# HAMISH

## Adaptive Direct Numerical Simulation of Combustion

## User Guide

Stewart Cant

Department of Engineering, University of Cambridge
Trumpington Street, Cambridge CB2 1PZ
United Kingdom
email: rsc10@cam.ac.uk

December 2017

# 1 Introduction

Direct Numerical Simulation (DNS) of turbulent flow involves the solution of the Navier-Stokes equations without modelling of the turbulence. DNS of turbulent combustion also requires an adequate treatment of the chemistry and the molecular transport of heat and mass as well as momentum. Ideally the chemistry would be represented using a detailed chemical reaction mechanism including all of the elementary steps and intermediate species. Similarly, the many different effects that contribute to molecular transport would all be represented in detail. In practice, a full treatment of chemistry and molecular transport remains computationally unaffordable, and some level of modelling in these areas must be tolerated.

The purpose of the `HAMISH` code is to facilitate combustion DNS with any desired level of chemistry, from single-step Arrhenius mechanisms through all classes of reduced reaction mechanisms up to fully detailed reaction mechanisms. Molecular transport can also be represented using a Fickian diffusion laws or more complex treatments. The Navier-Stokes momentum equations are solved in fully compressible form together with the continuity equation and a conservation equation for the stagnation internal energy, as well as any required number of balance equations for species mass fractions. Each component of the reacting mixture is assumed to obey the equations of state for a semi-perfect gas. Boundary conditions are specified using a characteristic formulation, and available boundary condition types include inflows, outflows and walls as well as periodic conditions.

The numerical framework makes use of a finite-volume approach for spatial discretisation together with a Runge-Kutta algorithm for time-stepping. An adaptive Cartesian mesh based on cubic cells allows for local mesh refinement and derefinement based on the local and instantaneous resolution requirements of the solution. The spatial discretisation scheme works by reconstruction of the solution using a polynomial approximation in each cell. The Taylor–series order of the scheme is set by the degree of the polynomial, and arbitrarily high order is attainable in principle. In practice, fourth order offers a good compromise between accuracy and computational cost. Alternatively, simpler second–order interpolation between cell centres may be used instead, with fourth–order interpolation for convected variables wherever possible. For time-stepping, the Runge-Kutta algorithm may be adjusted to have any number of sub-steps in order to achieve the required order of accuracy. Adaptive time-stepping is available with error control using an embedded Runge-Kutta scheme. The standard time-stepping algorithm is a third–order three–step Runge-Kutta scheme with a second–order embedded scheme together with a PID–type time step controller. The code is fully parallel using domain decomposition over an arbitrary processor topology.

The chemical and thermodynamic data required for each simulation is handled in a compact tabular form in order to maximise computational efficiency during each simulation. A pre-processor called `PPCHEM` is available to convert data from a human-readable format into the required form for `HAMISH`. This approach is common to the high–order combustion DNS code `SENGA2` [1, 2].

# 2 Mathematical Formulation

## 2.1 Governing Equations

The governing equations to be solved using DNS are the partial differential equations for compressible reacting flow, consisting of the continuity equation

$$\frac{\partial}{\partial t}\rho + \frac{\partial}{\partial x_k}\rho u_k = 0, \tag{1}$$

the Navier–Stokes momentum equations

$$\frac{\partial}{\partial t}\rho u_i + \frac{\partial}{\partial x_k}\rho u_k u_i = -\frac{\partial}{\partial x_i}p + \frac{\partial}{\partial x_k}\tau_{ki}, \tag{2}$$

and the internal energy equation

$$\frac{\partial}{\partial t}\rho E + \frac{\partial}{\partial x_k}\rho u_k E = -\frac{\partial}{\partial x_k}p u_k - \frac{\partial}{\partial x_k}q_k + \frac{\partial}{\partial x_k}\tau_{km}u_m, \tag{3}$$

supplemented by an equation for the mass fraction of each of the $N$ chemical species present in the reacting gas mixture

$$\frac{\partial}{\partial t}\rho Y_\alpha + \frac{\partial}{\partial x_k}\rho u_k Y_\alpha = w_\alpha - \frac{\partial}{\partial x_k}\rho V_{\alpha,k} Y_\alpha. \tag{4}$$

The thermal equation of state for the mixture is

$$p = \rho R^0 T \sum_{\alpha=1}^{N} \frac{Y_\alpha}{W_\alpha} \tag{5}$$

while the caloric equation of state provides the definition of the stagnation internal energy as

$$E = \sum_{\alpha=1}^{N} Y_\alpha h_\alpha - \frac{P}{\rho} + \frac{1}{2} u_k u_k \tag{6}$$

where the enthalpy of species $\alpha$ is defined as

$$h_\alpha = \int_{T_0}^{T} C_{p\alpha} dT + h_\alpha^0, \tag{7}$$

in which $C_{p\alpha}$ is the mass–based specific heat capacity of species $\alpha$ and $h_\alpha^0$ is the species enthalpy at the reference temperature $T_0$. The viscous stress tensor is given by

$$\tau_{ki} = \mu \left( \frac{\partial u_k}{\partial x_i} + \frac{\partial u_i}{\partial x_k} \right) - \frac{2}{3}\mu \frac{\partial u_m}{\partial x_m} \delta_{ki}, \tag{8}$$

and the heat flux vector by

$$q_k = -\lambda \frac{\partial T}{\partial x_k} + \sum_{\alpha=1}^{N} \rho V_{\alpha,k} Y_\alpha h_\alpha \tag{9}$$

The species mass fractions are subject to the compatibility condition

$$\sum_{\alpha=1}^{N} Y_\alpha = 1 \tag{10}$$

while the diffusion velocities must satisfy the compatibility condition

$$\sum_{\alpha=1}^{N} V_{\alpha,k} Y_\alpha = 0. \tag{11}$$

For a reaction mechanism involving $M$ steps of the form

$$\sum_{\alpha=1}^{N} \nu'_{\alpha,m} \mathcal{M}_\alpha \rightarrow \sum_{\alpha=1}^{N} \nu''_{\alpha,m} \mathcal{M}_\alpha, \quad m = 1, \ldots, M \tag{12}$$

the chemical production rate $w_\alpha$ for species $\alpha$ is expressed as

$$w_\alpha = W_\alpha \sum_{m=1}^{M} \left( \nu''_{\alpha,m} - \nu'_{\alpha,m} \right) k_m(T) \prod_{\beta=1}^{N} c_\beta^{\nu'_{\beta,m}}. \tag{13}$$

where the specific reaction rate coefficient $k_m(T)$ is given by the Arrhenius expression

$$k_m(T) = A_m T^{n_m} \exp\left( -\frac{E_m}{R^0 T} \right) \tag{14}$$

and the concentration $c_\beta$ of species $\beta$ is related to mass fraction by

$$c_\beta = \frac{\rho Y_\beta}{W_\beta}. \tag{15}$$

For the reaction rates the compatibility condition is

$$\sum_{\alpha=1}^{N} w_\alpha = 0. \tag{16}$$

## 2.2 Thermodynamic Quantities

For a semi–perfect gas, the molar specific heat capacity at constant pressure $\bar{C}_{p\alpha}$ is known to depend on temperature, and for present purposes the dependence is represented in an approximate manner using a polynomial of the form

$$\frac{\bar{C}_{p\alpha}}{R^0} = \sum_{j=1}^{J} \bar{a}_{\alpha,j}^{(l)} T^{j-1} \tag{17}$$

where the degree $J-1$ of the polynomial is typically taken to be 4 or 5. The polynomial coefficients $a_{\alpha,j}^{(l)}$ may take different values for each species $\alpha$ in different intervals $l$ of temperature. For example, the popular CHEMKIN database [3] uses polynomials of degree 5 with two temperature intervals: for most species the intervals are $0 < T < 1000\text{K}$ ($l = 1$) and $1000\text{K} \le T < 3000\text{K}$ ($l = 2$). The molar enthalpy for each species is given by integrating over $L$ successive temperature intervals $l = 1, \ldots, L$ to yield the recursive formula

$$\begin{aligned} \bar{h}_\alpha &= R^0 \int_{T_{L-1}}^{T} \sum_{j=1}^{J} \bar{a}_{\alpha,j}^{(L)} T^{j-1} dT + R^0 \sum_{l=1}^{L-1} \left[ \int_{T_{l-1}}^{T_l} \sum_{j=1}^{J} \bar{a}_{\alpha,j}^{(l)} T^{j-1} dT \right] + \bar{h}_\alpha^0 \\ &= R^0 \sum_{j=1}^{J} \frac{\bar{a}_{\alpha,j}^{(L)}}{j} T^j - R^0 \sum_{j=1}^{J} \frac{\bar{a}_{\alpha,j}^{(L)}}{j} T_{L-1}^j + \bar{h}_\alpha^{(L-1)}. \end{aligned} \tag{18}$$

where the reference temperature $T_0$ has been taken at 0K, and $h_\alpha^{(L-1)}$ denotes the enthalpy at the lower end of the current ($L$th) temperature interval. The last two terms may be combined to form a single coefficient $\bar{a}_{\alpha,J+1}^{(L)}$, and the final form for the molar enthalpy becomes

$$\bar{h}_\alpha = R^0 \left[ \sum_{j=1}^{J} \frac{\bar{a}_{\alpha,j}^{(L)}}{j} T^j + \bar{a}_{\alpha,J+1}^{(L)} \right] \tag{19}$$

Similarly, the molar entropy for each species may be found using the definition

$$\bar{s}_\alpha = R^0 \int_{T_0}^{T} \frac{\bar{C}_{p\alpha}}{T} dT + \bar{s}_\alpha^0, \tag{20}$$

substituting for $\bar{C}_{p\alpha}$ and integrating over successive temperature intervals to give

$$\begin{aligned} \bar{s}_\alpha &= R^0 \left[ \bar{a}_{\alpha,1}^{(L)} \ln T + \sum_{j=2}^{J} \frac{\bar{a}_{\alpha,j}^{(L)}}{j-1} T^{j-1} \right] \\ &- R^0 \left[ \bar{a}_{\alpha,1}^{(L)} \ln T_{L-1} + \sum_{j=2}^{J} \frac{\bar{a}_{\alpha,j}^{(L)}}{j-1} T_{L-1}^{j-1} \right] + \bar{s}_\alpha^{(L-1)} \end{aligned} \tag{21}$$

where once again the formula may be applied recursively, and $s_\alpha^{(L-1)}$ is the entropy at the lower end of the current temperature interval. Combining the last three terms to form a single coefficient $\bar{a}_{\alpha,J+2}^{(L)}$, the final form for the molar entropy is

$$\bar{s}_\alpha = R^0 \left[ \bar{a}_{\alpha,1}^{(L)} \ln T + \sum_{j=2}^{J} \frac{\bar{a}_{\alpha,j}^{(L)}}{j-1} T^{j-1} + \bar{a}_{\alpha,J+2}^{(L)} \right] \tag{22}$$

The mass-based specific heat capacity $C_{p\alpha}$ is given by

$$C_{p\alpha} = \frac{R^0}{W_\alpha} \sum_{j=1}^{J} \bar{a}_{\alpha,j}^{(l)} T^{j-1} = \sum_{j=1}^{J} a_{\alpha,j}^{(l)} T^{j-1} \tag{23}$$

where the mass-based polynomial coefficients are defined as

$$a_{\alpha,j}^{(l)} = \frac{R^0}{W_\alpha} \bar{a}_{\alpha,j}^{(l)}. \tag{24}$$

Then the mass-based specific enthalpy for each species is

$$h_\alpha = \sum_{j=1}^{J} \frac{a_{\alpha,j}^{(L)}}{j} T^j + a_{\alpha,J+1}^{(L)} \tag{25}$$

and the mass-based specific entropy for each species is

$$s_\alpha = a_{\alpha,1}^{(L)} \ln T + \sum_{j=2}^{J} \frac{a_{\alpha,j}^{(L)}}{j-1} T^{j-1} + a_{\alpha,J+2}^{(L)}. \tag{26}$$

### 2.2.1 Temperature

The temperature may be obtained using the caloric equation of state (6) in the form

$$E = \sum_{\alpha=1}^{N} Y_\alpha h_\alpha - R_m T + \frac{1}{2} u_k u_k \tag{27}$$

where the specific gas constant for the mixture is

$$R_m = \sum_{\alpha=1}^{N} Y_\alpha R_\alpha. \tag{28}$$

Substituting the polynomial form of the species enthalpy $h_\alpha$ from (25) gives

$$E = \sum_{\alpha=1}^{N} Y_\alpha \left[ \sum_{j=1}^{J} \frac{a_{\alpha,j}^{(l)}}{j} T^j + a_{\alpha,J+1}^{(l)} \right] - R_m T + \frac{1}{2} u_k u_k \tag{29}$$

and gathering terms in successive powers of $T$ produces the polynomial

$$\begin{aligned} f(T) &= \left[ \left( \frac{1}{2} u_k u_k - E \right) + \sum_{\alpha=1}^{N} Y_\alpha a_{\alpha,J+1}^{(l)} \right] \\ &+ \left[ \sum_{\alpha=1}^{N} Y_\alpha \left( a_{\alpha,1}^{(l)} - R_\alpha \right) \right] T + \sum_{j=2}^{J} \left[ \sum_{\alpha=1}^{N} Y_\alpha \frac{a_{\alpha,j}^{(l)}}{j} \right] T^j \end{aligned} \tag{30}$$

which may be used to form the non-linear algebraic equation $f(T) = 0$ whose solution determines the temperature.

