Spark Ignition Internal Combustion Engine Modelling using Matlab

David R Buttsworth

buttswod@usq.edu.au

http://www.usq.edu.au/users/buttswod/

October 16, 2002

Faculty of Engineering & Surveying Technical Reports
ISSN 1446-1846

Report TR-2002-02
ISBN 1 877078 02 6

Faculty of Engineering & Surveying
University of Southern Queensland
Toowoomba Qld 4350 Australia
http://www.usq.edu.au/

Abstract

A number of Matlab routines for combustion calculations and thermodynamic simulation of spark ignition internal combustion engine operation are described. Functions that return the thermodynamic curve coefficients for a variety of fuel, air, and combustion product species are described. A Matlab version of the Olikara and Borman method for determining the equilibrium state of combustion products is also presented. Additional routines specifically designed for spark ignition engine modelling are also described. Most of the routines included in this report are essentially Matlab versions of the FORTRAN programs presented by Ferguson for spark ignition engine calculations. Comparisons of results from the new Matlab routines and previous routines and data indicate that the new Matlab routines are reliable – typical deviations from previous results are less than 1 %.

Contents

1 Introduction

Matlab is popular for theoretical calculations and the analysis of experimental data. Although many thermodynamic and combustion routines are readily available in FORTRAN, the Matlab environment offers many advantages. Given the relative simplicity of internal combustion (IC) engine routines presented by Ferguson [1] it appeared reasonable to develop equivalent routines using Matlab. Having the fundamental routines available in Matlab affords workers that are familiar with this environment the capacity to rapidly adapt and extend the routines as required.

The Matlab routines described herein are based largely on the FORTRAN programs for IC engine calculations presented by Ferguson [1]. However, extensive use is made of the matrix and array data structures available in Matlab. In the subsequent sections of this report, the Matlab routines are described. Where appropriate, results obtained using the Matlab routines are compared to results obtained using established methods.

2 Basic Thermodynamic Data

2.1 Air and Combustion Products Data

Gordon and McBride [2] fitted curves to the tabulated JANAF data [3] (and a similar approach has been adopted for the Chemkin curves [4]) to give expressions for the thermodynamic properties of the form,

$$\frac{c_p}{R} = a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 T^4 \tag{1}$$

$$\frac{h}{RT} = a_1 + \frac{a_2}{2} T + \frac{a_3}{3} T^2 + \frac{a_4}{4} T^3 + \frac{a_5}{5} T^4 + a_6 \frac{1}{T} \tag{2}$$

$$\frac{s}{R} = a_1 \ln T + a_2 T + \frac{a_3}{2} T^2 + \frac{a_4}{3} T^3 + \frac{a_5}{4} T^4 + a_7 \tag{3}$$

The Matlab function airdata.m (Appendix A) returns the curve coefficients a_1 to a_7 for either the Gordon and McBride [2] curves or the Chemkin [4] curves over two different temperature ranges: 1) $300 < T < 1000\,\mathrm{K}$; and 2) $1000 < T < 5000\,\mathrm{K}$. There are 10 rows in the returned matrix and each row provides the seven coefficients (in ascending order) for a different species. The 10 species for which data is available are (in the following order): CO_2, H_2O, N_2, O_2, CO, H_2, H, O, OH, and NO. The options for airdata.m are listed in Table 1, and the numerical values for the curve coefficient can be found in the listing of airdata.m in Appendix A.

The specific heat curves for the 10 species using the Gordon and McBride [2] and the Chemkin [4] coefficients are presented in Fig. 1 and Fig. 2. These figures indicate that differences between the two sets of coefficients are relatively insignificant. For example, with the O_2 curve, the maximum difference between the values of c_p for the two different schemes is only 0.3 % at around 3500 K. By itself, this is a relatively small difference in engineering terms. When it is recognised that

scheme switch	meaning
`'GMcB_low'`	Gordon and McBride [2], $300 < T < 1000\,\mathrm{K}$
`'GMcB_hi'`	Gordon and McBride [2], $1000 < T < 5000\,\mathrm{K}$
`'Chemkin_low'`	Chemkin [4], $300 < T < 1000\,\mathrm{K}$
`'Chemkin_hi'`	Chemkin [4], $1000 < T < 5000\,\mathrm{K}$

Table 1: Options available in airdata.m

species	Gordon and McBride [2]		Chemkin [4]		JANAF [3]	
	$\overline{h}_f^0$ (kJ/kmol)	$\overline{s}^0$ (kJ/kmolK)	$\overline{h}_f^0$ (kJ/kmol)	$\overline{s}^0$ (kJ/kmolK)	$\overline{h}_f^0$ (kJ/kmol)	$\overline{s}^0$ (kJ/kmolK)
CO_2	-393 500.	213.697	-393 543.	213.735	-393 520.	213.69
H_2O	-241 817.	188.708	-241 843.	188.713	-241 810.	188.72
N_2	-0.3	191.502	1.4	191.509	0.	191.50
O_2	-0.4	205.037	-0.8	205.042	0.	205.04
CO	-110 526.	197.533	-110 540.	197.546	-110 530.	197.54
H_2	3.0	130.580	2.4	130.594	0.	130.57
H	217 977.	114.604	217 977.	114.604	218 000.	114.61
O	249 195.	160.944	249 195.	160.944	249 170.	160.95
OH	39 463.	183.594	38 986.	183.603	38 987.	183.60
NO	90 285.	210.639	90 297.	210.651	90 291.	210.65

Table 2: Values of $\overline{h}_f^0$ and $\overline{s}^0$ using the Gordon and McBride [2] and the Chemkin [4] coefficients compared with JANAF [3] values.

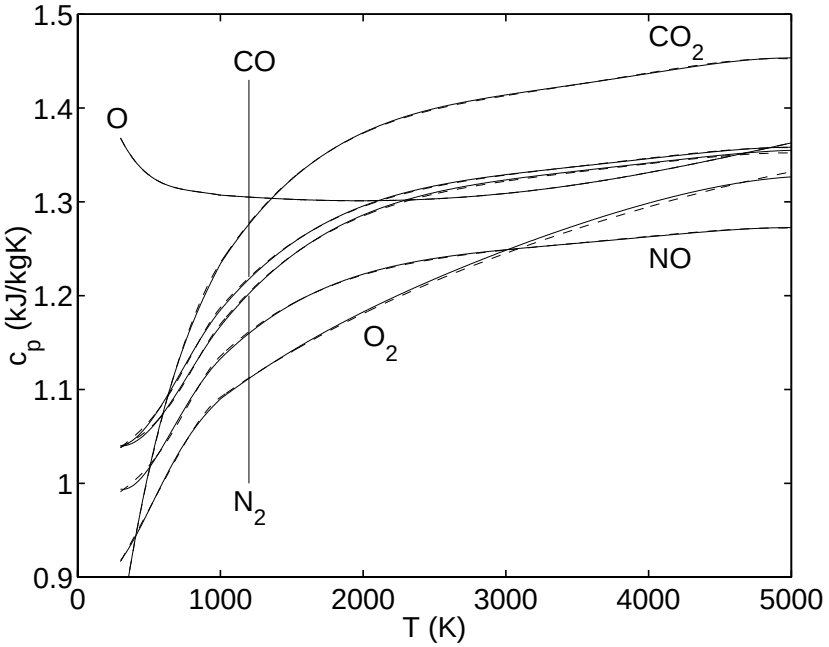

Figure 1: Comparison of specific heat curves for CO_2, N_2, O_2, CO, O, and NO using coefficients from Gordon and McBride [2] (solid lines) and Chemkin [4] (broken lines).

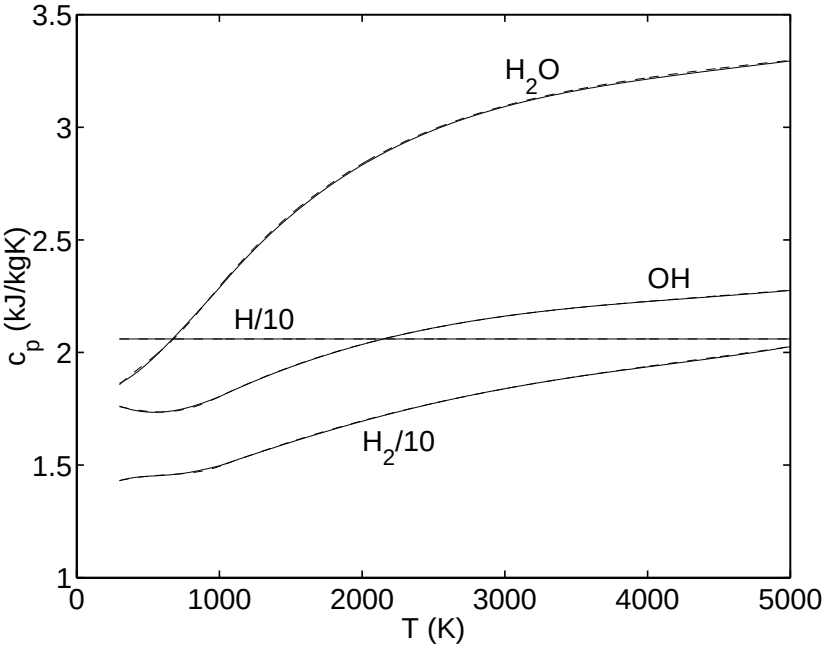

Figure 2: Comparison of specific heat curves for H_2O, H_2, H, and OH using coefficients from Gordon and McBride [2] (solid lines) and Chemkin [4] (broken lines). (The specific heat values for H_2 and H have been reduced by a factor of 10 for this figure.)

most of the O_2 will have dissociated by this temperature anyway, the equilibrium thermodynamic properties calculated by the two different schemes will be practically the same.

Reference values of enthalpy and entropy at $T = 298.15\,\mathrm{K}$ (and $p = 101.325\,\mathrm{kPa}$) calculated using the Gordon and McBride [2] and Chemkin [4] coefficients are presented in Table 2 where comparisons can also be made with the tabulated JANAF [3] values. Apart from the reference enthalpies for N_2, O_2, and H_2 (which should be zero), the maximum relative difference between curve values and JANAF values is about 1 % in the case of the Gordon and McBride [2] coefficients for $\left(\overline{h}_f^0\right)_{\mathrm{OH}}$. All of the other reference values (apart from zero reference enthalpies) typically deviate from the JANAF values by around 0.01 % only.

Through comparison of the Gordon and McBride [2] and Chemkin [4] curves, it was found that the coefficient a_1 reported by Ferguson [1] p129 for O in the high temperature range is incorrect. The correct value is $a_1 = 2.5420596$, and this value is used in airdata.m (see Appendix A) whereas the value reported by Ferguson is $a_1 = 5.5420596$ (which is incorrect).

2.2 Fuel Data

Heywood [5] represents the thermodynamic properties of selected fuels using curves that differ slightly in form to those of the air and combustion products species (Section 2.1). The fuel property curves of Heywood are given by,

$$\frac{c_p}{R} = a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 \frac{1}{T^2} \tag{4}$$

$$\frac{h}{RT} = a_1 + \frac{a_2}{2} T + \frac{a_3}{3} T^2 + \frac{a_4}{4} T^3 - a_5 \frac{1}{T^2} + a_6 \frac{1}{T} \tag{5}$$

$$\frac{s}{R} = a_1 \ln T + a_2 T + \frac{a_3}{2} T^2 + \frac{a_4}{3} T^3 - \frac{a_5}{2} \frac{1}{T^2} + a_7 \tag{6}$$

whereas other workers such as Ferguson [1] have adopted simplified versions of Eq. (4) to Eq. (6) in which $a_4 = a_5 = 0$.

The Matlab function fueldata.m (listed in Appendix B) returns a vector of curve coefficients corresponding to a_1 to a_7 for a number of different fuels. Actual numerical values for the curve coefficients can be found in Appendix B.

The various different fuel options and the sources of the data used in fueldata.m are presented in Table 3. Values of reference enthalpy and entropy for the curves at $T = 298.15\,\mathrm{K}$ (and $p = 101.325\,\mathrm{kPa}$) are also listed in Table 3. Where different curve fit coefficients are available for nominally the same fuel, the coefficients of Heywood [5] have been denoted with the suffix `_h`. Values for a_7 were not presented by Heywood [5], so these values have been obtained from other sources ([6], [1], and [7]). There is some scatter in the values of $\overline{h}_f^0$ and $\overline{s}^0$ reported by various authors – the maximum differences are around 0.8 %. The values of $\overline{h}_f^0$ and $\overline{s}^0$ from the present curves (Table 3) typically fall within the previously reported range of values.

