Rowan J. Gollan and Peter A. Jacobs

Eilmer4.0 流动仿真程序

----------热化学气体反应指南

（使用热完全气体模型）

2020 年 9 月 14 日更新

译者：卢顺、齐建荟

Technical Report 2018/04

School of Mechanical & Mining Engineering

The University of Queensland

摘要：

Eilmer 是一个模拟可压缩气体流动的程序。它是基于有限体积公式，再加上气体热化学模型的质量、动量、能量和物种守恒方程。本指南包含了气体单温度反应模型的描述，其中包括热完全气体物种的混合物和确定混合物组成的有限速率动力学方案。这个反应气体模块可以用作流动仿真的一部分，也可以用于您自己设计的脚本化计算。Lua、Python3 和 Ruby 的包装器可用于脚本计算，本指南中提供了算例。最后一个例子展示了反应气体模型在流动仿真中的应用。

本用户指南的英文原版以及几何、气体等配套文件可在以下网站下载：https://gdtk.uqcloud.net/docs/eilmer/user-guide/。

目录

1. 介绍

当您使用 Eilmer 流动求解器时，您应该从最简单的、最合适的气体模型开始分析。对于中等温度为几百度的流动，这就是理想的气体模型，它具有固定的组分和热容。但是，在更高的温度下，气流中的气体可能会发生化学反应。例如，在空气中，到 2000K 时氧分子就开始发生分解，到 4000K 时氮分子将开始分解。

进入大气层的航天器、管道中的燃烧和超声速流动是典型的高超声速流动例子，这些情况都可能用 Eilmer 分析。在这些情况下，温度可能在 1000K 到几千度之间，速度可能是每秒几公里。因此，气体化学成分的变化可能会非常显著，而且这些变化的时间尺度可能与气体通过流域的时间相当。现在，为了用 Eilmer 进行气体动力学分析，需要用一组热化学关系来补充流动求解器的方程，这些热化学关系描述了气体作为一种化学物种的混合物，经历有限速率化学反应的行为。

在本指南中，我们首先考虑具有一个或多个化学组分的热完全气体模型(第 2 节)，所有这些模型都具有完全的碰撞行为，但每个模型都具有激发到由单一温度描述的所有平衡态内能模式。这已成为 Eilmer 进行高超声速流动分析的工作气体模型。然后，我们描述了可用于此气体模型的通用有限速率化学反应方案(第 3 节)，接着描述了配置文件规范(第 4 节)。最后一节(第 5 节) 采用脚本化计算的形式，描述了一些类似的应用程序，并将其作为 Eilmer 流动求解器的气体仿真的一部分。

从公共库 https://bitbucket.org/cfcfd/dgd-git/中可以获得 Eilmer 源代码的气体模块及其它代码，Eilmer 的主页是 http://cfcfd.mechmining.uq.edu.au/eilmer/

本用户指南的英文原版以及几何、气体等配套文件可在以下网站下载：https://gdtk.uqcloud.net/docs/eilmer/user-guide/。

2. 热完全气体模型

本节给出了给定组成的气体混合物的热力学关系。但有限速率化学效应的实现、定义的组分如何随时间变化等问题，将在第 3 节中讨论。

2.1 单一种类的热完全气体

理想热完全气体的假设意味着在单一温度下，所有的内能模式都处于平衡状态。对于原子来说，这意味着平动能和电子能的玻尔兹曼分布是由一个温度值决定的。同样地，对于分子，玻尔兹曼分布的平动、转动、振动和电子能是由一个单一的温度值描述。

要模拟热完全气体，需要知道气体如何以温度的函数储存能量。可以很方便地得到定压比热 $C_p(T)$作为温度的函数，由此可计算出气体的比焓为

$$h = \int_{T_{ref}}^{T} C_p(T)dT + h(T_{ref}) \tag{2.1}$$

则熵是：

$$s = \int_{T_{ref}}^{T} \frac{C_p(T)}{T} dT + s(T_{ref}) \tag{2.2}$$

输运特性、粘度和热导率，可以作为一个温度函数的单一组成气体混合物来计算。单个组分的输运特性可以通过适当的混合规则组合起来，从而得到混合物的粘度和导热系数。

在实现过程中，热完全气体的特征是五条曲线都符合温度的函数:

1. 定压比热容$C_p(T)$；
2. 焓 $h(T)$；
3. 熵 $s(T)$；
4. 粘度μ(T)；
5. 热导率k(T)；

这些曲线都符合 McBride 和 Gordon[1]所使用的形式。无量纲形式的热力学

性质拟合曲线如下：

$$\frac{C_p(T)}{R} = a_0T^{-2} + a_1T^{-1} + a_2 + a_3T + a_4T^2 + a_5T^3 + a_6T^4 \quad (2.3)$$

$$\frac{H(T)}{RT} = -a_0T^{-2} + a_1T^{-1}logT + a_2 + a_3\frac{T}{2} + a_4\frac{T^2}{3} + a_5\frac{T^3}{4} + a_6\frac{T^4}{5} + \frac{a_7}{T} \quad (2.4)$$

$$\frac{S(T)}{R} = -a_0\frac{T^{-2}}{2} - a_1T^{-1} + a_2logT + a_3T + a_4\frac{T^2}{2} + a_5\frac{T^3}{3} + a_6\frac{T^4}{4} + a_8 \quad (2.5)$$

这些曲线拟合的系数可用于 CEA 程序[1](和相关的数据库文件)中的大量气体种类。这些曲线的每种拟合式只在一定温度范围内有效。例如，氮分子(N_2)的热力学曲线由 200.0-1000.0K、1000.0-6000.0K 和 600.0-20000.0K 三个部分组成。在这些多项式的边界上，这些值并不完全匹配。这种不匹配使得用于给定焓的温度迭代方法的实施过程很麻烦，因为这些方法依赖于底层函数的光滑变化。为了克服这个问题，我们使用了 Gupta 等人[2]所提出的思想，在边界附近混合多项式系数。在 1000K 断点处，我们对 400K 范围内的系数进行线性混合。线性混合从 800K 开始到 1200K 结束。在 6000K 断点处，混合范围为 1000K。需要注意的是，这些系数是混合的，然后在多项式中用来评估所需的性质。混合不会对值本身执行。在多项式集合的范围之外，值是外推的。外推是基于一个粗略的假设，常数 C_p 在范围之外。因此推断如下：

$$\frac{C_p(T < T_{low})}{R} = \frac{C_p(T_{low})}{R}$$

$$\frac{C_p(T > T_{high})}{R} = \frac{C_p(T_{high})}{R}$$

$$\frac{H(T < T_{low})}{RT} = \frac{1}{T}\{H(T_{low})T_{low} - C_p(T_{low})(T_{low} - T)\}$$

$$\frac{H(T > T_{high})}{RT} = \frac{1}{T}\{H(T_{high})T_{high} + C_p(T_{high})(T - T_{high})\}$$

$$\frac{S(T < T_{low})}{R} = S(T_{low}) - C_p(T_{low})log\left(\frac{T_{low}}{T}\right)$$

$$\frac{S(T > T_{high})}{R} = S(T_{high}) + C_p(T_{high})log\left(\frac{T}{T_{high}}\right)$$

该曲线适用于粘度和导热系数，其形式与 CEA 程序[1]相同。曲线如下:

$$log\mu(T) = a_0logT + \frac{a_1}{T} + \frac{a_2}{T^2} + a_3$$

$$logk(T) = b_0logT + \frac{b_1}{T} + \frac{b_2}{T^2} + b_3$$

2.2 热完全气体集合的混合规则

热完全气体混合物的热力学状态是由两个状态变量和混合物组成所唯一确定的。热力学能[1]是由各个热力学能的质量分数加权计算出来的：

$$e = \sum_{i=1}^{N} f_i \, e_i = \sum_{i=1}^{N} f_i \, (h_i - R_i T) \tag{2.6}$$

压强是根据道尔顿分压定律计算:

$$p = \sum_{i=1}^{N} \rho_i \, R_i T \tag{2.7}$$

混合物的气体常数定义为:

$$R = \sum_{i=1}^{N} f_i \, R_i \tag{2.8}$$

C_p 的计算是基于组分比热的质量分数加权：

$$C_p = \sum_{i=1}^{N} f_i \, C_{pi} \tag{2.9}$$

定容比热计算为:

$$C_v = C_p - R \tag{2.10}$$

比热比 γ 由定义式求得：

$$\gamma = \frac{C_p}{C_v} \tag{2.11}$$

混合物的冻结声速为：

$$a = \sqrt{\gamma R T} \tag{2.12}$$

在可压缩流动模拟中，ρ和 e 的值最容易从每次时间增量求解的守恒量中得到。这就导致了解决给定的气体混合物的热力学状态ρ、e 和混合物组分$\vec{f}$中存在一些问题。然而，前面提出的公式的前提是温度已知。所以我们需要用牛顿迭代法求解温度：

$$T_{n+1} = T_n - \frac{f_0(T_n)}{f_0{}'(T_n)} \tag{2.13}$$

其中，零函数$f_0(T)$是基于给定的热力学能 e，以及基于温度的热力学能假设：

[1] 注意，GasState 类使用符号 *u* 表示特定的热力学能。

$$f_0(T) = e_{guess} - e = \sum_{i=1}^{N} f_i \left(h_i - R_i T_{guess}\right) - e \tag{2.14}$$

利用$C_{vi} = \frac{de_i}{dt}$这一事实，我们可以很方便地通过计算混合C_v来找到牛顿法的导数函数:

$$\frac{\mathrm{d}f_0(T)}{\mathrm{dT}} = \sum_{i=1}^{N} f_i \frac{de_i}{dT} = \sum_{i=1}^{N} f_i C_{vi} = C_v \tag{2.15}$$

设定当温度值精度在$\pm 1.0 \times 10^6$K 以内时，牛顿迭代可以收敛。个人经验表明，当在有限速率化学计算中使用温度来计算成分变化率时，这种误差范围是必需的。

混合物输运性质的计算不像热力学性质那么直接。为了计算混合物的粘度和导热系数，需要一个混合规则。Wilke 的混合规则[3]已经在本文的工作中实现。具体来说，本文使用了 Gordon 和 McBride[4]在 CEA 程序中使用的混合规则来计算混合输运特性;这些规则是 Wilke 原始公式[3]的变体。

$$\mu_{mix} = \sum_{i=1}^{N} \frac{x_i \mu_i}{x_i + \sum_{j=1, j\neq i}^{N} x_j \phi_{ij}} \tag{2.16}$$

以及：

$$k_{mix} = \sum_{i=1}^{N} \frac{x_i k_i}{x_i + \sum_{j=1, j\neq i}^{N} x_j \psi_{ij}} \tag{2.17}$$

其中x_i是种类 i 的摩尔分数。

相互作用势ϕ_{ij}和ψ_{ij}可以用多种方法计算。同样，我们使用了 Gordon 和 McBride 提出的公式[4]进行计算：

$$\phi_{ij} = \frac{1}{4}\left[1 + \left(\frac{\mu_i}{\mu_j}\right)^{0.5} \left(\frac{M_j}{M_i}\right)^{0.25}\right]^2 \left(\frac{2M_j}{M_i + M_j}\right)^{0.5} \tag{2.18}$$

以及：

$$\psi_{ij} = \phi_{ij}\left[1 + \frac{2.41\left(M_i - M_j\right)\left(M_i - 0.142M_j\right)}{\left(M_i + M_j\right)^2}\right] \tag{2.19}$$

其中M_i和M_j分别对应于 i 和 j 种类的分子量。

一旦计算出混合物粘度和导热系数，就可以从混合物普朗特数的定义计算出

混合物普朗特数为：

$$Pr = \frac{\mu C_p}{k} \tag{2.20}$$

3. 有限速率的化学反应

3.1 由于化学反应而引起物质变化的速率

通过假设一系列简单的可逆反应，化学反应体系可以表示为:

$$\sum_{i=1}^{N} \alpha_i X_i \rightleftharpoons \sum_{i=1}^{N} \beta_i X_i \tag{3.1}$$

其中α_i和β_i分别为反应物和生成物的化学计量系数。在不可逆反应的情况下，可以把逆向反应速率设为零。对于给定的反应 j，物质 i 的浓度变化率为:

$$\left(\frac{\mathrm{d}[X_i]}{\mathrm{dt}}\right)_j = v_i \left\{ k_f \prod_i [X_i]^{\alpha_i} - k_b \prod_i [X_i]^{\beta_i} \right\} \tag{3.2}$$

其中$v_i = \beta_i - \alpha_i$。通过对所有反应的求和，总浓度变化率是：

$$\frac{\mathrm{d}[X_i]}{\mathrm{dt}} = \sum_{i=1}^{N_r} \left(\frac{\mathrm{d}[X_i]}{\mathrm{dt}}\right)_j \tag{3.3}$$

对于某些集成方案，将生成和反应率作为单独的数量来计算是很方便的。在这种情况下:

$$\frac{\mathrm{d}[X_i]}{\mathrm{dt}} = q_i - L_i = \sum_{j=1}^{N_r} \dot{\omega}_{app_{i,j}} - \sum_{j=1}^{N_r} \dot{\omega}_{va_{i,j}} \tag{3.4}$$

$\dot{\omega}_{app_{i,j}}$和$\dot{\omega}_{va_{i,j}}$的计算取决于达到 j 反应时的v_i值，如表 3.1 所示。

表 3.1：基于v_i的化学生成和损失量

	$v_i > 0$	$v_i < 0$
$\dot{\omega}_{app_{i,j}}$	$v_i k_f \prod_i [X_i]^{\alpha_i}$	$-v_i k_b \prod_i [X_i]^{\beta_i}$
$\dot{\omega}_{va_{i,j}}$	$-v_i k_b \prod_i [X_i]^{\beta_i}$	$v_i k_f \prod_i [X_i]^{\alpha_i}$

