Input Variable	Type	Default Value	Skill Level	Description
grid_format	integer	-1	basic	Specifies the format of the input grid file. -1 = Internally generated grid (automatically set for certain test cases) 1 = Gambit Neutral 2 = Gmsh 3 = Plot3D 4 = CGNS
gridfile	character(len=150)	" "	basic	File path to the input grid file relative to the current directory.
plot3d_cutfile	character(len=150)	" "	basic	File path to the input file that specifies the cut conditions for a multiblock Plot3D grid file.
plot3d_bcsfile	character(len=150)	" "	basic	File path to the input file that specifies the boundary conditions for a Plot3D grid file.
grid_scaling_factor	integer	1.0	basic	Multiplication factor (> 0.0) to scale the grid coordinates to the desired length units. For example, use a value of 0.0254 to convert a grid containing coordinates in inches to meters.
plot3d_i_stride	integer	-1	advanced	If > 0, applies a coarsening stride to the $\boldsymbol{i}$-direction of all Plot3D blocks. **NOTE:** This overrides the $\boldsymbol{i}$-direction coarsening stride given by *plot3d_all_strides*.
plot3d_j_stride	integer	-1	advanced	If > 0, applies a coarsening stride to the $\boldsymbol{j}$-direction of all Plot3D blocks. **NOTE:** This overrides the $\boldsymbol{j}$-direction coarsening stride given by *plot3d_all_strides*.
plot3d_k_stride	integer	-1	advanced	If > 0, applies a coarsening stride to the $\boldsymbol{k}$-direction of all Plot3D blocks. **NOTE:** This overrides the $\boldsymbol{k}$-direction coarsening stride given by *plot3d_all_strides*.
plot3d_all_strides	integer	-1	advanced	Applies a coarsening stride to all $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$ directions of all Plot3D blocks. **NOTE:** The coarsening stride for any given direction can be overridden by specifying the coarsening stride for that direction.
plot3d_agglomerate_order	integer	-1	basic	If > 0, agglomerates linear Plot3D cells into high-order grid cells of this order. **NOTE:** For all blocks, the number of cells in each direction must be divisible by this order.
solution_order	integer	3	basic	Specifies the degree of the polynomial space, $\mathcal{P}_S$, to use for the solution. **NOTE:** The order-of-accuracy is approximately $\mathcal{P}_S + 1$.
flux_divergence	integer	1	not working	Specifies the method to use for computing the divergence of the fluxes. 1 = using Lagrange polynomials 2 = using chain rule 3 = using chain rule with Jacobians

correction_function	integer	1	advanced	Specifies the correction function to use. 1 = g_{DG} correction function (recovers nodal DG method) 2 = g_2 correction function 3 = g_{Ga} correction function
loc_solution_pts	integer	1	advanced	Specifies the location of the solution points used for tensor products. 1 = Legendre-Gauss nodes 2 = Legendre-Gauss-Lobatto nodes
loc_flux_pts	integer	1	advanced	Specifies the location of the flux points on cell faces. 1 = Legendre-Gauss nodes 2 = Legendre-Gauss-Lobatto nodes
loc_triangle_pts	integer	2	developer	Specifies the location of the solution points used for triangles. 1 = Hesthaven-Warburton α-optimized nodes 2 = Barycentric Lobatto nodes
loc_init_pts	integer	0	expert	Nodal points used for initializing the flow, after which the initial solution is projected to the solution points. 1 = Legendre-Gauss nodes 2 = Legendre-Gauss-Lobatto nodes **NOTE:** Any other values besides these will result in *loc_init_pts* = *loc_solution_pts*.
Runge_Kutta_Scheme	integer	3	basic	Specifies which Runge-Kutta method to use. 1 = Classic n-stage RK method 2 = 2-stage/2$^{\mathrm{nd}}$-order SSP-RK method 3 = 3-stage/3$^{\mathrm{rd}}$-order SSP-RK method 4 = 5-stage/4$^{\mathrm{th}}$-order SSP-RK method 5 = 5-stage/4$^{\mathrm{th}}$-order Carpenter-Kennedy low storage RK method
num_rk_stages	integer	3	basic	Gives the number of Runge-Kutta stages if using the classic RK method (*Runge_Kutta_Scheme* = 1).
invs_flux_method	integer	1	advanced	Specifies the approximate Riemann solver used to compute the common inviscid fluxes at the interfaces. 1 = Roe flux with entropy fix 2 = HLLC flux 3 = LDFSS flux 4 = Rotated-Roe-HLL flux
visc_flux_method	integer	2	advanced	Specifies the method used to compute the common viscous fluxes at the interfaces. 1 = Bassi-Rebay 1 (BR1) (**WARNING:** high chances of stability issues) 2 = Bassi-Rebay 2 (BR2) 3 = Local DG (LDG) (smaller CFL limit than BR2)