In the case of methane, methanol, and iso-octane, recommended curve fits from different sources are available. Figure 3 provides a comparison c_p over the temperature range $300 < T < 1000\,\mathrm{K}$. In the case of methane, the values of c_p from the two different sources differ by less than 2 % over the specified temperature range.

3 Fuel, Air, and Residual Gases

Due to the volumetric inefficiency of IC engines, combustion products remain within the cylinder when the exhaust valve closes. These combustion products (also known as residual gases) are mixed with the fresh air-fuel mixture that enters the engine while the inlet valve remains open. The thermodynamic properties of the mixture of fuel, air, and residual gases can be determined using the routine farg.m which is listed in Appendix C. For details of the inputs and outputs from this function, see Appendix C, or from the Matlab base workspace, type: help farg.

The fuel air residual gas routine, farg.m is essentially a Matlab version of the FORTRAN subroutine of the same name that is presented by Ferguson [1] p111. For this subroutine, Ferguson [1] uses the results of Hires et al. [9] to determine the low temperature combustion products. farg.m utilizes the lower temperature range curve coefficients ($300 < T < 1000\,\mathrm{K}$) from airdata.m and the coefficients from fueldata.m. Hence, results from farg.m may be in error for $T > 1000\,\mathrm{K}$.

fuel switch	composition	source	$\overline{h}_f^0$ (kJ/kmol)	$\overline{s}^0$ (kJ/kmolK)
'methane'	CH_4	Ferguson [1]	-74 846	186.290
'methane_h'	CH_4	Heywood [5]	-74 870	186.271
'propane'	C_3H_8	Heywood [5]	-103 856	270.200
'benzene'	C_6H_6	Ferguson [1]	82 939	269.241
'hexane'	C_6H_{14}	Heywood [5]	-167 028	386.811
'toluene'	C_7H_8	Raine [8]	49 999	319.742
'isooctane'	C_8H_{18}	Raine [8]	-224 012	422.964
'isooctane_h'	C_8H_{18}	Heywood [5]	-224 109	422.960
'methanol'	CH_3OH	Ferguson [1]	-201 161	239.720
'methanol_h'	CH_3OH	Heywood [5]	-201 004	239.882
'ethanol'	C_2H_5OH	Heywood [5]	-236 266	280.640
'nitromethane'	CH_3NO_2	Ferguson [1]	-74 718	275.044
'gasoline'	C_7H_{17}	Ferguson [1]	-267 089	465.242
'gasoline_h1'	$C_{8.26}H_{15.5}$	Heywood [5]	-112 702	—
'gasoline_h2'	$C_{7.76}H_{13.1}$	Heywood [5]	-72 105	—
'diesel'	$C_{14.4}H_{24.9}$	Ferguson [1]	-99 946	645.445
'diesel_h'	$C_{10.8}H_{18.7}$	Heywood [5]	-180 940	—

Table 3: Fuel choices available in fueldata.m

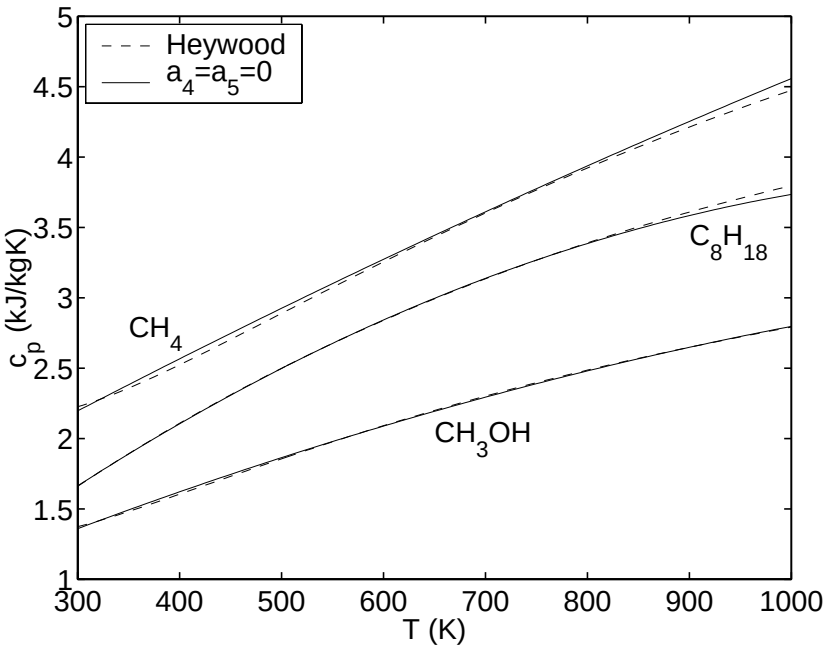

Figure 3: Comparison of specific heat curves for methane, methanol, and iso-octane from different sources.

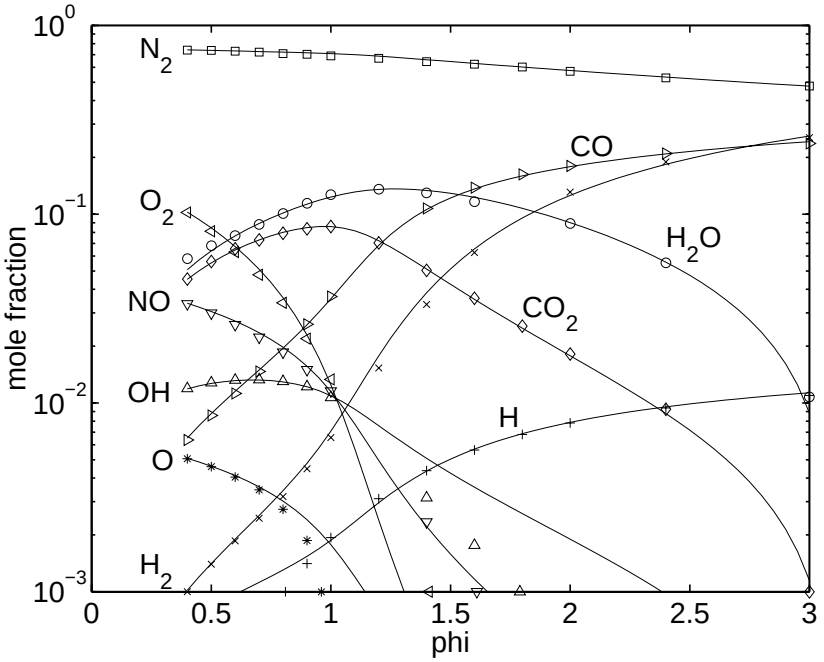

Figure 4: Comparison of equilibrium combustion products for $C_8H_{18} + 0.21\,O_2 + 0.79\,N_2$ at $T = 3000\,\mathrm{K}$ and $p = 50\,\mathrm{atm}$ from ecp.m (lines) and Figure 3.2 in Ferguson [1] (symbols).

4 Equilibrium Combustion Products

At elevated temperatures typical of combustion processes, the equilibrium state of the combustion products can be determined using the method of Olikara and Borman [10] provided the mixture is not too rich. The function ecp.m (listed in Appendix D) is a Matlab version of the FORTRAN subroutine presented by Ferguson [1] p128. For details of the inputs and outputs from this function, see Appendix D, or from the Matlab base workspace, type: help ecp.

To confirm the present implementation of the equilibrium combustion products routine, calculations were performed using ecp.m for isooctane-air mixtures at $T = 3000\,\mathrm{K}$ and $p = 50\,\mathrm{atm}$. In Fig. 4, results from ecp.m for equivalence ratios $0.4 \leq \phi \leq 3$ are compared with results from a complete equilibrium calculation (presented in Figure 3.2 of Ferguson [1] p120) using a NASA equilibrium program. It can be seen that the two results compare favourably for all 10 species except O_2, H, O, and OH when predicted mole fractions are less than about 0.3 %.

The performance of the Matlab version of ecp (Appendix D) has been further investigated by comparing the results from ecp.m with results from the FORTRAN code presented by Turns [7], TPEQUIL. This FORTRAN code is an implementation of the original Olikara and Borman [10] method that has been reworked and debugged over a number of years. The comparison of results is presented in Fig. 5. At each equivalence ratio, the two routines predict essentially the same concentrations of each species – actual molar differences are typically less than 0.2 %.

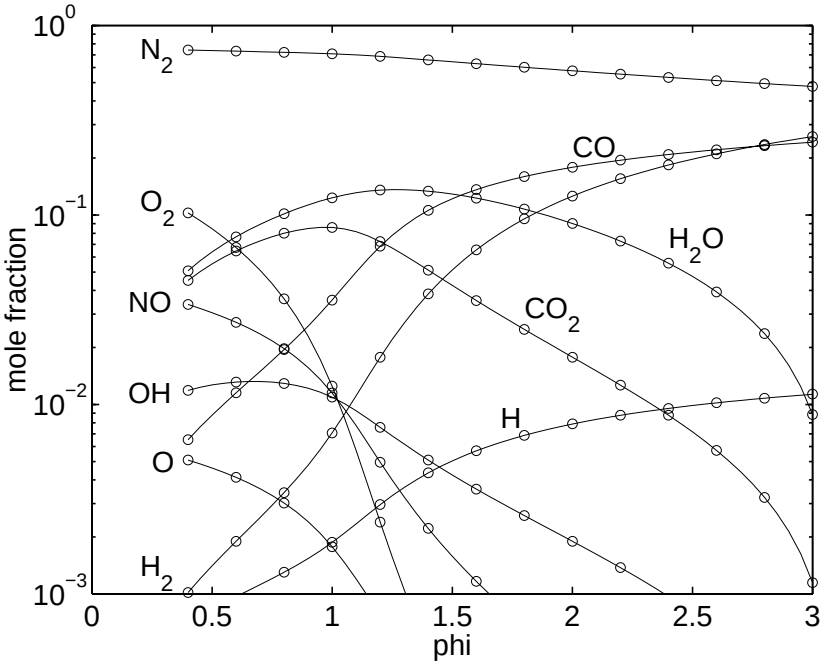

Figure 5: Comparison of equilibrium combustion products for $C_8H_{18} + 0.21\,O_2 + 0.79\,N_2$ at $T = 3000\,\mathrm{K}$ and $p = 50\,\mathrm{atm}$ from ecp.m (lines) and the Olikara and Borman routine by Turns [7] (symbols).

fuel	Tadiabatic.m (K)	T_{ad} (K) Turns [7]
`'methane'`	2225.7	2226
`'propane'`	2266.9	2267
`'benzene'`	2342.4	2342
`'hexane'`	2273.6	2273

Table 4: Comparison of selected adiabatic flame temperature results from Tadiabatic.m with results from Turns [7].

5 Adiabatic Flame Temperature

The adiabatic flame temperature is of interest in many combustion systems. In the context of the present spark ignition IC engine modelling, the adiabatic flame temperature is used as the initial temperature of the burned gas at the start of heat release. Tadiabatic.m (listed in Appendix E) first estimates the adiabatic flame temperature as 2000 K and compares the enthalpy of the equilibrium combustion products (from ecp.m) with the enthalpy of the unburned mixture (from farg.m). The temperature is then iteratively adjusted until the burned and unburned enthalpies are equal. For more detail, see Appendix E or from the Matlab base workspace, type: help Tadiabatic.

Table 4 compares results from Tadiabatic.m with results presented by Turns [7] for selected stoichiometric fuel-air mixtures at an initial temperature of 298.15 K. For the four cases tested, the adiabatic flame temperature results are generally within the round-off error which is about 0.05 %.

6 Arbitrary Heat Release

Ferguson [1] p175 presents a FORTRAN program for calculating the performance of a spark ignition IC engine based on a user-specified heat release as a function of crank angle. In this section of the report, a number of Matlab routines based on the work of Ferguson [1] are presented. The process of running a simulation using these Matlab routines is also outlined in this section.

6.1 Engine Specification

The engine geometry and operating parameters are specified in the file: enginedata.m. Appendix F lists the file enginedata.m. The current parameters specified in enginedata.m correspond to those used in Example 4.4 of Ferguson [1] p175. To model a different engine or operating condition, simply edit this file.

