NIRVANA v3.8

astrophysical gas dynamics code

Udo Ziegler

DOCU
- appendix

April 14, 2014

Leibniz-Institut für Astrophysik Potsdam

Contents

1 MHD equations in cylindrical coordinates

$(x, y, z) \longrightarrow (z, R, \phi)$

$(m_z, m_R, m_\phi) = (\varrho v_z, \varrho v_R, R\varrho v_\phi)$

$\mathbf{E} = -\mathbf{v} \times \mathbf{B}$

$\nabla \cdot \mathbf{f} = \partial_z f_z + \frac{1}{R}\partial_R \left[R f_R\right] + \frac{1}{R}\partial_\phi f_\phi; \quad \mathbf{f} = f_z \mathbf{e}_z + f_R \mathbf{e}_R + f_\phi \mathbf{e}_\phi$

$$
\begin{aligned}
\partial_t \varrho &= -\nabla \cdot [\varrho \mathbf{v}] \\
\partial_t m_z &= -\nabla \cdot \left[m_z \mathbf{v} - \frac{1}{\mu} B_z \mathbf{B} \right] - \partial_z p_{\rm tot} \\
\partial_t m_R &= -\nabla \cdot \left[m_R \mathbf{v} - \frac{1}{\mu} B_R \mathbf{B} \right] - \frac{1}{R}\partial_R \left(R p_{\rm tot} \right) \\
&\quad + \frac{1}{R}\left(\varrho v_\phi^2 + p_{\rm tot} - \frac{1}{\mu} B_\phi^2 \right) \\
\partial_t m_\phi &= -\nabla \cdot \left[m_\phi \mathbf{v} - \frac{1}{\mu} R B_\phi \mathbf{B} \right] - \partial_\phi p_{\rm tot} \\
\partial_t e &= -\nabla \cdot \left[(e + p_{\rm tot}) \mathbf{v} - \frac{1}{\mu} (\mathbf{v} \cdot \mathbf{B}) \mathbf{B} \right] \\
B_z &= -\frac{1}{R}\partial_R (R E_\phi) + \frac{1}{R}\partial_\phi E_R \\
B_R &= -\frac{1}{R}\partial_\phi E_z + \partial_z E_\phi \\
B_\phi &= -\partial_z E_R + \partial_R E_z
\end{aligned}
$$

2 MHD equations in spherical coordinates

$(x, y, z) \longrightarrow (r, \theta, \phi)$

$(m_r, m_\theta, m_\phi) = (\varrho v_r, \varrho v_\theta, r \sin\theta \varrho v_\phi)$

$\mathbf{E} = -\mathbf{v} \times \mathbf{B}$

$\nabla \cdot \mathbf{f} = \frac{1}{r^2}\partial_r \left[r^2 f_r\right] + \frac{1}{r\sin\theta}\partial_\theta \left[\sin\theta f_\theta\right] + \frac{1}{r\sin\theta}\partial_\phi f_\phi; \quad \mathbf{f} = f_r \mathbf{e}_r + f_\theta \mathbf{e}_\theta + f_\phi \mathbf{e}_\phi$

$$
\begin{aligned}
\partial_t \varrho &= -\nabla \cdot [\varrho \mathbf{v}] \\
\partial_t m_r &= -\nabla \cdot \left[m_r \mathbf{v} - \frac{1}{\mu} B_r \mathbf{B} \right] - \frac{1}{r^2}\partial_r (r^2 p_{\rm tot}) \\
&\quad + \frac{1}{r}\left[\varrho \left(v_\theta^2 + v_\phi^2 \right) + 2p + \frac{1}{\mu} B_r^2 \right] \\
\partial_t m_\theta &= -\nabla \cdot \left[m_\theta \mathbf{v} - \frac{1}{\mu} B_\theta \mathbf{B} \right] - \frac{1}{r\sin\theta}\partial_\theta \left(\sin\theta p_{\rm tot} \right) \\
&\quad + \frac{1}{r}\left[\cot\theta \left(\varrho v_\phi^2 + p_{\rm tot} - \frac{1}{\mu} B_\phi^2 \right) - \varrho v_r v_\theta + \frac{1}{\mu} B_r B_\theta \right] \\
\partial_t m_\phi &= -\nabla \cdot \left[m_\phi \mathbf{v} - \frac{1}{\mu} r \sin\theta B_\phi \mathbf{B} \right] - \partial_\phi p_{\rm tot}
\end{aligned}
$$

