SUNSET code: Scalable Unstructured Node-SET code for high fidelity fluid simulations in complex geometries

J. R. C. King*
Department of Mechanical, Aerospace and Civil Engineering,
The University of Manchester, Manchester, M13 9PL, UK
(Dated: November 30, 2022)

This document contains a brief description and guide to the SUNSET code. Numerical methods in this code follow [1] (also available at arXiv:2102.02019).

I. OVERVIEW

The **S**calable **U**nstructured **N**ode-**SET** code (SUNSET code) is designed for high fidelity simulations of fluid flows on non-trivial geometries. The spatial discretisation is by a combination of LABFM [1, 2] and finite differences. Order of accuracy of spatial discretisation can be specified (between 4^{th} and 12^{th}). Time integration via explicit 3^{rd} order Runge-Kutta scheme. Boundary conditions with the characteristic based formulation, with non-periodic boundaries dropping to 4^{th} order at walls, inflows and outflows. Resolution is non-uniform in first two dimensions. Scalable via OpenMP and MPI. Thermodynamics can vary from isothermal to thermal, with temperature dependent transport properties. Solves for an arbitrary number of miscible chemical species.

A. A comment on notation

Subscripts i, j, and k are vector indices for Einstein notation. Subscripts α and β indicate chemical species. Subscripts a and b indicate collocation point indices.

II. GOVERNING EQUATIONS

The code solves the compressible Navier-Stokes equations in conservative form

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

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_i u_k}{\partial x_k} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ki}}{\partial x_k} + \rho f_i \tag{1b}$$

$$\frac{\partial \rho E}{\partial t} + \frac{\partial \rho u_k E}{\partial x_k} = -\frac{\partial p u_k}{\partial x_k} - \frac{\partial q_k}{\partial x_k} + \frac{\partial \tau_{ki} u_i}{\partial x_k} + u_k f_k \tag{1c}$$

$$\frac{\partial \rho Y_\alpha}{\partial t} + \frac{\partial \rho u_k Y_\alpha}{\partial x_k} = \dot{\omega}_\alpha - \frac{\partial \rho V_{\alpha,k} Y_\alpha}{\partial x_k} \qquad \forall \alpha \in [1, N_S] \tag{1d}$$

$$\tag{1e}$$

where ρ is the density, u_i the i-th component velocity, p the pressure, E the total energy, τ the viscous stress, f a body force, q the heat flux, Y_α the concentration of species $\alpha \in [1, N_S]$, and $V_{\alpha,k}$ is the k-th component of the molecular diffusion velocity of species α. $\dot{\omega}_\alpha$ is the production rate of species α.

The viscous stress is defined as

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

where μ is the dynamic viscosity and δ_{ik} is the Kronecker delta. The energy is

$$E = \sum_\alpha h_\alpha Y_\alpha - \frac{p}{\rho} + \frac{1}{2} u_i u_i \tag{3}$$

* jack.king@manchester.ac.uk

where the sum over α is over all $\alpha \in [1, N_S]$, and h_α is the enthalpy of species α. The molecular diffusion term in (1d) is modelled as a Fickian diffusion process with

$$-\rho V_{\alpha,k} Y_\alpha = \rho D_\alpha \frac{\partial Y_\alpha}{\partial x_k} - Y_\alpha \sum_{\beta \in [1,N_S]} \rho D_\beta \frac{\partial Y_\beta}{\partial x_k}, \tag{4}$$

in which D_α is the molecular diffusivity of species α, and the final term is a correction to ensure that

$$\sum_\alpha Y_\alpha = 1. \tag{5}$$

The heat flux vector is then given by

$$q_k = -\lambda \frac{\partial T}{\partial x_k} + \sum_\alpha \rho V_{\alpha,k} Y_\alpha h_\alpha, \tag{6}$$

where λ is the thermal conductivity of the mixture. The system is closed with an ideal gas equation of state

$$p = \rho R_{mix} T = \rho R_0 T \sum_\alpha \frac{Y_\alpha}{W_\alpha} \tag{7}$$

where R_{mix} is the gas constant for the mixture, R_0 is the universal gas constant, and W_α is the molar mass of species α.