6.2 Functions for Differential Equations

The governing differential equations are integrated using the in-built Matlab function ode45.m. A slightly different set of equations is integrated depending on the process within the engine. Prior to any combustion, the differential equations to be integrated are specified in RatesComp.m; during combustion, RatesComb.m is used; and following combustion, RatesExp.m is used. These three functions are listed in Appendix G. RatesComp.m and RatesExp.m are essentially simplifications of RatesComb.m since there is no burned gas present prior to the start of combustion (RatesComp.m) and there is no unburned gas present following combustion (RatesExp.m).

6.3 Heat Transfer Modelling

Heat transfer from the working gas to the cylinder walls and piston can be modelled using either: 1) a constant heat transfer coefficient; or 2) the model by Woschni [11]. A switch is available in the enginedata.m file to set the model. When a constant heat transfer coefficient is used, two different heat transfer values can be set: one for the unburned zone (`hcu`), and the other for the burned zone (`hcb`) – see Appendix F. If the Woschni model is used, `hcu` and `hcb` should be set to values close to unity because they are treated as weighting factors for the model – these values can be tuned as necessary to improve agreement with experimental measurements.

6.4 Arbitrary Heat Release Routine

The main Matlab routine for the modelling of arbitrary heat release with a fuel inducted engine is ahrind.m (listed in Appendix H). The treatment of the burned fraction limits $x \to 0$ and $x \to 1$ differs slightly from that of Ferguson [1]. To run this script from the Matlab base workspace, simply type: ahrind. The parameters specified in enginedata.m will then be loaded, and the equations will be integrated for crank angles from $-\pi$ to π. Results are then saved to the file ahrind.mat.

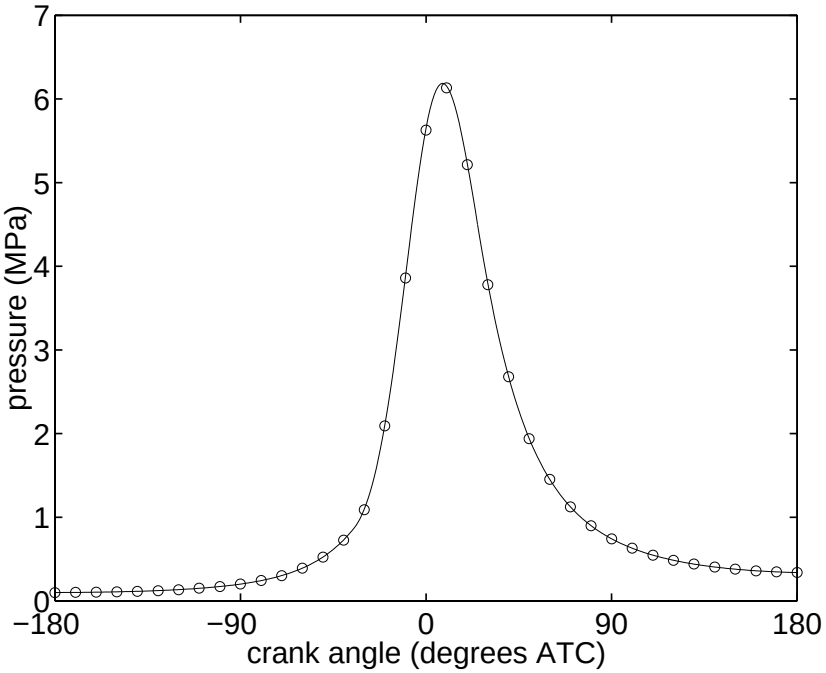

Figure 6: Pressure from ahrind.m (line) and Ferguson [1] (symbols).

6.5 Analysis of Results

To analyse and display the results, any number of Matlab scripts could be written. One such example is presented in Appendix I. To run this script from the Matlab base workspace, simply type: plotresults. This script file loads the results stored in ahrind.mat. It also loads results from the text file ferguson.txt which contains the tabulated output from the equivalent arbitrary heat release calculation of Ferguson [1] p178.

The Matlab script file plotresults.m prepares 6 figures for inspection. These figures are included in this report as Fig. 6 to Fig. 11. Comparisons between the present results and those of Ferguson [1] confirms that the present implementation produces the expected results. Figure 11 indicates the heat flux within the burned and unburned zones for the ferguson case. To produce these heat flux results, an additional function, calcq.m (Appendix J) is used to post-process the results.

The indicated mean effective pressure from the current calculations is 0.95278 MPa whereas Ferguson [1] obtains a value of 0.95102 MPa – a difference of around 0.2 %. Errors in the conservation of mass (`error1`) and energy (`error2`) are around 0.04 % for the current calculations.

7 Conclusions

A number of Matlab functions and script files have been written to simulate the thermodynamics of spark ignition internal combustion engines, based on the various FORTRAN codes described by Ferguson [1]. A variety of checks have been performed to confirm validity of the new Matlab routines.

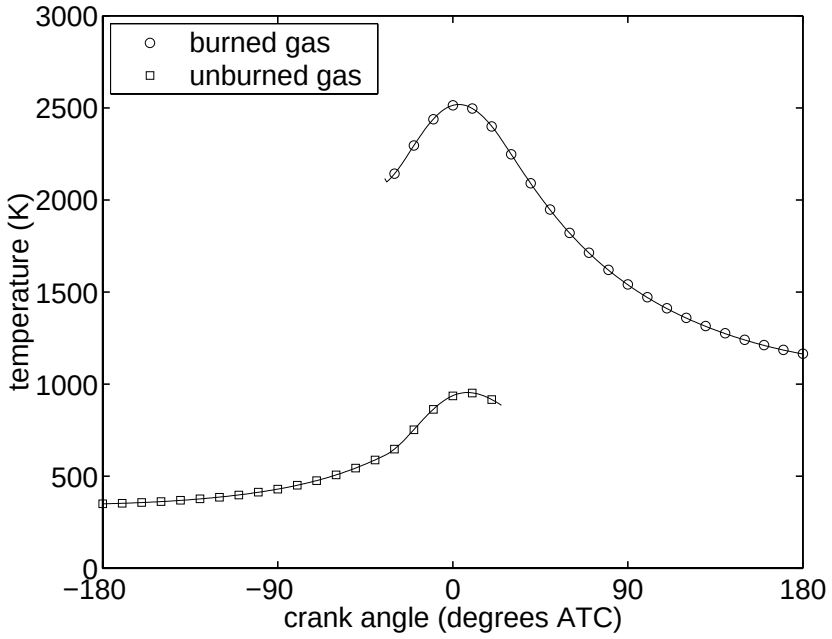

Figure 7: Temperature from ahrind.m (lines) and Ferguson [1] (symbols).

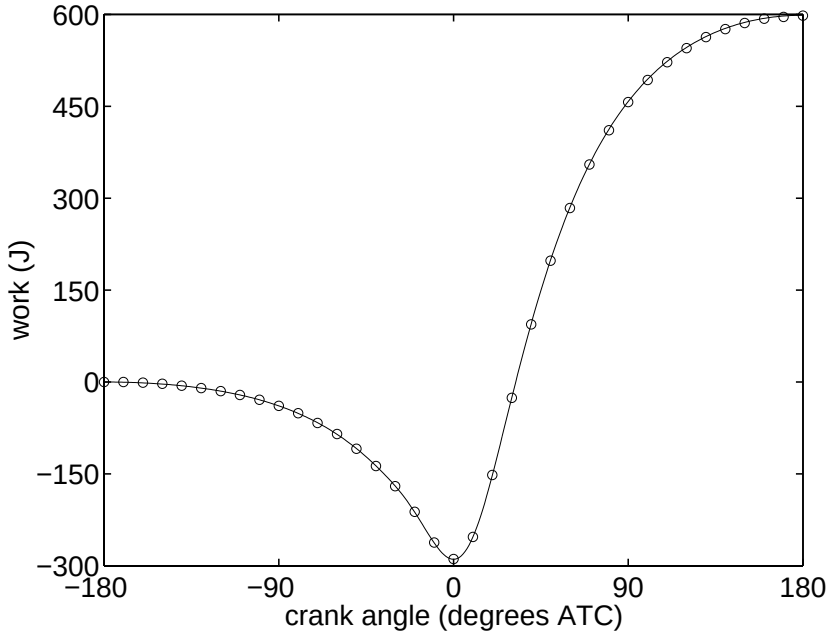

Figure 8: Work from ahrind.m (line) and Ferguson [1] (symbols).

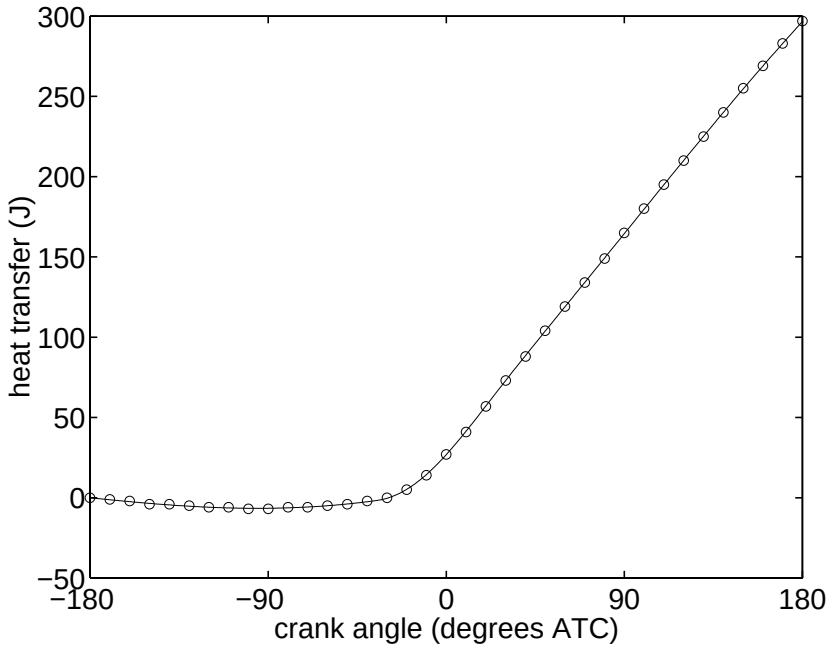

Figure 9: Heat transfer from ahrind.m (line) and Ferguson [1] (symbols).

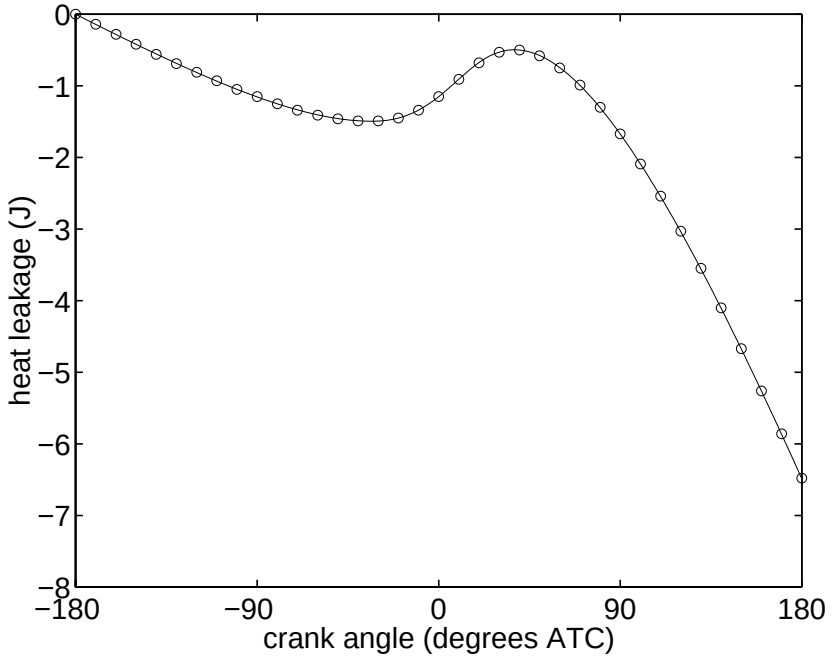

Figure 10: Heat leakage from ahrind.m (line) and Ferguson [1] (symbols).

