Internship Australia 2017

Centre for Hypersonics

University of Queensland, Brisbane

Extension of the Eilmer4 Gas Package

Author:
C.W. Lerink

Supervisors:
Dr. R.J. Gollan
Prof. Dr. P.A. Jacobs

April 5, 2017

The University Of Queensland
AUSTRALIA

UNIVERSITY OF TWENTE.

Abstract

Eilmer4 is the new compressible flow CFD solver developed within the Centre for Hypersonics of the University of Queensland [10]. As an open source solver Eilmer4 is constantly in development. For using Eilmer4 in methane combustion test-cases. Combustion modelers and other users involved in combustion research often make use of thermodynamic data from the GRI-Mech database. To make Eilmer4 a suitable simulation code for methane-combustion test-cases, thermodynamic data from GRI-Mech is implemented.

With a conversion file written in Lua, the thermodynamic data from GRI-Mech is stored in multiple species Lua-files which are added to the Eilmer4 gas library. To obtain a first impression of the successfulness of the conversion, C_p, T-plots are created and roughly analyzed. To get more insight in behavior of the GRI-Mech species, several test cases are used for validation.

The test cases selected all involve combustion of (stoichiometric) methane-air mixtures. The advantage of these test-cases is a high resemblance of future application of Eilmer4 in methane combustion. However, since chemical kinetics play a significant role during combustion, reaction schemes are necessary which describe the chemistry. As the DRM-19 uses GRI-Mech models [23] and existing trusted test cases are available in Eilmer3, the DRM-19 reaction scheme is used for validating Eilmer4 in the combustion test-cases. Furthermore, a research of Yungster et al. [24] shows results of a test case where an alternative reaction scheme is used (Yungster gas model). Both Yungster and DRM-19 reaction schemes are created for Eilmer4.

The first test case that is used for validation is the fixed volume reactor. Eilmer4 using DRM-19 and GRI-Mech species shows great agreements with Eilmer3. Using Eilmer4 with the DRM-19 model for the hypersonic Yungster test case also produces a convincing comparison. For correct application of the Yungster gas model in Eilmer4 more adjustments to Eilmer4 are necessary. With the DRM-19 model the application of Eilmer4 for methane-combustion test cases is validated.

Contents

1 Introduction

Monday the 28th of November 2016, a Boeing 737 takes off at Schiphol, Amsterdam to fly me to Heathrow Airport, London. My trip to Brisbane, Australia had started and after a 30 hours flight via Hong Kong, Qantas' A380's wheels touched ground Down Under, on the 30th of November.

On the 5th of December my internship at the University of Queensland (UQ) commenced. The University of Queensland was found in December 1909 and is one of Australia's leading research and educational institutions. The UQ is located in the capital of the state of Queensland, Brisbane [1]. The UQ covers three major campuses, where St. Lucia is the name of the main campus which is set on a magnificent 114-hectare site bounded by the Brisbane River, about seven kilometers from Brisbane CBD [2].

On the St. Lucia campus, the School of Mechanical and Mining Engineering is located. Research in mechanical engineering is directed within a number of area's, where hypersonic aerodynamics has been a major research activity over the last 20 years. Research in this area is captured within the Centre for Hypersonics (CfH), which is led by Professor Richard Morgan. The CfH's main interest for research lies within SCRAM-jet propulsion [3].

SCRAM stands for Supersonic Combustion ram-jets, which makes it a special case of ram-jet. Ram-jets are air-breathing jet-engines that, in comparison with conventional jet-engines, do not use a an axial compressor for increasing pressure before combustion. In ram-jets, the supersonic ($M > 1$) forward motion of the engine compresses air at the inlet to subsonic ($M < 1$), after which the flow undergoes subsonic combustion and leaves the engines in supersonic state ($M > 1$). The combustion section is where the scram-jet deviates itself from conventional ram-jets. As the name already declares, scram-jets operate via supersonic combustion ($M > 1$). As can be imagined, this demands an even higher operating velocity for the scram-jet, as the flow after compression is still supersonic. This concludes that scram-jets only operate in an efficient way within hypersonic velocities ($M > 5$) [4].

Worth mentioning is that CfH's HyShot team were the first ever to successful operate a scram-jet flight. On July 30, 2002 a scram-jet was operated at Mach 7.6 for six seconds. This flight was made possible by the experi-

ments executed in hypersonic conditions in the T4 shock tunnel at UQ's St. Lucia campus [5]. In March 2004, NASA's X-43 became the world's fastest free-flying scram-jet driven aircraft with a velocity of 9.6 times the speed of sound, recognized by the Guinness Book of Records. [6].

Within the CfH, research in hypersonic propulsion is conducted in two ways, both experimentally and by computational fluid dynamics (CFD). To conduct hypersonic experiments in an efficient way the UQ has created the state-of-the-art X-labs, where the shock tunnels are located. Here, staff and student can simulate hypersonic conditions for very short amounts of time. The X3 Shock Tunnel is captured in Figure 1 [7].

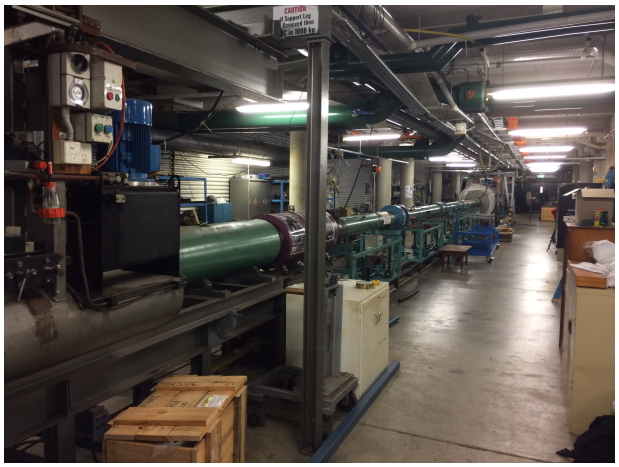


Figure 1: The X3 Shock Tunnel at St. Lucia campus, UQ. Photo by C.W. Lerink on 23-12-2016

With the shock tunnel facilities provided to validate CFD analysis, research in CFD simulations is continuously expanded. The CfH plays a prominent role in the development of the simulation code Eilmer, to improve CFD analysis of compressible flows (CFCFD). After the first editions of Eilmer, the Eilmer3 code is frequently used by students and staff to solve complex CFD problems. With Eilmer4 being still in development by the CFCFD-group, another optimized simulation code will be available soon [8].

2 Addition of GRI-Mech to Eilmer4

UQ's new computational fluid dynamics code Eilmer4 is evolving rapidly. Like it's predecessor Eilmer3, Eilmer4 is intended to be an open-source code that can be downloaded from the CFCFD portal [9]. The simulation code is mainly written in both Lua and D programming languages. As the Eilmer4 code is still in development, the addition of several thermodynamic databases is a requirement. But first, a deeper insight in, and understanding of the Eilmer4 code is obtained in this chapter.

2.1 Working with Eilmer4

As mentioned before, Eilmer4 is available as a source code from [9], making it an open source program. Eilmer4 is used for the numerical simulation of transient compressible gas-flows in 2D or 3D. First, the preparation mode is used to set a starting point for the simulation. An initial flow field (initial gas state), a database of simulation parameters (as time interval) and a multiblock grid that defines the flow domain are defined, after which the simulation computes a series of snapshots of the flow evolving. With the post-processing mode the flow data of interest can be extracted and reformatted.

During the simulation the gas flow is evolved (Eulerian/Lagrangian) following the rules of gas dynamics inside the flow domain. The flow domain consists of a mesh of finite-volume cells, where boundary conditions are applied, for instance at the walls (no-slip). Eilmer4 contains a mesh generator that creates the block-structured mesh, and is able to work with flow descriptions captured as boundary surfaces. The goal of Eilmer4 is to deal with turbulence, where high-velocity gradients are presents, as are friction effects and temperature gradients. Those effects mainly arise with the application of chemical reactions that are present in i.e. the combustion section of a scram-jet engine [10].

For familiarizing himself with the Eilmer4 simulation code the author has worked through the Eilmer4 User's Guide [10] provided by the supervisors.

2.2 The Eilmer4 Gas Package

As described at the end of section 2.1, Eilmer4 is capable of handling chemical reactions and their (presumably large) influence on the compressible flow. This is where thermodynamics and chemical behavior come in. The thermodynamics and diffusion properties of gas-mixtures are calculated with Eilmer4's gas package, where thermodynamic and chemical databases are used to express changes in for instance viscosity, enthalpy and thermodynamic state.

More specific, the gas package is seen as a support tool for the Eilmer simulation codes, and features gas models for ideal gases, mixtures of thermally perfect gases and supercritical CO_2 by calculation of thermodynamic properties, diffusion coefficients and gas species properties.

As the gas package deals with chemical reactions where different species are involved, it makes use of databases provided by NASA and other institutions. Those databases describe thermodynamic and chemical behavior and are built up by an extended range of experiments. An example of a database used is CEA, which is provided by NASA's Glenn Research Center and stands for Chemical Equilibrium with Applications (CEA). It describes itself as follows: *"CEA is a program which calculates chemical equilibrium product concentrations from any set of reactants and determines thermodynamic and transport properties for the product mixture"* [12].

For instance, the database can be used via the 'Thermo Build' option, where one can choose a species and extract various thermodynamic behavior among a certain temperature range.

For familiarizing himself with the Eilmer4 Gas Package the author has worked through the Dlang gas Package User's Guide [11] provided by the supervisors.

2.3 The Eilmer4 Programming Languages

As mentioned before, Eilmer 4 is the improved version of Eilmer3. Eilmer3 is mainly built on basis of previous simulation codes written in C++. With more viable alternative languages nowadays available, the writers decided

to rebuild the best parts of Eilmer3 into a new simulation code, Eilmer4. The both open-source programming languages used are the D programming language [13] coupled with the Lua writing language [14] to create the user interface.

Lua Language
The Lua language is developed by the department of computer science of PUC-Rio, Rio de Janeiro, Brazil. Lua is a free, small packaged and easily embedded into applications. Moreover, Lua is denoted as a very fast performing language and is used in many industry applications as Photoshop as well as in games technology, f.i. Angry Birds. The latest version is Lua 5.3.3. and is easily downloaded [15].

D Language
The D systems programming language is maintained by the D Language Foundation. It is considered as a convenient writable language, with C-like syntax and static typing. Furthermore, it is known "to combine efficiency, control and modeling power with safety and programmer productivity". Organizations from industry using D-lang are Facebook and eBay. The latest version is D 2.072.2 and is easily downloaded [13].

For familiarizing himself with the Lua Programming Language the author has worked roughly through the first edition of the Programming in Lua e-book, available online for free [16].

2.4 Thermodynamic Databases in Eilmer

The gas package is written in both D language and in Lua. The `gas-calc` program is executable with programs written in Lua which accompany the library for inclusion in D-lang programs.

As an example for the gas-libraries used in Eilmer3 and Eilmer4, NASA's CEA was stated in Section 2.2. This data is gas-specific and captures the properties of the gas species. However, besides CEA it is essential that the Eilmer4 code will make use of other databases than only CEA by NASA. The reason to be able to switch between different gas-databases is the fact that Eilmer4 will be used for different simulation purposes. Developers of Eilmer4 expect to have two groups of finite-rate chemistry users. The high-

temperature blunt body modelers will probably only be interested in data from CEA, while for instance the combustion modelers might prefer the database called GRI-Mech for some of their work. The reason behind this is that verification or comparison of different test cases in this case is made possible, as the same sources of thermodynamic or chemical data can be used in building the gas model.

The GRI-Mech database [17] is overseen by the University of Berkeley, California, and supported by the Gas Research Institute (GRI). The latest release is the GRI-Mech 3.0 version. The difference with the CEA database in provision of thermodynamic data can be explained by the build-up from their databases.

Both databases have allocated their information on thermodynamic data in 3 polynomials. This set of equations is stated in equation 1 [20].

$$\begin{aligned} \frac{C_p^o}{R} &= a_1 T^{-2} + a_2 T^{-1} + a_3 + a_4 T + a_5 T^2 + a_6 T^3 + a_7 T^4 \\ \frac{H^o}{RT} &= -a_1 T^{-2} + a_2 T^{-1} \ln T + a_3 + a_4 \frac{T}{2} + a_5 \frac{T^2}{3} + a_6 \frac{T^3}{4} + a_7 \frac{T^4}{5} + \frac{a_8}{T} \\ \frac{S^o}{R} &= a_1 \frac{T^{-2}}{2} - a_2 T^{-1} + a_3 \ln T + a_4 T + a_5 \frac{T^2}{2} + a_6 \frac{T^3}{3} + a_7 \frac{T^4}{4} + a_9 \end{aligned} \quad (1)$$

The databases differentiate from each other in the way the coefficients a_n are used. The CEA database defines all coefficients, $n = 1, .., 9$, in a three layer temperature range, as the GRI-Mech database uses only seven coefficients, $n = 3, .., 9$, in a two layer temperature range. Because in GRI-Mech, $a_1 = a_2 = 0$, the first part of the equations is equal to zero. Another observation is that CEA describes a larger temperature range in addition to GRI-Mech (the temperature range is however different for every species, still this is the general observation through the comparison of the databases).

To get an idea of how the data is delivered, one part of the database for hydrogen, H_2, is captured in Figure 2 and Figure 3 for both CEA and GRI-Mech, respectively.