A. Evaluation of temperature and heat capacity

The temperature dependence of the specific heat capacity of species α is given by a polynomial of order J

$$c_{p,\alpha} = \sum_{j=1}^{J} a_{\alpha,j} T^{j-1}, \tag{8}$$

which is valid over a specified temperature range $T \in [T_{low}, T_{high}]$, and in which the coefficients $a_{\alpha,j}$ are taken from CHEMKIN tables. Integration of this yields an expression for the enthalpy

$$h_\alpha = \sum_{j=1}^{J} \frac{a_{\alpha,j}}{j} T^j + a_{\alpha,J+1}, \tag{9}$$

whilst the temperature derivative of $c_{p,\alpha}$ is

$$\frac{dc_{p,\alpha}}{dT} = \sum_{j=1}^{J} (j-1)\, a_{\alpha,j} T^{j-2}. \tag{10}$$

Noting that the specific heat capacity of species α is R_0/W_α, where R_0 is the universal gas constant and W_α is the molar mass of species α, the energy can be written as

$$E = \sum_\alpha Y_\alpha \left(h_\alpha - R_0 \frac{T}{W_\alpha} \right) + \frac{1}{2} \boldsymbol{u} \cdot \boldsymbol{u}. \tag{11}$$

Substituting the polynomial for h_α and collecting terms in powers of T, we can write

$$f(T) = C_0 + \sum_{j=1}^{J} C_j T^j = 0, \tag{12}$$

where the coefficients C are given by

$$C_0 = \frac{1}{2} \boldsymbol{u} \cdot \boldsymbol{u} - E + \sum_\alpha Y_\alpha a_{\alpha,J+1} \tag{13a}$$

$$C_1 = \sum_\alpha Y_\alpha a_{\alpha,1} - R_{mix} \tag{13b}$$

$$C_j = \sum_\alpha Y_\alpha \frac{a_{\alpha,j}}{j} \qquad \forall j \in [2, J]. \tag{13c}$$

The derivative of f is

$$f'(T) = \sum_{j=1}^{J} j C_j T^{j-1}. \tag{14}$$

Having calculated the coefficients C, we solve the nonlinear equation (12) numerically via a Newton-Raphson method to obtain the temperature T.

Note that in the case where $a_{\alpha,1} \neq 0$ and $a_{\alpha,j\in[2,J]} = 0$ (i.e. $c_{p,\alpha}$ is independent of temperature), $f(T)$ is linear in T, and the Newton-Raphson method will converge (exactly) after a single iteration.

B. Temperature dependent transport properties

The temperature dependence of the transport properties is modelled by the relationship

$$\mu = \mu_{ref} \left(\frac{T}{T_{ref}} \right)^{r_T}, \tag{15}$$

where μ_{ref} is the viscosity at the reference temperature T_{ref}, and $r_T = 0.7$ is a constant exponent. The mixture thermal conductivity is given by

$$\lambda = \frac{\mu c_p}{Pr}, \tag{16}$$

in which c_p is the heat capacity of the mixture, and Pr is the Prandtl number, taken as constant. The molecular diffusivity of species α is given by

$$D_\alpha = \frac{\lambda}{\rho c_p Le_\alpha}, \tag{17}$$

where Le_α is the Lewis number for species α, which is assumed constant for each species.

C. Evaluation of reaction rates

The reaction mechanism is given by M equations written in the form

$$\sum_\alpha \nu'_{\alpha,m} \mathcal{M}_\alpha \rightarrow \sum_\alpha \nu''_{\alpha,m} \mathcal{M}_\alpha \qquad \forall m \in [1, M], \tag{18}$$

where $\mathcal{M}_\alpha$ symbolises the chemical symbol of species α, and $\nu'_{\alpha,m}$ and $\nu''_{\alpha,m}$ are the molar stoichiometric coefficients of species α in equation m.

The reaction rate of species α is given by

$$\omega_\alpha = W_\alpha \sum_{m=1}^{M} \hat{\omega}_{\alpha,m} \tag{19}$$

where $\hat{\omega}_{\alpha,m}$ is the molar production rate of species α due to the m-th reaction, and M is the total number of reactions. The molar production rates are given by

$$\hat{\omega}_{\alpha,m} = \left(\nu''_{\alpha,m} - \nu'_{\alpha,m}\right) \left[k_{f,m}(T) \prod_{\beta=1}^{N_S} \left(\frac{\rho Y_\beta}{W_\beta} \right)^{\nu'_{\beta,m}} - k_{b,m}(T) \prod_{\beta=1}^{N_S} \left(\frac{\rho Y_\beta}{W_\beta} \right)^{\nu''_{\beta,m}} \right], \tag{20}$$

in which $k_{f,m}$ and $k_{b,m}$ are the forward and backward (respectively) rate constants for reaction m. The forward rate constant is given by the Arrhenius expression

$$k_{f,m}(T) = A_m T^{n_m} \exp\left(\frac{E_{a,m}}{R_0 T} \right), \tag{21}$$

with A_m, n_m and $E_{a,m}$ being specified constants representing the pre-exponential factor, the temperature dependence of the pre-exponential factor, and the activation energy.