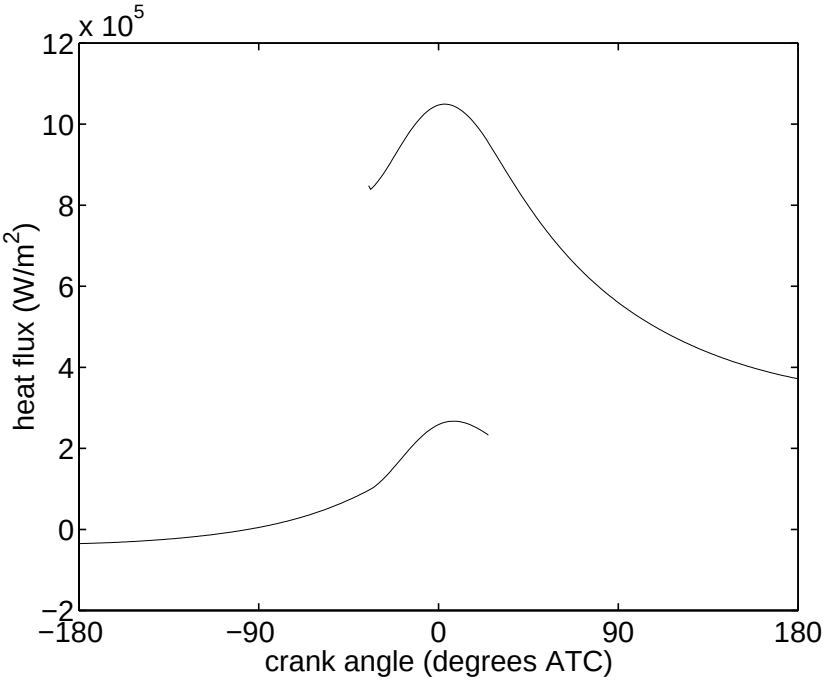

Figure 11: Heat flux results from ahrind.m.

The accuracy of the curve coefficients that are used to describe the thermodynamic properties of the fuel, air, and combustion products has been established through internal consistency checks and comparisons to well established properties at the reference conditions, $T = 298.15\,\mathrm{K}$ and $p = 101.325\,\mathrm{kPa}$. Based on these results, it is concluded that the fundamental thermodynamic properties of the fuel, air, and combustion product species are in agreement with previously published results to within the scatter of those results – typically less than 1 %.

The accuracy of the Matlab equilibrium combustion products routine was established through comparison with a comparable FORTRAN routine (Turns [7]) which has been revised over a number of years. Maximum differences in molar concentrations of less than 0.2 % were typically apparent over the range of conditions calculated by the two routines. Adiabatic flame temperature calculations also confirmed that differences between the new Matlab routines and established results are typically less than the round-off errors in the established results – around 0.05 %.

Comparison of an example engine calculation using the new Matlab routines and the original results obtained by Ferguson [1] demonstrates that routines produce accurate results. The indicated mean effective pressure calculated by the two methods differs by around 0.2 %.

References

[1] C. R. Ferguson, *Internal Combustion Engines, Applied Thermosciences*, John Wiley and Sons, New York, 1986.

[2] S. Gordon and B. J. McBride, "Computer Program for Calculation of Complex Chemical Equilibrium Composition, Rocket Performance, Incident and Reflected Shocks, and Chapman-Jouguet Detonations", NASA publication SP-273, 1971.

[3] "JANAF Thermochemical Tables", U.S National Bureau of Standards Publications NSRDS-NBS 37, June 1971.

[4] R. J. Kee, F. M. Rupley, and J. A. Miller, "The Chemkin Thermodynamic Data Base", Sandia Report SAND87-8215B, March 1991.

[5] J. B. Heywood, *Internal Combustion Engine Fundamentals*, McGraw-Hill, New York, 1988.

[6] R. Stone, *Introduction to Internal Combustion Engines*, Macmillan Press, Basingstoke, third edition, 1999.

[7] S. R. Turns, *An Introduction to Combustion, Concepts and Applications*, McGraw-Hill, New York, 1996.

[8] R. R. Raine, "ISIS_319 User Manual, computer modelling of nitric oxide formation in a spark ignition engine", Report, Revision 3, Oxford Engine Group, October 2000.

[9] S. D. Hires, A. Ekchian, J. B. Heywood, R. J. Tabaczynski, and J. C. Wall, "Performance and NOx Emissions Modeling of a Jet Ignition Pre-Chamber Stratified Charge Engine", *SAE Transactions*, vol. 85, Paper 760161, 1976.

[10] C. Olikara and G. L. Borman, "Calculating Properties of Equilibrium Combustion Products with Some Applications to I.C. Engines", SAE Paper 750468, 1975.

[11] G. Woschni, "A Universally Applicable Equation for the Instantaneous Heat Transfer Coefficient in the Internal Combustion Engine", SAE Paper 670931, 1967.

A airdata.m

```
function A=airdata(scheme);
%
% A=airdata(scheme)
%
% Routine to specify the thermodynamic properties of air and
% combustion products.
% Data taken from:
% 1. Gordon, S., and McBride, B. J., 1971, "Computer Program for
% Calculation of Complex Chemical Equilibrium Composition, Rocket
% Performance, Incident and Reflected Shocks, and Chapman-Jouguet
% Detonations," NASA SP-273.  As reported in Ferguson, C. R., 1986,
% "Internal Combustion Engines", Wiley.
% 2. Kee, R. J., et al., 1991, "The Chemkin Thermodynamic Data Base",
% Sandia Report, SAND87-8215B. As reported in
% Turns, S. R., 1996, "An Introduction to Combustion:
% Concepts and Applications", McGraw-Hill.
% ********************************************************************
% input:
% scheme switch:
% 'GMcB_low' - Gordon and McBride 300 < T < 1000 K
% 'GMcB_hi' - Gordon and McBride 1000 < T < 5000 K
% 'Chemkin_low' - Chemkin 300 < T < 1000 K
% 'Chemkin_hi' - Chemkin 1000 < T < 5000 K
% output:
% A - matrix of polynomial coefficients for cp/R, h/RT, and s/R
% of the form h/RT=a1+a2*T/2+a3*T^2/3+a4*T^3/4+a5*T^4/5*a6/T (for
% example) where T is expressed in K
% columns 1 to 7 are coefficients a1 to a7, and
% rows 1 to 10 are species CO2 H2O N2 O2 CO H2 H O OH and NO
% ********************************************************************

switch scheme
case 'GMcB_low'
   A=[ 0.24007797E+01  0.87350957E-02 -0.66070878E-05  0.20021861E-08 ...
       0.63274039E-15 -0.48377527E+05  0.96951457E+01
       0.40701275E+01 -0.11084499E-02  0.41521180E-05 -0.29637404E-08 ...
       0.80702103E-12 -0.30279722E+05 -0.32270046E+00
       0.36748261E+01 -0.12081500E-02  0.23240102E-05 -0.63217559E-09 ...
      -0.22577253E-12 -0.10611588E+04  0.23580424E+01
       0.36255985E+01 -0.18782184E-02  0.70554544E-05 -0.67635137E-08 ...
       0.21555993E-11 -0.10475226E+04  0.43052778E+01
       0.37100928E+01 -0.16190964E-02  0.36923594E-05 -0.20319674E-08 ...
       0.23953344E-12 -0.14356310E+05  0.29555350E+01
       0.30574451E+01  0.26765200E-02 -0.58099162E-05  0.55210391E-08 ...
      -0.18122739E-11 -0.98890474E+03 -0.22997056E+01
       0.25000000E+01  0.00000000E+00  0.00000000E+00  0.00000000E+00 ...
       0.00000000E+00  0.25471627E+05 -0.46011762E+00
```

```
       0.29464287E+01 -0.16381665E-02  0.24210316E-05 -0.16028432E-08 ...
       0.38906964E-12  0.29147644E+05  0.29639949E+01
       0.38375943E+01 -0.10778858E-02  0.96830378E-06  0.18713972E-09 ...
      -0.22571094E-12  0.36412823E+04  0.49370009E+00
       0.40459521E+01 -0.34181783E-02  0.79819190E-05 -0.61139316E-08  ...
       0.15919076E-11  0.97453934E+04  0.29974988E+01];
case 'GMcB_hi'
   A=[ 0.44608041E+01  0.30981719E-02 -0.12392571E-05  0.22741325E-09 ...
      -0.15525954E-13 -0.48961442E+05 -0.98635982E+00
       0.27167633E+01  0.29451374E-02 -0.80224374E-06  0.10226682E-09 ...
      -0.48472145E-14 -0.29905826E+05  0.66305671E+01
       0.28963194E+01  0.15154866E-02 -0.57235277E-06  0.99807393E-10 ...
      -0.65223555E-14 -0.90586184E+03  0.61615148E+01
       0.36219535E+01  0.73618264E-03 -0.19652228E-06  0.36201558E-10 ...
      -0.28945627E-14 -0.12019825E+04  0.36150960E+01
       0.29840696E+01  0.14891390E-02 -0.57899684E-06  0.10364577E-09 ...
      -0.69353550E-14 -0.14245228E+05  0.63479156E+01
       0.31001901E+01  0.51119464E-03  0.52644210E-07 -0.34909973E-10 ...
       0.36945345E-14 -0.87738042E+03 -0.19629421E+01
       0.25000000E+01  0.00000000E+00  0.00000000E+00  0.00000000E+00 ...
       0.00000000E+00  0.25471627E+05 -0.46011763E+00
       0.25420596E+01 -0.27550619E-04 -0.31028033E-08  0.45510674E-11 ...
      -0.43680515E-15  0.29230803E+05  0.49203080E+01
       0.29106427E+01  0.95931650E-03 -0.19441702E-06  0.13756646E-10 ...
       0.14224542E-15  0.39353815E+04  0.54423445E+01
       0.31890000E+01  0.13382281E-02 -0.52899318E-06  0.95919332E-10 ...
      -0.64847932E-14  0.98283290E+04  0.67458126E+01];
case 'Chemkin_low'
   A=[ 0.02275724E+02  0.09922072E-01 -0.10409113E-04  0.06866686E-07 ...
      -0.02117280E-10 -0.04837314E+06  0.10188488E+02
       0.03386842E+02  0.03474982E-01 -0.06354696E-04  0.06968581E-07 ...
      -0.02506588E-10 -0.03020811E+06  0.02590232E+02
       0.03298677E+02  0.14082404E-02 -0.03963222E-04  0.05641515E-07 ...
      -0.02444854E-10 -0.10208999E+04  0.03950372E+02
       0.03212936E+02  0.11274864E-02 -0.05756150E-05  0.13138773E-08 ...
      -0.08768554E-11 -0.10052490E+04  0.06034737E+02
       0.03262451E+02  0.15119409E-02 -0.03881755E-04  0.05581944E-07 ...
      -0.02474951E-10 -0.14310539E+05  0.04848897E+02
       0.03298124E+02  0.08249441E-02 -0.08143015E-05 -0.09475434E-09 ...
       0.04134872E-11 -0.10125209E+04 -0.03294094E+02
       0.02500000E+02  0.00000000E+00  0.00000000E+00  0.00000000E+00 ...
       0.00000000E+00  0.02547162E+06 -0.04601176E+01
       0.02946428E+02 -0.16381665E-02  0.02421031E-04 -0.16028431E-08 ...
       0.03890696E-11  0.02914764E+06  0.02963995E+02
       0.03637266E+02  0.01850910E-02 -0.16761646E-05  0.02387202E-07 ...
      -0.08431442E-11  0.03606781E+05  0.13588605E+01
       0.03376541E+02  0.12530634E-02 -0.03302750E-04  0.05217810E-07 ...
      -0.02446262E-10  0.09817961E+05  0.05829590E+02];
case 'Chemkin_hi'
   A=[ 0.04453623E+02  0.03140168E-01 -0.12784105E-05  0.02393996E-08 ...
```

```
      -0.16690333E-13 -0.04896696E+06 -0.09553959E+01
       0.02672145E+02  0.03056293E-01 -0.08730260E-05  0.12009964E-09 ...
      -0.06391618E-13 -0.02989921E+06  0.06862817E+02
       0.02926640E+02  0.14879768E-02 -0.05684760E-05  0.10097038E-09 ...
      -0.06753351E-13 -0.09227977E+04  0.05980528E+02
       0.03697578E+02  0.06135197E-02 -0.12588420E-06  0.01775281E-09 ...
      -0.11364354E-14 -0.12339301E+04  0.03189165E+02
       0.03025078E+02  0.14426885E-02 -0.05630827E-05  0.10185813E-09 ...
      -0.06910951E-13 -0.14268350E+05  0.06108217E+02
       0.02991423E+02  0.07000644E-02 -0.05633828E-06 -0.09231578E-10 ...
       0.15827519E-14 -0.08350340E+04 -0.13551101E+01
       0.02500000E+02  0.00000000E+00  0.00000000E+00  0.00000000E+00 ...
       0.00000000E+00  0.02547162E+06 -0.04601176E+01
       0.02542059E+02 -0.02755061E-03 -0.03102803E-07  0.04551067E-10 ...
      -0.04368051E-14  0.02923080E+06  0.04920308E+02
       0.02882730E+02  0.10139743E-02 -0.02276877E-05  0.02174683E-09 ...
      -0.05126305E-14  0.03886888E+05  0.05595712E+02
       0.03245435E+02  0.12691383E-02 -0.05015890E-05  0.09169283E-09 ...
      -0.06275419E-13  0.09800840E+05  0.06417293E+02];
end
```

