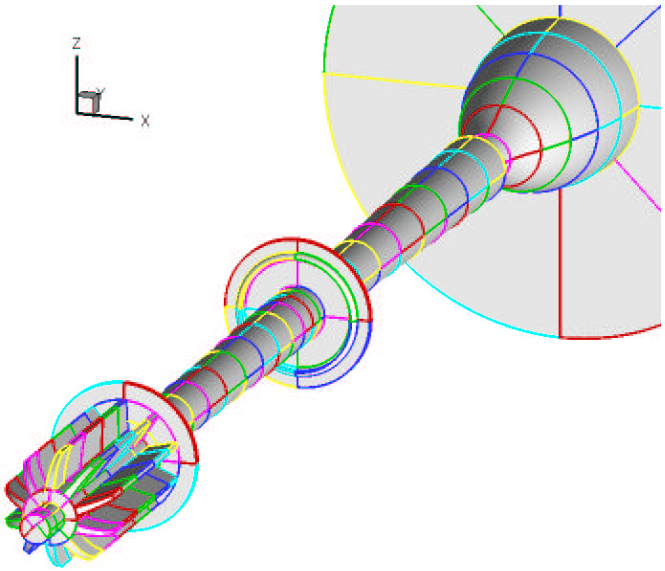
Rocflo User's Guide

Bono Wasistho and Jiri Blazek

Center for Simulation of Advanced Rockets
University of Illinois at Urbana-Champaign
Urbana, IL 61801



December 10, 2005

Contents

Chapter 1

Introduction

Rocflo is a general purpose, structured, multiblock, 3-D flow solver for the solution of the Euler or the Navier-Stokes equations. Rocflo contains plug-ins for the following physical modules:

- Turbulence (LES, DES and RANS models) – *Rocturb*
- Lagrangian particles (burning Al droplets) – *Rocpart*
- Eulerian particles (smoke) – *Rocsmoke*
- Chemical species – *Rocspecies*
- Radiation – *Rocrad*

In addition, it also contains plug-ins for specific periodic flow cases, called *Rocperi*.

The goal of this manual is to enable users to extract the source code from CVS, to compile and to run the base flow solver Rocflo. The physical modules are described in their respective manuals and hence are not further explained here. Also, a more detailed description of the numerical schemes and boundary conditions can be found in the Developer's Guide or in [1].

Chapter 2

Numerical Methodology

The base Rocflo code solves the time-dependent, 3-D Euler or Navier-Stokes equations on moving and/or deforming grids. The governing equations can be written in integral form as

$$\frac{\partial}{\partial t}\int_{\Omega} \vec{W}\, d\Omega + \oint_{\partial\Omega} (\vec{F}_c^M - \vec{F}_v)\, dS = \int_{\Omega} \vec{Q}\, d\Omega\,, \tag{2.1}$$

where $\vec{W}$ denotes the vector of conserved variables, $\vec{F}_c^M$ the convective fluxes on moving grid, $\vec{F}_v$ the viscous fluxes, and Ω is the control volume with the surface $\partial\Omega$. Furthermore, dS represents a surface element of $\partial\Omega$ and $\vec{Q}$ stands for the source term. The convective fluxes $\vec{F}_c^M$ become on a dynamic grid

$$\vec{F}_c^M = \vec{F}_c - V_t\vec{W} \tag{2.2}$$

with $\vec{F}_c$ denoting the standard convective fluxes and V_t being the contravariant velocity of the face of the control volume. Hence,

$$V_t = n_x\frac{\partial x}{\partial t} + n_y\frac{\partial y}{\partial t} + n_z\frac{\partial z}{\partial t}\,. \tag{2.3}$$

In Eq. (2.3), n_x, n_y, and n_z represent the components of the outward facing unit normal vector of the surface $\partial\Omega$.

In order to avoid errors induced by a deformation of the control volumes, the Geometric Conservation Law (GCL) must be satisfied. The integral form of the GCL reads [2]

$$\frac{\partial}{\partial t}\int_{\Omega} d\Omega - \oint_{\partial\Omega} V_t\, dS = 0\,. \tag{2.4}$$

The simultaneous solution of Eq. (2.1) and (2.4) is also termed the Arbitrary Lagrangian-Eulerian (ALE) method for dynamic grids.

Additional equations have to be solved along with Eqs. (2.1) and (2.4) in the case of multi-phase flow and for the species concentrations. Presently, the only gas model implemented assumes a thermally and calorically perfect fluid. The viscosity of the gas is calculated from Sutherland's law.

Rocflo employs a cell-centered, finite-volume type of spatial discretization. Currently, the only spatial discretization implemented is the central scheme with scalar artificial dissipation [3], which consists of a blend of adaptive second- and fourth-order differences. The scheme is second-order accurate on smoothly varying grids. Near shocks or other steep gradients of the solution, the scheme is reduced to first-order accuracy using a pressure switch. Two different pressure switches are implemented. The first one is the standard form based on normalized second differences. The second one is of TVD type [4].

Two different explicit schemes are available for the integration of the governing equations (2.1) in time. The first one is a 5-stage hybrid scheme [3] which is applied in the case of stationary flows. It is accelerated by local time stepping and implicit residual smoothing [5]. The second explicit time-stepping scheme is the classical 4-stage Runge-Kutta scheme, which is 4th-order accurate in time.

Rocflo provides a variety of boundary conditions, which are applicable in internal and external flows. These are:

- Injection
- Inflow / Outflow
- Far field
- Symmetry
- Block interface (conforming)
- Rotational or translational periodicity

Rocflo is implemented in Fortran-90 using its features for user defined data types, dynamic memory allocation, pointers, etc. The data structure is such that a different set of governing equations (e.g. Euler versus Navier-Stokes) can be solved in each block, using different spatial and temporal discretizations. Furthermore, the physical models can also vary between the blocks. Rocflo and the physical modules are parallelized using block decomposition and MPI.

Chapter 3

Building and Running

The source codes of Rocflo, of the physical modules, and of the utility programs are located in the `Rocstar/RocfluidMP/Codes` directory of the CVS repository. The code can be compiled either as a stand alone version or as a library to be linked with GENx. The user can also specify which physical modules are to be included during the compilation. The process of extracting the source code from CVS, compiling and running Rocflo is described in the next sections.

3.1 Extracting from CVS

In order to check out the source codes, type:

```
% cvs co -d RocfluidMP Rocstar/RocfluidMP/Codes
```

This will create the directory `RocfluidMP` in the current path. Besides the source codes, the directory will also contain a number of test cases and the documentation. The contents of the directory `RocfluidMP` is the following:

- `CODING_RULES` – coding rules for the fluids codes
- `Makefile` – main makefile used to compile Rocflo, the physical modules and utility programs
- `Makefile.AIX` – machine dependent makefile for IBM-SP
- `Makefile.IRIX64` – machine dependent makefile for SGI
- `Makefile.Linux` – machine dependent makefile for Linux
- `Makefile.SunOS` – machine dependent makefile for SUN
- `Makefile.common` – makefile with machine independent settings
- `Makefile.dep` – makefile for the generation of file dependencies

- `README` – quick overview of the fluid codes
- `TEMPLATE.F90` – template for subroutines
- `build_lib` – directory where the library is build
- `build_std` – directory where the standalone code is build
- `build_util` – directory where the utilities are build
- `calcs` – test cases for the verification and validation of Rocflo, cases for benchmarking the code
- `docs` – documentation including the user's and developer's manuals, and the description of code versions
- `genx` – routines required for the coupling to GENx
- `libflo` – library of routines specific to the Rocflo base solver
- `libfloflu` – library of routines common to Rocflo and the unstructured code
- `libflu` – library of routines specific to the unstructured code
- `modflo` – F90 modules specific to Rocflo
- `modfloflu` – F90 modules common to both fluids codes
- `modflu` – F90 modules specific to the unstructured code
- `rocflo` – routines of the base Rocflo solver
- `rocflu` – routines of the base unstructured solver
- `rocinteract` – routines of the interaction module
- `rocpart` – routines specific to the Lagrangian particles module
- `rocperi` – routines specific periodic flow cases
- `rocrad` – routines specific to the radiation module
- `rocsmoke` – routines specific to the Eulerian particles module
- `rocspecies` – routines specific to the species module
- `rocturb` – routines specific to the turbulence module
- `standalone` – routines specific to the stand alone code
- `utilities` – various utility programs (see Chapter 5)

3.2 Compiling

The source codes can be compiled by switching to the top-level directory (i.e. `RocfluidMP`) and typing:

```
% make SWITCH=1
```

on the command line. `SWITCH` can be one or multiple of the following:

- `RFLO` for Rocflo
- `PEUL` for Eulerian particles
- `PLAG` for Lagrangian particles
- `RADI` for radiation
- `SPEC` for species
- `TURB` for turbulence
- `PERI` for specifid periodic flows
- `MPI` for parallelization
- `DEBUG` for debugging
- `PROF` for profiling
- `POST` for post-processing

Hence,

```
% make RFLO=1 MPI=1 POST=1
```

compiles a parallel version of the base code with full optimization on and also compiler the post-processor.

The compilation process generates several executables in the top-level directory:

- `rflomp` – Rocflo solver
- `rflopost` – post-processor used to convert Rocflo's grid and solution files into the Tecplot format for visualization
- `rfloprep` – pre-processor used to generate the initial flow solution
- `rflosplit` – utility to split single-block grids into multiple blocks
- `rflosurf` – utility to output surface grids for the coupling with the structures code (Rocsolid or Rocfrac) within GENx

- `rflo2dto3d` – utility to convert 2-D grids into Rocflo's format
- `rfloblocks` – tool to find blocks located within a given box
- `rflo2flu` – tool to convert Rocflo's grid to Rocflu's hex grid
- `rocvav` – validation and verification utility used to compare Rocflo's results with experimental or theoretical data

If the user wishes, the executables can be copied to the directory `$HOME/bin` by typing:

```
% make RFLO=1 POST=1 install
```

3.3 Running

The flow solver is executed by typing the name (and path if necessary) of the executable (i.e. `rflomp`), followed by the name of the flow case and the verbosity level. So, for example:

```
% rflomp casename 2
```

One can choose one from the following verbosity levels:

- 0 – no output to console (*stdout*)
- 1 – moderate amount of output
- 2 – very verbose

Errors are always written to *stderr*. When Rocflo is run within GENx, the verbosity level is set per default to 1.

The descriptions how to run the utility codes can be found in Chapter 5. The input files required to run Rocflo as well as its output files are explained in the next chapter.