反应速率系数 k_f 和 k_b 的计算以及物质浓度变化常微分方程组的求解方法将在后面的部分进行讨论。

3.2 反应速率系数

反应速率系数可以通过实验(通常使用激波管研究)或从理论中确定。在许多情况下，根据理论对反应速率的估计与实验确定的值相差几个数量级。因此，与实验值的拟合是最常用的。

对于本章讨论的单温度气体模型，正反应速率系数采用广义 Arrhenius 阿伦尼乌斯形式计算：

$$k_f = AT^n exp\left(\frac{-E_a}{kT}\right) \tag{3.5}$$

k 是玻尔兹曼常数,A、n 和E_a是模型的常数。

逆向速率系数也可以用修正的 Arrhenius 阿伦尼乌斯形式计算：

$$k_b = AT^n exp\left(\frac{-E_a}{kT}\right) \tag{3.6}$$

或者可以先计算反应的平衡常数：

$$k_b = \frac{k_f}{K_c} \tag{3.7}$$

如果由平衡常数计算逆反应速率系数，则需要平衡常数的计算方法。某一特定反应的平衡常数可以从曲线拟合中计算出来，或者如本工作所做的那样，使用热力学原理进行计算。以浓度为基础的平衡常数与以压力为基础的平衡常数的关系式为：

$$K_c = K_p \left(\frac{p_{atm}}{\mathcal{R}T}\right)^v \tag{3.8}$$

其中p_{atm}是以 Pa 为单位的大气压，$\mathcal{R}$是通用气体常数，$v = \sum_i^{N_S} v_i$,及

$$K_p = exp\left(\frac{-\Delta G}{\mathcal{R}T}\right) \tag{3.9}$$

在任何涉及化学平衡的经典热力学介绍性文本中,都可以找到以分压为基础的平衡常数 K_p 公式的推导。反应的微分吉布斯函数ΔG 可由下式求得：

$$\Delta G = \sum_i^{N_S} v_i G_i \tag{3.10}$$

其中每个 G_i 都是根据吉布斯自由能的定义计算出来的:

$$G_i(T) = H_i(T) - T \times S_i(T) \tag{3.11}$$

G_i 的单位是 J/mol。H_i 和 S_i 可以用 CEA 多项式在规定的单位内计算，并分别乘以$\mathcal{R}$T 和$\mathcal{R}$。

对于特定的流动问题，在选择和使用反应速率时应谨慎操作。在许多情况下，

一组反应速率可能只针对特定的问题域进行调整。Oran 和 Boris 描述了反应速率调整集的问题及其产生原因(参考文献[5]第 38 页)，其内容如下：

> 在化学反应中经常出现的一个问题是，测量的反应速率基本不一致。例如，可能有正向和反向速率常数 k_f 和 k_r 的实验测量。然而，当其中一种与该反应的平衡系数结合时，另一种不会产生。这似乎代表了对平衡热力学的违背。解释通常是 k_f 和 k_r 是在相对不同的温度或压力下测量的，因此当它们被外推到实验的有效性范围之外时，就会产生不一致。

3.3 化学动力学常微分方程的求解

式 3.3 所表示的系统是一个常微分方程组（ODEs），可以用合适的方法求解。对于某些化学系统，由于某些物种的数量级发生较大变化，使得 ODEs 方程的求解产生较大的刚性问题。对于这些系统，需要有针对刚性 ODEs 的特定方法。目前在实现中提供了两种 ODE 方法:一种是针对非刚性系统的，另一种是针对刚性系统的。

1. Runge-Kutta-Fehlberg 方法（非刚性系统有效）
2. alpha-QSS 方法（专门针对刚性化学反应系统）

四阶 Runge-Kutta 方法使用五阶误差估计作为控制时间步长的方法，由 Fehlberg[6]提出。这对于非刚性系统非常有效。

alpha-QSS method：

在 Mott 的论文[7]中提出了准稳态 alpha-QSS（准稳态）方法。该论文对此方法的发展提供了很好的细节，然而，后来关于该方法的期刊文章[8]对更新方程提供了排版更正。我们的实施过程使用了这篇期刊文章中的方程。它是一个 ODE 求解器，专门针对化学系统的刚性问题。这个 ODE 求解器利用了式 3.4 计算的正向和逆向浓度变化率。这是一种预估校正型结构，在这种结构中，校正器被迭代，直到达到期望的收敛。预估步和校正步分别是：

$$[X_i]^1 = [X_i]^0 + \frac{\Delta t q_i^0}{1 + \alpha_i^0 \Delta t L_i^0} \tag{3.12}$$

$$[X_i]^{n+1} = [X_i]^0 + \frac{\Delta t(\bar{q}_i - [X_i]^0 \bar{L}_i)}{1 + \bar{\alpha}_i \Delta t \bar{L}_i} \tag{3.13}$$

在上述方程中

$$\bar{L}_i = \frac{1}{2}(L_i^0 + L_i^n) \tag{3.14}$$

$$\bar{q}_i = \bar{\alpha}_i q_i^n + (1 - \bar{\alpha}_i) Q_i^0 \tag{3.15}$$

该方法的关键是正确计算α。这个α参数控制着如何在给定的物种整合上更新工作。注意，α被定义为：

$$\alpha(L\Delta t) \equiv \frac{1 - (1 - e^{-L\Delta t})/(L\Delta t)}{1 - e^{-L\Delta t}} \tag{3.16}$$

使用 Pade 近似：

$$e^x \approx \frac{360 + 120x + 12x^2}{360 - 240x + 72x^2 - 12x^3 + x^4} \tag{3.17}$$

由于避免了耗时的指数函数计算，所以可以写出更易于计算α的表达式。α的近似为：

$$\alpha(L\Delta t) \approx \frac{180r^3 + 60r^2 + 11r + 1}{360r^3 + 60r^2 + 12r + 1} \tag{3.18}$$

其中$\mathrm{r} \equiv 1/(L\Delta t)$

对校正步进行迭代，直到达到收敛或已到达最大步数。对于所有满足条件的物种，Mott[7]给出如下收敛准则:

$$\|[X]^c - [X]^P\| < \varepsilon_1 [X]^c \tag{3.19}$$

参考前面使用的符号，$[X]^c = [X]^{n+1}$和$[X]^P = [X]^1$。Qureshi 和 Prosser[9]建议在 3.19 式中加入 δ 项，以降低需要满足物种首次从零浓度出现的标准。这种标准的改变也有后续效果，可以防止一些不必要的小时间步骤。Qureshi 和 Prosser 建议：

$$\|[X]^c - [X]^P\| < \varepsilon_1([X]^c + \delta) \tag{3.20}$$

在完成一个步长时，无论是成功的还是不成功的，Mott[7]都会为新的步长大小提供建议。该算法，包括 Qureshi 和 Prosser[9]的 δ 加法，为：

$$\sigma = \max\left(\frac{\|[X]^c - [X]^P\|}{\varepsilon_2([X]^c + \delta)}\right) \tag{3.21}$$

$$(\Delta t)_{new} = (\Delta t)_{old}\left(\frac{1}{\sqrt{\sigma}^*} + 0.005\right) \tag{3.22}$$

注意，在σ作为起始值的牛顿法中，$\sqrt{\sigma}^*$通常在第三步中计算。这是 Mott[7]为实

施效率提高的建议措施。

3.4 流动求解器耦合化学效应

本文还详细介绍了化学反应对气体动力学仿真的耦合作用。在非定常、时间精确的流动模拟中，允许时间步长受到 Courant-Friedrichs-Lewy (CFL)准则的约束。在粘性可压缩流动中，CFL 准则允许选择适当的时间步长，并将流动信息的传播限制在小于一个单元宽度的距离内。流动信息传播的速度是无粘波速度和粘性效应的函数。

当有限速率化学反应的影响从流动仿真中分离出来时，化学更新在一个单独的步骤中被解决，在这个步骤中流动被冻结。(事实上，在时间步长分裂的过程中，物理和化学过程都在不断更新。)因此，化学问题是在流动时间步的末尾找到更新的物种组成。

它可能是，而且极有可能是，流动时间步并不是一个合适的时间步来解决化学动力学 ODE 问题。当化学问题的时间步长小于流动时间步长时，化学问题被分若干次循环，直到总时间等于流动时间步长。通常有化学时间步比流动时间步小 100-1000 倍的模拟，也就是说，解决化学问题需要 100-1000 次循环。当化学反应的时间步长大于流动时间步长时，只需将其设置为流动时间步长的值。

在仿真过程中，我们对每个有限体积单元的化学时间步长进行了跟踪。虽然流体在随后的时间步中继续流动，但如果流动条件的变化不大，那么之前的化学时间步将是一个很好的估计，以便在随后的时间步中开始新的化学问题。一个例外情况是当激波通过单元格时:流动条件的变化确实变得很大。在这种情况下，旧的化学步骤被忽略，新的步骤被选择。下一段将讨论新步骤的选择程序。当使用 Runge-Kutta-Fehlberg 或 alpha-QSS 方法时，新化学时间步的估计将作为 ODE 更新例子的一部分提供。

因此，在模拟过程中，可以在一次迭代中使用旧的化学步骤来开始下一次迭代中的新化学问题。我们需要的是一种方法，在最初的迭代中选择化学步骤，或者在旧的建议不合理的时候(如在单元出现激波的情况下)。在本工作中，化学问题的初始步骤是根据 Young 和 Boris 的建议[10]：

$$\mathrm{dt}_{chem} = \epsilon_1 min\left(\frac{[X_i](0)}{[\dot{X}_i](0)}\right) \tag{3.23}$$

其中本文取ϵ_1为 1.0×10^{-3}，表达式在化学子问题的初值处求值。Young 和 Boris[10] 建议ϵ_1根据收敛准则进行缩放。我们发现，对于我们研究课题组感兴趣的问题，固定的值就足够了。

4. 反应方案文件

在 Lua 输入文件中描述了流动仿真中可能发生的化学反应。该输入文件由用户准备，使用 prep-chem 程序进行预处理，以生成详细的化学文件，供随后在主仿真代码或自定义脚本中使用。由于输入文件是基于 Lua 的，所以用户在准备文件时可以访问 Lua 脚本语言的全部内容。如果您不知道 Lua 语法，不要担心;这里给出的说明和例子应该足够让您开始建立反应方案文件[2]。

让我们继续看一个输入文件示例，并讨论关键字和语法。这里列出的是一个输入文件，它描述了氮的简单热分解。只有两种参与的物质，N_2 和 N，也只有两种反应。

```
Reaction{
    ’ N2 + N2 <=> N + N + N2’,
    fr={’Arrhenius’, A=7.0e21, n=-1.6, C=113200.0},
    br={’Arrhenius’, A=1.09e16, n=-0.5, C=0.0}
}
Reaction{
    ’ N2 + N <=> N + N + N’,
    fr={’Arrhenius’, A=3.0e22, n=-1.6, C=113200.0},
    br={’Arrhenius’, A=2.32e21, n=-1.5, C=0.0}
}
```

第一个反应是 N_2 与其它 N_2 分子碰撞而分解。采用广义 Arrhenius 模型计算了正反应速率系数，并给出了该模型的参数。同样，逆反应速率系数也用 Arrhenius 公式计算。

更一般地说，每个反应都在“Reaction”表[3]中指定。表由左括号和右括号({})分隔。表中的第一个条目总是一个字符串。这个字符串就是反应的化学方程。表中的其余项由关键字-值(形式为 key=val)表示，可以以任何顺序出现。表格中的每一项都用逗号[4]分隔。这个例子文件包含两个 Reaction 表，因此在反应方案中处理了两个反应。

在深入讨论输入文件之前，还有一些最后的注意事项。在反应文件中没有明

[2] 如果您担心刚开始就需要学习 Lua，那就多虑了。首先，您可能只是将其视为化学的输入格式，而忘记了它与 Lua 有任何关系。其次，Lua 在设计时考虑到了非程序员，因此它使用了简单的语法，特别是为了让那些非程序员能够快速地将 Lua 用作配置语言。

[3] 如果您精通 Lua，就会意识到 Reaction 是一个调用函数，只有一个参数，一个表

[4] Lua 还允许使用分号而不是逗号来分隔表项。

确提到参与的物种。参与的组分取自气体模型文件中用于相同流动仿真的组分。换句话说，如果您在反应体系中列出气体模型中不存在的物质，那么就会得到一个错误信息。

4.1 输入文件格式概述

在我看来，通过利用 Lua 作为输入数据描述语言，输入文件几乎是自我描述的。这提供了执行仿真时使用建模的良好数据记录。一个有效的反应输入文件将遵循以下规则。

1. 任何合法的 Lua 代码都是可以接受的，但是您不能重命名以下预定义函数:
   - Reaction
   - removeAllReactionsWithLabel
   - removeReaction
   - selectOnlyReactionsWithLabel
   - selectOnlyReactions
2. 在 Reaction 表中声明了反应
3. 当遇到两个破折号(--)时，代表着注释开始，并继续到行尾。(重复第 1 项，注释是合法的 Lua 代码。)

当反应列在文件中时，它们从 1 开始在内部编号。在某些情况下，将所有反应列在一个图表中，然后只使用其中的一些反应是很方便的。这是很常见的，比如您想用还原机制，或者如果您相信一种物质在感兴趣的流动条件下是惰性的，所以想要去除所有涉及到该物质转化的反应。同时提供了两个方便的函数，这样您就不必进入输入文件来删除不想要的反应：

- removeAllReactionsWithLabel
- removeReaction

这两个函数都接受单个项或其数组。数组是 Lua 表的一种特殊形式，用大括号括起来({})。第一个函数接受与反应标签对应的字符串。下一节将解释反应的标记。第二个函数接受与内部编号对应的整数。便利函数必须在声明相关反应之后调用。

通常，用户会将对这些函数的调用放在输入文件的末尾。以下是两个例子：

```
removeAllReactionsWithLabel({'r3', 'r5'})
```

这个调用将从参与的反应列表中删除标记为' r3 '和' r5 '的反应。

```
removeReaction(13)
```

在这这个调用中，第 13 个反应从列表中被删除了(因为我们都知道 13 是不吉利的，对吧?)[5]

类似地，有两个互补的方便函数，允许从全部反应中只选择某些反应:

- selectOnlyReactionsWithLabel
- selectOnlyReactions

它们的工作原理与 remove 移除函数相反:这些函数只会选择那些在它们的函数中列出的化学反应。

注意，不建议在一个反应脚本中混合使用 remove 移除和 select 选择函数。因为这种命令未经测试。现在来看看“反应”表的细节。

4.2 Reaction 表的细节

Reaction 表接受了一些项目，有些是强制性的，但大多数不是。这里显示了项目的完整列表，下面对每个项目进行了描述。

```
Reaction{
    ’equation string’,
    fr={...},
    br={...},
    ec=’model name’,
    efficiencies={...},
    label=’r1’
}
```

’**equation string**’ (强制性的)：

如前所述，此字符串必须首先出现在表中，并且没有与之关联的关键词。这个字符串表示反应方程。例如，氮的离解可以写成：

[5] 事实上，不像美国人和他们的建筑，您不可能那么容易摆脱。如果您有超过 13 个反应，编号高的反应会打乱一个位置，这样编号就保持从 1 开始的连续。这一切都发生在内部。

’N2 + N2 <=> N + N + N2’

如果反应涉及到与一般的第三物体的碰撞，那么这就被严格地称为 M 种物质。例如，由氧源和单原子氢形成过氧化氢需要有第三种物体的存在。这个反应写成：

’H + O2 + M <=> HO2 + M’.