From Figure 2 it can be seen that the data of CEA is built in three parts for the three temperature ranges: $200.000 < T_1 < 1000.000 < T_2 < 6000.000 < T_3 < 20000.000$, with T in Kelvin. In between the statement of the temperature range (beginning of third line, sixth line and ninth line), are

```
H2               Ref-Elm. Gurvich,1978 pt1 p103 pt2 p31.
 3 tpis78 H   2.00    0.00    0.00    0.00    0.00 0      2.01588          0.000
    200.000  1000.000 7 -2.0 -1.0  0.0  1.0  2.0  3.0  4.0  0.0         8468.102
 4.078323210E+04-8.009186040E+02 8.214702010E+00-1.269714457E-02 1.753605076E-05
-1.202860270E-08 3.368093490E-12 0.000000000E+00 2.682484665E+03-3.043788844E+01
   1000.000  6000.000 7 -2.0 -1.0  0.0  1.0  2.0  3.0  4.0  0.0         8468.102
 5.608128010E+05-8.371504740E+02 2.975364532E+00 1.252249124E-03-3.740716190E-07
 5.936625200E-11-3.606994100E-15 0.000000000E+00 5.339824410E+03-2.202774769E+00
   6000.000 20000.000 7 -2.0 -1.0  0.0  1.0  2.0  3.0  4.0  0.0         8468.102
 4.966884120E+08-3.147547149E+05 7.984121880E+01-8.414789210E-03 4.753248350E-07
-1.371873492E-11 1.605461756E-16 0.000000000E+00 2.488433516E+06-6.695728110E+02
```

Figure 2: Data with coefficients and temperature range for H_2 from CEA [18]

```
H2               TPIS78H   2              G   200.000  3500.000  1000.000    1
 3.33727920E+00-4.94024731E-05 4.99456778E-07-1.79566394E-10 2.00255376E-14    2
-9.50158922E+02-3.20502331E+00 2.34433112E+00 7.98052075E-03-1.94781510E-05    3
 2.01572094E-08-7.37611761E-12-9.17935173E+02 6.83010238E-01                   4
```

Figure 3: Data with coefficients and temperature range for H_2 from GRI-Mech [19].

the nine a-coefficients (there are ten, however the eight digit is equal to zero), that can be substituted in equation 1. Likewise can be seen for the GRI-Mech database in Figure 3. Here, the two temperature ranges are denoted at the end of the first line; $200.000 < T_{LTR} < 1000.000 < T_{HTR} < 3500.000$, with T in Kelvin (and the LTR and HTR standing for lower and higher temperature region, respectively). In lines two, three and four the a-coefficients are stated, starting with the seven for the higher temperature region, followed by a_2 to a_9 for the lower temperature region. Both databases are delivered in `.txt` format.

In the previous example hydrogen is used to describe the difference between the way the databases capture the polynomial coefficients. This is however not the only species available in the databases. Another big difference between GRI-Mech and CEA is the amount of species that is available. Where GRI-Mech delivers only 52 species, CEA captures nearly all species built from the periodic table, ions included [12]. As GRI-Mech will be the database of our greatest interest, all 52 species are given in Table 1 for an overview. Furthermore, the GRI-Mech template is given in Appendix A. In Table 1 the species already available in Eilmer4 at the time writing are underscored. Therefore, it is expected that those species will not be obtained from GRI-Mech. However, section 2.5 will tell otherwise.

$\underline{O}$	$\underline{O_2}$	$\underline{H}$	$\underline{H_2}$	$\underline{OH}$	$\underline{H_2O}$	$\underline{HO_2}$
$\underline{H_2O_2}$	C	CH	CH_2	$CH_2(S)$	CH_3	CH_4
CO	$\underline{CO_2}$	HCO	CH_2OH	CH_3O	CH_3OH	C_2H
C_2H_2	C_2H_3	C_2H_4	C_2H_5	C_2H_6	CH_2CO	$HCCO$
$HCCOH$	H_2CN	HCN	$\underline{HNO}$	$\underline{N}$	NNH	N_2O
NH	NH_2	NH_3	$\underline{NO}$	$\underline{NO_2}$	$HCNO$	$HOCN$
$HNCO$	NCO	CN	$HCNN$	$\underline{N_2}$	$\underline{AR}$	C_3H_8
C_3H_7	CH_3CHO	CH_2CHO				

Table 1: All species available in GRI-Mech. Underlined are the species already available in Eilmer4

To gain understanding in the way the coefficients are used by both CEA and GRI-Mech the author built a small Lua*-script. This script is also used to get more convenient with the* Lua *programming language and the coefficients en equations from the databases. This script is very simple and involved little real programming. It can be found in Appendix B.*

Comments are made in Lua by adding $--$ in front of the comment. On the second line, a value for the gas constant is stated, and printed to the terminal on line 5. Lines 7 to lines 12 define lower and upper temperature range coefficient from GRI-Mech. On line 15, a temperature is defined. The thermodynamic relations as in equation 1 are stated and used in line 18 to line 29, where the results are printed to the terminal. The comments from line 32 to line 38 represent the results that appeared in the terminal. From lines 42 to 82 the same procedure is followed for the CEA database. In the comments from line 84 to line 99 the data at the specified temperature is compared, and small conclusions are drawn.

The result of the comparison is mainly that the two databases, although delivering data in a different format, represent the same thermodynamic values, as to be expected. However, the main reason for comparison is to obtain understanding on how to read the databases, which was necessary for building future conversion scripts.

2.5 Conversion of the GRI-Mech species

However the GRI-Mech database may seem outdated, it still provides data for species that are not (yet) covered by the CEA database in Eilmer4. Furthermore, making the GRI-Mech species available in Eilmer4 will provide combustion modelers with their desired thermodynamic resource and enables them to validate their combustion test-cases. Therefore, it is desired that the Eilmer4 solver will provide the option that species can be selected with their thermodynamic properties from different databases. This means that all species from the GRI-Mech database have to be available in Eilmer4. To do this, a conversion file is desired.

The goal of the conversion file is to convert the thermodynamic data from GRI-Mech (see Appendix A) to a format that is readable by the source-code of Eilmer4. As declared earlier, the Eilmer4 Gas Package is currently built around the CEA database, and the `gas_prep`-program only knows how to read data from CEA-formats. Therefore, the desired ability of the conversion file is to deliver the data from GRI-Mech in a certain CEA-format that is readable for the `gas_prep`, without losing it's thermodynamic value from GRI-Mech. An example of the format of CEA-data used by Eilmer4 is denoted in Appendix C, where the species HNO is used. This format is a `species.lua`-file, and shows the template of what the output of the conversion file should look like.

On the other hand, the GRI-Mech database in Appendix A is defined as the input for the conversion file. However, there are some proceedings to be done first. From [17] the data has first to be saved as the input file: the text-file `grimech.txt`. Consequently, two important adjustments are made that allow the conversion file to read the text properly and enables Eilmer4 to handle the data properly:

- The text `CH2(S)` is changed to `CH2_S`
- The text `AR` is changed to `Ar`

The two changes have to be performed manually and are necessary for correct reading of the thermodynamic data. After saving the changes, the input file is ready to be handled by the conversion file.

Now the input file is ready to go, it is important to consider the desired output, see Appendix C. All species available in GRI-Mech (at this moment all 52) have to be converted to those `species.lua` files. This implies that the conversion file will write and save 52 separate `.lua` files. Looking at the example file found in Eilmer4's directory *dgd/src/gas/species-database*, the following lay-out is recognized.

1. a name part: `db.specie`
2. the atomic constituents part: `db.specie.atomicConstituents`
3. the molar mass part: `db.specie.M`
4. the charge part: `db.specie.charge`
5. the value for gamma part: `db.specie.gamma`
6. the thermodynamic coefficients part: `db.HNO.ceaThermoCoeffs`

The first five properties are specie specific and are not delivered by GRI-Mech, while only the sixth is. Therefore, there has been chosen to split the output file in two parts: an A-file and B-file. Merging the `specie-A.lua` and `specie-B.lua` file together later on, will eventually create the `specie.lua` file.

The A-file consists of the first five information lines (name, atomic constituents, molar mass, charge and gamma), and are manually built for every 52 species. This is done by copying the old specie files from Eilmer3, which can be found in Eilmer3's *cfcfd3/lib/gas/species*, and adjusting them to the desired A-file format. These changes are all necessary for the Eilmer4 solver code to read the specie files later on:

- Put `.db` in front of every part
- Changing the order of properties to the order of the six step list above
- Remove any comments
- Change `atomic_constituents` to `atomicConstituents`
- Delete other information the first five parts of the six step list above

- Save the file as a Lua A-file: `specie-A.lua`

The B-files are the files which have to deliver the sixth part with the thermodynamic coefficients from GRI-Mech. Looking at the output file (Appendix C) again as an example, it is desired that the polynomial coefficients are presented in the same way: a segmented list. Significant changes have been made to the name and notes of this part, as this part now contains data from GRI-Mech.

- `db.specie.ceaThermoCoeffs` is changed to `db.specie.grimechThermoCoeffs`
- add notes by adding a line `notes='datafromGRIMECH3.0'`

This makes possible that the Eilmer4 source code 'knows' that this data is from GRI-Mech, and makes it possible for the user later on to select different thermodynamic databases for their simulations. The notes make it clear for the user that GRI-Mech data is present in their gas model.

Those changes have to be considered in making the conversion file. The goal of the conversion file is to auto-generate B-files for all the species available in GRI-Mech. The conversion file itself will be written in Lua, as this is the main language used for operating and determination of input for Eilmer4. A final demand of the conversion file is that it is suitable for use in the future when the GRI-Mech database is updated or expanded.

After experience in writing Lua is obtained, the conversion file is produced for converting GRI-Mech data to CEA format (B-files): the B-files maker. The final version of the file can be found in Appendix D. To ensure proper use of the conversion file in the future, all instructions and necessary proceedings that an user must be aware of are added in `README.txt` *files. At the time of writing the script, GRI-Mech database consisted of 52 species.*

In Appendix D the conversion file `grimech-bfiles-maker.lua` is captured. As can be seen on line 5, a 'species'-table is defined with all species available in GRI-Mech. On line 6, the number of species, `nsp` is stated, after which the for-loop over all species is defined on line 8. This for loop will run over the rest of the script 52 times, to create 52 times a specie-specific B-file with output-name `specie-B.lua` (line 10). On line 24, the file with the output-name is opened in write-to modus. All write-commands in the

script will write to this file. In line 27, the input file `grimech.txt` is opened in read modus, so that all data can be read in line 28. Consequently, in line 31, the string to search the data with is defined. This is the name of the species, which was declared by the for loop. After a small check in line 32, the position of the string will be stored in variables j and k on line 35. Because the GRI-Mech database is built in a repeated pattern (Figure 3), an new variable for k $newk$ is created which indicates the end position of data for the searched specie. In line 38 the data for this specie is saved as $mydata$. This data is analyzed in line 41 by a pattern to find the right temperature ranges, subsequently storing the right temperature per variable T_0, T_1 and T_2. After in line 44 the first two polynomial coefficients are declared zero (to meet the difference in polynomials between CEA and GRI-Mech) in line 44, the other coefficients, a_n, are defined by a matching digit-pattern. The lower temperature is indicated with a normal a, as the upper temperature range is indicated by capital A. With now the variables and temperatures defined, the program continues with writing operations to state the coefficients per temperature region, fulfilling the CEA-format. This is done in lines 49 to 86. The `db.species.grimechThermoCoeffs` is traceable at line 51, just as the desired notes in line 52. At last, the writing file is closed in line 88, as is the data file. As the for loop is ended in line 91, it will start running over the remaining species in the 'species'-table.

Running the script from the Ubuntu terminal (see also Section 5.1) with `lua grimech-bfiles-maker.lua` will result in 52 different b-files, one for every species. The for-loop makes it possible to create 52 files in one run by the command line. However, it is also possible to make single b-files. This is an option when the block of commands, line 12 to line 21 is activated (by removal of the $'--'$). The $main()$ function makes it possible to operate from the terminal command line, where the name of the species has to be given as input. In case an user want to make single b-files, an extra Lua-script `grimech-single-bfiles.lua` is made, where the for-loop is removed and replaced by the activated line 12-21 part.

At this point, a folder with 52 separate A-, and B-files is present, and the possibility to generate the desired GRI-Mech species files in Lua arises. This is done by operating from the terminal, for for instance CH2: `cat CH2-A.lua CH2-B.lua > CH2.lua`. This makes the `CH2.lua` file where the B-file part in pasted after the A-file part, creating a complete specie file. For conve-

nience, an `allfiles.sh` script is automatically generated by the Lua script `make-allfiles.lua`. By running `./allfiles.sh` from the command line the A-, and B-files are converted to 52 species Lua-scripts by one command and saved in a folder named 'species'. As an example of the result, the specie file `H.lua` is captured in Appendix E.

With the species from GRI-Mech now available, they are copy-pasted to the Eilmer4 species database directory *dgd/src/gas/species-database*. However, as mentioned earlier, there are 15 species from GRI-Mech that were already available in Eilmer4. To still make this data available, the B-file-part (so basically the coefficients section) is manually copy and pasted in the existing Lua specie files for all 15 species. To be able to select between different databases in the `gas_prep` phase, a line in the gas input files can be added on the third line (after the species declaration):

- `options = {database='prefer-grimech'}`

In this way, for all species in GRI-Mech, the thermodynamic data from GRI-Mech is used. If there is no GRI-Mech data avaiable (in other words, the desired species are not in GRI-Mech), the gas-prep program switches to CEA. In this way, the gas model can be 'as GRI-Mech as possible'. After adding all GRI-Mech specie information, the `specieslist.txt` in directory is manually updated with the species names, chemical notations and filenames. Operating the `makefile` command from the terminal updates the directory and makes the species available for simulation (on that working station). After verification of the added GRI-Mech species, the species can be added to the Eilmer4 online open-source directory [9], so that users all over the world can access and work with thermodynamics from GRI-Mech. Getting the updates for Eilmer4 and `make install` Eilmer4 from the command line will update the directory */dgd-inst/*.

The conversion scripts, A-files, B-files and other support files are delivered to the supervisors. After verification of the GRI-Mech species files, they can be added to the open-source code of Eilmer4, making them available for users all over the world. Verification of the GRI-Mech data is covered in Chapter 4.

3 Chemistry for the Extended Gas Package

The goal of the addition of GRI-Mech species to the library of Eilmer4 is to provide users who run combustion simulations the option to select thermodynamic data from GRI-Mech. Combustion testcases are often performed with methane. The relatively abundance of methane on earth makes it an attractive fuel [22]. Therefore, to validate the GRI-Mech species, test cases which involve the combustion of methane are selected. Together with combustion arises the need for information on the chemical behavior of the species. As basically two gas-models will be used for validation, for both of these gas-models the reaction mechanisms have to be set. The two gas models are the DRM-19 gas model [23] and the Yungster et al. gas model (Yungster gas model) [24].