## 2.3 Transport Coefficients

The molecular transport coefficients are represented using the relationship

$$\frac{\lambda}{C_p} = A_\lambda \left(\frac{T}{T_0}\right)^r \tag{31}$$

for the mixture thermal conductivity $\lambda$, where $C_p$ is the mixture value of the specific heat capacity at constant pressure, and $A_\lambda$, $r$ and $T_0$ are constants. Then the mixture dynamic viscosity $\mu$ is given by

$$\mu = \frac{\lambda}{C_p}\mathrm{Pr} \tag{32}$$

where Pr is the mixture Prandtl number which is taken as a constant.

The diffusive mass flux for species $\alpha$ may be represented using Fick's law as:

$$\rho V_{\alpha,k} Y_\alpha = -\rho D_\alpha \frac{\partial Y_\alpha}{\partial x_k}. \tag{33}$$

in which the diffusion coefficient $D_\alpha$ for each species is given by

$$D_\alpha = \frac{\lambda}{\rho C_p Le_\alpha} \tag{34}$$

where $Le_\alpha$ is the Lewis number. The standard approach assumes that $Le_\alpha$ is constant for each species.

Applying this assumption in Fick's Law does not guarantee that the continuity equation will be recovered when all $N$ of the species mass fraction equations are summed. Substituting (33) into (4) and summing over all species yields

$$\frac{\partial}{\partial t}\rho + \frac{\partial}{\partial x_k}\rho u_k = \sum_{\alpha=1}^{N} \frac{\partial}{\partial x_k}\rho D_\alpha \frac{\partial Y_\alpha}{\partial x_k}. \tag{35}$$

where the compatibility conditions (10) and (16) have been applied to the mass fractions and reaction rates respectively. Clearly, by comparison with the continuity equation (1) the quantity on the right-hand side is an error term. This term can be removed by making a slight modification to Fick's law [4]:

$$\rho V_{\alpha,k} Y_\alpha = -\rho D_\alpha \frac{\partial Y_\alpha}{\partial x_k} + \rho V_k^{(c)} Y_\alpha \tag{36}$$

where the correction velocity $V_k^{(c)}$ is given by

$$\rho V_k^{(c)} = \sum_{\alpha=1}^{N} \rho D_\alpha \frac{\partial Y_\alpha}{\partial x_k}. \tag{37}$$

Applying the compatibility condition (11), the modified form (36) ensures that the continuity equation (1) is recovered as required.

## 2.4 Reaction Rate

Evaluation of the chemical production rate $w_\alpha$ for species mass fraction makes use of the general formulation expressed by (13)-(15). There is a summation over all steps in the reaction mechanism

$$w_\alpha = W_\alpha \sum_{m=1}^{M} \bar{w}_{\alpha,m} \tag{38}$$

where $\bar{w}_{\alpha,m}$ is the molar production rate of species $\alpha$ in the step. Each step in the reaction mechanism may involve one of several possible special cases.

### 2.4.1 Forward Reaction Rate

For a forward reaction step as expressed by

$$\sum_{\alpha=1}^{N} \nu'_{\alpha,m} \mathcal{M}_\alpha \rightarrow \sum_{\alpha=1}^{N} \nu''_{\alpha,m} \mathcal{M}_\alpha \tag{39}$$

the molar production rate $\bar{w}_{\alpha,m}$ is given by

$$\bar{w}_{\alpha,m} = \left(\nu''_{\alpha,m} - \nu'_{\alpha,m}\right) k_m(T) \prod_{\beta=1}^{N} c_\beta^{\nu'_{\beta,m}}. \tag{40}$$

with $k_m(T)$ given by (14) and $c_\beta$ by (15). This is the simplest and most common type of reaction step.

### 2.4.2 Backward Reaction Rate: Gibbs Function

In cases where a reaction step $m$ is specified using the equilibrium notation

$$\sum_{\alpha=1}^{N} \nu'_{\alpha,m} \mathcal{M}_\alpha \rightleftharpoons \sum_{\alpha=1}^{N} \nu''_{\alpha,m} \mathcal{M}_\alpha \tag{41}$$

the molar production rate for a single species is given by

$$\bar{w}_{\alpha,m} = \left(\nu''_{\alpha,m} - \nu'_{\alpha,m}\right) \left[ k_{f,m}(T) \prod_{\beta=1}^{N} c_\beta^{\nu'_{\beta,m}} - k_{b,m}(T) \prod_{\beta=1}^{N} c_\beta^{\nu''_{\beta,m}} \right]. \tag{42}$$

Often, only the data for the forward rate coefficient $k_{f,m}(T)$ are supplied, and it becomes necessary to evaluate the backward rate coefficient $k_{b,m}(T)$ using the equilibrium constant for concentrations

$$K_{c,m} = \prod_{\alpha=1}^{N} c_\alpha^{(\nu''_{\alpha,m} - \nu'_{\alpha,m})} = \frac{k_{b,m}}{k_{f,m}} \tag{43}$$

The relation between $K_{c,m}$ and the equilibrium constant for partial pressures $K^0_{p,m}$ is given by

$$K_{c,m} = K^0_{p,m} \left(\frac{p_0}{R^0 T}\right)^{\Delta\nu_m} \tag{44}$$

where $p_0$ is a reference pressure used to make $K^0_{p,m}$ dimensionless, and $\Delta\nu_m = \sum_{\alpha=1}^{N} (\nu''_{\alpha,m} - \nu'_{\alpha,m})$ is the difference in the total number of moles between reactants and products. In turn, $K^0_{p,m}$ is related to the change in the molar Gibbs function according to

$$R^0 T \ln K^0_{p,m} = \Delta \hat{G}_m = \sum_{\alpha=1}^{N} \bar{g}_\alpha (\nu''_{\alpha,m} - \nu'_{\alpha,m}) \tag{45}$$

where $\bar{g}_\alpha = \bar{h}_\alpha - T\bar{s}_\alpha$ is the molar Gibbs function for species $\alpha$. Using (19) and (22), $\bar{g}_\alpha$ can be expressed conveniently as

$$\frac{\bar{g}_\alpha}{R^0 T} = \frac{\bar{a}^{(L)}_{\alpha,J+1}}{T} - \bar{a}^{(L)}_{\alpha,1} \ln T + (\bar{a}^{(L)}_{\alpha,1} - \bar{a}^{(L)}_{\alpha,J+2}) - \sum_{j=2}^{J} \frac{\bar{a}^{(L)}_{\alpha,j}}{j(j-1)} T^{j-1} \tag{46}$$

Writing $k_{f,m}$ and $k_{b,m}$ in Arrhenius form according to (14), and using (43) together with (44) and (46) yields a set of expressions for the parameters of the backwards reaction rate coefficient

$$\begin{aligned}
\ln A_{b,m} &= \ln A_{f,m} \\
&+ \sum_{\alpha=1}^{N} (\nu''_{\alpha,m} - \nu'_{\alpha,m}) \left[ \bar{a}^{(L)}_{\alpha,1} - \bar{a}^{(L)}_{\alpha,J+2} + \ln\left(\frac{P_0}{R^0}\right) \right] \\
&- \ln A^{\Sigma}_{b,m}(T) \qquad (47) \\
n_{b,m} &= n_{f,m} - \sum_{\alpha=1}^{N} (\nu''_{\alpha,m} - \nu'_{\alpha,m}) \left[ \bar{a}^{L}_{\alpha,1} + 1 \right] \qquad (48) \\
\frac{E_{b,m}}{R^0} &= \frac{E_{f,m}}{R^0} + \sum_{\alpha=1}^{N} (\nu''_{\alpha,m} - \nu'_{\alpha,m}) \left[ \bar{a}^{L}_{\alpha,J+1} \right] \qquad (49)
\end{aligned}$$

Note that there is a non-Arrhenius contribution to (47) denoted by $\ln A^{\Sigma}_{b,m}(T)$. This term arises from the temperature dependence of $C_{p\alpha}$, and is given by

$$\ln A^{\Sigma}_{b,m}(T) = \sum_{\alpha=1}^{N} (\nu''_{\alpha,m} - \nu'_{\alpha,m}) \sum_{j=2}^{J} \frac{\bar{a}^{(L)}_{\alpha,j} T^j}{j(j+1)}. \qquad (50)$$

Due to this term the backward rate coefficient $k_{b,m}$ has a temperature dependence over and above that indicated by the Arrhenius form. This means that the contribution $\ln A^{\Sigma}_{b,m}(T)$ cannot be computed in advance and hence $k_{b,m}$ must be evaluated "on the fly" during a simulation. This may be done by combining (43)-(46) to yield

$$\ln k_{b,m} = \ln k_{f,m} + \sum_{\alpha=1}^{N} (\nu''_{\alpha,m} - \nu'_{\alpha,m}) \left( \frac{\bar{g}_\alpha}{R^0 T} + \ln \frac{p_0}{R^0} - \ln T \right) \qquad (51)$$

### 2.4.3 Third Bodies

In cases where a reaction step $m$ is of the form

$$\sum_{\alpha=1}^{N} \nu'_{\alpha,m} \mathcal{M}_\alpha + M \rightarrow \sum_{\alpha=1}^{N} \nu''_{\alpha,m} \mathcal{M}_\alpha + M \qquad (52)$$

the molar production rate for a species $\alpha$ is given by

$$\bar{w}_{\alpha,m} = \left( \nu''_{\alpha,m} - \nu'_{\alpha,m} \right) k_m(T) c_M \prod_{\beta=1}^{N} c_\beta^{\nu'_{\beta,m}}. \qquad (53)$$

where the concentration $c_M$ of the "third body" $M$ is given by

$$c_M = \sum_{\alpha=1}^{N} \eta_{\alpha,M} c_\alpha \qquad (54)$$

in which the coefficients $\eta_{\alpha,M}$ are the third-body efficiencies for $M$.

### 2.4.4 Lindemann Forms

Some three–body reaction steps may have a specific reaction rate coefficient $k$ which depends on pressure as well as temperature. The simplest representation is the Lindemann form [5]:

$$k_{L,m} = k_{\infty,m} \left( \frac{P_r}{1 + P_r} \right) F_m \tag{55}$$

where $k_{\infty,m}$ is the rate coefficient in the limit of high pressure and $F_m$ is a constant normally taken equal to unity although other values may be specified instead. The quantity $P_r$ is a "reduced pressure" defined as

$$P_r = \frac{k_{0,m}}{k_{\infty,m}} c_M \tag{56}$$

where $k_{0,m}$ is the rate coefficient in the limit of low pressure and $c_M$ is the third–body concentration as defined in (54). Both $k_{\infty,m}$ and $k_{0,m}$ have the standard Arrhenius form (14). A Lindemann reaction step requires the specification of seven values, i.e. the three parameters $A$, $n$ and $E$ for each of $k_{\infty,m}$ and $k_{0,m}$, together with the value of $F_m$.

### 2.4.5 Troe Forms

For some pressure–dependent three–body reaction steps the Lindemann form is found to be insufficient and the Troe form is preferred [6]. The Lindemann representation (55) is retained with the reduced pressure $P_r$ defined as in (56), but now $F_m$ is specified as a function

$$\log F_m = \left[ 1 + \left( \frac{\log P_r + c}{n - d(\log P_r + c)} \right)^2 \right]^{-1} \log F_{\text{cent}} \tag{57}$$

The quantities $c$ and $n$ are defined as:

$$c = c_1 - c_2 \log F_{\text{cent}}; \qquad n = n_1 - n_2 \log F_{\text{cent}} \tag{58}$$

where the standard values are $c_1 = -0.4$, $c_2 = 0.67$, $n_1 = -0.75$, $n_2 = 1.27$ and $d = 0.14$. The function $F_{\text{cent}}$ is given by

$$F_{\text{cent}} = (1 - \alpha) \exp\left( -\frac{T}{T^{***}} \right) + \alpha \exp\left( \frac{T}{T^*} \right) + \exp\left( -\frac{T^{**}}{T} \right). \tag{59}$$

A total of fifteen values must be specified for a Troe step, consisting of the three parameters $A$, $n$ and $E$ for each of $k_{\infty,m}$ and $k_{0,m}$, the quantities $\alpha$, $T^*$, $T^{**}$ and $T^{***}$, and the constants $c_1$, $c_2$, $n_1$, $n_2$ and $d$.

