NASA/TM–2020–220566

LAURA Users Manual: 5.6

Kyle B. Thompson, Brian R. Hollis, Christopher O. Johnston, Bil Kleb, Victor R. Lessard, and Alireza Mazaheri
Langley Research Center, Hampton, Virginia

February 2020

NASA/TM–2020–220566

LAURA Users Manual: 5.6

Kyle B. Thompson, Brian R. Hollis, Christopher O. Johnston, Bil Kleb, Victor R. Lessard, and Alireza Mazaheri
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

February 2020

Available from:

NASA STI Program / Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199
Fax: 757-864-6500

Abstract

This users manual provides in-depth information concerning installation and execution of Laura, version 5. Laura is a structured, multiblock, computational aerothermodynamic simulation code. Version 5 represents a major refactoring of the original Fortran 77 Laura code toward a modular structure afforded by Fortran 95. The refactoring improved usability and maintainability by eliminating the requirement for problem-dependent recompilations, providing more intuitive distribution of functionality, and simplifying interfaces required for multi-physics coupling. As a result, Laura now shares gas-physics modules, MPI modules, and other low-level modules with the Fun3D unstructured-grid code. In addition to internal refactoring, several new features and capabilities have been added, e.g., a GNU-standard installation process, parallel load balancing, automatic trajectory point sequencing, free-energy minimization, and coupled ablation and flowfield radiation.

Contents

1 Introduction

The users manual consists of seven sections. Section 2 gives an overview of new features, capabilities, and bug fixes. System requirements and installation are covered in Section 3, followed by code execution instructions in Section 4. Section 5 presents input files, their formats, and detailed information on their contents while Section 6 covers output files. Ancillary utilities are explained in Section 7, and the last section, Section 8, presents illustrative example cases.

2 New in This Version

LAURA v5.5 offers several new enhancements and bug fixes since the previous released version, v5.4 [1]:

- Major Enhancements
  - Addition of `'adiabatic catalytic'` surface temperature BC—see Section 5.4.12 on page 35 for more info.
  - Additional Constant Catalysis
  - Roughness height measurement
- Minor Enhancements
  - y^+ output in the `laura_surface.q` file for 1- and 2-eqn turbulent models.
  - Option to change the maximum temperature limit.
  - Option to change the maximum percentage change in the solution updates.
  - Separating diffusion component of heat flux from the total surface heat flux for multispecies gas.
  - Option to make the velocity boundary condition second order regardless of the `nordbc` value.
  - Option to activate the temperature contribution to the Jacobian.
  - Option to limit species vibrational-electronic heat capacity for more stability in some two-temperature cases.
  - Option to write the adapted grid with negative cell volume.
- Bug Fixes
  - Repositioning initialization of variable `'rough_edge'`
  - Reformulate the computation of variable `'re_kk'`
  - Correcting the aerodynamic forces and moments calculation
  - Correcting the B.L. edge detection routine for cases with negative total enthalpy
  - Fixing NaN occurring problem caused by undefined molecular weight in the viscous routine for the single-species turbulent cases

3 Installation

Laura requires a Fortran 95 compiler, and if parallel processing is desired, a Message Passing Interface (MPI) implementation.[1] Some optional utilities require Ruby.[2] The installation and subsequent execution of Laura assumes a Unix-like operating system or compatibility layer.[3] After the code is unpacked from the Laura release tarball,

```
% tar zxf laura-5.4-Z.tar.gz  (unpack gzipped tarball)
% cd laura-5.4-Z
```

where `Z` is a revision track number. Laura is installed via GNU build system,[4] which entails executing a sequence of four commands: `configure`, `make`, `make check`,[5] and `make install`.[6]

3.1 Sequential installation

To configure, compile, test, and install a sequential version of Laura for use with a single processor, first make a subdirectory of `laura-5.4-Z` to store the configuration. For example,

```
% mkdir g95-seq
% cd g95-seq
```

if using the g95 Fortran compiler;[7] and then proceed with the typical GNU build sequence,

```
% ../configure FC=g95 --prefix=$PWD
% make
% make check
% make install
```

Note that `configure`'s `--prefix` option specifies the root directory for installing build artifacts—the default is `/usr/local`. In this example, it is set to the current working directory, `g95-seq` so executables will be installed in `g95-seq/bin` and data files will be copied to `g95-seq/share/laura` and `g95-seq/share/physics_modules` directories.

To use Laura and associated utilities, set your search path to include `$PWD/bin`, e.g.,

```
setenv PATH ${PWD}/bin:$PATH  (for csh)
export PATH=${PWD}/bin:$PATH  (for sh)
```

[1]For example, OpenMPI or MPICH.

[2]See ruby-lang.org.

[3]For non-Unix-like systems, compatibility layers are available from mingw.org (minimal) and cygwin.com (maximal).

[4]See gnu.org/software/autoconf/.

[5]The `make check` command is optional. It will attempt to run small test cases.

[6]The `make install` command may require administrator privileges depending on your installation location.

[7]g95.org

3.2 MPI Installation

An MPI-enabled installation (to allow multiple processors) is similar to the sequential installation except the configuration command has the `--with-mpi` option instead of the `FC` Fortran compiler variable, e.g.,

```
% ../configure --prefix=$PWD \
               --with-mpi=/usr/local/pkgs/ompi_1.2.8-intel_11.0-028
```

Another difference is that an MPI-enabled configuration will produce an executable named `laura_mpi` instead of `laura`.

In either case, `config.log` contains a record of the `configuration` command used, and `configure`'s `--help` option details all available configuration options.

4 Execution

The following steps outline a typical simulation cycle.[8]

Step 1. To start LAURA, a PLOT3D structured grid file is needed — see Section 5.2 on page 11 for more info. You may externally generate a grid using grid generation packages, such as Gridgen™, Grid*Pro*, and so forth, or use LAURA's interactive `self_start` utility to generate a single-block structured grid for simple families of 2D, axisymmetric and 3D blunt bodies—see Section 7.19 on page 75.

Step 1a. Using `self_start`. To use `self_start` to generate a single-block grid, simply execute this interactive utility, e.g.,

```
% self_start
```

and answer all the questions. After a successful execution, this utility will have generated the following files:

```
               laura_bound_data
laura.g        laura_namelist_data  self_start.log
```

Examine the grid, `laura.g`, and proceed to Step 2.

Step 1b. External Grid Generation. Generate a single- or multi-block structured grid with the following rules:

i. Right-handed grid coordinates
ii. Longitudinal axis of the body aligned with the x-axis, oriented nose-to-tail

and write the grid coordinates into a PLOT3D file, `laura.g`. Run the interactive `bounds` utility (see Section 7.4 on page 71) and answer all the questions regarding the grid block topology:

```
% bounds
```

This utility will automatically generate `laura_bound_data`, the connectivity file.

Step 2. If you did not use `self_start`, create a `laura_namelist_data` file or copy the sample file from the `[install_prefix]/share/laura` directory, where `[install_prefix]` is the installation prefix specified when LAURA was installed. Edit this file for your case—see Section 5.4 on page 14 for more detail.

[8]See Section 8 on page 77 for complete worked examples and Appendix A on page 88 for how to restart cases run with versions of LAURA prior to version 5.

Step 3. Create a `tdata` file (see Section 5.5 on page 43) to define the gas model condition for your specific simulation.[9]

Step 4. Run Laura,

```
% laura
```

or

```
% mpirun -np [#] laura_mpi
```

where `#` is the number of available processors. By the end of this step, the following files will have been generated:

```
laura_conv.out      laura.g.fvbnd
laura_new.g         laura_new.rst
laura_surface.q     laura.q
laura_surface.nam   laura.nam
```

Examine these files before proceeding to the next step.[10]

Step 5. Change `irest` flag in the `laura_namelist_data` (see Section 5.4 on page 14) from `0` to `1`, and copy the new generated grid and solution files to `laura.g` and `laura.rst` files; i.e.,

```
% cp laura_new.g laura.g
% cp laura_new.rst laura.rst
```

Step 6. Repeat the previous two steps until iterative convergence.

[9] A sample `tdata` is available in the `[install_prefix]/share/physics_modules` installation directory. The other datafiles that reside in this directory, e.g., `kinetics_data`, `species_thermo_data`, `species_transp_data`, and `species_transp_data_0`, may also be copied and tailored to suit a different thermodynamic model, curve-fit data, or thermochemical reactions are needed. See Section 5.5 on page 43 for more detail.

[10] See Section 6 on page 64 for complete description of laura output files.

5 Input Files

Nominally, LAURA requires five input files as shown in the upper section of Table 1. Depending on the simulation requirements, however, other files may also be necessary and are shown in the second section of Table 1. All files are plain ASCII text unless otherwise noted.

Table 1: LAURA input files.

Filename	**Content**
REQUIRED FILES:	
`laura.g`[*]	PLOT3D grid
`laura_bound_data`	Grid block face boundary conditions
`laura_namelist_data`	Simulation configuration
`tdata`	Gas model
SIMULATION DEPENDENT FILES:	
`assign_tasks`	Sweep and relaxation directions
`hara_namelist_data`	Radiation mechanisms
`jdata`	Jet chamber conditions
`kinetic_data`	Specie reactants and products
`laura.rst`[†]	Flowfield solution for restart
`laura.trn`	Transition location and length
`laura_trajectory_data`	Trajectory points
`laura_vis_data`	Viscous term treatment
`species_thermo_data`	Specie thermodynamics
`species_transp_data`	Collision cross-sections
`species_transp_data0`	High-order collision cross-sections
`surface_property_data`	Thermochemical surface properties

[*] Fortran unformatted binary, 3-D whole, multiblock PLOT3D.
[†] Fortran unformatted binary.

The following subsections describe all input files in detail, beginning with the nominally required files and then proceeding alphabetically as shown in the table.

5.1 REQUIRED FILES

5.2 `laura.g`

This file is a multiblock, 3D-whole PLOT3D file in Fortran unformatted binary format with double-precision reals. For convenience, here is a sample of the

Fortran 95 code that LAURA uses to read this file:

```
open ( 25, file='laura.g', form='unformatted' )
read(25) nblocks
...allocate i,j,kblk(nb) and grid memory...
read(25) (iblk(nb),jblk(nb),kblk(nb),nb=1,nblocks)
do nb = 1, nblocks
  ix1 = iblk(nb) ; jx1 = jblk(nb) ; kx1 = kblk(nb)
  ...allocate grid(nb)%x,y,z memory...
  read(25) (((grid(nb)%x(i,j,k),i=1,ix1),j=1,jx1),k=1,kx1), &
           (((grid(nb)%y(i,j,k),i=1,ix1),j=1,jx1),k=1,kx1), &
           (((grid(nb)%z(i,j,k),i=1,ix1),j=1,jx1),k=1,kx1)
end do
```

A file of this format, but named `laura_new.g`,[11] is generated by `Laura` at the end of a successful run. This file is required and must have a right-handed coordinate system. Figure 1 shows the default `laura` coordinate orientation. `Laura` does not require a specific coordinate or grid orientation but the angle-of-attack definition is predefined—see Section 5.4.3 on page 20 for more info.

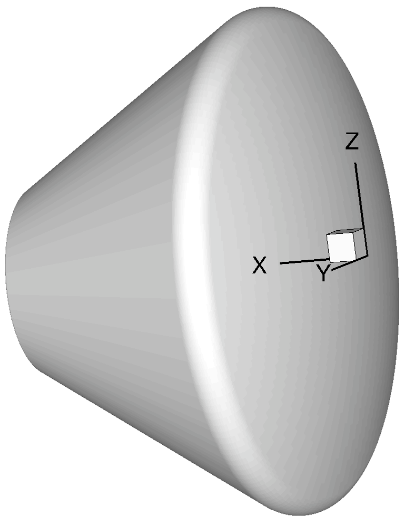


Figure 1: Default LAURA coordinate system orientation.

5.3 `laura_bound_data`

Grid block face boundary types are defined in `laura_bound_data` where each line corresponds to a grid block and contains six integers, one for each of the

[11]When running a trajectory sequence, the file will be named `laura_####.g` where `####` is the trajectory point index.

six faces: i_{min}, i_{max}, j_{min}, j_{max}, k_{min}, and k_{max}. Each integer specifies either a physical boundary condition or block-to-block interfaces. An illustrative example is analyzed toward the end of this section.

This file is required and is generated automatically for grids created by LAURA's `self_start` utility—see Section 7.19 on page 75. This file can also be created by using LAURA's interactive utility, `bounds`, by answering questions for each block.

Valid face types are as follows:

`-9,...,0`: Solid surface boundary. Up to ten different solid surface boundaries may be specified. Thermochemical properties of solid surfaces that are different than type `0`, which are specified in `laura_namelist_data`, are defined in `surface_property_data` file—see Section 5.18 on page 61.

`1`: Outflow boundary (extrapolation).

`2`: Symmetry boundary across y = constant.

`3`: Farfield/Freestream boundary.

`4`: Symmetry boundary across x = constant or z = constant.

`5`: Reflection boundary across $j = 1$ symmetry (valid for axisymmetric and/or 2D grids).

`6`: Venting boundary. (See Section 5.4.18 on page 43 for more details.)

`7`: Reflection boundary across i face singularity with periodic j boundary.

`8`: Characteristic (also known as subsonic) boundary condition.

`>1000000`: This seven digit boundary number defines block-to-block face connectivity. The first digit is always `1`. The next three digits identify the block number that is shared with the current block. The 5th digit defines which i, j, or k face of the neighboring block is shared where `1` corresponds to i_{min}, `2` corresponds to i_{max}, and so forth. The last two digits identify the relation of the remaining two indices: The 6th digit can be either `1`, `2`, `3`, or `4` where values of `1` or `3` mean the first index of the host face is in the same direction as the first or second index of the neighboring face, respectively, and values of `2` or `4` mean the adjoining indices are in the opposite direction. The last digit can be either a `1` or `2` and indicates whether the second indices of the host and neighboring faces are in the same direction or they are in the opposite direction, respectively.

For example, consider a block with the following `laura_bound_data` boundary condition numbers:

```
  1006421   1002111      2    1005121      0        3
```

The first integer, `1006421`, shows that i_{min} of this block is shared with j_{max} (fifth digit) of block 6 (the first three digits after `1`). The first and second indices of the host face are j and k, respectively. Because the j index is connected to i, the first and second indices of the neighboring face are i and k, respectively. The `2` in the 6th digit shows that the j index of the host face is in opposite direction of the i index of the neighboring face. The last digit, `1`, indicates that k indices of the host and and neighboring faces are along the same direction. This configuration is illustrated in Figure 2.

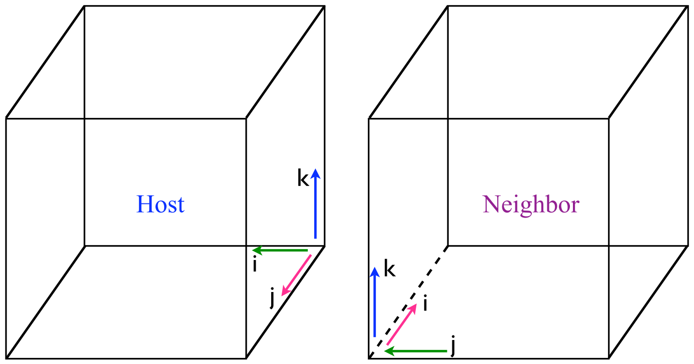


Figure 2: Illustration used for block connectivity example.

The second and fourth boundary-type integers can be explained similarly. The third boundary-type integer (corresponding to the j_{min} face) specifies a y-constant symmetry plane; the fifth boundary-type integer (corresponding to the k_{min} face) specifies is a solid-wall boundary condition; and the last digit (corresponding to the k_{max} face) specifies a freestream boundary.

5.4 laura_namelist_data

Simulation configuration is specified through `laura_namelist_data` and is required. This file is read as a Fortran 95 namelist and has the form,

```
  &laura_namelist
   velocity_ref = 5000.0 ! Free stream velocity, m/s
   density_ref  = 0.023  ! reference density, kg/s
   tref         = 250.0  ! Free stream temperature, K
   alpha        = 25.0   ! Angle-of-attack in xz plane, degrees
 [ variable     = value  ! Optional comment ]
  /
```

where `variable` and its possible `values` are described in the following sections, which are grouped according to farfield/freestream and aerodynamic coefficient reference quantities; thermochemical nonequilibrium flags; molecular transport flags; turbulent transport models flags; numerical parameters; grid adaptation, alignment, and doubling parameters; radiation and ablation flags; grid-file description; venting boundary condition flags; and solid surface boundary condition flags. Note that for all but the parameters shown in the above example, reasonable defaults have been chosen and only those parameters that differ from the defaults need to be specified.

Detailed description of parameters and/or flags with their units and default values is presented under each of the above categories. The order of these parameters is arbitrary but is given here alphabetically for better readability.

5.4.1 Ablation Flags

See Section 8.3 for a coupled ablation example using the options described in this section.

`treat_ablation`

Logical flag indicating that the injection of char and pyrolysis ablation products are modeled, and that the appropriate surface energy balance is implemented to determine the surface temperature. Setting this option to `.true.` engages the options described in this subsection, which override the typical nonablating surface options `catalysis_model_0` and `surface_temperature_type_0`.

`mdot_c_approach`

Character indicator for specifying the approach used for modeling the char rate. Default: `'equilibrium_char'`

Options are:

`'equilibrium_char'`

This option specifies that the char rate is computed assuming chemical equilibrium between a carbon char and the gas at the wall. Details of this assumption and its implementation are discussed by Johnston et al. [2,3].

`'finite_rate'`

This option specifies that the char rate is computed using finite-rate surface reactions, such as oxidation, nitridation, and sublimation. Details of the rate model applied are presented by Johnston et al. [4].

`'curve_fits'`

This option specifies a quasisteady ablation rate as a function of local pressure, heating rate, and temperature. The sublimation temperature and heat of ablation are specified by the user for a given ablator as a function of pressure. (See `t_sublimation_0` and `h_ablation_0`.) If the surface temperature is less than the sublimation temperature, a zero blowing rate is defined. Otherwise, the blowing rate is given by $\dot{m} = q/\Delta H_{abl}$ with appropriate nondimensionalization employed before use. The user must specify the elemental mass fractions for pyrolysis gas or default of 100% carbon will be employed. Any specification for elemental mass fraction of char is ignored in this option. This option automatically sets `mdot_g_approach` to `'quasi_steady'` and `temp_approach` to `'energy_equation'`.

`'pressure_function'`

This option specifies the char rate as a function of pressure (see `mdot_pressure_0`). Equilibrium chemistry at the surface is assumed when implementing this char rate.

`'specified'`

This option specifies that the char rate is defined according to the option chosen for `specified_source`.

<u>`mdot_g_approach`</u>

Character indicator for specifying the approach used for modeling the pyrolysis rate. Default: `'quasi_steady'`.

Options are:

`'quasi_steady'`

This option applies the quasisteady ablation assumption that the pyrolysis rate ($\dot{m}_g$) is proportional to the char rate ($\dot{m}_c$) through the following relationship:

$$\dot{m}_g = (\frac{\rho_v}{\rho_c} - 1)\dot{m}_c \tag{1}$$

where ρ_v and ρ_c are the density of the virgin ablator and char, respectively, which are defined by the name list options `virgin_density_0` and `char_density_0`.

`'specified'`

This option specifies that the pyrolysis rate is defined according to the option chosen for `specified_source`.

<u>`temp_approach`</u>

Character indicator for specifying the approach used for modeling the surface temperature. Default: `'energy_equation'`

Options are:

- `'energy_equation'`

  This option solves the surface energy balance for surface temperature.

- `'specified'`

  This option specifies that the surface temperature is defined according to the option chosen for `specified_source`.

<u>`specified_source`</u>

Character indicator for specifying approach used to defined the char rate, pyrolysis rate, or wall temperature when `mdot_c_approach`, `mdot_g_approach`, or `temp_approach` are set to `'specified'`.

Default: `'ablation_from_laura'`

Options are:

- `'ablation_from_laura'`

  This option specifies that values are read from an ascii file named `ablation_from_lauraXXXX.m`, where XXXX is the block number. The is file is created for `mdot_c_approach` = `'equilibrium_char'`.

- `'ablation_from_fiat'`

  This option specifies that values are read from an ascii file named `ablation_from_fiatXXXX.m`, where XXXX is the block number. The is file is applied for coupling with a material response code.

- `'restart_file'`

  Variables stored in the `laura.rst` file, which were defined from a previous run, are applied. This option should not be applied for a case freshly started from a nonablating solution.

<u>`ablator_material`</u>

Character indicator for specifying the ablation material. This option provides a shortcut for specifying `CHONSi_frac_pyrolysis_0`, `CHONSi_frac_char_0`, `virgin_density_0`, and `char_density_0`.

Default: `'carbon_phenolic'`

Options are:

- `'avcoat'`

  Assumes an Avcoat ablator defined by Bartlett et al. [5]. Requires that carbon, hydrogen, oxygen, nitrogen, and silicon species are treated in the flow field.

‘`pica`’

Assumes a PICA ablator as defined by Park [6]. Requires that carbon, hydrogen, oxygen, and nitrogen species are treated in the flow field.

‘`carbon_phenolic`’

Assumes heritage Galileo-era carbon phenolic [7]. Requires that carbon, hydrogen, and oxygen species are treated in the flow field.

‘`graphite`’

Assumes 100% carbon ablator with no pyrolysis gas. Requires that carbon species are treated in the flow field.

`ablation_verbose`

A logical flag to print out developer focused info on convergence of ablation. Default: `.true`.

`char_density_0`

Density of the char, kg/m^3. Default: 256.29536, which is for the heritage AVCOAT.

`virgin_density_0`

Density of virgin material, kg/m^3. Default: 544.627742, which is for the heritage AVCOAT.

`CHONSi_frac_char_0`

This rank 1 vector of extent 5 sets elemental mass fraction of C, H, O, N, and Si species from char. Elemental mass fractions must be in this order and the sum of elemental mass fractions must equal `1.0` . Default:
`CHONSi_frac_char_0 = 1.0, 0.0, 0.0, 0.0, 0.0`

`CHONSi_frac_pyrolysis_0`

This rank 1 vector of extent 5 sets the elemental mass fractions of pyrolysis gas species, which are C, H, O, N, and Si. Elemental mass fractions must be in this order and the sum of elemental mass fractions must be 1. Default is Graphite:
`CHONSi_frac_pyrolysis_0 = 1.0, 0.0, 0.0, 0.0, 0.0`

`compute_mdot_initial`

An integer defining if the ablation rates are computed before the first flowfield iteration.