In june 2012, B. O'Flaherty researched the combustion of low-concentrated methane-air mixtures [25]. During his work a numerical study has been completed to the ignition delay of methane-air mixtures. For this combustion the DRM19 gas model and reaction scheme were used in Eilmer3. To make the DRM-19 available in Eilmer4, the files for gas preparation `gas_prep` and chemical preparation `prep_chem` are obtained from Eilmer3's *cfcfd/examples/eilmer3/2D/methane-reactor*. The `drm-19.lua` reaction file is hereafter translated to the desired format for `prep_chem` in Eilmer4. The DRM-19 reaction mechanism for the combustion of methane consists of 22 species and 84 reactions. The species used in the DRM-19 model are not all available in GRI-Mech. However, to be able to run the validation test cases as 'GRI-Mech as possible' the missing specie CH_2O is added to the Eilmer4 gas library, converting the Eilmer3 format manually to an Eilmer4 format. Furthermore, CHO and HCO are assumed to be the same species, as all properties are similar. This file consists of thermodynamic data from CEA, which is used by default as GRI-Mech is not avaiable. In the DRM-19 reaction mechanism there are 8 out of 84 reactions that are pressure dependent. Converting those to Eilmer4 need special treat, as some of the reactions were not (yet) able to be handled by the Eilmer4 source code's chemistry solver.

The paper of Yungster et al. focuses more on hypersonic application of methane combustion. Yungster computed shock-induced combustion using a detailed methane-air mechanism, with the reactions and species used captured in Table 2 on page 612 of [24]. This reaction scheme uses 21 species

and 52 reactions, of which 3 reactions are pressure dependent. This reaction mechanism is built in a reaction mechanism Lua-file in Eilmer4 format.

At this point both the thermodynamical and the chemical tools for running methane-air combustion test cases are available in Eilmer4. This means that verification of the GRI-Mech species by using them in different test-cases is possible. This is done in Chapter 4.

The reaction mechanism files and gas input files of DRM-19 and Yungster that are made for Eilmer4 simulations, have been manually built and delivered to the supervisors.

4 Validation of the Extended Gas Package

At this point, the 52 new species have been added to the Eilmer4 gas library and the `Lua` species files can be found in the updated Eilmer4 directory `dgd/src/gas/species-database`. For a certified usable species database, it is necessary to validate the new species on thermodynamic behavior. Furthermore, since the goal is to obtain a working test case for the combustion of methane, methane reaction schemes are built for Eilmer4 to cover chemical behavior. To validate the new species and reaction schemes on consistency and correct data, a comparison with an existing methane test-case in Eilmer3 can give insights in causes of errors in the Eilmer4 database or conversion. The comparison with Eilmer3 is desired, because Eilmer3 is validated. Furthermore, the Yungster gas model is not available in Eilmer3, but the DRM-19 gas model however, is available. Therefore, the DRM-19 (thermally perfect) gas model is used for comparison and validation. The idea is that when results give a good comparison between Eilmer3 and Eilmer4, the DRM-19 gas model can be used with GRI-Mech species in Eilmer4 for different methane combustion test-cases. In a nutshell:

The goal is to compare thermal and chemical behavior during methane combustion between Eilmer3 and Eilmer4 using the DRM-19 gas model. Possible causes of errors between the solvers are:

- Thermodynamic input (the conversion of the GRI-Mech database)
- Chemistry input (the converted reaction files for DRM-19)
- Chemistry integrator (the Eilmer4 chemistry solver)

All errors that would occur are useful, as those are mistakes in conversion (thermodynamics and chemistry) or the Eilmer4 source code (chemistry integrator). Since the Eilmer4 solver is still in development phase, solving those errors will improve reliability of Eilmer4.

Within a chemical reaction, like the combustion of an air-methane mixture, the concentration of the species varies over time. This is captured in equation 2.

$$\frac{dY}{dt} = \sum_{j=1}^{N_{reac}} \dot{\omega}_{i,j} \qquad with \qquad \dot{\omega}_{i,j} = f(k_f, k_b) \tag{2}$$

4.1 Thermodynamic validation with DRM-19

First, the thermodynamic behavior is analyzed. By making C_p, T-graphics the species can be analyzed for continuity and differentiability. To make the graphs, Eilmer4's gas calculator is used. A small Lua program is built from the examples in [11]. This Lua script is captured in Appendix F. This program is used to obtain the thermodynamic data for all species not available yet in Eilmer4. First, the gas is prepared with `prep-gas` by obtaining the gas model out of the input file. With the `gas-calc` and the `thermo-curves-for-species.lua`-script in Appendix F the heat capacity C_p and enthalpy h values for every temperature T are calculated, with steps of $dT = 50K$ (see line 25, Appendix F). With the function `gnuplot` the data is subsequently plotted in an `.eps`-file. This file can be translated to a plot in PDF format by `ps2pdf`. Diagrams with all non-available GRI-Mech species (mainly carbons) are given for C_p and h in Appendix G and Appendix H, respectively. As can be seen from the diagrams the graphs are all smooth. Therefore, at first sight it can be concluded that the conversion of polynomial coefficients was successful, but for a more detailed verification the values should be compared with trusted data.

The graphs in the Appendices G and H are for a general impression on the successfulness of the conversion of the GRI-Mech species. Detailed graphs and data are provided digitally and are discussed with the supervisors, with the outcome that comparison with trusted data is necessary.

The goal is to create a loop over all species to check the enthalpy values at $T = 2000K$ and $p_{atm} = 101325Pa$. The tests are performed with Eilmer3 (using certified CEA data) and Eilmer4 (using the GRI-Mech species). In Eilmer3 a Python-script `thermo-test.py` is used (delivered by the supervisors), in which the gas is updated via evaluation of the thermodynamic state for the gas model. In Eilmer4, the DRM-19 gas model is used together with the gas calculator `gas-calc`. The Lua-script for calculating the enthalpies for all species in DRM-19 in captured in Appendix I. As can be seen, from lines 7 to 12 the conditions of the test are defined. Hereafter, the gas model is created and the thermodynamic state is updated in lines 14-19. Next an output text-file is created where for all species in DRM-19 the enthalpy is written to by the for loop in combination with the 'write' command. As the Eilmer3 Python script also delivers the enthalpy values in the same text

format, a Lua-comparison script can be built to automatically give insight in differences between Eilmer3 and Eilmer4.

The Lua comparison script `ThermoComp.lua` is captured in Appendix J. In lines 1 to 7 the files with the enthalpy values per species is read. In line 17 the for loop is stated which will iterate over the species noted in line 13. Within the loop the right data per species is traced and saved for enthalpy, which happens in lines 34 and 35. In line 37 the difference is calculated with a math statement, after which the error is analyzed with an if-statement in lines 41-43, and the information is printed to the user. The result of this comparison is stated in Appendix K. The results show differences between Eilmer3 and Eilmer4. However, the order of magnitude of most enthalpy values is of the order million, which means that the significant differences are between 1-5%. The difference will be caused by a difference in the use of the thermodynamic polynomials and their coefficients. For now, the difference is considered acceptable and further validation is considered first.

4.2 Chemical validation with DRM-19

For the chemical validation the conversion of the reaction rates in Chapter 3 are checked. This will be done in the same way as the thermodynamic validation. The rates of chemical processes in physics are indicated by chemical kinetics, which is why chemical validation is often considered as a kinetic validation. For obtaining the reaction rates in Eilmer3 the chemical kinetic system is captured in the Python-script `kinetics-test.py`. With this program all rate constant values k_b and k_f (see also equation 2) for all 84 reactions in DRM-19 are determined. This is done at the same conditions as the thermodynamic validation, $T = 2000K$ and $p_{atm} = 101325Pa$.

For Eilmer4 the script `kinetics-test.lua` is built to obtain the rate constant values for the converted DRM-19 reaction scheme. In Appendix L this script is captured. Until line 17 the script resembles the Lua script used in the thermodynamic validation. Reading further, the reaction mechanism is defined in lines 19-20. Hereafter, the output file is opened in write-mode, and in line 25 the rate constants are evaluated. The for loop in lines 26-29 makes sure the rate constants are written to the output file for all reactions in DRM-19. This script is ran from the terminal command line. As can be seen, after the gas model is prepared, the chemical model (the reactions file)

is prepared subsequently. Next, the gas calculator runs the script:

- `prep-gas DRM19.inp DRM19-gas-model.lua`
- `prep-chem DRM19-gas-model.lua DRM19-reaction-scheme.lua DRM19-reac-file.lua`
- `gas-calc kinetics-test.lua`

The result is a plain text file with the rate constants for all reactions 1-84, again in the same format as the Eilmer3 Python-script delivers. At this point, the rate constants can be compared. This is done by the Lua-script called `ChemoComp.lua`, and is denoted in Appendix M. In read the results from Eilmer3 and Eilmer4 in the first seven lines. Then, in line 16, the for loop is stated in which the reaction constants are traced and captured in a variables. In lines 43 and 44 the mathematic operation to obtain the difference is done, after which the if-statement from line 48 follows the same procedure as in the thermodynamic comparison, in this case for both constants. The results are again saved in a text file, which is given in Appendix N. As can be seen, there are small differences for k_f, while on the other hand the constants k_b are large amount for reactions 8 and 15. This is unexpected, as the DRM-19 Eilmer4 scripts is directly translated from Eilmer3. This implies that there could be errors in the Eilmer4 chemistry solver.

4.3 Chemistry Solver Eilmer4

The chemistry solver in Eilmer 4 is built in D-lang and is still in development. During first attempts of simulation of the test cases 'ignition delay' and the Yungster geometry [24] an error appeared whenever the combustion of methane started. The error was obtained during the Eilmer4 run time: *Hit the minimum allowable timestep in chemistry update: dt= 1.0000e-15*. After this error was solved by tweaking the chemistry delay and flow-over-body-length, there were also small errors found by the supervisor in the Eilmer4 source code for the chemistry solver.

As the Chemistry Solver in Eilmer4 is part of the source code, Rowan Gollan took care of possible errors in this part.

4.4 Fixed Volume Reactor

After updating the Eilmer4 chemistry solver, the errors in the thermodynamics and kinetics from section 4.2 and 4.3 need evaluation. A comment is demanded on the workability of these errors, and whether they influence the reliability of working with the GRI-Mech species and DRM-19 reaction mechanism in Eilmer4. To get an insight the relatively simple fixed-volume-reactor test case is used. As the name declares, this is a constant volume reactor, in which the methane-mixture is combusted. The thermodynamic conditions are set to $T = 2000K$ and $p_{atm} = 101325Pa$, after which combustion will occur. However, this will not occur instantly due to physical ignition delay. The fixed volume reactor is a present test case in Eilmer3, used with the DRM-19 gas model. This makes it well suitable for comparison with Eilmer4 along with the GRI-Mech species and DRM-19 reaction mechanism.

The fixed volume reactor is programmed in the Lua-script `fixed-volume-reactor`, which is captured in Appendix O. In this script a gas model is created which is updated to the thermodynamic state (lines 19-27). Next, with two functions the lay-out of written data is created. In line 49 the chemistry updater is called, after which a time-step for the chemistry simulation is defined and the chemical state is updated for all times ($t < t_{end}$) while data is instantly written to the output file. With `gnuplot` the data is plotted with Paraview, and is represented by Figure 4.

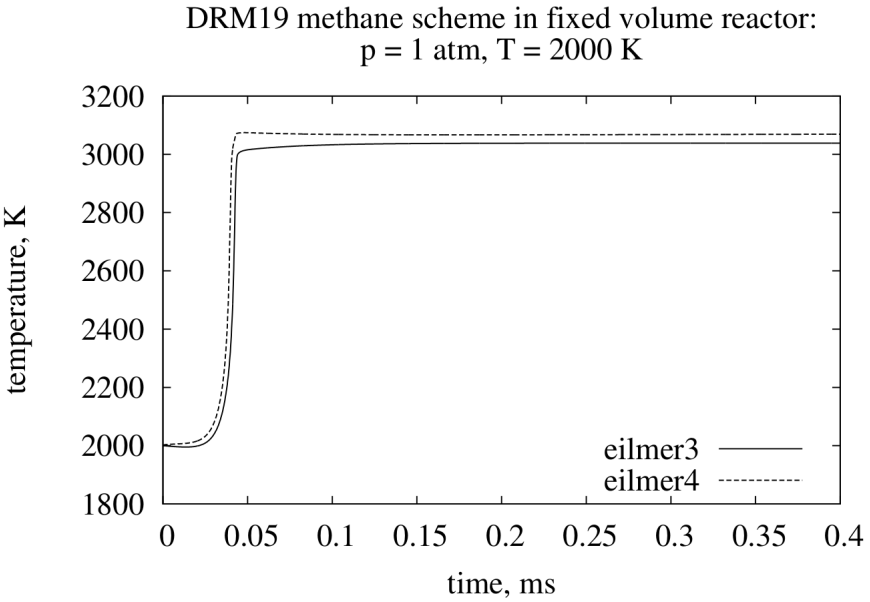


Figure 4: Diagram of the ignition delay of the fixed volume reactor test case

As seen from Figure 4, the Eilmer4 solution follows the Eilmer3 solution very well. This result is therefore taken as evidence that the errors found in the thermodynamical and chemical comparison can be justified by the fact that GRI-Mech uses coefficients in a different way as Eilmer3's CEA. At this point, with the DRM-19 test case it is assumed that the GRI-Mech species are verified. The next step is to run the DRM-19 gas model with GRI-Mech species in another test case.

In the 1994 Yungster et al. paper [24] 2D test case a cylinder is placed perpendicular to a hypersonic stoichiometric methane-air mixture flow ($M = 6.61$). This test-case will be used in Eilmer4 to run with the validated DRM-19 test case. In the next chapters first the non-reacting case will be discussed, after which the reacting test case will be issued. Eventually the goal is to run the Yungster test case along with the used Yungster methane-air gas model and reaction scheme as well.

