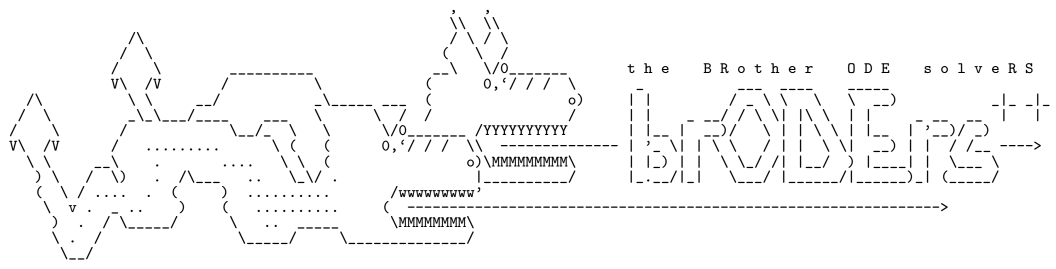
brODErs++ manual

ODE solvers for multi-temperature reacting flows.



- Shocking++
- LARSEN
- MultiLARSEN

Stefano BOCCELLI

AUGUST 2017

Contents

1 Introduction

This document describes the `brODErs++` set of solvers for multi-temperature reacting flows: `Shocking`, `LARSEN` and `MultiLARSEN`. Note that:

- This document refers to `brODErs++` v1.0;
- Research codes are by definition *work-in-progress*. You may need to tweak it a bit;
- The best documentation is the code itself.

1.1 Solvers overview

Shocking

Computes the thermochemical relaxation past a shock wave. Free-stream conditions are specified as input. First of all solves the Rankine-Hugoniot relations to find the post-shock state, then the Euler equations are solved for a chemically reacting flow.

LARSEN

This is a lightweight tool to introduce detailed thermo-chemical models in previous baseline solutions. `LARSEN` reprocesses the solution along a streamline, recomputing the concentration of chemical species according to the specified chemical model. Another possibility is starting from a baseline thermal equilibrium simulation and introducing internal temperature nonequilibrium (T-Tv model).

MultiLARSEN

MultiLARSEN is not included in this release. Contact the authors if you wish to obtain a copy (for example at: `stefano.boccelli -AT- polimi -DOT- it`).

The same as `LARSEN` but for multiple streamlines simultaneously, with mass and energy diffusion among them.

1.2 Citing the software

If you publish results obtained with this software, please cite:

- For the Lagrangian solver `LARSEN` (or `Multi-LARSEN`): Boccelli, S., Bariselli, F., Dias, B., & Magin T. E., *Lagrangian diffusive reactor for detailed thermo-chemical computations of plasma flows*, PSST, 2019.
- And for `Shocking`: Magin, T. E., Caillault, L., Bourdon, A., & Laux, C. O. (2006). *Nonequilibrium radiative heat flux modeling for the Huygens entry probe*, Journal of Geophysical Research: Planets, 111(E7).

2 Compiling and running

The `brODErs++` suite contains all the solvers. You just have to compile `brODErs++` and you will choose the desired solver at run time via the input file.

2.1 Requirements

Mutation++ library

The VKI `Mutation++` library needs to be installed. Mind that different versions of `Mutation++` may lead to slightly different results (although they should not). You'll have to specify the installation path to `cmake` during the configuration process.

Boost library

A version of the `Boost` library bigger than 1.56 is needed. You must pass its PATH to `cmake` during the configuration process. If you don't know where they are on your system or whether you have them or not, just download them, untar and place in a known location. Then pass to `cmake` that location, for example:

```
/home/user/my_libraries/boost_1_54_0
```

The version 1.54 runs as a charm... while subsequent versions are much slower, also throwing some segfaults while running `brODErs++`.

VTK library

An optional dependence is the VTK library, needed only for `MultiLARSEN`. Under Debian you can install the `libvtk6-dev` package: this should do the job. On Ubuntu it should be similar I guess. Otherwise, download the VTK package, compile it and place in a standard system location where CMake can find this.

2.2 Notes: other external libraries directly included

The `brODErs++` suite additionally uses the `Eigen3` library which is included from the `Mutation++` installation and you don't need to install it on your system.
Eigen3: is taken from the `Mutation++` installation. In case `Mutation++` will upgrade to a future version `Eigen4` or so, all one have to do is updating the `brODErs++` `CMakeLists.txt` file.

2.3 Compiling

`brODErs++` is compiled using CMake. The idea is that the building process is located into a directory “`build`”: if you want to clean up and restart the compiling process all you have to do is removing the `build` folder and starting again.

After uncompressing the tarball, go in the main directory of `brODErs++` (where you can see the directories `src`, `doc`, `input` etc), remove the `build` folder if existing and run the following:

```
$ mkdir build
$ cd build
$ ccmake ..
```

This will call the ncurses based version of cmake. If you see no options, type configure `[c]`. The options are:

BOOST_DIRECTORY - put here the directory of the `boost` library. Example: `/home/user/boost/boost_1_54_0`;

CMAKE_BUILD_TYPE - put `Release` or `Debug`. “`Release`” is fine if you are not a developer.

CMAKE_INSTALL_TYPE - automatically set (`/usr/local`);

MPP_INSTALL_DIR - put here the `install` directory of your `Mutation++` installation. Example: `/home/user/Mutation++/install`;

VTK_SUPPORT - enable this if you have the VTK libraries installed on your system. This is required by the `MultiLARSEN` solver.

Fill the fields by pressing `ENTER`, then press configure `[c]` and generate `[g]`. At this point the configuration is complete. To compile the program run:

```
$ make install
```

This will compile `brODErs++` and copy its executable in the local `bin` directory. You’re ready to go.

2.4 Running

Once `brODErs++` is compiled you can just copy the generated binary file (located in the `bin` directory) in your working directory and run:

```
$ ./brODErs++ input_file
```

where the `input_file` structure is explained in the solvers sections.

Otherwise, you can add the `brODErs++/bin` directory to the `PATH` variable, by adding the following line to the `~/.bashrc` file: [1]

```
 export PATH=$PATH:/path-to-brODErs++/brODErs++/bin
```

[1]Remember to close and open again the terminal for changes to take place, or just run the `.bashrc` script with: "`. ~/.bashrc`"

3 Shocking

3.1 Overview

`Shocking` computes the thermochemical relaxation past a 1D normal shock wave. Free-stream conditions are given in terms of pressure, temperature and velocity. The Rankine-Hugoniot conditions are invoked for computing the post-shock conditions, then the **Euler equations** are solved for a chemically reacting gas.