Options are:

0: Applies the ablation rates and wall temperatures already present in the restart file until `freq_mdot` is reached.

1: Computes the ablation rates and wall temperatures before the first flowfield iteration. This is required for cases that were freshly shuffled from a nonablating case.
Default = `1`

<u>`freq_mdot`</u>

An integer defining frequency for recomputing char rates or wall temperature, depending on `mdot_c_approach` and `temp_approach`. Char rates are recomputed at this frequency for `mdot_c_approach` = `'equilibrium_char'` or `'curve_fits'` (for `mdot_g_approach` = `'quas_steady'`, pyrolysis rates are updated accordingly). Wall temperatures are recomputed at this frequency for `temp_approach` = `'energy_equation'`. Note that `freq_mdot` should be large enough to allow the convective heating to reach realistic values.

Default = `5000`

<u>`freq_wall`</u>

An integer defining frequency of update to pseudocell flowfield quantities. This parameter defines how quickly the char rate, pyrolysis rate, and surface temperature, which are recomputed at `freq_mdot`, are introduced to the flowfield. At each instance of `freq_wall`, the char rate, pyrolysis rate, and surface temperature are incremented by `ept` and the pseudocell values for pressure, temperature, and species composition are updated.

Default: `50`

<u>`h_ablation_0`</u>

A rank 1 vector of extent 3 used to compute the heat of ablation in MJ/kg for `mdot_c_approach` = `'curve_fits'` as

$$\begin{aligned}\texttt{h\_ablation\_0(1)} &+ (\texttt{h\_ablation\_0(2)})\log_{10} p_w \\ &+ (\texttt{h\_ablation\_0(3)})(\log_{10} p_w)^2\end{aligned} \qquad (2)$$

where p_w is the local pressure, in atmospheres. For example, for Galileo heritage carbon-phenolic: $\Delta H_{abl} = 29.149 - 1.174\log_{10} p_w + 0.0577(\log_{10} p_w)^2$.
Default: `0.0`

<u>`t_sublimation_0`</u>

A rank 1 vector of extent 3 used to compute the sublimation temperature in degrees Kelvin for `mdot_c_approach` = `'curve_fits'` as

$$\begin{aligned}\texttt{t\_sublimation\_0(1)} &+ (\texttt{t\_sublimation\_0(2)})\log_{10} p_w \\ &+ (\texttt{t\_sublimation\_0(3)})(\log_{10} p_w)^2\end{aligned} \qquad (3)$$

where p_w is the local pressure,in atmospheres. For example, for Galileo heritage carbon-phenolic: $T_{sub} = 3867.9 + 351.0\log_{10} p_w + 29.0(\log_{10} p_w)^2$.
Default: `0.0`

mdot_pressure_0

A rank 1 vector of extent 2 is used to set the char rate distribution for mdot_c_approach = 'pressure_function' defined as

$$\texttt{mdot\_pressure\_0(1)} + (\texttt{mdot\_pressure\_0(2)})\frac{p}{\rho_\infty V_\infty^2} \tag{4}$$

where p is the local pressure, ρ_∞ is the reference density, and V_∞ is the reference velocity. Positive value produces blowing distribution, while negative value produces suction distribution. Default: 0.0

bprime_corrected

Logical flag for specifying the approach for computing equilibrium ablation when mdot_c_approach = 'equilibrium_char'.

5.4.2 Aerodynamic Coefficient Reference Quantities

bref

Yaw moment coefficient reference length, grid-units. Default: 1.0

cref

Pitching moment coefficient reference length, grid-units. Default: 1.0

sref

Reference area for aerodynamic coefficients, grid-units. Default: 1.0

xmc

x-coordinate of moment center, grid-units. Default: 0.0

ymc

y-coordinate of moment center, grid-units. Default: 0.0

zmc

z-coordinate of moment center, grid-units. Default: 0.0

5.4.3 Farfield/Freestream Reference Quantities

alpha

Angle-of-attack in xz plane, degrees, such that

$$u = cos(\alpha)cos(yaw); v = -sin(yaw); w = sin(\alpha)cos(yaw) \tag{5}$$

where u, v, and w are velocities in x-, y-, and z-coordinate, respectively. Default: 0.0

buoyancy

Logical flag indicating if buoyancy terms are engaged. Default: .false.

`closed_chamber`

Logical flag indicating a closed test chamber is being simulated. In this case, there is no external velocity. Boundaries are all solid surfaces. Flow is driven by buoyancy associated with heating some portion of a surface. The reference velocity in this case is reset to the square root of the product of `grid_conversion_factor` with the magnitude of the gravitational vector. Default: `false`

`density_ref`

Free stream density, kg/m^3. Default: `0.001`

`gravity_x, gravity_y, gravity_z`

Components of gravity vector in mks units required if buoyancy is engaged. Default: `0.0, 0.0, -9.79908`

`rpm`

Spin rate, RPM. This is applicable only to axisymmetric cases.
Default: `0.0`

`tref`

Free stream temperature, K. Default: `200.0`

`velocity_ref`

Free stream velocity, m/s. Default: `5000.0`

`yaw`

Sideslip angle in xy plane, degrees. Default: `0.0`

5.4.4 Grid Adaptation, Alignment, and Doubling Parameters

`beta_grd`

This parameter controls grid points normal to the body (k-grid points) are controlled by the following grid stretching function:

$$k_i^* = 1 - \beta \frac{\frac{\beta-1}{\beta+1}^{\frac{k_{max}-k_i}{k_{max}}} - 1}{\frac{\beta-1}{\beta+1}^{\frac{k_{max}-k_i}{k_{max}}} + 1} \tag{6}$$

where k_i^* are new grid points. The `beta_grd` parameter defines the β coefficient in equation 6. No stretching will be performed if $\beta < 1$. Recommended value is 1.15, if used. Default: `0.0`

`ep0_grd`

Grid clustering around the shock is designed using the following function:

$$\overline{k^{**}}_i = \epsilon_0 \overline{k_i^*}^2 (1 - \overline{k_i^*})(\overline{k_i^*} + \texttt{fsh}) + \overline{k_i^*} \tag{7}$$

where $\overline{k_i}$ refers to normalized value and k_i^* is defined by equation 6 on the preceding page. `ep0_grd` assigns the ϵ_0 coefficient in equation 7 on the previous page. Maximum recommended value is 25/4, if used. Default: `0.0`

`kmax_final`

The target number of grid cells in the k direction (an integer). Triggered by the global L2 error norm set by `kmax_error`, cells along the k direction are increased by a factor of `kmax_factor` until reaching `kmax_final`. Any value less than the number of k grid cells in `laura.g` file will be ignored, i.e., no coarsening. This option requires all blocks to be active. Default: `0`

`kmax_error`

When the global L2 error norm reaches this value, the number of grid cells in the k direction increases by a factor of `kmax_factor` until the maximum allowable grid cells, `kmax_final`, is reached. This option requires all blocks to be active. Default: `0.01`

`fctrjmp`

This parameter is used to detect bow shocks. Bow shock is first detected when the sensing parameter, defined by `jumpflag`, exceeds by this value, while searching from inflow boundary. Default: `1.05`

`frac_line_implicit`

This positive parameter, which must be less than or equal to 1, sets a fraction of the line-implicit direction that is either assigned in the namelist (see Section 5.4.9 on page 26) and/or in the optional `assign_tasks` file (see Section 5.7 on page 46). This parameter, which supersedes the given assignments in `assign_tasks`, will be applied to all the active blocks. This parameter is recommended where there might be an instability issue such as across strong shocks. Default: `0.7`

`fsh`

Fraction of arc length distance between body and inflow boundary where bow shock is captured. Default: `0.8`

`fstr`

This parameter approximately defines the fraction of k-cells in the boundary layer region. Default: `0.75`

`jumpflag`

An integer flag to select the sensing parameter to be used in detecting the position of the captured shock for grid adaption. Default: `2`

Options are:

0: Redistributes grid points in k-direction for the target cell Reynolds number defined by `re_cell` parameter, without changing the domain boundary.

1: Selects pressure as the sensing parameter.

2: Selects density as the sensing parameter.

3: Selects temperature as the sensing parameter.

4: Scales the grid distance in the k-direction up by the value defined by `fctrjmp` parameter.

<u>`interp_grid`</u>

An integer flag to select the method for moving points along the k-index grid lines during the align-shock grid adaptation. Default: `0`

Options are:

0: Move points using a linear interpolation routine. Works best when k-index grid lines have low curvature. May create negative cell volumes with k-grid lines of high curvature.

1: Move points using a monotonic piecewise cubic spline routine. Works best for highly curved k-index grid lines with wiggles and bumps. Spline removes these wiggles and bumps.

2: Move points using a natural piecewise cubic spline routine. Works best for highly curved k-grid lines without wiggles and bumps.

<u>`maxmoves`</u>

An integer number to assign the maximum number of times that grid adaption is performed. The value of zero, however, can be used an unlimited number of times. Default: `0`

<u>`max_distance`</u>

This real number defines the maximum distance from the body surface that the grid outer boundary can be moved away by any one of the flags. This is often useful especially when adapting the shock into wake, where the adapting grid may become skewed due to the presence of sharp gradients. This value defines the maximum length of the wake. Default: `1.0E+6`

<u>`movegrd`</u>

An integer number for the number of cycles between each grid alignment. A zero value disables any grid alignment. Default: `0`

<u>`re_cell`</u>

This value defines the target cell Reynolds number at the wall after a grid movement. Note : If the grid is moving radically Default: `0.1` Note:

Use a higher value for `re_cell` for the first grid alignment, if the grid movement causes radical changes in the grid.

The cell Reynolds number is defined as

$$Re_{cell} = \rho c \Delta n / \mu \tag{8}$$

Here ρ is the flow density, c is the sound speed, Δn is the cell height, and μ is the flow viscosity.

`write_negvol`

A logical flag to allow the adapted grid with negative cell volume to be written out to the `laura_new.g`. Default: `.false`.

5.4.5 Grid File Description

`dimensionality`

A string flag to select the dimensions of the problem. Default: `'3D'`. Available options are:

`'axisymmetric'`

This option selects an axisymmetric flow solution. This requires a domain with single-cell width in the j-direction.

`'2D'`

This selects a two-dimensional problem, which requires a domain with single-cell width in the j-direction.

`'3D'`

This option solves three-dimensional problems.

`grid_conversion_factor`

This parameter scales the grid to meter units. One grid unit equals `grid_conversion_factor` meters. Default: `1`

`single_precision_grid`

Logical flag: `.true.` if grid points in the `laura.g` file are stored as single precision values, otherwise double precision is assumed. Default: `.false`.

5.4.6 Grid Limiter

`g_limiter`

A real number, normally $< 1 \times 10^{-3}$, that limits the cell size of grid cells off the wall for a better quality grid after adaptation. A negative or zero value turns the use of grid limiter off. Default: `0`.

5.4.7 Initialization

init_vel_fctr

A real number between `0` and `1` to reduce the initial velocities u, v, w within the domain to avoid creating a vacuum for wake flow problems, which may lead to an invalid solution. The initial near zero velocity has also shown to ease up the formation of the bow shock. Default: `0.01`

5.4.8 Molecular Transport Flags

ambipolar

A logical flag to engage ambipolar diffusion of ions. Default: `.true`.

ivisc

An integer flag to engage viscous terms (inviscid=0/viscous=2). Default: `2`

mass_driven_diff

A logical flag to engage binary diffusion driven by mass fraction gradient. Default: `.false`.

multi_component_diff

A logical flag to engage multicomponent diffusion by Stefan-Maxwell equation subiterations. Default: `.false`.

mole_driven_diff

A logical flag to engage binary diffusion driven by the mole fraction gradient. Default: `.true`.

navier_stokes

A logical flag to select the equation set for thin-layer Navier Stokes or Full Navier Stokes. The `navier_stokes = .false`. may be used to select Thin-Layer Navier Stokes. Default: `.false`.

pressure_diffusion

A logical flag to engage the pressure diffusion term on Stefan-Maxwell approximation. Default: `.false`.

schmidt_number

A constant Schmidt number may be specified to calculate diffusivities. If the value is negative, diffusivities are computed directly from collision cross sections. Default: `-1.0` `prandtl_number`

A constant Prandtl number may be specified to calculate conductivities. If the value is negative, conductivities are computed directly from collision cross sections. Default: `-1.0`

5.4.9 Numerical Parameters

`augment_shock_dissipation`

Augment dissipation across the shock to make cell Re number approach 2 where the pressure ratio is greater than 3. Default: `.true`.

`cfl1`

Initial value of CFL number. Default: `5.0`

`cfl2`

Final value of CFL number. Default: `5.0`

`density_floor`

Lower limit on species density. Default: `1.e-30`

`epsa`

Eigenvalue limiting factor. Default: `0.3`

`frac_update`

Maximum percentage change in solution update. A negative value engages an adaptive limiter, which is useful for improving convergence and stability early in the solution process. Default: `0.1` (10%)

`hrs`

The maximum total CPU time for simulation, hours. Default: `10`

`iramp`

Number of cycles to ramp from `cfl1` to `cfl2`. Default: `200`

`irest`

An integer flag to start the simulation either using freestream values (`irest = 0`), or using the existing solution from `laura.rst` file (`irest = 1`). Default `0`

`jupdate`

Number of cycles between Jacobian updates. Default: `10`

`max_temperature_floor`

Maximum allowable temperature in the flowfield. The gas temperature will be reset to this value if it goes beyond the floor limit. Default: `50000` ncyc

Number of global iterations. Default: `1000`

nexch

Number of cycles between exchange of data between processors and updating boundary conditions. Default: `2`

nitfo

Number of cycles using 1st-order spatial accuracy. Default: `0`

nordbc

Boundary condition calculation using 1$^{\mathrm{st}}$-order (`nordbc = 1`) or 2$^{\mathrm{nd}}$-order (`nordbc = 2`) accuracy. Default: `1`

ntran

Number of cycles between each transport properties update. Default: `1`

pressure_damping

Factor on the pressure update, leaving the update to other primitive variables unchanged. Only active if `closed_chamber` is `.true.`. Default: `0.5`

relax_direction

An integer flag indicating the relaxation direction to be either 1, 2, or 3, corresponding to i, j, or k. The value `0` is used for point_implicit assignment. This assignment will be applied to all the grid blocks unless specified otherwise in the optional `assign_tasks` file.

Default: `0` if `point_implicit = .true.`, otherwise `3`

rf_inv

Inviscid relaxation factor. Default: `3.0`

rf_vis

Viscous relaxation factor. Default: `1.0`

rf_chem

Chemical source term reduction factor sometimes useful to "ease in" simulations very close to equilibrium. This factor, which must be greater than 1.0 when it is used, changes the answers and must ultimately equal 1.0 in the final simulation. Default: `1.0`

rmstol

A real number to stop the iterations after the L2 norm residual reaches this value. Default: `1.0E-10` `sweep_direction`

An integer flag indicating the sweep direction to be either 1, 2, or 3, corresponding to i, j, or k. This assignment will be applied to all the grid blocks unless specified otherwise in the optional `assign_tasks` file.

Default: `3` if `point_implicit = .true.`, otherwise `1`

`unlimited`

A logical flag to turn off minmod limiting. Limiting is required for stability in flows with strong shocks. May not be needed in other situations. Default: `.false.`

`vel_bc_2nd_order`

A logical flag to force the velocity BC at the wall to second order even if the `nordbc = 1`. Default: `.false.`

5.4.10 Output Parameters

`blayer_flagset`

A character string identifier to select from predefined output variable sets to include in the `laura_blayer.dat` file. Default: `'standard'`

Options are:

`'standard'`
standard output variable list

`'orion'`
configuration controlled output list for Orion program usage

`'wallonly'`
only output wall variables, no edge variables

`'legacy'`
legacy variable output list prior to this implementation

`blflag_[NAME]`

Logical flags to specify output of property `NAME` to `laura_blayer.dat` file. The user may set individual flags (tedious), or choose from the predefined sets by using the `blayer_flagset` variable. Default: `.false.`

Supported `NAME`s, grouped by types:

Wall:

`rhow`, `pw`, `tw`, `tvw`, `hw`, `muw`, `spec_concw`, `spec_densw`, `tauwx`, `tauwy`, `tauwz`, `tauw_total`, `recell`, `qw_conv`, `qw_cat`, `qw_rad`, `qw_total`, `ch_conv`, `kappaw`, `utau`, `qw_cond`, `pcoeff`, `emissw`

Boundary-layer edge:

`rhoe`, `pe`, `te`, `tve`, `he`, `mue`, `spec_conce`, `spec_dense`, `ue`, `ve`, `we`, `me`, `delta`, `deltastar`, `theta`, `retheta`, `retheta_me`, `standoff`, `reue_tot`, `reue_par`, `vele_tot`, `vele_par`

Roughness height:

`rhok`, `pk`, `tk`, `tvk`, `hk`, `muk`, `spec_conck`, `spec_densk`, `uk`, `vk`, `wk`, `mk`, `kdelta`, `kdeltastar`, `ktheta`, `rek`, `rekk`, `kscdelta`, `kscdeltastar`, `ksctheta`, `velk_par`, `velk_tot`, `kplus`

`blayer_io_freq`

Number of cycles between saves of the `laura_blayer.dat` file – see Section 6 on page 64. A negative value makes the write-out frequency to be the same as `iterwrt`. Default: `-1`

`isurf_freq`

The number of cycles between saves of `laura_surface_dtXXXX.q` files, where `XXXX` will be replaced with an integer number, starting from `1`. A time accurate simulation is required to engage this command. Default: `-1`

`it_start`

An integer flag to be used as a starting point for numbering `laura_surface_dtXXXX.q` file. This flag is only applicable with time accurate flag `itime`. Default: `1`

`iterwrt`

Number of cycles between saves of all output files – see Section 6 on page 64. Default: `200`

`roughness_height`

A positive real value for roughness height (in meters). Used in several algebraic turbulence transition models. Also specified height at which blayer properties are to be interpolated and printed to `laura_blayer.dat` file. Note: Prior to v5.6, this variable was named `roughness`. Default : `1.0e-20`

`scallop_depth`

A positive real value to define the depth of scalloping for a flexible thermal protection system (FTPS) stretched over a series of inflatable toroids. This variable is to provide for ongoing development. Default: `1.0e-20` `scallop_length`

A positive real value to define the distance between scallops for a flexible thermal protection system (FTPS) stretched over a series of inflatable toroids. This variable is to provide for ongoing development. Default: `1.0e-20`

`tvref`

A positive real value to define an excited freestream vibrational temperature that differs from the translational temperature. This variable applies to high-enthalpy shock-tube simulations where nonequilibrium freestream vibrational states may be produced. Default: `tref`

`traj_time`

An optional positive real value to identify the time along a trajectory to assist in post-processing of data. Default: `0.0`

`project_name`

An optional character string to define a project title. Default: `'default'`

`case_name`

An optional character string to define a case title. Default: `'default'`

`tecplot_aero`

A logical flag to output all the aerodynamic data in a Tecplot-readable file, `laura_aero.dat`. Default: `.true.`

`write_extra_q_variables`

A logical flag to output all the computed variables such as laminar and turbulent viscosity, thermal conductivity, species diffusivity, specific heat, enthalpy, species pressure Jacobians, turbulent into `laura.q` file. Default: `.false.`

`write_number_density`

A logical flag to output species number density in $1/cm^3$ instead of their mass fractions. Default: `.false.`

5.4.11 Radiation Flags

`radiation`

A logical flag to enable coupling between radiation equation(s) and flow equations. See Section 8.2 on page 82 for coupled radiation procedure. Default: `.false.`

`radiation_input_only`

A logical flag to enable the creation of the input file `hara_out.m` to compute radiation outside of LAURA when `radiation` is `.false`. Default: `.false`.

`maxrad`

An integer number to assign the maximum number of calls to the radiation interface. The value of zero can be used for an unlimited number of times. Default: 0

`nrad`

The number of iterations between calls to HARA. Default: `3000`

`iinc_rad, jinc_rad`

Increment between i and j lines, respectively, at which radiation profile is computed. Interim lines are interpolated. Default: `3`

`frac_rad_new`

Relaxation factor on radiation. $\nabla(q_{rad}) = frac\_rad\_new \nabla(q_{rad})^n + (1 - frac\_rad\_new)\nabla(q_{rad})^{n-1}$. Default: `1.0`

`absorptivity`

The fraction of incident radiative energy absorbed by the wall. Affects surface energy balance and computation of radiative equilibrium wall temperature. Default: `1.0`

`tw_rad_flag`

An integer flag to engage the Tauber-Wakefield approximation for radiation cooling on surface-energy balance (on=1/off=0). Default: 0

5.4.12 Solid Surface Boundary Condition Flags

`catalysis_model_0`

A character identifier for selecting a catalysis model for surface type 0 (see Section 5.3 on page 12). This flag is good only for multispecies reacting gases and will be ignored for single-species gas models. The catalysis model name must be surrounded by quotation marks, e.g., ‘ ’. Default: `‘super-catalytic’`

Available catalysis models are:

`‘CCAT-ACC’`

This option uses the following surface catalytic coefficients [8] for catalyzing atomic oxygen and nitrogen to molecular oxygen and nitrogen, respectively.

$$\gamma_O = \begin{cases} 13.5e^{-8350/T_w^*} & T_w^* \leq 1359.0\ K \\ 5.0 \times 10^{-8} e^{18023/T_w^*} & T_w^* > 1359.0\ K \end{cases} \quad (9)$$

$$\gamma_N = \begin{cases} 4.0e^{-7625/T_w^*} & T_w^* \leq 1475.0\ K \\ 6.2 \times 10^{-6} e^{12100/T_w^*} & T_w^* > 1475.0\ K \end{cases} \quad (10)$$

where T_w^* is calculated as

$$T_w^* = \begin{cases} \min(\ \max(1255.0, T_w),\ 1659.0) & \text{for } \gamma_O \\ \min(\ \max(1255.0, T_w),\ 1900.0) & \text{for } \gamma_N \end{cases} \quad (11)$$

‘`CSiC`’

This option uses the following surface catalytic coefficients [8] for catalyzing atomic oxygen and nitrogen to molecular oxygen and nitrogen, respectively.

$$\gamma_O = 6.415 \times 10^{-4} e^{3498.4/T_w^*} \quad (12)$$

$$\gamma_N = 3.993 \times 10^{-4} e^{4402/T_w^*} \quad (13)$$

where T_w^* is given as

$$T_w^* = \min(\ \max(1100.0, T_w),\ 1920.0) \quad (14)$$

‘`CSiC-SNECMA`’

This option uses the following surface catalytic coefficients [8] for catalyzing atomic oxygen and nitrogen to molecular oxygen and nitrogen, respectively.

$$\gamma_O = \gamma_N = 9.593 \times 10^{-5} e^{7002.9/T_w^*} \quad (15)$$

where T_w^* is given as

$$T_w^* = \min(\ \max(1350.0, T_w),\ 1920.0) \quad (16)$$

‘`equilibrium-catalytic`’

This option sets species concentrations to equilibrium values at wall pressure and temperature based on elemental mass fractions in the cell above the solid surface boundary.

