TAU-Code User Guide

Release 2010.1.0

March 26, 2010

Contents

The TAU-Code

The main programs of the TAU-Code are the *preprocessing*, *flow solver* and *adaptation* modules. To start a computation, the *preprocessing module* needs the following inputs:

- Primary grid (NetCDF format)
- Parameter file (ASCII format)

The preprocessing program computes the dual-grid data which is written to one or several files in NetCDF format, depending on the number of coarse grids required for multigrid calculations and the number of domains the grid file is split into for parallel calculations.

The *flow solver* computes the solution using the following inputs:

- Dual grid(s) (NetCDF format)
- Parameter file (ASCII format)

The solution is written in one (or several when used in parallel mode) result file(s), again in NetCDF format.

The *adaptation module* writes out a refined primary grid (NetCDF format), a second file containing information about non-isotropically refined elements (called critical elements) and a new result file (NetCDF format) derived by interpolating the old solution. The required inputs are:

- Primary grid (NetCDF format)
- Solution file (NetCDF format)
- Parameter file (ASCII format)

The preprocessing program is used again to compute the new dual grid data and together with the new solution, these are used to restart the solver. This adaptation cycle can be repeated several times.

If the solver is used in parallel mode, one solution file is written per domain, which is needed when making parallel restart computations. If, however, a single domain restart computation is wanted, or for postprocessing purposes (analysis or visualization) as well as for starting an adaptation cycle, the decomposed solution has to be gathered into a single result file using the program *gather-scatter*. Following postprocessing this program can also scatter a solution into several result files in order to restart the solver in parallel mode. Other utility-programs available in the distribution are explained later on in Section Utility programs (page 310).

1 Compilation of the TAU-Code

In case of having sources the TAU-Code is distributed with a top-level Makefile which can be found in the directory 'taudir'. In this directory a 'README'-file and an 'INSTALL'-file containing more information can also be found. Before compilation, adjustments at the top of the Makefile have to be done (e.g. definition of paths, the current architecture, etc.), which are explained in more detail in the leading comments of the Makefile.

The following `make` commands are the most useful:

- `make all` compiles all the sources contained in the distribution.
- `make help` displays all supported commands, which enable the user to compile program by program.
- `make taulib` compiles all the libraries, which need to be built first when compiling program by program.
- `make taupy` creates all TAU-Python bindings for using the python interface of TAU-Code.
- `make allclean` removes all object files and all binaries previously compiled.

2 General aspects of usage

The usage of all programs described here is similar. When executing the binary without additional parameters in the command-line, the syntax for using the program is printed to `stdout`. In addition the version-date and the compile-time date is printed. The common syntax for using of the preprocessing, flow solver and adaptation programs is:

```
binary-name parameter-file <log-file>
```

Optionally, a `log-file` name can be given as third argument. In this case the process information is written to the `log-file` instead to stdout.

For running TAU-Code-Modules in parallel (currently possible e.g. for the preprocessing, the solver, the adaptation and the mesh deformation) the mpirun script has to be used, e.g:

```
mpirun -np # bin-path/binary-name parameter-file <log-file>
```

The MPI options and the name of the mpirun script depend on the MPI version. Please refer to the manuals of the MPI version installed on the target machine.

The internal switch in the TAU-Code for the use of MPI is based on the number of command line arguments (MPI usage only if more than 3 command line arguments are given. Some

MPI-installations (e.g. mpich) add extra arguments to the command line, others do not (e.g. NEC mpisx). Using such an installation (which becomes obviously when the usage described above does not work) requires to add extra arguments to the command line. Then, the usage is

```
mpirun -np # bin-path/binary-name parameter-file log-file use_mpi
```

3 Parameter file

A parameter file is required for running all modules of the TAU-Code, hence it is important to understand how to build it. It is an ASCII-file which contains parameter names, called keywords, and parameter values. The parameters are read by the keywords, thus the order of the parameters can be chosen by the user. Unknown strings that are not recognized as keywords are not considered. This allows all global parameters needed for all programs to be written in one file, which is of advantage because some of the parameters are used by more than one program. Also, additional notes or comments can be written in the parameter file.

3.1 Starting from scratch

In order to ease the set up of the required input files when starting with a new configuration (i.e. the parameter file, the boundary mapping and eventually other input) sample input parameter files are provided. These sample files can be found in the directory `taudir/input`. Several sample files are available, as `para_short, para_long` and perhaps others in future.

These files contain the most useful parameters. Most of them are set to recommended values. Some of them, which are marked, need to be adapted, because no general defaults are available.

Thus, the simplest way to start is to copy e.g. `taudir/input/para_short` to the working directory, and modify it where necessary.

In addition to this User Guide, a tool named TauPDB (page 356) is available. It provides the ability to query the detailed information as described in the User Guide online. A full-text search, a search by parameter-name and a regular expression search, allowing the combination of several keywords, help finding the required parameter in less time.

3.2 Parameter details

Most of the parameters have a default value (see Section 4), which is used automatically if the parameter keywords are not found in the parameter file. If a parameter without default value is missing the program stops and all keywords and parameters are printed to stdout; the missing parameters are marked. This allows a new parameter file to be built by starting the code with an empty parameter file. The parameters can then be copied from stdout and pasted into the file: the parameter values without default have to be added, the default parameters can be adopted.

The parameters are read line by line. Each input line is split by a ':' into two parts. The left side consists of white spaces and the keywords, the right part provides the parameter value. Additional text or numbers at the end of the line are not considered. Typical parameter values are strings with and without blanks, floats and integers. A few parameters may be arrays, i.e. their keywords can be followed by several values. A special type (list of integers)

is used to read the boundary markers. Note, the length of a line is limited to 1024 characters. Longer lines can not be handled and even may lead to errors due to implementation details.

There is one danger: a misspelled keyword (an additional blank inside the keyword is also a mistake) of a parameter with a non-default value leads to the use of the default!

To achieve uniformity between parameter files, keywords should obey the following rules:

- A keyword starts with a capital letter. The following words start with small letters, except when a capital is required for the normal spelling of the word (e.g. Reynolds).
- Keywords belong to the English language (checked, for example, by ispell).
- Clear restrictions on the allowed range of values should be given in brackets as part of the keyword (i.e. (0/1)).
- The parts of a keyword like words or a range in brackets are separated by exactly one space in between.

Another important point is that each time a keyword is found it's corresponding value is assigned to that parameter. The last entry (if a keyword is found several times) overwrites previous ones. This feature is used for automatically updating parameters by writing the updated values or strings with the corresponding keywords at the end of the file. For example, if this feature is switched on, the solver updates the parameter file each time a solution file is written out. Thus, when restarting the solver the most recent result file is used. When several result files have been written out, the complete 'file-history' can be found at the end of the parameter file. Keywords at the end of the parameter file can be ignored using the word 'end' (without any other character in the same line), as this stops the search for any keywords below.

All parameters are printed to stdout when starting the program. This allows the user to check the correctness of the input.

3.3 Comments

In addition to the keywords used as parameters the user can add comments into the parameter-file. If a line in the parameter file starts with # it is considered and stored as a comment-line. Comments are also written into the header of the solution files. This allows the user to add a description of the problem to the final data. This description can be read from the solution files using the NetCDF-utility 'ncdump -h'.

4 Input parameter overview

This section summarizes all the valid parameters for the preprocessing, solver, deformation and adaptation modules of the TAU-Code . The parameters are grouped into the following categories:

- those used to control the running and various features of the code (General parameters - see below)
- those needed for the boundary conditions (Boundary mapping parameters - see page 127).
- those needed to define transition (Transition parameters - see page 142)

Note: the user guide contains some few parameters which are not active (not visible and not accessible) in a released version of the TAU-Code . These are parameters for functionalities in the development stage (implemented but not sufficiently tested).

4.1 General parameters

The general parameters used in the TAU-Code are listed in the table below. The parameters (1st column) which have to be defined at least in one of the modules (they have no default value) are highlighted in the table by **bold type** or underlined in the description section. The keywords are in alphabetical order. The 2nd column details the type of the parameter value, whether a string, an integer, a float or an array of integers or floats is expected. If there are recommended limits for a parameter these can be found in the column 'range'. Each module of the TAU-Code which uses the parameter is indicated in the column 'in' by: 'p' for the preprocessing, 's' for the solver, 'a' for the adaptation and 'd' for the deformation. A description of each parameter can be found in the subsection 'Input Parameters'.

parameter name	type	range	default	in	page
2D offset vector (0 / x=1,y=2,z=3)	int	$0 \leq i \leq 3$	0	p, a, d	39
2nd order dissipation coefficient	float	$0 < x$	0.5	s	39
Accumulate queue time (0/1)	int	0, 1	1	s	39
Adaptation sensitivity	float	$0 < x < 1$	0.9	a	39
Adjoint-adapt-data	string	-	(none)	a	39
Aerodynamic sweep angle	float	-	0.0	s	40
Agglomeration version	int	0, 1	0	p	40
Amplitude description filename	string	-	(none)	d	40

continued on next page

parameter name	type	range	default	in	page
Amplitude filename	string	-	(none)	d	40
Analysis incompr./compressible (0/1) [ICOMP]	int	0, 1	0	s	40
Apply Chimera wall projection (0/1)	int	0, 1	0	s	40
Assigned transition block	int	-	-1	s	40
Ausm scheme dissipation	float	$0 < x$	0.25	s	40
AUSMDV shock fix (0/1)	int	0, 1	0	s	40
AUSMP alpha	float	$-0.75 \leq x \leq 3/16$	0.1875	s	41
AUSMP beta	float	$-1/16 \leq x \leq 1/2$	0.125	s	41
AUSMP pressure weight	int	0, 1, 2	1	s	41
Automatic parameter update (0/1)	int	0, 1	0	s, a, d	41
Automatic parameter update mode (0/1)	int	0, 1	0	s, a, d	41
Axisymmetry (0/1)	int	0, 1	0	p	41
Axisymmetry axial direction	array-float	-	{1, 0, 0}	p	41
Axisymmetry radial direction	array-float	-	{0, 0, 1}	p	42
Bandwidth optimisation (0/1)	int	0, 1	1	p	42
BL edge criterion	int	0 - 3	3	s	42
Block	int	≥ 0	0	s	42
Boundary layer data mode	int	0, 1, 2, 3	0	s	42
Boundary mapping filename	string	-	not_defined	s, p, a, d	43
Boundary part list	string	-	-	s	43
Boundary part list for prescription	string	-	-	s	43
Boundary point (0/1)	array-int	0	0	s	43
Bypass transition prediction mode	int	0 - 4	0	s	43

continued on next page

parameter name	type	range	default	in	page
Cache-coloring (0/max_faces in color)	int	$0 \leq$ i	0	p	43
Central convective meanflow flux	string	Average_of_flux, Flux_of_average	Average_of_flux	s	44
Central convective turbulence flux	string	Average_of_flux, Flux_of_average, AUSMDV, Roe, Roe2nd	{Average_of_flux, Roe}	s	44
Central dissipation scheme	string	Scalar_dissipation, Matrix_dissipation	Scalar_dissipation	s	45
Cf-min offset for extrapolation mode 3	int	-	10	s	45
CF transition prediction mode	int	0 - 10	10	s	45
CFL number	float	$0 <$ x	1.8	s	45
CFL number (coarse grids)	float	$0 <$ x	1.8	s	45
CFL number (large grad p)	float	$0 <$ x $\leq$ CFL	1.8	s	45
Change boundary control volumes (0/1)	int	0, 1	0	p	45
Change wall normal distribution (0/1)	int	0, 1	1	a	46
Coarse grid upwind flux	string	-	Van_Leer	s	46
Coarse grid viscous flux type TSL/Full (0/1)	int	0, 1	0	s	46
Coco executable	string	-	./coco	s	46
Compressible mixing layer correction (0/1)	int	0, 1	0	s	46
Compute exact whirlflux (0/1)	int	0, 1	0	s, p	46
Compute flow statistics	string	(none), mean, variance, meanvelgrad	(none)	s	47
Compute harmonics of global forces (0/1..n)	int	-	0	s	47

continued on next page

parameter name	type	range	default	in	page
Compute harmonics on surface (0/1)	int	0, 1	0	s	48
Compute lusgs mapping (0/1)	int	0, 1	0	p	48
Compute parent faces (0/1)	int	0, 1	0	p	48
Compute passive sas correction	int	0, 1	0	s	48
Compute point-face connectivity (0/1)	int	0, 1	1	p	48
Compute statistics (0/1)	int	0, 1	0	s	48
Consider wall roughness	string	global, constant, pointwise	global	p	49
Constant wall roughness	float	$0 \leq$ x	0.	p	49
Contourline cut extraction mode	int	-	1	s	49
Contourline plane normal definition	string	-	xz	s	49
Control volume edge weight (0/1)	int	0, 1	1	p	49
Correct metric for boundary control volumes (0/1)	int	0, 1	1	p	50
Correction smooth epsilon	float	$0 \leq$ x	1.8	s	50
Correction smoother	string	-	Point_explicit	s	50
Correction smoothing steps	int	$0 \leq$ i	2	s	50
Cost function	string	Point-pressure, C-lift, C-drag, C-sidef, C-mx, C-my, C-mz, Fx, Fy, Fz	C-drag	s	50
Cost function part total/pressure/viscous (0/1/2)	int	0, 1, 2	0	s	50
Couple mean flow and turbulence equations (0/1)	int	0, 1	0	s	51
Critical N-factor CF	float	$0 <$ x	9.0	s	51
Critical N-factor TS	float	$0 <$ x	12.0	s	51
Cut-off value	float		1.0	s	51

continued on next page

parameter name	type	range	default	in	page
Deformation description filename	string	-	(none)	d	51
Deformation reduction factor	float	-	1	d	51
Deformed coordinates filename	string	-	(none)	d	51
Delta-Z at outer-Y	float	-	0	d	52
Delta frequency for task 30 loop	float	-	150.0	s	52
DES model	string	SA, XLES, SST	{SA, XLES}	s	52
Design variable	string	Farfield-density, Farfield-mach-number, Farfield-alpha, Farfield-beta	Farfield-alpha	s	52
Displacement scale	float	$0 < x$	0.	d	52
EARSM expansion order	int	1, 2, 3, 4	1	s	53
Edge relation to unstructured part	float	$0 < x$	0	a	53
Effective sweep angle	float	-	0.0	s	53
Error for Cauchy convergence cdrag	float	-1.0, $0 < x$	-1.0	s	53
Error for Cauchy convergence clift	float	-1.0, $0 < x$	-1.0	s	54
Error for Cauchy convergence cmy	float	-1.0, $0 < x$	-1.0	s	54
Error for Cauchy convergence cost function	float	$0 < x$	1.0e-6	s	54
Exclude cut block	array-int	≥ 0	-	s	54
Extended coefficient monitoring (0/1)	int	0, 1	0	s	55
Extended motion monitoring (0/1)	int	0, 1	0	s	55
Factor for dynamic Cauchy convergence	float	$0.0 < x < 1.0$	0.05	s	55

continued on next page

parameter name	type	range	default	in	page
Fd switch uses SA viscosity (0/1)	int	0, 1	0	s	55
Field output description file	string	-	(none)	s	55
Field output values	string	-	(none)	s	56
Filter width model	string	Edge, Volume, User	Edge	s	56
Final freeform box filename	string	-	(none)	d	56
Findiff block output format (0/1/2)	int	0, 1, 2	0	s	56
Findiff column global	int	0-npoints	0	s	56
Findiff consider diffflux	int	0, 1	1	s	56
Findiff consider invflux	int	0, 1	1	s	56
Findiff consider turbsource	int	0, 1	1	s	56
Findiff consider turbviscflux	int	0, 1	1	s	56
Findiff consider viscflux	int	0, 1	1	s	57
Findiff epsilon	float	1e-1 - 1e-16	1.e-8	s	57
Findiff maximum epsilon	float	1e-1 - 1e-16	1.e-2	s	57
Findiff minimum epsilon	float	1e-1 - 1e-16	1.e-14	s	57
Findiff row for convergence output	int	0-npoints	-1	s	57
First wave number [ALPHA]	float	-	0.3	s	57
Fix chimera boundaries (0/1)	int	0, 1	1	d	57
Fix negative values (0/1)	int	0, 1	0	s	57
Fixed RANS distance	float	-infinity, infinity	-1	s	58
Flow direction for sharp edges	array-float	-	{1, 0, 0}	s	58
FLOWer vortical correction coefficient	float	$0 < x$	4.0	s	58
Flux weighting scheme	string	-	AUSMDV	s	58
Form of scaling denominator	string	Shur, Smirnov	Shur	s	58
Freqdom RHS file	string	-	(none)	s	58
Frequency of instability wave in Hz (CF) [FRQ]	float	-	500.0	s	59

continued on next page

parameter name	type	range	default	in	page
Frustum max radius	array-float	-	0	a	59
Frustum min radius	array-float	-	0	a	59
Full multigrid central scheme first-order (0/1)	int	0, 1	1	s	59
Gas constant gamma	float	$0 < x$	1.4	s	59
Gas constant R	float	$0 < x$	287	s	59
General ratio mue-t/mue-l	float	$0 < x$	0.1	s	59
General turbulent intensity	float	$0 < x$	0.001	s	60
Geometric conservation law (0/1)	int	0, 1	1	s	60
Global wall roughness	float	$0 \leq x$	0	p	60
GMRes inner iterations	int	2-100	20	s	60
GMRes preconditioning iterations	int	1-100	10	s	60
Grid metric	string	Cell_Vertex, Cell_Centered	Cell_Vertex	p	60
Grid prefix	string	-	dual	s, p	60
Grid scale	float	$0 < x$	not_defined	s	61
Grid shield model	string	(none), DES, DDES, IDDES	(none)	s	61
h-scaling power	float	-	0.5	a	61
h-scaling reference length	float	-	0	a	61
Half span reference length	float	-	0.0	s	61
Hardwired weighting factor	float	$0 \leq x \leq 1$	-1	s	61
Hellsten vortical correction coefficient	float	$0 < x$	3.6	s	61
Hold static velocity field (0/1)	int	0, 1	0	s	61
Hole filename	string	-	(none)	s	62
Hybrid switch model	string	(none), Edge, Volume, User	(none)	s	62

continued on next page

parameter name	type	range	default	in	page
Implicit overrelaxation beta	float	$0.5 \leq x \leq 2.0$	1.0	s	62
Implicit overrelaxation omega	float	$0.7 \leq x \leq 2.0$	1.0	s	62
Increase memory (0/1)	int	0, 1	0	s	62
Indicator0 Ht-scaling	float	$0 \leq x$	1	a	62
Indicator0 Pt-scaling	float	$0 \leq x$	1	a	63
Indicator0 rho-scaling	float	$0 \leq x$	1	a	63
Indicator0 V-scaling	float	$0 \leq x$	1	a	63
Indicator type	string	diff, grad, recon, sum, all, del, vortex, adjoint	diff	a	63
Indicator user-scaling	array-float	$0 \leq x$	1.0	a	63
Indicator user-values	string	-	(none)	a	63
Init total conditions (0/1)	int	0, 1	0	s	63
Initial freeform box filename	string	-	(none)	d	63
Initialize deformation (0/1)	int	0, 1	0	s	64
Interpolate prescription values (0/1)	int	0, 1	1	s	64
Inverse 4th order dissipation coefficient	float	$0 < x$	64	s	64
Inviscid flux discretization type	string	Central, Upwind	Upwind	s	64
Jacobian constant dissipation coeffs (0/1)	int	0, 1	1	s	65
Jacobian constant laminar viscosity (0/1)	int	0, 1	1	s	65
Jacobian frozen turbulence (0/1)	int	0, 1	1	s	65
Jacobian includes timestep (0/1)	int	0, 1	0	s	65
Jacobian turb. includes timestep (0/1)	int	0, 1	0	s	65
Jacobian variables	string	Cons, Prim	Cons	s	65
Jacobian volume scaling (0/1)	int	0, 1	0	s	66

continued on next page

parameter name	type	range	default	in	page
K-omega limitation type	string	standard, Rudnik, Schwarz	standard	s	66
K-omega wall factor	float	0, 10	10	s	66
Kato Launder modification factor	float	$0 \leq x \leq 1$	0	s	66
Keep Coco auxiliary files (0/1)	int	0, 1	1	s	67
Keep Coco log files (0/1)	int	0, 1	1	s	67
Keep Coco profiles files (0/1)	int	0, 1	0	s	67
Keep Coco run files (0/1)	int	0, 1	0	s	67
Keep files from pre-mode (0/1)	int	0, 1	0	s	67
Keep Lilo auxiliary files (0/1)	int	0, 1	1	s	67
Keep Lilo log files (0/1)	int	0, 1	1	s	67
Keep Lilo run files (0/1)	int	0, 1	0	s	67
Keep N-factor files (0/1)	int	-	1	s	67
Keep Prepcp files (0/1)	int	0, 1	1	s	67
Kozlov modification	int	0, 1	0	s	67
Krylov loop	string	(none), GMRes, RPM, EvalsDirect, PETSc	(none)	s	68
Leading edge sweep angle	float	-	0	s	68
Lilo executable	string	-	./lilo	s	68
Lim. angle to detect sharp edges (deg)	float	$0 \leq x \leq 360$	30	s	68
Limiter freezing convergence	float	-	0	s	68
Linear residual type	string	Facemat, PETSc	Facemat	s	68
Linear restart-data prefix	string	-	(none)	s	69
Linear solver	string	-	Lusgs	s	69
Low Re model	string	(none), NTS, Breuer	NTS	s	69
Lowest pressure for 2nd order	float	$0 \leq x$	0	s	69
Lusgs increased parallel communication (0/1)	int	0, 1	0	s	69

continued on next page

parameter name	type	range	default	in	page
Lusgs treat whirl implicitly (0/1)	int	0, 1	0	s	69
Mach number limit for limiter	float	-	0	s	69
Matrix dissipation terms coefficient	float	$0 < x$	1.0	s	70
Max. distance for wallnormals	float	-	1e+20	s	70
Max. number of points on wallnormal	int	-	128	s	70
Maxima x-direction	array-float	-	0	a	70
Maxima y-direction	array-float	-	0	a	70
Maxima z-direction	array-float	-	0	a	70
Maximal point movement factor	float	$0 < x$	0.2	a	71
Maximal time step number	int	-	not_defined	s	71
Maximal time step number (coarse grids)	int	$0 < i$	1	s	71
Maximum delta cp for pmin/pmax search	float	-	0.5	s	71
Maximum delta for transition	float	-	1e+20	s	71
Maximum limit mue-t/mue-l	float	$0 < x$	20000	s	71
Maximum point number	int	$-1 \leq i$	-1	a	71
Maximum point number per partition	int	$-1 \leq i$	-1	a	71
Maximum refinement level	int	$-1 \leq i$	-1	a	72
Maximum surface angle (degree)	float	$30 < x < 180$	30	a	72
Maximum turbulence production/destruction	float	$1 < x$	20	s	72
Maxsize for the coarse graph for MGridGen	int	$1 < i$	4	p	72
Metis parameter CoarsenTo (int)	int	$0 < i$	100	p	72
Metis parameter IPtype (int)	int	see Metis doc	4	p	73

continued on next page

parameter name	type	range	default	in	page
Metis parameter Mtype (int)	int	see Metis doc	21	p	73
Metis parameter Rtype (int)	int	see Metis doc	13	p	73
MG description filename	string	-	4w	s	73
Min. number of points on wall-normal	int	-	32	s	73
Minima x-direction	array-float	-	0	a	73
Minima y-direction	array-float	-	0	a	74
Minima z-direction	array-float	-	0	a	74
Minimal density	float	$0 < x$	1e-12	s	74
Minimal energy	float	$0 < x$	1e-12	s	74
Minimal pressure	float	$0 < x$	1e-12	s	74
Minimum artificial dissipation for acoustic waves	float	$0 < x < 1$	0.2	s	74
Minimum artificial dissipation for velocity	float	$0 < x < 1$	0.2	s	74
Minimum edge length	float	$0 < x$	1.e-12	a	75
Minimum k	float	$0 < x$	1e-05	s	75
Minimum N-factor for extrapolation (CF)	float	-	0.001	s	75
Minimum N-factor for extrapolation (TS)	float	-	0.001	s	75
Minimum number of inner iterations per time step	int	$0 < i$	0	s	75
Minimum omega	float	$0 < x$	1e-05	s	75
Minimum residual	float	$0 < x$	1e-16	s	75
Minimum residual (coarse grids)	float	$0 < x$	0.001	s	75
Minsize for the coarse graph for MGridGen	int	$0 < i$	1	p	76
Mismatch of overlapping walls	float	$0 < x$	0.0	s	76

continued on next page

parameter name	type	range	default	in	page
Mixed inviscid fluxes (0/1)	int	0, 1	0	s	76
Modal amplitude factor	array-float	-	1	d	76
Modal perform maximum deflection (0/1)	int	0, 1	0	d	76
Modal reduced frequency	array-float	-	0.0	d	76
Modal reference length	array-float	-	0.0	d	76
Modal steps per period	int	-	0	d	76
Modify cp for Coco input	int	0 - 2	1	s	77
Monitor history (0/1)	int	0, 1	0	s	77
Monitoring significant figures	string	-	4_4_4_4	s	77
Monitoring values	string	-	(none)	s	77
Motion hierarchy filename	string	-	(none)	s	77
Multigrid indicator (0/1)	int	0, 1	0	s	78
Multigrid start level	int	$0 < i$	1	s	78
N-factor extrapolation mode	int	0 - 13	2	s	78
N-ts/N-cf Diagram (N-CF)	array-float	$0 < x$	-	s	78
N-ts/N-cf Diagram (N-TS)	array-float	$0 < x$	-	s	78
N-ts/N-cf Diagram (points)	int	$0, 2 < i$	0	s	79
Neglect 2/3 rho k term in k and omega production (0/1)	int	0, 1	1	s	79
New primary grid prefix	string	-	(none)	a, d	79
New restart-data prefix	string	-	(none)	a	79
NLR vortical correction - neglect 2/3 rho k term in omega production (0/1)	int	0.1	0	s	79
NLR vortical correction coefficient	float	positive integer	1.0	s	79

continued on next page

parameter name	type	range	default	in	page
Number of cut-out volumes	int	-	0	a	79
Number of domains	int	$0 < i$	1	p	80
Number of frequencies/wavelengths [NWAV]	int	0 - 40	40	s	80
Number of loops for task 30	int	-	3	s	80
Number of modes in file	int	-	0	d	80
Number of modes to be used	int	-	0	d	80
Number of multigrid levels	int	$0 < i$	5	p	80
Number of planes	int	$0 < i$	1	s	80
Number of points for global step (CF) [NPGLOB]	int	-	40	s	80
Number of points for global step (TS) [NPGLOB]	int	-	40	s	80
Number of points for local step (CF) [NPLOC]	int	-	100	s	80
Number of points for local step (TS) [NPLOC]	int	10 - 40	100	s	81
Number of prescription values	int	-	-	s	81
Number of primary grid domains	int	$0 < i$	1	p	81
Number of profiles	int	$0 \leq i$	0	s	81
Number of RANS cut-out boxes	int	0, 1, ...	0	s	81
Number of Runge-Kutta stages	int	$1 \leq i \leq 6$	3	s	81
Number of samples for Cauchy convergence	int	$10 < i \leq 1000$	20	s	81
Number of smoothing steps for sas correction	int	$0 \leq i$	0	s	82
Number of smoothing steps for vortical correction	int	$0 \leq i$	0	s	82
Number of stations in damped region [NDAMP]	int	-	5	s	82
Number of time steps per period	int	$0 < i$	50	s	82

continued on next page

parameter name	type	range	default	in	page
Offset for pmin criterion	float	-	0.0	s	82
Offset for polylines extrapolation	float	-	1.5	s	82
Old grid prefix	string	-	(none)	p	82
Omega boundary condition type	string	smooth_standard, smooth_Rudnik, smooth_Wilcox, rough_Wilcox, rough_DLR	smooth_standard	s	82
Order of additional equations (1-2)	float	1, 2	{1, 2}	s	83
Order of upwind flux (1-2)	float	1, 2	2	s	84
Origin	array-float	-	{0, 0, 0}	s	84
Origin coordinate x	float	-	0	s	84
Origin coordinate y	float	-	0	s	84
Origin coordinate z	float	-	0	s	84
Outer-Y for deformation	float	-	1	d	84
Output files prefix	string	-	not_defined	s	84
Output level	int	$0 \leq$ i	5	p, d	85
Output period	int	$0 <$ i	999999	s	85
Output shifted points grid (0/1)	int	0, 1	0	p	85
Part of low quality elements	float	$0.0 \leq$ p	0.0	a	85
Percentage of new points	float	$0 <$ x	40	a	85
PETSc options	string	-	(none)	s	86
Plane normal x	array-float	-	0	s	86
Plane normal y	array-float	-	1	s	86
Plane normal z	array-float	-	0	s	86
Plane output description file	string	-	(none)	s	86
Plane output period	int	$0 \leq$ i	all	s	86

continued on next page

parameter name	type	range	default	in	page
Plane output values	string	-	(none)	s	86
Plane support x	array-float	-	0	s	86
Plane support y	array-float	-	0	s	87
Plane support z	array-float	-	0	s	87
Point fusing reward	float	$1 < x$	1.2	p	87
Point of point pressure cost function	int	0-npoints	0	s	87
Points skipped near stagnation point	int	-	5	s	87
Positivity scheme	int		0	s	87
Prandtl number	float	$0 < x$	0.72	s	88
Pre-maximum delta for transition	float	$0 < x$	1.e20	s	88
Pre-prediction end iteration nr	int	$0 \leq i$	1000	s	88
Pre-prediction mode	int	1, 2(, 3)	2	s	88
Pre-prediction period	int	$0 \leq i$	20	s	88
Pre-prediction start iteration nr	int	$0 \leq i$	20	s	88
Pre-relaxation factor for transition	float	0.0, 1.0	1.0	s	88
Preconditioning	string	(none), PrimOld, PrimNew, Conservative	(none)	s	88
Prediction end iteration nr	int	$0 \leq i$	10500	s	89
Prediction info output level	int	0 - 4	3	s	89
Prediction iteration offset	int	-	0	s	89
Prediction period	int	$0 \leq i$	500	s	89
Prediction start iteration nr	int	$0 \leq i$	1000	s	89
Prepcp executable	string	-	./prepcp	s	89
Prepcp factor for number of stations	float	-	1.0	s	89

continued on next page

parameter name	type	range	default	in	page
Prepcp max. x/c on lower surface	float	0.0 - 1.0	1.0	s	89
Prepcp max. x/c on upper surface	float	0.0 - 1.0	1.0	s	90
Prepcp number of stations	int	-	-1	s	90
Preprocessing for incompressible solver (0/1)	int	0, 1	0	p	90
Prescription block name	string	-	(none)	s	90
Prescription info output level	int	0 - 4	1	s	90
Prescription input data structure	string	-	Block	s	90
Prescription values periodic (0/1)	int	0, 1	1	s	90
Primary grid filename	string	-	not_defined	p, a, d	90
Profile data file	string	-	-		90
Profile every n steps	int	$0 \leq$ i	0	s	90
Profile normal x	array-float	-	0	s	91
Profile normal y	array-float	$0 \leq$ x	≥ 0	s	91
Profile normal z	array-float	$0 \leq$ x	≥ 0	s	91
Profile output allowed (0/1)	int	0, 1	0	s	91
Profile output description file	string	-	-	s	91
Profile output period	int	$0 \leq$ i	≥ 0	s	91
Profile output values	string	see description	(none)		91
Profile range	array-float	$0 \leq$ x	0	s	91
Profile support x	array-float	-	0	s	91
Profile support y	array-float	-	0	s	91

continued on next page

parameter name	type	range	default	in	page
Profile support z	array-float	-	0	s	91
Project boundary control volumes coordinates (0/1)	int	0, 1	1	p	92
Project wall distance (0/1)	int	0, 1	0	p	92
RANS box support x	array-float	All real numbers	array of reals	s	92
RANS box support y	array-float	All real numbers	array of reals	s	92
RANS box support z	array-float	All real numbers	array of reals	s	92
Ratio of delta over h	float	>1	>1	s	92
Ratio Prandtl lam/turb	float	$0 < x$	0.8	s	93
RBF basis coordinates and deflections filename	string	-	(none)	d	93
RBF deflections reduction factor	float	-	1.0	d	93
RBF markers specifying group	list	$0 \leq i$	-	d	93
RBF matrix name	string	-	(none)	d	93
RBF maximum number of base points used	int	$i \geq 4$	1000	d	93
RBF name	string	volume-spline, cubic-spline, thin-plate-spline, thin-plate-spline-4	volume-spline	d	94
RBF number of groups	int	$0 \leq x$	0	d	94
RBF radius euclid	float	$x > 0.0$	1.0	d	94
RBF radius full weight	float	$x > 0.0$	1.0	d	94
RBF radius zero weight	float	$x > 0.0$	10.0	d	94
RBF surface format name	string	scattered, netcdf	(none)	d	94
RBF walldistances filename	string	-	(none)	d	94
Read partitioning filename	string	-	(none)	p	94
Reconstruction of gradients	string	Green_Gauss, Least_square	Green_Gauss	s	95

continued on next page

parameter name	type	range	default	in	page
Recover time step	float	$0 < x$	0	s	95
Reference bl-thickness	float	$0 \leq x$	1e+22	s	95
Reference density	float	$0 < x$	1.29251	s	95
Reference flow direction	array-float	-	{1, 0, 0}	s	95
Reference length (rolling/yawing momentum)	float	$0 < x$	1	s	96
Reference length (pitching momentum)	float	$0 < x$	1	s	96
Reference Mach number	float	$0 < x$	not_defined	s	96
Reference outer pressure	float	$0 \leq x$	reference state	s	96
Reference pressure	float	$0 < x$	101325	s	96
Reference relation area	float	$0 \leq x$	0	s	96
Reference system of forces and moments (tau/ln9300)	string	tau, ln9300	tau	s	97
Reference temperature	float	$0 < x$	273.15	s	97
Reference velocity	float	$0 < x$	-	s	97
Refinement mode	string	add, remove, both, none	add	a	97
Reinitialize flow averaging	array-int	0, 1	0	s	97
Relate period to start iteration (0/1)	int	0, 1	1	s	98
Relaxation factor for init station for task10	float	-	0.5	s	98
Relaxation factor for init station for task30	float	-	0.0	s	98
Relaxation factor for last station for task10	float	-	0.5	s	98
Relaxation factor for last station for task30	float	-	0.0	s	98

continued on next page

parameter name	type	range	default	in	page
Relaxation factor for modified cp	float	-	0.5	s	98
Relaxation factor for transition	float	$0 \leq x \leq 1$	0.8	s	98
Relaxation solver	string	Backward_Euler, Runge_Kutta	Runge_Kutta	s	98
Repair selection angle for hexas	float	$0 \leq x \leq 25$	3	d	98
Repair selection angle for prisms	float	$0 \leq x \leq 25$	3	d	99
Repair selection angle for pyras	float	$0 \leq x \leq 25$	0	d	99
Repair selection angle for tetras	float	$0 \leq x \leq 25$	3	d	99
Residual-data prefix	string	-	(none)	s	99
Residual monitoring type (0/1)	int	0, 1	1	s	99
Residual smooth epsilon	float	$0 < x$	1.8	s	99
Residual smoother	string	-	Point_explicit	s	99
Residual smoothing steps	int	$0 \leq i$	2	s	100
Restart-data prefix	string	-	(none)	s, a	100
Reynolds length	float	$0 < x$	1	s, a	100
Reynolds number	float	$0 < x$	not_defined	s	100
Roughness filename	string	-	(none)	p	100
RPM allow restarts (0/1)	int	0, 1	1	s	101
RPM Krylov acceptance ratio	float	1.0-10000.0	100.0	s	101
RPM maximum dimension of P	int	10-100	20	s	101
RPM maximum increase in dimension of P	int	2-10	2	s	101
RPM minimum iters before P update	int	1-100	4	s	101
RPM output eigenvectors (0/1)	int	0, 1	0	s	101
RPM preconditioning iterations	int	1-100	1	s	102
RPM verbose output (0/1)	int	0, 1	0	s	102
RSM DES constant	float	positive float	0	s	102
Rsm diffusion model	string	SGDH, GGDH	GGDH	s	102
Rsm dissipation model	string	isotropic	isotropic	s	102

continued on next page

parameter name	type	range	default	in	page
Rsm implementation version	string	FLOWer, Theory	FLOWer	s	102
Rsm length scale equation	string	Wilcox_omega, Kok_TNT_omega, Menter_BSL_omega, Hellsten_omega	Menter_BSL_omega	s	102
Rsm omega limiting factor	float	$0 \leq$ x	1.e-5	s	103
Rsm re-distribution model	string	Wilcox_RSM, SSG/LRR-w	SSG/LRR-w	s	103
Runge-Kutta coefficients	array-float	-	1	s	103
Runge-Kutta dissipation coefficients	array-float	-	{(1, 0, 0, 0, 0, 0)}	s	103
Runge Kutta steps for streamline integration	int	-	10	s	103
SA-DES constant	float	0.45 ... 0.65	0.65	s	104
SA attractor for zero value (0/1)	int	0.1	0	s	104
SA boundary condition type	string	smooth, rough, rough_mixed	smooth	s	104
SAMG matrix output prefix	string	filename prefix	(none)	s	104
SARC vortical correction coefficients	array-float	positive float	{(1, 12, 1)}	s	104
SAS flow correction model	string	(none), sst	(none)	s	105
Save only envelope of N-factors (0/1)	int	-	0	s	105
Scale for additional contourline cuts	float	-	0.25	s	105
Second wave number [BETA]	float	-	0.0	s	105
Selected-elements file	string	-	(none)	a	105
Set aerodynamic sweep angles	int	-	0	s	105
Set effective sweep angles	int	-	0	s	105
Set leading edge sweep angles	int	-	0	s	105
Set trailing edge sweep angles	int	-	0.0	s	106

continued on next page

parameter name	type	range	default	in	page
Set transition at solver start (0/1)	int	0, 1	0	s	106
Set transition info output level	int	0 - 2	1	s	106
Set zero time origin (0/1)	int	0, 1	0	s	106
SG start up steps (fine grid)	int	$0 \leq$ i	0	s	106
SGS Coefficient	float	positive integer	0.17	s	106
SGS model	string	(none), Smagorinsky, WALE	(none)	s	107
Sgs stages maximum	int	$0 <$ i	3	s	107
Sharp edge angle (degrees)	float	$0 <$ x ≤ 180	0	p	107
Smooth cp for Coco input (0/1)	int	0, 1	0	s	107
Smoothing eps for flux weighting	float	$0 <$ x	0.2	s	107
Smoothing eps for sas mode flow correction	float	$0 \leq$ x	0.2	s	107
Smoothing eps for vortical flow correction	float	$0 \leq$ x	0.2	s	108
Smoothing epsilon first	float	$0 \leq$ x	0.2	a	108
Smoothing epsilon for LES filter	float	positive real number	0.2	s	108
Smoothing epsilon last	float	$0 \leq$ x	0.2	a	108
Smoothing iterations for inner tetras	int	$0 <$ i	15	d	108
Smoothing iterations for LES filter	int	≥ 0	0	s	108
Smoothing relaxation factor	float	$0 <$ x	-1	d	108
Smoothing selection angle for tetras	float	$0 \leq$ x ≤ 90	15.	d	109
Smoothing steps first	int	$0 \leq$ i	5	a	109
Smoothing steps for flux weights	int	$0 \leq$ i	0	s	109
Smoothing steps last	int	$0 \leq$ i	10	a	109
Solve dissipation error equation (0/1)	int	0, 1	0	s	109

continued on next page

parameter name	type	range	default	in	page
Solve linear problem on grid level	int	1-6	1	s	109
Solver type	string	Flow, Output_only, Residual_only, DAdjoint, Primal, Freqdom, Findiff, CAdjoint, Volgrad	Flow	s	109
SRR limiter active (0/1)	int	0, 1	0	s	110
SRR limiter radius relaxation constant	float	1.0E-6, Infinity	0.03	s	110
SST-DES constant	array-float	0.6 .. 0.85, 0.5...0.61	{0.78, 0.61}	s	110
SST limitation version	string	Vorticity, Strain_rate	Vorticity	s	111
Start-Y for deformation	float	-	0	d	111
Streamline coordinates type set	string	-	xyz	s	111
Structured grid coarsening	float	$0 \leq x < 1$	0	p	112
Suppress error on orphaned points (0/1)	int	0, 1	0	s	113
Surface-weights file	string	-	(none)	a	113
Surface output description file	string	-	(none)	s	113
Surface output filename	string	-	(none)	d	113
Surface output period	int	-	200	s	113
Surface output values	string	-	xyz_p_other	s	113
Sutherland constant	float	$0 < x$	110.4	s	114
Sutherland reference temperature	float	$0 < x$	273	s	114
Sutherland reference viscosity	float	$0 < x$	1.716e-05	s	114
TAU recv-variables	string	-	(none)	d	114
TAU remote-code Id	int	-	0	d	114
TAU send-variables	string	-	(none)	d	114
Tecplot ascii output precision	int	0, 1	0		114

continued on next page

parameter name	type	range	default	in	page
Tecplot title	string	-	(none)		114
Time step smoothing factor	float	$1 \leq$ x	0	s	114
Trailing edge sweep angle	float	-	0.0	s	115
Transition block name	string	-	(none)	s	115
Transition blocks description file	string	-	(thisfile)	s	115
Transition history file name prefix	string	-	transition_history	s	115
Transition history output in premode (0/1)	int	0, 1	0	s	115
Transition history output values	string	-	set	s	115
Transition module description file	string	-	(thisfile)	s	116
Transition prediction (0/1)	int	0, 1	0	s	116
Transition prediction description file	string	-	(thisfile)	s	116
Transition prediction output directory prefix	string	-	Tranpred	s	116
Transition prescription (0/1)	int	0, 1	0	s	116
Transition prescription description file	string	-	(thisfile)	s	116
Transition prescription value	float	-	-	s	116
Transition prescription value name	string	-	-	s	116
Translation factor for shifted boundary points	float		1	p	116
TS transition prediction mode	int	0 - 12	10	s	117
Turbulence diffusion flux type TSL/Full (0/1)	int	0, 1	1	s	117
Turbulence equations use multigrid (0/1)	int	0, 1	0	s	117
Turbulence intensity for transition criteria	float	-	0.0	s	117

continued on next page

parameter name	type	range	default	in	page
Turbulence mode	string	RANS, DES, LES	RANS	s	117
Turbulence model version	string	see description	{SAO, Wilcox_k-w, RSM}	s	117
Turbulence shock correction (0/1)	int	0, 1	0	s	118
Type of agglomeration	string	private, MGridGen	private	p	119
Type of coarsening for MGridGen	int	1, 4	4	p	119
Type of grid movement	string	-	static	s	119
Type of partitioning (name)	string	private, kmetis, pmetis	private	p	119
Type of refinement for MGridGen	int	1, 2, 4, 6	6	p	119
Unsteady activate inner iteration output (0/1)	int	0, 1	0	s	119
Unsteady computational time step size	float	0 < x	-1	s	120
Unsteady extrapolation order	int	0, 1, 2, 3	1	s	120
Unsteady implicit scheme order	int	1, 2, 3	2	s	120
Unsteady inner iterations per time step	int	0 < i	40	s	120
Unsteady physical time offset	float	-max_float < value < +max_float	0.0	s	120
Unsteady physical time step size	float	0 < x	-2	s	120
Unsteady physical time steps	int	0 < i	1	s	120
Unsteady show pseudo time steps (0/1)	int	0, 1	0	s	120
Unsteady time stepping	string	-	(none)	s	121
Update transition blocks in para file (0/1)	int	-	0	s	121
Upwind flux	string	-	AUSMDV	s	121
Use Cauchy convergence control	string	off, relative, absolute, relative_dynamic, absolute_dynamic	off	s	121

continued on next page

parameter name	type	range	default	in	page
Use Coco script for task11 (0/1)	int	0, 1	1	s	122
Use Coco script for task21 (0/1)	int	0, 1	1	s	122
Use Coco TS database results as backup (0/1)	int	-	1	s	122
Use coordinates as displacement (0/1)	int	0, 1	0	d	122
Use cp,min/max (0/1) as reference for modified cp	int	0, 1	0	s	122
Use grid point as start coordinate (0/1)	int	0, 1	0	s	122
Use indifference point for task10 (0/1)	int	0, 1	1	s	123
Use logarithmic distribution for CF waves (0/1)	int	0, 1	0	s	123
Use logarithmic distribution for TS waves (0/1)	int	0, 1	1	s	123
Use modified dissipation for 2D (0/1)	int	0, 1	0	s	123
Use new multigrid (0/1)	int	0, 1	0	s	123
Use parallel initial partitioner (0/1)	int	0, 1	1		123
Use phi,le/phi,geo (0/1) to modify cp	int	0, 1	0	s	124
Use Prepcp (0/1/2)	int	0, 1, 2	0	s	124
Use separation point for task10 (0/1)	int	0, 1	0	s	124
User defined filter width	float	0.. positive float	0	s	124
Velocity factor for BL edge	float	-	0.95	s	124
Venkatakrishnan limiter constant	float	0, 10	0	s	124
Viscous calculation (0/1)	int	0, 1	0	s	124
Viscous flux type TSL/Full (0/1)	int	0, 1	1	s	124

continued on next page

parameter name	type	range	default	in	page
Vortical flow correction (0/1)	int	0, 1	0	s	125
Vortical flow correction model	string	(none)	0	s	125
Vorticity factor for BL edge	float	-	0.001	s	125
Wanted y+	float	0 < x	1	a	125
Which modes are used	array-int	-	0	d	125
Window size for pmin/pmax search	int	-	5	s	125
Write additional contourline data to file (0/1)	int	0, 1	0	s	125
Write boundary layer profiles to file (0/1)	int	0, 1	0	s	125
Write graph filename	string	-	(none)	p	126
Write pointdata dimensionless (0/1)	int	0, 1	0	s	126
Write streamline data to file (0/1)	int	0, 1	1	s	126
XLES constant	float	0.05 ...0.07	0.07	s	126

4.1.1 Sample parameter file

The parameter file containing the parameters needed for both the preprocessing and solver stages could look like:

```
######################################################################
# This are comment lines stored together with the solution data
# in order to allow the user to add a description of the problem
# to the final data.
# In order that this text is considered as a comment the #-sign
# has to be used in the 1st position of the line
######################################################################

------------------------------------------------------
required parameters
------------------------------------------------------
  Files/IO ---------------------------------------: -
                         Boundary mapping filename: (thisfile)
                             Primary grid filename: tau.grid
------------------------------------------------------
io
------------------------------------------------------
  Grid/Solution ----------------------------------: -
                               Output files prefix: my_solution
                               Restart-data prefix: tau.solution
  Indicator --------------------------------------: -
                                Adjoint-adapt-data: (none)
  Controls ---------------------------------------: -
                  Automatic parameter update (0/1): 0
               Write pointdata dimensionless (0/1): 0
  Files/IO ---------------------------------------: -
                                       Grid prefix: dual
------------------------------------------------------
preprocessing
------------------------------------------------------
  Agglomeration ----------------------------------: -
                              Agglomeration version: 0
                              Type of agglomeration: private
  partitioning -----------------------------------: -
                                  Number of domains: 1
```

```
                    Number of primary grid domains: 1
  Runtime optimisation --------------------------: -
                2D offset vector (0 / x=1,y=2,z=3): 0
                     Bandwidth optimisation (0/1): 1
            Cache-coloring (0/max_faces in color): 0
                      Compute lusgs mapping (0/1): 0
  Turbulence ------------------------------------: -
                           Consider wall roughness: global
                           Constant wall roughness: 0.
                             Global wall roughness: 0.
                                Roughness filename: global
  Extras ----------------------------------------: -
                       Project wall distance (0/1): 0
-------------------------------------------------------
adaptation
-------------------------------------------------------
  General ---------------------------------------: -
                               Maximum point number: -1
                Maximum point number per partition: -1
                          Maximum refinement level: -1
                    Maximum surface angle (degree): 30
                      Part of low quality elements: 0.0
                          Percentage of new points: 40
                                   Refinement mode: add
  Refinement regime -----------------------------: -
                          Number of cut-out volumes: 0
-------------------------------------------------------
solver
-------------------------------------------------------
  Transition Module -----------------------------: -
               Boundary part list for prescription: -
                 Bypass transition prediction mode: 0
                     CF transition prediction mode: 10
                                   Coco executable: ./coco
                        Half span reference length: 0.0
             Interpolate prescription values (0/1): 1
                   Keep Coco auxiliary files (0/1): 1
                        Keep Coco log files (0/1): 1
                    Keep Coco profiles files (0/1): 0
                        Keep Coco run files (0/1): 0
                   Keep Lilo auxiliary files (0/1): 1
```

```
                    Keep Lilo log files (0/1): 1
                    Keep Lilo run files (0/1): 0
                    Keep N-factor files (0/1): 1
                Keep files from pre-mode (0/1): 0
                              Lilo executable: ./lilo
                  N-factor extrapolation mode: 2
                 Number of prescription values: -
          Points skipped near stagnation point: 5
                  Prediction info output level: 3
                   Prediction iteration offset: 0
              Prepcp max. x/c on lower surface: 1.0
                       Prescription block name: (none)
                Prescription info output level: 1
             Prescription input data structure: Block
            Prescription values periodic (0/1): 1
        Relate period to start iteration (0/1): 1
              Relaxation factor for transition: 0.8
         Save only envelope of N-factors (0/1): 0
          Set transition at solver start (0/1): 0
              Set transition info output level: 1
                 TS transition prediction mode: 10
           Transition blocks description file: (thisfile)
          Transition history file name prefix: transition_history
    Transition history output in pre-mode (0/1): 0
              Transition history output values: set
           Transition module description file: (thisfile)
                 Transition prediction (0/1): 0
       Transition prediction description file: (thisfile)
  Transition prediction output directory prefix: Tranpred
               Transition prescription (0/1): 0
     Transition prescription description file: (thisfile)
               Transition prescription value: -
          Transition prescription value name: -
   Turbulence intensity for transition criteria: 0.0
    Update transition blocks in para file (0/1): 0
            Use Coco script for task11 (0/1): 1
            Use Coco script for task21 (0/1): 1
Use logarithmic distribution for CF waves (0/1): 0
Use logarithmic distribution for TS waves (0/1): 1
Write additional contourline data to file (0/1): 0
    Write boundary layer profiles to file (0/1): 0
```

```
              Write streamline data to file (0/1): 1
  Turbulence ------------------------------------: -
                              Rsm diffusion model: GGDH
                             Rsm dissipation model: isotropic
                         Rsm implementation version: FLOWer
                          Rsm length scale equation: Menter_BSL_omega
                          Rsm omega limiting factor: 1.e-5
                          Rsm re-distribution model: SSG/LRR-w
                         SA boundary condition type: smooth
                           Turbulence model version: SAO'  'Wilcox_k-w'  'RSM
  ------------------------------------------------: -
                 Inviscid flux discretization type: Upwind
                        Central dissipation scheme: Scalar_dissipation
                                        Upwind flux: AUSMDV
  Timestep Settings -----------------------------: -
                      Number of Runge-Kutta stages: 3
  References ------------------------------------: -
                             Reference Mach number: not_defined
                                   Reynolds length: 1
                                   Reynolds number: not_defined
  Dual time -------------------------------------: -
                            Unsteady time stepping: (none)
  Controls --------------------------------------: -
                    Monitoring significant figures: 4_4_4_4
  Timestepping Start/Stop -----------------------: -
                           Maximal time step number: 200
                                      Output period: 10
  Monitoring ------------------------------------: -
              Extended coefficient monitoring (0/1): 0
  Multigrid -------------------------------------: -
                                        CFL number: 1.8
                            MG description filename: 4w
                           Multigrid indicator (0/1): 0
  Geometry --------------------------------------: -
                                        Grid scale: not_defined
-----------------------------------------------------
deformation
-----------------------------------------------------
  Test deformation ------------------------------: -
                               Delta-Z at outer-Y: 0.01
                            Outer-Y for deformation: 100
```

```
                          Start-Y for deformation: 0
  External deformation ---------------------------: -
                       Deformation reduction factor: 0.85
                      Deformed coordinates filename: (none)
                        Final freeform box filename: (none)
                      Initial freeform box filename: (none)
             Use coordinates as displacement (0/1): 0
-----------------------------------------------------
tools
-----------------------------------------------------
                         Set zero time origin (0/1): 0
                    Tecplot ascii output precision: 0
                                     Tecplot title: (none)
-----------------------------------------------------
Motion-IO
-----------------------------------------------------
  Motion -----------------------------------------: -
            Degree of Fourier series for rotation: 0
         Degree of Fourier series for translation: 0
                Degree of polynomial for rotation: 0
             Degree of polynomial for translation: 0
              Evaluate forces and moments at node: -
    Fourier coefficients for rotation (cos) pitch: 0
     Fourier coefficients for rotation (cos) roll: 0
      Fourier coefficients for rotation (cos) yaw: 0
    Fourier coefficients for rotation (sin) pitch: 0
     Fourier coefficients for rotation (sin) roll: 0
      Fourier coefficients for rotation (sin) yaw: 0
     Fourier coefficients for translation (cos) x: 0
     Fourier coefficients for translation (cos) y: 0
     Fourier coefficients for translation (cos) z: 0
     Fourier coefficients for translation (sin) x: 0
     Fourier coefficients for translation (sin) y: 0
     Fourier coefficients for translation (sin) z: 0
   Hinge - Fourier coefficients for rotation (cos): 0
   Hinge - Fourier coefficients for rotation (sin): 0
      Hinge - polynomial coefficients for rotation: 0
            Hinge - reduced frequency for rotation: 0
       Hinge - reduced frequency reference length: 0
-----------------------------------------------------
chimera
```

```
-----------------------------------------------------
  Chimera wall projection ------------------------: -
              Apply Chimera wall projection (0/1): 0
                    Mismatch of overlapping walls: 0.0
  Hole Definition --------------------------------: -
                                            Block: 0
                                 Exclude cut block: -
                                     Hole filename: (none)
-----------------------------------------------------
cut plane output
-----------------------------------------------------
                                  Number of planes: 1
                                    Plane normal x: 0
                                    Plane normal y: 1
                                Plane output period: all
                                    Plane support x: 0
                                    Plane support y: 0
                                    Plane support z: 0
-----------------------------------------------------
extra field pointdata output
-----------------------------------------------------
                      Field output description file: (none)
                                 Field output values: (none)
-----------------------------------------------------
libs
-----------------------------------------------------
                             Profile output values: (none)
-----------------------------------------------------
partitioning
-----------------------------------------------------
            Use parallel initial partitioner (0/1): 1
-----------------------------------------------------
surface output
-----------------------------------------------------
                   Surface output description file: (none)
                             Surface output period: 200
                             Surface output values: xyz_p_other
-----------------------------------------------------
transition
-----------------------------------------------------
                             Transition block name: (none)
```

```
Output files prefix: new-configuration
                end

Output files prefix: new-new-configuration
```

Some lines contain unknown strings. Thus these lines are considered as comment lines.

The keyword ‘Output files prefix’ occurs three times. The string ‘new-configuration’ is assigned to this parameter. As second entry it overwrites the previous assigned name ‘configuration’. The last entry is not taken into account, because of the leading key ‘end’. This key stops the search for other keywords in this parameter file.

4.2 Input Parameters

`2D offset vector (0 / x=1,y=2,z=3):` integer, default 0

- This option is for two-dimensional computations, using 3d-grids with a width of e.g. one cell between two symmetry planes. If all points in the second symmetry plane has a constant offset w.r.t. their point numbers, the dual grid is reduced to a real 2D-grid consisting only of the first symmetry plane in order to reduce the solver CPU time by about 65%.
  Under the same assumption an axisymmetric grid for flows without swirl (setup taugrid) can be reduced in a similar way, except that one of the Euler walls, which are the replacements of the symmetry planes, will survive the reduction.
  Do not switch on this option for 3d-flows! If the grid is a quasi 2D grid but of one cell width but with a non appropriate offset of the point numbers the grid can not be reduced. In this case the edges in the offset direction are flagged such that the flux computation for the edges in the offset direction can be skipped, which results in about 15% CPU reduction for the solver.

`2nd order dissipation coefficient:` float, default 0.5

- Second order dissipation factor for capturing shocks and is used whenever the central scheme with scalar or matrix dissipation is employed.

`Accumulate queue time (0/1):` integer, default 1

- Switch on (1) or off (0) that CPU time is accumulated over all processes to compare CPU time with queue limit if in case that the code is started under a queueing system. The default is suited for PBS. Switching off the parameter is needed for the SGE queueing system.

`Adaptation sensitivity:` float, default 0.9

- This parameter has an effect only if the `Refinement mode` is `both`. In this case, the indicator limits for inserting new points (or removing old points) are divided by (or multiplied) with this value, respectively. The parameter is intended to stabilize computations with repeated adaptation. Values in the range (0.5, 1) are recommended and the parameter has to be a positive value. As the value of the parameter is reduced, fewer changes to the mesh are made.

`Adjoint-adapt-data:` character string, default (none)

- The name (including the path) of the NetCDF input file containing the adaptation sensors resulting from the run of the adjoint solver. If the parameter Indicator-type is set to adjoint and Adjoint-adapt-data is set to (none) or does not contain the specified sensor variable adaptation will stop with an error message.

`Aerodynamic sweep angle:` float, default 0.0

- Prescription of the aerodynamic sweep angle if Set aerodynamic sweep angles is set to 1.

`Agglomeration version:` integer, default 0

- Set to '0' is the old version. If set to '1' some modifications limit the maximal number of children per agglomerated volume to about 12 in 3d (6 in 2d). This option has helped in some cases, where multigrid crashes otherwise, and might become the future default.

`Amplitude description filename:` character string, default (none)

- File name (including path) of file containing keywords/parameters for the modal amplitude simulation. If this parameter is set to '(thisfile)', all of the input can be contained in the same file.

`Amplitude filename:` character string, default (none)

- File name (including path) of file containing modal data.

`Analysis incompr./compressible (0/1) [ICOMP]:` integer, default 0

- Lilo input parameter ICOMP: A compressible medium is assumed, unless ICOMP = 0. In this case, an incompressible medium is simulated by setting Ma to a value of one thousandth.

`Apply Chimera wall projection (0/1):` integer, default 0

- Switch on (1) or off (0) the Chimera wall projection method. The method should be activated if two grids have an overlap on a body surface. This will minimize interpolation errors close to curved surfaces.

`Assigned transition block:` integer, default -1

- Number of transition block (of the preprocessing), to which the prediction/ prescription block will be assigned.

`Ausm scheme dissipation:` float, default 0.25

- Dissipation factor used if the Ausm scheme is employed, not active otherwise.

`AUSMDV shock fix (0/1):` integer, default 0

- The AUSMDV suffers from the carbuncle phenomenon in special cases (Mach number higher than about 2.5 and a shock nearly aligned to the grid). The switched on (1) shock fix can then prevent the carbuncle. To avoid the calculation of the shock fix in other cases the default is set to zero.

`AUSMP alpha:` float, default 0.1875

- Scaling parameter in the AUSM+ scheme for the calculation of the pressure diffusion term in the subsonic range.

`AUSMP beta:` float, default 0.125

- Scaling parameter in the AUSM+ scheme for the calculation of split Mach number for the momentum flux in the subsonic range (4th-degree polynomial part).

`AUSMP pressure weight:` integer, default 1

- Pressure limiter for the AUSM+ scheme. 0: Classical AUSM+ scheme without pressure limiter; 1: AUSMPW scheme; 2: AUSMPWP scheme.

`Automatic parameter update (0/1):` integer, default 0

- Switch on (1) or off (0) the automatic update of parameters to be performed by the solver. Each time a new result file is written out the line 'Restart-data prefix: name' is added to the end of the parameter file, with 'name' addressing the new result file. Thus when the solver is restarted, the most recent result is used.

`Automatic parameter update mode (0/1):` integer, default 0

- No effect for sequential program mode. In parallel mode the update of the parameter file is done per default only from the process 0 in order not to add changed parameters N times to the parameter file when running N processes. In case of having local disks for each process with a copy of the parameter file on each of these disks, it might become necessary that each process updates its own copy of the parameter file. This parameter file update from each process can be switched on using the mode 1. Note: Mode 1 might lead to problems, when using shared disks, because all processes try to update the parameter file at the same time.

`Axisymmetry (0/1):` integer, default 0

- Specify this option when intending to calculate axisymmetric flows with an upwind scheme. It will have no effect when used with a central scheme, and is particularly recommended for use with the hypersonic version of the solver and structured grids. Turning this option on (1) should considerably improve solution quality by removing noise and asymmetry at the axisymmetry axis. To calculate an axisymmetric flow with this option, the user must provide an appropriate wedge-shaped grid with a small included angle. Additionally, the grid should contain an axisymmetry axis boundary, one cell wide, with the width being very small compared to the grid length scale.

`Axisymmetry axial direction:` array-float, default {1, 0, 0}

- A vector specifying the axis direction for axisymmetric flows. It doesn't matter whether the vector is given in either the 'backward' or 'forward' direction along the axis.

`Axisymmetry radial direction:` array-float, default {0, 0, 1}

- A vector specifying the radial direction for axisymmetric flows, which should be normal to the axisymmetry axis boundary. It doesn't matter whether the vector is given in either the 'inward' or 'outward' radial direction.

`Bandwidth optimisation (0/1):` integer, default 1

- Switch on (1) or off (0) an optimization of the bandwidth of point numbers along edges of the grid. This optimization can not be used for vector computers, for which a special vector coloring is required.
  The runtime optimization is of advantage for cache computers. The bandwidth is given by the difference of the numbers of the points connected by edges. The differences can vary between 1 and the total number of grid points. Small differences means that values stored for the two points of an edge are near by in memory. When accessing values of one edge point the values of the other edge point might be loaded into the cache with the same cache line. Thus, like in the case of the flux computation where values of both edge points are accessed at the same time in a loop over all edges, the loads into the cache can be reduced. This accelerates the run-time performance of the code: more than 50 percent CPU-time reduction can be achieved. The effect depends on the problem size (small problems fit into the cache anyway), on the hardware (i.e. the cache size) and on the specific case (i.e. the initial bandwidth of the given grid). In any case, the bandwidth optimisation option should never be of disadvantage with respect to solver runtime. However, one disadvantage is that the memory needed for the preprocessor is increased.

`BL edge criterion:` integer, default 3

- Criterion to detect boundary layer edge:
  - 0: 95% criterion (specify with Velocity factor for BL edge)
  - 1: chose best of 80%-100% criterion or apply local maximum
  - 2: vorticity criterion (specify with Vorticity factor for BL edge)
  - 3: same as mode 3 with advanced search for local maximum

`Block:` integer, default 0

- Parameter inside of a hole definition block to associate the hole definition grid with a chimera block. The numbering is implicitly given by the order the component grids are fused to a multiblock grid.

`Boundary layer data mode:` integer, default 0

- Selection of the source (TAU or COCO) for the boundary layer data if the transition is to be computed. COCO is a laminar BL code for swept tapered wings.
  - Mode 0: use TAU BL-data for transition prediction,
  - Mode 1: use COCO BL-data for transition prediction,
  - Mode 2: use COCO for CF, use TAU for TS,
  - Mode 3: use COCO for TS, use TAU for CF.

`Boundary mapping filename:` character string, input required

- The name (including the path) of the boundary mapping ASCII-file. The boundary mapping file contains all parameters used by the different boundary treatments.

`Boundary part list:` character string, default -

- Identifier of the boundary marker for which transition prediction is performed applying the individual block settings which follow: has to match the boundary mapping in the TAU para file.

`Boundary part list for prescription:` character string, default -

- Identifier of the boundary marker for which transition prescription is performed applying the individual block settings which follow: has to match the boundary mapping in the Tau para file.

`Boundary point (0/1):` array-int, default 0

- If this parameter is set to 1, then the solver knows that the corresponding profile point is located on a boundary. If now a profile line is selected, the slope of this line will be normal to the boundary at the starting point of the line. Only for Cartesian meshes.

`Bypass transition prediction mode:` integer, default 0

- Mode for the prediction of bypass transition.

Mode	Method	Available
0	none	release
1	Mayle	development
2	H12 bypass detection	development
3	Braslow/von Doenhoff	development
4	Braslow/von Doenhoff + H12 bypass detection	development

`Cache-coloring (0/max_faces in color):` integer, default 0

- This option determines the type of coloring. The default (0) is for vector-coloring, which has to be set when the code shall be executed on a vector-type machine. For non-vector machines this option is active only if no bandwidth optimisation is invoked, which activates an other algorithm for optimization, which is more effective. If bandwidth optimisation is turned off the parameter 'Cache-coloring' is turned on by default with a value of 200000. The 'Cache-coloring' is an older version for cache-optimization which is still kept as additional choice.
  For improved performances on cache-based scalar machines a value larger than zero has to be chosen. The value determines the number of faces (edges) to be grouped in different colors. Optimal performance is achieved if all edge-based quantities of a color fit in the cache (Note, 2equ-Turbulence models requires more storage than 1- or 0-Equation computations). However, for too small numbers a lot of colors have to be built, which increases (considerably) the CPU-time in the preprocessing step. Thus, the optimum value depends on the target architecture and can be determined by a single test for each architecture. (For 1MB-caches a value between 50000 and 500000 may be suitable)

`Central convective meanflow flux:` character string, default Average_of_flux

- This parameter defines the discretization scheme of the convective fluxes of the meanflow equations in case of a central scheme for the convective fluxes, "Inviscid flux discretization type: Central".
  - Average_of_flux:
    Standard TAU central scheme. The flux is the central average of the analytic flux on each side of the face.
  - Flux_of_average:
    Central scheme. The flux is the analytic flux of centrally averaged conservative variables on the face.

`Central convective turbulence flux:` character string, default {Average_of_flux, Roe}

- This parameter defines the discretization scheme of the convective fluxes of the turbulence equations in case of a central scheme for the convective fluxes, "Inviscid flux discretization type: Central". For stability reasons the central scheme is only recommended for Spalart-Allmaras models, while upwind schemes are recommended for k-ω models and RSM.
  - Average_of_flux:
    Central scheme (default for SA models). The flux is the central average of the analytic flux on each side of the face.
  - Flux_of_average:
    Central scheme. The flux is the analytic flux of centrally averaged conservative variables on the face.

- AUSMDV:
  First order AUSMDV upwind scheme.
- Roe:
  First order Roe upwind scheme (default for k-ω models and RSM).
- Roe2nd:
  Second order Roe upwind scheme.

`Central dissipation scheme:` character string, default Scalar_dissipation

- The central solvers dissipation scheme can be selected by the either the keywords: Scalar_dissipation or Matrix_dissipation.

`Cf-min offset for extrapolation mode 3:` integer, default 10

- Size of the region to be considered for extrapolation of N-factor curve for N-factor extrapolation mode:3.

`CF transition prediction mode:` integer, default 10

- Mode for the prediction of cross flow transition.

Mode	Method	Available
0	none	release
1	C1 (Arnal)	development
9	eN-Method with N-factors from Database method (Casalis, Arnal)	development
10	eN-Method with N-factors from linear stability solver Lilo	release

`CFL number:` float, default 1.8

- CFL number used on the fine grid level.

`CFL number (coarse grids):` float, default 1.8

- CFL number used on all coarse grid levels. Note: the optimal coarse grid CFL-number depends on the determination of the timestep size. The scaling factor for the default value also depend on this choice.

`CFL number (large grad p):` float, default 1.8

- This CFL number is used in the presence of large pressure gradients (i.e. shocks). Via a bridging function the general CFL number is reduced to this value near shocks, increasing the stability of the multigrid scheme.

`Change boundary control volumes (0/1):` integer, default 0

- This option enables a modification of the control volumes restricted to Euler walls, which gives more accurate solutions (reduction of pressure loss). The effect is distinct on structured grids or grids with structured sublayers (hexas or prisms). On tetrahedral grids only a small improvement is observed.

`Change wall normal distribution (0/1):` integer, default 1

- If turned on, the wall normal points in a hybrid mesh are redistributed depending on y^+.

`Coarse grid upwind flux:` character string, default Van_Leer

- Formerly "First order upwind solver".
  When using the "Upwind" inviscid flux type, the upwind scheme to use on the coarse grids: Van_Leer, AUSM_Van_Leer, AUSMDV, Roe, MAPS+, EFM, AUSMP, AUSMPWP. The Van_Leer scheme is recommended due to computational costs and high stability. AUSMPWP is still under research. If low Mach number preconditioning is used, the only choice is MAPS+.

`Coarse grid viscous flux type TSL/Full (0/1):` integer, default 0

- The type of viscous flux to use on coarse grids. The default is always TSL since solution gradients on the coarse grids are likely to be of poor quality, giving poor quality full viscous fluxes.

`Coco executable:` character string, default ./coco

- Path to Coco executable -> ./coco

`Compressible mixing layer correction (0/1):` integer, default 0

- The correction is designed to improve the prediction of the spreading rate in supersonic mixing layers. It should not affect boundary layers even in hypersonics. If the convective Mach number ($|u_1 - u_2|/(a_1 + a_2)$) is about one the unmodified k-ω model overpredicts the spreading rate by about a factor of 4. Using the correction the overprediction is reduced to less than 2, but attention have to be paid, when comparing with published data. The spreading rate depends not only on the convective Mach number but on the velocity levels and on the density difference as well as on which definition of the spreading rate is used. The correction is recommended if shear layers are calculated in regions where the turbulent Mach number ($\sqrt{2k}/a$, *Field output values*: macht) is above 0.25. The correction is implemented for the following two equation turbulence models: Wilcox_k-w, Menter_BSL, Menter_SST, Wilcox_k-w+SST and 2layer_k-eps.

`Compute exact whirlflux (0/1):` integer, default 0

- With this parameter the additional flux due to rigid body motion is integrated exactly over the control volume surface. For rotating flows the otherwise approximate calculation is not freestream consistent. For turbomachinery and helicopter applications it is strongly recommended to set this parameter to one. The calculation of exact whirl fluxes increases size of the dualgrid around 30 - 40 percent.

`Compute flow statistics:` character string, default (none)

- The parameter takes a string for which allowable values are "mean", "variance", and "meanvelgrad". The string "mean" activates computation of running means for laminar and turbulent viscosities, density, pressure, and velocity. The string "variance" actives computation of running cross- and covariances for the fluctuating velocity components (ie uu,uv, uw, vv, vw, ww) and covariances for fluctuating pressure (pp), while "meanvelgrad" activates computation of the mean of the velocity gradients. These three control strings are used in combination e.g. "mean" activates only running means, "mean+variance" activates both running mean and variance calculations. Note that the means MUST be stochastically stationary if one wishes to compute the variances effectively. The mean velocity gradients are used to derive time averaged values of the skin friction coefficient. The mean pressure is used to construct the mean Cp field. The time-averaged skin friction and Cp are activated by adding the strings "mean_cf" in the surface output control string and "mean-cp" in the field/surface output control strings. Setting the string to "mean+meanvelgrad" computes means plus the means of the velocity gradients. In short the input string is constructed as "a+b+c+.." where a,b,c.. correspond to the keywords "mean", "variance" or "meanvelgrad". In the present implementation a keyword can be repeated without causing problems.

`Compute harmonics of global forces (0/1..n):` integer, default 0

- Number of harmonics to be calculated for the global forces and moments. The Fourier-analysis can be applied to unsteady simulations using the dual-time stepping method. The harmonics are written to the screen output (stdout/log file) after the end of the unsteady simulation. The variables under consideration are `C-lift`, `C-drag`, `C-sidef`, `C-mx`, `C-my`, `C-mz`.
  Note: At least one period of rigid-body-motion must be completed for this computation. Mathematical background: a standard Fourier-analysis is applied to the history of forces and moments:
  $$F(t) = a_0 + \sum_{k=1}^{n} \left(a_k \cos \frac{2\pi}{T} k \cdot t + b_k \sin \frac{2\pi}{T} k \cdot t \right)$$
  with
  $$a_0 = \frac{1}{T} \sum_{i=1}^{T/\Delta t} f(t_i)$$

$$a_k = \frac{2}{T} \sum_{i=1}^{T/\Delta t} f(t_i) \cos \frac{2\pi}{T} k \cdot t_i \quad ; \quad 1 \leq k$$

$$b_k = \frac{2}{T} \sum_{i=1}^{T/\Delta t} f(t_i) \sin \frac{2\pi}{T} k \cdot t_i \quad ; \quad 1 \leq k$$

where T is the time for one period, Δt is the size of the time step, $t_i = t2 - T + \Delta t \cdot i$ is the time at time step i for a flow simulation starting at $t = t1 \leq t2 - T + \Delta t$ and finishing at $t = t2$. The symbol $f()$ denotes any of the forces or moments to be analyzed. a_k and b_k are the Fourier-coefficients of the k-th harmonic.

`Compute harmonics on surface (0/1):` integer, default 0

- The time-average and the 1st harmonic of the surface pressure coefficient are computed. Note: Output to surface-file is enabled if 'cpharm' is added to the list of values active for surface output. Meaningful output values are only obtained when the surface output corresponds to the end of a period, otherwise only zeros are written to the file.
  The mathematical background of the Fourier-analysis is the same as described for the parameter `Compute harmonics of global forces (0/1..n)`.

`Compute lusgs mapping (0/1):` integer, default 0

- If the implicit solvers `Lusgs` or `Sgs` are to be used in the solver, then the preprocessor must compute a special face-mapping to support them. This results in slightly larger dual-grids.

`Compute parent faces (0/1):` integer, default 0

- If this parameter is set, the parent faces are computed and stored in the dual grids. This information is required for the THETA-solver if the momentum equations are solved by a multigrid method.
  Ignore when using the TAU-Code

`Compute passive sas correction:` integer, default 0

- If this parameter is set, the SAS correction term is computed for output purposes only but the SAS mode is not activated.

`Compute point-face connectivity (0/1):` integer, default 1

- Some linearized "Solver type"s require point-to-face connectivity in the dualgrid, this provides that. Size of dualgrids and memory use of the solver can be significantly increased.

`Compute statistics (0/1):` integer, default 0

- When set this flag allows the code to set up computing the unsteady statistical field. At moment first- and second-order moments of the velocity and pressure are computed and stored. In case of the particle tracer it turns on/off the print out into the stdout file of some statistics about the time integrator.

`Consider wall roughness:` character string, default global

- Define how wall roughness shall be considered:
  - "global": global wall roughness (one value for all surfaces)
  - "constant": constant wall roughness per marker
  - "pointwise": pointwise specification of wall roughness

`Constant wall roughness:` float, default 0.

- Markerwise constant surface roughness value to be specified for each boundary marker. The value has to be specified in grid units (→ 'Grid scale'). Notice that this parameter specifies the equivalent sand grain roughness height, and not the geometrical roughness height. Parameter is only active, if parameter "Consider wall roughness" is set to "constant".

`Contourline cut extraction mode:` integer, default 1

- Calculation mode for contourlines (line-in-flight cuts):
  - 0: Contourline is calculated by cuts of plane with grid lines. Not all cutting points are saved to avoid clustering on points near grid points. The behaviour can be altered by the parameter Scale for additional contourline cuts.
  - 1: Contourline is calculated by cuts of plane with grid lines. All cutting points are saved and form the contourline.
  - 2: Old implementation of calculation method (Tau 2009.1.0): project grid points to cut plane.

`Contourline plane normal definition:` character string, default xz

- Character string to identify the cutting plane for contourline calculation).Valid names are:
  - xy: cutting plane is the x-y-plane
  - xz: cutting plane is the x-z-plane
  - yz: cutting plane is the y-z-plane

`Control volume edge weight (0/1):` integer, default 1

- This option chooses the method used for the computation of the dual grid control volumes. They are determined by the edge mid points, the centers of the element faces and the centers of the primary grid elements. The definition of the centers of the element faces and the centers of the primary grid elements is not unique. Quite often the barycenters of the face and elements are used (option 0). Option 1, which is of advantage for stretched elements, computes the face centers by an weighted average of the mid points of the edges connected to the face. The element centers are computed by an weighted averaged of the face centers. As weighting function the square of the related edge lengths is used.

`Correct metric for boundary control volumes (0/1):` integer, default 1

- Setting this parameter switches on the metric correction for control volumes surrounding shifted boundary points. The correction usually yield better results and a more robust behavior of the THETA-solver.
  Ignore when using the TAU-Code.

`Correction smooth epsilon:` float, default 1.8

- If the multigrid corrections (forcing terms) are smoothed after the prolongation step Correction smoothing steps the strength of the smoothing is dependent on this value. In the explicit case (Laplacian type smoother) the weight of the point value is (eps), the weight of its neighbors is (1-eps). In the implicit case, (eps-1) can be interpreted as the weight given to the neighboring point values (see also Residual smooth epsilon). In the implicit case, (eps) is interpreted as the ratio between the explicit (without smoothing) CFL-Number and the implicit CFL-Number. Not active, if no smoothing is employed.

`Correction smoother:` character string, default Point_explicit

- The correction smoothers available can be selected by the keywords:
  Point_explicit, Point_implicit. Since the implicit smoother is not robust and/or efficient, the use of the default smoother (Point_explicit) is recommended. The implicit smoothing is to be improved in future.

`Correction smoothing steps:` integer, default 2

- Number of smoothing steps for the multigrid corrections (forcing terms).

`Cost function:` character string, default C-drag

- For "Solver type" of "DAdjoint" or "CAdjoint": the cost function with respect to which the adjoint problem is constructed and solved. For "Primal", the derivative of this quantity with respect to the "Design variable" will be available for monitoring output with key "dJ/da".

`Cost function part total/pressure/viscous (0/1/2):` integer, default 0

- For "Solver type" of "DAdjoint", "Primal" or "CAdjoint": for viscous cases the component of the cost function to use in the adjoint/primal problem.

`Couple mean flow and turbulence equations (0/1):` integer, default 0

- The Matrix Dissipation operator is usually only applied to the mean flow equations and the turbulence equations are weigthed by the wave speed over the face, option 0. As the turbulence equations usually also depend on all variables one can also consider the full Matrix Dissipation operator including the coupling of the mean flow and the turbulence equations, option 1. So far this parameter only works for one- and two-equation turbulence models. Low Mach number preconditioning is also not yet implemented.

`Critical N-factor CF:` float, default 9.0

- The setting of this parameter and of 'Critical N-factor TS' is needed for 3d cases. In doing so, a rectangular stability boundary is used, which is the simplest (but not always the best) approximation of the stability boundary. For a better approximation see the parameter 'N-ts/N-cf Diagram'. The default value corresponds to free flight conditions.

`Critical N-factor TS:` float, default 12.0

- The setting of this parameter only is sufficient for 2d cases. The default value corresponds to free flight conditions.

`Cut-off value:` float, default 1.0

- This parameter is only used if preconditioning is switched on. It prevents the 'preconditioning' matrix from becoming singular. It is recommended to use the default value of 1.0. If the simulation becomes unstable, the value should be stepwise increased.

`Deformation description filename:` character string, default (none)

- The name (including the path) of the file containing a user-defined rigid body motion, which will be applied by the deformation to define the new surface coordinates in each time step. The content of the file is the same as required for a Motion description filename for the TAU-Code solver. The keywords are different to allow for the use of different motions by the solver and the deformation at the same time.

`Deformation reduction factor:` float, default 1

- If the deformation, i.e. the difference between the old and the deformed coordinates, shall be reduced, the displacement vector is multiplied by the given factor. In case that displacements are given instead of new coordinates (see keyword 'Use coordinates as displacement (0/1)') the displacements are multiplied by this factor before adding them to the old coordinates. Note, also negative values for the reduction factor are allowed.

`Deformed coordinates filename:` character string, default (none)

- Input file in NetCDF-format containing new coordinates of the grid points which are selected for the grid deformation. The content of the file is: double x(no_of_points), double y(no_of_points), double z(no_of_points) and global_id(no_of_points). The latter array contains the id of each new coordinate in the global grid (i.e. the non-partitioned grid). E.g. if the first entry of this array is 4711, the first coordinate x-y-z will be assigned to the grid point 4711.

`Delta-Z at outer-Y:` float, default 0

- This parameter defines the displacement at 'Outer-Y'. If this parameter equals zero (the default) no test deformation is applied and an "Internal deformation" or an "External deformation" needs to be defined to use the deformation program.

`Delta frequency for task 30 loop:` float, default 150.0

- Specifying the delta by which the frequency of the instability wave is reduced when executing task 30 in a loop (Number of loops for task 30).

`DES model:` character string, default {SA, XLES}

- This parameter sets the BASE DES model. In Spalart-Allmaras, there is only one BASE model: the SA DES variant - where only the destruction length scale is modified. In the K-W models, one can use either XLES (which modifies both dissipation and destruction scales) or SST base model(which modifies only the destruction length scale). The BASE can be considered as a grouping of DES models into a family. An example should suffice: for SA-DES the base model is "SA" with family members "DES", "DDES" and "IDDES" with sub-family elements including "Low Re model" and so on.

`Design variable:` character string, default Farfield-alpha

- For "Solver type" of "DAdjoint" and "CAdjoint": the derivative of the cost function with respect to this variable will be available for monitoring output with the "Monitoring values" key "Cf-gradientT", and will be written to stdout at the end of the computation. For "Primal" the right-hand side of the linear problem will be based on this design variable.

`Displacement scale:` float, default 0.

- If a grid is deformed for each point on the deformation surface a local rotation is computed. This rotation is applied to edges connected to this point and then, in a next step, to the edges connected to this points neighbors and so on. This rotation of grid lines is needed to keep the grid similar to the non-deformed grid, e.g.: to keep wall normal lines aligned with the surface normal direction. However, this rotation can be applied only near the surface and has to be reduced far away inside the grid, because typically the outer boundary of grid are fixed if e.g. a wing surface is deformed. For this

blending between full rotation (near the deformed surfaces) and zero rotation (far away from the surfaces) a characteristic scale is needed, which is the 'displacement scale'. For all points which are nearer to the displacement surface than the value given by the 'displacement scale' full rotation is applied. This rotation is decreased up to zero for points further away. If the value is zero (which is the default) the largest edge length on the displacement surface is taken as value for displacement scale. This seems to be a default value suited for many cases. However, in cases where the deformation fails a user-defined value might lead to success. It is assumes that the value to be defined should be of the order of the height of structured layers in hybrid grids or other typical scales like the thickness of a wing or something similar. The input value is considered to be in grid units.

`EARSM expansion order:` integer, default 1

- Formerly "EARSM expansion order (1/2/3/4)".
  The value of this switch determines the maximum order to which the anisotropy tensor is expanded. The experience with values higher than 2 is limited, but the higher order expansions should be beneficial, for example, in flows with secondary recirculation.
  This parameter is only active with the Hellsten EARSM k-ω model, i. e. if parameter "Turbulence model version" is set to "WJ+Hellsten".

`Edge relation to unstructured part:` float, default 0

- If the wall normal distribution is turned on, this sets ratio of the distance of the last two points on a wall normal ray and the average length of the adjacent edges. If set to 0, the distance is not changed, which is preferable (more robust) for grids with small overall heights of the prismatic layer. This parameter is only active if an equal number of layers is detected in the grid.

`Effective sweep angle:` float, default 0.0

- Prescription of the effective sweep angle if Set effective sweep angles is set to 1.

`Error for Cauchy convergence cdrag:` float, default -1.0

- If parameter Use Cauchy convergence control is activated, then inner iterations are stopped if the relative resp. absolute changes in drag coefficient during a certain number of previous inner iterations specified by the parameter Number of samples for Cauchy convergence are smaller than this tolerance value, i.e., if $\frac{|c_d^n - c_d^{n-k}|}{|c_d^n|} \leq tol\_drag$ in case of relative evaluation of convergence or if $|c_d^n - c_d^{n-k}| \leq tol\_drag$ in case of absolute evaluation of convergence for all $k = 1, \ldots n\_sample\_inner$. Therein, an upper index denotes the inner pseudo time step. The default value is set to -1.0, so that the convergence check for this coefficient is turned off unless a positive value is specified. The recommended value is $1. \times 10^{-6}$. When Use Cauchy convergence control is relative_dynamic or absolute_dynamic, the number of inner iterations considered for estimation of convergence is determined by the parameter Factor for dynamic Cauchy convergence.

`Error for Cauchy convergence clift:` float, default -1.0

- If parameter Use Cauchy convergence control is activated, then inner iterations are stopped if the relative resp. absolute changes in lift coefficient during a certain number of previous inner iterations specified by the parameter Number of samples for Cauchy convergence are smaller than this tolerance value, i.e., if $\frac{|c_l^n - c_l^{n-k}|}{|c_l^n|} \leq tol\_lift$ in case of relative evaluation of convergence or if $|c_l^n - c_l^{n-k}| \leq tol\_lift$ in case of absolute evaluation of convergence for all $k = 1, \ldots n\_sample\_inner$. Therein, an upper index denotes the inner pseudo time step. The default value is set to -1.0, so that the convergence check for this coefficient is turned off unless a positive value is specified. The recommended value is $1. \times 10^{-6}$. When Use Cauchy convergence control is relative_dynamic or absolute_dynamic, the number of inner iterations considered for estimation of convergence is determined by the parameter Factor for dynamic Cauchy convergence.

`Error for Cauchy convergence cmy:` float, default -1.0

- If parameter Use Cauchy convergence control is activated, then inner iterations are stopped if the relative resp. absolute changes in pitching momentum coefficient during a certain number of previous inner iterations specified by the parameter Number of samples for Cauchy convergence are smaller than this tolerance value, i.e., if $\frac{|c_{my}^n - c_{my}^{n-k}|}{|c_{my}^n|} \leq tol\_my$ in case of relative evaluation of convergence or if $|c_{my}^n - c_{my}^{n-k}| \leq tol\_my$ in case of absolute evaluation of convergence for all $k = 1, \ldots n\_sample\_inner$. Therein, an upper index denotes the inner pseudo time step. The default value is set to -1.0, so that the convergence check for this coefficient is turned off unless a positive value is specified. The recommended value is $1. \times 10^{-6}$. When Use Cauchy convergence control is relative_dynamic or absolute_dynamic, the number of inner iterations considered for estimation of convergence is determined by the parameter Factor for dynamic Cauchy convergence.

`Error for Cauchy convergence cost function:` float, default 1.0e-6

- If parameter Use Cauchy convergence control is activated, then inner iterations are stopped if the relative resp. absolute changes in cost function coefficient during a certain number of previous inner iterations specified by the parameter Number of samples for Cauchy convergence are smaller than this tolerance value, i.e., if $\frac{|c_d^n - c_d^{n-k}|}{|c_d^n|} \leq tol\_cost\_function$ in case of relative evaluation of convergence or if $|c_d^n - c_d^{n-k}| \leq tol\_cost\_function$ in case of absolute evaluation of convergence for all $k = 1, \ldots n\_sample\_inner$. Therein, an upper index denotes the inner pseudo time step. The recommended value is $1. \times 10^{-6}$. When Use Cauchy convergence control is relative_dynamic or absolute_dynamic, the number of inner iterations considered for estimation of convergence is determined by the parameter Factor for dynamic Cauchy convergence.

`Exclude cut block:` array-int, default -

- Optional parameter used for chimera hole definition. Integer list of additional block numbers to be excluded from hole cutting by the chimera hole definition. By default the associated block of the hole definition is always excluded from hole cutting and needs not to be specified with this parameter.

`Extended coefficient monitoring (0/1):` integer, default 0

- Enable/disable monitoring output of pressure and viscous parts of forces, moments, and their coefficients, both during a simulation and for specified boundary-parts at the end of the simulation.

`Extended motion monitoring (0/1):` integer, default 0

- This parameter controls whether extended information regarding the movement of each motion node is written to the log-file, or not. Please refer to the section *Hints for rigid-body motion* for a more in-depth description of this parameter.

`Factor for dynamic Cauchy convergence:` float, default 0.05

- If parameter Use Cauchy convergence control is relative_dynamic or absolute_dynamic, then the number of iterations used for convergence estimation are calculated by means of this factor. The considered iterations are determined by the current inner iteration step times the Factor for dynamic Cauchy convergence. However, at the beginning of the computation this may lead to problems. Therefore the minimum number of considered iterations is given by Number of samples for Cauchy convergence when using the _dynamic option. The maximum number of considered iterations is set to 1000.

`Fd switch uses SA viscosity (0/1):` integer, default 0

- The DDES switch can use either the sum of turbulent eddy plus the laminar eddy viscosities viscosity or the spalart-allmaras viscosity, in the approximation to the log law which is employed to active the delay function in both DDES and IDDES. This is also used in the NTS Low Reynolds number modification.

`Field output description file:` character string, default (none)

- Optional character string giving the parameter file from where a string, marking additional variables to be written to the output file, may be read. Note that the primitive variables do not need to be marked here as these are output in any case (Regardless of the users requirements, these data are required for proper restarts).
  If the filename is the parameter file itself, either the name (with path) of the parameter file has to be set or for simplicity the key '(thisfile)' can be used. The latter is of advantage because a renaming of the parameter is not necessary when renaming the parameter file.

`Field output values:` character string, default (none)

- Optional character string containing the names of the additional variables to be output. The names have to be joined with an underline e.g. *mach_volume_wdist_eddy_cf_yplus*. The valid names are listed in the subsection *Output files*.

`Filter width model:` character string, default Edge

- This parameter allows the user to specify the filter width model to use for LES or DES calculations. At present three (3) options are possible. "Edge": maximum edge length - this is the standard DES length scale choice, "Volume": cube root of volume - this should be the usual choice for LES calculations, "User": user defined length scale.

`Final freeform box filename:` character string, default (none)

- This file contains the deformed boxes used for the freeform deformation. The inputfile has to be setup similar to the Initial freeform box filename as shown in Figure 1.

`Findiff block output format (0/1/2):` integer, default 0

- For "Solver type: Findiff": control the output format of the comparison, factor difference, relative error or absolute error respectively.

`Findiff column global:` integer, default 0

- For "Solver type: Findiff": the column of the Jacobian to compare with finite differences. This corresponds to the global id of a point in the mesh.

`Findiff consider diffflux:` integer, default 1

- For "Solver type: Findiff": consider central dissipation terms in the finite-difference comparison.

`Findiff consider invflux:` integer, default 1

- For "Solver type: Findiff": when using Findiff it is very often useful to examine parts of the discretization individually, this switch turns consideration of the inviscid fluxes (not including dissipation in central schemes) on and off.

`Findiff consider turbsource:` integer, default 1

- For "Solver type: Findiff": consider turbulence source terms in the finite-difference comparison.

`Findiff consider turbviscflux:` integer, default 1

- For "Solver type: Findiff": consider turbulence diffusion terms in the finite-difference comparison.

`Findiff consider viscflux:` integer, default 1

- For "Solver type: Findiff": consider viscous terms in the finite-difference comparison.

`Findiff epsilon:` float, default 1.e-8

- For "Solver type: Findiff": the step-size for the finite-difference approximation of the Jacobian. This choice will determine the accuracy of the approximation.

`Findiff maximum epsilon:` float, default 1.e-2

- For "Solver type: Findiff": as for "Findiff minimum epsilon" the maximum epsilon to consider in the convergence output.

