User-manual for COOLFluiD Ideal Magnetohydrodynamics Solver (Finite Volume) (version 2011.01)

Mehmet Sarp Yalim, MehmetSarp.Yalim@wis.kuleuven.be
KU Leuven, Center for Plasma Astrophysics

1 General

As far as the simulation of ideal magnetohydrodynamics (MHD) is concerned (see [11, 12, 13, 14] for technical details), COOLFluiD [3, 4, 5, 6, 7] offers:

- 2D and 3D Finite Volume (FV) solver with/without $\vec{B}_0 + \vec{B}_1$ splitting technique [10] used for interaction of solar wind with planetary magnetosphere simulations [12, 14] on unstructured hybrid grids using modified Rusanov with tuning of dissipation and Roe's schemes with/without limited linear least squares reconstruction method to obtain second order spatial accuracy. Note that the dissipative part of the Rusanov scheme, which has originally first order spatial accuracy, can be reduced down to a certain extent to approach second order spatial accuracy without using linear least squares reconstruction method [13].
- Powell's source term [8] and hyperbolic divergence cleaning with hyperbolic Lagrange multiplier [1] are the two techniques used to satisfy the $\nabla \cdot \vec{B} = 0$ constraint.
- Steady and unsteady simulations can be made with satisfying the $\nabla \cdot \vec{B} = 0$ constraint up to machine accuracy at convergence for steady time marching simulations and at each subiteration for unsteady simulations. For this purpose, backward Euler method is utilized for steady and Crank-Nicholson and backward differentiation formula (BDF) are utilized for unsteady simulations. Note that the implicit time integration schemes are much more advantageous than the explicit time integration schemes especially when used with hyperbolic divergence cleaning technique with hyperbolic Lagrange multiplier [13]. Therefore, only configurable options related to the implicit time integration schemes will be covered in the rest of the manual.

2 Input File

2.1 Installation instructions

The installation instructions for COOLFluiD are available online at

`http://coolfluidsrv.vki.ac.be/trac/coolfluid`

For the moment, interested users must contact the project administrator (`lani@vki.ac.be`) to get a username and therefore access the source code. The latter is currently undergoing a major restructuring and will be released open source in 2012.

2.2 How to run

The command line to run COOLFluiD is

```
mpirun -np N ./coolfluid-solver --scase ./myfile.CFcase
```

from inside the testcase directory. The parameter N must be replaced by the number of processors. The format of the input file (called `myfile.CFcase` in our example) is described here after. For a serial run (N=1) the user can also use:

```
./coolfluid-solver --scase ./myfile.CFcase
```

WATCH OUT: *In order to be able to run successfully, a soft link to (or copy of) the* `coolfluid-solver` *executable and file* `coolfluid-solver.xml` *(providing useful info on the path to the COOLFluiD shared libraries) must be present in the working directory.*

2.3 Configuration file description

The format of the input file (with extension .CFcase) consists of lines in the form `KEY = VALUE`:

```
Simulator.OptionA = Value1                  # use Value1 as value for OptionA
Simulator.Value1.OptionB = Value2           # use Value2 as value for OptionB
Simulator.Value1.Value2.OptionC = Value3 # use Value3 as value for OptionC
Simulator.Value1.Value2.OptionD = Value4 # use Value4 as value for OptionD
```

where, in each line, the whole LHS is the keyword and the RHS is the value. The latter, depending on the actual types defined in the code for each configurable parameter, can be:

- an alpha-numerical string
- an integer
- a boolean (`true` or `false`)
- a floating point number
- an arbitrarily complex analytical function
- an array of all the previous.

The keyword is composed of literal strings separated by ".", corresponding to different *entities* (configurable objects or parameters) defined inside the actual code. The configuration is hierarchical and recursive from top to lowest level. **The order in which the options are declared in the file are irrelevant**. If needed, the value can be broken into different lines by using the continuation character (back slash) at the end of each line (note that the number of spaces at the end or before the line is irrelevant):

```
Simulator.Example.arrays = 4 4 10 \
                           10 4 \
                           4 3
```

Comments start with "#": they can occupy full lines or be placed at the end of the line.

2.4 Environment

```
CFEnv.ErrorOnUnusedConfig = true
```

If activated this option makes the simulation crash if there are spelling mistakes in the given options. This option must always be inactivated when starting from a Gambit file (see below).

```
Simulator.Modules.Libs = libCFmeshFileReader libMHD libFiniteVolume
    libFiniteVolumeMHD ...
```

gives a list of the COOLFluiD dynamic libraries needed for the present simulation. In the following description, each section will indicate the required libraries whenever applicable.

```
Simulator.Paths.WorkingDir = ./
Simulator.Paths.ResultsDir = ./RESULTS
```

define the paths to the working directory and to the directory where output files (convergence history, Tecplot files, CFmesh files) should be written.