B fueldata.m

```
function [alpha,beta,gamma,delta,Afuel]=fueldata(fuel);
%
% [alpha,beta,gamma,delta,Afuel]=fueldata(fuel)
%
% Routine to specify the thermodynamic properties of a fuel.
% Data taken from:
% 1. Ferguson, C.R., 1986, "Internal Combustion Engines", Wiley;
% 2. Heywood, J.B., 1988, "Internal Combustion Engine Fundamentals",
%    McGraw-Hill; and
% 3. Raine, R. R., 2000, "ISIS_319 User Manual", Oxford Engine Group.
% ****************************************************************
% input:
% fuel switch
% from Ferguson: 'gasoline', 'diesel', 'methane', 'methanol',
% 'nitromethane', 'benzene';
% from Heywood: 'methane_h', 'propane', 'hexane', 'isooctane_h',
% 'methanol_h', 'ethanol', 'gasoline_h1', gasoline_h2', 'diesel_h';
% from Raine: 'toluene', 'isooctane'.
% output:
% alpha, beta, gamma, delta - number of C, H, O, and N atoms
% Afuel - vector of polynomial coefficients for cp/R, h/RT, and s/R
% of the form h/RT=a1+a2*T/2+a3*T^2/3+a4*T^3/4-a5/T^2+a6/T (for
% example) where T is expressed in K.
% ****************************************************************
```

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

switch fuel
case 'gasoline' % Ferguson
   alpha=7; beta=17; gamma=0; delta=0;
   Afuel=[4.0652 6.0977E-02 -1.8801E-05 0 0 -3.5880E+04 15.45];
case 'diesel' % Ferguson
   alpha=14.4; beta=24.9; gamma=0; delta=0;
   Afuel=[7.9710 1.1954E-01 -3.6858E-05 0 0 -1.9385E+04 -1.7879];
case 'methane' % Ferguson
   alpha=1; beta=4; gamma=0; delta=0;
   Afuel=[1.971324 7.871586E-03 -1.048592E-06 0 0 -9.930422E+03 8.873728];
case 'methanol' % Ferguson
   alpha=1; beta=4; gamma=1; delta=0;
   Afuel=[1.779819 1.262503E-02 -3.624890E-06 0 0 -2.525420E+04 1.50884E+01];
case 'nitromethane' % Ferguson
   alpha=1; beta=3; gamma=2; delta=1;
   Afuel=[1.412633 2.087101E-02 -8.142134E-06 0 0 -1.026351E+04 1.917126E+01];
case 'benzene' % Ferguson
   alpha=6; beta=6; gamma=0; delta=0;
   Afuel=[-2.545087 4.79554E-02 -2.030765E-05 0 0 8.782234E+03 3.348825E+01];
case 'toluene' % Raine
   alpha=7; beta=8; gamma=0; delta=0;
   Afuel=[-2.09053 5.654331e-2 -2.350992e-5 0 0 4331.441411 34.55418257];
case 'isooctane' % Raine
   alpha=8; beta=18; gamma=0; delta=0;
   Afuel=[6.678E-1 8.398E-2 -3.334E-5 0 0 -3.058E+4 2.351E+1];
case 'methane_h' % Heywood
   alpha=1; beta=4; gamma=0; delta=0;
   Afuel=[-0.29149 26.327 -10.610 1.5656 0.16573 -18.331 19.9887/SVal].*SVec;
case 'propane' % Heywood
   alpha=3; beta=8; gamma=0; delta=0;
   Afuel=[-1.4867 74.339 -39.065 8.0543 0.01219 -27.313 26.4796/SVal].*SVec;
case 'hexane' % Heywood
   alpha=6; beta=14; gamma=0; delta=0;
   Afuel=[-20.777 210.48 -164.125 52.832 0.56635 -39.836 79.5542/SVal].*SVec;
case 'isooctane_h' % Heywood
   alpha=8; beta=18; gamma=0; delta=0;
   Afuel=[-0.55313 181.62 -97.787 20.402 -0.03095 -60.751 27.2162/SVal].*SVec;
case 'methanol_h' % Heywood
   alpha=1; beta=4; gamma=1; delta=0;
   Afuel=[-2.7059 44.168 -27.501 7.2193 0.20299 -48.288 31.1406/SVal].*SVec;
case 'ethanol' % Heywood
   alpha=2; beta=6; gamma=1; delta=0;
   Afuel=[6.990 39.741 -11.926 0 0 -60.214 8.01623/SVal].*SVec;
case 'gasoline_h1' % Heywood
   alpha=8.26; beta=15.5; gamma=0; delta=0;
```

```
   Afuel=[-24.078 256.63 -201.68 64.750 0.5808 -27.562 NaN].*SVec;
case 'gasoline_h2' % Heywood
   alpha=7.76; beta=13.1; gamma=0; delta=0;
   Afuel=[-22.501 227.99 -177.26 56.048 0.4845 -17.578 NaN].*SVec;
case 'diesel_h' % Heywood
   alpha=10.8; beta=18.7; gamma=0; delta=0;
   Afuel=[-9.1063 246.97 -143.74 32.329 0.0518 -50.128 NaN].*SVec;
end
```

C farg.m

```
function [h,u,v,s,Y,cp,dlvlT,dlvlp]=farg(p,T,phi,f,fueltype,airscheme);
%
% [h,u,v,s,Y,cp,dlvlT,dlvlp]=farg(p,T,phi,f,fueltype,airscheme)
%
% Routine to determine the state of mixtures of fuel, air
% and residual combustion products at low temperatures.
% Method closely follows that of:
% 1. Ferguson, C.R., 1986, "Internal Combustion Engines", Wiley, p108;
% who uses the results of:
% 2. Hires, S.D.,  Ekchian, A., Heywood, J.B., Tabaczynski, R.J., and
% Wall, J.C., 1976, "Performance and NOx Emissions Modeling of a Jet
% Ignition Pre-Chamber Stratified Charge Engine", SAE Trans., Vol 85,
% Paper 760161.
% ********************************************************************
% input:
% p,T,phi - pressure (Pa), temperature (K), and equivalence ratio
% f - residual mass fraction; set f=0 if no combustion products
% are present and f=1 if only combustion products are present
% fueltype - 'gasoline', 'diesel', etc - see fueldata.m for full list
% airscheme - 'GMcB' (Gordon and McBride) or 'Chemkin'
% output:
% h - enthalpy (J/kg), u - internal energy (J/kg),
% v - specific volume (m^3/kg), s - entropy (J/kgK),
% Y - mole fractions of 6 species: CO2, H2O, N2, O2, CO, and H2,
% cp - specific heat (J/kgK),
% dlvlT - partial derivative of log(v) wrt log(T)
% dlvlp - partial derivative of log(v) wrt log(p)
% ********************************************************************

[alpha,beta,gamma,delta,Afuel]=fueldata(fueltype);
switch airscheme
case 'GMcB'
   A=airdata('GMcB_low');
case 'Chemkin'
   A=airdata('Chemkin_low');
end
```

```
Ru=8314.34; % J/kmolK
table=[-1 1 0 0 1 -1]';
M=[44.01 18.02 28.008 32.000 28.01 2.018]'; % kg/kmol

MinMol=1e-25;

dlvlT=1; dlvlp=-1;
eps=0.210/(alpha+0.25*beta-0.5*gamma);

if phi <= 1.0 % stoichiometric or lean
   nu=[alpha*phi*eps beta*phi*eps/2 0.79+delta*phi*eps/2 ...
         0.21*(1-phi) 0 0]';
   dcdT=0;
else % rich
   z=1000/T;
   K=exp(2.743+z*(-1.761+z*(-1.611+z*0.2803)));
   dKdT=-K*(-1.761+z*(-3.222+z*0.8409))/1000;
   a=1-K;
   b=0.42-phi*eps*(2*alpha-gamma)+K*(0.42*(phi-1)+alpha*phi*eps);
   c=-0.42*alpha*phi*eps*(phi-1)*K;
   nu5=(-b+sqrt(b^2-4*a*c))/2/a;
   dcdT=dKdT*(nu5^2-nu5*(0.42*(phi-1)+alpha*phi*eps)+ ...
      0.42*alpha*phi*eps*(phi-1))/(2*nu5*a+b);
   nu=[alpha*phi*eps-nu5 0.42-phi*eps*(2*alpha-gamma)+nu5 ...
         0.79+delta*phi*eps/2 0 nu5 0.42*(phi-1)-nu5]';
end

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

% mole fractions and molecular weight of fuel-air
fuel=eps*phi/(1+eps*phi);
o2=0.21/(1+eps*phi);
n2=0.79/(1+eps*phi);
Mfa=fuel*(12.01*alpha+1.008*beta+16*gamma+14.01*delta)+ ...
   32*o2+28.02*n2;

% mole fractions of fuel-air-residual gas
Yres=f/(f+Mres/Mfa*(1-f));
Y=Y*Yres;
Yfuel=fuel*(1-Yres);
Y(3)=Y(3)+n2*(1-Yres);
Y(4)=Y(4)+o2*(1-Yres);

% component properties
Tcp0=[1 T T^2 T^3 T^4]';
Th0=[1 T/2 T^2/3 T^3/4 T^4/5 1/T]';
Ts0=[log(T) T T^2/2 T^3/3 T^4/4 1]';
```

```
cp0=A(1:6,1:5)*Tcp0;
h0=A(1:6,1:6)*Th0;
s0=A(1:6,[1:5 7])*Ts0;
Mfuel=12.01*alpha+1.008*beta+16.000*gamma+14.01*delta;
a0=Afuel(1); b0=Afuel(2); c0=Afuel(3); d0=Afuel(6); e0=Afuel(7);
cpfuel=Afuel(1:5)*[1 T T^2 T^3 1/T^2]';
hfuel=Afuel(1:6)*[1 T/2 T^2/3 T^3/4 -1/T^2 1/T]';
s0fuel=Afuel([1:5 7])*[log(T) T T^2/2 T^3/3 -1/T^2/2 1]';

% set min value of composition so log calculations work
if Yfuel<MinMol
   Yfuel=MinMol;
end
i=find(Y<MinMol);
Y(i)=ones(length(i),1)*MinMol;

% properties of mixture
h=hfuel*Yfuel+sum(h0.*Y);
s=(s0fuel-log(Yfuel))*Yfuel+sum((s0-log(Y)).*Y);
cp=cpfuel*Yfuel+sum(cp0.*Y)+sum(h0.*table*T*dcdT*Yres/tmoles);
MW=Mfuel*Yfuel+sum(Y.*M);

R=Ru/MW;
h=R*T*h;
u=h-R*T;
v=R*T/p;
s=R*(-log(p/101.325e3)+s);
cp=R*cp;
```

D ecp.m

```
function [h,u,v,s,Y,cp,dlvlT,dlvlp]=ecp(p,T,phi,fueltype,airscheme,Yguess);
%
% [h,u,v,s,Y,cp,dlvlT,dlvlp]=ecp(p,T,phi,fueltype,airscheme,Yguess)
%
% Routine to determine the equilibrium state of combustion products.
% Method closely follows that of:
% 1. Ferguson, C.R., 1986, "Internal Combustion Engines", Wiley, p122;
% which uses the method described by:
% 2. Olikara, C., and Borman, G.L., 1975, "A Computer Program for
% Calculating Properties of Equilibrium Combustion Products with
% Some Applications to I.C. Engines", SAE Paper 750468.
% ********************************************************************
% input:
% p,T,phi - pressure (Pa), temperature (K), and equivalence ratio
% fueltype - 'gasoline', 'diesel', etc - see fueldata.m for full list
% airscheme - 'GMcB' (Gordon and McBride) or 'Chemkin'
% Yguess - (optional) initial estimate for mole fractions of the
```