反应物和生成物用方向箭头标明。<=>表示反应在两个方向进行，而=>表示反应只在正向进行(不计算反向的转化率)。

fr (可选的，如果提供了 br)：

fr 关键字用于指定正向反应速率系数，并期望表的值。表的格式是一个命名模型的字符串，后面跟着提供模型参数的关键字-值。当前实现的计算反应速率系数的模型以及它们的输入格式，都可以在这一节的末尾查看。

br (可选的，如果提供了 fr)：

br 关键字用于指定逆向反应速率系数。使用方式与正向反应速率(fr)相同。

ec (可选的)：

ec 关键字用于指定计算平衡常数的模型。它接受一个命名模型的字符串。目前，根据热力学原理计算平衡常数的热力学模型只有一个’from thermo’。对于可逆反应，如果只指定了 fr 或 br 中的一个，则假定使用平衡常数，无需声明。

efficiencies (可选的)：

如果存在第三体反应，则假定混合物中所有物种的反应效率为 1.0。efficiencies 关键字接受值为 1.0 假设的列表。列表包含 species=efficiency_value。例如，表示 N_2 的效率值为 6 倍，O_2 的效率值为 3.5 倍，列表将是：

efficiencies={N2=6.0, O2=3.5}

请注意，除非列表中另有说明，否则所有物种的值都假定为 1.0。如果有一个物种不是作为第三体参与的，那么需要确保将其效率值设置为 0.0(例如 H=0.0)。另外，请注意如何使用 Lua 表构造函数来帮助处理离子和电子:在将它们放入效率表时，只需将它们用括号括起来。下面是一个例子：

efficiencies={[’O2+’]=9.0, [’e-’]=0.0}

label (可选的)

标签接受一个字符串，允许用户给反应命名。如果希望以后根据标签移除某些反应，可以使用 remove_reactions_by_label 函数。

注意，如果您指定了 fr, br 和 ec 这三个，您就过度指定了反应速率系数的模型。在这种情况下，没有给出错误。相反，ec 模型被忽略了。

反应系数模型的输入

不同的速率系数模型分别为正向速率系数、反向速率系数或两者分别指定。规范的形式是：fr={'ModelName', param1=..., param2=...}

可用的反应率系数模型有：

- 广义的 Arrhenius 公式
- 依赖压力的两种形式：
  1. Lindemann-Hinshelword 形式
  2. Troe 形式

下面我们将描述可用的模型、如何选择模型，以及它们的输入参数。

广义的 Arrhenius 公式：

广义 Arrhenius 速率系数计算为：

$$k = \mathrm{A}T^{n}exp(-C/T)$$

在输入文件中应该设置为：

fr={'Arrhenius', A=..., n=..., C=...}，其中

A：是以 cgs 单位给出的指前系数(因为它们在化学反应速率文献中最常见)。

n：T 的无因次幂。

C：活化温度，单位为开尔文。

注意，在每个方向，对一个特定的反应，指前系数的单位将取决于碰撞次数的大小。考虑氮分解反应：

' N2 + N2 <=> N + N + N2'

我们可以特殊处理速率方程(式 3.2)，来给出由于这个反应而产生氮分子的速率为：

$$\frac{\mathrm{d}[N_2]}{\mathrm{dt}} = -1\{k_f[N_2]^2 - k_b[N]^2[N_2]\}$$

浓度的单位是 mole.cm^{-3}, k_f 及其相关的指前系数 A 的单位是 mole^{-1}.cm^3.s^{-1}，而逆

反应速率的单位是 $mole^{-2}.cm^{6}.s^{-1}$

同样，不要被公式的指数部分出现的负号所迷惑;您需要输入活化温度，它在大多数情况下是一个正值。有时，活化温度是负的。这将在反应方案中给出。

Lindemann-Hinshelwood 形式，压力依赖率系数

Lindemann-Hinshelwood 压力依赖率系数是零压力和无限压力极限的混合:

$$k = \frac{k_\infty k_0[M]}{k_\infty + k_0[M]}$$

在 O'flaherty 论文[11]的第三章中，对 Lindemann-Hinshelwood 压力依赖率理论进行了讨论。要将此设置为输入文件中的速率系数，请使用:

```
fr={'pressure dependent',
    kInf={A=..., n=..., C=...},
    k0={A=..., n=..., C=...}}
```

其中：

kInf：用于设定压力无限大极限下的 Arrhenius 参数。

k0：用于设定零压力极限下的 Arrhenius 参数。

Troe 形式，压力依赖率系数

速率系数的 Troe 形式也是一个压力依赖率速率。它被定义为：

$$k = \frac{k_\infty k_0[M]}{k_\infty + k_0[M]} F$$

$$\text{logF} = \left[1 + \left(\frac{logP_r + c}{n - d(logP_r + c)}\right)^2\right]^{-1} logF_{cent}$$

其中

$$P_r = \frac{k_0[M]}{k_\infty}$$

$$c = -0.4 - 0.67 logF_{cent}$$

$$n = 0.75 - 1.27 logF_{cent}$$

$$d = 0.14$$

$$F_{cent} = (1-a)exp\left(\frac{-T}{T^{***}}\right) + aexp\left(\frac{-T}{T^{*}}\right) + exp\left(\frac{-T^{**}}{T}\right)$$

要将此设置为输入文件中的速率系数，使用:

```
fr={'pressure dependent',
    kInf={A=..., n=..., C=...},
    k0={A=..., n=..., C=...},
    Troe={a=..., T1=..., T2=..., T3=...}}
```

其中，

kInf：用于设定压力无限大极限下的 Arrhenius 参数。

k0：用于设定零压力极限下的 Arrhenius 参数。

Troe：用于设置出现在上面表达式中的参数，T1=T^*, T2=T^{**}, T3=T^{***}

注意:Troe 和 Lindemann-Hinshelwood 表单都选择了'pressure dependent'名称。Troe 参数集的存在与否用于区分这两种形式。

4.3 化学方案的其它控制

这里有许多解决有限率化学问题的细节，这是默认为用户设置的。但是，用户可以通过在输入文件中设置值来控制所有这些参数。用户可以在 Config 表中设置这些值。下面将描述可控选项的完整列表。首先，在速率系数评估中设置一些温度限制，并选择 Runge-Kutta-Fehlberg ODE 积分器,示例如下：

```
Config{
    tempLimits={lower=300.0, upper=10000.0},
    odeStep={method="rkf"}
}
```

tightTempCoupling：

默认情况下，不采用温度更新和物种更新之间的紧密耦合。这意味着，在正常操作中，速率常数只计算一次(常数，而不是速率)，并且只在化学更新开始时计算一次。这是一种在有限速率化学模块中非常常用的有效测量方法，因为速率常数计算相对繁琐。由于给定计算单元的温度在感兴趣的时间步长内变化不大，所以在大多数情况下，它取为近似值也能很好地工作。

然而，在某些情况下，当化学组分在选定的时间步长内变化非常快时，相应的温度梯度很大。这种情况下，在化学更新中重新评估每个子循环步骤的温度和速率常数会更加可靠和准确。要将该选项设置为激活，请将该参数设置为 true： tightTempCoupling = true

总之，默认情况下，在化学更新过程中，将温度视为常量。这在大多数情况下是很好的，并且是进行计算非常有效的方法。对于能量非常大的化学反应，选择紧密的温度耦合会更加可靠。这将更加精确，但代价是计算成本更高。

tempLimits：

这个设定用来控制反应速率系数的温度极限。用户提供温度限制的下限和上限。在这里的示例中，温度下限设置为 300K，上限设置为 50000K。如果用户没有设置这些值，那么就默认使用这些值。

tempLimits={lower=300.0, upper=50000.0}

这些值用于控制反应速率系数的温度极限。当局部温度超过限制(上限或下限)，速率简单地评估相应温度超过的极限。该代码为：

```
if T > T_upper
   then T = T_upper
if T < T_lower
   then T = T_lower
eval_rate_coeff(T)
```

maxSubcycles：

当与流动求解器耦合时，该设置控制最大允许的化学子循环。这是一个整数值，默认值是 10000。这种设置是为了确保化学更新保持合理，并且不会因为非常小的化学步骤而导致几乎没有进展。如果确实发生了这种情况，这通常是一个信号，表明存在更大的问题，即数值模型的设置，或对数值模型的要求。

maxSubcycles=10000

maxAttempts：

少数情况下，化学步骤会失败。这可能是因为步长增加了一点，超出了数字稳定的范围。这称为失败的尝试，代码将调整步长大小，并重新尝试该步骤。对于恢复失败步骤的尝试次数是有控制的。这是一个整数值，默认设置为 4。用户可以控制这个值。如果需要一个比 4 大得多的值，这通常表明仿真存在更大的问题。maxAttempts=4

odeStep：

该设置允许用户为化学 ODE 积分过程选择方法。用户传递一个表，该表至少包含一个方法名。例如:

odeStep = {method='alpha-qss'}

用户还可以提供一些特定于所选方法的信息，这些方法控制数字的各个方面。这就是使用表的原因:这样额外的信息就很容易包含进去。这里有一个例子，

一些更好控制 α-QSS 的方法：

odeStep = {method='alpha-qss', eps1=1.0e-4, maxIters=5}

目前，ODE 方法的选择是 rkf 和 alpha-qss。下面直接给出了这些方法和相关参数的描述。

4.3.1 ODE 方法的选择和额外控制

如上所述，通过设置 odeStep 参数来选择 ODE 方法，从而对化学问题进行积分。在这里，我们描述了每种方法(rkf 和 alpha-qss)的控制细节。

Runge-Kutta-Felhberg 方法：

Runge-Kutta-Fehlberg 为用户提供了一个控制参数:在每一步估计误差时使用的误差 errTol。该算法采用了 Press 等人[12]提出的残差容量来估计误差。根据作者的经验，默认值 1×10^{-3} 对于与钝体高超声速反应流相关的各种非刚性化学系统都很有效。

RKF 积分器的选择和配置如下:

odeStep = {method='rkf', errTol=1.0e-3}

Mott 的 α-QSS 方法：

该α-QSS 方法有 4 个可配置参数:eps1, eps2, delta 以及 maxIter。前三个参数与第 3.3 节中提出的更新和步长估计方程一致。maxIter 参数用于控制校正器步长的迭代。

下面是一个选择和配置α-QSS 方法的例子(如果没有指定默认值):

```
odeStep = {method='alpha-qss', eps1=0.001, eps2=5.0e-4,
           delta=1.0e-10, maxIters=10}
```

5. 使用的例子

虽然反应气体模型是为 Eilmer 流动求解器开发的，但它们的服务可通过一个简单的应用程序编程接口提供。在 gas 气体模型用户指南[13]中讨论的 GasModel 和 GasState 基本类上，ThermochemicalReactor 类提供 update_state 方法。这个方法定义了反应气体模型的应用程序编程接口(API)。在一个绝热、固定体积的反应器中，充分搅拌的反应物演变过程是这个更新功能背后的概念模型。

因此，给定一个不止包含一种化学物质的特定气体模型，就可以构造一个 GasState 对象，然后将其传递给 update_state 方法，以允许化学反应在指定的时间间隔内改变物质组分。在这段时间内，体积、密度和热力学能是固定的。物种质量或摩尔分数和其它热力学量，如压力和温度，都可能发生变化。

本章的以下部分提供了一些示例，这些示例本身可能会引起您的兴趣，也可能为您构建自己的分析工具提供了良好的起点。第一个(也是最简单的)示例展示了直接使用 ThermochemicalReactor 类来计算一个孤立的气体团在短时间内的演化过程。这个示例很小，但使用了所有三种受支持的脚本语言(Lua、Python 和 Ruby)编写。第二个例子是一个小型但典型的应用程序，用于评估空气中氢燃烧反应方案的选择。最后一个例子是利用 Eilmer 流动模拟中的反应气体模型。这里需要注意，由流动求解器代码调用 ThermochemicalReactor 的 update_state 方法，而不是由用户的输入脚本调用。

5.1 固定体积的氮反应器

继续以解离氮的最小反应气体为例，我们将使用 API 构建一个固定体积的反应器，初始物包含足够高温的氮分子和原子，使反应在 300 微秒的合理时间内发生。这对于高超声速的研究者来讲是合理的。

