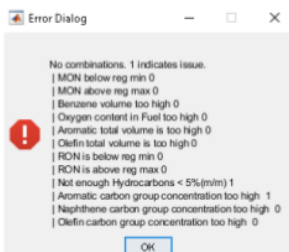


Figure 4.14 Aromatic carbon group concentration too high error

5 Recommended Further Work

Through development of MATLAB capability, it can now be seen that processes outlined in the start of the investigation are not the most optimal with a higher proficiency in MATLAB. Updating the scripts using more suitable techniques and functions could drastically improve the run time of the scripts. For example, in the script outlined in section 3.3, some of the iterations produced identical numbers in multiple columns but were written anyway. Future adjustments to the script would allow these columns to be ignored until the numbers are set to change. This would save minute amounts of time for each iteration, but this could have a large overall effect on the final run time.

Currently, the script does not accommodate for the regulations that ban certain substances or completely limit to a specific list. For example, oxygenates. This is largely a result of slight variations in fuel input names, which would bypass the script safety measure. It is possible that future research could explore informing the user if the fuel name included "diene" making it a diolefin and therefore not legal could be possible. Additionally, the oxygenates list could be added. However, substances with various names could cause issues, and they would need to be inputted accurately by the user.

The most predominant issue with the script refers to section 3.3. Specifically, the length of time it can take to check through every possible combination when there are a high number of hydrocarbons or large mole fraction ranges with small increments. The script could be redesigned so that once combinations reach over 100%, no more are looked at. Furthermore, combinations that are clearly below 100% would not be looked at. Alternatively, multiple computers could be used at once to check different combinations simultaneously. For example, if there were 100 total combinations and 5 computers, each computer would have 20 combinations to check.

6 Conclusion

Overall, it can be concluded that the script was successful. All test sections were passed by the script, indicating that only mixtures that are FIM regulation friendly will be produced. This does however rely on the competence of the user, as banning of specific substances is not implemented within the script. From the test section however, it can be verified that the script is actively monitoring all of the regulations that are listed within section 3.1. By only modifying the mole fraction ranges on the oxygenates table shown in Table 3-7 no banned oxygenates will be used within the mixtures.

The substances used within the gasoline mixtures are fully interchangeable, allowing for the user to add whatever substances are required. Additionally, the user is able to manually input the mole fraction range for each substance. This however is limited as too many substances with too high a range will take prolonged periods of time to calculate.

Finally, the script was able to produce an output for the user. The output showed the user how to create their specified mixture through volume breakdowns of each individual substance within it. Volumes are displayed to the user in litres and by modifying the script, the user can select the total volume created. This by default is set to 22 liters (Moto GP tank capacity).

7 References

Agarwal, A. K., Karare, H. & Dhar, A., 2014. Combustion, performance, emissions and particulate characterization of a methanol–gasoline blend (gasohol) fuelled medium duty spark ignition transportation engine. *Fuel Processing Technology*, Volume 121, pp. 16-24.

Al-Hasan, M., 2003. Effect of ethanol–unleaded gasoline blendson engine performance and exhaust emission. *Energy Conversion and Management*, 44(11), pp. 1547-1561.

ARASU, V. A., RAJA, A. S. & SORNAKUMAR , T., 2015. EFFECT OF GASOLINE - ETHANOL BLENDS ON PERFORMANCE AND EMISSION CHARACTERISTICS OF A SINGLE CYLINDER AIR COOLED MOTOR BIKE SI ENGINE. *Journal of Engineering Science and Technology*, 10(12), pp. 1540 -1552.

ARGUS, 2019. *Argus Russian Petrochemicals*. [Online]
Available at: <https://www.argusmedia.com/-/media/Files/methodology/argus-russian-petrochemicals.ashx>
[Accessed 22 04 2020].

AUS-e-TUTE, 2018. *Mass-Mole Calculations Chemistry Tutorial*. [Online]
Available at: <https://www.usetute.com.au/massmole.html>
[Accessed 21 04 2020].

Balaban, A. T., Kier, L. B. & Joshi, N., 1992. Structure-prroperty Analysis of Octane numbers for hydrocarbons. *match*, Volume 28, pp. 13-27.

Bayraktar, H., 2005. Experimental and theoretical investigation of using gasoline–ethanol blends in spark-ignition engines. *Renewable Energy*, 30(11), pp. 1733 - 1747.

Britannica, 2015. <https://www.britannica.com/science/butane>. [Online]
Available at: <https://www.britannica.com/science/butane>
[Accessed 01 04 2020].

Burroughs, D., 2015. *Growing Chorus of Complaints on Chemicals in Gasoline*. [Online]
Available at: <https://morningconsult.com/2015/04/22/growing-chorus-of-complaints-on-chemicals-in-gasoline/>
[Accessed 02 05 2020].

CarTech Books, n.d. *Engine Tuning Basics*. [Online]
Available at: <https://www.carttechbooks.com/techtips/engine-tuning-basics/>
[Accessed 22 04 2020].

Chemical Book, n.d. *2-Methylbutane*. [Online]
Available at: https://www.chemicalbook.com/ChemicalProductProperty_EN_CB5854378.htm
[Accessed 21 04 2020].

ChemSpider, n.d. *Search ChemSpider*. [Online]
Available at: <http://www.chemspider.com/>
[Accessed 22 04 2020].

CNBC International, 2014. *Formula One Fuel Explained I One Second in... F1 I CNBC International*. s.l.:s.n.

CONCOA, n.d. *Propane (C₃H₈)*. [Online]
Available at: https://www.concoa.com/propane_properties.html
[Accessed 21 04 2020].

Demirbas, A. et al., 2015. Octane Rating of Gasoline and Octane Booster. *Petroleum Science and Technology*, 33(11), pp. 1190 - 1197.

Dixon, M. & Halderman, J., 2012. *Changes in Gasoline Chemistry*. [Online]
Available at:
https://opensiuc.lib.siu.edu/auto_pres/25/?utm_source=opensiuc.lib.siu.edu%2Fauto_pres%2F25&utm_medium=PDF&utm_campaign=PDFCoverPages
[Accessed 20 04 2020].

El-Naggar, A. Y. & Al Majthoub, M. M., 2013. STUDY THE TOXIC EFFECTS OF AROMATIC COMPOUNDS. *International Journal of Chemical Sciences*, 11(1), pp. 106 - 120.

Elwardany, A. et al., 2016. *Modeling of Heating and Evaporation of FACE I Gasoline Fuel and its Surrogates*. s.l., s.n.

EnggCyclopedia, n.d. *Octane Number*. [Online]
Available at: <https://www.enggcyclopedia.com/2011/01/octane-number/>
[Accessed 22 04 2020].

Ester, S., 2018. *Paraffins*. [Online]
Available at: <https://www.e-education.psu.edu/fsc432/content/paraffins>
[Accessed 01 04 2020].

Eyidogan, M., Ozsezen, A. N., Canakci, M. & Turkan, A., 2010. Impact of alcohol–gasoline fuel blends on the performance and combustion characteristics of an SI engine. *FUEL*, 89(10), pp. 2713-2720.

FIA, 2020. *FIA FORMULA ONE WORLD CHAMPIONSHIP REGULATIONS*. [Online]
Available at: <https://www.fia.com/regulation/category/110>
[Accessed 19 04 2020].

FIM, 2020. *2020 FIM GP World Championship Regulations*. [Online]
Available at: <http://www.fim-live.com/en/sport/regulations-and-documents/grand-prix/>
[Accessed 19 04 2020].

Groote, S. D., 2006. *Fuel*. [Online]
Available at: <https://www.f1technical.net/articles/19>
[Accessed 20 04 2020].

Hamdan, M. A. & Sunaih, T. A., 2002. Improvement of locally produced gasoline and studying its effects on both the performance of the engine and the environment. *Energy Conversion and Management* 4, 43(14), pp. 1811-1820.

Houghton-alico, D., 1982. *Alcohol Fuels: Policies, Production, And Potential*. 1st ed. Colorado: Westview Press.

Lee, C. et al., 2017. Autoignition characteristics of oxygenated gasolines. *Combustion and Flame*, Volume 186, pp. 114 - 128.

McAllister, S., Chen, J.-Y. & Fernandes-Pello, A. C., 2011. *Fundamentals of Combustion Processes*. 1 ed. New York: Springer-Verlag.

Najafi, G. et al., 2015. Optimization of performance and exhaust emission parameters of a SI (spark ignition) engine with gasoline–ethanol blended fuels using response surface methodology. *Energy*, Volume 90 part 2, pp. 1815-1829.

Nazarova, G. Y., Ivashkina, E. N., Stebeneva, V. I. & Seytenova, G. Z., 2016. Effect of group composition of the vacuum distillate from heavy. *Procedia Engineering*, Volume 152, pp. 18-24.

NIST, n.d. *Search for Species Data by Chemical Name*. [Online]
Available at: <https://webbook.nist.gov/chemistry/name-ser/>
[Accessed 22 04 2020].

Prasenjeet, G., Karlton, H. & Stephen, J., 2006. Development of a Detailed Gasoline Composition-Based Octane Model. *Industrial & Engineering Chemistry Research*, Volume 45, pp. 337 - 345.

Pulkrabek, W. W., 1997. Alternate Fuels. In: *Engineering Fundamentals of the Internal Combustion Engine*. New Jersey: Prentice Hall PTR, pp. 155 - 156.

Robinson, W. R., 2012. *Chemistry*. [Online]
Available at: <https://opentextbc.ca/chemistry/chapter/5-3-enthalpy/>
[Accessed 01 05 2020].