### 2.4.6 SRI Forms

A third form for pressure–dependent three–body reactions is due to SRI International [7]. Again the Lindemann representation defined by (55) and (56) is retained, but now the function $F_m$ is defined as

$$F_m = d \left[ a \exp\left( -\frac{b}{T} \right) + \exp\left( -\frac{T}{c} \right) \right]^X T^e \tag{60}$$

where $d$ and $e$ are normally set to unity, and $X$ is given by

$$X = \frac{1}{1 + \log^2 P_r}. \tag{61}$$

Each SRI step requires eleven values, which are the three parameters $A$, $n$ and $E$ for each of $k_{\infty,m}$ and $k_{0,m}$, together with the five constants $a$, $b$, $c$, $d$ and $e$.

## 2.5 Boundary Conditions

The boundary conditions are specified according to a linearised characteristic formulation [8] that is compatible with the finite–volume discretisation scheme. The boundaries of the computational domain correspond to cell faces in the finite–volume scheme. Hence the boundary conditions are manifested in cell face fluxes rather than acting directly on the solution values. The characteristic variables across a boundary with normal in the $i$–direction are defined as

$$\begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \\ L_5 \\ L_{5+\alpha} \end{bmatrix} = \begin{bmatrix} p - \rho_0 a_0 u_i \\ p - a_0^2 \rho \\ u_{j1} \\ u_{j2} \\ p + \rho_0 a_0 u_i \\ Y_\alpha \end{bmatrix} \tag{62}$$

where $j \neq i$ denotes the transverse directions (two components in 3D) and where $\rho_0$ and $a_0$ are locally constant base values of density and sound speed. Here, the ordering of the characteristic variables is compatible with the Navier–Stokes Characteristic Boundary Condition (NSCBC) formulation commonly used in DNS [9, 10].

Boundary conditions on the set of primitive variables $\{\rho, u_i, u_{j1}, u_{j2}, P, Y_\alpha\}$ are derived by manipulation of the characteristic variables on the boundary. For example, the pressure is found by adding $L_1$ and $L_5$, while the boundary–normal velocity is found by subtracting $L_1$ from $L_5$.

It should be noted that it may be necessary also to impose boundary conditions on the viscous, thermal conduction and diffusive flux terms in order to meet the theoretical requirements for the Navier–Stokes equations [10]. This is done in an *ad-hoc* manner as indicated below.

### 2.5.1 Outflow Boundary Conditions

**Subsonic Non–reflecting Outflow**

An acoustically non–reflecting outflow boundary condition may be imposed by allowing outgoing acoustic waves to leave the domain while setting the amplitude of incoming acoustic waves to zero. On a right–hand boundary the outgoing wave amplitude is $L_5$ and the incoming wave amplitude is $L_1$. A non–reflecting outflow boundary condition is then given by

$$\begin{aligned} L_1 &= 0 \\ L_5 &= p_x + \rho_0 a_0 u_{i,x} \end{aligned} \tag{63}$$

where the subscript $x$ indicates that the value is extrapolated from the interior side of the boundary. Often in practice it is desirable to impose a partially–reflecting boundary condition in order to allow the pressure at the boundary to track some required value $p_\infty$. This is done using the prescription

$$\begin{aligned} L_1 &= C_\alpha \left(p - p_\infty\right) \\ L_5 &= p_x + \rho_0 a_0 u_{i,x} \end{aligned} \tag{64}$$

where $C_\alpha$ is a relaxation constant. On a left–hand boundary the prescription is

$$\begin{aligned} L_1 &= p_x - \rho_0 a_0 u_{i,x} \\ L_5 &= C_\alpha \left(p - p_\infty\right) \end{aligned} \tag{65}$$

All of the other characteristic variables are evaluated using extrapolated values. Then the primitive

variables on the boundary are given by

$$\begin{bmatrix} \rho \\ u_i \\ u_{j1} \\ u_{j2} \\ p \\ Y_\alpha \end{bmatrix} = \begin{bmatrix} \frac{1}{a_0^2}\left[\frac{1}{2}\left(L_1+L_5\right)-L_2\right] \\ \frac{1}{2\rho_0 a_0}\left(L_5-L_1\right) \\ L_3 \\ L_4 \\ \frac{1}{2}\left(L_1+L_5\right) \\ Y_\alpha = L_{5+\alpha} \end{bmatrix} \tag{66}$$

If required, the diffusive flux on the boundary may be modified by imposing conditions such as

$$\tau_{in} = 0; \quad q_n = 0; \quad \rho V_{\alpha,n} Y_\alpha = 0 \tag{67}$$

on the normal components of the viscous stress tensor, the heat flux vector and the diffusive flux vector.

**Subsonic Reflecting Outflow**
This boundary condition is imposed by setting $p = p_\infty$ where $p_\infty$ is the required value of pressure on the boundary. All other primitive variables retain their extrapolated values.

### 2.5.2 Inflow Boundary Conditions

**Subsonic Non–reflecting Inflow**
This boundary condition is not yet implemented.

**Subsonic Reflecting Inflow With Specified Temperature**
This condition is set by imposing the values of temperature, all three velocity components and all species mass fractions at the boundary. If required, any or all of these variables may be specified as functions of time, and hence this condition does allow in principle for a turbulent inflow condition. The value of the density is not imposed and instead is extrapolated from the interior side of the boundary.

**Subsonic Reflecting Inflow With Specified Density**
This condition is set by imposing the values of density, all three velocity components and all species mass fractions at the boundary. All of these variables may be constant or they may be specified as functions of time. The pressure retains its extrapolated value.

### 2.5.3 Wall Boundary Conditions

A wall boundary is treated using an impermeability condition and a no–slip condition, with all velocity components set to zero on the wall.

**Isothermal Wall**
For an isothermal wall there is an additional condition on the temperature:

$$T = T_{\text{wall}} \tag{68}$$

where $T_{\text{wall}}$ is a prescribed constant value of the temperature at the wall.

**Adiabatic Wall**
For an adabatic wall the temperature is not prescribed and instead the wall-normal component of the heat flux vector is set to zero:

$$q_n = 0. \tag{69}$$

## 2.6 Initial Conditions

Initial conditions are required for all of the conserved variables. Constant and spatially–uniform values are specified by default at start–up. This may not be sufficient and more realistic initial velocity and acalar fields may be provided using subroutines within the code.

### 2.6.1 Turbulence Initial Conditions

Initial conditions for turbulence are not yet implemented.

### 2.6.2 Thermochemical Initial Conditions

As well as the velocity field it is necessary also to specify the scalar field. This requires two thermodynamic variables (e.g., $P$ and $T$), together with a set of composition variables (e.g., $Y_\alpha$), to be specified at every point in the domain. This is done most conveniently by means of a subroutine named `FLAMIN` which is called during start–up, and which provides the flexibility to specify any required problem configuration by coding it in `FORTRAN`.

A common example of an initial thermochemical field is a laminar premixed flame solution. A straightforward approach to the initialisation of a one–dimensional laminar premixed flame is to use an error–function profile for an arbitrary reaction progress variable $c$:

$$c(x;t=0) = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{x - x_0}{\delta}\right)\right] \tag{70}$$

where $x$ is the coordinate normal to the flame, $x_0$ is the location of the centre of the profile and $\delta$ is the thickness. The mass fractions of the major species may be computed directly from the progress variable using:

$$Y_\alpha(x;t=0) = Y_{\alpha,R} + c(x)\,(Y_{\alpha,P} - Y_{\alpha,R}) \tag{71}$$

where the subscripts R and P denote the limiting values in the reactants and products respectively. The initial temperature may be specified using the same approach, and the initial density follows under the assumption of constant thermochemical pressure. The initial velocity may be computed by assuming constant mass flux through the flame. If required, initial profiles of minor species may be specified using a Gaussian function. The procedure as outlined is often sufficient to allow a laminar flame solution to develop fairly rapidly, even when employing a detailed chemical reaction mechanism. The resulting one–dimensional solution may be used subsequently to initialise a three–dimensional turbulent flame simulation.

# 3 Spatial Discretisation

## 3.1 General

Spatial discretisation is handled using a distributed octree approach [11] which is designed to scale to arbitrarily large meshes. The spatial mesh consists entirely of cubic cells. The mesh is adaptive and may be refined locally by splitting any cell of side $2h$ into eight identical cubic sub–cells each of side $h$. Alternatively, the mesh may be derefined locally by grouping eight adjacent cells each of side $h$ into a single cubic cell of side $2h$. Indexing of cells within the mesh is handled using Morton coding [12], which provides a unique mapping from three–dimensional physical space onto a one–dimensional computational space. Morton codes are stored globally in ascending order, and there is exactly one Morton code for every cell in the mesh. Thus the list of Morton codes is not necessarily contiguous. Searching for individual cells within the mesh is carried out using an octree data structure. The mesh is distributed in parallel across the available processors by assigning a unique range of Morton codes - and hence cells - to each processor. The corresponding global octree is also distributed across the processors. Parallel load–balancing is achieved by redistributing Morton codes (and the octree) between the processors as required.

## 3.2 A Cell

A cell is a cubic subdivison of physical space of side $h$ and volume $h^3$, and also corresponds logically to a data structure which stores all of the associated physical and numerical properties. This includes the

set of conserved variables $\{\rho, \rho u, \rho v, \rho w, \rho E, \rho Y_\alpha\}$ stored as cell–average values, and the set of cell indices describing the local stencil.

### 3.3 Morton coding

Each cell in the mesh is identified globally by a unique binary code according to the scheme due to Morton [12]. The Morton code is made up of a string of octal numbers together with a single integer which specifies the level of the cell within the mesh. An example is given in Fig.n, in which the Morton code shown contains 18 octal numbers and a 6-bit level code $L$. The value of $L$ indicates how many octal numbers are required to identify the cell, counting from the left–hand end of the Morton code as shown.

| 010 001 001 011 010 000 010 001 001 011 010 000 001 001 011 010 000 000 | 000100 |
|---|---|

Table 1: Example of a Morton code

The Morton code has several useful properties. Firstly, it provides a convenient encoding of the integer coordinates of each cell in the $x$-$y$-$z$ physical space. To obtain the cell coordinates from the Morton code, the first binary digit of each octal number from the left–hand end up to $L$ is extracted in turn to form a bit string. This procedure is repeated with the second digit of each octal number, and again with the third digit. The three bit strings, each taken with the highest–valued bit on the left, form integers which correspond respectively to the $x$, $y$ and $z$ coordinates of the cell. The coordinates are expressed in units of the cell side $h$ at the current level $L$, and specify the location of the bottom–left–lowest corner of the cell, relative to an origin at the bottom–left–lowest corner of the entire computational domain. The procedure can be reversed in a straightforward manner to construct the cell Morton code from the cell coordinates.

Secondly, the Morton codes form a sequence defining a path in physical space through all of the cells in the mesh, such that all cells are visited exactly once. The order in which the cells are visited is called Morton ordering or Z-ordering. In each group of eight cells, each cell is visited starting with the cell having the lowest $x$-$y$-$z$ coordinates, moving first in the $x$-direction, then in the $y$-direction and then in the $z$-direction, returning to the lowest value of the previous coordinates each time the direction is changed. This procedure essentially defines a space–filling curve, and allows all cells to be indexed in one dimension irrespective of their size or location in physical space.

Finally, the octal numbers in the Morton code taken from the left–hand end up to the level $L$ may be interpreted as a set of instructions giving the path through the octree from the root to the specified cell. This allows for rapid searching of the octree, for example to find the index of a cell whose integer coordinates are known. It means also that the octree and the list of Morton codes are effectively interchangeable, and one may be generated from the other.

In the `HAMISH` code, the Morton codes are made up as shown in Table 1, with each Morton code containing 18 octal numbers together with a level code of six binary digits. Each Morton code is stored in an ordered pair of 32–bit signed integers, with 10 octal numbers stored in the lower integer and the remaining 8 octal numbers plus the level code stored in the upper integer.

### 3.4 The Octree

The octree data structure provides a natural means of managing the three–dimensional spatial mesh. The root of the tree corresponds to an all-encompassing Big Cube which is assigned the level 0. In practice the Big Cube is taken to be just large enough to span the largest dimension of the computational domain. Then level 1 corresponds to the eight smaller cubes which are obtained by equal subdivision of the Big Cube. Further levels follow in the obvious manner, with each new subdivision creating exactly eight new cubes and exactly eight new branches of the tree. In two spatial dimensions a quadtree structure is sufficient to manage the four branches that arise with each subdivision, and in one dimension a binary tree can be used. Every branch of the tree starts either at the root or at a branchpoint, and it may end

either at another branchpoint or at a leaf. A branch ending in a leaf is called a twig, and each leaf is associated uniquely with a single cell of the spatial mesh.

The octree is distributed in parallel over all the processors. Every processor has its own copy of the root, together with just enough of the global octree to connect it with its own portion of the mesh. Branches which lead to parts of the octree which are located on a different processor are flagged as remote and are not handled locally.