2.5 Physical model

Required libs: libMHD.

```
Simulator.SubSystem.Default.PhysicalModelType = MHD2DProjection (or
    MHD3DProjection)
```

defines a generic 2D (or 3D) ideal MHD model that uses hyperbolic divergence cleaning method as the $\nabla \cdot \vec{B} = 0$ constraint satisfying technique.

```
Simulator.SubSystem.Default.PhysicalModelType = MHD2D (or MHD3D)
```

defines a generic 2D (or 3D) ideal MHD model that uses Powell's source term method as the $\nabla \cdot \vec{B} = 0$ constraint satisfying technique.

```
Simulator.SubSystem.MHD3DProjection(or MHD3D or MHD2DProjection or MHD2D).
    ConvTerm.gamma = 1.666666667
```

defines the ratio of specific heats for the plasma which we take most of the time to be equal to 5/3.

```
Simulator.SubSystem.MHD2DProjection(or MHD3DProjection).ConvTerm.refSpeed = 3.0
#Simulator.SubSystem.MHD2DProjection(or MHD3DProjection).ConvTerm.dissipCoeff =
    3.0
#Simulator.SubSystem.MHD2DProjection(or MHD3DProjection).ConvTerm.correctionType
    = Mixed
```

defines a constant reference speed for the hyperbolic divergence cleaning method with hyperbolic Lagrange multiplier. The two commented out lines should be uncommented while using the mixed hyperbolic/parabolic Lagrange multiplier [1] which we never had to use so far. The constant reference speed is chosen to be equal to the freestream flow speed or the maximum speed in the initial conditions for unsteady simulations where there is no freestream flow in the setup of the testcase (see OrszagTang vortex, Smooth Alfvén Wave and Rotor testcases [13]).

```
Simulator.SubSystem.SubSystemStatus.TimeStep = 5.0
```

defines the time step in an unsteady simulation.

2.5.1 $\vec{B}_0 + \vec{B}_1$ splitting method

For solar wind/planetary magnetosphere interaction simulations, the planetary intrinsic magnetic field is modelled as a dipole ($\vec{B}_0$).

```
Simulator.SubSystem.MHD3DProjection(or MHD2DProjection or MHD2D or MHD3D).
    ConvTerm.mX = 0.0
Simulator.SubSystem.MHD3DProjection(or MHD2DProjection or MHD2D or MHD3D).
    ConvTerm.mY = 0.0
Simulator.SubSystem.MHD3DProjection(or MHD3D).ConvTerm.mZ = -3000.0
```

define the magnetic dipole moment of the planetary magnetic field with the center of the dipole located at the origin.

```
Simulator.SubSystem.CellCenterFVM.Data.Centred.Flux =
    MHD3DProjectionConsLaxFriedTanaka (or MHD3DConsLaxFriedTanaka or
    MHD2DConsLaxFriedTanaka or MHD2DProjectionConsLaxFriedTanaka)
Simulator.SubSystem.CellCenterFVM.Data.Centred.MHD3DProjectionConsLaxFriedTanaka(
    or MHD3DConsLaxFriedTanaka or MHD2DConsLaxFriedTanaka or
    MHD2DProjectionConsLaxFriedTanaka).NameFluxFunction = Powell99 (or Tanaka94)
```

define the necessary additional lines when using the Rusanov scheme together with the above-mentioned type of problems. Note that in this type of problems Roe's scheme is not preferred as it was not observed to have the necessary robustness level. Therefore, we stick to the Rusanov scheme with tunable dissipation. Moreover, "Powell99" implementation made according to [9] for only 3D problems is preferred over "Tanaka94" implementation made according to [10] for both 2D and 3D problems as the latter implementation which involves matrix transformations is slower in comparison with the former.