Ruscic, B. & Bross, D. H., 2019. *Active Thermochemical Tables (ATcT) values based on ver. 1.122g of the Thermochemical Network*. [Online]
Available at: ATcT.anl.gov
[Accessed 21 04 2020].

Shayan, S. B. et al., 2011. Impact of Methanol–Gasoline Fuel Blends on the performance and exhaust emissions of a SI engine. *International Journal of Automotive Engineering Performance and Exhaust Emissions of a SI Engine*, 1(3), pp. 219-227.

Shenghua, L., Clemente, E. R. C., Tiegang, H. & Yanjv, W., 2007. Study of spark ignition engine fueled with methanol/gasoline fuel blends. *Applied Thermal Engineering*, 27(11-12), pp. 1904-1910.

Silva, R. d. et al., 2005. Effect of additives on the antiknock properties and Reid vapor pressure of gasoline. *FUEL*, 84(7-8), pp. 951-959.

Stratiev, D., 2011. Effect of gasoline hydrocarbon composition on the properties of the blend gasoline/bioethanol. *Oxidation communications*, 34(1), pp. 183-192.

U.S. Energy Information Administration, 2020. *Gasoline explained*. [Online]
Available at: <https://www.eia.gov/energyexplained/gasoline/>
[Accessed 20 04 2020].

UWTSO, 2018. *STRATEGIC ALLIANCE WITH JESUS COLLEGE, OXFORD IN THE FIELD OF CELTIC STUDIES*. [Online]
Available at: <https://uwtsd.ac.uk/news/press-releases/press-2018/strategic-alliance-with-jesus-college-oxford-in-the-field-of-celtic-studies.html>

Westphal, G. A. et al., 2010. Ether oxygenate additives in gasoline reduce toxicity of exhausts. *Toxicology*, 268(3), pp. 198-203.

Wichers, E. & Peiser, S. H., 2015. *Atomic weight*. [Online]
Available at: <https://www.britannica.com/science/mesic-atom>
[Accessed 21 04 2020].

WoC, n.d. *Toluene - additive for racing fuels, fuel octane booster*. [Online]
Available at: <https://www.worldofchemicals.com/428/chemistry-articles/toluene-additive-for->

[acing-fuels-fuel-octane-booster.html](#)

[Accessed 22 04 2020].

Yang, J., Roth, P., Durbin, T. & Karavalakis, G., 2019. Impacts of gasoline aromatic and ethanol levels on the emissions from GDI vehicles: Part 1. Influence on regulated and gaseous toxic pollutants. *Fuel*, Volume 252, pp. 799-811.

Yanowitz, J., 2011. *Utilization of Renewable Oxygenates as Gasoline Blending Components*, s.l.: s.n.

8 Pandemic

Without the university being open the use of 006 was removed. In 006 the use of multiple computers at once greatly benefitted the study as the script in section 3.3 can be ran with multiple different mole fraction ranges allowing for the creation of a mixtures that could produce top end of the regulations RON while still in the MON regulation range. Instead fuel composition main mixture had to be adjusted to unrealistic values that don't fit real gasoline composition. This still allowed for the script to be tested but using realistic mixtures would have been preferred. Running of the script in section 3.3 sometimes took 6 days of constant running meaning the use of a personal computer impacted sleep and limited to 1 run at a time. Some regulations also had to be modified for the testing section.

Removed a place to work peacefully and made it impossible to separate working from a place that would naturally be relaxing (home). Additionally, there are far too many distractions at home and a study area such as the library makes it so much easier to get work done. Personal relations within the house also made the home a very negative place for work to be completed and resulted in thoughts being elsewhere as no separation could be made between the problem.

Work that was sent by email the night the university was due to close ended up becoming corrupt. This meant I relied on the IT staff to resend the documents which took well over 2 weeks for a reply meaning one of the 6-day simulations had to be ran again as well as a slightly shorter one (these were complete just before the work was resent to me).

The situation in general is hard to remove from your head. With parents who are fairly old in comparison to me making them nearly high risk as is (and conditions that make at least one of them high risk anyway). The Pandemic also removed the ability to relax by doing activities that would reset your motivation to work again such as going outside for prolonged periods of time and other such activities.

9 Appendix

9.1 Appendix A

```
clc
clear
folder =
fileparts(which('Calculator_ChrisHopkins.xlsx'));
fullFileName = fullfile(folder,
'Calculator_ChrisHopkins.xlsx');
[status, sheetNames] = xlsfinfo(fullFileName);
numSheets = length(sheetNames);
PARAFFIN_TABLE = readtable(fullFileName, 'sheet',
'Paraffins');
Valid = 0;
%-----TABLE OF INFO-----

%-----FUEL INFORMATION VARIABLES-----
----
PARAFFIN = PARAFFIN_TABLE(:, 'Fuel');
PARAFFIN_PROPERTIES(:, 1) = PARAFFIN_TABLE(:, 'C');
PARAFFIN_PROPERTIES(:, 2) = PARAFFIN_TABLE(:, 'H');
PARAFFIN_PROPERTIES(:, 3) = PARAFFIN_TABLE(:, 'O');
PARAFFIN_PROPERTIES(:, 6) = PARAFFIN_TABLE(:, 'h');
PARAFFIN_PROPERTIES(:, 9) = PARAFFIN_TABLE(:, 'per');
aR = 78.09/20.95;
%-----FUEL INFORMATION VARIABLES-----
----
%-----MASS-----
Oxygen = 16;
Carbon = 12;
Hydrogen = 1;
Nitrogen = 14;
Air = (Oxygen*2)+(Nitrogen*2*aR);
%-----MASS-----
%-----ENTHALPY OF COMBUSTION-----
-----
hCO2 = 393.5;
hH2O_g = 241.83;
hH2O_l = 285.8;

%7 = mFuel
%8 = moles
%9 = mass Fraction
%mt = moles total
```

```

%10 = mole fraction
%13 = Carbon mix
%14 = Hydrogen mix
%15 = Oxygen mix
%16 = mFuel mix
%17 mass Fraction
PARAFFIN_PROPERTIES(:,7) =
(PARAFFIN_PROPERTIES(:,1)*Carbon)+(PARAFFIN_PROPERTIES(:,2)*Hydrogen)+(PARAFFIN_PROPERTIES(:,3)*Oxygen);
for i = 1:size(PARAFFIN_PROPERTIES,1)
PARAFFIN_PROPERTIES(i,8) =
1/PARAFFIN_PROPERTIES(i,7);
end
PARAFFIN_PROPERTIES(:,8) =
PARAFFIN_PROPERTIES(:,8).*PARAFFIN_PROPERTIES(:,9);
nt = sum(PARAFFIN_PROPERTIES(:,8))
PARAFFIN_PROPERTIES(:,10) =
PARAFFIN_PROPERTIES(:,8)/nt;
PARAFFIN_PROPERTIES(:,13) =
PARAFFIN_PROPERTIES(:,1).*PARAFFIN_PROPERTIES(:,10);
PARAFFIN_PROPERTIES(:,14) =
PARAFFIN_PROPERTIES(:,2).*PARAFFIN_PROPERTIES(:,10);
PARAFFIN_PROPERTIES(:,15) =
PARAFFIN_PROPERTIES(:,3).*PARAFFIN_PROPERTIES(:,10);
PARAFFIN_PROPERTIES(:,16) =
(PARAFFIN_PROPERTIES(:,13)*Carbon)+(PARAFFIN_PROPERTIES(:,14)*Hydrogen)+(PARAFFIN_PROPERTIES(:,15)*Oxygen)
;
mFuel = sum(PARAFFIN_PROPERTIES(:,16))
PARAFFIN_PROPERTIES(:,17) =
PARAFFIN_PROPERTIES(:,16)/mFuel;

```

9.2 Appendix B

```
clc
clear

folder =
fileparts(which('Calculator_ChrisHopkins.xlsx'));
fullFileName = fullfile(folder,
'Calculator_ChrisHopkins.xlsx');
[status, sheetNames] = xlsfinfo(fullFileName);
numSheets = length(sheetNames);
PARAFFIN_TABLE = readtable(fullFileName, 'sheet',
'Paraffins');

PARAFFIN = PARAFFIN_TABLE(:, 'Fuel');
PARAFFIN_PROPERTIES(:, 8) =
round(PARAFFIN_TABLE(:, 'min'), 4);
PARAFFIN_PROPERTIES(:, 9) =
round(PARAFFIN_TABLE(:, 'max'), 4);
PARAFFIN_PROPERTIES(:, 10) = PARAFFIN_TABLE(:, 'I');

H = size(PARAFFIN_PROPERTIES, 1);

for j = 1:H
    k=1;
    Percent=0;
    for i =
PARAFFIN_PROPERTIES(j, 8):PARAFFIN_PROPERTIES(j, 10):PA
RAFFIN_PROPERTIES(j, 9)
        Percent(k) = i;
        k=k+1;
    end
    Percent_Table{j, 1}=Percent;
end

N(1)= size(Percent_Table{1, 1}, 2);
for i = 1:H
    S(i)= size(Percent_Table{i, 1}, 2);
end
for i = 1:H-1
    N(i+1) = S(i+1)*N(i);
end
l=1;
j=1;
```

```

m=1;
OP = 0;
n=ones(1,H);
o=ones(1,H);
b=0;
for i=1:S(1)
    Combination(j,1)=Percent_Table{m,1}(1,i);
    j=j+1;
end
j=1;
m=2;
R(1) = 1;
T=1;
for i = 2:H
    R(i)=R(i-1)*S(i);
end
for b = S(1):S(1):N(H)
    for m = 2:H
        Combination(:,m)=Percent_Table{m,1}(1,o(m));
        n(m)=n(m)+1;
        if n(m) == (R(m-1)+1) && o(m) < S(m)
            o(m) = o(m)+1;
            n(m)= 1;
        elseif n(m) == (R(m-1)+1) && o(m) == S(m)
            o(m) = 1;
            n(m)= 1;
        end
    end
    CombinationTotal = sum(Combination,2);
    for i = 1:S(1)
        if CombinationTotal(i,1) == 1
            Possible(T,:) = Combination(i,:);
            T=T+1;
        end
    end
end
Possible = array2table(Possible);
save('Paraffins.mat', 'Possible');