‘`fully-catalytic`’

This option assumes that all the atomic and ionized oxygen, nitrogen, carbon, and so forth catalyzes to molecular oxygen, nitrogen, carbon, and so on, respectively; i.e.,

$$\gamma_O = \gamma_N = \gamma_C = ... = 1 \quad (17)$$

`‘non-catalytic’`

This option assumes that no atomic or ionized oxygen, nitrogen, carbon, and so forth catalyzes to molecular oxygen, nitrogen, carbon, and so on, respectively; i.e.,

$$\gamma_O = \gamma_N = \gamma_C = ... = 0 \tag{18}$$

`‘RCC-LVP’`

This option uses the following surface catalytic coefficients [8] for catalyzing atomic oxygen and nitrogen to molecular oxygen and nitrogen, respectively.

$$\gamma_O = \begin{cases} 7.5e^{-8283/T_w^*} & T_w^* \leq 1499.0\ K \\ 2.5 \times 10^{-7} e^{17533/T_w^*} & T_w^* > 1499.0\ K \end{cases} \tag{19}$$

$$\gamma_N = \begin{cases} 6.0 \times 10^{-2} e^{-2605/T_w^*} & T_w^* \leq 1529.0\ K \\ 1.5 \times 10^{-5} e^{10080/T_w^*} & T_w^* > 1529.0\ K \end{cases} \tag{20}$$

where T_w^* is calculated as

$$T_w^* = \begin{cases} \min(\ \max(1255.0, T_w),\ 1799.0) & \text{for } \gamma_O \\ \min(\ \max(1255.0, T_w),\ 1954.0) & \text{for } \gamma_N \end{cases} \tag{21}$$

`‘Scott-RCG’`

This option uses the following surface catalytic coefficients [9] for catalyzing atomic oxygen and nitrogen to molecular oxygen and nitrogen, respectively.

$$\gamma_O = 16.0e^{-10271/T_w} \quad 1400 \leq T_w \leq 1650 \tag{22}$$

$$\gamma_N = 7.14 \times 10^{-2} e^{-2219.0/T_w} \quad 1090 \leq T_w \leq 1670 \tag{23}$$

The same equations will be used even if the wall temperature, T_w, is out of the specified range, in which case a warning will be issued to the `stdout`.

`‘Stewart-RCG’`

This option uses the following surface catalytic coefficients [8] for catalyzing atomic oxygen and nitrogen to molecular oxygen and nitrogen, respectively.

$$\gamma_O = \begin{cases} 5.0 \times 10^{-3} e^{-400/T_w} & T_w \leq 502 \\ 1.6 \times 10^{-4} e^{1326/T_w} & 502 < T_w \leq 978 \\ 5.2e^{-8835/T_w} & 978 < T_w \leq 1617 \\ 39 \times 10^{-9} e^{21410/T_w} & 1617 < T_w \end{cases} \tag{24}$$

$$\gamma_N = \begin{cases} 5.0 \times 10^{-4} & T_w \leq 465 \\ 2.0 \times 10^{-5} e^{1500/T_w} & 465 < T_w \leq 905 \\ 10.0e^{-10360/T_w} & 905 < T_w \leq 1575 \\ 6.2 \times 10^{-6} e^{12100/T_w} & 1575 < T_w \end{cases} \tag{25}$$

`‘SiC-cloth’`

This option uses the following surface catalytic coefficients [10] for catalyzing atomic oxygen and nitrogen to molecular oxygen and nitrogen, respectively.

$$\gamma_O = min[1.0, 786.37e^{-6.0287\times10^{-3}T_W}] \tag{26}$$

and

$$\gamma_N = min[1.0, 3.6297 \times 10^{-5} e^{4.8279\times10^{-3}T_W}] \tag{27}$$

where

$$T_W = min[2000, max(1000, T_W)] \tag{28}$$

`‘SiC-composite’`

This option uses the following surface catalytic coefficients [8] for catalyzing atomic oxygen and nitrogen to molecular oxygen and nitrogen, respectively.

$$\gamma_O = min[1.0, 726.47e^{-7.2753\times10^{-3}T_W}] \tag{29}$$

and

$$\gamma_N = min[1.0, 3.4588 \times 10^{-5} e^{4.4116\times10^{-3}T_W}] \tag{30}$$

where

$$T_W = min[2000, max(1000, T_W)] \tag{31}$$

`‘super-catalytic’`

This option sets the species mass fractions to free stream values as defined in `tdata`—see Section 5.5 on page 43.

`‘Zoby-RCG’`

This option uses the following surface catalytic coefficients [11] for catalyzing atomic oxygen and nitrogen to molecular oxygen and nitrogen, respectively.

$$\gamma_O = 9.41 \times 10^{-3} e^{-658.9/T_w} \quad 900 \le T_w \le 1500 \tag{32}$$

$$\gamma_N = 7.14 \times 10^{-2} e^{-2219.0/T_w} \quad 1090 \le T_w \le 1670 \tag{33}$$

The same equations will be used even if the wall temperature, T_w, is out of the specified range; in that case a warning will be issued to the screen.

`co2_catalysis_model`

A logical flag that engages the CO_2 heterogeneous catalysis model proposed by Mitcheltree and Gnoffo [12]. Model requires `catalysis_model_O = ‘fully catalytic’` and CO, CO_2, and O to be present in the flow field. Default: `.false`.

`emiss_a_0`, `emiss_b_0`, `emiss_c_0`, `emiss_d_0`

Real number values to calculate emissivity, ϵ, for solid surface boundary type (see Section 5.3 on page 12) from the following equation:

$$\epsilon = \epsilon_a + \epsilon_b T_w + \epsilon_c T_w^2 + \epsilon_d T_w^3 \tag{34}$$

where T_w is the surface temperature. Values for ϵ_a, ϵ_b, ϵ_c, and ϵ_d are defined by `emiss_a_0`, `emiss_b_0`, `emiss_c_0`, and `emiss_d_0`, respectively. Default: `emiss_a_0=0.89`, `emiss_b_0=0.0`, `emiss_c_0=0.0`, `emiss_d_0=0.0`

`emiss_min`, `emiss_max`

Real number values to bound the emissivity used in the Equation 34. Note: This bound only applies to a variable emissivity function. Default: 0.4 and 0.9 for `emiss_min` and `emiss_max`, respectively.

`ept`

An underrelaxation parameter for radiative equilibrium wall temperature:

$$T_w^{n+1} = (1 - \texttt{ept})T_w^n + \texttt{ept}T_w^{n+1} \tag{35}$$

where n denotes the iteration level, T_w^{n+1} is the most recent value of the wall temperature, and T_w^n is the value of the wall temperature from the previous iteration as adjusted by the previous application of this formula. Default: `0.01`

`equil_surf_temp_floor_0`

The minimum temperature of the equilibrium catalytic surface for the purpose of determining equilibrium constants. Raising this value to 1000 may aide convergence if the transient solution includes low surface temperatures. If the converged surface temperature is less than this value than surface species mass fractions will be in error. Default: `100`.

`surface_temperature_type_0`

Character identifier for surface temperature model selection. Default: `'constant'`

Options are:

`'adiabatic'`

The surface temperature will be such that conduction heat transfer between the surface and the gas adjacent to the surface is zero.

`'adiabatic catalytic'`

The surface temperature will be such that the sum of conduction and diffusion heat transfer between the surface and the gas adjacent to the surface is zero.

`‘constant’`

The surface temperature stays constant as given by `twall_bc` value.

`‘radiative equilibrium’`

The surface temperature is calculated so that the heat flux to the wall, q_w, is in equilibrium with radiation heat flux:

$$q_w = \epsilon \sigma T_w^4 \tag{36}$$

where σ is the Stefan-Boltzmann constant, and ϵ is the surface emissivity defined by `emiss_a_0`, `emiss_b_0`, `emiss_c_0`, `emiss_d_0` values.

`surface_group_name_0`

Character descriptor for surfaces with solid surface boundary types (see Section 5.3 on page 12). Any character can be specified to group solid surface boundaries. Default: `‘default surface 0’`

`t_rad_eq_max`

Maximum allowed radiative equilibrium wall temperature. It is sometimes convenient to limit this temperature in anticipation of coupling ablation in subsequent runs. Default: `1.E+06`

`twall_bc`

Initial wall temperature for solid surface boundaries, K. (See Section 5.3 on page 12.) The wall temperature stays constant as specified by this parameter if `surface_temperature_type_0 = ‘constant’`. Default: `500.0`

5.4.13 Surface Recession Flags

`shape_change`

A logical flag engaging the geometry shape change due to ablation. Default : `.false`. A trajectory file is required if `shape_change=.true`.—see Section 5.13 on page 56.

5.4.14 Thermochemical Nonequilibrium Flags

`augment_kinetics_limiting`

A logical flag engaging extra limiting on reaction rates by modifying upper and lower temperature limits in forward and backward rates. Upper and lower limits on rate constants may be specified for individual reactions in the file `kinetic_data`. Default: `.false`.

`chem_flag`

An integer flag to engage the chemical source term for nonequilibrium flow (on=1/off=0) for nonperfect gas cases. Default: `1` `force_neutrality`

A logical flag to overwrite the result of the electron continuity equation and force the number of electrons to equal the number of positive charged ions. Default: `.true.`

`therm_flag`

An integer flag to engage the thermal source term for nonequilibrium flow (on=1/off=0) for nonperfect gas cases. Default: `1`

`cpiv_min_factor`

The floor for vibrational-electronic heat capacity for species `i` is set to `cpiv_min_factor` times the local value of the translational-rotational heat capacity for species `i`. Raising the floor helps suppress severe vibrational-electronic temperature undershoots that sometimes occur in two-temperature model simulations ahead of a strong bow shock. Values as large as `0.01` appear to work without a significant adverse effect on the aerothermodynamic environment. Default: `0.0001`

`park_vib_relax`

A logical flag engaging Park values for vibrational constants required for vibrational-translational energy relaxation. Engaging this option is recommended for radiation computations. Default: `.false.`

`eii_limit`

An upper limit for the electron-impact ionization source terms. Values on the order of 20–100 will allow for convergence using more aggressive relaxation parameters, but will change the nonequilibrium chemistry. Default: `1e+20`

5.4.15 Time Accurate Flags

`itime`

An integer flag to engage time accuracy: `itime = 0` produces 0th-order in time, `itime = 1` produces 1st-order in time, `itime = 2` produces 2nd-order in time. This flag is superseded by the value of `time_step`. A positive value for `time_step` forces temporal accuracy of at least first-order. A negative or unspecified value for `time_step` forces `itime=0`. Default: `0`

`subiters`

An integer number to specify the number of iterations between each time step for time-accurate simulations. Default: `10` `time_step`

A positive real value to specify the dimensional time step for time accurate simulation. A negative or zero value supersedes the `itime` value and disables time accurate simulation. Default: `-1`

5.4.16 Trajectory Related Flags

`trajectory_data_point`

Pickup simulation from this line in the file `laura_trajectory_data`.

Default: `0` if `laura_trajectory_data` is not present.

Default: `1` if `laura_trajectory_data` is present.

Note that the reference quantities are defined consistently across the trajectory using the values supplied in the namelist above. The reference quantities are NOT reset according to the time varying free stream conditions. Consequently, dimensionless values of density and velocity in the free stream will not generally equal 1. Dimensionless values of total enthalpy in the free stream will not equal 0.5, which may impact post-processing tools that are key on a specific number for total enthalpy to detect the boundary-layer edge.

The algorithm is most efficient if the restart solution is converged (or nearly converged so that line-implicit relaxation may be applied) at free stream conditions equal to the reference quantities in `laura_namelist_data`. If one wishes to pick up a computation for `trajectory_data_point` > 1 then it is best to start at the converged restart file for the previous trajectory point. Restart files and post processing files for each trajectory point have the trajectory point number included as part of the root name. Thus, if one wants to pick up at trajectory point 12, one should copy `laura_0011.rst` to `laura.rst` and (if the grid is being updated) `laura_0011.g` to `laura.g`. This advice is offered because restart solutions are converted according to the ratio of density and velocity between adjacent trajectory points to bring interior initial conditions closer to the new inflow boundary conditions.

5.4.17 Turbulence and Transition Options

`coupled_tke`

A logical flag indicating if turbulent kinetic energy is treated as part of the total energy in the energy equation. Default:`.true.`

`prandtl_turb`

Turbulent Prandtl number. Default: `0.9` `prod_to_des_limit`

A parameter to limit the production term for turbulent kinetic energy as a factor of the destruction term. Default: `20.`

`schmidt_turb`

Turbulent Schmidt number. Default: `0.9`

`turb_int_inf`

The farfield turbulent kinetic energy nondimensionalized by the square of the reference velocity. If negative, use recommended farfield boundary conditions: $(\rho k)_\infty = 0.01\mu_\infty$ and $(\rho\omega)_\infty = 2$. Default: `-0.0001`

`turbulence_model`

A character string identifier for selection of turbulence models. Default: `'no_model'`

Options are:

- `'baldwin-lomax'`
  legacy version of the Baldwin-Lomax algebraic model. [13]
- `'baldwin-lomax-plus'`
  The Baldwin-Lomax algebraic model with transition options.
- `'cebeci-smith'`
  The legacy version of the Cebeci-Smith algebraic model. [13]
- `'cebeci-smith-plus'`
  The Cebeci-Smith algebraic model with transition options. [13]
- `'no_model'`
  Laminar flow, i.e., no turbulence model.

`turb_vis_ratio_inf`

Farfield ratio of turbulent to laminar viscosity. Default: `0.001`

`xtr_byblock_file`

A logical variable to specify reading transition information for each block through an external file `laura.trn`. Replaces the `itrn_file_flag` variable, which is deprecated as of v5.5 but still included. The `laura.trn` file allows for specification of the transition state, transition location, and transition length scale factor. "by-block" specifications of laminar or turbulent, but not transitional, will override any of the `alg_trans_by_[NAME]` options discussed below. Default: `.false.`

The `laura.trn` file format requires a line for each block with the values

```
n  isturb(n)  xtr_location(n)  xtr_scale_factor(n)
```

where `n` is the block number; `isturb(n)` is the boundary layer state with values of `-1`, `0`, `1` for laminar, transitional, and turbulent flow, respectively; `xtr_location(n)` is the transition location value in terms of distance, x; and `xtr_scale_factor(n)` is the transition length scale factor.

This option is employed for situations where the state of a given block must be specified. Examples include defined a block with one (or more) boundary-layer trips that force the flow to be turbulent, or wake/aftbody blocks where algebraic turbulence models are not relevant and laminar flow is enforced. Note that for internal consistency, if the transition length within a block is greater than the block length, the transition length is rescaled to force the flow to reach fully turbulent conditions at the end of the block.

Note: The byblock options may produce aphysical results if the physical x-axis and the grid `i`-index are not aligned.

<u>`xtr_location`</u>

Real variable to globally specify an x-location for transition. Replaces the `xtr` variable as of v5.6. Default: `0.0`

<u>`xtr_scale_factor`</u>

Real variable to globally specify a multiplier to the default transition zone length. Default: var1.0

The global transition location along the x-axis in grid units. Engaged only for algebraic models. This value will be overridden by the `transition_location` value in the `laura.trn` file should the file be available.

<u>`alg_trans_by_[NAME]`</u>

Logical variables to specify one (or more) transition options for use with algebraic turbulence model subroutines. The options are available only for the `baldwin_lomax_plus` and `cebeci_smith_plus` turbulence models. More than one model may be applied, in which case the earliest predicted transition onset from all selected models will be applied. Default for all transition models is `.false.` except for `alg_trans_by_fully_turbulent`, which is `.true.`.

`NAME` options:

`alg_trans_by_x_distance`

Transition onset in terms of the lengthwise distance from a given location specified by the variable `alg_trans_location_x0`. Note that this model defines a straight-line, not surface distance. This model is more applicable to long geometries (e.g., the Shuttle Orbiter).

`alg_trans_by_r_distance`

Transition onset in terms of the radial-distance from a given location specified by the variable `alg_trans_location_r0`. Note that this model defines a straight-line, not surface distance. This model is more applicable to blunt geometries (e.g., planetary probes).

`alg_trans_by_x_3d_distance`

Transition onset in terms of the three-dimensional distance from a given location specified by the variables `alg_trans_location_3dx0`, `alg_trans_location_3dy0`, and `alg_trans_location_3dz0`. Note that this model defines a straight-line, not surface distance.

`alg_trans_by_retheta_smooth`

Transition onset in terms of the momentum-thickness Reynolds number specified by `alg_trans_value_retheta0`.

`alg_trans_by_retheta_medge_smooth`

Transition onset in terms of the momentum-thickness Reynolds number divided by edge Mach number specified by `alg_trans_value_retheta0_medge0`.

`alg_trans_by_roughness_pant`

Transition onset in terms of distributed roughness criteria based on PANT wind tunnel data, with height specified by variable `roughness_height`. Also requires setting `roughness` to `.true.`.

`alg_trans_by_roughness_reda`

Transition onset in terms of distributed roughness criteria based on Reda ballistic range data, with height specified by variable `roughness_height`. Also requires setting `roughness` to `.true.`.

`alg_trans_by_roughness_combined`

Transition onset in terms of distributed roughness criteria based on combination of flight, wind tunnel, and ballistic range data, with height specified by variable `roughness_height`. Also requires setting `roughness` to `.true.`.

`alg_trans_by_scallop`

Transition onset in terms of the flexible thermal protection system scalloping. Developmental model not yet recommend for use.

`alg_trans_by_fully_turbulent`

Default transition option that forces the entire flow field to be treated as fully-turbulent.

alg_inter_by_[NAME]

Logical variables to specify a mathematical function for transition intermittency with NAME options:

alg_inter_by_exp

Exponential function as per Dhawan-Narasimha. [14] Default: .true.

alg_inter_by_tanh

Hyperbolic function. Default: .false.

alg_inter_by_trip

Zero-length transition as per a boundary-layer trip. Default: .false.

alg_inter_factor

A real variable used to compute the exponential-based intermittency factor to specify percentile value to treat as "fully-turbulent." There is generally no reason for a user to modify this, but the option is provided just in case. Default: 0.99

alg_trans_length

A real variable used in scaling the transition length for the distance-based transition options. Default: 1.0

alg_trans_location_3dx0

The X-component of the 3D transition location specification for use with alg_trans_by_3d_distance option. Default: 0.0

alg_trans_location_3dy0

The Y-component of the 3D transition location specification for use with alg_trans_by_3d_distance option. Default: 0.0

alg_trans_location_3dz0

The Z-component of the 3D transition location specification for use with alg_trans_by_3d_distance option. Default: 0.0

alg_trans_location_x0

The length specification for the transition location for use with alg_trans_by_x_distance option. Default: 0.0

alg_trans_location_r0

The radial distance specification for the transition location for use with alg_trans_by_r_distance option. Default: 0.0

alg_trans_value_retheta0

The momentum-thickness Reynolds number value for transition using alg_trans_by_retheta_smooth option. Default: 250.0

alg_trans_value_retheta_medge0

The momentum-thickness Reynolds number / edge Mach number value for transition using alg_trans_by_retheta_smooth option. Default: 300.0

5.4.18 Venting Boundary Condition Flags

vacuum_pressure_coefficient

This nondimensional parameter sets the pressure coefficient behind type 6 boundaries (see Section 5.3 on page 12), which forces flow out of the domain. This coefficient is defined as

$$\texttt{vacuum\_pressure\_coefficient} = \frac{p_0 - p_\infty}{2\rho_\infty u_\infty^2}. \tag{37}$$

where p_0 is the back pressure where the gas is venting out, and p_∞ is the farfield pressure. Default: 0.0

Note: To activate this option vacuum_pressure_factor must be less than or equal to zero.

vacuum_pressure_factor

The factor on the pressure across a type 6 boundary to force flow out of the domain.

$$\texttt{vacuum\_pressure\_factor} = \frac{p_0}{p_1}, \tag{38}$$

where p_1 is the pressure just before the gas vents out. Default: 0.01

5.5 tdata

The gas model is defined in this file. It contains a list of keywords, sometimes followed by numeric values, which identify components of the gas model. One or more spaces must be present between a keyword and values when appearing on the same line. Spaces may appear to the left or right of any keyword. The first line of the file must not be blank, however.

The following subsections describe available gas model options.

5.5.1 Perfect Gas

The perfect-gas option is engaged with either of the following keywords: perfect_gas, PERFECT_GAS, Perfect_Gas, or Perfect_gas.

If no further data are provided in this file, this single line tdata file will assume the following parameter values in SI units:

```
gamma   =   1.4
mol_wt  =  28.8
suther1 =   0.1458205E-05
suther2 = 110.333333
prand   =   0.72
```

Here, `gamma` is the gas specific heat ratio, `mol_wt` is the gas molecular weight, `prand` is the gas Prandtl number, and `suther1` and `suther2` are the first and second Sutherland's viscosity coefficients, s_1 and s_2, respectively, defined as

$$\mu = s_1 \frac{T^{3/2}}{T + s_2} \tag{39}$$

These values can be modified and explicitly defined in `tdata` by the keyword `&species_properties` in the second line followed by the gas parameters and `/` at the last line of the file. For example,

```
perfect_gas
&species_properties
gamma = 1.4
mol_wt = 28.0
suther1 = 0.1E-05
suther2 = 110.3
prand   = 0.7
/
```

5.5.2 Equilibrium Gas

To engage the Tannehill curve fits for thermodynamic and transport properties of equilibrium air [15], the following keyword should be used in the first line of the `tdata` file: `equilibrium_air_t`. No additional inputs or files are required to engage the Tannehill option for equilibrium air.

To use a table look-up capability for equilibrium gases [16], the following keyword should be placed in the `tdata` file, instead: `equilibrium_air_r`. Note that this option still uses the Tannehill transport properties.

5.5.3 Mixture of Thermally Perfect Gases

If the gas is a mixture of thermally perfect gases and multispecies transport solution is desired the species names followed by their mass fractions must be provided in the `tdata` file. The thermal state of the gas may be defined as the first entry by either of the following flags:

`one`

This flag assumes that all the species are thermally in an equilibrium state. That is translational temperature, T, and vibrational temperature, T_v are equal. This is known as the *one-temperature*, 1-T, model.