visc_variables_to_ave	integer	1	expert	Determines which variables are averaged to the interfaces when computing the common viscous fluxes. 1 = Average the interface fluxes. Compute the common viscous flux by averaging the left and right viscous interface fluxes which are computed using the left and right interface solutions and gradients. 2 = Average the interface solutions and gradients. Compute common solutions and gradients by averaging the left and right interface solutions and gradients, and then use the common interface values to compute the common viscous flux.
num_timesteps	integer	25000	basic	Number of time steps for the simulation.
Final_Time	integer	10.0	basic	Stop when the time within the simulation reaches this value.
constant_dt	integer	1.0e-6	basic	Size of the time step that is used when *Timestep_Type* is set to a constant global time step.
Timestep_Type	integer	1	basic	Type of time stepping to use for the simulation. 0 = Constant global time step. 1 = global time step from minimum computed cell time step. -1 = local time stepping within each cell from computed cell time step. 2 = global time step from minimum computed solution point time step. (EXPERIMENTAL) -2 = local time stepping at each solution point from compute solution point time step. (EXPERIMENTAL)
minimum_timesteps	integer	100	basic	The minimum number of time steps to complete before checking whether any of the convergence criteria have been met. **NOTE:** When initializing a simulation to a uniform flow, the very small changes that occur during the first few time steps can trigger a false positive convergence check if this is set to a small number, e.g., ≤ 5.
CFL	integer	0.1	basic	Analogous to the CFL condition, this is used to scale the time step size to get a maximum stable time step. The maximum value this can be while also keeping a simulation stable is very dependent on the simulation/flow-conditions and input settings, and experimentation is generally required to find a good value. A lower value is more likely to keep a simulation stable, but it will require more time steps to reach either convergence or a given simulation time. Typical values are $0.1 \leq CFL \leq 2$, but past experience has found simulations that were stable with values up to 7.
CFL_Beg	integer	0.1	basic	Beginning *CFL* value to use with the linear CFL ramping that is active when *CFL_Cycles* > 2.
CFL_End	integer	0.1	basic	Ending *CFL* value to use with the linear CFL ramping that is active when *CFL_Cycles* > 2.

CFL_Cycles	integer	1	basic	The number of time steps used to linearly ramp up the value of *CFL*, starting at *CFL_Beg* and ending at *CFL_End*.
projection_order	integer	3	advanced	Specifies the degree of the polynomial space, $\mathcal{P}_P$, to project the residual onto when using over-integration by residual projection. The residual will be computed as a polynomial of degree $\mathcal{P}_S$ and projected down to a polynomial of degree $\mathcal{P}_P$. This can potentially improve stability if aliasing errors are suspected of destabilizing a simulation. **NOTE:** This is only used when $\textit{projection\_order} < \textit{solution\_order}$. **NOTE:** Using this means the order-of-accuracy of the simulation is now approximately $\mathcal{P}_P + 1$ instead of $\mathcal{P}_S + 1$.
quadrature_order	integer	3	not working	Specifies the degree of the quadrature polynomial space, $\mathcal{P}_Q$, to use when using over-integration by over-sampling at the quadrature points. Certain operations during a given residual evaluation (e.g., evaluating the common/upwind interface fluxes) produce either polynomials that are not within the solution polynomial space or functions that are not polynomials at all. These functions are over-sampled at the quadrature points to produce approximations of these functions as polynomials of degree $\mathcal{P}_Q$, which are in turn projected down to polynomials of degree $\mathcal{P}_S$ before adding their respective contribution to the residual. This can potentially improve stability if aliasing errors are suspected of destabilizing a simulation. **NOTE:** $\textit{quadrature\_order} \geq \textit{solution\_order}$ **NOTE:** This is only used when $$\text{or} \quad \begin{array}{l} \textit{quadrature\_order} > \textit{solution\_order} \\ \textit{loc\_quadrature\_pts} \neq \textit{loc\_solution\_pts} \end{array}$$ **NOTE:** Unlike over-integration by residual projection, the order-of-accuracy of the simulation remains $\mathcal{P}_S + 1$.
loc_quadrature_pts	integer	1	not working	Specifies the location of the quadrature points used for over-integration by over-sampling at the quadrature points. 1 = Legendre-Gauss nodes 2 = Legendre-Gauss-Lobatto nodes
error_order	integer	-3	advanced	Specifies the degree of the polynomial space to use when computing integrated error quantities, e.g., the L^2-norm of the residual. **NOTE:** $\textit{error\_order} \geq \textit{solution\_order}$ **NOTE:** This is only used when $$\text{or} \quad \begin{array}{l} \textit{error\_order} > \textit{solution\_order} \\ \textit{loc\_error\_pts} \neq \textit{loc\_solution\_pts} \end{array}$$

