

**Multi-Fluid Navier-Stokes Solver**  
**TransAT**  
**User Manual**

ASCOMP GmbH

Technoparkstrasse 1  
CH-8005 Zurich  
Switzerland  
**transat@ascomp.ch**

May 25, 2009

# Contents

<b>1</b>	<b>TransAT Specification</b>	<b>5</b>
1.1	Physical Models . . . . .	5
1.2	Numerical Methods . . . . .	5
<b>2</b>	<b>TransAT Installation</b>	<b>7</b>
2.1	Using TransAT . . . . .	8
<b>3</b>	<b>Rapid User Guide</b>	<b>9</b>
3.1	Basic Steps . . . . .	9
3.2	Project name . . . . .	9
3.3	Grid generation . . . . .	9
3.4	Boundary conditions . . . . .	10
3.5	Input files . . . . .	10
3.6	Initial conditions . . . . .	10
3.7	Execution and Post processing . . . . .	11
3.8	Restarting a simulation . . . . .	11
3.8.1	Saving Restart Files . . . . .	11
3.8.2	Using Restart File . . . . .	11
3.8.3	Using Output File . . . . .	11
<b>4</b>	<b>User Guide</b>	<b>13</b>
4.1	Simulations using TransAT . . . . .	13
4.2	Creating a Mesh . . . . .	13
4.3	Setting input file - transat.inp . . . . .	15
4.4	Setting boundary conditions file - projectname.bc . . . . .	25
4.4.1	Inputs for Wall Boundaries . . . . .	28
4.4.2	Inputs for Inflow Boundaries . . . . .	28
4.4.3	Inputs for Outflow Boundaries . . . . .	29
4.5	Setting initial conditions . . . . .	30
4.6	Output files and postprocessing . . . . .	37
4.7	Immersed Surfaces Technique . . . . .	38
4.7.1	Inputs for IST . . . . .	38
4.7.2	Specification of the immersed surface . . . . .	38
4.7.3	Specification of material properties . . . . .	39
4.8	Two Phase Flow Modelling . . . . .	42

4.8.1	Inputs for Level Set Method . . . . .	42
4.8.2	Surface tension and Marangoni effects . . . . .	42
4.8.3	Contact angle modelling . . . . .	42
4.8.4	Thin film boundary conditions . . . . .	44
4.8.5	Phase change modelling . . . . .	45
4.9	Reynolds Averaged Turbulence Modelling . . . . .	46
4.9.1	Inputs for RANS . . . . .	46
4.10	Large Eddy Simulation . . . . .	48
4.10.1	Inputs for LES . . . . .	48
4.11	Anisotropic Heat Conduction . . . . .	51
4.11.1	Inputs for Anisotropic Heat Conduction . . . . .	51
4.11.2	Anisotropic Heat Conduction with Immersed Surfaces . . . . .	51
4.12	Point Heat Sources . . . . .	52
4.12.1	Inputs . . . . .	52
4.13	Unsteady Inflow Specification . . . . .	53
4.13.1	Using pre-generated dataset . . . . .	53
4.13.2	Internally generated fluctuations: usage . . . . .	53
4.13.3	Internally generated fluctuations: theory . . . . .	53
4.14	Periodic Boundary Conditions . . . . .	54
4.14.1	Inputs for Periodic BCs . . . . .	54
4.15	Oscillating Body Force . . . . .	55
4.15.1	Inputs for Oscillating Body Force . . . . .	55
4.16	Compact Filtering . . . . .	56
4.16.1	Inputs for Compact Filtering . . . . .	56
4.16.2	Compact Filtering Theory . . . . .	56
4.17	Average Statistics . . . . .	57
4.17.1	Inputs for Statistics . . . . .	57
4.18	Post Processing Utilities . . . . .	58
4.18.1	Non-dimensional numbers . . . . .	58
4.18.2	Time-signal probes . . . . .	58
4.18.3	Center of Mass of drop or bubble . . . . .	58
4.18.4	Total gas-liquid interfacial area . . . . .	59
4.18.5	Detection of drops and bubbles . . . . .	59
4.18.6	Total kinetic energy of phases . . . . .	59
4.18.7	Forces acting on an embedded object . . . . .	59
4.18.8	Maximum and minimum values of variables . . . . .	59
4.18.9	Electronic component cooling . . . . .	60
4.19	Pressure Solvers . . . . .	61
4.19.1	Inputs for Pressure Solvers . . . . .	61
4.20	Run-time Intervention . . . . .	62
4.20.1	Inputs for run-time intervention . . . . .	62
4.21	Shared Memory Parallelism using OpenMP . . . . .	65
4.21.1	How To . . . . .	65

4.21.2	Parallel Performance	65
<b>5</b>	<b>Examples</b>	<b>66</b>
5.1	Channel Flow	66
5.2	Backward Facing Step Flow	67
5.3	Bubbles Merging	68
5.4	Film Boiling	69
5.5	Flow over a Cylinder	69
<b>6</b>	<b>Particle Tracking Module - Steinli</b>	<b>71</b>
6.1	Basic features	71
6.2	Overall interface	71
6.3	Input file	71
6.4	Use with TransAT	73
6.5	Particle Temperature	77
6.5.1	Inputs	77
6.6	Particle Dispersion	79
6.6.1	Inputs	79
6.6.2	Langevin model	79
6.7	Point Particles Sources	80
6.7.1	Inputs	80
6.8	Mathematical formulation	81
6.8.1	Momentum equations	81
6.8.2	Temperature equation	81
<b>7</b>	<b>Numerical Method</b>	<b>82</b>
7.1	Governing Equations	82
7.2	Layout and Indexing of Control Volumes	84
7.3	Geometric Quantities	84
7.4	Discretization of Governing Equations	87
7.4.1	Flux balance equations	87
7.4.2	Approximation of convection terms	87
7.4.3	Approximation of diffusion terms	88
7.4.4	Approximation of source terms	89
7.4.5	Final form of discretised equations	89
7.5	Pressure-Velocity Coupling	90
7.6	Boundary Conditions	91
7.6.1	Inflow planes	91
7.6.2	Outflow planes	91
7.6.3	Symmetry planes	92
7.6.4	Rigid wall	92
7.7	Solution Procedure	92
7.7.1	Convergence criterion	92
7.7.2	Calculation sequence	93

<b>8 Custom Utilities</b>	<b>94</b>
8.1 Wind Safety of Tower Cranes . . . . .	94
8.1.1 Description of setupcrane.inp . . . . .	94
8.1.2 Output . . . . .	95
8.2 Tecplot Animation Macro Utility: TransATMovie . . . . .	95
<b>9 Best Practice Guidelines</b>	<b>96</b>
9.1 Non-dimensional Numbers . . . . .	96
9.2 Time Step Restrictions . . . . .	96

# Chapter 1

## TransAT Specification

TransAT (Transport Phenomena Analysis Tool) is a structured, curvilinear grids, finite-volume, Navier-Stokes solver for incompressible and weakly compressible multiphase flows.

### 1.1 Physical Models

TransAT has the following physics modelling capabilities:

- Mass, momentum, heat transport modelling in single and two-phase flow conditions.
- Passive concentration transport.
- Axisymmetric flows. (Swirl flow in upcoming release)
- Reynolds averaged turbulence modelling: two equation  $k - \epsilon$  models with specialized near-wall modelling.
- Unsteady turbulence modelling: Large eddy simulation, Direct numerical simulation.
- Non-Newtonian transport property variations.
- Buoyancy driven flows.
- Two-phase flows with surface tension, and dynamic contact angle modelling (density ratios upto  $\approx 10,000$ ).
- Liquid-vapour flows with phase change models.
- Dispersed two-phase flows with particles, drops, bubbles.
- Conjugate heat transfer with solid boundaries or obstacles using Embedded Interface method.

### 1.2 Numerical Methods

The following numerical methods are available in TransAT:

- Higher-order explicit time stepping: Runge-Kutta 2 – 4<sup>th</sup> order.

- Implicit time stepping: 1<sup>st</sup> order and 2<sup>nd</sup> order Crank–Nicholson.
- Various monotonic convection schemes: HLLP, Quick, Hybrid.
- Solvers: Stone’s Implicit Procedure (SIP), Geometric Multigrid (GMG) using SIP, Preconditioned (SIP/GMG) GMRES, Preconditioned Bi-Conjugate Gradient Squared.
- Lagrangian particle tracking.
- Interface tracking with Level Sets and Volume of Fluid (VOF).
- Level Set reinitialization using 3<sup>rd</sup>-order WENO scheme.
- Inflow, Outflow boundary conditions: velocity or pressure specified.

## Chapter 2

# TransAT Installation

The distribution of TransAT is a file called `transat_vn.n.tar.gz`, which is a precompiled executable. To install TransAT, the above file should be copied to the directory where the user wants to install the executable (**installdir** - could be the home directory of a user) and unzipped. To extract the archive (.tar file), the following command can be used,

```
> tar xvf transat_vn.n.tar
```

This will create a folder called TransAT with the following sub-folders,

- **bin** - with precompiled executable **transat** and a shell scripts **makeinitial** and **makepost** which create initial conditions, and postprocess output files, respectively. TransAT executables come in single precision **transatSP** and double precision **transatDP**.
- **input** - has a sample input files,
  1. `transat.inp` - Input file for problem specification
  2. `particle.inp` - Input file for dispersed phase
  3. `transat.rti` - Input file for run-time control
  4. `xxxxxxxxxx.bc` - Sample boundary conditions file
  5. `initialconditions.f90` - Fortran template to create initial conditions
  6. `user_postprocess.f90` - Fortran template to process output files
- **manual** - has a PDF file **TransatUserManual.pdf** - which is file you are reading.
- **examples** - three examples including setup files are available,
  1. `channel` - flow between infinite parallel plates
  2. `backwardstep` - flow over a backward-facing step
  3. `bubblemerge` - two bubble rising in liquid (two-phase)
- **lib** - contains static library archive **libtransat.a**, which is used to generate initial conditions, and for post processing.

## 2.1 Using TransAT

For easy access to the executable and scripts, users should add the folder **TransAT/bin** to their search path. For the **bash** shell, the following line can be added to the **.profile** in the home directory,

```
export PATH="installdir/TransAT/bin:$PATH"
```

The scripts `makeinitial` and `makepost` require the environment variable `TRANSATDIR` to be set to **installdir/TransAT**. For the **bash** shell, the following line should be added to the **.profile** in the home directory,

```
export TRANSATDIR=installdir/TransAT
```

# Chapter 3

## Rapid User Guide

In this chapter the basic steps required to run a simulation using TransAT will be very briefly described. The next chapter will give further detailed support on each step, including description of input parameters, etc.

### 3.1 Basic Steps

1. Create project folder, e.g. **myproject**. (3.2)
2. Grid generation: create a coordinates file called **myproject.grda** if ASCII or **myproject.grdb** if Binary. (3.3)
3. Boundary conditions: create a file called **myproject.bc** where boundary conditions for all surfaces are specified. (3.4)
4. Input file specification: copy sample input file **transat.inp** and set appropriate input parameters such as density, convection scheme, etc. (3.5)
5. Initial conditions: create an initial conditions file called **myproject.ini** if necessary (e.g. two-phase flow where the interface between the two fluids has to be specified). (3.6)
6. Execute **transat** in the project folder. (3.7)
7. Restarting a simulation. (3.8)
8. Post process results using software such as Tecplot or freely available Paraview ([www.paraview.org](http://www.paraview.org)).

### 3.2 Project name

Create a folder with the chosen project name (e.g. **steam\_injection**). TransAT will write all simulation output in a sub-folder called **RESULT**.

### 3.3 Grid generation

Any grid generation software can be used to generate a structured body-fitted grid. TransAT reads the grid in Plot3D or Grid3D format as either ASCII or binary.

If using the in-house orthogonal grid generator OrthoMesh, create an input file called **orthomesh.inp** by executing the script `/ORTHOMESH/UTILS/input_template`. Edit **orthomesh.inp** to specify geometry, grid sizes, filename, etc. Refer to OrthoMesh User Manual for details.

### 3.4 Boundary conditions

Copy sample boundary conditions file **xxxxxxx.bc** from **installdir/TransAT/input** into file **myproject.bc** in folder **myproject**. Set the boundary conditions. (4.4)

**Note:** the boundary conditions file can be directly created using OrthoMesh by turning ON the 'write\_transat\_bcs' option in **orthomesh.inp**. Refer to OrthoMesh User Manual for details.

### 3.5 Input files

Copy sample input file **transat.inp** from **installdir/TransAT/input** into **myproject** folder. Copy sample input file **particle.inp** if the problem includes Lagrangian particle tracking. The input is in the form of Fortran Namelists; all input quantities have a default value already set, therefore only those quantities that need to be specified could be set. List of all input variables is available in the User Manual. (4.3)

**Important:** In **transat.inp** the input **projectname** in NAMELIST PROJECT\_NAME should be set to **myproject**. This tag will be used for all input data files such as **myproject.bc**, **myproject.grdb**, **myproject.ini**, etc. Also, this tag will be used to write output files **RESULT/myproject.000100.dat**, and restart files **myproject.runb**.

### 3.6 Initial conditions

If the problem is fully specified using the boundary conditions then TransAT can be run without specifying initial conditions. However, for more complex situations, for example, inflow velocity has a different profile, or for two-phase flow, etc. initial conditions could be specified by the user.

To specify problem specific initial conditions, copy sample initial conditions Fortran template **initialconditions.f90** from **installdir/TransAT/input** to **myproject** folder. Edit this Fortran file to set the initial conditions for variables that are being solved for, e.g.  $u$ ,  $v$ ,  $w$ ,  $p$ ,  $T$ , level-set function  $\phi$ , etc.

Execute the script **makeinitial ;compiler; ;precision;** from **myproject** folder. This will create a project specific executable **transatinitial** in the project folder and execute it to generate the initial conditions file **myproject.ini**. At the same time, an output file in the format specified in **transat.inp** will be written in folder **RESULT**. This can be viewed using a visualisation software and verified to be valid before proceeding with the simulation. [More details \(4.5\)](#)

## 3.7 Execution and Post processing

Execute TransAT (`transatDP` - if `TransAT/bin` has been added to the `PATH`) from the project folder; results will be written to sub-folder `RESULT`. The results can be viewed using software such as Tecplot or Paraview ([www.paraview.org](http://www.paraview.org)).

## 3.8 Restarting a simulation

### 3.8.1 Saving Restart Files

There are two ways of saving restart files in TransAT, which are complementary to each other. The relevant input variables are set in `Namelist: CONTROL_PARAMETERS`.

1. **Overwrite method:** In this way only one file *projectname.runb* is maintained in the project folder. This file is over-written every `NSAVE` time-steps. Only the restart file at the end of the simulation is available for restart.
2. **Sequence method:** In this way a sequence of restart files are written in the project folder. The files are named as, *projectname.0003000.runb*. The frequency of saving restart files is specified using the input variable `RESTART_STORE`.

The best way is to use both of the methods in conjunction, particularly for long simulations. The sequence method is storage intensive since, typically, restart files are larger in size compared to the output files. Therefore, the variable `RESTART_STORE` should be set to larger values as compared to `NSAVE`.

**Tip:** TransAT has the feature to compress output/restart files using `gzip`. This feature is turned ON by setting the input `GzipOutput` in `Namelist: Visualization` as `TRUE`.

### 3.8.2 Using Restart File

To restart a TransAT simulation, the following steps should be carried out:

1. Copy file *myproject.runb* to *myproject.rstb* in the **myproject** folder.
2. Edit input file *transat.inp* and set the variable `LREAD = .true.`. Save *transat.inp*.
3. Execute **transat** in the **myproject** folder. TransAT reads the *myproject.rstb* file and starts the run from that point.

### 3.8.3 Using Output File

Simulations can also be restarted using output files (currently only available for Tecplot format). This option is, however, not guaranteed to give an exact restart since it depends on which variables the user has chosen to write in the output file. Also, this option works only with full three-dimensional output files (`L3DPLOT = .TRUE.`)

To restart a TransAT simulation using Tecplot output files, the following steps should be carried out:

1. Copy the output file of your choice from the **RESULT** folder to the **myproject** folder. If it is gzipped, run **gunzip projectname.000xxxx.dat.gz**.
2. Find out what is the time stamp (**CTIME**) and accumulated time step number for this file (**NRTSTP**). This can be done by looking at **RESULT/cfl.dat** or in the output file itself in the header section under the tag: **AUXDATA time =**.
3. Set the following inputs in **transat.inp**

```
&INPUT_OUTPUT_OPERATIONS
LREAD = .true.
LREAD_FILE = "projectname.000xxxx.dat" !default: "restart"
LR_CTIME   = <time stamp of output file>
LR_NRTSTP  = <time step of output file (total/accumulated)>
/
```

4. TransAT reads the data from the output file and continues the simulation.

# Chapter 4

## User Guide

### 4.1 Simulations using TransAT

The following steps are necessary to perform a simulation using TransAT,

- Create a new folder **myproject**.
- Grid generation: create a coordinates file called **myproject.grda** if Ascii or **myproject.grdb** if Binary.
- Boundary conditions: create a file called **myproject.bc** where boundary conditions for all surfaces are specified.
- Input file specification: copy sample input file **transat.inp** and set appropriate input parameters such as density, convection scheme, etc.
- Initial conditions: create an initial conditions file called **myproject.ini**.
- Execute **transat** in the project folder.
- Post process results using software such as Tecplot or Paraview.

### 4.2 Creating a Mesh

TransAT reads a structured single-block corner mesh in Plot3D or Grid3D format. The format in Fortran 90 syntax is as below:

```
!Ascii
write(unitno,*)ni, nj, nk
write(unitno,15)((x_corner(i,j,k),i=1,ni),j=1,nj),k=1,nk)
write(unitno,15)((y_corner(i,j,k),i=1,ni),j=1,nj),k=1,nk)
write(unitno,15)((z_corner(i,j,k),i=1,ni),j=1,nj),k=1,nk)
15 Format(10(E14.5))
```

```
!Binary
write(unitno)ni, nj, nk
write(unitno)((x_corner(i,j,k),i=1,ni),j=1,nj),k=1,nk)
```

```
write(unitno)((y_corner(i,j,k),i=1,ni),j=1,nj),k=1,nk)
write(unitno)((z_corner(i,j,k),i=1,ni),j=1,nj),k=1,nk)
```

where, **ni**, **nj**, **nk** are the number of corner nodes in the  $x$ ,  $y$ ,  $z$  directions, respectively. The mesh file should be saved in the project folder in Ascii or Fortran Unformatted format, with names **myproject.grda** or **myproject.grdb**, respectively.

To create a mesh for TransAT using OrthoMesh, refer to the OrthoMesh User Manual.

TransAT also contains the Immersed Surfaces Technology (IST), where the geometry is immersed in a Cartesian mesh. The geometry is read in STL format ([http://en.wikipedia.org/wiki/STL\\_\(file\\_format\)](http://en.wikipedia.org/wiki/STL_(file_format))) into TransATMesh. To create a mesh for TransAT using TransATMesh, refer to the TransATMesh User Manual.