在运行脚本的准备工作中，我们需要一个详细的气体模型文件和详细的化学文件，以供仿真类使用。输入文件(称为 nitrogen-2sp.inp) 对于我们的氮气模型，我们想要使用热完全气体模型，然后列出感兴趣的氮分子和氮原子：

```
model = 'thermally perfect gas'
species = {'N2', 'N'}
```

通过 prep-gas 程序运行这个输入文件将生成一个详细的气体模型文件(nitrogen-2sp.lua)，仿真函数可以使用这个文件：

$ prep-gas nitrogen-2sp.inp nitrogen-2sp.lua

化学输入文件与第四节所示的文件不同，因为我们省略了逆反应速率表达式，需要让求解器从平衡常数提供它们。

```
-- nitrogen-2sp-2r-Keq.lua
--
-- This chemical kinetic system provides
-- a simple nitrogen dissociation mechanism.
--
-- Author: Rowan J. Gollan
-- Date: 13-Mar-2009 (Friday the 13th)
-- Place: NIA, Hampton, Virginia, USA
--
-- History:
-- 24-Mar-2009 - reduced file to minimum input
-- 11-Aug-2015 - updated for dlang module
Reaction{
' N2 + N2 <=> N + N + N2',
fr={'Arrhenius', A=7.0e21, n=-1.6, C=113200.0},
}
Reaction{
' N2 + N <=> N + N + N',
fr={'Arrhenius', A=3.0e22, n=-1.6, C=113200.0},
}
```

这个输入文件可以用以下命令生成详细的化学文件：

$ prep-chem nitrogen-2sp.lua nitrogen-2sp-2r-Keq.lua chem.lua

记录详细的化学反应文件 chem.lua，现在可以在仿真脚本中构造反应器的对象时使用。

5.1.1 Lua 编程接口

对于 Lua API，我们将通过 gas-calc 程序访问反应气体模型的功能。这个 Lua 编程接口很好地反映了在 Eilmer 流动仿真程序中使用的 D 语言接口。

下面的脚本在命令行中被传递给 gas-calc 程序:

$ gas-calc fvreactor.lua

在第 5 行，我们首先构建了一个气体模型，由反应的两种氮组成:氮分子和氮

原子。然后，我们通过调用 ChemistryUpdate:new(在第 6 行)构造一个 ThermochemicalReactor，传递氮气模型对象和详细化学文件的名称。

```
 1 -- fvreactor.lua
 2 -- A simple fixed-volume reactor.
 3 -- PJ & RJG 2018-04-21.
 4
 5 gm = GasModel:new{"nitrogen-2sp.lua"}
 6 chemUpdate = ChemistryUpdate:new{gasmodel=gm, filename="chem.lua"}
 7
 8 gs = GasState:new{gm}
 9 gs.p = 1.0e5 -- Pa
10 gs.T = 4000.0 -- degree K
11 molef = {N2=2/3, N=1/3}
12 gs.massf = gm:molef2massf(molef)
13 gm:updateThermoFromPT(gs)
14 conc = gm:massf2conc(gs)
15
16 tFinal = 300.0e-6 -- s
17 t = 0.0
18 dt = 1.0e-6
19 dtSuggest = 1.0e-11
20 print("# Start integration")
21 f = assert(io.open("fvreactor.data", 'w'))
22 f:write('# 1:t(s) 2:T(K) 3:p(Pa) 4:massf_N2 5:massf_N 6:conc_N2 7:conc_N\n')
23 f:write(string.format("%10.3e %10.3f %10.3e %20.12e %20.12e %20.12e %20.12e\n",
24               t, gs.T, gs.p, gs.massf.N2, gs.massf.N, conc.N2, conc.N))
25 while t <= tFinal do
26 dtSuggest = chemUpdate:updateState(gs, dt, dtSuggest, gm)
27 t = t + dt
28 -- dt = dtSuggest -- uncomment this to get quicker stepping
29 gm:updateThermoFromRHOE(gs)
30 conc = gm:massf2conc(gs)
31 f:write(string.format("%10.3e %10.3f %10.3e %20.12e %20.12e %20.12e %20.12e\n",
32           t, gs.T, gs.p, gs.massf.N2, gs.massf.N, conc.N2, conc.N))
33 end
34 f:close()
35 print("# Done.")
```

有了 gas 气体模型(作为变量 gm)，我们就可以构建一个 gas 气体状态对象(第 8 行)并继续设置它的热力学值(第 9 行到第 13 行)。物种片段被指定为一个数字表，物种名称被提供给这些值的关键字。似乎只需要指定具有非零分数的物种，然而，这些分数的总和需要为 1.0。用物质的摩尔分数或浓度来计算是很方便的，

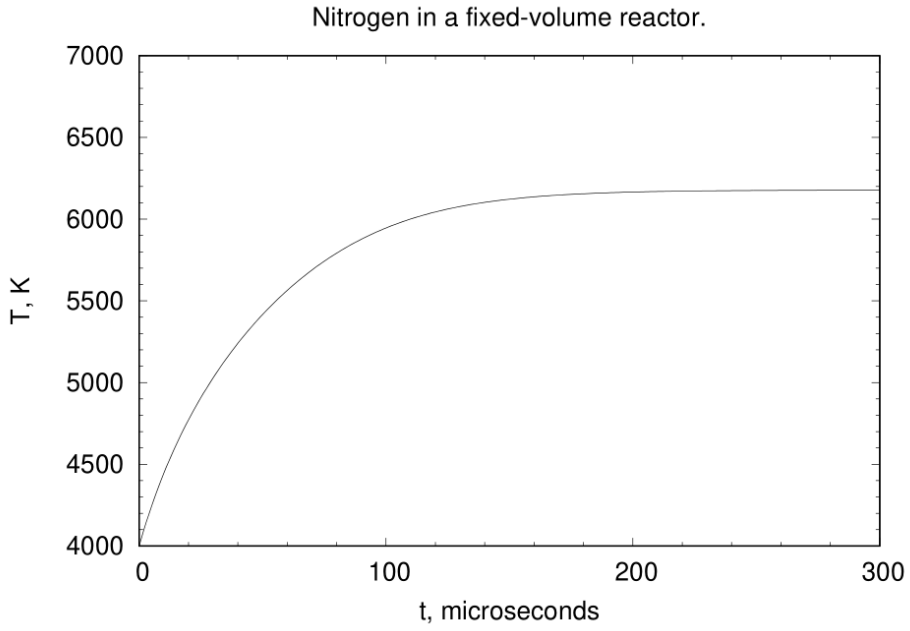
但我们必须把质量分数设为气体状态。为此，有许多方便的函数使这些转换变得容易，如第 12 行和第 14 行所示。注意，在这个例子中，气体开始时有一个显著的分解质量分数和一个足够高的温度，使反应快速进行。

从时间 0 到 tFinal，气体状态演化的计算以一些离散的时间步(持续时间 dt)进行。这是从第 25 行开始的循环，实际的 update 函数在第 26 行被调用。给 update 函数的参数是:

1. gs：气体状态对象，它的热力学值将更新；
2. dt：反应发生的时间间隔；
3. dtSuggest：热化学反应器内迭代的建议时间步长；
4. gm:气体模型对象

在完成它的工作后，update 函数返回一个新的值 dtSuggest，我们可以保留这个值以备以后使用。根据反应速率和内部步长方案，这个值可能与我们最初提供给更新函数的值显著不同。在热化学反应器更新了气体状态下的物质质量分数后，我们需要更新第 29 行上的其它热力学性质。

仿真结果如图 5.1 所示。正如预料的那样，一些氮原子重新组合成氮分子，在反应进行时提高了静态温度和压力。最终，反应器接近平衡状态。该瞬态仿真的最终状态为压力 p=145.5kPa，温度 T=6177.4K，质量分数 $massf_{N2}$= 0.86928, $massf_{N}$=0.13072，可以与 CEA2 计算的预期平衡状态进行检验。(见 5.1.4 节。)



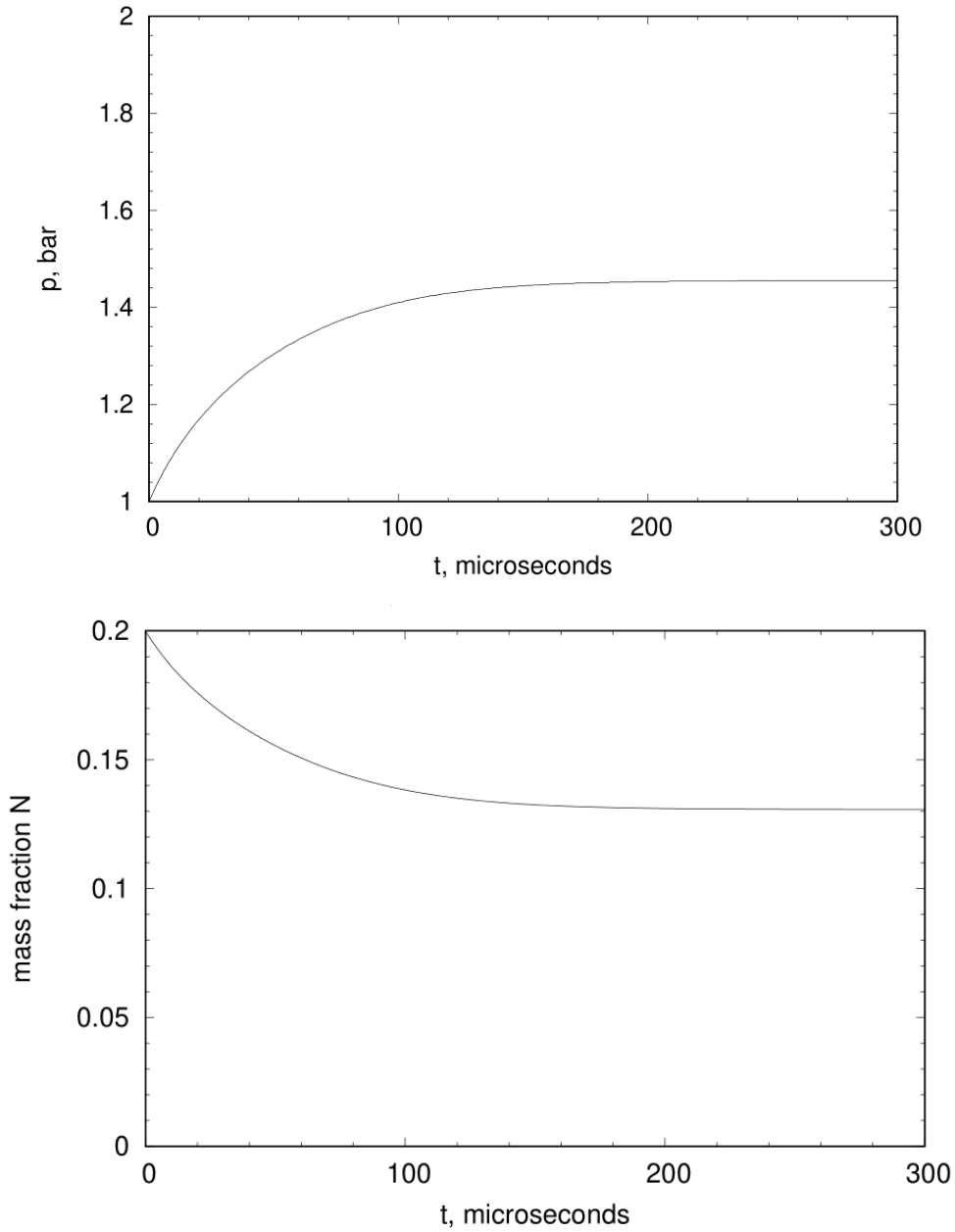


图 5.1:孤立反应器中温度、压力和原子质量分数的演化。

5.1.2 Python 编程接口

Python3 脚本语言也提供了类似的编程接口。Eilmer 软件包提供了 D 语言的 gas 气体模块作为一个可加载库，以及 Python 和 Ruby 包装器，每个包装器都可以通过它们访问这个可加载库 CFFI。

下面的脚本是对固定体积反应器计算的重新实现，这次使用的是 Python3 编程语言。我们首先通过第 12 行中的 import 语句将这个共享库加载到 Python 解释器中。

```
1 # fvreactor.py
2 # A simple fixed-volume reactor.
3 # PJ & RJG 2019-11-25
4 #
5 # To prepare:
6 # $ prep-gas nitrogen-2sp.inp nitrogen-2sp.lua
7 # $ prep-chem nitrogen-2sp.lua nitrogen-2sp-2r.lua chem.lua
8 #
9 # To run:
10 # $ python3 fvreactor.py
11
12 from eilmer.gas import GasModel, GasState, ThermochemicalReactor
13
14 gm = GasModel("nitrogen-2sp.lua")
15 reactor = ThermochemicalReactor(gm, "chem.lua")
16
17 gs = GasState(gm)
18 gs.p = 1.0e5 # Pa
19 gs.T = 4000.0 # degree K
20 gs.molef = {'N2':2/3, 'N':1/3}
21 gs.update_thermo_from_pT()
22
23 tFinal = 300.0e-6 # s
24 t = 0.0
25 dt = 1.0e-6
26 dtSuggest = 1.0e-11
27 print("# Start integration")
28 f = open("fvreactor.data", 'w')
29 f.write('# 1:t(s) 2:T(K) 3:p(Pa) 4:massf_N2 5:massf_N 6:conc_N2 7:conc_N\n')
30 f.write("%10.3e %10.3f %10.3e %20.12e %20.12e %20.12e %20.12e\n" %
31 (t, gs.T, gs.p, gs.massf[0], gs.massf[1], gs.conc[0], gs.conc[1]))
32 while t <= tFinal:
33 dtSuggest = reactor.update_state(gs, dt, dtSuggest)
34 t = t + dt
35 # dt = dtSuggest # uncomment this to get quicker stepping
36 gs.update_thermo_from_rhou()
37 f.write("%10.3e %10.3f %10.3e %20.12e %20.12e %20.12e %20.12e\n" %
38 (t, gs.T, gs.p, gs.massf[0], gs.massf[1], gs.conc[0], gs.conc[1]))
39 f.close()
40 print("# Done.")
```