`Findiff minimum epsilon:` float, default 1.e-14

- For "Solver type: Findiff": for examination of the convergence of the finite-difference approximation with various epsilon, the minimum epsilon to consider.

`Findiff row for convergence output:` integer, default -1

- For "Solver type: Findiff": for output of the convergence of the finite-difference approximation in epsilon, examine the block corresponding to this row of the Jacobian (global id), and the column specified in "Findiff column global". This must be a neighbor of that point if output is to be non-zero.

`First wave number [ALPHA]:` float, default 0.3

- Lilo input parameter ALPHA: first wave number α.

`Fix chimera boundaries (0/1):` integer, default 1

- This parameter has an effect only if the grid on which the deformation tool is applied has several chimera grid-blocks. If the default (1) is used chimera boundary points are connected by additional edges to their donor cell in the overlapping grid. This forces the chimera boundaries to obtain the same deformation as the overlap region of the related grid block and as a consequence it is guaranteed that the overlap region remains consistent, i.e. large enough. If the parameter is set to 0 there is no connection between the chimera blocks such that a relative motion between them occurs depending on the surface displacements. This setting is allowed only if all blocks contain a surface boundary for which appropriate displacements are given. Due to the relative motion of the blocks the deformation can be successful in some applications which could not be handled otherwise. However, inconsistent grid overlap regions might result from the deformation, which leads to problems in the solver. This needs to be checked by the user.

`Fix negative values (0/1):` integer, default 0

- Switches on (1) or off (0) additional checks after each change between primitive and conservative variables. In case of a negative density or a density smaller than a user defined Minimal density all variables of that point are set to small positive values. In case of a negative energy or pressure or respective values smaller than Minimal pressure or Minimal energy only this value is fixed. If something is fixed a message is printed out to the stdout. This usually unnecessary fixing gains importance during startup from scratch, especially in calculating supersonic flow fields.

`Fixed RANS distance:` float, default -1

- This is a DES parameter (this parameter makes sense only in Hybrid RANS-LES models). The RANS mode is enforced for all wall distances less than this value. To deactivate the functionality set the value to a negative number. At present the wall distance is set to a negative value for laminar regions. A simple comparison with an arbitrary negative value can then cause the LES region to be incorrectly switched on so that the user must ensure that the parameter value is set to a value which is given by the absolute value of the specified laminar height.

`Flow direction for sharp edges:` array-float, default {1, 0, 0}

- This direction is needed to determine if sharp edges lie on a leading edge or a trailing edge of a configuration.

`FLOWer vortical correction coefficient:` float, default 4.0

- The parameter sets the value of the tuning coefficient for the FLOWer rotation correction model. A default value of 4 is taken from the FLOWer-Code.

`Flux weighting scheme:` character string, default AUSMDV

- Choice of scheme for calculation of inviscid fluxes. Note that for RSM and kω calculations (RANS,URANS), the upwind scheme is used to compute inviscid fluxes for kω and RSM equations. In DES, a mixed scheme is available for all equations. Note that the Central+Upwind are only in development, and are not in the general release.

`Form of scaling denominator:` character string, default Shur

- The SARC vortical correction model nondimensionalizes the curvature correction coefficient using either the "Shur" or "Smirnov" forms for the dimensionless scaling. More detail is included on the section on turbulence modeling, however the default is the "Shur" form whilst users who wish to use SARC with the SST model should choose "Smirnov". The "Shur" form will be used if any other string other than the "Smirnov" string is given.

`Freqdom RHS file:` character string, default (none)

- For "Solver type: Freqdom": field-data file containing the RHS of the frequency-domain equation. This must be produced externally to the solver for a given motion.

`Frequency of instability wave in Hz (CF) [FRQ]:` float, default 500.0

- Lilo input parameter FRQ: frequency of the instability wave in Hertz.

`Frustum max radius:` array-float, default 0

- Definition of cut-out volumes, see also `Number of cut-out volumes`. A number of floats separated by blanks is read from this line. The first defines the radius at point 'max' of volume 1, the second the radius at point 'max' of volume 2 and so on. This parameter together with `Frustum min radius` determines the shape of the volume:

  box: if both radii are less or equal zero

  cylinder: if both radii are greater than zero and have the same value

  frustum: if both radii are greater than zero and have a different value

  If there are less than N values a zero is assumed.

`Frustum min radius:` array-float, default 0

- Definition of cut-out volumes, see also `Number of cut-out volumes`.
  For an explanation of the parameter, see `Frustum max radius`.

`Full multigrid central scheme first-order (0/1):` integer, default 1

- The full multigrid start for a flow simulation can accelerate convergence remarkable. Up to now we used only first order convective spatial discretization for the full multigrid to enhance robustness due to the agglomeration of the coarser grids. In some cases it can be useful to start with second order spatial discretization and this parameter allows by setting it to zero to get for full multigrid second order convective fluxes. This parameter is only valid for central schemes.

`Gas constant gamma:` float, default 1.4

- In perfect gas calculations the dimensionalized value of the gas constant (default: air) is used to restrict the choice of the reference variables (density, pressure and temperature) to meaningful values. → 'Reference density'.

`Gas constant R:` float, default 287

- In perfect gas calculations the dimensionalized value of the gas constant (default: air) is used to restrict the choice of the reference variables (density, pressure and temperature) to meaningful values. → 'Reference density'.

`General ratio mue-t/mue-l:` float, default 0.1

- The turbulent viscosity at farfield/inflow boundaries is set according to the defined ratio of turbulent and laminar viscosity (mue-t/mue-l).

`General turbulent intensity:` float, default 0.001

- The turbulent kinetic energy (in case of using a 2-equation model) at farfield/inflow boundaries is computed from the defined value.

`Geometric conservation law (0/1):` integer, default 1

- If active, the geometric conservation law is ensured for a moving grid and old control volumes do not need to be part of the restart file.

`Global wall roughness:` float, default 0

- Global surface roughness value. The value has to be specified in grid units (→ 'Grid scale'). Notice that this parameter specifies the equivalent sand grain roughness height, and not the geometrical roughness height. Parameter is only active, if parameter "Consider wall roughness" is set to "global".

`GMRes inner iterations:` integer, default 20

- For "Krylov loop: GMRes": the maximum size of the Krylov subspace GMRes creates before a restart. Increasing this parameter may make GMRes faster and more robust at the cost of increased memory requirements. Specifically this number of flow solutions must be stored additionally.

`GMRes preconditioning iterations:` integer, default 10

- For "Krylov loop: GMRes": Number of MG cycles to apply as a preconditioner. Typically one cycle is insufficient. Increasing this value tends to increases stability at the cost of performance.

`Grid metric:` character string, default Cell_Vertex

- The grid metric affects the arrangement of control volumes and update points for the flow variables. The Cell Vertex grid metric associates the flow variables with the cell vertices. The Cell Centered grid metric associates the flow variables with the cell centers.

`Grid prefix:` character string, default dual

- The name prefix (including the path) of the dual grid NetCDF input file(s). Assuming the `Grid prefix` is assigned to `configuration.edgedata` the complete name of an output file is `configuration.edgedata_domain_N_grid_M` with `N` indicating the number of the domain and `M` determining the number of the grid level; level 1 is the fine grid, level 2 is the next coarser grid and so on.

`Grid scale:` float, default not_defined

- Specifies the length of the grid unit in the unit system used for dimensional output (→ 'Reference density'), normally in meters as part of the SI. All other length parameters / coordinates should be specified in grid units. Example: for a grid unit of 1 mm, set Grid scale: 0.001.

`Grid shield model:` character string, default (none)

- The various form of DES are interpreted in terms of the Grid Induced Separation (GIS) shielding. "DES" is standard DES, "DDES" is delayed DES, and "IDDES" is the improved DDES or Wall modeled LES. Thus DDES is DES with the delay function introduced by Travin and others. IDDES is DES with additional changes which allow the LES to penetrate closer to the wall - in effect this is also a form of Wall Modelled LES (WM-LES)

`h-scaling power:` float, default 0.5

- Defines the edge length scaling power, which is available for the differences, gradient and reconstruction based indicators.

`h-scaling reference length:` float, default 0

- Lower edge length limit for using the above length scaling power.

`Half span reference length:` float, default 0.0

- Reference length for calculating non-dimensional η values for the spanwise position of contourlines. Only needed for output purposes.

`Hardwired weighting factor:` float, default -1

- This parameter is used to allow a fixed blending factor between upwind and central schemes in the computation of the inviscid fluxes. Experience from other published in the literature suggests that a value of 0.4 is useful, however this is problem dependent.

`Hellsten vortical correction coefficient:` float, default 3.6

- This parameter is used to modify the default value of the tuning coefficient for the vortical correction model published by A. Hellsten.

`Hold static velocity field (0/1):` integer, default 0

- When set to 1, the parameter disables the update of the momentum field variables, thus the velocity field is held frozen. This is useful when one has a velocity field and it is required to match a scalar field to the velocity. Note that this is not implemented for multigrid or for the Backward-Euler iterative scheme.

`Hole filename:` character string, default (none)

- Start of a hole definition block specifying a path to a NetCDF file containing a hybrid grid for chimera hole definition. From the NetCDF file the element-point connectivity of the volume elements and the coordinates are read. All points of a component grid which are inside the elements of this grid are blanked. At the hole boundary an interpolation zone of two edge layers is constructed with an advancing front algorithm. To get the motion information the hole definition grid is associated with a chimera block. The hole definition block ends with the keyword "hole end".

`Hybrid switch model:` character string, default (none)

- Use an alternative length scale for the hybrid switch (RANS/LES). The default setting is the LES filter width definition, which means that the same length scales are used in the LES/RANS switch as for the LES filter width. The facility is included as it has been found useful in particular grid configurations whereby the switching length scale based on the proper LES filter width definition, allowed penetration of the LES region into the upper part of the boundary layer. As usual, there is no substitute for a good mesh and this switch can only be considered as a partial fix for a poor mesh.

`Implicit overrelaxation beta:` float, default 1.0

- If `Backward_Euler` time stepping is used, this parameter directly modifies the diagonal dominance of the system matrix in the implicit scheme. For experts only.

`Implicit overrelaxation omega:` float, default 1.0

- If `Backward_Euler` with the `Lusgs` or `Sgs` schemes are used, this parameter acts as a trade off between convergence and stability . The default value of 1.0 should provide almost-best convergence and good stability for the majority of cases. If your case diverges however, try increasing the value of this parameter. In any case it should not be pushed above about 2.0. If you're keen for speed reduce to a minimum of about 0.7.

`Increase memory (0/1):` integer, default 0

- If this parameter is active (1) more memory is allocated, allowing alternative high-memory routines to be used which consume less CPU-time. Those routines are available for the Barth Jesperson limiter used for upwind calculations, for the Laplace smoother used for explicit smoothing and for limitation of gradients when interpolating multigrid corrections. The gain of CPU-time and the amount of additional memory needed depend on the parameters and their combinations (range: 0-25%).

`Indicator0 Ht-scaling:` float, default 1

- Weight of Ht for the differences based indicator.

`Indicator0 Pt-scaling:` float, default 1

- Weight of Pt for the differences based indicator.

`Indicator0 rho-scaling:` float, default 1

- Weight of rho for the differences based indicator.

`Indicator0 V-scaling:` float, default 1

- Weight of V for the differences based indicator.

`Indicator type:` character string, default diff

- Employing one of the following indicators: diff, grad, recon, sum, all, del, vortex, adjoint. For a detailed description of the indicators see also section adaptation in the TAU user guide.

`Indicator user-scaling:` array-float, default 1.0

- Weight for the specified solution values given by 'Indicator user-values'. Example: For the specified solution values *Indicator user-values: mach_eddy_Ht_Ptot* one can use for an equal weight of all values this construct *Indicator user-scaling: 1.0 1.0 1.0 1.0*. If the user would like to mix the values not equally weighted, than one may choose this construct *Indicator user-scaling: 0.5 2.0 0.8 1.2*.

`Indicator user-values:` character string, default (none)

- Employing specified solution variables for the chosen Indicator type. The names have to be joined with an underline e.g. *mach_eddy_Ht_Ptot*. The valid names are listed in Table 1.

`Init total conditions (0/1):` integer, default 0

- If a computation is started from scratch the initial solution is initialized with free stream values (0) or with total conditions (1).

`Initial freeform box filename:` character string, default (none)

- This file contains the nondeformed boxes used for freeform deformation. With defined boxes the parameterization of the surface points are performed. The inputfile has to be setup as shown in Figure 1

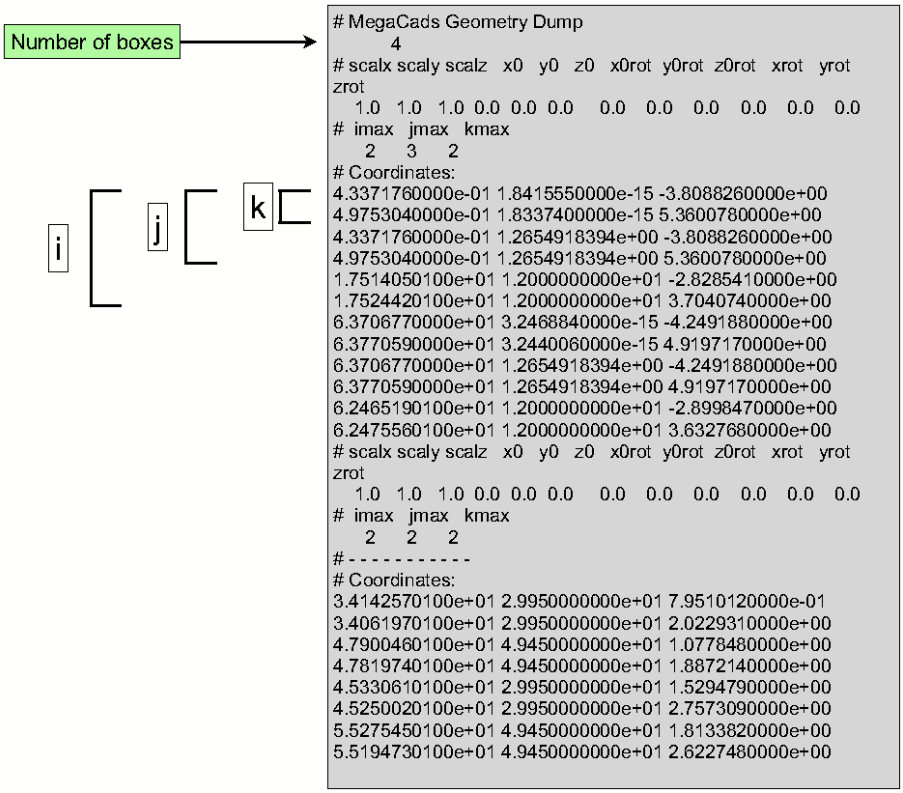


Figure 1: Prototype of a freeform box file with the appropriate i,j,k ordering.

. The box coordinates are setup in a structured manner.

`Initialize deformation (0/1):` integer, default 0

- For starting a coupled unsteady computation, a prior steady computation with this flag turned to 1 is needed to write extra data to the output files.

`Interpolate prescription values (0/1):` integer, default 1

- If prescription values are in Array-format (Prescription input data structure) the transition locations can be interpolated.

`Inverse 4th order dissipation coefficient:` float, default 64

- Inverse fourth order dissipation factor used if the central scheme with scalar and matrix dissipation is employed.

`Inviscid flux discretization type:` character string, default Upwind

- Formerly: "Convective RANS flux discretization type". This parameter selects the discretization type of the convective (or "inviscid") fluxes of the RANS equations. Possible choices are:

- Central: Central scheme, 2nd order.
- Upwind (default): Upwind scheme (to be specified via parameter "Upwind flux").

Note, that in case of a central scheme, the discretization scheme of the convective fluxes of the turbulence equations has to specified by "Central convective meanflow flux". In case of using an upwind scheme for the RANS equations, the same upwind scheme will be used for the turbulence equations, where the accuracy order for the turbulence equations can be specified by the parameter "Order of additional equations (1-2)".

`Jacobian constant dissipation coeffs (0/1):` integer, default 1

- For the discrete linear solver types: when constructing the Jacobian make the simplifying assumption that the coefficients of central dissipation are constant with respect to the linearization operation.

`Jacobian constant laminar viscosity (0/1):` integer, default 1

- For the discrete linear solver types: when constructing the Jacobian make the simplifying assumption that the laminar viscosity is constant with respect to the linearization operation.

`Jacobian frozen turbulence (0/1):` integer, default 1

- For the discrete linear solver types: when constructing the Jacobian make the simplifying assumption that the turbulence is constant with respect to the linearization operation. This is a much larger approximation than that made by "Jacobian constant dissipation coeffs (0/1)", or "Jacobian constant laminar viscosity (0/1)", and can result in substantial errors in the linear solution (and therefore gradients in adjoint mode). However solving the linear system with linearized turbulence is mostly only possible with the PETSc solvers.

`Jacobian includes timestep (0/1):` integer, default 0

- For the discrete linear solver types: make a contribution of the reciprocal of the timestep (based on the given CFL number) to the diagonal entries of the Jacobian.

`Jacobian turb. includes timestep (0/1):` integer, default 0

- As for "Jacobian includes timestep (0/1)" but only modify the diagonals of the Jacobian of the turbulence equations.

`Jacobian variables:` character string, default Cons

- For the discrete linear solver types: the variables with respect to which the Jacobian is obtained, conservative-conservative variables or conservative-primitive variables. For DAdjoint the resulting adjoint solution is identical. For Primal and Freqdom this choice affects the variables in which the field output is given.

`Jacobian volume scaling (0/1):` integer, default 0

- For the discrete linear solver types: scale the rows of the Jacobian by the local cell volume. This may improve the conditioning of the linear system under some circumstances. This parameter should have no effect on the final linear solution.

`K-omega limitation type:` character string, default standard

- Formerly "K-omega limitation type (0/1/2)".
  This switch controls the limitation of k and ω in kω turbulence models.
  - value = standard: Lower and upper limitation (default)
  - value = Rudnik: Only lower limitation according to Rudnik (FLOWer-Code). The lower limits are defined by parameters `Minimum k` and `Minimum omega`.
  - value = Schwarz: Lower limitation of k as for "Rudnik", lower limitation of ω derived from Schwarz inequality for the Reynolds stresses.

`K-omega wall factor:` float, default 10

- Modification to wall value of dissipation to ensure proper sublayer behavior.

`Kato Launder modification factor:` float, default 0

- This switch allows the user to modify the production term of the k equation to account for over-prediction of stress in stagnation regions, more precisely: The Kato-Launder modification can be used together with most two-equation models that have a production term formulated as above. The modification was originally developed for transient simulations of vortex-shedding behind square cylinders, where the normal k-epsilon model tends to produce too much turbulent energy in stagnation regions and in the small regions with strong acceleration and deceleration around the square corners. This over-production creates too much turbulent viscosity which in turns affects the vortex-shedding and the development of the vortex-street downstream of the square cylinder. With the modified production term Kato and Launder were able to produce much better results. The Kato-Launder modification has also been popular in the turbomachinery field, where the stagnation-point-problem of two-equation models can lead to significant errors. In these applications it is common with regions with very high acceleration and deceleration (leading edges, shocks, suction-side peaks, ...) where the Kato-Launder modification can improve the results with a two-equation model which otherwise predicts too much turbulence. Lately the popularity of the Kato-Launder modification has decreased though. Instead people are using more modern models that inherently have limiters and realizability constraints that reduce the stagnation-point problem. In pure shear-flows like boundary-layers and wakes the Kato-Launder modified production term will give exactly the same result as the unmodified production

term. However, outside of boundary-layers and wakes the Kato-Launder modified production term will give very different results. Essentially what it does is to turn off the turbulent production outside of boundary-layers and wakes. This has the good effect that the over-production of turbulent energy in stagnation regions and regions with very strong acceleration is eliminated. The downside is that turning off the turbulent production is not exactly accurate either. For example, the Kato-Launder modification will give too low turbulence levels in stagnation regions and this in turn will affect heat-transfer and skin-friction around the stagnation point.

`Keep Coco auxiliary files (0/1):` integer, default 1

- Save files containing some additional output from Coco.

`Keep Coco log files (0/1):` integer, default 1

- Save the log files from Coco.

`Keep Coco profiles files (0/1):` integer, default 0

- Save files containing the Coco velocity profiles in Tecplot format.

`Keep Coco run files (0/1):` integer, default 0

- Save all files needed to run Coco manually.

`Keep files from pre-mode (0/1):` integer, default 0

- Switch to decide if the output files from the pre-prediction phase will be saved.

`Keep Lilo auxiliary files (0/1):` integer, default 1

- Save files containing some additional output from Lilo.

`Keep Lilo log files (0/1):` integer, default 1

- Save the log files from Lilo.

`Keep Lilo run files (0/1):` integer, default 0

- Save all files needed to run Lilo manually.

`Keep N-factor files (0/1):` integer, default 1

- Save files containing the N-factors.

`Keep Prepcp files (0/1):` integer, default 1

- Save all Prepcp files.

`Kozlov modification:` integer, default 0

- Activate dynamic determination of SARC model coefficients using algorithm by Kozlov et al.

`Krylov loop:` character string, default (none)

- For all linear solver types: the Krylov method to use to stabilize the iteration. For "(none)" standard linear MG cycles with a Runge_Kutta or Backward_Euler smoother are applied. "GMRes" uses the same MG cycle as a preconditioner for a GMRes iteration. "RPM", which is not available in parallel, and only when TAU has been compiled with the LAPACK library, uses the Recursive Projection Method, which can be a useful alternative to GMRes when little memory is available. "EvalsDirect" is primarily a development option, which solves the eigenproblem corresponding to the MG preconditioned linear operator directly. This requires LAPACK, sequential operation, and only works for very small problems. "PETSc" uses any Krylov solver from the PETSc linear solver library preconditioned with MG, works in parallel but requires TAU binaries linked with the PETSc library.

`Leading edge sweep angle:` float, default 0

- Prescription of the leading edge sweep angle if Set leading edge sweep angles is set to 1.

`Lilo executable:` character string, default ./lilo

- Path to Lilo executable -> ./lilo

`Lim.  angle to detect sharp edges (deg):` float, default 30

- In degrees! This angle is used to detect sharp edges of the surface grid.

`Limiter freezing convergence:` float, default 0

- If an upwind scheme is used and the prescribed value is larger than zero the limiter is frozen when the solution converges. If the value is zero the freezing is switched off. A value of 3.5 means that the freezing begins after the solution is converged about 2.5 orders, that the freezing is employed to 50% after a convergence of 3.5 orders and that the limiters are frozen completely after a convergence of 4.5 orders. This technique is used to avoid limiter oscillating. Some caution is required when using limiter freezing. Too small values can lead to a wrong solution, which has been observed a few times. Thus we recommend to compute the solution with one or several restarts, which allow the frozen limiters to change, because the freezing coefficients are not stored. This is a safer procedure.

`Linear residual type:` character string, default Facemat

- For the discrete linear solver types: the manner in which the Jacobian is to be constructed. The only option is Facemat except in the case where TAU has been linked with the PETSc linear solver library, in which case the full Jacobian may be constructed explicitly in memory.

`Linear restart-data prefix:` character string, default (none)

- For the linear solver types: a linear solution to use as a restart. The flow restart-data solution must always be supplied in addition, whether or not a linear restart is specified.

`Linear solver:` character string, default Lusgs

- If `Backward_Euler` is specified in 'Relaxation solver', this switch determines how the linear system will be formed and solved. Each of the available solvers (`Lusgs`, `Sgs`, `Gmres`), has its own distinctive characteristics and result in completely different convergence histories and efficiencies. If you are unsure what to use, then use `Lusgs`. `Sgs` and `Gmres` are only useful in very particular circumstances, and `Gmres` is only available in the `DEVELOPMENT` code.

`Low Re model:` character string, default NTS

- Select function to deactivate low Reynolds terms of the DES models. "NTS" is the correction developed by Travin et al, while "Breuer" is the correction developed by Breuer et al. The NTS model is more general. At present , while this can be enabled for all RANS models in DES model, it is only active for the SA DES variants since the two-equation model implementations require additional work which is not completed as of 10.09.09.

`Lowest pressure for 2nd order:` float, default 0

- If the pressure is lower than the given fraction of the reference pressure then the limiter values for all variables are diminished to zero. Therefore, if the local pressure is lower than half of the given value the upwind solver uses only a first order reconstruction. This parameter can be used to increase the robustness of the solver in regions of strong flow expansion. The given value is expected to be in dimensionless form.

`Lusgs increased parallel communication (0/1):` integer, default 0

- Add an extra level of parallel communication within the LU-SGS sweeps. If the number of points per domain is very small this may help convergence.

`Lusgs treat whirl implicitly (0/1):` integer, default 0

- Consider the effect of the whirl fluxes in the Jacobian of the implicit treatment. This may help convergence in cases where these fluxes play a significant role in the flow.

`Mach number limit for limiter:` float, default 0

- Using an upwind scheme the limiter sometimes reduces the order of the reconstruction in regions where it is not necessary. Especially in a separating boundary layers this can lead to a bad calculation of the separation length. To suppress the limitation this parameter provides a Mach number. If the velocity normal to a surface forms a Mach number smaller than the given value the limiter is switched off. Up to twice the value the limiter is linearly increased to it's original value. Therefore, the default or any negative value of this parameter leaves the limitation unchanged.

`Matrix dissipation terms coefficient:` float, default 1.0

- Scaling factor for the dissipation if the central scheme with matrix dissipation is used. This factor acts as a global scaling to increase and decrease the dissipation. A value greater than 1.0 will add to each flux vector component more artificial dissipation than computed with the artificial dissipation algorithm and vice versa. This parameter should be handled with care and it is advised to keep the default value of 1.0.

`Max. distance for wallnormals:` float, default 1e+20

- In grid units! Maximum distance of the normals to be followed into the field. (Should be larger than the thickness of the structured grid part and of course larger than the maximum thickness of the laminar boundary layer!)

`Max. number of points on wallnormal:` integer, default 128

- Maximum number of points normal to wall used in the structured part of the grid for the computation of the laminar boundary layer parameters

`Maxima x-direction:` array-float, default 0

- Definition of cut-out volumes, see also `Number of cut-out volumes`. A number of floats separated by blanks is read from this line. The first defines the maximum x-value of box 1, the second the maximum x-value of box 2 and so on. If there are less than N values a zero is assumed.

`Maxima y-direction:` array-float, default 0

- Definition of cut-out volumes, see also `Number of cut-out volumes`. A number of floats separated by blanks is read from this line. The first defines the maximum y-value of box 1, the second the maximum y-value of box 2 and so on. If there are less than N values a zero is assumed.

`Maxima z-direction:` array-float, default 0

- Definition of cut-out volumes, see also `Number of cut-out volumes`. A number of floats separated by blanks is read from this line. The first defines the maximum z-value of box 1, the second the maximum z-value of box 2 and so on. If there are less than N values a zero is assumed.

`Maximal point movement factor:` float, default 0.2

- The value specifies the maximal allowed movement factor of surface points in relation to the scale of the attached elements. Therefore smaller values of this parameter will yield to less splined surface points. With this parameter the user can prevent degeneration of the attached tetras.

`Maximal time step number:` integer, default not_defined

- The solver stops if the current time step number is equal the prescribed number. The `Maximal time step number` criterion has no effect if a Minimum residual criterion is fulfilled or if a CPU-time limit on a batch computer is reached before. In the latter case the solver stops (and writes out the solution) if not enough CPU time remains for the next time step. If the `Maximal time step number` is set to 0 no iteration is performed, but output is written. This allows to overwrite the existing restart file by a new one which contains (more or less) data, according to the parameter settings

`Maximal time step number (coarse grids):` integer, default 1

- In full multigrid mode (Multigrid start level > 1) this parameter allows the number of time steps on the coarse grid levels to be fixed.

`Maximum delta cp for pmin/pmax search:` float, default 0.5

- Size of the allowed maximum jump in the pressure distribution.

`Maximum delta for transition:` float, default 1e+20

- A new transition point is determined as follows: xtr = xtr,old + MIN(maxdelta, (xtr,new - xtr,old)*relax). If no transition is found: xtr = xsep. xsep is the laminar separation point from the laminar boundary-layer code COCO if available, else for xsep the laminar separation point from TAU is used.

`Maximum limit mue-t/mue-l:` float, default 20000

- The ratio of the eddy viscosity over the laminar viscosity (mue-t/mue-l) defines an upper bound for the eddy viscosity in order to limit unrealistic large values (peaks).

`Maximum point number:` integer, default -1

- This parameter limits the number of grid points which can be reached because of repeated adaptations. If the value is larger than the current point number then the new grid point number will not exceed the specified value. If it is smaller then the adaptation will reduce the grid point number and try to meet the requirement.

`Maximum point number per partition:` integer, default -1

- This parameter limits the number of grid points which can be reached in one partition. The parameter can be useful to limit the usage of resources on a machine. Locally, the parameter works similar to parameter Maximum point number. If the edge refinement step of adaptation tries to mark to many edges this parameter causes a higher indicator limit in this partition. So in very special cases, the equidistribution of error will be disturbed until the next partition and adaptation.

`Maximum refinement level:` integer, default -1

- This parameter limits the number of refinement levels. The default value of -1 means the refinement level is not recognized. For a level of $l > 0$ the indicator will not mark any edges of a refinement level of l or above for refinement. So the feature is a weak limitation, a finer grid will not actively be derefined. But no additional refinements exceeding the maximum level are done.

`Maximum surface angle (degree):` float, default 30

- A value of zero switches off the splining of the surfaces (linear reconstruction everywhere). If the value is larger than zero the splining is switched on. If bar-elements are in the primary grid file the exact value has no further meaning. All the bar elements are considered as parts of surface edges. If those elements are not in the grid file (use `ncdump -h grid-file` to check it) the value defines an angle criteria to compute surface edges. If the normal vectors of neighboring surface elements have a larger angle in between the edge is considered as a part of a surface edge. Note that small angles lead to the detection of undesired surface edges. Values smaller than 30 degrees are not recommended (however, it depends on the configuration).

`Maximum turbulence production/destruction:` float, default 20

- The parameter represents the maximum ratio of k-production to k-destruction in kω turbulence models (not active for Spalart-Allmaras type and Reynolds stress models). Low values of this parameter may stabilize kω computations.
  Recommendation: For obtaining useful results, a value in the range between 10 and 20 should be specified (RAE 2822, case 10; L1T2); for complex geometries, a value of 5 may be necessary. Smaller values are not recommended.

`Maxsize for the coarse graph for MGridGen:` integer, default 4

- An upper bound on the cell size of the coarse graph. This parameter can only be used if the MGridGen program is linked to the preprocessing.

`Metis parameter CoarsenTo (int):` integer, default 100

- The number of points to coarsen down the graph, when using the 'metis' grid partitioner (inactive otherwise). For large grids, small values (e.g. 100) can become very time consuming.

`Metis parameter IPtype (int):` integer, default 4

- The ID of IPtype partitioning, when using the 'metis' grid partitioner (inactive otherwise).

`Metis parameter Mtype (int):` integer, default 21

- The ID of Mtype partitioning, when using the 'metis' grid partitioner (inactive otherwise).

`Metis parameter Rtype (int):` integer, default 13

- The ID of Rtype partitioning, when using the 'metis' grid partitioner (inactive otherwise).

`MG description filename:` character string, default 4w

- The name (including the path) of the ASCII input file defining the multigrid cycle. A sample file for a 2-level V-cycle contains the following numbers (in one or several lines):
  0 1 0 -1
  with:
  0: a relaxation step is performed on the current grid level
  1: the algorithm changes to the next coarser level
  -1: the algorithm changes to the next finer level
  (A relaxation step is ether a Runge-Kutta cycle or a backward Euler step.) The restriction in defining a MG-cycle is that the sum of all numbers has to be zero. Any number of multigrid levels can be defined up to the `number of domains` computed in the preprocessing program. Standard multigrid cycles do not have to be defined in an input file. If, instead of a filename, one of the strings `sg, 2v, 3v, 3w, 3w+, 3w++, 4v, 4w, 5v, 5w` is defined a predefined V- or W-cycle is performed. `3w++` is a cycle that makes quite a lot smoothing steps on the 3rd level and reaches a convergence rate which is in many cases even better than those of a `4w` cycle by approximately the same costs, which is at least true in parallel mode. Note that in parallel mode this can be of advantage because the lowest level contain more grid points which increase the parallel efficiency. `3w+` is a compromise of this which is still much better in terms of the convergence rate than a `3w` cycle but only almost as good as `4w` cycle. It is recommended if `3w++` or `4w` is not robust enough. The `sg` defines the single grid mode. Default is `4w`.

`Min. number of points on wallnormal:` integer, default 32

- Minimum number of points normal to wall used in the structured part of the grid for the computation of the laminar boundary layer parameters.

`Minima x-direction:` array-float, default 0

- Definition of cut-out volumes, see also `Number of cut-out volumes`. A number of floats separated by blanks is read from this line. The first defines the minimum x-value of box 1, the second the minimum x-value of box 2 and so on. If there are less than N values a zero is assumed.

`Minima y-direction:` array-float, default 0

- Definition of cut-out volumes, see also `Number of cut-out volumes`. A number of floats separated by blanks is read from this line. The first defines the minimum y-value of box 1, the second the minimum y-value of box 2 and so on. If there are less than N values a zero is assumed.

`Minima z-direction:` array-float, default 0

- Definition of cut-out volumes, see also `Number of cut-out volumes`. A number of floats separated by blanks is read from this line. The first defines the minimum z-value of box 1, the second the minimum z-value of box 2 and so on. If there are less than N values a zero is assumed.

`Minimal density:` float, default 1e-12

- Lower limit of density during iteration procedure. This parameter is only active, if parameter Fix negative values (0/1) is set to 1 (one).

`Minimal energy:` float, default 1e-12

- Lower limit of internal energy during iteration procedure. This parameter is only active, if parameter Fix negative values (0/1) is set to 1 (one).

`Minimal pressure:` float, default 1e-12

- Lower limit of pressure during iteration procedure. This parameter is only active, if parameter Fix negative values (0/1) is set to 1 (one).

`Minimum artificial dissipation for acoustic waves:` float, default 0.2

- This parameter is used for switching between the matrix and scalar dissipation scheme. Setting the value to 1.0 introduces the scalar dissipation scheme, whereas the value 0.0 gives complete matrix dissipation. For example, together with the parameter Minimum artificial dissipation for velocity and setting both values to 0.3 will act with 30 % scalar dissipation and 70 % matrix dissipation. Additionally, it prevents the matrix dissipation from becoming singular at regions where the velocity is equal to the speed of sound.

`Minimum artificial dissipation for velocity:` float, default 0.2

- This parameter is used again for switching between the matrix and scalar dissipation scheme. Setting the value to 1.0 introduces the scalar dissipation scheme, whereas the value 0.0 gives complete matrix dissipation. Additionally, it prevents the matrix dissipation from becoming singular at stagnation points were the velocity get close to zero. It is advised to use at least a small number (∼0.05) even for smooth flows.

`Minimum edge length:` float, default 1.e-12

- Edges shorter than this value are not marked for bisection by the refinement indicators. Note that due to side effects it is possible that those edges are refined anyway. The resulting minimum edge length can be one order less than prescribed by the value.

`Minimum k:` float, default 1e-05

- Lower limit of k in kω turbulence models, expressed as fraction of the far field value k_∞. This parameter is only active, if parameter K-omega limitation type is set to 1 (one).

`Minimum N-factor for extrapolation (CF):` float, default 0.001

- N-Factor envelope can be extrapolated if N > Nlimit (Nlimit: Minimum N-factor).

`Minimum N-factor for extrapolation (TS):` float, default 0.001

- N-Factor envelope can be extrapolated if N > Nlimit (Nlimit: Minimum N-factor).

`Minimum number of inner iterations per time step:` integer, default 0

- Minimum number of pseudo time steps per physical time step. It is possible that the residual starts to rise before the maximum number of iterations is completed. This parameter is a quick way to limit this behavior. Note that the user needs to be aware that this is happening by printing out the residuals on each inner iteration (only with dual-time stepping).

`Minimum omega:` float, default 1e-05

- Lower limit of ω in kω turbulence models, expressed as fraction of the far field value $\omega_{-\infty}$. This parameter is only active, if parameter K-omega limitation type is set to 1 (one).

`Minimum residual:` float, default 1e-16

- The solver stops if the current residual is less than the initial residual times the prescribed number.

`Minimum residual (coarse grids):` float, default 0.001

- Only active in 'full multigrid' mode. The solver stops the computation on a coarse multigrid level if the current residual is less than the initial residual times the prescribed number. The computation is automatically restarted on the next finer level.

`Minsize for the coarse graph for MGridGen:` integer, default 1

- An upper bound on the cell size of the coarse graph. This parameter can only be used if the MGridGen program is linked to the preprocessing.

`Mismatch of overlapping walls:` float, default 0.0

- Additional tolerance for Chimera wall projection method (see parameter `Apply Chimera wall projection (0/1)`). To be set if the projection method fails due to a misalignment of the overlapping surface grids. The tolerance is given in grid units.
  Caution: Use with care, since too large tolerance may cause malfunction of the projection method.

`Mixed inviscid fluxes (0/1):` integer, default 0

- Development only: mixed central-upwind inviscid flux activation

`Modal amplitude factor:` array-float, default 1

- For each mode in the modal amplitude file this parameter specifies a real factor with which that mode should be multiplied before being applied to the surface deflection. Thus production of a new amplitude file for each scaling of a deflection is rendered unnecessary.

`Modal perform maximum deflection (0/1):` integer, default 0

- Under its default settings, the modal deformation input of the deformation tool calculates a timestep based on the frequencies specified by the parameter "Modal reduced frequency", updates the global time with this timestep, and then computes the surface deflection and hence the deformation that would result at this time. On the other hand if this parameter takes the value 1, simply the point deflection specified by the first mode in the amplitude file is applied to the surface without modification.

`Modal reduced frequency:` array-float, default 0.0

- The reduced frequencies corresponding to the given modes in the Amplitude file.

`Modal reference length:` array-float, default 0.0

- The reference lengths corresponding to the given modes (in grid units).

`Modal steps per period:` integer, default 0

- The number of steps per period; applied universally for all frequencies/modes.

`Modify cp for Coco input:` integer, default 1

- Switch for the activation of simple pressure modification for the input of the boundary layer code Coco to account for the (effective) sweep angle (UNRESOLVED REFERENCE(use_phile_phigeo_to_modify_cp)). The theoretical stagnation pressure is calculated and the value from the Tau solution is altered to fit to this value.
  - 0: no pressure modification,
  - 1: pressure is modified relative to the position between stagnation point and last point (UNRESOLVED REFERENCE(use_cpmin_max_as_reference_for_modified_cp)). The shift in pressure is maximum at stagnation point and decreases to 0 at last point (and beyond),
  - 2: constant shift in pressure distribution.

`Monitor history (0/1):` integer, default 0

- Turns on/off the print out of the Monitoring values in Tecplot ASCII format in an extra file. Whenever a restart is done and the Output files prefix is not changed the output of the values will be continued in the previous monitoring file. The monitoring files are named as *prefix.monitoring.pval.dat* and *prefix.unsteady.monitoring.pval.dat* in a steady and unsteady computation. If a restart is made with the turned on monitoring history the monitoring values have to be fixed over the whole computation.

`Monitoring significant figures:` character string, default 4_4_4_4

- Specifies the number of significant figures that should be printed for each variable specified in Monitoring values in the monitoring (stdout) output. The form of the parameter is *4_8_4_3*, where the ordering corresponds to the ordering of the variables specified in Monitoring values. It is not possible to set significant figures for variables that are not given explicitly there. Too many or too few numbers may be given, in which case the extra numbers will be ignored/the missing ones replaced with 4. Note that Tecplot monitoring output provides machine accurate output for all variables all the time; activate with Monitor history (0/1).

`Monitoring values:` character string, default (none)

- Character string containing the names of the quantities to be monitored during execution of the solver. The names are separated by underline, e.g. *Residual_Max-res_C-lift_C-drag*. The available names with a short explanation are listed by the solver in the standard output and are listed in Table 3.

`Motion hierarchy filename:` character string, default (none)

- The name (including the path) of the ASCII input file defining the hierarchy-model for rigid-body movement. The format of the file is the same as for the boundary-mapping

file, with the keywords 'hdf end' to mark the end of a model node.
If the data is defined inside the parameter file itself, either the name (with path) of the parameter file has to be set or for simplicity the key '(thisfile)' can be used. The latter is of advantage because a renaming of the parameter is not necessary when renaming the parameter file.

`Multigrid indicator (0/1):` integer, default 0

- Multigrid as well as residual and correction smoothing are working well for continuous solutions. In the vicinity of discontinuities especially strong shocks problems occur with these acceleration techniques. If this parameter is switched on, an indicator is calculated to detect potentially problematic volumes. The indicator values are used to prohibit multigrid and smoothing locally. Therefore, this parameter makes the old parameter 'No smoothing near shocks (0/1)' superfluous and perhaps the parameter CFL number (large grad p) can be avoided in future as well.

`Multigrid start level:` integer, default 1

- Employing 'full multigrid' mode to start on the prescribed grid level.

`N-factor extrapolation mode:` integer, default 2

- N-factor envelope extrapolation mode:
  - 1: extrapolation at maximum slope
  - 2: extrapolation at maximum slope in the vicinity of cf,min
  - 3: extrapolation at maximum slope in the vicinity of cf,min, extend of this region can be altered by Cf-min offset for extrapolation mode 3
  - 11: same as 1
  - 12: same as 2
  - 13: same as 3

  Important: when using the boundary layer method extrapolation is switched off automatically for modes 1-3. To force extrapolation in this case use mode 11-13!

`N-ts/N-cf Diagram (N-CF):` array-float, default -

- Second value of the coordinate pairs specifying the points of the polygonal line which approximates the stability boundary. The number of values to be given here is specified by 'N-ts/N-cf Diagram (points)'!

`N-ts/N-cf Diagram (N-TS):` array-float, default -

- First value of the coordinate pairs specifying the points of the polygonal line which approximates the stability boundary. The number of values to be given here is specified by 'N-ts/N-cf Diagram (points)'!

`N-ts/N-cf Diagram (points):` integer, default 0

- 0=Not active. n $\geq$ 3= active (minimum: 3). Number of points to specify a polygonal line which approximates the stability boundary. Specifies the number of coordinate pairs which have to be given with the input parameters 'N-ts/N-cf Diagram (N-TS)' and 'N-ts/N-cf Diagram (N-CF)'. Note: If both the two parameters 'Critical N-factor TS' and 'Critical N-factor CF' and the parameter 'N-ts/N-cf Diagram (points)' are set in the input file and a set of coordinate pairs is specified, the N-ts/N-cf stability-diagram is applied, not the two parameters 'Critical N-factor TS' and 'Critical N-factor CF'.

`Neglect 2/3 rho k term in k and omega production (0/1):` integer, default 1

- Modification to production in k-equations. Using the default (1) the traceless mean strain rate tensor is used in the computation of the term modelling turbulence production by the mean shear stress. The production term is written as: production = muet * strain * strain - 2/3 * rh * k * vdif, where vdif is the trace of the velocity gradient tensor. Setting the parameter to 0 models the production term as: production = muet * strain * strain. which guarantees a positive sign for the production and can increase robustness.

`New primary grid prefix:` character string, default (none)

- The prefix (including the path) of the new grid file name. If the default is used a new name is built automatically.

`New restart-data prefix:` character string, default (none)

- The prefix (including the path) of the new restart-data file name.

`NLR vortical correction - neglect 2/3 rho k term in omega production (0/1):` integer, default 0

- 0: Disable the dilatation term of the production term for k-w models. 1: Retain the traceless form pf the velocity gradient tensor in the formulation of the production term.

`NLR vortical correction coefficient:` float, input required

- Scaling coefficient for NLR vortical flow correction model.

`Number of cut-out volumes:` integer, default 0

- The refinement of edges is performed per default in the whole domain. A set of boxes, cylinders and frustums can be defined to limit the refinement. For a positive or negative number, refinement is performed outside or inside these defined volumes. If the number differs from zero, the volumes have to be defined using the parameters `Minima x-direction`, `Maxima x-direction`, `Minima y-direction`, `Maxima y-direction`, `Minima z-direction`, `Maxima z-direction`, `Frustum min radius` and `Frustum max radius`.

`Number of domains:` integer, default 1

- Number of domains the dual grid is partitioned to when using the preprocessor in sequential mode. In parallel mode this parameter is not active. In this case the number of partitions equals the number of processes.

`Number of frequencies/wavelengths [NWAV]:` integer, default 40

- Lilo input parameter NWAV: umber of waves to be considered.

`Number of loops for task 30:` integer, default 3

- Number of consecutive executions of task 30 of Lilo. Task 30 will be executed until an amplified mode is found, the frequency reaches 0 Hz of the loop number is exceeded. Initial frequency is set by UNRESOLVED REFERENCE(frequency_of_instability_wave_in_hz), the delta is set by Delta frequency for task 30 loop.

`Number of modes in file:` integer, default 0

- Total number of modes contained in the 'Amplitude filename' file (a sequence of AX, AY, AZ makes up a mode).

`Number of modes to be used:` integer, default 0

- Number of modes contained in the 'Amplitude filename' file which are to be used for calculating the modal deformation.

`Number of multigrid levels:` integer, default 5

- The number of multigrid levels to be computed and written. Note that the flow solver can be run with any multigrid cycle that uses at most this number of levels.

`Number of planes:` integer, default 1

- Integer value defining the number of planes. Note for 2D grids output is done for maximal one cut plane, which corresponds to the 2D grid plane independent of other settings. A zero value disables plane output.

`Number of points for global step (CF) [NPGLOB]:` integer, default 40

- Lilo input parameter NPGLOB: number of grid points for the global step.

`Number of points for global step (TS) [NPGLOB]:` integer, default 40

- Lilo input parameter NPGLOB: number of grid points for the global step.

`Number of points for local step (CF) [NPLOC]:` integer, default 100

- Lilo input parameter NPLOC: number of grid points for the local step.

`Number of points for local step (TS) [NPLOC]:` integer, default 100

- Lilo input parameter NWAV: number of waves considered.

`Number of prescription values:` integer, default -

- Number of value pairs to be used for transition prescription. This number has to match the number of values (alpha/iter/time, x-transition location) which follows in the input.

`Number of primary grid domains:` integer, default 1

- To be set equal to the number of domains the partitioner has to compute, i.e. the number of processes for following parallel programs. This parameter is only active if the partitioner is employed in sequential mode. In parallel mode of the partitioner the target number of partitions is set equal to number of MPI-processes in use.

`Number of profiles:` integer, default 0

- Set the number of points plus the number of line profiles required. This is only used for checking purposes in the code but it must be set. The profiling option is only valid for Cartesian meshes. It is assumed the reader understands that a Cartesian mesh can still be stored in an unstructured format.

`Number of RANS cut-out boxes:` integer, default 0

- This parameter sets the number of cut-out boxes within which only the RANS mode of a hybrid LES/RANS method will be active. At the present time the forcing of the RANS mode is done only inside the RANS cut-out box. For a more detailed section see the LES/DES section in this userguide.

`Number of Runge-Kutta stages:` integer, default 3

- Formerly "Number Runge-Kutta steps".
  Number of the Runge-Kutta stages per iteration. This parameter is only active, if parameter "Relaxation solver" is set to "Runge_Kutta".

`Number of samples for Cauchy convergence:` integer, default 20

- If parameter Use Cauchy convergence control is activated, then this parameter gives the number of previous inner iterations considered to check whether Cauchy convergence of the integral coefficients lift and drag has been reached. For unsteady computations in dual time mode we recommend a value of at least 10 whereas for steady state computations we recommend a value of at least 50. When the option relative_dynamic or absolute_dynamic for parameter Use Cauchy convergence control is chosen, Number of samples for Cauchy convergence is the minimum number of iterations considered for convergence control. In either case, the maximum number of considered iterations is 1000.

`Number of smoothing steps for sas correction:` integer, default 0

- This parameter sets the number of steps used by the Laplacian smoother used to smooth the computed sas correction factor field. Setting this parameter to 0 disables smoothing and is recommended as default.

`Number of smoothing steps for vortical correction:` integer, default 0

- This parameter sets the number of steps used by the Laplacian smoother used to smooth the computed vortical correction factor field. Setting this parameter to 0 disables smoothing and so is recommended for the FLOWer model and the TNT-Kok modification.

`Number of stations in damped region [NDAMP]:` integer, default 5

- Lilo input parameter NDAMP: number of stations in damped region.

`Number of time steps per period:` integer, default 50

- Number of time steps per period for periodic motion.

`Offset for pmin criterion:` float, default 0.0

- Downstream offset in grid units for using cp,min transition criterion.

`Offset for polylines extrapolation:` float, default 1.5

- In grid units! The new transition line should be extrapolated beyond the edges of the corresponding boundary part.

`Old grid prefix:` character string, default (none)

- The prefix (including the path) of a dual grid generated in a previous run of the preprocessor, i.e. the name of 'Grid prefix' used before. If this prefix is defined part of the old dualgrid is read to avoid CPU-intensive recomputing of data. It is automatically checked which data can be re-used, which might be the fine grid metrics the agglomeration, the coarse grid metrics, the partitioning or the coloring. This reduces CPU-time, (e.g. if only the coloring needs to be changed because switching from a scalar to a vector computer) and/or allows to run on grids with identical agglomeration and partitioning even the grid coordinates have changed e.g. due to grid deformation.
  This option is available for the most usual settings only! It can not be applied for dual grids containing periodic or mirror plane boundaries, not for 2D grids and not for the parallel mode of the preprocessor!

`Omega boundary condition type:` character string, default smooth_standard

- Formerly "Omega boundary condition type (0/1/2)" or "Omega boundary condition type (0/1/2/3/4)".
  Switch for different boundary conditions for ω at walls in kω turbulence models.

– value = smooth_standard:
Default implementation for smooth walls. The value for ω is prescribed on the wall. As ω becomes (theoretically) infinite there, as proposed by Menter the wall value of ω is computed from the value of ω at the first node above the wall, i.e., based on the wall distance of the neighboring point. A correction is applied on coarse grids of the multigrid algorithm. This boundary condition is recommended if standard low-Re boundary conditions are used.

– value = smooth_Rudnik:
Implementation for smooth walls according to Rudnik (FLOWer-Code) based on a constant reference length. Thus, there is no coarse grid correction.

– value = smooth_Wilcox:
Implementation for smooth walls according to Wilcox. The boundary value for ω is prescribed at the first node above the wall. This boundary condition is recommended if wall functions (i.e., hybrid-Re boundary conditions) are used.

– value = rough_Wilcox:
Boundary condition at rough surfaces according to Wilcox (not recommended). Smooth surfaces are automatically treated according to Menter. Note that this roughness boundary condition can give poor results, in particular for transitional roughness and in the fully rough regime, i.e., for large values for the roughness parameter $k_r^+ = k_r u_\tau / \nu$. Therein, k_r denotes the equivalent sand grain roughness, u_τ is the friction velocity and ν denotes viscosity. Instead we recommend to use the DLR roughness model (value = rough_DLR).

– value = rough_DLR:
Recommended boundary condition at rough surfaces. It has been designed to give predictions for a turbulent boundary layer flow over a flat plate with surface roughness close to experimental data over a wide range of roughness values (transitional roughness and fully rough regime) and gives predictions which are very close to the roughness extension for the SA models with Dirichlet boundary condition. Smooth surfaces are automatically treated according to Menter.

  * Known Problem 1: In some cases starting with the rough_DLR boundary condition from scratch is unstable. In such cases, we recommend to perform 100-1000 start-up steps using the rough_Wilcox boundary condition, and then switch to the rough_DLR boundary condition.
  * Known Problem 2: In one particular case (airfoil with global roughness value $k_r = 1.0 \times 10^{-3}$ with a relatively large value for k_r^+ (i.e., $k_r^+ \geq 700$) both the Wilcox roughness modification and the DLR roughness modification may show convergence problems. In this case we suggest to use the Spalart-Allmaras model.

`Order of additional equations (1-2):` float, default {1, 2}

- In case of turbulence models used and an upwind discretization of the RANS equations this parameter defines the approximation order of the convective flux computation of the additional (turbulence) equations. It can be switched between first (1) and second (2) order, and the same upwind scheme is employed as for the RANS equations. Note, that the default value is 1 for Spalart-Allmaras type models and 2 for k-ω type and Reynolds stress models.
  This parameter is only active, if the parameter "Inviscid flux discretization type" is set to "Upwind".

`Order of upwind flux (1-2):` float, default 2

- In case of upwind discretization of the RANS equations this parameter defines the approximation order of the convective flux discretization of the RANS equations (excluding turbulence model equations). It can be switched between first (1) and second (2) order. To improve stability when changing from a first order to a second order solution, an intermediate value can be used to blend the reconstruction. Example: a value of 1.2 means that only 20% of the second order reconstruction is applied to improve the spatial accuracy of the first order approximation.
  This parameter is only active, if the parameter "Inviscid flux discretization type" is set to "Upwind".

`Origin:` array-float, default {0, 0, 0}

- The origin is needed for a rotating surface. The vector has to be given in grid units!

`Origin coordinate x:` float, default 0

- Defines the origin coordinate for monitoring of moments.

`Origin coordinate y:` float, default 0

- Defines the origin coordinate for monitoring of moments.

`Origin coordinate z:` float, default 0

- Defines the origin coordinate for monitoring of moments.

`Outer-Y for deformation:` float, default 1

- The test deformation at the given y-coordinate equal the parameter 'Delta-Z at outer-Y'. A quadratic shape is computed, with zero displacements below 'Start-Y', a displacement of "Delta-Z" at the given location and larger displacements further down the y-axis.

`Output files prefix:` character string, input required

- Character string giving the prefix (including path file) of the NetCDF output file. For unsteady calculations the string *"unsteady"* is appended to the *Output files prefix*. Note that integer values denoting domain number, and iteration number are also appended to the *Output files prefix*. A double value is appended to the final string, giving the "true time" of the timestep at which the file was written, for unsteady calculations. For example *test.unsteady.pval.50.t=1.0* means that the data was written at the 50th iteration after a numerical period of 1.0 seconds had been computationally evaluated.

`Output level:` integer, default 5

- Controls the amount of information being printed to the screen. A larger value causes more detailed information to be printed, a smaller value less information. Large values (> 50) invoke a first debug-level, which activates (depending on the program) additional test-functions. Values > 100 give full debug information.

`Output period:` integer, default 999999

- This parameter returns the frequency of output. For example, we have run an instationary calculation for 4 periods with 100 steps/period. The size of the stdout file is then 400 lines of time series data. We specify *Output period* = 100, so that data from the first period, second period, etc are written to different files. Use of the default value simply results in a single file to which all data has been written.

`Output shifted points grid (0/1):` integer, default 0

- If this parameter is switched on, the primary grid with an attached prism/hexaeder layer is written. This feature is only available if the parameter `Preprocessing for incompressible solver` is activated.

`Part of low quality elements:` float, default 0.0

- This parameter determines the part of elements which are considered as geometrically bad. These elements are refined isotropically if they are refined. If they can not be refined isotropically because they have some unrefinable edges (e.g. vertical layer edge) they are marked as unrefinable before the indication. The default value of 0.0 or lower means the geometric quality is not considered. The refinement quality (possibility to build valid elements after some refinement cases) is still regarded. The parameter value p is with respect to the initial grid. The actual part of affected elements in the current grid can differ but is limited to 0.1 in order to limit the disturbance of the equidistribution principle by spending of additional points for the isotropic refinement of geometrically bad elements. Because of this disturbance the parameter has to be used carefully. In test examples a values around $p = 0.005$ seemed to be reasonable. This can differ very much for a specific application.

`Percentage of new points:` float, default 40

- The dimensions of the resulting refined grid will be increased by this percentage (with an accuracy of ±5%). For Refinement mode `both` or `add` non-negative values are allowed. For Refinement mode `both` or `remove` non-positive values are allowed.

`PETSc options:` character string, default (none)

- For "Krylov loop: PETSc", and any situation where the linear solver library PETSc is applied, this string is read and passed to PETSc as if it were a command-line argument (the usual method for controlling PETSc-linked applications). For example to choose a GMRes solver with restart 20, this parameter would be "-ksp_type gmres -ksp_gmres_restart 20". See the PETSc user manual for more information.

`Plane normal x:` array-float, default 0

- The first value is the x-component of the normal vector of the first cutting plane, the second of the second cutting plane etc. Up to Number of planes values.

`Plane normal y:` array-float, default 1

- The first value is the y-component of the normal vector of the first cutting plane, the second of the second cutting plane etc. Up to Number of planes values.

`Plane normal z:` array-float, default 0

- The first value is the z-component of the normal vector of the first cutting plane, the second of the second cutting plane etc. Up to Number of planes values.

`Plane output description file:` character string, default (none)

- The parameter file name from which cutting plane output parameters may be read. If the filename is the parameter file itself, either the name (with path) of the parameter file has to be set or for simplicity the key '(thisfile)' can be used. The latter is of advantage because a renaming of the parameter is not necessary when renaming the parameter file.

`Plane output period:` integer, default all

- Gives the desired plane output period. If the value is set to 0 not any solver-iteration is performed, only output is written according to other output settings.

`Plane output values:` character string, default (none)

- The name of the output variables joined with an underline e.g. *xyz_cp_cf_yplus*. If it is not given, a default list is generated (the default variables depend on whether it is an Euler or viscous calculation) and is printed to the standard output. The valid names are listed in the subsection *Output files*.

`Plane support x:` array-float, default 0

- The first value is the x coordinate of the first cutting plane, the second of the second cutting plane etc. Up to Number of planes values.

`Plane support y:` array-float, default 0

- The first value is the y coordinate of the first cutting plane, the second of the second cutting plane etc. Up to Number of planes values.

`Plane support z:` array-float, default 0

- The first value is the y coordinate of the first cutting plane, the second of the second cutting plane etc. Up to Number of planes values.

`Point fusing reward:` float, default 1.2

- In a first step the fusing algorithm constructs a set S of all free volumes neighboring to a seeding volume. 'Free' here means that the points are not yet fused. 'Neighboring' control volumes have at least one point in common. In a second step a subset of S is selected to be fused with the seeding volume. This selection is controlled by the point fusing reward parameter in the tetrahedral part of the grid. The parameter has no effect on a structured part of the grid, which is composed of prisms or hexahedra. The selection tries to minimize the ratio of the cell surface divided by the cell volume. For small values of the parameter (between 1 and 1.6, e.g. 1.2) a semi coarsening effect is obtained in stretched cells. For large values (1.6 or even larger, e.g 1000) a full coarsening results in selecting the subset equal to S.

`Point of point pressure cost function:` integer, default 0

- For "Solver type" of "DAdjoint", "Primal" or "CAdjoint": for the case that "Cost function: Point-pressure" the global index of the node in the mesh whose pressure should be used as a cost function.

`Points skipped near stagnation point:` integer, default 5

- Number of points in the neighborhood of the stagnation point which are excluded for the application of empirical transition criteria based on integral boundary layer parameters.

`Positivity scheme:` integer, default 0

- It is well known that positivity of the discrete turbulent variables at any time during iteration is crucial in order to gain increased robustness in the iteration of an explicit Navier-Stokes solver coupled with a two-equation turbulence model and accelerated with a multigrid method. Unphysical negative values rapidly lead to instability and positivity must therefore be ensured. The scheme is applied to the time stepping and the multigrid update and guarantees that the explicit time stepping or the updating of the unknown during the iteration cycling is done such that positivity is retained.

`Prandtl number:` float, default 0.72

- The Prandtl number is used for the internal non-dimensionalization of the equations.

`Pre-maximum delta for transition:` float, default 1.e20

- Same as "Maximum delta for transition", but for pre-prediction mode.

`Pre-prediction end iteration nr:` integer, default 1000

- In pre-prediction mode, the number of the end iteration (only in defaults block!).

`Pre-prediction mode:` integer, default 2

- The so called "Pre-prediction mode" is a procedure which should be applied before the actual prediction process in order to stabilize the computation: a.) Set transition at laminar separation points to stabilize the computation in the transient phase or b.) Set transition at cp,min e.g. when started as fully turbulent calculation. Mode 1= transition at cp,min + offset (where offset = "Pre-Maximum delta for transition"). Mode 2= laminar separation from TAU. Mode 3= laminar separation from boundary-layer code Coco (NOT YET AVAILABLE!).

`Pre-prediction period:` integer, default 20

- In pre-prediction mode, the period length (only in defaults block!).

`Pre-prediction start iteration nr:` integer, default 20

- In pre-prediction mode, the number of the start iteration (only in defaults block!).

`Pre-relaxation factor for transition:` float, default 1.0

- Same as "Relaxation factor for transition", but for pre-prediction mode.

`Preconditioning:` character string, default (none)

- Selection of type of preconditioning. Preconditioning can only be used for the central scheme with scalar or matrix dissipation and for the MAPS+ upwind scheme.
  - value = (none):
    No preconditioning is applied (default).
  - value = PrimOld:
    In case of the MAPS+ upwind scheme, preconditioning is switched on, in case of the central schemes preconditioning is switched on and the artificial dissipation is based on primitive variables p, u, v, w, T. Preconditioner does not differ between subsonic and supersonic flows.

- value = PrimNew:
  In case of the MAPS+ upwind scheme preconditioning is switched on, in case of the central schemes preconditioning is switched on and the artificial dissipation is based on primitive variables p, u, v, w, T. The preconditioner differs whether a cell is in subsonic or supersonic flow.
- value = Conservative:
  Preconditioning is switched on and the artificial dissipation is based on conservative variables. This value can only be selected for the central scheme with scalar dissipation.

Preconditioning should be used for low-speed flows. In these cases convergence and accuracy are improved.

`Prediction end iteration nr:` integer, default 10500

- In prediction mode, the number of the end iteration (only in defaults block!).

`Prediction info output level:` integer, default 3

- Gives information about the transition prediction.

`Prediction iteration offset:` integer, default 0

- The value of prediction iteration offset will be added to the current Tau-Iteration number to decide, if the transition prediction will be called. This parameter is useful for an unsteady restart with transition prediction, as the iteration number in unsteady cases may be reset to 0 upon restart. This parameter is updated in the parameter file at the end of a unsteady calculation with transition prediction.

`Prediction period:` integer, default 500

- In prediction mode, the period length (only in defaults block!).

`Prediction start iteration nr:` integer, default 1000

- In prediction mode, the number of the start iteration (only in defaults block!).

`Prepcp executable:` character string, default ./prepcp

- Path to Prepcp executable.

`Prepcp factor for number of stations:` float, default 1.0

- Factor to modify number of streamline stations in Prepcp.

`Prepcp max. x/c on lower surface:` float, default 1.0

- Maximum local coordinate on lower surface of line-in-flight ccut up to which Prepcp will be executed.

`Prepcp max. x/c on upper surface:` float, default 1.0

- Maximum local coordinate on upper surface of line-in-flight ccut up to which Prepcp will be executed.

`Prepcp number of stations:` integer, default -1

- Fixed number of stations to be used in Prepcp. If set to -1, the number of stations is f_{prepcp}*n_c, where f_{prepcp} is set by the parameter Prepcp factor for number of stations and n_c is the number of points of the line-in-flight cut.

`Preprocessing for incompressible solver (0/1):` integer, default 0

- This parameter must be set to 1 to create dual grids suited for the THETA-solver. Ignore when using the TAU-Code.

`Prescription block name:` character string, default (none)

- Name of prescription block.

`Prescription info output level:` integer, default 1

- Gives information about the transition prescription.

`Prescription input data structure:` character string, default Block

- Defines the style how the transition is described.
  - Block: Transition prescription data is expected to be in a format like it is used in the Tau preprocessing
  - Array: Transition prescription data is given as a list of value pairs (only 2D)

`Prescription values periodic (0/1):` integer, default 1

- Switch to apply transition prescription periodic (only alpha and time for Transition prescription value name).

`Primary grid filename:` character string, input required

- The name (including the path) of the grid file in NetCDF-format.

`Profile data file:` character string, default -

- The filename of file in which the profile data is to be saved.

`Profile every n steps:` integer, default 0

- This parameter is given an integer value, n, which specifies that a profile data point is stored every n iterations. The operation is only valid for Cartesian meshes.

`Profile normal x:` array-float, default 0

- Specify the x component of the normal vector for wall normal data extraction on a Cartesian mesh.

`Profile normal y:` array-float, default ≥ 0

- Specify contribution of y coordinate to interpolation line normal vector. The profile option is only valid for a Cartesian grid.

`Profile normal z:` array-float, default ≥ 0

- Specify contribution of z coordinate to interpolation line normal vector. Only valid on Cartesian mesh.

`Profile output allowed (0/1):` integer, default 0

- Allow profile output on a boundary part. Will work only for Cartesian type meshes.

`Profile output description file:` character string, default -

- Name of file containing parameter list for profiling. Use only on Cartesian type meshes.

`Profile output period:` integer, default ≥ 0

- A profiled point set corresponds to a set of points extracted along a user-defined line. At present only nodal values in a Cartesian mesh can be extracted. This parameter determines the frequency of writing profiled data to disk. At present only point data is supported.

`Profile output values:` character string, default (none)

- List of variables required for the interpolation library - under development at present.

`Profile range:` array-float, default 0

- This option determines the length of the data line from origin over which line data is sampled. At present the profiling line options will work only for a Cartesian grid.

`Profile support x:` array-float, default 0

- This parameter sets the x coordinate of a point or the starting x-coordinate of a line.

`Profile support y:` array-float, default 0

- This parameter defines the y coordinate of a point or the start of a profiling line.

`Profile support z:` array-float, default 0

- Z coordinate of a point or the start of a profiling line.

`Project boundary control volumes coordinates (0/1):` integer, default 1

- If this parameter is set, the coordinates of the boundary faces of control volumes surrounding shifted boundary points are computed by a projection of these shifted points. Otherwise, the original boundary point coordinates are used as face coordinates. The influence on the accuracy and robustness of the THETA-solver should not be very strong.
  Ignore when using the TAU-Code.

`Project wall distance (0/1):` integer, default 0

- The wall distance of a field point is the lowest distance from the distances between the field point and all points of the viscous surface. It has influence on the results of some turbulence models particularly on the 1-eqn. Spalart-Allmaras turbulence models. The wall distance is inaccurately on non-orthogonal boundary meshes. For a non-orthogonal mesh the cell faces in the boundary aren't perpendicular to the surface.
  If this parameter is switch on (1) a correction by the use of a projection of the wall distance to the whole viscous surface (points, edges, elements) is calculated. It is recommended to turn this parameter on (1), if 1-eqn. turbulence models are used on non-orthogonal boundary meshes.
  ATTENTION: The result of the projected wall distance for different number of partitions isn't unique necessarily using the parallel preprocessing.

`RANS box support x:` array-float, default array of reals

- Let LS^i represent the lower left hand side of the i^{th} cut-out box, and RS^i represent the upper right hand side of the same cut-out box. If parameter "Number of RANS cut-out boxes" has a non-zero value n, then this parameter is a vector of length $2n$ with the x-coordinates of the box edges ordered as follows: $LS^1, RS^1, ..., LS^n, RS^n$.

`RANS box support y:` array-float, default array of reals

- Let LS^i represent the lower left hand side of the i^{th} cut-out box, and RS^i represent the upper right hand side of the same cut-out box. If parameter "Number of RANS cut-out boxes" has a non-zero value n, then this parameter is a vector of length $2n$ with the y-coordinates of the box edges ordered as follows: $LS^1, RS^1, ..., LS^n, RS^n$.

`RANS box support z:` array-float, default array of reals

- Let LS^i represent the lower left hand side of the i^{th} cut-out box, and RS^i represent the upper right hand side of the same cut-out box. If parameter "Number of RANS cut-out boxes" has a non-zero value n, then this parameter is a vector of length $2n$ with the z-coordinates of the box edges ordered as follows: $LS^1, RS^1, ..., LS^n, RS^n$.

`Ratio of delta over h:` float, default >1

- In LES the aim of filtering is to allow the resolved field to be adequate represented with fewer modes than are require to adequately resolve the unfiltered field. Allowing h to represent the characteristic cell length scale, and D to represent the filter width, then h/D characterises the resolution. Correspondingly, a large value of D/h implies a high level of numerical accuracy, whereas a smaller value corresponds to a larger range of resolved turbulent motions but with a reduced numerical accuracy. The optimum value is problem and model dependent. At present no definitive guide can be given.

`Ratio Prandtl lam/turb:` float, default 0.8

- The default value provides a turbulent Prandtl number of 0.9 in case of the default laminar Prandtl number 0.72. In the more general case of gases with a non constant laminar Prandtl number, the constant ratio provides a better approximation or the turbulent part of the heat flux than an approximation via a constant turbulent Prandtl number.

`RBF basis coordinates and deflections filename:` character string, default (none)

- This file includes the main input data needed for the deformation process. Within the file a set of coordinates and corresponding deflection data is specified. The file format can be either ASCII or NetCDF. The format is specified with the parameter <RBF surface format name>. The file includes the dimension (no of data sets included) and a list of coordinates and deflections. A file name has to be specified for each group.

`RBF deflections reduction factor:` float, default 1.0

- Can be used, to stabilize a coupled computation in the transient phase, if RBF deformation method is used. All deflections are multiplied by the reduction factor. Default value is 1.0. A deflections reduction factor can be specified for each group.

`RBF markers specifying group:` marker list, default -

- This parameter is used to specify, which boundary marker is part of a group. This information is needed, to compute the wall distances relative to the walls of a group.

`RBF matrix name:` character string, default (none)

- Name of the file for the interpolation matrix. This matrix is needed to compute the coefficients of the RBF. Because this matrix is only a function of the coordinates of the base points, it has to be computed only once.

`RBF maximum number of base points used:` integer, default 1000

- This is the number of desired base points, which shall be selected from all base points. The number has to be greater or equal 4. It is especially needed, if a large number of base points is given. An algorithm based on an octree approach tries to select a number of nodes, which have approximately the same distance.

`RBF name:` character string, default volume-spline

- This parameter is used to select the type of RBF. Currently, there are four types of basis functions implemented: volume spline, cubic spline, thin plate spline and a higher-order thin plate spline. The user needs to specify the key value <volume-spline>, etc.

`RBF number of groups:` integer, default 0

- This parameter specifies the number of groups, for which RBFs (Radial Basis Functions) are computed. For each group, a RBF is computed separately. The resulting deflection of a node in the volume mesh is computed by a weighted average of all RBFs. If the parameter value is equal 0 (default), the standard mesh deformation is used. So this parameter is also used to activate RBF mesh deformation.

`RBF radius euclid:` float, default 1.0

- Currently not in use!.

`RBF radius full weight:` float, default 1.0

- This parameter is used to control the influence of a RBF in dependency of the distance relative to the walls of a group. From distance 0.0 to the specified value, the RBF of current group is weighted with a factor of 1.0. Between distance <RBF radius full weight> and <RBF radius zero weight>, the weight of the RBF is blended from 1.0 linearly down to 0.0. That means, for a distance larger than <RBF radius zero weight> the RBF has no influence anymore on the deflection of a node.

`RBF radius zero weight:` float, default 10.0

- See parameter <RBF radius full weight>.

`RBF surface format name:` character string, default (none)

- Format of the file including the coordinates of the base points and corresponding deflections. Valid values of the parameter are scattered (= ASCII file) or netcdf.

`RBF walldistances filename:` character string, default (none)

- This file contains the wall distances, relative to the surface nodes of a group. A file name has to be specified for each group.

`Read partitioning filename:` character string, default (none)

- The name (including the path) of the input-file containing external grid partitioning information. The file is assumed to contain N integer values, with N being the number of grid points. The value of the first integer determines the domain number of point 0, the second the domain number of point 1 and so on. If this parameter is set, no internal grid partitioning is computed.

`Reconstruction of gradients:` character string, default Green_Gauss

- Upwind schemes need the reconstruction of gradients for higher order accuracy. If a first order Taylor series expansion is considered the gradient in the second term can be computed with different approaches. The most popular way to compute these gradients is the Gauss divergence theorem. However, for hybrid grids this method leads to numerical errors. To compensate these errors a least square approach, as keyword Least_square, is introduced.

`Recover time step:` float, default 0

- If greater that 0, this parameter allows the user to recreate the correct delta t for time series display in physical time.

`Reference bl-thickness:` float, default 1e+22

- This parameters has two meanings:
  1) In case of turbulent computations the transition management is done by limiting the production of turbulence (production $\leq$ destruction) in laminar regions. Thus no turbulence is produced (laminar flow remains laminar), but e.g. a jet of eddy viscosity can be convected through such a region. A grid point is considered to be in a laminar region if the nearest surface is a 'laminar no-slip wall' and if the distance is smaller than the defined height of the laminar region. This parameter defines an upper limit of laminar regions. The solver uses the minimum of the local heights (defined by the preprocessor) and this parameter. Thus the maximum height can be reduced without re-doing the preprocessing. Note: setting this value to 0 results in a fully turbulent computation, because all laminar regions computed by the preprocessor are neglected
  2) In case that the solver with a two-equation turbulence model uses a restart file produced with an one-equation turbulence model or from an Euler solution the non-available turbulent quantities are initialized. The bl-thickness is used to initialize near wall profiles. A reasonable estimate of the boundary-layer thickness might improve the initial values.

`Reference density:` float, default 1.29251

- The 'Reference density', 'Reference pressure', 'Reference temperature' and the 'Reynolds length' multiplied by the 'Grid scale' specify the physical dimensions to interpret the dimension free values of the calculated results. → 'Reference Mach number' Calculating an inviscid, perfect gas flow field the three thermodynamic variables of the reference state (density, pressure and temperature) are related via the perfect gas law and the 'Gas constant R'. In case of viscous flow fields the viscosity law (Sutherland constant) restricts the free choice further to exactly one of the three variables.

`Reference flow direction:` array-float, default {1, 0, 0}

- This direction is needed for the automatic determination of the geometrical sweep angles and for the generation of cutting planes for the contourline calculation from surface normal vectors.

`Reference length (rolling/yawing momentum):` float, default 1

- The monitored momentum coefficients are scaled with this length given in grid units (CM /= length), which is needed to output correct values according to the definition of CM. If no momentum length is defined a momentum length is assumed to equal 1.0.

`Reference length (pitching momentum):` float, default 1

- The monitored momentum coefficients are scaled with this length given in grid units (CM /= length), which is needed to output correct values according to the definition of CM. If no momentum length is defined a momentum length is assumed to equal 1.0.

`Reference Mach number:` float, default not_defined

- Most of the reference parameters specify a reference state which is used: by output routines to calculate values like the Pitot pressure, the pressure coefficient or heat flux coefficient, in some boundary routines for example as default values, during initialization of the flowfield, when starting without a restart file, to interpret other parameters like the Reynolds number or the Prandtl number, for the internal non-dimensionalization of the equations.

`Reference outer pressure:` float, default reference state

- This dimensionalized value is needed to compute the correct monitor forces, e.g. for internal flows interacting with the exterior, i.e. thrust nozzles where the force of the outer pressure reduces the thrust. For external flows (e.g. around a complete aircraft) this value has no effect, because in the integration of the pressure difference between the local and the outer pressure this value cancels out if the body is closed. When using a half model on a symmetry plane this would not be the case and the value has to be proper defined.

`Reference pressure:` float, default 101325

- → 'Reference density'.

`Reference relation area:` float, default 0

- The monitored forces (e.g. lift and drag) are related to the defined area given in grid units squared. If the value equals zero the surface area is computed. The computation is done by integrating all z-component of the surface elements and multiplying the sum

by 0.5, which is correct for a closed body with an upper and a lower surface part. A small error is introduced by this integration e.g. on trailing edges, because the normal vector is (mainly) directed along the x-axes such that the z-component of the area for the upper and lower side is not taken into account. However, this effect should have a small influence only, if trailing edges are sufficiently finely discretized.

`Reference system of forces and moments (tau/ln9300):` character string, default tau

- This parameter allows the user to select in which reference system forces and moments (and their coefficients) will be written. Option 'tau': Forces and moments are written in the 'tau' body-fixed system (x-positive nose-to-tail, z-positive up). Option 'ln9300': Forces and moments are written in the body-fixed system according to the LN9300 standard (DIN 9300) (x-positive tail-to-nose, z-positive down). EXCEPTION: C-lift, C-drag, and C-sidef are always written in a 'tau'-like aerodynamic system (x-positive in flow direction).

`Reference temperature:` float, default 273.15

- → 'Reference density'.

`Reference velocity:` float, default -

- → 'Reference Mach number'.

`Refinement mode:` character string, default add

- This parameter controls the basic behavior of the adaptation. In dependence on the indicator type and the required increase (Percentage of new points) of the grid, the adaptation will only add points to the grid (`add`), will only remove points from the grid (`remove`), will add and remove points (`both`). In order to do only y_+ adaptation (Change wall normal distribution (0/1)) its value can be set to (none).

`Reinitialize flow averaging:` array-int, default 0

- Input to the parameter is a vector of integers. Each integer can only take the value of 0 or 1. The vector position corresponds to the averaging key words "mean", "variance" and "meanvelgrad" in that order. See parameter "Average variables" for a short description of these key words. The default values are (0 0 0) which means that all averaging variables are restarted from the values in the restart file. If the values do not exist then the starting values for the respective averaged group (ie "mean", "variance"...) are set to zero. If any element of this input parameter array is set to 1 then the values for the respective group are set to zero after the restart. For example setting the parameter to (0 1 0) means that at the next restart the means will be restarted from data in the restart file, the variances will be initialized to 0, and the mean velocity

gradient will be also started from the data in the restart file. Note that for a proper restart the statistical data must be written to the restart file, so that the key words "mean", "variance" or "meanvelgrad" must be appended to the string set for the parameter "Field Output Values". Read the user guide under the DES/LES section for a full description.

`Relate period to start iteration (0/1):` integer, default 1

- Switch to decide if transition prediction period is related to first call of transition module (Prediction start iteration nr).

`Relaxation factor for init station for task10:` float, default 0.5

- Relaxation factor for the initial station for task 10 of Lilo.

`Relaxation factor for init station for task30:` float, default 0.0

- Relaxation factor for the initial station for task 30 of Lilo.

`Relaxation factor for last station for task10:` float, default 0.5

- Relaxation factor for the last station for task 10 of Lilo.

`Relaxation factor for last station for task30:` float, default 0.0

- Relaxation factor for the last station for task 30 of Lilo.

`Relaxation factor for modified cp:` float, default 0.5

- Relaxation factor for last point of pressure modification (see Modify cp for Coco input and UNRESOLVED REFERENCE(use_cpmin_max_as_reference_for_modified_cp))

`Relaxation factor for transition:` float, default 0.8

- Under-relaxation factor for iteration of the transition locations (see transition prediction parameter 'Maximum delta for transition').

`Relaxation solver:` character string, default Runge_Kutta

- Specifies which method of time-stepping to use. The explicit scheme is chosen with the `Runge_Kutta` keyword, the implicit schemes with `Backward_Euler`. The choice of time-stepping affects only the convergence of the solver, not the spatial discretization - and therefore the fully-converged solution - is unaffected. Some functionalities of the TAU-Code are not yet operational in conjunction with the implicit schemes - vectorisation .

`Repair selection angle for hexas:` float, default 3

- In the repair stage of the grid deformation all elements with angles lower than the given value are marked for repair. The value should be larger than 0, but small in order to select only bad cells which fall below this limit due to deformation. Thus, the limit should be smaller than the worst angle in this type of element of the grid before deformation. If the angle is chosen too large unnecessary computing time will be spent to repair cells which can not be improved, because they can not become better (in general) than they were in the original grid.

`Repair selection angle for prisms:` float, default 3

- → 'Repair selection angle for hexas'.

`Repair selection angle for pyras:` float, default 0

- → 'Repair selection angle for hexas'.

`Repair selection angle for tetras:` float, default 3

- → 'Repair selection angle for hexas'.

`Residual-data prefix:` character string, default (none)

- Whenever gradients are needed for the gradient based optimization they are composed from the adjoint flow variables and a finite difference step for the mesh sensitivity up to now. The file which is written from the solver 'Volgrad' contains the flow residuals from the whole volume domain. The final gradients are then computed while reading the flow variables, adjoint variables and residual variables into 'Volgrad'.

`Residual monitoring type (0/1):` integer, default 1

- If 1 (the default), normalize the monitoring output ρ-residual based on the initial residual. The residual will be renormalized at the start of full-multigrid iterations and dual-time inner iterations. Otherwise no normalization is undertaken.

`Residual smooth epsilon:` float, default 1.8

- If the residuals are smoothed after the flux evaluation (Residual Smoothing Steps $\neq 0$), the strength of the smoothing is dependent on this value. In the explicit case (Laplacian type smoother) the weight of the point value is (eps), the weight of its neighbors is (1-eps). In the implicit case, (eps) is interpreted as the ratio between the explicit (without smoothing) CFL-Number and the implicit CFL-Number. Not active, if no smoothing is employed.

`Residual smoother:` character string, default Point_explicit

- When Runge-Kutta relaxation is used convergence may be stabilized by smoothing the residuals. The residual smoothers available can be selected by the keywords:
  Point_explicit, Upwind_implicit.
  For the central spatial discretization the use of the default smoother (Point_explicit) is recommended. For the upwind schemes Upwind_implicit should be preferred. Residual smoothers do not apply to backward Euler time-stepping.

`Residual smoothing steps:` integer, default 2

- Number of smoothing steps for the residual at the last Runge-Kutta step. If a negative value (steps < 0) is given, (-steps) will be performed at each Runge-Kutta step.

`Restart-data prefix:` character string, input required

- The name prefix (including the path) of the NetCDF input file containing results from a previous run. If it is set to (none) the flow calculation is started from scratch.

`Reynolds length:` float, default 1

- Reference length to interpret the Reynolds number. The value has to be specified in grid units (→ 'Grid scale'). Therefore, the dimensionalized length scale of the Reynolds number is given by the product of 'Reynolds length' times 'Grid scale'.

`Reynolds number:` float, default not_defined

- Example: Assume perfect gas 'air' → $\gamma = 1.4, R = 287\,\frac{J}{kg\,K}$, normal density $\rho = 1.2925\,\frac{kg}{m^3}$, temperature $T = 273.15\,K$ and by that viscosity $\mu = 17.16 \cdot 10^{-6}\,Pa\,s$. Let the geometry be a wing with characteristic length (chord) of $0.75\,m$ and the grid be specified in mm ⇒ $Grid\,scale : 0.001$, $Reynolds\,length : 750$ (or grid scaled according to the chord length ⇒ $Grid\,scale : 0.75$, $Reynolds\,length : 1.0$).
  For a given Mach number $Ma = 0.8$ the Reynolds number is then calculated via:

$$Re = \frac{\rho\,u\,L}{\mu} = \frac{\rho\,Ma\sqrt{\gamma\,R\,T}\;Grid\,scale \cdot Reynolds\,length}{\mu} \approx 14.97 \cdot 10^6.$$

`Roughness filename:` character string, default (none)

- The name (including the path) of the file containing a user-defined pointwise wall roughness distribution. The structure of the file is of NetCDF-type and similar to e.g. Deformation description filename.
  Such a file can be created as follows:

  1. Run preprocessing and solver with constant wall roughness and write surface output of the wall roughness.

2. Modify the surface output file by pointwise specification of wall roughness. The roughness value has to be specified in grid units (→ 'Grid scale'). Notice that the distribution for the equivalent sand grain roughness height has to be specified, and not the geometrical roughness height.
3. Run preprocessing to read the pointwise roughness distribution from the modified surface output file.

Note that the roughness file is read only if option "Consider wall roughness: pointwise" is used.

`RPM allow restarts (0/1):` integer, default 1

- For "Krylov loop: RPM": smooth restarts with RPM require output of the basis of the unstable subspace found so far. This results in very large restart files in some circumstances. If this value is zero, restarts will still be possible, but RPM will attempt to find the basis from scratch again.

`RPM Krylov acceptance ratio:` float, default 100.0

- For "Krylov loop: RPM": determine how sensitive RPM is at identifying unstable modes. A low value implies the rapid extraction of a (likely) low quality basis; for high values RPM will be careful to ensure the accuracy of the basis at the cost of many iterations. In practice convergence rate and stability is very sensitive to the value of this parameter.

`RPM maximum dimension of P:` integer, default 20

- For "Krylov loop: RPM": maximum dimension of the solution subspace RPM identifies as unstable. This serves to limit the amount of memory consumed by RPM in a similar manner to the parameter "GMRes inner iterations" for GMRes. If this value is too low RPM will be unable to treat all unstable modes and the computation will diverge.

`RPM maximum increase in dimension of P:` integer, default 2

- For "Krylov loop: RPM": at each attempt at extending the basis for the unstable subspace P, add maximally this number of vectors. This value should be low for stability, but at least 2 to allow RPM to capture complex-conjugate pairs of unstable eigenvalues together.

`RPM minimum iters before P update:` integer, default 4

- For "Krylov loop: RPM": require at least this number of MG cycles before RPM attempts to identify an additional unstable basis vector.

`RPM output eigenvectors (0/1):` integer, default 0

- For "Krylov loop: RPM": output the eigenvectors of the preconditioned operator in field-data format.

`RPM preconditioning iterations:` integer, default 1

- For "Krylov loop: RPM": number of MG cycles to use as a preconditioner, comments for "GMRes preconditioning iterations" apply.

`RPM verbose output (0/1):` integer, default 0

- For "Krylov loop: RPM": output to stdout additional information on the unstable subspace. This may be useful for an experienced user to better choose the value of "RPM Krylov acceptance ratio".

`RSM DES constant:` float, default 0

- Detached eddy simulation for Reynolds Stress Models is not yet activated. At present no DES implementation of the RSM model is implemented. This is an area of active research.

`Rsm diffusion model:` character string, default GGDH

- Name of Reynolds stress diffusion model:
  SGDH: Simple gradient diffusion (Shir)
  GGDH: Generalized gradient diffusion (Daly-Harlow).

`Rsm dissipation model:` character string, default isotropic

- Name of Reynolds stress dissipation model:
  isotropic: isotropic dissipation tensor (Rotta).

`Rsm implementation version:` character string, default FLOWer

- Switch for details of the implementation.
  Theory: Account for turbulent diffusion in energy equation; account for turbulent kinetic energy in total energy; use simple strain rate tensor in re-distribution term.
  FLOWer: Turbulent diffusion neglected in energy equation; turbulent kinetic energy neglected in total energy; use traceless strain rate tensor in re-distribution term.

`Rsm length scale equation:` character string, default Menter_BSL_omega

- Name of length scale equation for Reynolds stress model:
  Wilcox_omega: Wilcox omega-equation
  Kok_TNT_omega: Kok omega-equation (TNT model)
  Menter_BSL_omega: Menter BSL-omega-equation
  Hellsten_omega: Hellsten omega-equation.
  Note: SSG/LRR-ω re-distribution model requires Menter_BSL_omega

`Rsm omega limiting factor:` float, default 1.e-5

- Lower limit of omega with respect to its far field value. Limiting is required for numerical stability.

`Rsm re-distribution model:` character string, default SSG/LRR-w

- Name of Reynolds stress re-distribution model:
  Wilcox_RSM: Wilcox stress-omega model
  SSG/LRR-w: SSG/LRR-omega model

`Runge-Kutta coefficients:` array-float, default 1

- Coefficients to be used in Runge-Kutta scheme. The number of coefficients given must match the order of the Runge-Kutta scheme as specified by 'Number Runge-Kutta steps'. Default values depend upon the order of the scheme and whether a central or upwind spatial discretization is used. They are as follows:
  Upwind
  - (1.0000)
  - (0.5000, 1.0000)
  - (0.1500, 0.5000, 1.0000)
  - (0.1500, 0.3275, 0.5700, 1.0000)
  - (0.2742, 0.2067, 0.5020, 0.5142, 1.0000)
  - (0.0730, 0.1380, 0.2200, 0.3340, 0.5000, 1.0000)

  Central
  - (1.0000)
  - (0.5000, 1.0000)
  - (0.6666, 0.6666, 1.0000)
  - (0.2500, 0.3333, 0.5000, 1.0000)
  - (0.2500, 0.1666, 0.3750, 0.5000, 1.0000)
  - (0.0730, 0.1380, 0.2200, 0.3340, 0.5000, 1.0000)

`Runge-Kutta dissipation coefficients:` array-float, default {(1, 0, 0, 0, 0, 0)}

- Together with the parameter "Runge-Kutta coefficients", allows the definition of a general hybrid Runge-Kutta method of the type popularized by Jameson. The entries in the array specify the RK stage coefficient of artificial viscosity to be used on the corresponding stage.

`Runge Kutta steps for streamline integration:` integer, default 10

- Number of Runge Kutta steps within one surface element during streamline integration.

`SA-DES constant:` float, default 0.65

- This parameter can be considered as the effective Smagorinsky constant for the SA+DES methods. A value of 0.65 is returned from calibration calculations on the Decaying Isotropic Turbulence (DIT) problem. In fact, over a range of 0.45...0.65 the calibration is insensitive to the parameter value, but this is dependent in a complicated way of the background numerical dissipation model. As a rule of thumb the parameter value could be reduced if the computation returns the impression that insufficient turbulent energy has been resolved. However, be careful - the model is tuned against DIT to return a proper modelling of the energy transfer cascade from large to smaller wavenumbers. Modification of the coefficient will influence this energy transfer process and damage the validity of the computed physical model. In other words - try and reduce dissipation first before changing this parameter!!!

`SA attractor for zero value (0/1):` integer, default 0

- Set to one to allow a basis of attract for zero Spalart-Allmaras variable solutions.

`SA boundary condition type:` character string, default smooth

- Formerly "SA boundary condition type (0/1/2)".
  Type of wall boundary condition for Spalart-Allmaras turbulence models:
  - value = smooth:
    Smooth walls, Dirichlet condition
  - value = rough:
    Rough walls, Dirichlet condition (released).
  - value = rough_mixed:
    Rough walls, mixed condition (not released). This condition is not recommended, since it seems not to lead to a monotonic increase of friction with the surface roughness.

`SAMG matrix output prefix:` character string, default (none)

- Output the Jacobian of the linear system in the ASCII compressed-row storage format of the commercial algebraic multigrid solver SAMG, see publicly available document "SAMG Data structure and file specification". If the argument is not "(none)", a files with the given prefix will be written, including a right-hand side and solution file for the linear system. Only for the Jacobian built using PETSc, i.e. "Linear residual type: PETSc"

`SARC vortical correction coefficients:` array-float, default {(1, 12, 1)}

- Coefficient set for the SARC turbulence model.

`SAS flow correction model:` character string, default (none)

- Select the appropriate form of the SAS correction model. At the present, only the "sst" form has been implemented.

`Save only envelope of N-factors (0/1):` integer, default 0

- Save only the envelopes in a file and not all N-factor curves.

`Scale for additional contourline cuts:` float, default 0.25

- Parameter to control the number of points used for contourlines (Contourline cut extraction mode, mode 0, only development).

`Second wave number [BETA]:` float, default 0.0

- Lilo input parameter BETA: first wave number β.