### 4.3 Setting input file - transat.inp

In this section detailed description of all input variables is given. The file *transat.inp* consists of Fortran Namelist input, such that the order of the namelists is not important. Some important points are listed below:

- The projectname specification is necessary.
- The number of computational nodes in each direction is equal to the number of corners in the mesh  $(Nxc) + 1$ . This is because TransAT uses the finite volumes in the mesh  $(Nxc - 1)$  plus two cells on the boundary face giving  $Nx = Nxc - 1 + 2 = Nxc + 1$  in each direction. This information is relevant for setting up the boundary conditions file.
- **Axisymmetry option:** For the axisymmetry option the  $x$ -axis is always chosen to be the axis of symmetry, and the  $y$ -axis represents the radial direction.
- **Two-phase flow:** PHASE1 is always the lighter phase. For single phase simulations, the fluid properties should be set in Namelist PHASE1
- TransAT writes out a full input file in the RESULT folder at the beginning of each simulation with the name *transat.inp*. It also writes out a sample **run time intervention file**, *transat\_rti*.

Variable	Type	Options/description
projectname	string	Template for input and output files

Table 4.1: Namelist PROJECT\_NAME

Variable	Type	Options/description
xgravity	real	Gravity component in $x$ direction
ygravity	real	Gravity component in $y$ direction
zgravity	real	Gravity component in $z$ direction
hydrostatic	logical	Turn on the hydrostatic pressure term (default=.F.)
steady	logical	Flag for steady simulation
axisym	logical	Flag for axisymmetry

Table 4.2: Namelist FLOW\_CONDITIONS

<b>Variable</b>	<b>Type</b>	<b>Options/description</b>
radius	real	Setting bubble/drop radius
ptotal_inflow	real	Pressure inflow condition
add_gravity_head	real	Add hydrostatic pressure to initial pressure
density_ref_phase	logical	The phase for hydrostatic head
reinit_domain	logical	Reinitialize the initial Level-Set function

Table 4.3: Namelist INITIAL\_CONDITIONS

<b>Variable</b>	<b>Type</b>	<b>Options/description</b>
rhoG	real	Density of lighter phase
viscG	real	Viscosity
lambG	real	Thermal conductivity
cpG	real	Heat capacity
c_diffG	real	Scalar diffusivity
TexpG	real	Coefficient of thermal expansion

Table 4.4: Namelist PHASE1, always lighter phase

<b>Variable</b>	<b>Type</b>	<b>Options/description</b>
rhoL	real	Density of heavier phase
viscL	real	Viscosity
lambL	real	Thermal conductivity
cpL	real	Heat capacity
c_diffL	real	Scalar diffusivity
TexpL	real	Coefficient of thermal expansion

Table 4.5: Namelist PHASE2, always heavier phase

<b>Variable</b>	<b>Type</b>	<b>Options/description</b>
Solve_Pressure	logical	Pressure
Solve_UVelocity	logical	x velocity - u
Solve_VVelocity	logical	y velocity - v
Solve_WVelocity	logical	z velocity - w
Solve_Temperature	logical	Temperature
Solve_Concentration	logical	Scalar concentration
Solve_LevelSet	logical	Level-Set
Solve_VoF	logical	Volume of Fluid (VoF)
Solve_k_epsilon	logical	$k - \epsilon$ equations
particle_tracking	logical	Particle Tracking
phase_change	logical	Phase Change
LargeEddySim	logical	Large eddy simulation

Table 4.6: Namelist EQUATION\_TO\_BE\_SOLVED

<b>Variable</b>	<b>Type</b>	<b>Options/description</b>
ntimet	integer	Number of iterations to run
stime	real	Initial time (secs)
ctimet	real	Final time of simulation (secs)
ntimp1	integer	Frequency of output files
nsave	integer	Frequency of restart files
restart_store	integer	Frequency to store restart file (>> nsave)
dtime	real	Time step (initial guess if adaptive stepping)
epsexp	real	Residue convergence criterion (explicit, steady)
epspep	real	Pressure convergence criterion (explicit)
epsimp	real	Residue convergence criterion (implicit)
epsimp	real	Pressure convergence criterion (implicit)
maxit	integer	Explicit: Maximum number of pressure-velocity iterations Implicit: Maximum number of iterations
nswp	integer	Number of inner iterations for pressure solver
imon	integer	i-index for solution monitoring
jmon	integer	j-index for solution monitoring
kmon	integer	k-index for solution monitoring

Table 4.7: Namelist CONTROL\_PARAMETERS

Variable	Type	Options/description
ntsch	integer	choice of time stepping scheme ntsch = 1 and 2 (Euler 1 <sup>st</sup> and 2 <sup>nd</sup> order) ntsch = 3 Runge-Kutta 2 <sup>nd</sup> order ntsch = 4 Runge-Kutta 2 <sup>nd</sup> order ntsch = 5 Runge-Kutta 3 <sup>rd</sup> order (default) ntsch = 6 Runge-Kutta 4 <sup>th</sup> order
adapttime	logical	for adaptive time stepping
cflmax, cflmin	real	max and min of CFL criterion
difmax, difmin	real	max and min of viscous criterion
stnmax, stnmin	real	max and min of surface tension criterion
foumax, foumin	real	max and min of heat flux criterion
solve_implicit	logical	activate implicit solver
autorelaxation	logical	set relaxation factor automatically
Scarborough	real	higher value for more relaxation for use in conjunction with autorelaxation higher value provides more relaxation
relax_impl()	real	relaxation factor for different equations (1) Density advection (2-4) Velocity advection (5) Temperature advection (6) Concentration advection (7) Level-Set advection (8) Heat Capacity update (9) $k$ (10) $\epsilon$ (11) pressure correction

Table 4.8: Namelist TIME\_SCHEME

Variable	Type	Options/description
scheme_density	integer	Density advection scheme
scheme_uvelocity	integer	Velocity advection scheme
scheme_vvelocity	integer	Velocity advection scheme
scheme_wvelocity	integer	Velocity advection scheme
scheme_temperature	integer	Temperature advection scheme
scheme_concentration	integer	Concentration advection scheme
scheme_levelset	integer	Level-Set advection scheme
scheme_tke	integer	$k$ advection scheme
scheme_epsilon	integer	$\epsilon$ advection scheme
		scheme = 1 Hybrid scheme
		scheme = 2 Quick scheme
		scheme = 3 Central scheme
		scheme = 4 HLLP scheme
		scheme = 6 TVD scheme

Table 4.9: Namelist CONVECTION\_SCHEME

Variable	Type	Options/description
lread	logical	true if restarting run
lread_file	string	default "restart", options "3doutputfilename"
lr_ctime	real	time of output file when restarting with output file
lr_nrtstp	integer	time step of output file when restarting with output file
rascii	logical	true if reading ASCII restart file
wascii	logical	true if writing ASCII restart file
gascii	logical	true if read ASCII grid file
nsuffix	integer	size of iteration tag to output file names
RejectPressure	logical	Reject the pressure from restart file
RejectVelocity	logical	Reject the velocity from restart file
RejectTemperature	logical	Reject the temperature from restart file
RejectConcentration	logical	Reject the concentration from restart file
RejectTurbulence	logical	Reject the turbulence from restart file

Table 4.10: Namelist INPUT\_OUTPUT\_OPERATIONS

<b>Variable</b>	<b>Type</b>	<b>Options/description</b>
L2Dplot	logical	if writing 2D results file
write_kplane	logical	activate xy plane for 2D results
write_jplane	logical	activate xz plane for 2D results
write_iplane	logical	activate yz plane for 2D results
IPLANE	integer	'i' plane number for xy plane
JPLANE	integer	'j' plane number for xy plane
KPLANE	integer	'k' plane number for xy plane
L3Dplot	logical	if writing 3D results file
wrtpres	logical	include pressure in results output
wrtuvel	logical	include x velocity in results output
wrtvvel	logical	include y velocity in results output
wrtwvel	logical	include z velocity in results output
wrtlset	logical	include level-set in results output
wrtlvoF	logical	include VoF in results output
wrtdens	logical	include density in results output
wrtvisc	logical	include viscosity in results output
wrttemp	logical	include temperature in results output
wrtconc	logical	include concentration in results output
wrtmean	logical	include MEAN p, u, v, w, T in results output
wrtustr	logical	include RMS u', v', w' in results output
wrtcstr	logical	include Reynolds stress u'v', u'w', v'w' in results output
wrttke	logical	include TKE in results output
wrteps	logical	include dissipation rate in results output
wrtmut	logical	include eddy viscosity in results output
wrttkem	logical	include TKE mean in results output
wrtepsm	logical	include dissipation rate mean in results output
wrtmutm	logical	include eddy viscosity mean in results output
wrtlseembed	logical	include embedded level-set in results output
data_format	string	"tecplot", "plot3d", "paraview"

Table 4.11: Namelist VISUALIZATION

<b>Variable</b>	<b>Type</b>	<b>Options/description</b>
defaultsolver	string	"sip", "multigrid" (for implicit)
pressuresolver	string	"sip", "gmres", "multigrid"
preconditioner	string	"sip" (default), "multigrid" (for GMRES)
levelm	integer	Maximum multigrid levels
ldstrt	integer	GMRES solution space size
ldstrtmax	integer	GMRES solution space size (Max)

Table 4.12: Namelist ImplicitSolvers

<b>Variable</b>	<b>Type</b>	<b>Options/description</b>
LS_narrowband	logical	Use narrow band approach (faster)
Marangoni	logical	Activate Marangoni forces
TCrit	real	Critical temperature
TWeber	real	Temperature at which surface tension is known
wall_film_thickness	real	Film thickness BC (Liquid $\neq 0$ , Gas $\neq 0$ )
interface_width	real	width of interface spread
weber	real	surface tension
correct_mass	string	"local", "global", "localglobal", "none"
num_reinit_steps	integer	number of reinitialization steps
eq_contact_angle	real	equilibrium contact angle (for all walls)
activate_tripeline	logical	activate dynamic contact angle model
write_dyn_contact_angle	logical	output facility

Table 4.13: Namelist LEVEL\_SET\_INPUT

<b>Variable</b>	<b>Type</b>	<b>Options/description</b>
correlation	string	"none", "fixed", "hiemenz"
mdot	real	value of $\dot{m}$ if fixed
latent	real	latent heat used for Hiemenz model
tsat	real	saturation temperature
tsc	real	temperature of cool phase
tsh	real	temperature of hot phase
Lliqmdot	real	length scale (liquid) for computing Reynolds number
Uliqmdot	real	velocity scale (liquid) for computing Reynolds number
Lgasmdot	real	length scale (gas) for computing Reynolds number
Ugasmdot	real	velocity scale (gas) for computing Reynolds number

Table 4.14: Namelist PHASE\_CHANGE\_INPUT

Variable	Type	Options/description
unsteady_inflow	string	default: "none", options: "data", "generator"
InflowGfile	string	file name of inflow grid (for "data" option)
InflowUfile	string	file name of inflow U velocity (for "data" option)
InflowVfile	string	file name of inflow V velocity (for "data" option)
InflowWfile	string	file name of inflow W velocity (for "data" option)
AutoCorrLx	real	Autocorrelation length in X direction ("generator")
AutoCorrLy	real	Autocorrelation length in Y direction ("generator")
AutoCorrLz	real	Autocorrelation length in Z direction ("generator")
RSTarget11	real	Reynolds stress targeted (1,1) ("generator")
RSTarget23	real	Reynolds stress targeted (2,3) ("generator")
RSTarget33	real	Reynolds stress targeted (3,3) ("generator")
RSTarget21	real	Reynolds stress targeted (2,1) ("generator")
RSTarget31	real	Reynolds stress targeted (3,1) ("generator")
RSTarget32	real	Reynolds stress targeted (3,2) ("generator")

Table 4.15: Namelist INFLOW\_CONDITIONS

Variable	Type	Options/description
LowRemodel	logical	True for low Reynolds number model
Twolayermodel	logical	True for two-layer model
UransFW	real	Filter based unsteady RANS - filter width
wallroughness	real	default wall roughness
inflowwallroughness	real	default inflow wall roughness
turb_intensity	real	for inflow conditions
mol_visc_factor	real	ratio of turbulent to molecular viscosity for inflow conditions
itkemax	integer	Number of $k - \epsilon$ sub-iterations
sigma_k	real	Diffusivity modification for $k$
sigma_eps	real	Diffusivity modification for $\epsilon$
Cmu	real	Model constant for $\mu_t$
Ce1	real	Model constant $\epsilon$ equation
Ce2	real	Model constant $\epsilon$ equation
Ce3	real	Model constant $\epsilon$ equation

Table 4.16: Namelist K\_EPSILON\_INPUT

Variable	Type	Options/description
nexit	integer	0 (default), 1 (second-order extrapolation)

Table 4.17: Namelist BC\_INPUT

Variable	Type	Options/description
refpres	real	Reference pressure
reftemp	real	Reference temperature
refl	real	Reference length
refu	real	Reference velocity
refconc	real	Reference concentration

Table 4.18: Namelist REF\_STATE

Further Namelists are described in the physics specific sections.

Variable	Type	Options/description
activate_embedded_interface	logical	To activate embedded interface model
LSformat	string	"ascii", "binary" - format of myproject.ls file
rhoembed	real	Density of embedded solid material
cpembed	real	Heat capacity of embedded solid material
lambdaembed	real	Conductivity of embedded solid material
roughnessembed	real	Wall roughness for solid material
heat_source_embed	real	Volumetric heat source from solid material
film_thickness_embed	real	Film thickness BC for solid material

Table 4.19: Namelist EMBEDDED\_INTERFACE

<b>Variable</b>	<b>Type</b>	<b>Options/description</b>
activate_nondim_numbers	logical	To print non-dimensional numbers in file RESULT/ND_numbers.dat
activate_timesignals	logical	To print time signal of monitor point
kinetic_energy	logical	To print average TKE versus time
interfacial_area	logical	To print total interfacial area versus time

Table 4.20: Namelist POSTPROCESSING

<b>Variable</b>	<b>Type</b>	<b>Options/description</b>
LOAVTI	logical	Perform time averaging
NUAVTI	integer	Sampling interval in iterations
NSTAAV	integer	Iteration number to start time average
iaver	integer	1: Averaging in $i$ direction
javer	integer	1: Averaging in $j$ direction
kaver	integer	1: Averaging in $k$ direction

Table 4.21: Namelist AVERAGING

## 4.4 Setting boundary conditions file - projectname.bc

The boundary conditions file can be directly generated from OrthoMesh or TransATMesh for cartesian grids. For body-fitted grids they have to be set by hand. Below is an example of a boundary conditions file with comments.

```
&BOUNDARY_CONDITIONS
NREBDA = 6
!-----
! nbound = 1
!-----
cbnd(1,1) = "west"
cbnd(1,2) = "wall"
cbnd(1,3) = "nofor"
!
bndtt(1) = "dirichlet" !temperature BC type "neumann", "dirichlet"
bndtv(1) = 1.0          !value of BC: it is a heat-flux if "neumann"
                        !or a temperature if "dirichlet"
!
vbnd(1,1) = 0.0         !Wall velocity if not equal to zero (default)
vbnd(1,2) = 0.0
vbnd(1,3) = 0.0
bnd_inflow_type(1) = 0 !0 => value specified here
                        = 2 !2 => value specified in initial conditions
inflowprofile(1) = "constant", "laminar", "turbulent"
!
volumeflowrate(1) = 1.0
!
turbulence_intensity(1) = 0.1
eddyvisc_ratio(1) = 100.0
!
contact_angle(1) = 90.0
film_thickness(1) = 10.0e-3
!
nbnd(1,1) = 2
nbnd(1,2) = 2
nbnd(1,3) = 1
nbnd(1,4) = 80
nbnd(1,5) = 1
nbnd(1,6) = 3
!-----
! nbound = 2
!-----
cbnd(2,1) = "east"
```

```

cbnd(2,2) = "sym"
cbnd(2,3) = "nofor"
!
nbnd(2,1) = 19
nbnd(2,2) = 19
nbnd(2,3) = 1
nbnd(2,4) = 80
nbnd(2,5) = 1
nbnd(2,6) = 3
!-----
! nbound = 3
!-----
cbnd(3,1) = "south"
cbnd(3,2) = "inflow"
cbnd(3,3) = "nofor"
!
bnd_inflow_type(3) = 0          !2: from initial conditions
inflowprofile(3)   = "constant" !"laminar", "turbulent"
vbnd(3,2)          = 1.0       !j-velocity at inflow
!
nbnd(3,1) = 1
nbnd(3,2) = 20
nbnd(3,3) = 2
nbnd(3,4) = 2
nbnd(3,5) = 1
nbnd(3,6) = 3
!-----
! nbound = 4
!-----
cbnd(4,1) = "north"
cbnd(4,2) = "outflow" !"opflow" - pressure outflow
cbnd(4,3) = "nofor"
!
pexit(4) = 0.0 !Fixed relative pressure
!
nbnd(4,1) = 1
nbnd(4,2) = 20
nbnd(4,3) = 79
nbnd(4,4) = 79
nbnd(4,5) = 1
nbnd(4,6) = 3
!-----
! nbound = 5

```

```

!-----
cbnd(5,1) = "bottom"
cbnd(5,2) = "sym"
cbnd(5,3) = "nofor"
!
nbnd(5,1) = 1
nbnd(5,2) = 20
nbnd(5,3) = 1
nbnd(5,4) = 80
nbnd(5,5) = 2
nbnd(5,6) = 2
!-----
! nbound = 6
!-----
cbnd(6,1) = "top"
cbnd(6,2) = "sym"
cbnd(6,3) = "nofor"
!
nbnd(6,1) = 1
nbnd(6,2) = 20
nbnd(6,3) = 1
nbnd(6,4) = 80
nbnd(6,5) = 2
nbnd(6,6) = 2
/

```

### Description of variables

- **cbnd(n,1)** is the boundary orientation ("west", "east", "south", "north", "top", "bottom").
- **cbnd(n,2)** is the boundary type ("sym", "wall", "inflow", "outflow", "ipflow", "opflow"), where "ipflow" is a total pressure specified inflow, and "opflow" is a static pressure specified outflow.
- **cbnd(n,3)** is useful is the force acting on a wall boundary needs to be calculated ("nofor", "for").
- **nbnd** is an array that should be set to the grid indices for the boundary surface. Indices 1 and 2 should be set to the beginning and ending indices in the  $x$  direction, 3 and 4 to the  $y$  direction indices, and 5 and 6 to the  $z$  direction indices. Note: The indices start from 2 and end with  $n_x-1$ ,  $n_y-1$ , and  $n_z-1$  for the  $x$ ,  $y$ ,  $z$  directions, respectively. For a "west" or "east" boundary the two  $x$  indices should be the same and so on for the other boundaries.
- A nonzero wall velocity can be set using the array **vbnd**.