The octree is stored as a set of branchpoints, each of which contains the address of its parent together with eight branch descriptors, Each branch descriptor contains an attribute and an address. The branch attributes are:

- "U" for Unset. The branch has been created but does not yet point to anything. The address is null.
- "N" for a Normal branchpoint. The address is the index of the next branchpoint in the octree.
- "O" for a branch that leads Out of the computational domain. The address is null.
- "P" for a branch that leads to a remote Parallel processor. The address is null.
- "T" for a Twig. The address is the index of a cell.

The logical structure of a branchpoint in the octree is shown in Table 2.

| N | N | N | N | N | N | N | N |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Table 2: Structure of a branchpoint in the octree

The octree is implemented in the `HAMISH` code as a set of three arrays whose common index is a unique identifier for a branchpoint. There is a one–dimensional integer array containing the parent index, a two–dimensional integer array containing the indices of the eight children, and a two–dimensional character array containing the attributes of the eight children.

## 3.5 Partition Interval Table

The only global information held by every processor is the Partition Interval Table (PIT), which contains the highest Morton code stored on each processor. The PIT acts as a global address space for parallel data transfers, and is updated dynamically during parallel load–balancing operations.

## 3.6 Adaptive Mesh Refinement

Each cell in the mesh may be refined according to any suitable set of criteria which may be based (for example) on some measure of the local length scale of the solution. Refinement is effected by subdividing each cubic cell into eight new cubic cells each with side $h$ half that of the parent cell. The opposite operation of derefinement is effected when all eight children of a single parent cell are found to satisfy a suitable set of derefinement criteria. The eight child cells are eliminated and replaced by a single cell of twice the original side length $h$.

Evaluation of the refinement and derefinement criteria produces a list of cells which are candidates for refinement and a separate list of cells which are candidates for derefinement. On entry to the adaptive mesh refinement procedure, the only physical data stored in each cell is the vector of conserved variables.

Derefinement is handled first in order to reduce peaks in memory usage. Cells marked for derefinement are checked to ensure that they form groups of eight (octets) that are all children of a single parent cell. Cells not meeting this requirement are removed from the derefinement list. For the remaining cells in the list, derefinement is enacted by first reducing the level of the lowest Morton code in the octant by one.

The corresponding cell is marked as a derefined parent cell while the others in the octant are marked as derefined child cells. When all cells in the list have been marked, garbage collection is carried out removing the redundant child–cell Morton codes and closing up the Morton code list. At this stage, no cell data is moved.

For each cell in the refinement list, refinement is enacted by first increasing the level of the cell Morton code to create the lowest member of a new octet. Seven new Morton codes are created for the remaining members of the octet and these are appended initially to the end of the Morton code list. When all cells in the refinement list have been refined, the expanded Morton code list is sorted into ascending order. This automatically gathers together the eight members of each new octet and moves them to the right place in the Morton code list. Again, at this stage no cell data is moved.

### 3.7 Tree balancing

Refinement and derefinement both give rise to new transitions in cell size between adjacent cells. In order to minimise the number of special cases required by the spatial discretisation scheme, no transition of more than one level of refinement is allowed. Hence $h$–$2h$ size transitions are permitted, but not transitions of $h$–$4h$ or greater. This is achieved by "balancing the tree" after each refinement or derefinement operation. Tree balancing involves searching for the six geometrical face–neighbours of each cell in the mesh. This is done by extracting the integer coordinates from the cell Morton code. For each face the relevant $x$, $y$ or $z$ integer coordinate is incremented or decremented by one to find the integer coordinates of the neighbouring cell. Assuming that the cell level does not change, a guessed Morton code is constructed for the neighbouring cell, and the octree is searched for that Morton code. The search returns the actual neighbour–cell Morton code. Remote cells are dealt with by an exchange of Morton codes between processors. The level of the actual neighbour–cell Morton code is checked against that of the original cell. If the neighbouring cell is found to be more then one level less refined then it is marked for refinement. When the neighbour search is complete for alls cells in the mesh, refinement actions as described above are carried out on all cells marked for refinement. The entire process is repeated in an iterative manner until there are no cells left to refine. Then the mesh is guaranteed to contain no transitions greater than $h$–$2h$ and the tree is balanced.

### 3.8 Load balancing

Refinement and derefinement may well change the number of cells allocated to each processor, resulting in an imbalance of computational work between processors. This is addressed using a dynamic load–balancing procedure which reallocates cells between adjacent processors until an equitable distribution is restored. The cells are arranged in ascending Morton code order and so too are the processors. Hence the load–balancing problem is one–dimensional, and can be solved using a diffusion analogy. Given a number of cells $N_p$ in each processor $p$, a discrete diffusion equation may be written as

$$N_p^{m+1} = N_p^m + \alpha_N \left[ \left( N_{p+1}^m - N_p^m \right) - \left( N_p^m - N_{p-1}^m \right) \right] \tag{72}$$

where the parameter $\alpha_N$ acts as a diffusivity and $m$ is an iteration index. The stability limit for the iteration is given by $\alpha_N = 1/2$. The form of the equation indicates that the diffusion of cells is driven by the difference in the number of cells between adjacent processors, and the stability limit indicates that the maximum number of cells transferred in any one iteration is equal to half of the difference. The data transferred between adjacent processors at each iteration consists of the Morton code and the set of conserved variables for each cell tranferred. The iteration proceeds until a suitable global load–balancing criterion has been met.

## 4 Stencil generation

The spatial discretisation scheme requires a stencil which consists of the local or central cell together with a set of other cells which are in close proximity in physical space. The size of the stencil is given by the

required order of accuracy of the discretisation scheme (see below). It is necessary to construct a stencil around each cell initially, and again whenever a refinement or derefinement operation has taken place.

During stencil construction, each cell in the mesh takes its turn to be the central cell, and its stencil is built up recursively by accumulating layers of cells that are face–neighbours of the cells in the previous layer. For the first layer, the previous (zeroth) layer contains only the central cell itself.

For every central cell there is a stencil list whose purpose is to store the cell index numbers of all the cells in the stencil. For the purposes of stencil generation, there is also a local remote–cell list for each central cell.

For every cell in the previous layer, all six faces are visited in turn. For each face, the neighbouring cell Morton code is constructed and a search of the octree is carried out to find the corresponding cell. There are five possible valid outcomes from the search:

1. Clean face: The neighbouring cell is inside the computational domain, is located on the same processor, and has the same size $h$ as the previous–layer cell. The neighbouring cell index is added to the stencil list for the central cell.

2. $h$–$2h$ transition: The neighbouring cell is inside the computational domain, is located on the same processor, and has twice the size of the previous–layer cell. The neighbouring cell index is added to the stencil list for thes central cell.

3. $2h$–$h$ transition: The neighbouring cell is inside the computational domain, is located on the same processor, and has half the size of the previous–layer cell. Of the eight sub–cells in the subdivided neighbour, four are direct face–neighbours of the previous–layer cell. All four Morton codes are constructed in turn, and the cell index for each is added to the stencil list for the central cell.

4. Remote cell: The neighbouring cell is inside the computational domain, but is allocated to a different processor. Further information is required to determine the location and size of the neighbouring cell. The neighbouring cell Morton code is added to the local remote–cell list together with the index of the central cell and a face code indicating which is the neighbouring face of the previous–layer cell. Nothing is added to the stencil list at this point.

5. Boundary face: The neighbouring cell is outside the computational domain, The neighbouring cell index is not added to the stencil list. If the neighbouring cell is in the first layer, a new boundary–condition cell index is generated.

When this procedure has been carried out for every cell in the previous layer, the relevant layer of the stencil is complete except for any remote cells. Each stencil list is then sorted in order of cell index number and any duplicate cells are removed.

The remote cells are handled by building a halo around the block of cells allocated to the local processor. The halo contains a copy of every remote cell identified by the stencil construction procedure on the local processor. Construction of each layer of the halo begins with the local remote–cell list for each central cell.

The local remote–cell list is first sorted into Morton code order. Any duplicate entries which have the same Morton code, central cell index AND face code are removed. The local remote–cell list is then added to a single remote–cell list for the entire layer. This list, in turn, is sorted into Morton code order and any duplicate entries (again for all three data items) are removed, producing the remote–cell requirement list for the layer.

The Morton codes alone are then extracted from the remote–cell requirement list and any duplicates are removed. This provides a Morton–code requirement list for the local processor. This list is exchanged between all processors, and an octree search is carried out on the local processor to find the cells that have been requested by all the other processors. A status code is returned to indicate the type of the requested cell, which can correspond to a clean face, or to either an $h$–$2h$ transition or a $2h$–$h$ transition. Remote cells and boundary faces are not valid outcomes at this stage. Using the returned status codes, the Morton–code requirement list is updated to form the layer–halo Morton code list, with a truncated

Morton code for each remote cell corresponding to a $h$–$2h$ transition, and eight extended Morton codes corresponding to the sub–cells implied by each $2h$–$h$ transition.

The layer–halo Morton code list is then used to build a new and purely local octree called the layer–halo octree. The layer–halo octree used to search the layer–halo Morton code list in turn for each member of the Morton–code requirement list, to identify and flag any potential layer–halo cells that are not actually members of at least one stencil. This step is needed since not every sub–cell included in the case of a $2h$–$h$ transition is necessarily a face–neighbour of a previous–layer cell.

The layer–halo octree is then searched again, using the list of Morton codes corresponding to cells that are already in the existing halo. Any existing–halo cell Morton code found in the layer–halo Morton code list is flagged to prevent duplication. Every member of the layer–halo Morton code list which has not been flagged is then used to assign a new cell within the halo.

Stencil construction is completed by searching the layer–halo octree once again, this time using the original remote–cell requirement list. The halo cell associated with each remote cell is identified and added to the stencil of the central cell that required it. Cell size transitions are identified. No special action is needed in the case of $h$–$2h$ transitions, but in the case of $2h$–$h$ transitions all face–neighbour sub–cells are added to the stencil.

For each central cell, the newly–completed layer of the stencil is now defined as the previous layer, and is used as the basis for building the next layer. For every cell in the new previous layer, all six faces are visited in turn, and the whole procedure is repeated, including the treatment of remote cells.

As each stencil becomes filled with the required total number of cells it is removed from the recursive loop over layers, until all the stencils have been completed.

### 4.1 Mesh connectivity

The solution algorithm requires the calculation of fluxes through the faces between neighbouring cells. This means that every cell must have a connection to its face–neighbours. During the stencil generation procedure a face–neighbour list is produced for every central cell. This contains six entries indexed by the direction of the outward–pointing normal vector for each face: West, East, South, North, Down, Up. Each entry in the face–neighbour list is a cell index number, and the face–neighbour list is similar to the first layer of the stencil. In the case of a $2h$–$h$ transition, the cell index number of the face–neighbour cell with the lowest Morton code is stored. The other three cell index numbers are found using their own face–neighbour lists in conjunction with a connectivity table.

## 5 Fluxes and Source Terms

### 5.1 Fourth–order reconstruction

The solution within each cubic cell is reconstructed in three dimensions using a polynomial of degree $p$ in the form [13]

$$u(x,y,z) = \bar{u}_0 + \sum_{k=1}^{K} a_k^{(u)} \phi_k(x,y,z) \tag{73}$$

where $u$ is a solution variable, $\bar{u}_0$ is the cell–averaged value of that variable in the central cell, $a_k^{(u)}$ is the $k$-th coefficient of the polynomial for the solution variable $u$ and the basis function $\phi$ is given by

$$\phi_k = \psi_k - \int_{-1/2}^{1/2}\int_{-1/2}^{1/2}\int_{-1/2}^{1/2} \psi_k dxdydz \tag{74}$$

where $\psi_k$ is the $k$–th monomial. Note that scaled local coordinates are used, treating the central cell as a unit cube of side $h_0$. The number of monomials $K$ is given by

$$K = p \ \ (1\text{D}); \quad K = \frac{1}{2}(p+1)(p+2) - 1 \ \ (2\text{D}); \quad K = \frac{1}{6}(p+1)(p+2)(p+3) - 1 \ \ (3\text{D}) \tag{75}$$

For a three–dimensional polynomial of degree $p = 3$, i.e. a fourth order accurate reconstruction, the number of monomials $K = 19$. The monomials for such a fourth order reconstruction are listed in Table 3. The ordering of the monomials is arbitrary, and has been chosen here to be essentially lexicographical, but modified to facilitiate reconstruction in one dimension (in $x$) or in two dimensions (in $x$ and $y$) as well as in three dimensions.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x$ | $x^2$ | $x^3$ | $y$ | $yx$ | $y^2$ | $yx^2$ | $y^2x$ | $y^3$ | $z$ | $zx$ | $zy$ | $z^2$ | $zx^2$ | $zxy$ | $z^2x$ | $zy^2$ | $z^2y$ | $z^3$ |

Table 3: Monomials $\psi_k$ for a fourth order reconstruction ($p = 3$)