```
% species CO2 H2O N2 O2 CO H2 H O OH and NO
% output:
% h - enthalpy (J/kg), u - internal energy (J/kg),
% v - specific volume (m^3/kg), s - entropy (J/kgK),
% Y - mole fractions of 10 species, cp - specific heat (J/kgK),
% dlvlT - partial derivative of log(v) wrt log(T)
% dlvlp - partial derivative of log(v) wrt log(p)
% ********************************************************************

[alpha,beta,gamma,delta,Afuel]=fueldata(fueltype);
switch airscheme
case 'GMcB'
   A0=airdata('GMcB_hi');
case 'Chemkin'
   A0=airdata('Chemkin_hi');
end

% Equilibrium constant data from Olikara and Borman via Ferguson
Kp=[ 0.432168E+00 -0.112464E+05  0.267269E+01 -0.745744E-04  0.242484E-08
     0.310805E+00 -0.129540E+05  0.321779E+01 -0.738336E-04  0.344645E-08
    -0.141784E+00 -0.213308E+04  0.853461E+00  0.355015E-04 -0.310227E-08
     0.150879E-01 -0.470959E+04  0.646096E+00  0.272805E-05 -0.154444E-08
    -0.752364E+00  0.124210E+05 -0.260286E+01  0.259556E-03 -0.162687E-07
    -0.415302E-02  0.148627E+05 -0.475746E+01  0.124699E-03 -0.900227E-08];

MinMol=1e-25;
tol=3e-12;

Ru=8314.34; % J/kmol.K
M=[44.01 18.02 28.008 32.000 28.01 2.018 1.009 16 17.009 30.004]'; % kg/kmol

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

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

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

```
end

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

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

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

DY3to6=2*tol*ones(4,1);
MaxIter=500;
MaxVal=max(abs(DY3to6));
Iter=0;
DoneSome=0;

while (Iter<MaxIter)&((MaxVal>tol)|(DoneSome<1))
   Iter=Iter+1;
   if Iter>2,
      DoneSome=1;
   end

   D76=0.5*c(1)/sqrt(Y(6));
   D84=0.5*c(2)/sqrt(Y(4));
   D94=0.5*c(3)*sqrt(Y(6)/Y(4));
   D96=0.5*c(3)*sqrt(Y(4)/Y(6));
   D103=0.5*c(4)*sqrt(Y(4)/Y(3));
   D104=0.5*c(4)*sqrt(Y(3)/Y(4));
   D24=0.5*c(5)*Y(6)/sqrt(Y(4));
   D26=c(5)*sqrt(Y(4));
   D14=0.5*c(6)*Y(5)/sqrt(Y(4));
   D15=c(6)*sqrt(Y(4));
   A(1,1)=1+D103;
   A(1,2)=D14+D24+1+D84+D104+D94;
   A(1,3)=D15+1;
   A(1,4)=D26+1+D76+D96;
   A(2,1)=0;
   A(2,2)=2*D24+D94-d(1)*D14;
   A(2,3)=-d(1)*D15-d(1);
   A(2,4)=2*D26+2+D76+D96;
   A(3,1)=D103;
   A(3,2)=2*D14+D24+2+D84+D94+D104-d(2)*D14;
   A(3,3)=2*D15+1-d(2)*D15-d(2);
   A(3,4)=D26+D96;
```

```
   A(4,1)=2+D103;
   A(4,2)=D104-d(3)*D14;
   A(4,3)=-d(3)*D15-d(3);
   A(4,4)=0;
   B(1)=-(sum(Y)-1);
   B(2)=-(2*Y(2)+2*Y(6)+Y(7)+Y(9)-d(1)*Y(1)-d(1)*Y(5));
   B(3)=-(2*Y(1)+Y(2)+2*Y(4)+Y(5)+Y(8)+Y(9)+Y(10)-d(2)*Y(1)-d(2)*Y(5));
   B(4)=-(2*Y(3)+Y(10)-d(3)*Y(1)-d(3)*Y(5));

   invA=inv(A);
   DY3to6=invA*B;
   MaxVal=max(abs(DY3to6));
   Y(3:6)=Y(3:6)+DY3to6/10;
   i=find(Y<MinMol);
   Y(i)=ones(length(i),1)*MinMol;

   Y(7)=c(1)*sqrt(Y(6));
   Y(8)=c(2)*sqrt(Y(4));
   Y(9)=c(3)*sqrt(Y(4)*Y(6));
   Y(10)=c(4)*sqrt(Y(4)*Y(3));
   Y(2)=c(5)*sqrt(Y(4))*Y(6);
   Y(1)=c(6)*sqrt(Y(4))*Y(5);
end

if Iter>=MaxIter
   warning('convergence failure in composition loop');
end

TdKdT=[1/T -1/T^2 1 2*T]';
dKdT=2.302585*K.*(Kp(:,[1 2 4 5])*TdKdT);
dcdT(1)=dKdT(1)/sqrt(patm);
dcdT(2)=dKdT(2)/sqrt(patm);
dcdT(3)=dKdT(3);
dcdT(4)=dKdT(4);
dcdT(5)=dKdT(5)*sqrt(patm);
dcdT(6)=dKdT(6)*sqrt(patm);
dcdp(1)=-0.5*c(1)/p;
dcdp(2)=-0.5*c(2)/p;
dcdp(5)=0.5*c(5)/p;
dcdp(6)=0.5*c(6)/p;
x1=Y(1)/c(6);
x2=Y(2)/c(5);
x7=Y(7)/c(1);
x8=Y(8)/c(2);
x9=Y(9)/c(3);
x10=Y(10)/c(4);
dfdT(1)=dcdT(6)*x1+dcdT(5)*x2+dcdT(1)*x7+dcdT(2)*x8+ ...
   dcdT(3)*x9+dcdT(4)*x10;
dfdT(2)=2*dcdT(5)*x2+dcdT(1)*x7+dcdT(3)*x9-d(1)*dcdT(6)*x1;
dfdT(3)=2*dcdT(6)*x1+dcdT(5)*x2+dcdT(2)*x8+dcdT(3)*x9+ ...
```

```
   dcdT(4)*x10-d(2)*dcdT(6)*x1;
dfdT(4)=dcdT(4)*x10-d(3)*dcdT(6)*x1;
dfdp(1)=dcdp(6)*x1+dcdp(5)*x2+dcdp(1)*x7+dcdp(2)*x8;
dfdp(2)=2*dcdp(5)*x2+dcdp(1)*x7-d(1)*dcdp(6)*x1;
dfdp(3)=2*dcdp(6)*x1+dcdp(5)*x2+dcdp(2)*x8-d(2)*dcdp(6)*x1;
dfdp(4)=-d(3)*dcdp(6)*x1;

B=-dfdT;
dYdT(3:6)=invA*B;
dYdT(1)=sqrt(Y(4))*Y(5)*dcdT(6)+D14*dYdT(4)+D15*dYdT(5);
dYdT(2)=sqrt(Y(4))*Y(6)*dcdT(5)+D24*dYdT(4)+D26*dYdT(6);
dYdT(7)=sqrt(Y(6))*dcdT(1)+D76*dYdT(6);
dYdT(8)=sqrt(Y(4))*dcdT(2)+D84*dYdT(4);
dYdT(9)=sqrt(Y(4)*Y(6))*dcdT(3)+D94*dYdT(4)+D96*dYdT(6);
dYdT(10)=sqrt(Y(4)*Y(3))*dcdT(4)+D104*dYdT(4)+D103*dYdT(3);

B=-dfdp;
dYdp(3:6)=invA*B;
dYdp(1)=sqrt(Y(4))*Y(5)*dcdp(6)+D14*dYdp(4)+D15*dYdp(5);
dYdp(2)=sqrt(Y(4))*Y(6)*dcdp(5)+D24*dYdp(4)+D26*dYdp(6);
dYdp(7)=sqrt(Y(6))*dcdp(1)+D76*dYdp(6);
dYdp(8)=sqrt(Y(4))*dcdp(2)+D84*dYdp(4);
dYdp(9)=D94*dYdp(4)+D96*dYdp(6);
dYdp(10)=D104*dYdp(4)+D103*dYdp(3);

% calculate thermodynamic properties
Tcp0=[1 T T^2 T^3 T^4]';
Th0=[1 T/2 T^2/3 T^3/4 T^4/5 1/T]';
Ts0=[log(T) T T^2/2 T^3/3 T^4/4 1]';
cp0=A0(:,1:5)*Tcp0;
h0=A0(:,1:6)*Th0;
s0=A0(:,[1:5 7])*Ts0;

% Y(1) and Y(2) reevaluated
Y(1)=(2*Y(3)+Y(10))/d(3)-Y(5);
Y(2)=(d(1)/d(3)*(2*Y(3)+Y(10))-2*Y(6)-Y(7)-Y(9))/2;
i=find(Y<MinMol);
Y(i)=ones(length(i),1)*MinMol;

% properties of mixture
h=sum(h0.*Y);
s=sum((s0-log(Y)).*Y);
cp=sum(Y.*cp0+h0.*dYdT*T);
MW=sum(Y.*M);
MT=sum(dYdT.*M);
Mp=sum(dYdp.*M);

R=Ru/MW;
v=R*T/p;
cp=R*(cp-h*T*MT/MW);
```

```
dlvlT=1+max(-T*MT/MW,0);
dlvlp=-1-max(p*Mp/MW,0);
h=R*T*h;
s=R*(-log(patm)+s);
u=h-R*T;
```

E Tadiabatic.m

```
function Tb=Tadiabatic(p,Tu,phi,f,fueltype,airscheme);
%
% Tb=Tadiabatic(p,Tu,phi,f,fueltype,airscheme)
%
% Routine for calculating the adiabatic flame temperature.
% Method involves iteratively selecting flame temperatures until
% the enthalpy of the combustion products (in equilibrium) matches
% the enthalpy of the initial gas mixture.
% farg.m is used to determine the enthalpy of the unburned mixture,
% and ecp.m is used to determine the enthalpy of the burned gas.
% ***********************************************************************
% input:
% p - pressure (Pa)
% Tu - temperature of the unburned mixture (K)
% phi - equivalence ratio
% f - residual mass fraction; set f=0 if no combustion products
% are present and f=1 if only combustion products are present
% fueltype - 'gasoline', 'diesel', etc - see fueldata.m for full list
% airscheme - 'GMcB' (Gordon and McBride) or 'Chemkin'
% output:
% Tb - temperature of the burned gas (K) - adiabatic flame temperature
% ***********************************************************************

MaxIter=50;
Tol=0.00001; % 0.001% allowable error in temperature calculation
Tb=2000; % initial estimate
DeltaT=2*Tol*Tb; % something big
Iter=0;
[hu,u,v,s,Y,cp,dlvlT,dlvlp]=farg(p,Tu,phi,f,fueltype,airscheme);

while (Iter<MaxIter)&(abs(DeltaT/Tb)>Tol)
   Iter=Iter+1;
   [hb,u,v,s,Y,cp,dlvlT,dlvlp]=ecp(p,Tb,phi,fueltype,airscheme);
   DeltaT=(hu-hb)/cp;
   Tb=Tb+DeltaT;
end

if Iter>=MaxIter
   warning('convergence failure in adiabatic flame temperature loop');
end
```

F enginedata.m

```
% enginedata.m
%
% Script file used by the function ahrind.m to
% define the engine properties and initial conditions

% ***** engine geometry **********************************************
b=0.1; % engine bore (m)
stroke=0.08; % engine stroke (m)
eps=0.25; % half stroke to rod ratio, s/2l
r=10; % compression ratio
Vtdc=pi/4*b^2*stroke/(r-1); % volume at TDC
Vbdc=pi/4*b^2*stroke+Vtdc; % volume at BDC

% ***** engine thermofluids parameters ********************************
Cblowby=0.8; % piston blowby constant (s^-1)
f=0.1; % residual fraction
fueltype='gasoline';
airscheme='GMcB';
phi=0.8; % equivalence ratio
thetas=-35*pi/180; % start of burning
thetab=60*pi/180; % burn duration angle
RPM=2000;
omega=RPM*pi/30; % engine speed in rad/s
heattransferlaw='constant'; % 'constant', or 'Woschni'
hcu=500; % unburned zone heat transfer coefficient/weighting
hcb=500; % burned zone heat transfer coefficient/weighting
Tw=420; % engine surface temperature

% ***** initial conditions ****************************************
p1=100e3;
T1=350;
theta1=-pi;
V1=Vbdc;
[h1,u1,v1,s1,Y1,cp1,dlvlT1,dlvlp1]=farg(p1,T1,phi,f,fueltype,airscheme);
mass1=Vbdc/v1;
U1=u1*mass1;
```