- Temperature boundary conditions at a no-slip "wall" can be set using `bndtt(n)` and `bndtv(n)`, where `bndtt` denotes the temperature boundary condition type ("dirichlet" or "neumann"), and `bndtv` denotes the value of the boundary condition (a temperature value if `bndtt(n) == "dirichlet"`, a heatflux value if `bndtt(n) == "neumann"`).
- Inflow boundary: A velocity profile can be set using `inflowprofile(n)` option: "constant", "laminar", "turbulent" options are available.
- Inflow boundary type: is set using `bnd_inflow_type(n)`. 0 means take values from boundary file, 2 means take values from initial conditions file.
- Inflow velocity: `vbnd(n,1-3)` - inflow velocity

#### 4.4.1 Inputs for Wall Boundaries

The following inputs are available for wall boundaries:

```

bndtt(n) = "dirichlet", "neumann" !Wall Temperature BC type
bndtv(n) = 300.0 (K) or 2.35 (W/m^2) !Either Dirichlet or Neumann
bndct(n) = "dirichlet", "neumann" !Wall Scalar BC type
bndcv(n) = 1.0 or 0.00 (/m^2) !Either Dirichlet or Neumann
contact_angle(n) = 90.0
film_thickness(n) = 10.0e-6
vbnd(n,1) = 0.0 !U: Special cases with wall velocity (eg. Driven Cavity)
vbnd(n,2) = 0.0 !V
vbnd(n,3) = 0.0 !W

```

#### 4.4.2 Inputs for Inflow Boundaries

The following inputs are available for inflow boundaries:

```

bnd_inflow_type(n) = 0 !0 => take from BC file
                    !1 => take from BC file (fixed pressure)
                    !2 => take from initial file

vbnd(n,1) = 0.0 !U: inflow velocity
vbnd(n,2) = 0.0 !V: inflow velocity
vbnd(n,3) = 0.0 !W: inflow velocity
ptot(n)   = 1.0 !Total pressure (relative)
volumeflowrate(n) = 20.0 !m^3/s (overrides vbnd)
!
inflowprofile(n) = "constant" !"laminar", "turbulent"
!
turbulent_intensity(n) = 0.1 !For k-e model
eddyvisc_ratio(n) = 20.0 !For k-e model
!
Bnd_AutoCorrLx = 1.0 !Unsteady inflow
Bnd_AutoCorrLy = 1.0

```

```
Bnd_AutoCorrLz = 1.0
Bnd_RSTarget11 = 0.02
Bnd_RSTarget22 = 0.02
Bnd_RSTarget33 = 0.02
Bnd_RSTarget21 = 0.00
Bnd_RSTarget31 = 0.00
Bnd_RSTarget32 = 0.00
!
inflow_particle_volfrac(n) = 1.0e-5      !Particle volume fraction
inflow_particle_temperature(n) = 1.0e-5 !Particle temperature
```

#### 4.4.3 Inputs for Outflow Boundaries

The following inputs are available for outflow boundaries:

```
pexit(n) = 0.0 !Static pressure (relative)
```

## 4.5 Setting initial conditions

To generate initial conditions, copy the file `installdir/TransAT/input/initialconditions.f90` in to the project folder and set the initial conditions for the variables to be solved. The user needs to have one of the following Fortran compilers available in the executable search path: viz. GFortran, G95 Intel Fortran.

- GFortran is the GNU Fortran compiler available as part of GCC (GNU C/C++ compiler). GFortran is not necessarily installed by default and would have to be installed using the package manager (Example: YaST in openSUSE Linux). Only GFortran 4.3 and later support `ISO_C_BINDING` used to achieve compiler independence.
- G95 is a free Fortran compiler that can be downloaded from [www.g95.org](http://www.g95.org) -> Download. Note that the following pre-compiler version should be downloaded: Linux x86\_64/EMT64 (32 bit D.I.) Default integer of 32 bits, compatible with older programs. Follow the installation instructions.
- Intel Fortran is freely available only for non-commercial personal use. The university license is available at a very affordable price. See: [www.intel.com](http://www.intel.com) -> Intel Fortran Compiler for Linux
- To check if the compilers are correctly added to your executable path, type the following command in your shell prompt.

```
> which ifort
ifort: Command not found.

> which g95
/home/myself/Software/g95-install/g95-integer32bit/bin/g95

> which gfortran
gfortran: Command not found.
```

The above output indicates that Intel Fortran and GFortran are not available (or not added to executable search path `$PATH`) and that G95 is available for use.

Execute the script `makeinitial (installdir/TransAT/bin/makeinitial ;compiler; ;precision; ;` e.g. `ifort`, `g95`, `gfortran`) in the project folder. This will create an executable **transatinitial** in the project folder. On execution `transatinitial` will create the initial conditions file named `projectname.ini` and an initial solution file in folder `RESULT`. The sample `initialconditions.f90` file is described below:

```
!-----!
! Template to create initial flow field !
!-----!
SUBROUTINE initialconditions
  Implicit none
```

```

include "precision.h"
include "get_set_interfaces.h"

Integer:: i, j, k, ii
!-----!
! TransAT: Do not edit above this line !
!-----!

Integer :: nijk, ni, nij
Real(iprec):: d1, d2, rad, factor
Real(iprec), Dimension(3):: c1, c2, pc

Real(iprec), Dimension(:), Pointer :: p, u, v, w
Real(iprec), Dimension(:), Pointer :: x, y, z, phi

Real(iprec), External :: sphere

nijk = get_integer('nijk')
ni    = get_integer('ni')
nij   = get_integer('nij')

call set_pointer(p, 'pressure')
call set_pointer(u, 'uvelocity')
call set_pointer(v, 'vvelocity')
call set_pointer(w, 'wvelocity')

call set_pointer(x, 'cellcenterX')
call set_pointer(y, 'cellcenterY')
call set_pointer(z, 'cellcenterZ')

call set_pointer(phi, 'levelset')

Do ii=1,nijk
  p(ii) = 0.0
  u(ii) = 0.00001
  v(ii) = 0.00001
  w(ii) = 0.00001
Enddo

factor = 0.001d0
c1(1) = factor*2.25d0; c1(2) = factor*1.5d0; c1(3) = factor*2.0d0 !bubble 1
c2(1) = factor*1.75d0; c2(2) = factor*3.8d0; c2(3) = factor*2.0d0 !bubble 2

```

```

rad = factor*1.0d0

Do ii = 1,nijk
  pc(1) = x(ii)
  pc(2) = y(ii)
  pc(3) = z(ii)
  d1 = sphere(pc,c1,rad,"gas")
  d2 = sphere(pc,c2,rad,"gas")
  phi(ii) = min(d1,d2)
Enddo
END SUBROUTINE initialconditions
!-----
! USAGE: Every variable is stored in a 1-Dimensional array.
!       The transformation from 3D to 1D is as follows
!       If i, j, k are indices for the x, y, and z directions and
!       ni,nj,nk are the maximum values, then:
!       ii = i + (j-1)*ni + (k-1)*nij, where nij = ni*nj, nijk = nij*nk
!-----
! Use routine 'set_pointer(your_pointer,keyword) to set a
! pointer to an array. The pointer has to be declared as
! Real(iprec), Dimension(:), Pointer :: your_pointer
!
! Pointers keywords examples:
! - cellcenterx
! - cellcentery
! - cellcenterz
! - cellcornerx
! - cellcornery
! - cellcornerz
! - uvelocity
! - vvelocity
! - wvelocity
! - pressure
! - temperature
! - concentration
! - levelset
! - vof
!
! There are three function to get a copy of global variables:
!
! get_integer(keyword)
!   -> keywords: - ni
!               - nj

```

```

!           - nk
!           - nij
!           - nijk
!           - scheme_density
!           - scheme_Uvelocity
!           - scheme_Vvelocity
!           - scheme_Wvelocity
!           - scheme_temperature
! get_real(keyword)
!     -> keywords: - dtime
!
! get_logical(keyword)
!     -> keywords: - solve_pressure
!                 - solve_Uvelocity
!                 - solve_Vvelocity
!                 - solve_Wvelocity
!                 - solve_temperature
!                 - solve_concentration
!                 - solve_levelset
!                 - solve_vof
!                 - activate_embedded_interface
!-----
!-----!
! To create a distance function for circles and spheres !
! NOTE: for circles one coordinate can be sent in as zero !
!-----!
FUNCTION sphere(pc,cc,radius,phase)
  include "precision.h"
  Real(iprec):: sphere
  Real(iprec), Dimension(3), Intent(in):: pc, cc
  Real(iprec), Intent(in):: radius
  Character(LEN=*), Intent(in):: phase

  Integer:: iliquid

  iliquid = -1
  If (phase == "liquid") iliquid = 1

  sphere = real(iliquid,iprec)*(radius - SQRT((pc(1)-cc(1))**2 &
                                             +(pc(2)-cc(2))**2 &
                                             +(pc(3)-cc(3))**2))
END FUNCTION sphere

```

The following points should be noted:

- For Level-Set, the lighter phase should have  $\phi < 0$  and the heavier phase should have  $\phi > 0$ . In the above example, two bubbles of radius (rad) located at  $c1(:)$  and  $c2(:)$  are initialized. Inside the bubbles  $\phi < 0$ .
- Velocities are initialized to small values.
- For incompressible flow pressure is usually set to zero.
- To specify an initial velocity to the heavier phase, the smoothed  $H$  function should be first calculated. The smooth  $H$  function is equal to one in the heavier phase and zero in the lighter phase.

```

delta = get_real("delta")

uheavy = 1.2
ulight = 0.2
Do ii = 1,nijk
  Hheavy = get_heavyside(phi(ii),delta,"tanh")
  u(ii) = uheavy*Hheavy + (1.0 - Hheavy)*ulight
Enddo

```

In this example, the heavy phase is set to a  $x$ -velocity of 1.2 and the lighter phase is set to a  $x$ -velocity of 0.2.

The following is the list of arrays, integers, reals, and logicals that are accessible to define initial conditions:

Array Pointers	Valid Input Strings
Fluid density:	"rhofluid","den","density"
Pressure:	"isp","pres","pressure"
U-velocity:	"isu","uvel","uvelocity"
V-velocity:	"isv","vvel","vvelocity"
W-velocity:	"isw","wvel","wvelocity"
Temperature:	"istemp","temp","temperature"
Concentration:	"isconc","conc","concentration"
Cell center X:	"isx","ccenx","cellcenterx"
Cell center Y:	"isy","cceny","cellcentery"
Cell center Z:	"isz","ccenz","cellcenterz"
Cell corner X:	"iscox","ccorx","cellcornerx"
Cell corner Y:	"iscoy","ccory","cellcornery"
Cell corner Z:	"iscoz","ccorz","cellcornerz"
Interpolation coefficient I:	"isfx","inti","interpolationi"
Interpolation coefficient J:	"isfy","intj","interpolationj"
Interpolation coefficient K:	"isfz","intk","interpolationk"

Level Set:	"isph","levelset"
Volume of Fluid:	"iss","vof"
Embedded Level Set:	"lsembed"
Closest Object number:	"closestobject" (integer array)
Embedded Heavyside:	"hembed"
Embedded Dirac Delta:	"deltaembed"
Block-out function:	"kblk"

---

Integer Variables	Valid Input Strings
Number of Grid points in I:	"ni"
Number of Grid points in J:	"nj"
Number of Grid points in K:	"nk"
Number of Grid points I*J:	"nij"
Number of Grid points I*J*K:	"nijk"
Advection scheme density:	"scheme_density"
Advection scheme u-velocity:	"scheme_uvelocity"
Advection scheme v-velocity:	"scheme_vvelocity"
Advection scheme w-velocity:	"scheme_wvelocity"
Advection scheme temperature:	"scheme_temperature"
Advection scheme concentration:	"scheme_concentration"
Advection scheme levelset:	"scheme_levelset"
Advection scheme turbulent kinetic energy:	"scheme_tke"
Advection scheme epsilon:	"scheme_epsilon"

---

Real Variables	Valid Input Strings
Saturation temperature:	"tsat"
Time step:	"dtime"
Gravity in X direction:	"xgravity"
Gravity in Y direction:	"ygravity"
Gravity in Z direction:	"zgravity"
Density of Phase 1:	"rhog"
Viscosity of Phase 1:	"viscg"
Heatcapacity of Phase 1:	"cpg"
Thermal conductivity of Phase 1:	"lambg"
Density of Phase 2:	"rhol"
Viscosity of Phase 2:	"viscl"
Heatcapacity of Phase 2:	"cpl"
Thermal conductivity of Phase 2:	"lamb1"
Surface tension coefficient:	"weber"

Interface width:	"delta"
Real constant zero:	"zero"
Real constant one:	"one"
Real constant Pi:	"pi"
Real constant small:	"small"
Real constant infinity:	"infty"

---

Logical Variables	Valid Input Strings
If solving pressure:	"solve_pressure"
If solving u-velocity:	"solve_uvelocity"
If solving v-velocity:	"solve_vvelocity"
If solving w-velocity:	"solve_wvelocity"
If solving temperature:	"solve_temperature"
If solving concentration:	"solve_concentration"
If solving level set:	"solve_levelset"
If solving VoF:	"solve_vof"
If solving embedded interfaces:	"activate_embedded_interface"

## 4.6 Output files and postprocessing

Output file writing can be controlled by setting input values in Namelist Visualization in **transat.inp**. Full three-dimensional solution can be written out by setting `L3DPLOT = .true.` Output can be written in three formats, "tecplot", "paraview" and "plot3d". The Plot3D output files can be visualised using either Paraview or Tecplot.

Two-dimensional planes of data can also be written by setting `L2DPLOT = .true.` For this option, one plane perpendicular to each direction has to be chosen. If `write_kplane = .true.` then `kplane` should be set, etc. The output files have the following naming convention. *xy*-plane files are prefixed with *xy*- followed by the project name, iteration number and suffix based on output format. A two-dimensional Tecplot output file for *xy*-plane at iteration number 2100 would be named: `xy-myproject.0002100.dat`, and the corresponding Paraview file will be named as: `xy-myproject.0002100.vtk`

## 4.7 Immersed Surfaces Technique

This section describes the Immersed Surfaces Technique (IST) of TransAT.

### 4.7.1 Inputs for IST

The Immersed Surfaces option is activated by setting the logical variable `activate_embedded_interface` to be `.true.` in `transat.inp` as described below.

```
&EMBEDDED_INTERFACE
activate_embedded_interface = .true./.false.
LSformat      = "ascii"/"binary" !Format of surface definition file
                                   !projectname.ls

rhoembed      = 1.0      !Density of immersed solid
cpembed       = 1.0      !Heat Capacity of immersed solid
lambdaembed   = 1.0      !Thermal Conductivity of immersed solid

MaterialEmbed = "iron"      !Material of immersed solid
               "copper"     !Will overwrite the user defined properties
               "aluminium" !above
               "concrete"

heat_source_embed = 0.0      !W
roughnessembed    = 1.0e-6    !metres - roughness of surface
film_thickness_embed = 1.0e-6 !metres - film boundary condition
/
```

### 4.7.2 Specification of the immersed surface

TransAT expects the immersed surface definition to be available in a file called `projectname.ls`. The immersed surface is represented as a level set function such that the level set values are positive inside the surface and negative outside (positive in the solid and negative in the fluid domains, respectively). The format of the file is as follows:

```
Do 10 k = 1,nk
Do 10 j = 1,nj
Do 10 i = 1,ni
  ii = i + (j-1)*ni + (k-1)*nij
  !- If LSformat is set to "binary" (Fortran unformatted)
  Read(LSunit, IOSTAT=ios) LSvar(ii), ClosestObject(ii)

  !- If LSformat is set to "ascii" (Formatted)
  Read(LSunit,*,IOSTAT=ios) LSvar(ii), ClosestObject(ii)
10 Continue
```

A level set value is required for each grid point, along with the number of the closest object in case the immersed surface is made up of many parts (each part being a separate closed surface). This is typically created using TransATMesh - the preprocessor to create meshes for the immersed surface technique.

### 4.7.3 Specification of material properties

For multiple objects, each object can have different properties (volumetric heat source, heat capacity, thermal conductivity, density, surface roughness). These can be specified in a file called *properties.dat* in the following format,

```
3
OBJECT 1
DENSITY 3000.0
HEATCAPACITY 1000.0
CONDUCTIVITY 1.0
HEATSOURCE 10.0
ROUGHNESS 0.000001
FILMTHICKNESS 0.00001
END OBJECT

OBJECT DEFAULT
DENSITY 3000.0
HEATCAPACITY 1000.0
CONDUCTIVITY 1.0
HEATSOURCE 10.0
ROUGHNESS 0.000001
FILMTHICKNESS 0.00001
END OBJECT
```

where the first line contains the number of objects. The Object DEFAULT will be used to set properties for objects not explicitly specified. For Anisotropic heat conduction (4.11) some extra inputs are available to specify the anisotropy in heat conductivity for each object.

```
OBJECT 1
DENSITY 3000.0
HEATCAPACITY 1000.0
HEATSOURCE 10.0
ROUGHNESS 0.000001
FILMTHICKNESS 0.00001
CONDUCTIVITY 1.025
LAMBDA RATIO B 2.0
LAMBDA RATIO C 1.0
EULER ANGLE PHI 0.0
EULER ANGLE THETA 0.0
```

```
EULERANGLEPSI 0.0
END OBJECT
```

More details about the anisotropic heat conduction model is available in the corresponding section.

The *properties.dat* file is also created using the preprocessor TransATMesh. If a *properties.dat* file is found, then the properties specified in *transat.inp* file will be overwritten.

### Manually creating *projectname.ls*

For simple geometries the *projectname.ls* file can be created manually in the *initialconditions.f90* file in a way similar to creating the initial conditions for two-phase flow using Level Set or VoF methods. If the flag **activate\_embedded\_interface** has been set to TRUE, the memory allocated for the immersed surface representation can be used in the *initialconditions.f90* routine by declaring an array pointer and using the interface routine (**set\_pointer()**) to correctly access this memory. Below is an example of how the *projectname.ls* can be created.

```
!-----!
!- Declaration section -!
!-----!
!-----!
!- Declare an array pointer to array containing the Immersed Surface -!
!-----!
Integer:: nijk
Real(iprec), Dimension(:), Pointer:: x, y, z, ISarray

!-----!
!- Execution section -!
!-----!
nijk = get_integer("nijk")
call set_pointer(x      ,"cellcenterX")
call set_pointer(y      ,"cellcenterY")
call set_pointer(z      ,"cellcenterZ")
call set_pointer(ISarray,"LEmbed")

!-----!
!- somewhere in the routine -!
!-----!
Open(21,file="projectname.ls",form="unformatted")
ISarray(:) = 0.0_iprec
Do ii = 1,nijk
  phi1 = SQRT((x(ii)-xc)**2 + (z(ii)-zc)**2) - rad
  phi2 = yc - y(ii)

  ISarray(ii) = phi1
```

```
if (y(ii) < yc) then
  if (phi1 > zero) then
    ISarray(ii) = MIN(phi1,phi2)
  endif
else
  ISarray(ii) = phi2
endif
write(21)ISarray(ii), 1    !writing unformatted output with 1 object
Enddo
Close(21)
```

## 4.8 Two Phase Flow Modelling

This section describes the Two-phase flow interface tracking capabilities of TransAT.

### 4.8.1 Inputs for Level Set Method

The main interface tracking method in TransAT is the Level Set method. The Level Set method is activated by setting the input **Solve\_LevelSet** in **Namelist: Equation\_To\_Be\_Solved**. Further inputs are specified in **Namelist: Level\_Set\_Input** as described.

```
Weber          = 0.072      !N/m: Surface tension coefficient
Eq_Contact_Angle = 90.0      !Equilibrium Contact Angle
Wall_Film_Thickness = 1.0e-6 !Film boundary condition
Activate_Triple_Line = T/F   !Dynamic contact angle model
!-
!- Numerical constants
!-
Interface_Width  = 1.0      !Smooth interface thickness
LS_Narrowband    = T/F     !Narrow band method
Reinit_Method    = "weno"   !Reinitialization method
Num_Reinit_Steps = 15      !Number of pseudo time steps
Correct_mass     = "local"  !Local mass conservation
                   !Options: "local", "global", "localglobal"
```

### 4.8.2 Surface tension and Marangoni effects

The surface tension coefficient is specified through the variable **weber** in **Namelist: Level\_Set\_Input**. For the case where the surface tension is a function of temperature, additional variables should be specified:

```
&Level_Set_Input
Marangoni = .TRUE.
TWeber = 300 !Temperature at which surface tension is specified
TCrit  = 430 !Temperature at which surface tension is zero
/
```

The following variation for the surface tension coefficient is in-built in TransAT:

$$\sigma(T) = \sigma_0 \frac{(T_{crit} - T)}{(T_{crit} - T_0)} \quad (4.1)$$

where,  $T_0$  represents the input variable **Tweber**.