Chapter 4

Input and Output Files

The execution of Rocflo depends on the following five files:

- User input file
- Input file with boundary values
- Topology file (description of the boundary patches)
- Grid file
- Solution file (must always be present - flow solver does not generate an initial solution)

It should be noted that there is one grid and one solution file for all grid blocks. Also the other files are common to all blocks. There can be additional files, which store distributions of flow quantities for boundary patches. Furthermore, Rocflo can generate so called probe files, where values of the density, the velocity components, the static pressure and temperature are stored along with the physical time or the iteration number for a given location within the flow field. Rocflo can also read in or generate file which contains a time averaged flow quantities such as the velocity components (statistics file). Finally, Rocflo also stores the convergence history, probe data and the thrust into separate files.

The formats of the grid, solution and statistics file can be either native ASCII or binary. All other files are in a native ASCII format. All files are denoted by the name of the flow case (specified on the command line when invoking the flow solver) and an additional extension:

- `.inp` for user input
- `.bc` for boundary values
- `.top` for topology
- `.grda` for grid file in ASCII format, `.grdb` for binary format
- `.sola` for solution file in ASCII format, `.solb` for binary format

- `.prb` for probe file(s)
- `.thr` for thrust
- `.stata` for statistics file in ASCII format, `.statb` for binary format
- `.con` for convergence history
- `.degec` for degenerate edges and corners

The files with the user input and the boundary values consist of a various number of sections (like reference values, time stepping, numerical data, etc.). A certain number of options is associated with each section. Keywords are employed to distinguish between the sections and to designate particular user options. The begin of a section is marked by:

```
# SECTION_KEYWORD
```

where the keyword has to be in capital letters. The end of a section is denoted by #. The ordering of sections within a file is free and sections can be repeated. All text outside the sections and between the options is considered as comment. The options consist of a keyword (must be upper case) and of one or multiple integer, real or character values. There can be a comment **behind** the value(s). No particular ordering is enforced for options within a section. However, it is important that the section designation and options are positioned at the **beginning** of a line to be recognized by the parser.

4.1 User Input

The user input file has the extension `.inp`. It consists of the sections:

- *INITFLOW* – flow initialization (initial solution)
- *BLOCKMAP* – mapping of blocks to processors
- *FORMATS* – formats of the grid and of the solution files
- *FLOWMODEL* – inviscid or viscous flow
- *PERIODICFLOW* – specific periodic flows
- *GRIDMOTION* – grid motion
- *REFERENCE* – reference values
- *VISCMODEL* – model for computing the laminar viscosity
- *PROBE* – number and position(s) of probe(s)

- *THRUST* – location of the thrust plane and type of thrust
- *ACCELERATION* – type and values of acceleration terms
- *FORCES* – pressure and viscous forces
- *TURBULENCE* – turbulence modeling
- *SPECIES* – gas model and chemical species
- *CONPART* – continuum particles (Eulerian and Fast Eulerian approach)
- *DISPART* – discrete particles (Lagrangian approach)
- *RADIATION* – radiation modeling
- *INRT* – models for interaction of physical modules
- *MATERIAL* – material definitions (particles, smoke)
- *TIMESTEP* – time stepping
- *STATISTICS* – statistics (time averaged values)
- *MULTIGRID* – multigrid or successive grid refinement
- *NUMERICS* – parameters of the numerical method
- *POST* – parameters for post processing

An example of the file with all available options can be seen in `calcs/rocflo_example.inp`. In case some of the options or sections are not found in the input file, Rocflo supplies default values for the parameters.

4.1.1 INITFLOW Section

It contains the options:

- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *NDUMMY* – number of dummy cells (larger or equal 2)
- *VELX* – velocity component in x-direction [m/s]
- *VELY* – velocity component in y-direction [m/s]
- *VELZ* – velocity component in z-direction [m/s]

- *PRESS* – static pressure [Pa]
- *DENS* – density [kg/m^3]
- *XSPLIT* – x-coordinate of split location of two initial fields [m]
- *DSVELX* – downstream velocity in x-direction [m/s]
- *DSVELY* – downstream velocity in y-direction [m/s]
- *DSVELZ* – downstream velocity in z-direction [m/s]
- *DSPRESS* – downstream static pressure [Pa]
- *DSDENS* – downstream density [kg/m^3]

If some of the above values are changed, it is important to run `rfloprep` again in order to generate an updates initial solution file.

4.1.2 BLOCKMAP Section

It contains the options:

- *NBLOCKS* – 0 = automatic mapping of blocks to processors; otherwise the number of blocks per processor

4.1.3 FORMATS Section

It contains the options:

- *GRID* – grid format: 0 = ASCII, 1 = binary
- *SOLUTION* – solution format: 0 = ASCII, 1 = binary

4.1.4 FLOWMODEL Section

It contains the options:

- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *MODEL* – Euler (0) or Navier-Stokes equations (1) solved
- *MOVEGRID* – grid is moving (0 = no, 1 = yes)

4.1.5 PERIODICFLOW Section

It contains the options:

- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *FLOWKIND* – 0=none, 1=CPR, 2=channel
- *ISPLIT* – domain split in I-direction for parallel run: 0:NO 1:yes
- *JSPLIT* – domain split in J-direction for parallel run: 0:NO 1:yes
- *KSPLIT* – domain split in K-direction for parallel run: 0:NO 1:yes
- *CPREPSILON* – ratio injection to bulk mass rate (only for CPR)
- *PGRADTYPE* – 0: mass flux base, 1: wall stress based (only CHANNEL)

4.1.6 GRIDMOTION Section

It contains the options:

- *TYPE* – grid motion algorithm: 0=block-TFI 1=block-WgLaplacian 2=global-WgLaplacian 3=global-NuLaplacian 4=global-EllipticPDE
- *NITER* – number of Jacobi iterations (if TYPE¿0)
- *POWER* – power of inverse node distance in frame motion (if TYPE¿0)
- *AMPLIFX* – amplification factor for frame motion (if TYPE¿1)
- *AMPLIFY* – amplification factor for frame motion (if TYPE¿1)
- *AMPLIFZ* – amplification factor for frame motion (if TYPE¿1)
- *NEIGHBOR* – number of closest block-corner neigbours ¡=26 (if TYPE¿1)
- *NSURFMATCH* – number iteration for block interface matching ¿=2 (if TYPE¿1)
- *ORTHOWGHT* – weighting factor for block level orthogonality (if TYPE¿1)
- *WEIGHT* – weighting factor for cell center averaging (if TYPE¿2)
- *ORTHOCELL* – weighting factor for cell level orthogonality (if TYPE¿2)

4.1.7 REFERENCE Section

It contains the options:

- *ABSVEL* – velocity magnitude [m/s]
- *PRESS* – static pressure [Pa]
- *DENS* – density [kg/m^3]
- *CP* – specific heat coefficient at constant pressure [J/kgK]
- *GAMMA* – ratio of specific heats
- *LENGTH* – length [m]
- *RENUM* – Reynolds number
- *PRLAM* – laminar Prandtl number
- *PRTURB* – turbulent Prandtl number
- *SCNLAM* – laminar Schmidt number
- *SCNTURB* – turbulent Schmidt number

Reynolds number is used together with the density, the velocity and the length to calculate the viscosity of the fluid at 297 K. This reference viscosity is then modified according to the local temperature using the Sutherland's law.

4.1.8 VISCMODEL Section

It contains the options:

- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *MODEL* – viscosity model: 0 = Sutherland's law, 1 = fixed, 2 = Antibes formula
- *VISCOSITY* – reference laminar dynamic viscosity [kg/(ms)] (if omitted, a value based on Reynolds number is used instead)
- *REFTEMP* – reference temperature for Sutherland's law (110.0 K)
- *SUTHCOEF* – coefficient in Sutherland's law (288.16 K)

4.1.9 PROBE Section

It contains the options:

- *NUMBER* – number of probes within the fluids domain; zero means there are no probes
- *block val1 val2 val3* – block number and cell indices in the i-, j-, and k-direction for each probe with the first physical cell has the number 1; actual x-, y-, and z-coordinate can instead be used by setting the block number to 0
- *WRITIME* – time offset [s] to store probe data
- *WRIITER* – offset between iterations to store probe data
- *OPENCLOSE* – open and close probe file every time (0 = no, 1 = yes)

4.1.10 THRUST Section

It contains the options:

- *TYPE* – type of thrust: 0 = none, 1 = momentum thrust only, 2 = momentum and pressure thrust
- *PLANE* – thrust plane: x = const. (1), y = const. (2), z = const. (3)
- *COORD* – coordinate of the plane
- *PAMB* – ambient pressure in [Pa] (only if TYPE = 2)
- *WRITIME* – time offset [s] to store thrust history
- *WRIITER* – offset between iterations to store thrust history
- *OPENCLOSE* – open and close file with thrust every time (0 = no, 1 = yes)

4.1.11 ACCELERATION Section

It contains the options:

- *TYPE* – acceleration terms: 0 = off, 1 = on
- *ACCELX* – acceleration in x-direction [m/s^2]
- *ACCELY* – acceleration in y-direction [m/s^2]
- *ACCELZ* – acceleration in z-direction [m/s^2]

4.1.12 FORCES Section

It contains the options:

- *TYPE* – type of force computed: 0 = none, 1 = pressure, 2 = pressure and viscous (not yet implemented)

4.1.13 TIMESTEP Section

It contains the option:

- *FLOWTYPE* – type of flow: 0 = steady, 1 = unsteady

If the flow is steady, the following options apply:

- *STARTITER* – current iteration number (restart if greater than 0)
- *MAXITER* – maximum number of iterations (simulation is stopped if the number is reached)
- *RESTOL* – maximum density residual to stop the iteration process as having reached the steady state
- *WRIITER* – offset between iterations to store the solutions
- *PRNITER* – offset between iterations to print the convergence data to the screen and to store them in a file

If the flow is unsteady, the following options apply:

- *STARTTIME* – current physical time [s] (restart if greater than 0)
- *TIMESTEP* – user specified time step [s]; if the time step based on the CFL-conditions is smaller, it is used instead
- *MAXTIME* – maximum physical time to run the simulation [s]
- *WRITIME* – time offset to store solution [s]
- *PRNTIME* – time offset to print convergence to the screen and to store it in a file

If the flow is unsteady and dual time-stepping is used (FLOWTYPE=1 and SOLVERTYPE=1) the following options apply:

- *ORDER* – 2=second-order, 3=third-order scheme
- *MAXSUBITER* – max. number of subiterations
- *TOLSUBITER* – convergence tolerance of subiterations
- *PREDICTSOL* – predict start solution for subiterations (0=no, 1=yes)

4.1.14 STATISTICS Section

It contains the options:

- *DOSTAT* – collect time averaged data: 0 = no, 1 = yes
- *RESTART* – start a new statistics (0) or use previous data (1)
- *MIXTNSTAT* – number of statistics variables
- *MIXTSTATID* – ID's of the statistics variables (all in one line); each ID consists of two numbers which allows to mix two variables (all in one line); the codes of the variables are:
  - 0 = no variable
  - 1 = density [kg/m^3]
  - 2 = u velocity component [m/s]
  - 3 = v velocity component [m/s]
  - 4 = w velocity component [m/s]
  - 5 = static temperature [K]
  - 6 = static pressure [Pa]
  - 7 = speed of sound [m/s]
  - 8 = laminar viscosity [kg/(ms)]
  - 9 = laminar conductivity [N/sK]
- *TURBNSTAT* – number of statistics variables pertinent to the turbulence module
- *TURBSTATID* – ID's of the statistics variables of the turbulence module (all in one line); the codes of the variables are:
  - 0 = no variable
  - 1 = turbulent viscosity [kg/(ms)]
  - 2 = turbulent conductivity [N/sK]
  - 3 = Cd dynamic coefficient of LES model
  - 4 = $\tau_{11}\ \overline{\rho}(\widetilde{uu} - \widetilde{u}\widetilde{u})$ turbulent stress
  - 5 = $\tau_{22}\ \overline{\rho}(\widetilde{vv} - \widetilde{v}\widetilde{v})$ turbulent stress
  - 6 = $\tau_{33}\ \overline{\rho}(\widetilde{ww} - \widetilde{w}\widetilde{w})$ turbulent stress
  - 7 = $\tau_{12}\ \overline{\rho}(\widetilde{uv} - \widetilde{u}\widetilde{v})$ turbulent stress
  - 8 = free user module (user can use this for variable of his own choice, e.g. to collect statistics of viscous stress, M_{ij} or L_{ij})
  - 9 = free user module

4.1.15 MULTIGRID Section

It contains the options:

- *START* – grid level to start the simulation; 1 denotes the finest grid
- *CYCLE* – type of multigrid cycle: 0 = no multigrid, 1 = V-cycle, 2 = W-cycle
- *REFINE* – number of iterations before switching to the next finer grid level (Full Multigrid or successive grid refinement)

4.1.16 NUMERICS Section

It contains the options:

- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *CFL* – CFL number
- *SMOOCF* – coefficient of implicit residual smoothing (steady flow only); if set to a value equal or smaller than zero, no smoothing will be applied
- *DISCR* – type of space discretization: 0 = central scheme, 1 = Roe upwind scheme
- *K2* – dissipation coefficient $k^(2)$ (usually 0.5), central scheme
- *1/K4* – dissipation coefficient $1/k^(4)$ (usually 128), central scheme
- *PSWTYPE* – type of the pressure switch: 0 = standard, 1 = TVD (see Chapter 2 or Developer's Guide), central scheme
- *PSWOMEGA* – blending coefficient for PSWTYPE = 1 (see Developer's Guide), central scheme
- *ORDER* – spatial accuracy (only 2nd-order with the central scheme)
- *LIMFAC* – limiter coefficient (5-10), upwind scheme, requires correct settings of the reference values
- *ENTROPY* – entropy correction coefficient (0.05), upwind scheme
- *FEAVERAG* – cell2face and cell2edge averaging (0=uniform, 1=linear)

4.1.17 POST Section

It contains the options:

- *PLTTYPE* – plot type: 1=grid-only, 2=grid+solution
- *TIME* – time-stamp of solution to be post-processed (if unsteady)
- *ITER* – iteration-number of solution to be post-processed (steady)
- *OUTFORMAT* – output plot fmt: 1=generic-bin, 2=Tecplt-bin, 3=Tecplt-ASCII
- *STATSFLAG* – include statistics solution: 0=no, 1=yes
- *TURBFLAG* – include turbulence solution: 0=no, 1=yes
- *PLAGFLAG* – include particles solution: 0=no, 1=yes
- *RADIFLAG* – include radiation solution: 0=no, 1=yes
- *SPECFLAG* – include species solution: 0=no, 1=yes

The meaning of the options for the sections related to the physical modules can be found in the respective User's Manuals.

4.2 Boundary Values

The input file for the boundary values has the extension `.bc`. It consists of the sections:

- *BC_SLIPW* – slip wall boundary (Euler)
- *BC_NOSLIP* – noslip wall boundary (Navier-Stokes)
- *BC_WALLMODEL* – wall stress (momentum transfer) model at noslip wall boundary (Navier-Stokes)
- *BC_INFLOW_TOTANG* – inflow boundary based on total pressure/temperature
- *BC_INFLOW_VELTEMP* – inflow boundary based on velocity and temperature
- *BC_INFLOW_VELPRESS* – inflow boundary based on velocity and pressure
- *BC_OUTFLOW* – partly non-reflecting outflow boundary
- *BC_FARF* – far field (external flow)
- *BC_INJECT* – injection boundary
- *BC_PEUL_INFLOW* – inflow boundary for smoke

- *BC_PEUL_FARF* – far field boundary for smoke
- *BC_PEUL_INJECT* – injection boundary for smoke
- *TBC_SINUSOIDAL* – sinusoidal function to modify boundary values in time
- *TBC_STOCHASTIC* – stochastic function to modify boundary values in time
- *TBC_WHITENOISE* – noise function to modify boundary values in time

An example of the file with all available options can be seen in `calcs/rocflo_example.bc`. The file has to include all the relevant sections and options for the flow case in question. Otherwise, the execution of Rocflo will be stopped with an error message.

4.2.1 BC_SLIPW Section

It contains the options:

- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *PATCH* – range of boundary patches within the above range of blocks to which the following options apply; if both values are zero, the options apply to all patches within the range of blocks
- *EXTRAPOL* – order of extrapolation to the dummy cells (0 or 1)
- *MAXCHANGE* – max. relative change of ρ or ρE before the extrapolation to the dummy cells is switched from 1st- to 0-th order (between 0.1 and 1.0)
- *MOTION* – moving boundary (0 fixed, 1 moving)
- *XVEL* – x-velocity of moving boundary [m/s] (if MOTION=1)
- *YVEL* – y-velocity of moving boundary [m/s] (if MOTION=1)
- *ZVEL* – z-velocity of moving boundary [m/s] (if MOTION=1)

4.2.2 BC_NOSLIP Section

It contains the options:

- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *PATCH* – range of boundary patches within the above range of blocks to which the following options apply; if both values are zero, the options apply to all patches within the range of blocks

- *ADIABAT* – wall boundary condition: 0 = wall temperature is given, 1 = wall is adiabatic
- *DISTRIB* – single value (0) for all faces of the patch or distribution (1) to be read in from a file
- *FILE* – name of the file to read the distribution from (if DISTRIB = 1)
- *TWALL* – wall temperature (if ADIABAT = 0)

4.2.3 BC_WALLMODEL Section

It contains the options:

- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *PATCH* – range of boundary patches within the above range of blocks to which the following options apply; if both values are zero, the options apply to all patches within the range of blocks
- *MODEL* – 0=no 1=loglay (2)=bndlay 3=external (feed τ_{wall} distribution)
- *REFPOINT* – reference point (1 to max. wall normal points in the block)
- *ROUGHNESS* – roughness size [m]

Without wall model (MODEL=0), the noslip wall boundary set the wall stresses by taking the normal difference of the wall parallel velocities directly, generally less accurate than using a wall model.

4.2.4 BC_INFLOW_TOTANG Section

It contains the options:

- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *PATCH* – range of boundary patches within the above range of blocks to which the following options apply; if both values are zero, the options apply to all patches within the range of blocks
- *TYPE* – type of inflow: 0 = supersonic, 1 = subsonic, 2 = mixed
- *DISTRIB* – single value (0) for all faces of the patch or distribution (1) to be read in from a file

- *FILE* – name of the file to read the distribution from (if DISTRIB = 1)
- *PTOT* – total pressure [Pa]
- *TTOT* – total temperature [K]
- *BETAH* – flow angle about the horizontal axis [deg]
- *BETAV* – flow angle about the vertical axis [deg]
- *MACH* – Mach number (if TYPE = 0 or 2)

Setting to BC_INFLOW is equivalent to BC_INFLOW_TOTANG which is the default inflow boundary condition.

4.2.5 BC_INFLOW_VELTEMP Section

It contains the options:

- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *PATCH* – range of boundary patches within the above range of blocks to which the following options apply; if both values are zero, the options apply to all patches within the range of blocks
- *TYPE* – type of inflow: 0 = supersonic, 1 = subsonic, 2 = mixed
- *DISTRIB* – single value (0) for all faces of the patch or distribution (1) to be read in from a file
- *PROFILE* – inlet velocity profile: 0 = uniform, 1 = cylindrical Taylor profile
- *FILE* – name of the file to read the distribution from (if DISTRIB = 1)
- *VELX* – velocity in x-direction [m/s]
- *VELY* – velocity in y-direction [m/s]
- *VELZ* – velocity in z-direction [m/s]
- *TEMP* – inlet temperature [K]
- *PRESS* – inlet static pressure [Pa] (only used in supersonic inlet)
- *AXIALPOWER* – power ratio in Taylor axial velocity (default=2.)
- *NORMALFACT* – division factor in Taylor transverse velocity (default=1.)
- *RECYCTURB* – 0=steady-inflow, 1=turbulence-recycling-inflow
- *AMPLITUDE* – fluctuations amplitude

4.2.6 BC_INFLOW_VELPRESS Section

It contains the options:

- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *PATCH* – range of boundary patches within the above range of blocks to which the following options apply; if both values are zero, the options apply to all patches within the range of blocks
- *TYPE* – type of inflow: 0 = supersonic, 1 = subsonic, 2 = mixed
- *DISTRIB* – single value (0) for all faces of the patch or distribution (1) to be read in from a file
- *PROFILE* – inlet velocity profile: 0 = uniform, 1 = cylindrical Taylor profile
- *FILE* – name of the file to read the distribution from (if DISTRIB = 1)
- *VELX* – velocity in x-direction [m/s]
- *VELY* – velocity in y-direction [m/s]
- *VELZ* – velocity in z-direction [m/s]
- *TEMP* – inlet temperature [K] (only used in supersonic inlet)
- *PRESS* – inlet static pressure [Pa]
- *AXIALPOWER* – power ratio in Taylor axial velocity (default=2.)
- *NORMALFACT* – division factor in Taylor transverse velocity (default=1.)
- *RECYCTURB* – 0=steady-inflow, 1=turbulence-recycling-inflow
- *AMPLITUDE* – fluctuations amplitude

4.2.7 BC_OUTFLOW Section

It contains the options:

- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *PATCH* – range of boundary patches within the above range of blocks to which the following options apply; if both values are zero, the options apply to all patches within the range of blocks

- *TYPE* – type of outflow: 0 = supersonic, 1 = subsonic, 2 = mixed
- *DISTRIB* – single value (0) for all faces of the patch or distribution (1) to be read in from a file
- *FILE* – name of the file to read the distribution from (if DISTRIB = 1)
- *MODEL* – 0=standard model, 1=partly non-reflecting
- *PRESS* – static pressure [Pa] (if TYPE = 1 or 2)
- *NRCOEF* – non-reflecting coefficient (default=1.)