The governing equations are described in the PhD thesis of Munafò [1] and Panesi [2]. A number of versions of `Shocking` are present at VKI. This version **does not support radiation**.

Currently, `Shocking` supports the one-temperature model (1T) and two temperatures model (T-Tv). The implementation should be ready for the N-temperatures case as well.

1T case

In this case, T = Tr = Tv = Tel = Te. The free-stream conditions are given via input file in terms of pressure, temperature and velocity. The Rankine-Hugoniot relations are used to compute the post-shock conditions. Additionaly, a sub-relaxation is performed to impose that the translational and internal energies are in equilibrium in the post-shock region. The chemical species composition is assumed to be frozen across the shock: relaxation starts in the post-shock region.

T-Tv case

In this case, T = Tr, Tv = Tel = Te. As for the 1T case, the freestream conditions are given via the input file in terms of pressure, temperature and velocity. The Rankine-Hugoniot relations are used to compute the post-shock conditions. The translational temperature is set equal to the post-shock temperature value, while the internal temperature Tv is kept frozen across the shock (equal to the pre-shock condition).

3.2 Input file

In the following an input file for `Shocking`. You can put comments (hash sign #) and blank lines wherever you wish in the input file. Instructions are made by a heading line (example: “FS Temp:”), followed by a line specifying the actual value (example: “100”). The order of instructions does not matter.

```
# ---------------   Mixture options   ---------------------
Problem type:
shocking

Name of the mixture:
air5

State model:
ChemNonEq1T

Thermodynamic database:
RRHO

# ----------------   Output parameters   ------------------
Species output type: # the default value is mass_fraction
mole_fraction

Print sol each N steps: # this is optional. Default is 1
5

# ---------------   Free-stream conditions   --------------
FS Press: # [Pa]
12.3

FS Temp:  # [K]
100

FS Vel:   # [m/s]
10000

# ---------------   Mesh parameters   ---------------------
Mesh X_init:  # [m]
0.0

Mesh X_final: # [m]
5.0e-3

Mesh dX:      # [m]
1.0e-8
```

The fields are:

Problem type: - can be shocking or larsen or multilarsen.

Name of the mixture: - can be anything defined in the `data/mixtures` directory of your `Mutation++` installation.

State model: - state models supported by `Mutation++`. Has been succesfully tested with `ChemNonEq1T` and `ChemNonEqTTv`.

Thermodynamic database: - thermodynamic database supported by `Mutation++`. Currently, can be `RRHO`, `NASA-7` or `NASA-9`.

Species output type: - specifies how the concentration of chemical species should be given: `mass_fraction` or `mole_fraction`.

Print sol each N steps: - optional.

FS Press: - freestream pressure, in Pascal.

FS Temp: - if 1T model: freestream temperature in Kelvin. If N-temperature models: put one temperature after the other in the same line, separated by spaces.
Example for T-Tv model: specifying "`220 130`" gives a freestream translational temperature of `220` K and a vibrational one of `140` K.
..Probably this setting is useless since the free stream is usually in equilibrium..

FS Vel: - freestream velocity, in m/s.

Mesh X_init: - initial point of the 1D grid.

Mesh X_final: - final point of the 1D grid (must be bigger than the initial one).

Mesh dX: - spacial step for the integration.

3.3 Output

`Shocking` prints the output on the screen. You can simply redirect it to a file using:

```
$ brODErs++ shocking_input_file > shocking_output
```

The output of `Shocking` consists in:

- Position;
- Chemical species (mass fraction or mole fraction);
- Velocity;
- Translational temperature;
- Internal temperature (for `ChemNonEqTTv` model).

`Shocking` writes the solution starting with "`Sol:`", so that you can easily grep it from the output:

```
$ brODErs++ shocking_input_file | grep "Sol:"
```

This is an example of the output for the input file shown above:

```
Loading Input file 'input_shocking.in'

Input file was read:
    Problem type: shocking
    Mixture: air5
    State model: ChemNonEq1T
    Thermodynamic database: RRHO

             -------
            (______()     ___ _  The       _   _               \W_W/
     \W/___(______()()   / __| |_  ___  __| |_(_)_ _  __ _     /o o\
     /M\   (__TNT_()()   \__ \ ' \/ _ \/ _| / / | ' \/ _` |  _( <   )_
            (______()    |___/_||_\___/\__|_\_\_|_||_\__, |  \\\ O ///
                                               Code  |___/   _))\_/((_
                                                            / /     \ \
                                                           /_/|     |\_\
Free-stream conditions:
    Press [Pa]: 12.3
    Temp [K]:   100
    Vel [m/s]:  10000
    Computed mass flux [kg/m2 s]: 4.26797
    Computed density [kg/m3]:     0.000426797

Mesh parameters:
    X_initial [m]: 0
    X_final [m]:   0.005
    dX [m]:        1e-08

Output settings:
    Species output type: mass_fraction
    Solution is printed each 5 steps.

======>  STARTING THE INTEGRATION  <======

Position x [m]      Y_N      Y_O      Y_NO      Y_N2      Y_O2      U [m/s]      T [K]
Sol: 6.84531e-07  0.00060414         ...............        1669.22650993  48188.26005209
Sol: 6.18367e-06  0.00585875         ...............        1662.48011099  47376.80682350
Sol: 1.32381e-05  0.01399921         .. et cetera ..        1652.59825438  46257.48470928
Sol: 1.98541e-05  0.02356588         .. et cetera ..        1641.51867570  45081.54352914
Sol: 2.57238e-05  0.03384775         ...............        1630.08370450  43928.89567356
Sol: 3.02003e-05  0.04283480         ...............        1620.52909112  42994.04941739
Sol: 3.66363e-05  0.05740202         ...............        1606.00360578  41597.81283569

... up to the end.
```

Output for T-Tv model

For the two-temperature model (`ChemNonEqTTv` state model), the output will be:

```
Position x [m]     Y_N     Y_O   ...   U [m/s]     T_0 [K]     T_1 [K]
Sol: ...
```

where the last two fields are respectively the translational temperature (T_0) and the internal one (T_1).