```

9.3 Appendix C

```

%% Initial Loading
% Loading external values and setting persistent
values

```

```

clc
clear

folder =
fileparts(which('Calculator_ChrisHopkins.xlsx'));
fullFileName = fullfile(folder,
'Calculator_ChrisHopkins.xlsx');
[status, sheetNames] = xlsfinfo(fullFileName);
numSheets = length(sheetNames);
PARAFFIN_TABLE = readtable(fullFileName, 'sheet',
'Paraffins');
AROMATIC_TABLE = readtable(fullFileName, 'sheet',
'Aromatics');
NAPHTHENE_TABLE = readtable(fullFileName, 'sheet',
'Naphthenes');
OLEFIN_TABLE = readtable(fullFileName, 'sheet',
'Olefins');
OXYGENATE_TABLE = readtable(fullFileName, 'sheet',
'Oxygenates');
Paraffin_Combinations = load('Paraffins');
Paraffin_Combinations =
table2array(Paraffin_Combinations.Possible);
Aromatic_Combinations = load('Aromatics');
Aromatic_Combinations =
table2array(Aromatic_Combinations.Possible);
Naphthene_Combinations = load('Naphthenes');
Naphthene_Combinations =
table2array(Naphthene_Combinations.Possible);
Olefin_Combinations = load('Olefins');
Olefin_Combinations =
table2array(Olefin_Combinations.Possible);
Oxygenate_Combinations = load('Oxygenates');
Oxygenate_Combinations =
table2array(Oxygenate_Combinations.Possible);

PARAFFIN(1,:) = PARAFFIN_TABLE(:, 'Fuel');
Paraffin_PROPERTIES(:,1) = PARAFFIN_TABLE(:, 'C');
Paraffin_PROPERTIES(:,2) = PARAFFIN_TABLE(:, 'H');
Paraffin_PROPERTIES(:,3) = PARAFFIN_TABLE(:, 'O');
Paraffin_PROPERTIES(:,4) = PARAFFIN_TABLE(:, 'RON');
Paraffin_PROPERTIES(:,5) = PARAFFIN_TABLE(:, 'MON');
Paraffin_PROPERTIES(:,6) = PARAFFIN_TABLE(:, 'h');
Paraffin_PROPERTIES(:,7) = PARAFFIN_TABLE(:, 'p');

```

```

AROMATIC(1,:) = AROMATIC_TABLE(:, 'Fuel');
Aromatic_PROPERTIES(:,1) = AROMATIC_TABLE(:, 'C');
Aromatic_PROPERTIES(:,2) = AROMATIC_TABLE(:, 'H');
Aromatic_PROPERTIES(:,3) = AROMATIC_TABLE(:, 'O');
Aromatic_PROPERTIES(:,4) = AROMATIC_TABLE(:, 'RON');
Aromatic_PROPERTIES(:,5) = AROMATIC_TABLE(:, 'MON');
Aromatic_PROPERTIES(:,6) = AROMATIC_TABLE(:, 'h');
Aromatic_PROPERTIES(:,7) = AROMATIC_TABLE(:, 'p');

NAPHTHENE(1,:) = NAPHTHENE_TABLE(:, 'Fuel');
Naphthene_PROPERTIES(:,1) = NAPHTHENE_TABLE(:, 'C');
Naphthene_PROPERTIES(:,2) = NAPHTHENE_TABLE(:, 'H');
Naphthene_PROPERTIES(:,3) = NAPHTHENE_TABLE(:, 'O');
Naphthene_PROPERTIES(:,4) = NAPHTHENE_TABLE(:, 'RON');
Naphthene_PROPERTIES(:,5) = NAPHTHENE_TABLE(:, 'MON');
Naphthene_PROPERTIES(:,6) = NAPHTHENE_TABLE(:, 'h');
Naphthene_PROPERTIES(:,7) = NAPHTHENE_TABLE(:, 'p');

OLEFIN(1,:) = OLEFIN_TABLE(:, 'Fuel');
Olefin_PROPERTIES(:,1) = OLEFIN_TABLE(:, 'C');
Olefin_PROPERTIES(:,2) = OLEFIN_TABLE(:, 'H');
Olefin_PROPERTIES(:,3) = OLEFIN_TABLE(:, 'O');
Olefin_PROPERTIES(:,4) = OLEFIN_TABLE(:, 'RON');
Olefin_PROPERTIES(:,5) = OLEFIN_TABLE(:, 'MON');
Olefin_PROPERTIES(:,6) = OLEFIN_TABLE(:, 'h');
Olefin_PROPERTIES(:,7) = OLEFIN_TABLE(:, 'p');

OXYGENATE(1,:) = OXYGENATE_TABLE(:, 'Fuel');
Oxygenate_PROPERTIES(:,1) = OXYGENATE_TABLE(:, 'C');
Oxygenate_PROPERTIES(:,2) = OXYGENATE_TABLE(:, 'H');
Oxygenate_PROPERTIES(:,3) = OXYGENATE_TABLE(:, 'O');
Oxygenate_PROPERTIES(:,4) = OXYGENATE_TABLE(:, 'RON');
Oxygenate_PROPERTIES(:,5) = OXYGENATE_TABLE(:, 'MON');
Oxygenate_PROPERTIES(:,6) = OXYGENATE_TABLE(:, 'h');
Oxygenate_PROPERTIES(:,7) = OXYGENATE_TABLE(:, 'p');
clear PARAFFIN_TABLE
clear AROMATIC_TABLE
clear NAPHTHENE_TABLE
clear OLEFIN_TABLE
clear OXYGENATE_TABLE

%-----FUEL INFORMATION VARIABLES-----
----
%-----MASS-----

```

```

Oxygen = 16;
Carbon = 12;
Hydrogen = 1;
Nitrogen = 14;
aR = 78.09/20.95;
Air = (Oxygen*2)+(Nitrogen*2*aR);
%-----MASS-----
%-----ENTHALPY OF COMBUSTION-----
-----
hCO2 = 393.5;
hH2O_g = 241.83;
%-----ENTHALPY OF COMBUSTION-----
-----
%%
Mix = [0.32 0.30 0.05 0.19 0.14];
if sum(Mix) ~= 1
    error = errordlg('Mole fraction of mixture does
not = 100%');
    uiwait (error)
    return
end

a_Paraffin(:,1) =
(Paraffin_Combinations*Paraffin_PROPERTIES(:,4))*Mix(
1);
a_Paraffin(:,2) =
(Paraffin_Combinations*Paraffin_PROPERTIES(:,5))*Mix(
1);
a_Aromatic(:,1) =
(Aromatic_Combinations*Aromatic_PROPERTIES(:,4))*Mix(
2);
a_Aromatic(:,2) =
(Aromatic_Combinations*Aromatic_PROPERTIES(:,5))*Mix(
2);
a_Naphthene(:,1) =
(Naphthene_Combinations*Naphthene_PROPERTIES(:,4))*Mi
x(3);
a_Naphthene(:,2) =
(Naphthene_Combinations*Naphthene_PROPERTIES(:,5))*Mi
x(3);
a_Olefin(:,1) =
(Olefin_Combinations*Olefin_PROPERTIES(:,4))*Mix(4);

```

```

a_Olefin(:,2) =
(Olefin_Combinations*Olefin_PROPERTIES(:,5))*Mix(4);
a_Oxygenate(:,1) =
(Oxygenate_Combinations*Oxygenate_PROPERTIES(:,4))*Mi
x(5);
a_Oxygenate(:,2) =
(Oxygenate_Combinations*Oxygenate_PROPERTIES(:,5))*Mi
x(5);

a_Paraffin(:,3) = 1:size(Paraffin_Combinations,1);
a_Aromatic(:,3) = 1:size(Aromatic_Combinations,1);
a_Naphthene(:,3) = 1:size(Naphthene_Combinations,1);
a_Olefin(:,3) = 1:size(Olefin_Combinations,1);
a_Oxygenate(:,3) = 1:size(Oxygenate_Combinations,1);

i=0;
RangeRON = zeros(4,5);
RangeRON(4,:) = [1 1 1 1 1];
if max(round(a_Paraffin(:,1),0))-
min(round(a_Paraffin(:,1),0)) == 0
    RangeRON(1,1) = max(round(a_Paraffin(:,1),2));
    RangeRON(2,1) = min(round(a_Paraffin(:,1),2));
else
    RangeRON(1,1) = max(round(a_Paraffin(:,1),1));
    RangeRON(2,1) = min(round(a_Paraffin(:,1),1));
end

while i==0
    if (RangeRON(1,1)-RangeRON(2,1))/RangeRON(4,1)<10
        RangeRON(4,1)=RangeRON(4,1)/10;
    else
        i = i+1;
    end
end
RangeRON(3,1) = round(((RangeRON(1,1)-
RangeRON(2,1)))/RangeRON(4,1))+1;

if max(round(a_Aromatic(:,1),0))-
min(round(a_Aromatic(:,1),0)) == 0
    RangeRON(1,2) = max(round(a_Aromatic(:,1),2));
    RangeRON(2,2) = min(round(a_Aromatic(:,1),2));
else
    RangeRON(1,2) = max(round(a_Aromatic(:,1),1));
    RangeRON(2,2) = min(round(a_Aromatic(:,1),1));