$$
\begin{aligned}
\partial_t e &= -\nabla \cdot \left[(e + p_{\rm tot})\mathbf{v} - \frac{1}{\mu}(\mathbf{v}\cdot\mathbf{B})\mathbf{B} \right] \\
B_r &= -\frac{1}{r\sin\theta}\partial_\theta(\sin\theta E_\phi) + \frac{1}{r\sin\theta}\partial_\phi E_\theta \\
B_\theta &= -\frac{1}{r\sin\theta}\partial_\phi E_r + \frac{1}{r}\partial_r(rE_\phi) \\
B_\phi &= -\frac{1}{r}\partial_r(rE_\theta) + \frac{1}{r}\partial_\theta E_r
\end{aligned}
$$

3 List of NIRVANA modules/functions

void **APdiffusion**(GRD *g, double dt, double ****Fx, double ****Fy, double ****Fz)
: computation of ampipolar diffusion part of electric field and energy flux.

void **APdiffusionCoeff**(GRD *g, double ***APdiff)
: ambipolar diffusion coefficient.

double **computeTimestep**(GRD **gc)
: time-step control.

void **conduction**(GRD *g, double dt, double ****Fx, double ****Fy, double ****Fz)
: heat flux computation.

void **conductionCoeff**(GRD *g, double ***cond, double ***cond_perp)
: computation of thermal conductivity coefficient.

void **diffusion**(GRD *g, double dt, double ****Fx, double ****Fy, double ****Fz)
: computation of diffusive part of electric field and flux.

void **diffusionCoeff**(GRD *g, double ***diff)
: magnetic diffusion coefficient.

void **init**(void)
: basic initialisation.

nirvana.h
: header file for basic macro/type/function declarations.

nirvana.in
: user-interface for main parameter specification.

nirvana.c
: module containing NIRVANA main function
  int **main**(int argc, char *argv[])
  : main function.

void **solveCoriolisForce**(GRD **gc, double dt)
: Coriolis force term solve (in case of rotating frame).

double **solveDissipation**(GRD **gc, double dt, COM *cSYNC, COM *cFIXUP,
        void (*dissipation)(GRD *, double, double ****, double ****, double ****))
: dissipation terms solve based on stabilized Runge-Kutta-Legendre method.

void **solveHeatLoss**(GRD **gc, double dt)
: heatloss term solve.

solveGravity.c
: selfgravity solver module containing functions
  void **computeMulipoles**(GRD **gc)
  : multipole computation of mass distribution.
  void **solveGravity**(GRD **gc, COM *cSYNC, COM *cFIXUP)

: selfgravity solver main.
void **operatorGravity**(GRD **gc, int level, COM *cFIXUP, flag_t var)
: operator of Poisson equation.
void **sourceGravity**(GRD *g, double ***s)
: source term in Poisson equation.

solveMHD.c
: MHD solver module containing functions
void **updateRK**(GRD **gc, double dt, COM *cFIXUP)
: RK-stage update.
void **regularizeEF**(GRD **gc)
: electric field regularization at geometric axis.
void **solveMHD**(GRD **gc, double dt, COM *cSYNC, COM *ccFIXUP)
: MHD solver.

void **solveNIRVANA**(GRD **gc, COM *ccSYNC, COM *ccFIXUP, COM *cSYNC)
: main solver module.

void **sourceHeatLoss**(int dummy, double *Tf, double **DTf, int dummy2, double *rho, double **Drhof)
: heatloss source term.

tabular.c: module containing look-up-table-related functions
TAB ***tab1**(createTab ct)
: create 1D tabular structure.
TAB ***tab2**(createTab ct)
: create 2D tabular structure.
TAB ***tab3**(createTab ct)
: create 3D tabular structure.
void **free_tab**(TAB *tab)
: memory freeing of tabular structure.

tabular.h
: header file for look-up table declarations.

void **viscosity**(GRD *g, double dt, double ****Fx, double ****Fy, double ****Fz)
: computation of viscous forces/flux.

void **viscosityCoeff**(GRD *g, double ***vis)
: computation of dynamic viscosity coefficient.