Post-processing tip (just to save one minute of googling)

Once you grep the lines starting with "Sol", you probably want to remove the first column for importing the data on Octave/MATLAB/... You can do it with `vim` (super easy) or you can run:

```
    $ cut -c 5- output_file > new_output_file
```

3.4 Density and pressure

Density is not explicitly given as output, since they can be easily obtained. The free-stream mass flow is printed as output by shocking, as can be seen in the previous output example:

```
Free-stream conditions:
    Press [Pa]: 12.3
    Temp [K]:   100
    Vel [m/s]:  10000
    Computed mass flux [kg/m2 s]: 4.26797
    Computed density [kg/m3]:     0.000426797
```

Since the mass flow $\dot{m}$ is conserved, the density in the post-shock relaxation region can be easily computed from the velocity (which is outputted by shocking): $\rho = \dot{m}/U$.

Finally, the pressure can be obtained using the perfect gas law[2]: For the multi-temperature case, remind that only the translational temperature and the free electrons temperature contribute to the pressure.

[2]Shocking in fact uses the perfect gas law. See Munafò [1].

4 LARSEN

4.1 Overview

`LARSEN` (LAgrangian Reactor for StrEams in Nonequilibtium) is a tool for introducing *a posteriori* a detailed thermo-chemical model in a previouly performed solution.

From a previous solution, one streamline has to be extracted: `LARSEN` takes the velocity and density as given along such streamline, plus the initial conditions in terms of temperature(s) and chemical composition. Then, chemical species and temperature(s) are re-computed according to the specified model. **Velocity and density are not recomputed, but taken as given.**
The workflow is as follows:

1. Run a baseline simulation, using CFD / DSMC / ... / analytical methods;
2. Extract a streamline, (`ParaView` / `Tecplot` / ...);
3. Format the values in the `LARSEN` accepted way;
4. Run `LARSEN`;
5. You have recomputed results along the streamline.

`LARSEN` solves one mass conservation equation for each chemical species (taking into account chemical reactions), plus one equation for the total enthalpy (giving the temperature) and one equation for the internal energy (if requested). A detailed explanation can be found in the thesis of Boccelli [3], with some testcases.

Fig. 1 shows a streamline extracted from the baseline solution. The solution must be given along it point by point (see section 4.3). `LARSEN` takes the initial solution from the initial point and integrates along the curvilinear abscissa s. The solution is not necessarily given at the same points that were given: `LARSEN` integrates in an adaptive way, automatically refining where the solution changes a lot. Note that from one point to the other one the streamline is supposed rectilinear: it is up to your to take the points reasonably close to each other and refined enough to reproduce reasonably well the baseline solution. Remember the principle: *Garbage In Garbage Out*.

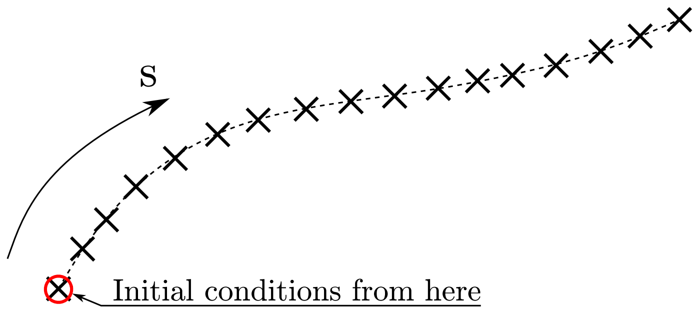


Figure 1: Streamline extracted from the baseline solution.

To sum up, you need two things to run `LARSEN`: an **input file** and a **baseline solution file** (storing the solution along the extracted streamline). `LARSEN` *will output the recomputed species* *concentrations and temperature(s), while the velocity (and consequently the density as well)* [3] *field is not recomputed, but is kept equal to the baseline solution.*

4.2 Input file

In the following, an input file for `LARSEN`. You can put comments (hash sign #) and blank lines wherever you wish in the input file. Instructions are made by a heading line (example: "`Total enthalpy diffusion:`"), followed by a line specifying the actual value (example: "`external`"). The order of instructions does not matter. Non requested instructions are simply ignored.

```
# ---------------   Mixture options ------------------
Problem type:
larsen

Name of the mixture:
air11

State model:
ChemNonEq1T

Thermodynamic database:
RRHO

# ---------------   LARSEN options   -----------------

External flowfield filename:
baseline_streamline.dat

Species input type: # default: mass_fraction
mole_fraction
#mass_fraction

Species output type: # default: mass_fraction
mass_fraction
#mole_fraction

Total enthalpy diffusion: # [adiabatic / external]
adiabatic

Diffusive mass fluxes: # [none / external_streamwise / external_stagline_sphere]
```

[3]The mass conservation equation only involves the flow velocity and density. Thus, if the velocity is prescribed from the initial solution, a unique density would be found by solving this equation, namely the baseline density. For this reason, if the velocity is prescribed, there is no point in recomputing the density also.

```
none
```

The fields are:

Problem type: - can be `shocking` or `larsen` or `multilarsen`.

Name of the mixture: - can be anything defined in the `data/mixtures` directory of your `Mutation++` installation. This is the *final* (recomputed) mixture.

State model: - state models supported by `Mutation++`. Has been succesfully tested with `ChemNonEq1T` and `ChemNonEqTTv`.

Thermodynamic database: - thermodynamic database supported by `Mutation++`. Currently, can be `RRHO`, `NASA-7` or `NASA-9`.

External flowfield filename: - name (and optionally path) of the file storing the baseline simulation streamline.

Species input type: - specifies how the concentration of chemical species is given in the baseline simulation file: `mass_fraction` or `mole_fraction`.

Species output type: - specifies how the concentration of recomputed chemical species should be outputted: `mass_fraction` or `mole_fraction`.

Print sol each N steps: - optional.

Total enthalpy diffusion: - the fluid particle can be supposed adiabatic, so that the total enthalpy is conserved (keyword “`adiabatic`”). Otherwise, the diffusion of total enthalpy can be taken from the baseline solution, by computing how much the total enthalpy changes from one streamline point to the next one (keyword “`external`”). This allows the solver to work in rarefied conditions as well. For further information, see [3].