`Selected-elements file:` character string, default (none)

- The name prefix (including the path) of the ASCII input file containing the element-numbers of tetrahedra and prisms which are to be refined. In the first line are the numbers of tetrahedra (Ntetra) and prisms (Nprism) to be refined. In the next Ntetra lines are the element-numbers of the tetrahedra and in the following Nprism lines are the element-numbers of the prisms. Such an adaptation can be executed with or without having a flow solution (defining or defining not a Restart-data prefix).

`Set aerodynamic sweep angles:` integer, default 0

- Switch to activate the setting of the aerodynamic sweep angle. The aerodynamic sweep angle is then used by to modify the local Reynolds number and the local chord length for Coco or Prepcp input. The aerodynamic sweep angle corresponds generally to the yaw angle β (Angle beta (degree)). The aerodynamic sweep angles have to be prescribed either in the streamline coordinates block (see Streamline coordinates type set) or with the parameter Aerodynamic sweep angle.

`Set effective sweep angles:` integer, default 0

- Switch to activate the setting of the effective sweep angle. The effective sweep angle is then used by Prepcp to modify the pressure distribution. If Prepcp is not used, this parameter has no effect. The effective sweep angles have to be prescribed either in the streamline coordinates block (see Streamline coordinates type set) or with the parameter Effective sweep angle.

`Set leading edge sweep angles:` integer, default 0

- Switch to deactivate automatic calculation of leading edge sweep angles. Sweep angles have then to be prescribed by the user, either in the streamline coordinates block (see Streamline coordinates type set) or with the parameter Leading edge sweep angle.

`Set trailing edge sweep angles:` integer, default 0.0

- Switch to deactivate automatic calculation of trailing edge sweep angles. Sweep angles have then to be prescribed by the user, either in the streamline coordinates block (see Streamline coordinates type set) or with the parameter Trailing edge sweep angle.

`Set transition at solver start (0/1):` integer, default 0

- Set transition locations prior to solution start in the Tau solver. Transition lines are read using the file specified under Transition blocks description file. Using this option, transition can be set regardless of the laminar/turbulent regions created by the preprocessing of Tau.

`Set transition info output level:` integer, default 1

- Gives information about the setting of transition (the actual flagging of laminar and turbulent regions in the grid).

`Set zero time origin (0/1):` integer, default 0

- If set to 1, then the start of the time series extracted from the list of stdout files is set to zero time. This feature is only to assist in graphics presentations so that several different series at different origins can be compared on identical axis.

`SG start up steps (fine grid):` integer, default 0

- Number of iterations steps to perform single grid iterations on the fine grid, afterwards multigrid iterations are performed. For RANS calculations i.e. 30 single grid iterations will result in a more robust start up.

`SGS Coefficient:` float, default 0.17

- The SGS coefficient parameter is used to set the Sub Grid Scale (SGS) model coefficient for the LES turbulence mode. At present a Smagorinsky and a WALE SGS model are implemented. It is important to note that for the Smagorinsky model, the near wall eddy viscosities are over-predicted so that an accurate solution requires that the near-wall eddy viscosity is damped using the Van Driest damping factor. As of 10.10.09 the proper treatment of the Van Driest damping has not been extended in parallel operations so that Van Driest is not yet available for the Smagorinsky SGS model in parallel and single domain operations. The DIT calibration tests suggest that optimal values of these coefficients are 0.13 for Smagorinsky, and 0,.55 for WALE Sub Grid Scale models. Again, the standard restrictions on modifying these coefficients applies.

If you feel that the solution returned contains insufficient resolved turbulence, then first reduced the level of numerical dissipation via the dissipation switches or by using an alternative numerical scheme. For example, if an upwind scheme is really required then the equilibrium flux method (EFM) should never be used as the numerical dissipation characteristic if this scheme is too large to return adequately resolved high wavenumber structure.

`SGS model:` character string, default (none)

- This parameter selects the Sub Grid Scale (SGS) model used in the LES mode of the solver. At present both Smagorinsky and WALE SGS model options are available. It must be pointed out that the near wall treatment of the Smagorinsky model is not formally correct as of 10.09.09 due to specific requirements in the calculation of the Van Driest damping factor in parallel that have not yet been implemented. This will be resolved shortly but until further notice the Smagorinsky model cannot be used with near-wall Van Driest damping as is recommended in the literature.

`Sgs stages maximum:` integer, default 3

- Number of inner linear `Sgs` iterations. This determines how accurately `Sgs` solves the linear system, a more accurate solution (usually) implies better non-linear (outer) convergence. Values of $2-5$ can be useful. If only one iteration is performed the solver is approximately equivalent to the `Lusgs` solver but computationally more expensive.

`Sharp edge angle (degrees):` float, default 0

- Sharp surface edges are detected using an angle criterion. If the inner angle between two neighboring surface triangles is smaller than the `Sharp edge angle`, a special treatment for the related boundary faces is employed. Using the default value no sharp edges are detected. The special treatment has no effect on 'viscous walls'. It can be of advantage for the inviscid calculation of e.g. the flow around a trailing edge of a wing.

`Smooth cp for Coco input (0/1):` integer, default 0

- Switch for the activation of a simple pressure smoothing. If activated, increase in pressure between stagnation point and pressure minimum is simply cut off. (It is assumed, that only a pressure decrease exists from stagnation to pressure minimum. A (small) increase in pressure may lead to early separation in the boundary layer code Coco.)

`Smoothing eps for flux weighting:` float, default 0.2

- This parameter sets the smoothing epsilon for the flux weighting smoother. This is only required to be greater that 0 for the "Strelets" smoother at present.

`Smoothing eps for sas mode flow correction:` float, default 0.2

- Number of steps used to smooth a “rough” SAS correction factor field.

`Smoothing eps for vortical flow correction:` float, default 0.2

- Number of steps used to smooth a “rough” vortical correction factor field. This is needed for SARC and SSARC models only.

`Smoothing epsilon first:` float, default 0.2

- Epsilon for the averaging of the adjusted first wall distance (Smoothing steps first).

`Smoothing epsilon for LES filter:` float, default 0.2

- The parameter is used to control the weighting of the Laplace smoother that is used to smooth the computed LES filter width. On unstructured meshes, one can imagine a situation where the distribution of characteristic control volume dimensions are non-smooth in space. This has the potential to cause some problems for DES and LES calculations where a non-smooth distribution of control volume length scales is compared against a smoother distribution of turbulent length scales. This can cause unphysical ”jumps” between LES and RANS modes in neigbouring field points - it is the ratio of these scales that acts as a switch between the LES and RANS modes. Smoothing the filter width can help reduce this problem, however the user is again reminded that you will get what you have paid for ... a bad DES grid will give a bad DES solution. As always the message is ... spend the time on ensuring you have a good DES/LES mesh and you will be rewarded.

`Smoothing epsilon last:` float, default 0.2

- Epsilon for the averaging of neighboring last points of a wall normal ray (Smoothing steps last).

`Smoothing iterations for inner tetras:` integer, default 15

- Number of iterations to be performed in tetra-smoother.

`Smoothing iterations for LES filter:` integer, default 0

- Number of smoothing iterations for the LES filter width.

`Smoothing relaxation factor:` float, default -1

- The smoother is an iterative procedure, which needs an under relaxation for convergence. If the input value is < 0 a relaxation value is computed internally, such that the largest point-movement in the 1st iteration is smaller than the local cell scale (i.e. 0.8 * scale). This is, according to current experience, robust in all cases. If a case will be found in which the tetra-smoothing diverges, the input of a smaller relaxation factor than the computed one (this is printed to stdout), will allow continuation.

`Smoothing selection angle for tetras:` float, default 15.

- All elements having an angle between its faces lower than the given value are selected for the smoother. The larger the value is, the more elements will be selected and the more CPU-time the smoother will require.

`Smoothing steps first:` integer, default 5

- Number of averaging steps on the adjusted first wall distance (Change wall normal distribution (0/1)).

`Smoothing steps for flux weights:` integer, default 0

- This parameter sets the number of smoothing steps for the Laplacian smoother used to smooth the flux weight field. This is only mandatory for the "Strelets" smoother.

`Smoothing steps last:` integer, default 10

- Number of averaging steps after adjustment of the last distance in the structured layers (Edge relation to unstructured part).

`Solve dissipation error equation (0/1):` integer, default 0

- For "Solver type" of "DAdjoint", "Primal" or "CAdjoint": solve a forward or reverse error transport equation, with source terms taken as the local level of dissipation in the discretization. In one of the adjoint modes field output (with extension e.g. ".ksensor.C-drag.pval.100") will consist of a goal-oriented mesh adaptation indicator for the cost function specified in "Cost function" (e.g. "C-drag"), in addition to the adjoint field. For the primal solver output will be an approximation of the discretization error everywhere in the field (with the standard primal solver output extension e.g. ".primal.pval.100").

`Solve linear problem on grid level:` integer, default 1

- For the linear solver types: construct and solve the entire problem on a coarse grid level at much reduced cost. This is useful for the case that significantly less accuracy is needed in the linear problem than in the flow problem.

`Solver type:` character string, default Flow

- The action the solver should perform. The default value of "Flow" solves the non-linear Euler or Navier-Stokes equations, i.e. a standard flow solver. "Output_only" reads a restart-data file and outputs surface, cut-plane and profile data depending on the output settings in the parameter file. "Residual_only" reads a restart-data file and outputs the residual of the specified spatial discretization at all points in the mesh for all equations in a field data format. "DAdjoint" also requires a restart-data file, and solves the discrete adjoint equations (a linear problem) corresponding to the given spatial discretization. "Primal" solves the flow equations linearized about the

restart-data input state. "Freqdom" solves for small perturbation time-periodic flows in the frequency domain, starting from a steady mean solution given in the restart-data file, inviscid flows only. "Findiff" is a debugging mode primarily for developers that compares the linearized version of the chosen spatial discretization, with a linearization by finite-differences on the actual spatial discretization, for verifying the linearized version. Note that only a subset of the flow spatial discretizations are implemented for the linearized solvers. If the chosen discretization is not available the code will terminate. "CAdjoint" solves the inviscid continuous adjoint equations about a flow solution given in restart-data.

`SRR limiter active (0/1):` integer, default 0

- Activate the SRR slope-limiter; for use in combination with upwind fluxes. The Spatial Radial Relaxed (SRR) - limiter presents an alternative to the modified Barth slope-limiter or Venkatakrishnan's slope-limiter when using upwind schemes. For a number of test cases the new SRR limiter has shown to enforce monotonicity of the solution while giving a decrease of the (implicit) dissipation in upwind schemes, and thus a reduction in pressure-drag and drag-coefficient. Convergence of the residual to machine-accuracy however is NOT guaranteed.

`SRR limiter radius relaxation constant:` float, default 0.03

- For upwind computations which are second order accurate in space, the solution within a control volume is reconstructed using the cell-averaged state and the computed gradient. To enforce monotonicity of the reconstructed solution in the presence of discontinuities in the distribution of the cell-averaged states limiters are needed. The limiter operator computes a scalar coefficient for every conserved quantity to reduce the computed gradient of that quantity in the case of a local discontinuity. The magnitude of the coefficient (a value between 0 and 1) is produced by comparison of reconstructed differences with allowable differences of the conserved quantity with help of a non-dimensional smooth shape function. The shape function and thus the action of the limiter are damped (relaxed to the value of one) in case of small differences (noise) to prevent the limiter from acting on noise and thus stalling convergence of the solution process. The SRR limiter radius relaxation constant (LRRC) is used in the scaling of the domain of the conserved quantity thus creating a range (radius) below which differences in the distributed quantity are identified as noise. Scaling of the domain of the difference of the conserved quantity is done with a relevant quantity of the local fluid state of the same physical dimension. Smaller values of LRRC result in a smaller radius, thus in the limiter acting more critical to overshoots, even of small magnitude. Larger values of LRRC relax action of the limiter over a larger range.

`SST-DES constant:` array-float, default {0.78, 0.61}

- The two parameter values set via this parameter are tuned to the DIT test case, as are all DES models, so that the energy cascade between neighbouring wavenumbers is properly modeled. The same restriction apply in modifying these coefficients as with the other DES model coefficients. If the level of resolved turbulence appears to be low, and you think that the grid is sufficiently fine to resolve the information that appears to be missing, you should first reduce the level of numerical dissipation.

`SST limitation version:` character string, default Vorticity

- This parameter defines the quantity used for the SST limitation:

  - Vorticity:
    Original model, Menter (1994). The original model might be too sensitive to separation in some cases.
  - Strain rate:
    Modified model, Menter et al. (2003). The modification might reduce the sensitivity to separation.

  The parameter is only active with turbulence models, using the SST limitation. Currently this holds for the following turbulence models:

  - Menter SST model
  - NLR TNT model (∗ not present in THETA ∗)
  - Wilcox k-ω model + SST correction (∗ not present in THETA ∗)
  - Menter SST-DES model (∗ not present in THETA ∗)

`Start-Y for deformation:` float, default 0

- Below the given y-value wall-surface points will not be displaced for a test deformation.

`Streamline coordinates type set:` character string, default xyz

- Character string containing the names of components of the streamline start coordinates block. The variables are joined with an underline (e.g. *xyz_lesweep_fixed*).Valid names are:

  - xyz: start coordinates of the streamlines/contourlines (three floats are expected).
  - nxyz: normal vector to define the cutting plane for contourlines (three floats are expected).
  - lesweep: leading edge sweep angle (one single float is expected).
  - tesweep: trailing edge sweep angle (one single float is expected).
  - efsweep: effective sweep angle (see also Effective sweep angle, one single float is expected).

- aesweep: aerodynamic sweep angle (see also Aerodynamic sweep angle, one single float is expected).
- plane: character string to identify the cutting plane for contourline calculation (character string expected):
  - xy: cutting plane is the x-y-plane
  - xz: cutting plane is the x-z-plane
  - yz: cutting plane is the y-z-plane
  - wn: cutting plane is built from Reference flow direction and local wall normal vector.
- fixed: character string to identify the treatment of the given coordinates (character string expected):
  - s: streamline coordinates are used as fixed transition point (no streamline/contourline calculation)
  - f: free transition (streamline/contourline calculation + transition prediction)
  - m: streamline coordinates are used as fixed transition point if this point lies in a laminar region (no streamline/contourline calculation)
  - t: transition is tripped at streamline start coordinates (streamline/contourline calculation + transition prediction)

Example for *xyz_lesweep_fixed*

```
StreamlineCoordinates
0.00 1.00 0.00 30.3 t
0.00 1.50 0.00 30.3 f
0.00 2.00 0.00 30.3 f
1.00 2.50 0.00 00.0 s
StreamlineCoordinates end
```

`Structured grid coarsening:` float, default 0

- In a first step the fusing algorithm constructs a set S of all free volumes neighboring a seeding volume. 'Free' here means that the volumes are not yet fused. 'Neighboring' control volumes have at least one point in common. In a second step a subset of S is selected to be fused with the seeding volume. This selection is controlled by the 'Structured grid coarsening' parameter in structured parts of the grid composed of prisms or hexahedra. The parameter has no effect in parts of the grid composed of tetrahedra. Those cells of S selected for fusing have a common face with the seeding volume larger than their maximum face multiplied with the parameter's value. Thus, the default results in a full coarsening (subset equal to S). Values between 0 and 1 (e.g. 0.5) cause a semi-coarsening depending of the cell aspect ratio. Note that semi-coarsening requires more computational work for each multigrid cycle because of the increased dimensions of the subgrids.

`Suppress error on orphaned points (0/1):` integer, default 0

- The calculation of interpolation coefficients for a Chimera grid fails if the overlap between two grids is too small. The resulting orphaned points will not be updated during a flow calculation. Therefore, the flow calculation stops with an error message. The user should now improve the grid overlap.
  If the switch "Suppress error on orphaned points" is activated, the error is suppressed and the flow computation continues.
  WARNING: Activate this switch only if you now what you are doing! The accuracy of the solution will be reduced! In case of a steady flow computation the free stream values will be set and kept, in case of unsteadily moving grids the flow values of the last valid update will be kept. It is recommended to never activate the switch for steady flow computations.

`Surface-weights file:` character string, default (none)

- The name prefix (including the path) of the ASCII input file containing the surface weights for each boundary marker. In each line a couple of numbers has to be written: the first is the boundary marker of the boundary to be weighted (integer), the second is the weight (float). Weights of value 1 seems to be reasonable if weighting is desired.

`Surface output description file:` character string, default (none)

- The Surface output description file defines the parameter file name from where the surface output parameters may be read.
  If the filename is the parameter file itself, either the name (with path) of the parameter file has to be set or for simplicity the key '(thisfile)' can be used. The latter is of advantage because a renaming of the parameter is not necessary when renaming the parameter file.

`Surface output filename:` character string, default (none)

- The name (including the path) of the file containing the surface output data of the TAU-Code solver containing the coordinates before deformation (with the names 'x', 'y', 'z') for the surface points to which an internally computed deformation shall be applied, as e.g. a user defined rigid body motion.

`Surface output period:` integer, default 200

- Optional integer value giving the desired surface output period. If the value is set to 0 not any solver-iteration is performed, only output is written according to other output settings.

`Surface output values:` character string, default xyz_p_other

- Optional character string containing the name of the output variables joined with an underline e.g. *rho_p_k1_k2_eddy_cf_yplus*. The valid names are listed in the subsection *Output files*. Only those boundary parts where the parameter *Write surface data (0/1)* is set to one will have their variables printed out. Please refer to *Reference system of forces and moments (tau/ln9300)* for special considerations regarding output of surface forces and moments.

`Sutherland constant:` float, default 110.4

- Constant Su in the Sutherland viscosity formula: $\mu(T) = \mu_0 \left(\frac{T}{T_0}\right)^{\frac{3}{2}} \frac{T_0 + Su}{T + Su}$.

`Sutherland reference temperature:` float, default 273

- Constant T_0 in the Sutherland viscosity formula: $\mu(T) = \mu_0 \left(\frac{T}{T_0}\right)^{\frac{3}{2}} \frac{T_0 + Su}{T + Su}$.

`Sutherland reference viscosity:` float, default 1.716e-05

- Constant $\mu_0 = \mu(T_0)$ in the Sutherland viscosity formula: $\mu(T) = \mu_0 \left(\frac{T}{T_0}\right)^{\frac{3}{2}} \frac{T_0 + Su}{T + Su}$.

`TAU recv-variables:` character string, default (none)

- This is the list of variables to be received by the TAU-Code. The input nomenclature is like that for the TAU-solver-surface output. Per default a list of coordinates is expected to be received.

`TAU remote-code Id:` integer, default 0

- This id is to identify the code coupled to the TAU-Code via the MPCCI interface. It is the id which needs to be defined in the MPCCI-input.

`TAU send-variables:` character string, default (none)

- This is the list of variables to be sent by the TAU-Code. The input nomenclature is like for the TAU-solver-surface output. Per default a list of forces is expected to be sent.

`Tecplot ascii output precision:` integer, default 0

- Switch from binary output (0) to ASCII output (>0) with as many significant digits as defined by this parameter.

`Tecplot title:` character string, default (none)

- This parameter allows definition of a title with the given string for the Tecplot display.

`Time step smoothing factor:` float, default 0

- Values below 1.0 are ignored. Higher values specify the maximum ratio of time steps in neighboring volumes. A value of unity should give results similar to global time stepping, whereas a very high value should hardly change the local time step sizes. This smoothing will slow down the convergence in cases without stability problems, but can stabilize the scheme in cases where a solution is harder to obtain.

`Trailing edge sweep angle:` float, default 0.0

- Prescription of the trailing edge sweep angle if Set trailing edge sweep angles is set to 1.

`Transition block name:` character string, default (none)

- Name of transition block.

`Transition blocks description file:` character string, default (thisfile)

- File name (including path) of file containing keywords/parameters for the transition setting of the preprocessing. Normally, this should be the boundary mapping file (Boundary mapping filename). If this parameter is set to '(thisfile)', all of the input can be contained in the same file.

`Transition history file name prefix:` character string, default transition_history

- Prefix for the name of the transition prediction history file.

`Transition history output in pre-mode (0/1):` integer, default 0

- Switch to decide if the transition_history file will be updated during the pre-prediction phase.

`Transition history output values:` character string, default set

- Character string containing the names of the output variables joined with an underline (e.g. *set_sep_pmin*).Valid names are:
  - set: The transition points applied to the code
  - new: The transition points coming from the applied prediction method
  - sep: Separation point from Tau
  - reatt: Re-attachment point from Tau
  - pmin: Pressure minimum
  - cfmin: Local skin friction minimum from Tau

  All parameters can also be applied in dimensionless form when using contourlines for the prediction of transition by adding */c* to the keyword (e.g. *set/c_sep/c_pmin/c*).

`Transition module description file:` character string, default (thisfile)

- File name (including path) of file containing keywords/parameters for the transition module. If this parameter is set to '(thisfile)', all of the input can be contained in the same file.

`Transition prediction (0/1):` integer, default 0

- Switch to activate the transition prediction.

`Transition prediction description file:` character string, default (thisfile)

- File name (including path) of file containing keywords/parameters for the transition prediction. If this parameter is set to '(thisfile)', all of the input can be contained in the same file.

`Transition prediction output directory prefix:` character string, default Tranpred

- Prefix for the directory of the output of a transition prediction step.

`Transition prescription (0/1):` integer, default 0

- Switch to activate the time dependant transition prescription.

`Transition prescription description file:` character string, default (thisfile)

- File name (including path) of file containing keywords/parameters for the transition prescription. If this parameter is set to '(thisfile)', all of the input can be contained in the same file.

`Transition prescription value:` float, default -

- Iteration number, angle of attack or time for which the prescription block is valid.

`Transition prescription value name:` character string, default -

- Parameter to define the way of the time dependant transition prescription:
  - alpha: Transition prescription dependant of angle of attack
  - iter: Transition prescription dependant of iteration number
  - time: Transition prescription dependant of physical time

`Translation factor for shifted boundary points:` float, default 1

- This is a global factor to decrease or increase the boundary shifting. The default value should be changed only, if the wall normal distance of the first and second grid layer differ strongly.
  Ignore when using the TAU-Code.

`TS transition prediction mode:` integer, default 10

- Mode for the prediction of Tollmien-Schlichting transition.

Mode	Method	Available
0	none	release
1	cp,min	release
2	laminar separation	release
6	eN-Method with N-factors from envelope method of Wuerz	development
7	eN-Method with N-factors from envelope method of Drela (1987)	development
8	eN-Method with N-factors from envelope method of Drela (1991)	development
9	eN-Method with N-factors from Database method (Stock, Degenhart)	development
10	eN-Method with N-factors from linear stability solver Lilo	release
11	Michel/Cebeci/Thwaites	development
12	AHD (Arnal, Habiballah, Delcourt)	development

`Turbulence diffusion flux type TSL/Full (0/1):` integer, default 1

- Evaluate the turbulence diffusion fluxes using a thin shear-layer approximation, or a gradient based ("full") approach. The default value should always be used by non-experts.

`Turbulence equations use multigrid (0/1):` integer, default 0

- Experience, both with structured and unstructured solvers, shows that applications of many multigrid operators to the turbulence equations can reduce the numerical stability of the system. For this reason it is recommended that the multigrid operators should be disabled (the parameter should be set to 0) when the momentum and turbulence equations are solved using multigrid convergence acceleration.

`Turbulence intensity for transition criteria:` float, default 0.0

- Specifies the turbulence intensity (in %) for transition criteria which are based on this value (e.g. Mayle, AHD).

`Turbulence mode:` character string, default RANS

- Use this parameter to specify if the solver is required to run in RANS, DES or LES modes. In RANS and DES modes the standard turbulence model or its DES variant are solved. In LES mode the specified Sub Grid Scale model is used in modeling the turbulent eddy viscosity.

`Turbulence model version:` character string, default {SAO, Wilcox_k-w, RSM}

- String identifier that defines the turbulence model to be used. All possible identifiers of the different available models are printed to stdout (to the screen or the log-file) and depend on the code version (ptau3d.turb1eq, ptau3d.turb2eq or ptau3d.turbrsm). For the 1-equation models the default is the SAO-model ('SAO'), for the 2-equation models the default is the Wilcox kω model ('Wilcox_k-w)', while Reynolds stress models are generally activated by the identifier 'RSM'. Note, that in former versions until release 2008.1.1-p1 the SAE-model ('SAE') has been the default for 1-equation models. As of September 2009, the DES models have been supplanted by function calls within the base turbulence models - thus model "SAO+DES" is identical to model "SAO" with the DES turbulence mode active. The models "SAO+DES, XLES, Menter_SST-DES" are simply retained to ease transition into the new DES framework (DES can be activated for all linear eddy viscosity RANS models) and will be removed in early 2010. For availability of models, please check the following table. Note that some models are still under development and thus are not available in any releases, which is indicated by an entry 'n.a.'.

Model	Code version	Model ID	Available
Spalart-Allmaras (Standard)	turb1	SAO	all
Spalart-Allmaras (Edwards)	turb1	SAE	all
Spalart-Allmaras modified	turb1	SAM	all
SA-DES	turb1	SAO+DES	Rel. 2005.1.0
SALSA	turb1	SALSA	n.a.
Wilcox kω	turb2	Wilcox_k-w	all
Menter Baseline model	turb2	Menter_BSL	all
Menter SST model	turb2	Menter_SST	all
LEA kω model	turb2	LEA	Rel. 2004.2.0
NLR TNT model	turb2	TNT	Rel. 2005.1.0
Wilcox kω + SST	turb2	Wilcox_k-w+SST	Rel. 2005.1.0
Menter 2layer k-eps model	turb2	2layer_k-eps	Rel. 2005.1.0
Rung RQEVM + Wilcox kω	turb2	RQEVM	Rel. 2005.1.0
WJ EARSM (2D) + Wilcox kω	turb2	WJ2D+Wilcox	Rel. 2005.1.0
WJ EARSM (3D) + TNT kω	turb2	WJ3D+TNT	Rel. 2005.1.0
Hellsten EARSM	turb2	WJ+Hellsten	Rel. 2005.1.0
XLES	turb2	XLES	Rel. 2005.1.0
Menter SST-DES	turb2	Menter_SST+DES	n.a.
Reynolds stress models	turbrsm	RSM	Rel. 2007.1.0

`Turbulence shock correction (0/1):` integer, default 0

- Most turbulence models overpredict the increase of turbulent kinetic energy due to the interaction of a turbulent flow with a shock. This correction searches for volumes around a shock and modifies locally the source terms to adjust the turbulent kinetic energy behind the shock. It is recommended for all supersonic applications as well as for those transonic applications, where for example a shock interacts with a turbulent wake of interest. The correction is implemented for all two equation turbulence models.

`Type of agglomeration:` character string, default private

- The name of the supported grid agglomeration tool to be employed. The supported names are written to stdout when executing the program. They are `private` and `MGridGen`. The latter ones can only be used if the MGridGen program is linked to the preprocessing.

`Type of coarsening for MGridGen:` integer, default 4

- The algorithm to be used for the coarsening phase. choose "1", a random agglomeration algorithm is used; choose "4", a globular agglomeration algorithm is used. This parameter can only be used if the MGridGen program is linked to the preprocessing.

`Type of grid movement:` character string, default static

- This parameter is a switch that informs the solver what kind of grid motion should be activated. The available motion types are the following:

parameter value	type of motion
static	Grid does not move
rigid	Rigid body motion
flutter	Two degree of freedom flutter
deformed	Deforming grid
deformed+rigid	Deforming grid combined with rigid body motion

`Type of partitioning (name):` character string, default private

- The name of the supported grid partitioner to be employed. The supported names are written to stdout when executing the program. They are `private`, `pmetis` and `kmetis`. The latter ones can only be used if the metis program is linked to the preprocessing. The parameter is not active if the chosen number of domains is 1.

`Type of refinement for MGridGen:` integer, default 6

- The algorithm to be used for the refinement phase. Please learn the meanings of options from the user manual of MGridGen. This parameter can only be used if the MGridGen program is linked to the preprocessing.

`Unsteady activate inner iteration output (0/1):` integer, default 0

- By activation of this parameter the option to restart the solver from within the unsteady pseudo-time step loop becomes available. Output from within the unsteady pseudo-time step loop is done either explicitly by the user from a Python script, or automatically by the solver if it receives a signal that the allowed wallclock time is about to expire.

`Unsteady computational time step size:` float, default -1

- Defines the dimensionless time step size. If no value is given the 'Unsteady physical time step size' is used.

`Unsteady extrapolation order:` integer, default 1

- Order of extrapolation for the new initial guess (only with dual-time stepping). Note: For some extreme flow conditions (very high angle-of-attack, vortex dominated flows, etc.) this may have to be dropped down to 0th order. The higher orders, 2nd and 3rd, are not recommended for use at this time.

`Unsteady implicit scheme order:` integer, default 2

- Order of backwards difference operator for time discretization (only with dual-time stepping).

`Unsteady inner iterations per time step:` integer, default 40

- Maximum number of pseudo time steps per physical time step. If the given minimum residual is reached before, this number of iterations might not be needed (only with dual-time stepping).

`Unsteady physical time offset:` float, default 0.0

- This parameter allows the user to reset the physical time of the computation by subtracting an offset from the physical time.

`Unsteady physical time step size:` float, default -2

- Physical movement in time per time step (only with dual-time and global-time stepping).

`Unsteady physical time steps:` integer, default 1

- Number of physical time steps, such that this number times the physical time step size provides the physical time to compute (only with dual-time and global-time stepping).

`Unsteady show pseudo time steps (0/1):` integer, default 0

- When set the monitor output is sent to stdout for each inner step of a dual time step cycle, or each iteration of the implicit solver is output when unsteady motion is being computed.

`Unsteady time stepping:` character string, default (none)

- This parameter controls the choice of the time stepping scheme. Local-time ((none)), dual-time (dual) or global-time (global).

`Update transition blocks in para file (0/1):` integer, default 0

- Updates the para-file at the end of a calculation if the transition locations have been changed by the transition module. This allows for a better restart usability.

`Upwind flux:` character string, default AUSMDV

- Formerly "Second order upwind solver".
  The upwind discretization schemes available can be selected by the keywords: Van_Leer, AUSM_Van_Leer, AUSMDV, Roe, MAPS+, EFM, AUSMP, AUSMPWP. This selection defines the scheme to be applied on the fine grid level at either 1st or 2nd order. Because of their accuracy, AUSMDV or Roe are recommended. In case of convergence problems the more diffusive solvers Van_Leer or EFM can be used. If low Mach number preconditioning is used, the only choice is MAPS+. AUSMPWP is still under research and should give results comparable with AUSMDV. AUSM_Van_Leer is available for direct comparisons with other DLR-codes like CEVCATS and FLOWER working on block-structured grids with a similar upwind scheme.

`Use Cauchy convergence control:` character string, default off

- Convergence control for inner-iterations based on convergence for integral coefficients of lift, drag and pitching momentum. Can be used for steady state calculations, and for unsteady calculations in dual time mode. (off: not active. relative: consider relative values for evaluation of convergence. absolute: consider absolute values for evaluation of convergence. When extension _dynamic is used, the number of iterations considered for convergence estimation is no more constant. It is determined by the parameter Factor for dynamic Cauchy convergence times the current number of iterations. The minimum number of iterations for convergence estimation is given by Number of samples for Cauchy convergence and the maximum number is 1000 iterations.)
  Related parameters are:
  - Error for Cauchy convergence clift
  - Error for Cauchy convergence cdrag
  - Error for Cauchy convergence cmy
  - Error for Cauchy convergence cost function
  - Number of samples for Cauchy convergence
  - Factor for dynamic Cauchy convergence

The aim is to give a practical criterion for convergence being reached for inner iterations in particular for unsteady calculations.
Inner iterations are assumed to be converged if relative resp. absolute changes in c-lift c_l, c-drag c_d and pitching momentum c_{my} are smaller than a specified tolerance value during the last n_sample_inner number of inner iterations.
Mathematically speaking, inner iterations stop if in case of relative convergence $\frac{|c_l^n - c_l^{n-k}|}{|c_l^n|} \leq tol\_lift$, $\frac{|c_d^n - c_d^{n-k}|}{|c\_d^n|} \leq tol\_drag$ and $\frac{|c_{my}^n - c_{my}^{n-k}|}{|c_{my}^n|} \leq tol\_my$ and in case of absolute convergence $|c_l^n - c_l^{n-k}| \leq tol\_lift$, $|c_d^n - c_d^{n-k}| \leq tol\_drag$ and $|c_{my}^n - c_{my}^{n-k}| \leq tol\_my$ for all $k = 1, \ldots n\_sample\_inner$ last iterations. Therein, the upper index n, e.g., c_l^n, denotes the value of the integral coefficient in inner iteration step n.

`Use Coco script for task11 (0/1):` integer, default 1

- Instead of performing task 10 of Lilo the database method in Coco can be used to produce initial data for task 11 of Lilo.

`Use Coco script for task21 (0/1):` integer, default 1

- Instead of performing task 30 of Lilo the database method in Coco can be used to produce initial data for task 21 of Lilo.

`Use Coco TS database results as backup (0/1):` integer, default 1

- If no N-factors are calculated by Lilo task 10, the N-factors from the Coco database method are used, if available.

`Use coordinates as displacement (0/1):` integer, default 0

- The coordinates given the file 'Deformed coordinates filename' are per default interpreted as new coordinates. If this switch is set to 1 the values are considered as delta values to the old coordinates and are added to them.

`Use cp,min/max (0/1) as reference for modified cp:` integer, default 0

- Switch to chose if the pressure distribution will be modified between
  - 0: stagnation point and pressure minimum,
  - 1: stagnation point and last point of streamline,

  when using non-constant pressure shift (Modify cp for Coco input). A relaxation factor can be applied (Relaxation factor for modified cp).

`Use grid point as start coordinate (0/1):` integer, default 0

- Use projected coordinates (0) or coordinates of nearest grid point (1) as start coordinates for streamline integration.

`Use indifference point for task10 (0/1):` integer, default 1

- Switch, to decide if indifference point from Drela/Wuerz will be used for determining initial station for task 10 of Lilo.

`Use logarithmic distribution for CF waves (0/1):` integer, default 0

- Switch, to decide if a logarithmic distribution for the wave lengths will be used when executing task 21 of Lilo.

`Use logarithmic distribution for TS waves (0/1):` integer, default 1

- Switch, to decide if a logarithmic distribution for the frequencies will be used when executing task 11 of Lilo.

`Use modified dissipation for 2D (0/1):` integer, default 0

- Switched on this parameter modifies the dissipation operator for 2D simulations only. The effect is that the dissipation becomes nearly the same as it is in 3D simulations, which is a bit smaller than in the 2D-formulation. This is because the dissipation on a grid point is scaled with the number of attached faces. In 2D the faces orthogonal to the offset direction are removed from the dual grid. Using this switch the number of attached faces is artificially increased by 2. The effect is that results between 2 and 3D can be better compared. Note, per default this modification is switched off. The intention is to allow users for evaluating this option, before we change the default or hardcode the modification without parameter switch in future

`Use new multigrid (0/1):` integer, default 0

- Controls one feature of the multigrid algorithm, namely whether eddy viscosity and central dissipation terms are updated to correspond to the actual solution before constructing the MG forcing term or not updated. After our understanding the most correct behavior is to update these terms, and this is selected with switch on (1) for using the new multigrid computation. However not updating them as was historically the case in TAU (e.g. release 2007) saves about 10% CPU time, and therefore it corresponds to the default value of 0.

`Use parallel initial partitioner (0/1):` integer, default 1

- By default, the new parallel initial partitioner is switched on. It becomes active when partitioning from 1 to N domains. It works in parallel and scales w.r.t. memory consumption. The grid is distributed across the partitions during the import of the data. At no time the complete grid is stored on a single process. Because this partitioner is not yet suited for grids containing a a grid hierarchy an internal logic switches back to the old parallel partitioner. The old one computes the partitions in parallel but stores initially the complete grid on process 0, which requires to have enough memory

on the first node. In cases that problems are encountered with the new partitioner, this parameter allows to always use the old partitioner to circumvent the problem. In all other cases the new partitioner is preferable.

`Use phi,le/phi,geo (0/1) to modify cp:` integer, default 0

- Switch to chose which sweep angle is used for the calculation of the theoretical value of the stagnation pressure to modify the pressure distribution (Modify cp for Coco input).
  - 0: leading edge sweep angle extracted from Tau grid is used,
  - 1: the so called geometrical sweep angle is used.

`Use Prepcp (0/1/2):` integer, default 0

- 0: Prepcp is not used — 1: Prepcp is only used to determine effective sweep angle and Re,theta,al — 2: Prepcp is used to modify cp-distribution.

`Use separation point for task10 (0/1):` integer, default 0

- Switch, to decide if task 10 of Lilo will be executed inside separation (1) or not (0).

`User defined filter width:` float, default 0

- If a user-defined filter width is chosen (see "Filter width model") this parameter specifies the required filter width. Note that only constant filter widths are set at present so that there is no facility for user-defined filter width to be expressed as a function of various parameters.

`Velocity factor for BL edge:` float, default 0.95

- The theoretical boundary layer edge velocity is calculated from the surface pressure. The boundary layer edge is detected when *0.xy, i.e. xy%* of this value is reached.

`Venkatakrishnan limiter constant:` float, default 0

- Airbus modification which can result in smoother convergence behavior, particularly when range of flow speeds in problem is significant. However this is not a form of low Mach number preconditioning.

`Viscous calculation (0/1):` integer, default 0

- This parameter defines if an Euler computation (0) or a laminar Navier-Stokes calculation (1) is performed when using the ptau3d.el solver. If a turbulent solver is compiled the parameter has no effect.

`Viscous flux type TSL/Full (0/1):` integer, default 1

- Evaluation of the viscous fluxes using the Thin Shear-Layer (TSL) approximation, or using a "full" gradient based approach. The default value should always be used by non-experts.

`Vortical flow correction (0/1):` integer, default 0

- This flag activates a submenu which provides additional control for the vortical flow correction algorithms. This flag must be set to use a vortical flow correction model.

`Vortical flow correction model:` character string, default 0

- This parameter allows the user to choose a vortical correction model. For the SA model variants the allowed options are "flower" and "sarc". For the two-equation model variants, the allowed option is "NLR". Detail of the models can be found in the turbulence section of the user guide.

`Vorticity factor for BL edge:` float, default 0.001

- The boundary layer edge is detected when the ratio between local vorticity and maximum vorticity inside the boundary layer is below this values.

`Wanted y+:` float, default 1

- Desired overall y^+-value of the resulting mesh, does not override the settings of 'Target y^+' made in the boundary mapping.

`Which modes are used:` array-int, default 0

- Specifies which of the modes contained in the 'Amplitude filename' file should be used for calculating the modal deformation.

`Window size for pmin/pmax search:` integer, default 5

- Half size (in numbers of streamline points) of the window for the search of pressure minimum and pressure maximum of a streamline.

`Write additional contourline data to file (0/1):` integer, default 0

- Write data of each contourline to a file. The data contains the spanwise position of the contourline (dimensionless, see Half span reference length and in grid units), the sweep angles (effective, leading edge, trailing edge), the local Reynolds number, the attachment line Reynolds number and informations on the transition, separation, etc. location on the contourline.

`Write boundary layer profiles to file (0/1):` integer, default 0

- Write boundary layer profiles data of each streamline to a file.

`Write graph filename:` character string, default (none)

- The name (including the path) of the output ASCII-file containing the grid-connectivity information in a 'graph-file format' which is supported by external grid partitioners (e.g. the public domain tools 'metis' or 'chaco'). After writing the graph file the preprocessing program is continued: the grid is partitioned using the prescribed internal grid partitioner.

`Write pointdata dimensionless (0/1):` integer, default 0

- Per default the output of the solution data is in SI-units. This parameter writes the data in non-dimensionalized form. The scaling to SI-units is stored as well to enable a re-dimensionalization in successive tools. Restarting from a non-dimensionalized solution will not invoke any scaling, but use the dimensionless values and interpret them as scaled with the references specified in the current parameter file. Therefore, continuing from a dimensionless or SI-units restart file will differ, if the references in the parameter file are changed at the same time. Usually the references should not be changed during a calculation to keep the data consistent. If you switch to references different from the references used to generate the restart solution, be aware of the two different ways of proceeding and make sure that your choice provides you with the intended results.

`Write streamline data to file (0/1):` integer, default 1

- Write boundary layer data of each streamline to a file.

`XLES constant:` float, default 0.07

- The parameter controls the scaling constant for the dissipation length scale and eddy viscosity length scale for the LES branch of XLES. As with the SA-DES models, the coefficient is calibrated against the DIT test case. In general the same limitations in modification of this coefficient arise as for the SA-DES model. Note that the original developers (NLR) recommend that the coefficient is initially tuned down to allow the development of sustainable turbulent structure. After some time, typically of the order of some integral time scales (to ensure good turbulent mixing), the coefficient is then increased (gradually) to ensure that the calibrated value (which is known to return a proper energy cascade across neighbouring wavenumbers) is used.

4.3 Boundary mapping parameters

All the parameters needed to define the boundary conditions are contained in the 'Boundary mapping file'. The name of this file needs to be contained in the parameter file using the **Boundary mapping filename** parameter (see Section 4.1). The complete boundary mapping file can be placed into the general parameter file, if the handling of all parameters within one file is preferred. In this case the **Boundary mapping filename** parameter has to be the filename of the general parameter file and the name inside this file has to be the filename itself. To avoid renaming the parameter when renaming the file, the key '(thisfile)' can be used, which is then renamed automatically when reading the file.

A boundary map is a unique relation between a list of boundary markers given in the primary grid and a boundary part of a special type (treatment) known by the solver. The names of boundary treatments which can be assigned to boundary markers are listed in alphabetical order below.

- `actuation`
  Defines a boundary for flow actuation (blowing, suction or a combination of both).
- `actuator inflow`
  Defines the inflow for an actuator disk. An actuator disk is a device which produces a discontinuity in the flow properties. For example, this can be used to simulate a propellor, or the fan in a turbofan engine, where the total pressure, temperature and swirl are discontinuous across the propellor/fan.
- `actuator exhaust`
  Defines the outflow for an actuator disk.
- `axisymmetry axis`
  Defines a boundary plane as an axis for axisymmetric simulations. The boundary should ideally be one cell wide, with the width being very small compared to the grid length scale. Cell normals should point in the radial direction.
- `chimera`
  Defines an artificial boundary inside the computational domain.
- `dirichlet`
  User defined setting of all values at the boundary.
- `engine exhaust`
  Defines an engine exhaust plane.
- `engine inflow`
  Defines an engine inflow plane.

- `euler wall`
  Defines a wall without viscous effects.

- `exit-pressure outflow`
  Defines an outflow boundary for internal flow, where all variables are extrapolated from the interior domain except the static pressure which is prescribed by a parameter.

- `farfield`
  The farfield is an inflow/outflow boundary far away from the investigated configuration for external flow. The presence of the configuration should hardly influence the state of the flow variables at the boundary. All gradients are assumed to be zero and therefore no viscous effects are taken into account.

- `laminar wall`
  Defines a laminar wall with viscous effects.

- `mirror plane`
  Defines a symmetry plane, where the symmetry is handled by creating extra points in the preprocessing stage which lie on the outside of the symmetry plane. These additional points are a mirror copy of the points connected to the symmetry plane in the original grid. In the solver the flow variables are copied to these additional points with the appropriate transformation. A maximum of 2 mirror planes per grid can be used.

- `periodic plane`
  Defines a periodic plane. The TAU-Code can handle either one translational or one rotational periodic boundary condition per grid. Rotational periodicity may also include a symmetry axis (i.e. where the two halves of the periodic boundary join on the axis of rotation). Adaptation of periodic boundaries is restricted to grids with both points of the periodic pairs in the same domain, e.g. in sequential cases and for actuator discs.

- `reservoir-pressure inflow`
  Defines an inflow boundary for internal flow with prescribed constant total pressure and total density. The inflow direction is by default perpendicular to each boundary face, or it can be set to a desired direction.

- `sharp edge`
  Special boundary treatment for euler flows around sharp (trailing) edges.

- `supersonic inflow`
  Defines a very simple setting of all values given by the user. Due to the theory of characteristics, this treatment is only valid if a supersonic velocity is defined entering the computational domain. Instead of the farfield treatment, this simplified treatment can be used for many boundary parts of the same grid with different values.

- `supersonic outflow`
  Defines a very simple extrapolation of all values from the interior. Despite the theory of characteristics, this treatment is very robust even if a boundary layer is leaving the computational domain. When a supersonic outflow abuts a viscous wall, this treatment should be selected in preference to the farfield boundary condition.

- `symmetry plane`
  Defines a symmetry plane, where the symmetry is handled by projecting the momentum flow variables onto the symmetry plane.

- `turbulent wall`
  Defines a turbulent wall with viscous effects.

- `viscous wall`
  Defines a wall with viscous effects. The type of viscous wall (i.e. laminar, transition or turbulent), is read in as a parameter.

The format of the file is that each boundary part is defined as a parameter block, where the end of a block is marked with the keyword 'block end' on a single line (see page 141 for an example boundary mapping file). Each block is read like the global parameter file. Therefore, once the preprocessing has read the parameter file the user should not change:

- the number of boundary parts
- the order of the boundary parts
- the type of the boundary parts

All other parameters can be changed without running into problems. Obviously, once the preprocessing is finished, the parameters needed only by the preprocessing could be changed or removed from the parameter file, because this information is contained in the dual grids. But the user should leave them unchanged to enable another preprocessing step, for example after an adaptation of the grid.

The agglomeration process to provide the coarser multigrid levels prohibits the fusing of disjunct boundary parts. Therefore the definition of different boundary parts with the same parameters (except the markers) becomes necessary to avoid coarse grid cells which connect for example different symmetry planes or a wing with its flap.

There are two categories of boundary mapping parameters, those which are common to every boundary mapping, and those specific to the type of boundary mapping. These parameters are summarized in the tables below, with the description of each parameter being provided in the 'Input boundary mapping parameters' of the relevant preprocessing, solver or adaptation section.

In comparison to older versions (Rel 2002.0.2 and older) the output of integral boundary part values is now no longer controlled by the Monitor list parameter. Rather, flags are now set in the bmapping lists. At present the following integral quantities are printed to stdout depending on parameter settings in the relevant boundary part mapping list;

- monitoring of a mass flux normal to a bpart surface is controlled through the **Monitor mass flow (0/1)** parameter,
- monitoring of the integrated pressure across a bpart surface is controlled through the **Monitor pressure (0/1)** parameter,
- and monitoring of impulse force at a boundary part surface is controlled through the **Monitor impulse (0/1)** parameter,
- for the engine boundaries the engineering quantity WAT may be monitored using **Monitor WAT (0/1)**.

4.3.1 Common boundary mapping parameters

The following parameters are common for every boundary part:

parameter name	type	range	default	in	page
Copy	string	-	-	s, p	149
Cutting plane allowed (0/1)	int	0, 1	0	s	176
Freeform box number	int	$0 \leq$ i	-1	d	306
Markers	list	$0 \leq$ i	not_defined	p, a, d	149
Multigrid priority	int	0, 1	treatment dependent	p	149
Name	string	-	(none)	s, p	149
Partname	string	-	(none)	s	176
Surface deformation type (0/1/2)	int	0, 1, 2	(none)	d	307
Type	string	-	-	s, p, a, d	149

4.3.2 Specific boundary mapping parameters

The parameters relevant to each of the boundary treatments are summarized below.

Actuation This boundary treatment needs the existence of an actuation boundary and the following parameters:

parameter name	type	range	default	in	page
Actuation direction	array-float	-	{0, 0, 0}	s	177
Actuation duty cycle	float	-	0.5	s	177
Actuation initial time shift	float	-	0.0	s	177
Actuation period	float	-	0.0	s	177
Actuation subtype	string	-	(none)	s	177
Actuation type	string	-	(none)	s	179
Compressible jet (0/1)	int	>0	0	s	182
Consistent pressure treatment (0/1)	int	-	0	s	182

continued on next page

parameter name	type	range	default	in	page
Extrapolate turbulent field data at inflow (0/1)	int	0, 1	0	s	182
Jet cut-off factor	float	>0	0.05	s	182
Jet diameter	float	> 0	-	s	183
Jet fluid density	float	>0	-	s	183
Jet fluid temperature	float	>0	-	s	183
Jet improved model for eddy viscosity (0/1)	int	0, 1	1	s	183
Jet Mach number from isentropic area-Mach number relation	float	>0	-	s	183
Jet ratio mue-t/mue-l	float	50-500	200.0	s	183
Jet skew angle beta	float	-3.14/2-+3.14/2	0.0	s	183
Jet total density max	float	>0	-	s	184
Jet total pressure max	float	>0	-	s	184
Jet total pressure min	float	>0	-	s	184
Jet turbulent intensity	float	0.001-0.5	0.05	s	184
Jet turbulent kinetic energy cut-off factor for turbulent viscosity	float	> 0	1.0e-3	s	185
Jet turbulent viscosity to jet velocity exponent	float	1.0-2.0	1.0	s	185
Jet yaw angle alpha	float	-3.14/2-+3.14/2	0.0	s	185
Maximum jet velocity	float	0 < x	0.0	s	185
Monitor forces (0/1)	int	0, 1	1	s	185
Prescribe fluxes (0/1)	int	0, 1	0	s	186
Prescribe time dependent eddy viscosity (0/1)	int	0, 1	1	s	186
Relaxation factor pressure correction	float	0 ≤ x	0.001	s	186
Restart use mean pressure (0/1)	int	0 ≤ x	0	s	186
Simple suction (0/1)	int	0, 1	0	s	186
Smoothing pulsed signal	float	0.0-0.4	0.025	s	187

continued on next page

parameter name	type	range	default	in	page
User defined actuation direction (0/1)	int	0, 1	0	s	187

Actuator inflow This boundary treatment needs the existence of an actuator disk outflow boundary and the following parameters:

parameter name	type	range	default	in	page
Blank ratio	float	$0 < x < 1$	0	p	149
Input file	string	-	(none)	s	187
Monitor mass flow (0/1)	int	0, 1	0	s	187
Number of blades	int	$0 \leq i$	0	s	187
Pitch angle	float	$-90 < x < 90$	0	s	188
Pitch point	float	$0 < x < 1$	0	s	188
Propeller rpm	float	$0 \leq x$	0	s	188
Write surface data (0/1)	int	0, 1	0	s	187

Actuator exhaust This boundary treatment needs the existence of a farfield boundary and the following parameters:

parameter name	type	range	default	in	page
Monitor mass flow (0/1)	int	0, 1	0	s	187
Write surface data (0/1)	int	0, 1	0	s	187

Axisymmetry axis

parameter name	type	range	default	in	page
Set gradients (0/1)	int	0, 1	0	s	188
Write surface data (0/1)	int	0, 1	0	s	187

Dirichlet

parameter name	type	range	default	in	page
Angle alpha (degree)	float	-	0	s	188
Angle beta (degree)	float	-	0	s	189

continued on next page

parameter name	type	range	default	in	page
Density	float	$0 < x$	reference state	s	189
Dirichlet data file	string	-	(none)	s	189
Mach number	float	$0 \leq x$	reference state	s	189
Monitor mass flow (0/1)	int	0, 1	0	s	187
Set gradients (0/1)	int	0, 1	1	s	189
Sideslip angle (degree)	float	-	0	s	189
Temperature	float	$0 < x$	reference state	s	189
Velocity	float	$0 \leq x$	reference state	s	189
Write surface data (0/1)	int	0, 1	0	s	187

Engine exhaust This boundary treatment requires the existence of a farfield boundary and takes the following parameters:

parameter name	type	range	default	in	page
Curvature reconstruction (0/1)	int	0, 1	1	a	292
Engine number	int	$0 \leq i$	1	p	150
Exhaust face mid point	array-float	-	{0.0, 0.0, 0.0}	s	190
Monitor mass flow (0/1)	int	0, 1	0	s	187
Monitor WAT (0/1)	int	0, 1	1	s	190
Number of polygon points for swirl	int	$0 \leq i$	0	s	190
Outflow condition type	string	Fixed_epsfan, Fixed_pressure, Fixed_massflow	Basic	s	191
Pressure ratio	float	$0 \leq x$	variable	s	191
Ratio mue-t/mue-l	float	$0 \leq x$	0.1	s	191
Swirl mid point distance of polygon point	array-float	$0 \leq x$	0.0	s	191
Swirl ratio of polygon point	array-float	$0 \leq x$	0.0	s	191
Target massflow	float	$0 \leq x$	0.0	s	191
Temperature ratio	float	$0 \leq x$	1	s	191

continued on next page

parameter name	type	range	default	in	page
Turbulent intensity	float	$0 \leq$ x	0.001	s	192
Write surface data (0/1)	int	0, 1	0	s	187

Engine inflow This boundary treatment no longer requires the existence of an engine exhaust boundary, in contrast to the situation in previous *TAU-Code* versions. In practice this boundary treatment implements one of three boundary conditions, which are characterized by the single flow quantity set on the inflow plane: ϵ_{fan}, pressure or massflow. The choice is made using the parameter `Inflow condition type`. Each of these conditions has its own parameter set and the possibility of setting various flow quantities. For the pressure based b.c. the engine pressure or mass flow may be specified by the user; the later being obtained by means of an iterative process. For the mass flow b.c. either the mass flow or the WAT value may be specified. Using the ϵ_{fan} b.c. only ϵ_{fan} may be specified.

parameter name	type	range	default	in	page
Area ratio eps_fan	float	$0 \leq$ x	0	s	192
Curvature reconstruction (0/1)	int	0, 1	1	a	292
Engine inflow direction	array-float	-	{0, 0, 0}	s	192
Engine number	int	$0 \leq$ i	1	p	150
Extrapolation type simple/characteristic (0/1)	int	0, 1	1	s	192
Fixed massflow	float	$0 \leq$ x	-1	s	192
Fixed WAT	float	$0 \leq$ x	-1	s	192
Fixed/initial pressure	float	$0 \leq$ x	Reference pressure	s	192
Inflow condition type	string	Fixed_epsfan, Fixed_pressure, Fixed_massflow	Fixed_pressure	s	192
Massflow convergence residual	float	$0 \leq$ x	1.0e-3	s	193
Measurement coordinates	array-float	-	{0, 0, 0}	s	193
Monitor mass flow (0/1)	int	0, 1	0	s	187
Monitor WAT (0/1)	int	0, 1	1	s	190
Regulator (0/1)	int	0, 1	0	s	194
Relaxation factor	float	$0 \leq$ x	0.1	s	194

continued on next page

parameter name	type	range	default	in	page
Solve quadratic eqn for rho/p (0/1)	int	0, 1	0	s	194
Type of mass coupling	string	Outflow_massflow, Fixed_massflow, None	None	s	194
Write surface data (0/1)	int	0, 1	0	s	187

Euler wall

parameter name	type	range	default	in	page
Curvature reconstruction (0/1)	int	0, 1	1	a	292
Monitor farfield state (0/1)	int	0, 1	0	s	195
Monitor forces (0/1)	int	0, 1	1	s	185
Monitor mass flow (0/1)	int	0, 1	0	s	187
Transpiration velocity file	string	-	(none)	s	195
Wall pressure correction (0/1)	int	0, 1	0	s	195
Write surface data (0/1)	int	0, 1	0	s	187

Exit-pressure outflow When `Match measured pressure` is set to one, the value of `Exit pressure` is only used for restart, after that it's calculated. The solver will set the value of `Exit pressure` so that the field pressure at the `Measurement coordinates` will match the value of `Measured pressure` when the computation has converged.

parameter name	type	range	default	in	page
Exit pressure	float	$0 < x$	reference state	s	195
Match measured pressure (0/1)	int	0, 1	0	s	195
Matching adjustment factor [0,1]	float	$0 \leq x \leq 1$	0.5	s	196
Matching iteration period	int	$0 \leq i$	150	s	196
Matching iteration start	int	$0 \leq i$	"5 times the given 'Matching iteration period"	s	196
Measured pressure	float	$0 < x$	reference state	s	196
Measurement coordinates	array-float	-	{0, 0, 0}	s	193

continued on next page

parameter name	type	range	default	in	page
Monitor farfield state (0/1)	int	0, 1	0	s	195
Monitor impulse (0/1)	int	0, 1	0	s	196
Monitor mass flow (0/1)	int	0, 1	0	s	187
Monitor pressure (0/1)	int	0, 1	0	s	196

Farfield The farfield state is defined by the farfield boundary part appearing in the boundary mapping file first. Following definitions are ignored except the parameter 'Name' prescribes to solve a Riemann problem. If no restart file exists, the farfield state is used to initialize the flow field.
Attention: All TAU output parameters are non-dimensionalized by the general reference state. **In case the farfield state is defined differently from the general reference state, the non-dimensional coefficients like C_L, C_D or C_p differ from their common definition based on the farfield state! Even round-off differences may have an impact of the coefficient values!**
The parameters to describe the farfield outside the computational domain are:

parameter name	type	range	default	in	page
Adjustment factor	float	$0 \leq x \leq 1$	0.01	s	197
Angle alpha (degree)	float	-	0	s	188
Angle beta (degree)	float	-	0	s	189
Chord length	float	$0 \leq x$	0	s	197
Constant alpha/clift (0/1)	int	0, 1	0	s	197
Density	float	$0 < x$	reference state	s	189
Lift iteration period	int	$0 \leq i$	20	s	197
Lift iteration start	int	$0 \leq i$	200	s	198
Mach number	float	$0 \leq x$	reference state	s	189
Modify farfield file	string	-	(none)	s	198
Monitor farfield state (0/1)	int	0, 1	0	s	195
Set gradients (0/1)	int	0, 1	0	s	199
Sideslip angle (degree)	float	-	0	s	189
Targeted clift	float	-	0	s	199
Temperature	float	$0 < x$	reference state	s	189
Velocity	float	$0 \leq x$	reference state	s	189

continued on next page

parameter name	type	range	default	in	page
Vortex correction (0/1)	int	0, 1	0	s	200
Write surface data (0/1)	int	0, 1	0	s	187

Mirror plane

parameter name	type	range	default	in	page
Plane normal vector	array-float	-	{0, 0, 0}	p	151

Periodic plane The periodic boundary implementation assumes that the periodic plane consists of pairs of points which can be identified in preprocessing using the periodic parameters provided by the user. Therefore, if the two sides of the periodic boundary have different markers (e.g. 1 for the left side, and 2 for the right) then both these markers must be in the same boundary part, otherwise the periodic pairs cannot be found. An exception to this is when the NetCDF primary grid file has been generated by Centaur, where the periodic pairs, as well as the Periodic translation vector or Periodic angle (degree) and Axis of rotation parameters are automatically read in from the NetCDF file and do not have to be supplied.

Experience has shown that the quality of the duals grids with periodic boundaries obtained after the preprocessing stage is highly dependent on the quality of the grid generator. For this reason it is recommended that before using the preprocessing, the periodic grid be checked (and corrected if necessary) using the *setup_taugrid* utility program. This program is briefly described on page 328, and ensures that the points on the periodic boundary are periodic to machine accuracy.

parameter name	type	range	default	in	page
Axis of rotation	array-float	-	{0, 0, 0}	p	151
Marker index	int	$0 \leq$ i	0	p	151
Monitor mass flow (0/1)	int	0, 1	0	s	187
Periodic angle (degree)	float	$0 \leq$ x ≤ 180	0	p	151
Periodic epsilon value	float	$0 \leq$ x	1e-13	p	151
Periodic translation vector	array-float	-	{0, 0, 0}	p	151
Write surface data (0/1)	int	0, 1	0	s	187

Reservoir-pressure inflow The parameters to describe flow field outside the computational domain are:

parameter name	type	range	default	in	page
Inflow direction	array-float	-	{0, 0, 0}	s	200
Monitor farfield state (0/1)	int	0, 1	0	s	195
Monitor forces (0/1)	int	0, 1	1	s	185
Monitor impulse (0/1)	int	0, 1	0	s	196
Monitor mass flow (0/1)	int	0, 1	0	s	187
Monitor pressure (0/1)	int	0, 1	0	s	196
Redirect velocity (0/1)	int	0, 1	1	s	201
Total density	float	$0 < x$	reference state	s	201
Total pressure	float	$0 < x$	reference state	s	201

Supersonic inflow

parameter name	type	range	default	in	page
Angle alpha (degree)	float	-	0	s	188
Angle beta (degree)	float	-	0	s	189
Density	float	$0 < x$	reference state	s	189
Mach number	float	$0 \leq x$	reference state	s	189
Monitor mass flow (0/1)	int	0, 1	0	s	187
Set gradients (0/1)	int	0, 1	1	s	189
Sideslip angle (degree)	float	-	0	s	189
Temperature	float	$0 < x$	reference state	s	189
Velocity	float	$0 \leq x$	reference state	s	189
Write surface data (0/1)	int	0, 1	0	s	187

Supersonic outflow

parameter name	type	range	default	in	page
Monitor mass flow (0/1)	int	0, 1	0	s	187
Set gradients (0/1)	int	0, 1	1	s	189
Write surface data (0/1)	int	0, 1	0	s	187

Symmetry plane

There are no specific parameters for symmetry plane.

Viscous wall

parameter name	type	range	default	in	page
Coolant reservoir temperature	float	$0 < x$	reference state	s	203
Curvature reconstruction (0/1)	int	0, 1	1	a	292
Effusion mass flux	float	$0 \leq x$	-1.0	s	203
Emissivity	float	$0 \leq x \leq 1$	0.8	s	203
Heat flux	string	adiabatic, isothermal, radiative_equilibrium	adiabatic	s	203
Heat flux correction (0/1)	int	0, 1	0	s	204
Heat flux evaluation	string	gradient, old_gradient, energy_balance	gradient	s	204
Monitor farfield state (0/1)	int	0, 1	0	s	195
Monitor forces (0/1)	int	0, 1	1	s	185
Monitor heat flow (0/1)	int	0, 1	0	s	204
Monitor mass flow (0/1)	int	0, 1	0	s	187
Moving wall (0/1)	int	0, 1	0	s	204
Moving wall omega	array-float	-	{(0, 0, 0)}	s	205
Moving wall origin	array-float	-	{(0, 0, 0)}	s	205
Moving wall trans	array-float	-	{(0, 0, 0)}	s	205
Set gradients (0/1)	int	0, 1	0	s	205
Structured layer refinement (0/1)	int	0, 1	1	a	293
Subtype	string	-	(none)	s, p	205
Target yplus	float	$0 \leq x$	1	s	205
Temperature	float	$0 < x$	reference state	s	189
Temperature filename	string	-	(none)	s	205
Use wall function (0/1)	int	0, 1	0	s, p	205
Write surface data (0/1)	int	0, 1	0	s	187

4.3.3 Sample boundary mapping file

A boundary mapping file could look like:

```
------------------------------------------------------------
BOUNDARY MAPPING
------------------------------------------------------------
                                                 Type: farfield
                                              Markers: 1
                                 Write surface data (0/1): 1
                               Cutting plane allowed (0/1): 1
                                      Angle alpha (degree): 1.0
                                       Angle beta (degree): 0

block end
------------------------------------
                                                 Type: symmetry plane
                                              Markers: 2
                                 Write surface data (0/1): 1
                               Cutting plane allowed (0/1): 1
block end
------------------------------------
                                                 Type: euler wall
                                              Markers: 3
                                                 Name: Wing
                                     Monitor forces (0/1): 1
                                 Write surface data (0/1): 1
                               Cutting plane allowed (0/1): 0
block end
------------------------------------
                                                 Type: viscous wall
                                              Subtype: turbulent
                                              Markers: 4
                                     Monitor forces (0/1): 1
                                 Write surface data (0/1): 1
                               Cutting plane allowed (0/1): 0
block end
------------------------------------
```

4.4 Transition parameters

All the parameters needed to define transition for boundary parts of type 'viscous wall' and subtype 'transition' are to be defined inside the **Boundary mapping file**.

The available parameters for defining transition are listed in the table below. The use of each of the parameters is described in the preprocessing section on page 152.

parameter name	type	range	default	in	page
Boundary part list	string	-	(none)	s	43
Boundary part namelist	string	-	(none)	p	153
Exclude surface normal/angle	array-float	any vector for 1st 3, 0 - 180 for 4th entry	{0, 0, 0, 0}	p	153
Laminar height	float	$0 < x$	0	p	154
Number of limiting planes	int	$0 < i$	0	s	??
Number of plane points	int	0, 3	0	p	154
Number of polyline points	int	$0 \leq i$	0	p	154
Number of streamline points	int	$0 \leq i$	0	s	??
Streamline type	int	1, 2, 3	2	s	??
Transition block name	string	-	(none)	s	115
Transition flow direction	array-float	any vector	{1, 0, 0}	p	154
Use contourlines of block	string	-	(none)	s	??

The format of the transition input is similar to that of the boundary mapping in that each set of transition information is defined as a parameter block, where the end of the block is marked with the keyword 'transition end' on a single line. Each block contains the user defined parameter. In addition point coordinates have to be defined, which are either to define a polyline or a plane. For this the keyword `TransitionCoordinates` has to be followed by a list of coordinate values for the X, Y and Z component before the end of the block ('transition end')

An example transition parameter input is provided on page 144.

The transition input allows to define either a plane or a polyline per block. Each block can be assigned to one or several transition boundary parts. Note: it is only allowed either to assign a set of planes or a set of polylines to a single boundary mapping. A set of polylines should define a closed line around or along a body which is outside the geometry defined by the boundary part. It is not necessary that the polyline coordinates are inside the computational grid. It is even allowed that the polylines are not connected, which is not recommended,

except e.g. in simple 2D configurations in which one polyline segment with one point for the upper part of an airfoil and another with one point for the lower part is sufficient. From the polyline input a transition line on the grid surface is computed by checking if a surface point is in front of it's nearest visible polyline segment or not. If the derived transition line is not closed, the algorithm ends with error messages (e.g. that a turbulent front hits the laminar region). This is because at the end all points not flagged previously which are surrounded by laminar points are flooded (i.e. flagged as laminar). This can only work if the laminar/turbulent boundary is closed!

Furthermore, it also can lead to undesired shapes of the transition lines if the polyline segments overlap. For that reason a closed line defined by one or more polygons is strongly recommended. The possibility of using several polylines for the definition of a transition line is in order to allow e.g. the variation of the laminar height along a transition line.

All points in a viscous wall boundary part, subtype transition, are initially marked as being turbulent. If a point is found to be on the laminar side of either the polyline or the plane which defines the point of transition, then the point is marked as laminar. A check can (and should) be made of the laminar/turbulent regions by using the field output value of 'wdist' in the solver. This value stores the distance that each point has from the nearest viscous wall. For those points that are both closest to a laminar wall, and within the laminar region ('Laminar height' parameter), the values of wdist are multiplied by -1.

4.4.1 Sample transition parameter input

A transition parameter input could look like this:

```
-----------------------------
      Boundary part namelist: flap
   Number of polyline points: 3
              Laminar height: 0.01
Exclude surface normal/angle: 0 0 1 90
TransitionCoordinates
1.066  -1. -0.0
1.066  0.0 -0.0
1.066  1.0 -0.0
transition end

      Boundary part namelist: flap
   Number of polyline points: 3
              Laminar height: 0.01
Exclude surface normal/angle: 0 0 -1 90
TransitionCoordinates
1.0  -1. -0.0
1.0  0.0 -0.0
1.0  1.0 -0.0
transition end

 -----------------------------
       Boundary part namelist: wing1 wing2 wing3
       Number of plane points: 3
               Laminar height: 0.2

TransitionCoordinates
0.4 -1.0 0.0
0.4  0.0 0.0
0.3 -1.0 0.0
transition end
-----------------------------
```

5 Preprocessing

5.1 Description

The preprocessing program reads a primary grid and creates a set of dual grids (edge based representation). The primary grid can be composed of tetrahedral, prismatic, hexahedral or pyramidal elements. The dual grids contain information about metric data, boundary types and neighboring domains.

When running the solver in multigrid mode, coarse grids are computed during preprocessing. In order to achieve a high performance in the flow computation on vector computers, a coloring algorithm for the edges of the hybrid grid is employed to avoid vector dependencies or to optimize cache loads. For parallel computations grid partitioning can be performed. The use of external grid partitioners is supported by writing or reading the required information to files.

The usage of the preprocessing program is

```
bin-path/ptau3d.preprocessing parameter-file <log-file>
```

For employing the preprocessor in parallel mode the primary grid have to be decomposed in several partitions before the preprocessor starts. Therefore a parallel partitioner for primary grids is started first automatically from the preprocessor main-routine, if the number of processes is larger than one. Here, a check is performed if as many grid partitions are available in files (file name convention is gridfilename_domain_i, with i being the partition number) as processes are started for the preprocessor. If these files exist, the preprocessor reads the grid partitions and no parallel partitioning is performed. If the grid partition information is not available in files, the parallel partitioning starts and hands over the grid partitions to the preprocessor in memory without additional file-IO.

This procedure allows forcing the preprocessor to use already computed grid partitions. This is necessary at present when using chimera or periodic grids, because both options are not yet supported by the parallel partitioner. Computing grid partitions in advance can also be of advantage if the preprocessor needs to be restarted several times during a computation (e.g. because of deforming grid partitions without re-partitioning). Performing the partitioning before the first step only allows to save the CPU-time for the second and all following partitioning steps.

In order to compute grid partitions in advance an additional tool is available, which is described in more detail in the utility section. This tool can be started by:

```
bin-path/ptau3d.subgrids parameter-file <log-file>
```

To start the preprocessor in parallel mode the mpirun-script have to be used, e.g:

```
mpirun -np # bin-path/ptau3d.preprocessing parameter-file <log-file>
```

The MPI options and the name of the mpirun script depend on the MPI version. Please refer to the manuals of the MPI version installed on the target machine.

The internal switch in TAU-Code for the use of MPI is based on the number of command line arguments (MPI usage only if more than 3 command line arguments are given. Some MPI-installations (e.g. mpich) add extra arguments to the command line, others do not (e.g. NEC mpisx). Using such an installation (which becomes obviously when the usage described above does not work) requires to add extra arguments to the command line. Then, the usage is

```
mpirun -np # bin-path/ptau3d.preprocessing parameter-file log-file use_mpi
```

5.2 Prerequisites

The preprocessing algorithm can be run after the definition of a suitable grid in NetCDF-format and the definition of the boundary-mapping file. A suitable parameter file has to be defined in addition.

5.3 Input options

The input options are defined in the parameter file. All input parameters, input file names and output file names can be defined in this file. The parameter names and the file names are explained below. They are listed in alphabetical order. Parameters without default values are printed underlined.

5.3.1 Files

The major input file is the parameter file. It contains the names of all further input files listed below.

5.3.2 Input general parameters

parameter section

Runtime optimisation parameter section

partitioning parameter section

Agglomeration parameter section

Controls parameter section

5.3.3 Input boundary mapping parameters

Common boundary mapping parameters

Common

`Copy:` character string, default -

- This is not a real parameter of the boundary part but a keyword to link the definitions of different boundary parts. If the keyword followed by ':' is found, the parameters of the preceding boundary part definition are used as default parameters for the current boundary part. Even the 'required' parameters can be provided by the preceding definition block.

`Markers:` marker list, default not_defined

- The integers in the list are separated by ',' or '-' if a complete range (i.e. 3-6 = 3,4,5,6) is covered. Each marker specifies a set of surface elements in the primary grid.

`Multigrid priority:` integer, default treatment dependent

- The multigrid priority determines the order in which the boundary volumes are touched by the advancing front during the agglomeration procedure.

`Name:` character string, default (none)

- This name is only used by the solver to set up the NetCDF surface data attribute names, but it is also assists in identifying different 'wall' boundary parts. Typical names can be 'wing', 'pylon' or 'nacelle'.

`Type:` character string, default -

- The value of the type is the name of the boundary treatment. The number and the kind of additional parameters for a boundary part depend on the boundary treatment (see following subsections).

Actuator inflow

`Blank ratio:` float, default 0

- For actuator disks intersecting with viscous walls the 2D blade element theory cannot be applied in the boundary layer because the axial velocity of the flow becomes zero whereas the rotational speed of the model is not affected. As a consequence the effective angle of attack approaches the blade twist close to the wall which causes unphysically high values of the aerodynamic coefficients of lift and drag. The numerical scheme does not allow to set forces for the points of the intersection line. These points are excluded

by default from the set of points for which forces are set. For a smooth transition of the sectional load from the inner of the disk up to zero at the inter section line 1D ray coordinate systems are constructed for the intersection points normal to the wall with an advancing front algorithm. The advancing front ends if the radial distance to the intersection point exceeds the fraction of the outer radius of the disk specified by the parameter value or if the structured region of quadrilateral elements ends. The last point of each ray is the closest point to the wall for which forces are set. This force is then linearly scaled for the other points of the ray with the radial distance to the intersection point at the wall. The scaling also forces a smooth transition at a viscous wall for a prescribed distribution of sectional loads. If the distribution already takes into account a smooth transition this parameter should be set to zero. Ray coordinate systems are also created if the disk intersects at the outer radius with a viscous wall.

Engine exhaust
`Engine number:` integer, default 1

- This parameter can be used to associate engine inflow and engine exhaust boundary conditions. Each pair of an engine inflow / engine exhaust boundary condition with the same *Engine number* may have a different parameter setting.

Engine inflow
`Engine number:` integer, default 1

- This parameter can be used to associate engine inflow and engine exhaust boundary conditions. Each pair of an engine inflow / engine exhaust boundary condition with the same *Engine number* may have a different parameter setting.

Global
`Use Theta (0/1):` integer, default 0

- The flow solvers of the **TAU-Code** and **THETA** use different names of valid boundary treatments. Therefore, the user has to specify in the boundary mapping file which flow solver he intends to use. This can be done with this parameter, which can be placed at any line of the boundary mapping file.
  Contrary to all other parameters, this global boundary mapping parameter will not be printed to standard output.

Heat exchanger
`Heat exchanger number:` integer, default 1

- User defined ID of heat exchanger. Option is under development and not yet operational.

Mirror plane

`Plane normal vector:` array-float, default {0, 0, 0}

- Defines a vector normal to the symmetry plane.

Periodic plane

`Axis of rotation:` array-float, default {0, 0, 0}

- Along with the periodic angle, the axis of rotation defines the rotational periodic boundary. It is currently assumed that this axis passes through {0, 0, 0}. When no periodic angle and axis of rotation is given by the user, it is assumed that a translational periodic boundary exists.

`Marker index:` integer, default 0

- This parameter is intended to handle periodic boundaries which consist of a different number of markers for each half (periodic / shadow). For periodic boundaries the markers of both halves have to be specified in the same mapping. The markers are expected to be ordered for each half with the index specifying the start index of the second half either periodic or shadow. If the parameter is not set it is assumed that each half has the same number of boundary markers which is the standard. Up to now no such case occurred.

`Periodic angle (degree):` float, default 0

- The included angle between the two sides of a rotational periodic boundary. If this parameter is used, the axis of rotation must also be given. When no periodic angle and axis of rotation is given by the user, it is assumed that a translational periodic boundary exists.

`Periodic epsilon value:` float, default 1e-13

- A small positive (error) value used for the determination of periodic point pairs. Inaccurate grid generation may require a larger value of this parameter.

`Periodic translation vector:` array-float, default {0, 0, 0}

- The displacement vector for a translational periodic boundary condition. When no periodic translation vector is given by the user, it is assumed that a rotational periodic boundary exists.

Viscous wall

`Subtype:` character string, default (none)

- Available options are 'laminar', 'transition' or 'turbulent'.

`Use wall function (0/1):` integer, default 0

- Switch between low-Re and hybrid-Re treatment of turbulent viscous walls.
  - value = 0:
    Default implementation. Low-Reynolds boundary condition, which requires low-Re grid, i.e. $y^+(1) = 1$ for the first node above the wall. The no-slip condition for momentum is prescribed.
  - value = 1:
    Wall-function boundary condition. Hybrid-Reynolds boundary condition, i.e., the restriction concerning the location of the first off-wall grid node $y^+(1)$ is eliminated. The first off-wall node can reside anywhere in the logarithmic part of the boundary-layer or below, i.e., $0 \leq y^+(1) \leq y^+_{max}$, where y^+_{max} depends on the Reynolds number and should be at most 0.05 times the boundary layer thickness to ensure at least three grid nodes in the log-layer. This boundary condition prescribes the wall-parallel component of the wall-shear stress and the no-penetration condition for the wall-normal velocity.
    Best practice: For highlift configurations, a good compromise between accuracy and speedup is obtain if $y^+(1) = 10$. For transonic cruise flight conditions, $y^+(1) = 50$ can be used if focus is on fast solutions. For flows with strong separation the standard low-Re boundary condition is recommended. Moreover the parameter Omega boundary condition type (0/1/2) should be set to 2.
    * WARNING: Values of $y^+(1)$>80 may lead to a significant modeling error.

5.3.4 Input transition parameters

For the usage of the transition input parameters some knowledge about the algorithm is useful which defines laminar or turbulent regions on surfaces of subtype 'transition'. Two different options are available to define transition lines: either defining planes or polygon-lines to divide the selected surface boundary parts into a laminar and a turbulent region.

When applying the plane option, for each surface point of the selected boundary part it is computed whether the point is located on the upstream or the downstream side of the plane. The upstream side is considered to be the laminar part.

When applying the polygon-line option, the nearest segment of each polygon-line is computed for each boundary point. On this segment the nearest coordinate is computed (i.e. the vector between the segment point and the surface point is perpendicular to the segment line). If the

surface point is upstream it is considered to be on the laminar side.
If several polygon-lines are defined, the second can only mark points, which are not already set to laminar or turbulent by the first one. The same holds for all following polygon-lines. That points are not marked by one of the polygon-line definitions can only happen if an exclude-option is defined. An example for the use of an exclude option is the transition -line setting for a wing. When defining a polygon-line from the upper wing root to the tip and back on the lower side to the wing root again, points on the upper surface can be nearer to a polygon segment of the lower part than to than to those which are above the wing. This can, which can lead to undesired results. One remedy is to separate a wing surface into a lower and into an upper surface boundary part (eventually also into an extra part for the wing tip) and to define single polygon-lines for the single parts. Another remedy is to define separate polygon-lines for the upper and for the lower part and to exclude surface points for each polygon by defining an 'Exclude surface normal/angle'. The lower wing surface points can be excluded from the upper wing polygon by defining a normal towards the ground with an angle of 90 degrees (for more detail see the parameter description). With inverse setting the upper wing surface points can be excluded from the wing polygon.
Using exclude options requires that the user ensures that all points excluded in one polygon-line setting will be marked by another polygon-line. At least all points on the border between laminar and turbulent regions have to be marked, such that a closed region for both surface types is defined. After all polygon-line settings are processed an advancing front-type method is applied to assign all remaining surface points, which have not been marked before. If a 'turbulent' front hits a 'laminar' one (or vice versa) the finally computed transition-line between 'laminar' and 'turbulent surfaces' will be of an accidental shape, thus warning-messages are printed to stdout each time both different fronts hit each other. In such a case the polygon-line definition needs to improved if the obtained (accidental) result is not acceptable.

Transition

`Boundary part namelist:` character string, default (none)

- The transition data are matched to the boundary parts by matching the names given in this parameter and the names of the boundary parts. Therefore it is possible to define one transition block which is valid for more than one boundary part. As an example, refer to the Sample transition parameter input on page 144.

`Exclude surface normal/angle:` array-float, default {0, 0, 0, 0}

- This parameter can be used to select certain sections of a boundary part for which a polyline is not applicable. For example, (see introduction) when defining different transition-lines for the lower and the upper wing surface, this option is needed if both surfaces are contained in the same boundary part.
  The 'Exclude surface normal/angle' is an array containing 4 floats. The first three

define a vector, the 4th defines an angle (in degree). If these values are set to: 1.0, 0, 0, 60 all surface points with a normal which has an angle lower to 60 degrees to the vector (1, 0 , 0) will not be marked by the polygon-line to which this parameter is assigned.

`Laminar height:` float, default 0

- For each field point the distance is calculated to the nearest viscous wall boundary point, and when this boundary point is laminar the wall distance is multiplied by -1 to define a laminar region. When for a transition viscous wall this parameter is set, only the field points which are nearest to the laminar points on THIS transition viscous wall AND have a wall distance less than this value are set as being laminar.

`Number of plane points:` integer, default 0

- This number has to be 0 (in case that a polyline is defined) or 3, which is the number of points to define a plane. The input for plane point coordinates equals the input of polygon line coordinates. As an example, refer to the Sample transition parameter input on page 144.

`Number of polyline points:` integer, default 0

- The number of points in the polyline or 0 if a plane is defined in this transition block. This number has to match the number of coordinates (x, y, z components) which follows in the input. As an example, refer to the Sample transition parameter input on page 144.

`Transition flow direction:` array-float, default {1, 0, 0}

- This parameter defines the projection of each polygon line to the surface. Per default it points along the X-axis.
  If the scalar product of the vector between a surface point and its nearest visible polygon-line coordinate and of the flow direction vector is positive the surface point is considered to be on the laminar side. If the scalar product is negative or zero the point is assumed to be on the turbulent side. Thus, the polygon-line is projected inside a plane normal to the flow direction (which is for the default a plane of constant x). This holds also if the nearest point of the nearest polyline segment is not in the interior of the segment but if it is either the start point or the end point of the segment. Thus, if a polygon-line ends over the half of the surface, the transition line would have a constant x value (in case of the default flow direction) for the rest of the surface.
  For more information about the Transition parameters see also Transition parameterson page 142

5.4 Description of process information

The output information of the preprocessing program starts with a line indicating the version and the compile time of the code. Below is a list of the implemented boundary treatments.

This is followed by a list of all parameters, as they are defined in the parameter file or, if not, by their default values. Following that, the information read from the boundary-mapping file is printed.

This is followed by process information about reading the input grid and about the computation of the dual-grid data. For all (multi-) grid levels the dual grid dimensions are written together with the maximum computed surface integral. This information is of importance. The meaning of this surface integral value is: Each dual grid control volume is composed of several facets, which are constructed from the primary grid elements. The integral of all the normal vectors of the facets of one control volume should be zero. The maximum of integrals for the inner (marked with `inner`) control volumes and for the control volumes at the boundary (marked with `bdry`) are printed together with the point coordinates of that control volume (x, y and z coordinates). A large maximum integral value for the grid of level 1 (larger than about 1.e-09 multiplied by a characteristic length of the current configuration) indicates that something is wrong in the primary grid (e.g. a wrong connectivity, a hanging node or at least very bad shape elements). Such grids should not be used! At the end of the program output grid statistics are printed. In the case of using several domains for parallel computations, the different grid dimensions on different domains as well as the number of communication domains (indicated with commdomains) allow the parallel overhead, the load balancing and/or the quality of the domain decomposition to be determined.

6 Dualcell2plt

6.1 Description

The dualcell2plt program reads a primary grid and, if computed, a set of dualgrids, and creates an output file (asci format) containing the dualgrid cells on different multigrid levels. This file is ready for visualization of the dualcells with Tecplot.

If the dualgrids are not available, the preprocessing program is called.

It is possible to visualize piles of dualcells in the structured region of the computational domain.

The usage of the preprocessing program is

```
bin-path/ptau3d.dualcell2plt parameter-file <log-file>
```

For employing the dualcell2plt program in parallel mode the primary grid has to be composed in several partitions.
The dualcell can be specified either by its global id or by a set of x- y- and z-coordinates. In the latter case the global id of the nearest point to these coordinates is computed (in the corresponding computational domain when called parallel).
To start the dualcell2plt program in parallel the mpirun-script has to be used, e.g.:

```
mpirun -np # bin-path/ptau3d.dualcell2plt parameter-file <log-file>
```

The MPI options and the name of the mpirun script depend on the MPI version. Please refer to the manuals of the MPI version installed on the target machine.

6.2 Prerequisites

The dualcell2plt algorithm can be run after the definition of a suitable grid in NetCDF-format and of the boundary-mapping file. A suitable parameter file has to be defined in addition.

6.3 Input options

The input options are defined in the parameter file. All input parameters, input file names and output file names can be defined in this file.

6.3.1 Files

The major input file is the parameter file. It contains the names of all further input files listed below.

6.3.2 Input general parameters

partitioning parameter section

Number of domains (page 80)

Files/IO parameter section

Boundary mapping filename (page 43)
Grid prefix (page 60)
Primary grid filename (page 90)

Agglomeration parameter section

Number of multigrid levels (page 80)

Controls parameter section

Output level (page 85)

6.3.3 Sample parameter file

The parameter file could look like this:

```
Files/IO --------------------------------------: -
                          Primary grid filename: rae2822_hybrid.nc.ad.2
                       Boundary mapping filename: rae_bdrymap
                                     Grid prefix: dual
                          Dualcell plot prefix: dualcell
Preprocessing parameter ------------------------: -
                      Number of multigrid levels: 4
Dualcell visualisation -------------------------: -
                                 Dualcell number: -1
                           Dualcell coordinates: 0.51 -0.07 0.06
                        Max dualcell plot level: 3
                             Plot dualcell pile: 1
                            Plot neighbor piles: 0
                  Attach facenormal to dualcell: 0
                        Plot dualcell in domain: 2
Partitioning -----------------------------------: -
                               Number of domains: 4
Parameter --------------------------------------: -
                                    Output level: 25
```

6.4 Description of the process

After determining the id of the dualcell, all parents up to the maximal visualization level are detected. On each level all dualcells that have been agglomerated to those parent cells are computed. This is equivalent to computing all primary grid points that are located inside the parent cell. These points are stored in a separate list for each multigrid agglomeration level.

On each level all primary grid elements, that contain at least one of the before computed points, are computed and also stored in separate lists. Thus one gets a list of all elements that have a nonempty intersection with the (agglomerated) dualcell.

In a loop over all those primary grid elements, the elements, that contain at least one point that is not in the corresponding point list, are detected. For the corresponding edges the barycenters of the edge, the two adjacent faces and the element are computed just like inside the preprocessing routine. The two triangles that are thereby derived are stored. This means the coordinates of the points and their connectivity.

If a pile of dualcells is visualized, the corner points of the above mentioned triangles are stored in a common list on each level. The other lists are separate for each cell in the pile.

The output file is in ASCI Tecplot format. Each cell is written in a separate zone, such that they can be independently switched on and off in Tecplot. If a single dualcell is visualized, the dualcells on all agglomeration levels are written in a file with the name '[Dualcell plot prefix]_cell[Dualcell number]'. If a pile of dualcells is visualized, all cells on one multigrid agglomeration level are written into one file and the levels are written into different files with the names '[Dualcell plot prefix]_pile[Dualcell number]_level[i]' with i = 1,...,[Max dualcell plot level].

6.4.1 Output information

The output information of the dualcell2plt program starts with a line indicating the version and the compile time of the code. If the dualcell number is given via coordinates, the coordinates of the computed point are displayed. In the visualization statistic the number of cells in the pile on each computed level is displayed and the parent cells of each cell are listed.

```
DLR Tau-Code, DualcellVisualisation, Version compiled on Mar  1 2007
------------------------------------------------------
Read grid 'rae2822_hybrid.nc.ad.2_domain_2' ...

Read grid 'dual_domain_2_grid_1' ...

Read grid 'dual_domain_2_grid_2' ...
```

```
Read grid 'dual_domain_2_grid_3' ...

Search for point with coordinates

                        0.510000
                        -0.070000
                        0.060000

results in Primgridgpoint 568 with coordinates

                        0.507398
                        -0.070377
                        0.061825

This point has globalid 3907

 Maximum number of layers is 24
 Minimum number of layers is 24

VISUALISATION STATISTIC

Level 1: 25 cells in pile
Level 2: 13 cells in pile
Level 3: 7 cells in pile

Cellnumbers on
   Level 1   Level 2   Level 3
     16096      2329       333
     15449      2155       291
     14802      2155       291
     14155      1973       291
     13508      1973       291
     12861      1791       241
     12214      1791       241
     11567      1609       241
     10920      1609       241
     10273      1427       191
      9626      1427       191
      8979      1245       191
      8332      1245       191
      7685      1063       141
```

```
      7038      1063       141
      6391       881       141
      5744       881       141
      5097       699        91
      4450       699        91
      3803       517        91
      3156       517        91
      2509       335        24
      1862       335        24
      1215       153        24
       568       153        24

write file dualcell_pile3907_level1.plt

write file dualcell_pile3907_level2.plt

write file dualcell_pile3907_level3.plt

Timing for: 'total_time'
  this time in seconds: real 7.919e+00, user 6.552e+00 sys 1.040e-01
 total time in seconds: real 7.919e+00, user 6.552e+00 sys 1.080e-01
Wallclock runtime 7.92 sec

Done
```

6.5 Examples

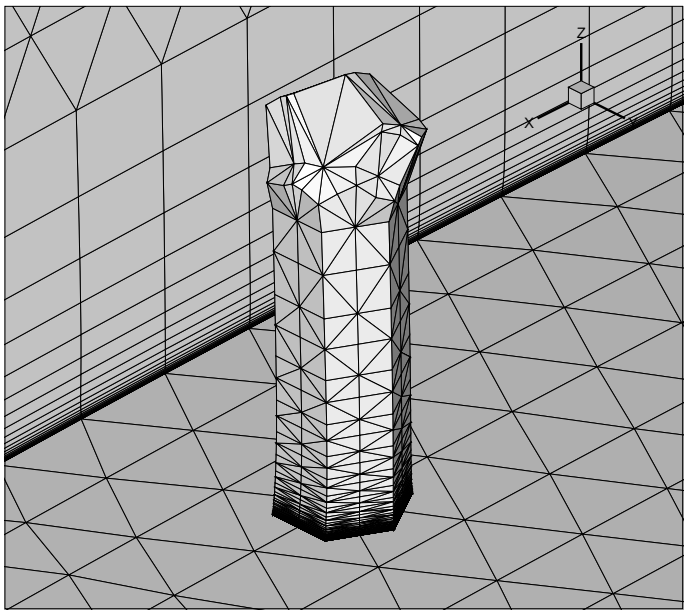


Figure 2: Pile of dualcells on multigrid level 1

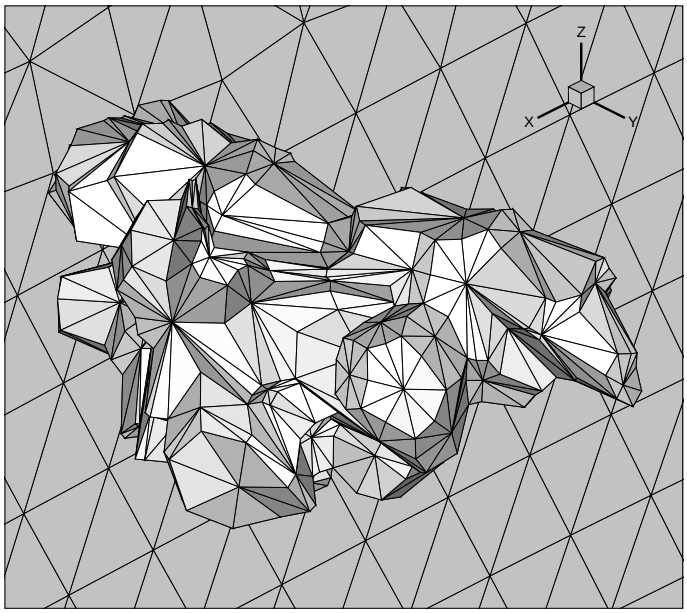


Figure 3: Dualcell in the unstructured region on multigrid level 2

7 Flow solver

7.1 Description

The flow calculation algorithm is based on the dual grid approach, which is well suited to three-dimensional hybrid grids. An edge-based data structure is employed to optimize the flow solver in terms of computer memory requirements and computational performance. The time-accurate three-dimensional Navier-Stokes equations are marched in time towards steady state by a Runge-Kutta time stepping method or a backward Euler implicit scheme solved with LU-SGS or SGS iterations. Local time stepping and multigrid is adopted to accelerate convergence.