User-related functions

User.h
: header file for user-defined/controlled macro/type declarations.

void **APdiffusionCoeffUser**(GRD *g, double ***APdiff)
: user-defined ambipolar diffusion coefficient.

void **analysisDataUser**(GRD **gc)
: user-defined data analysis.

void **bcbUser**(GRD *g, flag_t boundary)
: user-defined BC for the magnetic field.

void **bceUser**(GRD *g, flag_t boundary)
: user-defined BC for the total energy density.

void **bcetUser**(GRD *g, flag_t boundary)

: user-defined BC for the thermal energy density (dual energy mode).

void **bcmUser**(GRD *g, flag_t boundary)
: user-defined BC for the momentum.

void **bcrhoUser**(GRD *g, flag_t boundary)
: user-defined BC for the mass density.

void **bcrhoXUser**(GRD *g, flag_t boundary)
: not used.

void **bctrUser**(GRD *g, flag_t boundary)
: user-defined BC for tracers.

void **bcvUser**(GRD *g, flag_t boundary)
: user-defined BC for the fields component.

flag_t **checkDomainUser**(flag_t mode, double xl, double xu, double yl, double yu, double zl, double zu)
: checks user-defined regions for mesh refinement.

void **conductionCoeffUser**(GRD *g, double ***cond, double ***cond_perp)
: user-defined thermal conduction coefficients.

void **configUser**(GRD **gc)
: user-defined IC.

createTab **createTabUser**(flag_t taskUser)
: user-defined specification of sample data for look-up tables.

void **diffusionCoeffUser**(GRD *g, double ***diff)
: user-defined magnetic diffusion coefficient.

void **eosUser**(GRD *g, flag_t variable, double ***v)
: user-defined analytic EOS.

void **forceUser**(GRD *g, double ***fx, double ***fy, double ***fz)
: user-defined external specific force.

void **modifyConfigUser**(GRD **gc)
: user-specific modification of model data in a restart.

void **modifyFluxUser**(GRD *g, double dt, double ****Fx, double ****Fy, double ****Fz)
: user-specific modification of numerical fluxes.

double **sourceHeatingUser**(double T, double rho, flag_t *deriv, double *dfh)
: used-specific heating function.

double **sourceCoolingUser**(double T, double rho, flag_t *deriv, double *dfc)
: used-specific cooling function.

void **viscosityCoeffUser**(GRD *g, double ***vis)
: user-defined dynamic viscosity coefficient.

Basic functions in module util.c

double ***Array**(int n)
: allocation of a 1D array of type double.

int ***Arrayi**(int n)
: allocation of a 1D array of type int.

double ****Array2**(int m, int n)
: allocation of a 2D array of type double.

int **__Array2i__(int m, int n)
: allocation of a 2D array of type int.

uflag_t **__Array2B__(int m, int n)
: allocation of a 2D array of type unsigned char (uflag_t).

GRD ***__Array2G__(int m, int n)
: allocation of a 2D array of type GRD*.

double ***__Array3__(int l, int m, int n)
: allocation of a 3D array of type double.

flag_t ***__Array3C__(int l, int m, int n)
: allocation of a 3D array of type signed char (flag_t).

uflag_t ***__Array3B__(int l, int m, int n)
: allocation of a 3D array of type unsigned char (uflag_t).

GRD ****__Array3G__(int l, int m, int n)
: allocation of a 3D array of type GRD*.

double ****__Array4__(int k, int l, int m, int n)
: allocation of a 4D array of type double.

void **__free_Array2__**(double **array)
: memory freeing of type double 2D array.

void **__free_Array2i__**(int **array)
: memory freeing of type int 2D array.

void **__free_Array2B__**(uflag_t **array)
: memory freeing of type uflag_t 2D array.

void **__free_Array2G__**(GRD ***array)
: memory freeing of type GRD* 2D array.

void **__free_Array3__**(double ***array)
: memory freeing of type double 3D array.

void **__free_Array3C__**(flag_t ***array)
: memory freeing of type flag_t 3D array.

void **__free_Array3B__**(uflag_t ***array)
: memory freeing of type uflag_t 3D array.

void **__free_Array3G__**(GRD ****array)
: memory freeing of type GRD* 3D array.

void **__free_Array4__**(double ****array)
: memory freeing of type double 4D array.

void **__flipID__**(int lev, int *id0, int *id1, int *id2)
: flips block id at periodic/axis domain boundaries.

void **__terminate__**(const char *err_message, int err_stat)
: termination at code-controlled exception.

void **__checkpoint__**(int err_stat, const char *err_message)
: runtime checkpoint.

void **__divergence__**(GRD *g, double ***vx, double ***vy, double ***vz, double ***div, flag_t noc)
: FV divergence operator.

void **__curl__**(GRD *g, double ***vx, double ***vy, double ***vz,
double ***cx, double ***cy, double ***cz)

: FV curl operator.