尽管 Python 编程接口与 Lua 脚本语言几乎相同，但也有一些其它的便利之处。在第 20 行，我们直接指定摩尔分数，而不必首先转换为质量分数。浓度也可以直接从气体状态获得，如第 38 行所示。最后，gas-state 对象保留了用于其

构造的气体模型的引用(第 17 行)。这里的一点额外便利是，更新热属性的调用(在第 21 和 36 行)不需要显式地接收气体模型引用。

5.1.3 Ruby 编程接口

无论您在 Python 中做什么，都可以在 Ruby 中做类似的事情。虽然存在一些语言差异，但是 Ruby 脚本与 Python 脚本非常相似。反应气体模型的 Ruby 编程接口可以在 Eilmer 代码库中的自动化测试脚本中使用。

```
1 # fvreactor.rb
2 # A simple fixed-volume reactor.
3 # PJ & RJG 2019-11-27
4 #
5 # To prepare:
6 # $ prep-gas nitrogen-2sp.inp nitrogen-2sp.lua
7 # $ prep-chem nitrogen-2sp.lua nitrogen-2sp-2r.lua chem.lua
8 #
9 # To run:
10 # $ ruby fvreactor.py
11 $LOAD_PATH << '~/dgdinst/lib'
12 require 'eilmer/gas'
13
14 gm = GasModel.new("nitrogen-2sp.lua")
15 reactor = ThermochemicalReactor.new(gm, "chem.lua")
16
17 gs = GasState.new(gm)
18 gs.p = 1.0e5 # Pa
19 gs.T = 4000.0 # degree K
20 gs.molef = {'N2'=>2.0/3, 'N'=>1.0/3}
21 gs.update_thermo_from_pT()
22
23 tFinal = 300.0e-6 # s
24 t = 0.0
25 dt = 1.0e-6
26 dtSuggest = 1.0e-11
27 puts "# Start integration"
28 f = open("fvreactor.data", 'w')
29 f.write("# 1:t(s) 2:T(K) 3:p(Pa) 4:massf_N2 5:massf_N 6:conc_N2 7:conc_N\n")
30 f.write("%10.3e %10.3f %10.3e %20.12e %20.12e %20.12e %20.12e\n" %
31 [t, gs.T, gs.p, gs.massf[0], gs.massf[1], gs.conc[0], gs.conc[1]])
32 while t <= tFinal do
33 dtSuggest = reactor.update_state(gs, dt, dtSuggest)
```

```
34 t = t + dt
35 # dt = dtSuggest # uncomment this to get quicker stepping
36 gs.update_thermo_from_rhou()
37 f.write("%10.3e %10.3f %10.3e %20.12e %20.12e %20.12e %20.12e\n" %
38 [t, gs.T, gs.p, gs.massf[0], gs.massf[1], gs.conc[0], gs.conc[1]])
39 end
40 f.close()
41 puts "# Done."
```

5.1.4 CEAgas 模型

为了得到最终气体成分的平衡气体估价值，我们可以使用如下所示的 CEAgas 气体模型。在幕后，GasModel 对象调用 CEA2 程序[4]来进行详细的计算，因此，要使用这个模型，需要将 CEA2 可执行程序放在 PATH 中的某个位置。

```
# n2-n-eq-check.py
# Usage: python3 n2-n-eq-check.py
from eilmer.gas import GasModel, GasState
gmodel = GasModel('cea-n2-gas-model.lua')
gs = GasState(gmodel)
gs.p = 1.455e+05
gs.T = 6177.424
gs.update_thermo_from_pT()
print("eq gas state=", gs)
print("ceaSavedData=", gs.ceaSavedData)
```

下面的文本经过了少量的编辑，将行分成比原始输出更短的长度。

```
$ python3 n2-n-eq-check.py
eq gas state= GasState(rho=0.070243, p=145500, T=6177.42,
                       u=1.01128e+07, a=1530.8,
                       id=0, gmodel.id=0)
ceaSavedData= {'p': 145500.0, 'rho': 0.070243, 'u': 10112800.0,
               ' h': 12184200.0, 'T': 6177.42, 'a': 1530.8,
               ' Mmass': 0.024796, 'Rgas': 335.31658331989036,
               ' gamma': 1.1313, 'Cp': 8483.7, 's': 11208.3,
               ' k': 0.0, 'mu': 0.0,
               ' massf': {'N': 0.12976, 'N2': 0.87024}}
```

气体模型文件(cea-n2-gas-model.lua)的内容是给这个特定的练习量身定制的，因为我们需要指定 N2 和 N 反应物的比例。

```
model = "CEAGas"
CEAGas = {
   mixtureName = 'n2-n',
```

```
    speciesList = {"N2", "N"},
    reactants = {N2=2/3, N=1/3},
    inputUnits = "moles",
    withIons = false,
    trace = 1.0e-6
}
```

5.2 氢的点火时间

如前所述(在第 5.1 节)，完全搅拌的反应物在绝热的固定体积反应器中，其演化是主要化学更新功能背后的模型。我们可以用这一基本计算方法进行数值实验，比较几种已提出的氢燃烧反应方案的点火时间。这些计划是：

- 一项斯坦福(2011)计划，只有氢和氧参与反应(附录 A.1)；
- 只有氢和氧参与反应的 Evans 和 Schexnayder 方案(附录 A.2)；
- 在反应中包括氢、氧和氮物种的 Rogers 和 Schexnayder 方案(附录 A.3)；

下面的脚本显示了 Stanford-2011 方案的使用。使用其它方案所需要的惟一更改是第 1 行到第 3 行中提供的文件名。

```
1 spFile = "Stanford-2011-gas-model.lua"
2 reacFile = "Stanford-2011-reac-file.lua"
3 outFile = "Stanford-ignition-delay.dat"
4
5 tFinal = 1500.0e-6 -- s
6 pInit = P_atm
7 Tlow = 900.0 -- K
8 Thigh = 1300.0 -- K
9 dT = 10.0
10
11 igCriteria = 5.0e-3 -- mol/m^3 : OH
12
13 function ignition_delay(T, gm, chemUpdate)
14 local Q = gm:createGasState()
15 Q.p = pInit
16 Q.T = T
17 local total = 2 + 1 + 3.76
18 local molef = {H2=2/total, O2=1/total, N2=3.76/total}
19 Q.massf = gm:molef2massf(molef)
20 gm:updateThermoFromPT(Q)
21
```

```
22 local t = 0.0
23 local dt = 1.0e-6
24 local dtSuggest = 1.0e-11
25 while t <= tFinal do
26 dtSuggest = chemUpdate:updateState(Q, dt, dtSuggest, gm)
27 t = t + dt
28 dt = dtSuggest
29 gm:updateThermoFromRHOE(Q)
30 local conc = gm:massf2conc(Q)
31 if conc.OH > igCriteria then
32 return t
33 end
34 end
35 return false
36 end
37
38 function main()
39 local gm = GasModel:new{spFile}
40 local chemUpdate = ChemistryUpdate:new{filename=reacFile, gasmodel=gm}
41
42 local f = assert(io.open(outFile, 'w'))
43 f:write('# 1:T(K) 2:t(s)\n')
44
45 for T=Tlow,Thigh,dT do
46 local tIg = ignition_delay(T, gm, chemUpdate)
47 if tIg then
48 f:write(string.format("%20.12e %20.12e\n", T, tIg))
49 else
50 print("No ignition at T= ", T)
51 end
52 end
53
54 end
55
56 main()
```

该脚本运行了许多仿真，每一个开始都是氢、氧和氮分子的相同混合物，但初始混合物温度 T 不同，在这个数值实验中只考虑了一个压力值，那就是由气体模块(P_atm 在第 6 行)作为一个特殊常数提供的标准大气压力。

main 主函数(从第 38 行开始)协调初始温度的设置和点火时间的记录，而每个仿真的详细代码在 igntion_delay 函数中(第 13 行到 36 行)。仿真代码与前面示例中讨论的氮反应器非常相似。构建了初始气体状态(第 14 行到第 20 行)，反应

被允许在一小段时间内进行，直到点火发生。在第 31 行，点火时间定义为 OH 浓度达到指定值的时间，该值已在第 11 行给出。

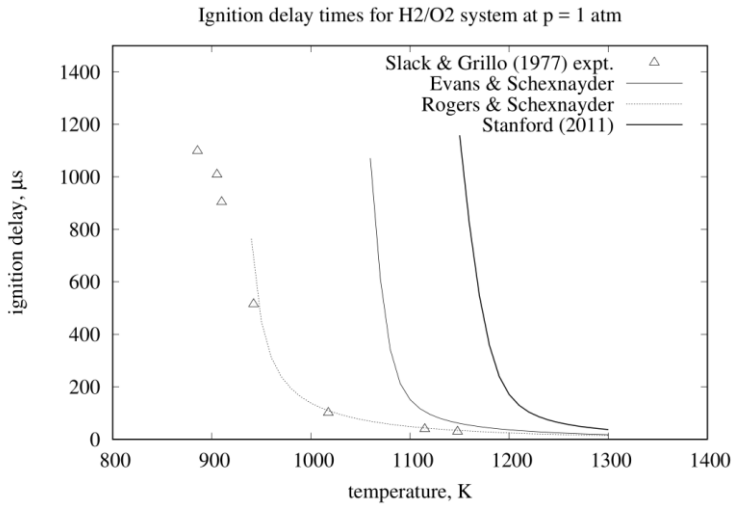


图 5.2:达到氢气-空气混合物点火的时间。

对比图 5.2 中的实验时间，我们可以得出结论：氮反应是氢着火的重要因素，特别是在温度低于 1100K 时；只有氢和氧参与的两种反应方案完全没有达到实验要求。

5.3 在二维圆柱体上离解的氮气流动

到目前为止，我们一直在独立计算中使用热化学模型。现在，我们来看一个例子，在可压缩流动模拟中与 Eilmer 配合使用模型完成。这个例子展示了在圆柱体周围建立一个简单的流域，并建立了氮离解的有限速率反应模型。氮在二维圆柱体上的高速流动是激波隧道和膨胀管设施的标志性实验，用于比较的数据来自 DLR-Göttingen 激波隧道实验室的同事。

图 5.3 的左侧显示了一个关于圆柱体部分流动区域的表示，东边界代表圆柱体的表面。在仿真中我们只考虑二维流动，但激波隧道中的实际流动将是真正的三维流动，在有限长圆柱的每一端都有显著的面外流动。这个假设是可行的，因为我们关注的重点是强烈的热化学效应，允许氮分子和原子的有限速率反应。

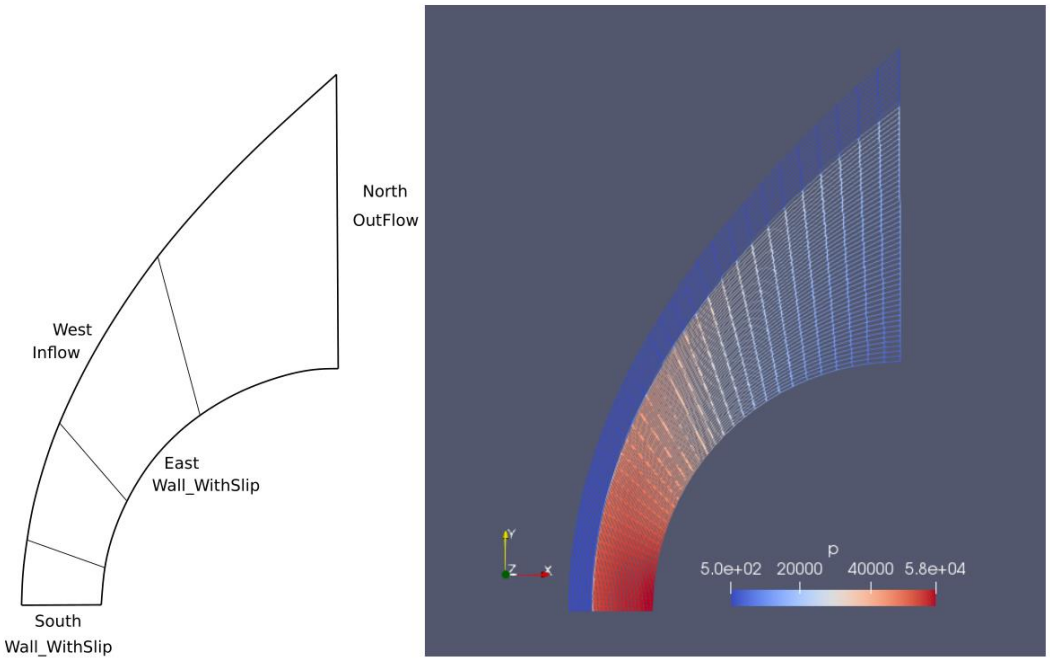


图 5.3:圆柱体前部流域几何示意图(左图)，右图为网格的压力示意图。

在二维流域中，南边界围绕着滞止线，东南交点位于柱体上的滞止点。西边界是超声速流入条件，这是由激波燃烧器产生的。北边界截断圆柱体顶部的流域，并指定为气体在(预测)超声速条件下穿过的简单外流边界。其它的径向线代表了两根线组成全域网格的多个块之间的边界，如图左所示。我们使用多个模块的目的是为了快速完成仿真。在一个好的工作站上，如果使用 4 核处理器，只需计算不到一分钟就能完成。