2.6 Interactive file

Some parameters can be changed interactively during the simulation by editing a separate file where the full option setting (key and value) has to be present.

```
Simulator.SubSystem.InteractiveParamReader.FileName = ./out.inter
```

tells the path to the interactive file and

```
Simulator.SubSystem.InteractiveParamReader.readRate = 10
```

specifies how often the file should be read by the solver in order to update the corresponding interactive parameters.
WATCH OUT: *In a parallel run, this rate must be defined with a safe margin (depending on the speed of the iterative process), allowing the user to quickly edit, modify and close the file before the solver tries to read the file as well.*

For the present solver, the interactive parameters mainly utilized are the CFL number, diffusion reduction coefficient for the modified Rusanov scheme with tunable dissipation and the residual threshold value below which either the limiter is frozen or historical modification of limiter is applied as a remedy for convergence hampering for TVD schemes. The content of the interactive file in its most general form is given as follows:

```
Simulator.SubSystem.BwdEuler(or BDF2).Data.CFL.Interactive.CFL = 1000.0
Simulator.SubSystem.CellCenterFVM.Data.LinearLS3D(or LinearLS2D).limitRes = -4.0
Simulator.SubSystem.CellCenterFVM.Data.LinearLS2D(or LinearLS3D).gradientFactor =
    0.
```

```
Simulator.SubSystem.CellCenterFVM.Data.Centred.LaxFried.DiffCoeff = 0.3
Simulator.SubSystem.CellCenterFVM.Data.Centred.MHD3DProjectionConsLaxFriedTanaka(
    or MHD2DProjectionConsLaxFriedTanaka).DiffCoeff = 0.3
Simulator.SubSystem.CellCenterFVM.Data.Centred.MHD3DConsLaxFriedTanaka(or
    MHD2DConsLaxFriedTanaka).DiffCoeff = 0.3
```

2.7 Output format

Required libs: libCFmeshFileWriter, libTecplotWriter.

```
Simulator.SubSystem.OutputFormat      = Tecplot CFmesh
```

defines list of requested output files (only `Tecplot` and `CFmesh` are supported for ideal MHD simulations).

```
# in parallel runs, a file out-P0.CFmesh is written
Simulator.SubSystem.CFmesh.FileName = out.CFmesh

# every how many iterations the file is saved
Simulator.SubSystem.CFmesh.SaveRate = 100

# append iteration number (or time for unsteady simulations) to the file name
Simulator.SubSystem.CFmesh.AppendIter = true
Simulator.SubSystem.CFmesh.AppendTime = true
```

specify the settings for the CFmesh file format (the internal format of COOLFluiD, including both mesh and solution). Only one CFmesh file is written even in a parallel simulation.

```
# in parallel runs, one file per processor out-P*.plt is written
Simulator.SubSystem.Tecplot.FileName        = out.plt

# write also components of the dipole magnetic field (B0) and total magnetic
    field (B0+B1), total energy based on the total magnetic field, pressure and
    div.B error value
Simulator.SubSystem.Tecplot.Data.printExtraValues = true

# update variables set
Simulator.SubSystem.Tecplot.Data.updateVar = Cons

# every how many iterations the file is saved
Simulator.SubSystem.Tecplot.SaveRate = 100

# append iteration number (or time for unsteady simulations) to the file name
Simulator.SubSystem.Tecplot.AppendIter = false
Simulator.SubSystem.Tecplot.AppendTime = false
```

specify the settings for the Tecplot file format. One file per processor will be written.

2.8 Stop condition

The simulation can be stopped by prescribing a maximum number of steps:

```
Simulator.SubSystem.StopCondition = MaxNumberSteps
Simulator.SubSystem.MaxNumberSteps.nbSteps = 2
```

or by looking at the norm of the residual:

```
Simulator.SubSystem.StopCondition          = Norm
Simulator.SubSystem.Norm.valueNorm         = -6.0
```

or by specifying the maximum time in an unsteady simulation:

```
Simulator.SubSystem.StopCondition          = MaxTime
Simulator.SubSystem.MaxTime.maxTime        = 2146.0
```

2.9 Mesh reader

Required libs: libCFmeshFileReader.

```
Simulator.SubSystem.Default.listTRS = InnerCells SlipWall SuperInlet SuperOutlet
```

specifies the list of all Topological Region Sets (TRS), i.e. the boundary patches as defined in the mesh file. `InnerCells` does not need to be included in the list.

```
Simulator.SubSystem.MeshCreator = CFmeshFileReader
Simulator.SubSystem.CFmeshFileReader.Data.FileName = ./input.CFmesh
```

specify the reading from a file called `input.CFmesh` in CFmesh format.

```
Simulator.SubSystem.CFmeshFileReader.ParReadCFmesh.ParCFmeshFileReader.
    NbOverlapLayers = 2
```

This option is obsolete, but kept for compatibility with older versions of the code. It specifies the number of overlap layers in parallel computations. This value should be 2 for second order calculations, but it is now automatically calculated.

```
Simulator.SubSystem.CFmeshFileReader.Data.builderName = FVMCC
Simulator.SubSystem.CFmeshFileReader.Data.polyTypeName = Lagrange
```

2.9.1 Reading from Gambit files

Required libs: libGambit2CFmesh.