Time-**I**ntegration-related functions in container utilTI.c

int **TIexprb32**(int nu, double *u0, double M, int np, double *p,
void (*f)(int, double *, double **, int, double *, double **),
double **Duf, double dt, double tol)
: exponential Rosenbrock method of order 3(2).

Thermo**D**ynamics-related functions in container utilTD.c

void **TDeos**(GRD *g, flag_t variable, double ***v)
: computation of ideal gas isothermal/polytropic/adiabatic EOS.

void **TDeosTab**(GRD *g, flag_t task, double ***v)
: computation of tabulated EOS.

void **TDgetMeanMolecularWeight**(GRD *g, double ***mw)
: computation of mean molecular weight.

IO-related functions in container utilIO.c

void **IOreadParameter**(char *fname)
: reads parameter from user interface **nirvana.in**.

void **IOrebuildMOD**(char *fname)
: rebuilds model from DATA_MOD type data file.

void **IOwriteLOG**(GRD **gc)
: writes data of DATA_LOG type into a file.

void **IOwriteMOD**(GRD **gc, flag_t mode)
: writes data of DATA_MOD type into a file.

void **IOwriteRAW**(GRD **gc)
: writes data of DATA_RAW type into a file.

Boundary-**C**onditions-related functions in container utilBC.c

void **BCb**(GRD *g, flag_t boundary)
: standard BC for the magnetic field.

void **BCe**(GRD *g, flag_t boundary)
: standard BC for the total energy density.

void **BCet**(GRD *g, flag_t boundary)
: standard BC for the thermal energy density (dual energy mode).

void **BCm**(GRD *g, flag_t boundary)
: standard BC for the momentum.

void **BCphi**(GRD *g, flag_t boundary, flag_t var)
: BC for the gravitational potential.

void **BCrho**(GRD *, flag_t boundary)

: standard BC for the mass density.

void **BCrhos**(GRD *g, flag_t boundary)
: not used.

void **BCtr**(GRD *g, flag_t boundary)
: standard BC for the tracers component.

void **BCv**(GRD *g, flag_t boundary)
: standard BC for the fields component.

Re**C**onstruction-related functions in container utilRC.c

void **RCcomputeDerivatives**(GRD *g, DER D)
: computation of derivatives of variables.

void **RCcomputeLRstates**(GRD *g, flag_t rd, DER D, double ****uL, double ****uR)
: computation of left/right-sided values at cell interfaces.

void **RCconToPrim**(GRD *g)
: conversion from conservative to primitive variables.

void **RCevalReconstruction**(GRD *g, DER D, double *x, double *y, double *z, double ****u)
: point value calculation with reconstructed cell solution.

void **RCprimToCon**(GRD *g)
: conversion from primitive to conservative variables.

Numerical-**S**olver-related functions in container utilNS.c

void **NScomputeBSQR**(GRD **gc)
: computation of $\mathbf{B}^2$.

void **NScomputeEF**(GRD *g, double dt, double ****Fx, double ****Fy, double ****Fz)
: computation of electric field for the HLLD_CT scheme.

void **NScomputeFlux**(GRD *g, double dt, DER D, double ****Fx, double ****Fy, double ****Fz)
: computation of numerical flux.

void **NScomputeInterfaceState**(GRD *g, flag_t dir, double *hz, double **uL, double **uR,
double *sL, double *sR, double **uI)
: computation of HLLD interface state.

void **NScomputeSignalSpeed**(GRD *g, flag_t rd, double ****uL, double ****uR,
double ***sm, double ***sp)
: computation of one-sided maximum signal speeds.

void **NScomputeSource**(GRD *g, double dt, DER D)
: computation of source terms.

void **NSevalFluxFunctionX**(GRD *g, double *hzI, double **uI, double **fI)
: evaluation of x-flux function.

void **NSevalFluxFunctionY**(GRD *g, double *hzI, double **uI, double **fI)
: evaluation of y-flux function.

void **NSevalFluxFunctionZ**(GRD *g, double *hzI, double **uI, double **fI)
: evaluation of z-flux function.