4.2.8 BC_FARF Section

It contains the options:

- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *PATCH* – range of boundary patches within the above range of blocks to which the following options apply; if both values are zero, the options apply to all patches within the range of blocks
- *DISTRIB* – single value (0) for all faces of the patch or distribution (1) to be read in from a file
- *FILE* – name of the file to read the distribution from (if DISTRIB = 1)
- *MACH* – Mach number
- *ATTACK* – angle of attack [deg]
- *SLIP* – angle of side slip [deg]
- *PRESS* – static pressure [Pa]
- *TEMP* – static temperature [K]

4.2.9 BC_INJECT Section

It contains the options:

- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks

- *PATCH* – range of boundary patches within the above range of blocks to which the following options apply; if both values are zero, the options apply to all patches within the range of blocks

- *EXTRAPOL* – order of extrapolation to the dummy cells (0 or 1); 0th-order results in a more robust scheme

- *DISTRIB* – single value (0) for all faces of the patch or distribution (1) to be read in from a file

- *FILE* – name of the file to read the distribution from (if DISTRIB = 1)

- *MFRATE* – mass flow rate [kg/(m^{2}s)]

- *TEMP* – injection temperature [K]

- *MAXCHANGE* – max. relative change of T before the extrapolation to the dummy cells is switched from 1st- to 0-th order (between 0.1 and 1.0)

4.2.10 BC_PEUL_INFLOW Section

It contains the options:

- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks

- *PATCH* – range of boundary patches within the above range of blocks to which the following options apply; if both values are zero, the options apply to all patches within the range of blocks

- *DISTRIB* – single value (0) for all faces of the patch or distribution (1) to be read in from a file

- *FILE* – name of the file to read the distribution from (if DISTRIB = 1)

- *DENS1* – smoke density for particle type 1

- *DENS2* – smoke density for particle type 2

- *DENS3* – smoke density for particle type 3

4.2.11 BC_PEUL_FARF Section

It contains the options:

- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *PATCH* – range of boundary patches within the above range of blocks to which the following options apply; if both values are zero, the options apply to all patches within the range of blocks
- *DISTRIB* – single value (0) for all faces of the patch or distribution (1) to be read in from a file
- *FILE* – name of the file to read the distribution from (if DISTRIB = 1)
- *DENS* – default smoke density for all particle types
- *DENS2* – smoke density for particle type 2
- *DENS5* – smoke density for particle type 5

4.2.12 BC_PEUL_INJECT Section

It contains the options:

- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *PATCH* – range of boundary patches within the above range of blocks to which the following options apply; if both values are zero, the options apply to all patches within the range of blocks
- *DISTRIB* – single value (0) for all faces of the patch or distribution (1) to be read in from a file
- *FILE* – name of the file to read the distribution from (if DISTRIB = 1)
- *FRAC* – default mass fraction for all particle types
- *FRAC3* – smoke mass fraction for particle type 3

4.2.13 TBC_SINUSOIDAL Section

It contains the options:

- – boundary condition (INFLOW, OUTFLOW, INJECT, NOSLIP, FARF) and name of the variable to modify (TTOT, PTOT, MFRATE, TWALL, etc.); it must be the first non-blank line
- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *PATCH* – range of boundary patches within the above range of blocks to which the following options apply; if both values are zero, the options apply to all patches within the range of blocks
- *ONTIME* – time to start [s] (on from beginning if negative: default)
- *OFFTIME* – time to stop [s] (on until end if negative: default)
- *AMP* – relative amplitude (function = (1 + AMP*sin(...))*mean)
- *FREQ* – frequency [1/s]
- *PHASE* – angle of sin argument at t = 0 [deg] (default = 0.)

4.2.14 TBC_STOCHASTIC Section

It contains the options:

- – boundary condition (INFLOW, OUTFLOW, INJECT, NOSLIP, FARF) and name of the variable to modify (TTOT, PTOT, MFRATE, TWALL, etc.); it must be the first non-blank line
- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *PATCH* – range of boundary patches within the above range of blocks to which the following options apply; if both values are zero, the options apply to all patches within the range of blocks
- *ONTIME* – time to start [s] (on from beginning if negative: default)
- *OFFTIME* – time to stop [s] (on until end if negative: default)
- *AMP* – standard deviation of log(process values)
- *TIMECOR* – integral time scale of autocorrelation

- *SHAPE* – autocorrelation: underdamped, SHAPE > 1. (default = 1.)
- *MINCUT* – minimal multiplicative factor (none if negative: default)
- *MAXCUT* – maximal multiplicative factor (none if negative: default)

4.2.15 TBC_WHITENOISE Section

It contains the options:

- – boundary condition (INFLOW, OUTFLOW, INJECT, NOSLIP, FARF) and name of the variable to modify (TTOT, PTOT, MFRATE, TWALL, etc.); it must be the first non-blank line
- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *PATCH* – range of boundary patches within the above range of blocks to which the following options apply; if both values are zero, the options apply to all patches within the range of blocks
- *SUBSTEP* – new value each step (0: default) or non-zero substep (1)
- *ONTIME* – time to start [s] (on from beginning if negative: default)
- *OFFTIME* – time to stop [s] (on until end if negative: default)
- *AMP* – relative amplitude (function = (1 + AMP*rand[-1,1])*mean)

4.2.16 TBC_PIECEWISE Section

It contains the options:

- – boundary condition (INFLOW, OUTFLOW, INJECT, NOSLIP, FARF) and name of the variable to modify (TTOT, PTOT, MFRATE, TWALL, etc.); it must be the first non-blank line
- *BLOCK* – range of blocks to which the following options apply; if both values are zero, the options apply to all blocks
- *PATCH* – range of boundary patches within the above range of blocks to which the following options apply; if both values are zero, the options apply to all patches within the range of blocks
- *ONTIME* – time to start using this TBC
- *OFFTIME* – time to stop using this TBC

- *ORDER* – 0 = piecewise constant (default), 1 = piecewise linear
- *NJUMPS* – number of points at which behavior changes (4 in example below)
- *FRAC* – 0.000 ! fraction of input value of variable before first time
- *TIME* – 0.001 ! 1^{st} time [s] at which behavior changes
- *FRAC* – 0.100 ! next fraction attained (constant) or ramped to (linear)
- *TIME* – 0.002 ! 2^{nd} time [s] at which behavior changes
- *FRAC* – 0.300
- *TIME* – 0.003
- *FRAC* – 0.600
- *TIME* – 0.004 ! final time at which behavior changes
- *FRAC* – 1.000 ! final value for constant case; *ignored* for linear case

4.3 Topology

The topology file has the extension `.top`. The file is automatically generated either by the `makeflo` utility when converting grid files from Gridgen's format into Rocflo's format, or by the `rflosplit` program (see Chapter 5).

The header of the topology file consists of two line with comments and the total number of blocks. For each block the topology file contains one line with the block number and the number of grid levels. A second line contains the number of patches on the particular block and the number of physical cells in i-, j- and k-direction. For each patch there is a single line with 13 columns:

1. type of the boundary condition:
   - 10 – inflow
   - 20 – outflow
   - 30 – block boundary (continuous grid)
   - 60 – slip wall, grid freely moved on fixed surface by Rocflo
   - 61 – slip wall, grid freely moved on fixed surface by Rocprop
   - 62 – slip wall, grid and surface fixed
   - 63 – slip wall, surface may slide or stretch in x-direction by Rocprop
   - 64 – slip wall, surface may slide or stretch in y-direction by Rocprop

- 65 – slip wall, surface may slide or stretch in z-direction by Rocprop
- 66 – slip wall, surface may slide in plane by Rocprop
- 70 – noslip wall
- 80 – far field
- 90 – injection
- 100 – symmetry
- 110 – translational periodicity
- 120 – rotational periodicity

2. number of the block face (see Fig. 4.1)
3. start index of the first patch coordinate (cf. Fig. 4.2; see Subsection 3.7.7 in the Developer's guide for the explanation of cyclic directions)
4. end index of the first patch coordinate
5. start index of the second patch coordinate (l_2 in Fig. 4.2)
6. end index of the second patch coordinate
7. number of the source block (or 0 if the boundary is not between blocks)
8. face number of the source block (see Fig. 4.1)
9. start index of the first patch coordinate of the source patch (cf. Fig. 4.2)
10. end index of the first patch coordinate of the source patch
11. start index of the second patch coordinate of the source patch (l_2 in Fig. 4.2)
12. end index of the second patch coordinate of the source patch
13. flag for coupling to an external solver (0 = not coupled, 1 = coupled)

Those local patch coordinates l_1, l_2 of the current and the source block which correspond to each other are denoted by the minus sign (l_1 could correspond to l_2 of the source patch). The end indices of l_1, l_2 of the source patch can be smaller than their start indices.

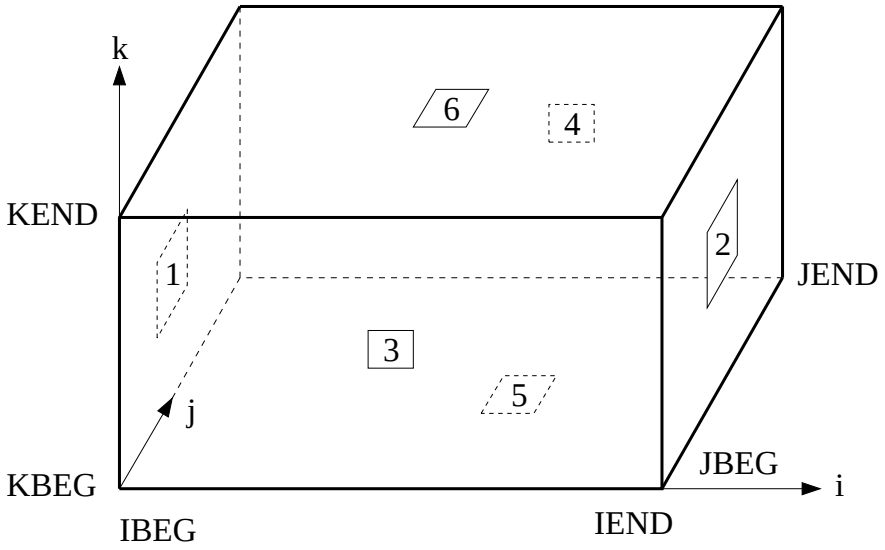


Figure 4.1: Numbering of the sides of the computational space and of the block boundaries.