5.3.1 Eilmer 输入脚本

使用 Eilmer 设置流动仿真的最佳指导可以在用户指南[13]、[14]和[15]中找到，所以这些报告中的材料是充分理解下面所示的仿真输入脚本的前提条件。除了非反应仿真所需的设置外，我们的基本任务是指定参与化学物质的初始质量分数和流入质量分数，并指定详细的化学文件。

```
1 -- n90.lua
2 -- RG & PJ 2015-03-09
3 -- 2015-04-22 build an original grid in this script
4 -- 2019-05-25 Billig shock-shape correlation used to shape grid.
5 --
6 config.title = "Cylinder in dissociating nitrogen flow."
7 print(config.title)
```

```
8
9 nsp, nmodes, gmodel = setGasModel('nitrogen-2sp.lua')
10 inflow = FlowState:new{p=500.0, T=700.0, velx=5000.0, massf={N2=1.0}}
11 initial = FlowState:new{p=5.0, T=300.0, massf={N2=1.0}}
12 Minf = inflow.velx / inflow.a
13 print("Minf=", Minf)
14
15 config.reacting = true
16 config.reactions_file = 'e4-chem.lua'
17
18 require "billig_patch"
19 R = 0.045 -- m
20 bp = billig_patch.make_patch{Minf=Minf, R=R, xc=R, scale=0.95}
21 cf = RobertsFunction:new{end0=true, end1=false, beta=1.1}
22 grid = StructuredGrid:new{psurface=bp.patch, niv=61, njv=41,
23                           cfList={west=cf, east=cf}}
24
25 -- We can leave east and south as slip-walls.
26 blks = FBArray:new{grid=grid, initialState=initial, label="blk",
27                    bcList={west=InFlowBC_Supersonic:new{flowState=inflow},
28                            north=OutFlowBC_Simple:new{}},
29                    nib=1, njb=4}
30
31 -- Set a few more config options.
32 -- To get a reasonable start, we needed to set dt_init.
33 config.max_time = 100.0e-6
34 config.max_step = 40000
35 config.dt_plot = 20.0e-6
36 config.dt_init = 5.0e-9
```

第 9 行是指定详细气体模型文件的地方。只能为仿真指定一个气体模型，但是，可以在输入脚本中使用其它气体模型，这就需要使用 5.1.1 节中描述的 Lua 编程接口。注意，setGasModel 函数返回了三个项，第三个是对已初始化气体模型的引用。现在可以在 Lua 输入脚本中使用它，例如，可以用来初始化一个 ThermochemicalReactor 对象。在这里，我们没有直接使用返回气体模型，但是知道它是可用的就很好了。

第 15 和 16 行指定我们希望在仿真过程中的活化化学反应，并指定详细的化学文件。稍后，当仿真运行时，将根据该文件中的信息初始化热化学反应器。

第 10 行和第 11 行，指定了流入和未分解氮的初始气体状态，质量分数被指定为只包含氮分子条目的表格。在每种情况下，未指定的氮原子质量分数都假定

为零。

输入脚本中的其它行描述了流域(第 18-20 行)，以及指定如何离散它(第 21-23 行)，并设置一个由网格和边界条件信息组成的 FluidBlocks 数组(第 26-29 行)。输入脚本的最后几行(第 33-36 行)设置了一些仿真控制参数。同样，请参阅流动求解器用户指南[14]以获得详细信息。

5.3.2 计算结果

在仿真的最后，一个高温高压气体的激波层已经发展到圆柱体之上。从图 5.3 右侧的压力值可以看到这个激波层的颜色。氮原子的温度和质量分数如图 5.4 所示。随着气体从左到右流动，经过冲击处理后，当气体接近圆柱体表面时，您可以看到分解的增加和相应的温度下降。

相对于非反应气体，有限速率化学反应的影响之一是降低激波层厚度。如果反应被抑制，激波层将比目前定义的允许区域更大。如果您想试验一个无反应仿真，在输入脚本的第 15 行上设置 config.reacting = false。同样，将 scale=0.95(在第 20 行)的值改为类似于 1.1 的值，以适应更厚的激波层。

图 5.5 显示了沿滞止线的温度。从图的左到右移动，激波的出现导致温度突然上升，然后在气体接近圆柱体表面的时间内，随着离解反应吸收热能，温度发生松弛。在无反应的情况下，气体温度会在激波处跳跃，然后随着气体接近滞止点而缓慢地继续增加。将 Eilmer 数据与 Sebastian Karl (DLR Göttingen)提供的参考数据进行比较是合理的，因为 Eilmer 仿真是在一个非常低分辨率的网格上进行的，而且根本没有对圆柱体上的边界层进行建模。

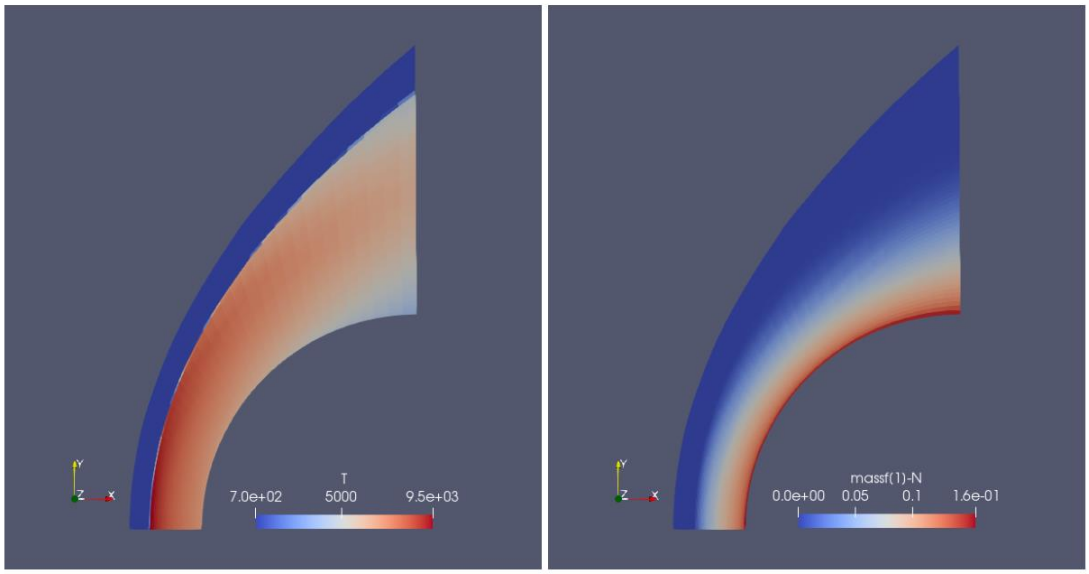


图 5.4:n90 钝体试验中氮原子的温度和质量分数。

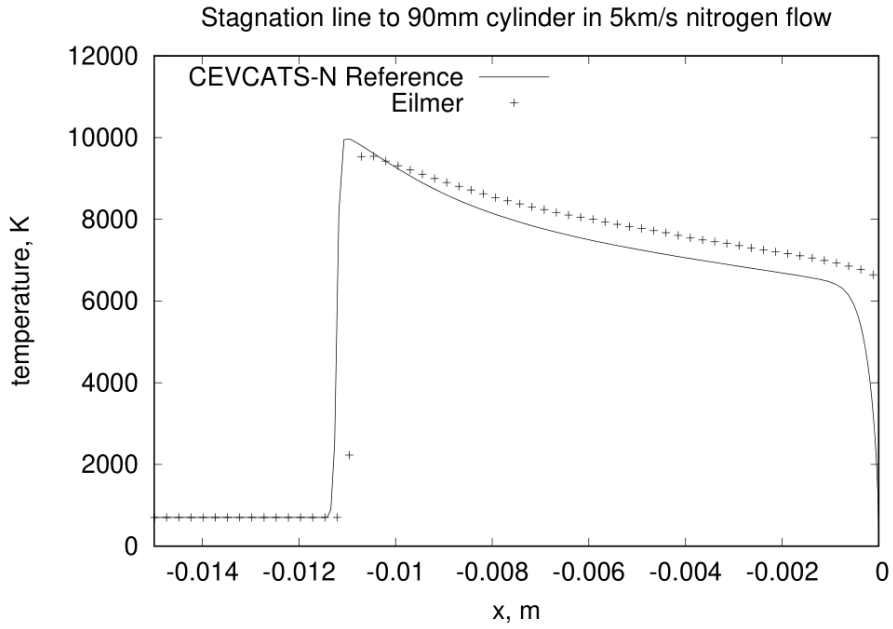


图 5.5:沿停滞线的温度数据。位置 x=0 处在圆柱体的表面上。

参考文献

[1]. B. J. McBride and S. Gordon. Computer program for calculation of complex chemical equilibrium compositions and applications. Part 2: Users manual and program description. Reference Publication 1311, NASA, 1996.

[2]. R. N. Gupta, J. M. Yos, R. A. Thompson, and K.-P. Lee. A review of reaction rates and thermodynamic and transport properties for an 11-species air model for chemical and thermal nonequilibrium calculations to 30 000 k. Reference Publication 1232, NASA, 1990.

[3]. C.R. Wilke. A viscosity equation for gas mixtures. Journal of Chemical Physics, 18:517–519, 1950.

[4]. S. Gordon and B. J. McBride. Computer program for calculation of complex chemical equilibrium compositions and applications. Part 1: Analysis. Reference Publication 1311, NASA, 1994.

[5]. E.S. Oran and J.P. Boris. Numerical Simulation of Reactive Flow. Cambridge University Press, New York, USA, 2nd edition, 2001.

[6]. E. Fehlberg. Low-order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems. Technical Report R-315, NASA, 1969.

[7]. D. R. Mott. New Quasi-Steady-State and Partial-Equilibrium Methods for Integrating Chemically Reacting Systems. PhD thesis, University of Michigan, 1999.

[8]. D. R. Mott, E. S. Oran, and B. van Leer. A quasi-steady-state solver fot the stiff ordinary differential equations of reaction kinetics. Journal of Computational Physics, 164:407–428, 2000.

[9]. S. R. Qureshi and R. Prosser. Implementation of α-qss stiff integrations methods for solving the detailed combustion chemistry. In Proceedings of the World Congress on Engineering 2007, volume II, pages 1352 – 1357, 2007.

[10]. T.R. Young and J.P. Boris. A numerical technique for solving stiff ordinary

differential equations associated with the chemical kinetics of reactive-flow problems. Journal of Physical Chemistry, 81(25):2424–2427, 1977.

[11]. B. T. O'Flaherty. Reducing the Global Warming of Coal Mine Ventilation Air by Combustion in a Free-Piston Engine. PhD thesis, The University of Queensland, 2012.

[12]. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. Numerical Recipes: The Art of Scientific Computing. Cambridge University Press, New York, USA, 3rd edition, 2007.

[13]. R. J. Gollan and P. A. Jacobs. The Eilmer 4.0 flow simulation program: Guide to the basic gas models package, including gas-calc and the API. School of Mechanical and Mining Engineering Technical Report 2017/27, The University of Queensland, Brisbane, Australia, February 2018.

[14]. Peter A. Jacobs and Rowan J. Gollan. The Eilmer 4.0 flow simulation program: Guide to the transient flow solver, including some examples to get you started. School of Mechanical and Mining Engineering Technical Report 2017/26, The University of Queensland, Brisbane, Australia, February 2018.

[15]. Peter A. Jacobs, Rowan J. Gollan, and Ingo Jahn. The Eilmer 4.0 flow simulation program: Guide to the geometry package, for construction of flow paths. School of Mechanical and Mining Engineering Technical Report 2017/25, The University of Queensland, Brisbane, Australia, February 2018.