Multi**G**rid-method-related functions in container utilMG.c

void **MGprolong**(GRD **gc, int level)
: prolongation operator.

void **MGrestrict**(GRD **gc, int level)
: restriction operator.

void **MGsmooth**(GRD **gc, int level, COM *cSYNC, flag_t noc)
: smoother based on GS-RB.

void **MGsolveSubMesh**(GRD **gs, COM *cSYNC,
void (*MGoperator)(GRD **, int, COM *, flag_t))
: submesh solver based on SOR-RB.

void **MGVcycle**(GRD **gc, GRD **gs, COM *cSYNC, COM *cFIXUP,
void (*MGsource)(GRD *, double ***),
void (*MGoperator)(GRD **, int, COM *, flag_t))
: multigrid V-cycle.

Mesh-**O**perations-related functions in container utilMO.c

GRD ****MOcreateSubMesh**(GRD *gb)
: creation of submesh.

GRD ***MOclusterMesh**(int level)
: block clustering algorithm – superblock generator.

void **MOdeclareMesh**(GRD *g, int *dim, flag_t grid_type)
: mesh declaration – memory allocations.

void **MOdefineMesh**(GRD *g, int *dim, int *pos, flag_t ngc, uflag_t from_heap)
: mesh definition – attributes/coordinates/metrics.

void **MOdumpMesh**(GRD **gc, int level, flag_t *comp)
: dumps variables from the clustered mesh to generic blocks.

void **MOkillMesh**(GRD **gc, int level)
: kills the clustered mesh.

Block-**R**egistry-related functions in container utilBR.c

GRD ***BRaddBlock**(int *id, int level, uflag_t from_heap)
: adds a grid block to the grid hierarchy.

void **BRassignBlockNeighbors**(GRD *g)
: assigns a block pointers to its grid neighbors.

void **BRfinalizeUpdate**(COM *co)
: finalizes a block registry update.

GRD ***BRg**(int lev, int id0, int id1, int id2)
: block pointer request.

int ***BRgetProcessNeighbors**(GRD *g, int *nproc)
: returns a list of process ranks in a block's spatial neighborhood.

void **BRinitUpdate**(COM *co)
: initializes a block registry update.

void **BRkill**(void)
: kills the block registry.

void **BRkillBlockList**(BLT *bl)
: kills a block list.

int ***BRpl**(int *nproc, uflag_t *b_local, int lev, int id0, int id1, int id2)
: returns a list containing a block's process ranks (block sharing technique).

int **BRpr**(int lev, int id0, int id1, int id2)
: returns a block's primary process rank.

void **BRremoveBlock**(GRD *g)
: removes a grid block from the grid hierarchy.

Ghost-**C**ell-assignment-related functions in container utilGC.c

void **GCbc**(COM *co, GRD *gl)
: superblock ghost cell assignment at physical boundaries.

void **GCcheck**(COM *co, GRD *gl, flag_t grid_type)
: checks for local ghost cells.

void **GCcp**(COM *co, GRD *g, int xs, int ys, int zs, flag_t xl, flag_t yl, flag_t zl,
GRD *gb, flag_t grid_type, int xss, int yss, int zss)
: superblock ghost cell assignment from local grids.

void **GCfillCommunicator**(COM *co, int level, flag_t grid_type)
: fills communicator data list for ghost cell remote copy.

void **GCinterpolate**(COM *co, GRD *gl)
: superblock ghost cell interpolation from coarser refinement level.

void **GCrcheck**(COM *co, int level, DLT *dl)
: checks for remote ghost cells.

void **GCrcp**(COM *co, GRD *g, int xs, int ys, int zs, flag_t xl, flag_t yl, flag_t zl,
DLT *d, flag_t grid_type, int xss, int yss, int zss)
: superblock ghost cell assignment from remote grids.