loc_error_pts	integer	1	advanced	Specifies the location of the error points used for computing integrated error quantities. 1 = Legendre-Gauss nodes 2 = Legendre-Gauss-Lobatto nodes **NOTE:** Any other values besides these will result in *loc_error_pts* = *loc_solution_pts*.
all_orders	integer	-3	basic	If > 0, sets *solution_order*, *projection_order*, and *quadrature_order* all to this value. **NOTE:** This overrides the values given for these three input parameters.
all_points	integer	-1	advanced	Sets *loc_solution_pts*, *loc_flux_pts*, and *loc_quadrature_pts* all to this value. 1 = Legendre-Gauss nodes 2 = Legendre-Gauss-Lobatto nodes **NOTE:** This overrides the values given for the three mentioned input parameters. It is only used if equal to one of the above values; otherwise, the three mentioned input parameters are unchanged.
metis_option	integer	3	expert	Specifies the METIS partitioning algorithm to use for partitioning the global grid. 1 = Create non-weighted partitions based on a multilevel recursive bisection while minimizing the edge cut. 2 = Create non-weighted partitions based on a multilevel k-way algorithm while minimizing the edge cut. 3 = Create non-weighted partitions based on a multilevel k-way algorithm while minimizing the total communication volume. 4 = Create weighted partitions based on a multilevel k-way algorithm while minimizing the total communication volume (weighting is not yet implemented so this is currently the same as option 3).
mms_opt	integer	0	experimental	Specifies the method of manufactured solutions (MMS) problem to use.
mms_init	integer	0.1	experimental	Initializes the solution to [*mms_init* $\times$ (exact MMS solution)].
mms_output_solution	integer	.FALSE.	experimental	Logical flag to completely skip the solver and just output the exact MMS solution.

itestcase	integer	9	basic	Specifies the test-case/problem to solve, primarily affecting the initialization of the simulation. 1 = Generic flow, initialize to reference conditions. 2 = Channel flow (not working). 4 = Laminar subsonic boundary layer, initialize to reference conditions. 5 = Diagonally propagating Shu version of the isentropic Euler vortex. -5 = Stationary Shu version of the isentropic Euler vortex. 6 = Isentropic Euler vortex with propagation direction determined by *alpha_aoaref*. -6 = Isentropic Euler vortex propagating only in the y-direction. 7 = Advection of a density profile using the linear advection equation. 8 = Advection of an entropy profile using the linear advection equation. 9 = Taylor-Green vortex problem. 10 = Special initialization function for SMC000 nozzle jet case. 11 = Double Mach reflection (not working). 12 = Infinite cylinder. 13 = Freestream preservation
load_restart_file	integer	.FALSE.	basic	Logical flag used to indicate whether or not to read in a restart file.
restart_interval	integer	0	basic	Number of time steps between writing restart files. **NOTE:** A restart file is always written after completing the last time step.
use_old_restart_format	integer	.FALSE.	developer	Logical flag to indicate that the restart file is in an older format used by an early conceptual version of GFR. This should NEVER be used!
restart_file	character(len=150)	"out.rst"	basic	File path to the solution restart file that will be read if *load_restart_file* is true.
restart_output_uses_host_roots	integer	.TRUE.	expert	Logical flag that indicates whether all MPI processes are involved in writing to the restart file or if only the root processes on each host node write to the restart file. **NOTE:** This is only applicable for parallel simulations run across multiple physical computers, e.g., multiple nodes on the Pleiades supercomputer. **NOTE:** Only change this if you encounter MPI errors that occur when trying to write restart files. Changing this to false will most likely result in higher I/O file contention and possibly increased memory usage.