1. Backwards rate constant

The backwards rate constant is related to the forward rate constant via

$$\ln k_{b,m} = \ln k_{f,m} + \sum_{\alpha} \left(\nu''_{\alpha,m} - \nu'_{\alpha,m}\right) \left(\frac{\hat{g}_\alpha}{R_0 T} + \ln\left(\frac{P_0}{R_0}\right) - \ln T\right), \tag{22}$$

in which P_0 is a reference pressure, and $\hat{g}_\alpha$ is the Gibbs function for species α, which is given by

$$\frac{\hat{g}_\alpha}{W_\alpha T} = \frac{a_{\alpha,J+1}}{T} - a_{\alpha,1} \ln T + (a_{\alpha,1} - a_{\alpha,J+2}) - \sum_{j=2}^{J} \frac{a_{\alpha,j}}{j\,(j-1)} T^{j-1}. \tag{23}$$

2. Third bodies

For a step m including third bodies $\mathcal{M}$, reaction rates are scaled by the third-body concentration $c_{M,m}$, which is given by

$$c_{M,m} = \sum_{\alpha} \eta_{\alpha,m} \frac{\rho Y_\alpha}{W_\alpha}, \tag{24}$$

and $\eta_{\alpha,m}$ are the third-body efficiencies for species α for step m.

3. Lindemann forms

Where the Arrhenius form is insufficient to describe the reaction rate, a pressure (partial pressure/concentration) dependancy is introduced, and the forward reaction rate takes the Lindemann form:

$$k_{f,m} = k_{\infty,m} \left(\frac{P_r}{1 + P_r}\right) F_m, \tag{25}$$

where F_m is the Troe fall-off rate, and a constant for steps with Lindemann form. P_r is a reduced pressure given by

$$P_r = \frac{k_{0,m}}{k_{\infty,m}}, \tag{26}$$

$k_{0,m}$ is the Arrhenius rate of the reaction, and $k_{\infty,m}$ is a second Arrhenius rate. Hence for Lindemann steps, in addition to the three Arrhenius coefficients, the constant F_m is prescribed, along with the pre-exponential factor, the temperature dependence of the pre-exponential factor and the activation energy for $k_{\infty,m}$. Note that after evaluation of the rate constants for Lindemann steps, the third-body concentration of the step is reset to unity.

III. NUMERICAL IMPLEMENTATION

A. Basic framework and principles

The domain is discretised with an unstructured node-set. Each node (or collocation point) i has position $\boldsymbol{r}_i$, discretisation length-scale s_i (the average local inter-node spacing), and holds the primary variables $\ln \rho_i$, $\boldsymbol{u}_i$, ρE_i. The governing equations are solved on the set of nodes. Spatial derivatives are evaluated using LABFM [1] in the first two dimensions (x, y) and high-order finite differences (FD) in the third dimension (z). Technically, the node-set is semi-unstructured...

Time integration is by third-order Runge-Kutta scheme. The solution is de-aliased with a high-order LABFM+FD filter after each time step [1]. The accuracy of the spatial discretisation can be changed through compiler flags, between 4^{th} and 12^{th} order accurate in space. In typical usage, 8^{th} order provides a good compromise between accuracy and cost.

At wall, inflow and outflow boundaries, the node-set is structured, with a band of 5 rows of uniformly spaced nodes. Wall boundaries may be curved, but *at present,* must have uniform resolution along their length. On boundaries

(and within the first 2 rows of uniform nodes) the derivatives and filters in the $x-y$ plane are calculated through combinations of FD (normal to boundaries), one-dimensional LABFM (tangential to boundaries), and 2D LABFM. Details in [1].

Aside from the complexities of indexing and derivative evaluation associated with the unstructured node-set approach, the framework is similar to a high-order (central) finite difference scheme.