```

```

end
while i==1
    if (RangeRON(1,2)-RangeRON(2,2))/RangeRON(4,2)<10
        RangeRON(4,2)=RangeRON(4,2)/10;
    else
        i = i+1;
    end
end
RangeRON(3,2) = round((RangeRON(1,2)-
RangeRON(2,2))/RangeRON(4,2))+1;
if max(round(a_Naphthene(:,1),0))-
min(round(a_Naphthene(:,1),0)) == 0
    RangeRON(1,3) = max(round(a_Naphthene(:,1),2));
    RangeRON(2,3) = min(round(a_Naphthene(:,1),2));
else
    RangeRON(1,3) = max(round(a_Naphthene(:,1),1));
    RangeRON(2,3) = min(round(a_Naphthene(:,1),1));
end
while i==2
    if (RangeRON(1,3)-RangeRON(2,3))/RangeRON(4,3)<10
        RangeRON(4,3)=RangeRON(4,3)/10;
    else
        i = i+1;
    end
end
RangeRON(3,3) = round(((RangeRON(1,3)-
RangeRON(2,3)))/RangeRON(4,3))+1;
if max(round(a_Olefin(:,1),0))-
min(round(a_Olefin(:,1),0)) == 0
    RangeRON(1,4) = max(round(a_Olefin(:,1),2));
    RangeRON(2,4) = min(round(a_Olefin(:,1),2));
else
    RangeRON(1,4) = max(round(a_Olefin(:,1),1));
    RangeRON(2,4) = min(round(a_Olefin(:,1),1));
end
while i==3
    if (RangeRON(1,4)-RangeRON(2,4))/RangeRON(4,4)<10
        RangeRON(4,4)=RangeRON(4,4)/10;
    else
        i = i+1;
    end
end
end

```

```

RangeRON(3,4) = round(((RangeRON(1,4)-
RangeRON(2,4)))/RangeRON(4,4))+1;
if max(round(a_Oxygenate(:,1),0))-
min(round(a_Oxygenate(:,1),0)) == 0
    RangeRON(1,5) = max(round(a_Oxygenate(:,1),2));
    RangeRON(2,5) = min(round(a_Oxygenate(:,1),2));
else
    RangeRON(1,5) = max(round(a_Oxygenate(:,1),1));
    RangeRON(2,5) = min(round(a_Oxygenate(:,1),1));
end
while i==4
    if (RangeRON(1,5)-RangeRON(2,5))/RangeRON(4,5)<10
        RangeRON(4,5)=RangeRON(4,5)/10;
    else
        i = i+1;
    end
end
RangeRON(3,5) = round(((RangeRON(1,5)-
RangeRON(2,5))/RangeRON(4,5))+1;
i = 0;

N=RangeRON(3,1);

RangeMON = zeros(4,5);
RangeMON(4,:) = [1 1 1 1 1];
if max(round(a_Paraffin(:,2),0))-
min(round(a_Paraffin(:,2),0)) == 0
    RangeMON(1,1) = max(round(a_Paraffin(:,2),2));
    RangeMON(2,1) = min(round(a_Paraffin(:,2),2));
else
    RangeMON(1,1) = max(round(a_Paraffin(:,2),1));
    RangeMON(2,1) = min(round(a_Paraffin(:,2),1));
end
while i==0
    if (RangeMON(1,1)-RangeMON(2,1))/RangeMON(4,1)<10
        RangeMON(4,1)=RangeMON(4,1)/10;
    else
        i = i+1;
    end
end
RangeMON(3,1) = round(((RangeMON(1,1)-
RangeMON(2,1)))/RangeMON(4,1))+1;
if max(round(a_Aromatic(:,2),0))-
min(round(a_Aromatic(:,2),0)) == 0

```

```

RangeMON(1,2) = max(round(a_Aromatic(:,2),2));
RangeMON(2,2) = min(round(a_Aromatic(:,2),2));
else
RangeMON(1,2) = max(round(a_Aromatic(:,2),1));
RangeMON(2,2) = min(round(a_Aromatic(:,2),1));
end
while i==1
if (RangeMON(1,2)-RangeMON(2,2))/RangeMON(4,2)<10
RangeMON(4,2)=RangeMON(4,2)/10;
else
i = i+1;
end
end
RangeMON(3,2) = round((RangeMON(1,2)-
RangeMON(2,2))/RangeMON(4,2))+1;
if max(round(a_Naphthene(:,2),0))-
min(round(a_Naphthene(:,2),0)) == 0
RangeMON(1,3) = max(round(a_Naphthene(:,2),2));
RangeMON(2,3) = min(round(a_Naphthene(:,2),2));
else
RangeMON(1,3) = max(round(a_Naphthene(:,2),1));
RangeMON(2,3) = min(round(a_Naphthene(:,2),1));
end
while i==2
if (RangeMON(1,3)-RangeMON(2,3))/RangeMON(4,3)<10
RangeMON(4,3)=RangeMON(4,3)/10;
else
i = i+1;
end
end
RangeMON(3,3) = round((RangeMON(1,3)-
RangeMON(2,3))/RangeMON(4,3))+1;
if max(round(a_Olefin(:,2),0))-
min(round(a_Olefin(:,2),0)) == 0
RangeMON(1,4) = max(round(a_Olefin(:,2),2));
RangeMON(2,4) = min(round(a_Olefin(:,2),2));
else
RangeMON(1,4) = max(round(a_Olefin(:,2),1));
RangeMON(2,4) = min(round(a_Olefin(:,2),1));
end

while i==3
if (RangeMON(1,4)-RangeMON(2,4))/RangeMON(4,4)<10
RangeMON(4,4)=RangeMON(4,4)/10;

```

```

        else
            i = i+1;
        end
    end
    RangeMON(3,4) = round(((RangeMON(1,4) -
    RangeMON(2,4)))/RangeMON(4,4))+1;
    if max(round(a_Oxygenate(:,2),0)) -
    min(round(a_Oxygenate(:,2),0)) == 0
        RangeMON(1,5) = max(round(a_Oxygenate(:,2),2));
        RangeMON(2,5) = min(round(a_Oxygenate(:,2),2));
    else
        RangeMON(1,5) = max(round(a_Oxygenate(:,2),1));
        RangeMON(2,5) = min(round(a_Oxygenate(:,2),1));
    end
    while i==4
        if (RangeMON(1,5)-RangeMON(2,5))/RangeMON(4,5)<10
            RangeMON(4,5)=RangeMON(4,5)/10;
        else
            i = i+1;
        end
    end
    RangeMON(3,5) = round(((RangeMON(1,5) -
    RangeMON(2,5)))/RangeMON(4,5))+1;

    for i = 1:4
        N(i+1) = RangeRON(3,i+1)*N(i);
    end
    R = [1 0 0 0 0];
    for i = 2:5
        R(i)=R(i-1)*RangeRON(3,i);
    end
    for j = 1:5
        k=1;
        Ranges=0;
        for i = RangeRON(2,j):RangeRON(4,j):RangeRON(1,j)
            Ranges(k) = i;
            k=k+1;
        end
        Ranges_Table_RON{1,j}=Ranges;
    end

%%

```

```

%---- Finds range of usable RON for each component
section to achieve input
FinalRange_RON = zeros(N(5),8);
tic
for j = 0:length(Ranges_Table_RON{1,1}):(N(5)-4)
    for i = 1:1:RangeRON(3,1)
        FinalRange_RON(i+j,1)=
Ranges_Table_RON{1,1}(1,i);
    end
end

i=1;
k=0;
for j = 0:length(Ranges_Table_RON{1,1}):(N(5)-4)

FinalRange_RON(j+1:j+length(Ranges_Table_RON{1,1}),2)
= Ranges_Table_RON{1,2}(1,i);
    k=k+length(Ranges_Table_RON{1,1});
    if k == R(1)*length(Ranges_Table_RON{1,1})
        k=0;
        if i == RangeRON(3,2)
            i=1;
        else
            i=i+1;
        end
    else
        end
end

i=1;
k=0;
for j = 0:length(Ranges_Table_RON{1,1}):(N(5)-4)

FinalRange_RON(j+1:j+length(Ranges_Table_RON{1,1}),3)
= Ranges_Table_RON{1,3}(1,i);
    k=k+length(Ranges_Table_RON{1,1});
    if k == R(2)*length(Ranges_Table_RON{1,1})
        k=0;
        if i == RangeRON(3,3)
            i=1;

```

```

        else
            i=i+1;
        end
    else
    end
end
i=1;
k=0;
for j = 0:length(Ranges_Table_RON{1,1}):(N(5)-4)

FinalRange_RON(j+1:j+length(Ranges_Table_RON{1,1}),4)
= Ranges_Table_RON{1,4}(1,i);
    k=k+length(Ranges_Table_RON{1,1});
    if k == R(3)*length(Ranges_Table_RON{1,1})
        k=0;
        if i == RangeRON(3,4)
            i=1;
        else
            i=i+1;
        end
    else
    end
end
i=1;
k=0;
for j = 0:length(Ranges_Table_RON{1,1}):(N(5)-4)

FinalRange_RON(j+1:j+length(Ranges_Table_RON{1,1}),5)
= Ranges_Table_RON{1,5}(1,i);
    k=k+length(Ranges_Table_RON{1,1});
    if k == R(4)*length(Ranges_Table_RON{1,1})
        k=0;
        if i == RangeRON(3,5)
            i=1;
        else
            i=i+1;
        end
    else
    end
end
toc
FinalRange_RON(:,8)= sum(FinalRange_RON,2);