The following spatial discretization schemes are available:

1. Classic central Jameson scheme on all grid levels.
2. Matrix dissipation scheme on fine grid and central Jameson on coarser grids.
3. Van_Leer upwind scheme
4. Hybrid AUSM-Van_Leer upwind scheme
5. AUSMDV upwind scheme
6. Roe upwind scheme
7. MAPS+ upwind scheme

The program is prepared to run with different numbers of equations. The basic version solves the 5 basic equations for Euler or laminar flow (abbreviation: `solver = el`). For employing a one- or a two-equation turbulence model (abbreviation: `solver = turb1eq` or `solver = turb2eq`) 6 or 7 equations respectively have to be solved. For the different turbulence models available refer to the subsection 'Turbulence Modeling'

The program is parallelized following the domain-decomposition approach. The parallel interface, which can be optionally applied, is MPI. For single processor mode the solver can be run without parallel interface, this means without mpirun.

The syntax for using the flow-solver program is

```
bin-path/ptau3d.(solver) parameter-file <log-file>
```

To start the solver in parallel the mpirun-script has to be used, e.g:

```
mpirun -np # bin-path/ptau3d.(solver) parameter-file <log-file>
```

The MPI options and the name of the mpirun script depend on the MPI version. Please refer to the manuals of the MPI version installed on the target machine. The internal switch in the TAU-Code for the use of MPI is based on the number of command line arguments (MPI usage only if more than 3 command line arguments are given. Some MPI-installations (e.g. mpich) add extra arguments to the command line, others do not (e.g. NEC mpisx). Using such an installation (which becomes obviously when the usage described above does not work) requires to add extra arguments to the command-line. Then, the usage is

```
mpirun -np # bin-path/ptau3d.(solver) parameter-file log-file use_mpi
```

In general all dimensional values should be specified in SI-units. However, to enable the use of grids which are not given in meters the grid scale can be specified as a solver parameter. This parameter is useful for grids which are non-dimensionalized by a characteristic length (such as a chord length, for example). Consequently all other parameters possessing a length dimension, including Reynolds length and the coordinates of the grid origin, have to be specified in grid units. Output of coordinates is also given in grid units to be consistent with the primary grid.

7.2 Prerequisites

The flow solver can be run after the dual-grid data have been processed using the preprocessing algorithm. A suitable parameter setting has to be defined in addition.

7.3 Input options

All parameters are defined in the general parameter file or in files specified in that file. The parameter names and the file names are explained below. They are listed in alphabetical order. Required parameters are printed underlined.

7.3.1 Input files

The major input file is the parameter file. It contains the names of all further input files listed below.

The parameters in the input file are the following:

7.3.2 Input general parameters

7.3.3 Input boundary mapping parameters

Common

`Copy:` character string, default -

- This is not a real parameter of the boundary part but a keyword to link the definitions of different boundary parts. If the keyword followed by ':' is found, the parameters of the preceding boundary part definition are used as default parameters for the current boundary part. Even the 'required' parameters can be provided by the preceding definition block.

`Cutting plane allowed (0/1):` integer, default 0

- If cutting plane output is desired, which outputs all data on cuts of a plane with the surfaces, it has to be defined here if this surface shall be taken into account. On viscous walls it defaults to 1.

`Name:` character string, default (none)

- This name is only used by the solver to set up the NetCDF surface data attribute names, but it is also assists in identifying different 'wall' boundary parts. Typical names can be 'wing', 'pylon' or 'nacelle'.

`Partname:` character string, default (none)

- This name can be used to assign a name for each single boundary. This name appears in the solver stdout-output for surface integrals. While the identifier 'Name' is used for groups of boundary parts (all parts having the same name) the 'Partname' helps to identify a single boundary part: e.g. if the Partname 'pylon' has the name 'wing' the force coefficients are integrated and displayed for the complete wing as well as for the single part 'pylon'.

`Type:` character string, default -

- The value of the type is the name of the boundary treatment. The number and the kind of additional parameters for a boundary part depend on the boundary treatment (see following subsections).

Actuation

`Actuation direction:` array-float, default {0, 0, 0}

- Direction of flow actuation specified by the three components (in x-, y- and z-direction) of jet velocity. Only used if the parameter User defined actuation direction (0/1) is activated. The direction is constant on each boundary marker.

`Actuation duty cycle:` float, default 0.5

- The actuation duty cycle is defined as the ratio of time where actuation is active (i.e., high flow rate time) to the actuation period. In case of harmonic blowing or pulsed blowing, the duty cycle is the ratio of time where blowing is active to the actuation period. In case of harmonic blowing or pulsed suction, the duty cycle is the ratio of time where suction is active to the actuation period. In case of zero net mass flux jet, it is defined as the ratio of time where blowing is active to the actuation period.
  Note that it is possible to choose a duty cycle different from 0.5. The following method is recommended in case that a jet with the following two properties is desired, viz., first a very large (e.g. supersonic) jet velocity and secondly a zero-net-mass-flux jet. Then a duty cycle smaller than 0.5 needs to be chosen. The massflow over one actuation period T is $\int_0^T \dot{m} dt = u_{jet} T_h + fac \times u_{jet} T_l$ with T_h being the time where blowing is active and $T_l = T - T_h$. The factor fac for multiplication of the target outflow jet velocity is set to T_h/T_l.

`Actuation initial time shift:` float, default 0.0

- For time-dependent blowing respectively suction, actuation start with the active part of the actuation cycle (i.e. with maximal jet velocity for pulsed actuation and with a sinus-type time-variation of velocity for harmonic actuation). This parameter allows to prescribe a time-shift t_{shift}.

`Actuation period:` float, default 0.0

- For time-dependent blowing/suction, this parameter gives the time for one actuation period.

`Actuation subtype:` character string, default (none)

- At the moment, the following alternative implementations are provided, which are designed for different physical situations of actuation inflow:
  - "standard_incompressible"
  - "standard_compressible"
  - "reservoir_pressure",
  - "convergent_divergent_nozzle"

In case of "standard_incompressible", the user has to specify 'Maximum jet velocity' and 'Jet fluid density'.
In all other cases, the user has to specify 'Jet total pressure max' and 'Jet total density max'.
In case of "standard_incompressible", density and jet velocity are prescribed. This boundary condition should be used if case that inflow at the actuation boundary is subsonic and effects of compressibility can be neglected. Pressure can be either extrapolated from the field or set to the reference pressure. Regarding the latter point, we recommend to set the parameter 'Consistent pressure treatment (0/1)' to zero.
In case of "standard_compressible", the pressure is extrapolated from the field, and for density and velocity suitable values computed from user specified total states are prescribed (see below). This boundary condition should be used if case that inflow at the actuation boundary is not supersonic but effects of compressibility cannot be neglected. Density is computed from the extrapolated pressure using isentropic relation for compressible flow $p/p_0 = (\rho/\rho_0)^\gamma$ where subscript $_0$ denotes total states. The (maximum) jet velocity is computed using the compressible version of Bernoulli's equation which states that under isentropic conditions total enthalpy $h = u^2/2 + e + p/\rho$ with $e = (\gamma - 1)p/\rho$ remains constant along streamlines. Therein h is specific enthalpy, u is jet velocity, and e is specific inner energy. (Sometimes this relation is referred to as relation by Saint Venant and Wantzel). We recommend to set the parameter 'Consistent pressure treatment (0/1)' to one.
In case of "reservoir_pressure" the same implementation as for the boundary condition type "reservoir-pressure inflow" is used, extended by the option that harmonic and pulsed blowing are also supported.
The option "convergent_divergent_nozzle" is designed for cases where the actuation slot is a convergent-divergent nozzle and that inside the nozzle flow is accelerated from subsonic to supersonic flow. Then the parameter 'Jet Mach number from isentropic area-Mach number relation' also needs to be specified.
This boundary condition is based on the relations for isentropic flow through convergent-divergent nozzles. Massflow $\dot{m} = \int\int \rho\vec{u}\cdot\vec{n}\mathrm{d}S$ is given by $\dot{m} = const\sqrt{\rho_{\mathrm{total}}}\sqrt{p_{\mathrm{total}}}$ where the constant *const* depends only on gas constant R and $\gamma = c_p/c_v$. For unsteady actuation, a time-dependent variation $\dot{m}(t) = \dot{m}_0 + \Delta\dot{m}\sin(\omega t)$ with mean value $\dot{m}_0$ and amplitude $\Delta\dot{m}$ is implemented by imposing

$$
\begin{aligned}
p_{\mathrm{total}}(t) &= p_{\mathrm{total,min}} + \Delta p_{\mathrm{total}}\sin(\omega t)^{\alpha_p} \\
\Delta p_{\mathrm{total}} &= \max\left[(p_{\mathrm{total,max}} - p_{\mathrm{total,min}})\ ,\ 0\right]\ , \quad \alpha_p = \frac{2\gamma}{1+\gamma} \\
\rho_{\mathrm{total}}(t) &= \rho_{\mathrm{total,max}}\left(\frac{p_{\mathrm{total}}(t)}{p_{\mathrm{total,max}}}\right)^{1/\gamma}
\end{aligned}
$$

This ensures that the isentropic relation $p(t_0)/p(t_1) = (\rho(t_0)/\rho(t_1))^\gamma$ is satisfied at each time t. Similarly, the time-variation of $\dot{m}$ for pulsed actuation (see 'Actuation type') is

achieved.
The jet Mach number from the isentropic area-Mach number relation $Ma_{\text{jet,isen}}$ ('Jet Mach number from isentropic area-Mach number relation') needs to be specified. $Ma_{\text{jet,isen}}$ is the solution of the area-Mach number relation, which returns the local Mach number Ma at any location of the duct with cross section area A, where A^* denotes the throat area where flow is sonic, i.e., $Ma^* = 1.0$,

$$\left(\frac{A}{A^*}\right)^2 = \frac{1}{Ma^2}\left[\frac{2}{\gamma+1}\left(1+\frac{\gamma-1}{2}Ma^2\right)\right]^{(\gamma+1)/(\gamma-1)}$$

Given $\frac{A}{A^*}$ from the nozzle geometry, Ma can be obtained from a table, e.g., in the textbook by Anderson [4].
Then the values for the conservative variables are prescribed based on the following values for the primitive variables (using $Ma \equiv Ma_{\text{jet,isen}}$ for abbreviation)

$$\begin{aligned}
\rho(t) &= \rho_{total}(t)\left(1+\frac{\gamma-1}{2}Ma^2\right)^{\frac{-1}{\gamma-1}} \\
\vec{u}(t) &= Ma\ a(t)\vec{\text{dir}}\ , \qquad\qquad a(t) = \sqrt{\gamma p(t)/\rho(t)} \\
p(t) &= \rho_{total}(t)\left(1+\frac{\gamma-1}{2}Ma^2\right)^{\frac{-\gamma}{\gamma-1}}
\end{aligned}$$

with vector $\vec{\text{dir}}$ describing the jet direction. HINT: It is strongly recommended to prescribe the conservative variables at the inlet, i.e., to deactivate ('Prescribe fluxes (0/1)').

`Actuation type:` character string, default (none)

- This parameter describes the time-dependent behavior of the jet. IMPORTANT Note: The most important parameter for the actuation boundary condition is 'Actuation subtype' which controls the values for the parameter 'Compressible jet (0/1)'. The latter parameter specifies (1) whether the jet is treated as incompressible or compressible flow, and (2) determines the parameters to specify the jet properties (i.e., jet density, velocity and temperature).
  Available options for 'Actuation type' are:
  - 'constant_blowing',
  - 'pulsed_blowing',
  - 'harmonic_blowing',
  - 'constant_suction',
  - 'pulsed_suction',
  - 'harmonic_suction',
  - 'pulsed_blowing_and_suction_net_zero_mass',

– 'harmonic_blowing_and_suction_net_zero_mass'

Short note on the implementation: Depending on whether or not the parameter 'Prescribe fluxes (0/1)' is activated, either the values for the conservative variables are set at the boundary nodes (Dirichlet type boundary condition) or the corresponding fluxes are prescribed using the Riemann solver (Neumann type boundary condition). It is strongly recommended to deactivate the parameter 'Prescribe fluxes (0/1)'.
In case of suction, a suitable outflow pressure is computed iteratively until the user specified massflow (determined by density and velocity) is obtained. For this purpose the parameter 'Simple suction (0/1)' should be de-activated unless choked nozzle effects (i.e., locally supersonic flow in the actuation slot) occur.
Some comments on unsteady actuation: Denote T_h the high flow rate time (in [s]), T_l the low flow rate time. Then the actuation period is $T_f = T_h + T_l$ and the duty cycle is defined as T_h/T_f. Denote $u_{\text{jet}} = u_{\text{jet}}(t)$ the jet velocity at time t which is constant across each boundary marker where actuation is used. Denote $u_{\text{jet,max}}$ the maximal jet velocity, which is positive for blowing and negative for suction (by definition here).

– Constant actuation: $u_{jet}(t) = u_{jet,max}$ for all t.
– Harmonic actuation: $u_{jet}(t) = u_{jet,max} \max\left(\sin\left(\frac{2\pi(t+t_{shift})}{T_f}\right); 0\right)$
– Pulsed actuation: $u_{jet}(t) = u_{jet,max} \times I_{[0,t_{duty}]}(t_{offset})$, where $I_{[t_a,t_b]}(t) = 1$ if $t \in [t_a, t_b]$ and zero else.

where $t_{offset} = t + t_{shift} - n * T_f$ with n such that $t_{offset} \in (0, T_f)$. The parameter t_{shift} is the 'Actuation initial time shift'.
For pulsed actuation, a smoothing method is used. We introduce t_{99} as the time over which u_{jet} is increased from zero to its maximum value. At $t = t_{99}$, $u_{\text{jet}} = 0.99 u_{\text{jet,max}}$. Smoothing is predominantly done in the time interval $(-t_{99}, t_{99})$ (to avoid a sharp jump from 0 to 1 at $t = 0$) and in $(T_h - t_{99}, T_h + t_{99})$ (to avoid a sharp jump from 1 to 0 at $t = T_h$), see figure 4. The duration for smoothing t_{99} is written as $t_{99} = c_{perc} t_{sm,ref}$ with $t_{sm,ref} = \frac{1}{2}\min(T_h, T_l)$ where $t_{sm,ref}$ is the smoothing reference time, and $c_{perc} \in (0, 1)$ is the percentage for smoothing being (predominantly) active with respect to $t_{sm,ref}$ which has to be specified by the user with the parameter 'Smoothing pulsed signal'.
WARNING: It is strongly recommended to model the geometry of the actuation slot. For the side walls, universal wall functions or an Euler wall can be prescribed. At the bottom wall of the slot, the actuation boundary condition should then be imposed. In case of actuation inflow (i.e., blowing) this is necessary to ensure an accurate massflow. Moreover this is the only method to ensure the jet direction properly. For suction this is the only possibility to ensure the correct outflow direction and also the correct massflow.
Regarding mesh generation, the Chimera method can be used. This allows for a simulation on overlapping grids consisting of a structured mesh for the actuation slot and a

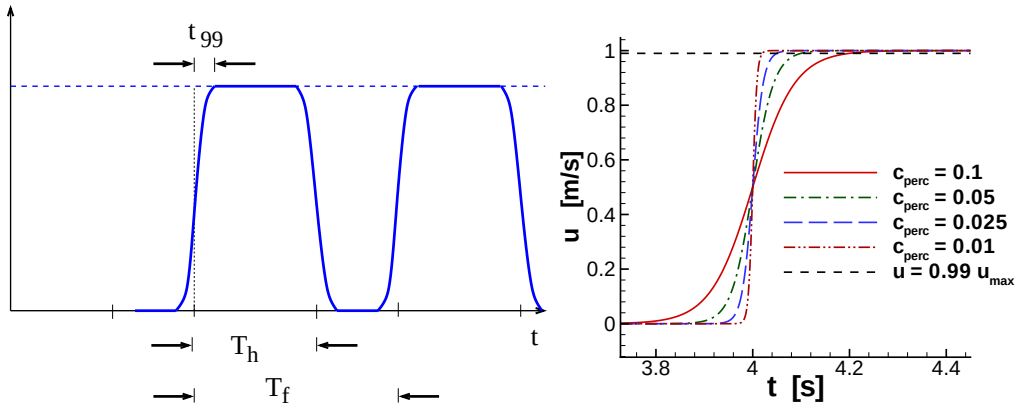


Figure 4: Left: Smoothing of jump signal. Right: Illustration of how smoothing can be controlled using the parameter 'Smoothing pulsed signal'. which is denoted by c_{perc} in the Figure. In the right figure, $T_f = 4$[s] and $T_h = 2$[s].

(hybrid) mesh for the airfoil or wing. Alternatively, in case of a single mesh approach without Chimera, it is possible to use universal wall functions in case of actuation boundary condition on a mesh with $y^+(1) = 10$ which makes mesh generation (e.g., using Centaur) much easier.

`Compressible jet (0/1):` integer, default 0

- This parameter specifies (1) whether the jet is treated as compressible or incompressible fluid flow, and (2) determines the related parameters to specify the jet properties (i.e., jet density, velocity and temperature). It is closely related to the parameter 'Actuation subtype'.
  If this parameter is activated, then the actuation jet is treated as compressible flow. In case of a compressible jet, user specified parameters are total pressure ('Jet total pressure max') $p_{\text{total}} \equiv p_{\text{total,max}}$ and total density $\rho_{\text{total}} \equiv \rho_{\text{total,max}}$ ('Jet total density max'). In case of unsteady actuation, $p_{\text{total,min}}$ ('Jet total pressure min') also needs to be specified.
  If this parameter is deactivated, then the actuation jet is treated as incompressible flow. In case of an incompressible jet, user specified parameters are 'Maximum jet velocity', 'Jet fluid density', and 'Jet fluid temperature'

`Consistent pressure treatment (0/1):` integer, default 0

- This parameters controls the treatment of pressure at an actuation boundary in case of inflow, i.e., blowing.
  According to the theory of characteristics for hyperbolic PDEs, four variables have to be specified at the boundary and the remaining variable has to be taken from the first inner field point. In other words, we prescribe the four variables of density and momentum and we extrapolate the pressure from the inner domain. This mode is used when choosing 'Consistent pressure treatment (0/1): 1'.
  For some actuation test cases the code was unstable when extrapolating the pressure. For this reason, a stable version was additionally implemented which uses the reference pressure instead of taking the field value of the first inner point. This mode is used when choosing 'Consistent pressure treatment (0/1): 0'.
  As a preliminary best practice guideline (from March 2010) we recommend to use a value '0' for the choice 'Actuation subtype': "standard_incompressible" and a value '1' else.

`Extrapolate turbulent field data at inflow (0/1):` integer, default 0

- Use extrapolated turbulent field data as inflow boundary condition for actuation.

`Jet cut-off factor:` float, default 0.05

- In case of unsteady suction, the method to compute the outflow pressure iteratively to obtain correct massflow might not converge in case of small outflow velocities. This

parameter therefore defines a minimum value for the outflow velocity

$$u_{jet,min} \;=\; cut \times u_{jet,max}$$

where $u_{jet,max}$ is the 'Maximum jet velocity' and *cut* is the 'Jet cut-off factor'.

`Jet diameter:` float, default -

- In case of inflow, the eddy viscosity at the inlet needs to be prescribed in a reasonable way. The recommended strategy is to activate the parameter 'Jet improved model for eddy viscosity (0/1)'. Then the prescription of the diameter of the jet at the inflow boundary as an additional parameter is also required. Input of the jet diameter has to be in grid units.

`Jet fluid density:` float, default -

- Density of jet fluid at actuation boundary. As default, the reference value is used.

`Jet fluid temperature:` float, default -

- Temperature of jet fluid at actuation boundary. As default, the reference value is used.

`Jet improved model for eddy viscosity (0/1):` integer, default 1

- This parameter provides a value for the turbulent viscosity at an actuation boundary in case of inflow which is in good agreement with experimental data. It is based on Chapter 5 in the textbook "Turbulent Flows" by Pope [11]. The following formula is used

$$\nu_{\mathrm{t}}(x,t) \;=\; 0.11\sqrt{k}D \;=\; 0.11\sqrt{\frac{3}{2}}\mathrm{Tu}\; U_{\mathrm{jet,max}}\frac{D}{2}$$

  with turbulence intensity Tu, jet diameter D ('Jet diameter') and 'Maximum jet velocity' $U_{\mathrm{jet,max}}$. In case of using a two-equation turbulence model, please also ensure a proper choice of the parameter ('Jet turbulent kinetic energy cut-off factor for turbulent viscosity').

`Jet Mach number from isentropic area-Mach number relation:` float, default -

- This parameter specifies the jet Mach number from the isentropic area-Mach number relation, see parameter 'Actuation subtype' for details.

`Jet ratio mue-t/mue-l:` float, default 200.0

- From the ratio of turbulent to laminar viscosity mue-t/mue-l of the jet, the eddy viscosity at actuation inlet boundary is determined.

`Jet skew angle beta:` float, default 0.0

- The jet skew angle beta describes (warning: only approximately!) the rotation of the local surface normal vector in spanwise direction after the rotation using the 'Jet yaw angle alpha' has been applied. It is defined as follows: The directional vector of unit magnitude of the jet $\vec{e}_{jet}$ at a surface area with normal vector $\vec{n}$ (pointing into the computational domain by definition here) is given by the linear combination

$$\vec{e}_{jet} = \cos(\beta) D(\alpha) \vec{n} + \sin(\beta) \vec{e}_y \quad \text{with} \quad D(\alpha) = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix}, \quad \vec{e}_y = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

  Therein, the matrix $D(\alpha)$ describes the rotation of the surface normal vector around the y-axis.
  Warning: The jet direction used in the code is given by the exact mathematical definition above. If the user prescribes jet yaw and skew angle, then he must check the direction! In the general case of a complex geometry, the jet direction cannot be described in a simple way using jet yaw and skew angle. Therefore it is recommended to prescribe the jet direction vector using the parameters 'User defined actuation direction (0/1)' and 'Actuation direction'

`Jet total density max:` float, default -

- For actuation using compressible jet ('Compressible jet (0/1)'), this parameter prescribes the total density at the jet inflow in case of steady actuation and the maximum total density at the jet inflow in case of time-dependent actuation. For time-dependent actuation, the time-dependent variation of the total density at the jet inflow is done according to the description see the parameter 'Actuation subtype'

`Jet total pressure max:` float, default -

- For actuation using compressible jet ('Compressible jet (0/1)'), this parameter prescribes the total pressure at the jet inflow in case of steady actuation and the maximum total pressure at the jet inflow in case of time-dependent actuation. See also 'Compressible jet (0/1)'.

`Jet total pressure min:` float, default -

- For time-dependent actuation using compressible jet ('Compressible jet (0/1)'), this parameter prescribes the minimum total pressure at the jet inflow in case of time-dependent actuation. In case of steady actuation, this parameter is not used. See also 'Compressible jet (0/1)'

`Jet turbulent intensity:` float, default 0.05

- Turbulence intensity of jet fluid at the actuation boundary.

`Jet turbulent kinetic energy cut-off factor for turbulent viscosity:` float, default 1.0e-3

- In case of actuation using a two-equation turbulence model, the value for turbulent kinetic energy at the inflow boundary is prescribed using the formula

$$k|_{\text{inflow}} = 1.5(\text{Tu}\max(u_{jet,cut-off}, u_{\text{jet}}(t)))^2$$

  with turbulence intensity Tu, $u_{\text{jet}}(t)$ being the jet velocity at the current time and $u_{jet,cut-off}$ being prescribed by the present parameter. The value has to be specified in $[m/s]$.

`Jet turbulent viscosity to jet velocity exponent:` float, default 1.0

- Control parameter for imposing time dependent eddy viscosity at an actuation inlet boundary. Only used if the parameter 'Prescribe time dependent eddy viscosity (0/1)' is activated and if turbulence quantities are not extrapolated from the first inner node to the boundary. The eddy viscosity μ_t at the actuation inlet boundary at time t is defined by $\mu_t(t) = \left(\frac{u_{jet}(t)}{u_{jet,max}}\right)^{\alpha} \mu_{t,0}$ where $u_{jet}(t)$ is the jet velocity at time t and $u_{jet,max}$ is the maximal jet velocity (amplitude). Therein $\mu_{t,0}$ is the eddy viscosity at the actuation boundary for $u_{jet,max}$ and α denotes the 'Jet turbulent viscosity to jet velocity exponent'.

`Jet yaw angle alpha:` float, default 0.0

- Parameter to specify the yaw angle of the direction of actuation. The jet yaw angle alpha describes the rotation of the local surface normal vector with the axis of rotation being the y-axis, i.e., the axis defined by the vector $(0, 1, 0)^T$. If the surface is plain in y-direction (i.e., no surface curvature in y-direction) and if the onflow direction is in the x-z-plane, then α is the angle with respect to the surface normal (pointing here into the computational domain by definition). See also the parameter 'Jet skew angle beta'.

`Maximum jet velocity:` float, default 0.0

- Jet velocity at this actuation boundary marker. At present only a markerwise constant jet velocity can be prescribed. For constant blowing, this parameter describes the constant jet velocity. For time-dependent blowing, this parameter describes the maximum jet velocity in one period. The value has to be specified in $[m/s]$.

`Monitor forces (0/1):` integer, default 1

- Turns on/off the consideration of all surfaces belonging to this boundary part when integrating the forces and moments for monitoring. On actuation boundaries the option is not yet complete and therefore set to `DEVELOPMENT`.

`Prescribe fluxes (0/1):` integer, default 0

- Compute fluxes with Riemann solver and prescribe as boundary condition instead of setting the conservative variables. This parameter should be used only for constant suction if the parameter 'Simple suction (0/1)' is activated in case of choked nozzle effects.

`Prescribe time dependent eddy viscosity (0/1):` integer, default 1

- Allows to relate the eddy viscosity of the jet at the actuation inlet boundary to the jet velocity for time-dependent actuation. For more details see the parameter 'Jet turbulent viscosity to jet velocity exponent'.

`Relaxation factor pressure correction:` float, default 0.001

- For actuation outflow boundary condition (suction) this is the relaxation parameter for iteratively computing outflow pressure until the user specified mass flow is obtained.

$$p_{new} = (1 - \alpha) \times p_{old} + \alpha \times \dot{m}_{act}/\dot{m}_{tar} \times p_{old}$$

  where p_{old} is the pressure from the previous pseudo-time iteration step, α is the relaxation factor, and $\dot{m}_{act}$, $\dot{m}_{targ}$ are the actual respectively the target massflow across the actuation boundary marker from the previous pseudo-time iteration step.
  IMPORTANT HINT: In case of suction during at least a part of the actuation cycle, this parameter has a large influence on the convergence rate. Albeit, stable TAU runs using the suction boundary condition require as a rule of thumb 500 inner iterations per time step when using a 3v-multigrid cycle.

`Restart use mean pressure (0/1):` integer, default 0

- For actuation outflow boundary condition (suction) this parameter allows to compute the mean pressure across the boundary marker from a restart solution and to use this as initial guess. This parameter should be used only for steady suction in case of restart.

`Simple suction (0/1):` integer, default 0

- Simple implementation of suction, where density and velocity are set according to the user specified way. The mathematical implementation then prescribes the fluxes at the outflow boundary using a Riemann solver. Inside the routine the outflow velocity is corrected in order to obtain the massflow calculated from density and velocity specified by the user. Note that in the parameter 'Maximum jet velocity' has to be positive.
  WARNING: This method should be used only for constant suction in case that the default method does not work.
  HINT: On the positive side, this formulation of the suction boundary condition allows for reasonable results also in the case of choked nozzle effects (i.e., locally supersonic flow inside the slot in suction). In such case, the default method does not work.

`Smoothing pulsed signal:` float, default 0.025

- This parameter controls how much the signal for pulsed actuation is smoothed. A value smaller than 1.0e-3 corresponds to small smoothing and a value larger than 0.1 corresponds to large smoothing. See 'Actuation type' for more details.

`User defined actuation direction (0/1):` integer, default 0

- Parameter to define whether or not the direction of flow actuation is prescribed by specifying the three components (in x-, y- and z-direction) of jet velocity. See also parameter 'Actuation direction'.

Actuator exhaust
`Monitor mass flow (0/1):` integer, default 0

- Turns on/off the monitoring of the mass flow over all surfaces belonging to this boundary part.

`Write surface data (0/1):` integer, default 0

- Turns on/off the output of specified data for of all surfaces belonging to this boundary.

Actuator inflow
`Input file:` character string, default (none)

- Actuator disk input file to calculate the distribution of sectional loads. The input file contains either a table for the direct prescription of the sectional load or a set of tables for the parametrization of the blade and the aerodynamic coefficient functions of lift and drag. The aerodynamic coefficient functions either may be specified as polynoms or as a discrete function of the radial position of the blade section and the effective angle of attack. The set of input tables can also be specified in the parameter within the block of the boundary mapping. Each set of input tables must be completely specified either in the parameter file or the external input file. A mixed declaration is not allowed. The name of the input file may be arbitrary with one exception. If the file name contains the string "PALE" a HOST output file in Tecplot ASCII format is expected for the direct prescription of the sectional load. The format of the different sets of input tables is given in the actuatordisk section.

`Monitor mass flow (0/1):` integer, default 0

- Turns on/off the monitoring of the mass flow over all surfaces belonging to this boundary part.

`Number of blades:` integer, default 0

- Number of blades of the related propeller or rotor of an actuator disk. The parameter is used to calculate the distribution of sectional loads. If set to zero the actuator disk runs without load.

`Pitch angle:` float, default 0

- Collective pitch angle of the related propeller of an actuator disk. The pitch angle is set at the relative radius r/R specified by the parameter <Pitch point> and describes the angle between the chord of the blade section and the propeller plane. The value is specified in degrees. The parameter is used to calculate the distribution of sectional loads with the 2D blade element theory.

`Pitch point:` float, default 0

- Relative radius r/R of the related propeller of an actuator disk at which the collective pitch is set. Typical values are 0.7 or 0.75. The parameter is used to calculate the distribution of sectional loads with the 2D blade element theory.

`Propeller rpm:` float, default 0

- Revolutions per minute of the related propeller of an actuator disk. The parameter is used to calculate the distribution of sectional loads with the 2D blade element theory.

`Write surface data (0/1):` integer, default 0

- Turns on/off the output of specified data for of all surfaces belonging to this boundary.

Axisymmetry axis

`Set gradients (0/1):` integer, default 0

- The gradients of all variables except the velocity are set to zero in the direction normal to the boundary. The gradients of the velocity components are split into normal and tangential parts. The gradient of the tangential velocity is set to zero in the normal direction. Whereas the gradient of the normal velocity is set to zero in the tangential direction, because the normal velocity has to be zero along the boundary.

`Write surface data (0/1):` integer, default 0

- Turns on/off the output of specified data for of all surfaces belonging to this boundary.

Dirichlet

`Angle alpha (degree):` float, default 0

- Angle of attack. Positive when the relative wind is coming from below the nose of the aircraft.

`Angle beta (degree)`: float, default 0

- Yaw angle. Positive when the relative wind is coming from the left of the nose of the aircraft.

`Density`: float, default reference state

- Dimensionalized density. **Attention: Non-dimensional coefficients are based on the general reference state. Defining the farfield state different from the reference state will yield different coefficient values, e.g.** C_L, C_D, C_p **and so on. Even round-off differences may have an impact!**

`Dirichlet data file`: character string, default (none)

- This parameter should contain the name of the NetCDF file containing the data values for a Dirichlet boundary patch.

`Mach number`: float, default reference state

- The parameter can be used to prescribe a state on the boundary independent of the parameter Reference Mach number. The parameter Velocity can be used instead as well. **Attention: Non-dimensional coefficients are based on the general reference state. Defining the farfield state different from the reference state will yield different coefficient values, e.g.** C_L, C_D, C_p **and so on. Even round-off differences may have an impact!**

`Monitor mass flow (0/1)`: integer, default 0

- Turns on/off the monitoring of the mass flow over all surfaces belonging to this boundary part.

`Set gradients (0/1)`: integer, default 1

- The gradients of all variables are set to zero in the direction normal to the boundary.

`Sideslip angle (degree)`: float, default 0

- Sideslip angle. Positive when relative wind comes from the right of the nose of the aircraft.

`Temperature`: float, default reference state

- Temperature at the boundary.

`Velocity`: float, default reference state

- Dimensionalized absolute value of the velocity. Not required but accepted instead of the Mach number. **Attention: Non-dimensional coefficients are based on the general reference state. Defining the farfield state different from the reference state will yield different coefficient values, e.g.** C_L, C_D, C_p **and so on. Even round-off differences may have an impact!**

`Write surface data (0/1):` integer, default 0

- Turns on/off the output of specified data for of all surfaces belonging to this boundary.

Engine exhaust
`Exhaust face mid point:` array-float, default {0.0, 0.0, 0.0}

- This parameter defines the coordinate of the mid point of the engine exhaust face and is used to compute the engine axis using this point and the face normal. This is needed to define swirl on the exhaust face. Input of the coordinates has to be in grid units

`Monitor mass flow (0/1):` integer, default 0

- Turns on/off the monitoring of the mass flow over all surfaces belonging to this boundary part.

`Monitor WAT (0/1):` integer, default 1

- Turns on/off the monitoring of WAT over all surfaces belonging to this boundary part. Mass flow and WAT may be defined as

$$\dot{m} = \iint \rho(U \cdot n)\, dA,$$

where the integral is over the engine face, n is the normal to the face and dA is a surface element, and

$$\Omega = \frac{\dot{m}\sqrt{\bar{T}_0}}{\bar{p}_0},$$

respectively, where $\bar{p}_0$ and $\bar{T}_0$ are the average total pressure and average total temperature *on the engine face*. The local values of those quantities may be written

$$\begin{aligned} p_0 &= p \cdot \left[1 + \frac{1}{2}(\gamma - 1)M^2\right]^{\frac{\gamma}{\gamma-1}}, \\ T_0 &= T \cdot \left[1 + \frac{1}{2}(\gamma - 1)M^2\right], \end{aligned}$$

where M is the local Mach number and γ is the ratio of specific heats. The dimensional units of the quantities are

$$\begin{aligned} \dot{m} &\rightarrow kg/s \\ \Omega &\rightarrow (kg\sqrt{K})/(kPa\, s) \end{aligned}$$

where the use of kilopascals in the units of Ω should be noted.

`Number of polygon points for swirl:` integer, default 0

- For the definition of swirl on the engine exhaust face a polygon for the swirl has to be defined. This parameter defines the number of polygon points

`Outflow condition type:` character string, default Basic

- Currently only one outflow boundary condition is implemented, as a result this parameter is useless.

`Pressure ratio:` float, default variable

- Ratio of total pressure at the exhaust and the static pressure in the reference state. The reference state should be identical to the farfield. The default is the isentropic ratio:

$$(1 + \frac{\gamma - 1}{2} \cdot Ma_{\infty}^2)^{\frac{\gamma}{\gamma - 1}}$$

`Ratio mue-t/mue-l:` float, default 0.1

- The turbulent viscosity (in case of turbulent computations) is set according to the defined ratio of turbulent and laminar viscosity (mue-t/mue-l).

`Swirl mid point distance of polygon point:` array-float, default 0.0

- For the definition of swirl on the engine exhaust face this list contains the distances of the polygon points to the `'Exhaust face mid point'`. The points have to be ordered with increasing distance

`Swirl ratio of polygon point:` array-float, default 0.0

- For the definition of swirl on the engine exhaust face this list the swirl ratios of the polygon points. The meaning of the 'Swirl ratio of polygon point' entries is that the length of the swirl vector component is given by this factor times the normal velocity vector component, which is computed by total pressure and temperature. The swirl component are added, thus they are introducing extra energy. Positive swirl ratios define right turning swirl(looking in the outflow direction). Negative values are for left turning swirl.

`Target massflow:` float, default 0.0

- When defining a value larger than zero the `Pressure ratio` is adjusted iteratively until the massflow over the exhaust face matches the given value. Units are (kg/s).

`Temperature ratio:` float, default 1

- Ratio of the total temperature at the exhaust and the total temperature in the reference state. The reference state should be identical to the farfield. The default is the isentropic ratio.

`Turbulent intensity:` float, default 0.001

- The turbulent kinetic energy (in case of using a 2-equation model) is computed according to the defined value.

`Write surface data (0/1):` integer, default 0

- Turns on/off the output of specified data for of all surfaces belonging to this boundary.

Engine inflow

`Area ratio eps_fan:` float, default 0

- When using the `Fixed_epsfan` inflow type: inverse ratio of the fan inflow area to the respective area of the undisturbed region. The mass flow through the engine is calculated from this value by strange and dubious means. If the default value is used the mass flow of the exhaust plane is taken. The boundary treatment needs the existence of a farfield boundary and a corresponding engine exhaust boundary if this parameter is not specified.

`Engine inflow direction:` array-float, default {0, 0, 0}

- If an inflow direction other than that prescribed by the average of the boundary face normal vectors is desired, then it can be set using this parameter.

`Extrapolation type simple/characteristic (0/1):` integer, default 1

- When using the `Fixed_massflow` inflow type: use boundary condition based on a characteristic treatment of the Euler equations (highly recommended), or a simplified "engineering" b.c. that may be more robust for nearly supersonic inflow speeds.

`Fixed massflow:` float, default -1

- When using the `Fixed_pressure` or `Fixed_massflow` inflow types: the dimensional mass flow through the inflow face which should be obtained - either by coupling for the pressure based b.c. or directly for the mass flow b.c. Units are (kg/s).

`Fixed WAT:` float, default -1

- When using the `Fixed_massflow` inflow type: the derived dimensional WAT value on the inflow boundary. Only one of `Fixed WAT` and `Fixed massflow` may be given.

`Fixed/initial pressure:` float, default Reference pressure

- When using the `Fixed_pressure` inflow type: the dimensional pressure to be fixed on the inflow for all iterations (if using no mass coupling), or initially (otherwise).

`Inflow condition type:` character string, default Fixed_pressure

- One of `Fixed_epsfan`, `Fixed_pressure` or `Fixed_massflow`. The type of inflow condition is characterized by the variable that is set on the boundary during the enforcement of the condition - not the variable that is specified by the user. For example it is possible for the user to specify mass flow using the pressure condition by means of an inner iteration. Depending on the condition chosen different parameters are available. Note that the `Fixed_epsfan` is depreciated, and may be removed in future versions.

`Massflow convergence residual:` float, default 1.0e-3

- When using the `Fixed_pressure` inflow type and a mass coupling type other than `None`: an attempt to improve the convergence of pressure iteration for the mass flow, this parameter only allows a pressure update once the mass flow residual has reached the value specified. This doesn't really work - the `Fixed_massflow` inflow type is recommended for specification of a mass flow.

`Measurement coordinates:` array-float, default {0, 0, 0}

- Coordinates where the field pressure should match the target pressure. Input of coordinates has to be in grid units. They should be near a grid boundary, because the field pressure compared is taken from the nearest point on a grid boundary

`Monitor mass flow (0/1):` integer, default 0

- Turns on/off the monitoring of the mass flow over all surfaces belonging to this boundary part.

`Monitor WAT (0/1):` integer, default 1

- Turns on/off the monitoring of WAT over all surfaces belonging to this boundary part. Mass flow and WAT may be defined as

$$\dot{m} = \iint \rho(U \cdot n)\, dA,$$

where the integral is over the engine face, n is the normal to the face and dA is a surface element, and

$$\Omega = \frac{\dot{m}\sqrt{\bar{T}_0}}{\bar{p}_0},$$

respectively, where $\bar{p}_0$ and $\bar{T}_0$ are the average total pressure and average total temperature *on the engine face*. The local values of those quantities may be written

$$\begin{aligned} p_0 &= p \cdot \left[1 + \frac{1}{2}(\gamma - 1)M^2\right]^{\frac{\gamma}{\gamma-1}}, \\ T_0 &= T \cdot \left[1 + \frac{1}{2}(\gamma - 1)M^2\right], \end{aligned}$$

where M is the local Mach number and γ is the ratio of specific heats. The dimensional units of the quantities are

$$\begin{aligned} \dot{m} &\rightarrow kg/s \\ \Omega &\rightarrow (kg\sqrt{K})/(kPa\,s) \end{aligned}$$

where the use of kilopascals in the units of Ω should be noted.

`Regulator (0/1):` integer, default 0

- The parameter `Regulator (0/1)` can be turned on (1) whenever the pressure coupling is used within the engine boundary conditions. It ensures that the pressure at the engine inflow is kept between the actual static pressure and the total pressure and prevents the inflow pressure of becoming an unrealistic high or low value.

`Relaxation factor:` float, default 0.1

- When using the `Fixed_pressure` inflow type: this factor is used for the mass flow coupling between engine inflow and outflow, or engine inflow and the given fixed mass flow. The mass flow is set with a pressure at the inflow calculated as

$$p_{fan} = (1-\alpha) * p_{fan} + \alpha * (p_{fan} + \Delta p)$$

where p_{fan} is the updated pressure at the inflow, α is the *Relaxation factor* and Δp is computed as

$$\Delta p = \frac{mass\ \ inflow}{mass\ \ outflow} - 1.$$

The result is a damped (non-linear) oscillator with level of damping controlled by α. Unfortunately the convergence is very sensitive to this parameter and some experimentation is usually necessary. For obtaining a given mass flow the inflow type `Fixed_massflow` is therefore recommended.

`Solve quadratic eqn for rho/p (0/1):` integer, default 0

- When using the `Fixed_massflow` inflow type: use boundary condition based on a solution of a quadratic equation for ρ (highly recommended) or p - the result will always be the same, but the stability may be affected.

`Type of mass coupling:` character string, default None

- Type of iteration used to obtain desired mass flow when using the `Fixed_pressure` inflow type; one of `None`, `Outflow_massflow` or `Fixed_massflow`. If `None` is chosen the pressure is fixed to the given value. If `Outflow_massflow` is chosen, there must exist an engine outflow boundary with the same engine number as this inflow. In this case the inflow pressure is iterated until parity of inflow and outflow mass flow is obtained. If `Fixed_massflow` is chosen the pressure is iterated until the mass flow specified by the parameter `Fixed massflow` is achieved.

`Write surface data (0/1):` integer, default 0

- Turns on/off the output of specified data for of all surfaces belonging to this boundary.

Euler wall

`Monitor farfield state (0/1):` integer, default 0

- Turns on/off the consideration of all surfaces belonging to this boundary part when computing the effective farfield state at the origin for monitoring.

`Monitor forces (0/1):` integer, default 1

- Turns on/off the consideration of all surfaces belonging to this boundary part when integrating the forces and moments for monitoring. On actuation boundaries the option is not yet complete and therefore set to `DEVELOPMENT`.

`Monitor mass flow (0/1):` integer, default 0

- Turns on/off the monitoring of the mass flow over all surfaces belonging to this boundary part.

`Transpiration velocity file:` character string, default (none)

- Name of a NetCDF file containing transpiration velocities. For each node of a Euler wall a transpiration velocity can be read in. A positive sign of the transpiration velocity is associated to outflow. It is assumed, that the transpiration velocity vector is normal to the wall. The transpiration velocities can be used for example in case of boundary layer coupling. The structure of the NetCDF file is the same, like other surface files. The file has to contain the dimension (keyword no_of_points, the global id's of the nodes (keyword global_id) and the transpiration velocity (keyword vtransp)

`Wall pressure correction (0/1):` integer, default 0

- Modifies the wall pressure to the value required to bring any wall-normal velocity component to zero. Particularly useful when using Euler walls as boundaries for the sideplanes of axisymmetric calculations.

`Write surface data (0/1):` integer, default 0

- Turns on/off the output of specified data for of all surfaces belonging to this boundary.

Exit-pressure outflow

`Exit pressure:` float, default reference state

- The static pressure used in for the boundary treatment.

`Match measured pressure (0/1):` integer, default 0

- When `Match measured pressure` is set to one, the value of `Exit pressure` is only used for restart, after that it's calculated. The solver will set the value of `Exit pressure` so that the field pressure at the `Measurement coordinates` will match the value of `Measured pressure` when the computation has converged.

`Matching adjustment factor [0,1]:` float, default 0.5

- If the iteration is switched on for adjusting the Exit pressure to a given value at a given coordinate, this factor under-relaxes the update in order to stabilize the iteration thus the range is between 0 and 1.

`Matching iteration period:` integer, default 150

- When `Match measured pressure` is switched on this parameter controls the period of updating the boundary conditions.

`Matching iteration start:` integer, default "5 times the given 'Matching iteration period"

- When `Match measured pressure` is switched on this parameter controls the the first iteration number for which the `Matching iteration period` is compared with.

`Measured pressure:` float, default reference state

- When `Match measured pressure` is set to one, the value of `Exit pressure` is only used for restart, after that it's calculated. The solver will set the value of `Exit pressure` so that the field pressure at the `Measurement coordinates` will match the value of `Measured pressure` when the computation has converged.

`Measurement coordinates:` array-float, default {0, 0, 0}

- Coordinates where the field pressure should match the target pressure. Input of coordinates has to be in grid units. They should be near a grid boundary, because the field pressure compared is taken from the nearest point on a grid boundary

`Monitor farfield state (0/1):` integer, default 0

- Turns on/off the consideration of all surfaces belonging to this boundary part when computing the effective farfield state at the origin for monitoring.

`Monitor impulse (0/1):` integer, default 0

- Turns on/off the monitoring of the impulse over all surfaces belonging to this boundary part.

`Monitor mass flow (0/1):` integer, default 0

- Turns on/off the monitoring of the mass flow over all surfaces belonging to this boundary part.

`Monitor pressure (0/1):` integer, default 0

- Turns on/off the monitoring of the pressure over all surfaces belonging to this boundary part.

Farfield

`Adjustment factor:` float, default 0.01

- This parameter controls the modification sensitivity of alpha in dependence of clift. 0 = no change of alpha, 1 = full change. In practical cases a value between 0.05 and 0.5 is recommended.

`Angle alpha (degree):` float, default 0

- Angle of attack. Positive when the relative wind is coming from below the nose of the aircraft.

`Angle beta (degree):` float, default 0

- Yaw angle. Positive when the relative wind is coming from the left of the nose of the aircraft.

`Chord length:` float, default 0

- The chord length is required for the vortex correction.

`Constant alpha/clift (0/1):` integer, default 0

- It's possible to define a 'Targeted clift'. You have to set the switch 'Constant alpha/clift (0/1)' = 1 in the farfield (bmap-file). Then the 'Angle alpha (degree)' is only the init-value. The calculation modifies the alpha until the agreement has been reached for 'Targeted clift'. But if there is a restart, the calculation starts with the init-value for 'Angle alpha (degree)'. In this case you have to copy the last result of 'Angle alpha (degree)' into the bmap-file. Note: the iteration is safe in the lower linear regime where the lift is a (nearly) linear function of the angle of attack. The algorithm can not detect the maximum lift if 'Targeted clift' is near or larger the Cl-max. It will pass alpha-max and thus the iteration will fail. For 'Targeted clift' values near alpha-max it is recommended to choose the related parameters appropriate: use a large 'Lift iteration period' and a small 'Adjustment factor', otherwise overshoots during the iteration can lead to passing alpha-max. However, these settings requires a lot of iteration steps.

`Density:` float, default reference state

- Dimensionalized density. **Attention: Non-dimensional coefficients are based on the general reference state. Defining the farfield state different from the reference state will yield different coefficient values, e.g.** C_L, C_D, C_p **and so on. Even round-off differences may have an impact!**

`Lift iteration period:` integer, default 20

- The value defines the period of iterations in which the angle of attack is iteratively modified in order to match the targeted clift. For targeted clift values in the (lower) linear regime a value between 5 and 10 for Euler-calculations and a value between 10 and 25 for laminar or turbulent flows are appropriate. The increase of this period makes the iteration more safe (less risk of passing alpha-max), but more costly.

`Lift iteration start:` integer, default 200

- The iteration to match the targeted clift is started if the current iteration number is larger or equal this start value. This parameter enables to converge a solution (partly) before starting the iterative procedure.

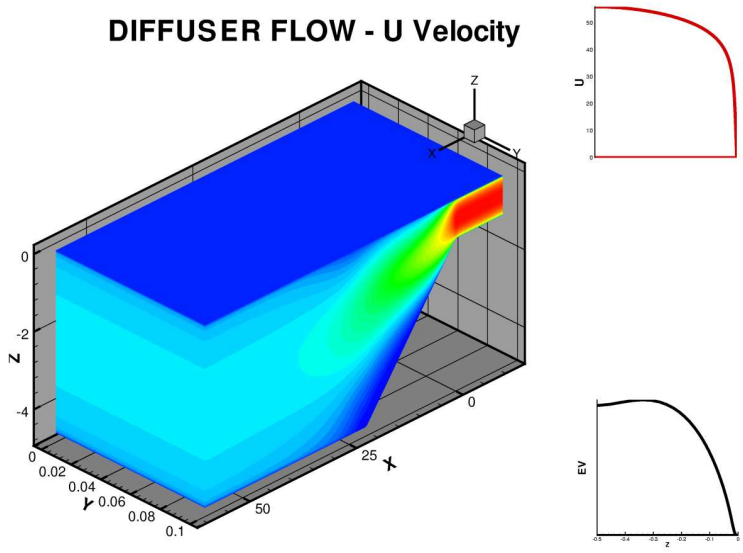
`Mach number:` float, default reference state

- The parameter can be used to prescribe a state on the boundary independent of the parameter Reference Mach number. The parameter Velocity can be used instead as well. **Attention: Non-dimensional coefficients are based on the general reference state. Defining the farfield state different from the reference state will yield different coefficient values, e.g. C_L, C_D, C_p and so on. Even round-off differences may have an impact!**

`Modify farfield file:` character string, default (none)

- This parameter allows one to specify the name of a file with which one can "patch" the farfield boundary conditions. In order to create the patching file, it is easiest to first dump a surface output file of the farfield boundary part. The user must then modify the contents of the surface data, either by using a NetCDF utility program or by writing a small code to read and modify the data points contained. In order to illustrate the salient points in the use of this parameter, an example is given in the following paragraph.
  Here a diffuser problem is considered. In order to obtain separation and reattachment at the end of the diffuser section it is essential to set the correct inflow boundary conditions. A channel flow is usually first computed. In order to achieve similarity within a channel, the domain must extend for much more than 100 channel heights if, for example, a top hat profile is specified on the inflow boundary plane. The flow development length can be shortened considerably if a developed channel profile is given at the infield. Thus a solution one can adopt is as follows. Firstly, a surface file of the channel inflow boundary is written. The initial top-hat profile is then replaced using experimental data and/or dimensional analysis. The new data is then written to a file identical to the original surface data file except for new values of velocity and other variables. The <modify farfield parameter> is set to the name of the new file, and the computation is allowed to proceeded to convergence. The profiles at the output boundary should then represented proper fully developed channel flow conditions. .
  We write the channel outflow plane to a surface output file and, using this file, we

can interpolate the data from the outflow channel plane to the diffuser inflow plane. The inflow boundary for the diffuser is now acceptable, and the diffuser calculation can proceed. This is illustrated in Fig. Modify farfield file(a).
It is important to recognize that the patched boundary data scales must be consistent with the scaling enforced through the reference variable settings. At the present time this feature only exists for the farfield boundary, however other boundary conditions may be modified to include this feature.



(a) Diffuser inflow boundary.

Figure 5: An example of using the modify farfield parameter.

`Monitor farfield state (0/1):` integer, default 0

- Turns on/off the consideration of all surfaces belonging to this boundary part when computing the effective farfield state at the origin for monitoring.

`Set gradients (0/1):` integer, default 0

- The gradients of all variables are set to zero in the direction normal to the boundary.

`Sideslip angle (degree):` float, default 0

- Sideslip angle. Positive when relative wind comes from the right of the nose of the aircraft.

`Targeted clift:` float, default 0

- If the 'Constant alpha/clift (0/1)' = 1 the calculation modifies the alpha until the agreement has been reached for 'Targeted clift'.

`Temperature:` float, default reference state

- Temperature at the boundary.

`Velocity:` float, default reference state

- Dimensionalized absolute value of the velocity. Not required but accepted instead of the Mach number. **Attention: Non-dimensional coefficients are based on the general reference state. Defining the farfield state different from the reference state will yield different coefficient values, e.g. C_L, C_D, C_p and so on. Even round-off differences may have an impact!**

`Vortex correction (0/1):` integer, default 0

- Turns on the vortex correction of the farfield values useful for 2d airfoil calculations. If it is turned on, chord length has to be > 0.

`Write surface data (0/1):` integer, default 0

- Turns on/off the output of specified data for of all surfaces belonging to this boundary.

Periodic plane

`Monitor mass flow (0/1):` integer, default 0

- Turns on/off the monitoring of the mass flow over all surfaces belonging to this boundary part.

`Write surface data (0/1):` integer, default 0

- Turns on/off the output of specified data for of all surfaces belonging to this boundary.

Reservoir-pressure inflow

`Inflow direction:` array-float, default {0, 0, 0}

- This parameter is only active, if `Redirect velocity (0/1)` is set to 1. By setting this parameter the user can control the inflow direction for this boundary type. The inflow direction for all boundary points is set with this vector. The vector is assumed to be pointing downstream. For example, for the standard coordinate system (positive x = downstream direction, positive y = spanwise direction, positive z = resulting direction from the right hand rule) with a plane parallel to the yz plane, setting this parameter to {1, 0, 0} results in an inflow direction normal to the inflow plane. If an inflow direction of 45 degrees from below is desired, this could be simulated by setting this parameter to {1, 0, 1} (the vector is normalized within the TAU-Code). If the default value is used, for each boundary point the flow is assumed to be parallel to the local surface normal.

`Monitor farfield state (0/1)`: integer, default 0

- Turns on/off the consideration of all surfaces belonging to this boundary part when computing the effective farfield state at the origin for monitoring.

`Monitor forces (0/1)`: integer, default 1

- Turns on/off the consideration of all surfaces belonging to this boundary part when integrating the forces and moments for monitoring. On actuation boundaries the option is not yet complete and therefore set to `DEVELOPMENT`.

`Monitor impulse (0/1)`: integer, default 0

- Turns on/off the monitoring of the impulse over all surfaces belonging to this boundary part.

`Monitor mass flow (0/1)`: integer, default 0

- Turns on/off the monitoring of the mass flow over all surfaces belonging to this boundary part.

`Monitor pressure (0/1)`: integer, default 0

- Turns on/off the monitoring of the pressure over all surfaces belonging to this boundary part.

`Redirect velocity (0/1)`: integer, default 1

- The reservoir condition is specified via `Total density` and `Total pressure`. An isentropic expansion from the reservoir condition to the velocity found in the iterated solution determines the outer state at the boundary. If this parameter is set to one, the parameter `Inflow direction` is used to redirect the velocity in order to ensure an inflow, independed of the iterated solution. In case of a non parallel inflow or respective an inflow not aligned to the local boundary normals, the redirection of the velocity can result in total pressures higher than the reservoir pressure. In such cases it is recommended to switch off the redirection of the velocity.

`Total density`: float, default reference state

- The total density on this boundary.

`Total pressure`: float, default reference state

- The total pressure on this boundary.

Supersonic inflow

`Angle alpha (degree):` float, default 0

- Angle of attack. Positive when the relative wind is coming from below the nose of the aircraft.

`Angle beta (degree):` float, default 0

- Yaw angle. Positive when the relative wind is coming from the left of the nose of the aircraft.

`Density:` float, default reference state

- Dimensionalized density. **Attention: Non-dimensional coefficients are based on the general reference state. Defining the farfield state different from the reference state will yield different coefficient values, e.g.** C_L**,** C_D**,** C_p **and so on. Even round-off differences may have an impact!**

`Mach number:` float, default reference state

- The parameter can be used to prescribe a state on the boundary independent of the parameter Reference Mach number. The parameter Velocity can be used instead as well. **Attention: Non-dimensional coefficients are based on the general reference state. Defining the farfield state different from the reference state will yield different coefficient values, e.g.** C_L**,** C_D**,** C_p **and so on. Even round-off differences may have an impact!**

`Monitor mass flow (0/1):` integer, default 0

- Turns on/off the monitoring of the mass flow over all surfaces belonging to this boundary part.

`Set gradients (0/1):` integer, default 1

- The gradients of all variables are set to zero in the direction normal to the boundary.

`Sideslip angle (degree):` float, default 0

- Sideslip angle. Positive when relative wind comes from the right of the nose of the aircraft.

`Temperature:` float, default reference state

- Temperature at the boundary.

`Velocity:` float, default reference state

- Dimensionalized absolute value of the velocity. Not required but accepted instead of the Mach number. **Attention: Non-dimensional coefficients are based on the general reference state. Defining the farfield state different from the reference state will yield different coefficient values, e.g. C_L, C_D, C_p and so on. Even round-off differences may have an impact!**

`Write surface data (0/1):` integer, default 0

- Turns on/off the output of specified data for of all surfaces belonging to this boundary.

Supersonic outflow
`Monitor mass flow (0/1):` integer, default 0

- Turns on/off the monitoring of the mass flow over all surfaces belonging to this boundary part.

`Set gradients (0/1):` integer, default 1

- The gradients of all variables are set to zero in the direction normal to the boundary.

`Write surface data (0/1):` integer, default 0

- Turns on/off the output of specified data for of all surfaces belonging to this boundary.

Viscous wall
`Coolant reservoir temperature:` float, default reference state

- The parameter is used only if `Effusion mass flux` is positive and `Heat flux` is set to 'radiative_equilibrium'. In this case the difference in the coolant's internal energy according to the temperature difference (wall temperature versus coolant reservoir temperature) is taken into account in the energy balance.

`Effusion mass flux:` float, default -1.0

- Only positive values are accepted and regarded as inflow through a permeable wall. The effusion mass flux is used for the mass balance and the momentum is set according to the mass flux divided by the local density. The influence on the energy equation depends on the chosen `Heat flux` model.

`Emissivity:` float, default 0.8

- Material constant to describe the emission compared to the radiation of a black surface. The emissivity is needed to calculate the heat flux according to the Stefan-Boltzmann law. This heat flux is taken into account if radiative equilibrium is assumed.

`Heat flux:` character string, default adiabatic

- Switch between heat flux treatments: 'adiabatic', 'isothermal' and 'radiative_equilibrium'. In case of 'adiabatic' the energy is balanced without heat transfer at the wall. In case of 'isothermal' the energy is set according to the specified temperature. In the last case the energy is balanced assuming that the heat flux in the fluid is equal to the energy emitted via radiation (see `Emissivity`) plus the energy gained by a coolant, if an `Effusion mass flux` is specified.

`Heat flux correction (0/1):` integer, default 0

- Correction of the heatflux evaluation via an energy_balance for a not converged calculation.

`Heat flux evaluation:` character string, default gradient

- Switch between heat flux output treatments: 'gradient' and 'energy_balance'. This parameter has no influence on the iterated solution, but changes the interpretation of the (same) results. In case of 'gradient' the wall heat flux is calculated with the Fourier law of heat conduction: $\vec{q} = -\lambda \nabla T$, where λ is the the thermal conductivity of the fluid and ∇T the gradient of the temperature at the wall. An improved discretization of the Fourier heat flux is implemented, but upto now only validated for a small number of cases. Therefore, the former discretization is still available as 'old_gradient', but may be removed in future releases. In case of 'energy_balance' the wall heat flux is calculated directly from the energy equation balanced at the wall. This evaluation is not usable together with wall functions or a radiative_equilibrium heat flux.

`Monitor farfield state (0/1):` integer, default 0

- Turns on/off the consideration of all surfaces belonging to this boundary part when computing the effective farfield state at the origin for monitoring.

`Monitor forces (0/1):` integer, default 1

- Turns on/off the consideration of all surfaces belonging to this boundary part when integrating the forces and moments for monitoring. On actuation boundaries the option is not yet complete and therefore set to `DEVELOPMENT`.

`Monitor heat flow (0/1):` integer, default 0

- Turns on/off the consideration of all surfaces belonging to this boundary part when integrating the heat flux for monitoring.

`Monitor mass flow (0/1):` integer, default 0

- Turns on/off the monitoring of the mass flow over all surfaces belonging to this boundary part.

`Moving wall (0/1):` integer, default 0

- Switch on (1) the moving wall option

`Moving wall omega:` array-float, default {(0, 0, 0)}

- Omega of rotation of this boundary around a given origin [deg/s].

`Moving wall origin:` array-float, default {(0, 0, 0)}

- Origin of the rotation of this boundary [grid units].

`Moving wall trans:` array-float, default {(0, 0, 0)}

- Prescribe the translation velocity vector for a moving wall [grid units/s].

`Set gradients (0/1):` integer, default 0

- The pressure gradient normal to the wall is set to zero. If the wall is treated as adiabatic, the density gradient normal to the wall is set to zero as well.

`Subtype:` character string, default (none)

- Available options are 'laminar', 'transition' or 'turbulent'.

`Target yplus:` float, default 1

- This parameter is used for y^+ calculation over selected surfaces.

`Temperature:` float, default reference state

- Temperature at the boundary.

`Temperature filename:` character string, default (none)

- In case of an isothermal heat flux the local wall temperature can be read in once for every run from the specified file. The format of the file is the same as a surface output file with necessary variables 'temp' and 'gidx'.

`Use wall function (0/1):` integer, default 0

- Switch between low-Re and hybrid-Re treatment of turbulent viscous walls.
  - value = 0:
    Default implementation. Low-Reynolds boundary condition, which requires low-Re grid, i.e. $y^+(1) = 1$ for the first node above the wall. The no-slip condition for momentum is prescribed.

– value = 1:
Wall-function boundary condition. Hybrid-Reynolds boundary condition, i.e., the restriction concerning the location of the first off-wall grid node $y^+(1)$ is eliminated. The first off-wall node can reside anywhere in the logarithmic part of the boundary-layer or below, i.e., $0 \leq y^+(1) \leq y^+_{max}$, where y^+_{max} depends on the Reynolds number and should be at most 0.05 times the boundary layer thickness to ensure at least three grid nodes in the log-layer. This boundary condition prescribes the wall-parallel component of the wall-shear stress and the no-penetration condition for the wall-normal velocity.
Best practice: For highlift configurations, a good compromise between accuracy and speedup is obtain if $y^+(1) = 10$. For transonic cruise flight conditions, $y^+(1) = 50$ can be used if focus is on fast solutions. For flows with strong separation the standard low-Re boundary condition is recommended. Moreover the parameter Omega boundary condition type (0/1/2) should be set to 2.

* WARNING: Values of $y^+(1)$>80 may lead to a significant modeling error.

`Write surface data (0/1):` integer, default 0

- Turns on/off the output of specified data for of all surfaces belonging to this boundary.

7.3.4 Output files

Output options are provided for three different classes of data at the present time: field data, surface data and cut plane data. Each of these is controlled through a number of parameters which are described below in Table 1 and Table 2. It should be noted that the term *primary data* pertains to the primitive variable and additional primitive variable data, whereas the term *derived data* describes data calculated as a function of the primary data, for example y^+ or the vorticity ω.

Table 1: Available output variable keywords and their function for the standard flow solver. Some of these quantities are not available for all solvers/computations, e.g. "eddy" is only available when performing turbulent calculations. Primitive variables and eddy viscosity (for turbulent calculations only) are written to field output in any case, without a definition of the related keyword (marked by "o" instead of "x"). Note that in table "D" denotes type Double, "I" denotes type Integer, N denotes "Newton", kg denotes "kilogram", m denotes "meter", K denotes "Degree Kelvin", W denotes "Watt", J denotes "Joule", ° denotes "degree" and s denotes "second". Despite the fact that $Watts = Joules/second$, the dimensioning adopted here is the most common form found in the literature. Also note that "F" denotes field output, "S" denotes surface output, and "C" denotes cut-plane output. Thus a keyword with an 'x' in the "F", "S" and "C" columns can be output as field, surface and cut-plane data. A unity symbol in the "Units" column will denote a dimensionless variable. In all cases a dimensioned variable is returned as a non-dimensional value if the parameter "Write pointdata dimensionless (0/1)" has been set to 1.

Keyword	Type	Value	Units	F	S	C
"acd-alpha"	D	effective angle of attack	° [deg]		x	
"acd-alphai"	D	induced angle of attack	° [deg]		x	
"acd-bld-secload"	D	system blade secload	N/m		x	
"acd-bld-vel"	D	system blade velocity	m/s		x	
"acd-cd"	D	coef of drag	1		x	
"acd-chlen"	D	chord length	m		x	
"acd-cl"	D	coef of lift	1		x	
"acd-dsk-xyz"	D	system disk coords	grid units		x	
"acd-raysc"	D	ray scaling factor	1		x	
"acd-secload"	D	sectional load	N/m		x	
"acd-twist"	D	blade twist	° [deg]		x	
"acd-vrot"	D	rotational velocity	m/s		x	
"apsimP"	D	apsim pressure	N/m^2	x	x	x
"blank"	D	chimera hole points	1	x	x	
"bldatacf"	D	boundary layer data cross flow	m		x	
"bldatasw"	D	boundary layer data streamwise	m		x	
"bldelta"	D	boundary layer data thicknesses	m		x	
"blinfo"	I, D	boundary layer information	1, ° [deg]		x	
"cf"	D	abs. skin friction value	1		x	x

continued on next page

Keyword	Type	Value	Units	F	S	C
“cfsgnx”	D	cf * sgn(cfx)	1			x
“cfxyz”	D	skin friction vector	1		x	x
“ch”	D	Stanton number	1		x	
“clydl”	D	local force coeff. vector	1		x	
“cmxyzl”	D	local moment coeff. vector	1		x	
“cp”	D	pressure coefficient	1	x	x	x
“cpharm”	D	time-average and 1st harmonic of cp	N/m^2		x	
“des”	D	length +(des),-(Rans)	%	x		
“divergence”	D	divergence of velocity field	$1/s$	x		
“domain”	D	domain number	1	x		
“eddy”	D	eddy viscosity	Ns/m^2	o	x	
“entropy”	D	normalized entropy difference	1	x	x	x
“fxyz”	D	force vector on wall surfaces	N		x	
“gid”	I	global point number	1	x	o	
“gradk”	D	k gradient	m/s^2	x		
“gradp”	D	pressure gradient	$kg/(m^2\ s^2)$	x		
“gradrho”	D	gradient of density	kg/m^4	x		
“gradu”	D	x-velocity gradient	$1/s$	x		
“gradv”	D	y-velocity gradient	$1/s$	x		
“gradw”	D	z-velocity gradient	$1/s$	x		
“gsid”	I	global surface id	1		x	
“H”	D	specific total enthalpy	m^2/s^2	x	x	
“heatfl”	D	wall heat flux	W/m^2		x	
“heli”	D	helicity	m/s^2	x		
“Hn”	D	normalized helicity	1	x		
“htc”	D	heat transfer coefficient	$kg/(s^3\ K)$		x	
“imach”	D	isentropic Mach number	1	x	x	
“ipzone”	D	chimera interpolated points	1	x	x	
“kappa”	D	thermal conductivity	$J/(s\ m\ K)$	x		
“krplus”	D	non-dim. surface roughness in wall units	1		x	

continued on next page

Keyword	Type	Value	Units	F	S	C
"krsm"	D	spec. kin. turb. energy for RSM	m^2/s^2	x	x	
"k1"	D	turb. kinetic energy	m^2/s^2	o	x	
"k2"	D	turb. dissipation	$1/s$	o	x	
"linear-uv"	D	Boussinesq approx. Reynolds Stress	m^2/s^2	x		
"ltflag"	D	laminar turbulent flag	1		x	
"l2"	D	lambda 2 criterion	$1/s$	x		
"mach"	D	Mach number	1	x	x	
"macht"	D	turbulent Mach number	1	x		
"massflux"	D	mass flux	$kg/(m^2\ s)$		x	
"mean"	D	$< v_i v_j >, < v_i p >$	$m^2/s^2, N/sm$	x	x	
"mean-cf"	D	field mean cf values	1	x	x	
"mean-cp"	D	field mean cp values	1	x	x	
"meanvelgrad"	D	mean velocity gradient	$1/s$	x	x	
"momentum"	D	momentum vector	$kg/(m^2\ s)$	x	x	
"muetmue"	D	turbulent/laminar viscosity	1	x		
"Nk"	D	kinematic vorticity number	1	x		
"notfound"	D	chimera points not found	1	x		
"oldvolume"	D	control volume (timestep N-1)	(grid units)^3	x		
"ooldvolume"	D	control volume (timestep N-2)	(grid units)^3	x		
"p"	D	pressure	N/m^2	o	x	x
"pl"	D	pressure loss	1			x
"Ptot"	D	total pressure	N/m^2	x	x	x
"Q"	D	second invariant Q	$1/s^2$	x		
"restart"	D	all variables used to continue calculation	SI-units	o	x	
"residual-stress"	D	residual turbulent stresses	m^2/s^2	x		
"rho"	D	density	kg/m^3	o	x	x
"rhoE"	D	density times total energy	$kg/(m\ s^2)$	x	x	
"rotcorr"	D	vortical flow correction	1	x		
"rotxyz"	D	rotation components	1/s	x		
"r11"	D	Reynolds stress component $R_{11} = \widetilde{u''u''}$	m^2/s^2	o	x	

continued on next page

Keyword	Type	Value	Units	F	S	C
"r22"	D	Reynolds stress component $R_{22} = \widetilde{v''v''}$	m^2/s^2	o	x	
"r33"	D	Reynolds stress component $R_{33} = \widetilde{w''w''}$	m^2/s^2	o	x	
"r12"	D	Reynolds stress component $R_{12} = \widetilde{u''v''}$	m^2/s^2	o	x	
"r13"	D	Reynolds stress component $R_{13} = \widetilde{u''w''}$	m^2/s^2	o	x	
"r23"	D	Reynolds stress component $R_{23} = \widetilde{v''w''}$	m^2/s^2	o	x	
"RE"	D	residual of energy	1	x		
"Rnue"	D	residual of SA viscosity (1EQ)	1	x		
"Rrho"	D	residual of density	1	x		
"Rv"	D	residual of velocity vector	1	x		
"Rk"	D	residual of turbulent kinetic energy	1	x		
"Rw"	D	residual of turbulent dissipation	1	x		
"Rwrsm"	D	residual of omega (RSM)	1	x		
"Rr11"	D	residual of Reynolds stress component	1	x		
"Rr22"	D	residual of Reynolds stress component	1	x		
"Rr33"	D	residual of Reynolds stress component	1	x		
"Rr12"	D	residual of Reynolds stress component	1	x		
"Rr13"	D	residual of Reynolds stress component	1	x		
"Rr23"	D	residual of Reynolds stress component	1	x		
"sa"	D	Spalart-Allmaras viscosity	Ns/m^2	o	x	
"sas"	D	Von Karman length	m	x		
"sst"	D	sst parameters F1, F2	1	x		
"temp"	D	temperature	K	x	x	x
"testgrad"	D	test gradients	$1/m$	x		
"tu"	D	turbulence intensity	1	x		
"variance"	D	$< u_i^{'} u_j^{'} >, < p^{'} p^{'} >$	SI units	x		
"v"	D	velocity vector	m/s	o	x	x
"velgrad"	D	gradients of velocity	$1/s$	x		
"visc"	D	laminar viscosity	Ns/m^2	x	x	
"volume"	D	control volume	(grid units)^3	x	x	
"vort"	D	vorticity	$1/s$	x		

continued on next page

Keyword	Type	Value	Units	F	S	C
“vxyzaero”	D	aerodynamic velocities	m/s	x	x	
“vxyzbody”	D	body-fixed velocities	m/s	x	x	
“vxyzedge”	D	boundary layer edge velocities	m/s		x	
“vxyzgeod”	D	geodesic velocities	m/s	x	x	
“vxyzmoving”	D	velocity in moving frame	m/s	x	x	
“wdist”	D	distance to nearest viscous wall	grid units	x		
“wrough”	D	surface roughness	grid units	x	x	
“wrsm”	D	turb. length scale omega (RSM)	$1/s$	o	x	
“wxyz”	D	grid velocity vector	grid units	x	x	
“xyz”	D	coordinate in grid units	grid units	x	x	x
“xyzaero”	D	aerodynamic coordinates	grid units	x	x	
“xyzbody”	D	body-fixed coordinates	grid units	x	x	
“xyzgeod”	D	geodesic coordinates	grid units	x	x	
“xyzold”	D	grid coordinates (timestep N-1)	grid units	x		
“xyzoold”	D	grid coordinates (timestep N-2)	grid units	x		
“yplus”	D	y+ above viscous walls	1		x	x

Table 2: Additionally available output variable keywords and their function if the input parameter “Solver type” is set to “DAdjoint” or “CAdjoint”. This input parameter marks the action the solver should perform. The solver type “DAdjoint” requires a restart-data file, and solves the discrete adjoint equations (a linear problem) corresponding to the given spatial discretization. The solver type “CAdjoint” solves the inviscid continuous adjoint equations about a flow solution given in restart-data. Note that in table “D” denotes type Double and “I” denotes type Integer. Also note that “F” denotes field output, “S” denotes surface output, and “C” denotes cut-plane output. Thus a keyword with an ‘x’ in the “F”, “S” and “C” columns can be output as field, surface and cut-plane data. A unity symbol in the ”Units” column will denote a dimensionless variable.

Keyword	Type	Value	Units	F	S	C
“gsid”	I	global surface id	1		x	
“psi1”	D	adjoint variable 1 (density)	1			x
“psi2”	D	adjoint variable 2 (u)	1			x
“psi3”	D	adjoint variable 3 (v)	1			x
“psi4”	D	adjoint variable 4 (w)	1			x
“psi5”	D	adjoint variable 5 (pressure)	1			x
“sens”	D	sensitivity of the cost function	1		x	

Table 3: Available monitoring variable keywords and their explanation.

Keyword	Comment
"Residual"	norm of rho-increments
"dvx/dt"	norm of u-increments
"dvy/dt"	norm of v-increments
"dvz/dt"	norm of w-increments
"dnue/dt"	norm of $\widetilde{\nu}$-increments (Spalart-Allmaras models)
"drhoE/dt"	norm of energy-increments
"drk/dt"	norm of k-increments (k-ω models)
"drk2/dt"	norm of ω-increments (k-ω models)
"dr11/dt"	norm of R_{11}-increments (Reynolds stress models)
"dr22/dt"	norm of R_{22}-increments (Reynolds stress models)
"dr33/dt"	norm of R_{33}-increments (Reynolds stress models)
"dr12/dt"	norm of R_{12}-increments (Reynolds stress models)
"dr13/dt"	norm of R_{13}-increments (Reynolds stress models)
"dr23/dt"	norm of R_{23}-increments (Reynolds stress models)
"dw-rsm/dt"	norm of ω-increments (Reynolds stress models)
"Max-res"	maxima of rho-increments
"X-max-res"	grid x-coordinate of Max-res
"Y-max-res"	grid y-coordinate of Max-res
"Z-max-res"	grid z-coordinate of Max-res
"C-lift"	lift-coefficient
"C-lift-p"	pressure part of C-lift
"C-lift-v"	viscous part of C-lift
"C-drag"	drag-coefficient
"C-drag-p"	pressure part of C-drag
"C-drag-v"	viscous part of C-drag
"C-sidef"	coeff of sideforce
"C-sidef-p"	pressure part of C-sidef
"C-sidef-v"	viscous part of C-sidef
"C-mx"	rolling moment coefficient

continued on next page

Keyword	Comment
“C-mx-p”	pressure part of rolling moment coefficient
“C-mx-v”	viscous part of rolling moment coefficient
“C-my”	pitching moment coefficient
“C-my-p”	pressure part of pitching moment coefficient
“C-my-v”	viscous part of pitching moment coefficient
“C-mz”	yawing moment coefficient
“C-mz-p”	pressure part of yawing moment coefficient
“C-mz-v”	viscous part of yawing moment coefficient
“C-fx”	force-coefficient (x-component)
“C-fx-p”	pressure part of force-coefficient (x-component)
“C-fx-v”	viscous part of force-coefficient (x-component)
“C-fy”	force-coefficient (y-component)
“C-fy-p”	pressure part of force-coefficient (y-component)
“C-fy-v”	viscous part of force-coefficient (y-component)
“C-fz”	force-coefficient (z-component)
“C-fz-p”	pressure part of force-coefficient (z-component)
“C-fz-v”	viscous part of force-coefficient (z-component)
“Fx”	x-component of force [N]
“Fx-p”	pressure part of x-component of force [N]
“Fx-v”	viscous part of x-component of force [N]
“Fy”	y-component of force [N]
“Fy-p”	pressure part of y-component of force [N]
“Fy-v”	viscous part of y-component of force [N]
“Fz”	z-component of force [N]
“Fz-p”	pressure part of z-component of force [N]
“Fz-v”	viscous part of z-component of force [N]
“Mx”	x-component of moment [Nm]
“Mx-p”	pressure part of x-component of moment [Nm]
“Mx-v”	viscous part of x-component of moment [Nm]
“My”	y-component of moment [Nm]

continued on next page

Keyword	Comment
“My-p”	pressure part of y-component of moment [Nm]
“My-v”	viscous part of y-component of moment [Nm]
“Mz”	z-component of moment [Nm]
“Mz-p”	pressure part of z-component of moment [Nm]
“Mz-v”	viscous part of z-component of moment [Nm]
“Min-mach”	minimum Machnumber in field
“Max-mach”	maximum Machnumber in field
“Heatflow”	total heatflow [W]
“Res-heat”	heatflux-increment
“Min-kr”	minimum value of kr above turbulent wall points
“X-min-kr”	grid x-coordinate of Min-kr
“Y-min-kr”	grid y-coordinate of Min-kr
“Z-min-kr”	grid z-coordinate of Min-kr
“Max-kr”	maximum value of kr above turbulent wall points
“X-max-kr”	grid x-coordinate of Max-kr
“Y-max-kr”	grid y-coordinate of Max-kr
“Z-max-kr”	grid z-coordinate of Max-kr
“Min-kr+”	minimum value of kr+ above turbulent wall points
“X-min-kr+”	grid x-coordinate of Min-kr+
“Y-min-kr+”	grid y-coordinate of Min-kr+
“Z-min-kr+”	grid z-coordinate of Min-kr+
“Max-kr+”	maximum value of kr+ above turbulent wall points
“X-max-kr+”	grid x-coordinate of Max-kr+
“Y-max-kr+”	grid y-coordinate of Max-kr+
“Z-max-kr+”	grid z-coordinate of Max-kr+
“Min-y+”	minimum value of y+ above turbulent wall points
“X-min-y+”	grid x-coordinate of Min-y+
“Y-min-y+”	grid y-coordinate of Min-y+
“Z-min-y+”	grid z-coordinate of Min-y+
“Max-y+”	maximum value of y+ above turbulent wall points

continued on next page

Keyword	Comment
“X-max-y+”	grid x-coordinate of Max-y+
“Y-max-y+”	grid y-coordinate of Max-y+
“Z-max-y+”	grid z-coordinate of Max-y+
“act-area”	Area actuation bdry
“act-targ-mf”	Actuation target massflow
“act-pres”	Actuation bdry pressure
“Min-eddyv”	min dim-less eddy viscosity
“Max-eddyv”	max dim-less eddy viscosity
“Min-k”	dim-less min kinetic energy
“Max-k”	dim-less max kinetic energy
“Min-k2”	dim-less min 2nd turb. quantity
“Max-k2”	dim-less max 2nd turb. quantity
“Max-mtm”	max eddy-viscosity/laminar-viscosity
“Angle-a”	angle of attack alpha [deg]
“Angle-b”	yaw angle beta [deg]
“Sideslip”	sideslip angle beta [deg]
“Farf-vx”	farfield vx [m/s]
“Farf-vz”	farfield vz [m/s]
“Mtm-2t”	number of pseudo time steps (unsteady)
“Alpha(t)”	pitch amplitude [deg]
“H(t)”	heave amplitude [grid unit]
“dalpha-dt”	pitch angular velocity [deg/s]
“dh-dt”	heave velocity [grid unit/s]
“S-k-e”	structural kinetic energy
“T/tperiod”	time normalized with period length
“Phi”	rotation angle phi [deg]
“Psi”	rotation angle psi [deg]
“Xi”	rotation angle xi [deg]
“Trans-x”	translation in x direction [grid unit]
“Trans-y”	translation in y direction [grid unit]

continued on next page

Keyword	Comment
“Trans-z”	translation in z direction [grid unit]
“Farf-mach”	effective farfield Mach number at origin
“Farf-alpha”	effective farfield angle of attack alpha at origin [deg]
“Farf-beta”	effective farfield angle of attack beta at origin [deg]
“les-dt”	timescale resolution of des [s]
“les-percent”	Percentage of des points
“hybcentpercent”	Percentage of fluxes computed with pure central
“vort-center-x”	x component of center of vorticity
“vort-center-y”	y component of center of vorticity
“vort-center-z”	z component of center of vorticity
“vort-total”	total vorticity
“dudx”	volume averaged velocity gradient
“dvdy”	volume averaged velocity gradient
“dwdz”	volume averaged velocity gradient
“total-ke”	volume averaged total/turbulent kinetic energy
“mean-dissp1”	volume averaged viscous dissipation
“mean-dissp2”	simplified volume averaged viscous dissipation
“mean-dissp3”	full isotropic form of volume av. viscous dissipation
“rmsu”	root mean square of instantaneous u vel. component
“rmsv”	root mean square of instantaneous v vel. component
“rmsw”	root mean square of instantaneous w vel. component
“rmsp”	root mean square of instantaneous pressure component
“skew”	Velocity derivative skewness statistic
“L2n-vc”	L2 norm of vortical correction
“Max-vc”	Max value of vortical correction
“Min-vc”	Min value of vortical correction
“area-z”	Area z
“Res-lift”	cl-increment
“Res-drag”	cd-increment
“R-time”	Total CPU time (real)

continued on next page

Keyword	Comment
“U-time”	Total CPU time (user)
“Rhs-time”	CPU time for one compute-RHS
“Norm-time”	Normalized CPU time (real)
“dJ/da-1”	Adjoint and Primal: Partial dJ/da
“dJ/da-2”	Adjoint: psi * partial dR/da, Primal: Partial dJ/dW * dW/da
“dJ/da”	Adjoint and Primal: derivative of cost fn wrt design var.
“Sens”	Functional sensitivity
“Freqdom-Cl-Amp”	Freqdom: Amplitude of CL
“Freqdom-Cl-Phase”	Freqdom: Phase shift of CL
“Freqdom-Cd-Amp”	Freqdom: Amplitude of CD
“Freqdom-Cd-Phase”	Freqdom: Phase shift of CD
“Freqdom-Cmx-Amp”	Freqdom: Amplitude of CM_X
“Freqdom-Cmx-Phase”	Freqdom: Phase shift of CM_X
“Freqdom-Cmy-Amp”	Freqdom: Amplitude of CM_Y
“Freqdom-Cmy-Phase”	Freqdom: Phase shift of CM_Y
“Freqdom-Cmz-Amp”	Freqdom: Amplitude of CM_Z
“Freqdom-Cmz-Phase”	Freqdom: Phase shift of CM_Z
“RPM-Res-P”	RPM residual on Newton subspace
“RPM-Res-Q”	RPM residual on complementary subspace.
“RPM-Dim-P”	RPM Newton subspace dimension

7.4 Signal handling

The solver is prepared to handle the following signals:
`kill -USR2`

- The solver writes out the solution after the current time step is finished.

`kill -TERM`

- The solver stops writes out the solution and exit after the current time step is finished.

Any other kill signal will not lead to a reaction of the solver signal handler.
Note: Older versions of the TAU-Code supported also ‘kill -USR1’ (reaction as today with ‘kill -USR2’ and other kill statements for ‘soft stops’. This is no longer supported for portability reasons, because Linux-mpich installations make use of the signal ‘USR1’ per default.

7.5 Description of process information

The output information of the flow solver starts with a line indicating the version and the compile time of the code. In the next line the command line is printed. This is followed by list of the implemented boundary treatments, the current monitor settings, the current multigrid cycle (only in multigrid mode) and the current parameters.
This is followed by a list showing the mapping of boundary treatments available in the preprocessing program (which has computed the dual grid) and the boundary treatments implemented in the solver (they can differ!). If incompatibilities are present they are indicated at that position: the program will stop. Behind that the active boundary mappings are printed (i.e. the boundary treatments which are in use for that configuration).
This is followed by a list of the monitoring output. Note that for the monitoring quantities Cl and Cd (lift and drag coefficient) it is assumed that the x-axis is in streamwise direction, the y-axis is in spanwise direction and that the vertical direction is along the z-axis. A different oriented grid will lead to wrong output values

8 Turbulence Modeling

8.0.1 Introduction

Virtually all aerodynamic flows of practical interest are turbulent, they are characterized by small velocity fluctuations in certain regions of the flow field. Unfortunately, owing to the immense computational demand, a direct simulation of the turbulent field is currently prohibitive in engineering applications. Thus, as the engineer's interest is geared towards the mean flow properties, the turbulent effects are usually modeled. A wide range of modeling levels are available, ranging from simple algebraic approaches to full Reynolds-Stress Transport Models (RSTM).

In the TAU-Code, the following models are available:

- One-equation eddy-viscosity models:
  - SAO-model (Spalart-Allmaras, original version)
  - SAE-model (Spalart-Allmaras, Edwards modification)
  - SAM-model (Spalart-Allmaras, modified version)
  - SALSA-model (Strain Adaptive Linear SA-model)
- Two-equation eddy-viscosity models:
  - Wilcox k-ω model
  - Menter Baseline model
  - Menter SST model
  - LEA k-ω model
  - NLR TNT Model
  - Wilcox k-ω model + SST
  - Menter 2layer k-ε model
- Explicit Algebraic Stress Models:
  - Rung RQEVM + Wilcox k-ω model
  - Wallin & Johansson EARSM (2D mean flows) + Wilcox k-ω model
  - Wallin & Johansson EARSM + TNT k-ω model
  - Hellsten EARSM k-ω model
- Reynolds stress transport models
  The Reynolds stress transport models are obtained by combining sub-models for various terms within the Reynolds stress transport equation:
  - Re-distribution (pressure-strain correlation)

    - Wilcox stress-ω (Launder-Reece-Rodi without wall-reflection)
    - SSG/LRR-ω
  - Dissipation
    - Isotropic (Rotta)
  - Diffusion
    - Simple gradient diffusion (SGDH)
    - Generalized gradient diffusion (GGDH)
  - Length scale equation
    - ω-equation according to Wilcox
    - ω-equation according to Kok
    - ω-equation according to Menter baseline model (BSL)
    - ω-equation according to Hellsten

    Note, that the SSG/LRR-ω model requires the Menter BSL ω-equation and cannot be combined with any of the other length scale equations.

- Detached Eddy Simulation,Large Eddy Simulation Models
  - Spalart-Allmaras DES
  - Menter-SST des
  - XLES

8.0.2 Model Descriptions

One-Equation Models

- **SALSA**

  The SALSA (**S***train-adaptive* **L***inear* **S***palart-Allmaras*) model is based on the original Spalart-Allmaras formulation. However, it offers an enhanced range of validity by sensitizing the production term to non-equilibrium effects. Unlike in standard one-equation approaches, which inherently contain the assumption of local equilibrium of production and destruction of turbulent energy, the reconstruction of P/ε from mixing-length hypothesis elements allows for a more realistic representation of non-equilibrium states.

Two-Equation Models

- **LEA** k-ω

  LEA (**L***inearized* **E***xplicit* **A***lgebraic Stress Model*) k-ω stems from an explicit solution to the second-moment closure in the limit of equilibrium turbulence. This explicit algebraic Reynolds-stress model can be regarded as a generalized (non-linear) two–parameter model, which retains the predictive benefits of the second–moment closure methodology, while numerical advantages of the Boussinesq-viscosity concept are conserved. Additional key features of the modeling practice are topography-independent low-Re formulations, obedience of the realizability principle, consistency to the hydrodynamic stability theory and an approximately self-consistent representation of non-equilibrium turbulence. Based on the resulting formulation, a family of low-Re number eddy-viscosity models (EVM) is devised, the simplest member being a linear truncation of the non-linear constitutive relation, cast here in terms of a Wilcox k-ω model. (Remark: The RQEVM is the corresponding quadratic model.)

Explicit Algebraic Stress Models Simple, standard Boussinesq-viscosity models have proven insufficient to correctly predict complex flow situations. Their inability to render the fundamental physics of turbulence, such as shear-stress variation and secondary effects, is a well-known fact. Thus, attention is drawn to more reliable approaches towards the modeling of turbulence. Although a general approach would, arguably, be based on a second-moment closure, present efforts try to adopt a computationally cheaper solution. Taking into account that RSTM have not yet reached the state of industrial maturity, and furthermore considering that no difference in priority between accuracy and efficiency is made in the design process, the attention is focused on Explicit Algebraic Stress Models (EASM). Unlike other, more empirical approaches to non-linear modeling, EASM are rational approximations to RSTM and thus yield clear prospects with respect to model validity.

- **RQEVM + Wilcox** k-ω

  The RQEVM (**R***ealizable* **Q***uadratic* **E***ddy* **V***iscosity* **M***odel)* approach stems from an explicit solution to the second-moment closure in the limit of equilibrium turbulence. This explicit algebraic Reynolds-stress model can be regarded as a generalized (non-linear) two–parameter model, which retains the predictive benefits of the second–moment closure methodology, while numerical advantages of the Boussinesq-viscosity concept are conserved. Additional key features of the modeling practice are topography-independent low-Re formulations, obedience of the realizability principle, consistency to the hydrodynamic stability theory and an approximately self-consistent representation of non-equilibrium turbulence. The linear truncation of the model is available as LEA k-ω

- **Wallin & Johansson EARSM (2D mean flows) + Wilcox k-ω model**

  In the Wallin & Johansson EARSM (**E***xplicit* **A***lgebraic* **R***eynolds* **S***tress* **M***odel)*, the eddy-viscosity relation is replaced by a more general constitutive relation for the Reynolds stress tensor in terms of the mean flow strain- and rotation rate tensors. This relation has been derived using a formal approximation of the corresponding second-order transport model equations in the weak-equilibrium limit. The EARSM may be applied to most existing eddy-viscosity models. Here, the simplified version for 2D mean flows it is used together with a standard Wilcox k-ω model.

- **Wallin & Johansson EARSM + TNT k-ω model**

  This is the full high-Re Wallin & Johansson EARSM, based on the TNT k-ω model by Kok. Its derivation and properties are described above, however, is has shown to yield better results than the simplified formulation based on the Wilcox model.

Stability Enhancements The TAU-Code solver shows convergence problems with two-equation turbulence models in 3D computations. To remedy this situation, measures enhancing the code's stability and robustness have been implemented.