If the mesh file is not yet in CFmesh format and it's coming from the ANSYS Gambit mesh generator, the following settings must be defined:

```
Simulator.SubSystem.CFmeshFileReader.convertFrom = Gambit2CFmesh
Simulator.SubSystem.CFmeshFileReader.Gambit2CFmesh.Discontinuous = true
Simulator.SubSystem.CFmeshFileReader.Gambit2CFmesh.SolutionOrder = P0
```

In this case the solver expects a file called `input.neu` placed inside the working directory. **WATCH OUT:** *all Gambit settings must be commented out when restarting from a previous CFmesh solution (see Restart option).*

2.9.2 Reading from THOR files

Required libs: libTHOR2CFmesh.

If the mesh file is not yet in CFmesh format and it's coming from the THOR code mesh format, the following settings must be defined:

```
Simulator.SubSystem.CFmeshFileReader.THOR2CFmesh.Discontinuous = true
Simulator.SubSystem.CFmeshFileReader.THOR2CFmesh.SolutionOrder = P0
Simulator.SubSystem.CFmeshFileReader.convertFrom = THOR2CFmesh
```

In this case the solver expects a mesh file called `input.thor` and a super patch file called `input.SP` placed inside the working directory.
WATCH OUT: *all THOR settings must be commented out when restarting from a previous CFmesh solution (see Restart option).*

2.10 Convergence method

Required libs: libBackwardEuler libBackwardEulerMHD libNewtonMethod.

We consider here both the steady implicit time stepping case corresponding to a first-order accurate backward Euler integration and the unsteady implicit time stepping case corresponding to the second-order accurate Crank-Nicholson and BDF.

```
Simulator.SubSystem.ConvergenceMethod = BwdEuler (or BDF2)

#the filename into which the convergence history is to be written
Simulator.SubSystem.BwdEuler(or BDF2).ConvergenceFile = ./convergence.plt

Simulator.SubSystem.BwdEuler.UpdateSol = UpdateSolMHD
Simulator.SubSystem.BwdEuler.UpdateSolMHD.pressureCorrectionValue =
    0.000000000001
```

define a correction value for negative pressure occurrences especially encountered during the solar wind/planetary magnetosphere simulations with sufficiently harsh initial conditions and planetary dipole field in the vicinity of the planet when started from an initial uniform solution. Thus, the simulation does not blow up due to the increase in the number of negative pressure occurrences and most of the time the negative pressure values are eliminated as the simulation continues.

```
# this value must be > 1 only for unsteady simulations where it will represent
    the number of subiterations to be made at each timestep
Simulator.SubSystem.BDF2.Data.MaxSteps = 1

Simulator.SubSystem.BDF2.Data.L2.MonitoredVarID = 8
```

indicates the ID of the variable to be monitored for convergence (see Stop condition).

```
Simulator.SubSystem.BDF2.Data.L2.ComputedVarID = 8
```

indicates the ID of the variable whose norm will be computed and written to screen. If this line is commented out, residuals for all variables will be computed.

```
Simulator.SubSystem.BDF2.Data.Norm = -3.
Simulator.SubSystem.BDF2.Data.PrintHistory = true
```

The CFL parameter which controls the stability of the calculation can be specified in three ways: Interactively, as a constant value or with a user-defined function.

2.10.1 Interactive CFL

```
Simulator.SubSystem.BDF2(or BwdEuler).Data.CFL.ComputeCFL = Interactive
```

declares the CFL interactive and its value will be read from the interactive file (`out.inter` in our example). In this case the line

```
Simulator.SubSystem.BDF2(or BwdEuler).Data.CFL.Interactive.CFL = 10.0
```

must be present and, if needed, modified in the interactive file.

2.10.2 Constant CFL Value

```
Simulator.SubSystem.BwdEuler(or BDF2).Data.CFL.Value = 10.0
```

declares the CFL as a constant value which will remain unchanged during the simulation.

2.10.3 Function CFL

```
Simulator.SubSystem.BwdEuler(or BDF2).Data.CFL.ComputeCFL = Function
Simulator.SubSystem.BwdEuler(or BDF2).Data.CFL.Function.Def = \
  if(i<1000,1.0,if(i<2000.,1.01*cfl, min(1200.,1.05*cfl)))
```

In order to automatize the iterative process, a function specifying an arbitrarily complex CFL law can be provided. The variable that can appear in this expression are: `i` (iteration number), `cfl` (previous CFL value), `r` (current residual), `ri` (initial residual), `rl` (last residual), `rmax` (maximum residual).