### 4.8.3 Contact angle modelling

TransAT has both static and dynamic contact angle modelling capabilities. Contact angle is defined as shown in Fig 4.1, where the contact angle is the angle made between the interface

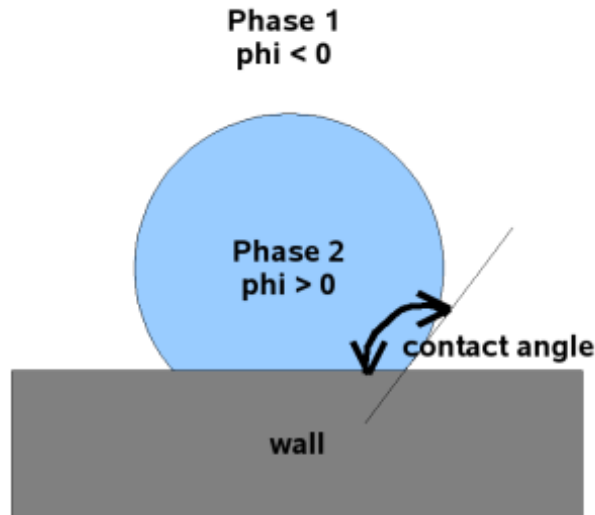


Figure 4.1: Definition of contact angle in TransAT

and the wall passing through the heavier phase. Note: in **transat.inp**, the lighter phase is represented by **Namelist: Phase1** and the heavier phase by **Namelist: Phase2**. The contact angle can be specified for all wall boundary conditions using the input variable **eq\_contact\_angle** in **Namelist: Level\_Set\_Input**. To specify different values for each wall boundary, the contact angle should be specified in the boundary conditions file **projectname.bc**. For example:

```
!-----
! nbound = 4
!-----
cbnd(4,1) = "north"
cbnd(4,2) = "wall"
cbnd(4,3) = "nofor"
!
contact_angle(4) = 130.0
!
nbnd(4,1) = 2
nbnd(4,2) = 89
nbnd(4,3) = 59
nbnd(4,4) = 59
nbnd(4,5) = 2
nbnd(4,6) = 2
```

The maximum contact angle that can be specified is  $170^\circ$  and the minimum is  $10^\circ$ . This limit is imposed due to the way the contact angle boundary condition is implemented. For situations very close to fully wetting or fully non-wetting, the *thin film boundary condition* as described in the following section can be used.

The dynamic contact angle model can be activated using the input variable **activate\_triple\_line** in **Namelist: Level\_Set\_Input** in **transat.inp**.

#### 4.8.4 Thin film boundary conditions

For flows with very low Capillary numbers, there exists a thin water film between the gas phase and the wall. This film can typically not be resolved by the near-wall grid. In this case, TransAT has the possibility that the user specified a minimum film thickness for a wall boundary (and/or for an embedded interface). This allows for a smooth motion of bubbles through a pipe (say).

The film thickness can be specified in **transat.inp** as a common value for all wall boundaries, or in the boundary conditions file **projectname.bc**, for each wall boundary.

```
!-----!  
!- For All Grid Walls -!  
!-----!  
&LEVEL_SET_INPUT  
...  
wall_film_thickness = 1.0e-7 !Unit: m  
                        !default: zero  
/  
  
!-----!  
!- For All Embedded Surfaces -!  
!-----!  
&EMBEDDED_INTERFACE  
...  
film_thickness_embed = 1.0e-6 !Unit: m  
                        !default: zero  
/  
  
!-----!  
!- For Specific Wall Boundaries -!  
!- projectname.bc file          -!  
!-----!  
!-----  
! nbound = 4  
!-----  
cbnd(4,1) = "north"  
cbnd(4,2) = "wall"  
cbnd(4,3) = "nofor"  
!  
film_thickness(4) = -6.0e-4  
contact_angle(4)  = 170.0 !contact angle is ignored  
                        !in this case  
  
nbnd(4,1) = 2  
nbnd(4,2) = 89  
nbnd(4,3) = 59
```

```

nbnd(4,4) = 59
nbnd(4,5) = 2
nbnd(4,6) = 2

```

Note: for walls where the film thickness is not specified, the default value specified in **transat.inp** will be used. If the value is 0.0, then the contact angle boundary condition is used (default: 90). The film boundary condition overrides the contact angle specification.

#### 4.8.5 Phase change modelling

This section describes the Phase change modelling capabilities of TransAT. Phase change modelling is turned on in TransAT by setting the input variable `Phase_Change` to be `.true.` in **Namelist: Equation\_To\_Be\_Solved**. Other input should be specified in **Namelist: Phase\_Change\_Input** described below.

```

&Phase_Change_Input
Correlation = "none","fixed"
mdot       = 0.1   !kg/(m^2 s)
mdotlocation = "everywhere","tripleline"
Latent     = 1000.0 !Latent Heat (J/Kg)
Tsat      = 373.0  !Saturation Temperature (K)
/

```

- If the input variable `Correlation` is set to "none", then  $\dot{m}$  is calculated from the energy/temperature equation (`Solve_Temperature = .true.`).
- If `Correlation` is set to "fixed", the value of  $\dot{m}$  should be specified in the input file via the variable `mdot` in SI units  $\text{kg}/\text{m}^2\text{s}$ .
- For the "fixed" option, a positive value for  $\dot{m}$  denotes evaporation and a negative value denotes condensation.
- The temperature difference driving phase change is either an initial or a boundary condition.
- The saturation temperature is locally adjusted to take into account the effect of curvature.

$$T_{sat}(\mathbf{x}) = T_{sat} \left( 1 + \frac{F(\rho)\kappa(\mathbf{x})\sigma(\mathbf{x})}{2L} \right) \quad (4.2)$$

where,  $F(\rho) = \rho_G^{-1} + \rho_L^{-1}$ ,  $\kappa$  is the local curvature,  $\sigma$  is the local surface tension coefficient, and  $L$  is the latent heat of phase change.

## 4.9 Reynolds Averaged Turbulence Modelling

This section describes the Reynolds Averaged Turbulence Modelling (RANS) capabilities of TransAT.

### 4.9.1 Inputs for RANS

The Reynolds Averaged Navier Stokes (RANS) turbulence modelling feature is turned on by setting the flag **Activate\_k\_epsilon** = **.TRUE.** in the **Namelist Equation\_To\_Be\_Solved** in **transat.inp**. Other options are set in the **Namelist K\_Epsilon\_Input**. **Note** that the  $k - \epsilon$  turbulence model can be used only in conjunction with the Implicit time stepping method.

```
&K_EPSILON_INPUT
!-
!- RANS Models
!-
LowRemodel   = .true./.false.   !Low-Reynolds number model
LReChoice    = 1/2/3            !1: Launder Sharma
                                           !2: Abe Kondoh Nagano
                                           !3: Lam Bremhorst

Twolayermodel = .true./.false. !Two-Layer Model
TLChoice      = 1/2            !1: Default 2-layer model
                                           !2: Rodi/Mansour/Michelassi

OneEqnmodel  = .true./.false. !One-Equation model

KatoLaunder  = .true./.false. !Kato-Launder correction for Stagnation

URansFW      = 1.0e15 (large) !Filter-width for filter based
                                           !Unsteady RANS computations

!-
!- Model Constants (Standard k-e model)
!-
sigma_k      = 1.0
sigma_eps    = 1.32
Cmu          = 0.09
Ce1          = 1.44
Ce2          = 1.92
Ce3          = 2.9556
WallRoughness = 0.1 !m - roughness of walls

!-
```

```
!- Inflow specification
!-
mol_visc_factor      = 100.0 !Ratio of eddy to laminar viscosity
turb_intensity       = 0.1   !10% turbulence intensity
InflowWallRoughness = 0.1   !m - roughness of incoming boundary layer

!-
!- Subiteration of k-e equations
!-
itkemax = 2 !(default)
/
```

## 4.10 Large Eddy Simulation

This section describes the Large Eddy Simulation (LES) capabilities of TransAT.

### 4.10.1 Inputs for LES

The Large eddy simulation feature is turned on by setting the flag **LargeEddySim** = .TRUE. in the **Namelist: Equation\_To\_Be\_Solved** in transat.inp. Other options are set in the **Namelist: Sgs\_Model**. The flag *Perturbation* should be set to .TRUE. for introducing initial perturbations in the flow (Gaussian random fluctuations).

&SGS\_MODEL

IMODEL: 0 - No SGS model.

1 - Smagorinsky model

2 - Dynamic model

!- -!

!- For Smagorinsky Model -!

!- -!

CS1 : Smagorinsky model constant

CAPPA : von Karman constant

IVANDR: 0 - No van Driest near-wall damping (default)

1 - van Driest damping is used

IMOSMA: 1 - Schmitt & Su near-wall correction

2 - Deardorff/Werner near-wall correction

0 - No modification to standard Smagorinsky (default)

IWALE : 1 - WALE model (Ducros and Nicoud) (default)

C2INH : Schmitt & Su model constant for near-wall damping

!- -!

!- For Dynamic Model -!

!- -!

ITYPF : Type of filter

1 - take averages in 3 directions (volume: x-y-z)

2 - take averages in 2 directions (plane : x-z )

3 - take averages in 1 direction (line : x )

ITYPA : Type of averaging of L and M tensors

0 - no averaging

1 - in x, y, or z directions

2 - along the x-z plane

ITYPC : Ad-hoc constraint on SGS viscosity  
1 - total viscosity is non-negative  
2 - SGS viscosity is non-negative

EPSDYN: Relaxation factor for the case ITYP = 0  
Default value is small 0.01 because dynamic model can  
be very noisy without averaging

```
!-                -!  
!- For Initial Conditions -!  
!-                -!
```

Perturbation = .true.

FGAUSS: Gaussian random perturbations on mean  
Default value = 1.0

/

The following Namelists are also needed to specify aspects such as periodicity, spatial/time averaging options for statistics, etc.

Time and spatial averaging for statistics:

&AVERAGING

```
LOAVTI = .true. or .false. !True: Perform time averaging  
NUAVTI = 1 (integer)       !sampling interval in iterations  
NSTAAV = 1000 (integer)    !Iteration number to start time average  
!  
iaver = 0/1                !0: no averaging in <i> direction  
javer = 0/1                !0: no averaging in <j> direction  
kaver = 0/1                !0: no averaging in <k> direction  
/
```

Periodicity specification:

&PERIODIC\_CONDITIONS

```
icycl = 0/1                !1: periodic in <i> direction  
jcycl = 0/1                !1: periodic in <j> direction  
kcycl = 0/1                !1: periodic in <k> direction  
/
```

Wall shear calculation:

&TAUW\_TIME\_AV

```
LOTAUW = .true. or .false. !True: Use new wall shear for length scale  
NUTAUW = 1 (integer)       !Schuman type wall treatment interval  
/
```

Wall shear treatment:

```
&BC_INPUT
```

```
nwals = 0 : for no-slip boundary condition (default)  
      = 1 : for Schumann type boundary condition  
      = 2 : for Werner/Wengle type boundary condition
```

```
/
```

## 4.11 Anisotropic Heat Conduction

This section describes the way to specify anisotropic heat conduction in TransAT.

### 4.11.1 Inputs for Anisotropic Heat Conduction

Anisotropic heat conductivity can be specified only for the material denoted by **Phase1** under the assumption that Phase1 is a solid and that the heat conduction equation is being solved. The following inputs in **Namelist: Phase1** can be used to specify the anisotropy in thermal conductivity.

```
&PHASE1
  RLambGB = 1.0          !Ratio of conductivity in second direction
  RLambGC = 1.0          !Ratio of conductivity in third direction
  EuleranglePhi = 0.0    !Euler angles of principal material frame
  EulerangleTheta = 0.0
  EuleranglePsi = 0.0
/
```

The conductivity in one of the principal directions (direction A) is specified in input variable **LambG**. The other two conductivities are represented as ratios, **RLambGB** and **RLambGC**, in the B and C directions, respectively. The Euler angles are used to specify the three rotations to transform the principal axes to the computational axes. The angles  $\phi$ ,  $\theta$ , and  $\psi$  represent three consecutive rotations about the  $x$ ,  $y$ , and  $z$  axes in that order.

For solving convective heat transfer problems, the fluid phases represented by Namelist inputs **Phase1** and **Phase2** should be chosen to be isotropic and the immersed objects can be defined as solids with anisotropic heat conduction as described below.

### 4.11.2 Anisotropic Heat Conduction with Immersed Surfaces

Anisotropic conductivities can also be specified for each immersed object separately in TransAT using the *properties.dat* file. The format of the *properties.dat* file are described in Section (4.7.3). This allows the modelling of conjugate heat transfer problems such that each immersed object also includes anisotropic heat conductivity.

## 4.12 Point Heat Sources

This section describes the capabilities of TransAT in defining point heat sources.

### 4.12.1 Inputs

This feature is activated by setting `activate_heat_sources` to be TRUE in **Namelist: Flow\_Conditions**

```
&Flow_Conditions
Activate_Heat_Sources = .TRUE.
/
```

The point heat sources are defined in a file `heat_sources.dat` in the project folder. It has the following format:

```
2
SOURCE 1
X 0.5
Y 0.5
Z 0.5
HEAT 10.0
STARTTIME 0.0
ENDTIME 1.0
END SOURCE
SOURCE 2
X 1.5
Y 1.5
Z 0.5
HEAT 10.0
STARTTIME 1.0
ENDTIME 2.0
END SOURCE
```

where, the first lines represents the number of point heat sources being defined, X, Y, Z are the coordinates of the source, HEAT is the heat source in Watts, STARTTIME and ENDTIME are the duration of the source in seconds. The heat source is applied in the cell in which the source is located. This means that the temperature of the point source can reach large values under grid refinement.

## 4.13 Unsteady Inflow Specification

This section describes the possibility of specifying unsteady inflow conditions in general, and particularly for Large Eddy Simulations in TransAT.

### 4.13.1 Using pre-generated dataset

This method requires four data files containing a planar grid file, and three files with velocity data over a time interval. TransAT reads data from this file, interpolates on to the inflow plane both in space and time. The file names are specified using the following inputs.

```
&INFLOW_CONDITIONS
```

```
Unsteady_Inflow = "data" !Options: "none", "generator"  
InflowGFile = "/home/../../inflowgrid.dat" !Full path name of grid  
InflowUFile = "/home/../../inflowUvel.dat" !Full path name of Uvel  
InflowVFile = "/home/../../inflowVvel.dat" !Full path name of Vvel  
InflowWFile = "/home/../../inflowWvel.dat" !Full path name of Wvel  
/
```

Please contact [transat@ascomp.ch](mailto:transat@ascomp.ch) for details on the data format for the inflow data files.

### 4.13.2 Internally generated fluctuations: usage

Will be available soon (contact: [transat@ascomp.ch](mailto:transat@ascomp.ch)).

### 4.13.3 Internally generated fluctuations: theory

Will be available soon (contact: [transat@ascomp.ch](mailto:transat@ascomp.ch)). The implementation is based on the reference below.

1. M. Klein, A. Sadiki and J. Janicka, A digital filter based generation of inflow data for spatially developing direct numerical or large eddy simulations, *J. Computational Physics* **186** (2003) 652–665.

## 4.14 Periodic Boundary Conditions

This section describes the possibility of specifying periodic boundary conditions in TransAT.

### 4.14.1 Inputs for Periodic BCs

Periodic boundary conditions can be specified through the **Namelist: Periodic\_Conditions**. The inputs are as follows:

```
&Periodic_Conditions
ICYCL = 0 !Periodicity in x-direction
JCYCL = 0 !Periodicity in y-direction
KCYCL = 0 !Periodicity in z-direction
/
```

Along with specifying the above inputs the boundary condition file, **projectname.bc**, should also be altered accordingly. For example:

```
cbnd(3,1) = "south"
cbnd(3,2) = "cyclic"
cbnd(3,3) = "nofor"
!
...
!
!-----
! nbound = 4
!-----
cbnd(4,1) = "north"
cbnd(4,2) = "cyclic"
!
...
!
/
```

If **JCYCL = 1**, then the South and East boundary types should be specified as **Cyclic**.

Limited support is available for specifying mass flow rate based pressure source term, in the case where the main streamwise direction is also periodic. For more details please contact: [transat@ascomp.ch](mailto:transat@ascomp.ch)

## 4.15 Oscillating Body Force

This section describes the way to specify the application of an oscillating body force in TransAT.

### 4.15.1 Inputs for Oscillating Body Force

An oscillating body force can be specified using the **Namelist: Wave\_Implementation** in transat.inp. The force is automatically applied if the input **VibFreq** is greater than zero.

```
&WAVE_IMPLEMENTATION
VIBFREQ = 0.0          !Default
VIBAMPL(1) = 0.0      !Amplitude in the x directions
VIBAMPL(2) = 0.0      !Amplitude in the y directions
VIBAMPL(3) = 0.0      !Amplitude in the z directions
VIBSTART = 1.0E+020    !Start time
VIBEND = 2.0E+020     !Snd time
VIBPHASE = 0.0        !Phase angle
VIBXBEG = -1.0E+035   !Apply only in select region
VIBXEND = 1.0E+035    !xbeg-xend
VIBYBEG = -1.0E+035   !ybeg-yend
VIBYEND = 1.0E+035    !zbeg-zend
VIBZBEG = -1.0E+035
VIBZEND = 1.0E+035
/
```

## 4.16 Compact Filtering

This section describes the capabilities of TransAT in filtering the velocity using very sharp low-pass filters to eliminate unwanted noise at the grid wave-number.

### 4.16.1 Inputs for Compact Filtering

The following inputs are available in **Namelist: Compact\_Filter** to control compact low-pass filtering.

```
&Compact_filter
Compact_Lowpass = .FALSE. !default
Compact_Factor  = 0.95    !default
/
```

The input **Compact\_Factor** can be set between [0 : 1], where 0 implies an explicit filter with a 27-point stencil, and 1 implies a filter that passes almost the whole velocity field. The filtering is applied to each velocity component separately. The resulting filtered velocity field is not exactly divergence free.

### 4.16.2 Compact Filtering Theory

## 4.17 Average Statistics

This section describes the possibility of time and space averaging for unsteady simulations, particularly for Large Eddy Simulations (LES) and Very Large Eddy Simulations (V-LES) using TransAT.

### 4.17.1 Inputs for Statistics

TransAT can generate average fields for unsteady simulations since it may be impossible to store the unsteady data in the frequency need to get a good averaged field.

Averaging can be controlled using the following inputs.

```
&AVERAGING
LOAVTI = .true. !Time averaging is performed
NUAVTI = 1      !Sampling interval in time steps
NSTAAV = 1000  !Time step to start time averaging
!
IAVER = 0/1 !Average in <i> direction
JAYER = 0/1 !Average in <j> direction
KAVER = 0/1 !Average in <k> direction
/
```

If space or time average is activated, TransAT automatically allocates memory for storing the averaged quantities. To output the averaged field in an output file, the following inputs should be set to .TRUE.

```
&VISUALIZATION
wrtmean = .TRUE. !writes mean p, u, v, w, T
wrtnstr = .TRUE. !writes normal Reynolds stresses u'u', v'v', w'w'
wrtcstr = .TRUE. !writes cross Reynolds stresses u'v', v'w', u'w'
wrttpkem = .TRUE. !writes mean turbulent kinetic energy
wrtepsm = .TRUE. !writes mean dissipation rate
wrtmutm = .TRUE. !writes mean eddy viscosity ratio
/
```

For extension of this feature, please contact us at: [transat@ascomp.ch](mailto:transat@ascomp.ch)

## 4.18 Post Processing Utilities

This section describes the various post processing utilities available in TransAT. As usual all input files related to the post-processing utilities are to be created in the project folder, and all results are by default written to the RESULT folder. The input variables are to be defined in **Namelist: Postprocessing**.