A. 氢燃烧的输入文件

A.1 Stanford, 2011

```
1 model = 'thermally perfect gas'
2 species = {'H2', 'O2', 'H2O', 'HO2', 'H2O2', 'OH', 'O', 'H', 'N2'}
```

```
1 -- Author: Rowan J. Gollan
2 -- Date: 2015-03-17
3 --
4 -- Reference:
5 -- Hong, Z., Davidson, D.F. and Hanson, R.K. (2011)
6 -- An improved H2/O2 mechanism based on recent
7 -- shock tube/laser absorption measurements.
8 -- Combustion and Flame, 158, pp. 633--644
9 --
10 -- NOTE:
11 -- Table 1 in Hong et al contains the suggested
12 -- reaction mechanism with reaction rates.
13 -- However, there is also a Chemkin input file
14 -- available online where the article is hosted.
15 -- The input file has some differences with regards
16 -- to efficiency values. I have adopted those
17 -- values from the supplied input file.
18 --
19 -- Updated: 2016-06-23
20 -- Use new format for Eilmer4
21
22 S = 1.0/1.987
23
24 Config{
25         odeStep = {method='alpha-qss'},
26 }
27
28 Reaction{
29             'H + O2 <=> OH + O',
30             fr={'Arrhenius', A=1.04e14, n=0.0, C=15286 * S},
31             label='r1'
32 }
33
34 Reaction{
```

```
35             'H + O2 (+ M) <=> HO2 (+ M)',
36             fr={'pressure dependent',
37             kInf={A=5.59e13, n=0.2, C=0.0},
38             k0={A=3.70e19, n=-1.0, C=0.0},
39             Troe={F_cent=0.8}
40 },
41     efficiencies={H2O=1.0,H=0.0,O2=0.0,OH=0.0,O=0.0,HO2=0.0,H2O2=0.0},
42     label='r2b'
43 }
44
45 Reaction{
46     'H + O2 (+ M) <=> HO2 (+ M)',
47     fr={'pressure dependent',
48         kInf={A=5.59e13, n=0.2, C=0.0},
49         k0={A=5.69e18, n=-1.1, C=0.0},
50         Troe={F_cent=0.7}
51     },
52     efficiencies={O2=1.0,H2O=0.0,H=0.0,OH=0.0,O=0.0,HO2=0.0,H2O2=0.0},
53     label='r2c'
54 }
55
56 Reaction{
57     'H + O2 (+ M) <=> HO2 (+ M)',
58     fr={'pressure dependent',
59         kInf={A=5.59e13, n=0.2, C=0.0},
60         k0={A=2.65e19, n=-1.3, C=0.0},
61         Troe={F_cent=0.7}
62     },
63     efficiencies={H2=2.5,H2O2=12.0,H2O=0.0,O2=0.0},
64     label='r2d'
65 }
66
67 Reaction{
68     'H2O2 (+ M) <=> 2OH (+ M)',
69     fr={'pressure dependent',
70         kInf={A=8.59e14, n=0.0, C=48560 * S},
71         k0={A=9.55e15, n=0.0, C=42203 * S},
72         Troe={F_cent=1.0}
73     },
74     efficiencies={N2=1.5,H2=2.5,H2O=15,H2O2=15},
75     label='r3'
76 }
77
78 -- Reaction 4 appears twice.
```

```
79 -- The reaction rate constants are added together
80 -- as proposed by Hong et al in Section 2.5.
81 -- To achieve the same effect, the reaction can just
82 -- be listed twice with different reaction rates.
83 Reaction{
84     'OH + H2O2 <=> H2O + HO2',
85     fr={'Arrhenius', A=1.74e12, n=0.0, C=318 * S},
86     label='r4a'
87 }
88 Reaction{
89     'OH + H2O2 <=> H2O + HO2',
90     fr={'Arrhenius', A=7.59e13, n=0.0, C=7269 * S},
91     label='r4b'
92 }
93
94 Reaction{
95     'OH + HO2 <=> H2O + O2',
96     fr={'Arrhenius', A=2.89e13, n=0.0, C=-500 * S},
97     label='r5'
98 }
99
100 Reaction{
101     'HO2 + HO2 <=> H2O2 + O2',
102     fr={'Arrhenius', A=1.30e11, n=0.0, C=-1603 * S},
103     label='r6a'
104 }
105 Reaction{
106     'HO2 + HO2 <=> H2O2 + O2',
107     fr={'Arrhenius', A=4.20e14, n=0.0, C=11980 * S},
108     label='r6b'
109 }
110
111 Reaction{
112     'H2O + M <=> H + OH + M',
113     fr={'Arrhenius', A=6.06e27, n=-3.31, C=120770 * S},
114     efficiencies={O2=1.5,H2=3.0,H2O=0.0},
115     label='r7a'
116 }
117 Reaction{
118     'H2O + H2O <=> H + OH + H2O',
119     fr={'Arrhenius', A=1.00e26, n=-2.44, C=120160 * S},
120     label='r7b'
121 }
122
```

```
Reaction{
    'OH + OH <=> H2O + O',
    fr={'Arrhenius', A=3.57e4, n=2.4, C=-2111 * S},
    label='r8'
}

Reaction{
    'O + H2 <=> H + OH',
    fr={'Arrhenius', A=3.82e12, n=0.0, C=7948 * S},
    label='r9a'
}
Reaction{
    'O + H2 <=> H + OH',
    fr={'Arrhenius', A=8.79e14, n=0.0, C=19170 * S},
    label='r9b'
}

Reaction{
    'H2 + OH <=> H2O + H',
    fr={'Arrhenius', A=2.17e8, n=1.52, C=3457 * S},
    label='r10'
}

Reaction{
    'H + HO2 <=> OH + OH',
    fr={'Arrhenius', A=7.08e13, n=0.0, C=300 * S},
    label='r11'
}

Reaction{
    'H + HO2 <=> H2O + O',
    fr={'Arrhenius', A=1.45e12, n=0.0, C=0.0},
    label='r12'
}

Reaction{
    'H + HO2 <=> H2 + O2',
    fr={'Arrhenius', A=3.66e6, n=2.087, C=-1450 * S},
    label='r13'
}

Reaction{
    'O + HO2 <=> OH + O2',
    fr={'Arrhenius', A=1.63e13, n=0.0, C=-445 * S},
```

```
167     label='r14'
168 }
169
170 Reaction{
171     'H2O2 + H <=> HO2 + H2',
172     fr={'Arrhenius', A=1.21e7, n=2.0, C=5200 * S},
173     label='r15'
174 }
175
176 Reaction{
177     'H2O2 + H <=> H2O + OH',
178     fr={'Arrhenius', A=1.02e13, n=0.0, C=3577 * S},
179     label='r16'
180 }
181
182 Reaction{
183     'H2O2 + O <=> OH + HO2',
184     fr={'Arrhenius', A=8.43e11, n=0.0, C=3970 * S},
185     label='r17'
186 }
187
188 Reaction{
189     'H2 + M <=> H + H + M',
190     fr={'Arrhenius', A=5.84e18, n=-1.1, C=104380 * S},
191     efficiencies={H2O=14.4,H2O2=14.4,H2=0.0,O2=0.0},
192     label='r18a'
193 }
194 Reaction{
195     'H2 + H2 <=> H + H + H2',
196     fr={'Arrhenius', A=9.03e14, n=0.0, C=96070 * S},
197     label='r18b'
198 }
199 Reaction{
200     'H2 + O2 <=> H + H + O2',
201     fr={'Arrhenius', A=4.58e19, n=-1.4, C=104380 * S},
202     label='r18c'
203 }
204
205 Reaction{
206     'O + O + M <=> O2 + M',
207     fr={'Arrhenius', A=6.16e15, n=-0.5, C=0.0},
208     efficiencies={H2=2.5,H2O=12,H2O2=12},
209     label='r19'
210 }
```

```
211
212 Reaction{
213     'O + H + M <=> OH + M',
214     fr={'Arrhenius', A=4.71e18, n=-1.0, C=0.0},
215     efficiencies={H2=2.5,H2O=12,H2O2=12},
216     label='r20'
217 }
```

A.2 Evans-Schexnayder

```
1 model = 'thermally perfect gas'
2 species = {'H2', 'O2', 'H2O', 'HO2', 'OH', 'O', 'H', 'N2'}
```

```
1 -- Author: Rowan J. Gollan
2 -- Date: 02-Feb-2010
3 -- Place: Poquoson, Virginia, USA
4 --
5 -- Adapted from Python file: evans_schexnayder.py
6 --
7 -- This file provides four chemical kinetic descriptions
8 -- of hydrogen combustion. You can select between the various
9 -- options below by setting the 'model' variable below to one of
10 -- the strings listed below.
11 --
12 -- REDUCED : a 7-species, 8-reactions description of hydrogen
13 -- combustion in pure oxygen
14 -- PURE_O2 : a 7-species, 16-reactions description of hydrogen
15 -- combustion in pure oxygen
16 -- IN_AIR : a 12-species, 25-reactions description of hydrogen
17 -- combustion in air (N2 and O2)
18 -- INERT_N2 : an 8-species, 16-reactions description of hydrogen
19 -- combustion in air with inert N2 (acting as diluent only).
20 --
21 -- The numbering of reactions in this file corresponds to
22 -- Table 1 in Evans and Schexnayder (1980).
23 --
24 -- Reference:
25 -- Evans, J.S. and Shexnayder Jr, C.J. (1980)
26 -- Influence of Chemical Kinetics and Unmixedness
27 -- on Burning in Supersonic Hydrogen Flames
```

```
28 -- AIAA Journal 18:2 pp 188--193
29 --
30 -- History:
31 -- 07-Mar-2006 -- first prepared
32 --
33 -- Species used in REDUCED: O, O2, H, H2, H2O, OH, N2
34 -- Species used in PURE_O2: O, O2, H, H2, H2O, OH, HO2
35 -- Species used in IN_AIR: O, O2, N, N2, H, H2, H2O, HO2, OH, NO, NO2, HNO2
36 -- Species used in INERT_N2: O, O2, H, H2, H2O, OH, HO2, N2
37
38 options = {
39     REDUCED=true,
40     PURE_O2=true,
41       IN_AIR=true,
42   INERT_N2=true
43 }
44
45 -- User selects model here
46 model = 'INERT_N2'
47
48 -- Check that selection is valid
49 if options[model] == nil then
50   print("User selected model: ", model)
51   print("is not valid.")
52   print("Valid models are:")
53   for m,_ in pairs(options) do
54     print(m)
55   end
56 end
57
58 Config{
59   odeStep = {method='alpha-qss'},
60     -- tightTempCoupling = true
61 }
62
63 Reaction{
64   'HNO2 + M <=> NO + OH + M',
65   fr={'Arrhenius', A=5.0e17, n=-1.0, C=25000.0},
66   br={'Arrhenius', A=8.0e15, n=0.0, C=-1000.0},
67   label='r1'
68 }
69
70 Reaction{
71   'NO2 + M <=> NO + O + M',
```

```
72    fr={'Arrhenius', A=1.1e16, n=0.0, C=32712.0},
73    br={'Arrhenius', A=1.1e15, n=0.0, C=-941.0},
74    label='r2'
75 }
76
77 Reaction{
78    'H2 + M <=> H + H + M',
79    fr={'Arrhenius', A=5.5e18, n=-1.0, C=51987.0},
80    br={'Arrhenius', A=1.8e18, n=-1.0, C=0.0},
81    label='r3'
82 }
83
84 Reaction{
85    'O2 + M <=> O + O + M',
86    fr={'Arrhenius', A=7.2e18, n=-1.0, C=59340.0},
87    br={'Arrhenius', A=4.0e17, n=-1.0, C=0.0},
88 label='r4'
89 }
90
91 Reaction{
92    'H2O + M <=> OH + H + M',
93    fr={'Arrhenius', A=5.2e21, n=-1.5, C=59386.0},
94    br={'Arrhenius', A=4.4e20, n=-1.5, C=0.0},
95    label='r5'
96 }
97
98 Reaction{
99    'OH + M <=> O + H + M',
100    fr={'Arrhenius', A=8.5e18, n=-1.0, C=50830.0},
101    br={'Arrhenius', A=7.1e18, n=-1.0, C=0.0},
102    label='r6'
103 }
104
105 Reaction{
106    'HO2 + M <=> H + O2 + M',
107    fr={'Arrhenius', A=1.7e16, n=0.0, C=23100.0},
108    br={'Arrhenius', A=1.1e16, n=0.0, C=-440.0},
109    label='r7'
110 }
111
112 Reaction{
113    'H2O + O <=> OH + OH',
114    fr={'Arrhenius', A=5.8e13, n=0.0, C=9059.0},
115    br={'Arrhenius', A=5.3e12, n=0.0, C=503.0},
```

```
116   label='r8'
117 }
118
119 Reaction{
120   'H2O + H <=> OH + H2',
121   fr={'Arrhenius', A=8.4e13, n=0.0, C=10116.0},
122   br={'Arrhenius', A=2.0e13, n=0.0, C=2600.0},
123   label='r9'
124 }
125
126 Reaction{
127   'O2 + H <=> OH + O',
128   fr={'Arrhenius', A=2.2e14, n=0.0, C=8455.0},
129   br={'Arrhenius', A=1.5e13, n=0.0, C=0.0},
130   label='r10'
131 }
132
133 Reaction{
134   'H2 + O <=> OH + H',
135   fr={'Arrhenius', A=7.5e13, n=0.0, C=5586.0},
136   br={'Arrhenius', A=3.0e13, n=0.0, C=4429.0},
137   label='r11'
138 }
139
140 Reaction{
141   'H2 + O2 <=> OH + OH',
142   fr={'Arrhenius', A=1.7e13, n=0.0, C=24232.0},
143   br={'Arrhenius', A=5.7e11, n=0.0, C=14922.0},
144   label='r12'
145 }
146
147 Reaction{
148   'H2 + O2 <=> H + HO2',
149   fr={'Arrhenius', A=1.9e13, n=0.0, C=24100.0},
150   br={'Arrhenius', A=1.3e13, n=0.0, C=0.0},
151   label='r13'
152 }
153
154 Reaction{
155   'OH + OH <=> H + HO2',
156   fr={'Arrhenius', A=1.7e11, n=0.5, C=21137.0},
157   br={'Arrhenius', A=6.0e13, n=0.0, C=0.0},
158   label='r14'
159 }
```

```
160
161 Reaction{
162   'H2O + O <=> H + HO2',
163   fr={'Arrhenius', A=5.8e11, n=0.5, C=28686.0},
164   br={'Arrhenius', A=3.0e13, n=0.0, C=0.0},
165   label='r15'
166 }
167
168 Reaction{
169   'OH + O2 <=> O + HO2',
170   fr={'Arrhenius', A=3.7e11, n=0.64, C=27840.0},
171   br={'Arrhenius', A=1.0e13, n=0.0, C=0.0},
172   label='r16'
173 }
174
175 Reaction{
176   'H2O + O2 <=> OH + HO2',
177   fr={'Arrhenius', A=2.0e11, n=0.5, C=36296.0},
178   br={'Arrhenius', A=1.2e13, n=0.0, C=0.0},
179   label='r17'
180 }
181
182 Reaction{
183   'H2O + OH <=> H2 + HO2',
184   fr={'Arrhenius', A=1.2e12, n=0.21, C=39815.0},
185   br={'Arrhenius', A=1.7e13, n=0.0, C=12582.0},
186   label='r18'
187 }
188
189 Reaction{
190   'O + N2 <=> N + NO',
191   fr={'Arrhenius', A=5.0e13, n=0.0, C=37940.0},
192   br={'Arrhenius', A=1.1e13, n=0.0, C=0.0},
193   label='r19'
194 }
195
196 Reaction{
197   'H + NO <=> N + OH',
198   fr={'Arrhenius', A=1.7e14, n=0.0, C=24500.0},
199   br={'Arrhenius', A=4.5e13, n=0.0, C=0.0},
200   label='r20'
201 }
202
203 Reaction{
```

```
204   'O + NO <=> N + O2',
205   fr={'Arrhenius', A=2.4e11, n=0.5, C=19200.0},
206   br={'Arrhenius', A=1.0e12, n=0.5, C=3120.0},
207   label='r21'
208 }
209
210 Reaction{
211   'NO + OH <=> H + NO2',
212   fr={'Arrhenius', A=2.0e11, n=0.5, C=15500.0},
213   br={'Arrhenius', A=3.5e14, n=0.0, C=740.0},
214   label='r22'
215 }
216
217
218 Reaction{
219   'NO + O2 <=> O + NO2',
220   fr={'Arrhenius', A=1.0e12, n=0.0, C=22800.0},
221   br={'Arrhenius', A=1.0e13, n=0.0, C=302.0},
222   label='r23'
223 }
224
225 Reaction{
226   'NO2 + H2 <=> H + HNO2',
227   fr={'Arrhenius', A=2.4e13, n=0.0, C=14500.0},
228   br={'Arrhenius', A=5.0e11, n=0.5, C=1500.0},
229   label='r24'
230 }
231
232 Reaction{
233   'NO2 + OH <=> NO + HO2',
234   fr={'Arrhenius', A=1.0e11, n=0.5, C=6000.0},
235   br={'Arrhenius', A=3.0e12, n=0.5, C=1200.0},
236   label='r25'
237 }
238
239 reactions_list = {}
240
241 if model == 'REDUCED' then
242   reactions_list = {'r3', 'r4', 'r5', 'r6', 'r8', 'r9', 'r10', 'r11'}
243 end
244
245 if model == 'PURE_O2' or model == 'INERT_N2' then
246   reactions_list = {'r3', 'r4', 'r5', 'r6', 'r7', 'r8', 'r9', 'r10',
247   'r11', 'r12', 'r13', 'r14', 'r15', 'r16', 'r17', 'r18'}
```

```
248 end
249
250
251 if model ~= 'IN_AIR' then
252 -- For all other models we select only a subset.
253 selectOnlyReactionsWithLabel(reactions_list)
254 end
```