`two`

This flag assumes that energy distribution in the translational and rotational modes of heavy particles (not electrons) are equilibrated at translational temperature, T, and all other energy modes (vibrational, electronic, electron translational) are equilibrated at vibrational temperature, T_v. This is known as the *two-temperature*, 2-T, model.

`FEM`

This option, called Free-Energy Minimization, causes the species continuity equations to be replaced with elemental continuity equations and equilibrium relations for remaining species.

One temperature model is assumed if the thermal state of the gas is not provided in the first line of the `tdata` file. In this case, the first line must contain species information. Note, the first line must not be blank.

Subsequent file entries include species names and their mass fractions at freestream/farfield boundary. Only one species per line is allowed. The species mass fraction at the boundary is defined in the same line as the species name separated by one or more spaces. If no value appears to the right of the species name then that species is assumed not to be present at the boundary but may be produced through chemical reactions elsewhere in the flowfield.

Example 1: 1-T, 5-species air model: In this example, only molecular oxygen and nitrogen are present on the freestream/farfield boundary, but atomic nitrogen and oxygen and nitric oxide may be produced elsewhere in the flow field due to chemical reactions.

```
one
N2  .767
N
O2  .233
O
NO
```

Example 2: 2-T, 11-species air model: In this example, the gas is assumed to be a mixture of 11 thermally perfect gases. A solution to a thermal nonequilibrium state of the gas is also desired (2-T model).

```
two
N2  .767
N
O2  .233
O
NO
```

```
O2+
O+
NO+
e-
```

5.6 Simulation Dependent Files

5.7 `assign_tasks`

This file defines sweep and relaxation directions for only grid blocks that do not fall under the default specification in the namelist. Each line has five integers[12] that are separated by at least one space. These integers correspond to `nbk`, `mbk`, `mbr`, `lstrt`, and `lstop` where

<u>`nbk`</u>

Block number.

<u>`mbk`</u>

Sweep direction. The assigned value can be either `1`, `2`, or `3`, corresponding to i-, j-, or k-direction, respectively.

<u>mbr</u>

The line relaxation direction. Note: must be different than the sweep direction.

Options are:

`0`: Point-implicit, i.e., no line-relaxation

`1`: Line-implicit along i coordinate

`2`: Line-implicit along j coordinate

`3`: Line-implicit along k coordinate

<u>lstrt</u>

The starting grid index in the sweep direction. Typically `1` .

<u>lstop</u>

The ending grid index in the sweep direction. Typically `0`, which is shorthand for the maximum index.

Options are:

`0`: maximum index

`-1`: makes the block inactive

[12]The code will not read data beyond the fifth column.

When starting a new simulation where the k-coordinate runs from the vehicle surface to the freestream boundary, sweeping in the k-coordinate and solving point-implicitly is recommended, i.e., `mbk = 3` and `mbr = 0`. This can be done simply by `point_implicit = .true.` in the namelist file. After the shock has stabilized, switch to streamwise sweeps and solve line-implicitly along the k-coordinate, i.e., `mbk = 1` and `mbr = 3`. The namelist variable `point_implicit = .false.` will do the same for all the blocks.

5.7.1 Example 1: Multiple Blocks per CPU or Vice-versa

Suppose the grid has 2 blocks, but the number of available processors is not the same as the number of grid blocks. The user does not need to change any of the input files and can simply specify the number of processors available, e.g.,

```
 % mpirun -np [#] laura_mpi
```

where `#` is the number of available processors. LAURA automatically assigns processors to blocks such that each processor receives approximately the same number of grid cells.

5.7.2 Example 2: Deactivating Grid Blocks

Suppose the grid has 50 blocks, but only blocks 20–25 need to be updated. In this case, `assign_tasks` would contain blocks 20–25 with `-1` on the 5th column.,

```
  20  3   0   1     -1     1
  21  3   0   1     -1     2
  22  3   0   1     -1     3
  23  3   0   1     -1     4
  24  3   0   1     -1     5
  25  3   0   1     -1     6
  nbk mbk mbr lstrt lstop dummy
```

5.8 hara_namelist_data

This file controls the radiation models used by the HARA radiation module [17,18]. It is optional for coupled radiation simulations. If it is not present, then the code automatically chooses the radiative mechanisms associated with species present in the flowfield (and have number densities greater than 1000 particles/cm^2), and other options are set to the defaults listed in the following section. For users not experienced in shock-layer radiation, it is recommended that this `hara_namelist_data` file not be applied (meaning it is removed from the working directory), therefore allowing the radiative mechanisms to be automatically chosen and the default model options applied.

5.8.1 Specifying radiation mechanisms for atomic species

The treatment of radiation resulting from atomic lines, atomic bound-free and free-free photoionization (referred to here as atomic continuum), and negative ion continuum is available for atomic carbon, hydrogen, oxygen, and nitrogen. These mechanisms are specified through the following binary flags (on=1/off=0). If any of these flags are not present in `hara_namelist_data`, then that flag is set to true only if the number density of the associated atomic species is greater than 1000 particles/cm^2 somewhere in the flowfield.

`treat_[?]_lines`

A binary flag to enable the treatment of atomic lines for species `[?]`, where `[?]` can be c, h, n, and o, for atomic carbon, hydrogen, nitrogen and oxygen, respectively.

`treat_[?]_cont`

A binary flag to enable the treatment of atomic bound-free and free-free continuum for species `[?]`, where `[?]` can be c, h, n, and o, for atomic carbon, hydrogen, nitrogen and oxygen, respectively.

`treat_[?]_other`

A binary flag to enable the treatment of the negative-ion continuum for species `[?]`, where `[?]` can be c, h, n, and o, for atomic carbon, hydrogen, nitrogen and oxygen, respectively.

5.8.2 Specifying radiation mechanisms for molecular species

The treatment of radiation resulting from numerous molecular band systems is available through the following flags (0 = off, 1 = SRB, 2 = LBL, 3 = opacity-binning for CO 4th-Positive, CN Violet, CN Red, and CO_2 IR). The smeared rotational band (SRB) approach applies a simplified and efficient treatment of each molecular band system, which is accurate for optically thin conditions. The line-by-line (LBL) approach is a detailed, but highly inefficient, treatment of each molecular band system. Band systems that do not satisfy the optically-thin requirement for SRB are typically limited to the CO 4th-Positive and CO_2 IR band systems for Mars entry and CN Red and Violet for Titan entry. To avoid the computational time required to accurately model these bands with LBL, the opacity-binning approach developed by Johnston et al. [19] was developed. This approach, which reproduces the LBL results for nonoptically-thin conditions within 2%, is only available for CO 4th-Positive, CN Violet, CN Red, and CO_2 IR. If any of these flags are not present in `hara_namelist_data`, then that flag is set to the SRB option only if the number density of the associated molecular species is greater than 1000 particles/cm^3 somewhere in the flowfield. Additional band systems are listed in Appendix B on page 91.

These additional band systems are generally considered negligible relative to those listed in this section, and therefore, for computational efficiency, they are not engaged by default. Definitions of each band system and the modeling data applied are discussed in Refs. [17, 20].

`treat_band_c2_swan`

A flag activating the C_2 Swan band system.

`treat_band_c2h`

A flag activating the C_2H band system.

`treat_band_c3`

A flag activating the C_3 and Vacuum Ultra-Violet (VUV) band systems.

`treat_band_cn_red`

A flag activating the CN red band system.

`treat_band_cn_violet`

A flag activating the CN violet band system.

`treat_band_co4p`

A flag activating the CO 4th-Positive band system.

`treat_band_co_ir`

A flag activating the CO X-X band system.

`treat_band_co2`

A flag activating the CO_2 Infrared band system.

`treat_band_h2_lyman`

A flag activating the H_2 Lyman band system.

`treat_band_h2_werner`

A flag activating the H_2 Werner band system.

`treat_band_n2fp`

A flag activating the N_2 1st-Positive band system.

`treat_band_n2sp`

A flag activating the N_2 2nd-Positive band system.

`treat_band_n2pfn`

A flag activating the N_2^+ 1st-Negative band system.

treat_band_n2_bh1

A flag activating the N_2 Birge-Hopfield I band system.

treat_band_n2_bh2

A flag activating the N_2 Birge-Hopfield II band system.

treat_band_no_beta

A flag activating the NO beta band system.

treat_band_no_delta

A flag activating the NO delta band system.

treat_band_no_epsilon

A flag activating the NO epsilon band system.

5.8.3 Atomic line models

There are various models available for atomic line radiation, one of which must be chosen for each species that engages atomic line radiation (as specified using `treat_[?]_lines`). This choice of atomic line model is made using the following flags. The listed defaults are applied if the individual flag is not present in `hara_namelist_data`, or if `hara_namelist_data` is not present in the working directory. All model types in this category must be surrounded by quotation marks, e.g., ‘ ’.

c_atomic_line_model, h_atomic_line_model

A character identifier for selecting the atomic line model for atomic carbon or hydrogen. Presently, the only available option is the model compiled in Ref. [20], which is referred to here as the Complete Line Model (CLM). Default : `‘clm’`

n_atomic_line_model, o_atomic_line_model

A character identifier for selecting the atomic line model for atomic nitrogen or oxygen. The available models are compiled and compared in Ref. [17], which is referred to here as the Complete Line Model (CLM). Default : `‘clm’` Available models are:

`‘all multiplets’`

This model treats all lines as grouped multiplets. This significantly reduces the number of lines treated as well as the computational expense. However, this grouped multiplet approximation will lead to errors for nonoptically-thin conditions.

`‘clm’`

This model, which stands for Complete Line Model, applies the individual treatment of strong atomic lines while applying multiplet averages for weak lines. This is the recommended model.

5.8.4 Electronic state population models

These flags specify the model applied for predicting the electronic state populations of atoms and molecules. The listed defaults are applied if the individual flag is not present in `hara_namelist_data`, or if `hara_namelist_data` is not present in the working directory. All model types in this category must be surrounded by a quotation marks, e.g., ‘ ’.

Atomic electronic states

The electronic state populations for atoms are required for computing atomic line and photoionization emission and absorption. The compilation and comparison of the available models are presented in Ref. [18].

`n_electronic_state`, `o_electronic_state`, `c_electronic_state`, `h_electronic_state`

A character identifier for selecting the electronic state model for atomic nitrogen and oxygen. Default: `‘CR’`

Available models are:

`‘boltzmann’`

Same as for `c_electronic_state`

`‘Gally_1st_order_LTNE’`

Same as for `c_electronic_state`

`‘CR’`

Applies the detailed Collisional Radiative (CR) model developed in Ref. [18].

Molecular electronic states

The electronic state populations for molecules are required for computing molecular band emission and absorption. The compilation and comparison of the available models are presented in Refs. [18, 21].

`molecular_electronic_state`

A character identifier for selecting molecular electronic state for all molecular band systems. Default: `‘CR Park’`

Available models are:

`‘boltzmann’`

Applies Boltzmann population of electronic states.

`‘CR Park’`

Applies a detailed Collisional Radiative model considering both heavy-particle and electron impact transitions. Some molecular states are still assumed Boltzmann with this model because no data are presently available for the CR model.

5.8.5 Other flags

`use_triangles`

A logical flag specifying whether optically-thin atomic lines are modeled as triangles to reduce computational time. This option has shown to result in a negligible loss of accuracy while greatly reducing the computational time, [17] and is therefore recommended. Default : `.true`. Note: This flag is automatically set to `.true`. when `n_` or `o_atomic_line_model= ‘clm’` — see Section 5.8.3 on page 50.

`use_edge_shift`

A logical flag to engage the photoionization edge shift [17] for atomic bound-free radiation. (on=1/off=0). Default: `.true`.

5.9 jdata

This file defines jet chamber conditions at faces of blocks with characteristic boundary conditions [22]. Jet species names and their mass fractions are also specified in this file. There must be one entry for each jet. Each jet can have a different chamber condition and species composition.

5.9.1 Multispecies Jet Composition

Consider a system with two jets; i.e., there are two characteristic boundary condition entries in the `laura_bound_data` file. N_2 and H_2 are being fired from one of the jets while pure H_2 is being used for the other jet. Note: The sum of the species mass fraction must equal 1.0 or the code stops with an error message. In this example, the `jdata` file should look like the following:

```
&jet_properties
nozzle_grid_block     = 9
plenum_t0             = 401.7
plenum_p0             = 2434000.
jet_number_of_species = 2
/
```

```
N2 0.8
H2 0.2

&jet_properties
nozzle_grid_block     = 6
plenum_t0             = 105.7
plenum_p0             = 10250.
/
H2 1.0
```

It should be noted here that the `tdata` must contain all the jet effluent species. The default value for `jet_number_of_species` is `1`. Jet properties can be placed in this file in a random order.

5.9.2 Perfect Gas Jet Composition

If `perfect_gas` is specified in the first line of the `tdata` file, the code ignores jet species composition even if they are left in `jdata` and assumes perfect gas with the same molecular weight and Prandtl number as specified in the `tdata` file.

```
&jet_properties
nozzle_grid_block     = 19
plenum_t0             = 401.7
plenum_p0             = 2434000.
jet_number_of_species = 1
/
N2 0.8
H2 0.2

&jet_properties
nozzle_grid_block     = 36
plenum_t0             = 105.7
plenum_p0             = 10250.
/
H2 1.0

...
...
...
```

5.10 `kinetic_data`

This file defines possible chemical reactions and is optional. By default, `kinetic_data` is read from `[install-prefix]/share/physics_modules`, but if present in the local run directory, LAURA will read reactions from the local file instead.[13]

Reactants and products can be any species defined in the `species_thermo_data` file—see Section 5.15 on page 58. A sample entry looks like this,

```
O2 + M    <=>   2O + M                 1
2.000e+21  -1.50  5.936e+04            2
teff1 = 2                              3
exp1 = 0.7                             4
t_eff_min = 1000.                      5
t_eff_max = 50000.                     6
C = 5.0                                7
O = 5.0                                8
N = 5.0                                9
H = 5.0                                10
Si = 5.0                               11
e- = 0.                                12
```

The first line specifies the reaction while line 2 provides three coefficients of an Arrhenius-like equation,

$$K_f = c_f T_{eff}^{\eta} e^{-\epsilon_0/kT_{eff}} \tag{40}$$

where c_f is the preexponential factor, η is the power of temperature dependence on the preexponential factor, ϵ_0 is the Arrhenius activation energy, and k is the Boltzmann constant. The arrowheads in line 1 signify the allowed directionality of the reaction. The symbol => denotes forward reaction only while <=> denotes forward and backward rates are computed. The coefficients in line 2 correspond to c_f, η, and ϵ_0/k, respectively. For reactions with a generic collision partner, `M`, such as this one, these coefficients correspond to Argon; and other collision partners and their efficiencies (multipliers of c_f) are specified on lines following line 5 and 6, which give the valid temperature range for the reaction. The effective temperature, T_{eff}, is defined according to a given integer number in line 3; Default: 2

`teff1`,`teff2`

Flag defining formula to compute the effective temperature T_{eff} for the forward rate and backward rate, respectively. It is engaged for the case of thermal nonequilibrium. Options for `teff` are:

1: $T_{eff} = T_{tr}$

2: $T_{eff} = T_{tr}^{exp1} T_v^{1-exp1}$

[13]The precise installation location is given by LAURA during startup. It can also be found on Unix-like systems from the executable itself by issuing the command `strings laura | grep share`.

3: $T_{eff} = T_v$

where T_{tr} and T_v are translational and vibrational temperatures, respectively. Default: `1`

`exp1`

The exponent used to define the effective temperature when `teff1` = 2 (forward rate) or `teff2` = 2 (backward rate). See previous equations for `teff` options. Default: `0.7`

`rf_max`

The upper limit on the forward reaction rate in cgs units engaged only if `augment_kinetics_limiting` is `.true`. See output file `kinetic_diagnostics_output` for unlimited rates as a function of temperature. Default: `1.e+20`

`rf_min`

The lower limit on the forward reaction rate in cgs units engaged only if `augment_kinetics_limiting` is `.true`. See output file `kinetic_diagnostics_output` for unlimited rates as a function of temperature. Default: `1.e-30`

`rb_max`

The upper limit on the backward reaction rate in cgs units engaged only if `augment_kinetics_limiting` is `.true`. See output file `kinetic_diagnostics_output` for unlimited rates as a function of temperature. Default: `1.e+30`

`rb_min`

The lower limit on the backward reaction rate in cgs units engaged only if `augment_kinetics_limiting` is `.true`. See output file `kinetic_diagnostics_output` for unlimited rates as function of temperature. Default: `1.e-30`

`t_eff_min`

The minimum temperature for T_{eff} to compute reaction rates to circumvent stiff source terms. Default: `1000.`

`t_eff_max`

The maximum temperature for T_{eff} to compute reaction rates to circumvent stiff source terms. Default: `50000.`

5.11 laura.rst

This Fortran-unformatted binary file has the flowfield solution for every grid cell, boundary surface values, and grid cell derivatives for each block face. The number of variables in this file varies depending on the number of conservation equations that Laura needs to solve as specified through the `tdata` and laura_namelist_data files. A file of this format, but named laura_new.rst,[14] is generated by LAURA at the end of a successful run. The laura.rst file is required only if restarting from a prior run, i.e., irest = 1 in laura_namelist_data as described in Section 5.4.9 on page 26.

5.12 laura.trn

Turbulent transition location and transition length are specified in laura.trn.[15] Global values may be set directly through laura_namelist_data file. One line of data consisting of four integers followed by at least one space with the following entries per specified blocks is required: block_number, turbulent_flag, transition_location, and transition_length_factor where

turbulent_flag

One of -1, 0, or 1, which correspond to laminar, transition location in block, or fully turbulent flow, respectively.

transition_location

Specified as an x coordinate. This value used only if turbulent_flag is 0.

transition_length_factor

Defined as (transition length)/L, where L is the distance from the nose to the transition location. Use 0 for instantaneous transition.

5.13 laura_trajectory_data

This file is used to sequentially simulate points along a trajectory. Each line of the file defines a single trajectory point. The trajectory point data are entered in free format with time, velocity, density, temperature, alpha, and yaw in MKS units (6 entries per line). The simulation will start at trajectory point 1 by default unless another line number is specified in the laura_namelist_data using the namelist variable trajectory_data_point — see Section 5.4.16 on page 38.

The restart solution files and post processing files for each trajectory point are saved with a four digit number added to the usual root name of these files corresponding to the trajectory point number.

[14]When running a trajectory sequence, the file will be named laura_####.g where #### is the trajectory point index.

[15]LAURA transition files from versions prior to 5 can also be used.

5.14 `laura_vis_data`

This file contains the data set that overrides the default viscous terms in the i-, j-, and k-directions. There are four integers consisting of the block number and toggles for each viscous term per line in this file: `blk_num`, `ivis`, `jvis`, and `kvis` where `ivis`, `jvis`, and `kvis` are viscous terms in the i-, j-, and k-directions, respectively. The viscous-term flag options are:

`0`: Viscous terms are not engaged in the respective direction; i.e., inviscid flow.

`1`: Viscous terms and a reduced eigenvalue limiter are engaged in the respective direction. The reduced eigenvalue limiter is to prevent distortion of computed heating.

`2`: Viscous terms are engaged but an unmodified eigenvalue limiter is retained to maintain stability.

The following defaults will be applied if this file is not present in the working directory:

If `ivisc = 0` in `laura_namelist_data` (see Section 5.4 on page 14) then inviscid flow is specified and `ivis`, `jvis`, and `kvis` are set to zero for all blocks by default.

If `ivisc = 2` and `navier_stokes = .false.` in the `laura_namelist_data` file then the default is a function of the wall boundary condition. If the wall boundary on the i_{min} or i_{max} face of block n is a solid surface (see Section 5.3 on page 12) then `ivis(n) = 2`, otherwise `ivis(n) = 0`. In like manner, if the wall boundary on the j_{min} or j_{max} face of block n is a solid surface then `jvis(n) = 2`, otherwise `jvis(n) = 0`. The default specification for `kvis(n)` is set to 1 if k_{min} or k_{max} of block n is a solid surface, otherwise `kvis(n) = 0`. This default recognizes that the standard orientation is for the solid wall on the k_{min} surface and the reduced eigenvalue limiter is required in this case.

If `ivisc = 2` and `navier_stokes = .true.` in `laura_namelist_data` then the default is `ivis = 2`, `jvis = 2`, and `kvis = 1` for all the blocks.

There may be circumstances where the user wishes to override these defaults. If a block is away from the stagnation streamline crossing the shock into the boundary layer then a more accurate heating on the side walls (i and/or j) will be returned using `ivis` and/or `jvis` set to 1 without sacrificing stability. In the case of cavities, a block may sit over a cavity and not have any solid boundaries itself but has a well defined boundary layer streaming into it from an adjacent block. In this case, even though the block has no solid boundaries itself, it should engage viscous terms to capture the entering shear layer.

Example: To override the default values and to reset viscous terms in block number 3 to `ivis = 1`, `jvis = 0`, and `kvis = 1`, and in block number 5 to `ivis =0`, `jvis =1`, and `kvis =1` the `laura_vis_data` would look like:

```
3 1 0 1
5 0 1 1
```

5.15 species_thermo_data

The `species_thermo_data` file is the master file for species thermodynamic data.[16] Each species record consists of the species name, a species properties namelist - `&species_properties`, the number of thermodynamic property curve fit ranges, and the curve fit coefficients for each range [23]. No blank line is allowed in this file.

Elements of the `&species_properties` namelist are:

`molecule`

A logical flag to determine whether the species is a molecule (composed of more than one atom); if the species is a molecule then `molecule = .true.`, otherwise `molecule = .false.`

`ion`

A logical flag to identify the charged particle. `ion = .true.` for charged particles except for neutrals and electrons. This flag initializes electron-neutral energy exchange cross section and sum of the charges.

`charge`

An integer number to determine the number of positive charges in the particle. If `ion = .false.` then `charge = 0`.

`elec_impct_ion`

This real number to set the energy for neutrals (i.e. `ion=.false.`) in electron volts (eV) that is required to liberate an electron when the neutral impacts with a free electron.

`mol_wt`

A real number to set the molecular weight of the particle. This parameter is always required.

`siga`

An array of three real numbers, which correspond to curve fit coefficients for electron-neutron energy exchange cross section defined as

$$\sigma_{en} = a + bT + cT^2 \tag{41}$$

where σ_{en} is the electron-neutron energy exchange collision cross section in m^2, a, b, and c are the curve fit coefficients, and T is the gas temperature [24,25]. The format to define these coefficients is `siga=a, b, c`. For example, `siga=7.5e-20, 0, 0`.