4.5 Yungster Test Case Non-reacting with DRM-19

For building the Yungster test case a Lua-script from another test case is used after modification. This is the Lehr test case. The modified script can be found in Appendix P, and is named `yungster-simulation-reactingLehr.lua`. This script is used for both the non-reacting and reacting case, with for the non-reacting case the chemical operating left out. The basics of the script will be explained here.

Following the script after the comments in Appendix P, on lines 21-22 the geometry of the cylinder is described. Next, the gas model is created and updated with conditions similar to the Yungster test case in lines 26-30. Hereafter, in lines 32-35, the stoichiometric mixture ratio is given and the flow state is obtained. In lines 37 to lines 52 some important configuration options are set. The body flow time is the time it takes for one particle to fulfill the distance of one streamline through the domain. This time is used (multiplied by 4) to delay the reactions to give the flow time to adapt to the geometry (stationary conditions). Because the cylinder is symmetric, the axis-symmetric option is turned to `false`. In the non-reacting case, the `config.reacting` is also turned to `false`. The maximum simulation time is set to twenty times the body flow time, as the maximum amount of steps is set to 800000 (for the prevention of a possible never ending simulation). For both cases, the shock-fitting option is not activated, as this is not (yet) available in Eilmer4. On lines 55-61 the geometry of the domain is defined and expressed as function of the radius, to ensure flexibility in varying the radius for different test cases. In lines 63-67 the grid and surface of the geometry is defined, after which in lines 69-72 the grid-blocks are given boundary conditions. As can be seen, there are four blocks (2x2). The grid consists of 64 times 64 cells (comparable with the Yungster test case). In lines 75-77 history points are created to be able to check the flow evolving over time.

```
#!/bin/bash
# run.sh
prep-gas drm19.inp DRM19-gas-model.lua
prep-chem DRM19-gas-model.lua drm19-reaction-scheme.lua drm19-reac-file.lua
e4shared --prep --job=yungster-simulation-reactingLehr
e4shared --run --job=yungster-simulation-reactingLehr --verbosity=1 --max-cpus=2
e4shared --post --job=yungster-simulation-reactingLehr --vtk-xml --add-
 vars="mach,pitot,total-p,total-h,T"


```

Figure 5: Terminal command for running the `yungster-simulation-reactingLehr.lua` *script.*

The script is ran from the terminal command line with the input as stated in Figure 5. The running of the simulation takes approximately 60 minutes, using two CPU's (this is for the reacting flow). With a successful operation of the preparation, running and post-processing phases, the `.pvd`-file can be analyzed with Paraview. In Figure 6 a first impression of the temperature field is displayed. The bow shock can be clearly identified, as is the temperature at the stagnation point, which approximates $2270K$.

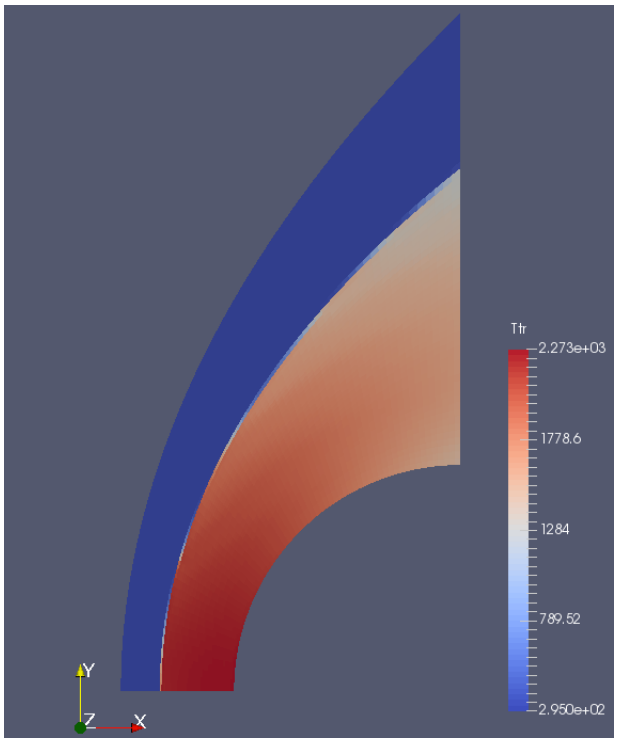


Figure 6: Non-reacting Yungster test case geometry with temperature field in K

The goal of the simulation is to make a comparison for the non-reacting case with the numerically determined results from Yungster, as seen in Figure 2, page 613 in [24]. In Figure 7 the dimensionless properties for pressure and temperature from Eilmer4, using the DRM-19 gas model, is compared for the Yungster test case. As this is the non-reacting case, the DRM-19 reaction mechanism is not used. Eilmer4 using the thermodynamic data from GRI-Mech, it can be concluded that the two graphs are very similar. This implies that the verification for the species used in DRM-19 are validated for thermodynamic behavior once again.

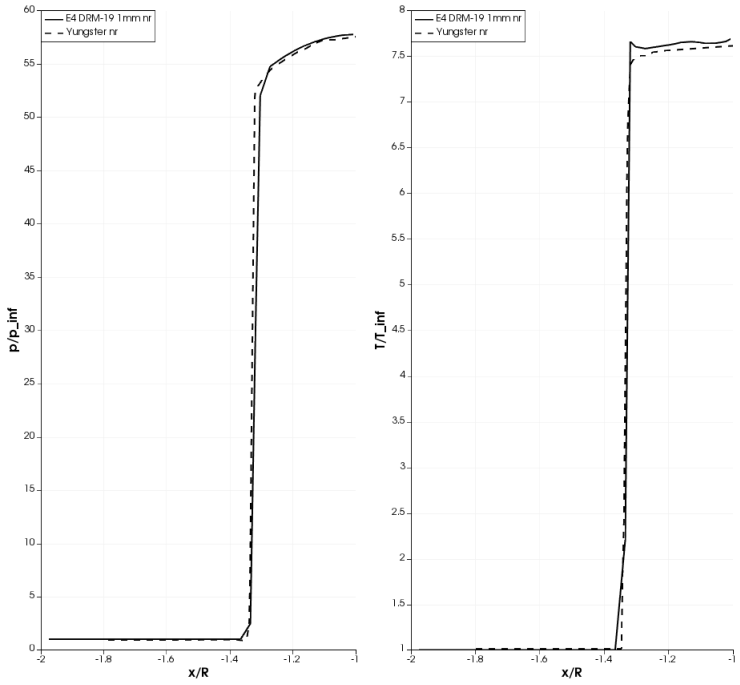


Figure 7: Non-reacting Yungster et al. test case: dimensionless pressure (left) and Dimensionless temperature (right) along stagnation streamline

4.6 Yungster Test Case Reacting with DRM-19

For the reacting case the Lua-script in Appendix P, is used with the reacting lines activated. With the combustion of the methane-air mixture activated, a combustion boundary layer will arise behind the shock. Simulations have been done for three different configurations following Yungster, for 1 mm,

3mm and 7mm diameter configurations. The output of Eilmer4, using the DRM-19 gas model and reaction mechanism is compared for the Yungster test case with the Yungster et al. data from [24]. The dimensionless pressure along stagnation streamline is displayed in Figure 8. For the 1mm and 3mm diameter test case, the results are quite similar. The small difference between the results is due to different gas and reaction models used, as for the Eilmer4 simulation the DRM-19 gas models and reaction schemes are used, where Yungster et al. makes use of the Yungster gas model and accessory reaction mechanism. In Figure 9 the results for the dimensionless temperature along stagnation streamline are displayed.

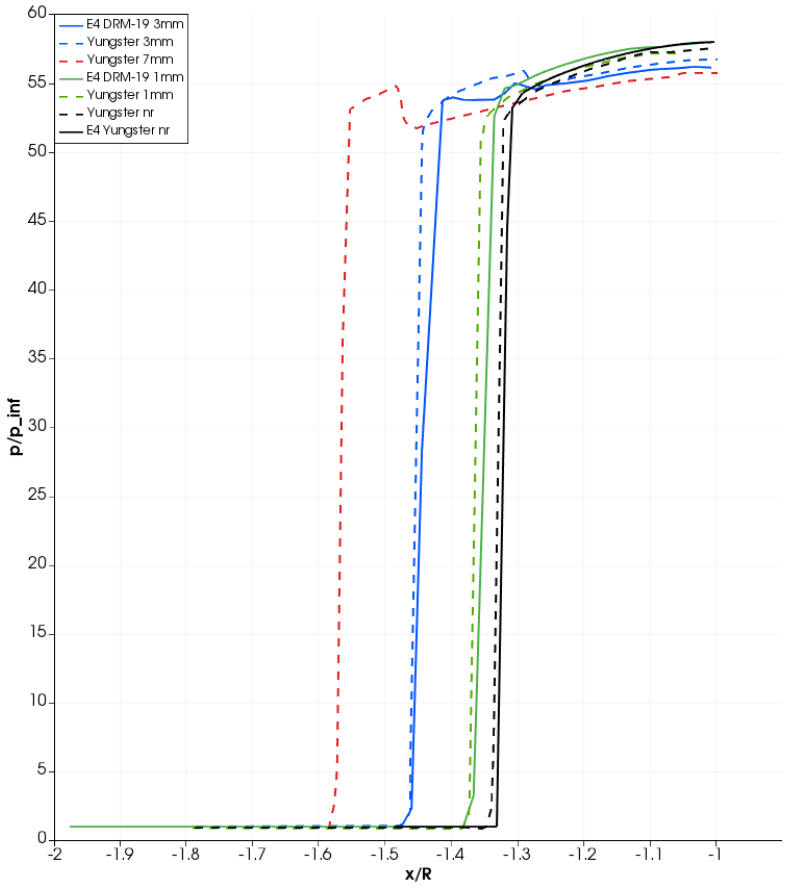


Figure 8: Dimensionless pressure along stagnation streamline graphs of Eilmer4 simulations with DRM-19, compared with data from Yungster et al.[24]

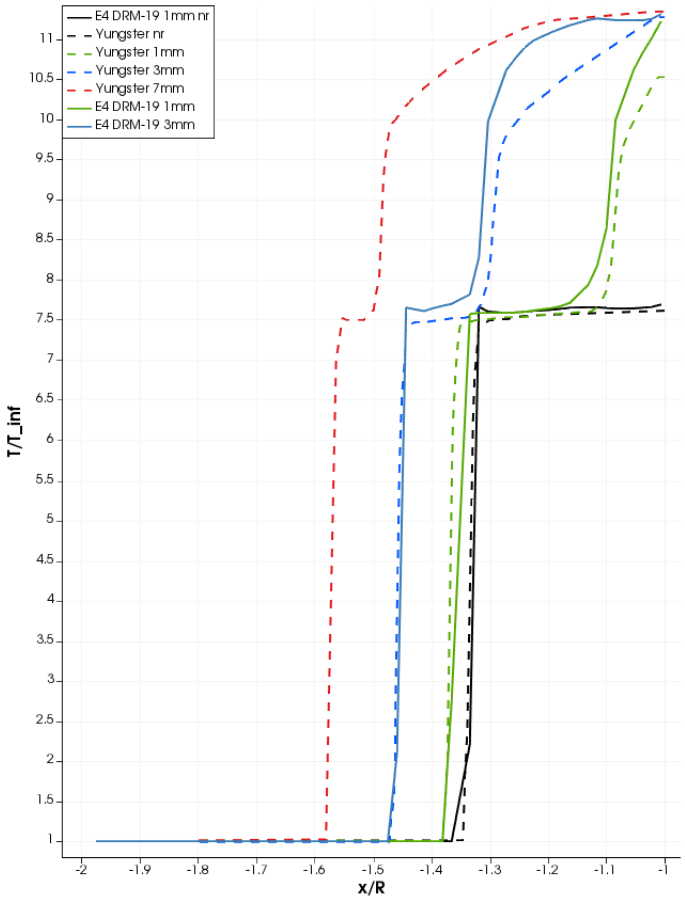


Figure 9: Dimensionless temperature along stagnation streamline graphs of Eilmer4 simulations with DRM-19, compared with data from Yungster et al.[24]

When analyzing Figure 9 it can be seen that for the 1 and 3 mm diameter test case the reacting simulation follows the Yungster results for the dimensionless temperature quite well. As the temperatures increase in the combustion layer, the results start to deviate slightly, after the shock. Still, this is considered as a convincing comparison with the Yungster et al. results, as the difference in reaction model has to be taken into account.

As can be noticed from Figures 8 and 9 the Eilmer4 results are not included for the 7mm diameter configuration. The reason behind this is that the results received were falling back to the non-reacting case. A reason for

this could be that this configuration needs a different grid, as in this case a larger shock layer arises that needs capturing. This is left for further research.

4.7 Yungster Test Case with Yungster Gas Model

In Figures 8 and 9 the results for the non-reacting case is plotted using the Yungster Gas Model (GRI-Mech) and Yungster reaction mechanism. As can be seen the black solid line shows high agreement with the dashed line, which is the data from Yungster et al. This shows that for the 19 species used in Yungster the thermodynamic properties of the gas model resembles the one used in Yungster et al. computations.

Attempts have been done in using the Yungster reaction scheme along with the Yungster gas model in the Yungster et al. test case in an Eilmer4 simulation. Unfortunately, this did not deliver the results as expected. In Figure 10 the results with activated chemistry for the Eilmer4 Yungster configuration with a rod of 3mm diameter are displayed. As can be seen, the results start to deviate towards the non-reacting case.