A.3 Rogers-Schexnayder

```
1 model = 'thermally perfect gas'
2 species = {'O', 'O2', 'N', 'N2', 'H', 'H2',
3 'H2O', 'HO2', 'OH', 'NO', 'NO2',
4 'HNO2', 'HNO3', 'O3', 'H2O2', 'HNO' }
```

```
1 -- Author: Rowan J. Gollan
2 -- Date: 29-Mar-2009
3 -- Place: Poquoson, Virginia, USA
4 --
5 -- Adapted from Python file: rogers_schexnayder.py
6 --
7 -- This file provides a reaction scheme for
8 -- hydrogen combustion in air.
9 -- NOTE: This scheme does not include carbonaceous compounds
10 -- or Argon (or any of the associated reactions).
11 --
12 -- Reference:
13 -- Rogers, R.C. and Schexnayder, Jr., C.J. (1981)
14 -- Chemical Kinetic Analysis of Hydroden-Air
15 -- Ignition and Reaction Times
16 -- NASA Technical Paper 1856
17 --
18 -- Species used: O, O2, N, N2, H, H2, H2O, HO2, OH, NO, NO2, HNO2, HNO3, O3, H2O2,
HNO
19 --
20 -- Updated 2016-07-31
21 -- Updated for eilmer4
22
23 Config{
24    odeStep = {method='alpha-qss'},
```

```
25    tightTempCoupling = true
26 }
27
28 Reaction{
29    'O2 + M <=> O + O + M',
30    fr={"Arrhenius", A=0.72e19, n=-1.0, C=59340.0},
31    efficiencies={O2=4.0, O=10.0, H2O=2.0},
32    label='r1'
33 }
34
35 Reaction{
36    'M + H2 <=> H + H + M',
37    fr={'Arrhenius', A=0.55e19, n=-1.0, C=51987.0},
38    efficiencies={H=5.0, H2=2.0, H2O=8.0},
39    label='r2'
40 }
41
42 Reaction{
43    'M + H2O <=> H + OH + M',
44    fr={'Arrhenius', A=0.52e22, n=-1.5, C=59386.0},
45    efficiencies={H2O=6.0},
46    label='r3'
47 }
48
49 Reaction{
50    'H + O2 + M <=> HO2 + M',
51    fr={'Arrhenius', A=0.23e16, n=0.0, C=-403.0},
52    efficiencies={H2=2.0, H2O=13.0},
53    label='r4'
54 }
55
56 Reaction{
57    'M + NO2 <=> NO + O + M',
58    fr={'Arrhenius', A=0.11e17, n=0.0, C=32710.0},
59    label='r5'
60 }
61
62 Reaction{
63    'M + NO <=> N + O + M',
64    fr={'Arrhenius', A=0.41e19, n=-1.0, C=75330.0},
65    label='r6'
66 }
67
68 -- not included, ignoring the carbonaceous compounds
```

```
69 r7 = 'M + O + CO <=> CO2 + M'
70
71 Reaction{
72   'M + H + NO <=> HNO + M',
73   fr={'Arrhenius', A=0.54e16, n=0.0, C=-300.0},
74   efficiencies={H2O=3.0},
75   label='r8'
76 }
77
78 Reaction{
79   'M + H2O2 <=> OH + OH + M',
80   fr={'Arrhenius', A=0.12e18, n=0.0, C=22899.0},
81   efficiencies={H2O=6.0},
82   label='r9'
83 }
84
85 Reaction{
86   'M + OH + NO <=> HNO2 + M',
87   fr={'Arrhenius', A=0.80e16, n=0.0, C=-1000.0},
88   label='r10'
89 }
90
91 Reaction{
92   'M + OH + NO2 <=> HNO3 + M',
93   fr={'Arrhenius', A=0.13e17, n=0.0, C=-1107.0},
94   label='r11'
95 }
96
97 Reaction{
98   'M + O3 <=> O2 + O + M',
99   fr={'Arrhenius', A=0.13e22, n=-2.0, C=12800.0},
100   efficiencies={O2=1.5},
101   label='r12'
102 }
103
104 r13 = 'M + HCO <=> CO + H + M'
105
106 Reaction{
107   'M + O + H <=> OH + M',
108   fr={'Arrhenius', A=0.71e19, n=-1.0, C=0.0},
109   label='r14'
110 }
111
112 Reaction{
```

```
113   'H2O + O <=> OH + OH',
114   fr={'Arrhenius', A=0.58e14, n=0.0, C=9059.0},
115   label='r15'
116 }
117
118 Reaction{
119   'H2 + OH <=> H2O + H',
120   fr={'Arrhenius', A=0.20e14, n=0.0, C=2600.0},
121   label='r16'
122 }
123
124 Reaction{
125   'O2 + H <=> OH + O',
126   fr={'Arrhenius', A=0.22e15, n=0.0, C=8455.0},
127   label='r17'
128 }
129
130 Reaction{
131   'H2 + O <=> OH + H',
132   fr={'Arrhenius', A=0.75e14, n=0.0, C=5586.0},
133   label='r18'
134 }
135
136 Reaction{
137   'H2 + O2 <=> OH + OH',
138   fr={'Arrhenius', A=0.10e14, n=0.0, C=21641.0},
139   label='r19'
140 }
141
142 Reaction{
143   'H + HO2 <=> H2 + O2',
144   fr={'Arrhenius', A=0.24e14, n=0.0, C=350.0},
145   label='r20'
146 }
147
148 Reaction{
149   'H2 + O2 <=> H2O + O',
150   fr={'Arrhenius', A=0.41e14, n=0.0, C=25400.0},
151   label='r21'
152 }
153
154 Reaction{
155   'H + HO2 <=> OH + OH',
156   fr={'Arrhenius', A=0.24e15, n=0.0, C=950.0},
```

```
157    label='r22'
158 }
159
160 Reaction{
161    'H2O + O <=> H + HO2',
162    fr={'Arrhenius', A=0.58e12, n=0.5, C=28686.0},
163    label='r23'
164 }
165
166 Reaction{
167    'O + HO2 <=> OH + O2',
168    fr={'Arrhenius', A=0.5e14, n=0.0, C=503.0},
169    label='r24'
170 }
171
172 Reaction{
173    'OH + HO2 <=> O2 + H2O',
174    fr={'Arrhenius', A=0.3e14, n=0.0, C=0.0},
175    label='r25'
176 }
177
178 Reaction{
179    'H2 + HO2 <=> H2O + OH',
180    fr={'Arrhenius', A=0.2e14, n=0.0, C=12582.0},
181    label='r26'
182 }
183
184 Reaction{
185    'HO2 + H2 <=> H + H2O2',
186    fr={'Arrhenius', A=0.73e12, n=0.0, C=9400.0},
187    label='r27'
188 }
189
190 Reaction{
191    'H2O2 + H <=> OH + H2O',
192    fr={'Arrhenius', A=0.32e15, n=0.0, C=4504.0},
193    label='r28'
194 }
195
196 Reaction{
197    'HO2 + OH <=> O + H2O2',
198    fr={'Arrhenius', A=0.52e11, n=0.5, C=10600.0},
199    label='r29'
200 }
```

```
201
202 Reaction{
203   'HO2 + H2O <=> OH + H2O2',
204   fr={'Arrhenius', A=0.28e14, n=0.0, C=16500.0},
205   label='r30'
206 }
207
208 Reaction{
209   'HO2 + HO2 <=> H2O2 + O2',
210   fr={'Arrhenius', A=0.2e13, n=0.0, C=0.0},
211   label='r31'
212 }
213
214 Reaction{
215   'O + O3 <=> O2 + O2',
216   fr={'Arrhenius', A=0.10e14, n=0.0, C=2411.0},
217   label='r32'
218 }
219
220 Reaction{
221   'O3 + NO <=> NO2 + O2',
222   fr={'Arrhenius', A=0.54e12, n=0.0, C=1200.0},
223   label='r33'
224 }
225
226 Reaction{
227   'O3 + H <=> OH + O2',
228   fr={'Arrhenius', A=0.70e14, n=0.0, C=560.0},
229   label='r34'
230 }
231
232 Reaction{
233   'O3 + OH <=> O2 + HO2',
234   fr={'Arrhenius', A=0.90e12, n=0.0, C=1000.0},
235   label='r35'
236 }
237
238 Reaction{
239   'O + N2 <=> NO + N',
240   fr={'Arrhenius', A=0.50e14, n=0.0, C=37940.0},
241   label='r36'
242 }
243
244 Reaction{
```

```
245   'H + NO <=> OH + N',
246   fr={'Arrhenius', A=0.17e15, n=0.0, C=24500.0},
247   label='r37'
248 }
249
250 Reaction{
251   'O + NO <=> O2 + N',
252   fr={'Arrhenius', A=0.15e10, n=1.0, C=19500.0},
253   label='r38'
254 }
255
256 Reaction{
257   'NO2 + H <=> NO + OH',
258   fr={'Arrhenius', A=0.35e15, n=0.0, C=740.0},
259   label='r39'
260 }
261
262 Reaction{
263   'NO2 + O <=> NO + O2',
264   fr={'Arrhenius', A=0.10e14, n=0.0, C=302.0},
265   label='r40'
266 }
267
268 Reaction{
269   'NO2 + H2 <=> HNO2 + H',
270   fr={'Arrhenius', A=0.24e14, n=0.0, C=14595.0},
271   label='r41'
272 }
273
274 Reaction{
275   'HO2 + NO <=> NO2 + OH',
276   fr={'Arrhenius', A=0.30e13, n=0.5, C=1208.0},
277   label='r42'
278 }
279
280 Reaction{
281   'NO2 + H2O <=> HNO2 + OH',
282   fr={'Arrhenius', A=0.32e13, n=0.0, C=22000.0},
283   label='r43'
284 }
285
286 Reaction{
287 ' NO2 + OH <=> HNO2 + O',
288   fr={'Arrhenius', A=0.21e13, n=0.0, C=12580.0},
```

```
289    label='r44'
290 }
291
292 r45 = 'CO + OH <=> CO2 + H'
293 r46 = 'CO2 + O <=> O2 + CO'
294 r47 = 'H2O + CO <=> HCO + OH'
295 r48 = 'OH + CO <=> HCO + O'
296 r49 = 'H2 + CO <=> HCO + H'
297 r50 = 'HO2 + CO <=> CO2 + OH'
298
299 Reaction{
300    'HNO + H <=> H2 + NO',
301    fr={'Arrhenius', A=0.48e13, n=0.0, C=0.0},
302    label='r51'
303 }
304
305 Reaction{
306    'HNO + OH <=> H2O + NO',
307    fr={'Arrhenius', A=0.36e14, n=0.0, C=0.0},
308    label='r52'
309 }
310
311 r53 = 'NO + CO <=> CO2 + N'
312 r54 = 'NO2 + CO <=> NO + CO2'
313
314 Reaction{
315    'NO + HO2 <=> HNO + O2',
316    fr={'Arrhenius', A=0.72e13, n=0.5, C=5500.0},
317    label='r55'
318 }
319
320 Reaction{
321    'HNO + O <=> NO + OH',
322    fr={'Arrhenius', A=0.5e12, n=0.5, C=0.0},
323    label='r56'
324 }
325
326 Reaction{
327    'HNO3 + O <=> HO2 + NO2',
328    fr={'Arrhenius', A=0.10e12, n=0.0, C=0.0},
329    label='r57'
330 }
331
332 Reaction{
```

```
333    'HO2 + NO2 <=> HNO2 + O2',
334    fr={'Arrhenius', A=0.20e12, n=0.0, C=0.0},
335    label='r58'
336 }
337
338 r59 = 'HCO + O2 <=> CO + HO2'
339
340 Reaction{
341    'O3 + HO2 <=> 2 O2 + OH',
342    fr={'Arrhenius', A=0.10e12, n=0.0, C=1409.0},
343    label='r60'
344 }
```