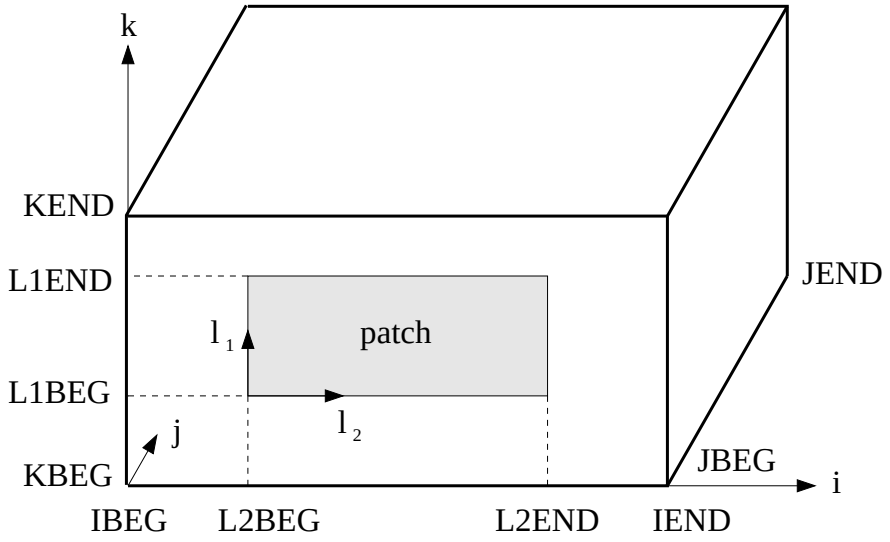


Figure 4.2: Coordinates of a boundary patch in computational space. The patch has its own local coordinate system l_1, l_2, which is based on cyclic directions.

4.4 Grid

The grid file has the extension `.grda[_stamp]` for the ASCII format and `.grdb[_stamp]` for the binary format, respectively. The additional key `_stamp` is added when the grid is moving and/or deforming (thus only for unsteady flow). In this case, `_stamp` denotes the physical time (in seconds) at which the grid file was stored. It has the format 1PE11.5. Hence,

```
casename.grdb_1.23000E-01
```

represents the grid at the time $t = 0.123$ sec. A grid file can be generated by converting a 2-D grid into Rocflo's format using the `rflo2dto3d` utility program. Another possibility is to generate the grid by Gridgen and to convert it into Rocflo's format using `makeflo`, as described in Chapter 5.

4.5 Solution

The solution file has the extension `.sola[_stamp]` for the ASCII format and `.solb[_stamp]` for the binary format, respectively. The additional key `_stamp` can take two different forms. In the case of a steady flow, it is a six digit integer (with all leading zeros - format I6.6) which represents the iteration number. Hence,

```
casename.sola_000123
```

represents the solution at the 123rd iteration.

In the case of an unsteady flow, `_stamp` denotes the physical time (in seconds) at which the solution file was stored. It has the format 1PE11.5. Hence,

```
casename.solb_1.23000E-01
```

represents the solution at the time $t = 0.123$ sec.

The initial solution file (zeroth iteration or $t = 0$) is generated by the pre-processor `rfloprep` using the user input file described in Section 4.1.

4.6 Statistics

The solution file has the extension `.stata[_stamp]` for the ASCII format and `.statb[_stamp]` for the binary format, respectively. The additional key `_stamp` denotes the physical time (in seconds) at which the solution file was stored. It has the format 1PE11.5. The format of the statistics file corresponds to that of the solution file. Variables, which are to be time averaged, are selected in the user input file as described in Subsection 4.1.14.

4.7 Probe

The extension of the probe file(s) is `.prb[_number]`, where `_number` is an integer in the format I3.3 representing the number of the probe. The file contains for each iteration or time step a single line with the iteration number or time and the values of the density, the velocity components, the static pressure and temperature. The number of probes and their location can be specified in the user input file (see Subsection 4.1.9). The offset (in terms of iterations or physical time) at which the data is written to the probe file can be set by the option *PRNITER* or *PRNTIME* in the user input file (cf. Subsection 4.1.13).

4.8 Thrust

The extension of the thrust file is `.thr`. The file contains for each iteration or time step a single line with the iteration number or time and the values of the momentum, pressure, and the total thrust. The location of the thrust plane can be specified in the user input file. The offset (in terms of iterations or physical time) at which the data is written to the thrust file can also be set in the user input file (see Subsection 4.1.13).

4.9 Convergence

The convergence file has the extension `.con`. For each iteration or time step, it contains a single line with with the iteration number or time, the density residual as $log(\Delta\rho/\Delta\rho_{initial})$, the forces in all three coordinate directions (in [N]), the inflow mass (in [kg/s]) and the outflow mass (also in [kg/s]). The offset (in terms of iterations or physical time) at which the data is written to the convergence file can be set by the option *PRNITER* or *PRNTIME* in the user input file (cf. Subsection 4.1.13).

4.10 Degenerate Edges and Corners

The information regarding degenerate edges and corners can be found in the file with the extension `.degec`. The file contains the region numbers in which degenerate edges and corners occur, and the degenerate edge and corner numbers, along with the degeneration type codes. The degeneration type codes are defined as follows.

Edge degeneration type:

- 1 – EDGE IN PATCH (¡ 4 interior cells meet at the edge)
- -1 – EDGE DETACHED (¿ 4 interior cells meet at the edge)

Corner degeneration type:

- 1 – CORNER IN EDGE (6 interior cells meet at the corner)
- 2 – CORNER IN PATCH (4 interior cells meet at the corner)
- -1 – CORNER DETACHED (¿ 8 interior cells meet at the corner)

Chapter 5

Utility Programs

The compilation process described in Section 3.2 creates the following utility programs:

- `rfloprep` – pre-processor used to generate the initial flow solution
- `rflopost` – post-processor used to convert Rocflo's grid and solution files into the Tecplot format for visualization
- `rflosplit` – utility to split single-block grids in Rocflo's format into multiple blocks
- `rflosurf` – utility to output surface grids for the coupling with the structures code (Rocsolid or Rocfrac) within GENx
- `rflo2dto3d` – utility to convert 2-D grids into Rocflo's format
- `rfloblocks` – tool to find blocks located within a given box
- `rflo2flu` – tool to convert Rocflo's grid to Rocflu's hex grid
- `rocvav` – validation and verification utility used to compare Rocflo's results with experimental or theoretical data

One further utility called `makeflo` has to be compiled separately. It can be found in `~olawlor/csar/makeflo/vers1.9`. This program is used to convert a grid generated by Gridgen into Rocflo's grid and topology file. It also allows to split a grid (in Gridgen's format only) into a given number of blocks.

5.1 rfloprep

The pre-processor is invoked as:

```
% rfloprep casename verbosity
```

with the required input from .inp file:

```
* grid level        => .inp file: # MULTIGRID: START
* input grd format  => .inp file: # FORMATS: GRID
* output sol format => .inp file: # FORMATS: SOLUTION
* unsteadiness      => .inp file: # TIMESTEP: FLOWTYPE
* spec. heat ratio  => .inp file: # REFERENCE: GAMMA
* initial solution  => .inp file: # INITFLOW: all
```

The program requires the grid file `casename.grda` or `casename.grdb`, the user input file `casename.inp` (see Section 4.1), the topology file `casename.top`, and the boundary condition file `casename.bc` to be present. The bc file is for a pre-run bc consistency check by rfloprep.

5.2 rflopost

The post-processor is invoked as:

```
% rflopost casename time/iter verbosity
```

with the required input from .inp file:

```
* grid level         => .inp file: # MULTIGRID: START
* unsteadiness       => .inp file: # TIMESTEP: FLOWTYPE
* turbulence         => .inp file: # TURBULENCE: TURBMODEL
*                    => .inp file: # TURBULENCE: OUTPUTNUMBER
* gas properties     => .inp file: # REFERENCE
*                    => .inp file: # VISCMODEL
*                    => .inp file: # MATERIAL (if MP is active)
* statistics         => .inp file: # STATISTICS (optional)
* plot type          => .inp file: # POST: PLTTYPE
* time (unsteady)    => .inp file: # POST: TIME (will be comand line input)
* iteration (steady) => .inp file: # POST: ITER
* output plot format => .inp file: # POST: OUTFORMAT
* include statistics => .inp file: # POST: STATSFLAG
* include turbulence => .inp file: # POST: TURBFLAG
* include particles  => .inp file: # POST: PLAGFLAG
* include radiation  => .inp file: # POST: RADIFLAG
* include species    => .inp file: # POST: SPECFLAG
```

The program requires the grid file `casename.grda` or `casename.grdb`, the user input file `casename.inp` (see Section 4.1), the topology file `casename.top`, and the solution file `casename.sola_stamp` or `casename.solb_stamp` to be present. The statistics solution file `casename.stata_stamp` or `casename.statb_stamp` and turbulence solution file `casename.turba_stamp` or `casename.turbb_stamp` should also exist if the statistics and turbulence are active, and STATSFLAG=1 and TURBFLAG=1. Similar holds for other physical modules. It should be noted that the binary format for Tecplot requires the library

`libtec.a` of the Tecplot distribution to be linked with `rflopost`. Otherwise, only the output plot formats 1 and 3 will be available.

5.3 rflosplit

The tool to split single-block grids is invoked as:

```
% rflosplit
```

with no additional flags. The program then asks for the case names of the old and the new grid, the grid format, the split direction (either i, j or k), and for the number of blocks to be generated.

5.4 rflosurf

The utility to generate surface grids for the interface between Rocflo and one of the structures codes is invoked as:

```
% rflosurf casename level grid
```

with the flags:

- *level* – grid level (> 0, 1 represents the finest grid)
- *grid* – format of Rocflo's grid file: 0 = ASCII, 1 = binary

The program generates the file `casename.im` with the surface grid in ASCII format. The file is then further processed by the `surfdiver` tool. The program requires the topology file `casename.top` and the grid file `casename.grda` (or `casename.grdb`) to be present.

5.5 rflo2dto3d

The tool to convert 2-D grids is invoked as:

```
% rflo2dto3d
```

with no additional flags. The program then asks for the name of the 2-D file, the case name, the number of cells in the third direction, the depth of the 3-D grid, and the format of Rocflo's grid file. The program generates a grid file in Rocflo's format, but **not** the topology file.

5.6 rfloblocks

The utility to find blocks within a specified box is invoked as:

```
% rfloblocks casename format xmin xmax ymin ymax zmin zmax
```

with the flags:

- *format* – format of Rocflo's grid file: 0 = ASCII, 1 = binary

and `xmin` ... `zmax` being the coordinates of the box. The program requires the topology file `casename.top` and the grid file `casename.grda` (or `casename.grdb`) to be present.

5.7 rflo2flu

The utility to convert Rocflo's grid to Rocflu's hex grid is invoked as:

```
% rflo2flu casename verbosity
```

with the required input from .inp file:

```
* grid level        => .inp file: # MULTIGRID: START
* Rocflo grd format => .inp file: # FORMATS: GRID
* Rocflu grd format => .inp file: # FORMATS: SOLUTION
```

The program requires the grid file `casename.grda` or `casename.grdb`, the user input file `casename.inp` (see Section 4.1), and the topology file `casename.top` to be present.