The basis functions $\phi_k$ are equal to the monomials $\psi_k$ for all $k$ except for three special cases:

$$\phi_2 = x^2 - \frac{1}{12} \qquad \phi_6 = y^2 - \frac{1}{12} \qquad \phi_{13} = z^2 - \frac{1}{12} \tag{76}$$

The coefficients $a_k^{(u)}$ are obtained using the cell–averaged values $u_j$ in the central cell ($j = 0$) and in $J$ other cells which are nearby in physical space forming a stencil as discussed above. It is a necessary condition to have $J \geq K$, and stencil must have enough cells in each direction to support the required order of accuracy in all of them. A stencil which meets this criterion and has $J = K$ is said to be a *poised* stencil. In practice it is desirable to take $K \leq J \leq 2K$ and for the stencil to be as isotropic as possible in order to collect data equally from all directions around the central cell [14]. Inevitably, compromises must be made in constructing a stencil close to a boundary, and especially close to the edges and corners of the domain. Integration of the polynomial over cell $j$ of the stencil yields

$$\bar{u}_j = \bar{u}_0 + \sum_{k=1}^{K} a_k^{(u)} A_{jk} \tag{77}$$

The elements of the non–square matrix $A_{jk}$ are given by integrals of the basis functions as

$$A_{jk} = \frac{1}{\hat{h}_j} \int_{-\hat{h}_j/2}^{\hat{h}_j/2} \int_{-\hat{h}_j/2}^{\hat{h}_j/2} \int_{-\hat{h}_j/2}^{\hat{h}_j/2} \phi_k(x - \hat{x}_j, y - \hat{y}_j, z - \hat{z}_j) dx dy dz \tag{78}$$

where $\hat{h}_j = h_j/h_0$ is the size of the $j$-th cell and $(\hat{x}_j,\hat{y}_j,\hat{z}_j)$ is the location of its centre, both expressed in the scaled coordinate system of the central cell. Note that the evaluation of $A_{jk}$ is based purely on geometrical information about the local mesh and requires no information from the solution. The coefficients $a_k^{(u)}$ are found by solving the matrix equation

$$A_{jk} a_k^{(u)} = b_j^{(u)} \tag{79}$$

where $b_j^{(u)} = \bar{u}_j - \bar{u}_0$. The solution procedure makes use of Singular Value Decomposition (SVD) to obtain the Moore–Penrose pseudo–inverse $A_{jk}^+$. Since $A_{jk}$ contains only geometrical information, the pseudo–inverse needs to be computed only once and is common to all solution variables. Finally, the coefficients for each solution variable are obtained using matrix-vector multiplication as $a_k^{(u)} = A_{jk}^+ b_j^{(u)}$.

## 5.2 Flux Evaluation

The governing equations are integrated over each cell to produce the general form

$$\frac{\partial \overline{\mathbf{U}}}{\partial t} = \overline{\mathbf{S}}(\mathbf{U}, t) + \frac{1}{h} \sum_{k=1}^{6} \overline{\mathbf{F}}_k(\mathbf{U}) \tag{80}$$

where $\overline{\mathbf{U}} = \left\{\overline{\rho}, \overline{\rho u_i}, \overline{\rho E}, \overline{\rho Y}_\alpha\right\}$ is the vector of cell–averaged conserved variables, $\overline{\mathbf{S}} = \{0, 0, 0, \overline{w}_\alpha\}$ is the vector of cell–averaged source terms and $\overline{\mathbf{F}}_k$ is the vector of the face–averaged fluxes on face $k$ of the cell.

The cell–averaged reaction rates are given by the integral

$$\overline{w}_\alpha = \int_{-1/2}^{1/2}\int_{-1/2}^{1/2}\int_{-1/2}^{1/2} w_\alpha\left(Y_\alpha, T\right) dxdydz \tag{81}$$

expressed in local scaled coordinates, in which the cell volume is unity. The integration is done numerically using Gaussian quadrature according to

$$\overline{w}_\alpha = \sum_{p=1}^{n_v} W_p^v w_\alpha\left(Y_\alpha(\underline{x}_p), T(\underline{x}_p)\right) \tag{82}$$

For fourth order there are $n_v = 8$ Gauss points located at $\underline{x}_p = \left(\pm 1/2\sqrt{3}, \pm 1/2\sqrt{3}, \pm 1/2\sqrt{3}\right)$ with weights $W_p^v = 1/8$ for all $p$.

The fluxes $\overline{\mathbf{F}}_k$ are split into convective and diffusive contributions according to $\overline{\mathbf{F}}_k = \overline{\mathbf{D}}_k - \overline{\mathbf{C}}_k$ with face averages defined by

$$\overline{\mathbf{C}}_k = \int_{-1/2}^{1/2}\int_{-1/2}^{1/2} \mathbf{C}_k dA_k; \qquad \overline{\mathbf{D}}_k = \int_{-1/2}^{1/2}\int_{-1/2}^{1/2} \mathbf{D}_k dA_k \tag{83}$$

where $dA_k$ is the element of area on the cell face $k$. The convective and diffusive contribution are given by

$$\mathbf{C}_k = \begin{bmatrix} \rho u_k \\ \rho u_k u_i - p\delta_{ki} \\ \rho u_k E + p u_k \\ \rho u_k Y_\alpha \end{bmatrix} \qquad \mathbf{D}_k = \begin{bmatrix} 0 \\ \frac{\partial}{\partial x_k}\tau_{ki} \\ -\frac{\partial}{\partial x_k}q_k + \frac{\partial}{\partial x_k}\tau_{km}u_m \\ -\frac{\partial}{\partial x_k}\rho V_{\alpha,k} Y_\alpha \end{bmatrix} \tag{84}$$

Again the integration is done numerically using Gaussian quadrature:

$$\overline{\mathbf{C}}_k = \sum_{p=1}^{n_f} W_p^f \mathbf{C}_k\left(\underline{x}_p\right) \tag{85}$$

where for fourth order there are $n_f = 4$ Gauss points all with weights $W_p^f = 1/4$. For the cell face normal to the positive $x$–direction the Gauss points are located at $\underline{x}_p = \left(1/2, \pm 1/2\sqrt{3}, \pm 1/2\sqrt{3}\right)$ in scaled local coordinates, with the locations for other faces following by simple transformation.

The Gaussian integration procedure requires values of the relevant physical quantities at each Gauss point. For the conserved quantities these point values are obtained directly by evaluating the reconstructed polynomials. Primitive variables are evaluated locally from the conserved quantities at each Gauss point in order to preserve the full order of accuracy of the spatial reconstruction scheme. Point values of the derivatives as required by the diffusive fluxes are obtained by differentiating the reconstructed polynomials for the conserved quantities, and extracting the derivatives of primitive quantites at each point using algebraic expressions:

$$\frac{\partial}{\partial x_j}u_i = \frac{1}{\rho}\left[\frac{\partial}{\partial x_j}\rho u_i - u_i\frac{\partial}{\partial x_j}\rho\right] \tag{86}$$

$$\frac{\partial}{\partial x_j}Y_\alpha = \frac{1}{\rho}\left[\frac{\partial}{\partial x_j}\rho Y_\alpha - Y_\alpha\frac{\partial}{\partial x_j}\rho\right] \tag{87}$$

$$\frac{\partial}{\partial x_j}T = \frac{1}{\rho C_v}\left[\frac{\partial}{\partial x_j}\rho E - \sum_{\alpha=1}^{N}\left(h_\alpha - \frac{R^0}{W_\alpha}T\right)\frac{\partial}{\partial x_j}\rho Y_\alpha - u_k\frac{\partial}{\partial x_j}\rho u_k + \frac{1}{2}u_k u_k\frac{\partial}{\partial x_j}\rho\right] \tag{88}$$

The fluxes for each cell face are computed from reconstructed values obtained in the cells on both sides of the face. These left (L) and right (R) values for each Gauss point are not equal to each other in general and must be reconciled. For the diffusive fluxes, the L and R values of the flux are computed separately at each Gauss point and are simply averaged. For the convective fluxes, the implied discontinuity between the L and R thermochemical states across the cell face is treated using a Riemann solver at each Gauss point location.

### 5.3 Riemann solver

The gasdynamic discontinuity between the L and R states defines a Riemann problem which must be solved at each Gauss point before the quadrature for the flux is done. The initial discontinuity in the pressure and normal velocity component at $x = 0$ and $t = 0$ gives rise to three waves, called LR for left–running, RR for right–running and C for contact surface. The LR and RR waves can be either expansion or compression waves, and indeed the initial discontinuity may be strong enough for these to be expansion fans and shock waves respectively. These waves move at the local sound speed or shock speed as appropriate. The space between the LR and RR waves is called the starred region. The contact surface C lies between the LR and RR waves, and marks the interface between the L and R gasdynamic states which may have different transverse velocity components and different chemical compositions as well as different densities and temperatures. The contact surface moves at the same speed as the local gas, and there is negligible diffusion across it on the timescale of the Riemann problem.

The initial conditions are set by evaluating the polynomial on the L and R sides to obtain the conserved variables and from these also to obtain the relevant primitive variables. The solution of the Riemann problem will provide the values of the physical variables in the starred region, and hence the corersponding values on the cell face ($x = 0$) at times $t > 0$. Using these values it is possible to construct the vector $C_k$ which forms the integrand for the flux integral.

There are many ways to solve such a Riemann problem and there is a rich literature on the topic, e.g. [15]. For present purposes the Mach number is expected to be low, the inter–cell discontinuities are expected to be fairly small in magnitude and hence it is sufficient to solve a linearised Riemann problem. In this case the LR and RR waves are assumed to be isentropic compression or expansion (acoustic) waves moving at the local speeds of sound $a_L$ and $a_R$.

The linearised Riemann problem is defined in terms of the set of primitive variables $\{\rho, u_1, u_2, u_3, p, Y_\alpha\}$ evaluated on the L and R sides of a Gauss point on a cell face whose normal is in the $i$–direction. Since the LR and RR waves are both isentropic, it follows from characteristic relations across these waves that

$$p_L + (\rho a)_L \, u_{i,L} = p_* + (\rho a)_L \, u_{i,*}; \qquad p_R - (\rho a)_R \, u_{i,R} = p_* - (\rho a)_R \, u_{i,*} \tag{89}$$

Hence the pressure and normal velocity in the starred region are given by

$$p_* = \frac{(\rho a)_R \, p_L + (\rho a)_L \, p_R + (\rho a)_L \, (\rho a)_R \, (u_L - u_R)}{(\rho a)_L + (\rho a)_R} \tag{90}$$

$$u_{i,*} = \frac{(\rho a)_L \, u_L + (\rho a)_R \, u_R + (p_L - p_R)}{(\rho a)_L + (\rho a)_R} \tag{91}$$

Note that both $p_*$ and i$u_{i,*}$ are both uniform throughout the starred region, even across the contact surface. For the density, the relevant characteristic relations across the LR and RR waves are

$$p_L - \rho_L a_L^2 = p_* - \rho_{*L} a_L^2; \qquad p_R - \rho_R a_R^2 = p_* - \rho_{*R} a_R^2 \tag{92}$$

from which the two different values of the density in the starred region on either side of the contact surface follow as

$$\rho_{*L} = \rho_L + \frac{p_* - p_L}{a_L^2}; \qquad \rho_{*R} = \rho_R + \frac{p_* - p_R}{a_R^2} \tag{93}$$

Note that the species mass fractions also have different values on either side of the contact surface. Assuming chemically–frozen flow within the starred region, the mass fractions are given by $Y_{\alpha,*L} = Y_{\alpha,L}$

and $Y_{\alpha,*R} = Y_{\alpha,R}$. Assuming also that there is no shear across the contact surface, the two transverse velocity components are given by $u_{j,*L} = u_{j,L}$ and $u_{j,R*} = u_{j,R}$ for directions $j \neq i$. Since the contact surface is moving with velocity $u_{i,*}$, the values of the mass fractions and transverse velocity components on the cell face are given by a simple upwinding procedure (see Fig n). Then the cell face values of all the primitive variables are given by

$$\begin{bmatrix} \rho_f \\ u_{i,f} \\ u_{j,f} \\ P_f \\ Y_{\alpha,f} \end{bmatrix} = \begin{bmatrix} \rho_{*L} \\ u_{i,*} \\ u_{j,*L} \\ P_* \\ Y_{\alpha,*L} \end{bmatrix} \text{ for } u_{i,*} \geq 0; \quad \begin{bmatrix} \rho_f \\ u_{i,f} \\ u_{j,f} \\ P_f \\ Y_{\alpha,f} \end{bmatrix} = \begin{bmatrix} \rho_{*R} \\ u_{i,*} \\ u_{j,*R} \\ P_* \\ Y_{\alpha,*R} \end{bmatrix} \text{ for } u_{i,*} < 0 \tag{94}$$