### 4.18.1 Non-dimensional numbers

Calculates a variety of non-dimensional numbers and writes them to file *ND\_numbers.dat*. Very useful utility. The length scale is taken from input variable **ReFL** in **Namelist: Ref\_State**. The reference velocity can also be specified, and if not, it is calculated as the maximum velocity in the simulation. This feature is turned ON by default.

Checking the *ND\_numbers.dat* file at the beginning of every simulation and specifying a reasonable value of the reference length and velocity is part of the best practise guidelines for using TransAT.

### 4.18.2 Time-signal probes

This utility saves time signal of pressure and velocities at a given set of points in the domain (the points do not have to match with a grid point - the values are interpolated). The points are given in a file named *timesignalxyz.dat* in the project folder. Each lines should have 4 columns: viz. probe number (integer), x-coordinate, y-coordinate, z-coordinate. Multiple sets of probes can be defined by using the tag **DATASET** before the beginning of each set. For example,

```
DATASET
1 0.1 0.5 0.7
2 1.0 2.5 3.7
DATASET
1 0.95 0.3 0.89
```

By default all the time signals are written to a single file, however, if the input **singlefile** is set to FALSE, a different file is created for each probe in each set. The filename has the template *timesignal.setx.00000p*. For large number of time-signal probes, a single file is compact, whereas, for a small number of sample points, a different file for each point allows immediate plotting and post-processing.

This feature is turned on by setting the input **activate\_timesignals** to be TRUE. A Further input is **TSstarttime**, which denotes the time at which saving time-signals should start. Signal is stored at every time step by default.

### 4.18.3 Center of Mass of drop or bubble

This utility works primarily for a single drop or bubble where the center of mass of either the gas or the liquid phase is calculated at every time step and save in a file (*center\_of\_mass.dat*). This option is turned on by setting input **bubble\_rise** to be TRUE. Input **phase\_choice** can be used to choose between “liquid” and “gas” phases.

This can be useful for problems such as bubble rise. In the case of multiple droplets or bubbles, the combined center of mass gets calculated.

#### 4.18.4 Total gas-liquid interfacial area

This utility calculates the total interfacial area in the simulation domain at every time step and saves in a file (*interfacialarea.dat*). This option is turned on by setting input **interfacial\_area** to be TRUE.

This can be useful for problems where there is significant change in flow topology (break-up and coalescence). The file reports, iteration number, time, interfacial area ( $\text{m}^2$ ), and interfacial energy =  $A\sigma$  (Nm), where  $\sigma$  is the surface tension coefficient.

#### 4.18.5 Detection of drops and bubbles

This utility detects and isolates separate liquid or gas blobs (droplets or bubbles) in the domain and write an output file with each dispersed object given a separate object number (*entitiesL.0001500.dat*). The utility automatically chooses the phase which is lower in volume fraction over the whole domain as the dispersed phase. This can then be further processed to extract dispersed phase statistics.

Simple quantities such as volume, surface area, sphericity etc. for each object is written in another file (*droplet\_stats.0001500* or *bubble\_stats.0001500*).

This utility can be turned on by setting the input **detect\_drops\_bubbles** to be TRUE.

#### 4.18.6 Total kinetic energy of phases

This utility calculates the total kinetic energy in the simulation domain for each phase (for two-phase flow), at every time step, and saves in a file (*KineticEnergy.dat*). This option is turned on by setting input **kinetic\_energy** to be TRUE.

The file has 8 columns: viz. iteration number, time, liquid-phase kinetic energy, gas-phase kinetic energy, total kinetic energy, liquid-phase kinetic energy intensity, gas-phase kinetic energy intensity, total kinetic energy intensity.

#### 4.18.7 Forces acting on an embedded object

This utility calculates and saves a time series of the pressure and viscous forces acting on each immersed object in a separate file. The files are named as *EIforces\_Objn.dat* where  $n$  denotes the object number. This feature is ON by default.

#### 4.18.8 Maximum and minimum values of variables

Writes the minimum and maximum of  $u, v, w, T, \phi, \rho$  in a file named *min\_max.dat*. Useful for debugging.

This utility can be turned on by setting the input **max\_values** to be TRUE.

#### 4.18.9 Electronic component cooling

This utility is applicable for electronic component cooling problems. For each immersed object in the domain quantities such as maximum and minimum temperature, average temperature, surface temperature, heat transfer coefficient etc. are calculated at the end of the simulation and listed in a file named *Ecooling.dat*.

This utility can be turned on by setting the input **activate\_ecooling** to be TRUE.

## 4.19 Pressure Solvers

This section describes the various pressure solver options available in TransAT.

### 4.19.1 Inputs for Pressure Solvers

The basic pressure solver available in TransAT is the SIP (Stone's Implicit procedure) solver. The geometric multigrid solver (GMG-SIP) uses the SIP solver at every multigrid level. This option can be turned on by setting the input **PressureSolver** to **"multigrid"**. TransAT's most efficient solver is the GMRES method, preconditioned by either **"sip"** or **"multigrid"**. The preconditioner can be set by the input variable **Preconditioner**, the default is **"sip"**.

```
&IMPLICITSOLVERS
SimpleC = .TRUE. !default
PressureSolver = "gmres" !default, options: "sip", "multigrid"
Preconditioner = "sip" !default, options: "multigrid"
DefaultSolver = "sip" !default, options: "gmres", "multigrid"
IterSIP = 3 !default
LDSTRT = 30 !default, GMRES restart size
LDSTRTMAX = 60 !default, GMRES restart size maximum
LEVELM = 5 !multigrid: Current number of levels
LEVMAX = 5 !multigrid: Maximum levels possible (=5)
/
```

- **Ldstrt:** is an input parameter to the GMRES solver which decides the size of the vector space of solutions that it can store. The memory required for GMRES increases linearly with LDSTRT. TransAT automatically tries to increase this variable if GMRES is unable to converge the solution in the maximum number of iterations (LDSTRTMAX+1).
- **Ldstrtmax:** is an input parameter to the GMRES solver which decides the maximum value of the vector space allowed. Increasing this value allows for better convergence for problems with large grid sizes (especially along one direction).

## 4.20 Run-time Intervention

The run-time behaviour of TransAT can be changed by creating run-time intervention file *transat.rti* in the project folder.

### 4.20.1 Inputs for run-time intervention

The file *transat.rti* is made up of the first line which indicates the action to be taken on reading the file and the **Namelist Run\_Time\_Control**. The following strings are accepted in the first line: **output**, **reread\_input**, **stop**, **noaction**.

- **output**: writes out an output file immediately after the file is read.
- **reread\_input**: reads the new input given in the **Namelist Run\_Time\_Control** in the file.
- **stop**: stops the simulation and writes a restart file.
- **noaction**: does not do anything. This option can be used while editing the Namelist in *transat.rti*. After the appropriate inputs are specified, **noaction** can be changed to **reread\_input**.

The **Namelist Run\_Time\_Control** consists of the following input variables:

```
&RUN_TIME_CONTROL
!-----
! Adaptive time stepping
! Sample values
!-----
adapttime = .true. or .false.
cflmax = 0.3
cflmin = 0.2
difmin = 0.3
difmax = 0.2
stnmax = 0.3
stnmin = 0.2
foumax = 0.3
foumin = 0.2

!-----
! Output files
! Sample values
!-----
l3dplot      = .true. or .false.
data_format = "tecplot" or "plot3d"
l2dplot      = .true. or .false.
write_iplane = .true. or .false.
```

```

write_jplane = .true. or .false.
write_kplane = .true. or .false.
iplane = 2
jplane = 2
kplane = 2

!-----
! Level Set related
! Sample values
!-----
reinit_method   = "sussman" or "none"
correct_mass    = "local" or "global"
num_reinit_steps = 5
interface_width = 1
stwidth        = 2

!-----
! Time steps/Restart
! Sample values
!-----
ntimet         = 1000
ctimet         = 1.0
dtime          = 1.0e-4
ntimp1         = 100
maxit          = 5
restart_store  = 1000

!-----
! Residuals
!-----
epsexp = 1.0e-4
epspep = 1.0e-6
epsimp = 1.0e-3
epspip = 1.0e-1
/

```

Example of *transat.rti* file:

```

reread_input
&run_time_control
ntimet = 20000
/

```

The number of iterations to be run can be changed without stopping the code using the above *transat.rti* file. Another option would be to stop the simulation using a *transat.rti* file with the

word **stop** in the first line and then change *transat.inp* as needed and restart the simulation (3.8).

Note: A file named *transat\_rti* file is written by TransAT in the RESULT folder. This file can be copied to the Project folder, renamed as *transat.rti* and edited as required.

## 4.21 Shared Memory Parallelism using OpenMP

OpenMP fine-grained parallelism has been implemented in TransAT to take advantage of multi-core machines. The parallelisation has been implemented at the loop level and the efficiency obtained depends on the problem size and the number of threads used.

### 4.21.1 How To

To activate OpenMP parallelism, the environment variable `OMP_NUM_THREADS` should be set to a number equal to the threads required. The following commands should be used based on the shell being used,

```
bash: export OMP_NUM_THREADS=2
csh : setenv OMP_NUM_THREADS 2
```

OpenMP parallelism is efficient only if a certain number of grid points exist for each core/node. TransAT automatically sets the number of threads to a lower value if there are not enough grid points ( $\approx 100,000$  grid points are assumed to be needed for each core). The upper bound on the number of threads remains the environment variable specification `OMP_NUM_THREADS`. To disable the automatic adjustment of the number of threads, use the Namelist Execution in *transat.inp*.

```
&Execution
AutoNumThreads = .false. (Default=.true.)
/
```

### 4.21.2 Parallel Performance

Figure 4.2 shows the parallel speedup and efficiency obtained for TransAT for a computation with  $\approx 500,000$  nodes.

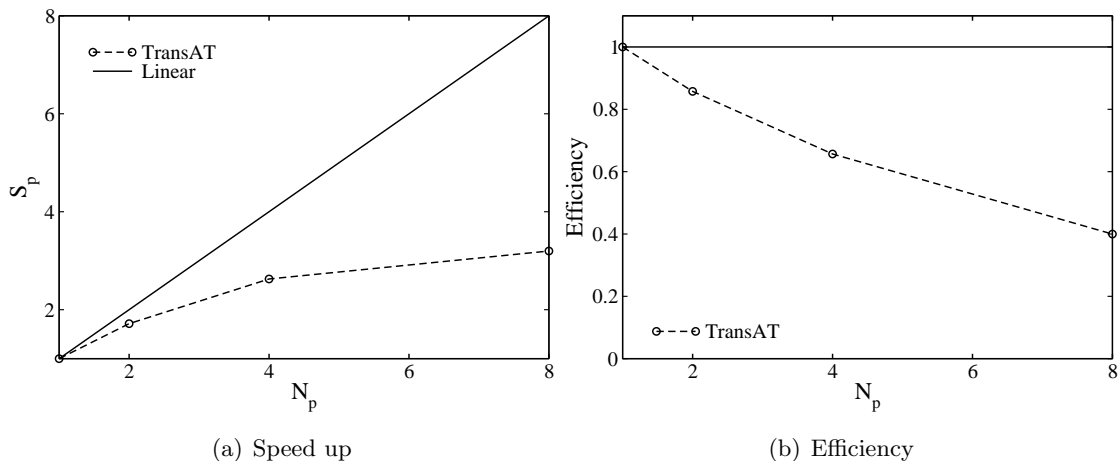


Figure 4.2: OpenMP Parallel: TransAT performance

# Chapter 5

## Examples

Four examples are included with the TransAT distribution, namely:

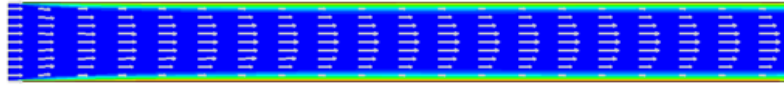
1. Laminar two-dimensional channel flow.
2. Laminar flow over a backward facing step.
3. Three-dimensional bubble merge problem.
4. Film boiling using a superheated wall.
5. Flow over a Cylinder with  $Re = 100$

### 5.1 Channel Flow

- Domain:  $10\text{ cm} \times 1\text{ cm}$  resolved by a mesh  $41 \times 21 \times 3$ .
- Left inflow: velocity  $1\text{ m/s}$ , constant velocity inflow, temperature  $T_{in} = 300\text{ K}$ .
- Right outflow (velocity extrapolation)
- South and north: no-slip walls, temperature  $T_w = 320\text{ K}$ .
- Top and bottom: symmetry boundaries.
- Fluid:  $\rho = 10\text{ kg/m}^3$ ,  $\mu = 6 \times 10^{-4}\text{ kg/ms}$ .
- Reynolds number = 167
- Note: for two-dimension problems the mesh size in the third direction should be three (one cell and two boundary nodes) and the "top" and "bottom" boundaries must be set to "symmetry".

The setup files for this problem are available in **examples/channel**. The files should be copied to another folder to be used to gain familiarity with using TransAT. As explained previously in this manual, the following steps should be performed to obtain a successful simulation:

1. Create your own channel flow folder **mychannel** and create a sub-folder **RESULT**.
2. Copy the contents of **examples/channel** to **mychannel**.



Channel flow: temperature contours

Figure 5.1: Temperature contours: channel flow

3. Look through **orthomesh.inp** and execute **orthomesh** in mychannel folder.
4. Look through **initialconditions.f90** and execute script **makeinitial**. Execute **transa-tinitial** which creates files channel.ini and RESULT/xy-channel.0000000.dat
5. Look through the boundary condition file channel.bc
6. Look through transat.inp
7. Execute **transat** in mychannel folder.
8. Visualize output files in RESULT folder.

Figure 5.1 shows the temperature contours obtained after a few time steps. The velocity profiles develops towards a fully-developed parabolic profile.

## 5.2 Backward Facing Step Flow

- Domain: 50 mm × 10 mm × 10 mm resolved by a mesh 98 × 50 × 10.
- The step is defined as a solid obstacle  $0 \geq x \geq 10$  mm and  $0 \geq y \geq 5$  mm.
- Left inflow: velocity 1 m/s, constant velocity inflow.
- Right outflow (velocity extrapolation)
- South, north and obstacle faces: no-slip walls.
- Top and bottom: symmetry boundaries.
- Fluid:  $\rho = 10$  kg/m<sup>3</sup>,  $\mu = 6 \times 10^{-4}$  kg/ms.
- Inflow Reynolds number = 83.33

Figure 5.2 shows the pressure contours and velocity vectors after a few time steps.

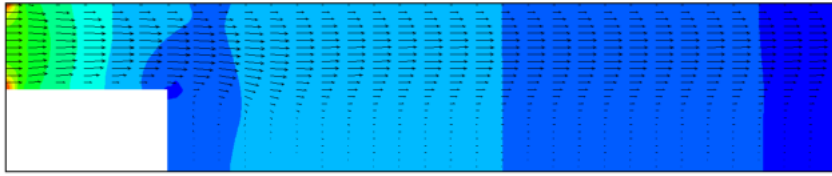


Figure 5.2: Pressure contours: backward facing flow

### 5.3 Bubbles Merging

- Domain:  $4 \text{ mm} \times 10 \text{ mm} \times 4 \text{ mm}$  resolved by a mesh  $40 \times 100 \times 40$ .
- Two bubbles with radii of 1 mm are initially located at center coordinates,  $c_1 = (2.25, 1.5, 2.0) \text{ mm}$  and  $c_2 = (1.75, 3.8, 2.0) \text{ mm}$ .
- Gravity vector:  $g_i = (0, 2.5, 0) \text{ m/s}^2$ .
- Bubbles:  $\rho = 1 \text{ kg/m}^3$ ,  $\mu = 18.16 \times 10^{-6} \text{ kg/ms}$ .
- Liquid:  $\rho = 1000 \text{ kg/m}^3$ ,  $\mu = 1 \times 10^{-3} \text{ kg/ms}$ .
- Surface tension  $\sigma = 0.002$ .
- All boundaries are "wall".

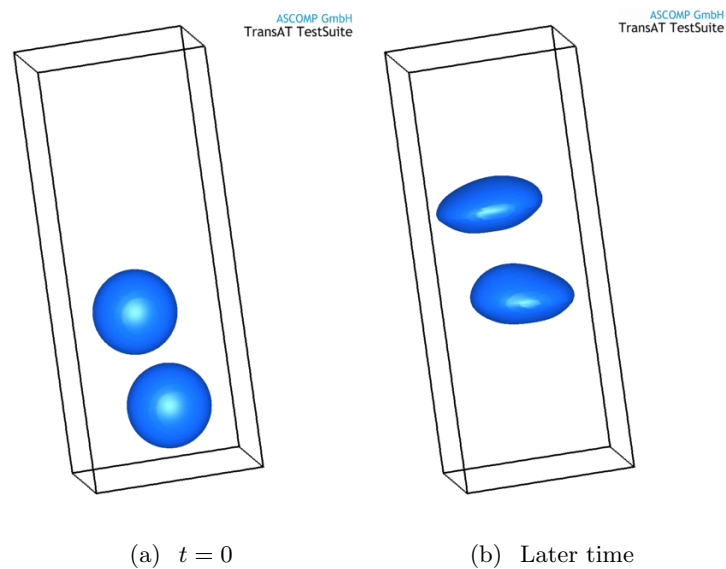


Figure 5.3: Bubble surfaces: bubble merge problem

## 5.4 Film Boiling

- Domain: 39.5 mm × 79 mm resolved by a mesh 70 × 70 × 3. The top is an outflow whereas the bottom is a wall with a defined temperature.
- Liquid:  $\rho = 200.0 \text{ kg/m}^3$ ,  $\mu = 0.1 \text{ kg/ms}$ ,  $c_{pL} = 400.0 \text{ J/kgK}$ ,  $\lambda = 40.0 \text{ W/mK}$ .
- Gas:  $\rho = 5.0 \text{ kg/m}^3$ ,  $\mu = 0.005 \text{ kg/ms}$ ,  $c_{pL} = 200.0 \text{ J/kgK}$ ,  $\lambda = 1.0 \text{ W/mK}$ .
- Surface tension:  $\sigma = 0.1$ .
- In the initial conditions, the heated surface of 5K is covered by a thin layer of vapor that separates the fluid with a temperature of 0K and a latent heat of 10.000J/kg from the wall completely. A Rayleigh - Instability of the liquid - vapor interface is triggered by gravity

$$\lambda = 2 \cdot \pi \cdot \sqrt{3 \cdot \sigma / g (\rho_v - \rho_l)}.$$

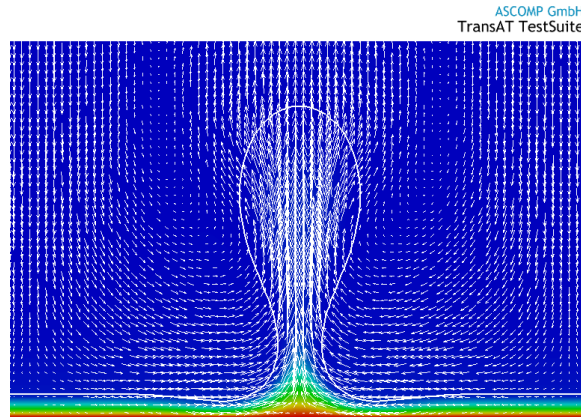


Figure 5.4: Film Boiling: Interface and Velocity Vectors, Colored by Temperature

## 5.5 Flow over a Cylinder

- Domain: 24 m × 12m. The first 9m in x - direction contain the cylinder as an embedded solid and they are resolved with a 100 × 99 × 2 mesh. The rear part is resolved with a coarser mesh containing 40 × 99 × 2 cells.
- The cylinder with a radius of 0.5m is located at (4.5, 6.0, 0).
- Liquid:  $\rho = 1.0 \text{ kg/m}^3$ ,  $\mu = 1.0E - 04 \text{ kg/ms}$ ,  $c_{pL} = 1005.0 \text{ J/kgK}$ ,  $\lambda = 0.025 \text{ W/mK}$ .
- A monitor point is set at (6.4, 6.1, 0.05) using the file "timesignalxyz.dat" and adding "activate\_timesignals = .true." in section "&Postprocessing" of "transat.inp". The file "timesignal.set1.0000001" contains the number of the point, the time, pressure and the velocities in x - , y - , and z - direction.