Diffusive mass fluxes: - “`none`” or taken from the baseline simulation. In this case they must be given in the baseline simulation file (see next sections). For external mass flows use the keyword “`external_streamwise`”. It’s possible to use also the keyword “`external_stagline_sphere`”, computing diffusion fluxes in a “*stagnation line fashion*”. See Munafò [1] and Klomfass and Müller [4]. Two more parameters are required in this case.

4.3 Baseline solution file

The basic format for the baseline solution file is as follows. For thermal equilibrium computation (1 temperature T = Tr = Tv = Tel = Te):

```
s   T   rho   U   X1   X2   X3   ...   XN
```

For a two-temperautes model (T, Tv with T = Tr, Tv = Tel = Te):

```
s   T   Tv   rho   U   X1   X2   X3   ...   XN
```

where:

- `s` - curvilinear abscissa along the streamline [m];
- `T` - temperature [K];
- `Tv` - internal temperature [K] (if TTv model);
- `rho` - mixture density [kg/m3]
- `U` - velocity **module** [m/s];
- `Xi` - species concentrations, in the order required by `Mutation++`. Can be mass fractions or mole fractions, according to the input file.

Some notes

The velocity and density fields are taken as they are and not recomputed along the streamline. The temperature(s) and chemical species are taken as initial condition at the beginning of the streamline.

The **spacing** among baseline streamline points is not really critical, since `LARSEN` automatically sub-refines the step. Of course, you need to provide enough points for the physics to be reasonably followed.

4.3.1 Example

An example for the 1-temperature case. Say that you have a baseline CFD / DSMC / ... simulation of a N_2, O_2 mixture and you want to recompute it using a more complex mixture made of N_2, O_2, N, O and NO (called `air5`). First of all you need to know the order of the species for the thermodynamic library. You can run the `Mutation++` tool `checkmix`:

```
 $ checkmix air5
```

The output will be something like:

```
Gas Species (5):
   1: N
   2: O
   3: NO
   4: N2
   5: O2
```

Then you have to create the baseline streamline file, specifying the mixture composition. The composition that you have to specify is the baseline composition: only N_2 and O_2 (species IDs 4 and 5) were present, while all the other species were zero (IDs 1, 2 and 3). Thus, the file would read (random values..):

```
 0.0  1000  0.003  5000  0.0  0.0  0.0  0.79  0.21
 0.1   988  0.002  5200  0.0  0.0  0.0  0.79  0.21
 0.2   977  0.001  5400  0.0  0.0  0.0  0.79  0.21
 ...
```

After the position (1st column), temperature (2nd column), density (3rd column) and velocity (4th column) we have the species, as said above.

Recall that mass/mole fractions sum up to one. Also note that you can't put anything but numbers (no headers or comments), but as many spaces as you want among them.

Mass diffusion

`LARSEN` is not able to *compute* diffusion along the streamline [4]. However, it's possible to *import* mass diffusion from the baseline simulation, with the input file flag "`external_streamwise`". For doing this, you need to specify the diffusion velocity of each species in the baseline streamline file, right after the species concentration and in the same format. For a 1-temperature simulation this would read:

```
s   T   rho   U   X1   X2  ...  XN   Vd1   Vd2  ...  VdN
```

This option might be used for example to compare `LARSEN` results with other software that also computes diffusion. Otherwise you could use this option to move a baseline simulation (that includes diffusion) from 1 temperature to 2 temperatures (T, Tv).

Stagline simulations

In order to reproduce stagnagion line simulations (Munafò [1] and Klomfass and Müller [4]), two more parameters are needed, namely the radius `r` and the "vertical velocity" parameter `V` [5]. Only the sphere case is now supported (cylinder not yet) and can be done with the input file flag "`external_stagline_sphere`".

Those parameters can be specified as the *last parameters* in the baseline streamline file. For example, for a 1-temperature simulation with diffusion:

```
s   T   rho   U   X1   X2  ...  XN   Vd1   Vd2  ...  VdN   R   V
```

4.3.2 Mass and energy diffusion in TTv

Suppose you want `LARSEN` to include mass and / or energy diffusion in the framework of a two-temperature TTv model (using the `external` flag in the input file).

If you specify in the input file a two-temperature thermal model (`ChemNonEqTTv`), then you shall also specify two baseline temperature columns in the baseline solution file. If you specify the `ChemNonEqTTv` model you need to specify two temperature columns in the baseline solution file, even if your baseline solution was a single-temperature solution.[6] Clearly, if the baseline solution is equilibrium, you will put in those temperture columns the same numerical values.

Let us recall a few points about the TTv modeling in `LARSEN`:

[4]Due to the marching approach, centered derivatives are not possible and only upwinding would be an option. However diffusion should be computed with a centered scheme.

[5]This vertical velocity is a mathematical artifice.

[6]No wonder this could be made more intuitive or authomatized in the code.

1. The temperatures values at the beginning of the streamline are taken as initial condition for the new `LARSEN` simulation. In each case.
2. Energy diffusion implies computing the baseline simulation enthalpy point by point. This means that `LARSEN` needs to know **both** the translational and internal temperatures point by point along the baseline streamline. These values are imported and fed to the thermodynamic library, that will feed the value of the enthalpy and energy diffusion will be computed. If the baseline simulation is single-temperature, just provide two columns with the same values.
3. Mass diffusion implies a transport of energy together with the diffused species, which can be translational or internal energy in the TTv framework. For this reason, again, if mass diffusion is enabled, you need to specify two columns in the baseline solution file, one for the translational and one for the internal temperature.
4. If you are running a LARSEN computation with **NO** mass or energy diffusion, the value of the translational and internal temperatures *along* the streamline do not matter and only the initial ones are actually used. Avoid to put crazy or negative values anyway, not to make `LARSEN` go crazy. Use standard values to be sure.

Let us stress the procedure to start from single-temperture simulation and refine it, moving to TTv. First of all, in the *input file* you should specify the `ChemNonEqTTv` thermal model, then in the *baseline solution file* you should put two temperature columns, that represent the translational and internal tempertures in the baseline solution. Since we are dealing with the case “single-temperature” baseline solution, those two columns will be equal among each other.

In any case, look at the governing equations in reference [3] and read the source code if you want to be sure how stuff works.