B. Evaluation of derivatives

Derivatives of the conservative variables ρ, ρu_i, ρE and ρY_α, and also the temperature and pressure, are evaluated directly with the LABFM method [1] in the $x-y$ plane, and central finite differences in z. To avoid the explicit evaluation of cross-second derivatives of velocity, the viscous stress divergence is evaluated as

$$\frac{\partial \tau_{ki}}{\partial x_k} = \frac{1}{\mu}\frac{\partial \mu}{\partial x_k}\tau_{ki} + \mu\left(\frac{\partial^2 u_i}{\partial x_k \partial x_k} + \frac{1}{3}\frac{\partial^2 u_k}{\partial x_i \partial x_k}\right), \tag{27}$$

where the final term is the gradient of the velocity divergence, which is calculated through two iterations of first derivative evalations. Laplacians are also evaluated directly using LABFM combined with finite differences. This formulation permits smaller computational stencils, and reduces the amount of MPI communication required.

The evaluation of derivatives using LABFM constitutes the bulk of the computational cost, and it is hence desirable avoid where possible. Gradients of secondary properties are evaluated from analytic expressions in terms of gradients of primary properties as follows

$$\frac{\partial \mu}{\partial x_k} = \frac{r_T \mu}{T}\frac{\partial T}{\partial x_k} \tag{28}$$

$$\frac{\partial \lambda}{\partial x_k} = \frac{r_T \lambda}{T}\frac{\partial T}{\partial x_k} + \frac{\lambda}{c_p}\frac{\partial c_p}{\partial x_k} \tag{29}$$

$$\frac{\partial \rho D_\alpha}{\partial x_k} = \frac{r_T D_\alpha}{T}\frac{\partial T}{\partial x_k} \tag{30}$$

The gradient of the enthalpy of species α is given by

$$\frac{\partial h_\alpha}{\partial x_k} = \frac{dh_\alpha}{dT}\frac{\partial T}{\partial x_k} = c_{p,\alpha}\frac{\partial T}{\partial x_k}, \tag{31}$$

and the gradient of the mixture specific heat capacity is

$$\frac{\partial c_p}{\partial x_k} = \sum_\alpha \left[Y_\alpha \frac{dc_{p,\alpha}}{dT}\frac{\partial T}{\partial x_k} + c_{p,\alpha}\frac{\partial Y_\alpha}{\partial x_k}\right]. \tag{32}$$

C. Boundaries

Symmetry and periodic boundary conditions are available. The only boundary condition available in the third dimension is periodic. Wall, inflow and outflow boundaries are imposed through the Navier-Stokes characteristic boundary condition formalism of [citeXX]. The basic framework follows that described for isothermal flows in [1].

As mentioned above, we (almost completely) avoid the explicit evaluation of cross-second derivatives (e.g. $\partial^2\phi/\partial x\partial y$). The only exception is the terms $\partial^2 u/\partial x\partial z$ and $\partial^2 u/\partial x\partial y$ on outflow boundaries.

For inflows and outflows (always taken to be in the x direction at present), we add

$$\mu\left[\frac{2}{3}\frac{\partial}{\partial x}(\nabla\cdot\boldsymbol{u}) - 2\frac{\partial^2 u}{\partial x^2}\right] - \frac{\nabla\mu}{\mu}\cdot\boldsymbol{\tau}\cdot\boldsymbol{e_x} \tag{33}$$

to the first component of the expression in (27), to impose $\frac{\partial \tau_{xx}}{\partial x} = 0$. For outflows, we also impose $\frac{\partial \tau_{yx}}{\partial x} = 0$, by adding

$$\mu\left[-\frac{\partial^2 v}{\partial x^2} - \frac{\partial^2 u}{\partial x \partial y}\right] - \frac{1}{\mu}\frac{\partial \mu}{\partial x}\tau_{xy} \tag{34}$$

to the second component, and $\frac{\partial \tau_{zx}}{\partial x} = 0$, by adding

$$\mu \left[-\frac{\partial^2 w}{\partial x^2} - \frac{\partial^2 u}{\partial x \partial z} \right] - \frac{1}{\mu} \frac{\partial \mu}{\partial x} \tau_{xz} \tag{35}$$

to the third component.

D. Parallelisation

The code is accelerated with a shared-distributed framework, using both OpenMP and MPI. It should be noted that the parallel efficiency of the OpenMP is low, due to memory-access issues. Provided sufficient memory is available, it is therefore recommended to run the code with very few threads per MPI rank: ideally 1, but any more than ≈ 5 and resources are being used inefficiently.