G RatesComp.m & RatesComb.m & RatesExp.m

```
function yprime=RatesComp(theta,y,flag);
%
% yprime=RatesComp(theta,y,flag)
%
% Function that returns the drivatives of the following 5 variables
% w.r.t. crank angle (theta) for the compression phase:
```

```
% 1) pressure; 2) unburned temperature;
% 3) work; 4) heat transfer; and 5) heat leakage.
% See Ferguson, C.R., 1986, "Internal Combustion Engines", Wiley,
% p174.

global b stroke eps r Cblowby f fueltype airscheme phi ...
   thetas thetab omega ...
   heattransferlaw hcu ...
   Tw theta1 Vtdc mass1 ...

p=y(1);
Tu=y(2);
yprime=zeros(5,1);

% mass in cylinder accounting for blowby:
mass=mass1*exp(-Cblowby*(theta-theta1)/omega);
% volume of cylinder:
V=Vtdc*(1+(r-1)/2*(1-cos(theta)+ ...
   1/eps*(1-(1-eps^2*sin(theta).^2).^0.5)));
% derivate of volume:
dVdtheta=Vtdc*(r-1)/2*(sin(theta)+ ...
   eps/2*sin(2*theta)./sqrt(1-eps^2*sin(theta).^2));

switch heattransferlaw
case 'constant'
   hcoeff=hcu;
case 'Woschni'
   upmean=omega*stroke/pi; % mean piston velocity
   C1=2.28;
   hcoeff=hcu*130*b^(-0.2)*Tu^(-0.53)*(p/100e3)^(0.8)*C1*upmean;
end

A=1/mass*(dVdtheta+V*Cblowby/omega);
Qconv=hcoeff*(pi*b^2/2+4*V/b)*(Tu-Tw);
Const1=1/omega/mass;

[h,u,v,s,Y,cp,dlvlT,dlvlp]=farg(p,Tu,phi,f,fueltype,airscheme);
B=Const1*v/cp*dlvlT*Qconv/Tu; % note typo on p174, eq. 4.76
C=0;
D=0;
E=v^2/cp/Tu*dlvlT^2+v/p*dlvlp;

yprime(1)=(A+B+C)/(D+E);
yprime(2)=-Const1/cp*Qconv+v/cp*dlvlT*yprime(1);
yprime(3)=p*dVdtheta;
yprime(4)=Qconv/omega;
yprime(5)=Cblowby*mass/omega*h;


function yprime=RatesComb(theta,y,flag);
```

```
%
% yprime=RatesComb(theta,y,flag)
%
% Function that returns the drivatives of the following 6 variables
% w.r.t. crank angle (theta) for the combustion phase:
% 1) pressure; 2) unburned temperature; 3) burned temperature;
% 4) work; 5) heat transfer; and 6) heat leakage.
% See Ferguson, C.R., 1986, "Internal Combustion Engines", Wiley,
% p174.

global b stroke eps r Cblowby f fueltype airscheme phi ...
   thetas thetab RPM omega ...
   heattransferlaw hcu hcb ...
   p1 T1 V1 Tw theta1 Vtdc Vbdc mass1

p=y(1);
Tb=y(2);
Tu=y(3);
yprime=zeros(6,1);

% mass in cylinder accounting for blowby:
mass=mass1*exp(-Cblowby*(theta-theta1)/omega);
% volume of cylinder:
V=Vtdc*(1+(r-1)/2*(1-cos(theta)+ ...
   1/eps*(1-(1-eps^2*sin(theta).^2).^0.5)));
% derivate of volume:
dVdtheta=Vtdc*(r-1)/2*(sin(theta)+ ...
   eps/2*sin(2*theta)./sqrt(1-eps^2*sin(theta).^2));
% mass fraction burned and derivative:
x=0.5*(1-cos(pi*(theta-thetas)/thetab));
dxdtheta=pi/2/thetab*sin(pi*(theta-thetas)/thetab);
if x<0.0001, x=0.0001; end;
if x>0.9999, x=0.9999; end;

switch heattransferlaw
case 'constant'
   hcoeffu=hcu;
   hcoeffb=hcb;
case 'Woschni'
   upmean=omega*stroke/pi; % mean piston velocity
   C1=2.28;
   C2=3.24e-3;
   Vs=Vbdc-Vtdc;
   k=1.3;
   pm=p1*(V1/V)^k; % motoring pressure
   hcoeffu=hcu*130*b^(-0.2)*Tu^(-0.53)*(p/100e3)^(0.8)* ...
      (C1*upmean+C2*Vs*T1/p1/V1*(p-pm))^(0.8);
   hcoeffb=hcb*130*b^(-0.2)*Tb^(-0.53)*(p/100e3)^(0.8)* ...
      (C1*upmean+C2*Vs*T1/p1/V1*(p-pm))^(0.8);
end
```

```
A=1/mass*(dVdtheta+V*Cblowby/omega);
Qconvu=hcoeffu*(pi*b^2/2+4*V/b)*(1-sqrt(x))*(Tu-Tw);
Qconvb=hcoeffb*(pi*b^2/2+4*V/b)*sqrt(x)*(Tb-Tw);
Const1=1/omega/mass;

[hu,uu,vu,s,Y,cpu,dlvlTu,dlvlpu]= ...
   farg(p,Tu,phi,f,fueltype,airscheme);
[hb,ub,vb,s,Y,cpb,dlvlTb,dlvlpb]= ...
   ecp(p,Tb,phi,fueltype,airscheme);
B=Const1*(vb/cpb*dlvlTb*Qconvb/Tb+ ...
   vu/cpu*dlvlTu*Qconvu/Tu);
C=-(vb-vu)*dxdtheta-vb*dlvlTb*(hu-hb)/cpb/Tb*(dxdtheta- ...
   (x-x^2)*Cblowby/omega);
D=x*(vb^2/cpb/Tb*dlvlTb^2+vb/p*dlvlpb);
E=(1-x)*(vu^2/cpu/Tu*dlvlTu^2+vu/p*dlvlpu);

yprime(1)=(A+B+C)/(D+E);
yprime(2)=-Const1/cpb/x*Qconvb+vb/cpb*dlvlTb*yprime(1)+ ...
   (hu-hb)/cpb*(dxdtheta/x-(1-x)*Cblowby/omega);
yprime(3)=-Const1/cpu/(1-x)*Qconvu+vu/cpu*dlvlTu*yprime(1);
yprime(4)=p*dVdtheta;
yprime(5)=Const1*mass*(Qconvb+Qconvu);
yprime(6)=Cblowby*mass/omega*((1-x^2)*hu+x^2*hb);


function yprime=RatesExp(theta,y,flag);
%
% yprime=RatesExp(theta,y,flag)
%
% Function that returns the drivatives of the following 5 variables
% w.r.t. crank angle (theta) for the expansion phase:
% 1) pressure; 2) unburned temperature;
% 3) work; 4) heat transfer; and 5) heat leakage.
% See Ferguson, C.R., 1986, "Internal Combustion Engines", Wiley,
% p174.

global b stroke eps r Cblowby f fueltype airscheme phi ...
   thetas thetab RPM omega ...
   heattransferlaw hcb ...
   p1 T1 V1 Tw theta1 Vtdc Vbdc mass1

p=y(1);
Tb=y(2);
yprime=zeros(5,1);

% mass in cylinder accounting for blowby:
mass=mass1*exp(-Cblowby*(theta-theta1)/omega);
% volume of cylinder:
V=Vtdc*(1+(r-1)/2*(1-cos(theta)+ ...
```

```
   1/eps*(1-(1-eps^2*sin(theta).^2).^0.5)));
% derivate of volume:
dVdtheta=Vtdc*(r-1)/2*(sin(theta)+ ...
   eps/2*sin(2*theta)./sqrt(1-eps^2*sin(theta).^2));

switch heattransferlaw
case 'constant'
   hcoeff=hcb;
case 'Woschni'
   upmean=omega*stroke/pi; % mean piston velocity
   C1=2.28;
   C2=3.24e-3;
   Vs=Vbdc-Vtdc;
   k=1.3;
   pm=p1*(V1/V)^k; % motoring pressure
   hcoeff=hcb*130*b^(-0.2)*Tb^(-0.53)*(p/100e3)^(0.8)* ...
      (C1*upmean+C2*Vs*T1/p1/V1*(p-pm))^(0.8);
end

A=1/mass*(dVdtheta+V*Cblowby/omega);
Qconv=hcoeff*(pi*b^2/2+4*V/b)*(Tb-Tw);
Const1=1/omega/mass;

if Tb<1000
   [h,u,v,s,Y,cp,dlvlT,dlvlp]=farg(p,Tb,phi,1,fueltype,airscheme);
else
   [h,u,v,s,Y,cp,dlvlT,dlvlp]=ecp(p,Tb,phi,fueltype,airscheme);
end
B=Const1*v/cp*dlvlT*Qconv/Tb;
C=0;
D=v^2/cp/Tb*dlvlT^2+v/p*dlvlp;
E=0;

yprime(1)=(A+B+C)/(D+E);
yprime(2)=-Const1/cp*Qconv+v/cp*dlvlT*yprime(1);
yprime(3)=p*dVdtheta;
yprime(4)=Qconv/omega;
yprime(5)=Cblowby*mass/omega*h;
```

H ahrind.m

```
% ahrind.m
%
% Script file to determine the performance of a fuel inducted engine
% based on a (user-specified) arbitrary heat release profile as a
% function of crank angle.
% Method closely follows that of:
% Ferguson, C.R., 1986, "Internal Combustion Engines", Wiley.
```

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

timestart=cputime;

global b stroke eps r Cblowby f fueltype airscheme phi ...
   thetas thetab omega ...
   heattransferlaw hcu hcb ...
   Tw theta1 Vtdc Vbdc mass1 ...
   p1 T1 V1

% load the engine parameters and initial conditions
enginedata

switch heattransferlaw
case 'Woschni'
   if (abs(hcu) > 10)|(abs(hcb) > 10),
      warning('Woschni model with weighting factor > 10')
   end
end

% integration parameters
dtheta=1*pi/180;
options=odeset('RelTol',1e-3);

% integration during compression phase
disp(['integrating over the compression phase']);
[thetacomp,pTuWQlHl]=ode45('RatesComp', ...
   [-pi:dtheta:thetas],[p1 T1 0 0 0],options);

% specification of initial conditions at start of combustion phase
% b - beginning of combustion
pb=interp1(thetacomp,pTuWQlHl(:,1),thetas);
Tub=interp1(thetacomp,pTuWQlHl(:,2),thetas);
Tbb=Tadiabatic(pb,Tub,phi,f,fueltype,airscheme);
Wb=interp1(thetacomp,pTuWQlHl(:,3),thetas);
Qlb=interp1(thetacomp,pTuWQlHl(:,4),thetas);
Hlb=interp1(thetacomp,pTuWQlHl(:,5),thetas);

% integration during combustion phase
disp(['integrating over the combustion phase']);
[thetacomb,pTbTuWQlHl]=ode45('RatesComb', ...
   [thetas:dtheta:thetas+thetab],[pb Tbb Tub Wb Qlb Hlb],options);
```

```
% specification of initial conditions at start of expansion phase
% e - end of combustion / start of expansion
pe=interp1(thetacomb,pTbTuWQlHl(:,1),thetas+thetab);
Tbe=interp1(thetacomb,pTbTuWQlHl(:,2),thetas+thetab);
We=interp1(thetacomb,pTbTuWQlHl(:,4),thetas+thetab);
Qle=interp1(thetacomb,pTbTuWQlHl(:,5),thetas+thetab);
Hle=interp1(thetacomb,pTbTuWQlHl(:,6),thetas+thetab);

% integration during expansion phase
disp(['integrating over the expansion phase']);
[thetaexp,pTbWQlHl]=ode45('RatesExp', ...
   [thetas+thetab:dtheta:pi],[pe Tbe We Qle Hle],options);