WATCH OUT: *no spaces are allowed within the expression and the supported operators and mathematical functions are indicated in [2].*

2.11 Linear system solver

Required libs: libPetsc.

We include here only the settings corresponding to the PETSC linear system solver library, even though Trilinos is also interfaced within COOLFluiD.

```
Simulator.SubSystem.LinearSystemSolver = PETSC
Simulator.SubSystem.LSSNames = BwdEulerLSS (or CrankNicholsonLSS)
Simulator.SubSystem.BwdEulerLSS(or CrankNicholsonLSS).Data.PCType = PCASM (or
    PCILU)
Simulator.SubSystem.BwdEulerLSS(or CrankNicholsonLSS).Data.KSPType = KSPGMRES
Simulator.SubSystem.BwdEulerLSS(or CrankNicholsonLSS).Data.MatOrderingType =
    MATORDERINGRCM
```

specify the basic settings for a GMRES solver combined with a parallel Additive Schwartz preconditioner or a serial ILU preconditioner.

```
Simulator.SubSystem.BwdEulerLSS(or CrankNicholsonLSS).Data.MaxIter = 1000
```

defines the maximum allowed number of GMRES iterations: should be typically kept $<= 1000$.

```
Simulator.SubSystem.BwdEulerLSS(or CrankNicholsonLSS).Data.RelativeTolerance = 1e
    -4
```

defines the relative tolerance for the GMRES solver: $1e-4$ or $1e-3$ are recommended values, since with higher values convergence can be very slow and requiring many more GMRES iterations per time step.

2.12 Space method

Required libs: libFiniteVolume, libMHD, libFiniteVolumeMHD.

The following standard settings for the implicit FV solver should not be changed except the save rate which should be the same as the save rate of the Tecplot solution data file in order to be able to write the $\nabla \cdot \vec{B}$ error values correctly in the data file:

```
Simulator.SubSystem.SpaceMethod = CellCenterFVM
Simulator.SubSystem.CellCenterFVM.ComputeRHS = NumJacobMHD
Simulator.SubSystem.CellCenterFVM.NumJacobMHD.SaveRate = 100
Simulator.SubSystem.CellCenterFVM.ComputeTimeRHS = StdTimeRhs (or BDF2TimeRhs
    when using BDF2)
Simulator.SubSystem.CellCenterFVM.BDF2TimeRhs.zeroDiagValue = 0 0 0 0 0 0 0 0 1
```

Moreover, with the last line, we make the time derivative of the last equation in the modified ideal MHD system due to the hyperbolic divergence cleaning vanish. Hence, we apply a pure Newton iteration on the last equation which reduced to the $\nabla \cdot \vec{B} = 0$ constraint which is linear in $\vec{B}$ even after discretization. Thus, we assure convergence of the constraint upto machine accuracy even at each subiteration when using BDF2 [13].

In order to restart from a previous CFmesh file (with saved solution), the following option must be set to true, otherwise must be commented out.

```
Simulator.SubSystem.CellCenterFVM.Restart = true
```

2.13 Numerical schemes for computing convective fluxes in ideal MHD

The user must choose one of the following (Rusanov scheme with tunable dissipation is recommended for most cases):

```
Simulator.SubSystem.CellCenterFVM.Data.FluxSplitter = Centred
```

defines the option for Rusanov scheme. Specific cases that involve additional options were explained in detail earlier.

```
Simulator.SubSystem.CellCenterFVM.Data.FluxSplitter = Roe
Simulator.SubSystem.CellCenterFVM.Data.Roe.Flux = MHD2DProjectionConsRoe (or
    MHD2DConsRoe or MHD3DProjectionConsRoe or MHD3DConsRoe)
```

define the option for Roe's scheme.

2.13.1 Variable sets for ideal MHD

The following settings do not change for the present solver:

```
Simulator.SubSystem.CellCenterFVM.Data.UpdateVar  = Cons
Simulator.SubSystem.CellCenterFVM.Data.SolutionVar = Cons
Simulator.SubSystem.CellCenterFVM.Data.LinearVar   = Cons
```

2.13.2 Source terms

```
Simulator.SubSystem.CellCenterFVM.Data.hasSourceTerm = true
Simulator.SubSystem.CellCenterFVM.Data.SourceTerm = MHDConsACAST
```

should always be present in simulations involving hyperbolic divergence cleaning technique.

```
Simulator.SubSystem.CellCenterFVM.Data.hasSourceTerm = true
Simulator.SubSystem.CellCenterFVM.Data.SourceTerm = MHD2DPowellST (or
    MHD3DPowellST)
```

should always be present in simulations involving Powell's source term technique.