Domain decomposition is three-dimensional, and *currently* static. The total number of MPI ranks is $N_p = N_{px} N_{py} N_{pz}$, where the subscripts x, y and z indicate the number of ranks in each direction. The physical sizes of the subdomains are adjusted such that each MPI rank contains the same number of collocation points, resulting in approximate load balancing. On simple domains the parallel efficiency is $> 96\%$ at 1024 MPI ranks (relative to 4 MPI ranks).

The domain-decomposition is likely to be restructured in the coming year, so I'll leave further details for now.

IV. GENERATING INPUT FILES

THIS SECTION TBC. CURRENTLY BASED ON OLD BUBBLY CODE

Input files are generated using the program `gen2d`, found in `source/gen/`. Navigate here then

```
make
./gen2d
cp I* ../../.
```

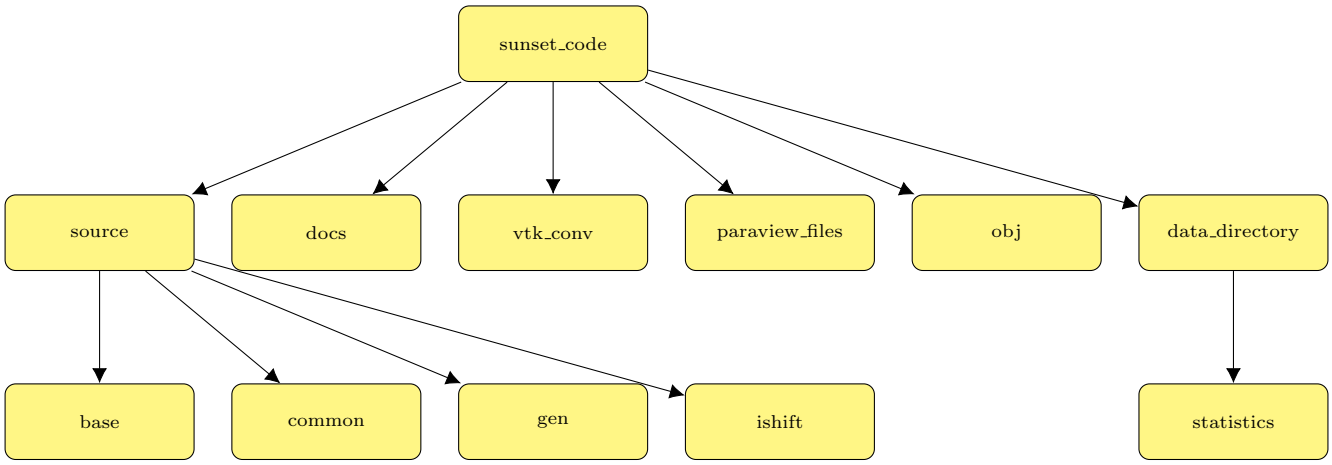
When `gen2d` is run, a list of cases is displayed. Input a number to specify a case. Once generated, you can use the command `cp I* ../../.` to copy the case files `INDAT`, `IPART`, and `IBOUND` to the main directory. Cases are hard-coded in the program `gen2d`. To modify them, or create new cases, you need to modify the file `datclass.F90`, and recompile with `make`.

Within each case in `datclass.F90`, domain sizes (`xl`, `yl`, and `h0`) are specified, along with the resolution. Then dimensionless parameters are specified. Followed by boundary conditions (described below), and then the specification of the initial particle distribution. Note that entire problem is non-dimensionalised, so the domain specification and intitial velocity fields should have characteristic scales of unity. Also note that `gen2d` generates two-dimensional case files, and a parameter `nz` which specifies the number of particles in the third dimension (if a three-dimensional simulation is run).

The following cases are currently included in `datclass.F90`:

1. Still water/bubble column
2. Dam Break (Lobovsky)
3. Taylor-Green
4. Poiseuille/channel flow
5. Third order Stokes wave

A. Boundary framework



- `source` contains source files for the code:
  - `base` contains main modules for `sunset_code`
  - `para` contains parameters and common variables
  - `gen` contains code to generate casefiles and initial conditions
- `vtk_conv` contains a program to convert output files into `.vtu` files, which can be read by Paraview.
- `paraview_files` the program in `vtk_conv` creates `.vtu` files here.
- `obj` contains `.o` and `.mod` files created during compilation.
- `docs` contains this document and associated files.
- `data_directory` contains output files produced by the code. `statistics` contains files describing global, time-varying measures (e.g. total K.E.).