[16]The `species_thermo_data` file should only be changed by developers.

disoc_ener

A real number to set dissociation energy of molecule in electron volts (eV).

alantel

A real number to set the Landau-Teller constant to compute vibrational relaxation time for molecule. These are defined in Ref. [26, 27]. This variable is required if `molecule=.true.`.

cprt0

A nondimensional real number that defines translational-rotational heat capacity that is normalized by gas constant. This is equal to

$$cprt() = \frac{f+2}{2} \tag{42}$$

where f is the number of degrees of freedom in translation and rotation. The defaults for atoms and diatomic molecules are 2.5 and 3.5, respectively.

Example: A portion of the `species_thermo_data` that provides thermodynamic properties of carbon is shown below.

```
C                                                                       1
&species_properties                                                     2
molecule = .false.                                                      3
ion = .false.                                                           4
charge = 0                                                              5
elec_impct_ion = 11.264                                                 6
siga = 7.5e-20, 5.5e-24, -1.e-28                                        7
mol_wt = 12.01070                                                       8
/                                                                       9
3                                                                       10
 0.64950315E+03 -0.96490109E+00  0.25046755E+01 -0.12814480E-04         11
 0.19801337E-07 -0.16061440E-10  0.53144834E-14  0.00000000E+00         12
 0.85457631E+05  0.47479243E+01   200.000  1000.000                     13
-0.12891365E+06  0.17195286E+03  0.26460444E+01 -0.33530690E-03         14
 0.17420927E-06 -0.29028178E-10  0.16421824E-14  0.00000000E+00         15
 0.84105978E+05  0.41300474E+01  1000.000  6000.000                     16
 0.44325280E+09 -0.28860184E+06  0.77371083E+02 -0.97152819E-02         17
 0.66495953E-06 -0.22300788E-10  0.28993887E-15  0.00000000E+00         18
 0.23552734E+07 -0.64051232E+03  6000.000 20000.000                     19
```

The species name is defined in line `1`. Between lines `2` and `9` species properties are defined. These parameters and/or flags state that the carbon molecular weight is 12.0107, and the species is neither a molecule nor a charged particle, but it can liberate an electron when its energy reaches 11.264 eV after it is impacted with a free electron.

Line `10` shows that there are three thermodynamic property curve fits for temperature ranges of 200 K $< T <$ 1,000 K, 1,000 K $< T <$ 6,000 K, and 6,000 K

$< T <$ 20,000 K. Each data range consists of 12 real numbers with a restriction of 4 real numbers per line. The first 10 real numbers are the thermodynamic curve fit coefficients, and the last two real numbers identify the temperature range for the given curve fit coefficients. These coefficients are used to calculate the following thermodynamic properties

$$c_p(T)/R = a_1T^{-2} + a_2T^{-1} + a_3 + a_4T + a_5T^2 + a_6T^3 + a_7T^4 \tag{43}$$

$$h(T)/RT = -a_1T^{-2} + a_2T^{-1}ln\ T + a_3 + a_4\frac{T}{2} + a_5\frac{T^2}{3} + a_6\frac{T^3}{4} + a_7\frac{T^4}{5} + \frac{a_9}{T} \tag{44}$$

$$s(T)/R = -a_1\frac{T^{-2}}{2} - a_2T^{-1} + a_3ln\ T + a_4T + a_5\frac{T^2}{2} + a_6\frac{T^3}{3} + a_7\frac{T^4}{4} + a_{10} \tag{45}$$

where T is the gas temperature, R is the universal gas constant, c_p, h and s are the species specific heat, enthalpy and entropy, respectively, and a_i are the provided curve fit coefficients in `species_thermo_data`.

The following corrections will be applied if the gas temperature is out of the given range for the given curve fit coefficients:

$$c_p(T) = c_p(T^*) \tag{46}$$

$$h(T) = h(T^*) + (T - T^*)c_p(T^*) \tag{47}$$

$$s(T) = s(T^*) + ln\ \frac{T}{T^*}c_p(T^*) \tag{48}$$

where

$$T^* = \begin{cases} T_{lower} & \text{for } T < T_{lower} \\ T_{upper} & \text{for } T > T_{upper} \end{cases} \tag{49}$$

5.16 `species_transp_data`

This file[17] contains log-linear curve fit coefficients for species collision cross section that are defined based on the temperature range of 2,000–4,000 K [25]. This temperature range spans boundary-layer temperatures for a typical hypersonic entry. The curve fit for the given coefficients is poor, however, at temperatures below 1,000 K. Higher-order curve fit data are available in `species_transp_data_0`, which supersedes that of `species_transp_data`—see Section 5.17 on the next page.

```
 Ar Ar
-14.6017 -14.6502 -14.5501 -14.6028 ! trr132+kestin et al
Ar+ Ar+
-11.48 -12.08 -11.50 -12.10
Ar N2
-14.5995 -14.6475 -14.5480 -14.5981 ! kestin et al
Ar CO
-14.5975 -14.6455 -14.5459 -14.5964 ! kestin et al
```

[17]The `species_transp_data` file should only be changed by developers.

5.17 species_transp_data_0

This file[18] provides collision cross section coefficients for higher-order curve fit data [28, 29] than those in the species_transp_data file—see Section 5.16. The data in species_transp_data_0 supersede the data in species_transp_data if the file is placed in the working directory hosting the simulation.

```
O2 N      1 1 1      (c)
         -1.1453028E-03  1.2654140E-02 -2.2435218E-01  7.7201588E+01
         -1.0608832E-03  1.1782595E-02 -2.1246301E-01  8.4561598E+01
          1.4943783E-04 -2.0389247E-03  1.8536165E-02  1.0476552E+00

NO N      1 1 1      (c)
         -1.5770918E-03  1.9578381E-02 -2.7873624E-01  9.9547944E+01
         -1.4719259E-03  1.8446968E-02 -2.6460411E-01  1.0911124E+02
          2.1014557E-04 -3.0420763E-03  2.5736958E-02  1.0359598E+00

NO O      1 1 1      (c)
         -1.0885815E-03  1.1883688E-02 -2.1844909E-01  7.5512560E+01
         -1.0066279E-03  1.1029264E-02 -2.0671266E-01  8.2644384E+01
          1.4145458E-04 -1.9249271E-03  1.7785767E-02  1.0482162E+00

END END   1 1 0
             0. 0. 0. 0.
             0. 0. 0. 0.
```

5.18 surface_property_data

This file is required if there are more than 1 solid surface boundary types (see Section 5.3 on page 12) and the surface conditions of these solid surfaces differ from those specified in laura_namelist_data (see Section 5.4 on page 14).

Surface boundary properties of each solid surface must be bounded by:

```
&surface_properties


/
```

The first instance of the parameters defines properties for surfaces of type -1 (note that properties of type 0 are defined in laura_namelist_data), the second instance defines properties for surfaces of type -2, and so on (see Section 5.3 on page 12 for different solid surface types.)

The parameters that can be defined for each solid surface boundary are:

blowing_model

See Section 5.4.1 on page 15 for more details and a complete list of options. Default: 'none'

[18]The species_transp_data_0 file should only be changed by developers.

catalysis_model

See Section 5.4.12 on page 31 for more details and a complete list of options. Default: `'super-catalytic'`

char_density

Density of the char, kg/m^3. Default: 256.29536, which is for the heritage AVCOAT.

CHONSi_frac_char

See Section 5.4.1 on page 15 for more details and a complete list of options. Default: `CHONSi_frac_char = 1.0, 0.0, 0.0, 0.0, 0.0`

CHONSi_frac_pyrolysis

See Section 5.4.1 on page 15 for more details and a complete list of options. Default: `CHONSi_frac_pyrolysis = 1.0, 0.0, 0.0, 0.0, 0.0`

emiss_a, emiss_b, emiss_c, emiss_d

Real numbers to assume emissivity constant coefficients for solid surface boundary type (see Section 5.3 on page 12) surface emissivity, ϵ, which is calculated as

$$\epsilon = \epsilon_a + \epsilon_b T_w + \epsilon_c T_w^2 + \epsilon_d T_w^3 \tag{50}$$

where T_w is the surface temperature. Values for ϵ_a, ϵ_b, ϵ_c, and ϵ_d are defined by `emiss_a`, `emiss_b`, `emiss_c`, and `emiss_d`, respectively. Default: `emiss_a=0.89`, `emiss_b=0`, `emiss_c=0`, and `emiss_d=0`

equil_surf_temp_floor

Minimum temperature of equilibrium catalytic surface for the purpose of determining equilibrium constants. Raising this value to 1000 may aide convergence if the transient solution includes low surface temperatures. If the converged surface temperature is less than this value than surface species mass fractions will be in error. Default: `100`.

h_ablation

See Section 5.4.1 on page 15 for more details and a complete list of options. Default: `0.0`

mdot_pressure

See Section 5.4.1 on page 15 for more details and a complete list of options. Default: `0.0`

surface_group_name

Character descriptor for surfaces with solid surface boundary types—see Section 5.3 on page 12. Any character can be specified to group solid surface boundaries. Default: `'undefined surface'`

surface_temperature

Initial wall temperature in K for the solid surface boundary. This variable is similar to `twall_bc` in Section 5.4.12 on page 31. The wall temperature stays constant as specified by this parameter if `surface_temperature_type = 'constant'`. Default: value of `twall_bc`.

`surface_temperature_type`

A character identifier for the surface temperature model. See Section 5.4.12 on page 31 for options and their descriptions. Default: `'constant'`

`t_ablation`

See Section 5.4.1 on page 15 for more details and a complete list of options. Default: `0.0`

`virgin_density`

Density of virgin material, kg/m^3. Default: 544.627742, which is for the heritage AVCOAT.

6 Output Files

Laura generates the files listed in Table 2 regardless of inputs specified. A description of each output file is presented after a brief discussion of `stdout` output.

Table 2: Laura output files.

Filename	Content	Format
`laura.aero`	Aerodynamic forces, moments, and coefficients	ASCII
`laura.g.fvbnd`	Boundary types in `laura.g`	FieldView™
`laura.nam`	Variables names in `laura.q`	Tecplot™
`laura.q`	Flowfield solution	Plot3D
`laura_blayer.dat`	Wall and Boundary layer edge data	Tecplot™
`laura_conv.out`	Time step, CPU time, and residuals	ASCII
`laura_new.g`	Volume grid	Plot3D
`laura_new.rst`	Flowfield solution for restart	Binary
`laura_surface.g`	Surface grid	Plot3D
`laura_surface.nam`	Variables names in `laura_surface.q`	Tecplot™
`laura_surface.q`	Surface solution	Plot3D

In addition to files, Laura writes some information on the screen. After initial configuration information such as grid block sizes, `laura_namelist_data` specifications, and `tdata` gas physics, the screen output is divided into five distinctive categories: block number, task number, L_∞ including location and equation, and L_2 of mixture continuity, xyz-momenta, and energy equations.

The L_2 norm is defined as

$$L_2 = \sum_{i=1}^{N} \left(\frac{|R_i|}{\rho_i} \right)^2 \tag{51}$$

where N is the total number of grid cells times the number of conservation equations, R is the residual, and ρ is the mixture density.

The location of the maximum L_∞ residual is shown by four integer numbers after the norm itself. The first three integer numbers correspond to i-, j-, and k-indices of the corresponding block, and the last number is the equation number, `n_eqn`. The total number of equations, `n_eqn_max`, is defined as

$$\texttt{n\_eqn\_max} = \texttt{n\_species} + 3 + \texttt{n\_energy} \tag{52}$$

where `n_species` and `n_energy` are the number of species and energy equations, respectively. The equation number corresponding to species is based on the species order defined in `tdata` followed by the x-, y-, and z-momentum equation numbers. For example, consider block 11 for the following sample screen output,

```
 Reading block sizes from laura.g
  block =   10 task =  10  Linf: 0.63E+02  23  24  86  1  L2eq: 0.31E+00 0.11E-01 0.12E-01 0.12E-01 0.32E-01 0.27E-02
> block =   11 task =  11  Linf: 0.12E+04  21   2  86 14  L2eq: 0.42E+01 0.10E+01 0.17E+00 0.95E+00 0.88E+00 0.28E-01 <
  block =   12 task =  12  Linf: 0.31E+03  11   7  87  5  L2eq: 0.20E+01 0.38E+00 0.22E+00 0.28E+00 0.35E+00 0.11E-01
  block =    6 task =   6  Linf: 0.68E+02   6  24  87  1  L2eq: 0.39E+00 0.17E-01 0.80E-02 0.28E-01 0.37E-01 0.32E-02
  block =    2 task =   2  Linf: 0.66E+02   3  24  87  1  L2eq: 0.49E+00 0.27E-01 0.88E-02 0.53E-01 0.53E-01 0.38E-02
  block =    4 task =   4  Linf: 0.17E+03  24   1  87 14  L2eq: 0.89E+00 0.12E+00 0.47E-01 0.14E+00 0.12E+00 0.75E-02
  block =    9 task =   9  Linf: 0.65E+02  21  24  86  1  L2eq: 0.43E+00 0.26E-01 0.36E-01 0.47E-01 0.48E-01 0.32E-02
  block =   14 task =  14  Linf: 0.78E+02   2  24  86  1  L2eq: 0.66E+00 0.63E-01 0.27E-01 0.22E-01 0.67E-01 0.36E-02
  block =    7 task =   7  Linf: 0.71E+03  14   2  87 14  L2eq: 0.27E+01 0.51E+00 0.11E+00 0.69E+00 0.55E+00 0.22E-01
  block =    1 task =   1  Linf: 0.22E+03  24   4  87 14  L2eq: 0.11E+01 0.13E+00 0.29E-01 0.14E+00 0.13E+00 0.90E-02
  block =    8 task =   8  Linf: 0.26E+03   6   1  87 14  L2eq: 0.10E+01 0.14E+00 0.97E-01 0.19E+00 0.16E+00 0.82E-02
  block =    5 task =   5  Linf: 0.78E+02   1  24  87  1  L2eq: 0.47E+00 0.22E-01 0.23E-01 0.52E-01 0.50E-01 0.35E-02
  block =    3 task =   3  Linf: 0.45E+03  24  16  87 14  L2eq: 0.21E+01 0.35E+00 0.80E-01 0.41E+00 0.34E+00 0.18E-01
  block =   13 task =  13  Linf: 0.15E+03  17   2  87  5  L2eq: 0.98E+00 0.73E-01 0.63E-01 0.65E-01 0.88E-01 0.39E-02
  step  =   10  time =    57.20 sum(abs(task error)) =  0.28E+02 L2 norm =   0.17E-02
```

which is for the 14-block, 11-species, two-temperature case. The maximum change is shown to be from z-momentum equation `n_eqn`=14 located on i=21, j=2, and k=86 of block 11.

See Section 7.15 on page 74 for a description of the `laura_stdout_to_tec` utility, which can convert this output into a Tecplot™-compatible format.

6.1 `laura.aero`

Aerodynamic forces, moments, and coefficients are written in this ASCII file. The `laura.aero` file is a plain text file, while the `laura_aero.dat` file is formatted to be read by Tecplot™. A sample of each are provided below:

`laura.aero` file:

```
Grid Conversion Factor                  :   1.000000000E+00
Reference Area      (grid unit squared) :   1.824150000E-02
Reference Length    (grid unit)         :   1.524000000E-01
x moment center     (grid unit)         :   0.000000000E+00
y moment center     (grid unit)         :   0.000000000E+00
z moment center     (grid unit)         :   0.000000000E+00

       :Environment:
Gas Model                :    1 species
Temperature    K         :   6.190000000E+01
Dyn. Pressure  N/m2      :   1.435646897E+04
Density        kg/m3     :   3.253000000E-02
Velocity       m/s       :   9.395000000E+02
Alpha          deg       :   2.800000000E+01
Mach                     :   5.939772642E+00
Reynolds  1/(grid unit) :   7.412138011E+06

       :Aerodynamic Forces:
 Force along the x-axis :   6.003123276E-01
 Force along the y-axis :  -1.127221398E-01
 Force along the z-axis :   4.232414705E-02

       :Aerodynamic Moments:
 Moment about the x-axis:   2.112152667E-05
 Moment about the y-axis:  -3.974829879E-02
 Moment about the z-axis:  -1.907680328E-02

       :Aerodynamic Coefficients:
 Cd  (Drag)              :   5.499143081E-01
 Cl  (Lift)              :  -2.444595628E-01
 L/D (Lift/Drag)         :  -4.445411935E-01
```

`laura_aero.dat` file:

```
TITLE = "AERO COEFFICIENTS"
VARIABLES =
"time"
"length_reference"
"sref"
"cref"
"xmc"
"ymc"
"zmc"
"temperature"
"dynamic_pressure"
"density"
"velocity"
"alpha"
"mach"
"cx"
"cy"
"cz"
"roll"
"pitch"
"yaw"
"cd"
"cl"
"lod"
DATASETAUXDATA ProjectTitle="Spherical-cap"
DATASETAUXDATA CaseTitle="Re=2E6/ft"
DATASETAUXDATA AngleOfAttack=" 28.000"
DATASETAUXDATA AngleOfYaw="  0.000"
DATASETAUXDATA FreestreamMachNumber="  5.940"
DATASETAUXDATA FreestreamUnitReynoldsNumber="  0.7412E+07"
DATASETAUXDATA FreestreamPressure="  0.2025E-01"
DATASETAUXDATA FreestreamDensity="  0.3253E-01"
DATASETAUXDATA FreestreamVelocity="  0.9395E+03"
DATASETAUXDATA FreestreamTemperature="    61.900"
DATASETAUXDATA FreestreamVibrationalTemperature="    61.900"
DATASETAUXDATA FreestreamViscosity="  0.4123E-05"
DATASETAUXDATA TrajectoryTime="     0.000"
DATASETAUXDATA GridConversionFactor="     1.000"
DATASETAUXDATA ReferenceArea="     0.018"
DATASETAUXDATA ReferenceLength="     0.152"
DATASETAUXDATA XMomentCenter="     0.000"
DATASETAUXDATA YMomentCenter="     0.000"
DATASETAUXDATA ZMomentCenter="     0.000"
ZONE T="laura_aero: Block 1"
I = 1, J = 1, K = 1, ZONETYPE = ORDERED
AUXDATA ProjectTitle="Spherical-cap"
AUXDATA CaseTitle="Re=2E6/ft"
AUXDATA AngleOfAttack=" 28.000"
AUXDATA AngleOfYaw="  0.000"
AUXDATA FreestreamMachNumber="  5.940"
AUXDATA FreestreamUnitReynoldsNumber="  0.7412E+07"
AUXDATA FreestreamPressure="  0.2025E-01"
AUXDATA FreestreamDensity="  0.3253E-01"
AUXDATA FreestreamVelocity="  0.9395E+03"
AUXDATA FreestreamTemperature="    61.900"
AUXDATA FreestreamVibrationalTemperature="    61.900"
AUXDATA FreestreamViscosity="  0.4123E-05"
AUXDATA TrajectoryTime="     0.000"
AUXDATA GridConversionFactor="     1.000"
AUXDATA ReferenceArea="     0.018"
AUXDATA ReferenceLength="     0.152"
AUXDATA XMomentCenter="     0.000"
AUXDATA YMomentCenter="     0.000"
AUXDATA ZMomentCenter="     0.000"
 0.000000000E+00  1.000000000E+00  1.824150000E-02  1.524000000E-01  0.000000000E+00  0.000000000E+00  0.000000000E+00
```

6.2 `laura.g.fvbnd`

FieldView™ boundary data file used to label boundary condition types contained in `laura.g` file. Tecplot™ automatically detects this file when the `laura.g` is loaded.

6.3 `laura.nam`

Tecplot™ name file used to label variables contained in `laura.q` file.

6.4 `laura.q`

PLOT3D function file for post-processing volume solution. Most of the variables in this file are nondimensionalized according to Table 3. The actual number of variables in this file depends on the condition specified in the input files.

Table 3: laura.q variables

Variables	**Definition**	**Normalized by**
$c_{N,N2,...}$	Species mass fraction	-
u, v, w	Velocity components	U_∞
E	Total energy per unit mass	U_∞^2
e_j	Energy mode j per unit mass	U_∞^2
T	Translational temperature	-
T_v	Vibrational temperature	-
ρ	Density	ρ_∞
M_w	Molecular weight	-
p	Pressure	$\rho_\infty U_\infty^2$
c	Sonic velocity	U_∞
e	Static energy per unit mass	U_∞^2
e_v	Vibrational energy per unit mass	U_∞^2

6.5 `laura_blayer.dat`

Boundary layer edge and wall surface properties are written in this ASCII and Tecplot™-readable file. Properties will also be written at a user-specified `roughness_height`=`value` if `roughness`=`.true`. is specified.

Below is an example showing the header of the file and the auxiliary data as generated using the default `blayer_flagset` = `standard` specification. Note that the species mass fraction will be written in the exact same order as provided by the user in `tdata` file.