# 6 Time Stepping

Time advancement of the solution is carried out using an explicit Total Variation Bounded (TVB) Runge–Kutta method [16] with adaptive time step control [17]. The governing equations (1)–(4) are written as

$$\frac{\partial \mathbf{U}}{\partial t} = \mathbf{R}(\mathbf{U}, t) \tag{95}$$

where $\mathbf{U}$ is the vector of conserved variables given by $\mathbf{U} = \{\rho, \rho u, \rho v, \rho w, \rho E, \rho Y_\alpha\}^T$ and $\mathbf{R}$ is the right–hand side vector containing all other terms in the equations. Clearly $\mathbf{R}$ depends on $\mathbf{U}$ and may also depend explicitly on time, for example through source terms or boundary conditions. The third–order three–stage explicit TVB Runge–Kutta scheme may be stated as:

$$\begin{aligned} \mathbf{U}^{(1)} &= \mathbf{U}^{(n)} + \delta t \mathbf{R}(\mathbf{U}^{(n)}, t^{(n)}) \\ \mathbf{U}^{(2)} &= \frac{3}{4}\mathbf{U}^{(n)} + \frac{1}{4}\mathbf{U}^{(1)} + \frac{1}{4}\delta t \mathbf{R}(\mathbf{U}^{(1)}, t^{(1)}) \\ \mathbf{U}^{(3)} &= \frac{1}{3}\mathbf{U}^{(n)} + \frac{2}{3}\mathbf{U}^{(2)} + \frac{2}{3}\delta t \mathbf{R}(\mathbf{U}^{(2)}, t^{(2)}) \end{aligned} \tag{96}$$

The scheme is equivalent to:

$$\begin{aligned} \mathbf{U}^{(n+1)} &= \mathbf{U}^{(n)} + \delta t \sum_{j=1}^{s} b_j \mathbf{R}(\mathbf{U}^{(j)}, t^{(j)}) \\ t^{(i)} &= t^{(n)} + c_i \delta t \\ \hat{\mathbf{U}}^{(n+1)} &= \mathbf{U}^{(n)} + \delta t \sum_{j=1}^{s} \hat{b}_j \mathbf{R}(\mathbf{U}^{(j)}, t^{(j)}) \end{aligned} \tag{97}$$

where $s = 3$ and the last line denotes the lower–order embedded scheme used to provide error estimates for the adaptive time–step controller. The numerical values of the coefficients are given in Table 5.
The time step is set adaptively using a PID–based controller [17]. The new time step $\delta t^{(n+1)}$ is given by

$$\delta t^{(n+1)} = \kappa \delta t^{(n)} \left( \frac{\varepsilon}{||E^{(n+1)}||_\infty} \right)^\alpha \left( \frac{||E^{(n)}||_\infty}{\varepsilon} \right)^\beta \left( \frac{\varepsilon}{||E^{(n-1)}||_\infty} \right)^\gamma \tag{98}$$

where $E$ is an error estimate obtained by finding the maximum of the normalised elements of $\hat{\mathbf{U}}$, $\varepsilon$ is a pre–set error tolerance, $\kappa$ is a safety factor and $\alpha$, $\beta$ and $\gamma$ are the parameters of the controller. The controller requires some tuning for optimal performance, but acceptable results have been obtained with $\varepsilon = 1.0 \times 10^{-3}$, $\kappa = 0.5$, $\alpha = 0.6/p$, $\beta = 0.2/p$ and $\gamma = 0$ where $p = 3$ is the order of the scheme.

| | | | | | |
|---|---|---|---|---|---|
| $c_1$ | 0 | $b_1$ | $\frac{1}{6}$ | $\hat{b}_1$ | $\frac{3}{4}$ |
| $c_2$ | 1 | $b_2$ | $\frac{1}{6}$ | $\hat{b}_2$ | $\frac{3}{16}$ |
| $c_3$ | $\frac{1}{2}$ | $b_3$ | $\frac{2}{3}$ | $\hat{b}_3$ | $\frac{3}{4}$ |

Table 4: Coefficients for the explicit TVB Runge–Kutta method.

# 7 Running the Code

Setting up a DNS run using `HAMISH` requires several steps. First the size of the computational domain and the size parameters for the thermochemical problem must be set by editing the `COMMON` blocks. Then the control parameters for the run must be set using the control data file, and the relevant chemical data must be provided using the chemical data file.

## 7.1 Common Data

The common data for `HAMISH` is contained in the `COMMON` block file `com_hamish.h`. This file must be edited in order to set the global and local sizes of the computational domain, the number of processors and the size of the parallel transfer array.

The section `HSIZES` is the first section of the file `com_hamish.h` and an example is shown below:

```
C     HSIZES-----------------------------------------------------------------

C     MAX NUMBER OF PROCESSORS
      INTEGER NPROMX
      PARAMETER(NPROMX = 16)

C     MAX SIZE OF LOCAL ARRAYS
      INTEGER NSIZMX
      PARAMETER(NSIZMX = 100000)

C     MAX SIZE OF BC DATA LISTS
      INTEGER NBCMAX
      PARAMETER(NBCMAX = 10000)

C     MAX SIZE OF REFINEMENT/DEREFINEMENT CELL LISTS
      INTEGER NRDMAX
      PARAMETER(NRDMAX = 100000)

C     MAX SIZE OF REMOTE CELL LISTS
      INTEGER NREMAX
      PARAMETER(NREMAX = 100000)

CC     MAX NUMBER OF STENCILS
C      INTEGER NSECMX
C      PARAMETER(NSECMX = 6)
```

```
C
CC     MAX SIZE OF STENCILS
C      INTEGER NTSTMX,NLAYMX,NCSTMX,NSSTMX
C      PARAMETER(NTSTMX = 300, NLAYMX = 150, NCSTMX = 38, NSSTMX = 38)

C      MAX SIZE OF STENCILS
       INTEGER NTSTMX,NLAYMX,NCSTMX
       PARAMETER(NTSTMX = 300, NLAYMX = 150, NCSTMX = 100)

C      MAX NUMBER OF COEFFICIENTS FOR POLYNOMIAL RECONSTRUCTION
       INTEGER NPCFMX
       PARAMETER(NPCFMX = 19)

CC     MAX NUMBER OF ELEMENTS IN THE OSCILLATION INDICATOR MATRIX
C      INTEGER NBISMX
C      PARAMETER(NBISMX = 37)

C      MAX SIZE OF PARALLEL TRANSFER ARRAY
       INTEGER NPARMX
       PARAMETER(NPARMX = 100000)

C      MAX SIZE OF PARALLEL TRANSFER HALO
       INTEGER NHALMX
       PARAMETER(NHALMX = 2)

C      MAX SIZE OF OCTREE ARRAY
C      NOTE: MIN VALUE = NSIZMX/8
       INTEGER NOCTMX
       PARAMETER(NOCTMX = 25000)

C      MAX NUMBER OF LEVELS IN OCTREE
       INTEGER NLEVMX
       PARAMETER(NLEVMX = 18)

C      MAX SIZE OF CELL INDEX CODES
       INTEGER NINDMX,NIOCMX,NIBTMX
       PARAMETER(NINDMX = 2, NIOCMX = 20, NIBTMX = 64)

C      MAX SIZE OF PARTITION INTERVAL TABLE
C      MIN VALUE: NTLSIZ = NINDMX*(NPROMX+1)
       INTEGER NTLSIZ
       PARAMETER(NTLSIZ = 34)

C      MAX NUMBER OF DIMENSIONS, OCTANTS, VELOCITY COMPONENTS, FACES
       INTEGER NDIMAX,NOCSMX,NCMPMX,NFACMX
       PARAMETER(NDIMAX = 3, NOCSMX = 8, NCMPMX = 3, NFACMX = 6)

C      MAX NO OF SPECIES
       INTEGER NSPCMX
       PARAMETER(NSPCMX = 18)

C      MAX NO OF PHYSICAL QUANTITIES
```

```
      INTEGER NQNTMX
      PARAMETER(NQNTMX = 2 + NCMPMX + NSPCMX)

C     MAX NO OF GAUSS POINTS
      INTEGER NG3DMX,NG2DMX,NGPFMX
      PARAMETER(NG3DMX = 8, NG2DMX = 4, NGPFMX = 16)

C     MAX SIZE OF CELL FACE HASH TABLE
      INTEGER NFCEMX
      PARAMETER(NFCEMX = 100000)

C     HSIZES-----------------------------------------------------------------
```

The parameter NPROMX must be set to the maximum number of processors that can be used for the run. The actual number of processors in use is specified in the command line when the codes is started.

The parameter NSIZMX must be set to define the maximum number of cells per processor. This implies that the global maximum number of cells is given by NPROMX$\times$ NSIZMX. Since the mesh is adaptive, the actual number of cells in the mesh at any time is likely to be somewhat less than the maximum.

The parameter NBCMAX must be set to define the local maximum number of boundary cells, and should correspond to the number of cells expected to occur on the outer boundary surface of the mesh.

The parameter NRDMAX must be set to define the local maximum number of cells that can be refined or derefined at any one time, which can be as large as NSIZMX but is likely to be significantly less.

The parameter NREMAX must be set to define the local maximum number of remote cells in the "halo" around the local domain that is required for the transfer of data between processors during parallel running.

The parameter NPARMX controls the size of the parallel transfer array, and must be set large enough to accommodate NREMAX remote cells.

The parameter NOCTMX controls the size of the octree array, and must be set to at least NSIZMX/8.

The parameter NSPCMX controls the number of species in the mixture.

The parameter NFCEMX controls the size of the cell face hash table, used for the evaluation of cell face fluxes.

The section CHEMIC of the file com_hamish.h contains the thermochemical data and is shown below:

```
C     CHEMIC-----------------------------------------------------------------

C     PARAMETERS
C     ==========
C     MAX NO OF SPECIES, NO OF STEPS
      INTEGER NSTPMX
      PARAMETER(NSTPMX=1)

C     THERMODYNAMIC DATA
C     MAX NO OF TEMPERATURE INTERVALS, THERMO POLYNOMIAL COEFFICIENTS
      INTEGER NTINMX,NCOFMX
      PARAMETER(NTINMX=2, NCOFMX=11)
C     MAX NO OF TEMPERATURE COEFFICIENTS, DITTO MINUS ONE
      INTEGER NCTMAX,NCTMM1
      PARAMETER(NCTMAX=5, NCTMM1=NCTMAX-1)

C     CHEMICAL RATE DATA
C     MAX NO OF THIRD BODIES
      INTEGER NBDYMX
      PARAMETER(NBDYMX=10)
```

```
C     MAX SIZE OF STEP SPECIES-LIST, STEP REACTANT-LIST
      INTEGER NSSMAX,NRSMAX
      PARAMETER(NSSMAX=10, NRSMAX=10)
C     MAX NO OF LINDEMANN STEPS
      INTEGER NLLMAX
      PARAMETER(NLLMAX=10)

C     TRANSPORT COEFFICIENTS
      DOUBLE PRECISION ALAMDC,RLAMDA,TLAMDA
      PARAMETER(ALAMDC=2.58D-5, RLAMDA=7.0D-1, TLAMDA=2.98D2)
      DOUBLE PRECISION PRANTL
      PARAMETER(PRANTL=7.0D-1)

C     UNIVERSAL GAS CONSTANT
      DOUBLE PRECISION RGUNIV
      PARAMETER(RGUNIV=8.3142D3)
      .
      .
      .
```

The parameter `NSTPMX` sets the maximum number of steps in the reaction mechanism. The minimum value 1, and it is expected that both `NSPCMX` and `NSTPMX` will be set equal to the corresponding values in the chemical data file (see below). This is checked automatically during startup. A legitimate exception occurs for non-reacting flow, when `NSTPMX=1` and the number of reaction steps is actually equal to zero.

The parameters `NTINMX`, `NCOFMX` and `NCTMAX` control the maximum sizes required to store the thermodynamic data for each species expressed in polynomial form (17). The parameter `NTINMX` sets the maximum number of temperature intervals required for any species while `NCOFMX` sets the maximum number of coefficients required in each interval. The parameter `NCTMAX` sets the maximum number of coefficients required in the polynomial for temperature (30).

Size parameters for the chemical rate data must also be set. The parameter `NBDYMX` must be set greater than or equal to the maximum number of distinct third bodies. The parameter `NSSMAX` must be set greater than or equal to the maximum number of species involved in any single reaction step. Similarly, the parameter `NRSMAX` must be set greater than or equal to the maximum number of reactant species involved in any single reaction step. The parameter `NLLMAX` must be set greater than or equal to the maximum number of reaction steps requiring Lindemann rate data. Note that these parameters are used only to set array sizes, and actual values for these quantities are set in the chemical data file.

The values of the parameters used to evaluate the transport coefficients according to (31) are set. Clearly `ALAMDA` and `RLAMDA` represent the multiplicative coefficient and temperature exponent respectively while `TLAMDA` is the reference temperature. The mixture Prandtl number in (32) is set using the parameter `PRANTL`.

Finally the value of the universal gas constant is set (in J/kmol K) using the parameter `RGUNIV`.

When the editing of `com_hamish.h` is complete, all Fortran source files for `HAMISH` must be recompiled in order to incorporate the changes into the code.