iter_out_interval	integer	1	basic	Number of time steps between writing time-stepping and residual/convergence statistics to standard output. **NOTE:** When running the Taylor-Green vortex problem, this is also used as the interval for writing integrated kinetic energy dissipation rate (KEDR) statistics to the file "Taylor-Green_KEDR.dat". **NOTE:** If = 0, $iter\_out\_interval = 1$ else if < 0, $iter\_out\_interval = \frac{num\_timesteps}{\lvert iter\_out\_interval \rvert}$
results_interval	integer	0	basic	Number of time steps between writing residual/convergence statistics for all conserved and primitive variables to the file "results.dat". **NOTE:** If = 0, $results\_interval = \frac{num\_timesteps}{500}$ else if < 0, $results\_interval = \frac{num\_timesteps}{\lvert results\_interval \rvert}$
output_dir	character(len=150)	"."	basic	File path to the directory used for writing output files. **NOTE:** If one is available, it is recommended that the output directory be located on a Lustre file system in order to get the best I/O performance possible.
output_interval	integer	-5	basic	Number of time steps between writing CGNS solution files.
continuous_output	integer	.FALSE.	experimental	Logical flag to enable/disable the post-processor that attempts to create a smooth, continuous solution across all cells before writing it to the CGNS solution file.
output_order	integer	-3	advanced	Specifies the degree of the polynomial space used to represent the solution in the CGNS solution files. This simply allows the ability to over-sample the solution polynomial which provides a smoother solution within each cells when visualizing the solution. **NOTE:** This is only used if $output\_order > solution\_order$.
loc_output_pts	integer	0	expert	Specifies the location of the nodal points within the grid cells when writing the solution to the CGNS solution files. 0 = Equi-distant nodes 1 = Legendre-Gauss nodes 2 = Legendre-Gauss-Lobatto nodes **NOTE:** Using Legendre-Gauss nodes for the CGNS solution files will result in empty space between all grid cells because there is no connectivity information between cells.
governing_equations	integer	2	basic	Specifies the governing equations to use. 1 = Euler equations 2 = Navier-Stokes equations

turbulence_model	integer	0	not working	Specifies the turbulence model to use. 0 = Laminar/Implicit LES 1 = 1998 k-ω (not working) -1 = 2006 k-ω (not working)
machref	integer	0.4	basic	Reference Mach number.
reyref	integer	-1.0	basic	Reference Reynolds number (default evaluates to 8.7319e+6).
gam	integer	1.4	basic	Reference γ (ratio of specific heats).
ptotref	integer	-1.0	not working	Reference total pressure (units of Pa).
ttotref	integer	-1.0	not working	Reference total temperature (units of Kelvin).
ptot2p_ratio	integer	-1.0	not working	Ratio of reference total pressure to reference static pressure.
rhoref	integer	-1.0	basic	Reference static density (default evaluates to 1.160833 $\frac{\mathrm{kg}}{\mathrm{m}^3}$).
tref	integer	300.0	basic	Reference static temperature (units of Kelvin).
pref	integer	100000.0	basic	Reference static pressure (units of Pa).
rgasref	integer	287.15	basic	Reference gas constant (units of $\frac{\mathrm{m}^2}{\mathrm{s}^2\mathrm{K}}$).
tinref	integer	0.0	not working	Reference turbulence intensity. (not working)
tvrref	integer	1.0e-6	not working	Reference ratio of eddy viscosity to molecular viscosity. (not working)
alpha_aoaref	integer	0.0	basic	Reference angle of attack in the x-y plane.
beta_aoaref	integer	0.0	basic	Reference angle of attack in the x-z plane.
Pr	integer	0.72	basic	Laminar Prandtl number.
Prt	integer	0.72	not working	Turbulent Prandtl number. (not working)
suth_muref	integer	1.716e-5	basic	Constant μ_0 in Sutherland's law.
suth_Tref	integer	273.0	basic	Constant T_0 in Sutherland's law.
suth_Sref	integer	110.4	basic	Constant S in Sutherland's law.