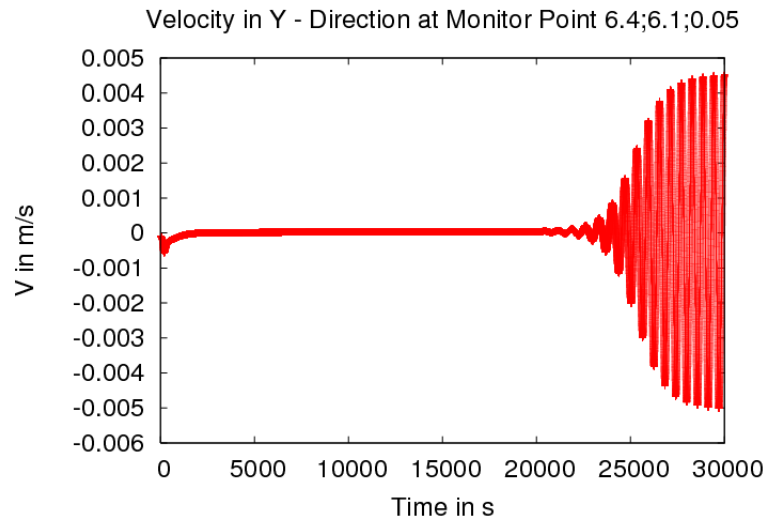


Figure 5.5: Monitor Point

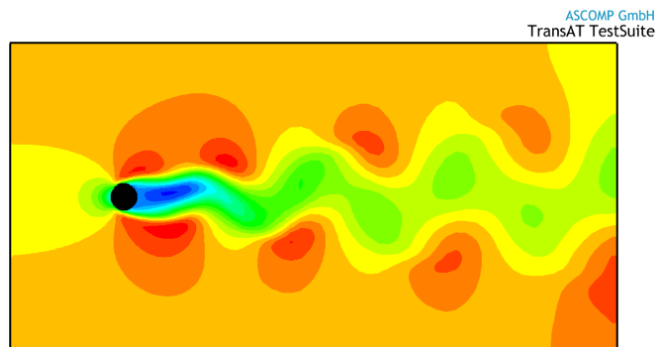


Figure 5.6: Cylinder Flow: Velocities

## Chapter 6

# Particle Tracking Module - Steinli

### 6.1 Basic features

The Lagrangian particle tracking program, **Steinli**, is an independent passive module. It has been coupled to TransAT. Using an input file, the user can specify the particle properties (density, diameter), the number of particles and their initial positions and velocities. The particle positions and velocities are updated at each time step in conjunction with the carrier flow field which evolves in time. The equation used to move the particles is given in Section 6.8. **Steinli** returns the fluid-particle coupling force to TransAT when the two-way coupled problem is to be solved. The module can only deal with curvilinear structured grid arrangements. It can also handle grids that change in time.

### 6.2 Overall interface

**Steinli** has been designed such that it can be coupled with any Eulerian flow solver or host program with minimum intrusion. Since all the particle information is stored inside the module very few new variables will have to be declared in the host program. The main additions to the host program would include modification of the data structures (arrays for example) to conform to the requirements of the interface described below. The module also has a separate input file *particle.inp*. Apart from the input file, the host program can interact with **Steinli** only through specific interface routines. The host program is needed to USE the module *lagrangian*.

### 6.3 Input file

The input file *particle.inp* uses NAMELIST formatting for input specification. If input values are not specified for some variables, default values are used. For most input parameters, the default values are just arbitrarily fixed and it is sensible to specify most if not all of the input parameters. The order of the NAMELISTS in the input file is not important. The input file parameters are tabulated. Some of the parameters that need further clarifications are elaborated here.

- The parameter  $M$  decides how many particles are computationally tracked as compared to the real number of particles dictated by the volume fraction. The physical correctness

of using  $M$  greater than 1 can be established by using it as a parameter. This factor is important only for two-way coupled cases. It is an integer  $\geq 1$ .

- The parameter *distfunc* decides how the number of particles in each size group is computed. The option “*const-phi*” divides the total volume fraction equally for each group resulting in less particles in the large size groups. “*const-np*” results in equal number of particles in each group and the “*log-normal*” distribution option uses the standard log-normal distribution (?).

$$n(d_p) = \frac{1}{\sqrt{2\pi}} \frac{1}{a_0 d_p} \exp \left[ - \left( \frac{\log d_p^+ - \log D_{nm}^+}{\sigma_0} \right)^2 \right] \quad (6.1)$$

where  $d_p^+ = d_p / (d_p^{max} - d_p)$ , and  $D_{nm}^+ = D_{nm} / (d_p^{max} - D_{nm})$  are the normalised particle diameters,  $d_p^{max}$  is the upper limit of particle diameter,  $D_{nm}$  the mean,  $\sigma_0$  the standard deviation, and  $a_0$  a factor used to normalize the distribution.

- The parameter *taufactor* is used to return, to the host program, the maximum allowable time-step from the particle side. The smallest particle response time ( $\tau_{p,min}$ ) is multiplied by *taufactor* to arrive at the maximal time-step. *taufactor* can be as high as 0.9 for 4<sup>th</sup> order Runge-Kutta time integration and should be maintained below 0.4 for the Adams-Bashforth and 2<sup>nd</sup> order Runge-Kutta methods.
- The forces that are to be included are to be explicitly chosen by the user based on the relevant physics of the problem at hand. None of the forces are hardwired. For example, in the case of bubbles it is left to the user to turn on the added-mass force so that life is not miserable.
- The parameter *Repmax* denotes the maximum particle Reynolds number that is allowed by the user. If at any time a particle acquires a higher value the program is stopped. This is based on the limitations imposed by the equations governing the particle motion.
- The parameter *outfilesize* fixes the maximum size of an output file. The output file names are generated using the parameter *datatmplt* and appending the beginning and ending iteration numbers to it. *outfilesize* together with the total number of particles being tracked decides how many iterations are stored in one output file. *saveitns* decides how frequently the particle data is stored and *particlestep* in NAMELIST *Outputctrl* decides what fraction of the total number of particles are stored.
- The parameters *minunitno* and *maxunitno* have to be specified so that there is no clash between unit numbers used in the host program.
- If *memorycheck* is .TRUE. then the code prints out an estimate of the RAM requirements and exits.
- The parameter *twcmethod* provides two ways in which the fluid-particle interaction force is distributed back to the fluid grid. When it is set to “pic” the force on each particle is distributed to the nodes of the cell in which the particle is located. When set to “sundaram” (?) the force is put back on to all the nodes which formed the interpolation stencil to obtain the fluid velocity at the particle position.

Namelist	Variable	Type	Options/description
PROPERTIES	ngroup	integer	number of particle diameter groups
	density	real	density of particle material
	heatcapacity	real	heat capacity of particle material
	thermalcond	real	thermal conductivity of particle material
	volfrac	real	total initial volume fraction
	M*	integer	ratio of number of real to computational particles
	diamin	real	minimum diameter of particles
	diamax	real	maximum diameter of particles
	distfunc*	char	“const-phi”, “const-np”, “log-normal” distributions
	lognormal_dmean	real	mean diameter in log-normal distribution
	lognormal_dmax	real	max diameter in log-normal distribution
	lognormal_sigma	real	standard dev. in log-normal distribution
	diadist	char	the distribution of diameters between diamin and diamax, “linear”, “log”
buffer_percent	integer	define extra particles than calculated, for 'add' operations	

- The parameter *randseed* with the option “*fixed*” allows repeatability when using random numbers (to initialize the particle positions). Maybe useful for debugging.

## 6.4 Use with TransAT

If using the particle tracking option in TransAT, the user needs to copy a sample input file **particle.inp** from `installdir/TransAT/input` and set the appropriate inputs.

Namelist	Variable	Type	Options/description
INTEGRATE	method	char	“2rk”, “4rk”, “adb”, “exp” schemes
	timelimit	char	“cfl”, “resptime”
	taufactor*	real	max-timestep = $\text{taufactor} \times \tau_{p,min}$
INTERPOLATE	scheme	char	“3-order”, 2-6 order Lagrangian polynomials

<b>Namelist</b>	<b>Variable</b>	<b>Type</b>	<b>Options/description</b>
FORCES*	dragflag	logical	viscous drag force
	addmflag	logical	added-mass force
	fluidflag	logical	fluid force
	buoyflag	logical	gravity force
	liftflag	logical	lift force (not implemented)
	bassetflag	logical	Basset history force (not implemented)
	thermoflag	logical	thermophoretic force (not implemented)
	Repmax*	real	maximum particle Reynolds number
	BiotPmax*	real	maximum particle Biot number
	langevinflag	logical	Langevin particle dispersion model
	tauL.constant	real	Langevin model: fluid eddy time
	sigmaF.constant	real	Langevin model: fluid fluctuation std. dev.

<b>Namelist</b>	<b>Variable</b>	<b>Type</b>	<b>Options/description</b>
FILES	outputfile*	char	filename to write standard output
	outfilesize	integer	maximum size of an output file (in MB)
	outformat	char	“formatted”, “unformatted”, output file format
	datatmpl*	char	output filename template
	saveitns*	integer	save output interval

<b>Namelist</b>	<b>Variable</b>	<b>Type</b>	<b>Options/description</b>
FLAGS	coupling	char	“one-way”, “two-way”, particle-fluid coupling
	twcmethod*	char	“pic”, “sundaram”, how the force is put back on the grid
	solve_temperature	logical	solve for particle temperature
	particle_source	logical	introduce particle sources
INITPOS	initdist	char	initial positions of particles “xy-plane”, “yz-plane”, “zx-plane” “volume”, “sphere”, “xy-circle” “yz-circle”, “xz-circle”
	planecoor	real	for planar initial positions, the constant coordinate value
	randseed*	char	for repeatable random distributions “fixed”, “not-fixed”
	seedvalue	integer	seed value for “fixed” option
	xrange(2)	real	fractional range of the domain in x-direction where particles can be placed initially
	yrange(2)	real	fractional range in y and
	zrange(2)	real	z directions, e.g. (0.1,0.9)
	vel_equilibrium	logical	initial velocity equal to fluid velocity
	mean_velocity(3)	real	initial particle mean velocity
	vel_std_dev(3)	real	initial particle mean velocity

<b>Namelist</b>	<b>Variable</b>	<b>Type</b>	<b>Options/description</b>
OUTPUTCTRL	particlestep*	integer	save only one in every particlestep particles
	savecnd	logical	save cell-number-densities
	cndfile	char	filename to store <i>cnd</i>
	calcpic	logical	tabulate the particles inside each cell
	calcmindist	logical	calculate the minimum-distance between the particles
	mindistfile	char	filename to store <i>mindist</i>
	calctpke	logical	compute the total-particle-kinetic-energy per unit volume
	tpkfile	char	filename to store <i>tpke</i>
	dumpoldvel	logical	write old velocity in output file
	dumpcell	logical	write cell indices in output file
dumpfvpp	logical	write fluid velocity at particle position in output file	

<b>Namelist</b>	<b>Variable</b>	<b>Type</b>	<b>Options/description</b>
BOUNDARY	wall_interaction	char	“reflect”, “absorb”
	inflow_volfrac	real	volume fraction of inflow stream
	inflow_randseed	char	for repeatable random distributions “fixed”, “not-fixed”
	inflow_seedvalue	integer	seed value for “fixed” option

Table 6.1: Input file description. The superscripted (\*) quantities are further described in the section.

## 6.5 Particle Temperature

This section describes the capabilities of Steinli in solving particle temperature equation.

### 6.5.1 Inputs

This feature is activated by setting `solve_temperature` to be TRUE in **Namelist: Flags** in input file `particle.inp` in the project folder.

```
&Flags
solve_temperature = .TRUE.
/
```

#### Initial temperature

The following inputs in **Namelist: Initpos** can be used to set the initial temperature of the particles.

```
&Initpos
init_temperature = 320.0 !Kelvin (default 0.0)
temp_equilibrium = .FALSE. !Set particle temperature equal to local
                        !fluid temperature (default FALSE)
/
```

The input `temp_equilibrium` overrides the input `init_temperature`.

#### Inflow temperature

Particle inflow temperature can be set in the `projectname.bc` file for each inflow boundary defined using the input `inflow_particle_temperature(nbc)`. For example:

```
&BOUNDARY_CONDITIONS
NRBND = 8
!-----
! nbound = 1
!-----
cbnd(1,1) = "west"
cbnd(1,2) = "inflow"
cbnd(1,3) = "nofor"
!
vbnd(1,1) = 0.78
bndtv(1) = 300.0
bnd_inflow_type(1) = 0
inflow_particle_volfrac(1) = 1.0e-4
inflow_particle_temperature(1) = 400.0
!
nbnd(1,1) = 2
```

$\text{nbnd}(1,2) = 2$   
 $\text{nbnd}(1,3) = 2$   
 $\text{nbnd}(1,4) = 10$   
 $\text{nbnd}(1,5) = 2$   
 $\text{nbnd}(1,6) = 2$   
...  
.

## 6.6 Particle Dispersion

This section describes the capabilities of Steinli in particle dispersion modelling using Langevin equation. The particle dispersion model is used in combination with the Reynolds-Averaged turbulence modelling approach.

### 6.6.1 Inputs

This feature is activated by setting **langevinflag** to be TRUE in **Namelist: Forces** in input file *particle.inp* in the project folder.

```
&Forces
Langevinflag = .TRUE.
tauL_constant = 0.11
sigmaF_constant = 0.31
/
```

Inputs **tauL\_constant** and **sigmaF\_constant** are constants in the Langevin dispersion model described below.

### 6.6.2 Langevin model

The particle fluctuation variance ( $\sigma_p$ ) is defined as a function of the fluid fluctuation variance ( $\sigma_f$ ) and the particle response time as:

$$\sigma_p = \sigma_f \left( 1 + \frac{\tau_p}{K\tau_f} \right)^{-1/2} \quad (6.2)$$

where  $\tau_f = \text{MIN}(\tau_e, \tau_r)$  is the effective eddy time scale,  $\tau_p = \rho_p d_p^2 / 18\mu$  is the particle inertial response time,  $K = 1 + 0.15Re_p^{2/3}$  is the drag correction for finite Reynolds number, and  $\sigma_f = \sqrt{C_\sigma k}$ . The model constants  $C_\tau$  and  $C_\sigma$  correspond to the input variables **tauL\_constant** and **sigmaF\_constant**, respectively and are assigned default values of 0.11 and  $1.69\sqrt{C_{mu}} \approx 0.5$ , respectively (?).

In  $\tau_f$ ,  $\tau_e = C_\tau k / \epsilon$  is the local eddy life time in an Eulerian sense, and  $\tau_r = L_e / |\mathbf{u}_p - \mathbf{u}_f|$  is the time taken by the particle to travel through the eddy (residence time), where  $L_e = C_\mu^{3/4} k^{3/2} / \epsilon$  is the characteristic length scale of an eddy.

The particle velocity fluctuation equation is written as follows:

$$\frac{du'_{pi}}{dt} = -\frac{u'_{pi}}{\tau'_{pi}} + \sigma_p \left( \frac{2}{\tau'_{pi}} \right)^{1/2} \zeta(t) \quad (6.3)$$

where  $\tau'_{pi}$  is the particle response time to turbulent fluctuations and  $\zeta(t)$  is a random function with a Gaussian probability density distribution and zero mean. The turbulent particle response time is given as:

$$\tau'_{pi} = \frac{\sigma_p \tau_f}{\sqrt{\sigma_p^2 + u_{ri}}} \left( 1 + \frac{\tau_p}{K\tau_f} \right) \quad (6.4)$$

where  $u_{ri}$  is the terminal velocity due to action of body forces such as gravity.

## 6.7 Point Particles Sources

This section describes the capabilities of Steinli in defining point particles sources.

### 6.7.1 Inputs

This feature is activated by setting `particles_source` to be `TRUE` in **Namelist: Flags** in input file `particle.inp` in the project folder.

```
&Flags  
particle_source = .TRUE.  
/
```

The point particle sources are defined in a file `particle_sources.dat` in the project folder. Each line in the file should contain 9 columns of data and the number of sources defined is calculated from the total number of lines in the file (no blank lines should be present).

```
x y z particlegroup frequency u v w T
```

where,  $x$ ,  $y$ ,  $z$ , are the coordinates of the source, *particlegroup* denotes the diameter group to which this source contributes (integer), *frequency* is the number of particles released per second (integer),  $u$ ,  $v$ ,  $w$ , are the particle injection velocity components, and  $T$  is the particle injection temperature.

## 6.8 Mathematical formulation

### 6.8.1 Momentum equations

The equation governing the particle motion is the well known equation given by Maxey and Riley (?). The equation is valid for a rigid sphere in non-uniform flow such that the sphere is isolated and far from any boundary, the Reynolds number for the relative motion between the particle and the fluid is small, and the particle size is much smaller than the smallest flow length scale. The equation is given as follows,

$$\begin{aligned} \frac{du_{pi}}{dt} = & (1 - \frac{\rho_f}{\rho_p}) g_i + \frac{\rho_f}{\rho_p} \frac{Du_i}{Dt} - \frac{\rho_f}{2\rho_p} \frac{d}{dt}(u_{pi} - u_i) - \frac{18\mu}{\rho_p d_p^2} (u_{pi} - u_i) \\ & - \frac{9}{\rho_p d_p} \sqrt{\frac{\mu\rho_f}{\pi}} \int_0^t \frac{d/d\tau(u_{pi} - u_i)}{\sqrt{t - \tau}} d\tau \end{aligned} \quad (6.5)$$

where  $d/dt$  denotes the time derivative along the particle path, and  $D/Dt$  denotes the time derivative following a fluid element. The terms in the right-hand side of Eq. (6.5) are referred to as the buoyancy, fluid (due to the fluid pressure gradient and viscous stresses), added-mass, Stokes drag, and Basset forces. For the case of particles much heavier than the fluid, Elghobashi and Truesdell (?) have shown that the only significant forces are the Stokes drag, buoyancy, and Basset forces. They also found that the Basset force was always an order of magnitude smaller than the drag and buoyancy forces.

Other forms of the forces and additional forces such as the lift force can be used under different circumstances.