- **Schwarz Limitation**

  This physically motivated limitation places a lower bound on the ω-equation. It is based on an integral formulation of the Schwarz inequality resulting from realizability considerations. Despite the fact that the Schwarz inequality does not constitute an additional constraint, this integral form has proven to be useful, especially during the transient phase of a computation. Owing to its derivation, this limitation is valid in a linear modeling framework only. Furthermore, strictly speaking, this formulation is coined for models with a fixed anisotropy parameter C_μ. Concerning the EASM, the Schwarz limitation should be used with caution due to its different theoretical background, although no significant problems are expected to occur in shear-dominated flows without excessive three-dimensional structures.

- **Positivity Scheme**

  It is well known that positivity of the discrete turbulent variables at any time during iteration is crucial in order to gain increased robustness in the iteration of an explicit Navier-Stokes solver coupled with a two-equation turbulence model and accelerated with a multi grid method. Unphysical negative values rapidly lead to instability and positivity must therefore be ensured.

The scheme is applied to the time stepping and the multi grid update and guarantees that the explicit time stepping or the updating of the unknown during the iteration cycling is done such that positivity is retained.

8.0.3 Input

- The turbulence models are activated via the keyword

  Turbulence model version:

- The Schwarz Limitation is invoked by

  K-omega limitation type (0/1/2): 2

- The Positivity Scheme is switched on by

  Positivity scheme: 1

8.1 Reynolds stress transport models

Reynolds stress transport models replace the Boussinesq hypothesis by individual transport equations for each component of the Reynolds stress tensor. Thus they do not involve an eddy viscosity any more, except as an auxiliary variable for computing, e. g. an eddy conductivity for the turbulent heat flux model. Furthermore they involve an additional transport equation for a length-scale supplying variable, e. g. ϵ oder ω. Currently, in TAU only ω-equations are implemented (see above).
While the production term in the Reynolds stress transport equation is given exactly, all other terms require modeling. The currently available sub-models are briefly described below.

8.1.1 Re-distribution (pressure-strain correlation)

The pressure and strain fluctuations are correlated in such a way, that fluctuations of one velocity component will excite all other velocity components to fluctuate too, thus re-distributing the kinetic energy to the different fluctuating velocity components. In incompressible flow with constant density this term is traceless.

Wilcox stress-ω model The re-distribution term of Wilcox' stress-ω model is equivalent to the well-known Launder-Reece-Rodi (LRR) model, when neglecting the so-called wall-reflection terms. The LRR model has been derived as the most general linear model (with respect to the Reynolds stresses), fulfilling the various constraints analyzed by Rotta. Wilcox has shown, that this model can be combined with an equation for ω as length scale supplying variable, when neglecting the wall-reflection terms, so that the equation system can be integrated down to the wall (low-Re model). In TAU this model can be combined with any of the provided ω-equations.

SSG/LRR-ω model The Speziale-Sarkar-Gatski (SSG) model is a highly estimated non-linear re-distribution model. However, this model is tied to an ϵ-equation for providing the length-scale. Since in aerodynamics ω-equations have shown to be more advantageous, the SSG-ϵ model has been combined with the LRR-ω (= Wilcox stress-ω) model near walls. This is achieved by blending the coefficients from the LRR values at the wall to the SSG values further apart, using the F_1 function of Menter's k-ω models.
This model must be combined with Menter's baseline (BSL)ω-equation which changes coefficients accordingly from the wall to the far field. Thus the SSG/LRR-ω model consequently transfers the principles of Menter's k-ω models into the framework of Reynolds stress transport equations.

8.1.2 Dissipation

There is one term in the Reynolds stress transport equation which describes the viscous dissipation of turbulence. Currently only one model for this process is available in TAU.

Isotropic dissipation Rotta has developed the idea of an isotropic process of dissipation, acting only on the main components of the Reynolds stresses. In TAU the dissipation ϵ is expressed in terms of ω.

8.1.3 Diffusion

There exist various diffusive processes which are usually combined within one single model. Typically the diffusive fluxes of the Reynolds stresses are assumed to be proportional to the gradient of the respective Reynolds stress component.

Simple Gradient Diffusion Hypothesis (SGDH) The diffusion coefficient is a scalar, where the turbulent contribution is proportional to an equivalent eddy viscosity.

Generalized Gradient Diffusion Hypothesis (GGDH) The diffusion coefficient is a tensor, where the turbulent contribution is proportional to the Reynolds stress tensor.

8.1.4 Length scale equation

Most Reynolds stress transport are based on an ϵ-equation. In contrast, all models implemented into TAU rely on an ω-equation.

Wilcox ω-equation The Wilcox ω equation corresponds to the original Wilcox k-ω model (1988).

Kok ω-equation The Kok ω-equation corresponds to the ω-equation of Kok's so-called TNT k-ω model, introducing a cross-diffusion term.

Menter baseline (BSL) ω-equation The Menter BSL ω-equation corresponds to Menter's baseline k-ω model, blending an ω-equation near walls with an ϵ-equation further apart. This equation is mandatory for the SSG/LRR-ω re-distribution model.

Hellsten ω-equation The Hellsten ω-equation corresponds to the k-ω/EARSM of Hellsten. It showed larger discrepancies from the log-law, when used with the Wilcox stress-ω model and is therefore not recommended to be used.

8.2 Vortical Correction Modeling

A description of the relevant parameters for activating the vortical correction terms in the turbulence model will be provided in this userguide and some advice for users will be noted. Some examples will be discussed. The gradient reconstruction that is recommended in conjuction with the SARC rotation correction model, at present, is the least-square formulation, however equivalent results are usually obtained with Green-Gauss. Problems may occur occasionally with the least-squares formulation and it is recommended that the user switch then back to the Green-Gauss formulation. The Kozlov modification, which can be activated for the SARC model, is valid only when the streamline curvature is expected to be small, for example in boundary layers which are fully attached to a surface with only a small amount of curvature. Here curvature is the second spatial derivative. There are two menu blocks associated with the vortical correction algorithm - the master block is used to set up an active vortical correction model as well as setting the smoothing control parameters. The parameter list associated with the master block, together with the active options, is given by the following:

- **Vortical flow correction model:** (none), flower, sarc, NLR, Hellsten
- **Smoothing eps for vortical flow correction:** $>= 0.0$
- **Number of smoothing steps for vortical correction:** > 00

The flower and sarc variants are active for the Spalart-Allmaras model family, whilst the NLR, Hellsten, and sarc models are active for the kω turbulence model families. It should be noted that the NLR model was developed for the TNT turbulence model, so that users are advised to use this combination. However, should the user wish to experiment, a second parameter menu block is associated with the specific vortical correction model chosen. Some figures in this section refer to a simplified SARC model (SSARC) that has not been published, however this model is valid only for a restricted set of conditions and the results are provided simply for comparison. These additional parameters are now listed, together with their default parameter values:

8.2.1 Spalart-Allmaras + flower

- **FLOWer vortical correction coefficient:** 4.0

The user is free to modify the coefficient for this model. Note that for this combination no smoothing is required.

8.2.2 Spalart-Allmaras + sarc

- **SARC vortical correction coefficients:** 1.0 12.0 1.0 1.0
- **Kozlov modification:** 0
- **Form of scaling denominator:** "Shur", "Smirnov"

The user is free to modify the coefficient for this model. Note that for this combination smoothing of the vortical correction field is usually required. The Kozlov modification is valid for small values of streamline curvature only. The form of the scaling denominator refers to the form of the term used to non-dimensionalise the estimate returned for the magnitude of the stream line curvature. "Shur" refers to the formulation noted in Shur et al. AIAA, V38, 5, pp 784-792. "Smirnov" refers to a form developed specifically to assist in removing problems associated with the "Shur" form in vanishing strain fields and is further described in Smirnov et a., Journal of Turbomachinery, V131, 2009. The "Shur" form is dominant in the literature for the SAO model with SARC corrections. Experience to data suggests that better results are obtained by first computing the problem with the SARC correction disabled (not active) and then, after a converged solution has been obtained, to activate the SARC correction. This reduces problems associated with a non-smooth vortical correction field and will minimise the amount of smoothing required.

8.2.3 k-ω + TNT

There are no additional coefficients for this model.

8.2.4 k-ω + Hellsten

- **Hellsten vortical correction coefficient:** 3.6

The user can modify the correction coefficient for the model. In order to avoid misunderstanding, this correction is not directly related to the EARSM correction also proposed by Hellsten. At present the Hellsten variant is under development.

8.2.5 k-ω + sarc

- **SARC vortical correction coefficients:** 1.0 12.0 1.0 1.0
- **Kozlov modification:** 0

- **Form of scaling denominator:** "Shur", "Smirnov"

The parameters have been described above, however the "Smirnov" form appears to be the most tested in the literature for the SST model with SARC correction. For the two equation models the sarc vortical correction model is under development. In general the advice given for SA + salsa applies here as well.

8.2.6 The influence of Vortical correction Models on a simple Delta Wing flow

In this test case we simply examine the influence of the vortical flow correction models on a simple symmetric delta wing profile. We note that the computational modeling is non physical since an axisymmetric boundary condition has been used at the wing centerline. Thus the influence of the opposing, counter-rotating vortex system generated by the opposite wing does not influence the vortex system we compute. However the point of the calculation is to simply explore the possibilities of the vortical correction models for a 3-D delta wing calculation and so we may accept this limitation. Note that the Mach number and angle of attack (AoA) for this problem have been chosen to ensure that the vortex remains stationary and with no additional phenomenon such as vortex bursting occurring. For this wing, these additional phenomena occur at AoA of sixteen degrees or greater.

Table 4 illustrates that the integrated lift, drag and moment coefficients are all quite close in value, with the greatest differences being found for the TNT model calculations in the lift coefficient(in comparison to the other models) and with the SARC model drag coefficient. However, due to the additional complexity of the SARC model, extra smoothing and perhaps more iterations may be required even though examination of the calculation residuals suggested that the value in the table is the final converged drag coefficient for the model. What is somewhat curious is the relative insensitivity of the moment coefficient to the model type. Typically the integrated moment is quite sensitive, however this flow has a vortex system dominated by the attached leading edge vortex and the intensity and location of the vortex centroid is relatively invariant from model to model. This is perhaps the reason why the moment coefficient shows little sensitivity to the model choice since the leading edge vortex is quite insensitive to the model chosen.

In Figure 6 we present contour data of vorticity taken at a plane in the spanwise direction of the wing and located halfway between the wing root at the leading edge and that at the trailing edge. For this test case it appears that the influence of the vortical correction models on the one-equation solutions is minimal at best, with the SARC model returning a qualitative improvement in vorticity distribution. The FLOWer correction model returns levels of vorticity which appear to be in good agreement with the standard turbulence model. As a point of note there is considerably more work involved in the correction of the SARC model so that for this flow, at least, it appears that one can compute without a vortical correction model. However, should one decide to use such a model then the cheapest computation is

Table 4: Comparison of Integrated coefficients for Delta wing ($\alpha = 9\,\mathrm{deg}, M = 0.4$)

Model	Abrev.	C_l	C_d	C_{my}
SA+FLOWer	RC1	0.382	0.0613	0.0122
SARC	RC2	0.384	0.0588	0.0122
TNT		0.340	0.061	0.0122
TNT+NLR	RC4	0.376	0.062	0.0123
EARSM		0.381	0.061	0.0124
XLES		0.380	0.062	0.0123

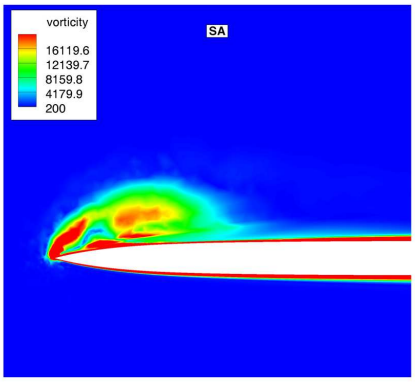
obtained with the Flower model. This situation, where the influence of the vortical flow correction play only a minor role, contrasts sharply to the two-equation calculations.

In Figure 7 it is immediately clear that TNT+NLR modification has significantly increased the resolved detail of structure in the resolved vorticity. Comparison of Figures 6 and 7 show that the basic TNT model resolves only the initial part of the vortex generated from the wing leading edge. However no secondary or induced vortices are present. In contrast, TNT+NLR appears to resolve similar structure to that returned by the SA model calculations, however the levels of vorticity are significantly lower for the induced vortical structures when compared with the SA model calculations. The extent of the leading edge vortex is also reduced.The EARSM model has reproduced all the features shown by the SA model calculation. At least, on this mesh there are no additional features resolved through the use of a non-linear eddy viscosity odel. There are small qualitative differences and it is reasonable to claim for the problem that the SARC and the EARSM models return equivalent results. Table 4 suggests that only the drag count returned by the SARC model is significantly different from that returned by the EARSM model. From a computational point of view the EARSM model is more attractive to use, since the computational requirements required for this model are only slightly more than that which is required for the SA model. However, for both the SA and $k-\omega$ implementations, it is seen that the influence of the vortical correction models has been to increase the strength of the vortex core.

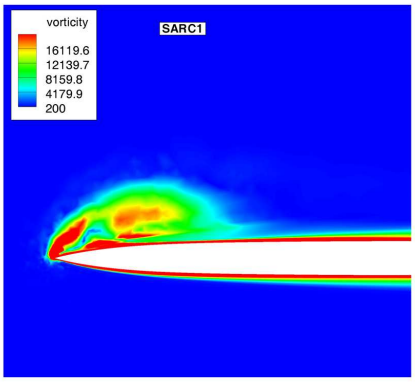
Experiences from users outside of the DLR indicates that the SARC model may reduce the effective local dissipation sufficiently low to cause convergence problems, however it may well be that for the cases tested (unknown to this author) the natural solutions were in fact unsteady solutions, which has then been recovered through the use of the SARC model. As noted earlier, if a model performs inadequately then the user must first establish that the model is in fact relevant to the problem in hand. For example, the SSARC model should not be used for a three-dimensional unsteady problem. Given the high numerical viscosity that has been associated with the base TAU $k-\omega$ implementation (Wilcox variant) the user should choose perhaps the TNT model, or perhaps the LEA model or even a full non-linear model. Upon concluding that the model is in fact valid, the user has the freedom to modify the model coefficients in accordance with the suggestions given in this section.

8.2.7 The influence of Vortical correction Models on a 2D Airfoil

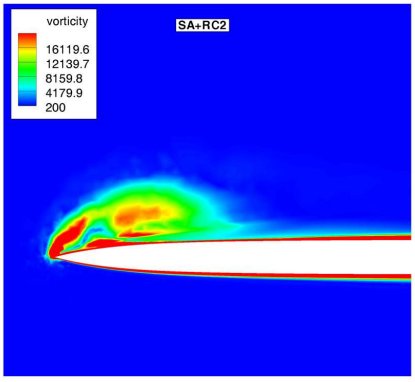
In this example the HGR01 airfoil, developed by the Technical University of Braunschweig, is used to assess the influence of the vortical correction models on the integral values of pressure about the HGR01 airfoil. The HGR01 airfoil is illustrate in Figure 8. The airfoil models a high-lift configuration, with trailing edge flaps extended, and this case is useful to assess the likely influence of the vortical correction models on more realistic high lift calculations. The



SAO

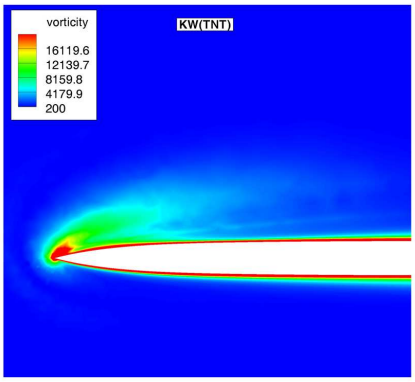


SAO+FLOWer

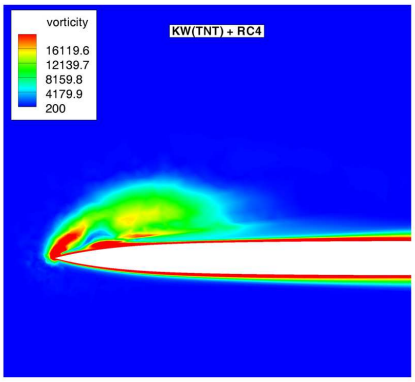


SAO+SARC

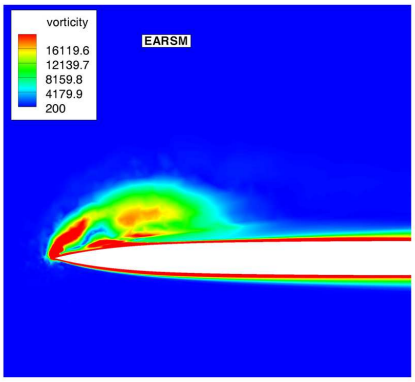
Figure 6: For this case, the qualitative improvement in field vorticity distribution due to the influence of the vortical flow correction models appears to be small for the SA model.



$k-\omega$+TNT



$k-\omega$+TNT+NLR



EARSM

Figure 7: While the base implementation of the TNT $k\omega$ turbulence model appears to be much too diffusive, significant improvement is shown with the use of the NLR vortical flow correction model. However, a non-linear eddy viscosity model (EARSM) provides better resolution of vortical structure and is only marginally more expensive

flow Mach number has been set to 0.073 for this problem, and the angle of attack (AoA) has been set to two degrees.

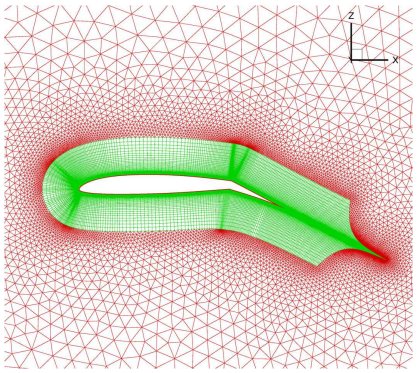


Figure 8: The HGR01 airfoil was developed by the Technical University of Braunschweig to examine the performance of the TAU-Code on a model high lift problem.

Figure 9 illustrates the surface pressure distribution about this high-lift configuration for a number of calculations based upon the Spalart-Allmaras turbulence model closure. It is immediately clear that all models return a very similar pressure distribution on the pressure side of the airfoil. However, there is a slight discrepancy between the Spalart-Allmaras plus FLOWer vortical correction model and the other computed profiles on the pressure side. At present, the exact cause for this difference has not been studied, however it is possible that an adjustment of the model coefficient can correct the problem. This may be an important observation because, as discussed above, none of the vortical correction models can be considered to be invariant under all possible transformations and this implies that there is always going to be a requirement to tune the model coefficients for certain classes of flow. On the suction side of the airfoil it can be seen that both the SARC and SSARC vortical correction models have significantly improved the peak pressure prediction at about x=0.28 meters. Here increased streamline curvature occurs since the flow remains attached to the flap surface. The pressure levels both upstream and downstream of the point where the streamline curvature suddenly increases are also improved by using the SARC model when comparison is made against the base model results.

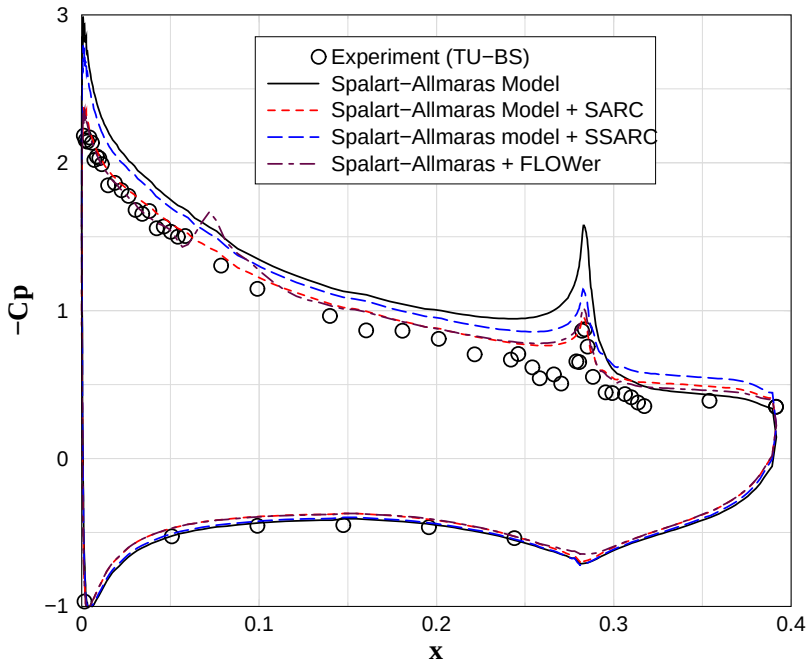


Figure 9: Surface pressure distributions computed for the HGR01 airfoil using the Spalart-Allmaras turbulence model with additional vortical flow correction models.

However, the FLOWer correction model appears to have resolved an additional feature at x = 0.06 meters. This feature does not appear to be suggested by the experimental data, or indeed by the other calculations. It may be necessary to refine the mesh in this region to better match the experimental data. Thus, while the vortical models can improve some features of a solution, these models will not in general result in an overall improvement of solution independent of the local flow topology.

9 Detached Eddy Simulation (DES) and Large Eddy Simulation (LES)

9.1 Introduction

Any turbulent flow field is comprised of a very large number of flow scales (events) which, when considered together, completely describe the evolution of the flow field in both time and space. In order to completely describe a flow field, there are then a large number of flow scales which must be accurately modeled. This can be referred to as a physical model of the flow. Detached Eddy Simulation (DES) and Large Eddy Simulation (LES) are numerical tools which allow a user to improve the level of physical modeling inherent in a solution. In order to resolve all scales in a flow, the computational work W required is of the order $W \approx O(Re^3)$, where Re is the Reynolds number of the flow. Since turbulence is characterized by three-dimensional vortex stretching and vortex-breakdown we assume only three-dimensional flow fields in this discussion, and spatial resolution requirements S scale as $S \approx Re^{9/4}$. These estimates are based on well-founded dimensional analysis that can be found in numerous sources, see for example [6] (Chapters 3 and 4). Some care must be taken in discussing the computational requirements for LES/DES in wall-bounded flows where the simple dimensional arguments given above do not hold. The reason is that the size of near wall motions containing significant energy scale with the viscous length scale which decreases with the Reynolds number relative to the flow length scale. Pope [11] references [5] who observes that for aerodynamic flows the cost of LES scales as $Re^{1.8}$, or a factor of $O(60)$ for each decade increase in Reynolds number.

There are three modes in which the TAU code can operate for viscous flow simulations and each mode corresponds to different levels of flow scale resolution. For high Reynolds number flows, the resolution of all scales present in a flow is intractable due to the significant computational resources required. For example, a DNS computation in the absence of boundary layers at $Re = 10^6$ requires $W \approx 10^{18}$ and $S \approx 10^{13}$. For low Reynolds numbers, however, simulation of all scales can be realized - this is known as Direct Numerical Simulation (DNS) and no physical modeling is present. Strictly speaking, we are solving the Navier-Stokes equations so that a claim that modeling is present in DNS represents an over-simplification which is acceptable within the context of the present discussion. At the present DNS methods are not implemented in the TAU code. The Reynolds Averaged Navier-Stokes (RANS) equations represent the situation where the largest integral scales are resolved and all other scales are modeled. In particularly all the scales involved in the transfer of energy into heat through viscous dissipation are modeled (i.e. -5/3 part of Kolmogorov spectrum - Unsteady RANS (URANS) cannot properly represent the energy cascade across neighboring wave numbers in the inertial interval). There is no requirement to resolve any particular range of scales and RANS represents the most cost effective way to compute flow problems at the loss of solution complexity. RANS represents the natural mode of operation for the TAU code. The

remaining two modes available to TAU are the DES and LES modes. Both DES and LES solutions contain higher resolved physical scale then URANS, but usually much less physical content than DNS. Correspondingly, the costs of DES and LES are higher than RANS but lower than DNS. Exact requirements for LES depend on the form of LES calculation undertaken: Near Wall Resolved LES (NWR-LES) resolves the significant energy containing near wall scales at a cost of approaching that of a DNS calculation. Near Wall-Modeled LES (NWM-LES) models all scales in the near wall regions and has a cost commensurate with that of URANS. The reader is referred to Pope for a more complete discussion on LES. As of Release 2009.2, the Smagorinsky Sub Grid Scale (SGS) model [18] is implemented in TAU, as is the Wall-Adapting Local Eddy-Viscosity (WALE) SGS model due to [7]. These models can be considered as NWR-LES implementations.

The DES mode will usually provide similar resolved content to NWM-LES with similar computational requirements. Detached Eddy Simulation (DES) is a comparatively new technique, the basic concepts of which were first elucidated by [15]. DES belongs to the class of turbulent models known as *Hybrid RANS-LES* methods. DES is a non-zonal hybrid method in that the user is not required apriori to specify which parts of the solution domain should be computed as RANS or LES. Since computational requirements for resolution of the scales responsible for flow evolution in the boundary layer are excessive, DES uses a hybrid length scale definition to switch between RAN in attached boundary layers and LES in the far field. Thus mesh resolution requirements are significantly reduced in near-wall regions when compared to LES. Computational requirements are correspondingly reduced for DES (as with NWM-LES). The cost of NWM-LES is of the order of DES, and is independent of Reynolds number or increases as a weak function of $ln(Re)$. The following table, adapted from [11] (Table 13.1) shows the range of applicability of the three turbulence modes (RANS, DES, LES) that are currently implemented in TAU,

Table 5: Physical Modeling implemented in TAU (10.09.09)

Turbulent mode .	Maximum Re	percent energy resolved
RANS	$> 10^7$	$O(0)$
NWR-LES	$O(10^6)$	$O(80)$
NWM-LES	$> 10^7$	$> O(80)$ in farfield, $O(0)$ near wall
DES	$> 10^7$	> 80 in farfield, $O(0)$ near wall

In summary, DES is a form of NWM-LES with an implicitly defined filter function (see [11]) for definition of filter function) and a non-zonal transition mechanism between RANS and LES branches of the DES model. In LES the filter function is explicitly defined, with the exception of MILES (Monotonic Implicit LES) methods where the filter function is again implicitly defined in some fashion via the numerical dissipation of the numerical method.

LES and DES return solution fields appear stochastic in nature so that statistical moments of the solution are required. In the following discussions, an over-bar denotes an averaging operator (eg. time or spatial) so that $\overline{a}$ represents an averaged value of the quantity a.

9.2 Setting the filter width Δ for LES/DES computation

In order to compute LES or DES solutions a filter width must first be chosen. At present there are three filter width specifications that have been implemented in the TAU code. These must be specified set in order to run either a DES or LES simulation - the code will terminate if the appropriate parameters are not set as the user must specifically choose the filter scale that he will use - default values are deliberately NOT set. Possible options are as follows:

1. **Edge length** filter width: This is the classic DES filter width definition and is computed as the maximum edge length of a dual cell control volume on the dual grid metric, or the maximum distance between neighboring grid points on the cell-centered metric. A dual-grid-like-edge-length algorithm for the cell-centered metric has not yet been implemented.
2. **Volume** filter width: This is the classic LES filter width definition and is given simply as $\Delta = V^{1/3}$ where V is the cell volume. This quantity is identical for both dual and cell-centered metrics (on the finest grid level).
3. **User** defined filter width: This setting allows the user to set an arbitrary fixed filter width in grid units. Users need to understand the implications of their choice of filter width, especially with relation to the grid length scales. Care is required if this definition is chosen.

The typical parameters for setting of the filter width model in TAU are shown below with their default settings:

```
LES/DES filter width selection --------------------: -
                                  Filter width model: Edge
                                 Hybrid switch model: Edge
                            User defined filter width: -1
                  Smoothing iterations for LES filter: 0
                     Smoothing Epsilon for LES filter: 0.2
                                Ratio of delta over h: 1
```

The **Filter width model** parameter is chosen by using the parameter values "Edge", "Volume" or "User" as defined above. The parameter **Hybrid switch model** is used to determine the length scale definition used to switch between the RANS and LES modes of the DES model. At present it has no meaning in the LES implementation. Its function is to offer

an alternative definition between the LES length scale and the switch length scale. It is often useful, on problematic grids, to smooth the LES filter width or the hybrid switch length scale. The smoothing parameter options can be set to ensure that the scales are smoothed helping to reduce numerical noise introduced by unphysical RANS/ LES mode switching. Note that smoothing of the wall normal cell spacing (required for IDDES) appears to be quite important for the IDDES mode of DES, but at present we have insufficient experience with this mode to make definitive suggests. Finally the parameter **Ratio of delta over h** can be used to either increase the resolved content of the solution at a cost of numerical accuracy (increase the parameter value), or to decrease the level of resolved structure whilst increasing the numerical accuracy of the resolved scales (reduce the parameter). See for example [11] for further discussion on this point. It should be pointed out that at present this parameter has the same influence as is achieved by rescaling the DES/LES coefficient for single coefficient models.

9.2.1 Large Eddy Simulation Sub Grid Scale Models

As can be seen in [11], the LES method requires the solution of filtered transport equations which are generated by integrating the product of a filter operator with the instantaneous Navier-Stokes equations over a suitable space. In order to close the filtered transport equations, a model for the residual terms is required. For the filtered momentum equations a residual stress tensor term must be modeled. Smagorinsky [18][1963] invoked the concept of a linear eddy-viscosity model together with the mixing length hypothesis (see for example [20]) to write the residual stress tensor in terms of a turbulent eddy viscosity:

$$\nu_T = l_m^2 \left(2\overline{S}_{ij}\overline{S}_{ij}\right)^{1/2} = \left(C_S\Delta\right)^2 \overline{S}, \tag{1}$$

where Δ is the filter width, and C_S is a dimensionless scaling coefficient. From the literature $C_S \approx O(0.16)$. Note that the explicit form of the filter does not appear in the filtered equations, nor in the modeled residual stress tensor. Nevertheless, if the exact form of the filter is known it is possible to derive theoretical bounds and asymptotes for the filtered equations which can help in analysis of residual error estimates and so on. This cannot be done for DES because the precise form of the filter function cannot be stated with any certainly. The specification of the modeled residual stress as described by the Smagorinsky closure is only valid in the inertial subrange of high Reynolds number turbulence. This is not the situation in the near-wall region where the Smagorinsky model results in non-zero residual viscosity at the wall. In order to reduce the near wall eddy viscosity for the Smagorinsky SGS model, Moin et al.[10] define a damping function

$$f_d = \left[1 - exp\left(y^+/A^+\right)\right] \tag{2}$$

so that the mixing length term is then written as $f_d C_S \Delta$. A^+ is a coefficient which takes a value of 26 usually in the literature. The WALE model from Ducros et al. [7] is an attempt

to recover the correct functional relationship between the normalized wall distance y^+ and the sub-grid scale viscosity ν_{sgs} as the wall is approached - $\nu_{sgs} \propto y^{+3}$. The WALE eddy viscosity model is written as

$$\nu_t = \Delta_s^2 \frac{\left(S_{ij}^d S_{ij}^d\right)^{3/2}}{\left(\overline{S_{ij}S_{ij}}\right)^{5/2} + \left(S_{ij}^d S_{ij}^d\right)^{5/4}}, \tag{3}$$

$$\Delta_s = \Delta C_S, \tag{4}$$

$$S_{ij}^d = \frac{1}{2}\left(\overline{u}_{i,j} + \overline{u}_{j,i}\right) - \frac{1}{3}\delta_{ij}\overline{u}_{k,k}. \tag{5}$$

In comparison to the Smagorinsky model no addition rescaling is required in the near wall regions. As of Release 2009.2 the WALE model is preferred over the Smagorinsky model since Van Driest damping is not yet fully implemented. Typical parameter settings required to drive the LES modes of the solver are given below, with the default coefficient values.

```
Turbulence ------------------------------------------: -
                                    Turbulence mode: LES
LES switches ----------------------------------------: -
                                          SGS model: Smagorinsky, WALE
                                    SGS Coefficient: 0.13, 0.55
```

Solver settings that can help optimize the LES solution (ie increase resolution of turbulence) are discussed below after a brief discussion of DES implementation in TAU. Note that for the WALE model it is recommended to start with a RANS solution or with a Smagorinsky LES solution to avoid possible overflow/underflow problems in computing the WALE SGS viscosity.

9.3 Detached Eddy Simulation within the Spalart-Allmaras model family

The transport equation for the turbulent eddy viscosity for the Spalart-Allmaras eddy viscosity model is given by the following equation [16]

$$\begin{aligned}\frac{D\tilde{\nu}}{Dt} &= C_{b1}\left[1 - f_{t2}\right]\tilde{S}\tilde{\nu} - \left[C_{w1}f_w - \frac{C_{b1}}{\kappa^2}f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^2 + \\ &\quad \frac{1}{\sigma}\left[\left(\nu + \tilde{\nu}\right)\tilde{\nu}_{,k}\right]_{,k} + \\ &\quad \frac{C_{b2}}{\sigma}\tilde{\nu}_{,k}\tilde{\nu}_{,k}.\end{aligned} \tag{6}$$

Note that the transition term and trip length specification found in the original model have not been included into the standard TAU Spalart-Allmaras implementation and are not included in this discussion. The expressions used to close eqn. (6) are given below and will

be referred to in the discussion on methods chosen to deactivate the low Reynolds number terms of the model equation above. For high Reynolds number flows, as will be discussed shortly, it is permissible to set $f_{t2} = 0$. This is the standard implementation within the TAU code, however in near viscous wall regions where the transport length scale does not scale with the reference Reynolds number and for computations at lower Reynolds number the default value of 0 may influence the solution. Specifically f_{t2} allows a basin of attraction to a zero value of the Spalart-Allmaras transport variable and is discussed in [13]. The parameter **SA attractor for zero value (0/1)** should be set to one to activate the f_{t2} in eqn. (6). It is recommended that this term be activated for DES computations from discussions with DESider colleagues. The DLR experience with this switch is limited at present.

$$\begin{aligned}
\nu_t &= f_{v1}\tilde{\nu}, && (7)\\
f_{v1} &= \frac{\chi^3}{\chi^3 + C_{v1}^3}, && (8)\\
\chi &= \tilde{\nu}/\nu, && (9)\\
f_{t2} &= C_{t3}\exp^{\left(-C_{t4}\chi^2\right)}, && (10)\\
\tilde{S} &= S + \frac{f_{v2}\tilde{\nu}}{\kappa^2\left(C\Psi\Delta\right)^2}, && (11)\\
f_{v2} &= 1 + \frac{\chi}{1+\chi f_{v1}}, && (12)\\
f_w &= g\left(\frac{1+c_{w3}^6}{g^6+c_{w3}^6}\right), && (13)\\
g &= r + c_{w2}\left(r^6 - r\right), && (14)\\
r &= \frac{\tilde{\nu}}{\tilde{S}\kappa^2\left(\Psi C\Delta\right)^2} && (15)
\end{aligned}$$

In equation (6), d is the length scale over which the destruction of turbulence occurs. It is this length scale that is redefined vis the hybrid model length scale definition. It will be shown shortly that if production and destruction of turbulence are equal and that both dominate convective and diffusive transport, the turbulent eddy viscosity can be expressed as $C_{Smag}^2 = C_{SA}^2 * \phi\left(SA\right)$ where $\phi\left(SA\right) = 0.042$ from the standard Spalart-Allmaras model coefficients and the destruction length scale is equal to the filter width $d = \Delta$. Thus by replacing the length scale of the destruction term we are able to generate LES content in a solution. It is possible to form a Smagorinsky eddy viscosity model from any eddy viscosity based turbulence model. These models are non-zonal because the switching between the RANS and LES parts of the flow is controlled by the prescription of the destruction length scale which cannot be specified a priori. In the hybrid Spalart-Allmaras model the switch is chosen according to

$$\tilde{l} = \min\left(d_w, C_d\Delta\right), \qquad (16)$$

where d_w is the distance to the nearest viscous wall and $C_d\Delta$ is the product of a scaling coefficient and the DES filter width Δ. The coefficient has a value of 0.65 when optimized for decaying homogeneous isotropic turbulence, however, for TAU, values in the range $C_d \in [0.45, 0.65]$ have been found to return reasonable spectral content for the DIT problem. Here reasonable spectral content implies that the energy spectrum is well-resolved over all resolvable wave numbers. The parameter **SA DES constant** can be used to modify the values of the scaling parameter for the DES length scale. For more information on the background of the DES approach we refer to e.g. the textbook by Sagaut, Deck and Terracol[17]. In order to activate an appropriate DES model, the user is required to set the following parameter as follows:

```
DES switches ------------------------------------: -
DES model: SA, SST, XLES
Number of RANS cut-out boxes: 2
RANS box support x:  0.3  0.4   0.8  0.95
RANS box support y: -10   10   -10   10
RANS box support z: -0.5  0.5   -2   2
```

The parameter **Number of RANS cut-out boxes** is used to set a number of boxes within which only the RANS length scales are active. The coordinates of these boxes are given by the parameters **RANS box support x** etc. The box is defined by the lower left hand side and the upper right hand side. In the example above the first box extends over x=0.3 to 0.4, y= -10 to 10, and z=-0.5 to 0.5. The second box extends from x=0.8 to 0.95, y = -10,10, z = -2, 2. We are still accumulating experience with this parameter but our initial studies suggest that in regions where the hybrid model can be interpreted as acting like an under-resolved LES, or as an RANS model with insufficient eddy viscosity, an excess of resolved turbulent energy can be produced. This produces structure which impacts into the RANS region and it is not clear how to properly treat the concept of resolved and residual turbulent stresses for this condition. One idea is to set a RANS cutout box in regions where the grid is not fine enough to properly resolve the turbulent kinetic energy, however further work on this area is in progress. The typical parameter settings for the DES modes of TAU are given by the following parameter settings where for "Grid shield model" either "DES", "DDES" or "IDDES" is chosen. Similarly, for the "Low Re model" parameter which deactivates the low Reynolds number terms of the model "(none)" should be chosen for $k\omega$ based DES since for the SST and XLES models there are no low Reynolds number terms in the model.

```
SA-DES switches----------------------------------: -
                                Grid shield model: DES, DDES, IDDES
                                     Low Re model: (none), NTS, Breuer
                 Fd switch uses SA viscosity (0/1): 1
                                   SA-DES constant: 0.65
                               Fixed RANS distance: -1
```

The parameter **Grid shield model** relates to the DES mode of protection against the problem of Modeled Stress Depletion, or Grid Induced Separation [14] The **DES** setting refers to the standard Detached Eddy Simulation model [15]. The **DDES** setting refers to the Delayed DES model which was first proposed by Spalart et al [2005]. In this version, a switch is used to sense the boundary layer and to inhibit the penetration of the LES mode into the boundary layer. The switch is written as follows:

$$r_d = \frac{\nu + \nu_t}{(S_{ij}S_{ij})^{1/2}\,\kappa^2 d^2} \tag{17}$$

$$f_d = 1 - \mathbf{tanh}((8r_d)^3) \tag{18}$$

$$l_d = l_r - f_d \mathbf{max}(0, l_r - l_l) \tag{19}$$

where l_d is the destruction length scale for the turbulence, l_r is the RANS turbulent length scale, and l_l is the LES length scale. Note that for the Spalart-Allmaras model it is also permissible to use the Spalart-Allmaras variable in lieu of $\nu + \nu_t$ in the definition of r_d. This is enabled via the parameter **Fd switch uses SA viscosity (0/1)**. This can result in a smoother profile of f_d in the wall-normal direction but this is somewhat grid dependent. The **IDDES** setting invokes the Improved Delayed DES model proposed by [3]. This is a new model and to date little experience has been gained with it. In principal the model extends the LES region into the BL if the grid can support a LES in the BL - this then is more akin to a NWM-LES. Note that filter smoothing should be used for the IDDES branch of DES.

The parameter **Low Re model** is used to deactivate the Low Reynolds number terms of the base SA model in high Reynolds number regions. In other words, the model disables the near wall corrections imposed on many RANS models. There are at present two options implemented: The **Breuer** modification is taken from Breuer et al. [9]. The NTS modification [14] is the preferred option. Note that of Release2009.2, the NTS option has only been implemented for the SA based DES variants. Note that is it active for DES, DDES and IDDES. It is also possible to force the DES mode into a simple zonal LES mode by using the parameter **Fixed RANS distance**. In all DES variants, the RANS mode is activated if the distance to the nearest wall is less than the value set by the parameter.

9.4 Detached Eddy Simulation within the $k\omega$ model family

As note above, any eddy viscosity model can be modified to provide LES solution content under an equilibrium constraint. The Menter-SST model was modified [2] due to its favorable properties in free shear flows. The length scale returned by the $k\omega$ turbulence model is given by

$$l_{k\omega} = k^{1/2} / (\beta^* \omega)\,. \tag{20}$$

In order to keep a simple formulation, and to allow the SST model to reduce to a Smagorinsky-like eddy viscosity model under the equilibrium assumption, the destruction term of the k

transport equation is modified as

$$D_k = \rho\beta^* \frac{k}{\omega} = \frac{\rho k^{3/2}}{l_{k\omega}}, \tag{21}$$

so that the length scale is set as $\tilde{d} = \min(l_{k\omega}, C_{DES}\Delta)$. Now the Menter-SST model has both $k-\epsilon$ and $k\omega$ branches. The model is designed so that the $k\omega$ branch is active in the near-wall region (since $k\omega$ model returns correct near wall behavior without recourse to wall-damping functions) and the $k-\epsilon$ model is active in the regions away from boundary layers (since the $k\omega$ model retains an unphysical reliance on the farfield boundary conditions). Thus it is the $k-\epsilon$ branch which is relevant from the DES point of view. Following Travin et al., we use the Menter blending function F (which links the two branches of the model) to blend the DES coefficients so that

$$C_{DES} = (1-F)\, C_{DES}^{k-\epsilon} + F C_{DES}^{k\omega}. \tag{22}$$

At present the TAU default values for the DES coefficients have been set as those recommended by Travin et al. so that $C_{DES}^{k-\epsilon} = 0.61$ and $C_{DES}^{k\omega} = 0.78$. The parameter **SST DES constants** can be used to modify $C_{DES}^{k\omega}$ and $C_{DES}^{k-\epsilon}$ in this order.

An additional model currently implemented in TAU is the X-LES formulation developed by [8]. This model uses a RANS $k\omega$ model with a k equation SGS model. The model directly modifies the dissipation term of the $k-$equation as well as the length scale of the eddy viscosity model. Choosing the RANS length scale as $l = \frac{k^{0.5}}{\omega}$ and the SGS filter width as Δ, the dissipation length scale is then chosen as $\tilde{l} = \min(l, C_1\Delta)$. The dissipation term and the eddy viscosity are then written as

$$\nu_T = \tilde{l}k^{0.5} \quad \text{and} \quad \epsilon = \beta_k \frac{k^{3/2}}{\tilde{l}}.$$

The default value of the parameter C_1 has been set to $C_1 = 0.07$, in accordance with the recommendations of the NLR for vortical flows. However, the model is recalibrated to $C_1 = 0.05$ for the DIT problem, and the user can select an appropriate constant by setting the parameter **XLES constant**.

In terms of usage, it has been found that it is sometimes necessary to tune the value of C_1 to a much lower value in order to allow the turbulence to develop. After and appropriate period, the coefficient can then be returned to the calibrated value. In general we have found the XLES model to be quite robust, however there are some differences between DES and XLES solutions, particularly in the structure of free shear layers, that are not yet fully understood. The model returns good comparative integral coefficients with other one and two-equation DES models for an airfoil at post-stall conditions. The model also performs well for cavity flows. The settings for two-equation DES modes are as follows

```
DES switches ----------------------------------------: -
                     DES model:  XLES,  SST
XLES-DES switches -----------------------------------: -
                           XLES constant:  0.06
                     Fixed RANS distance  -1

SST-DES switches ------------------------------------: -
                           Grid shield model: DES, DDES
                           Low Re model: (none)
                           SST-DES constant: 0.78 0.61
                           Fixed RANS distance: -1
```

Note that there are two possible choices for the parameter **DES model**. These have been described earlier. Note that the low Reynolds number modifications must be implemented for all two equation linear eddy viscosity models, so as of to 10.09.09 the parameter **Low Re model** has no influence. This will be modified before 2010. At present the XLES version of DES is well tested and the experience with this model is quite positive. The SST branch however remains relatively untested.

9.5 Deactivating Low Reynolds number model terms in the LES branch of DES

Many RANS models retain some specific damping functions that act to return physically correct $y+$ dependence of the solution in near wall regions. While these are necessary in the near wall region, the presence of these damping functions in the LES mode of the DES acts to detract from the LES capability of the DES model. Specifically, high wave number resolution is adversely affected. Consequently, it is usual to deactivate these additional model damping constraints in the LES branch by either specifically deactivating certain terms in the LES branch, or by modifying the length scale definition to account for the Low Reynolds number terms. The following subsections discuss the methods implemented in TAU to deactivate the Low Reynolds numbers terms in the LES branches of the respective DES models.

9.5.1 Deactivation of the Low Reynolds number terms in the Spalart-Allmaras model

The simplest way to deactivate the Low Reynolds number terms is to deactivate them explicitly. This was done in Breuer [9] where the low Reynolds number terms of the Spalart Allmaras model are specifically set to their high Reynolds number asymptotes in the LES mode of the hybrid model.

$$f_{v1} = 1.0. \tag{23}$$

$$f_{v2} = 0.0. \tag{24}$$

$$f_{w} = 1.0. \tag{25}$$

Travin et al. [2] approached the problem of Low Reynolds term deactivation in a different way. The derivation of the correction term was originally detailed in the DESider project deliverable [1]. Rather then directly disable the Low Reynolds number terms, they sought length scale solutions which automatically compensates for these corrections. They begin by redefining the DES length scale as

$$\tilde{l} = \min(l_{tm}, \Psi(\nu_t/\nu)C_d\Delta) \tag{26}$$

where $\tilde{l}$ is the destruction length scale of turbulence, l_{tm} is the length scale of the RANS model, and C_d is the DES coefficient which sets the LES length scale of the hybrid model. The difference between the traditional length scale definition (given by eqn. (16)) and eqn. (26) is in the addition of an additional function Ψ where the implied dependence on the eddy viscosity ratio will be suppressed for convenience. As the Reynolds number increases, the limit of ν_T/ν increases and the subscript $\star$ will be used to note the value of a variable at infinite Reynolds number. In practical terms it suffices that the Reynolds number is large. The following table illustrates the required limits of Ψ in order that eqn. (26) retains the properties of a hybrid lengthscale.

Table 6: Required limiting behavior as function of Reynolds Number Re

$f(Re)$	$\lim_{Re\to 0} f(Re)$	$\lim_{Re\to \infty} f(Re)$
ν_t/ν	0	∞
Ψ	∞	1

If production of turbulence P and destruction of turbulence D are equal and dominate over the convective and diffusive transport of turbulence, the transport equation for the Spalart-Allmaras variable (6) can be simplified to the following equations for time independent flow:

$$P = C_{b2}\left(1 - f_{t2}\right)\tilde{S}\tilde{\nu}, \tag{27}$$

$$D = \left[C_{w1}f_w - \frac{C_{b1}}{\kappa^2}f_{t2}\right]\left(\frac{\tilde{\nu}}{\Psi C_d\Delta}\right)^2, \tag{28}$$

$$P = D. \tag{29}$$

Substituting eqns. (27),(28) into eqn. (29) allows the following equation to be written for the equilibrium value of the Spalart-Allmaras transport variable:

$$\tilde{\nu} = \frac{C_{b1}\left[1 - f_{t2}\right]\left(\Psi C_d\Delta\right)^2\tilde{S}}{C_{w1}f_w - C_{b1}f_{t2}/\kappa^2}. \tag{30}$$

A Smagorinsky type eddy SGS model is obtained through dividing and multiplying eqn. (30) by the filtered strain invariant $\overline{S}$ after using the definition of turbulent eddy viscosity for the

Spalart-Allmaras model (7):

$$\tilde{\nu}_t = \frac{f_{v1}\kappa^2 C_{b1}\left[1-f_{t2}\right]\tilde{S}\left(\Psi C_d\Delta\right)^2\overline{S}}{\left(\kappa^2 C_{w1}f_w - C_{b1}f_{t_2}\right)\overline{S}}, \tag{31}$$

$$C = \frac{f_{v1}\kappa^2 C_{b1}\left[1-f_{t2}\right]\tilde{S}}{\left(\kappa^2 C_{w1}f_w - C_{b1}f_{t_2}\right)\overline{S}}, \tag{32}$$

$$\tilde{\nu}_t = C\left(\Psi C_d\Delta\right)^2\overline{S}. \tag{33}$$

Taking the limit $\nu_t/\nu \rightarrow \infty$ where $\Psi(\nu_t/\nu) \rightarrow 1$, implies at high Reynolds numbers the equivalence of equations (1) and (33), which can be written as

$$C\Psi^2 = C^\star\Psi^{\star 2} = C^\star. \tag{34}$$

which, upon rearrangement, provides an expression for Ψ.

$$\Psi^2 = \frac{C^\star}{C} \tag{35}$$

The high Reynolds number of the coefficient C must now be derived. From the limiting behavior required in table (6) and from equation (9), it is clear that $\chi^\star = \infty$ which then implies from equation (10) that $f_{t2}^\star = 0$. Similarly $f_{v1}^\star = 1$ in the specified limit. Substitution of $\chi^\star$, $f_{v1}^\star$ and $f_{t2}^\star$ into equation (31) then gives

$$\lim_{\nu_t/\nu\rightarrow\infty}\left[\text{Eqn. (31)}\right] = C^\star = \frac{C_{b1}}{C_{w1}f_w^\star}. \tag{36}$$

Substituting in equations (27) and (28) into (15) gives

$$r = \frac{C_{b1}\left[1-f_{t2}\right]}{\kappa^2 C_{w1}f_w - C_{b1}f_{t2}}. \tag{37}$$

which can be used to estimate $f_w^\star$. Substituting in (11) , (37) and (15) gives the following expression for the ratio of $\overline{S}/\tilde{S}$ in (31).

$$\frac{\overline{S}}{\tilde{S}} = 1 - \frac{f_{v2}C_{b1}\left[1-f_{t2}\right]}{\kappa^2 C_{w1}f_w - C_{b1}f_{t2}}. \tag{38}$$

Substituting (38) into (31) gives the following expression for $1/C$

$$\frac{1}{C} = \frac{\kappa^2 C_{w1}f_w - C_{b1}\left[f_{t2} + f_{v2}\left(1-f_{t2}\right)\right]}{\kappa^2 C_{b1}\left(1-f_{t2}\right)f_{v1}}. \tag{39}$$

Substituting (39), (36) into (35) gives a final expression which relates Ψ to the equilibrium coefficients of the Spalart-Allmaras variable.

$$\Psi^2 = \frac{\frac{f_w}{f_{w\star}} - \frac{C_{b1}}{\kappa^2 f_{w\star}}C_{w1}\left[f_{t2} + f_{v2}\left(1-f_{t2}\right)\right]}{\left(1-f_{t2}\right)f_{v1}} \tag{40}$$

Following the advice of Garbaruk et al., the $(1 - f_{t2})$ term in the denominator is given a lower bound of 10^{-10} to avoid floating point overflow problems. Similarly Ψ is given an upper bound of 100, however this bound may change as experience with these models grow. It now remains to fix a limiting value for $f_w^\star$. $f_w = f_w\left(g\left(\chi, f_w\right)\right)$ can be solved with an iterative scheme to yield a value of $f_w^\star \approx 0.424$. The logic developed by Garbaruk et al, can be applied to any base URANS model in order to properly deactivate the low Reynolds number terms of the model: 1) Assume equilibrium conditions, 2) express the reduced model in terms of an equivalent SGS model, 3) require that $\Psi\left(\nu_t/\nu \to \infty\right) \to 1$ and $\Psi\left(\nu_t/\nu \to 0\right) \to \infty$ and formulate an expression for Ψ in terms of the equilibrium coefficients and the limiting values of these coefficients. Clearly, slightly different treatments of the production and destruction terms may yield slightly different forms of Ψ and, as of Release 2009.2, the NTS based formulation implemented in TAU is formally correct for only the SAO model. There is now extensive experience with the hybrid SAO model and it is not recommended at present that SALSA or the other variants should be run in DES mode. This will change as more experience is gathered.

9.6 Methods to improve quality of DES results

Optimal LES/DES requires that a sufficiently low numerical dissipation can stabilize the code. With this in mind it is almost essential to use the matrix dissipation model for the Jameson Schmidt Turkel dissipation model if central differencing is used. Similarly, if the Mach number is sufficiently low then matrix dissipation with low Mach number pre-conditioning should be used. The following parameter settings are what one should normally use if the central differencing scheme is chosen.

```
 Flux main ---------------------------------------: -
                  Inviscid flux discretization type: Central
                   Viscous flux type TSL/Full (0/1): 1
Central flux -----------------------------------: -
                        Central dissipation scheme: Matrix_dissipation
                   Central convective meanflow flux: Average_of_flux
                                Inviscid flux form: SkewSymetric
                 Central convective turbulence flux: Average_of_flux
                  2nd order dissipation coefficient: 0.0
          Inverse 4th order dissipation coefficient: 84
              Use modified dissipation for 2D (0/1): 0
  Matrix Dissipation -----------------------------: -
               Matrix dissipation terms coefficient: 0.8
  Minimum artificial dissipation for acoustic waves: 0.1
        Minimum artificial dissipation for velocity: 0.1
```

```
Multigrid ---------------------------------------: -
                              MG description filename: >sg
```

We choose a Multigrid scheme eg. 3w++ to improve solution convergence rate. It is necessary that the equation error residuals will be sufficiently small to ensure that the continuity equation is sufficiently well satisfied. This is required to ensure that the skew symmetric inviscid flux operator performs correctly as an energy-conserving operator. For example see Kok[2004], If no shocks are present in the solution, the second order dissipation coefficient should be tuned as low as possible without damaging solution stability. We have less experience with optimal upwind settings, however at the present time the AUSMPUP scheme is preferred on the basis of limited tests. If the reference Mach number is low, or even if a sufficiently large area of recirculation exists in the flow, the low Mach number pre-conditioning will be useful. However, regardless of what tricks are employed to more accurately solve the discrete equation set or to improve the background mathematical model, the simple fact of the matter is that the DES method is very grid dependent - the use of DES with an inappropriate mesh will give poor results and you cannot blame the numerical implementation for this! It is highly advised for potential users to read and familiarize Spalart [12]. This can save a user considerable time and cost.

9.7 Using DES/LES

It is probably useful to note a short check list of points prior to starting a and whilst running a DES/LES computation.

1. On basis of your URANS calculations estimate the total resources required for LES/DES and ensure that they are adequate.

2. Does your grid adequately resolve the LES regions? It is probably useful to run a RANS simulation first and to then use the results to choose the criterion for the DES/LES mesh.

3. Does the chosen time step adequately resolve the time scales of the energy containing eddies in the flow of interest? Note that the standard URANS convergence metric criteria in determining an optimal time step is non-valid in LES (both LES and LES mode of DES).

4. Are equation residuals sufficiently reduced? The continuity equation residual must be sufficiently small to preserving the energy conservation property of Skew.Symmetric flux operator. If not, the CFL number/the number of inner iterations per time step can be reduced/increased.

5. Periodically check the convergence of the inner iterations.

6. Continuously monitor useful metrics such as the instantaneous integral coefficient values, the "running" means of the integral coefficients, the total resolved turbulent kinetic energy, the viscous dissipation.

7. For semi-infinite span problem one should check if the spacing of the distance across the span is sufficiently large to allow proper development of the solution. For example, it is known that for quasi-3d airfoils, the distance between the boundaries in the span wise direction needs to be at least 4-5 chord lengths.

8. For some problems it is helpful to start with a reduced LES/DES coefficient and to allow the resolved turbulent field to develop. Once developed, the LES/DES coefficients can be slowly readjusted to the calibrated values. You must finally use the calibrated values or you can damage the energy exchange cascade across neighboring wave numbers.

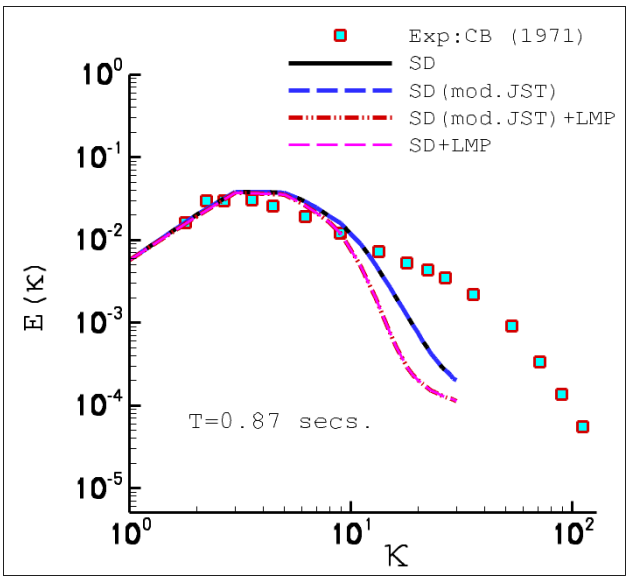
9. Examination of figure 10 illustrates that if the central scheme is active, the Matrix form of the dissipation model should be used. If the Mach number is less that 0.15 Low Mach number pre-conditioning should be used.

10. At present it is recommended that least-squares gradient reconstruction be used in preference to the Green-Gauss gradient reconstruction. Isotropic turbulence tests suggest that at high wave numbers the least-squares algorithm has more favorable dissipation properties that the Green-Gauss algorithm.

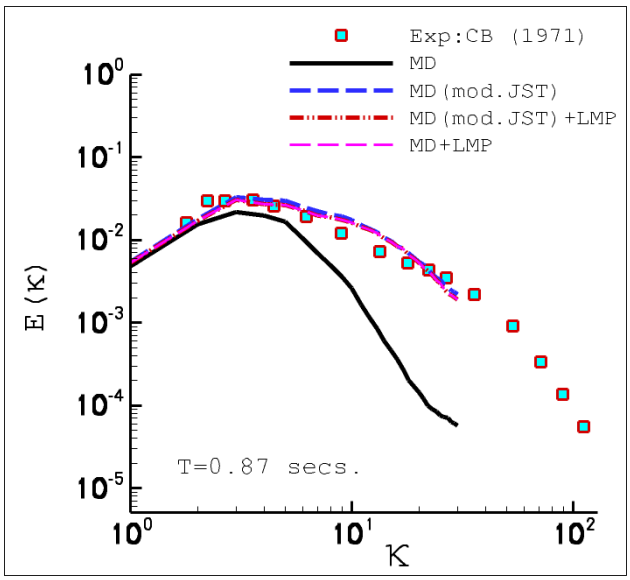
9.7.1 Using monitoring variables to assist in analysis of DES/LES solutions

. Choice of appropriate monitoring variables can assist in analysis of LES/DES solutions. To this end the "analysis" tool, described later in this guide is quite useful. A typical parameter file for the analysis tool is given below.

```
              Analysis file: <name of file containing list of stdout files>
           Analysis request: join
    Analysis profile request: (none)
     #     Analyse variables: Iter_C-lift_C-drag
    Analyse inner loop (0/1): 0
     Average variables (0/1): 1
     Number before averaging: 20
  Set zero time origin (0/1): 0
           Recover time step: 0
             Required format: %-15.7e
               Output period: 0
               Output format: tecplot
             Output filename: CASE7
```



(a) Scalar JST model



(b) Matrix JST model

Figure 10: These two figures show the resolved energy spectrum at t=0.087 obtained using the Smagorinsky SGS model and an initial velocity field derived from the Comte-Bellot (1971) experiments. The reference Mach number of the flow is $M_\infty = 0.008$, which corresponds to the maximum rms velocity since mean velocity is zero. In (a) the high numerical dissipation associated with the scalar form of the Jameson-Schmidt-Turkel (JST) dissipation model damps out all high wavenumber resolved turbulence. Neither low Mach number preconditioning or an alternative rescaling the dissipation model allow the resolution of high wavenumber turbulence. In comparison, the matrix form of the JST model improves the high wave number resolution and (at this Mach number) both pre-conditioning and rescaling of the dissipation model allows the computation to capture the resolvable wave numbers. As Mach number increases the influence of the low Mach number pre-conditioning will reduce. For LES and DES the Central scheme should be used with the matrix dissipation model only. However, on very fine grids it appears that the scalar dissipation model can also yield acceptable resolved physics, however at the time of writing this is under investigation.

Note that the analysis file must contain either a single stdout file name, or a list of stdout file names that **MUST** be ordered in physical time. This means that a list of files

```
f0
f1
...
fn
```

are ordered such that each file corresponds to the physical time interval fn($t_{n-1} < t <= t_n$) such that $t_0 < t_1 < ... < t_n$ where n is the time step count. Since it is possible to use small time steps in a LES/DES calculation, the number of significant values set for output of the monitoring variables to stdout should be sufficiently large to ensure a smooth profile of the monitor variable against time. In other words if the time step is less than 1.0e-6, but only three decimal places are written in floating point format for the monitor variables then, if the monitored variable change linearly with time, a change will be observed after approximately 1000 iterations. Setting the number of significant places to a larger number (e.g. 8) will ensure a smooth profile without sudden jumps. If the parameter **Average variables (0/1)** is set to one, then an additional file is created with the string ".AVERAGE" appended to the **Output filename**. This contains the running mean of the monitored variables over the length of the data set contained in the list of files contained in **Analysis file**. There is little purpose in setting this option active until at least several tens of integral time scale have been computed. Since the aim of LES/DES computations is to obtain statistically converged solutions, the running mean can be used to estimate when convergence of first and second order statistical moments are obtained. Now, integral quantities such as lift and drag provide an indirect measure of the convergence of second-order quantities. A more direct comparison can be obtained by looking at the root mean square velocity and pressure monitor variables, or the skewness factor of the velocity field. It should be noted that the monitor variable "total-ke" has two distinct physical meanings, depending on whether **Compute flow statistics** has been set been activated (setting to either "Mean" or "Variance"). In this case the volume averaged turbulent kinetic energy is computed (by subtracting the mean velocity from the instantaneous velocity), otherwise only the total volume averaged kinetic energy is computed. For a flow with zero mean velocity both statistics will coincide. Similarly, all averaged mean and second-order statistical moments represent volume averaged quantities over the solution domain. This has been implemented to account for possible biasing effects in strongly anisotropic meshes.

9.8 Computing Means, Variances and other statistical quantities

From an engineering point we are usually interested in the mean levels of any particular flow variable. In the TAU code we can achieve this by computing running averages on selected flow variables. The parameter set that we need to consider are listed in table 7. The following table lists the valid entries for ¡string1¿, which is used to control the basic averaging functions

Table 7: Controls for averaging field variables

Keyword	Function	Averaged variables
mean	compute mean	pressure,density. velocity,viscosity
variance	compute correlations	velocity, pressure
meanvelgrad	compute mean	velocity gradients

of the TAU code. It should be noted that averaging can be performed in local time stepping, dual time stepping and global time stepping modes. The parameter string "string1" is set as "var1+var2+var3+.." where var1...var3 are the keywords given in table 7. So if we use the following parameter setting

```
Compute flow statistics: mea+meanvelgrad+des
```

we are computing running averaged of the mean of the three velocity components, density, pressure, and the laminar and turbulent eddy viscosities. We will now sound a warning. If you wish to compute the variances (by setting the string "variance") you must first have a suitably converged mean field. The best way to do this is to compare the mean fields from two intermediate solutions and check, using a post-processing tool such as Tecplot, that the contour levels coincide over the region of interest. While an automatic tools can be implement to detect a well converged mean, the problem is that there are numerous constraints which must be satisfied for the tool to work well in a general sense, and until we have more experience there is little point in designing such a tool. In order that the averaging can be restarted properly, you must write the averaged variables to your restart file, This means that the control string for output of field variables must have the keywords in table 7 appended to the string. For example:

```
Field output values: mean_meanvelgrad_variance
```

Note that this can result in large files, and it is for this reason that the mean operation operated on two distinct sets of variables: mean + mean velocity gradients. The reason that we require a well converged mean before we start computing the variance can be seen in the equations that we use to compute the running means, We use the recurrence algorithms designed by Welford et al.[19] which aim to minimize the effects of accumulating errors through floating point arithmetic. The recurrent relations are written as

$$M_{1m} = x_0 \quad , \quad M_{km} = M_{km-1} + (x_k - M_{km-1}) \,, \text{and} \tag{41}$$

$$S_{1s} = 0 \quad , \quad S_{ks} = S_{ks-1} + (x_k - M_{ks-1})\,(x_k - M_{ks}) \tag{42}$$

$$\tag{43}$$

where M is the running mean, S is the running variance, and k is the index denoting the k^{th} sample, and km or ks are the running counts of the mean or variance operation counts. We keep running totals of the number of counts for each of the groups "mean", "variance" and so on. This means we can halt statistical evaluation on a particular group and recommence again after several restarts without damaging the statistics for the "restarted" group. In order to obtain a computationally effective algorithm it is essential that the error in M is small otherwise the estimate of S is polluted resulting in the requirement for an excessive number of iterations before a satisfactory variance is returned. In order to assist the user in determining if the solution has a suitably converged mean value, the following feature is now provided in the stdout file between the transition and APSIM parameter control outputs.

```
Transition module parameters:
                        Transition module description file: para
                      Set transition at solver start (0/1): 0
                             Transition prescription (0/1): 0
                               Transition prediction (0/1): 0
read variables, file: <SST.SARC.F0.4.pval.10>
Sample: mean-rho  length  0: Mean    0.0000e+00
Sample: mean-u  length  0: Mean    0.0000e+00
Sample: mean-v  length  0: Mean    0.0000e+00
Sample: mean-w  length  0: Mean    0.0000e+00
Sample: mean-p  length  0: Mean    0.0000e+00
Sample: mean-muet  length  0: Mean    0.0000e+00
Sample: mean-muel  length  0: Mean    0.0000e+00

Read APSIM parameter ...
             APSIM parameter ----------------------------
```

We see the volume averaged means of the first and second order statistical moments over the entire fields. Here the values are zero because we have have just started to average. These values can be tracked and compared after restarts to give a guideline as to when a converged mean field is obtained, after which visual post-processing verification as suggested above should be made. We have one additional major control, which is the parameter

```
Reinitialize flow averaging: <vector1>
```

The control vector takes an integer value 0/1 for each of the keywords listed in table 7. A 0 value indicates that a standard restart is made, reinitializing the variable in memory from the restart solution if available. The value 1 indicated the the keyword is not initialized and that a restart from a zero value is made. For example if you have been computing means and variances for some time, but you note that the mean field is not statistically stationary then you can set the parameter as

```
Reinitialize flow averaging: 0 1 0
```

which we reset all computed variances to zero and start recalculating again from a decent mean field. Note that if the keyword variable set do not exist in the restart file, the averaging will assume a zero initial value.

```
Sample: mean-rho  length  20: Mean    2.4822e-01
Sample: mean-u  length  20: Mean    2.4276e+02
Sample: mean-v  length  20: Mean   -3.5032e-15
Sample: mean-w  length  20: Mean   -7.3229e-03
Sample: mean-p  length  20: Mean    1.9490e+04
Sample: mean-muet  length  20: Mean    1.0502e-07
Sample: mean-muel  length  20: Mean    1.7124e-05
Sample: resolved-uu  length  20: Mean    9.8421e+01
Sample: resolved-vv  length  20: Mean    1.8134e-25
```

```
Sample: resolved-ww  length  20: Mean    3.0839e+00
Sample: resolved-uv  length  20: Mean    2.9356e-15
Sample: resolved-uw  length  20: Mean    1.8475e-01
Sample: resolved-vw  length  20: Mean   -1.3406e-15
Sample: resolved-pp  length  20: Mean    4.7563e+05
Sample: resolved-uu  length  20: Correlation corrected
Sample: resolved-vv  length  20: Correlation corrected
Sample: resolved-ww  length  20: Correlation corrected
Sample: resolved-uv  length  20: Correlation corrected
Sample: resolved-uw  length  20: Correlation corrected
Sample: resolved-vw  length  20: Correlation corrected
Sample: resolved-pp  length  20: Correlation corrected
```

Note that the string "Correlation corrected" means that the variance have been corrected by premultiplying by the number of iterations so that the correctly scaled values of S are used in equation 42. After restarting again with the parameter "Reinitialize flow averaging" set to 0 1 0, the following is obtained:

```
Sample: mean-rho  length  30: Mean    2.4755e-01
Sample: mean-u  length  30: Mean    2.4146e+02
Sample: mean-v  length  30: Mean   -3.4367e-15
Sample: mean-w  length  30: Mean   -8.0115e-03
Sample: mean-p  length  30: Mean    1.9452e+04
Sample: mean-muet  length  30: Mean    1.1329e-07
Sample: mean-muel  length  30: Mean    1.7132e-05
Sample: resolved-uu  length  0: Mean    0.0000e+00
Sample: resolved-vv  length  0: Mean    0.0000e+00
Sample: resolved-ww  length  0: Mean    0.0000e+00
Sample: resolved-uv  length  0: Mean    0.0000e+00
Sample: resolved-uw  length  0: Mean    0.0000e+00
Sample: resolved-vw  length  0: Mean    0.0000e+00
Sample: resolved-pp  length  0: Mean    0.0000e+00
Sample: resolved-uu  length  0: Correlation corrected
Sample: resolved-vv  length  0: Correlation corrected
Sample: resolved-ww  length  0: Correlation corrected
Sample: resolved-uv  length  0: Correlation corrected
Sample: resolved-uw  length  0: Correlation corrected
Sample: resolved-vw  length  0: Correlation corrected
Sample: resolved-pp  length  0: Correlation corrected
```

Note that means have changed since they were not statistically converged, and that the variables have been reset to zero. For this example we need to iterate more in order to obtain a converged mean - after which we can restart the variance calculations. It should be pointed out, and we will not detail the theory here, that the time to obtain a converged statistic is a function of the variance of the statistic. As variance increases so does the number of iterations required to obtain a converged mean statistic. In practice this means that the convergence rate to a stationary mean is dependent on position in the flow and the variable

in question. On this basis we make the claim that error norms presented in terms of a norm of differences across mean statistics and some reference make little sense. The user should endeavor to provide component-wise convergence criterion when stating that stationary first moments have been obtained.

10 Particle Tracer

10.1 Introduction

The main motivation for introducing a particle tracer into the TAU-Code was the wealth of information that such a tool could provide: The implementation allows to determine trajectories of particles with mass as well as the trajectories of massless particles (= streamlines) in a given flow field. Therefore, it will not only permit comparison between experimental and numerical data but also serve as a powerful tool for computer visualization, specifically in the interpretation of geometrical and flow field data and the identification of important flow features.

Applications range from bloodstream and aerosol tracking, water impingement on aircraft components, simulation of helium bubble flow visualization and the eruption of volcanos which suspend tons of particulate materials into the atmosphere. The main focus of this particle tracer lays on the visualization of particle trajectories through a flow field of an aircraft and the determination of water impingement rates which may be used subsequently in an ice accretion analysis.

10.2 Description of particle tracer

Assumptions For using the particle tracker, it is very important to be aware of the assumptions made while deriving the governing equations solved by the particle tracker. Although these assumptions are not uncommon, it is worth noting:

- The particle surface is assumed to be spherical. This assumption is only valid for sufficiently small particles and only if the relative speed between particle and surrounding air is not too large.
- Gravity and aerodynamic drag are the only forces affecting the particle motion. Other forces like the Basset force, the Saffman lift force or the effect of rarefaction on small droplets have been neglected because they are not important for the particles sizes considered.
- The flow field is not affected by the particle motion. This assumption is only true as long as the particle concentration is sufficiently small.

Governing equations In the following, the index p means particle and index f means surrounding fluid.
The governing equation of the particle motion in a given flow field is Newton's second law of motion,

$$m_p \frac{\partial \vec{v}_p}{\partial t} = \sum \vec{Forces}$$

The term *Forces* can be

$$Forces = Aerodynamic\ drag + Buoyancy + Gravity + ...$$

where at this stage the tracker neglects all terms except aerodynamic forces and forces resulting from the presence of gravity.
Therefore, the ordinary differential equation (ODE) becomes

$$m_p \frac{\partial \vec{v}_p}{\partial t} = \frac{1}{2}\rho_f C_D A_p \mid \vec{v}_{rel} \mid (\vec{v}_{rel}) - \rho_f V_p \vec{g} + m_p \vec{g}$$

where A_p is the cross section, V_p the volume, m_p the mass of the particle and ρ_f the density of the fluid. $\vec{g}$ is the gravity vector. $\vec{v}_{rel} = \vec{v}_f - \vec{v}_p$ is the relative velocity between fluid and particle. C_D is the drag coefficient of the particle. By rearranging, this equation can be expressed as

$$\frac{\partial \vec{v}_p}{\partial t} = -\frac{C_D Re_p}{24}\frac{18}{d_p^2}\frac{\mu_f}{\rho_p}(\vec{v}_p - \vec{v}_f) + \frac{\rho_p - \rho_f}{\rho_p}\vec{g}. \tag{44}$$

The diameter of the particle is d_p. Re_p is the particle Reynolds number defined by

$$Re_p = \frac{\mid v_p - v_f \mid d_p \rho_f}{\mu_f}.$$

It may be noted that it is possible to deduce one important time scale from the ODE. The so-called velocity response time τ_p defined by

$$\tau_p = \frac{1}{18}\frac{d_p^2 \rho_f}{\mu_f}.$$

It is a measure for the time interval that a particle needs to reach the fluid velocity of the surrounding fluid.
In equation (44) the drag coefficient is generally not known. Only for Stokes flow conditions, i.e. if $Re_p \ll 1$, can the Stokes relation can be used to determine C_D

$$C_D = \frac{24}{Re_p}.$$

Since much larger particle Reynolds numbers can occur during calculation of trajectories, an approximation to the experimental results of Langmuir and Blodgett[1], which can be found nowadays in the ADS-4[2], has been used. It should be noted that the approach used in this particle tracker is not the original analytical relationship proposed by Langmuir and Blodgett since it is very computationally expensive. Instead, a piecewise approximation to the experimental data is employed.

[1] I. Langmuir and K.B. Blodgett: A Mathematical Investigation of Water Droplet Trajectories. AAF Technical Report 5418, 19. February 1946

[2] D.T. Bowden, A.E. Gensemer and C.A. Skeen: Engineering Summary of Airframe Icing Technical Data. Technical Report ADS-4, FAA, Washington, 1963

Dimensionless quantities By non-dimensionalizing equation (44) several well known non-dimensional quantities appear in the equation. These are

- *Particle Reynolds number:* Is a measure for the ratio of inertial forces to viscous forces

$$Re_p = \frac{\mid v_p - v_f \mid d_p \rho_f}{\mu_f}$$

  $v_p - v_f$ is the velocity (m/s) between the particle and fluid, d_p is the particle diameter (m), ρ_f is the fluid density (kg/m^3) and μ_f is the kinematic fluid viscosity (kg/ms).

- *Froude number:* Defines the ratio between inertial and gravity forces. If the Froude number gets close to one the gravitational forces can not be neglected anymore.

$$Fr = \frac{v_{ref}}{\sqrt{g L_{ref}}}$$

  v_{ref} is the reference fluid velocity (m/s), g is the gravity constant (m/s^2) and L_{ref} is the corresponding reference length (m).

- *Inertia parameter:* This parameter is sometimes referenced as Langmuir parameter

$$K = \frac{1}{\tau_p} \frac{v_{ref}}{L_{ref}} = 18 \frac{\mu_f}{d_p^2 \rho_f} \frac{v_{ref}}{L_{ref}}.$$

  μ_f is the kinematic fluid viscosity (kg/ms) and ρ_f is the fluid density (kg/m^3). L_{ref} is the reference length (m) and v_{ref} is the reference fluid velocity (m/s)

Errors In assessing the accuracy of computed trajectories, users of the particle tracer should keep in mind different sources of errors that can reduce the accuracy of the results. Firstly, the accuracy depends on errors made during the computation of the flow solution. Furthermore, errors can be introduced in the particle tracker. The sources and types of errors within a particle trajectory are:

- *Errors in nodal velocities* These errors appear while computing the flow solution. The nodal velocities used for interpolation, are affected by numerical errors of the flow solver. Make sure to use a sufficiently refined grid and a well converged flow solutions.

- *Spatial interpolation (mesh errors)* Even if the nodal velocities are exact, the spatial interpolation within a volume element to the particle positions introduces further errors and may affect the results of the particle tracker.

- *Error during the numerical integration of ODE governing the particle motion (error of the integrator)* A numerical integration technique used for integrating an ODE introduces errors due to truncation and round-off. Although this particle tracker employs embedded Runge-Kutta integrators which try to minimize the relative truncation error, the global truncation error grows proportionally to the number of steps used.

10.3 Determination of local catch efficiency

The local catch efficiency β is defined as the flux of material through a plane normal to the undisturbed oncoming flow over the flux of material that impinges on a surface. As the flux of material can be thought of as being delimited by a stream tube that is extending from the undisturbed flow until it impinges on the surface, β corresponds to the cross section of this imaginary stream tube in the oncoming flow over the area on the surface that is wetted by the stream tube.
For calculating the local catch efficiency, the stream tube has usually a rectangular shape and is delimited by four particle trajectories. Thus for a three-dimensional flow, the catch efficiency β is calculated by the ratio of the area spanned by the four particles in the release plane A_0 and the area A_m spanned by the corresponding particle impingement points, see right Figure 11.

$$\beta = \frac{A_0}{A_m}.$$

For the case of a purely two-dimensional flow this expression can be simplified to, see left Figure 11.

$$\beta = \frac{\Delta y}{\Delta s}.$$

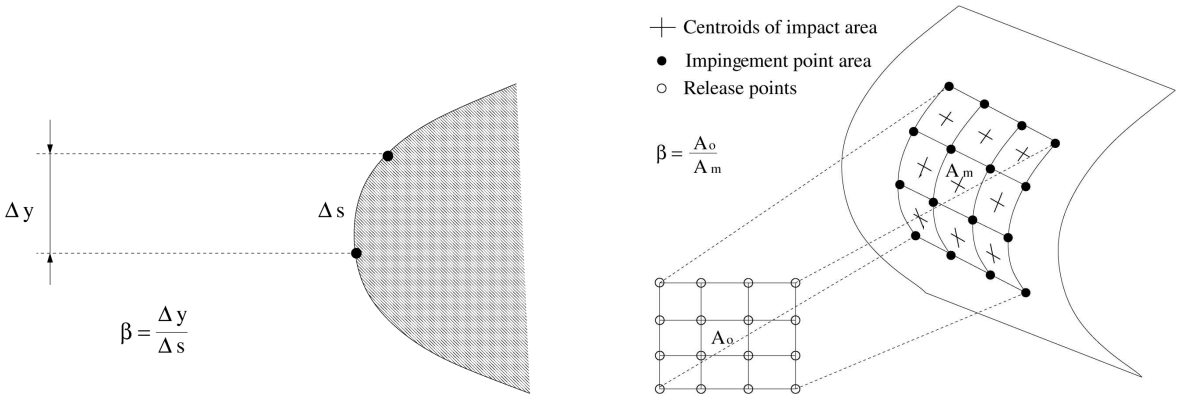


Figure 11: Determination of β for 2D and 3D flow.

Figure 12 shows the catch efficiency calculated for an ONERA M6 wing. As usual, β falls to zero at some distance downstream of the stagnation line on the upper and lower surface of the wing because behind these lines (the so-called impingement limits) no further droplets impinge. Additionally, a peak in the β distribution is visible near the stagnation line of the airfoil.

10.4 Determination of the impingement limits

Impingement limits are those geometrical locations where the last trajectories impinges in streamwise direction. After determining the impingement locations of a large number of

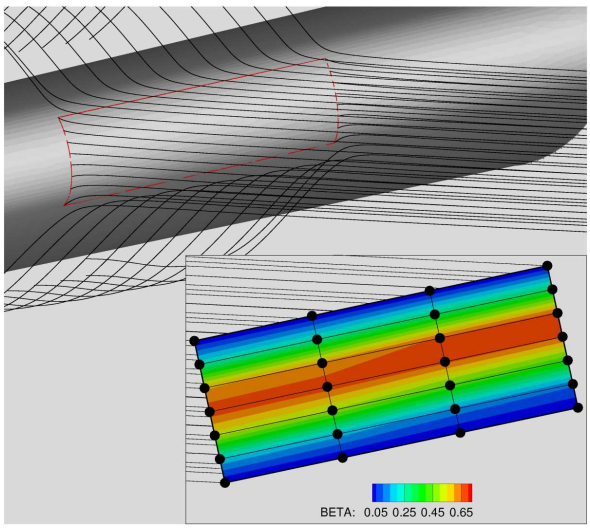


Figure 12: β distribution - testcase ONERA M6 wing.

particle trajectories, an approximation to the impingement limits can be constructed by determining the hull around all impingement points. The visualization of a line constructed in this way corresponding to approximated impingement limits can provide a good indication of the areas that are exposed to particle impingement. Figure 13 shows an example of impingement limits determined by the procedure explained above.

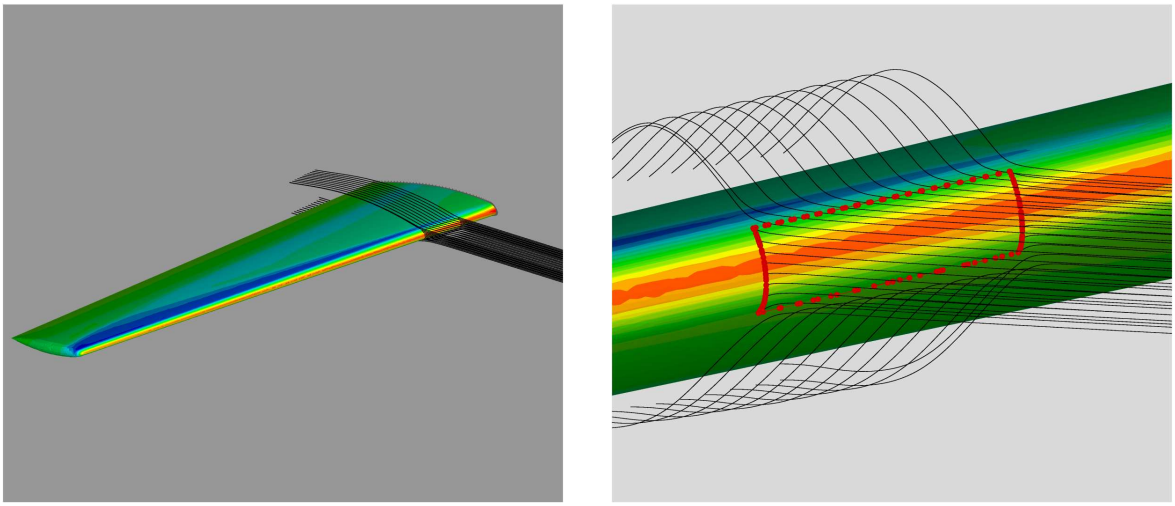


Figure 13: Impingement limits on an ONERA M6 wing. The wing is contoured with a C_p distribution and the impingement line is seen as a red dotted line in the close-up view.

A close-up view of the impingement limits depicted in Figure 13 reveals that the line approx-

imating the impingement limits is defined by a larger number of points than the number of impingement points delimiting the impingement area. This feature is due to the fact that often times surface element edges are situated between particle impingement points. This requires cutting of the surface elements and introducing additional auxiliary points on the element edges, especially if the distance between particle impingement points increases. Figure 14 depicts an example where the approach of cutting the surface triangles between two particle impingement points (black dots) is well visible.

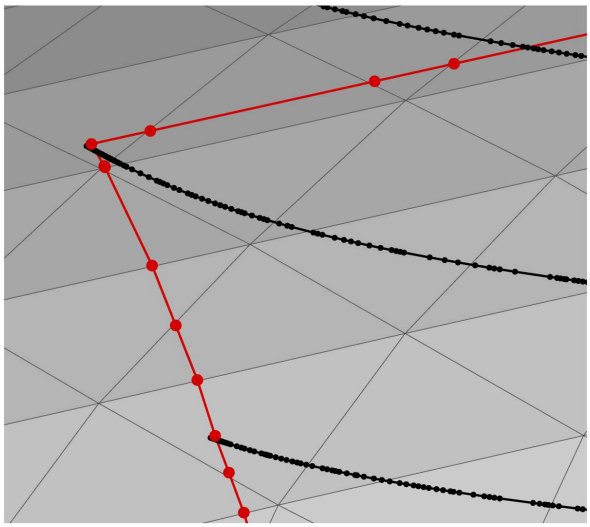


Figure 14: Upper left corner of the impingement line (red) seen with the surface triangulation, black lines are the particle trajectory.

10.5 Refinement and extension of initial particle starting grid

The level of catch efficiency detail or the resolution of impingement on thin structures can be greatly enhanced by using a refinement algorithm for the particle starting points. Controlled by the two parameters `Maximum change of distribution` and `Maximum distance` the refinement algorithm will introduce extra starting points between the initial particle starting points. Additionally, the algorithm automatically introduces particle starting points whenever the four trajectories delimiting a stream tube impinge only partially.
The refinement algorithm implemented is based on a Cartesian start grid. The refinement of the start squares is handled by a quadtree data structure. The two above-mentioned parameters can be used to obtain a smoother catch efficiency distribution. The enhanced resolution of the impingement limits will produce a better defined limit line as particle impingement points will vary significantly for almost tangent trajectories.
In the left picture of Figure 15 a well-defined, simple quadratic starting grid is depicted in front of a wing. The catch efficiency computed is relatively coarse and the borders of

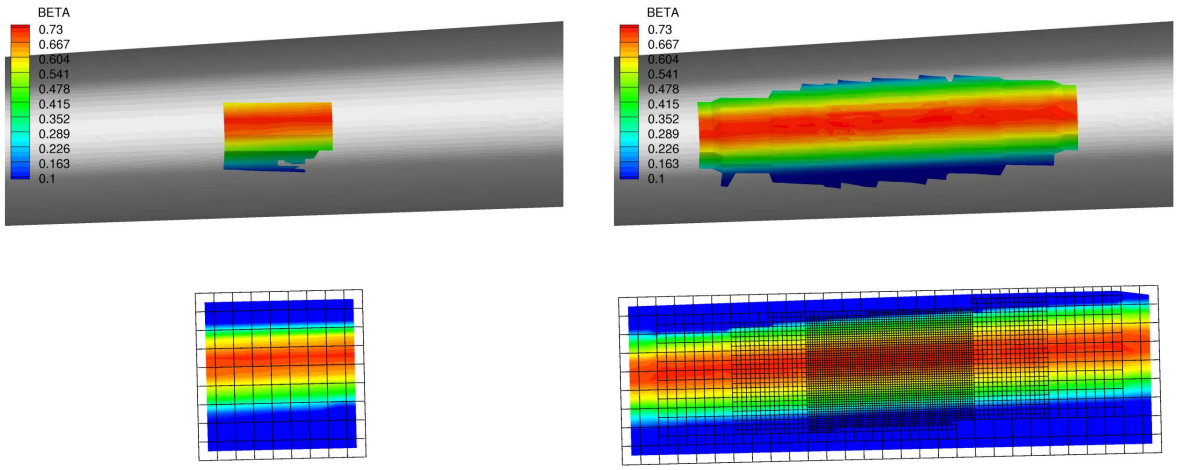


Figure 15: Improvement of catch efficiency detail and impingement limit line resolution due to refinement and extension of the initial starting grid (right) in comparison to the initial starting grid (left).

impingement are patchy. The right picture of Figure 15 shows, especially at the impingement borders, a much smoother catch efficiency distribution.
Extension of the starting grid is another automation introduced. As seen in the right figure of Figure 15 the grid extends in lateral direction as long as particles started on the rim of the starting grid are impinging. The combination of the refinement and extension feature allows users to specify a very coarse start grid where the code will automatically determine the right resolution and extension of the grid as long as at least a single impinging trajectory is computed on the initial start grid. Thus, the number of restarts may reduce.

10.6 Intersection planes for particle trajectories

Intersection planes provide access to the evolving particle concentration ratio in the flow before particles impinge on a wall. Whenever a plane is specified between the starting and the impingement point of a particles, the intersection point of a trajectory with this plane is save and used to determined the particle concentration ratio in this flow at the location of this plane. The resulting concentration ratio distribution will be written to file.
The left picture in Figure 16 shows the particle concentration ratio just before impingement on the lip of a nacelle. In the middle of the plane, the scoop effect of the nacelle is well visible as the catch efficiency becomes greater than one which means that the particle trajectories are closer to each other than they were in the starting plane. Far away from the nacelle the trajectories are almost undisturbed and the particle concentration ratio approaches one. The right picture in Figure 16 shows the particle concentration ratio just prior to trajectories impacting on the spinner of the fan engine.

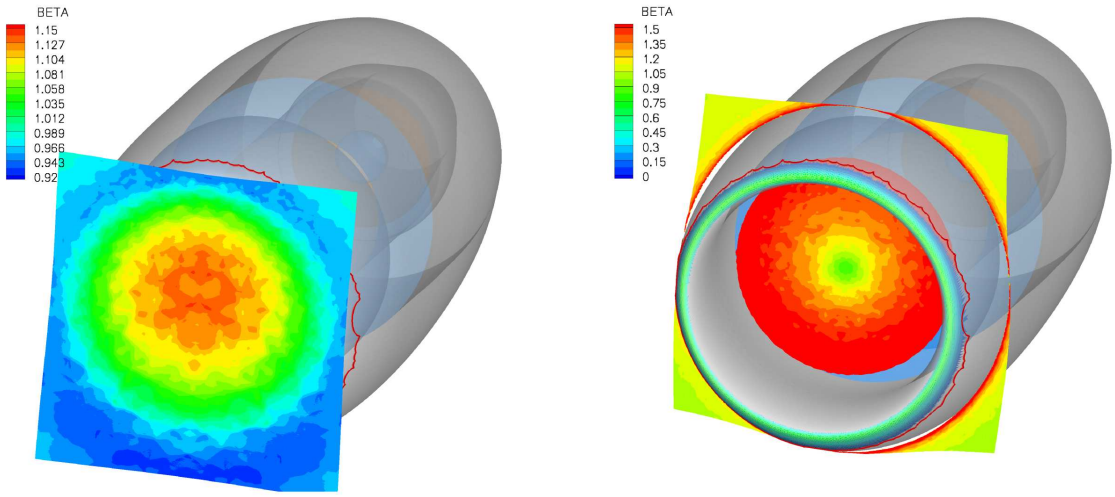


Figure 16: Particle concentrations determined in two different intersection planes in front of and in the intake of a fan engine.

The usage of the particle program is

`bin-path/ptau3d.particle parameter-file`

Optionally, a log-file can be defined as third argument.

10.7 Safety features

The trajectory output and monitoring for each timestep can create a large amount of data. If someone imagines the tracking of thousands of particles, especially the monitoring file, which is printed in ASCII format to stdout, can blow up to a non-editable file. To restrict the possible mass of output the particle tracker sums up the number of total particles trajectories on the initial grid. Whenever the sum is greater than 1000 and the parameter `Monitor particle (0/1/2)` is set to 2 it will be reset internally to a value of 1 (monitoring of the start and end state of each particle). Additionally, if the parameter `Save trajectory data (0/1)` is set to 1 it will be reset to a value 0. This measure has been taken to speed-up the particle tracker and to prevent files from becoming too large.