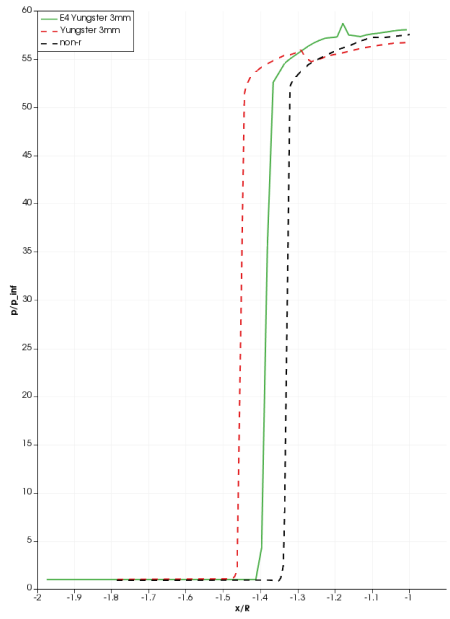


Figure 10: Dimensionless temperature along stagnation streamline graphs of Eilmer4 simulations with DRM-19, compared with data from Yungster et al.[24]

A possible reason for this might be that Eilmer4 is not (yet) capable of reading the pressure dependent reaction rates of the Yungster reaction mechanism in the right way. Therefore, before running more Eilmer4 simulation with the Yungster reaction scheme, the focus will be on improving the Eilmer4 chemistry updater.

All results of the test cases analyzed in previous sections are discussed with the supervisors. At this point, more work needs to be done on Eilmer4 to make it suitable for handling the Yungster reaction scheme. After this, a conclusion for validation can be drawn when results are obtained, in the same way as for the test cases using DRM-19.

5 Other Internship Experiences

As an occupational trainee at the School of Mechanical and Mining Engineering and the Centre for Hypersonics many other skills and knowledge is obtained, besides the results described in previous chapters. In this section, the author outlines some other new experience and knowledge that came along during the internship.

5.1 Working with Linux

As being relatively new to the world of computer programming, the author started with experience in MATLAB and some knowledge of C++ programming, both on Windows. This was the first experience for the author to work with a operating system different from Windows.

To be able to run and access the Eilmer3 and Eilmer4 packages, all students and staff work with Linux based operating systems, called Ubuntu. Ubuntu is an Linux distribution based on Debian and is developed by the company Canonical [29]. Ubuntu, like Microsoft's Windows, can be used as operating system for desktops as well for servers, laptops and mobile devices. It is knows as a fast and easy to handle system, with installing applications and software very easily. The experience is that it is more convenient to use as windows, and users operate the system often via the so-called *Terminal*, where software can be started by commands as `sudo` or be downloaded by commands as `apt_get`. In Ubuntu, the configuration and installation of Eilmer3, Eilmer4 and Lua were relatively easy and fast.

5.2 The Centre for Hypersonics Laboratory Equipment

For experiments in hypersonic conditions the Centre for Hypersonics laboratory has a large collection of shock tunnels. The University of Queensland is generally recognized for pioneering in the development of free-piston expansion tubes (shock tunnels). In the X-lab the X-expansion tubes are located, X2 and X3. In the basement of the Mansergh Shaw Building (Engineering) the T4 shock tunnel is accommodated. The shock tunnels are all free-piston driven. As an example of lay-out of the shock tunnels, a schematic overview

of X2 is given in Figure 11, as well as a photographic image of X3 in Figure 1.

The X2 and X3 Expansion Tubes

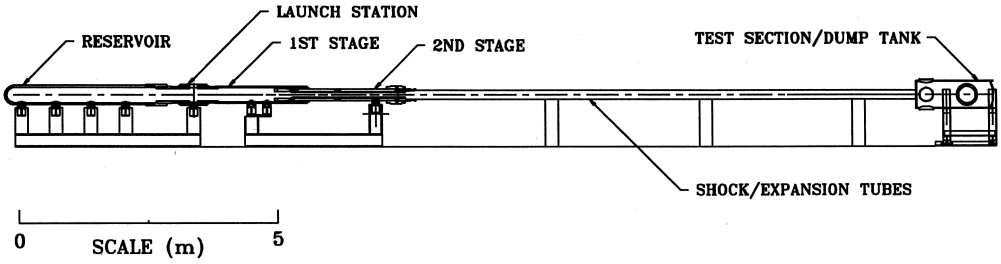


Figure 11: Schematic overview of the X2 Expansion Tube [26]

The X-lab is located in the basement of the Hawken Engineering Building, where the two X-expansion tubes can be found. The X2 is an expansion tube that has a test section diameter of 85 mm, which is larger than the previous X1. X2 consists a compound piston that exists of an lighter outside and heavier inside part, which results in the inner part creating a surplus compression in a smaller diameter. The piston is located inside the launch station in Figure 11, and is driven by the (nitrogen) gas at high pressure in the reservoir. The compound piston moves through the first stage, after which the inner part continues through the second stage. This drives a shock through the gas in the shock tube, which consists of the tube with several diaphragms that will burst and keep the shock in place. Finally, the shock arrives at the test section where it passes the model, after which it is captured in the dump tank. X2 is mainly used to measure shock stand-off distances in re-entry conditions. With high-enthalpy gas the freestream Mach number can increase to 7.3. The length of X2 is around 20 meters [26].

The big brother of X2 is the X3 shock tunnel, which has a test-section diameter of 180 mm. Just like X2, the X3 also generates a strong shock wave in a high temperature, rapidly compressed light gas (helium), by driving a heavy piston with compressed air at is pressurized in the reservoir. While the air test gas is accelerated through the shock tube and ruptures different diaphragms, an unsteady shock-wave propagates in the other direction and increases the stagnation pressure and enthalpy of the test gas [27].

The T4 Shock Tunnel The T4 Stalker Tube was first operated in 1987, and seemed an success in experimental used, due to the fact that 10,000 'shots were fired' by 2008. The T4 separates from the X-tunnels as T4 is specifically used for research to performance of scram-jets. However, the tunnel is also used for re-entry investigation, as for boundary layer transition phenomena. T4 is 26 meter in length, and can be equipped with nozzles for a Mach number range of $4 < M < 10$. In Figure 12 the test section for scram-jet geometries is denoted [28].

Figure 12: Picture of test section for scram-jet engine geometries [28]

The general observation is that the shock tunnels at UQ are operated in a very different way that the supersonic wind tunnel at the University of Twente. Where the supersonic wind tunnel does make use of a large pressure difference, it does not contain the essential parts of the shock tunnels: the diaphragms and the free-piston driver. And obviously, the dimensions are also very different, as the UT supersonic tunnel's test section is around 10-20 mm and is below five meters in length.

5.3 Seminars within the Centre for Hypersonics

During the four month internship within the Centre for Hypersonics of the University of Queensland (CfH) there have been several seminars given by the staff and students. The policy is that every week one of all the people involved in the CfH gives a presentation of one hour, after which there is some room for questions from the audience. Besides that it stimulates the interaction between research in the group, the seminars are also open for external people. For instance, students who might be interested in a PhD

in hypersonics this is a great solution. Some seminars had a very informal character, as other seminars were arranged as a final presentation of the submitting of a PhD. The following seminars have been attended with great interest during the period of 5-12-2016 to 05-04-2017:

- Presentation by Prof. Yunghwan Byun, Konkuk University, Seoul, Korea on Konkuk Aerospace Engineering research facilities and research.
- Seminar by Sholto Forbes, PhD student CfH on a multi-variable mathematical model for three stages to orbit optimization.
- Seminar by Christopher James, PhD student on the use of the X3 shock tunnel to simulate conditions of entry in gas giants atmosphere.
- Seminar by Tristan Vanyai, PhD student CfH on thermal compression in scram-jet combustion and their effect on trust generation.
- Final Seminar by Kevin Basore, PhD student CfH on scram-jet fuel-injection configurations and their effect on combustion.
- Final Seminar by Juan R. Llobet, PhD student CfH on scram-jet inlet geometry vortexes and their effect on fuel mixing.
- Final Seminar by Dawid Preller, PhD student CfH on the multidisciplinary design and optimization of pitch trimmed scram-jets.
- Final Seminar by Zachary Denman, PhD student CfH on carbon cavity flame holders in scram-jet combustion.
- Final Seminar by Timothy Cullen, PhD student CfH non-intrusive measurement for heat measurement on probes during shock tunnel operation.

5.4 New Competences and Impressions

During the four month internship new competences have been developed. Learning new programming languages can be a challenge, but is very rewarding when results and progress are noticed. Learning to work with Lua did not only result in understanding the language, but contributed in thinking in the 'programming way'. Achieve the programming output as desired

by dividing the bigger task in smaller subtasks and solving them one by one is something that came along with writing certain small Lua scripts, and may contribute in daily life where large problems are to be solved.

Being a short-term part of the Centre for Hypersonics group at the UQ contributed in new insight in research development as well. Obtaining a glimpse of the development of the Eilmer4 solver was fascinating, as it handles complicated flow and other physical phenomena. Working together with the supervisors and PhD-students on this task was very challenging. Working with new operating systems as Ubuntu and with programs like Paraview and the Engauge digitizer were new experiences that are likely to be usefully applied in future work as well.

Working is a research group also gave new insights in life of a PhD-student. At the University of Twente (master-)students are not as closely involved in the research that is done by the research department. Socializing with PhD-students within the Centre for Hypersonics and attending the weekly seminars gave a new insight in which way research is conducted, built up and thought about. Great respect and recognition is deserved for them working hard, late hours and contributing to science and engineering topics.

Another experience is the importance of communication with supervisors. The connection between an occupational trainee or intern with his or her supervisor is different from the relation between student and professor or lecturer. However, for the sake of a successful completion of research or assignment, it was learned that communication with supervisor is not only essential, but also very helpful. This experience will be of great value during future research as a graduation student and future employee or researcher.

Besides these experiences and competences, there was also the experience of living in a foreign country. Traveling through and living in Australia and New Zealand have been a mesmerizing experience. New people have been met, new cultures like the Maori and Aboriginals have been discovered and wonderful views of nature have left one speechless.

6 Acknowledgements

Brisbane, 5th of April 2017 - After four very interesting months my internship at the Centre for Hypersonics has come to an end. At the moment of writing there are two interfering feelings. One is sadness to say goodbye to the people involved in the CfH, the beautiful University of Queensland campus and my pleasant life in Brisbane. The second feeling is one which is focusing on the times ahead, a three week trip along Australia's east coast towards Darwin. But, those times ahead can not be enjoyed as much without a few people to be gratefully thanked.

Thinking about the first week at the CfH, the description 'new experience' would be an understatement. Although the office chairs and tables down under are quite similar to those at home, the desktop operating systems and programming language were undiscovered at that time. Working from a terminal, 'banter' in Lua and learning to simulate hypersonic flows in Eilmer was quite challenging, but very satisfying along the way when mastered. For this, I owe thanks to one person in general, which is my (direct) supervisor Dr. Rowan Gollan. If there were gold medals for patience, helpfulness and clear explanation they would all be rewarded to him. When the track we were on or the programming goals were unclear for me, Rowan was always there to calmly give direction. The pizza-afternoon and *Aussie barbie* at his place during my last week were, besides tasteful, also very good company.
Then there is the 'neighboring' neighbor, the Belgian PhD-student Jimmy-John Hoste, who was the first one to have a joke with during the inductions, and seated one chair down the office the remaining months. Besides being a very helpful source using the Ubuntu terminal, Paraview and Eilmer, JJ was more than great company during the one-week trip to New Zealand, our adventures in and around Brisbane and our games of tennis on the UQ campus. The feeling arises that JJ will, by all means, not forget the Tongariro Crossing, and Mt. Ngarahoe in particular, either.
Furthermore I would like to gratefully thank Prof. Peter Jacobs for 'hiring' me as an occupational trainee and having faith in me in the first place. Also I would like to send great thanks to Prof. Harry Hoeijmakers. Without your effort and transcendent global network the dream to visit *'Straya* would not have come trough in the way it has.

Gratefully yours, Cor Lerink

References

[1] *University Profile: The University of Queensland*, website of UQ, http://www.uq.edu.au/about/university-profile

[2] *University of Queensland: St. Lucia Campus*, website of UQ, http://www.uq.edu.au/about/st-lucia.

[3] *About the Centre for Hypersonics*, website of CfH, http://hypersonics.mechmining.uq.edu.au/about.

[4] *Scramjet Propulsion*, NASA Glenn Research Centre, website NASA, https://www.grc.nasa.gov/www/k-12/airplane/scramjet.html.

[5] *About the HyShot Flight Program*, UQ Centre for Hypersonics, website CfH, http://hypersonics.mechmining.uq.edu.au/hyshot-about.

[6] *Faster than a Speeding Bullet*, NASA News, June 20, 2005, website NASA, https://www.nasa.gov/home/hqnews/2005/jun/HQ_05_156_X43A_Guinness.html.

[7] *Facilities Centre for Hypersonics*, website of CfH, http://hypersonics.mechmining.uq.edu.au/facilities.

[8] *The CFCFD code collection*, Compressible Flow CFD documentation, website CFCFD-group, http://cfcfd.mechmining.uq.edu.au/intro.html.

[9] *The Eilmer4 code source*, Compressible Flow CFD Group, https://bitbucket.org/cfcfd/dgd.

[10] Peter A. Jacobs and Rowan J. Gollan *The User's Guide to the The Eilmer4 Flow Simulation Program*, School of Mechanical and Mining Engineering, University of Queensland, September 2016.

[11] Peter A. Jacobs and Rowan J. Gollan *The User's Guide to the Dlang gas Package*, School of Mechanical and Mining Engineering, University of Queensland, November 2015.

[12] *NASA Chemical Equilibrium with Applications (CEA)*, NASA Glenn Research Center, https://www.grc.nasa.gov/WWW/CEAWeb/.

[13] *D Systems Programming Language*, D Language Foundation, http://dlang.org/.

[14] *Lua The Programming Language*, PUC Rio, https://www.lua.org/.

[15] *About Lua*, PUC Rio, https://www.lua.org/about.html.

[16] Roberto Ierusalimschy *Programming in Lua*, Lua.org, ISBN 8590379817, https://www.lua.org/pil/contents.html, December 2013.

[17] *GRI-Mech*, University of Berkeley, California, http://combustion.berkeley.edu/gri-mech/index.html.

[18] *Polynomial coefficients CEA* https://cearun.grc.nasa.gov/cgi-bin2/properties-3.pl

[19] *Polynomial coefficients GRI-Mech* http://combustion.berkeley.edu/gri-mech/version30/files30/thermo30.dat.