Filter_Option	integer	0	advanced	Specifies the type of standard filter to use for stabilization. $= 0 \Rightarrow$ The standard filter is not used. **For Structured Elements** $= 1 \Rightarrow$ Use a tensor product of the 1D exponential filter which amounts to the 1D filter being applied separately for each direction where $N =$ *solution_order* for 1D, 2D, and 3D $= 2 \Rightarrow$ Use a true tensor product of the 1D exponential filter where $N = 2 \times$ *solution_order* for 2D $N = 3 \times$ *solution_order* for 3D **For Unstructured Elements** $\neq 0 \Rightarrow$ Use the exponential filter where $N =$ *solution_order* for 1D, 2D, and 3D The exponential filter is defined as: $N_c =$ filter cutoff and must have a value $0 \leq N_c \leq N$ $\eta_c = \frac{\text{real}(N_c)}{\text{real}(N)}$ `filter_diagonal`$(0\!:\!N) = 0.0$ `do` $i = N_c, N$ $\eta = \frac{\texttt{real}(i)}{\texttt{real}(N)}$ `filter_diagonal`$(i) = (1.0 - \varepsilon)\exp\left[-\alpha\left(\frac{\eta - \eta_c}{1.0 - \eta_c}\right)^s\right]$ `end do`
Filter_etac	integer	0.0	advanced	Modal cutoff (η_c) for the standard filter, below which the lower modes are left untouched. **NOTE:** The value of *Filter_etac* must be between 0 and 1.
Filter_Order	integer	16	advanced	Order (s) of the exponential filter.
Filter_Alpha	integer	36.0	advanced	Maximum damping parameter (α) for the standard parameter.
Filter_eps	integer	0.0	advanced	Leading damping parameter (ε) for the standard filter.

Limiter_Filter_Option	integer	0	advanced	The details for this are the same as for *Filter_Option*, but this is a second filter that is only used as a limiter on any cells that have been marked as a troubled cell. **NOTE:** The idea is you can use *Filter_Option* as a very weak filter that is constantly being applied regardless of whether or not the solution is oscillatory within the cell, and you can use *Limiter_Filter_Option* as a stronger filter that is only applied to cells that have been marked as a troubled cell to prevent the oscillatory solutions in these cells from causing the simulation to diverge.
Limiter_Filter_etac	integer	0.0	advanced	Same as *Filter_etac* but for the limiter filter.
Limiter_Filter_Order	integer	16	advanced	Same as *Filter_Order* but for the limiter filter.
Limiter_Filter_Alpha	integer	36.0	advanced	Same as *Filter_Alpha* but for the limiter filter.
Limiter_Filter_eps	integer	0.0	advanced	Same as *Filter_eps* but for the limiter filter.
Limiter_Option	integer	0	advanced	Specifies the type of limiter to use. $= 0$ $\Rightarrow$ Do not use a limiter. $= -1$ $\Rightarrow$ Use the resolution indicator to identify troubled cells. For any troubled cells, project the solution down to a order that is dynamically chosen by the value of the resolution indicator; cells with higher indicator values will be projected to lower orders, whereas cells with lower indicator values will be projected down only an order or two from the solution order. $= 1$ $\Rightarrow$ Use the resolution indicator to identify troubled cells. The resolution indicator is based on the integral of the (P-1)-order solution density divided by the (P)-order solution density. $= 2$ $\Rightarrow$ Use the jump indicator to identify troubled cells. The jump indicator is based on the pressure difference with neighboring cells for all the faces of a cell. If *Limiter_Filter_Option* $= 0$, the limiting will be performed by projecting the solution in each troubled cell down by one order, after which all troubled cells will be rechecked to see if they are still troubled. If any are still troubled the solution in those cells will be projected down another order, and this loop continues until there are no more troubled cells or the remaining troubled cells have been reduced to P1 solutions. If *Limiter_Filter_Option* $\neq 0$, the limiter filter will be applied once to all troubled cells.