5.8 rocvav

The utility to generate surface grids for the interface between Rocflo and one of the structures codes is invoked as:

```
% rocvav casename verbosity
```

with the flag *verbosity* being either 0 (minimum output), 1 (moderate), or 2 (output all). The program compares two streams to each other and writes the norm of the error to the screen. Stream 1 is composed of the grid and the solution of Rocflo (in a future version also of the unstructured solver). Stream 2 can either be a reference solution generated by Rocflo, or an analytical or experimental data.

The program depends on the following input files:

- `casename_s1.grda` or `casename_s1.grdb` for the grid file of stream 1
- `casename_s1.sola_stamp` or `casename_s1.solb_stamp` for the solution file of stream 1; the meaning of the keyword `_stamp` is explained in Section 4.5

- `casename.inp_RVAV` for the control file

In the case the second stream is also Rocflo's solution, the file `casename_s2.sola_stamp` or `casename_s2.solb_stamp` has to be supplied as well.

The control file `casename.inp_RVAV` consists of the following sections and options:

- *RVAV_STREAM1* – options for stream 1:
  - *RVAV_FLOW_TYPE* – steady (0) or unsteady (1) flow
  - *RVAV_FILE_TYPE* – type of solution: 10 = computed, 20 = experimental, 30 = analytical
  - *RVAV_GRID_TYPE* – format of grid file: 0 = ASCII, 1 = binary
  - *RVAV_SOLUTION_TYPE* – format of solution file: 0 = ASCII, 1 = binary
- *RVAV_STREAM2* – the same as above for stream 2 with the additional option:
  - *RVAV_SIMILARITY_TYPE* – type of similarity solution (in case of analytical solution, i.e. if *RVAV_FILE_TYPE* = 30):
    - 131 – Proudman-Culick solution (inviscid, injection driven flow)
    - 132 – compressible Blasius solution (laminar flat plate)
    - 133 – GAMM bump (inviscid subsonic flow with constant total pressure)
- *RVAV_COMPARISONS* – options related to the comparison of the streams:
  - *RVAV_NUMBER_OF_COMPARISONS* – number of comparisons
  - two lines for each comparison (one for stream 1 and one for stream 2) consisting of the keys:
    - operation: 10 = compute errors, 20 = plot errors, 30 = compute and plot (plot file is in Tecplot format - not implemented yet)
    - variable:
      1 = ρ
      2 = ρu
      3 = ρv
      4 = ρw
      5 = ρE
      11 = u
      12 = v
      13 = w
      14 = p
      15 = T
      101 = T_0
      102 = p_0

201 = s
301 = ω_x
302 = ω_y
303 = ω_z

* block number
* start and end index, step in i-direction
* start and end index, step in j-direction
* start and end index, step in k-direction

An example of the control file is provided in `calcs/example.inp_RVAV`.

5.9 makeflo

The tool to convert Gridgen's grid and boundary file into Rocflo's grid and topology file is invoked as:

```
% makeflo -splitaxis axis-number gridgen blocks topology grid
```

with the flags:

- *splitaxis* – option to split the blocks in certain direction
- *axis-number* – 0=x-, 1=y-, 2=z-direction
- *gridgen* – name of Gridgen's grid file (extension `.grd`)
- *procs* – number of blocks the grid should be split into
- *topology* – name of Rocflo's topology file (i.e. `casename.top`)
- *grid* – name of Rocflo's grid file (i.e. `casename.grda` or `casename.grdb`); the extension of the grid file determines whether the file should be stored in ASCII (`.grda`) or binary format (`.grdb`).

Omitting the *splitaxis axis-number* option lets Makeflo to peform splitting according to the most optimum for load balancing, but in general results in a non-conforming block partition. The program depends on a file which relates Gridgen's boundary numbers to Rocflo's boundary types. The name of the file is the same as of Gridgen's grid file with the extension `.bcmp`. The file consists of lines, one for each boundary number, with the format:

- Gridgen's boundary condition number
- Rocflo's boundary condition number
- flag for boundaries coupled to the structures code and/or Rocburn: 0 = no coupling, 1 = coupled

The following boundary condition numbers are recognized by Rocflo:

- 10 – inflow
- 20 – outflow
- 30 – block boundary (continuous grid)
- 60 – slip wall, grid freely moved on fixed surface by Rocflo
- 61 – slip wall, grid freely moved on fixed surface by Rocprop
- 62 – slip wall, grid and surface fixed
- 63 – slip wall, surface may slide or stretch in x-direction by Rocprop
- 64 – slip wall, surface may slide or stretch in y-direction by Rocprop
- 65 – slip wall, surface may slide or stretch in z-direction by Rocprop
- 66 – slip wall, surface may slide in plane by Rocprop
- 70 – noslip wall
- 80 – far field
- 90 – injection
- 100 – symmetry
- 110 – translational periodicity
- 120 – rotational periodicity

An example of the file is given in `calcs/makeflo_example.bcmp`. The source code of `makeflo` is located in ∼`olawlor/csar/makeflo/vers1.9`.

Chapter 6

Examples and Test Problems

In the next sections, it will be demonstrated how to set up and run some test problems with Rocflo. All necessary input data for the examples are obtained when the flow solver is checked out from CVS (see Chapter 3).

6.1 Transonic Channel Flow

The first test case represents a transonic internal flow in a channel with bump. The data of the test case is stored in `calcs/verification/channel/rocflo/2D`. The grid for the channel is depicted in Fig. 6.1. The height of the bump is 10% of its length. The upper boundary represents the symmetry line, the lower boundary is the wall. The flow enters the domain from the left side.

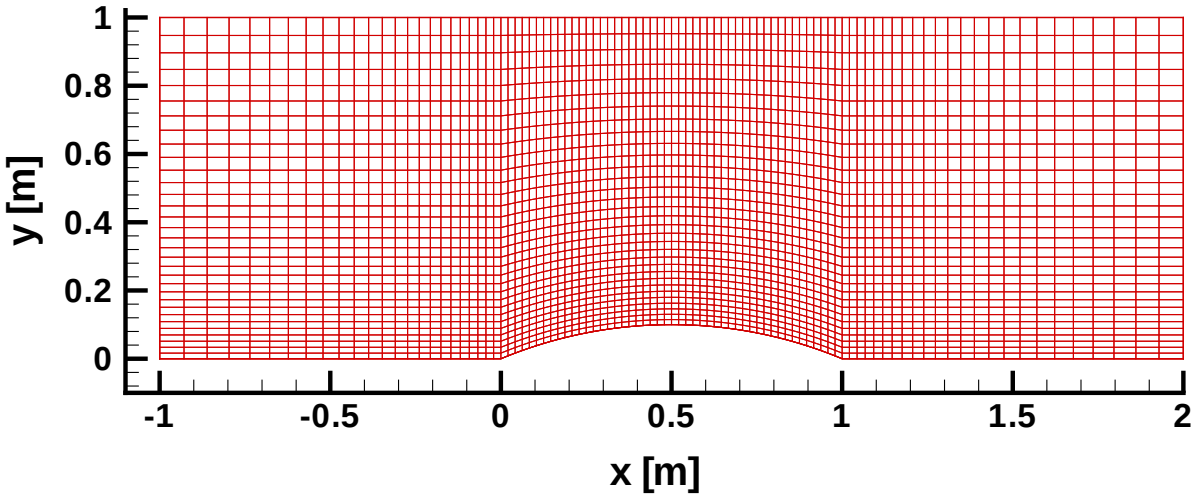


Figure 6.1: Channel flow; grid with 96x32x2 cells.

The initial solution can be generated by:

```
% rfloprep channel 1 0 0
```

which creates the file `channel.sola_000000`. The flow solver is started using:

```
% rflomp channel 2
```

The residual drops below 10^{-5} after 820 iterations and the solver stops. The directory contains now the solution file `channel.sola_000820` with the steady state flow field. It also contains the convergence history in the file `channel.con`. The convergence history is plotted in Fig. 6.2.

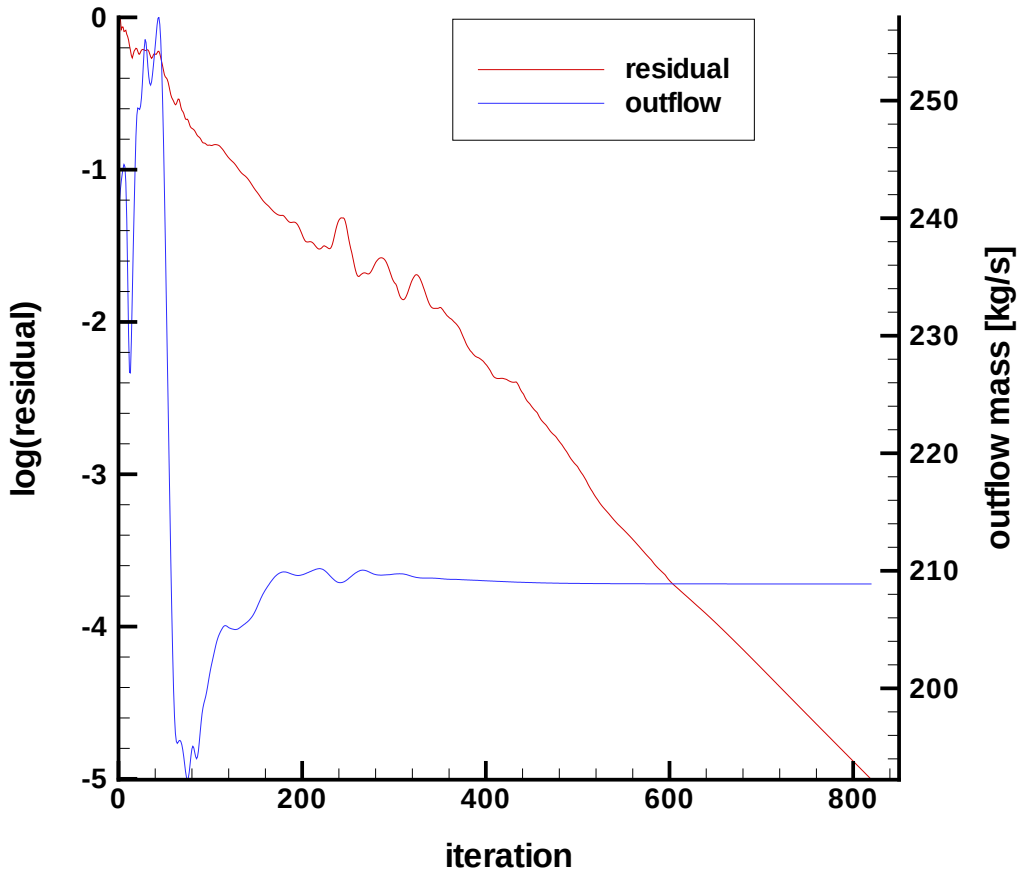


Figure 6.2: Channel flow; convergence history.