In addition to these wall and boundary-layer properties, reference information is also included as Tecplot™ auxiliary data both for the overall dataset (`AUXDATASET` variables) and for each zone (`AUXDATA` variables).

```
TITLE = "BL EDGE PROPERTIES"
VARIABLES =
"xw (m)"
"yw (m)"
"zw (m)"
"rhow (kg/m^3)"
"pw (Pa)"
"Tw (K)"
"Tvw (K)"
"Hw (J/kg)"
"muw (Pa.s)"
"c<sub>N2</sub>w"
"c<sub>O2</sub>w"
"c<sub>NO</sub>w"
"c<sub>N</sub>w"
"c<sub>O</sub>w"
"c<sub>N+</sub>w"
"c<sub>O+</sub>w"
"c<sub>NO+</sub>w"
"c<sub>N2+</sub>w"
"c<sub>O2+</sub>w"
"c<sub>e-</sub>w"
"qw_conv (W/m^2)"
"qw_rad (W/m^2)"
"qw_total (W/m^2)"
"tauwx (Pa)"
"tauwy (Pa)"
"tauwz (Pa)"
"tauw_total (Pa)"
"re-cell"
"kappaw (W/m.K)"
"emissw"
"Utau (m/s)"
"Cp"
"Utau*rhow/muw (1/m)"
"rhoe (kg/m^3)"
"pe (Pa)"
"Te (K)"
"Tve (K)"
"He (J/kg)"
"ue (m/s)"
"ve (m/s)"
"we (m/s)"
"Ue_tot (m/s)"
"Me"
"mue (Pa.s)"
"c<sub>N2</sub>e"
"c<sub>O2</sub>e"
```

```
"c<sub>NO</sub>e"
"c<sub>N</sub>e"
"c<sub>O</sub>e"
"c<sub>N+</sub>e"
"c<sub>O+</sub>e"
"c<sub>NO+</sub>e"
"c<sub>N2+</sub>e"
"c<sub>O2+</sub>e"
"c<sub>e-</sub>e"
"delta (m)"
"deltastar (m)"
"theta (m)"
"CH_conv (kg/m^2.s)"
"Re_theta"
"Re_theta/Me"
DATASETAUXDATA ProjectTitle="LOFTID"
DATASETAUXDATA CaseTitle="t=515, A=0, FC, CS"
DATASETAUXDATA AngleOfAttack="  2.000"
DATASETAUXDATA AngleOfYaw="  0.000"
DATASETAUXDATA FreestreamMachNumber=" 26.156"
DATASETAUXDATA FreestreamUnitReynoldsNumber="  0.3947E+04"
DATASETAUXDATA FreestreamPressure="  0.1044E-02"
DATASETAUXDATA FreestreamDensity="  0.7652E-05"
DATASETAUXDATA FreestreamVelocity="  0.7551E+04"
DATASETAUXDATA FreestreamTemperature="   206.400"
DATASETAUXDATA FreestreamVibrationalTemperature="   206.400"
DATASETAUXDATA FreestreamViscosity="  0.1464E-04"
DATASETAUXDATA TrajectoryTime="   515.000"
DATASETAUXDATA GridConversionFactor="     1.000"
DATASETAUXDATA ReferenceArea="    28.199"
DATASETAUXDATA ReferenceLength="     5.992"
DATASETAUXDATA XMomentCenter="     0.000"
DATASETAUXDATA YMomentCenter="     0.000"
DATASETAUXDATA ZMomentCenter="     0.000"
ZONE T="laura_blayer: Block  1"
I=   17 , J=   17 , K=1 , ZONETYPE = Ordered
DATAPACKING=BLOCK
AUXDATA ProjectTitle="LOFTID"
AUXDATA CaseTitle="t=515, A=0, FC, CS"
AUXDATA AngleOfAttack="  2.000"
AUXDATA AngleOfYaw="  0.000"
AUXDATA FreestreamMachNumber=" 26.156"
AUXDATA FreestreamUnitReynoldsNumber="  0.3947E+04"
AUXDATA FreestreamPressure="  0.1044E-02"
AUXDATA FreestreamDensity="  0.7652E-05"
AUXDATA FreestreamVelocity="  0.7551E+04"
AUXDATA FreestreamTemperature="   206.400"
AUXDATA FreestreamVibrationalTemperature="   206.400"
AUXDATA FreestreamViscosity="  0.1464E-04"
AUXDATA TrajectoryTime="   515.000"
AUXDATA GridConversionFactor="     1.000"
AUXDATA ReferenceArea="    28.199"
AUXDATA ReferenceLength="     5.992"
AUXDATA XMomentCenter="     0.000"
AUXDATA YMomentCenter="     0.000"
AUXDATA ZMomentCenter="     0.000"
[...]
```

6.6 `laura_conv.out`

Time steps and residuals history are written in this file. Some of the flow conditions, such as angle-of-attack, freestream conditions, Mach number, Reynolds-number-per-grid-unit, etc. are also repeated at the beginning of the file. As shown in the following sample, step number, clock time, sum of all the residuals in all active tasks, and the overall L_2 norm of the residuals are written in this file. The overall L_2 norm is defined as:[19]

$$L_2 = \frac{\sum_i (|rhs_i|/\rho_i)^2}{CFL^2} \tag{53}$$

where rhs is the residual and ρ is the density.

```
Step      0 time=     4.53
step  =   10  time =    10.87 sum(abs(task error)) =  0.27E-03 L2 norm =   0.92E-12
step  =   20  time =    17.13 sum(abs(task error)) =  0.26E-03 L2 norm =   0.88E-12
step  =   30  time =    23.40 sum(abs(task error)) =  0.25E-03 L2 norm =   0.84E-12
step  =   40  time =    29.65 sum(abs(task error)) =  0.25E-03 L2 norm =   0.81E-12
   .
   .
```

See Section 7.11 on page 73 for a description of the `laura_conv_to_tec` utility, which can convert this file into a Tecplot™-compatible format.

6.7 laura_new.g

The Plot3D grid file includes any changes associated with grid adaptation or grid doubling during the run. The name must be changed to `laura.g` if the user wants to restart with this new grid on the next run—see Section 5.2 on page 11 for more information on the file format.

6.8 laura_new.rst

The restart file contains volume and surface data required for restart from the end of the current run. The name must be changed to `laura.rst` if the user wants to restart from this new solution file.

6.9 laura_surface.g

The surface grid file in Plot3D "3-D whole" multiblock format. See Section 5.2 on page 11 for more information on the file format.

6.10 laura_surface.nam

Tecplot™ name file used to label variables contained in `laura_surface.q` file.

6.11 laura_surface.q

Plot3D function file for post-processing surface solution. The number of variables in this file depends on the conditions specified in the `laura_namelist_data`. Some of the surface variables are presented in Table 4.

[19]The overall L_2 norm definition is different than the L_2 norm defined for each equation by Equation 51.

Table 4: laura_surface.q variables.

Variables	**Definition**	**Unit**
T	Surface temperature	K
p	Surface pressure	N/m^2
τ_x, τ_y, τ_z	Wall shear stresses	N/m^2
q_{conv}	Convective heat flux	W/cm^2
q_{rad}	Radiative heat flux	W/cm^2
ϵ	Surface emissivity	-
$mdot$	Blowing or Suction rate	kg/m^2-s

7 Laura Utilities

Laura has several interactive utilities that automatically generate some of the required input files or otherwise aid running and post-processing Laura simulations. These utilities are explained here.

7.1 avrg_surf_files

This interactive utility, which is applicable only to time accurate simulations, generates a Plot3D file containing the mean of surface values for the user specified period of the simulation. It can also generate a histogram of surface values at any specified point in a Tecplot™ readable format. This utility requires presence of `laura_surface_dtXXXX.q` files generated at run time with the engagement of the command `isurf_freq` in the namlist (see Section 5.4).

7.2 blayer_average.py

This python script averages a series of `laura_blayer_XX.dat` files, writing the averaged values to `laura_blayer.dat`.

7.3 blayer_stats.py

This python script computes several statistical properties of a series of files named `laura_blayer_snapshot_XX.dat`, writing the averaged values to files with names such as `laura_blayer_avg.dat` and `laura_blayer_stdev.dat`.

7.4 bounds

This interactive utility creates `laura_bound_data` that contains block connectivity data. This utility reads the volume grid data from a Plot3D structured grid, `laura.g`. See Section 5.2 on page 11 for more detail on the file format. Here is a sample of an interactive session with `bounds`:

```
 Enter precision of laura.g : 1 = single, 2 = double
2
 Do you want all type 9 bounds to default to type 8?
 Enter 1 for yes or 0 for no: (0)
0
 .
 .
 BLOCK  =    1 k  =  1 BOUNDARY

 Area  =   0.26821087E+03
 (Xcntr,Ycntr,Zcntr) = (-0.63020306E+02, 0.91387367E+02,-0.63128987E+03)

 Enter ITYPE: (0)
0
 .
 .
```

The first question is about the precision of `laura.g`. Note that any grid file created by LAURA or one of its utilities is created in double-precision.

The second question is about type 8 and type 9 boundary data. These boundary types are given for the block faces that are shared between two blocks. A type 9 requires an identical orientation of indices across the shared boundary (increasing-i matched to increasing-i, increasing-j matched to increasing-j, and increasing-k matched to increasing-k). A type 8 accommodates more general connectivity.

7.5 coarsen

Use this utility to coarsen the grid, `laura.g`, and solution, `laura.rst`, files in i-, j-, and/or k-directions. The new files, `laura_new.g`, and `laura_new.rst`, will be double precision regardless of the input grid precision. This utility does not accept single precision `laura.rst`.

7.6 convert_bound_data

The utility converts old (pre LAURA.5) `STRTfiles/bound_data.strt` files to the new `laura_bound_data` format. Usage: `convert_bound_data bound_data.strt`.

7.7 convert_laura

This interactive utility converts cases run with prior versions of LAURA.5—see Appendix A on page 88. This utility generates `laura.g`, from `old.rst`, and a new restart, `laura_new.rst`.

The following files are either required or optional prior to the execution of this utility:

`old.ep+`

This file, which may have a different root name, has algebraic turbulent model information. This file is optional.

`old.qtw`

This file, which may have a different root name, has surface temperature information. This file is optional. If this file is provided, free stream density and velocity are also needed. LAURA uses this information to approximately calculate the related parameters needed to be in `laura_new.rst`.

`old.rst`

This file, which may have a different root name, is the old restart file that contains volume and surface data. The utility will ask for the precision of the data and the numbers of grid-blocks that are in this file. This file is required.

`laura_bound_data`

Use the `convert_bound_data` utility, or see Appendix A on page 88 to convert this file from old format. This file is required.

7.8 `laura_aero_components.py`

This executable python script calculates and plots aerodynamic components from a `laura_blayer.dat` file, broken down by inviscid, viscous, and total and by surface components (surface blocks in the grid file). This utility is good for calculating forebody and aftbody contributions or aerodynamics from control surfaces, for instance.
Usage: `laura_aero_components.py -help`.

7.9 `laura_aero_history.py`

This executable python script makes plots of aerodynamics versus iteration from the screen output, if concatenated into a file. The `laura_blayer.dat` file is also required.
Usage: `laura_aero_history.py -help`.

7.10 `laura_compare.py`

This executable python script compares two `laura_blayer.dat` files. Useful to gauge convergence or to compare laminar versus turbulent cases, for instance.
Usage: `laura_compare.py -help`.

7.11 `laura_conv_to_tec`

This utility translates the LAURA convergence history file, `laura_conv.out`, described in Section 6.6 on page 69 into a form usable by Tecplot™.
Usage: `laura_conv_to_tec [options]`.

7.12 `laura_indicators.py`

This executable python script curve fits results from a series of `laura_blayer.dat` files. User inputs body point locations for the indicators.
Usage: `laura_indicators.py -help`.

7.13 `laura_multiRun.sh`

This utility runs LAURA multiple times and generates a `laura_blayer.dat` that is the average of the same files from each run. For example, to generate an average `laura_blayer.dat` file from 10 successive runs of `mpiexec -np 2 laura > laura.out`:
Usage: `laura_multRun.sh 10 "mpiexec -np 2 laura > laura.out"`

7.14 `laura_residuals.py`

This executable python script makes plots of residuals and CPU time versus iteration number. Requires `laura_conv.out` or a file of the standard output from a LAURA run.
Usage: `laura_residuals.py -help`.

7.15 `laura_stdout_to_tec`

To allow task-specific convergence history plotting, this utility translates the LAURA standard output stream described in Section 6 on page 64 into a convergence history file for each task suitable for Tecplot™.
Usage: `laura_stdout_to_tec [options] [file]`.

7.16 `make_assign_tasks`

Given the number and blocks and processes, the `make_assign_tasks` utility will generate a default `assign_tasks` (point relaxation with k-sweeps).
Usage: `make_assign_tasks n_blocks n_processes`.

7.17 `mirror_rst_grid`

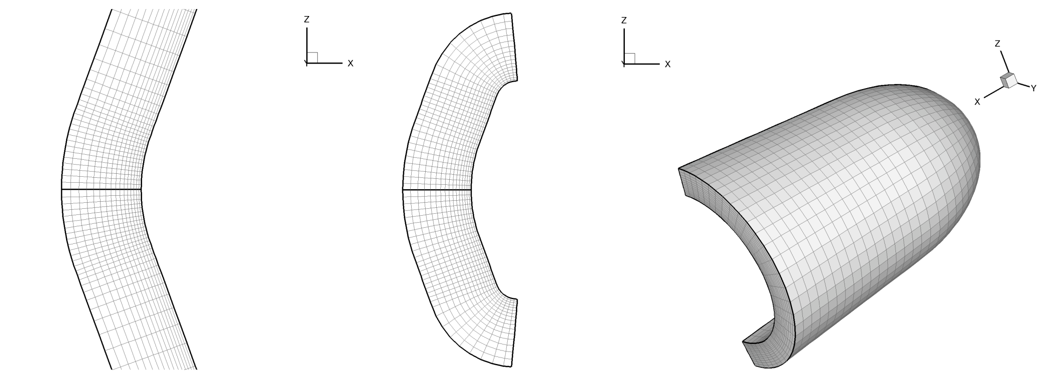
Use this interactive utility to mirror grid and or restart files. This utility reads `laura.g` and `laura.rst` for the original grid and generates `laura_new.g` and `laura_new.rst` files based on the user specified mirroring axis.

7.18 `prolongate`

Use this utility to generate fine adapted grid, `laura_new.g`, and solution, `laura_new.rst`, files from coarse adapted grid, `coarse.g`, and solution, `coarse.rst`, files. You will need to provide a fine unadapted grid filename and coarsening stride used in the `coarse.g` file.

7.19 `self_start`

This interactive utility generates a single-block structured grid, `laura.g`, for families of 2D, axisymmetric and 3D blunt bodies. This utility will also generate `laura_bound_data`. Schematics of several blunt body families are shown in Figure 3. Note that for axisymmetric geometries, the symmetry boundaries must be on the y-axis. Parameters for defining a probe shape are shown in Figure 4 on the following page.



(a) Spherically blunted cone. (b) Custom aerobrake. (c) Asymmetric 3D cone.

Figure 3: Sample geometries generated by `self_start` utility.

7.20 `shuffle_laura`

This interactive utility modifies (shuffles) variables in the LAURA restart, `laura.rst`, to enable continuation of the simulation if the gas model variables or parameters including number of species, thermal nonequilibrium, radiation, ablation, and turbulence model are modified. The users will be prompted to provide necessary information.

θ	body half-angle
δ	rake angle
τ	turning angle
ε	nose eccentricity, b/a
R_{Sh}	shoulder radius
R_N	nose radius
R_B	base plane radius

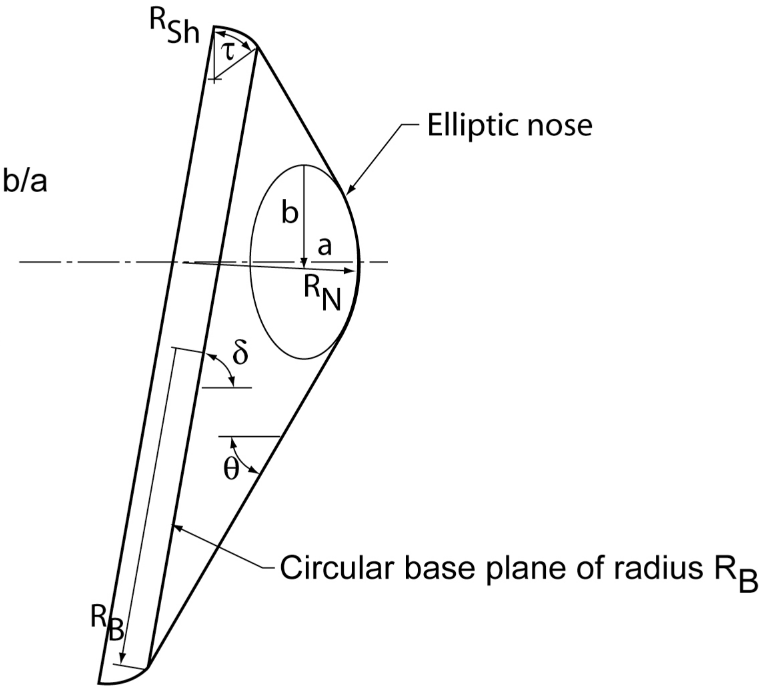


Figure 4: Definition of probe shape parameters used by `self_start`.

8 Sample Cases

8.1 Sphere: 5-species Air, Thermochemical Nonequilibrium

A working directory is created in which all requisite files are assembled. The utility `self_start` is used to generate a grid and the initial template for some required input files within this working directory. A verbatim transcript of an interactive session using `self_start` follows.

```
% self_start
 Select dimensionality:
 1 = Axisymmetric
 2 = Two-dimensional
 3 = Three-dimensional
1
 Select geometry:
 1 = Conic (cone/wedge, paraboloid, etc.).
 2 = Aerobrake (includes AFE without axis singularity).
2
 Select aerobrake type:
 0 = AFE
 1 = hemisphere
 2 = customized aerobrake
1
Enter radius (m) {     1.000000}:
1.
 Select number of cells along symmetry plane.
30
 Enter number of cells in k direction, prior to any doubling
16
```

At this point the grid file `laura.g`, the boundary data file `laura_bound_data`, and the namelist file `laura_namelist_data` are created. The boundary data file does not require any further modification. The task assignment file is set for point-implicit relaxation - the standard practice for starting any simulation. The namelist file requires editing to define free stream conditions and possibly alter default settings. The edited file used for the first run of the test case follows.

```
&laura_namelist
velocity_ref = 5000. ! reference velocity, m/s
density_ref  = 0.001 ! reference density, kg/m^3
tref         = 200.0 ! reference temperature, K
alpha        = 0.000 ! pitch angle, degrees
twall_bc     = 500.0 ! initial wall temperature, K
chem_flag  = 1  ! 0 chemically frozen, 1 chemical source on
therm_flag = 1  ! 0 thermally frozen,  1 thermal source on
```

```
irest    = 0    ! 0 for fresh start, 1 for restart
ncyc     = 2000 ! global steps
jupdate  = 4    ! steps between update of Jacobian
ntran    = 4    ! steps between update of transport properties
nitfo    = 1500 ! number of 1st-order relaxation steps
iterwrt  = 200  ! steps between saves of intermediate solution
rf_inv   = 3.0  ! inviscid relaxation parameter
rf_vis   = 1.0  ! viscous relaxation parameter
movegrd  = 100  ! number of steps between calls to align_shock
maxmoves = 0    ! maximum number of calls to align_shock
re_cell  = 0.1  ! target cell Reynolds number at wall
fsh      = 0.6  ! target bow shock position arc length fraction
kmax_error = 0.005 ! error norm for doubling grid in k-dir.
kmax_final = 64 ! max number cells in k dir after all doubling
nexch    = 2    ! steps between exchange of info in mpi
frac_line_implicit = 0.7 ! fraction of line by block tri-dia
surface_temperature_type_0 = `radiative equilibrium'
catalysis_model_0 = `super-catalytic'
emiss_a_0 = 0.89
ept = 0.010     ! relaxation factor on read eq wall bc
dimensionality = 'axisymmetric'
xmc =   0.0000
ymc =   0.0000
zmc =   0.0000
grid_conversion_factor =   1.0000
sref =  0.43633E-01
cref =   2.0000
/
```

The first 5 variables of the namelist in this particular template (other variable orderings are acceptable) deal with free stream conditions. The user must set these values, otherwise the user will get the default values assuming laminar flow of a perfect gas at 5 km/s and 0.001 kg/m^3. In this case, both the `chem_flag` and `therm_flag` are reset to 1 to turn on the chemical and thermal source terms. Other template values are defined to provide a reasonable compromise between solution robustness and convergence rate for a fresh start solution (irest = 0). The template calls for 2,000 relaxation steps (`ncyc`) in the initial run with Jacobian updates (`jupdate`) and transport property updates (`ntran`) requested every four relaxation steps. The first 1,500 iterations are executed using first-order spatial accuracy (`nitfo`)—second-order accuracy does not contribute significantly to the solution evolution in the initial relaxation period. The inviscid and viscous relaxation factors (`rf_inv` = 3 and `rf_vis` = 1) multiply the respective contributions to the Jacobian matrices and provide damping of the update. Larger values sometimes improve robustness for more energetic flows but are not required in this case.

Template values for grid movement are `movegrd` = 100, `maxmoves` = 0 (unlimited number of moves), `re_cell` = 0.1, and `fsh` = 0.6. These values are approximately tuned for optimal response in the opening relaxation process from a fresh start where the body materializes in a supersonic flow. Allowing the grid to move every 100 steps provides frequent opportunity to follow the evolution of the shock front as it initially is collapsed on the surface and then reflects off the surface into the oncoming flow. Setting `re_cell` to 0.1 provides very tight stretching near the wall. If there is a large difference between the wall temperature and the temperature of the first cell center off the wall then the upwind algorithm may fail to sense the wall - the boundary condition using the Roe's averaged interface may admit a supersonic flow directed toward the wall at the interface. This condition subverts the ability of the inviscid, no-slip boundary to properly engage. The tighter near wall resolution enables the upwind scheme to sense the wall under all wall temperature conditions tested to date. Setting `fsh` = 0.6 targets the captured bow shock location at 60% of the distance between the wall and the inflow boundary, providing adequate margin for the shock to reflect outward without striking the inflow boundary prior to the next grid update.

Automated grid doubling in the k direction is controlled by the triggering error norm magnitude (`kmax_error` = 0.005) and the maximum number of cells in the k direction (`kmax_final` = 64). Recall that the grid was initialized with only 16 cells in the k direction with the `self_start` utility. The grid will double to 32 cells in the k direction (normal to the wall) when the L2 norm first drops below 0.005. It will double again when the error norm next falls below this trigger point. The selection of a trigger point for more complex problems (three-dimensional, highly energetic) may require user experimentation: trigger too high and the grid doubles too early causing a lot of extra work; trigger too low and the solution may ring (limit cycle) on a coarse grid and never engage a finer grid.

The template setting therefore updates the boundary condition after completion of a forward and backward sweep through the domain. Other template values are not discussed here. They are consistent with default values as described in Section 5.4 and are included for the users convenience.