```

```

%%
%Finds range of usable MON for each component section
to achieve regs

N=RangeMON(3,1);
tic
for i = 1:4
    N(i+1) = RangeMON(3,i+1)*N(i);
end
R = [1 0 0 0 0];
for i = 2:5
    R(i)=R(i-1)*RangeMON(3,i);
end
for j = 1:5
    k=1;
    Ranges=0;
    for i = RangeMON(2,j):RangeMON(4,j):RangeMON(1,j)
        Ranges(k) = i;
        k=k+1;
    end
    Ranges_Table_MON{1,j}=Ranges;
end

FinalRange_MON = zeros(N(5),8);
for j = 0:length(Ranges_Table_MON{1,1}):N(5)-
RangeMON(3,1))
    for i = 1:1:RangeMON(3,1)
        FinalRange_MON(i+j,1)=
Ranges_Table_MON{1,1}(1,i);
    end
end

i=1;
k=0;
for j = 0:length(Ranges_Table_MON{1,1}):N(5)-
RangeMON(3,1))

FinalRange_MON(j+1:j+length(Ranges_Table_MON{1,1}),2)
= Ranges_Table_MON{1,2}(1,i);
    k=k+length(Ranges_Table_MON{1,1});

```

```

        if k == R(1)*length(Ranges_Table_MON{1,1})
            k=0;
            if i == RangeMON(3,2)
                i=1;
            else
                i=i+1;
            end
        else
            end
        end
    end

    i=1;
    k=0;
    for j = 0:length(Ranges_Table_MON{1,1}):(N(5)-
RangeMON(3,1))

FinalRange_MON(j+1:j+length(Ranges_Table_MON{1,1}),3)
= Ranges_Table_MON{1,3}(1,i);
    k=k+length(Ranges_Table_MON{1,1});
    if k == R(2)*length(Ranges_Table_MON{1,1})
        k=0;
        if i == RangeMON(3,3)
            i=1;
        else
            i=i+1;
        end
    else
        end
    end
end
i=1;
k=0;
for j = 0:length(Ranges_Table_MON{1,1}):(N(5)-
RangeMON(3,1))

FinalRange_MON(j+1:j+length(Ranges_Table_MON{1,1}),4)
= Ranges_Table_MON{1,4}(1,i);
    k=k+length(Ranges_Table_MON{1,1});
    if k == R(3)*length(Ranges_Table_MON{1,1})
        k=0;
        if i == RangeMON(3,4)
            i=1;
        else

```

```

            i=i+1;
        end
    else
    end
end
i=1;
k=0;
for j = 0:length(Ranges_Table_MON{1,1}):(N(5)-
RangeMON(3,1))

FinalRange_MON(j+1:j+length(Ranges_Table_MON{1,1}),5)
= Ranges_Table_MON{1,5}(1,i);
    k=k+length(Ranges_Table_MON{1,1});
    if k == R(4)*length(Ranges_Table_MON{1,1})
        k=0;
        if i == RangeMON(3,5)
            i=1;
        else
            i=i+1;
        end
    else
    end
end
toc
FinalRange_MON(:,8)= sum(FinalRange_MON,2);
%%
max(a_Paraffin(:,1))+max(a_Aromatic(:,1))+max(a_Napht
hene(:,1))+max(a_Olefin(:,1))+ max(a_Oxygenate(:,1))
max(a_Paraffin(:,2))+max(a_Aromatic(:,2))+max(a_Napht
hene(:,2))+max(a_Olefin(:,2))+ max(a_Oxygenate(:,2))
min(a_Paraffin(:,2))+min(a_Aromatic(:,2))+min(a_Napht
hene(:,2))+min(a_Olefin(:,2))+min(a_Oxygenate(:,2))
U=1;
while U == 1
RON_min = str2double(inputdlg('Please input a
Research Octane Number between 95 and 105'));
if RON_min >= 95 && RON_min <= 105
    U = 0;
else
    error = errordlg('Please select a value within
the range');
    uiwait (error);
end

```

```

end
if RON_min == 105
    RON_min= 104.9
    RON_max = 105
else
    RON_max = RON_min+0.1;
end
MON_min = 85;
MON_max = 90;
RON_Rmin = 95;
RON_Rmax = 105;

FinalRange RON = FinalRange RON(FinalRange RON(:,8)
>= RON_min, :);
FinalRange RON = FinalRange RON(FinalRange RON(:,8)
<= RON_max, :);
FinalRange MON = FinalRange MON(FinalRange MON(:,8)
>= MON_min, :);
FinalRange MON = FinalRange MON(FinalRange MON(:,8)
<= MON_max, :);

if isempty(FinalRange RON) == 1 ||
isempty(FinalRange MON) == 1
    error = errorlg('Current fuel combinations
cannot obtain a legal fuel at this RON');
    uiwait (error);
    return
else
end

tic
a_Paraffin = a_Paraffin(a_Paraffin(:,1) >=
min(FinalRange RON(:,1)), :);
a_Paraffin = a_Paraffin(a_Paraffin(:,1) <=
max(FinalRange RON(:,1)+RangeRON(4,1)), :);
a_Paraffin = a_Paraffin(a_Paraffin(:,2) >=
min(FinalRange MON(:,1)), :);
a_Paraffin = a_Paraffin(a_Paraffin(:,2) <=
max(FinalRange MON(:,1)+RangeMON(4,1)), :);

a_Aromatic = a_Aromatic(a_Aromatic(:,1) >=
min(FinalRange RON(:,2)), :);
a_Aromatic = a_Aromatic(a_Aromatic(:,1) <=
max(FinalRange RON(:,2)+1), :);

```

```

a_Aromatic = a_Aromatic(a_Aromatic(:,2) >=
min(FinalRange_MON(:,2)), :);
a_Aromatic = a_Aromatic(a_Aromatic(:,2) <=
max(FinalRange_MON(:,2)+0.01), :);

a_Naphthene = a_Naphthene(a_Naphthene(:,1) >=
min(FinalRange_RON(:,3)), :);
a_Naphthene = a_Naphthene(a_Naphthene(:,1) <=
max(FinalRange_RON(:,3)+0.01), :);
a_Naphthene = a_Naphthene(a_Naphthene(:,2) >=
min(FinalRange_MON(:,3)), :);
a_Naphthene = a_Naphthene(a_Naphthene(:,2) <=
max(FinalRange_MON(:,3)+0.01), :);

a_Olefin = a_Olefin(a_Olefin(:,1) >=
min(FinalRange_RON(:,4)), :);
a_Olefin = a_Olefin(a_Olefin(:,1) <=
max(FinalRange_RON(:,4)+0.01), :);
a_Olefin = a_Olefin(a_Olefin(:,2) >=
min(FinalRange_MON(:,4)), :);
a_Olefin = a_Olefin(a_Olefin(:,2) <=
max(FinalRange_MON(:,4)+0.01), :);

a_Oxygenate = a_Oxygenate(a_Oxygenate(:,1) >=
min(FinalRange_RON(:,5)), :);
a_Oxygenate = a_Oxygenate(a_Oxygenate(:,1) <=
max(FinalRange_RON(:,5)+0.01), :);
a_Oxygenate = a_Oxygenate(a_Oxygenate(:,2) >=
min(FinalRange_MON(:,5)), :);
a_Oxygenate = a_Oxygenate(a_Oxygenate(:,2) <=
max(FinalRange_MON(:,5)+0.01), :);
toc
if isempty(a_Paraffin) == 1 || isempty(a_Aromatic) ==
1 || isempty(a_Naphthene) == 1 || isempty(a_Olefin)
== 1 || isempty(a_Oxygenate) == 1
    error = errordlg('Current fuel combinations
cannot obtain a legal fuel at this RON');
    uiwait (error);
    return
else
end

A=size(Paraffin_Combinations,1);

```

```

B=size(Paraffin_Combinations,2)+1;
b_Paraffin = zeros(A,1);
b_Paraffin(1:size(a_Paraffin,1)) = a_Paraffin(:,3);
Paraffin_Combinations(:,B) = (1:A);
Paraffin_Combinations(~ismember(Paraffin_Combinations
(:,B),b_Paraffin), :) = [];

A=size(Aromatic_Combinations,1);
B=size(Aromatic_Combinations,2)+1;
b_Aromatic = zeros(size(Aromatic_Combinations,1),1);
b_Aromatic(1:size(a_Aromatic,1)) = a_Aromatic(:,3);
Aromatic_Combinations(:,B) = (1:A);
Aromatic_Combinations(~ismember(Aromatic_Combinations
(:,B),b_Aromatic), :) = [];

A=size(Naphthene_Combinations,1);
B=size(Naphthene_Combinations,2)+1;
b_Naphthene =
zeros(size(Naphthene_Combinations,1),1);
b_Naphthene(1:size(a_Naphthene,1)) =
a_Naphthene(:,3);
Naphthene_Combinations(:,B) = (1:A);
Naphthene_Combinations(~ismember(Naphthene_Combinatio
ns(:,B),b_Naphthene), :) = [];

A=size(Olefin_Combinations,1);
B=size(Olefin_Combinations,2)+1;
b_Olefin = zeros(size(Olefin_Combinations,1),1);
b_Olefin(1:size(a_Olefin,1)) = a_Olefin(:,3);
Olefin_Combinations(:,B) = (1:A);
Olefin_Combinations(~ismember(Olefin_Combinations(:,B
),b_Olefin), :) = [];

A=size(Oxygenate_Combinations,1);
B=size(Oxygenate_Combinations,2)+1;
b_Oxygenate =
zeros(size(Oxygenate_Combinations,1),1);
b_Oxygenate(1:size(a_Oxygenate,1)) =
a_Oxygenate(:,3);
Oxygenate_Combinations(:,B) = (1:A);
Oxygenate_Combinations(~ismember(Oxygenate_Combinatio
ns(:,B),b_Oxygenate), :) = [];

clear b_Paraffin