10.8 Prerequisites

The particle tracer can be started after the definition of a primary grid and the computation of a flow solution on this grid. A suitable setting of input parameters has to be defined in a

parameter file.

10.9 Input options

The input options are defined in the parameter file. All input parameters, input file names, output file names and output format can be defined in this file. The parameter names and the file names are explained below. They are listed in alphabetical order for the different sections (file-io, main, fluid/particle and tracer). Parameters without default values are printed underlined.

10.9.1 Input files

The major input file is the parameter file with the appropriate flow solution as a restart. The parameter file can contain all names of further input files as listed below.

10.9.2 Input general parameters

Geometry parameter section

Inviscid parameter section

Controls parameter section

References parameter section

`Allow grid extension N E S W (0/1):` array-int, default 0

- Allowing grid extension is optionally provided whenever impingement occurs at the border of the rectangular starting grid, Figure 20. If this parameter is on the extension will be performed in all four directions (North, East, South and West).

`Allow inner disk extension (0/1):` integer, default 0

- Considering a disc starting grid with an inner circle not zero this parameter can allow the extension up to the Disc center point (x,y,z).

`Allow line extension P1 P2 (0/1):` array-int, default 0

- Considering a line starting grid, Figure 21, with the two endpoints it can allow the extension of the starting grid outside the endpoints.

`Allow outer disk extension (0/1):` integer, default 0

- Considering a disc starting grid, Figure 18, with an outer circle this parameter can allow the extension of the starting grid in radial direction.

`Boundary marker id:` character string, default (none)

- This feature was introduced to choose all points corresponding to a boundary marker. The idea behind is the backward tracking form the desired boundary(s). It eases the definition the following set up of a starting line/quadrilateral. But note that backward tracking is only possible with the Type of tracking streamline due to the complex initial setting of the particles/boundary points.

`Catch plane (0/1):` integer, default 0

- This parameter stops the trajectory computation at the intersection plane if set to 1. The catch efficiency rate will then be mapped on this plane and printed out. This feature gives information about the behavior of the particles before impingement.

`Circumferential resolution:` integer, default 0

- The value of this parameter defines the amount of seeding starting particles in the circumferential direction, Figure 18. Equidistant spacing is supported and the value must be greater than one otherwise the disc starting grid is not affective.

`Consider gravity (0/1):` integer, default 0

- An observation of the Froude number (see Dimensionless numbers in Part 10.2) gives the answer of the affecting gravity. If the Froude number gets close to 1 it is advised to turn on the additional force acting on the particle. Whenever the Froude number is much greater than 1, no gravity has to be considered.

`Cube origin (x,y,z):` character string, default (none)

- A constrained cube is introduced to mark specific boundary points which lay inside the specified cube. This might be useful for generating starting points on exposed parts on a surface. The cube is set via the origin and three extra vector endpoints. All important features for a particle constrained cube definition can be seen in the following Figure 17.

`Density of second phase:` float, default 999

- Define the density of particles you want to trace through the flowfield. The default value of the particle density is set to water which is 999 kg/m^3.

`Disc center point (x,y,z):` character string, default (none)

- The circular starting grid, as shown in Figure 18, must be specified with the signed parameters in the Figure. The first one is the disc center point where the disc get placed in space. Together with this point and the Disc normal vector (x,y,z) a plane is spanned normal to the vector. The resolution of the seeding points is ordered with the Circumferential resolution and Radial resolution.

`Disc inner radius:` float, default 0

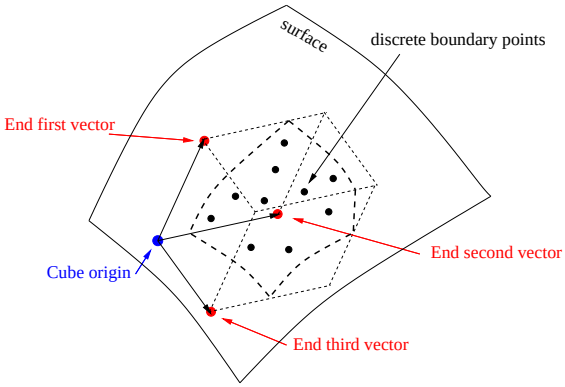


Figure 17: Specification for a constrained cube.

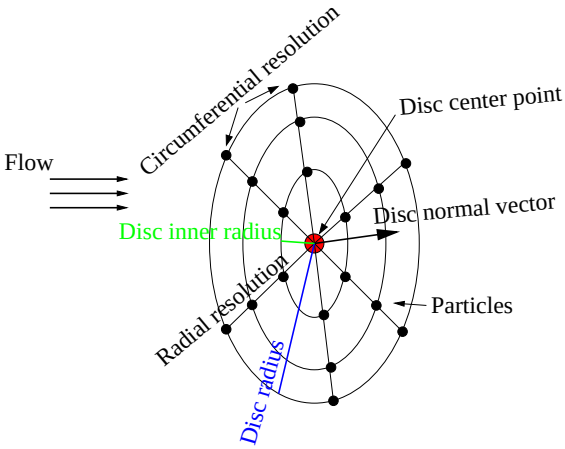


Figure 18: Specification for a particle disc.

- As seen in Figure 18, a Disc inner radius defines a torus area used for the seeding particle points. If at any reason the disc inner radius extends the Disc radius it is reseted automatically to zero.

`Disc normal vector (x,y,z):` character string, default (none)

- Together with the Disc center point (x,y,z) it spans a plane in space which holds the starting particle points. The resolution of the seeding particle points is defined with the two parameters Circumferential resolution and Radial resolution

`Disc radius:` float, default 0

- Input for the outer radius of the disc, see Figure 18. Whenever the outer disc radius exceeds the primary grid farfield and particle lay outside the physical domain, these points do not get considered for tracing

`End first vector (x,y,z):` character string, default (none)

- Specification of the first endpoint of a cube. See Figure 17

`End second vector (x,y,z):` character string, default (none)

- Specification of the second endpoint of a cube. See Figure 17.

`End third vector (x,y,z):` character string, default (none)

- Specification of the third endpoint of a cube. See Figure 17.

`First corner (x,y,z):` character string, default (none)

- A quadrilateral is specified by four endpoints which are set via the First corner (x,y,z), Second corner (x,y,z), Third corner (x,y,z) and the Fourth corner (x,y,z). All important features for a particle quadrilateral definition can be seen in the following Figure 19. Additionally a particle grid can be set up for computing the catch efficiency rate β,

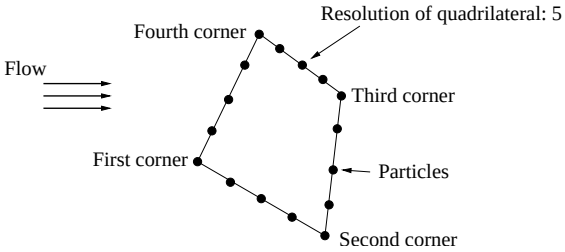


Figure 19: Specification for a particle quadrilateral.

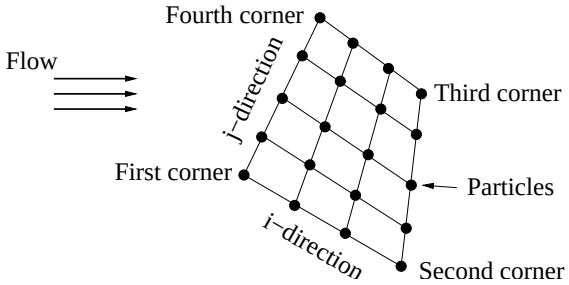


Figure 20: Specification for a particle grid.

see right Figure 19. The i-direction counts from the First corner (x,y,z) to the Second corner (x,y,z) and the j-direction counts from the Second corner (x,y,z) up to the Third corner (x,y,z).

`First point (x,y,z):` character string, default (none)

- A line is specified by two endpoints which are set via the First point (x,y,z) and the Second point (x,y,z). All important features for a particle line definition can be seen in Figure 21.

`Fourth corner (x,y,z):` character string, default (none)

- → 'First corner (x,y,z)'

`Gravity constant:` float, default 9.81

- The gravity constant, as it is known for the earth.

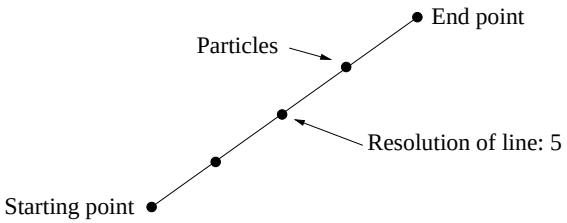


Figure 21: Specification for a particle line.

`Gravity direction:` array-float, default 0 0 1

- The gravity vector always points towards the middle of the earth. With an angle of attack of zero the vector becomes (0 0 1). Be aware if the angle of attack got changed the gravity vector has to be changed accordingly.

`Integration direction (forward/backward):` character string, default forward

- Forward tracking is only possible for a particle while forward/backward is allowed for a streamline, also refer to Type of tracking.

`Integrator epsilon:` float, default 1e-12

- The value handles the accuracy requirements for the Runge-Kutta timestepping. Additionally the timestep will depend on the value chosen for this parameter.

`Maximum change of distribution:` float, default 0

- To increase the effectiveness of the refinement and derefinement of particle starting points, this parameter specifies the maximum change of the catch efficiency rate distribution. This value should be used between 0 and 1. For example, the value of 0.2 indicates that refinement around a particle in a particle grid has to be performed if the surrounding catch efficiency rates differ more than 20 per cent. this parameter will be only used for refinement. Derefinement will not be affected.

`Maximum distance:` float, default -

- The parameter checks that the distance to cell centers of both neighboring cells does not exceed the threshold value.

`Maximum number of timesteps:` integer, default 10000

- The number of time steps specifies the Runge- Kutta stages performed for one particle.

`Maximum x trajectory coordinate:` float, default 999999

- To shorten the particle trajectory computation someone might use a x-coordinate where the time integrations stops.

`Monitor particle:` character string, default Start_End

- Turns on/off the print out of the Monitoring particle values in the stdout file as ASCII format. The setting *Off* produces no output at all, *Start_End* monitors the starting and endpoint of each particle and *Trajectory* prints the whole trajectory of a particle. Whenever *Trajectory* is switched on be sure to trace only a few particle because it fills up the stdout file very quickly to a high amount of data.

`Monitoring particle values:` character string, default 1

- Character string containing the names of the quantities to be monitored during execution of the particle tracer. The names are separated by underline, e.g. *dt_vx_vy_vz*. The available names with a short explanation are listed by the particle tracer in the standard output and are listed in Table 5

`Number of refinement levels:` integer, default 20

- Choose the maximum level of refinements with this parameter. The value of 10 means 10 refinements of the initially specified grid will be made.

`Output format:` character string, default tecplot

- In analysis both "ascii" and "tecplot" options are available at present.
- In tau2plt: Set the conversion format which can be the following macros
  - *ensight6* $\cdots$ use for ENSIGHT version 6
  - *ensight_gold* $\cdots$ use for ENSIGHT GOLD
  - *fieldview* $\cdots$ use for FIELDVIEW (binary only), NOTE: There are no surface data files allowed
  - *opendx* $\cdots$ use for OPENDX
  - *tecplot6* $\cdots$ use for TECPLOT version 6
  - *tecplot* $\cdots$ use for TECPLOT 7 or higher (default)
- particle tracer: The tracer can handle the following output formats
  - *tecplot* $\cdots$ use for TECPLOT 7 or higher (default)
  - *opendx* $\cdots$ use for OPENDX
  - *vtk* $\cdots$ use for Visualization tool kit

`Particle diameter (micron):` character string, default 1

- The dimension of the particle diameter is 10^{-6} meters, so called microns. Whenever desired, more than one particle diameter can be specified, e.g. 10,20,30.3,43.567. For the difference in particle diameters the output filename will contain the particle diameter, e.g. flow solution prefix + d20 + flow solution suffix.

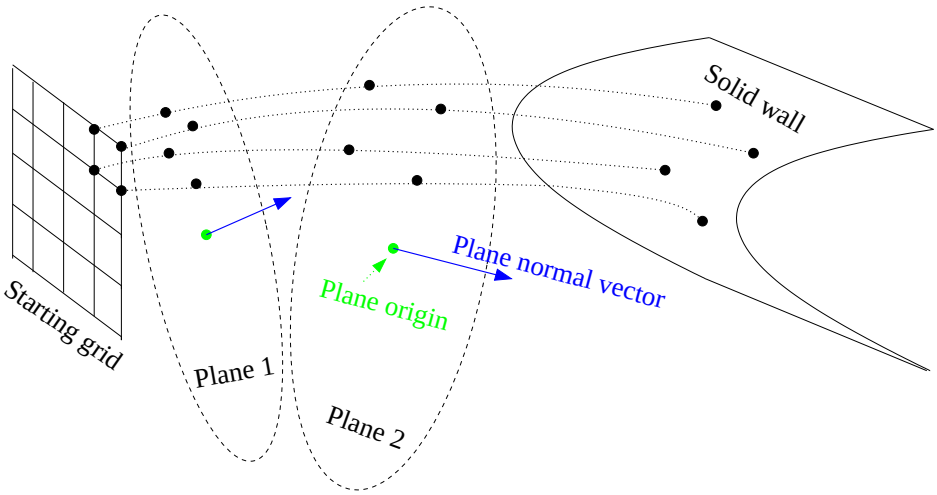


Figure 22: Specification of plane between starting grid and solid wall

`Plane normal vector (x,y,z):` character string, default (none)

- The plane normal vector defines the planes orientation in space, see Figure 22. . The particle trajectories which intersect this planes will be marked and the catch efficiency rate at this plane will be printed out.

`Plane origin (x,y,z):` character string, default (none)

- The origin of the intersection plane should be set between particle starting grid and solid wall, see Figure 22. In case of any intersecting of plane and solid wall the plane will not consider the particles which already reached the wall. The result will be a partial mapping of the catch efficiency information at the plane

`Points in i-direction:` integer, default 2

- To set up a structured grid for a catch efficiency rate β computation, the points in the i-direction and Points in j-direction have to be specified. The maximum value for i and j are 100.

`Points in j-direction:` integer, default 2

- To set up a structured grid for a catch efficiency rate β computation, the points in the j-direction and Points in i-direction have to be specified. The maximum value for i and j are 100.

`Radial resolution:` integer, default 0

- The value of this parameter defines the amount of seeding starting particles in the radial direction, Figure 18. Equidistant spacing is supported and the value must be greater than one otherwise the disc starting grid is not affective.

`Resolution of line:` integer, default 2

- This parameter specifies the number of particle for a single line. If this number is not set only the two endpoint are tracked. For better understanding refer to Figure 21.

`Resolution of quadrilateral:` integer, default 2

- This parameter specifies the number of particles for a quadrilateral. The resolution number puts the specified particles equidistantly spaced on each side of the quadrilateral. If this number is not set only the four endpoints are tracked. For better understanding refer to Figure 19.

`Save beta data (0/1):` integer, default 1

- The particle tracer computes in addition the catch efficiency rate β. The β distribution shows the particle/droplet impingement over a hard surface. The output is directly converted into the desired Output format.

`Save trajectory data (0/1):` integer, default 1

- Turns on/off the print out of the complete particle trajectory into a file. The file is printed in the desired Output format. The complete name of the file is composed of the prefix of the flow solution file, the particle diameter chosen and the appropriate format suffix.

`Save trajectory summary (0/1):` integer, default 1

- Turns on/off the outpout of additional trajectory data which are the following:

  impinge flag Is an integer value which shows particle impinged at the surface (flag=1) and non impinged ones (flag=0).

  boundary map Includes the bounary marker were the particular particle has impinged. Otherwise the value will be -2 which declares to be outside.

  time end Time of the particle to travel from release point until may encountering impingement or exit through farfield.

  v normal to wall Normal velocity magnitude at the solid surface. Otherwise the velocity will be marked with -1.0.

  impingement angle Angle measured at impingement from trajectory to surface normal. Otherwise marked with -999.

  xp end x-coordinate of impinged particle at the solid surface.

  yp end y-coordinate of impinged particle at the solid surface.

  zp end z-coordinate of impinged particle at the solid surface.

  vx start x-velocity component at the starting point of the particle trajectory. NOTE: All particles will have a release point and therefore a velocity unequal zero.

vy start y-velocity component at the starting point of the particle trajectory.

vz start z-velocity component at the starting point of the particle trajectory.

vx end x-velocity component at the impingement point of the particle trajectory. NOTE: All particles will have an end point which can be either a solid wall, farfield or when time integration ends.

vy end y-velocity component at the impingement point of the particle trajectory.

vz end z-velocity component at the impingement point of the particle trajectory.

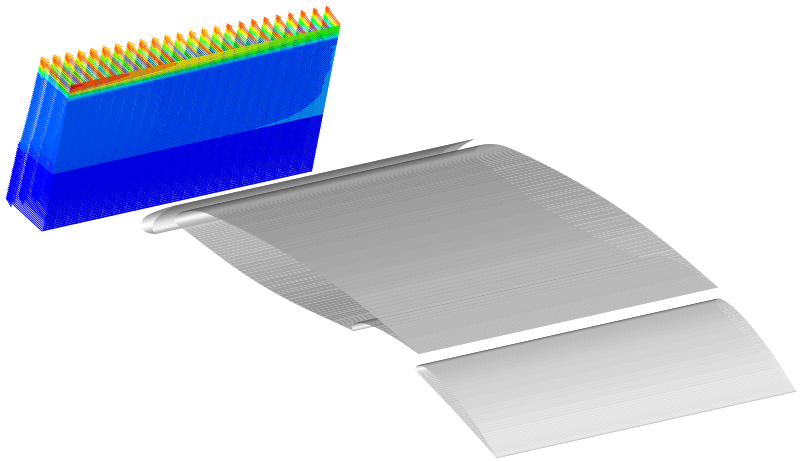


Figure 23: Trajectory summary output. Visualized is the starting grid with contours of the normal velocity magnitude on the impinged surface, further is shown the particle velocity vector of the impingement colored with the impingement angle. NOTE: The relations between starting grid and aerofoil are arbitrary for better visualization.

All vectors, e.g. velocities and impingement coordinates, are displayed at the starting grid.

`Search epsilon:` float, default 1e-12

- This value controls the accuracy of locating a particle in each element. For detecting a particle in an element, shape functions are used. Except for triangles and tetrahedrons these shape functions have to be computed with an iterative process. The terminating condition of such an iteration procedure is controlled via the search epsilon.

`Second corner (x,y,z):` character string, default (none)

- → 'First corner (x,y,z)'

`Second point (x,y,z):` character string, default (none)

- → 'First point (x,y,z)'

`Start grid type:` character string, default (none)

- The Start grid type defines the release structure of seeding points for the particle tracking. The following keywords are available:
  - Point
  - Line
  - Plane
  - Marker
  - Cube constrained
  - Frame
  - Grid
  - Disc

  where *Frame*, *Grid* and *Disc* are only valid for threedimensional applications. These keywords are used for the block parameters (trajectory end) to specify additional information about the distribution of the seeding particles in the flow field.

`Starting point of particle (x,y,z):` character string, default (none)

- At this parameter a single starting point for a particle can be specified.

`Third corner (x,y,z):` character string, default (none)

- → 'First corner (x,y,z)'

`Trajectory ODE integrator:` character string, default Dormand_Prince

- – *Cash_Karp* embedded five stage fourth order (5(4)) Runge-Kutta integrator
  - *Dormand_Prince_54* embedded five stage fourth order (5(4)) Runge-Kutta integrator
  - *Dormand_Prince_86* embedded eight stage sixth order (8(6)) Runge-Kutta integrator
  - *Bogacki_Shampine* three stage Runge-Kutta integrator

`Type of tracking:` character string, default particle

- There are two different types to choose. First is the ordinary *particle* tracking. A particle can be tracked only forward. The second type is the *streamline* which can be tracked forward and backward.

Table 8: Available monitoring variable keywords and their explanation.

Keyword	Comment
“dt”	physical timestep
“x”	x coordinate of particle
“y”	y coordinate of particle
“z”	z coordinate of particle
“vx”	x velocity of particle
“vy”	y velocity of particle
“vz”	z velocity of particle
“elem”	particle is in element (id)
“p”	number of present particle
“R-time”	Total CPU time (real)
“U-time”	Total CPU time (user)
“Norm-time”	Normalized CPU time (real)

10.10 Sample parameter file

The parameter file could look like this:

```
-----------------------------------------------------
required parameters
-----------------------------------------------------
  Files/IO ---------------------------------------: -
                         Boundary mapping filename: (thisfile)
                             Primary grid filename: tau.grid
-----------------------------------------------------
io
-----------------------------------------------------
  Grid/Solution ----------------------------------: -
                               Restart-data prefix: tau.solution
-----------------------------------------------------
solver
-----------------------------------------------------
  References -------------------------------------: -
                            Reference Mach number: not_defined
                                  Reynolds number: not_defined
  Geometry ---------------------------------------: -
                                       Grid scale: not_defined
  Controls ---------------------------------------: -
```

```
                    Monitoring significant figures: 4_4_4_4
-----------------------------------------------------
particle
-----------------------------------------------------
  Files/IO ---------------------------------------: -
                             Save beta data (0/1): 1
                        Save trajectory data (0/1): 1
                     Save trajectory summary (0/1): 0
  Tracking ---------------------------------------: -
                           Consider gravity (0/1): 0
                      Maximum number of timesteps: 10000
                  Maximum x trajectory coordinate: 10.2
  Refinement/Extension ---------------------------: -
                   Maximum change of distribution: 0.2
                                  Maximum distance: 0.5
                      Number of refinement levels: 15
  Intersection plane -----------------------------: -
                      Plane normal vector (x,y,z): 1  0  0
  Controls ---------------------------------------: -
                                Monitor particle: Start_End
                       Monitoring particle values: dt_x_y_z
  Fluid/Particle ---------------------------------: -
                       Particle diameter (micron): 20  30  40
-----------------------------------------------------
tau2plt
-----------------------------------------------------
  Output Control ---------------------------------: -
                                   Output format: tecplot

    Single trajectory ----------------------------: -
                                 Start grid type: Point
              Starting point of particle (x,y,z): -90,-0.5,-8.5
  trajectory end
  --------------
     Bounding line -------------------------------: -
                                 Start grid type: Line
                              First point (x,y,z): 3,2,4
                             Second point (x,y,z): 3,3,4
                               Resolution of line: 2
                Allow line extension (P1 P2: 0/1): 0
  trajectory end
```

```
--------------
   Bounding quadrilateral ----------------------: -
                                Start grid type: Grid
                             First corner (x,y,z): 1,2,3
                            Second corner (x,y,z): 1,-6,-2
                             Third corner (x,y,z): 1,-6,3
                            Fourth corner (x,y,z): 1,2,-2
                            Points in i-direction: 10
                            Points in j-direction: 10
             Allow grid extension (N E S W: 0/1): 0
trajectory end
--------------
   Bounding disc -------------------------------: -
                                Start grid type: Disc
                      Disc center point (x,y,z): 1,3,5
                     Disc normal vector (x,y,z): -0.5,2,0.8
                                    Disc radius: 10
                              Disc inner radius: 1
                              Radial resolution: 20
                     Circumferential resolution: 40
                Allow outer disk extension (0/1): 0
                Allow inner disk extension (0/1): 0
trajectory end
--------------
   Constrained cube  ---------------------------: -
                                Start grid type: Cube constrained
                           Cube origin (x,y,z): 0,0,0
                         End first vector (x,y,z): 1,2,4
                        End second vector (x,y,z): 5,3,7
                         End third vector (x,y,z): 7,4,3
trajectory end
--------------
   Boundary marker -----------------------------: -
                                Start grid type: Marker
                              Boundary marker id: 2
trajectory end
--------------
 Cut plane -----------------------------------: -
                                Start grid type: Plane
                            Plane origin (x,y,z): 0.0,1.5,0
                     Plane normal vector (x,y,z): 1.0,0.0,0.0
```

```
                                Catch plane (0/1): 0
  trajectory end
  --------------
```

10.11 Description of process information

The output information of the particle program starts with a line indicating the version and the compile time of the code. Then a list of the parameters used is printed. This is followed by a list of the used flow and particle reference values.

Next detailed information about the computed particle trajectories may be printed the granularity of which is dependent on the setting of the parameter `Monitor particle (0/1/2)`. This parameter can be set either to 0, 1, or 2. Setting this parameter to zero means no information will be printed at all. Setting it to one, the starting and end point information will be printed. Setting this parameter equal to two will print out the complete particle trajectory. But beware, setting the parameter to two should be used with care and only for a small number of trajectories. Otherwise, the log file fills up quickly.

Extra output will be provided if particles reached a wall. The catch efficiency rate is computed automatically after the particle tracking.

An example output file for setting the 'Monitor particle' parameter to 1 could look like this:

```
DLR Tau-Code, particle, Version 10/31/2004 compiled on Mar  9 2005
------------------------------------------------------
Implemented ODE integrators:
  Cash_Karp
  Dormand_Prince_54
  Dormand_Prince_86
  Bogacki_Shampine

Implemented tracking types:
  streamline
  particle

Implemented monitoring values (to be separated by '_'):
    KEY        COMMENT
  dt         physical timestep
  x          x coordinate of particle
  y          y coordinate of particle
  z          z coordinate of particle
  vx         x_velocity of particle
  vy         y_velocity of particle
```

```
  vz          z_velocity of particle
  elem        particle is in element (id)
  p           number of present particle
  R-time      Total CPU time (real)
  U-time      Total CPU time (user)
  Rhs-time    CPU time for one compute_RHS
  Norm-time   Normalized CPU time (real)

Implemented output formats:
  tecplot
  opendx
  vtk
  netcdf

-----------------------------------------------------

Parameters section io:
    Files/IO ---------------------------------------: -
                                Primary grid filename: airfoil.grid
                                  Restart-data prefix: solution.pval.2500
                             Boundary mapping filename: para1
                                       Output format: tecplot
                                        Output level: 25
     Save History ----------------------------------: -
                            Save trajectory data (0/1): 1
                                  Save beta data (0/1): 1
                              Compute statistics (0/1): 1
    -------------------------------------------------: -
-------------------------------------------------------
Reading boundary mapping: file 'para1'

Parameters section main:
     General Parameter -----------------------------: -
                    2D offset vector (0 / x=1,y=2,z=3): 2
    Monitoring ------------------------------------: -
                              Monitor particle (0/1/2): 1
                            Monitoring particle values: dt_x_y_z_vx_vy_vz_elem
                        Monitoring significant figures: 4_8_4_4_4_4_4_4
    Perfect gas thermodynamic ---------------------: -
                                       Gas constant R: 287
                                   Gas constant gamma: 1.4
```

```
    Transport coefficients --------------------------: -
                                     Reynolds number: 10000000
                                      Prandtl number: 0.72
                                 Sutherland constant: 110.4
                       Sutherland reference viscosity: 1.716e-05
                    Sutherland reference temperature: 273
    Geometry ----------------------------------------: -
                                           Grid scale: 1
                               Reference relation area: 0
                 Reference length (pitching momentum): 1
           Reference length (rolling/yawing momentum): 1
                                 Origin coordinate x: 0
                                 Origin coordinate y: 0
                                 Origin coordinate z: 0
    References --------------------------------------: -
                                   Reference density: 2.35546682679853
                              Reference temperature: 273.15
                                 Reference pressure: 184654.584193385
                                    Reynolds length: 1
                              Reference Mach number: 0.22
                                 Reference velocity: 72.8833371080112
                             Reference outer pressure: 184654.584193385

Parameters section fluid/particle:
    Particle fluid interaction ----------------------: -
                               Particle reference length: 1
    Particle properties -----------------------------: -
                             Particle diameter (micron): 20,40,60
                                Density of second phase: 999
                                      Gravity constant: 9.81
                                     Gravity direction: 0 0 1
     -------------------------------------------------: -

-------------------------------------------------
 Flow reference values:
  Reference velocity [m/s] ...... 279.989
  Reference density [kg/m3] ..... 2.35547
  Reference pressure [N/m2] ..... 184655
  Reference temperature [K] ..... 273.15
  Reference length [m] .......... 1
  Reference viscosity [kg/m/s] .. 659.506
```

```
-----------------------------------------------------
 Flow initial dimensionless values:
  Initial velocity ..............  0.260308
  Initial Mach number ...........  0.22
-----------------------------------------------------


Parameters section tracer:
    Single trajectory --------------------------------: -
                                      Start grid type: Point
                    Starting point of particle (x,y,z): -90,-0.5,-8.5
  trajectory end
  --------------
     Bounding line for trajectories (2D) -------------: -
                                       Start grid type: Line
                                  First point (x,y,z): (none)
                                 Second point (x,y,z): (none)
                                   Resolution of line: 2
  trajectory end
  --------------
                                      Start grid type: Grid
                                 First corner (x,y,z): -2,0.066,-0.217
                                Second corner (x,y,z): -2,-0.066,-0.217
                                 Third corner (x,y,z): -2,-0.066,-0.352
                                Fourth corner (x,y,z): -2,0.066,-0.352
                                Points in i-direction: 112
                                Points in j-direction: 268
  trajectory end
  --------------
      Cut plane ---------------------------------------: -
                                       Start grid type: Plane
                                  Plane origin (x,y,z): -0.06,1.5,0
                             Plane normal vector (x,y,z): 1.0,0.0,0.0
                                      Catch plane (0/1): 0
  trajectory end
  --------------
      Cut plane ---------------------------------------: -
                                       Start grid type: Plane
                                  Plane origin (x,y,z): 0.0,1.5,0
                             Plane normal vector (x,y,z): 1.0,0.0,0.0
                                      Catch plane (0/1): 0
```

```
  trajectory end
  --------------
-----------------------------------------------------
 Number of Planes: 2
-----------------------------------------------------

     Boundary marker tracking ------------------------: -
                                     Start grid type: Marker
                                  Boundary marker id: (none)
  trajectory end
  --------------
     ------------------------------------------------: -

Parameters section tracking:
     Integrator ---------------------------------------: -
                                    Type of tracking: particle
                                Consider gravity (0/1): 0
                             Trajectory ODE integrator: Dormand_Prince_54
              Integration direction (forward/backward): forward
                                  Integrator epsilon: 1e-09
    Algorithm properties -----------------------------: -
                                           Fuzzyness: 0
                                     Search epsilon: 1e-09
      Limitations --------------------------------------: -
                            Maximum number of timesteps: 10000
                        Maximum x trajectory coordinate: 999999
     Refinement/Extension -----------------------------: -
                            Number of refinement levels: 4
                  Maximum change of distribution: 0.2
                                       Maximum distance: 0
-----------------------------------------------------
read primitive variables, file: <solution.pval.2500>
-----------------------------------------------------

-----------------------------------------------------
 Particle reference values:
  Particle diameter ..... 20          micron
  Inertia Parameter ..... 0.0942488   [1]
  Particle Reynolds number ..... 200  [1]
  Reference velocity ..... 279.989    m/s
  Reference nue ..... 7.28833e-06     m2/s
```

```
  TAU Inertia Parameter ..... 0.362067     [1]
  TAU Particle Reynolds number ... 768.322      [1]
  Particle Froude number ..... 89.3937      [1]
-----------------------------------------------------


-----------------------------------------------------
 BetaGrid: BetaGrid
 Loop 0: Added cells 1, Refined cells 0
-----------------------------------------------------

-------------------------------------------------------------------------------
 Particle Nr.: 1, x0 ...    -90, y0 ...  -0.5, z0 ... -8.5, Start Element: 1129
-------------------------------------------------------------------------------
 Particle Nr.: 1, xe ...   95.3, ye ...  -0.5, ze ... 30.3, Last Element: 5140
-------------------------------------------------------------------------------
-----------------------------------------------------
  Time Integration Statistics
  Number of global timesteps ................. 2506
  Number of rejected steps ................... 252
  Number of function evaluations ............. 13791
-----------------------------------------------------

-----------------------------------------------------
  Catch efficiency rate at wall

  Writing beta distribution: solution.particle.d20.beta.pval.2500.dat ... done
-----------------------------------------------------

-----------------------------------------------------
Output-file format ..: Tecplot 7 or higher (binary)
Output-file name ....: solution.particle.d20.trajectory.pval.2500.plt
-----------------------------------------------------

  Variables .........: X Y Z
  Writing Particle: Point1 ... done
done


-----------------------------------------------------
 Particle reference values:
```

```
  Particle diameter ..... 40          micron
  Inertia Parameter ..... 0.376995    [1]
  Particle Reynolds number ..... 400  [1]
  Reference velocity ..... 279.989    m/s
  Reference nue ..... 7.28833e-06     m2/s
  TAU Inertia Parameter ..... 1.44827 [1]
  TAU Particle Reynolds number ... 1536.64 [1]
  Particle Froude number ..... 89.3937 [1]
-----------------------------------------------------


-----------------------------------------------------
 BetaGrid: BetaGrid
 Loop 0: Added cells 1, Refined cells 0
-----------------------------------------------------

-------------------------------------------------------------------------------
 Particle Nr.: 1, x0 ...    -90, y0 ...  -0.5, z0 ... -8.5, Start Element: 1129
-------------------------------------------------------------------------------
 Particle Nr.: 1, xe ...   95.3, ye ...  -0.5, ze ... 30.3, Last Element: 5140
-------------------------------------------------------------------------------
-----------------------------------------------------
  Time Integration Statistics
  Number of global timesteps ..... 834
  Number of rejected steps ..... 212
  Number of function evaluations ..... 5231
-----------------------------------------------------

-----------------------------------------------------
  Catch efficiency rate at wall

  Writing beta distribution:  solution.particle.d40.beta.pval.2500.dat ... done
-----------------------------------------------------

-----------------------------------------------------
Output-file format ..: Tecplot 7 or higher (binary)
Output-file name ....: solution.particle.d40.trajectory.pval.2500.plt
-----------------------------------------------------

  Variables .........: X Y Z
  Writing Particle: Point1 ... done
```

```
done


-----------------------------------------------------
 Particle reference values:
  Particle diameter ..............  60           micron
  Inertia Parameter ..............  0.848239     [1]
  Particle Reynolds number .......  600          [1]
  Reference velocity .............  279.989      m/s
  Reference nue ..................  7.28833e-06  m2/s
  TAU Inertia Parameter ..........  3.25861      [1]
  TAU Particle Reynolds number ...  2304.97      [1]
  Particle Froude number .........  89.3937      [1]
-----------------------------------------------------


-----------------------------------------------------
 BetaGrid: BetaGrid
 Loop 0: Added cells 1, Refined cells 0
-----------------------------------------------------

-------------------------------------------------------------------------------
 Particle Nr.: 1, x0 ...    -90, y0 ...  -0.5, z0 ... -8.5, Start Element: 1129
-------------------------------------------------------------------------------
 Particle Nr.: 1, xe ...   95.3, ye ...  -0.5, ze ... 30.3, Last Element: 5140
-------------------------------------------------------------------------------
-----------------------------------------------------
  Time Integration Statistics
  Number of global timesteps ................. 565
  Number of rejected steps ................... 189
  Number of function evaluations ............. 3771
-----------------------------------------------------

-----------------------------------------------------
  Catch efficiency rate at wall

  Writing beta distribution:  solution.particle.d60.beta.pval.2500.dat ... done
-----------------------------------------------------

-----------------------------------------------------
Output-file format ..: Tecplot 7 or higher (binary)
```

```
Output-file name ....: solution.particle.d60.trajectory.pval.2500.plt
-----------------------------------------------------

  Variables .........: X Y Z
  Writing Particle: Point1 ... done
done

Wallclock runtime 5.68 sec
Tau particle solver done

-----------------------------------------------------
 total time in seconds: real 5.685e+00, user 5.268e+00 sys 1.665e-01
-----------------------------------------------------
```

11 Adaptation

11.1 Description

The adaptation program reads a hybrid (or tetrahedral) primary grid and a matching flow solution. With the aid of refinement sensor functions it is determined which edges of the primary grid have to be bisected depending on the desired dimensions for the resulting grid. Thus for all edges the value of

$$I_e \;=\; \Delta V_e ||x_e||_2{}^{\alpha}$$

will be calculated. With $x_e = x_{p_1} - x_{p_2}$, α an edge length scaling factor and

$$\Delta V_e \;= \max\left(c_{\phi_i} \frac{\Delta\phi_i}{(\Delta\phi_i)_{max}} \right), \text{ with } 0 \leq i < N,$$

where N the number of the selected solution values for the sensor function is and c_{ϕ_i} user defined scaling values.
For an equilibrated scaling of each part the calculated values must be distributed in $[0, 1]$. Therefore the maximum of all values is determined by

$$(\Delta\phi_i)_{max} \;= \max\left((\Delta\phi_i)_e\right), \text{ for all edges } e.$$

Currently there are three different sensor functions available which all base on differences of

1. the flow variables (diff)
$$\Delta\phi_i = |\phi_i(x_{p_1}) - \phi_i(x_{p_2})|;$$

2. the gradients of the flow variables (grad)
$$\Delta\phi_i = |grad(\phi_i(x_{p_1})) - grad(\phi_i(x_{p_2}))|;$$

3. the flow variables reconstructed to the edge midface (recon)
$$\Delta\phi_i = |(\phi_i(x_{p_1}) + \frac{1}{2} x_e \cdot grad(\phi_i(x_{p_1}))) - (\phi_i(x_{p_2}) - \frac{1}{2} x_e \cdot grad(\phi_i(x_{p_2})))|, \text{ with } x_e = x_{p_1} - x_{p_2};$$

4. the sum of the flow variables (sum)
$$\Delta\phi_i = |\phi_i(x_{p_1}) + \phi_i(x_{p_2})|.$$

Furthermore there are also some special indicators available, that one might use instead of the above mentioned :

- the global refinement indicator (all), which refines all available edges (This global refinement indicator can be used together with the refinement regions);
- the global derefinement indicator (del), which derefines all available edges;

- the vortex detection indicator, which will only work if also a specific variable (*Nk, Hn, l2, Q or VbI*) is present in the solution file. This indicator can only be employed to one of the solution values in the following way: The user has to specify the value of *Nk, Hn, l2 Q or VbI* for the parameter *Indicator user-values:* and set a threshold (δ) for this value in *Indicator user-scaling:*. The adaptation will then calculate the sensor as follows:

  1. For the flow variable $\phi = Nk$ or $\phi = VbI$

$$\Delta\phi = \begin{cases} 1 & : \quad \phi(x_{p_1}) > \delta \ \vee \phi(x_{p_2}) > \delta \\ 0 & : \quad \phi(x_{p_1}) \leq \delta \ \wedge \phi(x_{p_2}) \leq \delta \end{cases},$$

  2. for the flow variable $\phi = Hn$

$$\Delta\phi = \begin{cases} 1 & : \quad |\phi(x_{p_1})| > \delta \ \vee |\phi(x_{p_2})| > \delta \\ 0 & : \quad |\phi(x_{p_1})| \leq \delta \ \wedge |\phi(x_{p_2})| \leq \delta \end{cases},$$

  3. for the flow variable $\phi = l2$

$$\Delta\phi = \begin{cases} 1 & : \quad \phi(x_{p_1}) < \delta \ \vee \phi(x_{p_2}) < \delta \\ 0 & : \quad \phi(x_{p_1}) \geq \delta \ \wedge \phi(x_{p_2}) \geq \delta \end{cases},$$

  4. for the flow variable $\phi = Q$ is currently no sensor defined

$$\Longrightarrow \Delta\phi = 0.$$

Additionally, if the primary grid is hybrid (i.e. prismatic or hexahedral cells in the boundary layer), one can move the points on rays normal to the wall, depending on a user-defined y_+-value. This value can be set globals or separately for each viscous wall by using the specified parameters.
The refined grid is computed and the solution is interpolated to the new grid. New surface points are projected to the surface represented by a Bézier-Spline reconstruction from the old surface points. The new grid and the new solution are written out in NetCDF format.
Alternatively, the adaptation program can be used, by defining cells to be refined using external refinement information. This information has to be put into an ASCII-file containing the cell numbers.
The usage of the adaptation program is

```
bin-path/adaptation parameter-file
```

Optionally, a log-file can be defined as third argument.

11.2 Prerequisites

The adaptation algorithm can be started after the definition of a primary grid and the computation of a flow solution on this grid or alternatively the set up of a Selected-elements file. A suitable parameter setting has to be defined in addition.

11.3 Input options

The input options are defined in the parameter file. All input parameters, input file names and output file names can be defined in this file. The parameter names and the file names are explained below. They are listed in alphabetical order for the different sections (file-io, main, y_+ and refinement). Parameters without default values are printed underlined.

11.3.1 Input files

The major input file is the parameter file. It can contain all names of further input files listed below.

11.3.2 Input general parameters

Runtime optimisation parameter section

Indicator parameter section

Controls parameter section

General parameter section

Grid/Solution parameter section

11.3.3 Input boundary mapping parameters

Common
`Markers:` marker list, default not_defined

- The integers in the list are separated by ',' or '-' if a complete range (i.e. 3-6 = 3,4,5,6) is covered. Each marker specifies a set of surface elements in the primary grid.

`Type:` character string, default -

- The value of the type is the name of the boundary treatment. The number and the kind of additional parameters for a boundary part depend on the boundary treatment (see following subsections).

Engine exhaust
`Curvature reconstruction (0/1):` integer, default 1

- This parameter is used to switch surface reconstruction on or off.

Engine inflow
`Curvature reconstruction (0/1):` integer, default 1

- This parameter is used to switch surface reconstruction on or off.

Euler wall
`Curvature reconstruction (0/1):` integer, default 1

- This parameter is used to switch surface reconstruction on or off.

Viscous wall

`Curvature reconstruction (0/1):` integer, default 1

- This parameter is used to switch surface reconstruction on or off.

`Structured layer refinement (0/1):` integer, default 1

- If this parameters is switched off (0) the structured layer composed of hexahedra or prisms above this viscous wall is faded out for grid refinement

11.4 Sample parameter file

The parameter file could look like this:

```
Primary grid filename: configuration.0.grid
New primary grid filename: configuration.1.grid
Restart-data prefix: configuration.0.pval.50
New restart-data prefix: configuration.1.pval.0
Maximum surface angle (degree): 60
Percentage of new points: 50
```

11.5 Description of process information

The output information of the adaptation program starts with a line indicating the version and the compile time of the code. Then a list of the parameters used is printed. This is followed by information about the refinement indicator used. Then the iteration steps are printed out for marking so many edges that the new grid will have the desired dimensions.
If surface splining is invoked some information is printed behind the output about resulting local refinements. One line indicates a `dmax` and `dmin` value which are important. These values reflect the ratio between the surface point movement due to the projection to the reconstructed B-spline surface and the length of the local edge. If the maximum value becomes larger than 0.2 it is an indication that (accuracy-) problems can have been occurred. In this case, the value of the `maximum surface integrals` in the next preprocessing step should be checked.
The next output indicates which quantities (all found in the result file) have been interpolated to the new solution. This is followed by the new grid dimensions. At the end a volume check is performed. Using low quality input grids the surface splining can lead to collapsed elements. If negative volumes are found (indicated by a warning message), the grid should not be used. Work-arounds could be to rerun the adaptation without splining of surfaces.
An output file could look like this:

```
Parameters section io:
    Files/IO ---------------------------------------: -
```

```
                                Primary grid filename: rae2822_hybrid.nc
                               New primary grid prefix: rae.grid
                                   Restart-data prefix: output.pval.50
                               New restart-data prefix: restart
                                Selected-elements file: (none)
                     Automatic parameter update (0/1): 1
                                          Number of blocks: 1
    ------------------------------------------------: -

Parameters section main:
      General Parameter ------------------------------: -
                 Change wall normal distribution (0/1): 0
                                       Refinement mode: add
                                Detailed output (0/1): 0
                    2D offset vector (0 / x=1,y=2,z=3): 0
    Reference values -------------------------------: -
                                       Reynolds length: 1
     ------------------------------------------------: -
Check primary grid input
  7516 surface tris orientated
  11232 surface quads orientated

Current adaptation level is: 1

Parameters section refinement:
     Files/IO ---------------------------------------: -
                              Boundary mapping filename: (thisfile)
     Refinement Parameter ---------------------------: -
                                   Maximum point number: -1
                                Percentage of new points: 30
                                  Adaptation sensitivity: 0.9
                                     Minimum edge length: 1e-12
     Global Indicator settings ----------------------: -
                                        Indicator type: diff
                                       h-scaling power: 0.5
                             h-scaling reference length: 1
                                   Surface-weights file: (none)
      Indicator settings for user defined values -----: -
                                  Indicator user-values: (none)
                                 Indicator user-scaling: 0
       Equidistribution Indicator  -------------------: -
```

```
                                Indicator0 V-scaling: 1
                               Indicator0 rho-scaling: 1
                                Indicator0 Ht-scaling: 1
                                Indicator0 Pt-scaling: 1
                                 Indicator0 1-scaling: 0
     Refinement regime ------------------------------: -
                           Number of cut-out volumes: 0
                                  Minima x-direction: 0
                                  Minima y-direction: 0
                                  Minima z-direction: 0
                                  Maxima x-direction: 0
                                  Maxima y-direction: 0
                                  Maxima z-direction: 0
                                  Frustum min radius: 0
                                  Frustum max radius: 0
     Surface Reconstruction Parameter ----------------: -
                       Maximum surface angle (degree): 30
                        Maximal point movement factor: 0.2
     ------------------------------------------------: -
-----------------------------------------------------
Reading boundary mapping: file 'para'
Reading boundary parameter: file:'para'
-----------------------------------------------------
                                                 Map: 0
                                                Type: farfield
                                             Subtype:
                                             Markers: 4
                                                Name:
                       Curvature reconstruction (0/1): 0
                                      Target yplus: 0.0
                                         block end
-----------------------------------------------------
                                                 Map: 1
                                                Type: symmetry plane
                                             Subtype:
                                             Markers: 6
                                                Name:
                       Curvature reconstruction (0/1): 0
                                      Target yplus: 0.0
                                         block end
-----------------------------------------------------
```

```
                                                Map: 2
                                               Type: symmetry plane
                                            Subtype:
                                            Markers: 7
                                               Name:
                          Curvature reconstruction (0/1): 0
                                        Target yplus: 0.0
                                           block end
-------------------------------------------------------
                                                Map: 3
                                               Type: viscous wall
                                            Subtype: laminar
                                            Markers: 2
                                               Name: wall-1
                          Curvature reconstruction (0/1): 1
                                        Target yplus: 0.9
                                           block end
-------------------------------------------------------
                                                Map: 4
                                               Type: viscous wall
                                            Subtype: turbulent
                                            Markers: 1
                                               Name: wall-2
                          Curvature reconstruction (0/1): 1
                                        Target yplus: 1.0
                                           block end
-------------------------------------------------------
                                                Map: 5
                                               Type: viscous wall
                                            Subtype: turbulent
                                            Markers: 3
                                               Name: wall-3
                          Curvature reconstruction (0/1): 1
                                        Target yplus: 1.1
                                           block end

 No existing bars found in the grid !
 11232 from  11232 quads have not full edges !
Mesh-connectivity calculated:   562266 edges

 148411 points of 148411  (100.00 percent) for refinement considered
```

```
Differences indicator: Maximum / Reference values
                      |v| :     1.0 / 0.440246
                      rho :     1.0 / 0.326783
                       Ht :     1.0 / 0.513936
                       Pt :     1.0 / 0.495436

Range of edge length: min 1.970e-03 max 3.255e-01
Range of indicator:   min 4.707e-17 max 2.543e-02
  1.limit=1.272e-02   0.00% increase of point number  100.00% marked points
  2.limit=6.359e-03   0.04% increase of point number  100.00% marked points
  3.limit=3.179e-03   0.27% increase of point number  100.00% marked points
  4.limit=1.590e-03   1.13% increase of point number  100.00% marked points
  5.limit=7.948e-04   3.54% increase of point number  100.00% marked points
  6.limit=3.974e-04   9.76% increase of point number   99.04% marked points
  7.limit=1.987e-04  24.45% increase of point number   96.65% marked points
  8.limit=9.936e-05  54.66% increase of point number   88.97% marked points
  9.limit=1.490e-04  34.11% increase of point number   95.38% marked points

Resulting local refinements
  418426 tetras estimated (from   190957)
   66256 parent entries for tetras estimated

  185304 prisms estimated  (from   180384)
    2232 parent entries for prisms counted

  116616 triangles (from    85028)
   14215 parent entries for triangles estimated

   11352 quads (from    11232)
     120 parent entries for quads counted


Starting calculation of new grid points !
Starting surface reconstruction over selected walls !

  1348 curve-points on surface-edges
   105 marked surface-edges for reconstruction

Starting curve-line reconstruction !
```

```
 Starting curved-surface reconstruction !

 Starting point movement in prism piles !

Interpolation of given values!
density
x_velocity
y_velocity
z_velocity
pressure
sa_viscosity
eddy_viscosity
done

Dimensions of Grid:
  old:   85028 nstri   190957 ntetra   180384 nprism   11232 nsquad    1355 nbar_2  14841
  new:  116616 nstri   418426 ntetra   185304 nprism   11352 nsquad       0 nbar_2  19903

Test tau grid:
 Test elements:
  orientation of 146201 tetras set
 Test face to element connectivity:
  face->element connectivity is ok.
tau grid is ok after changes.

write file restart.ad.1.pval.50
Output file <restart.ad.1.pval.50> written
Output file <rae.grid.ad.1> written


Times in seconds:
Wallclock: 34.5  User: 26.9  System: 1.2

done
```

12 Deformation

12.1 Introduction

In case of an unstructured grid, there are mainly two possibilities to change the volume grid according to surface deflections: The first would be to remesh the whole grid which can became time consuming or the second way is to apply a volume grid deformation. The usage of the second possibility will be explained in the following chapter in more detail.

The assumptions to a volume grid deformation is usually the amount of surface deflections. Small perturbation can be handled quickly and still result in a good quality grid for a flow simulation. Higher perturbations usually include a cell repair to provide again a suitable computational grid. Regardless of that extra algorithms the volume deformation is a very reliable and efficient feature which can safe a lot of computational time in comparison to any remeshing procedure performed.

Restriction to the deformation apply especially if intersection lines changes or deflections overcome a certain amount that cell repairing is not anymore sufficient during a fluid simulation task, e.g. at optimization loops. At the moment intersection lines which are remodeled by a CAD-System implies the remeshing of the computational domain at any time.

12.2 Description

The deformation program reads a hybrid primary grid and optionally a file containing new coordinates for some grid points including a mapping of these points to the originally grid point number. Alternatively, a displacement of boundary points can be defined by input parameters. In order to apply the new coordinates surrounding grid points have to be moved in order to keep a valid grid without collapsed cells. This deformation of the grid is computed and a new deformed grid is written to disk. There are many different deformation algorithms known in the literature. The one implemented into TAU are now explained in more detail.

Freeform deformation The freeform deformation acts on the surface mesh only and the volume grid is deformed with the algorithm explained in Paragraph 12.2. In the present implementation a trivariate B-Spline volume is defined by:

$$\vec{P}_{object}(u_0, v_0, w_0) = \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \sum_{k=0}^{n_w} N_{i,m_u}(u_0) N_{j,m_v}(v_0) N_{k,m_w}(w_0) \vec{Q}_{i,j,k} \tag{45}$$

with B-Spline basis functions, calculated recursively:

$$
\begin{aligned}
N_{i,m}(u) &= \frac{u-u_i}{u_{i+m-1}-u_i}N_{i,m-1}(u) + \frac{u_{i+m}-u}{u_{i+m}-u_{i+1}}N_{i+1,m-1}(u) \\
&\text{and} \\
N_{i,1}(u) &= \begin{pmatrix} 1 & if & u_i \le u \le u_{i+1} \\ & 0 & else \end{pmatrix}
\end{aligned} \tag{46}
$$

The control points $\vec{Q}_{i,j,k}$ define the basic deformation lattice, that encloses the object, defined by polygonal curves and surfaces, with points $\vec{P}_{Object}$, which may form a human head, see left Figure 24. In the first step of the deformation procedure all object points $\vec{P}_{Object}$ are mapped into the B-Spline volume by solving for u_0, v_0, w_0 using a Newton iteration method, see middle Figure 24.

In a second step the previously calculated u_0, v_0, w_0 are re-mapped to achieve the deformed objects points $\vec{P}_{Newobject}$ from:

$$
\vec{P}_{Newobject}(u_0,v_0,w_0) = \sum_{i=0}^{n_u}\sum_{j=0}^{n_v}\sum_{k=0}^{n_w} N_{i,m_u}(u_0)N_{j,m_v}(v_0)N_{k,m_w}(w_0)\vec{R}_{i,j,k} \tag{47}
$$

Eq. 45 is identical to Eq. 47 but has different lattice control points $\vec{R}_{i,j,k}$, which determines finally the deformation of an object, right Figure 24.

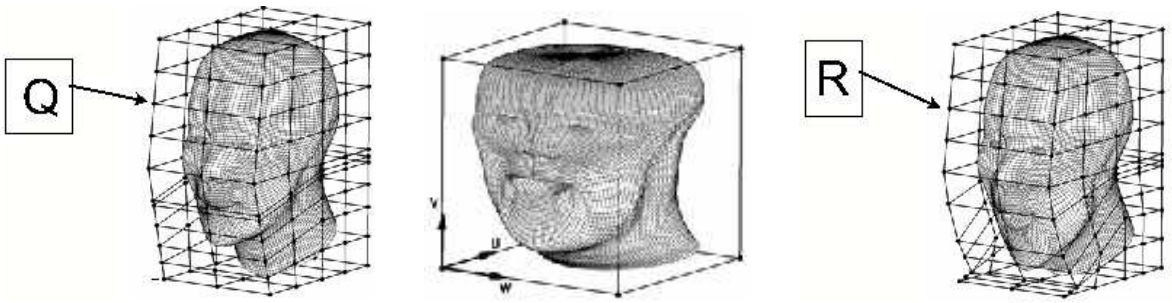


Figure 24: Initial lattice box with control points Q, left figure. Middle Figure shows the parameterized space, and right Figure shows the deformed surface with control points R.

As the present method of freeform deformation is an **indirect** method, where the surrounding space is deformed, rather than the object itself, proper deformation lattices have to be provided, to achieve a desired geometrical variation of an object. The lattices required for deformation are structured up to now and can therefor be easily generated.

Volume deformation The volume grid deformation consists of three different stages, which are first an algebraic prolongation of the input displacements into the volume grid, secondly a repair stage which is followed then by a smoothing step.

For the algebraic prolongation the displacements are transported from the deformed surface to interior points by averaging the displacements of neighboring points in the back and by superimposing the local point movement due to the rotation of points on the deformed surface. The rotation component is reduced with increasing distance from the displacement surface. The local displacements are reduced by a small fraction of the local cell size.
In the second stage all cells which are collapsed and all those which have angles smaller than a given limit are marked for repair. All coordinates of these cells and a certain surrounding are recomputed by local volume splines based on the coordinates of the initial grid and the new coordinates at the surrounding boundary.
In the third stage a smoother is applied on all points of tetrahedra which contain inner angles below a given limit.

The success of the deformation depends a lot on the quality of the initial grid. As more regular the initial elements are as higher is the probability for a successful deformation. Sliver or very skew elements in a region which need not only to be rotated or translated but also to be deformed makes it very difficult to keep a valid grid. An output of the worst angles of the different element types of the initial grid might give an idea of its quality. A grid with element angles below a few degree (it is difficult to give a general threshold, thus let us say below 5 degree) seem not very well suited for grid deformation. They will probably allow for very small deformations only.

12.3 Prerequisites

The deformation algorithm can be started after the definition of a primary grid and a given deformation. A suitable parameter setting has to be defined in addition.

12.4 Input options

The input options are defined in the parameter file. All input parameters, input file names and output file names can be defined in this file. The parameter names and the file names are explained below. They are listed in alphabetical order. Parameters without default values are printed underlined.

12.4.1 Input files

The major input file is the parameter file. It can contain all names of further input files listed below.

Concerning the additional input file for the freeform deformation we will give an impression of the format used for describing the lattice boxes. Using the parametric grid generation system *MCADS* will provide directly the files read in by the TAU deformation tool. A native *MCADS* ASCII file may look like:

```
# MegaCads Geometry Dump
# PROJECT= "/MCADS/"
# SCRIPT= "/MCADS/SAVE_PROC_FILE"
# number of datasets in this file:
          4
# - - - - - - - - - - -
# -  Dataset  header  -
# - - - - - - - - - - -
# Scaling and Translating:
# scalx scaly scalz   x0   y0   z0   x0rot  y0rot  z0rot   xrot   yrot   zrot
    1.0   1.0   1.0  0.0  0.0  0.0     0.0    0.0    0.0    0.0    0.0    0.0
# - - - - - - - - - - -
# Dimensions:
#  imax   jmax   kmax
      2      3      2
# - - - - - - - - - - -
# Coordinates:
4.3371760000e-01 1.8415550000e-15 -3.8088260000e+00
4.9753040000e-01 1.8337400000e-15 5.3600780000e+00
4.3371760000e-01 1.2654918394e+00 -3.8088260000e+00
4.9753040000e-01 1.2654918394e+00 5.3600780000e+00
1.7514050100e+01 1.2000000000e+01 -2.8285410000e+00
1.7524420100e+01 1.2000000000e+01 3.7040740000e+00
6.3706770000e+01 3.2468840000e-15 -4.2491880000e+00
6.3770590000e+01 3.2440060000e-15 4.9197170000e+00
6.3706770000e+01 1.2654918394e+00 -4.2491880000e+00
6.3770590000e+01 1.2654918394e+00 4.9197170000e+00
6.2465190100e+01 1.2000000000e+01 -2.8998470000e+00
6.2475560100e+01 1.2000000000e+01 3.6327680000e+00
# - - - - - - - - - - -
# -  Dataset  header  -
# - - - - - - - - - - -
# Scaling and Translating:
# scalx scaly scalz   x0   y0   z0   x0rot  y0rot  z0rot   xrot   yrot   zrot
    1.0   1.0   1.0  0.0  0.0  0.0     0.0    0.0    0.0    0.0    0.0    0.0
# - - - - - - - - - - -
# Dimensions:
#  imax   jmax   kmax
      2      4      2
# - - - - - - - - - - -
# Coordinates:
```

```
1.7514050100e+01 1.2000000000e+01 -2.8285410000e+00
...
3.8485890100e+01 1.7500000000e+01 2.1548380000e+00
# - - - - - - - - - - -
# -  Dataset  header  -
# - - - - - - - - - - -
# Scaling and Translating:
# scalx scaly scalz   x0   y0   z0   x0rot  y0rot  z0rot   xrot   yrot   zrot
    1.0   1.0   1.0  0.0  0.0  0.0     0.0    0.0    0.0    0.0    0.0    0.0
# - - - - - - - - - - -
# Dimensions:
#  imax   jmax   kmax
      2      2      2
# - - - - - - - - - - -
# Coordinates:
2.5374690100e+01 1.7500000000e+01 9.3022430000e-01
...
4.5250020100e+01 2.9950000000e+01 2.7573090000e+00
# - - - - - - - - - - -
# -  Dataset  header  -
# - - - - - - - - - - -
# Scaling and Translating:
# scalx scaly scalz   x0   y0   z0   x0rot  y0rot  z0rot   xrot   yrot   zrot
    1.0   1.0   1.0  0.0  0.0  0.0     0.0    0.0    0.0    0.0    0.0    0.0
# - - - - - - - - - - -
# Dimensions:
#  imax   jmax   kmax
      2      2      2
# - - - - - - - - - - -
# Coordinates:
3.4142570100e+01 2.9950000000e+01 7.9510120000e-01
...
5.5194730100e+01 4.9450000000e+01 2.6227480000e+00
```

Lines marked with a # do not need to be considered and are only information output lines written by *MCADS*.

12.4.2 Input general parameters

The following list displays all general parameters ordered according to the parameter groups.

parameter section

Runtime optimisation parameter section

Grid deformation parameter section

Internal deformation parameter section

Controls parameter section

External deformation parameter section

Motion parameter section

Test deformation parameter section

12.4.3 Input boundary mapping parameters

Common

`Freeform box number:` integer, default -1

- The freeform boxes, either for the initial and deformed one, are given through an external file which is presented in the right figure Figure 25. The box numbering in the external file is in ascending order. The box numbers have then to be inserted into the right wall boundary map, see figure Figure 25.

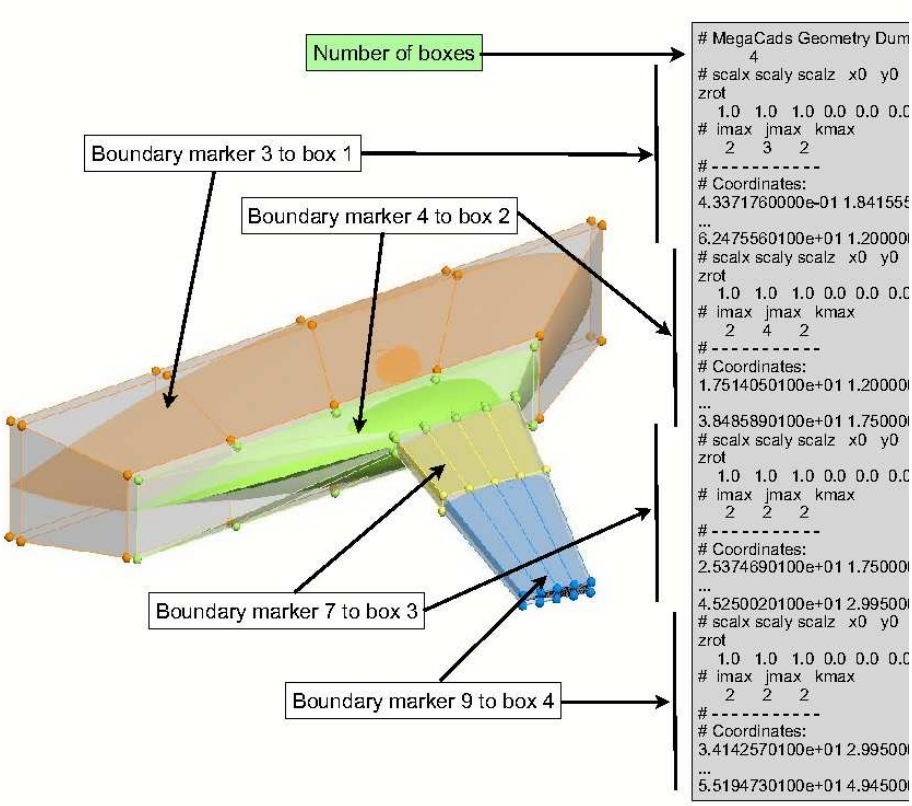


Figure 25: Corresponding Freeform boxes and boundary markers.

`Markers:` marker list, default not_defined

- The integers in the list are separated by ',' or '-' if a complete range (i.e. 3-6 = 3,4,5,6) is covered. Each marker specifies a set of surface elements in the primary grid.

`Surface deformation type (0/1/2):` integer, default (none)

- This parameter defines how to handle points of this boundary type: the three possibilities (0/1/2) are "fixed", "project", "free". Per default boundaries of wall type are "fixed", boundaries of chimera type or farfield type are "free", all others are handled of type "project".
  The handling "project" means that points can move along this boundaries only. This type should not be applied to curved surfaces or surfaces with a shape which needs to be conserved, because the projection is not based on CAD-data. It is a simple projection to the plane defined by the local surface element.
  The handling "free" means that these boundary points are allowed to move in any direction. The handling "fixed" means that these boundary points are fixed, either to the initial coordinates, or to the deformed coordinates defined in the displacement input.

`Type:` character string, default -

- The value of the type is the name of the boundary treatment. The number and the kind of additional parameters for a boundary part depend on the boundary treatment (see following subsections).

12.5 Sample parameter file

The parameter file could look like this:

```
-----------------------------------------------------
required parameters
-----------------------------------------------------
  Files/IO ---------------------------------------: -
                         Boundary mapping filename: (thisfile)
                             Primary grid filename: tau.grid
-----------------------------------------------------
io
-----------------------------------------------------
  Controls ---------------------------------------: -
                   Automatic parameter update (0/1): 0
-----------------------------------------------------
preprocessing
-----------------------------------------------------
  Runtime optimisation ---------------------------: -
                2D offset vector (0 / x=1,y=2,z=3): 0
-----------------------------------------------------
deformation
-----------------------------------------------------
  Test deformation -------------------------------: -
                                Delta-Z at outer-Y: 0.01
                            Outer-Y for deformation: 100
                            Start-Y for deformation: 0
  External deformation ---------------------------: -
                       Deformation reduction factor: 0.85
                      Deformed coordinates filename: (none)
                        Final freeform box filename: (none)
                      Initial freeform box filename: (none)
             Use coordinates as displacement (0/1): 0
-----------------------------------------------------
Motion-IO
-----------------------------------------------------
  Motion -----------------------------------------: -
             Degree of Fourier series for rotation: 0
```

```
     Degree of Fourier series for translation: 0
            Degree of polynomial for rotation: 0
         Degree of polynomial for translation: 0
Fourier coefficients for rotation (cos) pitch: 0
 Fourier coefficients for rotation (cos) roll: 0
  Fourier coefficients for rotation (cos) yaw: 0
Fourier coefficients for rotation (sin) pitch: 0
 Fourier coefficients for rotation (sin) roll: 0
  Fourier coefficients for rotation (sin) yaw: 0
 Fourier coefficients for translation (cos) x: 0
 Fourier coefficients for translation (cos) y: 0
 Fourier coefficients for translation (cos) z: 0
 Fourier coefficients for translation (sin) x: 0
 Fourier coefficients for translation (sin) y: 0
 Fourier coefficients for translation (sin) z: 0
```

13 Utility programs

Additional programs for pre- and postprocessing purposes are available. There are grid converters for translating hybrid grids generated with a specific grid generator from the native format into the format used by the TAU-Code:

- centaur2tau
- patran2tau
- icem2tau
- flower2tau
- make_conform
- smooth_taugrid

For partitioning of primary grids into several load-balanced domains with one cell layer of overlap the program to use is:

- ptau3d.subgrids

For simple manipulation of the grids (in TAU-Format) and for grid checks another program exists, which is:

- setup_taugrid

For enhancing the quality of the grids (in TAU-Format) the following program exists, which is:

- smooth_taugrid

For generation of a primary grid by interpolation of point coordinates of pregenerated grids:

- intermesh

For gathering result files (field output data or surface output data) from several grid-partitions into a single file, or for scattering data from one single file to several partitions, the utility programs are:

- gather
- scatter

ASCII cut plane output from the solver can be postprocessed using:

- sort_cut

ASCII log-files from the TAU-Code solver, containing monitoring output and and further information, can be postprocessed using:

- analysis

For visualization of TAU-Code grids and results the converter to Tecplot, Ensight, OpenDX or Fieldview is:

- tau2plt

Finally, for visualization of TAU-Code grids and results to CGNS the converter is:

- tau2cgns

Further useful programs are available in the NetCDF-package - which is used by the TAU-Code. The most important extra-functionality for the handling of TAU-Code data is the 'ncdump'-program.
These utilities are described in more detail below in alphabetical order.

13.1 The 'analysis' program

The analysis program may be used to quickly combine/split data written to solver log-files. It thus provides a significantly faster alternative (with more functionality) to the *awk* scripts.

Either all data may be used, or the user may specify the data he/she wishes to save for further analysis. The variable names used correspond to the names of the Monitor list variables.
The 'analysis' program is controlled through settings made in a parameter file.

The usage is

```
bin-path/analysis parameter-file
```

The parameter file settings are described in the following section. The names of the stdout files to be analyzed are read from the "Analysis" file. A typical "Analysis" file may look like the following:

```
RUN_MODEL_0.stdout.0
RUN_MODEL_1.stdout.0
#RUN_MODEL_2.stdout.0
#RUN_MODEL_3.stdout.0
```

Here the hatch mark is used to skip a stdout file. Note that the best way to create an "Analysis" file is to redirect the output of the unix *ls* command to a file. The file may then be edited by hand if required.

An additional feature that has been recently included is the "profiling" option. The profiling option is designed to extract sets of data (points, lines) from the solver and to store the data

in vector form. A typical profiling parameter file for the solver is shown below. Here, as one can see by examining the "Profile output values" string, the user has requested that data at two points in the field be stored, and that at each point the coordinates (x,y,z), the velocities (u,v,w) and the pressure is stored. Thus there are seven storage vectors created, with the length of each vector being determined by the number of points and the number of profile data sets written (Profile every n steps). Thus, letting a_{in} denote the value of variable a for point i at step n, the contents of the storage vector for pressure are then

$$p_{11}, p_{21}, ..., p_{n1}, p_{12}, ...P_{n2}, p_{31}..., p_{n3}, ...p_{nm}$$

where m is the profile output period. After m iterations all vectors are written to a NetCDF data file, which must be then analyzed by the analysis tool to allow extraction of individual data sets for each point or line.

```
Number of profiles: 2
Profile output period: 1000
Profile every n steps: 1
Profile output values: xyz_v_p
Boundary point (0/1): 0      0
Profile support x: -0.1      0.99
Profile support y:  0.5      0.5
Profile support z:  0.004    0.032
Profile normal x: 0 0
Profile normal y: 0 0
Profile normal z: 0 0
Profile range: 0 0
```

After completion of the solver run one has the following two profile files (from above parameter file].

```
<user_defined_prefix>.profile_output_stream.1000
<user_defined_prefix>.profile_output_stream.2000
```

Copy the names of these two files to a file, let us call the file **plist**. The contents of **plist** look as follows

```
<user_defined_prefix>.profile_output_stream.1000
<user_defined_prefix>.profile_output_stream.2000
```

Important - no blank spaces after last name or code will fail. Now create an analysis driver file as follows:

```
             Analysis file: plist
          Analysis request: profile
   Analysis profile request: all
           Output filename: B
```

and then invoke the analysis program:

```
taudir/bin/analysis <analysis>
```

and you will then create two files (one for each point).

```
B .point.0 - contains x,y,z,vx,vy,vz,p for point 0
B.point.1  - contains x,y,z,vx,vy,vz,p for point 1
```

13.1.1 Sample parameter file

Typical parameter file:

```
-------------------------------------------------------
solver
-------------------------------------------------------
  Timestepping Start/Stop ------------------------: -
                                     Output period: 10
-------------------------------------------------------
analysis
-------------------------------------------------------
                           Analyse inner loop (0/1): 0
                                  Analyse variables: (none)
                                      Analysis file: (none)
                                   Analysis request: join
                       Extract nearest profile (0/1): 0
                                    Output filename: CONCAT
                                    Required format: (none)
-------------------------------------------------------
surface output
-------------------------------------------------------
                              Surface output period: 200
-------------------------------------------------------
tau2plt
-------------------------------------------------------
  Output Control ---------------------------------: -
                                     Output format: tecplot
```

13.1.2 Input parameters

Output format (page 339)
Output period (page 85)
Surface output period (page 113)

`Analyse inner loop (0/1):` integer, default 0

- Include inner loop monitor output when analyzing dual time output. The default value turns this option off.

`Analyse variables:` character string, default (none)

- This parameter returns a list of requested variables for analysis. The format of the string is identical to that of the monitor list in the solver. If the default value "**(none)**" is entered, ALL monitored variables in the monitor file are read and stored in a temporary memory location. Alternatively the user may specify a list of variables(in a manner identical to that of solver output monitoring) as follows:

  ```
  Analyse variables: Iter_Residual_C-lift_C-drag_Angle-a
  ```

  Note that the "_" symbol is used as a name separator (earlier releases used a # symbol) as this is consistent with the solver monitoring. In this case only the variables listed in the string are stored for further analysis.

  NOTE: If the user wishes to scan a list of files with the "(none)" option he/she must ensure that the number of monitoring variables in each file is identical or memory allocation problems in the tool will cause the code to crash (this may be changed in a later release). However, this is usually not a problem since specifying a list of variables (and ensuring that the variables are present in all files) means that the files may contain different sized sets of data. In other words, if you are trying to concatenate and analyze data items from files with different numbers of monitored variables you must specify the variables you require to be extracted. Note that if a variable is not present in a file the code will terminate.

`Analysis file:` character string, default (none)

- This parameter returns the name of the Analysis file, where the names of the stdout files (to be analyzed) reside.

`Analysis profile request:` character string, default (none)

- This parameter instructs the analysis tool to treat profile data files. At present this functionality is being improved since the current algorithm implemented in the TAU-Code will only work for a orthogonal structured mesh when extracting line data. To activate the parameter value should be (all).

`Analysis request:` character string, default join

- This parameter returns a user request. At present the option 'join' is defined. The option 'join' simply combines the requested elements of the stdout files, and then writes them into a user-defined file.

`Average variables (0/1):` integer, default 0

- Setting the parameter to 1 activates computation of the running means for the data stored in the stdout file list. The list of variables which is extracted from the is set by the parameter "Analyse variables". In the case of a DES computation which is restarted from a preliminary RANS calculation (this is advisable when e.g. the flow has strong stationary vortex systems which develop more quickly in a RANS solution) the initial transients present in the solution will significantly increase the length of data required to obtain a stochastically stationary mean. In order to avoid this problem the running mean can be started at the Nth data sample, where N is set by the parameter "Number before averaging". Averaging integral coefficient contained in the stdout file is useful to see if the integral coefficients computed for a wing in an unsteady flow have reached statistical convergence. Additionally, the mean integral coefficients are usually returned in experimental measurements so that use of this feature should help in proper comparison with experiment.

`Extract nearest profile (0/1):` integer, default 0

- Setting this parameter to 1 allows a search over the interpolated data for all domains to find which domain contains the interpolated point, and then extracts the point data from the domain file. This is a quicker method for the data sizes considered when compared to a search algorithm to determine which points belong to which domain prior to interpolation.

`FFT variable:` character string, default (none)

- At present only a LOMB periodogram calculation may be activated. The code is activated by giving the parameter a name of a variable. This variable must be present in the list of active variables noted above. Activation of this analysis results in file containing the following data `circular frequency , contribution to spectrum, x data, y data` A small section of output is written to stdout:
  - NO OF SAMPLES: number of data samples;
  - Index max element: Index of data set corresponding to peak frequency contribution;
  - Ratio of max signal to random noise: low values of this indicator indicate the presence of a periodic signal.

  The file is named **FFT.Output filename** by default and no provision for automatic renaming of the files is provided. Allocation of the output filename is described immediately below.

`Number before averaging:` integer, default 20

- This parameter is used to specify a delay before the instantaneous means are computed. This is to prevent initial transients from influencing the computed means.

`Output filename:` character string, default CONCAT

- This parameter returns the <Base file name> or the root name for the saved series of files.

`Required format:` character string, default (none)

- This parameter is used to specify the format used in writing the data to a file. If the default *(none)* is specified, then the formatted output is equivalent to a *%12.4e* C-format statement. A typical use for this parameter might be for an unsteady calculation in dual time or global time where the time step is quite small. An appropriate setting of the parameter might then be as shown below.

```
Required format: %18.9e
```

13.2 The 'centaur2tau' program

The centaur2tau program converts a grid (both 2D and 3D) generated using Centaur into NetCDF format suitable for preprocessing. Usage is self explanatory - all command line options can be viewed by calling 'centaur2tau' without any arguments. One argument (-y) needs additional explanation. The default behavior of the program includes the remeshing of some pyramids in order to disconnect chains of pyramids connected with their vertical edges. The option -y switches off this operation. This can be useful if one intends to use the Centaur adaptation. However in some grids (grids with a lot of pyramids), this option leads to a bad behavior of the tau adaptation during marking edges for refinement.
The 'centaur2tau' program can also be used to convert the surface (.fvs), prism (.fvs) and tetrahedra (.fvt) files generated by Centaur during the grid generation process.
Note: because the binary file format of centaur is not platform independent errors during reading the files might occur if not using this program on the same machine which had been used for the grid generation!

13.3 The 'icem2tau' program

The icem2tau program converts a unstructured or hybrid grid from Icem-domain format to TAU-format or vice versa. This converter needs Icem-domain-libraries, which are not freely available. It can be compiled only if an Icem-CFD package is installed. The usage is self-explaining (usage output when starting the binary).

13.4 The 'flower2tau' program

The flower2tau program converts 2D and 3D FLOWer-format grids into primary grids in NetCDF-format suitable for TAU preprocessing. The program can be driven via the command line or via a parameter file. All command line options can be viewed by calling 'flower2tau' without any arguments. A sample parameter-file is printed when starting the executable with a dummy file. The form of the parameter file is given below.

```
Common ----------------------------------------: -
                              Grid input filename: (none)
                           Grid topology filename: (none)
                             Grid output filename: tau.grid
                    Boundary mapping output (0/1): 0
             Flower grid output (ascii=0,binary=1): 0
                        Maximal coarse grid level: 1
                    Swap y and z coordinate (0/1): 0
                  Removal of singular lines (0/1): 0
2D grid----------------------------------------: -
                       Cells in spanwise direction: 1
                      Offset in spanwise direction: 1.0
Test grid--------------------------------------: -
                           Test grid (no=0,yes=1): 0
```

Following is a description of the parameters used in the flower2tau program.

`Boundary mapping output (0/1):` integer, default 0

- If this flag is turned on (1) a boundary mapping file in TAU-format is written.

`Cells in spanwise direction:` integer, default 1

- A 2D grid in TAU is a 3D grid with one cell layer in spanwise direction (default). This parameter creates a 3D grid from a 2D FLOWer-format grid with the specified number of layers in spanwise direction.

`Flower grid output (ascii=0,binary=1):` integer, default 0

- The flag declares if a coarser flower grid is written in ASCII (0) or binary (1) format

`Grid input filename:` character string, default (none)

- The name (including the path) of the grid file in ASCII or binary FLOWer-format. The program identifies a grid in ASCII FLOWer-format by reading the first line as a comment line. The first character in an ASCII grid file has to be the $ character.

`Grid output filename:` character string, default tau.grid

- The name of the output primary grid file in NetCDF format.

`Grid topology filename:` character string, default (none)

- The name (including the path) of the topology file of the FLOWer grid.

`Maximal coarse grid level:` integer, default 1

- If the maximal coarse grid level is greater than one, coarser grids up to the specified level are generated in FLOWer-format and TAU-format. The level is added to the names of the output grid files.

`Offset in spanwise direction:` float, default 1.0

- A 2D grid in TAU is a 3D grid with one cell layer in spanwise direction. This parameter declares the offset of the layer in spanwise direction. If the parameter `Cells in spanwise direction` is specified the offset applies to all cell layers together.

`Removal of singular lines (0/1):` integer, default 0

- A singular line in a flower grid is composed of degenerated hexahedrons. A degenerated hexahedron is a hexahedron whose edges or faces may have a length or an area of zero. If this flag is turned on (1) the degenerated hexahedrons are replaced by prisms, pyramids or tetrahedrons. If a degenerated hexahedron does not reduce to a prism, pyramid or tetrahedron a, new grid point at the barycenter of the degenerated hexahedron is created and the degenerated hexahedron is decomposed into pyramids and tetrahedrons.

`Swap y and z coordinate (0/1):` integer, default 0

- When using the preset coordinate system of the FLOWer-Code the angle of attack is determined by the inclination of the geometry in the x-y plane so that the upper wing surface normal is directed to the positive y-direction, the span lies along the z-axis and the onflow direction is in the positive x-direction. Compared to the TAU-Code the y- and z-axis are exchanged. If this flag is turned on (1) the y- and z-coordinates in the TAU-grid are swapped explicitly. If this flag is turned off (0) the coordinate system is taken from the parameter `icoord` in the topology file and transformed to the TAU coordinate system.

`Test grid (no=0,yes=1):` integer, default 0

- The flag declares if a grid test is to be performed. The grid test includes a volume check as in the test of the program `setup_taugrid`. If the grid test fails by reason of badly shaped elements in the converted grid, a Tecplot file consisting of these elements is written to enable the examination of the badly shaped elements.

13.5 The 'make_conform' program

The make_conform program removes hanging faces and degenerated elements by subdivision of cells. Input of this tool can be a hexahedral grid in StarCD format or a grid in TAU-format with degenerated prisms and/or hexahedra and hanging faces, e.g. grids which arise from Solar's grid generation. Because this is not Solar's native grid format and the tool in the current stage only works with special StarCD grids we avoided the names starcd2tau or solar2tau.
The types of hanging face connections the converter can remove is shown in figure 26. The set of possible or remeshable degenerated elements consists of the regular prism, pyra-

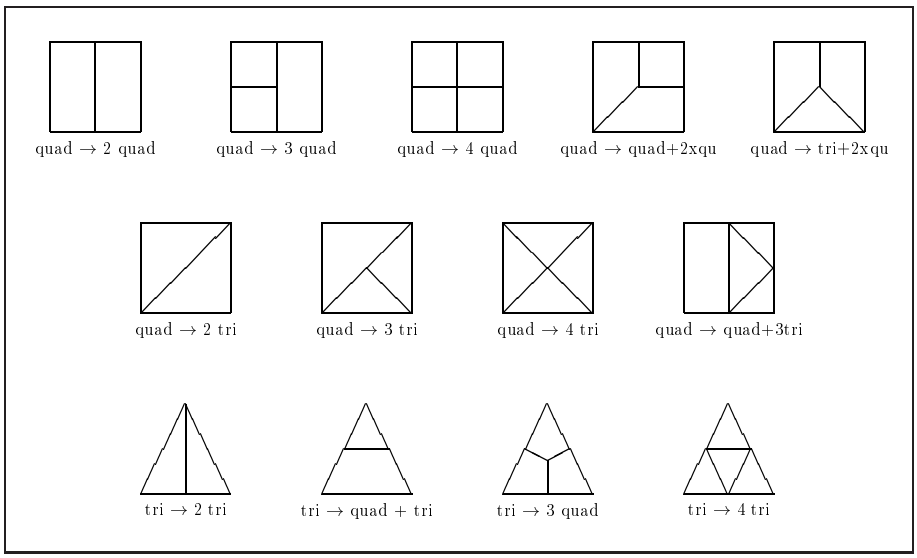


Figure 26: Hanging face connection types which can be removed by make_conform.

mid, tetrahedron elements and of three non-regular element types called degenerated-5-node, degenerated-6-node and degenerated-7-node. Figure27 shows their shape. They originate from identifying some points in hexahedra or prisms. This process can transform a hexahedron into any other regular element type or in to one of the three non-regular ones. A degenerated prism can form a tetrahedron, a pyramid or a degenerated-5node element.
The usage of the converter is self-explaining (usage output when starting the binary). After a call `make_conform <grid_name>`, the converter tries to read the input from a TAU-Code NetCDF file. If there is not such a file `<grid_name>` is taken for a StarCD grid name prefix and the converter tries to read a StarCD grid from `<grid_name>.vrt`, `<grid_name>.cel`, `<grid_name>.brd` and `<grid_name>.cpl`. If the grid can successfully be converted, a valid TAU-Code grid with the name `<grid_name>.conf` is written. The input format can be specified with the options `-r_tau` or `-r_starcd`. The name of the output grid can be chosen with the option `-o <output_grid_name>`.

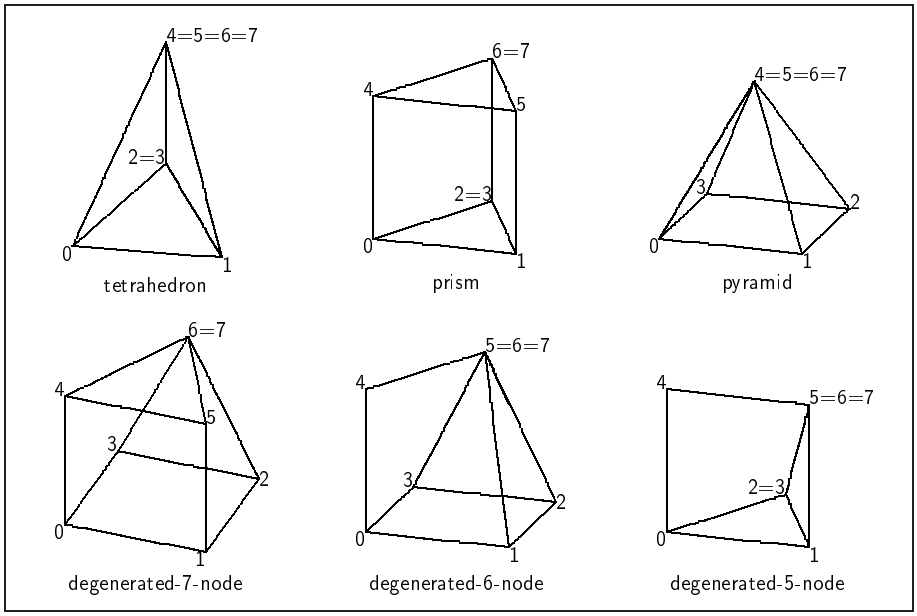


Figure 27: Types of degenerated elements which can be removed by make_conform.

The requirements of the converter with respect to the elements quality are higher than those of the standard grid check. If cells of poor quality have to be subdivided (and there is no subdivision to valid elements with only non-negative tetrahedra parts) the converter will fail. Also if the converter finds degenerations other than those described above, or can not interpret all open inner face connections as one of the described hanging face types, a conversion of the grid is not possible. The log output indicates the problems.

13.6 The 'tau2cgns' program

At present the tau2cgns program is a very preliminary implementation. The code can be used to write a TAU-Code grid and solution file to a CGNS formatted file (and vice versa). The code is driven via a parameter file; the form of the parameter file is given later in this section.

The tau2cgns code will not be compiled by a general make all call at the TAU-Code root directory. Since the CGNS library is required, the user must first obtain a copy of the cgns library for his/her operating system. A simple alternative is to build from the source code distribution found under the CGNS web site. Once the cgns lib is installed, the next task is to place the following lines in your .taudef file.

```
HAVE_CGNS_LIB = 1
CGNS_INC   = <path to your CGNS include files>
CGNS_LIB   = <path to your CGNS library>
```

When this has been done you should then issue the command < *make tau2cgns* > to build the TAU-Code to CGNS converter. **Note that the compilation will fail unless you have**

built the TAU-Code libraries first!!! In order to clean up prior to another compilation (you may have installed a more recent version of the CGNS library) you should issue the command < *make tau2cgnsclean* >, and all object files will be removed from the tau2cgns src directory, as well as the binary from the taudir/bin path. In order to use the compiled program, a control parameter file is invoked in much the same way that control is invoked for other TAU-Code program modules. Following is a list of the parameters used in the tau2cgns program.

Typically, a user will need to set the following parameters in order to write a TAU-Code grid and solution file to a CGNS data base. Please note that the grid file must always be given whilst the solution file is optional.

```
Para file settings to create CGNS data base entry from a TAU-Code file

  Restart-data prefix: (none)          #name of solution file
  Primary grid filename: tau.grid      #name of TAU-Code primary grid file
  Boundary mapping filename: tau.para  #name of TAU-Code boundary mapping file
  CGNS filename: junk.cgns             #name of cgns data file to create
  Conversion direction: tau2cgns       #convert TAU-Code to CGNS
 Full CGNS error messaging (0/1): 0    #enable output of all warnings and errors
```

Similarly, the following parameter file will allow the creation of a grid and a solution file from a CGNS data base.

```
 Para file to extract unstructured CGNS database data to TAU-Code formatted file

  Restart-data prefix: tau.pval       #name of sol. file read from CGNS data base
  Primary grid filename: tau.grid     #name of prim. grid file read from  CGNS data base
  Boundary mapping filename: tau.para #bmapping file
  CGNS filename: junk.cgns            #name of input CGNS data base file
  Conversion direction: cgns2tau      #convert CGNS to TAU-Code
 Full CGNS error messaging (0/1): 0   #enable full error messaging
```

The structure of the CGNS data file is demonstrated in the figure 28. Most of what is shown in this figure can be easily understood after reference to the CGNS documentation has been made, but there is an important point that needs to be explained. It can be seen that additional data is stored under each boundary marker, and that the additional data is stored under the node name "additional info". The marker information is written here in the sub-node "marker" when a TAU-Code grid file is converted into CGNS format. If this additional marker information is not present in the CGNS data set, then a new set of boundary markers will be written to the TAU-Code grid file. Note that a boundary mapping is written out as a file at the end of a CGNS to TAU-Code conversion. Note also that the boundary mapping file must also be given as input when writing a TAU-Code data set into

a CGNS data base. When checking for failure of the tau2cgns tool in reading a CGNS data base, please ensure that the format of the CGNS data base is equivalent to that shown in figure 13.6. It is important to note that all data is written under one zone, and that the writing of different elements types under different zones is not supported.

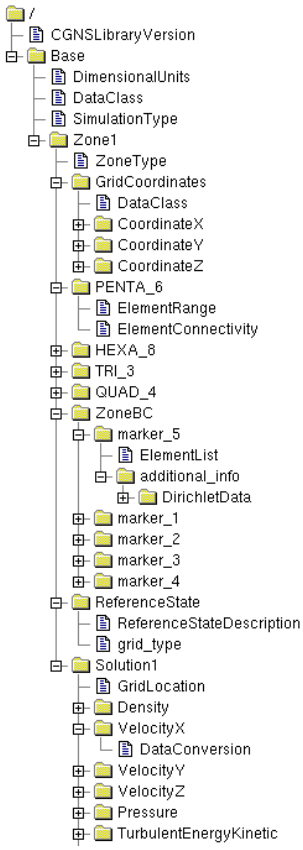


Figure 28: This figure illustrates the internal structure of a CGNS file that has been created using the **tau2cgns** program.

NOTE: After creating a CGNS file from a TAU-Code grid file with a possible solution file, it is a good idea to rewrite the TAU-Code data files from the CGNS file, and to then compare with the original data files. The ADF Viewer program can be downloaded from the CGS website and a TK/TCL based CGNS file viewer can be found on the internet under the ICEMCFD/CGNS webpage:

```
Web site     ->  http://www.icemcfd.com/cgns/Utilities.html
Source Name  -> CGNS Viewer, by C.Lundh
To compile TCL/TK must be present on your system
```

NOTE: At present, the writing of surface data files is not supported.

13.7 The 'Interpolate' program

The interpolate program is used to complement the Dirichlet boundary conditions which has been currently implemented in the TAU-Code solver. Unfortunately the program name 'interpolate' is not exactly accurate at the present time. In order to explain, a full description of the program is required.

In computing certain problems it is usually necessary to precompute a related problem so that proper boundary conditions for the problem of interest are available. A typical example would be computation of flow in a diffuser where the computation is highly dependent on the quality of the flow condition into the diffuser. In this case it is necessary to compute an initial channel problem, and to then interpolate the solution of the channel problem at the outflow boundary to the inflow boundary of the diffuser problem. The interpolation tool can be used to copy the solutions from one boundary plane to another, however at present no interpolation algorithm has been implemented, rather both boundary planes are required to have identical point distributions so that a copy from one data plane to another via the stream library is made. Extensions to this tool to enable interpolate and mapping from one unstructured plane to another are planned but have not yet been implemented.

Additionally, it is possible to compute profiles based on similarity arguments and dimensional analysis for flows which demonstrate self-similarity behavior. For example, in starting a channel profile calculation it is useful to precompute the similarity solutions so that a good approximation of the momentum distribution and the distributions of the turbulence variables can be used to initialize the channel calculation. This reduces the time required to achieve a solution of sufficient quality with which boundary conditions for an additional problem may be generated.