The `tdata` file is generally the only file that requires editing by the user. For the case of 5 species air in thermochemical nonequilibrium, the file `tdata` is defined as follows.

```
Two Temperature
N  6.217e-20
O  7.758e-09
N2  0.737795
O2  0.262205
NO  1.e-09
```

The fresh start run for this example case is now ready to be executed. It is assumed that the executable file laura is available in the working directory. This case is small enough to be run in interactive mode. For the purposes of discussion, it is convenient to capture the output in a file called `out_01`.

```
% laura >& out_01       (for csh)
% laura > out_01 2>&1   (for sh)
```

The user should note several types of information in `out_01`. The beginning of this file contains diagnostic information regarding the presence of optional files, free stream conditions, and kinetic model diagnostics including warnings regarding the absence of some allowed third body collision partners. Lines beginning with "step =" keep track of the relaxation step number, elapsed wall time, sum of the L1 norms overall conservation equations, and the L2 norm. Lines beginning with "block =" keep track of the L1 norm for mixture continuity, x-momentum, y-momentum, z-momentum, total energy, and vibrational-electronic energy residuals. The statement `Calling align_shock`... appears after every 100 steps (`movegrd` = 100). The statement `Saving restart and plot files.` followed by intermediate values of aerodynamic coefficients appear after every 200 steps (`iterwrt` = 200). The grid doubles automatically after step 472 from 16 to 32 cells in the k direction when the error norm first drops below 0.005.

```
block =    1  task =   1 err:  0.17E+01  0.30E+00  0.61E-15  0.44E+00  0.38E+00  0.53E-01
step  =  472  time =    24.65 sum(abs(task error)) =  0.29E+01 L2 norm =   0.50E-02

Increased kmax to  32
block =    1  task =   1 err:  0.69E+05  0.54E+02  0.31E-14  0.51E+02  0.14E+04  0.40E+03
step  =  476  time =    25.09 sum(abs(task error)) =  0.70E+05 L2 norm =   0.97E+08
```

In general, there is a large jump in the error norm following a grid move or grid doubling, which rapidly diminishes to preadjustment levels. A second doubling from 32 to 64 cells occurs after step 1348.

```
block =    1  task =   1 err:  0.15E+01  0.47E+00  0.13E-14  0.34E+00  0.45E+00  0.47E-01
step  = 1348  time =   113.30 sum(abs(task error)) =  0.28E+01 L2 norm =   0.49E-02

Increased kmax to  64
block =    1  task =   1 err:  0.47E+05  0.46E+02  0.39E-14  0.41E+02  0.82E+03  0.23E+03
step  = 1352  time =   114.19 sum(abs(task error)) =  0.49E+05 L2 norm =   0.46E+08
```

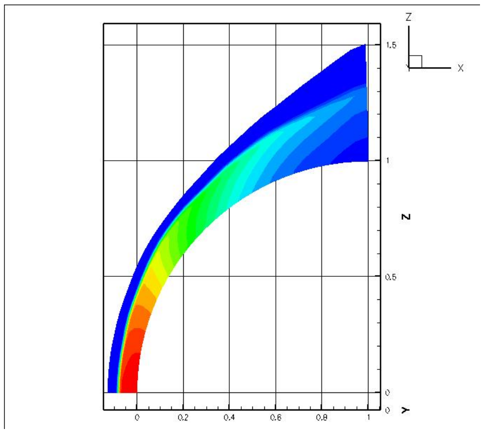
The interim solution for pressure contours after the first 2,000 steps is shown in Figure 5a on the next page. The corresponding surface pressure and heating distributions are shown in Figure 5b. The deep blue zone in front of the sphere represents undisturbed free stream conditions. The deep red indicated the high pressure stagnation region. The captured shock is approximately located at 60% of the distance between the spherical surface and the inflow boundary. The surface pressures and shock shape will be shown to be nearly converged at this point but the heating rates are still far from converged. Converging the boundary layer profile is the focus of the remaining relaxation steps.

Line relaxation is engaged at this point by `point_implicit = .false.` in the namelist file. The default is sweeping around the sphere in the i direction and applying line relaxation across the boundary layer in the k direction.

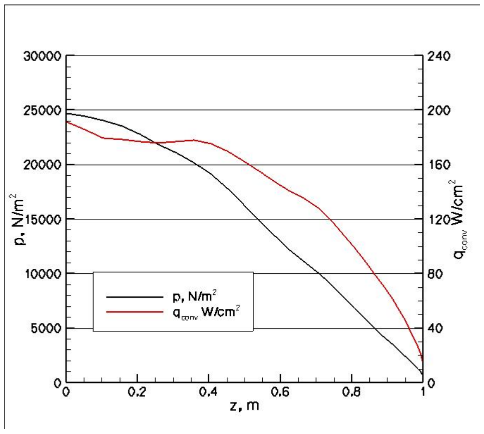
```
sweep_direction = 1 ! i-direction
relax_direction = 3 ! k-direction
```

The adapted grid and restart files are renamed to start the second run.

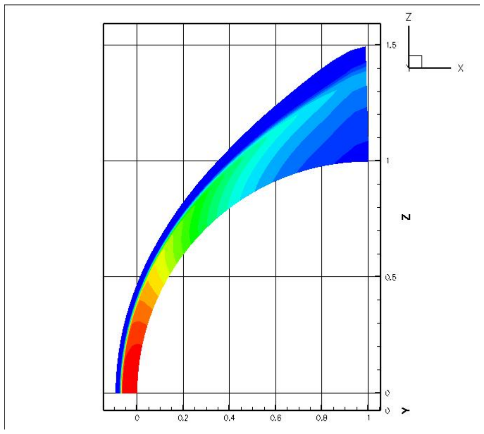
```
% cp laura_new.g laura.g
% cp laura_new.rst laura.rst
```



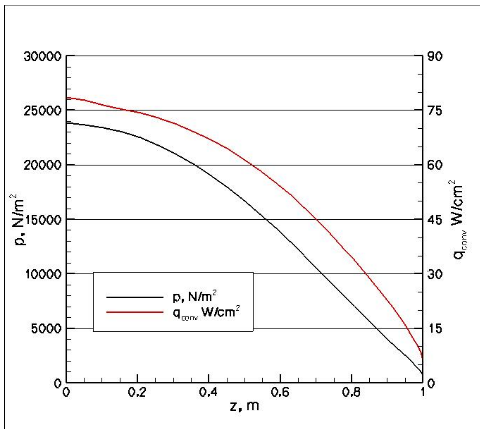
(a) Pressure contours - interim solution after 2000 steps



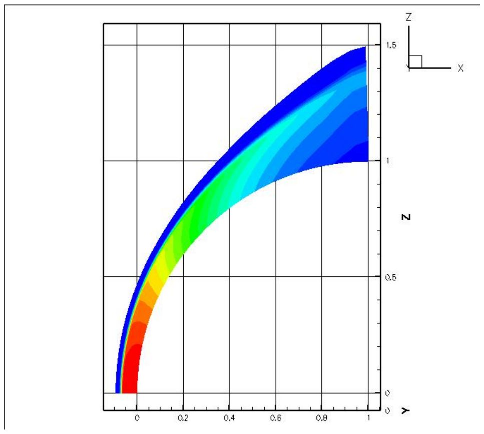
(b) Surface pressure and heating - interim solution after 2000 steps



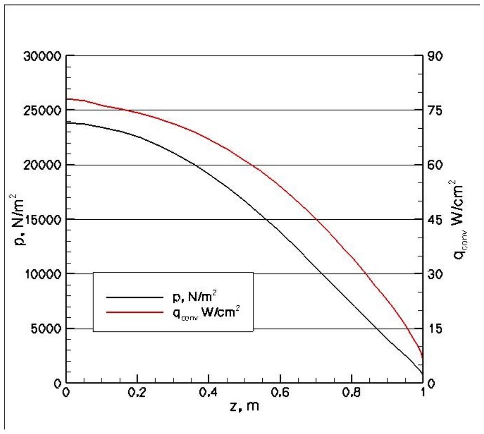
(c) Pressure contours - converged solution after 4000 steps.



(d) Surface pressure and heating - converged solution after 4000 steps.



(e) Pressure contours - 1.e-12 error norm solution after 5600 steps.



(f) Surface pressure and heating - 1.e-12 error norm solution after 5600 steps.

Figure 5: Solutions to the sphere test problem: $V_\infty = 5{,}000$ m/s, $\rho_\infty = 0.001$ kg/m^3, $T_\infty = 200$ K, 5-species air, thermochemical nonequilibrium, radiative equilibrium wall, $\epsilon = 0.89$, and supercatalytic wall boundary.

Changes or additions to the file `laura_namelist_data` are indicated below.

```
irest   = 1 ! 0 for fresh start, 1 for restart
jupdate = 20 ! steps between update of Jacobian
ntran   = 20 ! steps between update of transport properties
nitfo = 0 ! number of 1st-order relaxation steps
rf_inv = 2.0 ! inviscid relaxation parameter
rf_vis = 2.0 ! viscous relaxation parameter
movegrd = 400 ! number of steps between calls to align_shock
maxmoves =  3 ! maximum number of calls to align_shock
frac_line_implicit = 0.7
```

At the conclusion of these next 2,000 steps (4,000 total), the L2 norm has dropped to 0.74e-06 and the solution is converged. The solution for pressure contours after 4,000 steps is shown in Figure 5c. The corresponding surface pressure and heating distributions are shown in Figure 5d. Significant reduction in heating level and smoothing of the stagnation region profile has occurred using the line relaxation across the boundary layer during this second set of 2,000 relaxation steps.

Line relaxation across the entire shock layer (`frac_line_implicit` = 1.0) can be accommodated in this case if the relaxation factors (`rf_inv` and `rf_vis`) are increased to 5. The tolerance for convergence of the L2 norm (rmstol) is set to 1.e-12. Grid movement is switched off (movegrd = 0). The latest grid and restart files are renamed as before to start the third run. The third run reaches the convergence criteria in 1,600 additional relaxation steps, a drop of 6 orders of magnitude in the L2 norm. The solutions (Figure 5e and Figure 5f) are nearly identical to the corresponding figures at 4,000 steps.

```
block =    1  task =   1 err:  0.14E-04  0.50E-05  0.11E-14  0.36E-05  0.40E-05  0.53E-06
step  = 1580  time =   335.39 sum(abs(task error)) =  0.27E-04 L2 norm =   0.10E-11

block =    1  task =   1 err:  0.14E-04  0.48E-05  0.12E-14  0.35E-05  0.39E-05  0.51E-06
step  = 1600  time =   339.63 sum(abs(task error)) =  0.26E-04 L2 norm =   0.98E-12

Aerodynamic Coefficients
c_x = -0.8854
c_y = -0.0000
c_z =  0.6950
c_l = -0.0000
c_m =  0.3514
c_n =  0.0000
```

8.2 Coupled radiation procedure

Starting with the converged nonradiating LAURA solution, the `shuffle_laura` routine must be applied to the `laura.rst` file, and the option to convert from uncoupled to coupled radiation must be chosen—see Section 5.4.11 on page 30 for more info on radiation flags. The new `.rst` file created by `shuffle_laura` must then be renamed to `laura.rst`. Furthermore, the radiation command must be added to the `laura_namelist_data` file:

```
radiation = .true.
```

The first of these, `radiation`, must be set to `.true.`. With this flag other radiation related flags (`nrad`, `frac_rad_new`, `iinc_rad`, `jinc_rad`) will get turned on. The parameter `nrad` specifies the number of flowfield iterations between calls to Hara, which is set to 3000 if it is not specified. `frac_rad_new` specifies the fraction of the most recent Hara calculation applied to the LAURA solution, which is set to `0.8` if it is not specified. `iinc_rad` and `jinc_rad`, respectively, specify the increment in the treated points along the surface and in the spanwise direction; both are default to 3. For `axisymmetric` cases, `jinc_rad` must be `1`. For relatively weakly coupled radiation, such as Earth lunar-return conditions [30], `nrad` may be set to 1,500 for the first two calls to Hara, and 3,000–5,000 for the subsequent calls. For strongly coupled flows, such as Mars-return entry to Earth [20], `nrad` should be set to 500 for the first two calls to Hara, and 1,000 for the rest.

The radiation models applied by the radiation code (HARA) are defined by the optional file `hara_namelist_data` described in Section 5.8 on page 47. If this file is not present in the working directory, then the code determines which radiative mechanisms to apply based on the species number densities in the flowfield.

8.3 Unspecified ablation procedure - Coupled

The recommended procedure for an unspecified ablation computation, meaning the ablation rate and wall temperature is computed as part of the flowfield solution (instead of being specified by the user), is as follows:

1: Obtain a nonablating solution assuming an equilibrium catalytic and radiative equilibrium wall. Include only species required for a nonablating solution.

2: Apply the `shuffle_laura` utility to the converged nonablating solution. Choose the ablation option and increase the number of species to the amount required to accommodate ablation species. Add the ablation species to `tdata`.

3: Modify `laura_namelist_data` to include the following—see Section 5.4.1 on page 15 for more info on ablation parameters:

```
treat_ablation = .true.
mdot_c_approach = 'equilibrium_char'
mdot_g_approach = 'quasi_steady'
temp_approach = 'energy_equation'
ablator_material = 'avcoat'
```

4: Run Laura for roughly 24,000 iterations. During each ablation computation, data are printed out for each point on the body, indicated by `l`. The level of convergence is indicated with `mdot residual` at the end of ablation computation:

$$\texttt{mdot residual} = \sum_l (\Delta \dot{m})^2 \tag{54}$$

Usually, `mdot residual = 1.E-2` or lower indicates that ablation computation is adequately converged within 1%.

5: If the user considers the b-prime approach of sufficient accuracy, then the computation is finished. If the user wishes to apply a rigorous diffusion model at the surface, consistent with that applied throughout the flowfield, then the following modifications should be made to `laura_namelist_data`:

```
      bprime_corrected = .true.
```

Setting `bprime_corrected = .true.` specifies the rigorous diffusion model at the surface [2]. This model is significantly less robust than the default b-prime approach (`bprime_corrected = .false.`), which is why it requires the solution of the b-prime approach as an initial condition.

6: Reevaluate step 4.

References

1. Mazaheri, A.; Gnoffo, P. A.; Johnston, C. O.; and Kleb, B.: Laura Users Manual: 5.4-54166. NASA TM 217092, May 2011.

2. Johnston, C. O.; Gnoffo, P. A.; and Mazaheri, A.: A Study of Ablation-Flowfield Coupling Relevant to the Orion Heatshield. AIAA Paper 2009–4318, 2009.

3. Johnston, C. O.; Gnoffo, P. A.; and Mazaheri, A.: Influence of Coupled Radiation and Ablation on the Aerothermodynamic Environment of Planetary Entry Vehicles. VKI Lecture STO-AVT-218, 2013.

4. Johnston, C. O.: Study of Aerothermodynamic Modeling Issues Relevant to High-Speed Sample Return Vehicles. VKI Lecture STO-AVT-218, 2014.

5. Bartlett, E. P.; Anderson, L. W.; and Curry, D. M.: An Evaluation of Ablation Mechanisms for the Apollo Heat Shield Material. *Journal of Spacecraft and Rockets*, vol. 8, no. 5, 1971, pp. 463–469.

6. Park, C.: Calculation of Stagnation-Point Heating Rates Associated with Stardust Vehicle. *Journal of Spacecraft and Rockets*, vol. 44, no. 1, 2007, pp. 24–32.

7. Moss, J. N.; and Simmonds, A. L.: Galileo Probe Forebody Flowfield Predictions During Jupiter Entry. AIAA Paper 1982–0874, 1982.

8. Stewart, D. A.: Surface Catalysis and Characterization of Proposed Candidate TPS for Access-to-Space Vehicles. NASA TM 112206, July 1997.

9. Scott, C. D.: Catalytic Recombination of Nitrogen and Oxygen on High Temperature Reusable Surface Insulation. *AIAA Progress in Astronautics and Aeronautics: Aerotermodynamics and Planetary Entry*, A. L. Crosbie, ed., AIAA, 1981, pp. 192–213.

10. Steward, D. A.: Determination of Surface Catalytic Efficiency for Thermal Protection Materials — Room Temperature to Their Upper Use Limit. AIAA Paper 1996–1863, 1996.

11. Zoby, E. V.; Gupta, R. N.; and Simmonds, A. L.: Temperatuure-Dependent Reaction Rate Expression for Oxygen Recombination. *AIAA Progress in Astronautics and Aeronautics: Thermal Design of Aeroassisted Orbital Transfer Vehicles*, H. F. Nelson, ed., AIAA, 1985, pp. 445–464.

12. Mitcheltree, R. A.; and Gnoffo, P. A.: Wake Flow about the Mars Pathfinder Entry Vehicle. *Journal of Spacecraft of Rockets*, vol. 32, no. 5, Sep–Oct 1995, pp. 771–776.

13. Cheatwood, F. M.; and Thompson, R. A.: The Addition of Algebraic Turbulence Modeling to Program LAURA. NASA TM 107758, Apr. 1993.

14. Dhawan, S.; and Narasimha, R.: Some Properties of Boundary Layer Flow During the Transition from Laminar to Turbulent Motion. *Journal of Fluid Mechanics*, vol. 3, 1958, pp. 318–436.

15. Srinivasan, S.; Tannehill, J. C.; and Weilmuenster, K. J.: Simplified Curve Fits for the Thermodynamic Properties of Equilibrium Air. NASA RP 1181, June 1987.

16. Prabhu, R. K.; and Erickson, W. D.: A Rapid Method for the Computation of Equilibrium Chemical Composition of Air to 15,000 K. NASA TP 2792, Mar. 1988.

17. Johnston, C. O.; Hollis, B. R.; and Sutton, K.: Spectrum Modeling for Air Shock-Layer Radiation at Lunar-Return Conditions. *Journal of Spacecraft and Rockets*, vol. 45, no. 6, Nov–Dec 2008, pp. 865–878.

18. Johnston, C. O.; Hollis, B. R.; and Sutton, K.: Non-Boltzmann Modeling for Air Shock-Layer Radiation at Lunar-Return Conditions. *Journal of Spacecraft and Rockets*, vol. 45, no. 6, Nov–Dec 2008, pp. 879–890.

19. Johnston, C.; Sahai, A.; and Panesi, M.: Extension of Multiband Opacity-Binning to Molecular, Non-Boltzmann Shock Layer Radiation. *Journal of Thermophysics and Heat Transfer*, vol. 32, no. 3, 2018, pp. 815–820.

20. Johnston, C. O.; Gnoffo, P. A.; and Sutton, K.: The Influence of Ablation on Radiative Heating for Earth Entry. *Journal of Spacecraft and Rockets*, vol. 46, no. 3, May–June 2009, pp. 481–491.

21. Johnston, C. O.; Hollis, B. R.; and Sutton, K.: Radiative Heating Methodology for the Huygens Probe. *Journal of Spacecraft and Rockets*, vol. 44, no. 5, Sep–Oct 2007, pp. 993–1002.

22. Mazaheri, A.: Multi-Species Subsonic Inlet Boundary Condition Formulation with RCS and SRP Applications. International Planetary Probe Workshop (IPPW) 7, Barcelona, Spain, June 2010.

23. Gordon, S.; and McBride, B. J.: Computer Program for calculation of Complex Equilibrium Compositions and Applications. NASA RP 1311, 1994.

24. Gnoffo, P. A.; Gupta, R. N.; and Shinn, J. L.: Conservation Equations and Physical Models for Hypersonic Air Flows in Thermal and Chemical Nonequilibrium. NASA TP 2867, Feb. 1989.

25. Gupta, R.; Yos, J.; Thompson, R. A.; and Lee, K.: A Review of Reaction Rates and Thermodynamic and Transport Properties for an 11-Species Air Model for Chemical and Thermal Nonequilibrium Calculations to 30,000 K. NASA RP 1232, Aug. 1990.

26. Millikan, R. C.; and White, D. R.: Systematics of Vibrational Relaxation. *Chem. Phys.*, vol. 39, no. 12, Dec. 1963, pp. 3209–3213.

27. Ali, A. W.: The Harmonic and Anharmanic Models for Vibrational Relaxation and Dissociation of the Nitrogen Molecule. U.S. Navy NRL Memo 5924, Dec. 1986.

28. Wright, M.: Recommended Collision Integrals for Transport Property Computations Part 1: Air Species. *AIAA J.*, vol. 43, no. 12, 2005, pp. 2558–2564.

29. Wright, M.: Recommended Collision Integrals for Transport Property Computations Part 2: Mars and Venus Entries. *AIAA J.*, vol. 45, no. 1, 2005, pp. 281–288.

30. Gnoffo, P. A.; Johnston, C. O.; and Thompson, R. A.: Implementation of Radiation, Ablation, and Free-Energy Minimization Modules for Coupled Simulations of Hypersonic Flow. AIAA Paper 2009–1399, 2009.

31. Cunto, W.: TOPbase at the CDS. *Astronomy and Astrophysics*, vol. 275, 1993, pp. L5–L8.

Appendix A

Migrating Cases from Prior Versions

Follow the following steps to start LAURA simulations run with LAURA prior to version 5.

Step 1. Create a new working directory and change to it

```
% mkdir [Working Directory]
```

Step 2. Copy the following required files from the old directory:

```
% cp /old/root.rst old.rst
% cp /old/STRTfiles/bound_data.strt laura_bound_data
```

and, depending on the case, these files:

```
% cp /old/root.qtw old.qtw
% cp /old/root.2eq old.2eq
```

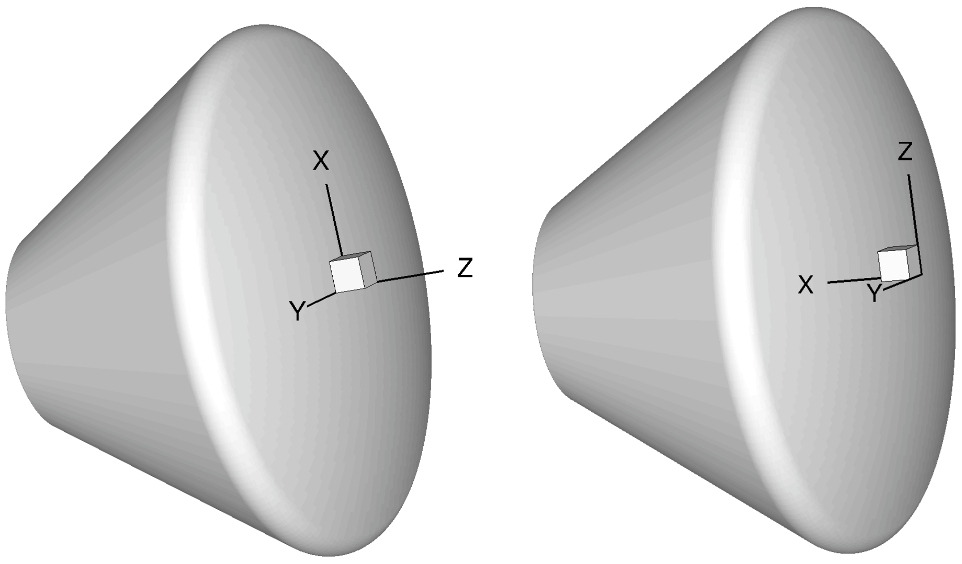
Step 3. Use the LAURA `convert_bound_data` utility, or edit `laura_bound_data` so that there are only integer numbers in this file and there are 6 integers separated by at least one space per line per computational block. For example, the original file may look something like:

```
C:CDATADATADATADATADATADATADATADATADATADATADATADATADATADATADATADATA

C::::: $Name:  $

      data ( itype(i,  1), i=1,6 )
     &         /       1, 1002111,       2,       1,       0,       3 /
      data ( itype(i,  2), i=1,6 )
     &         / 1001211,       1,       2,       1,       0,       3 /

C::CDATADATADATADATADATADATADATADATADATADATADATADATADATADATADATADATA
```

Edit the file to look like this:

```
      1  1002111         2         1         0         3
1001211        1         2         1         0         3
```

Step 4. Run the interactive `convert_laura` utility and answer all the questions. The restart file from LAURA prior to version 5 is single-precision but files produced by LAURA 5 and its utilities are all double-precision. This utility creates `laura.rst` and `laura.g` files. You will have an option either to keep the prior-to-version-5 coordinate system orientation or to rotate the grid to the version 5 coordinate system orientation. As of version 5, LAURA uses $+x$-axis as the body normal direction while prior to version 5 LAURA used the $+z$-axis as the body-normal direction. The two orientations are show in Figure A6.