```

```

clear b_Aromatic
clear b_Naphthene
clear b_Olefin
clear b_Oxygenate

clear a_Paraffin
clear a_Aromatic
clear a_Naphthene
clear a_Olefin
clear a_Oxygenate

clear A
clear B
Paraffin_Combinations(:,size(Paraffin_Combinations,2))= [];
Aromatic_Combinations(:,size(Aromatic_Combinations,2))= [];
Naphthene_Combinations(:,size(Naphthene_Combinations,2))= [];
Olefin_Combinations(:,size(Olefin_Combinations,2))= [];
Oxygenate_Combinations(:,size(Oxygenate_Combinations,2))= [];

A_Paraffin_Mix =
zeros(size(Paraffin_Combinations,1),10);
A_Paraffin_Mix(:,1) =
Paraffin_Combinations*Paraffin_PROPERTIES(:,1);
%C and CO2
A_Paraffin_Mix(:,2) =
Paraffin_Combinations*Paraffin_PROPERTIES(:,2);
%H
A_Paraffin_Mix(:,3) =
Paraffin_Combinations*Paraffin_PROPERTIES(:,3);
%O
A_Paraffin_Mix(:,4) = A_Paraffin_Mix(:,2)/2;
%H2O
A_Paraffin_Mix(:,5) =
(A_Paraffin_Mix(:,1)*Carbon)+(A_Paraffin_Mix(:,2)*Hyd
rogen); %mFuel
A_Paraffin_Mix(:,6) =
Paraffin_Combinations*Paraffin_PROPERTIES(:,6);
%Enthalpy of Formation

```

```

A_Paraffin_Mix(:,7) =
(A_Paraffin_Mix(:,1)*hCO2)+(A_Paraffin_Mix(:,4)*hH2O_
g)-A_Paraffin_Mix(:,6);    % hfNet_g
A_Paraffin_Mix(:,8) =
Paraffin_Combinations*Paraffin_PROPERTIES(:,4);
%RON
A_Paraffin_Mix(:,9) =
Paraffin_Combinations*Paraffin_PROPERTIES(:,5);
%MON
A_Paraffin_Mix(:,10) =
A_Paraffin_Mix(:,7)./A_Paraffin_Mix(:,5);
%LHV
A_Paraffin_Mix = A_Paraffin_Mix*Mix(1);

A_Aromatic_Mix =
zeros(size(Aromatic_Combinations,1),10);
A_Aromatic_Mix(:,1) =
Aromatic_Combinations*Aromatic_PROPERTIES(:,1);
%C and CO2
A_Aromatic_Mix(:,2) =
Aromatic_Combinations*Aromatic_PROPERTIES(:,2);
%H
A_Aromatic_Mix(:,3) =
Aromatic_Combinations*Aromatic_PROPERTIES(:,3);
%O
A_Aromatic_Mix(:,4) = A_Aromatic_Mix(:,2)/2;
%H2O
A_Aromatic_Mix(:,5) =
(A_Aromatic_Mix(:,1)*Carbon)+(A_Aromatic_Mix(:,2)*Hyd
rogen);    %mFuel
A_Aromatic_Mix(:,6) =
Aromatic_Combinations*Aromatic_PROPERTIES(:,6);
%Enthalpy of Formation
A_Aromatic_Mix(:,7) =
(A_Aromatic_Mix(:,1)*hCO2)+(A_Aromatic_Mix(:,4)*hH2O_
g)-A_Aromatic_Mix(:,6);    % hfNet_g
A_Aromatic_Mix(:,8) =
Aromatic_Combinations*Aromatic_PROPERTIES(:,4);
%RON
A_Aromatic_Mix(:,9) =
Aromatic_Combinations*Aromatic_PROPERTIES(:,5);
%MON

```

```

A_Aromatic_Mix(:,10) =
A_Aromatic_Mix(:,7)./A_Aromatic_Mix(:,5);
%LHV
A_Aromatic_Mix = A_Aromatic_Mix*Mix(2);

A_Naphthene_Mix =
zeros(size(Naphthene_Combinations,1),10);
A_Naphthene_Mix(:,1) =
Naphthene_Combinations*Naphthene_PROPERTIES(:,1);
%C and CO2
A_Naphthene_Mix(:,2) =
Naphthene_Combinations*Naphthene_PROPERTIES(:,2);
%H
A_Naphthene_Mix(:,3) =
Naphthene_Combinations*Naphthene_PROPERTIES(:,3);
%O
A_Naphthene_Mix(:,4) = A_Naphthene_Mix(:,2)/2;
%H2O
A_Naphthene_Mix(:,5) =
(A_Naphthene_Mix(:,1)*Carbon)+(A_Naphthene_Mix(:,2)*Hydrogen); %mFuel
A_Naphthene_Mix(:,6) =
Naphthene_Combinations*Naphthene_PROPERTIES(:,6);
%Enthalpy of Formation
A_Naphthene_Mix(:,7) =
(A_Naphthene_Mix(:,1)*hCO2)+(A_Naphthene_Mix(:,4)*hH2O_g)-A_Naphthene_Mix(:,6); % hfNet_g
A_Naphthene_Mix(:,8) =
Naphthene_Combinations*Naphthene_PROPERTIES(:,4);
%RON
A_Naphthene_Mix(:,9) =
Naphthene_Combinations*Naphthene_PROPERTIES(:,5);
%MON
A_Naphthene_Mix(:,10) =
A_Naphthene_Mix(:,7)./A_Naphthene_Mix(:,5);
%LHV
A_Naphthene_Mix = A_Naphthene_Mix*Mix(3);

A_Olefin_Mix = zeros(size(Olefin_Combinations,1),10);
A_Olefin_Mix(:,1) =
Olefin_Combinations*Olefin_PROPERTIES(:,1); %C
and CO2
A_Olefin_Mix(:,2) =
Olefin_Combinations*Olefin_PROPERTIES(:,2); %H

```

```

A_Olefin_Mix(:,3) =
Olefin_Combinations*Olefin_PROPERTIES(:,3);           %O
A_Olefin_Mix(:,4) = A_Olefin_Mix(:,2)/2;
%H2O
A_Olefin_Mix(:,5) =
(A_Olefin_Mix(:,1)*Carbon)+(A_Olefin_Mix(:,2)*Hydroge
n); %mFuel
A_Olefin_Mix(:,6) =
Olefin_Combinations*Olefin_PROPERTIES(:,6);
%Enthalpy of Formation
A_Olefin_Mix(:,7) =
(A_Olefin_Mix(:,1)*hCO2)+(A_Olefin_Mix(:,4)*hH2O_g) -
A_Olefin_Mix(:,6); % hfNet_g
A_Olefin_Mix(:,8) =
Olefin_Combinations*Olefin_PROPERTIES(:,4);
%RON
A_Olefin_Mix(:,9) =
Olefin_Combinations*Olefin_PROPERTIES(:,5);
%MON
A_Olefin_Mix(:,10) =
A_Olefin_Mix(:,7)./A_Olefin_Mix(:,5);
%LHV
A_Olefin_Mix = A_Olefin_Mix*Mix(4);

A_Oxygenate_Mix =
zeros(size(Oxygenate_Combinations,1),10);
A_Oxygenate_Mix(:,1) =
Oxygenate_Combinations*Oxygenate_PROPERTIES(:,1);
%C and CO2
A_Oxygenate_Mix(:,2) =
Oxygenate_Combinations*Oxygenate_PROPERTIES(:,2);
%H
A_Oxygenate_Mix(:,3) =
Oxygenate_Combinations*Oxygenate_PROPERTIES(:,3);
%O
A_Oxygenate_Mix(:,4) = A_Oxygenate_Mix(:,2)/2;
%H2O
A_Oxygenate_Mix(:,5) =
(A_Oxygenate_Mix(:,1)*Carbon)+(A_Oxygenate_Mix(:,2)*H
ydrogen)+(A_Oxygenate_Mix(:,3)*Oxygen); %mFuel
A_Oxygenate_Mix(:,6) =
Oxygenate_Combinations*Oxygenate_PROPERTIES(:,6);
%Enthalpy of Formation

```

```

A_Oxygenate_Mix(:,7) =
(A_Oxygenate_Mix(:,1)*hCO2)+(A_Oxygenate_Mix(:,4)*hH2
O_g)-A_Oxygenate_Mix(:,6);    % hfNet_g
A_Oxygenate_Mix(:,8) =
Oxygenate_Combinations*Oxygenate_PROPERTIES(:,4);
%RON
A_Oxygenate_Mix(:,9) =
Oxygenate_Combinations*Oxygenate_PROPERTIES(:,5);
%MON
A_Oxygenate_Mix(:,10) =
A_Oxygenate_Mix(:,7)./A_Oxygenate_Mix(:,5);
%LHV
A_Oxygenate_Mix = A_Oxygenate_Mix*Mix(5);

p=1;
I=size(FinalRange_RON,1);
for i = 1:I
    A=size(Paraffin_Combinations,1);
    Usable_Mix_Par = A_Paraffin_Mix(:,8:10);
    Usable_Mix_Par(:,4) = (1:A);

    A=size(Aromatic_Combinations,1);
    Usable_Mix_Aro = A_Aromatic_Mix(:,8:10);
    Usable_Mix_Aro(:,4) = (1:A);

    A=size(Naphthene_Combinations,1);
    Usable_Mix_Nap = A_Naphthene_Mix(:,8:10);
    Usable_Mix_Nap(:,4) = (1:A);

    A=size(Olefin_Combinations,1);
    Usable_Mix_Ole = A_Olefin_Mix(:,8:10);
    Usable_Mix_Ole(:,4) = (1:A);

    A=size(Oxygenate_Combinations,1);
    Usable_Mix_Oxy = A_Oxygenate_Mix(:,8:10);
    Usable_Mix_Oxy(:,4) = (1:A);
    Usable_Mix_Par(Usable_Mix_Par(:, 1) <
FinalRange_RON(i,1)-0.05 | Usable_Mix_Par(:, 1) >
FinalRange_RON(i,1)+0.1, :) = [];
    Usable_Mix_Aro(Usable_Mix_Aro(:, 1) <
FinalRange_RON(i,2)-0.05 | Usable_Mix_Aro(:, 1) >
FinalRange_RON(i,2)+0.1, :) = [];