In order to visualize the solution, the post-processing utility has to be run:

```
% rflopost channel 2 1 820 2
```

This generates the file `channel.plt`. The plot file can be directly visualized by Tecplot. Contours of the static pressure are shown in Fig. 6.3. The Mach number distribution along the wall of the channel is displayed in Fig. 6.4.

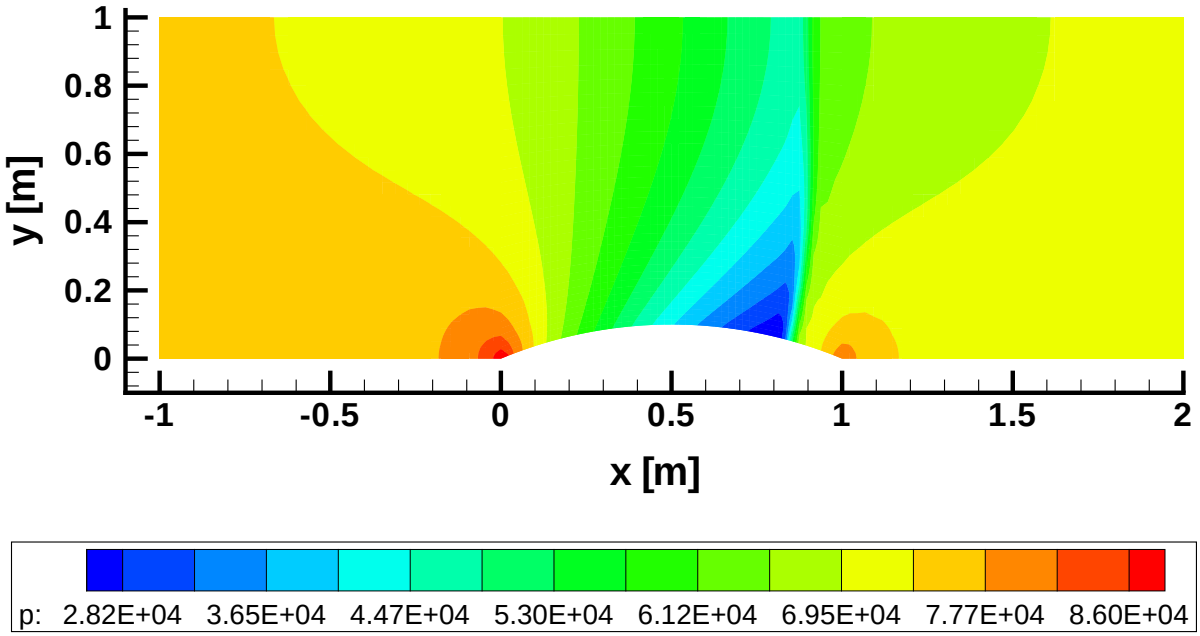


Figure 6.3: Channel flow; contours of static pressure.

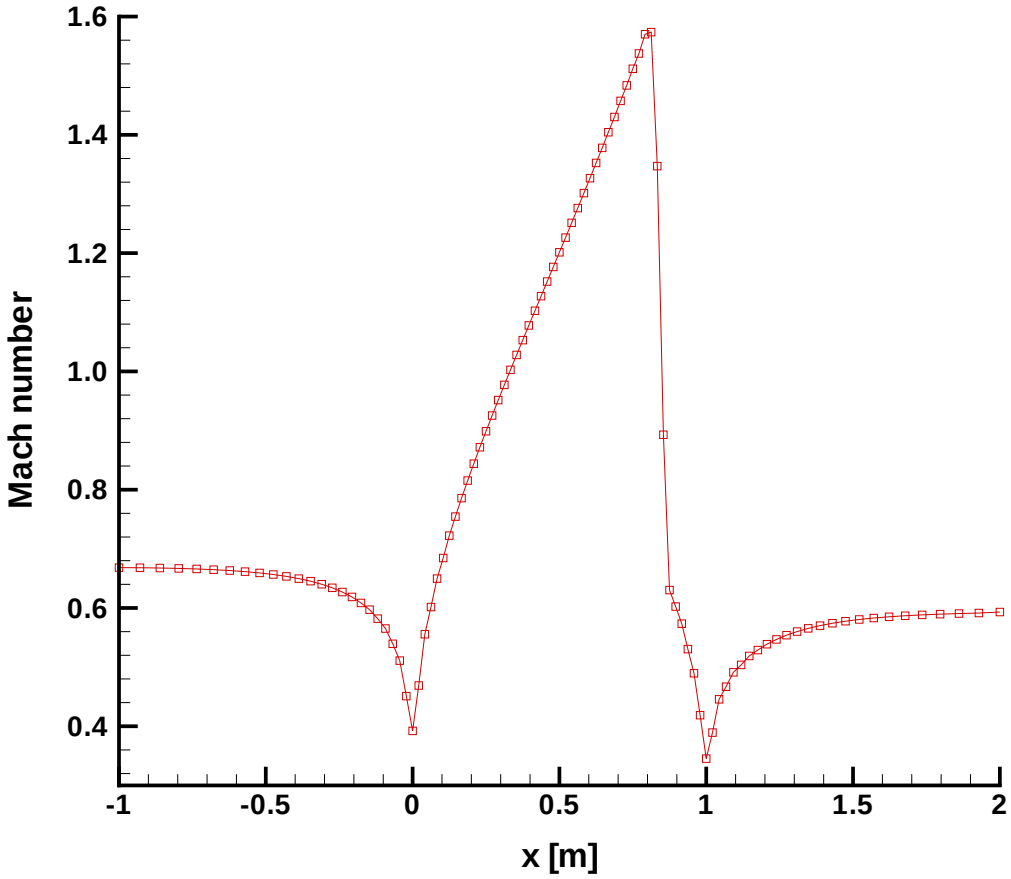


Figure 6.4: Channel flow; Mach number distribution along the wall.

6.2 Injection Driven Flow

The second test case represents an injection driven flow in a square chamber. The flow is inviscid and subsonic. The flow is injected from the bottom side of the grid, which is shown in Fig. 6.5. The left side of the domain is wall, the upper side is symmetry line, and the right side represents outflow boundary. The data of the test case is provided in `calcs/verification/onerac0/rocflo`. Since the analytical solution is known [6], the configuration can be used to verify the flow solver.

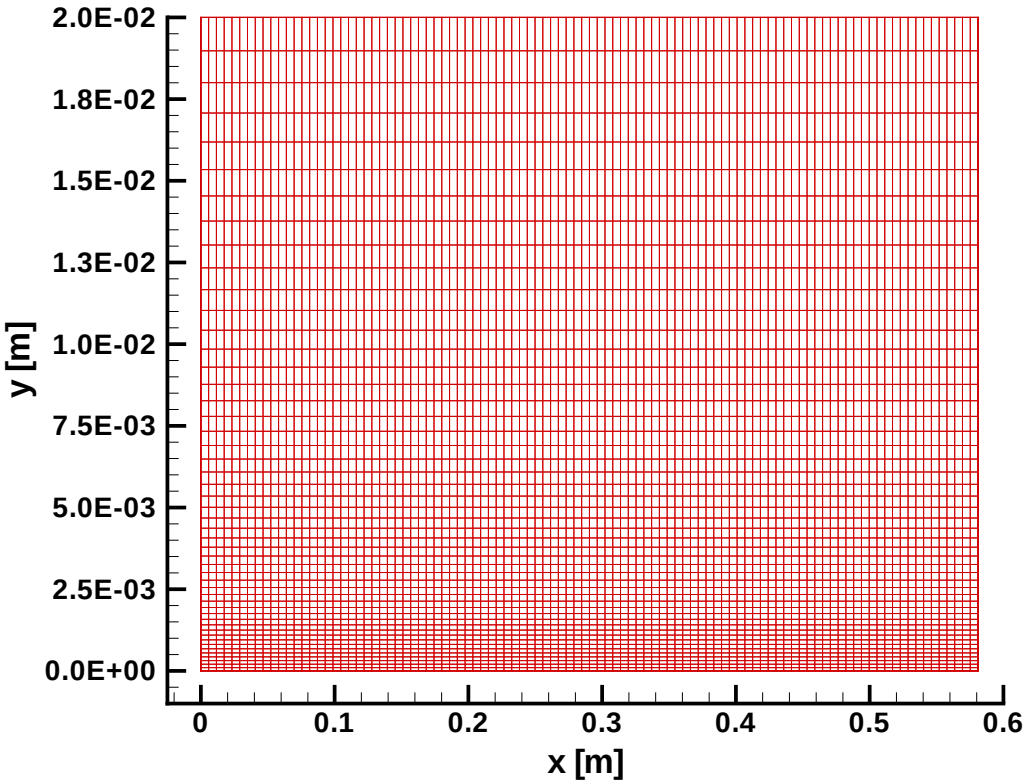


Figure 6.5: Injection flow; grid with 100x50x2 cells.

The initial solution can be generated by:

```
% rfloprep c0_100_50 1 0 0
```

which creates the file `c0_100_50.sola_000000`. The flow solver is started using:

```
% rflomp c0_100_50 2
```

The residual drops below 10^{-5} after 7520 iterations and the solver stops. The directory contains now the solution file `c0_100_50.sola_007520` with the steady state flow field. It also contains the convergence history in the file `c0_100_50.con`. The convergence history is plotted in Fig. 6.6.