Filtering_Interval	integer	0	advanced	Specifies the frequency to apply the standard filter. $= 0 \Rightarrow$ Apply the standard filter after each Runge-Kutta stage. $> 0 \Rightarrow$ Apply the standard filter after the last Runge-Kutta stage if $\text{mod}\,(\textit{Filtering\_Interval}, \textit{current\_time\_step}) == 0$. $< 0 \Rightarrow$ The standard filter will NOT be applied even if $\textit{Filter\_Option} \neq 0$
Limiting_Interval	integer	0	advanced	Specifies the frequency to apply the limiter. $= 0 \Rightarrow$ Apply the limiter after each Runge-Kutta stage. $> 0 \Rightarrow$ Apply the limiter after the last Runge-Kutta stage if $\text{mod}\,(\textit{Limiting\_Interval}, \textit{current\_time\_step}) == 0$. $< 0 \Rightarrow$ The limiter will NOT be applied even if $\textit{Limiter\_Option} \neq 0$
vortex_bigR	integer	1.0	special case	Constant used for the isentropic Euler vortex problem.
vortex_bigGam	integer	5.0	special case	Constant used for the isentropic Euler vortex problem.
VortexGrid_Lref	integer	10.0	special case	Reference length for the grid domain, i.e., the grid is a square with bounds of $[-\textit{VortexGrid\_Lref}, +\textit{VortexGrid\_Lref}]$ in both the x- and y-directions.
VortexGrid_DOF	integer	48	special case	Number of solution points /degrees-of-freedom (DoF) in each direction to use for the isentropic Euler vortex and Taylor-Green vortex problems.
VortexGrid_random_perturb	integer	.FALSE.	special case	Logical flag for applying a random perturbation to all the interior grid nodes for the isentropic Euler vortex and Taylor-Green vortex problems.
VortexGrid_perturb_percent	integer	30	special case	Percent of the uniform grid cell length to perturb the interior grid nodes if *VortexGrid_random_perturb* is true.
channel_body_force	integer	0.0	not working	Body force used to drive the channel flow problem. (not working)
channel_init_file	character(len=150)	“ ”	not working	File path to the file used to initialize the solution for the channel flow problem. (not working)

walls_are_exact	integer	.TRUE.	advanced	Logical flag that determines if the solution on wall boundary conditions are treated as either the exact flow conditions or the ghost state that equals the exact flow conditions when averaged with the interior state. .TRUE. = The wall boundary solution is the exact wall boundary condition, e.g., $[u_{\text{wall}}, v_{\text{wall}}, w_{\text{wall}}] = 0$ for a no-slip wall. .FALSE. = The average of the wall boundary solution and the interior solution on the boundary face recovers the exact wall boundary condition, e.g., $[u_{\text{wall}}, v_{\text{wall}}, w_{\text{wall}}] = -[u_{\text{wall}}, v_{\text{wall}}, w_{\text{wall}}]$ for a no-slip wall. **NOTE:** Hartmann (2014) reports that using exact wall boundary conditions seems to be more accurate on coarse grids and for lower values of *solution_order*, whereas using the averaging method seems to be a little more stable. Details can be found at `http://elib.dlr.de/90967/1/hartmann_leicht_VKI_LS_2014-3.pdf`
sub_inflow_method	integer	1	advanced	Specifies which subsonic inflow boundary condition algorithm to use. 1 = Hold the total pressure and total temperature constant at the inflow, use the outgoing characteristic from the interior to perform a Newton iteration to get the static temperature, speed of sound, and normal velocity magnitude for the exterior state. 2 = Use the inflow static density and velocity, and the interior static pressure. 3 = Hold the inflow total pressure and total temperature constant, and use the interior velocity. 4 = Hold the inflow total pressure and total temperature constant, and use the interior static pressure. **NOTE:** It is highly recommended to use option 1.
use_bc_conflicts	integer	.TRUE.	special case	Logical flag for determining whether or not to overwrite boundary solutions in adjacent cells if co-located solution points have different boundary conditions. **NOTE:** This is only relevant if using Legendre-Gauss-Lobatto points for solution points.
cgns_use_queue	integer	.FALSE.	developer	Flag enabling/disabling the CGNS queuing system when writing parallel CGNS solution files. **NOTE:** This is only applicable if using a version of the CGNS library older than 3.3.0, e.g., 3.2.1.
interpolate_before_output_variable	integer	.TRUE.	developer	Flag determining whether the solution variables are computed before or after interpolating to the output points.
convert_restart_to_cgns	integer	.FALSE.	utility	Flag used to read in a restart file and immediately write a CGNS solution file, bypassing the solver entirely.