```

```

        Usable_Mix_Nap(Usable_Mix_Nap(:, 1) <
FinalRange_RON(i,3)-0.05 | Usable_Mix_Nap(:, 1) >
FinalRange_RON(i,3)+0.1, :) = [];
        Usable_Mix_Ole(Usable_Mix_Ole(:, 1) <
FinalRange_RON(i,4)-0.05 | Usable_Mix_Ole(:, 1) >
FinalRange_RON(i,4)+0.1, :) = [];
        Usable_Mix_Oxy(Usable_Mix_Oxy(:, 1) <
FinalRange_RON(i,5)-0.05 | Usable_Mix_Oxy(:, 1) >
FinalRange_RON(i,5)+0.1, :) = [];
        if isempty(Usable_Mix_Par)==1 ||
isempty(Usable_Mix_Aro)==1
||isempty(Usable_Mix_Nap)==1
||isempty(Usable_Mix_Ole)==1
||isempty(Usable_Mix_Oxy)==1
            else
                sortrows(Usable_Mix_Par,3);
                sortrows(Usable_Mix_Aro,3);
                sortrows(Usable_Mix_Nap,3);
                sortrows(Usable_Mix_Ole,3);
                sortrows(Usable_Mix_Oxy,3);
                Usable(p,1)=Usable_Mix_Par(1,4);
                Usable(p,2)=Usable_Mix_Aro(1,4);
                Usable(p,3)=Usable_Mix_Nap(1,4);
                Usable(p,4)=Usable_Mix_Ole(1,4);
                Usable(p,5)=Usable_Mix_Oxy(1,4);
                p=p+1;
            end
        end
        Usable_Mix =
A_Paraffin_Mix(Usable(:,1),:)+A_Aromatic_Mix(Usable(:,
2),:)+A_Naphthene_Mix(Usable(:,3),:)+A_Olefin_Mix(Us
able(:,4),:)+A_Oxygenate_Mix(Usable(:,5),:);
        for j = 1:size(Usable,1)
            for i = 1:size(Paraffin_PROPERTIES,1)
                UP(i,1:3)=
Paraffin_Combinations(Usable(j,1),i).*Paraffin_PROPER
TIES(i,1:3);
            end
            UsableVolumes =
((UP(:,1)*Carbon)+(UP(:,2)*Hydrogen)+(UP(:,3)*Oxygen)
)*Mix(1);
            Um{j,1} = UsableVolumes;
            for i = 1:size(Paraffin_PROPERTIES,1)

```

```

        UsableVolumes(i,1) =
UsableVolumes(i,1)/Paraffin_PROPERTIES(i,7);
    end
    UV{j,1} = UsableVolumes;
    clear UsableVolumes;
    clear UP;
    for i = 1:size(Aromatic_PROPERTIES,1)
        UP(i,1:3)=
Aromatic_Combinations(Usable(j,2),i).*Aromatic_PROPER
TIES(i,1:3);
    end
    UsableVolumes =
((UP(:,1)*Carbon)+(UP(:,2)*Hydrogen)+(UP(:,3)*Oxygen)
)*Mix(2);
    Um{j,2} = UsableVolumes;
    for i = 1:size(Aromatic_PROPERTIES,1)

        UsableVolumes(i,1) =
UsableVolumes(i,1)/Aromatic_PROPERTIES(i,7);
    end
    UV{j,2} = UsableVolumes;
    clear UsableVolumes;
    clear UP;
    for i = 1:size(Naphthene_PROPERTIES,1)
        UP(i,1:3)=
Naphthene_Combinations(Usable(j,3),i).*Naphthene_PROP
ERTIES(i,1:3);
    end
    UsableVolumes =
((UP(:,1)*Carbon)+(UP(:,2)*Hydrogen)+(UP(:,3)*Oxygen)
)*Mix(3);
    Um{j,3} = UsableVolumes;
    for i = 1:size(Naphthene_PROPERTIES,1)

        UsableVolumes(i,1) =
UsableVolumes(i,1)/Naphthene_PROPERTIES(i,7);
    end
    UV{j,3} = UsableVolumes;
    clear UsableVolumes;
    clear UP;
    for i = 1:size(Olefin_PROPERTIES,1)
        UP(i,1:3)=
Olefin_Combinations(Usable(j,4),i).*Olefin_PROPERTIES
(i,1:3);

```

```

        end
        UsableVolumes =
        ((UP(:,1)*Carbon)+(UP(:,2)*Hydrogen)+(UP(:,3)*Oxygen)
        ) *Mix(4);
        Um{j,4} = UsableVolumes;
        for i = 1:size(Olefin_PROPERTIES,1)

            UsableVolumes(i,1) =
            UsableVolumes(i,1)/Olefin_PROPERTIES(i,7);
        end
        UV{j,4} = UsableVolumes;
        clear UsableVolumes;
        clear UP;
        for i = 1:size(Oxygenate_PROPERTIES,1)
            UP(i,1:3)=
            Oxygenate_Combinations(Usable(j,5),i).*Oxygenate_PROP
            ERTIES(i,1:3);
        end
        UsableVolumes =
        ((UP(:,1)*Carbon)+(UP(:,2)*Hydrogen)+(UP(:,3)*Oxygen)
        ) *Mix(5);
        Um{j,5} = UsableVolumes;
        for i = 1:size(Oxygenate_PROPERTIES,1)

            UsableVolumes(i,1) =
            UsableVolumes(i,1)/Oxygenate_PROPERTIES(i,7);
        end
        UV{j,5} = UsableVolumes;
        clear UsableVolumes;
        clear UP;
    end
    for j = 1:5
        for i = 1:size(Usable_Mix,1)
            Usable_Mix(i,11+j) =
            sum(cell2mat(Um(i,j)))/sum(cell2mat(UV(i,j)));
        end
    end
    for i = 1:size(Usable_Mix,1)
        Usable_Mix(i,11) =
        Usable_Mix(i,5)/((sum(cell2mat(Um(i,1)))/Usable_Mix(i
        ,12)))+(sum(cell2mat(Um(i,2)))/Usable_Mix(i,13)))+(sum(
        cell2mat(Um(i,3)))/Usable_Mix(i,14)))+(sum(cell2mat(Um
        (i,4)))/Usable_Mix(i,15)))+(sum(cell2mat(Um(i,5)))/Usa
        ble_Mix(i,16));
    end

```

```

end          %Density of a mixture = m/(m/d+m/d)

Usable_Mix(:,17) = Usable_Mix(:,5)./Usable_Mix(:,11);
Usable_Mix(:,18) = (1:size(Usable_Mix,1));
Benzene = Usable_Mix(:,17)*0.01;
VolumeA = Usable_Mix(:,17)*0.35;
VolumeO = Usable_Mix(:,17)*0.18;
mOxygenfuel = Usable_Mix(:,5)*0.027;
j=1;
Del = [];
Oxygenates_mass =
(A_Oxygenate_Mix(Usable(:,5),1)*Carbon)+(A_Oxygenate_
Mix(Usable(:,5),1)*Hydrogen)+(A_Oxygenate_Mix(Usable(
(:,5),1)*Oxygen);
mLess = Usable_Mix(:,5)*0.05;
for i = 1:size(Usable_Mix,1)
Hydrocarbon_mass(i,1) =
sum(Um{i,1}(cell2mat(Um(i,1))>mLess(i,1),
:))+sum(Um{i,2}(cell2mat(Um(i,2))>mLess(i,1),
:))+sum(Um{i,3}(cell2mat(Um(i,3))>mLess(i,1),
:))+sum(Um{i,4}(cell2mat(Um(i,4))>mLess(i,1), :));
end
Hydrocarbon_mass(:,1)=(Hydrocarbon_mass(:,1)./Usable_
Mix(:,5))*100;
Oxygenates_mass(:,1)
=(Oxygenates_mass(:,1)./Usable_Mix(:,5))*100;
CarbonCon = [100 100 1.2 35 35 30,
             0 5 10 10 10 10,
             5 20 20 15 10 10]/100;
    for i = 1:5
        CAro(:,i) =
((Aromatic_PROPERTIES(:,1))== (i+3));
        CNap(:,i) =
((Naphthene_PROPERTIES(:,1))== (i+3));
        Cole(:,i) =
((Olefin_PROPERTIES(:,1))== (i+3));
    end
        CAro(:,6) =
((Aromatic_PROPERTIES(:,1))>=9)
        CNap(:,6) =
((Naphthene_PROPERTIES(:,1))>=9);
        Cole(:,6) = ((Olefin_PROPERTIES(:,1))>=9);
        [CA(:,1), CA(:,2)] = find(CAro == 1)
        [CN(:,1), CN(:,2)] = find(CNap == 1)