As is customary in the TAU-Code, program control is achieved through use of a parameter file, and this parameter file is listed below.

```
          TAU input file: aa.tau
         TAU output file: bb.tau
           Append string: (none)
Channel parameter file: ch.para
                  Task: 1
```

13.7.1 Input parameters

13.7.2 Channel parameter file

The following parameter list shows a typical parameter file used to generate a channel profile, which can then be used as a boundary condition for the solver. The parameter

```
modify farfield file: <name of modified file>
```

gives a short description of how the interpolate tool is used to generate a useful inflow boundary profile.

```
        Angle alpha (degree): 0
         Angle beta (degree): 0
          Gas constant gamma: 1.4
           Dynamic viscosity: 1.7894e-05
         Kinematic viscosity: 1.5e-05
                Gas constant: 287
              Bulk velocity: 0
             Channel height: 1
      Friction velocity Re: 0
          Bulk velocity Re: 0
                  Sub task: 0
               Include wake: 0
  Tecplot test output (0/1): 0
```

It is planned to add most of the flows which demonstrate similarity behavior to the list of profiles which can be generated by the tools. Examples would include mixing layers and jets, and this work will be completed at some later date.

13.8 The 'ptau3d.subgrids' program

The program is for sequential or parallel (re-)partitioning of hybrid grids. Partitioned hybrid grids are needed to run programs working on primary grids in parallel mode, as e.g. the preprocessor, the adaptation or the grid deformation. If the grid partitions are computed in advance and stored in files the partitioning do not need to be repeated each time one of these program is started. If the partitioning is not performed in advance, the parallel partitioner (included as library) starts automatically with a parallel preprocessor or deformation, etc., which makes parts of the description below relevant for the usage of these programs.
The partitioning contains two major parts, which are a sequential and a parallel partitioner. The switch between both is done automatically in dependence of the number of current processes, i.e. if only one process is started the sequential partitioner is executed, otherwise the parallel one.

At present the parallel and the sequential partitioner do not cover the same functionality with respect to different grid types. Thus, there is no free choice of the usage of both programs in any case. This deficiency will be removed by future developments. The status for the code version 2006.1.0 is as follows:

The first partitioning of the grid (from 1 to N domains) needs to be done with the sequential version. The parallel partitioner is prepared for the initial partitioning, but not yet reliable for all cases.

From this point on, having a set of subgrids, further partitioning steps might be needed, either for changing the number of grid partitions (i.e. changing the number of processes) in a parallel run, or for re-balancing the grid partitions after refinement or de-refinement of the grid.

A grid which has been adapted contains information about the grid hierarchy, which is needed for following de-refinements. The repartitioning of the grid hierarchy is only supported by the parallel grid partitioner. Thus, if further adaptation steps are intended, the parallel partitioner has to be used from this point on. Applying the sequential grid partitioner to an adapted grid leads to a removal of the hierarchy information and thus to the effect that previously refined elements can not be de-refined any more.

A shortcoming of both, the sequential and the parallel partitioner is that periodic grids can not be handled. Thus, for periodic grids the preprocessor and the adaptation have to applied in sequential mode.

`sequential partitioning`

This tool can be started by:

`bin-path/ptau3d.subgrids parameter-file <log-file>`

If a global grid file exists it is read, the grid is partitioned in as many partitions as defined by the parameter setting and the grid partitions are written out into separate files (file name convention is gridfilename_domain_i, with i being the partition number). If no global grid file exists, but files containing the partitioned grids, these are first read and merged into a global grid connectivity, before the partitioning starts. If a flow field solution exists for the given grid (i.e. if a 'Restart data prefix' is defined in the parameter file) the solution files are finally (re-) partitioned according to the new grid partitions and written back to files.

`parallel partitioning`

This partitioning can be started in parallel by:

```
mpirun -np # bin-path/ptau3d.subgrids parameter-file <log-file>
```

The MPI options and the name of the mpirun script depend on the MPI version. Please have a look at the manuals of the MPI version installed on the target machine.
The parallel partitioner consists of the re-partitioning itself and of the migration of points and elements. The modules are designed such, that no process knows the complete grid, which allows partitioning of very large grids by the use of at least so many processors that enough memory is available for each processor to handle its own grid domain (as a thumb rule: approximately 500 MB per 1 million grid points).
The parallel partitioner reads the initial grid partitions onto the different processes before writing out as many load-balanced grid partitions as processes are active, i.e. each process writes out one grid partition. In case that a flow field solution exists (i.e. if a 'Restart data prefix' is defined in the parameter file) the solution is redistributed according to the new grid partitions. In case that the parallel partitioner is used as library linked to another program (like it is linked to the adaptation and executed automatically after a parallel run for load re-balancing) the partitions are handed over to the calling program in memory and no file-IO takes place in between both programs.
If the parallel partitioner is used for re-partitioning from N to M domains (processes) it has to be executed on P processes with P being the maximum of N and M. When enlarging the number of partitions ($M > N$) no extra input is needed, because the default value for the 'Number of primary grid domains' is the current number of process P which equals M. When reducing the number of partitions ($M < N$) the 'Number of primary grid domains' has to be set in the parameter file, because P does not equal the target number.

13.8.1 Sample parameter file

Typical parameter file:

```
-----------------------------------------------------
required parameters
-----------------------------------------------------
  Files/IO ---------------------------------------: -
                         Boundary mapping filename: (thisfile)
                             Primary grid filename: tau.grid
-----------------------------------------------------
preprocessing
-----------------------------------------------------
  partitioning -----------------------------------: -
                    Number of primary grid domains: 1
```

Boundary mapping filename (page 43)
Number of primary grid domains (page 81)
Output level (page 85)

13.9 The 'gather/scatter' utilities

The gather or scatter program allows to scatter a result file into as many result files as processors shall be used for a parallel computation or to gather several result files into one. The usage of the programs is printed to stdout when starting the executable. It is:

`bin-path/gather <para-file> [log-file]`

or

`bin-path/scatter <para-file> [log-file]`

Scattering a single result file has to be performed:

- Scattering of result file(s)

For restarting the flow solver with a different number of processors than in the previous run the result-files need to be scattered according to the new partitions. The input for the scatter program is the prefix of the previous result file(s) (`Restart-data-prefix`) and the decomposed dual grid data (`Grid-prefix`). These filenames are read from the parameter file. If the parameter option for 'automatic parameter update' is enabled these parameters are always kept up to date. This scattering of files is automatically invoked from the preprocessor when a new dual grid for several domains is generated. Thus, the stand-alone scatter program is need only if the preprocessor runs on another file-system as the solver and the restart files are not copied across.

- Gathering decomposed result files

The inputs needed are the decomposed result or surface data files (`Restart-data-prefix` and/or `Surface output filename`). Gathering of files might be needed for other (external) programs reading the set of data or for reducing the number of files. All TAU-Code programs reading field or surface data are able to read gathered or scattered files.

13.10 The 'ncdump' program

The ncdump program is part of the NetCDF-package. This is very useful for the handling of TAU-Code data in order to extract information out of the binary NetCDF-data files. `ncdump FILE` dumps all data in the NetCDF-file 'FILE' to ASCII-format. Most often, more useful is `ncdump -h FILE` which shows the complete header of the NetCDF-file. The header contains all information about the dimensions and the variables and the defined attributes. Thus `ncdump -h GRID` allows e.g. to check for the grid-dimensions. The Restart-files contain beneath the point values all information about the parameters used for the computation, as

well as the date on which the results have been produced. Thus `ncdump -h RESULT` allows to get a lot of information about the data set. For a more detailed description please see the NetCDF documentation, which may be found at
http://www.unidata.ucar.edu/packages/netcdf/docs.html

13.11 The 'patran2tau' program

The patran2tau program converts a unstructured or hybrid grid from patran format to TAU-format. The usage is self-explaining (usage output when starting the binary). Note: because the input binary file format is not platform independent errors during reading the files might occur if not using this program on the same machine which had been used for the generation of the grid file in patran format!

13.12 The 'setup_taugrid' program

The program setup_taugrid is a utility program which can perform various (useful) grid manipulations. Some of the options available are:

- rotate
- mirror
- scale
- stretch
- translate
- grid double
- grid and solution double
- repair periodic grid
- repair boundary plane coordinates
- convert a 2D grid to 2D axisymmetric (periodic)
- convert a 2D grid to 2D axisymmetric (not periodic)
- convert 2D axisymmetric (not periodic) to 3D grid

The computational geometry (when using the TAU-Code) must conform to the following restrictions; the angle of attack is determined by the inclination of the geometry in the x-z plane so that the upper wing surface normal is directed towards the positive z-direction, the span must lie along the y-axis and the flow-direction is in the positive x-direction.

Rotate and/or mirror can be used to reorient the initial mesh according to these restrictions. Scale can be used to obtain the correct dimension (meter) for the grid. Stretch can be used, for example, to extend the farfield boundary of a grid, while translate is self explanatory. Grid double (and solution double if required) doubles the grid in a user defined direction (x, y or z). The grid doubling is achieved by copying the grid (via translation or mirroring) and then fusing these two grids together. Hence the original grid must be periodic in the direction of translation.

Periodic grids can be repaired before the preprocessing stage. Any small errors in the coordinates of periodic pairs has a negative effect on the quality of the dual grids. This function repairs the coordinates of periodic points in a grid to machine accuracy by projecting the periodic points onto user defined planes. It is recommended that this check be made for all periodic grids before preprocessing. Similarly, the coordinates of the points on a planar boundary can be corrected to machine accuracy by projecting the points onto a user defined plane. This is recommended, for example, before doubling a grid using the symmetry plane.

For axisymmetric cases containing swirl in the circumferential direction, a periodic axisymmetric grid should be created. However, if no swirl is present, the use of a non-periodic axisymmetric grid affords several advantages: the facility for grid adaptation, faster solver operation, and increased accuracy when using structured cells at the axisymmetry axis. The axisymmetry options in the TAU-Code solver are designed for use with non-periodic axisymmetric grids. Axisymmetric non-periodic grids created by setup_taugrid will be topologically identical to the input grid, but may be translated to center the designated base plane over the axisymmetry axis. For both periodic and non-periodic axisymmetry grids, the axisymmetry axis is assumed to pass through the origin.

The generation of a 3D grid based on a (not periodic) 2D grid is primarily designed for two purposes: the coupling with a code modeling the transport of energy via gas radiation and for postprocessing the data with line of sight methods to compare with optical measurements. A 3D solution related to a solution on the given 2D grid can therefore be generated together with the grid. To enable an access to data calculated by the radiation transport model on the 3d grid, the integration of a value on the 3D grid and its appending to the 2d solution is available as well. Flow solver calculations on the 3D grid for consistency checks are not investigated, but have to face the following problem: when starting with triangles at the symmetry axis, the 3d grid will probably not pass the grid test without modification. The reason is the remeshing of neighbored pyramids, which enables the local adaptation, but disturbs the symmetry of these special grids.

The usage for all options is self explanatory. A list of options is printed to stdout when starting the executable.

13.13 The 'smooth_taugrid' program

The program *smooth_taugrid* is developed to improve the quality of the unstructured part of a hybrid mesh. The main features are

- repairing of negative elements,
- to improve the quality of elements,
- to create anisotropic meshes.

The modifications are always locally and the surface is untouched (except symmetry planes which are optionally allowed). The modified elements are prisms for 2D meshes and tetraeder and pyramids for 3D meshes.
The program *smooth_taugrid* is controlled by command line options which will be explained in the section 13.13.2. In the simplest way it can be executed a in shell by
`bin-path/smooth_taugrid -i <grid-file>`

The program will run one loop of the default enabled optimization routines. The output is described in the next section. For a more optimal use of the program the command line options and typical examples are shown in 13.13.2 and 13.13.3, respectively.

13.13.1 Output of the quality

The typical output beside the program status information is the output of the quality and of some mesh properties. This output is given before and after the optimization if no other command line option is set (e.g. *–Statistic*) A typical output is:

```
quality measure:    mean q = 0.827956 +- 2.15335e-05
                    min q = 0.00032683 (element 40442299)
                    max q = 0.999962 (element 6915655)
                    N(q<0.2) = 170799, N(q<0.1) = 108001
                    N(q<=0.0)= 0
dihedral angle:     worst alpha = 0.000476578 deg (element 40442299)
                    N(alpha<5deg) = 152117, N(alpha<1deg) = 73314

no of points        = 21991175
no of tetrahedrons  = 40725282
```

The first part of the message shows some statistical values of the mean ratio quality measure of equation (48).

$$q_i = \begin{cases} 4\sqrt{3} \cdot \dfrac{A_i}{\sum_{j=1}^{3} l_{ij}^2} & \text{for 2D (triangles)} \\ 12 \cdot \operatorname{sign}(V_i) \cdot \dfrac{\sqrt[3]{(3V_i)^2}}{\sum_{j=1}^{6} l_{ij}^2} & \text{for 3D (tetrahedron)} \end{cases}, \tag{48}$$

First the mean value of the quality and its standard deviation is listed. The mean value should be as high as possible if no information of the flow is given. The extremal quality values and the element number is given in the second and third line. In the following line the number of elements with a quality less the 0.2, 0.1 and 0. (invalid elements) are shown, respectively. If $N(q <= 0.0)$ not equal zero TAU is not able to compute on it because there were elements inside the mesh which have a negative volume or no volume. The other values should be as low as possible, for $N(q < 0.1)$ zero. Values of $N(q < 0.1)$ larger than zero indicates that the mesh is chopped or that for other reasons bad shaped tetrahedron are included in the mesh.
The second part shows information about the more intuitive quality measure of the minimal dihedral angle (49).

$$\xi_i = \min_j(\phi_{ij}, 180^\circ - \phi_{ij}) \tag{49}$$

Here the worst dihedral angle is shown and the number of its element. In the next line the number of elements with a minimal dihedral angle less than five and one degree is given. Similar to quality measure block the number of elements with a minimal dihedral angle less than one degree should be zero, at least for isotropic mesh improvement. Note that the dihedral angle for elements with negative volumes are always positive.
The last two lines give information about the number of grid points and the number of tetrahedral elements. Both values are only from interest if the grid is manipulated with *smooth_taugrid*.

13.13.2 Command line options

The call of *smooth_taugrid* in the introduction of section 13.13 was only a simple example. There exists several command line options to control the program. The following table gives an overview about these options.

Table 9: Available keywords and their function

Option	Description	Default
`--AnisotropicMeasure` *name*	Use an anisotropic metric for the quality. For this a solution file is needed in which the scalar *name* is stored. The possible variables are shown in the table of section 7.3.4. Useful scalars are the local Mach number *mach* or the pressure *p*.	
`--DihedralAngleLimit` ξ_{min}	Lower bound for the dihedral angle ξ within the optimization process.	
`--eps_dl` $\epsilon_{\Delta l}$	Error bound for the maximal movement.	1e-6
`--force`	Force *smooth_taugrid* to work with meshes which have negative volumes. (only 3D)	not enabled
`--GoalFunction` *fctname*	Select the goal function. *fctname* can be:	min
	min for $\min_i q_i$	
	ls for $1/N \cdot \sqrt{\sum_i q_i^2}$	
	inv_ls for $N/\sqrt{\sum_i q_i^{-2}})$	
`-h, -?, --help`	Output of a short help.	
`-i` *gridfile*	Name of the input TAU mesh file.	
`-is` *solutionfile*	Name of TAU solution file which is related to the grid file. For the quality check the solution is only needed if an anisotropic metric should be applied.	
`--ImproveDihedralAngle`	Improve always the minimal dihedral angle.	
`--ImproveMeanQuality`	Improve always the mean quality.	
`--MaxInnerIterations` n	Maximal number of iteration steps of the optimizer of the point movement algorithm.	6

continued on next page

Option	Description	Default
`--MaxLaplaceIterations` n	Maximal number of iterations where Laplacian. smoothing is allowed.	2 (2D), 0 (3D)
`--MaxOuterIterations` n	Maximal number of optimization loops.	1
`--MaxSmoothingIterations` n	Maximal number of smoothing iteration.	4
`--ModifyPyramids`	Allow to modify pyramids.	disabled
`--[Not]AllowEdgeCollapsing`	(Not) Allow edge collapsing. Changes also the number of nodes! Therefore if afterwards a restart is applied the solution has to be interpolated with *smooth_taugrid*.	not allowed
`--[Not]AllowEdgeSwapping`	(Not) Allow edge swapping.	allowed
`--[Not]AllowFaceSwapping`	(Not) Allow face swapping.	allowed
`--MaxSwappingIterations` n	Maximal number of swapping/collapsing (edge/face) iterations.	20
`-o` *gridfile*	Name of output TAU grid file. This keyword is optional. If it is not specified the mesh will be written back in a file with the suffix *smooth_taugrid*.	
`-os` *solutionfile*	Name of the output TAU solution file.	
`--OnlyRepairNegElements`	Abort if all elements are valid.	
`--[Only]Statistic`	Extended output of statistics (in files and on screen). With the option `--OnlyStatistic` the program exits without optimizing the mesh.	
`--OptimizationLimit` ϵ_q	Optimize only elements with a minimal quality less then the limit ($q < \epsilon_q$).	1.0
`--RestrictionLevel` n	Restriction level for optimization: **0** Improve the minimal quality **1** Improve the minimal quality and dihedral angle **2** Improve the minimal quality, mean quality, and dihedral angle	0

continued on next page

Option	Description	Default
`--SkipTauGridTest`	Skip the Tau grid test at the end of the modification of negative elements.	
`--SymmetryMarkers` $m_1, m_2, \ldots$	Markers of the symmetry separated by commas. (no space between them!). Only for 3D meshes.	
`--SymmetryPlane` n	Orientation of the symmetry plane(s). Case of n the plane is parallel to **0** none of the following planes **1** xy-plane **2** xz-plane **3** yz-plane A value of zero disables manipulation on the symmetry plane. Only for 3D meshes.	0
`--WriteBadElements` q	Write elements with a quality less then q in a netcdf-grid-file. If q has the suffix deg (e.g. 1deg) the dihedral angle ξ is used for the selection of the elements instead of the quality q.	
`--WriteBadElementsWithSur` q	Like `--WriteBadElements` but writes the selected elements with its surroundings. This option is sometimes useful to see why elements have a bad shape and why this element couldn't be modified.	
`--WriteQuality`	Write the local minimal quality and the local minimal dihedral angel in a TAU-like NetCDF solution file. Because the solution of TAU is point and not element based the program will write out the minimal quality and the minimal dihedral angle of the node surroundings. Elements with no quality information (e.g. hexahedrons, pyramids) were set to a default quality of 2.0 and a default dihedral angle of 180 degree.	

continued on next page

Option	Description	Default

Additionally the program *smooth_taugrid* handles following signals:

USR1 The intermediate grid will be written. Afterwards *smooth_taugrid* continuous the optimization process. Note that an intermediate interpolated solution will not be written.

USR2 Stops the optimization process and terminates the program regularly with all output files, especially the optimized grid and solution file. If a solution file should be written the termination could take a longer time due to the interpolation process.

13.13.3 Typical examples

Isotropic mesh optimization In the following four examples are given to optimize the mesh.

save way:

Features:

- Number of points kept constant.
- Improvement of minimal quality
- Improvement of mean quality
- Improvement dihedral angle.
- Well shaped elements with $q > 0.5$ will be skipped.

Command:

```
smooth_taugrid -i input_grid_file -o output_grid_file \
   --OptimizationLimit 0.5 --MaxOuterIterations 4 \
   --RestrictionLevel 2 --Statistic
```

typical way 1:

Features:

- Number of points kept constant.
- Improvement of minimal quality
- Improvement of mean quality

Command:

```
smooth_taugrid -i input_grid_file -o output_grid_file \
   --MaxOuterIterations 4 --ImproveMeanQuality --Statistic
```

typical way 2:

Features:

- Improvement of minimal quality
- Improvement of mean quality
- Modify pyramids

Command:

```
smooth_taugrid -i input_grid_file -o output_grid_file \
   --MaxOuterIterations 4 --ImproveMeanQuality \
   --AllowEdgeCollapsing --Statistic --ModifyPyramids
```

most aggressive way:

Features:

- Improvement of minimal quality

Command:

```
smooth_taugrid -i input_grid_file -o output_grid_file \
   --AllowEdgeCollapsing --MaxOuterIterations 6 --Statistic \
   --ModifyPyramids
```

repairing of negative elements:

Features:

- Repairing of negative elements and some bad shape elements

Command:

```
smooth_taugrid -i input_grid_file -o output_grid_file \
   --MaxOuterIterations 4 --Statistic --ModifyPyramids\
   --OptimizationLimit 0.1 --force
```

only statistic:

Features:

- Output of the initial quality without modifying the mesh
- Output of bad shaped elements (elements with dihedral angle less then 1°)
- Output of quality as solution file

Command:

```
smooth_taugrid -i input_grid_file -o output_grid_file \
   --OnlyStatistic --WriteQuality --WriteBadElements 1deg
```

Optionally if a solution was given it can be interpolated on the new grid by adding the options `-is` *input_solution_file* and `-os` *output_solution_file*.
For three-dimensional grids with symmetry planes the options `--SymmetryMarkers` and `--SymmetryPlane` can be set to enable manipulations on the symmetry plane.
The option `--OptimizationLimit` could enhance the speed of the program if the limit is set to lower values than unity. In this case elements with $q > \epsilon_q$ are neglected in the optimization process.

Anisotropic mesh optimization To use an anisotropic metric following points have to be satisfied:

1. The command line option `--AnisotropicMeasure` *name* must be set. The possible names are defined in section 7.3.4.

2. A solution file must be specified with the option `-is` *solution_file* where inside the scalar variable exits from which the new metric will be derived.

3. `--ImproveDihedralAngle` should not be set. Otherwise the stretching of elements is forbidden due to this restriction.

4. `--RestrictionLevel` should not be set to higher value than zero. Otherwise the stretching of elements is blocked by the restriction.

5. For three-dimensional grids with symmetry planes the options `--SymmetryMarkers` and `--SymmetryPlane` should be set to enable stretched elements on the symmetry plane.

To avoid too flat elements the option `--DihedralAngleLimit` ξ_{min} can be set, e.g. with $\xi_{min} = 1$.

13.13.4 Known limitation

- Chimera grids are not supported by the program.
- Single domains of a partitioned grid are not supported by the program.
- Face Swapping/Edge Collapsing is only apply able in 3D
- Informations of the TAU adaptation get lost.

13.14 The 'sort_cut' program

The solver allows writing of cut-plane data in a number of predefined sections. In case of parallel computations the cut-plane data may be split into several domains. The `Plane output period` is also user defined. Depending on the number of planes, the number of outputs and the number of domains a lot of cut-plane files are written All these files contain

a series of ASCII values, which may be `x_y_z_cp_cf` (or other values) per line. This data is not sorted, such that it can not be used for polygon-plots (with standard 2D plot-utilities). The 'sort_cut' program permits postprocessing of the cut-plane data. The postprocessing is performed for all files at once (different planes at different iteration numbers). Data from different domains is gathered automatically together. The data is sorted for polygon plots. In addition several manipulations can be performed, as e.g. extracting specific values, splitting values depending on surface markers and/or scaling of values. The different possibilities can be chosen by command line options. The usage and the available 0 options are printed to stdout when starting the program.

13.15 The 'tau2plt' program

The tau2plt program converts data (grid and solution) from the TAU-format into a desired visualization format used by Tecplot, or optionally into other formats (currently supported: Ensight, Fieldview). Additionally it is possible to convert only specific information from either the primary grid or the solution files. This may be in the prismatic layer around an airfoil or by specifying a bounding box for visualization a small part of whole flow domain. Other features are the automatic gathering of partitioned primary grids or solution files. A sample parameter file is printed when starting the executable with a dummy file.

Valid call of the 'tau2plt' program is:

```
bin-path/tau2plt <para-file>
```

13.15.1 Sample parameter file

Typical parameter file:

```
------------------------------------------------------
required parameters
------------------------------------------------------
  Files/IO ---------------------------------------: -
                          Boundary mapping filename: (thisfile)
                              Primary grid filename: tau.grid
------------------------------------------------------
io
------------------------------------------------------
  Grid/Solution ----------------------------------: -
                                Restart-data prefix: tau.solution
------------------------------------------------------
tau2plt
------------------------------------------------------
  Volume element options -------------------------: -
```

```
                              Specify prism list: (none)
                           Specify tetrahedral list: (none)
                           Volume data output (0/1): 1
  Surface element options ------------------------: -
           Create surface zone for boundary marker: 1  2
                         Surface data output (0/1): 1
  Output Control ---------------------------------: -
                                      Ascii (0/1): 0
                                    Output format: tecplot
  Other ------------------------------------------: -
                                    Variable list: cp_mach_Ptot
```

13.15.2 Input parameters

Boundary mapping filename	(page 43)
Output format	(page 339)
Output level	(page 85)
Primary grid filename	(page 90)
Restart-data prefix	(page 100)

`Output format:` character string, default tecplot

- In analysis both "ascii" and "tecplot" options are available at present.
- In tau2plt: Set the conversion format which can be the following macros
  - *ensight6* $\cdots$ use for ENSIGHT version 6
  - *ensight_gold* $\cdots$ use for ENSIGHT GOLD
  - *fieldview* $\cdots$ use for FIELDVIEW (binary only), NOTE: There are no surface data files allowed
  - *opendx* $\cdots$ use for OPENDX
  - *tecplot6* $\cdots$ use for TECPLOT version 6
  - *tecplot* $\cdots$ use for TECPLOT 7 or higher (default)
- particle tracer: The tracer can handle the following output formats
  - *tecplot* $\cdots$ use for TECPLOT 7 or higher (default)
  - *opendx* $\cdots$ use for OPENDX
  - *vtk* $\cdots$ use for Visualization tool kit

`Amif aero coordinate frame:` integer, default 1

- Specify the Aero coordinate frame ID for AMIF files.

`Ascii (0/1):` integer, default 0

- Creates an ASCII output for Tecplot and Ensight only.

`BDF basic coordinate frame for frame CD:` character string, default (none)

- Specify the identification number for a user-specified coordinate frame.

`BDF basic coordinate frame for frame CP:` character string, default (none)

- Specify the identification number for a user-specified coordinate frame.

`BDF coordinate frame CD:` character string, default (none)

- Specify the identification number of coordinate frame CD.

`BDF coordinate frame CP:` character string, default (none)

- Specify the identification number of coordinate frame CP.

`BDF field format (8/16):` integer, default 8

- Specify the accuracy in the BDF-file of the grid-point coordinates.

`BDF points defining coordinate frame CD:` array-float, default {0, 0, 0, 0, 0, 1, 1, 0, 1}

- Three points used to define the user-specified coordinate frame for frame CD.

`BDF points defining coordinate frame CP:` array-float, default {0, 0, 0, 0, 0, 1, 1, 0, 1}

- Three points used to define the user-specified coordinate frame for frame CP.

`BDF scale data entry ID:` character string, default (none)

- Specify the identification number for a scale-data entry.

`BDF scale reference lengths:` array-float, default {1, 1, 1}

- Reference lengths for the scale-data entry.

`Bounding x coordinate range (3D):` character string, default (none)

- Bounding x coordinate range from `min` < x < `max`. Be aware to handle the right sequence from minimum to maximum. Otherwise the whole grid is taken again. If someone leaves a direction with (none) but have specified the other one's the maximum range will be automatically set for the nonspecified direction.

`Bounding y coordinate range (3D):` character string, default (none)

- Bounding y coordinate range from `min` < y < `max`

`Bounding z coordinate range (3D):` character string, default (none)

- Bounding z coordinate range from `min` < z < `max`

`Chimera component output (Tecplot, Ensight Gold) (0/1):` integer, default 0

- If this switch is turned on, each chimera block is exported to a separate Tecplot zone or Ensight Gold part.

`Create one boundary:` integer, default 0

- Does not split between different boundary types. Merges each boundary to one single boundary.

`Create surface zone for boundary marker:` character string, default (none)

- Creates surface zones for specified boundary markers, e.g. 1,3,5 → boundaries 1, 3 and 5 are in the output file.

`Create surface zone for quadrilateral elements:` character string, default (none)

- Creates an output of quadrilateral elements which are entered via their ID number. NOTE: The maximum list size is 10000.

`Create surface zone for surface element:` character string, default (none)

- Creates additional zones for different surface elements, e.g. triangles.

`Create surface zone for triangular elements:` character string, default (none)

- Creates an output of triangular elements which are entered via their ID number. NOTE: The maximum list size is 10000.

`Element types for zone:` character string, default (none)

- The output contains element zone(s) only for volume elements declared in the list (0 = tetrahedra, 1 = pyramids, 2 = prism, 3 = hexahedra).

`Old grid size:` integer, default 0

- The point-size of a previous grid (before adaption) can be entered.

`One zone for all volume elements:` integer, default 0

- Some grids contain different volume types, e.g. prisms and tetrahedra. Turning on the flag inhibits the splitting between volume elements.

`Percentage of BL reduction (0-100):` integer, default 1

- This option was introduced for reducing the high amount of grid points at viscous flows which arise at three dimensional complex computations. In case of a vortex calculation someone is not too much interested on near wall effects instead of a good resolution in vortex generation regions most outside or behind a wing. The most points are paid for the boundary layer region which is in relation to the TAU-Code a prismatic layer around the surface. With the parameter you can specify the percentage of the prismatic boundary layer reduction in the converted solution data file.

`Precision:` integer, default 9

- The precision describes the decimal places used for an ASCII output. For example, 9 will write for each value 9 decimal places.

`Separate original and adapted file:` integer, default 0

- Create separate zones for original and adapted volume elements (Ensight 6 format only!).

`Specify hexaeder list:` character string, default (none)

- Creates an output of hexahedral elements which are entered via their ID number. NOTE: The maximum list size is 10000.

`Specify prism list:` character string, default (none)

- Creates an output of prismatic elements which are entered via their ID number. NOTE: The maximum list size is 10000.

`Specify pyramid list:` character string, default (none)

- Creates an output of pyramid elements which are entered via their ID number. NOTE: The maximum list size is 10000.

`Specify tetrahedral list:` character string, default (none)

- This option provides an output of single volume elements which have to be known with their ID number. For example, entering 2,3,4,100,110 into the tetrahedral list prints to the output file the five volume elements with the ID number 2,3,4,100,110. NOTE: The maximum list size is 10000.

`Surface data output (0/1):` integer, default 1

- If this flag is turned on (1) it looks automatically for a surface output file written by the solver and additionally converts this to the chosen output format. Whenever the parameter is turned on and the flag *Volume data output (0/1): 0* is turned off (0) it first seeks for an extra surface file **.surface.pval.** and when found it converts the file to the chosen format otherwise the surface data will be extracted from the restart data file (volume data).

`Swap yz:` integer, default 0

- Switch y and z coordinates and velocities.

`Title of output file:` character string, default (none)

- The specified title appears on top of the converted file.

`Use node ID:` integer, default 0

- Keeps the node ID numbering in the output file.

`Variable list:` character string, default (none)

- Reduce solution to only the variables in list (e.g. cp_mach). The keywords are the same as used for the solver (`Field output values: cp_mach`).

`Volume data output (0/1):` integer, default 1

- The flag declares if a volume grid is desirable or not. Whenever the flag is turned off the boundary surfaces are written only which are extracted form the volume data.

The old style of 'tau2plt' is still available. A short description is printed to stdout when starting the executable without parameters. That feature will disappear from time to time to set work on the TAU-Code for his Python functionality. If there is still a need for the command line parameters, it can be started with:

```
bin-path/tau2plt flags netCDFgrid
```

```
bin-path/tau2plt flags netCDFgrid netCDFpointdata
```

Where possible flags `flags` are:

- `-g`
  grid file only

Bounding box

- `-X <min, max>` (3D)
  Bounding x coordinate range to `min` $< x <$ `max`
- `-Y <min, max>` (3D)
  Bounding y coordinate range to `min` $< y <$ `max`
- `-Z <min, max>` (3D)
  Bounding z coordinate range to `min` $< z <$ `max`

Volume element options

- `-V`
  Do not write volume elements
- `-x`
  Create one volume zone (Tecplot, Ensight Gold)

Surface element options

- `-S`
  Do not write surface elements
- `-m <list>`
  Create surface zone(s) only for markers in list
- `-b`
  Create one boundary zone (Tecplot, Ensight Gold)

Output formats

- `-6`
  Create output for Tecplot version 6
- `-E`
  Create output for Ensight version 6 (ascii)
- `-G`
  Create output for Ensight Gold (ascii)
- `-F`
  Create output for Fieldview (binary)
- `-a`
  Create Ascii output (Tecplot and Ensight only)
- `-p <precision>`
  Tecplot ascii file precision (default 9)

Other

- `-r <list>`
  reduce solution to only the variables in list (e.g. p_v means that the pressure and the velocities will be kept)
- `-w`
  Switch y and z coordinates and velocities
- `-n`
  write tau node-id's (Tecplot, Ensight Gold)

13.16 The 'pltdata' library

This is a library which can be used with the python interface of the TAU-Code. The functionality which can be addressed by python scripts covers a lot from the functionality provided by 'tau2plt'. In addition this library can be used in parallel mode and provides extra functionality due to the flexibility achieved by python scripting. This allows not only to read in a dataset for conversion to a visualization software as a stand alone program, it also allow for extraction of data and conversion in between other parts of the TAU-Code running under python. One example is that it is possible to output data in a native format of a visualizer during the solver iterations without the need for intermediate data-IO via NetCDF-files.

The idea behind this library is to provide basic functions to select elements (tetrahedras, prisms, hexahedras, pyramids, surface triangles and quadrilaterals) and to reduce the complete dataset according to the selection. This is explained below in more detail by some examples.

```
tau_plt_select_stris('zone-1')
tau_plt_reduce_dataset('zone-1')
tau_plt_select_squads('zone-2')
tau_plt_reduce_dataset('zone-2')
tau_plt_tecplot_zones('filename.plt')
```

With this script a Tecplot file is generated containing all surface triangle in 'zone-1' and all surface quads in 'zone-2'. Putting all elements in the same zone would be achieved by

```
tau_plt_select_stris('zone-1')
tau_plt_select_squads('zone-1')
tau_plt_reduce_dataset('zone-1')
tau_plt_tecplot_zones('filename.plt')
```

After reduction of a dataset select functions can be applied again.

```
tau_plt_select_stris('zone-1')
tau_plt_reduce_dataset('zone-1')
tau_plt_unselect_all('zone-1')
tau_plt_select_treatment('zone-1', 'euler wall')
tau_plt_tecplot_zones('filename.plt')
```

In this example all surface triangles of the boundaries to which the treatment 'euler wall' is applied are written to a Tecplot zone. 'tau_plt_unselect_all()' is needed because after dataset reduction all elements in this dataset are flagged as being selected. In order to output only some of them all elements are first unselected and then a select function for the treatment is applied.

The examples above show that a lot of combinations are possible which provides maximum

flexibility. Elements can be selected by the element number or by other criteria (e.g. by_angle, bad_volume, neg_volume, elements_in_sphere, refined_elements, etc.). Surface elements can be selected by bdryname, partname, treatment or marker. Additional criteria will be build in on request.

In addition to the select/reduce function grid cuts can be extracted by 'tau_plt_extract_plane(...)' at any position inside the grid. After generation of one or more reduced datasets variables can be assigned to the datasets.

Note, if many different datasets are extracted reduce the data after selection before adding a new dataset (by choosing a new name, see above 'zone-2'). This is of advantage to avoid a considerable memory overhead because each new dataset contain some mapping arrays of the current size of the grid, which is the complete grid size before reduction.

All interfaces available for the user can be found in the file 'tau_python.i'. More examples for usage can be found in sample scripts.

13.17 Intermesh

13.17.1 Introduction

Apart from grid deformation or rigid body motion, it is possible to realize motion of the hole grid or parts of it by calculating the grid point coordinates from the point coordinates of pregenerated grids.

If these pregenerated grids represent different states of a motion and these states of the motion is identified by a motion parameter, the coordinates of the new grid are interpolated with respect to a parameter for the new grid.

To provide the correct point coordinates of the pregenerated grids for the interpolation, these grids must be of the same structure, i.e. the must have the same number of points, the same connectivity, and the same sequence of global point ids. The term structure in this context should not mislead to structured grids, since the interpolation of point coordinates is independent of any block structure of grids. Thus it is possible to use the tool intermesh for interpolating the point coordinates of both structured and unstructured grids. It is advisable to generate the pregenerated grids by grid deformation, at which the structure of the grid as described above is preserved.

By specifying a rule for the calculation of the motion state parameter, it is possible to realize a transient motion for a time accurate simulation by interpolating the new grid for every time step.

13.17.2 Description

The addon tool intermesh reads a number of given pregenerated grids and the corresponding motion state parameters. These parameters must be provided as input for every grid block of these grids. Furthermore rules for the calculation of the motion state parameters for the grid blocks of the new mesh must be given. There are several ways to provide these rules to realize different motions of the grid blocks. The ways of specifying these rules and the algorithms to interpolate the point coordinates are now explained in more detail.

13.17.3 Interpolation of point coordinates

General description To interpolate the grid point coordinates every grid point is associated to the motion state parameter of the grid block. The first step is to set up a new grid by using the given data of one of the pregenerated grids. Thus the new grid is at first a copy of this pregenerated grid. Afterwards the new coordinates are interpolated from the point coordinates of the corresponding points (points with the same global id) in the pregenerated grids with respect to the motion state parameter of the grid block containing the point. If the number of given pregenerated grids is is bigger than the necessary number of grids, the interpolation is only performed from the grids where the difference between motion state parameter of the block in new grid and the pregenerated grid is minimal. Since these parameters can be different for the blocks in all the grids, intermesh reads the most appropriate pregenerated grids for every block.

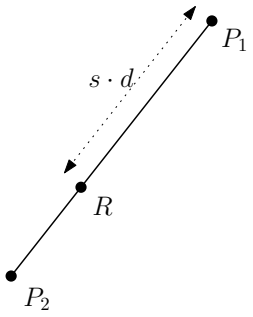


Figure 29: Points on a line

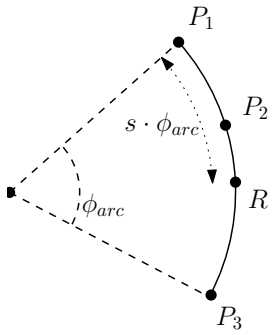


Figure 30: Points on a circular arc

Coordinate interpolation for translatory motion To realize a translatory motion of the hole grid or parts of it, at least two pregenerated grids have to be provided. It is assumed that every grid point moves on a straight line (see Figure 29). This line can be constructed from the corresponding points of the two pregenerated grids P_1 and P_2 :

$$\vec{R} = \vec{P}_1 + s(\vec{P}_2 - \vec{P}_1) \tag{50}$$

The coordinates of the corresponding point of the new grid (marked by subscript $_N$) can then be calculated as the vector $\vec{R}$ by calculating the scalar variable s by interpolation of the motion state parameters m :

$$s = \frac{m_N - m_1}{m_2 - m_1} \tag{51}$$

The coefficient s can be interpreted as a distance coefficient for the movement of the interpolated point R from point P_1 to P_2 .

Coordinate interpolation for rotatory motion For a rotatory motion of the grid or of parts of it, e.g. pitching oscillation of an airfoil or the deflection of a flap it is assumed that every grid point moves on a circular arc (see Figure 30). This arc can be constructed by using the corresponding points of three pregenerated grids $\vec{P}_1$, $\vec{P}_2$ and $\vec{P}_3$. These three points are sorted with respect to their motion state parameter so that the point associated to the intermediate value is the intermediate point on the circular arc. The center point $\vec{C}$ of the arc is defined by the intersection of three planes that can be constructed by the points $\vec{P}_1$, $\vec{P}_2$, $\vec{P}_3$. The plane E_1 is the plane in spanned by the tree points. The plane E_2 (E_3) is constructed as an intermediate plane between $\vec{P}_1$ and $\vec{P}_2$ ($\vec{P}_2$ and $\vec{P}_3$) with a normal vector parallel to the vector $\vec{P}_2 - \vec{P}_1$ ($\vec{P}_3 - \vec{P}_2$). The aperture angle ϕ_{arc} of the arc is and the unit normal vector $\vec{n}_{arc}$ are calculated by:

$$\alpha_{arc} = acos\left(\frac{(\vec{P}_1 - \vec{C}) \cdot (\vec{P}_3 - \vec{C})}{\left|\vec{P}_1 - \vec{C}\right| \cdot \left|(\vec{P}_3 - \vec{C})\right|}\right) \tag{52}$$

$$\vec{n}_{arc} = \frac{(\vec{P}_1 - \vec{C}) \cdot (\vec{P}_3 - \vec{C})}{\left|\vec{P}_1 - \vec{C}\right| \cdot \left|\vec{P}_3 - \vec{C}\right|} \tag{53}$$

For the rotatory motion the coefficient s can be interpreted as an angular coefficient for the movement of the interpolated point R from point P_1 to P_3 .
The coordinates of the point R can now be calculated by using the Euler parameters:

$$e_0 = cos\left(\frac{\phi}{2}\right) \tag{54}$$

$$\vec{e} = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix} = \vec{n}_{arc} \cdot sin\left(\frac{\phi}{2}\right) \tag{55}$$

Using the relation of the Euler parameters to the rotation angle

$$cos(\phi) = e_0^2 - e_1^2 - e_2^2 - e_3^2 \tag{56}$$

the rotation of a vector r_1 from point C to P_1 by an angle ϕ towards point P_3 is calculated by:

$$\vec{r}_N = \vec{r}_1 cos\,(\phi) + 2\vec{e}\,(\vec{e} \cdot \vec{r}_1) + (\vec{r}_1 \times \vec{n}_{arc})\, sin\,(\phi) \tag{57}$$

Therefore the new coordinates of a grid point can be calculated by:

$$\vec{R} = \vec{C} + \vec{r}_N \tag{58}$$

13.17.4 Calculation of the motion state parameter

General description To realize a transient motion of the grid for a time accurate simulation, the motion state parameters of the blocks of the new grid have to be time dependent. There are several ways to specify this dependency which are described in detail below. The value of physical time is retrieved from a restart file from the TAU solver. If no restart file is available, the value of the physical time is set to $t = 0$ by default. To provide an interpolated grid for a steady simulation, the same way of specifying the time dependency of the motion state parameter has to be followed, at which the default value of physical time $t = 0$ is used.

Piecewise linear time dependency The piecewise linear time dependency is specified by a series of pairs of values of physical time and motion state parameter. These pairs have to be sorted for an ascending value of physical time. For values of physical time between the input values the motion state parameter is calculated by linear interpolation between the two most nearby pairs of value, where the value of physical time of one of these pairs must be smaller and the value of the other pair must be larger than the value of current physical time. If the value current physical time is smaller than the smallest given physical time or larger than the largest given physical time, the value of the motion state parameter of the pair with the most nearby (i.e. the smallest / largest) value of physical time is used without interpolation. Since the input values are treated as pairs of values, the number of input values must be even.

Polynomial time dependency The polynomial time dependency is specified by a series of coefficients of a polynomial function to calculate the motion state parameter of the new grid. The motion state parameter m is calculated by:

$$m = \sum_{i=0}^{n} a_i \cdot t^i \tag{59}$$

where the coefficients a_i are the provided input coefficients.

Periodic time dependency The periodic time dependency is specified by a series of coefficients that are used as coefficients of a fourier series. The motion state parameter is calculated by:

$$m = \frac{a_0}{2} + \sum_{i=1}^{n} (a_i \cdot \cos(\omega t) + b_i \cdot \sin(\omega t)) \tag{60}$$

where the coefficient a_i, b_i must be given in the following sequence: a_0, a_1, b_1, a_2, b_2, ... To calculate the frequency ω of the periodic behavior of the motion state parameter, additional inputs of the reduced frequency, the reference length and the number of timesteps per period are needed. The reference speed is retrieved from a restart file of the TAU solver. Since the input values are treated as coefficients of a fourier series of the notation described above, the number of input values must be uneven.

Time dependency by kriging interpolation To provide a smoother variation of the motion state parameter than realized by the piecewise linear time dependency, there is the possibility to use a kriging interpolation between several given pairs of values of physical time and motion state parameter. Kriging interpolation provides a smooth, spline-like behavior without the tendency to result in large oscillations at clustered sample points. Since the input values are treated as pairs of values, the number of input values must be even.

13.17.5 Parallelization

As described above, it is advisable to generate the needed grids by deformation of one grid, to make sure all pregenerated grids have the same structure. In case of large grids and therefore the necessity of performing a simulation on multiple processors, the grid interpolation can also be performed on multiple processors, so that the interpolated grid does not need to be partitioned afterwards.

The number of domains of each pregenerated grids has to equal the number of started processes, and the number of domains of the new grid will also equal that number of processes. There are two different approaches to generated the needed subdivided grids to interpolate a new one from.

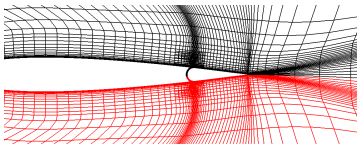


Figure 31: Flap 0 deg, Subgrid structure A

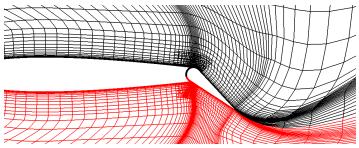


Figure 32: Flap 40 deg, Subgrid structure A

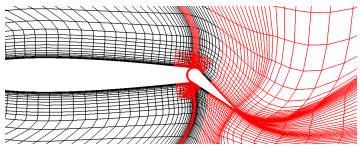


Figure 33: Flap 40 deg, Subgrid structure B

The first is to subdivide a pregenerated grid, and afterwards performing a deformation on all the domains and saving the deformed domains as part of a second pregenerated grid. In

this case both grids are equally partitioned as shown in Figure 31 and Figure 32. The second way is to perform a deformation on a grid, saving he grid as a second pregenerated grid and performing the subdivision afterwards on both grids. In this case it is not guaranteed, that both grids are equally partitioned (see Figure 33), i.e. corresponding points with the same global_id may reside in domains at different processors. In this case a parallel interpolation of the point coordinates is not possible and the interpolation of the point coordinates is performed sequentially. The coordinates of all corresponding points of the pregenerated grids are gathered to interpolate the new coordinates. Because of the communication between the processes to gather the coordinates, this procedure is actually slower than the interpolation of the same grid performed on one single processor. However the memory consumption necessary for large grids may result in the necessity to perform a distributed grid interpolation. Note: The boundaries of the domains of the new grid are taken over from the first given pregenerated grid.

In case of equally partitioned grids a parallel interpolation of the point coordinates is possible, which should result in less computational time compared to the interpolation on a single processor.

13.17.6 Input options

The input options are defined in a parameter file. This can be done in the main TAU parameter file or in a separate file. In the case of a separate file, the filename must be defined in the main parameter file by using the parameter `Intermesh parameter filename`.

Sample input file A sample intermesh parameter file is described here. The parameter file defines a case, where a new grid is interpolated from two pregenerated grid which consist of two chimera blocks each. The interpolation is independent from physical time and it is assumed that the corresponding grid points are located on straight lines (In this case different angular settings of a flap. Grids are shown in Figure 31 and Figure 32).

The intermesh parameter file:

```
                    Intermesh mesh filename prefix: wing_im_30
Intermesh interpolation parameter by timestep (0/1): 0
    Equal partitioning in pregenerated meshes (0/1): 1

   ------------------------------------------------
                       Pregenerated mesh filename: wing_gap_0_structA
                        Pregenerated mesh file ID: 0
   pm end
   ------------------------------------------------
                       Pregenerated mesh filename: wing_gap_40_structA
```

```
                      Pregenerated mesh file ID: 1
pm end

-------------------------------------------------
                      Pregenerated mesh file ID: 0
                     Pregenerated mesh block ID: 0
                Pregenerated mesh block parameter: 0
pmb end
-------------------------------------------------
                      Pregenerated mesh file ID: 0
                     Pregenerated mesh block ID: 1
                Pregenerated mesh block parameter: 0
pmb end

-------------------------------------------------
                      Pregenerated mesh file ID: 1
                     Pregenerated mesh block ID: 0
                Pregenerated mesh block parameter: 40
pmb end
-------------------------------------------------
                      Pregenerated mesh file ID: 1
                     Pregenerated mesh block ID: 1
                Pregenerated mesh block parameter: 40
pmb end

-------------------------------------------------
                               New mesh block ID: 0
           New mesh parameter calculation method: POLYNOM
           New mesh parameter calculation coeffs: 28.3
                   New mesh interpolation method: LINEAR
nmb end
-------------------------------------------------
                               New mesh block ID: 1
           New mesh parameter calculation method: POLYNOM
           New mesh parameter calculation coeffs: 28.3
                   New mesh interpolation method: LINEAR
nmb end
```

At the beginning of the input file, the global parameters of grid filename, time-dependency and equal partitioning are defined. The filename of the new grid which may be appended by the domain number in case of a partitioned grid and some timestep information in case of a

time-dependent grid interpolation.
In lines 5-12 the filenames of the pregenerated grids are defined and these grids are each associated with a grid ID number.
This ID number is used to associate the motion state parameter of each block with the corresponding pregenerated grid in the lines 14 - 34.
From line 36 to the end the parameters for each block of the new grid are defined.
Note that the block ID numbers have to match the global block ids of the TAU primgrid data structure.

14 Python-Interface

14.1 Introduction

The Python-Interface provides the possibility of using the functionality provided by the TAU-Code modules from inside a script, such that the single modules are not longer monolithic stand-alone programs exchanging data by file-IO. With the python script the functionality of the different modules can be accessed in a more flexible way, allowing for combining TAU-Code programs (running them coupled together without file-IO) and to tailor the programs according to the needs for specific applications.

In order to illustrate, what is meant above, a simple example is given in the following: In principle the solver consists of different program parts, which are initialization (e.g. parameter settings), computation (iteration in time or versus steady state), output of monitoring data of field-, surface- and cut-plane data. When computing a polar for different angles of attack for a specific Mach number a loop over the different angles of attack is needed in which the solution is converged and final results are written out but the input of the dual grid is needed only once. Having access to the single functionalities allows to program this procedure in a script once and using it in future for computations of polars. Depending on the needs it can be decided if the preprocessor is started automatically at the beginning by the script, such that no dual grid file needs to be stored on the disk. Furthermore, it is possible to adapt the grid during the computation.

14.2 Usage of the Python-Interface

For the usage of the python interface a (very) limited knowledge about this scripting language is necessary. For more information about python see 'www.python.org'.

In order to use the python interface of TAU-Code python needs to be installed and needs to be included in the user-bin-path, such that typing 'python' in a shell invokes the interactive python-mode.

With the python interface in the TAU-Code the bin-directory of the TAU-Code distribution contains subdirectories for the different code version: py_el, py_turb1eq and py_turb2eq for the Euler-code, the code with one-equation turbulence models and the code with two-equation turbulence models, respectively. These directories contain the dynamic library 'tau_pythonmodule.so', for the specific code version and a file 'tau.py', which contains the part of code interpreting or executing input scripts for the TAU-Code. In addition some sample scripts (script_...) can be found in these directories, which can be used for specific applications, as templates to generate new scripts or even as 'tutorial scripts'.

In order to start the TAU-Code with a python script the environment variable 'PYTHONPATH' might have to be set to the current working directory, for example (using a cshell):

setenv PYTHONPATH $(TAUHOME)/taudir/bin/py_el

The usage to start the TAU-Code with a python script is similar to the usage of the stand-alone programs of the TAU-Code, but the script-name has to be added in addition, like:

```
bin-path/py_el/tau.py <script_name> <parameterfile> [logfile]
```

The same holds if starting the code with MPI, then the usage is (see also the descriptions how to start the solver in stand-alone mode)

```
mpirun -np # bin-path/py_el/tau.py <script_name> <parameterfile> [logfile] use_mpi
```

14.3 Status of the Python-Interface

The python interface is ongoing development. Additional interface functions will be added. At present for example, not all of the TAU-Code modules are wrapped for the usage under python. Currently, the preprocessor, the solver, the adaptation and the deformation tool can be used with python.
In addition to the wrapped functionality of the TAU-Code modules, some extra functionality is implemented in python-language. This code can be found also in the python subdirectories, which are all files with name prefix 'Py' and suffix 'py'. They contain for example the implementation of an online-monitoring of the solver, a handling of solver, preprocessor and adaptation interfaces, etc. These functions are in use inside the sample scripts. Since the ongoing implementation is not yet complete a detailed documentation is not yet available. At that status, the sample scripts might give the best introduction on how to make use of the python interface of the TAU-Code.

15 TAU-Code parameterfile database

This python-script-set named TauPDB handles a parameter database for the **Tau-Code** in BibTEX format.
The parameter database contains all parameters that are valid in the code. It simplifies the update of parameter documentation to make the User Guide more up-to-date and useful. It also offers several functions to query the database. All functions regarding the update and query procedures are described below.
All necessary paths and definitions can be set in pdb_bibconst.py
The main paths are the userguide_path, the database_path and the sourcecode_path.
Do not edit apc_const.py because this file is used for parameter-file creation and generated by TauPDB.
All functions provided by TauPDB can be accessed via the pdb_interface.py file.
It offers a text-based menu or if tkinter is available a GUI.

```
Use pdb_interface.py -g to start the GUI mode.
Use pdb_interface.py -h to display a quick-help.
```

15.1 Query parameter information

To gain quick access to a parameter description or to find parameters by full-text-search, two features are implemented. They search in all databases in the dataset. The matching database is displayed in the details.
The first one is the full text search that searches all values for the given search word. The second one is the search by keyword that queries only the name and is a bit faster and gives less output. It is impossible to combine keywords (AND, OR). All characters are allowed because the search is performed with one string. The third way is using regular expressions, here combining multiple words is possible by setting an .* between them

```
"dual.*grid" performs a search for dual AND grid (grid must appear after dual)
"dual|grid" performs a search for dual OR grid

Usage: pdb_interface.py
   -q  SEARCHSTRING performs the keyword search for the given SEARCHSTRING.
   -qf SEARCHSTRING performs the full text search for the given SEARCHSTRING
   -qr REGEXP performs the full-text-search for the given regular expression
```

15.2 Template file creation

Dependent on which database is selected (bibconst.DEFAULTDBTYP) or menu item 2 or OptionMenu in the headline of GUI mode, a template file for the automated parameter-file creation is built. This script set no longer exists as a standalone version. If you use the template files for another purpose do not hesitate to contact me. Templates will not be supported in future releases. It is necessary to rename the created file or to give it another name, because otherwise it will be overwritten. An append mode is not available at present.

```
Usage: pdb_interface.py -d 3.2
```

To create a valid parameter-file TauPDB has to know which groups and blocks and levels are valid and in which order they should appear. Cause these values may change when adding or deleting parameters it is recommended to create a new apc_const.py before creating a new template. The apc_const.py is independent from the selected database and its name (usually apc_const.py) is set in the pdb_bibconst-file.

```
Usage: pdb_interface.py -d 3.3
```

Note: These features are for compatability purpose only and deprecated.

15.3 Parameter-file creation

TauPDB has the ability to create parameter files. The user has to prepare three inputs. The first is the name of new parameter file. This is set in the pdb_bibconst.py file and can be changed in the GUI before processing the data.
The second input is the name of the file where the own data are saved. See the part about the parameter file verification there is described how such a file can be created automatically. It is proposed to create one and see how it is set up.
Or have a look at `taudir/input/para_dbs/mycase.db`.
The third input is the desired level. The higher the level is set the more output is generated. It is possible to set the OPTown_level option (e.g. see mycase.db) to modify the given levels and create more or less output than there will be by default. Finally press start to create the parameter-file.

```
Usage: pdb_interface.py -pc LEVEL
```

15.4 Parameter-file verification

We know, that it is easy to copy an old parameter-file and change existing values, but a lot of parameters change and or no longer exist. Now it is possible to check whether all parameters in a parameter-file exist in the database. All others are printed to stdout. In addition a file

is created that is set up like mycase.db. So you do not have to keep the whole parameter-file it is enough to keep this file with the changed values. These values can be applied to a new parameter file. See the parameter-file creation section.
The values are written to file named as set in the pdb_bibconst.py file at OWN_VALUES_FILE. In GUI mode this value can be changed.
The parameter-file that is verified can be entered in GUI or TUI mode interactive or if used at the command line it is the second argument.
Watch the console window carefully because all wrong parameters are printed to stdout.

```
Usage: pdb_interface.py -pv PARAMETERFILENAME
```

16 Hints for the grid generation

For many kinds of different boundary types a lot of experience was collected on how to establish the most appropriate grid for the solver. Mostly it is related to algorithms which were developed with special note to grid topology. Creating a complex, fully 3D grid is a time-consuming issue and the following hints might help to avoid difficulties during the simulation.

16.1 Engine inflow and Engine exhaust in an inviscid flow

The physical aspect of an inviscid exhaust plane is to provide a constant or concentric velocity profile given by the parameters of the boundary type. In this case some treatments have to be introduced to connect the exhaust plane with the flow field. The connection between flowfield and exhaust plane is established with an extrapolation of the pressure or the velocity from the nearest flowfield points.

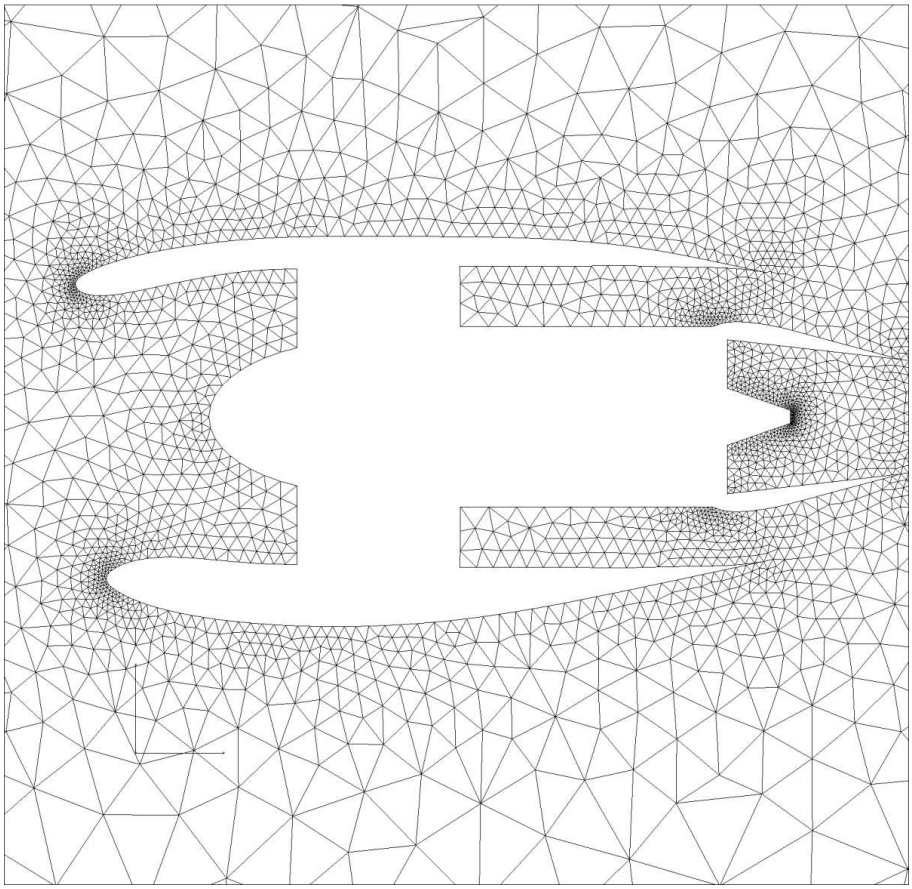


Figure 34: Tetrahedral grid around a jet engine.

In the cut plane of a jet engine through the middle axis, Figure 34, the boundaries on the fan

inflow (engine inflow), fan outflow (engine outflow), core outflow (engine outflow) and fixed walls (puler walls) can be observed. At the fan outflow there is a cylindrical exhaust tube. Using a tetrahedral mesh at the exit plane creates a non-constant distance from the plane to the first layer of points. The solver extrapolates the flow variables at different stages. To avoid unnecessarily high gradients on the outflow plane, which may actually hinder a convergence of the flow calculation, one can make a modification at the exhaust plane. In Figure 35, the first layers

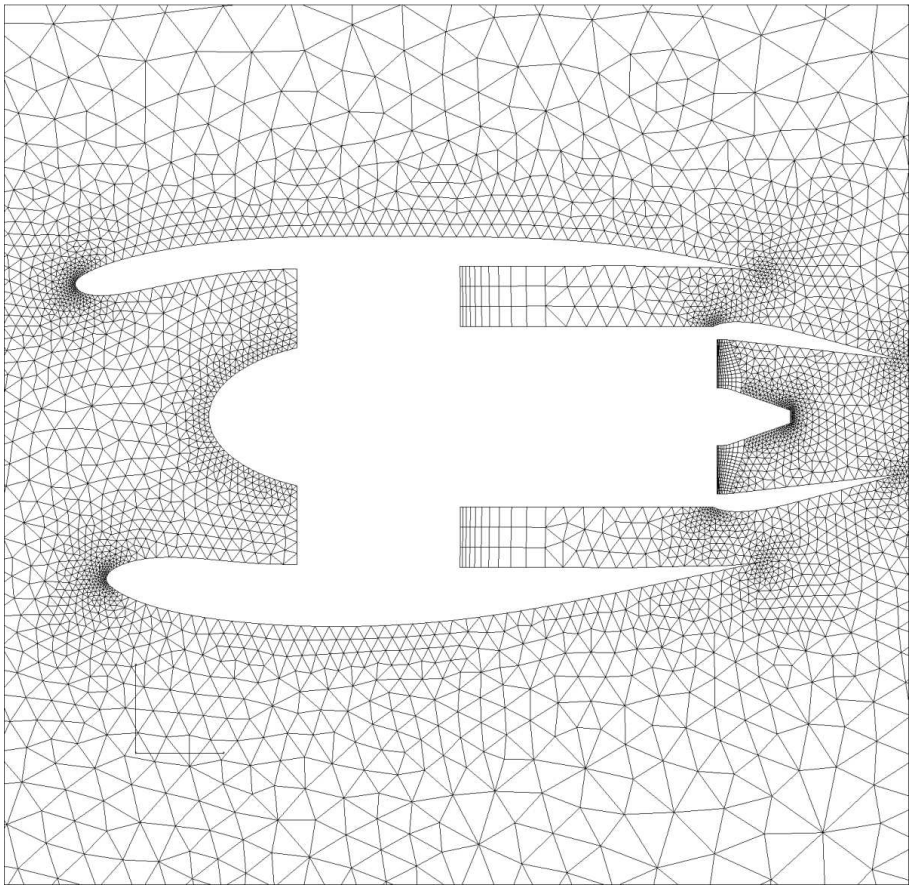


Figure 35: Structured grid behind fan exhaust plane.

behind the fan exit plane are structured. This enables a constant distance between the exit plane and the first point layer. Another idea is not to create the first distance too close to the plane or even a viscous type mesh. The time-step size is dependent on the smallest cell size and would increase the calculation time enormously. The aspect ratio for the prismatic layer should be close to one. It is not necessary to mesh the whole exhaust tube structured. In principle one layer is sufficient, in the present case in Figure 35, 10 sublayers were generated. The extrapolation needed for the engine boundary condition is performed at the finest grid level (primary grid) only which takes the first point layer into account. There is no update

on coarser multigrid levels, if used.
Special care should be taken on any kind of diverging or converging tubes, as seen in Figure 36. The velocity vectors are normal to the exit plane and if the mounted exit tube is not perpendicular it can again create difficulties for the convergence

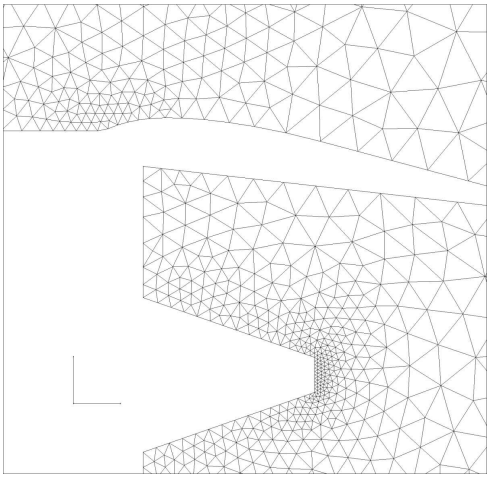


Figure 36: tetrahedral grid around the core jet

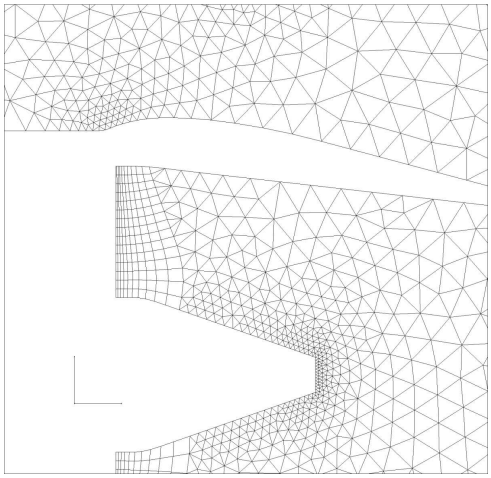


Figure 37: Modified geometry and structured grid around a core jet.

of the calculation. Just expand the exhaust tube upwards with a short cylindrical tube as can be seen in Figure 37. Always try to realize this feature on the exhaust plane, because the solver expects this modification.

16.2 Viscous and inviscid flow at the engine inflow.

The inflow and outflow of a jet turbine is comparable to a tube flow, as shown in Figure 38. To overcome oscillations on the inflow plane the flow variables can be determined with a theoretical approach in relation to a laval nozzle.

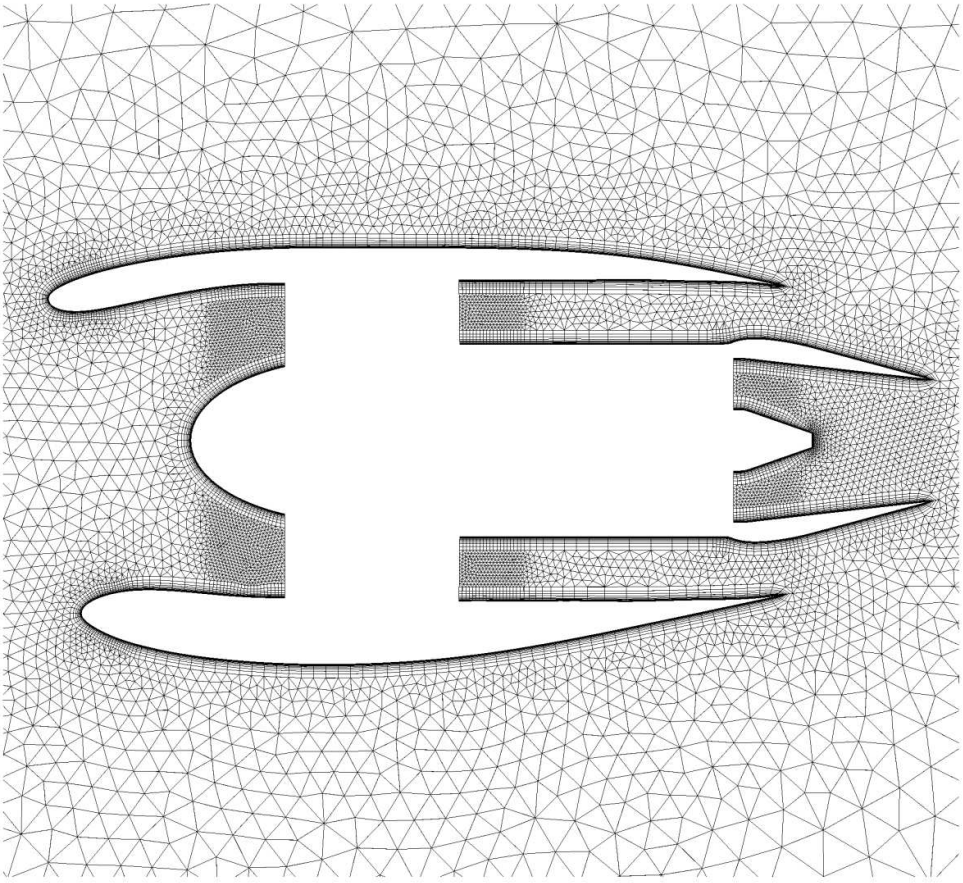


Figure 38: Structured grid around tubes for viscous flow.

On the other hand there is no coupling between the physical inlet and one plane before it. A compromise can be reached, and considerations lead to the coupled approach which reproduces the better physical solution. In case of an established flow at the inlet tube the characteristics of the flow should be mapped at the inflow plane, e.g. boundary layer flow. The engine boundary condition extrapolates the flow quantities one plane before the inlet. Only the pressure, for inviscid computations, is calculated using the parameter ϵ_{fan}, defined in the boundary mapping file, were it is introduced with the energy equation. The oscillations observed are negligible. The Figures 39 and 40 show the hybrid meshed planes. It is not recommended to create a viscous sublayer on the inflow and outflow plane concerning the flow in a tube, because in the middle there are no boundary flow characteristics. From another

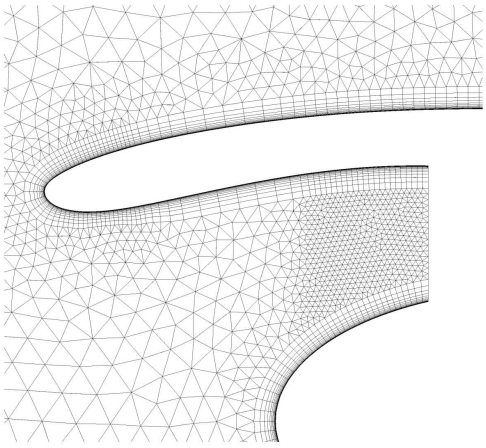


Figure 39: Structured grid around the inflow.

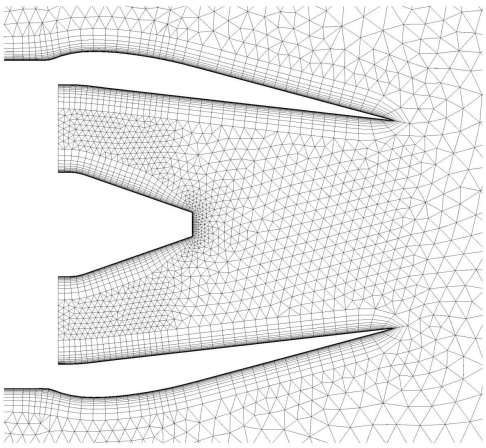


Figure 40: Structured grid around the core jet.

point of view, when structured parts meet each other at a too close angle, the dual meshes made at the corner in the preprocessing step are not usable for flux computations anymore. In general, diminish the creation of too fine spatial discretization wherever it is possible. It reduces the time stepping size and can create convergence difficulties. The grid made in the inflow and outflow region, especially for the planes, should be done with equal surface and volume elements. Some grid generators allow a volume source (cylindrical or cube type source) to be set around these regions, where this feature
is easy to control, Figure 38 and Figure 39. The result of this treatment is shown in Figure 41. The boundary layer is seen through the whole tube.

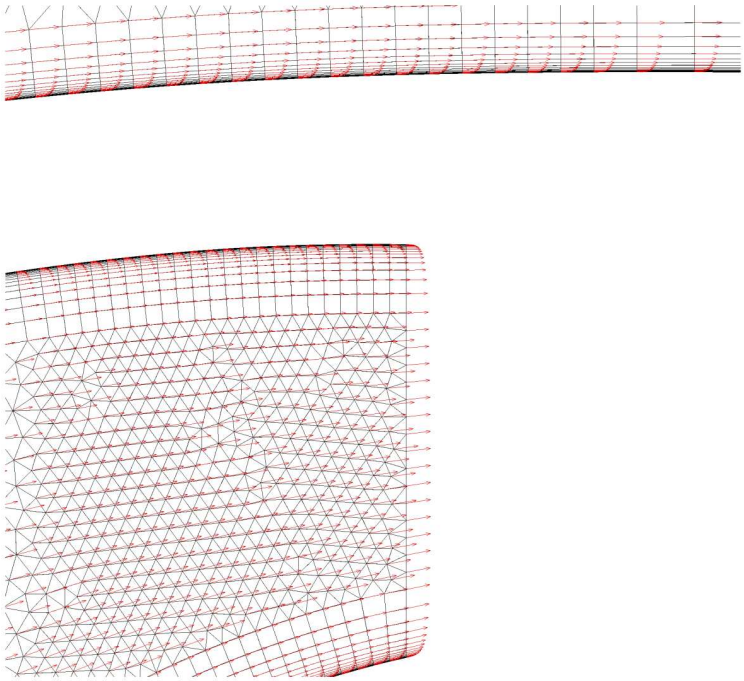


Figure 41: Velocity vectors at the inflow.

17 Coordinate Systems, Force and Moment Coefficients, and Motion

The sections in this part will cover some of the details of the following features of the TAU-Code:

- Coordinate Systems: explanation of the four coordinate systems present in the TAU-Code, and how they are used
- Force and Moment Coefficients: how they are calculated and in which coordinate system they are given
- Motion: how to setup a flow simulation for a single or multi-block grid in a transient rigid-body motion

17.1 Coordinate Systems of the TAU-Code

In Figure 42, the coordinate systems used by the TAU-Code are shown for a NACA0012 airfoil with the moment reference-point at the 1/4-chord. The subscripts defining the different systems have the following meanings:

τ	TAU-Code-Grid frame	f	Body-Fixed frame
g	Geodesic frame	a	Aerodynamic frame

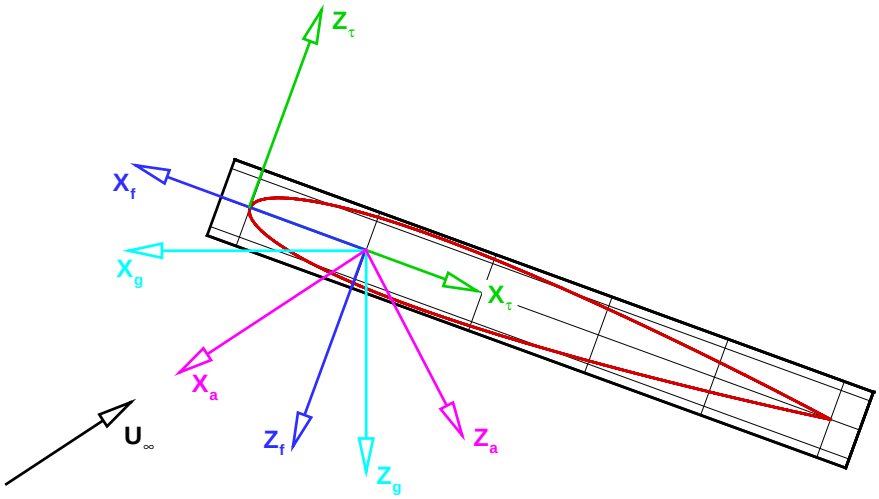


Figure 42: Coordinate systems of the TAU-Code.

- Both the TAU and Body-Fixed systems move and rotate with the body
- The origin of the Body-Fixed system is specified by the user in the parameter-file
- The origin of the TAU system is always at coordinate (0, 0, 0) of the primary grid
- If no translation is specified, the origin of the Body-Fixed and Geodesic systems will coincide
- The Geodesic system is considered to be the inertial system for the simulation
- The Aerodynamic system is as defined by the LN9300 norm, where the x-axis is in a direction parallel to the flight-velocity vector (see Figures 42 and 66)

17.2 Force and Moment Coefficients

The force coefficients C_{Fi} and moment coefficients C_{Mi} are defined as

$$C_{Fi} = \frac{F_i}{\rho/2 \ \mathrm{v}^2 A}, \tag{61}$$

$$C_{Mi} = \frac{M_i}{\rho/2 \ \mathrm{v}^2 A \, l}, \tag{62}$$

where F_i are the components of the integral force and M_i are the components of the integral moment given in one of the body-fixed systems (TAU or Body-Fixed system), ρ is the reference density, v the reference velocity, A the reference relation area, and l the reference length for either pitching or rolling/yawing momentum.
The integral coefficients shown during a simulation are the combined values for the whole configuration. At the end of the simulation a table is shown which lists the contributions of the individual components to the total force and moment values.
When using TAU-Python the contributions of the individual components can be accessed during the simulation.

The standard aerodynamic force coefficients, C_L, C_D, and C_{Sidef} are *always* given in the aerodynamic coordinate system, with the convention that a positive C_D and C_L are in the negative x_a and z_a directions, respectively.

17.2.1 Moment Reference Point

If rigid-body motion is not activated, the moment reference point is specified by the parameters:

```
Origin coordinate x: <value in grid units>
Origin coordinate y: <value in grid units>
Origin coordinate z: <value in grid units>
```

If rigid-body motion is activated, the moment reference point is specified as the origin of the main motion node using the parameter *Evaluate forces and moments at node* in combination with the *Origin of local coordinate system* for the given motion node, as shown here:

```
----------------------------------------: -
      Evaluate forces and moments at node: naca0012
----------------------------------------: -
                    Motion description id: naca0012
        Origin of local coordinate system: <x, y, z coordinates in grid units>
----------------------------------------: -
```

17.3 Motion

In this section the procedure of how to setup a flow simulation with rigid-body motion activated will be explained. For more in-depth examples of rigid-body motion simulations, please refer to the document `Tau-Code: Examples for Unsteady Simulations` contained in the archive-file `TAU-unsteady-example.tar.gz` which came with your TAU-Code distribution. If your distribution did not contain this archive-file, please contact Gunnar Einarsson (gunnar.einarsson@dlr.de).

17.3.1 Basic Rigid-Body Motion Parameters

To enable the basic rigid-body motion capabilities of the TAU-Code, the following parameters must be specified in the main parameter-file:

```
          Type of grid movement: rigid
    Motion hierarchy filename: <path/filename or (thisfile)>
Motion description filename: <path/filename or (thisfile)>
```

the settings shown above enable both single and multi-body motions. The *Type of grid movement* parameter may of course be any of the other available movement types. The following parameter, which is also in the main parameter-file, is optional:

```
Extended motion monitoring (0/1): 0
```

when activated, extended information regarding the movement of each motion node will be written to the log-file; the output will look similar to the following example:

```
 ------------------------------------------------------------------------------
Motion parameters for Node 0 :
  Node name.............................:  aircraft
  Parent-node name......................:  inertial
  Deflection angles (roll, pitch, yaw)..:  0.0000e+00 -3.6000e+00  0.0000e+00 [deg]
  Angular velocity (body-fixed frame)...:  0.0000e+00 -2.5267e-01  0.0000e+00 [1/s]
  Translational velocity (geod. frame)..:  0.0000e+00  0.0000e+00  0.0000e+00 [m/s]
  Displacement of origin (geod. frame)..:  0.0000e+00  0.0000e+00  0.0000e+00 [m]

 ------------------------------------------------------------------------------
```

17.3.2 Motion Hierarchy Definition - Single-Body

A motion hierarchy model, which contains information regarding reference-frame connectivity and the motion-node to grid-block connectivity, must be defined by the user in the `Motion hierarchy file`, even for the one motion-node needed for basic single-body motion. The following example shows a typical definition for a single-body motion hierarchy model:

```
                    Node name: naca0012
        Node reference frame: inertial
  Node controls grid block: 1
Node motion description id: pitch
                    hdf end
```

17.3.3 Motion Description Definition - Single-Body

For each unique *Node motion description id* defined in the motion hierarchy, a corresponding Motion Description must be defined in the `Motion description file`. For the Node example given above, the following Motion Description could be used to create a periodic pitching motion around the 1/4-chord point of an airfoil:

```
                    Motion description id: pitch
                         Type of movement: periodic
            Origin of local coordinate system: 0.25 0.0 0.0
     Degree of Fourier series for rotation: 1
           Reduced frequency for rotation: 0.0 0.1628 0.0
        Reduced frequency reference length: 1.0 1.0 1.0
Fourier coefficients for rotation (sin) pitch: 0.0 2.51
                                    mdf end
```

17.3.4 Motion hierarchy definition - Multi-Body

As before, a motion hierarchy model must be defined by the user in the `Motion hierarchy file`. The following example shows a typical definition for a wing-flap motion hierarchy model:

```
                    Node name: wing
        Node reference frame: inertial
  Node controls grid block: 1
Node motion description id: Wing
                    hdf end

                    Node name: flap
        Node reference frame: wing
  Node controls grid block: 2
Node motion description id: Flap
                    hdf end
```

17.3.5 Motion description definition - Multi-Body

For each unique *Node motion description id* defined in the motion hierarchy, a corresponding Motion Description must be defined in the `Motion description file`. For the wing-flap

example given above, the following Motion Description could be used to create a periodic pitching motion of the wing around the 1/4-chord point, and of the flap around its hinge-point:

```
                          Motion description id: Wing
                               Type of movement: periodic
                  Origin of local coordinate system: 0.25 0.0 0.0
           Degree of Fourier series for rotation: 1
                  Reduced frequency for rotation: 0.0 0.1628 0.0
              Reduced frequency reference length: 1.0 1.0 1.0
  Fourier coefficients for rotation (sin) pitch: 0.0 2.51
                                        mdf end
  ---------------------------------------------: -
                          Motion description id: Flap
                               Type of movement: periodic
                  Origin of local coordinate system: 0.81 0.0 0.0
           Degree of Fourier series for rotation: 1
                  Reduced frequency for rotation: 0.0 0.1628 0.0
              Reduced frequency reference length: 1.0 1.0 1.0
  Fourier coefficients for rotation (sin) pitch: 0.0 5.0
                                        mdf end
```

17.3.6 General Rotation Axis

If a rotational motion of a body can not be easily described using the main reference-frame axes, the use of a general rotation axis might help. The general rotation axis is a vector in space around which the body-fixed reference frame of the current node is rotated. When a general rotation axis is activated, the polynomial and Fourier series described above are no longer used - the 'Hinge' specific parameters listed in the general user-guide section are used instead. The following example shows the same pitching oscillation as above, but this time using a general rotation axis:

```
  ---------------------------------------------: -
                         Hinge - specify vector: 0.0 1.0 0.0
         Hinge - reduced frequency for rotation: 0.1628
     Hinge - reduced frequency reference length: 1.0
Hinge - Fourier coefficients for rotation (sin): 0.0 2.51
                                        mdf end
  ---------------------------------------------: -
```

Explanation of motion parameters:

parameter name	type	range	default	in	page
Degree of Fourier series for rotation	int	$0 \leq$ i	0	s	374
Degree of Fourier series for translation	int	$0 \leq$ i	0	s	374
Degree of polynomial for rotation	int	$0 \leq$ i	0	s	374
Degree of polynomial for translation	int	$0 \leq$ i	0	s	374
Evaluate forces and moments at node	string	-	-	s	374
Fourier coefficients for rotation (cos) pitch	array-float	-	0	s	374
Fourier coefficients for rotation (cos) roll	array-float	-	0	s	374
Fourier coefficients for rotation (cos) yaw	array-float	-	0	s	375
Fourier coefficients for rotation (sin) pitch	array-float	-	0	s	375
Fourier coefficients for rotation (sin) roll	array-float	-	0	s	375
Fourier coefficients for rotation (sin) yaw	array-float	-	0	s	375
Fourier coefficients for translation (cos) x	array-float	-	0	s	375
Fourier coefficients for translation (cos) y	array-float	-	0	s	375

continued on next page

parameter name	type	range	default	in	page
Fourier coefficients for translation (cos) z	array-float	-	0	s	375
Fourier coefficients for translation (sin) x	array-float	-	0	s	375
Fourier coefficients for translation (sin) y	array-float	-	0	s	375
Fourier coefficients for translation (sin) z	array-float	-	0	s	375
Hinge - Fourier coefficients for rotation (cos)	array-float	-	0	s	376
Hinge - Fourier coefficients for rotation (sin)	array-float	-	0	s	376
Hinge - polynomial coefficients for rotation	array-float	-	0	s	376
Hinge - reduced frequency for rotation	float	-	0	s	376
Hinge - reduced frequency reference length	float	non-zero	0	s	376
Hinge - specify vector	array-float	-	0	s	376
Initial phase shift	array-float	-	{0, 0, 0}	s	376
Motion description filename	string	-	(none)	s	376
Motion description id	string	-	default	s	376
Node controls grid block	array-int	-1, ≥ 1	1	s	376
Node motion description id	string	-	default	s	377

continued on next page

parameter name	type	range	default	in	page
Node name	string	-	default	s	377
Node reference frame	string	-	inertial	s	377
Origin of local coordinate system	array-float	-	{0, 0, 0}	s	377
Polynomial coefficients for rotation pitch	array-float	-	0	s	377
Polynomial coefficients for rotation roll	array-float	-	0	s	377
Polynomial coefficients for rotation yaw	array-float	-	0	s	377
Polynomial coefficients for translation x	array-float	-	0	s	377
Polynomial coefficients for translation y	array-float	-	0	s	377
Polynomial coefficients for translation z	array-float	-	0	s	377
Reduced frequency for rotation	array-float	-	{0, 0, 0}	s	378
Reduced frequency for translation	array-float	-	{0, 0, 0}	s	378
Reduced frequency reference length	array-float	-	{1, 1, 1}	s	378
Type of movement	string	-	periodic	s	378

Translation:

$$x(t) = \sum_{k=0}^{k=N_{PT}} p_k t^k + a_0 + \sum_{k=1}^{N_{FT}} (a_k \cos(k\omega_T t) + b_k \sin(k\omega_T t))$$

Rotation:

$$\phi(t) = \sum_{k=0}^{k=N_{PR}} r_k t^k + c_0 + \sum_{k=1}^{N_{FR}} (c_k \cos(k\omega_R t) + d_k \sin(k\omega_R t))$$

Where ω_T is the angular velocity for the translation,

$$\omega_T = \frac{F_T q_\infty}{c}$$

and ω_R is the angular velocity for the rotation,

$$\omega_R = \frac{F_R q_\infty}{c}.$$

(F_T, F_R: reduced frequencies, q_∞: free stream velocity, c: chord length)

- The movement of a node is described in grid units by a translation and a rotation of the moving coordinate system about the origin x_0. Translation as well as rotation are defined by polynomials and Fourier series.

`Degree of Fourier series for rotation:` integer, default 0

- These parameters control the order of the series that the code will use to determine rotational motion.

`Degree of Fourier series for translation:` integer, default 0

- These parameters control the order of the series that the code will use to determine translational motion.

`Degree of polynomial for rotation:` integer, default 0

- These parameters control the order of the series that the code will use to determine rotational motion.

`Degree of polynomial for translation:` integer, default 0

- These parameters control the order of the series that the code will use to determine translational motion.

`Evaluate forces and moments at node:` character string, default -

- The name of the node around whose origin and in whose body-fixed reference frame the forces and moments will be calculated.

`Fourier coefficients for rotation (cos) pitch:` array-float, default 0

- Cosine series coefficients for rotation, given in degrees in the local body-fixed reference frame.

`Fourier coefficients for rotation (cos) roll:` array-float, default 0

- Cosine series coefficients for rotation, given in degrees in the local body-fixed reference frame.

`Fourier coefficients for rotation (cos) yaw:` array-float, default 0

- Cosine series coefficients for rotation, given in degrees in the local body-fixed reference frame.

`Fourier coefficients for rotation (sin) pitch:` array-float, default 0

- Sine series coefficients for rotation, given in degrees in the local body-fixed reference frame.

`Fourier coefficients for rotation (sin) roll:` array-float, default 0

- Sine series coefficients for rotation, given in degrees in the local body-fixed reference frame.

`Fourier coefficients for rotation (sin) yaw:` array-float, default 0

- Sine series coefficients for rotation, given in degrees in the local body-fixed reference frame.

`Fourier coefficients for translation (cos) x:` array-float, default 0

- Translation, cosine series coefficients given in grid units, in the body-fixed reference frame of the parent-node.

`Fourier coefficients for translation (cos) y:` array-float, default 0

- Translation, cosine series coefficients given in grid units, in the body-fixed reference frame of the parent-node.

`Fourier coefficients for translation (cos) z:` array-float, default 0

- Translation, cosine series coefficients given in grid units, in the body-fixed reference frame of the parent-node.

`Fourier coefficients for translation (sin) x:` array-float, default 0

- Translation, sine series coefficients given in grid units, in the body-fixed reference frame of the parent-node.

`Fourier coefficients for translation (sin) y:` array-float, default 0

- Translation, sine series coefficients given in grid units, in the body-fixed reference frame of the parent-node.

`Fourier coefficients for translation (sin) z:` array-float, default 0

- Translation, sine series coefficients given in grid units, in the body-fixed reference frame of the parent-node.

`Hinge - Fourier coefficients for rotation (cos):` array-float, default 0

- This parameter defines the cosine set of Fourier coefficients for rotation.

`Hinge - Fourier coefficients for rotation (sin):` array-float, default 0

- Fourier coefficients for sine part of rotation series.

`Hinge - polynomial coefficients for rotation:` array-float, default 0

- Coefficients of polynomial describing rotation. What is important to realize is that with the polynomial description a constant displacement in time is easily achieved.

`Hinge - reduced frequency for rotation:` float, default 0

- The reduced frequency of a periodic motion is defined here. The reduced frequency k is given by $k = \frac{\omega * l}{U_\infty}$ where ω is the circular frequency of the motion (radians/second), l is a reference length, and U_∞ is the freestream velocity.

`Hinge - reduced frequency reference length:` float, default 0

- Reference length for computing the reduced frequency. Typically this is the chord length in aerodynamic calculations, although half chord is also common.

`Hinge - specify vector:` array-float, default 0

- Vector, given in the TAU-Code grid reference frame, that represents the axis of rotation for the given motion node.

`Initial phase shift:` array-float, default {0, 0, 0}

- The initial starting angle-of-attack, α, may now be set to minimum α, maximum α, or mean α (using -1, 1, or 0 as parameter values, respectively).

`Motion description filename:` character string, default (none)

- The name (including the path) of the ASCII input file defining the motion-description for rigid-body movement. The format of the file is the same as for the boundary-mapping file, with the keywords 'end mdf' to mark the end of a motion description.
  If the data is defined inside the parameter file itself, either the name (with path) of the parameter file has to be set or for simplicity the key '(thisfile)' can be used. The latter is of advantage because a renaming of the parameter is not necessary when renaming the parameter file.

`Motion description id:` character string, default default

- The unique name used to identify this motion-description block.

`Node controls grid block:` array-int, default 1

- The identification number(s) of the grid block(s) controlled by this node. A node which is not directly connected to a grid block is defined with the -1 option.

`Node motion description id:` character string, default default

- The name of the motion-block description belonging to this node. A corresponding motion-description, using the same name as given here, must be defined in the Motion description filename.

`Node name:` character string, default default

- A unique name to identify this motion-node.

`Node reference frame:` character string, default inertial

- The name of the reference (parent) frame of the current node. (NOTE: *One and only one* node must use the inertial frame as its reference frame).

`Origin of local coordinate system:` array-float, default {0, 0, 0}

- The absolute origin of this motion-block, given in the TAU-Code Grid coordinate system.

`Polynomial coefficients for rotation pitch:` array-float, default 0

- Coefficients for rotation, given in degrees, in the local body-fixed reference frame.