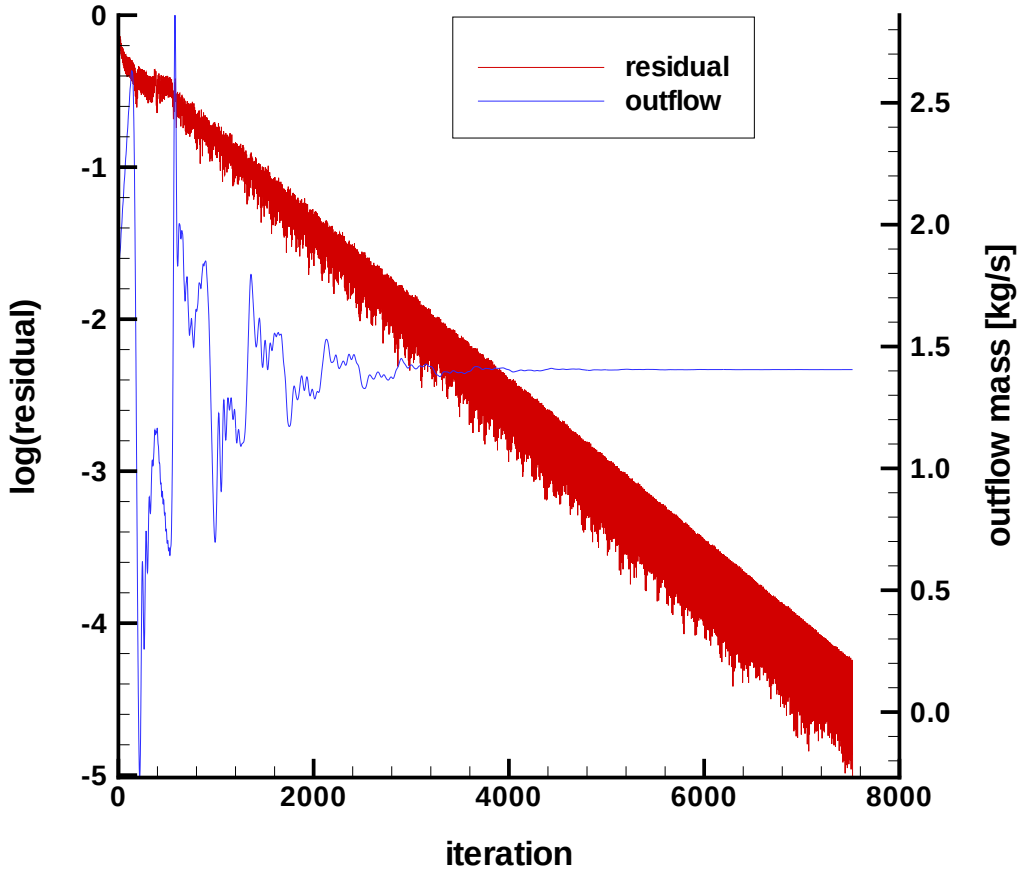


Figure 6.6: Injection flow; convergence history.

In order to visualize the solution, the post-processing utility has to be run:

```
% rflopost c0_100_50 2 1 7520 2
```

This generates the file `c0_100_50.plt`. The plot file can be directly visualized by Tecplot. Contours of the Mach number are shown in Fig. 6.7. Streamlines and contours of static pressure are depicted in Fig. 6.8.

The result of Rocflo's simulation can be compared to the analytical solution of Proudman and Culick [6]. This can be done using `rocvav` (see Section 5.8):

```
% ln -s c0_100_50.grda         c0_100_50_s1.grda
% ln -s c0_100_50.sola_007520 c0_100_50_s1.sola_000000
% rocvav c0_100_50 0
```

This results in the following output by `rocvav`:

```
  ******************************************************
  *                                                    *
  *      ROCVAV: Verification And Validation Tool      *
  *      ========================================      *
  *                                                    *
  *          Version: 1.2.0-0, Date: 08/14/02          *
  *     Copyright (c) by the University of Illinois    *
  *                                                    *
  ******************************************************

  - region 00001

 RocVaV passed the test for Comparison     1
  Percentage of Normalized L2 Norm=   8.2642666E-02
  Percentage of Normalized L8 Norm =  8.2642666E-02

 RocVaV passed the test for Comparison     2
  Percentage of Normalized L2 Norm=   8.2888011E-02
  Percentage of Normalized L8 Norm =  8.4347953E-02

 RocVaV passed the test for Comparison     3
  Percentage of Normalized L2 Norm=   4.6662325E-01
  Percentage of Normalized L8 Norm =  5.3007521E-01

 RocVaV passed the test for Comparison     4
  Percentage of Normalized L2 Norm=   3.0436462E-02
  Percentage of Normalized L8 Norm =  7.5802204E-02

RocVAV Finished.
```

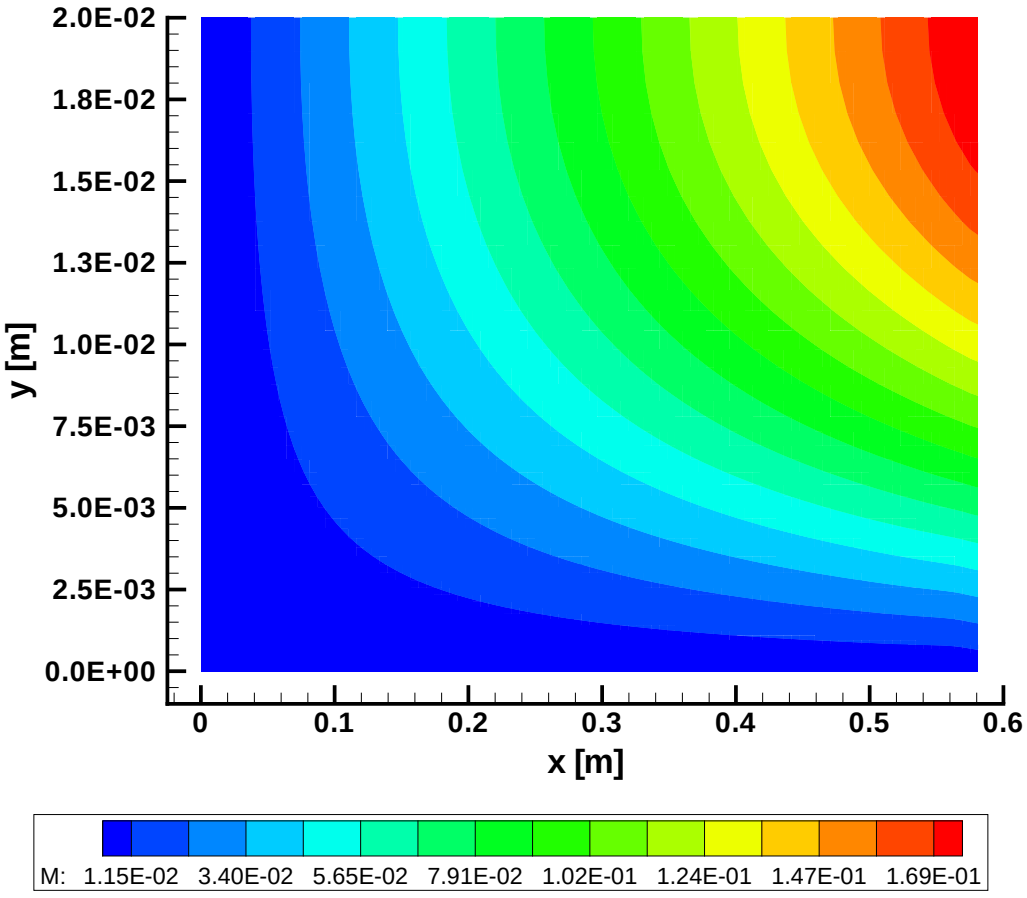


Figure 6.7: Injection flow; Mach contours.

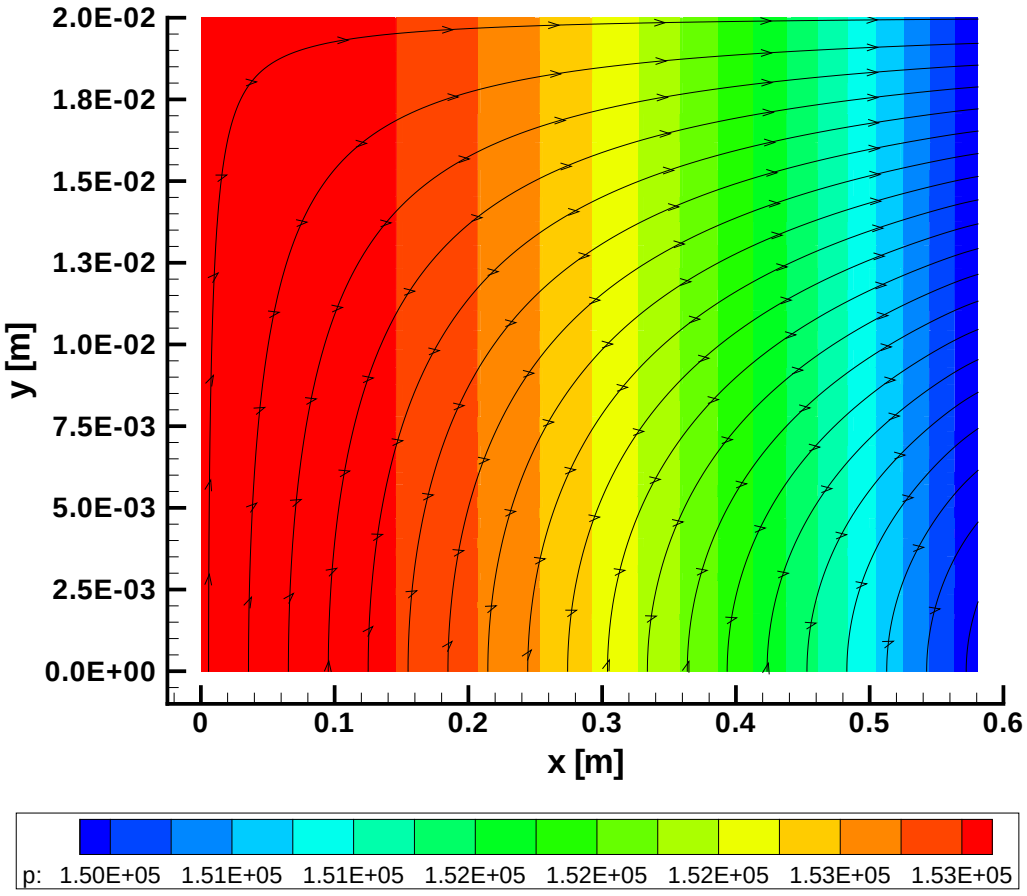


Figure 6.8: Injection flow; contours of static pressure and streamlines.

Bibliography

[1] Blazek, J.: *Computational Fluid Dynamics: Principles and Applications*. Elsevier Science Ltd., 2001.

[2] Thomas, P.D.; Lombard, C.K.: *Geometric Conservation Law and Its Application to Flow Computations on Moving Grids*. AIAA Journal, 17 (1979), pp. 1030-1037.

[3] Jameson, A.; Schmidt, W.; Turkel, E.: *Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes*. AIAA Paper 81-1259, 1981.

[4] Turkel, E.; Swanson, R.C.; Vatsa, V.N.; White, J.A.: Multigrid for Hypersonic Viscous Two- and Three-Dimensional Flows. AIAA Paper 91-1572, 1991.

[5] Jameson, A.; Baker, T.J.: *Solution of the Euler Equations for Complex Configurations*. AIAA Paper 83-1929, 1983.

[6] Ciucci, A.; Iafrati, A.; Schettino, A.: *Numerical Analysis of Pressure Oscillations in a Duct - Test Case C0*. CIRA Technical Report TR-96-102, Centro Italiano Ricerche Aerospaziali, Sept. 1996.