completely_disable_cgns	integer	.FALSE.	advanced	Flag used to completely disable CGNS solutions from being output at the output interval.
convergence_order_abs_res	integer	-16.0	advanced	This is the convergence goal for the orders of magnitude of the absolute residual.
convergence_order_max_res	integer	-16.0	advanced	This is the convergence goal for the orders of magnitude reduction of the residual relative to the maximum value of the residual during the simulation.
profile_io_all	integer	.FALSE.	developer	If true, sets both *profile_io_cgns* and *profile_io_restart* to true.
profile_io_cgns	integer	.FALSE.	developer	Flag to activate some simple timers that time how long it takes to write each CGNS solution file, and output the results to standard output.
profile_io_restart	integer	.FALSE.	developer	Flag to activate some simple timers that time how long it takes to write each restart or time-averaged restart file, and output the results to standard output.
output_bnd_profiles	integer	.FALSE.	experimental	Flag used to output the solution profiles on certain boundary faces (highly experimental).
use_unrolled_dot_products	integer	.TRUE.	developer	Flag for enabling/disabling the manually unrolled dot product functions.
lustre_stripe_count	integer	64	expert	Specifies the approximate number of stripes to use when writing a CGNS solution file or a restart file. **NOTE:** This is only used if *output_dir* is on a Lustre file system. **NOTE:** The code will take this value and find the closest integer value satisfying: $\mathrm{mod}\,(ncpu, lustre\_stripe\_count) == 0$
lustre_stripe_size	character(len=10)	"4m"	expert	Specifies the number of bytes to store on each stripe before moving to the next stripe. **NOTE:** This is only used if *output_dir* is on a Lustre file system. **NOTE:** The value of *lustre_stripe_size* must be a multiple of 65,536 bytes (64 KB), and the suffixes 'k', 'm', or 'g' can be used to specify units of KB, MB, or GB, respectively. **EXAMPLE:** "4m" $\Rightarrow$ 4 MB $\Rightarrow 4,194,304$ bytes
output_time_averaging	integer	.FALSE.	basic	Logical flag that enables/disables the accumulation of time-averaged flow variables. If enabled, time-averaged restart and CGNS solution files will be written whenever the standard restart and CGNS solution files are written. **NOTE:** In order to save disk space, the name of the time-averaged restart file will switch between "*output_dir*/Restart_files/out.1.ave" and "*output_dir*/Restart_files/out.2.ave". After successfully writing a new time-averaged restart file, a symbolic link "*output_dir*/Restart_files/out.current.ave" will be created that points to the latest time-averaged restart file.

time_ave_file	character(len=150)	“out.ave”	basic	File path to the time-averaged solution restart file that will be read in if *load_restart_file* and *output_time_averaging* are both true. The code will add the accumulation of time-averaged flow variables from the new running simulation to the time-averaged flow variables of the previous simulation from which the new simulation restarted.
time_scaling_factor	integer	1.0	basic	This is a scaling factor that is used to scale the value of the running time within the simulation that is written to standard output and the results file. This scaling has no affect on a simulation besides the cosmetic scaling of these values when they are output.
time_ave_vel_is_axisymm	integer	.FALSE.	advanced	Logical flag that determines what velocity components to compute for the time-averaged flow variables. If true, the grid is assumed to be axisymmetric about the x-axis and the time-averaged velocity components are: the velocity in the streamwise direction (v_x), the velocity in the radial direction (v_r), and the azimuthal velocity (v_θ). If false, the Cartesian coordinate velocities (v_x, v_y, v_z) are the time-averaged velocity components.
time_average_restart_files	integer	.FALSE.	utility	Logical flag to run a utility that creates a time-averaged solution from a list of existing restart files given in the file specified by the *restart_list_file* input parameter. This utility will finish by writing out a time-averaged restart file and a time-averaged CGNS solution file. This utility does not execute anything within the solver.
restart_list_file	character(len=150)	“ ”	utility	File path to a simple ASCII file that contains a list of file paths to existing restart files that will be used to create a time-averaged solution. This is only used if the input flag *time_average_restart_files* is true.
dump_memory_usage	integer	.FALSE.	developer	Flag used to profile the memory usage of GFR.
dump_fluxes	integer	.FALSE.	developer	Flag used to output the fluxes computed in the solver.
check_flux_point_coordinates	integer	.FALSE.	developer	Flag used to output some files to check that co-located face quantities are consistent in terms of interpolation between two cells sharing a face.
postproc_debug_grid	integer	.FALSE.	developer	Flag used to activate using an internally created grid that is useful for debugging the post-processor used for the *continuous_output* input flag.
output_plot3d_to_tecplot	integer	.FALSE.	utility	Flag to output a Tecplot file of the input Plot3D grid file after reading it.
output_bface_array	integer	.FALSE.	developer	Flag to have each MPI process output the contents of its own local bface array to the text file “bface.[MPI process #].dat”. The bface array contains the boundary conditions and connectivity information for each boundary face.