% error checks
mass4=mass1*exp(-Cblowby*2*pi/omega);
p4=interp1(thetaexp,pTbWQlHl(:,1),pi);
T4=interp1(thetaexp,pTbWQlHl(:,2),pi);
W4=interp1(thetaexp,pTbWQlHl(:,3),pi);
Ql4=interp1(thetaexp,pTbWQlHl(:,4),pi);
Hl4=interp1(thetaexp,pTbWQlHl(:,5),pi);
[h4,u4,v4,s4,Y4,cp4,dlvlT4,dlvlp4]= ...
   farg(p4,T4,phi,1,fueltype,airscheme);
U4=u4*mass4;
error1=1-v4*mass4/Vbdc;
error2=1+W4/(U4-U1+Ql4+Hl4);

% indicated mean effective pressure and thermal efficiency
imep=W4/(pi*b^2/4*stroke);
eta=W4/mass1*(1+phi*0.06548*(1-f))/phi/0.06548/(1-f)/47870/1e3;

% calcuate the heat flux in W/m^2
qcomp=calcq(thetacomp,pTuWQlHl,'comp'); % compression
qcombu=calcq(thetacomb,pTbTuWQlHl,'combu'); % combustion-unburned zone
qcombb=calcq(thetacomb,pTbTuWQlHl,'combb'); % combustion-burned zone
qexp=calcq(thetaexp,pTbWQlHl,'exp'); % expansion

timefinish=cputime;
timetaken=timefinish-timestart;

% save all data
save ahrind.mat

clear
```

I plotresults.m

```
% plotresults.m
%
```

```
% Script file to plot output from ahrind.m and compare results
% with output listed in:
% Ferguson, C.R., 1986, "Internal Combustion Engines", Wiley, p178;

load ahrind.mat; % results from ahrind.mat
load ferguson.txt; % tabulated output from ferguson p178-179

% Set some parameters to make figures look attractive
NLW=1; % normal line width
NFS=18; % normal font size
NMS=1; % normal marker size

close all;

figure(1);
plot(thetacomp*180/pi,pTuWQlHl(:,1)/1e6); hold on;
plot(thetacomb*180/pi,pTbTuWQlHl(:,1)/1e6);
plot(thetaexp*180/pi,pTbWQlHl(:,1)/1e6);
plot(ferguson(:,1),ferguson(:,4)*1e5/1e6,'o');
axis([-180 180 0 7]);
set(gca,'FontSize',NFS)
set(gca,'LineWidth',NLW)
set(gca,'XTick',[-180 -90 0 90 180]);
set(gca,'XTickLabel',[-180 -90 0 90 180]);
xlabel('crank angle (degrees ATC)')
ylabel('pressure (MPa)')
print -deps p_ahr.eps

figure(2);
plot(ferguson(:,1),ferguson(:,5),'o'); hold on;
plot(ferguson(:,1),ferguson(:,6),'s');
plot(thetacomp*180/pi,pTuWQlHl(:,2));
plot(thetacomb*180/pi,pTbTuWQlHl(:,3));
plot(thetacomb*180/pi,pTbTuWQlHl(:,2));
plot(thetaexp*180/pi,pTbWQlHl(:,2));
axis([-180 180 0 3000]);
set(gca,'FontSize',NFS)
set(gca,'LineWidth',NLW)
set(gca,'XTick',[-180 -90 0 90 180]);
set(gca,'XTickLabel',[-180 -90 0 90 180]);
xlabel('crank angle (degrees ATC)')
ylabel('temperature (K)')
legend('burned gas','unburned gas',2);
print -deps T_ahr.eps

figure(3);
nn=4; % for work plot
plot(thetacomp*180/pi,pTuWQlHl(:,nn-1)); hold on;
plot(thetacomb*180/pi,pTbTuWQlHl(:,nn));
plot(thetaexp*180/pi,pTbWQlHl(:,nn-1));
```

```
plot(ferguson(:,1),ferguson(:,7),'o');
axis([-180 180 -300 600]);
set(gca,'FontSize',NFS)
set(gca,'LineWidth',NLW)
set(gca,'XTick',[-180 -90 0 90 180]);
set(gca,'XTickLabel',[-180 -90 0 90 180]);
set(gca,'YTick',[-300 -150 0 150 300 450 600]);
set(gca,'YTickLabel',[-300 -150 0 150 300 450 600]);
xlabel('crank angle (degrees ATC)')
ylabel('work (J)')
print -deps W_ahr.eps

figure(4);
nn=5; % for heat transfer plot
plot(thetacomp*180/pi,pTuWQlHl(:,nn-1)); hold on;
plot(thetacomb*180/pi,pTbTuWQlHl(:,nn));
plot(thetaexp*180/pi,pTbWQlHl(:,nn-1));
plot(ferguson(:,1),ferguson(:,8),'o');
axis([-180 180 -50 300]);
set(gca,'FontSize',NFS)
set(gca,'LineWidth',NLW)
set(gca,'XTick',[-180 -90 0 90 180]);
set(gca,'XTickLabel',[-180 -90 0 90 180]);
xlabel('crank angle (degrees ATC)')
ylabel('heat transfer (J)')
print -deps Ql_ahr.eps

figure(5);
nn=6; % for heat leakage plot
plot(thetacomp*180/pi,pTuWQlHl(:,nn-1)); hold on;
plot(thetacomb*180/pi,pTbTuWQlHl(:,nn));
plot(thetaexp*180/pi,pTbWQlHl(:,nn-1));
plot(ferguson(:,1),ferguson(:,10),'o');
axis([-180 180 -8 0]);
set(gca,'FontSize',NFS)
set(gca,'LineWidth',NLW)
set(gca,'XTick',[-180 -90 0 90 180]);
set(gca,'XTickLabel',[-180 -90 0 90 180]);
xlabel('crank angle (degrees ATC)')
ylabel('heat leakage (J)')
print -deps Hl_ahr.eps

figure(6);
doflamearrivalplot=0;
thetaflame=-5*pi/180;
lengththeta=length(thetacomb);
plot(thetacomp*180/pi,qcomp); hold on;
if doflamearrivalplot==1,
   nflameb=min(find(thetacomb>thetaflame));
   nflameu=nflameb-1;
```

```
   plot(thetacomb(nflameu:nflameb)*180/pi, ...
      [qcombu(nflameu) qcombb(nflameb)]);
else
   nflameb=1;
   nflameu=length(thetacomb);
end
Nunburned=1:nflameu;
Nburned=nflameb:lengththeta;
plot(thetacomb(Nunburned)*180/pi,qcombu(Nunburned));
plot(thetacomb(Nburned)*180/pi,qcombb(Nburned));
plot(thetaexp*180/pi,qexp);
axis([-180 180 -0.2e6 1.2e6]);
set(gca,'FontSize',NFS)
set(gca,'LineWidth',NLW)
set(gca,'XTick',[-180 -90 0 90 180]);
set(gca,'XTickLabel',[-180 -90 0 90 180]);
xlabel('crank angle (degrees ATC)')
ylabel('heat flux (W/m^2)')
print -deps qflux_ahr.eps
```

J calcq.m

```
function q=calcq(theta,pTarray,phase);
%
% calculation of the heat flux (W/m^2) from the data generated
% by ahrind.
% theta is an array of crank angles
% pTarray is the corresponding array of pressure, Temperature,
% Work, etc data as generated by running arhind.m
% phase is a switch indicating the part of the cycle:
% 'comp' - compression phase
% 'combu' - combustion phase, unburned gas zone
% 'combb' - combustion phase, burned gas zone
% 'exp' - expansion phase

global b stroke eps r ...
   omega ...
   heattransferlaw hcu hcb ...
   Tw Vtdc Vbdc ...
   p1 T1 V1

switch phase
case 'comp'
   p=pTarray(:,1);
   T=pTarray(:,2);
   hc=hcu;
   C2=0;
case 'combu'
```

```
   p=pTarray(:,1);
   T=pTarray(:,3);
   hc=hcu;
   C2=3.24e-3;
case 'combb'
   p=pTarray(:,1);
   T=pTarray(:,2);
   hc=hcb;
   C2=3.24e-3;
case 'exp'
   p=pTarray(:,1);
   T=pTarray(:,2);
   hc=hcb;
   C2=3.24e-3;
end

switch heattransferlaw
case 'constant'
   hcoeff=hc;
case 'Woschni'
   V=Vtdc*(1+(r-1)/2*(1-cos(theta)+ ...
      1/eps*(1-(1-eps^2*sin(theta).^2).^0.5)));
   upmean=omega*stroke/pi; % mean piston velocity
   Vs=Vbdc-Vtdc;
   k=1.3;
   C1=2.28;
   pm=p1*(V1./V).^k; % motoring pressure
   hcoeff=hc*130*b^(-0.2)*T.^(-0.53).*(p/100e3).^(0.8).* ...
         (C1*upmean+C2*Vs*T1/p1/V1*(p-pm)).^(0.8);
end

q=hcoeff.*(T-Tw);
```

K ferguson.txt

```
-180.     698.   0.        1.00     NaN     350.     0.     0.     0.714     0.
-170.     695.   0.        1.01     NaN     353.    -0.    -1.     0.714    -0.14
-160.     684.   0.        1.04     NaN     357.    -1.    -2.     0.713    -0.28
-150.     666.   0.        1.08     NaN     362.    -3.    -4.     0.713    -0.42
-140.     641.   0.        1.14     NaN     369.    -6.    -4.     0.712    -0.56
-130.     609.   0.        1.23     NaN     377.   -10.    -5.     0.712    -0.69
-120.     571.   0.        1.35     NaN     386.   -15.    -6.     0.711    -0.81
-110.     527.   0.        1.50     NaN     398.   -21.    -6.     0.711    -0.93
-100.     477.   0.        1.72     NaN     413.   -29.    -7.     0.710    -1.05
 -90.     424.   0.        2.01     NaN     430.   -39.    -7.     0.710    -1.15
 -80.     368.   0.        2.43     NaN     451.   -51.    -6.     0.709    -1.25
 -70.     312.   0.        3.02     NaN     476.   -67.    -6.     0.709    -1.34
 -60.     257.   0.        3.91     NaN     507.   -85.    -5.     0.708    -1.41
```

-50.	205.	0.	5.24	NaN	544.	-109.	-4.	0.708	-1.46
-40.	160.	0.	7.28	NaN	588.	-137.	-2.	0.708	-1.49
-30.	122.	0.017	10.90	2143.	647.	-170.	-0.	0.707	-1.49
-20.	93.	0.146	20.93	2296.	752.	-212.	5.	0.707	-1.45
-10.	76.	0.371	38.59	2439.	863.	-262.	14.	0.706	-1.34
0.	70.	0.629	56.28	2514.	936.	-289.	27.	0.706	-1.15
10.	76.	0.854	61.31	2497.	952.	-253.	41.	0.705	-0.91
20.	93.	0.983	52.13	2400.	916.	-152.	57.	0.705	-0.68
30.	122.	1.000	37.80	2248.	NaN	-26.	73.	0.704	-0.53
40.	160.	1.000	26.80	2091.	NaN	94.	88.	0.704	-0.50
50.	205.	1.000	19.40	1948.	NaN	198.	104.	0.703	-0.58
60.	257.	1.000	14.51	1822.	NaN	284.	119.	0.703	-0.75
70.	312.	1.000	11.24	1714.	NaN	355.	134.	0.702	-0.99
80.	368.	1.000	8.99	1621.	NaN	411.	149.	0.702	-1.30
90.	424.	1.000	7.42	1541.	NaN	457.	165.	0.701	-1.67
100.	477.	1.000	6.29	1472.	NaN	493.	180.	0.701	-2.09
110.	527.	1.000	5.47	1412.	NaN	522.	195.	0.700	-2.54
120.	571.	1.000	4.86	1360.	NaN	545.	210.	0.700	-3.03
130.	609.	1.000	4.40	1315.	NaN	563.	225.	0.700	-3.55
140.	641.	1.000	4.05	1276.	NaN	576.	240.	0.699	-4.10
150.	666.	1.000	3.79	1241.	NaN	586.	255.	0.699	-4.67
160.	684.	1.000	3.60	1212.	NaN	593.	269.	0.698	-5.26
170.	695.	1.000	3.47	1186.	NaN	596.	283.	0.698	-5.86
180.	698.	1.000	3.39	1165.	NaN	598.	297.	0.697	-6.48