2.13.3 Polynomial reconstruction

The following option should be kept uncommented in case of utilization of a constant reconstruction:

```
Simulator.SubSystem.CellCenterFVM.Data.PolyRec = Constant
```

The following options should be kept uncommented in case of utilization of a TVD reconstruction:

```
Simulator.SubSystem.CellCenterFVM.SetupCom = LeastSquareP1Setup
Simulator.SubSystem.CellCenterFVM.SetupNames = Setup1
Simulator.SubSystem.CellCenterFVM.Setup1.stencil = FaceVertexPlusGhost
Simulator.SubSystem.CellCenterFVM.UnSetupCom = LeastSquareP1UnSetup
Simulator.SubSystem.CellCenterFVM.UnSetupNames = UnSetup1
Simulator.SubSystem.CellCenterFVM.Data.IntegratorQuadrature = GaussLegendre
Simulator.SubSystem.CellCenterFVM.Data.IntegratorOrder = P1
Simulator.SubSystem.CellCenterFVM.Data.PolyRec = LinearLS2D (or LinearLS3D)
Simulator.SubSystem.CellCenterFVM.Data.Limiter = Venktn2D (or Venktn3D or
    BarthJesp2D or BarthJesp3D)
Simulator.SubSystem.CellCenterFVM.Data.Venktn2D(or Venktn3D).coeffEps = 1.0
Simulator.SubSystem.CellCenterFVM.Data.Venktn2D(or Venktn3D).useFullStencil =
    true
# set true the following for backward compatibility, but false should behave
    better
Simulator.SubSystem.CellCenterFVM.Data.Venktn2D.useNodalExtrapolationStencil =
    false
Simulator.SubSystem.CellCenterFVM.Data.Venktn2D(or Venktn3D).length = 1.0
Simulator.SubSystem.CellCenterFVM.Data.LinearLS2D(or LinearLS3D).freezeLimiter =
    true (if commented out, historical modification of limiter will be executed
    for a residual below limitRes (see below))
```

The value of the following factor determines if the simulation is of first, second or in-between order:

```
# 0 <= gradientFactor <= 1, with  0. (first order), 1. (second order)
Simulator.SubSystem.CellCenterFVM.Data.LinearLS2D(or LinearLS3D).gradientFactor =
     0.
```

This is an interactive parameter that can be placed into the interactive file. Another interactive parameter important for second order computations is

```
Simulator.SubSystem.CellCenterFVM.Data.LinearLS2D(or LinearLS3D).limitRes = -4.0
```

This corresponds to the minimum residual at which the freezing of the flux limiter should be applied for flows exhibiting discontinuities. In practice, `limitRes` can be kept at $-4.$ till

when the simulation reaches a limit cycle and then can be increased to 8. in order to exit the cycle. This cure is not always effective and it often depends on the moment when `limitRes` is increased.

WATCH OUT: *Before restarting a simulation from a second order solution, if the limiter has not been explicitly saved in the CFmesh file (see below),* `limitRes` *has to be set back to* -4.

In order to save the limiter in second order calculations (when `gradientFactor = 1.`), the following options must be added to the CFcase **before** starting the computation:

```
Simulator.SubSystem.CFmesh.Data.ExtraStateVarNames = limiter

# the following must be the total number of equations
Simulator.SubSystem.CFmesh.Data.ExtraStateVarStrides = 8 (for Powell's source
    term) and 9 (for hyperbolic divergence cleaning)
```

Finally, in order to restart from a file in which the limiter **has been already saved**, the following line should be included in the CFcase file:

```
Simulator.SubSystem.CFmeshFileReader.Data.ExtraStateVarNames = InitLimiter

Simulator.SubSystem.CellCenterFVM.Data.NodalExtrapolator = DistanceBased
Simulator.SubSystem.CellCenterFVM.Data.DistanceBased.TRSPriorityList = SuperInlet
     SlipWall SuperOutlet
```

defines the priority list of the TRSs in case of intersection between the two TRSs.

2.13.4 Initial field

The initial field conditions are typically prescribed on the full domain with a set of user-defined analytical functions depending on the position vector (x,y,z) [2], one for each of the update variables (`Cons` in our current example).