Appendix A: Equations written out in full, in Einstein notation

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0 \tag{A1}$$

$$\frac{\partial \rho u_i}{\partial t} + \rho u_k \frac{\partial u_i}{\partial x_k} + u_i \frac{\partial \rho u_k}{\partial x_k} = -\frac{\partial p}{\partial x_i} + \mu \left[\frac{\partial^2 u_i}{\partial x_k \partial x_k} - \frac{1}{3} \frac{\partial}{\partial x_i} \left(\frac{\partial u_k}{\partial x_j} \right) \right] + \left[\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i} - \frac{2}{3} \delta_{ik} \frac{\partial u_j}{\partial x_k} \right] \frac{\partial \mu}{\partial x_j} + \rho f_i \tag{A2}$$

$$\frac{\partial \rho E}{\partial t} + u_i \frac{\partial \rho E}{\partial x_i} = - \rho E \frac{\partial u_i}{\partial x_i} - u_i \frac{\partial p}{\partial x_i} - p \frac{\partial u_i}{\partial x_i} \tag{A3}$$

$$+ u_i \left\{ \frac{\mu}{\rho} \left[\frac{\partial^2 u_i}{\partial x_j \partial x_j} - \frac{1}{3} \frac{\partial}{\partial x_i} \left(\frac{\partial u_j}{\partial x_j} \right) \right] + \frac{1}{\rho} \left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \delta_{ij} \frac{\partial u_k}{\partial x_k} \right] \frac{\partial \mu}{\partial x_j} \right\} \tag{A4}$$

$$+ \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \delta_{ij} \frac{\partial u_k}{\partial x_k} \right) \frac{\partial u_i}{\partial x_j} \tag{A5}$$

$$+ \sum_\alpha \left\{ \rho D_\alpha \frac{\partial Y_\alpha}{\partial x_i} \frac{\partial h_\alpha}{\partial x_i} + h_\alpha \left[\rho D_\alpha \frac{\partial^2 Y_\alpha}{\partial x_i \partial x_i} + \frac{\partial \rho D_\alpha}{\partial x_i} \frac{\partial Y_\alpha}{\partial x_i} \right] \right\} \tag{A6}$$

$$- \sum_\alpha Y_\alpha h_\alpha \sum_\beta \left[\rho D_\beta \frac{\partial^2 Y_\beta}{\partial x_i \partial x_i} + \frac{\partial \rho D_\beta}{\partial x_i} \frac{\partial Y_\beta}{\partial x_i} \right] \tag{A7}$$

$$- \sum_\alpha \left[h_\alpha \frac{\partial Y_\alpha}{\partial x_i} + Y_\alpha \frac{\partial h_\alpha}{\partial x_i} \right] \sum_\beta \rho D_\beta \frac{\partial Y_\beta}{\partial x_i} \tag{A8}$$

$$+ u_i f_i \tag{A9}$$

and

$$\frac{\partial \rho Y_\alpha}{\partial t} + \rho u_i \frac{Y_\alpha}{\partial x_i} + Y_\alpha \frac{\partial \rho u_i}{\partial x_i} = \dot{\omega}_\alpha + \left[\rho D_\alpha \frac{\partial^2 Y_\alpha}{\partial x_i \partial x_i} + \frac{\partial \rho D_\alpha}{\partial x_i} \frac{\partial Y_\alpha}{\partial x_i} \right] \tag{A10}$$

$$- Y_\alpha \sum_\beta \left[\rho D_\beta \frac{\partial^2 Y_\beta}{\partial x_i \partial x_i} + \frac{\partial \rho D_\beta}{\partial x_i} \frac{\partial Y_\beta}{\partial x_i} \right] - \frac{\partial Y_\alpha}{\partial x_i} \sum_\beta \rho D_\beta \frac{\partial Y_\beta}{\partial x_i} \tag{A11}$$

---

[1] J. King, S. Lind, High-order simulations of isothermal flows using the local anisotropic basis function method (labfm), Journal of Computational Physics 449 (2022) 110760. doi:doi:https://doi.org/10.1016/j.jcp.2021.110760.

[2] J. King, S. Lind, A. Nasar, High order difference schemes using the local anisotropic basis function method, Journal of Computational Physics 415 (2020) 109549. doi:doi:10.1016/j.jcp.2020.109549.