[20] Stanford Gorden and Bonnie J. McBride *Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications*, NASA Reference Publication 1311, October 1994, https://www.grc.nasa.gov/WWW/CEAWeb/RP-1311.pdf, page 19-20.

[21] GRI-Mech Polynomials http://combustion.berkeley.edu/gri-mech/data/nasa_plnm.html.

[22] Wikipedia on Methane https://en.wikipedia.org/wiki/Methane.

[23] A. Kazakov and M. Frenklach *The DRM-19 Gas Model*, University of California, Berkeley, CA 94720-1740, USA, http://www.me.berkeley.edu/drm/

[24] S. Yungster and M. J. Rabinowitz *Computation of Shock-Induced Combustion Using a Detailed Methane-Air Mechanism*, NASA Lewis Research Center, Cleveland, Ohio 44135, Journal of Propulsion and Power, Vol. 10, No. 5, Sept. - Oct. 1994

[25] B.T. O'Flaherty *Reducing the Global Warming Potential of Coal Mine Ventilation Air by Combustion in a Free Piston Engine*, School of Mechanical and Mining Engineering, University of Queensland, June 2012

[26] *X2 Expansion Tube* Centre for Hypersonics, School of Mechanical Engineering, University of Queensland, http://hypersonics.mechmining.uq.edu.au/x2.

[27] *X3 Expansion Tube* Centre for Hypersonics, School of Mechanical Engineering, University of Queensland, http://hypersonics.mechmining.uq.edu.au/x3.

[28] *T4 Shock Tunnel* Centre for Hypersonics, School of Mechanical Engineering, University of Queensland, http://hypersonics.mechmining.uq.edu.au/t4

[29] *About Ubuntu*, Canonical Ltd., https://www.ubuntu.com/about/about-ubuntu

A The GRI-Mech database

```
THERMO
   300.000  1000.000  5000.000
! GRI-Mech Version 3.0 Thermodynamics released 7/30/99
! NASA Polynomial format for CHEMKIN-II
! see README file for disclaimer
O                 L 1/90O   1               G   200.000  3500.000
1000.000    1
 2.56942078E+00-8.59741137E-05 4.19484589E-08-1.00177799E-11 1.22833691E-15
2
 2.92175791E+04 4.78433864E+00 3.16826710E+00-3.27931884E-03 6.64306396E-06
3
-6.12806624E-09 2.11265971E-12 2.91222592E+04 2.05193346E+00
4
O2                TPIS89O   2               G   200.000  3500.000
1000.000    1
 3.28253784E+00 1.48308754E-03-7.57966669E-07 2.09470555E-10-2.16717794E-14
2
-1.08845772E+03 5.45323129E+00 3.78245636E+00-2.99673416E-03 9.84730201E-06
3
.
.
[left out]
.
.
.
 0.54041108E+01 0.11723059E-01-0.42263137E-05 0.68372451E-09-0.40984863E-13
2
-0.22593122E+05-0.34807917E+01 0.47294595E+01-0.31932858E-02 0.47534921E-04
3
-0.57458611E-07 0.21931112E-10-0.21572878E+05 0.41030159E+01
4
CH2CHO            SAND86O   1H   3C   2     G   300.000  5000.000
1000.000    1
 0.05975670E+02 0.08130591E-01-0.02743624E-04 0.04070304E-08-0.02176017E-12
2
 0.04903218E+04-0.05045251E+02 0.03409062E+02 0.10738574E-01 0.01891492E-04
3
-0.07158583E-07 0.02867385E-10 0.15214766E+04 0.09558290E+02
4
END
```

B Comparison Thermo Coefficients CEA and GRI-Mech

```
--program properties of O2 by GRI-Mech
R = 1.9872036  --cal_th .K.mol
--R =  8.31451/14.0067000e-3;
-- 1 cal_th = *4.18400 J
print("R = ",R)

-- These below are O2's LTR values
a1 = 3.78245636E+00; a2 =-2.99673416E-03; a3 = 9.84730201E-06;
a4 =-9.68129509E-09; a5 = 3.24372837E-12; a6 =-1.06394356E+03; a7 = 3.65767573E+00;
--These below are O2's HTR values
--a1 = 3.28253784E+00; a2 = 1.48308754E-03; a3 =-7.57966669E-07;
--a4 = 2.09470555E-10; a5 =-2.16717794E-14; a6 =-1.08845772E+03; a7 = 5.45323129E+00

-- State temperature input?
T = 300 --K

-- Used thermodynamic relations
CpoverR = a1 + a2*T + a3*T^2 + a4*T^3 + a5*T^4;
print("CpoverR = ",CpoverR)
Cp = CpoverR*R
print("Cp =",Cp) --> matches with GRI-Mech

SoverR  = a1*math.log(T) + a2*T + a3*T^2/2 + a4*T^3/3 + a5*T^4/4 + a7
S=SoverR*R
print("S =",S) --> matches with GRI-Mech

HoverRT = a1 + a2*T/2 + a3*T^2/3 + a4*T^3/4 + a5*T^4/5 + a6/T
H = HoverRT*R*T
print("H =",H)


--RESULT verification with GRI-Mech at T = 300K.
--uqclerin@uqclerin-OptiPlex-990:~/Test Envi$ lua O2testGRIM.lua
--R =   1.9872036
--CpoverR =     3.534572525267
--Cp =  7.0239152466717 --> matches! in cal/molK
--S =   49.075044657146 --> matches! in cal/molK
--H =   12.992055590927 --> no, it's actually delta-Hf

--Matches the test with CEA that is currently used by Eilmer?
```

```
41
42 --program properties of O2 by CEA NASA
43 R = 1.9872036*4.18400 --cal_th .K .mol
44 --R =  8.31451/14.0067000e-3;
45 -- 1 cal_th = 4.18400 J
46 print("R = ",R)
47
48 -- O2's values for T_lower = 200.0, T_upper = 1000.0 in calories. CEA
49        aa0 =-3.425563420e+04;
50        aa1 = 4.847000970e+02;
51        aa2 = 1.119010961e+00;
52        aa3 = 4.293889240e-03;
53        aa4 =-6.836300520e-07;
54        aa5 =-2.023372700e-09;
55        aa6 = 1.039040018e-12;
56        aa7 =-3.391454870e+03;
57        aa8 = 1.849699470e+01;
58
59 -- State temperature input?
60 T = 300 --K
61
62 -- Used thermodynamic relations on CEA
63 cp_on_R = aa0/(T*T) + aa1/T + aa2 + aa3*T + aa4*T*T + aa5*T*T*T + aa6*T*T*T*T;
64 print("cp_on_R = ",cp_on_R)
65 cp = cp_on_R*R
66 print("cp =",cp)
67
68 s_on_R = -aa0/(2*T*T) - aa1/T + aa2*math.log(T) + aa3*T + aa4*T*T/2 + aa5*T*T*T/3 +
69 s=s_on_R*R
70 print("s =",s)
71
72 h_on_RT = -aa0/(T*T) + aa1/T * math.log(T) + aa2 + aa3*T/2 + aa4*T*T/3 + aa5*T*T*T/4
73 h = h_on_RT*R*T
74 print("h =",h)
75
76 --RESULT verification with some link on CEA on O2 data of enthalpy, pressure coef, a
77 --uqclerin@uqclerin-OptiPlex-990:~/Test Envi$ lua O2testCEA.lua
78 --R =   1.9872036
79 --cp_on_R =     3.534485021788
80 --cp =  7.0237413594432
81 --s =   49.075044169637
82 --h =   12.991886894774
83
84 --In Joule: compared with CEA own data
85 --R =   8.3144598624
```

```
--cp_on_R =     3.534485021788
--cp =  29.387333847911 --> matches quite good (due to conv cal --> J)
--s =   205.32998480576 --> matches quite good
--h =   54.358054767736 --> matches

--CONCLUSION: does this matches with CEA that is currently used by Eilmer?
--ANSWER: pretty good
--CEA at T = 300K.
--uqclerin@uqclerin-OptiPlex-990:~/Test Envi$ lua O2testCEA.lua
--R =   1.9872036
--cp_on_R =     3.534485021788
--cp =  7.0237413594432
--s =   49.075044169637
--h =   12.991886894774
```

C Example of the species-format of Eilmer4

```
db.HNO = {}
db.HNO.atomicConstituents = {H=1,N=1,O=1}
db.HNO.M = {
   value = 31.01404e-3,
   units = 'kg/mol',
   description = 'molecular mass',
   reference = 'CEA2::thermo.inp'
}
db.HNO.charge = 0
db.HNO.gamma = {
   value = 1.325,
   units = 'non-dimensional',
   description = 'ratio of specific heats at...
   ...room temperature (= Cp/(Cp - R))',
   reference = 'using Cp evaluated from CEA2...
   ...coefficients at T=300.0 K'
}
db.HNO.ceaThermoCoeffs = {
   nsegments = 2,
   segment0 = {
      T_lower = 200.0,
      T_upper = 1000.0,
      coeffs = {
         -6.854764860e+04,
          9.551627200e+02,
         -6.000720210e-01,
          7.995176750e-03,
         -6.547079160e-07,
         -3.670513400e-09,
          1.783392519e-12,
          6.435351260e+03,
          3.048166179e+01,
      }
   },
   segment1 = {
      T_lower = 1000.0,
      T_upper = 6000.0,
      coeffs = {
         -5.795614980e+06,
          1.945457427e+04,
         -2.152568374e+01,
          1.797428992e-02,
```

```
            -4.976040670e-06,
             6.397924170e-10,
            -3.142619368e-14,
            -1.104192372e+05,
             1.818650338e+02,
        }
    }
}
```

D The GRI-Mech Conversion File

```
#!/usr/bin/env dgd-lua
-- FOR MAKING ALL B-FILES
-- Cor W. Lerink

species = {'O', 'O2', 'H', 'H2', 'OH', 'H2O', 'HO2', 'H2O2', 'C', 'CH', 'CH2', 'CH2_
nsp = 52

for i=1,nsp do
sp = species[i]
outname = string.format("%s-B.lua",sp)

--1) Give the opportunity for user to operate from the terminal command line
--function main()
--   sp = arg[1] --input: species
--   outputFile = outname -- arg[2] --input: output file name
--print("You defined ", #arg, " number of arguments.")
--print("Species Name: ",sp)
--print("Output File Name:",outputFile)
--end

--main()

--2) open write-to file with output file name
f = io.open(outname, "w")

--3) Open the thermodata in read mode and read all lines:
data = io.input("grimech.txt","r")
alldata = io.read("*all")

--4) Search for the right data via species name
search = string.format("\n%s ",sp)
print("This is the used term for searching data check: ",search) --CHECK IN TERMINAL

--5) Find the position of the species in i,j and select the used data
j, k = string.find(alldata, search)
--(Position species begins at i+1 (because of the "\n" four rows are 324 characters
newk = j+324
mydata = string.sub(alldata, j, newk)

--6) read temperature ranges
local T0, T2, T1 = mydata:match(" (%d+%.%d+)%s+(%d+%.%d+)%s+(%d+%.%d+) ")

```

```
--7) read thermo coefficients
a0 = 0.00000000E+00; a1 = 0.00000000E+00; A0 = 0.00000000E+00; A1 = 0.00000000E+00;
local A2, A3, A4, A5, A6, A7, A8, a2, a3, a4, a5, a6, a7, a8 = mydata:match("(.%d%.%

--8) write to files
-- Every species in GRI-Mech uses 2 temperature segments
local nsegments = 2
local ncoefficients = 9
f:write(string.format("db.%s.grimechThermoCoeffs = {\n", sp))
f:write(string.format("   notes = 'data from GRIMECH 3.0',\n"))
f:write(string.format("   nsegments = %d, \n", nsegments))
        seg = "segment"..0
        f:write(string.format("   segment%d ={\n",0))
        f:write(string.format("      T_lower = %.1f,\n", T0))
        f:write(string.format("      T_upper = %.1f,\n", T1))
        f:write("      coeffs = {\n")
        f:write(string.format("         %s,\n",a0))
        f:write(string.format("         %s,\n",a1))
        f:write(string.format("         %s,\n",a2))
        f:write(string.format("         %s,\n",a3))
        f:write(string.format("         %s,\n",a4))
        f:write(string.format("         %s,\n",a5))
        f:write(string.format("         %s,\n",a6))
        f:write(string.format("         %s,\n",a7))
        f:write(string.format("         %s,\n",a8))
f:write("      }\n")
f:write("   },\n")
seg = "segment"..1
        f:write(string.format("   segment%d = {\n",1))
        f:write(string.format("      T_lower = %.1f,\n", T1))
        f:write(string.format("      T_upper = %.1f,\n", T2))
        f:write("      coeffs = {\n")
        f:write(string.format("         %s,\n",A0))
        f:write(string.format("         %s,\n",A1))
        f:write(string.format("         %s,\n",A2))
        f:write(string.format("         %s,\n",A3))
        f:write(string.format("         %s,\n",A4))
        f:write(string.format("         %s,\n",A5))
        f:write(string.format("         %s,\n",A6))
        f:write(string.format("         %s,\n",A7))
        f:write(string.format("         %s,\n",A8))
f:write("      }\n")
f:write("   }\n")
f:write("}\n")