(a) Prior to Version 5. (b) Version 5.

Figure A6: LAURA coordinate system orientations.

To run LAURA with LAURA grid orientation prior to version 5, use the following formulas to correct the angle-of-attack α, and the center of the moments coordinates:

$$\alpha_{new} = \alpha_{old} - 90 \tag{A55}$$

$$\begin{aligned} (xmc)_{new} &= -(zmc)_{old} && \text{(A56)} \\ (ymc)_{new} &= +(ymc)_{old} && \text{(A57)} \\ (zmc)_{new} &= +(xmc)_{old} && \text{(A58)} \end{aligned}$$

Step 5. Copy the example `laura_namelist_data` file to your working directory from the `[installs_prefix]/share/laura` directory, where `install_prefix` is the installation prefix specified when LAURA was installed.

```
% cp [install_prefix]/share/laura/laura_namelist_data .
```

Use this file as a template to define the simulation parameters. Refer to Section 5.4 on page 14 for a complete list of options. You must change `irest = 0` to `irest = 1`, otherwise the solution will be initialized to free stream conditions.

Step 6. If necessary, create the file `laura_vis_data` (see Section 5.14 on page 57 for more detail.)

Step 7. Modify `tdata` file (see Section 5.5 on page 43) to define the gas model condition for your specific simulation. The species order in this file must match the species order used in the prior version of LAURA. The other data files should not be changed.[A20]

Step 8. Run LAURA as usual—see Section 4 on page 9.

[A20]These files may be changed if different thermodynamic model, curve-fit data, or thermochemical reaction is needed—see Section 5.5 on page 43 for more detail.

Appendix B

Additional Molecular Band Systems

This appendix lists molecular band systems available in addition to those listed in Section 5.8 on page 47. The band systems listed here are generally weak emitters and absorbers, and are therefore not engaged as a default (unlike those listed in Section 5.8 on page 47). Therefore, for these band systems to be engaged, the following flags (0 = off, 1 = SRB, 2 = LBL) must be present in the `hara_namelist_data` file. The LBL treatment of these bands is not recommended.

`treat_band_c2_br`

A flag activating the C_2 Ballik-Ramsay band system.

`treat_band_c2_da`

A flag activating the C_2 Deslandres-d'Azambuja band system.

`treat_band_c2_fh`

A flag activating the C_2 Fox-Herzberg band system.

`treat_band_c2_mulliken`

A flag activating the C_2 Mulliken band system.

`treat_band_c2_philip`

A flag activating the C_2 Philips band system.

`treat_band_co3p`

A flag activating the CO 3+ band system.

`treat_band_co_angstrom`

A flag activating the CO angstrom band system.

`treat_band_co_asundi`

A flag activating the CO Asundi band system.

`treat_band_co_triplet`

A flag activating the CO triplet band system.

`treat_band_co_bx`

A flag activating the CO B-X band system.

`treat_band_co_cx`

A flag activating the CO C-X band system.

`treat_band_co_ex`

A flag activating the CO E-X band system.

`treat_band_n2_cy`

A flag activating the N_2 Carrol-Yoshino band system.

`treat_band_n2_wj`

A flag activating the N_2 Worley-Jenkins band system.

`treat_band_n2_worley`

A flag activating the N_2 Worley band system.

`treat_band_no_gamma`

A flag activating the NO gamma band system.

`treat_band_no_betap`

A flag activating the NO beta-prime band system.

`treat_band_no_gammap`

A flag activating the NO gamma-prime band system.

`treat_band_o2_sr`

A flag activating the O_2 Schumann-Runge band system.

`treat_[?]_photo_dis`

A binary flag activating the molecular photo-dissociation mechanism [31] for `[?]` species, where `[?]` can be O_2 or N_2. This mechanism is not technically a molecular band system.

`treat_[?]_photo_ion`

A binary flag activating the molecular photo-ionization mechanism [31] for `[?]` species, where `[?]` can be O_2 or N_2. This mechanism is not technically a molecular band system.

`treat_no_photo`

A binary flag activating the molecular photo-ionization mechanism [31] for NO.

Appendix C

Trouble Shooting

This appendix gives some tips if you experience trouble installing or running LAURA. This is far from all-inclusive; it is only a modest attempt to capture some of the experience shared by customers over the years through the community LAURA-users@lists.nasa.gov and private LAURA-support@lists.nasa.gov email lists.

Of course the first step is to be sure that you have the latest version of LAURA—releases are announced on the LAURA-news@lists.nasa.gov mailing list.

C.1 Installation

C.1.1 Unterminated Constant / Line Truncated

An error during compilation like,

```
ifort [...] -DDATADIR=\"[some really long path]\" [...] \
  -c -o datadir_file_manager.o datadir_file_manager.F90
In file datadir_file_manager.F90:30
   search_paths(2) = "[some really long pa
                    1
Error: Unterminated character constant beginning at (1)
 In file datadir_file_manager.F90:30
pa
 1
Warning: Line truncated at (1)
make[5]: *** [datadir_file_manager.o] Error 1
```

are due to your installation path violating Fortran's "free-form" line limit of 132 characters. Many compilers do not enforce this; but for those that do, compiler options are usually available to circumvent this problem—for example, gfortran's `-ffree-line-length-none` or g95's `-ffree-line-length-huge`.

C.2 Running

LAURA is designed to recognize when obvious mistakes in input are made: these errors will be reported to you via standard output (nominally, the screen). Even when the code runs successfully, the user should always check the standard output for warning messages.

The ideas given below are mostly for when the code blows up, and the code's output "advice" is not very helpful (NaN detected, segmentation faults, etc.).

Most of the time, problems running the code can be traced to the use of an inadequate grid, block boundary condition errors, or other configuration missteps. The grid should look "nice". There should be "gentle" stretching factors (shoot for less than 1.15), the grid should be as orthogonal as possible—especially in viscous regions, and dramatic changes in distance or orientation from one grid line to the next should be avoided. Furthermore, the grid minimum spacing (at walls) should be appropriate to the problem you are running. In other words, "viscous" grids should have reasonably low cell-Reynolds numbers at the wall (`re_cell` = 0.1–1.0 depending on the amount of diffusion); and one should not try to run inviscid flow on a "viscous" grid.

The first thing to try after a case blows up is to raise the inviscid and viscous relaxation factors—see Section 5.4.9 on page 26. Typically, you can start with values like `rf_inv=4` and `rf_vis=3` and end with values like 3 and 1.5, but sometimes you need to raise them much higher (e.g., 20 and 10 or 200 and 100) to get a solution past a particularly nasty transient. Other things to try include running first-order, increasing the frequency of Jacobian, transport, and interprocessor communication updates, or successively ramping up the Mach number.

Running on a coarser version of the grid sometimes helps get the solution going– see the `coarsen` utility described in Section 7.5 on page 72.

C.2.1 NaNs

NaNs (Not a Numbers) are a type of IEEE Floating Point Exception. Most compilers have options (either during compilation or at runtime via environment variables) to trap these exceptions and stop the code in its tracks when one occurs. For example, the g95 compiler has environment variables of the form `G95_FPU_INVALID` that can control this while the Intel compiler has the `-fpe` compilation option. Note: to pin point where in the code the exception occurs, you will also want to turn on tracing (e.g., `-traceback` for Intel and `-ftrace=full` for g95) and symbols (`-g`) so the compiler will give you the precise source-code line number.

C.2.2 Segmentation Faults

Segmentation faults usually mean you've hit a shell-based memory limit, you've run out of memory, or you've uncovered a coding error.

To check the first, remove your shell memory limits, i.e.,

```
csh: limit stacksize unlimited
bsh: ulimit -s unlimited
     ulimit -d unlimited
     ulimit -m unlimited
```

and try rerunning. Note: For an MPI job, these have to be in your shell startup environment (e.g., `.cshrc` or `.bashrc`).

To check the second, try a smaller case, request more resources (e.g., use less cores per node), and/or use the batch option of the `top` command (`-b`) to monitor memory usage.

For the third, please see Support, section F on page 100.

Appendix D

Radiative Heating Uncertainty Analysis

Radiative heating predictions are highly sensitive to modeling parameters, such as chemical kinetics and non-Boltzmann excitation rates. The significant uncertainty in these modeling parameters corresponds to potentially significant uncertainty in the simulated radiative heating. This radiative heating parametric uncertainty represents a significant fraction of the radiative heating margin applied for vehicle design, with the rest of the margin based primarily on comparisons with available experimental data (mostly from shock tubes). This radiative heating margin is often greater than 50%, which means it can a have significant impact on the heatshield sizing. However, it is typically computed (through detailed parametric uncertainty analyses and comparisons with shock tube measurements) for a single vehicle body point (usually the stagnation-point) at perhaps only a single trajectory point (usually peak heating). These limited evaluations are the result of complexity in the parametric uncertainty analysis, due to the large number of parameters involved, and difficulty in relating shock tube measurements to flight vehicle radiative heating.

To remove these difficulties and enable routine evaluations of the parametric uncertainty and shock-tube based components to the radiative heating margin, the `hara_uq` code was developed. This code, which is run as a post-processing step to a converged `laura` solution, performs these analyses automatically. The application of `hara_uq` is recommended as the standard approach for radiative heating simulations that support vehicle design projects. It provides both the nominal ray-tracing radiative heating values and the corresponding parametric uncertainty and shock-tube informed bias, both of which represent the dominant contributions to the radiative heating margin.

D.1 Running `hara_uq`

The `hara_uq` code is designed to require minimal user input. Staring with a converged `laura` solution that includes coupled radiation (the convergence here refers to the radiative heating not changing with further iterations, not necessarily a small residual), the primary task of the user is to decide the number of processors to use. This number should be a multiple of the number of blocks. Because `hara_uq` internally runs simultaneous `laura` cases, the larger the number of processors results in the code running more simultaneous `laura` cases. In other words, increasing the number of processors to a large multiple of the number of blocks will not result in multiple processors per block, which may result in performance penalties.

Once the number of processors is chosen, `hara_uq` is run identically to `laura`. The defaults are set so that the code considers radiation spectrum modeling parameters, non-Boltzmann atomic and molecular rates from the strongest radiating atom and molecule, and flowfield chemical kinetics in determining the radiative heating parametric uncertainty. A sensitivity analysis is first run for each of these parameters assuming tangent-slab. The results of this sensitivity analysis are saved in the `laura_sensitivity_rad.q` and `uncertainty_matlab_out.m` files. This sensitivity analysis informs a set of parameter adjustments to determine the parametric uncertainty. Ray-tracing is applied by default for this parametric uncertainty computation. Finally, the shock tube informed correction to the radiative heating is evaluated using the approach presented by Johnston. Both the parametric uncertainty and the shock-tube informed correction to the radiative heating are automatically added to the `laura_surface.q` and `laura_blayer.dat` files, which are renamed `laura_surface_uq.q` and `laura_blayer_uq.dat`. The nominal ray-tracing radiative heating computed by `hara_uq` is also added to these files.

The computational time required by `hara_uq` can be significant. It is roughly equivalent to 20 to 40 `laura` solutions run for 5000 flowfield iterations and ray-tracing performed for all body points defined by `iinc_rad` and `jinc_rad` in `laura_namelist_data`. Fortunately, these 20 to 40 solutions are embarrassingly parallel, so that increasing the number of processors (in increments of the number of blocks) reduces this time accordingly. However, for larger scale problems where sufficient processors are not available, many of the default options may be altered to reduce the computational time. This can be done with minimal loss in accuracy assuming some knowledge of the problem of interest. For example, an Earth entry case at a velocity below 9 km/s will experience minimal atomic radiation, and therefore, the atomic radiation modeling parameters may be ignored by `hara_uq`, which reduces the computational time significantly. Options such as this are implemented through the `uq_namelist_data` file, which is only required if a change to a default option is required. These options are described in the following section.

D.2 `uq_namelist_data`

Below are the parameters that control the `hara_uq` analysis.

`apply_ray_tracing`

A logical flag that controls if the uncertainty is computed using ray-tracing instead of tangent-slab. Setting this option to `.true.` is recommended to maintain accuracy away from the stagnation point, although it increases the computational time significantly. The sensitivity analysis applies tangent-slab regardless.

Default: `.true.`

`apply_streamline_uncertainties`

A logical flag that controls if the shock-tube informed bias to the radiative heating is computed using the streamline approach presented by Johnston. This option is currently available only for Earth, Mars and Titan entry.

Default: `.true.`

`compute_refined_nominal`

A logical flag that controls if the nominal radiative heating is computed using ray-tracing on fully-refined angular grid (the uncertainty analysis computes relative differences in heating on a coarsened angular grid). Engaging this option results in the nominal ray-tracing radiation being output in the `laura_surface.q` and `laura_blayer.dat` files along with the associated uncertainty. This is recommended for final vehicle design database simulations.

Default: `.true.`

`molec_nonboltz_study`

A logical flag that controls if the non-Boltzmann rates for molecules are considered in the uncertainty analysis. This flag defaults to `.true.` for entry velocities below 10.4 km/s, and to `.false.` above this velocity where the molecular band contribution becomes small.

`atomic_nonboltz_study`

A logical flag that controls if the non-Boltzmann rates for atoms are considered in the uncertainty analysis. This flag defaults to `.true.` for entry velocities above 9.3 km/s, and to `.false.` below this velocity where the atomic line contribution becomes small.

Default: `.true.`

`ionization_potential_study`

A logical flag that controls if the ionization potential for atoms are considered in the uncertainty analysis. This modifies both the ionization potential used for the equilibrium constant in the flowfield, but also the ionization limit in the radiation computation.

Default: `.true.`

`diffusion_ccs_study`

A logical flag that controls if diffusion cross sections are considered in the uncertainty analysis. The default for this is `.false.` because these typically have a negligible impact on radiative heating.

Appendix E

Ray-Tracing Radiation Utility

When running coupled radiation as described in Section 8.2, the radiative heating and the divergence of the radiative flux (the latter being the term applied to the flowfield energy equations) are computed assuming a tangent slab geometry along grid lines normal to the body. This approximation typical results in a 5–15% overprediction in the radiative flux on the forebody of a vehicle, where the shock layer is relatively thin. For the backshell region of a vehicle, however, the non-tangent-slab nature of the flow results in the tangent slab approximation being in significant error. The `ray_tracing` utility included with LAURA and described in this section removes the tangent slab approximation and provides a rigorous ray-tracing computation for the radiative heating, which allows for accurate radiative heating predictions in non-tangent-slab regions of the flow. Note that `ray_tracing` may be applied to cases that do not include coupled radiation.

The `ray_tracing` utility computes the radiative flux at surface point by numerically integrating the following equation:

$$q_{rad} = \int_0^{+\infty} \int_0^{+\pi/2} \int_0^{2\pi} I_\nu(\theta, \phi) \sin\phi \cos\phi \, d\theta \, d\phi \, d\nu \qquad \text{(E59)}$$

where the radiative intensity I_ν is computed for a specified number of angles ϕ and θ, which are defined in Fig. E7.

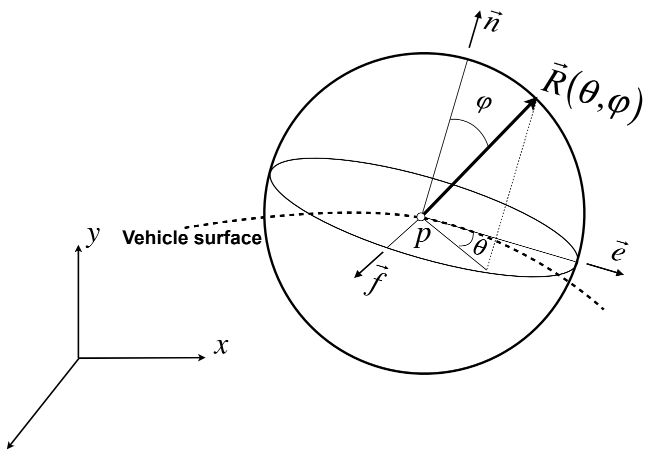


Figure E7: Rays polar coordinate definition.

The `ray_tracing` code is intended to require minimal user input, beyond the options chosen for `hara_namelist_data` and choosing the number of processors. Because the evaluation of radiation along the numerous lines-of-sight required by `ray_tracing` is embarrassingly parallel, `ray_tracing` should be run using 50–500 processors. The default application of `ray_tracing` will compute the ray-tracing radiative flux at every surface point defined by `iinc_rad` and `jinc_rad` in `laura_namelist_data` (they are both 3 by default). A total of 240 lines-of-sight are computed for each surface point. And, an axisymmetric solution will automatically be rotated to form a full three-dimensional solution, and a half-body three-dimensional solution will automatically be mirrored (ray-tracing will not be computed to the additional surface points introduced by this rotation or mirroring). The computed ray-tracing radiative flux values are added to the `laura_surface.q` and `laura_blayer.dat` files, which are renamed `laura_surface_ray.q` and `laura_blayer_ray.dat`. If these defaults are not sufficient, then the following two files may be used to control `ray_tracing`.

1: `ray_tracing.nml`: Controls the number of rays and output options.

2: `run_list`: Controls the surface locations at which the radiation is computed.

These two files are discussed in the following subsections.

E.1 `ray_tracing.nml`

Below are the parameters that control the ray-tracing utility.

`auto_mirror`

A logical flag that controls if `laura.g` and `laura.q` are automatically mirrored to convert a half-body 3D solution to a full-body solution. The mirrored axis is defined by `mirror_axis`, with the y-axis as the default. This option is ignored for axisymmetric cases.

Default: `.true.`

`mirror_axis`

Integer that defines the mirrored axis, where x-axis = 1, y-axis = 2, and z-axis = 2. Default: `2`

`nphi`

Number of equally-spaced ϕ increments, in degrees. Note that ϕ spans from 0 to 90 degrees. Default: `19`

`ntheta_min`

Number of θ increments at $\phi = 0$ degrees. Default: `10`

`ntheta_max`

Number of θ increments at $\phi = 90$ degrees. Default: `20`

`tangent_slab_call`

A logical flag to select if a tangent slab computation is to be made in addition to the ray tracing. The ray normal to the surface is used for this computation, which may differ from the k-line used by LAURA for radiation computations. Default: `.true.`

add_to_laura_surface

A logical flag to select if the ray-tracing radiation is added to the `laura_surface.q` and `laura_blayer.dat` files. This option forces the all body points defined by `iinc_rad` and `jinc_rad` in `laura_namelist_data` to be considered. Default: `.true.`.

centerline_only

A logical flag to select if the ray-tracing radiation is computed along only the centerline of a half-body three-dimensional solution. This provides a convenient approach for reducing the number of body points to consider for large three-dimensional cases. Default: `.false.`.

E.2 run_list

Because ray tracing computations are relatively computationally expensive, it is typically not convenient to compute the radiation to equally spaced i and j locations along the surface, especially for three-dimensional cases. The `run_list` files provide a convenient approach for defining the points at which ray tracing is computed. The first line of the file defines the number of points to be treated, followed by the corresponding block number, i, and j location for each surface point to be treated. For example:

```
2
5 2 1
6 4 2
```

computes the radiative flux on block 5, i=2, j=1 and block 6, i=4, j=2.

Appendix F

Support

Because LAURA's primary purpose is to serve NASA missions and it is not offered as commercial software, support is available on an "as time allows basis" via two channels: the public LAURA-users@lists.nasa.gov community email list, which you are encouraged to join via lists.nasa.gov/mailman/listinfo/LAURA-users, and the private LAURA-support@lists.nasa.gov email list.

If your issue is not proprietary or otherwise sensitive, you are strongly encouraged to use and become a member of the LAURA-users@lists.nasa.gov list.

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704–0188*

1. REPORT DATE *(DD-MM-YYYY)* 01-02-2020	**2. REPORT TYPE** Technical Memorandum	**3. DATES COVERED** *(From - To)*

4. TITLE AND SUBTITLE LAURA Users Manual: 5.6	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Thompson, Kyle B.; Hollis, Brian R.; Johnston, Christopher O.; Kleb, Bil; Lessard, Victor R.; Mazaheri, Ali R.	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA	**8. PERFORMING ORGANIZATION REPORT NUMBER** L–21103
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001	**10. SPONSOR/MONITOR'S ACRONYM(S)** NASA
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) NASA/TM–2020–220566

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category 64
Availability: NASA STI Program (757) 864-9658

13. SUPPLEMENTARY NOTES
An electronic version can be found at http://ntrs.nasa.gov.

14. ABSTRACT
This users manual provides in-depth information concerning installation and execution of LAURA, version 5. LAURA is a structured, multiblock, computational aerothermodynamic simulation code. Version 5 represents a major refactoring of the original Fortran 77 LAURA code toward a modular structure afforded by Fortran 95. The refactoring improved usability and maintainability by eliminating the requirement for problem-dependent recompilations, providing more intuitive distribution of functionality, and simplifying interfaces required for multi-physics coupling. As a result, LAURA now shares gas-physics modules, MPI modules, and other low-level modules with the FUN3D unstructured-grid code. In addition to internal refactoring, several new features and capabilities have been added, e.g., a GNU-standard installation process, parallel load balancing, automatic trajectory point sequencing, free-energy minimization, and coupled ablation and flowfield radiation.

15. SUBJECT TERMS
Aerodynamics; Aerothermodynamics; Ablation; Radiation

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			STI Information Desk (help@sti.nasa.gov)
U	U	U	UU	105	**19b. TELEPHONE NUMBER** *(Include area code)* (757) 864-9658