### 6.8.2 Temperature equation

Each particle is assumed to be represented by a uniform temperature  $T_p$ . This assumption is valid under low particle Biot ( $Bi_p$ ) number conditions, where  $Bi_p = Nu_p \lambda / \lambda_p$ . The rate of change of particle temperature is given as:

$$\frac{dT_p}{dt} = \frac{Nu_p}{2\tau_p^T} (T_f(\mathbf{x}_p) - T_p) \quad (6.6)$$

where,  $Nu_p$  is the particle Nusselt number given as

$$Nu_p = 2 + 0.6 Re_p^{\frac{1}{2}} Pr^{\frac{1}{3}} \quad (6.7)$$

$\tau_p^T$  is the particle thermal response time defined as

$$\tau_p^T = \frac{\rho_p C_p d_p^2}{12\lambda} \quad (6.8)$$

$T_f(\mathbf{x}_p)$  is the fluid temperature at the particle position,  $C_p$  is the particle heat capacity, and  $\lambda$  is the fluid thermal conductivity.

### High Biot number model

Currently being implemented.

# Chapter 7

## Numerical Method

### 7.1 Governing Equations

The partial differential equations governing steady incompressible flows in nonorthogonal coordinates may be written in the following general form:

$$\frac{\partial}{\partial x_i}(C_i\phi - D_{i\phi}) = JS_\phi, \quad i = 1, 2, 3 \quad (7.1)$$

where the coefficients  $C_i$  related to convection,  $D_{i\phi}$  related to diffusion and the source terms  $S_\phi$  are given in Table 1 for different dependent variables  $\phi$ .  $J$  is the Jacobian of coordinate transformation between the general curvilinear system ( $x_i$ ) and a reference rectangular system ( $y_i$ ). Eq. 7.1 uses the velocity components  $V_i$  which are along the coordinates  $y_i$  instead of along the grid-aligned directions  $x_i$ , as shown in Figure 7.1.

Table 1 contains 7 conservation equations. The first is the continuity equation. The next three for  $\phi = V_1, V_2$  and  $V_3$  are the Navier-Stokes equations for laminar flows and the Reynolds equations for turbulent flows for which  $V_1, V_2$  and  $V_3$  are the time-averaged velocities. The eddy viscosity  $\mu_t$  is calculated by using the standard k- $\epsilon$  turbulence model ?. The transport equations for the turbulent kinetic energy  $k$  and its dissipation rate  $\epsilon$  are of the same form (Eq. 7.1), and their source terms as well as the various turbulence model constants are also given in Table 1. The last equation is for the transport of a scalar quantity  $S$  such as temperature, enthalpy

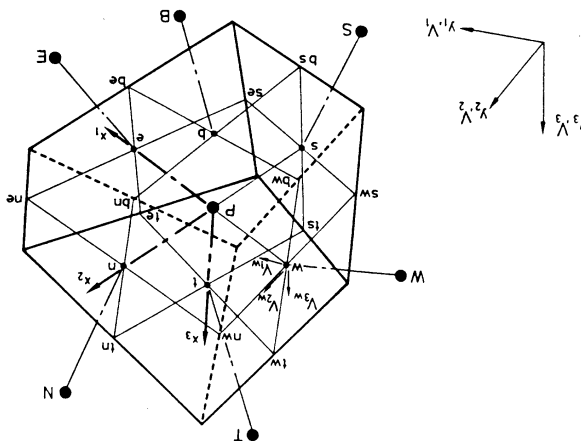


Figure 7.1: A typical control volume centered at node P

or species concentration; the specification of the constant  $\sigma_s$  and the source term  $S_s$  for this equation is left to the user.

Eq. 7.1 has two important features which are worthy of mention: (i) it is of conservation form in which all terms arising from the divergence operator are under differential operators, and (ii) it does not contain second-derivatives of coordinates (curvature terms) which are very sensitive to grid smoothness.

Table 1: Form of terms in the individual equations

No.	$\phi$	$D_{i\phi}$	$S_\phi$
1	1	0	0
2	$V_1$	$\frac{\Gamma_2}{J}(B_j^i \frac{\partial V_1}{\partial x_j} + \beta_j^i \omega_1^j)$	$-\frac{1}{J} \frac{\partial}{\partial x_j} (\beta_1^j p)$
3	$V_2$	$\frac{\Gamma_3}{J}(B_j^i \frac{\partial V_2}{\partial x_j} + \beta_j^i \omega_2^j)$	$-\frac{1}{J} \frac{\partial}{\partial x_j} (\beta_2^j p)$
4	$V_3$	$\frac{\Gamma_4}{J}(B_j^i \frac{\partial V_3}{\partial x_j} + \beta_j^i \omega_3^j)$	$-\frac{1}{J} \frac{\partial}{\partial x_j} (\beta_3^j p)$
5	k	$\frac{\Gamma_5}{J} B_j^i \frac{\partial k}{\partial x_j}$	$G - \rho\epsilon$
6	$\epsilon$	$\frac{\Gamma_6}{J} B_j^i \frac{\partial \epsilon}{\partial x_j}$	$(C_{1\epsilon} G - C_{2\epsilon} \rho\epsilon) \frac{\epsilon}{k}$
7	$S$	$\frac{\Gamma_7}{J} B_j^i \frac{\partial S}{\partial x_j}$	$S_s$

$$C_i = \rho \beta_j^i V_j, \quad \omega_j^i = \beta_j^n \frac{\partial V_i}{\partial x_n},$$

$$\beta_j^i = \text{cofactor of } \partial y_j / \partial x_i \text{ in } J, \quad J = \begin{vmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \frac{\partial y_3}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_3}{\partial x_2} \\ \frac{\partial y_1}{\partial x_3} & \frac{\partial y_2}{\partial x_3} & \frac{\partial y_3}{\partial x_3} \end{vmatrix}$$

$$B_j^i = \beta_n^i \beta_n^j,$$

$$\Gamma_n = \mu + \frac{\mu_t}{\sigma_n}, \quad \mu_t = \rho C_\mu k^2 / \epsilon,$$

$$G = \frac{\mu_t}{2J^2} \left( \frac{\partial V_i}{\partial x_n} \beta_j^n + \frac{\partial V_j}{\partial x_n} \beta_i^n \right)^2,$$

$$C_\mu = 0.09, \quad C_{1\epsilon} = 1.44, \quad C_{2\epsilon} = 1.92,$$

$$\sigma_2 = \sigma_3 = \sigma_4 = 1, \quad \sigma_5 = 1, \quad \sigma_6 = 1.3, \quad \sigma_7 = 1.$$

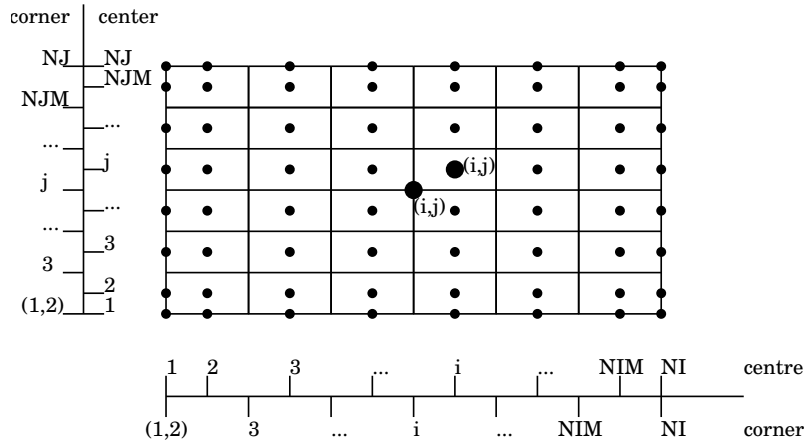


Figure 7.2: Layout and indexing of cell center and corner points

## 7.2 Layout and Indexing of Control Volumes

The flow field is discretised into a finite number of cells (control volumes) by a grid. The program adopts the practice of identifying the grid lines as cell faces and placing a computational node at the geometric center of each cell. Figure 7.1 shows a typical control volume centered at node P.

Figure 7.2 shows the layout and indexing of the center points (computational nodes) and corner points (grid nodes) of the control volumes. The indices  $(i, j, k)$  of the center points vary from 1 to NI for  $i$ -index, from 1 to NJ for  $j$ -index, and from 1 to NK for  $k$ -index, while those of the corner points start with 2 for each of the  $i$ -,  $j$ - and  $k$ -indices. The corner points with  $i=1$ ,  $j=1$  or  $k=1$  are actually not needed. They are set to the same as those with  $i=2$ ,  $j=2$  or  $k=2$  as dummy quantities in the program. It is clear from Figure 7.2 that in  $i$ -direction:

- the number of computational nodes = NI,
- the number of grid nodes = NI-1,
- the number of control volumes = NI-2.

The same holds in  $j$ - and  $k$ -directions. Besides the triple-index  $(i, j, k)$  the program also uses a single-index  $ii$  to identify the computational or grid nodes, which is necessary for vectorization. The single-index  $ii$  is related to the triple-index  $(i, j, k)$  by

$$ii = i + (j-1)NI + (k-1)NIJ \quad (7.2)$$

Thus, if the node P in Figure 7.1 has the triple-index  $(i, j, k)$  and the single-index  $ii$ , the corresponding indices for the neighboring node N are  $(i, j+1, k)$  and  $ii+NI$ .

Unlike the conventional staggered-grid arrangement, **TransAT** uses the same control volumes for all the flow variables. All the variables are stored at the computational nodes.

## 7.3 Geometric Quantities

All the geometric quantities required in the program can be calculated from the coordinates of the cell-vertices (grid nodes).

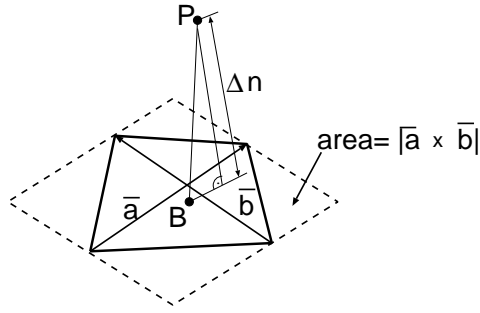


Figure 7.3: Normal distance from and area of a cell-face

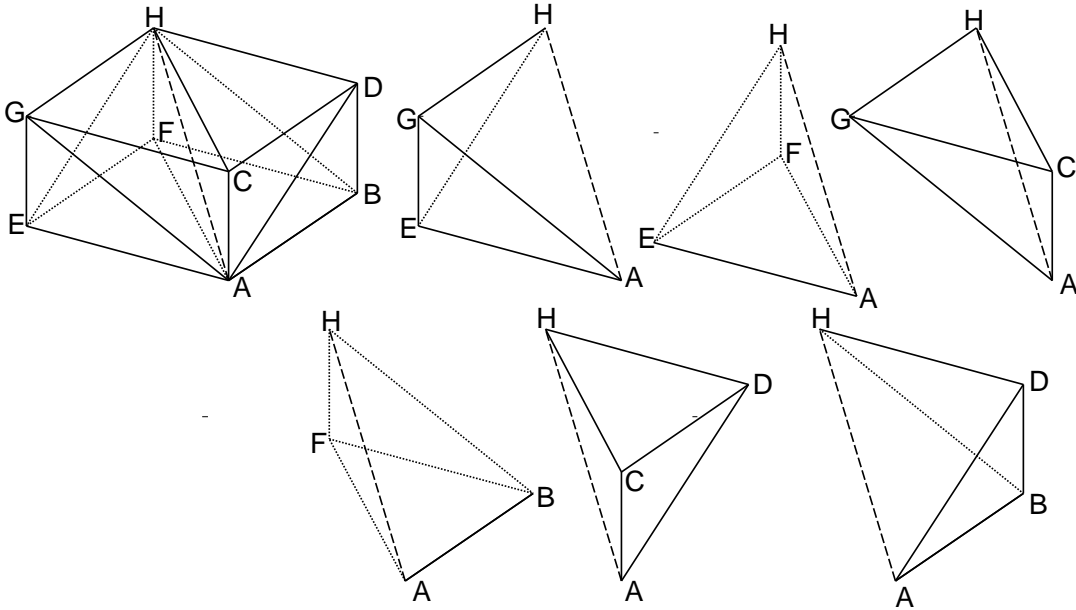


Figure 7.4: Volume of a cell

Normal Distance from and Area of a Cell-Face. These quantities which are needed in the formulation of boundary conditions can be easily calculated by using vector algebra. The normal direction of a cell-face can be determined by the vector product of two vectors  $\vec{a}$  and  $\vec{b}$  which connect opposite vertices of the cell-face, as shown in Figure 7.3. Therefore the unit normal vector is

$$\vec{n} = \vec{N} / |\vec{N}| \quad (7.3)$$

where

$$\vec{N} = \vec{a} \times \vec{b} \quad (7.4)$$

The normal distance  $\Delta n$  of a point  $P$  from the cell-face can be calculated as

$$\Delta n = \vec{BP} \cdot \vec{n} \quad (7.5)$$

where  $\vec{BP}$  is a vector joining the center point  $B$  of the cell-face and the point  $P$ . The area of the cell-face is

$$\Delta S = \frac{1}{2} |\vec{N}| \quad (7.6)$$

Volume of a Cell. The method suggested by ? is used to calculate the volume of a cell. Referring to Figure 7.4, the cell with points A, B, ..., H as its 8 vertices is decomposed into 6 tetrahedra, all containing the same diagonal joining the points A and H. The volume of the tetrahedron with vertices A, E, G and H can be calculated as:

$$\Delta V_1 = \frac{1}{6} |(\vec{AE} \times \vec{AG}) \cdot \vec{AH}| \quad (7.7)$$

Thus, the volume of the total cell can be written as :

$$\begin{aligned} \Delta V = \frac{1}{6} & |[(\vec{AE} \times \vec{AG}) + (\vec{AF} \times \vec{AE}) + (\vec{AG} \times \vec{AC}) \\ & + (\vec{AB} \times \vec{AF}) + (\vec{AC} \times \vec{AD}) + (\vec{AD} \times \vec{AB})] \cdot \vec{AH}| \end{aligned} \quad (7.8)$$

It is of interest to note that this way of calculating the cell volumes ensures the conservation of space, i.e., the sum of all cell volumes gives exactly the total volume of the solution domain, which is a necessary condition for guaranteeing true conservation of transported quantities.

Cofactors. The coefficient  $\beta_j^i$  appearing in the general transport equations in Table 1 is a cofactor of  $\partial y_j / \partial x_i$  in the Jacobian of the coordinate transformation. For  $i=2$  and  $j=3$  for example, we have

$$\beta_3^2 = - \left( \frac{\partial y_1}{\partial x_1} \frac{\partial y_2}{\partial x_3} - \frac{\partial y_2}{\partial x_1} \frac{\partial y_1}{\partial x_3} \right) \quad (7.9)$$

The discretised form of  $\beta_j^i$  can be expressed as:

$$\beta_j^i = \frac{b_j^i \Delta x_i}{\Delta x_1 \Delta x_2 \Delta x_3} \quad (\text{no summation on } i) \quad (7.10)$$

where  $\Delta x_i$  are the grid spacings along the coordinates  $x_i$ . The actual values of  $\Delta x_i$  are immaterial because the final form of the discretised equations will not contain any quantity related to the transformed space ( $x_i$ ). The  $b_j^i$  can be calculated from the respective increments in the Cartesian coordinates of the grid lines. Referring to Figure 7.1, the  $b_j^i$  at the point  $P$  for  $i=2$  and  $j=3$  can be calculated as:

$$(b_3^2)_P = [(y_2)_e - (y_2)_w] [(y_1)_t - (y_1)_b] - [(y_1)_e - (y_1)_w] [(y_2)_t - (y_2)_b] \quad (7.11)$$

where the subscripts  $e, w, t$  and  $b$  represent the center points of the corresponding cell-faces.

Interpolation Factors. Three sets of interpolation factors are given in the program for interpolating cell-face values from cell-center values. The factors associated with the cell centered at  $P$  (Figure 7.1) are defined as:

$$\begin{aligned} f_x &= l_{wW} / (l_{Pw} + l_{wW}) \\ f_y &= l_{sS} / (l_{Ps} + l_{sS}) \\ f_z &= l_{bB} / (l_{Pb} + l_{bB}) \end{aligned} \quad (7.12)$$

where  $l_{wW}$ ,  $l_{Pw}$ , ...,  $l_{bB}$  are the lengths of linear segments  $wW$ ,  $Pw$ , ...,  $bB$  (Figure 7.1). Therefore, the values of  $\phi$  at the cell-faces  $w$ ,  $e$ , ...,  $t$  can be calculated as:

$$\begin{aligned} \phi_w &= f_x(i, j, k) \phi_P + [1 - f_x(i, j, k)] \phi_W \\ \phi_e &= f_x(i+1, j, k) \phi_E + [1 - f_x(i+1, j, k)] \phi_P \\ \phi_s &= f_y(i, j, k) \phi_P + [1 - f_y(i, j, k)] \phi_S \\ \phi_n &= f_y(i, j+1, k) \phi_N + [1 - f_y(i, j+1, k)] \phi_P \\ \phi_b &= f_z(i, j, k) \phi_P + [1 - f_z(i, j, k)] \phi_B \\ \phi_t &= f_z(i, j, k+1) \phi_T + [1 - f_z(i, j, k+1)] \phi_P \end{aligned} \quad (7.13)$$

## 7.4 Discretization of Governing Equations

### 7.4.1 Flux balance equations

Integrating Eq. 7.1 over a typical control volume centered at P (Figure 7.1) leads to a flux balance equation

$$I_e - I_w + I_n - I_s + I_t - I_b = \int_{\Delta V} S_\phi dV \quad (7.14)$$

where  $I_f$  represents the total flux of  $\phi$  across the cell-face  $f$  ( $= e, w, n, s, t$  or  $b$ ). Each of the surface fluxes  $I_f$  contains a convective contribution  $I_f^C$  and a diffusive contribution  $I_f^D$ , that is

$$I_f = I_f^C + I_f^D \quad (7.15)$$

Eq. 7.14 involves no approximation and represents a finite-volume analogue of the differential equation ( 7.1).