bc_input(1:20)	bc_input_t		basic	Derived type to specify boundary conditions. An iterative loop will go through the specified flow conditions and try to compute the remaining unspecified flow conditions. **NOTE:** A run-time error will occur if either of the following conditions are true: A) The flow conditions are over-specified and inconsistent B) The flow conditions are under-specified and it is unable to compute the remaining unspecified flow conditions. **EXAMPLE:** An example of a well defined boundary condition using bc_input: bc_input(1)%name = "INFLOWSUBSONIC" bc_input(1)%set_bc_value = 2 ! Subsonic Inflow BC bc_input(1)%p_static = 101325.0 bc_input(1)%p_total = 121286.025 bc_input(1)%t_total = 293.15
bc_input(1)%name	character(len=32)	" "	basic	Character string that matches the name of a boundary group in the grid file. **NOTE:** Case (upper/lower-case) does NOT matter.
bc_input(1)%set_bc_value	integer	-9999	basic	This overrides the boundary condition type specified in the grid file, using this boundary condition type for the boundary group given by *bc_input(1)%name*. Valid values for boundary condition types are: 1 = Generic inflow, subsonic or supersonic 2 = Subsonic inflow 3 = Supersonic inflow 11 = Generic outflow, subsonic or supersonic 12 = subsonic outflow 13 = supersonic outflow 21 = Characteristic inflow/outflow based on incoming/outgoing Riemann invariants, can be subsonic or supersonic 22 = Generic inflow/outflow, subsonic or supersonic 23 = Fixed to freestream/reference flow conditions 24 = Fixed flow conditions (equivalent to supersonic inflow) 50 = Slip wall 52 = Adiabatic no-slip wall 53 = Isothermal no-slip wall 80 = Symmetry boundary condition (equivalent to slip wall)
bc_input(1)%t_static	integer	-1.0	basic	Static temperature (units of Kelvin)
bc_input(1)%p_static	integer	-1.0	basic	Static pressure (units of Pa)

bc_input(1)%rho_static	integer	-1.0	basic	Static density (units of $\frac{\text{kg}}{\text{m}^3}$)
bc_input(1)%mach	integer	-1.0	basic	Mach number (dimensionless)
bc_input(1)%vx	integer	0.0	basic	Velocity in the x coordinate direction (units of m/s)
bc_input(1)%vy	integer	0.0	basic	Velocity in the y coordinate direction (units of m/s)
bc_input(1)%vz	integer	0.0	basic	Velocity in the z coordinate direction (units of m/s)
bc_input(1)%t_total	integer	-1.0	basic	Total temperature (units of Kelvin)
bc_input(1)%p_total	integer	-1.0	basic	Total pressure (units of Pa)
bc_input(1)%rho_total	integer	-1.0	basic	Total density (units of $\frac{\text{kg}}{\text{m}^3}$)
bc_input(1)%alpha_aoa	integer	0.0	basic	Angle of attack with respect to the x-y plane (dimensionless)
bc_input(1)%beta_aoa	integer	0.0	basic	Angle of attack with respect to the x-z plane (dimensionless)
bc_input(1)%wall_temp	integer	-1.0	basic	Temperature for isothermal wall (units of Kelvin)
bc_input(1)%relax%value	integer	1.0	experimental	Relaxation value to limit the difference between the interior and exterior states ($0.0 <$ bc_input(1)%relax%value ≤ 1.0)
bc_input(1)%relax%time	integer	0.0	experimental	Solution time to end relaxation (> 0.0)
bc_input(1)%relax%iter	integer	0	experimental	Time step to end relaxation (> 0)