The following settings are always required:

```
# "InitState" is the name of the object implementing an initial state
Simulator.SubSystem.CellCenterFVM.InitComds = InitState

# "InField" is a user-defined alias that will be used to configure InitState
Simulator.SubSystem.CellCenterFVM.InitNames = InField

# from now on, only "InField" is used for the initialization settings
# "InnerFaces" is the boundary patch (TRS) on which InField is active
Simulator.SubSystem.CellCenterFVM.InField.applyTRS = InnerFaces
```

The following settings define the analytical functions.

```
# independent variables (use "x y z" in 3D)
Simulator.SubSystem.CellCenterFVM.InField.Vars = x y

# arbitrarily complex function definitions (one per update variable, 8 or 9 in
    this case)
# no space allowed within a single function
# NOTE: this is just an illustrative example with no physical sense!
Simulator.SubSystem.CellCenterFVM.InField.Def = \
 1. 0. 0. if(sqrt(x^2+y^2)<1.,0.00002854,0.00002854/2.) \
 0. if(sqrt(x^2+y^2)<1.,0.00000866,0.00000866/2.) 0. 0. (0. in case of 9
     equations)
```

If analytical expressions are particularly complex, the user can use a more advanced 2-step initializer. The previous example can be simplified as:

```
# "InitStateAddVar" is used instead of "InitState"
Simulator.SubSystem.CellCenterFVM.InitComds = InitStateAddVar
Simulator.SubSystem.CellCenterFVM.InitNames = InField
Simulator.SubSystem.CellCenterFVM.InField.applyTRS = InnerFaces

Simulator.SubSystem.CellCenterFVM.InField.InitVars = x y
Simulator.SubSystem.CellCenterFVM.InField.InitDef = sqrt(x^2+y^2)

# here "rad" is a new user-defined variable that can be used
# to simplify the final expressions
Simulator.SubSystem.CellCenterFVM.InField.Vars = x y rad
Simulator.SubSystem.CellCenterFVM.InField.Def = \
 1. 0. 0. if(rad<1.,0.00002854,0.00002854/2.) \
 0. if(rad<1.,0.00000866,0.00000866/2.) 0. 0. (0. in case of 9 equations)
```

IMPORTANT NOTE: While specifying the initial conditions for steady simulations, the magnetic field is set to zero inside the whole domain. The magnetic field is frozen-in the ideal MHD plasma flow at the inlet coming inside the domain and therefore spreads into the domain with the flow. This practice is observed to be robust even for the most challenging testcases tried so far.

2.13.5 Boundary conditions

Boundary condition fields will be applied also during initialization on the corresponding boundary TRS, in such a way that *ghost states* (dummy cell centers that lie outside the computational domain) are set consistently before starting computing numerical fluxes.
The following example shows how to specify a full set of boundary conditions (three in this case, but real settings will obviously depend on the mesh in use).

```
# list of the names of the objects defining each boundary condition
Simulator.SubSystem.CellCenterFVM.BcComds = \
      MirrorMHD3DProjectionTanakaPFixFVMCC \
      SuperInletFVMCC \
      SuperOutletMHD3DProjectionFVMCC

# list of aliases that the user must define for configuring each BC
Simulator.SubSystem.CellCenterFVM.BcNames = Wall Inlet Outlet
```

- Ionosphere/magnetosphere boundary condition [12, 14]

  ```
  # apply MirrorMHD3DProjectionTanakaPFixFVMCC to the SlipWall
  # (TRS name coming from the initial mesh file)
  Simulator.SubSystem.CellCenterFVM.Wall.applyTRS = SlipWall

  # imposed density and pressure
  Simulator.SubSystem.CellCenterFVM.Wall.rhoFixed = 1.0
  Simulator.SubSystem.CellCenterFVM.Wall.pFixed = 8.0 (This value should be 8
      times the freestream solar wind plasma pressure value which is equal to
      1.0 in this example.)
  ```

  This boundary condition is based on [9].

- Superfast inlet boundary condition [11, 12, 13, 14]

```
# apply SuperInletFVMCC to the SuperInlet TRS
Simulator.SubSystem.CellCenterFVM.Inlet.applyTRS = SuperInlet
Simulator.SubSystem.CellCenterFVM.Inlet.Vars = x y z
Simulator.SubSystem.CellCenterFVM.Inlet.Def =  1.05100 \
                                              -3.99981 \
                                              0.283107 \
                                              -0.0381893 \
                                              0.403907 \
                                              0.399628 \
                                              -0.489089 \
                                              8.09741 \
                                              (0.0 in case of 9 equations)
```

- Superfast outlet boundary condition [11, 12, 13, 14]

```
Simulator.SubSystem.CellCenterFVM.Outlet.applyTRS = SuperOutlet
Simulator.SubSystem.CellCenterFVM.Outlet.refPhi = 0.0
```

  where the scalar potential function, ϕ, value is kept constant in the ghost cells typically equal to 0 in all the testcases tried so far.