## 7.2 Control Data

The control data file `cont.dat` must be edited to set up the control parameters for the run. The file format is as shown:

```
************************************************
**                                            **
**       HAMISH: Run control data file        **
**                                            **
```

```
**********************************************

Global domain size (x,y,z) in metres
 1.0D-2    0.0D0    0.0D0

Global cells (nx) initial, base/max levels, step switch (0=fixed, 1=adaptive)
  2048      5    18     0

Time step; start step, no of steps, step switch (0=fixed, 1=adaptive)
 1.0D-12    1   100000   1

Intervals between dumps, reports, stats, results, dump o/p flag, results flag
  100000    1    100000    100000    1    1

Cold start switch (0=cold start, 1=restart), dump i/p flag (0/1=un/formatted)
 1    1

Initial turbulence generator
switch (0=off, 1=new, 2=inlet); random seed; turbulence parameters Rij, L
 0      -1      1.0D0  1.0D0  1.0D0  0.0D0  0.0D0  0.0D0  5.0D-4

Initial mesh adaption switch, flame generator switch (0=off, 1=on)
 0        0

Default initial conditions
pressure, temperature, velocity components u,v,w, level set
 1.0D5   3.0D2   2.05D0   0.0D0  0.0D0  0.0D0
mass fractions
 9
 1    2.8312571D-2
 2    2.26500566D-1
 3    0.0D0
 4    0.0D0
 5    0.0D0
 6    0.0D0
 7    0.0D0
 8    0.0D0
 9    7.45186863D-1

Global outer boundary condition types
one per line: x-left; x-right; y-left; y-right; z-left; z-right
(1=periodic; 1a=inlet; 2b=outlet; 3c=wall; a,b,c denotes BC subtype)
(four integer and four real parameters allowed for each)
13     1    0    0   0   2.05D0    0.0D0    0.0D0    0.0D0
21     0    0    0   0   9.7817D4  2.87D-1  1.0D-3   0.0D0
1      0    0    0   0   0.0D0     0.0D0    0.0D0    0.0D0
1      0    0    0   0   0.0D0     0.0D0    0.0D0    0.0D0
1      0    0    0   0   0.0D0     0.0D0    0.0D0    0.0D0
1      0    0    0   0   0.0D0     0.0D0    0.0D0    0.0D0

Mesh refinement/derefinement criteria, interval, min/max levels
 1.0D-1   2.5D-2     1     6    18
```

```
Mesh refinement scaling factors
density, temperature, velocity components u,v,w, level set
 1.0D-1   1.0D-1   1.0D-1  1.0D-1  1.0D-1  1.0D-1
mass fractions
 1    5.0D-1
 2    5.0D-1
 3    5.0D-1
 4    5.0D-1
 5    5.0D-1
 6    5.0D-1
 7    5.0D-1
 8    5.0D-1
 9    5.0D-1

End of file
```

The first five lines are treated as a header and are read but ignored by the `HAMISH` code. The various data items are then listed in groups, and the format of the control file is fixed to the extent that the number of lines in each group and the number of data items per line must be preserved.

The global size of the domain in the $x$, $y$ and $z$ directions must be specified in metres. If the value specified in the $z$–direction is zero, then the domain is taken to be two–dimensional in $x$ and $y$. If the value specified in the $y$–direction is also zero, then the domain is taken to be one–dimensional in $x$ only.

The number of cells required for the initial mesh is specified for the $x$–direction only. The specified global domain size in the $x$–direction is divided by the specified number of cells to yield a value for the mesh spacing $h$. This spacing is used to define the number of cells required in each of the other two directions. This information is used to construct the initial mesh and the corresponding octree, which are both distributed automatically across the available processors. The root of the octree is taken as level 0. This corresponds to the Big Cube which is just large enough to encompass the whole computational domain. The base mesh level specifies the coarsest mesh to be used initially for the present problem, i.e. no derefinement will take place at this level. The maximum level specifies the finest mesh, at which no further refinement will take place. A switch is provided to turn the spatial adaption off (switch =`1`) for a fixed mesh, or on (switch =`1`) for adaptive mesh.

The initial time step must be set (in seconds), together with the index of the first time step and the required number of time steps for the run. A switch must be set to control whether a fixed time step (switch=`0`) or adaptive time-stepping (switch=`1`) is required. The number of time steps between dumps must be set. A restart file is written for each processor at the start of a run, and a further restart file is dumped for each processor after the specified number of time steps. Throughout the remainder of the run. In order to help guard against loss of data, two restart files are kept for each processor and the older file of the two is overwritten at each dump. The restart file names conform to the pattern `dmpiXXXXXXY.dat` where `XXXXXX` is the rank index of the corresponding processor (starting from `000000`) and `Y` is the restart file index which is either `0` or `1`. The latest dump may be contained in either restart file and alternates between the two. The number of time steps between updates to the report file, the statistics file and the results files must also be set. A single report file and a single statistics file are each created at the start of the run by the lowest-ranked processor. A results file is created for every processor. The report file has the fixed name `rept.res` and the statistics file has the fixed name `stat.res`. The report file name has the format `resoXXXXXXYYYY.res` where `XXXXXX` is the processor rank index and `YYYY` is a sequence number starting from `0000` that is updated each time a set of results file is written. In all cases, the file is updated after the specified number of time steps.

Initial conditions for the run must be set. The cold start switch must be set to `0` for a cold start, in which the run is initialised from scratch using specified initial data. Alternatively, the cold start switch must be set to `1` for a restart, using data from a previous dump. A restart file must exist for each processor, and the restart file names must conform to the pattern `dmpiXXXXXXY.dat` where `XXXXXX` is the rank index of the corresponding processor and `Y` is the restart file index. It is expected that `Y` will be set to `0` for a restart.

The initial turbulence generator switch must be set to `1` if an initial turbulent field is to be generated, or set to `0` if not. If the switch is set to `1` then the random seed must also be set in order to initialise the random number generator. If the (integer) value of the random seed is set to be non-negative, the value set is added to the rank of each processor in order to produce a different random seed for each processor. This ensures that there is no repetition of the same random sequence on each processor. Conversely, if the value of the random seed is set to be a negative integer, the value is used globally in order to ensure that the same global initial turbulent field is generated for a given global grid size irrespective of the number of processors in use. Up to seven parameters may be set in order to control the properties of the initial turbulence. These may be (for example) the six independent components of the Reynolds stress tensor together with a measure of the turbulence length scale. Note that the initial turbulence generator is not yet fully implemented.

The initial mesh adaption switch must be set to `1` if the mesh is to be adapted statically before the start of the run, or set to `0` if not.

The initial flame generator switch must be set to `1` if an initial scalar field is to be specified, or set to `0` if not. This switch controls whether or not the subroutine `FLAMIN` is called within `HAMISH`. This subroutine is designed to allow for the specification of an initial scalar field of arbitrary complexity.

Default initial conditions must be set for each variable. Pressure, temperature and the three velocity components must be set using SI units, and there is an option also to specify a general dimensionless level–set variable. The total number of species must be set and must correspond to the number of species specified in `com_hamish.h`. This is checked by `HAMISH` during startup. For each species an index number and a value for its initial mass fraction must be set. Each index number must correspond to the species number specified in the chemical data file `chem.dat` as described below. The total of all mass fractions must be equal to unity, and this is checked by `HAMISH` during startup.

Boundary condition types must be specified for the global domain. This is done using an indexing system, and up to four real and four integer parameters may be set for each boundary condition subtype.

Periodic boundary conditions are specified using an index value of `1`. Both of the corresponding periodic faces of the domain must have this index value and this is checked during startup.

Inlet boundaries are specified using an index value of the form `1a`, where `a` denotes the inlet boundary condition subtype. Three inlet boundary condition subtypes are implemented:
`11` Subsonic non-relecting laminar inflow
This boundary condition is not yet implemented.
`12` Subsonic reflecting turbulent inflow with specified temperature
By default the inlet temperature is set to the same value as specified for the default initial temperature.
`13` Subsonic reflecting turbulent inflow with specified density
By default the inlet density is set to the same value as the initial density as computed from the values specified for the default initial pressure, temperature and species mass fractions.
For both of these inlet boundary condition subtypes, the value of the first integer parameter determines the nature of the inlet velocity field. A value of `1` specifies a laminar inflow with constant velocity, whereupon the first real parameter is taken to specify the inflow velocity component normal to the boundary.

A value of 2 specifies a laminar inflow with velocity varying sinusoidally in time, whereupon the first two real parameters specify the amplitude and period of the oscillation. A value of 3 specifies a turbulent inflow. Note that sinusoidal and turbulent inflow conditions are not yet implemented.

Outlet boundaries are specified using an index of the form 2b, where b denotes the outlet boundary condition subtype.
21 Subsonic non-reflecting outflow
This condition requires three real parameters to be set. In order, these specify the pressure at infinity, the relaxation parameter and the nominal boundary Mach number.
22 Subsonic reflecting outflow
This condition requires one real parameter to be set, to specify the constant pressure at the outlet.
Wall boundaries are specified using an index of the form 3c, where c denotes the wall boundary condition subtype. The list of boundary condition types as implemented in the code is given below: 31 Adiabatic no-slip wall
This condition requires no additional parameters to be set.
32 Isothermal no-slip wall
This condition requires a single real parameter to be set in order to specify the wall temperature.

A set of mesh refinement criteria must be set to control the spatial adaption. The first real number is the maximum value of the normalised local spatial error at which refinement will take place, while the second real number is the minimum value of the same quantity at which derefinement will take place. The first integer is the interval between refinement/derefinement steps, while the final two integers specify the minimum and maximum refinement levels. note that these two values override the initial values set above.

The final block of data provides scaling factors for the estimation of the normalised local spatial error for the refinement/derefinement procedure. A scaling factor can be set for each variable named.

This completes the description of the control data file.

## 7.3 Chemical Data

The chemical data file chem.dat must be set up to contain the thermodynamic and chemical data required for the run. The file may be edited directly or it may be generated using the chemical preprocessor PPCHEM (see below). The file format is as shown:

```
*****************************
*                           *
*  Output file from ppchem  *
*                           *
*****************************
Species list:
   9
   1   H2
   2   O2
   3   H2O
   4   O
   5   OH
   6   H
   7   HO2
   8   H2O2
   9   N2
```

```
Species data:
 1.0000E+05
   1  2.0000E+00
 3.0000E-01
   1
 3.0000E+02  3.0000E+03   11
 2.9521377E+00
 3.5829958E-03
-8.7977261E-06
 1.0524709E-08
-6.2289022E-12
 1.7507051E-15
-1.2174423E-19
-4.1415591E-23
 6.7401263E-27
-9.7947131E+02
-1.8717840E+00
       .
       .
       .
   9  2.8000E+01
 1.0000E+00
   1
 3.0000E+02  3.0000E+03   11
 2.9063202E+00
 5.4218160E-03
-2.0239021E-05
 3.9419891E-08
-4.0993331E-11
 2.4291129E-14
-8.2509206E-18
 1.4969804E-21
-1.1250780E-25
-9.8969384E+02
 5.4798346E+00
Step rate data:
  21
   1  3.5500E+12 -4.1000E-01  6.9501E+07
       .
       .
       .
  21  5.8000E+11  0.0000E+00  4.0026E+07
Step species-list:
   1    4
   1    1       2
   1    2       4
   1    3       5
   1    4       6
       .
       .
       .
  21    4
```

```
  21    1       3
  21    2       5
  21    3       7
  21    4       8
Step reactant-list:
   1    2
   1    1       2
   1    2       6
        .
        .
        .
  21    2
  21    1       5
  21    2       8
Step product-list:
   1    2
   1    1       4
   1    2       5
        .
        .
        .
  21    2
  21    1       3
  21    2       7
Step reactant coefficient-list:
   1    0
        .
        .
        .
  21    0
Step product coefficient-list:
   1    0
        .
        .
        .
  21    0
Species delta-list:
   1    1 -1.0
   1    2  1.0
   1    3  1.0
   1    4 -1.0
        .
        .
        .
  21    1  1.0
  21    2 -1.0
  21    3  1.0
  21    4 -1.0
Third-body list:
   2
   1   M1
   2   M2
```

```
Third-body step-list:
   1       0
        .
        .
        .
  21       0
Third-body efficiencies:
   1    1  2.5000E+00
        .
        .
        .
   1    9  1.0000E+00
   2    1  2.0000E+00
        .
        .
        .
   2    9  1.0000E+00
Gibbs step-list:
  21
   1       1
        .
        .
        .
  21      21
Lindemann step-list:
   2
   1       0
        .
        .
        .
  21       0
Lindemann step rate data:
   1  6.3700E+14 -1.7200E+00  2.1771E+06  8.0000E-01
   2  1.2000E+14  0.0000E+00  1.9050E+08  5.0000E-01
Troe step-list:
   0
Troe step rate data:
SRI step-list:
   0
SRI step rate data:
End of file
```

The example shown is an abbreviated version of the chemical data file for the 9–species 21–step hydrogen oxidation mechanism by Li et al. [18]. In most cases it is expected that the chemical data file will be generated using the chemical data preprocessor `PPCHEM` described below, although some manual editing may be required also.