```

```
88 f:close()
89 data:close()
90
91 end
```

E Result Species File

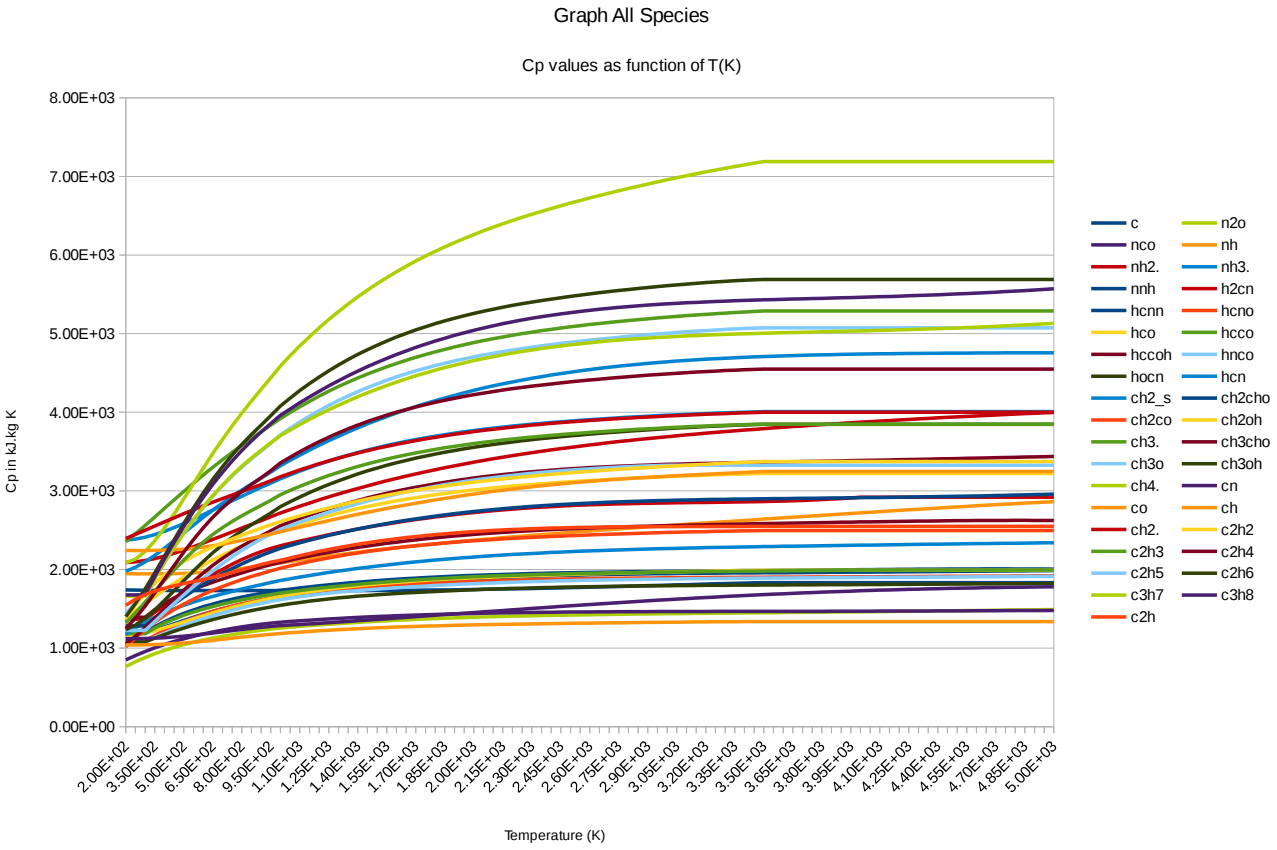
```
db.H = {}
db.H.atomicConstituents = {H=1,}
db.H.charge = 0
db.H.M = {
   value = 1.0079400e-03,
   units = 'kg/mol',
   description = 'molecular mass',
   reference = 'molecular weight from CEA2'
}
db.H.gamma = {
   value = 1.6667e+00,
   units = 'non-dimensional',
   description = 'ratio of specific heats at 300.0K',
   reference = 'evaluated using Cp/(Cp - R) from Chemkin-II coefficients'
}
db.H.grimechThermoCoeffs = {
   notes = 'data from GRIMECH 3.0',
   nsegments = 2,
   segment0 ={
      T_lower = 200.0,
      T_upper = 1000.0,
      coeffs = {
         0,
         0,
          2.50000000E+00,
          7.05332819E-13,
         -1.99591964E-15,
          2.30081632E-18,
         -9.27732332E-22,
          2.54736599E+04,
         -4.46682853E-01,
      }
   },
   segment1 = {
      T_lower = 1000.0,
      T_upper = 3500.0,
      coeffs = {
         0,
         0,
          2.50000001E+00,
         -2.30842973E-11,
          1.61561948E-14,
```

```
43            -4.73515235E-18,
44             4.98197357E-22,
45             2.54736599E+04,
46            -4.46682914E-01,
47        }
48    }
49 }
```

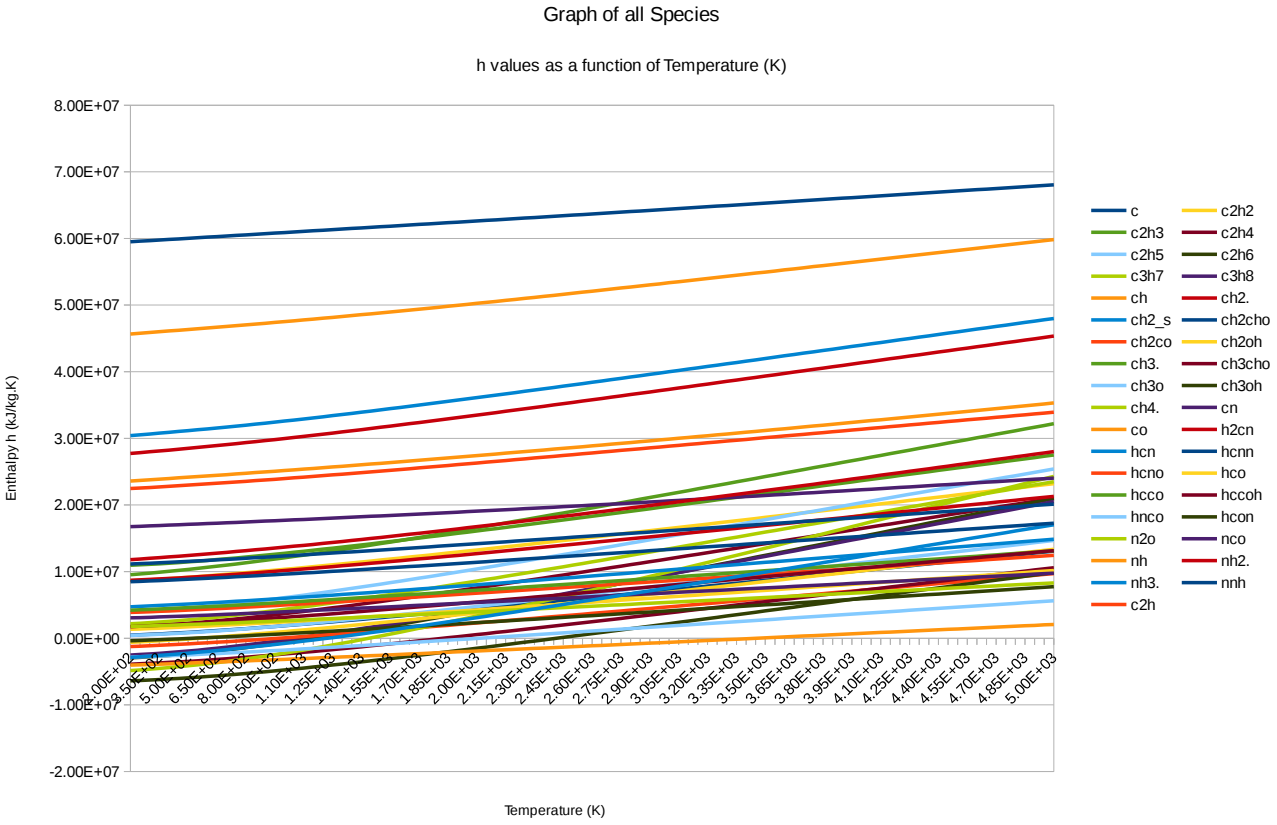
F Thermo Curves Gas Calculator

```
-- A script to output Cp and h for species over temperature range 200--5000 K.
--
-- Author: C.W. Lerink
-- Date: 2017-01-30
--
-- To run this script:
-- $ prep-gas CH4.inp CH4-gas-model.lua
-- $ gas-calc thermo-curves-for-CH4.lua
--

gasModelFile = 'CH4-gas-model.lua'
gmodel = GasModel:new{gasModelFile}

Q = GasState:new{gmodel}
Q.p = 1.0e5 -- Pa
Q.massf = {CH4=1.0}

outputFile = 'CH4-thermo.dat'
print("Opening file for writing: ", outputFile)
f = assert(io.open(outputFile, "w"))
f:write("#  1:T[K]      2:Cp[J/kg/K]     3:h[J/kg]\n")

Tlow = 200.0
Thigh = 5000.0
dT = 50.0

for T=Tlow,Thigh,dT do
   Q.T = T
   gmodel:updateThermoFromPT(Q)
   Cp = gmodel:Cp(Q)
   h = gmodel:enthalpy(Q)
   f:write(string.format(" %12.6e %12.6e %12.6e\n", T, Cp, h))
end

f:close()
print("File closed. Done.")
```

G C_p, T-Diagrams for GRI-Mech Species



H h, T-Diagrams for GRI-Mech Species



I Thermodynamic Enthalpy Calculator Lua

```
-- CWL DRM19 in Eilmer4 (.Lua)
-- Compute enthalpies of each species as a test against eilmer3.

spFile = "DRM19-gas-model.lua"
outFile = "DRM19-enthalpy-values.dat"

--Condition of Check
p0 = 101.325e3 --Pa
T0 = 2000 --K
total = 1+2+7.52
molef = {CH4=1/total, O2=2/total, N2=7.52/total}
print("Initial gas state: p = ", p0, "T =", T0)

gm = GasModel:new{spFile}
Q = gm:createGasState()
Q.T = T0
Q.p = p0
Q.massf = gm:molef2massf(molef)
gm:updateThermoFromPT(Q) --Now updated thermodynamic state from begin p,T

nsp = gm:nSpecies()

f = assert(io.open(outFile, "w"))

f:write("# speciesName \t h,J/kg\n")

for isp=0,nsp-1 do
   spName = gm:speciesName(isp)

   h = gm:enthalpy(Q, isp)
   f:write(string.format("%s \t\t % 12.6e\n",  spName, h))
end

f:close()
```

J Lua Thermo-Comparison Script

```
--1) Open the data on enthalpy in read mode and read all lines:
dataE3 = io.input("enthalpy-values-drm19-EILMER3.dat","r")
alldataE3 = io.read("*all")

dataE4 = io.input("DRM19-enthalpy-values.dat","r")
alldataE4 = io.read("*all")

--Check:
--print(alldataE3)
--print(alldataE4)

species = {'H2', 'H', 'O', 'O2', 'OH', 'H2O', 'HO2', 'CH2', 'CH2_S', 'CH3', 'CH4', '
nsp = 21
--check print(species[1])

for i = 1,nsp do
--i = 19

specie = string.format(species[i])
search = string.format("\n%s ",specie)
--check print(search)

i,j = string.find(alldataE3, search)
k,l = string.find(alldataE4, search)
--check print(i,j,k,l)
newj = j+17
newl = l+17

line3 = string.sub(alldataE3, i, newj)
line4 = string.sub(alldataE4, k, newl)
--print(line3)

he3 = line3:match(".%d%.%d+e.%d+")
he4 = line4:match(".%d%.%d+e.%d+")
--check print(he3,he4)
x = math.abs(he3-he4)
--check print(x)
seterror = 1e-3

if x < seterror then do print("CHECK, ",specie," is all good") end
else print("ERROR, ",specie," is NOT good, error = ",x)
end
```

```
43
44 end
45 print("\n Set error was ",seterror)
```

K Results Thermo-Comparison

```
ERROR,  H2       is NOT good, error =   4230
ERROR,  H        is NOT good, error =   400
ERROR,  O        is NOT good, error =   60
ERROR,  O2       is NOT good, error =   133
ERROR,  OH       is NOT good, error =   121584
ERROR,  H2O      is NOT good, error =   343
ERROR,  HO2      is NOT good, error =   16416
ERROR,  CH2      is NOT good, error =   34660
CHECK,  CH2_S    is all good
ERROR,  CH3      is NOT good, error =   16180
CHECK,  CH4      is all good
CHECK,  CO       is all good
ERROR,  CO2      is NOT good, error =   78
CHECK,  CHO      is all good
CHECK,  CH2O     is all good
CHECK,  CH3O     is all good
CHECK,  C2H4     is all good
CHECK,  C2H5     is all good
CHECK,  C2H6     is all good
ERROR,  N2       is NOT good, error =   154
CHECK,  Ar       is all good

 Set error was  0.001
```

L Reaction Rate Calculator Lua

```
--C.W. Lerink

spFile = "DRM19-gas-model.lua"
reacFile = "DRM19-reac-file.lua"
outFile = "DRM19-rate-constant-values.dat"

--Condition of Check
p0 = 101.325e3 --Pa
T0 = 2000 --K
total = 1+2+7.52
molef = {CH4=1/total, O2=2/total, N2=7.52/total}
print("Initial gas state: p = ", p0, "T =", T0)

gm = GasModel:new{spFile}
Q = gm:createGasState()
Q.T = T0
Q.p = p0
Q.massf = gm:molef2massf(molef)
gm:updateThermoFromPT(Q) --Now updated thermodynamic state from begin p,T

rmech = ReactionMechanism:new{filename=reacFile, gasmodel=gm}
nReac = rmech:nReactions()

f = assert(io.open(outFile, "w"))
f:write("reac no. k_f \t\t k_b\n")

rmech:evalRateConstants(Q)
for i=0,nReac-1 do
   f:write(string.format("%d \t %12.6e \t %12.6e\n",
                         i+1, rmech:k_f(i), rmech:k_b(i)))
end

f:close()
```