4.4 Output

`LARSEN` prints the output on the screen. You can simply redirect it to a file using:

```
$ brODErs++ larsen_input_file > larsen_output
```

The output of `LARSEN` consists in:

- Position;
- Chemical species (mass fraction or mole fraction);
- Translational temperature;
- Internal temperature (for `ChemNonEqTTv` model).

`LARSEN` writes the solution starting with “`Sol:`”, so that you can easily grep it from the output:

```
$ brODErs++ larsen_input_file | grep "Sol:"
```

This is an example of the output for the input file shown above:

```
Loading Input file ’input_larsen.in’

Input file was read:
    Problem type: larsen
    Mixture: air11
    State model: ChemNonEq1T
    Thermodynamic database: RRHO


      /\                                _______ ___
     /  \         ____________          \      \  /
    /    \       /            \____      \      \/0_______        __   ___   ___  _________  __
    V\  /V      /   .........      \     (      0,‘/ / /  \      / /  / _ | / _ \/ __/ __/ |/ /
      \ \     __\    .       ....   \    (                o)    / /__/ __ |/ , _/\ \/ _//    /
       ) \   /  \)   .  /\___   ..   \__/ .                |   /____/_/ |_/_/|_/___/___/_/|_/
       (  \ / ....  .  (     )  ..........       /wwwwwwwww’
        \  v .  _ ..    )    (   ..........     (  ---------------------------------------------->
         ). .  / \_____/      \   ..  _____      \MMMMMMMM\   LAgrangian Reactor for StrEams
         \ .  /                \_____/     \______________/                     in Nonequilibrium
          \__/


LARSEN settings from input file:
    External flowfield filename: baseline_streamline.dat
    Species input type: mole_fraction
    Total enthalpy diffusion: adiabatic
    Diffusive mass fluxes: none

Output settings:
    Species output type: mole_fraction
    Solution is printed each 1 steps.

======>  STARTING THE INTEGRATION  <======

Curvilinear Abscissa s [m]    X_e-    X_N   ...   X_O2+    X_NO+    T [K]
 LARSEN - Step 0 of 199
Sol: 0.0255  0.0069348  0.34263  ............... 0.00021284  0.00026634  30483.41
Sol: 0.0256  0.0069348  0.34263  .. et cetera .. 0.00021284  0.00026634  30483.40
Sol: 0.0257  0.0069348  0.34263  .. et cetera .. 0.00021284  0.00026634  30483.40
Sol: 0.0258  0.0069348  0.34263  ............... 0.00021284  0.00026634  30483.38

... up to the end.
```

Output for T-Tv model

For the two-temperature model (`ChemNonEqTTv` state model), the output will be:

```
Curvilinear Abscissa s [m]    X_e-    X_N   ...   X_NO+    T_0 [K]    T_1 [K]
Sol: ...
```

where the last two fields are respectively the translational temperature (`T_0`) and the internal one (`T_1`).

Post-processing tip (just to save one minute of googling)

Once you grep the lines starting with “`Sol`”, you probably want to remove the first column for importing the data on Octave/MATLAB/... You can do it with `vim` (super easy) or you can run:

```
    $ cut -c 5- output_file > new_output_file
```

4.5 Verification testcases

A few verification testcases are available for `LARSEN` in the directory:
`brODErs++/verification/LARSEN`. **After each modification you make to `LARSEN`, you should run those testcases to check that you did not introduce some weird behavior in the solver.**

Some baseline simulations were performed with `Shocking`, `Stagline` and a DSMC method (`SPARTA`) and the result was recomputed by `LARSEN` providing good accuracy. Some plots obtained with the working code are provided in the folders. Running the testcases runs the current `LARSEN` executable on the baseline solutions, so that you can check how well it behaves.

Go inside the desired testcase and run the `./run_testcase.sh` script. This will run `LARSEN` on a given baseline solution.

Note that the result will strongly depend on the version of `Mutation++` that you are corrently using. Use this as a simple benchmark to verify your new implementation.

5 MultiLARSEN

5.1 Overview

`MultiLARSEN` (Multi - LAgrangian Reactor for StrEams in Nonequilibtium) is the multi- streamline version of `LARSEN`. `MultiLARSEN` allows to recompute a baseline simulation by adding a more elaborated chemical model.

The solver takes as input a number of streamlines from the baseline simulation and recomputes the chemical species and the temperature, additionally computing the diffusion of energy and chemical species across neighboring streamlines. `MultiLARSEN` **does not recompute the velocity (and density) fields. See the `LARSEN` section for more insight.**

The solver was originally created for flows mainly developed in one direction, such as the flow in the trail of a high speed meteoroid, where temperature and concentration gradients along the streamlines are much smaller than transverse gradients. In this case, diffusion of heat and energy are mainly transverse and the "purely transverse diffusion" hypothesis works well.

Due to the streamlines coupling procedure, the integration is carried along the x axis. Streamlines can be non-parallel, but they **can not** become perpendicular to the x axis or the solver will get crazy. For analyzing generic streamlines, use `LARSEN` instead. Of course, streamlines can't cross each other also. Fig. 2 shows some available and not available streamlines.

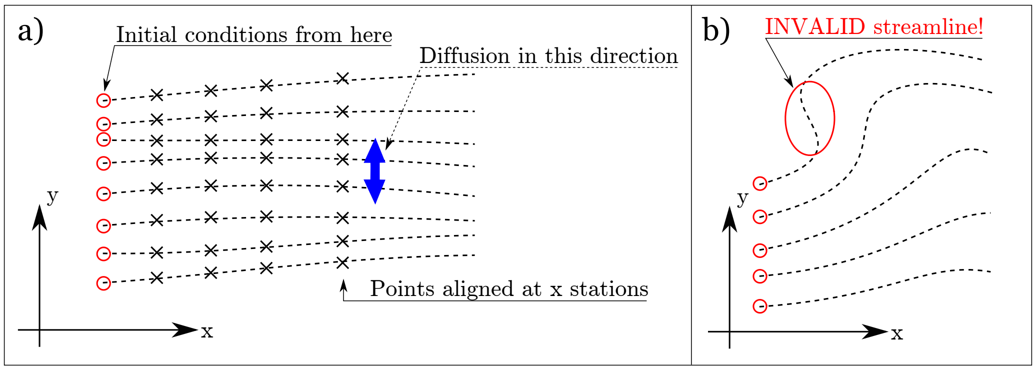

a)
Initial conditions from here
Diffusion in this direction
Points aligned at x stations
y
x
b)
INVALID streamline!
y
x

Figure 2: Case a) Valid streamlines: quite aligned and along x. Case b) A streamline reaches 90 degrees and is thus invalid. Moreover there is huge distortion.