`Polynomial coefficients for rotation roll:` array-float, default 0

- Coefficients for rotation, given in degrees, in the local body-fixed reference frame.

`Polynomial coefficients for rotation yaw:` array-float, default 0

- Coefficients for rotation, given in degrees, in the local body-fixed reference frame.

`Polynomial coefficients for translation x:` array-float, default 0

- Translation coefficients, given in grid units in the body-fixed reference frame of the parent-node.

`Polynomial coefficients for translation y:` array-float, default 0

- Translation coefficients, given in grid units in the body-fixed reference frame of the parent-node.

`Polynomial coefficients for translation z:` array-float, default 0

- Translation coefficients, given in grid units in the body-fixed reference frame of the parent-node.

`Reduced frequency for rotation:` array-float, default {0, 0, 0}

- The reduced frequency similarity parameter k, for rotation :

$$k = \frac{\omega * l_k}{q_\infty}$$

.

`Reduced frequency for translation:` array-float, default {0, 0, 0}

- The reduced frequency similarity parameter k, for translation:

$$k = \frac{\omega * l_k}{q_\infty}$$

.

`Reduced frequency reference length:` array-float, default {1, 1, 1}

- The characteristic length, l_k, used for calculating the reduced frequency. Usually the chord length c is taken.

`Type of movement:` character string, default periodic

- One of currently three keywords used to identify the basic motion type for this node.
  periodic a general periodic motion described using Fourier series and reduced frequencies (and polynomial series, if desired). The code determines the unsteady physical timestep size.
  rotate a periodic motion for the specific case of a constant 360^o rotation around a given axis, described using reduced frequencies only. The code determines the series coefficients and the unsteady physical timestep size.
  rigid general motion described using Fourier and polynomial series. The user must specify the unsteady physical timestep size.

18 Overset unstructured grids method (Chimera method)

The following sections will explain how to setup a flow simulation for a multi-block grid for steady problems. This document will be updated in the near future to cover multi-block using automatic hole cutting. For more information about the implementation of Chimera method in the TAU-Code see **www.arl.hpc.mil/Overset2002** and
www.mit.jyu.fi/eccomas2004 .
The current method has three principal elements: the creation of the overlapping region, the calculation of the interpolation coefficients and the exchange of the flow variables among the blocks.The first step mentioned above is part of the grid generator, the user should specify the analytic shapes, such as cubes, spheres, cones, etc., as hole cutting geometry allows points to be classified very quickly. The second step is embodied in the preprocessing module, where the primary grid is still known. And the third element is part of the solver.
To illustrate the above three steps, we consider, for example simple three grids discretization of the background mesh B_1, wing B_2 and flap B_3, see figure 43.

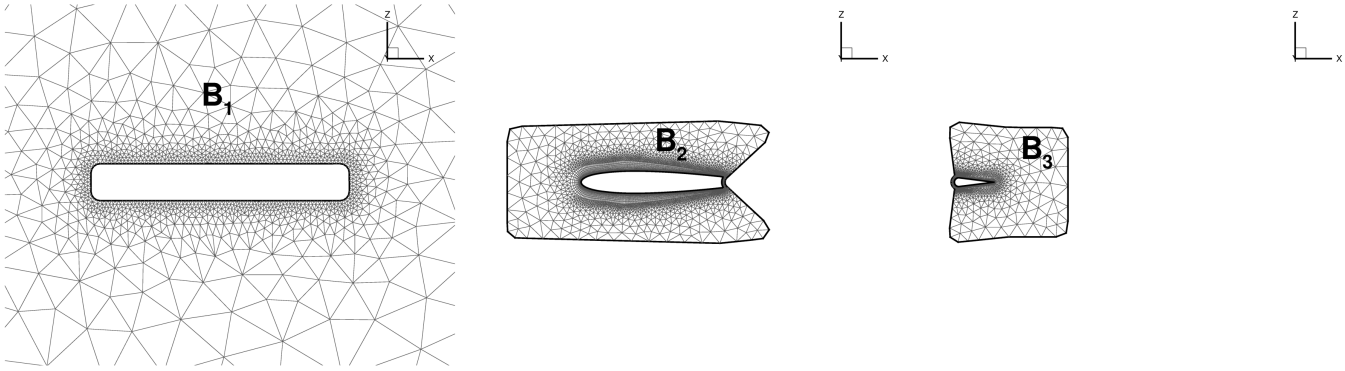


Figure 43: Mesh independently generated, and the definition of the boundary chimera overlap.

Chimera meshes introduce holes and artificial boundaries inside the computational domain, see figure43. The three blocks are added to get one data set using the setup_taugrid tool, see figure 44.
The syntax to merge the connectivity is

```
bin-path/setup_taugrid B1

Option (1 = rewrite options):  20

Grid file name:  B2

Option (1 = rewrite options):  20

Grid file name:  B3
```

```
Option (1 = rewrite options):  99

Output file name:  B123.grid

Option (1 = rewrite options):  0
```

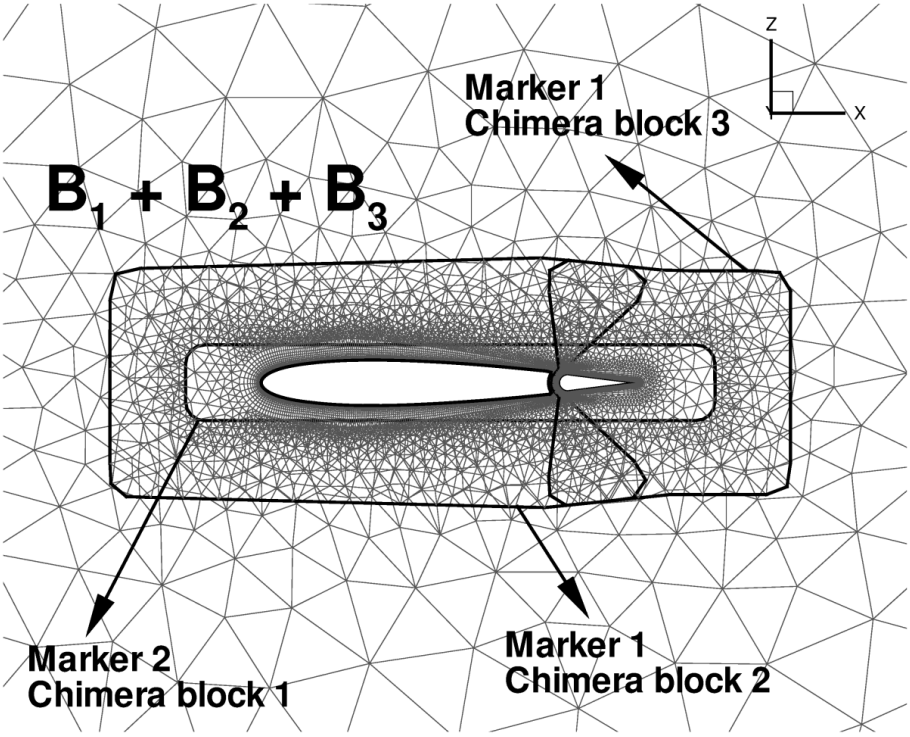


Figure 44: Merge the grid in one data structure.

18.1 Parameter file for chimera

A parameter file for the above example could look like:

```
                        Type : farfield
                     Markers: 1
                Chimera block: 1
 block end
                        Type : chimera overlap
                     Markers: 2
                Chimera block: 1
 block end
                        Type : symmetry plane
                     Markers: 3
                Chimera block: 1
 block end
```

```
                        Type : symmetry plane
                      Markers: 4
                Chimera block: 1
block end

                        Type : turbulent wall
                      Markers: 2
                Chimera block: 2
block end
                        Type : symmetry plane
                      Markers: 3
                Chimera block: 2
block end
                        Type : symmetry plane
                      Markers: 4
                Chimera block: 2
block end
                        Type : chimera overlap
                      Markers: 1
                Chimera block: 2
block end
                        Type : chimera overlap
                      Markers: 1
                Chimera block: 3
block end
                        Type : symmetry plane
                      Markers: 3
                Chimera block: 3
block end
                        Type : symmetry plane
                      Markers: 4
                Chimera block: 3
block end
                        Type : turbulent wall
                      Markers: 2
                Chimera block: 3
      Write surface data (0/1): 0
block end

              Number of blocks: 3
    Number of multigrid levels: 5
```

```
Number of primary grid domains: 4
             Number of domains: 4
```

18.2 Parallel chimera

For parallel computations, subsets of the primary grids are created by domain decomposition, see Figure 45, the following parameters must be specified in the main parameter-file:

```
Number of primary grid domains: 4
             Number of domains: 4
```

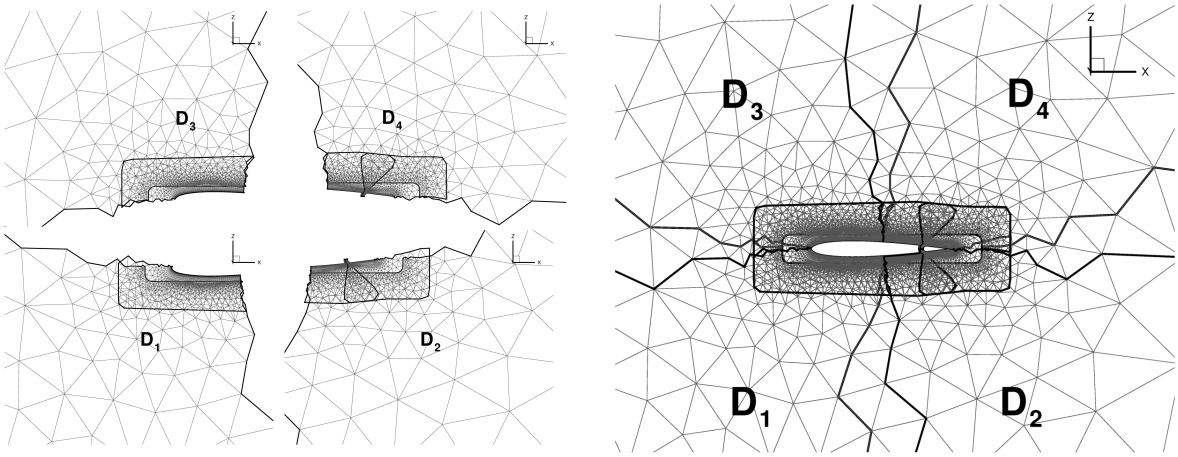


Figure 45: Multi-block, subdivided into four subdomains D_1, D_2, D_3 and D_4, and one-element wide overlapping region that is shared by neighboring subdomains.

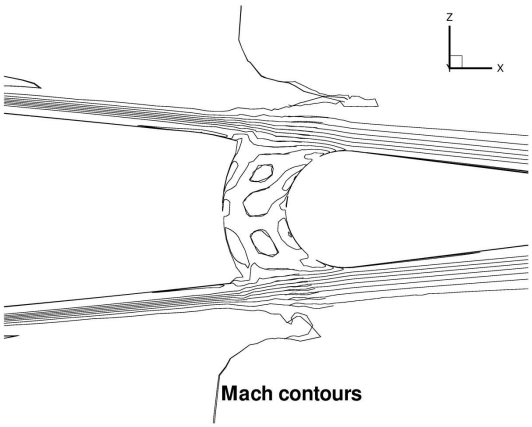


Figure 46: Mach contours around the flap region.

The communication between the subdomains is based on MPI.
To run Chimera in parallel for Euler or Navier-Stokes Equations, we can use the following step :

- Inviscid computation (Euler):
  `mpirun -np # bin-path/py_el/tau.py <script_name> <parameterfile> [logfile] use_mpi`

- Viscous computation (Navier-Stokes) using two-equation turbulence models:
  `mpirun -np # bin-path/py_turb2eq/tau.py <script_name> <parameterfile> [logfile] use_mpi`

18.3 Remark for the overlapping layers

The overlapping layer usually has a width of one or two cell see figure 47, in order to get converged solution.

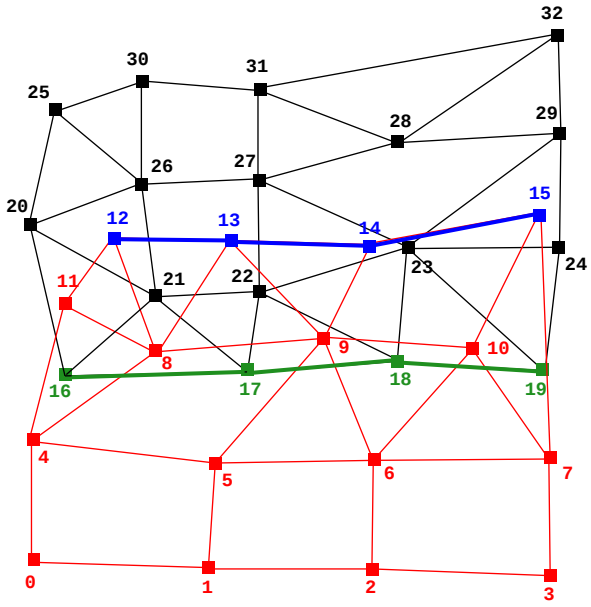


Figure 47: Overset unstructured grids system.

18.4 Grid overlap on body surfaces

18.4.1 Description

Chimera grids may overlap on body surfaces. One example is presented in Figure 48 showing the junction of a wing and a body, where the wing and the body are discretized with different

grids.

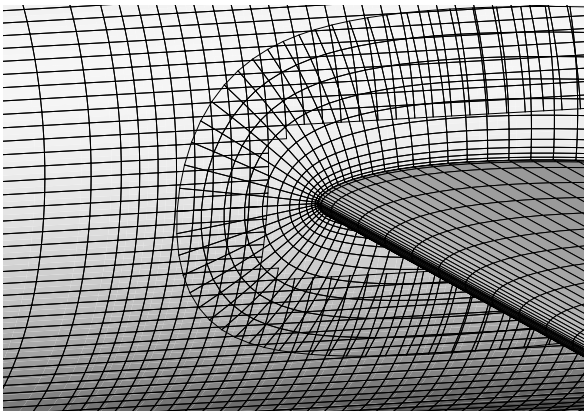


Figure 48: Grids with overlap on body surfaces

A grid overlap on body surfaces requires a modification in the computation of interpolation coefficients because the standard interpolation may yield inaccurate results: Due to the different discretizations of a curved analytic surface with the two overlapping grids the distance of a point P to the discrete surface representations is not equal, $\delta_1 \neq \delta_2$, see Figure 49, left. Therefore, all interpolation points close to a curved surface will interpolate flow data from locations with different wall distance than expected. This may for example cause problems when transferring data within boundary layers. Close to a concave surface some interpolation points may even be outside the other grid which makes an interpolation impossible.

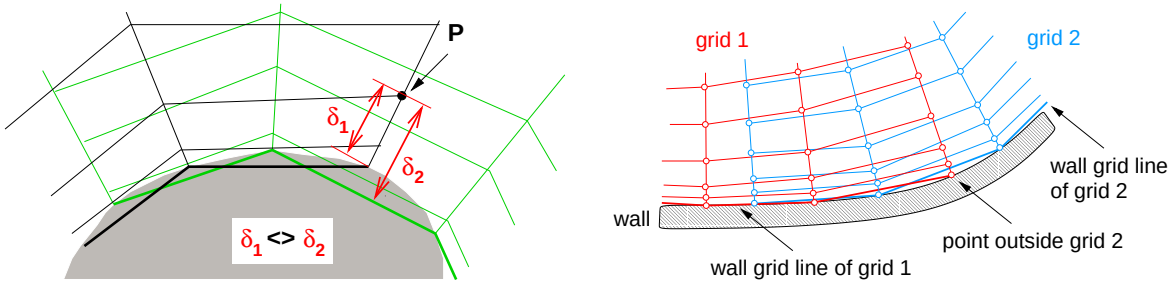


Figure 49: Error when interpolating close to curved surfaces. Left: convex surface leads to ambiguous wall distance, right: on concave surfaces an interpolation is impossible.

The interpolation problem can be solved by a projection method: The procedure starts by searching the grid containing the interpolation point P for the closest point P_s on the surface, see Figure 50, left. The point P_s can be located on a surface element, on the edges of the element or on its nodes. Now the point P_s is projected onto the discrete surface representation of the overlapping grid. The projection vector is denoted with $\vec{\varepsilon}$.
The projection vector is now multiplied with a scaling function and added to the coordinates

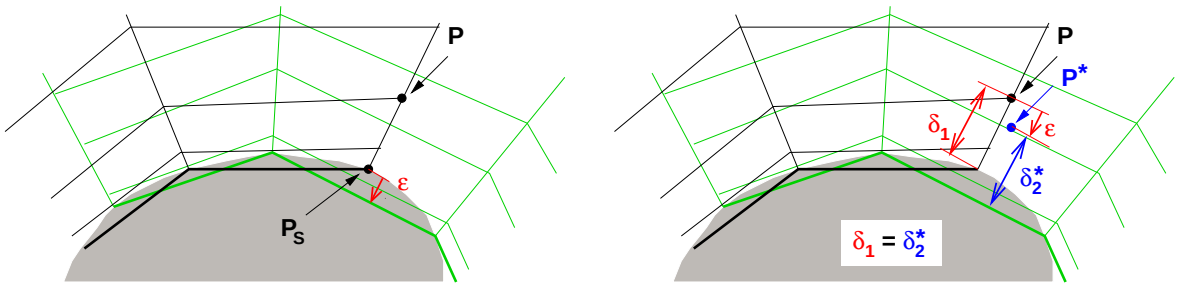


Figure 50: Left: projection of point on surface onto surface of overset grid, right: introduction of virtual target point P^*

of the interpolation point $\vec{x}_P$ giving the coordinates $\vec{x}_{P^*}$ of a virtual interpolation point P^*

$$\vec{x}_{P^*} = \vec{x}_P \cdot w + \vec{\varepsilon}$$

see Figure 50, right. The weighting function w is used to apply a full correction close to the surface and to reduce the correction with increasing distance from the surface

$$w = \begin{cases} 1 & \text{if} \quad 0 \leq |\vec{x}_P - \vec{x}_{P_s}| < d_1 \\ \frac{d_2 - |\vec{x}_P - \vec{x}_{P_s}|}{d_2 - d_1} & \text{if} \quad d_1 \leq |\vec{x}_P - \vec{x}_{P_s}| < d_2 \\ 0 & \text{if} \quad d_2 \leq |\vec{x}_P - \vec{x}_{P_s}| \end{cases}$$

where

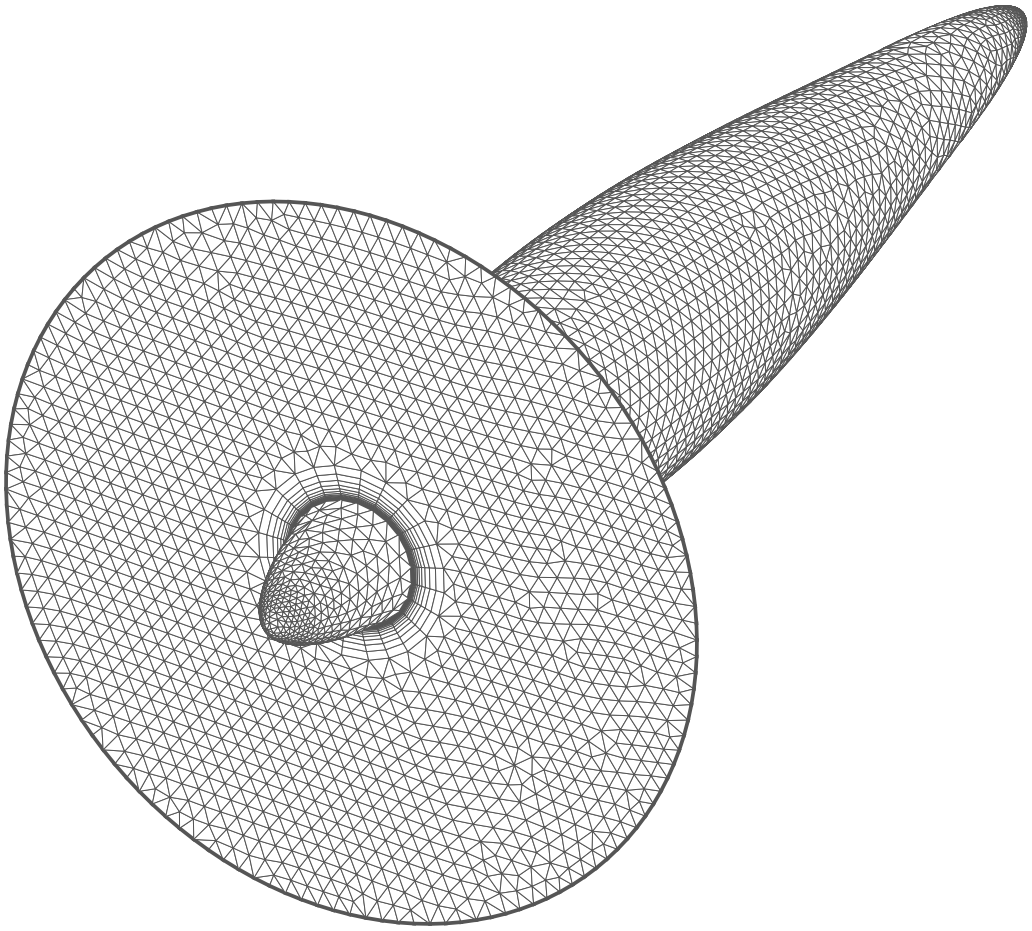
$$d_1 = 10 \cdot |\vec{\varepsilon}| \quad , \quad d_2 = 30 \cdot |\vec{\varepsilon}|$$

gives good results. The coordinates $\vec{x}_{P^*}$ of the virtual interpolation point P^* are now used to search a donor cell and to compute the interpolation coefficients. The presented algorithm ensures an identical wall distance of the interpolation point and the interpolated data, $\delta_1 = \delta_2^*$, see Figure 50, right.

18.4.2 Input parameters for Chimera wall projection method

Apply Chimera wall projection (0/1) (page 40)
Mismatch of overlapping walls (page 76)

19 Actuator Disks in TAU



19.1 Introduction

Propellers are a reliable and effective mean of propulsion for many aircrafts in the low and middle speed range. The slipstream of a propeller affects the flow around the aircraft and is of great importance to the stability and aerodynamic capability of the design. The principles of propellers also apply to helicopter rotors which produce thrust for lift and propulsion. The numerical simulation of rotating blades requires an instationary calculation. This is often too time consuming for the calculation of the flow around a complete aircraft configuration. In addition the discretization of the blade geometry increases the grid size especially for viscous calculations. In many cases the flow around the blade is not of interest and the slipstream can be considered as stationary. This motivates to replace the propeller with a stationary geometrical simpler model propeller.

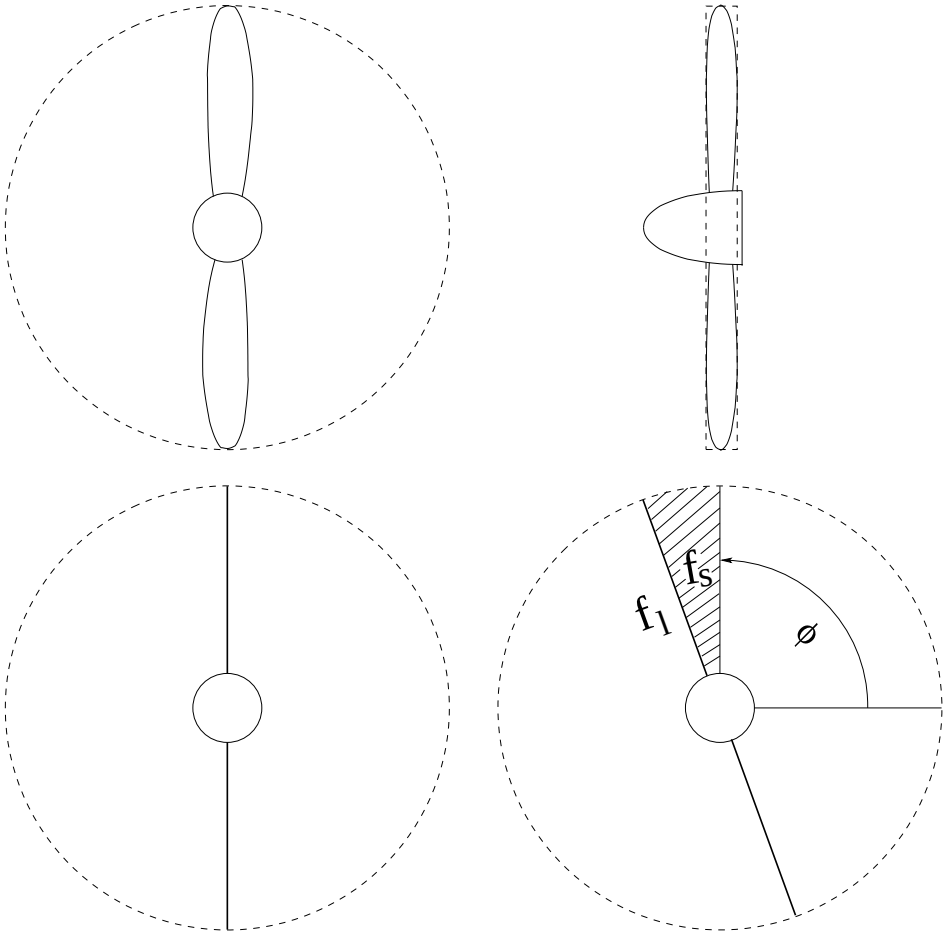


Figure 51: Idealization of two bladed propeller and actuator disk.

19.2 The Actuator Disk Model

Propellers generate thrust due to aerodynamic forces acting on the surface of rotating blades. As a consequence reaction forces act on the fluid which increase the momentum and the energy of the flow. The model propeller must apply equivalent stationary forces to the fluid. The derivation of the model propeller is explained in figure 51. Because the extension of the blade in circumferential direction is small compared to the extension in radial direction the blade is replaced with a rotating line. The force distribution on the surface of the blade is partially integrated in circumferential direction to a line force distribution in radial direction. An equivalent stationary surface force distribution is calculated by time averaging the rotating line force distribution over the time span T for one revolution. For an observer fixed at the position (r, ϕ) on the disk the rotating line drops by and during the small timespan dt the line force distribution active on the line element dr sweeps the sector $d\phi = \omega dt$ with the angular velocity ω. For the rest of the time $T - dt$ no force is acting on this area. Time averaging the

momentum applied to this area over one revolution leads to an equivalent stationary surface force at this point. Time averaging the momentum for each point swept by the rotating line leads to an equivalent stationary surface force distribution for a disk.
The representation of a propeller by a disk at which there is a sudden increase of pressure without any discontinuity of velocity was first introduced by R.E. Froude[3] for incompressible flow and is generally known as Froude's actuator disk model. To hold for compressible flow and swirl in the slipstream the actuator disk model is slightly generalized to a zero thickness disk at which there is a sudden increase of momentum and energy of the flow whereas the fluxes are conservative through the disk.
The general input for the actuator disk is a line force distribution which is denominated as distribution of sectional loads. The output in form of an equivalent stationary surface force distribution goes into the governing equations as external force and an external power.

19.3 The Grid Topology

Figure 52 and figure 53 show an analytical result of the velocity and the pressure distribution along the axis of the actuator disk. It was calculated with the potential theory for an incompressible flow with a farfield velocity of $v = 50\,m/s$ and a pressure jump of $\triangle p = 5\,hPa$. The application of an external force via a zero thickness surface leads to jump discontinuities in the state of the flow at the disk whereas the fluxes are conservative through the disk. The objective for the discretization is to maintain these properties, a sharp rendering of jump discontinuities and a conservative balancing of fluxes through the disk. For a sharp rendering of jump discontinuities the grid needs two points at the same geometric position with different sets of variables of state. Because zero edges between two points at the same geometric position are not allowed these points define an inner surface. This surface can be interpreted as an inner periodic boundary with the identity as transformation. Figure 55 shows the topology of the grid. The points of the periodic and the shadow half are geometrical identical but topological different. The boundary is closed with a fix point line similar to the axis line of a rotatory periodic boundary. The fix points on this line are geometrical and topological identical. Figure 56 shows the dual grid cells of a cut section through the disk. Due to the periodicity two boundary cells complete each other to an inner cell with two points. This cell enables the conservative balancing of fluxes through the disk and a sharp rendering of jump discontinuities.

19.4 Grid Generation with Centaur

Before the grid generation starts boundary groups have to be defined for the surface panels of the CAD model. If an actuator disk would be constructed as watertight domain the unique identification of the periodic and the shadow half would not be possible because they are geometrical identical. Instead the actuator disk is constructed as single panel and an

[3]Froude, R.E., On the Part Played in Propulsion by Differences of Fluid Pressure. Trans. Inst. Nav. Arch., Vol. 30, p.390, 1889.

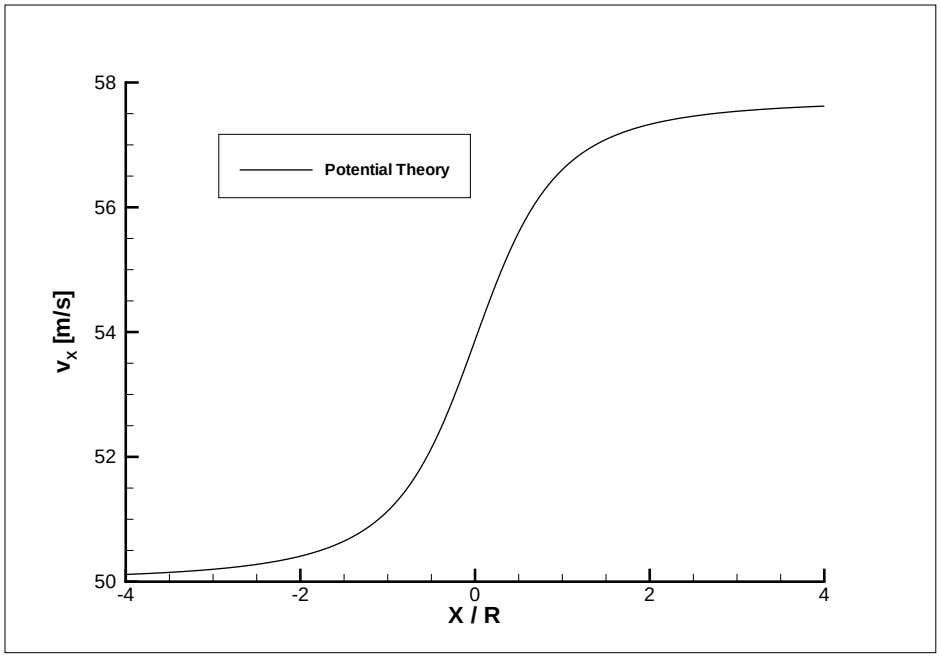


Figure 52: Velocity distribution on actuator disk axis.

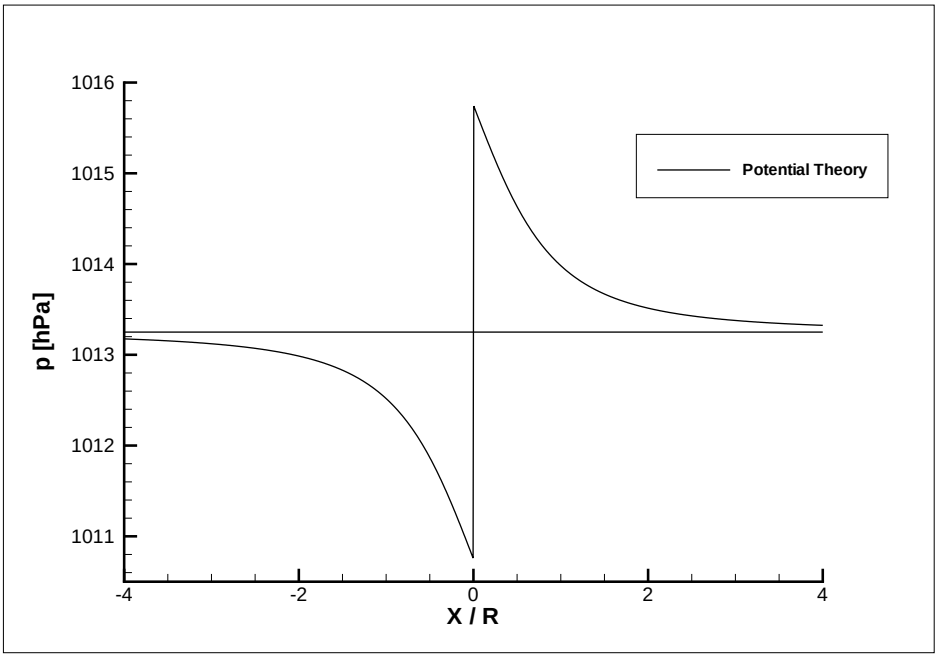


Figure 53: Pressure distribution on actuator disk axis.

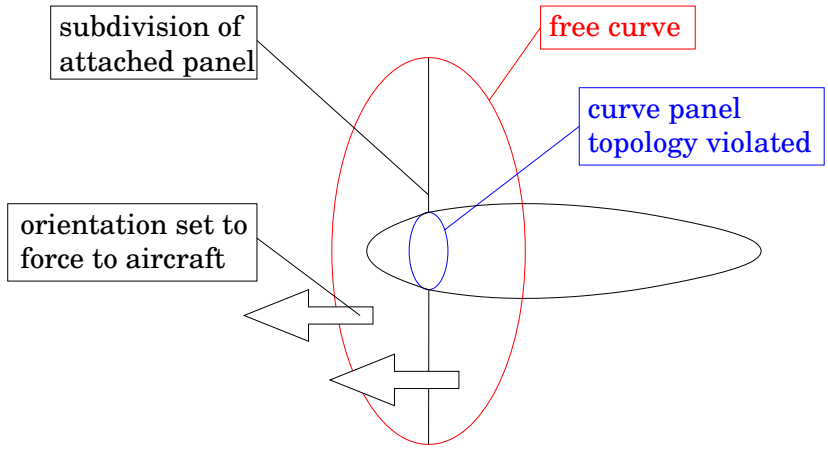


Figure 54: Actuator disk CAD model.

orientation is specified for this panel. In the CAD converter of the grid generator a special actuator group has to be defined which doubles the surface during the grid generation process. It is convention that the doubling takes place to the opposite of the orientation. For actuator disks attached to walls the panel must be subdivided because Centaur does not accept inner trimming loops for panels intersecting with walls. Figure 54 shows the CAD model of an actuator disk with the the Centaur CAD diagnostics and settings.
With the CAD converter *"setupgrid"* one boundary group has to be created for each actuator disk. The boundary condition has to be set to *"actuator"* and the element type to *"Hybrid"* for actuator disks attached to viscous walls and *"Tetrahedra"* in all other cases. The group name in the field *"Group Name"* will be suffixed with *"_cpy"* for the doubled actuator group. The side on which the doubling takes place can be swapped with the toolbar option *"CAD - Panel - sWap UV for Selected Panels"*. The orientation of a panel is graphically shown by a yellow line. The line is the third axis of a right handed coordinate system with the tangential vector of the continuous u-line as first axis and the tangential vector of the dashed v-line as the second axis. The orientation is interpreted from TAU as the direction into which the force is applied to the aircraft. All panels of one group must point into the same direction or the result is undefined.
Because the CAD surface of an actuator disk is non watertight the option *"CAD - Run CAD Diagnostics"* generates error messages. For actuator disks the message *"Free Curves in CAD Model"* must be displayed for all bounding curves which are not intersection line. For the curves of the intersection line the error message *"Curve-Panel Topology Violated in CAD Model"* must be displayed. If this is not the case these curves must explicitly be merged with the attached curves of the wall panels. This can be done with the tool bar option *"CAD - curve - Join Selected Curves"*. These error messages are normal for actuator disks and do not disturb the grid generation process. If the external program CADFix is used to correct CAD data a special option to export free curves in the CAD model has to be activated to

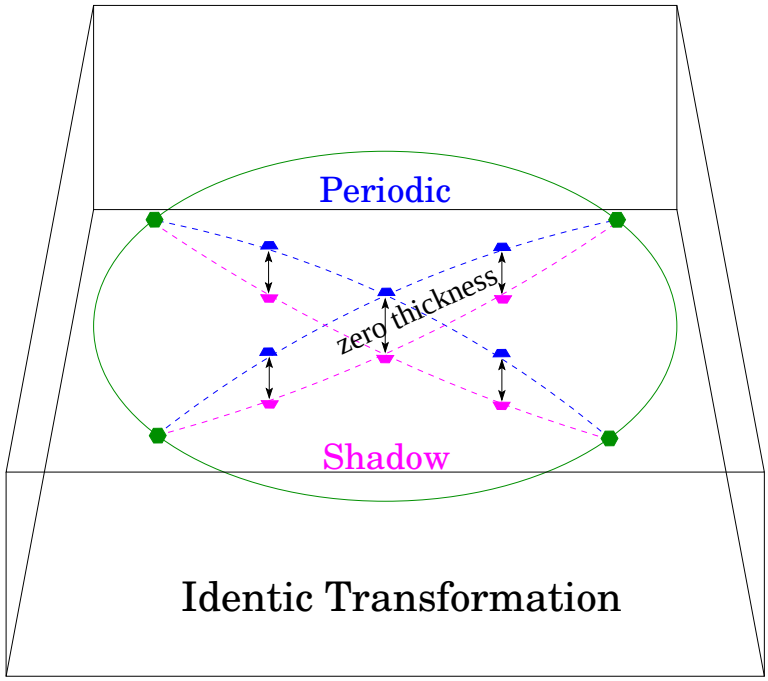


Figure 55: Grid topology of the actuator disk.

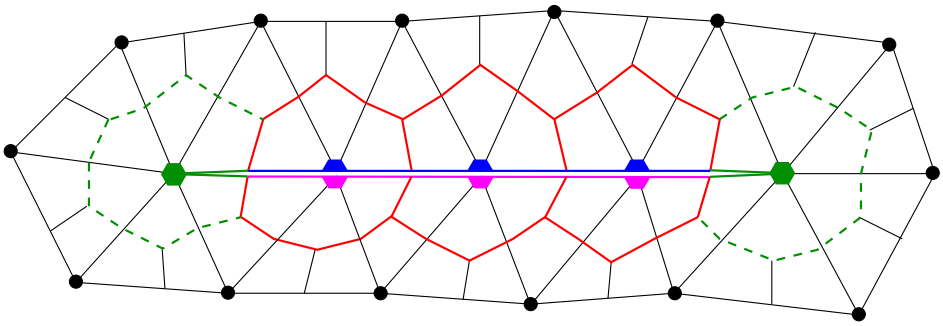


Figure 56: Dualgrid of cut section through the disk.

write the output file in DAT format. If Centaur reads the file the tool bar option *"Options - Switch to IGES mode"* has to be activated.
The Centaur hyb format can be converted with the program *"hybconvert"*. After choosing the output format the question *"Ignore wake surface? (1 - yes, 2 - no)* has to be answered with *"no"* for actuator disk grids. Otherwise actuator disks would be removed because they have been derived from wake surfaces.

19.5 Setup_taugrid

After converting the centaur grid into Tau format with *"centaur2tau"* the grid has to be prepared for Tau with *"setup_taugrid"*. For actuator disks there is the special option *"23: Correct actuator disk"*. This option displays a table with the geometrical data of each actuator disk and the option to correct this data. For actuator disks the axis unit vector, the center and the radius of the disk are stored as geometrical information in the primary grid. If exact information is available from the CAD model this data should be corrected. If this information is not available this is not crucial because the information is calculated with the accuracy of the coordinates of the points at the outer radius of the disk. Based on this geometrical data the points of the periodic half are shifted in axial direction to the disk plane and the coordinates are copied to the periodic partners on the shadow half so that the coordinates are identical up to machine accuracy. If this option is not used the last digits may contain arbitrary values and it can happen that a periodic pair will be separated by the geometric bisectional partitioning algorithm of Tau. This has not been forseen and a parallel run could fail.

19.6 Actuator Disk Coordinate Systems

For parameter input and visualization a local coordinate system is needed for the disk. Because only the axis vector normal to the disk is fixed the axis vectors on the disk plane must be constructed according to a convention. For helicopters there is a standard definition of the local coordinate system for the main and the tail rotor which is shown in figure 57. To avoid confusion the construction of the local coordinate system of the disk is in agreement with these definitions. For zero angles α and β the local system of the disk $(e_{x_d}, e_{y_d}, e_{z_d})$ coincides with the global system of Tau (e_x, e_y, e_z). For the main rotor the final position of the normal unit vector e_{z_d} is reached by a rotation first about the y-axis with the shaft angle α and then by a rotation about the x_d-axis with the angle β. If the inclination of the normal unit vector e_{z_d} to the y-axis is within a cone of 60° the coordinate system is interpreted as tail rotor system. For zero angles α and β this system coincides with the system $(e_x, -e_z, e_y)$ and the final position of e_{z_d} is reached by a rotation first about the z-axis with the angle α and then about the x_d-axis with the angle β. With the axis unit vector being e_{z_d} and the system of Tau (e_x, e_y, e_z) the axis unit vectors in the disk plane can be defined as

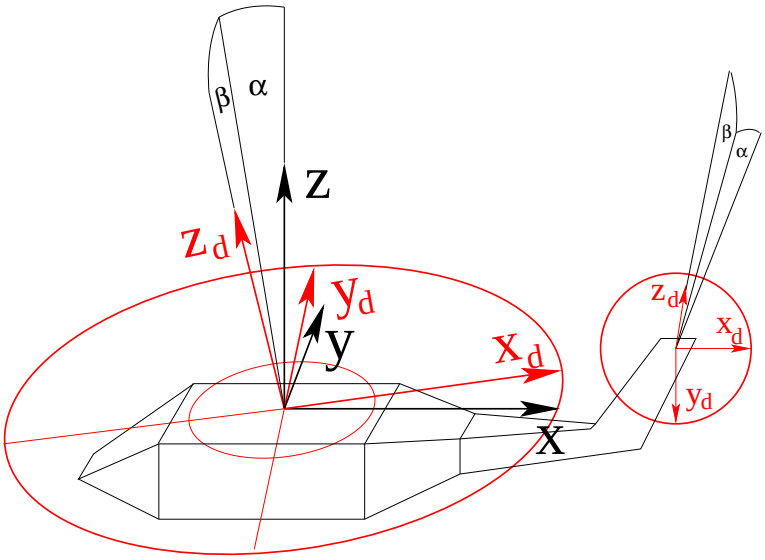


Figure 57: Local coordinate system of the disk for helicopters.

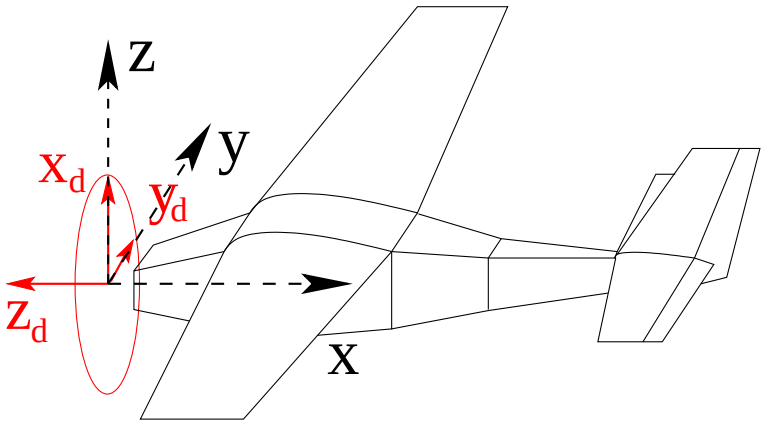


Figure 58: Local coordinate system of the disk for propellers.

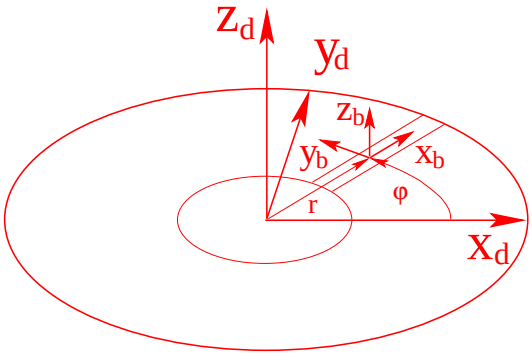


Figure 59: Local coordinate system of the blade section.

$$e_{x_d} = \begin{cases} (e_y \times e_{z_d}) \,/\, \mid e_y \times e_{z_d} \mid & \text{if } \mid \cos(\angle(e_y, ez_d)) \mid < 0.5, \\ (e_{z_d} \times e_z) \,/\, \mid e_{z_d} \times e_z \mid & \text{else} \end{cases} \tag{63}$$

and to get a right-handed coordinate system

$$e_{y_d} = e_{z_d} \times e_{x_d} \,. \tag{64}$$

Because the definition is the same for all axis unit vectors in the xz-plane it is adequate for propellers too. Figure 58 shows the local coordinate system of the disk with $\alpha = 90°, \beta = 0°$ for a typical configuration of an aircraft with propeller. On the disk a local system of the blade section is defined which is shown in figure 59. The system is used for the calculation of forces and for the output. The position of each blade section is specified by the radial distance r to the center of the disk and the azimuth angle ϕ relative to the x_d-axis. The positive direction is defined by the sense of rotation related to the axis unit vector. If nothing is specified the sense of rotation is assumed to be mathematical positive. The axes are defined by the radial unit vector e_{x_b}, the azimuthal unit vector e_{y_b} and the normal unit vector e_{z_b}.

19.7 Parameter Input Solver

For the pair of boundary markers of an actuator disk there are the boundary condition types *"actuator inflow"* and *"actuator exhaust"*. The pair of boundary markers and the geometrical data of the disk as the center, the axis unit vector and the radius R are stored with the periodic information in the primary grid. They need not to be specified in the parameter file. For the actuator exhaust boundary no parameters need to be specified. For the actuator inflow boundary the distribution of sectional loads can be prescribed. If nothing is specified the actuator disk runs without any load and the flow passes undisturbed through the disk. The distribution of sectional loads can either be prescribed directly or calculated with the 2D blade element theory. With a direct prescription the state of the flow depends from the surface load on the disk. With the 2D blade element theory the surface load on the disk depends also from the state of the flow and there is a full coupling.

19.7.1 Format of Tables

The specification of the distribution of sectional loads requires data sets which are specified in the format of a table. The data sets represent a structured grid in a one- or two-dimensional parameter space so the format must allow for the specification of multi-dimensional data sets. Each table begins with a header line containing a keyword optionally followed by a list of tokens

$$\langle keywort \rangle \, [d1 = \langle n_1 \rangle], \, [d2 = \langle n_2 \rangle], \, \ldots$$

Each token consists of a dimension specifier $d\langle k \rangle$, the equal sign and a number n_k specifying the number of rows of the k-th dimension. Each row contains the data set of a grid point $(i_1, \ldots, i_n)$ in the n-dimensional parameter space. The rows are arranged in the sequence of a nested loop

$$\begin{aligned}
&\text{for}(i_1 = 1; i_1 <= n_1; ++i_1) \\
&\quad \text{for}(i_2 = 1; i_2 <= n_2; ++i_2) \\
&\qquad \ldots \\
&\qquad\quad \text{for}(i_n = 1; i_n <= n_n; ++i_n) \\
&\qquad\qquad \text{print line}(i_1, \ldots, i_n)
\end{aligned}$$

The the coordinate i_k of the dimension k varies the more the higher the index of the dimension is. The keyword and the tokens are separated by tabs, blanks or other characters which are not a letter or a digit. The parts of a token are separated by tabs or blanks. Between the header line and the data set there may be newlines or lines containing only tabs or blanks. The data set begins with the first line starting with a number. If no dimensions are specified the data set is one-dimensional and ends with the first line starting not with a number. Tabs and blanks at the beginning of a line are ignored. Columns are separated by tabs or blanks and are detected automatically. Because Tau uses linear interpolation for the mapping of a data set to the unstructured grid of the disk a table must contain 2^D entries if D is the dimension of the data set.

19.7.2 External Input File

The tables required for the prescription of the sectional load can be written into the parameter file or into an external file. If an external file is used the parameter

Input File : -

has to be specified in the actuator inflow boundary mapping with a string value containing the filename with the relative path to the working directory of the code. In the parameter file the tables have to be written into the block of the actuator inflow boundary mapping of the actuator disk. The complete set of tables has to be specified either in the parameter file or into an external file. Mixing both file input methods for one actuator disk has not been forseen.

19.7.3 Direct Prescription of Sectional Loads

For propellers a direct prescription of the distribution of sectional loads is done for preliminary studies if the propeller design is not known in detail. For helicopters the built in blade element theory is not applicable because it does not account for the elasticity of the blade and it assumes that the slipstream is approximately perpendicular to the rotor plane for the calculation of the induced velocity at the blade section. Therefore the sectional load is calculated by external programs with a coupled simulation of a structural model and a simplified rotor aerodynamics.

Parameter Input The data set is specified in the form of a structured grid in polar coordinates as a two-dimensional table. The header line begins with the keyword "sectional_load" followed by the

Markers : -

Type : actuator inflow

Name : -

Write surface data (0/1) : -

Number of blades : -

sectional_load $d1 = n$, $d2 = m$

$\phi_1 \quad r_1/R \quad f_{x_{b11}} \quad f_{y_{b11}} \quad f_{z_{b11}}$

$\phi_1 \quad r_2/R \quad f_{x_{b12}} \quad f_{y_{b12}} \quad f_{z_{b12}}$

. . . . . .

$\phi_1 \quad r_m/R \quad f_{x_{b1m}} \quad f_{y_{b1m}} \quad f_{z_{b1m}}$

$\phi_2 \quad r_1/R \quad f_{x_{b21}} \quad f_{y_{b21}} \quad f_{z_{b21}}$

$\phi_2 \quad r_2/R \quad f_{x_{b22}} \quad f_{y_{b22}} \quad f_{z_{b22}}$

. . . . . .

$\phi_2 \quad r_m/R \quad f_{x_{b2m}} \quad f_{y_{b2m}} \quad f_{z_{b2m}}$

. . . . . .

$\phi_n \quad r_m/R \quad f_{x_{bnm}} \quad f_{y_{bnm}} \quad f_{z_{bnm}}$

block end

TITLE = "COUPLAGE AVEC CFD - EFFORTS AERO 2D"							
VARIABLES = "PSI","R/RAYON","FXAER2D","FYAER2D",							
"FZAER2D","MXAER2D","MYAER2D","MZAER2D"							
ZONE T = "EFFORTS 2D", I=m J=n F=POINT							
ϕ_1	r_1/R	$f_{x_{b11}}$	$f_{y_{b11}}$	$f_{z_{b11}}$	$m_{x_{b11}}$	$m_{y_{b11}}$	$m_{z_{b11}}$
ϕ_1	r_2/R	$f_{x_{b12}}$	$f_{y_{b12}}$	$f_{z_{b12}}$	$m_{x_{b12}}$	$m_{y_{b12}}$	$m_{z_{b12}}$
.....							
ϕ_1	r_m/R	$f_{x_{b1m}}$	$f_{y_{b1m}}$	$f_{z_{b1m}}$	$m_{x_{b1m}}$	$m_{y_{b1m}}$	$m_{z_{b1m}}$
ϕ_2	r_1/R	$f_{x_{b21}}$	$f_{y_{b21}}$	$f_{z_{b21}}$	$m_{x_{b21}}$	$m_{y_{b21}}$	$m_{z_{b21}}$
ϕ_2	r_2/R	$f_{x_{b22}}$	$f_{y_{b22}}$	$f_{z_{b22}}$	$m_{x_{b22}}$	$m_{y_{b22}}$	$m_{z_{b22}}$
.....							
ϕ_2	r_m/R	$f_{x_{b2m}}$	$f_{y_{b2m}}$	$f_{z_{b2m}}$	$m_{x_{b2m}}$	$m_{y_{b2m}}$	$m_{z_{b2m}}$
.....							
ϕ_n	r_m/R	$f_{x_{bnm}}$	$f_{y_{bnm}}$	$f_{z_{bnm}}$	$m_{x_{bnm}}$	$m_{y_{bnm}}$	$m_{z_{bnm}}$

Table 10: HOST output file in Tecplot ASCII format.

dimension "d1" specifying the number of azimuthal coordinates and the dimension "d2" specifying the number of radial coordinates. Each line holds the data set of a grid point and contains the azimuthal coordinate ϕ, the radial position r scaled with the radius R of the disk and the components of the sectional force $(f_{x_b}, f_{y_b}, f_{z_b})$ in the local system of the blade section as shown in figure 57. The azimuthal coordinate is specified in [°] and the components of the sectional force in $[N/m]$. The coordinates are ordered to increasing size, $\phi_i < \phi_{i+1}$ and $r_j/R < r_{j+1}/R$. The azimuthal coordinate counts the angle 0° and 360° double. To complete the data set the number of blades has to be specified.

HOST Output Files in Tecplot ASCII Format For the external program HOST from Eurocopter there exists a special interface to read output files in Tecplot ASCII format. A file is recognized as a file in HOST output format if the filename contains the string "PALE". This corresponds to the standard naming convention of HOST to prefix these output files with this string. The format is shown in table 10. It represents a structured grid in polar coordinates. Each line corresponds to a grid point and contains the azimuthal coordinate ϕ, the radial coordinate r scaled with the radius R of the disk, the components of the sectional

Markers : -

Type : actuator inflow

Name : -

Input file : PALE...

Write surface data (0/1) : -

Number of blades : -

force $(f_{x_b}, f_{y_b}, f_{z_b})$ and the sectional torque $(m_{x_b}, m_{y_b}, m_{z_b})$ in the local system of the blade section as shown in figure 59. The azimuthal coordinate is specified in [°], the components of the sectional force in $[N/m]$ and the components of the sectional torque in $[N]$. The sectional torque is not read or interpreted by Tau. Contrary to the dimension specification in the parameter file the dimension specifier in the header line I refers to the inner loop and the dimension specifier J to the outer loop.

19.7.4 2D Blade Element Theory

With the blade element theory the distribution of sectional loads is calculated as sectional lift l and sectional drag d of the blade over the azimuth of the disk. The sectional lift l and sectional drag d are calculated as

$$l = \frac{\rho}{2} {v_e}^2 c\, C_l(\alpha_e), \tag{65}$$

$$d = \frac{\rho}{2} {v_e}^2 c\, C_d(\alpha_e). \tag{66}$$

The kinematics of the flow at the blade section is shown in figure **??**. The effective angle of attack α_e is the difference of the blade twist β and the induced angle of attack α_i

$$\alpha_e = \beta - \alpha_i. \tag{67}$$

The induced angle of attack α_i and the effective velocity v_e depend on the rotational speed Ωr and the induced velocity of the flow with its normal component v_n and tangential component v_t

$$\alpha_i = \arctan\left[\frac{-v_n}{v_t - \Omega r}\right], \qquad v_e = \sqrt{(v_t - \Omega r)^2 + v_n^2}. \tag{68}$$

For a given rotational speed Ωr and increasing flight velocity the normal velocity v_n through the disc and the induced angle of attack α_i increase. For fixed twist β the effective angle of

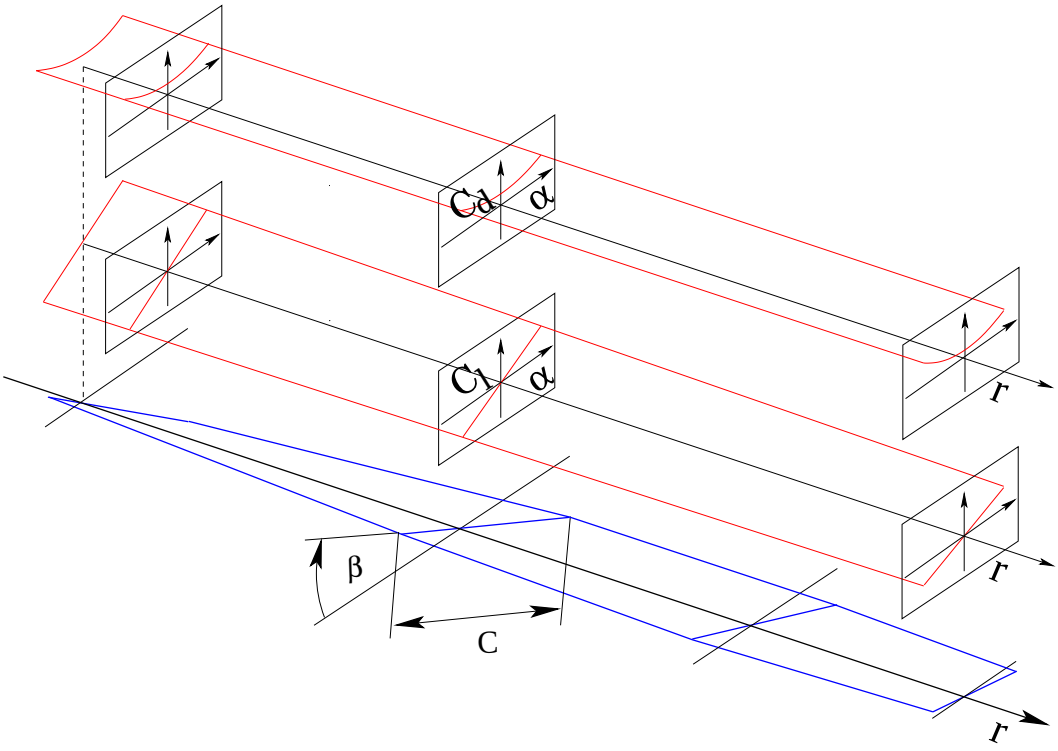


Figure 60: Parametrization of the blade.

attack α_e and the thrust of the propeller decrease. This explains why the thrust of a propeller with constant pitch and rpm decreases with increasing flight velocity. The sectional lift l and drag d are transformed to the azimuthal component f_{y_b} and the normal component f_{z_b} in the local system of the blade section

$$f_{y_b} = -l\sin(\alpha_i) - d\cos(\alpha_i), \tag{69}$$
$$f_{z_b} = \quad l\cos(\alpha_i) - d\sin(\alpha_i). \tag{70}$$

The normal component f_{z_b} is the sectional thrust and causes the acceleration of the slip stream. The product $f_{y_b}r$ is the sectional torque and causes the swirl of the slipstream.

Parameter Input The parameter set for the 2D blade element theory contains global paramters referring to the propeller and tables referring to the geometry and the aerodynamic properties of the blade. Parameters referring to the propeller are the propeller rpm, the number of blades, the pitch point and the pitch angle. The sign of the propeller rpm defines the sense of rotation. From a pilots view into flight direction no sign or a positive sign is interpreted as a clockwise rotation and a negative sign is interpreted as an anticlockwise rotation. The flight direction is defined by the axis unit vector of the disk. The pitch point specifies the radial position r scaled with the radius of the disk R at which the pitch angle

Markers : -

Type : actuator inflow

Name : -

Write surface data (0/1) : -

Propeller rpm : -

Number of blades : -

Pitch point : -

Pitch angle : -

r/R_c/R_beta

$r_1/R \quad c_1/R \quad \beta_1$

$r_2/R \quad c_2/R \quad \beta_2$

...

is set. Typical values for the pitch point are $r/R = \{0.7, 0.75\}$. The pitch angle has to be specified in [°].
As shown in figure 60 the geometry of the blade is parametrized with the chord length c and the twist angle β which is the angle of the chord relative to the propeller plane. The parameters are a function of the radial position r along the blade. The aerodynamic properties of the blade are characterized by the aerodynamic coefficient functions of lift $C_l(\alpha)$ and drag $C_d(\alpha)$. The aerodynamic coefficient functions either can be approximated by polynoms or be written as a discrete function. For the discretization each parameter set is specified for a finite number of blade sections and linearly interpolated in between and extrapolated at the boundaries. The geometry parameters are specified in the form of a one-dimensional table. This table starts with the keyword "r/R_c/R_beta". Each line contains the radial position r scaled with the radius R of the disk, the chord length c scaled with the radius R of the disk and the blade twist in [°].

Polynomial Approximation of the Aerodynamic Coefficient Functions of Lift and Drag The aerodynamic coefficient functions are approximated by polynoms at the blade

r/R_cl

$r_1/R \quad C_{l_{0_1}} \quad C_{l_{1_1}} \quad \cdots$

$r_2/R \quad C_{l_{0_2}} \quad C_{l_{1_2}} \quad \cdots$

$\cdots$

r/R_cd

$r_1/R \quad C_{d_{0_1}} \quad C_{d_{1_1}} \quad C_{d_{2_1}} \quad \cdots$

$r_2/R \quad C_{d_{0_2}} \quad C_{d_{1_2}} \quad C_{d_{2_2}} \quad \cdots$

$\cdots$

block end

sections

$$C_l(\alpha) = \sum_i C_{l_i} \alpha^i, \qquad C_d(\alpha) = \sum_i C_{d_i} \alpha^i. \tag{71}$$

and the polynomial coefficients of each aerodynamic coefficient function are written to a one-dimensional table. The table of the aerodynamic coefficient function of lift starts with the keyword "r/R_cl". Each line contains the radial position r scaled with the radius R of the disk and the polynomial coefficients of lift in a sequence of increasing power. The coefficient C_{l_i} is specified in $[(1/^\circ)^i]$. The table of the aerodynamic coefficient function of drag starts with the keyword "r/R_cd". Each line contains the radial position r scaled with the radius R of the disk and the polynomial coefficients of drag in a sequence of increasing power. The coefficient C_{d_i} is specified in $[(1/^\circ)^i]$.

Discrete Function Approximation of the Aerodynamic Coefficient Functions of Lift and Drag The aerodynamic coefficients of lift $C_l(\alpha)$ and drag $C_d(\alpha)$ are specified for a series of angles of attack α at each blade section and written as a structured grid of quadrilaterals in the two-dimensional parameter space (r, α). The structure implies that each blade section must have the same number of alpha coordinates. In this version of the code the grid is additionally supposed to be cartesian. This means each blade section must have the same α-coordinates. In the next version this assumption is abandoned. The grid is specified in the form or a two-dimensional table. The header line begins with the keyword "r/R_alpha_cl_cd" followed by the dimension "d1" specifying the number of radial positions of the blade sections and the dimension "d2" specifying the number of alphas for each blade

r/R_alpha_cl_cd $d1 = n$, $d2 = m$

$r_1/R \quad \alpha_1 \quad C_{l_{11}} \quad C_{d_{11}}$

$r_1/R \quad \alpha_2 \quad C_{l_{12}} \quad C_{d_{12}}$

. . . . . .

$r_1/R \quad \alpha_m \quad C_{l_{1m}} \quad C_{d_{1m}}$

$r_2/R \quad \alpha_1 \quad C_{l_{21}} \quad C_{d_{21}}$

$r_2/R \quad \alpha_2 \quad C_{l_{22}} \quad C_{d_{22}}$

. . . . . .

$r_2/R \quad \alpha_m \quad C_{l_{2m}} \quad C_{d_{2m}}$

. . . . . .

$r_n/R \quad \alpha_m \quad C_{l_{nm}} \quad C_{d_{nm}}$

block end

section. Each line contains the data set of a grid point. The data set contains in a sequence the the radial coordinate r/R scaled with the radius R of the disk, the α-coordinate specified in [°], the coefficient of lift C_l and the coefficient of drag C_d. The coordinates are ordered to increasing size, $r/R_i < r/R_{i+1}$ and $\alpha_j < \alpha_{j+1}$. The aerodynamic coefficients C_l and C_d are bi-linearly interpolated inside the quadrilateral elements of the grid and bi-linearly extrapolated outside at the boundary.

19.7.5 Parameter Input Preprocessing

Markers : -

Type : actuator inflow

Name : -

Blank ratio : 0.035

For actuator disks intersecting with a viscous wall the no-slip boundary condition requires a

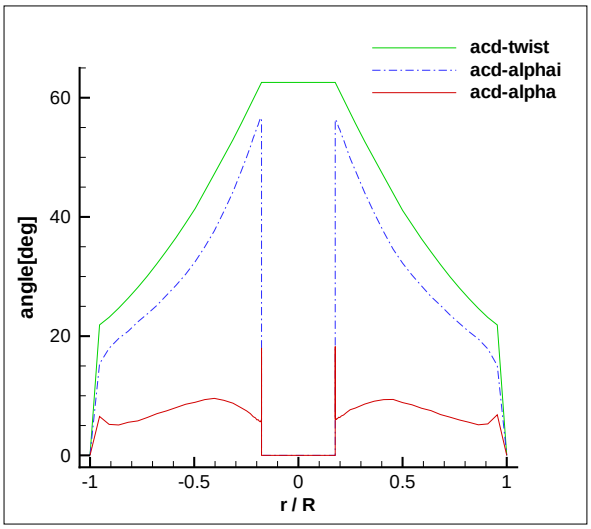


Figure 61: Cut section of an actuator disk intersecting with a viscous wall.

zero sectional load for the points of the intersection line. Otherwise a non-physical pressure jump occurs which is destabilizing the numerical scheme. The sectional load calculated with the blade element theory does not provide a smooth transition to zero in the boundary layer. The velocity of the flow becomes zero at the wall but the rotational speed assumed for the blade element theory is not visible for the flow and therefore not affected by the boundary layer. According to figure **??** the effective angle of attack converges to the blade twist at the wall. The high twist angle causes unphysical high values for the aerodynamic coefficients of lift and drag and because of the rotational velocity also for the sectional load. To obey the limiting condition the sectional load is set to zero for the points of the intersection line and the points are excluded from the prescription of the sectional load. Figure 61 shows the distribution of the blade twist, the induced and effective angle of attack of a radial section through a disk intersecting with a viscous hub. The induced angle of attack α_i which is the angle between the propeller plane and the vector of the relative velocity at the blade section converges rapidly to zero due to the influence of the boundary layer. As a consequence the effective angle of attack α as the difference of the blade twist and the induced angle of attack α_i shows a sharp peak close to the wall. The peak does not reach the maximum value of the blade twist at the hub because the points of the intersection line are excluded from the application of the blade element theory and the angle has been set to zero for the visualization.

To get a robust scheme for the application of the blade element theory the near wall region is blanked for the prescription of the sectional load. For a smooth transition in the blanked region 1D ray coordinate systems are constructed for the intersection points in the structured region of the boundary layer with an advancing front algorithm. The 1D ray coordinate

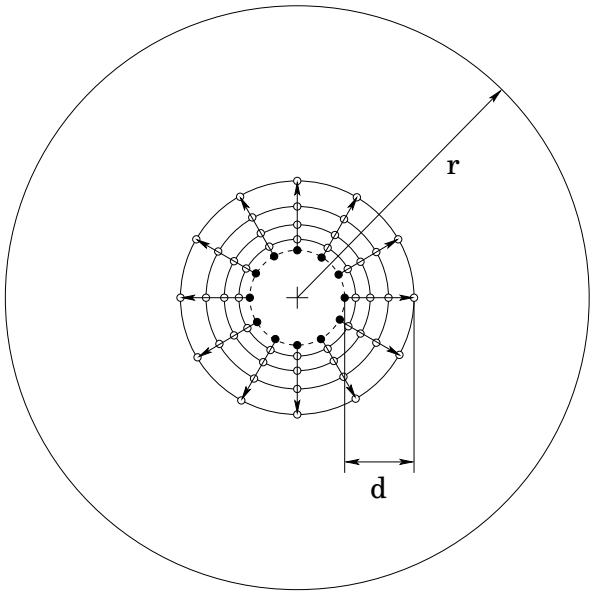


Figure 62: 1D ray coordinate systems of an actuator disk intersecting with a viscous wall.

systems are shown in figure 62. The advancing front stops if the ratio of the extension of the coordinate system d and the radius of the disk r exceeds the value of the parameter "Blank ratio" specified in the boundary mapping file or if the structured layer of quadrilateral elements ends. The last point of each ray is the closest point to the wall for which the sectional load is calculated with the blade element theory or externally prescribed. For each ray the sectional load of this point is linearly scaled with the radial distance to the intersection point. The default value of this parameter has been set to 0.035. This gives a good result for a wide variation of the inner radius. If the blanked region is not sufficient the value can be increased. The interpolation area can be visualized with the surface output parameter "acd_raysc" which plots the scaling factor of the ray points and zero for all other points. Ray coordinate systems are also constructed for the intersection with a viscous wall at the outer radius of the disk. For a direct prescription of the sectional load the ray scaling forces a smooth transition for an arbitrary distribution of sectional loads. This is convenient if the distribution of sectional loads comes from an external source. If a smooth transition for the intersection with a viscous wall already has been taken into account for the distribution of sectional loads this parameter should be set to zero.

19.7.6 Output

Surface Output For actuator disks there are special surface output functions:

acd-raysc	– ray scaling function,
acd-chlen	– chord length,
acd-twist	– blade twist,
acd-vrot	– rotational speed,
acd-alphai	– induced angle of attack,
acd-alpha	– effective angle of attack,
acd-cl	– coefficient of lift,
acd-cd	– coefficient of drag,
acd-secload	– (vector) sectional load in tau system,
acd-dsk-xyz	– (vector) coordinates in disk system,
acd-bld-vel	– (vector) flow velocity in blade system,
acd-bld-secload	– (vector) sectional load in blade system

These output functions are activated in the actuator inflow boundary mapping by setting the parameter:

Write surface data (0/1) : 1

They are specified in the parameter file:

Surface output description file : (thisfile)

Surface output values : acd-vrot_acd-chlen_acd-twist ...

The function *"acd-raysc"* shows the scaling factors of the 1D ray coordinate systems of the disk. Values of grid points which are not part of rays are initialized with zero. This function visualizes the near wall area of the disk where the sectional load is not calculated or prescribed and allows if necessary to adapt the preprocessing parameter *"Blank ratio"*. The remaining scalar output functions are only defined in conjunction with the 2D blade element theory. These functions allow to check if the operating mode of the propeller is within its physical range. To get independent of the position of the disk in the global system of Tau the coordinates of the disk have to be written in the local system of the disk $(e_{x_d}, e_{y_d}, e_{z_d})$ by specifying the output function *"acd-dsk-xyz"*. For convenience the coordinates are normed

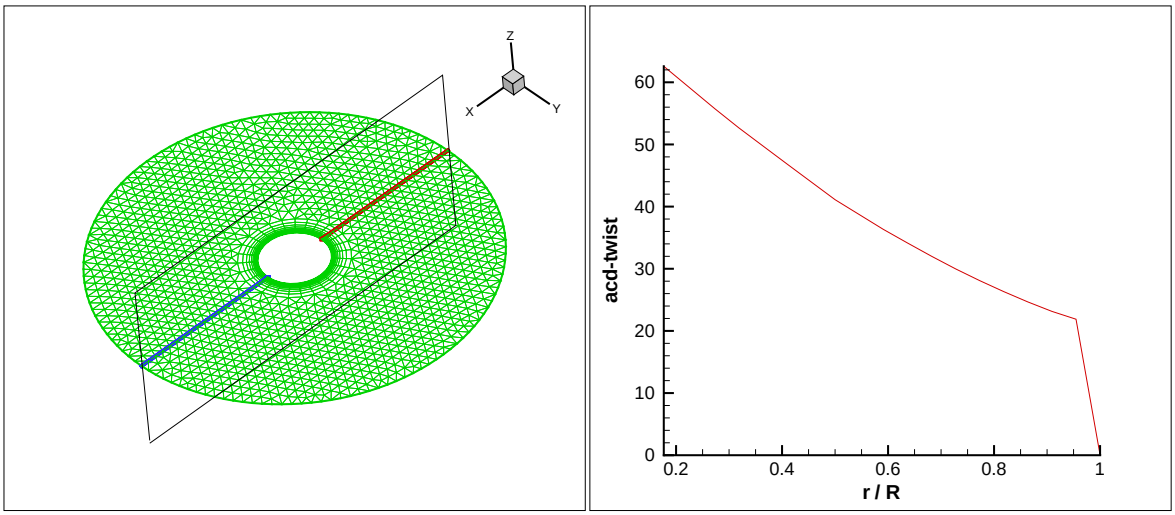


Figure 63: Zone of constant azimuth and xy-plot over radius.

with the radius of the disk. For the analysis of the flow at the blade there are output functions which write vector components in the local system of the blade section. These functions have the prefix *"acd-bld-"*. There are output functions for the sectional load and the velocity vector of the flow in the local system of the blade section. To keep track of the blade over one revolution Tecplot can define cut planes rotating around an axis. In the local system of the disk $(e_{x_d}, e_{y_d}, e_{z_d})$ the rotation around the z_d-axis at the origin generates intersection lines of constant azimuth angle ϕ. The extracted lines can be plotted as xy-plot over the radius. This is shown in figure 63. The procedure to create an xy-plots over the radius for a constant azimuth angle ϕ is

- Specify *"Surface output values: acd-dsk-xyz_..."* in the parameter file. Open TecPlot with the surface solution file and set the plot type to *"3D Cartesian"*.
- Open the *"Plot–Assign XYZ..."* menu. Assign *"X: acd-dsk-x"*, *"Y: acd-dsk-y"* and *"Z: acd-dsk-z"*.
- Open the *"Data–Alter–Specify Equations"* menu and specify the the equation {r/R} = SQRT({acd-dsk-x}**2 + {acd-dsk-y}**2).
- Open the *"Data–Extract–Slice from Plane"* menu. Switch off *"Force Extraction to Single Zone"* to create separate zones for both azimuth angles intersecting with the plane. Choose *"Arbitrary"* and *"Origin and Normal"*. Specify *"Slice Origin: 0 0 0"* and *"Slice Normal: 0 1 0"*.
- In the field *"Rotate About"* set *"Stepsize(deg)"* and rotate the cut plane to the dedicated azimuthal position. Extract the lines.

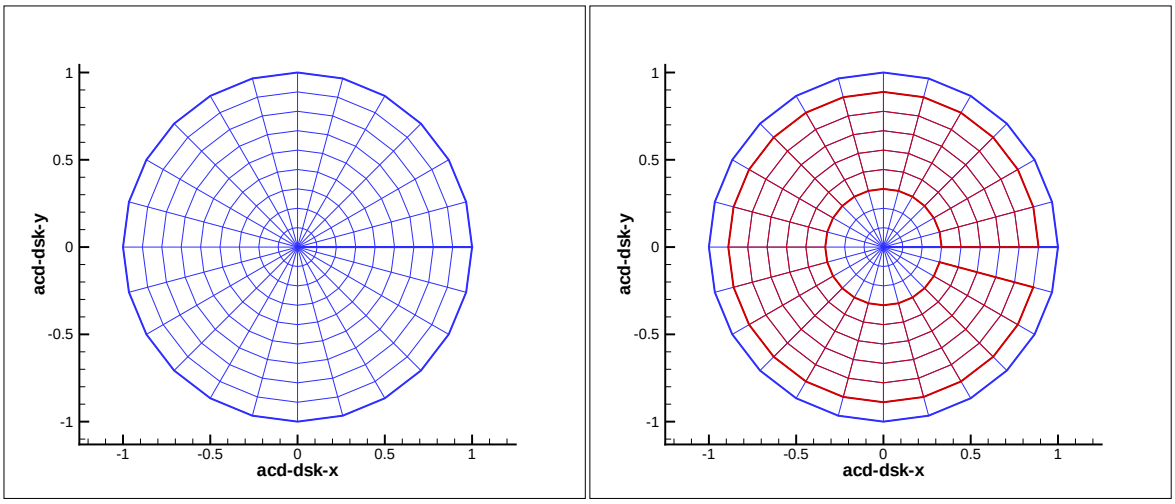


Figure 64: Circular zone and extracted circular subzone.

- Open the *"Plot–Zone Style"* menu and activate the extracted zones to get the correspondent to the azimuthal position.
- Set the plot type to *"XY Line"*. Choose *"Mapping Style ..."* and set *"X-Axis Variable: r/R"*, *"Y-Axis Variable: PlotVar"*, *"Zone: LineZone"* and activate the mapping.

To get a survey of all azimuthal positions it is convenient to create a plot in polar coordinates. Polar coordinates are discontinuous along a line where the azimuth switches back. Elements intersecting this line cannot reasonably be plotted. It is not straightforward to exclude elements and split up the closed topology of the unstructured grid for the output. Instead of changing the grid topology TecPlot has the option to create a circular zone. This is a structured 2D grid in cartesian coordinates as it is used for plots with polar coordinates. From this zone a sub zone can be extracted by specifying I-ranges for the radial lines and J-ranges for the azimuthal lines. Figure 64 shows the circular zone and an extracted subzone. So the closed topology in azimuthal direction can be split up and the inner hole of the unstructured grid can be mapped to the subzone. From the unstructured grid all variables are linearly interpolated to the structured grid of the subzone. From the subzone single lines can be extracted so that further interpolation is not necessary. Without harm the interpolated coordinates of the subzone can now be transformed to polar coordinates. In polar coordinates the subzone can be plotted as rectangular zone. Figure 65 shows the mapping of the unstructured grid to the structured grid of the circular subzone and a 3D plot in polar coordinates.
Propellers with a negative rpm require a positive azimuth ϕ in clockwise circumferential direction. To apply the same transformation from cartesian to polar coordinates as for anticlockwise rotation the grid has to be mirrored at the x_d-axis by reverting the sign of

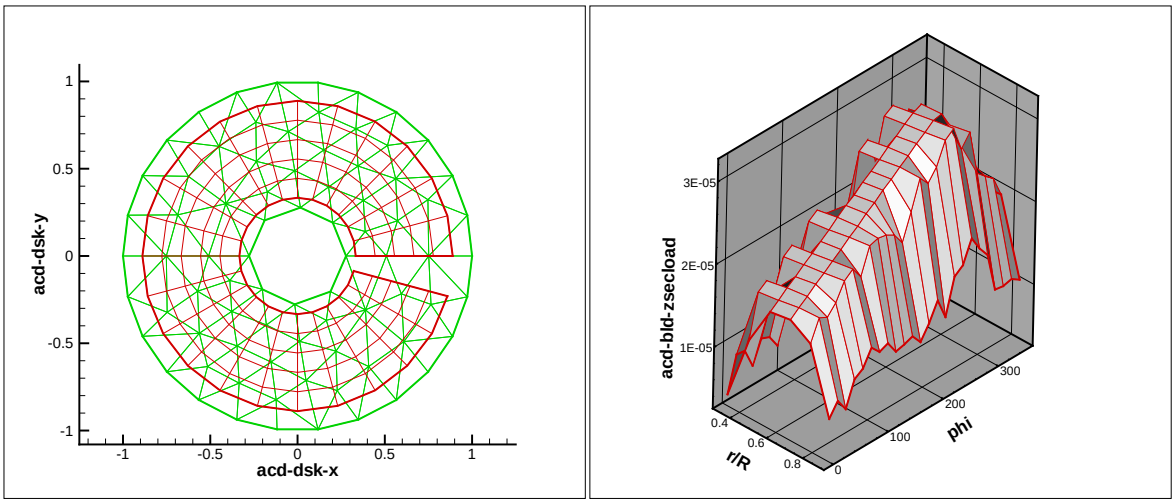


Figure 65: Circular subzone of disk and 3D plot in polar coordinates.

the y_d-coordinates in the *"Data–Alter–Specify Equations"* menu. The general procedure for plots of the disk in polar coordinates is

- Specify *"Surface output values: acd-dsk-xyz_..."* in the parameter file and open TecPlot with the surface solution file.
- Open the *"Plot–Assign XY..."* menu. Assign *"X: acd-dsk-x"* and *"Y: acd-dsk-y"*.
- For clockwise rotating propellers with negative rpm: Open the *"Data–Alter–Specify Equations"* menu. Mirror the local y-axis by specifying the equation {acd-disk-y} = -{acd-disk-y}.
- Open the *"Data–Create Zone–Circular"* menu. Set the origin to zero and the radius to 1.0. Determine the number of lines in radial and circumferential direction and create a circular zone.
- Open the *"Data–Extract–Sub-Zone"* menu. Select the circular zone. Set the range for the radial I-index so that the smallest and the biggest radial line have a distance of one element layer from the boundary of the disk. So the linear interpolation to the subzone is precise. The boundary of the disk is not treated as part of the actuator disk. Set the upper limit of the circumferential J-index to *"Mx-1"* to split up the grid topology and create the subzone.
- Open the *"Data–Delete–Zone"* menu and delete the circular zone.
- Open the *"Data–Interpolate–Linear"* menu. Select the actuator disk zone as source zone. Select the subzone as destination zone and interpolate all variables.

- Open the *"Data–Alter–Specify Equations"* menu. Load the equations from the path specified below or type the specification below manually and save it. Transform the local variables of the disk to the polar coordinates r/R and ϕ.
- Open the *"Plot–ZoneStyle"* menu. Select the subzone and set the plot type to *"3D Cartesian"*.
- Open the *"Plot–Assign XYZ..."* menu. Assign *"X: r/R"*, *"Y: phi"* and *"Z: PlotVar"*.

The equations are stored in the file *"taudir/contrib/acd_polar.eqn"*. If this file is not available or cannot be adapted the equations can manually be typed with the editor of the *"Data–Alter–Specify Equations"* menu from the specification below. The range of the azimuth is $0 \leq \phi < 360°$. The sin and cos functions are inverted in the range of a slope of approximately one to minimize amplification of errors. The formula editor does not forsee distinction of cases so the sign function has been used instead. Each equation has to be written into one line because line breaks are not accepted.

```
   {PI} = 2 * ACOS(0.0)
  {r/R} = SQRT({acd-dsk-x}**2 + {acd-dsk-y}**2)
 {cosp} = {acd-dsk-x} / {r/R}
 {sinp} = {acd-dsk-y} / {r/R}

   {V1} = 0.25 * (1 + SIGN({cosp} - 0.7))
        * ( (1 + SIGN({sinp})) * ASIN({sinp})
           +(1 - SIGN({sinp})) * (2 * {PI} + ASIN({sinp})))

   {V2} = 0.25 * (SIGN({cosp} + 0.7) - SIGN({cosp} - 0.7))
        *  ( (1 + SIGN({sinp})) * ACOS({cosp})
            +(1 - SIGN({sinp})) * (2 * {PI} - ACOS({cosp})))

   {V3} = 0.50 * (1 - SIGN({cosp} + 0.7))
        *  ({PI} - ASIN({sinp}))

  {phi} = ({V1} + {V2} + {V3}) * 180 / {PI}
```

Integral Values At the end of a calculation two tables are printed to the standard output containing force coefficients of the actuator disks in the grid. Table 11 contains the global disk number, the marker pair actuator inflow - actuator exhaust, the radius R, the thrust coefficient C_t, the power coefficient C_p, the force coefficient C_f and the total thrust F of the disk. The thrust coefficient C_T, the power coefficient C_P and the force coefficient C_F are

defined as

$$C_T = \frac{T}{\rho\, n^2 D^4}, \tag{72}$$

$$C_P = \frac{P}{\rho\, n^3 D^5}, \tag{73}$$

$$C_F = \frac{T}{\rho/2\ \mathrm{v}^2 A}. \tag{74}$$

The thrust T is the integral force of the actuator disk projected to the axis. $D = 2R$ is the diameter of the disk and n is the number of revolutions per second. The power P is the power applied to the axis of rotation which is the integral torque Q related to the axis times the angular velocity $\omega = 2\pi/n$. The density ρ is specified by the parameter *"Reference density"*, the velocity v by the parameter *"Reference velocity"* and the area A by the parameter *"Reference relation area"* in the parameter file. The thrust coefficient C_T and the power coefficient C_P are typically used for propellers whereas the force coefficient C_F is used for helicopter rotors. Because the thrust coefficient C_F is not scaled with specific data of the actuator disk it can be added up. Table 12 contains the force and torque coefficients in the aerodynamic system. The force coefficients C_{Fi} and torque coefficients C_{Mi} are defined as

$$C_{Fi} = \frac{F_i}{\rho/2\ \mathrm{v}^2 A}, \tag{75}$$

$$C_{Mi} = \frac{M_i}{\rho/2\ \mathrm{v}^2 A\, l}. \tag{76}$$

The components F_i are the components of the integral force and the components M_i are the components of the integral moment of the aerodynamic system. The index i stands for $\{x_a, y_a, z_a\}$. The Moment is related to a reference point specified by the parameters *"Origin coordinate x"*, *"Origin coordinate y"* and *"Origin coordinate z"* in the parameter file. The density ρ is specified by the parameter *"Reference density"*, the velocity v by the parameter *"Reference velocity"* and the area A by the parameter *"Reference relation area"* in the parameter file. The reference length l for the y_a-axis is specified by the parameter *"Reference length (pitching momentum)"* and for the x_a-axis and z_a-axis by the parameter *"Reference length (rolling/yawing momentum)"* in the parameter file.
The aerodynamic system $(e_{x_a}, e_{y_a}, e_{z_a})$ is shown in figure 66. For a zero angle of attack α and yaw angle β the axis of the aerodynamic system coincide with the axis of the body fixed system $(e_{x_f}, e_{y_f}, e_{z_f})$. The body fixed system $(e_{x_f}, e_{y_f}, e_{z_f})$ is the reference system of the aircraft. The x_f-axis is into the direction of the fuselage oriented into flight direction, the y_f-axis into the direction of the span width oriented to the right wing and the z_f-axis orthogonal to the x_f- and y_f-axis oriented to the ground. The body fixed system is identical with the Tau system (e_x, e_y, e_z) as the design system of the aircraft except that the x_f- and z_f-axis are oriented to the opposite direction. The x_a-axis of the aerodynamic system is parallel to the

Disk	Marker	R[m]	C_T	C_P	C_F	F[N]
0	9, 14	0.1474	0.2643	0.3789	0.0405	92.2
1	10, 15	0.1474	0.2115	0.3032	0.0324	73.8
Sum					0.0730	166.0

Table 11: Actuator disk propeller coefficients.

Disk	Marker	C_{Fx}	C_{Fy}	C_{Fz}	C_{Mx}	C_{My}	C_{Mz}
0	9, 14	4.0e-2	-5.5e-8	-9.6e-9	2.7e-3	-1.6e-7	7.6e-9
1	10, 15	3.2e-2	5.0e-7	5.0e-6	-2.1e-3	5.4e-6	1.0e-7
Sum		7.2e-2	4.5e-7	5.0e-6	5.4e-4	5.3e-6	1.0e-7

Table 12: Force and torque coefficients in the aerodynamic system.

vector of the flight velocity v. The final position of the x_a-axis is reached first by a clockwise positive rotation around the y_f-axis with the angle α and then by an anticlockwise positive rotation around the z_a-axis with the angle β. The anticlockwise positive definition of the yaw angle β in Tau is in opposite to the norm LN9300 where it is defined clockwise positive. In the farfield boundary mapping the angle of attack α is specified by the parameter `"Angle alpha (degree)"` and the yaw angle β by the parameter `"Angle beta (degree)"`.

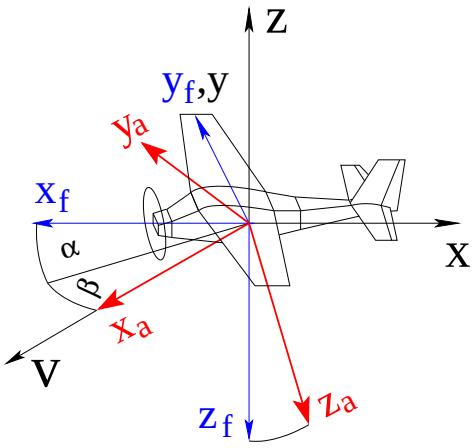


Figure 66: Aerodynamic system.

Index

G

H

I

N

O

P

R

S

T

U

V

W

X

References

[1] A.Garbaruk, M.Shur, M.Strelets, and A.Travin. Preliminary report on des method investigation. Technical report, Desider, Deliverable D3.2.12, 2004.

[2] A.K.Travin, M.L.Shur, and M.Kh.Strelets. Advances in les of complex flows. chapter Physical and Numerical Upgrades in the Detached-Eddy Simulation of Complex Turbulence Flows, pages pp239–254. Kluwer Academic Publishers, 2002.

[3] A.K.Travin, M.L.Shur, P.R.Spalart, and M.K. Strelets. Improvement of delayed detached-eddy simulation for les with wall-modelling. In *European Conference on Computational Fluid Dynamics, ECCOMAS CFD 2006*, 2006.

[4] J. D. Anderson. *Fundamentals of Aerodynamics*. McGraw-Hill, 2001.

[5] D. Chapman. Computational aerodynamics development and outlook. *AIAA*, 17:pp1293–1313, 1979.

[6] C.R. Doering and J.D. Gibson. *Applied Analysis of the Navier-Stokes Equations*. Cambridge University Press, Edinburgh Building, Cambridge, UK, 1995 edition, 1995.

[7] F.Ducros, F. Nicoud, and T.Poinsot. Wall-adapting local eddy-viscosity models for simulations in complex geometries. In *6th ICFD Conference on numerical methods for fluid dynamics*, pages pp293–299*, 1998.

[8] J.C.Kok, H.S.Dol, B. Oskam, and H. Van der Ven. Extra-large eddy simulation of massively separated flows. In *42nd AIAA Aerospace Meeting*, pages pp1–12, Reno, NV, 2004.

[9] M.Breuer, N.Jovii, and K.Mazaev. Comparison of des, rans, and les for the separated flow around a flat plate at high incidence. *Int. Jnl. of Numerical Methods in Fluids*, 41(4):pp357–388, 2003.

[10] P.Moin and J.Kim. Numerical investigation of turbulenct channel flow. *J. Fluid Mech.*, 118:pp341–377, 1982.

[11] S.B. Pope. *Turbulent Flows*. Cambridge University Press, Edinburgh Building, Cambridge, UK, 2001 edition, 2001.

[12] P.R.Spalart. Young person's guide to detached-eddy simulation grids. Technical report, NASA-2001-cr211032, 2001.

[13] P.R.Spalart and C.L.Rumsey. Effective inflow conditions for turbulence models in aerodynamic calcualtions. *AIAA J.*, 45:pp2544–2553, 2007.

[14] P.R.Spalart, S. Deck, M.L. Shur, K.D. Squires, M.Kh.Strelets, and A.Travin. A new version of detached-eddy simulation resistant to ambiguous grid densities. *Theoretical and Computational Fluid Dynamics*, 20(3):pp181–195, 2006.

[15] P.R.Spalart, W.H. Jou, M.R.Strelets, and S.R. Allmaras. Comments of the feasibility of les for wings, and on a hybrid rans/les approach. In *First AFOSR International Conference on DNS/LES*, Ruston, Louisiana, Aug.4-8,1997.

[16] P.R.Spalart and S.R.Allmaras. A one-equation turbulence model for aerodynamic flows. *Recherche Aerospatiale*, 1:pp5–21, 1994.

[17] P.Sagaut, S.Deck, and M.Terracol. *Multiscale and Multiresolution Approaches in Turbulence*. Imperial College Press, Cmabridge, UK, first edition, 2006.

[18] J. Smagorinsky. General circulation experiments with the primitive equations: 1. the basic equations. *Mon. Weather Review*, 91:pp99–164, 1963.

[19] B.P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4:pp419–420, 1962.

[20] D.C. Wilcox. *Turbulence Modeling for CFD*. D.C.W. Industries, California, second edition, 2002.