```

```

[CO(:,1), CO(:,2)] = find(COle == 1)
LocationC = {0 0 0 0 0 0,
              0 0 0 0 0 0,
              0 0 0 0 0 0};
for i = 1:6
    if ismember(i,CA(:,2)) == 1
        LocationC{1,i}= (CA(CA(:,2)==i, 1));
    end
end
for i = 1:6
    if ismember(i,CN(:,2)) == 1
        LocationC{2,i}= (CN(CN(:,2)==i, 1));
    end
end
for i = 1:6
    if ismember(i,CO(:,2)) == 1
        LocationC{3,i}= (CO(CO(:,2)==i, 1));
    end
end

CA(1:size(Usable_Mix,1),1:6) = 0;
for k = 1:6
    for i = 1:size(Usable_Mix,1)
        if cell2mat(LocationC(1,k)) == 0
            else
                clear A
                A(:,:)=cell2mat(LocationC(1,k));
                for l = 1:size(A,1)
                    CA(i,k) = CA(i,k)+Um{i,2}(A(l),1);
                end
            end
        end
    end
end
end
CN(1:size(Usable_Mix,1),1:6) = 0;
for k = 1:6
    for i = 1:size(Usable_Mix,1)
        if cell2mat(LocationC(2,k)) == 0
            else
                clear A
                A(:,:)=cell2mat(LocationC(2,k));
                for l = 1:size(A,1)
                    CN(i,k) = CN(i,k)+Um{i,3}(A(l),1);
                end
            end
        end
    end
end
end

```

```

        end
        end
        CO(1:size(Usable_Mix,1),1:6) =
0;
        for k = 1:6
        for i = 1:size(Usable_Mix,1)
            if cell2mat(LocationC(3,k)) == 0
            else
                clear A
                A(:, :)=cell2mat(LocationC(3,k));
                for l = 1:size(A,1)
                    CO(i,k) = CO(i,k)+Um{i,4}(A(l),1);
                end
            end
        end
        end
        end
for i = 1:size(Usable_Mix,1)
    if UV{i,2}(3,1) > Benzene(i,1)
        Del(j,1)=i;
        Del(j,2)=3;
        j=j+1;
    elseif Usable_Mix(i, 9) < MON_min
        Del(j,1)=i;
        Del(j,2)=1;
        j=j+1;
    elseif Usable_Mix(i, 9) > MON_max
        Del(j,1)=i;
        Del(j,2)=2;
        j=j+1;
    elseif Usable_Mix(i,3)*Oxygen > mOxygenfuel
        Del(j,1)=i;
        Del(j,2)=4;
        j=j+1;
    elseif sum(cell2mat(UV(i,2))) > VolumeA(i,1)
        Del(j,1)=i;
        Del(j,2)=5;
        j=j+1;
    elseif sum(cell2mat(UV(i,4))) > VolumeO(i,1)
        Del(j,1)=i;
        Del(j,2)=6;
        j=j+1;
    elseif Usable_Mix(i, 8) < RON_Rmin
        Del(j,1)=i;

```

```

        Del(j,2)=7;
    j=j+1;
    elseif Usable_Mix(i, 8) > RON_Rmax
        Del(j,1)=i;
        Del(j,2)=8;
    j=j+1;
    elseif 100-Oxygenates_mass(i,1)-
Hydrocarbon_mass(i,1) < 30
        Del(j,1)=i;
        Del(j,2)=9;
        j=j+1;
    elseif Usable_Mix(i, 5)*CarbonCon(1,1)<
CA(i,1) || Usable_Mix(i, 5)*CarbonCon(1,2)< CA(i,2)
|| Usable_Mix(i, 5)*CarbonCon(1,3)< CA(i,3) ||
Usable_Mix(i, 5)*CarbonCon(1,4)< CA(i,4) ||
Usable_Mix(i, 5)*CarbonCon(1,5)< CA(i,5) ||
Usable_Mix(i, 5)*CarbonCon(1,6)< CA(i,6)
        Del(j,1)=i;
        Del(j,2)=10;
        j=j+1;
    elseif Usable_Mix(i, 5)*CarbonCon(2,1)<
CN(i,1) || Usable_Mix(i, 5)*CarbonCon(2,2)< CN(i,2)
|| Usable_Mix(i, 5)*CarbonCon(2,3)< CN(i,3) ||
Usable_Mix(i, 5)*CarbonCon(2,4)< CN(i,4) ||
Usable_Mix(i, 5)*CarbonCon(2,5)< CN(i,5) ||
Usable_Mix(i, 5)*CarbonCon(2,6)< CN(i,6)
        Del(j,1)=i;
        Del(j,2)=11;
        j=j+1;
    elseif Usable_Mix(i, 5)*CarbonCon(3,1)<
CO(i,1) || Usable_Mix(i, 5)*CarbonCon(3,2)< CO(i,2)
|| Usable_Mix(i, 5)*CarbonCon(3,3)< CO(i,3) ||
Usable_Mix(i, 5)*CarbonCon(3,4)< CO(i,4) ||
Usable_Mix(i, 5)*CarbonCon(3,5)< CO(i,5) ||
Usable_Mix(i, 5)*CarbonCon(3,6)< CO(i,6)
        Del(j,1)=i;
        Del(j,2)=12;
        j=j+1;

    end
end
if isempty(Del) == 0
Usable_Mix(Del, :) = [];
UV(Del, :) = [];

```

```

Um(Del, :) = [];
else
end
if isempty(Usable_Mix) == 1
    tf(1) = any(Del(:,2)==1);
    tf(2) = any(Del(:,2)==2);
    tf(3) = any(Del(:,2)==3);
    tf(4) = any(Del(:,2)==4);
    tf(5) = any(Del(:,2)==5);
    tf(6) = any(Del(:,2)==6);
    tf(7) = any(Del(:,2)==7);
    tf(8) = any(Del(:,2)==8);
    tf(9) = any(Del(:,2)==9);
    tf(10) = any(Del(:,2)==10);
    tf(11) = any(Del(:,2)==11);
    tf(12) = any(Del(:,2)==12);

    error = errorlog(['No combinations. 1 indicates
issue.', sprintf('\n') ' | MON below reg min ',
num2str(tf(1)), sprintf('\n') ' | MON above reg max
', num2str(tf(2)), sprintf('\n') ' | Benzene volume
too high ', num2str(tf(3)), sprintf('\n') ' | Oxygen
content in Fuel too high ', num2str(tf(4)),
sprintf('\n') ' | Aromatic total volume is too high
', num2str(tf(5)), sprintf('\n') ' | Olefin total
volume is too high ', num2str(tf(6)), sprintf('\n') '
| RON is below reg min ', num2str(tf(7)),
sprintf('\n') ' | RON is above reg max ',
num2str(tf(8)), sprintf('\n') ' | Not enough
Hydrocarbons < 5%(m/m) ', num2str(tf(9)),
sprintf('\n') ' | Aromatic carbon group concentration
too high ', num2str(tf(10)), sprintf('\n') ' |
Naphthene carbon group concentration too high ',
num2str(tf(11)), sprintf('\n') ' | Olefin carbon
group concentration too high ', num2str(tf(12))] );
    uiwait (error);

    return
end
Usable_Mix(:,18) = [];
T = array2table(Usable_Mix);
T.Properties.VariableNames = {'C' 'H' 'O' 'H2O'
'mFuel' 'Enthalpy of Formation' 'hfNet_g' 'RON' 'MON'

```

```

'LHV' 'p' 'p Par' 'p Aro' 'p Nap' 'p Ole' 'p Oxy'
'Total Volume'} ;
T.Properties.RowNames =
cellstr(string(1:size(Usable_Mix,1)));
figure('Position', [500 600 1385 500]);
X =
uitable('Data',T{:,:},'ColumnName',T.Properties.VariableNames,...
    'RowName',T.Properties.RowNames,'Units',
    'Normalized','Position',[0, 0, 1, 1]);
L=0;
while L == 0
    pause(10)
    B=0;
    Check = questdlg(['Continue?'], 'Parameter
    Check','Yes','No','Yes');
        switch Check
            case 'Yes'

                L = 1;

            case 'No'

                L = 0;

        end
    end
while U == 0
    select = []
while B == 0
    Select = str2double(inputdlg(['Select fuel ID
    from 1 - ', num2str(size(Usable_Mix,1))])));
    if Select > size(Usable_Mix,1)
        B=0
    else
        B=1
    end
end
end
Par_Amount(1,:) =
cell2mat(UV(Select,1))*(22/Usable_Mix(Select,17));
Aro_Amount(1,:) =
cell2mat(UV(Select,2))*(22/Usable_Mix(Select,17));
Nap_Amount(1,:) =
cell2mat(UV(Select,3))*(22/Usable_Mix(Select,17));

```

```

Ole_Amount(1,:) =
cell2mat(UV(Select,4))*(22/Usable_Mix(Select,17));
Oxy_Amount(1,:) =
cell2mat(UV(Select,5))*(22/Usable_Mix(Select,17));
Mix_Amount = [Par_Amount, Aro_Amount, Nap_Amount,
Ole_Amount, Oxy_Amount];
A = array2table(Mix_Amount, 'VariableNames', [PARAFFIN,
AROMATIC, NAPHTHENE, OLEFIN, OXYGENATE]);
figure('Position', [500 1100 1385 80]);
X =
uitable('Data',A{:,:}, 'ColumnName',A.Properties.Varia
bleNames,...
        'RowName',A.Properties.RowNames, 'Units',
'Normalized', 'Position', [0, 0, 1, 1]);
Check = questdlg(['Change Mix?'], 'Parameter
Check', 'Yes', 'No', 'Yes');
        switch Check
            case 'Yes'

                U = 0;
                B = 0;

            case 'No'

                U = 1;

        end
end

```