To sum up:

- Streamlines can be non-parallel, but **can not** become perpendicular to the x axis;
- In case diffusion is to be computed, streamlines *should* be quasi parallel, for the "purely transverse diffusion" hypotesis to be meaningful;
- Steamwise mass diffusion is neglected anyway, so it's up to you using the diffusion solver in cases where this is hypothesis is meaningful.

From a previous solution, a number of streamlines have to be extracted: `MultiLARSEN` takes the velocity and density as given along such streamlines, plus the initial conditions in terms of temper-

ature and chemical composition. Then, chemical species and the temperature are re-computed according to the specified chemical model.
The workflow is as follows:

1. Run a baseline simulation, using CFD / DSMC / ... / analytical methods;
2. Extract some streamlines, (`ParaView` / `Tecplot` / ...);
3. Format the values in the `MultiLARSEN` accepted way;
4. Run `MultiLARSEN`;
5. `MultiLARSEN` outputs a `.vtp` file with the recomputed values (read it with `Paraview`).

`MultiLARSEN` solves one mass conservation equation for each chemical species (taking into account chemical reactions), plus one equation for the total enthalpy (giving the temperature). **Only 1-temperature model is currently supported** (`ChemNonEq1T`). A detailed description can be found in the VKI Research Master project report of Boccelli [5], together with some applications.

To sum up, you need two things to run `MultiLARSEN`: an **input file** and some **baseline solution files**, one per streamline.

5.1.1 Important notes

Streamlines **must** be synchronized along the x axis: the streamline points of all the streamlines need to be given at the same x values. Clearly, all the streamlines will have the same number of points. Refer to Boccelli [5] for more insight.

The numerical procedure is implicit, so that the number and packing of streamlines does not really matter, as long as there are enough streemlines to catch the flow features. You can still run some streamlines-number independence study for your case.

5.2 Input file

In the following, an input file for `MultiLARSEN`. You can put comments (hash sign #) and blank lines wherever you wish in the input file. Instructions are made by a heading line (example: "`Geometry:`"), followed by a line specifying the actual value (example: "`2D`"). The order of instructions does not matter.

```
# --------------   Mixture options -------------------
Problem type:
multilarsen

Name of the mixture:
air11

State model:
ChemNonEq1T

Thermodynamic database:
```

```
RRHO

# ---------------   MultiLARSEN options   -------------

Geometry: # [2D / axisymmetric]
axisymmetric

External flowfield filename:
str_

Streamlines number:
55

Species input type: # default: mass_fraction
#mass_fraction
mole_fraction

Species output type: # default: mass_fraction
number_density
#mass_fraction
#mole_fraction

Total enthalpy diffusion: # [adiabatic / external / computed]
computed

Diffusive mass fluxes: # [none / computed]
computed
```

The fields are:

Problem type: - can be `shocking` or `larsen` or `multilarsen`.

Name of the mixture: - can be anything defined in the `data/mixtures` directory of your `Mutation++` installation. It is the *final* (recomputed) mixture.

State model: - state models supported by `Mutation++`. Only `ChemNonEq1T` for now.

Thermodynamic database: - thermodynamic database supported by `Mutation++`. Currently, can be `RRHO`, `NASA-7` or `NASA-9`.

Geometry: - `2D` or `axisymmetric`, influences how divergence terms are computed for diffusion.

External flowfield filename: - name (and optionally path) of the files storing the baseline simulation streamlines. The streamlines names must be a string followed by 5 digits, starting from `00000` for the streamline closer to the x axis. Only the base string needs to be specified, while the digits are assumed by the solver. In the input file example, the specified string is "`str_`": the solver looks for files named "`str_00000`", "`str_00001`" and so on.

Streamlines number: - number of streamline files to be read. Say that you specify to read 39 streamlines, then the solver will try to load the files "`str_00000`" up to "`str_00038`".

Species input type: - specifies how the concentration of chemical species is given in the baseline simulation file: `mass_fraction` (default) or `mole_fraction`.

Species output type: - specifies how the concentration of recomputed chemical species should be outputted: `mass_fraction` (default) or `mole_fraction` or `number_density`. A VTK file called `MultiLarsenSol.vtp` will be produced in the working directory.

Print sol each N steps: - optional.

Total enthalpy diffusion:

- Keyword "`adiabatic`": fluid elements are supposed adiabatic, so that total enthalpy [7] is conserved. Since the velocity field is taken from outside and not recomputed, this may lead to unphysical results like temperatures going to 0 K.

- Keyword "`external`": the diffusion of total enthalpy is taken from the baseline solution, by computing how much the total enthalpy changes from one streamline point to the next one. This allows the solver to give good results even in rarefied conditions as well. If the flow changes significantly, you should use this keyword, since the others may give unphysical results[8], see [3] for more info.

- Keyword "`computed`": computes the heat fluxes across streamlines (in the y direction: transverse diffusion - not streamwise). Zero heat flux is imposed at the 0^{th} streamline (the one closer to the x axis), to impose the symmetry condition. Zero gradient for the heat flux is imposed at the last streamline (everything coming in comes out from the upper side).

Diffusive mass fluxes: - "`none`" or "`computed`". The computation happens only across streamlines, in the y direction. No streamwise diffusion is computed. This is fine if the concentration gradients are mainly transverse. At the lower streamline (0^{th} streamline) diffusive mass fluxes are set to zero (symmetry condition). At the upper streamline, zero gradient is imposed for the mass fluxes.

5.3 Baseline solution file

The basic format for the baseline solution file is as follows.

```
x   y   T   rho   u   v   X1   X2   X3   ...   XN
```

where:

- `x, y` - position of the streamline points [m]
- `T` - temperature [K];
- `rho` - mixture density [kg/m3]
- `u, v` - velocity components along `x` and `y` axis [m/s];
- `Xi` - species concentrations, in the order required by `Mutation++`. Can be mass fractions or mole fractions, according to the input file.