Mesh-**R**efinement-related functions in container utilMR.c

flag_t **MRcheckRefinementCriteria**(GRD *g, int px, int py, int pz)
: checks criteria for mesh refinement.

void **MRinitBlock**(GRD *g)
: child block initialisation via prolongation techniques.

void **MRrefineMesh**(GRD **gc, int level, flag_t mode)
: mesh refinement algorithm.

Mesh-**S**yncronization-related functions in container utilMS.c

void **MSfixupMesh**(GRD **gc, int level, COM *co)
: mesh fixup at refinement interfaces.

void **MSrestrictMesh**(GRD **gc, int level, flag_t *comp, flag_t var)

: performs a local mesh restriction operation.

void **MSsaveFlux**(GRD *g, double ****Fx, double ****Fy, double ****Fz, COM *cFIXUP)
: numerical flux storage for mesh fixup.

void **MSsyncMesh**(GRD **gc, int level, COM *co, flag_t task)
: mesh syncronization.

COmunicator-related functions in container utilCO.c

DLT ***COaddToList**(DLT *dl, GRD *gs, int id0, int id1, int id2, int it0, int it1, int it2)
: adds mesh fixup data to a data list.

COM ***COgetClassCommunicator**(GRD **gc, BLT *bl, flag_t type)
: declaration of a class communicator.

COM ***COgetCommunicator**(COM *cc, flag_t *comp, flag_t special)
: definition of a communicator.

void **COkillCommunicator**(COM *co)
: kills a communicator.

Parallel-**I**nterface-related functions in container utilPI.c

int ***PIaddProcess**(int *pl, int *nproc, int lev, int id0, int id1, int id2)
: adds processes to a block's process environment list.

int ***PIallocProcessEnvironment**(GRD *g, int *nproc)
: allocates the process environment of a block's spatial neighborhood.

void **PIcomputeSFCMap**(ELT *e)
: computes Hilbert curve map of the base level block distribution.

DLT ***PIfinalizeTransmission**(COM *co, int level)
: finalizes data transmission.

int **PIgetSFCIndex**(int px, int py, int pz)
: computes Hilbert curve index.

void **PIinitTransmission**(COM *co, int level)
: initializes data transmission.

void **PIkillDataList**(DLT *dl)
: kills a data list.

void **PIrepartitionMesh**(void)
: mesh repartioning algorithm.

void **PItransmitInterfaceData**(DLT **dl)
: transmission of interface field data.

OLT ***PIupdateListRP**(OLT *ol, GRD *g, flag_t dir, double *dload, int ip)
: updates the sender list in the mesh repartioning process.

4 List of CAIVS modules/functions

caivs.c
: module containing CAIVS main function
    int **main**(int argc, char *argv[])
    : main function.

caivs.h
: header file containing basic macro/type/function declarations.

caivs.in
: user interface for main parameter specification.

flag_t **assignBlockNeighbors**(GRD *gb, GRD **go)
: assign a block pointer to its spatial neighbors.

GRD ****importMOD**(char *fname)
: imports NIRVANA data of DATA_MOD type.

GRD ****importRAW**(char *fname)
: imports NIRVANA data of DATA_RAW type.

void **readParameter**(void)
: reads parameters from module **caivs.in**.

void **defineMesh**(GRD *g, int *dim)
: mesh definition.

void **killMesh**(GRD **gm)
: kills the grid hierarchy.

util2.c
: container for basic functions

void **exportIDL**(GRD **go, char *out_file)
: exports data as formatted files accessible through the IDL procedure **readIDL.pro**.

void **exportSILO**(GRD **go, char *out_file)
: exports data in SILO format.

utilMT.c
: container for Meshdata-Transformation-related functions
    void **MTclusterMesh**(GRD **go)
    : clustering of output mesh.
    void **MTsyncMesh**(GRD **go)
    : syncronizes output mesh at refinement interfaces (AMR).
    void **MTmapToUnigrid**(GRD **go)
    : maps an adaptive mesh to a unigrid.
    void **MTdblToflt**(GRD *g, double ***U, float ***Uf)
    : copies an array of datatype double into an array of datatype float.
    void **MTmapToCenter**(GRD **go)
    : node-to-cell interpolation.
    void **MTcurveToQuad**(GRD *g, double ***x, double ***y, double ***z, flag_t var)
    : transformation of curvilinear coordinates/variables to quad coordinates/variables.

void **writeOutputMesh**(GRD **gc, GRD **go, int rank)
: writes output mesh.