The first five lines are treated as a header and are read but ignored by the `HAMISH` code. The various data items are then listed in groups, where the format of each group is fixed and is determined by the nature of the data involved. Note that not all groups are required for all reaction mechanisms, and in this case a one-line header appears as a placeholder for the group.

The *species list* associates an integer identifier with each species involved in the reaction mechanism. After the one-line header, the first line contains an integer $N$ specifying the number of species in the list and hence the length of the list. Each subsequent line contains an integer identifier $\alpha$ and a string representing the species chemical symbol $\mathcal{M}_\alpha$. Each integer identifier must be unique, and each species string must contain no spaces. The default maximum length of a species string is 10 characters.

The *species data* group contains the thermodynamic data required for each species. Following the one-line header the first line specifies the reference pressure in Pa. There follows a series of data blocks with one block per species. For each block the first line contains the species integer identifier and the molar mass of the species in kg/kmol, and the second line specifies the Lewis number for the species. The next line contains an integer which specifies the number of temperature ranges over which the thermodynamic data is defined. The data for each temperature range is specified in a sub-block whose first line contains two real numbers specifying the lower and upper limits of temperature (in Kelvin) and an integer specifying the number of temperature coefficients for that range. The remainder of the sub-block contains the coefficients arranged one per line. There are as many sub-blocks as there are temperature ranges for each species, and as many data blocks as there are species in the species list.

The *step rate data* group contains the Arrhenius rate parameters for each forward step in the reaction mechanism. The group consists of a one-line header followed by a line containing an integer $M$ specifying the number of steps in the reaction mechanism. This is followed in turn by one line for each step containing a unique integer identifier $m$ for the step followed by three real numbers specifying the Arrhenius rate parameters $A_m$,$n_m$ and $E_m/R^0$, in accordance with eq.(40).

The *step species list* provides a compact list of the species involved in each step of the reaction mechanism, i.e. a species for which either $\nu'_{\alpha,m}$ or $\nu''_{\alpha,m}$ (or possibly both) takes a non-zero value. Note that third bodies are not included in this list. The group consists of a one-line header followed by a number of data blocks, one for each step in the mechanism. Each block begins with a line containing two integer values, the first being the value of $m$ that uniquely identifies the step and the second specifing the number of species involved in that step. Subsequent lines in each block contain three integer values of which the first is a repeat of the step identifier $m$, the second is an index number applicable only within the current step, and the third identifies a species using the unique integer identifier $\alpha$ as previously defined in the species list. There are as many lines in each block as there are species taking part in the step, and there are as many blocks as there are steps in the mechanism.

The *step reactant list* provides a compact list of the species involved in each step of the mechanism as molecular reactant species, i.e. a species for which $\nu'_{\alpha,m}$ takes an integer value greater than zero. Note that third bodies are not included in the list. The group has the same structure as the step species list, consisting of a one-line header followed by a number of data blocks, one for each step in the mechanism. Each block begins with a line containing two integer values, the first being the value of $m$ that uniquely identifies the step and the second specifing the number of entries in the list for that step. Subsequent lines in each block contain three integer values of which the first is a repeat of the step identifier $m$, the second is an index number applicable only within the current step, and the third uses the unique species identifier $\alpha$ to identify a reactant species within the current step. If $\nu'_{\alpha,m}$ has an integer value greater than unity then the entry for species $\alpha$ is repeated to appear a total of $\nu'_{\alpha,m}$ times, each with a unique index number but with the same species identifer. There are as many lines in each block as required for the number of species for each step including repeats, and there are as many blocks as there are steps in the mechanism.

The *step product list* provides a compact list of the species involved in each step of the mechanism as molecular product species, i.e. a species for which $\nu''_{\alpha,m}$ takes an integer value greater than zero. Note that third bodies are not included. The group has the same structure as the step species list, consisting of a one-line header followed by a number of data blocks, one for each step in the mechanism. Each block

begins with a line containing two integer values, the first being the value of $m$ that uniquely identifies the step and the second specifing the number of entries in the list for that step. Subsequent lines in each block contain three integer values of which the first is a repeat of the step identifier $m$, the second is an index number applicable only within the current step, and the third uses the unique species identifier $\alpha$ to identify a product species within the current step. If $\nu''_{\alpha,m}$ has an integer value greater than unity then the entry for species $\alpha$ is repeated to appear a total of $\nu''_{\alpha,m}$ times each with a unique index number but with the same species identifer. There are as many lines in each block as required for the number of species in each step including repeats, and there are as many blocks as there are steps in the mechanism.

The *step reactant coefficient list* provides a list of the reactant species for which the value of $\nu'_{\alpha,m}$ is non-zero but is not a positive integer. Note that third bodies are not included. The group has the same structure as the step species list, consisting of a one-line header followed by a number of data blocks, one for each step in the mechanism. Each block begins with a line containing two integer values, the first being the value of $m$ that uniquely identifies the step and the second specifing the number of entries in the list for that step. Subsequent lines in each block contain three integer values and one real value. The first integer is a repeat of the step identifier $m$, the second is an index number applicable only within the current step, and the third uses the unique species identifier $\alpha$ to identify a reactant species within the current step. The real value is the value of $\nu'_{\alpha,m}$ for that species within the step. There are as many lines in each block as required to specify all of the non-zero non-positive-integer values of $\nu'_{\alpha,m}$ for that step, and there are as many blocks as there are steps in the mechanism.

The *step product coefficient list* provides a list of the product species for which the value of $\nu''_{\alpha,m}$ is non-zero but is not a positive integer. Note that third bodies are not included. The group has the same structure as the step species list, consisting of a one-line header followed by a number of data blocks, one for each step in the mechanism. Each block begins with a line containing two integer values, the first being the value of $m$ that uniquely identifies the step and the second specifing the number of entries in the list for that step. Subsequent lines in each block contain three integer values and one real value. The first integer is a repeat of the step identifier $m$, the second is an index number applicable only within the current step, and the third uses the unique species identifier $\alpha$ to identify a product species within the current step. The real value is the value of $\nu''_{\alpha,m}$ for that species within the step. There are as many lines in each block as required to specify all of the non-zero non-positive-integer values of $\nu''_{\alpha,m}$ for that step, and there are as many blocks as there are steps in the mechanism.

The *step species delta-list* contains the value of the difference of stoichiometric coefficients $\left(\nu''_{\alpha,m} - \nu'_{\alpha,m}\right)$ for each species in each step in the mechanism. Note that third bodies are not included. The group consists of a one-line header followed by a number of data blocks, one for each step in the mechanism. Each line in each block contains two integer values and one real value. The first integer is a repeat of the step identifier $m$ while the second is an index number applicable only within the current step. The real value is the value of $\left(\nu''_{\alpha,m} - \nu'_{\alpha,m}\right)$ for that species within the step. There are as many lines in each block as required for the number of species involved in the step, and there are as many blocks as there are steps in the mechanism.

The *third body list* associates an integer identifier with each third body involved in the reaction mechanism. After the one-line header, the first line contains an integer specifying the number of distinct third bodies required by the mechanism and hence the length of the list. Each subsequent line contains an integer identifier and a string representing the generic chemical symbol for the corresponding third body. Each third-body integer identifier must be unique, and each third-body string must contain no spaces. The default maximum length of a third-body string is 10 characters.

The *third-body step-list* indicates which steps in the reaction mechanism involve a third body. After the one-line header, each line contains two integer values. The first integer corresponds to the step number $m$. If a step does not involve a third body then the second integer value is set equal to zero. If a step does

involve a third body then the second integer value is the third body identifier for that step, as already defined in the third body list.

The *third-body efficiencies* are listed for each third body identified in the third body list. For every third body there is a data block consisting of a number of lines equal to the number of species $N$. Each line contains two integer values and a real value. The first integer is the unique identifier for the third body as previously specified in the third body list. The second integer is the species number $\alpha$ as previously specified in the species list, and the real value is the value of the third body efficiency $\eta_{\alpha,M}$.
The *Gibbs step-list* indicates which steps in the mechanism need to have the backward reaction rate evaluated using the Gibbs function. The first line contains a single integer value specifying the number of Gibbs steps. If this number is non-zero, there follows a number of lines equal to the number of steps in the reaction mechanism. Each line contains two integer values, of which the first is the step index number $m$ as previously specified in the step species list. If the step requires evaluation of the reaction rate using the Gibbs function then the second integer value is equal to $m$, otherwise it is equal to zero.

The *Lindemann step-list* indicates which steps in the mechanism need to have the reaction rate evaluated using a Lindemann form. The first line contains a single integer value specifying the number of Lindemann steps. If this number is non-zero, there follows a number of lines equal to the number of steps in the reaction mechanism. Each line contains two integer values, of which the first is the step index number $m$ as previously specified in the step species list. If the step requires evaluation of the reaction rate using a Lindemann form then the second integer value specifies a unique integer identifier for the Lindemann step, otherwise it is equal to zero.

The *Lindemann step rate data* is listed for each Lindemann step. Each entry consists of an integer identifier for the Lindemann step together with the real-number values of the Lindemann fall-off rate parameters $A_{\rm fall}$, $n_{\rm fall}$ and $E_{\rm fall}/R^0$. The length of the list is equal to the number of Lindemann steps as already specified in the Lindemann step list.

# References


[1] K.W. Jenkins, R.S. Cant: Direct Numerical Simulation of turbulent flame kernels, in "Recent Advances in DNS and LES", eds. D. Knight and L. Sakell, pp191–202, Kluwer Academic, New York, 1999.

[2] R.S. Cant: `SENGA2` User Guide, Tech. Report CUED–THERMO–2012/04, Cambridge University Engineering Department, 2012.

[3] R.J. Kee, F.M. Rupley, E. Meeks, J.A. Miller: CHEMKIN–III: A Fortran chemical kinetics package for the evaluation of gas–phase chemical and plasma kinetics, report SAND96-8216, Sandia National Laboratories, 1996.

[4] A. Ern, V. Giovangigli: *Multicomponent Transport Algorithms*, Lecture Notes in Physics **24**, Springer–Verlag, 1994.

[5] F.A. Lindemann, Discussion on "The Radiation Theory of Chemical Action", Trans. Faraday Soc. **17**, 598–606, 1922.

[6] R.G. Gilbert, K. Luther, J. Troe: Theory of thermal unimolecular reactions in the fall–off range. II. Weak collision rate constants, Ber. Bunsenges. Phys. Chem. **87**, 169–177, 1983.

[7] P.H. Stewart, C.W. Larson, D. Golden: Pressure and temperature dependence of reactions proceeding via a bound complex. 2. Application to $2CH_3 \rightarrow C_2H_5 + H$, Combust. Flame **75**, 25–32, 1989.

[8] M.B. Giles: Nonreflecting boundary conditions for Euler equation calculations, AIAA Journal **28**, 2050–2058, 1990.

[9] T. Poinsot, S. Lele: Boundary conditions for direct simulations of compressible viscous flows, J. Comp. Phys, **101**, 104–129, 1992.

[10] J.C. Sutherland, C.A. Kennedy: Improved boundary conditions for viscous, reacting compressible flows, J. Comp. Phys. **191**, 502–524, 2003.

[11] T. Tu, H. Yuy, L. Ramirez–Guzmanz, J. Bielakx, O. Ghattas, K–L. Mak, D.R. O'Hallaron: From mesh generation to scientific visualization: An endi–toi–end approach to parallel supercomputing, Supercomputing 2006, Tampa, Florida, USA, IEEE, 2006.

[12] G. M. Morton: A computer oriented geodetic data base and a new technique in file sequencing, IBM Tech. Report, 1966.

[13] L. Ivan, C.P.T. Groth: High-order solution–adaptive central essentially non–oscillatory (CENO) method for viscous flows, J. Comput. Phys. **257**, 830–862, 2014.

[14] P. McCorquodale, P. Colella: A high-order finite–volume method for conservation laws on locally–refined grids, Comm. App. Math. Comp. Sci. **6**, 1–26, 2011.

[15] J.J. Gottlieb, C.P.T. Groth: Assessment of Riemann solvers for unsteady one–dimensional inviscid flows of perfect gases, J. Comput. Phys. **78**, 437–458, 1988.

[16] C–W. Shu, S. Osher: Efficient implementation of essentially non-oscillatory shock–capturing schemes, J. Comput. Phys. 77, 439–471, 1988.

[17] C.A. Kennedy, M.H. Carpenter, R.M. Lewis: Low–storage, explicit Runge–Kutta schemes for the compressible Navier–Stokes equations, Appl. Numer. Math. **35**, 177–219, 2000.

[18] J. Li, Z. Zhao, A. Kazarov, F.L. Dryer: Int. J. Chem. Kinet. **36**, 566–575, 2004.