[7] Total enthalpy $H = h + u^2/2$.

[8] Recall that `LARSEN` and `MultiLARSEN` are all an approximation. Taking the enthalpy variation from outside makes the method closer to the baseline computation.

Some notes

The velocity and density fields are taken as they are and not recomputed along the streamline. The temperature and chemical species are taken as initial condition at the beginning of the streamline.

If the total enthalpy diffusion is flagged as `external`, the temperature and mixture composition along the baseline streamline are used to compute the enthalpy variation from one streamline point to the next one.

The **spacing** among consecutive streamline points is not really critical, since `MultiLARSEN` automatically sub-refines the step. Of course, you need to provide enough points for the physics to be reasonably followed.

The syntax is pretty similar to the `LARSEN` syntax, so you can refer to section 4.3.1 for an example.

5.4 Output

```
Loading Input file 'input_multilarsen.in'

Input file was read:
    Problem type: multilarsen
    Mixture: air11
    State model: ChemNonEq1T
    Thermodynamic database: RRHO
                                                                                  __
                                                           __  ___       __ __   /_/
                                                          /  |/  /__ __ / // /_ __   ____
                                                         / /|_/ // // // // __// /  /___/
                                        _______ ___     /_/  /_/ \_,_//_/ \__//_/
      /\                                \      \  /                  __   ___   ___  _________  __
     /  \         ____________           \      \/O_______   ____   / /  / _ | / _ \/ __/ __/ |/ /
    /    \       /            \____      (      O,'/ / /  \ /___/  / /__/ __ |/ , _/\ \/ _//    /
    V\  /V      /  ..........      \     (                o)      /____/_/ |_/_/|_/___/___/_/|_/
      \ \     __\   ..       ....   \    (       /wwwwwwwww'
       ) \   /  \)  ..  /\___   ..   \__/ .     (   ------------------------------------------->
       (  \ / .... ..  (     )  ..........      (  ---------------------------------------->
        \  v .  _ ....  )    (   ..........     ( ------------------------------------->
         ). .  / \_____/      \   ..  _____      \MMMMMMMM\   Multi - LAgrangian Reactor
         \ .  /                \_____/     \______________/        for StrEams in Nonequilibrium
          \__/


MULTILARSEN settings from input file:
    Geometry:                    axisymmetric
    External flowfield filename: streamlines_1000km/str_
    Number of streamlines:       55
    Species input type:          mole_fraction
    Total enthalpy diffusion:    computed
    Mass fluxes:                 computed

Reading streamlines: 0.. 1.. 2.. 3.. 4.. 5.. 6.. [.. cropped ..] 52.. 53.. 54.. Done.
Processing streamlines: 0.. 1.. 2.. 3.. 4.. 5.. 6.. [.. cropped ..] 52.. 53.. 54.. Done.
Exporting baseline solution - 33912 cells... Done.
Output settings:
    Species output type: number_density
    Solution is printed each 1 steps.
```

```
======>  STARTING THE INTEGRATION  <======


 MULTILARSEN - Step 1 of 314
==========  ctrl_rkck54 - Controlled Stepper  =======
Exporting solution - 33912 cells... Done.

 MULTILARSEN - Step 2 of 314
==========  ctrl_rkck54 - Controlled Stepper  =======
Exporting solution - 33912 cells... Done.

 MULTILARSEN - Step 3 of 314
==========  ctrl_rkck54 - Controlled Stepper  =======

.. and so on.
```

The solution is exported as a VTK file named `MultiLARSEN_sol.vtp`, you can open it with `ParaView`. This file is written when the solver reaches the location of the streamline x stations. If you brutally stop the simulation with `CTRL + C` you'll end up with the solution up to the point that you reached [9].

In the exported files, you can find the chemical species concentrations among the fields. The species names are preceeded by a prefix "`Y_`" if they represent **mass fractions**, "`X_`" for **mole fractions** and "`n_`" for **number densities** (particles per cubic meter).

Also the baseline solution is exported, in a file named `BaselineSol.vtp`. Finally, when the computation is over, another file is written storing both the baseline solution and the recomputed one, called `AllSol.vtp`.

[9]Unless you were so unlucky to stop the simulation while the software was writing the file.. In this case the file will be corrupted.

6 FAQ - Frequently Asked Questions

List of questions:

Q: Should I trust LARSEN/MultiLARSEN results?

A: `LARSEN` and `MultiLARSEN` assume that the velocity from the baseline solution is good enough and do not recompute it. This is an approximation: creation of more dissociated / ionized species requires more energy from the system, the enthalpy will thus be different and the velocity *should* change as well. However, if the change is not too big, the velocity will be reasonably close to the previous one and the result can thus be trusted. Yet, `(Multi)LARSEN` is useful because it gives the concentration of many more chemicals, but keeping down the computational times.

You can do two tests:

1. You should run `(Multi)LARSEN` with the same settings as the baseline solution - i.e. the same thermochemical model. You should be able to retrieve the exact solution (or a similar one at least).
2. Then, you can run `(Multi)LARSEN` with the new mixture and see how well the already existing species match with the baseline solution.
   Example: say you start from a neutrals-only baseline solution and go to a neutrals plus ions solution: if the ions are present in a small quantity, the recomputed neutrals will match quite well the baseline solution. This implies that the energy balance didn't really change a lot, and the solution can thus be trusted, since using the old velocity field is fine.

Q: LARSEN gets stuck at a certain point.

A: If the streamline reaches the surface (stagnation point), when the velocity reaches zero `LARSEN` gets crazy. You can safely stop the simulation with `CTRL+C` when this happens: all the previous points were already outputted, and for sure you're not going to have any other point after the surface.

You can try to refine the baseline solution points near the stagnation point to allow `LARSEN` to get closer and closer to the surface.

Q: LARSEN explodes near the stagnation point.

A: See Q: LARSEN gets stuck at a certain point.

Q: LARSEN / MultiLARSEN - the temperature reaches 0 K or goes negative.

A: Remember: `LARSEN / MultiLARSEN` are approximated solvers. They take the velocity field as given and recompute the species and temperature(s). Adding new chemical species (chemical refinement) with `(Multi)LARSEN` implies changing a bit the energy balance: if you are adding new minor species, you can add as many as you wish and this will not be a problem at all. If you are adding new *very* relevant species, this may change the energy balance too much, and the hypothesis of "keep the velocity as it was before" may not hold any more.