### 7.4.2 Approximation of convection terms

The convective contribution in Eq. 7.15 can be written as:

$$I_f^C = C_f \phi_f \quad (7.16)$$

where  $C_f$  is the mass flux across the cell-face  $f$ , and can be calculated, for w-, s- and b-faces, as:

$$\begin{aligned} C_w &= (\rho b_j^1 V_j)_w \\ C_s &= (\rho b_j^2 V_j)_s \\ C_b &= (\rho b_j^3 V_j)_b \end{aligned} \quad (7.17)$$

The determination of  $\phi_f$  is a key element for both accuracy and stability of numerical solutions. The more accurate schemes tend to be less stable, and vice versa. Four schemes are incorporated in the program, leaving a choice to the user. They are the standard hybrid differencing ? which is basically of first-order accuracy for convection-dominant flows, third-order unbounded QUICK ?, second-order bounded SOUCUP ? and second-order bounded HLP ? . Consider the w-face of the control volume shown in Figure 7.5 and assume, without loss of generality, that  $V_w \geq 0$  ( $V_w$  is the normal velocity at the w-face). These schemes evaluate the face-value  $\phi_w$  as follows:

**a) Hybrid scheme** uses either central or upwind differencing according to a cell Peclet number  $Pe = |C_w/D_w|$  ( $D_w$  is the diffusive coefficient defined in Eq. 7.27). Thus

$$\phi_w = \begin{cases} 0.5(\phi_P + \phi_W) & \text{if } Pe \leq 2 \\ \phi_W & \text{otherwise} \end{cases} \quad (7.18)$$

**b) QUICK** approximates the face-value  $\phi_w$  by fitting a parabolic curve through three nodal values  $\phi_P$ ,  $\phi_W$  and  $\phi_{WW}$ :

$$\phi_w = \frac{3}{8}\phi_P + \frac{3}{4}\phi_W - \frac{1}{8}\phi_{WW} \quad (7.19)$$

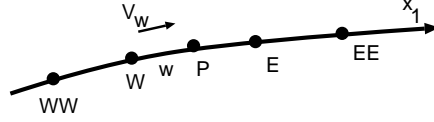


Figure 7.5: Nodes required by convection schemes in  $x_1$ -direction

c) **SOUCUP** combines three schemes, that is, second-order upwind, central and upwind differencing, with the switch from one scheme to another being controlled by an upwind-biased normalized variable

$$\hat{\phi}_W = (\phi_W - \phi_{WW}) / (\phi_P - \phi_{WW}) \quad (7.20)$$

Therefore

$$\phi_w = \begin{cases} 1.5\phi_W - 0.5\phi_{WW} & \text{if } 0 < \hat{\phi}_W \leq 0.5 \\ 0.5(\phi_P + \phi_W) & \text{if } 0.5 < \hat{\phi}_W < 1 \\ \phi_W & \text{if } |\hat{\phi}_W - 0.5| \geq 0.5 \end{cases} \quad (7.21)$$

d) **HPLA** can be written as

$$\phi_w = \phi_W + \gamma_w(\phi_P - \phi_W) \frac{\phi_W - \phi_{WW}}{\phi_P - \phi_{WW}} \quad (7.22)$$

where

$$\gamma_w = \begin{cases} 1 & \text{if } |\hat{\phi}_W - 0.5| < 0.5 \\ 0 & \text{otherwise} \end{cases} \quad (7.23)$$

All the schemes but Hybrid are implemented in the program in a deferred way proposed by ?

$$\phi_f^{l+1} = \phi_f^{u,l+1} + \lambda(\phi_f^{h,l} - \phi_f^{u,l}) \quad (7.24)$$

where  $u$  and  $h$  indicate the upwind and higher-order schemes, and  $l$  represents the iteration level; the parameter  $\lambda$  blends the two schemes with limiting values  $\lambda = 0$  for the upwind and  $\lambda = 1$  for the higher-order scheme. The differred correction leads to an always diagonally dominant coefficient matrix, thus lending the necessary stability to the numerical process while restoring higher-order solution at convergence.

All the schemes but Hybrid require two upstream nodes for each cell-face, which will involve a value outside the solution domain for a near-boundary control volume. Therefore in the program the Hybrid scheme is used for all the control volumes adjacent to boundaries.

### 7.4.3 Approximation of diffusion terms

The diffusive flux  $I_f^D$  in Eq. 7.15 can be divided into two parts

$$I_f^D = I_f^{DN} + I_f^{DC} \quad (7.25)$$

The first part  $I_f^{DN}$  contains only one term which has the first derivative of  $\phi$  in the direction "normal" to the cell-face  $f$ . It can be written, for the w-face for example, as:

$$I_w^{DN} = D_w(\phi_P - \phi_W) \quad (7.26)$$

where

$$D_w = [(b_1^1 b_1^1 + b_2^1 b_2^1 + b_3^1 b_3^1) \Gamma_\phi / \Delta V]_w \quad (7.27)$$

The second part  $I_f^{DC}$  contains all the other terms. Only the normal derivative diffusion flux,  $I_f^{DN}$ , is coupled with the convective flux,  $I_f^C$ , to calculate the main coefficients of the difference equations, while the cross-derivative diffusion flux,  $I_f^{DC}$ , is treated explicitly as a pseudo-source term to avoid the possibility of producing negative coefficients in an implicit treatment.

#### 7.4.4 Approximation of source terms

The source terms  $S_\phi$  are linearized as

$$S_\phi = S_\phi^U + S_\phi^P \phi_P \quad (7.28)$$

The coefficient  $S_\phi^P$  is defined so that it is always not larger than zero for all the conservation equations, and  $S_\phi^U$  always assumes non-negative values for both k- and  $\epsilon$ -equations. This enhances the stability of the numerical process and prevents the calculated values of k and  $\epsilon$  from becoming negative in low turbulence regions. The volume integral of the source term can therefore be approximated as

$$\int_{\Delta V} S_\phi dV \simeq S_\phi^U \Delta V + S_\phi^P \Delta V \phi_P \quad (7.29)$$

#### 7.4.5 Final form of discretised equations

After replacing all the terms in Eq. 7.14 by their discretised analogies, the following difference equation results:

$$A_P \phi_P = \sum_{nb} A_{nb} \phi_{nb} + S_U, \quad nb = W, E, S, N, B, T \quad (7.30)$$

where

$$A_P = \sum_{nb} A_{nb} - S_P \quad (7.31)$$

The main coefficients  $A_{nb}$  that relate the principal unknown  $\phi_P$  to its neighbors  $\phi_{nb}$  contain the combined contribution from convection and normal diffusion. The physical source term (Eq. 7.29) and the cross-derivative diffusion flux  $I^{DC}$  (Eq. 7.25) are included in the coefficients  $S_U$  and  $S_P$ .

In order to stabilize the iterative solution process, it is often necessary to under-relax the current solution by

$$\phi_P := \phi_P^o + \alpha_\phi (\phi_P - \phi_P^o) \quad (7.32)$$

where  $\alpha_\phi (\in [0, 1])$  is an under-relaxation factor and the superscript "o" refers to the old value at the previous iterative level. Introducing Eq. 7.32 into Eq. 7.30 leads to an under-relaxed equation which is of the same form as Eq. 7.30 except that the coefficients  $S_U$  and  $A_P$  are replaced by

$$S_U := S_U + \frac{1 - \alpha_\phi}{\alpha_\phi} A_P \phi_P^o \quad (7.33)$$

$$A_P := A_P / \alpha_\phi \quad (7.34)$$

## 7.5 Pressure-Velocity Coupling

The successful use of the non-staggered arrangement, which was long considered un-feasible in the past ?, relies on the way to establish a coupling between the pressure and velocity fields in the discretised momentum and continuity equations. **TransAT** uses the SIMPLE algorithm of ? and the momentum interpolation procedure of ? to achieve such coupling.

Under a guessed pressure field  $p^*$ , the discretised momentum equations can be written from Eq. 7.30 as:

$$V_{iP}^* = \alpha_u [H_{iP}^* + D_{1i}(p_w^* - p_e^*) + D_{2i}(p_s^* - p_n^*) + D_{3i}(p_b^* - p_t^*)] + (1 - \alpha_u)V_{iP}^o, \quad i = 1, 2, 3 \quad (7.35)$$

where

$$H_{iP}^* = \left( \sum_{nb} A_{nb} V_{i,nb}^* + S'_U \right) / A_P \quad (7.36)$$

$$D_{ij} = b_j^i / A_P \quad (7.37)$$

$V_{iP}^o$  is the old value of  $V_{iP}$  at the previous iteration,  $\alpha_u$  is the under-relaxation factor used for the momentum equations,  $S'_U$  is the part of  $S_U$  not containing the pressure gradients, and  $A_P$  and  $S_U$  are not replaced by Eqs. 7.33 and 7.34. The values of  $p^*$  at the cell-faces w, e, ..., t are calculated from linear interpolation between two adjacent cell-centers lying on either side of the faces. In general, the velocities  $V_i^*$  will not satisfy the continuity equation. Subtracting Eqs. 7.35 from their counterparts under the correct pressure field  $p$  and neglecting the terms  $H_{iP} - H_{iP}^*$  lead to the velocity and pressure corrections at the cell-center

$$V_{iP} = V_{iP}^* + \alpha_u [D_{1i}(p'_w - p'_e) + D_{2i}(p'_s - p'_n) + D_{3i}(p'_b - p'_t)], \quad i = 1, 2, 3 \quad (7.38)$$

$$p = p^* + p' \quad (7.39)$$

Using the momentum interpolation procedure, we can write the velocities at the w-face as ?:

$$V_{iw}^* = \alpha_u [\overline{H_{iP}^* + D_{2i}(p_s^* - p_n^*) + D_{3i}(p_b^* - p_t^*)} + \overline{D_{1i}(p_W^* - p_P^*)}] + (1 - \alpha_u)V_{iw}^o, \quad i = 1, 2, 3 \quad (7.40)$$

where the over-bar refers to those quantities which are calculated from linear interpolation on the cell-centers P and W. Subtracting Eqs. 7.40 from their counterparts under the correct pressure field  $p$  and neglecting the terms  $H_{iP} - H_{iP}^*$ ,  $H_{iW} - H_{iW}^*$  and the changes due to the cross-derivative pressure gradients, we obtain the velocity corrections at the w-face

$$V_{iw} = V_{iw}^* + \alpha_u \overline{D_{1i}} (p'_W - p'_P) \quad (7.41)$$

The discretised continuity equation can be written as:

$$C_e - C_w + C_n - C_s + C_t - C_b = 0 \quad (7.42)$$

From Eqs. 7.41 and 7.17, the mass flux  $C_w$  can be written as

$$C_w = C_w^* + A_W (p'_W - p'_P) \quad (7.43)$$

where

$$A_W = \alpha_u \rho_w (b_1^1 \overline{D_{11}} + b_2^1 \overline{D_{12}} + b_3^1 \overline{D_{13}})_w \quad (7.44)$$

and  $C_w^*$  is the mass flux calculated from the velocities  $V_{iw}^*$  given in Eqs. 7.40. Substituting the mass flux calculated by Eq. 7.43 and similar expressions for the other faces into Eq. 7.42 yields the following equation to determine the pressure correction  $p'$

$$A_P p'_P = \sum_{nb} A_{nb} p'_{nb} - Q, \quad nb = W, E, S, N, B, T \quad (7.45)$$

where

$$A_P = \sum_{nb} A_{nb} \quad (7.46)$$

$$Q = C_e^* - C_w^* + C_n^* - C_s^* + C_t^* - C_b^* \quad (7.47)$$

## 7.6 Boundary Conditions

The four most frequently encountered boundary conditions, i.e., inflow, outflow, symmetry and rigid wall conditions, are implemented in the program. For all dependent variables except pressure, these boundary conditions are specified in Sections 2.6.1–2.6.4. The pressures at all the boundary nodes are evaluated by linear extrapolation from values at interior nodes. The program also provides a provision allowing the user to specify boundary conditions other than these four types.

### 7.6.1 Inflow planes

At the inflow planes, the values of  $\phi$  ( $\phi = V_1, V_2, V_3, k, \epsilon, S$ ) are prescribed. Furthermore, the program defines the following 7 factors to normalize the residual (Section 2.7.1):

$$F_1 = \sum_{ii} \dot{m}_{ii}, \quad F_n = \sum_{ii} \dot{m}_{ii} \phi_{n,ii}, \quad n = 2, 3, \dots, 7 \quad (7.48)$$

with

$$\begin{aligned} \phi_2 = \phi_3 = \phi_4 &= (V_1^2 + V_2^2 + V_3^2)_{in}^{1/2}, \\ \phi_5 &= k_{in}, \quad \phi_6 = \epsilon_{in}, \quad \phi_7 = S_{in} \end{aligned} \quad (7.49)$$

where the summation index  $ii$  runs over all the inlet computational nodes,  $\dot{m}$  is the mass flux and the subscript "in" refers to the inlet values.  $F_n$  ( $n=1, 2, 3, 4, 5, 6$  or  $7$ ) is set to 1 if the corresponding value calculated by Eq. 7.48 is 0.

### 7.6.2 Outflow planes

At the outflow planes, the streamwise gradients of all variables are set to zero, implying a fully developed flow condition. In addition, the velocities  $V_j$  ( $j = 1, 2, 3$ ) are corrected in the following way to ensure the overall mass conservation at the outflow planes:

$$V_{j,ii} := V_{j,ii} F_a \quad (7.50)$$

$$F_a = F_1 / \sum_{ii} \dot{m}_{out,ii} \quad (7.51)$$

where  $ii$  runs over all the outlet computational nodes and  $\dot{m}_{out}$  is the outflow mass flux.

### 7.6.3 Symmetry planes

At the symmetry planes, the normal velocity component and the normal gradients of other variables are set to zero. The convective and normal diffusive fluxes  $I_f^C$  (Eq. 7.16) and  $I_f^{DN}$  (Eq. 7.25) are also explicitly set to zero, where "f" refers to the boundary surface.

### 7.6.4 Rigid wall

At the rigid walls, the no-slip condition is used for laminar flows, i.e., the velocity components of the flow are set to those of the wall. The standard wall-function approach ? is used for turbulent flows. In this, the resultant wall shear stress  $\vec{\tau}_w$  is related to the flow velocity vector  $\vec{V}$  by

$$\vec{\tau}_w = -\lambda_w \vec{V}_P \quad (7.52)$$

where

$$\lambda_w = \begin{cases} \mu/y_P & \text{if } y_P^+ < 11.6 \\ \rho C_\mu^{1/4} k_P^{1/2} \kappa / \ln(E y_P^+) & \text{otherwise} \end{cases} \quad (7.53)$$

$$y_P^+ = \rho C_\mu^{1/4} k_P^{1/2} y_P / \mu, \quad \kappa = 0.41, \quad E = 8.432 \quad (7.54)$$

the subscript  $P$  refers to the first control volume center from the wall, and  $y_P$  is the normal distance from the wall. Further, the diffusive flux of  $k$  is set to zero at the wall, and the near-wall values of the production rate  $G$  and the dissipation rate  $\epsilon$  are determined from

$$G_P = \frac{\tau_w^2}{\kappa \mu y_P^+} \quad (7.55)$$

$$\epsilon_P = \frac{C_\mu^{3/4} k_P^{3/2}}{\kappa y_P} \quad (7.56)$$

## 7.7 Solution Procedure

### 7.7.1 Convergence criterion

The residua are defined for the continuity equation as:

$$R_1 = \left[ \sum_{ii} (C_e - C_w + C_n - C_s + C_t - C_b)_{ii}^2 \right]^{1/2} / F_1 \quad (7.57)$$

and for all the other equations with numbers  $m=2$  to 7 in Table 1 as:

$$R_m = \left[ \sum_{ii} \left( \sum_{nb} A_{nb} \phi_{nb} + S_U - A_P \phi_P \right)_{ii}^2 \right]^{1/2} / F_m \quad (nb = E, W, N, S, T, B) \quad (7.58)$$

where  $ii$  runs over all the computational nodes and  $F_m$  ( $m=1$  to 7) are the normalization factors defined in Eq. 7.48. The calculation is declared converged when

$$R_{max} = \text{MAX}(R_1, R_2, R_3, R_4, R_5, R_6, R_7) \leq \text{EPS} \quad (7.59)$$

where EPS is the convergence criterion prescribed by the user.

### 7.7.2 Calculation sequence

The sequence in which the calculation is carried out is as follows:

- a. Initialize all field values by guess.
- b. Solve the  $V_1, V_2$  and  $V_3$ -momentum equations using the guessed pressure field.
- c. Solve the pressure-correction equation to obtain the pressure-correction at the cell-centers; correct the convective fluxes at the cell-faces, the velocities and pressure at the cell-centers.
- d. Solve the  $k$ - and  $\epsilon$ -equations and update  $\mu_t$ , if the flow is turbulent.
- e. Solve the scalar  $S$ -equation, if required.
- f. Return to step b with updated field values.

Sequence of steps b to f is repeated until the convergence criterion (Eq. 7.59) is satisfied.

A complete sequence of steps b to f is defined as an **outer** loop, and a solution of the system of Eqs. 7.30 for each variable over the entire solution domain as an **inner** loop. The system of Eqs. 7.30 is solved by using the strongly implicit procedure of ?. For all variables but the pressure-correction, one inner loop is normally enough to reduce the corresponding residue to a reasonable low level, especially in case of highly convective flows. However, sufficient inner loops are required for the pressure-correction  $p'$ -equation, mainly due to the fact that the  $p'$ -equation is of diffusion type in which the coefficient matrix is symmetric but without any convective fluxes, thus making convergence relatively slow. The inner loops of solving the  $p'$ -equation are stopped when either of the following two criteria is satisfied ?:

$$R^m \leq \lambda R^i \quad (7.60)$$

$$m > \text{NSWC}_1 \quad (7.61)$$

where  $R^i$  and  $R^m$  are the residua calculated in the same way as in Eq. 7.58 at the initial stage and after the  $m$ -th loop,  $\lambda = 0.1$  and  $\text{NSWC}_1=20$  are used in the program.

# Chapter 8

## Custom Utilities

In this chapter description of custom utilities are given.

### 8.1 Wind Safety of Tower Cranes

To analyse the safety of tower crane installations in a proposed construction site, the following data should be collected together.

- The STL file describing the site geometry.
- File(s) containing the coordinates of the crane location(s), where unsteady flow field is required - coordG1.txt
- Inflow data files - Inflow grid and u, v, w, signal files.

The utility *setupcrane* available in \$TRANSATDIR/bin should be used to set up simulation folders and input based on the flow directions of interest. The input should be specified in a file called *setupcrane.inp* in the site folder. The following is the description of how to set up the site folder.

- Create a folder, for example, Paris
- Copy STL file (paris.stl) to folder Paris
- Copy coordinate files (coord1.txt ... coordn.txt) to folder Paris
- Create a file called setupcrane.inp with appropriate inputs (see description below)
- Execute utility *setupcrane* in folder Paris

#### 8.1.1 Description of setupcrane.inp

The utility input file should contain the following inputs:

```
paris           !Name of project
paris.stl       !Name of STL file describing the surroundings
97 97 49        !Grid size in x, y, z (vertical) directions
1.0             !Scale of geometry file (= 1.0 if in meters)
```

```

7                !Number of Angles required
90              !Angles
120
Angle3
Angle4
Angle5
Angle6
Angle7
3                !Number of coordinate files
coord1.txt      !Coordinate file names
coord2.txt      !Coordinate file names
coord3.txt      !Coordinate file names
/home/user/.... /inflowgrid.dat !Location of Inflow grid file
/home/user/.... /inflowUvel.dat !Location of Inflow X velocity
/home/user/.... /inflowVvel.dat !Location of Inflow Y velocity
/home/user/.... /inflowWvel.dat !Location of Inflow Z velocity

```

The utility *setupcrane* will create the following items in the site folder (e.g. Paris).

- Sub-folders with names `paris_angle` (`paris_90`, `paris_120`, `paris_angle3`, etc.)
- The sub-folders have all the required inputs for TransAT, including grid, boundary conditions, problem definition, etc.
- A run script to sequentially run all the angles (`paris.run`). To run the script `$(installdir)/TransAT-n.n/bin` should be included in the environment variable `PATH`.
- Note: the utility sets up TransAT to run with a time step of 1 sec for 180 seconds. The inflow data should cover this time range.
- The utility requires that the folder `$(installdir)/TransATMesh/bin` be included in the environment variable `PATH`.

### 8.1.2 Output

For each angle simulated the corresponding time signal files are available in `paris_angle/ RESULT/ timesignal.set1.0000001`, where the set number corresponds the coordinate files in `setupcrane.inp`

## 8.2 Tecplot Animation Macro Utility: TransATMovie

## Chapter 9

# Best Practice Guidelines

9.1 Non-dimensional Numbers

9.2 Time Step Restrictions