References

[1] A. Dedner, F. Kemm, D. Kröner, C. D. Munz, T. Schnitzer and M. Wesenberg, *Hyperbolic Divergence Cleaning for the MHD Equations*, Journal of Computational Physics, Vol. 175, pp. 645-673, 2002, doi:10.1006/jcph.2001.6961.

[2] J. Nieminen and J. Yliluoma, *Function Parser for C++*, http://warp.povusers.org/FunctionParser/, 2009.

[3] A. Lani, T. Quintino, D. Kimpe and H. Deconinck, *The COOLFluiD Framework - Design Solutions for High-Performance Object Oriented Scientific Computing Software*, International Conference Computational Science 2005, Atlanta (GA), LNCS 3514, Vol.1, pp. 281-286, Springer-Verlag, 2005.

[4] A. Lani, T. Quintino, D. Kimpe, H. Deconinck, S. Vandewalle and S. Poedts, *Reusable Object-Oriented Solutions for Numerical Simulation of PDEs in a High Performance Environment*, Scientific Programming. ISSN 1058-9244, Vol. 14, N. 2, pp. 111-139, IOS Press, 2006.

[5] A. Lani, *An Object Oriented and High Performance Platform for Aerothermodynamics Simulation*, Ph.D. thesis,von Karman Institute, Rhode-Saint-Genèse, Belgium, 2008.

[6] T. L. Quintino. *A Component Environment for High-Performance Scientific Computing. Design and Implementation*, Ph.D. thesis,von Karman Institute, Rhode-Saint-Genèse, Belgium, 2008.

[7] T. Wuilbaut, *Algorithmic Developments for a Multiphysics Framework*, Ph.D. thesis,von Karman Institute, Rhode-Saint-Genèse, Belgium, 2008.

[8] K. G. Powell, P. L. Roe, R. S. Myong, T. I. Gombosi and D. L. De Zeeuw, *An Upwind Scheme for Magnetohydrodynamics*, 12^{th} AIAA Computational Fluid Dynamics Conference, p. 661-674, Am. Inst. of Aeron. and Astron., San Diego, Calif., 1995.

[9] K. G. Powell, P. L. Roe, T. J. Linde, T. I. Gombosi and D. L. De Zeeuw, *A Solution-Adaptive Upwind Scheme for Ideal Magnetohydrodynamics*, Journal of Computational Physics, Vol. 154, pp. 284-309, 1999, doi:10.1006/jcph.1999.6299.

[10] T. Tanaka, *Finite Volume TVD Scheme on an Unstructured Grid System for Three-Dimensional MHD Simulation of Inhomogeneous Systems Including Strong Background Potential Fields*, Journal of Computational Physics, Vol. 111, pp. 381-389, 1994, doi:10.1006/jcph.1994.1071.

[11] M. S. Yalim, D. Vanden Abeele and A. Lani, *Simulation of Field-aligned Ideal MHD Flows around Perfectly Conducting Cylinders Using an Artificial Compressibility Approach*, Proceedings of the 11^{th} International Conference on Hyperbolic Problems held in Ecole Normale Supérieure, Lyon, France, July 17-21, 2006, pp. 1085-1092, Springer-Verlag, 2008.

[12] M. S. Yalim, *An Artificial Compressibility Analogy Approach for Compressible Ideal MHD: Application to Space Weather Simulation*, Ph.D. thesis,von Karman Institute, Rhode-Saint-Genèse, Belgium, 2008.

[13] M. S. Yalim, D. Vanden Abeele, A. Lani, T. Quintino and H. Deconinck, *A Finite Volume Implicit Time Integration Method for Solving the Equations of Ideal Magnetohydrodynamics for the Hyperbolic Divergence Cleaning Approach*, Journal of Computational Physics, Vol. 230, N. 15, pp. 6136-6154, 2011, doi:10.1016/j.jcp.2011.04.020.

[14] M. S. Yalim, J. Majewski, H. Deconinck and S. Poedts, *Adaptive Unstructured Grid Simulation of Interaction of Solar Wind with Planetary Magnetosphere Using an Implicit Finite Volume Method*, Journal of Geophysical Research Space Physics, submitted, 2011.