Temperature going to 0 K often happens if you take a baseline simulation with heat fluxes and then you simulate an adiabatic particle with `(Multi)LARSEN`. Usually, taking the heat flux as `external` fixes the problem.

For `MultiLARSEN`, this may also be due to diffusion: see Q: MultiLARSEN gets crazy with diffusion enabled

So, in general: when the new thermo-chemical model is *much too far* from the original one this might happen. Of course it's hard to say when it's the case. One thing you can do is trying to make the integration step smaller, in the code.

Q: MultiLARSEN gets crazy with diffusion enabled.

A: This may happen if you specified a very different concentration of species among the streamlines, changing in a non-smooth manner. An instability may develop in this region and grow up.

Make the species concentration change in a smoother way, and/or try to change the distance among streamlines in the region.

You can try to change the integrator type in the code (`Solver.cpp` file). This was happening to me with the `rosenbrock4` stiff solver. Runge-Kutta Cash-Karp 5-4 was working fine for me. See the `boost odeint` documentation for a list of solvers.

Q: Where is the LARSEN / MultiLARSEN recomputed velocity / density?

A: `(Multi)LARSEN` does not recompute the velocity and density fields but takes them from the baseline solution. See section 4.

Q: LARSEN does not reproduce correctly a shock.

A: If the a baseline streamline crosses a shock and LARSEN is not able to correctly reproduce this, make sure that the baseline solution is fine enough. If the shockwave is too roughly computed in the baseline solution (if you totally miss the shock peak because you are using a too-much coarse mesh) then LARSEN cannot do miracles. Remember the saying: *Garbage In Garbage Out*.

Q: I have a number of streamlines: can LARSEN automatically process them all?

A: No, automatical processing of many streamlines is not currently supported by `LARSEN`. However, all you have to do is making a `bash` script calling `LARSEN` once for each streamline you have to process. Take inspiration by the bash scripts in the `verification` folder.

Can I use MultiLARSEN instead?
Note that `MultiLARSEN` inherently works with multiple streamlines, but it's based on the assumption that the streamlines are somehow aligned to the x axis (and never get perpendicular to it). If your streamlines have a lot of geometrical variation and you don't care about inter-streamline diffusion, call separate instances of `LARSEN` instead.

Also, you may want to consider that the output of `MultiLARSEN` is a `vtp` file, while `LARSEN` directly outputs the result on the screen.

7 Structure of the code

TODO.

8 Adding a new solver

This section quickly goes through the steps you need to introduce a new solver. We will call it simply “`NewSolver`”. **This section is not complete and probably not even accurate** - take this as a hint. Start by reading section 7 to understand the structure of the code.

Step 1 - The src folder

Create a directory `newsolver` in `brODErs++/src/`, by copying one of the existing solvers directories. You will probably need the following files:

“`SetupNewSolver.cpp`” and `.h` - builds the solver by creating pointers to the needed classes.

“`DataNewSolver.cpp`” and `.h` - class storing the data. This also does some parsing of the input file looking for flags (Example: Total enthalpy duffusion: external / adiabatic).

“`NewSolver.cpp`” and `.h` - class defining the functions doing the actual numerical computations. The most important function here is `computedxdt(...)`, that is called at each integration step. Copy its basic syntax from an existing solver.

Step 2 - CMakeLists.txt files

For your source to be compiled by CMake, you need to modify the `CMakeLists.txt` files. First of all, modify the file `[...]/brODErs++/CMakeLists.txt`, adding the line:

```
include_directories(src/newsolver)
```

Then modify the file `[...]/brODErs++/src/CMakeLists.txt`, adding the line:

```
add_subdirectory(newsolver)
```

Finally, you need a `CMakeLists.txt` file in your new solver directory. Copy it from another solver directory and modify accordingly. It will be something like:

```
cmake_minimum_required(VERSION 2.6)

# Source code here
add_sources(brODErs++
    DataNewSolver.cpp
    NewSolver.cpp
    SetupNewSolver.cpp
)

# Headers here
install(FILES DataNewSolver.h DESTINATION bin/include)
install(FILES NewSolver.h DESTINATION bin/include)
install(FILES SetupNewSolver.h DESTINATION bin/include)
```

Step 3 - Modify the SetupProblem.cpp and .h

You need to modify the files `src/general/SetupProblem.h` and `.cpp`, that create the "problem" (aka the solver) that you choose via input file. This file creates the appropriate solver according to the input file.

Step 4 - Modify the Solver.cpp and .h

You need now to modify the `Solver.cpp` and probably the `Solver.h` files, in the `src/general` directory. [This step will be probably modified in future versions: the solver will become a base class and every solver will implement their own solver derived class].

Steps 5, 6, ..., N - 1

There are probably many more modifications not reported here. Take this section as a rough hint.

Step N - The most important one

Try to ensure that your solver is usable and understandable by other people: comment it, avoid unnecessarily complicated programming style[10] and try to follow the KISS[11] principle. Finally, make some torough testing and **update the documentation!**

[10]But try to program better than I do (Stefano) :-)

[11]Keep it simple stupid.

References

[1] A. Munafò, *Multi-Scale Models and Computational Methods for Aerothermodynamics*, PhD thesis, Ecole Centrale Paris, von Karman Institute for Fluid Dynamics, 2014.

[2] M. Panesi, *Physical Models for Nonequilibrium Plasma Flow Simulations at High-Speed Re-Entry Conditions*, PhD thesis, Universitá degli Studi di Pisa, von Karman Institute for Fluid Dynamics, 2009.

[3] S. Boccelli, *Development of a Lagrangian Solver for Thermochemical Nonequilibrium Flows*, MSc Thesis, Politecnico di Milano, 2016.

[4] A. Klomfass and S. Müller, *Calculation of a Stagnation Streamline Quantities in Hypersonic Blunt Body Flows*, Shock Waves, Vol. 7, pp. 12-23, 1997.

[5] S. Boccelli, *Simulations of electron concentration in the wake of meteors and application to radio observations*, VKI Research Master project report, von Karman Institute for Fluid Dynamics, 2017.