M Lua Chemical-Comparison Script

```
-- C.W. Lerink
--1) Open the data on enthalpy in read mode and read all lines:
dataE3 = io.input("rate-constant-values-drm19-EILMER3.dat","r")
alldataE3 = io.read("*all")

dataE4 = io.input("DRM19-rate-constant-values.dat","r")
alldataE4 = io.read("*all")

--Check:
--print(alldataE3)
--print(alldataE4)

nsr = 84


for i = 1,nsr do
-- i = 1
-- print("For reaction ",i) gets messy

search = string.format("\n%s ",i)


a,j = string.find(alldataE3, search)
k,l = string.find(alldataE4, search)
--print(a,j,k,l)
newj = j+30
newl = l+30
--print(newj, newl)

line3 = string.sub(alldataE3, j, newj)
line4 = string.sub(alldataE4, l, newl)
--print(line3,line4)
poskf1 = 4
poskf2 = poskf1 + 11
poskb1 = poskf2 + 4
poskb2 = poskb1 + 11

kfe3 = string.sub(line3, poskf1, poskf2)
kbe3 = string.sub(line3, poskb1, poskb2)
kfe4 = string.sub(line4, poskf1, poskf2)
kbe4 = string.sub(line4, poskb1, poskb2)

```

```
xkf = math.abs(kfe3-kfe4)
xkb = math.abs(kbe3-kbe4)
--print(xkf,xkb)
seterror = 1e-3

if xkf < seterror and xkb < seterror
then do print("CHECK, reaction ",i," is all good") end
else print("ERROR, reaction ",i," is NOT good, error kf = ",xkf," and error kb = ",x
end

end
print("\nSet error was ",seterror)
print("\nThis data is created using position searching")
```

N Results Kinetics-Comparison

```
CHECK, reaction          1         is all good
ERROR, reaction          2         is NOT good, error kf =         58
and error kb =          37
CHECK, reaction          3         is all good
ERROR, reaction          4         is NOT good, error kf =         0
and error kb =          0.00939068
CHECK, reaction          5         is all good
ERROR, reaction          6         is NOT good, error kf =         0
and error kb =          0.5968
ERROR, reaction          7         is NOT good, error kf =         120
and error kb =          5168.1
ERROR, reaction          8         is NOT good, error kf =         0.0012000000000398
and error kb =          1358662999.9985
CHECK, reaction          9         is all good
ERROR, reaction          10        is NOT good, error kf =         0
and error kb =          8524285999.9904
ERROR, reaction          11        is NOT good, error kf =         80
and error kb =          0.049999999999272
ERROR, reaction          12        is NOT good, error kf =         10
and error kb =          4.4121
ERROR, reaction          13        is NOT good, error kf =         0
and error kb =          0.0050906
ERROR, reaction          14        is NOT good, error kf =         360
and error kb =          1.6000000000058
ERROR, reaction          15        is NOT good, error kf =         0.00097000000000058
and error kb =          26308499999970
ERROR, reaction          16        is NOT good, error kf =         0.23200000000088
and error kb =          92
CHECK, reaction          17        is all good
CHECK, reaction          18        is all good
CHECK, reaction          19        is all good
CHECK, reaction          20        is all good
CHECK, reaction          21        is all good
ERROR, reaction          22        is NOT good, error kf =         43
and error kb =          166
CHECK, reaction          23        is all good
CHECK, reaction          24        is all good
CHECK, reaction          25        is all good
CHECK, reaction          26        is all good
CHECK, reaction          27        is all good
ERROR, reaction          28        is NOT good, error kf =         30
and error kb =          9.0000000000146e-05
```

```
29 ERROR, reaction          29        is NOT good, error kf =         100
   and error kb =         0.00070000000005166
30 ERROR, reaction          30        is NOT good, error kf =         0.60000000003492
   and error kb =         1.320835
31 ERROR, reaction          31        is NOT good, error kf =         4
   and error kb =         5.99
32 ERROR, reaction          32        is NOT good, error kf =         143
   and error kb =         7357.1
33 ERROR, reaction          33        is NOT good, error kf =         0.039999999997235
   and error kb =         0.0005000000000166
34 CHECK, reaction          34        is all good
35 ERROR, reaction          35        is NOT good, error kf =         0.0033000000000243
   and error kb =         80
36 ERROR, reaction          36        is NOT good, error kf =         130
   and error kb =         0.15000000000146
37 ERROR, reaction          37        is NOT good, error kf =         0
   and error kb =         423.27
38 ERROR, reaction          38        is NOT good, error kf =         0.47999999999593
   and error kb =         39
39 ERROR, reaction          39        is NOT good, error kf =         6.5999999999767
   and error kb =         0.016000000000076
40 ERROR, reaction          40        is NOT good, error kf =         330
   and error kb =         2.2999999999884
41 ERROR, reaction          41        is NOT good, error kf =         6.3200000000017e-09
   and error kb =         0.093299999999999
42 ERROR, reaction          42        is NOT good, error kf =         41
   and error kb =         4.7000000000698
43 ERROR, reaction          43        is NOT good, error kf =         15
   and error kb =         2.5999999999767
44 ERROR, reaction          44        is NOT good, error kf =         20
   and error kb =         9.9999999996214e-06
45 ERROR, reaction          45        is NOT good, error kf =         0
   and error kb =         3.18402
46 CHECK, reaction          46        is all good
47 ERROR, reaction          47        is NOT good, error kf =         21
   and error kb =         31828.8
48 ERROR, reaction          48        is NOT good, error kf =         0
   and error kb =         689170
49 ERROR, reaction          49        is NOT good, error kf =         38
   and error kb =         754.02
50 ERROR, reaction          50        is NOT good, error kf =         0.099999999976717
   and error kb =         2.4408899999973e+17
51 CHECK, reaction          51        is all good
52 ERROR, reaction          52        is NOT good, error kf =         20
   and error kb =         0.0019999999999527
```

```
53 ERROR, reaction          53        is NOT good, error kf =         30
   and error kb =         0.020000000000437
54 CHECK, reaction          54        is all good
55 ERROR, reaction          55        is NOT good, error kf =         0
   and error kb =         0.86515
56 ERROR, reaction          56        is NOT good, error kf =         0
   and error kb =         2466.4
57 ERROR, reaction          57        is NOT good, error kf =         12.799999999988
   and error kb =         1290648999998.6
58 ERROR, reaction          58        is NOT good, error kf =         18
   and error kb =         0.00405808
59 ERROR, reaction          59        is NOT good, error kf =         3.2000000000116
   and error kb =         64431.3
60 ERROR, reaction          60        is NOT good, error kf =         0
   and error kb =         353.482
61 ERROR, reaction          61        is NOT good, error kf =         14
   and error kb =         11010.86
62 ERROR, reaction          62        is NOT good, error kf =         10
   and error kb =         91381.4
63 ERROR, reaction          63        is NOT good, error kf =         7
   and error kb =         54828.8
64 CHECK, reaction          64        is all good
65 CHECK, reaction          65        is all good
66 ERROR, reaction          66        is NOT good, error kf =         0
   and error kb =         66037
67 ERROR, reaction          67        is NOT good, error kf =         0
   and error kb =         212549
68 ERROR, reaction          68        is NOT good, error kf =         10
   and error kb =         0.47916
69 ERROR, reaction          69        is NOT good, error kf =         20
   and error kb =         1759.5
70 ERROR, reaction          70        is NOT good, error kf =         0
   and error kb =         63764.8
71 ERROR, reaction          71        is NOT good, error kf =         0
   and error kb =         49594.9
72 ERROR, reaction          72        is NOT good, error kf =         0
   and error kb =         1.2182019999986
73 ERROR, reaction          73        is NOT good, error kf =         0.75
   and error kb =         1271470
74 ERROR, reaction          74        is NOT good, error kf =         0.046000000000276
   and error kb =         0.000102734
75 ERROR, reaction          75        is NOT good, error kf =         2.2999999999884
   and error kb =         5566.3
76 ERROR, reaction          76        is NOT good, error kf =         10.79999999993
   and error kb =         4132000
```

```
77 ERROR, reaction          77        is NOT good, error kf =         0
   and error kb =          0.0076171
78 ERROR, reaction          78        is NOT good, error kf =         12
   and error kb =          494.09
79 ERROR, reaction          79        is NOT good, error kf =         3.5
   and error kb =          503.14
80 ERROR, reaction          80        is NOT good, error kf =         370
   and error kb =          0.021399999999971
81 ERROR, reaction          81        is NOT good, error kf =         30
   and error kb =          0.0017900000000139
82 ERROR, reaction          82        is NOT good, error kf =         3
   and error kb =          0.0010000000002037
83 ERROR, reaction          83        is NOT good, error kf =         70
   and error kb =          0.01299999999992
84 ERROR, reaction          84        is NOT good, error kf =         1.7000000000116
   and error kb =          0.11999999999898
85
86 Set error was   0.001
87
88 This data is created using position searching
```

O Fixed Volume Reactor Script

```
-- Author: Rowan J. Gollan
-- Date: 2017-03-23
--
-- A simple fixed-volume reactor.
--
-- This script is designed to run with the gas-calc program
-- that comes as part of the dgd collection. To run this and
-- capture the output in a data file, do:
--
-- > gas-calc fixed-volume-reactor.lua > output.dat
--
-- This script is a conversion of Peter Jacobs'
-- fixed_volume_reactor.py that is part of the cfcfd3
-- code collection. That script is, in turn, a cut-down
-- version of Brendan O'Flaherty's master program for
-- testing many versions of the chemical reactor.
--

gmodel = GasModel:new{'drm19-gas-model.lua'}
Q = gmodel:createGasState()
total = 1+2+7.52
molef = {CH4=1/total, O2=2/total, N2=7.52/total}
Q.T = 2000.0 -- K
Q.p = 101.325e3 -- Pa
Q.massf = gmodel:molef2massf(molef)
gmodel:updateThermoFromPT(Q)
nsp = gmodel:nSpecies()

function writeHeader()
   str = "# t, T_0, p_0, T, p, rho"
   for isp=0,nsp-1 do
      str = str..", "..gmodel:speciesName(isp)
   end
   print(str)
   return
end

function writeData(t)
   str = string.format("%.5e %.5e %.5e %.5e %.5e %.12e ",
                       t, Q.T, Q.p, Q.T, Q.p, Q.rho)
   molef = gmodel:massf2molef(Q)
   for isp=0,nsp-1 do
```

```
      str = str..string.format(" %.12e", molef[gmodel:speciesName(isp)])
   end
   print(str)
   return
end

chem = ChemistryUpdate:new{filename='drm19.lua', gasmodel=gmodel}

tEnd = 4.0e-4
dtChem = -1.0
dt = tEnd/2000
t = 0.0

writeHeader()
writeData(t)

while t < tEnd do
   dtChem = chem:updateState(Q, dt, dtChem, gmodel)
   gmodel:updateThermoFromRHOE(Q)
   t = t + dt
   writeData(t)
end

print("# Done.")
```

P Yungster Test Case Script

```
-- Author: Rowan J. Gollan (modified by Cor. W. Lerink)
-- Date: 2016-01-27
--
-- This script is used to setup a simlulation of Lehr's
-- hemispherical projectile fired into a detonable gas.
--
-- Reference:
--   Lehr, H. (1972)
--   Acta Astronautica, 17, pp.589--597
--
-- History:
--   2017-02-23 : CWL updated it for Yungster Test Case
--   2016-01-27 : RJG updated for eilmerD Lua input
--   2015-03-15 : RJG re-worked eilmer3 example for M = 3.55
--   2010-02-27 : PJ adapted for eilmer3
--                bits taken from sphere-heat-transfer and
--                mbcns2/lehr_sphere
--

config.title = "2D cylinder in an M=6.61 reacting flow"
D = 1e-3 --m
R = D/2 --nose radis, metres

-- free stream conditions
-- taken from Table 1, Entry 5
nsp, nmodes, gmodel = setGasModel('DRM19-gas-model.lua')
p_inf = 51000 -- Pa
M_inf = 6.61 -- m/s (M=6.61)
T_inf = 295 -- K
u_inf = 2330.0 --m/s
-- Stoichometric combustion of CH4: 1 CH4 + 2 O2 + 7.52 N2 (total 10.52) --> complet
molef_inf = {CH4=0.095, O2=0.19, N2=0.715}
massf_inf = gmodel:molef2massf(molef_inf)
inflow = FlowState:new{p=p_inf, T=T_inf, velx=u_inf, massf=massf_inf}
initial = FlowState:new{p=p_inf/5, T=T_inf, velx=0, massf=massf_inf}

-- Now set some configuration options (no shock fitting (yet))
body_flow_time = R/u_inf
ni = 64; nj = 64
config.axisymmetric = false -- from true --> false (2D)
config.reacting = true
config.reactions_file = 'drm19-reac-file.lua' --or 'methane-air-reaction-mechanism-y
```

```
config.reaction_time_delay = 4 * body_flow_time --from 4 --> 1
config.flux_calculator = 'adaptive'
config.gasdynamic_update_scheme = "predictor-corrector"
-- config.grid_motion = "shock_fitting"
-- config.shock_fitting_delay = 2 * body_flow_time
config.max_time = 20 * body_flow_time  -- allow time to establish
config.max_step = 800000
config.dt_init = 1.0e-10
config.cfl_value = 0.4
config.dt_plot = config.max_time/50.0

-- Set up the geometry for defining the grid
a = Vector3:new{x=0.0,   y=0.0}
b = Vector3:new{x=-R,    y=0.0}
c = Vector3:new{x=0.0,   y=R}
d = { Vector3:new{x=-2*R,   y=0.0},
      Vector3:new{x=-2*R,   y=R},
      Vector3:new{x=-R,       y=2.5*R},
      Vector3:new{x=0.0,      y=4*R} }
-- Set up surface and grid
psurf = makePatch{north=Line:new{p0=d[#d], p1=c},
                  east=Arc:new{p0=b, p1=c, centre=a},
                  south=Line:new{p0=d[1], p1=b},
                  west=Bezier:new{points=d}}
grid = StructuredGrid:new{psurface=psurf, niv=ni+1, njv=nj+1}
-- Set up block as SBlockArray
blk = SBlockArray{grid=grid, fillCondition=inflow, label='blk',
                  bcList={west=InFlowBC_Supersonic:new{flowCondition=inflow},
                          north=OutFlowBC_Simple:new{}},
                  nib=2, njb=2}

-- Add some history points.
setHistoryPoint{x=b.x, y=b.y}
setHistoryPoint{x=c.x, y=c.y}
config.dt_history = 1.0e-8
```